IT University of Copenhagen

Doctoral Thesis

Towards Improved Marketing Mix Decisions through Deep Learning and Human-AI Collaboration

Author
Mathias Kristian Kyndlo Löwe

Principal Supervisor
Dr. Sebastian Risi

Company Supervisor
Dr. Per Lunnemann

A thesis submitted in partial fulfillment of the requirements for the conferral of Doctor of Philosophy

in the

Robotics, Evolution, & Art Lab
Digital Design Department

An industrial PhD project conducted in collaboration with Blackwood Seven

IT UNIVERSITY OF COPENHAGEN

BLACK WOOD SE7EN

November, 2021

IT UNIVERSITY OF COPENHAGEN

Abstract

Digital Design Department

Doctor of Philosophy

Towards Improved Marketing Mix Decisions through Deep Learning and Human-AI Collaboration

by Mathias Kristian Kyndlo Löwe

With the recent boom in *machine learning* (ML), people are interacting with a growing number of ML-advised products in many aspects of their everyday life. However, the underlying logic in the decision-making process of ML systems generally lacks transparency. This becomes an issue when the ultimate decision is made by the end user and the output of the ML tool disagrees with the user's belief system. Such disagreements, combined with incomprehensible ML reasoning, cause difficulties in *human-AI interaction* (HAII). One domain in which such predicaments often occur is the choice coordination of *mixed-marketing plan*s (MMPs). ML methods like *probabilistic graphical model*s (PGMs) can be used to optimize a MMP's effect on the *key performance indicator* (KPI) of a company, but the optimized MMPs appear synthetic to marketing employees, who are therefore reluctant to adopt these recommendations. This thesis aims to mitigate such gaps between ML and user in the context of marketing planning and HAII. The contributions of this thesis are: a) An approach to combine PGMs and *neural network*s (NNs) through approximation. This combination enables rapid feedback from the AI-recommender system to the user. By extension, this affords additional human-AI feedback loops before reaching user fatigue. In turn, this provides the users with a better understanding of the AI behavior. b) The introduction of the NN-based game *iNNk* to be used as a case study for HAII. This study gives empirical insights on how humans and NNs perceive and classify the same data differently. These differences cause game-breaking player strategies to emerge. c) A simple method to ease identified limitations of the NN in iNNk. These limitations were discovered by observing HAII with non-expert users playing the game. d) A stepping stone toward ultimately combining the previous insights to augment the AI-recommender system used for marketing planning. This stepping stone is a method to modify the HAII flow in marketing planning by enforcing diversity. This diversification opens new possibilities, as it allows marketing employees to develop an improved mental model of the recommender system. In addition, it allows for better adaptation of user preferences in complex, high-dimensional optimization tasks.

Resumé

Gennem de seneste år er der sket et boom inden for computerbaseret mønstergenkendelse – også kendt som maskinlæring. Som et resultat heraf forekommer der i stigende grad interaktion mellem mennesker og maskinlæringalgoritmer. En generel udfordring ved disse algoritmer er dog, den menneskelige bruger oplever en mangel på gennemsigtighed i algoritmens bagvedliggende logik. Da maskinlæringalgoritmen skal rådgive en eller flere brugere i et felt med høj kompleksitet, kan den manglende transparens udgøre et problem, såfremt brugerens egen erfaring modstrider algoritmens rådgivning. I det tilfælde, at brugerens egen erfaring modstrider den givne rådgivning, bliver den manglende transparens i maskinlæringsalgoritmens beslutningsproces et problem. Et felt, hvori sådanne uoverensstemmelser mellem menneske og maskinlæring forekommer, er tilrettelæggelse af marketingsplaner. Maskinlæringmetoder, såsom *probabilistic graphical models* (PGM)[1], kan benyttes til at optimere denne tilrettelægning, således at marketingsplanen påvirker virksomhedens nøgletal mest muligt. Disse automatisk tilrettelagte marketingsplaner forekommer ofte kunstige over for de ansatte, der står for markedsføringen. Brugerne af maskinlæringalgoritmen er derfor tilbageholdende ved at benytte sig af de autogenererede planer. Denne afhandling søger at adressere sådanne forståelseskløfter mellem menneske og maskinlæring i en marketingkontekst. Afhandlingens bidrag spænder over fire dele: a) En metode til at kombinere PGM med neurale netværk (NN) gennem en approksimation. Denne approksimation muliggør hurtig feedback fra maskinlæring til menneske. I forlængelse heraf muliggøres en mere effektiv modning af brugerens mentale model af maskinlæringsalgoritmen, da flere menneske-maskine-menneske iterationer kan foretages inden for den samme tidsramme. b) Introduktion af spillet iNNk, der bygger på brug af et NN. Dette spil benyttes som et casestudie af menneske-maskinlæring interaktion. Dette studie frembringer empirisk indsigt i, hvordan mennesker opfatter og klassificerer de samme data anderledes end et NN. Disse forskelle foranlediger spillerne til at udvikle strategier, der ophæver spillets balance. c) En simpel metode til at afhjælpe identificerede begrænsninger i iNNks NN. Disse begrænsninger bliver afdækket ved at observere ikke-eksperter interagere med det NN. d) Et springbræt mod målet med at kombinere de nævnte erfaringer til at forbedre rådgivningssystemer baseret på maskinlæring i en marketingkontekst. Dette springbræt er en metode, der modificerer interaktionsprocessen ved at gennemtvinge diversificering. Dette åbner nye døre, da det giver marketingsansatte muligheden for at udvikle bedre mentale modeller ud fra disse rådgivningssystemer. Ydermere tillader metoden bedre mulighed for at imødekomme de ansattes præferencer mht. udformning af marketingsplaner.

[1] Der findes endnu ikke en veletableret dansk term for dette koncept, derfor benyttes den engelske.

Acknowledgements

First and foremost, I would like to thank my principal supervisor Sebastian Risi for encouraging me to pursue a PhD. Your inspiration and guidance have been rock solid from beginning to end. Without your support in bad times, I would not have been able to finish the project.

A special thanks go to Per Lunnemann, my company supervisor. Coming from two different worlds, our collaboration experienced a few hiccups at the beginning of the PhD. However, quite quickly, we managed to turn our differences into a fruitful partnership. Your supervision has been vital for the university-company collaboration to function, and I am forever grateful for your tutelage.

I would also like to thank all my intermediate company supervisors who have crossed my path: Mikkel Settnes, Per Kær, Jökull Snæbjarnarson, and Michael Green.

Thank you to Innovation Fund Denmark and Blackwood Seven for funding my PhD. I am a firm believer in combining academia and industry for joint collaboration to push the frontier of science, and you have made that possible.

I want to extend my thanks to Jes Frellsen for taking time to assist me in the first paper of this work. You made a real difference to me and provided some decisive ideas for this research.

COVID-19 hindered my physical stay abroad during the PhD. Instead, I had the great pleasure of a virtual stay at Drexel University. I would like to extend my gratitude to Jichen Zhu for allowing me to join her amazing team in the Procedural Expression lab. It was a pleasure collaborating with you on our project *iNNk*. To Jen, Boyd, Rush, and Alex, I owe you a great deal of gratitude. I truly enjoyed our teamwork and joint paper writing sessions.

To my PhD therapy partner and friend Philip, thank you for always being up for a good PhD rant. I am glad to see that we have both made it through despite the odds.

I would not have arrived at this point without my family. Pernille and Stephan, thank you for your continuous encouragement and loving support throughout this process. Your repeated attempts to understand my work have always been appreciated and challenged my dissemination skills.

Lastly, to my dear Amalie, thank you for your everlasting loyalty and protection. I can always count on you.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI artificial intelligence . . . 3

AL active learning . . . 31

AR auto-regressive . . . 28

BM Bayesian model . . . 6

BW7 Blackwood Seven . . . 3

ES early stopping . . . 36

HAII human-AI interaction . . . iii

HCI human-computer interaction . . . 68

KNN *k* nearest neighbor . . . 94

KPI key performance indicator . . . iii

LR learning rate . . . 52

LSTM long short-term memory . . . 17

MAP maximum a posteriori . . . 24

MAP-Elites multi-dimensional archive of phenotypic elites . . . 93

MC Monte Carlo . . . 24

ML machine learning . . . iii

MLE maximum likelihood estimate . . . 24

MMM marketing mixture model . . . 6

MMP mixed-marketing plan . . . iii

MSE mean squared error . . . 38

MTAM multi-touch attribution model . . . 26

NLP natural language processing . . . 18

NN neural network . . . iii

To my family.

Part I

Prologue

Chapter 1

Introduction

Human decision-making is a convoluted and latent cognitive process [22], the purpose of which is to narrow down choices in a given situation [20]. If the selection is based on rational reasoning, the decision is the outcome of an intelligent deliberation to maximize the likelihood of achieving a desired goal [239].

Modern *artificial intelligence* (AI) and *machine learning* (ML) technologies can be used to replace human involvement in several decision-making processes [132]. Examples of such fully automated, data-driven decision-making processes can be found in domains like music recommendation [223], bot detection [256], and risk management [40].

However, not all domains are suited for a fully automated decision. Instead, the AI and human collaborate on the decision-making in a *human-AI interaction* (HAII) setup [15]. These hybrid structures occur in domains where the two parties complement each other to maximize speed and accuracy. Examples of such setups are crowdsourcing [146], speech transcription [89], and disease diagnosis [17, 291].

In such HAIIs, the AI and human work as a team to solve a task better than either party can alone. The human considers a recommendation from the AI before making the ultimate decision. In such human-AI teaming, it is the task of the human to recognize situations in which the AI cannot be trusted and take a different action [15].

Another example of an AI-advised human decision-making setup is marketing planning [46, 273]. In this example, one or more marketing employees receives an optimized marketing plan recommended by a ML model. Each marketing plan is a compound of many synergistic decisions for each day it covers [221]. These are decisions like where, what, and how many advertisements to run [310].

Blackwood Seven (BW7) (the company partner of this industrial PhD) is a company that has specialized in AI for marketing planning. It is a business-to-business Software as a Service company that has successfully applied *Bayesian optimization* to marketing planning. BW7 has built a Bayesian probabilistic network that is tailored to each customer's individual business. This allows BW7 to capture the different drivers of each customer's sales, e.g., pricing, media, promotion, distribution, macroeconomic factors, etc. On top of this model sits a media optimization engine

that can generate optimally profitable media plans for each customer. The profit uplift created by BW7's AI engine is estimated to be 50-300%. Even though these plans are potentially very effective, they are hard to adopt for their customers because the generated plans lack a resemblance to plans traditionally made by humans[1]. *Even the most optimal AI solutions do not have commercial value if the customer cannot relate to why the algorithm came up with them*.

Humans not being able to relate to why an AI came up with a certain solution is a symptom of a broader issue with AI systems. For instance, whenever a *neural network* (NN) classifies an image, a human is able to visually verify the output of the NN, in this case whether the NN's prediction is correct or not. However, understanding *why* the NN came up with that particular classification is far more difficult. If two humans disagree, the two parties are able to engage in a discussion on their differing perspectives to reach a common understanding. Such dialogues are not possible when one of the parties is an AI, and in effect, this can cause the human to distrust the AI.

Cognitive psychology research has found that whenever humans interact and use a system, they do so with a conceptualization of that system's capabilities [208]. This conceptualized view is an ever-evolving *mental model* that the user has of the system. This mental model is iteratively updated through the course of numerous interactions with the system. These models provide the user with explanations for why the system, in this case an AI, makes certain decisions [162], but it also tells the user when not to trust the AI [15], i.e., the user learns the error boundary of the AI [14]. Notably, Kulesza et al. [162] found that users with a more accurate mental model of an AI system are more likely to perceive the HAII satisfactorily. Furthermore, accurate mental models lead to AI output more closely aligned with user intentions [162].

If the marketing employees using BW7's platform are to adopt and appreciate the ML-generated marketing plans, improvements to the mental models of the employees are required.

However, in addition to the marketing employees' mental model of BW7's ML tool, these employees also have their own sophisticated mental models on the marketing landscape and what characterizes a good spending pattern. After all, they are educated employees with years of experience. The two mental models might give rise to conflicts in one or more ways.

A way to minimize these discrepancies is for the ML tool to consider and incorporate the preferences of these employees. However, formalizing user preferences and incorporating these into AI models is tricky. Such preferences need to be condensed into an objective function suitable for an AI and whose optimum is the solution of interest [252]. Even just verbally describing preferences can be difficult – especially if these preferences stem from intuition. One example is putting into words why one prefers a particular painting over another [302].

[1]Source: *Michael Green*, former Chief AI Officer at Blackwood Seven

Given said challenges with HAII and marketing planning, one might question whether an AI-advised decision setup is even the right approach. For example, web analytics tools like Google Analytics[2] neatly report data collected from marketing campaigns running on various websites. These detailed insights involve tracking who is responding to the ads, where they came from, what parts of the website have the users' attention, etc. This is with the goal of adding transparency to which advertisements seem to work well and which are ill spent. However, the collected data consist of over 400 metrics [191]. This dimensionality makes the data overwhelming. Manually determining an advertisement's effect based on all information captured in such a dataset is infeasible. As a result, one is left with one of two options: either neglect the complexity of the domain and utilize only simplified, aggregated numbers that are "easy to understand," or abandon such tools and rely on gut feelings and intuition for planning marketing campaigns [191].

The dimensionality of the collected dataset is not the only reason for being hesitant when considering whether to use such web analytic tools as a basis for marketing planning. These tools only cover online advertisements, whereas more traditional marketing approaches such as newspapers, TV, and radio are advertising formats left out. Yet these still have a significant impact – especially on establishing brand awareness [244].

Additionally, it has been shown that a silo-based marketing strategy focusing only on a single platform can yield diminishing returns. Because of this, one should have a compound marketing strategy that enables synergies across advertising platforms, new as well as old. Such a diversification of advertising formats can influence consumer behavior throughout all stages of the purchasing process [79, 126].

Optimizing the spending of advertisements on traditional marketing formats, however, is also challenging. One of the reasons for this challenge is imprecision. Audience targeting is imprecise because of the scattergun effect of traditional marketing formats. For example, all viewers of a specific TV channel are exposed to the same advertisement, regardless of its relevance to the individual viewer. In contrast, most online-based marketing cherry-picks the users exposed to a specific advertisement by comparing each user's profile with the advertisement's target-profile.

Another imprecision of traditional marketing lies in the dataset collected for subsequent performance evaluation. Such datasets contain a significant amount of noise from various sources, such as stochastic consumer, competitor, and macroeconomic trends. The ever-changing world makes it difficult to make intelligent decisions based on such a noisy and sparse dataset.

For these reasons, a data-efficient ML tool like that of BW7 is indeed justified. Their ML engine can handle all marketing formats as well as the noisy, sparse dataset from traditional marketing formats. However, for their customers to accept the generated marketing plans, the HAII with this AI-advised system calls for improvements. Such improvements should enhance the degree to which preferences and

[2] analytics.google.com

intuition of an arbitrary marketing employee can be addressed in such a setup. In effect, this will reduce the distance between human and AI and, in turn, increase the effectiveness with which the customers can work with, accept, and understand the AI-driven media planning.

To alleviate this gap, this thesis presents a method for combining *Bayesian model*s (BMs) (namely, *probabilistic graphical model*s (PGMs)) and NNs [175]. The purpose of the approach is to increase the speed of risk-minimized predictions with a PGM. Since many real-world PGMs (like those created by BW7) consist of several recurrent components and have no closed-form solution, making risk-minimized predictions can be computationally heavy. The experiments show that for such predictions, even a simple, synthetic Bayesian regression model scales significantly worse than the proposed method as the number of predictors in the model increases. Moreover, the proposed approximation method using a NN yields only a negligible loss in accuracy. This highlights the general applicability of the method introduced.

In addition to these insights, this thesis tailors this method to a Bayesian *marketing mixture model* (MMM) [174]. This Bayesian MMM is a PGM introduced as a real-world case study based on the AI-infused system created by BW7. A key finding is that the approach can generate optimized *mixed-marketing plan*s (MMPs) more than a full order of magnitude faster than the traditional method. In effect, by using this method, the customers of BW7 obtain feedback from the AI-system much faster than previously possible.

Getting rapid feedback from the AI is important, but the generated MMPs are effectively exhibiting the same patterns as before. Hence, the method does not address the issue with users not being able to relate to these solutions. For that to be rectified, the HAII will have to be improved by other means. Historically, games have often been the driver and testbed for novel AI developments [122, 125, 144] and have an inherent focus on end-user experience [188]. This thesis explores what NN-based games can tell about designing HAII by evaluating such games on a set of preexisting guidelines for HAII [319]. An insight of this work is how player-AI interaction (HAII in games) is an approach that can expand the discussion around productivity-based AI applications, like that of BW7. One of the interesting findings is that failures and imperfections of the AI can be used positively by encouraging players to fix and improve them.

Based on these findings, the game *iNNk* was developed to further study how teams of humans interact and perceive the capabilities of a NN. This game was developed in collaboration with *user experience* (UX) researchers at Drexel University. The game provided some key insights into the players' mental models of the NN's capabilities as play progressed in a NN-player-NN feedback-loop. The iterative feedback between players and NN increased their understanding of the capabilities and limitations of the NN. The research led to the discovery that the identified limitations of the NN can effectively be mitigated using simple ML techniques and just a few samples [176].

In essence, the limitations of the NN identified by the players were a result of humans and NN perceiving the same data differently. This disparity is comparable to the issue faced by BW7, with customers not being able to relate to the optimized MMP as they see the marketing landscape differently.

With the insights from above in mind, an improved human-machine collaboration setup for marketing planning is presented. This setup is a stepping-stone to combining the power of AI with human intuition to create MMPs that are both effective and relatable. The setup uncovers the existence of more than one good solution candidate in the high-dimensional optimization problem of marketing planning. Disclosing a diversified collection of good solutions to a marketing employee increases the likelihood of accommodating the employee's preferences.

In summary, the contribution of this thesis is threefold: 1) Increasing the speed at which marketing employees can acquire feedback from a marketing model. This, in turn, facilitates further experimentation with the model as optimizations become inexpensive. 2) Examining how teams of users interact and engage with an AI – in this case, a NN. This includes the mental model users develop of the NN and mitigating flaws in the NN identified using the users' mental models. 3) Enabling improved human-AI decision collaboration. This is achieved by exploration of the search space at which the AI operate. The exploration allows the users to pick the most appealing option. This is in contrast to the current situation at BW7, in which the users are only given the single best solution in the search space. The ability to choose among a set of good candidate solutions allows for better accommodation of arbitrary user inclinations and preferences based on intuition.

1.1 Thesis Structure

This work is structured in five parts. The first part provides the contextual frame and motivation for this thesis and an overview of its composition.

The second part encapsulates the first two research projects conducted in this work. The topic of this part is combining NNs and BMs to get the best of both worlds. This part combines the two approaches in order to speed up the generation of optimized marketing plans. This combination allows for an improved feedback loop from AI to the user and, by extension, increases the number of feasible experiments to be executed using the model in a marketing setup.

The third part has human-AI interaction at its core. The part highlights challenges with human-AI interaction and how easily NNs can be misled. The part introduces an online multiplayer game with a NN at its core. The game was created as part of this work to function as a testbed for human-AI interaction and how the two parties perceive the same data.

Part four combines and utilizes the insights from parts two and three. The methods in this part are based on an exploration for diversity rather than directly encoding user preferences in the AI. This approach allows one to choose the optimized

marketing plan that best suits one's preferences, instead of being stuck with a single optimized solution, as is currently the method of BW7.

The last part, part five, summarizes the key findings in this work and potential future directions for the research. The thesis ends with a few takeaway messages and a conclusion based on the achieved experimental results.

This thesis utilizes a wide range of existing ML methods and tools. Some of these tools are only used in a limited number of chapters, whereas others are used repeatedly. Methods reused at a later stage of this work are not reintroduced. For clarity, Table 1.1 is an index of the methods applied, which chapter initially presents them, and which chapters revisit them later.

TABLE 1.1: **An index of existing methods and concepts applied in this work.** The methods are only introduced once but are reused and revisited several times. This table provides an overview of these methods, when they are first introduced, and which chapters subsequently utilize these.

Method	**Introduced in chapter**	**Also applied in chapter(s)**
Bayesian models	2	3, 4, 7, 8
Bayes' Risk	2	3, 4
Marketing plans and marketing modeling	2	4, 7, 8
Optimization of marketing plans	2	4, 7, 8
Neural networks	2	3, 4, 5, 6, 7, 8
Active learning	3	-
Mental models	5	6, 7, 8
Neural networks in games	5	6
Neural networks and adversarial attacks	6	8
Ensembles	6	-
Transfer learning	6	-
Exploring for quality diversity	7	8

1.2 Notation

Notation: x is a scalar value. $\mathbf{x}$ is a column vector. $\mathbf{x}_i$ is the scalar at row i of the column vector $\mathbf{x}$. $\mathbf{X}$ is a matrix. $\mathbf{X}_{i,j}$ is the scalar at the ith row and jth column of the $\mathbf{X}$ matrix. Declarations of vectors or matrices are sometimes accompanied by subscript uppercase Roman letters to denote the dimensionality of the declared variable, e.g., $\mathbf{X}_{I\times J}$ is a matrix with I rows and J columns. This is only used for clarity in certain

contexts. Uppercase Roman letters like X are random variables. $\tilde{\cdot}$ is a prediction. $\bar{\cdot}$ is the mean. $\top$ denotes the transpose of a vector or matrix such that $\mathbf{x} = [i, j]^\top$ is the column vector with the scalars $\mathbf{x}_1 = i$ and $\mathbf{x}_2 = j$. Similarly, to represent repeated operations, the following notation is used: $\mathbf{x} = [i]_{i=1,\dots,I}^\top$. This results in a column vector of dimension I with the scalars $\mathbb{N}_1^+$. Where $\mathbb{N}_1^+$ is all the natural, positive numbers. Likewise, $\mathbb{R}_N$ is the N-dimensional real space.

1.3 Published Work

In this thesis, four papers and one extended abstract are included, as listed below. Four of these were published at a conference, and the remaining journal paper is still under review.

Paper 1 Mathias Löwe, Per Lunnemann, and Sebastian Risi. "Rapid Risk Minimization with Bayesian Models Through Deep Learning Approximation". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9534258 [175]

Paper 2 Mathias Löwe, Per Lunnemann Hansen, and Sebastian Risi. "Combining Deep Learning and Bayesian Models for Swift Generation of Mix-Marketing Plans". Submitted for peer review at the Elsevier Journal Intelligent Systems with Applications. 2021 [174]

Paper 3 Jichen Zhu, Jennifer Villareale, Nithesh Javvaji, S. Risi, Mathias Löwe, Rush Weigelt, and C. Harteveld. "Player-AI Interaction: What Neural Network Games Reveal About AI as Play". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021) [319]

Paper 4 Mathias Löwe, Jennifer Villareale, Evan Freed, Aleksanteri Sladek, Jichen Zhu, and Sebastian Risi. "Dealing with Adversarial Player Strategies in the Neural Network Game INNk through Ensemble Learning". In: *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*. FDG'21. Montreal, QC, Canada: Association for Computing Machinery, 2021. ISBN: 9781450384223. DOI: 10.1145/3472538.3472540. URL: https://doi.org/10.1145/3472538.3472540[3] [176]

Extended abstract Jennifer Villareale, Ana V. Acosta-Ruiz, Samuel Adam Arcaro, Thomas Fox, Evan Freed, Robert C. Gray, Mathias Löwe, Panote Nuchprayoon, Aleksanteri Sladek, Rush Weigelt, Yifu Li, Sebastian Risi, and Jichen Zhu. "INNk: A Multi-Player Game to Deceive a Neural Network". In: *Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*. New York, NY, USA:

[3] *Winner of a best paper award at FDG'21.*

Association for Computing Machinery, 2020, pp. 33–37. ISBN: 9781450375870. URL: https://doi.org/10.1145/3383668.3419858 [287]

Part II

Accelerating the Generation of Optimized Mixed-Marketing Plans

Chapter 2

Background

This chapter describes various methods and theories applied in this work. While some of the concepts presented here are described in detail, this chapter is not an extensive introduction to ML, NNs, Bayes' risk, or MMMs. The scope of this chapter is to provide a short introduction of the applied methods and furnish sources with additional information. As NNs are at the core of every experiment conducted in this work, these are introduced first in Section 2.1. This section describes a feed-forward NN, how the parameters of the network are adjusted, and several different layer types. The section ends with a discussion on the capabilities and limitations of this method. Next, section 2.2 is a brief introduction of BMs along with some of their (dis)advantages. After that follows an introduction of Bayes risk in Section 2.3. This section introduces the concept of making predictions with a model that minimizes the expected risk. Finally, this chapter concludes with the introduction of Bayesian MMMs and how to use these for optimizing MMPs. Chapters 3 and 4 both build extensively on the theory as presented in Sections 2.2 and 2.3. Further, to fully grasp the domain at hand in Chapter 4, an understanding of the concepts of MMMs as introduced in Section 2.4 is required.

2.1 Neural Networks

(Artificial) NNs are a ML technique notably different from that of BMs and PGMs. They consist of a network of units.

2.1.1 Feed-Forward Neural Networks

Feed-forward NNs are widely used as a network architecture. In such an architecture, information is propagated through a set of layers L. Here, each layer $l \in L$ can be considered a function $g^{(l)}$. The output of the whole network is then the composite function $g(\mathbf{x}) = (g^{(3)} \circ g^{(2)} \circ g^{(1)})(\mathbf{x})$ for a network with three layers and some observation $\mathbf{x}$. The "correct" output of each intermediate function, and, by extension, the layer, is unknown and therefore named a hidden layer. Only the output of the last layer $g^{(3)}$ can be directly compared to $g^*(\mathbf{x})$, where g^* is the underlying function to be approximated by the NN [99]. Figure 2.1 visualizes an example of one such NN architecture. The example illustrated on this figure contains an input layer, two hid-

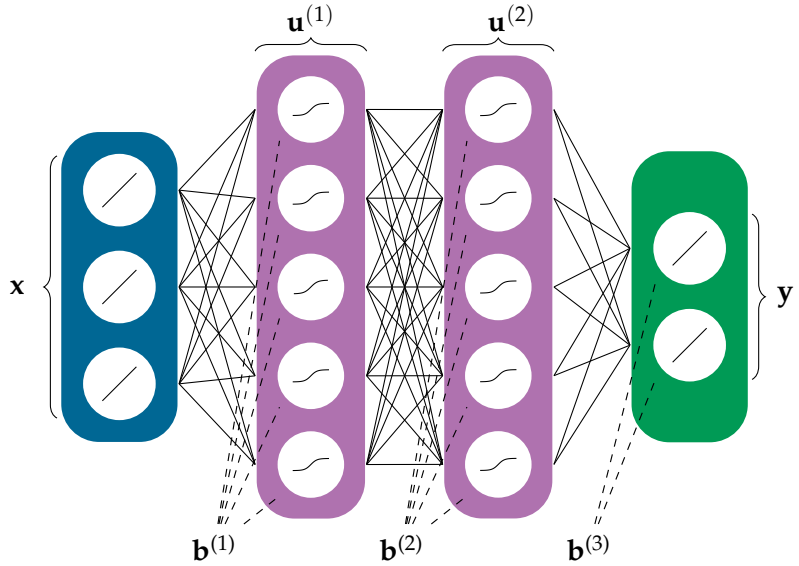


FIGURE 2.1: **An illustration of a simplistic feed-forward NN.** The plates are colored to reflect the different types of layers. The input, hidden, and output layers are all densely connected in this setup. Being densely connected means all the units (white circles) in a layer are connected to any other units in the preceding layer – except for the input layer. However, units are isolated from all other units within the same layer, as no intra-layer connections exist. Each connection between two units has an associated weight and is indicated by a black line. If this line connects node i and j, then its weight is denoted $\mathbf{W}_{i,j}$. Further, each layer is affiliated with a curly bracket and a variable. These variables are vectors that highlight the output of each layer as data flows through the model from input to output in accordance with the composite function $g(\mathbf{x})$. Each unit i in a trainable layer l also has an associated bias scalar $\mathbf{b}_i^{(l)}$ indicated by dashed line. The use of a bias is inspired by linear transformations and is required for fitting datasets not passing through *origo*. Each unit in the network is indicated by a white circle encapsulating a function. This is the activation function related to each unit. In this case, the input and output layers use a simple identity function, whereas the hidden layers apply the tanh function. This function is applied on the aggregated, weighted input to the unit, including the bias. The output of each layer is denoted as $\mathbf{x}$, $\mathbf{u}^{(1)}$, $\mathbf{u}^{(2)}$, and $\mathbf{y}$, respectively

den layers, and an output layer. The input layer is the only non-*trainable* layer, as it is not associated with any adjustable (free) parameters. The layer simply passes an observation to the weights of the first hidden layer. The hidden and output layers can be represented as functions mapping $\mathbf{x} \in \mathbb{R}_3 \rightarrow \mathbf{u}^{(1)} \in \mathbb{R}_5$, $\mathbf{u}^{(1)} \in \mathbb{R}_5 \rightarrow \mathbf{u}_{(2)} \in \mathbb{R}_5$, and $\mathbf{u}^{(2)} \in \mathbb{R}_5 \rightarrow \mathbf{y} \in \mathbb{R}_2$, respectively. Each layer consists of a set of units, sometimes also referred to as neurons, nodes, or perceptrons. Every unit is associated with an activation function. This activation function is plotted within the corresponding unit in Figure 2.1. In this example, the units of the input and output layer simply use the identity function. The hidden units, on the other hand, apply the *hyperbolic tangent function* (tanh).

The dense layers are connected to each other with weights, depicted with solid

black lines. No intra-layer connections exist between units; therefore, units within the same layer are isolated from each other. Let $\mathbf{W}^{(l)}_{I\times J}$ denote the matrix mapping layer $l-1$ and l with I and J units, respectively. The scalar $\mathbf{W}^{(l)}_{ij}$ scales the information from unit i to j and is one of the several free parameters to be inferred during training. The vector of free parameters is denoted $\boldsymbol{\phi}$. In addition to the inter-layer weighting of information, each unit i in a trainable layer l has an associated bias scalar, $\mathbf{b}^{(l)}_i$. The biases are also free parameters and thus part of $\boldsymbol{\phi}$. With the definition of the layers, their parameters, and activation functions in place, one can define the function computing the output of each layer. As an example, let us define $g^{(2)}$, i.e., the output of the second hidden layer. The output of this layer denoted $\mathbf{u}^{(2)}$ is then given by

$$\begin{aligned}\mathbf{a}^{(2)} &= \mathbf{W}^{(2)\top} g^{(1)}_{\boldsymbol{\phi}}(\mathbf{x}) + \mathbf{b}^{(2)} \\ \mathbf{u}^{(2)} &= g^{(2)}_{\boldsymbol{\phi}}(\mathbf{x}) = \tanh\left(\mathbf{a}^{(2)}\right)\end{aligned} \tag{2.1}$$

The input to the unit, $\mathbf{a}^{(2)}$, is simply an affine transformation of the output of the preceding layer. This is a simple linear model with the transformation controlled through $\boldsymbol{\phi}$. The biases $\mathbf{b}$ can be viewed as units in the network with a constant activation function of one and an associated weight to be trained [25]. This allows for less cluttered notation and the term is therefore omitted in the equations going forward. To go beyond a linear regression model, the activation function is essential. This function adds the flexibility needed to go beyond linear models [99]. In this example, a tanh function was used, but several other functions can be applied instead, such as Binary Step, Sigmoid, ReLU, and Leaky ReLU. Regardless of the activation function used, for the transformation to be useful, it is essential that the parameters $\boldsymbol{\phi}$ have a suitable value. The adjustment of these parameters is the topic of the following section.

2.1.2 Training Neural Networks

The purpose of the training procedure of the NN is to adjust $\boldsymbol{\phi}$ such that the adjustments minimize the loss. Hence, the objective is to identify $\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathcal{L}(g_{\boldsymbol{\phi}}(\mathbf{X}), g^*(\mathbf{X}))$ where $\mathbf{X}$ is the full dataset. To achieve such goal, an iterative process computes the gradient of the parameters with respect to the loss [25, 30]. Let $\boldsymbol{\phi}^t$ be the free parameters at step t, and $J(\mathbf{x}, \boldsymbol{\phi}) \coloneqq \mathcal{L}\left(g_{\boldsymbol{\phi}}(\mathbf{x}), g^*(\mathbf{x})\right)$

$$\boldsymbol{\phi}^+ = \boldsymbol{\phi} - \eta \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\phi}} \mathrm{J}(\mathbf{X}_n, \boldsymbol{\phi}) \tag{2.2}$$

where N is the size of the dataset and η is the *learning rate* [59, 99]. This is a *hyperparameter* defining the magnitude of the modification to the parameters based on the computed gradients $\nabla_{\boldsymbol{\phi}}$. Setting η correctly is of significant importance for the training process. There are several approaches one can take in setting the value of η.

The simplest way is using a fixed positive value [181], whereas other strategies use either a cyclic pattern [258] or an adaptive approach based on the gradients [72, 92].

The loss landscape is typically nonlinear, even for many trivial problems. The reason the gradients can still be used to identify a minimum lies in having a differentiable function. By recomputing the gradients after each update to $\boldsymbol{\phi}$, the optimization will carefully follow the curvature of the slope – even in case of a nonlinear loss landscape [265]. Smaller steps (i.e., learning rate) yield a trajectory that follows said curvature more closely toward a minimum, but at the cost of computational resources.

In (2.2) each parameter in $\boldsymbol{\phi}$ is adjusted through the gradients. The gradients is a vector of partial derivatives with respect to the loss, one entry for each free parameter. Let $E = \mathrm{J}(\mathbf{X}_n, \boldsymbol{\phi})$ be the error of the network's prediction. To improve the value of a single parameter of the network, e.g., the connection between unit j and k in layer l, the change to the error with respect to said weight is needed

$$\nabla_{\mathbf{W}_{jk}^{(l)}} E = \frac{\partial E}{\partial \mathbf{W}_{jk}^{(l)}} \tag{2.3}$$

Since $\partial E / \partial \mathbf{W}_{jk}^{(l)}$ is not directly available, the chain use of calculus is applied [25, 246]. In brief, the chain rule of calculus enables the calculation of (partial) derivatives of variables in composite functions. The rule states that if the composite function $h(x) = (g \circ r)(x)$ then $h'(x) = (g' \circ r)r'(x)$ [265]. Using this rule, one can propagate derivatives from the output layer back to the input layer [59]. Applying this rule to $\nabla_{\mathbf{W}_{jk}^{(l)}} E$ gives

$$\frac{\partial E}{\partial \mathbf{W}_{jk}^{(l)}} = \frac{\partial E}{\partial \mathbf{a}_k^{(l)}} \frac{\partial \mathbf{a}_k^{(l)}}{\partial \mathbf{W}_{jk}^{(l)}} \tag{2.4}$$

Further,

$$\frac{\partial \mathbf{a}_k^{(l)}}{\partial \mathbf{W}_{jk}^{(l)}} = \frac{\partial}{\partial \mathbf{W}_{jk}^{(l)}} \left[\sum_{j'} \mathbf{W}_{j'k}^{(l)} \mathbf{u}_{j'}^{(l-1)} \right] = \mathbf{u}_j^{(l-1)} \tag{2.5}$$

That is, modifying the weight between unit j and k changes the input to unit k at a rate proportional to the output of unit j. Finally, the first term on the right-hand side of (2.4) is the error with respect to the input to unit k. To obtain this partial derivative, the chain rule of calculus is applied again [25]

$$\frac{\partial E}{\partial \mathbf{a}_k^{(l)}} = \frac{\partial E}{\partial \mathbf{u}_k^{(l)}} \frac{\partial \mathbf{u}_k^{(l)}}{\partial \mathbf{a}_k^{(l)}} \tag{2.6}$$

Here, the second term is simply the partial derivative of the activation function. The first term is the error with respect to the output of unit k in layer l. If this unit is part of a hidden layer, a change to this unit will affect all subsequent layers $g^{(l')}$ with $l' > l$. Errors from $g^{(l')}$ are *propagated backwards* to the previous layers. Thus,

$\partial E/\partial \mathbf{u}_k^{(l)}$ is equivalent to the sum over all the partial derivatives of the error with respect to each unit receiving a signal from $\mathbf{u}_k^{(l)}$ in a forward pass [59]. These partial derivatives are then weighted by the connection between the two units as such:

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{u}_k^{(l)}} &= \sum_i \frac{\partial E}{\partial \mathbf{a}_i^{(l+1)}} \frac{\partial \mathbf{a}_i^{(l+1)}}{\partial \mathbf{u}_k^{(l)}} \\ &= \sum_i \frac{\partial E}{\partial \mathbf{a}_i^{(l+1)}} \mathbf{W}_{ki}^{(l+1)}\end{aligned} \tag{2.7}$$

To see the recursive nature of this formula, let $\delta_k^{(l)} := \partial E/\partial \mathbf{a}_k^{(l)}$. Then, one can apply (2.7) to (2.6) and compute this partial derivative as

$$\begin{aligned}\delta_k^{(l)} := \partial E/\partial \mathbf{a}_k^{(l)} &= \frac{\partial E}{\partial \mathbf{u}_k^{(l)}} \frac{\partial \mathbf{u}_k^{(l)}}{\partial \mathbf{a}_k^{(l)}} \\ &= \frac{\partial \mathbf{u}_k^{(l)}}{\partial \mathbf{a}_k^{(l)}} \sum_i \frac{\partial E}{\partial \mathbf{a}_i^{(l+1)}} \mathbf{W}_{ki}^{(l+1)} \\ &= h'^{(l)}\left(\mathbf{a}_k^{(l)}\right) \sum_i \delta_i^{(l+1)} \mathbf{W}_{ki}^{(l+1)}\end{aligned} \tag{2.8}$$

Here, $h'^{(l)}$ is the derivative of the activation function used in layer l. This recursive definition is applied until reaching the output layer. $\partial E/\partial \mathbf{u}_k^{(l)}$ for a unit in the output layer is simply applying the derivative of the chosen loss function evaluated using the output of the NN [246].

Applying (2.2) to update $\mathbf{W}_{jk}^{(l)}$ based on a single observation gives

$$\begin{aligned}\mathbf{W}_{jk}^{+(l)} &= \mathbf{W}_{jk}^{(l)} - \eta \nabla_{\mathbf{W}_{jk}^{(l)}} E \\ &= \mathbf{W}_{jk}^{(l)} - \eta \frac{\partial E}{\partial \mathbf{u}_k^{(l)}} \frac{\partial \mathbf{u}_k^{(l)}}{\partial \mathbf{a}_k^{(l)}} \frac{\partial \mathbf{a}_k^{(l)}}{\partial \mathbf{W}_{jk}^{(l)}} \\ &= \mathbf{W}_{jk}^{(l)} - \eta \sum_i \left(\delta_i^{(l+1)} \mathbf{W}_{ki}^{(l+1)}\right) h'^{(l)}\left(\mathbf{a}_k^{(l)}\right) \mathbf{u}_j^{(l-1)} \\ &= \mathbf{W}_{jk}^{(l)} - \eta \delta_k^{(l)} \mathbf{u}_j^{(l-1)}\end{aligned} \tag{2.9}$$

The backpropagation algorithm [245, 246] is extensively used for training networks. Backpropagation uses (2.9) in an intelligent manner by keeping track of the intermediate results in each computation. First, backpropagation performs a *forward* pass of an observation through the composite function g. During the forward pass, it records the output of each unit for later use. Next, it applies (2.4) going backwards from output to input layer.

Up until this point, the discussion on NNs has centered around simple feed-forward NNs consisting solely of dense layers. In the following sections, two more sophisticated NN layers are introduced: convolutional and *long short-term memory* (LSTM) layers.

