DIRICHLET WRAPPER TO QUANTIFY CLASSIFICATION UNCERTAINTY IN BLACK-BOX SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Nowadays, machine learning models are becoming a utility in many sectors. AI companies deliver pre-trained encapsulated models as application programming interfaces (APIs) that developers can combine with third party components, their models, and proprietary data, to create complex data products. This complexity and the lack of control and knowledge of the internals of these external components might cause unavoidable effects, such as lack of transparency, difficulty in auditability, and the emergence of uncontrolled potential risks. These issues are especially critical when practitioners use these components as black-boxes in new datasets. In order to provide actionable insights in this type of scenarios, in this work we propose the use of a wrapping deep learning model to enrich the output of a classification black-box with a measure of uncertainty. Given a black-box classifier, we propose a probabilistic neural network that works in parallel to the black-box and uses a Dirichlet layer as the fusion layer with the black-box. This Dirichlet layer yields a distribution on top of the multinomial output parameters of the classifier and enables the estimation of aleatoric uncertainty for any data sample. Based on the resulting uncertainty measure, we advocate for a rejection system that selects the more confident predictions, discarding those more uncertain, leading to an improvement in the trustability of the resulting system. We showcase the proposed technique and methodology in two practical scenarios, one for NLP and another for computer vision, where a simulated API based is applied to different domains. Results demonstrate the effectiveness of the uncertainty computed by the wrapper and its high correlation to wrong predictions and misclassifications.

1 INTRODUCTION

The popularity of machine learning is giving birth to new business models based on the productization and service of these models. In the market there are many application programming interfaces (APIs) serving predictions in object recognition for images (Vision AI[1]), language detection or sentiment analysis in natural language processing (Cloud Natural Language API[2]), to mention just a few. As this Machine Learning-as-a-Service model starts to grow, it becomes easier to find these APIs as an integral component of more complex products.

The use of pre-trained models gives rise to two different problems. First, we do not know how these models are going to operate in our intended application domain. In order to address this issue, there is a vast literature on transfer learning that can be applied. However, when using third-party proprietary software or APIs, we may not have access to the internals or the possibility of fine-tuning the model to our domain. If we are to use the model as it is, one must at least understand when the model is going to work and when it is not, to have some confidence metric that tells about the expected performance of the methods when applied to our problem. However, this information is not always provided, especially in deep learning models. This effect can be worsened when these components are just one of the many different parts of a data product. This complexity leads us to the second problem: when different models might interact in complex pipelines, the construction of the appropriate confidence measures can be a challenging task.

[1] https://cloud.google.com/vision/

[2] https://cloud.google.com/natural-language/

In order to solve the previous issues, in this article, we propose a deep learning wrapper algorithm that equips any black-box model with uncertainty prediction. Here a wrapper is understood as a machine learning model that takes any other model and operates without accessing its internals. Because it does not have access to the internal states, parameters, or architecture of the model it is wrapping, the wrapper is model agnostic and can be used on top of any other algorithm as long as it satisfies some desideratum. In this article, we only require the black-box model to produce as output a distribution over the classes, a soft requirement as any model with a soft-max layer satisfies it. More specifically, the proposed wrapper uses a deep learning model and introduces a Dirichlet layer as the fusion layer with the black-box. This Dirichlet layer yields a distribution on top of the multinomial output parameters. By sampling from this Dirichlet distribution, the wrapper enables the estimation of aleatoric uncertainty.

Uncertainty has been an important topic in machine learning for many years (Koller & Friedman, 2009). With the emergence of deep learning, the reinterpretation of some existing mechanisms such as dropout, or the proposal of stochastic mechanisms such as Montecarlo approaches, has broadened the use of these techniques for accounting for uncertainty in deep models (Gal, 2016).

Uncertainty can be categorized into epistemic and aleatoric uncertainty. While the first accounts for the uncertainty that is associated to model parameters, the second corresponds to the uncertainty inherently present in the data[3].

Uncertainty plays a key role when reporting a decision because it accounts for the reliability of the prediction and can help to show the limitations of the applicability of a machine learning model. In this respect, we advocate for the use of selective prediction (aka rejection techniques) when the uncertainty metric is large in order to avoid potential harm or avoid risks. Selective prediction is a set of techniques based on abstaining from deciding according to some metric threshold. As previously commented, uncertainty is a good candidate for a rejection metric. In literature, we find examples of different rejection functions (Geifman & El-Yaniv, 2019) (De Stefano et al., 2000) and some of them use uncertainty measures (Geifman & El-Yaniv, 2017) for rejection.

