UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Markus Kängsepp

Calibration of Convolutional Neural Networks

Master's Thesis (30 ECTS)

Supervisor: Meelis Kull, PhD

Tartu 2018

Calibration of Convolutional Neural Networks

Abstract: Deep neural networks have become more popular over time and nowadays these are used for many practical applications. However, the precise output by itself might not be enough, as in some areas it is also important to know how confident the model is. As recently shown, deep neural network predictions are not well-calibrated, in contrast to shallow ones. For example, deep neural networks tend to be over-confident.

In 2017, Guo et al. published temperature scaling method (Guo et al., 2017) and compared it to other existing confidence calibration methods. Later that year, Kull et al. published beta calibration method (Kull et al., 2017), however, it was not tested on neural networks. The thesis evaluates beta calibration in context of convolutional neural networks and in order to compare the results with other calibration methods, some of the Guo et al. results were replicated.

This thesis compares histogram binning, isotonic regression and temperature scaling methods from Guo et al. and beta calibration by Kull et al. on various state-of-the-art convolutional neural networks. In addition to loss measures used by Guo et al., Brier score was added. The results were in accordance with Guo et al. outcome. The beta calibration was a little bit worse for most of the models compared to temperature scaling, however, in case of error rate, it was a bit better compared to temperature scaling.

Keywords: neural networks, calibration, convolution, image classification, residual networks, replication

CERCS: P170

Konvolutsiooniliste närvivõrkude kalibreerimine

Lühikokkuvõte: Süvanärvivõrgud koguvad aina populaarsust ja tänapäeval on need kasutusel ka mitmetes praktilistes rakendustes. Sellest hoolimata, ainult klassi märgendi ennustamine ei pruugi olla enam piisav, sest mõningatel aladel on ka tähtis teada kui kindel mudel enda väljundis on. Hiljuti näidati, et sügavate närvivõrkude ennustused pole nii hästi kalibreeritud, võrreldes madalamate võrkudega. Näiteks sügavad närvivõrgud kipuvad olema liigselt enesekindlad.

Aastal 2017, Guo et al. avaldas temperatuuri skaleerimise (Guo jt, 2017) (*temperature scaling*) meetodi ning võrdles seda teiste olemas olevate kalibreerimismeetoditega. Samal aastal avalikustas Kull et al. beta kalibreerismeetodi (Kull jt, 2017) (*beta calibration*), kuid seda ei testitud närvivõrkudel. Antud töö käigus hinnati beta kalibreerimise headust konvolutsioonilistel närvivõrkudel ja selleks, et võrrelda tulemusi teiste kalibreerimismeetoditega on osa Guo et al. tulemustest replitseeritud.

See lõputöö võrdleb histogrammimeetodit (*histogram binning*), isotoonilist regressiooni (*isotonic regression*) ja temperatuuri skaleerimine Guo et al. artiklist ja beta kalibreerimist Kull et al. artiklist erinevatel uusimatel konvolutsioonilistel närvivõrkudel. Lisaks Guo et al. poolt kasutatavatele kaomõõtudele (*loss measure*), Brieri skoor lisati võrdlusesse. Töös saadud tulemused olid kooskõlas Guo et al. tulemustega. Beta kalibreerimine oli enamustel mudelitel veidi halvem kui temperatuuri skaleerimine. Vaatamata sellele, veamäära korral oli beta kalibreerimine vähekene parem kui teised võrdluses olevad kalibreerimise meetodid.

Võtmesõnad: tehisnärvivõrgud, kalibreerimine, konvolutsioon, pildi klassifitseerimine, residuaalne närvivõrk, replikatsioon

CERCS: P170

Contents

1 Introduction

Neural networks have lately gained huge popularity in machine learning community. Deep learning models have broken into many industries, including self-driving cars (Bojarski et al., 2016), healthcare (Caruana et al., 2015) and natural language processing (Sutskever et al., 2014). In these industries, it is not only necessary to achieve high accuracy, but also the confidence of prediction is a matter of interest. For example, the medical diagnosis tool needs to know when to rely on machine learning model or when to contact medical crew for confirmation (Jiang et al., 2011). However, a high confidence alone is not beneficial, if it is not well-calibrated. The calibrated model is a model, which confidence matches to the actual accuracy of the prediction. For example, there are 100 predictions where confidence of a model is 90 percent, then about 90 of the predictions should be correct.

Although shallow neural networks give well-calibrated results (LeCun et al., 1998), modern deep networks tend to output overconfident predictions as shown by Guo et al. (Guo et al., 2017). They have shown that deeper and wider models, networks with batch normalization and smaller weight decay may result in less calibrated predictions.

The aim of this thesis is to evaluate beta calibration (Kull et al., 2017) in context of convolutional neural networks. In order to give a valid comparison with other calibration methods, some of the results of the paper "On Calibration of Modern Networks" by Guo et al. (Guo et al., 2017) are replicated. Specifically, several variations of convolutional neural network models are trained on 4 different image datasets: CIFAR, ImageNet, SVHN and Birds. From calibration methods the following are used: histogram binning, isotonic regression, temperature scaling and beta calibration. However, the temperature scaling, one parameter version of Platt Scaling, seems to still be the best method for image classification and for convolutional neural networks.

In the following Chapter 2, a background information about calibration, datasets and neural networks is described. In Chapter 3, a more detailed description of the previous work is given. It describes findings of Guo et al. and some other work done in this area. Next, Chapter 4 is about datasets preparation and training procedure, also giving links to trained model weights and to uncalibrated predictions. Chapter 5 has tables and figures with the results and a description of the outcomes. Lastly, there is Chapter 6 about overall thoughts and problems faced in the process of replication. In addition, Appendix has extra figures with reliability diagrams for all trained models.

2 Background

This chapter gives the necessary background information about the calibration and neural network models that were used in the thesis.

2.1 Calibration

First, in order to calibrate confidence of a model, the model has to output estimated probabilities of predictions. In other words, the output should contain predicted confidences $\hat{P}$ for all the classes $\hat{Y}$.

The confidence of a prediction indicates how certain the model is about given prediction, thus it can also be called an uncertainty estimation. However, it usually does not come out of the model well-calibrated, hence, the need for confidence calibration.

The idea of confidence calibration is to modify the estimated confidences in such way that these are more reliable. In other words, for perfectly calibrated estimations, the confidence matches the actual accuracy of the prediction. For example, let there be 100 predictions with the confidence of 0.8, thus 80 of these should be correct. However, it may also be important to achieve reliable probabilities for all the classes, not only for predicted class. For example, in the case of log-loss (subsection 2.1.2), the confidence of actual true class matters.

Next, there are some ways to visualize and measure calibration error.

2.1.1 Reliability Diagrams

Reliability diagram (Figure 1) is a great way to visualize the calibration error (Niculescu-Mizil and Caruana, 2005). It compares the average confidence of predictions to the average accuracy of predictions. These are split into M bins, with interval size of $\frac{1}{M}$. That way it is possible to see the gap (red bar in Figure 1) between the confidence and the accuracy of a certain interval. However, the reliability diagram does not indicate the number of samples that fall into one bin.

The bin B_m is defined as a set of the indices of samples, which prediction confidences fall into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$. The following equations demonstrate how the confidence and accuracy of B_m is calculated:

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} (\hat{p}_i),$$

where $\hat{p}_i$ is the confidence of prediction at index i.

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i),$$

where $\hat{y}_i$ is predicted label and y_i is the actual label of the class at index i.

In order to achieve perfect calibration, the $acc(B_m)$ and $conf(B_m)$ must be equal for every bin. Meaning there would be no gap (red bar) visible on the actual diagram. However, that would give perfect calibration in expectation. This means that if the confidence of prediction falls into the bin, it is expected to have mean confidence of the bin. On the other hand, to achieve perfect calibration in the test phase, the confidence should exactly match with the accuracy. So, in practical cases, it is almost impossible to achieve perfect calibration.

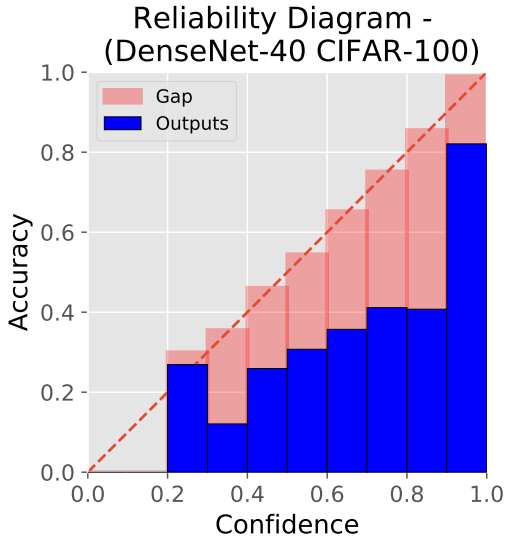


Figure 1. Example of a reliability diagram with uncalibrated confidences.

In Figure 1, the reliability diagram has no bars in the first two bins, because there were no predictions in the confidence interval of $[0, 0.1]$ and $(0.1, 0.2]$. Furthermore, the bar of the last bin does not follow the diagonal as other bars do, because it depends on the average confidence and for the last bin it is close to 1.0.

2.1.2 Scoring Measures

The visual methods (i.e reliability diagrams) are not the best way to accurately compare a large number of calibration methods, for that the scalar measures are superior. This is the case because it is hard to compare a large number of diagrams and the diagrams do not summarise the overall quality of calibration very well. For instance, the number of samples in each bin is not visualised.

Expected Calibration Error (ECE) is a calibration measure that gives an overall estimate of error between confidence and accuracy over all the bins (Naeini et al., 2015). First, the predictions are divided into M bins based on the confidence, similar to reliability diagrams. Then the error is approximated by finding a weighted average of the difference between accuracy and confidence for all the bins,

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|,$$

where n is the number of samples.

However, the score is dependent on the selection of the parameter M. For example, if M is small, the size of the confidence intervals are big, which means that the confidences in the bins vary a lot. On the other hand, if M is big, some of the bins have likely a small number of samples, so the average accuracy of the samples might not reflect the reality. The ECE measure is used as a primary error indicator in the replicated paper (Guo et al., 2017).

Maximum Calibration Error (MCE) shows the biggest error the calibrated confidences have (Naeini et al., 2015). It is similar to ECE as it also finds errors inside the bin, however the maximum error is returned instead of weighted average,

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

This measure shows the biggest gap between confidence and accuracy. For example, in medicine, it might be crucial to know the confidence interval of a prediction, because the 10 percent error in the confidence in the worst case, may turn odds against certain treatment or cure. In addition, in the case of MCE, the bins with few instances might

affect the score heavily, because the average accuracy might differ a lot from the average confidence.

Negative Log Likelihood (NLL) is a commonly used loss measure for neural networks and in deep learning, it is also called log loss or cross-entropy loss (Bengio et al., 2015). It is a standard way to measure the quality of a probabilistic model (Friedman et al., 2001). Let $\hat{\pi}(Y|X)$ be a probabilistic model with n samples that outputs probability of class Y given instance X, then NLL is defined as follows:

$$NLL = -\sum_{i=1}^{n} log(\hat{\pi}(y_i|x_i)),$$

where y_i is a true label and x_i is an instance.

NLL is also used as a loss measure for optimizing the parameters of temperature scaling and Beta calibration.

Brier Score is another way to measure the accuracy or calibration of model outcomes (Brier and Allen, 1951). The Brier score, in essence, is mean square error, however, it takes binary or categorical values and the set of probabilities assigned to these values as an input. The output is in the range of 0 to 1 and the lower the score, the better the model performs. Let there be n samples with y as vector of true labels and $\hat{p}$ as vector of predicted probabilities. The length of the vectors is r and it depends on the number of classes. Brier score is defined as:

$$Br = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{r}(\hat{p}_{ij} - y_{ij})^2$$

2.1.3 Logits and Neural Networks

Neural networks usually output probabilities for a prediction, however, logits $z_i \in \mathbb{R}$ can be used as an input of calibration methods. The logits are outputs of a neural network model, before using softmax function. The softmax function or sigmoid function $\hat{p_i} = \sigma_{SM}(z_i)$ is used to fit all the probabilities $\hat{p_i}$ in the range of 0 to 1, so that the sum of values is 1. The softmax function is defined as:

$$\sigma_{sm}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)},$$

where K is the number of classes.

Logits are used as an input for some of the calibration methods. More about these methods in following subsection 2.1.4.

2.1.4 Calibration Methods

This chapter gives an overview of the process of calibration. First, different calibration methods are described for binary class models and afterwards for multiclass calibration.

