Neural Networks

Project 2:

Understanding Calibration on CNNs

Mario Coronado Fernández (100496637)

Jorge Chamorro Pedrosa (100496527)

1. Dataset Preparation

In this assignment we are asked to create a classifier trained on images of birds and cats from the CIFAR-10 dataset. Our first task 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.

Instead of applying transformations during the dataset import, we first filter and extract the relevant images (label 2 for birds and 3 for cats in CIFAR-10). This approach saves computation time by avoiding unnecessary transformations on irrelevant data.

2. LeNet-5 Training

We implemented the architecture of LeNet-5 (*Figure 1*), with particular attention to the choice of activation function, i.e.: **Softmax** in the output layer. This is because, unlike Sigmoid, Softmax generates a normalized probability distribution over the two classes (*bird* and *cat*), which suits best for single-label classification.

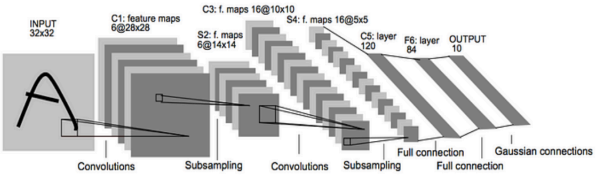

Figure 1: LeNet-5 Architecture

Aditionally, in *eval_performance* function, model's accuracy and calibration is evaluated using Platt's Scaling, which adjusts logits by a scaling factor **a** before applying **Softmax** to refine confidence estimates. The predicted class is determined using *torch.max* , and accuracy is computed by comparing predictions with true labels[1].

3. Calibration Evaluation

Regarding the calibration evaluation, 3 more functions were implemented:

The *plot_reliability_diagram* function visualizes model calibration by plotting a **Reliability Diagram**, comparing predicted confidence levels with actual accuracy across bins[1].

The *ece* function computes the **Expected Calibration Error (ECE)**, quantifying the mismatch between confidence and accuracy by averaging their differences across bins[1].

Finally, the *plot_loss* function tracks model training progress by plotting **training and validation loss curves**, helping to identify overfitting or underfitting trends.

---

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

We trained three different models, initially without regularization, but this approach resulted in overfitting. To address this, we implemented a second model with **batch normalization** and a third model with **batch normalization and dropout**. After comparing their performance (ECE and Accuracy), we selected the model incorporating both batch normalization and dropout as the final one.

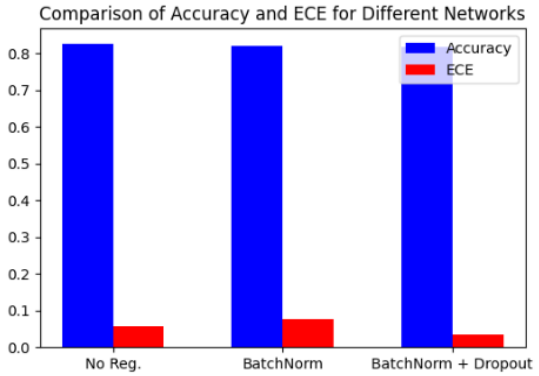

Figure 2: ECE and Accuracy Comparison between our 3 models

In addition to preventing overfitting, this model achieved the lowest validation loss after training, reaching 0.4157. Furthermore, it demonstrates a low Expected Calibration Error (ECE), as evidenced by the reliability diagram.

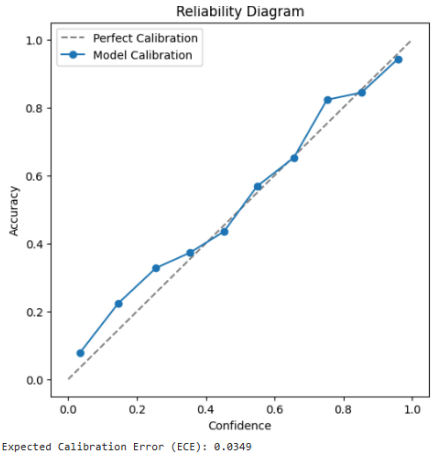

Figure 3: Reliability Diagram and ECE of BatchNorm + Dropout Model

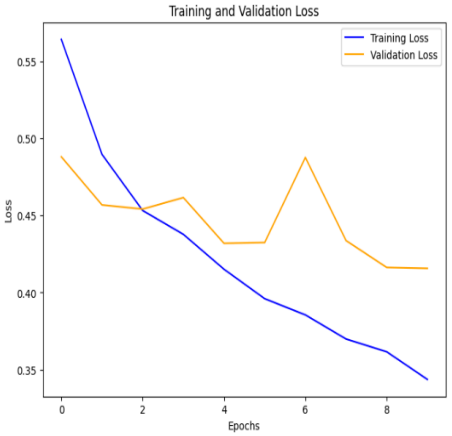

Figure 4: Training and Validation Loss using BatchNorm + Dropout

4. Temperature Scaling

To improve calibration, we apply temperature scaling by adjusting the logits with a scaling factor **a**. This helps refine the model's confidence estimates, reducing overconfidence. We evaluate its impact by analyzing reliability diagrams and Expected Calibration Error (ECE) across different values of **a**, selecting the optimal one based on the lowest Negative Log-Likelihood (NLL) on the test set.

