# MLP Coursework 4: On Overconfidence of Modern Neural Networks

Laszlo Treszkai, and a student who wished to remain anonymous (equal contribution)

## Abstract

In many classification problems, it is helpful or crucial that models produce not only predictions that are correct, but also probability estimates that are well-calibrated, so that predictions are neither overly confident nor insufficiently confident.

Recent work has shown deep neural networks to achieve high accuracy but worse calibration than shallow nets. We replicate these findings, and compare different approaches for improving the calibration of neural networks. As the baseline approach, we consider the calibration of the softmax outputs from a single network; this is compared to *Deep Ensembles*, *MC Dropout*, and *Concrete Dropout*. Through experiments on the CIFAR-100 data set, we find that a large neural network can be significantly over-confident about its predictions. We show on a classification problem that an ensemble of deep networks has better classification accuracy and calibration compared to a single network, and that MC Dropout and Concrete Dropout significantly improves the calibration of a large network.

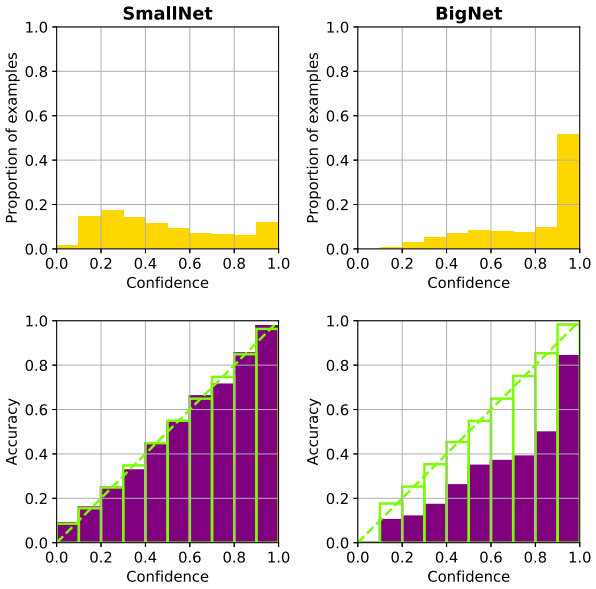


*Figure 1.* Confidence and calibration plots for a small CNN and a much deeper one, each trained on the CIFAR-100 dataset. The bigger network is poorly calibrated on validation data, with most predictions being overly confident. (See Section 5.1 for details.)

## 1. Introduction

Deep neural networks are frequently employed in models for classification tasks. Given a test input, a neural network classifier typically produces not only a predicted class label but also a measure of how confident the network is about its prediction. A standard approach is to take the maximum softmax output of a neural network as confidence.

In the last decade, deep neural networks have achieved ever higher accuracies in various classification tasks. However, classification accuracy is not the only important metric in many situations: it can also be desirable or crucial that the classifier is *well-calibrated*, so that its predictions are neither overly confident nor insufficiently confident.

Calibration matters wherever overly confident yet incorrect predictions can be harmful or offensive, such as in medical diagnosis and in language models handling offensive text (Amodei et al., 2016). In high-stakes applications such as autonomous driving systems, incorrect predictions made with high confidence can even be fatal.

Despite its importance, model calibration has not been studied as extensively in machine learning literature as metrics like classification accuracy (Naeini et al., 2015). It has been noted that calibration is an orthogonal concern to accuracy: the predictions of a neural network may be highly accurate yet poorly calibrated, and vice versa.

Much of our project was motivated by a recent paper named *On Calibration of Modern Neural Networks* (Guo et al., 2017), which showed that modern neural networks can be very poorly calibrated. Through extensive experiments on CIFAR-100 and other datasets, the paper investigated how factors such as network depth and weight decay affect model calibration. A key result was that a 110-layer ResNet (He et al., 2016) can exhibit much poorer calibration than a 5-layer LeNet (LeCun et al., 1998) on CIFAR-100.

It has been shown that model calibration can be significantly improved through *Deep Ensembles* (Lakshminarayanan et al., 2017), where an ensemble consists of networks with the same architecture but trained with different random weight initialisations and input ordering.

A fruitful line of research from the last few years stems from an interpretation of Dropout (Srivastava et al., 2014) as approximate inference in a Bayesian neural network (Gal, 2016). Following this interpretation, *MC Dropout* uses Dropout at test time to yield stochastic forward passes. A novel variant called *Concrete Dropout* (Gal et al., 2017) is designed to improve performance and calibration.

For two network architectures, we compare the calibration of 1) a single network with Dropout, 2) the same network with Dropout enabled at test time, 3) a Deep Ensemble, and 4) a single network with Concrete Dropout. The next section gives a technical definition of model calibration, along with different measures to analyse miscalibration. Section 3 describes the methods we use for obtaining confidence estimates from our neural networks. Section 4 gives a brief overview of the used data set and neural network architectures. Section 5 details our main experiments and Section 6 goes in greater detail about our observations. Finally, the paper closes with our conclusions and avenues for further work in Sections 7 and 8.

