Empirically Calibrating Neural Network Confidence under Distribution Shift

Thomas Larsen

April 1, 2022

1 Introduction

Deep Neural Networks (DNNs) have seen widespread success in a large variety of applications including image classification [1], medical diagnoses [2], and even autonomous vehicles [3]. With sufficient training data from the source distribution, DNNs can provide remarkably accurate predictions. Unfortunately, in many different real life applications, test data is often drawn from a different distribution than that of the original training set. We call these test points *out-of-distribution* (OOD), and in this case, we say that there is a *distribution shift*. DNNs tend to generally perform poorly on OOD test data [4, 5]. For example, an autonomous car trained only by driving on roads in good weather conditions will often perform poorly in the face of new unseen weather conditions, e.g., rain or snow [6].

Worse, many classifiers report high confidence for incorrect predictions [7]. This overconfidence raises serious concerns for the application of DNNs to real world systems [8]. Moreover, many systems lack proper uncertainty quantification [9]. In particular, uncertainty is important in safety-critical applications (e.g. the autonomous car in an unfamiliar locale) because low predictive confidence allows a control system to self-suspend or defer to human feedback. One metric of uncertainty is *calibration*, which is how well the model's predicted probabilities correspond to the empirical frequency of the model being correct. Under a distribution shift, probabilistic predictions usually cease to become calibrated [10]. In the most concerning cases, this results in inaccurate predictions with high confidence attached to them.

We propose a novel method to improve the calibration of neural networks under distribution shift. This method involves including an expected calibration error (ECE) term in the loss function in addition to a normal cross entropy term. We implement and evaluate this method on a variety of distribution shifts using the MNIST and CIFAR datasets as our training sets. We create a custom scale to measure the intensity of different shifts with intensities ranging from $1 - 10$. We give an example of such a shift on MNIST in Figure 1. The proposed solution significantly outperforms a baseline model trained without explicit uncertainty quantification. We evaluate these models on each shift and each intensity. On CIFAR-10, our method achieves an overall average of .065 test ECE, significantly outperforming the baseline model, which achieves .234 ECE.

2 Prior Work

2.1 Pointwise vs Distributional approaches

There are different ways of handling distribution shifts.

- The **pointwise approach** involves classifying a specific point as OOD if it is sufficiently distinct from the points in the training set. We call such a point an **outlier**.
- The **distributional approach** involves designing neural networks that can handle points sampled from a distribution that differs from the training distribution.

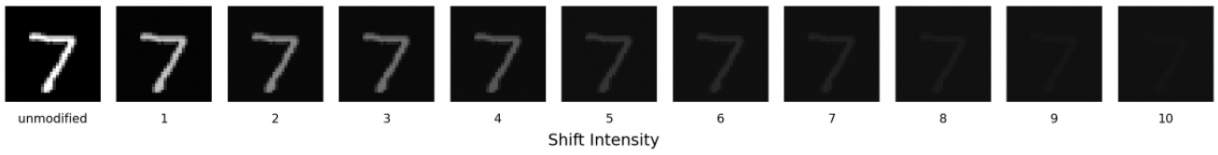

Figure 1: An example of a distribution shift (contrast) applied to an MNIST image

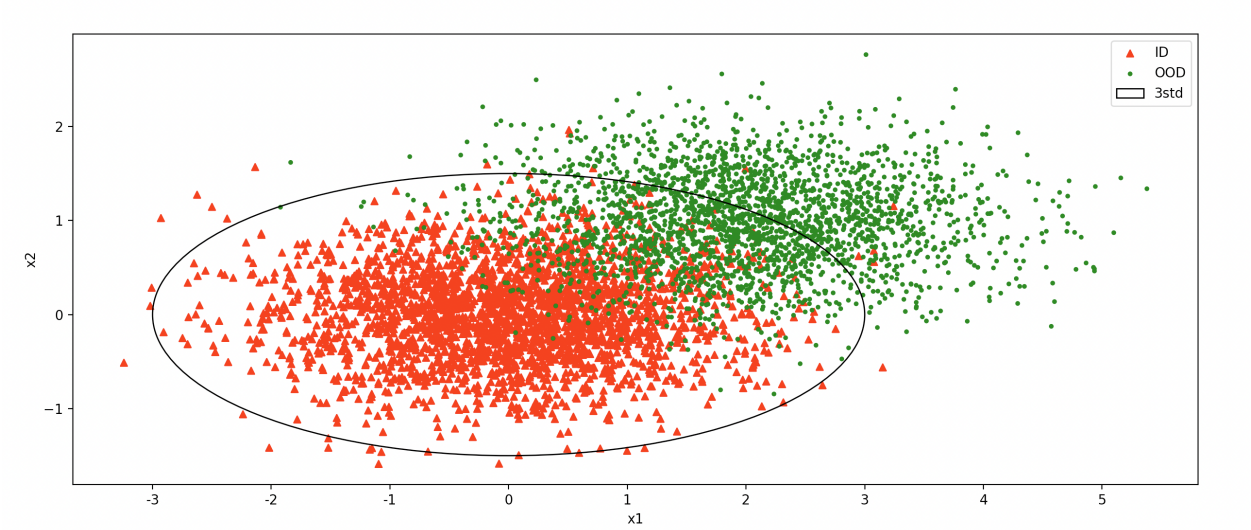

Figure 2: An example of a simple distribution shift

In Figure 2 we show a simple example of points sampled from different distributions:

– The in-distribution (ID) points are sampled from $x_{in} \sim \mathcal{N}\left(\mu = \begin{bmatrix}0\\0\end{bmatrix}, \Sigma = \begin{bmatrix}1 & 0\\0 & .5\end{bmatrix}\right)$

– The out-of-distribution (OOD) points are sampled from $x_{out} \sim \mathcal{N}\left(\mu = \begin{bmatrix}2\\1\end{bmatrix}, \Sigma = \begin{bmatrix}1 & 0\\0 & .5\end{bmatrix}\right)$

For an example to illustrate the difference between pointwise and distributional approaches, suppose that we train a model with the plotted training dataset sampled from P_{in}, along with associated labels. The pointwise approach involves learning a decision boundary separating ID and OOD points. In this example, we draw the 3-standard deviation boundary away from the center of our training distribution. Points within the interior of this ellipse are predicted as ID, and anything outside of this boundary are considered outliers.

However, classifying these as pointwise does not align to the distributional viewpoint. The OOD points are all sampled from a quite different distribution: P_{out}, but only about half of these datapoints, when viewed pointwise, are outliers. When viewed from the distributional perspective, all of these points are OOD, even though many of them are within the boundary of the ellipse.

In more realistic applications, we do not have access to the underlying distributions – we only have access to the dataset sampled from them. In addition, these are often very high dimensional samples, for example even a small image might have $32 \times 32 \times 3 = 3072$ dimensions. Thus, even a large dataset will be a sparse sampling of this space, meaning that in these contexts, we have a poor understanding of the underlying distribution that images are sampled from.

The OOD Classification approach in Section 2.2 takes a pointwise approach to this problem. These methods involve creating a classifier that detects and discards OOD images. On the other hand, both OOD Robust prediction (Section 2.3) and OOD Calibrated Uncertainty (Section 2.4) follow the distributional approach, because they do not commit to labeling a single point as OOD or ID. These approaches intentionally allow for OOD images to be input to the model, and attempt to train the model to perform well in these situations. We derive inspiration from the methods used in the pointwise approach to improve upon the uncertainty of models under distribution shift.

2.2 OOD Classification

Initial work in this area primarily used a pointwise approach to detect and discard OOD test images. In this paradigm, one limits the model scope such that the model only guarantees performance on data that is similar to the train distribution. Then, a classifier (often using the original network) is used to detect if a point is OOD.

Hendrycks et al. showed that for common image classifiers, the predicted probability for the correct class in test images under extreme shifts tends to be lower than for unshifted test images [11]. The authors used a threshold on the maximum of the class-specific probabilities reported by the classifier as a measure of how likely a given sample is to be OOD. Lee et al. trained a CNN to perform prediction for in-distribution samples and to be uncertain on OOD samples by using a generative adversarial network (GAN) to generate OOD samples and a custom loss function explicitly forcing the network to predict close to a uniform distribution on OOD samples [7]. On these OOD samples, they minimized the KL-divergence of the predicted distribution from the uniform distribution. This works well when the shift is extreme enough that there is no meaningful labeling of input points.