2.1.3 Convolutional layers

In 1998, LeCun et al. [167] presented the convolutional network LeNet-5. The authors demonstrated the capabilities of LeNet-5 on a visual classification task. The network consists of a set of specialized layers which process an observation before it reaches the traditional, densely connected, feed forward layers. Two of these specialized layers are *convolutional layers*. This type of layer has proven successful due to its composition and extensive use of *weight sharing* between intra-layer connections. Sharing weights significantly reduces the number of parameters to be fitted in the convolutional layer. As a result, LeNet-5 consists of only 60 000 trainable parameters despite having 345 308 connections. Each unit in the output of a convolutional layer is not fully connected to the input as in a dense layer; instead, such a unit is only locally connected to the input layer. This local connectivity gives each output unit a limited *receptive field* of the input. The way the weight sharing is composed is through the usage of *filters*. In case of a two-dimensional input, each filter is also two-dimensional and has a set of associated weights that are free parameters. The filter is then convolved over the input data and produces an output that is referred to as a *feature map* [303]. The shared weights make the feature maps equivariant under translation – that is, shifting the input to the convolutional layer shifts the output equivalently [99]. This property significantly increased the robustness and generalization of LeNet-5. LeNet-5 is even relatively invariant to other transitions of the input, such as rotation [166, 167].

The original idea of local connections and receptive fields is strongly inspired by studies of the visual cortex of mammals from the early 1960's [130]. These receptive fields serve the function of being feature extractors. Stacking several convolutional layers makes the NN learn a hierarchical representation of features [73]. One motivational factor for replicating these receptive fields in an artificial NN is that the mere presence of a feature in an image is more important than the exact location of the feature in the observation. When trained for an image classification task, it is common for the convolutional layer to respond strongly to certain features such as edges, corners, and color conjunctions [156, 312]. Such features indicate either separation or connectivity. To detect these features, the same pattern is searched for throughout the entire input. Using the same weights for identifying these features eases the optimization task and improves generalization. In effect, convolutional layers assist the NN in learning a higher-level representation of the observations [171].

2.1.4 Recurrent layers

In contrast to traditional, dense layers as introduced in 2.1.1 and convolutional layers from 2.1.3, recurrent layers in NNs use persistent states. These states allow the layer to store and reuse information across observations. Persisting information allows for contextualized processing of subsequent data. This is of significant importance for sequence-to-sequence modeling such as *natural language processing* (NLP)

applications [106]. In such setups, the context is crucial for understanding the inherent meaning of natural language. Further, this design allows for processing data of arbitrary length, as the layer processes a potentially infinite stream of data in a sequential manner – each time updating its persistent (hidden) state.

Taking the most simplistic vanilla recurrent layer, it computes three vectors given an observation $\mathbf{x}^{(t)}$ at timestep t [74]

$$\begin{aligned}\mathbf{a}^{(t)} &= \mathbf{b}^{(a)} + \mathbf{W}^{(x)}\mathbf{x}^{(t)} + \mathbf{W}^{(h)}\mathbf{h}^{(t-1)} \\ \mathbf{h}^{(t)} &= \tanh\left(\mathbf{a}^{(t)}\right) \\ \mathbf{y}^{(t)} &= \mathbf{b}^{(y)} + \mathbf{W}^{(y)}\mathbf{h}^{(t)}\end{aligned} \tag{2.10}$$

$\mathbf{a}^{(t)}$ is a combination of the observation at t and the previous hidden layer. $\mathbf{a}^{(t)}$ is an intermediate variable used as input for the nonlinearity that results in the next hidden state of the layer, $\mathbf{h}^{(t)}$. In this example, the nonlinearity used is the tanh. $\mathbf{y}^{(t)}$ is an affine transformation of the updated hidden state. Depending on the application, this affine transformation could be processed by a nonlinearity like Softmax in case the output should parameterize a probability distribution. $\mathbf{W}^{(x)}$, $\mathbf{W}^{(h)}$, and $\mathbf{W}^{(y)}$ are all weight matrices whose elements are to be inferred. The dimensions of these matrices are defined based on the size of the input, hidden state, and output. Likewise, $\mathbf{b}^{(a)}$ and $\mathbf{b}^{(y)}$ are two bias vectors also to be inferred. Thus, the processing of information is in many ways similar to that of the vanilla dense layers as exemplified in (2.1).

One of the most prominent issues with such recurrent layers is having either exploding or vanishing gradients [18, 216]. This phenomenon happens as the backpropagation algorithm is propagating gradients backward through time. Each timestep of backpropagation involves matrix multiplication. This is an expensive procedure that results in exponentially growing or decaying gradients. Exploding gradients make the learning process highly unstable, as it will cause extreme changes to the parameters when using (2.2) to update the weights. Similarly, vanishing gradients can render the training process computationally infeasible, as the change to the parameters is nearly nonexistent [18, 216].

A more sophisticated recurrent layer architecture is *long short-term memory* (LSTM). This layer type is based on *gates*. The output of a gate is an H-dimensional vector $\mathbf{v} \in [0,1]^H$. The purpose of a gate is to repeat ($\mathbf{v}_i = 1$), remove ($\mathbf{v}_i = 0$), or reduce ($\mathbf{v}_i < 1 \wedge \mathbf{v}_i > 0$) data passing through it [169]. The LSTM layer contains three different gates, each serving a different purpose. These gates are *forget* (f), *input* (i), and

output (o). The computational steps of these gates are

$$\begin{aligned}\mathbf{f} &= \sigma\left(\mathbf{b}^{(f)} + \mathbf{W}^{(f)}\mathbf{x}^{(t)} + \mathbf{U}^{(f)}\mathbf{h}^{(t-1)}\right)\\ \mathbf{i} &= \sigma\left(\mathbf{b}^{(i)} + \mathbf{W}^{(i)}\mathbf{x}^{(t)} + \mathbf{U}^{(i)}\mathbf{h}^{(t-1)}\right)\\ \mathbf{o} &= \sigma\left(\mathbf{b}^{(o)} + \mathbf{W}^{(o)}\mathbf{x}^{(t)} + \mathbf{U}^{(o)}\mathbf{h}^{(t-1)}\right)\end{aligned} \tag{2.11}$$

As can be seen in (2.11), all gates work in the same fashion, as they all apply the same nonlinearity on an affine transformation based on an observation $\mathbf{x}^{(t)} \in \mathbb{R}_D$ and the previous output of the layer $\mathbf{h}^{(t-1)}$. The nonlinearity is chosen such that the output is within the range $[0, 1]$. However, each gate has its own set of free parameters for the affine transformation, namely $\mathbf{b}^{(\cdot)} \in \mathbb{R}_H$, $\mathbf{W}^{(\cdot)} \in \mathbb{R}_{H\times D}$, and $\mathbf{U}^{(\cdot)} \in \mathbb{R}_{H\times H}$. Therefore, each gate can be seen as an individual mini feed-forward NN taking two separate inputs [99].

The role of the input gate is to prevent polluting the *cell state* with irrelevant information [120]. The cell state (denoted $\mathbf{c}$) is another distinctive factor of LSTMs compared to vanilla recurrent layers. This state is intended for internal purposes only and serves the role of carrying an internal state of information from one timestep to the next. It is therefore noteworthy that the three gates from (2.11) do not observe this state, but instead take the previous output into consideration.

The role of the forget gate (introduced by Gers et al. [94]) is to prune obsolete information from this cell state as to make space for new information. This enables the LSTM to reset its own state and eases the learning of continuous tasks [108] Finally, the output gate controls how much of the cell state is to be revealed to "the public." The output of the LSTM layer and the cell state is computed as follows:

$$\begin{aligned}\tilde{\mathbf{c}}^{(t)} &= \tanh\left(\mathbf{b} + \mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)}\right)\\ \mathbf{c}^{(t)} &= \mathbf{f} \odot \mathbf{c}^{(t-1)} + \mathbf{i} \odot \tilde{\mathbf{c}}^{(t)}\\ \mathbf{h}^{(t)} &= \mathbf{o} \odot \tanh\left(\mathbf{c}^{(t)}\right)\end{aligned} \tag{2.12}$$

where $\odot$ is the Hadamard product, i.e., element-wise multiplication. $\tilde{\mathbf{c}}^{(t)}$ is the vector of candidate values to be written to the new cell state in the case that the input gate passes the information through it. The nonlinearity used for $\tilde{\mathbf{c}}^{(t)}$ is the tanh such that $\tilde{\mathbf{c}}^{(t)} \in [-1, 1]^H$. This nonlinearity allows for negative associations such that it is possible to decrease values in the cell state. Likewise, the updated cell state $\mathbf{c}^{(t)}$ potentially contains values >1. Therefore, this state is passed through the tanh nonlinearity when computing the final output of the layer. This aids stabilizing the behavior of the layer.

The architecture of the layer is engineered such that it addresses the aforementioned issues with exponentially exploding or vanishing gradients in vanilla recurrent layers [154]. Despite the seemingly complex nature, the LSTM layer has proven successful in a wide range of different applications [43, 107, 120, 247].

2.1.5 Universal Approximators

It has been shown that NNs can approximate any continuous function g^* given appropriate network structure [54]. For this reason, NNs are universal approximators [99, 124, 260]. Further, NNs are *model-free*, meaning that one does not carefully handcraft a mathematical model of how one believes the observed data came to be. The internal operation of a NN is heavily reliant on mathematical operations, but, it is the task of the NN to identify the relation between input and output by itself through adjustment of its free parameters. This flexibility makes for a powerful tool when the input/output relation is unknown.

Notwithstanding the immense flexibility of this tool, it requires a substantial amount of training data. It is not uncommon to have NNs with millions of free parameters trained on several gigabytes of data. In addition, due to the structural complexity, entangled nature, and large number of free parameters in NNs, they are typically hard to interpret. The general lack of interpretability of some "black-box" ML methods like NNs has led to the field of *eXplainable AI* (XAI) to address issues such as making ML decisions transparent and explainable [249]. Despite the large structure of NNs, their feed-forward architecture makes them exceptionally computationally efficient. This is a result of the utilization of optimized matrix multiplication executed on GPUs.

This concludes the introduction of NNs, their basics, and the method's strengths and weaknesses. In the following, a completely different ML approach is introduced.

2.2 Bayesian Models

Model based ML like PGMs is becoming an increasingly popular tool. PGMs are handcrafted models describing the process of *how* the observed data arose. To learn the parameters of these models, *Bayesian inference* can be applied through the use of Bayes' theorem (2.13)

$$p\left(\theta \mid \mathcal{D}\right) = \frac{p\left(\mathcal{D} \mid \theta\right) p\left(\theta\right)}{p\left(\mathcal{D}\right)} \tag{2.13}$$

Throughout this work, we will refer to such constructs as *Bayesian model*s (BMs). BMs are generative, meaning it is possible to generate synthetic data through sampling [59]. Unlike many other ML approaches, BMs are data-efficient and have integrated uncertainty handling. In order to fit BMs, the objective is to infer a probability distribution over the free parameters (θ) of the model given some observed dataset, $\mathcal{D}$. This distribution is expressed as $p\left(\theta \mid \mathcal{D}\right)$ and is referred to as the *posterior distribution*. The posterior distribution expresses our belief of θ after observing some data and maps each possible value of θ to a probability.

In (2.13), $p\left(\mathcal{D}\right)$ is called the *evidence*, as it is the probability of obtaining the observed dataset. As it only works as a normalizing constant such that $p\left(\theta \mid \mathcal{D}\right)$ sums to 1, it is typically neglected, and (2.13) is simplified to

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta) \tag{2.14}$$

$p(\theta)$ is the prior distribution. It defines our belief of θ before seeing any data and is set by the model architect after a thorough examination of the domain. Setting prior belief on θ has fostered some discussion in the scientific community, raising the concern that results should be purely objective and not influenced by one's belief system [25, 160, 181, 243]. While it is true that one should aim for objective results, "you cannot do inference without making assumptions" ([181]). Assumptions permeate all ML models; the priors in a BM are simply more explicit and thus more open for discussion.

Finally, $p(\mathcal{D} \mid \theta)$ is the likelihood function. It maps the likelihood of getting our observed dataset, given a value of θ. Altering the posterior distribution over θ changes the likelihood of the observed data.

PGMs allow one to *1)* specify prior beliefs on the parameters to be inferred, *2)* explicitly model interactions between features, and *3)* make decisions on how these features should affect the predictive distribution [91, 160]. This requires a thorough understanding of the data and domain at hand, which can be difficult, if not impossible, in some cases. On the other hand, this construct results in data-efficient models, capable of fitting even on very sparse datasets [50, 99]. The use of prior, likelihood, and posterior distribution results in a strong, consistent method for handling uncertainty throughout all aspects of the model. This is a result of the extensive use of marginalization over these distributions.

2.3 Bayes' Risk

Having an inferred posterior distribution, we often want to use it for making predictions on new data. Assume we can obtain pairs of data samples (x, y) from some unknown joint probability distribution, $p(X, Y)$, and intend to predict y. Let $\widetilde{y}$ denote the model's prediction. Basic decision theory establishes how to choose $\widetilde{y}$ such that it minimizes the expected loss $\mathbb{E}[\mathcal{L}(\widetilde{y}, y)]$, which is risk minimization [25, 59].

To do so, first, we need to quantify how "wrong" any prediction is with respect to the correct value. Throughout this paper, we are interested in regression problems and will assume a Euclidean loss function. Let $\mathcal{L}(\widetilde{y}, y) = (\widetilde{y} - y)^2$. Our goal is to choose $\widetilde{y}$ as to minimize the expected loss, hence to find $\arg\min_{\widetilde{y}} \mathbb{E}[\mathcal{L}(\widetilde{y}, y)]$. This expectation is simply a weighted average over all the possible values for Y multiplied with the conditional probability of that value for Y:

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\widetilde{y}, y) \mid X] &= \int_Y \mathcal{L}(\widetilde{y}, y)\, p(Y \mid X)\, dy \\ &= \int_Y (\widetilde{y} - y)^2\, p(Y \mid X)\, dy \end{aligned} \tag{2.15}$$

As we are interested in knowing what the optimal value of $\widetilde{y}$ is, we take the partial derivative of $\widetilde{y}$. This will inform us how our prediction affects our expected loss:

$$\begin{aligned}\frac{\partial \mathbb{E}\left[\mathcal{L}\left(\widetilde{y}, y\right) \mid X\right]}{\partial \widetilde{y}} &= 2\int_{Y}\left(\widetilde{y}-y\right)p\left(Y \mid X\right)\,dy \\ &= 2\left(\widetilde{y}-\int_{Y} yp\left(Y \mid X\right)\right)\,dy \\ &= 2\left(\widetilde{y}-\mathbb{E}\left[Y \mid X\right]\right)\end{aligned} \tag{2.16}$$

As we aim to minimize the loss, we set the partial derivative to 0, and solve for $\widetilde{y}$

$$\begin{aligned}2\left(\tilde{y}-\mathbb{E}\left[Y \mid X\right]\right) &= 0 \\ 2\tilde{y}-2\mathbb{E}\left[Y \mid X\right] &= 0 \\ 2\tilde{y} &= 2\mathbb{E}\left[Y \mid X\right] \\ \tilde{y} &= \mathbb{E}\left[Y \mid X\right]\end{aligned} \tag{2.17}$$

Hence, the optimal prediction is the conditional expectation of the underlying data distribution. An oracle predicting $\widetilde{y} = \mathbb{E}\left[Y \mid X\right]$, would still incur some error due to the stochasticity of the data generation process. This error is sometimes called Bayes' risk or Bayes' error [99, 240]. Without access to an oracle, the ground true data distribution for most interesting problems is unknown, and we cannot compute the conditional expectation directly. Instead, one can use the posterior distribution $p\left(\theta \mid \mathcal{D}\right)$ to find $\mathbb{E}\left[\widetilde{Y} \mid X\right]$ and use this as a surrogate for the ground true expected conditional. $\mathbb{E}\left[\widetilde{Y} \mid X\right]$ is given by

$$\mathbb{E}\left[\tilde{Y} \mid X\right] = \int_{\widetilde{Y}} \widetilde{y}p\left(\widetilde{Y} \mid X\right)d\widetilde{y} \tag{2.18}$$

Identifying this expectation thus requires *marginalization* over the full posterior distribution

$$p\left(\widetilde{Y} \mid \mathcal{D}, X\right) = \int_{\Theta} p\left(\widetilde{Y} \mid \theta, X\right)p\left(\theta \mid \mathcal{D}\right) \tag{2.19}$$

It is this marginalization that makes BMs significantly different from most other ML methods [298]. The marginalization used here involves iterating over all possible values of θ, then, for each such value, computing the *probability density function* over $\widetilde{Y}$ conditioned on X and that particular setting for θ, and weighing the distribution with the model's belief that θ should take that value (using the posterior). The result is the *posterior predictive distribution*. From this distribution, we can find $\widetilde{y}$

using (2.18) and the rules of integration:

$$\begin{aligned}
\mathbb{E}\left[\widetilde{Y} \mid X\right] &= \int_{\widetilde{Y}} \widetilde{y} p\left(\widetilde{Y} \mid X\right) d\widetilde{y} \\
&= \int_{\widetilde{Y}} \widetilde{y} \int_{\Theta} p\left(\widetilde{Y} \mid X, \theta\right) p\left(\theta \mid \mathcal{D}\right) \, d\theta \, d\widetilde{y} \\
&= \int_{\widetilde{Y}} \int_{\Theta} \widetilde{y} p\left(\widetilde{Y} \mid X, \theta\right) p\left(\theta \mid \mathcal{D}\right) \, d\theta \, d\widetilde{y} \\
&= \int_{\Theta} \int_{\widetilde{Y}} \widetilde{y} p\left(\widetilde{Y} \mid X, \theta\right) p\left(\theta \mid \mathcal{D}\right) d\widetilde{y} \, d\theta \\
&= \int_{\Theta} \int_{\widetilde{Y}} \widetilde{y} p\left(\widetilde{Y} \mid X, \theta\right) \, d\widetilde{y} \, p\left(\theta \mid \mathcal{D}\right) \, d\theta \\
&= \int_{\Theta} \mathbb{E}\left[\widetilde{Y} \mid X, \theta\right] p\left(\theta \mid \mathcal{D}\right) \, d\theta
\end{aligned} \tag{2.20}$$

The result is the integral over the posterior distribution with an inner expectation for $\widetilde{Y}$. Integrating out the posterior distribution is in most cases computationally intractable. Nevertheless, it is this marginalization that makes BMs better equipped to account for the various sources of uncertainty. Further, it makes the predictions more robust and less prone to be overly confident – especially for small datasets. To make a single prediction, one needs to evaluate the double integral from (2.20). These integrals can be computed analytically for some simple BMs, meaning the expectation can be computed with a finite number of mathematical operations. However, as BMs grow increasingly complex, these analytical solutions are rare. As an approximation, one could ignore the full distribution over the posterior and simply use a point estimate of the posterior, this point being the one with the highest probability, θ_{MAP} [99]. This would constitute a *maximum a posteriori* (MAP) estimate. With uninformative priors, this would even be the *maximum likelihood estimate* (MLE) [181]. This estimate will not reflect the model's uncertainty and is *not* a prediction that minimizes risk. Further, using θ_{MAP} instead of marginalizing over the posterior can lead to more extreme predictions [181, 189].

2.3.1 Monte Carlo Simulation

Luckily, one can do better than MLE and MAP. One can obtain an estimate of (2.20) through the simple *Monte Carlo* (MC) estimator [157, 159, 227]:

$$\begin{aligned}
\boldsymbol{\theta} &= \left[\theta_m \sim p\left(\theta \mid \mathcal{D}\right)\right]_{m=1,\ldots,M}^{\top} \\
\widetilde{y} &\approx \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}\left[\widetilde{Y} \mid X, \theta_m\right],
\end{aligned} \tag{2.21}$$

where M is the number of MC samples. Equation (2.21) can be calculated in parallel across CPU cores as each simulation runs independently of the other. Despite MC simulation being a simple, efficient, and fairly accurate approximation, the degree of parallelism is limited to the number of CPU cores available, and consequently insufficient in time-sensitive domains with the need for a high level of accuracy, i.e.,

more posterior samples used for prediction. For the most simplistic BMs, this is not a computational burden and involves only a few matrix operations. However, as the complexity of the model increases, so does the cost of making predictions. *This is particularly the case for BMs requiring some form of recursion.*

The "error" in this estimator is given by $\sigma(\widetilde{Y})/\sqrt{M}$, and thus, to halve this error, M should be increased by a factor of four [32]. Minimizing the error in this estimator is the goal, but it comes at an ever-increasing computational cost each time this error is to be halved.

2.4 Marketing Plans and Marketing Modeling

With PGMs and Bayes' risk in mind, next follows an introduction of a real-world optimization problem for which such ML tools can be used. Namely, the planning of optimized MMPs. This problem will serve as the domain at hand for which a BM is created for modeling marketing effect. A MMP outlines how a company intends to run commercials for an upcoming period. It establishes various commercial properties in great detail. These are properties such as *when* to execute each commercial, *which* product to advertise, and *what* platform to use. Careful planning of MMPs is needed, as commercials are generally expensive to run. It is easy to waste large quantities of money on commercial campaigns that are executed sub-optimally. One can view the whole planning process as an optimization problem. The objective is to optimize a utility function while obeying the budget and potentially other constraints set forth by a company's circumstances. Such constraints include, for instance, a minimum daily spend on a specific TV channel or a maximum total spend on online marketing. The utility function is typically measuring some *key performance indicator* (KPI) of interest, which can vary but is often net profit, daily clicks, or new subscribers.

The media landscape has changed rapidly with the introduction of the fourth industrial revolution. With the advance of online advertisements in unprecedented formats, this optimization problem has become tremendously convoluted. The increasing complexity happens as a result of advancements in customizing the four Ps (product, price, promotion, and place [221, 310]) at an unprecedented granular level. The endowment to pinpoint advertisements to specific demographic groups increases the likelihood of displaying relevant commercials and lessens inefficient ones. While it is reasonable to assume most companies have a decent idea of who their target audience is, it is far more difficult to account for the efficiency of individual campaigns targeted towards that audience. Is a 20-second TV commercial twice as good as a 10-second one? Should the commercial promote product A or B? At what time of day should the commercial run? Which TV channel? How does the TV commercial influence the effect of the other commercials? The number of decisions one needs to make when compiling these MMPs grows exponentially by introducing new insertions and even more granular configurations of each commercial.

These new possibilities for granular decisions are accompanied by tracking technologies such as *Urchin traffic monitor* (UTM) parameters and cookies. When combined, accurate customer targeting and tracking is achieved as a powerful asset enabling new data-mining and ML possibilities. Recent research in the field of ML for marketing [52, 56, 63, 97, 220], builds on these developments and therefore has direct, Internet-based marketing at its core. Here, the focus is on targeting high-value customers [56], addressing customer churn [52, 63], customer prospecting [220], or increasing advertisement personalization [97]. This preexisting work builds on a wide array of ML techniques ranging from genetic algorithms [195] to transfer learning [212] and specialized data-driven algorithms [97].

To measure the effect of online marketing, data processing tools such as *multi-touch attribution model*s (MTAMs) [253, 313] can be used. These are also built on online tracking mechanics like UTM parameters and cookies. MTAMs aim to add a degree of transparency by mapping the user journey from initial ad exposure to conversion and assigning credit to each step of the journey. With the worldwide trend of increasingly strict privacy legislation, the future of Multi-Touch Attribution is filled with uncertainty. Further, MTAMs have difficulties accounting for many influential parameters such as competitor activities, offline commercials, brand impact, and seasonal effect. For instance, a MTAM cannot answer whether there is a synergy effect of running TV commercials with in-store promotions [75, 133]. Consequently, MTAMs and online advertisement in general will not provide a complete picture of the effect of marketing campaigns and cannot account for non-media events.

Another privacy-friendly approach is through the use of MMMs based on econometric regression [273]. The technique is well established in the field and has roots dating back to the 1960s and 1970s [16, 46, 109, 140, 273]. The simplest example of such a model is the linear regression model:

$$y_t = \alpha + \beta a_t + \varepsilon_t \tag{2.22}$$

where a_t is the total amount spent on advertisements (aggregated across insertions) at timestep t, α and β are parameters to be inferred, ε_t is stochastic noise, and y_t is a KPI, e.g., sales [273]. In such an elementary setup, the prediction KPI is a sum of the current spending on advertisements. To infer the free parameters ($\theta = (\alpha, \beta)$) of such a model, one can, for instance, apply linear regression using *ordinary least squares* (OLS) to get a MLE. The MLE of these parameters is the single, most probable value. Real-world data, however, usually has multiple sources of uncertainty. Generally, one can classify a source of uncertainty into one of two categories. The first category is *aleatory* uncertainty and comes from the Latin word "alea," which can be translated as "the roll of dice." Aleatory uncertainty is the internal randomness of phenomena [61]. The second category is *epistemic*. The second category is the uncertainty caused by the lack of knowledge, meaning that we are only uncertain of an outcome due to the lack of information [61, 321]. For instance, we might be

uncertain about a company's sales due to only having aggregated data. Changes to the model from (2.22) could also reduce our epistemic uncertainty if these changes better reflect the real world. Creating a suitable model for a specific problem first and foremost requires a deep understanding of the domain and data.

Modeling a company's KPI given its historical data on direct and indirect marketing is challenging. The information on indirect marketing such as newspaper and radio advertisements is sparse, aggregated, and noisy compared to the online tracking technology counterpart. This lack of clean, large, disaggregate datasets, combined with large amounts of aleatory and epistemic uncertainty, makes for a delicate exercise to obtain accurate models. For instance, an observed customer pattern might be caused by an unmonitored event not reflected in the dataset due to the lack of offline tracking technologies.

When doing OLS on (2.22), one is trying to identify the single most probable set of values used to generate the observed data. It is bold to assume that only a single set of values is, in fact, correct. Instead, one should find a *distribution* over probable values for the parameters of a model to reflect the stochasticity in the underlying data generation process. The distribution should reflect the degree of belief that each possible assignment on these parameters is correct. To infer such distributions, one can use Bayes' theorem and BM as introduced in Section 2.2. In the following, we will introduce a well-known *Bayesian* MMM [170, 272, 273].

2.4.1 Bayesian Models for Mixture Marketing Plans

Businesses like Nepa Sweden AB[1], Blackwood Seven[2] and Adobe Experience Cloud[3] are founded on the premise of using BMs for modeling marketing. BMs are favorable for this type of domain, as the datasets are typically small compared to the number of predictors and free parameters of the model. Further, the significant amount of uncertainty (of both types) in this domain makes a MLE of the model parameters, θ, insufficient. There is aleatory uncertainty in consumer behavior and epistemic uncertainty due to the complexity of the field – removing all epistemic uncertainty is unrealistic. There will always be effects in the real world not captured by the model. Because of these domain-specific properties, BMs are suitable as they can capture the aleatory uncertainty.

Existing research has identified seven essential patterns of response that influence the effectiveness of advertising. These are current, carryover, shape, dynamic, content, and media effect [272, 273]. For more details on each of these effects, we will refer to [170, 272, 273]. This work is the foundation of the following MMM:

[1] https://nepa.com
[2] blackwoodseven.com
[3] adobe.com/experience-cloud

$$
\begin{aligned}
R_t\left(\mathbf{X}\right) &\sim \mathcal{N}\left(\beta_0 + \beta_{\text{coeff}} \sum_{j=1}^{J} A_t\left(\mathbf{X}\right), \sigma^2\right) \\
A_t\left(\mathbf{X}\right) &= \begin{cases} S_t\, \boldsymbol{\beta}_{\text{effect}} \tanh\left(\dfrac{\mathbf{X}_t}{\boldsymbol{\beta}_{\text{sat}}}\right) + \boldsymbol{\lambda}\, A_{t-1}\left(\mathbf{X}\right), & \text{if } t > 0 \\ A_{\text{inc}}, & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.23}
$$

This is an *auto-regressive* (AR) model with a Koyck lag (AR(1)). It works at a disaggregate level, as it models each of the J insertions' marketing effect at timestep t. The objective is to accurately predict a company's KPI given a marketing spend. A list of each variable involved is provided in Table 2.1, along with a short explanation and dimensionality. $\boldsymbol{\beta}_{\text{effect}}$, $\boldsymbol{\beta}_{\text{sat}}$, and $\boldsymbol{\lambda}$ are all J dimensional vectors, as they are mapped to each of the J insertions. The model uses the hyperbolic tangent to incorporate the shape effect. This function determines when increasing marketing spend no-longer affects the KPI. This is called *saturation*. The point of saturation is controlled through the scaling parameter $\boldsymbol{\beta}_{\text{sat}}$.

$\boldsymbol{\beta}_{\text{sat}}$ is needed, as some insertions saturate at a faster rate than others. $\boldsymbol{\beta}_{\text{effect}}$ controls each insertion's general effect on the KPI. $\boldsymbol{\lambda}$ controls the carryover effect of spending on marketing in the past, as spending in the past typically has a decaying effect on the future KPI. The model uses different carryover coefficients for each insertion to reflect that some have a higher long-term effect than others. Finally, the season influences the effect of marketing. The seasonal effect operates at different levels: time of day, week, month, and year. This effect is less dominant in some domains but is generally substantial. For instance, previous research has shown drastic changes in consumer behavior due to the seasonal effect for certain consumer-based companies [229]. The seasonal effect is controlled through S_t in this model.

As presented in (2.23), the model is simply a result of existing research, and we will not go into further details of the model and the underlying mathematical implementation, as it is not part of the core research of this work. Indeed, more sophisticated models could be considered that would allow for more complex phenomena like competitor effects or product price. Nonetheless, despite the simplicity of the model presented here, it can accurately predict the KPI for real-world companies and is thus sufficient for the purpose of this research [170].

2.4.2 Optimizing Mixture Marketing Plans

As described in Section 2.4, the planning process of a MMP can be viewed as an optimization process. Having a BM able to predict a company's KPI given a MMP, one can optimize the input to the BM to maximize the KPI. This optimization can, for instance, be done using *stochastic gradient descent* (SGD). In such a setting, the input to the model is free parameters, and the posterior distribution is kept fixed. Each step of SGD requires computing the KPI using the BM. This involves making a

TABLE 2.1: Explanation of variables for the model over marketing effect as presented in (2.23).

Symbol	Explanation	Dimensionality
R_t	A time-dependent function returning the total marketing effect from day 0 to t	Scalar
A_t	A time-dependent function returning the marketing effect for each insertion from day 0 to t	J dim. vector
A_{inc}	Initial effect	Scalar
β_0	Intercept	Scalar
β_{coeff}	Overall marketing effect	Scalar
S_t	Seasonal effect	Scalar
$\boldsymbol{\beta}_{\text{effect}}$	The marketing effect	J dim. vector
$\boldsymbol{\beta}_{\text{sat}}$	The saturation point	J dim. vector
$\boldsymbol{\lambda}$	The carryover effect	J dim. vector
$\mathbf{X}$	A MMP withinsertion spend	$T \times J$ matrix

point-estimate prediction for $\widetilde{Y}$ as in (2.20) – often through the use of MC simulation as in (2.21). With algorithms such as SGD, typically thousands of iterations and hence predictions are required before it converges to an optimum.

2.5 Chapter Summary

In this chapter, several ML concepts have been introduced. Although these concepts might seem fairly unrelated, the following chapters will apply these in various combinations. To get an overview of which concepts are applied in what chapters, Table 1.1 can be of assistance. In Chapter 3, we will present a method circumventing the need for the numerous time-consuming evaluations of (2.21) using the BM. In Chapter 4, we will expand on this method for swift generation of optimized MMPs, and demonstrate the method's appealing properties.

Chapter 3

Rapid Risk Minimization of Bayesian Models through Deep Learning

There are two computational issues with BMs 1) obtaining the posterior distribution over the free parameters, and 2) making predictions using the posterior predictive distribution as shown in Section 2.3. With the rise of sophisticated techniques for posterior inference like *variational inference* (VI) [27, 143, 289] and *Markov chain Monte Carlo* [192], BMs have gained an increasing level of popularity over the past decades. In contrast to the former computational issue, the latter has gained little attention in academic research. This chapter builds on NNs as presented in Section 2.1 and takes the best of BMs and NNs to address the computationally expensive predictions with BMs through approximation. This chapter first brings an introduction to NNs. Following is a section on some preexisting combinations of the two ML methods (PGMs and NNs) and with what purpose the combination was invented. Next follows a description of the actual method proposed in this work along with an analysis on the reduction in computational complexity gained by using this method. The method involves two algorithms used to minimize the computational burden of the approximation. These algorithms are tested on a regular Bayesian regression model and evaluated on the precision of the approximation and reduction in measured wall-clock time.

3.1 Bayesian Models versus Neural Networks

The toolbox of ML is ever-growing and contains a wide spectrum of tools. Each new tool has its advantages and drawbacks. This is especially true when comparing BMs and NNs.

As discussed in Section 2.3, marginalization is what distinguishes the Bayesian method from other ML tools like NNs [298]. In fact, the Bayesian framework has been a source of inspiration for many advances in the work with NNs, as it is inherently founded on probability theory [24, 105, 117, 183, 201, 203, 218, 277, 298, 299]. Bishop [24] performed a thorough walkthrough of some of these adopted concepts,

including model comparison, hyperparameter optimization, *active learning* (AL), ensemble methods, and ℓ_2 regularization [24, 25, 182].

It would seem that NNs have little to contribute to Bayesian methods, as they possess several unattractive properties compared to BMs. First and foremost, the inferred MLE of the free parameters of a NN is far more difficult to reason about and interpret compared to a PGM due to the entangled structure of NNs. Secondly, the number of free parameters of a NN is often several orders of magnitude higher than most BMs. For instance, GPT-3 uses 175 billion parameters [37]. Such a large number of free parameters gives them a high model capacity. A model with a high capacity is flexible, but dependent on large quantities of training data to prevent overfitting. The *Vapnik-Chervonenkis* dimension can be used to obtain an upper bound on the generalization loss of a model given its effective capacity and the size of the training dataset [284]. As the capacity of the model increases, the required size of the training dataset increases as well to maintain the same generalization performance [25, 99, 261]. NNs with millions of free parameters are, as a result, depending on much larger training datasets than that of BMs. In spite of that, one of their main assets is their extreme flexibility as they are model-free universal approximators [99, 124, 260]. This is appealing when it is infeasible to create a PGM for domains in which the underlying data-generation process is unknown. Finally, despite having far more free parameters, their extended use of matrix multiplication, execution on TPUs, and lack of marginalization make them very fast compared to BMs.

3.1.1 Bayesian Models and Neural Networks as Companions

With the aforementioned strengths and weaknesses in mind, the two ML approaches seem to complement each other nicely. In fact, numerous approaches for such a combination have already been proposed for overcoming various research challenges. In general, the application of NNs in these approaches is done with the purpose of utilizing either their predictive speed or approximation flexibility. For instance, recent development applies the Bayesian probabilistic framework and one or more NNs in a compound for doing VI of complex distributions. Examples of such approaches are generative models like *variational autoencoder*s [151] and normalizing flows [213, 234]. The latter can be used for approximating the posterior distribution of a Bayesian NN [173].

For such generative models, the posterior distribution and parameters of the NN are optimized jointly and thus tightly coupled. The composition leverages the flexibility of NNs and the principled probabilistic framework to infer distributions over data.

Another method of combining Bayesian methods and NNs is fitting a NN to the posterior distribution of a BM. In this setup, the posterior distribution of the BM is first inferred, and only afterward is the NN introduced. This is done with the purpose of leveraging the predictive speed of the NN.

A preexisting example of this method is presented by Jia et al. [138]. In this case, the NN learns to perform inference on the free parameter in the BM (θ) given some observations.

In the work of Pavone et al. [217], another approach was taken to approximate a NN to a BM. In this research, they used the domain of particle physics and sampled a target $t \sim p(T)$ using a joint prior distribution over the target variables. Then, I samples of data points $\mathbf{x} = [x_i \sim p(X \mid T = t)]_{i=1,\dots,I}^{\top}$ served as I input examples to the NN. The NN gained robustness and learned the uncertainty of the model through the data sampling process. The NN itself, however, predicted a single MLE for target t and thus had no way to represent the posterior predictive distribution in its output. The results show how using a NN for predictions rather than a BM can reduce the prediction time from over four hours to a matter of milliseconds. This demonstrates the potential benefits of using such setups [217].

3.2 The Best of Both Worlds for Risk Minimized Predictions

As mentioned at the beginning of this chapter, the ambition for this work is alleviating the time-demanding iterations over the full posterior predictive distribution of a BM for risk minimized predictions. This section presents the proposed method and algorithms for achieving this.

The method yields an approximation using a NN. When fitted, the NN is a function of the observation space and predicts an approximation to the full posterior predictive distribution of the BM conditioned on the same observation. In Section 3.6, we show how our method allows for such predictions faster than standard methods for BMs with many predictors. This happens due to the architecture as presented in Figure 3.1. The output of the NN is a *point-wise for θ estimate of the BM's mean posterior predictive distribution*. This approximation is obtained in a *single* feed-forward pass of the NN. Having access to the posterior predictive is beneficial for subsequent evaluation of the model, as it provides more detailed information about the predictions. This distribution can, for instance, be used for quantifying the model's uncertainty in the prediction.

Assuming only limited data was obtained and preprocessed for inferring the posterior distribution of the BM, this dataset is insufficient for training the NN. This issue is resolved by using the BM to generate a dataset of arbitrary size for the NN. Having a set of posterior samples from the BM, one can simply generate synthetic input data and pass it through the BM. This data needs to be evaluated on the BM multiple times, one time for each posterior sample used. The result is a discrete approximation of the mean of the posterior predictive distribution. It is important to note that the NN has no notion of the actual posterior samples of the BM. These are latent in the dataset generated for training the NN. In effect, this approach is fundamentally different from that of Jia et al. [138], as our NN has no notion of the parameters of the BM, it is simply latent in the dataset. In this way, our method

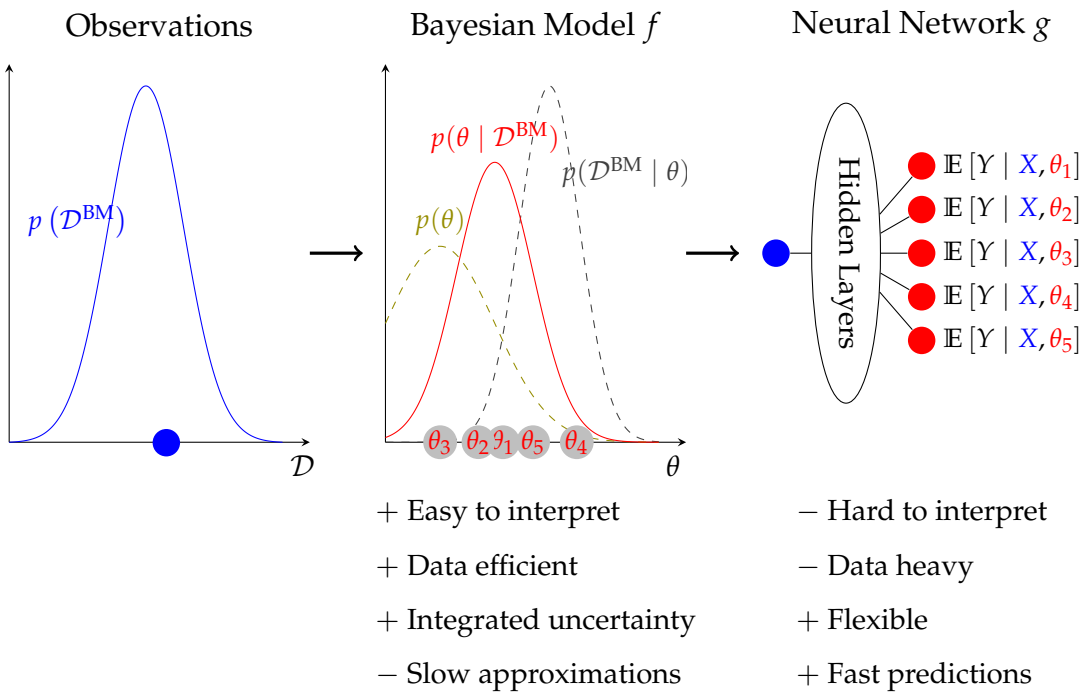


FIGURE 3.1: **The transition from data, to BM, to NN.** The colored dots indicate a sample from the corresponding distribution. First, the data is sampled. Then, the BM is fitted, and M samples from the posterior distribution are taken. Finally, the NN is trained such that for a given observation, the output of the NN corresponds to a prediction by the BM using each of the M posterior samples.

preserves the benefits from the BM but also gains the speed improvement from the NN.