## 2. Confidence and calibration

Say we have a classifier which, given any input $\mathbf{x}$, outputs a vector $\hat{\mathbf{p}}$ such that $\hat{p}_k$ is an estimate of the probability that $\mathbf{x}$ belongs to class $k$. (The softmax output of a neural network gives such a vector $\hat{\mathbf{p}}$: see Section 3.1.) We shall call $\hat{p}_k$ the *confidence* of the classifier about the true class of $\mathbf{x}$ being $k$.

We restrict our attention to predictions that are in the form of single class labels. In the situation above, the classifier would predict that $\mathbf{x}$ belongs to class $\hat{k} := \operatorname{argmax}_k (\hat{p}_k)$ with confidence $\hat{p} := \hat{p}_{\hat{k}} = \max_k \hat{p}_k$.

Ideally, the model confidence $\hat{p}$ should reflect the "true" probability $p$ that $\mathbf{x}$ belongs to class $\hat{k}$, so if a model makes predictions with about 90% confidence on some inputs, then about 90% of predicted class labels should be correct. If this is the case, then the model is said to be *well-calibrated*.

### 2.1. Measuring calibration

To see whether a model is well-calibrated, we can make a *confidence plot* and a *calibration plot*. From these plots, we can compute the *expected calibration error* (ECE). This section gives a brief review of these concepts, following the notation in (Guo et al., 2017).

Given $N$ inputs $\{\mathbf{x}^{(i)}\}_{i=1}^N$ where the true class of $\mathbf{x}^{(i)}$ is $k^{(i)}$, suppose that a model predicts that $\mathbf{x}^{(i)}$ belongs to class $\hat{k}^{(i)}$ with confidence $\hat{p}^{(i)}$.

First, the indices $i$ are placed into $L$ equally sized bins $B_\ell$ according to the value of $\hat{p}^{(i)}$. We took $L = 10$ which gives the bins $[0, 0.1), \ldots, [0.8, 0.9), [0.9, 1]$.

We visualise the bins with **yellow bars** in a *confidence plot*. There are two examples on the top row of Figure 1.

For every bin $B_\ell$, we define the *average accuracy* $\operatorname{acc}(B_\ell)$ and *average confidence* $\operatorname{conf}(B_\ell)$ in the natural manner:

$$\operatorname{acc}(B_\ell) := \frac{1}{|B_\ell|} \sum_{i \in B_\ell} \mathbb{1}[\hat{k}^{(i)} = k^{(i)}].$$

$$\operatorname{conf}(B_\ell) := \frac{1}{|B_\ell|} \sum_{i \in B_\ell} \hat{p}^{(i)}.$$

For a perfectly calibrated model, every bin $B_\ell$ would have $\operatorname{acc}(B_\ell) = \operatorname{conf}(B_\ell)$. The difference $\operatorname{acc}(B_\ell) - \operatorname{conf}(B_\ell)$ measures a *calibration gap* for bin $B_\ell$, with negative values indicating over-confidence and positive values indicating under-confidence.

We plot the average accuracy and average confidence in a *calibration plot*, with **purple bars** showing $\operatorname{conf}(B_\ell)$ and **hollow green bars** showing $\operatorname{acc}(B_\ell)$ for each bin $B_\ell$. The bottom row of Figure 1 gives two examples. The bottom right plot has $\operatorname{acc}(B_\ell) < \operatorname{conf}(B_\ell)$ for every bin $B_\ell$, which indicates that the predictions are over-confident.

For a scalar measure of calibration, we shall consider the *expected calibration error* (ECE), defined as the weighted average of the absolute calibration gaps:

$$\mathrm{ECE} = \sum_{\ell=1}^{L} \frac{|B_\ell|}{N} |\operatorname{acc}(B_\ell) - \operatorname{conf}(B_\ell)|.$$

where $N$ is the total number of inputs $\mathbf{x}^{(i)}$. Note that the ECE cannot be computed from a calibration plot by itself: we need the bin sizes as shown in a confidence plot.

Guo et al. (2017) considers a model well-calibrated if its ECE is below 0.01.

## 3. Statistical models

This section describes three statistical models that make use of neural networks to perform classification: the standard model, a uniform mixture model, and a Bayesian model.

In practice, these models lead to different approaches for obtaining predictions:

- Take the softmax output from a single neural network.
- **Deep Ensembles**: compute the average of outputs from several networks with identical architecture.
- **MC Dropout**: enable Dropout at test time to obtain stochastic forward passes of test inputs.
- **Concrete Dropout**: a novel variant of MC Dropout.

Although understanding these models is not essential for experiments, we discuss them below so that a more concrete discussion of our results is possible later.