Using the same maximum of class-specific probabilities as Hendrycks et al., this new loss function significantly improved the capability to detect OOD test images. Using a similar approach, Papadopolous et al. defined a loss function with two regularization terms. The first term was the KL-divergence between predictions on OOD data and the uniform distribution, and the second term was the difference of the average confidence and the accuracy over the training set. Their technique achieves state of the art (SOTA) results in OOD detection without requiring access to OOD samples and even works in conjunction with other OOD detection methods such as Mahalanobis Detector and Gramian Matrices [12].

This type of classification takes a pointwise approach and relies on classifying points that are sufficiently different than training data as out of distribution. With sufficiently aggressive classification, this can solve some of the problems, because if inputs are detected as OOD, the network can decline to classify. However, this has the downside that the model does not give any information about OOD points. Thus, a user must instead rely on an alternative process such as deferring to human feedback, which is a time and resource intensive process that is often intractable. Additionally, if the classifier is not sufficiently aggressive in classifying OOD input points, the network will be exposed to OOD data and will thus perform poorly. For example, in Figure 2, the model may perform substantially worse on points from the shifted distribution that are pointwise ID (the green points within the 3σ boundary) than the ID points, because these points follow a different distribution than the training distribution.

2.3 OOD Robust Prediction

Another section of work on distribution shifts has tried to focus on making predictions that are accurate even on a collection of OOD data.

The noisy student method involves iterative teacher-student training on EfficientNet models. An ImageNet dataset of 1,000 categories and 150,000 images is used to train an initial teacher model. Then the teacher pseudolabels 300 million images (either one-hot or continuous) and is used to train a "student" model of equal or greater size, while introducing generated noise into the training set [13]. This methods significantly improves accuracy under moderate shifts. However, under sufficiently intense shifts, accuracy will inevitably drop. For example, the signal on the OOD set could be reversed from what it was on the ID set. For a simple example of this, consider performing regression to learn the function $f(x) = |x|$. However, our training samples are only drawn from the interval [0, 1]. If we try to extrapolate to negative numbers, a natural prediction (even from a very intelligent model) would be to predict a linear relationship, since this is a very simple prediction that perfectly fits the training data. This is related to the No Free Lunch Theorem, which roughly states that universal optimizers are impossible [14].

2.4 OOD Calibrated Uncertainty

The final approach we consider is an uncertainty-based approach. In this approach, we relax the goal from trying to correctly classify OOD images to simply having an accurate uncertainty measure. This allows for the model to have a decrease in accuracy so long as the model also has a corresponding decrease in confidence.

In a 2019 paper, Ovadia et al. evaluated models expected calibration and accuracy after performing synthetic shifts [10]. We show their results in Figure 3. They created a library of synthetic shifts with intensity from 1 to 5, and tested several methods to see how accurate and how calibrated they were as distribution shifts became more intense. They show a box and whisker plot showing the quartiles of the performance over each shift that that they tested. Across all methods, we see the trend that as intensity increases, accuracy decreases and ECE gets bigger, both of which correspond to worse performance.

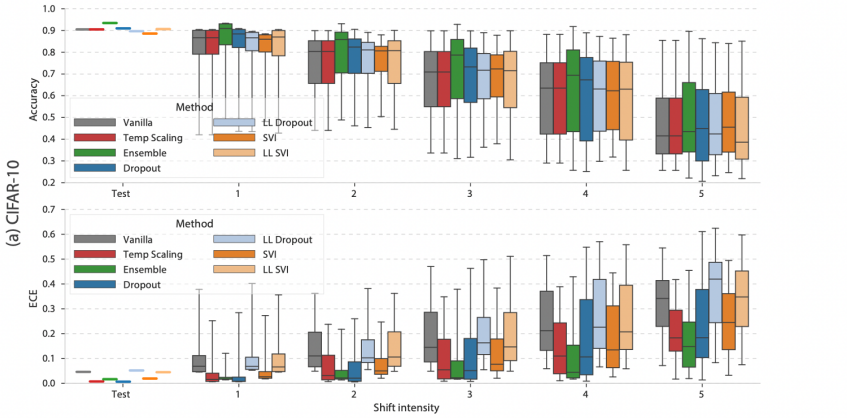

Figure 3: Model performance against magnitude of shift from Ovadia et al. [10]

	Classification (2.2)	Robustness (2.3)	Uncertainty (2.4)
Small shift	Classifies as in-distribution	Performs well	Performs well
Moderate shift	Classification depends on decision of what counts as OOD	Performs less well, but as well as possible	Performs less well, but with decreasing confidence
Heavy shift	Classifies as out-of-distribution	Poor accuracy and often overconfident	Poor accuracy, but calibrated confidence

Table 1: Characterization of the performance of different methods on different intensity shifts

2.5 Overview

When the shift is small and characteristics of the training set are still present, it is possible to predict on the OOD test samples (e.g., a slight blurring). When the test data has no common characteristics with the training data (e.g. pure Gaussian noise), the model cannot give any meaningful prediction because the learned characteristics are not relevant. There is a continuous space of possible shifts representing distribution shifts of varying magnitudes. Focusing on model uncertainty in order to have a confidence measure attached to the point prediction allows for calibrated uncertainty across all shift intensities. We present a summary of when these different methods are effective in Table 1.

We extend this work by finding methods that improve upon the methods attempted in [10]. We also extend the synthetic shifts used by [10], increasing the scale of intensity from 1-5, up to 1-10 (Between 1-5 our shifts overlap with this paper). This extension creates some shifts that are so extreme that DNNs are unable to do better than random guessing, giving us access to points on the full spectrum of possible intensities. We use a similar loss function as [7] and [12], except instead of using this in the OOD detection domain, we instead use it to improve the robustness of calibration of neural networks on OOD data.

3 Background

3.1 Notation

Suppose that we are given a training set $X = [x^{(1)} \dots x^{(n)}]$, where $x^{(i)} \in \mathbb{R}^d$ with labels $y^{(i)} \in \{1 \dots N\}$ sampled from a training probability called $P_{in}(x, y)$. Let $P_{out}(x, y)$ be the distribution that test data is sampled from. If $P_{in} \neq P_{out}$, we say that there is **distribution shift**. Our goal is to learn a predictor $f_\theta : \mathbb{R}^d \to \Delta^N$ (where Δ^N is the probability simplex over the N output classes) that performs as well as possible over samples from both P_{in} and P_{out}. Our model is paramaterized by θ. Let $\hat{p}^{(i)}$ be the output of

i	$\hat{p}_{cat}^{(i)}$	$y^{(i)}$	i	$\hat{p}_{cat}^{(i)}$	$y^{(i)}$
1	.9	cat	8	.9	dog
2	.9	cat	9	.6	dog
3	.9	cat	10	.4	dog
4	.9	cat	11	.1	dog
5	.6	cat	12	.1	dog
6	.4	cat	13	.1	dog
7	.1	cat	14	.1	dog

Table 2: An example of a models predictions for the cat/dog binary classification problem

f_θ on $x^{(i)}$.
When we say perform well, we are referring to two separate, but related attributes:

1. The model performs as accurately as possible on both P_{in} and P_{out}.
2. The model is calibrated well.

3.2 Calibration

Informally, we say that a model is calibrated if the predicted probabilities match with the observed chance that the classification prediction is correct. Ideally, the model's prediction $\arg\max_j \hat{p}_j^{(i)}$ should have a $\hat{p}_j^{(i)}$ probability of being correct, in the sense defined below.

$$\mathbb{P}(\arg\max_j \hat{p}_j^{(i)} = y^{(i)}) = \max_j \hat{p}_j^{(i)} \tag{1}$$

The left hand side of Equation 1 is the probability that the classifier is correct on the i-th classification example. This probability is over all images predicted with the same confidence level, and can be estimated via the fraction of correct predictions over each image with the specific confidence. The right side is the confidence itself, which is simply the largest of the class specific probabilities. Thus, this expression reads that our predicted probabilities correspond to real probabilities that the models prediction is correct. When this equation holds, this allows a subsequent decision maker to have a concrete interpretation of the output of a DNN which may play a key role in subsequent decision-making.

