Meta-Learning for Cancer Phenotype Prediction from Gene Expression Data

Mandana Samiei

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science)
Concordia University
Montréal, Québec, Canada

January 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Mandana Samiei**
Entitled:	**Meta-Learning for Cancer Phenotype Prediction from Gene Expression Data**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
Dr. Aiman Hanna

______________________ Examiner
Dr. Adam Krzyzak

______________________ Examiner
Dr. Andrew Delong

______________________ Supervisor
Dr. Thomas Fevens

Approved ______________________
Dr. Lata Narayanan, Chair of Department

January 21, 2020 ______________________

Dr. Amir Asif,
Dean, Gina Cody School of Engineering and Computer Science

Abstract

Meta-Learning for Cancer Phenotype Prediction from Gene Expression Data

Mandana Samiei

Deep learning has become an essential element in various applications of technology over the past decades. Deep neural networks are now reaching performance on par with, or even beyond, human-level on a broad range of tasks. However, there are still several concerns and deficiencies that make these models impractical for some real-world applications. One of the important issues comes from a data-efficiency perspective. Most of the deep learning techniques need a large number of training samples in order to achieve a high performance on a given problem. This procedure is far from human general intelligence. Humans are good at learning from a few number of samples and quickly adapting to new tasks. In this work, we leverage the meta-learning framework in which the model can learn new tasks by developing prior knowledge over past experiences. For this purpose, we propose a Meta-Dataset that contains 174 genomics and clinical tasks. Furthermore, we suggest a meta-model under the few-shot learning regime that can learn new genomics tasks. Finally, a comparison between the performance of the meta-learner and the performance of other classical baselines is also presented.

Acknowledgments

This work would have not been possible without the support of my supervisor, Dr. Thomas Fevens, who tirelessly worked to provide me with countless opportunities to learn and grow as a researcher. In the moments when I needed help and when I was stressed, he always supported me and gave me incredible advice, which were always great motivations to keep me going. It might happen during graduate studies to be disappointed and lost in your goals. It may happen to be suspicious of what you have done so far. But during all these moments, you should keep moving forward, and keep working for your ambitions. In tough times, you should seek advice from your supervisor and other researchers, such as Ph.D. students who have also been on a similar path in the past.
I would also like to extend my gratitude to Joseph Paul Cohen who I worked with during my collaboration with Mila. He provided many insightful comments and feedback for me.
Also, I would like to thank Yoshua Bengio, the director of Mila, who allowed me to work in an excellent environment, and for sharing his perspectives on developing human general intelligence.
I would like to thank my colleagues and fellow lab mates at Concordia who were always heart-warming supporters to me. Thanks to Mehrzad Mortazavi, Laya Rafiee and Justin Whatley.
Furthermore, I would like to offer a special thanks to my friends and all the people at Mila who have helped me during this collaboration. Thank you all for your unwavering support, patience, and encouragement. Thanks to Basile Dura and Tristan Deleu for all the helpful discussions and brainstorming. Also thanks to Clément Jumel, Paul Bertin, Tobias Würfl, Geneviève Boucher, Hannah Akbarzadeh, Stanislas Le Guisquet and Raphaël Limbourg .
Last but not least, I'm so grateful for my family for their unconditional kindness and support throughout the whole journey. Of course, none of this would have been possible without them.

Contents

List of Figures **viii**

List of Tables **x**

1 Introduction **1**

1.1 Meta-learning and Few-shot Learning 1
1.2 Phenotype Prediction Problem 2
1.3 Contributions of this Thesis 3
1.4 Outline 4

2 Background **6**

2.1 Biological Context 6
2.1.1 Introduction 6
2.1.2 Motivations 6
2.1.3 Genomics Basic Concepts 7
2.1.4 Gene Expression Data 10
2.2 Machine Learning Context 12
2.2.1 Introduction 12
2.2.2 Supervised Learning Tasks 12
2.2.3 Semi-supervised Learning Tasks 14
2.2.4 Unsupervised Learning Tasks 14
2.3 Optimization and Evaluation 14
2.3.1 Gradient Optimization 15
2.3.2 Gradient Descent 16
2.4 Baselines Introduction 17
2.4.1 K-Nearest Neighbor Classifier 17
2.4.2 Logistic Regression 17

2.4.3 Multilayer perceptrons (MLP) and Deep Neural Networks 19
2.5 The Development of Deep Learning 20
2.6 From Deep Learning to Meta Learning 22

3 Meta-learning 23
3.1 Introduction 23
3.2 Applications of Meta-Learning 24
3.3 Meta-Learning Vs. Multitask Learning, Transfer Learning, and Domain Adaptation 24
3.3.1 Multitask Learning 25
3.3.2 Transfer Learning 25
3.3.3 Domain Adaptation 26
3.4 Meta-Learning Algorithms 27
3.4.1 Gradient-Based Meta-Learning 28
3.4.2 Metric-Based Meta-Learning 28
3.5 Mathematical Formulation 32
3.5.1 Meta-Dataset and Meta-Parameters 32
3.5.2 Adaptation to a New Task 34
3.6 Meta-learner Selection Choice 36

4 Learning Cancer Phenotypes Using Meta-learning 37
4.1 Introduction 37
4.2 Motivations 38
4.3 Dataset 38
4.3.1 Types of Cancers Selected for Study 39
4.3.2 Genomics and Clinical Attributes Selected for Study 43
4.4 Task Definition 45
4.5 Related Works 46
4.6 Torchmeta Library for Gene Expression Data 46
4.7 Data-Loaders for Few-Shot Learning 47
4.8 Experimental Setup and Technical Details 48
4.8.1 Evaluation Strategy 50
4.8.2 Training Prototypical Networks 50
4.8.3 Simple Mode Setup 52
4.8.4 Complex Mode Setup 58
4.9 Results 58

4.10 Conclusion 61

5 Conclusion and Future Work **62**

5.1 Future Work 63

5.1.1 Meta-Learning Sensitivity to Task Distributional Changes 63

5.1.2 Task Specification 63

5.1.3 Task Distribution Diversity Quantification 63

5.1.4 Similarity Metric Definition 64

5.1.5 The Role of Memory in Meta-Learning 64

Bibliography **71**

List of Figures

1 Task definition process. 4

2 From genome to DNA, genome is located inside the nucleus and it includes 23 pairs of chromosomes. The figure is inspired from [5]. 8

3 DNA building blocks showing how nitrogenous bases are paired together. 9

4 This figure shows the procedure of transcription and translation. Protein-coding genes are transcribed to mRNA first, then translated to proteins. RNA-coding genes are transcribed to a functional non-coding RNA [4]. 10

5 Gene expression data matrix where n and m are the number of genes and samples respectively and w_{ij} is miRNA abundance. 11

6 An overview of gene expression data matrix from different cancer studies and their corresponding phenotype. 11

7 Gradient descent optimization procedure, conceptual representation of the gradient, initial weight, learning step and derivative of cost. 16

8 An example of a multilayer perceptron with 3 input features. 18

9 A sigmoid function. 19

10 A visual representation of a neural network with 3 layers, a single hidden layer with h hidden units, input x $\in R^m$ and output $y \in R^t$. 20

11 Visual representation of meta-learning framework where training tasks are different but coming from the same source domain, aims at learning a model generalizes to new tasks which are still from the same source domain. The image is inspired from R. Aljundi.[11]. 25

12 Visual representation of multitask learning framework where training and test tasks are coming from the same distribution. The image is inspired from [11]. 26

13 Visual representation of transfer learning setting. Inspired by [11]. 26

14 Visual representation of domain adaptation where source and target tasks are the same but drawn from different data distribution. Inspired by [11]. 27

15 Visual representation of embedding learning where f and g are train and test embedding functions and s is the similarity metric. The image is illustrated by inspiration from [61]. 29
16 Prototypical networks in the few-shot learning framework. Few-shot prototypes v_k are computed as the mean of embedded train examples for each class. The distances from class 1 and 2, are d_1 and d_2 respectively, since $d_2 < d_1$, x^{test} is classified as class 2. The figure is illustrated by inspiration from [56]. 30
17 Prototypical networks algorithm from J. Snell's work [56]. N is the number of examples in the training set, K is the number of classes in the training set. RANDOMSAMPLE(S,N) denotes a set of N elements chosen uniformly at random from support set S, without replacement. 31
18 Conceptual representation of inductive biases for a binary classification task on images. J_i in the figure denotes an optimization track that leads the model to the right hypothesis space. 33
19 Visual representation of meta-learning phase to learn a prior knowledge over hypothesis space through θ and adaption phase when the model learns a new task through θ'. 35
20 Few-shot classification results on Meta-dataset [59] using models trained on ILSVRC-2012. The image is credited to [59]. 36
21 A visual representation of the training pipeline. 51
22 The average loss and accuracy for event prediction using 18 tasks in the meta train set. 53
23 The average train loss and accuracy for gender prediction using 16 tasks in the meta-train set. 54
24 The average train loss and accuracy for oct embedded prediction using 12 tasks in the meta-train set. 55
25 The average train loss and accuracy for histological type prediction using 4 tasks in meta train set. 56
26 The average train loss and accuracy for colon polyps presence prediction using 1 task in the meta-train set. 57
27 Prototypical Network performance on the defined genomics tasks. 58

List of Tables

1 Definition to the basics of genomics if we consider our genome profile similar to a book. 8

2 The results of hyperparameter search, learning rate sets to 1e-3 when tweaking the architecture. Also, when tuning the learning rate, the neural network has 2 layers with 128 and 64 hidden nodes in each layer respectively. 49

3 The accuracy of event prediction tasks across models. ProtoNet has been trained on 18 different cancers and tested on 5 new ones. 53

4 The accuracy of gender prediction tasks across models. The meta-learner has been trained on 16 different cancers and tested on 5 new cancers. 54

5 The accuracy of oct embedded prediction tasks across models. In ProtoNet, the model has been trained on 12 different cancer types and tested on 4 new ones. . . . 55

6 The accuracy of histological type prediction tasks across models. In the prototypical network, the model has been trained on 4 different cancers and tested on a new cancer. 56

7 The accuracy of colon polyps presence prediction task across models. In the prototypical network, the model has been trained on COAD cancer and tested on COADREAD. 57

8 Few-shot classification results on Meta-test set tasks using classical models and meta-learner trained on Meta-training set tasks only using 5 samples. The evaluation metric is accuracy and it has been provided in an ascending order. 59

9 Accuracy and 95% confidence interval of models over multiple trials. 59

Chapter 1

Introduction

In this chapter, I give a brief introduction to my thesis. First of all, I introduce the meta-learning and few-shot learning techniques. Secondly, I describe the importance of phenotype prediction problem using gene expression data. Thirdly, the contributions of this thesis are explained. Finally, I present the outline of this thesis.

1.1 Meta-learning and Few-shot Learning

There is a close connection between Meta-learning and Few-shot learning. Few-shot learning includes algorithms that are used for learning to predict a task using only a handful of samples. In other words, these tasks are learned under a small-data regime. One of the well-known algorithm to solve these problems is Prototypical Networks [56]. In which the classifier generalizes to new unseen classes during the training phase using only a few samples.
Meta-learning [14, 12, 22, 27, 49, 54] is a mechanism which can be used for automating human decisions such as parameter tuning, algorithm designing and automatically revisiting and optimizing those decisions dynamically in the light of new experience obtained during learning [6]. Another application is transferring knowledge between multiple related tasks. Meta-learning may reduce the efforts for designing domain-specific learning algorithms, and lead to more robust and general learning architectures. Apart from practical applications, research on meta-learning algorithms may also lead to new insights into human learning [32]. In this work, we are interested in a specific form of meta-learning called Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (MAML) [22]. Meta-learning in this context is often addressed as a framework rather than an algorithm, as proposed by Chelsea Finn [22], "it is compatible with any model trained

with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning". Meta-learning includes two optimization steps. First is a slow learning process that extracts a general transferable knowledge across all defined tasks, this guides the second process which is a faster learning scheme and extracts task-specific information. Meta-learning is one of the methods to approach few-shot learning problems where the dataset is small. However, it is not only specific to few-shot learning settings. This is a broad research direction that is worth exploring. In this work, I'm particularly interested in improving data efficiency, meaning how meta-learning proceeds using a lower amount of data as opposed to the original deep learning framework. More technical detail on Meta-learning is given in Chapter 3. The words small data regime and few-shot learning setting are used interchangeably in this thesis. Although meta-learning gives interesting perspectives on the future of artificial intelligence, I have to direct your attention to the fact that current algorithms have not shown significant success on industrial applications and at this stage still remain a challenging research problem. Nevertheless, I believe that a better comprehension of existing methods and failure cases will help us bridge this gap faster.

1.2 Phenotype Prediction Problem

The term "phenotype" refers to the observable physical properties of an organism; these constitute the organism's appearance, development, and behavior. Examples of phenotypes include height, skin type, and hair color. Phenotypes also include biochemical and physiological properties that can be measured in laboratories, such as levels of hormones or blood type [1]. In this thesis, we are predicting cancer-specific phenotypes using gene expression data. The input of each task is gene expression data (miRNA abundances) and the output is the value/label of a phenotype. Microarray gene-expression profile has generally been used to predict cancer outcomes [45]. Gene-expression data provide a molecular signature to doctors and medical team allowing them to facilitate the subsequent clinical management of patients [34]. General studies of gene expression data help doctors to speed up the diagnosis and prognosis of cancer. This is becoming a necessity in cancer research, as it can improve the treatments of patients based on the severity of cancer. Predicting a phenotype from gene expression data is an important task in genomics decision making as in cancer prognosis and diagnosis [25]. Often, some of these phenotypes are related to each other under a particular cancer study. Most current research in the literature seeks to develop techniques and methods for predicting one particular clinical outcome. This approach is far from the reality of medical decision making in which you have to consider several factors simultaneously [26]. In

this work, we define each phenotype as a task for the model to learn based on the gene-expression profile of cancer patients. The model is trained using a few samples from each phenotype and eventually predict a new task. Deep learning models are often trained on a huge amount of examples of a particular task and then tested on the same type of samples (i.e. from the same data distribution). In the current literature of deep learning, we have seen intelligent systems performing well on detecting a specific phenotype such as gender, height [37], skin color, etc. [39], but this needs training the models on a large dataset and they are still not acquiring high performance on new tasks. Even though there are popular works published on multi-task learning mechanisms as in [13], the samples are used during test time are coming from the same tasks as those ones the model trained on. Therefore, we need more powerful tools to approach this issue.
It is also worth mentioning that there are some variability and systematic biases in gene-expression data. Usually, these biases are results of the technology and measuring mechanisms. For example, much research has been done to remove artifacts and spatial correlations in DNA microarray data [33, 57].

1.3 Contributions of this Thesis

In this assay, we are focusing on the small-data scenario due to data availability and privacy concerns in healthcare applications specifically cancer diagnosis. The amount of available data in different areas and applications is varying. For instance, doctors and physicians might have access to only a few samples for some rare diseases and cancers. Furthermore, there are many rules and regulations for data acquisition depending on the research domain. By using *meta-learning* [14, 54], we can improve data efficiency [63]. I have to mention that data efficiency and privacy criteria can also be addressed through other transfer learning approaches. We are particularly interested in transferring knowledge under a few-shot learning regime. In this thesis, the goal is to learn a model for phenotypes prediction of a given cancer from gene-expression data using a few samples. We define a collection of genomics tasks derived from the publicly available data hub called The Cancer Genome Atlas Program (TCGA). The input data is an individual expression levels (mRNA abundances) of every one of our 20,000 protein-coding genes. The tasks are phenotype prediction based on these expressions. We believe those tasks could be good proxy tasks to develop methods that can work on a few samples of gene expression data. The defined tasks cover a wide range of phenotypes including predicting tumor tissue site, white cell count, cancer histological type, family history of cancer, gender, and a few others which we explain later in Chapter 4. Each task represents an independent dataset associated with a phenotype and a cancer study. We propose a pipeline in which we use a model called Prototypical Network (ProtoNet)[56] which can learn new tasks using only a handful of training samples and compare its performance to logistic regression, k-nearest neighbor and a fine-tuned neural network. We implement ProtoNet [56] in a meta-learning framework similar to MAML [22]. Although ProtoNet implementation is already available at Jake Snell's repository [8], we implement it from scratch in order to leverage the MAML framework [22] and apply multiple meta-learning algorithms on our meta-dataset. Furthermore, we aim to use our dataloader written for the TCGA dataset. Below, a big picture of the task definition procedure is given. More information is provided in Chapter 4.

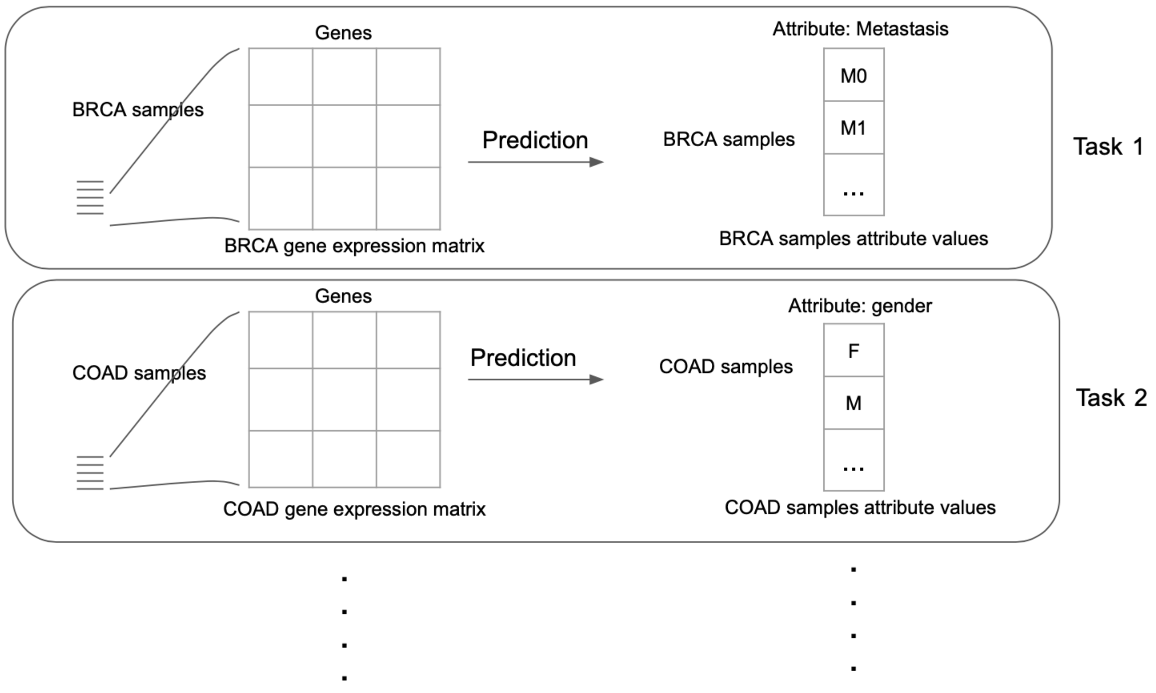


Figure 1: Task definition process.

1.4 Outline

This thesis is structured as follows. Chapter 2 is presenting a supplementary section on the biology and related assays that are optional for interested readers. Later in this chapter in section 2.2, we introduce the fundamentals of machine learning and deep learning by discussing a supervised learning framework, optimization and evaluation methods, and a description of classical machine learning models applied in this thesis. Then, we provide a brief introduction to deep learning and its link to meta-learning. In Chapter 3, a mathematical formulation of meta-learning, a couple of famous algorithms in this framework, and the challenges of meta-learner selection choice are presented. Chapter 4 explains the dataset and the task definition protocol. Later, we present Torchmeta dataloader for gene expression data, experimental and training setup, evaluation method, as well as a comparison of models' performances across different genomics tasks. Chapter 5 concludes this thesis and discusses some future work and research directions.

Chapter 2

Background

2.1 Biological Context

In this chapter, I will give a review of the basics of biology required to follow the work described in the following chapters.

2.1.1 Introduction

Genomics data is of great value as they can be used in many important applications such as detecting cancers, customizing disease treatments, predicting clinical attributes that can help doctors to improve their analysis and also helping researchers to discover patterns in these attributes for particular cancers. Most researchers develop new methods for one particular clinical task at a time. As an example, we can name survival prediction [65] or tumor cell type classification [42]. This approach is far from the reality of clinical decision making in which you have to consider several clinical variables simultaneously that is often performed by physicians and clinical teams. In some cases, those prediction tasks are associated with each other.

