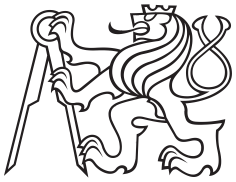
Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Adaptation of CNN Classifiers to Prior Shift

Tomáš Šipka

Supervisor: Ing. et Ing. Milan Šulc, Ph.D.
Field of study: Open informatics
Subfield: Computer Vision and Image Processing
August 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Šipka Tomáš** Personal ID number: **457428**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Computer Vision and Image Processing**

II. Master's thesis details

Master's thesis title in English:

Adaptation of CNN Classifiers to Prior Shift

Master's thesis title in Czech:

Adaptace CNN klasifikátorů na změny apriorních pravděpodobností

Guidelines:

1. Familiarize yourself with Convolutional Neural Network (CNN) classifiers and the standard training procedure of empirical risk minimization with cross entropy loss [1].
2. Familiarize yourself with common domain adaptation scenarios, with focus on prior shift (also known as label shift or target shift).
3. Review the state-of-the-art methods for prior shift adaptation of CNN and other probabilistic classifiers.
4. Review the state-of-the-art methods for calibration of CNN (and other probabilistic) classifiers.
5. Propose a new method (or methods) for classifier adaptation to a change in categorical priors (prior shift, label shift) between training and test data. Explore the problem of inconsistent estimates in the confusion matrix-based methods [7,9] and the possibility of regularization [3].
6. Experimentally evaluate the existing and proposed methods on different prior-shift scenarios and datasets, with and without classifier calibration.

Bibliography / sources:

[1] Bengio, Yoshua, Ian Goodfellow, and Aaron Courville. Deep learning. Vol. 1. Massachusetts, USA: MIT press, 2017.
[2] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. ArXiv, 1901.06852v5, 2019.1
[3] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. arXiv preprint arXiv:1903.09734,2019. 1, 2
[4] Marthinus Christoffel du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. CoRR, abs/1206.4677, 2012.1, 2, 4
[5] Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors, 2018. 1, 2, 4
[6] Geoffrey J. McLachlan. Discriminant Analysis and Statistical Pattern Recognition. John Wiley & Sons, Inc, Hoboken, NJ,USA, 1992-03-27. 2
[7] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Comput., 14(1):21–41, Jan.2002. 1, 2, 4
[8] Milan Sulc and Jiri Matas. Improving CNNclassifiers by estimating test-time priors. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct 2019. 1, 2, 4
[9] Slobodan Vucetic and Zoran Obradovic. Classification on data with biased class distribution. In European Conference on Machine Learning, pages 527–538. Springer, 2001. 1

Name and workplace of master's thesis supervisor:

Ing. Milan Šulc, Ph.D., Visual Recognition Group, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **28.01.2021** Deadline for master's thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Milan Šulc, Ph.D. Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor Ing. et Ing. Milan Šulc, Ph.D. for his valuable advice, relentless support and guidance not only during the time I was working on this thesis but also during my time at CMP, where the foundation stones of this work were laid. I also wish to thank prof. Ing. Jiří Matas, Ph.D. for the opportunity to be part of CMP and for his helpful advice and constructive criticism. The discussions with both of them were very helpful, inspiring and especially great lessons. Thanks should also go to my family for their support during my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 13. August 2021

Abstract

In many classification tasks, the test set's relative class frequencies (class priors probabilities) differ from the relative class frequencies at training time. Such a phenomenon, called *label shift* or *prior shift*, can negatively affect the classifier's performance. Considering a probabilistic classifier approximating posterior probabilities, the predictions can be adapted to the label shift by re-weighting with a ratio of the test set and training set priors. Labels in the test set are usually unknown, therefore the prior ratio has to be estimated in an unsupervised manner. This thesis reviews existing methods for adapting probabilistic classifiers to label shift and for estimating test priors in an unlabeled test set. Moreover, we propose novel algorithms to address the problems of estimating new priors and prior ratio. The methods are designed to handle a known issue in confusion matrix-based methods, where inconsistent estimates of decision probabilities and confusion matrices lead to negative values in estimated priors. Experimental evaluation shows that our method improves the stability of prior estimation and the adapted classifier's accuracy compared to the baseline confusion matrix-based methods and achieves state-of-the-art performance in prior shift adaptation.

Keywords: Machine Learning, Computer Vision, Domain Adaptation, Transfer Learning, Classification, Label Shift, Prior Shift, Classifier Calibration, Convolutional Neural Networks, Confusion Matrix

Supervisor: Ing. et Ing. Milan Šulc, Ph.D.
Visual Recognition Group,
Department of Cybernetics,
FEE CTU in Prague

Abstrakt

V mnoha klasifikačních úlohách se relativní četnosti tříd (apriorní pravděpodobnosti tříd) na testovací sadě liší od relativních četností během trénování klasifikátoru. Tento jev, taktéž nazýván *label shift* nebo *prior shift*, může negativně ovlivnit přesnost klasifikace. Budeme-li uvažovat pravděpodobnostní klasifikátor aproximující aposteriorní pravděpodobnosti, mohou být jeho predikce adaptovány na label shift převážením poměrem testovacích a trénovacích apriorních pravděpodobností. Jelikož jsou anotace v testovací sadě obvykle neznámé, musí být poměr apriorních pravděpodobností odhadnut metodou učení bez učitele. Tato práce rekapituluje existující řešení pro adaptaci pravděpodobnostních klasifikátorů na label shift a odhad apriorních pravděpodobností na testovací sadě bez anotací. Navíc, navrhuje nové algoritmy pro odhad apriorních pravděpodobností na testovací sadě a pro odhad poměru tesotovacích a trenovacích apriorních pravděpodobností. Tyto metody jsou navrženy tak, aby řešily známý problém v metodách založených na matici záměn, kde nekonzistentní odhad pravděpodobnosti rozhodnutí klasifikátoru a jeho chybové matice může vést k záporným hodnotám v odhadnutých četnostech. Experimentální vyhodnocení ukazuje, že naše metoda zlepšuje stabilitu odhadu apriorních pravděpodobností a přesnost adaptovaného klasifikátoru v porovnání s jinými metodami založenými na matici záměn a současně dosahuje nejlepších výsledků mezi metodami pro adaptaci klasifikátorů na prior shift.

Klíčová slova: Strojové Učení, Počítačové Vidění, Domain Adaptation, Transfer Learning, Klasifikace, Label Shift, Prior Shift, Kalibrace Klasifikátorů, Konvoluční Neuronové Sítě, Matice Záměn

Překlad názvu: Adaptace CNN klasifikátorů na změny apriorních pravděpodobností

Contents

Chapter 1

Introduction

"It's going to be interesting to see how society deals with artificial intelligence, but it will definitely be cool."

— *Colin Angle*

Due to its wide area of applications and great potential to reduce costs, human resources and managing risks, increase growth and productivity as well as the potential to overcome human limitations, *artificial intelligence* (AI) is a recently rapidly growing area of active research. There are two main approaches to define artificial intelligence [1]. The first one defines AI as the ability of machines or computers to act like human beings. From this point of view, the goal of AI is to design a machine simulating human-like behaviour. Based on this human-centred approach, in 1950 Alan Turing proposed the so-called Turing Test, designed to determine if machine intelligence is close to that of a human. The test was composed of several questions asked by the examiner. A computer passed the test if the examiner could not distinguish between the computer's and human's answers. The second approach to AI avoids humans in its definition and instead of comparing computers to humans, it compares their level of rationality. According to this rationalist approach, AI is the ability of an agent to learn from experience and solve problems to achieve its goals.

An important sub-field of artificial intelligence is *machine learning* [2, 3]. Machine learning studies algorithms, which are able to learn and adapt themselves from experience. The term "experience" is usually represented by data in electronic form [2], which were either collected in the past (offline learning) or directly at the training time (online learning). These data sets,

also called "training data", are being provided to the algorithm. An example of such data are digital images, time series of stock prices, disease symptoms etc. The data serves as an input into the algorithm and based on these inputs, we aim to receive an output corresponding to a solution or prediction for some task. E.g. for image, the desired output could be a description of the captured scene, from stock prices in the past we may want to predict prices in the future, from symptoms we could predict disease etc. If the training data also contains the desired outputs, then the learning procedure is called *supervised*. Otherwise, we talk about *unsupervised* learning. A recently popular area of machine learning is deep learning [4]. Deep learning is specific by making use of artificial neurons, inspired by real neurons in the biological brain, and combining them into more complex structures, called artificial neural networks (ANNs). ANNs possess a great ability to extract complex information from the training data and learn from them. Different designs of neural networks were proposed for different data structures e.g. Convolutional Neural Network (CNN) [5] for image processing, Recurrent Neural Networks (RNN) [6] for time series data or Graph Neural Networks (GNN) [7] for graph-structured data. Machine learning and deep learning can find use in recommendation systems, medical diagnosis, speech recognition, computer vision and many other areas.

One of the fundamental tasks in machine learning is classification. Classification aims to assign a label (category, class) to a given observation. Usually, the number of categories is small, in a range from tens to thousands, but can also be unbounded [2]. The observation can be anything. For example, an image of a vehicle can be labelled as a car, truck or bus; animals can be assigned label corresponding to their species name and text can be labelled with a sentiment (positive, negative, or neutral). The algorithm assigning categories to observations is called a classifier. The input of the classifier is a feature vector, i.e. a vector of values (features) representing a particular observation. For instance, an animal can be represented by its colour, weight, height and width, yielding a 4-dimensional feature vector, but can also be represented by its digital grey-scale image with resolution 1000×1000 pixels, yielding a 1 million-dimensional feature vector. The set of all categories, that can be assigned to the observations is referred to as label space, denoted as $\mathbb{Y}$. D-dimensional space of feature vectors is called feature space and is denoted as $\mathbb{X}$. It is common to train classifiers to output probabilities over label space, representing the classifier's confidence that a given datum belongs to a particular class. In other words, model's output $f(X)$ is learned to approximate posterior probabilities $p(Y|X)$, i.e. $f(X) \approx p(Y|X)$, where X

and Y are random variables over feature space and label space respectively. An example of such a classifier is a Convolutional Neural Network (CNN) trained with cross-entropy loss.

When deploying a classifier into a real-world application, it may happen that the real-world data, also referred to as "test data", are different from those during training. In the most general case, even the label space and feature space can differ. For example, if we do not have enough data to learn classifying dog's breeds, we can first train our classifier on an inter-species data set with a sufficient amount of samples and then reuse gained knowledge to classify dogs. The field dealing with knowledge transfer from one task (source domain) to another (target domain) is called *transfer learning* [8]. However, even though we have enough data for our specific task defined over some feature space and label space, there still can be a difference between training and test data. For instance, when the inputs into the classifier are images. they can be taken under different lighting conditions or disease symptoms can be slightly changed over time. In other words, the probability density generating training data is not equal to the density generating test data, i.e. $p_{\mathcal{T}}(Y, X) \neq p_{\mathcal{E}}(Y, X)$. The sub-field of transfer learning associated with adapting algorithms to a new data distribution, while feature space and label space are fixed, is called *domain adaptation* [9].

It is reasonable to simplify a general difference between the train density $p_{\mathcal{T}}(Y, X)$ and the test density $p_{\mathcal{E}}(Y, X)$ by decomposition $p(Y, X) = p(Y|X)p(X) = p(X|Y)p(Y)$ and then make assumptions about its particular terms. Transferring knowledge without any assumptions about source and target domains would not be possible. For example, one could assume that density over feature space can change, i.e. $p_{\mathcal{T}}(X) \neq p_{\mathcal{E}}(X)$, but posterior probability remains the same, i.e. $p_{\mathcal{T}}(Y|X) = p_{\mathcal{E}}(Y|X)$. This scenario is called *covariate shift* [9, 10, 11, 12, 13]. As example of covariate shift consider a diagnosis of breast cancer. Suppose that the data for the training set were collected from mostly elderly women. Indeed such data set is not a representative sample of the population with respect to the age, resulting in a bias within the marginal distribution $p(X)$. Another data set shift is *conditional shift* [14] were the relative class frequencies are the same, but conditional densities $p(X|Y)$ may differ. Last but not least is a situation where the class appearance over data sets does not change, i.e. $p_{\mathcal{T}}(X|Y) = p_{\mathcal{E}}(X|Y)$, and the only difference is in relative class frequencies (class priors), i.e. $p_{\mathcal{T}}(Y) \neq p_{\mathcal{E}}(Y)$. This phenomenon is called *prior shift* or *label shift* [9, 15, 16, 17, 18, 19, 20]. To have a better notion about the label shift, consider an image-based recognition system for some animal species

in the wild to be applied in a real-world application. It is obvious that the distribution of species will vary from place to place in the world as well as from season to season during the year. However, we have to assume that the appearance of species we want to recognize does not change with place and season. Indeed, this is one of the drawbacks of this assumption, because in many cases this is not true (e.g. trees may look different in summer and winter). As another example consider a diagnosis of a disease. The number of infected people will depend on the season during the year or even on the location around the world. Yet another example, consider sentiment analysis, where emotions are assigned to text messages. The priors over emotions will differ depending on the environment where the classifier is deployed. For instance, emotions in messages exchanged between new partners in love versus messages between partners after a long marriage. In all those situations the class priors at test time have changed. To maintain close to the optimal performance we should adapt the classifier's predictions $f(X) \approx p_{\mathcal{T}}(Y|X)$ to reflect the change in the class priors.

This thesis aims to evaluate and extend the methodology for adapting classifier to prior shift and compare it to the existing methods. The algorithm proposed in this thesis is estimating test set priors, allowing to adapt classifier's predictions to label shift by re-weighting them in post-processing step. The test set prior estimation is based on classifier's confusion matrix and constrained likelihood maximization of probability of classifier's decision. By constraining the optimization, we handle the known problem [21, 16, 18, 20] of negative values in the test set prior estimated by confusion matrix. The evaluation is primarily performed in the domain of computer vision with CNN classifiers.

The thesis has the following structure: Chapter 2 gives a brief description of classification task and general approaches to domain adaptation. The main focus of Chapter 2 is on unsupervised methods for classifier adaptation to label shift and on classifier calibration. Chapter 3 proposes new algorithms based on confusion matrix to approach the label shift adaptation. Chapter 4 contains an evaluation of algorithms proposed in Chapter 3 and their comparison to the existing methods on standard long-tailed data sets. The final conclusion can be found in Chapter 5.

Chapter 2

Related Work

"So many books, so little time."

— *Frank Zappa*

2.1 Classification

Let $\mathbb{X}$ be a feature space and $\mathbb{Y}$ a finite set of K classes. The goal of classification task is to learn mapping (classifier) $h : \mathbb{X} \to \mathbb{Y}$ such that for every sample $\mathbf{x} \in \mathbb{X}$ with a ground truth label $y \in \mathbb{Y}$ the assignment $h(\mathbf{x})$ is equal to y. It is common practice to mark the labels (in general label can be anything e.g. dog, cat,...) with integer values $1, ..., K$ and encode them as K-dimensional one-hot vectors. Meaning that for some label $y \in \{1, ..., K\}$, the vector will have 1 at y-th position and 0 at all other positions. For example, set of four classes $\{1, 2, 3, 4\}$ can be encoded as $\{(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)\}$. Such encoding allows us to train the classifier to output probability distribution over label space $\mathbb{Y}$, representing classifier's confidence that sample $\mathbf{x}$ belongs to the particular class. From this point of view, the (probabilistic) classifier f is a mapping from feature space $\mathbb{X}$ to the probability simplex Δ_{K-1}, i.e. $f : \mathbb{X} \to \Delta_{K-1}$. Based on this probabilistic output, the input sample $\mathbf{x}$ is assigned label $d \in \mathbb{Y}$, by taking class with the highest probability

$$d = h(\mathbf{x}) \coloneqq \underset{k \in \{1,...,K\}}{\arg\max}\ f(\mathbf{x}) = \underset{k \in \{1,...,K\}}{\arg\max}\ (f^{(1)}, f^{(2)}, \ldots, f^{(K)})(\mathbf{x}), \tag{2.1}$$

where $\sum_{i=1}^{K} f^{(i)} = 1$ and $f^{(i)} \geq 0$. It is often assumed that the classifier f approximates posterior probability

$$f(\mathbf{x}) \approx p(Y|X). \tag{2.2}$$

2.1.1 Training Classifier

To train a classifier, we need data from which our model will be able to extract some useful information to solve the classification task. In supervised learning the training data $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^{M}$ are a set of features $\mathbf{x}_i$ and corresponding labels y_i (see Figure 2.1 for example). Once the model is trained, it is useful to know how well the classifier performs. For this purpose, we need to define some performance measure. The simplest choice is 0-1 loss, which is 1 for $h(\mathbf{x}) \neq y$ and 0 for $h(\mathbf{x}) = y$. The expected value of 0-1 loss is called error rate. The opposite value to error rate is accuracy. The relation between accuracy (acc) and error rate (err) is as follow

$$acc = 1 - err. \tag{2.3}$$

Usually, we do not have access to all examples from feature space and therefore the error rate and accuracy have to be estimated from a finite number of samples. One could estimate the performance on the training set, but due to over-fitting, the estimated values do not reflect how well the classifier generalizes to new samples. Consequently, it is a common practice to estimate the values on a set of samples $\mathcal{V} = \{\mathbf{x}_i, y_i\}_{i=1}^{M'}$, called validation set, unseen during the training stage. The accuracy and error rate can be estimated as

$$\begin{aligned} acc &= \frac{1}{M'} \sum_{(x_i,y_i)\in\mathcal{V}} [\![h(\mathbf{x}_i) = y_i]\!], \\ err &= \frac{1}{M'} \sum_{(x_i,y_i)\in\mathcal{V}} [\![h(\mathbf{x}_i) \neq y_i]\!]. \end{aligned} \tag{2.4}$$

Minimizing the error rate directly is typically intractable, because it is a discontinuous and non-convex function, but can be minimized indirectly using some smooth, more tractable function under the Empirical Risk Minimization (ERM) [2] framework.

