Universidad
Carlos III de Madrid

Project II: Understanding Calibtration in CNNs

Deep Learning

Authors

Juan Muñoz Villalón
100502334@alumnos.uc3m.es

Mario Golbano Corzo
100504162@alumnos.uc3m.es

Elena Almagro Azor
100504241@alumnos.uc3m.es

Universidad Carlos III de Madrid

November 11, 2024

Index

Página

This project explores CNN calibration in classification by training a LeNet5 model to classify cats and birds in CIFAR-10. Using reliability diagrams and ECE, we assessed calibration before and after applying temperature scaling to adjust output probabilities.

1 Calibration

1.1 Calibration Study

To study calibration, we developed functions to visualize the reliability diagram (or calibration curve) and estimate the Expected Calibration Error (ECE). These are *get_y_true_y_prob*, which returns the true labels and predicted probabilities from the model, *plot_reliability_diagram*, which plots the calibration curve based on these values, and *ece_cal*, which calculates the ECE.

1.2 Temperature Scaling

To improve calibration, we used *Temperature Scaling*, a simplified version of *Platt Scaling*. Here, given logits z_i, the new confidence prediction is given by the sigmoid (as we are implementing a binary output model) of the logits divided by a Temperature factor T:

$$\hat{q} = \sigma(z_i/T) \tag{1}$$

To implement this, we added the method *forward_logits* to the model, which returns the logits from the last layer before applying the activation function. These logits are then scaled in *apply_scaling*, a function similar to *get_y_true_y_prob* but with a Temperature parameter T. This function subsequently calls *temperature_scaling*, which actually performs the scaling.

2 Lenet5 CNN

For this project, we used a binary classification version of the LeNet5 Convolutional Neural Network, adapted from previous Labs. We modified the output layer to use a sigmoid function instead of the log-softmax to output a single probability for one class.

2.1 Initial Model

To intentionally train an uncalibrated model, we overfit it by training for 15 epochs, as shown in Figure 1, even though training would normally stop at 6 or 7 epochs.

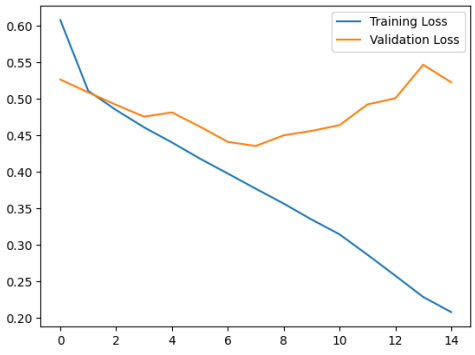

Figure 1: Training Loss vs Validation Loss

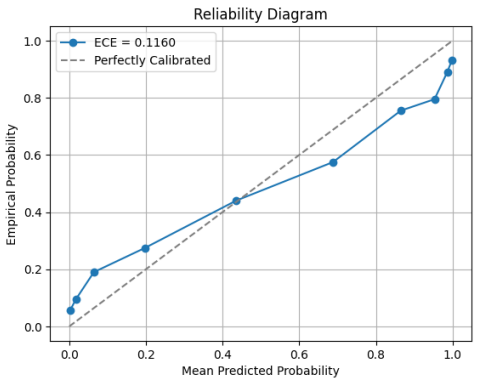

Figure 2: Uncalibrated model

For this uncalibrated model, the initial ECE was 0.116, which represents an uncalibrated model, as shown in Figure 2.

2.2 Model after Temperature Scaling

Following the outlined process, we tested various Temperature factors to minimize the ECE. In Figure 3, we show some of the studied parameters, ECEs obtained, and compare the best T factor with $T = 1$ (uncalibrated). The best ECE (lowest T factor) is represented by a thick red line, while the original model (uncalibrated) appears in blue.

