Misclassification Risk and Uncertainty Quantification in Deep Classifiers

Murat Sensoy[1], Maryam Saleki[1], Simon Julier[2], Reyhan Aydogan[1], John Reid[3]
[1]Department of Computer Science, Ozyegin University, Turkey
[2]Department of Computer Science, University College London, UK
[3]Blue Prism AI Labs, UK

Abstract

In this paper, we propose risk-calibrated evidential deep classifiers to reduce the costs associated with classification errors. We use two main approaches. The first is to develop methods to quantify the uncertainty of a classifier's predictions and reduce the likelihood of acting on erroneous predictions. The second is a novel way to train the classifier such that erroneous classifications are biased towards less risky categories. We combine these two approaches in a principled way. While doing this, we extend evidential deep learning with pignistic probabilities, which are used to quantify uncertainty of classification predictions and model rational decision making under uncertainty.

We evaluate the performance of our approach on several image classification tasks. We demonstrate that our approach allows to (i) incorporate misclassification cost while training deep classifiers, (ii) accurately quantify the uncertainty of classification predictions, and (iii) simultaneously learn how to make classification decisions to minimize expected cost of classification errors.

1. Introduction

Deep neural networks have shown superhuman performance in many classification tasks. However, unlike humans, existing deep models do not take into account the consequences of their possible mistakes. This is evidenced by the choice of the cross entropy as the most common measure of loss in deep classifiers. The cross entropy loss is computed using the predicted probability of the correct category and completely ignores how the remaining probability mass is distributed over the wrong categories. Thus, the classifier is trained without regard to the risk of incorrect classification decisions.

An agent solely depending on the predictions of such classifiers to make a decision or take an action may pay a high cost when these prediction are wrong. A striking example is an incident happened on 7th May 2016, near Williston, Florida, USA. A car operating with automated vehicle control systems crashed into a truck. Unfortunately, the car driver died due to the sustained injuries. The car manufacturer stated that the accident originated from the vision system which incorrectly classified the white truck as a bright sky [16] and acted based on this erroneous prediction. As deep learning is used more widely and in more critical decision making operations, these difficulties will become more prevalent.

To incorporate the cost of misclassification into the training of deep classifiers, one approach is to use cost-sensitive learning [4]. This aims to minimize the expected cost of classification errors, e.g., by avoiding predictions placing high probabilities for high-risk categories, whilst maintaining classification accuracy. However, similarly to standard classifiers, cost-sensitive classifiers do not have any mechanism for quantifying the uncertainty of their predictions. Thus an autonomous agent using these classifiers cannot know if it can rely on their predictions to make a decision or take an action. Hence, the main advantage of these classifiers for the agent is limited to decreasing the cost of classification errors due to their tendency to predict less risky categories. On the other hand, if the agent is equipped with tools to quantify the uncertainty of these predictions, it can avoid taking actions based on ungrounded and most likely wrong predictions. Recently, a number of methods have been proposed to quantify uncertainty of deep classifier predictions. Among those, evidential deep learning is the state-of-the-art and a practical approach for uncertainty quantification in deep classifiers [18]. However, these methods do not take into account the risk of classification errors and may still be overconfident for some high-risk categories.

In this paper, we propose *risk-calibrated* evidential classifiers and made the following contributions:

- We reformulate evidential deep learning within a Bayesian learning framework and regularize the generated evidence using misclassification risk.
- We incorporate decision theoretical principles with deep neural networks through pignistic probabilities.
- We empirically show that we do not trade off accuracy

or uncertainty quantification for risk reduction.

The paper is organised as follows. Section 2 describes our approach for risk-calibrated deep classifiers. Section 3 evaluates our approach with respect to other methods using well-known datasets. Section 4 discusses the related work and the proposed approach. We conclude the paper with an overview of our contributions and findings in Section 5.

2. Risk-calibrated Classifiers

In a typical dataset for classification, each sample $\boldsymbol{x}$, (e.g., an image), has a label y representing its category. This label is related to a latent categorical distribution $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, which represents the probability that the sample belongs to each of K categories. Vanilla neural network classifiers aim to directly estimate $\boldsymbol{\pi}$ using the *softmax* function. We can quantify the uncertainty of this predictive distribution using statistics such as the (information theoretic) entropy, $\mathbb{H}[y|\boldsymbol{\pi}] = -\sum \pi_i \log \pi_i$. However quantifying uncertainty as $\mathbb{H}[y|\boldsymbol{\pi}]$ presupposes the classifier's estimate of $\boldsymbol{\pi}$ is precise.

In this work, we wish to construct neural networks that are aware of and can quantify confidence in their predictions. One natural (Bayesian) approach is to design classifiers that output probability distributions over $\boldsymbol{\pi}$ rather than point estimates and allow uncertainty estimation using these distributions. Since the Dirichlet distribution is a prior for the categorical distribution, it can be used as a distribution over all possible softmax outputs for the classification of any given sample.

2.1. Dirichlet Distributions and Pseudocounts

The Dirichlet distribution is a probability density function (pdf) for categorical distributions, $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$ over K categories. It is characterized by parameters $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]$ and is given by

$$\text{Dirichlet}(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \begin{cases} \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} \pi_i^{\alpha_i - 1} & \text{for } \boldsymbol{\pi} \in \mathcal{S}_K, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathcal{S}_K$ is the K-dimensional unit simplex and $B(\boldsymbol{\alpha})$ is the K-dimensional multinomial beta function [10].

Figure 1 illustrates several Dirichlet distributions over three categories with different choices of parameter values. When all the parameters are one (i.e., $\boldsymbol{\alpha} = [1, 1, 1]$), the Dirichlet distribution is uniform, i.e., all categorical distributions over these three categories are equally likely. As the parameter values become more unequal, the distribution becomes more concentrated.