To illustrate this, suppose that we want to train a model to perform classification to separate cats and dogs. Thus, we have two classes. Suppose that we train a model and test it on some labeled data, and receive the results shown in Table 2. For each example, we show only the first component of $\hat{p}^{(i)} = [\hat{p}_{cat}, \hat{p}_{dog}]$, which is model's predicted probability that $x^{(i)}$ is a cat. The corresponding $\hat{p}_{dog}$ is then simply $1 - \hat{p}_{cat}$.

The confidence associated with the predicted class for each example is $\max_j \hat{p}_j^{(i)} = \max \hat{p}_{cat}^{(i)}, \hat{p}_{dog}^{(i)}$ (regardless of the predicted class, i.e. either when predicting cat or dog in this example). The predicted class is whichever class is associated with the larger probability. Let us see how similar the accuracy of this classifier is of the predicted probability. The points with confidence .9 are at $i = 1, 2, 3, 4, 7, 8, 11, 12, 13,$ and 14. Of these points, only 7 and 8 are predicted incorrectly, so the accuracy of these points is $\frac{8}{10} = .8$. This suggests this model is overconfident among these points by about $.9 - .8 = .1$, or 10%. Similarly, the accuracy among points with .6 confidence is $\frac{2}{4} = .5$, meaning that the model is again overconfident by $.6 - .5 = .1$, or 10%.

3.3 Expected Calibration Error

This example suggests a measure of the calibration of a model. We call this measure **Expected Calibration Error (ECE)**, which is defined as the average difference between the accuracy of the model and its confidence. To compute ECE, we need to split predictions with similar confidences $\max_j \hat{p}_j^{(i)}$ into K bins labeled as $B_1 \dots B_K$. We partition the unit interval into evenly sized bins.

$$B_k = \left\{ i : \max_j \hat{p}_j^{(i)} \in \left[\frac{k-1}{K}, \frac{k}{K} \right) \right\} \tag{2}$$

We define the confidence of a bin B_k to be the average prediction confidence of the items in the bin.

$$\text{conf}(B_k) = \frac{1}{|B_k|} \sum_{i \in B_k} \max_j \hat{p}_j^{(i)} \tag{3}$$

These individual confidences should be very close to each other, because they are by definition are in the same bin, and so are guaranteed to be within $\frac{1}{K}$ of each other. Similarly, the accuracy of a bin is the fraction of correct predictions in that bin:

$$\text{acc}(B_k) = \frac{1}{|B_k|} \sum_{i \in B_k} [\![\arg\max_j \hat{p}_j^{(i)} = y^{(i)}]\!] \tag{4}$$

The **Expected Calibration Error (ECE)** is thus:

$$\text{ECE}(X, Y) = \frac{1}{n} \sum_{k=1}^{K} |B_k| \cdot |\text{acc}(B_k) - \text{conf}(B_k)| \tag{5}$$

This is a discrete approximation of the average distance between the confidence of a model and its actual accuracy.

3.4 Differentiability

ECE is not a well-behaved function. It is differentiable almost everywhere, but due to the absolute values and the binning procedure, there are points at which ECE is non-differentiable and even discontinuous. The points at which this occurs are:

- At the edges of bins, when $\max_j \hat{p}_j = \frac{k-1}{K}$. At this point, changing this maximum element to be lower causes the point's bin to change, which results in a different calculation. At this point, ECE is not continuous.
- When the accuracy is equal to the confidence (i.e. $\text{acc}(B_k) = \text{conf}(B_k)$), the absolute value function is non-differentiable.
- When the maximum predicted probability is tied among two classes, one of which is the true class. Specifically, if $\hat{p}_y = \hat{p}_k = \max_j \hat{p}_j$, for $k \neq y$. At this point, the ECE is again not continuous.

Fortunately, these points make up a measure 0 subset of Δ^N, because each of these conditions defines a set of dimension less than N. At all other points, $\hat{p}_\ell^{(i)}$ is contained in a neighborhood whose points are all in the same bin, and is either overconfident or underconfident, and all have the same class prediction. In this neighborhood, the bins and predicted class are constant, and so the partial derivative of ECE with respect to $p_\ell^{(i)}$ is:

$$\frac{\partial}{\partial p_\ell^{(i)}} \text{ECE} = \begin{cases} 0 & \ell \neq \arg\max_j \hat{p}_j^{(i)} \\ -\frac{1}{n} & \text{if } \text{acc}(B_k) > \text{conf}(B_k) \text{ and } \ell = \arg\max_j \hat{p}_j^{(i)} \\ \frac{1}{n} & \text{if } \text{acc}(B_k) < \text{conf}(B_k) \text{ and } \ell = \arg\max_j \hat{p}_j^{(i)} \end{cases} \tag{6}$$

The cases in Equation 6 correspond to:

1. The chosen index is not maximal, and so it does not change the confidence, and hence does not change the ECE.
2. The accuracy of the corresponding bin is greater than the confidence. Hence, the model is underconfident, suggesting that increasing $p_\ell^{(i)}$ (the confidence) would decrease the ECE.
3. The accuracy of the corresponding bin is less than the confidence. Hence, the model is overconfident, suggesting that increasing $p_\ell^{(i)}$ (the confidence) would increase the ECE.

3.5 ECE is not a Proper Scoring Rule

Although low ECE is very desirable, it is insufficient to simply use ECE as the only term in a loss function because ECE is not a strictly proper scoring rule. There is a trivially optimal solution – assigning an equal probability to each class – that minimizes the ECE to zero. A model that does this will be correct with probability $\frac{1}{N}$, which is equal to the confidence that it predicts with Equation [15] (assuming a balanced dataset). For example, suppose that we find a new model that predicts $\hat{p}^{(i)} = [.5, .5]$ for our previous dataset of cats and dogs, for each $i = 1 \dots 14$. Suppose that the ties for arg max are broken in favor of the earlier index – so this model predicts cat for each point in the dataset. Since this is a balanced dataset (meaning that each label contains the same number of examples – here there are 7 cats and 7 dogs), the accuracy of this model is $\frac{7}{14} = .5$. This perfectly matches the confidence of .5 associated with each prediction, so ECE = 0. However, this classifier is completely uninformative – it does not distinguish between cats and dogs whatsoever. Thus, it is not useful to minimize ECE unless one can additionally maintain good performance on some other metric such as accuracy or cross entropy.

On severely shifted data, however, the model with low confidence is in fact desirable. If the signal learned during training is no longer present, we in fact want to encourage this trivial behavior, because the model has no way of knowing what the label could be and so should therefore assign probabilities according to its prior of prevalence in the population. In a balanced dataset, this corresponds to assigning equal probabilities to each possible class. A model that behaves in this way signals to the user that it was unable to infer anything from the input by predicting with such low confidence. Unfortunately, there is not a known way to ensure that the ECE term dominates as the shift increases.

3.6 Problem Setup

Ideally, DNNs should be able to perform with high accuracy under a mild distribution shift. On the other hand, given extreme distribution shift, DNNs should be able to express their uncertainty on the OOD input.

We propose an algorithm that allows models to express their uncertainty in a meaningful way. Under extreme distribution shifts, these models will inevitably perform poorly, but they will also express that they are uncertain by assigning a low confidence to their predicted label. This confidence will be meaningful, and it will correspond to the actual chance that the model is correct.

Thus, our goal is to minimize the calibration error of our data of our model over shifted data without sacrificing accuracy. If a model has low ECE on the OOD data, the decrease in accuracy will necessarily be associated with a proportional decrease in confidence. That low confidence reflects to the user that this is a difficult task, and allow for the model to either suspend itself or query human feedback.

4 Method

4.1 Expected Calibration Error Loss

We trained a model using a custom loss function in a manner similar to [16]. However, instead of penalizing through KL-Divergence with the uniform distribution on OOD data, we instead penalize miscalibration using ECE. Denote the prediction for each x by $f(x) = \hat{p} \in \Delta^N$, and y be the true label for x. Our loss function is a linear combination of cross entropy and an ECE term.

