Detecting and adapting to domain shifts from output distribution in a multiclass setting

Rémy Sun[1]
remy.sun@ens-rennes.fr
and under supervision of Christoph Lampert[2]
chl@ist.ac.at

[1] ENS Rennes
Computer Science department
Campus de Ker lann, Bruz, France
[2] IST Austria
Computer vision and Machine Learning group
Am Campus 1, 3400 Klosterneuburg, Austria

Abstract. Differences between the domain on which a classifier is trained and the domain on which it is actually applied can cause a loss of performance. We postulate that some relevant domain shifts can be inferred from the distribution of classifier outputs. This report proposes to perform hypothesis testing on large subsets of samples to detect changes in distribution before narrowing down the number of suspicious samples.

Keywords: Domain shift; Hypothesis testing; Kolmogorov-Smirnov; Gaussian mixture

1 Introduction

Training a classifier on a *training dataset* stems from the expectation that if the performance is satisfying on the training data, it should also be so on some, different, application data. In practice, this is not always the case as training sets are rarely exactly representative of the domain which actually needs to be studied by the classifier. *Shifts* between the training domain for which the classifier optimized its internal parameters and the application domain mean the classifier has little reason to maintain a satisfying performance.

This is *particularly problematic* as machine learning takes on a more and more prominent part in large scale task automation and classification tasks difficult to human beings. Beyond the fact that classifiers rarely have perfect performance on the training set to begin with, lies a *more* troubling issue. Indeed, if it is not necessarily meant to work on the application data it is being used on, the classifier's decisions should not be trusted.

An instinctive but problematic answer would be to check by manually labeling the data. Such a method would however often defeat the purpose of using the classifier in the first place. Even if one were to only check part of the data, this part would need to be large enough to provide statistically significant discrepancies. Dynamic changes in streaming data would complicate the task even further, as it is not clear how often the data would need to be checked. It is therefore necessary to find an automated way to spot possible domain shifts. Most such approaches rely on studying the inputs to detect domain shifts, but such shifts are not necessarily a good measure for classifier performance.

We postulate *classifier output distribution* could actually be indicative of such shifts, and a good target for statistical testing. After shortly discussing vocabulary and our general framework (Section 2), we will describe the following parts of our framework: methods to detect windows of problematic samples at application time (Section 3) and means to extract information as to the nature of the problem (Section 4). Finally, after a brief overview of related works (Section 5) we will propose experiments (Section 6) and discuss results (Section 7).

2 Detecting domain shifts

The task we study here is the automated detection of shifts in the domain that lead to loss of classifier performance in a multiclass setting. This setting can be justified for a number of applications as labeling even a fraction of the data can become very costly. That might be either because of the sheer size of the application dataset (when analyzing tweets for instance) or because the action of labeling is costly in itself (as would be the case if the labeling must be done through a destructive test). In a large number of those applications, binary classification does not provide enough information and it is much more convenient to use a multiclass classifier.

The nature of our problem, detecting and dealing with domain shifts that harm classifier performance, requires some specific vocabulary and notions to

be properly understood. In this section we define notations and give a broad overview of our contribution. Two elements of the framework are further discussed in later sections: find a large group of samples that indicate a shift in domain (Section 3) and analyse a group of problematic samples to isolate the most problematic ones (Section 4).

2.1 Problem formulation and notations

Notations Throughout this paper, sets are denoted by calligraphic letters (such as $\mathcal{X}$) and random variables by capital letters (s.a. X). Probability distributions are given by capital letter P with density or mass functions given by p. Other entity denomination will be introduced as such in the relevant sections.

Domains In a dataset, some types of samples can be more abundant than others, even though in theory those samples are drawn from an underlying set $\mathcal{X}$ (s.a. pictures, sentences, ...) where there is no such difference in representation. Loosely speaking, this is what we refer to as a domain. Formally, we define a domain as a pair made of a distribution P on a set from which samples are drawn $\mathcal{X}$ and a joint probability $P(X, Y)$ where X is a random variable representing samples and Y a random variable representing output classes. As pointed out in the introduction, what interests us are shifts between a source domain $\langle P_S \rangle$ on which the classifier would be trained and an application domain $\langle P_A \rangle$ on which it would be used. It is important to note that neither of those domains are tractable in practice as we can only observe samples drawn from both but not the exact distributions or joint probabilities.