In general we would like to minimize true error

$$R(h) = \mathbb{E}_{(X,Y)\sim P}[l(y, h(x))], \tag{2.5}$$

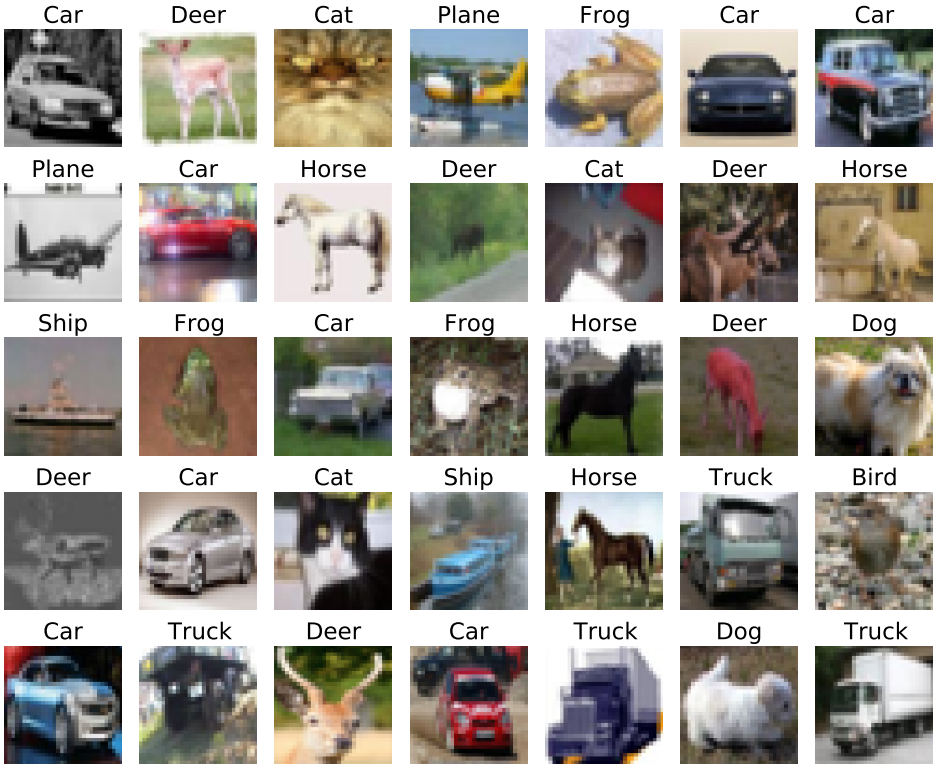


Figure 2.1: An example of training data from CIFAR10 dataset. The dataset contains RGB images of size 32×32 pixels. Each image is assigned one of ten classes.

where $\mathbb{E}$ denotes expected value and $l : \mathbb{Y} \times \mathbb{Y} \to R$ is a loss function. Since we do not have an access to $P(X, Y)$, the problem is approached within the ERM framework by minimizing the empirical risk on a training set

$$R(h) = \frac{1}{M} \sum_{(x_i, y_i) \in \mathcal{T}} l(y_i, h(\mathbf{x}_i)). \tag{2.6}$$

For a probabilistic classifier f, one of the most common choices for l is the cross-entropy loss

$$l(y_i, f(\mathbf{x}_i)) = -\sum_{j=1}^{K} y_i^{(j)} \log f^{(j)}, \tag{2.7}$$

where $y_i^{(j)}$ denotes the j-th position of one-hot encoded ground truth label

and $f^{(j)}$ is the j-th output of probabilistic classifier. It can be observed that unlike error rate in Equation (2.4), the cross-entropy loss is smooth and convex and we can easily use this function to train our classifier with gradient descent.

2.2 Convolutional Neural Networks

Recently, Convolutional Neural Networks [4] are very popular and widely used in many computer vision tasks like classification, image segmentation, image completion etc.

The main building unit of CNN is the convolution operator. Depending on the form of data, the convolution can be defined over any number of dimensions. In the case of image data, CNNs are based on two-dimensional convolution

$$(I * F)(i, j) = \sum_m \sum_n I(m, n)F(i - m, j - n), \tag{2.8}$$

where I is an input image, F is a filter applied to the image and values $F(k, l)$ are learnable parameters. Convolution is very similar to cross-correlation and in some literature, the operator in CNNs is called cross-correlation instead of convolution. The difference between them in a two-dimensional case is that convolution rotates the filter by 180 degrees. Since in deep learning the parameters in the filter are learnable, both operations will lead to the same result. In fact, in many frameworks cross-correlation is used for CNN implementation.

Neural networks are composed from several layers. A single CNN layer is made out of many filters, followed by a non-linear function. Their arrangement in the layer affects, together with the layer stacking, the network architecture. One of the first architectures was LeNet [5] (schema of LeNet-5 architecture is shown on image (2.2)), designed to recognize handwritten digits. Since then, many new architectures have been proposed: VGG [22], GoogLeNet [23], ResNet [24], ResNeXt [25], ResNeSt [26] and Efficientnet [27] to name a few.

2.3 Unsupervised Domain Adaptation

At test time it is often assumed that the training set is a good reflection of test distribution $p_{\mathcal{E}}(X, Y)$. When this assumption is not met, the classifier

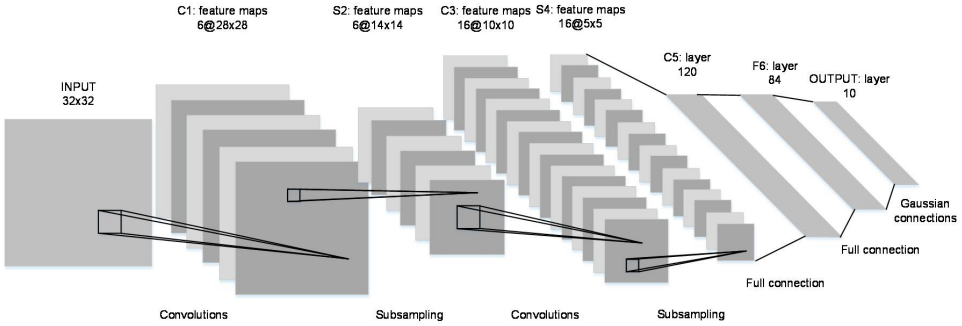


Figure 2.2: Schema of a LeNet-5 CNN architecture. Six 5×5 convolutional filters are applied on input, 32×32 grayscale image, resulting in $6 \times 28 \times 28$ feature map. Dimension of this feature map is reduced by subsampling layer and then another six convolutional filters are applied, again followed by subsampling. At the end, there are 3 fully connected layers. Image taken from [28].

fails to generalize to samples drawn from $p_{\mathcal{E}}(X, Y)$ and the performance is deteriorated. The aim of the domain adaptation is to train a model on the training distribution $p_{\mathcal{T}}(X, Y)$ capable to generalize well to a new samples from distribution $p_{\mathcal{E}}(X, Y)$, when $p_{\mathcal{T}}(X, Y) \neq p_{\mathcal{E}}(X, Y)$. Domain adaptation is unsupervised when we have access to labeled source data $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^{M}$, but unlabeled target data $\mathcal{E} = \{\mathbf{x}_i\}_{i=1}^{N}$.

One of the approaches to domain adaptation is the *importance re-weighting* [14, 12, 13, 11, 29], commonly used in the label and covariate shift scenarios. This technique is based on minimization of an expected loss on the target domain by re-weighting the loss function on samples from the source domain:

$$\begin{aligned} R_{\mathcal{E}}(h) &= \mathbb{E}_{(X,Y)\sim p_{\mathcal{E}}(X,Y)}[l(y, h(x))] = \int p_{\mathcal{T}}(X, Y) \frac{p_{\mathcal{E}}(X, Y)}{p_{\mathcal{T}}(X, Y)} l(y, h(x)) dx dy = \\ &= \mathbb{E}_{(X,Y)\sim p_{\mathcal{T}}(X,Y)} \left[\frac{p_{\mathcal{E}}(X, Y)}{p_{\mathcal{T}}(X, Y)} l(y, h(x)) \right]. \end{aligned} \tag{2.9}$$

By factorizing the joint distribution $p(X, Y)$ to $p(X)p(Y|X)$ or $p(Y)p(X|Y)$, we get following ratios $p_{\mathcal{E}}(X)/p_{\mathcal{T}}(X)$, $p_{\mathcal{E}}(Y|X)/p_{\mathcal{T}}(Y|X)$ and $p_{\mathcal{E}}(Y)/p_{\mathcal{T}}(Y)$, $p_{\mathcal{E}}(X|Y)/p_{\mathcal{T}}(X|Y)$, respectively, in the Equation (2.9). Estimating one of the two ratio pairs allows us to re-weight a loss function at the training stage and therefore to adapt the classifier to the target domain. Note, that estimating ratios containing conditional densities $p(X|Y)$ and $p(Y|X)$ is very difficult, as they are hard to model, unless we have some prior information about how the densities change. Consequently, the importance re-weighting is common

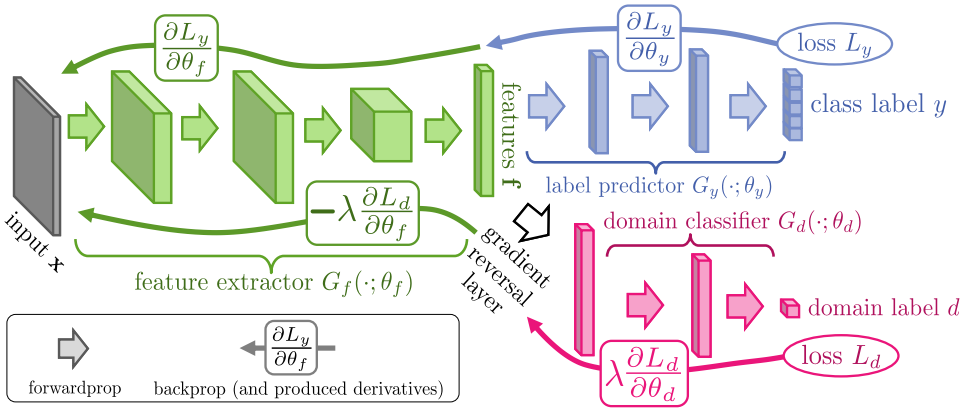


Figure 2.3: Schema of an adversarial neural network for domain adaptation. Feature extractor (green) and label predictor (blue) together form a standard neural network for classification. Domain-invariant mapping is learnt in an adversarial manner with domain classifier (red) by applying reversed gradient in feature extractor with respect to the domain loss L_d. The image is taken from [35].

to use with the assumption that one of the conditional densities does not change.

Another technique is based on the assumption that some learn-able transformation [9, 30, 31, 32] exists between the domains. Unfortunately, this method tends to overfit when working with high-dimensional data and complicated domain transformation. This is common for example in computer vision, where feature space can have thousands or even millions of dimensions.

Instead of transforming one domain into another, an alternative approach attempts to find domain-invariant representation [9, 33, 34]. This technique learns a transformation from the feature space $\mathbb{X}$ into a possibly lower-dimensional feature space $\mathbb{Z}$, which is invariant to the source and target domain. Attempting to reach this goal, promising results were recently achieved by methods [35, 36, 37, 38, 39] leveraging adversarial learning (AL) [40]. Domain adaptation based on AL is using a feature extractor that maps original features to domain-invariant space. Two classifiers are learned on top of this inter-mediate feature space. The first classifier is predicting class labels, while the second one predicts domain in an adversarial manner (see Figure 2.3).

2.4 Classifier Adaptation to Label Shift

Recall that label shift makes the following assumptions about the data generating distributions:

1. $p_{\mathcal{T}}(Y) \neq p_{\mathcal{E}}(Y)$
2. $p_{\mathcal{T}}(X|Y) = p_{\mathcal{E}}(X|Y)$

Let us assume probabilistic classifier $f_{\mathcal{T}}(\mathbf{x}) \approx p_{\mathcal{T}}(Y|\mathbf{x})$ trained on the training set $\mathcal{T}$ with relative class frequencies $p_{\mathcal{T}}(Y)$. The formula for adapting the classifier's predictions to the new priors $p_{\mathcal{E}}(Y)$ in a test set $\mathcal{E} = \{\mathbf{x}_i\}_{i=1}^{N}$ can be simply derived by applying Bayes theorem to the assumption about equality of class-conditional distributions

$$p_{\mathcal{T}}(\mathbf{x}|Y) = p_{\mathcal{E}}(\mathbf{x}|Y) = \frac{p_{\mathcal{T}}(Y|\mathbf{x})p_{\mathcal{T}}(\mathbf{x})}{p_{\mathcal{T}}(Y)} = \frac{p_{\mathcal{E}}(Y|\mathbf{x})p_{\mathcal{E}}(\mathbf{x})}{p_{\mathcal{E}}(Y)}. \tag{2.10}$$

By expressing adapted posteriors, we get

$$p_{\mathcal{E}}(Y|\mathbf{x}) = p_{\mathcal{T}}(Y|\mathbf{x})\frac{p_{\mathcal{E}}(Y)p_{\mathcal{T}}(\mathbf{x})}{p_{\mathcal{T}}(Y)p_{\mathcal{E}}(\mathbf{x})}. \tag{2.11}$$

Note that the sum over posteriors is equal to one

$$\sum_{i=1}^{K} p_{\mathcal{E}}(Y|\mathbf{x}) = \sum_{i=1}^{K} p_{\mathcal{T}}(Y|\mathbf{x})\frac{p_{\mathcal{E}}(Y)p_{\mathcal{T}}(\mathbf{x})}{p_{\mathcal{T}}(Y)p_{\mathcal{E}}(\mathbf{x})} = 1 \tag{2.12}$$

and the ratio $p_{\mathcal{T}}(\mathbf{x})/p_{\mathcal{E}}(\mathbf{x})$ can be factored out before the sum, thus

$$\frac{p_{\mathcal{T}}(\mathbf{x})}{p_{\mathcal{E}}(\mathbf{x})} = \left[\sum_{i=1}^{k} p_{\mathcal{T}}(Y|\mathbf{x})\frac{p_{\mathcal{E}}(Y)}{p_{\mathcal{T}}(Y)}\right]^{-1} \tag{2.13}$$

is normalization constant and Equation (2.11) can be expressed as

$$\boxed{p_{\mathcal{E}}(Y|\mathbf{x}) \propto p_{\mathcal{T}}(Y|\mathbf{x})\frac{p_{\mathcal{E}}(Y)}{p_{\mathcal{T}}(Y)}}, \tag{2.14}$$

which gives us the recipe for adaption of classifier's predictions under the label shift.

The remaining question is how to estimate test and training priors, which are used to re-weight the predictions. If we have access to the training set

$\mathcal{T}$, we can estimate training priors $p_{\mathcal{T}}(Y)$ by simply counting ground-truth labels in the data set (although other options are possible, see experiments in the Section 4.2.1). Estimating test set priors $p_{\mathcal{E}}(Y)$ is typically more difficult because usually, we do not have access to the ground truth labels. This thesis is focused on answering the question "how to estimate test priors". An alternative approach is to estimate the priors ratio $w(Y) = p_{\mathcal{E}}(Y)/p_{\mathcal{T}}(Y)$ directly. Note that once we have estimated priors $p_{\mathcal{T}}(Y)$ and $p_{\mathcal{E}}(Y)$, instead of re-weighting the predictions on test set according to the Equation (2.14), we can use the ratio to train a new classifier, adapted to the target domain, by minimizing the expected loss in the Equation (2.9). This can be done by directly re-weighting the loss function during training or by sampling training examples according to the estimated importance weights.

2.4.1 Confusion Matrix

One of the first procedures [41, 16] for prior estimation is based on a $K \times K$ left stochastic matrix $\mathbf{C}_{d|y}$ called confusion matrix (CM). The value in the k-th column and i-th row is interpreted as probability $p(D = i|Y = k)$ of the classifier h deciding for class i when the true class is k. Both D and Y are random variables over the label space, but D represents the classifier's decision, while Y represents the ground-truth label.

Useful property of confusion matrix is that it does not change with prior shift [20]:

Lemma 2.1. *Let Y be a random variable representing the ground truth label of a sample $\mathbf{x}$ and D a random variable representing classifier's decision $h(\mathbf{x})$. If assumption $p_{\mathcal{T}}(\mathbf{x}|Y) = p_{\mathcal{E}}(\mathbf{x}|Y)$ holds, then*

$$p(D|Y) \coloneqq p_{\mathcal{T}}(D|Y) = p_{\mathcal{E}}(D|Y),$$

$$\mathbf{C}_{d|y} \coloneqq \mathbf{C}^{\mathcal{T}}_{d|y} = \mathbf{C}^{\mathcal{E}}_{d|y}.$$

Proof. From the law of total probability

$$p_{\mathcal{T}}(D|Y) = \sum_{\mathbf{x}\in\mathbb{X}} p_{\mathcal{T}}(D, \mathbf{x}|Y) = \sum_{\mathbf{x}\in\mathbb{X}} p_{\mathcal{T}}(D|\mathbf{x}, Y)p_{\mathcal{T}}(\mathbf{x}|Y).$$

Note that the classifier's decision depends only on $\mathbf{x}$. From the property of conditional independence, equality $p_{\mathcal{T}}(D|\mathbf{x}, Y) = p_{\mathcal{T}}(D|\mathbf{x})$ holds. And since $p_{\mathcal{T}}(D|\mathbf{x})$ depends only on a fixed classifier h, it is equal to $p_{\mathcal{E}}(D|\mathbf{x})$. Following

this observation and applying the label shift assumption, the equation above can be rewritten as

$$\sum_{\mathbf{x}\in\mathbb{X}} p_{\mathcal{E}}(D|\mathbf{x},Y)p_{\mathcal{E}}(\mathbf{x}|Y) = \sum_{\mathbf{x}\in\mathbb{X}} p_{\mathcal{E}}(D,\mathbf{x}|Y) = p_{\mathcal{E}}(D|Y).$$

□

Marginalizing over the joint density $p(D,Y)$

$$\begin{aligned} p(D=i) &= \sum_{k=1}^{K} p(D=i|Y=k)p(Y=k), \\ p(D) &= \mathbf{C}_{d|y}p(Y). \end{aligned} \tag{2.15}$$

McLachnan [41] and Saerens et al. [16] simply compute the new priors $p_{\mathcal{E}}(Y)$ from the Equation (2.15):

$$\hat{p}_{\mathcal{E}}(Y) = \hat{\mathbf{C}}_{d|y}^{-1}\hat{p}_{\mathcal{E}}(D), \tag{2.16}$$

using an estimate of $\mathbf{C}_{d|y}$ computed on a validation set $\mathcal{V}$

$$\hat{p}(D=i|Y=k) = \frac{|\{(\mathbf{x}_j,y_j)\in\mathcal{V} : d(x_j)=i \wedge y_j=k\}|}{|\{(\mathbf{x}_j,y_j)\in\mathcal{V} : y_j=k\}|}, \tag{2.17}$$

where $|\cdot|$ is the set cardinality, and an estimate of $p(D)$ computed by counting the classifier's decisions on the test set $\mathcal{E}$

$$\hat{p}_{\mathcal{E}}(D=i) = \frac{|\{(\mathbf{x}_j,y_j)\in\mathcal{E} : d(x_j)=i\}|}{|\mathcal{E}|}. \tag{2.18}$$

Let us also consider a *soft confusion matrix*[1] (SCM) $\mathbf{C}_{d|y}^{\text{soft}}$ estimated from the classifier's soft predictions $\mathbf{f}$ as

$$\hat{\mathbf{c}}_{:|k}^{\text{soft}} = \frac{1}{N_k}\sum_{\mathbf{x}_i:y_i=k} \mathbf{f}(\mathbf{x}_i), \tag{2.19}$$

where $\hat{\mathbf{c}}_{:|k}^{\text{soft}}$ denotes the k-th column of the SCM. The probability $p_{\mathcal{E}}^{\text{soft}}(D)$ can be estimated by averaging predictions $f(\mathbf{x})$ over the test set. The new priors are then computed similarly to the Equation (2.16).