### 3.1. Softmax output from a single network

In neural networks that are designed for classification tasks, the final layer is usually a softmax layer, whose output $\hat{\mathbf{p}}$ has non-negative elements which sum to 1. It is standard to treat the softmax output as defining a probability distribution over class labels, as follows:

Let us denote the network weights by a vector $\mathbf{w}$. For any network input $\mathbf{x}$, the softmax output $\hat{\mathbf{p}}(\mathbf{x}; \mathbf{w})$ is interpreted as specifying a probability distribution over the possible class labels of input $\mathbf{x}$:

$$P[k \mid \mathbf{x}, \mathbf{w}] = \hat{p}_k(\mathbf{x}; \mathbf{w}).$$

This interpretation motivates using cross-entropy as the loss function, since cross-entropy is the negative log-likelihood under this model (Blundell et al., 2015).

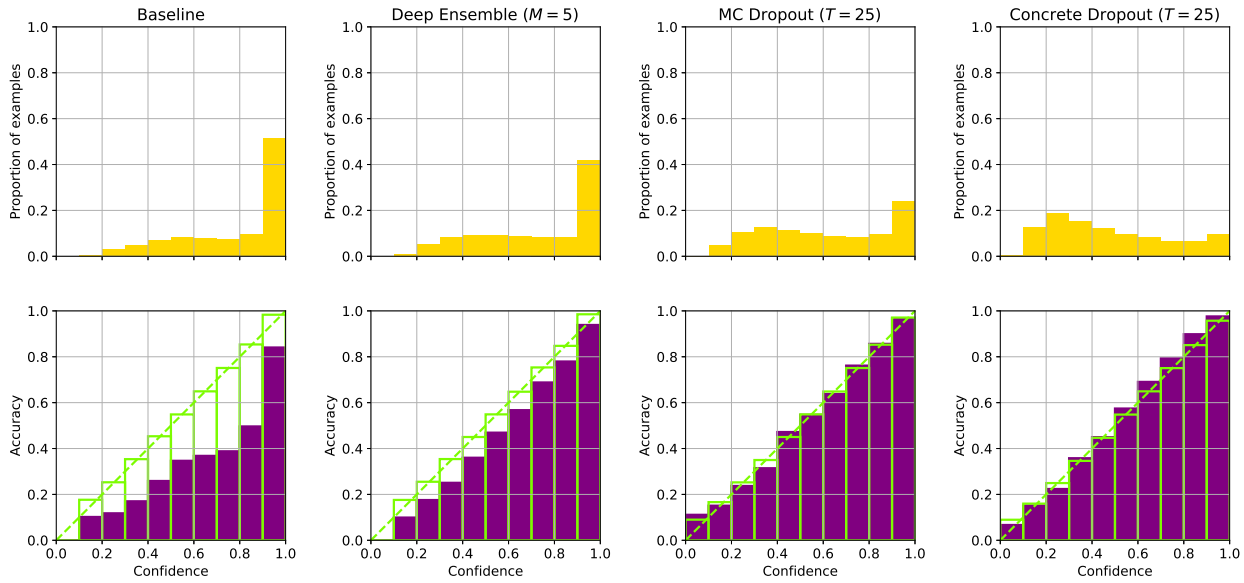


*Figure 2.* Confidence and calibration plots for **BigNet**. For each approach, the results are from a fixed random seed.

We think of the softmax output $\hat{p}_k$ as the neural network's *confidence* that the input $\mathbf{x}$ belongs to class $k$.[1]

### 3.2. Deep Ensembles

It has long been known that ensembling models improves predictive performance (Dietterich, 2000). However, it is not clear whether an ensemble of neural networks produces probability estimates that are better-calibrated compared to a single network. In a recent paper, Lakshminarayanan et al. (2017) showed that this is indeed the case.

A *Deep Ensemble* (Lakshminarayanan et al., 2017) is an ensemble of neural networks with the same architecture and hyperparameters but trained with different random weight initialisations and input ordering. Such networks can be trained independently in parallel.

Given an input $\mathbf{x}$, the $m^{\text{th}}$ neural network gives a softmax output $\hat{\mathbf{p}}^{(m)}(\mathbf{x}; \mathbf{w}^{(m)})$ which specifies a predictive distribution $P^{(m)}(k \mid \mathbf{x}, \mathbf{w}^{(m)})$ over the classes to which $\mathbf{x}$ might belong. The ensemble is treated as a uniformly-weighted mixture model, so that the overall predictive distribution is given by

$$P[k \mid \mathbf{x}, \mathbf{w}] = \frac{1}{M} \sum_{m=1}^{M} P^{(m)}[k \mid \mathbf{x}, \mathbf{w}^{(m)}] = \frac{1}{M} \sum_{m=1}^{M} \hat{p}_k^{(m)}$$

where the vector $\mathbf{w} = \{\mathbf{w}^{(m)}\}_{m=1}^{M}$ encodes the parameters of all the neural networks.

In our experiments, we use $M = 5$ different random seeds to obtain a Deep Ensemble. (The random seed affects both the weight initialisation and the shuffling of training data.)

[1] There seems to be disagreement over what the phrase "model confidence" should mean. In his thesis, Gal (2016) argues that it is erroneous to interpret the softmax output as "confidence". However, this is not the majority view held by authors including Lakshminarayanan et al. (2017, §3.5) and Guo et al. (2017, §4.2).

