# Parameter Search Transfer Learning

by

Mohan Sai Singamsetti

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

Master of Science

Department of Computing Science

University of Alberta

# Abstract

Deep learning approaches have had success in many domains recently, particularly in domains with large amounts of training data. However, there are domains without a sufficient quantity of training data, or where the training data present is of insufficient quality. Transfer learning approaches can help in such low-data problems, but still tends to assume access to sufficient source domain data and a sufficient signal for transfer. In this work, we propose a novel approach for transfer learning called Parameter Search Transfer Learning (PSTL) which directly searches over parameters of a neural network in order to minimize the impact of low training samples in both source and target domains. Across Reinforcement Learning (RL), Regression, and Classification tasks we demonstrate that PSTL meets or exceeds the performance of transfer learning baselines, which we hypothesize is due to its ability to identify a better gradient.

# Preface

This thesis presents original work by Mohan Sai Singamsetti under the supervision of Dr. Matthew Guzdial. The work is currently under review at NeurIPS 2023. Some of the experiments conducted in this work are related to Health Domain tasks and have conducted in collaboration with Dr. Jane Cook (Clinical Lecturer, University of Calgary), and Dr. David Olson (Professor, Univeristy of Alberta). A few of the techniques in this thesis were from our previous work CENAS [29].

*Dream is not that you see in sleep, the dream is something that does not let you sleep.*

– A. P. J. Abdul Kalam, Former President of India, 1943.

# Acknowledgements

Firstly, I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Matthew Guzdial, for his invaluable guidance and support throughout my research journey. His expertise, encouragement, and dedication have been critical in shaping this work and pushing me to reach new heights.

I would like to extend my gratitude to Dr. Matthew Guzdial for providing me with a remarkable opportunity to undertake an internship during my undergraduate studies. I am immensely thankful for the invaluable experience and exposure to various companies that I gained through my research collaboration with Dr. Matthew Guzdial. I am truly grateful for his unwavering commitment to my success and for the opportunities he has provided me. Thank you, Dr. Guzdial, for your mentorship and belief in my abilities.

I extend my thanks to my committee members, Dr. Matt Taylor and Peter Chun, for their insightful questions and valuable feedback during my Master's examination. Additionally, I would like to thank my committee chair, Dr. Mohammad Salvatipour, and the entire Computing Science Department for supporting me throughout the completion of my program.

Special thanks go to Sudhakar Singamsetti, Sujatha Singamsetti, and Rupesh Kumar Singamsetti for their consistent support throughout my academic journey. Their unwavering belief in me has been a constant source of motivation.

Finally, I am grateful to my friends Samridhi Vaid, Subhojeet Pramanik, Akash Sasikumar, Kushankur, and Kushagra Chandak for their support throughout my master's program at the University of Alberta. Their friendship and encouragement have made this journey even more fulfilling.

To all those mentioned above, I am sincerely grateful for your support, guidance, and belief in me. Thank you all for being an integral part of my academic and personal growth.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Deep neural networks (DNNs) have transformed machine learning, achieving state-of-art performance in a wide variety of tasks including computer vision, speech recognition and natural language processing [32]. Modern deep neural networks tend to perform well when they have large structures (i.e., millions of parameters) and are trained on large amounts of data [4], [12]. However, this results in two key challenges. First, DNNs require a significant amount of computation to learn the representation between the input and output features through backpropagation. Second, there are application domains where we do not have enough training data, and there is some speculation that such domains may become more common in the future [35]. As a result, we cannot presently utilize these approaches in all domains.

Transfer learning refers to the process of training on a source task and transferring the knowledge learned from the source task to a target task. Transfer learning [15] can reduce the size requirement for training data by utilizing pre-trained models trained on existing, sufficiently large datasets for similar tasks. Many approaches exist including zero-shot learning [39], few-shot learning [8] and domain adaptation to transfer knowledge from a source domain to a target domain. Few-shot learning refers to the process of training ANNs on few samples of training data. Zero-shot learning is a process that allows machine learning models to generalize to new unseen classes or task without direct training examples. However, they tend to require either the same set of classes or features (zero-shot and few-shot) or some other parallelism between the domains (domain adaptation), learning additional feature representations or manually-designed features for adapting to the new domain. Fine-tuning is a common transfer learning approach. With a fine-tuning approach, we adapt a model trained on source domain dataset to a target domain by training the model via backpropagation on a target domain dataset. Fine-tuning can lead to other issues like overfitting and catastrophic forgetting where the model loses its generalization ability on the target task or abruptly forgets the knowledge learned from the source domain, particularly when dealing with limited training data. It is therefore often ill-suited to domains with very limited training data.

Search algorithms are a set of techniques that attempt to find an optimal solution to a problem by exploring the space of possible solutions in a structured and efficient way. Many applications of search to DNNs like Neural Architecture Search (NAS) [17] and Hyperparameter Optimization (HPO) [1] have shown significant success. However, these approaches tend to require a large amount of training data and well-designed fitness functions. NAS searches for network architectures within the space of possible architectures. HPO searches over hyperparameters of networks. We take inspiration from an approach for low data transfer first introduced by Singamsetti et.al [29]. In this thesis, we propose a similar approach that leverages the search for low-data problems but search over the parameters of fixed architectures.

In this thesis, we introduce Parameter Search Transfer Learning (PSTL), a stochastic greedy optimization over a neural network's parameters for transfer learning tasks. More complex optimization approaches like gradient descent backpropagation have shown great success in parameter optimization but require significant computation and training data. Our results show that PSTL, an approach that relies on a simple parameter search optimization, can outperform more complex approaches across a variety of cases.

## 1.1 Research Questions and Related Contributions

Considering the above motivation for the challenges with low-sample datasets and common issues with transfer learning across domains, we would like to answer the following research questions in this thesis.

- How can Artificial Neural Networks (ANNs) provide advantages in tasks with low-data compared to other conventional machine learning approaches?
- Can PSTL achieve optimal parameters with sufficient training data on both the source and target domain?
- Can PSTL achieve optimal parameters with limited training on the source and target domain?
- How well can PSTL perform compared to naive transfer learning approaches across different domains?

The contributions of this thesis are summarized as follows:

- Proposed an efficient stochastic greedy optimization method that searches over the parameter values of an architecture Parameter Search Transfer Learning (PSTL).
- Demonstrated how PSTL helps to achieve better performance at the early stages of training compared to complex optimization approaches like backpropagation (gradient descent) on target domains with sufficient data.
- Provided evidence that, on domains with insufficient data, using search-based approaches can help us to find more optimal models.
- Evaluated PSTL on three different domains: Chip-Placement Problem (Reinforcement Learning), Health outcomes prediction (Regression) and Image Classification benchmarks (Classification).

## 1.2 Outline

This thesis is organized into six main parts. It begins with an introduction that addresses the research questions related to Artificial Neural Networks (ANNs) in transfer learning problems. The subsequent section, Chapter 2, provides an overview of the background concepts and related work relevant to the thesis. In Chapter 3, we introduce our method Parameter Search Transfer Learning (PSTL). This chapter explores the various problem settings where PSTL is applicable and presents a comprehensive system overview, detailing the three-step process involved: Source training, Parameter Searching, and Postprocessing. Chapter 4 delves into the experimental setup and familiarizes readers with the domains we used for evaluating PSTL. These domains are chip-placement problems in an RL-based environment, a regression-based task focused on DNA-Health Outcomes prediction with limited data, and Image Classification tasks. Moving on to Chapter 5, the thesis describes the appropriate baselines for the problem and conducts a performance analysis, comparing the effectiveness of PSTL against other baseline approaches. We run experiments considering two cases, one with limited data in both the source and target domains, and one with sufficient data in both domains. Finally, Chapter 6 serves as the conclusion, providing an overview of how the proposed approach benefits tasks with low data and discussing potential implications for future work.

# Chapter 2

# Background

This chapter provides readers with the background knowledge required to understand the work presented in this thesis. In section 2.1 and 2.2 we introduce the Artificial Neural Network and Long Short-Term Memory (LSTM) to familiarize readers with the neural network approaches used in this thesis. In section 2.3 we give an outline of Transfer Learning, our proposed approach PSTL is a transfer learning approach, and so we discuss current approaches and challenges in transfer learning. In section 2.4 we introduce parameter searching and overview similar problem domains that deal with search-based problems. In section 2.5 we introduce readers to neuro-evolutionary optimization due to it and PSTL relying on search to optimize neural network parameters. In sections 2.6, 2.7 and 2.8 we provide readers with the necessary background information and related work in our application domains (Chip-Placement, DNA to Health Outcomes and Image Classification) that we use for evaluation.