Because we specifically study shifts in *classifier output distribution*, $\mathcal{X}$ is taken to be the set of *possible classifier outputs*. As such, the joint probability $P(X, Y)$ is of little interest as it is usually straightforward to compute from classifier outputs and only the probability distribution P over possible outputs $\mathcal{X}$ interests us here.

Classifiers We define classifiers as functions - e.g. some kind of neural network but any class of functions would do - mapping the input space $\mathcal{I}$ to a decision space $\mathcal{D}$: $C : \mathcal{I} \to \mathcal{D}$. Typically this decision space can be taken as $\mathcal{I} = \{labels\}$ but we can take some freedom with this definition and look at the *classifier output space*, which we take as a probability simplex $\mathcal{D} = \mathcal{I} = S^n_+(0, 1)$ where n is the number of labels. In this case, the output is a *probability vector* $(P(label|input))_{label \in \{labels\}}$ where every coordinate gives the probability that the input belongs to the corresponding label. In recent neural network classifiers, the classification decision is often obtained by applying the $argmax$ function to such a probability vector. The **softmax function** $\sigma : \mathbb{R}^n \to S^n_+(0, 1), (l_1, l_2, \ldots, l_n) \mapsto \frac{e^{l_j}}{\sum_{i=1}^n e^{l_i}}$ is typically used in the last layer of neural networks to convert *logits* to such a *probability vector*.

Now that we have laid out the notions of domain and classifier which we work on, we introduce the general framework we propose to detect shifts in domain distribution that impact classifier performance.

2.2 Framework

After some preprocessing described here, the general framework we propose first focuses on detecting domain shifts over large groups of samples (Section 3), before narrowing down the analysis to isolate actually problematic samples(Section 4). The problematic samples pinpointed in this way are meant to allow for a decision to be made by either the system or the user on the nature of the shift and then actions to be taken.

We make the assumption that changes in the *distribution of classifier outputs* could actually be symptomatic of significant shifts between training domain and application domain. Looking at outputs, as opposed to the inputs, should give insight into the 'quality' of the classifier's decisions. If two input samples lead to very different classifier outputs, they should definitely be treated as distinct. Conversely, if two distinct samples actually yield exactly the same classifier output, it might not be pertinent to treat them as separate.

Testing over groups of samples We make the choice to conduct detection of domain shift over large groups (or windows) of samples, as opposed to individual samples. Indeed, testing over individual samples is problematic as one outlier might not be enough to convincingly prove a change in the underlying probability. Since we would therefore have to detect a significant number of outliers in a certain timeframe anyway to declare a shift in the domain, we chose a number n_w of samples to look at in order to make a decision.

To obtain more reliable detection, we use larger values of n_w during the detection part of the framework. Although this will result in many non-problematic samples being isolated by this detection phase, an analysis phase is performed afterwards to narrow down the amount of problematic samples. This step is important if manual labeling by a user is required on problematic samples for instance

Temperature scaling We need the underlying output distribution to be *continuous* for some of the results we use to hold, but this assumption is sometimes challenged by recent neural networks. A well documented quirk of more recent neural networks is a tendency to be largely overconfident in their prediction ([6]). In practice, this translates to some sort of delta peak around 1 in the probability distribution of outputs, and therefore a *discontinuity*.

We address the issue of discontinuous output distribution through the use of *temperature scaling*, a technique suggested to be very efficient in a recent study on network calibration ([6]). Temperature scaling is a technique that only intervenes after the network is trained and has *no incidence* on the actual accuracy of the network. Normally, given output logits $v \in \mathbb{R}^n$ and the softmax operator σ, the probability output vector is given by $\sigma(v)$. In the case of temperature scaling, the output probability vector is given by $\sigma(\frac{v}{T})$ (where $\frac{\cdot}{T}$ is an element wise division by $T \in \mathbb{R}$). Determining the value of T can be done by checking output probability distribution on some validation data. Technically, such calibration should be done on a small part of the validation dataset set aside for that specific

purpose so as to avoid skewing any statistical test done on the validation data afterwards, but this influence is minimal and ignored in our work.