Formally, we let $g_\phi : \mathbf{x} \subset \mathbb{R}_J \rightarrow \widetilde{\mathbf{y}} \subset \mathbb{R}_M$ be our approximation of the model (i.e., our NN), where ϕ are the free parameters of the NN. For inferring ϕ, we apply the loss function $\text{Smooth}_{\text{L1}}(\mathbf{y}, \widetilde{\mathbf{y}})$, with $\mathbf{y} = \left[y_m = \mathbb{E}\left[Y \mid \mathbf{x}, \boldsymbol{\theta}_m\right]\right]_{m=1,\dots,M}^{\top}$. $\boldsymbol{\theta}$ is a vector of size M with $\theta \sim p(\theta \mid \mathcal{D}^{\text{BM}})$ sampled prior to data generation and afterwards kept fixed[1]. For computing the expectations, the posterior predictive, $p\left(Y \mid X, \theta\right)$, from the BM is used. Finally, $\text{Smooth}_{\text{L1}}$ is the Smooth L1-loss, a combination of L1-loss and L2-loss [233].

The core algorithm contains three steps: *1)* Let $\boldsymbol{\theta}$ be a vector of M posterior samples taken from $p\left(\theta \mid \mathcal{D}^{\text{BM}}\right)$, *2)* sample the training dataset, $\mathcal{D}^{\text{NN}}$, using Algorithm 1, and *3)* fit g_ϕ to $\mathcal{D}^{\text{NN}}$. The sampling algorithm is outlined in Algorithm 1. Here, $\odot$ denotes the Hadamard product. We randomly *dropout* with probability $1 - \tau$ for each of the J sampled values in vector $\mathbf{x}$. This is done as a generalization technique to assist the NN recognizing invariants in the underlying model. This is similar to randomly removing pixels for an image recognition task, only in our case, the predicted

[1] *A note on notation*: As the predictions of the BM now become the ground true values to be approximated by the NN, the output of the BM is now denoted $\mathbf{y}$ and the output of the NN as $\widetilde{\mathbf{y}}$. Likewise, to distinguish the datasets used for inferring θ and ϕ, we accompany each dataset with a superscript for clarity.

Algorithm 1: Algorithm for the data sampling process for fitting the NN to a BM for risk minimization.

Input : I, $p(X)$, τ, $\boldsymbol{\theta}$
Output: $\mathcal{D}^{\text{NN}}$

$\mathcal{D}^{\text{NN}} \leftarrow \varnothing$
for $i \leftarrow 0$ **to** I **by** 1 **do**
  $\boldsymbol{\rho} \leftarrow \left[\boldsymbol{\rho}_j \sim \text{Bern}(\tau)\right]_{j=1,\dots,J}^{\top}$
  $\mathbf{x} \sim p(X)$
  $\mathbf{x} \leftarrow \mathbf{x} \odot \boldsymbol{\rho}$ // Dropout
  $\mathbf{y} \leftarrow [\mathbf{y}_m \leftarrow \mathbb{E}[Y \mid X = \mathbf{x}, \boldsymbol{\theta}_m]]_{m=1,\dots,M}^{\top}$
  $\mathcal{D}^{\text{NN}} \leftarrow \mathcal{D}^{\text{NN}} \cup \{(\mathbf{x}, \mathbf{y})\}$
end

value, $\mathbf{y}$, is changed with respect to the alteration of $\mathbf{x}$.

3.3 Computational Complexity

Generating data and fitting a NN to a BM is a computationally more time-consuming task compared to doing a MC simulation for a single prediction. If each BM prediction with MC has complexity $\mathcal{O}(m)$, with m being the number of samples, then making n predictions has complexity $\mathcal{O}(nm)$. As the number of predictions increases, the processing time increases linearly. In contrast, generating a dataset $\mathcal{D}^{\text{NN}}$ and doing n predictions has complexity $\mathcal{O}(\kappa m + n)$ with κ being the number of dataset samples required for training the NN. When $n \geq \kappa m/(m-1)$, our method has the lowest overall complexity. One might wonder if this is a real use case, as κ, is usually in the thousands. Some optimization problems and domains, however, are dependent on solving the conditional expectation, $\mathbb{E}[Y \mid X]$, a vast amount of times for varying input. Moreover, for a real-time application, once trained, the NN will provide better response times and thus improve user experience.

Having a fitted NN and $m > 1$, using the NN allows for risk minimized predictions with a lower complexity for a single observation compared to the BM. This makes our method appealing in some time-sensitive domains. These are domains with surplus time when fitting the BM, but at some point later require accurate predictions quickly for new input. Examples of such domains could be stock market trading or autonomous vehicles: both needing rapid, correct reactions to new observations, but having surplus time when the stock market is closed or research is being conducted. Further, *our method is beneficial when the rate of new data incoming surpasses the time it takes to evaluate* (2.21) *using the BM*. Examples include particle physics and big data applications with a high data velocity.

Some software frameworks such as Jax [33] enable executing BMs on the GPU rather than the CPU. This provides some gain in speed. However, the computational

complexity analysis remains the same, as the same sequence of statements needs to be computed for the BM.

3.4 Active Learning

As the computational complexity of using our method depends on κ (the size of the training set), minimizing κ increases the computational benefit of our proposed method. Thus, we apply AL as an extension of the regular training algorithm to maximize the computational benefit. The approach is outlined in Algorithm 2 and works by iteratively expanding the training dataset for the NN. This is done through modifications to the data sampling distribution, $p(X)$, from which new data is generated. The iterative, data-augmented approach helps to minimize the computational burden of evaluating the BM. During the first iteration, a uniform distribution is used for generating the initial dataset of size I^{Init}. Subsequently, this distribution is changed to a categorical distribution with probabilities reflecting the NN's predictive *uncertainty* on entries from a large, unlabeled, uniformly sampled dataset $\mathbf{X}^{\text{uncert.}}$. From this categorical distribution, I^{AL} entries are sampled with replacement. These are then labeled using the BM and added to the training dataset for the next round.

The aforementioned uncertainty is measured through the use of dropout. For each example $\mathbf{x}^{\text{uncert.}} \in \mathbf{X}^{\text{uncert.}}$, we measure the NN's predictive uncertainty. This is done by performing K feed-forward passes through the network for $\mathbf{x}^{\text{uncert.}}$ with dropout enabled, such that the predictions, $\widetilde{\mathbf{Y}}^{\text{uncert.}}$, are a $M \times K$ matrix. To reduce this matrix to a single value reflecting uncertainty on $\mathbf{x}^{\text{uncert.}}$, we take the standard deviation over K to obtain a vector of M standard deviations, for which we calculate the mean:

$$\sigma_{\mathbf{x}^{\text{uncert.}}} = \frac{1}{M} \sum_{m=1}^{M} \sigma\left(\widetilde{\mathbf{Y}}_m^{\text{uncert.}}\right) \tag{3.1}$$

The larger standard deviation means a higher uncertainty and, as a result, a larger sampling probability from the categorical distribution. This operation is performed over each instance in the unlabeled dataset, and the resulting set of σ values is transformed using Softmax to be used as probabilities.

Finally, we apply *early stopping* (ES) [225] at two levels. We will refer to these as *intra-* and *inter*-ES. Intra-ES regulates when to stop fitting the NN on $\mathcal{D}^{\text{NN}}$, while inter-ES determines when to stop the AL iterations. When inter-ES chooses to stop, it restores the state of the NN to the best-performing one. Both ESs are using the same fixed validation dataset to ascertain the performance of the NN.

Algorithm 2: Active Learning algorithm.

Input : τ, $p\left(\theta \mid \mathcal{D}^{\text{BM}}\right)$, patience
Output: g

stopper $\leftarrow$ EarlyStopping(patience)
$X \sim \mathcal{U}\left(\mathbf{0}_J, \mathbf{1}_J\right)$
$g \leftarrow$ initialize NN
$\boldsymbol{\theta} \leftarrow \left[\boldsymbol{\theta}_m \sim p\left(\theta \mid \mathcal{D}^{\text{BM}}\right)\right]_{m=1,\ldots,M}^{\top}$
$\mathcal{D}^{\text{NN}} \leftarrow$ Algorithm 1(I^{Init}, $p\left(X\right)$, τ, $\boldsymbol{\theta}$)
do
  $g \leftarrow$ (re)train g on $\mathcal{D}^{\text{NN}}$
  $\boldsymbol{\sigma} \leftarrow$ MeasureUncertainty $\left(g\right)$
  $\boldsymbol{\pi} \leftarrow$ Softmax $\left(\boldsymbol{\sigma}\right)$
  $X \sim$ Cat $\left(\boldsymbol{\pi}\right)$
  $\mathcal{D}^{\text{NN}} \leftarrow \mathcal{D}^{\text{NN}} \cup$ Algorithm 1(I^{AL}, $p\left(X\right)$, τ, $\boldsymbol{\theta}$)
while stopper.should continue $\left(g\right)$

3.5 Experiments

For the experiments in this chapter, we assume a Bayesian regression model, $f_\theta : \mathbf{x} \subset \mathbb{R}_J \rightarrow y \subset \mathbb{R}$, where $\mathbf{x}$ is a single example. The model is as follows

$$\begin{aligned} \boldsymbol{\alpha} &\sim \mathcal{N}\left(1.5,\, \mathbf{I}_J\right); & \boldsymbol{\beta} &\sim \mathcal{N}\left(0.5,\, 0.25\mathbf{I}_J\right) \\ \sigma^2 &\sim \mathcal{N}\left(0,\, 1\right); & \gamma &\sim \mathcal{N}\left(0,\, 0.5\right) \\ f\left(\mathbf{x}\right) = \gamma + \sum_{j=1}^{J} \boldsymbol{\beta}_j \psi\left(\mathbf{x}_j \boldsymbol{\alpha}_j\right); & & Y &\sim \mathcal{N}\left(f\left(\mathbf{x}\right),\, \sigma^2\right) \end{aligned} \tag{3.2}$$

with $\mathbf{I}_J$ being the identity matrix with rank J. The idea behind this model is that the output is the sum of a linear transformation of each feature that is altered by some function, ψ. This function is domain-dependent, and could be $\sqrt{}$, sin, log, sigmoid, or the like. $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\sigma}$ are J-dimensional column vectors and represent the parameters of the BM that needs to be inferred. The BM outlined above does not directly resemble any real-world model. Nonetheless, one can easily imagine such a setup for modeling periodic data as the sum of sine functions, with each feature having a different phase. Another example could be a model for diagnosing the risk of having a particular disease. Here, $\mathbf{x}$ could be a patient's blood test results, $\boldsymbol{\beta}$ the impact of each examined property, ψ the sigmoid function to saturate the influence of each property, and $\boldsymbol{\alpha}$ a scaling factor.

Next, we are interested in using the model for predictions over the full posterior on new data. We will assume no closed-form solution of the posterior predictive distribution. Thus, for doing predictions with the BM, we will apply MC-sampling, as in (2.21), and assume a sufficient posterior fit to the observed data $\mathcal{D}^{\text{BM}}$.

For predictions using this model, each of the $m \in \{1, \ldots, M\}$ posterior samples

of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and γ needs to be used. The computational complexity of (3.2) is linearly dependent on J. Thus, making predictions over the full posterior becomes increasingly computationally heavy as the number of features grows. One could imagine other terms that reflect more complex effects such as synergies across features to take place in the model. *A synergy effect between all features would transform the linearly growing complexity into an exponential one.*

In contrast to the complexity growth of the BM for increasingly complex tasks, the same growth is not necessarily present for NNs. Using the property of NNs being universal approximators, adding more complexity to the BM does not necessarily require a larger NN as long as its capacity is not fully utilized. Even using a *single* hidden layer in the NN, with enough width, it remains a universal approximator [206]. Consequently, in the extreme case, the NN approximation provides extremely fast predictions, as it will only involve two matrix multiplications, two addition terms, and an activation function.

To examine this, our experiments will focus on approximating the BM from (3.2) with a NN. We will vary the computational complexity of the model through J and evaluate the NN's fit to the BM. The main motivational factor for introducing a NN as an approximation of a BM is the gain of speed when making multiple predictions. To quantify this, we measure the runtime for making predictions on the whole testing dataset using M posterior samples. We will refer to this runtime as prediction time. Our results will focus on the prediction time as a function of model complexity, and the required size of the training dataset.

Throughout all experiments, we use a simple, feed-forward NN with a single hidden, dense, layer with a width of 5000. We use dropout [262] with a rate of 0.5, and batch normalization [12] after the hidden layer. We let $I^{\text{Init}} = 10\,000$, $I^{\text{AL}} = 1000$, $\tau = 0.8$, and $M = 2000$. We keep M fixed for all experiments and instead vary J, but one could make similar experiments for an increasing number of posterior samples. For training the NN, we use a learning rate of 3×10^{-4}, and set the patience to 10 and 20 for inter- and intra-ES, respectively. The validation and testing dataset each contain 50 000 examples. The experiments are executed on an Intel Core i7-6700K with 48GB ram and a NVIDIA GeForce RTX 2080TI GPU. We use PyMC3 [248] version 3.9.3 for Bayesian inference with the No-U-Turn Sampler [121] using 2000 warmup steps and 2000 samples. $\mathcal{D}^{\text{BM}} = \{\mathbf{X}^{\text{BM}}, \mathbf{Y}^{\text{BM}}\}$ contains $N = 5000$ examples with $\mathbf{X}^{\text{BM}}$ being sampled from a standard Gaussian distribution, and $\mathbf{Y}^{\text{BM}} = \left[\mathbf{Y}_n^{\text{BM}} = f\left(\mathbf{X}_n^{\text{BM}}\right)\right]_{n=1,\dots,N}^{\top}$ from (3.2) with the ground-true values presampled using $\widehat{\gamma} \sim \mathcal{U}\left(-0.5, 0.5\right)$, $\widehat{\boldsymbol{\alpha}} \sim \mathcal{U}\left(0.3 \cdot \mathbf{1}_J, 3.0 \cdot \mathbf{1}_J\right)$, and $\widehat{\boldsymbol{\beta}} \sim \mathcal{U}\left(0.1 \cdot \mathbf{1}_J, \mathbf{1}_J\right)$.

3.6 Results

In Figure 3.2, one can see the prediction time on the testing set for increasing model complexities, the corresponding *mean squared error* (MSE), and the size of the training dataset used for fitting the NN. When measuring the prediction time of the BMs,

the operation is run in parallel using all cores of the CPU when iterating over the M posterior samples. This is done using a pool of threads. All calculations make use of the Numpy library with vectorized operations to obtain the best possible performance. The PyMC3 framework also provides a method for making predictions using vectorized computations. However, this does not make use of parallelization and therefore yields a worse prediction time than those we have reported here using both vectorized operations and concurrency.

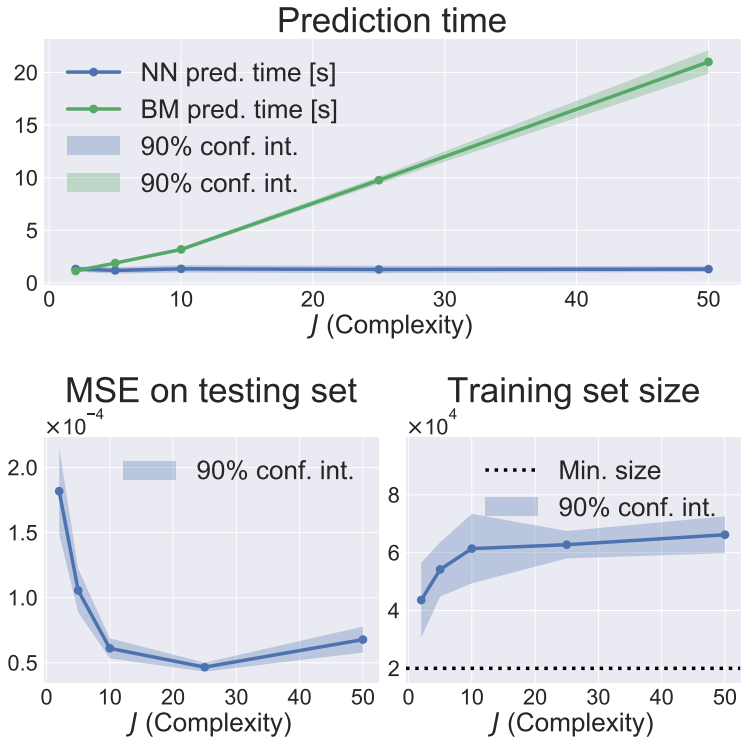


FIGURE 3.2: **Prediction time (top), MSE on the testing dataset (bottom left), and the ultimate size of the training dataset (bottom right), shown as a function of model complexity J.** The lines indicate the mean of each experiment, which is repeated five times. The shades are the 90% confidence interval bands, and the markers show the values of J for which experiments were conducted. The prediction time on the testing dataset using the BM versus the NN is shown in the top figure. This figure illustrates the linear relation between the model complexity and the prediction time using the BM while being constant for the NN. The bottom left figure shows the MSE of the NN calculated on the testing dataset. The bottom right figure shows how the size of the training dataset increases nonlinearly with the complexity of the BM to be approximated. Here, the dotted line indicates the lowest possible size of the training dataset, as each experiment starts with 10 000 examples and passes at least 10 AL iterations.

Measuring runtimes on modern computers is generally an error-prone task, and therefore, we repeated each experiment five times. From the figure, the linear relationship between complexity and prediction time using the BM is clear, while it remains constant for the NN. As complexity increases, we see a small decreasing tendency in the MSE. The MSEs is for all experiments considered low, spanning from

4.3×10^{-5} to 2.1×10^{-4}. The results indicate a negative correlation between the MSE and model complexity. We find this somewhat surprising, as it seems intuitive that more complex models would be harder for the NN to approximate. We interpret this phenomenon as a result of the central limit theorem [224], as the normalized distribution over $\mathbf{Y}^{\text{NN}}$ shrinks to a normal, making it easier for the NN.

3.6.1 Active Learning and the Size of the Training Dataset

To examine the strength of the correlation between the NN's uncertainty and the predictive error on the $\mathbf{X}^{\text{uncert.}}$ dataset, a calibration plot is shown in Figure 3.3. This figure shows the NN's uncertainty (calculated using (3.1)) as a function of the average *root mean squared error*, μ_{RMSE}. This average is taken over the $K \times M$ predictions. The figure shows the correlation for an experiment with complexity $J = 5$. Although the correlation is not equally strong for all examples, it shows a general tendency. If the correlation were not present, the data newly added to the training dataset would provide little to no improvement on the validation dataset. If that were the case, the ES algorithm would terminate the training process as soon as the iteration counter reached the patience threshold. This level is indicated by the dotted line in Figure 3.2. The line illustrates the smallest possible size of the training dataset (20 000) when using the hyperparameters $I^{\text{Init}} = 10\,000$, $I^{\text{AL}} = 1000$, and an inter-ES patience of 10.

3.7 Discussion

The use of dropout as a measure of uncertainty assumes a correlation between the standard deviation over the K predictions and the actual ground true error. [279] showed that such a correlation exists for a range of different problems for which $\widetilde{y} \subset \mathbb{R}$. In our domain presented here, we have increased complexity, as $\widetilde{\mathbf{y}} \subset \mathbb{R}_M$ with $M \gg 1$. In (3.1), we take the mean over M to obtain a single scalar reflecting the NN's joint uncertainty over all predicted values. Whether this is sufficient summary statistics for the use of AL is unsettled, and an area for future research. One could potentially simplify the method by making the NN predict $\mathbb{E}\left[\widetilde{Y} \mid X\right]$ instead. Such a change would mean $M = 1$ and only require an alteration to the generation of $\mathcal{D}^{\text{NN}}$ by performing the averaging step in (2.21) when computing $\mathbf{Y}^{\text{NN}}$ using the BM. We argue that one might as well do the MC simulation step on the output of the NN rather than when generating the training data for the NN. The MC simulation involves a summation operation, and thus information to the NN is lost. Therefore, we argue that the training task of the NN simply becomes easier. Further, having a discrete approximation of the posterior predictive distribution allows one to recover the uncertainty of the BM in a way the summation function would make irrecoverable.

The BM presented in (3.2) is relatively simplistic. More complex effects such as synergies across features or an autoregressive component can significantly increase

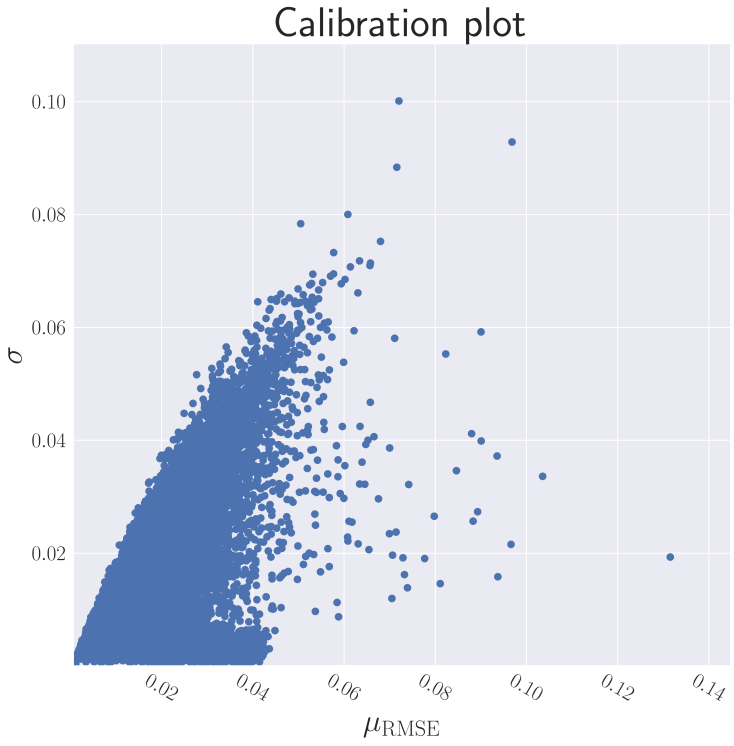


FIGURE 3.3: **Calibration plot showing the correlation between the uncertainty of the NN, σ, and its prediction's root mean squared error on newly sampled data, μ_{RMSE}.** The correlation allows for sampling additional data purely based on the uncertainty estimate of the data points.

the prediction time of a BM. Future research should extend the BM with such effects and evaluate the NN's ability to approximate such constructs, as these leave room for even larger speed-gains over the BM.

A drawback of our method is the assumption that the BM is fixed. Changes to the BM's construct or posterior distribution would require a refit of the NN. Depending on the magnitude of the change in the BM, one might achieve good results by using transfer learning [99, 276], by transferring the state of the original NN to fit the updated BM.

3.8 Chapter Summary

This chapter builds on the theory from Chapter 2. Specifically, Chapter 2 discussed the computationally heavy risk minimized predictions with BMs. This issue serves as the motivation for the method proposed in this chapter. As the proposed method uses NNs, this chapter began with an introduction to the general concept of NNs. Then, building on this concept, a formal introduction to the proposed method was given in Section 3.2. Using this method, the NN predicts a vector of point-wise estimations of the BM's posterior predictive distribution $\mathbb{E}\left[\widetilde{Y} \mid X, \boldsymbol{\theta}\right]$. The fitted NN becomes a function of the same observational space as the BM, and the output is

an approximation of the BM's posterior predictive distribution conditioned on the same observation.

The method was tested on a Bayesian regression model. The experiments disclosed the appealing properties of the method as it enables computationally cheap risk minimized predictions and scales better as a function of domain complexity.

These results indicate how the approach can be useful in domains with a need for fast, precise, risk minimized predictions. An example of such a setting is an application with a high data-velocity, i.e., when data arrives faster than one can generate predictions with the BM.

Despite the impressive gain in speed with a negligible loss in prediction accuracy, the results presented in this chapter are based on an artificially constructed regression model.

To further justify the approach, the following chapter applies the method in a real-world setting within the field of marketing.

Chapter 4

Combining Deep Learning and Bayesian Models for Swift Generation of Mix-Marketing Plans

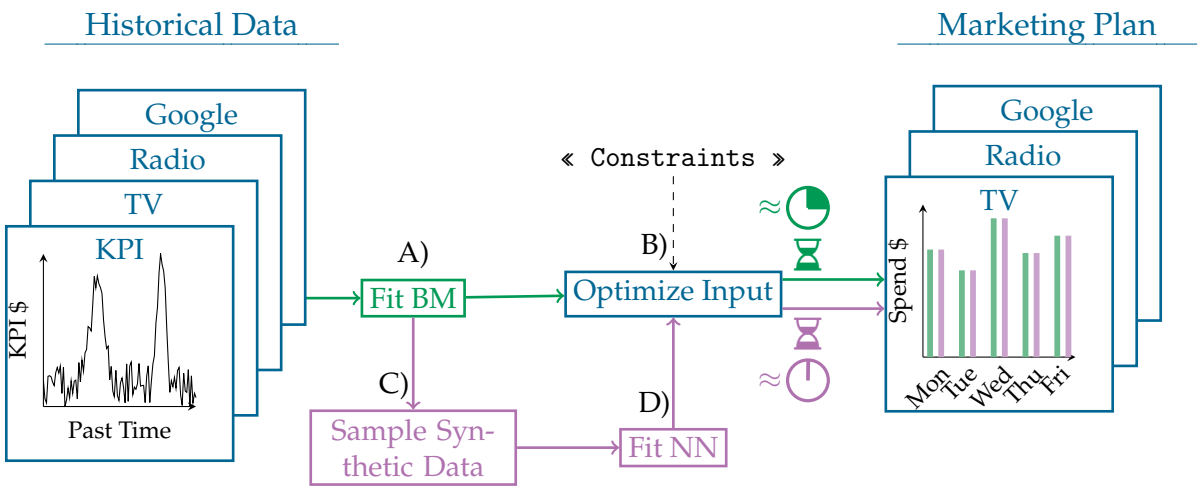


FIGURE 4.1: **The steps for obtaining an optimized MMP based on historical data.** The ordinary approach involves only step A) and B) through the use of a BM. Our proposed method includes two additional steps, C) and D). These obtain a NN *approximated* to the BM. The NN is then used for step B) as a surrogate. In turn, performing step B) becomes up to 65× faster than using the BM with only a 0.46% loss in KPI on average. Hence, the MMP deriving from the proposed method is almost identical to that of the ordinary approach. The « `constraints` » are set forth by the marketing employee querying for an optimized MMP. An example of such a constraint could be the minimum spending on TV commercials on a specific day.

The previous chapter discussed two computationally heavy operations with Bayesian models: inferring the posterior distribution and conducting risk minimized predictions. The chapter focused on the latter and proposed a method for alleviating the

computational burden using an approximation with NNs and AL. So far, the application of the method has been purely academic but has suggested intriguing gains in speed compared to standard methods. This chapter contains a proposition for extending the approach in order to apply the method in a real-world setting with the purpose of generating optimized MMPs. Thus, the chapter relies heavily on concepts introduced in Section 2.4 and 2.4.2.

Unlike the previous experiments, which used a fabricated BM, this chapter applies the generic method to marketing and demonstrates how it can be used for input optimization to obtain KPI-optimized MMPs. *The overall purpose is to bypass the need for thousands of time-demanding risk minimized predictions with the BM when using SGD to optimize a MMP.*

Results from the previous experiments indicate that the proposed method can be used for such risk minimized predictions using an approximated NN instead of the BM with a negligible loss in prediction accuracy on the testing dataset. However, there is a significant difference between making predictions on a testing dataset sampled from the same distribution as the training data and performing input optimization. Having a good performance on such a testing dataset does not guarantee that the optimized input is in fact optimal. The input optimization can drift into areas of the search space in which the NN is not fitted correctly. Such areas can occur when the data sampling distribution does not uniformly cover all parts of the search space, which can be challenging to accomplish.

Samples from *out-of-distribution* areas can lead to suboptimal predictions by the NN, and are closely related to the concept of adversarial attacks. Adversarial attacks are an ongoing field of research in the ML community and will be discussed in greater detail in part III. In short, adversarial attacks trick a NN to make highly confident incorrect predictions on generated, malicious, out-of-distribution examples.

As one cannot know beforehand on which areas of the search space the NN will have to make predictions during the input optimization, the generated MMPs will depend on how well the NN generalizes. Thus, it is of high importance that the NN learn the underlying properties of the BM and invariants.

This chapter will be using a Bayesian MMM, as presented in (2.23) and repeated below for ease of access:

$$\begin{aligned} R_t\left(\mathbf{X}\right) &\sim \mathcal{N}\left(\beta_0 + \beta_{\text{coeff}} \sum_{j=1}^{J} A_t\left(\mathbf{X}\right), \sigma^2\right) \\ A_t\left(\mathbf{X}\right) &= \begin{cases} S_t\, \boldsymbol{\beta}_{\text{effect}} \tanh\left(\dfrac{\mathbf{X}_t}{\boldsymbol{\beta}_{\text{sat}}}\right) + \boldsymbol{\lambda}\, A_{t-1}\left(\mathbf{X}\right), & \text{if } t > 0 \\ A_{\text{inc}}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.1}$$

This model is more complex than the Bayesian regression model from Chapter 3, as it is an AR(1) model with a seasonal component. The increased model and domain complexity requires some modifications to the NN and data generation process from the previous chapter, this being the subject of Section 4.1 and 4.2, respectively. After

this, a domain-dependent loss function is introduced in Section 4.3. This loss function is used for the input optimization procedure. To visualize the domain at hand, a low-dimensional experiment is carried out in Section 4.4.1. Here, an artificial setup with only two insertions is carefully designed as a proof-of-concept. This setup is used to visualize the properties of the task and the loss landscape of the input optimization, and to evaluate the solution vector of the input optimization using the NN versus the ground true for a set of budgetary constraints.

Section 4.4.2 highlights the potential of the method being applied in this part of the work. The experiments in that section are a result of applying the method in an industrial setting with >100 insertions. The generated MMPs are compared to ones used in the industry. These results stress the real-world potential of the approach which yields high-performing MMPs that are generated more than a full order of magnitude faster than through traditional methods. This raises new possibilities for the optimization process itself, as it now becomes feasible to generate numerous optimized MMPs that differ in their solution. Figure 4.1 visualizes the method's change in the workflow when generating optimized MMPs. Once the NN has been trained, the new steps allow for much faster generation of MMPs.

4.1 Data Generation

Company data on past marketing spending and KPI is often sparse. The historical data is, in the cases considered here, disaggregated only to a daily level. Further, information dating back several years does not reflect the current marketing situation of a company; new insertions arise, the line of products changes, the competitor landscape is ever more dynamic, etc. The dynamic scenery quickly renders the historical data obsolete. Therefore, it is often the case that only the past few years of data are applicable, leaving only ≈ 730 datapoints for a high-dimensional setting. While this can be enough to watchfully infer the posterior distribution of a BM, this is not sufficient for training and testing a NN.

Instead, the approach taken here is an extension of the data generation process presented in Algorithm 1 (p. 35) to sample synthetic data for training the NN. First, a pre-sampled set of $\boldsymbol{\theta}$ is created by sampling from the fitted posterior of a BM $\boldsymbol{\theta} = [\boldsymbol{\theta}_m \sim p(\theta \mid \mathcal{D})]_{m=1,\dots,M}^{\top}$. Next, an observation $\mathbf{x}$ is sampled unconditionally. Finally, the BM is used to make a set of predictions conditioned on the sampled observation. These predictions are generated by using the posterior predictive distribution of the BM $\boldsymbol{y} = [\mathbf{y}_m = \mathbb{E}[Y \mid \boldsymbol{\theta}_m, X = \mathbf{x}]]_{m=1,\dots,M}^{\top}$. In this way, the NN approximates a pointwise for θ posterior predictive distribution of the original BM conditioned on an arbitrary observation. List 4.1 outlines the steps involved in the data generation process for sampling a single observation. In the list, T indicates the end date of the period of interest. $\mathbf{0}_{T\times J}$ denotes a $T \times J$ matrix of zeros. $\widehat{\mathbf{X}}$ is the historical spend data. $\mathbf{e}_j$ is the standard basis column vector. F^{-1} is the inverse cumulative distribution function over $\widehat{\mathbf{X}}\mathbf{e}_j$, i.e., the historical spending on insertion j. $\odot$ is the Hadamard

LIST 4.1: The list of steps involved for generating a single example in the dataset.

Step 1. $\mathbf{X} \leftarrow \mathbf{0}_{T \times J}$ — *Initialize spending to 0*

Step 2. $\mathbf{x}^{(\max)} \leftarrow \left[F^{-1}_{\widehat{\mathbf{X}}\mathbf{e}_j}(1) \right]^{\top}_{j=1,\dots,J}$ — *Obtain upper bound spend for each insertion*

Step 3. $\boldsymbol{\rho} \leftarrow \left[\boldsymbol{\rho}_j \sim \text{Bern}(\tau) \right]^{\top}_{j=1,\dots,J}$ — *Sample spending dropout with probability* τ

Step 4. $t \sim \text{Mult}(\mathbf{p} = \mathbf{1}_T / T)$ — *Sample spending date uniformly from day* 0 *to* T

Step 5. $\mathbf{X}_t \leftarrow \left[\mathbf{x}_j \sim \mathcal{U}\left(0, \mathbf{x}_j^{(\max)}\right) \right]^{\top}_{j=1,\dots,J}$ — *Sample spending and update row* t *of* $\mathbf{X}$

Step 6. $\mathbf{X}_t \leftarrow \boldsymbol{\rho} \odot \mathbf{X}_t$ — *Dropout spending*

Step 7. $\mathbf{y} \leftarrow \left[\mathbf{y}_m = \sum_{i=1}^{\infty} R_i(\mathbf{X}) \right]^{\top}_{m=1,\dots,M}$ — *Compute BM output*

product. Finally, R_i is used from (2.23) and is evaluated using the specific posterior sample θ_m.

$\mathbf{X}$ in the list of sampling steps is a sparse matrix with only row t being nonzero. In effect, $\mathbf{y}$ will only reflect the total marketing effect for spends on a single day. For this reason, only the vector $\mathbf{x}_t$ is taking part in the collected dataset. The data sampling process in List 4.1 is repeated I times to collect an entire dataset. Let $\mathcal{D}^{\text{NN}} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \dots, (\mathbf{x}^{(I)}, \mathbf{y}^{(I)})\}$ denote the full dataset with I observations used for fitting and evaluating the NN.

To keep the training process efficient, the input to the NN is still a matrix $\mathbf{X}_{K \times J}$, with K being the batch size. The output of the NN denoted $\widetilde{\mathbf{Y}}_{K \times M}$ is the predicted marketing effect for each of the K examples and M posterior samples. The prediction for each example $k \in K$ is computed *independently* of the other examples. Thus, $\mathbf{X}_{K \times J}$ will contain data for dates approximately uniformly distributed over the examined period $[0, T]$.

One might question why the NN does not contain a recurrent component like the original model being approximated from (2.23). Having a recurrent NN like LSTM would seemingly be more beneficial for this application. To see why this is unnecessary, let $\mathbf{X}$ be a matrix with some spending at day t. The effect of the

particular spending at day t, namely $\mathbf{x}_t$, will have an immediate effect of

$$\begin{aligned} \text{Effect}_{\text{imm}} &= R_t\left(\mathbf{X}\right) - R_t\left(\mathbf{X}'\right) \\ &= S_t\, \boldsymbol{\beta}_{\text{effect}} \tanh\left(\mathbf{X}_t / \boldsymbol{\beta}_{\text{sat}}\right) \end{aligned} \tag{4.2}$$

where $\mathbf{X}'$ differs from $\mathbf{X}$ only in row t with $\mathbf{X}'_t = \mathbf{0}_J$. In addition to $\text{Effect}_{\text{imm}}$, $\mathbf{X}_t$ will also affect the future KPI through carryover. Assuming $\mid \boldsymbol{\lambda} \mid < 1$ (the carryover effect), this future effect of spend $\mathbf{X}_t$ will decrease exponentially. The *total* effect of $\mathbf{X}_t$ is the immediate effect plus all future effects

$$\begin{aligned} \text{Effect}_{\text{total}} &= \sum_{i=1}^{\infty} R_i\left(\mathbf{X}\right) - R_i\left(\mathbf{X}'\right) \\ &= \underbrace{0 + \cdots + 0}_{i<t} + \\ &\quad \underbrace{S_t \boldsymbol{\beta}_{\text{effect}} \tanh\left(\mathbf{X}_t / \boldsymbol{\beta}_{\text{sat}}\right)}_{i=t} + \\ &\quad \underbrace{\boldsymbol{\lambda}\left(S_t \boldsymbol{\beta}_{\text{effect}} \tanh\left(\mathbf{X}_t / \boldsymbol{\beta}_{\text{sat}}\right) + \boldsymbol{\lambda}(\dots)\right)}_{i>t} \\ &= \text{Effect}_{\text{imm}} \boldsymbol{\lambda}^0 + \text{Effect}_{\text{imm}} \boldsymbol{\lambda}^1 \dots \text{Effect}_{\text{imm}} \boldsymbol{\lambda}^{\infty} \\ &= \sum_{i=1}^{\infty} \text{Effect}_{\text{imm}} \boldsymbol{\lambda}^i \end{aligned} \tag{4.3}$$

This is a sum of a geometric series and thus can be rewritten to $\text{Effect}_{\text{total}} = \text{Effect}_{\text{imm}} / (1 - \boldsymbol{\lambda})$. This is a significant result, as the infinite sum requiring a recurrent model is reduced to a closed-form solution for which no recurrent component is needed. Instead, the carryover effect of $\mathbf{X}_t$ is latent in the corresponding $\mathbf{y}$ to be predicted by the NN. For similar reasons, the infinite sum in step seven of the enumerated data generation process above has a closed-form solution.

It could be argued that this model is too simplistic, as the total effect of a spend at day t is invariant to previous spends leading up to t. For instance, it could be argued that the effect of Google Advertisements at day t becomes affected by the number of related TV commercials at day $t-1$. Similarly, the carryover effect is also invariant to whatever other marketing spend used that day. This property results from the underlying model in (2.23) and a simplification of the real world. An improved model could have a time-dependent synergy in the total effect.

4.1.1 Learning the Correct Attribution

From the model in (4.1), the marketing effect of insertion j does not alter the marketing effect of insertion $\neq j$ as there is no *synergy* effect present. The NN will have to learn that the effect of spending on insertion j is *invariant* to all other spends. In the real world, one could imagine a synergy effect between one or more insertions, and

improvements to the model could incorporate such an effect. Nonetheless, in this simple model, there is no such effect.

Due to the nature of a NN, it needs to learn the correct attribution for all values and combinations of $\mathbf{x}$; hence, large quantities of data are needed. As each entry in the dataset needs labeling using the BM, collecting data is a time-consuming process despite the data being synthetic. The "dropout" effect of the data generation process helps the NN learn the invariant properties of the domain by randomly assigning a predictor to the value zero.