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{P_{in}(x,y)} [-\log f(x)_y + \beta \cdot \text{ECE}(x, y)] \tag{7}$$

The ECE term acts as a regularization that punishes miscalibration in the training set. We refer to this method as **ECE Loss**. We see the pseudocode for this method in Algorithm 1 – this is the standard batch gradient descent applied to this custom loss function. The hyperparameter β controls the strength of the ECE term in the loss. β must be chosen with care because if β is too high, the desire for perfect calibration will override the ability of the network to learn trends in the data. On the other hand, if β is too low, the regularization will be negligible causing this loss to simplify to standard cross entropy loss. We recommend using cross-validation to choose the highest value of β that does not result in decreased accuracy or cross-entropy performance.

Unlike standard loss functions, this loss function requires a large batch size to be effective because ECE requires enough samples to estimate the calibration of a model. Given a single training example, one can

compute the cross entropy of that sample, but this does not provide any information about the calibration of a model, which is a more global property that necessarily must refer to batches of training examples.

Algorithm 1 ECE Loss Training Loop

for epoch $= 1, 2 \ldots$
  for batchX, batchY in Training Set
    Loss $\leftarrow$ CE(batchX, batchY) $+\beta \cdot$ ECE(batchX, batchY)
    $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta$Loss

CE represents standard cross entropy loss defined in Equation 8 and γ is the learning rate. CE is also the first term in the ECE Loss objective defined in Equation 7.

$$\text{CE(X, Y)} = -\frac{1}{n}\sum_{i=1}^{n} \log \hat{p}^{(i)}_{y^{(i)}} \tag{8}$$

4.2 Outlier Exposed Expected Calibration Error Loss

If we have access to OOD data we can expose DNNs to this data during training. One way of doing this is to use the ID data to optimize for performance, but use the OOD data to calibrate the model.

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{P_{in}(x,y)} [-\log f(x)_y] + \beta \cdot \mathop{\mathbb{E}}_{P_{out}(x,y)} [\text{ECE}(x, y)] \tag{9}$$

We refer to this method as **Outlier Exposed Expected Calibration Error Loss (OE ECE Loss)**.

Algorithm 2 OE ECE Loss Training Loop

for epoch $= 1, 2 \ldots$
  for batchX, batchY in Training Set
    Loss $\leftarrow$ CE(batchX, batchY)
    $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta$Loss
  for batchX, batchY in Shifted Training Set
    Loss $\leftarrow \beta \cdot$ ECE(batchX, batchY)
    $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta$Loss

The pseudocode for this approach is shown in Algorithm 2. We iteratively minimize each term in the loss. In each epoch, we loop over each sample from P_{in} and each sample from P_{out}.

The key difference between ECE Loss (Equation 7) and ECE OE Loss (Equation 9) is that ECE OE Loss requires access to some labeled set of OOD data. We use the 'brightness' shift for training using the OE ECE loss, and we test on all other shifts. This ensures a fair evaluation because we cannot predict which distribution shifts might occur, so it is important for models to perform well under novel, unseen distribution shifts.

ECE OE Loss helps the model learn uncertainty in a process grounded by outlier data. By seeing shifted data, this loss function trains the model to expect that it might see data that it hasn't been trained to perform well on. This loss updates the model to improve its calibration on shifted data, but not its performance.

4.3 Training Details

Our MNIST models used a simple four-layer convolutional architecture. All CIFAR-10 models used the EfficientNetB0 Architecture [17]. We used transfer learning to train the CIFAR-10 models – weights were initialized from a model trained on ImageNet. We used a batch size of 128, and we used $K = 10$ bins. The hyperparamaters γ (learning rate) and β (ECE term weight) were selected by grid search using 5-fold cross validation. We chose the largest possible β without sacrificing significant ($\tilde{1}\%$) validation accuracy on the training distribution. We did not test the performance of any models on OOD data during hyperparameter selection. We used an NVIDIA GeForce RTX 2080 Ti GPU with CUDA 11.4 and Tensorflow 2.6.2 for training and evaluation.

Figure 4: A pair of twos, left is from MNIST and right is from SVHN.

5 Datasets

We classify distribution shifts into two categories.

- **Natural Shifts** involve unmodified images from two different datasets. For an example of a natural shift, see Figure 4.
- **Synthetic shifts** involve well-defined image modifications at the pixel level (e.g. image corruption [18], style transfer [19], or adversarial modifications [20]). For an example of a synthetic shift, see Figure 5.

Models in the real world tend to face natural shifts. However, synthetic shifts are much easier to use: instead of needing a whole new dataset, one can algorithmically generate shifted data from existing datasets. Although recent work has shown that synthetic shifts do not always generalize to natural shifts [4], we will focus primarily on synthetic shifts because we can monitor the intensity of these shifts, and we are interested in understanding the trend in quality of uncertainty shift over time. We include one natural shift in order to demonstrate the efficacy of the method on non-synthetic examples.

5.1 Library of Synthetic shifts

We based our library of synthetic shifts on a selection of the shifts used in the MNIST-C dataset [21], but we increased the intensity of some of these shifts. MNIST-C contains 15 shifts, but we select 10 of these shifts: Gaussian Noise, Shot Noise, Impulse Noise, Motion Blur, Brightness, Contrast, Pixelate, JPEG Compression, Speckle Noise, and Gaussian Blur. We chose these because they were easier to extend in intensity. These are a wide range of shifts, spanning shifts which mimic natural distribution shifts (such as Motion Blur) from shifts that happen during synthetic digital image processing (such as JPEG compression). For each shift, we scale their intensities from 1 to 10. We implemented these using OpenCV, and increased the paramaters linearly. Note that the intensity of one shift does not correspond in a big sense to the intensity of another shift. For some shifts, maximum intensity completely overwrites the image, whereas for others it is still possible to tell what the original image was. For examples of each of these shifts, refer to the Appendix.

5.2 MNIST

MNIST is a database of grayscale handwritten digits. Each image has shape $(28, 28)$. There are 60,000 train images and 10,000 test images. An example is shown on the left in Figure 4. We perform both natural and synthetic shifts on MNIST.

5.2.1 MNIST to SVHN

SVHN is a dataset of $(32, 32, 3)$ images of street signs labeled $0 - 9$. An example of an SVHN image is shown on the right in Figure 4. There are several factors that make SVHN significantly more difficult than MNIST. First, simply due to the shift, there will be a decrease in performance. Second, SVHN includes colors, which often means that there is significantly less contrast than MNIST. Additionally, the background of SVHN is not constant; there are often portions of other numbers sticking into the frame, as well as background edges and shapes. This is very different than MNIST data, where the background of each image is a solid black. This is a natural distribution shift because these are all unmodified images – the SVHN is a real world separate dataset from MNIST.

We added basic pre-processing to allow both datasets to be able to be input to the same neural network. This is necessary because we wish to train on MNIST and then test the model on SVHN. We made each data set have the same dimensionality by zero padding MNIST and then copying the data into each color channel to increase the size from $(28, 28)$ to $(32, 32, 3)$. This does not add any information to the training set.

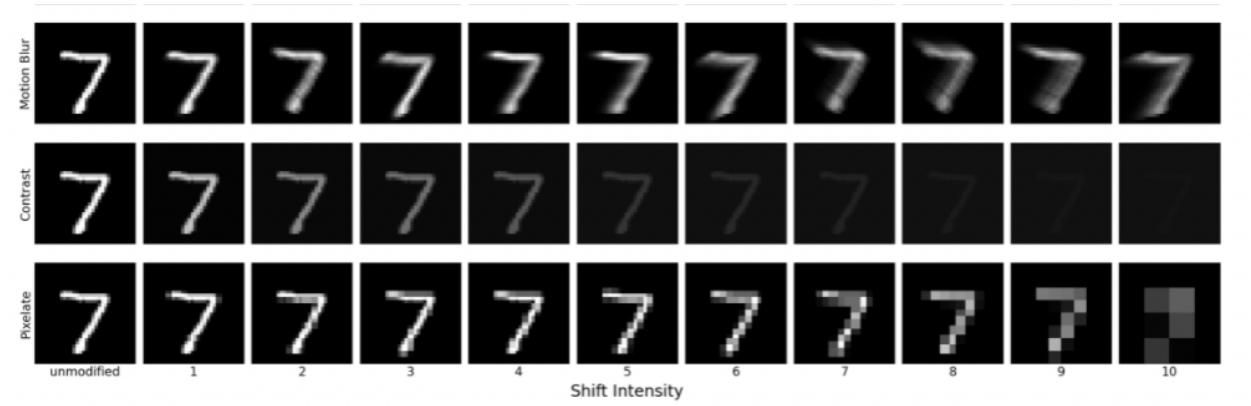

Figure 5: A selection of the synthetic shifts that we perform on MNIST Data.

