# Machine learning for medical applications with limited data: Incorporating domain expertise and addressing domain-shift

by

Roberto Ivan Vega Romero

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

Doctor of Philosophy

in

Statistical Machine Learning

Department of Computing Science

University of Alberta

# Abstract

Machine learning has the potential to help medical experts to deliver better healthcare. There are, however, important technical challenges that need to be solved before we can develop reliable models for clinical practice, including: (1) Limited number of labeled instances, (2) Uncertainty of the labels used during training, and (3) Differences between the distributions that generated the training and test data. This dissertation focuses on strategies for effectively applying machine learning under these circumstances.

For learning models from a limited number of labeled instances, we propose incorporating domain expert knowledge during the training process. This domain expertise can be encoded in the form of probabilistic labels, which provide more information per instance than the commonly used categorical labels, or by using machine learning to extend the medical models currently used by human experts. We demonstrate the effectiveness of the probabilistic labels in three medical image classification tasks: for diagnosing hip dysplasia, fatty liver, and glaucoma. We observed gains up to 22% in terms of classification accuracy when compared with the use of categorical labels. We also show how to use machine learning to extend an SIR (Susceptible-Infected-Removed) epidemiological model for predicting the evolution in the number of people infected with COVID-19, achieving state-of-the-art results in terms of mean absolute percentage error (MAPE) in data from the United States and Canada.

For addressing the uncertainty around the labels, we use probabilistic

graphical models. Instead of providing a point-estimate, probabilistic models predict an entire probability distribution, which accounts for the uncertainty in the data. Probabilistic models are a key component of the probabilistic labels mentioned above, and they also allow the incorporation of human decision making for tracking the number of new infections when using machine learning with the SIR model.

Finally, a consequence of training machine learning models with a limited number of labeled instances is that the training set might not be an accurate reflection of the data used during inference – in particular, the test set might not follow the same probability distribution that generated the training data. This means that a predictor learned from one dataset might do poorly when applied to a second dataset. This problem is known as *batch effects* or *dataset shift*, while approaches to correct for the discrepancies in these probability distributions fall under the umbrella term *domain adaptation*. Depending on the assumptions on what causes the discrepancy, these problems might be studied under specific names, such as covariate-shift, class-imbalance, etc.

Here, we first propose an algorithm for domain-shift adaptation when the discrepancy between distributions is caused by linear transformations, and then empirically show that style transfer can alleviate domain-shift caused by changes in texture. We provide empirical results for the task of segmentation of the hip in ultrasound images, with gains of up to 20% in terms of Dice score when applying style transfer for unsupervised domain adaptation.

Although all the applications in this dissertation are related to the medical domain, we expect that the techniques shown here are applicable when: (1) expert knowledge can be encoded as probabilities, (2) there exist a parametric model currently used by domain experts for analyzing a phenomenon, and/or (3) the discrepancy between the source and target domains is caused by affine transformations or differences in texture.

# Preface

Some chapters of this thesis have been published in conferences or journals.

Chapter 2 was published as Vega, Roberto, et al. “Sample efficient learning of image-based diagnostic classifiers via probabilistic labels.” *International Conference on Artificial Intelligence and Statistics. PMLR, 2021.* The machine learning components were done by myself, Pouneh Gorji and Rusell Greiner. Zichen Zhang, Xuebin Qin, Abhilash Rakkunedeth, Jeevesh Kapur and Jacob Jaremko performed the data collection and manual data labeling.

Chapter 3 was published as Vega, Roberto, Leonardo Flores, and Russell Greiner. “SIMLR: Machine Learning inside the SIR model for COVID-19 Forecasting.” *Forecasting 4.1 (2022): 72-94.* I developed and implemented the machine learning algorithms, while Leonardo Flores and Russell Greiner played the role of supervisors.

The first part of chapter 4 was published as Vega, Roberto, and Russ Greiner. “Finding effective ways to (machine) learn fmri-based classifiers from multi-site data.” *Understanding and Interpreting Machine Learning in Medical Image Computing Applications. Springer, Cham, 2018. 32-39.* The second part is published as Vega, Roberto, and Russell Greiner. “Domain-shift adaptation via linear transformations.” *arXiv preprint arXiv:2201.05282 (2022).* In both cases I developed and implemented the machine learning algorithms, while Russell Greiner played the role of supervisor.

*To AnaLi*

*For being always there*

# Acknowledgements

The completion of my Ph.D. is the result of the hard work of many individuals. Some of them contributed in the academic setting, some of them contributed by providing resources, and some others contributed in a personal level.

Firstly, I want to thank AnaLi, my wife, for being with me during all these years. I would not have accomplished half of the things that I have done without you. You are my inspiration, my guide, and the love of my life. Spending my life with you is the best reward that I can ever have. I love you.

Secondly, but not less importantly, I want to thank Russ Greiner, my supervisor during both my masters and my Ph.D. I have learned a lot from you, not only as an academic and researcher, but as a person. I aspire to be like you one day.

I want to thank all the people who were part of my committee: Dr. Alona Fyshe, Dr. Nilanjan Ray, Dr. Dale Schuurmans, Dr. Pierre Boulanger, Dr. Ross Mitchell and Dr. Mahmoud El-Sakka, for taking the time to read my research and provide feedback.

I started my journey as a researcher many years ago, before coming to Canada, and I have been inspired by very smart people whose passion and love for research left a mark in me. Among these people are Leonardo Flores, Dr. Alejandro García, Dra. Rita Fuentes, Dr. Mauricio Antelis, and Dr. Gildardo Sánchez. Gildardo opened the doors of research to me, first as a professor in the Robotics course, and then as a mentor, when he took me as a research assistant. Thanks for giving me to opportunity to learn from you.

I have met many amazing people during my PhD journey, and I am greatful to all the member of the Russ Greiner's lab. I enjoyed a lot all the interactions and exchange of ideas that we had! I had the fortune to meet people at

the university who I consider now as friends: Neil Borle, Jacqueline Harris, Negar Hassanpour, Amira Aissiou, Fei Wang, Sunil Kalmady, Fatima Davelouis, Bernal Manzanilla, Carolina Quiroz, Farzane Aminmansour, Nouha Dziri, Omar Rivasplata, and Pouneh Gorji. Thanks for making this an incredible journey!

I also want to thank all the people at Medo.AI (now Exo) for allowing me to be part of some amazing projects in the intersection of machine learning and medical imaging during my PhD. I want to specially thank Dornoosh Zonoobi and Masood Dehghan for their trust, mentorship, flexibility and for being amazing friends. Important components of my research arose from discussions with them.

I also want to thank the different agencies that supported my during all these years: CONACYT, who gave me an amazing scholarship during the first four years of my program. The Alberta Machine Intelligence Institute (AMII) for providing funding opportunities as a Research Assistant. The Google Cloud Platform, for providing research credits and access to infrastructure that allowed us to accelerate our research. The Computing Science department, who gave me the opportunity to be a Teaching Assistant during the first years of my program.

Finally, I want to thank my parents Roberto Vega and Altagracia Romero, my brother Alejandro Vega, and my family in-law (especially Ana Gonzalez) for always being there to support me. You were always part of this journey and I'm proud to be part of this family. I love you.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The combination of medical expertise and machine learning (ML) can lead to tools for screening, diagnosis, prognosis, and hence better patient outcomes [53, 127, 25, 37]. For example, in areas like medical imaging, machine learning might reduce the time that radiologists spend manually analyzing images while keeping the high quality in the image interpretation [97]. Similarly, it can be combined with epidemiological models to predict the new number of infections in the short and medium term, allowing decision makers to act accordingly.

In a nutshell, machine learning algorithms seek relevant patterns in the data, and then use these patterns to make accurate predictions. For the case of supervised machine learning, its objective is to learn a model that maps an input, $x$, to the appropriate output $y$, where $x \in \mathbb{R}^p$ is a $p$-dimensional vector containing the values of a set of features (here, perhaps a description of a patient), and $y$ is its corresponding label (perhaps a bit indicating if a person has or does not have a disease). Usually, $y \in \mathbb{R}$ for regression problems (*e.g.*, a patient's weight), while $y \in \{1, ..., K\}$ for classification problems (*e.g.*, has a disease or not).

The model that maps inputs to outputs is typically a parameteric function $y = f_\theta(x)$, whose parameters, $\theta$, are unknown to the learner. The general strategy is to apply a learning algorithm, $L(\cdot)$, to a labeled training set with $n$ instances $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$, where $D \sim P_{train}(x, y)$. The output of the learning algorithm are parameters, $\theta^*$, that (we hope) minimize an expected cost between the true labels and predictions, $c(y, f_\theta(x))$. After

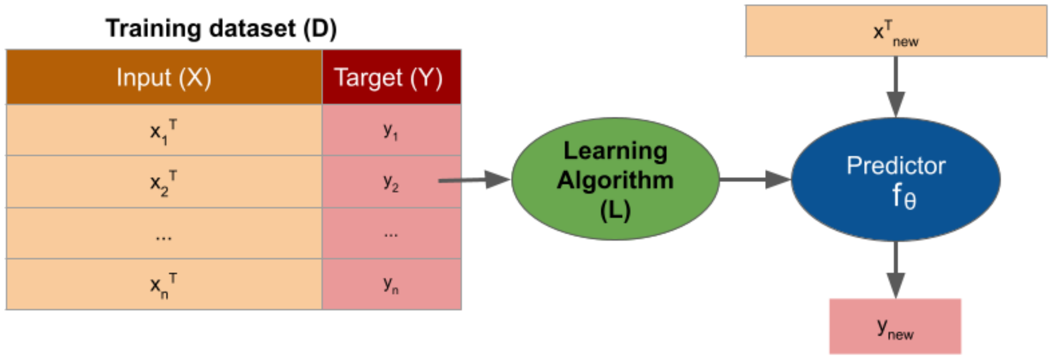


Figure 1.1: Machine learning approach for classification problems

training, it is then possible to make predictions on new instances $(x_{new}, y_{new}) \sim P_{new}(x, y)$ where $y_{new} = f_{\theta^*}(x_{new})$ [87], as illustrated in Figure 1.1.

The framework described so far makes 3 implicit assumptions; if violated, the traditional training of machine learning models might not be enough to achieve the desired performance.:

1. That the labels $y_1, \ldots y_n$ correctly identify the instances described by $x_1, \ldots, x_n$, respectively. The algorithm learns from the pairs $\{(x_i, y_i)\}_i$ in the dataset $D$, so if some of the labels are noisy, then the learning algorithm might learn 'wrong concepts'. In the task of medical diagnosis from an image, for example, we usually face uncertainty in the labels used as ground truth, since different medical experts often give different labels to the same image.

2. That we already know a useful representation, $x$, of the data, or that the instances in the dataset $D$ suffice to allow an algorithm to learn such representation. This is hardly ever the case. Take the case of the automatic analysis of medical images as an example. These images are usually high-dimensional vectors (in the range of $10^4$ to $10^6$ pixels), while the size of the labeled dataset is just in the range of $10^2$ to $10^4$ images.

3. That the joint probability distribution of the new instances, $P_{new}(x, y)$, where we will use the learned predictor, is the same as the probability distribution of our training set $D \sim P_{train}(x, y)$. This assumption is often violated when we train a model using data from one source (*e.g.*,

scanner from manufacturer A), and then apply the model to data from a different one (*e.g.*, scanner from manufacturer B). This problem, known as *batch effects* or dataset shift [108, 93, 73], is caused by technical noise that might confound the real biological signal.

Additionally, when using a model to make critical decisions, which is the case in most medical tasks, there are factors to consider that go beyond the accuracy of the predictions. In some cases it might be important for the models to provide not only the predicted class, but also the confidence in its predictions. In other instances, it might need to provide decision-makers additional information to justify a recommendation [104].

Many parametric machine learning models typically either learn concepts from scratch (random initialization of weights), or are initialized with previously trained models [131, 24]. In either case, those weights are completely data driven; however, there is vast information in the medical literature that gives insight about the relevant characteristics of the data that might be useful for an accurate diagnosis. It is therefore important to find learning methodologies that produce accurate and calibrated models, and that are sample efficient, allow the incorporation of prior knowledge, and generalize across datasets.

The research presented in this document addresses the aforementioned challenges. For learning with a limited number of instances in the presence of uncertainty, we propose to use a combination of probabilistic graphical models, deep learning architectures, and human domain expert knowledge. We will also explore techniques that can often decrease the discrepancy between the probability distributions of the training and test datasets.

Chapter 2, based on *Sample Efficient Learning of Image-Based Diagnostic Classifiers Using Probabilistic Labels* [120], presents probabilistic labels, an approach that replaces categorical labels by probabilities. To compute those probabilities we encode domain specific knowledge in the form of probabilistic graphical models. There "probabilistic labels" give a learner more information per training instance, allowing it to learn useful concepts with fewer instances. In our experiments, this approach led to gains of up to 22% in the accuracy of models, as compared with the use of categorical labels, in three classification tasks: diagnosing hip dysplasia, fatty liver, and glaucoma.

Additionally, probabilistic labels improve the calibration of the learned models, which give a straightforward interpretation of the output as the probability of diagnosis. Although it is tempting (and common) to view the output of a neural network with sigmoid or soft-max activation function as such a probability [39], the typical training of deep neural networks leads to models that do not produce calibrated probabilities [45], which argues that they should not be interpreted as probabilities.

Chapter 3, based on *SIMLR: Machine Learning inside the SIR model for COVID-19 Forecasting* [121], describes SIMLR, which uses machine learning techniques to extend an epidemiological SIR model, for forecasting the number of people newly infected with COVID-19, 1 to 4 weeks in advance. SIMLR is a probabilistic graphical model that tracks changes in the government policies over time, and uses them to forecast the evolution of the pandemic. The use of epidemiological models for this task is particularly important because the available data is scarce [105], while the use of probabilistic models naturally handles the uncertainty in the reported numbers of infections over time [9, 60]. We applied SIMLR to data from Canada and the United States, and show that its mean average percentage error is as good as state-of-the-art forecasting models, with the added advantage of being an interpretable model.

Chapter 4, based on both *Finding Effective Ways to (Machine) Learn fMRI-Based Classifiers from Multi-site Data* [118] and *Domain-shift adaptation via linear transformations* [119], gives a general description of batch effects by casting it as an instance of domain-shift. Given data from a source domain (S), and a target domain (T), domain-shift occurs when $P_S(X,Y) \neq P_T(X,Y)$. Specifically, we focus on the case where $P_S(Y) = P_T(Y)$, $P_S(X) \neq P_T(X)$, and $P_S(Y \mid X) \neq P_T(Y \mid X)$. However, there is a function $f(\cdot,\cdot)$, with parameters $\lambda_S$ and $\lambda_T$, such that $P_S(\ f(X,\ \lambda_S)\ ) = P_T(\ f(X,\ \lambda_T)\ )$ and $P_S(\ Y \mid f(X,\ \lambda_S)\ ) = P_T(\ Y \mid f(X,\ \lambda_T)\ )$.

The first part of Chapter 4 presents a method for combining datasets from different sources (*i.e.*, , drawn from different $P_i(X,Y)$ distributions), when the datasets from all the sources are fully labeled. We show that this method can correct for batch effects caused by arbitrary affine transformations. We test the performance of this method in the task of diagnosis of schizophrenia based on functional magnetic resonance imaging (fMRI).

The second part of the chapter proposes an unsupervised approach to project the source and target domains into a lower-dimensional, common space, by (1) projecting the domains into the eigenvectors of the empirical covariance matrices of each domain, then (2) finding an orthogonal matrix that minimizes the maximum mean discrepancy between the projections of both domains. For arbitrary affine transformations, there is an inherent unidentifiability problem when performing unsupervised domain adaptation, which can be alleviated in the semi-supervised case. We show the effectiveness of our approach in first simulated data and then in a set of binary digit classification tasks (0 vs 1, 0, vs 2, ... 8 vs 9), obtaining improvements up to 48% accuracy when correcting for the domain shift in the data.

Chapter 5, based on *Style Transfer for Unsupervised Domain Adaptation in Ultrasound Image Segmentation*, discusses the application of style-transfer for correcting batch effects in ultrasound images. By assuming that a given imaging device generates images that are elements of a Julesz ensemble, we propose an unsupervised domain adaptation method that transfers the texture from the images of the target distribution to the images of the training distribution. We observed gains of up to 20%, in terms of Dice score, in the segmentation of ultrasound images of the hip. We expect this approach to be successful in cases where the discrepancy between the source and target domains is caused by differences in texture.

## 1.1 Contributions

This dissertation ...

- ... empirically shows, on three real-life datasets, (1) the effectiveness and advantages of using probabilistic labels over categorical labels, and (2) how to obtain these probabilities using medical expert knowledge.

Models trained with probabilistic labels were more accurate, and better calibrated.

- ... develops SIMLR, an interpretable model that uses machine learning to extend an SIR (Susceptible-Infected-Removed) epidemiological model for predicting the number of newly infected people, 1- to 4- weeks in advance. SIMLR achieves state-of-the-art results in data from the US and Canada.

- ... casts batch effects as an instance of domain-shift, and develops an unsupervised domain-shift adaptation algorithm for correcting discrepancies caused by arbitrary affine transformations. This approach led to improvements up to 48% accuracy in simulated data and in binary digit classification tasks.

- ... empirically explores applying style-transfer algorithms for unsupervised domain-shift adaptation. This approach achieved gains of up to 20%, in terms of Dice score, in the segmentation of ultrasound images of the hip when training a model with images acquired with one type of probe, and evaluating its performance on images from a different type.

# Chapter 2

# Probabilistic Labels

## 2.1 Introduction

Deep learning has proven useful in solving problems that involve structured inputs, such as images, voice, and video [39]. Those inputs tend to be high-dimensional, so machine learning models use many parameters (in the order of $10^4 - 10^7$) to provide accurate predictions. Fitting such large models usually requires datasets whose number of instances are at least in the same range as the number of parameters, complicating its use in many fields.

As explained in Chapter 1, it is important to find learning methodologies that are sample efficient, allow the incorporation of prior knowledge, and produce accurate and calibrated models. We argue that traditional learning paradigms in deep learning do not meet these characteristics, so we propose using *probabilistic labels* as an alternative way of achieving those objectives. The basic idea is: instead of using categorical labels as the target during the training procedure, use dense vectors whose $k$-th entry represents the probability of belonging to the $k$-th class. We anticipate that encoding the targets as real numbers, instead of just categorical labels, might compensate for the small number of training instances by providing more information per instance in the form of those probabilistic labels.

A key contribution of our approach is a mechanism to compute the probabilistic labels when we only have access to the categorical ones. We propose encoding the medical knowledge as probabilities. First, during the training phase, we extract from every image a set of features considered in the medical literature as relevant for making a diagnosis. Then, we build a probabilistic model over these features and the categorical labels. Next, for every training instance, we compute the probability of each class given the features. Finally, we use these probabilities as targets of a convolutional neural network whose inputs are the raw images. This approach is similar to model distillation [54], but instead of distilling a complex model, we "distill the medical knowledge".

To help motivate this work, below we discuss three natural questions: why use deep learning if the relevant features are known?, what if the probabilistic labels are misleading?, and is the extraction of the features worth the effort?

Even when medical experts can identify the relevant markers in the image, they still have to compute the features manually. Since these features are mostly visual, or require special software to perform measurements, it is unclear how to automate this process during inference. By combining deep learning with probabilistic labels, the network learns a good representation of the data using a limited amount of training instances. This representation removes the need of any manual computation of features during the inference process. In other words, a deep learning model can provide a prediction directly from the image, without the need of any human intervention.

We also incorporated a regularization parameter, $\lambda$, that controls the influence of the probabilistic labels during training. These labels help the algorithm to learn the model parameters accurately with fewer training instances, assuming the labels are a good approximation of the real probabilities. Even with misleading probabilistic labels, our approach can recover the true probabilities from the categorical labels, given a large enough dataset.

Our experiments suggest that the effort of providing the extra features during training is justified. We used first a simple toy dataset that exemplifies the advantages of training with probabilistic labels. Then, we used 3 real-world imaging datasets for the diagnosis of (respectively) hip dysplasia, glaucoma, and fatty liver. Using probabilistic labels not only improved the classification accuracy up to 22%, relative to using categorical labels, but it also produced models that are calibrated – *i.e.*, the output of the model can be interpreted as probabilities [45, 57].

Section 2.2 describes relevant literature for the problem of calibration and sample efficient learning. Section 2.3 describes probabilistic labels and justifies their use to train deep learning models. Section 2.4 compares the performance of models trained using probabilistic labels versus other training approaches. Finally, Section 2.5 highlights the important elements of this approach, emphasizing where it is expected to excel.

## 2.2 Foundations and related work

### 2.2.1 Soft Labels

Traditionally, the label ($y$) of each instance is encoded as a one-hot vector – *e.g.*, the encoding $h_i = [0, 1, 0]$ indicates that $x_i$ belongs to the second class, $y_i = 2$, since the 1 is located at the $2nd$ entry. We call this a *categorical label.* Note that categorical labels allocate all the probability mass to a single class, encouraging big differences between the largest logit and all others in networks that use the soft-max activation function in the output layer [111]. This is undesirable in applications like medical imaging, where many cases are "borderline", in that it is not clear to which class they belong. Creating an artificial gap between logits might then cause overfitting and reduce the ability of the network to adapt [111].

By contrast, *soft labels* encode the label of each instance as a vector of real values, whose $k$-th entry represents $p(Y = k \mid X = x) \in [0, 1]$ [33]. For example, the soft-label $s_i = [0.1, 0.7, 0.2]$ indicates that $p(Y = 2 \mid X = x_i) = 0.7$. By using real numbers instead of single bits, soft-labels provide to the learning algorithm extra information that often reduces the number of instances required to train a model [54], while improving the performance during inference [30, 33, 59].

The main challenge in using soft labels is their proper computation. Nguyen et al. (2011) proposed directly asking domain experts for their best estimates of $p(Y = k \mid X = x)$. Models trained with these soft labels learned more accurate classifiers, using fewer labeled instances, than classifiers trained with categorical labels [88]. One complication is that human experts struggle to give reliable and consistent estimates of the probabilities. One effective way of reducing this problem is to group the probabilities into bins [129]. However, this still relies on human estimates.

A different approach is the use a smoothing parameter that distributes a fraction of the probability mass, $\epsilon$, over all the possible classes (*e.g.*, if $\epsilon = 0.1$, then the label $[0, 1]$ becomes $[0.1, 0.9]$). This solution achieved an increase of 0.2% in accuracy on the ImageNet dataset [111]. Pereira et al. (2017) suggested a similar approach: Directly penalize "confident predictions" of a neural network by adding the negative entropy of the output to the negative log-likelihood during training [96]. This strategy, whose performance was similar to label smoothing, penalizes the allocation of all the probability mass on a single class at inference time. Note that these approaches apply the same smoothing to all the labels; however, Norouzi et al. (2016) empirically demonstrated that not all the classes should receive the same probability mass [92]. In fact, one can argue that arbitrarily penalizing confident predictions is not a good strategy in the medical domain, since there are cases when we want the classifier to have high confidence in the predictions.

A third strategy, which is based on work on model compression [6, 12], is to use model distillation [54, 86]. Here, we first train a complex model that outputs a vector of real numbers, whose $k$-th entry is interpreted as the probability of belonging to the $k$-th class. Then, train a second, simpler, model whose target is the output of the first model. This is a very effective approach, but it requires enough data to train that accurate complex model first. Unfortunately, for medical tasks, the scarcity of labeled data complicates using this solution.

The results obtained by the aforementioned approaches strongly argue for using soft labels in classification tasks, but they highlight two unresolved problems: (1) It is still not clear how to properly obtain the labels, and (2) these approaches ignore the original true labels, so unreliable soft-labels will lead to unreliable results. We propose using *probabilistic labels* to alleviate those problems. First, train a simple probabilistic model based on the categorical labels, whose features are manually extracted by medical experts. The predictions of this model, which encode the expert medical knowledge, can be treated as reliable and consistent soft labels. Next, "distill" the knowledge of this probabilistic model and transfer it to a deep neural network. This neural network will receive as an input a raw image, and uses as targets both, the soft-labels produced by the probabilistic model, and the original categorical labels. The influence of each label is determined by a regularization parameter $\lambda$, which can be determined using cross-validation.

### 2.2.2 Calibration

A probabilistic classifier is considered "calibrated" if the probabilities it returns are good estimates of the actual likelihood of an event [45, 47, 57]. For example, if a calibrated classifier predicts that the probability of having a disease is 30% for 10 individuals, then we would expect 3 of those individuals to actually have the disease. Calibration is particularly relevant for critical decision-making tasks. Common metrics to determine if a predictor with parameters $\theta$, $p_\theta(Y \mid X)$, outputs calibrated probabilities is the Hosmer-Lemeshow goodness-of-fit statistical test [57] and the expected calibration error [45].

Typically, the output of a neural network that uses a sigmoid or soft-max activation function in its last layer is interpreted as the probability of the classes given the input [39]. These activation functions indeed output values in $[0, 1]$ whose values add to 1; however, there is evidence that traditional learning approaches in modern neural networks lead to poorly calibrated models and therefore do not represent "real probabilities" [45].

The poor calibration problem becomes evident after analyzing the cross-entropy cost function, which is commonly used to train classifiers:

$$c\left(y, f(x)\right) \;=\; -\frac{1}{M}\sum_{i=1}^{M}\sum_{k=1}^{K} p(y_i = k \mid x_i)\;\log(f(x_i, k)) \tag{2.1}$$

where $M$ is the number of training instances, $K$ is the number of classes, and $f(x_i, k)$ is the predicted probability that $x_i$ belongs to the $k$-th class. Note that for a fixed $x$ and $k$, the prediction that minimizes the cost, when using categorical labels, is:

$$f(x,k) \;=\; \frac{1}{M_x}\sum_{i=1}^{M_x} I(y_i = k) \tag{2.2}$$

*i.e.*, the optimal prediction is the proportion of observations labeled with class $Y = y$ out of the total number of instances, $M_x$, where $X = x$ . By the law of large numbers, as $M \to \infty$, $f(x,k) \to p(Y = k \mid x)$; however, when the number of training instances is small, $f(x,k)$ might not be a good approximation of $p(Y = k \mid x)$, meaning the predictions are not calibrated.

A second problem for calibration arises when the inputs are high-dimensional. In gray-scale medical images, most of the pixels take values in the interval $[0, 255]$. Therefore, the sample space is $[0, 255]^p \times \{0, 1, \ldots, K\}$, where $p = |x|$ is the number of pixels, which is typically around $10^4$. The sample space immediately highlights the difficulties that any learner has to learn $p(Y \mid X)$: It needs a very large number of instances to approximate this probability directly from the images and the categorical labels.

One way of improving the calibration of traditional machine learning models (linear models, decision trees, etc.) is via Platt scaling or isotonic regression [89]. Similarly, using a temperature parameter helps the calibration on modern neural networks [45]. These simple, yet effective, methods improve the calibration of the predictions. However, none of these methods improve the accuracy of the models –*i.e.*, they only modify the *confidence* in the predicted class for a novel instance. Here, we empirically show that, by using *probabilistic labels*, it is possible improve both the calibration of the predictions and the classification performance.

## 2.3 Probabilistic labels

The high dimensionality of images poses important challenges for learning calibrated and accurate predictors [27], so dimensionality reduction is a common step in the machine learning pipeline [46]. Similarly, medical experts do not analyze the images at the pixel level. Instead, they are trained to identify relevant features in the images, and then combine those features to produce the diagnosis.

The idea behind probabilistic labels is to first obtain the relevant features, $Z(X)$, from raw images $X$. Then, use the categorical labels along with a probabilistic model to estimate $p(\ Y \mid Z(X)\ )$; see Section 2.3.1. Since those features are assumed to be a good representation of the image, then it is valid to assume that $p(Y \mid Z(X)) \approx p(Y \mid X)$. Since $|Z(X)| \ll |X|$ –*i.e.*, $Z(X)$ has fewer features than the raw $X$, we expect the estimation of $p(Y \mid Z(X))$ to be more accurate than the one of $p(Y \mid X)$, given the same number of training instances.

The last step is to "distill" the medical expert knowledge encoded in the probabilistic model. To do this, train a deep learning model using the raw images $X$ as inputs, and $p(Y \mid Z(X)\ )$ as targets. The learning problem is then to learn function $q(x) = p(\,y\,|\,z(x)\,)\ \in [0,1]^K$ that maps a medical image, $x$, to the probability of being classified as each of the $K$ classes.

To learn such a function we apply a learning algorithm, $L(\cdot)$ to a labeled training set with $n$ instances $D = \{[x_1,\, p(\,y_1\,|\,z_1\,)],\ \ldots,\ [x_n,\, p(\,y_n\,|\,z_n\,)]\}$ to get an estimate of the function, $\hat{q} = L(D)$. It is then possible to make predictions on new instances $\hat{y}_{new} = \hat{q}(x_{new})$. Note that once the model has been learned, the only input is a raw image, and it is no longer necessary to compute the feature vector $z$.

Given the success of deep learning models on images, we chose the learning algorithm $L(\cdot)$ to be a convolutional neural network with fully connected layers with a softmax activation function in the last layer. The detailed architecture will be described in the next section. Since this target vector is now a probability distribution, it makes sense to use a loss function that measures the distance between the target distribution and the predicted outputs of the network.

A common measure for distance between probability distributions is the KL divergence:

$$D_{KL}(\,P\,||\,Q_\theta\,) \ = \ -\sum_{y \in Y} p(\,y\,|\,x\,)\, \log\left(\frac{q_\theta(y \mid x)}{p(\,y\,|\,x\,)}\right) \tag{2.3}$$

where $p(\,y\,|\,x\,)$ is the real conditional distribution of the labels given the inputs, and $Q_\theta(y \mid x)$ is the probability distribution predicted by a model parameterized by $\theta, q_\theta(\cdot)$. Minimizing the KL divergence between the distributions is equivalent to minimizing the negative cross entropy:

$$\theta^* \;=\; \arg\min_\theta - \sum_{i=1}^{m}\sum_{k=1}^{K} p(Y=k|x_i)\;\log\left(q_\theta(x_i,k)\right) \tag{2.4}$$

Note that this objective is identical to the one we use for training with categorical labels; the only difference is that instead of using the indicator function as target, $p(\,Y=k\,|\,x\,) = I(Y=k|x)$, we use the *probabilistic label* $p(\,Y=k\,|\,x\,) = p(\,Y=k\,|\,z(x)\,)$.

The quality of the learned model will depend on the quality of the estimation of the probabilistic label. Suppose that for a fixed $x$ the real probability is $p(y=1\mid x)=0.75$, but we only have access to 5 instances (2 positive, 3 negative). Using the traditional approach, the model that optimizes Eq. 2.4 will converge to $f_\theta(x)=0.4$. If our guess of the probabilistic label is $y_{pr}=0.73$, then we can expect a better performance by using the probabilistic label. Note that although the evidence given by the categorical labels indicates that $f_\theta(x)=0.4$, our "confidence" in that evidence is small, due to the small number of instances.