In the proposed scenario, where we are trying to characterize the uncertainty of a black-box non-mutable model, many of the state-of-the-art techniques are not applicable. For example, some rejectors have to be trained together with the classifier and need access to the internals of the model. In the same line, current models for uncertainty in deep learning need to have access to its internal states (Gal, 2016).

The contributions of this article can be summarized as follows:

- We propose a wrapper algorithm that equips any other classification model that outputs a distribution over predicted classes with Bayesian treatment without having knowledge or access to its internals.
- We use the wrapper to empirically estimate aleatoric uncertainty and show that the computed uncertainty can identify prone to err samples.
- Finally, we show that the computed uncertainty can be used by rejection techniques to increase the performance and robustness of the original black-box model in the target domain. We show improvements in transfer problems in natural language processing and computer vision problems.

In section 2, we introduce the method proposed for building an uncertainty wrapper around a black-box model. In section 3, we describe how to obtain an uncertainty score from the wrapper output. Section 4 introduces the concept of rejection and rejection performance metrics. In section 5, we showcase the proposed method in four different scenarios for sentiment analysis in natural language processing and one for computer vision. The results obtained corroborate the importance of the rejection method and show the success of the proposed methodology. Finally, section 6 concludes the article.

[3]Observe that in this application only aleatoric uncertainty matters since we are dealing with pre-trained, non-mutable models. In this particular scenario, aleatoric uncertainty also serves as a measure of the fitness of the model to the data.

2 BUILDING AN UNCERTAINTY WRAPPER

Our goal is to build a wrapper algorithm that takes another black-box model and operates on top of it. As such, there are several constraints to observe. First, we need to exclusively operate on the inputs and outputs of the black-box classifier. We are not allowed to use any intermediate or internal value of the black-box model as we need to be agnostic to it. Second, the input of the wrapper has to be compatible with the original distribution over the output classes.

In the literature, other proposals suggest a deep learning model for estimating uncertainty. The problem with those approaches, like in (Kendall & Gal, 2017) where they use independent Gaussian random variables to model the pre-activation value of the logits, is that they do not conform to the constraints in our setting. First, having access to the logits before the softmax breaks the black-box assumption; and, second, independent Gaussian distributions impose unnecessary assumptions and need of additional normalization steps. A more natural approach is to consider the output distribution coming from a *Dirichlet* probability distribution function.

2.1 DIRICHLET CONCENTRATION REPARAMETERIZATION

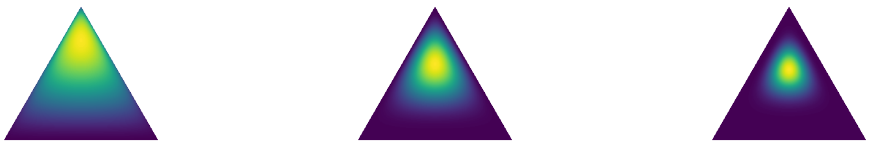
As commented, given a data set $\mathcal{D}$ composed of pairs $(x_i, y_i), i = 1 \dots N$, with $y_i \in \mathbb{R}^C$, being C the number of different classes, the wrapper output is assumed to come from a Dirichlet probability density function:

$$p(y_w|X, w^*) \sim Dir(\boldsymbol{\alpha}), \tag{1}$$

where w^* are the parameters of the wrapper. We propose to use a decomposition of the concentration parameter in two terms to relate the output of the black-box classifier, y_m, with the concentration parameter, α, in the Dirichlet distribution of the wrapper. To that effect, we recall some basic statistics of the Dirichlet distribution. Given a Dirichlet random variable $\boldsymbol{x} \in \mathbb{R}^C$ with concentration parameter $\boldsymbol{\alpha} \in \mathbb{R}^C$, the expected value of the distribution is defined as $\mathbb{E}(\boldsymbol{x}) = \boldsymbol{\alpha} / \sum_{i=1}^{C} \alpha_i$.

