Applying Temperature Scaling to Reduce DNN Miscalibration

K.M. Kalani Hasantha

199471X

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

Faculty of Information Technology

University of Moratuwa Sri Lanka

March 2022

Applying Temperature Scaling to Reduce DNN Miscalibration

K.M. Kalani Hasantha

199471X

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

Faculty of Information Technology

University of Moratuwa Sri Lanka

March 2022

Declaration

I declare that this is my work and this thesis does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic, or another medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of Student:
Kalani Hasantha
Signature of Student: Date:

The above candidate has carried out research for the Master's Dissertation under my supervision.
Name of Supervisor:
Dr. Subha Fernando

Signature of Supervisor: Date:

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr. Subha Fernando for her immense support and guidance throughout the research. This research would not have been completed without her extensive expertise and continuous support. Further, I would like to thank all academic staff of the Department of Computational Mathematics for providing sufficient knowledge to complete this research.

Finally, I wish to express my sincere thanks to my family and colleagues for their support and encouragement.

Abstract

Deep Neural Networks (DNNs) exhibit state of art performance in a wide variety of fields. However, with the modern architectural trends and increased network capacity, DNN tends to be poorly calibrated. Only well-calibrated models can make probabilistic predictions that reflect real-world probabilities. This issue limits the usage of DNN base models in critical decision scenarios. For achieving a reliable probability score for pre-trained models, several techniques have been introduced including histogram binning, Bayesian Binning into Quantiles, isotonic regression, and Platt scaling. Temperature scaling (TS) was recently introduced, which achieves superior performance compared to the other techniques. But TS does not work properly with small validation datasets, highly accurate networks, and non-highly accurate ones. Therefore, DNN calibration remains a research challenge.

This research focuses on improving the performance of DNN calibration by introducing changes to the state of art post-hoc calibration technique, Temperature Scaling. To overcome the limitations of the Temperature Scaling technique, the novel Class-wise Temperature Scaling technique is proposed. The proposed technique calculates the optimum temperature for each class by binning the predictions on class and the confidence values are rescaled using class-wise temperature values. Class-wise Temperature Scaling can further minimize calibration error in classification models, up to 10% than the standard Temperature scaling technique.

Table of Content

List of Figures

List of Tables

Abbreviation

Abbreviation	Definition
DNN	Deep Neural Network
CNN	Convolutional Neural Networks
ECE	Expected Calibration Error
MCE	Maximum Calibration Error
NLL	Negative Log Likelihood
TS	Temperature Scaling
ATS	Attended temperature scaling
BTS	Bin-wise temperature scaling
UTS	Unsupervised temperature scaling
PTS	Parameterized temperature scaling

Chapter 1 Introduction

1.1 Prolegomena

Miscalibration is a common issue in most of recent Deep Neural Network models. Calibration in Deep learning means that the model's prediction probabilities reflect the true probabilities of the outcome. Model calibration is an important measure when it comes to high risk and safety critical applications. Temperature Scaling is state of art post-hoc calibration technique, which is comparatively effective at minimizing miscalibration. In this paper, Temperature Scaling is extended by introducing class-wise Temperature scaling approach.

1.2 Background and motivation

In recent years, Deep Neural Networks (DNNs) based classifiers have been achieved state of art performance in a wide variety of applications. With the increased model size and architectural complexity, these models tend to be poorly calibrated despite their higher accuracies (Guo et al.,2017 [1]). Intuitively, calibration in neural network's predictions means that if a model assigns a class with 80% probability, the model should correctly identify the 8 images out of 10 images from the class. If the model confidence is less than the accuracy, then the model is under confidence. On the contrary, the model is overconfident if model confidence is higher than the accuracy of the model. This uncertainty in the model's probabilistic predictions limits the usage of DNNs in critical decision scenarios like medical applications, autonomous vehicles, and safety-critical applications.

Several preprocessing (regularization, data augmentation, variational methods, ensembles) and post-processing (histogram binning, isotonic regression, Platt scaling, temperature scaling) techniques have been introduced for achieving a reliable probability score. Temperature scaling was recently introduced and achieves superior performance compared to the other recalibration techniques (Alzubaidi et al.,2021 [2]). Temperature scaling (TS) does not require model retraining and it can calibrate the model probabilities by finding one optimal parameter (temperature) using sample data set, which reflects the actual data distribution. Temperature(T) is a scalar parameter that is used to soften the probability distribution over the classes. The best advantage of the temperature scaling technique is parameter T does not change the maximum of the SoftMax function. So, the class predictions remain unchanged.

But TS does not work properly with small validation datasets, out-of-distribution data sets, highly accurate networks, and non-highly accurate ones. Further probability binning-based calibration measuring matrices can be misleading sometimes due to fewer data points for the low probabilities. Therefore, DNN calibration remains a research challenge.

This research focuses on improving the performance of temperature scaling by introducing a novel approach that leads to finding the optimal temperature values for recalibrating model confidence values.

1.3 Aims and objectives

This research aims to adapt temperature scaling to reduce the miscalibration of Deep Neural Networks. To reach this aim, the following objectives have been defined.

• A critical review of the latest research on neural network calibration.

• In-depth study of technologies used for Temperature scaling.

• Design and develop an approach to improve the temperature scaling technique.

• Evaluate the proposed approach with benchmark datasets.

1.4 Problem definition

Recent deep neural networks are not well calibrated and this limits the usage of DNN in critical decision scenarios. Although Temperature scaling approaches have reduced the expected calibration error, it does not work properly with small validation datasets, highly accurate networks, and non-highly accurate ones. Further, existing approaches do not address calibrating each class in a classifier separately. Instead, these approaches calculate one temperature parameter addressing all the classes which may do not calibrate probability scores of some classes. In addition, current calibration measuring matrices like Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) are based on probability binning, and this approach partitions all highest prediction scores into equal-width bins between zero and one. Since the predicted class have higher probabilities, bins in lower probability regions will be sparsely populated and bins in higher probability regions will be densely populated. This leads only a few bins (between 0.4 -1)

to contribute to the Expected Calibration Error. In this scenario calculating one Calibration error for all the classes can be misleading. Therefore, DNN calibration remains a research challenge.

1.5 Class-wise Temperature Scaling

This research focuses on minimizing expected and maximum calibration error by applying temperature scaling. It is a three-step process. In the first step, model confidence scores will be binned based on predicted classes instead of prediction probabilities. The second step is to calculate the optimum temperature value for each class in the classifier using the logit vector (neural network output before applying the SoftMax or Sigmoid function) of the model. The third step is rescaling the logit values with the calculated temperatures for each class. ECE and MCE can be calculated for each class after scaling logit values.

1.6 Resource requirements

This research requires a high GPU, capable of training a deep neural network. GPU is not required for calculating optimum temperatures.

1.7 Structure of the thesis

The rest of the thesis is outlined as follows. Chapter 2 is dedicated to reviewing the other research work in the domain of DNN Calibration. The third section describes the core theories, definitions, and techniques used in the project. In the approach chapter overview of the proposed solution is described. In the next chapter, the design of the proposed solution is presented. The implementation chapter describes the development process and the results are evaluated in the next chapter. The final chapter discusses the overall conclusion, limitations, and further developments of the proposed solution.

1.8 Summary

This chapter presented a brief description of aims, objectives, the background of the problem, motivation for selecting the problem, problem definition, and the proposed solution. In the next chapter, a detailed discussion will be given on the development and issues in DNN calibration approaches.

Chapter 2 Uncertainty Scaling

2.1 Introduction

Chapter 1 provided an overall picture of the research presented in this thesis. This chapter gives a critical review of the literature in neural network calibration. In doing so, we have structured the literature review under several subheadings, namely, early approaches of uncertainty scaling, a breakthrough in DNN calibration techniques, challenges in DNN calibration techniques, and problem definition.

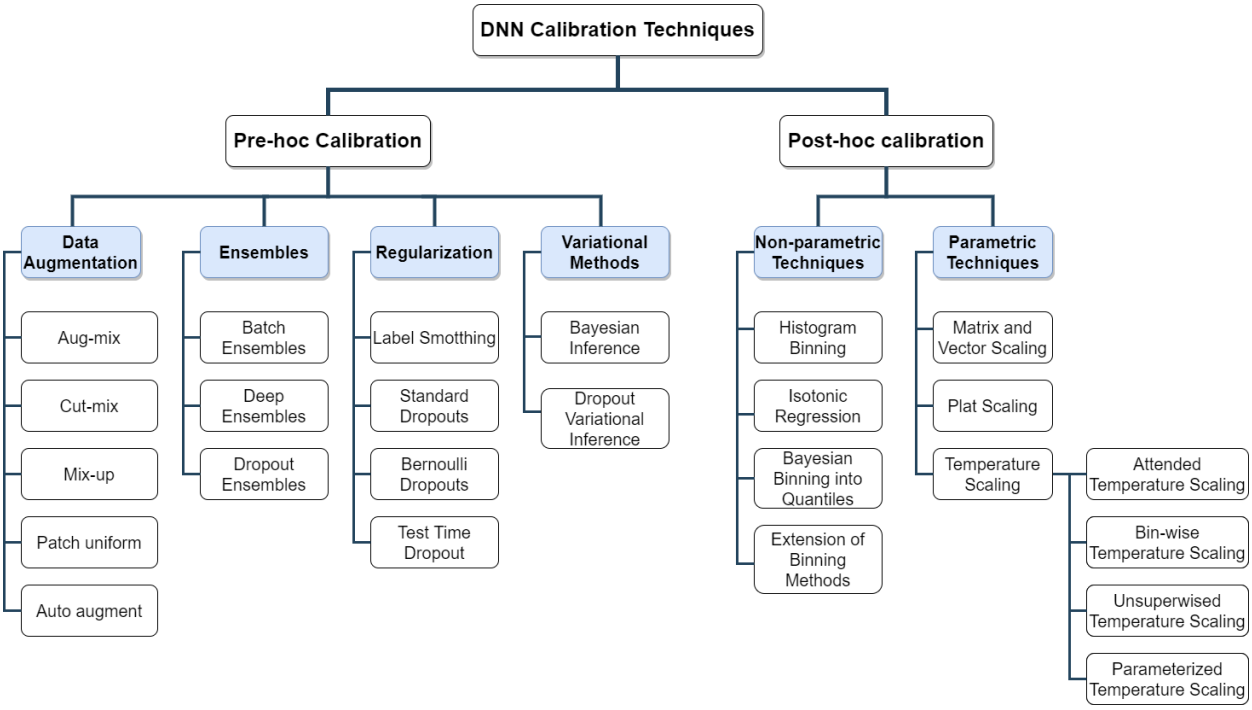

Figure 2. 1 Overview of DNN Calibration Techniques

2.2 Early approaches of uncertainty scaling

Understanding whether a model's predictions are trustworthy is crucially important in any machine learning application. Model to be trustworthy and interpretable, model's confidence should reflect the true correctness likelihood. The output of the SoftMax layer of a neural network is considered as the confidence of a model [6]. Miscalibration is the dissimilarity between model confidence and

accuracy. Evaluating and eliminating calibration errors in neural network-based classifiers has been a research topic for a decade.