The parameters of the Dirichlet distribution can be considered to be real-valued pseudocounts [15], where the higher the pseudocount, the more evidence exists for that category. This behaviour is formalised when the Dirichlet distribution is used as a conjugate prior for the categorical distribution. In this case, if the prior beliefs about a categorical distribution, $\boldsymbol{\pi}$, are represented as Dirichlet$(\boldsymbol{\pi}; \boldsymbol{\beta})$ and we observe evidence (a total of n pseudocounts),

$$\boldsymbol{c} = [c_1, \ldots, c_K] \sim \text{Multinomial}(n, \boldsymbol{\pi}),$$

then our belief about $\boldsymbol{\pi}$ is updated to Dirichlet$(\boldsymbol{\pi}; \boldsymbol{\beta}+\boldsymbol{c})$. For example, let $\boldsymbol{c} = [1, 4, 14]$ be the evidence to be added to the prior pseudocounts $\boldsymbol{\beta} = [1, 1, 1]$, then the posterior Dirichlet distribution will have the parameters $\boldsymbol{\alpha} = \boldsymbol{\beta} + \boldsymbol{c} = [2, 5, 15]$. This indicates that the categorical distributions which place more mass on the third category are more likely than others, as shown in Figure 1(b). Similarly, if $\boldsymbol{c} = [9, 9, 9]$, the resulting Dirichlet distribution parameters are $\boldsymbol{\beta} = [10, 10, 10]$, which indicates that the categorical distributions placing similar amount of mass on all categories become more likely, as shown in Figure 1(c).

The mean and variance of the probability of category k when $\boldsymbol{\pi}$ is distributed as Dirichlet$(\boldsymbol{\pi}; \boldsymbol{\alpha})$ are

$$\mathbb{E}[\pi_k] = \frac{\alpha_k}{\alpha_0}, \;\; \mathbb{V}[\pi_k] = \frac{\alpha_k(\alpha_0 - \alpha_k)}{\alpha_0^2(\alpha_0 + 1)}, \text{where } \alpha_0 = \sum_{i=1}^{K} \alpha_i.$$

Note that scaling each α_i by a constant greater than one leaves the mean the same but reduces the variance. This fits the pseudocount (evidence) interpretation of the α_i. For example, the predictive categorical distribution for the Dirichlet distributions with parameters $\boldsymbol{\alpha} = [1, 1, 1]$ and $\boldsymbol{\alpha} = [10, 10, 10]$ are both $[1/3, 1/3, 1/3]$ — the uniform categorical distribution with the maximum entropy. However, the variance of the latter is much smaller than that of the former, as also shown in Figures 1(a) and 1(c).

2.2. Evidential Deep Classifiers

Discriminative classifiers (e.g. logistic regression, vanilla neural network classifiers) [15] are models of the form $p(y|\boldsymbol{x})$ where a distribution over the K possible categories is estimated directly. The primary concept underlying *evidential classifiers* is to estimate how much evidence a sample holds for each category. By estimating the evidence we can estimate both $p(y|\boldsymbol{x})$ and reason about how confident we are in this prediction of $p(y|\boldsymbol{x})$.

We describe the underlying discriminative model here and show it in plate notation in Figure 2. Our model assumes that every input sample $\boldsymbol{x}$ has a weight of evidence $\boldsymbol{c} = [c_1, \ldots, c_K]$, where $c_i > 0 \; \forall i$ associated with it. The c_i are effectively pseudocounts representing how much evidence the sample holds for the i^{th} category. This evidence is combined with a prior in order to parameterise a distribution over a per-sample latent variable $\boldsymbol{\pi}|\boldsymbol{\beta}, \boldsymbol{c} \sim$ Dirichlet$(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = \boldsymbol{\beta} + \boldsymbol{c}$. Finally, the label y is sampled from $\boldsymbol{\pi}$. Note that whilst a typical discriminative classifier makes

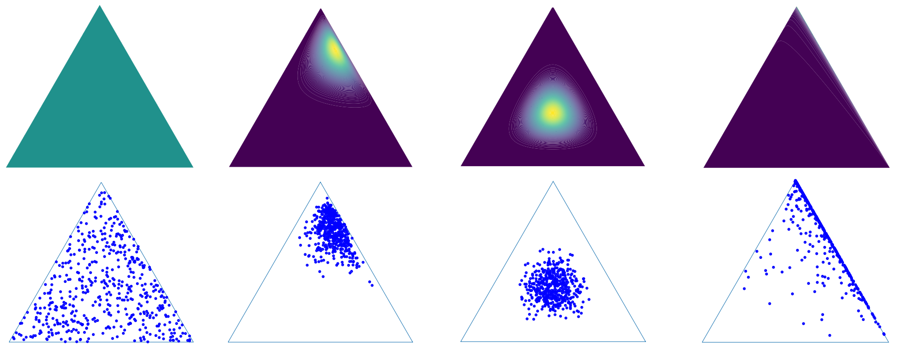


Figure 1. *Top*: The pdf of the Dirichlet$(\boldsymbol{\pi};\boldsymbol{\alpha})$ distribution over the probability simplex in $\mathbb{R}^3$ (blue = low, yellow = high). *Bottom*: 500 draws from the Dirichlet$(\boldsymbol{\pi};\boldsymbol{\alpha})$ distribution for various values (a, b, c, d) of the $\boldsymbol{\alpha}$ parameter.