The opposite effect can also happen. When the estimation of the probabilistic label is incorrect, the model will converge to that label regardless of the evidence given by the categorical labels. Ideally, we should find a balance between the influence of categorical and probabilistic labels. We propose to achieve this behavior by training our model in two steps: (1) Let $\theta_p$ be the parameters of a model $q_{\theta_p}(x)$ trained to optimize Eq. 2.4 using the probabilistic labels $p(\,Y\,|\,z_i\,)$. (2) Use the weights $\theta_p$ as a prior to learn a second model, with parameters $\theta$, that uses the *categorical labels* and optimizes the regularized cross-entropy:

$$\theta^* = \arg\min_\theta - \sum_{i=1}^{m}\sum_{k=1}^{K} I(Y=k|x_i)\log\left(q_\theta(x_i,k)\right) + \lambda||\theta-\theta_p||_2^2 \tag{2.5}$$

Intuitively, this loss function penalizes deviations from the model learned with probabilistic labels. Note that as the number of instances, $m$, increases, the influence of the regularization term decreases. It can be shown that

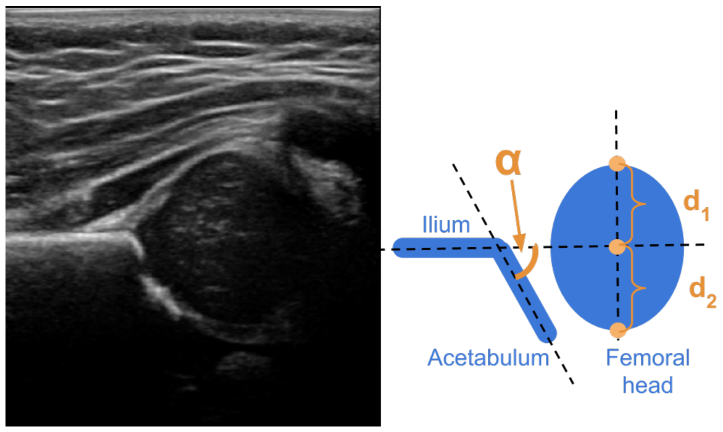


Figure 2.1: **(Left)** Typical 2D ultrasound image of the hip. **(Right)** The structures of interest: the acetabulum, the ilium and the femoral head, as well as the angle between the acetabulum and ilium ($\alpha$), and the information for computing the coverage: $c = \frac{d_2}{d_1+d_2}$

this regularized loss function is equivalent to setting a Gaussian prior on the weights [87]. The mean of this Gaussian prior is $\theta_p$, and the covariance matrix is $\frac{1}{2\lambda}I$, where $I$ is the identity matrix. Therefore, a high value of $\lambda$ means that the confidence in the probabilistic labels is high. However, as the number of instances increases, the influence of the prior decreases. In practice, we can set the value of the regularization parameter $\lambda$ using cross-validation. Although we describe this algorithm in two steps, in practice both models are implemented under a single routine.

### 2.3.1 Example: Hip dysplasia

Developmental dysplasia of the hip is a deformity of the hip joint at birth that affects close to 3% of infants [51]. Ultrasound imaging is one way to diagnose this condition. To do so, the medical expert measures the angle $\alpha$ between the acetabulum and ilium, and the coverage $c$ (ratio between the two segments $d_1$ and $d_2$); as shown in Figure 2.1 [41, 50, 51].

We can encode this knowledge as a simple probabilistic model where the random variable $Y \in \{0, 1\}$ encodes healthy people as $y = 0$ and people with hip dysplasia as $y = 1$, and the random variable $Z \in [0, \pi] \times \mathbb{R}$ is a vector that contains the pair of computed features: (angle, coverage). The sample space for this new model is $[0, \pi] \times \mathbb{R} \times \{0, 1\}$, where the last bit is the label. Note

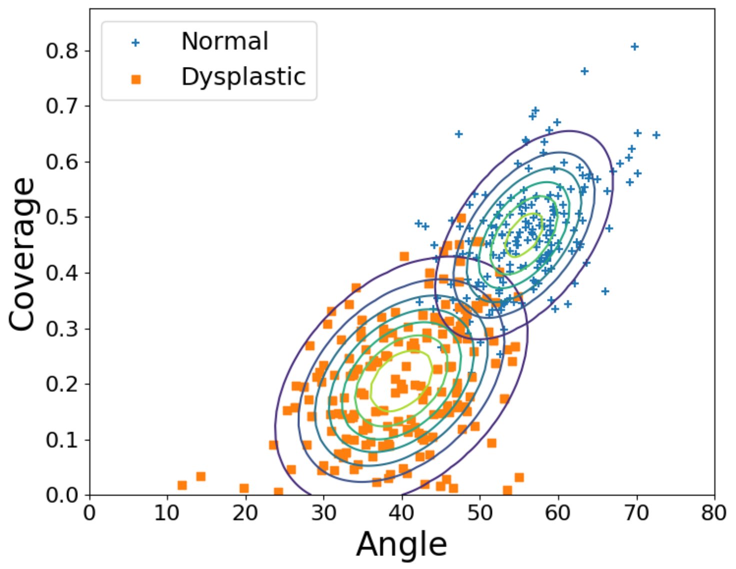


Figure 2.2: Gaussian distributions learned for images classified as normal or dysplastic, based on the angle ($\alpha$) and coverage ($c$)

that this is much simpler than the sample space of the entire image.

Although the proper probability distributions for modelling angles and ratios are the Von-Mises distribution and a ratio distribution, we assume that a bi-variate Gaussian with full covariance matrix is a good approximation for $P(Z \mid Y)$. Figure 2.2 shows a scatter plot of the values for $\alpha$ and $c$ for both, people with dysplasia and healthy controls.

The computation of the soft labels $p(\,Y\,|\,Z = z_i\,)$ is given by Eq 2.6, where $p(\,Z\,|\,y = i\,) \sim \mathcal{N}(\mu_i, \Sigma_i)$, and $p(\,Y = i\,)$ is the prior probability of having (or not having) dysplasia. It is straightforward to estimate the parameters $\mu_i$, $\Sigma_i$ from the data, while the prior probabilities $p(\,Y = i\,)$ can be extracted from medical literature. Note that this prior will play a fundamental role in determining if the probabilities are calibrated, as its value should reflect the probability that we expect to observe *where the model will be used*.

$$p(\,Y = 1\,|\,z_i\,) \;=\; \frac{p(\,z_i\,|\,Y = 1\,)\,p(\,Y = 1\,)}{\sum_{j\in\{0,1\}} p(\,z_i\,|\,Y = j\,)\,p(\,Y = j\,)} \tag{2.6}$$

The last step consists in training a model using the soft labels obtained with Eq. 2.6, along with the cost functions described by Eq. 2.4 and Eq. 2.5. Note that, although we modelled the features as a bi-variate Gaussian, this approach is very general and admits other distributions as well. Probabilistic graphical models offer a good set of tools for modelling distributions with a larger number of features, or that require a combination of discrete and continuous covariates [55, 72, 74].

## 2.4 Experiments and results

### 2.4.1 Experiment 1: Simulated Data

We generated 2,000 instances from a mixture of 2 bi-variate Gaussians (1,000 from each Gaussian) and used it as a test set. The parameters of the Gaussians were: $\mu_1 = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$, $\Sigma_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$. Additionally, we generated 60 instances for training purposes (30 from each Gaussian). The machine learning task was to use the training instances to build a classifier that assigned every instnace to one of the two possible classes: Gaussian 1, or Gaussian 2. Figure 2.4 (right) shows the test instances generated for the experiments.

We trained models with different numbers of instances, starting with 2 (one from each Gaussian), and progressively increased the number until we used the 60 instances. We compared the accuracy of the models learned (respectively) with categorical, correct and incorrect probabilistic labels, as well the regularized categorical/probabilistic labels. Incorrect probabilistic labels are soft labels that do not represent true probabilities, but still are in [0,1], and all the entries add up to 1. We repeated these experiments 100 times, using logistic regression as the classifier, and show the average performance in Figure 2.3 (left). The second critical consideration is learning a calibrated classifier, a task whose complexity increases with unbalanced datasets. In a second experiment, we kept 10 training instances from Gaussian 1 and progressively changed the number of training instances of the second Gaussian from 1 to 10. Figure 2.3 (right) shows the expected calibration error for different imbalance ratios.

Three things are important about this figure: (1) The probabilistic labels lead to more accurate classifiers when the number of instances is small. As

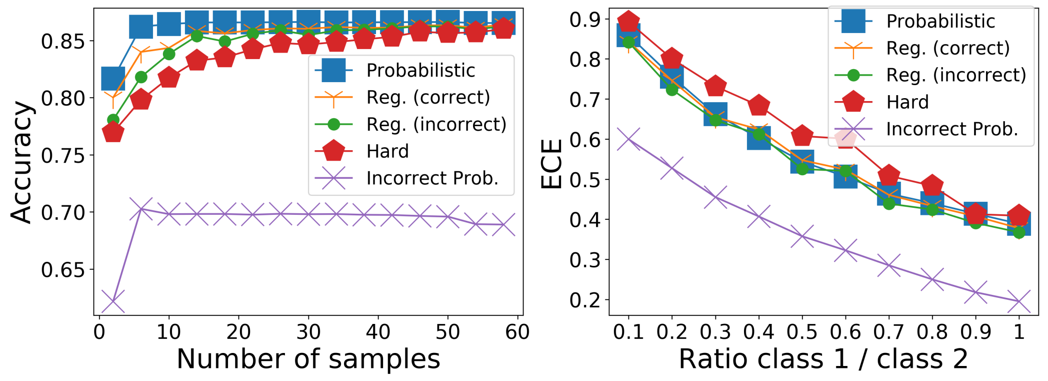


Figure 2.3: Accuracy as a function of the number of training instances (left). Calibration as a function of the class imbalance (right).

this number increases, the model trained with categorical labels can also learn the real probabilities. (2) Providing incorrect probabilistic labels, and training the model using exclusively those labels, is worse than providing the categorical labels. However, the regularized categorical/probabilistic labels allow the model to converge to the real probabilities, even if the probabilistic labels are misleading. (3) Probabilistic labels improve calibration, relative to categorical labels, even in the presence of class imbalance.

Figure 2.4 illustrates the change in the decision boundary changes when using probabilistic labels and a small training set. The boundary decision learnt by the categorical labels tries to separate the 4 training instances perfectly (Figure 2.4, left); however, given the small training dataset the decision boundary does not generalize well (Figure 2.4, right). Probabilistic labels, on the hand, provide additional information that inform the learning algorithm "how close" each training instance is to the decision boundary. The decision boundary learnt by the probabilistic labels on Figure 2.4 (left) does not perfectly classify the training instances; however, it generalizes much better than the one learnt with categorical labels. This simple experiment illustrates why we expect the probabilistic labels to generalize better when the number of training instances is small.

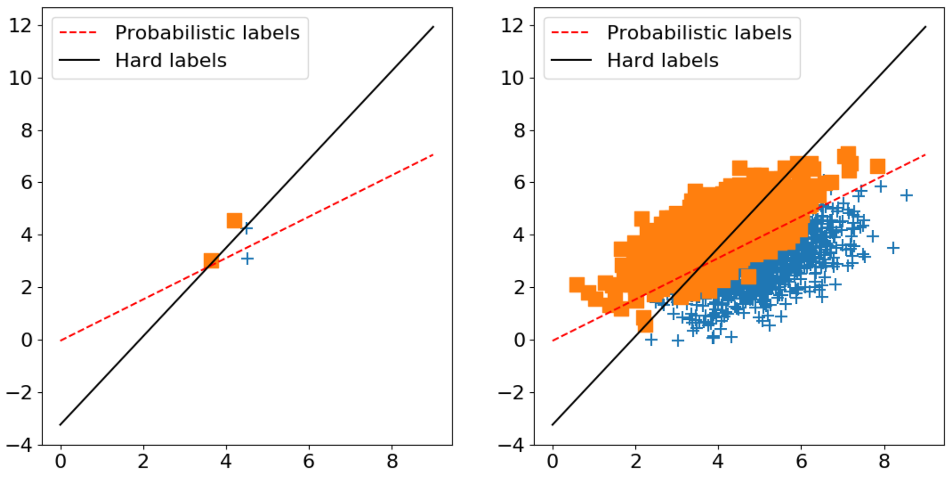


Figure 2.4: Decision boundaries learnt with categorical labels and soft labels in the training set (left), and how well they generalize to the test set (right).

### 2.4.2 Experiment 2: Hip Dysplasia

The machine learning task is to train a model that, given an ultrasound image of the hip, identifies if it should be considered as normal or dysplastic. We used a private dataset collected from a multi-year clinical study of developmental dysplasia of the hip, which contains 685 labeled, conventional, b-mode ultrasound images of the hip. A clinical expert labeled 342 of the images as dysplastic, while the remaining were considered normal.

We divided this dataset into a training set (70 % of the individuals = 429 images) and a hold-out set (30% of the individuals = 256 images). The hold-out set was used exclusively for testing purposes. After computing the soft labels using the procedure described in Section 2.3.1, we trained a deep learning model that receives an image as an input and whose output is the probability of being diagnosed with dysplasia.

The deep neural network consisted of 5 layers (with 32, 64, 128, 256, and 512 3-by-3 convolutional filters). We added 2 fully connected layer with 1000 neurons in each hidden layer, and a single neuron in the output layer. The network used ReLU units as the activation function in the intermediate layers, and a sigmoid function in the output layer. The input to the network were ultrasound images resized to 128 by 128 pixels.

We compared four different training scenarios: with categorical labels,

Table 2.1: Accuracy (threshold of 0.5 in the predictions) and area under the ROC curve (AUC) for performance (higher is better); and HL-Statistic and the expected calibration error (ECE) for calibration (lower is better)

| **Model** | **Dysplasia** | | | |
|---|---|---|---|---|
| **Labels** | **Cat** | **Soft** | **Prob** | **Reg** |
| **Accuracy** | 74% | 68% | 80% | **83%** |
| **AUC** | 0.82 | 0.74 | **0.87** | **0.87** |
| **HL Stat.** | $> 100$ | 59 | 12.9 | **9.2** |
| **ECE** | 0.56 | **0.34** | 0.41 | 0.46 |
| **Model** | **Fatty liver** | | | |
| **Labels** | **Cat** | **Soft** | **Prob** | **Reg** |
| **Accuracy** | **90%** | 85% | 81% | **89%** |
| **AUC** | **0.97** | 0.94 | 0.94 | **0.97** |
| **HL Stat.** | $> 100$ | 17 | **8.5** | 12.9 |
| **ECE** | 0.53 | **0.41** | **0.41** | 0.48 |
| **Model** | **Glaucoma** | | | |
| **Labels** | **Cat** | **Soft** | **Prob** | **Reg** |
| **Accuracy** | 55% | 59% | 69% | **77%** |
| **AUC** | 0.66 | 0.65 | 0.79 | **0.83** |
| **HL Stat.** | $> 100$ | 16.5 | **12.8** | 15.3 |
| **ECE** | 0.50 | 0.32 | **0.26** | 0.30 |

soft labels, probabilistic labels, and regularized categorical/probabilistic labels. For the soft labels experiments, we followed the approach proposed by Szegedy et al. (2016), setting $\epsilon = 0.1$ [111] We evaluated the performance exclusively on the hold-out set measuring the classification accuracy, the area under the ROC curve, the HL-statistic, and the expected calibration error. Table 2.1 shows the results.

### 2.4.3 Experiment 3: Fatty Liver

The machine learning task is to produce a model that predicts if an ultrasound image of the liver (see Figure 2.5, right) should be diagnosed as fatty, or normal. We used a private dataset that contains 505 images, with labels made by an expert radiologist. We placed 353 (70%) in the training set, and the remaining 152 in the hold out set. The percentage of normal cases in both sets was 62%.

Similarly to the experiment with hip dysplasia, we first trained a probabilis-

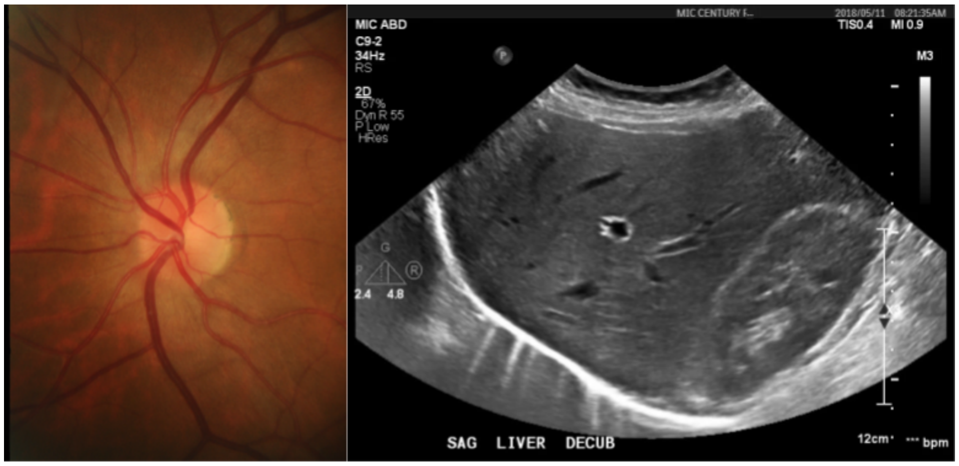


Figure 2.5: Typical images used for the diagnosis of glaucoma (left) and fatty liver (right).

tic model to encode the medical knowledge. Hamer et al. (2006) reports that the diagnosis of fatty liver depends on the difference in echogenicity among the liver, diaphragm, and the periportal zone [49] – more precisely, an increased hepatic echogenicity that obscures the periportal and diaphragm echogenicity suggest a liver is fatty [70].

We manually defined regions of interest over the relevant anatomical parts in the images, and extracted their mean pixel intensity. The vector $Z = [m_1, m_2, m_3]$ contains the mean intensity value of the regions of interest extracted (respectively) from liver, diaphragm and periportal zone. This time, we modelled $P(Y \mid Z) = \sigma(Z^T\theta)$, where $\sigma(\cdot)$ is the sigmoid function. We then use the probabilistic labels $P(Y \mid Z)$ to train a network with the same architecture described in Section 2.4.2, and perform the comparison among models trained with categorical, soft, and probabilistic labels. Table 2.1 shows the results.

### 2.4.4 Experiment 4: Glaucoma

Here, we wanted to learn a model that classifiers the fundus image of a retina (Figure 2.5, left) as healthy, or as suspicious of glaucoma. We used the publicly available dataset RIM-ONE r3 [28], which contains 85 images classified as normal, and 74 as suspects of glaucoma. We used 70% of the data for training purposes, while the remaining 30% was used as hold-out set. We kept the same healthy over glaucoma ratio in both datasets. Besides the diagnosis, which we used as ground truth, the dataset contains the masks of the disc and the cup masks of the optic nerve.

MacCormick et al. (2019) identifies the vertical and horizontal cup-to-disc-ratio as an important feature for the diagnosis of glaucoma [82]. The vector $Z = [r_1, r_2] \in \mathbb{R}^2$ contains these vertical and horizontal ratios. We assumed that the data comes from a mixture of 2 bi-variate Gaussians (one per each class) and, after learning the parameters of the probabilistic model, we used Eq. 2.6 to compute the probabilistic labels.

We used the same network architecture than in previous experiments, and trained models with the same types of labels: categorical, soft, and probabilistic. Table 2.1 shows the results of the experiments.

## 2.5 Discussion

Table 2.1 (Dysplasia, Glaucoma) shows that the probabilistic labels, and the regularized categorical/probabilistic approach greatly increases the accuracy of the classifiers when compared with the rest of the strategies. They also improve the calibration of the predicted outputs. It is important to highlight that all models were trained with the same architecture, and exactly the same training instances. The only change was the labels used during the training procedure.

These results suggests that the network indeed benefits from labels that encode how close a given instance is to the decision boundary. On the other hand, the simple soft labels ($y \in \{0.1, 0.9\}$), help to improve the calibration of the outputs; however, they do not improve the classification accuracy, suggesting that simply penalizing high confidence in the predictions is not enough to achieve good results.

The liver dataset in Table 2.1 shows an interesting case. The accuracy achieved by the model trained with categorical labels is clearly superior to the one that used just probabilistic labels. As mentioned in Section 2.3, when

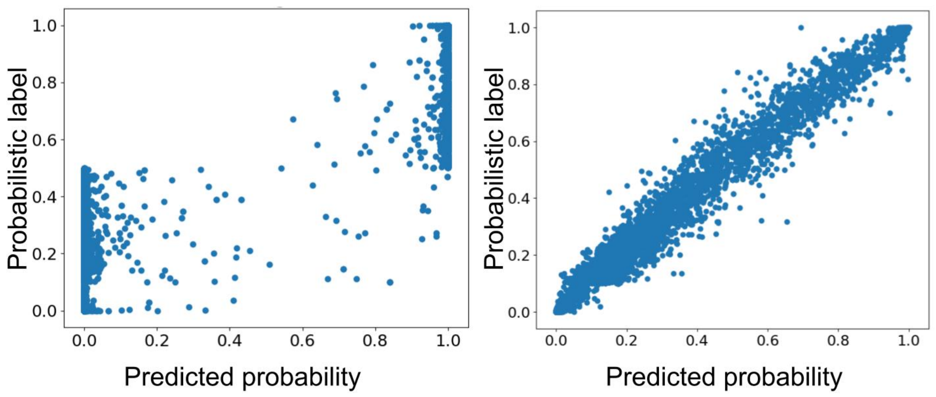


Figure 2.6: Predicted values vs probabilities of a model trained with categorical labels (left) and probabilistic labels (right) for the hip dysplasia experiment.

the probabilistic labels are misleading, the model will converge to the wrong probabilities, which in turn increase the error in the predictions. This is not surprising, since the problem of diagnosis of fatty liver is very subjective in nature, and it is not clear what are the biomarkers that allow for an accurate diagnosis [109].

Despite the complexity of obtaining reliable probabilistic labels for the fatty liver task, the regularized model achieves almost the same performance as with the categorical labels. This is a strong argument for the computation of probabilistic labels. The potential gains are important if they are reliable (9% and 22% in our experiment), while the potential losses are small (1% in our experiments). Although computing the probabilistic labels require some manual measurements during training, the inference part is completely automated and does not require any manually extracted features.

Besides accuracy, for sensitive tasks, it is desirable to have models that not only make accurate predictions, but also that express their confidence on the predicted label in the form of probabilities. Our experiments indicate that the traditional training procedures for deep neural nets produce models whose real-valued output might not correspond to the probability of belonging to a specific class. Probabilistic labels are an alternative that not only improves the calibration of the outputs, but also improve the accuracy.

Figure 2.6 provides insight on why the model trained with the probabilistic labels is better calibrated. When using categorical-labels, the deep learning models are penalized for not outputting 1 for the correct class, and 0 everywhere else. Naturally, their output tends to be closer to those values, so they tend to assign a "high confidence" to almost all their predictions. This is problematic for tasks where there is no clear boundary between the classes, such as medical diagnosis. Probabilistic labels, on the other side, encourage the model to make predictions in the entire range $[0, 1]$, assigning different degrees of confidence to the predictions.

The results from the different datasets strongly argue for the incorporation of probabilistic labels as priors when training classifiers. These labels allow for a more sample-efficient learning, since models using probabilistic labels provide equal or better accuracy than the ones trained with the traditional categorical labels. Additionally the output of models that use probabilistic labels are better calibrated, providing a straightforward interpretation of the predictions as probabilities. The potential improvement in performance justifies the extra annotations needed during training when the size of the dataset is small, and when there is expert knowledge that can be leveraged during training. We anticipate that this approach will apply to any other tasks that maps high-dimensional inputs to categorical outputs, and where the features that determine the class can be encoded into a probabilistic model.

# Chapter 3

# Combining Machine Learning and Epidemiological Models

## 3.1 Introduction

In prediction problems like time-series forecasting, probabilistic labels do not arise as naturally as in the case of image-based medical diagnosis. There are, however, other ways to incorporate domain specific knowledge. In this chapter, we combine an SIR epidemiological model with machine learning to predict the number of new reported infections of COVID-19 1 to 4 weeks in advance. The dynamics of infectious diseases depends not only on the nature of the virus, but also on policy decisions taken at government level. Our proposed model tracks and predicts changes on these policies to improve the accuracy of the predictions.

Using such prior models – here epidemiological models– is particularly important when the available data is scarce [105]. At the same time, machine learning models need to acknowledge that the reported data on COVID-19 is imperfect [9, 60]. Using probabilistic graphical models allows SIMLR to account for this uncertainty on the data, and the probability tables associated with this graphical model can be manually modified to adapt SIMLR to the specific characteristics of a region.

Since its identification in December 2019, COVID-19 has posed critical challenges for the public health and economies of essentially every country in the world [22, 26, 77]. Government officials have taken a wide range of measures in an effort to contain this pandemic, including closing schools and workplaces, setting restrictions on air travel, and establishing stay at home requirements [48]. Accurately forecasting the number of new infected people in the short and medium term is critical for the timely decisions about policies and for the proper allocation of medical resources [5, 76].

There are three basic approaches for predicting the dynamics of an epidemic: compartmental models, statistical methods, and ML-based methods [5, 124]. **Compartmental models** subdivide a population into mutually exclusive categories, with a set of dynamical equations that explain the transitions among categories [10]. The *Susceptible-Infected-Removed* (SIR) model [68] is a common choice for the modelling of infectious diseases. **Statistical methods** extract general statistics from the data to fit mathematical models that explain the evolution of the epidemic [76]. Finally, **ML-based methods** use machine learning algorithms to analyze historical data and find patterns that lead to accurate predictions of the number of new infected people [124, 101].

Arguably, when any approach is used to make high-stake decisions, it is important that it be not just accurate, but also interpretable: It should give the decision-maker enough information to justify the recommendation [104]. Here, we propose SIMLR, which is an interpretable probabilistic graphical model (PGM) that combines compartmental models and ML-based methods. As its name suggests, it incorporates machine learning (ML) within an SIR model. This combines the strength of curve fitting models that allow accurate predictions in the short-term, involving many features, with mechanistic models that allow to extend the range to predictions in the medium and long terms. [56]

SIMLR uses a *mixture of experts* approach [61], where the contribution of each *expert* to the final forecast depends on the changes in the government policies implemented at various earlier time points. When there is no recent change in policies (2 to 4 weeks before the week to be predicted), SIMLR relies on an SIR model with time-varying parameters that are fitted using machine learning methods. When a change in policy occurs, SIMLR instead relies on a simpler model that predicts that the new number of infections will remain constant. Note that forecasting the number of new infections 1 and 2 weeks in advance ($\Delta I_1$ and $\Delta I_2$) is relatively easy as SIMLR knows, at the time of the prediction, whether the policy has changed recently. However, for 3- or 4-week forecasts ($\Delta I_3$ and $\Delta I_4$), our model needs to estimate the likelihood of a future change of policy. SIMLR incorporates prior domain knowledge to estimate such policy-change probabilities.

This chapter makes three important contributions. (1) It empirically shows that an SIR model with time-varying parameters can describe the complex dynamics of COVID-19. (2) It describes an interpretable model that predicts the new number of infections 1 to 4 weeks in advance, achieving state-of-the-art results, in terms of mean absolute percentage error (MAPE), on data from Canada and the United States. (3) It presents a machine learning model that incorporates the uncertainty of the input data and can be tailored to the specific situations of a particular region.

The rest of Section 3.1 describes the related work and the basics of the SIR compartmental model. Section 3.2 then describes in detail our proposed SIMLR approach. Section 3.3 shows the results of the predicting the number of new infections in the United States and provinces of Canada.

### 3.1.1 Basic SIR Model

The *Susceptible-Infected-Removed* (SIR) compartmental model [68] is a mathematical model of infectious disease dynamics that divide the population into 3 disjoint groups [10]. *Susceptible (S)* refers to the set of people who have never been infected but can acquire the disease. *Infected (I)* refers to the set of people who have and can transmit the infection. *Removed (R)* refers to the people who have either recovered or died from the infection and cannot transmit the disease anymore. This model is defined by the differential equations:

$$\frac{dS}{dt} = -\frac{\beta S(t)I(t)}{N}, \quad \frac{dI}{dt} = \frac{\beta S(t)I(t)}{N} - \gamma I(t), \quad \frac{dR}{dt} = \gamma I(t) \qquad (3.1)$$

SIR assumes an homogeneous and constant population, and it is fully defined by the parameters $\beta$ (transmission rate) and $\gamma$ (recovery rate). The intuition behind this model is that every infected patient gets in contact with

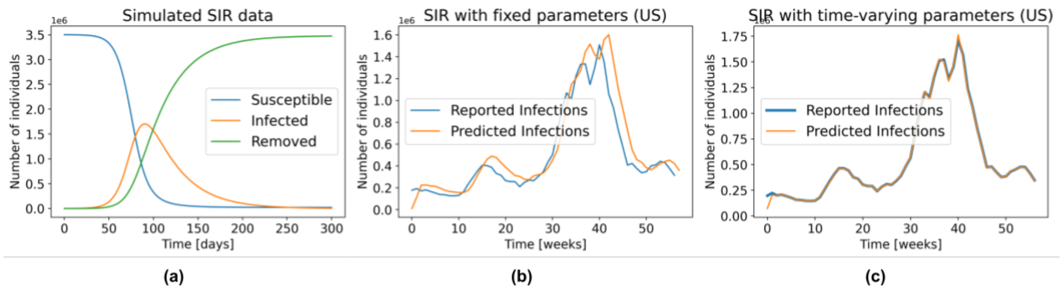


Figure 3.1: **(a)** General behaviour of the SIR model. **(b)** The number of infections predicted by the SIR model with fixed parameters, fitted to the US data for 1 week prediction. **(c)** Similar to (b), but with time-varying parameters.

$\beta$ people. Since only the susceptible people can become infected, the chance of interacting with a susceptible person is simply the proportion of susceptible people in the entire population, $N = S + I + R$. Likewise, at every time point, $\gamma$ proportion of the infected people is removed from the system. Figure 3.1(a) depicts the general behaviour of an SIR model.

### 3.1.2 Related Work

The main idea behind combining compartmental models with machine learning is to replace the fixed parameters of the former with time-varying parameters that can be learned from data [3, 16, 76, 78]. However, most of the approaches focus on finding the parameters that can explain the past data, and not on predicting the number of newly infected people. Although those approaches are useful for obtaining insight into the dynamics of the disease, it does not mean that those parameters will accurately predict the behaviour in the future.

Particularly relevant to our approach is the work by Arik et al. [5], who used latent variables and autoencoders to model extra compartments in an extended *Susceptible-Exposed-Infected-Removed* (SEIR) model. Those additional compartments bring further insight into how the disease impacts the population [123, 71]; however, our experiments suggest that they are not needed for an accurate prediction of the number of new infections. One limitation of their model is a decrease in performance when the trend in the number of new infections changes. We hypothesize that those changes in trend are related to the government policies that are in place at a specific point in time. SIMLR is able to capture those changes by tracking the policies implemented at the government level.

A different line of work replaces epidemiological models with machine learning methods to directly predict the number of new infections [62, 66, 85, 94]. Importantly, Yeung et al. [130] added non-pharmaceutical interventions (policies) as features in their models; however, their approach is limited to make predictions up to 2 weeks in advance, since information about the policies that will be implemented in the future is not available at inference time. Our SIMLR approach differs by being interpretable and also by forecasting policy changes, which allows it to extend the horizon of the $\Delta I$ predictions.

There are many models that attempt to predict the evolution of the COVID-19 epidemic. The Centers for Disease Control and Prevention (CDC) in the United States allows different research teams across the globe to submit their forecasts of the number of cases and deaths 1 to 8 weeks in advance. [18]. More than 100 teams have submitted at least one prediction to this competition. We compare SIMLR with all of the models that made predictions 1 to 4 weeks in advance in the same time span as our study.

## 3.2 Materials and Methods

We view SIMLR as a probabilistic graphical model that uses a mixture of experts approach to forecast the number of new COVID-19 infections, 1 to 4 weeks in advance. Figure 3.2 shows the intuition behind SIMLR. Changes in the government policies are likely to modify the trend of the number of new infections. We assume that stronger policies are likely to decrease the number of new infections, while the opposite effect is likely to occur when relaxing the policies. These changes are reflected as a change in the parameters of the SIR model. Using those parameters, we can then predict the number of new infections, then use that to compute the likelihood of observing other new policy changes in the short term.