## 2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the human brain. ANNs are the most dominant component of modern machine learning and have shown huge success in numerous domains like Computer Vision (CV), Natural Language Processing (NLP), speech recognition, anomaly detection recommendation systems, and more [23]. ANNs are known as function approximators, where the goal of the neural networks is to build a function that can map between input and output data. ANNs are most commonly op-

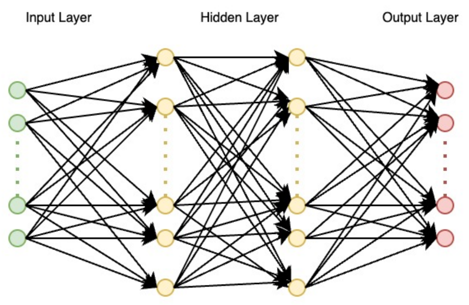

Figure 2.1: Artificial Neural Network Structure

timized via minimizing loss [11], where the loss is the difference between the output prediction and the ground truth labels.

ANNs are composed of interconnected nodes called neurons, which are used to process and analyze complex data. The structure of ANNs can generally be divided into three parts: input layer, hidden layer(s) and output layer. This standard structure of ANNs is shown in 2.1. The input data to ANNs is pre-processed and then passed to the input layer of the ANN. The dimensions of the input layer and the input data should be the same. This ensures that there is no mismatch in dimensionality between the input data and the input layer of the ANN. Dimensionality varies from problem-to-problem.

ANNs receive input data from the input layers, then process it in the hidden layers and finally output the values in the output layers, where the number of neurons in the output layers depends on the tasks (e.g., for a classification task, the number of classes equals the number of neurons in the output layer). ANNs employ hidden layers to enhance the learning capacity and enable the modelling of more complex relationships within the data.

Each neuron in a layer is connected to every other neuron in the previous layer with weights and bias forming a network of connections. Weights represent the strength of connections between neurons, while bias is an additional parameter added to each neuron, adjusting the overall output independently of the inputs. Each neuron employs an activation function to modify the input into the output to introduce non-linearity. This helps to learn non-linear representations between the input and the output, and to learn more complex relationships present in the data. The choice of activation functions in neural networks depends on the specific task. For example, sigmoid activation is common when the output needs to be bounded between 0 and 1, as in binary classification problems. Another common activation function is Rectified Linear Unit (ReLU), which is suitable for most cases and commonly used in hidden layers of deep networks, which promotes sparsity and helps in avoiding the vanishing gradient problem in ANNs [18]. We discuss this problem in section 2.2

Training ANNs involves optimizing parameters (the process of iteratively adjusting the weights and biases of the network) to minimize the difference between the predicted output and the desired outputs. For data preparation, we split the dataset into training, testing and validation sets. The validation set is used to evaluate the performance of the ANNs during training and guide the model's hyperparameter tuning. Pre-processing of the data also includes techniques like normalization, scaling and handling missing values. The network is either randomly initialized for general tasks or we use pre-trained weights in transfer learning tasks. The traditional model training process involves two steps:

**Forward Process:** passes the training data through the network in the forward direction, where each layer applies non-linear activations to compute the output prediction. A loss function then measures the difference between the output prediction and the actual labels. The most commonly used loss functions Mean Squared Error (MSE) for regression tasks are and cross-entropy for classification tasks.

**Backward Process:** refers to the process of computing gradients through a backpropagation algorithm. Backpropagation computes the gradients of the loss with respect to the weights and biases, starting from the output layer and moving backward through the network. Then the weights are updated through an optimization algorithm like gradient descent, to minimize the loss function and guide the network towards the optimal parameters.

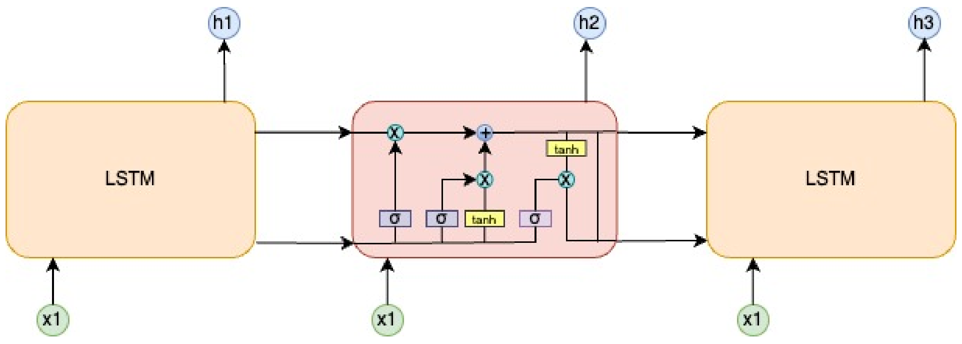

Figure 2.2: Long short term memory structure.

## 2.2 Long Short-Term Memory

Recurrent Neural Networks are a type of ANN designed to deal with sequential or time-series data. Sequential data is a type of data that has a significant order or sequence of elements, where the order of the elements carries important information and influences the behaviour of the data. RNNs could be applied in any task that deals with the analysis and prediction of sequences like NLP, series analysis, speech recognition, and machine translation [30]. ANNs handle the input data in a single pass, whereas RNNs have an extra module with internal memory that allows them to retain information from the previous input. The memory component of the RNN enables them to capture temporal dependencies present in sequential data.

One of the major issues with general RNNs is they struggle to handle longer dependencies, making it difficult for them to capture information that is far away from the current time step. This common issue in RNNs is known as the "Vanishing Gradient" problem. The weights in the RNN are updated using gradients, which are computed using the chain rule of calculus during backpropagation. The gradients are multiplied at every timestep, representing the cumulative effect of network parameters on the overall loss function. However, when the gradients are multiplied over multiple timesteps this makes the gradients smaller and thus they have almost no impact on the weights at the initial time steps. To tackle this problem more complex approaches such as Long Short-Term Memory (LSTM) RNNs have been introduced and used for predicting time series data more effectively.

LSTMs contain additional gating mechanisms which control the flow of information, which helps to selectively remember or forget information from previous time steps. LSTMs have different gating mechanisms consisting of input, forget and output gates as shown in Figure 2.2. These gates help to determine which part of the input and the previous memory state should remain in the memory gate. The forget gate helps to determine which part of the information in the memory should be discarded. This uses a sigmoid activation which outputs values between 0 and 1, representing the degree to forget. The output gate determines what information will be output as the hidden state of the LSTM for the current timestep.

The gating mechanisms help the LSTMs capture longer dependencies and which makes them capable of processing long sequences, carrying forwarding knowledge to the end of longer sequences. LSTMs have shown great success in a wide range of domains such as stock market prediction [22], handwriting recognition [37], machine translation [6] and many other tasks.

## 2.3 Transfer Learning

Transfer Learning with DNNs involves the transfer of knowledge and parameters from a DNN trained on a source problem dataset to a DNN for a similar target problem. This approach has achieved significant attention and success in tasks with limited data. A wide range of techniques exist such as zero-shot [39], one-shot [8] and few-shot [26] learning to transfer knowledge from a source domain to a target domain with limited samples of data. But these kinds of approaches often rely on manual-authored features or additional data to effectively guide the knowledge transfer process. In this thesis, our proposed approach does not require any additional manually authored features or training to adapt to the target domain. We also focus on domains that have limited data in the source and the target domain. There have been many previous approaches [36] that deal with problems where the target domain data is limited but the source domain data is abundant. To the best of our knowledge, there are currently no approaches that utilize DNNs in this transfer learning setting to address the problem of both limited source and target domain data.

## 2.4 Parameter Searching

Parameter searching in Deep Neural Networks is the process of finding the optimal values for the parameters in a neural network. The search space of the parameters includes the weights and biases associated with the connections between the neurons. One of the most common search problems in neural networks is hyperparameter optimization (HPO) [9], where is the goal is to find the optimal hyperparameter of an architecture, however in this thesis we focus on optimizing the parameters of the architecture, which has a much larger search space compared to the HPO.

