Modelling Individual Humans via a Secondary Task Transfer Learning Method

by

Anmol Mahajan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

© Anmol Mahajan, 2021

Abstract

This thesis presents a novel approach towards modelling individual human behaviour on tasks with insufficient data via transfer learning. In most cases, deep neural networks (DNNs) require a great deal of data to train and adapt towards a particular problem. But there exist different tasks in which we do not have sufficient data available to train DNNs from scratch. There are approaches, such as zero-shot and few-shot learning, that can produce high quality DNNs with smaller amounts of data. However, these approaches still assume a large source dataset or a large secondary dataset to guide the transfer of knowledge from a source task to the target task. These are not assumptions that hold true when our goal is to model individual humans, who tend to produce much less data.In this work we present a novel transfer learning method for producing a DNN for modeling the behaviour of a specific individual on an unseen target task, by leveraging a small dataset produced by that same individual on a secondary task. We make use of a specialized transfer learning representation and Monte Carlo Tree Search (MCTS). We demonstrate that our approach outperforms standard transfer learning approaches and other optimization methods on two different human modeling domains.

Preface

This thesis presents an original work by Anmol Mahajan under the supervision of Dr. Matthew Guzdial. This work may be restructured to get published under different research venues in the near future.

To the Count

For teaching me everything I need to know about math.

I think there is a world market for maybe five computers.

– Thomas J. Watson, IBM Chairman, 1943.

Acknowledgements

There are a lot of people to be thankful for during these 2 years of my Master's research. First and foremost, I would like to thank my supervisor Dr. Matthew Guzdial who guided me throughout my research at each and every step and believed in me. His mentorship has been the most crucial component due to which I am able to successfully complete my research. I would also like to thank Alberta Machine Intelligence Institute (AMII) and Mitacs Accelerate Research Funding for supporting my research. Similarly, I would like to thank Compute Canada for providing me with the necessary resources to conduct my research smoothly. Moreover, I would like to thank all the members of GRAIL Research Lab for healthy research discussions which I enjoyed a lot!

I would like to thank Maurice Sevigny and Servus Credit Union for playing an important role throughout my research for providing me with an opportunity to work on a real world project. I would also like to thank my committee members Dr. Levi Lelis and Dr. Matt Taylor for their in-depth questioning and valuable feedback in my Master's defense examination. I would like to extend my thanks to my committee chair Dr. Lili Mou and the entire Computing Science Department for the completion of my program.

Completing research during the pandemic was challenging and I would like to thank my family members Anil Mahajan, Sushma Mahajan and Shagun Mahajan for always supporting me. Moreover, I would like to thank my friends Dhawal Gupta, Katyani Singh, Ashish Mathew, Mayank Verma, Pavan Kulkarni and Tanisha Rima for their constant support. I would like to extend my thanks to my undergraduate friends Himank, Saksham, Vaibhav, Ritwik, Anshul and Kushagar.

Last but not least, I would like to thank Dr. Baidyanath Saha for support-

ing me throughout my initial stay in Canada. Thank you all for being a part of my Master's program, wish you all the best!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep Neural Networks have achieved remarkable success in many domains. Modern artificial neural networks tend to do well when trained on large amounts of data [14]. But on tasks with less data, deep neural networks (DNNs) can struggle to achieve the same level of performance. For example, when one wishes to train a DNN to model a specific individual [1]. For the purpose of this thesis, modelling humans refers to predicting the behaviour of each person individually on a specific task. This refers to being able to understand how beneficial a specific person's behaviour is in a particular scenario. There are cases in which we need specialized trained models for specific individuals; to maintain privacy and security and/or in domains where modelling individuals leads to better performance. The more common approach is to model all individuals in aggregate. However, this method is also less likely to lead to a final model that adequately handles outliers. Due to lack of training data, we are usually forced to model the behaviour of all individuals in aggregate instead of modelling individual behaviour. One solution to this problem would be to employ transfer learning: to adapt a pretrained model to an individual.

Different approaches exist for adapting DNNs trained on larger datasets to smaller ones, including: transfer learning [18], few-shot learning [16], and zero-shot learning [20]. These approaches are useful in transferring the knowledge gained from a source task and applying it on a target task with insufficient data. The source task represents a task consisting of sufficient data on which the DNN is trained initially. The target task represents the problem with less

available data on which the transfer learned model will be applied. Transfer learning focuses on solving a target task using the learnt knowledge from a similar but different task. Few-shot learning refers to the training of neural networks on tasks with less available training data. Zero-shot learning is used in the scenarios where no data related to the target domain is available. All of these approaches generally require re-training a model, or authoring or learning secondary features to guide the adaptation. [6], [10]

It will often be the case that we lack data to train a model to predict the behaviour of an individual for a target task. But there might exist some data belonging to an unrelated secondary task which can be used to predict the behaviour of an individual on the target problem. If we could develop a general method to approximate models of individual behaviour for an unseen target task based on data of that individual doing another task we could solve the problem of model specialization on low data tasks. Such an approach can be used to model the behaviour of each individual specifically on a target task, especially the behaviour of outliers. Outliers represent the set of individuals who represent different behaviour than the typical behaviour observed over a certain task. Moreover, developing such a method will also help ensure privacy as this will allow us to have specifically trained neural networks corresponding to each individual.

We introduce an approach named CE-MCTS for approximating DNNs for modeling individuals on an unseen target task, by employing data of other individuals on that task and data of that specific individual on a secondary task. We call this problem *model specialization*, as it can be understood to be a process to specialize a general model to an unseen individual. This approach combines Monte Carlo Tree Search (MCTS) with a transfer learning representation suited to low data problems. This low data transfer learning representation allows us to approximate new models and MCTS allows us to explore our unbounded representation to optimize these models. We evaluate our approach on 2 distinct domains, outperforming other transfer learning methods particularly on outlier individuals.

1.1 Research Questions and Related Contributions

Under the preceding motivation, this thesis aims towards further understanding modelling individual human behaviour for low data problems. With that objective in consideration, the major research questions of this thesis are as follows:

- How can Artificial Intelligence be used to model individuals specifically in a low-data problem?
- Can existing knowledge belonging to an unrelated secondary task be used to model human behaviour on a target task?
- How can we discover neural networks with better performance for modelling humans in a low-data problem with the help of an unrelated secondary task?
- Are the current optimization techniques sufficient for modelling individual-specific human behaviour?

The related contributions of this thesis are as follows:

- A background survey on combining existing knowledge and connecting it with human behaviour modelling via machine learning.
- A complete step-by-step detailed walk-through over our proposed approach 'Conceptual Expansion via Monte Carlo Tree Search (CE-MCTS)'
- Detailed explanation of two different experimental setups for modelling individual human behaviour.
- Results and analysis based on the evaluation of our proposed approach in comparison with existing approaches.
- Evidence demonstrating how CE-MCTS is better than current existing approaches and overall outperforms them, particularly on outlier individuals.

1.2 Outline

This thesis is divided into 6 parts including the introduction. Following the outline of this thesis, we introduce the readers to the essential background and related work information in Chapter 2. We introduce our proposed approach named 'Conceptual Expansion with Monte Carlo Tree Search (CE-MCTS)' in Chapter 3. Afterwards, in Chapter 4, we familiarize the readers with the experimental setup of the two different domains in which the performance of CE-MCTS is evaluated for modelling individuals in a low-data target problem. In chapter 5, we present the results and analysis of the performance of CE-MCTS with other baseline approaches. The first domain presents individual human behaviour modelling in the financial fitness. The second domain refers to the modelling of human behaviour in game level design. Details of the experimental setup are followed by the introduction to the baselines used for the analysis of the performance of CE-MCTS. Then we present the in-depth evaluation and results highlighting the performance of CE-MCTS in comparison with the prior existing methods. Finally, in Chapter 6, we conclude our thesis with an overview of how our approach proves beneficial in task of model specialization followed by its implications and future work.

Chapter 2

Background

This chapter provides readers with necessary background required to understand the work presented in this thesis. In sections 2.1 and 2.2, we start with a basic introduction of Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs) to provide readers with an overview of the basic architecture of neural networks followed by how they are used for tackling sequential data. In section 2.3, we introduce the readers to Long Short-Term Memory (LSTM) Recurrent Neural Networks. We use a LSTM network as the primary model throughout this thesis. This is followed by a general outline of transfer learning in section 2.4. Transfer learning is used in this thesis to obtain a preliminary trained model for the target problem and lays the foundation of our proposed transfer learning based CE-MCTS approach. In sections 2.5 and 2.6, we introduce readers to background work related to financial time series forecasting and game level designing. These sections provide the readers with a better understanding of the areas on which the work presented in this thesis is evaluated and analyzed. Finally, sections 2.7 and 2.8 introduce readers to an overview of Combinational Creativity (CC) and Monte Carlo Tree Search (MCTS) for the background necessary to understand our approach.