The calibration is done as a post-processing step for all the models, however, it can be also added as an extra step to the end of a neural network model training procedure. For calibration, an extra hold-out validation set is needed. It is used for fitting the parameters of the calibration model and may be used for model hyper-parameter tuning too. Finally, the test set is used to check the final calibration error and error rate of the model. The error rate is checked, because some calibration methods are changing the outcomes of the model, in addition to the confidences of the outcome. It is assumed that the training set, validation set and test set share the same distribution.

Next, the calibration methods used in this thesis are described. In total, there are 3 methods used from Guo et al. (Guo et al., 2017) paper, including temperature scaling, histogram binning and isotonic regression. Furthermore, one extra method called Beta calibration introduced by Kull et al. (Kull et al., 2017) is added to the comparison.

The following methods are described at first for binary outcomes and an extension to multiclass models is introduced in the Subsection 2.1.5 (Multiclass Calibration), similarly to Guo et al. (Guo et al., 2017). In the case of the binary models, the input is x_i and the outcome is $y_i = \{0, 1\}$. However, for simplicity, it is assumed that the model outputs only a positive class $y_i = 1$ and its probability $\hat{p}_i$. Additionally, logit value z_i of true class is available, for example, it is needed for temperature scaling. The goal is to get calibrated confidence $\hat{q}_i$ using y_i, $\hat{p}_i$ and z_i.

Histogram Binning is a simple non-parametric calibration method (Zadrozny and Elkan, 2001). The idea is to divide the samples into M equally sized mutually exclusive bins $\{B_1, B_2, ..., B_M\}$. The boundaries of the bins are defined by the number of the bins, so that each bin has an equal interval length. Another option is to have an equal number

of samples in each bin and based on that calculate the intervals. The boundaries of bins are defined as $0 = a_1 \leq a_2 \leq ... \leq a_M \leq a_{M+1} = 1$, so the bin B_i covers the interval $(a_i, a_{i+1}]$. For each bin, a new confidence C_m is found based on the predicted probabilities that fall into that bin. After the calibration process is done, the new probability of a prediction $\hat{y}_i$ is the calibrated confidence C_m, if its probability $\hat{p}_i$ falls into bin B_m.

The calibrated confidence of a bin is calculated by selecting C_m values so that it minimizes following function:

$$\min_{C_1,...,C_M} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m < \hat{p}_i \leq a_{m+1})(C_m - y_i)^2,$$

where the $\mathbf{1}$ is an indicator function, showing whether the probability belongs to given bin or not. In essence, the calibrated confidence of a bin is the average accuracy of the bin, $C_m = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(y_i = 1)$.

Isotonic Regression is another non-parametric method that is widely used for calibration (Zadrozny and Elkan, 2002a). It learns a piecewise constant function f that outputs calibrated probabilities $\hat{q}_i = f(\hat{p}_i)$. The function f tries to minimize the squared error,

$$\min_{\substack{M \\ C_1,...,C_M \\ a_1,...,a_{M+1}}} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m < \hat{p}_i \leq a_{m+1})(C_m - y_i)^2$$

$$\text{subject to } \begin{aligned} &0 = a_1 \leq a_2 \leq ... \leq a_m \leq a_{m+1} = 1, \\ &C_1 \leq C_2 \leq ... \leq C_M, \end{aligned}$$

where M is number of bins, $a_1, ..., a_{M+1}$ are bin boundaries and $C_1, ..., C_M$ are output values of function f. Despite the number of parameters, it turns out that it is easy to optimise with Pool Adjacent Violators (PAV) algorithm (Ayer et al., 1955).

Platt Scaling is a parametric calibration method that uses non-probabilistic predictions as an input for calibration (Platt et al., 1999). The input is used to train a logistic regression model with the parameters $a, b \in \mathbb{R}$, which returns calibrated probabilities. Platt scaling was not included in the final comparison of the calibration methods, however, it is brought out here because a couple of other methods are extensions of Platt scaling.

Temperature Scaling is used in the context of distilling (G. Hinton et al., 2015) and statistical mechanics (Jaynes, 1957), however Guo et al. (Guo et al., 2017) took it to neural network calibration. It is a modification of Platt Scaling using only single scalar parameter T. The parameter T is used for all the classes, so this method is suitable for multiclass calibration. Temperature scaling uses logits vector z_i as an input to find calibrated confidence, the new confidence prediction is calculated as

$$\hat{q}_i = \max_k \sigma_{SM}(z_i/T)^{(k)},$$

where k indicates the class.

The optimal temperature T can be found by optimizing it using NLL as a loss measure. In addition, the temperature scaling changes probabilities of predictions, but it does not affect the maximum probability. This means that the predicted class does not change and thus, the accuracy of the model does not change.

Beta Calibration (Kull et al., 2017) is a parametric calibration method that is based on beta distribution. It is similar to logistic regression (Platt Scaling), except, the beta calibration is derived based on beta distribution, but logistic regression is derived based on Gaussian distribution. Beta calibration has 3 parameters a, b and c, and is able to fit more distributions compared to logistic regression as more shapes are possible in beta calibration map family,

$$\mu_{beta}(\hat{p}; a, b, c) = \frac{1}{1 + 1/(e^c \frac{\hat{p}^a}{(1-\hat{p})^b})}$$

where $a, b \geq 0$ and $c \in \mathbb{R}$.

The predicted probabilities are converted into logarithmic space to use logistic regression for fitting. The parameters are fit using bivariate logistic regression on logarithmically transformed probabilities using NLL loss,

$$LR_{bilogistic}(\ln \hat{p}, -\ln(1 - \hat{p}); a, b, c)$$

The fitted parameters are used with $\mu_{beta}(\hat{p}_{test}; a, b, c)$ to calculate calibrated probabilities $\hat{p}_{test}$.

2.1.5 Multiclass Calibration

All the datasets used in this thesis have multiple classes to predict. Meaning some of the calibration methods have to be changed a bit to be able to calibrate predictions for multiple classes ($K > 2$). Additionally, in the case of neural networks, the logits z_i and probabilities $\hat{p}_i$ are vectors with size of k and class labels are $\hat{y}_i = \{0, 1, ..., k\}$. The true label $\hat{y}_i$ of the class is $\text{argmax}_k z_i^{(k)}$ and the probabilities $\hat{p}_i$ are derived using softmax function $\hat{p}_i = \max_k \sigma_{sm}(z_i)^{(k)}$.

A common way of doing a multiclass calibration would be using K 1-vs-all models (Zadrozny and Elkan, 2002b). So there is $k = \{1, 2, ..., K\}$ binary calibration problems, where the label is $\mathbf{1}(y_i = k)$ and predicted probability is $\hat{p}_i^{(k)} = \sigma_{sm}(z_i)^{(k)}$. This means that a single class is fixed as a positive class and all others as negative, and repeating the process for all the classes. As a result, a vector with unnormalized calibrated probabilities is formed $Q_i = [\hat{q}_i^{(1)}, ..., \hat{q}_i^{(K)}]$, where $\hat{q}_i^{(k)}$ is a calibrated confidence of the class k. As a new confidence $\hat{q}_i{}'$ a maximum value from the vector is selected and normalized by the sum of the vector $\sum_{k=1}^{K} \hat{q}_i^{(k)}$. The new class label $\hat{y}_i{}'$ is argmax of Q_i.

2.2 Datasets

In this thesis, 4 different datasets were used. This subsection describes briefly all the datasets.

2.2.1 CIFAR-10

The CIFAR-10 dataset (Krizhevsky, 2012) has colour images with dimensions of 32x32 from 10 different classes: aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. In total, there are 60 000 images, originally split into 50 000 training images and 10 000 test images. There are 10 balanced classes, so there are 6000 images in each class. In given replication task the training data is randomly split into two, in order to get a validation set. In the end, there are 45 000 training, 5000 validation and 10 000 test images.

Figure 2. Example images of CIFAR-10 dataset.

2.2.2 CIFAR-100

The CIFAR-100 (Krizhevsky, 2012) dataset is a lot like the CIFAR-10, except that instead of 10 classes, there are 100 classes. So for each class, there are 600 images, 500 for training and 100 for testing. These are split into 45 000 training, 5000 validation and 10 000 test images, similarly to CIFAR-10.

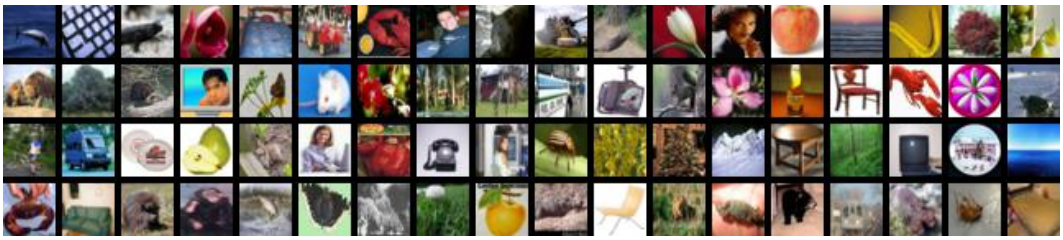


Figure 3. Example images of CIFAR-100 dataset.

2.2.3 ImageNet

The Imagenet (Deng et al., 2009) dataset consists of colour images with various sizes from 1000 different classes. In total there are 1.28 million training images and 50 000 testing images. The test set is randomly split into two, so actually, there are 25 000 images for both validation and testing.

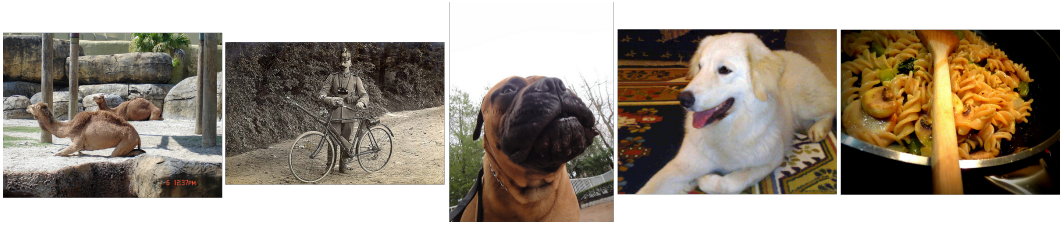


Figure 4. Example images of ImageNet.

2.2.4 Street View House Numbers (SVHN)

The SVHN dataset (Netzer et al., 2011) contains 32x32 colour images from 10 classes. The images are centred around the single digit, however, there are other digits from the house number visible too. The dataset has 73 257 training images, 26 032 test images and 531 131 additional training images. The additional training set is combined with the original training set. The validation set is formed by taking 400 images from each class from the training set and 200 images from the additional set. In the end, there are 598 388 training, 6 000 validation and 26 032 test images.

Figure 5. Example images of SVHN dataset.

2.2.5 Birds

The Caltech-UCSD Birds (Welinder et al., 2010) data set consists of 5994 training images and 5794 test images. The test set is split into two: leaving 2897 images in validation set and 2897 in test set.

Figure 6. Example images of Birds dataset.

2.3 Neural Networks

This subsection is based on "Deep Learning" book written by Goodfellow et al. (Goodfellow et al., 2016). Deep neural networks are a part of machine learning methods that attempt to learn data representation. The smallest unit of a neural network is a node,

also called neuron, because of its similar function to an actual neuron. Each node has a certain amount of inputs and outputs based on the architecture of a neural network. To compute the output of a node, inputs of it are multiplied by weights and summed together. The weights are optimized in a training process. Nodes of a neural network are grouped into layers, so each layer has input from the previous layer and gives output to the following layer. First layers learn simple concepts, which are used in following layers as a base for more complex concepts. Figure 7 shows a simplified version of a neural network for image classification.