### 3.3. MC Dropout

Dropout (Srivastava et al., 2014) is a popular regularisation technique for neural networks. In a standard Dropout layer, each input vector is multiplied elementwise with a binary vector $\mathbf{z}$ with $z_i \sim \text{Bernoulli}(p)$ for some fixed $p \in [0, 1]$, effectively "switching off" neurons in the previous layer with probability $p$. At test time, it is usual to turn off Dropout and scale the outputs of any Dropout layer by $1 - p$ to account for the higher number of active neurons.

Recent work has shown that training a neural network with Dropout can in fact be interpreted as approximate inference in a Bayesian interpretation of the network (Gal, 2016; Gal & Ghahramani, 2016). More precisely, tuning the network weights to minimise the loss function can be understood as optimising the parameters $\boldsymbol{\theta}$ of a variational approximation $q_{\boldsymbol{\theta}}(\mathbf{w})$ to the true posterior $p(\mathbf{w} \mid \mathcal{D})$ over weights, for some prior distribution $p(\mathbf{w})$.

Following the Bayesian interpretation above, we can sample from the predictive distribution through *test-time Dropout*. Using Dropout at test time gives us *stochastic forward passes* of test inputs. The softmax output $\hat{\mathbf{p}}$ from such a stochastic forward pass is then a single sample from the predictive distribution: $P[k \mid \mathbf{x}, \mathbf{w}] \approx \hat{\mathbf{p}}_k$.

To better approximate the predictive distribution $P[k \mid \mathbf{x}, \mathbf{w}]$ for a given input $\mathbf{x}$, we can averaging the softmax outputs $\{\hat{\mathbf{p}}^{(t)}\}_{t=1}^{T}$ from multiple stochastic forward passes:

$$P[k \mid \mathbf{x}, \mathbf{w}] \approx \frac{1}{T} \sum_{t=1}^{T} \hat{p}_k^{(t)}.$$

Gal (2016) refers to this procedure as *MC Dropout*.

### 3.4. Concrete Dropout

According to Gal (2016), it is necessary to perform a grid search over dropout probabilities for MC Dropout to return well-calibrated probability estimates. Such a grid search is infeasible for deep models with multiple Dropout layers and impossible when the amount of training data is variable (as is the case in reinforcement learning systems). As a solution to both of these problems, Gal et al. (2017) proposed a continuous relaxation of Dropout called *Concrete Dropout*.

In standard Dropout, the dropout vector $\mathbf{z}$ is binary, so the network loss function is not differentiable w.r.t. the dropout probability $p$. As a result, it is impossible to optimise $p$ with backpropagation. In Concrete Dropout, the elements of the dropout vector $\mathbf{z}$ are instead sampled from the continuous *Concrete distribution* (Maddison et al., 2016):

$$\tilde{z}_i = \text{sigmoid}\left(\frac{\log p - \log(1-p) + \log u - \log(1-u)}{t}\right)$$

with $u \sim \text{Uniform}(0, 1)$. Since the relation between $\tilde{\mathbf{z}}$ and $u$ is differentiable w.r.t $p$, this allows the dropout probabilities to be optimised as parameters of the neural network.

At test time, the outputs of multiple stochastic forward passes are averaged, just like with MC Dropout.

## 4. Methodology

Much like Guo et al. (2017), our main goal is to investigate the calibration of neural networks trained in accordance to standard practice in machine learning, with no regard given to calibration until test time. With this in mind, we chose the following training procedure and network architectures.

### 4.1. Data set and training

To allow comparison with the results by Guo et al. (2017), we focus on the CIFAR-100 data set (Krizhevsky, 2009) for evaluating our neural networks. CIFAR-100 consists of images belonging to 100 balanced classes. Each image has $32 \times 32$ colour pixels, each encoded with 3 values (red, green, blue).

We applied the default training / testing split to obtain a training set of size 50,000 and a validation set of 10,000. Each class has 500 training and 100 validation images.

Each of our neural networks is trained to minimise the standard categorical cross-entropy loss (i.e. the NLL). Model checkpoints are created throughout training, and only the weights which achieve the lowest *validation loss* are kept. Note that *calibration is not part of the loss function or the early stopping criterion*, and ECE is not directly correlated with NLL (G48, 2018; Guo et al., 2017). We evaluate classification accuracy and calibration only during analysis, when all training is complete.

The smaller networks were trained with RMSprop (Tieleman & Hinton, 2012) with a learning rate of 0.0002, while the larger networks were trained with stochastic gradient descent with a fixed learning rate of 0.01.

### 4.2. Network architectures

Our experiments centred on two convolutional neural network (CNN) architectures, with different numbers of layers. Each of our networks follows a layer pattern that is standard for CNNs (Karpathy et al., 2017).

We mimic the setup in Guo et al. (2017) and compare a relatively simple and shallow CNN, which we hereafter call *SmallNet*, to another CNN that is considerably deeper and more complex in size, which we refer to as *BigNet*. Our smaller network is based on a standard CNN pattern, while the larger network is taken from previous work by Liu & Deng (2015).