[1] Following the terminology of Lipton et al. [20].

2.4.2 Adaptation Based on Maximum Likelihood

Saerens et al. [16] suggested to estimate the test time priors $p_{\mathcal{E}}(Y)$ by maximizing the likelihood

$$L(\mathcal{E}) = \prod_{i=1}^{M} p_{\mathcal{E}}(\mathbf{x}_i) = \prod_{i=1}^{M} \sum_{k=1}^{K} p_{\mathcal{E}}(\mathbf{x}_i, Y = k) = \prod_{i=1}^{M} \sum_{k=1}^{K} p_{\mathcal{E}}(\mathbf{x}_i | Y = k) p_{\mathcal{E}}(Y = k). \tag{2.20}$$

The proposed EM algorithm is working iteratively in two steps:

$$\hat{p}_{\mathcal{E}}^{(s)}(Y = k|\mathbf{x}_i) \coloneqq \frac{\hat{p}_{\mathcal{T}}(Y = k|\mathbf{x}_i) \frac{\hat{p}_{\mathcal{E}}^{(s)}(Y=k)}{\hat{p}_{\mathcal{T}}(Y=k)}}{\sum_{j=1}^{N'} \hat{p}_{\mathcal{T}}(Y = j|\mathbf{x}_i) \frac{\hat{p}_{\mathcal{E}}^{(s)}(Y=j)}{\hat{p}_{\mathcal{T}}(Y=j)}}, \tag{2.21}$$

$$\hat{p}_{\mathcal{E}}^{(s+1)}(Y = k) \coloneqq \frac{1}{M} \sum_{i=1}^{M} \hat{p}_{\mathcal{E}}^{(s)}(Y = k|\mathbf{x}_i). \tag{2.22}$$

In the first step, it recomputes the posterior predictions (Equation (2.21)). The classifier outputs are denoted as $\hat{p}_{\mathcal{T}}(Y = k|\mathbf{x}_i)$. In the second step, it estimates the test priors by averaging all updated predictions (Equation (2.22)). The algorithm requires initialization of test priors $\hat{p}_{\mathcal{E}}^{(0)}(Y = k)$. This can be done for example by the relative class frequencies in the training set, i.e. $\hat{p}_{\mathcal{T}}(Y = j) = M_j/M$.

Du Plessis and Sugiyama [15] showed that the EM algorithm, proposed by Saerens et al. [16], can be derived from minimization of KL divergence between $p_{\mathcal{E}}(\mathbf{x})$ and its approximation $p'_{\mathcal{E}}(\mathbf{x})$ modeled as

$$p'_{\mathcal{E}}(\mathbf{x}) = \sum_{k=1}^{K} \hat{P}_k p_{\mathcal{T}}(\mathbf{x}|Y = k), \tag{2.23}$$

where $\hat{P}_k \coloneqq \hat{p}_{\mathcal{E}}(Y = k)$. The KL divergence is expressed as

$$\begin{aligned} KL(p_{\mathcal{E}}||p'_{\mathcal{E}}) &= \int p_{\mathcal{E}}(\mathbf{x}) \log \frac{p_{\mathcal{E}}(\mathbf{x})}{p'_{\mathcal{E}}(\mathbf{x})} d\mathbf{x} = \\ &= \int p_{\mathcal{E}}(\mathbf{x}) \log p_{\mathcal{E}}(\mathbf{x}) d\mathbf{x} - \int p_{\mathcal{E}}(\mathbf{x}) \log \sum_{k=1}^{K} \hat{P}_k p_{\mathcal{T}}(\mathbf{x}|Y = k) d\mathbf{x}. \end{aligned} \tag{2.24}$$

The first integral depends only on $\mathbf{x}$. Consequently it is a constant for a fixed input and it can be ignored during minimization. The second integral

can be viewed as an expectation over $\log p'_{\mathcal{E}}(\mathbf{x})$ and can be approximated by empirical average

$$\int p_{\mathcal{E}}(\mathbf{x}) \log \sum_{y \in \mathbb{Y}} \hat{P}_k p_{\mathcal{T}}(\mathbf{x}|Y=k) d\mathbf{x} \approx \frac{1}{N} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \hat{P}_k p_{\mathcal{T}}(\mathbf{x}_i|Y=k). \quad (2.25)$$

These observations give the following optimization problem:

$$\begin{aligned}
\hat{\mathbf{P}}^* &= \arg\max_{\hat{\mathbf{P}}} \frac{1}{N} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \hat{P}_k p_{\mathcal{T}}(\mathbf{x}_i|Y=k) = \\
&= \arg\max_{\hat{\mathbf{P}}} \frac{1}{N} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \hat{P}_k \frac{p_{\mathcal{T}}(Y=k|\mathbf{x}_i) p_{\mathcal{T}}(\mathbf{x}_i)}{p_{\mathcal{T}}(Y=k)} = \\
&= \arg\max_{\hat{\mathbf{P}}} \frac{1}{N} \sum_{i=1}^{N} \log p_{\mathcal{T}}(\mathbf{x}_i) \sum_{k=1}^{K} \hat{P}_k \frac{p_{\mathcal{T}}(Y=k|\mathbf{x}_i)}{p_{\mathcal{T}}(Y=k)} = \\
&= \arg\max_{\hat{\mathbf{P}}} \frac{1}{N} \sum_{i=1}^{N} \log p_{\mathcal{T}}(\mathbf{x}_i) + \frac{1}{N} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \hat{P}_k \frac{p_{\mathcal{T}}(Y=k|\mathbf{x}_i)}{p_{\mathcal{T}}(Y=k)} = \\
&= \arg\max_{\hat{\mathbf{P}}} \frac{1}{N} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \hat{P}_k \frac{p_{\mathcal{T}}(Y=k|\mathbf{x}_i)}{p_{\mathcal{T}}(Y=k)} \\
&\quad \text{s.t.} \sum_{k=1}^{K} \hat{P}_k = 1; \quad \forall k : \hat{P}_k \geq 0
\end{aligned} \quad (2.26)$$

Indeed, this is equivalent to maximizing the logarithm of likelihood function from Equation (2.20). In other words, it is maximization of log-likelihood $l(\mathcal{E}) = \log L(\mathcal{E})$.

Sulc and Matas [17] experimented with maximizing the log-likelihood function by projected gradient ascent

$$\hat{P}_k^{s+1} = \pi \left(\hat{P}_k^s + \frac{\partial l(\mathcal{E})}{\partial \hat{P}_k} \right), \quad (2.27)$$

where $\pi(\cdot)$ denotes projection onto the probability simplex and

$$\frac{\partial l(\mathcal{E})}{\partial \hat{P}_k} = \sum_{i=1}^{M} \frac{\frac{p_{\mathcal{T}}(Y=k|\mathbf{x}_i)}{p_{\mathcal{T}}(Y=k)}}{\sum_{j=1}^{K} \hat{P}_k \frac{p_{\mathcal{T}}(Y=j|\mathbf{x}'_i)}{p_{\mathcal{T}}(Y=j)}}. \quad (2.28)$$

The experimental results showed that the EM algorithm converged faster than the gradient ascent, while achieving similar results.

2.4.3 Adaptation Based on Maximum Aposteriori

As an extension to the projected gradient ascent algorithm given in the Equation (2.27), Sulc and Matas [17] proposed maximum aposteriori estimate

$$\begin{aligned}\hat{\mathbf{P}}^* &= \arg\max_{\hat{\mathbf{P}}} p(\hat{\mathbf{P}}|\mathcal{E}) = \arg\max_{\hat{\mathbf{P}}} p(\hat{\mathbf{P}})p(\mathcal{E}|\hat{\mathbf{P}}) = \\ &= \arg\max_{\hat{\mathbf{P}}} \log p(\hat{\mathbf{P}}) + \underbrace{\log p(\mathcal{E}|\hat{\mathbf{P}})}_{l(\mathcal{E})},\end{aligned} \tag{2.29}$$

adding a hyper-prior $p(\hat{\mathbf{P}})$ to the log-likelihood function $l(\mathcal{E})$. The hyper-prior represents a prior knowledge about the categorical distribution $p_{\mathcal{E}}(Y)$ and was modeled by symmetric Dirichlet distribution

$$p(\hat{\mathbf{P}}) = \mathrm{Dir}(\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \hat{P}_k^{\alpha-1}, \tag{2.30}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, $\alpha_1 = \ldots = \alpha_K = \alpha$ and $B(\cdot)$ is the multivariate beta function. The intuition behind the Dirichlet distribution is that for $\alpha \geq 1$ the probability vector $\hat{\mathbf{P}}$ is forced to be a dense distribution, while for $0 \leq \alpha \leq 1$ being a sparse one. See Figure 2.4 for visualization.

The solution to the maximum a-posteriori can be found by projected gradient ascent, adding the derivative of $\log \mathrm{Dir}(\boldsymbol{\alpha})$ into the $\pi(\cdot)$ function in the Equation (2.27)

$$\frac{\partial \log p(\mathbf{P})}{\partial P_k} = \frac{\partial \log \mathrm{Dir}(\boldsymbol{\alpha})}{\partial P_k} = \frac{\alpha - 1}{P_k}. \tag{2.31}$$

2.4.4 BBSE

So far, all of the mentioned algorithms were approaching the problem of prior shift adaptation through estimating the test time prior $p_{\mathcal{E}}(Y)$ and the training prior $p_{\mathcal{T}}(Y)$ separately. But, as proposed by Lipton et al. [20], it is also possible to estimate directly the prior ratio $w(Y) = p_{\mathcal{E}}(Y)/p_{\mathcal{T}}(Y)$ using the confusion matrix $\mathbf{C}_{d,y}$ with joint probability $p(D = i, Y = k)$, unlike the conditional probability used in the Section 2.4.1.

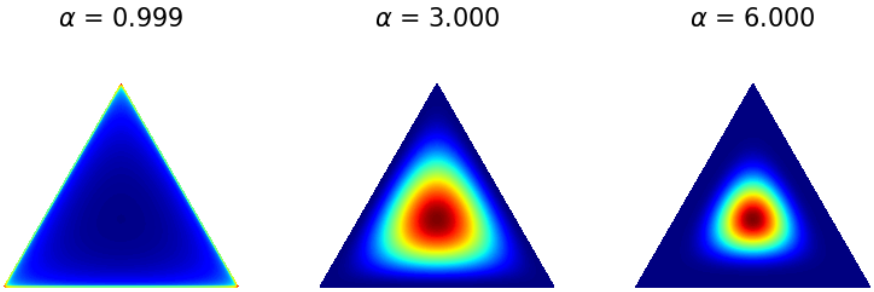


Figure 2.4: Visualization of Dirichlet distribution on 3-dimensional probability simplex. Red encodes higher probabilities, blue smaller probabilities.

Let us again consider the Equation (2.15) and multiply it by $\frac{p_{\mathcal{T}}(Y=k)}{p_{\mathcal{T}}(Y=k)}$

$$\begin{aligned} p_{\mathcal{E}}(D=i) &= \sum_{k=1}^{K} p_{\mathcal{T}}(D=i|Y=k)p_{\mathcal{E}}(Y=k) = \\ &= \sum_{k=1}^{K} p_{\mathcal{T}}(D=i|Y=k)p_{\mathcal{E}}(Y=k)\frac{p_{\mathcal{T}}(Y=k)}{p_{\mathcal{T}}(Y=k)} = \\ &= \sum_{k=1}^{K} p_{\mathcal{T}}(D=i,Y=k)\underbrace{\frac{p_{\mathcal{E}}(Y=k)}{p_{\mathcal{T}}(Y=k)}}_{w(Y=k)} = \\ &= \sum_{k=1}^{K} p(D=i,Y=k)w(Y=k), \\ p_{\mathcal{E}}(D) &= \mathbf{C}_{d,y}w(Y). \end{aligned} \tag{2.32}$$

The result is similar to the Equation (2.15), therefore it is not surprising that the ratio $w(Y)$ is computed in the very same way

$$\hat{w}(Y) = \hat{\mathbf{C}}_{d,y}^{-1}\hat{p}_{\mathcal{E}}(D). \tag{2.33}$$

The only difference is in the normalization of the confusion matrix: while in the conditional confusion matrix $\mathbf{C}_{d|y}$ columns sum up to one, in the joint confusion matrix $\mathbf{C}_{d,y}$ all elements sum up to one. The CM $\mathbf{C}_{d,y}$ with the joint probability is estimated as

$$\hat{p}(D=i,Y=k) = \frac{|\{(\mathbf{x}_j,y_j)\in\mathcal{V} : d(x_j)=i \wedge y_j=k\}|}{|\mathcal{V}|}. \tag{2.34}$$

It is also possible to use the soft confusion matrix $\mathbf{C}^{\text{soft}}_{d,y}$ computed from the classifier's predictions

$$\hat{\mathbf{c}}^{\text{soft}}_{:,k} = \frac{1}{N} \sum_{\mathbf{x}_i : y_i = k} \mathbf{f}(\mathbf{x}_i), \tag{2.35}$$

where $\hat{\mathbf{c}}^{\text{soft}}_{:,k}$ denotes the k-th column of SCM. The probability $p^{\text{soft}}_{\mathcal{E}}(D)$ can be estimated by averaging predictions $f(\mathbf{x})$ over the test set. The ratio is then computed in the same way as in the Equation (2.33).

This estimation is called the Black Box Shift Estimation (BBSE). A variant using a soft confusion matrix $\mathbf{C}^{\text{soft}}_{d,y}$ is denoted BBSE-S.

Note that the validation set used for the estimation of $\mathbf{C}_{d,y}$, should have the same label distribution as the training set. Otherwise we have to re-weight rows of $\mathbf{C}_{d,y}$ by the ratio of training and validation set priors $p_{\mathcal{T}}(Y)/p_{\mathcal{V}}(Y)$.

The fact that $\mathbf{C}_{d,y}$ estimated on a validation set with $p_{\mathcal{T}}(Y) \neq p_{\mathcal{V}}(Y)$ cannot be simply used for estimation of $w(Y)$ is apparent from the following inequality:

$$p_{\mathcal{V}}(D, Y) = p_{\mathcal{V}}(D|Y)p_{\mathcal{V}}(Y) = p_{\mathcal{T}}(D|Y)p_{\mathcal{V}}(Y) \neq p_{\mathcal{T}}(D|Y)p_{\mathcal{T}}(Y) = p_{\mathcal{T}}(D, Y). \tag{2.36}$$

In order to use the validation set, we need to multiply $p_{\mathcal{V}}(D, Y)$ by $p_{\mathcal{T}}(Y)/p_{\mathcal{V}}(Y)$, to make it equal to $p_{\mathcal{T}}(D, Y)$. This observation means that the BBSE also requires access to the training set.

2.4.5 RLLS

Azizzadenesheli et al. [19] proposed to estimate the prior ratio $w(Y)$ in two steps. First, because in case of no label shift prior ratio is equal to the vector of ones, i.e. $w(Y) = \mathbf{1}$, they define amount of weight shift

$$\theta \coloneqq w(Y) - \mathbf{1} \tag{2.37}$$

and propose to find θ instead of $w(Y)$. This way it is possible to use l_2-regularization in the optimization procedure to penalize large shift. Plugging θ into the Equation (2.32) we get

$$\begin{aligned} p_{\mathcal{E}}(D) &= \mathbf{C}_{d,y}(\theta + \mathbf{1}) \\ p_{\mathcal{E}}(D) - \mathbf{C}_{d,y}\mathbf{1} &= \mathbf{C}_{d,y}\theta \\ b &= \mathbf{C}_{d,y}\theta, \end{aligned} \tag{2.38}$$

where $b \coloneqq p_{\mathcal{E}}(D) - \mathbf{C}_{d,y}\mathbf{1}$. Azizzadenesheli et al. choose to find θ in the Equation (2.38) by minimizing l_2-norm with a regularization term to prevent large shifts

$$\hat{\theta} = \arg\min_{\theta} \left\| \hat{\mathbf{C}}_{d,y}\theta - \hat{b} \right\|_2 + \Delta_C \left\| \theta \right\|_2 , \tag{2.39}$$

where Δ_C is a hyper-parameter, which can be found by cross validation and $\hat{\mathbf{C}}_{d,y}$ and $\hat{p}_{\mathcal{E}}(D)$ are estimated in the same way as in the Section 2.4.4. The solution to the problem given by the Equation (2.39) is found by a third party solver.

In the second step the prior ratio is computed as

$$\hat{w}(Y) = \mathbf{1} + \lambda\hat{\theta}, \tag{2.40}$$

where

$$\lambda = \begin{cases} 1, & \text{if } N \geq \frac{1}{\theta_{max}^2(\sigma_{min} - \frac{1}{\sqrt{M}})^2} \\ 0, & \text{otherwise.} \end{cases} \tag{2.41}$$

M is the number of samples in the training set, N is the number of samples in the test set, σ_{min} is the smallest singular value corresponding to the confusion matrix $\hat{\mathbf{C}}_{d,y}$ and θ_{max} represents our belief about the size of the shift.

Determination of λ in the Equation (2.41) serves as an indicator of how bad our estimate is. If the estimation of θ is based on an insufficient number of samples or the confusion matrix is close to the singular matrix (its smallest singular value is close to zero), then it is better to decide for no shift and set λ to zero.

2.5 Classifier Calibration

It is important to know how much we can trust the decision of our classifier. Therefore, having a probabilistic classifier $f(x)$ trained to approximate posterior probabilities $p(Y|X)$, we would like the outputs to reflect the true probabilities of correct and incorrect classification. For example, from all the samples being classified with a confidence of 0.9, ninety percent of them should be classified correctly and ten percent of them incorrectly. Such property is desirable also in algorithms for adapting classifiers to the label shift because most of the methods work with the assumption that the classifiers approximate posterior probability. Unfortunately, common probabilistic classifiers, such as Convolution Neural Networks, tend to have overconfident predictions due to over-fitting to the training set.

When the classifier's outputs reflect its predictive uncertainty, we call the classifier calibrated. In the literature there are several definitions for calibrated classifier [42, 43]:

Definition 2.2. *A probabilistic classifier f is **multiclass-calibrated**, or simply **calibrated**, if for set of all samples with prediction vector $f(\mathbf{x})$, the proportions of classes in this set are equal to $f(\mathbf{x})$:*

$$p(Y|f(\mathbf{x})) = f(\mathbf{x}). \tag{2.42}$$

Another weaker definition was proposed by [44]. The definition is weaker in the sense that it does not consider whole prediction vectors, but every class in the vector separately.