Framework We propose to obtain output logits and perform temperature scaling to obtain probability vectors, and then look at subsets of n_w probability vectors from the testing data with same scaling. If a shift is detected, we announce it (Section 3). On the subset that triggered the detection test, we perform some analysis to narrow down the set of problematic samples (Section 4). Afterwards, a decision can be taken with the information analyzed (manually label some samples, retrain the classifier, ...)

The questions of window detection and window analysis are developed more in depth over the course of the next sections.

3 Leveraging classifier output as an indicator of domain shift

In this section, we present two approaches to domain shift detection, one distance based non parametric approach relying on the Kolmogorov-Smirnov statistic and a direct density estimation using Gaussian mixture models.

Recall that we make the assumption the *distribution of classifier outputs* would shift between training time and application time from P_S to P_A in the case of a change in the domain that causes a loss of classifier performance. For this we need to be able to decide that a window of samples is not drawn from P_S.

Hypothesis testing Given a *null hypothesis* and an *alternative hypothesis*, the two being mutually exclusive, a statistical hypothesis test either rejects the null hypothesis in favor of the alternative hypothesis or fails to reject the null hypothesis with confidence α (s.a. $0, 0.2, 0.5, 1$). It analyzes sample data to evaluate the null hypothesis, typically by computing a single *test statistic* on which the test makes a decision. This decision is reliant on the confidence level α that represents the rate (such as $0\%, 20\%, 50\%, 100\%$) at which the null hypothesis could be wrongly rejected.

Here, we take the *null hypothesis* $(\mathcal{H}_0) P_S = P_A$ (and the *alternative hypothesis* $(\mathcal{H}_1) P_S \neq P_A$) and need to define *methods for sample analysis* given confidence level α. A rejection of the null hypothesis with confidence α means the tested window is caused by $P_A \neq P_S$, e.g. there is indeed a shift in the domain that impacts classifier performance.

Parametric and non-parametric Simple parametric assumptions on the sample analysis are unrealistic as it is doubtful the output distribution can be perfectly modeled by a Gaussian or some other distribution. Nevertheless, it might be possible to model the output distribution well enough by looking at more complicated parametric models like Gaussian Mixtures that should be able to model

the output distribution closely enough at the cost of a more complicated density function and estimation process.

There exist non-parametric tests that make no assumption on the supposed distribution of the tested sample. The advantage of those tests lies in their greater flexibility, although parametric tests are better studied and performant as long as their underlying assumptions hold.

The following two subsection outline sample analysis methods for statistical hypothesis testing that use two distinct approaches: a two sample non parametric test over windows with a well known test statistic over real valued outputs and a parametric density estimation of P_S over complete probability vector outputs. We first describe the two sample approach.

3.1 Kolmogorov-Smirnov test

Here we present a first approach to domain shift detection that takes the output space to be $\mathcal{D} = [0, 1]$ and makes use of well known properties of the two sample Kolmogorov-Smirnov test to detect domain shifts.

Confidence level We propose to simplify classifier output by collapsing the probability vector to one single value that intuitively represents the *confidence level* with which the classifier took its decision. To this end, we make the choice of using the *maximum probability* of the output vector. This provides us with *one real random variable* as output. Our problem in this setting can therefore be interpreted as *deciding if two windows of samples, each window containing i.i.d. samples, were drawn from the same underlying distribution.*

A non-parametric statistical test In the case of statistical tests on samples of real random variables, parametric assumptions are typically fairly simple (Gaussian distribution, ...) which we already pointed out is a problem. We therefore rely on a well known non-parametric test on real random variables: the **Kolmogorov-Smirnov** test ([14]).

Definition 1 (Two samples Kolmogorov-Smirnov statistic [14]). *Let $X_1, X_2, \ldots, X_n$ be $n \in \mathbb{N}$ i.i.d. samples drawn from P_X (resp. $Y_1, Y_2, \ldots, Y_m$ be $m \in \mathbb{N}$ i.i.d. samples drawn from P_Y). $F : x \mapsto \frac{1}{n} \sum_{i=1}^{n} I_{]-\infty,x]}(X_i)$ where $I_{]-\infty,x]}(X)$ is the indicator function over $]-\infty, x]$ (resp. G) denotes the empirical cumulative distribution function for $(X_i)_{i \leq n}$ (resp. $(Y_i)_{i \leq m}$).*