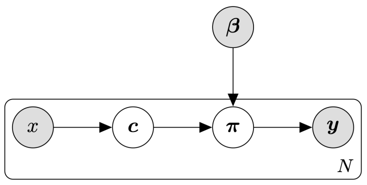


Figure 2. A graphical representation of discriminative evidential classifiers using plate notation.

a point estimate of $\boldsymbol{\pi}$ directly, an evidential classifier estimates a distribution over $\boldsymbol{\pi}$. Even though $\boldsymbol{\pi}$ is latent, we can marginalise over it to calculate the predictive distribution for the label exactly as $p(y=k|\boldsymbol{c},\boldsymbol{\beta}) = p(y=k|\boldsymbol{\alpha}) = \alpha_k/\alpha_0$. In general, the prior parameters $\boldsymbol{\beta}$ can take any values. In this work we set them to $[1,\ldots,1]$ to have the uniform Dirichlet distribution as the prior for $\boldsymbol{\pi}$. This induces a flat prior where each $\boldsymbol{\pi}$ is equally likely. For any given $\boldsymbol{x}$, the prior is combined with the evidence, $\boldsymbol{c}$, to estimate $\boldsymbol{\pi}$. Then, the entropy of the predictive categorical distribution can be computed as $\mathbb{H}[y|\boldsymbol{\alpha}] = -\sum_{i=1}^{K}(\alpha_i/\alpha_0)\log(\alpha_i/\alpha_0)$. If the evidence is low, the prior will dominate and the uncertainty in y will be high. If the evidence is high and concentrated in one category, the uncertainty will be low. Having a Dirichlet distribution as the output of a classifier instead of a single softmax output, we can exploit the uncertainty of the predictive categorical distribution to avoid making possibly wrong decisions based on vague predictions. In the next section, we describe how to predict parameters of Dirichlet distributions for classification.

2.3. Learning to Predict Evidence

We predict a Dirichlet distribution for each sample. The predicted Dirichlet distribution represents both the predictive categorical distribution and its uncertainty in a principled way. Let $\boldsymbol{f}_\theta(\cdot)$ be the output of the penultimate layer (i.e., logits layer) of any neural network for classification with arbitrary architecture, i.e., $\boldsymbol{z} = \boldsymbol{f}_\theta(\boldsymbol{x})$. Instead of using *softmax*$(\boldsymbol{f}_\theta(\boldsymbol{x}))$ to predict a single categorical distribution for the sample $\boldsymbol{x}$, we propose to use another activation function $a(\cdot)$ to calculate $\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}) = a(\boldsymbol{f}_\theta(\boldsymbol{x}))$, which will then be interpreted as evidence to update the prior counts. For any input $-\infty < z_i < \infty$, this activation function should take values $0 < a(z_i) < \infty$. We used the *exponential* function, i.e., $a(z_i) = e^{z_i}$; however, others such as the *softplus* function $\log(e^{z_i}+1)$ can also be used. For any input sample $\boldsymbol{x}$, having total predicted evidence very close to zero implies that the resulting distribution $p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}),\boldsymbol{\beta})$ is very similar to the uniform Dirichlet distribution, so we have a very *uncertain* predictive categorical distribution for the sample.

We can use maximum likelihood estimation (MLE) to train the neural network to predict an evidence vector for each sample $\boldsymbol{x}$. For this purpose, we select a base loss function $\mathcal{L}_{\text{base}}(y|\boldsymbol{\pi})$ conditioned on the latent categorical distribution $\boldsymbol{\pi}$; then, we calculate the overall loss by aggregating individual sample losses and marginalising out the latent $\boldsymbol{\pi}$ values using the predicted Dirichlet distributions,

$$\mathcal{L}(\theta) = \sum_{\langle \boldsymbol{x},y\rangle \in D} \mathbb{E}_{p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}),\boldsymbol{\beta})}[\mathcal{L}_{\text{base}}(y|\boldsymbol{\pi})].$$

If the *cross-entropy* is used, then $\mathcal{L}_{\text{base}}(y|\boldsymbol{\pi}) = -\log \pi_y$ and

$$\mathcal{L}(\theta) = \sum_{\langle \boldsymbol{x},y\rangle \in D} \Big[\psi\Big(\sum_{i=1}^{K}\big(\hat{c}_{\theta i}(\boldsymbol{x})+1\big)\Big) - \psi\big(\hat{c}_{\theta y}(\boldsymbol{x})+1\big)\Big] \tag{1}$$

where ψ is the digamma function [18]. Without any regularisation, the network may generate high pseudocounts to maximize the likelihood. We prevent this during training by encouraging the network to reduce the counts for incorrect categories via an additional loss term. This loss term uses the Kullback–Leibler (KL) divergence between the predicted Dirichlet distribution and a target (or desirable) Dirichlet distribution. This target distribution agrees with the predicted Dirichlet distribution for the correct category, but places zero evidence in all other categories. For a sample $\boldsymbol{x}$ and its label y, the regularization term is written as

$$\mathrm{KL}[p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x})+\mathbf{1}) \parallel p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}) \odot \boldsymbol{hot}(y)+\mathbf{1})],$$

where $\boldsymbol{hot}(\cdot)$ is the one-hot encoding function for K categories, $\odot$ represents element-wise product, and $\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}) \odot \boldsymbol{hot}(y)+\mathbf{1}$ are the parameters of the target distribution.