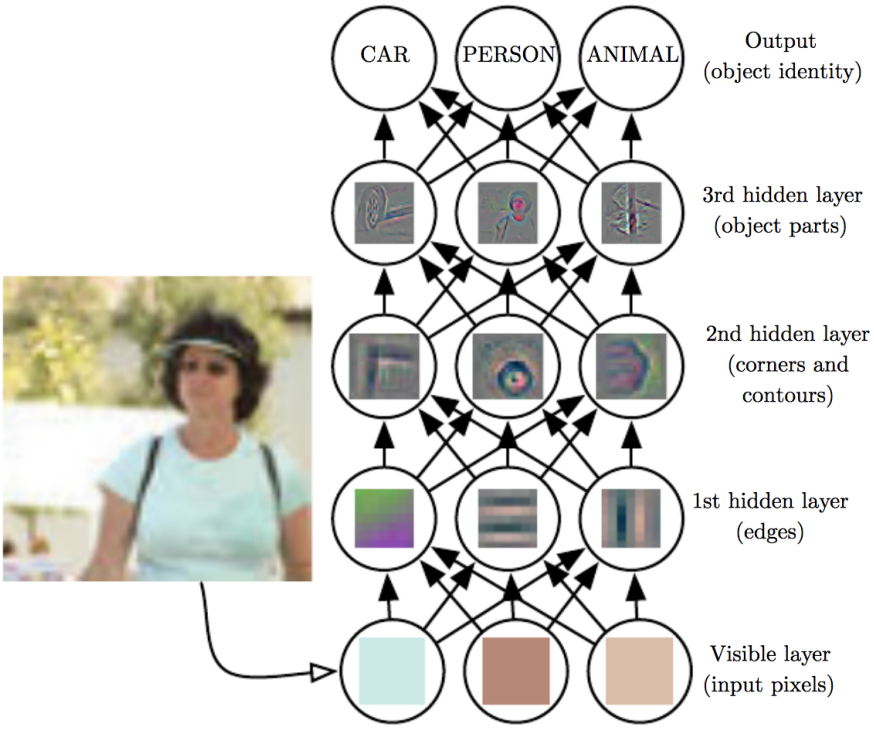


Figure 7. An example of a neural network. The images on the nodes visualize possible features captured by hidden units. (Goodfellow et al., 2016)

Neural networks start with a visible layer consisting of input, followed by hidden layers that extract abstract features of input, in this case, an image. The second layer is called "hidden" because the values for those layers are not given in the data.

In this thesis, convolutional neural networks (ConvNets) (LeCun et al., 1989) were used. These networks use grid-like topology called filters (kernels) for processing data.

In case of image processing, these filters can be thought of as a 2D grid of pixels that slide over the input image spatially. The filters compute dot-products, which are added to feature maps. The feature maps are used as an input for next layer (Figure 8). After training the network, the filters try to match certain patterns in images, for example, edges, circles, corners (Figure 7). These filters are combined into more complex representations, which can be used to classify input data.

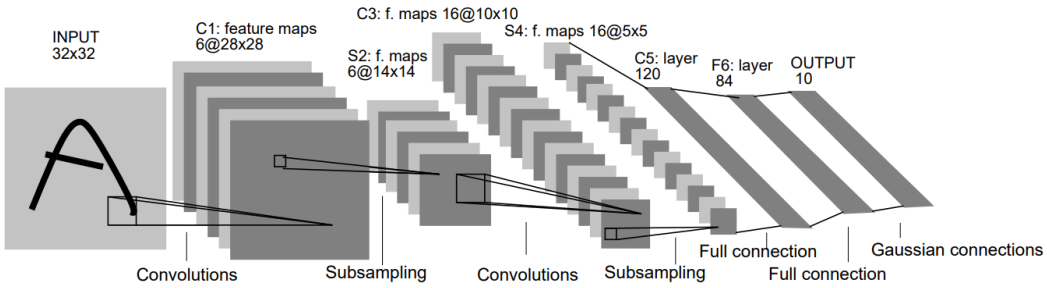


Figure 8. The architecture of LeNet-5 convolutional neural network for digit recognition (LeCun et al., 1998).

Most convolutional neural networks use also a pooling layer, which makes the representation more invariant to smaller translations of the input. Meaning that small changes in input leave most of the pooled outputs unchanged. Pooling takes surrounding pixels into account, whether taking maximum pixel intensity, an average of pixels or some other aggregation of nearby pixels. For example, the max pooling (Zhou and Chellappa, 1988) returns maximum output from a rectangular area. The pooling is useful because it allows the inputs to vary a little, meaning that the feature of input image does not have to be in the exact same spot. For example, in case of face recognition, the position of eyes does not have to be always the same, however, it is necessary that there are 2 eyes, one on the left and another on the right side of the face. Additionally, pooling can be used to downsample inputs, as it summarizes the pixels over nearby images. In order to achieve a downsampling with a stride of 2, the maximum pixel intensity is returned for every other pixel.

Models in this thesis use rectified linear activation function (ReLU), which only takes positive input into account using function $f(x) = max(0, x)$ (Nair and G. E. Hinton, 2010). ReLU has sparse activations and helps to propagate gradients. The sparse activation means that many of the nodes are not activated, this occurs because ReLU

converts a negative input into zero. ReLU helps to propagate gradients because the derivative of the function is 0 or 1, meaning the gradient is cancelled out or kept the same throughout the backpropagation.

In addition, batch normalization (Ioffe and Szegedy, 2015) is used to scale and normalize the activations, so that layers have inputs with zero mean and unit variance. The mean and unit variance is calculated based on current mini-batch, thus the name "batch" normalization. Batch normalization helps to improve performance and stability of neural networks.

Following subsections give an overview of convolutional models used in the thesis, also, the structure of models is described for datasets used in this thesis. More information about these datasets is given in sections 2.2 and 4.1

2.3.1 Residual Network

The residual network (ResNet) (He et al., 2015a) is a convolutional network that fits residual functions $F(x) = H(x) - x$, where x is input to the layer and $H(x)$ is underlying mapping fit by few weight layers. The input x, also called as identity mapping, is later added to $F(x)$, as shown in Figure 9.

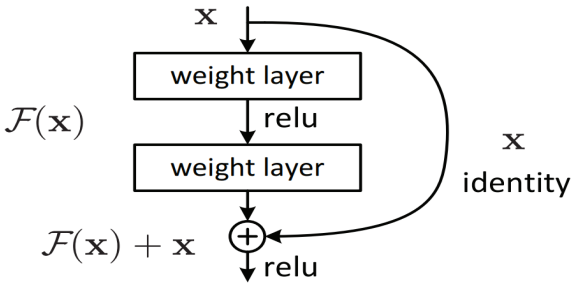


Figure 9. The building block of residual learning (He et al., 2015a).

Ultimately, a residual block still tries to fit underlying mapping $H(x)$, however the nonlinear layers of residual block do not have to approximate the identity mapping.

The residual architecture (He et al., 2015a) is used for all the datasets (section 2.2): CIFAR-10, CIFAR-100, ImageNet, Birds. In case of CIFAR datasets, the network has 110 layers. It is constructed of $6n + 2$ weight layers, first starting with the 3x3 convolutional layer. Next, there is a stack of $2n$ layers (or n residual blocks) for each feature map size $\{32, 16, 8\}$ with the filters $\{16, 32, 64\}$. Figure 10 shows the structure of residual block

with ReLU activation. After each convolutional layer, the batch normalization is also used. At the end of the model, there is used global average pooling and dense layer with softmax. The output size of last dense layer is based on the number of classes. The Table 1 gives an overview of the network architecture.

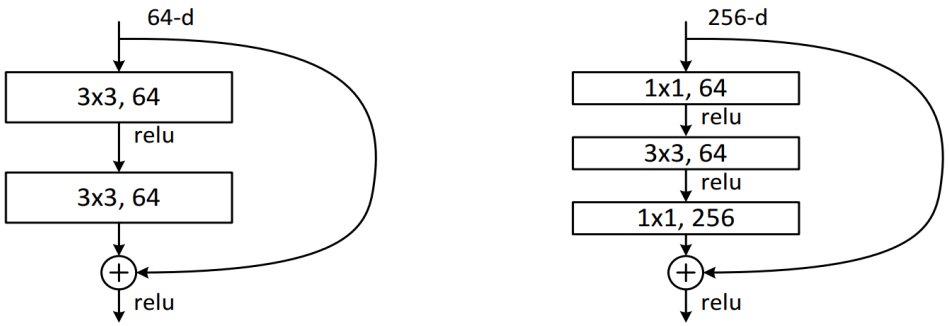


Figure 10. On the left, there is a usual residual block and on the right a "bottleneck" residual block (He et al., 2015a).

Table 1. The architecture of ResNet 110 (n=18) for CIFAR datasets (He et al., 2015a).

Output map size	**32x32**	**16x16**	**8x8**
Number of layers	1+2n	2n	2n
Number of filters	16	32	64

In case of ImageNet, the network has 50 or 152 layers, however, the structure is a bit different from the network for CIFAR datasets. Namely, it uses residual blocks with bottlenecks as shown in Figure 10. A more precise overview is given in Table 2, it is worth noting that after each convolutional layer, in addition to activation (ReLU), a batch normalization is used.

Table 2. The architecture for ResNet for ImageNet dataset for 50 and 152 weight layers (He et al., 2015a). After residual blocks down-sampling with a stride of 2 is done.

Layer name	**Output size**	**50-layer**	**152-layer**
conv1	112x112	[7x7, 64], stride 2	
conv2_x	56x56	3x3 max pool, stride 2	
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28x28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14x14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7x7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1x1	Average pool, 1000-d fc, softmax	

2.3.2 ResNet SD

Residual Network with Stochastic Depth (ResNet SD) (Huang, Sun, et al., 2016) is similar to the original residual network. The idea is to drop a random set of layers with a certain probability for each mini-batch training. The dropped layers are bypassed using identity function. The survival probabilities for the set of layers decay linearly according to the number of layers (Figure 11). In testing phase, all the layers are used for making a prediction.

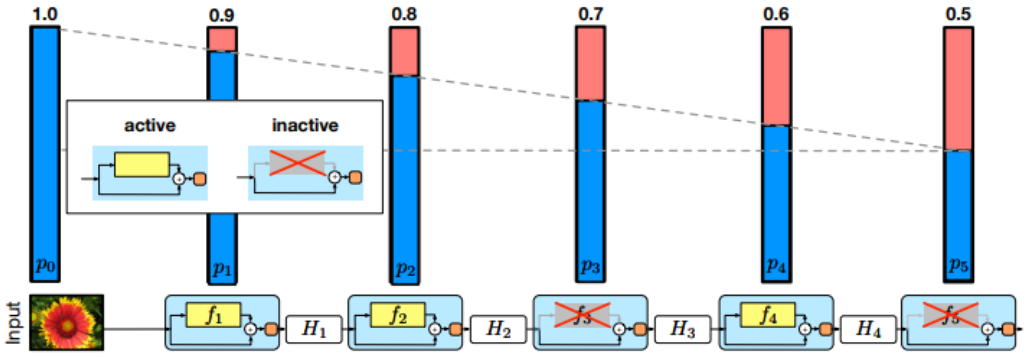


Figure 11. The linear decay of survival probability (p) for ResNet with stochastic depth (Huang, Sun, et al., 2016)

The ResNet SD architecture is used for CIFAR-10, CIFAR-100 and SVHN datasets. The architecture is exactly the same as shown previously in Table 1, except for adding stochastic depth. In addition, for SVHN a 152 layer network is used (n = 25).

2.3.3 DenseNet

Dense convolutional network (Huang, Liu, et al., 2016) contains dense blocks, where each layer takes identity mapping (feature maps) as an input from all the previous layers (Figure 12). By using the feature maps of previous layers, the dense blocks allow the maximum information flow, which helps to enhance feature propagation and encourage feature reuse. The number of filters added after each dense block is determined by growth rate k. The initial number of filters are $2 * k$. More precise overview of structures used in given thesis can be found in Table 3 and Table 4.

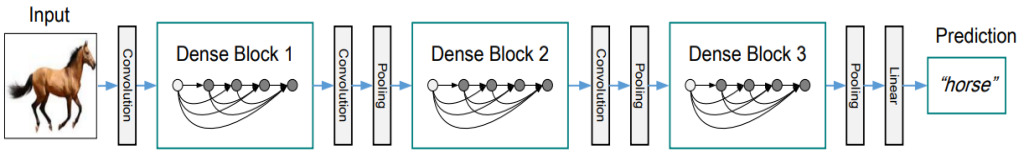


Figure 12. DenseNet with 3 dense blocks (Huang, Liu, et al., 2016).

Table 3. Structure of DenseNet 161 for ImageNet dataset with the growth rate of 48. All the convolutional layers (inside brackets) consist of sequence BN-ReLU-Conv (Huang, Liu, et al., 2016).

Layers	**Output size**	**DenseNet-161**
Convolution	112×112	$\left[7 \times 7\right] \times 1$, stride 2
Pooling	56×56	3×3 max pool, stride 2
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 6$
Translation Layer (1)	56×56	$\left[1 \times 1\right] \times 1$
	28×28	2×2 average pool, stride 2
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 12$
Translation Layer (2)	28×28	$\left[1 \times 1\right] \times 1$
	14×14	2×2 average pool, stride 2
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 32$
Translation Layer (3)	14×14	$\left[1 \times 1\right] \times 1$
	7×7	2×2 average pool, stride 2
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool
		1000D fully-connected, softmax

Table 4. Structure of DenseNet 40 for CIFAR-10/100 datasets with growth rate of 12. All the convolutional layers (inside brackets) consist sequence BN-ReLU-Conv (Huang, Liu, et al., 2016).

