BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

ASSESSMENT OF UNCERTAINTY OF NEURAL NET PREDICTIONS IN THE TASKS OF CLASSIFICATION, DETECTION AND SEGMENTATION

HODNOCENÍ NEURČITOSTI PREDIKCÍ NEURONOVÝCH SÍTÍ V ÚLOHÁCH KLASIFIKACE, DETEKCE A SEGMENTACE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JIŘÍ VLASÁK**
AUTOR PRÁCE

SUPERVISOR **prof. Ing. ADAM HEROUT, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Vlasák Jiří**
Program: Informační technologie
Název: **Hodnocení neurčitosti predikcí neuronových sítí v úlohách klasifikace, detekce a segmentace**
Assessment of Uncertainty of Neural Net Predictions in the Tasks of Classification, Detection and Segmentation
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s problematikou učení a nasazení neuronových sítí v oblasti počítačového vidění a zpracování obrazu.
2. Prostudujte problematiku hodnocení věrohodnosti predikcí neuronových sítí.
3. Vyhledejte a popište vhodné datové sady a hotové neuronové sítě pro úlohy klasifikace, detekce a segmentace obrazů.
4. Experimentujte s metodami hodnocení věrohodnosti predikcí NN na různých neuronových sítích, nad různými datovými sadami.
5. Zhodnoťte dostupné metody hodnocení věrohodnosti predikcí a formulujte doporučení pro konkrétní aplikace.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011
- Yarin Gal, Zoubin Ghahramani: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, ICML 2016
- Alex Kendall, Yarin Gal: What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?, EML 2018, https://arxiv.org/abs/1703.04977
- Loquercio et al.: A General Framework for Uncertainty Estimation in Deep Learning, IEEE Robotics and Automation Letters, 2020
- Jungo A, Reyes M: Assessing Reliability and Challenges of Uncertainty Estimations for Medical Image Segmentation, MICCAI 2019
- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 13. dubna 2022

Abstract

This work focuses on comparing three widely used methods for improving uncertainty estimations: Deep Ensembles, Monte Carlo Dropout, and Temperature Scaling. These methods are applied to six computer vision models that are pretrained as well as trained from scratch. The models are then evaluated on computer vision datasets for classification, semantic segmentation, and object detection using a wide range of metrics. The models are also evaluated on distorted versions of these datasets to measure their performance on out-of-distribution data.

These modified models achieve promising results. Ensembles outperform the other models by as high as 70 % in accuracy and 0.2 in IOU on the distorted MedSeg COVID-19 segmentation dataset while also outperforming the other models on the CIFAR-100 and FMNIST datasets.

Abstrakt

Tato práce se zaměřuje na porovnání tří široce používaných metod pro zlepšení odhadů neurčitosti: hlubokých ansámlů, monte carlo dropout a temperature scaling. Tyto metody jsou aplikovány na šest modelů pro počítačové vidění, mezi nimiž jsou předtrénované modely i modely trénované od nuly. Tyto modely jsou hodnoceny na datasetech počítačového vidění pro úlohy klasifikace, sémantické segmentace a detekce objektů, při použití široké škály metrik. Modely jsou rovněž evaluovány na transformovaných datasetech, kvůli jejich ohodnocení na datech mimo trénovací distribuci.

Tyto modifikované modely dosahují slibných výsledků. Ansámbly překonávají ostatní modely až o 70 % v přesnosti a o 0.2 v IOU na transformovaném segmentačním datasetu MedSeg COVID-19 a zároveň překonávají ostatní modely na datasetech CIFAR-100 a FMNIST.

Keywords

monte carlo dropout, deep ensembles, temperature scaling, classification, semantic segmentation, object detection, dataset shift, model calibration, FMNIST, CIFAR-100, PASCAL-VOC

Klíčová slova

monte carlo dropout, hluboké ansámbly, temperature scaling, klasifikace, sémantická segmentace, detekce objektů, posunutí datasetu, kalibrace modelů, FMNIST, CIFAR-100, PASCAL-VOC

Reference

VLASÁK, Jiří. *Assessment of Uncertainty of Neural Net Predictions in the Tasks of Classification, Detection and Segmentation.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Adam Herout, Ph.D.

Rozšířený abstrakt

Moderní architektury hlubokých konvolučních neurálních sítí dosahují výborných výsledků v mnoha úlohách počítačového vidění. Bohužel, i přes dobrou úspěšnost těchto predikcí, mohou mít tyto modely problém s kvalitním odhadem neurčitosti těchto predikcí. Predikce těchto modelů bývají totiž často příliš sebevědomé, a to i pro rozdílná data než na kterých byl model natrénován. Toto je zvlášť problematické při používání těchto modelů v reálném světě, kde může model přijít do styku s daty, kterým nemůže rozumět. V takových případech je důležité, aby model poskytoval spolehlivé odhady neurčitosti. Predikce, u kterých si model není jistý, poté zkontroluje člověk, případně se tyto data mohou použít pro dotrénování modelu.