Different optimization algorithms have been used to find the optimal parameters of neural networks. These include random search [3], which involves randomly sampling parameter combinations from a predefined range of parameter distributions, grid search [24], which exhaustively evaluates the predefined set of parameter combinations, and Bayesian optimization [1], which selects the parameters based on the relationship between the parameter values and the objective function using surrogate models, and the well-known parameter optimization algorithm gradient descent [27], which computes the updates for the parameters of the network in an iterative fashion by computing gradients of the parameters. Backpropagation-based approaches have shown huge success in various domains in finding optimal parameters, but these approaches demand heavy computational costs and require large amounts of data to compute optimal parameter values. Our proposed approach PSTL finds better parameter values in the early stages of training compared to other popular optimization approaches like gradient descent. In cases with insufficient training data samples, PSTL outperforms naive gradient descent-based fine-tuning approaches.

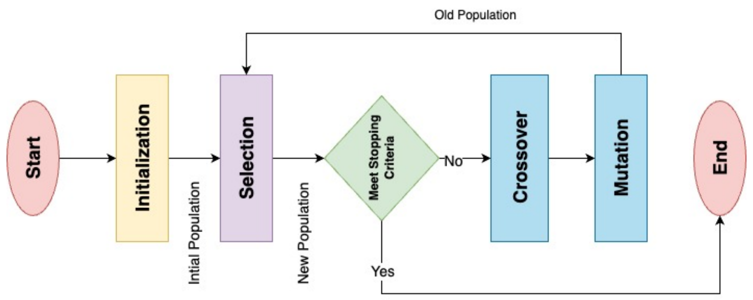

Figure 2.3: Neuro-evolutionary Workflow

## 2.5 Neuro-evolutionary Optimization

Neuro-evolutionary optimization is an evolution-inspired algorithm, which combines the concepts of ANNs and ideas from biological evolution. Neuro-evolutionary is a type of search algorithm often used to optimize the structure of the network [17], along with its parameters [7]. Genetic algorithms are a type of Neuro-evolutionary algorithm. There are six main components of the genetic algorithms w.r.t. searching neural networks and the workflow is as shown in Figure 2.3.

**Population:** The population consists of the set of possible architecture or network structures. Each individual in the population represents a network with different structures or parameters.

**Fitness Evaluation:** The fitness function is used to evaluate the performance of an individual in the population. which helps to rank the population from the best to the worst. The choice of the fitness function differs from task to task. Different metrics such as accuracy, loss, error, and other performance metrics could be used to compute the fitness score based on the task.

**Selection:** Selection of individuals (i.e., NN models) from the population is the process of choosing parents for the next population. Selection of the parent models is based on various strategies such as rank-based selection, tournament selection and roulette wheel selection[2].

**Mutation & Crossover:** This involves creating new child models from selected parent models. Different operations such as mutation and crossover are applied over the parent model to generate the offspring models. Mutation refers to the introduction of random changes or perturbations to architecture, while crossover refers to the process of combining the characteristics of two parent models to generate an "offspring".

**Termination Criteria:** The termination criteria defines when to stop the algorithm. There are different approaches to terminate the evolutionary process, the most commonly used methods are setting the maximum number of generations, and maximum threshold value. But one could even generate till the computation is exhausted.

These components iterate over multiple generations until the terminal condition is reached. Once the generation is terminated, the best-performing individual in the final population is selected as the final result for the specific task. Our PSTL approach is inspired by neuro-evolutionary optimization, with slight modifications to the off-spring generation and selection mechanism. In terms of off-spring generation, we employ mutation operations to generate the child models, and for the selection mechanism, we utilize a simple greedy method to choose the best parent for the next generation.

## 2.6 Chip-Placement Problem

Chip-Placement refers to the problem of designing a hardware chip via determining the placement of a number of macro components. This is a graph optimization problem, where the nodes in the graph are the macro-units and edges are the wire connections. Placement of the macro components and standard cells on the chip design is a complex task, which involves multi-objective optimization, optimizing variables like wirelength, congestion and density of the circuit. Where the macro components refer to the memory units and standard cells components like logic gates. A wide range of approaches have been proposed to solve this problem from using naive divide-and-conquer approaches [10], [33] to deep reinforcement learning [20]. One of the early approaches, that has shown long-term success in the Chip-Placement Problem is Simulated Annealing (SA) [28].

Circuit Training (CT) [20] is a graph optimization approach for the chip-placement problem, where an agent learns to make chip-placement decisions by interacting with the environment and receiving rewards (i.e. minimizing wirelength, density and congestion). Their approach is based on Reinforcement Learning (RL), a branch of machine learning in which agents interact with environments and learn to behave optimally by receiving rewards [31]. This is in contrast to the supervised learning branch of machine learning in which we train a model on existing input and output data pairs, which is the branch of the other two evaluation domains in this thesis. The original Circuit Training paper showed that transfer learning applied to placement problems could help to achieve lower costs at early stages. A recent paper [5] compared the performance of CT with standard baselines like SA, finding SA outperforms CT. We ignore the performance of SA as a baseline in this thesis because we wanted to see the performance gain of our PSTL approach w.r.t naive transfer learning approaches. In this thesis, we present a transfer learning approach that searches and finds the placements of the chip designs at earlier stages compared to fine-tuning. In this thesis, we include CT as a baseline for chip-placement and meet or exceed its performance across various transfer tasks.

## 2.7 Medical Domain Problem

DNA methylation(DNAm) is the process where the small chemical tag i.e. methyl group gets added to a DNA sequence. This process controls how the gene expression either by turning them on or off at the different point of the DNA sequence. Mapping the relationship between humans DNAm sequences to health outcomes can help in tasks like predicting fetal health outcomes from DNA of Maternal smoking during pregnancy (MSP). MSP is linked to many non-fatal, complex diseases, including depression and anxiety. Some instances of prior work have attempted to link DNA data to health outcomes [21], [34], [38]. However, these approaches have most commonly attempted this with statistical analysis or linear or logistic regression [21] to find the mapping between these features. It may be possible to outperform this mapping of the

DNAm sequence and health outcomes with DNNs. In this thesis, we use a Deep Neural Network model for the task and PSTL to find a better representation in a transfer learning setting.

## 2.8 Image Classification

Many papers have used image classification benchmarks to evaluate transfer learning approaches [17], [29]. Most similar to our own work, Singamsetti et al. [29] have used a low-sample image classification target dataset for transfer learning and simultaneous Neural Architecture Search (NAS), for finding optimal architecture and parameters for a model. In this thesis, we evaluate a similar setting for finding the optimal parameters of the network but with limited data in both source and target domain. In addition, we do not attempt NAS.

# Chapter 3

# Parameter Search Transfer Learning(PSTL)

In this chapter, we introduce the readers to an in-depth explanation of Parameter Search Transfer Learning (PSTL), In Section 3.1 we introduce readers to the types of problems our approach is applicable and well-suited to, and detail the domains that we chose to evaluate PSTL. In Section 3.2 we give a general overview of the PSTL approach. We overview the method across three sections. In section 3.3, we describe the first step of the PSTL approach, which describes the source model training. In section 3.4, we describe the searching mechanisms and subsection 3.4.1 describes the mutation operations that were used to explore the search space. Subsection 3.4.2, describes the fitness functions that were used based on different domains. In section 3.5 we describe an additional post-processing step required for further tuning the model under certain conditions.

## 3.1 Problem Definition

In this section we define the type of problems our approach, Parameter Search Transfer Learning (PSTL), is well-suited for. Specifically, we identify the low-data transfer problem. We assume the existence of two domains: a source domain and a target domain representing different problems, with different environments or datasets. For this problem, we assume two cases, with different effects on PSTL. First, where a source domain with a larger dataset is available and a target domain with less training data. Second, with low data for both the source and target domain. In this thesis, we mainly focus on the second case. When given enough training samples on the source domain, the source model will have a better representation of source task, compared to the model trained with limited data on the source task, But we expect to see an improvement from the PSTL approach in both the cases, as this searches for the relevant gradients that can help to optimize the model on the target task.

Our low-data transfer learning problem has two major characteristics. (1) A small amount of data is available for training. By a small amount of data, we do not indicate that this type of problem is solely relevant to supervised learning approaches. It is also relevant in RL setups in which we wish to minimize the number of interactions in an environment (i.e., if the environment is dangerous or costly to interact in). (2) We have a source domain with additional data or more episodes of training available.