Definition 2.3. *A probabilistic classifier f is **classwise-calibrated**, if for any class i and any predicted probability q_i for this class:*

$$p(Y = i|f^{(i)}(\mathbf{x}) = q_i) = q_i. \tag{2.43}$$

The definition basically says that among all samples predicted to belong into class i with probability q_i, the proportion of samples with ground truth class i is equal to q_i. Yet, one more even weaker definition exists [45]:

Definition 2.4. *A probabilistic classifier f is **confidence-calibrated** if*

$$p(Y = \arg\max f(\mathbf{x})|\max f(\mathbf{x})) = \max f(\mathbf{x}). \tag{2.44}$$

In order to better understand the definitions 2.2, 2.3 and 2.4 let us demonstrate them on several examples:

Example 1. Suppose classification task into a three classes $\mathbb{Y} = \{1, 2, 3\}$, with label distribution $p(Y) = (0.5, 0.1, 0.4)$, and a classifier with a constant output $f(\mathbf{x}) = (0.5, 0.1, 0.4)$. According to Definition 2.2 classifier $f(\mathbf{x})$ is multiclass-calibrated. Note that the classifier is also classwise-calibrated and confidence-calibrated according to Definition 2.3 and 2.4 respectively.

Example 2. Let us consider classification task into a three classes $\mathbb{Y} = \{1, 2, 3\}$. Classifier $f(\mathbf{x})$ outputs four predictions, displayed in the first column of Table 2.1. The true proportions of classes among samples having particular output vector $f(\mathbf{x})$ are in the second column of Table 2.1. The classifier is classwise-calibrated and confidence calibrated, but not multiclass-calibrated.

$f(X)$	$p(Y\|f(X))$
(0.2, 0.5, 0.3)	(0.3, 0.4, 0.3)
(0.2, 0.3, 0.5)	(0.1, 0.3, 0.6)
(0.3, 0.5, 0.2)	(0.2, 0.6, 0.2)
(0.3, 0.2, 0.5)	(0.4, 0.2, 0.4)

Table 2.1: Classifier $f(X)$ is given by its outputs in the first column. The true distribution of samples having prediction $f(X)$ are in second column.

Example 3. Let us consider classification task into a three classes $\mathbb{Y} = \{1, 2, 3\}$. Table 2.2 gives outputs of classifier $f(\mathbf{x})$ in the first column and corresponding ground truth relative frequencies of classes in the second column. The classifier is only confidence-calibrated according to Definition 2.4.

$f(X)$	$p(Y\|f(X))$
(0.2, 0.7, 0.1)	(0.0, 0.7, 0.3)
(0.2, 0.2, 0.6)	(0.3, 0.1, 0.6)

Table 2.2: Classifier $f(X)$ is given by its outputs in the first column. The true distribution of samples having prediction $f(X)$ are in second column.

Vaicenavicius et al. [43] formulated an evaluation framework for classifier calibration, under which all weaker definitions are only special cases of the Definition 2.2. In the framework, the authors introduced so-called *calibration lenses*, which allow us to evaluate calibration on simpler induced problems. It is particularly useful in multiclass problems, where we usually do not have enough samples to evaluate general calibration. An example of such induced problem is transforming the K-classification task into binary classification, where a subset of K classes is merged into one class and the rest into the second class. Indeed, evaluating a classifier on an induced problem has one drawback: Although the calibrated classifier is also calibrated on the induced problem, it does not apply contrariwise [43].

2.5.1 Evaluating Calibration

Let us first define *canonical calibration function* [43]:

Definition 2.5. ***Canonical calibration function*** $\mu : \Delta_{K-1} \to \Delta_{K-1}$ *of classifier* $f : \mathbb{X} \to \Delta_{K-1}$ *is*

$$\mu(q) \coloneqq p(Y|f(\mathbf{x}) = q) \quad \forall q \in \Delta_{K-1}. \tag{2.45}$$

Canonical calibration function is a mapping, which transforms the classifier's outputs into calibrated predictions. Therefore, the classifier is multiclass-calibrated if its calibration function is identity. Now we can define expected miscalibration [43]:

Definition 2.6. ***Expected miscalibration*** *on $A \subseteq \Delta_{K-1}$ with respect to some distance d is*

$$\eta_d \coloneqq \mathbb{E}[d(\mu(f(X)), f(X))|f(X) \in A]. \tag{2.46}$$

Here A only denotes a subset of probability simplex on which we want to evaluate the calibration of our classifier.

Let $\Phi = \{\Phi_i\}_{i=1}^{L}$ be a partition of $A \subseteq \Delta_{K-1}$, then the estimator of calibration function is defined as

$$\hat{\mu}^{(j)}(q) \coloneqq \frac{|\{i : f(x_i) \in \Phi[q] \wedge y_i = j\}|}{|\{i : f(x_i) \in \Phi[q]\}|} \quad \text{for} \quad j = 1 \dots K, \tag{2.47}$$

where $\Phi[q]$ denotes the partition for which $q \in \Phi_k$ holds. To summarize it, we split subset A into L partitions (bins) and in every bin we compute relative frequencies for every class (from samples falling into the bin based on the classifier's predictions). These relative frequencies then determine the estimate of the calibration function in a particular bin. The estimator of the canonical function is constant on every partition, therefore we can define $\hat{\mu}_i$ as a value of the estimator in the partition Φ_i.

Next, we can define an estimator of expected miscalibration as

$$\hat{\eta}_d \coloneqq \sum_{i=1}^{L} \hat{p}_i d(\hat{\mu}_i, \hat{f}_i), \tag{2.48}$$

where $\hat{p}_i$ is the proportion of samples from the data set falling into the bin Φ_i

$$\hat{p}_i = \frac{|\{j : f(\mathbf{x}_j) \in \Phi_i\}|}{|\{j : f(\mathbf{x}_j) \in A\}|} \tag{2.49}$$

and $\hat{f}_i$ is the average of all predictions in the bin Φ_i

$$\hat{f}_i = \frac{\sum_{j:f(\mathbf{x}_j)\in\Phi_i} f(\mathbf{x}_j)}{|\{j : f(\mathbf{x}_j) \in \Phi_i\}|}. \tag{2.50}$$

Note that whenever the denominator in Equations (2.47),(2.49), (2.50) is zero, the value of the particular estimator is set to 0. The estimator of expected

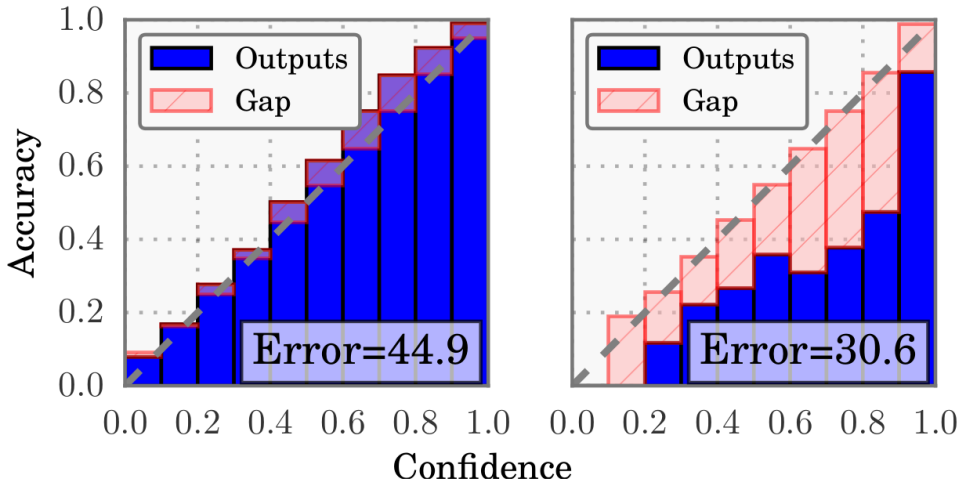


Figure 2.5: An example of reliability diagram, taken from [45]. The error denotes Expected calibration error.

miscalibration is nothing else than weighted distance between the histogram based estimation of calibration function and the classifier's predictions.

Vaicenavicius et al. [43] showed that as the number of partitions and the number of data samples grow to infinity, the estimator of expected miscalibration $\hat{\eta}_d$ is equal to η_d. However, if the number of partitions is fixed and only the number of data samples grow to infinity, then the estimator is only a lower bound $\hat{\eta}_d \leq \eta_d$ and $\hat{\eta}_d$ is an inconsistent estimate. The intuition behind this fact is that when the predictions in the bin are averaged, the overconfident and underconfident estimates can cancel out.

Guo et al. [45] used expected miscalibration for induced binary classifier with total variation distance $d(x, y) = \|x - y\|_1$. This special case of expected miscalibration is called *Expected calibration error* (ECE).

The visualization of model calibration is done with *reliability diagrams* [43, 45, 46]. They are mostly used as one-dimensional graphs, plotting accuracy vs. confidence (see Figure 2.5), but multidimensional diagrams were proposed as well [43]. Note that the multidimensional diagrams suffer from a lack of data due to the curse of dimensionality. In general, reliability diagrams plot information about $\hat{\mu}$ and $\hat{f}$ for all partitions Φ_i.

2.5.2 Calibration Methods

Guo et al. [45] evaluated several calibration methods in the context of neural network classifiers and observed that temperature scaling performs best in most cases.

Temperature scaling (TS) has a single real valued parameter $T > 0$, called temperature. The neural network's logits are divided by T right before the softmax layer

$$f_{TS}^{(i)}(\mathbf{x}) = \frac{\exp(z_i(\mathbf{x})/T)}{\sum_{j=1}^{K} \exp(z_j(\mathbf{x})/T)}. \tag{2.51}$$

TS does not affect the accuracy of the model, because the class with maximum probability remains the same, only the confidence for the decision changes. For $T > 1$ the resulting prediction is closer to the uniform distribution (confidence is decreasing)

$$\begin{aligned} &\lim_{T\to\infty} \frac{\exp(z_i(\mathbf{x})/T)}{\sum_{j=1}^{K} \exp(z_j(\mathbf{x})/T)} = \lim_{T\to\infty} \frac{1}{\sum_{j=1}^{K} \frac{\exp(z_j(\mathbf{x})/T)}{\exp(z_i(\mathbf{x})/T)}} = \\ &= \frac{1}{\sum_{j=1}^{K} \lim_{T\to\infty} \frac{\exp(z_j(\mathbf{x})/T)}{\exp(z_i(\mathbf{x})/T)}} = \frac{1}{\sum_{j=1}^{K} \lim_{T\to\infty} e^{(z_j(\mathbf{x})-z_i(\mathbf{x}))/T}} = \\ &= \frac{1}{\sum_{j=1}^{K} 1} = \frac{1}{K}, \end{aligned} \tag{2.52}$$

while for $T < 1$ the prediction for the most likely class is getting closer to one and the prediction for the rest of the classes are getting closer to zero (the confidence is increasing). This can be seen by computing the limit

$$\begin{aligned} &\lim_{T\to 0^+} \frac{\exp(z_i(\mathbf{x})/T)}{\sum_{j=1}^{K} \exp(z_j(\mathbf{x})/T)} = \lim_{T\to 0^+} \frac{1}{\sum_{j=1}^{K} \frac{\exp(z_j(\mathbf{x})/T)}{\exp(z_i(\mathbf{x})/T)}} = \\ &= \frac{1}{\sum_{j=1}^{K} \lim_{T\to 0^+} \frac{\exp(z_j(\mathbf{x})/T)}{\exp(z_i(\mathbf{x})/T)}} = \frac{1}{\sum_{j=1}^{K} \lim_{T\to 0^+} e^{(z_j(\mathbf{x})-z_i(\mathbf{x}))/T}} = \\ &= \frac{1}{1 + \sum_{j:j\neq i} \lim_{T\to 0^+} e^{(z_j(\mathbf{x})-z_i(\mathbf{x}))/T}}. \end{aligned} \tag{2.53}$$

If the output for class i is not the maximum, then there is at least one j for which $z_j > z_i$, therefore making limit $\lim_{T\to 0^+} e^{(z_j(\mathbf{x})-z_i(\mathbf{x}))/T} = \infty$ and thus the limit in the Equation (2.53) is equal to 0. If the output for class i is the maximum, then $z_j \leq z_i \quad \forall j \in \{1 \dots K\}$ and $\lim_{T\to 0^+} e^{(z_j(\mathbf{x})-z_i(\mathbf{x}))/T} = 0 \quad \forall j : j \neq i$ Therefore the limit in Equation (2.53) is equal to 1.

Besides temperature scaling Guo et al. [45] experimented also with *matrix scaling* (MS) and *vector scaling* (VS). Matrix scaling transforms the logit vector $\mathbf{z}(\mathbf{x})$ by linear transformation and then applies the softmax

$$f_{MS}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{z}(\mathbf{x}) + \mathbf{b}). \tag{2.54}$$

where σ denotes softmax function. In case of vector scaling, the matrix $\mathbf{W}$ is restricted to be diagonal matrix. It can be rewritten in the form, where every logit is multiplied by a different scalar weight

$$f_{VS}^{(i)}(\mathbf{x}) = \frac{\exp(W_i z_i(\mathbf{x}) + b_i)}{\sum_{j=1}^{K} \exp(W_j z_j(\mathbf{x}) + b_j)}. \tag{2.55}$$

While TS does not change the argument of maxima, in the case of MS and VS, there is no such guarantee.

Alexandari et al. [47] proposed the Bias-Corrected Temperature Scaling (BCTS) as an intermediary between vector scaling and temperature scaling

$$f_{BCTS}^{(i)}(\mathbf{x}) = \frac{\exp(z_i(\mathbf{x})/T + b_i)}{\sum_{j} \exp(z_j(\mathbf{x})/T + b_j)}. \tag{2.56}$$

They showed that although the TS can effectively reduce the ECE error, it is not suitable in combination with prior shift adaptation, because of systematic bias and that BCTS in combination with EM algorithm [16] from the Section 2.4.2, outperforms both BBSL [20] ad RLLS [19].

The parameters in all of the aforementioned methods are learned via negative log-likelihood (NLL), also called cross-entropy (given by Equation (2.7), on the validation set. The parameters are learned separately from the training procedure, otherwise, they would be learned along with other classifier's parameters to output overconfident predictions.

Chapter 3

The Proposed Method

"When I find myself in the company of scientists, I feel like a shabby curate who has strayed by mistake into a room full of dukes."

— *W. H. Auden*

As observed in the literature [21, 41, 18], in some cases the Equation (2.16) can result in a vector outside of the Δ_{K-1} simplex, i.e. the estimate can contain negative values.

Following the Equation (2.15), the probability of the classifier's decisions $p(D)$ is a convex combination of columns in $\mathbf{C}_{d|y}$ as $p(Y) \in \Delta_{K-1}$. Since the columns of the confusion matrix are probability vectors, they define a convex set $\Phi_{\mathbf{C}}$ of feasible values $p(D)$ within the probability simplex Δ_{K-1}. In other words, a classifier with the confusion matrix $\mathbf{C}$ will result in decisions from $p(D) \in \Phi_{\mathbf{C}}$. The class distribution $p(Y)$ determines the value of $p(D)$ within $\Phi_{\mathbf{C}}$. See Figure 3.1 for illustration. For the true distribution of decisions $p(D)$ and confusion matrix $\mathbf{C}_{d|y}$, the Equation (2.15) holds. The problem occurs when we work with estimates of the distribution $\hat{p}(D)$ and confusion matrix $\hat{\mathbf{C}}_{d|y}$. If the estimates computed from a limited set of samples are not consistent, meaning that the classifier with confusion matrix $\hat{\mathbf{C}}_{d|y}$, on data set with priors $\hat{p}(Y)$, would never produce decisions with distribution $\hat{p}(D)$, there may be no prior probability $\hat{p}(Y)$ satisfying the Equation (2.15). For example, having

$$\hat{\mathbf{C}}_{d|y} = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}, \; \hat{p}(D) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

the unique solution to the Equation (2.15) is $\hat{p}(Y) = \begin{bmatrix} \frac{4}{3} \\ -\frac{1}{3} \end{bmatrix}$.

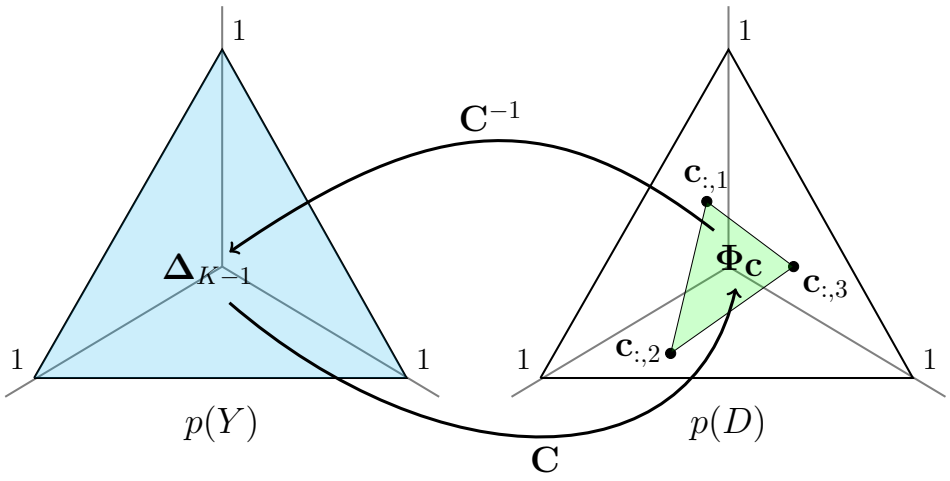


Figure 3.1: The convex set $\Phi_{\mathbf{C}} \subset \Delta_{K-1}$ of all possible values of $p(D)$ for a classifier with confusion matrix $\mathbf{C}$.

The prior ratio estimation also suffers from the problem of inconsistent estimates. Let us consider the following example, resulting in a negative value in the estimated prior ratio, which – as a ratio of two probabilities – should be non-negative:

$$\hat{\mathbf{C}}_{d,y} = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix}, \; \hat{p}(D) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow \hat{w}(Y) = \begin{bmatrix} \frac{8}{3} \\ -\frac{2}{3} \end{bmatrix}. \tag{3.1}$$

3.1 Maximum Likelihood Approach

This thesis proposes a novel procedure for the prior estimation based on maximizing the likelihood of $p_{\mathcal{E}}(D)$, which handles inconsistent estimates of $\hat{p}(D)$ and the confusion matrix $\hat{\mathbf{C}}_{d|y}$ and can even work with singular matrices $\hat{\mathbf{C}}_{d|y}$, as it does not use matrix inversion.