Observe that the expected value has the same properties as a probability distribution and that the output of the black-box $\boldsymbol{y_m} \in \mathbb{R}^C$ is already a probability distribution. In this sense, we could directly use the output of the black box as the concentration parameter. However, each term of the concentration parameter is not necessarily constrained to the interval $[0, 1]$. Let us introduce a new scalar parameter, $\beta \in \mathbb{R}$ that will model this difference, such that $\boldsymbol{\alpha} = \beta \boldsymbol{y_m}$.

Observe that the value of β does not change the expected value of the output of the wrapper and coincides with the output of the black-box model, i.e. $\mathbb{E}(\boldsymbol{y}_w) = \boldsymbol{y}_m$.



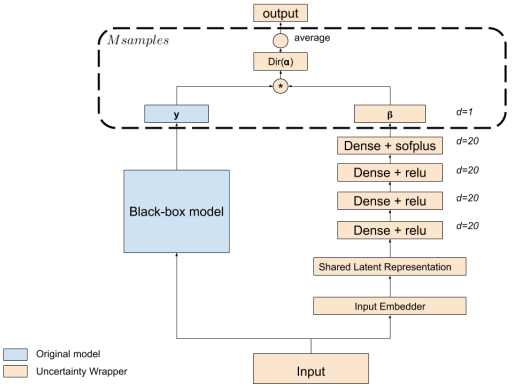
(a) $\beta = 20$ (b) $\beta = 20$ (c) $\beta = 20$

Figure 1: Dirichlet distribution in 3 dimensions for different β values given a prediction of $[0.25, 0.25, 0.50]$

This decomposition has a simple interpretation: While the output of the black-box classifier stands for the mean, parameter β accounts for the spread of the distribution. The same or similar decomposition can be found in other works in a different context(Malinin & Gales, 2018)(Chen et al., 2018)[4]. An example of the effect of varying this parameter in a three dimensional Dirichlet distribution is shown in Figures (1a) to (1c). Observe that the higher the value of β, the more pointy the distribution is.

[4]It is worth noting that in the context of those works, there is a degradation in performance when using Dirichlet. This does not happen in our case since the black-box model is non-mutable.

Figure 2: Model used to estimate the aleatoric uncertainty from the original black-box model



This decoupling allows to effectively isolate the contribution of the black-box and the contribution that remains to be computed, i.e. the value of parameter β. Figure 2 shows the integration of the wrapper (in light orange colour) with the black-box classifier (in light blue colour). Observe that the wrapper consists of two blocks: the Dirichlet reparameterization layer of the wrapper that decouples the influence of the black-box model from the rest (see the dashed line), and a deep learning architecture which aims to compute the scalar value of β[5].

2.2 Inference in the Dirichlet setting

SImilarly to (Kendall & Gal, 2017), we approximate the expected value of the classification probabilities using Monte Carlo sampling from the learned Dirichlet distribution for each sample, $\hat{y}_{.,i} \sim Dir(\alpha_i)$ as $\mathbb{E}[\hat{y}_i] = \frac{1}{M}\sum_{m=1}^{M} \hat{y}_{m,i}$.

This distribution is used to define the loss function for our learning stage. Given a set of N training samples, we will use a regularized version of the cross-entropy loss function as follows:

$$\mathcal{L}(W) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{C}\sum_{c=1}^{C} y_{i,c}\log \mathbb{E}[\hat{y}_i]_c + \lambda\|\beta\|_2 = -\frac{1}{N}\frac{1}{C}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log\big(\frac{1}{M}\sum_{m=1}^{M}\hat{y}_{m,i,c}\big) + \lambda\|\beta\|_2.$$

Observe that we introduce the norm of the β value in the minimization function. This term is required since the unregularized cross-entropy forces the value of β to grow unbounded. By adding this term, we control its growth and govern the trade-off with a scalarization parameter λ.

3 Obtaining an uncertainty score from the wrapper

The described Dirichlet layer effectively allows studying the variability of the parameters of the black-box output. This variability can be used to approximate a value for the heteroscedastic aleatoric uncertainty. In this work, we use Monte Carlo simulation sampling from the obtained Dirichlet function in order to characterize the uncertainty (Gal, 2016).