Layers	**Output size**	**DenseNet-40**
Convolution	32×32	$\boxed{3 \times 3} \times 1$
Dense Block (1)	32×32	$\boxed{3 \times 3} \times 12$
Translation Layer (1)	32×32	$\boxed{1 \times 1} \times 1$
	16×16	2×2 average pool, stride 2
Dense Block (2)	16×16	$\boxed{3 \times 3} \times 12$
Translation Layer (2)	16×16	$\boxed{1 \times 1} \times 1$
	8×8	2×2 average pool, stride 2
Dense Block (3)	8×8	$\boxed{3 \times 3} \times 12$
Classification Layer	1×1	8×8 global average pool
		10D fully-connected, softmax

2.3.4 Wide ResNet

Wide residual network (Wide ResNet) (Zagoruyko and Komodakis, 2016) increases the width and decreases the depth of a network. The idea is to alleviate the problem of diminishing feature reuse by using less residual blocks and increasing the number of filters of residual blocks. The diminishing feature reuse means that the features computed in earlier layers are fading away in the process of going through many layers. The network is widened by enlarging the number of filters in each block by factor k. The parameter N indicates how many blocks are in one group as shown in Table 5. Finally, to achieve 32-layer network, 5 blocks per group are used, meaning $N = 5$ and the growth factor is set to $k = 10$.

Table 5. Structure of Wide Residual Network (Zagoruyko and Komodakis, 2016) for CIFAR-10/100 datasets. The convolutional layers are in brackets with the filter size, the parameter k indicates the growth factor. The parameter N shows the number of residual blocks in one group. The downsampling is done in the first layers of "conv3" and "conv4"

Group name	Output size	Layers
conv1	32×32	$[3 \times 3,\ 16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3 \times 3,\ 32 \times k \\ 3 \times 3,\ 32 \times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$
output		10D fully-connected, softmax

3 Related Work

3.1 Necessity of Calibration

In this subsection, the findings of Guo et al. (Guo et al., 2017) about the necessity of calibration are brought out.

Firstly, the capacity of the model has an effect on calibration error. Namely, deeper and wider models result in higher calibration error, but on the other hand, these kind of models are necessary to achieve smaller classification error. In Figure 13, the far left plot shows classification error and expected calibration error (ECE) on ResNet model (He et al., 2015a) with 64 convolutional filters and a varying number of layers (depth). The middle left plot displays scores on ResNet model with 14 layers with various filter sizes (width).

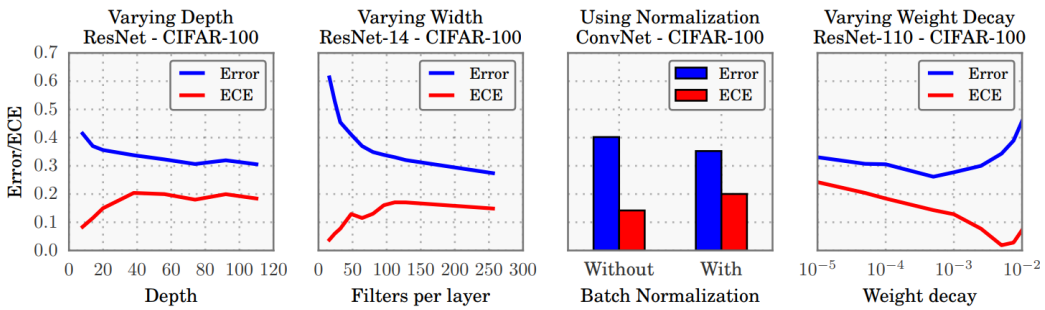


Figure 13. Calibration error (ECE) and error rate on models with varying depth (far left), varying width (middle left), using normalization or not (middle right) and varying weight decay (far right). Taken from Guo et al. paper (Guo et al., 2017).

In addition, Guo et al. (Guo et al., 2017) demonstrated an association of batch normalization to error rate and calibration error. The middle right plot of Figure 13, shows that on a 6-layer convolutional network. The normalization certainly helps to achieve better accuracy and enables using deep architectures but at the cost of accurate confidence estimations.

Last, but not least, Guo et al. (Guo et al., 2017) pointed out that calibration error is dependent on weight decay. More precisely, the smaller the weight decay, the higher the calibration error. The far right plot of Figure 13, displays weight decay in relation to calibration error on ResNet model with 110 layers.

3.2 Calibration in Neural Networks

Next, the calibration methods for different approaches and contexts are mentioned. All the papers below are written in the past few years and approach the calibration problem of neural networks from a different angle.

Kuleshov et al. (Kuleshov and Ermon, 2016) approach the calibration problem in online context, where it is possible that the inputs are potentially adversarial. As a result, a recalibration technique is proposed and tested on real-world datasets. Moreover, the technique does not make *i.i.d.* (independent and identical distribution) assumptions.

Kuleshov et al. (Kuleshov and Liang, 2015) get calibrated results for structured prediction problems, for example, speech recognition, optical character recognition and medical diagnosis. Calibration of structured predictions is challenging because of large output space and different types of probability queries by users. As a result, a special calibration method for structured predictions was tested on real-world datasets.

Lakshminarayanan et al. (Lakshminarayanan et al., 2016) demonstrate that deep ensembles can be used for uncertainty estimation. Well-calibrated uncertainty estimates that are as good or even better than approximate Bayesian neural networks are achieved. The method is also able to output reliable estimates in case of dataset shift.

Pereyra et al. (Pereyra et al., 2017) penalize overconfident predictions as a way of regularizing neural networks. In the end, this method is able to achieve better results in various domains: image classification, language modeling, machine translation and speech recognition.

Hendrycks et al. (Hendrycks and Gimpel, 2016) show a way to detect out-of-distribution samples using the confidence of predictions. In order to achieve that, simple statistics derived from softmax prediction probabilities are used as a baseline.

Gal et al. (Gal and Ghahramani, 2015) represent model uncertainty using dropout (Srivastava et al., 2014) in neural networks. The dropout is used as a Bayesian approximation without sacrificing computational complexity or test accuracy.

Bendale et al. (Bendale and Boult, 2016) propose a way to adapt deep neural networks for open set recognition. For that, a new model layer, called OpenMax, is introduced. It outputs probability of the sample being from an unknown class.

Zhang et al. (Z. Zhang et al., 2018) make confidence analysis for each individual image. In order to do that, a progressive neural network structure is proposed, which on each stage has to check whether the prediction is confident enough to proceed or a more

complex model has to be used.

Keren et al. (Keren et al., 2018) output calibrated prediction intervals in case of neural network regressors. Instead of outputting a single scalar value, the model gives out an interval and calibrated confidence of output being in that range.

4 Methods

This chapter gives an overview of used datasets and training procedures to obtain the final replicated results. The models and training procedures can be found in a Github repository[1].

4.1 Data Preparation

This subsection gives overview of the preprocessing of previously mentioned datasets (subsection 2.2). All the datasets follow same preprocessing and data augmentation process as mentioned in Guo et al. paper (Guo et al., 2017). The data preparation was not described in the paper, instead references to the original articles were given.

4.1.1 CIFAR-10

The CIFAR-10 dataset is preprocessed before using it for training. The images are normalized by subtracting the per-channel mean of the training images and dividing by the standard deviation. The only exception was ResNet 110 model, where per-pixel normalization was used.

In order to increase the number of images for training, the data is also augmented. The images are flipped horizontally in 50 percent of cases. Additionally, the training images are also padded by 4 pixels from every side and then randomly cropped, so the final image size is 32x32. The padded pixels are either reflection of the image or constant value 0, also dependent on the model.

In case of testing the model, the preprocessed images from a test set with no augmentations are used.

4.1.2 CIFAR-100

The data preprocessing and data augmentation of CIFAR-100 is done in the same manner as for CIFAR-10. Normalizing the images by subtracting the mean and dividing by the standard deviation. Also, horizontal flipping and translation techniques are used.

[1] https://github.com/markus93/NN_calibration

4.1.3 ImageNet

Imagenet data preprocessing and augmentation depends on the model and may differ in reality, because in given thesis pre-trained weights were used for Imagenet models. The preprocessing covers scale augmentation (Simonyan and Zisserman, 2014), more precisely the image is scaled so the shorter side is either 256 or 480 pixels. That image is randomly cropped to get dimensions 224x224, horizontally flipped and the mean subtracted from it. Finally, the colour augmentation (Krizhevsky et al., 2012) is used.

In case of testing, the images are resized so the shorter side would be 256 pixels and then centre cropped for final 224x224 pixels.

4.1.4 Street View House Numbers (SVHN)

For SVHN dataset preprocessing, the images are normalized by subtracting the per-channel mean of the training set and dividing by the standard deviation. However, no data augmentation is performed as noted in the paper (Huang, Sun, et al., 2016).

4.1.5 Birds

The Caltech-UCSD Birds dataset is used for fine-tuning ImageNet model, so the per-channel means of ImageNet training set are subtracted from images. Also, the data preparation process is similar, scaling the images so the shorter side would be 256 pixels and after the image is randomly cropped into the size of 224x224 and horizontally flipped.

While testing, the images are also scaled so the shorter side is 256 pixels and then centre cropped into a 224x224 image.

4.2 Training

All the models had to be trained by the author of this thesis, except for ImageNet. This occurred because for the calibration process a separate validation set is essential and in order to keep the same split ratios to Guo et al. (Guo et al., 2017) the train set had to be split. However, the available pre-trained models used all the training data or did not bring out the indices of a validation split. Fortunately, pre-trained models for ImageNet were usable, as following the Guo et al. (Guo et al., 2017), the test set was split for validation

dataset, instead of the training set. All the model weights trained and used for this thesis are available in the same Github project[2].

The models are implemented using open source deep learning library Keras (version 2.1.4) (Chollet et al., 2015) with TensorFlow (version 1.4.1) backend (Abadi et al., 2015). The models are trained using same hyperparameters as presented in papers, if available, nonetheless, the hyperparameters are still brought out for clarity in following subsections.

The models, if not stated differently, are trained using Stochastic Gradient Descent (SGD) optimizer with a Nesterov momentum of 0.9 without dampening. The weights are initialized using He normalization (He et al., 2015b) and the weight decay is set to 0.0001. In the end, the final model, after training for all the epochs, is used for testing.

4.2.1 ResNet

In case of CIFAR-10 and CIFAR-100 datasets, a script from a public repository is used (Li, 2018) as a base and necessary modifications are done to match the parameters of the original model (He et al., 2015a). The 110-layer network is trained with a mini-batch size of 128. The starting learning is 0.1 and it is divided by 10 at 100 and 150 epochs, running up to 200 epochs.

For Birds dataset, the model with pre-trained weights of 50-layer ResNet on ImageNet dataset is provided by Keras library (Chollet et al., 2015). These weights are used as a starting point for fine-tuning Birds dataset. The learning rate is set to 0.0001 with a decay of 10^{-6}. The model is trained up to 250 epochs or up to moment the validation loss has not changed for 10 previous epochs.

Pre-trained weights of ResNet 152 are used (Yu, 2017b) also for ImageNet dataset. The initial learning rate of the model is 0.1 and is divided by 10, each time the model stops improving. The model is trained up to 120 epochs with a mini-batch size of 256. The test is run on the model provided by the same author (Yu, 2017b).

4.2.2 ResNet SD

The stochastic depth model implementation is taken from a public repository (Chen, 2018) and essential modifications to match the proposed model are done.

The CIFAR-10 and CIFAR-100 datasets are trained on 110-layer network (Huang, Sun, et al., 2016). The mini-batch size is 128 and starting learning rate is 0.1, which is

[2]https://github.com/markus93/NN_calibration/tree/master/models

divided by 10 at epoch 250 and 375, running up to 500 epochs. Finally, the model with best validation accuracy is used.

The SVHN data set is trained on 152-layer network (Huang, Sun, et al., 2016) with same parameters as for CIFAR datasets, however, the learning rate is divided by 10 at epoch 30 and 35 and the model is trained for 50 epochs. Additionally, for first epoch learning rate 0.04 is used, otherwise, the model did not start converging.

4.2.3 DenseNet

The dense network model implementations are taken from a public repository (Majumdar, 2018a; Yu, 2017a) adding only modifications crucial to match original training procedure.

The CIFAR-10 and CIFAR-100 datasets are trained on the 40-layer network with the growth rate of 12 (Huang, Liu, et al., 2016). The mini-batch size is 64 and starting learning rate is 0.1, which is divided by 10 at epoch 150 and 225. After training for 300 epochs, the model with best validation accuracy is used.