2.4. Decision Making under Uncertainty

Agents usually use classifiers when they need to choose one option among several alternatives during decision making. Pignistic probabilities have been introduced in decision theory to represent the probability that a rational agent will choose a particular option when it is required to make a decision. The pignistic probabilities $\boldsymbol{p}=[p_1,\ldots,p_K]$ are mathematically equivalent to the Shapley value in game theory [3] and inherently incorporate the decision maker's uncertainty when choosing one of K options (i.e., the uncertainty related to $\boldsymbol{\pi}$) and the incurred risk of choosing each one. Hence, while calculating the expected risk of choosing one category as the label of a sample, the pignistic probabilities ($\boldsymbol{p}$) replace the categorical probabilities ($\boldsymbol{\pi}$) to account for the risk of misclassification. In the settings where there is no risk for misclassification or each misclassification has the same risk, $\boldsymbol{p}$ should be equal to $\boldsymbol{\pi}$. However, in our setting, we have an explicit risk matrix, which may lead to some divergence between these two probabilities.

We choose to model the pignistic probabilities for the classification of the sample $\boldsymbol{x}$ as a Dirichlet distribution

$$q_\theta(\boldsymbol{p}|\boldsymbol{x}) = \mathrm{Dirichlet}(\boldsymbol{p}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x})+\boldsymbol{\gamma}_\theta(\boldsymbol{x})),$$

where we replace the prior $\boldsymbol{\beta}$ by $\boldsymbol{\gamma}_\theta(\boldsymbol{x})$. This is the prior count for the pignistic probabilities for the sample and is a function of $\boldsymbol{x}$. The reason we do this is to model the fact that the risk of misclassification may change from sample to sample (e.g., based on its true category). By doing so, we let the neural network learn the less risky categories and increase its prior count through the softmax output.

The term has the form $\boldsymbol{\gamma}_\theta(\boldsymbol{x}) = K\mathrm{softmax}(\boldsymbol{W}\boldsymbol{f}'_\theta(\boldsymbol{x})+\boldsymbol{b})$, which is a per-sample redistribution of uniform prior counts over K categories. $\boldsymbol{W}$ and $\boldsymbol{b}$ are additional weight and bias variables, respectively, to calculate $\boldsymbol{\gamma}_\theta(\boldsymbol{x})$ based on $\boldsymbol{f}'_\theta(\boldsymbol{x})$, which represents the input of the *logits layer* of the neural network. Note that we still have $\sum_i \gamma_{\theta i}(\boldsymbol{x}) = K$. Hence, the total prior counts for the pignistic probabilities $\boldsymbol{p}$ is same as that of the categorical probabilities $\boldsymbol{\pi}$. This allows us to keep the evidence predicted by the network for the categorical probabilities (i.e., $\hat{\boldsymbol{c}}_\theta(\boldsymbol{x})$) to be commensurate with the prior counts for the pignistic probabilities. Figure 4 overviews the network architecture used in our approach.

Figure 1(d) shows the resulting Dirichlet distribution after redistributing the counts in the uniform Dirichlet distribution in Figure 1(a), using $[0.033, 0.3, 0.667]$ as the softmax output to calculate $\boldsymbol{\gamma}_\theta(\boldsymbol{x})$. The resulting Dirichlet distribution generates almost no probability mass for the first category while placing more probability mass on the third category. This is a desired prior for the pignistic probabilities of a sample if the misclassification risk for the sample is inversely proportional to the redistributed counts.

Given its pignistic probabilities, the average risk of misclassifying $\boldsymbol{x}$ from category y is $\mathrm{risk}(\boldsymbol{x}) = \sum_{i=1}^K R_{yi}p_i$, where R_{yi} indicates the risk or cost of misclassifying a sample from category y to the category i. These risk values are stored in an asymetric non-negative square matrix $\boldsymbol{R} \in [0,\infty)^{K\times K}$ such that $R_{yi} \geq R_{yy} = 0$. We can integrate out the pignistic probabilities using their Dirichlet distribution parametrized by $\boldsymbol{\alpha}$ with $\alpha_i = \hat{c}_{\theta i}(\boldsymbol{x}) + \gamma_{\theta i}(\boldsymbol{x})$, to compute the expected risk

$$\mathbb{E}[\mathrm{risk}(\boldsymbol{x})] = \mathbb{E}_{q_{\theta}(\boldsymbol{p}|\boldsymbol{x})}\left[\mathbb{E}_{i\sim\boldsymbol{p}}[R_{yi}]\right] = \frac{\sum_{i=1}^K R_{yi}(\hat{c}_{\theta i}(\boldsymbol{x})+\gamma_{\theta i}(\boldsymbol{x}))}{K+\sum_{j=1}^K \hat{c}_{\theta j}(\boldsymbol{x})}$$

To minimize the risk of misclassification and increase the uncertainty for the misclassified samples, we combine the loss introduced before with the expected risk of misclassification

$$\begin{aligned}\mathcal{L}_{total}(\theta) = \sum_{\langle \boldsymbol{x},y\rangle\in D} \Big[&\psi\big(\sum_{i=1}^K \big(\hat{c}_{\theta i}(\boldsymbol{x})+1\big)\big) - \psi\big(\hat{c}_{\theta y}(\boldsymbol{x})+1\big) \\ &+ \mathrm{KL}[p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x})+\mathbf{1}) \parallel p(\boldsymbol{\pi}|\hat{\boldsymbol{c}}_\theta(\boldsymbol{x}) \odot \boldsymbol{hot}(y)+\mathbf{1})] \\ &+ \kappa \sum_{i=1}^K \big(\underbrace{R_{yi}\hat{c}_{\theta i}(\boldsymbol{x})}_{(1)} + \underbrace{R_{yi}\gamma_{\theta i}(\boldsymbol{x})}_{(2)}\big)\Big],\end{aligned} \quad (2)$$