While Figure 3.2 is an schematic diagram used for pedagogical purposes; Figure 3.3 depicts the formal probabilistic graphical model , as a plate model, that we use to estimate the parameters of the SIR model, the number of new infections, and the likelihood of observing changes in policies 1 to 4 weeks in advance. The blue nodes are *estimated* at every time point, while the values of the green nodes are either *known* as part of the historical data, or *inferred* in a previous time point. The random variables are assumed to have the following distributions:

$$\begin{array}{rcll}
\mathrm{CT}_{t+1} \mid & \{\mathrm{CP}_{t-\tau}\}_{\tau\in\{1,2,3\}} & \sim & Cat_{K\in\{-1,0,1\}}(\theta_{CT}) \\
\beta_{t+1} \mid & \{\beta_{t-\tau}\}_{\tau\in\{0,1,2\}},\ \mathrm{CT}_{t+1} & \sim & \mathcal{N}(\mu_\beta, \Sigma_\beta) \\
\gamma_{t+1} \mid & \{\gamma_{t-\tau}\}_{\tau\in\{0,1,2\}}, \mathrm{CT}_{t+1} & \sim & \mathcal{N}(\mu_\gamma, \Sigma_\gamma) \\
\mathrm{SIR}_{t+1} \mid & \beta_{t+1},\ \gamma_{t+1} & \sim & \mathcal{N}(\mu_{SIR}, \Sigma_{SIR}) \\
U_t \mid & \{\mathrm{SIR}_{t-\tau}\}_{\tau\in\{0,1,2\}} & \sim & Cat_{K\in\{-1,0,1\}}(\theta_U) \\
O_t \mid & W_t & \sim & Cat_{K\in\{0,1\}}(\theta_O) \\
\mathrm{CP}_{t+1} \mid & O_t,\ U_t & \sim & Cat_{K\in\{-1,0,1\}}(\theta_{CP})
\end{array} \tag{3.2}$$

where $t$ indexes the current week, $SIR_t = [S_t, I_t, R_t]$, $\mu_{SIR} \in \mathbb{R}^3$ is given below by Equation 3.3, $\mu_\beta = (\alpha_{0,CT_{t+1}}) + (\alpha_{1,CT_{t+1}})\beta_{t-1} + (\alpha_{2,CT_{t+1}})\beta_{t-2} + (\alpha_{3,CT_{t+1}})\beta_{t-3}$ and $\mu_\gamma = (\omega_{0,CT_{t+1}}) + (\omega_{1,CT_{t+1}})\gamma_{t-1} + (\omega_{2,CT_{t+1}})\gamma_{t-2} + (\omega_{3,CT_{t+1}})\gamma_{t-3}$ are linear combinations of the three previous values of $\beta$ and $\gamma$ (respectively). The coefficients of those linear combinations depend on the value of the random variable $\mathrm{CT}_{t+1}$. We did not specify a distribution for the node $\texttt{New\_infections}_{t+1}$ because its value is deterministically computed as $S_t - S_{t+1}$.

Informally, the assignment $\mathrm{CT}_t = -1$ means that we expect a change in trend from an increasing number of infections to a decreasing one. The opposite happens when $\mathrm{CT}_t = 1$, while $\mathrm{CT}_t = 0$ means that we expect the population to follow the current trend (either increasing or decreasing). We assume these changes in trend depend on changes in the government policies 2 to 4 weeks prior to the week of our forecast – *e.g.*, we use $\{\mathrm{CT}_{t-3}, \mathrm{CT}_{t-2}, \mathrm{CT}_{t-1}\}$ when predicting the number of new infections at $t+1$, $\Delta I_{t+1}$; and we need $\{\mathrm{CT}_t, \mathrm{CT}_{t+1}, \mathrm{CT}_{t+2}\}$ when predicting $\Delta I_{t+4}$. Note that, at time $t$, we will not know $\mathrm{CT}_{t+1}$ nor $\mathrm{CT}_{t+2}$. We chose this interval based on the assumption that the incubation period of the virus is 2 weeks.

The status of $\mathrm{CT}_{t+1}$ defines the coefficients that relate $\beta_{t+1}$ and $\gamma_{t+1}$ with their three previous values $\beta_t, \beta_{t-1}, \beta_{t-2}$ and $\gamma_t, \gamma_{t-1}, \gamma_{t-2}$, respectively. Since

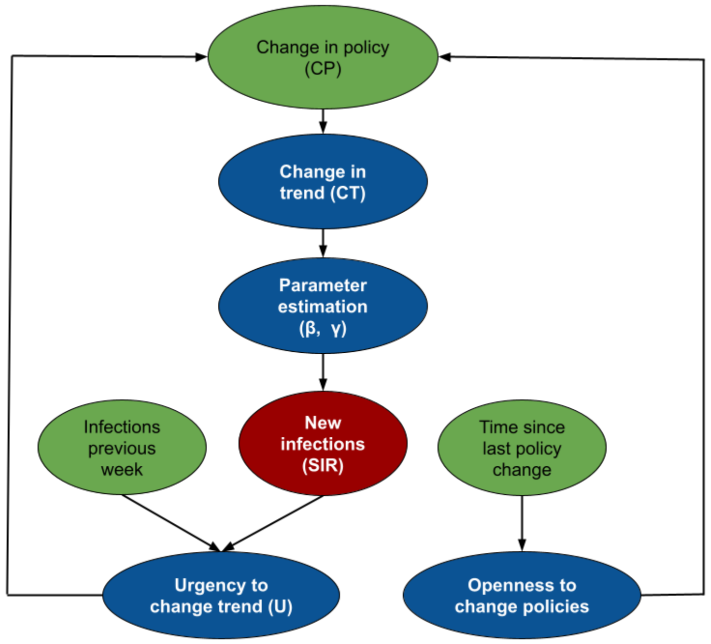

Change in policy (CP)
Change in trend (CT)
Parameter estimation (β, γ)
Infections previous week
New infections (SIR)
Time since last policy change
Urgency to change trend (U)
Openness to change policies

Figure 3.2: Intuition behind SIMLR. The policies currently in place determine the value of the parameters needed to infer the next values, using an SIR model. Those predictions are then used to estimate how the policies might change in the future.

$\beta_{t+1}$ and $\gamma_{t+1}$ fully parameterize the SIR model in Equation 3.1, we can estimate the new number of infected people, $\Delta I_{t+1}$, from these parameters (as well as the SIR values at time $t$).

The random variables $U_t \in \{-1, 0, 1\}$ and $O_t \in \{0, 1\}$ are auxiliary variables designed to predict the probability of observing a change in policy at time $t + 1$. Intuitively, $U_t$ represents the "urgency" of modifying a policy. As the number of cases per 100K inhabitants and the rate of change between the number of cases in two consecutive time points increases, the urgency to set *stricter* government policies increases. As the number (and rate of change) of cases decreases, the urgency to *relax* the policies increases. Finally, $O_t$ models the "willingness" to execute a change in government policies. As the number of time points without a change increases, so does this "willingness".

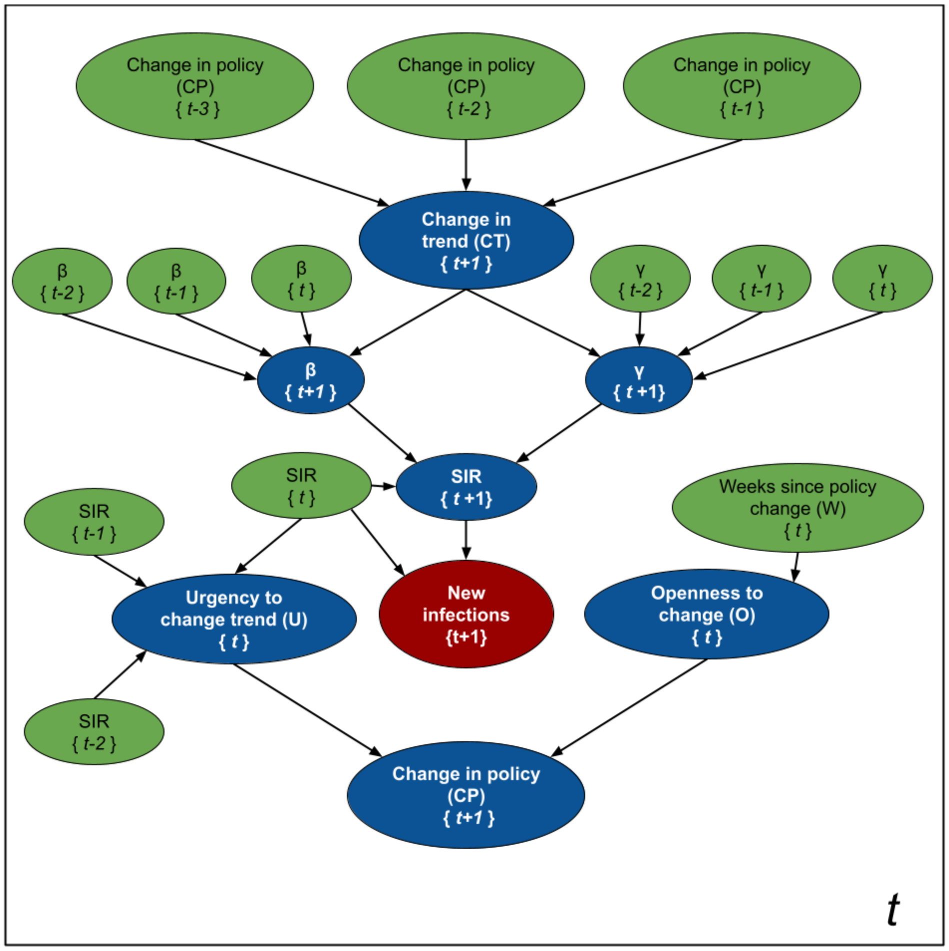

Change in policy (CP) { t-3 }
Change in policy (CP) { t-2 }
Change in policy (CP) { t-1 }
Change in trend (CT) { t+1 }
β { t-2 }
β { t-1 }
β { t }
γ { t-2 }
γ { t-1 }
γ { t }
β { t+1 }
γ { t +1}
SIR { t }
SIR { t +1}
SIR { t-1 }
Weeks since policy change (W) { t }
Urgency to change trend (U) { t }
New infections {t+1}
Openness to change (O) { t }
SIR { t-2 }
Change in policy (CP) { t+1 }
t

Figure 3.3: Modeling SIMLR as a PGM for forecasting new cases of COVID-19. The blue nodes are estimated at each time point, while the green ones are either based on past information, or where estimated in a previous iteration.

### 3.2.1 SIR with Time-Varying Parameters

We can approximate an SIR model by transforming the differential Equation 3.1 into the equations of differences:

$$\begin{aligned} S_t &= -\beta \frac{S_{t-1} I_{t-1}}{N} + S_{t-1} \\ I_t &= \beta \frac{S_{t-1} I_{t-1}}{N} - \gamma I_{t-1} + I_{t-1} \\ R_t &= \gamma I_{t-1} + R_{t-1} \end{aligned} \tag{3.3}$$

where $S_t, I_t, R_t$ are the number of individuals in the groups Susceptible, Infected and Removed, respectively, at time $t$. Similarly $S_{t-1}, I_{t-1}, R_{t-1}$ represent the number individuals in each group at time $t-1$. $\beta$ is the transmission rate, and $\gamma$ is the recovery rate.

While the SIR model is non-linear with respect to the states (S, I, R), it is linear with respect to the parameters $\beta$ and $\gamma$. Therefore, under the assumption of constant and known population size (*i.e.*, $N = S_t + I_t + R_t$) we can re-write the set of Equations 3.3 as:

$$\begin{aligned} \begin{bmatrix} S_t \\ I_t \end{bmatrix} &= \begin{bmatrix} -\frac{S_{t-1} I_{t-1}}{N} & 0 \\ \frac{S_{t-1} I_{t-1}}{N} & -I_{t-1} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} S_{t-1} \\ I_{t-1} \end{bmatrix} \\ R_t &= N - S_t - I_t \end{aligned} \tag{3.4}$$

Given a sequence of states $x_1, \ldots, x_n$, where $x_t = [S_t \; I_t]^T$, it is possible to estimate the optimal parameters of the SIR model as:

$$(\beta^*,\ \gamma^*) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} ||x_i - \hat{x}_i||^2 + \lambda_1 (\beta - \beta_0)^2 + \lambda_2 (\gamma - \gamma_0)^2 \tag{3.5}$$

where $\hat{x}_i$ is computed using Equation 3.4, and $\lambda_1$ and $\lambda_2$ are optional regularization parameters that allow the incorporation of the priors $\beta_0$ and $\gamma_0$. For the case of Gaussian priors – *i.e.*, $\beta \sim \mathcal{N}(\beta_0, \sigma_\beta^2)$ and $\gamma \sim \mathcal{N}(\gamma_0, \sigma_\gamma^2)$ – we use $\lambda_1 = \frac{1}{2\sigma_\beta^2}$ and $\lambda_2 = \frac{1}{2\sigma_\gamma^2}$ [87]. Intuitively, Equation 3.5 computes the transmission rate ($\beta^*$) and the recovery rate ($\gamma^*$) that best explain the number of new infections, deaths, and recovered people in a fixed time frame. If we know a standard recovery rate and transmission rate a priori ($\beta_0, \gamma_0$), it is possible to incorporate them into the Equation 3.5 as regularization parameters. The weights $\lambda_1$ and $\lambda_2$ control how much to weight those prior parameters. Small weights means we basically use the parameters learned by the data, and large weights mean more emphasis on the prior information.

In the traditional SIR model, we set $\lambda_1 = \lambda_2 = 0$ and fit a single $\beta$ and $\gamma$ to the entire time series. However, as shown in Figure 3.1(a), an SIR model with fixed parameters is unable to accurately model several waves of infections. As illustration, Figure 3.1(b) shows the predictions produced by fitting an SIR model with fixed parameters (Equation 3.5) to the US data from 29/March/2020 to 3/May/2021, and then using those parameters to make predictions one week in advance, over this same interval. That is, using this learned $(\beta, \gamma)$, and the number of people in the $S$, $I$, and $R$ compartments on 28/March/2020, we predicted the number of observed cases during the week of 29/March/2020 to 4/April/2020. We repeated the same procedure for the entire time series. Note that even though the parameters $\beta$ and $\gamma$ were found using *the entire time series* – *i.e.*, using information that was not available at the time of prediction – the resulting model still does a poor job fitting the reported data.

Figure 3.1(c), on the other hand, was created by allowing $\beta$ and $\gamma$ to change every week. Here, we first found the parameters that fit the data from 29/March/2020 to 4/April/2020 – call them $\beta_1$ and $\gamma_1$ – then used those parameters along with the SIR state on 28/March/2020 to predict the number of new infections one week ahead – *i.e.*, , the sampled week of 29/March/2020 to 4/April/2020. By repeating this procedure during the entire time series we obtained an almost perfect fit to the data. Of course, these are also not “legal” predictions since they too use information that is not available at prediction time. – *i.e.*, they used the number of reported infections during this first week to find the parameters, which were then used to estimate the number of cases over this time. However, this “cheating” example shows that an SIR model, with the optimal time-varying parameters, can model the complex dynamics of COVID-19. Recall from Figure 3.1(b) that this is not the case in the SIR model with fixed parameters, which cannot even properly fit the training data.

### 3.2.2 Estimating SIR parameters

Naturally, the challenge is "legally" computing the appropriate values of $\beta_{t+1}$ and $\gamma_{t+1}$, for each week, using only the data that is known at time $t$. Figure 3.3 shows that computing $\beta_{t+1}$ and $\gamma_{t+1}$ depends on the status of the random variable $\mathrm{CT}_{t+1}$. When $\mathrm{CT}_{t+1} = 0$ – *i.e.*, there is no change in the current trend – we assume that:

$$\begin{aligned} \beta_{t+1} &\sim \mathcal{N}(\alpha_0 + \alpha_1\beta_t + \alpha_2\beta_{t-1} + \alpha_3\beta_{t-2},\ \sigma_\beta^2) \\ \gamma_{t+1} &\sim \mathcal{N}(\omega_0 + \omega_1\gamma_t + \omega_2\gamma_{t-1} + \omega_3\gamma_{t-2},\ \sigma_\gamma^2) \end{aligned} \tag{3.6}$$

At time $t$, we can use the historical *daily* data $x_1, x_2, \ldots, x_t$ to find the *weekly* parameters $\beta_1, \beta_2, \ldots, \beta_{t/7}$ and $\gamma_1, \gamma_2, \ldots, \gamma_{t/7}$. Note that there is just one value for each week, so if there are 140 days, there are $140/7 = 20$ weeks. The first *weekly* pair $(\beta_1, \gamma_1)$ is found by fitting Equation 3.5 to $x_1, \ldots, x_7$; $(\beta_2, \gamma_2)$ to $x_8, \ldots, x_{14}$; and so on. Finally, we find the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ in Equation 3.6 by maximizing the likelihood of the computed pairs. After finding those parameters, it is straightforward to infer $(\beta_{t+1}, \gamma_{t+1})$. Note that this approach is the probabilistic version of linear regression. To estimate the parameters $\sigma_\beta^2$ and $\sigma_\gamma^2$ we can simply estimate the variance of the residuals. An advantage of also computing these variances is that it is possible to obtain confidence intervals by sampling from the distribution in Equation 3.6 and then using those samples along with Equation 3.3 to estimate the distribution of the new infected people.

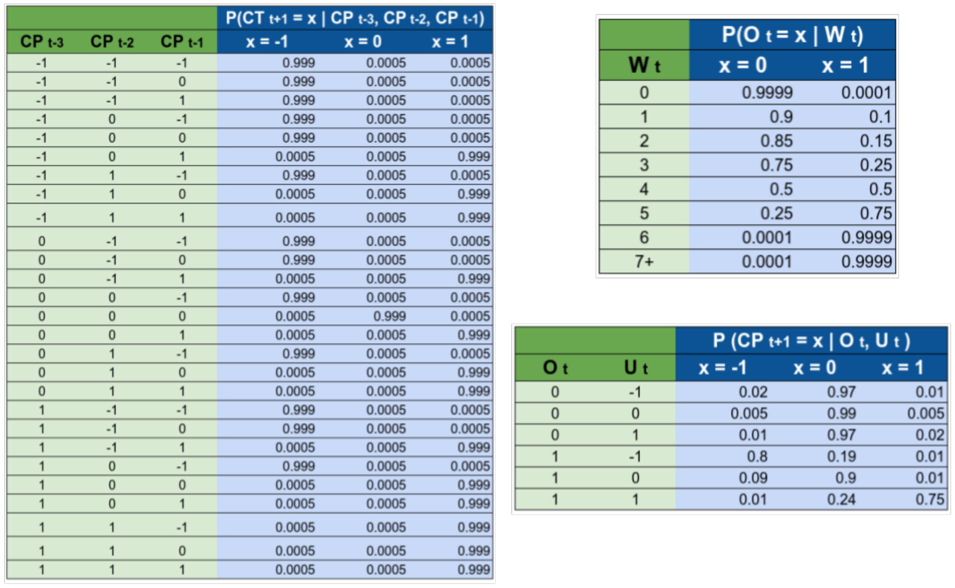
We estimated $\beta_{t+1}$ and $\gamma_{t+1}$ as a function of the 3 previous values of those parameters since this allows them to incorporate the velocity and acceleration at which the parameters change. We computed the velocity of $\beta$ as $v_{\beta,t} = \beta_t - \beta_{t-1}$ and its acceleration as $a_{\beta,t} = v_{\beta,t} - v_{\beta,t-1}$. Then, estimating $\beta_t = \theta_0 + \theta_1\beta_{t-1} + \theta_2 v_{\beta,t-1} + \theta_3 a_{\beta,t-1}$ is equivalent to the model in Equation 3.6. The same reasoning applies to the computation of $\gamma_t$. We call this approach the "trend-following varying-time parameters SIR", tf-v-SIR.

For the case of $\mathrm{CT}_t = -1$ and $\mathrm{CT}_t = 1$ (which represents a change in trend from increasing number of infections to decreasing number of infections or vice-versa), we set $\beta_{t+1}$ and $\gamma_{t+1}$ to values such that the predicted number of new cases at week $t+1$ is identical to the one at week $t$. We call this the "Same as the Last Observed Week" (SLOW) model. As shown in Section 3.3, SLOW is a baseline with very good performance despite its simplicity. Given that the pandemic is a physical phenomenon that changes relatively slowly from one week to the next, making a prediction that assumes that the new number of cases will remain constant is not a bad prediction.

### 3.2.3 Estimating changes in policies

The random variables $\mathrm{CT}_{t+1}, \mathrm{CP}_{t+1}$ and $\mathrm{O}_t$ in Figure 3.3 are all discrete nodes with discrete parents, meaning their probability mass functions are fully defined by conditional probability tables (CPTs). Learning the parameters of such CPTs from data is challenging due to the scarcity of historical information. The random variable $\mathrm{CT}_{t+1}$ depends on the random variable changes in policy (CP) at times $t-1, t-2, t-3$; however, there are very few changes in policy in a given region, meaning it is difficult to accurately estimate those probabilities from data. For the random variable O, which represents the "willingness" of the government to implement a change in policy, there is no observable data at all. We therefore relied on prior expert knowledge to set the parameters of the conditional probability tables for these random variables. Figure 3.4 shows the conditional probability tables (CPT) for the random variables $\mathrm{CT}_{t+1}$, $\mathrm{CP}_{t+1}$, $\mathrm{O}_t$. The intuition used to generate the CPT's is as follows:

We considered that a change in trend in the current week depends on changes in policies during the previous three weeks. We chose 3 weeks using the hypothesis that the incubation period for the virus is 2 weeks. Then the effects of a policy will be reflected approximately 2 weeks after a change. We decided to analyze also one week after, and one week before this period, giving as a result the tracking of $CP_{t-3}$ to $CP_{t-1}$. Secondly, we also assume that whenever we observe a change of policy that will move the trend from going up to going down, then that event will most likely happen. This is why most of the probability mass is located in a single column. For example, if we observe that the policies are relaxed at any point during the weeks t- 3, t-2, or t-1, then we assume that we will observe a change in trend with 99.9%



| | | | $P(CT_{t+1} = x \mid CP_{t-3}, CP_{t-2}, CP_{t-1})$ | | |
|---|---|---|---|---|---|
| $CP_{t-3}$ | $CP_{t-2}$ | $CP_{t-1}$ | x = -1 | x = 0 | x = 1 |
| -1 | -1 | -1 | 0.999 | 0.0005 | 0.0005 |
| -1 | -1 | 0 | 0.999 | 0.0005 | 0.0005 |
| -1 | -1 | 1 | 0.999 | 0.0005 | 0.0005 |
| -1 | 0 | -1 | 0.999 | 0.0005 | 0.0005 |
| -1 | 0 | 0 | 0.999 | 0.0005 | 0.0005 |
| -1 | 0 | 1 | 0.0005 | 0.0005 | 0.999 |
| -1 | 1 | -1 | 0.999 | 0.0005 | 0.0005 |
| -1 | 1 | 0 | 0.0005 | 0.0005 | 0.999 |
| -1 | 1 | 1 | 0.0005 | 0.0005 | 0.999 |
| 0 | -1 | -1 | 0.999 | 0.0005 | 0.0005 |
| 0 | -1 | 0 | 0.999 | 0.0005 | 0.0005 |
| 0 | -1 | 1 | 0.0005 | 0.0005 | 0.999 |
| 0 | 0 | -1 | 0.999 | 0.0005 | 0.0005 |
| 0 | 0 | 0 | 0.0005 | 0.999 | 0.0005 |
| 0 | 0 | 1 | 0.0005 | 0.0005 | 0.999 |
| 0 | 1 | -1 | 0.999 | 0.0005 | 0.0005 |
| 0 | 1 | 0 | 0.0005 | 0.0005 | 0.999 |
| 0 | 1 | 1 | 0.0005 | 0.0005 | 0.999 |
| 1 | -1 | -1 | 0.999 | 0.0005 | 0.0005 |
| 1 | -1 | 0 | 0.999 | 0.0005 | 0.0005 |
| 1 | -1 | 1 | 0.0005 | 0.0005 | 0.999 |
| 1 | 0 | -1 | 0.999 | 0.0005 | 0.0005 |
| 1 | 0 | 0 | 0.0005 | 0.0005 | 0.999 |
| 1 | 0 | 1 | 0.0005 | 0.0005 | 0.999 |
| 1 | 1 | -1 | 0.0005 | 0.0005 | 0.999 |
| 1 | 1 | 0 | 0.0005 | 0.0005 | 0.999 |
| 1 | 1 | 1 | 0.0005 | 0.0005 | 0.999 |

| | $P(O_t = x \mid W_t)$ | |
|---|---|---|
| $W_t$ | x = 0 | x = 1 |
| 0 | 0.9999 | 0.0001 |
| 1 | 0.9 | 0.1 |
| 2 | 0.85 | 0.15 |
| 3 | 0.75 | 0.25 |
| 4 | 0.5 | 0.5 |
| 5 | 0.25 | 0.75 |
| 6 | 0.0001 | 0.9999 |
| 7+ | 0.0001 | 0.9999 |

| | | $P(CP_{t+1} = x \mid O_t, U_t)$ | | |
|---|---|---|---|---|
| $O_t$ | $U_t$ | x = -1 | x = 0 | x = 1 |
| 0 | -1 | 0.02 | 0.97 | 0.01 |
| 0 | 0 | 0.005 | 0.99 | 0.005 |
| 0 | 1 | 0.01 | 0.97 | 0.02 |
| 1 | -1 | 0.8 | 0.19 | 0.01 |
| 1 | 0 | 0.09 | 0.9 | 0.01 |
| 1 | 1 | 0.01 | 0.24 | 0.75 |

Figure 3.4: Conditional probability tables used by SIMLR. The names of the variables refer to the nodes that appear on Figure 2 on the main text

probability.

The rationale for the CPT $P(O_t \mid W_t)$ is that the government becomes more open to implement changes after long periods of 'inactivity'. For example, if they implement a change in policy this week ($W_t = 0$), then the probability of considering a second change of policy during the same week is very small (0.01 %). We are assuming that, after a change in policy, the government will wait to see the effect of that change before taking further action. If 4 weeks have passed since the last change in policy, we estimated the probability of considering a change in the policy as 50%, while if more than 7 weeks have passed, then they are fully open to the possibility of implementing a new change.

$P(O_t \mid W_t)$ estimates the probability of considering a change in the policy. The probability of actually implementing a change, $P(CP_{t+1} \mid O_t, U_t)$ depends not only on how willing the government is, but also on how urgent it is to make a change. In general, if the government is open to implement a change, and the urgency is "high", then the probability of changing a policy is high. We also considered that the government "prefers" to either not make changes in policy or relax the policies, rather than to implement more strict policies.

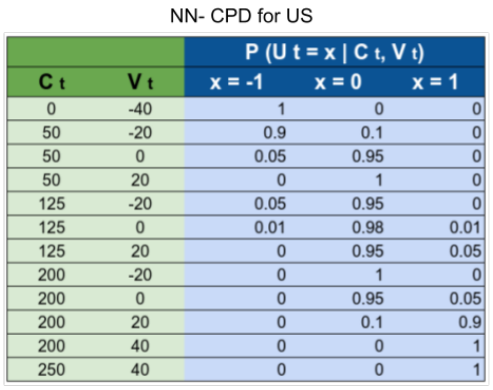
For modelling the random variable $\mathrm{U}_t$, which represents the "Urgency to change the trend", we use an NN-CPD (neural-network conditional probability distribution), which is a modified version of the multinomial logistic conditional probability distribution [72].

**Definition 1** (NN-CPD)**.** Let $Y \in \{1, \ldots, m\}$ be an $m$-valued random variable with $k$ parents $X_1, \ldots, X_k$ that each take on numerical values. The conditional probability distribution $P(Y \mid X_1, \ldots, X_k)$ is an NN-CPD if there is an function $z = f_\theta(X_1, \ldots, X_k) \in \mathbb{R}^m$, represented as a neural network with parameters $\theta$, such that $p(Y = i \mid x_1, \ldots, x_k) = \exp(z_i)/\sum_j \exp(z_j)$, where $z_i$ represents the *i-th* entry of $z$.

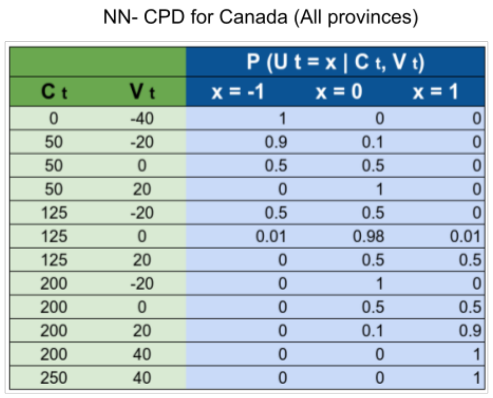
Note $\mathrm{U}_t$ is a latent variable, so there is no observable data at all. We again rely on domain knowledge to estimate its probabilities. To compute $P(\mathrm{U}_t \mid \mathrm{SIR}_{t-2}, \mathrm{SIR}_{t-1},\ \mathrm{SIR}_t)$, we extract two features: $c_t = 10\mathrm{E}5(S_{t-1} - S_t)/N$, which represents the number of new reported infections per 100K inhabitants; and $v_t = c_t - c_{t-1}$, which estimates the rate of change of $c_t$. Then define $P(\mathrm{U}_t \mid \mathrm{SIR}_{t-2}, \mathrm{SIR}_{t-1},\ \mathrm{SIR}_t) = P(\mathrm{U}_t \mid c_t, v_t)$.

To learn the parameters $\theta$ we created the dataset shown in Figure 3.5. Note that the targets in such dataset are probabilities. We relied on the probabilistic labels approach proposed by Vega et al. [120] to use a dataset with few training instances along with their probabilities to learn the parameters of a neural network more efficiently. We trained and a simple neural network with a single hidden layers with 64 units, and 3 output units with softmax activation.

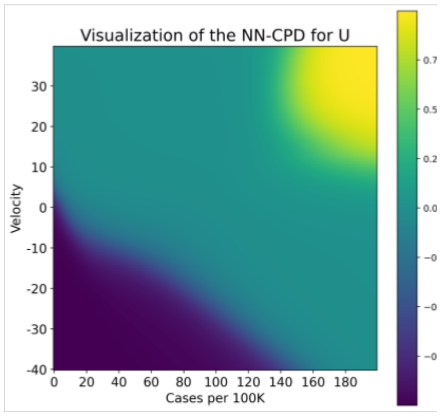
The random variables $U_t \in \{-1, 0, 1\}$ and $O_t \in \{0, 1\}$ are auxiliary variables designed to predict the probability of observing a change in policy at time $t + 1$. Intuitively, $U_t$ represents the "urgency" of modifying a policy. As the number of cases per 100K inhabitants and the rate of change between the number of cases in two consecutive time points increases, the urgency to set *stricter* government policies increases. As the number (and rate of change) of cases decreases, the urgency to *relax* the policies increases. Most of the parameters in both NN-CPD tables are similar for the US and Canada, the difference arises from a perceived preference for not setting very strict policies in the US during the first year of the pandemic.