Then the Kolmogorov-Smirnov $KS(F, G)$ statistic is given by

$$KS(F, G) = sup_x |F(x) - G(x)|.$$

Critical values A very interesting property of the two samples Kolmogorov-Smirnov statistic is that, assuming both $(X_i)_{i \leq n}$ and $(Y_i)_{i \leq m}$ are *i.i.d.* from the same *continuous* distribution P (our null hypothesis), the statistic is *distribution free* (it is not affected by the underlying distribution P). This is important as it means we have a criterion to reject the null hypothesis with confidence level α. In fact, the critical values of the statistic in the null hypothesis for confidence level α have been approximated ([14]).

Statistical testing for window detection. A first possible framework we propose is, given $(X_i)_{i\leq n}$ (outputs from the entire validation set) and $(Y_i)_{i\leq m}$ (from the application set), to compute the corresponding statistic $KS(F_X, F_Y)$ and detect a domain shift with significance level α if $KS(F_X, F_Y) > c_\alpha$. It has however been shown ([5]) that the two sample Kolmogorov Smirnov statistic loses in power if $n >> m$ as would typically be the case if we take the whole validation set to compare to application samples.

To remedy this we propose a similar second framework that uses equal sized windows such that $m = n$ by randomly sampling m samples from the validation set. It is not advisable to always re-use the same window from validation as this short sample window is unlikely to be representative of the actual source distribution and will skew the test.

We presented a non parametric distance based approach using the Kolmogorov-Smirnov statistic. We will now propose to use a parametric explicit density estimation approach sensible to internal hyperparameters.

3.2 Estimating P_S with Gaussian mixture matrixes

Performing an estimation of the source distribution P_S as a sum of Gaussians is a well studied problem for which there exist a number of efficient solutions. We propose to use validation outputs to train a Gaussian Mixture model (GMM) in order to obtain a density estimation. A significant advantage of this method is that contrarily to the previous test, it is not constrained to real random variables.

Sorted output vectors Therefore, we use the complete output vectors *sorted by increasing probabilities* as samples. The reason for sorting probability in the output vector by increasing value is so as to avoid explicitly taking into account class distribution. While a shift in class proportion is not innocuous, we feel it is better to let the detection operate on more subtle information.

Framework We propose to fit a Gaussian mixture over a portion of the validation data and use a threshold over the likelihood of a sample window in this approximated distribution. The threshold can be determined by computing likelihood values of windows from the remaining validation data and keeping the α percentile.

In this section we have presented methods to detect suspicious output distribution on fairly large windows. Such windows however are unlikely to only contain problematic samples, and it seems to us necessary to study those windows in more detail before making decisions as to the nature of the problem.

4 Pinpointing possible problematic samples

This section describes the second part of our contribution: the analysis and characterization of a domain shift once it has been detected.

In the case where we detect shifts over windows of samples, not all samples in the window are necessarily problematic. It would therefore be better to be able to refine the detection further by outlining samples that are likely to have caused the detection of a domain shift. Such samples could then be presented to someone who could ideally diagnose the underlying issue (apparition of a new class, ambiguous inputs, ...). At the very least, it would be possible to obtain labeled samples for problematic cases.

Therefore, the problem treated in this section is the following: *given a window of samples and knowing this window is likely to have been taken from a different domain than the one training data is from, highlight informative aspects of the window that are caused by the detected domain shift.*

Biggest difference in mass The method we propose is to look at the maximum probabilities that are much more likely to appear in the tested window than in the reference. The idea behind this is that, while the distribution of confidence levels is a bit more peaked around 1 in the reference than in tested windows, this probability mass is 'spread out' over the rest of the possible confidence values. We assume that the problematic samples will be more concentrated around confidence values with inflated representation in the tested window. Formally, we actually look at samples around $argmax_x \widetilde{P}_{window}(x) - \widetilde{P}_S(x)$. In practice this is implemented through binning of the probabilities and finding the bin with the biggest population difference between the reference and the suspicious window.

In the previous three section we developed tools and methods to detect and adapt to shift in the application domains. The following section provides some context on the issue.

5 Related work

The issue of domain shift detection has surfaced over the years in one form or the other and we now discuss two instances of problems close to it: unknown class detection and concept drift detection.

