Neural networks - Project 2

Olga Bonachera del Pozo - Daniel Kwapien - Alejandro Sánchez

1. Initial Dataset

In this assignment we are asked to create a classifier trained on images of birds and cats from the CIFAR-10 dataset. So our first challenge will be to extract these images in a way that they can be used as an input for training our neural network. In order to do this, we create a new class which is extended from the Dataset class from torch utilities, we recommend checking out this class in the provided code.

As we import the dataset we do not apply the transformation directly in order to save computation time, first we extract the images we are interested in, which corresponds to the labels 2 and 3 in the CIFAR-10 dataset and then transform them into a numpy array. Then we apply the transformation.

2. Train a LeNet-5 and evaluate the confidence calibration

We are asked to create a LeNet-5 neural network from scratch and trained with our birds and cats dataset. In this creation of this dataset we directly implement dropout and batch normalization in order to increase the performance [1], so we can see more clearly the effects of calibration methods.

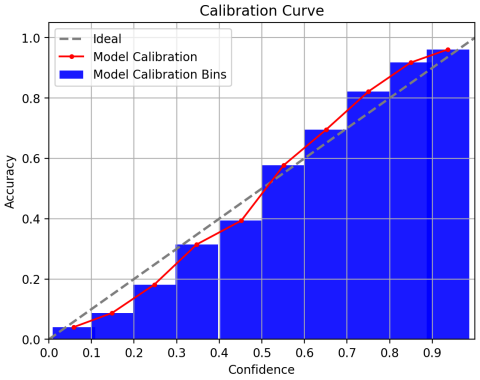
Additionally from our training and evaluation functions, we implement three additional functions. The first one called `compute_probs` is used to compute the probabilities from the outputs of the model, which are log probabilities. In it we return the true labels and the probabilities.

The second one is called `plot_diagram` and in it, once we have our true labels and probabilities, we use scikit's function `calibration_curve` to obtain two arrays, one corresponding to the true probabilities and to the predicted probabilities respectively. Finally we plot these probabilities against each other, which is the reliability diagram.

The third and last function `expected_calibration_error`, which computes the Expected Calibration Error (ECE) by creating a series of bins, then from the probabilities we compute the predictions and create an array from which we can compute the accuracy for each bin. Finally we enter a loop for each interval (each bin), compute the accuracy and average confidence in that bin and compute the ECE with the provided formula in the paper.

Now, given our neural network, we train it with our dataset and plot our reliability diagram (figure 1).

[1] Guo et al., "On Calibration of Modern Neural Networks."



We can see in figure 1 how there is a miscalibration between the average confidence and the accuracy. It shows how the model is being slightly under confident, i.e for a confidence between 0.7 and 0.8, we have an average accuracy of almost 0.9.

These are the results we expected from training a LeNet-5 with a Negative Log-Likelihood, since it is not a really deep model there is no room for the model to maximize the probabilities producing a situation where the model is being under confident.

3. Implement Temperature Scaling

In this section we will explain how we implemented Platt's scaling, specifically Temperature scaling, which consist in introducing a smoothing constant T that divides the logits, such as $p = \sigma(z/T)$ where z is the logit. This way, we do not affect the model accuracy and we are able to adjust the confidences of our model. We run an experiment to see the effects of different values of T on our model. Figure 2 represents the different temperatures tried from left to right and from top to bottom.

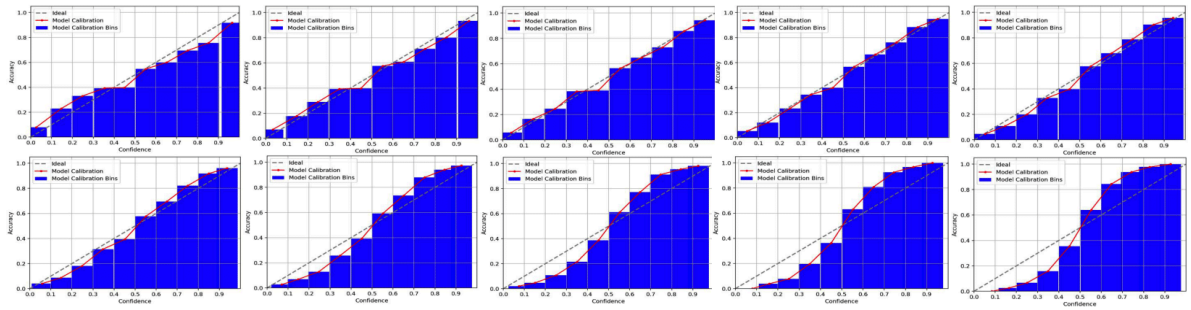


Figure 2: Trainings for temperatures [0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0]