NN- CPD for US

| | | $P(U_t = x \mid C_t, V_t)$ | | |
|---|---|---|---|---|
| $C_t$ | $V_t$ | x = -1 | x = 0 | x = 1 |
| 0 | -40 | 1 | 0 | 0 |
| 50 | -20 | 0.9 | 0.1 | 0 |
| 50 | 0 | 0.05 | 0.95 | 0 |
| 50 | 20 | 0 | 1 | 0 |
| 125 | -20 | 0.05 | 0.95 | 0 |
| 125 | 0 | 0.01 | 0.98 | 0.01 |
| 125 | 20 | 0 | 0.95 | 0.05 |
| 200 | -20 | 0 | 1 | 0 |
| 200 | 0 | 0 | 0.95 | 0.05 |
| 200 | 20 | 0 | 0.1 | 0.9 |
| 200 | 40 | 0 | 0 | 1 |
| 250 | 40 | 0 | 0 | 1 |


NN- CPD for Canada (All provinces)

| | | $P(U_t = x \mid C_t, V_t)$ | | |
|---|---|---|---|---|
| $C_t$ | $V_t$ | x = -1 | x = 0 | x = 1 |
| 0 | -40 | 1 | 0 | 0 |
| 50 | -20 | 0.9 | 0.1 | 0 |
| 50 | 0 | 0.5 | 0.5 | 0 |
| 50 | 20 | 0 | 1 | 0 |
| 125 | -20 | 0.5 | 0.5 | 0 |
| 125 | 0 | 0.01 | 0.98 | 0.01 |
| 125 | 20 | 0 | 0.5 | 0.5 |
| 200 | -20 | 0 | 1 | 0 |
| 200 | 0 | 0 | 0.5 | 0.5 |
| 200 | 20 | 0 | 0.1 | 0.9 |
| 200 | 40 | 0 | 0 | 1 |
| 250 | 40 | 0 | 0 | 1 |



[b]

Figure 3.5: Dataset used to create the NN-CPD for the variable $U_t$ and its visualization. Values closer to 1 (yellow) increase $p(U_t = 1 \mid C_t, V_t)$. Values closer to 0 (green) increase $p(U_t = 0 \mid C_t, V_t)$. Values closer to $-1$ (blue) increase $p(U_t = -1 \mid C_t, V_t)$

### 3.2.4 Evaluation

We evaluated the performance of SIMLR, in terms of the mean absolute percentage error (MAPE) and mean absolute error (MAE), for forecasting the number of new infections 1 to 4 weeks in advance, in data from United States (as a country and individually for every state) and the 6 biggest provinces of Canada: Alberta (AB), British Columbia (BC), Manitoba (MN), Ontario (ON), Quebec (QB), and Saskatchewan (SK). For each of the regions, the predictions are done on a weekly basis, over the 39 weeks from 26/Jul/2020 to 1/May/2021. This time span captures different waves of infections. Equation 3.7 show the computation of the metrics used for evaluating our approach.

$$\begin{aligned} \text{MAPE} &= \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t-\hat{y}_t}{y_t}\right| \\ \text{MAE} &= \frac{1}{n}\sum_{t=1}^{n}\left|y_t-\hat{y}_t\right| \end{aligned} \tag{3.7}$$

At the end of every week, we fitted the SIMLR parameters using the data that was available until that week. For example, on 25/Jul/2020, we used all the data available from 1/Jan/2020 to 25/Jul/2020 to fit the parameters of SIMLR. Then, we made the predictions for the number of new infections during the weeks: 26/Jul/2020 – 1/Aug/2020 (1 week in advance), 2/Aug/2020 – 8/Aug/2020 (2 weeks in advance), 9/Aug/2020 – 15/Aug/2020 (3 weeks in advance), and 16/Aug/2020 – 22/Aug/2020 (4 weeks in advance). After this, we then fitted the parameters with data up to 1/Aug/2020 and repeated the same process, for 38 more iterations, until we covered the entire range of predictions.

We compared the performance of SIMLR with the SIR compartmental model with time-varying parameters learned using Equation 3.6 but no other random variable (tf-v-SIR), and with the simple model that forecasts that the number of cases 1 to 4 weeks in advance is the "Same as the Last Observed Week" (SLOW). For the United States data, we also compared the performance of SIMLR against the publicly available predictions at the COVID-19 Forecast Hub, which are the predictions submitted to the Center for Disease Control and Prevention (CDC) [19].

For training, we used the publicly available dataset OxCGRT [48], which contains the policies implemented by different regions, as well as the time period over which they were implemented. We limited our analysis to 3 policy decision: `Workplace closing`, `Stay at home requirements`, and `Cancellation of public events` in the case of Canada. For the case of the United States we used `Restrictions on gatherings`, `Vaccination policy`, and `Cancellation of public events`. For information about the new number of reported cases and deaths, we used the publicly available COVID-19 Data Repository by the Center for Systems Science and Engineering at Johns Hopkins University [22]. The code for reproducing the results presented here are discussed in Appendix A.

## 3.3 Results

### 3.3.1 Data Preprocessing

Before inputting the time-series data to SIMLR, we performed some basic preprocessing *during the training phase*, and *exclusively on the training data*. We evaluated of our models by comparing its predictions with the results reported by the different health agencies –i.e. we did not fill in the data on the test sets:

1. The original data contains the cumulative number of reported infections/deaths on a daily basis. We trivially transformed this time-series into the number of new daily infections/deaths.

2. We considered negative values from the new daily infections/deaths time-series as missing, assuming these negative values arose due to inconsistencies during the data reporting procedure.

3. We "filled-in" the missing values. When the number of new infections/deaths was missing at day $d$, we assumed that the entry at $d+1$ contained the cases for both $d$ and $d+1$, and divided the number of new infections/deaths evenly between both days.

4. We eliminated outliers. For each day $d$, with number of reported new infections, $\Delta I_d$, we computed the mean ($\mu_d$) and standard deviation ($\sigma_d$ of the set $\Delta I_{d-10}, \ldots, \Delta I_{d-1}$; we then set $\Delta I_d := \min\{\Delta I_d, \mu_d + 4\sigma_d\}$.

5. We used the number of new infections and new deaths to produce the SIR vector $SIR_t = [S_t, I_t, R_t]$.

In step 5, we assumed that everyone in a given region was susceptible at the start time – *i.e.*, $S_0 = N$. At each new time point, we transfer the number of new infections from $S$ to $I$, and the number of new deaths and recovered from $I$ to $R$. If the number of new recovered people is not reported, we used the surveillance definition of recovered used by Canadian health agencies. This definition is based on the assumption that a recovered person is one who is not hospitalized and is 14 days past the day when they tested positive [2, 95]:

> ”Active and recovered status is a surveillance definition to try to understand the number of active cases in the population. It is not related to clinical management of cases. It is based on the assumption that a case is recovered 14 days after a particular date...”

### 3.3.2 MAPE and MAE

Figure 3.6 shows the MAPE of the 1- to 4-week forecasts for the United States as a country and the 6 biggest provinces of Canada. Note that SIMLR has a consistently lower MAPE than tf-v-SIR and SLOW. Figure 3.7 shows a similar result in terms of MAE. Tables 3.1 and 3.2 show the mean and standard deviations of the metrics corresponding to the Figures 3.6 and 3.7. In addition Table 3.3 show the correlation coefficient between the time series of the reported new infections every week and the predictions made by the different models.

Figure 3.8(c) shows how our proposed SIMLR approach compares with the 18 models that submitted predictions at the country level to the CDC during the same span of time (results at the state level are included in the appendix). Note that SIMLR and the model *LNQ-ens1* are the best performing models, with no statistically significant difference ($p > 0.05$ on a paired t-test) with respect to MAPE.

## 3.4 Discussion

Figure 3.8 illustrates the actual predictions of SIMLR one week in advance for the province of Alberta, Canada; and two weeks in advance for the US as a country. These two cases exemplify the behaviour of SIMLR. As noted above, there is a 2- to 4-week lag after a policy changes, before we see the effects. This means the task of making 1-week forecasts is relatively simple, as the

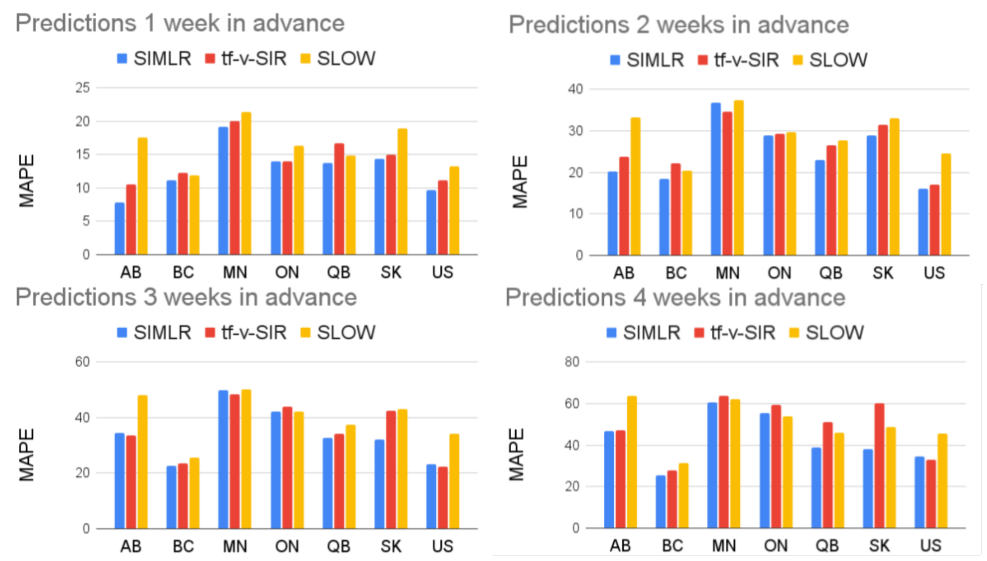


Figure 3.6: Comparison of SIMLR, SIR model with time-varying parameters, and SLOW. Table 2 in supplementary material contains the numerical information.

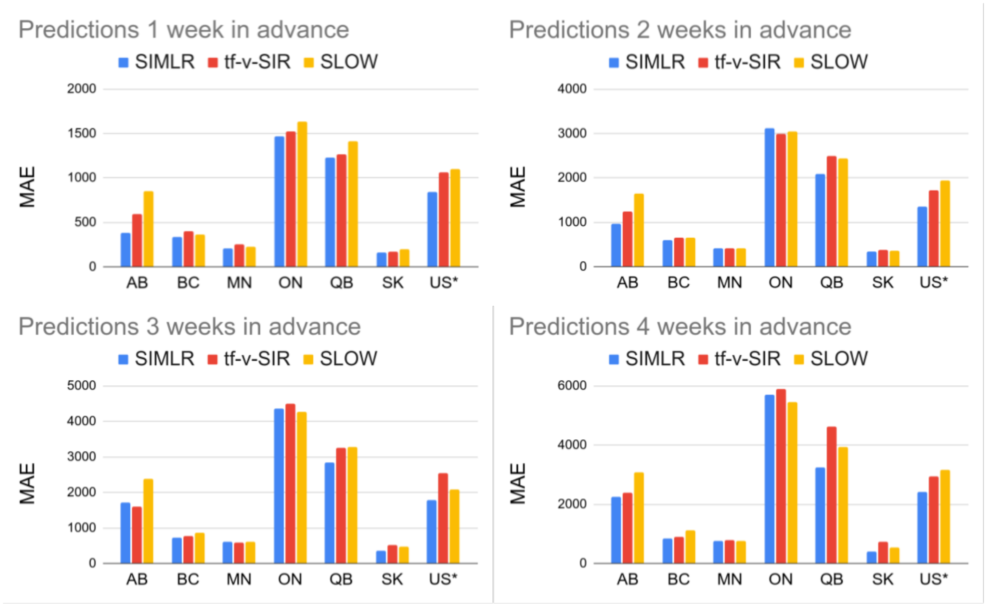


Figure 3.7: Comparison of SIMLR, SIR model with time-varying parameters, and SLOW in terms of MAE. To make the numbers comparable, the figures each show the US MAE values divided by 100.

Table 3.1: MAPE of the 6 biggest provinces in Canada and United States as a country, 1- to 4-week in advance. The number in parenthesis is the standard deviation.

| | 1 Week | | | 2 Weeks | | |
|---|---|---|---|---|---|---|
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 7 (8) | 10 (10) | 17 (9) | 20 (14) | 23 (16) | 33 (17) |
| BC | 11 (8) | 12 (10) | 11 (8) | 18 (10) | 22 (15) | 20 (13) |
| MN | 19 (14) | 20 (13) | 21 (15) | 36 (24) | 34 (22) | 37 (24) |
| ON | 14 (9) | 14 (10) | 16 (10) | 28 (21) | 29 (24) | 29 (19) |
| QB | 13 (11) | 14 (11) | 16 (11) | 23 (20) | 26 (30) | 27 (19) |
| SK | 14 (9) | 15 (12) | 18 (13) | 28 (17) | 31 (18) | 33 (18) |
| US | 9 (6) | 11 (8) | 13 (9) | 16 (13) | 19 (16) | 24 (17) |

| | 3 Weeks | | | 4 Weeks | | |
|---|---|---|---|---|---|---|
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 34 (21) | 33 (22) | 48 (26) | 46 (35) | 47 (33) | 63 (35) |
| BC | 22 (14) | 23 (16) | 25 (18) | 25 (20) | 27 (21) | 31 (20) |
| MN | 49 (31) | 48 (34) | 50 (27) | 60 (38) | 63 (42) | 62 (33) |
| ON | 42 (37) | 44 (40) | 42 (30) | 55 (51) | 59 (58) | 53 (40) |
| QB | 32 (28) | 34 (36) | 37 (27) | 38 (41) | 51 (64) | 45 (35) |
| SK | 32 (23) | 42 (32) | 43 (22) | 38 (24) | 60 (50) | 49 (26) |
| US | 23 (23) | 25 (26) | 34 (28) | 36 (38) | 38 (41) | 45 (40) |

Table 3.2: MAE of the 6 biggest provinces in Canada and United States as a country, 1- to 4-week in advance. The number in parenthesis is the standard deviation. For the case of the US the number of cases was divided by 100.

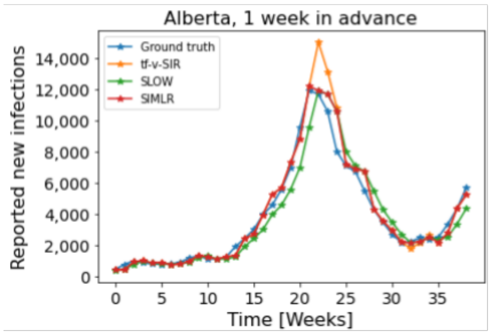
| | 1 Week | | | 2 Weeks | | |
|---|---|---|---|---|---|---|
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 385 (559) | 598 (905) | 850 (724) | 966 (971) | 1245 (1430) | 1651 (1258) |
| BC | 339 (304) | 397 (426) | 361 (294) | 594 (443) | 661 (480) | 648 (485) |
| MN | 204 (227) | 252 (271) | 221 (224) | 422 (371) | 418 (379) | 413 (346) |
| ON | 1471 (1343) | 1520 (1662) | 1635 (1388) | 3124 (2632) | 3001 (2847) | 3044 (2351) |
| QB | 1229 (1443) | 1265 (1354) | 1410 (975) | 2098 (2264) | 2496 (3270) | 2446 (1743) |
| SK | 161 (161) | 171 (203) | 194 (174) | 339 (294) | 382 (324) | 355 (264) |
| US* | 841 (796) | 1061 (1149) | 1103 (913) | 1361 (1398) | 1729 (1979) | 1933 (1580) |
| | 3 Weeks | | | 4 Weeks | | |
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 1719 (1381) | 1601 (1558) | 2378 (1649) | 2261 (1863) | 2385 (2087) | 3074 (1858) |
| BC | 731 (566) | 777 (672) | 853 (716) | 835 (709) | 883 (703) | 1127 (892) |
| MN | 609 (504) | 591 (501) | 602 (467) | 749 (612) | 775 (630) | 753 (571) |
| ON | 4357 (3672) | 4511 (3983) | 4266 (3053) | 5702 (4427) | 5910 (4988) | 5447 (3417) |
| QB | 2854 (2527) | 3261 (4096) | 3288 (2389) | 3244 (3131) | 4636 (6115) | 3947 (2788) |
| SK | 351 (320) | 522 (500) | 472 (306) | 410 (287) | 736 (733) | 541 (348) |
| US* | 1793 (2012) | 2089 (2768) | 2538 (2151) | 2414 (2755) | 2933 (4027) | 3157 (2679) |

Table 3.3: Pearson correlation coefficient between the ground truth and the predictions of the 6 biggest provinces in Canada and United States as a country 1- to 4-week in advance.

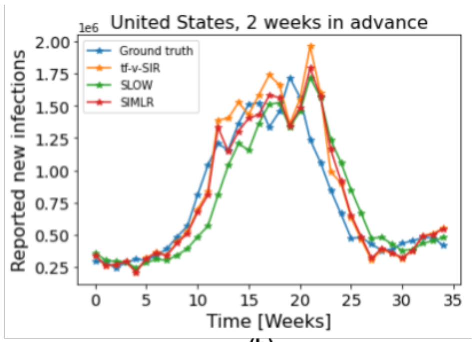
| | 1 Week | | | 2 Weeks | | |
|---|---|---|---|---|---|---|
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 0.99 | 0.98 | 0.96 | 0.94 | 0.95 | 0.84 |
| BC | 0.97 | 0.97 | 0.97 | 0.90 | 0.89 | 0.90 |
| MN | 0.96 | 0.96 | 0.95 | 0.85 | 0.87 | 0.86 |
| ON | 0.96 | 0.97 | 0.96 | 0.83 | 0.85 | 0.85 |
| QB | 0.97 | 0.97 | 0.96 | 0.93 | 0.89 | 0.86 |
| SK | 0.97 | 0.96 | 0.95 | 0.89 | 0.89 | 0.86 |
| US | 0.97 | 0.97 | 0.96 | 0.93 | 0.93 | 0.87 |

| | 3 Weeks | | | 4 Weeks | | |
|---|---|---|---|---|---|---|
| | SIMLR | tf-v-SIR | SLOW | SIMLR | tf-v-SIR | SLOW |
| AB | 0.90 | 0.90 | 0.68 | 0.84 | 0.85 | 0.50 |
| BC | 0.84 | 0.83 | 0.83 | 0.80 | 0.81 | 0.75 |
| MN | 0.69 | 0.75 | 0.73 | 0.51 | 0.56 | 0.57 |
| ON | 0.67 | 0.68 | 0.71 | 0.51 | 0.53 | 0.58 |
| QB | 0.83 | 0.84 | 0.74 | 0.71 | 0.71 | 0.61 |
| SK | 0.82 | 0.81 | 0.78 | 0.73 | 0.69 | 0.71 |
| US | 0.88 | 0.90 | 0.77 | 0.80 | 0.84 | 0.65 |

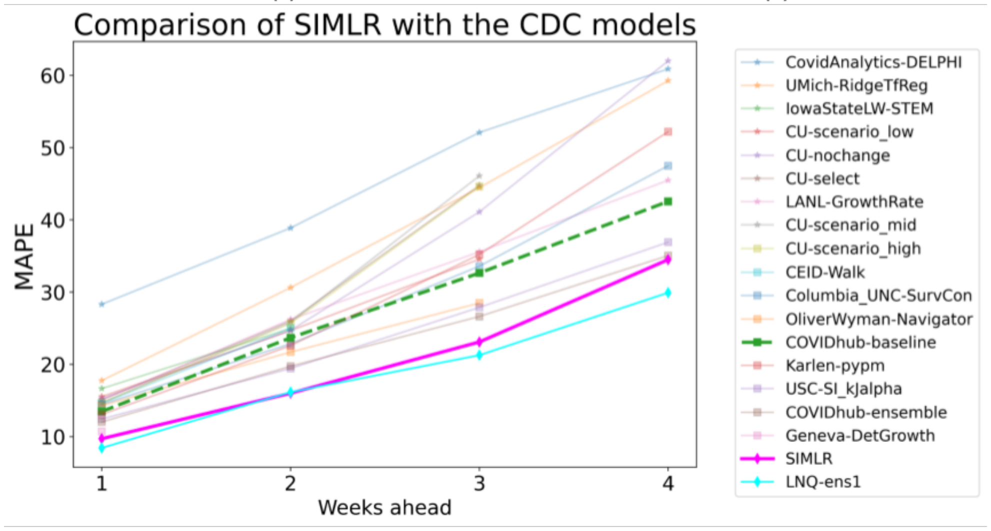
relevant policy (at times $t-3$ to $t-1$) is fully observable. This allows SIMLR to directly compute $\text{CT}_{t+1}$, which can then choose whether to continue using the SIR with time-varying parameters if no policy changed at time $t-1$, $t-2$, or $t-3$, or using the SLOW predictor if the policy changed.



(a)



(b)



(c)

Figure 3.8: (a) 1-week forecasts SIMLR, tf-v-SIR, and SLOW, for Alberta, Canada. (b) 2-week forecasts, of the same models, for US data. (c) Comparison of SIMLR versus models submitted to the CDC (on US data).

Figure 3.8(a) shows a change in the trend of reported new cases at week 22. However, just by looking at the evolution of number of new infections before week 22, there is no way to predict this change, which is why tf-v-SIR predicts that the number of new infections will continue growing. However, since SIMLR observed a change in the government policies at week 20, it realized it could no longer rely on its estimation of parameters and so switched to the SLOW model, which is why it was more accurate here. A similar behaviour occurs in week 34, when the 3rd wave of cases in Alberta started. Due to a relaxation in the policies on week 31, SIMLR (at week 31) correctly predicted a change of trend around weeks 33–35.

This behaviour is not exclusive for the data of Alberta and it explains why the performance of SIMLR is consistently better than the baselines used for comparison in Figure 3.6 and Figure 3.8(c). A striking result is how hard it is to beat the simple SLOW model (COVIDhub-baseline). Out of the 19 models considered here, only 5 (including SIMLR) do better than this simple baseline when predicting 3 to 4 weeks ahead! This brings some insight into the challenge of making accurate prediction in the medium term – probably due to the need to predict, then use, policy change information. Tables B.1 - B.4 in the appendix show a comparison between our proposed SIMLR and tf-v-SIR against the models submitted to the CDC for all the states in the US. SIMLR consistently ranks among the best performers, with the advantage of being an interpretable model.

A deeper analysis of Tables B.1 - B.4 shows that, in some states, the performance of SIMLR degrades for longer range predictions. This occurs because we are monitoring only the same 3 policies for all the states; however, different states might have implemented different policies and reacted differently to them. For example, closing schools might be a relevant policy in a state where there is an outbreak that involves children, but not as relevant if most of the cases are in older people.

Tracking irrelevant policies might degrade the performance of SIMLR. If the status of an irrelevant policy changes, then the dynamics of the disease will not be affected. The model however, will assume that the change in the policy will cause a change of trend and it will rely on the SLOW model, instead of the more accurate tf-v-SIR. Although SIMLR can be adapted to track different policies, the policies that are relevant for a given state must be given as an input. So while we think our overall approach applies in general, our specific model (tracking these specific policies, etc.) might not produce accurate predictions across all the regions. This is also a strength, in that it is trivial to adapt our specific model to track the policies of interest within a given region.

Predictions at the country level are more complicated, since most of the time policies are implemented at the state (or province) level instead of nationally. For making predictions for an entire country, as well as predictions 3 or 4 weeks in advance, SIMLR first predicts, then uses, the likelihood of observing a change in trend, at every week. In these cases, the random variable $CT_{t+1}$ no longer acts like a "switch", but instead it mixes the predictions of the tf-v-SIR and SLOW models, according to the probability of observing a change in the trend.

Figure 3.8(b) shows that whenever there is a stable trend in the number of new reported infections – which suggests there has been no recent policy changes – SIMLR relies on the predictions of the tf-v-SIR model; however, as the number (and rate of change) of new infections increases, so does the probability of observing a change in the policy. Therefore, SIMLR starts giving more weight to the predictions of the SLOW model. Note this behaviour in the same figure during weeks 13 – 20.

One limitation of SIMLR is that it relies on conditional probabilities that are hard to learn due to lack of data, which forced us to build them based on domain knowledge. If this prior knowledge is inaccurate, then the predictions might be also misleading. Also, different regions might have different "thresholds" for taking action. Despite this limitation, SIMLR produced state-of-the-art results in both forecasting in the US as a country and at the provincial level in Canada, as well as very competitive results in predictions at the state level in the US.

Note that modelling SIMLR as a PGM does not imply causality. Although changes in the observed policy influence changes in the trend of new reported cases, the opposite is also true in reality. However, using probabilistic graphical models does makes it interpretable. It also allows us to incorporate domain knowledge that compensates for the relatively scarce data. SIMLR's excellent performance – comparable to state-of-the-art systems in this competitive task – show that it is possible to design interpretable machine learning models without sacrificing performance.

Forecasting the number of new COVID-19 infections is a very challenging task. Many factors play a role on how the disease spreads, including the government policies and the adherence of citizens to such policies. These elements are difficult to model mathematically; however, the collected data (number of new infections and deaths, for example) are a reflection of all those complex interactions.

Machine learning, on the other side, excels at learning patterns directly from the data. Unfortunately, training many models from scratch can require a great deal of data, especially to learn complex patterns, such as the evolution of a pandemic.

We proposed SIMLR, a methodology that uses machine learning (ML) techniques to learn a model that can set, and adjust, the parameters of mathematical model for epidemiology (SIR). SIMLR augments that SIR model by incorporating expert knowledge in the form of a probabilistic graphical model. In this way, human experts can incorporate their believes in the likelihood that a policy will change, and when. By combining both components we substantially reduce the data that machine learning usually requires to produce models that can make accurate predictions.

Importantly, besides providing state-of-the-art predictions in terms of MAPE in the short and medium term, the resulting SIMLR model is interpretable and probabilistic. The first means that we can justify the predictions given by the algorithm – *e.g.*, "SIMLR predicts 1,000 cases for the next week due to a change in the government policies that will decrease the transmission rate". The second means we can produce probabilistic values – so instead of predicting a single value, it can predict the entire probability distribution – *e.g.*, the probability of 100 cases next week, or of 200 cases or of 1000, etc.

This chapter demonstrated that a model that explicitly models and incorporates government policy decisions can accurately produce 1- to 4-week forecasts of the number of COVID-19 infections. This involved showing that an SIR model with time-varying parameters is enough to describe the complex dynamics of this pandemic, including the different waves of infections. We expect that this approach will be useful not only for modelling COVID-19, but other infectious diseases as well. We also hope that its interpretability will leads to its adoption by researchers, and users, in epidemiology and other non-ML fields.

# Chapter 4

# Batch effects

## 4.1 Introduction

As explained in Chapter 1, the goal of supervised machine learning is to produce a model that accurately predicts a value, $y$, given a vector input, $x$, corresponding (implicitly) to an unknown function $y = h(x)$ [87]. In the supervised setting, we learn an approximate $\hat{y} = \hat{h}(x) \approx h(x)$, by applying a learning algorithm to a (source) training dataset $D_S = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$. We can then apply this $\hat{h}$ to new instances $D_T = \{x_1, x_2, \ldots x_m\}$.

Chapters 2 and 3 explain strategies to incorporate domain-specific information during the training process. Still, those strategies assume that the source (S) and the target (T) domains follow the same probability distribution–i.e. $P_S(X, Y) = P_T(X, Y)$. This is a very common assumption in machine learning, but when the assumption is incorrect, a predictor learned using $D_S$ might not generalize when used on $D_T$ [107]. The performance on the target domain depends on its performance on the source domain, and on the similarity between the distributions of the domain and target domains [8].

Especially problematic is the case where there are just a few training instances available ($n$), each represented by a large number of features ($p$), which might range from a few hundreds to millions. In this undesirable situation, known as "*small n, large p*" [52], the training sample might not be a good approximation of the real distribution of the data [4].

A well known model to explain the discrepancy between distributions is covariate shift, where $P_S(X) \neq P_T(X)$, but $P_S(Y \mid X) = P_T(Y \mid X)$ [106].

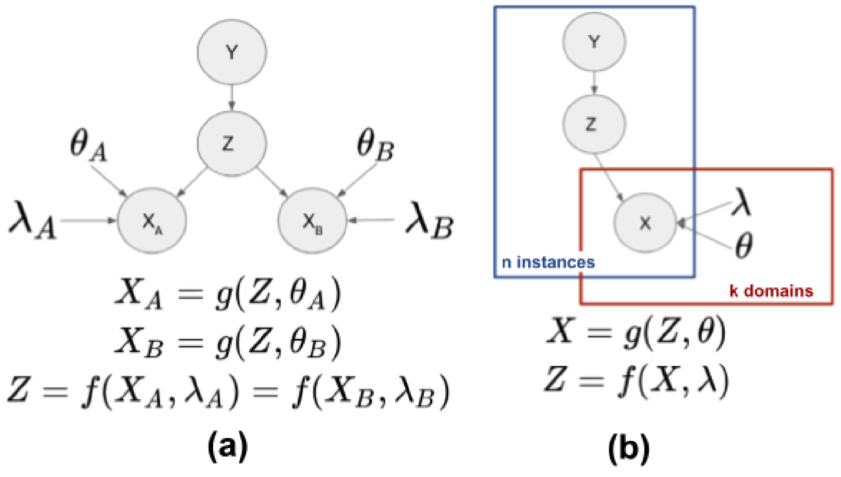


Figure 4.1: **(a)** Domain-shift for domains A and B. $Y$ represents the label, $Z$ represent a latent common feature space. $X_A$ and $X_B$ are the observations for the two different domains. Here, $g_\theta : Z \to X$ and $f_\lambda : X \to Z$. **(b)** Plate model for representing more than 2 domains.

Other assumptions lead to different models [107, 73], which motivate algorithms that decrease the negative impact of the discrepancies under different circumstances [20, 125].

Here, we focus on the case where $P_S(X) \neq P_T(X)$, $P_S(Y \mid X) \neq P_T(Y \mid X)$, and $P_S(Y) = P_T(Y)$. However, we assume the existence of a function $f(\cdot)$, with parameters $\lambda_S$ and $\lambda_T$, such that $P_S(\ f(X,\ \lambda_S)\ ) = P_T(\ f(X,\ \lambda_T)\ )$ and $P_S(\ Y \mid f(X,\ \lambda_S)\ ) = P_T(\ Y \mid f(X,\ \lambda_T)\ )$. This implies that there is a common feature space where the source and target domains follow the same distribution; see Figure 4.1. This model is known as domain-shift [107], or covariate observation shift [73]. Specifically, Sections 4.2 and 4.3 describe approaches for correcting for batch effects when the mapping from source and target domains to a common representation is an affine function (*i.e.*, $Z_i = f(X_{S,i}, \lambda_S)$ and $Z_j = f(X_{T,j}, \lambda_T)$ have the form $f(X, \lambda) = \lambda X + \lambda_0$).

Figure 4.2(a) exemplifies domain shift. Note that the decision boundary learned from the source domain (shown with solid squares and triangles) has poor performance on the target domain (shown with dashed squares and triangles). Domain-shift adaptation aims to find a common representation that minimizes the divergence between the domains. If successful, the decision function learned during training will have a good performance when doing

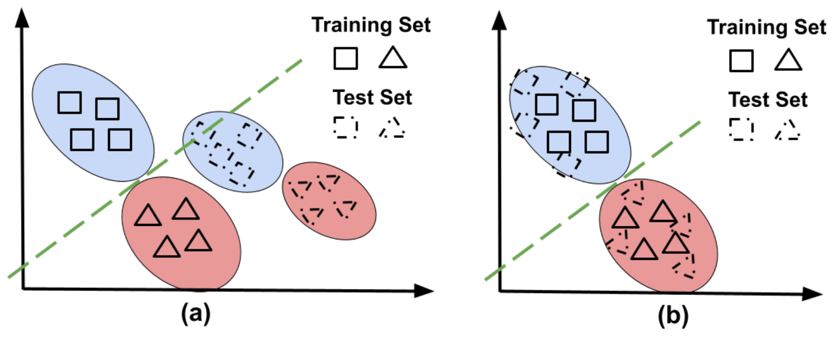


Figure 4.2: **(a)** The training and test set follow different probability distributions **(b)** After correcting for domain-shift, the original decision boundary now applies for both domains.

inference; see Figure 4.2(b).

### 4.1.1 Related work

The model presented in Figure 4.1 is closely related to the probabilistic versions of principal components analysis [114] and canonical correlation analysis [7]. In both cases, $Z \sim \mathcal{N}(0, I)$ and $X \mid Z \sim \mathcal{N}(Wz+\mu, \Psi)$. When the transformation matrix, $W$, is diagonal –*i.e.*, a location-scale transformation– it is possible to perform domain-shift adaptation without the Gaussianity assumption by minimizing the maximum mean discrepancy between both domains [42, 132]. In our study, we allow $W$ to be an arbitrary matrix without the Gaussianity assumption. Domain adaptation with arbitrary affine transformation has been explored in the context of mixing small datasets from different domains to increase the size of the training set. While this is often successful, this approach still requires a supervised dataset for each of the different sources [118]. Here, our objective is to learn a predictor in the source domain, and then apply it to the target domain in the unsupervised and semi-supervised scenarios.