Let $\mathbf{n} = (n_1, \ldots, n_K)$ be the numbers of the classifier's decisions for class $1, \ldots, K$ on test set $\mathcal{E}$ and let us denote $\mathbf{Q} = (q_1, \ldots, q_k) \coloneqq p_{\mathcal{E}}(D)$ the probabilities of the classifier decisions on the test distribution $p_{\mathcal{E}}(X, Y)$. Assuming the independence of classifier decisions on the test set $\mathcal{E}$, the

likelihood of $\mathbf{Q}$ follows by the multinomial distribution

$$L(\mathbf{Q}) = p(\mathbf{n}|\mathbf{Q}) = \frac{(n_1 + \ldots + n_K)!}{n_1! \cdot \ldots \cdot n_K!} \cdot q_1^{n_1} \cdot \ldots \cdot q_K^{n_K}. \tag{3.2}$$

Substituting the Equation (2.15) into the likelihood function $L(\mathbf{Q})$, we can express the likelihood function of class priors $\mathbf{P}$

$$L(\mathbf{P}) = p(\mathbf{n}|\mathbf{P}) = \frac{(n_1 + \ldots + n_K)!}{n_1! \cdot \ldots \cdot n_K!} \prod_{k=1}^{K} (\mathbf{c}_{k|:} \cdot \mathbf{P})^{n_k}, \tag{3.3}$$

where $\mathbf{c}_{k|:}$ is the k-th row of $\mathbf{C}_{d|y}$.

The log-likelihood is

$$\ell(\mathbf{P}) = \log p(n|\mathbf{P}) = \sum_{k=1}^{K} n_k \log(\mathbf{c}_{k|:} \cdot \mathbf{P}) + \theta_{\mathbf{n}}, \tag{3.4}$$

where $\theta_{\mathbf{n}}$ is constant for a fixed $\mathbf{n}$.

We estimate the new class priors by maximizing the log-likelihood from Equation (3.4):

$$\hat{\mathbf{P}} = \arg\max_{\mathbf{P}} \ell(\mathbf{P}) = \arg\max_{\mathbf{P}} \sum_{k=1}^{K} n_k \log \mathbf{c}_{k|:}\mathbf{P} \tag{3.5a}$$

$$\text{s.t.:} \quad \sum_{k=1}^{K} P_k = 1; \quad \forall k : P_k \geq 0. \tag{3.5b}$$

By this formulation we are avoiding estimation of $p_{\mathcal{E}}(D)$ as a potential source of inconsistency. Note that $p_{\mathcal{E}}(D)$ is estimated implicitly with $\mathbf{P}$. The convex objective can be iteratively maximized using projected gradient ascent

$$\hat{\mathbf{P}}^{s+1} = \pi\left(\hat{\mathbf{P}}^{s} + \nabla\ell(\mathbf{P}^{s})\right), \tag{3.6}$$

where $\pi(\cdot)$ denotes projection onto the probability simplex and the gradient is computed as

$$\nabla\ell(\mathbf{P}) = \sum_{k=1}^{K} \frac{n_k}{\mathbf{c}_{k,:} \cdot \mathbf{P}} \mathbf{c}_{k,:}. \tag{3.7}$$

We denote this method using $\mathbf{C}_{d|y}$ as CM^L and the one using soft confusion matrix $\mathbf{C}_{d|y}^{soft}$ as SCM^L.

3.2 Maximum Aposteriori Approach

Additional knowledge about the distribution $\mathbf{P}$ can be formulated as a hyper-prior $p(\mathbf{P})$. We can then extend the proposed procedure from Section 3.1 to formulate maximum a-posteriori (MAP) estimation:

$$\begin{aligned}\hat{\mathbf{P}}_{\mathbf{MAP}} &= \arg\max_{\mathbf{P}} p(\mathbf{P}|\mathbf{n}) = \arg\max_{\mathbf{P}} p(\mathbf{P})p(\mathbf{n}|\mathbf{P}) \\ &= \arg\max_{\mathbf{P}} \log p(\mathbf{P}) + \arg\max_{\mathbf{P}} \log p(\mathbf{n}|\mathbf{P}) \\ \text{s.t.:} \quad & \forall k : P_k \geq 0; \quad \sum_{k=1}^{K} P_k = 1,\end{aligned} \tag{3.8}$$

where $p(\mathbf{P})$ denotes a hyper-prior on $\mathbf{P}$ and $\log p(\mathbf{n}|\mathbf{P})$ is log-likelihood given by Equation (3.4).

Following [17] we use a symmetric Dirichlet hyper-prior $\mathrm{Dir}(\alpha)$, given by Equation (2.30), favoring dense distributions $\mathbf{P}$ with $\alpha > 1$, and a sparse distribution for $0 < \alpha < 1$.

The solution to maximum a-posteriori can be found by projected gradient ascent, adding the hyper-prior derivative from Equation (2.31) into the $\pi(\cdot)$ function in Equation (3.6).

We denote this method using $\mathbf{C}_{d|y}$ as CM^M and the one using a soft confusion matrix $\mathbf{C}^{soft}_{d|y}$ as SCM^M.

3.3 Applying Regularization

As extension to the method described in the Section 3.1, we propose to add a regularization term in the Equation (3.5):

$$\hat{\mathbf{P}} = \arg\max_{\mathbf{P}} \ \ell(\mathbf{P}) - \lambda||\mathbf{P} - p_{\mathcal{T}}(Y)||_2^2 =$$

$$= \arg\max_{\mathbf{P}} \sum_{k=1}^{K} n_k \log \mathbf{c}_{k|:}\mathbf{P} - \lambda||\mathbf{P} - p_{\mathcal{T}}(Y)||_2^2 \tag{3.9a}$$

$$\text{s.t.:} \quad \sum_{k=1}^{K} P_k = 1; \quad \forall k : P_k \geq 0, \tag{3.9b}$$

where λ is a hyperparameter. The intuition behind the regularization term is to penalize too large shifts from the training priors $p_{\mathcal{T}}(Y)$. Note the minus sign before regularization term to ensure minimization of the l_2-norm.

The task given by Equation (3.9) can be solved by projected gradient ascent

$$\hat{\mathbf{P}}^{s+1} = \pi\left(\hat{\mathbf{P}}^{s} + \nabla\ell(\mathbf{P}^{s}) - \lambda\nabla||\mathbf{P}^{s} - p_{\mathcal{T}}(Y)||_2^2\right), \tag{3.10}$$

where $\nabla\ell(\mathbf{P})$ is given by Equation (3.7) and

$$\nabla||\mathbf{P} - p_{\mathcal{T}}(Y)||_2^2 = 2(\mathbf{P} - p_{\mathcal{T}}(Y)). \tag{3.11}$$

We denote this method using $\mathbf{C}_{d|y}$ as CM^{LR} and the one using a soft confusion matrix $\mathbf{C}_{d|y}^{soft}$ as SCM^{LR}.

3.4 Ratio Estimation

Following the derivation of CM^{L} and SCM^{L} methods in Section 3.1, we can also estimate priors ratio $w(Y) = p_{\mathcal{E}}(Y)/p_{\mathcal{T}}(Y)$, by using joint confusion matrix $\mathbf{C}_{d,y}$ instead of conditional one $\mathbf{C}_{d|y}$

$$\hat{w}^{*} = \arg\max_{w} l(w) = \arg\max_{w} \sum_{k=1}^{K} n_k \log \mathbf{c}_{k,:} w \tag{3.12a}$$

$$\text{s.t.:} \quad w_i \geq 0 \quad \forall i \in \{1, \ldots, K\}; \quad w^T p_{\mathcal{T}}(Y) = 1. \tag{3.12b}$$

The solution to the problem given by Equation (3.12) can be found by projected gradient ascent

$$\hat{w}^{s+1} = \kappa\left(\hat{w}^{s} + \nabla\ell(\hat{w}^{s})\right), \tag{3.13}$$

where $\kappa(\cdot)$ is projection onto the priors ratio simplex (see Section 3.4.1) and

$$\nabla\ell(w) = \sum_{k=1}^{K} \frac{n_k}{\mathbf{c}_{k,:} \cdot w} \mathbf{c}_{k,:}. \tag{3.14}$$

We denote this method using $\mathbf{C}_{d,y}$ as CM^{W} and the one using a soft confusion matrix $\mathbf{C}_{d,y}^{soft}$ as SCM^{W}.

3.4.1 Projection onto the Priors Ratio Simplex

The projection of vector $\mathbf{y}$ to the probability ratio $\mathbf{w}$ (in this section bold $\mathbf{w}$ will be used instead of w to emphasize that it is a vector) is a quadratic problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{y}||_2^2 \tag{3.15a}$$

$$\text{s.t.:} \quad \mathbf{w} \geq \mathbf{0}; \quad \mathbf{w}^T\mathbf{z} = 1, \tag{3.15b}$$

where $z_i > 0 \; \forall i \in \{1, \ldots, K\}$. Note that in our task for the ratio estimation (3.12) the vector $\mathbf{z} = \hat{p}_{\mathcal{T}}(Y)$ and thus sums up to 1. However, the algorithm proposed to solve the task given by the Equation (3.15) only requires the vector $\mathbf{z}$ to have positive values, therefore we do not mention the condition here as it is only a special case. We denote the set of possible solutions, given by the constraints in Equation (3.15b), as *priors ratio simplex*. The solution to the task can be found by the Algorithm 1, which is a generalization of the algorithm for projection onto the probability simplex, described in [48]:

Algorithm 1: Projection onto the priors ratio simplex

Input : $\mathbf{y}, \mathbf{z} \in R^D$, where $z_i > 0 \quad \forall i \in \{1, \ldots D\}$

Compute $y'_i = \frac{y_i}{z_i} \quad \forall i \in \{1, \ldots D\}$;

Sort y'_i into $u : u_1 \geq u_2 \geq \ldots \geq u_D$ Find

$\rho = max\{1 \leq j \leq D : u_j + \frac{1}{\sum_{i=1}^{j} z_i^2} \left(1 - \sum_{i=1}^{j} u_i z_i^2\right) > 0\}$;

Define $\lambda = \frac{1}{\sum_{i=1}^{\rho} z_i^2} \left(1 - \sum_{i=1}^{\rho} u_i z_i^2\right)$;

Output : $\mathbf{w}$, where $w_i = max\{y_i + \lambda z_i, 0\}$

We proof correctness of the Algorithm 1 by modifying the derivation of algorithm for projection onto the probability simplex from [48]:

Proof. Lagrangian of the problem, given by the Equation 3.15 is

$$\mathcal{L}(\mathbf{x}, \lambda, \boldsymbol{\beta}) = \frac{1}{2}||\mathbf{w} - \mathbf{y}||_2^2 - \lambda(\mathbf{w}^T\mathbf{z} - 1) - \boldsymbol{\beta}^T\mathbf{w}, \tag{3.16}$$

where λ is a lagrange multiplier for equality constraint and $\boldsymbol{\beta}$ is a lagrange multiplier for inequality constraint. Let us write KKT conditions for optimal

solution $\mathbf{w}^*$

$$w_i - y_i - \lambda z_i - \beta_i = 0, \quad \forall i \in \{1 \dots K\} \tag{3.17a}$$
$$w_i \geq 0, \quad \forall i \in \{1 \dots K\} \tag{3.17b}$$
$$\beta_i \geq 0, \quad \forall i \in \{1 \dots K\} \tag{3.17c}$$
$$\beta_i w_i = 0, \quad \forall i \in \{1 \dots K\} \tag{3.17d}$$
$$\sum_{i=1}^{D} w_i z_i = 1, \quad \forall i \in \{1 \dots K\}. \tag{3.17e}$$

Next, let us define $w'_i = w_i/z_i$, $y'_i = y_i/z_i$ and $\beta'_i = \beta_i/z_i$. Note that $z_i > 0$, whence w'_i and y'_i have the same sign as w_i and y_i respectively. From condition (3.17d) we observe that if $w_i > 0$, then $\beta_i = 0$ and $w_i = y_i + \lambda z_i \rightarrow w'_i = y'_i + \lambda$. If $w_i = 0$, then $\beta_i \geq 0$ and $y_i + \lambda z_i = -\beta_i < 0 \rightarrow y'_i + \lambda = -\beta'_i < 0$. This observation means that smaller components in $\mathbf{w}'$ correspond to smaller components in $\mathbf{y}'$ and when $w'_i = 0$, then $y'_i + \lambda < 0$. Without the loss of generality we can assume the components in $\mathbf{w}'$ and $\mathbf{y}'$ to be sorted

$$w'_1 \geq \ldots \geq w'_\rho \geq w'_{\rho+1} \geq \ldots \geq w'_D,$$
$$y'_1 \geq \ldots \geq y'_\rho \geq y'_{\rho+1} \geq \ldots \geq y'_D.$$

Here ρ denotes the last index where w'_i is greater than zero, i.e. $\forall i \leq \rho : w'_i > 0$ and $\forall i > \rho : w'_i = 0$. From the last condition in Equation (3.17e), we get

$$1 = \sum_{i=1}^{D} w_i z_i = \sum_{i=1}^{D} w'_i z_i^2 = \sum_{i=1}^{\rho} w'_i z_i^2 = \sum_{i=1}^{\rho} (y'_i + \lambda) z_i^2.$$

Now we can express λ as

$$\lambda = \frac{1}{\sum_{i=1}^{\rho} z_i^2} \left(1 - \sum_{i=1}^{\rho} y'_i z_i^2\right). \tag{3.18}$$

Therefore to obtain final solution $\mathbf{w}^*$ we need to know only ρ, which will allow us to compute λ and then the final solution $w_i = y_i + \lambda z_i$, for $\forall i : 1, \ldots, \rho$ and $w_i = 0$, for $\forall i : \rho + 1, \ldots, D$. The ρ can be found by computing $y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2} \left(1 - \sum_{i=1}^{j} y'_i z_i^2\right)$ for every $j : 1 \leq j \leq D$ and then selecting the greatest j for which the quantity is greater than 0. Whence in the last step of the proof we need to show that $y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2} \left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) > 0$, $\forall j \leq \rho$ and $y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2} \left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) < 0$, $\forall j > \rho$:

- For $j = \rho$:

$$y'_\rho + \frac{1}{\sum_{i=1}^{\rho} z_i^2}\left(1 - \sum_{i=1}^{\rho} y'_i z_i^2\right) = y'_\rho + \lambda > 0$$

- For $j < \rho$:

$$y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2}\left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) = \frac{1}{\sum_{i=1}^{j} z_i^2}\left(y'_j \sum_{i=1}^{j} z_i^2 + 1 - \sum_{i=1}^{j} y'_i z_i^2\right) =$$

$$= \frac{1}{\sum_{i=1}^{j} z_i^2}\left(y'_j \sum_{i=1}^{j} z_i^2 + \sum_{i=j+1}^{\rho} y'_i z_i^2 + \underbrace{1 - \sum_{i=1}^{\rho} y'_i z_i^2}_{\lambda \sum_{i=1}^{\rho} z_i^2}\right) =$$

$$= \frac{1}{\sum_{i=1}^{j} z_i^2}\left(\sum_{i=1}^{j} z_i^2 \underbrace{(y'_j + \lambda)}_{>0} + \sum_{i=j+1}^{\rho} \underbrace{(y'_i + \lambda)}_{>0} z_i^2\right) > 0$$

  Note that for $j \leq \rho$ is $y'_j + \lambda > 0$, therefore $y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2}\left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) > 0,\ \forall j < \rho$.

- For $j > \rho$:

$$y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2}\left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) = \frac{1}{\sum_{i=1}^{j} z_i^2}\left(y'_j \sum_{i=1}^{j} z_i^2 + 1 - \sum_{i=1}^{j} y'_i z_i^2\right) =$$

$$= \frac{1}{\sum_{i=1}^{j} z_i^2}\left(y'_j \sum_{i=1}^{\rho} z_i^2 + y'_j \sum_{i=\rho+1}^{j} z_i^2 + \underbrace{1 - \sum_{i=1}^{\rho} y'_i z_i^2}_{\lambda \sum_{i=1}^{\rho} z_i^2} - \sum_{i=\rho+1}^{j} y'_i z_i^2\right) =$$

$$= \frac{1}{\sum_{i=1}^{j} z_i^2}\left(\underbrace{(y'_j + \lambda)}_{<0} \sum_{i=1}^{j} z_i^2 + \sum_{i=1}^{j} \underbrace{(y'_j - y'_i)}_{<0} z_i^2\right) < 0$$

  Note that because of ordering $y'_j \leq y'_i \ \rightarrow\ y'_j - y'_i \leq 0$ holds, hence $y'_j + \frac{1}{\sum_{i=1}^{j} z_i^2}\left(1 - \sum_{i=1}^{j} y'_i z_i^2\right) < 0,\ \forall j > \rho$.

□

Chapter 4

Experiments

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

— *Albert Einstein*

In this section, we compare the existing and proposed methods for prior shift adaptation on existing long-tailed versions of standard image classification datasets: the CIFAR100-LT [49], Places365-LT [50] and ImageNet-LT [50] (see Figure 4.1 for corresponding class distributions on the training sets). Unlike Cao et al. [49], our experiments require a validation set. Therefore, our training set, denoted as CIFAR100-LT*, is smaller than the original CIFAR100-LT, keeping 50 samples from each class for the validation set. Using the same script[1] as Cao et al. [49] to sample the training set, the resulting imbalance ratio of 112.5 slightly differs from the original ratio of 100. For Places365-LT and ImageNet-LT, we use the same training and validation splits as Liu et al. [50]. Networks trained on these long-tailed datasets are then evaluated on uniformly distributed test sets (UNI). We also provide experiments in the other direction, denoted as UNI→LT, where the networks trained on the full CIFAR100 and Places365 datasets are evaluated on test sets subsampled from the full test sets following the prior distributions of CIFAR100-LT* and Places365-LT.

The classifiers are evaluated on each dataset twice, without calibration and with calibration computed on the validation set. We choose BCTS [47] as the calibration method since it is to the best of our knowledge the best performing method in combination with label shift adaptation.

[1] `https://github.com/kaidic/LDAM-DRW`

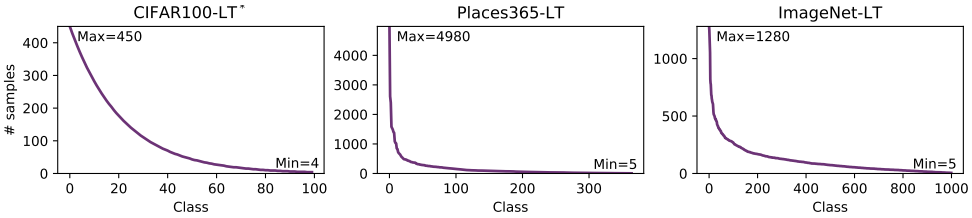


Figure 4.1: Long-tailed class distributions used in the CIFAR100-LT* [49], Places365-LT [50] and ImageNet-LT [50] datasets. Note that our CIFAR-100-LT* slightly differs from the original CIFAR-100-LT [49], which did not have a validation set.