Unknown class detection Open set recognition is the issue of detecting the apparition of classes unknown to classifiers (and adapting to them). As to the detection of unknown classes, it has been proposed ([9]) to use thresholds on maximum output probability as a baseline and use extreme value theory to derive classifier outputs more fit to such thresholding. However, this method is specific to an adaptation of one class Support Vector Machines to the multiclass setting and cannot be compared to our methods that are meant to work out of the box on trained state of the art classifiers.

Concept drift detection Concept drift is the issue of detecting and adapting to a stream of input data changing distribution over time. Most works on this particular case have focused on either having access to true labels and detecting shifts through issues in classifier performance ([18],[16],[7]) or using statistical

techniques on the input data which leads to detecting shifts irrelevant to the classifier ([4], [17],[12],[11]).

Nevertheless, the idea of leveraging classifier output probabilities to detect concept drift has surfaced over the years but has been mostly applied to binary classification problems to our knowledge. The use of theoretical statistical bounds on windows of Support Vector Machines margins was explored in a binary classification setting ([13],[19],[2],[3]). These approaches are similar to our window detection methods, and in particular our Kolmogorov-Smirnov test. The use of windows of SVM margins (albeit linear SVMs) as proposed in [13],[2],[3] is functionally the same thing as the max of the softmax output we track here. However, we argue that the theoretical statistical bounds derived in [3] are too loose. Furthermore, the $\mathcal{A}-distance$ used in [2] is calibrated directly on the (small) reference which skews the test. Both [13] and [2] use a fixed reference window (the first possible one in the data stream), but this is unlikely to be representative of source distribution. Although [19] uses the Kolmogorov-Smirnov statistic among others (and concludes the Kolmogorov-Smirnov test detects changes faster), it looks at posterior probability $P(label|input)$. Although the experiments have only been run for binary tasks, it is suggested to track n such sequences of probabilities for a n class problem. We argue this is a misleading information to monitor as what truly matters in classification is the confidence of the classifier in the decision, not probabilities of other labels. The windows used in this approach are two contiguous sliding windows, which only allows to detect abrupt changes.

6 Experiments

We have performed experiments to evaluate the various proposed methods for the two tasks we outlined previously: detection of problematic windows and analysis of problematic windows. Experiments were carried out in `python` using the `tensorflow` ([1]) and `scikit-learn` ([15]) libraries among others.

6.1 Evaluation of detection criterions

We evaluate the separation criterions on an easily quantifiable domain shift: the apparition of classes unknown to the classifier. For this task, we train on the CIFAR10 dataset ([10]). The CIFAR10 dataset is a set of small 32×32 pictures from 10 different categories: *plane, trucks, birds, cats, deers, dogs, frogs, horses, cars and ships*. There are 50 000 training pictures with 5000 samples for each category and 10 000 test samples with 1000 samples for each category. We also split 10 000 samples from the training set for validation.

To provide unknown classes to discriminate from, we take pictures from the CIFAR 100 dataset ([10]). The CIFAR100 dataset is a dataset containing pictures from 20 different superclasses, with each 2500 samples normally used for training. We use the dataset's training set from 10 superclasses that do not overlap with

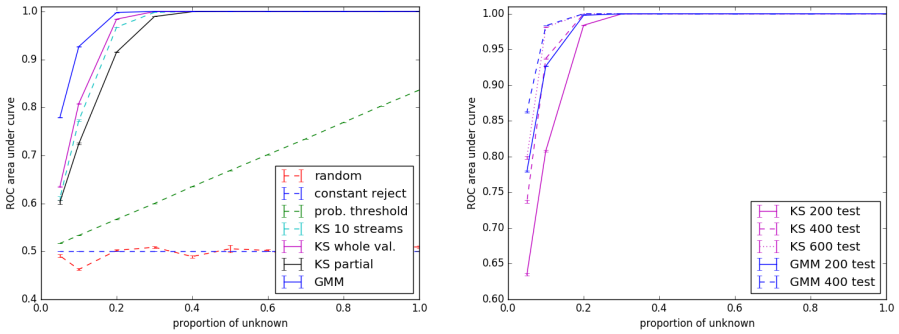
CIFAR 10: *aquatic mammals, fish, flowers, food containers, fruits and vegetables, household furniture, insects, large carnivores, trees and people.*

Once the classifier has been trained on the 40 000 training samples, we have at our disposal 10 000 validation samples, 10 000 test samples from known classes and 25 000 samples from unknown classes. The validation set will be used to calibrate our statistical test.