2.1.2 Motivations

Advances in human genome research are opening the door to many useful applications. Cancer diagnostics is an active field of research. Personalized medicine and targeted therapies derived from an individual genome profile, have both impacted the way patients get treated these days [44], hoping for more efficient treatments that save time and money for the patients. By knowing about various phenotypes using patients' genome profiles, we can achieve a more reliable outcome of cancer prediction. To solve complex clinical problems, it can be more promising to consider underlying inter-related tasks within them. Additionally, in some cases, these tasks have small datasets. By considering a few-shot learning setting, we can leverage the samples from a collection of tasks and construct a transferable knowledge over previously seen problems. This ability of versatility is a key aspect of intelligence as articulated in [15]. This idea has been developed in several applications of robotics and image recognition/classification as in [22, 21, 59]. In this work we would like to expand this hypothesis to phenotype prediction problem.

2.1.3 Genomics Basic Concepts

In this section, I will explain a brief overview of common terminologies in biology. Most of the material that has been explained in the chapter is based on Genome-Quebec Tutorials [5].

Cells are the smallest units of life. In all living beings, cells share common abilities such as nutrition, adaptation, response, and reproduction. A cell interacts with its environment, but it can also adapt to it by modifying its physiology or behavior when changes occur. It also relates to its environment by responding to the various stimuli it encounters. The cell nucleus is the largest part of the cell. The nucleus holds most of the DNA (deoxyribonucleic acid) in the shape of chromosomes and the mechanism needed for its decoding and replication. The genome, located in the nucleus of each cell, is divided into 23 pairs of chromosomes. Each chromosome is coiled and compacted so that it fits inside the nucleus. If it is unwinded, a long strand of DNA will be obtained. Genes are stretches of DNA. Most biological traits are inherited from many different genes and theirs interactions with environment. Some of these traits are visible, such as eye color or skin color, and some are not, such as blood type. DNA is a molecule found in all living beings and carries the genetic information needed for the organism to develop and function. The range of lengths of human protein-coding genes, is from 100bp (base pair) to 1Mbp.
A DNA molecule is shaped like a long twisted ladder, which is why it is often referred to as a double helix. The building components of the DNA molecule are pairs of nitrogenous bases facing one another. There are four different nitrogenous bases: adenine (A), thymine (T), cytosine (C) and guanine (G). When forming the DNA block, A always pairs with T and C always pairs with G. The order of the nitrogenous bases (ACCATTCGCT...) is what determines DNAs instruction, also known as the genetic code. DNA is associated with different proteins and found in the nucleus of the cell. Figure 2 present a good visualization of above-mentioned concepts including DNA, chromosome and nucleus. [1]

[1] All the images in this work have been drawn by the author using https://www.draw.io/ unless otherwise is mentioned.

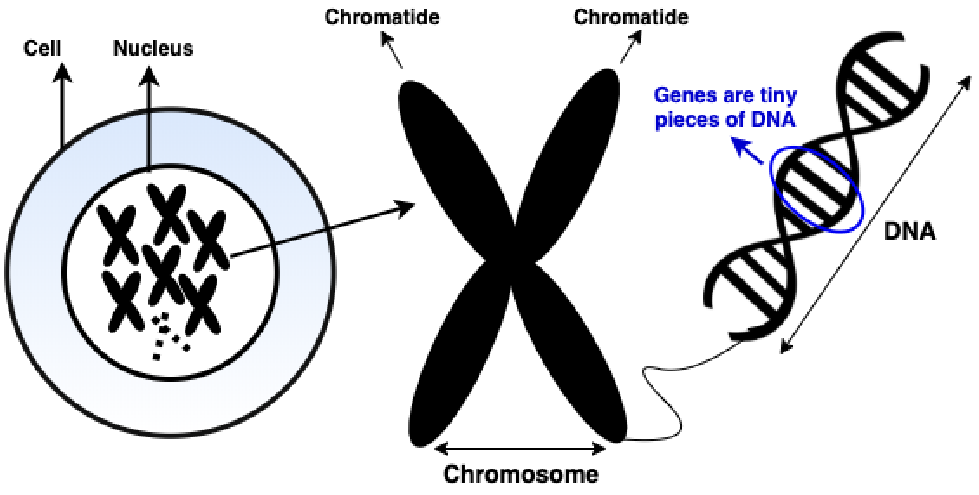


Figure 2: From genome to DNA, genome is located inside the nucleus and it includes 23 pairs of chromosomes. The figure is inspired from [5].

Terminology	**Definition**
Genome Profile	Similar to an instruction manual that contains the specific traits of individual and their body's functioning mode.
Chromosome	The genome is divided into 23 pairs of chromosomes that we inherit from our biological parents.
Genes	Genes are like the sentences in the book of our genome.
DNA	DNA, is written with an alphabet of 4 letters (A,T,C,G) that combine to form all the words in the book.
Mutation	A mutation is similar to an alteration that can disrupt the development or functioning of the body.

Table 1: Definition to the basics of genomics if we consider our genome profile similar to a book.

2.1.3.1 Transcription and Translation

In order to read the encoded information in a gene's DNA and produce the protein it specifies, two steps are required. First, a copy of an RNA should be created from gene's DNA as a template for producing a protein. This copy, called a messenger RNA (mRNA) molecule. The mRNA acts as an intermediate between the DNA gene and its final protein product. It leaves nucleus and enters cytoplasm where it guides the synthesis of the protein. This process, is called **transcription**.

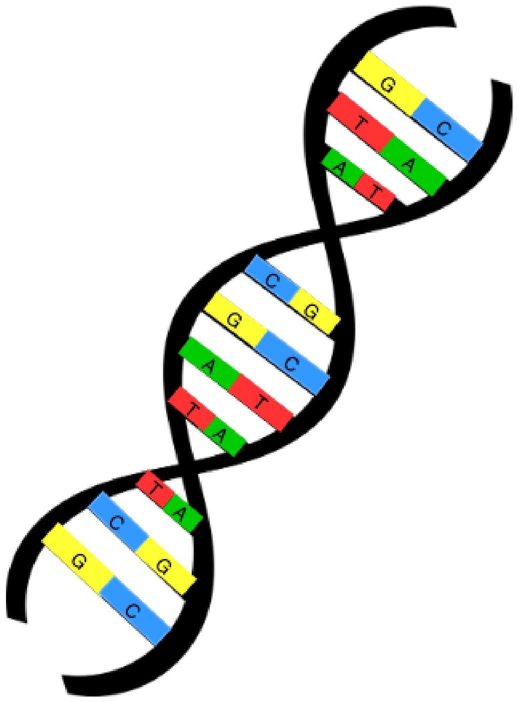


Figure 3: DNA building blocks showing how nitrogenous bases are paired together.

Second, that mRNA molecules are translated to protein. This step, is called **translation**. There are two types of genes that are participating to these processes. Protein-coding and RNA-coding genes. RNA-coding genes are parts of an organism's DNA that do not encode proteins. However, these genes must still go through transcription step but are not translated into proteins. Figure 4, distinguish these differences. In translation step, a sequence of mRNA molecules is translated to a sequence of amino acids during protein synthesis. In the cell cytoplasm, ribosomes are responsible for this process. They read the sequence of mRNAs in groups of three nucleotides at a time, in units called codons, to assemble the protein. The genetic code describes the relationship between the sequence of base pairs (nucleotides) in a gene and the corresponding amino acid sequence. The genetic code is the set of rules used by living cells to translate information encoded within genetic material such as DNA, RNA and Necleotide [4].

2.1.3.2 Genome Sequencing

Genome sequencing is the process of decoding DNA to its bases. In this procedure, the order of DNA nucleotide, or bases, in a genome (i.e. As, Cs, Gs, and Ts) are specified. The human genome is made up of over 3 billion of these genetic letters.

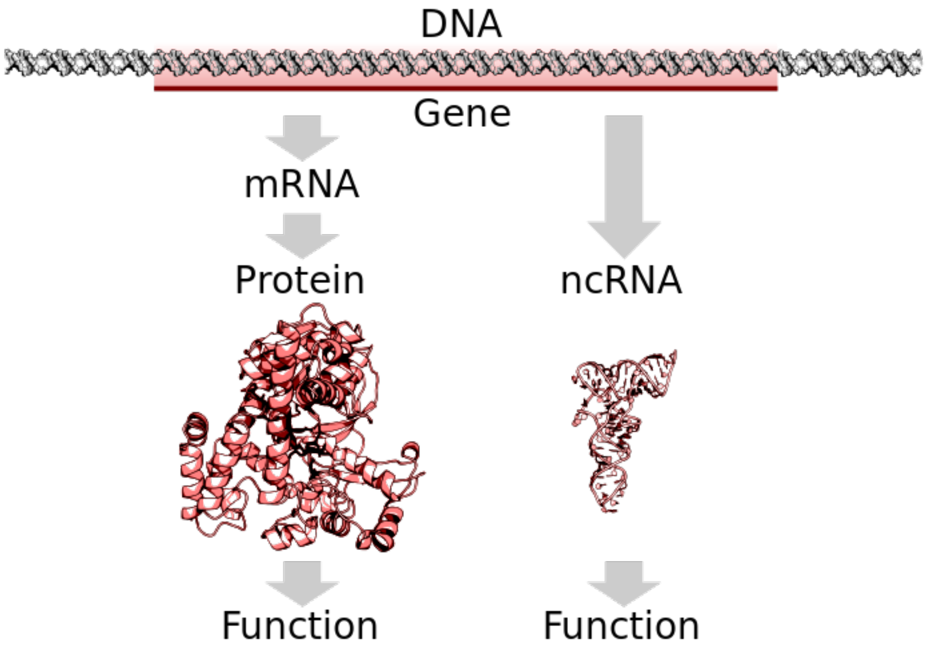


Figure 4: This figure shows the procedure of transcription and translation. Protein-coding genes are transcribed to mRNA first, then translated to proteins. RNA-coding genes are transcribed to a functional non-coding RNA [4].

2.1.4 Gene Expression Data

The process of producing a biologically functional molecule of either RNA or protein is called gene expression, and the resulting molecule is called a gene product. According to [20], gene expression measurement is achieved by quantifying levels of the gene product, which is often a protein. Understanding the level of gene expression in a cell, tissue or organism can provide valuable information such as determining cancer susceptibility. Using DNA chips (also known as DNA microarrays) and other biological engineering techniques, we can measure the expression level of a large number (possibly all) of genes belonging to a given organism. Roughly speaking, the gene expression data or DNA microarray data are conceptually a gene-sample/condition matrix, where each row corresponds to one gene, and each column corresponds to one sample or condition. Each element in the matrix is a real number and records the expression level of a gene under a specific condition [24]. The gene expression level is inferred by measuring the abundance of mRNA copied from each gene.

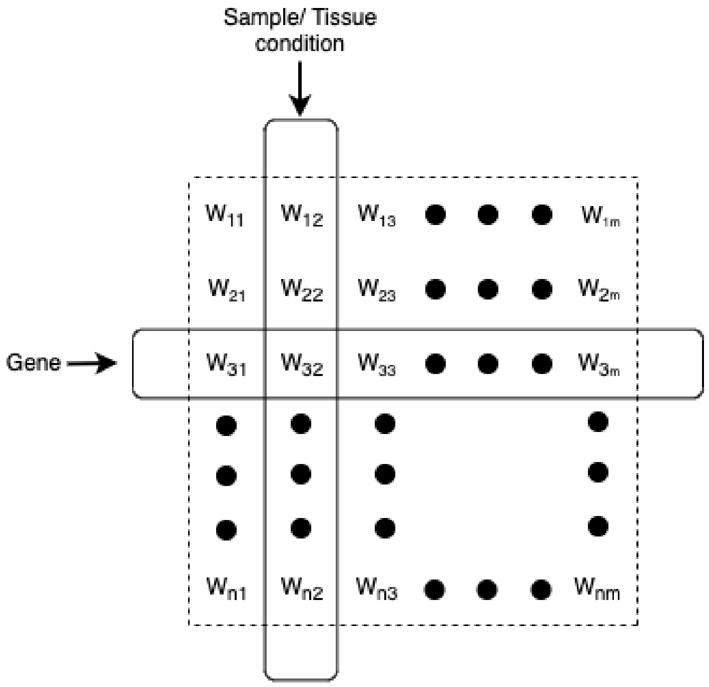


Figure 5: Gene expression data matrix where n and m are the number of genes and samples respectively and w_{ij} is miRNA abundance.

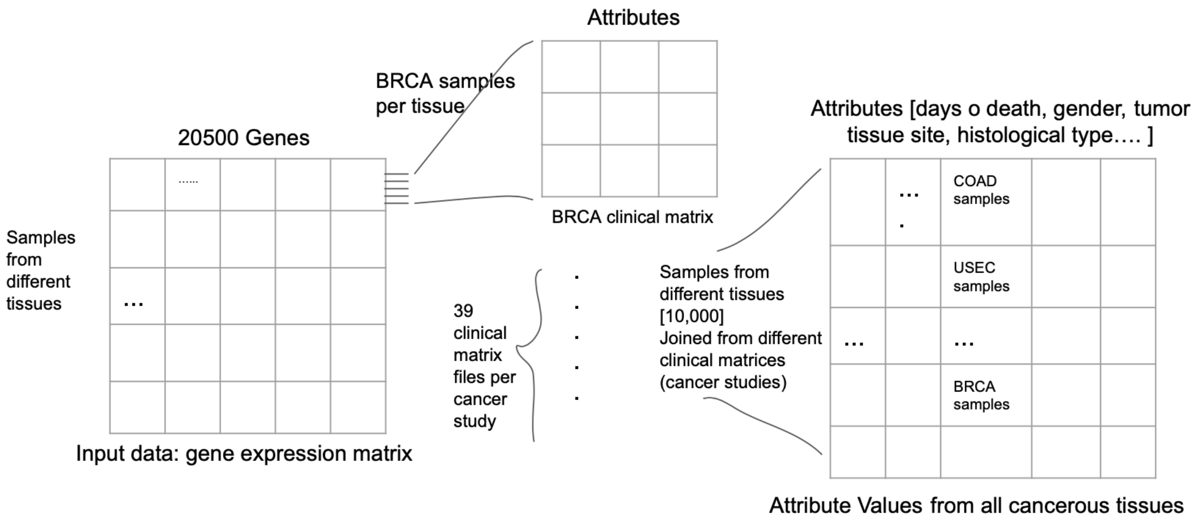

Attributes
BRCA samples per tissue
20500 Genes
Samples from different tissues
Input data: gene expression matrix
BRCA clinical matrix
39 clinical matrix files per cancer study
Attributes [days o death, gender, tumor tissue site, histological type….]
Samples from different tissues [10,000] Joined from different clinical matrices (cancer studies)
COAD samples
USEC samples
BRCA samples
Attribute Values from all cancerous tissues

Figure 6: An overview of gene expression data matrix from different cancer studies and their corresponding phenotype.

2.2 Machine Learning Context

In this section, the basics of machine learning will be reviewed. This background is required to follow the next chapters. This does not serve as a full review of machine learning nor deep learning. Should the reader be interested in a thorough review of machine learning and deep learning concepts the Deep Learning book [23] is an excellent resource.

2.2.1 Introduction

Machine learning is the study of computer algorithm that can learn and improve from a set of experiences. Machine learning algorithms builds a mathematical model from observations or training data and makes prediction on given examples (i.e test data) in future without being explicitly programmed to do so. Machine learning algorithms are applied in a broad range of applications, such as text classification, computer vision, language modelling, language translation, search engines and many others. Machine learning is closely related to statistics, linear algebra and mathematical optimization. Having knowledge on these mentioned fields is required to understand the basis of machine learning algorithms. Furthermore, machine learning is a valuable tool for several fields of research such as, computer vision, natural language processing, computational biology, neuroscience and computational psychology.
Machine learning tasks can be boiled down to a couple of categories of tasks in terms of availability and labeled data. Supervised learning, semi-supervised, unsupervised learning, and reinforcement learning tasks. I will be focusing on explaining supervised to unsupervised learning distinctions. Reinforcement learning is out of the scope of this work, if the reader is interested in knowing more, Richard Sutton and Andrew Barto's book [58] is highly recommended.

2.2.2 Supervised Learning Tasks

Supervised learning problems can be seen as tasks where the objective is to make a prediction based on the corresponding input data. Generally speaking, the objective is to build models such that for similar input, it makes similar predictions in the hopes of being able to make a correct prediction for a new sample. In other words, supervised learning problems aim to learn a mapping function from the input data to the associated label or output $Y = f(X)$.

2.2.2.1 Classification

A classification task consists of making a prediction regarding which class or group a given input associates with. Tasks including predicting if an image displays a cat or a dog, or classifying digits, or even facial recognition are all examples of classification tasks. Datasets used for classification tasks consist of the input samples as well as a corresponding label for each of the data samples. To design a classification algorithm, it is often required to know the possible labels or classes that the input samples may represent. Besides, we have to define an objective function to improve the model's prediction by minimizing the error. In a binary classification task when there are only two classes, we may predict the probability of each example belonging to the first class. In the case of multiple-class classification, we predict a probability for the example belonging to each of the classes. Under the maximum likelihood framework, the error between two probability distributions is measured using cross-entropy. Therefore, we often use the cross-entropy loss between the label distribution of the training samples and the model's probabilistic prediction.
Given training data $D_{train} = (x^{(1)}, y^{(1)}), ..., (x^{(i)}, y^{(i)}), ..., (x^{(N)}, y^{(N)})$, where $x \in R^d$ (d is input space dimension), $y \in \{1, 2, ..., C\}$ and $f_\theta(x^{(i)}) = [P(y^{(i)} = 1|x^{(i)}), P(y^{(i)} = 2|x^{(i)}), ..., P(y^{(i)} = C|x^{(i)})]$
The global cross-entropy objective can be defined as:

$$L(f_\theta(x^{(i)}), y^{(i)}) = -\sum_{j=1}^{C} 1_{\{y^{(i)}=j\}} log f_\theta(x^{(i)})_j \quad (1)$$

2.2.2.2 Regression

Similarly to classification, regression tasks can be seen as predicting a value for a given input. In regression, instead of selecting a possible category to associate a given sample, the goal is to estimate a real value. In other words, regression algorithms attempt to build a mapping function (f) from the input variable to a numerical or continuous output variable. A trivial example to understand the concept better is to consider estimating the value of a house. A real estate agent acts similarly to a machine learning model in a regression task. Given all the information about the characteristics of a house and its neighborhood, it attempts to determine a good price for the house to be listed on the market. The dataset in regression is similar to classification, but instead, a real value is associated with each of the data samples. The loss function is often the mean-square error (MSE), as it allows for computing losses between real numbers. Given training data $D_{train} = (x^{(1)}, y^{(1)}), ..., (x^{(i)}, y^{(i)}), ..., (x^{(N)}, y^{(N)})$, where $x \in R^d$ (d is input space dimension), $y \in R$ and $f_\theta(x^{(i)}) = \hat{y}^{(i)}$.

The MSE objective is as following:

$$\mathcal{L}(f_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(y^{(i)} - f_\theta(x^{(i)})\right)^2 \tag{2}$$

2.2.3 Semi-supervised Learning Tasks

In semi-supervised learning setting, a small amount of labeled data with a large amount of unlabeled data are often used for training. In general, labeled data is employed to help identifying the specific assortment of samples in the data. Then, the algorithm is trained on unlabeled data to define the region of those categories and might even find new types of data that have not been specified in the human-labeled input. Similarly to a supervised learning framework, a set of L examples $x_1, ..., x_L \in X$ with corresponding labels $y_1, ..., y_L$ has been given. In addition, there are U unlabeled samples $x_{L+1}, ..., x_{L+U}$. The semi-supervised method attempts to make use of this combined information to tackle the classification task that could be achieved either by discarding the unlabeled data and doing supervised learning or by dropping the labels and doing unsupervised learning. The goal is to infer the correct mapping from X to Y.

2.2.4 Unsupervised Learning Tasks

An unsupervised machine learning algorithm makes use of input data without any labels. In other words, no supervision will be provided to the program (learner) when it makes the right prediction or when it makes a mistake. Unsupervised learning focuses on the actual data itself rather than its relationship with a corresponding label or output. There is a broad range of unsupervised learning tasks, however; most of them aim to estimate the underlying distribution of the input data. Generally speaking, an important distinction between supervised and unsupervised learning methods is the absence of labels. Unsupervised learning often works well to discover new patterns in a dataset and to cluster the data into several categories based on several features. Popular methods are clustering, density estimation, and generative models. I do not provide more detail about the mentioned techniques as it's not the focus of this work. In the following sections, I give a brief introduction to the machine learning models I used in this project.