In this thesis, we focus on three domains. First, we evaluated PSTL on the chip placement problem in a reinforcement learning-based environment [20]. Second, we evaluated this approach in the medical domain to evaluate the predictive performance of the health outcomes, in a regression-based framework [21], To further show the generalizability of the PSTL approach, we further demonstrated it in the image classification domain.

We now define the similarities of the source and target domains, In the RL task, we have similar actions in terms of placing macro components on a chip canvas. For the regression task, we predict the same output value prediction for both domains, but with entirely distinct input features. For the image classification task, we have the same output features between the source and target domains, but largely distinct input features.

## 3.2 System Overview

In this section, we present our approach Parameter Search Transfer Learning (PSTL). We divide our PSTL method into three main steps. First, we train the model on the source domain, which can be accomplished in whatever fashion best suits the domain. Second, we transfer the weights from the source to the target domain and apply PSTL to further optimize the parameter values to attempt to improve the model representation for the target domain. Third, apply fine-tuning on the target domain dataset to further converge and improve the model representation for the target task. This third step is omitted in the second case of our problem definition in which we lack sufficient knowledge in the source and target domain.

## 3.3 Step 1: Source Training

Our first step is to train a model on the source domain. For each of the domains, we use a domain-dependent training setup. We employ an existing policy and value network architecture [20] for the Chip Placement Problem and train it using a PPO optimization and proxy cost (detailed in Section 4.2.2) to estimate the reward. For the medical domain, we use a simple 4-layer LSTM architecture and train it using gradient descent using the Adam optimizer with Mean Absolute Error(MAE) as the loss function. For the image classification domain we use CifarNet [14] as our model and categorical cross-entropy as the loss function.

## 3.4 Step 2: Parameter Search

The second step of our approach is our parameter search which acts as transfer learning to adapt the source domain model to the target domain. We use a neuroevolution-inspired method for the optimization process, as is commonly applied in Neural Architecture Search [29]. We adapt the usage of search-based optimization over a neural network, but in this case, focused on a neural network's parameters instead of its architecture. We find that this approach allows us to identify different gradients than traditional backpropagation, as it allows us to more specifically target individual parameters.

We first initialize a population of models with different parameters, based upon a set of defined mutation operations, then optimize the parameters of the architectures guided by a fitness function. We represent the whole process in Algorithm 1.

The first line of the algorithm refers to step 1 of PSTL, training the model on the source domain. All the remaining lines are responsible for parameter searching. Line 2-3 refers to initializing the source model to the population of the networks which acts as the base model for searching parameters. We generate a population of architectures with different parameters using mutation operations and append the child models to the population of the networks as mentioned in lines 6-8. In Line 10 we then compute the fitness score for all the models in the population based on the training loss and select the best model based on its fitness score. Line 11 refers to the reduce operation, where we rank the population of the models using the fitness score, where we pick the best model the one with the highest fitness score. Finally, we update the population of the models with just the best model as this follows the greedy optimization, where we choose the best model of the population as shown in line 12. The mutation operation directly alters the model parameter values. We describe the details of the mutation operations in the next section.

**Algorithm 1:** PSTL Workflow

```
Input: An source $A$, the population size $pop\_size$, maximal
       generations $gen$, the $source$ dataset, and the $target$ dataset.
Output: Best performing architecture(optimal parameters)
1  $A \leftarrow$ train $A$ on $source$;
2  $A = best\_model$;
3  $pop = \{A\}$;
4  i $\leftarrow$ 0;
5  while $i < gen$ do
6      while $|pop| < pop\_size$ do
7          $network \leftarrow$ Mutation($best\_model$);
8          $pop$.append($network$);
9      end
10     $fitness\_pop$ = Fitness($pop$);
11     $best\_model \leftarrow$ Reduce($pop$, $fitness\_pop$);
12     $pop = \{best\_model\}$;
13     i $\leftarrow$ i + 1;
14 end
15 architecture = best_model(pop);
16 Return architecture;
```

### 3.4.1 Mutation Operations

The design of the mutation functions is a crucial aspect of PSTL, as these functions implicitly define the possible gradients for our optimization approach to follow. Our mutation operations consist of four distinct operations, that directly alter the parameter values of the source model. Our mutation operations are similar to the first four mutation operations from CENAS [29], which in turn are typical mutation operations for the neural architecture search process. We adopt these operations as they follow simple numeric operations that give a systematic range of options to explore the search space in order to improve model performance on the target domain. Our mutation operations are as follows:

- The first operation randomly chooses a particular layer or a filter of the network and adds scalar values from the range of [0, 1], using a uniform distribution.
- The second operation acts as the first, but instead subtracts values.
- The third operation acts the same as the first two, but instead multiplies the values.
- The fourth operation acts the same as the first three, but instead divides the values.

### 3.4.2 Fitness Function

We define the fitness function that guides PSTL in a domain-dependent manner. In this subsection, we describe the fitness functions for our three domains. First, for the chip placement problem, we use a fitness that's very similar to the reward function from circuit training [20]. The reward function here is the proxy cost estimation, which is a combination of the wirelength, congestion and density of the placement as shown in equation 3.1. These are three variables identified by domain experts to be the most important in terms of a particular chip design. The values of the $\gamma$ and $\lambda$ are set to 0.5 as mentioned in the original paper [20].

$$\text{proxy cost} = \text{Wirelength} + \gamma \cdot \text{Density} + \lambda \cdot \text{Congestion} \tag{3.1}$$

Second, we have the regression task, which predicts future health outcomes given an input DNA sequence. We use the mean absolute error of the target output values as the fitness function to evaluate the model parameters as shown in equation 3.2, where $y_{ij}$ is the actual value and $\hat{y}_{ij}$ is the predicted value and $m$ is number of output features in a sample and $n$ is the number of training samples.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}|y_{ij} - \hat{y}_{ij}| \tag{3.2}$$

Third, for the image classification task, we employ a simple categorical cross entropy as shown in the equation 3.3, Where $N$ is the number of classes in the classification task and $y_{ij}$ represent the ground truth label for j-th class in the i-th image and $p_{ij}$ is the predicted probability of the j-th class in the i-th image.

$$\text{cross\_entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n} y_{ij}\log(p_{ij}) \tag{3.3}$$

## 3.5 Step 3: Postprocessing

In step 3, we perform an additional post-processing fine-tuning step. We apply this step on the tasks with enough training data (i.e., on Chip Placement and Image Classification tasks) as this could help to further improve our representation via gradient descent. This step is also beneficial for cases with limited training samples, in cases with low-variance in the output features. This refers to the case where there are no distinct classes (output values of range 0-1) in the dataset (i.e., regression tasks like our health outcome prediction). But this is not as helpful in the case where there are limited samples and high-variance datasets (i.e., low-sample image classification tasks). We hypothesize that the reason is due to the gradients that backpropagation can identify in these cases and a tendency to diverge rather than converge in the latter case.

# Chapter 4

# Experimental Setup

This chapter provides a comprehensive overview of the experimental setup employed in this thesis. Section 4.1 presents readers with an overview of the target problem we are tackling and the related domains used to evaluate the performance of the PSTL approach. In section 4.2 we introduce readers to ablations of PSTL and general transfer learning baselines, which allow us to compare the performance between PSTL and these approaches. In section 4.3 we cover details about the architectures selection of all three different domains. Sections 4.4, 4.5 and 4.6 detail the experimental setup of the three different domains. In section 4.4 we cover the details of the chip-placement problem, section 4.5 covers the DNA Health Outcomes task, and section 4.6 details the image classification task.

## 4.1 Experiment Overview

In this work, we dealt with two different transfer learning problems one with sufficient training data in the source and target domain, and the other with limited data in the source and target domain. We compare the effect of our PSTL approach in terms of how quickly the model can achieve results at early stages in tasks with sufficient training data and how well PSTL can achieve generalizable models in tasks with insufficient data compared to naive transfer learning approaches. To evaluate our PSTL approach, we have chosen three different problem settings in different domains that require transfer learning.

- The first is a reinforcement learning problem setting, the chip-placement

Problem, which has sufficient training data for both the source and target domains.