Recent approaches attempt to minimize the divergence between source and target distributions by using data transformations. CORAL [110] matches the first two moments of the source and target distributions, while a different line of work uses variants of autoencoders to find a common mapping between source and target data [38, 14, 15, 81]. After finding this common feature space, they learned a predictor using only the source data. Their approach achieved better performance when applied to the target dataset, relative to not correcting for domain-shift, in natural language processing tasks.

Adversarial domain adaptation learns the mapping to the common space and the predictor at the same time [29, 80, 134]. It combines a discriminator (which distinguishes instances from the different domains); a predictor (which tries to minimize the prediction error on the labeled instances); and a third function (which maps the instances into a common space). The three functions are optimized together. If successful, the discriminator should be unable to distinguish the domain on an instance (based on its common space encoding), while the predictor should have good performance under the metric of interest [115].

Despite the success of neural networks for domain adaptation in natural language processing and computer vision tasks, it is hard to define what type of problems can be solved with this approach. For example, they might fail when $P_S(Y) \neq P_T(Y)$ [112]. Even when they successfully learn an invariant representation across domains that preserves the predictive power in the source domain, this do not guarantee a successful adaptation. It is possible to have invariant representations and small error in the training set, and still have a large error in the test set [135]. Therefore, it is important to explicitly determine under which conditions we expect an algorithm to work. Our goal with this chapter is to analyze the problem of domain-shift under affine transformations, and to propose an approach for domain adaptation for this scenario.

### 4.1.2 A simple example

Imagine that we want to build a classifier that can identify the biological sex of a person $Y \in \{male(M), female(F)\}$, given their weight, $X$. Also, imagine that we use two different scales, one that measures in lb and the other in kg. Figure 4.3 shows the histogram of the instances measured in each scale. Note that $p_{lb}(Y = M) = p_{kg}(Y = M) = 0.5$; however $p_{lb}(X < 70) \neq p_{kg}(X < 70)$. In Figure 4.3 we can observe that, while the 70 kg or less is a frequent weight for people, we do not expect to see many people with a weight below 70 lb.

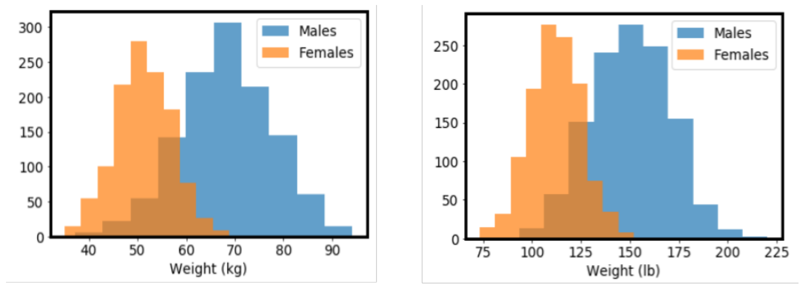


Figure 4.3: Left: Histogram of the instances weighted in the scale in kg. Right: Histogram of the instances weighted in the scale in lb.

Finally $P_{lb}(Y = M \mid X = 80) \neq P_{kg}(Y = M \mid X = 80)$. Note that a classifier trained on the kg data would predict male in this case, while one trained on the lb data would predict female.

We can set this problem in the context of the graphical model of Figure 4.1. Define $f(X;\lambda) = \lambda_0 + \lambda Z$. Then, by setting $\lambda_0^{(kg)} = \lambda_0^{(lb)} = 0$, $\lambda^{(kg)} = 1$, and $\lambda^{(lb)} = 2.2$ we can model the distributions of Figure 4.3. In this case, we define the latent space to be the measurements in kg. Therefore, we can go from the space in kg to the space in lb by multiplying the weight by 2.2. This solution is not unique. Note that, using our definition of $f(X;\lambda)$ we can set the latent space to be any of the measurements in kg, the measurements in pounds, the z-score (zero mean and standard deviation of 1), etc.

### 4.1.3 Objectives of batch effects correction

From the machine learning perspective, we want to correct for batch effects in order to get either more accurate classifiers by mixing dataset acquired from different sources, or to create a classifier, trained on a source domain, and then use that classifier on a target domain. These two cases are depicted in Figure 4.4.

The different scenarios depend on the labeled data available for training. In both cases, we have two inter-related objectives: (1) Find a transformation that maps the data from its original space to the common, latent space. (2) To find a latent space where we can learn an accurate classifier. In Sections 4.2 and 4.3 we describe different approaches for addressing batch effects for the two scenarios described above.

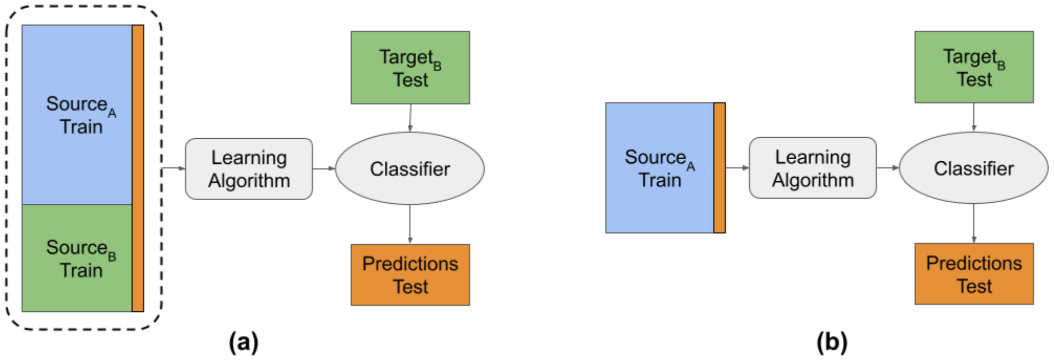


Figure 4.4: (a) The training set is from a source domain, while the test set is from a target domain (b) The training set contains labeled data from the source and target domain. The test set contains data from the target domain.

## 4.2 Supervised domain adaptation under affine transformations

In an effort to mitigate the "*small n, large p*" problem, many researchers aggregate data obtained from different sources into a single, larger dataset. Unfortunately, this dataset might be subject to batch effects possibly caused by hardware differences or imaging protocols [43, 93], and its main consequence in prediction studies is that researchers have observed a *decrease* in classification accuracy on multi-source studies compared with that obtained using a source [93, 122, 11].

In this section, we propose an approach to combine datasets from different sources when the discrepancies in their probability distributions is caused by affine transformations. We test the performance of our approach in the task of identifying people with schizophrenia given a functional magnetic resonance image (fMRI) of the brain.

### 4.2.1 Machine learning and functional connectivity graphs

The standard approach for applying machine learning to fMRI data begins by parcellating the (properly preprocessed) brain volumes into $m$ regions of interest. It then forms a symmetric $m \times m$ pairwise connectivity matrix, whose $(i, j)$ entry each correspond to some measure of statistical dependence between regions $i$ and $j$, whose upper-triangle is vectorized into a vector of length $p = \frac{1}{2}m(m-1)$.

The vectors corresponding to each of the $n$ subjects in the training set are arranged into a matrix $X$ of dimensions $n \times p$. Similarly, a vector $Y$ of length $n$, contains the labels of $X$. Finally, this labeled training data $(X,\ Y)$ is given to a learning algorithm, that produces the final classifier. A detailed description of this procedure can be found elsewhere [102, 122].

A critical aspect in assessing the impact of batch effects in classification studies, as well as the effectiveness of the techniques applied to removed them, is the methodology used to measure the performance of the classifiers. Some studies pool together the data from the different sites and then randomly split the data into a training and test set, while others use the data from $(r-1)$ sites for training and the $r^{th}$ site for testing [34, 90, 93, 1]. The first approach might mask the influence of batch effects because it artificially makes the distribution of the training set and test set more similar. This is an unrealistic scenario. In a real application, a clinician cares about the performance of the classifier on the patients that s/he is evaluating. The second approach is more realistic, but also more complicated and it will be addressed in Section 4.3.

Therefore, we propose a third evaluation scenario: Fix the test set to be a specific subset of the data from site A. Then consider two training sets: just the remaining instances from site-A versus those remaining site-A instances *and also the instances from site B.* This approach, illustrated in Figure 4.5, has the advantage of identifying if there is a benefit of mixing data from different sites, or if it is better to train one classifier independently for every site. Note that this methodology requires having a labeled dataset from both scanning sites.

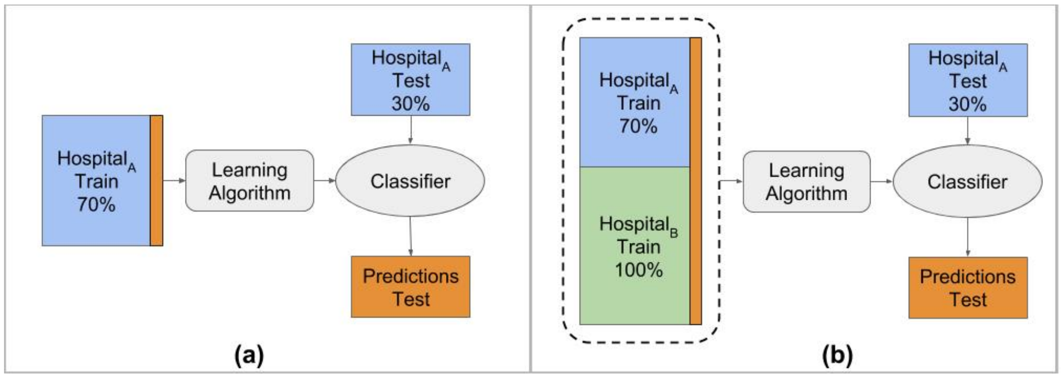


Figure 4.5: Evaluating a classifier in single site (a) and multi-site (b) scenarios.

### 4.2.2 Batch effects correction techniques

#### Adding site as covariate

This technique involves augmenting each instance with its site information – encoded as a 1-hot-encoding. (That is, using $r$ additional bit features, where the $j^{th}$ feature is 1 if that instance comes from the $j^{th}$ site, and the other features here are 0.)

When using a linear classifier, this method assumes that the only difference between sites is in the threshold that we use to classify an instance as belonging to one class, or another. If we assume that the decision function for one site is given by $w^T x = 0$, where the $x$ vector represent the features and $w$ is the vector of the coefficients (or weights) of the features, then the decision function for a second site is given by $w^T x + c = 0$. This method is effective when the batch effect is caused by a translation (adding a constant) to each instance of the dataset, but it will be ineffective otherwise. Figure 4.6(a) shows an illustration of this case. Note how the learned decision boundary is appropriate for one of the sites (red), but suboptimal for the other (blue). Note that this technique forces both decision boundaries to have the same slope, and only the bias changes.

#### Z-score normalization

This approach modifies the probability distribution of the features extracted from *both* sites, A and B, by making the values of each individual feature, for each site, zero-mean with unit variance – i.e., for each site, for the $i^{th}$ feature, subject its mean (for that site), and divide by its empirical standard deviation (for that site). Using this technique, only the marginals are the same in both sites, but the covariance structure is not. Applying this "Z-score normalization" to the data from every scanning site independently, will effectively remove batch effects caused by translation and scaling of features (see Appendix C.1). However it fails with more complex transformations, such as rotations or linear transformations in general; see Figure 4.6(b). Note that this scaling and translation is in the feature space, and so it is different to the affine transformations that are corrected during the preprocessing stage (which are applied in the coordinate space).

**Whitening**

Whitening is a linear transformation that can be viewed as a generalization of the z-score normalization. Besides making the mean of every feature equal to zero and its variance equal to one, it also removes the correlation between features by making the overall covariance matrix the identity matrix. One of the most common procedures to perform this process is *PCA Whitening* [69]. This transformation first rotates the data, in each site, by projecting it into its principal components, and then it scales the rotated data by the square root of its eigenvalues (which represent the variance of each new variable in the PCA space). Applying this whitening transformation to every dataset independently will remove the batch effects caused by a rotation and translation of the datasets, since in this cases the principal components of the different sites will be aligned; see Appendix C.2 for the mathematical derivation. However, since there is no guarantee that the principal components will be aligned in general, it might not work with other linear transformations; see Figure 4.6(c).

### 4.2.3 Solving linear transformations

Note that z-score normalization and whitening solve specific cases of

$$X_B = \alpha X_A + \beta \quad \alpha \in \mathbb{R}^{p\times p}, \quad \beta \in \mathbb{R}^p \tag{4.1}$$

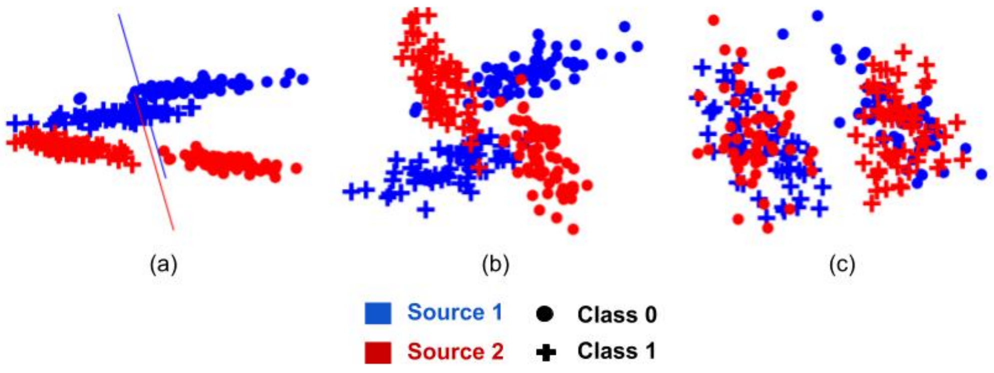


Figure 4.6: Examples of linear transformation where the methods fail. (a) Including site as covariate, (b) z-score normalization, (c) whitening.

(corresponding to Equation C.2 in Appendix C.2.) Z-score solves batch effects when the associated matrix $\alpha$ is diagonal, while whitening solves them when $\alpha$ is orthogonal with determinant 1. Nevertheless, both methods fail to solve batch effects for a general matrix $\alpha$. Note also that the previous approaches did not explicitly compute $\alpha$ and $\beta$, but instead, applied a transformation that removed their effects under the specified circumstances. Of course, if we could compute $\alpha$ and $\beta$, or even a good approximation $\hat{\alpha}$ and $\hat{\beta}$, we could then solve for any batch effect corresponding to an arbitrary linear transformation.

For any two random vectors $X_A$ and $X_B$, such that $X_B = \alpha X_A + \beta$ :

$$\begin{aligned} \mu_B &= E[X_B] &= E[\alpha X_A + \beta] &= \alpha E[X_A] + \beta &= \alpha \mu_A + \beta \\ \Sigma_B &= COV[X_B] &= COV[\alpha X_A + \beta] &= \alpha COV[X_A] \alpha^T &= \alpha \Sigma_A \alpha^T \end{aligned} \tag{4.2}$$

Although we can obtain empirical estimates of $\mu_A$, $\mu_B$, $\Sigma_A$, $\Sigma_B$ from the dataset, the problem is in general ill-defined – *i.e.*, there is an infinite number of solutions. Now note that every site includes (at least) two different subpopulations – *e.g.*, healthy controls versus cases (perhaps people with schizophrenia). Each subpopulation has its own mean vector and covariance matrix ($\mu_A^{HC}, \mu_A^{SCZ}$, $\mu_B^{HC}, \mu_B^{SCZ}$, and $\Sigma_A^{HC}, \Sigma_A^{SCZ}, \Sigma_B^{HC}, \Sigma_B^{SCZ}$) . A reasonable assumption is that the batch effects affect both populations in the same way, but by computing the mean and covariance matrix of every population and site independently we are effectively increasing the number of equations available. We can then get an estimate for $\alpha$ and $\beta$ as follows:

$$\hat{\alpha},\ \hat{\beta} \quad = \quad \arg\min_{\alpha,\beta} \sum_{j\in\{HC,SCZ\}} \sqrt{p}||\mu_B^j-(\alpha\mu_A^j+\beta)||_2+||\Sigma_B^j-(\alpha\Sigma_A^j\alpha^T)||_F \quad (4.3)$$

where $p$ is the dimensionality of the feature set, and $||\cdot||_F$ is the Frobenius norm of a matrix. Note that it is possible to combine data from more than two datasets by finding a linear transformation for every pair of sites.

### 4.2.4 Experiments and Results

#### Dataset

We applied the four aforementioned methods to the task of classifying healthy controls and people with schizophrenia using the data corresponding to the Auditory Oddball task to the FBIRN phase II dataset, which is a multisite study developed by the Function Biomedical Informatics Research Network (FBIRN). Keator *et al.* provides a complete description of the study [67].

After preprocessing the data, we eliminated the subjects who presented head movement greater than the size of one voxel at any point in time in any of the axis, a rotation displacement greater than 0.06 radians, or that did not pass a visual quality control assessment. The original released data contains scans extracted from 6 different scanning sites; however, we only used 4 of them. One of the sites was discarded because it lacked T1-weighted images, which were required as part of our preprocessing pipeline. The second discarded site contained only 6 subjects (5 with schizophrenia) after the quality control assessment, so it was not suitable for our analysis. In summary, we have 21 participants from Site 1, 22 from Site 2, 23 from Site 3 and 23 from Site 4. In all cases, the proportion of healthy controls vs people with schizophrenia is $\sim 50\%$.

#### Empirical evaluation

To obtain the feature vector of every fMRI scan, we used the subset corresponding to the Fronto-Parietal Network for a total of $k = 25$ out of the 264 regions of interest defined by Power *et al.* [99]. The time series corresponding

Table 4.1: Average accuracy after correcting batch effects. The number in entry $(i, j)$ is the accuracy, over instances from the target site $i$, of the classifier learned by adding all of site $j$ to the training subset of site $i$. The colored cells indicate results whose difference improves (green) or decrease (red) relative to the single site classification.

| | S 1 | S 2 | S 3 | S 4 |
|---|---|---|---|---|
| S 1 | 62.8 | 72.3 | 65.7 | 67.3 |
| S 2 | 67.8 | 66.4 | 70.0 | 59.5 |
| S 3 | 55.0 | 60.9 | 58.3 | 56.9 |
| S 4 | 62.3 | 57.8 | 76.4 | 71.4 |

(a) No correction

| | S 1 | S 2 | S 3 | S 4 |
|---|---|---|---|---|
| S 1 | 62.8 | 70.7 | 64.7 | 68 |
| S 2 | 67.1 | 66.4 | 68.1 | 57.6 |
| S 3 | 55.7 | 57.6 | 58.3 | 56.9 |
| S 4 | 67.1 | 57.6 | 75.7 | 71.4 |

(b) Site as covariate

| | S 1 | S 2 | S 3 | S 4 |
|---|---|---|---|---|
| S 1 | 62.8 | 64.7 | 57.6 | 65.7 |
| S 2 | 68.5 | 66.4 | 67.6 | 62.3 |
| S 3 | 48.0 | 54.0 | 58.3 | 58.0 |
| S 4 | 63.5 | 56.0 | 74.0 | 71.4 |

(c) Z-score normalization

| | S 1 | S 2 | S 3 | S 4 |
|---|---|---|---|---|
| S 1 | 62.8 | 55.7 | 52.8 | 49.5 |
| S 2 | 51.6 | 66.4 | 52.1 | 50.4 |
| S 3 | 54.2 | 54.2 | 58.3 | 53.8 |
| S 4 | 50.7 | 47.3 | 52.6 | 71.4 |

(d) Whitening

| | S 1 | S 2 | S 3 | S 4 |
|---|---|---|---|---|
| S 1 | 62.8 | 65.9 | 66.4 | 66.2 |
| S 2 | 66.6 | 66.4 | 67.8 | 67.8 |
| S 3 | 49.5 | 50.2 | 58.3 | 51.4 |
| S 4 | 73.5 | 72.8 | 73.5 | 71.4 |

(e) Linear transformation

to every region was simply the average time series of all the voxels inside the region. In order to obtain the functional connectivity matrix, we computed the Pearson's correlation between the time series of all $\binom{k}{2}$ pairs of regions.

We produced classifiers using a support vector machine (SVM) with linear kernel using the SVMLIB library [13]. The parameters of the SVM were set using cross validation. We applied the batch effect correction techniques previous to merging the datasets into a single training set, and repeated the experiment 15 times with different train/test splits. All the parameters required for the batch effects correction techniques were obtained using only the training sets. Table 4.1 reports the average accuracy over the 15 rounds.

In each of the sub-tables in Table 4.1, the $(i, j)$ entry represent the average accuracy when the training set has instances from the $i$th and $j$th site, and the test set has instances only from the $i$th site. Ideally, all the off-diagonal values should be higher than the diagonal ones; however, this is not the case. In most of the cases we have mixed and inconsistent results. The only method that consistently improves the performance of the classifiers is the one that solves for arbitrary linear transformations (Table 4.1e). Note that site S3 is an exception, where we do not see any improvement; however, this particular site has a low performance even in the single site scenario. It is likely that the signal in this particular site is too low and cannot be properly detected by the used methods.

These results reinforce the idea that batch effects play predominant role in classification studies, and motivate the need to develop techniques that address them in order to be able to effectively combine multi-site datasets. We can additionally conclude that whitening, z-score normalization and adding the site as covariate are insufficient to solve batch effects in fMRI data. Our method for solving linear transformations is the one who consistently improves the results in a multi-site scenario, indicating that it is a step in the right direction.

## 4.3 Unsupervised domain adaptation under affine transformations

The approach described in Section 4.2 is useful for merging dataset when we have access to a labeled dataset from the target source; however, that is not always the case. In this section we tackle the scenario where we train a predictor based on a training set from the source domain, and then use it to make predictions on a dataset from a target domain.

Under the assumption that the mapping from source and target domains to a common representation is an affine function, we propose an algorithm for unsupervised and semi-supervised domain adaptation. We find the parameters $\lambda$ and $\lambda_0$, which project the data into a common space, by computing the first $p$ eigenvectors of the covariance matrices of the probability distributions of each domain, and then finding an orthogonal matrix that minimizes the maximum mean discrepancy between both distributions.

There is an inherent unidentifiability problem with unsupervised domain adaptation. Observe in Figure 4.7 that, in the absence of labeled data from the source and target domains, it is impossible to distinguish between the different "distribution alignments" presented there. This problem can be alleviated in the *semi*-supervised case, where few labeled instances allow the distinction between both scenarios.

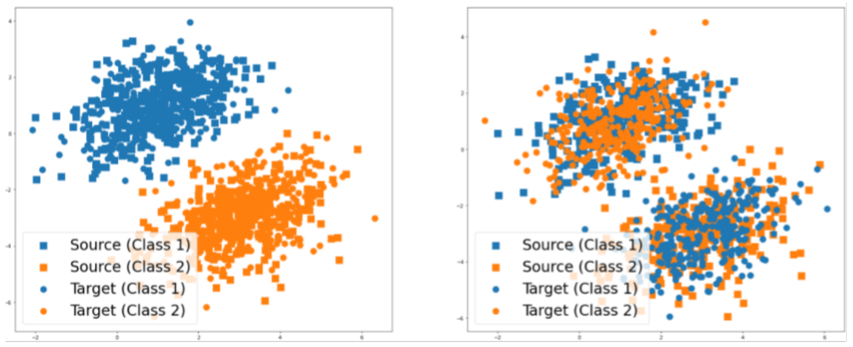


Figure 4.7: Anti-alignment example. In the unsupervised case it is impossible to distinguish between the scenarios of the left (correct alignment) and right (anti-alignment) graphs.

### 4.3.1 Domain-shift adaptation via linear transformations

Under the assumption that the domain-shift is caused by affine transformations, the equations on Figure 4.1 become:

$$\begin{gathered} x_A = \theta_A z + \mu_A \\ x_B = \theta_B z + \mu_B \\ x_A \in \mathbb{R}^m, \;\; x_B \in \mathbb{R}^n, \;\; z \in \mathbb{R}^p \end{gathered} \tag{4.4}$$

Note that the latent variable, $Z$, can have a different (lower) dimensionality than the observations $x_A$ and $x_B$, which in turn can have different dimensionality between themselves. Importantly, we do not assume that we have paired data between the source and target domain. In other words, for a given instance, $i$, we can either observe its representation in the source domain, $x_A^i$, or the target domain, $x_B^i$, but not both.

If we knew the parameters $\theta_A, \mu_A, \theta_B, \mu_B$, and assuming they are non-degenerate, we could do the inverse mapping from the observations $x_A$ and $x_B$ to $z$ by solving the following optimization problem:

$$z^* = \arg\min_z ||(x - \mu) - \theta z||^2$$

whose solution (see the Appendix C.3) is given by

$$z = (\theta^T \theta)^{-1} \theta^T (x - \mu) \tag{4.5}$$

Once we map the data from the source and target domains to a common space, we can use the labeled data from the source domain to learn a predictor that we can apply to data from any of both domains (after the appropriate projection into the common space).

**Estimating the transformation parameters**

Without loss of generality, we assume that $E[Z] = 0$ and $Cov[Z] = I$. Then, given a dataset with instances drawn from the source domain $X_A$ and a dataset with instances drawn from the target domain $X_B$:

$$\begin{aligned} E[X] &= \theta E[Z] + \mu = \mu \\ Cov[X] &= \theta Cov[Z] \theta^T = \theta\theta^T \end{aligned} \tag{4.6}$$

Note that we can compute the empirical estimates $\hat{\mu}_A, \hat{\mu}_B, \hat{\Sigma}_A, \hat{\Sigma}_B$, given the datasets $X_A$ and $X_B$. The empirical estimators of the mean directly give us half of the transformation parameters. For the case of the covariance matrix, we can compute the singular value decomposition:

$$\begin{aligned} \Sigma &= USU^T \\ \Sigma &= US^{\frac{1}{2}}S^{\frac{1}{2}}U^T \\ \Sigma &= US^{\frac{1}{2}}QQ^T S^{\frac{1}{2}}U^T \quad ;\text{where} \quad QQ^T = I \\ \Sigma &= (US^{\frac{1}{2}}Q)(US^{\frac{1}{2}}Q)^T \end{aligned} \tag{4.7}$$

Since $\Sigma$ is a positive semi-definite matrix, its eigenvalues are non-negative, which allows us to decompose the diagonal matrix as $S = S^{\frac{1}{2}}S^{\frac{1}{2}}$. By comparing Equations 4.6 and 4.7, we can estimate the parameters $\theta$ as:

$$\theta = US^{\frac{1}{2}}Q \tag{4.8}$$

for any orthogonal matrix, $Q$. After substituting the parameter $\theta$ into Equation 4.5, then applying some algebraic manipulations (see the Appendix C.4) , we observe that:

$$z = Q^T S^{-\frac{1}{2}} U^T (x - \mu) \tag{4.9}$$

A consequence of Equation 4.9 is that matching the empirical mean and covariance matrices of the source and target domain is not enough to correct for domain-shift adaptation: The orthogonal matrix $Q$, which represents rotations or reflections, might cause misalignment in the data; see Figure 4.8(a).

The matrices $(S_A, U_A)$ and $(S_B, U_B)$, for the source and target domains, respectively, can be computed from the SVD of their empirical covariance matrices, $\hat{\Sigma}_A$ and $\hat{\Sigma}_B$. Since the objective of domain adaptation is to align the distributions, regardless of the "direction" of the alignment, we arbitrarily set $Q_A = I$. Then, we find an orthogonal matrix that minimizes the divergence between both probability distributions:

$$Q_B^* = \arg\min_{Q_B} Div(X_A \mid\mid Q_B^T X_B), \quad s.t. \ \ Q_B Q_B^T = I \tag{4.10}$$

where $Div(\cdot||\cdot)$ is an empirical measure of the divergence between the two domains.

### Maximum Mean Discrepancy

A common measure of the divergence between two probability distributions is the Maximum Mean Discrepancy (MMD) [42].

**Definition 1** (Maximum Mean Discrepancy)**.** *Let $p$ and $q$ be Borel probability measures defined on a domain $\mathcal{X}$. Given observations $X := \{x_1, \ldots, x_m\}$ and $Y := \{y_1, \ldots, y_n\}$, drawn independently and identically distributed from $p$ and $q$, respectively. Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$, the MMD is defined as:*

$$MMD[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left( E_{x \sim p}[f(x)] - E_{y \sim q}[f(y)] \right)$$

Informally, the purpose of the MMD is to determine if two probability distributions, $p$ and $q$, are different. The associated algorithm involves taking samples from $p$ and $q$, then finding a function that take large values on samples from $p$ and small (or negative) values on samples of $q$. The MMD is then the difference between the mean values of the function of the samples.

By defining the class of functions $\mathcal{F}$ as the unit ball in a reproducing kernel Hilbert space, Gretton et al. (2012) proposed (biased) empirical estimator of the MMD$^2$ as follows:

$$\mathrm{MMD}_b^2[\mathcal{F}, X, Y] = \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(y_i, y_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i, y_j) \tag{4.11}$$

where $k(\cdot,\cdot)$ is a valid kernel. In our case, we use the Gaussian kernel $k(x, y) = \exp\left(-\frac{1}{2}\sigma^{-2}||x - y||^2\right)$.

Equation 4.11 has two nice properties. (1) It computes an estimation of the MMD with a finite number of instances from each domain. (2) It is a differentiable function, so it can be optimized with iterative methods, such as gradient descent.

### Optimization with orthogonality constraints

For solving the optimization problem with orthogonality constraints presented in Equation 4.10, we used the Wen and Yin (2013) algorithm (Algorithm 1), which is an iterative method based on the Cayley transform. Their algorithm is similar to gradient descent, but instead of looking for solutions in the Euclidean space, they look for solutions in the Stiefel manifold, which is the set that contains all the orthogonal matrices.

Formally, their proposed algorithm solves:

$$\min_{X \in \mathbb{R}^{n \times p}} \mathcal{F}(X), \ \ s.t. \ \ X^T X = I \tag{4.12}$$

where $\mathcal{F} : \mathbb{R}^{n \times p} \to \mathbb{R}$ is a differentiable function. For our purposes, $\mathcal{F}(Q_B) = \mathrm{MMD}^2(Z_A, Q_B^T Z'_B)$, where $Z_A$ and $Z'_B$ are the projections of $X_A$ and $X_B$, respectively, using Equation 4.9 with $Q_A = Q'_B = I$. $Q_B$ is an orthogonal (rotation or reflection) matrix that multiplies $Z'_B$.

**Algorithm 1** Optimization with orthogonality contraints

**Input**: $\mathcal{F}, X_0$
**Parameter**: Learning rate ($\tau$), Max iterations (M)
**Output**: $\arg\min_X \mathcal{F}(X)\ s.t.\ X^T X = I$

```
 1: Given an initial orthogonal matrix $X_0$.
 2: Let $t = 0$
 3: while $t < M$ do
 4:     Compute the Gradient $G_t = \mathcal{DF}(X_t) = \left(\frac{\partial \mathcal{F}(X_t)}{\partial X_t i,j}\right)$
 5:     Compute $A_t = G_t X_t^T - X_t G_t^T$
 6:     Compute $Q_t = \left(I + \frac{\tau}{2} A_t\right)^{-1} \left(I - \frac{\tau}{2} A_t\right)$
 7:     Compute $X_{t+1} = Q_t X_t$
 8:     Update $t := t + 1$
 9: end while
10: return $X_M$
```

Algorithm 1 is guaranteed to converge when the learning rate ($\tau$) meets the Armijo-Wolfe conditions [91]. However, it is not guaranteed to find the global minimum of $\mathcal{F}(X)$. Similarly to gradient descent approaches, the algorithm might converge to a local minimum. One heuristic to alleviate this problem is to perform multiple restart with different seed points; however, this is still not guaranteed to convergence to the global minimum.