Even though the confidence calibration of LeNet-5 is quite good, there is room for improvement. As we know, if we choose values such that $T < 1$ and $T \to 0$ the probabilities will tend to collapse to a point mass (i.e $p = 1$), on the other hand for values such that $T > 1$ as $T \to \infty$ the probability will tend to $p = 1/K$, in this case, $p = 0.5$. So depending on the situation we are in, we should optimize our confidence that T will be adjusted in a way that we get closer to the ideal confidence (i.e minimize ECE). In this case since we are in a situation of under confidence, **we are interested in** $T < 1$

Tº	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**	**1.25**	**1.50**	**1.75**	**2.0**
ECE	0.0564	0.0361	0.0163	**0.0150**	0.0311	0.0481	0.0842	0.1132	0.1364	0.1554

Table 1: ECE for different temperature

A **unique minimal ECE** value is found around 0.0150, obtained at the sweetspot around $T = 0.8$. The original ECE value was around 0.0481.

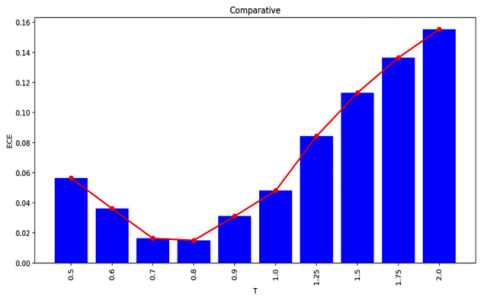


Figure 3: ECE among different values of T

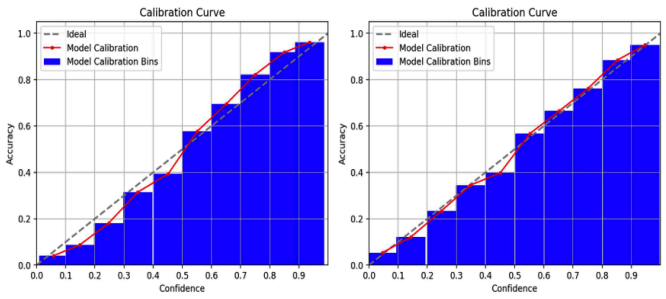


Figure 4: Non scaled ECE vs best scaled ECE

Overall, deviating from this T value in any direction leads to less calibrated results.

4. (Optional part) Repeat the experiment with a more complex model

We repeated the experiment with the DenseNet model. The features part works perfectly with our dataset but a replacement on the classifier needs to be made so it works for the CIFAR-10 when classifying two classes.

To replace the classifier we created a MLP binary classifier with three dense layers and hidden layers of size 120 and 84. We replace the classifier part with the MLP class, after creating it, with `model.classifier = MLP(1024.` Later, with the class `Tran_eval()`, we train the classifier with 100 batches.

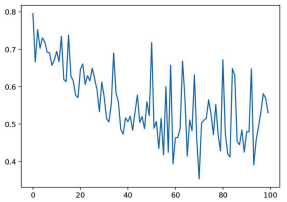


Figure 5: loss during training

The reliability diagram leads to think that the model is being under confident so we are going to apply temperature scaling to see if we can reduce the Expected Calibration Error which is around 0.0332.

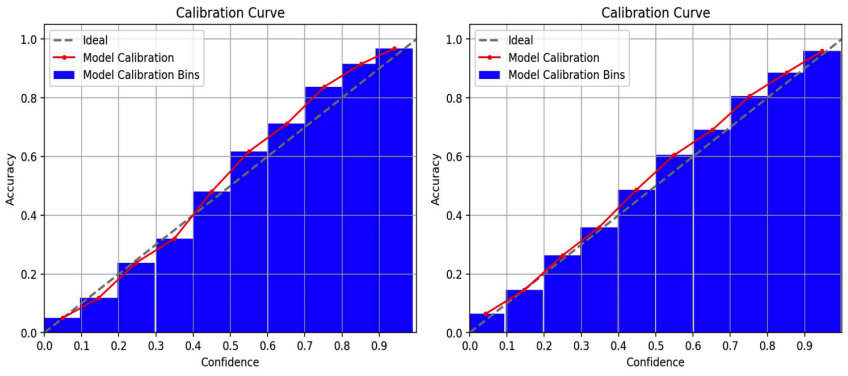


Figure 6: T = 1 scaled vs T = 0.8 scaled

With the temperature we used for our previous model, $T = 0.8$, the model looks much more calibrated and the Expected Calibration Error decreases significantly to 0.0134.