The model for ImageNet dataset is a pre-trained 161-layer network (Huang, Liu, et al., 2016), however, the learning process for this was similar to CIFAR datasets, except it was trained for 90 epochs with batch size 256 and the learning rate is divided by 10 at 30 and 60 epoch.

4.2.4 Wide ResNet

In case of Wide Residual networks, implementation from a public repository is used (Majumdar, 2018b).

The CIFAR-10 and CIFAR-100 datasets are trained on 32-layer network with growth factor k of 10. However, in the article (Guo et al., 2017) there was no information about the actual growth factor given. In addition, the depth measure from the original article (Zagoruyko and Komodakis, 2016) did not match with the one used by other residual network authors (He et al., 2015a; Huang, Sun, et al., 2016).

The model is trained in mini-batches with the size of 128 and starting learning rate of 0.1 (Zagoruyko and Komodakis, 2016). The model is trained for 200 epochs and the learning rate is divided by 20 at epoch 60, 120 and 160. In addition, the weight decay is set to 0.0005, which is different from all the other models.

5 Results

This chapter gives an overview of the achieved results and comparison to Guo et al. More precisely, there are detailed tables of ECE, MCE, NLL, error rate and Brier score. In addition, some insight into reliability diagrams is given.

In the comparison the scores on uncalibrated models were used as well as the calibrated probabilities obtained with histogram binning, isotonic regression, temperature scaling and beta calibration. The histogram binning models were trained using bin size M of 15, and the bins were divided into intervals of equal lengths. BetaCal package (Kull et al., 2017) was used for 3 parameter beta calibration and Scikit-learn (Pedregosa et al., 2011) library was used for isotonic regression.

The final results are achieved using an hold-out test set, it is separate from the training set and the validation set. In order to calculate ECE and MCE, the number of bins was set to $M = 15$. Following tables contain all the models and calibration methods used for the datasets. Information about these can be found from Chapters 4 and 2.1. The abbreviated names of models are followed by the number of convolutional layers in them. In addition, a grey colour represents the original score achieved by Guo et al. (Guo et al., 2017) and the best results are in bold. The average rank is added in the end of the tables for easier comparison. The subscripted numbers after scores indicate the rank. The scores of replicated results and Guo et al. results are ranked separately.

The model output logits used for this thesis are available in same Github project[3], these are in a serialized format in Python programming language (version 3.6.4) and include labels and logits of validation and test set.

5.1 ECE

In Table 6, expected calibration errors (ECE) are shown for various models and calibration methods. The best performing method is temperature scaling and it is the case for all the image classification models. The follow-up method is beta calibration, mostly it is rather close to temperature scaling score, however, in some cases, it performs much worse (i.e. DenseNet 40 with CIFAR-100).

[3] https://github.com/markus93/NN_calibration/tree/master/logits

Table 6. ECE (%) ($M = 15$ bins) results for various models and calibration methods. The best scores are in bold and results achieved by Guo et al. (Guo et al., 2017) are coloured in grey for comparison.

Method	Data	Uncal	Uncal	Histo	Histo	Iso	Iso	Temp	Temp	Beta
Resnet 50	**Birds**	2.35_2	9.19_4	12.63_5	4.34_2	8.65_4	5.22_3	$\mathbf{1.69_1}$	1.85_1	3.11_3
Resnet 110	**CIFAR-10**	4.75_5	4.6_4	1.25_2	0.58_1	1.47_4	0.81_2	$\mathbf{1.13_1}$	0.83_3	1.42_3
Resnet 110	**CIFAR-100**	18.48_5	16.53_4	9.06_4	2.66_2	6.54_3	4.99_3	$\mathbf{2.38_1}$	1.26_1	4.60_2
Resnet 110 (SD)	**CIFAR-10**	4.11_5	4.12_4	1.19_4	0.67_2	1.03_3	1.11_3	$\mathbf{0.56_1}$	0.60_1	0.98_2
Resnet 110 (SD)	**CIFAR-100**	15.86_5	12.67_4	7.60_4	2.46_2	5.21_3	4.16_3	$\mathbf{1.21_1}$	0.96_1	3.55_2
DenseNet 40	**CIFAR-10**	5.50_5	3.28_4	2.13_4	0.44_2	1.68_2	0.61_3	$\mathbf{0.95_1}$	0.33_1	1.70_3
DenseNet 40	**CIFAR-100**	21.16_5	10.37_4	11.97_4	2.68_2	5.25_2	5.85_3	$\mathbf{0.90_1}$	1.18_1	6.03_3
Wide ResNet 32	**CIFAR-10**	4.51_5	4.52_4	1.01_3	0.72_2	1.19_4	1.08_3	$\mathbf{0.80_1}$	0.54_1	0.97_2
Wide ResNet 32	**CIFAR-100**	18.78_5	15_4	7.64_4	3.01_2	5.82_2	4.51_3	$\mathbf{0.78_1}$	2.32_1	7.01_3
ResNet 152	**ImageNet**	6.65_3	5.48_4	8.92_5	4.36_2	7.05_4	4.77_3	$\mathbf{1.47_1}$	1.86_1	3.47_2
DenseNet 161	**ImageNet**	5.72_3	6.28_4	8.71_5	4.52_2	6.90_4	5.18_3	$\mathbf{1.94_1}$	1.99_1	3.10_2
Resnet 152 (SD)	**SVHN**	0.86_5	0.44_4	0.53_3	0.14_1	$\mathbf{0.25_1}$	0.28_3	0.61_4	0.17_2	0.50_2
Average Rank		4.42	4.00	3.92	1.83	3.00	2.92	1.25	1.25	2.42

In comparison with Guo et al. (Guo et al., 2017) results, the calibration errors are very similar, the only larger difference is between errors of DenseNet 40 with CIFAR-100 dataset, and there is 10 units difference already in uncalibrated errors, however the ECEs of temperature scaling are still smaller compared to Guo et al. result. Additionally, the uncalibrated ECEs of the replicated results differ from Guo et al. results on average by 2.59 units, the errors of histogram binning and isotonic regression differ on average by 3.84 and 1.06 units respectively. On the other hand the ECEs of temperature scaling differ only by 0.45 units on average.

5.2 Reliability Diagrams

The reliability diagrams on Figure 14 give a comparison between different calibration methods on Resnet 110 on CIFAR-10 dataset. It is possible to see that some of the bins with lower confidence have rather high gap (red bar) - a difference between confidence and accuracy. That may be the case, because of low number of elements in the bin or unlucky validation split. Based on the ECE score, one could say that histogram binning is rather good at calibration, however looking at the Figure 14, the calibrated output of histogram binning is rather bad at higher confidences. This occurs because of the distribution of samples in the bins. Figure 15 shows that most of the samples fall into

last bin.

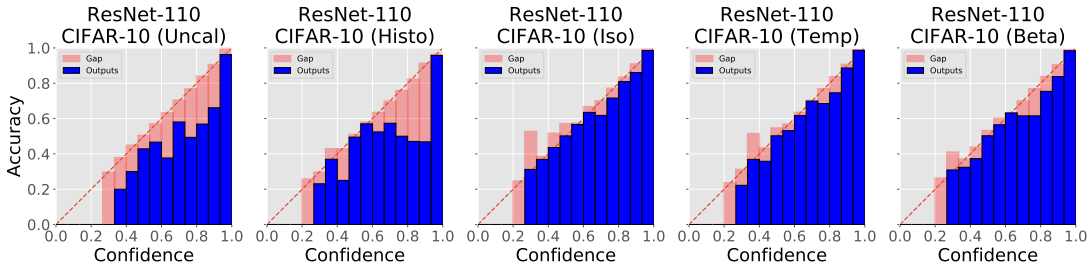


Figure 14. Reliability diagrams ($M = 15$) for ResNet 110 on CIFAR-10 dataset for various calibration methods.

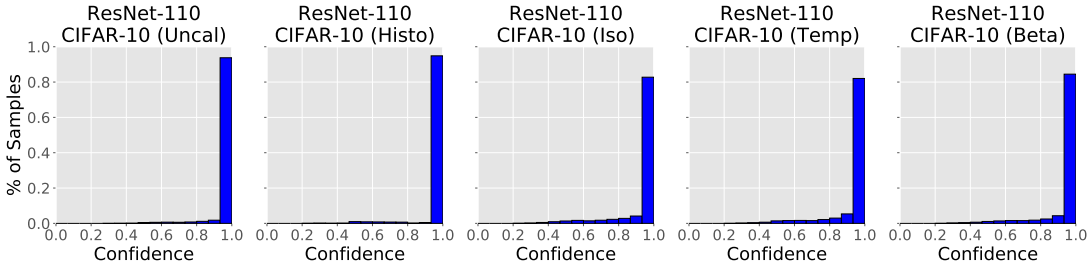


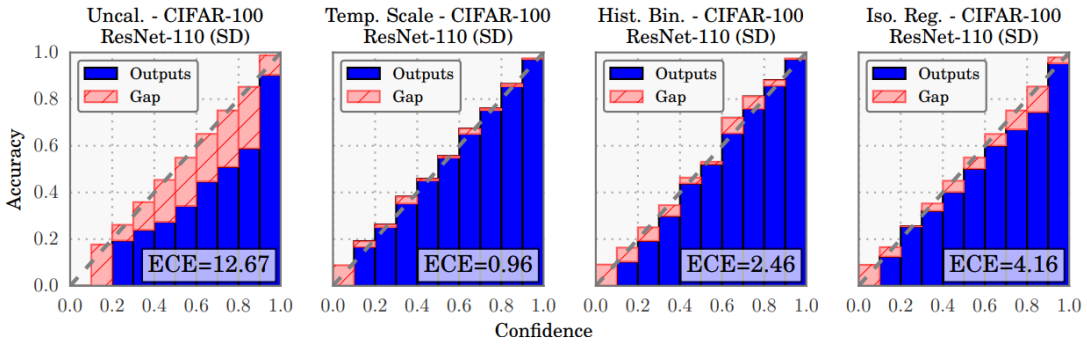
Figure 15. Distribution ($M = 15$) of ResNet 110 on CIFAR-10 dataset for various calibration methods.

On Figure 16, there are reliability diagrams created by Guo et al. (Guo et al., 2017) and by the author of this thesis shown side by side. The diagrams look really similar, except for histogram binning, which looks much worse in second case. Last is also indicated by the difference of the ECE scores, 2.46 compared to 7.60.

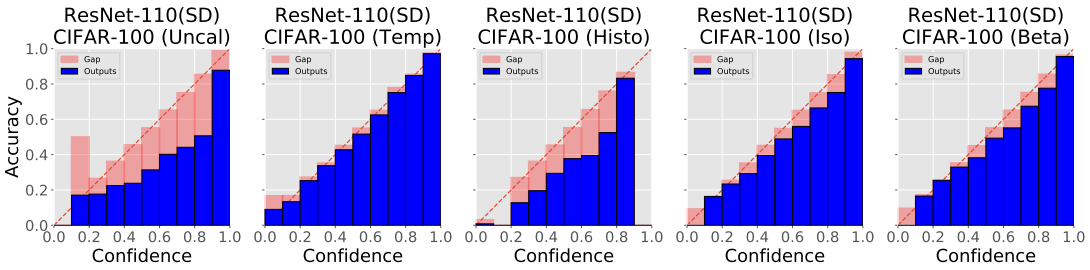
5.3 MCE

Maximum Calibration Error is the lowest for temperature scaling (Table 7), except for two models. On reliability diagrams (Figure 14) the largest red gap corresponds to MCE. Based on that the largest error is in confidence interval $(3/15, 4/15]$, except for histogram binning, where the interval is $(13/15, 14/15]$.

The ECE results were quite similar in comparison with Guo et al. (Guo et al., 2017). On the other hand, the MCE scores vary a lot, this is largely affected by the model training procedure and validation set that is used for calibration.



(a) Reliability diagrams ($M = 10$) for ResNet 110 (SD) on CIFAR-100 dataset for various calibration methods by Guo et al. (Guo et al., 2017).



(b) Reliability diagrams ($M = 10$) of ResNet 110 (SD) on CIFAR-100 dataset for various calibration methods.

Figure 16. Comparison of reliability diagrams, (a) is created by Guo et al. (Guo et al., 2017) and (b) is created by the author of this thesis.

Table 7. MCE (%) ($M = 15$ bins) results for various models and calibration methods.