where $\kappa \in [0,1]$ is the weight of the expected risk in the loss. In this loss, we neglect the denominator of the expected risk to prevent the neural network from producing high evidence for less risky categories when it cannot predict the correct category. By excluding the denominator in the loss, the term (1) in Equation 2 acts as a risk-based regularizer for the generated evidence. During training, it regularizes the evidence and discourages network to reduce the expected risk by generating evidence for less risky wrong categories. This is similar to learning evidence using maximum a posteriori (MAP), instead of maximum likelihood estimation. Furthermore, the term (2) in the loss serves as an objective to optimize prior counts for the pignistic probabilities, which influence classification decisions only when the total evidence is very low, so the predictive uncertainty is high. Note that the evidence regularization in the total loss is distinct from L_1 or L_2 regularization methods, which are used to regularize network parameters, not to directly regularize the output of the network. Through the redistributed prior counts of the pignistic probabilities, the loss function reduces the expected risk for misclassification when $\boldsymbol{\pi}$ is highly uncertain. If the categorical probabilities were used to calculate the risk of misclassification, the expected risk in the loss would encourage more evidence for the wrong categories with lower risk to reduce the expected risk when the correct category cannot be predicted correctly. However, the formulation of pignistic probabilities encourages zero evidence for the wrong categories if the redistributed prior counts reduce the expected risk sufficiently.

3. Evaluation

In this section, we evaluate how agents may use our approach (i) to minimize their risk when making classification decisions based on the predictions of deep neural networks and (ii) to determine if these predictions are wrong based on their uncertainty. For this purpose, we use the LeNet5 neural network architecture [13] with three well-known datasets: MNIST[1], FashionMNIST[2], and CIFAR10[3].

Our approach is very general and can be used for an arbitrary deep neural network for classification. However, in our evaluations, we choose LeNet5 as a neural network architecture, because it has been used as a benchmark recently to evaluate uncertainty quantification in deep neural networks [18, 14]. It is a simple convolutional neural network architecture, which cannot achieve the state-of-the-art performance in terms of accuracy. It fits very well to our purpose, since we aim to analyse classification errors of neural networks in our evaluations. We desire to equip even simple neural networks with the ability of estimating their uncertainty when making mistakes and choosing less risky categories when they are uncertain. The configuration of the network architecture used for MNIST and FashionMNIST is presented in Table 1. For CIFAR10, we used the same architecture with 192 filters for Conv_1 and Conv_2, and 1000 neurons in FC_1 as described in [14]. We used L2 regularization coefficient 0.005 in the fully-connected (FC) layers.

We compare our approach with three approaches: (i) *standard* learning uses *cross-entropy* loss; (ii) *cost-sensitive* learning [9] incorporates the risk matrix into the standard loss to minimize the cost of misclassifications; and (iii) evidential deep learning (EDL) [18] uses the regularized loss in Equation 1. Let us note that [18] introduces three different loss functions while we use only one of them in this work. Other loss functions in [18] can also be incorporated with our approach. We name our model as *risk-calibrated evidential deep learning*, which is referred to as *Risk EDL*. It uses the expectation of the predicted Dirichlet distribution for pignistic probabilities, i.e., $q_\theta(\boldsymbol{p}|\boldsymbol{x})$, as the predictive distribution for classification. All these models uses the same LeNet5 architecture, but with different loss functions.

In our experiments, we used two different matrices for $\boldsymbol{R}$

Table 1. LeNet5 architecture for MNIST and FashionMNIST.

Layer	Filters/Neurons	Patch Size	Stride	Activation
Conv_1	20	5×5	1	ReLU
Max Pool	—	2×2	2	—
Conv_2	50	5×5	1	ReLU
Max Pool	—	2×2	2	—
FC_1	500	—	—	ReLU
FC_2	10	—	—	—

Table 2. Misclassification cost

Model	MNIST	Fashion	CIFAR10
Cross-entropy	11.16	8.42	5.8
Cost-sensitive	5.9	6.1	4.7
EDL	11.0	10.04	6.1
Risk EDL	3.44	3.48	3.54

as shown in Figure 3 (a) and (d), and set $\kappa = 1/\max(\boldsymbol{R})$. For MNIST and FashionMNIST, we used the category indices to create the risk matrix. Let i refers to the true category index of a sample and j refers to the index of the predicted category for the sample. We set $R[i,j] = (i-j)^2$ if $j > i$; otherwise, it is set to $i - j$. For MNIST, this risk matrix implies that overestimating the value of the digit in an image incurs much higher cost than under estimating it. For CIFAR10, we divided the categories into two groups: *animals* and *vehicles*. In this case, we set $R[i,j] = 1$ if i and j are from the same group. If the true index i is animal, but the predicted index j is not, we set $R[i,j] = 10$; otherwise, we set it to 50 as shown in Figure 3(d).