2.1 Artificial Neural Networks

Recently, Artificial Neural Networks (ANNs) have become one of the most popular components of Machine Learning. ANNs are used in almost every domain to train AI capable of learning from huge amounts of data and gen-

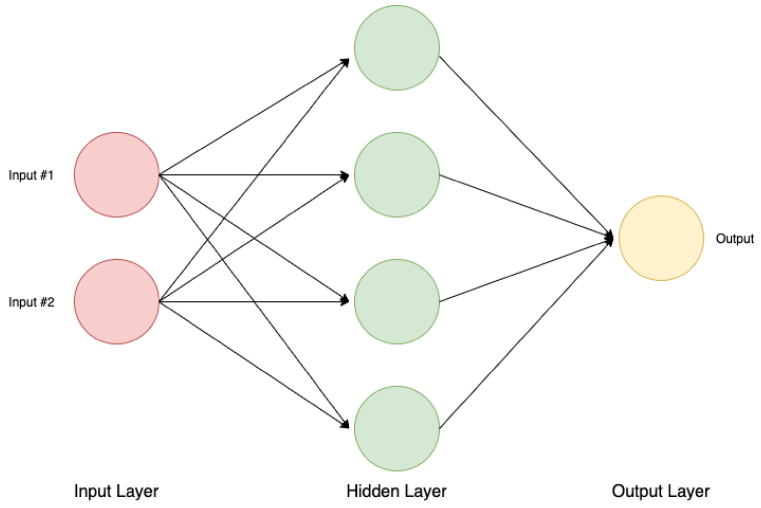

Figure 2.1: Artificial Neural Network (ANN) architecture

eralize over specific types of data. These networks are built using 'neurons' which are responsible for learning the optimization of parameters for minimizing the loss between predictions and true values. In general, 'Loss' defines a penalty for how much the predicted and original labels differ from each other. The training of ANNs consists of pre-processing the input data and passing it through the neural network so that it can be trained by calculating the gradients throughout the training of the neural network. ANNs consist of an interconnected layered structure of neurons which are responsible for processing information. These neurons give better outputs after being trained over large amounts of data by learning hierarchical knowledge.

The structure of an ANN can be divided into 3 parts:

- Input layer
- Hidden layers
- Output layer

We include the architecture of a simple ANN in Figure 2.1.

As after the data is prepared to be processed by a neural network, it is fed to the input layer of ANN. The input layer is designed to be of the same dimensions as that of the data input. This ensures that the input can be fed to

the neural network without any problem due to dimension mismatch. Then, the data is progressed further in the ANN as it moves forward through the hidden layers. There can be any number of hidden layers inside an ANN as it depends on the structure of the network to be used for a specific problem. The hidden layers are followed by an output layer whose dimension matches the size of the output required for a specific problem.

Neurons use activation functions to process the information and provide an output that is passed in as the input to the neurons in the next layer in an ANN. Selection of activation functions to be used in a neural network depends upon the type of problem one is trying to solve. Some examples of activation functions include identity function, sigmoid function and ReLU function. For example, the sigmoid function limits the output values between 0 and 1 and is widely used in classification based problems.

For training the ANN, first the data is pre-processed to the point where it can be fed to the neural network as the input. The data is divided into 3 splits: training set, validation set and the test set. The ANN is then trained on the training set in which the training set input is provided to the network. The model is trained for a fixed number of iterations in which the input data is processed throughout the model and the layered neurons are trained over it. In this process, gradients are calculated and the weights associated with the neurons are updated over the training process. In most of the cases, we have multiple gradients corresponding to each datapoint in a training batch. Therefore, we need to obtain a single gradient to update the neural network. For selection of the gradients during the training process, we use optimizers. Some examples of optimizers include Gradient Descent, Stochastic Gradient Descent, Momentum, and Adagrad. These optimizers are responsible for the gradient selection during the training process of a neural network. For example, in Stochastic Gradient Descent, the gradients are changed after the calculation of loss on every training sample. Moreover, optimizers such as Adam work with first and second order momentum to slow down the speed of gradient search in order to avoid skipping over optimal minimum. The validation set is used to validate the performance of the trained ANN over unseen

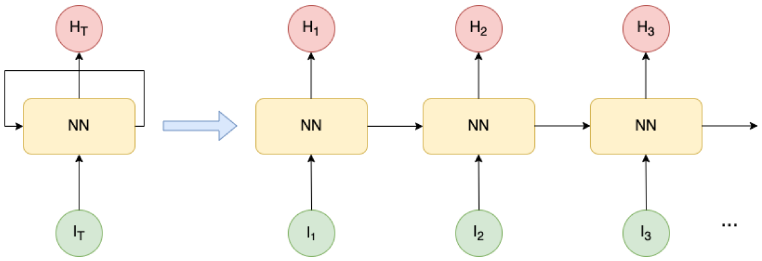

Figure 2.2: Recurrent Neural Network (RNN) architecture

data. After the neural network is trained, it is evaluated over the unseen test data.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) represent a type of neural network designed to handle sequential data. To tackle problems such as word prediction, previous input is valuable for future prediction. In such problems, RNNs are used to carry forward important information of the previous input.

A simple RNN is presented in figure 2.2. In an RNN, the next output is dependent on the prior output. Unlike conventional neural networks, each state of an RNN is provided with a combined input consisting of the current input to be processed and output from the previous states of the RNN carrying knowledge of part of the sequence processed already. This makes RNNs a multiple feedforward neural network capable of generalizing better over sequential datasets.

Simple RNNs lead to a major problem known as 'Vanishing Gradient'. This affects the performance of RNNs in processing longer sequences. In an RNN, as the same weights are used to calculate the final output, these same weights are also used in the multiplication step during backpropagation. As the input in the network progresses through the layers, the error becomes lower and eventually vanishes, having almost no impact. This results in difficulty for the network in learning to process long-term dependencies of the inputs which are further apart in longer sequences. To tackle this problem, more complex approaches such as Long Short-Term Memory Recurrent Neural Net-

works (LSTMs) are used for predicting time series data containing long term dependencies.

2.3 Long Short-Term Memory

As discussed in the last section, Recurrent Neural Networks (RNNs) struggle to provide high quality performance on long term dependent sequential data. To achieve better performance on longer sequences, a network capable of carrying previously learnt information all the way to the end of the sequence is required.

Long Short-Term Memory (LSTM) networks represent a type of Recurrent Neural Network (RNNs) which are capable of processing longer dependencies while carrying forward the processed knowledge all the way to the end in the sequential data. Learning sequence predictions is helpful in different domains such as speech recognition [21], handwriting recognition [19], stock market prediction [15] and machine translation [5].

An LSTM performs similarly as typical Recurrent Neural Networks, with major key differences in the internal operations. The core functionalities of LSTMs are carried out by the cell states and various gates. The gates help the cell states carry relevant information all the way to the end of the sequence. An LSTM network consists of sigmoid activations as gates which are responsible for clipping the output values in the range between 0 and 1. Sigmoid activation helps in the forget gates as the outputs multiplied by values closer to 0 are forgotten as the input progresses through the LSTM network. Sigmoid activation ensures that only relevant and important information is carried along in the LSTM recurrent neural network. LSTM networks also consist of 'tanh' activation for the regulation and maintenance of values between -1 and 1 throughout the network. A simple LSTM network is shown in Figure 2.3.

2.4 Transfer Learning

Transfer learning is a technique used in machine learning which focuses on reusing knowledge learned from one domain to a different, but related, domain.

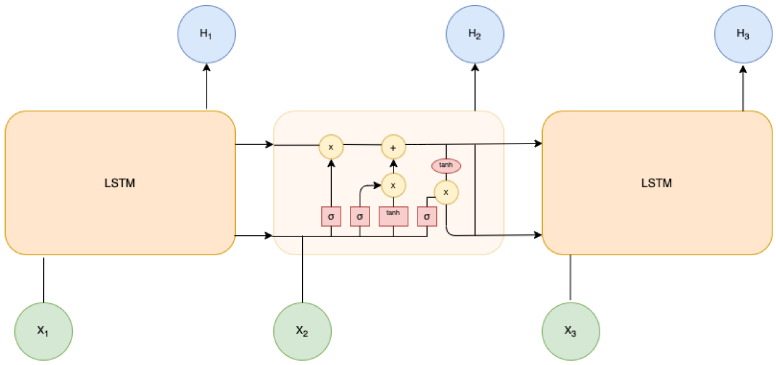

Figure 2.3: Long Short-Term Memory Recurrent Neural Network (LSTM) architecture.