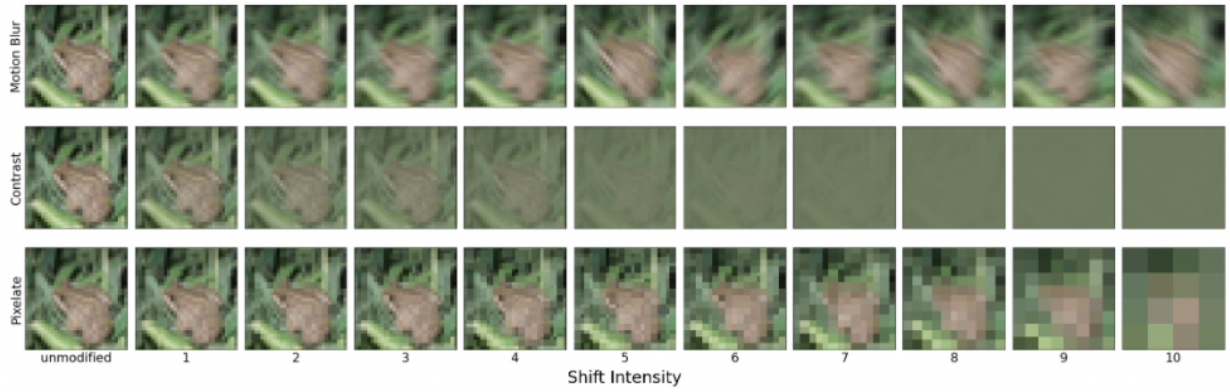

Figure 6: A selection of the synthetic shifts that we perform on CIFAR.

5.2.2 MNIST Synthetic shifts

We performed our library of synthetic shifts on MNIST. Three of these shifts are shown in Figure 5. We can see that the image starts out very legible, and the quality dramatically degrades as the shift intensity increases. Eventually, even a human is rendered unable to identify the digit, especially in shifts such as Contrast shift, where the image eventually becomes a constant black. For a complete example of each shift we perform on MNIST, refer to the Appendix.

5.3 CIFAR-10

CIFAR-10 is a dataset that contains 60,000 images with shape $(32, 32, 3)$. It contains 10 labels: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. This is a more difficult dataset for DNNs to classify than MNIST, because the concepts are more complicated than handwritten digits.

5.3.1 CIFAR-10 Synthetic shifts

We perform the same synthetic shifts on CIFAR-10 as MNIST, and again show three of these shifts in Figure 6. Similarly to MNIST, the image clarity degrades rapidly. For a complete example of each shift we perform on CIFAR, refer to the Appendix. We can see that these shifts, starting with the same image, strongly blur the initial image. The most difficult shift is contrast, which removes nearly all detail, making it impossible for even a human to classify the image at high severity.

6 Results

We analyze both the performance of each method using both accuracy and ECE, and we analyze this on both OOD and ID data. Over both datasets and most shifts, as measured by ECE, we find that OE ECE

	Naive Training	ECE Loss	ECE OE Loss
MNIST Accuracy	.992	.984	**.996**
MNIST ECE	.041	**.001**	.012
SVHN Accuracy	**.160**	.151	.137
SVHN ECE	.180	.051	**.038**

Table 3: Test results using MNIST as ID and SVHN as OOD. Best row performance is bold.

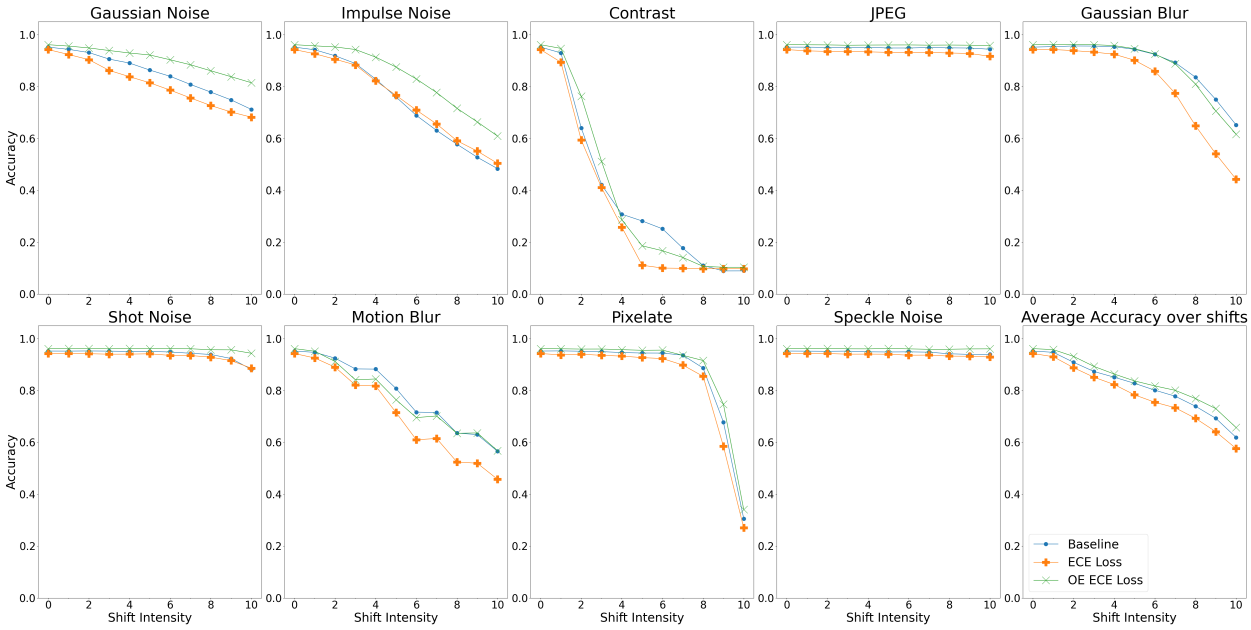

Figure 7: The accuracy of each classifier on each MNIST shift.

Loss performs the best, followed by ECE Loss, and the baseline model performs the worst. These methods perform similarly on MNIST, but ECE Loss and OE ECE Loss see a substantial improvement in performance on CIFAR. This is consistent with the results from Ovadia et. al. that robustness performance on MNIST data does not predict the performance on other datasets [10].

6.1 MNIST to SVHN Results

To evaluate our model on this shift, we evaluate performance metrics on the $10,000$ test images from the SVHN dataset [22] that each contain a single number.

All three methods achieve near-perfect test accuracy on MNIST and achieve quite good test ECE, although ECE Loss performs the best. This was expected because this method has an explicit minimization for low ECE on the train distribution. However, on the OOD Data (SVHN), both ECE and OE ECE Loss outperform the baseline model with respect to calibration, with OE ECE OE performing best. This is suggests some generalization of calibration from synthetically shifted data to naturally shifted data. The exposure to OOD data that is hard for a model to classify accurately helps to maintain its calibration. All models have extremely low accuracy on SVHN, but our loss functions involving ECE allow for superior calibration.