All of our convolutional layers have kernel size $3 \times 3$ with strides $1 \times 1$. Unless otherwise specified, zero-padding is then applied so that each output channel has the same shape as the input channels. Any max-pooling layer between convolutional layers has a pool size of $2 \times 2$.

Every dense or convolutional layer is followed by a ReLU activation layer, except for the output layer, which is always a dense layer followed by softmax. Before this output layer, there is always a dense layer with 512 units followed by a Dropout layer with fixed Dropout rate 0.5.

**SmallNet**: Our smaller network has 4 convolutional layers. After the input layer, there are two convolutional layers with 32 output filters, the first of which has zero-padding, followed by a max-pooling layer and a fixed Dropout layer. Next there is a similar block which has 64 output filters in each of two convolutional layers but is otherwise identical. An example script provided by Keras states that this network achieves 79% accuracy on the CIFAR-10 dataset, whose distribution is similar to that of CIFAR-100.

**BigNet**: Our larger network has 13 convolutional layers. The network is a modified version of VGG-16 (Simonyan & Zisserman, 2014) that Liu & Deng (2015) tuned to achieve good performance on CIFAR-10. We took the version which has Batch Normalization and fixed Dropout layers as this achieved the highest accuracy in Liu & Deng (2015).

The larger network has 5 groups of convolutional layers, with each group specified by 1) the number of convolutional layers, 2) the number of filters in each convolutional layer, and 3) a fixed Dropout rate. Max-pooling is applied after each group.

Each group has either 2 or 3 convolutional layers, each of which is followed by a Batch Normalization layer. Each Batch Normalization layer before the final one is followed by a fixed Dropout layer.

The groups in our network have the same specification as chosen by Liu & Deng (2015):

- 2 conv layers / 64 filters / dropout rate 0.3.
- 2 conv layers / 128 filters / dropout rate 0.4.
- 3 conv layers / 256 filters / dropout rate 0.4.
- 3 conv layers / 512 filters / dropout rate 0.4.
- 3 conv layers / 512 filters / dropout rate 0.4.

### 4.3. Implementation

Our experiments were implemented with Keras.

To train a Deep Ensemble, we simply set different random seeds before initialising the network. The random seed affects both the random weight initialisation and input ordering, so our procedure is an implementation of the algorithm given by Lakshminarayanan et al. (2017).

For MC Dropout, the networks are trained as with standard Dropout, but at test time, we set `training=True` in calls to every `Dropout` layer to obtain stochastic forward passes.[2]

For Concrete Dropout with SmallNet, every convolutional and dense layer (except for the output layer) was wrapped in a Concrete Dropout wrapper. With BigNet, we wrapped only those layers that preceded a dropout layer. For Concrete Dropout, we take every layer that is preceded by a fixed Dropout layer and remove the Dropout layer. Then, we take the reference implementation of Concrete Dropout as a layer wrapper in Keras, given by Gal et al. (2017), and apply this to the remaining layer.

## 5. Experiments

| | Classification accuracy | | | |
|---|---|---|---|---|
| | Baseline | DE | MC | CD |
| SmallNet | 50.2% | 52.5% | 47.4% | 36.3% |
| BigNet | 59.7% | 67.1% | 61.6% | 48.8% |
| | **Mean confidence** | | | |
| SmallNet | 52.3% | 45.2% | 38.2% | 38.6% |
| BigNet | 79.7% | 73.8% | 62.0% | 47.8% |
| | **ECE** | | | |
| SmallNet | 0.027 | 0.073 | 0.092 | 0.026 |
| BigNet | 0.200 | 0.067 | 0.012 | 0.022 |

*Table 1.* Metrics for each of Baseline, Deep Ensembles ($M = 5$), MC Dropout ($T = 25$ passes), Concrete Dropout ($T = 25$ passes) on CIFAR-100. For each approach, the results are from one fixed random seed. (Corresponding confidence and calibration plots for BigNet are shown in Figure 2.)

### 5.1. Baselines

As a baseline, we trained each of SmallNet and BigNet on CIFAR-100 with 5 random seeds.

The networks achieved classification accuracies of 48.1% and 62.0% respectively, where each accuracy figure is the average from different random seeds.

[2] Another alternative would have been to directly train a model with test-time dropout enabled, but then one needs multiple forward passes with the validation set for a precise early stopping criterion.

More important for our purposes are the calibration metrics: SmallNet achieved an ECE of 0.013 (±0.007 std. dev.), while BigNet had 0.234 (±0.020). For a particular instance of each architecture, the confidence and calibration plots are shown in Figure 1.

From the confidence plots, we can see that BigNet is much more confident than SmallNet on validation data. Indeed, SmallNet had a mean confidence of 48.7% (+0.6% of its accuracy) while BigNet had 85.3% (+23.3%). Along with ECE values, these figure suggest that BigNet is significantly over-confident on the validation set. The calibration plot for BigNet also demonstrates that BigNet is less accurate than its confidence might suggest.