This method is primarily used to transfer knowledge to tasks where there is less available data to achieve better performance than training neural networks from scratch.

Transfer learning can be categorized into 2 types:

Training on a similar problem

In cases where the target problem matches a different problem for which there is an abundance of data available, the neural network can be trained on the source problem and then can transfer the knowledge to be applied on the target task. In such cases, first a problem similar to the target task is identified and initially the model is trained over that. Then this trained neural architecture is used as the base model for the target task and is fine-tuned further to obtain the best performing hyperparameters such as learning rate, batch size, accuracy and loss.

Using a pre-trained neural architecture

For tackling the problems for which finding a similar problem with an abundance of available data is difficult, a pre-trained model can be used as the starting point for the target task. Number of models pre-trained on large data sets can suitably be used to solve the target problem. Once the pre-trained architecture to be used is finalised, it is fine-tuned further to obtain the best values of hyperparameters such as learning rate and batch size.

2.5 Financial Time Series Prediction

Financial time series forecasting is the prediction of future trends based on financial decision-making. Breaking down this term, time series refers to the data represented in a sequential fashion and forecasting means an attempt to predict an upcoming event. When the prediction is made on the sequential data representing a financial event such as trends in stock market and retail price, it is called financial time series forecasting. Other examples of financial time series forecasting include sales predictions [2] and unemployment rate forecasting [17]. Privacy is one of the key components to be considered when modelling humans. There exist techniques in Machine Learning such as federated learning [4], which uses decentralized training of models across different devices without compromising the privacy of data. We view federated learning as complimentary to our approach. One could apply our method in a federated learning scenario by training CE-MCTS with multiple different sources of data for the same individual.

There are different crucial components of time series events which affect the models we can learn. Examples of such components include observed trends over the sequential data, cycles representing the repetitiveness of certain trends and the seasonality of certain fluctuations observed overtime in the sequential data. These components contribute towards how a neural network will obtain knowledge from the sequential data and the overall error that will persist between the actual results and the predicted results.

2.6 Combinational Creativity

Combinational creativity, also known as conceptual combination refers to an efficient, general cognitive ability to recombine existing knowledge to produce new knowledge [3], [9]. Human beings are capable of combining their pre-existing knowledge to understand new concepts [7]. For example, humans can easily break down phrases such as farm cat or a black cat to understand what it means. Combinational creativity aims to explore how recombining pre-existing

knowledge can be used to identify and understand new knowledge.

There exists prior work in which existing concepts are combined computationally to represent new knowledge, but they have traditionally required hand-authored input data. [8] introduced 'Conceptual Blending' to signify how already existing elements or concepts can be mixed and blended together to construct the meaning of the knowledge represented by their meaningful combination. An approach named 'Conceptual Expansion (CE)' was developed to be applied to deep neural networks [13]. CE is a way of representing combinational creativity in which knowledge gained by DNNs can be taken as an input and recombined to represent the combination of existing knowledge. 3.1 represents CE, in which the weights of the DNN model are used as the components that are recombined:

$$\mathrm{CE}(F, \alpha) = \alpha_1 * f_1 + \alpha_2 * f_2 + ... + \alpha_n * f_n \tag{2.1}$$

Where W represents a weight in the final output model, $F = f_1,\ f_2,\ ...,\ f_n$ represents existing weights and α=$\alpha_1,\ \alpha_2,\ ...,\ \alpha_n$ are alpha value filters, which undergo pairwise multiplication with the weights. The alpha filters are used to transform the weights during the combination. The same f value can occur an arbitrary number of times in F with different α values, allowing CE to represent a wide-range of combinations. The f and α values are modified for various output weights or introduce new ones to search over a CE representation.

The work presented in this thesis makes use of conceptual expansion along with Monte-Carlo Tree Search (MCTS). MCTS is well-known as an efficient method for searching large search spaces.

2.7 State Space

The combination of alpha filters and weights leads to an infinite search space consisting of all possible parameter values of a Neural Network represented with Conceptual Expansion (CE). Each state in this state space represents a unique set of Neural Network weights, represented with as a combination of alpha filters and the initial weights.

2.8 Naive Tree Search

Simple tree search is a search approach based on a definite set represented in a tree like structure. In this method, the nodes of the tree represent the data points of a set and the tree search can be used to extract these node values depending upon the type of query associated with the task.

2.9 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a search algorithm in the domain of Artificial Intelligence. Due to Conceptual Expansion (CE), there can be nearly infinite number of possible DNNs that can be explored for model specialization. MCTS combines the naive tree search approach along with balanced exploration and exploitation based policy for discovering better states in large unbounded spaces. Unlike simple tree search, MCTS does not rely on the assumption that the current action is going to be the best possible action every time at a particular step. MCTS algorithm is structured as a tree search approach in which the next possible states are represented as the 'nodes' of the tree. MCTS is beneficial in discovering better possible nodes as throughout the search, it continues to explore better possible alternatives by not only exploiting the existing nodes but also exploring new nodes. MCTS carries out a balanced exploration-exploitation based search with the help of simulating 'roll-outs' of fixed length for a specific number of iterations.

MCTS algorithm consists of 4 major steps. These steps are also demonstrated in the figure below.

Selection

In this step, the MCTS algorithm makes a decision to select the next node that will be used as the starting point for expansion. Here expansion signifies the exploration of undiscovered nodes in the MCTS algorithm. Selection of the node is dependent upon if the exploration or exploitation will be carried out at the current step. This selection procedure is called 'selection policy'. In exploitation, the node with the maximum node value is selected for further

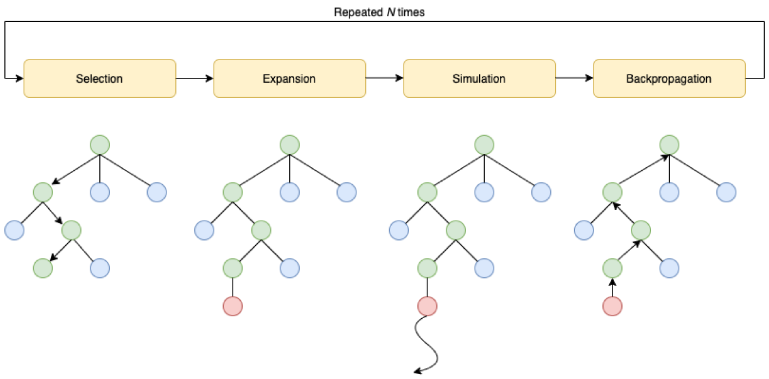

Figure 2.4: Monte Carlo Tree Search (MCTS) algorithm.

expansion and in exploration, a new node is created and expanded further so that the MCTS search does not miss out on the possible better unexplored nodes.

Expansion

A new child node is added or the current best child node is selected in this step on the basis of the selection made in the previous step. The node selected on the basis of exploration of a new node or exploitation of an already existing node becomes the node which will be further expanded in the current roll-out. A new node is only created if the decision of exploration is made in the Selection step. Moreover, exploration is chosen if we have never explored any neighbours of the previously selected node.

Rollout and Fitness

This step involves the implementation of a series of changes of a fixed size of roll-out length that needs to be performed to obtain a final state in the MCTS algorithm for a particular iteration. In a roll-out length, a fixed number of consecutive dependent changes are made until a final state is reached.

At the end of the roll-out, the quality of the final state obtained is estimated based on a pre-decided metric. In our experimental setup, the pre decided metric defines the fitness of a DNN represented at a specific node. This measured value lays the foundation of the final step 'Back-propagation' in which the tree values are updated accordingly.

Back-propagation

In this final step, the values of the nodes are updated all the way up from the final node reached to the original root node. This step makes sure that the node values are up-to-date for the node selection step in the next iteration. For example, in figure 2.4, after we have reached the last node in a roll-out (marked as red), all the fitness values of the parent nodes traversed in that roll-out are updated from bottom to top all the way to the root node (as shown with the upward arrows). Updating the fitness values of the nodes discovered throughout the roll-out is important for reflecting how optimum is the selection of a set of nodes in a roll-out.

Chapter 3

Conceptual Expansion based Monte Carlo Tree Search (CE-MCTS)

In this chapter, we introduce the readers to an in-depth explanation of our approach: Conceptual Expansion-based Monte Carlo Tree Search (CE-MCTS). In section 3.1, we first provide the readers with an overview of the CE-MCTS system. This is followed by an explanation of Conceptual Expansion (CE) in section 3.2 and how it is used to approximate new neural network weights. Section 3.3 provides the readers with an overview of the CE-MCTS search process. Section 3.4 and 3.5 covers the internal structure of the CE-MCTS algorithm, namely how nodes are represented in this search approach and the criteria behind the selection of the final model.