Platt Scaling is the one of earliest approaches to minimize the calibration in machine learning models. Platt et al., 1999 [5] proposed a parametric post processing approach to calibrate the probabilities in Support Vector Machine (SVM). They first trained the SVM model and then trained the scalar parameter of an additional sigmoid function to map the SVM output into calibrated probabilities. In context of neural networks, Plat Scaling learns scalar parameters a and b. Calibrated output can be defined as $\hat{q}i = \sigma(a * zi + b)$, where zi is the unscaled prediction probability. Parameters a and b are optimized using the NLL loss over the validation set. Since Platt scaling is a post-processing calibration technique, it can be used to calibrate already trained classifiers. But it is only limited to binary classifications. The other limitation of Platt scaling is, it only works when the distortion in the predicted probabilities is sigmoid shaped.

Since Platt scaling is only used for S-shaped distortions, Histogram Binning was introduced to overcome this limitation. Histogram binning (Zadrozny & Elkan, 2001 [18]) is a binary and non-parametric calibration method. It divides unscaled predictions into mutually exclusive m number of bins and assigns a calibration score to each bin. Then the predictions are scaled using in a way that minimizes the bin-wise squared loss. This approach has several limitations including defining the number of bins and boundaries. Since boundaries associated with bins are fixed, calibration scores can be biased and highly suffer from sampling issues.

So as to overcome the sampling issues in Histogram Binnig, Isotonic Regression and Bayesian Binning were introduced. Isotonic regression (Zadrozny and Elkan, 2002 [19]) is a generalization of histogram binning in which bin predictions and bin boundaries are optimized jointly. It is a non-parametric post-processing calibration approach and is mostly used to obtain the monotonic fit for one-dimensional data. It is used for recalibrating binary classification models by minimizing the squared loss. Although Isotonic regression is an effective calibration approach for any monotonic distortion, it is more prone to overfitting and the performance is very low when data is scarce.

Naeini et al.,2015 [24] proposed Bayesian Binning into Quantiles (BBQ) as an extension of histogram binning. BBQ considers multiple binning models with different bins and uses Bayesian model averaging instead of simple histogram binning. The main challenge in BBQ is how to select

the model and how to select the averaging methods. Although the BBQ approach could overcome the bias issue in Histogram binning, this approach cannot be used for multi-class classifiers.

2.3 Breakthrough in DNN calibration techniques

Most of the earliest approaches in the miscalibration domain were focused on statistical machine learning models and were not applicable to any neural network model. But miscalibration in the complex and deep neural network architectures continued to increase. So, more investigations were continued in this domain so as to find generalized and optimized calibration solutions. Mizil and Caruana, 2015 [7] investigated calibration in binary classifiers and concluded that binary classifiers are well-calibrated when compared to multi-class classifications. Zhang et al., 2017[23] showed that very wide or deep models easily overfit the training data than the smaller models. Although increased width and depth could reduce the model classification error, it increases the model miscalibration. Guo et al.,2017 [1] experimentally proved that modern deep neural networks are not well calibrated, although they tend to display higher accuracy. The average confidence of 5-layer LeNet [16] on CIFAR-100 closely matches its accuracy, while the average confidence of the 110-layer ResNet [17] is substantially higher than its accuracy. They found that increased model capacity, batch normalization, and lack of regularization are the main causes for neural network miscalibration.

Pereyra et al., 2017[22] discovered that confidence penalty and label smoothing improve the model confidence without changing existing hyperparameters by systematically exploring with 6 benchmark datasets on state of art deep learning models.

Temperature scaling was proposed addressing the limitations of the earliest calibration approaches. Temperature scaling (TS) is a single parameter variant of Platt Scaling. It is a post-processing technique and achieves superior performance compared to the other techniques (Alzubaidi et al, 2021 [2]). TS has been used to distill knowledge (Hinton et al., 2015 [3]) and calibrate DNN (Guo et al, 2017 [1]). For classification tasks, the neural network outputs a vector known as the logits. TS rescales this logit vector to calibrate the neural network. TS uses a single scalar parameter T >

0 for all classes, which is called temperature. For a given logit vector Zi, TS calculate new confidence qi using the following equation:

$$\mathrm{q}i = \frac{\exp(Zi/T)}{\sum_j \exp\,(Zj/T)}$$

TS softens the neural network outputs. This makes the model's predictions slightly less confident, which makes the confidence scores reflect true probabilities. The Optimum T parameter is found by minimizing NLL with respect to T on the validation set. Since TS learns only one parameter from the validation set, the calibration process is more concise and efficient. The greatest benefit of TS is that it does not affect the accuracy of the classifier since it does not change the maximum of SoftMax function. So, the prediction accuracy remains unchanged. Although TS is a very effective method for confidence scaling, it does not perform well with small, out of distribution and noisy datasets. Also, this approach fails to calibrate highly accurate networks as well as non-highly accurate networks.

Multiple approaches have been proposed to find the optimal T value including Attended temperature scaling (ATS), Bin-wise temperature scaling (BTS), Unsupervised temperature scaling (UTS) and Parameterized temperature scaling (PTS). These approaches address the limitations of TS. ATS decreases the dissimilarity between the true conditional distribution of data and calibrated probability by decreasing NLL according to each class distribution whereas TS decreases NLL according to total distribution (Mozafari et al, 2019 [8]). ATS sample data into subsets using true labels and add extra data points to the classes which are located near the boundary. The subsampling method was introduced to increase the data points in small validation sets. But this approach can add noise to the calibration process and because of that reason, ATS does not work well with noisy and imbalanced datasets.

So as to overcome the biased temperature calculation BTS was proposed. BTS divides the validation set into multiple bins and calculates the temperature for each bin (Ji et al, 2019 [9]). In this approach, optimum temperature values for each bin can be changed according to the number of bins used. Same as all other temperature scaling approaches, BTS divides predictions into bins using confidence values. Most of the time the bins with a low confidence range will have few or zero data points. Because of this reason low prediction values will be not well scaled using BTS. So, this approach is very unstable for low confidence probabilities.

UTS utilizes a novel loss function, weighted NLL, which allows unsupervised calibration (Mozafari et al, 2019 [10]). PTS introduces a dependency of the temperature on the logit tuple itself and proposes a new loss function inspired by expected calibration error (Tomani et al, 2021 [12]). In PTS we have to train an extra neural network to find the optimum temperature, which does not cost and time-effective as a post-processing calibration method. In addition, this approach is more prone to overfitting and does not work well with a small validation dataset.

All the previously described approaches focus on recalibrating pre-trained models. Achieving more calibrated probabilities in the training process is time and cost-effective. So, there was a need for preprocessing calibration techniques. As a solution for this, many Preprocessing uncertainty scaling strategies have been introduced recently including Data augmentation, Regularization, Variational methods, and Ensembles.

Calibration error can be reduced by increasing the accuracy of the model. Higher accuracy can be obtained by retraining the model with more data samples from each class. When we consider image classifiers, image augmentation techniques can be used to increase the training data set size while helping the model withstand unforeseen corruptions. Aug-mix (Hendrycks et al.,2020 [20]), Mix-up (Liang et al., 2021 [21]), Cut-mix (Yun et al., 2019 [24]) and Auto augment (Cubuk et al., 2019[25]) are some recently introduced image augmentation techniques, which can be used to augment training data. These approaches can be only used with image classifiers and cannot be used with language models. In addition to this limitation, models should be retrained to use augmentation techniques and the model retraining process can change the model accuracy.

Since augmentation techniques were only applicable for vision-based models, more approaches were proposed as calibration solutions. At present, deep ensembles are used to increase the accuracy of the models and also to achieve calibrated outputs. The deep Ensemble method is a machine learning technique where we train multiple neural networks individually and average model predictions. This technique is proved widely successful for improving prediction accuracy and miscalibration. Batch ensembles (Wen et al., 2020 [26]), Deep ensembles (Lakshminarayanan et al., 2016 [27]), and Dropout ensembles (Hara et al., 2016[28], Laakom et al.,2021[29]) are commonly used ensemble methods for improving predictive uncertainty. However, the cost of training and testing an ensemble increase linearly with the number of networks used.

It is found that regularization techniques can be used to minimize the miscalibration in DNN. Label smoothing is a commonly used regularization technique that can smooth output probabilities. Label smoothing is a regularization technique that can be used to smooth the overconfident probabilities of a poorly generalized classification model. Despite having a positive effect on calibration and generalization, label smoothing can hurt distillation because of the erasure of information (Müller et al., 2019[31]).

According to the experimental study of We et al, 2021[30] combination of data augmentation and ensembles can harm model calibration. But combining any pre-hoc calibration techniques with post-hoc calibration technique, Temperature scaling can further improve the model calibration. In a recent study Minderer et al.,2021 [32] show that temperature scaling is the simplest and most cost-effective post-hoc calibration method to improve the miscalibration in large trained networks. In addition, the study shows that recent non-convolution models like MLP-Mixer and Vision transformers are well-calibrated despite their number of layers and the complicity of the model architecture. Using a systematic comparison of recent classification models, they suggest that the model architecture tends to show a higher effect on model uncertainty.

2.4 Challenges in DNN calibration

Pre-hoc calibration techniques can be only used when training a new model. Retraining a model with preprocessed training data can cause accuracy changes in both positive and negative ways. This is the main challenge in pre-hoc calibration techniques. In addition, retaining a model which already has good accuracy is not time and cost-effective when it comes to models which have millions of parameters.

Post-hoc calibration techniques are more useful for calibrating already trained models. These techniques are cost and time effective since these techniques require optimizing a few parameters over a sample validation set. Although Temperature scaling is more effective when compared to other post-hoc calibration techniques, it does not work well with small and noisy validation data set. Also, TS techniques cannot rescale the confidence values of highly accurate networks as well as non-highly accurate ones.

One of the main challenges in confidence calibration is the inability to well calibrate low confidence values. Any of the proposed approaches could not achieve perfect calibration, which means both expected and maximum calibration error is zero.

Technique	Benefits	Limits
Isotonic regression	Effective calibration for any monotonic distortion.	More prone to overfitting. Not effective when data is scarce. Can be only used for binary classifiers.
Platt scaling	Only two parameters will be learned from validation data.	Can be only used for binary classifiers. Only works when the distortion in the predicted probabilities is sigmoid shaped.
Temperature scaling	Only one parameter will be learned from validation data. Accuracy remains unchanged through the calibration process	Does not work well with a small and noisy validation set. Cannot calibrate highly accurate networks as well as non-highly accurate ones.
Attended temperature scaling	Work well with small validation data set.	Does not work well with noisy and imbalanced data. Cannot calibrate highly accurate networks as well as non-highly accurate ones.
Bin-wise Temperature scaling	Reduce the bias toward high-confidence samples.	Unstable for low confidence probabilities.
Unsupervised temperature scaling	Work well with small validation data set.	Does not work well with noisy and imbalanced data.
Parameterized temperature scaling	Better expected calibration error.	Does not work well with small and noisy data. More prone to overfitting. Need to train an extra neural network for predicting temperature.