2.3 Optimization and Evaluation

To solve the previously mentioned machine learning tasks, an optimization framework of the problem has to be formulated. Once that is achieved, some algorithms can be used to maximize or minimize the objective. Machines learn by optimizing an objective function. It's a technique to evaluate how well an algorithm models the given data. If the prediction is far from the actual result, the loss function will punish the model by adjusting the model parameters. With the help of some optimization function, the loss function learns to reduce the error in predictions. Given a function f with parameters θ and a training dataset D_{train}, the global objective $J(\theta)$ can be written as follows:

$$J(\theta) = \frac{1}{|D_{train}|} \sum_{i=1}^{|D_{train}|} \mathcal{L}(f_\theta(x^{(i)}), y^{(i)}) \tag{3}$$

In the optimization procedure, what we want to achieve is to find the parameters θ that minimize $J(\theta)$, the empirical risk over the training data. The solution to the optimization problem can be defined as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta)$$

Due to the complexity of many machine learning tasks, the above-mentioned loss function is not easily tractable. This means solving the optimization problem is analytically close to impossible. Machine learning models usually do not minimize/maximize the true loss over the training data. In fact, the mathematical formulation is a surrogate loss that is easily tractable as in many cases when the real objective is not. For instance, the goal in a classification task is to minimize the miss-classification error on those samples that have not been seen by the model, i.e. generalization error. However, for the gradient optimization algorithms to work, the mathematical formulation has to be continuous and differentiable, where the generalization error may not be. If the loss is difficult to optimize, we use iterative techniques.

2.3.1 Gradient Optimization

Since many of the loss functions used in optimizing machine learning tasks are complex, instead of finding the analytical solution, the iterative methods can be used. In the case when the objective function is analytically solvable, the optimization procedure will find the regions that $\nabla_\theta J(\theta) = 0$, where the gradients of the objective function with regards to the parameters θ are zero. An alternative to bypass this is to consider an iterative solution such as gradient descent or ascent.

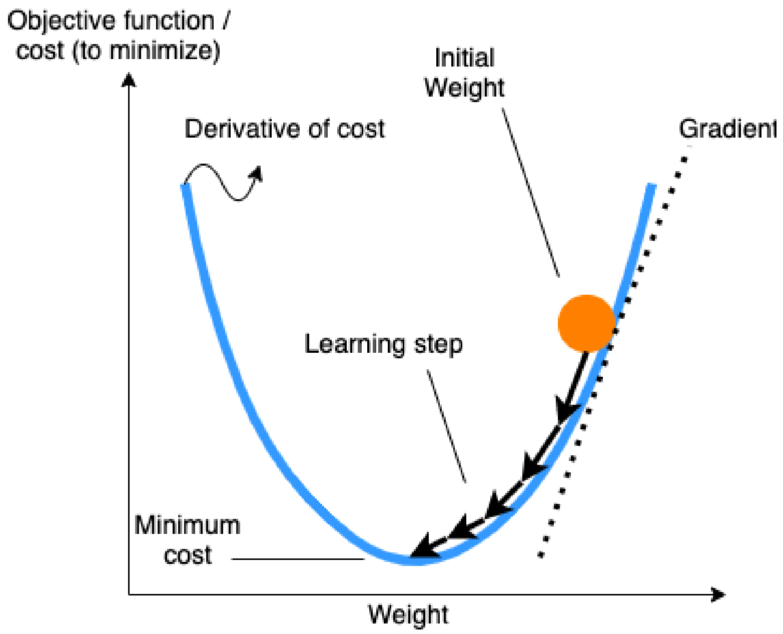


Figure 7: Gradient descent optimization procedure, conceptual representation of the gradient, initial weight, learning step and derivative of cost.

2.3.2 Gradient Descent

To find the optimum weights that minimize the objective function, we use an optimization algorithm called Gradient Descent. The gradient descent procedure can be interpreted as going down a hill, where we do not know the actual route down. According to Fig 7, a heuristic way to go downhill would be by looking for the angle of the hill at each step and take a step in that direction. If the hill is convex such that there are no valleys to restrict access to the bottom point, this approach is guaranteed to allow to reach the bottom of the hill. The direction of the hill will be computed by the gradient $\nabla_\theta J(\theta)$. Usually, this gradient is computed based on the whole dataset which is called batch(or standard) gradient descent. Unlike batch gradient descent, stochastic(or incremental) gradient descent(SGD) converges much faster, since it updates the weights more frequently. However, SGD tries to find optimums by iteration from a single randomly selected training example, the error is typically more noisy than in standard gradient descent. Also, to reduce the risk of getting stuck in local minima during the optimization process, instead of computing the gradient with respect to the full train set, noisy estimates of the gradients can be measured by picking a random subset of the dataset. This method is called minibatch stochastic gradient descent that is a compromise between computing the true gradient and the gradient at a single example, it has the advantages of faster and smoother convergence.

Algorithm 1 Minibatch Stochastic Gradient Descent

```
1: Given $f_\theta$ is a continuous and differentiable model
2: Choose an initial vector for $\theta$ and learning rate $\eta$.
3: Initial m as the size of minibatch
4: while Not Convergence do
5:     Randomly select m data samples from the training set
6:     $\theta \leftarrow \theta - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \mathcal{L}(f_\theta(x^{(i)}), y^{(i)})$
7: end While
```

2.4 Baselines Introduction

Here, I describe the machine learning and deep learning models that I used in this work. I start with more simple baselines such as k-nearest neighbor and logistic regression. Then I extend this to deep neural networks and meta-learning models.

2.4.1 K-Nearest Neighbor Classifier

The k-nearest neighbors' algorithm (KNN) is a supervised classification algorithm. To label a new example, the model considers the K nearest labeled samples to the query point. Then the model assigns the majority of k neighbors' labels to the query sample. This is the simplest form of KNN.

2.4.2 Logistic Regression

Logistic Regression (LR) is a transformation of linear regression using the sigmoid function. The logistic function also called the sigmoid function. One of the main issues in classification tasks occurs when the algorithm is not converging in the weight's update, while training. This often happens when the classes are not linearly separable. Logistic regression is a simple method but still powerful enough for classification tasks. LR uses a similar representation as in linear regression. To understand better where the sigmoid function is coming from, I will explain the odd ratio and the logit function in the following paragraphs [50]. The odds ratio is one important concept to understand in order to grasp the idea of Logistic Regression. The odds ratio is the probability that a certain event will occur. It can be written as follows where P stands for the probability of

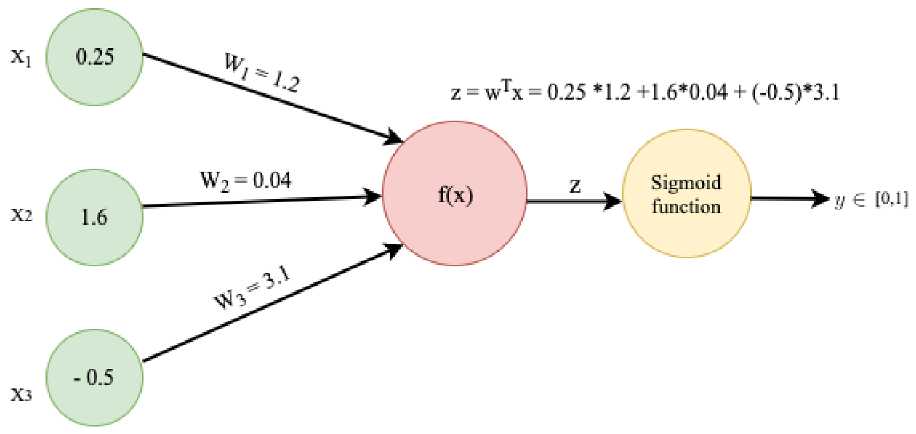


Figure 8: An example of a multilayer perceptron with 3 input features.

the event we want to predict.

$$OddsRatio = \frac{P}{1-P} \tag{4}$$

Derived from this we can define the logit function as in equation 5. The logit function takes input values in the range of $[0, 1]$ and transforms them to values over the entire real-number range $[-\infty, \infty]$.

$$logit(P) = log\frac{P}{1-P} \tag{5}$$

We use the logit function to express the linear relationship between features and the log-odds.

$$logit(P(y=1|x)) = W_0X_0 + ... + W_mX_m = \sum_i(W_iX_i) = W^TX \tag{6}$$

Where $P(y = 1|x)$ is the conditional probability that a particular sample belongs to class 1 given its features x. However, we care about the inverse of the logit function and is called the sigmoid function. The motivation is to predict the probability that a particular sample belongs to class 1 given its features x.
The sigmoid function is:

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{7}$$

Where z is the linear combination of weights and sample features and can be calculated as follows. Also, you can find an example of such a network in figure 8.

$$z = w^Tx = w_0 + w_1x_1 + ... + w_mx_m \tag{8}$$

As we can see from figure 9, $\phi(z)$ approaches to 1 when z goes to infinity and approaches to 0 if z tends to minus infinity. According to 8, the logistic regression model puts a sigmoid function

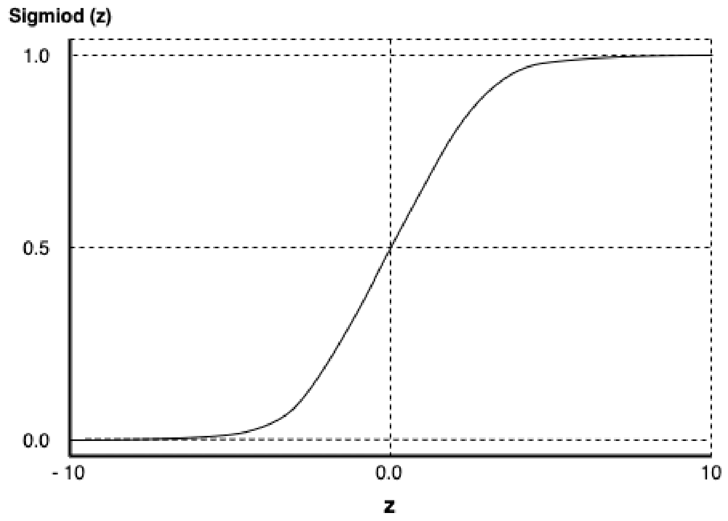


Figure 9: A sigmoid function.

as the last layer of the network which results in a probability distribution over samples given its features and its weights w.

2.4.3 Multilayer perceptrons (MLP) and Deep Neural Networks

2.4.3.1 Perceptron

A perceptron is a linear classifier that divides the input space into separating categories with a straight line. Input is usually a feature vector x multiplied by weight w and added to a bias: $y = w * x + b$. A single layer perceptron is only capable of linear classification as it does not include multiple layers to model a feature hierarchy.

2.4.3.2 Feedforward Neural Network and Multi-Layer Perceptron

A multilayer perceptron (MLP) is an artificial neural network (ANN). It has more than one perceptron. It includes an input layer to receive the data, an output layer that makes a prediction about the input, and in the middle, an arbitrary number of hidden layers that are the computational core of the MLPs.

MLPs are usually applied to supervised learning tasks. They are trained on a set of input and output pairs and learn to model the correlation between those pairs. The training involved updating the parameters, or the weights and biases of the model in order to minimize the loss.

Backpropagation is used to make these weight and bias adjustments with regard to the error.

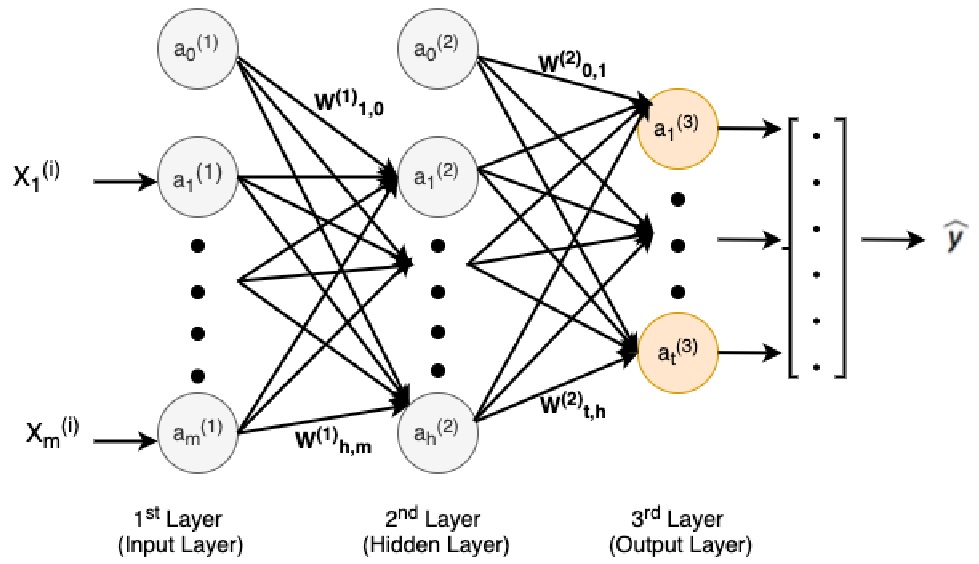


Figure 10: A visual representation of a neural network with 3 layers, a single hidden layer with h hidden units, input x $\in R^m$ and output $y \in R^t$.

Feedforward networks such as MLPs are bidirectional; there is a constant back and forth adjustments. In the forward pass, the signal moves from the input layer to the hidden layers then the output layer. The prediction of the output layer is measured against the ground truth labels. In the backward pass, by using backpropagation and the chain rule, partial derivatives of the loss function are backpropagated through the MLP with regards to the weights and biases. The fact of differentiation gives us an insight into the error. This can be done by any gradient-based optimization algorithm such as stochastic gradient descent. The network continues adjusting the parameter until the error can not go lower. This state of the network is called convergence.
However, the word *perceptron* in MLPs can be a bit confusing as we do not want only linear neurons in the networks. The goal is to learn the complex correlations between input and output space to solve non-linear problems. Thus, the network is conventionally composed of one or multiple hidden layers that connect the input and output layer. Now, this is where *deep learning* comes into play.

2.5 The Development of Deep Learning

In principle, deep learning contributes to a set of methods and algorithms that benefit us to parameterize deep neural network architectures. As you can see in figure 10, a deep neural network comprises many hidden layers and parameters. The essential idea behind deep learning is to derive high level features from the given dataset. Therefore, deep learning aims to solve the challenge of the often slow feature engineering procedure and serves with parameterizing traditional neural networks with many layers. Now, the main issue with deep neural networks is the *vanishing gradient* problem. As we add more layers to the network, it becomes more difficult to update the weights because the propagated signal becomes weaker and weaker. Since our network's weights might be off in the beginning because of random initialization, it can become almost impossible to parameterize a deep neural network with backpropagation. This is one of the reasons that different activation functions such as ReLU have been introduced to the community. I will cover a brief description of a few common ones in the next section.

2.5.0.1 Non-Linearity and Activation Functions

The neural networks can be decomposed into two functions. Firstly, a linear combination of the parameters, or weights and the input that gives $h(x) = \sum_{i=1}^{d} w_i * x_i + b$ and a step function that gives the final output $f(x) = sgn(h(x))$. Similarly, any function could be applied to h in order to get an output. These functions that are denoted by "a"s in figure 10 are called non-linearity or activation functions and are a key concept of allowing neural networks to become high capacity models. Without the use of a non-linearity, the model can only represent linear relationships. Although any function could be used as a non-linearity, given the use of gradient optimization, it is required that it be differentiable almost everywhere. The ReLU [46], the sigmoid and hyperbolic tangent (tanh) functions are the most common activation functions. They serve different purposes, ReLU is often used as a hidden layer activation, while sigmoid can be used to provide a value between 0 and 1, as a probability. The hyperbolic tangent (tanh) function used to be a popular hidden layer activation function, but it is mostly used now to force values between -1 and 1. Recently, ReLU activation function has become the default function for all neural networks implementation as it solved the gradient vanishing issue found in the sigmoid and hyperbolic tanget (tanh) functions. It also allows models to learn faster and perform better. The vanishing gradient issue happens when the error signal backpropagates through the network. If the network is deep enough, this error signal from output layer can get to zero on it's way back towards the input layer. This attenuation procedure is called vanishing gradient problem, the reason behind is that when we use chain rule to backpropagate the loss through the network, in every layer the error signal is multiplied by the derivative of the activation function and as the derivative of sigmoid and tanh are always near zero, this signal is getting weaker hence vanishing. The units in the linear region of the sigmoid does not attenuate the error signal that much but in general it can be problematic in deep networks. That's why ReLU is often used in neural network. The derivative of RelU is

zero if input is negative, otherwise equals to one. The other reason that makes ReLU a favorable choice, is it's sparsity effect. They result in highly sparse neural nets and sparsity means efficient and reliable performance.
In essence, we can think of deep learning as algorithms for automatic *feature engineering*, or we could simply consider them as feature extractors, which help us to expedite the learning process in neural networks with many layers.
In the context of image classification tasks, convolutional neural networks (ConvNets) are good examples of deep learning models. The convolutional layer acts as a feature extractor to the fully connected neural network. We intend to extract the valuable features from the images by using the convolution, and we aim at making the features scale and translation invariant by deploying the pooling layers.

2.6 From Deep Learning to Meta Learning

Given that, deep learning has become an essential element in various applications of technology over the past decade and we all agreed on a general recipe shared between training deep models. First, acquire a large number of samples for the given task, and then optimize a surrogate loss function with gradient descent methods as a learning paradigm. Such an extensible learning procedure, coupled with fast-growing available data has enabled researchers to develop better and more complex architectures for many tasks. However, the universality of this learning procedure comes with a cost, learning a new task is both data inefficient and computationally expensive.
A high-performing machine learning model often requires a large number of examples. However, if we compare this to the human learning process we see several contradictions. Humans learn new concepts more efficiently. Children who have seen dogs and kittens for a few times can detect them quickly. People who know how to play a game are likely to discover new games with little or even no demonstration. Is it possible to design a machine learning model that learns new concepts and skills as fast by using a few training samples? Conceptually, that's what meta-learning attempts to solve.

Chapter 3

Meta-learning

3.1 Introduction

The concept of meta-learning has been of interest in machine-learning for decades because of its favorable applications to many areas of research [14, 27, 30, 66, 54]. Meta-learning is a sub-field of machine learning where the automatic learning algorithms are applied on some meta-data. The goal is to find a flexible automatic learning algorithm that can be applied to solve learning problems. In other words, we aim to learn the learning algorithm itself. That's why meta-learning is also called *learning to learn* [7]. By using some meta-data from the learning process such as the performance of an algorithm, the properties of a learning problem or previously derived pattern from the data, we can learn, improve or change a learning algorithm for the given problem. The inspiration of meta-learning comes from evolutionary methods [66]. As also stated by Jurgen Schmidhuber's early work [30] and Yoshua Bengio et al.'s work [14], "Genetic evolution learns the learning procedure encoded in genes and executed in each individual's brain. In an open-ended hierarchical meta learning system using Genetic Programming, better evolutionary methods can be learned by meta evolution, which itself can be improved by meta meta evolution, etc." In 2001, A. Steven Younger and S. Hochreiter [66] showed that any recurrent network topology and its corresponding learning algorithm(s) is a potential meta-learning system thanks to gating mechanism of LSTM networks, they can store information for the longer periods of time needed to do meta-learning. They concluded with an LSTM based system that is able to develop a learning algorithm that could learn any quadratic function using only 35 training samples. There is an interesting statement presented by M. Andrychowicz1 et al. in 2016 [12] relating meta-learning and transfer learning. When we select a model for a prediction task, we specify a set of inductive biases about how the function should perform at points that have not observed. In this scenario, generalization corresponds to the capacity of the model to make decisions about the behavior of the target function with regard to unseen samples. In the transfer learning setting, the samples are problem instances themselves meaning generalization corresponds to the ability to transfer knowledge between different problems. “This reuse of problem structure is commonly known as transfer learning, and is often treated as a subject in its own right.” [12]. The meta-learning perspective can be seen as the problem of transfer learning as one of the generalization matters.

3.2 Applications of Meta-Learning

The perspective of the small-data regime opens many directions for applications that would not be practical under the current deep learning framework. For instance, in cancer diagnosis, the sensitivity of the data gathered from patients might be an issue when training large-scale models, unless one uses public datasets. If we can build some prior knowledge from different patients (e.g. demographic information about the population) without gathering all of the data, this would be a great help for retaining data privacy. The advantage of meta-learning, in this case, is that only the knowledge of the learning process on each patient will contribute to the overall prior, not their data.

3.3 Meta-Learning Vs. Multitask Learning, Transfer Learning, and Domain Adaptation

The ideas of knowledge sharing, adaptation and transfer learning have been previously studied in other related machine learning fields. Here I explain each of them briefly and their differences with meta-learning.
Meta-Learning, also known as *learning to learn* aims to design models supplied with two important criteria. First, it is to be able to learn new skills quickly. Secondly, it is important to adapt to new environments using only a few training examples. The adaptation process is a mini learning phase, and it happens during the test time with a limited exposure to the new task samples. Meta-learning considers the assumption of offline learning (i.e. all training samples are available at train time).