3.1 Model Specialization

In this thesis, our work is focused on the problem of modeling the behaviour of a specific individual on a target task, when we have no record of that individual doing that task. This problem is specifically targeted to the scenarios in which we want to model individuals on tasks with little or no data, but we have sufficient data for these individuals on a secondary task.

For tackling such problems, we assume in this work that we have some available data of the same individual on a similar, secondary task. Moreover, we further assume that we have data on other individuals performing the target

task. We name this problem as 'model specialization'.

3.2 System Overview

The approach described in this work focuses on using the specific individual's secondary task data to guide a novel transfer learning method to fine tune a model trained on other individuals undertaking the target task. This method is named Conceptual Expansion-based Monte Carlo Tree Search (CE-MCTS). Conceptual expansion (CE) provides the representation space of possible fine tuned models, which we then search with the MCTS algorithm. MCTS is well-suited to searching the CE representation as it is an unbounded search space. MCTS is well suited to large search spaces with sparse high values in games such as Chess and Go. CE-MCTS is visualized in Fig 3.1.

CE-MCTS consists of 2 key steps:

- First, we train a model on the available data of other individuals undertaking the target task, which we call the source model.
- Second, we employ the data from the target individual on the secondary task to guide MCTS through the space of re-combinations of weights from the source model. This allows us to approximate a final model for the individual on the the target task without any data of that individual undertaking that task.

3.3 Conceptual Expansion

In this work, conceptual expansion is used to define the search space of possible output models. Conceptual Expansion (CE) is a method of representing combinations of existing knowledge, in this case the weights of our source model. We represent Conceptual Expansion in Equation 3.1:

$$\mathrm{CE}(F, \alpha) = \alpha_1 * f_1 + \alpha_2 * f_2 + ... + \alpha_n * f_n \tag{3.1}$$

Where W represents a weight in the final output model, $F = f_1,\ f_2,\ ...,\ f_n$ represents existing weights and α=$\alpha_1,\ \alpha_2,\ ...,\ \alpha_n$ are alpha value filters, which

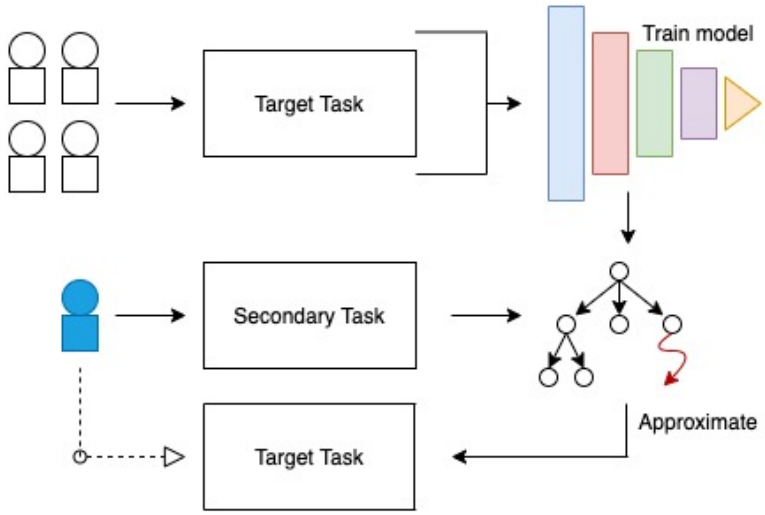

Figure 3.1: The complete CE-MCTS approach. First we train a model on the target task training data for other individuals, which we call a source model. Then we use the behaviour of the individual on a secondary task to guide CE-MCTS to finetune the source model. We then output a final model from this process meant to approximate the behaviour of an individual on the target task, with no record of that individual undertaking that task.

undergo pairwise multiplication with the weights. The alpha filters are used to transform the weights during the combination. The same f value can occur an arbitrary number of times in F with different α values, allowing CE to represent a wide-range of combinations. When we search over a CE representation we modify the f and α values for various output weights or introduce new ones. If CE is a reasonable computational approximation of combinational creativity, it should demonstrate the same ability to efficiently approximate useful combinations for this problem [3], [9]. Combination of the behaviour of individuals on a specific task can prove useful in approximating someone's behaviour on the same task. Combinations can be helpful to approximate the solution to model specialization problems.

3.4 CE-MCTS Search Overview

Conceptual Expansion defines an unbounded search space to explore neural networks. In the problem targeted in this work, there is very little data available which can be used to guide the search for the approximation of individual

Algorithm 1 CE-MCTS

Input:Base prediction model(M_n)
Parameters:Iterations (n), Rollout count (c), Rollout Length (l), Epsilon (ϵ)
Output:Finalised prediction model(O_n)

$t \leftarrow 0$
Root node $\leftarrow M_n$
while $t < n$ **do**
  $t \leftarrow t+1$
  for i in roll-out length **do**
    $P_{current} \leftarrow$ Selection Probability(i)
    if $P_{current} \leq \epsilon$ **then**
      $Node_{current} \leftarrow$ Exploit($currentnodes$)
    else
      $Node_{current} \leftarrow$ Explore($newnodes$)
    end if
  end for
  Backpropagate(c, l)
end while
return O_n

specialized neural networks. We can also expect the space to demonstrate a sparsity in terms of high quality models because of the large unbounded space that needs to be explored to discover better possible models for modelling individuals specifically. For exploration of such a sparse and unbounded space, a balance between exploration and exploitation is crucial. Thus we draw on Monte Carlo Tree Search (MCTS). MCTS performs a well-balanced exploration and exploitation based search over an unbounded space to discover better existing states. This requires that we initialize the number of iterations, a roll-out length and policy to balance exploration and exploitation. We represent this whole process in Algorithm 1. We take a model trained on other individuals for the target task domain as input (M_n). This model is represented with CE and becomes the root node. We initialize the number of iterations (n), the number of rollouts (c), and the length of rollouts (l). The number of iterations represent how many times CE-MCTS will be executed. The number of rollouts represent how many times the simulation will be carried out in each iteration. The length of rollouts depict the number of nodes to be explored in each rollout. We make use of ϵ-greedy for our node selection policy. We chose

ϵ=0.30 to bias our approach towards the exploration of the space. This value of epsilon is derived via experimentation using different possible values and picking the one which represents a balanced exploration-exploitation. In each roll-out, we select a random probability ($P_{current}$) which is compared with ϵ to decide if the already explored node with the greatest value will be chosen as the next node (exploitation) or a new node should be discovered (exploration). At the end of a rollout, we backpropagate through all the nodes in the rollout to update the node values of the parents based on the values of the child nodes as described below. This complete process is continued for n iterations.

3.5 Node Representation

MCTS is typically represented as a tree of nodes. In general, nodes in the MCTS algorithm represent different action states. In our CE based MCTS approach, each node represents a possible output model, its weights represented as a combination of weights from our input model. Each node is connected to one parent node and an unbounded set of child nodes. In CE-MCTS, each node stores the information for all of the weights of a potential neural network (as alpha filters and weights), a reference to it's parent node, references to it's child nodes and it's current fitness value. Node representation is depicted in Figure 3.2. We initialize the root node as the input model, represented in CE by assigning each weight a f value of itself and an α matrix of all 1's of the appropriate dimensions. This makes the root node equivalent to the input model at inference time, but allows us to add f and α values or modify existing values to produce new child nodes.

We employ a total of four functions to produce child nodes. We used these four functions as they have previously been employed for Conceptual Expansion [13]. These four function are responsible for altering the combined values of f and α to provide a variation of weights representing a new model:

- The first function multiplies a randomly selected index of a randomly selected α with a scalar value in the range [-2, 2].

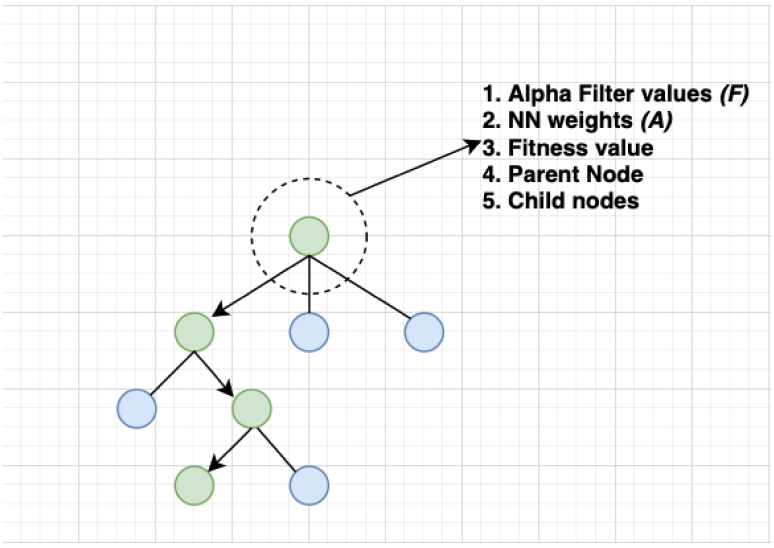

Figure 3.2: Node representation in CE-MCTS. Each node stores the information about the weights represented by that specific node in CE-MCTS, parent node, child nodes and the fitness value of the node.