- The second is a regression-based problem setting, the health outcomes prediction task, which has limited data for both the source and target domains.
- The third is an image classification problem setting, we evaluate both settings with sufficient and limited training data in both the source and target domains.

## 4.2 Baselines

In this work, we employ a total of three different baselines. The first two baselines (i.e., Scratch Model and Fine-tuning) are used to compare the performance of Full PSTL across all three steps (includes source training, greedy parameter search and postprocessing), as all these baselines employ gradient descent. The third baseline (zero-shot) can be compared against 2 Step PSTL given that neither employs traditional gradient descent on the target domain data.

### 4.2.1 Scratch Model:

This refers to the model achieved when training from scratch on the target domain, which means the parameters of the networks are initialized with random values, without any pre-trained parameter values or information from other source models. The Scratch Model serves as a fundamental baseline in neural networks as this provides a reference point to evaluate the effectiveness of transfer learning approaches. We identify domain-dependent scratch model training setups based on prior work.

### 4.2.2 Fine-tuning:

This involves taking the parameters of the source model, and further training that model on the new, target domain dataset. This functionally acts as an ablation of PSTL as it is just the first and third steps of our approach. This allows us to identify the value of the second step, and to support our claim about the second step allowing us to reach a better gradient. The details of the fine-tuning approach for different domains are presented in the corresponding experiment sections.

### 4.2.3 Zero-shot learning:

This refers to a model trained on the source domain data that inferences on the unseen data of the target task. This is a common transfer learning approach [39], and it is also functionally an ablation of PSTL, just including the first step. This method helps us determine the impact of the latter two steps of PSTL.

### 4.2.4 2 Step PSTL:

This refers to the first two steps of PSTL, where we train the model from scratch on the source domain and then perform the parameter search on the target domain data. As discussed above, we hypothesize that this approach will outperform the full three steps in cases with limited source and target domain data and high variance in the output features, the term variance refers to the range of NN output value prediction range.

### 4.2.5 Full PSTL:

This indicates all three steps of the PSTL approach. This is our complete proposed approach, which we hypothesize will outperform existing transfer learning approaches in cases with sufficient data and in cases with limited source and target domain with low-variance in the output features.

We performed all experiments using the cloud computing resource of Compute Canada with 18 CPU cores and 2xNVIDA v100 Volta (48GB memory). We maintained a consistent random seed across all experiments for reproducibility.

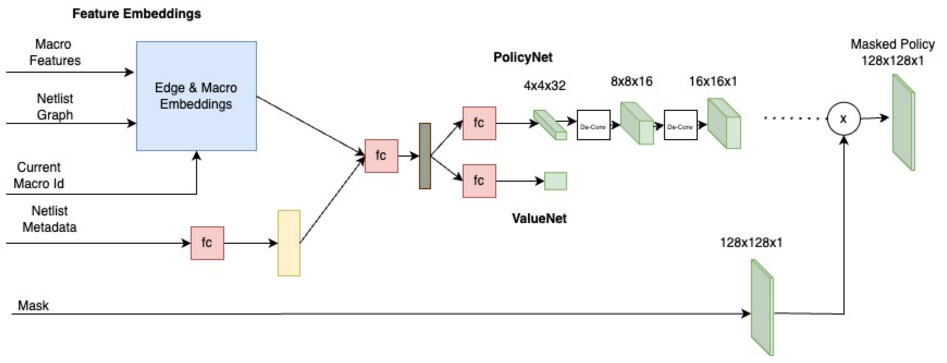

Figure 4.1: Policy and value network architecture for the chip-placement architecture.

## 4.3 Architectures

In this thesis, we focus on the experiments with transfer learning with both limited and sufficient training data in the source and target domains. Since we deal these settings in different domains (i.e., three domains in this case, RL-domain, regression-problem, and image classification), we adapt different architectures for each use-case based on the domain.

The first domain is an RL-based problem, which deals with the chip-placement task approached from a transfer learning perspective, in which we have sufficient data in the source and target domain. The chip-placement problem with reinforcement learning was first attempted by a team from Google [20], which also demonstrated results for fine-tuning, a transfer learning approach. This is a type of graph optimization, where the nodes in the graph are the macro units to be placed and the edges are wire connections between macrounits. We have employed a similar architecture to Google [20] for this task to compare PSTL and other baselines in the transfer learning setting. The architecture for this tasks has two main parts as shown in Figure 4.1. The first part computes the graph embedding for the input netlist (an adjacency matrix, which indicates the connection between two macro units). These graph embeddings are then used as input to the Policy and Value Network, where the policy network outputs the probability distribution over all the possible locations on the canvas of the chip and the value network approximates the estimated reward for the current placement. The mask is used

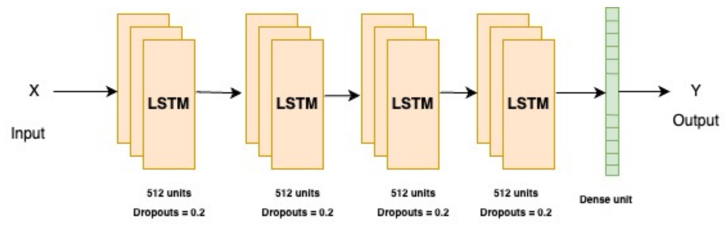

Figure 4.2: LSTM-based architecture used for predicting the Health outcomes task.

to avoid repeated placements at the same location of the canvas. The whole placement process is divided into two steps. First, using the above policy and value network we first place the macro units onto the canvas. Second, we use a force-directed method to place the standard cells (usually very tiny and millions in number), this approach clusters the standard-cells and places the cluster onto the canvas after the macro-units are placed. The RL-agent only approximates the reward once the macro-units and standard-cells are placed.

The second domain is the task of predicting the health outcomes given DNA methylation. Since DNA methylation is a hugely long sequence, where each part of the sequence is dependent on other parts we choose to consider an LSTM-based model that can capture the long-term dependencies in the sequence. The details about the data and information about the pre-processing steps are described in Section 4.5. We chose a simple LSTM-based architecture inspired by prior work, which also dealt with a low-data transfer learning problem [19], as this model solved a similar task in predicting financial health outcomes instead of physical health outcomes. Our model has 4 LSTM layers and 512 units and droprate of 0.2 at each layer, followed by a Dense layer with 44 output neurons to predict the health outcomes. We implemented this in the Keras framework, and the model uses all the default parameters for its LSTM and Dense layers. The whole LSTM architecture for this task is shown in Figure 4.2. Moreover, we intentionally choose not to rely on a more complex architecture, because this is a low-sample dataset, which can cause a more complex architecture to easily overfit. We use the Adam optimizer with a learning rate of 0.001 and MAE for the loss function.

The first domain has sufficient data in the source and target domain. The second domain has limited data in the source and target domain. To evaluate the effectiveness of the PSTL approach we introduce a third domain, the image classification domain. For this task, we use a simple Convolutional Neural Network(CNN) architecture. We employ CifarNet as the base architecture due to its use in other transfer learning work [29] which dealt with limited data for the target domain. CifarNet has two convolutional layers and each layer is followed by max-pooling, at the end it has two fully connected layers. We do not change this architecture, and employ a similar implementation from the original paper [14]. We use the default parameters for the experiments as used in the previous work [29]. The details of our benchmarks for the image classification domain are described in Section 4.6.

## 4.4 Chip-Placement Problem

Our first experiment focused on the chip-placement problem, here we had sufficient training data in both the source and target domain, meaning that we let the RL agent attempting to solve a given chip-placement task have enough interactions with the environment while training to converge. We evaluate the performance of PSTL on a total of 6 chip design tasks taken from prior work. The first design is Ariane RISC-V CPU design from the Google circuit training repository with 133 macro units, which is used as a standard reference for the chip-placement problem [20], the second design is Araine133 [5] which is a slight variant of the first design, with 133-macro units as well. We also evaluate the performance on three designs from the ICCAD04 benchmark [5]. We only pick three smaller designs for this benchmark, IBM01, IBM02 and IBM03 with 246, 271 and 290 macro units respectively, because these were closer to the other problems in size. These designs are referred to as “netlists” and “tasks” interchangeably in the literature, and we follow that trend.