Standard techniques for measuring uncertainty includes *variation ratios* or *predictive entropy*. *Variation ratios* measures the variability of the predictions obtained from the sampling (Freeman, 1965) by computing the fraction of samples with the correct output. Alternatively, *predictive entropy* considers the average amount of information contained in the predictive distribution. Those results with low entropy values correspond to confident predictions, whereas high entropy leads to large uncertainty. Since the output of the black-box model $\boldsymbol{y}^m$ already describes a probability distribution, one could compute its predictive entropy and obtain a measure of its uncertainty with $\mathbb{H} = -\sum_c y_c^m \log y_c^m$

[5]The architecture used in this figure corresponds to the one used in the experimental section.

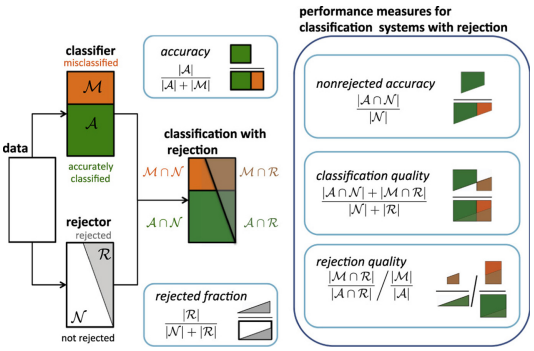
However, as the wrapper allows us to model the variability of the parameters of the black-box output distribution, we can compute a predictive entropy that takes into account the variability of the predicted value. In this case, the sampled predictive entropy is defined as $\mathbb{H} = -\sum_c \mathbb{E}[\hat{y}]_c \log \mathbb{E}[\hat{y}]_c$. As we show in the experimental section, this latter approach captures better the uncertainty compared to the predictive entropy of the original model.

4 USING UNCERTAINTY FOR REJECTION

Rejection is a mechanism that, given a particular metric related to the confidence in the decision, allows discarding a prediction if the metric value is below some threshold. In our proposal, we use the wrapper computed uncertainty as this rejection metric. In the context of our use cases, the hypothesis is that texts or images with high uncertainty are prone to be misclassified by the black-box model.

In order to use the uncertainty score for evaluating the performance of the black-box in a new dataset, we first proceed to obtain the predictions applying the original model. Then for each pair of data and prediction, we obtain the associated uncertainty score using the wrapper. Next, we sort the predictions based on the uncertainty score, from more uncertain to more confident. From that ordering, we set the rejection threshold that marks where to start trusting the classification model.

Figure 3: Rejection performance metrics as proposed in (Condessa et al., 2015)



In order to evaluate the rejection metric, we split the dataset using two criteria: whether the method **R**ejects the data point or **N**ot; and whether the point is **A**ccurately classified, or **M**isclassified named as R, N, A or M respectively. Using this terminology, we follow the guidelines in (Condessa et al., 2015) for rejection quality metrics. We have three quality metrics, illustrated in 3:

- **Non-rejected Accuracy** measures the ability of the classifier to classify non-rejected samples accurately: $NRA = \frac{|A \bigcap N|}{|N|}$
- **Classification Quality** measures the ability of the classifier with rejection to classify non-rejected samples accurately and to reject misclassified samples: $CQ = \frac{|A \bigcap N| + |M \bigcap R|}{|N| + |R|}$
- **Rejection Quality** measures the ability to concentrate all misclassified samples onto the set of rejected samples:$RQ = \frac{|M \bigcap R||A|}{|A \bigcap R||M|}$

A good rejection point will show a trade-off between the three metrics, being able to divide the misclassified predictions from the right ones and preserve only those points that provide useful information. The higher the value displayed, the better that metric performs for rejection.

5 EXPERIMENTS AND RESULTS

This section describes the experiments run for validating the wrapper proposal and results obtained. The experiments include two different scenarios: a use case for sentiment analysis using natural language processing and, another, for image classification.