6.2 MNIST Synthetic Shifts Results

Figure 7 shows the accuracy plotted against the shift intensity of each classifier. We held out the brightness shift because used this shifted data to train the model which used OE ECE Loss. The accuracy follows a similar trend on each model and shift – models start with near perfect accuracy, and then gradually degrade as shift intensity increases. However, even given large amounts of noise, MNIST is a sufficiently easy dataset

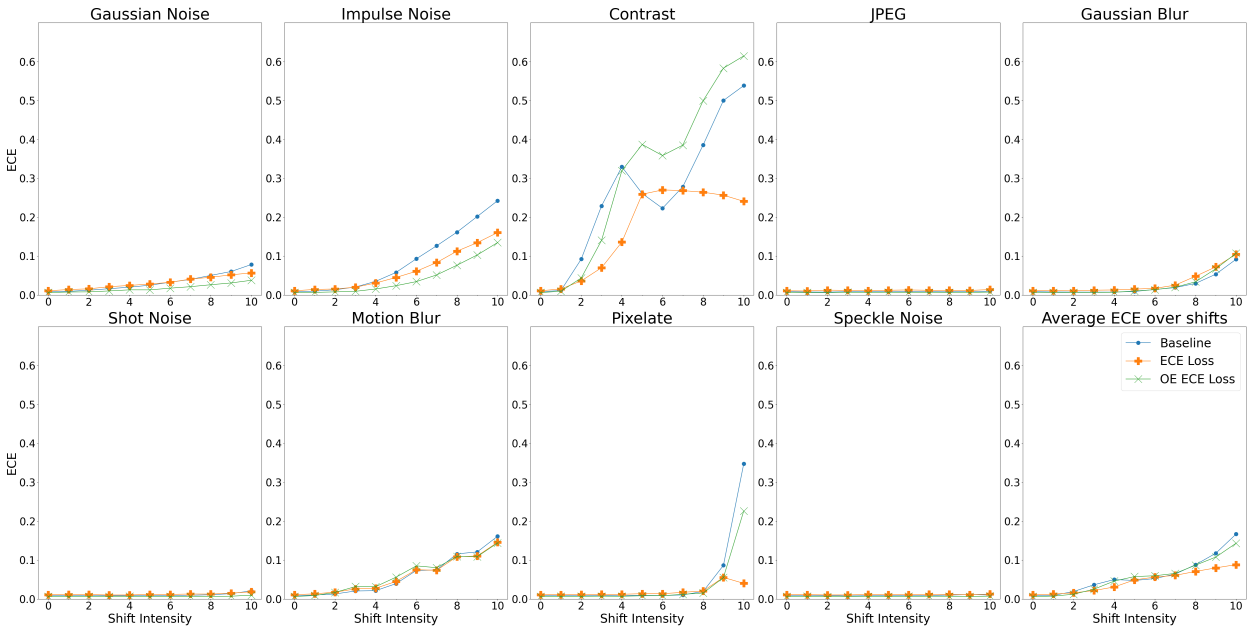

Figure 8: The ECE of each classifier on each MNIST shift.

that classifiers are naturally quite robust to these shifts. In particular, JPEG compression, Shot Noise, and Speckle noise all do very little to harm either the accuracy or the ECE. However, with other shifts we observe the predicted trend of decreasing accuracy and increasing ECE as shift intensity gets larger – everything seems to get worse. On net, however, ECE Loss and OE ECE Loss do perform very similarly on most shifts and deviations on accuracy remain small. ECE Loss performs the worst, and this is likely because it optimizes for ECE directly on the training distribution. The ECE term is an independent optimization to accuracy, so when accuracy is at an optimum, this term acts to slightly reduce accuracy. Overall, due to the simplicity of MNIST, these models are naturally robust to many of these shifts, as evidenced by the minor reduction in accuracy.

In Figure 8, we show the ECE performance of each classifier plotted against shift intensity. The bottom right figure shows the (macro) average trend of models. Note that there is a lot of variance in these trends, but this covaries with the accuracy graph. We are primarily worried about high ECE paired with high accuracy, because this corresponds to a highly confident and highly wrong classifier, which is the kind of model that can cause lots of harm in the real world. In particular, these are shifts like high severity Contrast and Pixelate. In both of these shifts, the baseline model is the most miscalibrated. Overall, over most shifts, the ECE is quite low, which means that the models are performing well. In particular, the only shifts that induce poor ECE performance are contrast, impulse noise, and motion blur. In the average plot, we see that each method performs very similarly over each shift on MNIST data. This is largely because MNIST classifiers, due to the simplicity of the dataset, are naturally quite robust. We must look at more complicated datasets to get a more detailed understanding of the performance of these methods.

6.3 CIFAR-10 Results

We evaluate the performance of these methods on the CIFAR-10 dataset. We perform all of our shifts, but we again hold out the brightness shift from the evaluation because we used this shift to calibrate the OE ECE Loss model. Table 4 summarizes the performance of the models averaged over all shift magnitudes and all shifts. All models have similar accuracy, but the calibration of models trained using the methods described have superior ECE. The accuracy decreased by 1.5%, but ECE Loss improves ECE by 13% and OE ECE Loss improves ECE by 16%, suggesting significantly more robust uncertainty than the baseline.

More detailed graphs of these results are shown in Figures 9 and 10. In these figures, we averaged the performance over each shift intensity for each model and synthetic distribution shift. Each model's performance on both accuracy and ECE degraded with a shift. However our methods significantly improved

	Baseline	**ECE Loss**	**OE ECE Loss**
Accuracy	**.577**	.562	.566
ECE	.234	.106	**.065**

Table 4: Summary of the CIFAR performance of methods averaged over synthetic shifts and intensity. Best row performance is bold.

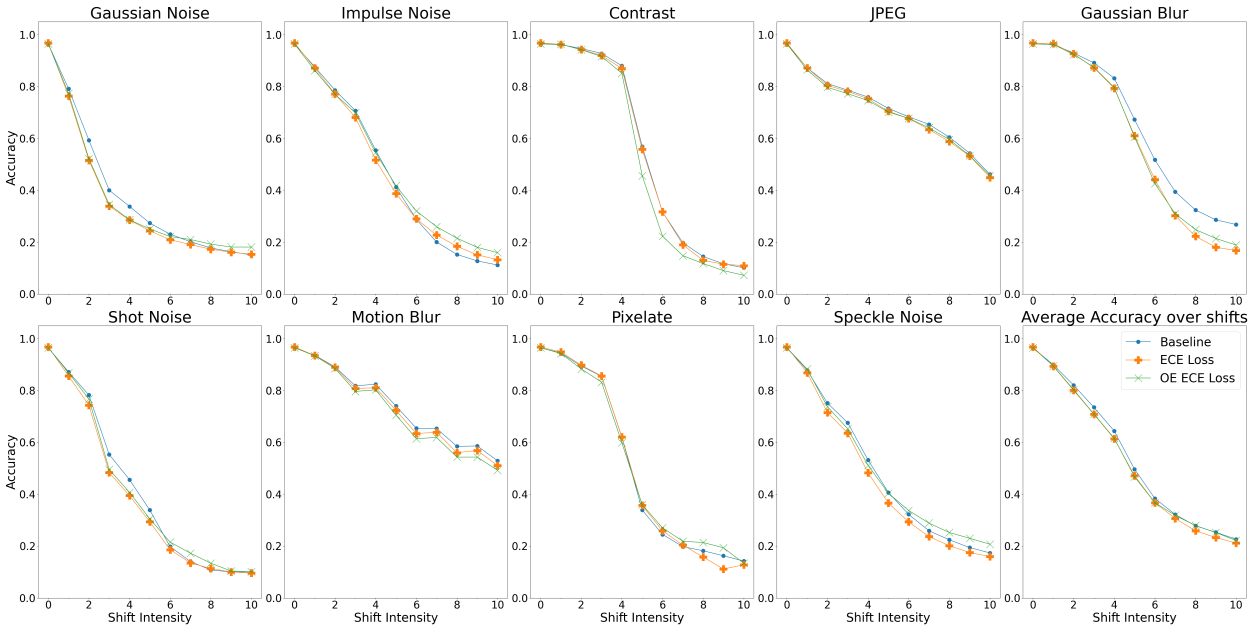

Figure 9: The accuracy of each classifier on each CIFAR shift.

performance. In particular, the ECE of both novel training methods was significantly lower than the baseline performance on both shifted and raw test data. Moreover, the average ECE of the baseline model increases monotonically, whereas for both ECE Loss and OE ECE Loss, the average ECE flattens out and even slightly decreases, suggesting that with enough shift intensity, the model is learning to be almost completely uncertain.

The same distribution shifts adversely affect the accuracy of CIFAR models much more strongly than MNIST models. Despite this, not all shifts are severe enough to reduce the accuracy to random guessing, which is a .1 accuracy, since there are 10 classes. Shot noise and Contrast seem to be the most severe, whereas Brightness, Motion Blur, and JPEG maintain an accuracy substantially larger than guessing. This agrees with human intuition; at maximum shift, a human predictor could certainly label the example shown in the appendix for the Brightness, Motion Blur, and JPEG, whereas the Contrast shift becomes a solid background, making it impossible for even a human to classify. Similarly, shot noise is also an intense shift.