For the transfer Learning setup in the chip-placement problem, we follow a kind of cross-fold validation, where training contains different netlists (tasks) that could be used to transfer the knowledge to a new netlist (task). A visualization of the difference between training a policy from scratch and using the

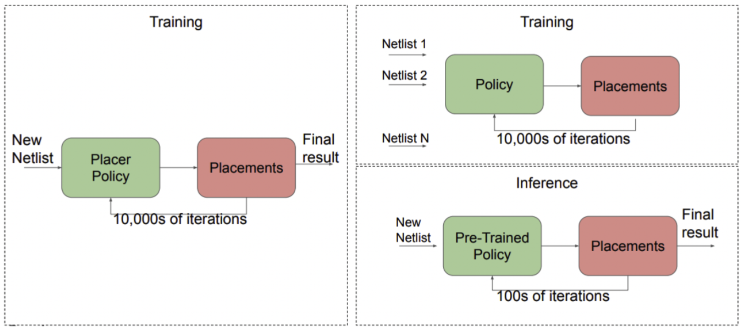

Figure 4.3: **Left:** Training the model from scratch on every Netlist file for the chip-placement problem. **Right:** Transfer Learning setup for the chip-placement problem, training on multiple netlists and inference on a new netlist.

transfer learning setup is demonstrated in Figure 5.1.

In this transfer learning setup, we consider one design as a source domain and the other as a target domain. For example, we use the Araine RISC-V design as the source domain and transfer the trained model to the remaining five target tasks (i.e., Ariane133, IBM01, IBM02, IBM03). We use the same training setup as the Google Circuit Training Repository [20]. We train all the models from scratch and run fine-tuning training for 700k iterations, which is approximately 4 GPU days of computation. For the second step of PSTL, we search for 50 generations.

## 4.5 DNA Health Outcomes

We next evaluate the performance of PSTL on a Regression task, i.e., prediction of health outcomes given a DNAm (DNA methylation) sequence. Here we have limited training data on both the source and the target domain. In this transfer learning problem, we have a total of 3 datasets, survey data from parents of children at birth, which has around 44 features, DNAm sequences of the same children at birth (187k sequence length), and health outcomes including features like height, and weight for the same children at set intervals (7 years, 15 years, etc.) This final dataset also has around 44 features. The data is collected from the lab of Jane Cook (Clinical Lecturer at the University of Calgary) and David Olson (Professor at the University of Alberta, Health

Sciences).

For the transfer learning setting, we train the model from survey data to health outcomes as the source domain and attempt to transfer from the DNAm data to the health outcomes as the target domain. This is based on insights from our domain expert partner (Dr. Cook), who hypothesized that such a transfer should be possible. Our domain expert partner further identified that a successful model predicting health outcomes from DNAm data would help improve pediatric care.

The DNAm sequences are 187k characters sequence long. We tried several approaches to compress this information to a latent vector, but Principal Component Analysis (PCA) gave us the best representation. We represent the 187k sequence in 500 principal components. The dataset size for both the source and target domain are 400 samples for training and 40 samples for testing. Furthermore, 10 percent of the training data is used for the validation. To prevent the model from overfitting, the baselines undergo limited training followed by backpropagation with early stopping.

## 4.6 Image Classificaiton

For our third setting, we evaluate the performance of PSTL on image classification tasks. We use four tasks and benchmarks from prior domain adaptation work [13]. The datasets used are MNIST, USPS, CIFAR-10 and STL-10, which are commonly used benchmarks for image classification tasks, where MNIST and USPS are digit classification datasets and CIFAR-10 and STL-10 are object classification datasets with mostly overlapping classes. In our transfer learning setting, we use one dataset as the source and the other as the target domain to create four tasks: CIFAR-10 $\rightarrow$ STL-10, STL-10 $\rightarrow$ CIFAR-10, MNIST $\rightarrow$ USPS and USPS $\rightarrow$ MNIST (i.e., USPS is the source domain and the MNIST is the target domain.) We produce two variant tasks. One where we have sufficient data, using all available train and test data ("enough"), and a toy variant with insufficient data ("limited") which follows a similar ratio of train-test split from the DNA Health outcomes task i.e., 400 train samples and 40 test samples and 10 percent of the training data is used for validation.

# Chapter 5

# Experiments

In this chapter, we discuss our experimental results on different domains. In sections 5.1, 5.2 and 5.3 we provide detailed results and key insights from the experimental results on different domains. In section 5.1 we provide detailed results for the chip-placement problem, in section 5.2 we discuss the results for the health outcomes prediction task and in section 5.3 we detail the results of the image classification domain.

In this thesis, we focus on three different transfer learning problem settings, varying in terms of the amount of data available, as mentioned in Problem Definition of Chapter 3. We categorize the experiments into two main settings based on the availability of the data. First, we evaluate PSTL on several tasks that have been used in prior transfer learning work [17], [20]. These tasks have sufficient training data on the source and the target domain, with sufficiency determined by the success of existing transfer learning approaches. This serves to demonstrate evidence to our claims that PSTL can output a better model that outperforms approaches that solely employ backpropagation on the target domain dataset. Second, we also evaluate PSTL on tasks with limited training data on both the source and target domain, this serves to demonstrate evidence to claims on how PSTL can deal in limited data settings.

For the parameter search phase of PSTL we run for 50 generations and set the maximum population size of the generated architectures to 10. We intentionally selected these small values for the search generation and population size to showcase the effectiveness of the parameter search with minimal

Table 5.1: Average proxy cost estimation of the baselines on different chip designs.

| **Method/Design** | **Ariane** | **Ariane133** | **IBM01** | **IBM02** | **IBM03** |
|---|---|---|---|---|---|
| Scratch Model | 1.06578 | 0.73733 | **1.99398** | **2.51126** | 3.19955 |
| Zero Shot | 1.11318 | 0.97043 | 2.62337 | 3.8361 | 3.08051 |
| Fine-tuning | **0.80677** | 0.7777 | 2.54906 | 3.05547 | 3.76074 |
| 2 Step PSTL | 1.25811 | 0.84247 | 2.28112 | 2.87091 | 2.47178 |
| Full PSTL | 1.05392 | **0.67646** | 2.26223 | 2.75768 | **2.27273** |

computational resources.

## 5.1 Chip-Placement Problem

For the chip-placement problem, we train the scratch model, fine-tuning and backprop of the Full-PSTL approach for around 700k iterations which are approximately around four GPU days of computing. The reward of the placement is only approximated once the macro unit and the standard cells are completely placed. For the 2 Step PSTL method we iterated the search process for 50 iterations.

| **Method/Design** | **Ariane** | **Ariane133** | **IBM01** | **IBM02** | **IBM03** |
|---|---|---|---|---|---|
| Zero Shot | 0.0489 | 0.0838 | 0.1512 | 0.7781 | 0.2489 |
| Fine-tuning | 0.1528 | 0.1443 | 0.2026 | 0.7753 | 0.2180 |
| 2 Step PSTL | 0.0051 | 0.0662 | 0.0775 | 0.1995 | 0.1496 |
| Full PSTL | 0.1239 | 0.0072 | 0.0691 | 0.1038 | 0.2005 |

Table 5.2: Standard Deviation of the proxy cost estimation of the baselines on different chip designs.

### 5.1.1 Results

We present the performance of the baseline approaches in terms of the average and standard deviation using proxy cost estimation, calculated by equation 3.1 in Table 5.1 and Table 5.2 respectively. Here, the design represents the target domain, and the values in the table are the average proxy cost of the baselines for the target domain, when transferred from different source domains (i.e., all the designs except the one chosen from the target domain). Overall, Full PSTL outperforms the baselines for the Ariane133 and IBM03 designs and the