The dropout component also alleviates another issue with the approximation. Namely, when the marketing effect of an insertion is small, e.g., a local newspaper, the effect is easily overshadowed by insertions with substantial spending and effect. Without the random removal of spend on insertions, the NN will be biased toward attributing all effect to the insertions with a substantial effect. Such a bias can be vital for the subsequent input optimization, as the NN will underestimate the effect of insertions with a less sizable effect. The dropout ensures that the training dataset contains examples with and without spending on the insertions with substantial influence on the KPI.

The rate at which this dropout occurs is a new hyperparameter introduced, τ. Setting τ too high removes the effect of dropout and makes it more challenging to learn the invariants.

On the other hand, setting τ too low removes most predictors in the dataset, making the NN biased towards instances with primary zeros. Such bias will, in effect, make the NN underestimate the effect of marketing plans with a large budget. One should aim for a *sweet spot* in-between, for which the data generation remains as efficient as possible while still demonstrating, to a sufficient extent, the invariants of the domain.

Appendix A contains visualized empirical results for different settings of τ and its effect on the fitted NN when predicting the KPI for two distinct predictors.

4.2 Replicating Parts of the BM in the Architecture of the NN

The hyperbolic tangent function taking part in the BM as seen in (4.1) is a key component to model saturation in consumer response. It is important to identify the saturation point at which additional marketing spending does not lead to an increase in KPI. To assist the otherwise model-free NN in identifying this property, it can easily and efficiently be replicated in the NN architecture. For doing so, a specially designed first layer is created. The layer and NN architecture is shown in Figure 4.2. Here, one can see a specially designed first layer. This layer's design utilizes the domain-specific knowledge possessed by the underlying model. Namely, the BM applies scaling followed by a hyperbolic tangent function to account for saturation. This effect can be modeled directly in the first layer of the NN, as shown in the figure. As NNs are universal approximators, one might obtain a similar fit to the

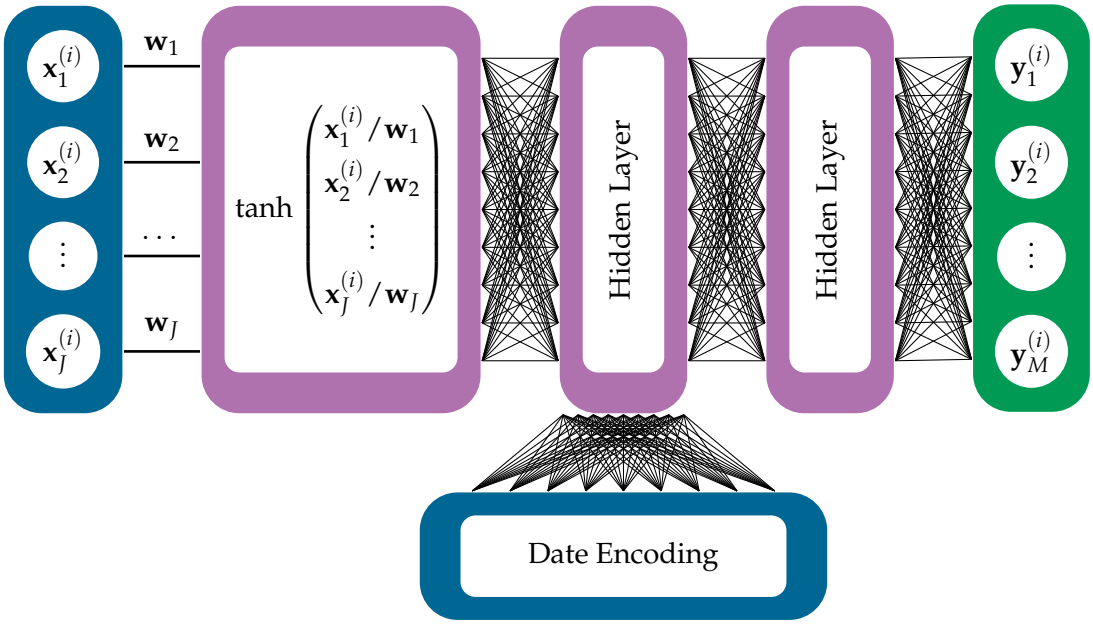


FIGURE 4.2: **The customized NN architecture used**. The figure shows how a single observation, $\mathbf{x}^{(i)}$, is processed by the NN. First, the input data is transformed by the *customized* first layer. This layer resembles the saturation component of the BM. Next, the data is processed by a set of fully connected hidden layers. Further, date encoding is provided to make the NN seasonally aware like the BM is. The final output for a single observation is a vector of predicted KPIs – one element for each posterior sample taken from the BM's posterior distribution. Each output $\mathbf{y}_m^{(i)}$ is an approximation of the BM's expectation over the posterior predictive distribution conditioned on a particular sample, $\mathbb{E}[Y \mid X, \boldsymbol{\theta}_m]$. The mean of this vector is then an approximation of $\mathbb{E}[Y \mid X]$ from (2.20). This approach avoids the need to evaluate the expensive double integral from (2.20) through this approximation. For this reason, the input optimization step in Figure 4.1 gains a significant speed improvement.

dataset using a NN without the domain-dependent first layer. However, one could argue that using the knowledge one has about model-specific details is beneficial, as it eases the approximation.

4.2.1 Making the NN Seasonally Aware

From the model in (4.1), the seasonal effect, S, plays an essential role in scaling the marketing effect of a spend at timestep t. Without S, the marketing effect of a commercial will be constant throughout the week, month, and year. In order to make the NN time-aware, a *date encoding* is provided as additional input to the NN, as shown in Figure 4.2. This encoding is concatenated to the spend data after the first layer has processed the spend. Date information can be encoded in several different ways suitable for NNs. The specific encoding is typically domain dependent, and section 4.4.3 elaborates on the encoding chosen for one of the experiments in this chapter.

4.3 The Loss Function for Optimizing a Mix-Marketing Plan

Having a NN fitted using $\mathcal{D}^{\text{NN}}$, this can now be used to generate an optimized MMP. To do so, let $\mathbf{X}_{T\times J}$ be a MMP spanning T days, covering J different insertions. The predicted KPI for this MMP is also a matrix, $\mathbf{Y}_{T\times M}$, one for each day and each posterior sample in $\boldsymbol{\theta}_M$. It requires a careful, continuous adjustment to the weight between optimizing the KPI and conforming to a prefixed budget, B. The following function is used to compute the loss for SGD to perform the input optimization

$$\begin{aligned}\mathcal{L}_{\text{P}} &= \sum_{t=1}^{T}\frac{1}{M}\sum_{m=1}^{M}\mathbf{Y}_{t,m}\\ \mathcal{L}_{\text{B}} &= \left(\text{B}-\sum_{t=1}^{T}\sum_{j=1}^{J}\mathbf{X}_{t,j}\right)^2\\ \mathcal{L} &= \pi\mathcal{L}_{\text{B}}-(1-\pi)\,\mathcal{L}_{\text{P}}\end{aligned} \tag{4.4}$$

Here, B is the fixed budget set by the user. π balances the weight between the two loss terms. π is dynamically tuned to *anneal* the budget loss until the budget has converged. Starting with a low value for π and slowly increasing it allows for the optimization to find an initially well-performing MMP without too much regard to the budget constraint. Slowly increasing π forces the optimization to eventually converge to a MMP which satisfies the budget constraint while attempting to uphold a high KPI.

4.4 Experiments

This chapter will conduct experiments in two different settings: an illustrative, low-dimensional example and an industry-level setting.

4.4.1 An Illustrative Example

The first example is a constructed, illustrative example with only two insertions. This experiment serves as a low-dimensional, proof-of-concept example that is easy to visualize. In this domain, the BM does not use a carry-over effect, nor does it have a temporal component. It is simply defined as

$$\begin{aligned}\boldsymbol{\alpha} &\sim \mathcal{N}_{\text{Half}}\left(\mathbf{1}_2/2,\mathbf{I}_2\right)\\ \boldsymbol{\beta} &\sim \mathcal{N}_{\text{Half}}\left(\mathbf{1}_2,2\mathbf{I}_2\right)\\ \sigma^2 &\sim \text{Gamma}\left(0,1\right)\\ f_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\mathbf{x}) &= \boldsymbol{\alpha}_1\tanh\left(\frac{\mathbf{x}_1}{\boldsymbol{\beta}_1}\right)+\boldsymbol{\alpha}_2\tanh\left(\frac{\mathbf{x}_2}{\boldsymbol{\beta}_2}\right)\\ Y &\sim \mathcal{N}\left(f_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\mathbf{x}),\sigma^2\right)\end{aligned} \tag{4.5}$$

where $\mathbf{1}$ denotes a column vector of ones, $\mathbf{I}$ is the identity matrix, and $\mathcal{N}_{\text{Half}}$ is the bounded normal distribution to prevent negative values as we assume a marketing campaign can never harm the KPI. We choose the ground true parameters such that

$$\left(\widehat{\boldsymbol{\beta}}_1 < \widehat{\boldsymbol{\beta}}_2\right) \wedge \left(\widehat{\boldsymbol{\alpha}}_1 < \widehat{\boldsymbol{\alpha}}_2\right) \tag{4.6}$$

This decision on the ground true parameters is to have two predictors that saturate at different levels, as illustrated in Figure 4.3. The figure shows that insertion x_1 saturates much faster than x_2, but it cannot yield the same level of KPI at a maximum level.

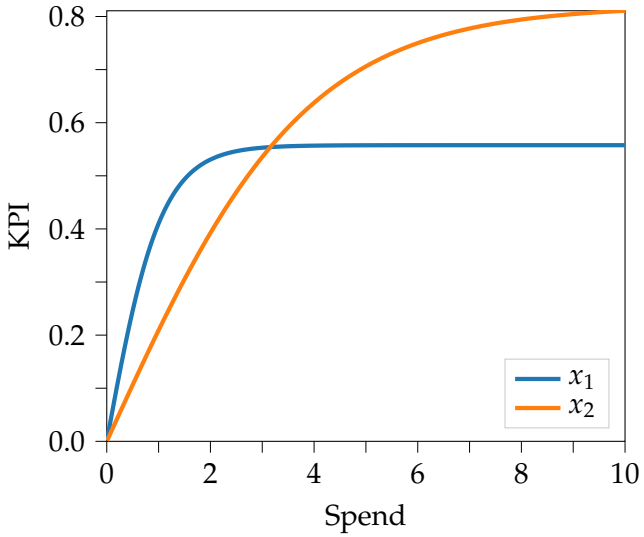


FIGURE 4.3: **The illustrative MMP domain.** The figure shows the KPI as a function of each predictor. For measuring the KPI for increasing values of $\mathbf{x}_1$ we set $\mathbf{x}_2 = 0$, and *vice versa*.

The task for the NN is to approximate the transition function $\mathbf{x} \in \mathbb{R}_2^+ \rightarrow \mathbf{y} \in \mathbb{R}_M^+$ where $\mathbf{y} = \left[\mathbf{y}_m = f_{\boldsymbol{\alpha}_m, \boldsymbol{\beta}_m}(\mathbf{x})\right]_{m \in 1,\dots,M}^{\top}$ and $(\boldsymbol{\alpha}_m, \boldsymbol{\beta}_m) \in \boldsymbol{\theta}$. Once the NN has been fitted to a subset of $\mathcal{D}^{\text{NN}}$ constituting the training dataset, the input optimization procedure is conducted on a set of different constraints. The results obtained here will not only demonstrate the overall method, but also shed light on potential discrepancies between the ground true solution and the solution obtained from the NN using the input optimization.

4.4.2 An Industry-level Example

The second experiment is performed on a large, industry-level, BM. This BM is used for generating optimized MMPs for a domain with ≈ 100 insertions. The model is AR(1) as presented in (2.23), for which the parameters are inferred using RStan [263] and *No-U-Turn Sampler* [121] based on an anonymous company's data. Thus, our results are directly applicable in a real-world setting to convert the company's data to value.

We compare the optimized MMPs generated using the BM versus the approximated NN. To extensively test the robustness of the NN, we generate multiple MMPs spanning a month with different budgetary constraints. Finally, we compare the computational time it takes to generate a MMP using the original BM to that of the NN.

4.4.3 Experimental Settings

The dataset on which the NN is trained consists of 100 000 and 10 000 000 examples for the illustrative and industrial experiments, respectively. This data is generated using the steps from List 4.1 with $\tau = 0.8$. Both datasets are split using a 10% validation and testing set. Min-Max scaling is applied as a preprocessing step to standardize the predictors to the range $[0, 1]$.

The NN, as illustrated in Figure 4.2, consists of the customized layer as described in section 4.2, followed by multiple hidden, fully connected, feed-forward layers. For the illustrative experiment, these hidden layers have sizes 500 and 200. The industry-level experiment uses a slightly more extensive network with layers of sizes 5000, 1000, and 200, respectively. Batch normalization [134] and dropout [119] is applied after each of the hidden layers to regularize and stabilize the network. Finally, these layers use the Leaky Relu activation function [180] and He initialization [116] on the weights.

In contrast, the customized first layer uses the tanh activation function. When initializing the training procedure, the weights of this layer are initialized such that $\mathbf{w}_j \sim \mathcal{U}(0.1, 0.9)$. This range is chosen such that $\mathbf{X}_{i,j}/\mathbf{w}_j$ stays within the non-flat regions of the tanh function to prevent $\partial\mathcal{L}/\partial\mathbf{w}_j \approx 0$. Further, after each iteration of back-propagation, any weight $\mathbf{w}_j < 0$ is clamped to 10^{-4} since it is assumed a spend can never harm the KPI.

The Adam optimizer [139] is used both for training and input optimization. The *learning rate* (LR) is set to 10^{-4} during training. The input optimization generally requires larger changes to the parameters than the training procedure. Therefore, the LR is increased to 10^{-3} for this action. In order to prevent overfitting the NN, ES [26, 99] with a patience of 40 is employed to trigger termination of the training process. In contrast, the input optimization is simply run for 5000 iterations. In this setting, the batch size is determined by the length of the desired MMP, T, whereas a batch size of 256 is used during training.

Finally, for the illustrative experiment, the ground true parameters are set as follows: $\widehat{\boldsymbol{\alpha}}_1 = 0.5$, $\widehat{\boldsymbol{\alpha}}_2 = 0.9$, $\widehat{\boldsymbol{\beta}}_1 = 1$, $\widehat{\boldsymbol{\beta}}_2 = 4$, and $\widehat{\sigma} = 0.1$. These settings were used for generating the training data needed to fit the posterior distribution of the BM.

Date-Encoding

In ML, there are different approaches one can take to encode time such that the representation is suitable for a NN to process. To make the NN temporal-aware,

the date-encoding used in for the industrial experiment operates at three different scales; weekly, monthly, and yearly. The weekly scale uses a *one-hot encoding* (OHE), the monthly scale uses a linear encoding, and finally, the yearly scale uses a cyclic encoding. OHE is used to model the day of the week because, in this domain, Monday is not necessarily more closely related to Sunday than Thursday is. The OHE does not scale well and increases the risk of overfitting by the NN. For this reason, OHE is only applied on a weekly scale. To encode the day of the month, a simple linear encoding is designed. This encoding is chosen because days at the beginning of the month are closer related than days towards the end of the month and *vice versa*. Finally, to represent various seasons of the year, a cyclical temporal encoding with a period of $\phi = 365$ is used. This is such that $[\sin(2\pi t/\phi), \cos(2\pi t/\phi)]^\top$ is a two-dimensional encoding of day t for the time of year. A limitation of our applied encoding scheme is the fact that it does not reflect changes across years. The impact of a MMPs can indeed change over the years, and an extension of our approach should take this into account. Yet, as our generated MMPs are only placed within the same year for which the training data is generated, this limitation is not an issue in this application. Future research could apply Time2Vec [149], which introduces inferable parameters as part of the date-encoding process.

4.5 Results

To assess the performance of our approach and the accuracy of the approximation, the input optimization was repeated for a range of budgets using the NN. For each experiment, the loss/gain in KPI is compared to a reference point. Afterward, we evaluate the elapsed wall-clock time when optimizing a MMP for the industry-level experiment, as gaining speed is the primary motivator behind the proposed method.

4.5.1 KPI Loss and Budget Deviations

For each experiment, the input optimization was carried out on a set of budgetary constraints $\mathcal{B}$ by setting $\mathrm{B} = b$ in (4.4) $\forall b \in \mathcal{B}$. Using the NN, this repeated process yields a set of optimized inputs, denoted $\mathcal{M}_{\mathrm{NN}}$. Subsequently, the budget deviation and potential change in KPI were measured by comparing m to a reference point $\forall m \in \mathcal{M}_{\mathrm{NN}}$.

For the illustrative experiment, this reference point is the ground true solution that strictly satisfies the budget constraint. For the industry-level experiment, the same reference point is not accessible, as the ground true parameters are unknown for this real-world, high-dimensional example. Instead, the reference points are in the set $\mathcal{M}_{\mathrm{BM}}$. This is a set of MMPs obtained using the same optimization process as with $\mathcal{M}_{\mathrm{NN}}$, but using the BM and its full posterior distribution instead.

Illustrative Experiment

Taking the illustrative example from section 4.4.1, let $\mathcal{B} = \{1/2, 1, 2, 4, 6\}$. Figure 4.4 visualizes the KPI landscape ($\mathcal{L}_P$ from (4.4)), with the curvature indicating the KPI calculated using the posterior of the BM.

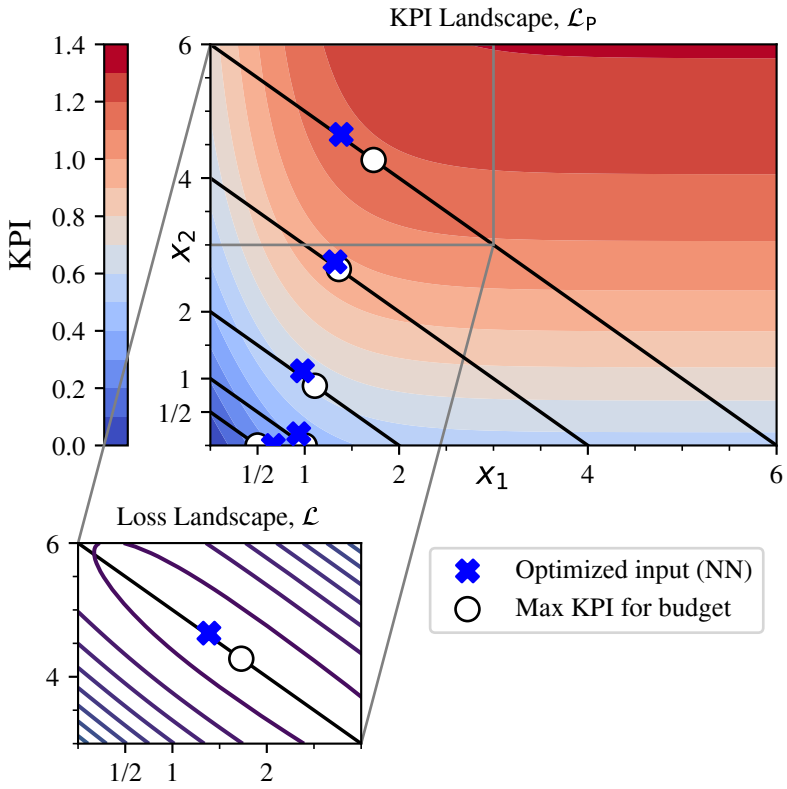


FIGURE 4.4: **The KPI-landscape for optimizing a simple MMP**. The curvatures indicate KPI. The warmer the color, the higher the KPI. The solid line is where the $\sum \mathbf{x} = b$, where b is some budget. The circle indicates the optimal solution along this line that maximizes the KPI. The blue crosses indicate the solution generated by the NN for the related budget.

All possible combinations of $\mathbf{x}_1$ and $\mathbf{x}_2$ are computed such that $\sum \mathbf{x} = b$ for each $b \in \mathcal{B}$. These combinations are shown as a solid line in the figure, one for each budget. The white circle on each line indicates the *optimal* solution for the corresponding budget, this being our reference point. In the figure, each blue cross represents an element in $\mathcal{M}_{NN}$.

As the elements in $\mathcal{M}_{NN}$ are a result of the loss function and SGD, in some cases, these deviate slightly from the corresponding reference point. This discrepancy is due to the weighting between complying with the budget and maximizing the KPI. For instance, in the case of $b = 1/2$, the budget deviation is pronounced, and the solution in $\mathcal{M}_{NN}$ yields a higher KPI than the ground true, as $\mathcal{L}_B$ does not form too tight of a bound on the spend. In addition, for $b = 6$, the loss landscape of $\mathcal{L}$ is a plateau for a wide range of solutions. As a result, despite the relatively large

euclidean distance between the solution in $\mathcal{M}_{\text{NN}}$ and the ground truth, the loss in KPI and budget deviation is $\ll 1\%$ when comparing the two.

To illustrate this plateau, the zoomed section of the figure is a snippet of the loss landscape. The curvatures show the loss function $\mathcal{L}$ with $b = 6$ and indicate the flat plateau in the center. This plateau is shifted marginally toward the upper right corner as the gain in KPI pushes toward an increase in spending. In this simplistic setting, we let $\pi = 1/2$ in (4.4) to equally weight the two terms in the loss function.

Industry-level Experiment

Next, we assess the performance of our approach in the high-dimensional, industry-level example.

For this experiment, the set of budgetary constraints $\mathcal{B}$ is computed relative to the historical daily spend of the particular company $\widehat{\mathbf{X}}$. The intuition behind this set of budgets is to cover both relatively low and high bounds. Let

$$\begin{aligned} \mathbf{d} &= \left[d_t = \sum_{j=1}^{J} \widehat{\mathbf{X}}_{t,j} \right]^{\top}_{\forall t=1,\dots,T'} \\ \mathcal{B} &= \left\{ \frac{n}{4} TF_{\mathbf{d}}^{-1}(.75) : n \in \mathbb{N}_1^+ \wedge n \leq 16 \right\} \end{aligned} \tag{4.7}$$

where $\mathbf{d}$ is a vector of daily total spends based on the company's historical data for a period of T' days. F^{-1} denotes the inverse cumulative distribution function over $\mathbf{d}$, and T is the number of days the optimized MMPs should span – in this case, a period of one month. In effect, the result is a set of budget constraints $\mathcal{B}$ spanning from $.25\times$ to $4\times$ a typical budget for a particular company. Thus, $\mathcal{M}_{\text{NN}}$ and $\mathcal{M}_{\text{BM}}$ each consist of 16 MMPs, one for each $b \in \mathcal{B}$.

An element-wise comparison of $\mathcal{M}_{\text{NN}}$ and $\mathcal{M}_{\text{BM}}$ is shown in Figure 4.5. Here, the horizontal axis indicates the budget scale used, i.e., $n/4$. The blue line indicates the total spend of $\mathcal{M}_{\text{NN}}$ relative to $\mathcal{M}_{\text{BM}}$ in percent, denoted $\Delta B\%$. Similarly, the orange line indicates the predicted KPI of $\mathcal{M}_{\text{NN}}$ relative to $\mathcal{M}_{\text{BM}}$ in percent, denoted ΔKPI%. ΔKPI% is computed using the BM. The dotted lines indicate the mean of the corresponding metric as indicated by the color. As indicated by the figure, ΔKPI% lies in the range $[-1.7; 2.1]$ with a mean of $\overline{\Delta\text{KPI}\%} = -0.46\%$, i.e., an average loss of 0.46% in KPI using our method. When the predicted KPI of $\mathcal{M}_{\text{NN}}$ surpasses that of $\mathcal{M}_{\text{BM}}$, it is caused by a well-performing MMP combined with a higher total spend. As with the illustrative experiment, the total spend of an optimized input surpasses the allocated budget in some cases. Forcing the generated MMPs to comply strictly with the budget can be done by adjusting π in (4.4). Having too strict a budget violation penalty in the loss function prevents the SGD process from reallocating spend sufficiently and thus yields sub-optimal solutions. For this reason, it was decided not to further increase the impact of $\mathcal{L}_B$ on $\mathcal{L}$. On average, the budget deviation of $\mathcal{M}_{\text{NN}}$ was $\overline{\Delta B\%} = 0.43\%$.

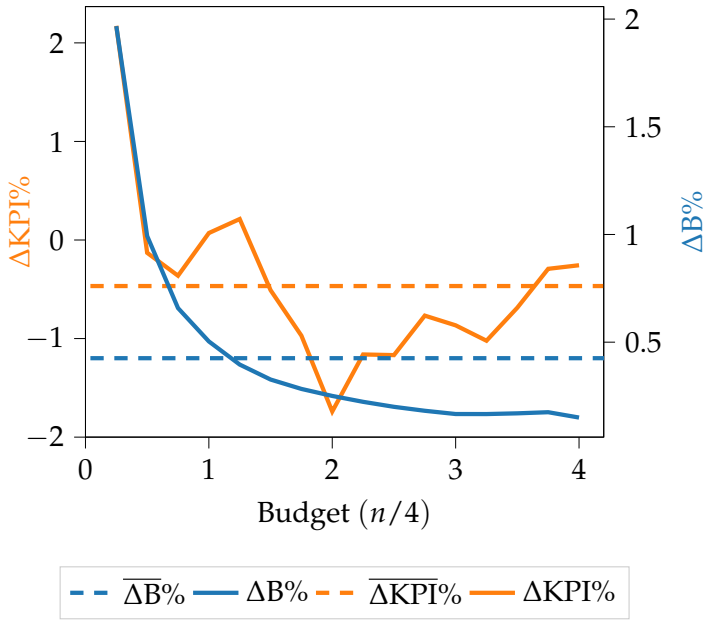


FIGURE 4.5: **The relative difference in expected KPI between the two methods.** The KPI of the MMP generated using the NN versus that of the BM (orange line). The difference is calculated using the BM. The left y-axis indicates the relative difference in KPI, and the right y-axis indicates the relative difference in spend (blue line). The dashed lines indicate the mean of each metric.

4.5.2 Optimization Times

Speed improvements being the key reason for the proposed method, the wall-clock time elapsed when generating a MMP using the NN versus the BM is evaluated next. This measurement is repeated for three different configurations. These configurations differ only in the length of the period of the MMP to be optimized. The first spans only a single week, the next one month, and the last a full quarter. In Table 4.1, one can see the measured timings.

TABLE 4.1: **A comparison of the time it takes to generate an optimized MMP using the BM versus the NN.** The experiment is done for three different periods. The results show how the NN generates optimized MMPs more than one magnitude faster than the BM. In addition, the results suggest how this method scales better as the period spanned by the MMP increases.

MMP Period	**Input Optimization Time (S)**	
	NN	BM
Week	1.2×10^1	6.2×10^2
Month	1.2×10^1	6.6×10^2
Quarter	1.4×10^1	9.2×10^2

From the table, the significant difference in the elapsed wall-clock time is clear. On average, the NN provides MMPs $\approx 5.7 \times 10^1$ times faster compared to the BM.

Further, while there is a clear correlation between the elapsed time and the length of the period for the BM, this correlation is weaker for the NN.

These significant improvements in elapsed wall-clock time, combined with the high-quality solutions as demonstrated in Figure 4.5, render the task of optimizing a MMP a cheap operation that can be repeated quickly for various periods and constraints.

For reproducibility, it should be noted that the experiments are run on two separate machines. The BM is executed on a compelling Amazon m5.12xlarge instance with a 48-core processor and 192 GiB of RAM, whereas the NN experiments are executed on a less potent Intel Core i7-6700K processor with 4 cores, 48 GiB of RAM, and an Nvidia Geforce RTX 2080 Ti. Even if the experiments were executed on the same machine, the computations would still occur on two separate computational units, the NN running on the GPU and the BM on the CPU. Thus, we did not further pursue executing the experiments on the same instance.

4.6 Discussion

With the significant improvements in elapsed wall-clock time for the input optimization as reported in Table 4.1, one might wonder what causes our method to outperform the original so significantly. The answer lies mainly in the construction of the BM itself. Despite the extensive use of matrix multiplication in the BM, some operations cannot be done without recursion. For the BM to estimate the total effect of a marketing plan spanning T days, it needs to sum the total marketing effect of each day. In (4.3), we showed how the $\text{Effect}_{\text{total}}$ of a spend at a single day is an infinite sum over the future effect but can be reduced to the sum of a geometric series with a finite solution. However, this computation will have to be carried out $\forall t \in 1, \ldots, T$, each t requiring a complete iteration over the BM for all M posterior samples. As long as T does not exceed the number of CPU cores, this can be run fully concurrently, which is why the elapsed time for a MMP spanning one week is virtually identical to one spanning a whole month due to the 48-core processor. The same limitation is not present for the NN due to the architecture of the NN and the electronic design of a GPU.

Finally, it is important to note that our approach is not limited to the specific BM from (4.1) but is generalizable to other BMs. For instance, introducing a synergy effect in (4.1) would require changes to the BM, while the NN would remain unchanged.

Accelerating the generation of optimized MMPs allows for novel applications. Now it becomes computationally feasible to generate thousands of optimized MMPs in a computationally efficient manner. For instance, one could generate MMPs for a range of various budget constraints. These constraints could be either on an aggregate level or on individual insertions applying to one or more days. Next, having

found an appealing configuration, the user can either use the MMP directly or run the optimization using the BM with the identified desirable settings.

In addition, in a high-dimensional domain such as the real-world, industrial setup as described in section 4.4.2, there is potentially more than a single high-performing MMP for the same budget and period. With the gain in speed, explorative algorithms like MAP-Elites [286] can be used to find quality diversity – a set of competent, different solutions to the same problem.

4.7 Chapter Summary

In this chapter, the general approach from Chapter 3 was extended to fit the domain of mixed marketing. The experiments indicate the approach allows for rapid generation of optimized MMPs. The method was tested on two experiments: a low-dimensional illustrative example and a high-dimensional, real-world, industry-level BM fitted on real company data. Each experiment was evaluated on a wide range of constraints of the input optimization. The resulting solution vectors exhibited only minor deviations in terms of budget violation and expected KPI compared to those obtained with the BM. These results indicate a robust method that gives consistently well-performing solutions. Further, for the industry-level experiment, our method yields MMPs up to 65$\times$ faster than using the BM alone and scales better for MMPs spanning longer periods of time.

Part III

Exploring Human-AI Interaction through Games

Chapter 5

Human-AI Interaction in Games

In the previous chapter we build the foundation for addressing a key challenge with human-AI systems like that of BW7. The challenge at hand was the speed at which the AI can generate candidate solutions for the human to evaluate. The chapter concluded with a significant speedup in the generation of optimized MMPs using a NN.

These MMPs were generated based on an approximation to an industry-grade BM used by BW7. The quality of the generated MMPs was measured against the BM itself. This measurement exposes the accuracy of the approximation and, by extension, the durability of the proposed method as a whole.

If the approximation is good, the two models will agree on the expected KPI for an arbitrary MMP and generate the same MMP for a given budget. That is, they will have the same mapping from marketing plan to KPI. The results showed that the BM's predicted KPI of the MMPs generated with the NN was on par with that of the MMPs created by the BM. In effect, these solutions comprise the same tendencies and spending structures. Hence, using the NN as a proxy yields MMPs that are essentially *identical* to those generated with the BM – but composed much faster.

As mentioned in the introduction, another key challenge with human-AI systems is the human not being able to relate to the candidate solutions of the AI. This is also the case for the AI-generated MMPs created by BW7's BM. The AI-generated plans look nothing like the plans manually composed by the employees. Since the MMPs generated with the NN and BM are similar, the method from the previous chapters does not address this, and therefore, the gap between human and AI solutions remains.

A possible reason for this gap lies in the *mental models* the marketing employees have developed of the marketing landscape. The introduction already touched upon the concept of mental models. In summary, a mental model is a dynamic, internal representation a human develops when engaging with an external reality [162]. This internal representation provides cause-effect relationships for the user and is employed during decision-making when interacting with external systems [142]. These models have their roots in cognitive research that has established that they are developed over time through learning and interactions with a system [208]. The

models are inherently dynamic, inconsistent, and incomplete conceptualized representations [142]. Hence, it is unreasonable to expect marketing employees to have a fully accurate mental model of the marketing landscape.

Another apparent reason for this human-AI gap is the simple explanation that the AI models are incorrect and that research should strive to improve these. While this claim is likely correct, it is unrealistic to have a completely accurate model of how a marketing campaign affects a company's KPI [31] just as it is unrealistic with a mental model. In fact, research has found that solely pushing for improved AI accuracy can hurt the collaborative human-AI team performance [15].

Instead, improvements to the human-AI team performance could lie in better mental models of the error boundary of the AI. An accurate model of the error boundary provides the user with insights on when not to trust the AI recommendation [14]. Since these models are evolved through interactions with the system, additional human-AI-human iterations are required.

Liu and Heer [172] found that latencies as small as 500 ms from user input to model output significantly decrease the user's interaction and exploration with data processing tools. The achieved 65× speedup of the ML tool as presented in the previous chapter is still far from providing real-time feedback. It still takes significantly longer than the 500 ms threshold studied by Liu and Heer [172] to generate optimized MMPs. That said, the reduced ML latency provides a more seamless HAII experience and increases the likelihood of maintaining the user's attention for longer periods of time [205]. In effect, users are more likely to reiterate experiments with the ML tool. This, in turn, helps the users develop mature mental models of the ML algorithm.

Delay in the human-AI-human feedback loop is not the only challenge with HAII. There is a growing recognition that compared to traditional interactive systems, AI-infused products impose additional challenges (e.g., technical barrier, low interpretability) to the current UX design process [71]. Furthermore, new interdisciplinary research areas have emerged around topics such as XAI [23, 230, 275, 318], ethics & fairness [38, 71, 123], and ML as a design material for UX [304, 305].

To shed light on how BW7's platform can improve other aspects of the HAII than latency, one could conduct user studies on how the platform is used in practice. However, there are several unappealing challenges to this. First, BW7 has a limited user base. Relatively few marketing employees engage with the platform on a daily basis. In addition, these users are already heavily experienced with the platform. Consequently, studies on how their mental model develops from scratch would not be possible. Once a mental model is built, it has proven surprisingly difficult to modify it – even when given evidence of an incorrect model [162, 280]. For these reasons, uninitiated users are required.

Acquiring new customers, and thus new users to the platform, is an ongoing task carried out by BW7. However, the onboarding process takes a substantial amount of time, since each customer has to share confidential data with BW7 to

get a customized BM created. Further, adding one or two new customers to BW7 is not enough to provide a substantial user base. Finally, two marketing employees who work in different domains, have differing educational backgrounds, or are geographically distant are likely to have substantial differences in their mental models of the marketing landscape. This further complicates such a study and calls for a large user base to identify trends.

Therefore, to understand how the HAII with BW7's platform can be improved, other applications of HAII are studied instead. Thus, this part has HAII at its core and aims to improve our understanding of how teams of people engage and collaborate to develop mental models of a user-facing ML system.

5.1 Why Games?

In AI research, there is an extended history of using games as a rich domain to motivate algorithmic advancements. Salient examples include *Chess* in the era of "Good Old Fashioned AI" (GOFAI) [42], *Go* [257], classic Atari video games [196], and even the popular AAA game *StarCraft* [209, 288]. The advances in game AI in turn opened new design spaces of player experience in research [185, 187, 282, 308, 317] as well as commercially released games and game engines (e.g., *Versu* [76], *Left4Dead* [283] and *Civilization VI* [84]).

However, with few exceptions, games have only recently started to be used as a serious domain for HAII research. For instance, Gomme and Bartle [98] used strategy games to study players' expectations for what they consider to be a worthy AI-controlled opponent. Along those lines, several researchers proposed using games and playful experiences to help designers and users learn AI [81, 200, 219].

Most existing work has focused on high-level metaphors (often referred to as "design patterns") of how players and designers can interact with AI. For instance, Treanor et al. [278] derived nine patterns based on what players do: AI as role-model, trainee, editable, co-creator, adversary, villain, or spectacle, and whether AI is visible or guided. Cook et al. [49] further examined design patterns in procedural content generation (PCG)-based games and derived different AI design patterns. In the context of assisting the game development process, Riedl and Zook [235] proposed that AI plays the role of actor, designer, and producer. While the above work provides a critical starting point for our work, they are "meant to be a tool for thinking about creating AI-based games, rather than serve as a comprehensive taxonomy of methods" [278]. Finally, Guzdial et al. [112] used the taxonomy of friend, collaborator, student, and manager to describe the different interaction metaphors for how game designers interact with an AI-based game level editor. The following builds on this tradition of using human-human interaction as metaphors to structure the interaction between humans and AI.

In addition, there is a significant body of work in games research to understand player experience [2, 60, 62, 178, 202]. For example, the *game engagement*

questionnaire [36] is a widely used instrument for measuring player engagement, although recently it has been approached with increasing criticism [165]. Other notable frameworks include game involvement [41], game usability [62], and design heuristics [178]. While these frameworks are useful to improve the general player experience, they do not have sufficient focus on the interaction between players and AI to guide the HAII design of games.

5.2 Player-AI Interaction in Games

The following proposes the first set of guidelines to design and evaluate player-AI interaction. The insights presented here led to our paper "Player-AI Interaction: What Neural Network Games Reveal About AI as Play" [320]. While not being the main part of this work, the key findings are summarized here.

These results lead to the new construct of *player-AI interaction* to highlight how people interact with AI in the context of play, especially through computer games.

To provide an overview of existing work in this area, our paper conducted the *first systematic review of player-AI interaction* in the scope of *Neural Network games* – computer games in which players interact with a NN as part of the core gameplay.

NN-based games were chosen for three key reasons. First, given the wide adoption of AI in games, it was necessary to constrain the focus to NNs instead of all types of AI. Second, the advancements to the AI-infused system of BW7 as proposed in Part II are based on NNs; therefore, it is natural to study HAII with such tools. Third, NN-based games provide insights into some of the most pressing open problems in HAII. For example, NNs are notorious among UX designers because of NNs' low interpretability of the underlying process and the frequent unpredictability of their outcomes. Studying NN games can thus provide valuable information on how UX designers have to work with these challenges.

5.2.1 A Systematic Review of Player-AI Interaction

The systematic review started out by identifying existing NN-based games satisfying a set of inclusion criteria.

The main criteria was to include computer games wherein players can interact with a NN as part of the core gameplay (i.e., gameplay loop). Here, the definition of core gameplay as "the set of actions the player iterates on the most while playing the game [and which] should directly influence the outcomes of the game" ([110]) was used. Work with no clear win condition *and* no clear feedback on how player interaction with the AI impacts the game was excluded, as these games lack the basic elements for meaningful player-AI interaction. Notice that sandbox games with clear feedback to player interaction are included [11, 104, 115, 254, 290]. For the same reason, games where the NN did not interact with players were also excluded (e.g., ML agents that can automatically play games [259]). Finally, digitized versions

of traditional board/card games were excluded (e.g., *Chess*, *Go*, *Poker*) to focus on computer games. Future research is needed for investigating player-AI interaction in traditional games.

Based on these criteria, a dataset with 38 games was collected. These games were identified after searching at two popular web gaming portals – *Steam* and *itch.io* – and a widely used game AI book, *Artificial intelligence and games* [306].

After careful evaluation of these games, a framework to classify these games was developed. This framework has three components at its core: a key technical characteristic of the NN, interaction metaphors, and the visibility of the NN in the *user interface* (UI). Each component is elaborated in the next section.

Characteristics of NN

From the technical point of view, the collected dataset coveres a wide range of varieties. A key technical feature is associated with different gameplay characteristics – namely, whether the player-AI interaction can modify the behavior of the NN itself. This led to the term *online* and *offline* learning games being used as part of the framework for classification. In online learning games, the network is (further) trained as the player interacts with it. Therefore, these games can adapt to individual players' actions in real-time. Offline learning games, on the other hand, are shipped with fixed NNs and are not adaptive in the same way. However, offline learning games have the advantage of handling more complex user input, such as natural language [83, 103, 290].