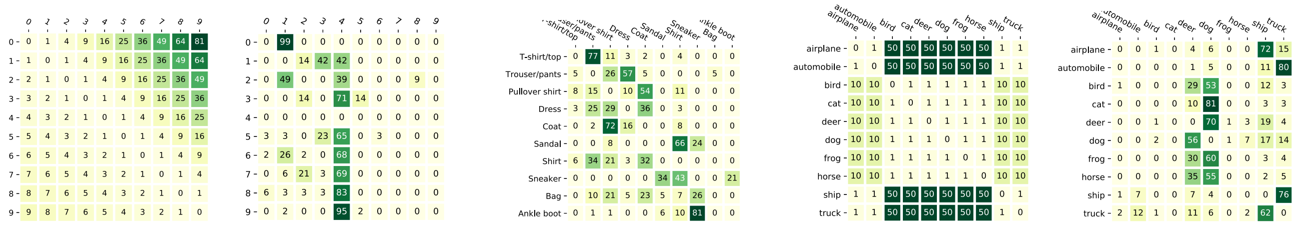
MNIST and FashionMNIST datasets contain 60000 and 10000 training and test samples, respectively, while the number of training and test samples are 50000 and 10000 for CIFAR10 dataset. We train the models for 50 epochs and evaluated the trained models using the test samples. Table 2 and 3 present our results for the average misclassification cost and accuracy on the test samples, respectively. Our results indicates that cost-sensitive training achieves lower misclassification cost than standard or evidential deep learning. This is reasonable, since these methods do not consider the risk matrix during training. Our model achieves significantly lower misclassification cost than the cost-sensitive training, while also enhancing the accuracy of the model on the test samples in all datasets. That is, when compared with cost-sensitive training, our approach reduces the cost of misclassification by 42%, 43%, and 25%, for MNIST, FashionMNIST, and CIFAR10, respectively.

An important problem that should also be addressed is the detection of misclassified samples. If an agent can detect misleading classification predictions of a classifier (instead of blindly following these predictions), it can actively reduce the cost of misclassification by avoiding making classification decisions when possible. In the literature, for uncertainty quantification of deep classifiers [14], the entropy of their

[1]In MNIST, samples are 28×28 images of handwritten digits $0 - 9$.

[2]FashionMNIST similar to MNIST but contains fashion products.

[3]In CIFAR10, samples are low-resolution images of 10 animal and vehicle categories.



(a) MNIST risk mat. (b) MNIST misclas. % (c) FashionMNIST misclas.% (d) CIFAR10 risk mat. (e) CIFAR10 misclas.%

Figure 3. The risk matrices in the experiments and the percentage of misclassification for each category for the proposed approach.

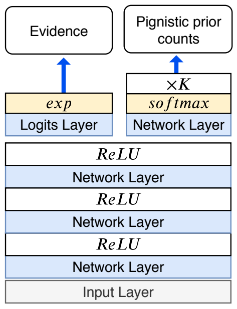
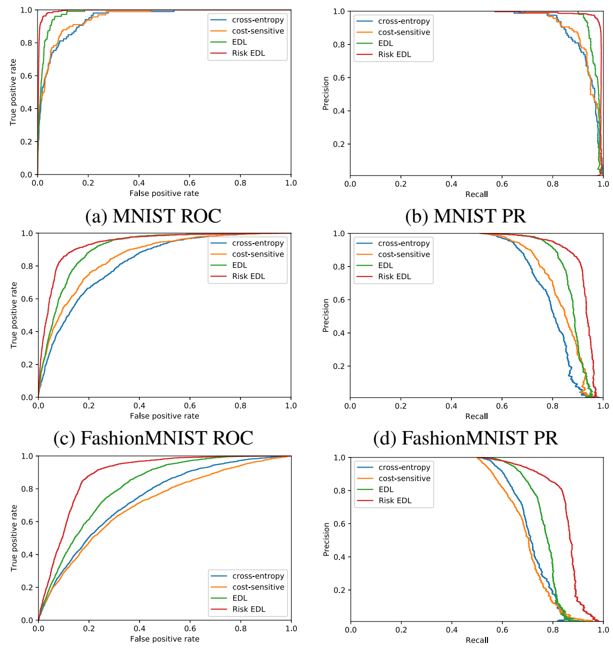


Figure 4. The network architecture of the proposed approach.



(a) MNIST ROC (b) MNIST PR (c) FashionMNIST ROC (d) FashionMNIST PR (e) CIFAR10 ROC (f) CIFAR10 PR

Figure 5. ROC and PR curves for misclassification detection.

prediction are used as a proxy for their uncertainty. This metric for uncertainty is very general and can be used for any neural network, which provides a predictive categorical distribution for classification. Hence, to be able to compare them fairly, we also evaluate different approaches using the entropy of their predictions and measure how well we can separate the misclassified and correctly classified samples using the entropy. Ideally, when it is unable to correctly classify a sample, a model should say *I do not know* by providing a very uncertain prediction (i.e., a prediction with maximum entropy). We used the area under the Precision-Recall (PR) and Receiver Operating Characteristic (ROC) curves[4] to measure the successful detection of misclassified samples using the entropy of predictions for these samples. Figure 5 shows PR and ROC curves for misclassification detection. The area under curve (AUC) for PR and ROC curves are presented in Table 4 and 5 and indicate that our approach allows much better misclassification detection and separation of misclassified samples from others for all datasets.

Table 3. Test accuracy

Model	MNIST	Fashion	CIFAR10
Cross-entropy	99.0	90.0	73.0
Cost-sensitive	99.1	90.0	73.1
EDL	98.8	89.4	73.6
Risk EDL	99.3	91.3	75.0

Table 4. Misclass. (PR AUC)

Model	MNIST	Fashion	CIFAR10
Cross-entropy	0.938	0.79	0.71
Cost-sensitive	0.942	0.83	0.69
EDL	0.967	0.87	0.76
Risk EDL	0.992	0.92	0.85

Table 5. Misclass. (ROC AUC)

Model	MNIST	Fashion	CIFAR10
Cross-entropy	0.945	0.82	0.71
Cost-sensitive	0.945	0.85	0.71
EDL	0.979	0.90	0.81
Risk EDL	0.996	0.94	0.85

Figure 3 (b) and (c) demonstrate misclassification matrices of our approach for the MNIST and FashionMNIST datasets. The rows of this matrix represent the true category indices and columns represent indices of the predicted cate-

[4]ROC curves are not suitable for analysing imbalanced data. That is why we used oversampling to balance the data before computing the ROC curves.