4.1 Implementation Details

4.1.1 Settings for Classifier Training

In the experiments conducted on ImageNet and Places365, we used the ResNet-18 [51] classifier architecture. On ImageNet, the networks were trained from scratch for 90 epochs with the Stochastic Gradient Descent (SGD), initial learning rate set to 0.1 and decaying every 30 epochs by a factor of 10. On Places365, the network was finetuned from an ImageNet pre-trained checkpoint for 30 epochs with the initial learning rate set to 0.01 and decaying by a factor of 10 every 10 epochs. For experiments on CIFAR100, we trained a ResNet-32 [52] adjusted to image input of 32x32. The network was trained from scratch for 200 epochs, with a learning rate of 0.1 and decaying every 80 epochs by a factor of 10. In all experiments, the batch size was set to 256. Momentum and weight decay were set to 0.9 and 0.0001 respectively.

4.1.2 Implementation of Label Shift Adaptation

This section describes the setting of the hyperparameters in methods for label shift adaptation. The proposed methods (S)CM^L and (S)CM^W take a single hyperparameter - learning rate in gradient ascent. Since both methods have well-defined objectives, we can find the learning rate by solving the task for several different rates and then select the one maximizing objective given by the Equation (3.5) and (3.12) for (S)CM^L and (S)CM^W respectively. In the same way, we select the learning rate for (S)CM^{LR}, but the objective is given by the Equation (3.9). The regularization constant is selected manually as

$\lambda = 0.001$ and $\lambda = 1$. Note that λ cannot be determined by cross-validation because in general, the validation set has different label distribution from the test set. (S)CM^M also has two parameters: learning rate and α - parameter of symmetric Dirichlet distribution. We search only for the learning rate by maximizing objective given by the Equation (3.8) and set $\alpha = 3$, following [17]. In all proposed methods the gradient ascent runs for 1000 iterations. The results for RLLS are based on the authors' code[2], using the *cvxpy* optimizer. As termination condition for the EM and MAP algorithms, we set a threshold to l_2-distance between two consecutive solutions. The threshold is set to 0.001.

4.2 Label Shift Adaptation

4.2.1 Re-weight or Retrain?

Recall from Section 2.4 that we have basically two options how to deal with the label shift if we know the test priors $p_{\mathcal{E}}(Y)$ or priors ratio $w(Y)$:

1. Adapting the predictions of a previously trained classifier $f_{\mathcal{T}}(\mathbf{x}) \approx p_{\mathcal{T}}(Y|\mathbf{x})$, according to Equation 2.14.
2. Training a classifier $f_{\mathcal{E}}(\mathbf{x})$ with a sampler following the known new class priors $p_{\mathcal{E}}$.

When we choose a method estimating the test priors $p_{\mathcal{E}}(Y)$ rather than the priors ratio, the training priors can be determined either as a proportion of class labels in the training set, i.e. $\hat{p}^M_{\mathcal{T}}(Y = k) = \frac{M_k}{M}$, or as an average of predictions $f(\mathbf{x})$ on the training set, $\hat{p}^f_{\mathcal{T}}(Y) = \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x_i})$. The intuition behind the latter one is that the classifier has learned some inner trained prior, possibly different from the training set prior, due to some unknown phenomenon at the training stage.

The retraining and re-weighting strategies are experimentally compared in the Table 4.1 on datasets CIFAR100-LT*, Places365-LT and ImageNet-LT. The results show that re-weighting the classifier by the ratio of test and training priors outperforms retraining in all of the cases. In most of the cases, re-weighting with the training priors estimated by averaging predictions on

[2] `https://github.com/Angie-Liu/labelshift/tree/5bbe517938f4e3f5bd14c2c105de973dcc2e0917`

D	C	Standard training sampler NA	Standard training sampler $\frac{p_{\mathcal{E}}}{\hat{p}^{M}_{\mathcal{T}}}$	Standard training sampler $\frac{p_{\mathcal{E}}}{\hat{p}^{f}_{\mathcal{T}}}$	Sampler follows $p_{\mathcal{E}}$ NA
CIFAR100*					
LT→UNI	✗	$31.66^{\pm 1.27}$	$33.99^{\pm 1.41}$	$\mathbf{34.06}^{\pm \mathbf{1.35}}$	22.44
LT→UNI	✓	$31.71^{\pm 1.29}$	$30.41^{\pm 1.57}$	$\mathbf{34.54}^{\pm \mathbf{1.32}}$	22.44
UNI→LT	✗	$63.83^{\pm 0.82}$	$69.14^{\pm 0.61}$	$\mathbf{69.13}^{\pm 0.58}$	67.34
UNI→LT	✓	$63.83^{\pm 0.82}$	$70.63^{\pm 0.75}$	$\mathbf{70.65}^{\pm \mathbf{0.75}}$	67.34
Places365					
LT→UNI	✗	$25.14^{\pm 0.14}$	$28.03^{\pm 6.09}$	$\mathbf{32.99}^{\pm \mathbf{0.46}}$	24.98
LT→UNI	✓	$25.16^{\pm 0.14}$	$27.36^{\pm 6.18}$	$\mathbf{33.51}^{\pm \mathbf{0.25}}$	25.00
ImageNet					
LT→UNI	✗	$34.30^{\pm 0.19}$	$\mathbf{37.36}^{\pm \mathbf{0.07}}$	$37.34^{\pm 0.15}$	30.01
LT→UNI	✓	$34.31^{\pm 0.19}$	$36.07^{\pm 0.30}$	$\mathbf{37.45}^{\pm \mathbf{0.19}}$	30.01

Table 4.1: *"Adapt or Re-Train?"* Accuracy (± std. dev.) of classifiers adapted to new known priors $p_{\mathcal{E}}(Y)$ with different estimates of trained priors ($\hat{p}^{M}_{\mathcal{T}}$,$\hat{p}^{f}_{\mathcal{T}}$), compared to training a classifier with a sampling strategy following $p_{\mathcal{E}}(Y)$. NA denotes no adaptation of predictions. Results of classifier adaptation on CIFAR are averaged from 10 experiments, on Places365 and ImagNet from 5 experiments respectively. Re-training the classifier with a sampler following $p_{\mathcal{E}}(Y)$ was only experimented once for each dataset. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

a training set performs better than label proportions. Hence, we will use $\hat{p}_{\mathcal{T}}(Y) = \hat{p}^{f}_{\mathcal{T}}(Y)$ in all of our following experiments.

4.2.2 Improving Estimates from Confusion Matrix

Table 4.2 compares accuracy after adaptation with new prior estimate based on confusion matrix (CM) inversion [16] and our proposed method from Section 3.1 (CM^{L}) . The proposed method handles inconsistent estimates $\hat{p}(D)$ and $\hat{\mathbf{C}}_{d|y}$ and consistently improves the results both using the confusion matrix (CM^{L}) and the soft confusion matrix (SCM^{L}). In all cases, the proposed SCM^{L} method using soft confusion matrix achieves the best results.

D	C	NA	CM	CM^L	SCM	SCM^L	Oracle
CIFAR100*							
LT→UNI	✗	$31.66^{\pm 1.27}$	$21.37^{\pm 2.68}$	$\mathbf{33.00}^{\pm \mathbf{1.56}}$	$26.64^{\pm 3.85}$	$\mathbf{33.47}^{\pm \mathbf{1.33}}$	$34.06^{\pm 1.35}$
LT→UNI	✓	$31.67^{\pm 1.27}$	$19.11^{\pm 2.98}$	$\mathbf{32.41}^{\pm \mathbf{1.50}}$	$26.98^{\pm 3.76}$	$\mathbf{33.42}^{\pm \mathbf{1.51}}$	$34.40^{\pm 1.38}$
UNI→LT	✗	$63.83^{\pm 0.82}$	$68.06^{\pm 0.92}$	$\mathbf{68.08}^{\pm \mathbf{0.75}}$	$68.10^{\pm 0.81}$	$\mathbf{68.24}^{\pm \mathbf{0.75}}$	$69.13^{\pm 0.58}$
UNI→LT	✓	$63.83^{\pm 0.82}$	$69.08^{\pm 0.94}$	$\mathbf{69.10}^{\pm \mathbf{0.98}}$	$69.31^{\pm 0.94}$	$\mathbf{69.40}^{\pm \mathbf{0.77}}$	$70.65^{\pm 0.75}$
Places365							
LT→UNI	✗	$25.14^{\pm 0.14}$	$17.45^{\pm 0.30}$	$\mathbf{27.77}^{\pm \mathbf{0.45}}$	$19.78^{\pm 2.21}$	$\mathbf{28.47}^{\pm \mathbf{0.14}}$	$32.99^{\pm 0.46}$
LT→UNI	✓	$25.14^{\pm 0.14}$	$16.24^{\pm 1.39}$	$\mathbf{27.69}^{\pm \mathbf{0.51}}$	$18.88^{\pm 1.61}$	$\mathbf{27.83}^{\pm \mathbf{0.27}}$	$33.38^{\pm 0.31}$
UNI→LT	✗	$58.17^{\pm 1.01}$	$81.16^{\pm 0.61}$	$\mathbf{81.64}^{\pm \mathbf{0.63}}$	$\mathbf{82.04}^{\pm \mathbf{0.15}}$	$\mathbf{82.04}^{\pm \mathbf{0.63}}$	$88.14^{\pm 0.27}$
UNI→LT	✓	$58.17^{\pm 1.01}$	$81.20^{\pm 0.61}$	$\mathbf{81.65}^{\pm \mathbf{0.61}}$	$82.04^{\pm 0.15}$	$\mathbf{82.07}^{\pm \mathbf{0.66}}$	$88.15^{\pm 0.30}$
ImageNet							
LT→UNI	✗	$34.30^{\pm 0.19}$	$19.02^{\pm 0.26}$	$\mathbf{33.57}^{\pm \mathbf{0.33}}$	$23.94^{\pm 2.04}$	$\mathbf{35.91}^{\pm \mathbf{0.20}}$	$37.34^{\pm 0.15}$
LT→UNI	✓	$34.30^{\pm 0.19}$	$17.28^{\pm 0.48}$	$\mathbf{32.34}^{\pm \mathbf{0.41}}$	$24.78^{\pm 3.06}$	$\mathbf{35.86}^{\pm \mathbf{0.17}}$	$37.39^{\pm 0.16}$

Table 4.2: *"Improve Estimates from Confusion Matrix."* Accuracy (±std. dev.) after adaptation with new prior estimate based on confusion matrix (CM) inversion [16] and our proposed method from Section 3.1 (CM^L) and Section 3.4 (CM^W). SCM denotes soft confusion matrix, NA denotes no adaptation, Oracle is an adaptation with ground truth priors. Results on CIFAR are averaged from 10 experiments, results on Places and ImageNet are averaged from 5 experiments. The best results are displayed in bold. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

4.2.3 Comparing Regularization

Comparison of (S)CM^L method with its regularized version (S)CM^{LR} is in Table 4.3. We show the results for (S)CM^{LR} with two different regularization constants $\lambda = 0.001$ and $\lambda = 1$. For $\lambda = 0.001$ the effect of regularization is very small and the results are very close to (S)CM^L. With $\lambda = 1$ the results start to be deteriorated in the case of UNI→LT shift. We evaluated the regularization also for bigger regularization constants and with increasing λ the results are getting even more deteriorated. This is due to the stronger effect of regularization, where the estimated test prior gets closer to the training prior. Note that the setting of our experiments disadvantage the methods with regularization because the purpose of regularization is to make the estimation of the test prior more stable when the label shift is small, while the shifts between the training and test distributions in our experiments are large. In several cases (S)CM^{LR} outperforms (S)CM^L by a small margin. We conjecture this is because the likelihood close to the optimum doesn't have to correspond to the best accuracy due to an inaccurate estimate of the

D	C	NA	CM^{L}	$CM^{LR}_{\lambda=0.001}$	$CM^{LR}_{\lambda=1}$	SCM^{L}	$SCM^{LR}_{\lambda=0.001}$	$SCM^{LR}_{\lambda=1}$	Oracle
CIFAR100*									
LT→UNI	✗	$31.66^{\pm1.27}$	$33.00^{\pm1.56}$	$33.06^{\pm1.52}$	$\mathbf{33.14^{\pm1.50}}$	$33.47^{\pm1.33}$	$\mathbf{33.48^{\pm1.32}}$	$33.34^{\pm1.41}$	$34.06^{\pm1.35}$
LT→UNI	✓	$31.67^{\pm1.27}$	$32.41^{\pm1.50}$	$32.42^{\pm1.50}$	$\mathbf{33.34^{\pm1.47}}$	$33.42^{\pm1.51}$	$33.38^{\pm1.48}$	$\mathbf{33.48^{\pm1.50}}$	$34.40^{\pm1.38}$
UNI→LT	✗	$63.83^{\pm0.82}$	$\mathbf{68.08^{\pm0.75}}$	$68.04^{\pm0.76}$	$67.86^{\pm0.85}$	$\mathbf{68.24^{\pm0.75}}$	$68.24^{\pm0.72}$	$67.91^{\pm0.69}$	$69.13^{\pm0.58}$
UNI→LT	✓	$63.83^{\pm0.82}$	$\mathbf{69.10^{\pm0.98}}$	$69.07^{\pm1.00}$	$69.02^{\pm0.85}$	$\mathbf{69.40^{\pm0.77}}$	$69.39^{\pm0.79}$	$69.19^{\pm0.80}$	$70.65^{\pm0.75}$
Places365									
LT→UNI	✗	$25.14^{\pm0.14}$	$\mathbf{27.77^{\pm0.45}}$	$27.77^{\pm0.45}$	$27.69^{\pm0.46}$	$28.47^{\pm0.14}$	$28.54^{\pm0.10}$	$\mathbf{28.55^{\pm0.15}}$	$32.99^{\pm0.46}$
LT→UNI	✓	$25.14^{\pm0.14}$	$\mathbf{27.69^{\pm0.51}}$	$27.69^{\pm0.51}$	$27.69^{\pm0.50}$	$27.83^{\pm0.27}$	$\mathbf{27.85^{\pm0.26}}$	$27.82^{\pm0.25}$	$33.38^{\pm0.31}$
UNI→LT	✗	$58.17^{\pm1.01}$	$81.64^{\pm0.63}$	$\mathbf{81.67^{\pm0.66}}$	$80.29^{\pm0.67}$	$82.04^{\pm0.63}$	$\mathbf{82.09^{\pm0.64}}$	$80.55^{\pm1.15}$	$88.14^{\pm0.27}$
UNI→LT	✓	$58.17^{\pm1.01}$	$81.65^{\pm0.61}$	$\mathbf{81.72^{\pm0.67}}$	$80.30^{\pm0.69}$	$82.07^{\pm0.66}$	$\mathbf{82.10^{\pm0.64}}$	$80.53^{\pm1.12}$	$88.15^{\pm0.30}$
ImageNet									
LT→UNI	✗	$34.30^{\pm0.19}$	$33.57^{\pm0.33}$	$33.57^{\pm0.33}$	$\mathbf{33.89^{\pm0.31}}$	$\mathbf{35.91^{\pm0.20}}$	$35.78^{\pm0.17}$	$35.75^{\pm0.26}$	$37.34^{\pm0.15}$
LT→UNI	✓	$34.30^{\pm0.19}$	$32.34^{\pm0.41}$	$32.34^{\pm0.41}$	$\mathbf{32.74^{\pm0.42}}$	$\mathbf{35.86^{\pm0.17}}$	$35.59^{\pm0.14}$	$35.63^{\pm0.15}$	$37.39^{\pm0.16}$

Table 4.3: *"Compare Regularization."* Accuracy (±std. dev.) after adaptation by proposed method from Section 3.1 ((S)CM^{L}) and method proposed in Section 3.3 ((S)CM^{LR}) making use of regularization. The (S)CM^{LR} methods is evaluated with two different regularization constants $\lambda = 0.001$ and $\lambda = 1$. NA denotes no adaptation, Oracle is adaptation with ground truth priors. Results on CIFAR are averaged from 10 experiments, results on Places and ImageNet are averaged from 5 experiments. The best results are displayed in bold. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

D	C	NA	$CM^{LR}_{\lambda=0.001}$	$CM^{LR}_{\lambda=1}$	$SCM^{LR}_{\lambda=0.001}$	$SCM^{LR}_{\lambda=1}$	RLLS	Oracle
CIFAR100*								
LT→UNI	✗	$31.66^{\pm1.27}$	$33.06^{\pm1.52}$	$33.14^{\pm1.50}$	$\mathbf{33.48^{\pm1.32}}$	$33.34^{\pm1.41}$	$32.75^{\pm1.40}$	$34.06^{\pm1.35}$
LT→UNI	✓	$31.67^{\pm1.27}$	$32.42^{\pm1.50}$	$33.34^{\pm1.47}$	$33.38^{\pm1.48}$	$\mathbf{33.48^{\pm1.50}}$	$32.62^{\pm1.46}$	$34.40^{\pm1.38}$
UNI→LT	✗	$63.83^{\pm0.82}$	$68.04^{\pm0.76}$	$67.86^{\pm0.85}$	$\mathbf{68.24^{\pm0.72}}$	$67.91^{\pm0.69}$	$68.02^{\pm0.77}$	$69.13^{\pm0.58}$
UNI→LT	✓	$63.83^{\pm0.82}$	$69.07^{\pm1.00}$	$69.02^{\pm0.85}$	$\mathbf{69.39^{\pm0.79}}$	$69.19^{\pm0.80}$	$69.05^{\pm0.97}$	$70.65^{\pm0.75}$
Places365								
LT→UNI	✗	$25.14^{\pm0.14}$	$27.77^{\pm0.45}$	$27.69^{\pm0.46}$	$28.54^{\pm0.10}$	$\mathbf{28.55^{\pm0.15}}$	$26.94^{\pm0.41}$	$32.99^{\pm0.46}$
LT→UNI	✓	$25.14^{\pm0.14}$	$27.69^{\pm0.51}$	$27.69^{\pm0.50}$	$\mathbf{27.85^{\pm0.26}}$	$27.82^{\pm0.25}$	$27.03^{\pm0.40}$	$33.38^{\pm0.31}$
UNI→LT	✗	$58.17^{\pm1.01}$	$81.67^{\pm0.66}$	$80.29^{\pm0.67}$	$\mathbf{82.09^{\pm0.64}}$	$80.55^{\pm1.15}$	$82.04^{\pm0.69}$	$88.14^{\pm0.27}$
UNI→LT	✓	$58.17^{\pm1.01}$	$81.72^{\pm0.67}$	$80.30^{\pm0.69}$	$\mathbf{82.10^{\pm0.64}}$	$80.53^{\pm1.12}$	$82.04^{\pm0.69}$	$88.15^{\pm0.30}$
ImageNet								
LT→UNI	✗	$34.30^{\pm0.19}$	$33.57^{\pm0.33}$	$33.89^{\pm0.31}$	$\mathbf{35.78^{\pm0.17}}$	$35.75^{\pm0.26}$	$34.69^{\pm0.14}$	$37.34^{\pm0.15}$
LT→UNI	✓	$34.30^{\pm0.19}$	$32.34^{\pm0.41}$	$32.74^{\pm0.42}$	$35.59^{\pm0.14}$	$\mathbf{35.63^{\pm0.15}}$	$34.31^{\pm0.08}$	$37.39^{\pm0.16}$

Table 4.4: *"Compare regularization."* Accuracy (±std. dev.) after adaptation with methods based on regularization, i.e. RLLS [19] and method proposed in Section 3.3 (S)CM^{LR}. NA denotes no adaptation, Oracle is adaptation with ground truth priors. The (S)CM^{LR} methods is evaluated with two different regularization constants $\lambda = 0.001$ and $\lambda = 1$. Results on CIFAR are averaged from 10 experiments, results on Places and ImageNet are averaged from 5 experiments. The best results are displayed in bold. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

confusion matrix.