Interaction Metaphors

Critical AI studies revealed the importance of metaphors to AI [4, 186, 316]. Our analysis found four interaction metaphors that provide familiar structures for players to interact with the AI: NN as *Apprentice*, *Competitor*, *Designer*, and *Teammate*. This finding is consistent with recent work in the game design literature. Based on their expert knowledge and intuition, game developers discuss how interaction metaphors (often referred to as "design patterns") have been used in game design [49, 278] and in game production [235]. The analysis summarized here extends the existing literature by conducting the first empirical work that uses deep qualitative analysis to analyze the interaction metaphors.

The largest portion of NN games (34%) adopt what this work calls **Neural Network as Apprentice**. In these games, the player interacts with the NN as its mentor, and the focus of the gameplay is *how the player changes the NN over time*. The player's mentoring of the NN can be achieved by providing direct feedback to the NN's behaviors [11, 104, 155]. For example, in *Creatures*, the player provides positive feedback (petting) when the NN-controlled creature displays desirable behavior (e.g., eat when hungry) and punishes it (slapping) for the opposite. A second way the player can mentor the NN is by configuring the right training setting for it [39, 83, 150, 222].

The gameplay afforded by this interaction metaphor focuses on getting the player to train the NN.

Another interaction metaphor used by the collected NN games is **Neural Network as Competitor**. The key characteristic of this group, consisting of 26% of the games, is that player-AI interaction is adversarial. For example, in *Supreme Commander 2* [228], the player fights an NN through their respective army platoons. As the player customizes their army, the NN weighs the player's unit composition against its own and makes tactical battle decisions, such as how its army will respond, which enemy to target first, and when to retreat. The NN can exploit players who are over-reliant on a single strategy and counter the player in order to create an evolving challenge [13, 86, 87, 198, 207, 228, 267]. In these games, the NN counters the player during gameplay, thus encouraging them to adapt and try new strategies. A key distinction in this category is that the NN learns the player's actions to create a more difficult challenge for the player to overcome.

For 21% of the games, the interaction metaphor **Neural Network as Teammate** was used. Neural Network as a Teammate happens if the interaction between the player and the NN is structured like those between colleagues. In these games, the player and the NN work together toward a shared goal. For example, in *Evolution* [66], players and the NN create a stick-figure-like creature together. Players assemble the creature by placing bones, muscles, and joints in different ways. The NN takes the player's creation and improves it through evolving it over many iterations. This interaction creates a collaborative cycle between the player and the NN. A unique characteristic of this interaction metaphor is that the player and the NN have complementary skills. Both are needed to complete the game objective.

The final 19% of the games used the **Neural Network as Designer** metaphor. In these games, the NN acts as a creator and the player as its client. The NN generates new content [168, 290] or customizes content based on the preferences of the player [115, 237], usually determined passively through players frequently interacting with a particular game element. For example, in *Petalz* [237], players arrange and nurture a balcony of flowers, which are generated by an NN. The NN generates each flower (shape and color) based on the player's selection of flowers to breed or cross-pollinate. The NN extends the game's playability by creating flowers that match the preferences of the player. Notice that compared to NN as Teammate, the player here generally has less well-defined goals to accomplish with the NN.

Visibility of Neural Network in Core User Interface

A significant number of games (26%) foregrounded the existence of the NN through what this work calls **NN-Specific UI**. These UIs highlight the presence of the NN during core gameplay through linguistic features (e.g., using the term "neural network" [21, 67, 285, 287]). For instance, *How to Train your Snake* describes each NN-controlled snake as "...hooked up to a Neural Network" [21]. Some games use visual features (e.g., visualizing the underling NN [21, 39, 211, 285, 287]). In *iNNk* (Middle,

FIGURE 5.1: From left to right, we display *Neat Race* [211] categorized as *NN-Specific*, *iNNk* [287] categorized as *NN-Specific*, and *Blitzkrieg 3* [207] categorized as *NN-Limited*.

Figure 5.1), the word "neural network" is prominently featured in the core game UI along with the NN's confidence meter. More interesting, some games visualize the parameters of the NN training algorithm to make the training process playable. For example, in *Neat Race* [211] (Left, Figure 5.1), the game visualizes the NN's internal structure (bottom left of the screenshot) and displays its parameters as sliders (top right).

The majority of the collected games (40%) used **NN-Limited UI**. These games acknowledge the presence of the NN in the game, but only through non-essential UI, such as using technical terminology in tutorials [104, 266, 290], menus outside the core gameplay loop [67, 150, 168, 267], or explicitly referring to the NN only in title screens [103, 290]. For instance, *Blitzkrieg 3* [207] is a WWII strategy game where players build and command a variety of units to defeat the opposing NN-controlled enemy. The game's opening screen (Right, Figure 5.1) personifies the NN as an evil-looking person with the text "Meet Boris, a neural-network AI you can fight against..."

Finally, 34% of the games used **NN-Agnostic UI**, which does not reference the NN. By masking the NN, these games maintain the narrative immersion of the game worlds without revealing the algorithms used to build them.

5.2.2 Key findings

An interesting insight is that the strongest designs for making NNs more interpretable come from simulation games where the player can tweak different training parameters. In these cases, even though the names of the parameters are sometimes too technical for players without an AI background, the behavioral change feedback given by the NN through different iterations of trial-and-error gameplay helps the player develop intuition. In other words, most games in our dataset manage to reframe the difficulties of interacting with an NN as a puzzle and thus make it more engaging.

This finding is in accordance with the introduction of mental models from the beginning of this chapter. As the players mature their mental model of the AI through iterative interaction, they gain a better understanding of the complex system. This is in spite of the players not having a theoretically founded technical understanding of the different training parameters that are tweakable.

Figure 5.2 visualizes the result of applying the above definitions to classify each of the 38 studied games. In the figure, each black circle indicates a game. The raw data of each classified game can be found in Table B.1, located in Appendix B. As indicated by the figure, the games are distributed fairly non-uniformly. Instead, there are clusters of popular applications for NNs in games for player-AI interaction. For example, the most popular combination (6 games) is having a NN as an apprentice combined with an online learning NN and a NN-Specific UI. Meanwhile, there are several underexplored areas within this framework. Some combinations contain no games at all, while others are represented by only a single instance. One explanation for this clustering is the possibility that some combinations of NN, games, and player-AI interaction are easier to make function than others.

To infer what NN games can tell us about HAII, we used the human-AI design guidelines proposed by Amershi et al. [8]. It is the most recent and comprehensive manner in which the design for HAII is documented thus far by the *human-computer interaction* (HCI) community. These guidelines were applied to the dataset with the 38 identified NN-based games. In summary, our study ([320]) led to the proposition of the following *design considerations*:

Use flow to structure the learning curve of human-AI interaction. For many users, interaction with AI can be overwhelming, especially when they encounter unexpected output from the algorithm. One important lesson from our study is that the concept of flow [55], widely used to balance game difficulty and player engagement over time, can be useful in designing HAII. The use of flow can be helpful in structuring how to gradually expose users to different AI features (see also [53]).

Incorporate enhanced discovery-based learning. Many games in our analysis, especially simulation games, offer discovery-based learning [7] with mixed success. Since players come with different background knowledge and needs, explicit instruction for AI is challenging to design. Discovery-based learning offers players the opportunity to play around with the NN at their own pace and observe the consequences of their actions on the NN and the game world. However, most NN games in our dataset offered very little scaffold, making it difficult for players without a technical background to succeed. We suggest that UX designers use enhanced discovery-based learning and provide feedback, worked examples, scaffolding, and elicited explanations to further assist their users.

Extend the invitation to play. Finally, for researchers and designers interested in exploring new forms of HAII, we believe offering users an invitation to play can unleash their imagination and empower them to explore new ways to interact with even the same technology. As we can see from *Hey Robot!*, the magic circle of play turns the smart speaker user from the seeker of information to the provider. The

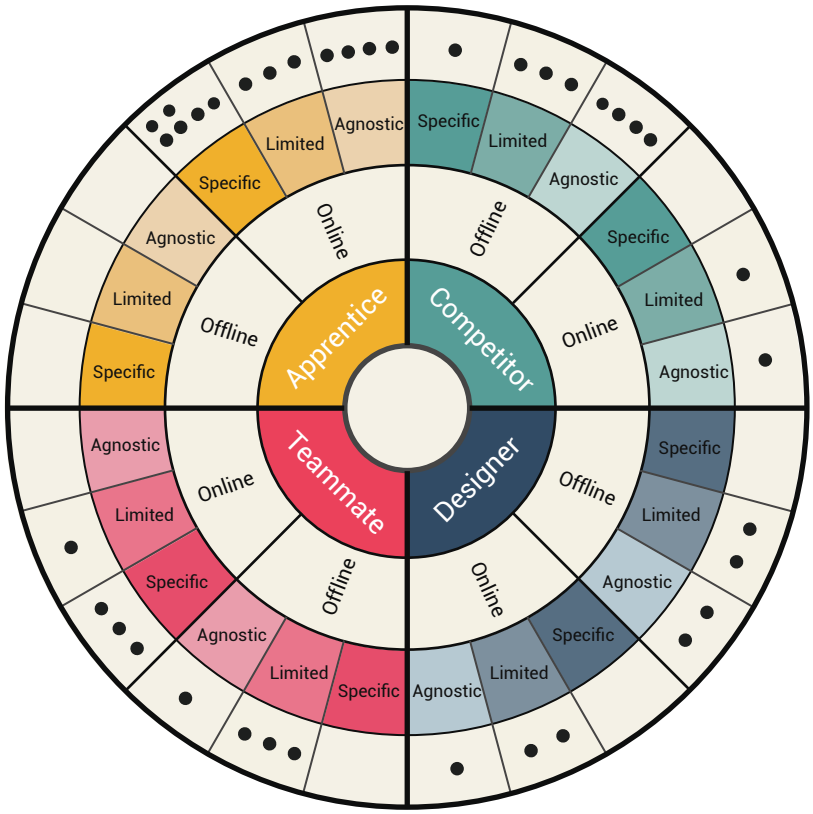


FIGURE 5.2: Distribution of the 38 NN games categorized by interaction metaphor, online/offline learning, and UI visibility. Each black dot represents one NN game.

voice assistant's inability to understand user command/intent is transformed from failure to perform to the source of fun.

5.3 Chapter Summary

This chapter has studied player-NN interaction in games. The results and insights gained here are a summary of our findings in the paper "Player-AI Interaction: What Neural Network Games Reveal About AI as Play" [320]. The contributions of this chapter are threefold: *1)* a framework for classifying various forms of player-NN interaction settings in games, *2)* an overview of existing games and how these are distributed based on this framework, and *3)* some key design considerations to bear in mind when designing such NN-based games. These considerations can be taken into account for other HAII applications outside the field of games as well.

Next, we will present a NN-based game developed as part of this work. This game will serve as a case study for player-AI interaction that will probe how emerging mental models can alter the intended dynamics of the player-AI interaction. Section 6.8 will complete the circle by highlighting how these key insights can inspire

augmentations to the HAII experience of BW7's platform. These insights will constitute the base for the general approach taken in Part IV.

Chapter 6

Case Study: iNNk – A Multi-Player Game to Deceive a Neural Network

The previous chapter developed a framework for classifying NN-based games. The development of said framework revealed underexplored areas in the design space of Player-NN interactions. One explanation for these underexplored areas is the possibility that these areas represent constellations that are more difficult to make function properly. One such constellation is having an offline learning NN as a competitor with a NN-Specific UI (see Chapter 5). Figure 5.2 discloses that only a single game has these properties. That game is *iNNk*, which came to life as part of this work. INNk has some novel characteristics allowing for unique player-NN interactions.

This chapter first introduces the game and its mechanics. After presenting the fundamentals of the game, a summary of the main strategies the players developed during playtesting is given. These insights reveal that player strategies were developed that broke the balance of the game. Therefore, subsequent to the introduction of the game is the concept of adversarial strategies on NNs and possible defense mechanisms. These defense mechanisms are tested out in iNNK to measure their effect on addressing human-generated adversarial examples and thus regaining a balanced gameplay.

The general motivation for this part is to study NNs through games in order to guide improvements to the HAII in BW7's platform. Therefore, to see how the contributions of this part can guide such enhancements, the chapter concludes by revisiting the HAII of BW7's AI-infused system. This revisit discloses some surprising similarities between BW7's platform and the player-NN interaction in iNNk.

6.1 iNNk

iNNK[1] is a web-based multiplayer drawing game in which two or more people play together to deceive a NN. To win the game, the players need to successfully communicate a secret code word to each other through drawings without it being deciphered by the NN. Each game is composed of five short rounds, allowing players to

[1]Currently hosted on this page: innk.gaims.dev:3167

experiment with different drawing strategies. Whoever (i.e., humans or the NN) has the most points at the end of the five rounds wins the match. For the human players to be successful, they must develop drawing strategies that can only be interpreted by their human teammates and not the NN.

Players are assigned one of the two roles during the game: the Sketcher and the Guesser (Figure 6.1). The Sketcher is tasked with drawing something based on the

FIGURE 6.1: left: A screenshot of the Sketcher's interface. In the white canvas, the Sketcher draws to communicate the secret code word, indicated above the canvas (*cat*). The NN's guess and its confidence are on the upper right corner. Right: A screenshot of the Guesser's interface. Guessers can type in their guess at the bottom.

code word assigned by the game. The goal is to draw the code word in such a way that the human Guesser may be able to accurately interpret the code word before the NN. In general, if the Sketcher draws something that is prototypical, it would be straightforward to other human players. But, it will also be easy for the NN to guess.

The Guessers are tasked with entering their guess of the code word based on the Sketcher's drawing before the NN guesses correctly. The NN always plays the role of a Guesser, and its goal is to decipher correctly first. While there is no penalty for wrong guesses, the human Guessers must be mindful of their guesses. As described below in section 6.1.3, the NN takes into account all previous attempts at the code word. Human Guessers' wrong guesses will increase the NN's chance for correct interpretation.

The game is structured around five 30-second rounds. If either the human team (consisting of one Sketcher and at least one Guesser) or the NN guesses the code word correctly within 30 seconds, the respective side wins the round. Otherwise, the round ends in a draw. Whichever team gets the most points at the end of round five wins the match. A duration of 30 seconds was chosen because it provides enough time for players to draw (or interpret) the code word while being quick enough to encourage frequent moments of surprise and failure. Ultimately, this was intended to provoke explorations of different drawing strategies and encourage the players to think creatively about how to defeat the AI.

iNNk's intended audience is the general public, especially players who are interested in NNs. As a multiplayer game, the game is intended to be played in a group setting, with players encouraged to discuss between each round and collaboratively

develop different strategies. It should be acknowledged that the design currently does not account for the possibility for players to "cheat" by directly telling each other the code word outside the game. This design decision was made partly following the convention of similar multiplayer games such as *Pictionary* [241], *Cranium* [297], and *Charades* [193]. These games assume that players are more incentivized to have a good game than to win too easily. In addition, it is desirable to encourage players to co-develop creative strategies to defeat the AI. Leaving their communication channel open might be a good way to accomplish this goal.

6.1.1 Highlighting the NN's Presence

Many existing NN-based games obscure the existence of the NN. NNs are often seen as an underlying tool that the player only interacts with indirectly. For example, in the game *Black & White* [294], players directly interact with the Creature without the knowledge that it is controlled by an NN.

Since the conceptual goal of iNNk is to encourage people to go from passive users of NNs to active and creative challengers, a design decision was made to highlight the NN's presence as is. Rather than using a more abstract metaphor, the NN is represented as a NN *non-player character* (NPC) opponent. The NPC was specifically designed to look like a characterized computer system. The character is also explicitly called "Neural Network" in the core UI and exposes its confidence value to further emphasize the functional aspects of the system. The confidence value of the NN is displayed as a percentage under the NPC character, which can be seen in the top right image of Figure 6.1, to draw players' attention to how certain the NN is in their current guess. When the NN becomes more certain, the character's screen color changes, and the confidence value increases. This provides players with a visual indication of when the NN may correctly guess the code word. From the Sketcher's perspective, knowing which line significantly increased the NN's confidence will help them build a better mental model of how the technology works and thus how to subvert it.

However, the underlining data processing conducted by the NN is still kept hidden from the user. This was decided to maintain a light, intuitive interaction from the user's perspective. In general, visualizing the decision process and providing meaningful insights to the latent intermediate matrix operations of a NN is a problem still being researched and a main subject in the XAI field [249].

6.1.2 Balancing the Game with an Ink Meter

An Ink Meter is also included as a game mechanic. The Ink Meter can be seen on the bottom of the left image in Figure 6.1. The purpose of the Ink Meter is to limit the amount a Sketcher can draw during the round.

The depletion of ink as the Sketcher draws defines a maximum length that drawn strokes can cumulatively cover across the canvas. As different drawings require

different amounts of ink to be adequately represented (e.g., the "line" code word can be sufficiently represented with much less ink than the "car" code word), this maximum length is defined separately for each drawing. The *Quick, Draw!* [102] dataset was used to calculate an average distance that each category of drawings covers to determine Ink Meter values for iNNk's drawings.

6.1.3 Neural Network Setup

The NN of the game is based on a tutorial by Tensorflow [274]. The architecture leverages a wide range of layers, namely, convolutional, recurrent, and dense. First, a series of three 1-dimensional convolutional layers maps the input into $[48, 64, 96]$ feature maps using kernel sizes $[5, 5, 3]$. The first convolutional layer takes three input channels due to the nature of the data (as described in the next section). After each layer, dropout [262] and batch normalization [136] are applied to regularize the network. The feature maps are then consumed by the recurrent (LSTM) layers. These layers are responsible for memorizing features across time. Three bi-directional LSTM layers were used [120], each with a hidden state of size 256. Each LSTM layer is followed by a dropout layer. Finally, a dense, fully connected layer computes a class-wise list of classification likelihoods. This layer takes an input vector with a cardinality of 512 due to the bi-directional LSTM layers. The output of the NN is a vector of logits. These are modified using the Softmax function to convert them into probabilities over the 345 distinct classes.

The loss computed during training is the cross entropy between the NN prediction and the ground true label.

To determine when to terminate the training procedure, early stopping [226] was applied with a patience of 20. Further, a LR of 3×10^{-4} and a batch size of 256 were used. The Adam optimizer was used for gradient descent [152], a dropout [262] rate of 0.3 for all dropout layers, and Xavier initialization [96] for the free parameters of the baseline model. Finally, gradient clipping was used to prevent exploding gradients of the recurrent model [99] with a maximum ℓ_2-norm of 1. The experiments were executed on a machine with an Intel Core i7-5820K CPU, with 64GB of RAM and an NVIDIA GP102 TITAN X GPU.

Once trained, the model starts to make predictions (i.e., internal guesses) from the moment the Sketcher makes the first stroke on the canvas. The label associated with the highest predicted confidence constitutes the guess of the NN. The NN continues to generate guesses; however, these are only presented to the player when above a certain confidence value. Previous, incorrect guesses by both the NN and the human players are used to mask the output of the NN by removing these categories before rendering the guess of the NN. In this way, the NN is able to participate in a way that mimics the other human Guessers.

6.1.4 The "Quick, Draw!' Dataset

The NN was trained on hand-labeled sketch data from a canvas similar to the one used in iNNk itself. This sketch data is from Google's publicly available *Quick Draw!* [102, 113] dataset and includes 50 million drawings across 345 categories (i.e., 345 supported secret code words) of example sketches. When training the NN, 10% of this dataset was used as a validation dataset and 20% as a testing dataset. The sketches in this dataset are human-made digital drawings represented as temporally ordered sequences of *strokes* that make up the drawing. Each *stroke* in this sequence is a continuous curve. The curve starts when the Sketcher's drawing tool (such as a computer mouse) is activated and ends when the drawing tool is deactivated. This curve is represented as a sample of temporally ordered 2-dimensional point coordinates along the curve. In addition to the recording of the sketch, the dataset also contains metadata for each sketch. The most relevant metadata for training the NN was the label of what each sketch depicts. The "simplified" version of this dataset (provided by Lab [164]) was used in the training process, in which the sketches had been preprocessed by centering and scaling them into a 256×256 space. The strokes were also simplified with the *Ramer-Douglas-Peucker* algorithm [232]. Finally, a min-max normalization of the stroke data was conducted as the last step of the data preprocessing. In addition to the temporally ordered 2-D coordinates, the NN also receives a binary signal indicating when a stroke is ended.

6.1.5 Creating Moments of Surprise and Failure

Many game designers use repeated failure to encourage players to rethink their gameplay strategies [145]. Through failure, players can reflect on their current actions [145], self-correct, and use these experiences to become better in the game [10, 90].

INNk is an attempt to create a playful experience through the game design, where human players are encouraged to become more active and creative challengers against a NN. Through moments of surprise and failure, iNNk intends to provoke explorations of different drawing strategies and encourage the players to think creatively about how to defy the AI.

These moments are supported through the game structure (i.e., timed rounds) and the NN's confidence meter (i.e., NN confidence value) to facilitate player reflection and new drawing strategies. A moment of surprise in iNNk might be to "wow" players with the strengths of the NN's image recognition. For example, in most cases, the NN is exceptionally good at guessing the code word from a limited or incomplete drawing. This strength is intended to surprise and defeat the human players to trigger player reflection on how they may draw the code word differently for the next round.

The confidence meter provides a visual reference for players to use when performing a strategy. For example, if a player tries to mislead the NN by drawing

extra lines before they begin to draw the code word, they can see this strategy take effect on the NN's confidence meter by the percentage decrease. As a result, players are able to link their gameplay (i.e., drawing and guesses) to what decreases the confidence value. The visualization of this value in the interface is intended to provide players with another source of visual feedback, in addition to winning or losing the round, on what strategies may impact the NN's certainty.

6.2 Observed Player Strategies

Based on playtesting, it was observed that three different adversarial strategies were commonly developed (Figure 6.2). These strategies would consistently stump the NN while the drawing was still easily decipherable by the human Guessers – thus breaking the balance of the game. The three strategies are not exhaustive, and a more thorough user study will likely reveal many more strategies.

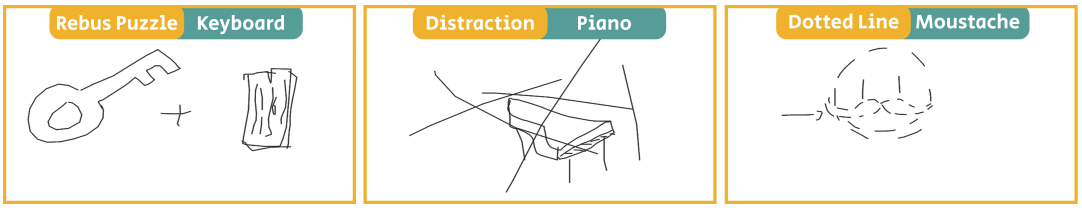


FIGURE 6.2: **Examples of the three adversarial player strategy sketches used for retraining: Rebus Puzzle (left), Distraction (center), and Dotted Line (right).** For the example of the Rebus Puzzle strategy, the code word is "Keyboard." The Sketcher decided to divide this into two separate sketches, one of a key and one of a wooden board. An example of the Distraction strategy is a drawing for the code word "Piano." Using this strategy, the Sketcher added straight lines to stump the NN. The superimposed sketch of a piano is otherwise unchanged. Finally, for the Dotted Line strategy, the example shown is for the code word "Moustache." Here, the Sketcher only modified the line style.

The first strategy involves the Sketcher drawing the code word in a sequential set of images (i.e., as a rebus). The Sketcher was given the code word "eyeglasses" and sketched two separate images, an eye and a pair of drinking glasses, in an attempt to stump the NN. The second strategy involves adding visual noise or other shapes in addition to the drawing of the code word. In this case, the player would crosshatch or draw extra lines to mislead the NN. The players added these extra lines at the beginning of the round. Adding them later would allow the NN to classify the drawing correctly before adding the distraction lines. The third strategy includes drawing the code word using a different stylistic technique (here, using a dotted line instead of a solid line).

Based on these three observed adversarial player strategies, the game designers manually drew around 100 sketches for each one (105 for Distraction, 93 for Dotted, and 66 for Rebus). This process was followed to create a clean dataset of each strategy with a sufficient size while minimizing labeling cost. For the Distraction

strategy, this was as simple as adding arbitrarily placed lines on some part of the drawing canvas at the beginning of the sketch. It was similarly straightforward to generate data for the Dotted Line strategy by drawing while releasing the mouse periodically. The Rebus Puzzle strategy presented some issues, as not all of the classes in our dataset lend themselves to this approach. For example, as seen in Figure 6.2, the class "keyboard" can be broken into a key and a board of wood, which a human player may understand while the NN cannot. However, if the class chosen by the game is "dog," the word cannot be broken down using this strategy at all. This resulted in a sparser dataset with a less diverse set of classes for training a model on this strategy.

Next, the objective is to consider a method for dealing with emerging player strategies and defending against such adversarial examples in NNs.

6.3 Fooling Neural Networks

Fooling a NNs is not an uncommon endeavor. Adversarial attacks against NNs are an established field of research in the ML community. Research has repeatedly shown how NNs are easily misled, and has demonstrated how to trigger NNs to make incorrect predictions with high confidence [6, 101, 197, 204]. For instance, the concept of *generative adversarial network*s like StyleGAN3 [147] is founded on its ability to trick a NN to believe an artificially generated image is real [100]. This problem has led to the creation of many interesting artifacts, from inferring missing information [309] to the generation of completely new instances [35, 231]. Despite these seemingly innocent applications, the technique can also be used as an attack with severe consequences. An example is the modification of traffic signs such that a NN would misclassify a "Stop" sign as a "Speed Limit" sign [77]. To fool a NN, a common approach is using gradient descent to generate adversarial examples [100, 101, 163]. Others use *evolutionary algorithm*s [131, 269] or human-generated data [77].

Recently, a new research area has emerged to counter these adversarial methods and provide alternative defensive solutions [111, 214, 242, 293]. This work has ignited an arms race in the field, with continuous developments for both attack and defense mechanisms. Despite the numerous defense methods developed, they focus on computer-generated data to strengthen the robustness of the NN. However, what seems to have gained little attention thus far is developing defense mechanisms against adversarial attacks using human-generated data and making the NN adapt to such datasets.

6.4 Dealing with Adversarial Strategies

Section 6.2 discussed three emerging drawing strategies that have been shown to overexploit the NN to an extent where players would always win regardless of the object being drawn.

To regain a balanced gameplay experience, the aim is to decrease the model's bias on the adversarial strategies while maintaining the best possible performance on the original dataset. The most straightforward approach would be to simply append these new instances to the existing dataset and continue the training process with this augmented dataset. This approach, however, might not yield the best player adaptation, as discussed next.

In case the model's capacity is already fully used, reducing bias on the adversarial examples would result in reduced performance on the original dataset. This happens because the adversarial examples are out-of-distribution samples, meaning each developed player strategy can potentially result in sketches looking substantially different from the ones in the original training dataset. Hence, the function the NN needs to fit, capturing both the original data *and* the out-of-distribution samples, is more complex [99]. Without additional model capacity, the NN will have to "forget" some of the learned rules that applied to the original dataset, to learn rules applicable to the out-of-distribution data.

If the model's capacity is *not* fully utilized, the NN could learn the features of the new dataset without losing performance on the old. However, for the case studied in this chapter, the labeled dataset of an adversarial strategy is heavily underrepresented compared to the original training data. The sparsity of such collected datasets raises some difficulties, as it would be easier for the NN to neglect this small fraction of outliers than to reduce the bias on these. Therefore, it is unlikely for such an approach to result in a NN capable of accurately recognizing and accounting for the emerged player strategies.

Instead, inspired by *ensemble methods* and *transfer learning*, a combination with an ensemble of model specialists is proposed. These concepts are introduced next.

6.4.1 An Ensemble of Networks

There are several approaches to defend against adversarial attacks [111, 214, 242, 293]. One approach is the use of *ensemble methods*. For instance, Abbasi and Gagné [1] used a confusion matrix to choose the training dataset for fitting an ensemble of NN specialists. In general, ensemble methods have been shown to yield more robust models [114]. The baseline approach to ensemble methods involves having a set of ensemble members (weak learners) that, when combined, perform better than individual members alone [210]. This approach assumes the ensemble members are accurate and diverse [64, 114]. The ensemble approach can also leverage different types of weak learners and have proven successful with NNs [114].

Alternate approaches to ensemble methods have a considerable variation in current literature [34, 80, 215, 300]. Some methods, like boosting [80], train each ensemble member sequentially. Each weights the importance of each entry in the training dataset by the previous model's loss in that example – a higher loss gives a higher priority. Boosting decreases the ensemble's bias by focusing on ill-performing parts of the dataset. However, this also increases the risk of overfitting. Other ensemble

methods, like bagging, have proven less prone to overfitting as they decrease the variance of the predictions. They are also effective with unstable learners due to the smoothing effect of model averaging [64, 315]. To achieve a generalization improvement with bagging, diversity among the ensemble members is essential. This diversity results in a lower error correlation, as the ensemble members are less likely to make the same mistake on the same data [99].

6.4.2 Transfer Learning

Training robust, supervised ML models from the ground up usually requires a large and labeled dataset. If one does not have access to such a dataset, one can utilize the ML technique called *transfer learning*. This technique leverages the finding that using a fitted model on one dataset might serve as a good starting point for training a new model on a similar dataset [307]. Transfer learning is of particular use in domains with sparsely labeled datasets to help prevent overfitting [271, 314]. Further, it allows for utilizing fewer resources to acquire a model for a new task by adjusting the capabilities of an existing model [281]. Transfer learning has proven successful across a wide range of ML problems, from NLP [28, 129, 199] to image recognition [58, 88, 255].

6.4.3 Method

To address the emerging adversarial player strategies in iNNk, we created an ensemble of NNs. The NN structure as detailed in section 6.1.3 is reused for all ensemble members. Initially, a model is fitted to the original, non-adversarial dataset. This will be refereed to as the baseline model.

A key challenge to addressing the observed player strategies was the lack of examples. On average, the collected datasets with labeled, adversarial drawings contain 88 examples. Training such large NNs on datasets of this size can easily lead to severe overfitting. However, since the collected examples of the player strategies are of a similar nature to the original dataset, transfer learning is used to circumvent this challenge and minimize the cost of adapting to each adversarial strategy.

The training process is initialized by transferring the state of the baseline NN to a new NN model. This NN is then trained *independently* of the other NNs on its designated adversarial dataset. This training phase uses the same set of hyperparameters as that of the baseline model listed in section 6.1.3.

Due to the small datasets containing the user strategies, we used a testing and validation dataset of 10% for this method.

This training process is repeated for each collected adversarial dataset such that a set of specialists is obtained. An overview of the training procedure is shown in Figure 6.3.

After the training phase, all models are combined into an ensemble, as illustrated in Figure 6.4. To make predictions with the ensemble, each member is queried for

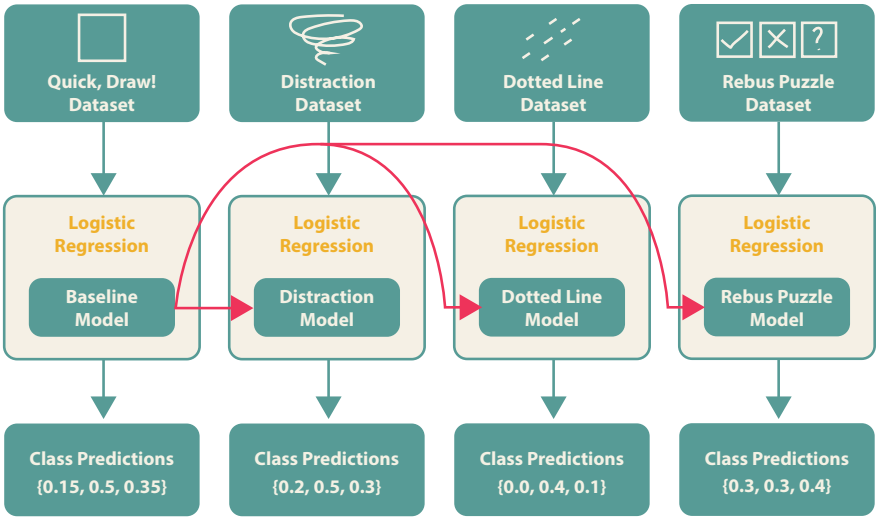


FIGURE 6.3: **An overview of the ensemble training procedure.** Before training a model, its state requires initialization. For this, transfer learning is used. A red arrow indicates the initialization of a model's state. The state of the model at the arrow's end is initialized to the state of the model at the origin of the arrow. Each model is assigned its own, distinct dataset and is specialized in that particular set. All models are trained using logistic regression for the classification of stroke data to one of 345 classes.

a prediction given the same observation. The resulting set of predictions needs to be combined into a single vector of class-wise probabilities. While any combinatoric method can be applied, plain model averaging was employed in these experiments.

6.5 Results

To test the performance of our proposed ensemble model with transfer learning, we evaluated each ensemble member and the ensemble itself on each dataset. Fitting the baseline model took ≈ 22 days, while fitting each ensemble specialist took <1 minute. The main objective is to maintain consistent performance across all datasets without losing performance on the original, non-adversarial "Quick, Draw!" dataset. Maintaining a good performance on the original dataset is of high importance in order to preserve an entertaining gameplay experience for players with non-adversarial strategies. The results are outlined in Table 6.1 and are averaged over three repeated experiments.

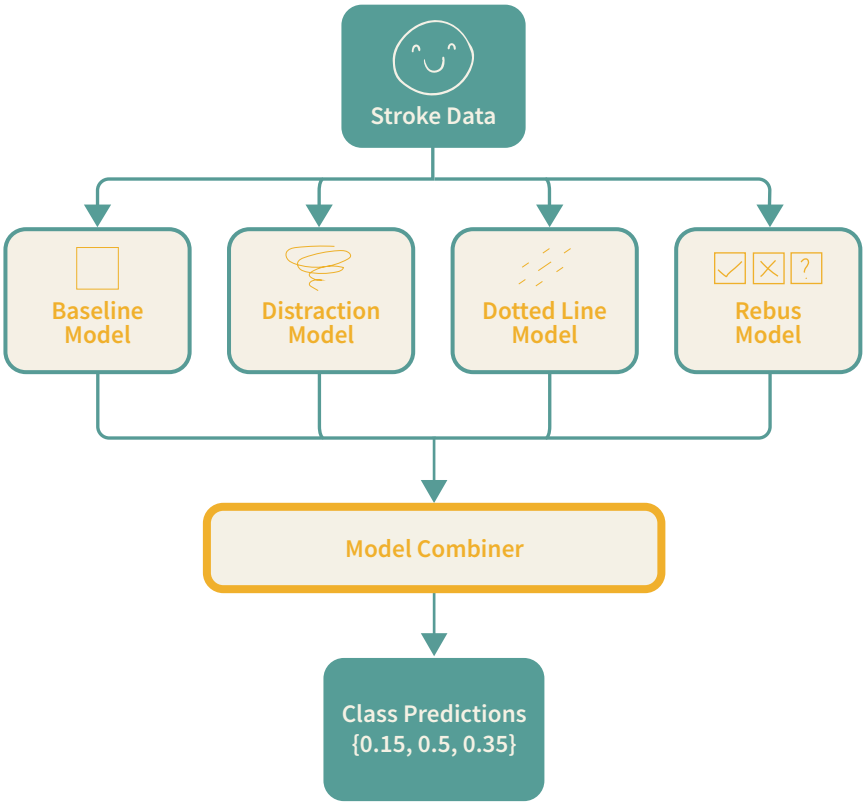


FIGURE 6.4: **Making predictions with the ensemble.** To make predictions, each ensemble member is queried for a prediction based on the same observation. The set of model predictions needs to be combined into a single prediction to compute the final, class-wise probabilities.

Table 6.1: Performance on the various datasets for each model on a set of performance metrics.

Neural Network	Dataset											
	Top-1 Accuracy				Top-5 Accuracy				Cross entropy			
	Quick, Draw!	Distraction	Dotted	Rebus	Quick, Draw!	Distraction	Dotted	Rebus	Quick, Draw!	Distraction	Dotted	Rebus
Baseline Model	**83.52**%	0.00%	0.00%	0.00%	**96.20**%	20.00%	0.00%	0.00%	**0.62**	5.22	7.52	6.97
Distraction Model	80.51%	**70.00**%	0.00%	16.67%	95.35%	**86.67**%	25.92%	16.67%	0.74	**1.82**	5.39	11.46
Dotted Model	77.80%	10.00%	**70.37**%	0.00%	94.33%	30.00%	**88.89**%	16.67%	0.84	6.00	**1.69**	7.50
Rebus Model	79.32%	13.33%	7.41%	**11.11**%	95.05%	50.00%	14.81%	**44.44**%	0.77	4.45	5.76	**4.28**
Ensemble Model	82.59%	50.00%	0.00%	16.67%	95.98%	70.00%	44.29%	33.33%	0.65	2.51	4.42	4.98

We evaluated each model on each dataset on three performance metrics: Top-1 Accuracy, Top-5 Accuracy, and Cross entropy loss. With the small testing dataset from each of the adversarial strategies at hand, the cross entropy loss is the better performance metric since accuracy is not a smooth measure. A decrease in cross entropy is often correlated with increased accuracy, but this is not always the case. With small datasets, minor modifications to a single prediction can yield high variations in the measurement accuracy. We included the accuracy metrics simply because these are more intuitive, but the cross entropy better describes discrepancies in the model's output distribution and the ground true target distribution. A higher cross entropy can indicate a less confident prediction or indicate a confident, incorrect prediction. Finally, one should be careful comparing cross entropy loss across tasks, and the reported cross entropy losses should only be compared to other models on the same dataset. An example of considerable discrepancies between the cross entropy loss and the Top-1 Accuracy is the Distraction model's performance on the Rebus dataset. In this case, the model obtains a 16.67% Top-1 Accuracy, which is higher than the Rebus model (on average), but with a stunning 11.46 cross entropy loss compared to the 4.28 of the Rebus model.

As shown in the table, each ensemble member performs the best on the dataset for which it is a specialist (the diagonal marked in bold). Yet, when used as standalone, each ensemble member generally does not perform well on the other datasets and suffer a loss of at least three percentage point on the "Quick, Draw!' dataset compared to the Top-1 Accuracy of the baseline model. The worst-case is the Dotted specialist with a performance loss of over five percentage points on the original task.

Further, it is clear from the table that the baseline model performs inadequately across all adversarial datasets. The only model performing consistently across all presented datasets is the ensemble model. While it suffers a performance loss on the Dotted dataset compared to the Dotted specialist, its cross entropy loss for this dataset is still highly improved over any other model. This is another example of disparities between the cross entropy loss and the Top-1 Accuracy. The Top-1 Accuracy suggests the ensemble model has made no improvement on the Dotted dataset. Yet, significant improvements are seen in the cross entropy, and these are reflected in the Top-5 Accuracy as well. Overall, the ensemble makes significant improvements on all adversarial datasets with a minimal loss in performance on the "Quick, Draw!' dataset across all performance metrics.

6.6 Discussion

The Dotted model's loss on the original dataset could indicate that the parameters of the model need substantial adjustments for improving performance on the Dotted dataset. One explanation could be the low-level feature maps in the first convolutional layers, which typically work as edge and curve detectors. These might transfer poorly from the "Quick, Draw!' to the Dotted dataset, as an edge in the Dotted

dataset no longer indicates separation but is most often part of a continuous line.