- The second function multiplies an entire randomly selected α matrix by a scalar value in the range [-2, 2].
- The third function swaps two randomly chosen α and f values (with equivalent dimensions) .
- The fourth function adds two randomly chosen α and f values (with equivalent dimensions) to a CE approximation.

These 4 neighbour functions are visualized in Figure 3.3.

During CE-MCTS, all the nodes are assigned with a value to determine the utility of it and the likely utility of its children, or following this "path" through the search space. Equation 2 is how we approximate the value of each node during a CE-MCTS roll-out.

$$\frac{Predictions_{Correct}}{Predictions_{Total}} - (\frac{1}{n}) \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{3.2}$$

This equation combines a representation of accuracy over the secondary task data and the Mean Square Error (MSE) loss on the target task data. The first part represents the accuracy measured on the basis of correct predictions made ($Predictions_{Correct}$) out of all the predictions ($Predictions_{Total}$) made

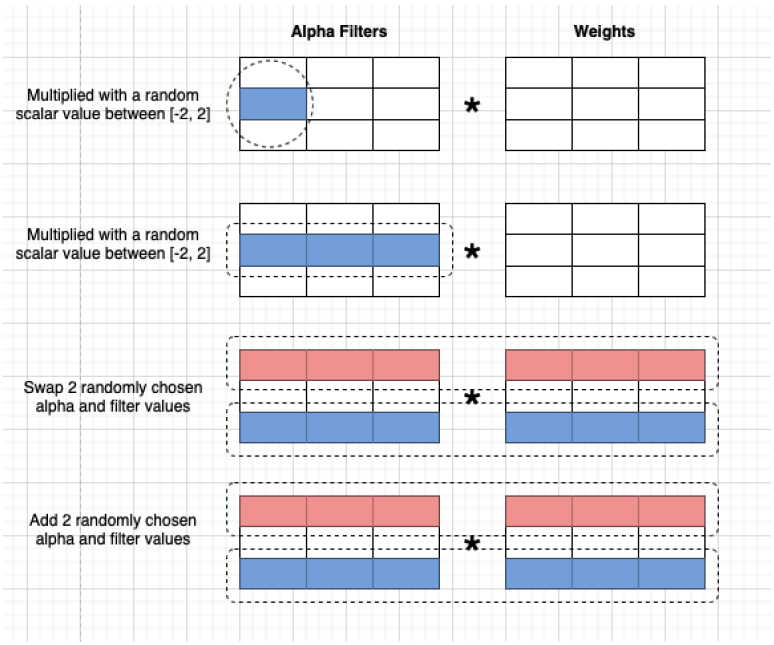

Figure 3.3: 4 CE neighbour functions.

on the secondary task for a specific individual. We transform our regression task into a classification task by mapping each value to (-1,0,1). We prefer this smooth guidance from the secondary task as it is not directly related to our target task. The second part represents the MSE loss on the training data calculated as the average of the sum of squared differences between the original values (y_i) and predicted values ($\hat{y_i}$). This makes the search space more granular, allowing MCTS to more easily find gradients to climb. After reaching the last node in the rollout length, all the nodes traversed during the rollout are updated with the same metric value combined with a discount factor ($\gamma = 0.5$ in this paper). The discount factor is responsible for updating the value of the parent nodes by adding the discounted value of the child node from the current rollout to the value of the parent node.

3.6 Final Model Selection

After finishing up the iterations of the MCTS search, our aim is to identify a single final output model. Typically in MCTS one would select the child of the initial root node with the maximum or minimum value. However, given

the way we bias CE-MCTS towards exploration we use a different criteria to select our final output model. For the financial time series prediction domain we identify the five most similar individuals in terms of their secondary task performance to our chosen individual. We then use the average performance (MSE) of each model on these individuals for the target task to select a final model. For the video game designer modeling domain we use the performance of each model (MSE) on the secondary task. The method of selection of a better performing model which best represents an individual's behaviour is a domain specific measure. Notably in both cases we still do not make use of, or have access to, any data on the specific individual for the target task.

Chapter 4

Experimental Setup

This chapter covers the experimental setup details for this thesis. Section 4.1 introduces the readers with an overview of the target problem and related domains for the evaluation of CE-MCTS. In section 4.2 and 4.3, we talk about the architecture selection and the details of the transfer learning method associated in this work. This is followed by the details of the experimental setup in sections 4.4 and 4.5 of the 2 domains evaluated in this work. Section 4.4 covers the Financial Time Series Prediction domain and section 4.5 explains the Video Game Designer Modelling domain.

4.1 Experiment Overview

In this work, we are interested in model specialization problems where we do not have any data available for a specific individual for a target task but we have data of other individuals for the same task. We also assume we have a secondary dataset which is not related to the target task, but in which we have data for the specific individual. For the evaluation of CE-MCTS, we have selected two completely different problems that require modeling the behaviour of individuals.

- The first problem requires modeling the financial behaviour of a set of individuals during two different time periods.
- The second problem requires modeling the evaluative behaviour of a set of individuals in two different game level design tasks.

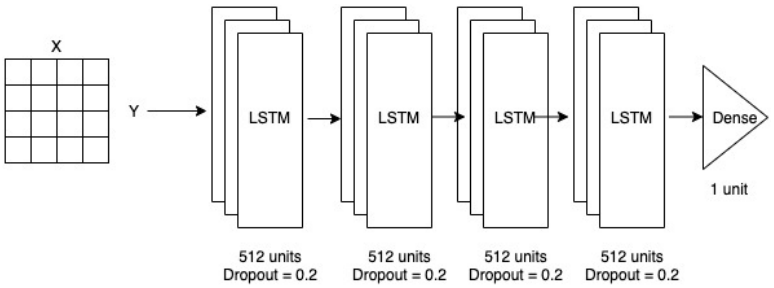

Figure 4.1: The recurrent neural network (RNN) architecture consisting of 4 LSTM layers. Each of the 4 LSTM layers contain 512 units with a dropout size of 0.2. This model contains a dense layer with the unit size of 1 at the end.

4.2 Architecture

In this work, as our experiments are focused towards modelling individual human behaviour over a period of time, our targeted problems deal with time series data. Therefore, we use Recurrent Neural Networks (RNNs) as our neural network architecture. For our approach and baselines, we use a simple recurrent neural network (RNN) architecture consisting of 4 LSTM layers. Each of the 4 LSTM layers contain 512 units with a dropout size of 0.2. These 4 LSTM layers are followed by a Dense prediction layer at the end with a unit size of 1. We employed Keras and used all the default values for its LSTM and Dense layers otherwise. Figure 4.1 represents the neural network architecture. Our focus is on how CE-MCTS is able to solve the model specialization problem in different domains compared to our baselines. Therefore, we are less interested in measuring the performance of more complex models that may obfuscate our primary goal. Moreover, we are looking at low data problems where more complex architectures may not be appropriate. We use the Adam optimizer along with a learning rate of 0.001, and mean square error (MSE) for calculating loss.

4.3 Transfer Learning

In our experiments, both of our domains involved low training data problems. Therefore, we pretrain our LSTM model on a stock market dataset. [1] This stock market dataset contains 20 years of daily information such as the opening value, highest value, lowest value and closing value of the stocks. We chose a stock market dataset to pretrain our model due to the large amount of available data and as it related to both of our evaluation domains. We used a batch size of 32 for 100 epochs, with a learning rate of 0.001. We used this learning rate across all approaches that employed backpropagation. This pretrained LSTM model is then used as the input to CE-MCTS and as the basis for all of our baselines but one.

4.4 Domain 1: Financial Time Series Prediction

For the first domain, we employ Servus Credit Union's records as our dataset. The data consisted of anonymous financial records of 320 individual across two distinct time periods of their lives. For the purpose of this paper, and due to an ongoing agreement, we will call these periods T1 and T2, as T2 took place after T1 in all cases. These time periods represent 2 different financial phases of each individual's life.