5.1 A NATURAL LANGUAGE PROCESSING SCENARIO

In order to validate the proposal, we use an NLP-based sentiment analysis system applied to product reviews. The goal of the system is to classify each review on whether it is positive or negative. The goal of the experiment is two-fold. First, we want to show how to apply the wrapper for a given NLP task. Second, we demonstrate how the proposed method additionally captures the uncertainty caused by the change in domains. To this end, we include different combinations of training and prediction domains in the experiment. The details on the datasets used are the following:

- Stanford Sentiment Treebank (Socher et al., 2013), SST-2, binary version where the task is to classify a movie review in positive or negative. The dataset is split in 65,538 test samples, 872 for validation and 1,821 for testing.
- Yelp challenge 2013[6], the goal is to classify reviews about Yelp venues where their users rated them using 1 to 5 stars. To be able to reuse a classifier trained with the SST-2 problem, we transform the Yelp dataset from a multiclass set to a binary problem, grouping the ratings below three as a negative review, and as positive otherwise. The dataset is split in 186,189 test samples, 20,691 for validation and 22,991 for testing.
- Amazon Multi-Domain Sentiment dataset contains product reviews taken from Amazon.com from many product types (domains) (Blitzer et al., 2007). As in Yelp, the dataset consists on ratings from 1 to 5 stars that we label as positive for those with values greater or equal to 3, and negative otherwise, split into training, validation and test datasets. We use two of the domains available: music (109,733/12,193/52,254 examples) and electronics (14,495/1,611/6,903 examples).

5.2 AN IMAGE CLASSIFICATION SCENARIO

In addition to the NLP use case presented above, we include here a use case for image classification. The task, in this case, is to classify images in one of the categories defined in the dataset. As in NLP, an image classifier trained using a source dataset, acting as the original API, is then applied to a new set of images belonging to a different dataset. Both datasets share almost the same output classes except for one. By predicting the uncertainty of the different class, we will show how the predicted uncertainty can also be used to detect out of sample images. The details on the datasets used for the vision use case are the following:

- STL-10 (Coates et al., 2011), The STL-10 dataset is an image recognition dataset for developing unsupervised feature learning, deep learning, self-taught learning algorithms. It is inspired by the CIFAR-10 dataset but with some modifications. It includes 500 training images, 800 test images per class, belonging to 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck.
- CIFAR10 (Krizhevsky, 2009), The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes(airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), with 6000 images per class. There are 50000 training images and 10000 test images.

5.3 EXPERIMENT SET UP

On every experiment, we use two datasets: (i) a source dataset for training a model, that will be considered the black-box model from that moment on and (ii) a target dataset that corresponds to the domain we want to apply the black-box model and where to measure the uncertainty using the proposed wrapper . Specifically, the steps followed on each case are:

- Train the black-box. First, we train a classifier with the source dataset. In real scenarios, this step would not be necessary as we would be using a pre-trained model or third-party API.
- Apply the black-box to the target domain. In this step, we use the black-box to obtain the predictions and evaluate the accuracy of the target dataset, and we can compute the predictive entropy based on the prediction outputs.

[6] `https://www.yelp.com/dataset/challenge`

- Compute the uncertainty for the target domain using the wrapper model. Once we have the predictions for the target domain, we proceed to train the uncertainty wrapper to approximate the Dirichlet pdf for each input. By sampling the pdf, we compute the sampling predictive entropy of the average of the outputs to get the uncertainty score for each element in the target dataset.
- Apply the rejection mechanism. Finally, we use the uncertainty score to sort the predictions from more to less uncertain, and we search for a rejection point that maximizes the three performance measures: non-rejected accuracy, and classification and rejection quality.

We run five different scenarios, including the training the black-box with the *Yelp* dataset and applying it to SST-2 and vice-versa, training the black-box with the *Amazon electronic products reviews* dataset and applying it to *Amazon music products reviews*, and vice-versa, and training with STL-10 and applying the black-box to CIFAR10. For each scenario, a selection of optimal training parameters was carried out, including learning rates, batch sizes, number of units and number of epochs. Details on the architectures used for the black-box are given in the Appendix A.

5.4 RESULTS

In order to show the effect of the application of the uncertainty wrapper on each of the target domains, we compute the uncertainty score using the three different metrics described in section 4: the predictive entropy of the black-box output (*baseline*), the predictive entropy obtained after training the aleatoric wrapper (*pred. entropy*), and the variation ratios (*var. ratios*). Figures 4 to 8 show the results obtained on each combination for the rejection performance metrics for the three uncertainty scores analysed. From left to right, we find the values for non-rejected accuracy, classification quality and rejection quality. The higher the value in the plot, the better the result.