Method	Data	Uncal	Uncal	Histo	Histo	Iso	Iso	Temp	Temp	Beta
Resnet 50	**Birds**	27.29_4	30.06_4	42.90_5	25.35_3	22.89_2	16.59_2	27.14_3	9.08_1	$\mathbf{13.18_1}$
Resnet 110	**CIFAR-10**	29.58_4	33.78_4	44.45_5	26.87_3	24.58_2	7.80_1	$\mathbf{23.64_1}$	8.56_2	26.22_3
Resnet 110	**CIFAR-100**	39.88_5	35.50_4	31.64_4	7.03_2	13.38_3	10.36_3	$\mathbf{7.10_1}$	4.74_1	11.44_2
Resnet 110 (SD)	**CIFAR-10**	32.48_4	34.52_4	49.09_5	17.00_3	8.14_2	16.45_2	$\mathbf{7.82_1}$	15.45_1	10.03_3
Resnet 110 (SD)	**CIFAR-100**	48.29_5	26.42_4	28.97_4	9.12_2	11.58_3	10.95_3	$\mathbf{4.10_1}$	8.85_1	10.83_2
DenseNet 40	**CIFAR-10**	33.40_4	22.44_4	45.10_5	7.77_2	$\mathbf{8.49_1}$	19.54_3	9.93_2	4.58_1	25.03_3
DenseNet 40	**CIFAR-100**	45.40_5	21.52_4	16.94_4	9.36_1	12.16_2	10.59_2	$\mathbf{2.21_1}$	19.40_3	12.92_3
Wide ResNet 32	**CIFAR-10**	37.22_4	27.97_4	32.98_3	12.19_3	26.72_2	6.19_1	$\mathbf{7.06_1}$	9.11_2	74.58_5
Wide ResNet 32	**CIFAR-100**	45.64_5	33.11_4	41.87_4	6.22_2	14.07_2	14.87_3	$\mathbf{3.61_1}$	5.33_1	15.16_3
ResNet 152	**ImageNet**	14.29_4	12.20_2	28.32_5	14.57_4	11.03_3	8.74_1	$\mathbf{6.78_1}$	12.29_3	8.85_2
DenseNet 161	**ImageNet**	13.07_3	14.07_4	27.74_5	13.14_3	13.39_4	11.57_1	$\mathbf{4.97_1}$	12.29_2	10.88_2
Resnet 152 (SD)	**SVHN**	25.03_4	19.36_4	22.28_2	11.16_1	24.78_3	18.67_3	$\mathbf{18.24_1}$	18.05_2	25.90_5
Average Rank		4.25	3.83	4.25	2.42	2.42	2.08	1.25	1.67	2.83

In this case, MCE gives good idea how bad the model can perform in the worst case scenario, however it does not indicate in which confidence range it is more likely to be poorly-calibrated.

5.4 Brier Score

Brier scores are really similar for all the calibration methods (Table 8). The best is mostly histogram binning, except for Resnet 152 (SD), and for Wide ResNet 32, where isotonic regression and beta calibration are the best, respectively.

The results from Brier score are opposing to ECE and MCE results, where the histogram binning was performing the worst. Guo et al. (Guo et al., 2017) did not include Brier score, so it cannot be confirmed whether Brier score should give conflicting results to ECE and MCE.

Table 8. Brier score results for various models and calibration methods.

Method	Data	Uncal	Histo	Iso	Temp	Beta
Resnet 50	**Birds**	0.5263_2	$\mathbf{0.5072_1}$	0.5303_3	0.5313_4	0.5687_5
Resnet 110	**CIFAR-10**	0.8316_5	$\mathbf{0.7792_1}$	0.7904_4	0.7881_3	0.7864_2
Resnet 110	**CIFAR-100**	0.6611_5	$\mathbf{0.5110_1}$	0.5378_4	0.5249_3	0.5196_2
Resnet 110 (SD)	**CIFAR-10**	0.8338_5	$\mathbf{0.7854_1}$	0.7971_4	0.7920_2	0.7951_3
Resnet 110 (SD)	**CIFAR-100**	0.6614_5	0.5428_2	0.5666_4	$\mathbf{0.5356_1}$	0.5523_3
DenseNet 40	**CIFAR-10**	0.8192_5	$\mathbf{0.7631_1}$	0.7783_4	0.7702_2	0.7747_3
DenseNet 40	**CIFAR-100**	0.6550_5	$\mathbf{0.4711_1}$	0.5170_4	0.4914_2	0.4955_3
Wide ResNet 32	**CIFAR-10**	0.8388_5	$\mathbf{0.7811_1}$	0.7968_4	0.7897_2	0.7946_3
Wide ResNet 32	**CIFAR-100**	0.6987_5	$\mathbf{0.5347_1}$	0.5849_4	0.5631_3	0.5626_2
ResNet 152	**ImageNet**	0.6476_5	0.5933_2	$\mathbf{0.5856_1}$	0.5949_3	0.5962_4
DenseNet 161	**ImageNet**	0.6518_5	0.5976_2	$\mathbf{0.5975_1}$	0.6049_3	0.6067_4
Resnet 152 (SD)	**SVHN**	0.9072_5	0.8859_2	0.8881_4	0.8871_3	$\mathbf{0.8847_1}$
Average Rank		4.75	1.33	3.42	2.58	2.92

5.5 Error Rate

The error rates (Table 9) are really similar compared to Guo et al. (Guo et al., 2017). The most consistently performing is naturally temperature scaling as it does not affect predictions. However, the best overall error rates are achieved by beta calibration, although it reduces the accuracy of the model in a couple of cases. On the other hand, histogram binning method increases the error rate the most and isotonic regression affects error rate in a negative direction in case of larger number of classes.

Table 9. Error rate (%) results for various models and calibration methods.

Method	Data	Uncal	Uncal	Histo	Histo	Iso	Iso	Temp	Temp	Beta
Resnet 50	**Birds**	$\mathbf{26.86_1}$	22.54_1	3.77_5	55.02_4	34.31_4	23.37_3	$\mathbf{26.86_1}$	22.54_1	27.20_3
Resnet 110	**CIFAR-10**	6.44_2	6.21_1	6.59_5	6.45_4	$\mathbf{6.36_1}$	6.36_3	6.44_2	6.21_1	6.44_2
Resnet 110	**CIFAR-100**	28.52_2	27.83_1	31.26_5	34.78_4	29.31_4	28.41_3	28.52_2	27.83_1	$\mathbf{28.36_1}$
Resnet 110 (SD)	**CIFAR-10**	5.96_3	5.64_3	6.20_5	5.59_1	5.91_2	5.62_2	5.96_3	5.64_3	$\mathbf{5.86_1}$
Resnet 110 (SD)	**CIFAR-100**	27.17_2	24.91_1	29.74_5	33.78_4	27.47_4	25.42_3	27.17_2	24.91_1	$\mathbf{26.32_1}$
DenseNet 40	**CIFAR-10**	$\mathbf{7.58_1}$	5.91_1	7.93_5	6.12_4	7.65_4	5.96_3	$\mathbf{7.58_1}$	5.91_1	7.59_3
DenseNet 40	**CIFAR-100**	30.00_2	26.45_1	32.49_5	34.78_4	30.22_4	26.73_3	30.00_2	26.45_1	$\mathbf{29.81_1}$
Wide ResNet 32	**CIFAR-10**	6.07_3	6.96_1	6.18_5	7.30_4	5.98_2	7.01_3	6.07_3	6.96_1	$\mathbf{5.94_1}$
Wide ResNet 32	**CIFAR-100**	26.18_2	28.00_1	28.89_5	34.29_4	26.33_4	28.61_3	26.18_2	28.00_1	$\mathbf{25.98_1}$
ResNet 152	**ImageNet**	23.80_2	22.31_1	31.32_5	48.10_4	28.16_4	22.94_3	23.80_2	22.31_1	$\mathbf{23.72_1}$
DenseNet 161	**ImageNet**	22.95_2	22.57_1	30.66_5	48.32_4	26.82_4	23.20_3	22.95_2	22.57_1	$\mathbf{22.77_1}$
Resnet 152 (SD)	**SVHN**	1.85_2	1.98_1	2.07_5	2.06_4	1.96_4	2.04_3	1.85_2	1.98_1	$\mathbf{1.82_1}$
Average Rank		2.00	1.17	5.00	3.75	3.42	2.92	2.00	1.17	1.42

5.6 Negative Log Likelihood

Negative log likelihood acts similarly to error rate (Table 10). Meaning it is negatively affected in case of histogram binning and isotonic regression and performs the best using temperature scaling and beta calibration.

Table 10. NLL results for various models and calibration methods.

Method	Data	Uncal	Uncal	Histo	Histo	Iso	Iso	Temp	Temp	Beta
Resnet 50	**Birds**	$\mathbf{0.9859_1}$	0.9786_2	10.5162_5	1.6226_4	4.3645_4	1.4128_3	0.9862_2	0.8792_1	1.2926_3
Resnet 110	**CIFAR-10**	0.3583_4	0.3285_4	0.5472_5	0.2532_3	0.2708_3	0.2237_2	$\mathbf{0.2093_1}$	0.2102_1	0.2139_2
Resnet 110	**CIFAR-100**	1.6937_3	1.4978_4	4.2139_5	1.4379_3	1.8926_4	1.2070_2	$\mathbf{1.0917_1}$	1.0442_1	1.1318_2
Resnet 110 (SD)	**CIFAR-10**	0.3033_4	0.2959_4	0.4994_5	0.2027_3	0.2544_3	0.1867_2	$\mathbf{0.1776_1}$	0.1718_1	0.1856_2
Resnet 110 (SD)	**CIFAR-100**	1.3525_3	1.1157_3	4.0187_5	1.1985_4	1.6371_4	1.0317_2	$\mathbf{0.9421_1}$	0.8613_1	0.9643_2
DenseNet 40	**CIFAR-10**	0.4282_4	0.2228_4	0.5725_5	0.2120_3	0.2773_3	0.1969_2	$\mathbf{0.2251_1}$	0.1750_1	0.2392_2
DenseNet 40	**CIFAR-100**	2.0174_4	1.0134_2	4.1829_5	1.2156_4	1.6491_3	1.0615_3	$\mathbf{1.0571_1}$	0.9026_1	1.1532_2
Wide ResNet 32	**CIFAR-10**	0.3817_4	0.3293_4	0.5132_5	0.2778_3	0.2327_3	0.2428_2	$\mathbf{0.1915_1}$	0.2283_1	0.2020_2
Wide ResNet 32	**CIFAR-100**	1.8022_4	1.3434_3	3.9352_5	1.4499_4	1.5607_3	1.2086_2	$\mathbf{0.9445_1}$	1.0565_1	1.0386_2
ResNet 152	**ImageNet**	0.9885_2	0.8961_2	6.1591_5	1.4507_4	2.8520_4	1.1859_3	$\mathbf{0.9421_1}$	0.8657_1	0.9947_3
DenseNet 161	**ImageNet**	0.9440_2	0.9338_2	6.1903_5	1.4716_4	2.8857_4	1.1912_3	$\mathbf{0.9093_1}$	0.8885_1	0.9733_3
Resnet 152 (SD)	**SVHN**	0.0854_3	0.0842_2	0.2887_5	0.1137_4	0.1093_4	0.0950_3	$\mathbf{0.0786_1}$	0.0821_1	0.0796_2
Average Rank		3.17	3.00	5.00	3.58	3.50	2.42	1.08	1.00	2.25

5.7 Computational efficiency

Temperature scaling calibration method was computationally the most efficient, and arguably it was the easiest one to include into a deep learning model. Furthermore, temperature scaling was 5 times faster than its closest competitor beta calibration in case of ImageNet dataset, which also had the largest validation dataset and most classes. Otherwise, the running times were close to 2-3 seconds for all the methods. Temperature scaling was able to converge in less than 10 iterations for all the models.

The models were trained using a single GPU (NVIDIA Tesla P100) provided by the High Performance Computing Center of University of Tartu. The training for one epoch took up to 160 seconds, so in total up to 13 hours for CIFAR-10, CIFAR-100 and Birds datasets. ResNet with Stochastic Depth trained for 13 hours, Wide ResNet and DenseNet both trained for 9 hours and ResNet for 5 hours. The ResNet (SD) on SVHN dataset was trained for 19 hours.

6 Discussion

In process of replication, it was confirmed that temperature scaling works surprisingly well on neural networks with image datasets. Temperature scaling outperformed other calibration methods on almost every dataset and model using various error measures. Only the error rates of beta calibration were slightly smaller than for temperature scaling. One might think that other methods performed worse than temperature scaling because of overfitting. However, it was not the case because using temperature scaling the ECE scores were also smaller on the validation data (Figure 17).