4.4.1 Dataset and Preprocessing

We were provided with 35 months of data belonging to time period T1 followed by 10 months of financial data for the same individuals in time period T2. For every day we have information on 15 transaction types including bill payments, debit transactions, credit transactions, and so on representing the financial behaviour of these individuals. The financial behaviour of the individuals differs significantly in T1 and T2, which we confirmed with the Mann Whitney U-test across the 15 transaction types ($p \ll 0.01$).

[1] `https://www.kaggle.com/rohanrao/nifty50-stock-market-data?select=HDFC.csv`

The input size for this domain is 31x15 representing the 15 different types of transactions across a month. The output from the neural network is a single value that varies from [-1,1] and represents the financial fitness of the individual during that month.

4.4.2 Experimental Details

The goal of CE-MCTS and the baselines is to approximate a model that accurately predicts this financial fitness during T2 for individuals we have not observed during T2, based on their behaviour during T1. We can expect model specialization to benefit this domain due to the high degree of variance in individual financial behaviour.

In this task, we evaluate our CE-MCTS approach over a 5-fold cross validation experiment, given that we cannot publicly release the dataset and it has no standard train-test splits. Each fold consists of 64 test individuals and 256 training individuals. For each test individual we assume we do not have access to their T2 data, and only make use of it during testing. For this first problem domain we used CE-MCTS with 5 iterations, each with 20 rollouts of length 10.

4.5 Domain 2: Video Game Designer Modelling

To analyze how CE-MCTS performs in modelling humans in general, we decided to choose a completely different task from domain 1 as our second domain for the experimental setup. In the second domain, we have a dataset of 84 individuals who worked with an AI partner on video game level design tasks [12]. Figure 4.2 represents the AI partner developed in this work named as 'Morai Maker'. A video game level is a standalone environment or piece of game structure. Each individual was given two tasks, to design two distinct types of levels for the game "Super Mario Bros.". In this domain, tasks T1 and T2 represent designing two different types of Super Mario Bros. game levels with an AI agent. The two level types are "underground" and "above

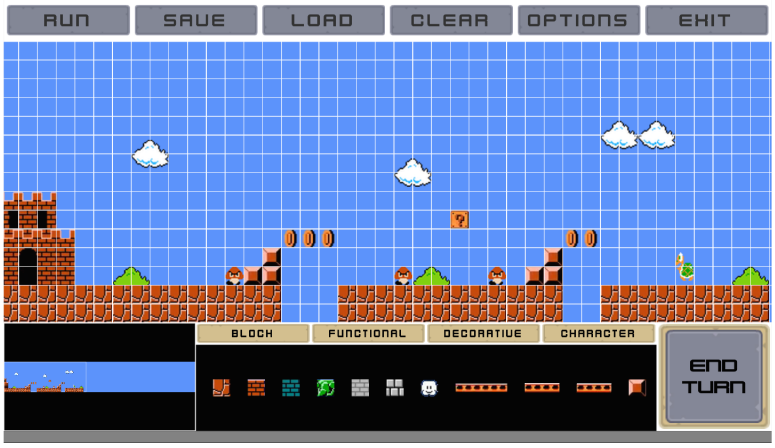

Figure 4.2: Morai Maker: AI based game level designer tool.

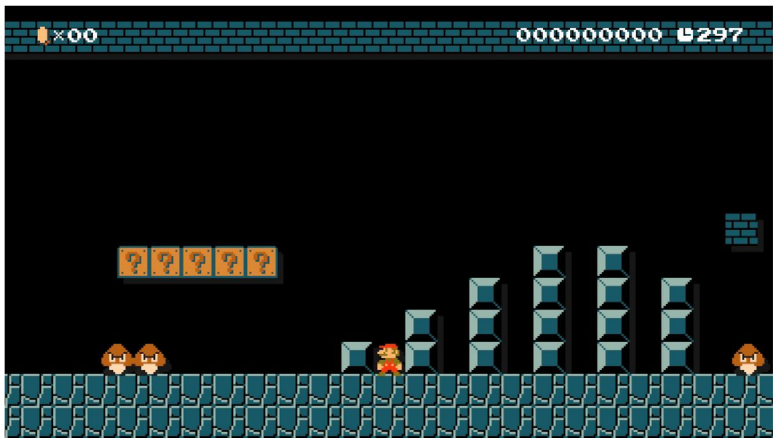

Figure 4.3: Super Mario Bros underground level example.

ground", which differ in terms of structure and in terms of what kinds of entities appear within them. Figure 4.3 and 4.4 represents samples of Super Mario Bros 'underground' and 'aboveground' game levels respectively. The human designer and the AI took turns making additions to a video game level until it was complete. While undertaking these tasks the designers gave implicit feedback to the agent by retaining or deleting its suggested additions. This was then transformed into explicit, numerical reward as described in the original paper [12].

The user base of this study was comprised of 84 participants out of which 62% users had prior experience in designing Mario levels at least once. Participants involved in this case study can be sub-categorized as 26% who had no

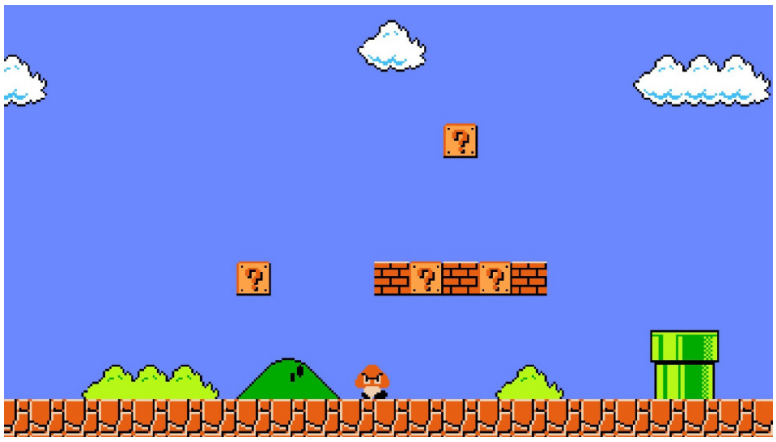

Figure 4.4: Super Mario Bros aboveground level example.

experience in designing a level, 36% who had designed a game level at least once, and 38% having previous experience in designing multiple game levels. In general, all the participants had played games and 77 users had previously played Super Mario Bros..

4.5.1 Dataset and Preprocessing

The goal for this dataset is to correctly predict how a particular designer would evaluate an action taken by the AI agent while designing an unseen type of level. The input is of size 40x15x34, which we reshape to 600x34, where 40 is the width of a chunk of level, 15 is the height of a chunk of level, and 34 corresponds to a one-hot encoding of game entities. Thus if a particular cell has a 1 in it that indicates that the AI just added that entity at that location. This corresponds to only part of the level (which has dimensions 100x15), but it is the part of the level where the AI and/or human designer have recently made changes, and was employed in the original work. The output is a single value that varies from [-1,1] and represents the designer's evaluation of that AI action. Because of a widely varying type of user composition involved in this study, the behaviour of participants while designing the game level can differ from each other a lot. Moreover, their thought process while designing the 'aboveground' and 'underground' Super Mario Bros level can also vary from each other. We can expect model specialization to be beneficial in this domain

as individuals tend to have highly varied design styles and preferences [11].

4.5.2 Experimental Details

We make use of the train-test split from prior work for comparison purposes [12], which used 73 train individuals and 11 test individuals. We also only make use of two of the baselines from our prior experiment: Random Search and Aggregate. We made this choice given that these two baselines performed the best in the first experiment. We also include the approach put forth in the original Guzdial and Riedl paper as a baseline [12], which modeled the problem with a convolutional neural network instead of an LSTM. We note that this approach is similar to the 2$^{\text{nd}}$ Task-trained baseline, as it included training on the first level design task data. For this domain, due to its smaller size, we used CE-MCTS with 3 iterations of 20 rollouts of length 10.

Chapter 5

Results

This chapter covers the results representing the performance of CE-MCTS in comparison with other baseline approaches over 2 different domains. Section 5.1 introduces the readers with the baseline approaches that are used in this work for the comparison of the performance of CE-MCTS approach. Section 5.2 presents the results and analysis of CE-MCTS in comparison with other baseline methods on the financial time series prediction domain. In section 5.3, we discuss the performance of CE-MCTS along with the other baselines for the video game designer modelling domain.

5.1 Baselines

In this work we employed six baselines. The first three represent how one might traditionally attempt to solve a model specialization problem, while the next three employ the CE representation, and are included to clarify the benefits of MCTS for this problem.