Our finding mirrors a key result from Guo et al. (2017) who observed a 110-layer ResNet (He et al., 2016) can exhibit much poorer calibration than a 5-layer LeNet (LeCun et al., 1998) on CIFAR-100. The same paper found that increasing the network width/depth and applying Batch Normalization lead to worse calibration. These factors do distinguish BigNet from SmallNet, so it comes as no surprise to us that BigNet is much more poorly calibrated.

### 5.2. Deep Ensembles

For each network architecture, we created a Deep Ensemble by simply grouping together the 5 networks trained from different random seeds. As described in Section 3.2, the output of a Deep Ensemble is simply the uniform average of the softmax outputs from the 5 networks.

The resulting classification accuracies were higher than the baselines: 52.5% for SmallNet and 67.1% for BigNet. Again, we are not surprised as ensembling models has long been known to improve accuracy (Dietterich, 2000).

The effect of ensembling on calibration is different for the two architectures. As we can see in Table 1, the ECE actually increased for SmallNet (0.027 to 0.073), while the ECE of BigNet dropped dramatically (0.200 to 0.067). We analyse the situation for BigNet further in Section 6.2.

### 5.3. MC Dropout

Using Dropout at test time, we performed $T = 25$ stochastic forward passes for an instance of each architecture.

The figures in Table 1 show that the accuracy remained roughly the same for each of SmallNet and BigNet while the mean confidence decreased: from 52.3% to 38.2% for SmallNet and from 79.7% to 62.0% for BigNet.

Curiously, the impact on calibration is notably different for the two architectures: the ECE of SmallNet rose to 0.092 from 0.027, while the ECE of BigNet dropped dramatically to 0.200 from 0.012. We will attempt to see why this is the case in Section 6.3.

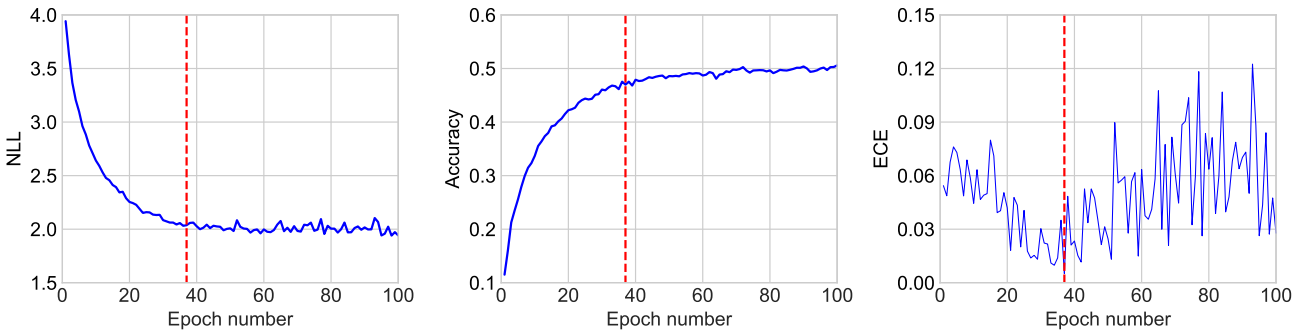


*Figure 3.* Learning curves for our baseline SmallNet trained on CIFAR-100. The dashed red line indicates the early stopping at epoch 37.

### 5.4. Concrete Dropout

Concrete Dropout has a lengthscale parameter $\ell$, which affects the regularization of the weights and of the dropout probabilities. Following the work of Gal et al. (2017), we trained multiple networks with $\ell$ in $\{10^{-2}, 10^{-3}, \ldots, 10^{-9}\}$ and intended to select the one that achieves the highest likelihood. However, the difference between performance was negligible: every model reached an NLL between 1.38 and 1.41, so we chose to perform experiments with the default lengthscale of $\ell = 10^{-4}$.

For testing, we performed $T = 25$ stochastic forward passes just as for MC Dropout. The accuracies turned out to be noticeably lower than the baselines: from 50.2% to 36.3% for SmallNet and from 59.7% to 48.8% for BigNet. However, Concrete Dropout did improve the calibration of BigNet significantly, with the ECE dropping to 0.022 from 0.200.

For each network, similar results (omitted here) were obtained from different random seeds and prior lengthscales. To investigate why the metrics are so different compared to those from other approaches, we could perhaps look at the final dropout probabilities from Concrete Dropout and track metrics such as NLL over training. We leave the analysis of the Concrete Dropout results as possible future work.

## 6. Analysis

### 6.1. Evolution of metrics over training

As Figure 3 shows, the baseline SmallNet did not overfit with RMSprop within 100 epochs. The validation accuracy and NLL actually continued to generally improve after the point of early stopping, but the gains were not significant.