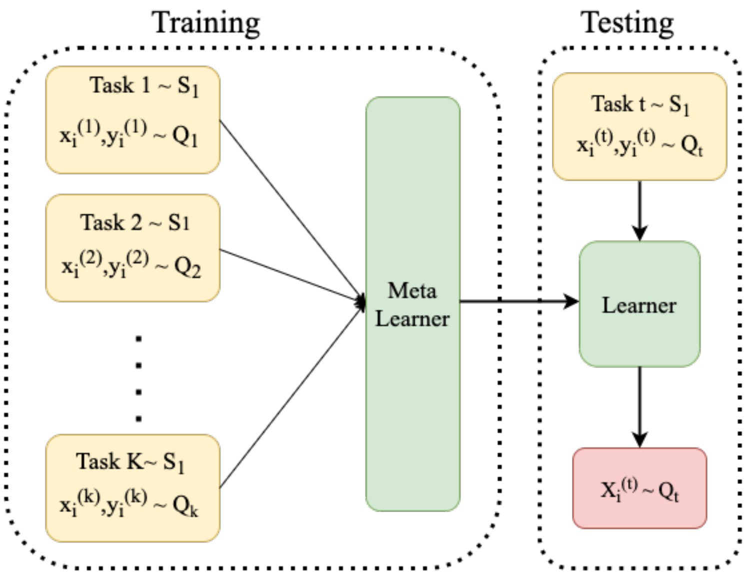

Training
Testing
Task 1 ~ S_1
$x_i^{(1)}, y_i^{(1)} \sim Q_1$
Task 2 ~ S_1
$x_i^{(2)}, y_i^{(2)} \sim Q_2$
Task K ~ S_1
$x_i^{(k)}, y_i^{(k)} \sim Q_k$
Meta Learner
Task t ~ S_1
$x_i^{(t)}, y_i^{(t)} \sim Q_t$
Learner
$X_i^{(t)} \sim Q_t$

Figure 11: Visual representation of meta-learning framework where training tasks are different but coming from the same source domain, aims at learning a model generalizes to new tasks which are still from the same source domain. The image is inspired from R. Aljundi.[11].

3.3.1 Multitask Learning

In practice, meta-learning is very close to multitask learning. The model is usually trained on related tasks in an offline mode. As shown in figure 12, the tasks used during test-time are coming from the same distribution as training tasks. Therefore, no adaptation is required at test-time as opposed to meta-learning. In multi-task learning, the parameters are shared between tasks for *better generalization*.

3.3.2 Transfer Learning

As illustrated in figure 13, the goal in transfer learning is to transfer knowledge gained from the source domain(Q_s) and source task(T_s) where sufficient training data is available, to the target domain(Q_t) and target task(T_t) where training data is limited, also, $Q_s \neq Q_t$, or $T_s \neq T_t$. An example of transfer learning is finetuning, where models are pre-trained on large tasks then used as initialization for tasks with limited training data. There is no task adaption required in transfer learning as the model is only evaluated on the samples coming from the target domain.

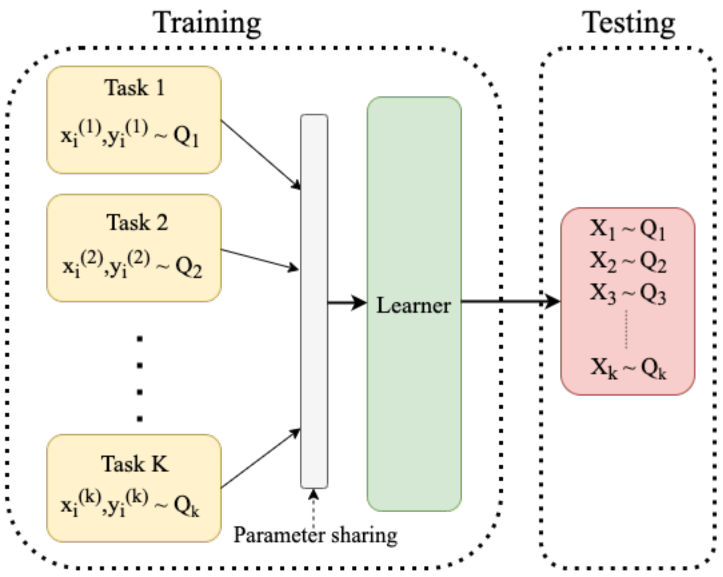


Figure 12: Visual representation of multitask learning framework where training and test tasks are coming from the same distribution. The image is inspired from [11].

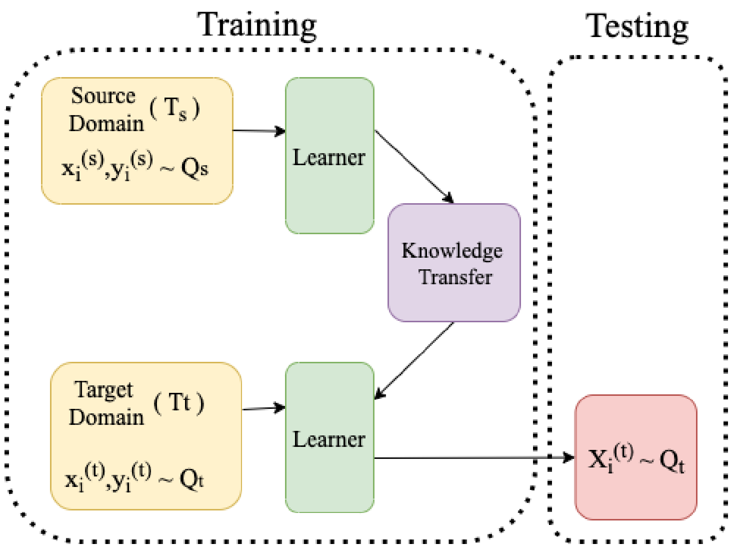


Figure 13: Visual representation of transfer learning setting. Inspired by [11].

3.3.3 Domain Adaptation

Domain adaptation is a type of transfer learning where the source and target tasks are the same but drawn from different input domains. The target domain examples are unlabeled and the aim is to train a model on the source domain in a way that it acquires high performance on the unlabeled target domain samples, as you can see in figure 14. In other words, it relaxes the classical machine learning assumption of having training and test data drawn from the same distribution [11], [17]. One of the approaches to domain adaptation is trying to learn a shared space to match the distributions of the source and target datasets. For instance, learning to detect different types of planes in black and white images (source distribution) then try to generalize this task to color images (target distribution).

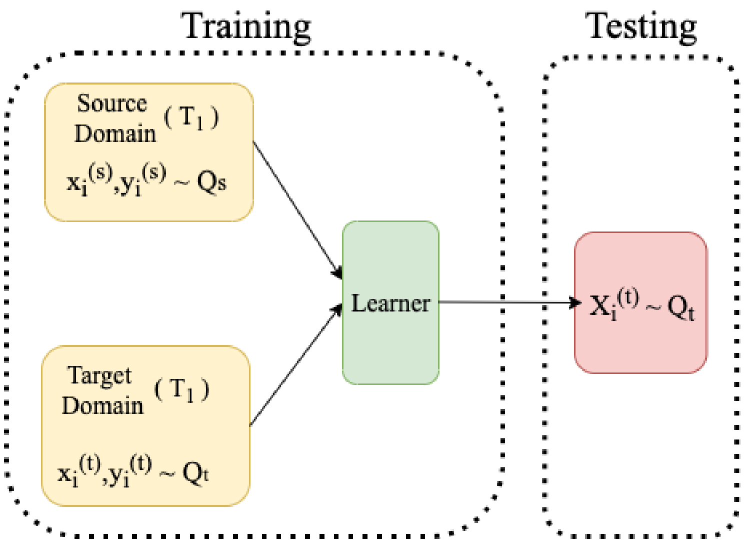


Figure 14: Visual representation of domain adaptation where source and target tasks are the same but drawn from different data distribution. Inspired by [11].

3.4 Meta-Learning Algorithms

According to the recent works, there are three main categories of algorithms in meta-learning, metric-based, gradient-based, and model-based. I briefly describe the metric-based and gradient-based methods in the following sections. Should the reader be interested in a thorough review of these approaches, the meta-learning blog [64] and the survey on few-shot learning [63] are great resources. In Section 3.5, I followed a probabilistic perspective to give a general introduction to meta-learning. Although, there are other meta-learning algorithms that do not necessarily require a distribution over model meta-parameters $p(\theta' \mid \mathcal{D}_{\text{meta}})$, we need one in algorithmic methods to infer $\theta'^{\star}$ from the meta-training set $\mathcal{D}_{\text{meta}}$. Similarly, the adaptation procedure does not need to be a maximum a posteriori. For instance, gradient-based meta-learning models use gradient descent for the adaptation.

3.4.1 Gradient-Based Meta-Learning

3.4.1.1 Model-Agnostic Meta-Learning for Fast Adaptation (MAML)

The inspiration behind MAML [22] is the fact that some underlying representations are more transferable than others. This inspiration is also shared between other multi-task learning algorithm and even transfer learning. For instance, there might be some internal features that are broadly applicable to several tasks, rather than a single task. In the MAML scenario, during training, the meta-learner parameterized by θ provides information to $\theta'_{\mathcal{T}_i}$ of the learner for task $\mathcal{T}_i$, and the learner returns error signals such as gradients to meta-learner to improve its parameters θ. Model parameters θ are also known as task-invariant parameters that is shared between all given tasks. MAML provides the same initialization for all tasks, while neglecting the task-specific information. This only suits a set of very similar tasks and does not work well when tasks are distinct. An illustration of MAML algorithm has been presented in figure 19. The equations for MAML are as follows. Below denotes task specific parameters θ' that is computed using one or more gradient descent updates on task $\mathcal{T}_i$:

$$\theta_i'^{(t)} = \theta^{(t)} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta(\mathcal{D}_i^{(train)})) \tag{9}$$

The model parameters are trained by optimizing for the performance of $f_{\theta' i}$ with respect to θ across tasks sampled from $p(\mathcal{T})$. The model parameters θ are updated as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}(\mathcal{D}_i^{(val)})) \tag{10}$$

There is also another variation of MAML that is called first-order MAML (fo-MAML) which is an approximation to the full-order MAML. MAML normally needs computing second-order gradients, which can be computationally expensive both in terms of time and memory. Therefore, an approximation of MAML is often used where the within-tasks gradients are ignored.

3.4.2 Metric-Based Meta-Learning

Metric-based methods are a sub-category of embedding-learning approaches where the model learns to find a good representation (also presented as a latent variable z) of the train and test samples by embedding them to smaller hypothesis space that is called H. According to figure 15, we also need a similarity measurement in this type of method that is used to calculate the similarity between $f(x^{train})$ and $g(x^{test})$ in the embedding space. Functions $f(.)$ and $g(.)$ are used to embed train

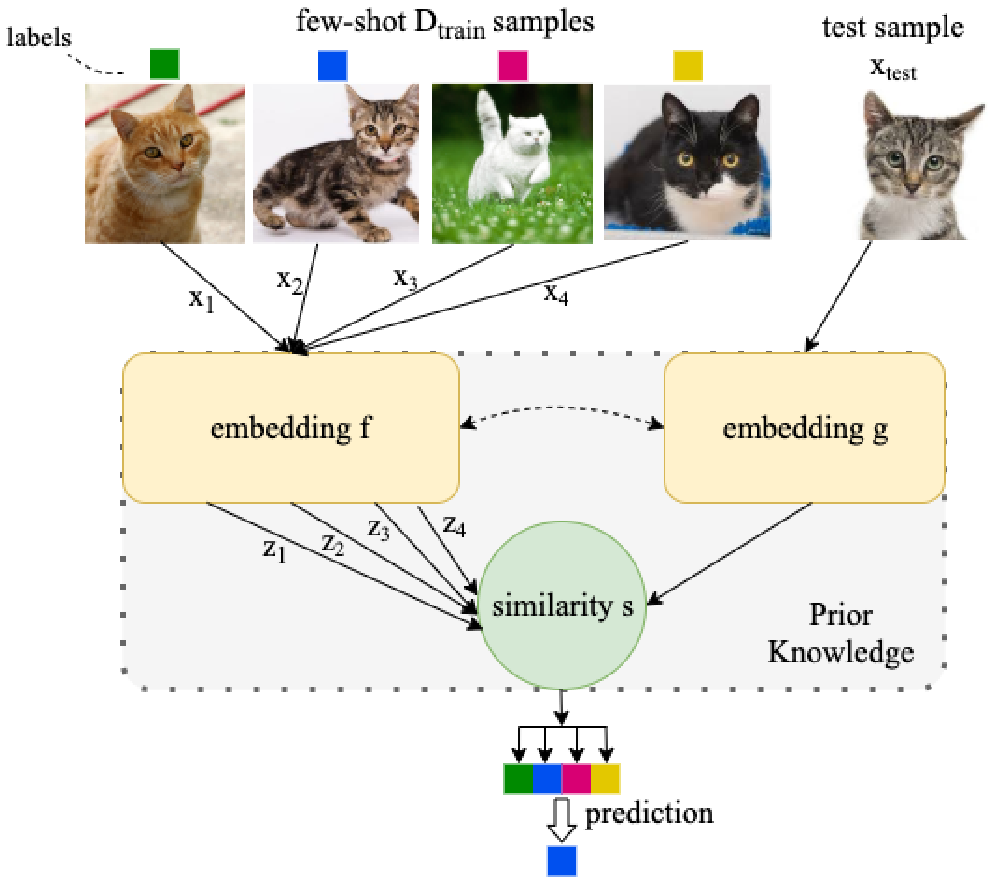


Figure 15: Visual representation of embedding learning where f and g are train and test embedding functions and s is the similarity metric. The image is illustrated by inspiration from [61].

and test samples respectively. Even though sometimes g can be different from f, they are in most cases the same, as in the cases that I am explaining here

3.4.2.1 Prototypical Networks

Prototypical networks (ProtoNet) [56] are types of embedding-learning methods where the model performs a comparison between $x^{(test)} \in D^{(test)}$ and the prototype of each class in $D^{(train)}$. A prototype is a mean of the embeddings of train samples drawn from the same class in dataset D_i^{train}. For instance, the prototype for class k in i_{th} dataset is defined as:

$$v_k^i = \frac{1}{|D_i^{train}|} \sum_{(x_i,y_i)_i^{train}} f_\theta(x^i) \tag{11}$$

The ProtoNet is using the same network to embed both train and test samples (i.e. f and g are the same), the reader can refer to figure 15 for a better comprehension. For a given test sample

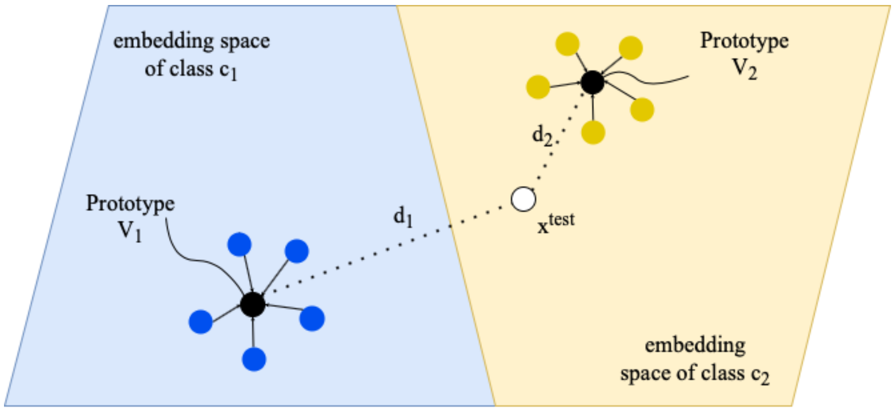
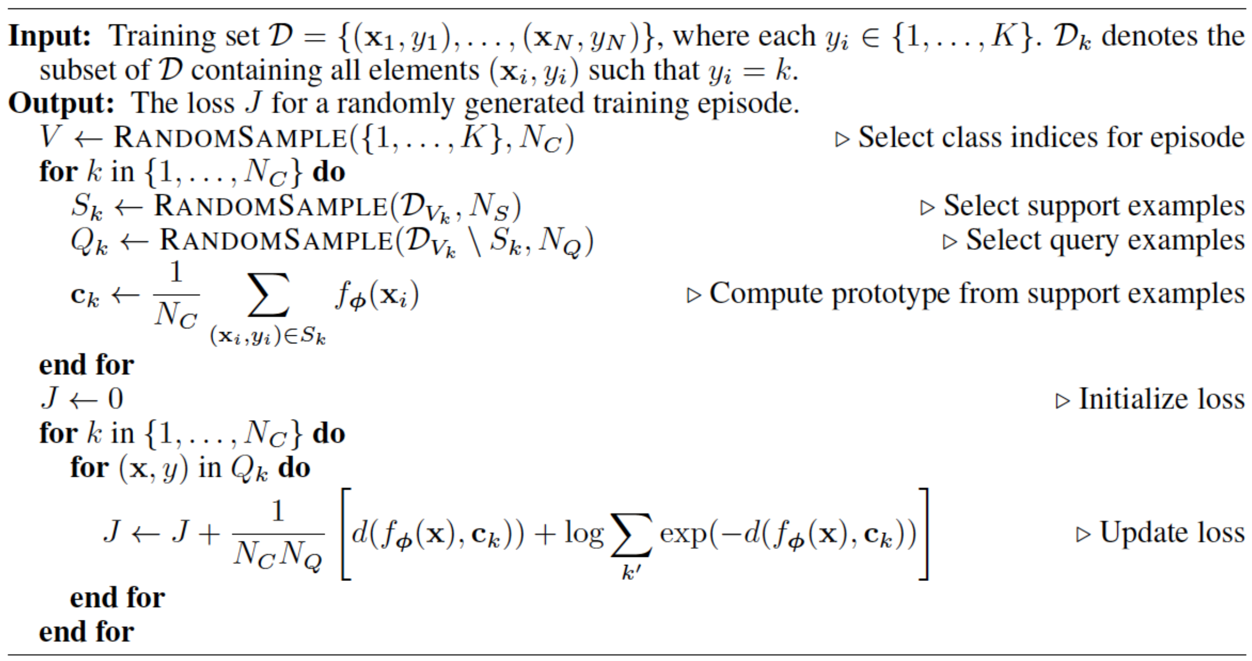

embedding space of class c_1
Prototype V_2
d_2
Prototype V_1
d_1
x^{test}
embedding space of class c_2

Figure 16: Prototypical networks in the few-shot learning framework. Few-shot prototypes v_k are computed as the mean of embedded train examples for each class. The distances from class 1 and 2, are d_1 and d_2 respectively, since $d_2 < d_1$, x^{test} is classified as class 2. The figure is illustrated by inspiration from [56].

x_i, a distribution over classes is formed which is a softmax over the inverse of distances between the test data embedding and prototypes. This can be written as:

$$p_\theta(y = c_k|x) = Softmax(-d(f_\theta(x), v_k)) = \frac{\exp(-d(f_\theta(x), v_k))}{\sum_{k'} \exp(-d(f_\theta(x), v_{k'}))} \tag{12}$$

The loss function is the negative log-likelihood: $L_\theta = -logP_\theta(y = c_k|x)$
In equation 12, d is the squared Euclidean distance. To classify a new test point, the class of the nearest prototype to the embedded test point is assigned.
Prototypical networks can be related to clustering as class means are prototypes when distances are computed with the squared euclidean distance. As visualized in figure 16, prototypical networks are based on the idea that there exists an embedding space in which points cluster around a single prototype representation for each class.
In figure 17, there are two notions of support and query sets which are similar to train and test sets in a supervised setting. In prototypical networks similar to matching networks, support set includes labeled samples that are used to learn an embedding to predict classes for the unlabeled points (the query set). In this algorithm, there is a single training loop and for each episode it computes a single loss and backpropagates that loss to update the embedding weights; the embedding model is a convnet and the Protonet model feeds both the 'support' samples (N_S) and 'query' samples (N_Q) through the convnet to compute the loss, returning that loss to the training loop.



```
Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the
  subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss $J$ for a randomly generated training episode.
  $V \leftarrow \text{RANDOMSAMPLE}(\{1, \ldots, K\}, N_C)$                          ▷ Select class indices for episode
  for $k$ in $\{1, \ldots, N_C\}$ do
    $S_k \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{V_k}, N_S)$                       ▷ Select support examples
    $Q_k \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{V_k} \setminus S_k, N_Q)$                  ▷ Select query examples
    $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$        ▷ Compute prototype from support examples
  end for
  $J \leftarrow 0$                                                        ▷ Initialize loss
  for $k$ in $\{1, \ldots, N_C\}$ do
    for $(\mathbf{x}, y)$ in $Q_k$ do
      $J \leftarrow J + \frac{1}{N_C N_Q} \left[ d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k)) \right]$    ▷ Update loss
    end for
  end for
```

Figure 17: Prototypical networks algorithm from J. Snell's work [56]. N is the number of examples in the training set, K is the number of classes in the training set. RANDOMSAMPLE(S,N) denotes a set of N elements chosen uniformly at random from support set S, without replacement.