- The first baseline, called "Naive Aggregate" simply involves finetuning our source model on the target task data for all available individuals for 30 epochs. This represents the standard approach one might take to "solve" a model specialization problem: not trying to solve the problem at all and assuming that unseen individuals will fall into the learned distribution. This model is employed as the input for all other baselines and CE-MCTS.

- The second baseline, called "CE-2nd Task-finetuned" takes our Naive Aggregate model and then further trains it on the available secondary task data for our specific individual for 10 epochs. This represents the common approach of finetuning a model on a secondary task when it is similar initial to a target task. If it is the case that the secondary task is close enough to the target task, this should lead to improved performance.

- The third baseline, named "Random" represents a random exploration of the neighbouring models to the "Aggregate Naive" model. It is implemented identically to the below baseline, but only searches over the default DNN representation, not the CE representation. It is included to compare the greedy finetunng of the above baseline, and as a comparison point to the below baseline.

- Our fourth baseline, named "CE-Random" uses a random walk instead of MCTS to explore the space defined by CE. The same root node and child functions are employed as described above. At each step, a random child with random arguments is selected and repeat for 100 steps. Selection of the best model is made according to the final model selection criteria described above. This will allow us to test if our MCTS approach leads to better exploration than a pure random walk.

- The fifth baseline, called "CE-Greedy" uses a greedy or hill-climbing optimization instead of MCTS. This is roughly equivalent to the original conceptual expansion on DNN work [13]. At each step ten random neighbours using the above child functions are produced, each neighbour is evaluated and the one that maximizes the node value between all neighbours and the current node is chosen. This process is repeated for 100 steps for up to a total of 1000 node evaluations. The best final model is selected according to the same selection criteria. This baseline allows us to test if exploration is needed at all.

- The sixth and final baseline, called "CE-Beam Search" employs a simple

beam search. We send out 400 random beams of length 10 from the initial root node. This is similar to the random search baseline, but essentially checks the case of a search approach with more coverage but a limited depth. We again employ the same final model selection criteria over these 4000 nodes.

For a fair comparison, the training time for all these baselines and our approach CE-MCTS was kept the same. All of these baseline approaches and CE-MCTS are executed roughly for around 6 hours independently. Keeping the time constraint the same across all the approaches helps us in validating which approach is able to model individuals better.

5.2 Domain 1: Financial Time Series Prediction

5.2.1 Results

The Average Mean Square Error (MSE) loss and Standard Deviation (SD) across all baselines and our proposed CE-MCTS approach are provided in Table 5.1. These same values for the outlier individuals are presented in Table 5.2. CE-MCTS outperforms all other baselines for 4 of 5 folds across all test individuals, and for all folds for the outliers. The Naive Aggregate baseline performed fairly well, which follows from it being the standard way of handling problems like this, without engaging in specialization. Its performance demonstrates that the majority of held-out individuals did fall within the learned distribution in terms of their T2 behaviour. However, it does up to an order of magnitude worse when it comes to the outliers. Given that we are interested in modeling individuals, this performance is insufficient, demonstrating a need to further adapt this initial model. The 2^{nd} Task-finetuned baseline, where we finetuned the Naive Aggregate model on the available secondary task (T1) data for each individual (420 datapoints), performed significantly worse than all other approaches across all test individuals. This demonstrates that for problems like this traditional transfer methods lead to worse models due to the fact that the secondary task (T1) is so different from the target task (T2).

However, on the outliers 2nd Task-finetuned outperformed Naive Aggregate, but no other approaches. Thus, we conclude that outlier individuals exhibited some similar behaviour between T1 and T2, but insufficiently similar to make naive finetuning sufficient.

Both of the random approaches present intriguing results. Over all test individuals, CE-Random performed surprisingly well, even outperforming CE-MCTS on the third fold. However, pure Random baseline performed much worse. This provides an evidence that CE is more likely to represent good models in comparison to naively altering DNN weights. This is also a clear indicator of the importance of exploration in the CE representation, as CE-Random outperformed CE-Greedy in all cases. However, the format of the exploration matters, as CE-Beam Search performed as well or slightly worse than the Naive Aggregate baseline. This indicates that the models "nearby" to the CE-Aggregate approach in the CE representation were all fairly similar. This could have been addressed with longer beams, but we already identify the importance of exploration from CE-Random's performance. Despite this CE-MCTS outperforms CE-Random in almost all cases, indicating that exploration alone is insufficient, and that exploiting based on our value function is beneficial. On average, CE-MCTS outperformed the closest baseline by 0.003 MSE across all individuals and by 0.016 MSE on outliers. While this may seem like a small amount, its value ranges from 5.19 USD and 186,194.42 USD based on the individual, with a median value of 216.50 USD. Practically, these improvements are impactful in terms of the amount corresponding to this margin of enhancement in modelling individuals. Servus Credit Union, our partner for this task, found these results helpful in terms of how they can model individuals to reach better financial health.

5.2.2 Statistical Significance Results

We have established that CE-MCTS performs the best in this domain on average, and that these results were viewed as an improvement by domain experts. However, this does not tell us if the results are significantly different. To understand whether CE-MCTS is statistically different from other baseline

Approach	fold 1	fold 2	fold 3	fold 4	fold 5	Average
Naive Aggregate	0.089 ±0.104	0.110 ±0.134	0.081 ±0.096	0.128 ±0.150	0.089 ±0.118	0.089 ±0.111
2nd Task-finetuned	0.137 ±0.180	0.210 ±0.203	0.170 ±0.225	0.268 ±0.247	0.213 ±0.262	0.196 ±0.223
Random	0.088 ±0.104	0.110 ±0.134	0.082 ±0.096	0.132 ±0.152	0.089 ±0.118	0.100 ±0.121
CE-Random	0.089 ±0.103	0.105 ±0.130	**0.076 ±0.089**	0.128 ±0.150	0.088 ±0.119	0.088 ±0.111
CE-Greedy	0.092 ±0.099	0.146 ±0.170	0.149 ±0.225	0.136 ±0.156	0.167 ±0.251	0.129 ±0.175
CE-Beam Search	0.089 ±0.101	0.113 ±0.138	0.081 ±0.094	0.128 ±0.150	0.090 ±0.117	0.089 ±0.109
CE-MCTS	**0.085 ±0.101**	**0.103 ±0.130**	0.081 ±0.095	**0.120 ±0.143**	**0.085 ±0.112**	**0.085 ±0.107**

Table 5.1: Average Mean Square Error (MSE) loss over 5 cross-validation folds of the financial time series prediction dataset.

Approach	fold 1	fold 2	fold 3	fold 4	fold 5	Average
Naive Aggregate	0.231 ±0.259	1.087 ±0.294	1.084 ±0.288	1.115 ±0.354	1.149 ±0.385	0.933 ±0.316
2nd Task-finetuned	0.324 ±0.307	0.476 ±0.244	0.292 ±0.223	0.328 ±0.233	0.439 ±0.213	0.372 ±0.224
Random	0.231 ±0.260	0.170 ±0.188	0.172 ±0.234	0.175 ±0.190	0.203 ±0.194	0.190 ±0.213
CE-Random	0.229 ±0.261	0.160 ±0.179	0.162 ±0.230	0.173 ±0.189	0.198 ±0.194	0.184 ±0.211
CE-Greedy	0.243 ±0.257	0.260 ±0.343	0.223 ±0.327	0.172 ±0.187	0.407 ±0.318	0.261 ±0.286
CE-Beam Search	0.231 ±0.260	0.181 ±0.196	0.167 ±0.231	0.177 ±0.191	0.195 ±0.196	0.190 ±0.215
CE-MCTS	**0.209 ±0.241**	**0.127 ±0.136**	**0.157 ±0.224**	**0.161 ±0.174**	**0.186 ±0.208**	**0.168 ±0.196**

Table 5.2: Average Mean Square Error (MSE) loss over 5 cross-validation folds over the outliers of the financial time series prediction dataset.

approaches or not, we conducted paired t-tests between the predictions made by our baselines in comparison with CE-MCTS and between the Mean Square Error (MSE) observed between CE-MCTS and the other baseline approaches.

Table 5.3 represents the p-values observed for the paired t-test conducted for the predictions between CE-MCTS and other baseline approaches across the 5 cross validation folds and the outliers. Table 5.4 represents the p-values for the paired t-test comparing the MSE between CE-MCTS and our baseline approaches across the 5 cross validation folds and the outliers.

These results indicate that CE-MCTS leads to models with significantly different performance compared to everything but CE-Beam Search. Moreover, CE-MCTS leads to significantly better results than the naive baseline approach, the 2nd task fine-tuned approach, and CE-Greedy on the outliers. This strengthens the value of exploration over the outliers and demonstrates the difference encountered with a balanced exploration-exploitation search method.