According to the results obtained, the proposed method shows better behaviour in all scenarios and metrics. As we remove more samples according to the uncertainty, the proposed method displays much better accuracy and quality than its counterparts. These results validate the hypothesis that the heteroscedastic aleatoric uncertainty computed by the wrapper effectively captures the confidence in the prediction and the samples prone to error. On the contrary, variation ratios are the worst performant method. Note that, although our proposal performs much better, its absolute gain depends on the scenario. In those domains where the black-box model performs worse, there is more to gain by using the wrapper. If we observe the classification quality (plot at the center of each figure) and the rejection quality, we can see that the proposed metric is also excellent at rejecting the misclassified points. A detailed table with numerical results for the same experiments is included in Appendix B.

Results demonstrate how the usage of the uncertainty for rejecting uncertain predictions helps with the adaptation of a pre-trained model to new domains of application. In some cases, the results obtained for the test dataset of the target domain by rejecting 10% of the less certain points overtake those obtained by the source dataset used for training the original model. As a curiosity, the use case where we trained a black-box model using the reviews of Amazon's electronics products achieves better results when applied to the test target dataset than to the original test dataset. Even in this case, where the applied classifier reaches an accuracy of more than 90 %, the proposed method increases it in almost 5 points. In Appendix C, we analyse how, for the case of images, the proposed method can detect out-of-sample images that belong to an unseen category.

6 CONCLUSIONS AND FUTURE WORK

In this work, we introduced a deep learning wrapper technique that can endow any black-box model with uncertainty features. The wrapper uses a reparameterization trick on the Dirichlet distribution, and it can capture the distribution on the multinomial parameters of the output of the black-box classifier.

We use the predicted uncertainty to fuel a rejection method and show how this helps in assessing the fitness of a model to a new domain or data set. By measuring the sampling uncertainty and using it for rejection, we can improve the accuracy results by 4%-8% by rejecting just 10% of the samples. Additionally, the method displays a significant value on rejection quality. These results tell us that