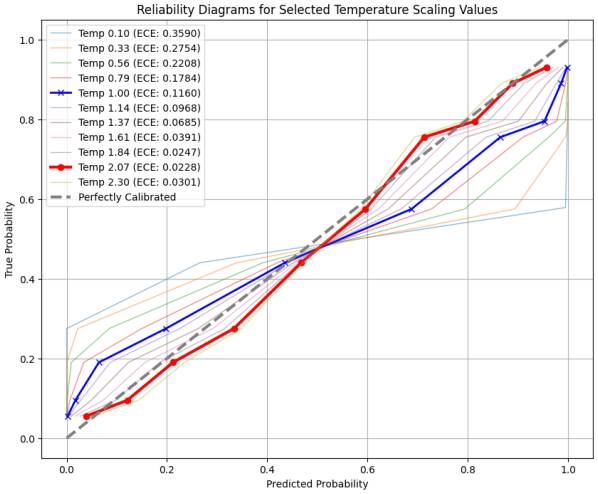

Figure 3: Reliability Diagrams for different T factor values for Temperature Scaling

As shown, a Temperature Factor of 2.07 reduced the ECE significantly, achieving a value of 0.0228, which greatly improved calibration. As seen in Figure 3, this calibrated model is almost completely fit to the perfectly calibrated model.

3 DenseNet 121

For this extra section we have used the pre-trained DenseNet121 model from torchvision.models.The original layers of DenseNet121 are "frozen" *(param.requires_grad = False)* to prevent retraining, preserving the learned features from its pre-training. Instead of retraining the entire network, the original classifier layer is replaced with a custom multi-layer perceptron (MLP) tailored to our classification task.

The custom MLP model architecture takes an input tensor of 1024 features, reduces it to a hidden layer of 500 units, and then maps it to a final output layer of 2 classes. A 'LogSoftmax' layer is applied this time. So now we have outputs that are are log-probabilities, which are suitable for binary classification tasks using negative log-likelihood loss ('NLLLoss'). Due to this modification, small adjustments were made in *get_y_true_y_prob* and *apply_scaling* to adapt them to this new case. With these modifications, and before applying temperature scaling for calibration, we observe the model's uncalibrated performance in figures **??** and **??**

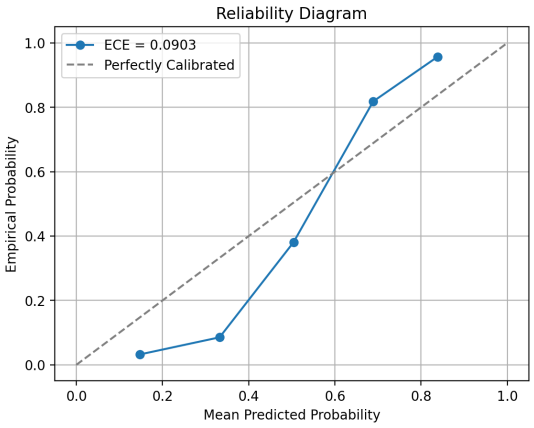

Figure 4: Mean Predicted Probability

For this uncalibrated model, the ECE obtained was 0.0542.

3.1 Model after Temperature Scaling

Following the same process as with Lenet5 model, we tested again for different values of Temperature factors, as it is observed in figure 5

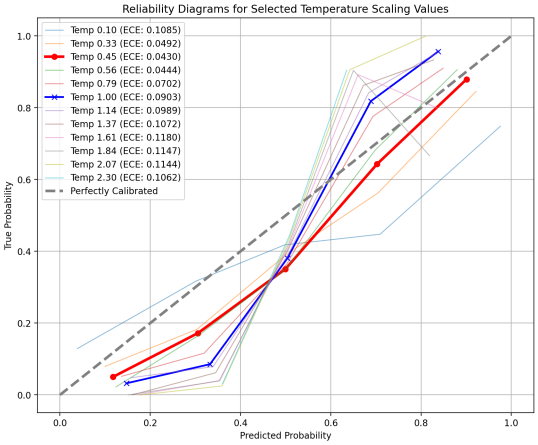

Figure 5: Reliability Diagrams for different T factor values for Temperature Scaling (DesNet121)

For this scenario we can observe that a temperature factor of 0.68 seems to be the one that reduces the ECE the most, obtaining a value of 0.0305.

When comparing both models we obtain a similar value of ECE for both uncalibrated models. For LeNet5 the ECE value was around 0.11, and the DesNet121 was about 0.09. When applying the temperature scaling we can see that the improvement is clearly more evident in the LeNet5 model, given its simplicity versus the DesNet121.

It is noteworthy that for the LeNet5 model we get a value for the best Temperature factor bigger than 1, meaning that it diminishes the confidence for higher probabilities. We can see this fact in figure 3. On the other hand, for the second model, we observe the opposite. We need to increase the confidence for higher probabilities, while decreasing it for lower probabilities, as we can see in figure 5.