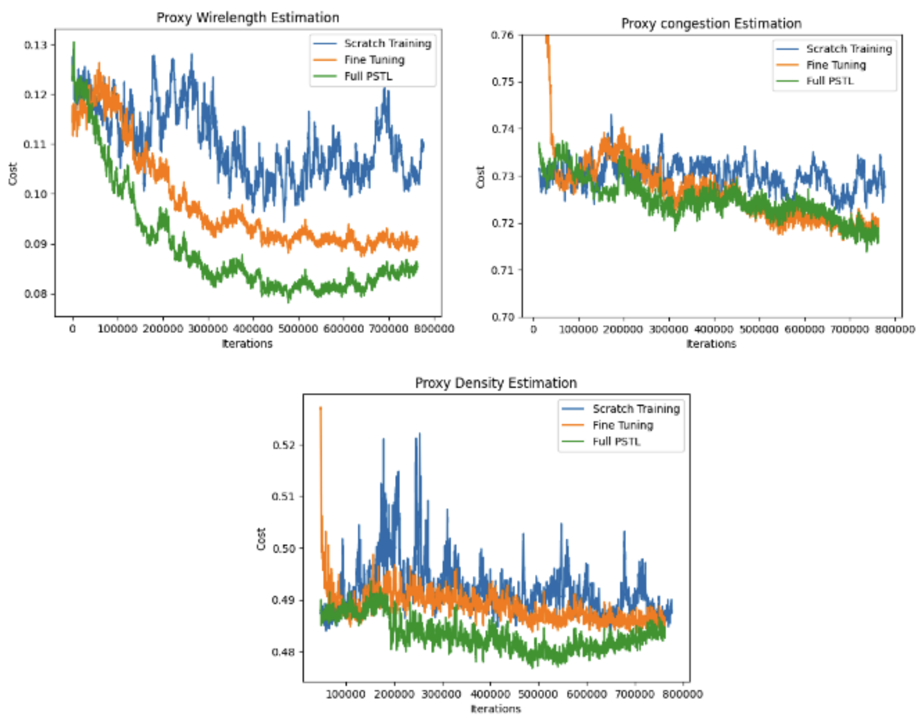

Figure 5.1: Proxy cost estimation of different metrics on 3 baselines when transferred from Ariane to Ariane133. We note that these results are only one of the four runs used to derive the averages presented in Table 5.1.

Scratch Model outperforms in two out of five designs. We also observe that Full PSTL is either the best or the second-best model in all the transfer domains. This indicates the backpropagation after the first two PSTL steps benefits from the better gradient discovered by the parameter search. These experiments work towards answering RQ2 and RQ4, demonstrating how PSTL can help to achieve better performance in target tasks by computing the variation of the gradient in 2-Step PSTL, and how further post-processing can help to achieve reliable performance compared to naive transfer learning approaches.

We note that fine-tuning is equivalent to the CT baseline approach from Google [20], which we outperform in all but one case. After roughly 10K episodes, the scratch model can achieve the best results and eventually, the fine-tuned model will converge to the results of the scratch model, which follows the results found in this prior work [20]. This prior work also found that the fine-tuned model could achieve better results in the early stages of the relevant target task. For example, we found that using a generic design like Ariane as the source design could help to achieve better fine-tuning results in almost all the target domains. But not all the source domains helped to achieve better fine-tuning results. We hypothesize this is because the target domain must unlearn some unwanted features in order to adapt to the target task. Additionally, we note that the proxy cost estimation for the Ariane design is roughly equivalent to the performance of current, non-transfer learning approaches [5].

In most cases, we observe that Full-PSTL can achieve better performance even faster than the fine-tuning approach. For example, in Figure 5.1 we observe the learning curves for our three metrics (wirelength, congestion and density values) for the task, where the source domain is Ariane and the target domain is Ariane133. The graph indicates that Full-PSTL was able to converge the fastest in the early stages of training (around 200-400k iterations), especially in the case of the wirelength and congestion. For the wirelength metric Full-PSTL was able to achieve the fastest results and to find the best result. For this use-case, we could observe that the transfer learning-based approach has achieved the best and fastest results compared to scratch training, in all three metrics. In general, PSTL, is able to achieve better and faster results on the target task when transferred from a relevant source task.

Comparing the 2-Step PSTL and Zero-Shot results, we can see that the former outperforms the latter in the majority of cases. This supports the value of the parameter search step and suggests that the 2 Step variant may be helpful in cases where it is dangerous or expensive to train on the target domain traditionally.

## 5.2 DNA Health Outcomes

For the health outcomes setting, we follow a similar fashion for training, we choose the architectures from the section 4.3 to train both the source and target domain, where the source domain is a mapping between survey data to the health outcomes and the target domain maps DNAm sequences to health

Table 5.3: Mean absolute loss on the test data for the health outcomes prediction

| **Model** | **Test loss** |
|---|---|
| Scratch Model | 0.6441 |
| Zero-Shot | 0.6639 |
| Fine-tuning | 0.6449 |
| 2 Step PSTL | 0.6312 |
| Full PSTL | **0.6163** |

Table 5.4: Mean absolute error on test data for NN vs non-NN approaches

| **Baseline** | **Test error** |
|---|---|
| Linear Regression | 8.8128 |
| Logistic Regression | 2.9364 |
| NN (Zero-Shot) | 0.6639 |
| NN (Fine-tuning) | 0.6449 |
| NN (Full PSTL) | **0.6163** |

outcomes. We use gradient-descent to train the fine-tuning, scratch-training and Full-PSTL baselines and use early-stopping to avoid over-fitting. For 2-Step PSTL we follow a 50 step iteration process.

### 5.2.1 Results

For this setting, we found that Mean Absolute Error (MAE) led to a consistent performance in terms of training loss. We present the performance of different baselines on the target task using MAE in Table 5.3. The performance of both PSTL variants clearly shows it is beneficial in low-data tasks. The 2-step PSTL approach gained a significant improvement in terms of MAE compared to the Zero-Shot approach. We also observe that the training of this task ends within very few epochs, because this task is a low-sample dataset which can easily lead to overfitting in the target task. These experiment results address our RQ3 and RQ4, demonstrating the benefit of using PSTL approaches in low-sample data tasks. The results suggest that by targeting individual parameters we can achieve more generalizable models compared to other transfer learning baselines.

The general benchmarks for these tasks in this domain are linear and logistic regression [21]. We also compare against these approaches as shown in Table 5.4. The NN-based models clearly outperform linear and logistic regression, which is because the sequence-based models help to capture the dependencies of the long DNA sequence and so can capture the information more effectively. Although there is a slight improvement in terms of MAE relatively among the NN models, PSTL has the best performance. These

Table 5.5: Accuracy of the image classification benchmarks on the target domain

| **Baseline** | **STL10 → CIFAR10** | | **CIFAR10 → STL10** | | **USPS → MNIST** | | **MNIST → USPS** | |
|---|---|---|---|---|---|---|---|---|
| | enough | limited- | enough | limited | enough | limited | enough | limited |
| Scratch Model | 82.1 | 10 | 52.35 | 11.1 | 99.44 | 10 | 80.32 | 10 |
| Zero-shot | 10.53 | 11 | 11.587 | 9.2 | 14.71 | 10 | 8.47 | 10 |
| Fine-tuning | 81.05 | **12.5** | **68.09** | 11.1 | **99.55** | 10 | 84.853 | 10 |
| 2 Step PSTL | 10.66 | **12.5** | 11.637 | **11.7** | 12.48 | 18 | 12.51 | **15** |
| Full PSTL | **82.37** | 5 | 57.61 | 11.1 | 99.53 | **92** | **94.52** | 10 |

experiments work towards answering RQ1, where the ANNs provide an advantage in low-data tasks compared to traditional ML approaches. Overall, the results are not perfect, but our domain expert partner identified that they were sufficient for their purposes and that the Full PSTL approach being the most precise made it the most useful in terms of identifying links between DNAm and health outcomes. We note that the goal of this task was to find important input DNA sequences responsible for health outcomes, and also to remove unwanted - variability that is correlated to the input DNAm sequence, but these topics are a bit out of the scope of this thesis. Here we solely try to show the effect of the PSTL approach in these domains.

## 5.3 Image Classification

This section provides results for the image classification tasks. We train our baselines as in the prior section and our 2 steps PSTL follows a 50 step iterations.

### 5.3.1 Results

We present the test accuracy for each task in Table 5.5 for our two settings with sufficient and insufficient data. Overall, for the task with sufficient training data, PSTL has the best or roughly equivalent performance except for the CIFAR10 → STL10 task, where the fine-tuning model has the best accuracy instead. We anticipate this is because STL10 shares the same input features given that it shares the same classes and style (photography) as CIFAR-10 making this a "near transfer" problem, which gives an edge for fine-tuning when transferring the knowledge from the source task.

The performance of the non-transfer learning approaches (i.e., scratch model training for MNIST and CIFAR-10), closely aligns with other non-transfer approaches with similar models [25]. It is worth noting that the architecture used in their paper was LeNet [16], which is also small, but differs slightly from CIFARNet. Regardless, this helps confirm that our non-transfer learning approaches are accurate implications of typical ways to train these models.