Table 6. OoD (PR AUC)

Model	MNIST	Fashion	CIFAR10
Cross-entropy	0.975	0.77	0.76
Cost-sensitive	0.965	0.80	0.76
EDL	0.970	0.91	0.98
Risk EDL	0.993	0.97	0.98

Table 7. OoD (ROC AUC)

Model	MNIST	Fashion	CIFAR10
Cross-entropy	0.972	0.78	0.78
Cost-sensitive	0.961	0.81	0.78
EDL	0.97	0.89	0.99
Risk EDL	0.993	0.96	0.99

gories. The value at index $[i, j]$ represents what percentage of the misclassified samples from category i are predicted as belong to category j. Similarly, Figure 3 (e) represents the misclassification matrix for CIFAR10. These matrices indicate how our approach avoid classifying samples to high-risk categories and explain lower cost of misclassification for our approach.

Recent studies indicate that deep classifiers are very vulnerable to out-of-distribution (OoD) samples [7]. In other words, the deep classifiers are usually very confident when they classify samples that are not from their training set distribution. Ideally, classifiers should give a very uncertain prediction in such settings. If the entropy is taken as a measure of uncertainty, their predictions should be a uniform probability distribution over possible categories, which indicates highest predictive uncertainty. To evaluate our approach for the predictive uncertainty for OoD samples, we use samples from the notMNIST and CIFAR100 datasets. The notMNIST dataset [14] contains images of letters from A to J on various typefaces. It is used to evaluate classifiers trained with MNIST and FashionMNIST datasets. CIFAR100 contains images similar to images from CIFAR10, but from 100 different categories. We used it to evaluate classifiers trained with CIFAR10 dataset. Figure 6 shows the empirical entropy CDF of the predictions for the OoD samples for each dataset. An ideal classifier should give the near maximum entropy predictions for OoD samples, so the entropy CDF curve should be very close to the bottom right corner. The figure indicates that our approach is closer to the ideal; the entropy of its predictions for OoD samples are much more closer to the maximum entropy than others.

We also compare the uncertainty of in- and out-of-distribution samples to see how easy to separate these two using only the predictive uncertainty. For this purpose, we use predictions for the correctly classified samples from the training distribution, i.e., in-distribution samples, and predictions for the OoD samples. We calculate their entropy and use the PR and ROC curves to see how easy to separate these two using only the entropy of predictions. Our results are shown in Figure 7, which clearly indicates that our approach allows much better separation between in-distribution and out-of-distribution samples using the predictive uncertainty. We also summarized our results also in Table 6 and 7 for all datasets.

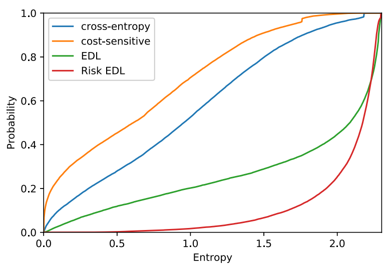
In this section, we evaluated our approach and showed that (i) it allows an agent to minimize the cost of misclassification by learning and exploiting pignistic probability distributions, and (ii) it allows an agent to detect misclassified and out-of-distribution samples through predictive uncertainty.

4. Related Work

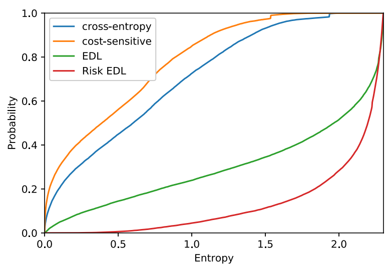
Machine learning and especially classifiers are frequently used in different fields and applications of autonomous and multiagent systems. Deep neural networks and deep learning in general are becoming a fundamental component in intelligent systems and already deployed in autonomous systems such as self-driving vehicles. However, recent studies indicates that these methods are very confident when they do mistakes and fail severely when exposed to input different from what they are trained for. That is why uncertainty quantification in deep learning stands as an important problem.

Bayesian deep learning [14, 8, 5, 12, 17] has emerged as a field combining deep neural networks with Bayesian probability theory, which provides a principled way of modelling uncertainty of machine learning models by employing prior distribution on their parameters and inferring the posterior distribution for these parameters using approximations such as Variational Bayes [1, 5, 19]. Then, the posterior predictive distribution is approximated with sampling methods, which brings a significant computational overhead and leads to noise in predictive uncertainty estimates. In these models, predictive uncertainty is modelled by taking samples from the posterior distributions of model parameters and using the sampled parameters to create a distribution of predictions for each input of the network. However, modelling uncertainty of network parameters may not necessarily lead to good estimates of the predictive uncertainty of networks [6].

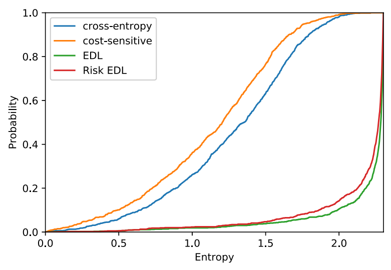
Recently, a number of approaches, such as evidential deep learning [18], have been proposed to use the outputs of neural networks to estimate the parameters of Dirichlet distributions for classification; instead of predicting a categorical distribution through the softmax function. Then, the resulting Dirichlet distribution is used to calculate the predictive uncertainty for classification. These approaches are more practical since they do not require sampling methods and are easier to implement, but also disregard the different risk of misclassification for different categories. In this paper, we extend evidential deep learning by incorporating the risk of misclassification into its loss function in a principled way. We use the risk of incorrectly classifying samples from



(a) MNIST

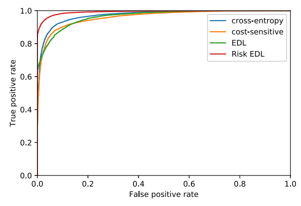


(b) FashionMNIST

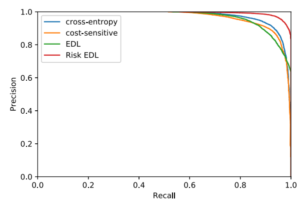


(c) CIFAR10

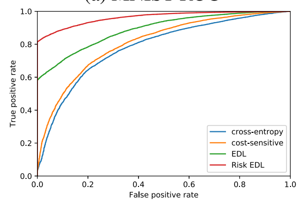
Figure 6. Empirical entropy CDFs of the predictions for the out-of-distributions samples.