The accuracy of ECE and OE ECE Loss is quite correlated between models. This suggests that there is a similar overall difficulty to each shift at each magnitude. Due to this similar accuracy being different between shifts at different intensities, the shift intensities are not equivalent between shifts. For example, the intensity 4 Contrast shift has an accuracy of about .88 over each model. On the other hand, the intensity 4 Shot Noise shift has an accuracy between .4 and .5.

For the majority of shifts, we see a monotonically increasing trend of the ECE for all classifiers. This is expected behavior because performance on any metric tends to decrease with distribution shift. For the shifts Speckle Noise, Gaussian Noise, Motion Blur, and JPEG, this is the observed trend. Furthermore, in each of these shifts, the baseline model performs worst, ECE Loss performs better, and OE ECE Loss performs best. The success of OE ECE Loss suggests that calibration on known shifted data can help the model stay

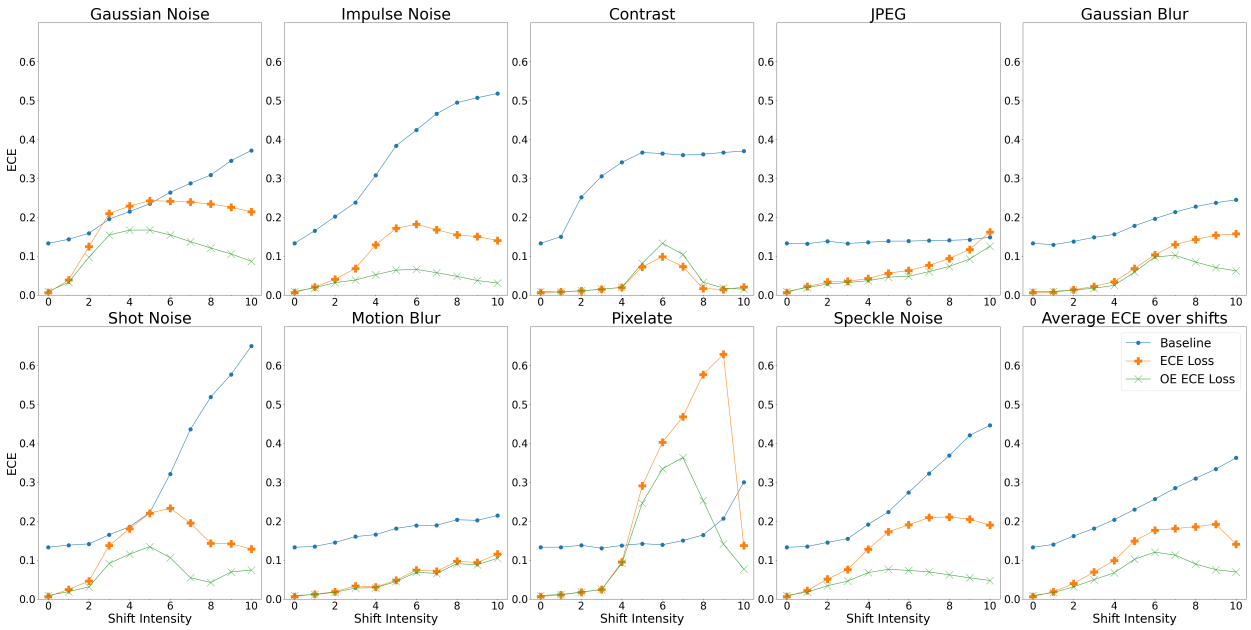

Figure 10: The ECE of each classifier on each CIFAR shift.

calibrated even on new, unseen, test distribution shifts. However, on some shifts, we see deviations from the norm. Both Shot Noise and Contrast exhibit a similar and interesting trend: the two ECE methods had an increase in ECE for the middling intensity shifts, but then decreased after that regime. An explanation for this is that with a mild distribution shift, the model behaves with the confidence it learned on the training set, resulting in overconfident predictions. Once the shift became extreme enough, the calibration training is more apparent, and the model recognizes that the data is too shifted for it to classify accurately, and adjusts its confidence downward accordingly.

7 Conclusion

We propose a novel method for improving the calibration of neural networks under distribution shift. This method involves including Expected Calibration Error (ECE) in the loss function. The first method (ECE Loss) minimizes the ECE over the training set, while the second (ECE OE Loss) additionally uses a selection of OOD data as outliers to calibrate on shifted data. We implement and evaluate this method on shifted versions of MNIST and CIFAR-10. We find that ECE Loss and ECE OE Loss both outperform a baseline network significantly: on CIFAR-10, ECE Loss gives .106 test ECE and ECE OE loss .065 test ECE, both of which are significantly lower than the .234 test ECE from a baseline model. It achieves this while remaining at similar test accuracy in each bin of distribution shift.

We advocate development and usage of ML models that consider not only model performance, but also model calibration, especially on OOD input. This would help avoid the undesirable scenarios in which a model performs with poor accuracy and high confidence. Work to avoid this scenario helps to improve the trustworthiness of future ML systems.

In addition, there are several specific research directions that directly build on the work done here. These include:

- Evaluating the Bayesian Neural Network approach in comparison with these approaches [23].
- Training models with ECE and ECE OE Loss using more sophisticated datasets (e.g. including images with no label). See if this method can be used for OOD detection as well.
- Incorporating methods that improve the robustness of models such as Noisy Student [13] to improve accuracy, to see if we can improve robustness while also achieving lower ECE.

- Comparing calibration results to postprocessing methods for DNNs such as Dirchlet Calibration [24].
- Scaling up this method to evaluate it on ImageNet and the distribution shifts in ImageNetV2 [10].

8 Acknowledgements

I would like to thank Dr. Xun Huan for useful advice on research direction. Your advice greatly improved both the technical content of the article and the clarity of my thoughts and writing. In addition, you have inspired new research directions that I am excited to pursue. I would also like to thank Stephen Zekany for discussions of these ideas, as well as for your help in discussing and clarifying my communication. Your thoughtful feedback helped me sharpen my thinking and brought my work to a higher level. I also thank Dr. Thomas Wenisch, Dr. Sinhdu Kutty and Dr. Ronald Dreslinski for their guidance and support, without you this thesis would not have been possible. Finally, I thank Daniel Mishins for his useful feedback on early drafts of this thesis, Frank Seidl for detailed comments on later drafts of the thesis, and Katie Joy Knister for her enthusiasm and support throughout this process.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115–118, Jan. 2017.

[3] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[4] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring robustness to natural distribution shifts in image classification," 2020.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[6] G. Franchi, X. Yu, A. Bursuc, R. Kazmierczak, S. Dubuisson, E. Aldea, and D. Filliat, "Muad: Multiple uncertainties for autonomous driving benchmark for multiple uncertainty types and tasks," *arXiv preprint arXiv:2203.01437*, 2022.

[7] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," 2018.

[8] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016.

[9] M. Valdenegro-Toro, "I find your lack of uncertainty in computer vision disturbing," *arXiv preprint arXiv:2104.08188*, 2021.

[10] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Advances in neural information processing systems*, vol. 32, 2019.

[11] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," 2018.

[12] A.-A. Papadopoulos, M. R. Rajati, N. Shaikh, and J. Wang, "Outlier exposure with confidence control for out-of-distribution detection," *Neurocomputing*, vol. 441, pp. 138–150, 2021.

[13] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

[14] Y. Ho and D. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of Optimization Theory and Applications*, vol. 115, pp. 549–570, Dec. 2002.

[15] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction," tech. rep., and estimation. Technical Report 463, Department of Statistics, University . . . , 2004.

[16] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," 2018.

[17] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[18] T. Saikia, C. Schmid, and T. Brox, "Improving robustness against common corruptions with frequency biased models," *arXiv preprint arXiv:2103.16241*, 2021.

[19] S. Gowal, C. Qin, P.-S. Huang, T. Cemgil, K. Dvijotham, T. Mann, and P. Kohli, "Achieving robustness in the wild via adversarial mixing with disentangled representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1211–1220, 2020.