The 35 000 remaining samples are the ones actually used to draw conclusions. We can emulate generating sample windows from the source distribution by drawing from the 10 000 CIFAR 10 samples. To approximate corrupted sample windows, we can draw some samples from the 25 000 unknown CIFAR100 samples and some from the 10 000 CIFAR10 samples in fixed proportion to simulate varying degrees of domain shifts.

In this way, we try to separate between *non-overlapping* windows of length 200 of completely known data and of mixed data. For a threshold calibrated for confidence level $\alpha = 0.05$, we provide both the *true positive rate* (the proportion of mixed window detected as part of a shift) and the *false positive rate* (the proportion of known classes only windows wrongly detected). To evaluate the separative power of the methods, we also look at the area under the ROC curve (which plots true positive rate against false positive rate by varying the threshold used). The closer this area is to 1, the more powerful the method.

We use a standard Resnet 32 architecture ([8]) (Hyperparameters in Annex). The network is trained on the 40 000 training samples for 50000 iterations on batches of size 128 and typically achieves around 90% accuracy on the 10 000 validation samples.

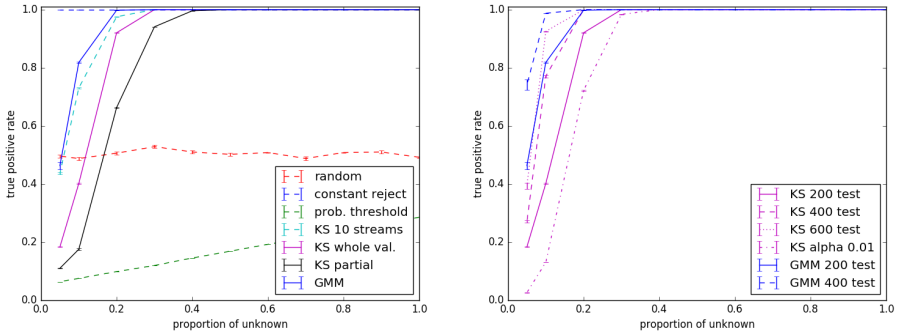


(a) different methods (b) hyperparameter influence

Fig. 1: Roc area under curve in function of the proportion of unknown classes in the mixed set.

We present results for false positive rates (Table 1, our methods are bolded), true positive rates (Figure 2, our methods are in solid line when compared to

others) and ROC area under curve (Figure 1). We include two baseline methods (random rejection of half and always reject tested windows) for reference. For further reference, we test thresholding the maximum output probabilities ([9]) and the Kolmogorov-Smirnov statistic in the case it is used to track posterior probabilities for every class ([19]) but still scale the softmax activations as the result would otherwise be equivalent to always rejecting the window. A 100 components GMM is used with 8000 validation samples used for fitting and 2000 kept for threshold calibration. For each measure, we train ten classifiers and report average values.



(a) different methods (b) hyperparameter influence

Fig. 2: True positive rate in function of the proportion of unknown classes in the mixed set.

method	threshold	KS 10 streams	**KS**	**KS** $\alpha = 0.01$	**KS partial**	**GMM**
false positive rate	$0.053 \pm 6 \cdot 10^{-6}$	$0.243 \pm 5 \cdot 10^{-3}$	$0.057 \pm 1 \cdot 10^{-3}$	$0.004 \pm 7 \cdot 10^{-5}$	$0.049 \pm 1 \cdot 10^{-3}$	$0.122 \pm 3 \cdot 10^{-3}$

Table 1: False positive rates for various methods

6.2 Window analysis

We proposed a method to isolate more problematic samples. As is the case for shift detection, there are distinct possible criterions to look at when evaluating the quality of the analysis. In this case, there are at least two things that need detecting: samples representing unknown classes, and samples on which the classifier made a mistake.

We reuse most of the setup from the unknown class detection experiment (classifier, datasets). In practice, this means selecting at random a window of 200

samples from the mixed set and dividing into bins of width 0.025 to apply the proposed method. Results are presented in Table 2, proportions in the analyzed windows are provided as a baseline. For each measure, we train ten classifiers and report average values and variance.