Table 2.4.1 summary of benefits and limitations of post-hoc calibration techniques

2.5 Problem definition

Recent deep neural networks are not well calibrated and this limits the usage of DNN in critical decision scenarios. Although Temperature scaling methods have reduced the expected calibration error, it does not work properly with small validation datasets, highly accurate networks, and non-highly accurate ones. In addition, higher confidence values show more contribution towards optimum Temperature. So, the low probabilities will not be well scaled using the optimized temperature value. Therefore, DNN calibration still remains a research challenge, and thus a more optimized calibration approach is required.

2.6 Summary

This chapter presented a critical review of literature in DNN calibration and identify achievements in the field and research challenges. In the next chapter, we give a detailed discussion on the theories and technologies adopted for implementing a novel calibration approach using the temperature scaling technique.

Chapter 3 Definitions and Technologies

3.1 Introduction

In chapter 2 we have presented a comprehensive critical review of literature on the use of DNN calibration techniques. Among them, temperature scaling-based techniques are widely discussed. In this chapter, we describe the background theories and the DNN architectures which were used for the process of evaluating existing techniques and experimenting with novel temperature scaling-based approaches.

3.2 Reliability Diagram

A reliability diagram is a simple way of visualizing model calibration by plotting expected accuracy as a function of confidence. Since confidence should reflect the accuracy, perfectly calibrated models should plot the identity function for the reliability diagram. Any deviation from the identity line (perfect diagonal) represents miscalibration in models.

To calculate expected accuracy and the confidence of finite data samples, prediction probabilities should be grouped into m number of bins. Each bin size will be 1/m (If the number of bins is three, bin probability ranges will be [0 - 0.33], [0.33 – 0.66], [0.66-1] and each prediction will be grouped into these three bins according to the probability). In practice, 10 -20 bin sizes are used (Nixon et al.,2022 [13]).

Figure 2.2.1 visualizes the reliability diagram of predictions of the Resnet50 architecture model, which was trained on the cifar10 data set. Here the used bin size is 10 and the gray color diagonal line is the identity line. In this diagram, the confidence values of most of the bins are higher than the expected accuracy except for bins 2 and 3. Overall this model predictions display overconfidence. Figure 2.2.2 visualize the same reliability graph using a line chart. This graph clearly displays the gap between expected accuracy and actual confidence.

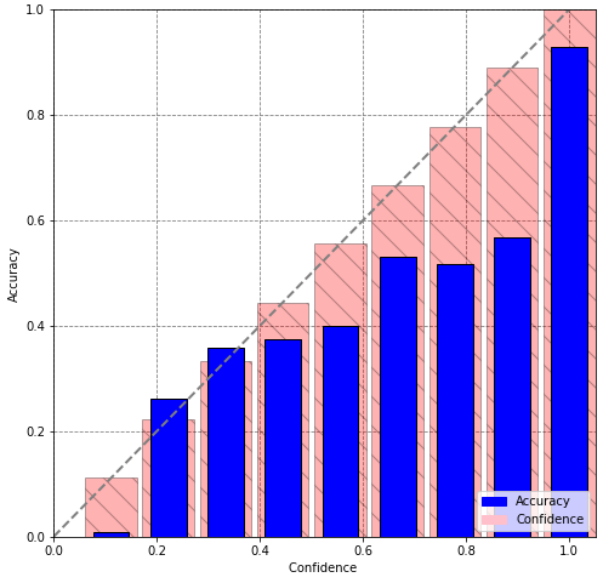

Figure 3.2. 1 Reliability diagram (bar chart)

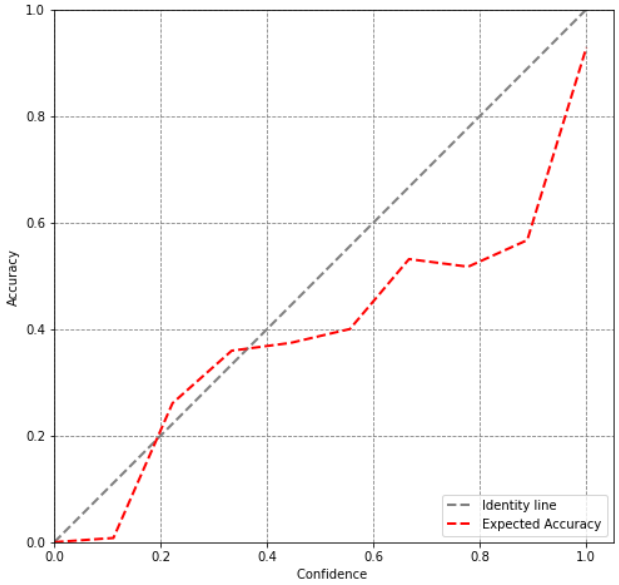

Figure 3.2. 2 Reliability diagram (line chart)

3.3 Expected Sample Accuracy

Expected sample accuracy is the proportion of true outcomes in a subset. Simply we can define bin accuracy as the number of correct predictions in a bin out of all items in the bin. Let yi is the true class label, $\hat{y}$i is the predicted class label and Bm be the equally spaced confidence bin for sample i. We can define the accuracy of the bin Bm as

$$Acc\ (Bm) = \frac{1}{|\ Bm|} \sum_{i \in Bm} 1(\hat{y}i = yi),$$

3.4 Average Sample Confidence

Average sample confidence is equal to the mean prediction value in a subset. We can calculate bin confidence by dividing the summation of predictions by the number of items in the bin. Let $\hat{P}$i be the confidence for sample i. We can define the average confidence of the bin Bm as

$$Conf\ (Bm) = \frac{1}{|\ Bm|} \sum_{i \in Bm} \hat{P}i,$$

3.5 Measurements of Model Calibration

While the reliability diagram is the visual representation of model calibration, Expected Calibration Error and Maximum Calibration Error express the scalar summary statistic of model calibration (Geo et al., 2017 [1]).

Suppose $\hat{Y}$ is a class prediction and $\hat{P}$ is the it's associated confidence value. Model to be well-calibrated, $\hat{P}$ should represents a true probability. Formally we can define perfect calibration as

$$\boldsymbol{P}\left(\hat{Y} = y \,\middle|\, \hat{P} = p\right) = p, \qquad \forall p \in [0, 1]$$

Here, the left-hand side of the equation denotes the true data distribution's probability of a class. The right-hand side of the equation denotes the model prediction value. The difference between the left side and right sides for a given p (expected disagreement) is known as the calibration error of a model (Nixon et al.,2022 [13]).

3.5.2 Maximum Calibration Error (MCE)

Similar to ECE, MCE is calculated using binning prediction probabilities. MCE is the largest calibration gap across all bins, while ECE is the weighted average of all calibration gaps. Similar to ECE for a perfectly calibrated classifier, MCE equals zero. MCE is mostly used in high-risk applications, where we need to minimize the worst-case deviation between accuracy and confidence. MCE can be defined as

$$MCE = max_{\,m \in \{1,..,M\}} |\ acc(Bm) - conf(Bm)|$$

3.5.1 Expected Calibration Error (ECE)

ECE is calculated by partitioning model predictions into M equally spaced bins and then taking the weighted average of the calibration gap in each bin. For a given bin calibration gap is the difference between the accuracy and the confidence for that bin (Naeini et al.,2015 [14]). For a perfectly calibrated classifier model, ECE equals zero.

Let N be the total number of data points and n_b is the number of predictions in bin B_m. ECE can be defined as

$$ECE = \sum_{m=1}^{M} \frac{nb}{N} \; | acc(Bm) - conf(Bm) |$$

3.5.3 Negative Log-Likelihood (NLL)

The quality of a probabilistic model can be measured using NLL by maximizing the probability of data distribution of the model. In the context of deep learning NLL is referred as the cross-entropy loss (Goodfellow et al.,2016 [4]). Lower NLL reflects a more calibrated model and higher NLL means that model is not well calibrated.

Assume Q (x, y) is the distribution function of the train data set and $S_y(x)$ is the SoftMax output function of a model. The output of the SoftMax function represents the probability distribution of a list of potential outcomes. Using NLL as a loss function, we can reduce the dissimilarity between $S_y(x)$ and true conditional distribution Q (y | x). NLL can be defined as

$$NLL = -\sum_{(xi,yi)} \log\left(S_{y=yi}(xi)\right), \qquad (xi, yi) \sim Q(x, y)$$

3.6 Neural Network Architectures

LeNet-5, VGG16, VGG19, ResNet and DenseNet architectures were used to evaluate the calibration techniques in the research process.

3.6.1 LeNet-5

LeNet-5(LeCun et al.,1998 [33]) is one of the earliest Convolutional Neural Network (CNN) architecture which only has 7 layers including 3 convolutional layers, 2 average pooling layers, and 2 fully connected layers. The network has 5 layers with learnable parameters and hence named Lenet-5. This simple network contains only 60,000 trainable parameters and has displayed low calibration error.

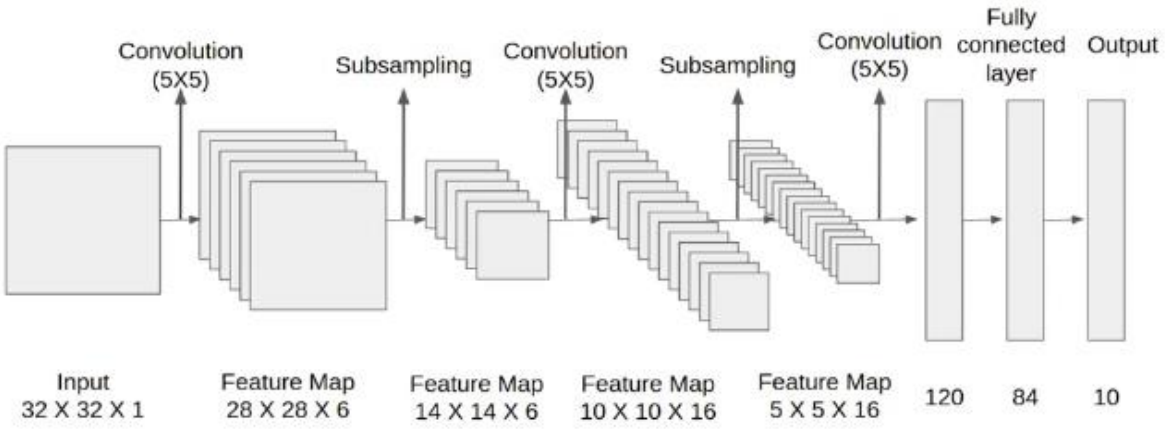

Figure 3.6.1 LeNet5 architecture diagram

Source: https://cdn.analyticsvidhya.com/wp-content/uploads/2021/03/Screenshot-from-2021-03-18-12-52-17.png