The methods leveraging regularization to prevent large shifts from training prior $p_{\mathcal{T}}(Y)$ are compared in Table 4.4. Concretely, we compare RLLS [19], which is estimating priors ratio $w(Y)$ by minimizing l_2-norm with regularization term penalizing large shifts, against our method proposed in Section 3.3 with two different regularization constants $\lambda = 0.001$ and $\lambda = 1$. The best performing method is SCM^{LR} with $\lambda = 0.001$.

4.2.4 Comparing Ratio Estimation

Having an estimate of the trained priors, Table 4.5 compares prior ratio estimation with BBSE, BBSE-S [20] and RLLS [19] against the best performing prior estimation method, SCM^{L}, and our methods (CM^{W} and SCM^{W}) proposed to handle the negative weights in the estimates from BBSE and BBSE-S. The improvement of our method (S)CM^{W} compared to BBSE and BBSE-S is apparent on datasets CIFAR100 and ImageNet, while on Places365 the result is deteriorated. The results indicate that it is better to estimate the new priors with SCM^{L} than to directly estimate the prior ratio.

4.2.5 Methods for MLE and MAP Prior Estimation

Existing methods for maximum likelihood and maximum a-posteriori prior estimation are compared against the methods proposed in Sections 3.1 and 3.2 respectively in Table 4.6. Note that the methods maximize a different likelihood function: The EM algorithm of Saerens et al. [16] maximizes the likelihood of observed classifier outputs $f(\mathbf{x}_i)$, while the proposed methods based on the confusion matrix (CM^{L}) and the soft confusion matrix (SCM^{L}) maximize the likelihood of classifiers decisions. The same difference in likelihood functions holds for the MAP approach of Sulc and Matas [17] and MAP estimate proposed in Section 3.2, but we use the same hyper-prior on $p_{\mathcal{E}}(Y)$ for all methods: $Dir(\alpha = 3)$. Although the (S)CM^{W} is also based on maximum likelihood estimation, we do not compare this method here since it is outperformed by (S)CM^{L} as experimentally shown in Section 4.2.4.

From the maximum likelihood estimators, the proposed SCM^{L} achieves the best results in most cases, except for Places365 "UNI→LT", where the EM algorithm performed slightly better. Similarly, the Maximum A-Posteriori version of the proposed method, SCM^{M} performs better than the existing MAP estimate [17] in most cases. As expected, the MAP estimation improves

D	C	NA	SCM^L	CM^W	SCM^W	RLLS	BBSE	BBSE-S	Oracle
CIFAR100*									
LT→UNI	✗	$31.66^{\pm 1.27}$	$\mathbf{33.47}^{\pm \mathbf{1.33}}$	$\underline{32.77}^{\pm 1.40}$	$32.75^{\pm 1.28}$	$32.75^{\pm 1.40}$	$31.28^{\pm 1.58}$	$31.92^{\pm 1.62}$	$34.06^{\pm 1.35}$
LT→UNI	✓	$31.67^{\pm 1.27}$	$\mathbf{33.42}^{\pm \mathbf{1.51}}$	$\underline{33.11}^{\pm 1.29}$	$32.98^{\pm 1.35}$	$32.62^{\pm 1.46}$	$26.47^{\pm 1.87}$	$29.06^{\pm 2.65}$	$34.40^{\pm 1.38}$
UNI→LT	✗	$63.83^{\pm 0.82}$	$\mathbf{68.24}^{\pm \mathbf{0.75}}$	$68.04^{\pm 0.76}$	$\underline{68.24}^{\pm 0.72}$	$68.02^{\pm 0.77}$	$67.95^{\pm 0.96}$	$68.12^{\pm 0.90}$	$69.13^{\pm 0.58}$
UNI→LT	✓	$63.83^{\pm 0.82}$	$69.40^{\pm 0.77}$	$69.08^{\pm 0.97}$	$69.41^{\pm 0.80}$	$69.05^{\pm 0.97}$	$69.30^{\pm 0.99}$	$\underline{\mathbf{69.51}}^{\pm \mathbf{0.98}}$	$70.65^{\pm 0.75}$
Places365									
LT→UNI	✗	$25.14^{\pm 0.14}$	$\mathbf{28.47}^{\pm \mathbf{0.14}}$	$23.14^{\pm 2.72}$	$19.75^{\pm 2.99}$	$\underline{26.94}^{\pm 0.41}$	$24.79^{\pm 0.74}$	$25.55^{\pm \pm 0.69}$	$32.99^{\pm 0.46}$
LT→UNI	✓	$25.14^{\pm 0.14}$	$\mathbf{27.83}^{\pm \mathbf{0.27}}$	$23.16^{\pm 3.59}$	$18.80^{\pm 1.32}$	$\underline{27.03}^{\pm 0.40}$	$23.12^{\pm 0.79}$	$23.68^{\pm 0.75}$	$33.38^{\pm 0.31}$
UNI→LT	✗	$58.17^{\pm 1.01}$	$\mathbf{82.04}^{\pm \mathbf{0.63}}$	$81.49^{\pm 0.51}$	$81.57^{\pm 0.32}$	$\underline{82.04}^{\pm 0.69}$	$80.66^{\pm 0.57}$	$81.69^{\pm 0.20}$	$88.14^{\pm 0.27}$
UNI→LT	✓	$58.17^{\pm 1.01}$	$\mathbf{82.07}^{\pm \mathbf{0.66}}$	$81.52^{\pm 0.57}$	$81.64^{\pm 0.34}$	$\underline{82.04}^{\pm 0.69}$	$80.71^{\pm 0.56}$	$81.69^{\pm 0.20}$	$88.15^{\pm 0.30}$
ImageNet									
LT→UNI	✗	$34.30^{\pm 0.19}$	$\mathbf{35.91}^{\pm \mathbf{0.20}}$	$35.21^{\pm 0.13}$	$\underline{35.28}^{\pm 0.15}$	$34.69^{\pm 0.14}$	$30.77^{\pm 0.31}$	$31.31^{\pm 0.90}$	$37.34^{\pm 0.15}$
LT→UNI	✓	$34.30^{\pm 0.19}$	$\mathbf{35.86}^{\pm \mathbf{0.17}}$	$34.75^{\pm 0.06}$	$\underline{35.00}^{\pm 0.17}$	$34.31^{\pm 0.08}$	$26.89^{\pm 0.49}$	$28.05^{\pm 1.69}$	$37.39^{\pm 0.16}$

Table 4.5: "*Estimate test priors or directly the prior ratio?*" Accuracy (± std. dev.) after adaptation with the priors estimated by SCM^L or with the prior ratio estimated by BBSE [20] (without re-training), RLLS [19] (without re-training), CM^W and SCM^W. Results on CIFAR are averaged from 10 experiments, results on Places and ImageNet are averaged from 5 experiments. The best results among all methods are displayed in bold. The best results among methods estimating priors ratio are underlined. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

D	C	NA	CM^L	SCM^L	EM	CM^M	SCM^M	MAP	Oracle
CIFAR100*									
LT→UNI	✗	$31.66^{\pm 1.27}$	$33.00^{\pm 1.56}$	$\mathbf{33.47}^{\pm \mathbf{1.33}}$	$32.81^{\pm 1.41}$	$33.49^{\pm 1.45}$	$\mathbf{33.50}^{\pm \mathbf{1.40}}$	$32.73^{\pm 1.42}$	$34.06^{\pm 1.35}$
LT→UNI	✓	$31.67^{\pm 1.27}$	$32.41^{\pm 1.50}$	$\mathbf{33.42}^{\pm \mathbf{1.51}}$	$29.43^{\pm 1.59}$	$33.99^{\pm 1.43}$	$\mathbf{34.13}^{\pm \mathbf{1.53}}$	$24.46^{\pm 12.43}$	$34.40^{\pm 1.38}$
UNI→LT	✗	$63.83^{\pm 0.82}$	$68.08^{\pm 0.75}$	$\mathbf{68.24}^{\pm \mathbf{0.75}}$	$67.23^{\pm 0.88}$	$\mathbf{67.01}^{\pm \mathbf{0.91}}$	$67.00^{\pm 0.87}$	$66.72^{\pm 0.91}$	$69.13^{\pm 0.58}$
UNI→LT	✓	$63.83^{\pm 0.82}$	$69.10^{\pm 0.98}$	$\mathbf{69.40}^{\pm \mathbf{0.77}}$	$69.17^{\pm 0.91}$	$\mathbf{68.42}^{\pm \mathbf{0.84}}$	$68.38^{\pm 0.73}$	$68.30^{\pm 0.77}$	$70.65^{\pm 0.75}$
Places365									
LT→UNI	✗	$25.14^{\pm 0.14}$	$27.77^{\pm 0.45}$	$\mathbf{28.47}^{\pm \mathbf{0.14}}$	$28.02^{\pm 0.92}$	$\mathbf{28.02}^{\pm \mathbf{0.24}}$	$27.68^{\pm 0.13}$	$25.22^{\pm 0.13}$	$32.99^{\pm 0.46}$
LT→UNI	✓	$25.14^{\pm 0.14}$	$27.69^{\pm 0.51}$	$27.83^{\pm 0.27}$	$\mathbf{28.09}^{\pm \mathbf{1.32}}$	$27.92^{\pm 0.24}$	$27.41^{\pm 0.15}$	$\mathbf{28.57}^{\pm \mathbf{0.24}}$	$33.38^{\pm 0.31}$
UNI→LT	✗	$58.17^{\pm 1.01}$	$81.64^{\pm 0.63}$	$82.04^{\pm 0.63}$	$\mathbf{82.63}^{\pm \mathbf{0.31}}$	$76.13^{\pm 0.56}$	$73.27^{\pm 0.46}$	$\mathbf{76.97}^{\pm \mathbf{0.45}}$	$88.14^{\pm 0.27}$
UNI→LT	✓	$58.17^{\pm 1.01}$	$81.65^{\pm 0.61}$	$82.07^{\pm 0.66}$	$\mathbf{82.63}^{\pm \mathbf{0.26}}$	$76.16^{\pm 0.54}$	$73.30^{\pm 0.46}$	$\mathbf{77.00}^{\pm 0.41}$	$88.15^{\pm 0.30}$
ImageNet									
LT→UNI	✗	$34.30^{\pm 0.19}$	$33.57^{\pm 0.33}$	$\mathbf{35.91}^{\pm \mathbf{0.20}}$	$34.63^{\pm 0.29}$	$36.41^{\pm 0.17}$	$\mathbf{36.57}^{\pm \mathbf{0.16}}$	$34.64^{\pm 0.20}$	$37.34^{\pm 0.15}$
LT→UNI	✓	$34.30^{\pm 0.19}$	$32.34^{\pm 0.41}$	$\mathbf{35.86}^{\pm \mathbf{0.17}}$	$27.26^{\pm 2.25}$	$36.18^{\pm 0.12}$	$\mathbf{36.80}^{\pm \mathbf{0.14}}$	$20.65^{\pm 18.77}$	$37.39^{\pm 0.16}$

Table 4.6: *"How to estimate new priors?"* Accuracy (± std. dev.) after adaptation to new priors estimated with different Maximum Likelihood and Maximum A Posteriori estimates. NA denotes no adaptation, Oracle is adaptation with ground truth priors. Results on CIFAR are averaged from 10 experiments, results on Places and ImageNet are averaged from 5 experiments. The best results are displayed in bold. The column denoted as D determines the training and the test distribution for a particular row, while the column denoted as C denotes if BCTS [47] calibration was applied.

upon MLE on the dense test distributions, favoured by the Dirichlet hyper-prior.

4.2.6 Dependence on the Number of Test Samples

Figure 4.2 displays the accuracy on the uniformly distributed sets after the adaptation of classifiers trained on the CIFAR100-LT* and Places365-LT datasets with different prior estimation methods, as a function of the number of test examples used for prior estimation. While the proposed SCM^{L} method achieves slightly higher accuracy with more samples, the EM algorithm works slightly better with a low number of samples. With an extremely low number of samples, the prior estimation should better be omitted. For the sake of clarity, we exclude the other proposed methods ((S)CM^{LR}, (S)CM^{W}, (S)CM^{M}) from the comparison since they are just extensions to the (S)CM^{L} method or perform worse compared to the (S)CM^{L}.

4.3 Confusion Matrices Illustrated on an Artificial Dataset

Let us consider an illustrative classification problem with two classes $\{0, 1\}$, generated from known normal distributions: $p(x|Y = 0) = \mathcal{N}(-2, 2)$ and $p(x|Y = 1) = \mathcal{N}(2, 2)$ with equal priors, $p(Y = 0) = p(Y = 1) = 0.5$. We use 3 different classifiers modeled by logistic regression in the form

$$f(x) = \frac{1}{1 + e^{-(ax+b)}}. \tag{4.1}$$

The first classifier, $f_t(x)$, was trained with *scikit-learn* on 4 randomly generated samples. The other two are Bayes classifiers: $f_c(x)$ is perfectly calibrated with parameters $a = 1$ and $b = 0$; $f_o(x)$ is overconfident with parameters $a = 2$ and $b = 0$. All three classifiers are illustrated together with their decision thresholds and the known posterior probabilities in Figure 4.3.

Following the Section 2.4.1, we denote $\hat{\mathbf{C}}_{d|y}$ the confusion matrix estimated from top-1 predictions and $\hat{\mathbf{C}}^{soft}_{d|y}$ the confusion matrix estimated using the softmax outputs, following the Equation (2.19). In this artificial example with known distributions, we can compute the *true confusion matrix* $\mathbf{C}_{d|y}$ as follows

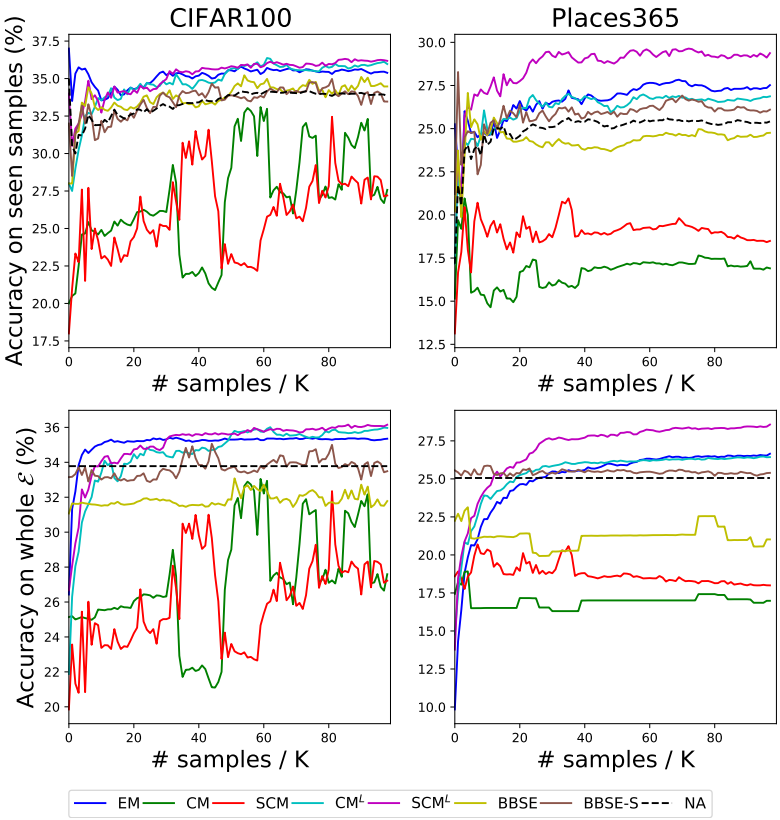


Figure 4.2: *"How many samples do I need?"* Accuracy after adapting CIFAR100-LT→UNI (left) and Places365-LT→UNI (right) using #samples for prior estimation.

$$\mathbf{C}_{1|j} = \int_t^\infty p(x|Y=j)\,dx; \quad \mathbf{C}_{0|j} = \int_{-\infty}^t p(x|Y=j)\,dx. \tag{4.2}$$

Figure 4.4 compares the distance of the two estimated confusion matrices from the true confusion matrix, depending on the size of the validation set. Note that the soft confusion matrix may not converge to the true confusion matrix even with a perfectly calibrated classifier, but it provides a better estimate in low-sample scenarios.

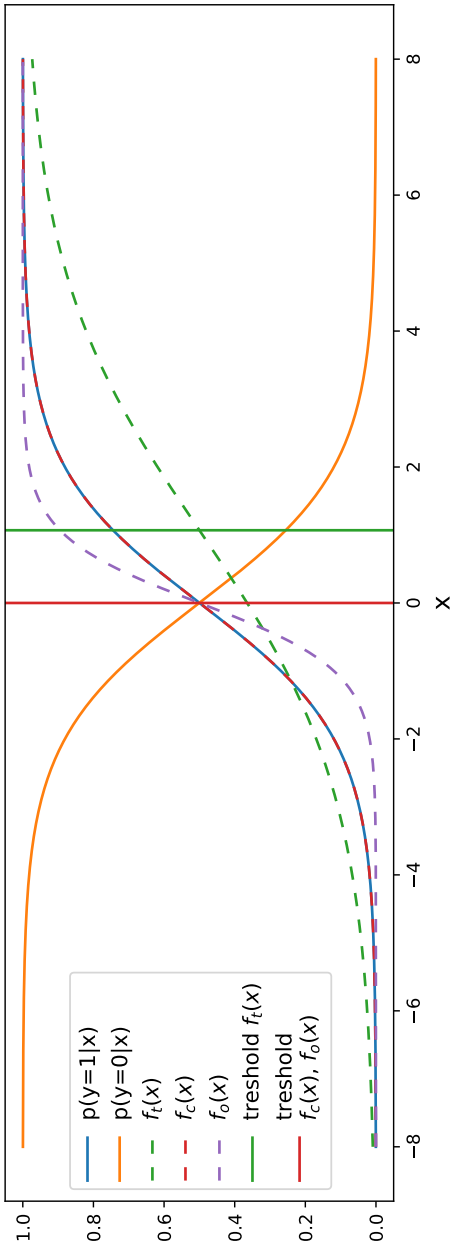
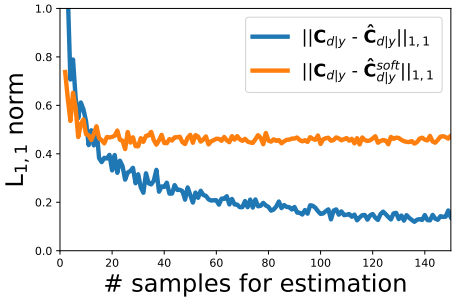


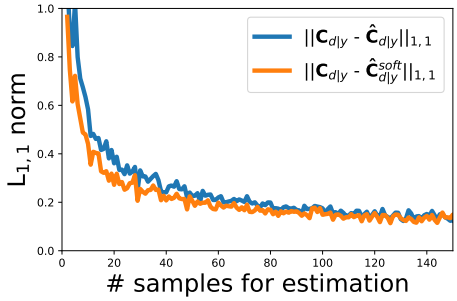
Figure 4.3: An illustrative 2-class example with known class posteriors (solid curves), outputs of 3 classifiers for given x (dashed lines) and their decision thresholds (vertical lines). Note that $f_c(x)$ and $f_o(x)$ are optimal Bayes classifiers minimizing the 0/1 loss.