Unknown proportion	1	0.8	0.6	0.4	0.2	0.05
Unknown bin	1.000 ± 0	$0.948 \pm 4 \cdot 10^{-3}$	$0.851 \pm 2 \cdot 10^{-2}$	$0.646 \pm 3 \cdot 10^{-2}$	$0.427 \pm 4 \cdot 10^{-2}$	$0.155 \pm 1 \cdot 10^{-2}$
Misclassified	1.000 ± 0	$0.820 \pm 6 \cdot 10^{-5}$	$0.631 \pm 9 \cdot 10^{-5}$	$0.454 \pm 4 \cdot 10^{-4}$	$0.271 \pm 9 \cdot 10^{-5}$	$0.147 \pm 1 \cdot 10^{-4}$
Misclassified bin	1.000 ± 0	$0.954 \pm 4 \cdot 10^{-3}$	$0.878 \pm 2 \cdot 10^{-2}$	$0.762 \pm 2 \cdot 10^{-2}$	$0.546 \pm 6 \cdot 10^{-2}$	$0.365 \pm 7 \cdot 10^{-2}$

Table 2: Proportions of unknown and misclassified samples

7 Discussion

In the last section we described experiments over both parts of our contribution and we first discuss the results of our detection module.

Detection False positive rates recorded in Table 1 show that the Kolmogorov-Smirnov based methods we propose follow the expected false positive rates set by our confidence level. Using the theoretical threshold value for the method derived from [19] however leads very high false positive rates as would be expected. While this could be alleviated by dividing critical values by the number of classes to obtain an upper bound, this approach would also make it difficult to reject problematic windows. The false positive rate using GMMs is also difficult to calibrate and sensitive to the number of components used.

True positive rates, as shown by Figure 2 indicate that GMMs are the better detectors of mixed windows followed by the method outlined in [19]. Nevertheless both of those methods exhibit higher false positive rates, and the Kolmogorov-Smirnov based method using the entire validation set is not far worse. While the version using only a small, randomly drawn part of the validation performs slightly worse than the rest, this version only has access to 400 samples as opposed to 10200. The comparison of hyperparameters suggest that both the density estimation and distance based methods benefit significantly from larger window sizes and lowering the confidence level also slightly impacts the ability to reject problematic windows as would be expected.

Regarding the overall discriminative power of the metrics, Figure 1 suggests that the GMM based method would outperform the Kolmogorov-Smirnov based ones, but this is difficult to leverage as it is not clear how to choose the likelihood threshold. All Kolmogorov-Smirnov based methods perform similarly regarding the ROC area under curve, and hyperparameter influence graphs confirm the conclusions drawn from Figure 2.

On the whole, the GMM based approach seems more powerful than the Kolmogorov Smirnov based approaches given properly tuned hyperparameters

(which are not self evident to tune). Nevertheless, it is necessary to keep in mind such a method scales poorly with more classes as the number of components and the time/samples needed to fit a mixture would keep growing too, something also true for the multi stream Kolmogorov-Smirnov baseline used. Therefore, it seems to us that the GMM based method should be used for classifiers with fewer classes while the Kolmogorov-Smirnov methods should be able to be used for larger classifiers at the cost of less separative power.

Analysis Experiments in table 2 show that problematic samples outlined by our mass difference method are significantly more likely to be from unknown classes compared to samples from the window in general. This phenomenon is especially flagrant for windows with low amounts of unknown classes (tripling the proportion of unknown samples for 5% of unknown classes), though the method is not enough to only isolate unknown samples. A similar conclusion can be reached looking at misclassified samples, and the proportion of misclassified samples could be considered as justification enough to have users manually label the samples.

8 Conclusion

We proposed a modular framework to detect domain shifts relevant to classifier performance and isolate problematic samples in the test domain. By looking at classifier outputs, we could detect shifts directly affecting the classifier and use hypothesis testing on large windows - implemented through both a density estimation and a sample comparison method - to spot shifts in the domain. Leveraging differences in probability mass between problematic samples and a reference source distribution, we could then isolate a small portion of problematic samples from those large windows.