[20] H. Singh, S. Joshi, F. Doshi-Velez, and H. Lakkaraju, "Learning under adversarial and interventional shifts," *arXiv preprint arXiv:2103.15933*, 2021.

[21] N. Mu and J. Gilmer, "MNIST-C: A robustness benchmark for computer vision," *CoRR*, vol. abs/1906.02337, 2019.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[23] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

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

9 Appendix

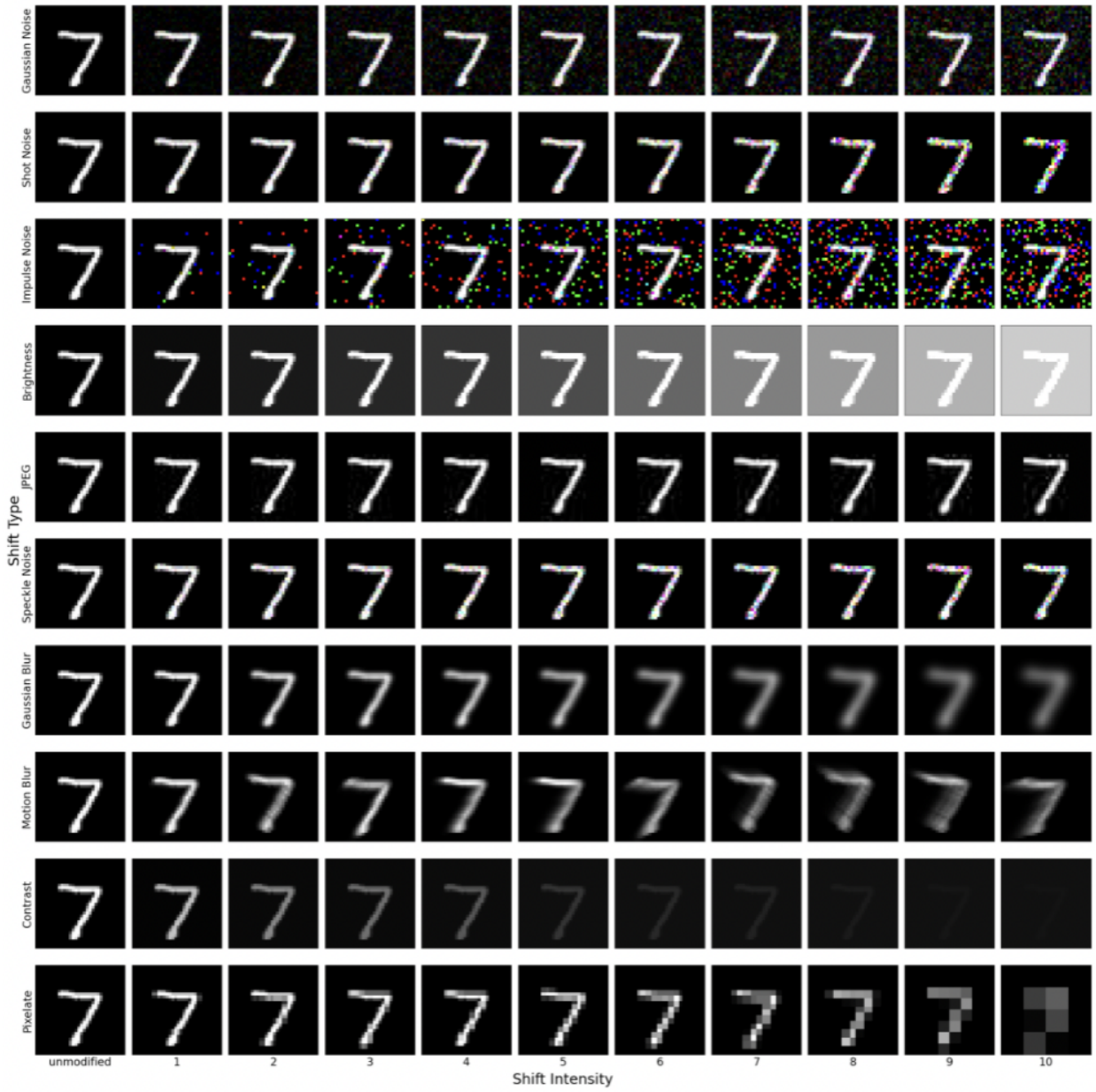

Figure 11: All of the synthetic shifts that we perform on MNIST Data.

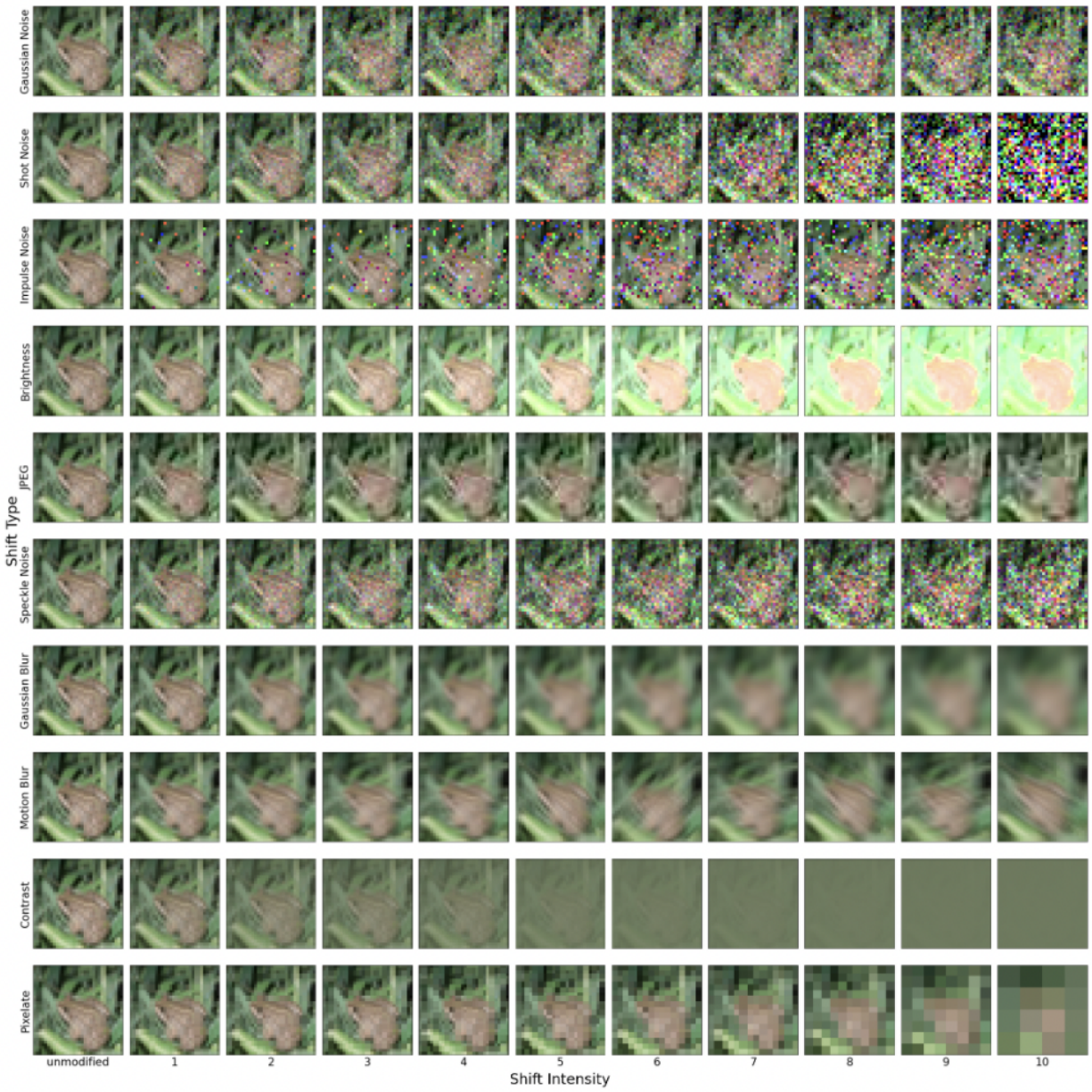

Figure 12: All of the synthetic shifts that we perform on CIFAR Data.