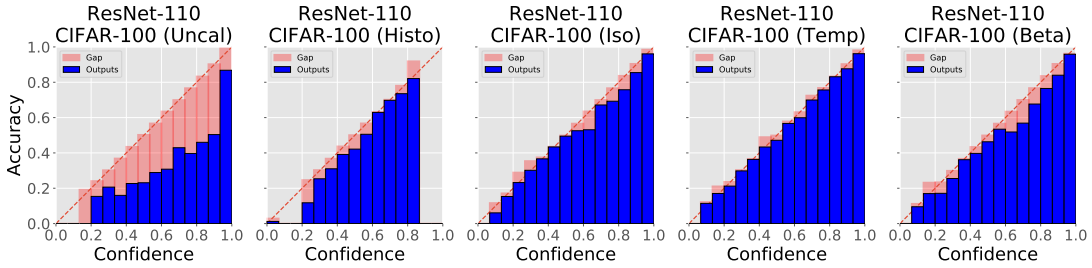


Figure 17. Reliability diagrams (M = 15) for ResNet 110 on CIFAR-100 validation dataset for various calibration methods.

6.1 Problems with Replication

Unfortunately, there was not much information about how Guo et al. (Guo et al., 2017) implemented their models, it was only said that implementation details from original papers were followed. However, in some cases, the original paper was rather vague about the implementation and had two different approaches for data preprocessing (Zagoruyko and Komodakis, 2016; Huang, Liu, et al., 2016) and had separate results for data augmentation (Huang, Liu, et al., 2016). Some of the papers also did not mention whether per-pixel or per-channel normalization was used or were vague about the preprocessing (He et al., 2015a; Huang, Sun, et al., 2016; Zagoruyko and Komodakis, 2016).

Furthermore, the SVHN did not start converging with the suggested learning rate of 0.1, even after 30 epochs. The starting learning rate was changed to 0.04 for the initial epoch. Additionally, in case of Birds data set, the fine-tuning approach was used, but the hyperparameters and data preprocessing was not described at all.

In addition, Guo et al. (Guo et al., 2017) did not write about the growth factor of

Wide ResNet model and did not follow the depth measure in the original paper, because following the format there, it was impossible to get 32 layer network. Actually, the weight layers were counted as in original Residual network paper (He et al., 2015a), which left out convolutional projection layers that carry residuals. These projections were used when the number of filters or size of input changed.

In case of ImageNet, it is usually tested using 4 corner crops and 1 centre crop from the image and its horizontal flip, getting total of 10 patches. Based on these, predictions of the model are averaged. However, in this thesis, only centre crop was used for testing. Saying that, in given thesis, the last model was used, instead of a model with the best validation score.

These problems made it much harder to train the models and to achieve similar test error rates to the replicated paper. It was also the cause why scores for LeNet model (LeCun et al., 1998) and Stanford Cars dataset (Krause et al., 2013) were left out from final results. More precisely, in case of LeNet, the preprocessing and hyperparameters were not described and the error rates achieved were 10 percent higher compared to Guo et al. results. ResNet 50 model on Stanford Cars dataset started converging, but only for training set as it did not learn anything for the validation set. Although the preprocessing and fine-tuning for Cars dataset was done similarly to training on Birds dataset.

It is worth noting that some details of the models might be different from the original. Nonetheless, the results achieved in this thesis are comparable to Guo et al. (Guo et al., 2017) results and the minor differences should not matter too much.

6.2 Contributions

Firstly, the author's contributions to the work were modifying models according to the needs. Despite having models readily available on the web, there were some complications with couple of the models. For example, one of the ResNet models with Stochastic depth was not working properly and did not achieve good enough results even though the same hyperparameters as described in original paper were used.

Secondly, all the datasets except CIFAR-10 and CIFAR-100, had to be loaded in from scratch. The datasets preprocessing needed to be done, because of the extra validation set. Meaning pre-calculated means and variances could not be used. Fortunately Keras had data augmentation implemented, only random cropping had to be added.

Lastly, reliability diagrams and evaluation of the models were implemented by the

author. In addition, temperature scaling and histogram binning were added to work-flow, however, those were rather easy to implement. All the replication process is available on public Github repository[4].

[4]https://github.com/markus93/NN_calibration

7 Conclusion

In conclusion, many of the results of Guo et al. (Guo et al., 2017) paper were replicated and compared to beta calibration. In the process, there were used 4 convolutional neural network models: Residual Network (ResNet), ResNet with Stochastic Depth, Wide ResNet and DenseNet. These models were trained on following datasets: CIFAR-10/100, Imagenet, SVHN and Caltech-UCSD Birds. The results of the models were calibrated using histogram binning, isotonic regression and temperature scaling. Additionally, beta calibration method was added to the comparison. As a result, temperature scaling method was superior and outperformed other calibration methods on various error measures. Beta calibration had performed a little bit better in case of error rate, but regarding other calibration measures, temperature scaling clearly outperformed it.

Future work is to add more models with different architectures and find out why Brier scores give conflicting results compared to ECE and MCE. Also to add some extra multiclass calibration methods that were not done by Guo et al. to get more in-depth analysis. Furthermore, it would be interesting to understand what qualities neural network has or should have in order to be suitable for temperature scaling.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. (Cit. on p. 31).

Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T., & Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. *The annals of mathematical statistics*, 641–647. (Cit. on p. 12).

Bendale, A. & Boult, T. E. (2016). Towards open set deep networks. In *The ieee conference on computer vision and pattern recognition (cvpr)*. (Cit. on p. 27).

Bengio, Y., Goodfellow, I. J., & Courville, A. (2015). Deep learning. *Nature*, *521*(7553), 436–444. (Cit. on p. 10).

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... Zhang, J. et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*. (Cit. on p. 6).

Brier, G. W. & Allen, R. A. (1951). Verification of weather forecasts. In *Compendium of meteorology* (pp. 841–848). Springer. (Cit. on p. 10).

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1721–1730). ACM. (Cit. on p. 6).

Chen, L. (2018). Implementation of stochastic depth networks in Keras. accessed 05/14/2018, from https://github.com/transcranial/stochastic-depth. (Cit. on p. 31)

Chollet, F. et al. (2015). Keras. https://keras.io. (Cit. on p. 31).

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255). doi:10.1109/CVPR.2009.5206848. (Cit. on p. 15)

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning*. Springer series in statistics New York. (Cit. on p. 10).

Gal, Y. & Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ArXiv e-prints*. arXiv: 1506.02142 [stat.ML]. (Cit. on p. 27)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. http://www.deeplearningbook.org. MIT Press. (Cit. on pp. 16, 17).

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *CoRR*, *abs/1706.04599*. arXiv: 1706.04599. (Cit. on pp. 2, 3, 6, 9, 11, 13, 26, 29, 30, 32–38, 41, 42, 44)

He, K., Zhang, X., Ren, S., & Sun, J. (2015a). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. arXiv: 1512.03385. (Cit. on pp. 19–21, 26, 31, 32, 41, 42)

He, K., Zhang, X., Ren, S., & Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, *abs/1502.01852*. arXiv: 1502.01852. (Cit. on p. 31)

Hendrycks, D. & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, *abs/1610.02136*. arXiv: 1610.02136. (Cit. on p. 27)

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. (Cit. on p. 13).

Huang, G., Liu, Z., & Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, *abs/1608.06993*. arXiv: 1608.06993. (Cit. on pp. 22–24, 32, 41)

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. *CoRR*, *abs/1603.09382*. arXiv: 1603.09382. (Cit. on pp. 21, 22, 30–32, 41)

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*. (Cit. on p. 19).

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, *106*(4), 620. (Cit. on p. 13).

Jiang, X., Osl, M., Kim, J., & Ohno-Machado, L. (2011). Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, *19*(2), 263–274. (Cit. on p. 6).

Keren, G., Cummins, N., & Schuller, B. (2018). Calibrated Prediction Intervals for Neural Network Regressors. *ArXiv e-prints*. arXiv: 1803.09546 [stat.ML]. (Cit. on p. 28)

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *Computer vision workshops (iccvw), 2013 ieee international conference on* (pp. 554–561). IEEE. (Cit. on p. 42).

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. (Cit. on pp. 14, 15).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th international conference on neural information processing systems - volume 1* (pp. 1097–1105). NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc. (Cit. on p. 30).

Kuleshov, V. & Ermon, S. (2016). Reliable confidence estimation via online learning. *CoRR*, *abs/1607.03594*. arXiv: 1607.03594. (Cit. on p. 27)

Kuleshov, V. & Liang, P. (2015). Calibrated structured prediction. In *Proceedings of the 28th international conference on neural information processing systems - volume 2* (pp. 3474–3482). NIPS'15. Montreal, Canada: MIT Press. (Cit. on p. 27).

Kull, M., Silva Filho, T., & Flach, P. (2017). Beta calibration: A well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial intelligence and statistics* (pp. 623–631). (Cit. on pp. 2, 3, 6, 11, 13, 33).

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2016). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *ArXiv e-prints*. arXiv: 1612.01474 [stat.ML]. (Cit. on p. 27)

LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, 143–155. (Cit. on p. 17).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. (Cit. on pp. 6, 18, 42).

Li, W. (2018). Learn deep learning with cifar datasets. accessed 05/14/2018, from https://github.com/BIGBALLON/cifar-10-cnn. (Cit. on p. 31)

Majumdar, S. (2018a). Densenet implementation in Keras. accessed 05/14/2018, from https://github.com/titu1994/DenseNet. (Cit. on p. 32)

Majumdar, S. (2018b). Wide residual networks in Keras. accessed 05/14/2018, from https://github.com/titu1994/Wide-Residual-Networks. (Cit. on p. 32)

Naeini, M. P., Cooper, G. F., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Aaai* (pp. 2901–2907). (Cit. on p. 9).

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814). (Cit. on p. 18).

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Nips workshop on deep learning and unsupervised feature learning 2011*. (Cit. on p. 16).

Niculescu-Mizil, A. & Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on machine learning* (pp. 625–632). ICML '05. Bonn, Germany: ACM. doi:10.1145/1102351.1102430. (Cit. on p. 7)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. (Cit. on p. 33).

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., & Hinton, G. E. (2017). Regularizing neural networks by penalizing confident output distributions. *CoRR*, *abs/1701.06548*. arXiv: 1701.06548. (Cit. on p. 27)

Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, *10*(3), 61–74. (Cit. on p. 12).

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*. arXiv: 1409.1556. (Cit. on p. 30)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. (Cit. on p. 27).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112). (Cit. on p. 6).

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., & Perona, P. (2010). *Caltech-UCSD Birds 200* (tech. rep. No. CNS-TR-2010-001). California Institute of Technology. (Cit. on p. 16).

Yu, F. (2017a). Densenet implementation in keras with imagenet pretrained models. accessed 05/14/2018, from https://github.com/flyyufelix/DenseNet-Keras. (Cit. on p. 32)

Yu, F. (2017b). Fine-tune CNN in keras. accessed 05/14/2018, from https://github.com/flyyufelix/cnn_finetune. (Cit. on p. 31)

Zadrozny, B. & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml* (Vol. 1, pp. 609–616). Citeseer. (Cit. on p. 11).

Zadrozny, B. & Elkan, C. (2002a). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 694–699). ACM. (Cit. on p. 12).

Zadrozny, B. & Elkan, C. (2002b). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 694–699). ACM. (Cit. on p. 14).

Zagoruyko, S. & Komodakis, N. (2016). Wide residual networks. *CoRR*, *abs/1605.07146*. arXiv: 1605.07146. (Cit. on pp. 24, 25, 32, 41)

Zhang, Z., Ning, G., Cen, Y., Li, Y., Zhao, Z., Sun, H., & He, Z. (2018). Progressive Neural Networks for Image Classification. *ArXiv e-prints*. arXiv: 1804.09803 [cs.CV]. (Cit. on p. 27)

Zhou, Y. & Chellappa, R. (1988). Computation of optical flow using a neural network. In *Ieee international conference on neural networks* (Vol. 27, pp. 71–78). (Cit. on p. 18).

Appendix

I. Reliability Diagrams

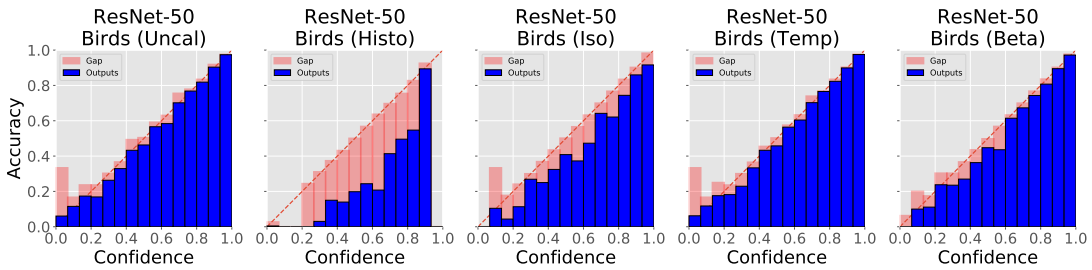


Figure 18. Reliability diagrams (M = 15) for ResNet 50 on Birds dataset for various calibration methods.