While the curves for accuracy and NLL are relatively smooth and predictable, the ECE curve show significant fluctuations. After the point of early stopping, the ECE fluctuations were especially wild, but it can be observed that the ECE generally increased (from a minimum of 0.01 to 0.07 ± 0.05). Thus, as training progressed past a certain point, the network learned to increase its accuracy at the expense of well-calibrated predictions. (We made the same observation in our interim report on the EMNIST By-Class dataset with a fully connected network, where the calibration decrease near the end of the training was even clearer (G48, 2018).)

Guo et al. (2017, Section 3) made a similar observation on CIFAR-100 with a 110-layer ResNet, but using NLL an indirect measure of calibration. Our previous results (G48, 2018) and Figure 3 suggest that NLL is not necessarily a good measure of calibration. Depending on the purpose, ECE might be a better indicator. (Note that ECE assigns equal penalty to an accuracy of 0.5 for confidence 0.65 and an accuracy of 0.8 for confidence 0.95, whereas NLL does not.) This suggests that the use of NLL as a measure of calibration by Lakshminarayanan et al. (2017) might be incorrect.

### 6.2. Calibration of Deep Ensembles

For BigNet, we observe that the confidence and calibration plots of the baseline and the Deep Ensemble look similar in Figure 2. The main difference appears to be the increased heights of the bins in the calibration plots, with confidences remaining roughly the same. Table 1 shows that the mean confidence indeed did not change much. (The behaviour of SmallNet is similar.) However, the confidence of many predictions actually changed when we ensembled the 5 networks: Figure 5 shows the change in confidence compared to the baseline for the examples in the validation set.

It is unclear why ensembling had vastly different effects on BigNet and SmallNet. A possible approach to investigating this is to vary the ensemble size and see if any relationships can be observed. We leave this as possible future work.

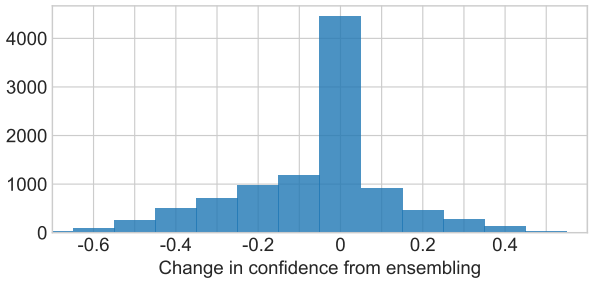


*Figure 5.* Histogram of confidence changes as a result of ensembling, in BigNet.

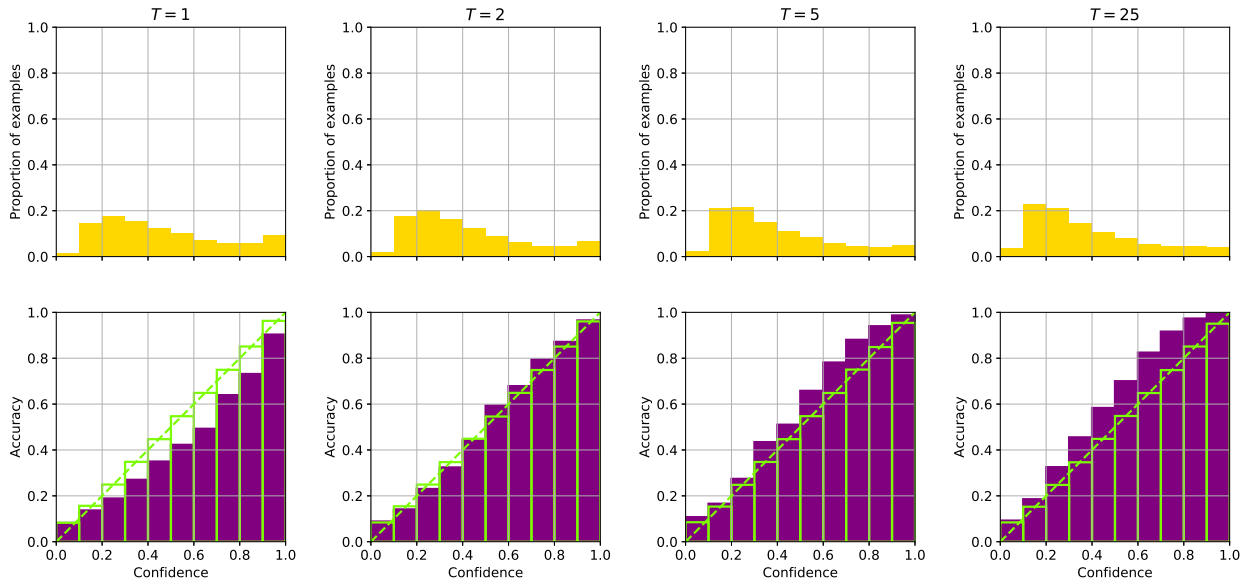


*Figure 4.* Calibration plots for different number of stochastic forward passes with MC Dropout on SmallNet.