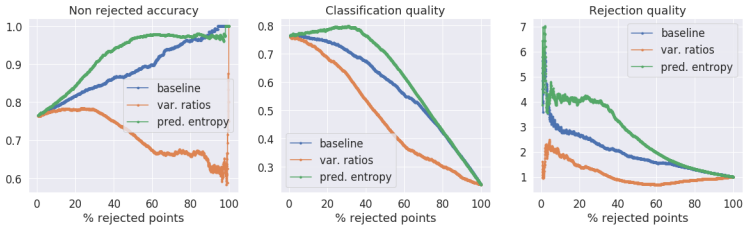


Figure 4: Apply Yelp BB to SST-2

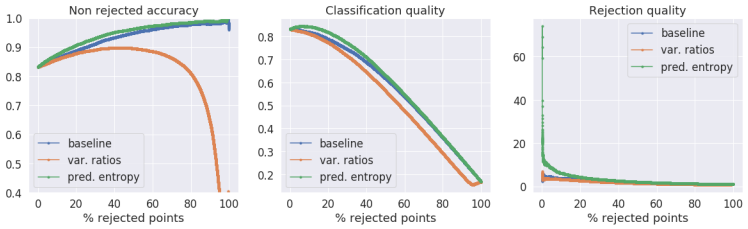


Figure 5: Apply SST-2 BB to Yelp

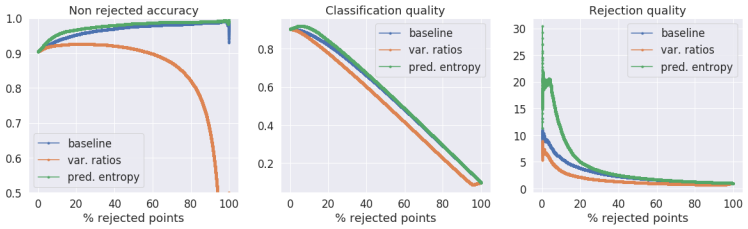


Figure 6: Apply electronics BB to Music

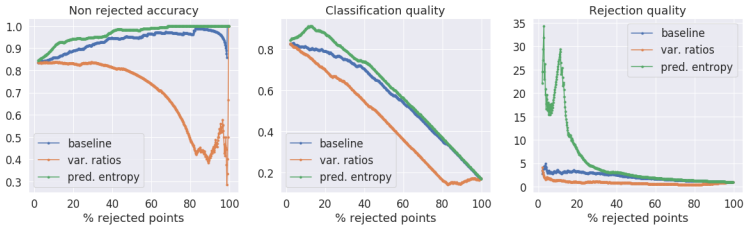


Figure 7: Apply music BB to electronics

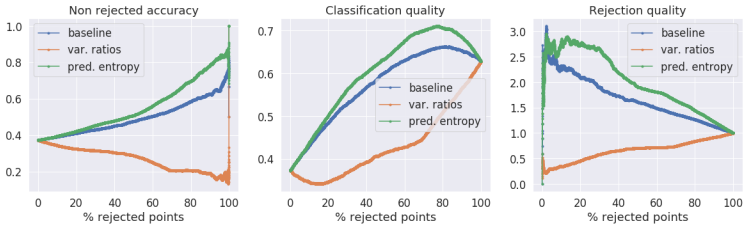


Figure 8: Apply STL-10 BB to CIFAR10

the predicted uncertainty focuses on intricate, ambiguous, or prone to error cases. We show results in NLP and computer vision domains with successful and encouraging results.

As future work, we are planning to keep exploring different architectures and strategies for the wrapper implementation and focus on other usual cases found in real-life implementations, such as how to deal with high dimensional and categorical outputs.

REFERENCES

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-1056.

Wenhu Chen, Yilin Shen, Xin Wang, and William Wang. Enhancing the robustness of prior network in out-of-distribution detection. *CoRR*, abs/1811.07308, 2018. URL http://arxiv.org/abs/1811.07308.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudk (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/coates11a.html.

Filipe Condessa, Jelena Kovacevic, and José M. Bioucas-Dias. Performance measures for classification systems with rejection. *CoRR*, abs/1504.02763, 2015. URL http://arxiv.org/abs/1504.02763.

C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: That is the question-an answer in case of neural classifiers. *Trans. Sys. Man Cyber Part C*, 30(1):84–94, February 2000. ISSN 1094-6977. doi: 10.1109/5326.827457. URL http://dx.doi.org/10.1109/5326.827457.

L.C. Freeman. *Elementary applied statistics: for students in behavioral science*. Wiley, 1965. URL https://books.google.es/books?id=r4VRAAAAMAAJ.

Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4878–4887. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7073-selective-classification-for-deep-neural-networks.pdf.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 2151–2159, 2019. URL http://proceedings.mlr.press/v97/geifman19a.html.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7047–7058. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7936-predictive-uncertainty-estimation-via-prior-networks.pdf.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL http://arxiv.org/abs/1801.04381.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

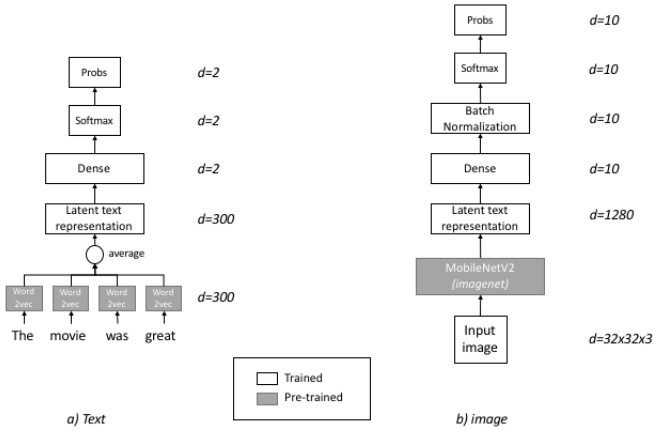
Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

A APPENDIX A

For the sake of reproducibility, this Appendix details the architectures used for training the black-box systems. Figure 9 describes the model used for training the black-box models in the two use

Figure 9: Models used to train the black-boxes: a) is the one used for text and b) describes that used for images



cases. As stated before, the only purpose of this model is to obtain a black-box classifier for a given source domain. The goal, in this case, is not to obtain the best classifier but to obtain a model which is easy to train and offers good performance.