3.4.2.2 Matching Networks

In matching networks, the model assigns $x^{test} \in \mathcal{D}^{test}$ to the most similar train point's embedding in Z where x^{train} and x^{test} are embedded differently using f and g. The classifier defines a probability distribution over output labels y given a test point x. Similar to other metric-based models, the classifier output is defined as a sum of labels of support samples weighted by attention kernel $a(x, x_i)$ which should be proportional to the similarity between x and x_i [64].

The output of the classifier is as follows where S denotes a support set $S = \{x_i, y_i\}_{i=1}^{k}$ for k-shot classification problem:

$$p(y|x, S) = \sum_{i=1}^{k} a(x, x_i) y_i, \tag{13}$$

The attention weight between two data points is the cosine similarity, cosine(.), between their embedding vectors, normalized by softmax: (x simply presents support samples and x_i is a given test point)

$$a(x, x_i) = Softmax(\cos(f(x)), g(x_i))) = \frac{\exp(\cos(f(x), g(x_i)))}{\sum_{j=1}^{k} \exp(\cos(f(x), g(x_j)))} \tag{14}$$

3.5 Mathematical Formulation

In standard supervised learning, a model is typically specified by its likelihood function $(y \mid x; \theta)$, where x is the input of the model, y is the output and θ is its parameters. The likelihood function $p(y \mid x; \theta)$ can be parametrized by a deep neural network. Learning such a model then refers to inferring the parameters θ from a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. To find the optimal parameters $\theta^\star$ under the maximum likelihood framework we can write:

$$\theta^\star = \underset{\theta}{\operatorname{argmax}} \log p(\theta \mid \mathcal{D}) = \underset{\theta}{\operatorname{argmax}} \log p(\mathcal{D} \mid \theta) + \log p(\theta) \tag{15}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y_i \mid x_i; \theta) + \log p(\theta), \tag{16}$$

where the prior term $\log p(\theta)$ acts as a *regularizer* on the overall objective. While learning with this kind of objective is reasonable in the large data regime, the role of the prior $p(\theta)$ becomes essential when $\mathcal{D}$ only comprises a small amount of data under a few-shot setting.

Conventionally, this prior distribution over network parameters, is initialized randomly (i.e. the weights are initialized randomly) and often turned to be far from the desired hypothesis space as you can see in figure 18.

3.5.1 Meta-Dataset and Meta-Parameters

According to the illustration in figure 18, in meta-learning we aim to learn inductive biases [60] via two update rules. First, a slow learning parameter that generalizes across tasks (meta-training phase). Second, a fast learning parameter that adapts quickly to a new task (adaptation phase). As I briefly introduced in the previous section, the initialized prior distribution over parameters is usually far from the desired hypothesis space and this is a critical issue in fast and efficient learning. To mitigate the effect of choosing a poor prior, we can incorporate additional data $\mathcal{D}_{\text{meta}}$ to compensate for the lack of data in $\mathcal{D}_i$. $\mathcal{D}_i$ is a part of $\mathcal{D}_{\text{meta}}$. The objective in Equation (15) now becomes

$$\theta^\star = \underset{\theta}{\operatorname{argmax}} \log p(\theta \mid \mathcal{D}_i, \mathcal{D}_{\text{meta}}). \tag{17}$$

In the context of meta-learning, this additional data $\mathcal{D}_{\text{meta}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$ is a collection of datasets, called the *meta-training set*. Each element $\mathcal{D}_i$ is a collection of input/output pairs $\mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_n^i, y_n^i)\}$. For instance, the meta-training set can contain samples of multiple

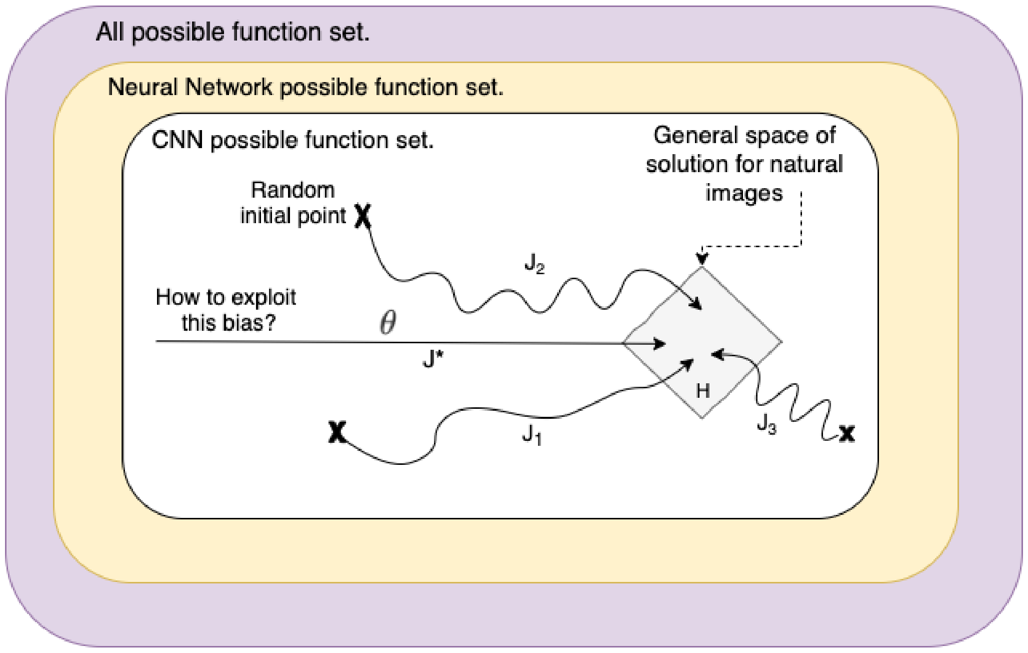

All possible function set.
Neural Network possible function set.
CNN possible function set.
General space of solution for natural images
Random initial point
How to exploit this bias?
θ
J_2
$J^\star$
H
J_1
J_3

Figure 18: Conceptual representation of inductive biases for a binary classification task on images. J_i in the figure denotes an optimization track that leads the model to the right hypothesis space.

tasks the agent has performed in the past (e.g. classifying different birds, then detecting planes, etc...), which can then be leveraged to learn a new task more efficiently (e.g. classifying different species of birds from a few examples).

However, storing and retrieving all information caught by the previous experiences of the model of $\mathcal{D}_{\text{meta}}$ every time the model is required to learn a new task might be computationally expensive (i.e. optimizing Equation (17)). Here, I am focusing on a specific form of meta-learning where the goal is to learn some prior over the parameters that will be useful for subsequent learning. Among the possible interpretations of meta-learning I'm focusing on a specific form, in the style of MAML, where the goal is to learn some prior over the parameters that will be useful for subsequent learning. The goal of meta-learning in this setting is to integrate all the experiences from $\mathcal{D}_{\text{meta}}$ into more versatile *meta-parameters* θ', by optimizing a second objective called the *meta-objective*

$$\theta'^{\star} = \underset{\theta'}{\operatorname{argmax}}\, p(\theta' \mid \mathcal{D}_{\text{meta}}). \tag{18}$$

For a better understanding of the notion of *model parameters and meta-parameters*, an illustration has been provided in figure 19 that will help one understand the subsequent ones better.
Similar to how the likelihood model $p(y \mid x; \theta)$ or the regularizer $p(\theta)$ in supervised learning is a

model selection task, a deep neural network, $p(\theta' \mid \mathcal{D}_{\text{meta}})$ is also a model selection paradigm. I will explain some of meta-learning algorithms in Section 3.4. Furthermore, if these meta-parameters contain all the essential information from $\mathcal{D}_{\text{meta}}$ to infer the model's parameters θ, then we can approximate the objective from Equation (17) with the following equation: (θ and θ' denote the model's parameters and meta-parameters respectively)

$$\begin{aligned}\log p(\theta \mid \mathcal{D}_i, \mathcal{D}_{\text{meta}}) &= \log \int p(\theta \mid \mathcal{D}_i, \theta') p(\theta' \mid \mathcal{D}_{\text{meta}}) \, \mathrm{d}\theta' &(19)\\ &\approx \log p(\theta \mid \mathcal{D}_i, \theta'^{\star}) + \log p(\theta'^{\star} \mid \mathcal{D}_{\text{meta}}). &(20)\end{aligned}$$

3.5.2 Adaptation to a New Task

Once the optimal meta-parameters $\theta'^{\star}$ have been found, they can be used as a substitute for $\mathcal{D}_{\text{meta}}$ in Equation (17) to learn a new task. In addition to $\theta'^{\star}$, a small dataset of input/output examples called $\mathcal{D}'_i$ is provided for a new task to learn the model parameters θ. The significant distinction between meta-learning and standard supervised learning is that the model needs an extra *adaptation* step before making predictions, even if $\theta'^{\star}$ is known. However, a model trained with standard supervised learning can be used to make predictions with $p(y \mid x; \theta^{\star})$ right away. To draw a parallel with maximum a posteriori from Equation (15), this adaptation phase can be written as:

$$\begin{aligned}\theta^{\star} = \underset{\theta}{\operatorname{argmax}} \log p(\theta \mid \mathcal{D}'_i, \mathcal{D}_{\text{meta}}) &\approx \underset{\theta}{\operatorname{argmax}} \log p(\theta \mid \mathcal{D}'_i, \theta'^{\star}) &(21)\\ &\approx \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y_i \mid x_i; \theta, \theta'^{\star}) + \log p(\theta \mid \theta'^{\star}). &(22)\end{aligned}$$

The meta-parameters $\theta'^{\star}$ integrated into the prior distribution $p(\theta \mid \theta'^{\star})$ over model parameters. Besides, the regularization term $\log p(\theta \mid \theta'^{\star})$ is not just a random initialization, but is now driven by the data from the meta-training set $\mathcal{D}_{\text{meta}}$.

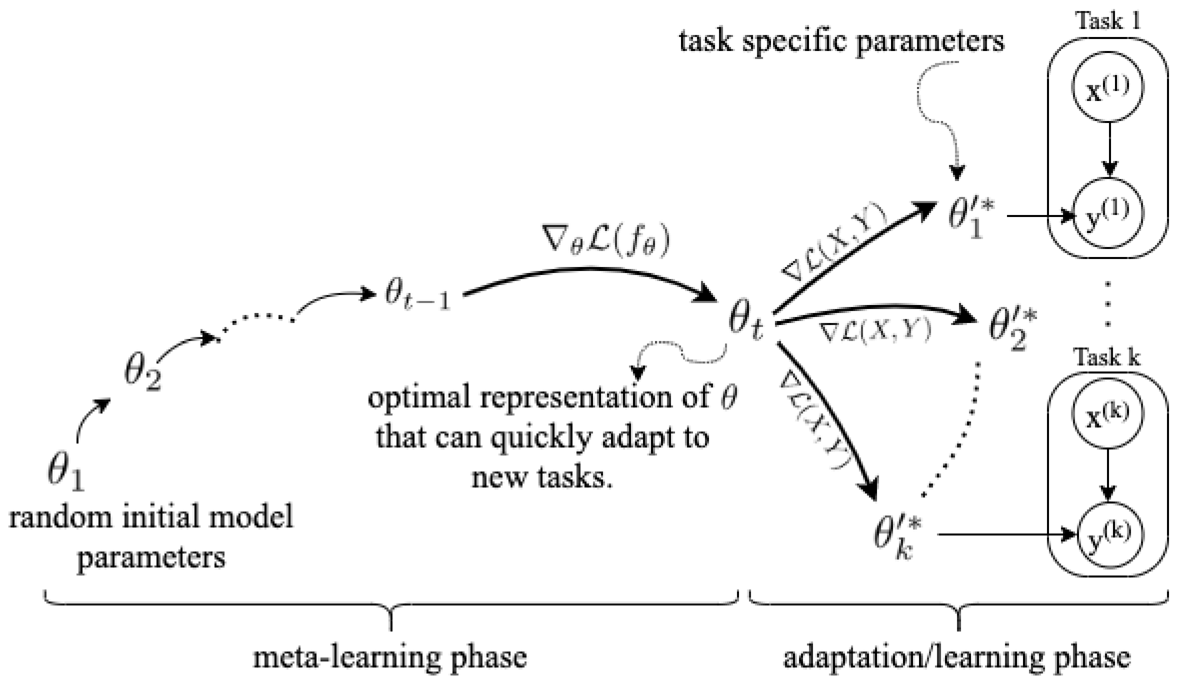
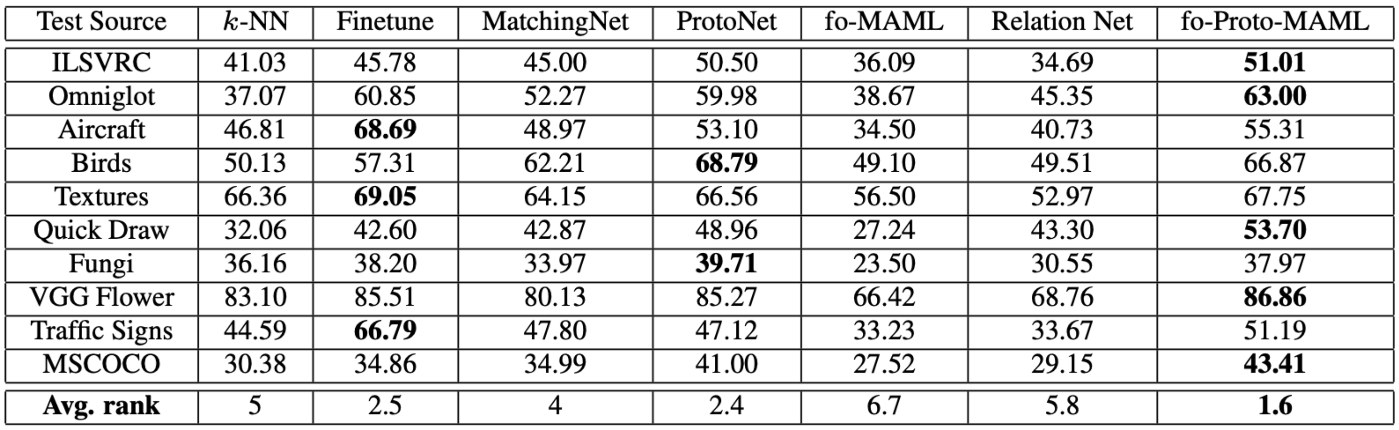

task specific parameters
Task 1
$x^{(1)}$
$y^{(1)}$
Task k
$x^{(k)}$
$y^{(k)}$
$\nabla_\theta \mathcal{L}(f_\theta)$
θ_1
θ_2
θ_{t-1}
θ_t
$\nabla \mathcal{L}(X, Y)$
$\theta_1'^*$
$\theta_2'^*$
$\theta_k'^*$
optimal representation of θ that can quickly adapt to new tasks.
random initial model parameters
meta-learning phase
adaptation/learning phase

Figure 19: Visual representation of meta-learning phase to learn a prior knowledge over hypothesis space through θ and adaption phase when the model learns a new task through θ'.

3.6 Meta-learner Selection Choice

In conclusion, the previously described methods are the most common approaches of meta-learning algorithms, as they showed a pretty good performance on few-shot classification tasks. This has been studied by Triantafillou, 2019 [59]. According to figure 20, ProtoNet has the second-highest rank between other methods. As explained by Snell et al., 2017 [56], the choice of distance metric is very important. As the euclidean distance used in Prototypical Networks considerably outperforms the more common one, i.e. cosine similarity used in Matching Networks. Prototypical Networks achieve state-of-the-art performance on several common benchmark tasks. Besides, they are simpler and more efficient than other meta-learning algorithms, making them an appealing approach for few-shot learning problems. MAML does not work well when tasks are distinct. It provides the same initialization for all tasks, while neglecting the task-specific information. This only suits a set of very homogeneous tasks, which is not the case in my experiments. From all the possible choices, I decided to go with ProtoNet as the meta-learner for my project. At the time I was reviewing the literature, ProtoNet was the most powerful meta-learner that had been proposed, however in October 2019, a new method has been designed by Triantafillou et al., called fo-Proto-MAML, which is also a powerful approach that combines the complementary strengths of Prototypical Networks and MAML. However, one will realize by looking at figure 20, that the improvement is not significant. Also, this method is computationally more expensive than ProtoNet.



Test Source	k-NN	Finetune	MatchingNet	ProtoNet	fo-MAML	Relation Net	fo-Proto-MAML
ILSVRC	41.03	45.78	45.00	50.50	36.09	34.69	**51.01**
Omniglot	37.07	60.85	52.27	59.98	38.67	45.35	**63.00**
Aircraft	46.81	**68.69**	48.97	53.10	34.50	40.73	55.31
Birds	50.13	57.31	62.21	**68.79**	49.10	49.51	66.87
Textures	66.36	**69.05**	64.15	66.56	56.50	52.97	67.75
Quick Draw	32.06	42.60	42.87	48.96	27.24	43.30	**53.70**
Fungi	36.16	38.20	33.97	**39.71**	23.50	30.55	37.97
VGG Flower	83.10	85.51	80.13	85.27	66.42	68.76	**86.86**
Traffic Signs	44.59	**66.79**	47.80	47.12	33.23	33.67	51.19
MSCOCO	30.38	34.86	34.99	41.00	27.52	29.15	**43.41**
Avg. rank	5	2.5	4	2.4	6.7	5.8	**1.6**

Figure 20: Few-shot classification results on Meta-dataset [59] using models trained on ILSVRC-2012. The image is credited to [59].

Chapter 4

Learning Cancer Phenotypes Using Meta-learning

4.1 Introduction

This chapter contributes to the main core of the project where we propose a meta-model that can learn new genomic tasks without having been seen by the classifier during train time. In this work, the goal is to study the generalizability of neural networks across different datasets in the context of genomics. This work also proposes a public Meta-Dataset that provides 174 defined clinical tasks that we named TCGA Meta-Dataset. Moreover, this includes a meta-dataloader that is available on the Github repository that can be found at [52].
The organization of this chapter is as follows. Section 4.2 presents the motivations of the work. In Section **??**, the meta-learning framework has been described. Section 4.3 states the dataset, cancer studies, and phenotypes. Sections 4.4 and 4.5 explain the task and related works. Sections 4.6 and 4.7 present the tool we developed to load the data. In section 4.8, I contribute to a description of the experimental and training setup, the machine learning and meta-learning models we implemented and applied in this work. Then, in section 4.9, I present the results on the generalizability of meta-learning on the genomic tasks and a comparison between meta-learning and classical machine learning baselines. In the end, I conclude the results and expand this to potential future directions.

A part of this work has been done as a collaboration with Joseph Paul Cohen, Tobias Würfl and Tristan Deleu at Mila institute where I also worked with a biologist Genevive Boucher [53].

4.2 Motivations

One of the key concerns that makes the meta-learning more demanding comes from a perspective of *data efficiency.* Meta-learning algorithms are mostly applied to *few-shot learning* problems (i.e. problems with only a few training samples). One can inquire about the importance of this "small-data" regime problems versus the more common "big-data" scenario which can even perform better in several cases. Here are the 3 important reasons that justify the importance of this small-data setting:

1. **Data Availability**: Even though, the amount of data gathered using smart sensors keeps increasing, as a direct consequence, the amount of available data to use does not necessarily increase at a similar rate. The amount of available data depends on the application. For example, there is not enough number of samples for some rare diseases and cancers. Therefore, only the few that have access to large databases can afford to develop expert systems under the current deep learning paradigm.

2. **Data Privacy**: There are many rules and regulations for data acquisition depending on the application and the field that data is associated with. More specifically, in hospitals and clinics, data protection rules require the explicit consent of individuals to provide the information to external parties. The local legislation restricts the effective amount of available data, and this is a high-ranking limitation for training deep learning models.

3. **Notion of General Intelligence (as opposed to specialists)**: Deep neural networks can already solve several hard tasks and can achieve superhuman performance. Recent progress in the field of Machine and Reinforcement Learning have developed agents that outperform human performance in playing games such as Go. This is an example of an expert system. However, we might be interested in more general intelligent systems capable of performing multiple tasks and adapt to new situations, not necessarily at a superhuman level. We are particularly interested in models that are able to learn tasks by using a few samples, and this reduces the cost of data acquisition and the limitation of the expert systems.

4.3 Dataset

For this work, we used a public dataset called The Cancer Genome Atlas (TCGA). This dataset includes clinicopathologic annotation data along with multi-platform molecular profiles of more than 11,000 human tumors across 38 different cancer types [41]. TCGA provided more than 2.5

petabytes of genomic, epigenomic, transcriptomic, and proteomic data. This data-hub is publicly available for anyone in the research community and has already enabled many improvements for better diagnosis and treatments as well as preventing cancer severity. For this project, we leverage the gene expression profile (20530 genes) called IlluminaHiSeq. The gene expression profile was measured experimentally using the Illumina HiSeq 2000 RNA Sequencing platform by the University of North Carolina TCGA genome characterization center. There is also the TGCA pan-cancer analysis that examines the similarities and differences among the genomic and cellular alterations found in the first dozen tumor types to be profiled by TCGA. Genomic alterations in diverse cell types at different sites in the body give rise to hundreds of different forms of cancer, and these changes result in tumors with different biology, pathology and treatment strategies which are beginning to be characterized.
We used 35 different cancer types to predict particular genomics and clinical variables which I explain in the following.