### 6.3. Varying the number of MC Dropout passes

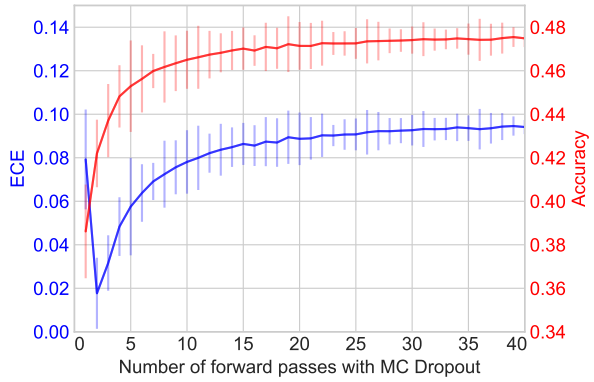


*Figure 6.* ECE and accuracy of **SmallNet** with **MC Dropout** for different numbers of stochastic forward passes, for one random seed. Error bars represent 2 standard deviations of the values from 10 stochastic forward passes of the whole validation set.

For each of SmallNet and BigNet, the accuracy improves steadily as more forward passes are averaged; Figure 6 shows this relationship for one instance of SmallNet. A probable explanation is that increasing the number of stochastic forward passes yields a closer approximation of the predictive distribution (see Bayesian interpretation of Dropout, Section 3.3). For SmallNet, the accuracy with a single forward pass is 38.8% ± 1.1%, whereas with 25 forward passes, it is 47.3% ± 0.3%.

With one forward pass on SmallNet, the mean ECE is $0.078 \pm 0.013$. Best calibration is achieved with $T = 2$ at an ECE of $0.020 \pm 0.010$, i.e. it is reasonably well calibrated. As the number of forward passes increases, calibration error grows steadily, and beyond 25 passes the model has an ECE above 0.09. Such behaviour was not specific to the random seed used: the results from all 5 different seeds were qualitatively similar. Figure 4 gives details on the results for $T \in \{1, 2, 5, 25\}$. From the confidence plots (top row) we can observe that with more forward passes averaged, the average predictions become less confident (i.e. data points in the bars move left on average), while as observed above, accuracy increases (i.e. the heights of the confidence bars increases). This causes the confidence bars to rise just to the ideal diagonal line with $T = 2$, but move beyond the diagonal for $T > 2$, making the model underconfident. As in the calculation of ECE we take the absolute difference of bar heights, this results in the shape of the blue curve in Figure 6.

With BigNet, calibration error decreases monotonically for more forward passes (not shown in the report). Compared with the baseline, this is partly because the mean confidence decreases while the overall accuracy increases (Figure 2 and Table 1).

## 7. Conclusions

Our experiments gave empirical evidence that convolutional neural networks of modern scale can exhibit much poorer calibration than simpler networks, corroborating previous work by Guo et al. (2017). We found that NLL and ECE don't always follow the same trajectory, despite the use of NLL as a measure of calibration by some authors. We found that ensembling networks trained from different random seeds increases classification accuracy, as shown by Lakshminarayanan et al. (2017). Ensembling did not improve calibration for a small network, but it improved calibration significantly for a large convnet, while still giving highly confident predictions that were accurate. We discovered that performing Dropout at test time and averaging the outputs of multiple stochastic forward passes can be an effective way to significantly improve calibration, but that the optimal number of forward passes can depend on the network architecture.

## 8. Further work

Apart from the lines of enquiry suggested in Sections 5-6, there are many others that can be pursued. One is whether novel approaches designed explicitly to improve calibration, such as *Platt scaling* (Guo et al., 2017), are more effective in setups similar to ours. Another possible extension is to test other approaches that involve stochastic forward passes. While MC Dropout and Concrete Dropout are relatively easy to test, there are other notable techniques arising from Bayesian interpretations of neural networks: stochastic gradient variational Bayes (Kingma & Welling, 2013) and Variational Dropout (Kingma et al., 2015) are some examples.

## References


Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul F., Schulman, John, and Mané, Dan. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural network. In Bach, Francis and Blei, David (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/blundell15.html.

Dietterich, Thomas G. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pp. 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.

G48. MLP Coursework 3: Project interim report, 2018.

Gal, Yarin. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gal, Yarin, Hron, Jiri, and Kendall, Alex. Concrete Dropout. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

Guo, Chuan, Pleiss, Geoff, Sun, Yu, and Weinberger, Kilian Q. On calibration of modern neural networks. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/guo17a.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Karpathy, Andrej et al. Lecture notes for CS231n: Convolutional neural networks for visual recognition, 2017. URL https://cs231n.github.io/convolutional-networks/#layerpat.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2575–2583. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Lakshminarayanan, Balaji, Pritzel, Alexander, and Blundell, Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Liu, Shuying and Deng, Weihong. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, Nov 2015. doi: 10.1109/ACPR.2015.7486599.

Maddison, Chris J., Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712, 2016.

Naeini, Mahdi Pakdaman, Cooper, Gregory F., and Hauskrecht, Milos. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2901–2907. AAAI Press, 2015. ISBN 0-262-51129-0.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.