The main difference between the model for NLP and Image classification comes from the embedding component. In the case of NLP, we opted for representing a sentence as the average value of the embedding of each word using pre-trained word2vec embeddings. In the case of images, we trained a MobileNET v2 model (Sandler et al., 2018), initialized with imagenet weights, using as input the STL-10 images, resized to 32x32x3 to accommodate them to the CIFAR10 dataset.[7]

B APPENDIX B

Table 1 shows a detail of the numerical results obtained during the experiments for the four combinations tested. The first column, black-box source acc, describes the accuracy obtained for the source dataset after training the original classifier. Next, column black-box target acc describes the accuracy obtained when applying the black-box to the target dataset. The rest of the columns show the non-rejected accuracy and the classification and rejection quality after rejecting 10, 20 an 30% of the points, using the proposed predictive entropy as a rejector.

[7] we tried other embeddings such as ELMO, and Seq2seq for text, or VGG-16 (Simonyan & Zisserman, 2015) and ResNET50 (Szegedy et al., 2015) for images, but we stick to word2vec and MobileNet due to limitations on the computing resources.

Table 1: Accuracy obtained by training an standalone classifier, applying the API and the proposed wrapper for each domain

	BB source acc.	**BB target acc.**	**Non-reject. acc. (10/20/30%)**	**Class. quality (10/20/30%)**	**Reject. quality (10/20/30%)**
Apply Yelp BB to SST-2	89.18±0.08%	77.13±0.52%	81.38±0.72% 85.83±0.88% 90.08±0.94%	78.82±0.91% 79.66±1.15% 78.46±1.31%	4.66±0.63 4.33±0.44 3.69±0.31
Apply SST-2 BB to Yelp	83.306±0.18%	82.106±0.88%	86,34±0.18% 89.44±0.38% 92.08±0.33%	83.27±0.88% 80.95±0.38% 76.77±0.46%	5.98±1.63 4.10±0.27 3.21±0.10
Apply Electronics BB to Music	86.39±0.22%	90.38±0.13%	95.04±0.43% 96.45±0.35% 97.26±0.31%	90.67±0.88% 83.93±0.67% 75.77±0.54%	10.7±1.65 4.82±0.35 3.25±0.14
Apply Music BB to Electronics	93.10±0.02%	83.06±0.0%	91.79±0.31% 94.90±0.85% 96.00±0.83%	90.27±0.54% 86.22±1.33% 79.91±0.98%	19.19±2.9 6.60±0.84 4.02±0.25
Apply STL-10 to CIFAR10	53.53±0.12%	39.29±0.08%	42.53±0.04% 45.33±0.04% 47.78±0.05%	46.22±0.06% 52.18±0.06% 56.55±0.08%	2.56±0.05 2.62±0.02 2.25±0.01

C APPENDIX C

This Appendix shows detailed results on the image case. Although the resulting quality obtained for the rejection mechanism in the case of images is not as large as in texts, when comparing to the predictive entropy of the original classifier, we observe that the proposed measure is still excellent for detecting out of sample images. The main difference between STL-10 and CIFAR10 is a variation on one of the classes. Where in STL-10 class 6 held monkeys, in CIFAR10 it corresponds to frogs. As one can expect, the black-box model trained with STL-10 will struggle on detecting frogs.

Figure 10: Distribution of the predicted entropies for two of the CIFAR10 classes.

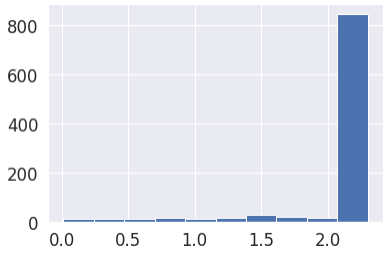


Figure 11: Entropies for frogs

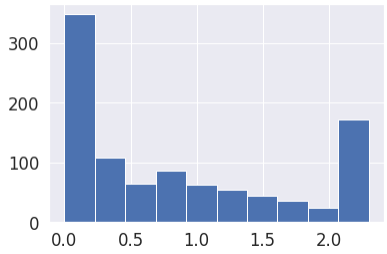


Figure 12: Entropies for trucks

In figure 11 and 12, we can see the distributions of the images that belong to the frogs class and images that belong to the trucks class. For the frogs class, we see that the values of uncertainty are concentrated in the higher band of the diagram, whereas in the case of trucks, we find many with lower uncertainty. This detail shows that the metric assigns significant uncertainty to out-of-sample class points.