A natural continuation of the work presented here would be proposing ways to leverage the problematic samples singled out after window analysis, like retraining the classifier. One solution could be to only retrain the last layer of a deep classifier to make up for the limited amount of new data. Nevertheless, this solution would leave open questions regarding the training set used, and more importantly what validation set would be used afterwards. This last point would be crucial as one of the great advantages of our method over concept drift methods is that we can rely on a (large) validation set, something heavily used in our methods.

Beyond the issue of retraining the classifier, there is still much to explore as to the possible ways to leverage the outputs of our window analysis methods. For instance, a purely statistical study over problematic output vectors might be used to provide preliminary indications as to the nature of the shift even before any kind of user feedback. It has been discussed that non maximal probabilities carry some information as to the decision process of a classifier, which makes those a good target for statistical tests and metrics.

References

[1] Martin Abadi et al. *Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[2] Mark Dredze, Tim Oates, and Christine Piatko. "We're not in Kansas anymore: detecting domain changes in streams". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2010.

[3] Anton Dries and Ulrich Rückert. "Adaptive concept drift detection". In: *Statistical Analysis and Data Mining* 2.5-6 (2009), pp. 311–327. ISSN: 1932-1872.

[4] Elaine R Faria, Joao Gama, and Ande CPLF Carvalho. "Novelty detection algorithm for data streams multi-class problems". In: *Proceedings of the 28th annual ACM symposium on applied computing*. ACM. 2013.

[5] Alexander Y Gordon, Lev B Klebanov, et al. "On a paradoxical property of the Kolmogorov–Smirnov two-sample test". In: *Nonparametrics and Robustness in Modern Statistical Inference and Time Series Analysis: A Festschrift in Honor of Professor Jana Jurečková*. Institute of Mathematical Statistics, 2010, pp. 70–74.

[6] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *CoRR* (June 14, 2017).

[7] Maayan Harel et al. "Concept drift detection through resampling". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014.

[8] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *CoRR* abs/1603.05027 (2016). arXiv: `1603.05027`. URL: `http://arxiv.org/abs/1603.05027`.

[9] Lalit P. Jain, Walter J. Scheirer, and Terrance E. Boult. "Multi-class Open Set Recognition Using Probability of Inclusion". In: *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*. Ed. by David Fleet et al. Springer International Publishing, 2014, pp. 393–409. ISBN: 978-3-319-10578-9.

[10] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: Technical report, University of Toronto, 2009.

[11] Ludmila I Kuncheva and William J Faithfull. "PCA feature extraction for change detection in multidimensional unlabeled data". In: *IEEE transactions on neural networks and learning systems* 25.1 (2014), pp. 69–80.

[12] Jeonghoon Lee and Frederic Magoules. "Detection of concept drift for learning from stream data". In: *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*. IEEE. 2012.

[13] Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. "Drift detection using uncertainty distribution divergence". In: *Evolving Systems* 4.1 (Mar. 2013), pp. 13–25. ISSN: 1868-6486.

[14] Frank J Massey Jr. "The Kolmogorov-Smirnov test for goodness of fit". In: *Journal of the American statistical Association* 46.253 (1951), pp. 68–78.

[15] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[16] Gordon J. Ross et al. "Exponentially weighted moving average charts for detecting concept drift". In: *Pattern Recognition Letters* 33.2 (2012), pp. 191–198. ISSN: 0167-8655.

[17] Tegjyot Singh Sethi, Mehmed Kantardzic, and Hanquing Hu. "A grid density based framework for classifying streaming data in the presence of concept drift". In: *Journal of Intelligent Information Systems* 46.1 (2016), pp. 179–211.

[18] Heng Wang and Zubin Abraham. "Concept drift detection for streaming data". In: *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE. 2015.

[19] I. Zliobaite. "Change with Delayed Labeling: When is it Detectable?" In: *2010 IEEE International Conference on Data Mining Workshops*. Dec. 2010.

A Classifier architecture

The classifiers used in experiments were based on the Residual Network architecture proposed in [8] with following hyperparameters according to the implementation proposed at `https://github.com/wenxinxu/resnet-in-tensorflow`:

- 5 residual blocks (32 layers)
- Initial learning rate of 0.1 with decay to 0.01 after 40000 iterations on batches of size 128
- Weight decay of scale 0.0002 for regularization