CE-MCTS VS	fold 1	fold 2	fold 3	fold 4	fold 5	Outliers
Naive Aggregate	0.850	**7.475e-26**	**1.874e-23**	**2.536e-29**	**5.695e-42**	**0.008**
2nd Task-finetuned	0.429	0.453	0.344	0.913	0.080	**0.008**
Random	0.872	0.259	0.851	0.248	0.182	**0.0365**
CE-Random	0.174	0.765	0.250	0.207	0.069	**0.004**
CE-Greedy	0.162	**6.724e-05**	**1.256e-10**	**0.023**	**0.002**	0.454
CE-Beam Search	0.250	0.054	0.512	0.124	0.418	0.481

Table 5.3: Paired T-test p-values comparing the predictions observed across the 5 cross validation folds and the outliers between CE-MCTS and the different baseline approaches.

CE-MCTS VS	fold 1	fold 2	fold 3	fold 4	fold 5	Outliers
Naive Aggregate	0.871	**3.502e-06**	**3.278e-05**	**8.917e-06**	**0.0003**	**1.084e-18**
2nd Task-finetuned	0.145	**0.002**	**0.024**	0.097	0.317	**7.468e-07**
Random	0.871	0.791	0.943	0.651	0.852	0.224
CE-Random	0.859	0.934	0.786	0.745	0.868	0.221
CE-Greedy	0.719	0.116	**0.031**	0.537	**0.020**	**8.728e-07**
CE-Beam Search	0.840	0.682	0.988	0.742	0.810	0.553

Table 5.4: Paired T-test p-values compare the MSE observed across the 5 cross validation folds and the outliers between CE-MCTS and the different baseline approaches.

5.3 Domain 2: Video Game Designer Modelling

5.3.1 Results

The results for this experiment are presented in Table 5.5. CE-MCTS outperforms all three of the baselines, including the original approach used for this dataset [12]. Notably both of the other baselines outperformed this original approach as well, which we identify as likely due to pretraining the LSTM network on the Stock Market dataset prior to training the Naive Aggregate model, which is otherwise the closest new approach in performance. To put this in context, CE-MCTS' performance demonstrated a third of the error of the original approach, which has already shown an impressive ability to adapt to individuals [11]. Same trend can be observed from the first domain of CE-Random performing well, but still outperformed by 0.013 MSE. We do not recommend directly comparing the results from this domain to the first domain, given they are modeling two very different kinds of behaviour. However, the similar improvement over outlier or outlier-rich datasets of individuals suggests that this approach should extend to similar problems in other domains.

Approach	MSE ±SD
CE-Random Search	0.197 ±0.131
Aggregate	0.500 ±0.246
Guzdial & Riedl [12]	0.596 ±0.483
CE-MCTS	**0.184 ±0.130**

Table 5.5: Average MSE loss and Standard Deviation (SD) for the video game design modeling dataset.

Chapter 6

Conclusion

This chapter sums up the conclusive thoughts on the work presented in this thesis. In section 6.1, we present the implications of our proposed CE-MCTS approach. Section 6.2 provides the readers with further discussion on possible future work using CE-MCTS. Finally, section 6.3 presents the readers with the closing thoughts for this work.

6.1 Implications

In this work, our focus was on a problem we called 'model specialization'. This thesis presents a novel approach called CE-MCTS to model an individual on an unseen target task based on the behaviour of others on that target task, and data of that individual on a secondary task. We observed that CE-MCTS can be utilized to identify relationships between a secondary task and target task data, which is difficult to achieve with the current existing transfer learning and search-based approaches. Morevoer, CE-MCTS also proves beneficial in discovering how a secondary task can be indirectly connected to the target task and can be used to better model individuals. Our approach outperforms relevant transfer learning baselines for this problem in two domains, which indicates support for the appropriateness of this approach to this type of problem.

6.2 Future Work

While the results are positive, this does not yet represent strong evidence that CE-MCTS is appropriate in any domain where this problem formulation exists and might prove useful. In future work, we are interested in exploring how CE-MCTS will perform in modelling individual human behaviour in different domains. In particular, we are interested in modeling individuals in the medical domain. Our interest is more aligned towards the medical domain because CE-MCTS can prove beneficial in tackling privacy, security and specialized care concerns in healthcare. We presented an initial implementation of CE-MCTS that focused on a relatively simple architecture and relatively low amounts of computation time. Computation time was kept fixed across our baselines (except the prior Guzdial and Riedl approach in the second domain [12]) for comparison purposes. It is unclear how CE-MCTS will perform with other hyper-parameter choices. We expect CE-MCTS to discover better Neural Networks to model individuals if trained for a longer period of time due to the balanced exploration-exploitation based search in the state space. We hope to explore this in future work and evaluate the impact of hyperparameter selection.

6.3 Closing Thoughts

In this thesis, we introduced the model specialization problem, and Conceptual Expansion-based Monte Carlo Tree Search (CE-MCTS) as a potential solution. This approach uses CE to define the search space and MCTS to search over this space. We evaluated the performance of CE-MCTS in comparison to existing transfer learning approaches over two different domains: financial time series prediction and video game designer modelling. The results indicate that CE-MCTS is a good solution for model specialization and outperforms existing transfer learning approaches. These positive results lay the foundation for exploring how CE-MCTS will perform in modelling individual human behaviour across different domains. In the future, we look forward

to evaluating how our proposed approach performs on more general problems.

References

[1] A. Almeida and G. Azkune, "Predicting human behaviour with recurrent neural networks," *Applied Sciences*, vol. 8, no. 2, p. 305, 2018.

[2] K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. Tran, and B. Seaman, "Sales demand forecast in e-commerce using a long short-term memory neural network methodology," in *International Conference on Neural Information Processing*, Springer, 2019, pp. 462–474.

[3] M. A. Boden, "Creativity and artificial intelligence," *Artificial Intelligence*, vol. 103, no. 1, pp. 347–356, 1998, Artificial Intelligence 40 years later, ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00055-1. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370298000551.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečn, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[5] Y. Cui, S. Wang, and J. Li, "Lstm neural reordering feature for statistical machine translation," *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016. DOI: 10.18653/v1/n16-1112. [Online]. Available: http://dx.doi.org/10.18653/v1/N16-1112.

[6] W. Dai, Y. Chen, G.-r. Xue, Q. Yang, and Y. Yu, "Translated learning: Transfer learning across different feature spaces," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21, Curran Associates, Inc., 2009, pp. 353–360. [Online]. Available: https://proceedings.neurips.cc/paper/2008/file/0060ef47b12160b9198302ebdb144dcf-Paper.pdf.

[7] G. Fauconnier, "Conceptual blending and analogy," *The analogical mind: Perspectives from cognitive science*, pp. 255–286, 2001.

[8] G. Fauconnier and M. Turner, *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, 2008.

[9] C. Gagné and E. Shoben, "Influence of thematic relations on the comprehension of modifier–noun combinations," *Journal of Experimental Psychology: Learning, Memory and Cognition*, vol. 23, pp. 71–87, 1997.

[10] E. Gavves, T. Mensink, T. Tommasi, C. G. M. Snoek, and T. Tuytelaars, "Active transfer learning with zero-shot priors: Reusing past datasets for future tasks," *CoRR*, vol. abs/1510.01544, 2015. arXiv: 1510.01544. [Online]. Available: http://arxiv.org/abs/1510.01544.

[11] M. Guzdial, N. Liao, J. Chen, S.-Y. Chen, S. Shah, V. Shah, J. Reno, G. Smith, and M. 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*, ser. CHI '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13, ISBN: 9781450359702. DOI: 10.1145/3290605.3300854. [Online]. Available: https://doi.org/10.1145/3290605.3300854.

[12] M. Guzdial, N. Liao, and M. Riedl, "Co-creative level design via machine learning," *arXiv preprint arXiv:1809.09420*, 2018.

[13] M. Guzdial and M. O. Riedl, "Combinets: Creativity via recombination of neural networks," *arXiv preprint arXiv:1802.03605*, 2018.

[14] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[15] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1419–1426. DOI: 10.1109/IJCNN.2017.7966019.

[16] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[17] R. Singhania and S. Kundu, "Forecasting the united states unemployment rate by using recurrent neural networks with google trends data," *International Journal of Trade, Economics and Finance*, vol. 11, no. 6, 2020.

[18] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[19] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a cnn-lstm network," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 639–645. DOI: 10.1109/ICDAR.2017.110.

[20] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4582–4591.

[21] Y. Zhang and X. Lu, "A speech recognition acoustic model based on lstm -ctc," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, 2018, pp. 1052–1055. DOI: 10.1109/ICCT.2018.8599961.