3.6.2 VGGNet

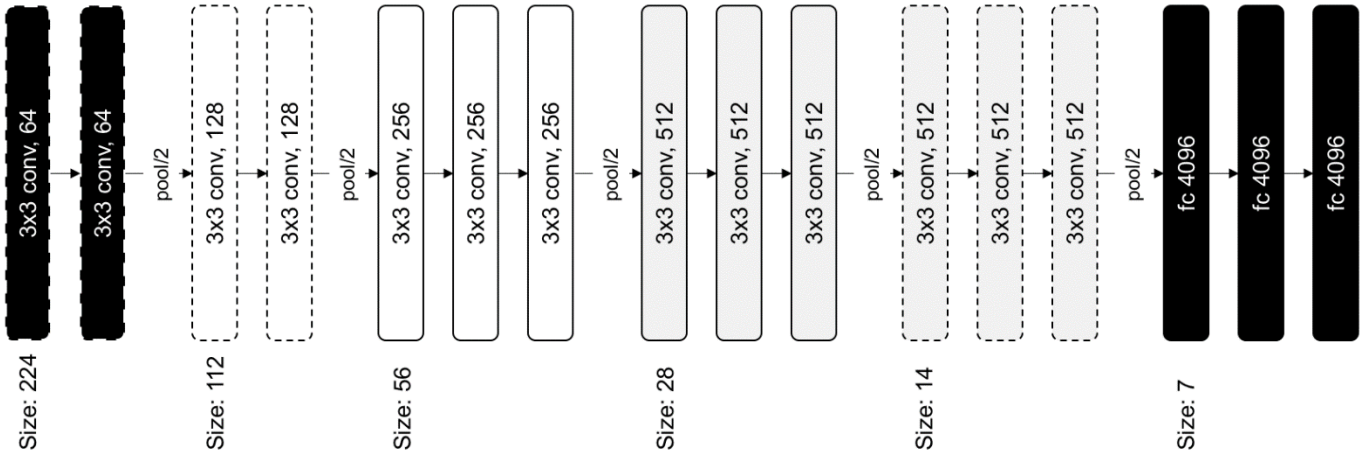

Figure 3.6.2 VGG16 architecture diagram

Source: https://www.oreilly.com/library/view/deep-learning-for/9781788295628/assets/21bcc2d1-25e2-41c6-8303-9e062dfde978.png

VGGNet is a deep CNN architecture proposed by Simonyan and Zisserman,2015 [34]. This model investigates depth of the layers with a very small convolutional filter size (3 × 3) to deal with large-scale images. The authors have released a series of VGG models with different layer lengths from 11 to 19. Among them VGG16 and VGG19 architectures are mostly used. VGG16 consist of 16

trainable layers with 138 million trainable parameters which can be a bit challenging to handle. VGG19 consist of 3 more convolution layers than VGG16.

3.6.3 ResNet

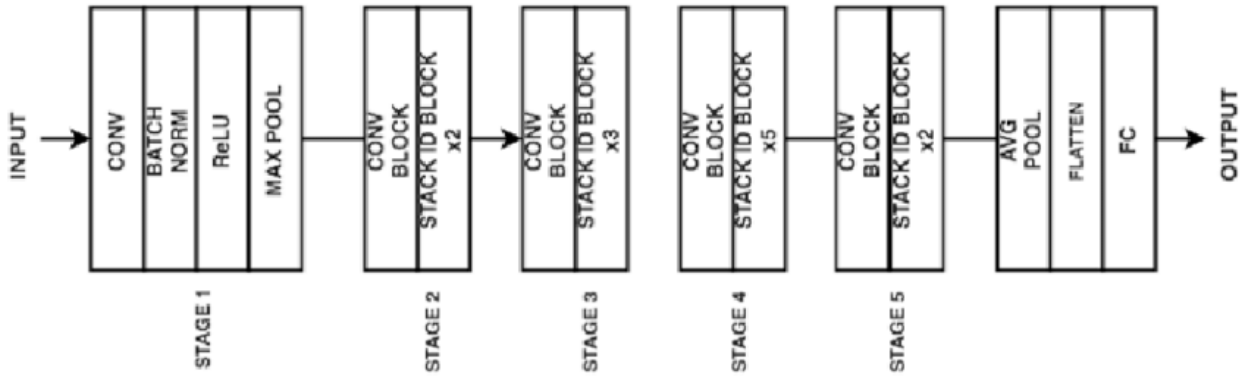

Figure 3.6.3 ResNet50 architecture diagram

Source: https://www.researchgate.net/publication/333521352/figure/fig3/AS:871956081557512@1584901938790/Architecture-of-our-model-ResNet50.png

Residual neural network (ResNet) is proposed by He et al.,2016 [35] as a solution to the vanishing gradient problem in very deep neural networks. As the number of layers increases in the network, training become extremely challenging and it results higher training and testing error. ResNet architecture introduced technique called “skip connections/ residual connections”, which skip training from a few layers and connect directly to output. With the residual blocks, the network performance get better as the number of layers increases. ResNet-34, ResNet-50, ResNet-101, ResNet-152 respectively have 21.8 M, 25.6 M, 44.5 M and 60.2 million parameters.

3.6.4 DenseNet

DenseNet was developed by Huang et al.,2017 [36] especially focusing on making DNN go even deeper, but same time making the network more efficient to train. In DenseNet, each layer is connected directly with every other layer, hence the name Densely Connected Convolutional Network. Dense blocks encourage feature reuse, strengthen feature propagation, and substantially reduce the number of parameters. DenseNet-121, DenseNet-169, DenseNet-201 networks respectively have 7.2M, 12.8M and 20 million parameters. DenseNet-201 with 20M parameters

has displayed a similar validation error as the ResNet-101 with more than 44.5M parameters. So, the DenseNet architecture is a very efficient solution for the vanishing grading problem.

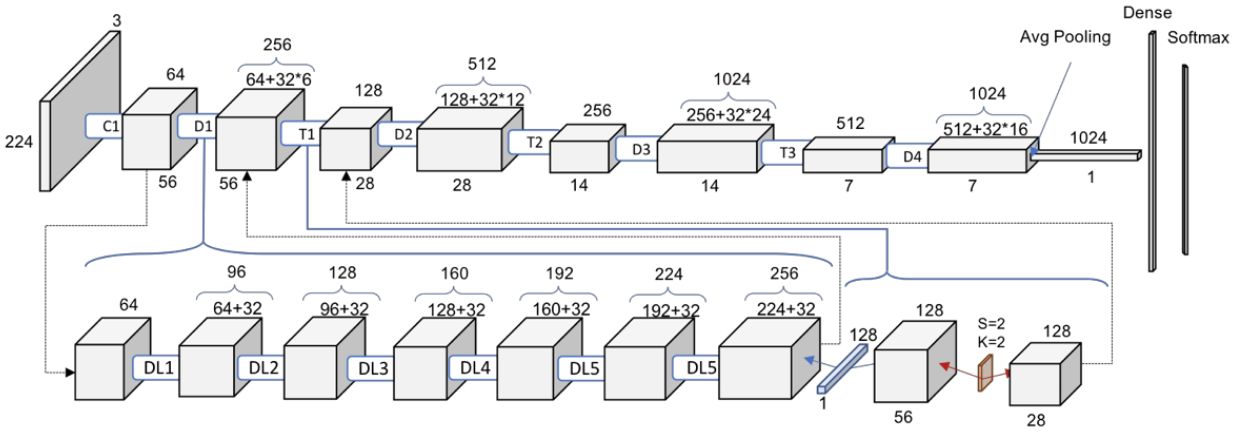

Figure 3.6.4 DenseNet201 architecture diagram (D: Dense Block, T: Transition Block, DLx: Dense Layer x)

Source: https://miro.medium.com/max/1400/1*SsphOqMwglCVGDWB-jdT5Q.png

3.7 Summary

This chapter describes the theories, notations, and technologies that are used in DNN calibration. Reliability diagram is a visual representation of DNN miscalibration and it can be measured using ECE and MCE. LeNet-5, VGGNet, ResNet, and DenseNet which have a different number of layers were used to measure the miscalibration in the research process. In the next chapter, we give a detailed discussion on the proposed approach.

Chapter 4 Class-wise Temperature Scaling Approach

4.1 Introduction

Using the technologies described in Chapter 3, the Class-wise Temperature Scaling approach for deep neural network calibration is proposed as follows.

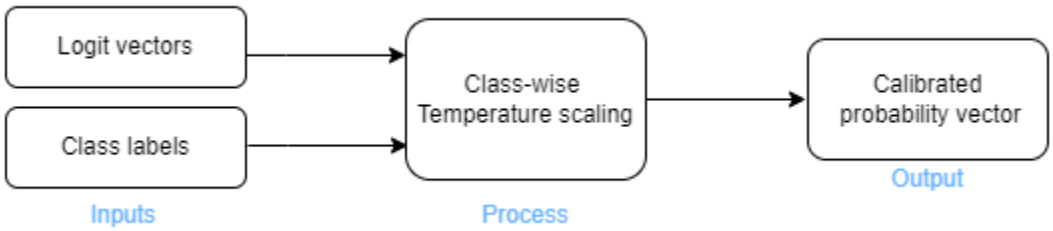

Figure 4.1 Approach

The rest of the chapter has been structured with the subheading hypothesis, input, output, process, users, and features.

4.2 Hypothesis

This research hypothesizes that Calibration errors in Deep Neural Networks can be further reduced by adapting class-wise temperature scaling alone with the class-wise binning approach.

4.3 Input

For classification problems, the neural network output a vector known as the logits. Normally, this logits vector is passed through a SoftMax function to get class probabilities. For the proposed solution, the logit vector and the class labels of the validation dataset will be used as the inputs.

4.4 Output

The output of the system will be the calibrated probability vector and it is obtained by scaling each prediction with the respective temperature value for each class.

4.5 Process

The process includes three main steps. The first step is to train multiple deep learning models with different architectures and analyze the miscalibration of each architecture by using visualization and statistical metrics. The second step is to calibrate the trained model using Temperature Scaling (TS) and identify the problems in TS-based approaches. The third step is to experiment with existing approaches and come up with a novel approach to overcome the identified pitfalls of the TS. This step includes evaluating the proposed approach with the Temperature scaling technique using Expected Calibration Error and Maximum Calibration Error.

4.5.1 Analyzing Miscalibration in DNN

For analyzing miscalibration in DNN, the following models with different architectures and the different number of layers are used, which were trained with the datasets Cifar10 and Cifar100.