**Unsupervised domain adaptation**

For finding the parameters $\theta_A$ and $\theta_B$, we can arbitrarily set $Q_A = I$ in Equation 4.8, and then use Algorithm 1, with the Maximum Mean Discrepancy, for computing $Q_B$:

$$Q_B^* = \arg\min_{Q_B} \mathrm{MMD}_b^2(Z_A, Q_B^T Z_B'),\ \ s.t.\ Q_B Q_B^T = I \tag{4.13}$$

where $Z_B'$ is a dataset that contains the transformed instances $x_B$ using $Q_B' = I$. Finally, we can project the source and target domains to a common representation using Equation 4.9.

Note that Algorithm 1 requires the gradient of the $\mathrm{MMD}^2$ with respect to the matrix $Q_B^T$. By applying standard matrix calculus we compute (see Appendix C.5 for details):

**Algorithm 2** Unsupervised domain adaptation with linear transformations

**Input**: $X_A \in \mathbb{R}^{i \times m}, X_B \in \mathbb{R}^{j \times n}$. Every row in these matrices represents an instance.
**Parameter**: Variance of Gaussian kernel ($\sigma^2$)
**Output**: $Z_A \in \mathbb{R}^{i \times p}, Z_B \in \mathbb{R}^{j \times p}$. Every row in each matrix represents an instance in the shared space.

```
1: Compute $\mu_A$ and $\Sigma_A$ of the dataset $X_A$.
2: Compute $\mu_B$ and $\Sigma_B$ of the dataset $X_B$.
3: $U_A, S_A, V_A = \mathrm{SVD}(\Sigma_A)$
4: $U_B, S_B, V_B = \mathrm{SVD}(\Sigma_B)$
5: $\theta_A = U_A S_A^{\frac{1}{2}}$ {Use only the positive eigenvalues (and their corresponding eigenvectors)}
6: $\theta'_B = U_B S_B^{\frac{1}{2}}$ {Use only the positive eigenvalues (and their corresponding eigenvectors)}
7: $Z_A = (\theta_A^T \theta_A)^{-1} \theta_A^T (X_A - \mu_A)$
8: $Z'_B = (\theta_B'^T \theta'_B)^{-1} \theta_B'^T (X_B - \mu_B)$
9: Use Algorithm 1 to find the $Q_B^T \in \mathbb{R}^{p \times p}$ that minimizes Equation 4.13. Compute the MMD using a Gaussian kernel with variance $\sigma_2$.
10: $Z_B = Z'_B Q_B$
11: **return** $Z_A, Z_B$
```

$$\begin{aligned} G(Q_B^T) &= \frac{\partial \mathrm{MMD}^2(Z_A, Q_B^T Z'_B)}{\partial Q_B^T} \\ &= -\frac{2}{mn} \sum_{i,j}^{n,m} \exp\left(-\frac{1}{2\sigma_2} ||z_A^{(i)} - Q_B^T z_B^{(j)}||^2\right) \left(\frac{z_A^{(i)} z_B^{(j)T}}{\sigma^2}\right) \end{aligned} \tag{4.14}$$

Algorithm 2 shows the procedure to map the source and target domain into a common space in an unsupervised way (The code is publicly available; see Appendix A ). For notation, the source domain (resp., target, shared space) is $m$-dimensional space, (resp. $n$-dimensional, $p$-dimensional; here we assume that $p \leq \min(m, n)$. For mapping into this lower dimensional space, we project the data into the first $p$ eigenvectors of the empirical covariance matrice $\Sigma_A$ (resp. $\Sigma_B$). The eigenvalues corresponding these eigenvectors are positive, while the other $m - p$ and $n - p$ eigenvalues will be equal to zero.

After mapping both domains to a common space, we can use the labels of the source domain to learn a predictor, and then use it to make predictions in the target domain. Section 4.3.2 will show that the Maximum Mean

Discrepancy is not convex with respect to the orthogonal matrix $Q_B^T$, meaning Algorithm 1 might converge to a local minimum. Additionally, in the unsupervised case there is an inherent identifiability problem caused by the missing labels [21, 72] – *i.e.*, there are $\theta_B$ and $\theta'_B$ such that $P(\ f(X, \theta_B)\ ) = P(\ f(X, \theta'_B)\ )$. This can create an "anti-alignment" problem; see Figure 4.7.

Note than when the "anti-alignment" occurs, the source and target domains have the same marginal probability $P_S(Z) = P_T(Z)$, but $P_S(Y \mid Z) \neq P_T(Y \mid Z)$. In other words, a classifier learned on the source domain will have good performance on more data from the same domain; however, it will have very poor performance on the target domain. Zhao et al. (2019) shows that aligning the marginal probability of the covariates, then learning a good predictor on the source domain, is not sufficient for successfully performing domain adaptation.

### Semi-supervised domain adaptation

If we have access to a few labeled instances in the target domain, we might reduce the chance of converging to an "anti-alignment". Since the MMD is not convex with respect to the rotation matrix $Q_B^T$, a common strategy is to attempt multiple re-start (with different seed points) of an iterative optimization algorithm. We then choose the one with the lowest cost. In the unsupervised case, the MMD itself is the cost function. For the semi-supervised case, we can first run Algorithm 2 for each seed point, then learn a predictor, using only the labeled data from the source domain. Then, for every alignment generated by each of the seed points, evaluate those predictors on the labeled data of the target domain, and choose the one with the lowest error.

Alternatively, we could incorporate the cross entropy loss of the source and target domains into Equation 4.13. This approach requires optimizing a weighted linear combination of three terms in the loss function: the MMD, the cross-entropy in the source domain, and the cross-entropy in the target domain. Since this path requires setting these three extra weights, we limited our experiments to the first approach.

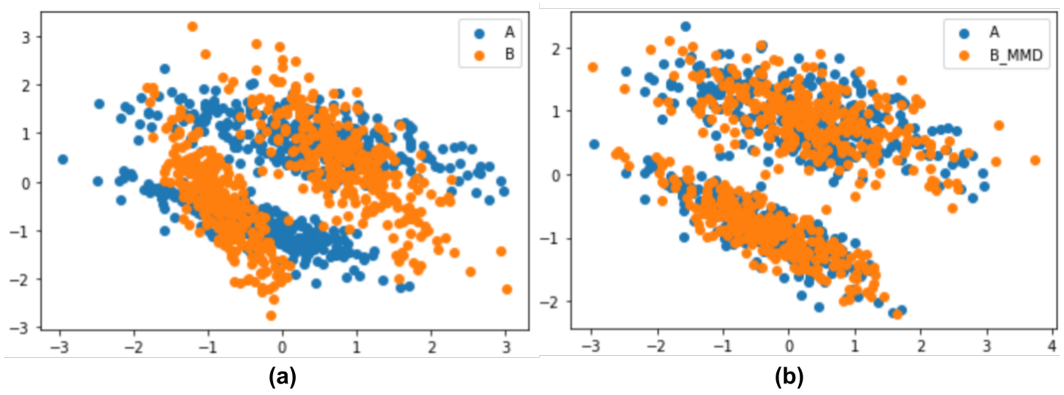


Figure 4.8: Results of using Algorithm 2 (a) without correcting for the rotation, (b) after correcting for the rotation.

### 4.3.2 Experiments and Results

We first show the performance of our approach to perform domain-shift adaptation in a simulated dataset, and then, in a modified version of the MNIST digit classification task.

#### Simulated data

For the simulated data we sampled 600 instances to creaet a dataset, $D_z \in \mathbb{R}^{600\times 2}$, from a mixture of bi-variate Gaussians with parameters $\mu_1 = \begin{bmatrix}1\\1\end{bmatrix}$, $\mu_2 = \begin{bmatrix}5\\-5\end{bmatrix}$, $\Sigma_1 = \begin{bmatrix}2 & 0.7\\0.7 & 1\end{bmatrix}$, $\Sigma_2 = \begin{bmatrix}2 & 1\\1 & 4\end{bmatrix}$. These instances correspond to a common shared space $Z$. We then created two random transformation matrices $\theta_A, \theta_B \in \mathbb{R}^{5\times 2}$, and two random translation vectors $\mu_A, \mu_B \in \mathbb{R}^5$ to create the observations. Of course, neither the real parameters, nor the instances in the shared space are visible to our algorithm.

We randomly divided $D_z$ into two disjoint datasets $D_A$ (source domain) and $D_B$ (target domain) with 300 instances each. Then, we created the datasets $X_A = D_A\theta_A^T + \mu_A$ and $X_B = D_B\theta_B^T + \mu_B$. Our algorithm only sees $X_A$ and $X_B$, which each contain 5-dimensional vectors.

Figure 4.8(b) shows the result of applying Algorithm 2 to the simulated datasets $X_A$ and $X_B$. Figure 4.8(a), on the other hand, shows the effect of ignoring the effect of the orthogonal matrix $Q_B^T$. In this last case we successfully mapped both datasets into the same lower-dimensional space, and that both

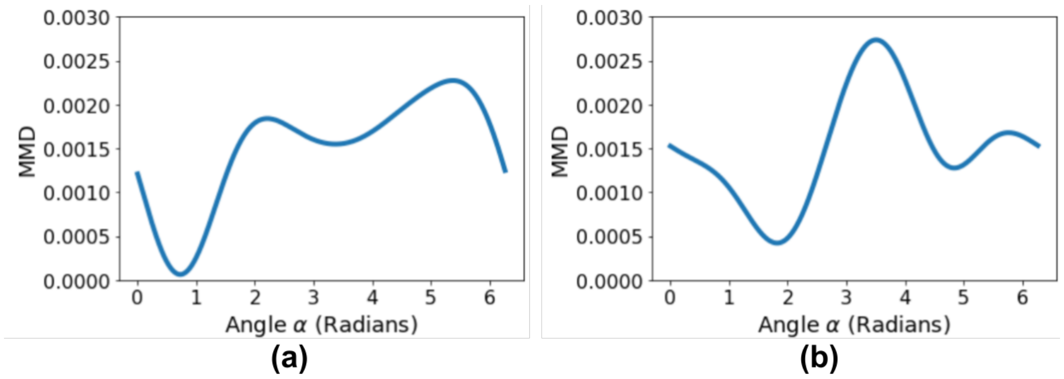


Figure 4.9: MMD of the simulated data for (a) rotation matrices and (b) reflection matrices.

datasets have zero mean and an identity covariance matrix; however, they are not aligned. By finding the orthogonal matrix that minimizes the MMD between the source and target datasets we can obtain the correct alignment.

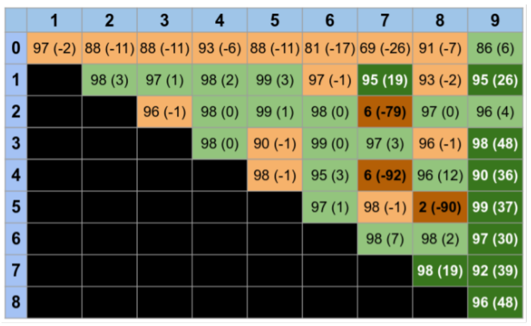
As mentioned in Section 4.3.1, the maximum mean discrepancy is not convex with respect to the orthogonal matrix $Q_B^T$. For 2-dimensional spaces, an orthogonal matrix is either a rotation $R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$ or a reflection $S = \begin{bmatrix} -\cos\alpha & -\sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$ [128]. Figure 4.9 shows the MMD between the projection of $X_A$ into $Z_A$ and the rotated (or reflected) projection of $X_B$ into $Z_B$ at different angles. Note that we have a total of 4 local minima for the simulated data. The global minimum corresponds to the proper alignment, shown in Figure 4.8(b). Similar to gradient descent, Algorithm 2 might converge to a local minimum depending on the seed point.

## Binary digit classification

The second experiment is a variation of the digit classification task with the dataset MNIST (source domain) [75] and USPS (target domain) [58]. We simplified the task from 10-class digit classification to 45 binary digit classification (0 vs 1, 0 vs 2, ... , 8 vs 9).

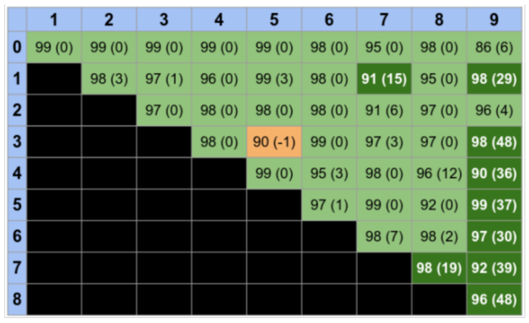
We first trained a 10-class convolutional neural network on the training data of the source domain (60,000 images) to create image embeddings in a 20-dimensional space. The convolutional neural network contained 4 convolutional layers (32, 128, 256 and 512 filters, respectively), each followed by a

MaxPooling layer ($2 \times 2$ kernel). Then we added a fully connected layer with 20 hidden neurons, and finally the output layer with 10 output neurons. While the output layer used a softmax activation function, the other layers used a rectified linear unit (ReLU) as the activation function; see Appendix A.



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 97 (-2) | 88 (-11) | 88 (-11) | 93 (-6) | 88 (-11) | 81 (-17) | 69 (-26) | 91 (-7) | 86 (6) |
| 1 | | 98 (3) | 97 (1) | 98 (2) | 99 (3) | 97 (-1) | 95 (19) | 93 (-2) | 95 (26) |
| 2 | | | 96 (-1) | 98 (0) | 99 (1) | 98 (0) | 6 (-79) | 97 (0) | 96 (4) |
| 3 | | | | 98 (0) | 90 (-1) | 99 (0) | 97 (3) | 96 (-1) | 98 (48) |
| 4 | | | | | 98 (-1) | 95 (3) | 6 (-92) | 96 (12) | 90 (36) |
| 5 | | | | | | 97 (1) | 98 (-1) | 2 (-90) | 99 (37) |
| 6 | | | | | | | 98 (7) | 98 (2) | 97 (30) |
| 7 | | | | | | | | 98 (19) | 92 (39) |
| 8 | | | | | | | | | 96 (48) |

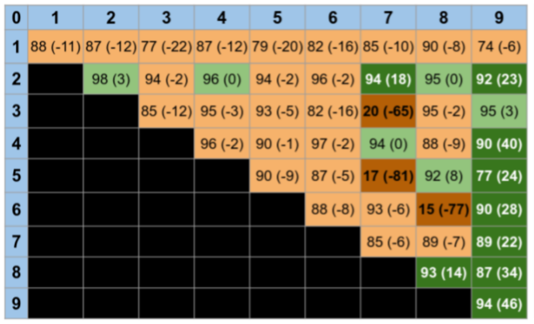
(a)



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 99 (0) | 99 (0) | 99 (0) | 99 (0) | 99 (0) | 98 (0) | 95 (0) | 98 (0) | 86 (6) |
| 1 | | 98 (3) | 97 (1) | 96 (0) | 99 (3) | 98 (0) | 91 (15) | 95 (0) | 98 (29) |
| 2 | | | 97 (0) | 98 (0) | 98 (0) | 98 (0) | 91 (6) | 97 (0) | 96 (4) |
| 3 | | | | 98 (0) | 90 (-1) | 99 (0) | 97 (3) | 97 (0) | 98 (48) |
| 4 | | | | | 99 (0) | 95 (3) | 98 (0) | 96 (12) | 90 (36) |
| 5 | | | | | | 97 (1) | 99 (0) | 92 (0) | 99 (37) |
| 6 | | | | | | | 98 (7) | 98 (2) | 97 (30) |
| 7 | | | | | | | | 98 (19) | 92 (39) |
| 8 | | | | | | | | | 96 (48) |

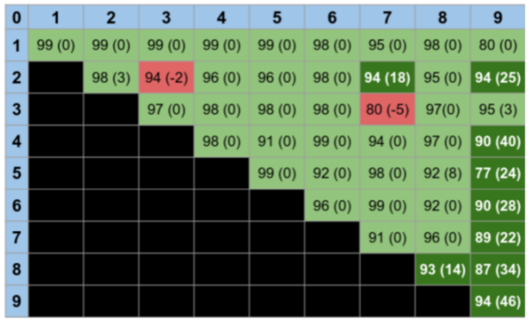
(b)

Figure 4.10: Results of (a) unsupervised and (b) semi-supervised domain-shift adaptation in binary digit classification.



| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 88 (-11) | 87 (-12) | 77 (-22) | 87 (-12) | 79 (-20) | 82 (-16) | 85 (-10) | 90 (-8) | 74 (-6) |
| 2 | | 98 (3) | 94 (-2) | 96 (0) | 94 (-2) | 96 (-2) | 94 (18) | 95 (0) | 92 (23) |
| 3 | | | 85 (-12) | 95 (-3) | 93 (-5) | 82 (-16) | 20 (-65) | 95 (-2) | 95 (3) |
| 4 | | | | 96 (-2) | 90 (-1) | 97 (-2) | 94 (0) | 88 (-9) | 90 (40) |
| 5 | | | | | 90 (-9) | 87 (-5) | 17 (-81) | 92 (8) | 77 (24) |
| 6 | | | | | | 88 (-8) | 93 (-6) | 15 (-77) | 90 (28) |
| 7 | | | | | | | 85 (-6) | 89 (-7) | 89 (22) |
| 8 | | | | | | | | 93 (14) | 87 (34) |
| 9 | | | | | | | | | 94 (46) |

(a)



| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 99 (0) | 99 (0) | 99 (0) | 99 (0) | 99 (0) | 98 (0) | 95 (0) | 98 (0) | 80 (0) |
| 2 | | 98 (3) | 94 (-2) | 96 (0) | 96 (0) | 98 (0) | 94 (18) | 95 (0) | 94 (25) |
| 3 | | | 97 (0) | 98 (0) | 98 (0) | 98 (0) | 80 (-5) | 97(0) | 95 (3) |
| 4 | | | | 98 (0) | 91 (0) | 99 (0) | 94 (0) | 97 (0) | 90 (40) |
| 5 | | | | | 99 (0) | 92 (0) | 98 (0) | 92 (8) | 77 (24) |
| 6 | | | | | | 96 (0) | 99 (0) | 92 (0) | 90 (28) |
| 7 | | | | | | | 91 (0) | 96 (0) | 89 (22) |
| 8 | | | | | | | | 93 (14) | 87 (34) |
| 9 | | | | | | | | | 94 (46) |

(b)

Figure 4.11: Results of (a) unsupervised and (b) semi-supervised domain-shift adaptation in binary digit classification in 13 dimensional space.

We used the output of the fully connected layer with 20 neurons as the image embeddings for both, the training data of the MNIST (60,000 images) and the test data USPS datasets (2,007 images). Then, for each of the 45 binary classification tasks, we compared three scenarios: (1) Baseline: learn the parameters of a logistic regression model using the MNIST dataset, and then test it on the USPS dataset. (2) Use Algorithm 2 to project the MNIST and USPS dataset into a common space of dimension 5. Then learn the parameters of a logistic regression model using only the projected data of the MNIST, and test it on the projected data of the USPS dataset. We chose a “small” dimensionality of the shared space because computing distances in high-dimensional spaces is harder because the instances sparsely populate the input space [27]. (3) Similar to the second scenario, but now assume that 10% of the data in the USPS dataset is labelled (semi-supervised case).

Figure 4.10(a) shows the classification accuracy of the unsupervised domain adaptation approach. The number in parenthesis indicates the improvement (or decrease) in performance relative to the Scenario 1. Similarly, Figure 4.10(b) shows the classification accuracy of the semi-supervised experiment.

### 4.3.3 Discussion

As expected, the results in Figure 4.10(a) show that even after the reducing the discrepancy between the source and target domain, and learning an accurate classifier on the source domain, this classifier is not guaranteed to generalize to the target domain. All the boxes in orange indicate that applying our algorithm for domain-shift had a lower performance than not doing any transformation at all. Specially interesting are the cases marked in dark orange, where we observe the effect of the “anti-alignment”. On the other hand, when the alignment is done properly, there are very significant improvements in the classification accuracy. Note that for unsupervised domain adaptation there is no way to distinguish between correct and incorrect alignments.

Figure 4.10(b), on the other hand, shows that we avoid incorrect alignments in the semi-supervised case. Having access to a small set of labelled data allows the algorithm to identify when no domain-shift adaptation is needed (because the classifier already generalizes to the target domain), or detect the “anti-alignments” and choose the proper alignment instead; see the case of 2 vs 7, where an improper alignment occurs in the unsupervised case, but the proper alignment of the semi-supervised case increases the classification accuracy 6%. In the case of proper alignments of the data the accuracy improved in essentially all the cases up to 48%.

The dimensionality of the shared space plays an important role when performing domain-shift adaptation. While Figures 4.10(a) and 4.10(b) show the performance obtained in a shared space of dimension 5, Figures 4.11(a) and 4.11(b) show the performance when the dimension of the shared space is the number of positive eigenvalues in the empirical covariance matrix (13 in our experiments). The performance of the unsupervised domain adaptation degrades significantly, while the semi-supervised approach remain roughly the same. We hypothesize that this decrease in performance is due to the difficulty of reliably estimating metrics on probability distributions in high dimensional spaces with a limited number of instances [27].

In summary, we present an algorithm for domain-shift adaptation caused by arbitrary affine transformations. Our approach first projects the data into a shared low-dimensional space using the first $p$ eigenvectors of the empirical covariance matrices of the data. Then, it find an orthogonal matrix that minimize the maximum mean discrepancy between the source and target data. For the unsupervised domain adaptation, there is an unavoidable identifiabiliy problem that can be alleviated by having a few labels of the target domain (semi-supervised domain adaptation). When using the correct othogonal matrix, this effectively maps both domains into a shared space where the $P(Z_A, Y) = P(Z_B, Y)$. In those cases, we can expect that a predictor learned using data from the source domain to generalize to data from a target domain.

# Chapter 5

# Style Transfer for Unsupervised Domain Adaptation in Ultrasound Image Segmentation

## 5.1 Introduction

Linear transformation might not be enough to model the discrepancy between source and target domains for the problem of image segmentation. Semantic image segmentation is the task of assigning every pixel in an image to one of $K$ possible classes. For medical images, these classes typically are anatomical structures of interest such as bones and organs, or anomalies within these anatomical structures, such as tumors or lesions.

Deep learning approaches, such as U-net [103], fully convolutional networks [79], and extensions of these models [84, 136, 100], have shown promising results for segmentation tasks. However, the generalization capabilities of deep models is closely tied to the amount and variety of the data used to train them [39]. Unfortunately, training data is scarce for many medical-related tasks [17, 44, 120]. This may force researchers to use training instances from other sources. Unfortunately, these training instances might not be a good representation of the data that will be present during inference.

Finding transformations that decrease the divergence between the probability distribution of source and target domains is a common technique in the field of domain adaptation [8]. Many techniques have been proposed to address this challenge [20]; however, the lack of labeled data in medical tasks presents a major challenge that limits the application of some of these approaches [44].

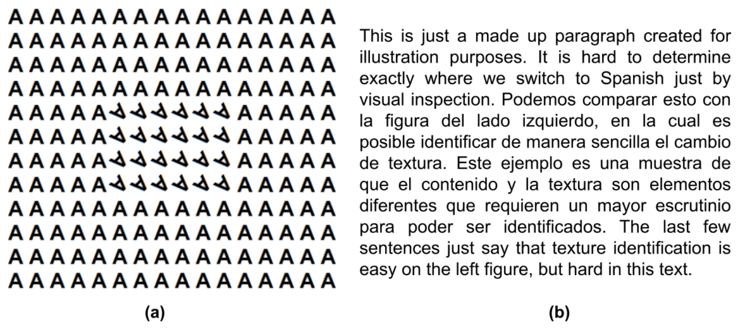
Recent advances in the field of style transfer [35] can also be framed in terms of the graphical model of Figure 4.1. The goal of style transfer is to synthesize a texture from a source image while preserving the semantic content of a target image [32]. Here, letting the random variable $Z$ represent that semantic content, we view $g(Z, \lambda)$ as the style. The objective of style transfer is then to create $X_B$ given $X_A$, $\theta_A$ and $\lambda_B$. Under the assumption that every scanner applies a different texture during the process of generating a medical image, the techniques of style transfer can be applied to reduce batch effects – here, to be able to use data from one source (using one style) to produce a model that can help label instances from a target domain (using a different style).

Two important advantages of style transfer for this purpose are that (1) they are *unsupervised* learning methods, and (2) they can be applied between a pair of images. Therefore, we can apply this approach even to small *unlabeled* datasets.

The main contribution of this chapter is to empirically show that this technology which was originally proposed for artistic purposes [35], is also useful for unsupervised domain adaptation in medical image segmentation. After successfully transferring the style (texture) from one domain to another, we demonstrate that a segmentation network trained using only data from the source domain, can still accurately segment images from the target domain. Importantly, there is no need to retrain the style transfer network, making it very useful for applications with limited target-domain data.

## 5.2 Foundations and Related Work

Informally, a texture is a set of visual patterns that have some common characteristics [64]. Research on visual perception suggests that texture discrimination and form recognition are two different, separable tasks [65]. In other words, the texture of an image is essentially independent from its content; see Figure 5.1.



This is just a made up paragraph created for illustration purposes. It is hard to determine exactly where we switch to Spanish just by visual inspection. Podemos comparar esto con la figura del lado izquierdo, en la cual es posible identificar de manera sencilla el cambio de textura. Este ejemplo es una muestra de que el contenido y la textura son elementos diferentes que requieren un mayor escrutinio para poder ser identificados. The last few sentences just say that texture identification is easy on the left figure, but hard in this text.

Figure 5.1: Texture discrimination and content recognition involve different processes. **(a)** Example with same content, different texture. **(b)** Example with same texture, different content. Note that half of the paragraph is written in Spanish.

A common assumption in texture modelling is that there is a joint probability distribution over the intensity of the pixels, $p(X)$, for each set of images that have perceptually similar texture appearance [138]. The objective is then to estimate $p(X)$ using a model, $f(X)$, learned from a set of observed images with this texture. Once learned, it is possible to draw samples from $f(X)$ to generate textures that are visually similar to the one of the training images.

Because of the high dimensionality of real world images it is difficult to estimate $p(X)$ with a limited number of images; instead $f(X)$ is used only to reproduce a set of observed statistics [138]. Formally, given a set of $K$ vector-valued functions $S = \{\phi_\alpha \mid \alpha \in \{1, \ldots, K\}\}$ we learn a model $f(X)$ such that $E_p[\phi_\alpha(I)] = E_f[\phi_\alpha(I)]$. We usually estimate $E_p[\phi_\alpha(X)]$ using a set of training images – *i.e.*, given a training set of images $D = \{X_1, X_2, \ldots, X_M\}$:

$$E_p[\phi_\alpha(X)] \approx \mu_\alpha = \frac{1}{M}\sum_{i=1}^{M}\phi_\alpha(X_i) \tag{5.1}$$

Common choices for $\phi_\alpha$ are a statistic (*e.g.*, mean or covariance) of the convolution of a filter $F_\alpha$ and an image $X$ [98], or the correlations between features responses in the convolutional layers of classification neural networks [31]. All the images generated by $p(X)$ are said to have the same texture, which is formally defined via the *Julesz Ensemble*, $(\Omega_{K,\epsilon})$ [139]:

$$\Omega_{K,\epsilon} = \{X : \mathcal{L}\left(\phi_\alpha(X), \mu_\alpha\right) \ \leq \epsilon \ \ \forall \alpha \in \{1, \ldots, K\}\} \tag{5.2}$$

where $\mathcal{L}$ is a loss function such as $L_1$ or $L_2$ distance.

Based on these ideas, there are algorithms to mix the content of an image (below denoted c) with the style (or texture) of a second image (denoted s) [32]. The objective is to find the image $x$ that solves:

$$x^* = \arg\min_x \quad \alpha \sum_{i,j} (\ F_{i,j}(x) - F_{i,j}(c)\ )^2 \ + \beta \sum_{l,i,j} \left(\ G^l_{i,j}(x) - G^l_{i,j}(s)\ \right)^2 \tag{5.3}$$

where $F_{i,j}(\cdot)$ is the feature of the *i-th* filter at position $j$ in a predefined convolutional layer, and $G^l_{i,j}(x) = \sum_k \langle F^l_{i,k}(x), F^l_{j,k}(x)\rangle$ is the inner product of the feature maps $i$ and $j$ at layer $l$, where $k$ indexes the pixel position inside a feature map. The weights $\alpha$ and $\beta$ control the trade-off between emulating the content and the style, respectively.

One drawback is that solving Equation (5.3) is computationally expensive. In order to accelerate this process, the optimization problem of Equation (5.3) can be approximated by a generative convolutional neural network [63, 117]. The input to these networks is a random noise image, and the output is an image with the desired style [40].

Although using convolutional neural networks improved the speed of the algorithms for style transfer, they required to train a different neural network for every style. Further research found that by applying conditional instance normalization, it was possible to share all the convolutional weights in a network to transfer different styles [23, 116]. The key was to tune the parameters of an affine transformation for each of the 32 styles that they modeled. After using that transformation to normalize the convolutional neural network it was possible to use a single neural network to transfer different styles.

Recently Ghiasi et al. [35] showed that a neural network could approximately learn the normalization parameters to transfer different styles. The input to this network is an image whose style we want to transfer to a content image, and the output is the normalization parameters. This allowed them to train a network that takes two arbitrary images as input (one for content, one for style), and merge them together in a very efficient way. This network is publicly available at [113].

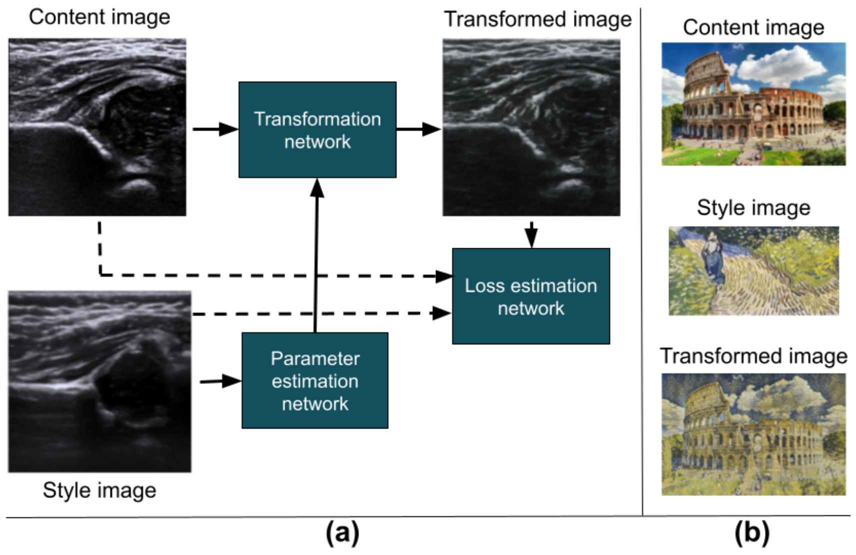


Figure 5.2: (a) Schematic diagram of the network proposed by Ghiasi et al. [35]. The loss estimation network is just used during training. (b) Style transfer between a photograph and a painting.

A key advantage of this network [35], depicted in Figure 5.2 (a), is that it can transfer the style between two arbitrary images without the need to retrain the network. This network was originally designed to transfer texture from painting to photographs, producing visually appealing results; see Figure 5.2 (b). Here, we explore its use for reducing the negative impact of batch effects for segmentation in medical images. Our metric for success is not the quality of the image, but rather an increase in the performance of a model (learned using these transformed images) for the automatic segmentation algorithms.

Specially relevant to our approach is the work of Zhang et al. [133], who modeled the domain adaptation problem as a noise style transfer task and applied the perceptual losses proposed by Johnson et al. [63] in an adversarial way. Similarly, other groups have explored using CycleGAN [137] to reduce the influence of batch effects [36, 83]. CycleGAN is also an adversarial model that attempts to perform image-to-image translation, a task that is closely related to style transfer. Unlike the approach that we propose, the previously mentioned methods need to be trained per each source-target domain pair.

## 5.3 Style transfer for domain adaptation

Given a labeled dataset, $D_S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, with pairs of images, $X_i$, and their corresponding segmentation masks, $Y_i$, acquired from one or more sources (*e.g.*, imaging devices), we want to learn an accurate predictor $Y = h(X)$ that will be evaluated on a disjoint target dataset $D_T = \{X_{n+1}, \ldots, X_m\}$ acquired with a different imaging device.