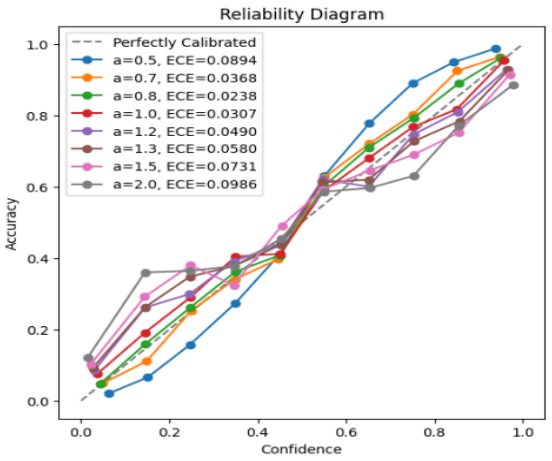

Figure 5: Reliability Diagram for multiple values of scaling factor

When **a**<1 and **a** → 0, the calibration curve bends upwards for higher confidence predictions, causing the predicted probabilities to collapse toward a point mass (i.e., p=1). This means the model becomes overconfident, assigning near-certain probabilities to its predictions.

Conversely, when **a**>1 and **a** → ∞, the opposite occurs: the calibration curve for higher confidence predictions bends downwards, and the predicted probability converges to p=1/K – where K is the number of classes — in our case, p=0.5. This makes the model underconfident, as it predicts both classes with similar probabilities, regardless of input.

So depending on the situation we are in, we should optimize our confidence that **a** 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 **a** < 1.

A minimal ECE value is found around 0. 0238, obtained at the sweetspot around **a = 0. 8**.

5. Fine-tuning a Pre-Trained Model: VGG-16

In order to be able to work with with **VGG-16** for our binary classification task, we needed to implement a series of modifications to the pretrained model, that were:

- **Freezing pre-trained weights,** to ensure that only the new classifier layers are trained, preserving the learned feature representations from the pretrained.
- **Replacing the Classifier.** We implemented a MLP classifier, with the final output layer producing two logits (one for each class).
- **Transformations.** Since VGG-16 was trained on images of size 224x224, we resized the CIFAR-10 images (originally 32x32). We also applied normalization using the same mean and standard deviation as the original VGG-16.

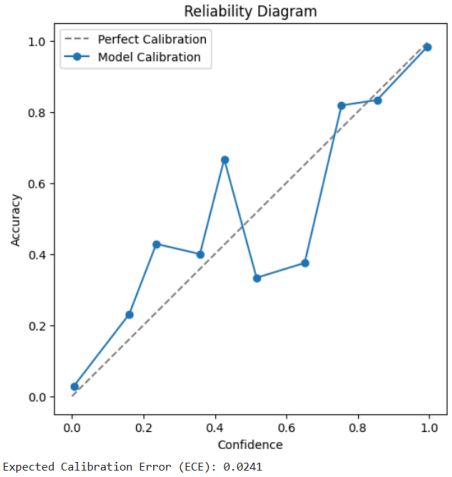

Figure 6. Reliability Diagram VGG16 Model without Platt Scaling

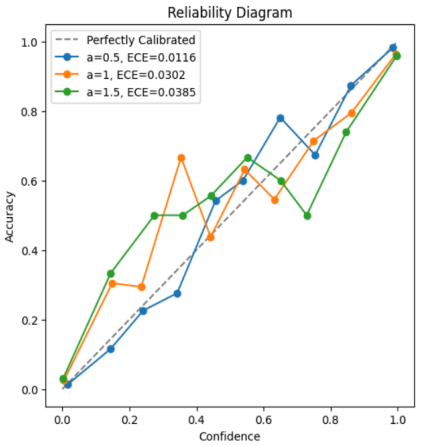

Figure 7. VGG16 Reliability Diagram for different scaling values

Lowering a (e.g., to 0.5) accentuates the difference between logits, increasing the relative confidence of the most likely class. This scaling operation brings the predicted probabilities closer to the actual accuracy, resulting in a lower ECE as it's shown.

VGG-16 outperforms LeNet-5 since it has a significantly larger number of parameters that allows it to capture more complex patterns. This combined with the use of a pre-trained VGG-16 model allows leveraging features learned from a bigger dataset, allows to achieve better results even though we fine-tune it for binary classification.