For tasks with limited data for the source and the target domain, the 2 Step PSTL approach benefits all the transfer tasks (i.e., improvement compared to zero-shot accuracy). Overall, the performance of the baselines with backpropagation don't seem to have much improvement, where the performance is the same as the zero-shot approach or slightly better. We anticipate that this is because backpropagation is not beneficial in the case of training with low samples, as the model starts over-fitting in just 4-10 epochs for all tasks. But for the USPS $\rightarrow$ MNIST tasks backpropagation after the first two PSTL steps helps to a massive degree. We expect this is because MNIST is a simple dataset, where even a few images can be sufficient to learn a generalizable representation of the dataset. This section contributes to addressing RQ2, RQ3 and RQ4, confirming the results from the previous domains.

## 5.4 Discussion

PSTL demonstrated robust performance both in cases with sufficient data for traditional transfer learning, and in cases with insufficient data for traditional transfer learning. This supports our hypothesis that searching the parameters of the network allows us to find better gradients than backpropagation alone for many transfer learning problems.

For the task with limited data in the source and target domain, we observed that in general for these tasks, using greedy optimization for identifying gradients for individual parameters of the network can help to find better parameters of the network for the target task. Whereas the effect of backpropagation differed from dataset to dataset depending on the variance of the dataset. For example, the health outcomes prediction is a low-variance dataset, where the target label is predicting values in the range [0-1] following something like a normal distribution. In comparison, the high-variance image classification dataset required predicting a class in the range [0-9] as a one-hot vector. We anticipate that the first two steps of PSTL would prove helpful in high-variance, low-data transfer problems in the future.

# Chapter 6

# Conclusion

In this thesis, we propose Parameter Search Transfer Learning (PSTL), an effective stochastic greedy optimization method designed for transfer learning tasks. It searches over the parameter values of Deep Neural network models and identifies novel gradients, which cannot be achieved through naive optimization approaches. The thesis aims to address the challenge of dealing with limited training data in both the source and target domains and also compare the effectiveness of the PSTL approach in the task with sufficient training data on both the source and target task. We demonstrated the effectiveness of PSTL in an RL-environment, a regression-based problem and an image classification problem. We demonstrated that PSTL can help to achieve better performance at the early stages of training compared to complex optimization approaches like backpropagation in cases with sufficient data. Our results also demonstrate that on tasks with limited training data, using PSTL can help us to find more generalizable models. We find that our approach is especially valuable in low-training cases and in cases with low-data and high-variance. We hope PSTL will serve as a robust tool to expand the set of domains where we can apply neural networks.

## 6.1 Future Work

There are many potential improvements that could be explored for the PSTL approaches. This works only considers the relevant mutation functions that were used from our prior work. The current mutation functions are static, which means we fix the size of the values that need to be added to the network, it could be possible to explore more complex mutation functions which can find gradients in a more dynamic manner. It could also help to explore more optimization functions to support tasks with low data. It would also be interesting to explore the relevancy of relevant factors from the source tasks that contribute to transfer to the target task, as well as investigating the generalizability of the search algorithm. We hope to explore this in future and evaluate the impact on the tasks with low-sample problems.

## 6.2 Takeaways

Looking back on the research questions this thesis attempted to answer, for RQ1 we did observe that using ANN-based models for the tasks is very effective, but this is mainly dependent on the complexity of the architecture. For RQ2 and RQ3 using PSTL on tasks with sufficient and limited training data respectively, we were able to achieve equivalent or better performance compared to our baselines. These results do not fully answer if this is useful in all cases, and we identify a need to explore the effect of this approach with "near" vs "far" transfer in future work. Some of our PSTL results also demonstrated a risk of getting stuck in a local minimum, which might lead to sub-optimal solutions. For RQ4, in many tasks we did observe that PSTL did perform equally or better in comparison to general transfer learning baselines. This doesn't suggest this is always the optimal solution for these transfer learning settings. This further indicates directions for exploring different optimizations for PSTL.

# References

[1] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2021, pp. 1551–1559.

[2] A. Azzini and A. G. Tettamanzi, "Evolutionary anns: A state of the art survey," *Intelligenza Artificiale*, vol. 5, no. 1, pp. 19–35, 2011.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[4] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "Assessment of reinforcement learning for macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, 2023, pp. 158–166.

[6] Y. Cui, S. Wang, and J. Li, "Lstm neural reordering feature for statistical machine translation," *arXiv preprint arXiv:1512.00177*, 2015.

[7] A. Dahou, M. A. Elaziz, J. Zhou, and S. Xiong, "Arabic sentiment classification using convolutional neural network and differential evolution algorithm," *Computational intelligence and neuroscience*, vol. 2019, 2019.

[8] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories. ieee trans," *Pattern Recognition and Machine Intelligence*,

[9] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated machine learning: Methods, systems, challenges*, pp. 3–33, 2019.

[10] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Papers on Twenty-five years of electronic design automation*, 1988, pp. 241–247.

[11] F. Günther and S. Fritsch, "Neuralnet: Training of neural networks.," *R J.*, vol. 2, no. 1, p. 30, 2010.

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

[13] J. Hoffman, E. Tzeng, T. Park, *et al.*, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*, Pmlr, 2018, pp. 1989–1998.

[14] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[15] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*, IEEE, 2009, pp. 951–958.

[16] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[17] Y. Li and X. Peng, "Network architecture search for domain adaptation," *arXiv preprint arXiv:2008.05706*, 2020.

[18] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with relu activation," *Advances in neural information processing systems*, vol. 30, 2017.

[19] A. Mahajan and M. Guzdial, "Modeling individual humans via a secondary task transfer learning method," in *Federated and Transfer Learning*, Springer, 2022, pp. 259–281.

[20] A. Mirhoseini, A. Goldie, M. Yazgan, *et al.*, "Chip placement with deep reinforcement learning," *arXiv preprint arXiv:2004.10746*, 2020.

[21] K. Miyake, C. Miyashita, A. Ikeda-Araki, *et al.*, "Dna methylation of gfi1 as a mediator of the association between prenatal smoking exposure and adhd symptoms at 6 years: The hokkaido study on environment and children's health," *Clinical epigenetics*, vol. 13, no. 1, pp. 1–11, 2021.

[22] D. M. Nelson, A. C. 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)*, Ieee, 2017, pp. 1419–1426.

[23] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: A review of applications," *Expert systems with applications*, vol. 36, no. 1, pp. 2–17, 2009.

[24] F. J. Pontes, G. Amorim, P. P. Balestrassi, A. Paiva, and J. R. Ferreira, "Design of experiments and focused grid search for neural network parameter optimization," *Neurocomputing*, vol. 186, pp. 22–34, 2016.

[25] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," 2017.

[26] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International conference on learning representations*, 2017.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[28] M. Sarrafzadeh, M. Wang, and X. Yang, *Modern placement techniques.* Springer Science & Business Media, 2003.

[29] M. Singamsetti, A. Mahajan, and M. Guzdial, "Conceptual expansion neural architecture search (cenas)," *International Conference on Computational Creativity*, 2021.

[30] K. Smagulova and A. P. James, "A survey on lstm memristive neural network architectures and applications," *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[32] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[33] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, "Proud: A sea-of-gates placement algorithm," *IEEE Design & Test of Computers*, vol. 5, no. 6, pp. 44–56, 1988.

[34] L. Valeri, S. L. Reese, S. Zhao, *et al.*, "Misclassified exposure in epigenetic mediation analyses. does dna methylation mediate effects of smoking on birthweight?" *Epigenomics*, vol. 9, no. 3, pp. 253–265, 2017.

[35] P. Villalobos, J. Sevilla, L. Heim, T. Besiroglu, M. Hobbhahn, and A. Ho, "Will we run out of data? an analysis of the limits of scaling datasets in machine learning," *arXiv preprint arXiv:2211.04325*, 2022.

[36] 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.

[37] 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)*, IEEE, vol. 1, 2017, pp. 639–645.

[38] P. Wiklund, V. Karhunen, R. C. Richmond, *et al.*, "Dna methylation links prenatal smoking exposure to later life health outcomes in offspring," *Clinical epigenetics*, vol. 11, no. 1, pp. 1–16, 2019.

[39] 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.