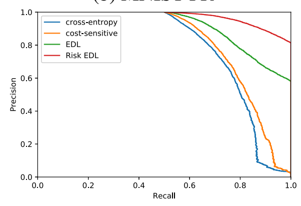
(a) MNIST ROC



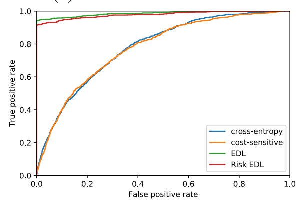
(b) MNIST PR



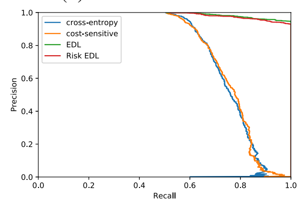
(c) FashionMNIST ROC



(d) FashionMNIST PR



(e) CIFAR10 ROC



(f) CIFAR10 PR

Figure 7. ROC and PR curves for out-of-distribution detection.

a category to regularize the generated evidence and avoid fictitious evidence.

Cost-sensitive learning aims training models for the settings where different misclassification errors incur different penalties [4, 11]. The standard learning techniques try to achieve high classification accuracy while cost-sensitive learning has a joint objective of minimizing the cost of misclassification in addition to achieving high accuracy. Integration of cost-sensitive learning with deep neural networks for classification has not been studied well. Chung *et al.* has extended this approach for deep neural networks to create cost-sensitive deep classifiers [2]. Khan *et al.* proposed to incorporate cost (or risk) of misclassification directly to the cross-entropy loss by weighting the terms in the softmax function using the entries from the risk matrix [9]. None of these methods consider the uncertainty of classification predictions. On the other hand, in this paper, we do not only decrease the overall cost of incorrect predictions, but also increase the uncertainty of wrong predictions so that these predictions can easily be discriminated from the correct predictions. Hence, our approach allows a more accurate detection of classification errors and creates an opportunity for avoiding their undesirable consequences.

We would like to extend our approach in future to enhance robustness of neural networks against adversarial perturbations, which are engineered to result in the maximum damage through misclassification of samples into high-risk categories. Recently, Zhang *et al.* proposed to combine certified robustness against adversarial examples with the risk of misclassification [20] and compared it with cost-sensitive learning of [9] as we do. Certified robustness used in their work is based on linear programming and guarantees that small perturbations of samples will not change the predictions of their categories. However, it is applicable only to very simple neural network architectures and low-dimensional classification tasks.

5. Conclusions

As a result of striking success of deep learning in recent years, deep classifiers are now an indispensable part of autonomous systems. However, these black-box models may be very confident when their predictions are wrong and lead agents to make mistakes in their decisions. Furthermore, standard training of deep models neglects that different mistakes involve in different level of risk for the agents depending them. In this paper, we significantly extend the evidential deep learning to address this problem. Our approach allows agents to incorporate their own risk for classification mistakes into the training of evidential classifiers [18]. By incorporating the notion of risk and pignistic probabilities into the loss function, our approach improves the uncertainty quantification of evidential networks and, at the same time, significantly minimizes the cost of misclassification for autonomous agents.

Acknowledgments

Dr. Sensoy thanks to the U.S. Army Research Laboratory for its support under grant W911NF-16-2-0173.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wiestra. Weight uncertainty in neural networks. In *ICML*, 2015.

[2] Yu-An Chung, Hsuan-Tien Lin, and Shao-Wen Yang. Cost-aware pre-training for multiclass cost-sensitive deep learning. *arXiv preprint arXiv:1511.09337*, 2015.

[3] Didier Dubois, Henri Prade, and Philippe Smets. A definition of subjective possibility. *International Journal of Approximate Reasoning*, 48(2):352–364, 2008.

[4] Charles Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[5] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[6] Danijar Hafner, Dustin Tran, Alex Irpan, Timothy Lillicrap, and James Davidson. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv preprint arXiv:1807.09289*, 2018.

[7] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction. *arXiv preprint arXiv:1910.09457*, 2019.

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

[9] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.

[10] S. Kotz, N. Balakrishnan, and N.L. Johnson. *Continuous Multivariate Distributions*, volume 1. Wiley, New York, 2000.

[11] Matjaz Kukar, Igor Kononenko, et al. Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449, 1998.

[12] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

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

[14] C. Louizos and M. Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *ICML*, 2017.

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

[16] NHTSA. Pe 16-007. technical report, u.s. department of transportation, national highway traffic safety administration, jan 2017. tesla crash preliminary evaluation report. 2016.

[17] Nick Pawlowski, Andrew Brock, Matthew CH Lee, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.

[18] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3179–3189. 2018.

[19] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

[20] Xiao Zhang and David Evans. Cost-sensitive robustness against adversarial examples. In *International Conference on Learning Representations*, 2019.