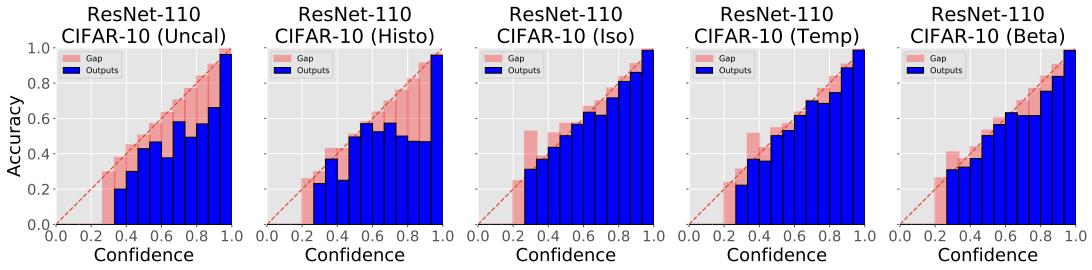


Figure 19. Reliability diagrams (M = 15) for ResNet 110 on CIFAR-10 dataset for various calibration methods.

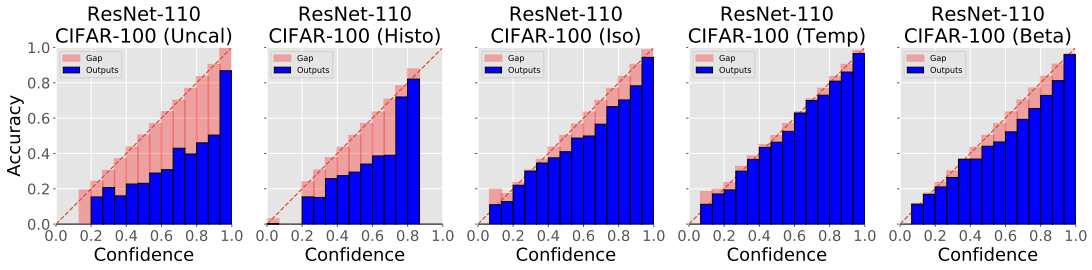


Figure 20. Reliability diagrams (M = 15) for ResNet 110 on CIFAR-100 dataset for various calibration methods.

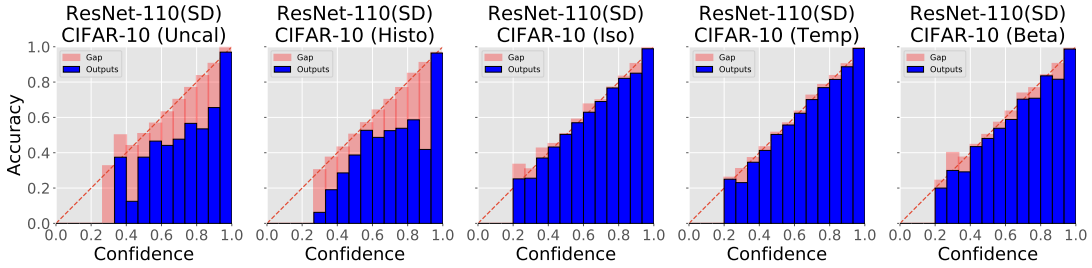


Figure 21. Reliability diagrams (M = 15) for ResNet 110 (SD) on CIFAR-10 dataset for various calibration methods.

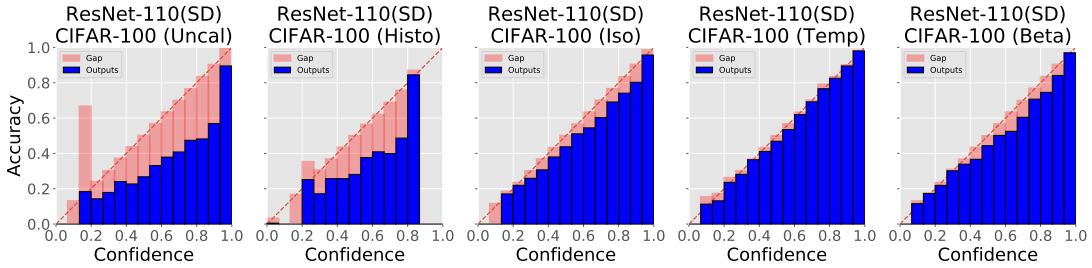


Figure 22. Reliability diagrams (M = 15) for ResNet 110 (SD) on CIFAR-100 dataset for various calibration methods.

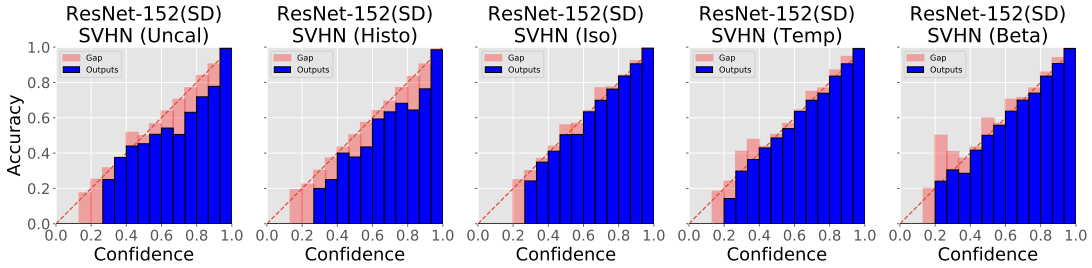


Figure 23. Reliability diagrams (M = 15) for ResNet 152 (SD) on SVHN dataset for various calibration methods.

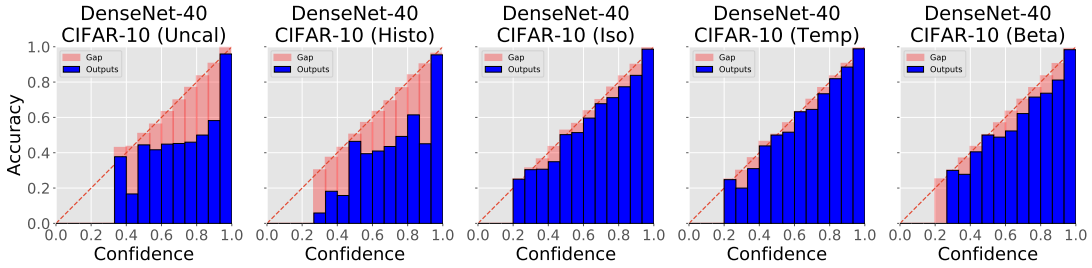


Figure 24. Reliability diagrams (M = 15) for DenseNet 40 on CIFAR-10 dataset for various calibration methods.

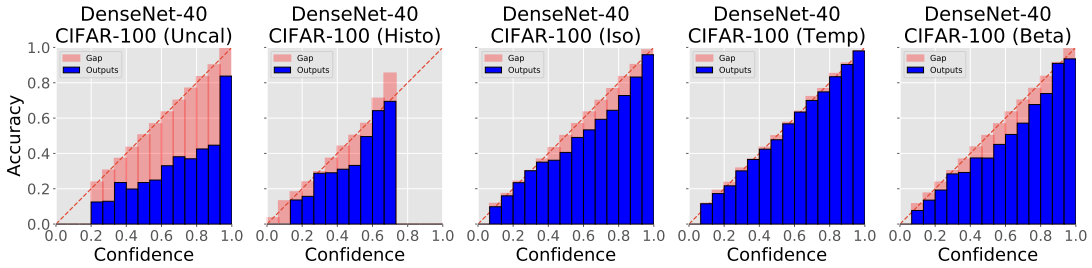


Figure 25. Reliability diagrams (M = 15) for DenseNet 40 on CIFAR-100 dataset for various calibration methods.

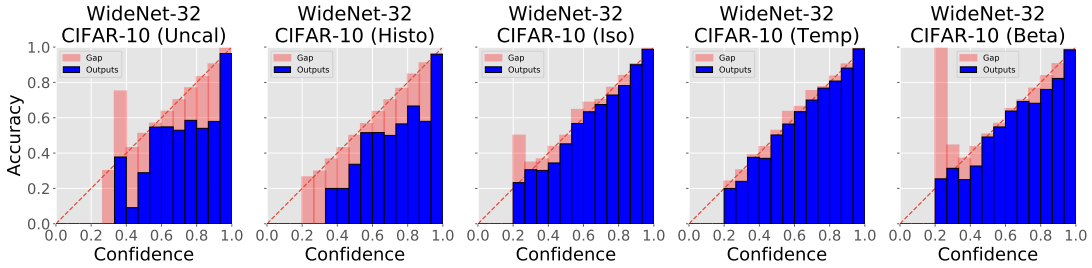


Figure 26. Reliability diagrams (M = 15) for Wide ResNet 32 on CIFAR-10 dataset for various calibration methods.

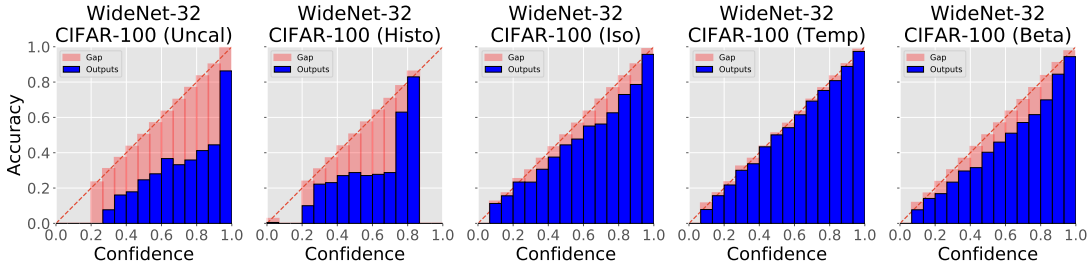


Figure 27. Reliability diagrams (M = 15) for Wide ResNet 32 on CIFAR-100 dataset for various calibration methods.

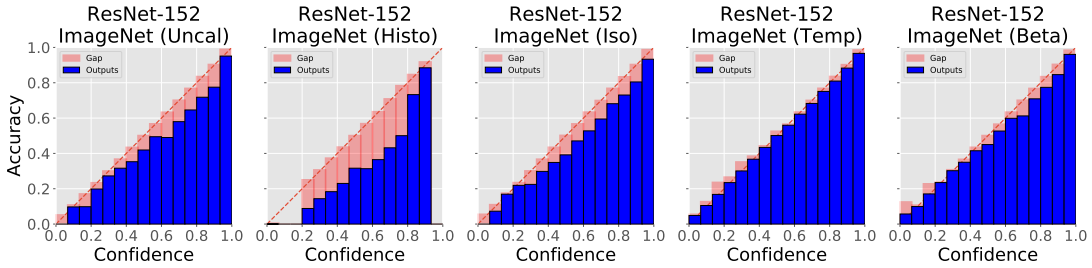


Figure 28. Reliability diagrams (M = 15) for ResNet 152 on ImageNet dataset for various calibration methods.

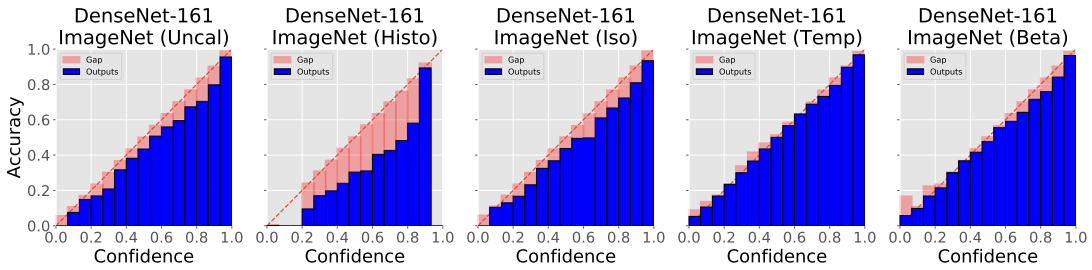


Figure 29. Reliability diagrams (M = 15) for DenseNet 161 on ImageNet dataset for various calibration methods.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Markus Kängsepp**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Calibration of Convolutional Neural Networks

   supervised by Meelis Kull

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 21.05.2018