(a) : Classifier $f_t(x)$ trained on 4 samples:



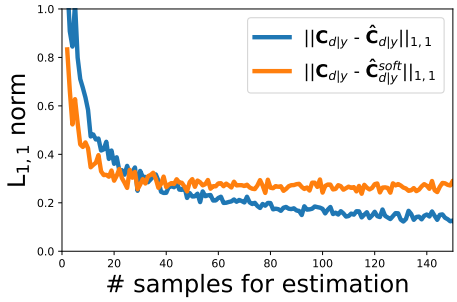
$$\underset{\hat{\mathbf{C}}_{d|y}}{\begin{bmatrix}0.94 & 0.30\\ 0.06 & 0.70\end{bmatrix}}, \underset{\hat{\mathbf{C}}^{soft}_{d|y}}{\begin{bmatrix}0.785 & 0.396\\ 0.215 & 0.604\end{bmatrix}}, \underset{\mathbf{C}_{d|y}}{\begin{bmatrix}0.938 & 0.321\\ 0.062 & 0.679\end{bmatrix}}$$

(b) : Overconfident classifier $f_o(x)$:



$$\underset{\hat{\mathbf{C}}_{d|y}}{\begin{bmatrix}0.82 & 0.16\\ 0.18 & 0.84\end{bmatrix}}, \underset{\hat{\mathbf{C}}^{soft}_{d|y}}{\begin{bmatrix}0.798 & 0.174\\ 0.202 & 0.826\end{bmatrix}}, \underset{\mathbf{C}_{d|y}}{\begin{bmatrix}0.841 & 0.159\\ 0.159 & 0.841\end{bmatrix}}$$

(c) : Calibrated classifier $f_c(x)$:



$$\underset{\hat{\mathbf{C}}_{d|y}}{\begin{bmatrix}0.82 & 0.16\\ 0.18 & 0.84\end{bmatrix}}, \underset{\hat{\mathbf{C}}^{soft}_{d|y}}{\begin{bmatrix}0.764 & 0.215\\ 0.236 & 0.785\end{bmatrix}}, \underset{\mathbf{C}_{d|y}}{\begin{bmatrix}0.841 & 0.159\\ 0.159 & 0.841\end{bmatrix}}$$

Figure 4.4: Top: The distance (sum of absolute differences) of estimated confusion matrix $\hat{\mathbf{C}}_{d|y}$ and soft confusion matrix $\hat{\mathbf{C}}^{soft}_{d|y}$ from the true confusion matrix $\mathbf{C}_{d|y}$, depending on the number of validation samples. The distance values are averaged over 50 trials. Bottom: Confusion matrices $\hat{\mathbf{C}}_{d|y}$, $\hat{\mathbf{C}}^{soft}_{d|y}$ estimated from 50 samples and the true confusion matrix $\mathbf{C}_{d|y}$ computed from Equation (4.2).

4.4 Convergence Speed

The convergence of all algorithms on the test set of Places365-LT is displayed in the Figure 4.5. For the sake of clarity, we exclude (S)CM$^{\mathrm{LR}}$, (S)CM$^{\mathrm{W}}$ methods from comparison in the Figure 4.5 since they are just extensions to the (S)CM$^{\mathrm{L}}$ method or perform worse compared to the (S)CM$^{\mathrm{L}}$. Note that the optimization code for our methods is experimental and not optimized for run time. The results for RLLS are based on the authors' code[3], using the *cvxpy* optimizer. Table 4.7 compares the runtimes of all algorithms on Cifar100-LT, Places365-LT and ImageNet-LT, using the termination conditions used in all our experiments and described in Section 4.1.2. While the proposed optimization in (S)CM^{L} and (S)CM^{M} takes longer than the baseline (S)CM, all methods converge within 0.5 seconds on Places365-LT. Even though the predictions are adapted on CPU and the classifier evaluation is computed on GPU, the time to adapt classifier to label shift is negligible compared to the time it takes to evaluate the classifier on the test set, which takes several minutes on ImageNet-LT.

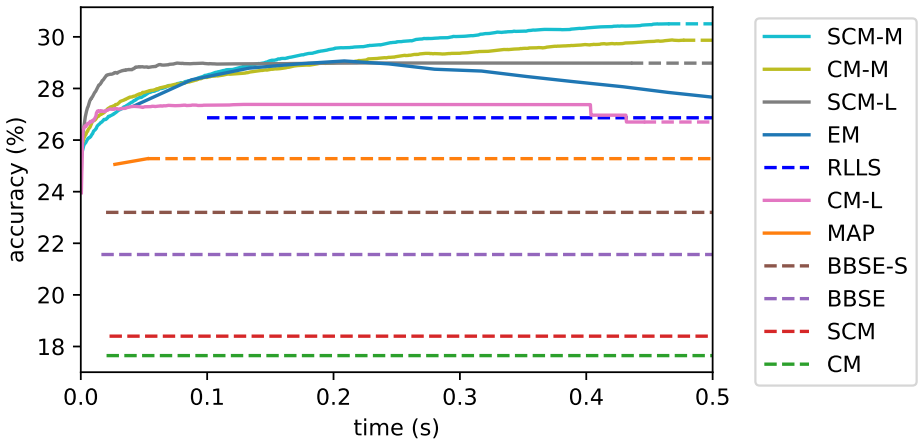


Figure 4.5: Convergence of prior estimation on Places365-LT. The y-axis shows the classification accuracy after prior shift adaptation. Bold lines depict a particular method before the termination criterion is met. Dashed lines show the constant accuracy after the computation is finished. Note that (S)CM, BBSE and BBSE-S are single-step methods. Since RLLS uses a third party optimizer, we only report the total optimization time.

[3] `https://github.com/Angie-Liu/labelshift/tree/5bbe517938f4e3f5bd14c2c105de973dcc2e0917`

	Cifar100-LT	Places365-LT	ImageNet-LT
CM	0.0038	0.0697	0.3486
SCM	0.0034	0.0555	0.2946
BBSE	0.0089	0.0678	0.3357
BBSE-S	0.0033	0.0506	0.2927
EM	0.0068	0.8293	2.2329
CM-L	0.2001	0.3104	1.3035
SCM-L	0.1828	0.2955	1.3595
CM-LR	0.2021	0.3349	1.3127
SCM-LR	0.1975	0.3197	1.3151
CM-W	0.1735	0.2985	1.2144
SCM-W	0.1665	0.2847	1.2456
MAP	0.0041	0.0916	0.3388
CM-M	0.2061	0.3355	1.3943
SCM-M	0.1998	0.3114	1.3510
RLLS	0.0268	0.1946	2.1280

Table 4.7: Run time (in seconds) of methods estimating test time prior $p_{\mathcal{E}}(Y)$ or prior ratio $p_{\mathcal{E}}(Y)/p_{\mathcal{T}}(Y)$. The run time was measured on a laptop with Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 8 and averaged over 10 runs.

Chapter 5

Conclusion

"If we knew what it was we were doing, it would not be called research, would it?"

— *Albert Einstein*

This thesis reviews and compares the existing methods for adapting classifiers to label shift and proposes new methods based on the confusion matrix, likelihood maximization and maximum a-posteriori estimation. The proposed methods handle a known problem [21, 16, 18, 20] of inconsistent estimates of decision probabilities and of the confusion matrix, which can result in negative values in the estimated prior probabilities or in the estimated prior ratio. The (S)CM^L method for test set prior estimation deals with the problem by maximization of the likelihood of classifier's decisions on the new unlabeled test set. The next method proposed in this thesis, (S)CM^M, extends (S)CM^L by adding Dirichlet hyper-prior on the test set label distribution, leading to a maximum a-posteriori formulation. The third method proposed in the thesis is (S)CM^{LR}, which extends (S)CM^L by adding a regularization term to penalize large shifts. The last proposed method, (S)CM^W, is estimating the prior ratio instead of the test set prior. As part of the (S)CM^W method, this thesis proposed an algorithm for projection onto the prior ratio simplex by generalizing an algorithm for projection onto the probability simplex described by Wang et al. [48].

From the experimental analysis of the existing and the proposed methods for prior shift adaptation we can observe and conclude the following:

- Adapting classifier predictions typically performs better than re-training the classifier with sampling matching the shift, and it is significantly

computationally cheaper.

- The proposed (S)CM$^{\mathrm{L}}$ method handles inconsistent estimates of decision probabilities $\hat{p}(D)$ and confusion matrix $\hat{\mathbf{C}}_{d|y}$, and it consistently improves the results compared to the (S)CM baseline.
- From the compared maximum likelihood estimators, the proposed SCM$^{\mathrm{L}}$ achieves the best results in most cases.
- Extending (S)CM$^{\mathrm{L}}$ with regularization, denoted (S)CM$^{\mathrm{LR}}$, does not lead to any advantage on datasets with large shifts. Investigation of the effects on datasets with smaller shifts is left for future work.
- The proposed Maximum A-Posteriori approach, SCM$^{\mathrm{M}}$, performs better than the existing MAP estimator [17].
- It is better to estimate the training and test priors separately than to directly estimate their ratio with BBSE, RLLS or the proposed (S)CM$^{\mathrm{W}}$ method.
- Among the methods estimating priors ratio, the (S)CM$^{\mathrm{W}}$ outperforms all other methods on CIFAR and ImageNet, while on Places365 the best result is achieved by RLLS.
- In a small sample scenario, the EM algorithm [16, 47] and BBSE-S perform the best among the compared methods. With an extremely low number of samples, prior estimation should better be omitted at all.
- Soft confusion matrix is a biased estimate of the classifier's confusion matrix, but for a small number of samples in the validation set it gives a better estimate than the confusion matrix computed by counting decisions $\arg\max_k f(\mathbf{x})$.
- All compared methods perform nearly real-time: The time to adapt classifier's predictions to label shift is negligible compared to the time it takes to make the predictions on the test set.
- The (S)CM baselines are faster than the proposed methods, since they compute matrix inversion in a single step, while the proposed methods are iterative. EM algorithm has very fast convergence on a dataset with a small number of classes (CIFAR100-LT), where it is also faster than the proposed methods. On the other hand, on datasets with a higher number of classes (Places365-LT and ImageNet-LT) the convergence of EM is about twice slower than the convergence of the proposed methods.

- Prior shift adaptation relies on a well-calibrated classifier, assumed in Eq. (2.14). In [47], BCTS improves prior shift adaptation of classifiers trained on uniform distribution, similarly to our UNI→LT experiments. With class-specific parameters, BCTS may overfit to errors on the validation set. For classifiers trained on LT datasets, we show BCTS is not a reliable calibration method, as it often decreases the final recognition accuracy.

Future Work. This thesis reviewed previous work on classifier calibration, but the effects of calibration on adaptation to label shift were examined only briefly. Deeper understanding of the mutual influence between calibration and label shift adaptation will be needed for the future work to improve the accuracy of calibrated classifiers after label shift.

Investigation into how regularization affects the proposed (S)CM$^{\mathrm{L}}$ method in small shift scenarios is left for future work, which will evaluate the methods on a wider range of dataset shifts. The assumption is that regularization will help when employed on smaller dataset shifts, as it prevents large deviations from the original distribution. Inspired by Azizzadenesheli et al. [19], the future work can attempt to find theoretical guarantees for our methods, which could give us better insight into how to set up the regularization constant in (S)CM$^{\mathrm{LR}}$ or the hyperparameter of Dirichlet hyperprior in (S)CM$^{\mathrm{M}}$ properly.

All methods in the thesis were solely focused on the classification task. Adaptation to dataset shifts in other tasks, such as regression, object detection or semantic segmentation, remains an interesting topic for future work.

Bibliography

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 ed., 2010.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Adaptive Computation and Machine Learning, Cambridge, MA: MIT Press, 2 ed., 2018.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2018.

[7] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2021.

[8] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2020.

[9] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," 2019.

[10] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *ICML*, 2004.

[11] M. Sugiyama *et al.*, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *NIPS*, 2008.

[12] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning under covariate shift," *Journal of Machine Learning Research*, vol. 10, no. 75, pp. 2137–2155, 2009.

[13] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), vol. 19, MIT Press, 2007.

[14] K. Zhang *et al.*, "Domain adaptation under target and conditional shift," in *ICML*, 2013.

[15] M. C. du Plessis and M. Sugiyama, "Semi-supervised learning of class balance under class-prior change by distribution matching," *CoRR*, vol. abs/1206.4677, 2012.

[16] M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure," *Neural Comput.*, vol. 14, p. 21–41, Jan. 2002.

[17] M. Sulc and J. Matas, "Improving cnn classifiers by estimating test-time priors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[18] S. Vucetic and Z. Obradovic, "Classification on data with biased class distribution," in *European Conference on Machine Learning*, pp. 527–538, Springer, 2001.

[19] K. Azizzadenesheli, A. Liu, F. Yang, and A. Anandkumar, "Regularized learning for domain adaptation under label shifts.," in *ICLR*, 2019.

[20] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," 2018.

[21] G. Forman, "Quantifying counts and costs via classification," *Data Mining and Knowledge Discovery*, vol. 17, pp. 164–206, Oct 2008.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[25] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," 2017.

[26] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," 2020.

[27] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[28] V. Tra, J. Kim, S. A. Khan, and J.-M. Kim, "Bearing fault diagnosis under variable speed using convolutional neural networks and the stochastic diagonal levenberg-marquardt algorithm," *Sensors*, vol. 17, no. 12, 2017.

[29] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *Journal of Machine Learning Research*, vol. 10, no. 48, pp. 1391–1445, 2009.

[30] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *2013 IEEE International Conference on Computer Vision*, pp. 2960–2967, 2013.

[31] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Domain adaptation on the statistical manifold," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[32] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," 2014.

[33] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, p. 677–682, AAAI Press, 2008.

[34] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proceedings of*

the 19th International Conference on World Wide Web, WWW '10, (New York, NY, USA), p. 751–760, Association for Computing Machinery, 2010.

[35] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," 2016.

[36] R. Tachet, H. Zhao, Y.-X. Wang, and G. Gordon, "Domain adaptation with conditional distribution matching and generalized label shift," 2020.

[37] Y. Li, M. Murias, S. Major, G. Dawson, and D. E. Carlson, "On target shift in adversarial domain adaptation," 2019.

[38] H. Liu, M. Long, J. Wang, and M. Jordan, "Transferable adversarial training: A general approach to adapting deep classifiers," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 4013–4022, PMLR, 09–15 Jun 2019.

[39] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," 2017.

[40] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[41] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. Hoboken, NJ, USA: John Wiley & Sons, Inc, 1992-03-27.

[42] M. Kull, M. Perelló-Nieto, M. Kängsepp, T. de Menezes e Silva Filho, H. Song, and P. A. Flach, "Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with dirichlet calibration," *CoRR*, vol. abs/1910.12656, 2019.

[43] J. Vaicenavicius, D. Widmann, C. R. Andersson, F. Lindsten, J. Roll, and T. B. Schön, "Evaluating model calibration in classification," *CoRR*, vol. abs/1902.06977, 2019.

[44] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), p. 609–616, Morgan Kaufmann Publishers Inc., 2001.

[45] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *CoRR*, vol. abs/1706.04599, 2017.

[46] A. H. Murphy and R. L. Winkler, "Reliability of Subjective Probability Forecasts of Precipitation and Temperature," *Journal of the Royal Statistical Society Series C*, vol. 26, pp. 41–47, March 1977.

[47] A. Alexandari, A. Kundaje, and A. Shrikumar, "Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation," in *ICML*, pp. 222–232, 2020.

[48] W. Wang and M. Á. Carreira-Perpiñán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," *ArXiv*, vol. abs/1309.1541, 2013.

[49] K. Cao *et al.*, "Learning imbalanced datasets with label-distribution-aware margin loss," in *NeurIPS*, 2019.

[50] Z. Liu *et al.*, "Large-scale long-tailed recognition in an open world," in *CVPR*, 2019.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[52] Y. Idelbayev, "Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch." `https://github.com/akamaster/pytorch_resnet_cifar10`. Accessed: 2021-03-23.

Appendix A

Code

A.1 Sample Code

We attach jupyter notebook with sample code to the thesis with the implementation of the proposed methods and algorithms used for comparison. Classifier's predictions precomputed on CIFAR100-LT* are provided as well for easy demonstration. Packages required to run the code can be installed via *pip* or *Conda* using `requirements.txt` or `env.yml` files. The algorithms for adaptation are implemented in separated script `algorithms.py`. The sample code contains the following files:

```
sample_code
├── env.yml
├── requirements.txt
├── example.ipynb
├── algorithms.py
└── CIFAR100_LT_outputs
    └── outputs.pth.tar
```

To run the code start jupyter notebook and run all cells.

A.2 Full Reproducibility Code

In addition to the sample code we also provide experimental code, used to generate all results in the main text. Use `.../experimental_code/requirements.txt` file to set up the python environment.

To train 10 classifiers on CIFAR100 on UNI and LT distributions run

following command (assuming current directory is the root of experimental code, i.e. `.../experimental_code/`):

```
python3 multi_train.py -c configs/train_configs/train_
config_Cifar100.json -s
```

For Places365:

```
python3 multi_train.py -c configs/train_configs/train_
config_Places365.json -s
```

And for ImageNet:

```
python3 multi_train.py -c configs/train_configs/train_
config_ImageNet.json -s
```

Note that you will need to set the path to the images of original CIFAR100, ImageNet and Places365 datasets. This is done in corresponding config file `.../experimental_code/configs/dataset_configs/{dataset}.json` by setting `"root"` tag.

To run evaluation with label shift adaptation you can run following command:

```
python3 multi_eval.py -c adaptation/eval_configs/eval.json -s
```

Before running this command you need to set paths to the saved classifier's outputs into the `eval.json` config file (or another one you used in the argument). The tag determined for this information is `"outputs"`, taking the paths as a list.

Appendix B

List of Attachments

- `sample_code.zip`
- `experimental_code.zip`