4.3.1 Types of Cancers Selected for Study

In this section, I explain the cancer studies, their full name and a brief description of each. If one needs more detailed information about each cancer type, the articles at [2] and [10] are good resources. These cancer studies have been selected by TCGA program according to specific criteria. I also decided to stick to these cancer types as they have been globally accepted and approved by National Cancer Institute (NCI). One of the important criteria that has been important for the community is the poor prognosis of these cancers. This can help doctors and physicians for a better and more accurate diagnosis. The other reason is the overall public health impact that has been reported from these cancer types. Also, the availability of samples from patients as well as the quality and quantity of them are of an essential matter.

- **ACC (TCGA Adrenocortical Cancer)**: This is the case when cancer cells form in the outer layer of the adrenal gland.
- **BLCA (TCGA Bladder Cancer)**: Bladder cancer typically affects men rather than women and it often occurs in older adults. This type of cancer most frequently begins in the urothelial cells that line the inside of the bladder.
- **BRCA (TCGA Breast Cancer)**: Breast cancer starts when a group of cancer cells grow into breast tissues and invade nearby ones.

- **CESC (TCGA Cervical Cancer)**: This type of cancer occurs in the cervix when cervical cells grow abnormally and destroy other tissues and organs in the body. However, the progress of cervical cancer is usually slow and it allows for early detection and treatment. The average age of women who are diagnosed with cervical cancer is in the mid-50s.

- **CHOL (TCGA Bile Duct Cancer)**: For a better understanding of this cancer, one should know the role of bile ducts. They move a fluid called bile from the liver and gallbladder to the small intestine where it allows a better digestion of the fats in foods. Sometimes the cells in the bile ducts change and grow abnormally. These changes can cause benign conditions or lead to bile duct cancer. Most often, bile duct cancer begins in the cells of the inner layer of the bile duct.

- **COAD (TCGA Colon Cancer)**: Colon cancer often occurs in older adults. It usually starts as a small, noncancerous mass of cells called polyps that form on the inside of the colon. Some of these polyps can turn into colon cancers over time.

- **COADREAD (TCGA Colon and Rectal Cancer)**: Colon and Rectum are parts of the digestive system and the large intestine. This type of cancer is also known as colorectal cancer as these organs have the same type of tissues and there is no clear border between them.

- **DLBC (TCGA Large B-cell Lymphoma Cancer)**: Lymphoma is a complicated cancer that begins in lymphocyte cells of the immune system. These cells are white blood cells that are responsible for fighting infections.

- **ESCA (TCGA Esophageal Cancer)**: This cancer occurs in the esophagus which is a long, hollow tube that moves the food from the back of the throat to the stomach to be digested.

- **GBM (TCGA Glioblastoma Cancer)**: Glioblastoma is an aggressive type of cancer that usually affects the brain cells or spinal cord. Glioblastoma is made of cells called astrocytes that support nerve cells.

- **GBMLGG (TCGA Lower-Grade Glioma & Glioblastoma)**: Glioma is a type of cancer that develops in the glial cells of the brain. Glial cells support the brain nerve cells and provide them with the required nutrients. Tumors are classified into grades I, II, III, and IV based on standards established by the World Health Organization. TCGA studied lower-grade glioma, which consists of grades II and III. GBM or glioblastoma is classified as grade IV, which is the most aggressive one.

- **HNSC (TCGA Head and Neck Cancer)**: Cancers of the head and neck are categorized by the area of the head or neck in which they occur. They usually start in the squamous cells that have moist and mucosal surfaces inside the head and neck such as inside the mouth, the nose, and the throat.

- **KICH (TCGA Kidney Chromophobe)**: Chromophobe renal cell carcinoma is a rare type of kidney cancer that forms in the cells lining the small tubules in the kidney. These small tubules help filter waste from the blood to make urine.

- **KIRP (TCGA Kidney Papillary Cell Carcinoma)**: This is the second most common type of kidney cancer and usually develops from inside the kidney's tubules.

- **LAML (TCGA Acute Myeloid Leukemia)**: LAML is one of the most common acute leukemia cancer in North America. The average age of LAML patients is 67. LAML is a cancer of the blood and bone marrow. It's quite dangerous and should be treated quickly as it can result in death within months.

- **LGG (TCGA Lower Grade Glioma)**: As I have already explained in the GBMLGG type of cancer, lower-grade glioma is of grade II and III types of tumors in glial cells that support the brain nerve cells.

- **LIHC (TCGA Liver Cancer)**: Liver cancer occurs in the cells of the liver. The liver plays an important role in the body by cleaning the blood and discarding harmful materials.

- **LUNG (TCGA Lung Cancer)**: There are two types of lung cancers. Non-small cell lung cancer and small cell lung cancer are based on the type of cell in which cancer starts.

- **LUAD (TCGA Lung Adenocarcinoma)**: Non-small cell lung cancer usually starts in glandular cells on the outer part of the lung. This type of cancer is called adenocarcinoma.

- **LUSC (TCGA Lung Squamous Cell Carcinoma)**: Non-small cell lung cancer can also start in flat, thin cells called squamous cells. This type of cancer is called squamous cell carcinoma of the lung.

- **MESO (TCGA Mesothelioma)**: Malignant mesothelioma is a type of cancer that starts in the thin layer of tissue that covers the majority of the internal organs (mesothelium).

- **OV (TCGA Ovarian Cancer)**: Ovarian cancer is a type of cancer that occurs in the ovaries. The female reproductive system contains two ovaries, one on each side of the uterus. The ovaries make hormones estrogen and progesterone as well as ova.
- **PAAD (TCGA Pancreatic Cancer)**: Pancreatic cancer begins in the tissues of your pancreas. The pancreas produces enzymes that help digestion and makes hormones that help manage blood sugar.
- **PCPG (TCGA Pheochromocytoma & Paraganglioma)**: These are rare tumors that come from the same type of tissue. Paraganglioma starts in nerve tissues in the adrenal glands and near certain blood vessels and nerves. Paragangliomas that form in the adrenal glands are called pheochromocytomas.
- **PRAD (TCGA Prostate Cancer)**: Prostate cancer is a type of cancer that occurs in the prostate. This is one of the most common cancers among men.
- **READ (TCGA Rectal Cancer)**: This cancer occurs in the rectum that is the last part of the large intestine.
- **SARC (TCGA Sarcoma)**: Soft tissue sarcoma is a rare type of cancer that occurs in the tissues that connect, support and surround other body structures. This can contain nerves, muscle, blood vessels, fat, tendons and the lining of the joints.
- **SKCM (TCGA Melanoma)**: Melanoma, the most serious type of skin cancer, develops in the cells called melanocytes that make melanin which is the pigment gives the skin its color.
- **STAD (TCGA Stomach Cancer)**: Stomach cancer usually begins in the mucus-producing cells that line the stomach.
- **TCGT (TCGA Testicular Cancer)**: Testicular cancer occurs in the testicles. The testicles release male sex hormones and sperm for reproduction.
- **THCA (TCGA Thyroid Cancer)**: Thyroid cancer occurs in the cells of the thyroid. The thyroid makes hormones that regulate blood pressure, heart rate, body temperature, and weight.
- **THYM (TCGA Thymoma)**: Thymoma and thymic carcinoma are diseases in which cancer cells form on the outside surface of the thymus. The thymus is an organ in the neck that makes T-cells for the immune system.

- **UCEC (TCGA Endometrioid Cancer)**: Endometrial cancer occurs in the cells that form the inner layer of the uterus (endometrium) and is sometimes called uterine cancer. Other types of cancer can form in the uterus, including uterine sarcoma, but they are much less common than endometrial cancer.

- **UCS (TCGA Uterine Carcinosarcoma)**: This type of cancer starts in the inner layer of the tissue lining the uterus, while sarcoma begins in the outer layer of muscle of the uterus.

- **UVM (TCGA Ocular Melanomas)**: As explained in TCGA Melanoma, this is a type of cancer that develops in the cells that produce melanin. The eyes also have cells that make melanin and can develop melanoma. Eye melanoma is also called ocular melanoma.

Now that we have a more clear idea of different types of cancer, we can proceed with the definition of the phenotypes that I studied in my experiments.

4.3.2 Genomics and Clinical Attributes Selected for Study

After understanding the cancer studies in TCGA, we need to know the phenotype that has been used for prediction corresponding to a cancer study. There are 44 different clinical and genomic variables (phenotypes) in the defined tasks. The tasks cover a range of clinical problems including modeling:

- **Event**: This is a boolean value that shows whether the patient had a new tumor event (e.g. metastatic, recurrence or new primary tumor) after their initial treatment for the tumor. This phenotype is also named as "new tumor event type".

- **Gender**: A binary variable that gets either male or female.

- **Tumor Tissue Site**: A clinical site that collects and provides patient samples and clinical metadata for research use.

- **White Cell Count**: The number of white cells in the given tissue sample.

- **Histological Type**: Usually cancers are classified in two ways. First by the type of tissue in which cancer originates (histological type) and by primary site, or the location in the body where cancer first developed. For example, in Esophageal cancer, there are 2 categories of histological type, Esophagus Squamous Cell Carcinoma and Esophagus Adenocarcinoma. The first type often forms in the thin, flat cells lining the inside of the esophagus and the second usually occurs in mucus-secreting glands. This variable has a different variety of classes depending on the task.

- **OCT Embedded**: A boolean value indicating whether the Optimal Cutting Temperature compound (OCT) is used to embed tissue samples before frozen sectioning on a microtome-cryostat.

- **Metastasis**: Metastasis is the major cause of breast cancer-associated deaths. Metastatic cancer (also called stage IV breast cancer) is a breast cancer that has spread to another part of the body, most commonly the liver, brain, bones, or lungs. In this dataset, metastasis is a binary feature and includes the possible labels of M0 and M1. M0 means cancer has not spread to other parts of the body and M1 means cancer has already spread to other parts of the body.

- **Mental Status Changes**: This phenotype is presented in LGG and GBMLGG. It indicates if the patient/participant presented with mental status changes prior to the diagnosis of cancer. It is a binary variable with values of 'Yes' and 'No'.

- **Family History of Cancer**: This phenotype shows whether the patient/participant has a first degree relative (parents, siblings, children) with a history of cancer. This is also a binary feature with 'Yes' and 'No' values.

- **Evidence of Active Hepatic Inflammation in Adjacent Tissue**: This is a phenotype of liver cancer and indicates whether the patient had evidence of active hepatic inflamed adjacent tissues. It's a 3-class classification problem with values including Mild, Severe, and None.

- **Colon Polyps Present**: Indicates if polyps were present in the colon, surgically and/or pathologically, at the time of tissue collection for the tumor submitted to TCGA. It's a binary feature with the values of 'Yes' and 'No'.

- **Lymphovascular Invasion Present**: Indicates whether large vessel (vascular) invasion and/or small, thin-walled (lymphatic) invasion was detected. This is also a binary classification problem with 'Yes' and 'No' values.

- **Anatomic Neoplasm Subdivision**: This is a phenotype measured in Ovarian cancer. Using the patient's pathology/laboratory report, it shows the anatomic site of the tumor used for TCGA. It is a 3-class classification problem with the values right, left, and bilateral.

- **Pan-Cancer DNA Methylation**: As I explained in section 4.3, Pan-cancer analysis is a project to identify the similarities and differences between genomic unusual modifications across different tumor types. DNA Methylation is an epigenetic alternation that can have a significant impact on changing the healthy regulation of gene expression to a disease pattern. Epigenetic modifications are a mechanism the cell uses to turn gene expression on/off without changing the DNA sequence itself. The methylation of genomic DNA in malignant cells is different across various cancer types. In my experiments, this phenotype tested on Head and Neck Cancer that is a 4-class classification problem from clusters 1 to 4. The prediction of this attribute can help in treatment of cancer.

- **Pan-Cancer MicroRNA Expression**: This is a 5-class classification attribute from clusters 1 to 5. miRNA expression is dysregulated in human cancer through various mechanisms. They have been grouped to different clusters to indicate the cause of this dysregulation which can be a result of amplifications or deletions of miRNA genes, abnormal transcriptional control of miRNAs, etc.

and a few others that are described in detail in [9].

4.4 Task Definition

A collection of 174 genomic tasks is proposed in this work. Tasks are combinations of a phenotype to predict and a cancer study corresponds to that phenotype. They are classification problems and the input space is gene-expression data with 20530 genes. Each task denotes an independent dataset. The purpose of each dataset to predict a phenotype of a particular cancer type. The purpose can also be inferred by the name of the dataset. For instance, one of the datasets is named 'gender-LAML' means we predict 'gender' for 'Acute Myeloid Leukemia' samples. Tasks have been trained and evaluated under a few-shot learning regime. I evaluated the performance of each task using the prototypical model, k-nearest neighbors, logistic regression and a neural network that will predict across cancer studies. Should you need a background on any of these models, please read sections 2.4 and 3.4.2.1. This framework is an initiative to develop Meta-Learning techniques that make use of the interrelated tasks of gene-expression data to learn effective classification models despite the small sample size of each individual task. There are 44 phenotypes and 25 cancer studies.

4.5 Related Works

A recent paper in Meta-learning literature which is closely related to this work is Meta-Dataset by Triantafillou et al., 2019 [59] that offers an environment for training and evaluating meta-learners for few-shot image classification by evaluating various baselines and meta-learners on this Meta-Dataset. This benchmark includes 10 datasets from various image classification tasks including ILSVRC-2012 (ImageNet) [51], Omniglot [35], Aircraft [43], MSCOCO [38], CUB-200-2011 (Birds) [62], Describable Textures [16], Quick Draw [29], Fungi [55], VGG Flower [47] and Traffic Signs [28]. Despite having a small number of tasks, developing this benchmark opens the door to the use of multiple datasets for few-shot learning by leveraging the use of samples from different datasets to construct the prior knowledge from multiple data sources and allows researchers to evaluate more challenging generalization problem.

In the literature of genomics applications, most of the works are developing algorithms and techniques that can solve only one particular clinical task at a time. To the best of our knowledge, there is no particular work that is doing general genomics and clinical feature prediction using gene expression data; although one of the works that has a similar initiative is "Multitask learning and benchmarking with clinical time-series data" by Harutyunyan, 2019 [26]. In this work, only 4 different tasks from MIMIC-III database have been defined. There are three main differences between our work and the one mentioned. Firstly, we defined a higher variety of tasks. Secondly, the dataset that our tasks are originated from is TCGA. Finally, we consider a few-shot setting for each task.

4.6 Torchmeta Library for Gene Expression Data

The existence of standardized benchmarks has played a crucial role in the progress we have observed over the past few years in meta-learning research. They make the evaluation of existing methods easier and fair, which in turn serves as a reference point for the development of new meta-learning algorithms; this creates a virtuous circle, rooted into these well-defined collections of tasks. Unlike existing datasets in supervised learning, such as MNIST [36] or ImageNet [51], the benchmarks in meta-learning accommodate datasets of datasets. This adds a layer of complexity to the data pipeline, to the extent that a majority of meta-learning projects implement their specific data-loading component adapted to their method. The lack of a standard at the input level creates variances in the mechanisms surrounding each meta-learning algorithm, which makes a fair comparison more challenging [19].

4.7 Data-Loaders for Few-Shot Learning

While there has been an extreme growth in Machine Learning research for Genomics, several barriers have slowed the potential progress. One particular barrier is the absence of global publicly available benchmark datasets in the field of gene expression works to evaluate the performance of models, facilitate reproducibility and allow the community to focus on a set of challenges. There are consistent public benchmarks in the field of computer vision for image classification tasks. For example, ImageNet and Large Scale Visual Recognition Challenge (ILSVRC) [51] allowed a significant improvement in the classification acccuracy from 25% in 2011 to 95% in 2017 for benchmark tasks. In contrast, practical progress in genomics applications has been difficult to measure due to variability and inconsistency in data sets and task definitions. Although there is a fairly well-known challenge called DREAM that can be found at [3], it has not imposed itself as a standard challenge such as ImageNet.

The meta-dataset we provide in this work offers a collection of datasets corresponding to few-shot classification problems. The interface was created to support modularity between datasets, to simplify the process of evaluation on benchmarks; the data-loader developed for this meta-dataset has also been included in Torchmeta library developed by Deleu et al, 2019 [19]. The data-loaders from Torchmeta are fully compatible with standard data components of PyTorch, such as `Dataset` and `DataLoader`. Before going into the details of the meta-dataloader, I first briefly recall the problem setting before going into more detail.

To balance the lack of data inherent in few-shot learning, meta-learning algorithms acquire some prior knowledge from a collection of datasets $\mathcal{D}_{\text{meta}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, called the *meta-training set*. In the context of few-shot learning, each element $\mathcal{D}_i$ contains only a few inputs/output pairs (x, y), where y depends on the nature of the problem. For instance, these datasets can contain examples of different tasks performed in the past. Torchmeta offers a solution to automate the creation of each dataset $\mathcal{D}_i$, with a minimal amount of problem-specific components.

We developed a meta-dataloader for TCGA that allows loading a batch of tasks that can be iterated over to generate datasets. The possibility of loading only one clinical task by specifying a phenotype and a cancer study is also provided. The file `TCGA.py` includes the detail of the dataset and meta-dataloader that can be found at the provided GitHub repository [52]. You can see an example below in which the variable 'datasets' is the collection of all tasks and 'task' is one of these datasets named (histological type, LGG). TCGA meta-dataloader is a part of the Torchmeta Library [19].

```
# Helper function, equivalent to Section 4.7
datasets = meta_dataloader.TCGA.TCGAMeta(download="True", min_samples_per_class=10)

task = meta_dataloader.TCGA.TCGATask((‘histological_type’, ‘LGG’))

print(task.id)# output: (‘histological_type’, ‘LGG’)
print(task._samples.shape)# output: (529, 20530)
print(collections.Counter(task._labels))# output: Counter(2: 198, 0: 197, 1: 134)
```

4.8 Experimental Setup and Technical Details

We implement four different supervised models to evaluate the performance of each task using regression and neural network baselines. For logistic regression, we use a 1-layer neural network (zero-hidden layer neural net with Softmax output) from PyTorch [48] and LBFGS function [40] for optimization. We also apply a k-nearest neighbors classifier using the sklearn library and search over possible values of K in the range of 2 to 10. K equals 4 gives a better performance when compared to other possible values. We also use the euclidean distance as a distance metric in KNN. For a neural network, we tweak the network using different number of layers and hidden layer size. The results of our hyperparameter search is given in Table 2. The learning rate of 1e-4 and the network with 2 hidden layers with 128 and 64 hidden nodes respectively have been chosen for our further analysis. Also, we use fully-connected layers and ReLU as the activation function. In our experiments, batch size equals 32 and weight decay is 0.0 (no regularization effect). We use Adam [31] for optimization in the neural network. We run the network for 250 epochs for each task and the performance is averaged over 10 different trials (by changing seed from 0 to 9). All tasks have 150 samples in total. We split 150 samples between train and test sets and valid set if needed for early stopping. Train and test sets contain 50 and 100 samples respectively. The train set has less number of examples to serve the few-shot learning regime.

Number of Layers	1	Hidden Layer Size	[512]	67.55
			[256]	67.25
			[128]	**68.07**
			[64]	67.70
			[32]	65.48
	2	Hidden Layer Size	[512, 256]	67.92
			[256, 128]	67.70
			[128, 64]	**68.14**
	3	Hidden Layer Size	**[128, 64,32]**	**68.24**
			[256, 128, 64]	66.87
			[512, 128, 64]	67.96
			[256, 128, 32]	67.36
	4	Hidden Layer Size	[512, 256, 128, 64]	62.833
			[256, 128, 64, 32]	**62.834**
			[128, 64, 32, 16]	62.833
Learning Rate	Range of Values		1e-5	67.55
			1e-4	**68.33**
			1e-3	68.0
			1e-2	63.33

Table 2: The results of hyperparameter search, learning rate sets to 1e-3 when tweaking the architecture. Also, when tuning the learning rate, the neural network has 2 layers with 128 and 64 hidden nodes in each layer respectively.

4.8.1 Evaluation Strategy

We consider accuracy over test tasks as the evaluation metric; however, it may not be the most relevant metric in an applied setting given a new dataset. In meta-learning algorithms during the training, the model maximized its averaged performance over a batch of tasks. Whereas, in practice, we are concerend with the behaviour of the model on an individual task during test time. This contradiction between the desired objective and the meta-learning goal can make unexpected outcomes where the performance of the model may decrease on some tasks during training [18] while there is no improvement on the target tasks during test time. This is highly critical in healthcare applications, where guarantees on the learning phase are necessary.