Architecture	Number of trainable layers	Number parameters	Topological depth of the network
LeNet-5	5	60 K	7
Vgg-16	16	138.4 M	16
Vgg-19	19	143.7M	19
ResNet-50	50	25.6 M	107
ResNet-101	101	44.7 M	209
ResNet-152	152	60.4 M	331
DenseNet-121	121	8.1 M	242
DenseNet-169	169	14.3 M	338
DenseNet-201	201	20.2 M	402

Table 4.5.1 DNN architecture depth and parameters comparison

Reliability diagrams are used to visualize the miscalibration in these models. According to the analysis with the increasing number of layers and the complexity of the architecture, the gap between the accuracy and the confidence value got increased. Figure 4.5.1 shows the reliability diagrams for LeNet-5, Vgg-16, Resnet-101 and DenseNet-201. According to the diagrams, most of the models displayed higher confidence value than the actual accuracy and the accuracy of the models lie below the identity line. This means models are overconfidence.

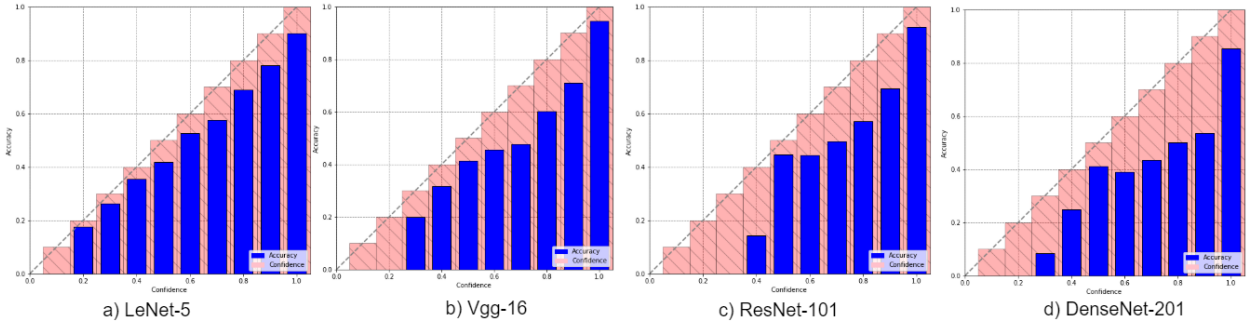

Figure 4.5.1 Reliability diagrams for uncalibrated models trained with Cifar10

4.5.2 Identifying drawbacks of Temperature Scaling

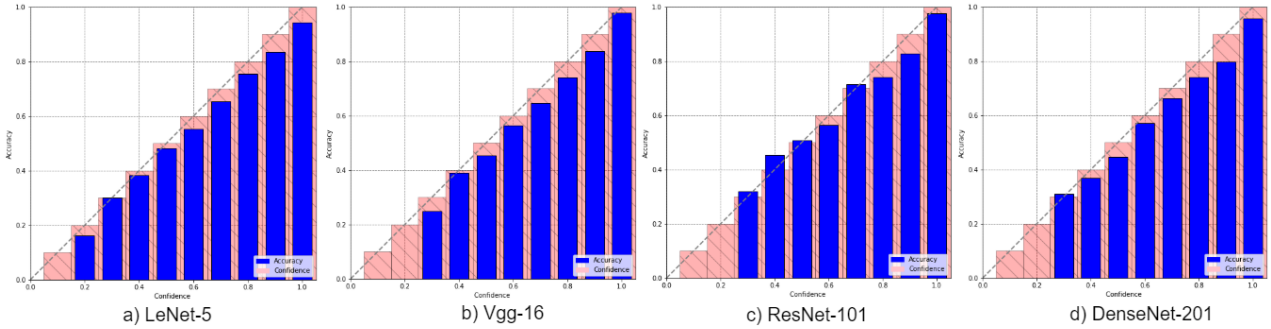

Figure 4.5.2.1 Reliability diagrams after calibrating with TS (models trained with Cifar10)

Figure 4.5.2.1 shows the reliability diagrams after calibrating the model predictions with temperature scaling. Here the gap between the accuracy and the confidence has been reduced and some of the bins have reached the identity line. Since the probability binning approach uses the highest predictions from the SoftMax output vector, most of the low-range probability bins do not have any predictions. So, the number of bins plays a critical role here. Since the calibration error is calculated by averaging the bin confidence gap, empty bins will negatively affect the calibration measures. In addition, the temperature scaling approach results in one optimum temperature value for all the bins or the dataset. So, only some bins are contributed for the optimum temperature value.

Figure 4.5.2.2 shows the Data distribution between each bin for the model predictions (LeNet-5, Vgg-16, Resnet-101 and DenseNet-201 have been selected for the visualization). According to the data distribution, we can see that left-side bins (low probability range) are sparsely populated and the right-side bins (high probability range) are densely populated. With the increased accuracy,

models become more confident about the predictions. So, most of the models have more data points in the bins which lie in the higher probability range.

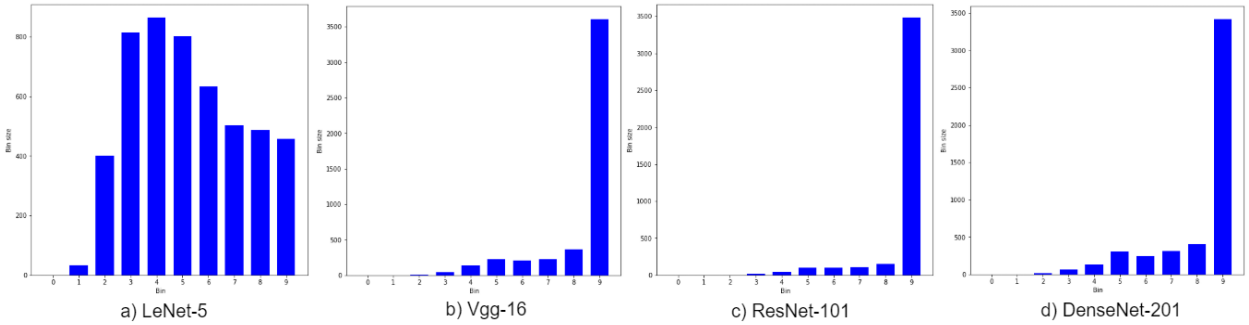

Figure 4.5.2.2 Data distribution between Bins in probability range binning approach

4.5.3 Identifying Approaches to Prevent drawbacks of Temperature Scaling

Since a fair data distribution for each bin is important for calibrating all the confidence values, as the next step we analyzed the confidence value distribution according to the class. Here instead of assigning a probability range for each bin, we assigned data points to the bin according to their predicted class. Figure 4.5.2.3 shows the data distribution for each bin when the predictions are binned based on the predicted class. We can see that the data points are fairly distributed among the bins in this approach.

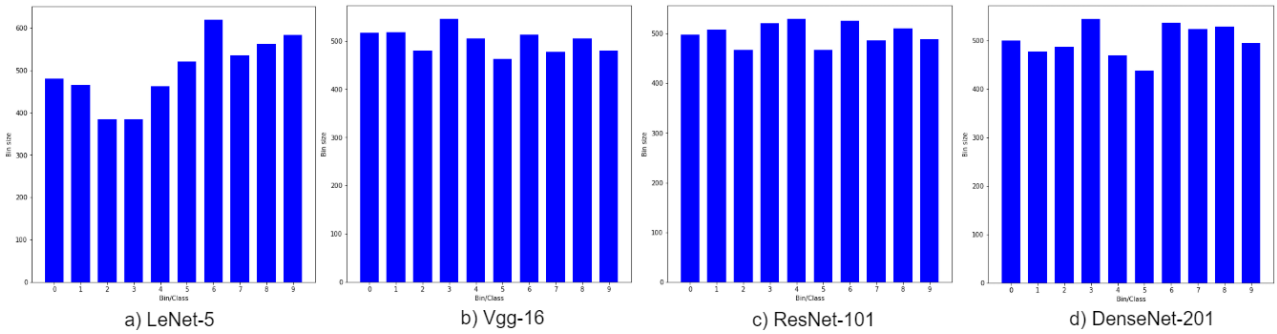

Figure 4.5.3 Data distribution between Bins in class wise binning approach

The conclusion of the analysis is as follows:

- Only some bins contribute to the optimum temperature value.
- Better binning approach is required for calculating the calibration measures.
- Data points are fairly distributed in class-based binning.

- Class-wise temperature scaling is required to achieve better confidence calibration.

4.5.4 Class-wise Temperature Scaling System

The proposed system includes two main sub-modules as temperature calculator module and the confidence scaler module. Both the modules take the logit vector of the neural network as one of the inputs. The temperature calculator module divides the logits into bins by considering the predicted class. Then calculate the optimum temperature for each class by minimizing the negative log-likelihood between actual and predicted probability distribution. This module outputs a temperature vector that includes the optimum temperature value for each class. The confidence scaler module takes the logit vector and the temperature vector as inputs. Then scale the confidence values by the class temperature, apply the SoftMax function on the scaled confidence values and output the calibrated probability vector.

4.6 Users

The proposed solution is focused on further reducing expected and maximum calibration error in deep learning models. So, the users of safety-critical applications like medical image classification and self-driving cars will be most beneficial from this approach. Since this is a post-processing approach, industry-level users who are looking for a cost and time-effective way to recalibrate their model's confidence value will be also benefited from this approach.

4.7 Features

The proposed solution will have the following features.

- Softer distribution among classes.
- Confidence which reflects real world probabilities.
- Low cost.
- Accuracy preserving nature.
- Increased expressive power (model interpretability).

4.8 Summary

This chapter brief each module in novel class-wise temperature scaling approach including input, output, features and users. In the next chapter, design process of the proposed solution is discussed in detail.

Chapter 5 Design of Class-wise Temperature Scaling

5. 1 Introduction

Chapter 4 presented the process of coming up with the proposed novel approach and the main modules of the proposed system to reduce the calibration error in deep neural networks. The proposed approach consists of two main modules: class-wise temperature calculation module and confidence scaling module. The main modules are further discussed in this chapter.

Figure 5.1 shows the high-level design of the Class-wise Temperature Scaling approach. This approach is focused on pre-trained DNN based classifiers. The logit vector of the pre-trained neural network and the class labels are used as the inputs for the proposed design so as to output the calibrated confidence values.