We propose an approach designed to work when a given imaging device generates images that are elements of the same Julesz ensemble (*i.e.*, they have the same texture), and that Julesz ensemble is different for each device (*i.e.*, images acquired with different scanners have a different texture). The idea of style transfer is to transform the images of the source scanner so they have a similar texture as the images of the target scanner. Specifically, before learning the segmentation function $h(\cdot)$, we transfer the style of the images in $D_T$ to the images in $D_S$ to create $D'_S = \{(X'_1, Y_1), \ldots, (X'_n, Y_n)\}$.

The style-transfered images $X'_i = g(X_i, X_{n+k})$ are created by computing $g(\cdot, \cdot)$, which is the pre-trained style-transfer neural network by Ghiasi et al. [35], to the images $X_i \in D_S$ and $X_{n+k} \in D_T$, where $X_{n+k}$ is an image randomly taken from $D_T$. Note that this procedure is unsupervised, since our algorithm does not have access to the labels of the target dataset, $D_T$, also we are assuming that the applied change in texture does not modify the original segmentation label $Y$ of the images in $D_S$.

Finally, we can learn $h(X)$ by using $D'_S$ along with a machine learning algorithm. For our experiments, $h(X)$ is learned by training a traditional U-Net architecture [103].

## 5.4 Experiments and Results

We evaluated the performance of style transfer for domain adaptation in the task of segmenting the acetabulum and femoral head in 2D ultrasound images of the hip. The identification of these anatomical structures is critical for the identification of problems such as developmental dysplasia of the hip.

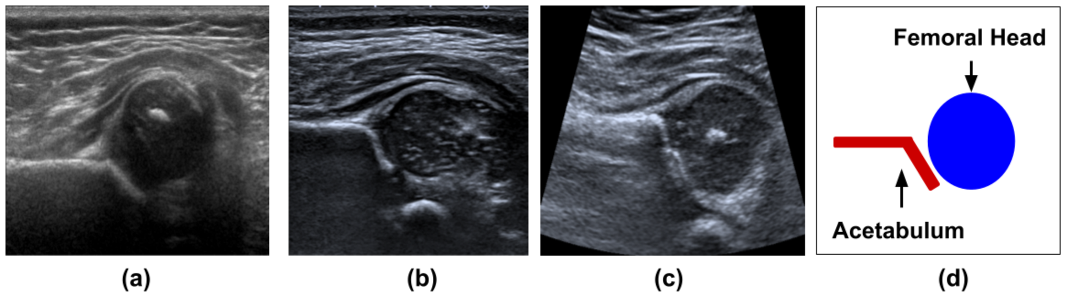


Figure 5.3: Images acquired with different scanners and the structures of interest. (a) Phillips (Linear), (b) Toshiba (Linear), (c) Toshiba (Conic), (d) Structures of interest.

The dataset contains 385 images acquired with a Phillips linear probe, 151 with a Toshiba linear probe, and 268 images with a Toshiba conic probe. Figure 5.3 shows some sample images from the dataset, as well as a diagram of the anatomical structures of interest. Note that the images from the different scanners look different, but in all of them the acetabulum and femoral head are easily recognizable.

We compared the performance of a U-net [103], in terms of Dice score, under two scenarios: (1) Training the U-net using the untouched data from one probe, and then test it on untouched data from a different probe; and (2) Applying style transfer to the training data, so it mimics the texture of the target dataset, and then test it on the untouched data of the target data. We repeated this experiment for every pair of probes, using each probe in each pair once for training and once for testing, for a total of 12 experiments (2 anatomical parts $\times$ 6 pairs of probes).

Table 5.1 compares both scenarios in terms of mean dice score, with standard deviation in parenthesis. In general, the models trained after applying style transfer perform better than their counterparts trained with the original, untouched data. In 10 of the 12 experiments the style-transfered data lead to a classifier up to 20% more accurate for the segmentation of the acetabulum, and up to 11% better for the femoral head. One of the experiments resulted in a tie of 90% Dice score, and in only one case did style-transfer decrease the performance –here by only 3%. Also, the standard deviation of the dice score tends to be smaller when using the models trained with style transfer.

Table 5.1: Mean dice score for the segmentation of the acetabulum (AC) and femoral head (FH). Columns identify the training set, while rows identify the test set. The numbers in parenthesis are the standard deviation.

| | **Training Set** | | | | | |
|---|---|---|---|---|---|---|
| | Phillips Linear | | Toshiba Linear | | Toshiba Conic | |
| | Original | Transf. | Original | Transf. | Original | Transf. |
| Phil-Lin-AC | - | - | 71 (13) | 68 (14) | 61 (17) | 66 (12) |
| Phil-Lin-FH | - | - | 68 (19) | 82 (11) | 77 (20) | 79 (15) |
| Tosh-Lin-AC | 62 (15) | 73 (9) | - | - | 60 (14) | 63 (11) |
| Tosh-Lin-FH | 90 (6) | 90 (5) | - | - | 83 (10) | 86 (8) |
| Tosh-Con-AC | 44 (13) | 64 (11) | 55 (18) | 60 (13) | - | - |
| Tosh-Con-FH | 80 (7) | 88 (10) | 81 (12) | 83 (8) | - | - |

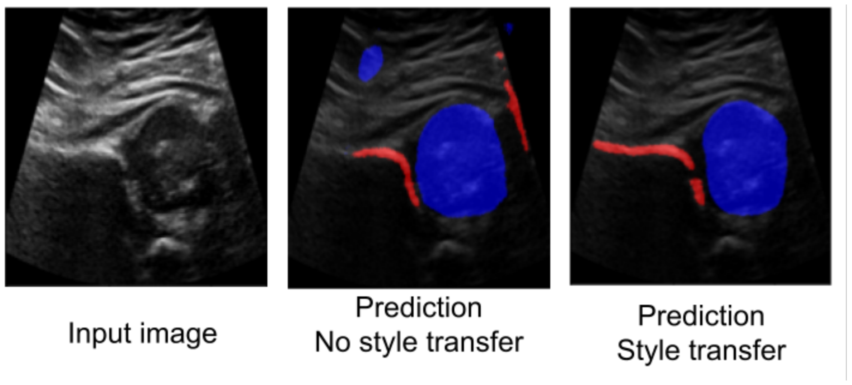


Figure 5.4: Visual comparison between the segmentations with models trained with and without style transfer

In other words, style transfer not only improves the segmentation in terms of dice score, but it is also more consistent in the predictions.

Figure 5.4 exemplifies the differences in the predictions made by the U-net trained with data from the Phillips linear probe, but then tested in images from a Toshiba conic probe. Note that the predictions made by applying style transfer before training the network greatly improves the produced segmentation mask not only numerically, but also visually.

## 5.5 Discussion

The diversity of image acquisition devices creates a challenge for learning models that can automatically segment medical images. The main challenge is that most machine learning methods assume that the data used during training follows the same probability distribution as the data used during inference, which is clearly not the case in this situation. The field of domain-adaptation explores techniques for decreasing the negative impacts of this discrepancy.

For cases where the differences between images acquired under different conditions can be modeled as a difference in texture, style-transfer offers a simple, yet effective solution: use a neural network to make the texture of the training set similar to the one of the test set. We explored the application of a pre-trained network, which was originally designed for artistic purposes [35], to improve the performance of a U-net architecture trained to segment the acetabulum and femoral head in ultrasound images of the hip. Using style transfer improved the performance up to 20% in terms of average Dice score while also reducing its variance. (After publication, we will release both, the dataset and the code to reproduce our results).

Two final remarks: (1) Since our proposed approach alters the texture of the images used for training, the task of interest should not rely on texture as a discriminating factor. This is not a problem for the segmentation of bones in ultrasound images, but there are other tasks where texture plays an important role (*e.g.*, tumor segmentation). We recommend using style-transfer for tasks where the relevant information is on the structure of the image, rather than in differences in texture. (2) Style-transfer is a computationally inexpensive operation. As it is also unsupervised, it does not require a labeled dataset from the target distribution. Importantly, pre-trained networks originally designed for artwork can be directly applied to new datasets, avoiding the need of training a style-transfer network, which usually requires a large amount of data.

# Chapter 6

# Conclusions

The machine learning technology has the potential to improve healthcare, but before it can produce models that can be used in clinical practice, we often need to address important challenges, some related to the limited amount of labeled data available for training, and the uncertainty around the target labels. This dissertation provides some steps in this direction, by developing learning methodologies that are sample efficient, allow the incorporation of prior knowledge, handle uncertainty in the labels, and correct for discrepancies between source and target domains.

For the case of image classification, we proposed using probabilistic labels when domain-specific information can be encoded as probabilities. These probabilistic labels provide more information per every training instance than traditional categorical labels, allowing machine learning to learn more accurate predictors. We provided empirical studies showing that deep learning architectures, when trained with probabilistic labels, increased their performance up to 22%, in terms of accuracy, in three binary classification tasks: diagnosis of hip dyslpasia, diagnosis of fatty liver, and diagnosis of glaucoma. Besides their improved accuracy, models trained with probabilistic labels were better calibrated than their counterparts trained with categorical, or other soft labels. This is important for medical applications, where predictors need to also report their confidence in their prediction.

For problems like time-series forecasting, we proposed to incorporate domain specific knowledge not in the labels, but in the models to be trained.

We developed SIMLR, a probabilistic graphical model that achieved state-of-the-art accuracy in predicting the number of new COVID-19 infections, 1- to 4-weeks in advance. SIMLR uses machine learning to learn the parameters of a time-varying epidemiological SIR model, which involves the likelihood of changing government policies in a certain geographical region, encoded using a probabilistic graphical model. The use of a prior epidemiological model, and a probabilistic models structure, helped SIMLR to make accurate predictions, despite using only a very small training dataset, with uncertain data.

Finally, we explored batch effects, which occur when technical noise obscures the real biological signal. For the case of medical images, batch effects might arise because of differences in the hardware used in imaging devices, or because of different imaging protocols. Machine learning views this problem as domain-shift, as the source and domain datasets follow different probability distributions. Here, however, there are functions that map each domain to a common space, where the probability distributions are equal. We explored domain shift adaptation caused by affine transformations in the supervised, unsupervised, and semi-supervised scenarios.

Although affine transformation might be too simple to explain batch effects in medical images, they provided important insights into the domain adaptation problem and allowed us to develop algorithms that improved the performance of binary classifiers in simulated data, a set of binary digit classification tasks, and diagnosis of schizophrenia based on fMRI data. Then, to deal with the domain shift caused by training a segmentation model on ultrasound images of the hip acquired with one imaging device, then applying this model to images acquired from a different device, we proposed an approach based on style-transfer, which proved able to deal with the domain-shift caused by differences in texture. This approach improved the Dice score by up to 20% (relative to not using domain adaptation at all.

Even though all the applications in this dissertation are related to the medical domain, we expect that the techniques shown here are applicable in other domains, especially when: (1) few training instances are available and expert knowledge can be encoded as probabilities, (2) there is a parametric model currently used by domain experts for analyzing a phenomenon, or (3) the discrepancy between the source and target domains is caused by linear transformations or differences in texture.

# References

[1] Alexandre Abraham, Michael P Milham, Adriana Di Martino, R Cameron Craddock, Dimitris Samaras, Bertrand Thirion, and Gael Varoquaux. Deriving reproducible biomarkers from multi-site resting-state data: An autism-based example. *NeuroImage*, 147:736–745, 2017.

[2] Government Alberta. Covid-19 alberta statistics. `https://www.alberta.ca/stats/covid-19-alberta-statistics.htm#data-notes`, 2021. Accessed: 2021-06-30.

[3] Cleo Anastassopoulou, Lucia Russo, Athanasios Tsakris, and Constantinos Siettos. Data-based analysis, modelling and forecasting of the covid-19 outbreak. *PloS one*, 15(3):e0230405, 2020.

[4] Mohammad R. Arbabshirani, Sergey Plis, Jing Sui, and Vince D. Calhoun. Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *Neuroimage*, 145:137 – 165, 2016. ISSN 10538119.

[5] Sercan Arik, Chun-Liang Li, Jinsung Yoon, Rajarishi Sinha, Arkady Epshteyn, Long Le, Vikas Menon, Shashank Singh, Leyou Zhang, and Martin Nikoltchev. Interpretable sequence learning for covid-19 forecasting. *NeurIPS*, 33, 2020.

[6] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[7] Francis R Bach and Michael I Jordan. A probabilistic interpretation of canonical correlation analysis. *Technical Report 688, Department of Statistics, University of California ...*, 2005.

[8] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *NeurIPS*, 19:137, 2007.

[9] HR Bhapkar, Parikshit N Mahalle, Nilanjan Dey, and KC Santosh. Revisited covid-19 mortality and recovery rates: are we missing recovery time period? *Journal of Medical Systems*, 44(12):1–5, 2020.

[10] Julie C Blackwood and Lauren M Childs. An introduction to compartmental modeling for the budding infectious disease modeler. *Letters in Biomathematics*, 5(1):195–221, 2018.

[11] Matthew R G Brown, Gagan S Sidhu, Russell Greiner, Nasimeh Asgarian, Meysam Bastani, Peter H Silverstone, Andrew J Greenshaw, and Serdar M Dursun. ADHD-200 global competition: diagnosing ADHD

using personal characteristic data can outperform resting state fMRI measurements. *Frontiers In Systems Neuroscience*, 6:69, 2012. ISSN 1662-5137.

[12] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[14] Minmin Chen, Zhixiang Xu, Kilian Q Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1627–1634, 2012.

[15] Minmin Chen, Kilian Q Weinberger, Zhixiang Xu, and Fei Sha. Marginalizing stacked linear denoising autoencoders. *JMLR*, 16(1):3849–3875, 2015.

[16] Yi-Cheng Chen, Ping-En Lu, Cheng-Shang Chang, and Tzu-Hsuan Liu. A time-dependent sir model for covid-19 with undetectable infected persons. *IEEE Transactions on Network Science and Engineering*, 7(4): 3279–3294, 2020.

[17] Veronika Cheplygina, Marleen de Bruijne, and Josien PW Pluim. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. *Medical image analysis*, 54:280–296, 2019.

[18] Estee Y Cramer, Yuxin Huang, Yijin Wang, Evan L Ray, Matthew Cornell, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Katie House, Dasuni Jayawardena, Abdul H Kanji, Ayush Khandelwal, Khoa Le, Jarad Niemi, Ariane Stark, Apurv Shah, Nutcha Wattanachit, Martha W Zorn, Nicholas G Reich, and US COVID-19 Forecast Hub Consortium. The united states covid-19 forecast hub dataset. *medRxiv*, 2021. doi: 10.1101/2021.11.04.21265886. URL `https://www.medrxiv.org/content/10.1101/2021.11.04.21265886v1`.

[19] Estee Y Cramer, Velma K Lopez, Jarad Niemi, Glover E George, Jeffrey C Cegan, Ian D Dettwiller, William P England, Matthew W Farthing, Robert H Hunter, Brandon Lafferty, et al. Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the us. *medRxiv*, 2021.

[20] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[21] Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *AISTATS'10*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

[22] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 20(5):533–534, 2020.

[23] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[24] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11 (Feb):625–660, 2010.

[25] Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 37 (2):505–515, 2017.

[26] Anthony S Fauci, H Clifford Lane, and Robert R Redfield. Covid-19—navigating the uncharted, 2020.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[28] Francisco Fumero, Silvia Alayón, José L Sanchez, Jose Sigut, and M Gonzalez-Hernandez. Rim-one: An open retinal image database for optic nerve evaluation. In *2011 24th international symposium on computer-based medical systems (CBMS)*, pages 1–6. IEEE, 2011.

[29] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1): 2096–2030, 2016.

[30] et al. Gao. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.

[31] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *NeurIPS*, 28:262–270, 2015.

[32] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *ICCVPR*, pages 2414–2423, 2016.

[33] Xin Geng. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748, 2016.

[34] Mina Gheiratmand, Irina Rish, Guillermo A Cecchi, Matthew R G Brown, Russell Greiner, Pablo I Polosecki, Pouya Bashivan, Andrew J Greenshaw, Rajamannar Ramasubbu, and Serdar M Dursun. Learning stable and predictive network-based patterns of schizophrenia and its clinical symptoms. *NPJ Schizophrenia*, 3:22, 2017. ISSN 2334-265X.

[35] Golnaz Ghiasi et al. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

[36] Amir Gholami et al. A novel domain adaptation framework for medical image segmentation. In *MICCAI Brainlesion*, pages 289–298. Springer, 2018.

[37] Maryellen L Giger. Machine learning in medical imaging. *Journal of the American College of Radiology*, 15(3):512–520, 2018.

[38] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[40] Ian Goodfellow et al. Generative adversarial nets. *NeurIPS*, 27, 2014.

[41] R Graf. Fundamentals of sonographic diagnosis of infant hip dysplasia. *Journal of pediatric orthopedics*, 4(6):735–740, 1984.

[42] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 13(1):723–773, 2012.

[43] Douglas N Greve, Gregory G Brown, Bryon A Mueller, Gary Glover, and Thomas T Liu. A survey of the sources of noise in fMRI. *Psychometrika*, 78(3):396 – 416, 2013. ISSN 00333123.

[44] Hao Guan and Mingxia Liu. Domain adaptation for medical image analysis: a survey. *arXiv preprint arXiv:2102.09508*, 2021.

[45] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[46] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[47] Humza Haider, Bret Hoehn, Sarah Davis, and Russell Greiner. Effective ways to build and evaluate individual survival distributions. *arXiv preprint arXiv:1811.11347*, 2018.

[48] Thomas Hale, Noam Angrist, Rafael Goldszmidt, Beatriz Kira, Anna Petherick, Toby Phillips, Samuel Webster, Emily Cameron-Blake, Laura Hallas, Saptarshi Majumdar, et al. A global panel database of pandemic policies (oxford covid-19 government response tracker). *Nature Human Behaviour*, 5(4):529–538, 2021.

[49] Okka W Hamer, Diego A Aguirre, Giovanna Casola, Joel E Lavine, Matthias Woenckhaus, and Claude B Sirlin. Fatty liver: imaging patterns and pitfalls. *Radiographics*, 26(6):1637–1653, 2006.

[50] H Theodore Harcke and B Pruszczynski. Hip ultrasound for developmental dysplasia: the 50% rule. *Pediatric radiology*, 47(7):817–821, 2017.

[51] Abhilash R Hareendranathan, Dornoosh Zonoobi, Myles Mabee, Dana Cobzas, Kumaradevan Punithakumar, Michelle Noga, and Jacob L Jaremko. Toward automatic diagnosis of hip dysplasia from 2d ultrasound. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 982–985. IEEE, 2017.

[52] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction.* Springer series in statistics. Springer, New York, 2009. ISBN 978-0-387-84857-0.

[53] William R Hendee, Gary J Becker, James P Borgstede, Jennifer Bosma, William J Casarella, Beth A Erickson, C Douglas Maynard, James H Thrall, and Paul E Wallner. Addressing overutilization in medical imaging. *Radiology*, 257(1):240–245, 2010.

[54] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[55] Søren Højsgaard, David Edwards, and Steffen Lauritzen. *Graphical models with R.* Springer Science & Business Media, 2012.

[56] Inga Holmdahl and Caroline Buckee. Wrong but useful—what covid-19 epidemiologic models can and cannot tell us. *New England Journal of Medicine*, 383(4):303–305, 2020.

[57] David W Hosmer and Stanley Lemesbow. Goodness of fit tests for the multiple logistic regression model. *Communications in statistics-Theory and Methods*, 9(10):1043–1069, 1980.

[58] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5): 550–554, 1994.

[59] Ehsan Imani and Martha White. Improving regression performance with distributional losses. *arXiv preprint arXiv:1806.04613*, 2018.

[60] John PA Ioannidis, Sally Cripps, and Martin A Tanner. Forecasting for covid-19 has failed. *International journal of forecasting*, 2020.

[61] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87, 1991.

[62] Xiaoyong Jin, Yu-Xiang Wang, and Xifeng Yan. Inter-series attention model for covid-19 forecasting. In *SIAM International Conference on Data Mining (SDM)*, pages 495–503. SIAM, 2021.

[63] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[64] Bela Julesz. Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92, 1962.

[65] Bela Julesz et al. Inability of humans to discriminate between visual textures that agree in second-order statistics. *Perception*, 2(4):391–405, 1973.

[66] Rahele Kafieh, Roya Arian, Narges Saeedizadeh, Zahra Amini, Nasim Dadashi Serej, Shervin Minaee, Sunil Kumar Yadav, Atefeh Vaezi, Nima Rezaei, and Shaghayegh Haghjooy Javanmard. Covid-19 in iran: forecasting pandemic using deep learning. *Computational and Mathematical Methods in Medicine*, 2021, 2021.

[67] David B. Keator and et al. The function biomedical informatics research network data repository. *NeuroImage*, 124, Part B:1074 – 1079, 2016. ISSN 1053-8119. Sharing the wealth: Brain Imaging Repositories in 2015.

[68] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[69] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

[70] Se Hyung Kim, Jeong Min Lee, Jong Hyo Kim, Kwang Gi Kim, Joon Koo Han, Kyoung Ho Lee, Seong Ho Park, Nam-Joon Yi, Kyung-Suk Suh, Su Kyung An, et al. Appropriateness of a donor liver with respect to macrosteatosis: application of artificial neural networks to us images—initial experience. *Radiology*, 234(3):793–803, 2005.

[71] Edward S Knock, Lilith K Whittles, John A Lees, Pablo N Perez-Guzman, Robert Verity, Richard G FitzJohn, Katy AM Gaythorpe, Natsuko Imai, Wes Hinsley, Lucy C Okell, et al. Key epidemiological drivers and impact of interventions in the 2020 sars-cov-2 epidemic in england. *Science Translational Medicine*, 2021.

[72] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

[73] Meelis Kull and Peter Flach. Patterns of dataset shift. In *LMCE at ECML-PKDD.*, 2014.

[74] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[75] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[76] Zhifang Liao, Peng Lan, Zhining Liao, Yan Zhang, and Shengzong Liu. Tw-sir: time-window based sir for covid-19 forecasts. *Scientific reports*, 10(1):1–15, 2020.

[77] HaiYue Liu and Aqsa et al Manzoor. The covid-19 outbreak and affected countries stock markets response. *Int'l J Environmental Research and Public Health*, 17(8):2800, 2020.

[78] Xinzhi Liu and Peter Stechlinski. Infectious disease models with time-varying parameters and general nonlinear incidence rate. *Applied Mathematical Modelling*, 36(5):1974–1994, 2012.

[79] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *ICCVPR*, pages 3431–3440, 2015.

[80] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1647–1657, 2018.

[81] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S Zemel. The variational fair autoencoder. In *ICLR*, 2016.

[82] Ian JC MacCormick, Bryan M Williams, Yalin Zheng, Kun Li, Baidaa Al-Bander, Silvester Czanner, Rob Cheeseman, Colin E Willoughby, Emery N Brown, George L Spaeth, et al. Accurate, fast, data efficient and interpretable glaucoma diagnosis with automated spatial analysis of the whole cup to disc profile. *PloS one*, 14(1):e0209409, 2019.

[83] Ilja Manakov et al. Noise as domain shift: Denoising medical images by unpaired image translation. In *D.A.R.T.M.I.L.L.L.I.D.*, pages 3–10. Springer, 2019.

[84] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[85] Ramesh Kumar Mojjada, Arvind Yadav, AV Prabhu, and Yuvaraj Natarajan. Machine learning models for covid-19 future forecasting. *Materials Today: Proceedings*, 2020.

[86] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019. URL `http://arxiv.org/abs/1906.02629`.

[87] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[88] Quang Nguyen, Hamed Valizadegan, and Milos Hauskrecht. Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*, pages 477–486. IEEE, 2011.

[89] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005.

[90] Jared A Nielsen, Brandon A Zielinski, P Thomas Fletcher, Andrew L Alexander, Nicholas Lange, Erin D Bigler, Janet E Lainhart, and Jeffrey S Anderson. Multisite functional connectivity mri classification of autism: Abide results. *Frontiers in human neuroscience*, 7:599, 2013.

[91] Jorge Nocedal and Stephen Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

[92] Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.

[93] Emanuele Olivetti, Susanne Greiner, and Paolo Avesani. ADHD diagnosis from multiple data sources with batch effects. *Frontiers In Systems Neuroscience*, 6:70, 2012. ISSN 1662-5137.

[94] Nahla F Omran, Sara F Abd-el Ghany, Hager Saleh, Abdelmgeid A Ali, Abdu Gumaei, and Mabrook Al-Rakhami. Applying deep learning methods on time-series data for forecasting covid-19 in egypt, kuwait, and saudi arabia. *Complexity*, 2021, 2021.

[95] Government Ontario. Covid-19 case data:glossary. `https://covid-19.ontario.ca/data/covid-19-case-data-glossary`, 2021. Accessed: 2021-06-30.

[96] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[97] Filippo Pesapane, Marina Codari, and Francesco Sardanelli. Artificial intelligence in medical imaging: threat or opportunity? radiologists again at the forefront of innovation in medicine. *European radiology experimental*, 2(1):35, 2018.

[98] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1):49–70, 2000.

[99] Jonathan D Power, Alexander L Cohen, Steven M Nelson, Gagan S Wig, Kelly Anne Barnes, Jessica A Church, Alecia C Vogel, Timothy O Laumann, Fran M Miezin, Bradley L Schlaggar, and Steven E Petersen. Functional network organization of the human brain. *Neuron*, 72(4):665 – 678, 2011. ISSN 1097-4199.

[100] Xuebin Qin et al. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.

[101] Pouria Ramazi, Arezoo Haratian, Maryam Meghdadi, Arash Mari Oriyad, Mark A Lewis, Zeinab Maleki, Roberto Vega, Hao Wang, David S Wishart, and Russell Greiner. Accurate long-range forecasting of covid-19 mortality in the usa. *Scientific Reports*, 11(1):1–11, 2021.

[102] Jonas Richiardi, Sophie Achard, Horst Bunke, and Dimitri Van De Ville. Machine learning with brain graphs: Predictive modeling approaches for functional imaging in systems neuroscience. *IEEE Signal Processing Magazine*, 30(3):58–70, 2013.

[103] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[104] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[105] KC Santosh. Covid-19 prediction models and unexploited data. *Journal of medical systems*, 44(9):1–4, 2020.

[106] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[107] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28, 2009.

[108] Amos Storkey et al. When training and test sets are different: Characterizing learning transfer. In *Dataset Shift in Machine Learning*, pages 3–28. Yale University Press in association with the Museum of London, 2008.

[109] Simon Strauss, Ella Gavish, Paul Gottlieb, and Ludmila Katsnelson. Interobserver and Intraobserver Variability in the Sonographic Assessment of Fatty Liver. *American Journal of Roentgenology*, 189(6):W320–W323, December 2007. ISSN 0361-803X, 1546-3141. doi: 10.2214/AJR.07.2123. URL `http://www.ajronline.org/doi/10.2214/AJR.07.2123`.

[110] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, volume 30, 2016.

[111] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[112] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *NeurIPS*, 33, 2020.

[113] Tensorflow. Fast Style Transfer for Arbitrary Styles. `https://www.tensorflow.org/hub/tutorials/tf2_arbitrary_image_stylization`, 2017. [Online; accessed 6-Feb-2022].

[114] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[115] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[116] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *ICCVPR*, pages 6924–6932, 2017.

[117] Dmitry Ulyanov et al. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016.

[118] Roberto Vega and Russ Greiner. Finding effective ways to (machine) learn fmri-based classifiers from multi-site data. In *U.I.M.L.M.I.C.A.*, pages 32–39. Springer, 2018.

[119] Roberto Vega and Russell Greiner. Domain-shift adaptation via linear transformations. *arXiv preprint arXiv:2201.05282*, 2022.

[120] Roberto Vega, Pouneh Gorji, Zichen Zhang, Xuebin Qin, Abhilash Rakkunedeth, Jeevesh Kapur, Jacob Jaremko, and Russell Greiner. Sample efficient learning of image-based diagnostic classifiers via probabilistic labels. In *International Conference on Artificial Intelligence and Statistics*, pages 739–747. PMLR, 2021.

[121] Roberto Vega, Leonardo Flores, and Russell Greiner. Simlr: Machine learning inside the sir model for covid-19 forecasting. *Forecasting*, 4(1): 72–94, 2022.

[122] Roberto I Vega Romero. The challenge of applying machine learning techniques to diagnose schizophrenia using multi-site fmri data. Master's thesis, University of Alberta, 2017.

[123] Patrick GT Walker, Charles Whittaker, Oliver J Watson, Marc Baguelin, Peter Winskill, Arran Hamlet, Bimandra A Djafaara, Zulma Cucunubá, Daniela Olivera Mesa, Will Green, et al. The impact of covid-19 and strategies for mitigation and suppression in low-and middle-income countries. *Science*, 369(6502):413–422, 2020.

[124] Gregory L Watson, Di Xiong, Lu Zhang, Joseph A Zoller, John Shamshoian, Phillip Sundin, Teresa Bufford, Anne W Rimoin, Marc A Suchard, and Christina M Ramirez. Pandemic velocity: Forecasting covid-19 in the us with a machine learning & bayesian time series compartmental model. *PLoS computational biology*, 17(3):e1008837, 2021.

[125] Junfeng Wen, Chun-Nam Yu, and Russell Greiner. Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. In *ICML*, pages 631–639. PMLR, 2014.

[126] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1):397–434, 2013.

[127] Miles N Wernick, Yongyi Yang, Jovan G Brankov, Grigori Yourganov, and Stephen C Strother. Machine learning in medical imaging. *IEEE signal processing magazine*, 27(4):25–38, 2010.

[128] David J Winter. *Matrix algebra*. Macmillan, 1992.

[129] Yanbing Xue and Milos Hauskrecht. Efficient learning of classification models from soft-label information by binning and ranking. In *The Thirtieth International Flairs Conference*, 2017.

[130] Arnold YS Yeung, Francois Roewer-Despres, Laura Rosella, and Frank Rudzicz. Machine learning–based prediction of growth in confirmed covid-19 infection cases in 114 countries using metrics of nonpharmaceutical interventions and cultural dimensions: Model development and validation. *Journal of Medical Internet Research*, 23(4):e26628, 2021.

[131] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[132] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML*, pages 819–827. PMLR, 2013.

[133] Tianyang Zhang et al. Noise adaptation generative adversarial network for medical image analysis. *IEEE transactions on medical imaging*, 39 (4):1149–1159, 2019.

[134] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *NeurIPS*, 31:8559–8570, 2018.

[135] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *ICML*, pages 7523–7532. PMLR, 2019.

[136] Zongwei Zhou et al. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018.

[137] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCVPR*, pages 2223–2232, 2017.

[138] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.

[139] Song Chun Zhu, Xiu Wen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a" trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):554–569, 2000.

# Appendix A

# Code Availability

## A.1 SIMLR Code

The code for reproducing the main results of this manuscript are publicly available at: `https://github.com/rvegaml/SIMLR`.

There are six jupyter notebooks on that repository. All the experiments were run using an e2-standard-4 (4 vCPUs, 16 GB memory) computer in the Google Cloud Platform.

- **CDC_models.ipynb**: It contains the code used to compile the predictions of the models submitted to the CDC. The dataset required to run this script was not included due to the size, but it is publicly available.
- **Comparison_CDC.ipynb**: It contains the code to create the graphs that compare SIMLR with the models submitted to the CDC. It uses the files created by the previous notebook.
- **Model_Canada_Provinces.ipynb**: It contains the data to predict the number of cases 1 to 4 weeks in advance in the 6 biggest provinces in Canada.
- **Model_US_Country.ipynb**: Similar to the previous one, but for the predictions on US at the country level.
- **Model_US_States.ipynb**: Similar to the previous one, but for the predictions on US at the state level.
- **SIR_Simulations.ipynb**: Code to create the simulated SIR, and to show how a simple SIR model with time-varying parameters can describe the complexities of the COVID-19 dynamics.