4.8.2 Training Prototypical Networks

I developed a prototypical network from scratch using Pytoch and tested it in meta-learning framework as described in MAML [22]. I evaluated its performance on the Omniglot dataset to verify my model according to the original paper [56]. As presented in Algorithm 2, to train a ProtoNet in MAML setting, we define two for-loops to perform two optimization processes, an outer loop that is over batches of multiple tasks and the 'meta epoch' variable (M) specifies the number of iterations. The outer loop aims to minimize the average loss across batches of tasks. The inner loop iterates over tasks in one batch and calculates the task-specific loss. The 'meta batch size' variable (MB) indicates the number of tasks per batch. The variable 'number of shots' (NS) specifies the number of samples the ProtoNet employs to compute the protos (i.e. class means). However, during test time, we do not use the number of shots and instead employ all 50 samples for training the model and all 100 samples of test set for the adaptation phase. Figure 21 presents the training pipeline.

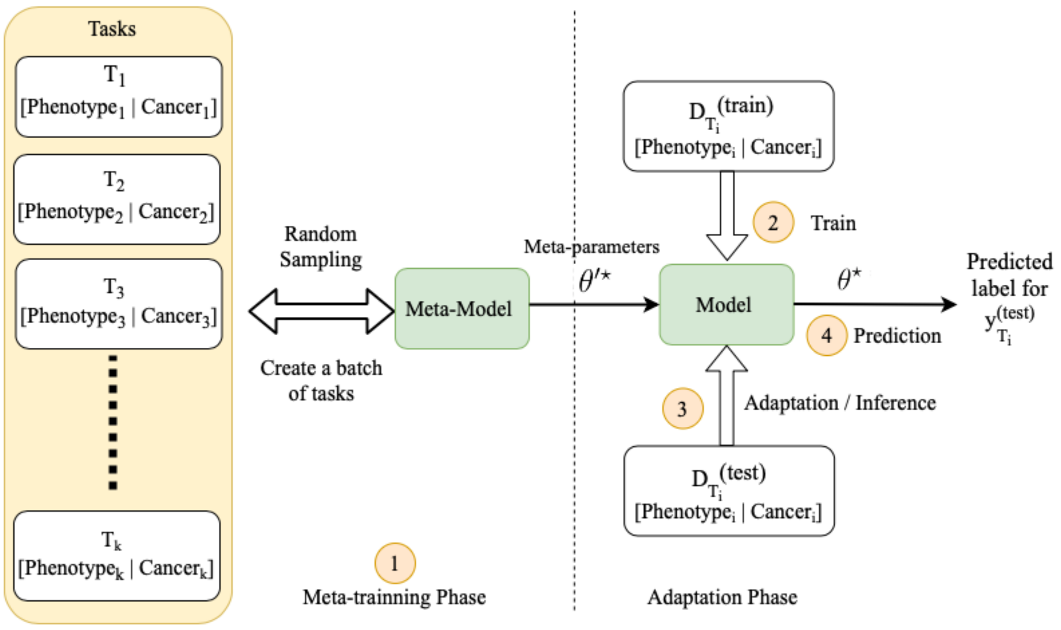

Tasks
T1 [Phenotype1 | Cancer1]
T2 [Phenotype2 | Cancer2]
T3 [Phenotype3 | Cancer3]
Tk [Phenotypek | Cancerk]
Random Sampling
Create a batch of tasks
Meta-Model
Meta-parameters
θ′*
Model
θ*
Predicted label for y(test)Ti
D(train)Ti [Phenotypei | Canceri]
2 Train
4 Prediction
3 Adaptation / Inference
D(test)Ti [Phenotypei | Canceri]
1 Meta-trainning Phase
Adaptation Phase

Figure 21: A visual representation of the training pipeline.

Algorithm 2 An overview of meta-training procedure

```
1:  procedure META-TRAINING($\{\mathcal{T}_1, ..., \mathcal{T}_k\}$, $\{\mathcal{D}_1, ..., \mathcal{D}_k\}$, $model\, f_\theta$)
2:      Choose an initial vector for $\theta$ and learning rate $\alpha$, $\beta$ and number of tasks per batch (MB).
3:      Initial M as the size of meta epochs and NS as the number of shots.
4:      for m=1 to M do
5:          Create meta-batch $\mathcal{K}_m$ by random sampling |MB| tasks from $\{\mathcal{T}_1, ..., \mathcal{T}_k\}$
6:          for each task $\mathcal{T}_j \in \mathcal{K}_m$ do                ▷ $\mathcal{K}$ is a batch of tasks.
7:              Calculate Prototypes and Proto-loss for task $\mathcal{T}_j$.
8:                                      ▷ Only use NS number of samples to calculate Prototypes.
9:              Adapt model with $\theta'$ as in Equation 23 using sample from $D_j$
10:         Update model parameter with $\theta$ as in Equation 24
```

$$\theta'^{(t)}_j = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_\theta(\mathcal{D}^{train}_j)) \tag{23}$$

$$\theta^{(t+1)} = \theta^{(t)} - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(f_{\theta'_j}(\mathcal{D}^{val}_j)) \tag{24}$$

I designed two types of experiments for my meta-learning framework. Simple mode and complex mode. In both experiments, the number of shots and meta batch size are equal to 5.

4.8.3 Simple Mode Setup

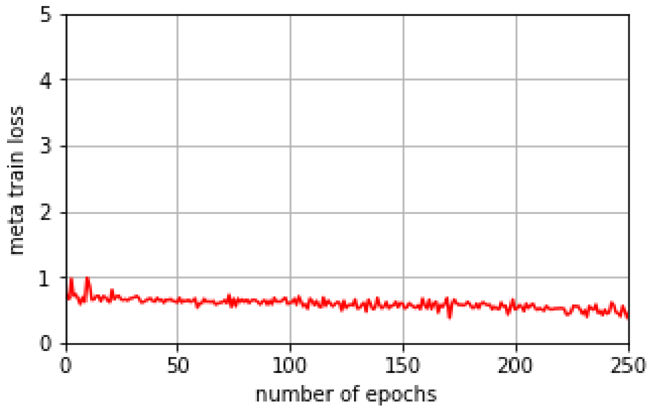
In this mode, I expose the model to only one type of phenotype across multiple cancer studies. For example, I trained the meta-learner to learn to predict histological type in 4 different types of cancers and tested on a new one. In this set of experiments, the size of the meta-train set (i.e. the number of datasets) in various tasks is different. In all experiments, the samples from test tasks have not been shown to the model over previous runs. The full description of phenotypes is provided in section 4.3.2.

4.8.3.1 Event Prediction

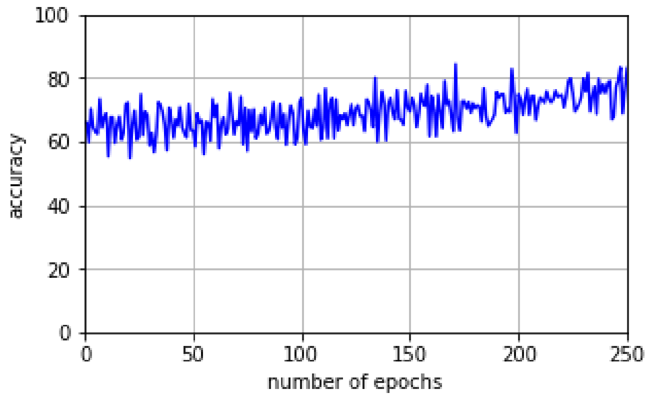
Trained on 18 various types of cancer and tested on 5 new ones that has been provided in table 3. The average train and test accuracy are 65.5 and 72.0 respectively. Figure 22 shows loss decreases and accuracy increases over 250 epochs at a very slow rate. This can be due to the high variety of cancers in the event prediction tasks and also the lack of samples under the few-shot learning scenario. The description of cancer types is presented in section 4.3.1.

Phenotype	Cancer	Logistic Regression	KNN	MLP	Meta-Learner (ProtoNet)
Event	STAD	**61.0**	57.9	52.1	**61.0**
Event	THCA	**97.0**	**97.0**	56.2	**97.0**
Event	SKCM	52.0	56.0	**74.1**	58.99
Event	SARC	63.0	59.0	**97.0**	63.0
Event	LIHC	61.0	63.0	**63.9**	63.0

Table 3: The accuracy of event prediction tasks across models. ProtoNet has been trained on 18 different cancers and tested on 5 new ones.



(a) The average meta batch loss



(b) The average meta batch accuracy

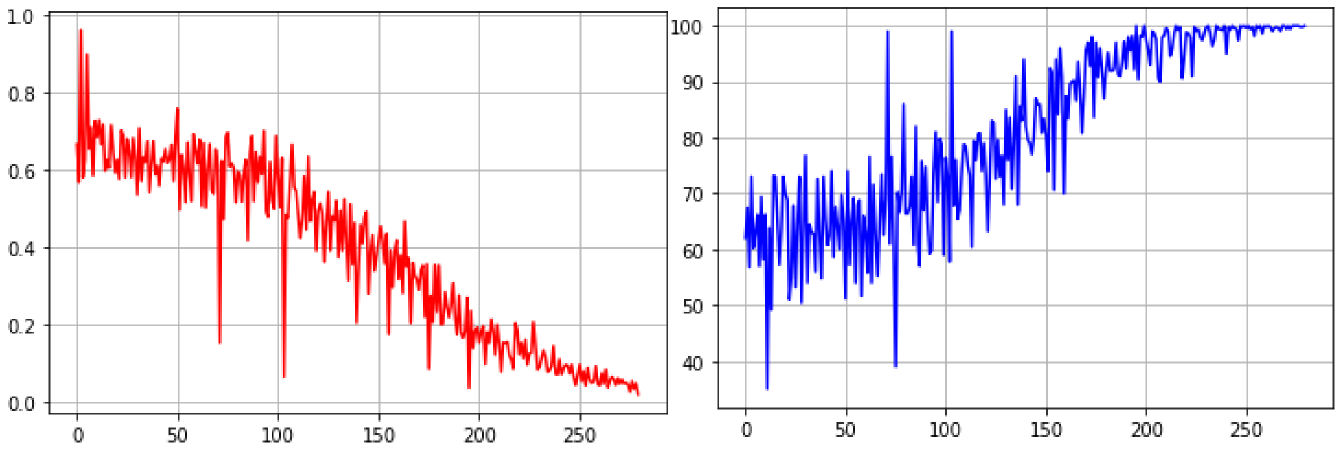
Figure 22: The average loss and accuracy for event prediction using 18 tasks in the meta train set.

4.8.3.2 Gender Prediction

Trained on 16 types of cancers including LUNG, LAML, PAAD, etc. and tested on 5 other types namely THCA, STAD, PCPG, SARC, and SKCM. The average train and test accuracy are 78.41 and 77.8 respectively. In figure 23, one can see the trend of decreasing loss and increasing accuracy over epochs. However, there are many fluctuations due to the variety of tasks in each batch. The results can be found in table 4.

Phenotype	Cancer	Logistic Regression	KNN	MLP	ProtoNet
Gender	THCA	**92.6**	75.0	80.8	81.0
Gender	STAD	82.1	61.0	73.4	**87.0**
Gender	SARC	74.7	61.0	63.8	**96.0**
Gender	SKCM	73.3	57.9	66.1	**75.0**
Gender	PCPG	87.3	63.0	64.9	**99.0**

Table 4: The accuracy of gender prediction tasks across models. The meta-learner has been trained on 16 different cancers and tested on 5 new cancers.



(a) The average meta batch loss (b) The average meta train accuracy

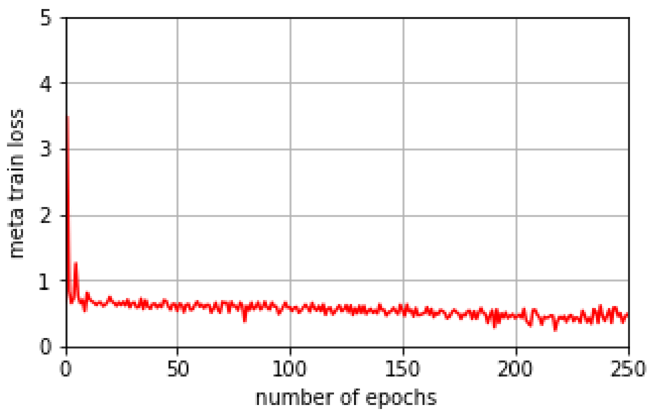
Figure 23: The average train loss and accuracy for gender prediction using 16 tasks in the meta-train set.

4.8.3.3 Oct Embedded Prediction

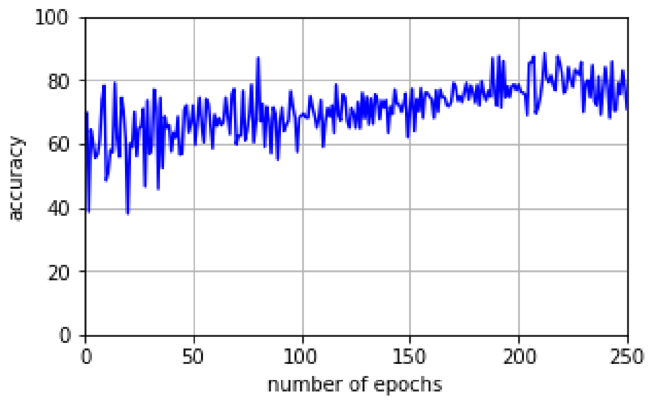
Trained on 12 cancers including BLCA, LIHC, LGG, CESC, GBMLGG, KIRP, and a few others and tested on 4 new cancers including THCA, SKCM, STAD, and UCEC. The average train accuracy is 67.9 and the average test accuracy is 65.75. The results are presented in table 5. Moreover, the learning curve has been provided in figure 24.

Phenotype	Cancer	Logistic Regression	KNN	MLP	ProtoNet
Oct embedded	THCA	**64.9**	52.0	**66.8**	58.0
Oct embedded	SKCM	75.3	67.0	79.6	**80.0**
Oct embedded	STAD	60.58	**64.0**	59.6	60.0
Oct embedded	UCEC	57.1	53.0	57.1	**65.0**

Table 5: The accuracy of oct embedded prediction tasks across models. In ProtoNet, the model has been trained on 12 different cancer types and tested on 4 new ones.



(a) The average meta batch loss



(b) The average meta train accuracy

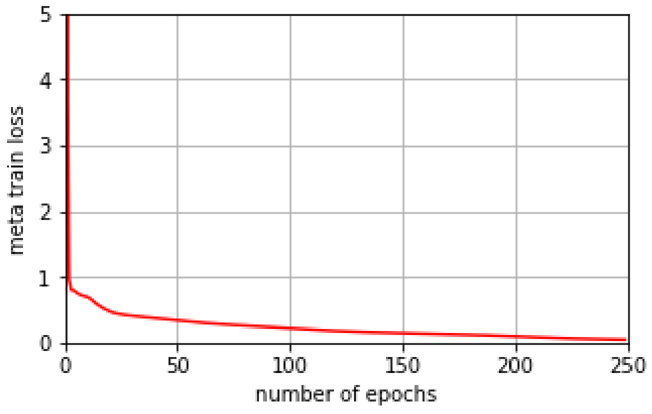
Figure 24: The average train loss and accuracy for oct embedded prediction using 12 tasks in the meta-train set.

4.8.3.4 Histological Type Prediction

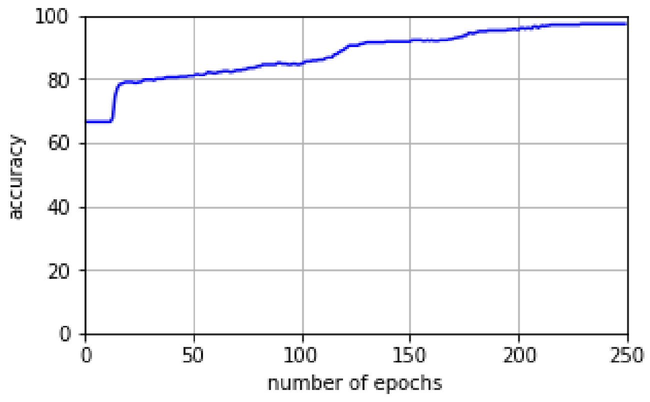
The model has been trained on 4 different cancer studies including LGG, PCPG, ESCA, PRAD with an average accuracy of 83.41. As presented in table 6, the model tested on UCEC with an accuracy of 62.0. Histological type for UCEC cancer is a 3-classes classification task. One can find the learning curve of this task in figure 25.

Phenotype	Cancer	Logistic Regression	KNN	MLP	ProtoNet
Histological type	UCEC	68.4	**75.0**	71.7	62.0

Table 6: The accuracy of histological type prediction tasks across models. In the prototypical network, the model has been trained on 4 different cancers and tested on a new cancer.



(a) The average meta batch loss



(b) The average meta batch accuracy

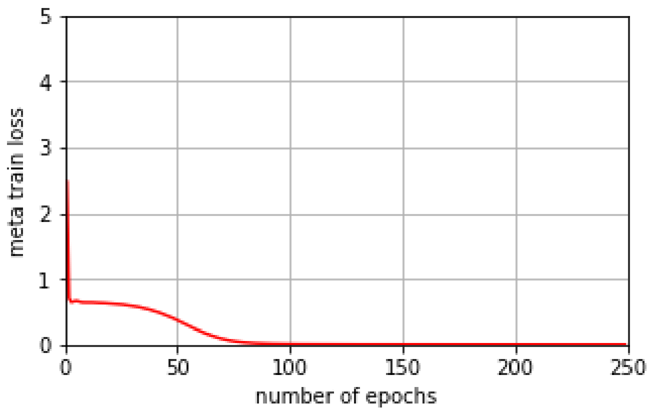
Figure 25: The average train loss and accuracy for histological type prediction using 4 tasks in meta train set.

4.8.3.5 Colon Polyps Presence Prediction

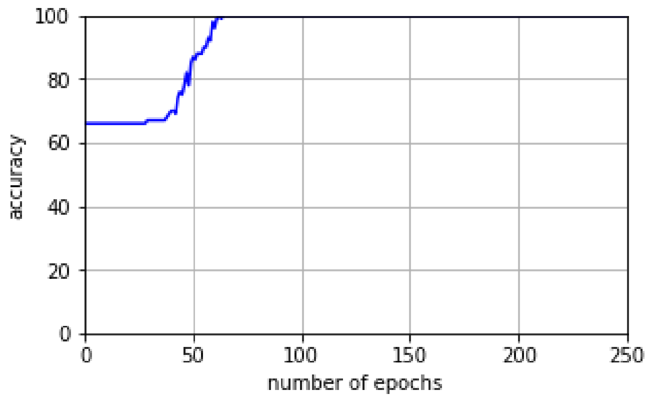
Trained on only one type of cancer that is called COAD and tested on a new type named as COADREAD. The average train accuracy is 89.40 and the test accuracy is 71.0. Table 7 shows a comparison of accuracy among models. As there is only one task in the meta-train set, there is no fluctuation in the learning curve 26.

Phenotype	Cancer	Logistic Regression	KNN	MLP	ProtoNet
Colon polyps present	COADREAD	65.6	72.0	68.8	**71.0**

Table 7: The accuracy of colon polyps presence prediction task across models. In the prototypical network, the model has been trained on COAD cancer and tested on COADREAD.



(a) The average meta batch loss



(b) The average train accuracy

Figure 26: The average train loss and accuracy for colon polyps presence prediction using 1 task in the meta-train set.

4.8.4 Complex Mode Setup

In this scenario, I trained the meta-learner on the full meta-dataset that I defined earlier in Section 4.4. The meta-train set includes 96 tasks and the meta-test set contains 25 different tasks that have been shown in table 8 (only the tasks with at least 150 samples have been selected). In table 8, one can see the performance of the ProtoNet meta-learner over 25 tasks provided. As illustrated, the ProtoNet does not perform as well as a finetune neural net on all tasks. This result was expected as the finetune neural network is a specialized on each particular task. Also, the meta-test tasks are all new and have not been seen by the model over past experiences. To have a fair comparison, all models have been provided using 50 training and 50 test samples meaning the training procedure has been done under few-shot learning setting.

4.9 Results

Table 9 shows that the finetune neural network has the highest overall performance compared to the prototypical network, logistic regression, and k-nearest neighbors. The main limitation of logistic regression is that it cannot model the interactions between input variables. While neural networks are dealing better with clinical classification tasks, logistic regression has higher performance on gender tasks. This is because gender tasks can be classified by using only one gene, i.e. whether it includes y chromosome or not.