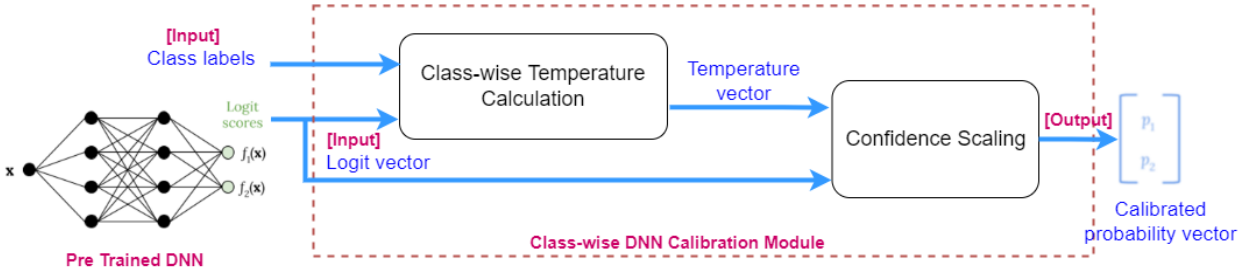

Figure 5.1 High-level design of class-wise Temperature Scaling approach

5.2 Class-wise Temperature Calculation Module

This module consists of two main steps: class-wise logit vector binning and temperature calculation for each bin. First, the batch of logit vectors of predictions for the validation dataset is obtained. Logit vector for one prediction contains prediction probabilities for each class in the classifier and the class which has the highest probability value is the predicted class for the specific data point. The obtained batch of logit vectors is divided into bins according to their predicted class. Then the optimum temperatures for each bin/class are calculated by minimizing the difference between actual data distribution and the predicted probability distribution. Here Negative Log-Likelihood is used to measure the distribution difference. Finally, this module outputs a vector of temperatures, which includes optimum temperature values for each class. The length of this temperature vector is equal to the number of classes in the selected classifier.

5.3 Confidence Scaling Module

This module consists of two main steps: scaling logit values by selecting relevant temperature and applying the SoftMax function on the scaled logit vector. The confidence scaling module uses both the logit vector and the temperature vector as inputs. According to the predicted classes, relevant temperature values are selected from the temperature vector for all the data points in the validation dataset. Then the logit vectors are scaled by the selected temperature and the SoftMax function is performed on these scaled logits so as to output the scaled probabilities. SoftMax function normalizes the output of the logit vector into probability distribution over the predicted output classes.

5.4 Summary

This chapter gives an insight into the overall design components of the proposed class-wise temperature scaling approach. In the next chapter, detailed discussion on implementation of the proposed solution is presented.

Chapter 6 Implementation

6.1 Introduction

The designs described in the previous chapter are realized in this chapter. Each module explained in the Design chapter is expanded in the aspect of implementation using algorithms, code segments, and hardware requirements. In addition, the model training process and the used datasets were described in this chapter.

6.2 Data Sets

Benchmark computer vision datasets, CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009 [38]) were used for training the models in this project.

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes and it is a subset of the Tiny Image dataset. There are 50,000 training images and 10,000 test images in the dataset. The training dataset consists of 5000 images from each class.

The CIFAR-100 dataset is also a subset of the Tiny Image dataset and it includes 60,000 32x32 color images in 100 classes. There are 600 images per class in the dataset and those classes are grouped into 20 super classes. The training dataset size is 60,000 and the test dataset includes 10,000 images.

Both the data sets are freely available at https://www.cs.toronto.edu/~kriz/cifar.html

6.3 Model Training

The complete implementation of the project was done using the python programming language. Open-source machine learning software library, TensorFlow, and the high-level deep learning API Keras, which runs on top of TensorFlow are used to implement the neural network models. In addition, the scikit-learn python machine learning library was used for handling and preprocessing the datasets. Following models were trained using transfer learning on Keras Applications.

Architecture Model	Trained models
LeNet	LeNet-5
VGGNet	Vgg-16, Vgg-19
ResNet	ResNet-50, ResNet-101, ResNet-152
DenseNet-121	DenseNet-121, DenseNet-169, DenseNet-201

Table 6.3 Implemented models and the architectures

Keras Applications are DNN models which are available with pre-trained weights. Most of these models were initially trained on the ImageNet dataset, which is consist of 224 x 224 color images and 1000 classes. So, when fine-tuning these models with CIFAR-10 and CIFAR-100 datasets, images were upscaled. In addition, the fully connected layer at the end of the neural network was changed according to the number of classes in the datasets. All the models were trained over 20 iterations and the weights of the models which had the best validation accuracy were saved using the "h5" format for analyzing, testing, experimenting, and evaluating purposes. Following algorithms and parameters were used in the model training process.

- Optimizer: Adam
- Loss function: Categorical Cross entropy
- Learning rate: 2e-5
- Epsilon:1e-08
- Batch size: 128, 256
- Epochs: 20 - 50

The model training was performed on a PC with the following specifications.

- RAM: 32 GB
- GPU: 10 GB, NVIDIA GeForce RTX 3060
- Processor: AMD Ryzen 7 3700X 8-Core Processor 3.60 GHz

6.4 Class-wise Temperature Scaling Algorithm

Algorithm 1: Class-wise Temperature Scaling

Goal: Find optimum temperature(T_c) for each class that minimize the calibration error

Require: Pre-trained classifier model, Validation set (X, Y)

Inputs: Batch of Logit vector for X_i(h_i= [h_i^1, h_i^2, … … … … , h_i^k]), True class Y_i

Initialization: T_c = T where c = 1, ……, K; (K: number of classes) and T > 0

Training:

1. Divide predictions into bins on classes (Bins= [B_i, … … … , B_k])
2. **for** B_i in Bins **do**
   while convergence **do**

$$\boldsymbol{T_c^*} = \arg min_T(-\sum_{i=1}^{n} \log\left(S_{y=yi}(X_i, T_c)\right)),$$

$$where, \qquad S_{y=yi}(X_i, T_c) = \frac{\exp(h_i^{yi}/T_c)}{\sum_j \exp\left(h_i^j/T_c\right)}$$

   end while

   end for

3. Calibrate SoftMax output

$$Sy(X, \boldsymbol{T_c^*}) = \frac{\exp(h_i^{yi}/\boldsymbol{T_c^*})}{\sum_j \exp\left(h_i^j/\boldsymbol{T_c^*}\right)}$$

6.5 Class-wise Temperature Calculator Implementation

Validation dataset is used for the temperature calculation process. This should be separate data set from training and testing data set, which was used to train and test the classifier model. NumPy mathematical library was used to bin the prediction according to the predicted class. Logit layer values for each predicted class and the true classes were taken as the inputs to the temperature calculation method.

First logit values were divided by temperature value and then algorithm analyses the deviation between scaled logits and the true class distribution. With more iterations, temperature value will be updated in a way that minimize the deviation between true and predicted distributions using SoftMax Cross Entropy on logits. Following algorithms and parameters were used for the optimum temperature calculation process.

- Optimizer: Adam
- Learning rate = 0.01
- Initial temperature = 1.0
- Iterations = 10, 000

Appendix I presents the implemented method for calculating optimum temperature for each class.

6.6 Confidence Scaler Module Implementation

Calculated optimum temperatures and the logit values are the inputs for the confidence scaling method. First the logit values were divided by relevant class temperature and then SoftMax function is applied for the scaled logit so as to output the calibrated confidence values. Appendix II presents the implemented python method for the confidence scaling.

6.7 Summary

Implementation details of the proposed approach is explained in this chapter. The next chapter describes how the implemented system was evaluated against the standard temperature scaling technique and the calibration measures.

Chapter 7 Evaluation

7.1 Introduction

Main goal of this chapter is to compare the standard temperature scaling technique with the proposed class-wise temperature scaling approach with respect to Expected Calibration Error and Maximum Calibration Error.

7.2 Model Calibration Evaluation

Miscalibration analysis in DNN and the issues in data distribution of standard binning approach were described in chapter 4. To overcome the pitfalls in standard Temperature Scaling approach, class-wise binning and class-wise Temperature Scaling were proposed. Table 7.2 displays ECE and MCE comparison between uncalibrated, Temperature scaled and Class-wise temperature scaled models.

Model	Dataset	Uncalibrated		Temperature Scaling		Class-wise Temperature Scaling	
		ECE	MCE	ECE	MCE	ECE	MCE
LeNet-5	CIFAR-10	4.04	9.07	3.64	6.63	2.06	4.29
Vgg-16	CIFAR-10	6.16	16.30	2.09	6.42	0.99	3.16
Vgg-19	CIFAR-10	7.83	15.33	1.72	6.52	0.84	2.52
ResNet-50	CIFAR-10	6.67	15.50	1.26	3.73	1.00	2.15
ResNet-101	CIFAR-10	6.42	13.94	1.88	3.41	0.74	1.77
ResNet-152	CIFAR-10	5.13	12.69	1.70	4.04	0.65	1.64
DenseNet-121	CIFAR-10	10.94	21.89	2.71	8.74	1.18	4.16
DenseNet-169	CIFAR-10	16.32	24.21	1.97	3.83	1.23	2.33
DenseNet-201	CIFAR-10	15.65	26.01	2.03	5.90	1.17	2.31
LeNet-5	CIFAR-100	4.45	15.71	4.08	16.65	3.57	14.49
Vgg-16	CIFAR-100	5.53	26.58	4.87	22.75	3.46	12.02
Vgg-19	CIFAR-100	5.89	20.62	4.98	17.95	3.65	12.71
ResNet-50	CIFAR-100	12.71	34.88	5.27	17.56	3.11	9.38
ResNet-101	CIFAR-100	12.98	28.52	4.69	15.91	3.09	8.78
ResNet-152	CIFAR-100	11.89	37.99	4.90	21.55	3.05	15.25
DenseNet-121	CIFAR-100	8.46	23.62	5.64	17.40	3.58	11.66
DenseNet-169	CIFAR-100	10.37	22.37	5.21	13.60	3.19	12.78
DenseNet-201	CIFAR-100	14.75	32.53	5.74	18.74	3.91	11.40

Table 7.2.1 ECE, MCE comparison before and after applying calibration techniques

It is worth noting that most of the models have some degree of miscalibration. Although Temperature scaling could reduce both the Expected calibration Error (ECE) and Maximum calibration Error (MCE), there is still much room for improvements. MCE lies between 3.4% - 18.7% for all the models after calibrating with temperature Scaling. This is still a high value when it comes to safety critical applications like medical applications, automated transport, Bio medical engineering and high integrity software where high precision, reliability and explainability is required. Proposed Class-wise Temperature Scaling have further reduced both the ECE and MCE. This approach was able to reduce ECE to 0.65% – 3.6% and MCE to range of 1.6% - 14.49% where respective error range for TS are 1.7% - 5.7% and 3.4% – 22.7%. If we consider calibration error for all the models, class-Wise TS approach have been able to further reduced the calibration error by 1.6% – 10.7% than the standard Temperature Scaling approach.