## A.2 Unsupervised domain adaptation code

The provided repository in addition contains the in-house developed python library *MLib*. This library contains custom code for inference in probabilistic graphical models.

The code for reproducing the results presented in this paper is publicly available at `https://github.com/rvegaml/DA_Linear`. It contains three main elements:

- **MLib:** An in-house developed library that contains the implementation of Algorithms 1 and 2, and auxiliary functions required to reproduce our main results.
- **Simulations.ipynb:** A jupyter notebook with the code to reproduce the simulated experiments.
- **BinaryDigits.ipynb:** A jupyter notebook with the code to reproduce our results with MNIST dataset.

During our experiments, we did not tune any parameter. The CNN was trained for a maximum of 500 epochs, using a learning rate of $10^{-5}$, and the default parameters of the Adam Optimizer. For the computation of the MMD we used a Gaussian kernel with $\sigma^2 = 2$.

# Appendix B

# Additional SIMLR tables

Table B.1: Comparison of MAPE between different models across all the states in the US 1 week in advance. The number in parenthesis represents the standard deviation of the MAPE.

| | 1 Week | | | | | |
|---|---|---|---|---|---|---|
| State | tf-v-SIR | SLOW | SIMLR | LNQ-ens1 | Best | Rank |
| Alabama | 20(16) | 19(16) | 20(12) | 21(15) | 20(12) | 1/16 |
| Alaska | 16(13) | 18(15) | 17(15) | 18(10) | 15(14) | 4/15 |
| Arizona | 21(18) | 25(19) | 22(21) | 18(16) | 18(16) | 3/16 |
| Arkansas | 20(18) | 21(29) | 24(29) | 19(19) | 19(19) | 13/16 |
| California | 15(11) | 20(15) | 13(10) | 13(10) | 13(10) | 1/16 |
| Colorado | 15(15) | 19(11) | 16(12) | 13(8) | 13(8) | 2/16 |
| Connecticut | 17(12) | 19(10) | 17(11) | 23(17) | 17(11) | 1/16 |
| Delaware | 20(14) | 18(14) | 19(13) | 15(11) | 15(11) | 4/16 |
| Washington DC | 23(15) | 19(13) | 23(15) | 15(10) | 15(10) | 8/16 |
| Florida | 12(11) | 13(7) | 12(8) | 9(7) | 9(7) | 2/16 |
| Georgia | 16(12) | 16(13) | 16(14) | 16(15) | 16(15) | 3/16 |
| Hawaii | 27(22) | 23(15) | 25(17) | 18(13) | 18(13) | 13/15 |
| Idaho | 16(11) | 16(10) | 14(10) | 14(10) | 14(10) | 2/16 |
| Illinois | 13(12) | 17(10) | 12(9) | 12(8) | 12(9) | 1/17 |
| Indiana | 11(10) | 17(10) | 15(10) | 13(11) | 13(11) | 3/17 |
| Iowa | 23(18) | 21(15) | 22(15) | 20(22) | 20(14) | 5/16 |
| Kansas | 16(15) | 20(15) | 18(12) | 21(14) | 18(12) | 1/16 |
| Kentucky | 16(11) | 16(8) | 15(9) | 12(9) | 12(9) | 2/16 |
| Louisiana | 24(17) | 23(22) | 24(22) | 21(19) | 21(19) | 3/16 |
| Maine | 17(15) | 19(15) | 18(15) | 14(11) | 14(11) | 2/16 |
| Maryland | 14(12) | 15(12) | 13(12) | 11(7) | 11(7) | 2/16 |
| Massachusetts | 15(10) | 16(11) | 13(9) | 14(10) | 13(9) | 1/16 |
| Michigan | 15(10) | 20(10) | 16(11) | 19(11) | 16(11) | 1/16 |
| Minnesota | 19(17) | 21(16) | 20(14) | 15(12) | 15(12) | 4/16 |
| Mississippi | 19(16) | 17(16) | 19(15) | 16(12) | 16(12) | 5/16 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Missouri | 20(14) | 19(13) | 21(15) | 12(38) | 11(36) | 14/16 |
| Montana | 19(17) | 21(12) | 19(15) | 35(104) | 18(13) | 2/16 |
| Nebraska | 20(18) | 20(16) | 20(15) | 18(13) | 18(13) | 5/16 |
| Nevada | 18(17) | 20(15) | 20(15) | 15(11) | 15(11) | 5/16 |
| New Hampshire | 18(14) | 18(13) | 16(14) | 17(11) | 16(14) | 1/16 |
| New Jersey | 11(10) | 13(10) | 11(9) | 14(10) | 11(9) | 1/16 |
| New Mexico | 15(10) | 20(12) | 15(11) | 15(11) | 15(11) | 2/16 |
| New York | 12(9) | 14(10) | 13(8) | 11(9) | 11(9) | 2/16 |
| North Carolina | 12(10) | 14(10) | 13(9) | 12(9) | 12(9) | 2/16 |
| North Dakota | 22(22) | 23(24) | 23(23) | 16(13) | 16(13) | 8/16 |
| Ohio | 12(9) | 16(10) | 13(10) | 11(8) | 11(8) | 2/16 |
| Oklahoma | 22(23) | 24(25) | 23(24) | 15(11) | 15(11) | 13/16 |
| Oregon | 19(13) | 18(13) | 18(13) | 13(10) | 13(10) | 4/16 |
| Pennsylvania | 13(11) | 15(12) | 15(11) | 11(8) | 11(8) | 3/17 |
| Rhode Island | 14(11) | 17(11) | 13(11) | 23(15) | 13(11) | 1/16 |
| South Carolina | 16(13) | 16(11) | 16(13) | 12(8) | 12(8) | 7/16 |
| South Dakota | 18(12) | 17(14) | 17(11) | 15(10) | 15(10) | 2/16 |
| Tennessee | 18(15) | 19(15) | 22(16) | 18(12) | 18(13) | 12/16 |
| Texas | 24(22) | 23(28) | 25(29) | 20(18) | 20(21) | 7/16 |
| Utah | 14(14) | 17(11) | 16(13) | 11(10) | 11(10) | 7/16 |
| Vermont | 25(20) | 20(15) | 21(14) | 21(15) | 21(14) | 1/16 |

Table B.2: Comparison of MAPE between different models across all the states in the US 2 weeks in advance. The number in parenthesis represents the standard deviation of the MAPE.

| | | | 2 Weeks | | | |
|---|---|---|---|---|---|---|
| State | tf-v-SIR | SLOW | SIMLR | LNQ-ens1 | Best | Rank |
| Alabama | 32(27) | 32(30) | 32(27) | 30(19) | 30(24) | 3/16 |
| Alaska | 30(32) | 30(25) | 27(25) | 28(22) | 27(24) | 2/15 |
| Arizona | 41(32) | 46(37) | 38(36) | 32(28) | 32(28) | 4/16 |
| Arkansas | 39(56) | 40(61) | 45(61) | 32(43) | 30(28) | 14/16 |
| California | 22(20) | 41(31) | 24(21) | 25(19) | 24(21) | 1/16 |
| Colorado | 31(28) | 30(19) | 33(26) | 24(19) | 24(19) | 11/16 |
| Connecticut | 27(25) | 29(18) | 29(26) | 33(18) | 29(26) | 1/16 |
| Delaware | 26(19) | 26(19) | 26(19) | 20(16) | 20(16) | 5/16 |
| Washington DC | 34(22) | 26(16) | 34(23) | 23(13) | 23(13) | 8/16 |
| Florida | 20(14) | 22(11) | 20(13) | 14(10) | 14(10) | 3/16 |
| Georgia | 25(18) | 31(19) | 27(19) | 22(20) | 22(20) | 4/16 |
| Hawaii | 41(38) | 32(30) | 39(36) | 29(23) | 28(23) | 7/15 |
| Idaho | 25(24) | 27(20) | 24(23) | 24(16) | 24(23) | 1/16 |
| Illinois | 23(18) | 31(19) | 27(19) | 23(16) | 23(16) | 3/17 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Indiana | 27(21) | 31(23) | 31(23) | 24(22) | 23(16) | 13/17 |
| Iowa | 36(45) | 33(21) | 33(26) | 34(32) | 31(24) | 3/16 |
| Kansas | 32(28) | 35(29) | 33(30) | 24(17) | 24(17) | 5/16 |
| Kentucky | 26(22) | 28(14) | 25(22) | 19(15) | 19(15) | 7/16 |
| Louisiana | 31(35) | 31(39) | 31(39) | 29(24) | 29(24) | 3/16 |
| Maine | 34(28) | 31(27) | 34(30) | 23(18) | 23(18) | 6/16 |
| Maryland | 24(18) | 26(19) | 23(18) | 22(16) | 22(16) | 3/16 |
| Massachusetts | 26(18) | 28(19) | 25(19) | 24(16) | 24(16) | 2/16 |
| Michigan | 33(22) | 35(19) | 33(20) | 31(16) | 27(16) | 4/16 |
| Minnesota | 40(34) | 39(32) | 41(35) | 28(23) | 28(23) | 10/16 |
| Mississippi | 26(23) | 32(25) | 31(24) | 22(18) | 22(18) | 11/16 |
| Missouri | 32(30) | 29(26) | 31(27) | 18(41) | 13(38) | 14/16 |
| Montana | 34(29) | 35(20) | 36(28) | 30(25) | 26(18) | 13/16 |
| Nebraska | 29(22) | 32(20) | 30(20) | 27(14) | 27(14) | 3/16 |
| Nevada | 31(22) | 37(25) | 33(26) | 23(18) | 23(18) | 5/16 |
| New Hampshire | 29(23) | 32(18) | 30(24) | 28(16) | 28(16) | 2/16 |
| New Jersey | 19(14) | 23(13) | 19(14) | 25(13) | 19(14) | 1/16 |
| New Mexico | 29(23) | 36(20) | 30(24) | 25(21) | 25(21) | 4/16 |
| New York | 24(18) | 24(15) | 24(18) | 21(13) | 21(13) | 4/16 |
| North Carolina | 22(14) | 26(18) | 25(18) | 17(14) | 17(14) | 6/16 |
| North Dakota | 42(39) | 41(42) | 48(44) | 32(24) | 31(20) | 13/16 |
| Ohio | 25(22) | 30(19) | 29(24) | 20(15) | 20(15) | 10/16 |
| Oklahoma | 34(30) | 34(32) | 37(31) | 25(21) | 25(21) | 13/16 |
| Oregon | 29(24) | 28(18) | 30(24) | 18(15) | 18(15) | 10/16 |
| Pennsylvania | 29(19) | 27(16) | 31(19) | 19(14) | 19(14) | 9/17 |
| Rhode Island | 21(17) | 29(19) | 24(17) | 30(19) | 24(17) | 1/16 |
| South Carolina | 27(19) | 26(20) | 27(21) | 18(13) | 18(13) | 13/16 |
| South Dakota | 30(26) | 30(28) | 32(25) | 27(20) | 27(20) | 4/16 |
| Tennessee | 30(24) | 29(26) | 34(27) | 24(19) | 24(19) | 12/16 |
| Texas | 38(49) | 35(52) | 38(51) | 26(26) | 25(34) | 8/16 |
| Utah | 27(29) | 30(20) | 32(27) | 20(19) | 20(19) | 10/16 |
| Vermont | 29(24) | 26(22) | 28(25) | 29(25) | 27(23) | 3/16 |

Table B.3: Comparison of MAPE between different models across all the states in the US 3 weeks in advance. The number in parenthesis represents the standard deviation of the MAPE.

| | | | 3 Weeks | | | |
|---|---|---|---|---|---|---|
| State | tf-v-SIR | SLOW | SIMLR | LNQ-ens1 | Best | Rank |
| Alabama | 40(43) | 42(41) | 34(36) | 34(27) | 34(27) | 2/16 |
| Alaska | 36(41) | 37(35) | 32(35) | 39(36) | 32(35) | 1/15 |
| Arizona | 49(44) | 70(59) | 59(60) | 42(35) | 42(35) | 6/16 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Arkansas | 49(52) | 54(69) | 53(70) | 40(38) | 37(26) | 12/16 |
| California | 41(49) | 67(53) | 48(50) | 34(29) | 34(29) | 6/16 |
| Colorado | 50(54) | 39(26) | 39(27) | 31(27) | 31(27) | 5/16 |
| Connecticut | 38(42) | 39(24) | 40(35) | 39(21) | 39(21) | 2/16 |
| Delaware | 39(36) | 34(28) | 39(35) | 30(23) | 30(23) | 5/16 |
| Washington DC | 48(44) | 32(23) | 35(33) | 26(20) | 26(20) | 5/16 |
| Florida | 33(26) | 34(20) | 29(20) | 19(14) | 19(14) | 3/16 |
| Georgia | 41(27) | 47(26) | 39(27) | 29(22) | 29(22) | 5/16 |
| Hawaii | 64(79) | 41(38) | 54(61) | 34(28) | 34(28) | 6/15 |
| Idaho | 38(39) | 40(31) | 35(35) | 34(26) | 33(25) | 4/16 |
| Illinois | 38(29) | 40(31) | 40(28) | 33(26) | 32(21) | 5/17 |
| Indiana | 40(33) | 44(38) | 42(34) | 35(33) | 32(23) | 11/17 |
| Iowa | 45(48) | 43(34) | 42(33) | 47(41) | 41(38) | 2/16 |
| Kansas | 47(47) | 51(46) | 45(43) | 31(20) | 31(20) | 5/16 |
| Kentucky | 38(39) | 38(25) | 31(23) | 25(18) | 25(18) | 5/16 |
| Louisiana | 36(41) | 48(58) | 46(58) | 38(28) | 38(28) | 4/16 |
| Maine | 50(39) | 43(41) | 46(39) | 33(27) | 33(27) | 5/16 |
| Maryland | 34(36) | 36(33) | 37(37) | 32(25) | 32(25) | 5/16 |
| Massachusetts | 38(34) | 40(28) | 38(30) | 33(23) | 33(23) | 2/16 |
| Michigan | 49(35) | 48(27) | 45(24) | 43(22) | 39(24) | 3/16 |
| Minnesota | 55(54) | 51(51) | 51(50) | 40(35) | 40(37) | 6/16 |
| Mississippi | 43(38) | 47(41) | 46(38) | 29(23) | 29(23) | 12/16 |
| Missouri | 36(29) | 39(39) | 39(39) | 23(47) | 19(43) | 12/16 |
| Montana | 51(46) | 42(32) | 40(34) | 40(31) | 34(21) | 7/16 |
| Nebraska | 42(33) | 44(33) | 43(33) | 37(26) | 37(26) | 4/16 |
| Nevada | 41(35) | 55(42) | 47(44) | 34(25) | 34(25) | 6/16 |
| New Hampshire | 43(38) | 42(24) | 38(22) | 34(21) | 34(21) | 3/16 |
| New Jersey | 27(24) | 31(20) | 25(17) | 34(16) | 25(17) | 1/16 |
| New Mexico | 46(47) | 52(29) | 42(32) | 33(32) | 33(32) | 8/16 |
| New York | 37(35) | 33(18) | 30(28) | 29(17) | 29(17) | 2/16 |
| North Carolina | 32(21) | 36(28) | 32(24) | 22(15) | 22(15) | 4/16 |
| North Dakota | 61(67) | 61(54) | 66(67) | 50(36) | 45(28) | 12/16 |
| Ohio | 43(42) | 41(31) | 38(31) | 28(19) | 28(19) | 5/16 |
| Oklahoma | 51(50) | 46(47) | 49(48) | 33(22) | 33(22) | 12/16 |
| Oregon | 47(49) | 39(23) | 35(26) | 28(21) | 28(21) | 2/16 |
| Pennsylvania | 46(40) | 37(23) | 38(23) | 27(17) | 27(17) | 5/17 |
| Rhode Island | 27(26) | 37(31) | 32(28) | 39(21) | 32(28) | 1/16 |
| South Carolina | 36(25) | 35(28) | 33(28) | 23(15) | 23(15) | 3/16 |
| South Dakota | 44(41) | 47(40) | 43(40) | 40(29) | 40(29) | 3/16 |
| Tennessee | 34(29) | 40(35) | 38(37) | 31(25) | 31(25) | 3/16 |
| Texas | 52(54) | 48(55) | 44(55) | 31(27) | 31(27) | 6/16 |
| Utah | 42(44) | 43(30) | 46(33) | 32(24) | 32(24) | 10/16 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Vermont | 44(29) | 37(21) | 41(28) | 39(24) | 38(24) | 3/16 |

Table B.4: Comparison of MAPE between different models across all the states in the US 4 weeks in advance. The number in parenthesis represents the standard deviation of the MAPE.

|  |  |  | 4 Weeks |  |  |  |
|---|---|---|---|---|---|---|
| State | tf-v-SIR | SLOW | SIMLR | LNQ-ens1 | Best | Rank |
| Alabama | 54(48) | 56(52) | 51(46) | 40(27) | 40(27) | 3/16 |
| Alaska | 58(66) | 50(36) | 47(38) | 49(32) | 46(36) | 2/15 |
| Arizona | 70(80) | 104(103) | 93(102) | 65(68) | 61(64) | 7/16 |
| Arkansas | 59(56) | 68(86) | 66(87) | 46(53) | 45(49) | 8/16 |
| California | 64(87) | 95(83) | 81(84) | 50(47) | 50(47) | 7/16 |
| Colorado | 73(90) | 52(26) | 52(33) | 41(36) | 39(24) | 6/16 |
| Connecticut | 60(65) | 46(34) | 58(49) | 44(23) | 44(23) | 6/16 |
| Delaware | 44(47) | 39(37) | 46(41) | 34(34) | 34(34) | 5/16 |
| Washington DC | 65(64) | 37(30) | 46(48) | 35(34) | 35(34) | 5/16 |
| Florida | 47(46) | 52(42) | 47(45) | 27(26) | 27(26) | 5/16 |
| Georgia | 48(40) | 64(35) | 59(33) | 38(32) | 38(32) | 7/16 |
| Hawaii | 102(153) | 55(43) | 77(98) | 45(43) | 45(43) | 6/15 |
| Idaho | 55(53) | 54(41) | 53(44) | 41(40) | 41(40) | 7/16 |
| Illinois | 53(38) | 51(43) | 54(40) | 43(37) | 39(27) | 5/17 |
| Indiana | 56(51) | 61(55) | 56(54) | 45(49) | 44(33) | 6/17 |
| Iowa | 61(72) | 55(46) | 53(46) | 55(56) | 50(45) | 2/16 |
| Kansas | 66(68) | 68(69) | 59(58) | 43(26) | 43(26) | 5/16 |
| Kentucky | 50(49) | 47(39) | 43(36) | 34(23) | 34(23) | 5/16 |
| Louisiana | 48(49) | 68(66) | 64(67) | 44(35) | 44(35) | 7/16 |
| Maine | 69(64) | 56(55) | 62(59) | 43(40) | 43(40) | 6/16 |
| Maryland | 51(71) | 45(45) | 53(61) | 42(43) | 42(43) | 6/16 |
| Massachusetts | 49(52) | 50(40) | 47(45) | 45(38) | 45(38) | 2/16 |
| Michigan | 62(67) | 56(36) | 51(37) | 53(32) | 51(44) | 2/16 |
| Minnesota | 74(87) | 65(61) | 64(62) | 55(51) | 47(41) | 5/16 |
| Mississippi | 52(49) | 62(52) | 59(51) | 38(43) | 38(43) | 6/16 |
| Missouri | 48(45) | 54(57) | 54(57) | 32(47) | 28(44) | 9/16 |
| Montana | 70(72) | 53(40) | 55(39) | 52(48) | 42(36) | 9/16 |
| Nebraska | 53(43) | 57(46) | 56(45) | 47(31) | 47(35) | 5/16 |
| Nevada | 67(54) | 77(61) | 71(65) | 41(43) | 41(43) | 8/16 |
| New Hampshire | 52(50) | 50(30) | 43(33) | 40(25) | 40(25) | 2/16 |
| New Jersey | 45(62) | 36(24) | 40(54) | 43(24) | 38(23) | 3/16 |
| New Mexico | 69(80) | 73(39) | 65(48) | 46(48) | 45(29) | 7/16 |
| New York | 48(43) | 41(22) | 39(31) | 37(25) | 33(22) | 4/16 |
| North Carolina | 41(33) | 48(41) | 45(36) | 29(22) | 29(22) | 6/16 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| North Dakota | 79(112) | 83(77) | 94(93) | 72(63) | 60(53) | 9/16 |
| Ohio | 60(58) | 54(44) | 52(44) | 35(31) | 35(31) | 6/16 |
| Oklahoma | 81(92) | 61(67) | 70(81) | 42(32) | 42(40) | 9/16 |
| Oregon | 63(65) | 49(33) | 43(35) | 39(27) | 39(27) | 2/16 |
| Pennsylvania | 63(54) | 47(29) | 47(30) | 35(27) | 35(27) | 5/17 |
| Rhode Island | 37(39) | 46(45) | 42(43) | 44(27) | 42(43) | 1/16 |
| South Carolina | 45(31) | 49(37) | 49(36) | 28(21) | 28(21) | 8/16 |
| South Dakota | 51(47) | 64(46) | 62(45) | 54(40) | 52(30) | 9/16 |
| Tennessee | 48(48) | 58(49) | 58(50) | 43(31) | 43(31) | 5/16 |
| Texas | 63(62) | 59(67) | 58(67) | 37(34) | 37(34) | 6/16 |
| Utah | 55(70) | 56(44) | 58(50) | 40(36) | 40(36) | 7/16 |
| Vermont | 56(67) | 41(26) | 49(55) | 45(27) | 41(26) | 4/16 |

# Appendix C

# Mathematical details

## C.1 Z-score normalization

Let $X_i^A$ and $X_i^B$ represent the values of the $i^{th}$ feature extracted from scanning sites $A$ and $B$ respectively. Then, we can represent the operations of scaling and translation as:

$$X_i^B = \alpha_i X_i^A + \beta_i, \qquad i = 1, 2, \ldots, m \tag{C.1}$$

where $\alpha_i$ and $\beta_i$ are the scaling and translation coefficients of the $i$th feature. In order to apply Z-score normalization we need to subtract the mean of every feature and divide by the standard deviation. Then the z-score normalized features from scanning sites A and B, $\bar{X_i^A}$, $\bar{X_i^B}$, are:

$$\begin{aligned}
\bar{X_i^A} &= \frac{X_i^A - E[X_i^A]}{\sqrt{Var(X_i^A)}} \\
\bar{X_i^B} &= \frac{X_i^B - E[X_i^B]}{\sqrt{Var(X_i^B)}} \\
&= \frac{\alpha_i X_i^A + \beta_i - E[\alpha_i X_i^A + \beta_i]}{\sqrt{Var(\alpha_i X_i^A + \beta_i)}} \\
&= \frac{\alpha_i \left(X_i^A - E[X_i^A]\right)}{\sqrt{\alpha_i^2 Var(X_i^A)}} \\
&= \frac{X_i^A - E[X_i^A]}{\sqrt{Var(X_i^A)}}, \quad \text{for } \alpha_i > 0 \\
&= \bar{X_i^A}
\end{aligned}$$

Therefore, after applying Z-score normalization, we are effectively removing the effects of translation and scaling.

## C.2 Whitening

To see why whitening removes the effects of rotation and scaling, consider the case where the datasets $X_B$ is a rotation and translation of $X_A$. This can be represented in matrix form as:

$$X^B \;=\; X^A\alpha + \mathbf{1}\beta^T \quad \alpha \in \mathbb{R}^{p\times p},\;\; \beta \in \mathbb{R}^p \tag{C.2}$$

where $\alpha$ is a rotation matrix – *i.e.*, is an orthogonal matrix with determinant $\det(\alpha) = 1$. The zero-mean datasets, $\bar{X}_A$, can be obtained as:

$$\begin{aligned}\bar{X}_A &= X_A - \mathbf{1}E[X_A] \\ E[X_A] &= [\ E[X_A^1],\ E[X_A^2],\ \ldots,\ E[X_A^p]\ ]\end{aligned} \tag{C.3}$$

while for the case of $\bar{X}_B$:

$$\begin{aligned}\bar{X}_B &= X_A\alpha + \mathbf{1}\beta^T - \mathbf{1}E[X_A\alpha + \mathbf{1}\beta^T] \\ &= X_A\alpha + \mathbf{1}\beta^T - \mathbf{1}\left(E[X_A\alpha] - E[\mathbf{1}\beta^T]\right) \\ &= (X_A - \mathbf{1}[X_A])\,\alpha \\ &= \bar{X}_A\alpha\end{aligned} \tag{C.4}$$

The eigenvalues of the covariance matrix $\Sigma_A = \frac{1}{n-1}\bar{X}_A^T\bar{X}_A$ are obtained by solving the equation $\det(\Sigma_A - \lambda I) = 0$. For the special case when $\alpha$ is a rotation matrix[1], $\alpha^T = \alpha^{-1}$, the eigenvalues of the covariance matrix of $\bar{X}_B$:

$$\begin{aligned}0 &= \det\left(\frac{1}{n-1}(\bar{X}_A\alpha)^T(\bar{X}_A\alpha) - \lambda I\right) \\ &= \det\left(\frac{1}{n-1}\alpha^T\bar{X}_A^T\bar{X}_A\alpha - \lambda I\right) \\ &= \det\left(\alpha^T\Sigma_A\alpha - \lambda I\right) \\ &= \det\left(\alpha^{-1}\Sigma_A\alpha - \alpha^{-1}\lambda I\alpha\right) \\ &= \det\left(\alpha^{-1}(\Sigma_A - \lambda I)\alpha\right) \\ &= \det(\alpha^{-1})\det(\Sigma_A - \lambda I)\det(\alpha) \\ &= \det(\Sigma_A - \lambda I)\end{aligned} \tag{C.5}$$

As for the eigenvectors: if $v$ is an eigenvector of $\Sigma_A$ with an associated eigenvalue $\lambda$, then $\Sigma_A v = \lambda v$. Doing some mathematical manipulations:

$$\begin{aligned}\alpha\Sigma_A v &= \alpha\lambda v \\ \alpha\Sigma_A I v &= \alpha\lambda v \\ \alpha\Sigma_A\alpha^{-1}\alpha v &= \lambda\alpha v \\ \Sigma_B(\alpha v) &= \lambda(\alpha v)\end{aligned} \tag{C.6}$$

Equations C.5 and C.6 show that, when the transformation matrix $\alpha$ is an orthogonal matrix with positive determinant, $X_A$ and $X_B$ will have the same eigenvalues, and the eigenvectors of $X_B$ are just a rotation of the eigenvectors of $X_A$. Therefore, by projecting the data into those eigenvector, we obtain the exact same representation, removing the effects of translation and rotation.

[1]All orthogonal matrices $\alpha$ have a determinant equal to +1, or -1. If it is positive, $\alpha$ is a rotation matrix. When the determinant is negative, it is a reflection matrix.

## C.3 Proof of Equation 4.5

$$\begin{aligned}
z^* &= \arg\min_z ||(x-\mu)-\theta z||^2 \\
&= \arg\min_z \left((x-\mu)-\theta z\right)^T \left((x-\mu)-\theta z\right) \\
&= \arg\min_z ((x-\mu)^T - z^T\theta^T)\left((x-\mu)-\theta z\right) \\
&= \arg\min_z \left(z^T\theta^T\theta z - 2(x-\mu)^T\theta z\right)
\end{aligned}$$

Taking the derivative with respect to $z$ and making it equal to the zero vector:

$$\begin{aligned}
0 &= \frac{\partial}{\partial z}\left(z^T\theta^T\theta z - 2(x-\mu)^T\theta z\right) \\
&= 2\theta^T\theta z - 2\theta^T(x-\mu) \\
\rightarrow z &= (\theta^T\theta)^{-1}\theta^T(x-\mu)
\end{aligned}$$

Note that the second derivative is always non-negative, so $z$ is a minimum.

## C.4 Proof of Equation 4.9

Substituting $\theta = US^{\frac{1}{2}}Q$ in $z = (\theta^T\theta)^{-1}\theta^T(x-\mu)$, and using that $Q^T = Q^{-1}$ for orthogonal matrices, and $(AB)^{-1} = B^{-1}A^{-1}$ for invertible matrices $A$ and $B$.:

$$\begin{aligned}
z &= (\theta^T\theta)^{-1}\theta^T(x-\mu) \\
&= \left((US^{\frac{1}{2}}Q)^T(US^{\frac{1}{2}}Q)\right)^{-1}(US^{\frac{1}{2}}Q)^T(x-\mu) \\
&= \left(Q^TS^{\frac{1}{2}}U^TUS^{\frac{1}{2}}Q\right)^{-1}(US^{\frac{1}{2}}Q)^T(x-\mu) \\
&= \left(Q^TS^{\frac{1}{2}}S^{\frac{1}{2}}Q\right)^{-1}(US^{\frac{1}{2}}Q)^T(x-\mu) \\
&= \left(S^{\frac{1}{2}}Q\right)^{-1}\left(Q^TS^{\frac{1}{2}}\right)^{-1}(US^{\frac{1}{2}}Q)^T(x-\mu) \\
&= Q^TS^{-\frac{1}{2}}S^{-\frac{1}{2}}QQ^TS^{\frac{1}{2}}U^T(x-\mu) \\
&= Q^TS^{-1}S^{\frac{1}{2}}U^T(x-\mu) \\
&= Q^TS^{-\frac{1}{2}}U^T(x-\mu)
\end{aligned}$$

## C.5 Proof of Equation 4.14

For the case of the Equation 4.11 using the Gaussian kernel, and since $R^TR = I$, the gradient of the MMD between $X$ and a linear transformation of $Y$, $RY$, with respect to $R$ is:

$$\frac{\partial}{\partial R}\mathrm{MMD}_b^2[\mathcal{F}, X, RY]$$
$$= \frac{1}{m^2}\sum_{i,j=1}^{m}\frac{\partial}{\partial R}k(x_i, x_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n}\frac{\partial}{\partial R}k(x_i, Ry_j) +$$
$$\frac{1}{n^2}\sum_{i,j=1}^{n}\frac{\partial}{\partial R}k(Ry_i, Ry_j)$$

$$= \frac{1}{m^2}\sum_{i,j=1}^{m}\frac{\partial}{\partial R}\exp\left(-\frac{1}{2\sigma^2}||x_i - x_j||^2\right)$$
$$- \frac{2}{mn}\sum_{i,j=1}^{m,n}\frac{\partial}{\partial R}\exp\left(-\frac{1}{2\sigma^2}||x_i - Ry_j||^2\right)$$
$$+ \frac{1}{n^2}\sum_{i,j=1}^{n}\frac{\partial}{\partial R}\exp\left(-\frac{1}{2\sigma^2}||Ry_i - Ry_j||^2\right)$$

$$= -\frac{2}{mn}\sum_{i,j=1}^{m,n}\frac{\partial}{\partial R}\exp\left(-\frac{1}{2\sigma^2}(x_i - Ry_j)^T(x_i - Ry_j)\right)$$
$$+ \frac{1}{n^2}\sum_{i,j=1}^{n}\frac{\partial}{\partial R}\exp\left(-\frac{1}{2\sigma^2}(Ry_i - Ry_j)^T(Ry_i - Ry_j)\right)$$

$$= -\frac{2}{mn}\sum_{i,j=1}^{m,n}\frac{\partial}{\partial R}\exp\left(-\frac{x_i^T x_i - 2x_i^T Ry_j + y_j^T y_j}{2\sigma^2}\right)$$
$$+ \frac{1}{n^2}\sum_{i,j=1}^{n}\frac{\partial}{\partial R}\exp\left(-\frac{y_i^T y_i - 2y_j^T y_i + y_j^T y_j}{2\sigma^2}\right)$$

$$= -\frac{2}{mn}\sum_{i,j=1}^{m,n}\exp\left(-\frac{||x_i - Ry_j||^2}{2\sigma^2}\right)\left(\frac{x_i y_j^T}{\sigma^2}\right)$$