The motivation behind reporting the Top-5 Accuracy as a performance metric is due to the nature of the game *iNNk*. Here, an incorrect guess gets blacklisted, and neither players nor NN are allowed to make this prediction again. Technically, this happens by setting the NN's predicted logits to $-\infty$ for these blacklisted classes. For instance, when the class with the highest probability is blacklisted, the NN would predict its perceived second most likely class. This is to prevent the repeating of incorrect guesses. To prevent exhausting the list of whitelisted classes, the NN is only queried for a prediction every 2.5 seconds. For this reason, having the correct class among the Top-5 predictions is important for the gameplay, as the four incorrect guesses would be blacklisted within 10 seconds at worst. One could argue that not having a perfect Top-1 accuracy score even improves the gameplay experience, as an oracle-like NN might discourage players from playing the game. From the results in Table 6.1, it is clear that the proposed ensemble model gains significant improvements for all datasets on Top-5 Accuracy.

An inherent limitation in our approach on how to prepare for adversarial player strategies is that to identify one, it must have a recognizable pattern that the NN can be trained on. Therefore, our approach will not work on strategies developed when players communicate outside of the game to achieve the correct answer. This information cannot be made available to the NN, meaning there is no way to compensate for this type of strategy. Other strategies may involve inside knowledge within the player group, which would also be challenging and most likely not particularly useful to train the NN on.

Additionally, the proposed process of identifying and collecting examples of player strategies is limited in terms of scalability. For instance, our work identified player strategies through human observations and analyses. However, identifying strategies on a much larger scale or using a more complex game with this process may not be as easily executed. To address these limitations, one could investigate an automation of these processes.

Finally, each time a NN is added as a member of the ensemble, it prolongs the time it takes to make predictions. Making predictions with the ensemble can be run in parallel on separate GPUs. In practice, however, there is a limit to the scalability of the method and hence the number of addressable, distinct player strategies.

6.7 Future Work

This chapter has looked at three quite different player strategies that stump the baseline NN. Given that there could exist a large number of player strategies that dynamically emerge over the game's life cycle, it would be beneficial to automate the process of discovering new player strategies and the retraining of the NN. This would lead to a more dynamic experience of the game evolving over time and also reduce the amount of work in achieving this. Therefore, one could explore methods

for automating the player strategy discovery process using *clustering* ML methods, for example.

Given the substantial amount of manual labor in generating a dataset reflecting a player strategy by hand, one could also explore automating this process using generative methods. Finally, further fine-tuning on our proposed ensemble could be carried out – for example, by experimenting with different model prediction aggregation (voting) methods.

An additional point of investigation could be to assess the impact of batch normalization. Recent studies suggest the use of batch normalization can increase a NN's vulnerability to adversarial attacks [19, 82]. While more analysis is needed on this subject, future work could explore whether the baseline model applied in this research could benefit from abandoning the use of batch normalization in the model architecture. While such modification potentially increases model robustness, it will also remove the benefits from applying batch normalization – faster model convergence being one of them [135]. This could lead to an unacceptable increased cost in model training, as the current model already takes >3 weeks to fit.

6.8 The Player-AI Framework, iNNk, and BW7

Circling back to the AI-infused platform of BW7, this platform is not a game that novice users can download. It is not even a NN-based tool unless BW7 incorporates the proposed method from Chapter 4 in their production pipeline.

Nevertheless, applying the player-AI interaction framework introduced in Chapter 5 to this AI platform can still be useful. In principle, the framework can be applied on an arbitrary AI system to identify relevant games utilizing a similar form of HAII. These identified games can then reveal promising avenues for improving the HAII experience of the original system.

To apply the framework on BW7's AI platform, we first determine the visibility of the AI. The AI tool is purchased by companies who want an AI-advised MMP. The onboarding process for new customers consists of a non-negligible amount of prior instruction and guidance before working with the tool. The acquisition of new users and the marketing of BW7's product is centered around their AI capabilities. BW7's website[2] can be used as an example of just how AI-centered their marketing actually is. This page mentions "AI" more than 10 times[3].

In addition, BW7 introduces some overall ML properties used in their setup to aid recently acquired customers' understanding of the AI. In effect, classifying the UI as *NN-Specific* is deemed the best fit to this setup (as per definition of the framework presented in Chapter 5). This judgment is not because the UI directly mentions the use of a ML-model, but because of the circumstantial context provided to the user.

[2] blackwoodseven.com

[3] Website visited: 23rd of November, 2021

Next, the interaction metaphor used in the system is to be determined. The users of BW7's tool are teams of marketing employees. During the HAII, each team seeks an agreement on a MMP generated by the AI for an upcoming period of time. These employees engage with the AI in order to get an optimized output that ideally suits their preferences. Thus, the interaction metaphor applied is *"Neural Network" as a Teammate*.

Last is the characteristics of the AI-system. The AI-models created by BW7 are updated occasionally as new data arrives. Yet, during the HAII sessions, the AI remains static. Therefore, it is an "offline learning" AI since its behavior does not change continuously in response to user input.

As a result, the AI system of BW7 is an offline learning, teammate AI employed in a NN-Specific context. Looking at the results from Figure 5.2, the cell corresponding to those characteristics contains no entries, as none of the studied games exhibit this type of player-NN interaction.

In contrast, iNNk is an offline learning, competitor AI employed in a NN-Specific context and therefore placed at the diametrical opposite end of the figure from the cell representing the characteristics for BW7's platform. Yet the only difference between the two is in the interaction metaphor.

This difference in interaction metaphors is more delicate in the context of HAII than one might think. When a NN is employed as a teammate, the human(s) must develop a mental model of the NN in order to anticipate its actions to maximize the benefit of the co-operation. The same is true when a NN is employed as a competitor, in which case the human(s) uses the developed mental model to hinder the NN instead of enabling it. It can be said that for both interaction metaphors, the player(s) uses the mental model to trigger a desired output of the NN.

Consequently, insights to HAII and UX design in iNNk are transferable to the context of BW7. Observing players in iNNk suggested just how efficiently the iterative feedback loop can improve mental models even though the underlying decision process of the NN remains opaque to the players.

Based on these insights, the next chapter has HAII in the context of BW7's platform at the center and proposes a stepping stone toward a seamless HAII.

6.9 Chapter Summary

This chapter started out by introducing the NN-based game *iNNk*. INNk was motivated by the player-AI interaction framework developed in Chapter 5, which revealed several underexplored areas in the field of player-NN interaction. INNk was used as a case study for studying novel player-NN interactions. Studying these interactions indicated that such setups can transform players from passive users into creative and reflective thinkers, a crucial step toward a more mature relationship with AI. These insights can be directly transferred to augment the HAII experience with BW7's platform.

The study conducted in this chapter also highlights how fragile otherwise high-performing NNs can be to out-of-distribution examples. This fragility rendered the player-NN interaction less balanced, as the emergent drawing strategies would consistently stump the NN.

The chapter continued by introducing an approach to adapt a NN to combat these adversarial player strategies in order to regain balanced gameplay. This approach combines transfer learning and ensemble methods to strengthen the classifier based on sparse datasets. We evaluated the performance of our approach using three different performance measures on the testing datasets. As a result, we found that each model dedicated to a strategy lost performance on the baseline dataset, while the baseline model was ineffective on all player strategies. We found our ensemble approach to provide the best, most consistent performance across all datasets, with $<$1% loss on Top-1 Accuracy on the original dataset compared to the baseline model. Since our method is effective and efficient to set up, despite our ensemble of models being trained on a very limited amount of data, it can be used by others to create more competent NNs in their own domains. Lastly, we present potential ways to develop this method further that include automating the process and building more sophisticated ensembles, which would result in more accessible and more interesting use cases for NNs in games.

Part IV

Towards Preference-based Mixture Marketing Plans through Diversified Options

Chapter 7

Pursuing Explainable Diversity in the Context of Mixed Marketing Plans using a Unified Concept

The previous part looked at player-NN interaction in games. Studies of players interacting with the NN in the game iNNk stressed how differently humans and NNs classify the same data. In the course of the interactive sessions, the users developed a mental model of the NN. As the model grew accurate, these users successfully identified and applied exploits to fool the NN. The repeated player-NN interaction played a key role in these achievements.

This chapter aims to serve key facilities for future research to achieve the same effect in the context of BW7's platform, only in this case, the NN is not a competitor. An efficient maturation of mental models will ease the process of introducing new users to the platform.

As hinted in the introduction of this work, users of BW7's platform often have a hard time *relating* to the marketing decisions of the AI[1]. Improving the user's mental model of BW7's AI system does not render the generated MMP more relatable. The improvements will only refine the user's perceived error boundary of the AI. It is still the user who makes the ultimate decision on whether to follow the proposed MMP.

Psychology research can provide hints as to what might alter the user's ultimate decision in this setting. Such research has found several interesting properties of the (ir)rational human decision-making process [141]. One of these findings is the *status quo* bias. When given a set of new alternative options, Zeckhauser and Samuelson [311] found a strong bias in decision-makers sticking with *status quo*. Hence, if the generated MMP heavily contradicts the experience of a marketing employee, that person is unlikely to follow the advised MMP, and instead take the *status quo* option: manual planning.

[1]Here, *relating* means that the user can justify the rationale behind the decisions and that the MMP made from these decisions conform to the user's experience of what constitutes a good MMP. In short, it should be XAI [68]

Incorporating past experience of a person in a ML optimization, however, is non-trivial. To do so, these experiences have to be condensed into an objective function that is suitable for a ML optimization and whose optimum is the solution of interest [252].

Rather than composing such an objective function, this chapter takes a different approach based on diversity.

Throughout the remainder of this work, we will refer to a user's experience as a user preference, since experience can be considered a latent form of preference and to prevent confusion with the term *user experience* (UX) as used in the field of UI design.

When using the method described in Chapter 4 for generating MMPs, the optimization outputs only a single solution candidate for a given setting. If the search space contains two or more local optima with near-identical performance and budget, the optimization yields only the best of the two options – assuming the optimization acts as intended, that is.

With the complexity of the marketing landscape in mind, the existence of two or more high-performing MMPs is not improbable. If one of these MMPs is more harmonious with the user's preferences than the others, that solution is paramount. It is paramount because the marketing employee is more likely to abandon the *status quo* and appreciate that solution candidate.

This solution candidate might not be located at the global optimum of the search space. However, it is still the result of a data-driven optimization process for marketing, and therefore, likely a better solution than *status quo* with manual planning.

This chapter proposes a method for exploring the marketing landscape within the optimization. The method traverses the search space and collects various identified local optima as solution candidates. The idea being that one of these solution candidates is closer to the preferences of an arbitrary marketing employee. This approach is only feasible due to the gain in speed and accuracy achieved in Part II through approximation. Otherwise, thoroughly traversing the search space would be outside the time frame of marketing planning.

The scope of this chapter is limited to *how* such an exploration can be conduced. Thus, it does not dive further into the UX considerations for such a system with multiple solutions for the same problem.

This chapter is structured as follows: First, a discussion on different approaches one can take for exploring this search space is given. This leads to the approach taken in this chapter, based on a concept appreciable by both human and AI. This concept provides structure to the repeated traversing of the search space for both parties. It is a highly domain-specific concept based on *groupings of insertions* and constraints.

The approach also introduces the algorithm used for modifying the loss function for the input-optimization procedure based on the introduced constraints. Next, the

full setup that utilizes the aforementioned components is presented. The approach is then tested on the industrial setting from Chapter 4.

Finally, the chapter will conclude with a discussion on the obtained results and potential future applications for this approach.

7.1 Quality Diversity Search Methods

The introduction of this chapter motivated the gathering of a set of candidate solutions for the same marketing optimization task. This collection can be used as a mean to accommodate user preferences. The question is then *how* to traverse the search space of the ML optimization to acquire such candidate solutions.

Various different approaches could be taken for such a task. A simple method could be to store the state of each point traversed during gradient descent. Effectively, one could obtain thousands of solution candidates with such an approach with no additional cost to the optimization. This idea is inspired by Huang et al. [128], in which the authors take snapshots during training to collect an ensemble of NNs for free. However, there is no guarantee that such an approach will thoroughly cover all local minima of the search space.

Instead, the approach taken in this chapter is inspired by diversity-seeking population-based *evolutionary algorithm*s, as detailed below.

An example of such a diversity-seeking algorithm is *multi-dimensional archive of phenotypic elites* (MAP-Elites) [57]. In short, MAP-Elites collects a set of diverse solution candidates to solve a particular problem. Each solution candidate is assigned to a niche bin in a behavioral space based on its characteristics. The best solution candidate of each niche bin is part of the "elite." An example of this method's successful application is robots that can adapt when faced with a malfunctioning part like a leg [57].

Diversity-seeking algorithms like MAP-Elites generate a vast number of solution candidates [57, 238]. For example, for the damage-recovering robot, 40 million solution candidates were created over a period of two weeks with MAP-Elites [57]. These results are somewhat discouraging for the case of BW7. Even with the gain in speed from Chapter 4, generating 40 million MMPs spanning one month will take >15 years.

Aside from the computational cost, user fatigue is a salient issue with HCI in general. User fatigue occurs when the user of a platform generally loses interest. At this point, the user becomes less consistent in the choices made [251] and tends to apply a *satisficion strategy* – picking the first reasonable option [158].

A literature review of research conducted in *interactive evolutionary computation* can provide guidelines in terms of how many MMPs it is feasible to present an end user of BW7's platform before they experience fatigue. Semet [251] found that user fatigue occurs after just 20-25 human-AI feedback-loops.

This chapter does not dive into details on how the UX design could be constructed for the method presented later. Yet, it is important to keep in mind the reality in which the method has to function. Due to the risk of user fatigue, the MMPs supplied to the user should be chosen with care. Also, the timeframe in which the marketing planning happens is quite limited – this also served as the motivation for Part II. For both reasons, the exploration of the marketing landscape when creating diverse MMPs should be fairly efficient.

Another example of diversity-seeking algorithms is *novelty search* (NS) [238]. NS manages a population of solution candidates. Each member of the population is given a fitness score based on its novelty. The members with the highest fitness, and thus the most novel behavior, are randomly combined and mutated in the hope of getting even more novelty in the population [236]. Hence, this method *pushes* members toward unexplored areas simply through a scoring function. An example of such a scoring function is simply the Euclidean distance to the *k nearest neighbor*s (KNNs) in the behavioral space [238]. The higher the distance, the more novel the behavior.

Inspired by these algorithms, how could such an approach be adapted to the marketing domain? Both NS and MAP-Elites construct a behavior space to reward diverse/novel solutions. Rewarding diversity in the context of marketing planning could be done in a similar manner. Let us first recapitulate the loss function used for generating optimized MMPs. This loss function was introduced in (4.4) from section 4.3 and is repeated in (7.1) for convenience.

$$\begin{aligned} \mathcal{L}_{\mathrm{P}} &= \sum_{t=1}^{T} \frac{1}{M} \sum_{m=1}^{M} \mathbf{Y}_{t,m} \\ \mathcal{L}_{\mathrm{B}} &= \left(\mathrm{B} - \sum_{t=1}^{T} \sum_{j=1}^{J} \mathbf{X}_{t,j} \right)^2 \\ \mathcal{L} &= \pi \mathcal{L}_{\mathrm{B}} - (1 - \pi)\, \mathcal{L}_{\mathrm{P}} \end{aligned} \tag{7.1}$$

This loss is a trade-off between increasing the gain in KPI, i.e., $\mathcal{L}_{\mathrm{P}}$, and penalizing for deviating from a predefined total budget $\mathcal{L}_{\mathrm{B}}$.

To reward diversity in the optimization process, a diversity scoring function needs to be added to (7.1). This function should alter the loss landscape by decreasing[2] the loss in areas not covered by any MMP created in previous optimizations.

Such a reward function should be carefully balanced with the two other terms in the loss function, $\mathcal{L}_{\mathrm{P}}$ and $\mathcal{L}_{\mathrm{B}}$. In case one term overshadows the others, it will dominate the direction of the optimization. If balanced correctly, however, repeated iterations of the MMP optimization should yield a set of candidate solutions that covers all local minima in the loss landscape.

Leaning on existing work with NS [238], one could use a Euclidean distance function to the KNNs as a measure of diversity in MMPs. There is a few issues with

[2] Assuming gradient descent.

such an approach, though. First and foremost, for the industrial setup introduced in Chapter 4, a MMP spanning one month is a 31×106 dimensional matrix. With such high-dimensional optimization problems, a Euclidean distance and KNN approach performs poorly due to the curse of dimensionality [3, 65].

Another distance metric like *cosine similarity* could be used instead. In that way, two MMPs with the same spending pattern but with different magnitudes would be measured as being identical.

However, there is a fundamental issue with the approach. There is nothing to prevent the optimizer from highly increasing or decreasing the spend on a specific insertion at a specific day while leaving all other dimensions unaffected. This is a simple way for the optimization to obtain an improved diversity score while suffering only a limited loss in KPI.

From a human perspective, such a seemingly unpredictable fluctuation in spending pattern at a single day and insertion is hard for the user to justify. The intended diversity effect will appear as nothing but an error by the AI – an outlier. As a result, the unexpected behavior violates the user's mental model and thus hurts the human-AI team performance rather than improves it.

Instead, the members of the collection with optimized MMPs should express diversity through trends in spending patterns easily recognizable and distinguishable by the users. In other words, the diversity should ultimately be *explainable*, as with XAI in general.

Therefore, this chapter takes an approach besides pushing away from existing solutions. Instead, the approach *pulls* the optimizer into unexplored regions. Much like the considered pushing approach, the pulling is done by adding a new term to the loss function in (7.1). However, this term alters the loss landscape in a fundamentally different way than a pushing term would. The new term guides gradient descent into areas based on a grouping of insertions. This grouping is based on an established concept in marketing and is thus interpretable by humans. Further, it aids the AI in generating diverse MMPs with more consistent trends that are easily identifiable by the end user when compared with other MMPs. This grouping of insertions is explained next.

7.2 Approach

The approach taken in this chapter builds on the concept of channels to *unify* humans and AI. Channels work at a level of abstraction meaningful to marketing employees and easily implementable in the AI. Next, we define the concept of channels in a marketing context.

7.2.1 Channels

As detailed in section 2.4, a MMP lays out a company's spending on marketing over the set of all possible *insertions* and a given time period. An insertion is, for instance,

a specific newspaper or TV channel. The industrial setting introduced in section 4.4.2 is the only domain studied in this chapter. This domain allowed marketing spending on a total of 106 distinct insertions. However, many of these insertions are of similar format. For instance, one can gather all newspaper and journal insertions in a single collection named "Print," as they collectively make up the total spending on paid advertisements in a printed format. This grouping can be done for all insertions and provide a higher level of abstraction when portraying a MMP to marketing employees. Throughout the remainder of this work, such a collection of insertions will be referred to as a *channel*. The number and type of channels present depends on the marketing domain at hand. In our case, there is a total of *nine* different channels as outlined and described in Table 7.1. This usage of channels in relation to marketing differs a bit from the commonly accepted meaning of the term [184, 295, 296]. In the widespread usage of the term, a marketing channel is defined as

> [...] the set of people, organizations, and activities that work together to transfer goods (products and services) from the point of origin to the point of consumption ([296]).

This definition encapsulates a wide range of activities spanning from blog posts and live chat to *search engine optimization* and digital advertising. This work, however, has paid advertising at its core, and concentrate on optimizing these advertisements with respect to the four P's (see section 2.4). Due to this restriction, the term "channel" in relation to marketing is, in this work, a bit more narrowly used. Marketing channels not based on paid advertisement are disregarded. An example of such a disregarded marketing channel is influencer marketing. Instead, a detailed set of paid marketing channels are used (Table 7.1). The concept of channels as used in this work is more formally defined in the following, and is given in the form of propositional logic.

Let $\mathcal{I} = \{i^{(1)}, i^{(2)}, \ldots, i^{(J)}\}$ be the set of all J insertions and $\mathcal{F}^{(c)}$ be one of the $C = 9$ channels. $\mathcal{F}^{(c)}$ is a set of insertions such that

$$\bigcup_{c=1}^{C} \mathcal{F}^{(c)} = \mathcal{I} \quad \wedge \quad \bigcap_{c=1}^{C} \mathcal{F}^{(c)} = \varnothing \quad \wedge \quad \nexists \mathcal{F}^{(c)} = \varnothing$$

Further, let $F(i^{(j)}, i^{(j')})$ be the predicate that insertion $i^{(j)}$ and $i^{(j')}$ are of the same advertising format, then any two insertions $\{i^{(j)}, i^{(j')}\} \in \mathcal{F}^{(c)}$ satisfy this predicate.

With these quantifiers at hand, the next section utilizes these to obtain diversity in a set of optimized MMPs.

7.2.2 Placing Constraints on Channel Spend

As motivated in section 7.1, the general approach is to *pull* the optimization into unexplored areas. This is done by setting spending requirements on a specific channel through modifications to the loss function. Applying various constraints to the

TABLE 7.1: A table with the nine different types of channels for this particular domain. Each listed channel is accompanied with a descriptive text and examples of insertions it contains.

Channel Name	**Description**	**Number of insertions**
Cinema	Insertions such as movie theaters showing advertisements before movies.	2
Digital Display	Insertions with online advertisement as a format. Examples are advertisements on third-party webpages such as eb.dk and bbc.com.	18
Out of Home	Insertions for which the advertisements are displayed in a public area, for instance at train stations.	7
Print	Insertions with printed advertisements in a format such as physical newspapers.	37
Radio	Radio channels with an option for running commercials. Each insertion is a specific radio channel.	2
Search	Search-related insertions, such as advertisements shown based on a Google search result.	2
Social Media	Digital advertisements shown on social media platforms like Facebook and Snapchat.	23
TV	Traditional TV commercials. Each insertion is a specific TV channel.	6
Video	Includes insertions that show an online video commercial before some content, e.g., a Youtube video.	9

optimization enforces diversity in the output. For this method to be applicable in a real-world scenario, it is vital that the input optimization process is a cheap operation, as it is repeated numerous times. For this reason, the method is heavily reliant on the results achieved in Chapter 3 and 4.

Consider again (7.1). One can reuse the concept of penalizing for deviating from the total budget to construct constraints at a lower level. An example of such a low-level constraint is minimum/maximum spending on a specific channel or even insertion. Such a constraint can be further detailed to cover only a subset of the period spanned by the MMP.

To do so, let $\mathcal{L}_B$ be redefined using Algorithm 3. In brief, the algorithm modifies the loss used for generating MMPs based on the spending constraints in use. This is needed to ensure the resulting MMP adheres to all constraints. In the algorithm, c is a constraint and a member of the set of constraints $\mathcal{C}$. An example of such a constraint is the total budget for the MMP. Unlike a minimum/maximum spending

constraint, the total budget constraint is a *fixed* constraint, meaning it penalizes for going either below or above the specified amount. In contrast, a minimum constraint penalizes only for going below the bound. Line three in the algorithm is almost identical to $\mathcal{L}_{\mathrm{B}}$ in (7.1), only now it involves an additional operation. This operation is to ensure that the computed loss only considers the spending on the days and insertions the constraint is set to cover. For this reason, each constraint has an associated mask – a matrix $\mathbf{M}^{(c)}$ to filter out spending unattached to the constraint. In addition, $\mathrm{B}^{(c)}$ indicates the prespecified spending bound related to constraint c. Complying with various constraints through modifications of the loss function for

Algorithm 3: Algorithm for computing $\mathcal{L}_{\mathrm{B}}$ given a set of spending constraints $\mathcal{C}$

Input : A set of spending constraints $\mathcal{C}$, a MMP $\mathbf{X}$
Output: The budget loss $\mathcal{L}_{\mathrm{B}}$

1 $\mathcal{L}_{\mathrm{B}} \leftarrow 0$
2 **for** $c \in \mathcal{C}$ **do**
3 $\quad \ell_c \leftarrow \left(\mathrm{B}^{(c)} - \sum_{t=1}^{T} \sum_{j=1}^{J} \mathbf{M}_{t,j}^{(c)} \mathbf{X}_{t,j} \right)^2$
4 $\quad$ **if** c **is a** *Minimum Constraint* **and** $\ell_c < 0$ **then** `// Constraint satisfied`
5 $\quad\quad \ell_c \leftarrow 0$
6 $\quad$ **else if** c **is a** *Maximum Constraint* **and** $\ell_c > 0$ **then** `// Constraint satisfied`
7 $\quad\quad \ell_c \leftarrow 0$
8 $\quad$ **end**
9 $\quad \mathcal{L}_{\mathrm{B}} \leftarrow \mathcal{L}_{\mathrm{B}} + \ell_c$
10 **end**
11 **return** $\mathcal{L}_B$

an optimization procedure is in itself not a novel endeavor [127, 148, 161, 301]. In fact, such constraints were also depicted as part of the original procedure outlined in Figure 4.1. Marketing employees currently using the platform developed by BW7 use these constraints to ensure circumstantial conditions are being met, such as utilizing prepaid TV commercial slots.

7.2.3 Using Constraints as an Enabler for Diversity

Our method considers each of the nine available channels in turn. For each channel, the input optimization process is executed with a spending constraint[3] on that particular channel. This process is repeated $E^{(max)}$ times for said channel. Each time the process is repeated, the amount of spend required to minimize the loss related to the spending constraint increases by a factor of $\mathrm{B}^{(total)} / E^{(max)}$. Here, *total* is a fixed constraint used to penalize deviations from the total allocated budget for the MMP.

[3] Can be either a minimum and maximum constraint.

Using this method, $\mathcal{C}$ from Algorithm 3 contains two constraints, $\mathcal{C} = \{total, diversity\}$. Since the constraint *total* covers all days and insertions involved in the generated MMP, let $\mathbf{M}^{(total)} = \mathbf{1}_{T \times J}$. In contrast, the mask for the arbitrary constraint named *diversity* is constructed as

$$\mathbf{M}^{(diversity)} = \left[\mathbf{M}_{t,j}^{(diversity)} = \begin{cases} 1, & \text{if } j \in \mathcal{F}^{(diversity)} \\ 0, & \text{otherwise} \end{cases} \right]_{\forall t \in 1,\ldots,T \cup j \in 1,\ldots,J} \tag{7.2}$$

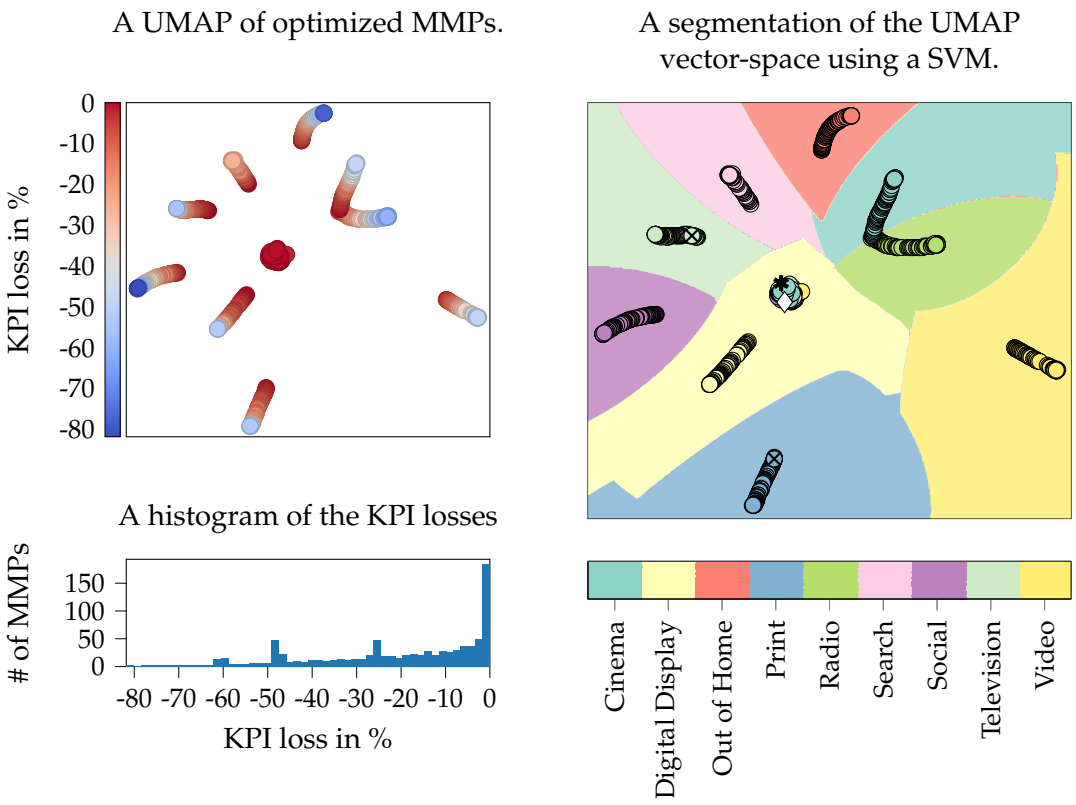
where $\mathcal{F}^{(diversity)}$ denotes the channel related to constraint *diversity*. The last modification to the loss function from (7.1) is related to the scaling variable π. π is, as before, slowly annealed. Since there is now more than a single constraint to be satisfied, π is annealed until a maximum of 1% violation occurs over all constraints in $\mathcal{C}$. Here, 1% is relative to $\mathrm{B}^{(c)}$ for each constraint under consideration.

7.3 Experiments

The NN fitted to the industrial domain, as described in Chapter 4, is reused for the experiments presented next. The industrial domain used here allows for spending on $J = 106$ different insertions distributed across nine channels. For these experiments, let $T = 31$ and $\mathrm{B}^{(total)} = TF_{\mathbf{d}}^{-1}(.75)$, similar to (4.7). That is, the total spending constraint $\mathrm{B}^{(total)}$ is set such that it corresponds to the 75$^{\text{th}}$ quantile of the particular company's historical spending for a marketing period of 31 days. These experiments use a *minimum* spending constraint as a driver for diversity. For each of the nine channels, an optimized MMP is generated $E^{(max)} = 100$ times – each iteration increases the bound $\mathrm{B}^{(diversity)}$ for the minimum spending constraint on the channel at hand. As thus, the final result is a set of 900 MMPs, each MMP is a 31×106 matrix.

7.4 Results

For the set of MMPs to provide any value, an overview of the generated MMPs is called for. This overview should allow one to easily identify MMPs that significantly differ in their spending pattern and those that are more closely related. Furthermore, the overview should shed light on which of these are particularly high performing MMPs, if any. Such an overview would considerably aid the pursuit for quality diversity in marketing plans. To do so, a dimensionality-reducing technique is used, namely, *uniform manifold approximation and projection for dimension reduction* (UMAP) [190]. Each MMP is reduced to a single point in a 2-dimensional vector space defined by the UMAP. Figure 7.1 visualizes said UMAP in two plots. On the left figure, a heatmap indicates each MMP's predicted KPI. This KPI is measured using the ground true BM, and is relative to the MMP generated by the BM *without* any such minimum spending constraint. As the heatmap indicates, several MMPs suffer only a small loss in KPI compared to the ground true. To further examine how



(A) **A dimensionality-reduced 2D map of diverse MMPs (top).** The dimensionality reduction is done using UMAP based on a dataset consisting of optimized MMP. In this dataset, each MMP is obtained using a minimum spending constraint on a specific channel. In the figure, each marker represents such a MMP. The color indicates the loss in KPI compared to the unconstrained MMP generated using the BM. These losses are computed using the BM and thus serve as ground true labels. The bottom figure depicts the same KPI losses as a histogram. While some of the generated MMPs suffer from significant losses in KPI, the largest bin, with over 150 MMPs, performs quite adequately ($< 2\%$ loss). As the amount of spending needed to satisfy the constraint increases, the optimization procedure gets increasingly tied to the specific channel, regardless of the channel's performance. For this reason, one can see significant drops in some of the MMPs' KPIs.

(B) **The same 2-dimensional manifold as in 7.1a**. In this figure, however, each marker is colored according to the spending constraint's designated channel. I.e., if a MMP was generated while employing a minimum spending constraint on "Print", the marker is colored accordingly. Similarly, each coordinate in the vector space is colored by using a SVM. The SVM predicts the channel on which a minimum spending constraint is needed to obtain an optimized MMP that corresponds to said coordinate in the UMAP manifold. The white diamond at the center is the MMP obtained using *no* spending constraint. While the SVM segments the plane aptly, it suffers difficulties around the cluster containing the white diamond. This cluster contains MMPs generated with spending constraints on a wide variety of channels. When the amount of spending needed to satisfy the minimum spending constraint is sufficiently low, the resulting solution vector is largely unaffected by it. The cluster around the white diamond is the result of this property.

FIGURE 7.1: **A UMAP of optimized MMPs**. A set of highly diverse MMPs is obtained using constraints on the optimization procedure. While many MMPs perform well, they differ significantly in spending pattern.

many of the MMPs that perform adequately, a histogram is shown at the bottom of Figure 7.1a. On the histogram, one can see the 900 MMPs distributed among 50 bins.

Although some of these MMPs suffer significant losses in KPI, out of the 900 generated MMPs, > 150 perform on par with the ground true ($< 2\%$ loss in predicted KPI).

Most of the MMPs in Figure 7.1a are situated in bands. To assess the meaning of these bands and, in effect, the local and global structure of the UMAP, a *support vector machine* (SVM) [29, 51] is employed. This SVM is fitted to the UMAP embedding of each MMP. Given such an embedding, the SVM makes a prediction in the shape of a channel, namely $\widetilde{\mathcal{F}}^{(diversity)}$. $\widetilde{\mathcal{F}}^{(diversity)}$ is a prediction of the ground true $\mathcal{F}^{(diversity)}$, i.e., the channel on which the diversity constraint was enforced when the MMP represented by the embedding was generated. Once fitted, the SVM is queried for a prediction for each coordinate in the UMAP.

Figure 7.1b visualizes these predictions by the SVM. Here, each coordinate is colored in accordance with $\widetilde{\mathcal{F}}^{(diversity)}$. In addition, each of the 900 MMPs is indicated by a circle. Each circle is colored according to the corresponding MMP's $\mathcal{F}^{(diversity)}$; hence, these are the ground true labels. Two of the 900 circles are brought forward and marked with an "X". This is done to pinpoint two MMPs that are later examined and compared in greater detail. Figure 7.1b also contains a white diamond. This diamond represents the MMP obtained using the NN and no minimum spending constraint on any channel. Likewise, the black asterisk on the figure represents the MMP generated by the *BM* – also without a minimum spending constraint on any channel.

Figure 7.1 indicates a wide range of high-performing MMPs that are far apart in the UMAP vector-space. How much these MMPs actually differ requires further examination. Figure 7.2 inspects two such MMPs in greater detail. The figure is a 1:1 comparison between the two MMPs (MMP1 and MMP2) marked with an "X" in Figure 7.1b. MMP1 and MMP2 both perform well with a loss in KPI compared to the *point-of-reference* (POR) of 2%. Here, the POR is the MMP generated by the BM without any spending constraint. The figure shows how MMP1 consistently spends more on Print than the two other MMPs.

The large quantity of high-performing MMPs as indicated by the histogram in Figure 7.1a could be caused by an ineffectual spending constraint caused by a negligible $\mathrm{B}^{(diversity)}$ spending bound. Figure 7.3 sheds light on how the predicted KPI changes as this bound increases. The dotted line on the figure visualizes the 2% loss in predicted KPI. Each circle on the figure represents the same MMPs as those visualized in the UMAPs in Figure 7.1. From Figure 7.3, it is clear that putting a constraint on Radio immediately causes a drop in performance. This performance declines steadily as the bound increases relative to the total budget (the x-axis). On the other hand, it turns out one can place significant spending requirements on channels like Television and Digital Display without suffering a noteworthy loss in performance. In fact, one can place almost 40% of the total budget on either of these channels before saturating the insertions and experiencing loss. In comparison, the POR MMP dedicates only 22% and 28% of the total budget to these channels, respectively.

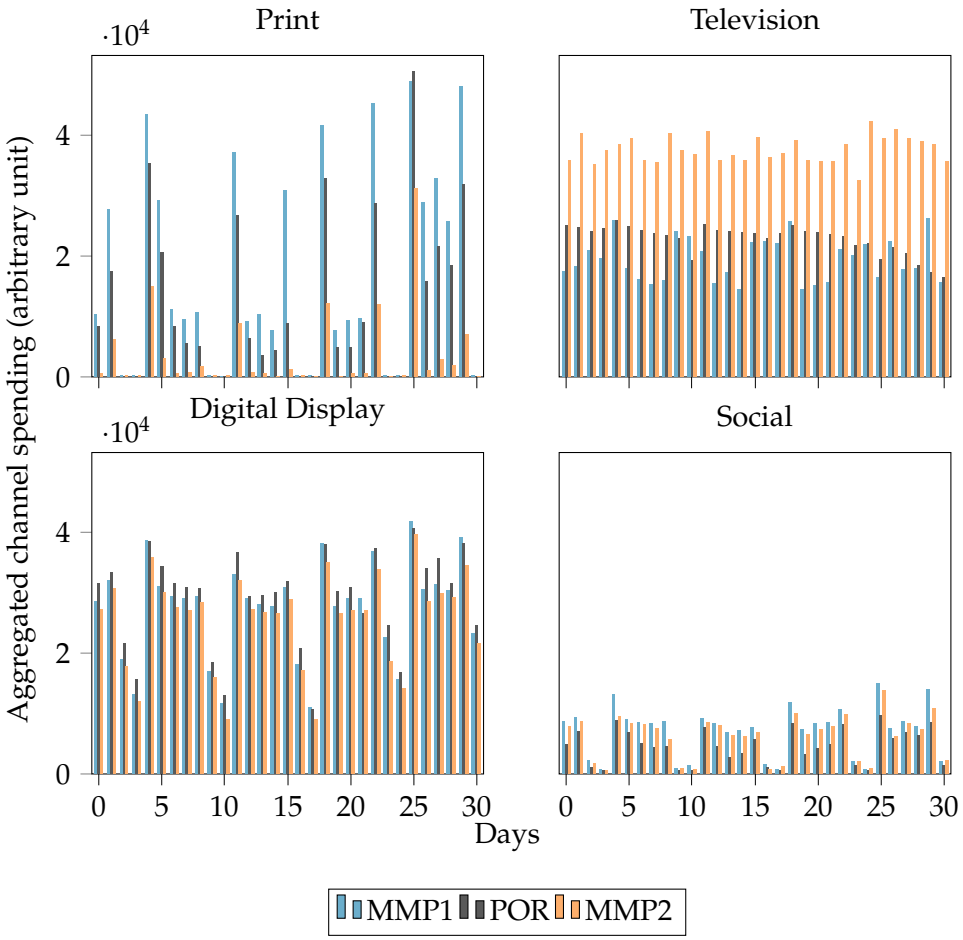


FIGURE 7.2: **A 1:1 comparison between two generated MMPs**. As the MMPs are 31×106 matrices, an extensive comparison of each feature is infeasible. Instead, the figure visualizes patterns in spending at a channel level over the full 31-day period. Each bar chart illustrates this spending for four chosen channels: Print, Television, Digital Display, and Social. MMP1 is the MMP marked in Figure 7.1b with a spending constraint on the channel named Print. Likewise, MMP2 is the highlighted MMP with a spending constraint on Television on the same figure. For comparison, POR is the point of reference. This is the MMP generated by the BM and corresponds to the black asterisk from Figure 7.1b. MMP1 and MMP2 are both high-performing MMPs with a loss in KPI compared to POR of 2%. As expected, MMP1 has a clear tendency to spend more on Print than both the POR and MMP2. Moreover, the increased spending on Print reflects the seasonal effect. Since MMP1 spends more on printed advertisements than POR and MMP2, it will necessarily have to decrease spending on other channels in order to comply with the same total budget as the other MMPs. Clearly, the channel Television is one such channel for which MMP1 has decreased spending compared to both POR and MMP2. In contrast, the optimization procedure decided on the same spending pattern on Digital Display and Social for MMP1 as the other two MMPs. The same properties hold for MMP2 but with an increase on Television instead.