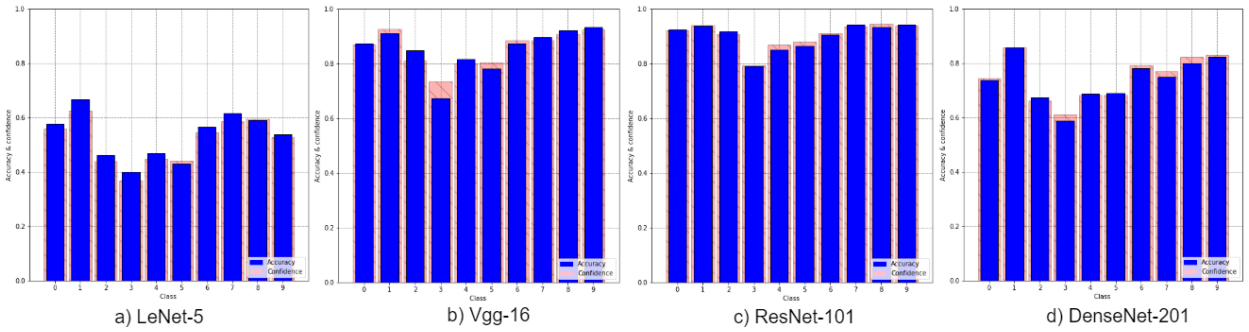

Figure 7.2.1 Reliability diagrams after calibrating with Class-wise TS

Figure 7.2.1 shows the reliability diagrams after calibrating the model predictions with class-wise temperature scaling. Here the accuracy of the model is visualized by blue color bars and the confidence values are visualized in red color bars. The gap between confidence and the accuracy is very small and for some classes the gap is nearly zero. That means calibration error for these classes are very low and the confidence value reflect the true accuracy.

Figure 7.2.2 presents the ECE and the MCE on CIFAR-10 and CIFAR-100 datasets for all the experimented models. All the diagrams include the uncalibrated, Temperature Scaled and Class-wise Temperature Scaled calibration measures. Diagrams clearly visualizes high miscalibration in all the uncalibrated models. In addition, diagrams prove the exemplary and outstanding

performance of Class-wise Temperature Scaling technique in minimizing both calibration measures, ECE and MCE.

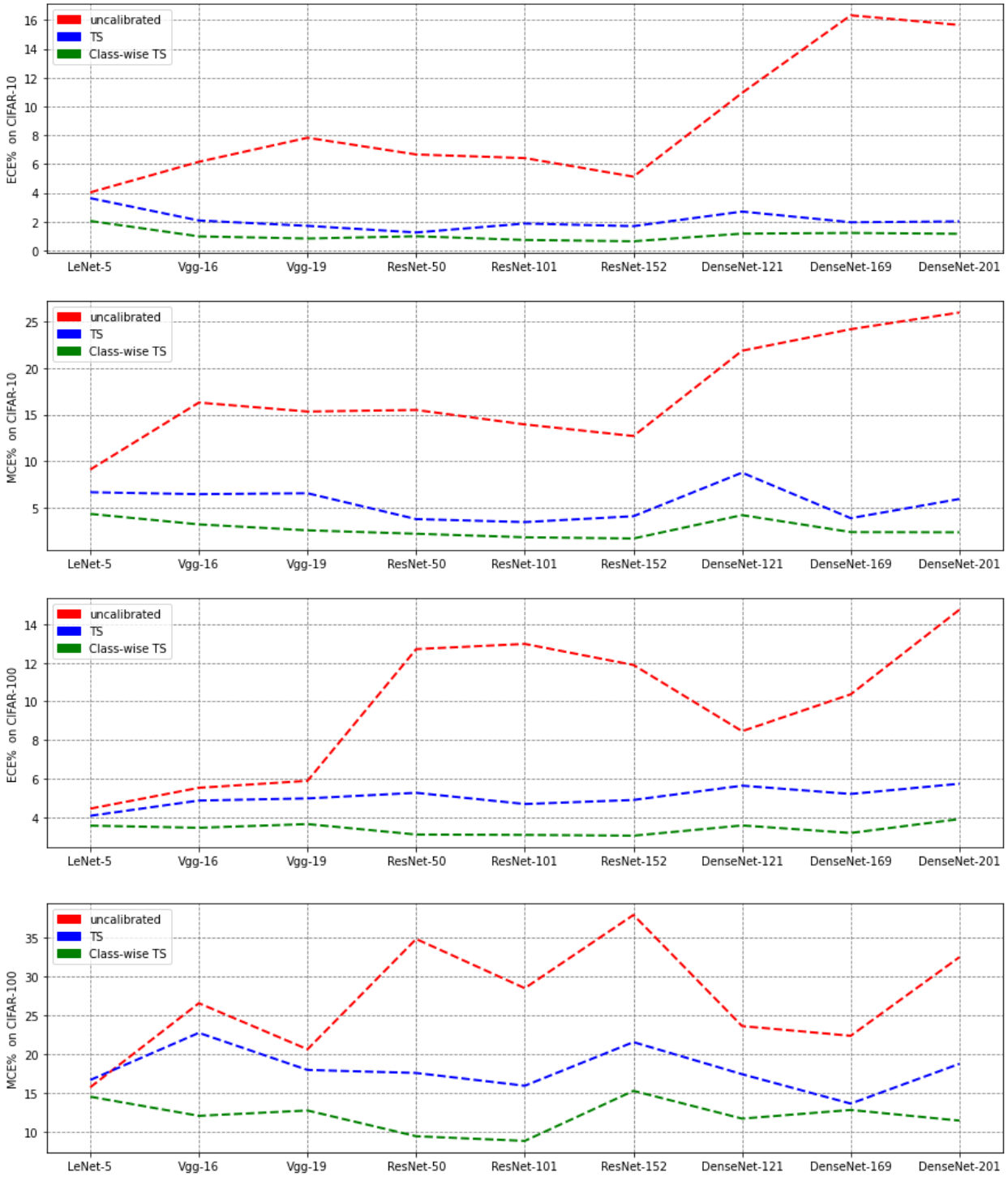

Figure 7.2.2 ECE, MCE evaluation of the model with CIFAR10 and CIFAR100 datasets

7.3 Summary

This chapter evaluate the State of art post processing model calibration technique, Temperature scaling with the proposed class-wise Temperature Scaling technique. Overall conclusion will be pointed out in the next chapter.

Chapter 8 Conclusion & Further Work

8.1 Introduction

In this research, a novel Class-wise post-hoc Temperature Scaling technique was introduced to calibrate the confidence values of trained classifiers. Move over new changes to standard binning approach was brought up to overcome the data out of distribution problem in probability range binning approach. In the previous chapter, a thorough evaluation was presented on the proposed Class-wise Temperature scaling approach and the state art calibration approach, Temperature Scaling technique. This chapter will conclude the dissertation by emphasizing the conclusions and further work to be conducted to further improve the calibration in modern Deep Neural Networks.

8.2 Conclusion

8.2.1 Achievement of Project Objectives

The overall project process was reckoned on achieving the objectives defined in the introduction chapter. All of the defined objectives were achieved as expected by following the proper project process.

With an in-depth literature review about neural network calibration, drawbacks in existing calibration approaches and the pitfalls in calibration measuring metrics were identified. In addition, attempts taken to overcome some drawbacks of the calibration techniques were realized. By observing the current research trends and state of art performance in minimizing the calibration error in deep neural network-based classifiers, one of the recently Introduced Temperature Scaling technique was considered to lay down a new solution for DNN miscalibration. Optimum temperature calculation using validation set was identified as the core of the solution. Class-wise Temperature Scaling was proposed by introducing class-wise binning and class-wise optimum temperature calculation for each predicted class by minimizing the deviation between true and predicted class probability distributions. Solution was designed with two main modules: class-wise temperature calculator and confidence scaler. Python programming language and the TensorFlow library were used to implement the solution. Proposed approach was evaluated with state of art calibration technique, Temperature Scaling by measuring the calibration error with ECE and MCE.

According to the evaluation, proposed approach was successful in further minimizing the calibration error in deep neural network-based classifiers.

8.2.2 Overall Conclusion

Modern Deep Neural Networks are miscalibrated although they displayed higher accuracy. Model architecture, depth of the model, normalization, number of layers are the key factors that affect the model miscalibration. According to the experiments and the evaluation, calibration error tends to increase with the depth and complexity of the model. In addition, the miscalibration measures like ECE and MCE are biased to the higher probability ranged bins. This bias can be avoided by using class-wise binning approach for calculating the calibration measures.

Calibration error of pre-trained classifiers can be minimized by applying temperature scaling on this model by using validation dataset. Accuracy of the models were not affected by the temperature scaling since it does not change the maximum of SoftMax function. Calibration error could be further minimized by using proposed class-wise Temperature Scaling technique. Same as Temperature scaling, this technique does not affect the accuracy of the model.

According to evaluation, Class-wise Temperature Scaling approach was able to reduce ECE to 0.65% – 3.6% and MCE to range of 1.6% - 14.49% where respective error range for TS are 1.7% - 5.7% and 3.4% – 22.7%. For all evaluated models, class-Wise Temperature Scaling approach have been able to further reduced the calibration error by 1.6% – 10.7% than the standard Temperature Scaling approach. So, Class-wise Temperature Scaling is able to further minimize the calibration error up to 10% than the Standard Temperature Scaling technique.

8.3 Limitations and further work

Although Class-wise Temperature Scaling can minimize the ECE to very small amount, still MCE is between 1.6% - 15.25% for some models and this should be further minimized. This can be done using prioritizing and ensuring the data quality in validation data set and also ensuring the label quality of data, which is used to calculate the optimum temperature values for each class. By using Active Learning strategies, data points can be prioritized and remove the noisy data before calculating the optimum temperatures. This technique also can reduce the time for the temperature

calculation process. In addition, Active Learning strategies can be used to generate and increase more data points, when the validation data set is very small.

8.4 Summary

By bringing the end to the thesis, this chapter describes the achievements of the project objectives, overall conclusion, the limitations and the further works to be done to further improve the calibration in Deep Neural Networks.

References

[1] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On Calibration of Modern Neural Networks,” arXiv:1706.04599 [cs], Aug. 2017. Available: http://arxiv.org/abs/1706.04599

[2] L. Alzubaidi et al., “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” J Big Data, vol. 8, no. 1, p. 53, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[3] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” arXiv:1503.02531 [cs, stat], Mar. 2015. Available: http://arxiv.org/abs/1503.02531

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. 2016. MIT Press

[5] Platt, John et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers, 10(3): 61–74, 1999

[6] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. ICLR, 2017

[7] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in Proceedings of the 22nd international conference on Machine learning - ICML ’05, Bonn, Germany, 2005, pp. 625–632. doi: 10.1145/1102351.1102430.

[8] A. S. Mozafari, H. S. Gomes, W. Leão, S. Janny, and C. Gagné, “Attended Temperature Scaling: A Practical Approach for Calibrating Deep Neural Networks,” arXiv:1810.11586 [cs, stat], May 2019.Available: http://arxiv.org/abs/1810.11586

[9] B. Ji, H. Jung, J. Yoon, K. Kim, and Y. Shin, “Bin-wise Temperature Scaling (BTS): Improvement in Confidence Calibration Performance through Simple Scaling Techniques,” arXiv:1908.11528 [cs], Sep. 201. [Online]. Available: http://arxiv.org/abs/1908.11528

[10] A. S. Mozafari, H. S. Gomes, and C. Gagne, “A Novel Unsupervised Post-Processing Calibration Method for DNNS with Robustness to Domain Shift,” arXiv:1911.11195 [cs, stat], Nov. 2019.Available: http://arxiv.org/abs/1911.11195

[11] A. Mozafari, H. Gomes, S. Janny, and C. Gagné, A New Loss Function for Temperature Scaling to have Better Calibrated Deep Networks. 2018.