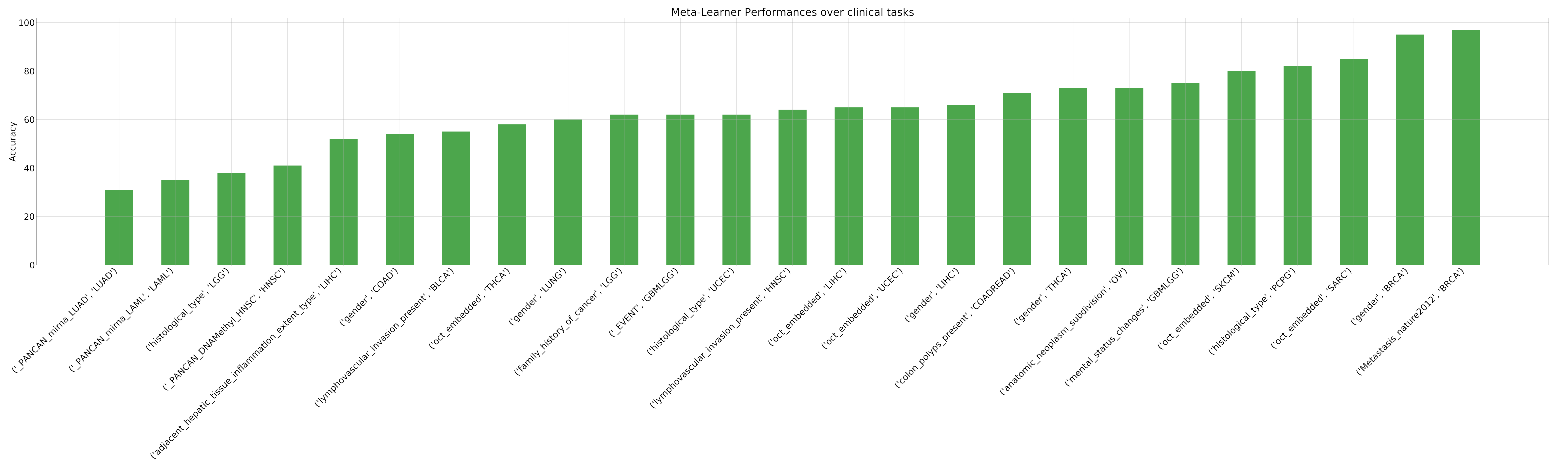


Figure 27: Prototypical Network performance on the defined genomics tasks.

In table 9, you can see the average accuracy of each model over tasks as well as 95% confidence interval of models over multiple trials . This result has been concluded over 10 trials. In figure 27,

ID	Phenotype	Cancer	LR		KNN		MLP		ProtoNet	
			Acc	Rank	Acc	Rank	Acc	Rank	Acc	Rank
1	Pancan Micro RNA	LUAD	37.3	3	**47.0**	1	39.7	2	31.0	4
2	Pancan Micro RNA	LAML	64.9	3	69.0	2	**71.3**	1	35.0	4
3	Histological Type	LGG	54.7	2	46.0	3	**55.6**	1	38.0	4
4	Pancan DNAMethyl	HNSC	64.8	2	54.0	3	**74.1**	1	41.0	4
5	Adj. Hepatic Tissue Inflamm. Extent Type	LIHC	51.6	4	**56.9**	1	54.6	2	52.0	3
6	Gender	COAD	**95.1**	1	63.0	3	79.8	2	54.0	4
7	**Lymphovascular Invasion Present**	BLCA	53.8	2	48.0	4	53.1	3	**55.0**	**1**
8	Oct Embedded	THCA	64.9	2	52.0	4	**66.8**	1	58.0	3
9	Gender	LUNG	**87.5**	1	76.0	3	77.1	2	60.0	4
10	**Family History of Cancer**	LGG	50.7	4	61.0	2	55.2	3	**62.0**	**1**
11	Event	GBMLGG	**74.0**	1	71.0	3	**74.0**	1	62.0	4
12	Histological Type	UCEC	68.4	3	**75.0**	1	71.7	2	62.0	4
13	Lymphovascular Invasion Present	HNSC	60.0	4	**68.0**	1	64.3	2	64.0	3
14	Oct Embedded	LIHC	61.2	4	**67.0**	1	64.6	3	65.0	2
15	**Oct embedded**	UCEC	57.1	2	53.0	4	57.1	2	**65.0**	**1**
16	Gender	LIHC	**87.4**	1	72.0	3	78.2	2	66.0	4
17	Colon Polyps Present	COADREAD	65.6	4	**72.0**	1	68.8	3	71.0	2
18	Gender	THCA	**92.6**	1	75.0	3	80.8	2	73.0	4
19	**Anatomic Neoplasm Subdivision**	OV	59.6	4	72.0	2	72.0	2	**73.0**	**1**
20	**Mental Status Changes**	GBMLGG	65.3	4	74.0	2	67.3	3	**75.0**	**1**
21	**Oct Embedded**	SKCM	75.3	3	67.0	4	79.6	2	**80.0**	**1**
22	Histological Type	PCP	82.6	2	74.0	4	**84.5**	1	82.0	3
23	**Oct Embedded**	SARC	75.6	3	70.0	4	80.1	2	**85.0**	**1**
24	Metastasis	BRCA	**98.0**	1	**98.0**	1	**98.0**	1	97.0	4
25	Gender	BRCA	**99.0**	1	**99.0**	1	**99.0**	1	95.0	4
		Average Rank		2.48		2.44		1.88		2.84

Table 8: Few-shot classification results on Meta-test set tasks using classical models and meta-learner trained on Meta-training set tasks only using 5 samples. The evaluation metric is accuracy and it has been provided in an ascending order.

Model	**Accuracy**
Logistic Regression	69.96 ± 6.78
K-Nearest Neighbor	67.20 ± 5.55
Fine-tuned Neural Network	**70.73 ± 5.65**
Prototypical Network	64.04 ± 7.18

Table 9: Accuracy and 95% confidence interval of models over multiple trials.

the overall performance of ProtoNet over 25 tasks has been provided. The meta-learning algorithm (MAML) from [22] was indeed evaluated but performed so poorly that the specific results were not worth including in the thesis. By looking at this figure, one can realize which tasks are easier for meta-learners to learn. The reason behind the poor performance for the tasks at the beginning of the plot is either that they have more labels in comparison to the others meaning the tasks are intrinsically more complicated, or the knowledge learned so far is not sufficient for these tasks to provide good insight to the model. This is the result of a complex mode setup which is the ultimate goal of meta-learning, the ability of the model to generalize to new tasks coming from different distributions. However, I also designed a set of simplified experiments as I explained in 4.8.3, where I trained the meta-learner only on one type of phenotype across different cancer studies. This way, I keep the distribution of the phenotype the same over experiments. By looking at figures 22 to 25, one can discern that by training and testing only on one type of phenotype, the meta-learner achieves higher performance than the complex setup where the model is trained on a higher variety of tasks. This is expected as it is easier for the model to learn the shared structural information only across cancers rather than learning the higher-level features over both various phenotypes and cancers.

In the majority of existing meta-learning algorithms including ProtoNet, the meta-objective is averaged over tasks distributions (playing a role similar to the distribution over datasets in Section 3.5.1.

Therefore, the learning rule achieved by the meta-learning procedure is conditioned on the optimization procedure of θ^*. The model should on average see its performance increase after the adaptation phase. The behavior on average across several tasks does not ensure good performance on an individual task. The adaptation could significantly improve the performance of the model on a subset of tasks while decreasing it on another. However, at test-time, only the performance of the model on a single new task is of interest. This is an issue that does not appear in classical supervised learning since the model is only trained on one task.

This contradiction between the meta-learning objective and evaluation can lead to unexpected outcomes, where the performance of the model might decrease on specific tasks after adaptation [18]. This is especially critical in high-risk applications, such as healthcare, where guarantees on the learning phase are necessary. Can the learning rule resulting from the meta-learning be adequately restrained to avoid this sort of uninformative behavior?

4.10 Conclusion

This chapter presented an analysis of the incorporation of prior knowledge into neural networks for genomics and clinical task prediction. My goal was to use gene expression data to learn the higher-level features between various phenotypes. For this purpose, I implemented three classical machine learning models as well as a meta-learner called prototypical network. According to the results of the experiments that have been provided in Section 4.9, a finetune neural network with 2 hidden layers has the highest overall performance compared to the prototypical network, logistic regression, and k-nearest neighbors. While neural networks are dealing better with clinical classification tasks, logistic regression has higher performance on gender tasks. As provided in table 8, ProtoNet shows a higher performance on some tasks including detecting lymphovascular invasion for bladder cancer, family history of cancer for lower-grade glioma, oct embedded for endometrioid, melanoma, and sarcoma as well as predicting mental status change in lower-grade glioma and glioblastoma These tasks are all binary classification problems and there are only two types of labels.
When the tasks are similar to each other during meta-train and meta-test, the meta-learner shows a higher performance in comparison to the case that there is a high diversity of tasks in the meta-train set. However, this diversity is also important to certain extents because a batch of similar tasks might not be informative enough for the meta-learning algorithm to learn a good inductive bias.

Chapter 5

Conclusion and Future Work

In this work, I proposed a Meta-learner that can learn multiple genomics tasks using gene expression data and a Meta-dataset of tasks where each task is combinations of a phenotype to predict and a cancer study. This dataset can be used as a test-bed for developing algorithms for genomics applications which can leverage a few-shot learning regime. I also evaluated the performance of the defined tasks using supervised models and compared their accuracy to provide a baseline of what current techniques can achieve on. Since there is a limited number of samples available in the few-shot learning regime, we leverage the samples from a collection of tasks and construct a prior knowledge for a general prediction by using meta-learning.

In chapter 2, first, we give an introduction to genomic basics and present a couple of machine learning problems including supervised, semi-supervised, and unsupervised learning with more emphasis on supervised setting. Later, we explain the development of deep learning and how it is linked to meta-learning.

Chapter 3 gives detail about meta-learning framework, different categories of meta-learning models and their difference. We conclude this chapter by explaining the meta-learner selection choice for the current application.

In chapter 4, we present a study of meta-learning on genomics task predictions using gene expression data by using the prototypical network as a meta-learner, and also logistic regression, k-nearest neighbors, and an MLP as classical models. By sharing knowledge between tasks and using the adaptation procedure, we could learn new genomics tasks. This would be pretty helpful for the prediction and diagnosis of rare cancers that do not have enough number of samples. Even though there are still some failures in the prediction of particular tasks (either complicated or under-represented), I believe that addressing these shortcomings constitutes an important research goal of cancer detection moving forward.

I believe meta-learning with the mentioned definition in Chapter 3, is one more step towards a higher level of intelligence as it can uncover new designs that outperform hand-designed architectures. Applying meta-learning methods allows us to construct shared structural information within interrelated tasks by using multiple sources of data.

5.1 Future Work

In this section, we provide potential ideas for future research directions.

5.1.1 Meta-Learning Sensitivity to Task Distributional Changes

The standardized benchmarks, based on fixed task distributions, were developed to get a fair comparison between different meta-learning algorithms. However, it is unclear how robust these algorithms are across different task distributions. A future goal can be to design more robust and more consistent meta-learning methods across multiple task distributions.

5.1.2 Task Specification

Task specification is an important matter in the meta-learning procedure. Currently, the notion of a task is not clearly defined in the meta-learning framework. The meta-learning foundations are based on the assumption that we have access to distribution of tasks. In the conventional benchmarks for few-shot supervised learning, this distribution is often provided manually for convenience. However, the structure of this distribution is crucial in learning the appropriate inductive biases since the prior knowledge is formed based on the dataset. Therefore, it contributes greatly to the success or failure of meta-trained solutions. Nonetheless, the way this distribution is defined has never been questioned.

5.1.3 Task Distribution Diversity Quantification

The task distribution has an important impact on the task efficiency during meta-training, which specifies the optimal value for the number of tasks required in the meta-training set or the number of meta-training steps. For example, if the algorithm operates on batches of tasks, a step on a batch of similar tasks might not be informative in the overall meta-training procedure. How diverse the distribution of tasks should be during the meta-training and how we can quantify this diversity, are interesting questions.

5.1.4 Similarity Metric Definition

A distribution over tasks implicitly requires a notion of distance between these tasks. In this case, what does it mean for tasks to be similar (i.e. to be close to each other)? Computing this distance between tasks is also related to how these tasks are represented in an embedding space. The notion of similarity between tasks needs an implicit definition of distance metric.

5.1.5 The Role of Memory in Meta-Learning

The role of memory in meta-learning is an interesting direction. It is easier for the network to quickly integrate new information and not forget them in the future by using external memory. An agent is unlikely to receive a full set of tasks at once in real-world applications. It would instead experience these tasks sequentially. This setting is commonly introduced as continual learning, or lifelong learning, which is closely related to meta-learning. There is also an open question related to this topic that is “what would be a good encoding and decoding scheme for new information?”

Bibliography

[1] A Collaborative Learning Space for Science. https://www.nature.com/scitable/definition/. Accessed: 2019-12-10.

[2] Canadian Cancer Society. https://www.cancer.ca/. Accessed: 2019-12-03.

[3] DREAM Challenges. http://dreamchallenges.org/. Accessed: 2019-11-05.

[4] Gene Wikipedia. https://en.wikipedia.org/wiki/Gene#cite_note-Edd01-46. Accessed: 2019-12-25.

[5] Genome Quebec Tutorials. https://www.genomequebec-education-formations.com/. Accessed: 2019-10-17.

[6] Meta-Learning Scholarpedia. http://www.scholarpedia.org/article/Metalearning. Accessed: 2020-01-12.

[7] Meta-Learning Wikipedia. https://en.wikipedia.org/wiki/Meta_learning_(computer_science). Accessed: 2020-01-06.

[8] Prototypical Network Github Repository. https://github.com/jakesnell/prototypical-networks. Accessed: 2019-09-11.

[9] TCGA Cancer Phenotypes. https://docs.cancergenomicscloud.org/docs/. Accessed: 2019-12-11.

[10] Mayo Clinic. https://www.mayoclinic.org/, Accessed: 2019-12-03.

[11] Rahaf Aljundi. *Continual Learning in Neural Networks.* PhD thesis, KU Leuven, Faculty of Engineering Science, 2019.

[12] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *ArXiv*, abs/1606.04474, 2016.

[13] Yoshua Bengio, Frdric Bastien, Arnaud Bergeron, Nicolas BoulangerLewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Ct, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, Franois Savard, and Guillaume Sicard. Deep learners benefit more from out-of-distribution examples. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 164–172, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[14] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. *IJCNN-91-Seattle International Joint Conference on Neural Networks*, ii:969 vol.2–, 1991.

[15] Derek Browne. Cognitive versatility. volume 6, pages 507–523. Minds and Machines, 1996.

[16] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, and Andrea Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, 118(1):65–94, 2016.

[17] Gabriela Csurka. *Domain Adaptation in Computer Vision Applications*. Springer International Publishing, 2017.

[18] Tristan Deleu and Yoshua Bengio. The effects of negative adaptation in Model-Agnostic Meta-Learning. *2nd Workshop on Meta-Learning at NeurIPS 2018, Montreal, Canada*, 2018.

[19] Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. `https://github.com/tristandeleu/pytorch-meta`.

[20] MD Dr. Ananya Mandal. Gene expression measurement, March 2019. `https://www.news-medical.net/life-sciences/Gene-Expression-Measurement.aspx`.

[21] Chelsea Finn. *Learning to Learn with Gradients*. PhD thesis, UC Berkeley, 2018.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*, 2017.

[23] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[24] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining: Concepts and techniques. *Morgan Kaufmann Imprint*, 22nd June 2011.

[25] Douglas Hanahan and RobertA. Weinberg. Hallmarks of cancer: The next generation. *Cell*, 144(5):646 – 674, 2011.

[26] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, Volume. 6:96, 2019.

[27] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 87–94, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[28] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. 2013.

[29] Jonas Jongejan, Takashi Kawashima Henry Rowley, Jongmin Kim, and Nick Fox-Gieg. Quick, Draw. A.I. experiment. `quickdraw.withgoogle.com`, 2016.

[30] Schmidhuber Juergen. *Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook.* PhD thesis, Institut fr Informatik, Technische Universitt Mnchen, 1987.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[32] Kevin B. Korb. Introduction: Machine learning as philosophy of science. *Minds Mach.*, 14(4):433440, November 2004.

[33] Amnon Koren, Itay Tirosh, and Naama Barkai. Autocorrelation analysis reveals widespread spatial biases in microarray experiments. *BMC Genomics*, 8:164 – 164, 2007.

[34] Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis, and Dimitrios I. Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8 – 17, 2014.

[35] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. The Omniglot Challenge: A 3-Year Progress Report. 2019.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998.

[37] Louis Lello, Steven G. Avery, Laurent Tellier, Ana I. Vazquez, Gustavo de los Campos, and Stephen D. H. Hsu. Accurate genomic prediction of human height. *Genetics*, 210(2):477–497, 2018.

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[39] Christoph Lippert, Riccardo Sabatini, M. Cyrus Maher, Eun Yong Kang, Seunghak Lee, Okan Arikan, Alena Harley, Axel Bernal, Peter Garst, Victor Lavrenko, Ken Yocum, Theodore Wong, Mingfu Zhu, Wen-Yun Yang, Chris Chang, Tim Lu, Charlie W. H. Lee, Barry Hicks, Smriti Ramakrishnan, Haibao Tang, Chao Xie, Jason Piper, Suzanne Brewerton, Yaron Turpaz, Amalio Telenti, Rhonda K. Roby, Franz J. Och, and J. Craig Venter. Identification of individuals by trait prediction using whole-genome sequencing data. *Proceedings of the National Academy of Sciences*, 114(38):10166–10171, 2017.

[40] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(1-3):503–528, August 1989.

[41] Jianfang Liu, Tara Lichtenberg, Katherine A. Hoadley, Laila M. Poisson, Alexander J. Lazar, Andrew D. Cherniack, Albert J. Kovatich, Christopher C. Benz, Douglas A. Levine, Adrian V. Lee, Larsson Omberg, Denise M. Wolf, Craig D. Shriver, Vesteinn Thorsson, The Cancer Genome Atlas Research Network, and Hai Hu. An integrated tcga pan-cancer clinical data resource to drive high quality survival outcome analytics. *Cancer Research*, 78(13 Supplement):3287–3287, 2018.

[42] Boyu Lyu and Anamul Haque. Deep learning based tumor type classification using gene expression data. *ACM-BCB*, pages 89–96, 2018.

[43] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013.

[44] Sunil Mathur and Joseph Sutton. Personalized medicine could transform healthcare. volume 7, pages 3–5, Berlin, Heidelberg, 2017 Jul. Biomedical Report PMC.

[45] Stefan Michiels, Serge Koscielny, and Catherine Hill. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *The Lancet*, 365:488–492, 02 2005.

[46] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[47] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, ICVGIP 08, page 722729, USA, 2008. IEEE Computer Society.

[48] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[49] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[50] Victor Roman. Supervised learning: Basics of classification and main algorithms. `https://towardsdatascience.com/supervised-learning-basics-of-classification-and-main-algorithms-c16b06806cd3`, 2019. Accessed: 2019-11-22.

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[52] Mandana Samiei. Tcga benchmark tasks for clinical attribute prediction based on genome. `https://github.com/mandanasmi/TCGA_Benchmark`, 2019.

[53] Mandana Samiei, Tobias Wrfl, Tristan Deleu, Martin Weiss, Francis Dutil, Thomas Fevens, Genevive Boucher, Sebastien Lemieux, and Joseph Paul Cohen. The TCGA Meta-Dataset Clinical Benchmark. CoRR, 2019.

[54] Tom Schaul and Juergen Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, January 2010.

[55] B. Schroeder and Y. Cui. Fgvcx fungi classification challenge. 2018. `github.com/visipedia/fgvcx_fungi_comp`.

[56] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

[57] Doris Steger, David Berry, Susanne Haider, Matthias Horn, Michael T. Wagner, Roman Stocker, and Alexander Loy. Systematic spatial bias in dna microarray hybridization is caused by probe spot position-dependent variability in lateral diffusion. *PLoS ONE*, 6, 2011.

[58] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[59] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019.

[60] P. Utgoff. Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, volume 2, pages 163–190. Morgan Kaufmann, Los Altos, CA, 1986.

[61] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS16, page 36373645, Red Hook, NY, USA, 2016. Curran Associates Inc.

[62] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[63] Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *CoRR*, abs/1904.05046, 2019.

[64] Lilian Weng. Meta-learning: Learning to learn fast. `https://lilianweng.github.io/lil-log/`. 2018-11-30.

[65] Wessel Wieringen, David Kun, Regina Hampel, and Anne-Laure Boulesteix. Survival prediction using gene expression data: A review and comparison. *Computational Statistics & Data Analysis*, 53:1590–1603, 2009.

[66] Arthur Younger, Sepp Hochreiter, and P.R. Conwell. Meta-learning with backpropagation. volume 3, pages 2001 – 2006 vol.3, 02 2001.