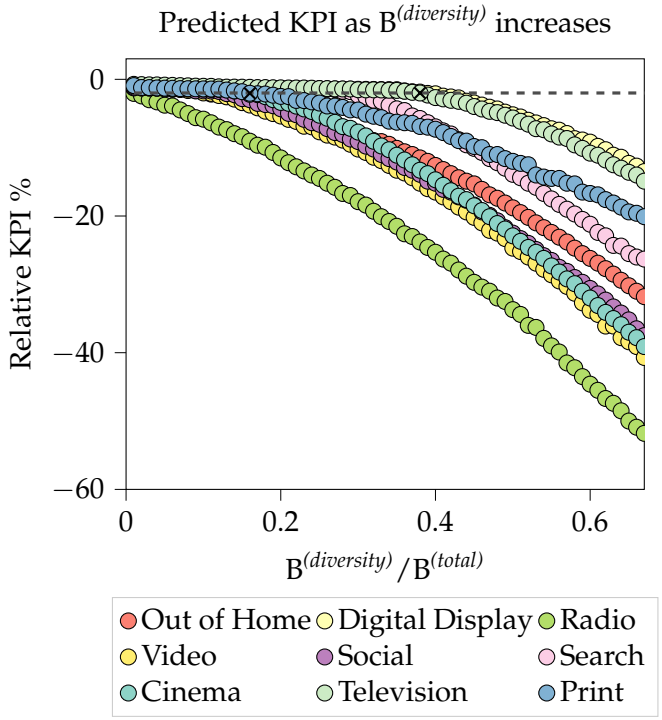


FIGURE 7.3: **The predicted KPI as the minimum spending constraint increases its bound on a channel.** The circles represent the same MMPs as in the UMAPs from Figure 7.1. Each time the optimization is repeated with a minimum spending constraint on a channel, the bound $B^{(diversity)}$ is increased relative to the fixed total budget $B^{(total)}$. Eventually, the increased bound leads to a saturation of all insertions covered by the constraint's channel. In turn, the optimization reduces the spending on other insertions in order to comply with the total budget. This eventually leads to a decline in the BM's predicted KPI for the resulting MMP. The dotted line indicates the 2% KPI loss relative to the BM's generated MMP for the same total budget but without any minimum spending constraint. The two circles marked with an "X" indicates the same MMPs as those marked in Figure 7.1b and visualized in Figure 7.2.

7.5 Discussion

As stated in Section 7.1, the pursuit of user-preference-based MMPs is, in this work, addressed by exploring a set of high-quality, diverse MMPs. This set of MMPs should efficiently cover the solution space and exhibit diversity through consistent trends to provide transparency to the user. As was shown in Figure 7.1a, the method proposed in this chapter does indeed achieve a large set of MMPs. $\approx 1/6$ of the 900 MMPs perform on par with the default MMP obtained using the BM and no constraint on spending. Hence, a large set of high-quality solutions has been obtained using this approach. Figure 7.3 visualizes which of these MMPs are performing well, i.e., on which channel was the constraint set and with what bound relative to the total budget $\mathrm{B}^{(diversity)}/\mathrm{B}^{(total)}$. The figure shows how the spending on some channels (e.g., Television) can be pushed toward much larger spending than the default, unconstrained solution without notable losses. In contrast, increasing the spending on other channels (e.g., Radio) instantly yields MMPs with declining KPI. These curvatures indicate that some channels allow for a broader spectrum of spending that permits a high degree of diversity, whereas others are more limited in options before declining in performance.

This leaves the question of whether these high-performing MMPs exhibit diversity with clear, identifiable trends. In Figure 7.1b, the segmentation of the plane into clusters of channels is clear. This segmentation indicates a high degree of local structure in UMAP. This signifies that two MMPs generated with a spending constraint on the same channel exhibit in same overall tendencies in spending. While two such MMPs express the same spending tendencies, they do not have the exact same embedding in the UMAP. This is caused by differences in their solution matrix. Despite the two having a minimum spending constraint on the same channel, the bound at which the constraint is satisfied, $\mathrm{B}^{(diversity)}$, differs. UMAP's ability to correctly identify and place any two MMPs with the same $\mathcal{F}^{(diversity)}$ in the same cluster is significant. It aids the interpretability and opportunities within the field of HAII.

The clustering of MMPs is, however, not consistent in all areas of the UMAP. In Figure 7.1b, it is evident that the middle cluster containing the white diamond is "incorrectly" classified by the SVM. This cluster contains MMPs with all categories of $\mathcal{F}^{(diversity)}$. In Figure 7.1a, one can see that this cluster contains exclusively high-performing MMPs. The reason for the vibrant cluster lies in the annealing of $\mathrm{B}^{(diversity)}$. In the first iteration of a channel, $\mathrm{B}^{(diversity)}$ is set quite low ($\mathrm{B}^{(total)}/E$). In case the loss landscape of $\mathcal{L}_{\mathrm{P}}$ is shaped such that the unconstrained solution matrix contains spending at $\mathcal{F}^{(diversity)}$ equal to or exceeding $\mathrm{B}^{(total)}/E$, the solution is invariant to the diversity constraint. For this reason, the results of the first couple of iterations are potentially unaffected by the minimum spending constraint. Hence, some of the MMPs are largely indistinguishable.

Next, the global structure of the UMAP is examined. The global structure of

a dimensionality-reduced dataset is about *relatedness* between clusters and inter-cluster distances. Two MMPs located in separate clusters of the UMAP should indicate structural differences in the two underlying matrices. Yet Euclidean distances in the global structure of a manifold can be misleading, and UMAP does not guarantee to preserve inter-cluster distances correctly [292]. Kobak and Linderman [153] argues UMAP is still largely reliant on the specific initialization used and thus hyperparameter dependent. For this reason, one should exercise caution when drawing conclusions based on such distances between clusters without further investigation.

Even when two such MMPs differ significantly and are located in distinct clusters in the manifold with large inter-cluster Euclidean distances, these differences must be transparent to marketing employees. Therefore, to shed light on how these MMPs actually differ, the spending patterns of three MMPs (MMP1, MMP2, and POR) are examined in greater detail in Figure 7.2. The two clusters in which MMP1 and MMP2 belong have a large Euclidean distance in the 2-dimensional manifold. In addition, both MMPs have a high predicted KPI. Figure 7.2 clearly indicates the structural differences in the trends of the three MMPs. For instance, MMP1 consistently spends more on "Print" than the two other MMPs.

Despite these structural differences between MMP1, MMP2, and POR, the MMPs also have a lot in common. For instance, their spending pattern on the two channels "Digital Display" and "Social" is largely identical. Similarly, they all have strong correlations in their spending. On days for which the POR places a relatively high spend, MMP1 and MMP2 also place a relatively high spend. This is a property of the seasonal effect as introduced in section 2.4. The seasonal effect describes how good a day is to run advertisements on. The MMPs shown in Figure 7.2 all strictly adhere to this seasonal effect, as they are striving to optimize for KPI. Hence, obtaining high-quality solutions with respect to predicted KPI adds limitations to the possible diversity in spending patterns. For this reason, a marketing employee disapproving of the underlying model's seasonal effect will have limited options with this setup.

7.6 Future Work

The results from Figure 7.1 and 7.2 demonstrate the existence of high-performing MMPs with explainable differences in their spending trends. The newly acquired ability to create a set of high-quality, diverse MMPs enables novel research opportunities and applications. Future research could involve user studies to examine how marketing employees perceive and interact with this set of diverse MMPs. For instance, one could study whether these employees would pick the default, unconstrained MMP solution as their preferable option, or instead one of the diverse MMPs obtained from the method introduced in this chapter. From a commercial point of view, one can easily construct a UI allowing such marketing employees to interact with the generated UMAP from Figure 7.1. This UI could, for instance, provide a quick comparison between MMPs in the UMAP, similar to Figure 7.2.

The experiments presented in this chapter used a minimum spending constraint on a channel level to obtain diversity. One could easily replace this constraint with another, more sophisticated version. An example could be a "clustering" constraint, restricting the spending to occur only in bursts. Such a constraint could be applied to various extents, similar to the bound associated with the minimum spending constraint. Another example of such a sophisticated constraint could be to penalize variance in the spend on each channel. Such a constraint would be biased toward less volatility in the resulting MMPs. One could even apply a combination of several different types of constraints to obtain even more diversity in the set of MMPs. Such modifications to the diversity constraint could increase variety in the set of generated MMPs and thus accommodate even more types of user preferences.

Lastly, one could consider other groupings of insertions than the channel-based approach taken in this work. Such a change would require only a modification of the proposition $F(i^{(j)}, i^{(j')})$ that was used in Section 7.2.1 to define the grouping into channels based on advertising format. Another such grouping could, for instance, be online versus offline insertions or a region-based grouping, just to name a few possibilities.

7.7 Chapter Summary

This chapter started by motivating the need to pursue diversity as a method to accommodate the latent preferences of marketing employees using BW7's AI-infused platform.

Instead of simply maximizing a distance metric between the set of solution candidates, a different idea based on *pulling* was proposed. At the core of this idea was setting constraints based on groupings of insertions, namely channels. This grouping was based on a conceptualization familiar to marketing employees. Moreover, the concept was easily implementable as an extension to the preexisting loss function.

The setup enabled the optimization to efficiently generate a set of high-quality, diverse MMPs. Several of these differing MMPs obtained a performance on par with the default MMP that was generated without pursuing diversity.

The MMPs generated were clustered using UMAP, which clearly separated the MMPs. These clusters reflected the channel on which the spending restriction was imposed during the optimization of the MMPs.

It was also demonstrated that setting a minimum spending requirement on some channels immediately resulted in a KPI drop, even with a relatively weak bound. However, for other channels, the optimization managed to distribute large sums of spend within the channel in a cost-efficient manner. This, in turn, provided a set of diverse, high-quality solutions.

Lastly, through visual inspection, three MMPs were examined to review how these differences come to light and whether these MMPs exhibit diverse spending trends from a human viewpoint.

The method and results presented in this chapter have been made possible based on the findings in Chapter 3 and 4. These chapters laid the groundwork for approximating a NN to a Bayesian MMM's posterior predictive distribution. This, in turn, can be used to accelerate the MMP optimization procedure. Without this gain in speed, executing the optimization 900 times would be too time-consuming for the method to be usable in the HAII setup considered here.

Part V

Epilogue

Chapter 8

Outlook and Conclusive Remarks

This chapter begins by discussing open problems and potential future avenues for exploration. This is followed by a conclusion based on the main results of this thesis.

8.1 Outlook and Open Questions

The future outlook and open questions discussed here are from both a research oriented perspective and BW7's perspective, i.e., how BW7 can utilize the provided insights as a guide for future directions of their product.

8.1.1 Fragile Human-AI Interaction

Our work with iNNk stressed how fragile HAII can be. The adversarial attacks generated by the players did not require carefully constructed, peculiar data to fool an otherwise high-performing NN. At the same time, the generated drawings were easily recognizable to a human. Our approach from section 6.4 alleviated this issue in the context of iNNk for the identified adversarial player strategies. However, it emphasizes how difficult HAII can be, as users might behave in a creative, unexpected manner.

Addressing adversarial attacks is an ongoing field of research, but most work is looking at auto-generated attack methods. We invite researchers to focus on human generated adversarial attacks methods, and how these might affect HAII. Such research could provide helpful guidelines for researchers and companies aiming for valuable HAII.

8.1.2 Towards Augmented Human-AI Teaming

Closely related to the HAII in iNNk is human-AI teaming in AI-advised constellations. In this case, the human and AI work together. In such setups, the success of the collaboration depends on whether the user has an accurate understanding of the AI's error boundary or not. This accuracy can be reinforced if the AI-decision making is transparent and explainable.

Genuinely explainable AI-advised decision-making is a budding, interdisciplinary subject [118]. Which, when achieved, can aid the human in determining whether to follow the AI-advised decision or not.

One of the most fundamental challenges in HAII occurs when human and AI disagree on a decision. Neither party is able to efficiently communicate their underlying reasoning to the other. In contrast, two humans disagreeing on a decision can engage in a discussion to present each party's point-of-view.

XAI seeks to address such issues through interpretable AI-driven decisions. However, it is still unclear at what point AI-advised decisions are sufficiently explained, and in what format such explanations should be given [70, 118].

A popular approach in XAI is the use of decision trees [177, 179]. Such trees are based on simple rules easy to visualize. Thus, one gets the impression of transparency in the decision-making. However, such decision trees only answer the question of *how* a decision is made, not *why* the decision is made in that particular way [69].

This thesis takes a different approach to augment the human-AI teaming experience and performance. The goal is not the provide condensed explanations of the underlying reason behind an AI-decision. Instead, the aims to adapt to user preferences through a set of high quality diverse solutions.

The approach successfully applied in this work was only possible due to the domain-dependent concept of which marketing employees are familiar, and easily implementable in the AI. Future research could investigate other approaches to adapt to user preferences in the context of marketing. One approach could be to leverage a user's historical data.

8.1.3 Design Considerations for Blackwood Seven

In Chapter 5, we identified three design considerations for NNs in games:

1) Use flow to structure the learning curve of HAII. This design consideration could be incorporated to BW7's platform by introducing the users to successively more sophisticated BMs. This could be done by first introducing the most simplistic BM without advanced features such as carryover or seasonal effect (see Chapter 2.4). This can aid users to improve their mental model of the AI, and gain intuitive understanding of how each component of the BM alters the generated MMP.

2) Incorporate enhanced discovery-based learning. The core of this design consideration is to let players play around with the AI. Although the users of BW7's platform already interact with the BM, getting feedback from the model is time consuming, thus limiting the number of interactive sessions before reaching user fatigue [172]. Decreasing the time elapsed when the model processes data can improve the HAII and, in turn, accelerate the onboarding process of new users to the platform. Thus, future works should consider incorporating the method from Part II in their production pipeline.

3) Extend the invitation to play. Highlighting failures of the AI can help the users identify the error boundary of the AI. Highlighting failures of BW7's model could be based on its predictions on historical data that was the most incorrect. This can aid the development of the user's mental model, as it visualizes at which points the BM is incorrect.

8.2 Conclusion

This thesis explored *human-AI interaction* (HAII) and how one can augment the human-AI collaboration in the context of marketing mix decisions. The introduction motivated this research by taking a starting point in *Blackwood Seven* (BW7), the industrial partner of this PhD project. The *artificial intelligence* (AI) developed by BW7 created *mixed-marketing plan*s (MMPs) to optimize other companies' *key performance indicator* (KPI). BW7 estimated that these MMPs created an uplift in KPI of 50-300% compared to manual marketing planning.

The problem was, however, that marketing employees using BW7's AI-infused platform had a hard time relating to the AI's decisions. The new, alternative, AI-based, MMPs seemed too far from the users' past experience of how a marketing plan should look. This, in turn, caused *status quo* bias, inducing the users to neglect or partially deviate from the AI-advice.

We argued that such challenges could be caused by contradictions between the AI's output and the marketing employees' mental models of the marketing landscape.

A way to address these contradictions could be to take the employee using the AI-system into consideration. One could incorporate that person's preferences in the AI-driven optimization. However, quantifying and incorporating preferences in an AI-optimization can be challenging.

As a stepping stone, we started out by addressing a fundamental issue with BW7's platform. Namely, with the current setup, generating a MMP with BW7's AI was a time consuming task. BW7's AI-system is build upon *Bayesian model*s (BMs). BW7's BMs involve a large number of predictors and recursive components. These properties were the root cause of the time consuming MMP generation. Every time a MMP was generated with the BM, it required thousands of risk minimized predictions to be used for gradient descent.

Consequently, to speedup the MMP generation, we first introduced a method to approximate a BM with a *neural network* (NN). This approximation was conducted by learning a point-wise approximation of the BM's posterior predictive distribution given an observation.

We found that making risk minimized predictions with this method scaled better than a fabricated Bayesian regression model as the number of predictors grew large. Our research also demonstrated that *active learning* (AL) can be applied to minimize the size of the dataset needed to perform this approximation.

Subsequently, the method was tested on BW7's platform. With a few adjustments to the setup, the posterior predictive distribution of the industry-grade BM was approximated. The NN was trained to predict the BM's measured KPI when given a MMP as input. The fast, accurate predictions, enabled the NN to generate optimized MMPs up to 65× faster than the BM with a negligible loss in KPI. This application showcased a real-world application of the proposed method.

With faster AI feedback, we were in a good position to study potential improvements to the HAII of BW7's system as a way to remedy the human-AI gap.

To identify prominent directions for such improvements, we explored NN-based games. Games were studied since AI research has an extended history of using games as a rich domain to motivate algorithmic advancements [196, 257, 288]. In addition, there is a significant body of work in games research to understand user experience [36, 62, 178].

Our research identified three design considerations: *1)* Use flow to structure the learning curve of HAII. *2)* Incorporate enhanced discovery-based learning. *3)* Extend the invitation to play.

Applying these design considerations to BW7's setup led to a set of suggested future avenues for their platform. These propositions were all heavily reliant on getting faster AI feedback using our NN approximation to prevent user fatigue. Incorporating these design considerations in BW7's system can lead to improved *user experience* (UX) design and ease the onboarding process of new users to BW7's platform.

This was followed by examining the game iNNk as a casestudy for player-NN interaction. INNk was developed, as it has many similarities to the AI setup of BW7. These similarities were identified through a framework for classifying the use of NNs in games.

The casestudy highlighted how fragile HAII can be. As users interacted with the NN, their mental model of the AI's error boundary grew accurate, and they managed to consistently stump the NN. Based on our findings, we presented a data-efficient method to mend the NN in case the HAII becomes too unbalanced. However, the essential issue remains: User and NN classifying the same data differently can easily transpire.

This issue is in line with the users of BW7 not being able to relate to the AI generated MMPs. What the AI deems a good plan is by the human classified as irregular. To remedy this situation, a search for quality diversity in the AI-generated MMPs was sought for. This approach also functioned as a stepping stone towards incorporating user preferences in the marketing planning process. By seeking quality diversity, we assumed the existence of more than one good solution for the same marketing task. Exploring for quality diversity required repeated MMP generation using the AI.

Part IV discussed various ways to achieve such quality diversity in the field of marketing. The approach taken here was based on a level of abstraction meaningful

to marketing employees and easily implementable in the AI optimization, namely *marketing channels*. Using this concept, we applied spending constraints to the MMP generation. This led to a set of diverse MMPs. From visual inspection, these differences between the generated MMPs were unfold. The visualizations showed consistent differences in spending trends easily interpretable to humans. At the same time, several of these diverse MMPs performed on par with the default MMP.

In this manner, we addressed the lack of user preferences in the AI optimization indirectly through quality diversity. Future research can use this technique to design the UX for such a system. Improvements to the UX design should trigger improved human-AI engagement by presenting the user with this set of diverse, high-performing solutions.

Appendices

Appendix A

Adjustment of the Data Sampling Dropout-Rate

In this appendix, we will elaborate on some empirical results for determining the best setting for a specific hyperparameter used for the experiments in Chapter 3 and 4 and, by extension, also in Chapter 7, namely the experiments with approximating a NN to a BM.

The hyperparameter under loop was used to assist the NN in learning invariant features in the examined domains. While there are several approaches to learning invariants, such as tangent propagation [25] and weight sharing [48], the experiments conducted in this work do so through the data sampling process.

Common to all domains examined in the aforementioned chapters is that the effect of $\mathbf{x}_j$ on $\mathbf{y}$ is invariant to $\mathbf{x}_{\neq j}$. Inferring this invariance is particularly challenging for predictors having a low effect on $\mathbf{y}$.

Using the data sampling procedure as presented in Algorithm 1, we argue that the use of setting $\mathbf{x}_j = 0$ with probability Bern (τ) helps in learning these invariants.

To show τ's effect on the fitted NN, we choose two predictors, j and j', from the industry-grade Bayesian *marketing mixture model* (MMM) introduced in Chapter 4. The predictor j has a low effect on $\mathbf{y}$ (i.e., a low marketing effect), thus making it hard for the NN to single out this effect, while j' has a relatively large impact on $\mathbf{y}$.

Starting with predictor j, to determine the NN's performance on the invariants present in the domain with respect to this predictor, we are interested in the *relative* effect $\mathbf{x}_j$ has on the NN's output.

Let

$$y = \frac{1}{M} \sum_{m=1}^{M} \left(g_\phi \left(\mathbf{x} \right) - g_\phi \left(\mathbf{x}' \right) \right)_m$$

where g is the NN function with parameters ϕ and $\mathbf{x}' = \mathbf{x}$ except for $\mathbf{x}'_j = 0$, and set $\mathbf{x}_{\neq j}$ to a constant, c.

Then, five NNs are fitted to a dataset with 100 000 entries. Each NN is trained on a separate dataset. These datasets differ, as they are sampled using a distinct value for τ, with $\tau \in \{0.2, 0.4, 0.6, 0.8, 1\}$.

To visually inspect how well each of the five NNs accounts for the invariances in the domain for predictor j, we use the NN for predictions. These predictions are carried out for $\mathbf{x}_j \in \{0, 1\}$ and are repeated for each $c \in \{0, 0.5, 1\}$.

Since the NN has a temporal component, date encoding is needed for the NN input. For simplicity, a single, random day was sampled for these experiments and kept fixed throughout.

The result is shown at the top row of Figure A.1. These experiments are repeated for predictor j', which is illustrated in the bottom row of the figure.

Learning the invariant properties of this domain to perfection would result in curves being identical across all values for c and all lying on the dashed black curve. Performing well on this experiment is generally hard, as two of the chosen values for c are at the extremes (0 and 1). These extremes were included because the subsequent input optimization for MMP generation tends to push some of the predictors to such extremes.

As can be seen in Figure A.1, disabling the "dropout" effect of the data sampling procedure completely (i.e., $\tau = 1$) performs worse compared to $\tau = 0.6$ and $\tau = 0.8$. This is an observation carried out by comparing the predicted response curves to the target (the black dotted line). In fact, for $\tau = 1.0$ and $c = 0$, the NN has no perception of the effect $\mathbf{x}_j$ has on y. The same case holds for j'.

Likewise, setting τ too low biases the NNs to underestimate the predictors' effect. In addition, this renders the data sampling procedure inefficient, so the sampled data become more sparse.

Based on these experiments, it was concluded that setting $\tau = 0.8$ provides the best performance across various cs when evaluating both j and j'. This value for τ was therefore used in all subsequent experiments.

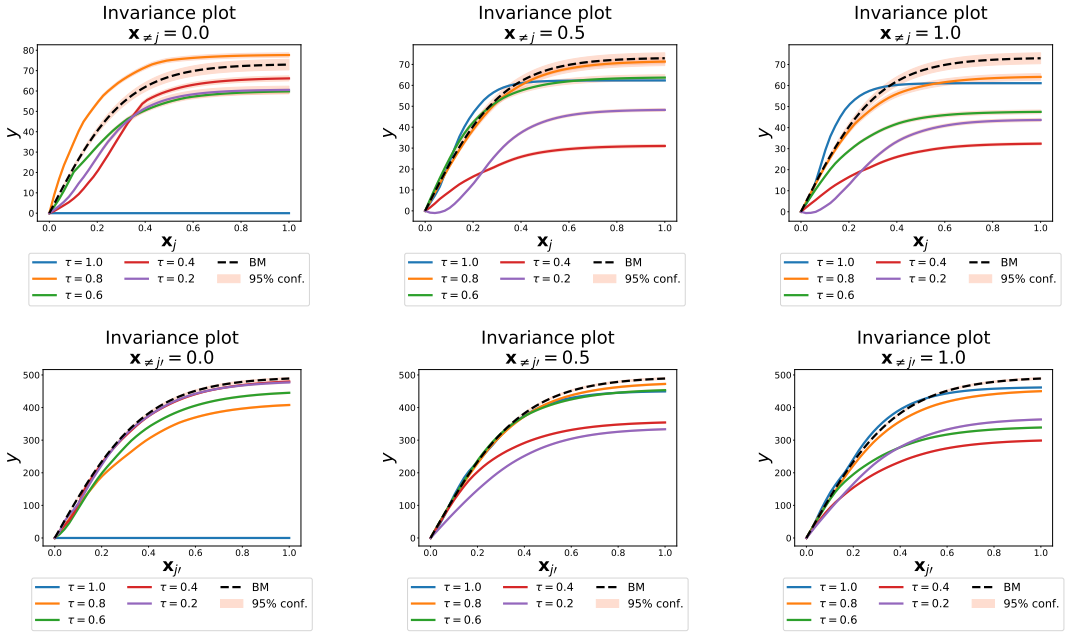


FIGURE A.1: Experiments for determining the NNs' inference on the invariants in the domain. We show five trained NNs for various values of τ for predictor j with $\mathbf{x}_{\neq j} = c$ with $c \in \{0, 0.5, 1\}$. Predictor j (top row) has a relatively weak impact on y, making it harder for the NN to learn its effect. In contrast, predictor j' (bottom row) has a higher impact, also reflected on the y-axis. Across both predictors and all values of c, the NN trained with $\tau = 0.8$ yields the best overall performance.

Appendix B

Game Analysis Table

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
2D Walk Evolution [194]	Indie	Simulation	No	Controls creature movement	Behaviors	Offline	Teammate	NN-Specific
AI Dungeon [290]	Indie	Adventure	No	Creates natural language responses	Behaviors	Offline	Designer	NN-Limited
AIvolution [266]	Indie	Simulation	Unknown	Controls creature movement	Behaviors	Online	Teammate	NN-Limited
Audioin-Space* [9]	Research	Action	Yes	Creates weapon visuals and audio	Content	Online	Designer	NN-Agnostic
Black & White [11]	AAA	Role-play	Yes	Creates creature desires	Behaviors	Online	Apprentice	NN-Agnostic
Blitzkrieg 3 [207]	AAA	Strategy	Unknown	Controls "Boris" battle behavior	Behaviors	Offline	Competitor	NN-Limited
BrainCrafter* [222]	Research	Puzzle	No	Controls robot movement	Behaviors	Online	Apprentice	NN-Specific UI

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
Colin McRae Rally 2.0* [47]	AAA	Sports	Unknown	Controls the car's driving performance	Behaviors	Offline	Competitor	NN-Agnostic
Competitive Snake [250]	Indie	Puzzle	No	Controls enemy snake behavior	Behaviors	Offline	Competitor	NN-Agnostic
Corral [254]	Indie	Simulation	Unknown	Controls chicken movement and preservation skills	Behaviors	Online	Apprentice	NN-Agnostic
Creatures [104]	AAA	Role-play	Yes	Controls the creature's sensor-motor coordination	Behaviors	Online	Apprentice	NN-Agnostic
Darwin's Avatar* [168]	Research	Action	No	Controls creature movement	Content	Offline	Designer	NN-Limited

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
Democracy 3 [85]	AAA	Role-play	Unknown	Creates motivations and desires of the public	Behaviors	Offline	Designer	NN-Agnostic
Dr. Derk's Mutant Battlegrounds [268]	Indie	Simulation	Unknown	Controls creature movement and behavior	Behaviors	Online	Apprentice	NN-Limited
Evo-Commander* [137]	Research	Simulation	No	Controls tank movement and shooting behavior	Behaviors	Online	Apprentice	NN-Specific
Evolution [66]	Indie	Simulation	Unknown	Controls creature movement	Behaviors	Online	Teammate	NN-Specific
Evolution for Beginners [155]	Indie	Simulation	Unknown	Controls creature movement and sensory input	Behaviors	Online	Apprentice	NN-Limited

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
Football Evo [39]	Indie	Simulation	No	Controls player movement and behavior	Behaviors	Online	Apprentice	NN-Limited
Forza Car Racing [95, 267, 270]	AAA	Sports	Unknown	Controls the car's driving performance	Behaviors	Online	Competitor	NN-Limited
GAR [115]	Research	Action	Yes	Creates particle weapons	Content	Online	Designer	NN-Limited
Gridworld [67]	Indie	Simulation	Unknown	Controls creature behavior	Behaviors	Online	Designer	NN-Limited
Guess the Word [5, 93]	Research	Puzzle	Yes	Creates natural language responses	Behaviors	Offline	Teammate	NN-Limited
Hey Robot [83]	Indie	Puzzle	No	Controls language processing	Behaviors	Offline	Teammate	NN-Agnostic

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
How to Train Your Snake [21]	Indie	Simulation	No	Controls snake movement	Behaviors	Online	Apprentice	NN-Specific
Idle Machine Learning Game [285]	Indie	Simulation	No	Controls performance of the vehicle's movement	Behaviors	Online	Apprentice	NN-Specific
iNNk [287]	Research	Puzzle	No	Identifies sketches drawn by the player	Behaviors	Offline	Competitor	NN-Specific
Machine Learning Arena* [78]	Research	Simulation	Unknown	Controls robot behavior	Behaviors	Online	Teammate	NN-Specific
MotoGP19 [198]	AAA	Sports	Unknown	Controls the car's driving performance	Behaviors	Offline	Competitor	NN-Agnostic
Neat Race [211]	Indie	Simulation	No	Controls car movement	Behaviors	Online	Apprentice	NN-Specific

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
NERO [150, 264]	Research	Simulation	No	Controls robot movement and shooting behavior	Behaviors	Online	Apprentice	NN-Specific
Oui Chef!! [44, 45]	Research	Role-play	No	Controls chef behavior	Behaviors	Online	Apprentice	NN-Agnostic
Petalz* [237]	Research	Simulation	No	Creates flowers	Content	Offline	Designer	NN-Agnostic
Quick, Draw! [102]	Research	Puzzle	No	Identifies sketches drawn by the player	Behaviors	Offline	Teammate	NN-Limited
Race for the Galaxy [86]	AAA	Strategy	Unknown	Controls opponent behavior	Behaviors	Offline	Competitor	NN-Limited
Roll for the Galaxy [87]	AAA	Strategy	Unknown	Controls opponent behavior	Behaviors	Offline	Competitor	NN-Limited

Continued on next page

Table B.1: Overview of the 38 NN games and the results of the analysis. (*Games without available playable versions.) (Continued)

Game Characteristics			NN Characteristics			Player-NN Framework Characteristics		
Game Title	**Publisher**	**Genre**	**Multiple AIs?**	**NN's Role**	**NN Output**	**Learning**	**Interaction Metaphor**	**UI**
Semantris [103]	Research	Puzzle	No	Controls the classification of words	Behaviors	Offline	Teammate	NN-Limited
Supreme Commander 2 [228]	AAA	Strategy	Yes	Controls enemy unit flight and fight behavior	Behaviors	Offline	Competitor	NN-Agnostic
The Abbattoir Intergrade [13]	Indie	Strategy	Unknown	Controls enemy unit offense behavior	Behaviors	Online	Competitor	NN-Agnostic

Bibliography

[1] Mahdieh Abbasi and Christian Gagné. "Robustness to adversarial examples through an ensemble of specialists". In: *arXiv preprint arXiv:1702.06856* (2017).

[2] Vero Vanden Abeele, Katta Spiel, Lennart Nacke, Daniel Johnson, and Kathrin Gerling. "Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences". In: *International Journal of Human-Computer Studies* 135 (2020), p. 102370.

[3] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Database Theory — ICDT 2001*. Ed. by Jan Van den Bussche and Victor Vianu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434. ISBN: 978-3-540-44503-6.

[4] Philip E Agre. *Computation and human experience*. Cambridge University Press, 1997.

[5] IBM Research AI. *Guess the Word*. Website. 2020. URL: https://word-game-ui-dev.mybluemix.net/.

[6] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W. Ku, and A. Nguyen. "Strike (With) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4840–4849. DOI: 10.1109/CVPR.2019.00498.

[7] Louis Alfieri, Patricia J Brooks, Naomi J Aldrich, and Harriet R Tenenbaum. "Does discovery-based instruction enhance learning?" In: *Journal of educational psychology* 103.1 (2011), p. 1.

[8] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. "Guidelines for human-ai interaction". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–13.

[9] Amy K. Hoover, William Cachia, Antonios Liapis, Georgios N. Yannakakis. *Audioinspace*. Research game. 2015.

[10] Craig G Anderson, Jen Dalsen, Vishesh Kumar, Matthew Berland, and Constance Steinkuehler. "Failing up: How failure in a game environment promotes learning through discourse". In: *Thinking Skills and Creativity* 30 (2018), pp. 135–144.

[11] Electronic Arts. *Black & White*.

$$CD - ROM$$

. 2001.

[12] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016). arXiv: 1607.06450.

[13] Jared Bagley. *The Abbattoir Intergrade*. itch.io. 2018. URL: https://cabrill.itch.io/ai.

[14] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. "Beyond accuracy: The role of mental models in human-AI team performance". In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 7. 1. 2019, pp. 2–11.

[15] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. "Updates in Human-AI Teams: Understanding and Addressing the Performance/Compatibility Tradeoff". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 2429–2437. DOI: 10.1609/aaai.v33i01.33012429. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4087.

[16] Frank M. Bass and Darral G. Clarke. "Testing Distributed Lag Models of Advertising Effect". In: *Journal of Marketing Research* 9.3 (1972), pp. 298–308. ISSN: 00222437. URL: http://www.jstor.org/stable/3149541.

[17] Mohsen Bayati, Mark Braverman, Michael Gillam, Karen M. Mack, George Ruiz, Mark S. Smith, and Eric Horvitz. "Data-Driven Decisions for Reducing Readmissions for Heart Failure: General Methodology and Case Study". In: *PLOS ONE* 9.10 (Oct. 2014), pp. 1–9. DOI: 10.1371/journal.pone.0109264. URL: https://doi.org/10.1371/journal.pone.0109264.

[18] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[19] Philipp Benz, Chaoning Zhang, and In So Kweon. *Batch Normalization Increases Adversarial Vulnerability: Disentangling Usefulness and Robustness of Model Features*. 2020. arXiv: 2010.03316 [cs.LG].

[20] Chester Bernard. *The Function of the Executive, Harvard*. Cambridge, mass, 1938.

[21] Bewelge. *How to Train Your Snake*. 2017. URL: https://bewelge.itch.io/how-to-train-your-snake.

[22] Ankita Bhatia, Arti Chandani, and Jagriti Chhateja. "Robo advisory and its potential in addressing the behavioral biases of investors — A qualitative study in Indian context". In: *Journal of Behavioral and Experimental Finance* 25 (2020), p. 100281. ISSN: 2214-6350. DOI: https://doi.org/10.1016/j.jbef.2020.100281. URL: https://www.sciencedirect.com/science/article/pii/S2214635019302394.

[23] Reuben Binns, Max Van Kleek, Michael Veale, Ulrik Lyngs, Jun Zhao, and Nigel Shadbolt. "'It's Reducing a Human Being to a Percentage' Perceptions of Justice in Algorithmic Decisions". In: *Proceedings of the 2018 Chi conference on human factors in computing systems*. 2018, pp. 1–14.

[24] Christopher M Bishop. *Bayesian methods for neural networks*. Aston University, 1995.

[25] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[26] Christopher M. Bishop. "Regularization and complexity control in feed-forward networks". English. In: *Proceedings International Conference on Artificial Neural Networks ICANN'95*. International Conference on Artificial Neural Networks ICANN'95. EC2 et Cie, 1995, pp. 141–148. ISBN: 2-910085-19-8.

[27] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[28] John Blitzer, Mark Dredze, and Fernando Pereira. "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification". In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 440–447.

[29] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[30] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning". eng. In: *SIAM review* 60.2 (2018), pp. 223–311. ISSN: 0036-1445.

[31] George E. P. Box. "Science and Statistics". In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799. DOI: 10.1080/01621459.1976.10480949.

[32] Phelim Boyle, Mark Broadie, and Paul Glasserman. "Monte Carlo methods for security pricing". In: *Journal of economic dynamics and control* 21.8-9 (1997), pp. 1267–1321.

[33] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: http://github.com/google/jax.

[34] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[35] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=B1xsqj09Fm.

[36] Jeanne H Brockmyer, Christine M Fox, Kathleen A Curtiss, Evan McBroom, Kimberly M Burkhart, and Jacquelyn N Pidruzny. "The development of the Game Engagement Questionnaire: A measure of engagement in video game-playing". In: *Journal of Experimental Social Psychology* 45.4 (2009), pp. 624–634.

[37] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[38] Joanna J Bryson. "Robots should be slaves". In: *Close Engagements with Artificial Companions: Key social, psychological, ethical and design issues* (2010), pp. 63–74.

[39] Buddhaman. *Football Evo*. itch.io. 2018. URL: https://buddhaman.itch.io/football-evo.

[40] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. "Explainable AI in Fintech Risk Management". In: *Frontiers in Artificial Intelligence* 3 (2020), p. 26. ISSN: 2624-8212. DOI: 10.3389/frai.2020.00026. URL: https://www.frontiersin.org/article/10.3389/frai.2020.00026.

[41] Gordon Calleja. "Digital game involvement: A conceptual model". In: *Games and culture* 2.3 (2007), pp. 236–260.

[42] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. "Deep blue". In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83.

[43] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[44] Gabriele Cimolino. *Oui Chef!!* itch.io. 2018. URL: https://anna-mae.itch.io/oui-chef.

[45] Gabriele Cimolino, Sam Lee, Quentin Petraroia, and TC Nicholas Graham. "Oui, Chef!!: Supervised Learning for Novel Gameplay with Believable AI". In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. 2019, pp. 241–246.

[46] Darral G Clarke. "Econometric measurement of the duration of advertising effect on sales". In: *Journal of Marketing Research* 13.4 (1976), pp. 345–357.

[47] Codemasters. *Colin McRae Rally 2.0*. Playstation. 2000.

[48] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 160–167. ISBN: 9781605582054. DOI: 10.1145/1390156.1390177. URL: https://doi.org/10.1145/1390156.1390177.

[49] Michael Cook, Mirjam Eladhari, Andy Nealen, Mike Treanor, Eddy Boxerman, Alex Jaffe, Paul Sottosanti, and Steve Swink. "PCG-Based Game Design Patterns". In: *arXiv preprint arXiv:1610.03138* (2016).

[50] M. Correa, C. Bielza, and J. Pamies-Teixeira. "Comparison of Bayesian networks and artificial neural networks for quality detection in a machining process". In: *Expert Systems with Applications* 36.3, Part 2 (2009), pp. 7270–7279. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2008.09.024.

[51] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning*. 1995, pp. 273–297.

[52] Kristof Coussement and Koen W. De Bock. "Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning". In: *Journal of Business Research* 66.9 (2013). Advancing Research Methods in Marketing, pp. 1629–1636. ISSN: 0148-2963. DOI: https://doi.org/10.1016/j.jbusres.2012.12.008. URL: https://www.sciencedirect.com/science/article/pii/S0148296312003530.

[53] Christian Arzate Cruz and Jorge Adolfo Ramirez Uresti. "Player-centered game AI from a flow perspective: Towards a better understanding of past trends and future directions". In: *Entertainment Computing* 20 (2017), pp. 11–24.

[54] Balázs Csanád Csáji et al. "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24.48 (2001), pp. 7–7.

[55] Mihaly Csikszentmihalyi. *Flow: The psychology of optimal experience*. Vol. 1990. Harper & Row New York, 1990.

[56] Geng Cui, Man Leung Wong, and Xiang Wan. "Targeting High Value Customers While Under Resource Constraint: Partial Order Constrained Optimization with Genetic Algorithm". In: *Journal of Interactive Marketing* 29 (2015), pp. 27–37. ISSN: 1094-9968. DOI: https://doi.org/10.1016/j.intmar.2014.09.001. URL: https://www.sciencedirect.com/science/article/pii/S1094996814000462.

[57] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. "Robots that can adapt like animals". In: *Nature* 521.7553 (May 2015), pp. 503–507. ISSN: 1476-4687. DOI: 10.1038/nature14422.