[12] C. Tomani, D. Cremers, and F. Buettner, "Parameterized Temperature Scaling for Boosting the Expressive Power in Post-Hoc Uncertainty Calibration," arXiv:2102.12182 [cs], Feb. 2021, Available: http://arxiv.org/abs/2102.12182

[13] J. Nixon et al., "Measuring Calibration in Deep Learning," arXiv:1904.01685 [cs, stat], Aug. 2020, Available: http://arxiv.org/abs/1904.01685

[14] Naeini, Mahdi Pakdaman, Cooper, Gregory F, and Hauskrecht, Milos. Obtaining well calibrated probabilities using bayesian binning. In AAAI, pp. 2901, 2015.

[15] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in Proceedings of the 22nd international conference on Machine learning - ICML '05, Bonn, Germany, 2005, pp. 625–632. doi: 10.1145/1102351.1102430.

[16] LeCun, Yann, Bottou, Leon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998

[17] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun,Jian. Deep residual learning for image recognition. In CVPR, pp. 770–778, 2016.

[18] Zadrozny, Bianca and Elkan, Charles. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In ICML, pp. 609–616, 2001.

[19] Zadrozny, Bianca and Elkan, Charles. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In ICML, pp. 609–616, 2001.

[20] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty," arXiv:1912.02781 [cs, stat], Feb. 2020. Available: http://arxiv.org/abs/1912.02781

[21] S. Liang, Y. Li, and R. Srikant, "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks," arXiv:1706.02690 [cs, stat], Aug. 2020. [Online]. Available: http://arxiv.org/abs/1706.02690

[22] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. In ICLR, Workshop Track Proceedings, 2017.

[23] Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. In ICLR, 2017.

[24] Naeini, Mahdi Pakdaman, Cooper, Gregory F, and Hauskrecht, Milos. Obtaining well calibrated probabilities using bayesian binning. In AAAI, pp. 2901, 2015.

[24] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 6023–6032, 2019.

[25] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 113–123, 2019.

[26] Y. Wen, D. Tran, and J. Ba. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In International Conference on Learning Representations, 2020.

[27] Lakshminarayanan, Balaji, Pritzel, Alexander, and Blundell, Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. arXiv preprint arXiv:1612.01474, 2016

[28] Hara, K., Saitoh, D. & Shouno, H. Analysis of dropout learning regarded as ensemble learning. In Proc. 25th Int. Conf. Artificial Neural Networks 72–79 (ICANN, 2016).

[29] F. Laakom, J. Raitoharju, A. Iosifidis, J. Nikkanen, and M. Gabbouj, “Monte carlo dropout ensembles for robust illumination estimation,” in 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021, pp. 1–7.

[30] Y. Wen, G. Jerfel, R. Muller, M. W. Dusenberry, J. Snoek, B. Lakshminarayanan, and D. Tran, “Combining ensembles and data augmentation can harm your calibration,” in International Conference on Learning Representations, 2021

[31] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? In Advances in Neural Information Processing Systems, 2019

[32] M. Minderer, J. Djolonga, R. Romijnders, F. Hubis, X. Zhai, N. Houlsby, D. Tran, and M. Lucic. Revisiting the calibration of modern neural networks. arXiv preprint arXiv:2106.07998, 2021.

[33] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998a). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, 2278–2324.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[36] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten. Densely connected convolutional networks. In CVPR, 2017.

[37] L. Frenkel and J. Goldberger. Network Calibration by Class-based Temperature Scaling. In European Signal Processing Conference (EUSIPCO) , 2021.

[38] Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Appendix

Appendix I: Standard Temperature Calculation

```
def cal_TS(Y_test, y_logit):

    temp = tf.Variable(initial_value=1.0, trainable=True, dtype=tf.float32)
    optimizer = tf.optimizers.Adam(learning_rate=0.01)

    def calculate_optimum_temperature():
        scaled_logits = tf.math.divide(y_logit, temp)
        loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
            labels=tf.convert_to_tensor(K.utils.to_categorical(Y_test)),
            logits=scaled_logits))
        return loss

    for j in range(10000):
        opts = optimizer.minimize(calculate_optimum_temperature, var_list=[temp])

    optimum_temp = temp.numpy()
    return optimum_temp
```

Appendix II: Binning on Confidence Range

```
def calc_bins(y_pred_class,y_pred_class_proba, y):

    num_bins = 10
    bins = np.linspace(0.1, 1, num_bins)
    binned = np.digitize(y_pred_class_proba, bins)

    # Save the accuracy, confidence and size of each bin
    bin_accs = np.zeros(num_bins)
    bin_confs = np.zeros(num_bins)
    bin_sizes = np.zeros(num_bins)

    for bin in range(num_bins):
        bin_sizes[bin] = len(y_pred_class_proba[binned == bin])
        if bin_sizes[bin] > 0:
            #calculate number of correct predictions in each bin/class
            bin_indexes = np.argwhere(binned == bin)
            y_pred_class_val = np.take(y_pred_class, bin_indexes)
            y_val  = np.take(y, bin_indexes)

            count = sum(ypc == yv for ypc, yv in zip(y_pred_class_val, y_val))[0]
            bin_accs[bin] = count / bin_sizes[bin]
            bin_confs[bin] = (y_pred_class_proba[binned==bin]).sum() / bin_sizes[bin]

    return bins, binned, bin_accs, bin_confs, bin_sizes
```

Appendix III: Standard ECE and MCE Calculation

```
def get_metrics(y_pred_class,y_pred_class_proba, y):
    ECE = 0
    MCE = 0
    bins, binned, bin_accs, bin_confs, bin_sizes = calc_bins(y_pred_class,y_pred_class_proba, y)

    for i in range(len(bins)):
        abs_conf_dif = abs(bin_accs[i] - bin_confs[i])
        ECE += (bin_sizes[i] / sum(bin_sizes)) * abs_conf_dif
        MCE = max(MCE, abs_conf_dif)

    return ECE, MCE ,bins,bin_accs, bin_confs
```

Appendix IV: Reliability Diagram

```
def draw_reliability_graph(y_pred_class,y_pred_class_proba, y):
    ECE, MCE ,bins,bin_accs, bin_confs = get_metrics(y_pred_class,y_pred_class_proba, y)
    print("ECE: ", str(ECE*100))
    print("MCE: ", str(MCE*100))

    fig = plt.figure(figsize=(8, 8))
    ax = fig.gca()

    # x/y limits
    ax.set_xlim(0, 1.05)
    ax.set_ylim(0, 1)

    # x/y labels
    plt.xlabel('Confidence ')
    plt.ylabel('Accuracy')

    ax.set_axisbelow(True)
    ax.grid(color='gray', linestyle='dashed')
    plt.bar(bins, bins,  width=0.1, alpha=0.3, edgecolor='black', color='r', hatch='\\')

    # Draw bars and identity line
    plt.bar(bins, bin_accs, width=0.07, alpha=1, edgecolor='black', color='b')
    plt.plot([0,1],[0,1], '--', color='gray', linewidth=2)

    plt.gca().set_aspect('equal', adjustable='box')
    acc = mpatches.Patch(color='blue', label="Accuracy") # acc
    conf = mpatches.Patch(color='pink', label="Confidence") #conf
    plt.legend(handles=[acc,conf], loc="lower right")
    plt.show()
```

Appendix V: Confidence Calibration using Temperature Scaling

```
def calibrate_with_TS(Y_test, y_logit, y_pred_class,y_pred_class_proba, y, num_bins):
    optimum_temp = cal_TS(Y_test, y_logit)
    scaled_logits_with_opt_T = tf.math.divide(y_logit, optimum_temp)
    scaled_soft_proba_T = softmax(scaled_logits_with_opt_T.numpy())
    calibrated_confidence = np.max(scaled_soft_proba_T, axis=-1)
    return calibrated_confidence
```

Appendix VI: Class-wise temperature calculation

```
def cal_T_for_classes(y_pred_class ,Y_test ,y_logit,num_classes):
    temperature_list = np.zeros(num_classes)
    for i in range(num_classes):
        temp = tf.Variable(initial_value=1.0, trainable=True, dtype=tf.float32)
        optimizer = tf.optimizers.Adam(learning_rate=0.01)
        index_list = np.argwhere(y_pred_class==i)
        Y_test_temp = np.take(Y_test, index_list)
        y_logit_temp = np.take(y_logit, index_list , axis=0)
        y_logit_temp =  y_logit_temp.reshape(y_logit_temp.shape[0],y_logit_temp.shape[2])

        def calculate_optimum_temperature():
            scaled_logits = tf.math.divide(y_logit_temp, temp)
            loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
                labels=tf.convert_to_tensor(K.utils.to_categorical(Y_test_temp, num_classes =num_classes)),
                logits=scaled_logits))
            return loss

        for j in range(10000):
            opts = optimizer.minimize(calculate_optimum_temperature, var_list=[temp])

        optimum_temp = temp.numpy()
        temperature_list[i] = optimum_temp
    return temperature_list
```

Appendix VII: Confidence Calibration using Temperature Scaling

```
def calibrate_softmax_output(temperature_vector, y_logit,y_pred_class):
    scaled_logits_with_opt_T = []
    for lgt,ypc in zip(y_logit,y_pred_class):
        lgt = lgt / temperature_vector[ypc]
        scaled_logits_with_opt_T.append(lgt)
    scaled_logits_with_opt_T = tf.convert_to_tensor(scaled_logits_with_opt_T, dtype=tf.float32)
    scaled_soft_proba_T = softmax(scaled_logits_with_opt_T.numpy())
    calibrated_confidence = np.max(scaled_soft_proba_T, axis=-1)
    return calibrated_confidence
```

Appendix VIII: Binning on Predicted Class

```
def cal_class_bins(y_pred_class,y_pred_class_proba, y, num_bins):
    num_bins = num_bins
    bins = [i for i in range(num_bins)]
    binned = np.digitize(y_pred_class, bins)
    # digitize start from 1. so recorect class values sustract 1
    binned  = binned -1

    # Save the accuracy, confidence and size of each bin
    bin_accs = np.zeros(num_bins)
    bin_confs = np.zeros(num_bins)
    bin_sizes = np.zeros(num_bins)

    for bin in range(num_bins):
        bin_sizes[bin] = len(y_pred_class[binned == bin])
        if bin_sizes[bin] > 0:

            #calculate number of correct predictions in each bin/class
            count = 0
            for ypc, yc in zip(y_pred_class,y):
                if ypc==bin and ypc==yc:
                    count +=1

            bin_accs[bin] = count / bin_sizes[bin]
            bin_confs[bin] = (y_pred_class_proba[binned==bin]).sum() / bin_sizes[bin]

    return bins, binned, bin_accs, bin_confs, bin_sizes
```