[58] Wang Dawei, Deng Limiao, Ni Jiangong, Gao Jiyue, Zhu Hongfei, and Han Zhongzhi. "Recognition pest by image-based transfer learning". In: *Journal of the Science of Food and Agriculture* 99.10 (2019), pp. 4524–4531. DOI: https://doi.org/10.1002/jsfa.9689. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jsfa.9689. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jsfa.9689.

[59] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[60] Alena Denisova and Paul Cairns. "The Placebo Effect in Digital Games: Phantom Perception of Adaptive Artificial Intelligence". In: *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. CHI PLAY '15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 23–33. ISBN: 9781450334662. DOI: 10.1145/2793107.2793109. URL: https://doi.org/10.1145/2793107.2793109.

[61] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural safety* 31.2 (2009), pp. 105–112.

[62] Heather Desurvire and Charlotte Wiberg. "Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration". In: *International conference on online communities and social computing*. Springer. 2009, pp. 557–566.

[63] Floris Devriendt, Jeroen Berrevoets, and Wouter Verbeke. "Why you should stop predicting customer churn and start using uplift models". In: *Information Sciences* 548 (2021), pp. 497–515. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.12.075. URL: https://www.sciencedirect.com/science/article/pii/S0020025519312022.

[64] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[65] Pedro Domingos. "A Few Useful Things to Know about Machine Learning". In: *Commun. ACM* 55.10 (Oct. 2012), pp. 78–87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755. URL: https://doi.org/10.1145/2347736.2347755.

[66] Keiwan Donyagard. *Evolution*. 2019. URL: https://keiwan.itch.io/evolution.

[67] DopplerFrog. *Gridworld*. Steam. 2015. URL: https://store.steampowered.com/app/396890/Gridworld/.

[68] Derek Doran, Sarah Schulz, and Tarek R. Besold. *What Does Explainable AI Really Mean? A New Conceptualization of Perspectives*. 2017. arXiv: 1710.00794 [cs.AI].

[69] Derek Doran, Sarah Schulz, and Tarek R. Besold. "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives". In: *CoRR* abs/1710.00794 (2017). arXiv: 1710.00794. URL: http://arxiv.org/abs/1710.00794.

[70] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. "Explainable artificial intelligence: A survey". In: *41st International Convention on Information and Communication Technology, Electronics and Microelectronics*. 2018, pp. 0210–0215. DOI: 10.23919/MIPRO.2018.8400040.

[71] Graham Dove, Kim Halskov, Jodi Forlizzi, and John Zimmerman. "UX design innovation: Challenges for working with machine learning as a design material". In: *Proceedings of the 2017 chi conference on human factors in computing systems*. 2017, pp. 278–288.

[72] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[73] Rosana El Jurdi, Caroline Petitjean, Paul Honeine, Veronika Cheplygina, and Fahed Abdallah. "High-level prior-based loss functions for medical image segmentation: A survey". In: *Computer Vision and Image Understanding* 210 (2021), p. 103248. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2021.103248. URL: https://www.sciencedirect.com/science/article/pii/S1077314221000928.

[74] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[75] Entropy. *Marketing Mix Modelling Econometrics consultancy*. 2020. URL: https://entropyconsulting.io/marketing-mix-modelling-econometrics/.

[76] Richard Evans and Emily Short. "Versu—a simulationist storytelling system". In: *IEEE Transactions on Computational Intelligence and AI in Games* 6.2 (2013), pp. 113–130.

[77] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[78] Grant Ferguson. *Machine Learning Arena*. Research game. 2019.

[79] The Advertising Research Foundation. *The Advertising Research Foundation Reveals Ground Breaking Research: How Advertising Works Today*. Mar. 2016. URL: https://www.prnewswire.com/news-releases/the-advertising-research-foundation-reveals-ground-breaking-research-how-advertising-works-today-300235257.html (visited on 11/04/2021).

[80] Yoav Freund and Robert E. Schapire. "Experiments with a New Boosting Algorithm". In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML'96. Bari, Italy: Morgan Kaufmann Publishers Inc., 1996, pp. 148–156. ISBN: 1558604197.

[81] Laura Beth Fulton, Ja Young Lee, Qian Wang, Zhendong Yuan, Jessica Hammer, and Adam Perer. "Getting Playful with Explainable AI: Games with a Purpose to Improve Human Understanding of AI". In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–8.

[82] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. *Batch Normalization is a Cause of Adversarial Vulnerability*. 2019. arXiv: 1905.02161 [cs.LG].

[83] Everybody House Games. *Hey Robot*. Website. 2019. URL: https://everybodyhousegames.com/heyrobot.html.

[84] Firaxis Games. *Sid Meier's Civilization® VI*. 2016. URL: https://civilization.com/.

[85] Positech Games. *Democracy 3*. Steam. 2013. URL: https://store.steampowered.com/app/245470/Democracy_3/.

[86] Temple Gates Games. *Race for the Galaxy*. Steam. 2017. URL: https://store.steampowered.com/app/893840/Roll_for_the_Galaxy/.

[87] Temple Gates Games. *Roll for the Galaxy*. Steam. 2020. URL: https://store.steampowered.com/app/893840/Roll_for_the_Galaxy/.

[88] Yuqing Gao and Khalid M. Mosalam. "Deep Transfer Learning for Image-Based Structural Damage Recognition". In: *Computer-Aided Civil and Infrastructure Engineering* 33.9 (2018), pp. 748–768. DOI: https://doi.org/10.1111/mice.12363. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12363. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12363.

[89] Yashesh Gaur, Walter S. Lasecki, Florian Metze, and Jeffrey P. Bigham. "The Effects of Automatic Speech Recognition Quality on Human Transcription Latency". In: *Proceedings of the 13th International Web for All Conference*. W4A '16. Montreal, Canada: Association for Computing Machinery, 2016. ISBN: 9781450341387. DOI: 10.1145/2899475.2899478. URL: https://doi.org/10.1145/2899475.2899478.

[90] James Paul Gee. "What video games have to teach us about learning and literacy". In: *Computers in Entertainment (CIE)* 1.1 (2003), pp. 20–20.

[91] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[92] Abraham P George and Warren B Powell. "Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming". In: *Machine learning* 65.1 (2006), pp. 167–198.

[93] Katy Ilonka Gero, Zahra Ashktorab, Casey Dugan, Qian Pan, James Johnson, Werner Geyer, Maria Ruiz, Sarah Miller, David R Millen, Murray Campbell, et al. "Mental Models of AI Agents in a Cooperative Game Setting". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–12.

[94] F.A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM". In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[95] Jonathan M Gitlin. *War Stories: How Forza learned to love neural nets to train AI drivers*. Sept. 2020. URL: https://arstechnica.com/gaming/2020/09/war-stories-how-forza-learned-to-love-neural-nets-to-train-ai-drivers/.

[96] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, 13–15 May 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

[97] Negin Golrezaei, Hamid Nazerzadeh, and Paat Rusmevichientong. "Real-time optimization of personalized assortments". In: *Management Science* 60.6 (2014), pp. 1532–1551.

[98] Daniel Gomme and Richard Bartle. "Strategy Games : The Components of A Worthy Opponent". In: *Proceedings of 2020 Foundation of Digital Games*. 2020.

[99] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[100] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[101] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *CoRR* abs/1412.6572 (2015).

[102] Google. *Quick, Draw!* online. Nov. 2016. URL: https://quickdraw.withgoogle.com (visited on 01/17/2021).

[103] Google. *Semantris*. 2018. URL: https://research.google.com/semantris/.

[104] Stephen Grand, Dave Cliff, and Anil Malhotra. "Creatures: Artificial life autonomous software agents for home entertainment". In: *Proceedings of the first international conference on Autonomous agents*. 1997, pp. 22–29.

[105] Alex Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011, pp. 2348–2356.

[106] Alex Graves. "Supervised sequence labelling". In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.

[107] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: IEEE, May 2013, pp. 6645–6649. ISBN: 978-1-4799-0356-6. DOI: 10.1109/icassp.2013.6638947. URL: http://dx.doi.org/10.1109/icassp.2013.6638947.

[108] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924.

[109] Zvi Griliches. "Distributed Lags: A Survey". In: *Econometrica* 35.1 (1967), pp. 16–49. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1909382.

[110] Emmanuel Guardiola. "The gameplay loop: a player activity model for game design and analysis". In: *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*. 2016, pp. 1–7.

[111] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. "Countering Adversarial Images using Input Transformations". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=SyJ7ClWCb.

[112] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O Riedl. "Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–13.

[113] David Ha and Douglas Eck. "A neural representation of sketch drawings". In: *arXiv preprint arXiv:1704.03477* (2017).

[114] Lars Kai Hansen and Peter Salamon. "Neural network ensembles". In: *IEEE transactions on pattern analysis and machine intelligence* 12.10 (1990), pp. 993–1001.

[115] Erin J Hastings, Ratan K Guha, and Kenneth O Stanley. "Evolving content in the galactic arms race video game". In: *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE. 2009, pp. 241–248.

[116] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[117] José Miguel Hernández-Lobato and Ryan P. Adams. "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks". In: *arXiv:1502.05336* (2015). arXiv: 1502.05336 [stat.ML].

[118] Michael Hind. "Explaining Explainable AI". In: *XRDS* 25.3 (Apr. 2019), pp. 16–19. ISSN: 1528-4972. DOI: 10.1145/3313096. URL: https://doi.org/10.1145/3313096.

[119] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (2012). URL: http://arxiv.org/abs/1207.0580.

[120] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". eng. In: *Neural computation* 9.8 (Dec. 1997), pp. 1735–1780. ISSN: 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

[121] Matthew D Hoffman and Andrew Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1593–1623.

[122] Sean D. Holcomb, William K. Porter, Shaun V. Ault, Guifen Mao, and Jin Wang. "Overview on DeepMind and Its AlphaGo Zero AI". In: *Proceedings of the 2018 International Conference on Big Data and Education*. ICBDE '18. Honolulu, HI, USA: Association for Computing Machinery, 2018, pp. 67–71. ISBN: 9781450363587. DOI: 10.1145/3206157.3206174.

[123] Lars Erik Holmquist. "Intelligence on tap: artificial intelligence as a new design material". In: *interactions* 24.4 (2017), pp. 28–33.

[124] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[125] Feng-Hsiung Hsu. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. USA: Princeton University Press, 2002. ISBN: 0691090653.

[126] Ye Hu, Leonard Lodish, and Abba Krieger. "An Analysis of Real World TV Advertising Tests: A 15Year Update". In: *Journal of Advertising Research* 47 (Sept. 2007). DOI: 10.2501/S0021849907070353.

[127] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. "Harnessing Deep Neural Networks with Logic Rules". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2410–2420. DOI: 10.18653/v1/P16-1228. URL: https://aclanthology.org/P16-1228.

[128] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. "Snapshot ensembles: Train 1, get m for free". In: *arXiv preprint arXiv:1704.00109* (2017).

[129] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 7304–7308.

[130] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[131] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. *Black-box Adversarial Attacks with Limited Queries and Information*. 2018. arXiv: 1804.08598 [cs.CV].

[132] "In AI we trust? Perceptions about automated decision-making by artificial intelligence". In: *AI and Society* 35.3 (2020), pp. 611–623. ISSN: 14355655. DOI: 10.1007/s00146-019-00931-w. URL: https://doi.org/10.1007/s00146-019-00931-w.

[133] DMC Insights. *Benefits & Limitations of Multi-Touch Attribution Model*. English. 2019. URL: http://dmcinsight.com/attribution-modelling/benefits-and-limitations-of-multi-touch-attribution-model/.

[134] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[135] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France:

PMLR, July 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[136] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[137] Daniel Jallov, Sebastian Risi, and Julian Togelius. "EvoCommander: A novel game based on evolving and switching between artificial brains". In: *IEEE Transactions on Computational Intelligence and AI in Games* 9.2 (2016), pp. 181–191.

[138] Jie Jia, Honggang Zhou, and Yunchun Li. "Using Deep Neural Network Approximate Bayesian Network". In: *arXiv:1801.00282* (2017).

[139] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[140] Yuxue Jin, Yueqing Wang, Yunting Sun, David Chan, and Jim Koehler. *Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects*. Tech. rep. Google Inc., 2017.

[141] Jeff Johnson. "Chapter 12 - Human Decision-Making is Rarely Rational". In: *Designing with the Mind in Mind (Third Edition)*. Ed. by Jeff Johnson. Third Edition. Morgan Kaufmann, 2021, pp. 203–223. ISBN: 978-0-12-818202-4. DOI: https://doi.org/10.1016/B978-0-12-818202-4.00012-X. URL: https://www.sciencedirect.com/science/article/pii/B978012818202400012X.

[142] Natalie A. Jones, Helen Ross, Timothy Lynam, Pascal Perez, and Anne Leitch. "Mental models: An interdisciplinary synthesis of theory and methods". In: *Ecology and Society* 16.1 (2011). ISSN: 17083087. DOI: 10.5751/ES-03802-160146.

[143] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2 (Nov. 1999), pp. 183–233. ISSN: 0885-6125. DOI: 10.1023/A:1007665907178. URL: https://doi.org/10.1023/A:1007665907178.

[144] Niels Justesen and Sebastian Risi. "Learning macromanagement in starcraft from replays using deep learning". In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. 2017, pp. 162–169. DOI: 10.1109/CIG.2017.8080430.

[145] Jesper Juul. *The art of failure: An essay on the pain of playing video games*. MIT press, 2013.

[146] Ece Kamar, Severin Hacker, and Eric Horvitz. "Combining Human and Machine Intelligence in Large-Scale Crowdsourcing". In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '12. Valencia, Spain: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 467–474. ISBN: 0981738117.

[147] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Alias-Free Generative Adversarial Networks". In: *Proc. NeurIPS*. 2021.

[148] William Karush. "Minima of Functions of Several Variables with Inequalities as Side Conditions". MA thesis. Chicago, IL, USA: Department of Mathematics, University of Chicago, 1939.

[149] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. "Time2vec: Learning a vector representation of time". In: *arXiv preprint arXiv:1907.05321* (2019).

[150] Kenneth O Stanley, Bobby D Bryant, Risto Miikkulainen. *NERO*. Website. 2005. URL: http://nn.cs.utexas.edu/nero/.

[151] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations*. 2014. arXiv: 1312.6114 .

[152] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[153] Dmitry Kobak and George C. Linderman. "UMAP does not preserve global structure any better than t-SNE when using the same initialization". In: *bioRxiv* (2019). DOI: 10.1101/2019.12.19.877522. eprint: https://www.biorxiv.org/content/early/2019/12/19/2019.12.19.877522.full.pdf. URL: https://www.biorxiv.org/content/early/2019/12/19/2019.12.19.877522.

[154] John F. Kolen and Stefan C. Kremer. "Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies". In: *A Field Guide to Dynamical Recurrent Networks*. 2001, pp. 237–243. DOI: 10.1109/9780470544037.ch14.

[155] Nils Krautkremer. *evolution for beginners*. Steam. 2020. URL: https://store.steampowered.com/app/1276350/evolution_for_beginners/.

[156] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[157] Dirk P. Kroese, T. Brereton, T. Taimre, and Z. Botev. "Why the Monte Carlo method is so important today". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6 (2014), pp. 386–392.

[158] Steve Krug. *Don't make me think!: a common sense approach to Web usability*. Pearson Education India, 2000.

[159] Fabian Krüger, Sebastian Lerch, Thordis Thorarinsdottir, and Tilmann Gneiting. "Predictive Inference Based on Markov Chain Monte Carlo Output". In: *International Statistical Review* (2020). DOI: https://doi.org/10.1111/insr.12405. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/insr.12405.

[160] John Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.

[161] H. W. Kuhn and A. W. Tucker. "Nonlinear programming". In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, 1951, pp. 481–492.

[162] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. "Tell me more? the effects of mental model soundness on personalizing an intelligent agent". In: *Conference on Human Factors in Computing Systems - Proceedings* May 2014 (2012), pp. 1–10. DOI: 10.1145/2207676.2207678.

[163] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial Machine Learning at Scale*. 2017. arXiv: 1611.01236 [cs.CV].

[164] Google Creative Lab. *The Quick, Draw! Dataset*. online. Aug. 2018. URL: https://github.com/googlecreativelab/quickdraw-dataset (visited on 01/18/2021).

[165] Effie L-C Law, Florian Brühlmann, and Elisa D Mekler. "Systematic review and validation of the game experience questionnaire (geq)-implications for citation and reporting practice". In: *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. 2018, pp. 257–270.

[166] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

[167] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[168] Dan Lessin and Sebastian Risi. "Darwin's Avatars: A Novel Combination of Gameplay and Procedural Content Generation". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 2015, pp. 329–336.

[169] Fei Fei Li, Justin Johnson, and Serena Young. *Lecture notes in Computer Vision*. Stanford University School of Engineering. May 2017. URL: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf.

[170] Yong Liu, Jorge Laguna, Matt Wright, and Hua He. "Media mix modeling – A Monte Carlo simulation study". In: *Journal of Marketing Analytics* 2.3 (Sept. 2014), pp. 173–186. ISSN: 2050-3326. DOI: 10.1057/jma.2014.3.

[171] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. "Richer Convolutional Features for Edge Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[172] Zhicheng Liu and Jeffrey Heer. "The Effects of Interactive Latency on Exploratory Visual Analysis". In: *Visualization and Computer Graphics, IEEE Transactions on* 20 (Dec. 2014), pp. 2122–2131. DOI: 10.1109/TVCG.2014.2346452.

[173] Christos Louizos and Max Welling. "Multiplicative normalizing flows for variational bayesian neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2218–2227.

[174] Mathias Löwe, Per Lunnemann Hansen, and Sebastian Risi. "Combining Deep Learning and Bayesian Models for Swift Generation of Mix-Marketing Plans". Submitted for peer review at the Elsevier Journal Intelligent Systems with Applications. 2021.

[175] Mathias Löwe, Per Lunnemann, and Sebastian Risi. "Rapid Risk Minimization with Bayesian Models Through Deep Learning Approximation". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9534258.

[176] Mathias Löwe, Jennifer Villareale, Evan Freed, Aleksanteri Sladek, Jichen Zhu, and Sebastian Risi. "Dealing with Adversarial Player Strategies in the Neural Network Game INNk through Ensemble Learning". In: *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*. FDG'21. Montreal, QC, Canada: Association for Computing Machinery, 2021. ISBN: 9781450384223. DOI: 10.1145/3472538.3472540. URL: https://doi.org/10.1145/3472538.3472540.

[177] Octavio Loyola-González, Andres Eduardo Gutierrez-Rodríguez, Miguel Angel Medina-Pérez, Raúl Monroy, José Francisco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa, and Milton García-Borroto. "An Explainable Artificial Intelligence Model for Clustering Numerical Databases". In: *IEEE Access* 8 (2020), pp. 52370–52384. DOI: 10.1109/ACCESS.2020.2980581.

[178] Andrés Lucero, Jussi Holopainen, Elina Ollila, Riku Suomela, and Evangelos Karapanos. "The playful experiences (PLEX) framework as a guide for expert evaluation". In: *Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces*. 2013, pp. 221–230.

[179] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* 2.1 (Jan. 2020), pp. 56–67. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0138-9.

[180] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. Citeseer. 2013, p. 3.

[181] David J.C. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[182] David JC MacKay. "Bayesian interpolation". In: *Neural computation* 4.3 (1992), pp. 415–447.

[183] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. "A Simple Baseline for Bayesian Uncertainty in Deep Learning". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[184] *Marketing Channels in the Supply Chain*. URL: https://courses.lumenlearning.com/boundless-marketing/chapter/marketing-channels-in-the-supply-chain/.

[185] Michael Mateas. "An Oz-centric review of interactive drama and believable agents". In: *Artificial intelligence today*. Springer, 1999, pp. 297–328.

[186] Michael Mateas. "Expressive AI: A semiotic analysis of machinic affordances". In: *3rd Conference on Computational Semiotics for Games and New Media*. 2003, p. 58.

[187] Michael Mateas and Andrew Stern. "Façade: An experiment in building a fully-realized interactive drama". In: *Game developers conference*. Vol. 2. 2003, pp. 4–8.

[188] Graham McAllister and Gareth White. "Video Game Development and User Experience". In: Dec. 2010, pp. 107–128. ISBN: 978-1-84882-962-6. DOI: 10.1007/978-1-84882-963-3_7.

[189] Richard McElreath. *Statistical rethinking: A Bayesian course with examples in R and Stan*. CRC press, 2020.

[190] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. "UMAP: Uniform Manifold Approximation and Projection". In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861. URL: https://doi.org/10.21105/joss.00861.

[191] Darren McManus. *Google Analytics Measurement Planning: A Not So Quick Guide*. Sept. 2019. URL: https://www.google-analytics.ie/blog/google-analytics-measurement-planning-guide/.

[192] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[193] Mindware. *Family Charades In-A-Box*. Board Game. Roseville, MN, 1990.

[194] MiorSoft. *2D Walk Evolution*. itch.io. 2020. URL: https://miorsoft.itch.io/2d-walk-evolution.

[195] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996. ISBN: 0262133164.

[196] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[197] Nir Morgulis, Alexander Kreines, Shachar Mendelowitz, and Yuval Weisglass. *Fooling a Real Car with Adversarial Traffic Signs*. 2019. arXiv: 1907.00374 [cs.CR].

[198] MotoGP. *MotoGP19*. Steam. 2019. URL: https://2019.motogpvideogame.com/.

[199] Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. "A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media". In: *Complex Networks and Their Applications VIII*. Ed. by Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha. Cham: Springer International Publishing, 2020, pp. 928–940. ISBN: 978-3-030-36687-2.

[200] Chelsea M. Myers, Jiachi Xie, and Jichen Zhu. *A Game-Based Approach for Helping Designers Learn Machine Learning Concepts*. 2020. eprint: arXiv:2009.05605.

[201] Pavel Myshkov and Simon Julier. "Posterior distribution analysis for Bayesian inference in neural networks". In: *Workshop on Bayesian Deep Learning, NIPS*. 2016.

[202] Lennart Nacke, Anders Drachen, Kai Kuikkaniemi, Joerg Niesenhaus, Hannu J Korhonen, Wouter M Hoogen, Karolien Poels, Wijnand A IJsselsteijn, and Yvonne AW De Kort. "Playability and player experience research". In: *Proceedings of digra 2009: Breaking new ground: Innovation in games, play, practice and theory*. DiGRA. 2009.

[203] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[204] A. Nguyen, J. Yosinski, and J. Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640.

[205] JAKOB NIELSEN. "Chapter 5 - Usability Heuristics". In: *Usability Engineering*. Ed. by JAKOB NIELSEN. San Diego: Morgan Kaufmann, 1993, pp. 115–163. ISBN: 978-0-12-518406-9. DOI: https://doi.org/10.1016/B978-0-08-052029-2.50008-5. URL: https://www.sciencedirect.com/science/article/pii/B9780080520292500085.

[206] Michael A Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[207] Nival. *Blitzkrieg 3*. Steam. 2017. URL: https://store.steampowered.com/app/235380/Blitzkrieg_3/.

[208] Donald Norman. "Some Observations on Mental Models". In: *Mental Models*. Ed. by Dedre Gentner & Albert L Stevens. Cognitive Science. New Jersey: Lawrence Erlbaum Associates, Inc., 1983. Chap. 1, pp. 7–14.

[209] Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. "A survey of real-time strategy game AI research and competition in StarCraft". In: *IEEE Transactions on Computational Intelligence and AI in games* 5.4 (2013), pp. 293–311.

[210] David Opitz and Richard Maclin. "Popular ensemble methods: An empirical study". In: *Journal of artificial intelligence research* 11 (1999), pp. 169–198.

[211] Oxygenium. *Neat Race*. itch.io. 2020. URL: https://oxygenium.itch.io/neat-race.

[212] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[213] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64.

[214] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 582–597. DOI: 10.1109/SP.2016.41.

[215] Bambang Parmanto, Paul W Munro, and Howard R Doyle. "Improving committee diagnosis with resampling techniques". In: *Advances in neural information processing systems*. 1996, pp. 882–888.

[216] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training Recurrent Neural Networks". eng. In: (2012).

[217] A Pavone, J Svensson, A Langenberg, U Höfel, S Kwak, N Pablant, RC Wolf, et al. "Neural network approximation of Bayesian models for the inference of ion and electron temperature profiles at W7-X". In: *Plasma Physics and Controlled Fusion* 61.7 (2019).

[218] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. "Uncertainty in Neural Networks: Approximately Bayesian Ensembling". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 234–244.

[219] Derrick Pemberton, Zhiguo Lai, Lotus Li, Shitong Shen, Jue Wang, and Jessica Hammer. "AI or Nay-I? Making Moral Complexity More Accessible". In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. CHI PLAY '19 Extended Abstracts. Barcelona, Spain: Association for Computing Machinery, 2019, pp. 281–286. ISBN: 9781450368711. DOI: 10.1145/3341215.3358248. URL: https://doi.org/10.1145/3341215.3358248.

[220] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. "Machine learning for targeted display advertising: transfer learning in action". In: *Machine Learning* 95.1 (Apr. 1, 2014), pp. 103–127. ISSN: 1573-0565. DOI: 10.1007/s10994-013-5375-2. URL: https://doi.org/10.1007/s10994-013-5375-2.

[221] William D Perreault and E Jerome McCarthy. *Basic marketing: A global managerial approach*. McGraw-Hill/Irwin, 2002.

[222] Peter Greve, Jan Piskur. Sebastian Risi. *Braincrafter*. Research game. 2014.

[223] Martin Pichl, Eva Zangerle, and Günther Specht. "Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation." In: *Grundlagen von Datenbanken*. 2014, pp. 35–40.

[224] Georg Pólya. "Über den zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung und das Momentenproblem". In: *Mathematische Zeitschrift* 8.3-4 (1920), pp. 171–181.

[225] Lutz Prechelt. "Early Stopping - But When?" In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69. ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_3.

[226] Lutz Prechelt. "Early stopping-but when?" In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[227] William H Press, William T Vetterling, Saul A Teukolsky, and Brian P Flannery. *Numerical recipes*. Vol. 818. Cambridge university press Cambridge, 1986.

[228] Steven Rabin. *Game AI pro 2: collected wisdom of game AI professionals*. AK Peters/CRC Press, 2015.

[229] Sonja Radas and Steven M Shugan. "Seasonal marketing and timing new product introductions". In: *Journal of Marketing Research* 35.3 (1998), pp. 296–315.

[230] Emilee Rader, Kelley Cotter, and Janghee Cho. "Explanations as mechanisms for supporting algorithmic transparency". In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–13.

[231] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06434.

[232] Urs Ramer. "An iterative procedure for the polygonal approximation of plane curves". In: *Computer Graphics and Image Processing* 1.3 (1972), pp. 244–256. ISSN: 0146-664X. DOI: 10.1016/S0146-664X(72)80017-0. URL: http://www.sciencedirect.com/science/article/pii/S0146664X72800170.

[233] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[234] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1530–1538.

[235] Mark Owen Riedl and Alexander Zook. "AI for game production". In: *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE. 2013, pp. 1–8.

[236] Sebastian Risi, Charles E Hughes, and Kenneth O Stanley. "Evolving plastic neural networks with novelty search". In: *Adaptive Behavior* 18.6 (2010), pp. 470–491. DOI: 10.1177/1059712310379923.

[237] Sebastian Risi, Joel Lehman, David B D'Ambrosio, Ryan Hall, and Kenneth O Stanley. "Petalz: Search-based procedural content generation for the casual gamer". In: *IEEE Transactions on Computational Intelligence and AI in Games* 8.3 (2015), pp. 244–255.

[238] Sebastian Risi, Sandy D. Vanderbleek, Charles E. Hughes, and Kenneth O. Stanley. "How Novelty Search Escapes the Deceptive Trap of Learning to Learn". In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. GECCO '09. Montreal, Québec, Canada: Association for Computing Machinery, 2009, pp. 153–160. ISBN: 9781605583259. DOI: 10.1145/1569901.1569923.

[239] Stephen P. Robbins and Mary Coulter. *Management*. 11th ed. Pearson Education, 2012.

[240] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[241] Gary Everson Robert Angel. *Pictionary*. Board Game. Pawtucket, RI, 1994.

[242] Andrew Ross and Finale Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. New Orleans, Louisiana, USA, 2018.

[243] Peter E Rossi, Greg M Allenby, and Rob McCulloch. *Bayesian statistics and marketing*. John Wiley & Sons, 2012.

[244] Joel Rubinson. "Empirical Evidence of TV Advertising Effectiveness". In: *Journal of Advertising Research* 49 (June 2009). DOI: 10.2501/S0021849909090321.

[245] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[246] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[247] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *INTERSPEECH*. 2014.

[248] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3". In: *PeerJ Computer Science* 2 (2016), e55.

[249] Ian Sample. *Computer says no: why making AIs fair, accountable and transparent is crucial*. Nov. 2017. URL: https://www.theguardian.com/science/2017/nov/05/computer-says-no-why-making-ais-fair-accountable-and-transparent-is-crucial.

[250] Alper Sekerci. *Competitive Snake*. itch.io. 2020. URL: https://alpersekerci.itch.io/competitive-snake.

[251] Yann Semet. "Interactive Evolutionary Computation: a survey of existing theory". In: *University of Illinois* (2002).

[252] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

[253] Xuhui Shao and Lexin Li. "Data-Driven Multi-Touch Attribution Models". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: Association for Computing Machinery, 2011, pp. 258–264. ISBN: 9781450308137. DOI: 10.1145/2020408.2020453. URL: https://doi.org/10.1145/2020408.2020453.

[254] The Sherrinford. *Corral*. Steam. 2018. URL: https://store.steampowered.com/app/833400/Corral/.

[255] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162.

[256] Husna Siddiqui, Elizabeth Healy, and Aspen Olmsted. "Bot or not". In: *2017 12th international conference for internet technology and secured transactions (ICITST)*. IEEE. 2017, pp. 462–463.

[257] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[258] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[259] Sam Snodgrass and Santiago Ontañón. "Experiments in map generation using Markov chains." In: *FDG*. 2014.

[260] Sho Sonoda and Noboru Murata. "Neural network with unbounded activation functions is universal approximator". In: *Applied and Computational Harmonic Analysis* 43.2 (2017), pp. 233–268.

[261] Eduardo D Sontag. "VC dimension of neural networks". In: *NATO ASI Series F Computer and Systems Sciences* 168 (1998), pp. 69–96.

[262] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[263] MC-Stan. *RStan*. Oct. 9, 2020. URL: https://mc-stan.org/.

[264] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. "Evolving neural network agents in the NERO video game". In: *Proceedings of the IEEE* (2005), pp. 182–189.

[265] J. Stewart, D. Clegg, and S. Watson. *Calculus: Early Transcendentals, Metric Edition*. Blue Kingfisher, 2020. ISBN: 9780357113516.

[266] Lykke Studios. *AIvolution*. Steam. 2019. URL: https://store.steampowered.com/app/1014990/Aivolution/.

[267] Microsoft Studios. *Forza Motorsport 5*. Xbox One. 2013.

[268] Mount Rouke Studios. *Dr. Derkś Mutant Battleground*. Steam. 2020. URL: https://store.steampowered.com/app/1102370/Dr_Derks_Mutant_Battlegrounds/.

[269] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Transactions on Evolutionary Computation* 23.5 (Oct. 2019), pp. 828–841. ISSN: 1941-0026. DOI: 10.1109/tevc.2019.2890858. URL: http://dx.doi.org/10.1109/TEVC.2019.2890858.

[270] Dean Takahashi. *How Microsoft's Turn 10 fashioned the A.I. for cars in Forza Motorsport 5 (interview)*. Dec. 2018. URL: https://venturebeat.com/2013/11/06/how-microsofts-turn-10-fashioned-the-ai-for-cars-in-forza-motorsport-5-interview/.

[271] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. "A survey on deep transfer learning". In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279.

[272] Gerard J Tellis. *Effective advertising: Understanding when, how, and why advertising works*. Sage Publications, 2003.

[273] Gerard J Tellis. "Modeling marketing mix". In: *Handbook of marketing research* (2006), pp. 506–522.

[274] Tensorflow. *Recurrent Neural Networks for Drawing Classification*. English. online. tensorflow.org, Dec. 9, 2020. URL: https://github.com/tensorflow/docs/blob/master/site/en/r1/tutorials/sequences/recurrent_quickdraw.md (visited on 01/17/2021).

[275] Nava Tintarev and Judith Masthoff. "A survey of explanations in recommender systems". In: *2007 IEEE 23rd international conference on data engineering workshop*. IEEE. 2007, pp. 801–810.

[276] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[277] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. "A Bayesian data augmentation approach for learning deep models". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2798–2807. arXiv: 1710.10564.

[278] Mike Treanor, Alexander Zook, Mirjam P Eladhari, Julian Togelius, Gillian Smith, Michael Cook, Tommy Thompson, Brian Magerko, John Levine, and Adam Smith. "AI-based game design patterns". In: (2015).

[279] Evgenii Tsymbalov, Maxim Panov, and Alexander Shapeev. "Dropout-based Active Learning for Regression". In: *CoRR* abs/1806.09856 (2018). arXiv: 1806.09856.

[280] Joe Tullio, Anind K. Dey, Jason Chalecki, and James Fogarty. "How it works". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, Apr. 2007, pp. 31–40. ISBN: 9781595935939. DOI: 10.1145/1240624.1240630. URL: https://dl.acm.org/doi/10.1145/1240624.1240630.

[281] Stanford University. *Lecture notes in Computer Vision*. online. 2016. URL: http://cs231n.stanford.edu (visited on 01/20/2021).

[282] Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. "Graph grammar-based controllable generation of puzzles for a learning game about parallel programming". In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 2017, pp. 1–10.

[283] Valve. *Left 4 Dead*. 2008. URL: https://www.l4d.com.

[284] Vladimir N Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[285] vashgard. *Idle Machine Learning Game*. 2019. URL: https://vashgard.itch.io/idle-machine-learning.

[286] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. "Using Centroidal Voronoi Tessellations to Scale Up the Multidimensional Archive of Phenotypic Elites Algorithm". In: *IEEE Transactions on Evolutionary Computation* 22.4 (2018), pp. 623–630. DOI: 10.1109/TEVC.2017.2735550.

[287] Jennifer Villareale, Ana V. Acosta-Ruiz, Samuel Adam Arcaro, Thomas Fox, Evan Freed, Robert C. Gray, Mathias Löwe, Panote Nuchprayoon, Aleksanteri Sladek, Rush Weigelt, Yifu Li, Sebastian Risi, and Jichen Zhu. "INNk: A Multi-Player Game to Deceive a Neural Network". In: *Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 33–37. ISBN: 9781450375870. URL: https://doi.org/10.1145/3383668.3419858.

[288] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[289] Martin J. Wainwright and Michael I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305. ISSN: 1935-8237. DOI: 10.1561/2200000001. URL: http://dx.doi.org/10.1561/2200000001.

[290] Nick Walton. *AI Dungeon*. Website. 2019. URL: https://aidungeon.io/.

[291] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H. Beck. "Deep Learning for Identifying Metastatic Breast Cancer". In: *ArXiv* abs/1606.05718 (2016).

[292] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. "Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization". In: *CoRR* abs/2012.04456 (2020). arXiv: 2012.04456. URL: https://arxiv.org/abs/2012.04456.

[293] Katy Warr. *Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery*. 1st ed. O'Reilly Media, Inc., 2019, p. 246.

[294] James Wexler. "Artificial Intelligence in Games". In: *Rochester: University of Rochester* (2002).

[295] *What are marketing channels?* URL: https://web.archive.org/web/20140812205947/http://blog.mbaco.com/what-are-marketing-channels/.

[296] *What is a Marketing Channel?* URL: https://directiveconsulting.com/resources/glossary/marketing-channel/.

[297] Richard Tait Whit Alexander. *Cranium*. Board Game. Pawtucket, RI, 1997.

[298] Andrew Gordon Wilson. "The Case for Bayesian Deep Learning". In: *arXiv:2001.10995* (2020), pp. 1–6. arXiv: 2001.10995.

[299] Andrew Gordon Wilson and Pavel Izmailov. "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". In: *arXiv:2002.08791* (2020). arXiv: 2002.08791.

[300] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[301] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. "A semantic loss function for deep learning with symbolic knowledge". In: *International conference on machine learning*. PMLR. 2018, pp. 5502–5511.

[302] Ayumi Yamada. "Appreciating art verbally: Verbalization can make a work of art be both undeservedly loved and unjustly maligned". In: *Journal of Experimental Social Psychology* 45.5 (2009), pp. 1140–1143. ISSN: 0022-1031. DOI: https://doi.org/10.1016/j.jesp.2009.06.016. URL: https://www.sciencedirect.com/science/article/pii/S0022103109001474.

[303] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

[304] Qian Yang, Nikola Banovic, and John Zimmerman. "Mapping machine learning advances from hci research to reveal starting places for design innovation". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–11.

[305] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. "Investigating how experienced UX designers effectively work with machine learning". In: *Proceedings of the 2018 Designing Interactive Systems Conference*. 2018, pp. 585–596.

[306] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*. Vol. 2. Springer, 2018.

[307] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 3320–3328. URL: https://proceedings.neurips.cc/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf.

[308] R Michael Young, Mark O Riedl, Mark Branly, Arnav Jhala, RJ Martin, and CJ Saretto. "An architecture for integrating plan-based behavior generation with interactive game environments." In: *J. Game Dev.* 1.1 (2004), pp. 1–29.

[309] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. "Generative Image Inpainting with Contextual Attention". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5505–5514. DOI: 10.1109/CVPR.2018.00577.

[310] Julian Yudelson. "Adapting McCarthy's four P's for the twenty-first century". In: *Journal of Marketing Education* 21.1 (1999), pp. 60–67.

[311] Richard Zeckhauser and William Samuelson. "Status Quo Bias in Decision-Making". In: *Journal of Risk and Uncertainty* 1 (Feb. 1988), pp. 7–59. DOI: 10.1007/BF00055564.

[312] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[313] Y. Zhang, Y. Wei, and J. Ren. "Multi-touch Attribution in Online Advertising with Survival Theory". In: *2014 IEEE International Conference on Data Mining*. 2014, pp. 687–696. DOI: 10.1109/ICDM.2014.130.

[314] Wei Zhao. "Research on the deep learning of the small sample data based on transfer learning". In: *AIP Conference Proceedings* 1864.1 (2017), p. 020018. DOI: 10.1063/1.4992835. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.4992835.

[315] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[316] Jichen Zhu. "Intentional systems and the artificial intelligence (ai) hermeneutic network: Agency and intentionality in expressive computational systems". PhD thesis. Georgia Institute of Technology, 2009.

[317] Jichen Zhu and D Fox Harrell. "Daydreaming with Intention: Scalable Blending-Based Imagining and Agency in Generative Interactive Narrative." In: *AAAI Spring Symposium: Creative Intelligent Systems*. Vol. 156. 2008.

[318] Jichen Zhu, Antonios Liapis, Sebastian Risi, Rafael Bidarra, and G Michael Youngblood. "Explainable AI for designers: A human-centered perspective on mixed-initiative co-creation". In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2018, pp. 1–8.

[319] Jichen Zhu, Jennifer Villareale, Nithesh Javvaji, S. Risi, Mathias Löwe, Rush Weigelt, and C. Harteveld. "Player-AI Interaction: What Neural Network Games Reveal About AI as Play". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021).

[320] Jichen Zhu, Jennifer Villareale, Nithesh Javvaji, Sebastian Risi, Mathias Löwe, Rush Weigelt, and Casper Harteveld. "Player-AI Interaction: What Neural Network Games Reveal About AI as Play". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. ISBN: 9781450380966. URL: https://doi.org/10.1145/3411764.3445307.

[321] Enrico Zio and Nicola Pedroni. *Uncertainty characterization in risk analysis for decision-making practice*. FonCSI, 2012.