Tato práce se zaměřuje na porovnání tří oblíbených metod pro zlepšení odhadů neurčitosti: hlubokých ansámlů, monte carlo dropout a temperature scaling. Tyto metody jsou aplikovány na šest modelů pro počítačové vidění: LeNet-5, ResNet-18, MobileNetV2, U-Net, DeepLabV3 a SSD300. U těchto metod byly prozkoumány možnosti využití na předtrénovaných modelech, i na modelech natrénovaných od nuly. Modely jsou hodnoceny na 6 datasetech počítačového vidění pro úlohy klasifikace, sémantické segmentace a detekce objektů. Modely jsou validovány za použití velkého množství metrik, hodnotící jak přesnost predikcí, tak i přesnost odhadů neurčitosti. Modely jsou rovněž evaluovány na datasetech transformovaných pomocí rotace, náhodného šumu, rozostření a změny jasu, kontrastu a saturace obrazu. Tyto experimenty nám dávají důležité informace o odolnosti modelů vůči různým typům změn v datasetech.

Tyto modifikované modely dosahují slibných výsledků. V klasifikačním datasetu FMNIST, který obsahuje 28x28 obrázků oblečení ve stupních šedi, dosáhl ansámbl pěti modelů LeNet-5 o 1,2 % lepší přesnosti než základní LeNet-5 model. Na rotovaném FMNIST datasetu dosáhl LeNet-5 model s přidanými dropout vrstvami až o 5 % lepší přesnosti a zároveň měl kvalitnější odhady neurčitosti.

V klasifikačním datasetu CIFAR-100, který obsahuje 32x32 realistické fotografie 100 kategorií různých objektů, dosáhl ansámbl pěti modelů ResNet-18 téměř o 11 % lepší přesnosti než základní model ResNet-18 a téměř o 6 % lepší kalibrační chyby. Modifikované modely s přidanými dropout vrstvami dosahují lepší přesnosti až o 4 %, mají spolehlivější odhady neurčitosti a zároveň jsou odolnější vůči změnám v transformovaných datasetech.

V segmentačním datasetu MedSeg COVID-19, který obsahuje CT skeny plic pacientů s COVID-19, s přidaným náhodným šumem dosahuje ansámbl modelů U-Net zvýšení přesnosti až o 70 % a o 0.2 v IOU v porovnání se základním U-Net modelem. Ansámbl modelů dosahuje rovněž lepších odhadů neurčitosti. U-Net model s přidanými dropout vrstvami dosahuje na datasetu s přidaným šumem zlepšení přesnosti až o 20 % a zároveň má spolehlivější predikce neurčitosti.

V segmentačním datasetu PASCAL-VOC, který obsahuje fotografie realistických scén, transformovaného pomocí náhodného šumu, dosahuje DeepLabV3 model s přidanými dropout vrstvami lepší přesnosti až o 5 % a zároveň má kvalitnější odhady neurčitosti.

V detekčním datasetu PASCAL-VOC dosahují všechny varianty modelu SSD300 téměř totožných výsledků.

Z těchto výsledků je patrné, že ansámbly a přidávání dropout vrstev může zlepšit přesnost i kvalitu odhadů neurčitosti klasifikačních a segmentačních modelů natrénovaných od nuly i předtrénovaných. Tyto výsledky se ale nepotvrdily na modelu SSD300 určenému pro detekci objektů.

Temperature scaling vylepšil kvalitu odhadů neurčitosti u všech modelů, které neměly už předtím kvalitní odhady. Velkou výhodou této metody je, že není potřeba zasahovat do architektury modelu nebo ho znovu trénovat.

Monte carlo dropout sice dosáhl lepších výsledků než základní modely, toto bylo ale díky přidaným dropout vrstvám, nikoliv samotnému monte carlo dropout vzorkování. Monte carlo dropout vzorkování samo o sobě zlepšuje kvalitu odhadů neurčitosti, ale pouze u modelů které mají přehnaně sebejisté predikce. Velkou nevýhodou této metody je nutnost provést několik predikcí, které se poté zprůměrují. Toto značně zpomaluje inferenci. Proto v této práci nebyl objeven důvod, který by zdůvodnil použití Monte carlo dropout vzorkování namísto například temperature scalingu a přidáním dropout vrstev do modelu.

Ansámbly dosahují dobrých výsledků na standartních test setech, ale na transformovaných datech často dosahují horších výsledků než architektury s přidanými dropout vrstvami. Pro optimální výsledky je možné zkombinovat několik modelů s přidaným dropoutem do ansámblu a tento ansámbl poté ještě upravit temperature scalingem.

Assessment of Uncertainty of Neural Net Predictions in the Tasks of Classification, Detection and Segmentation

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of prof. Ing. Adam Herout PhD. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Jiří Vlasák
May 8, 2022

Acknowledgements

I thank my supervisor for regular helpful consultations and my colleagues at Konica Minolta Global R&D for suggesting this assignment and lending me a server with GPU.

Contents

Chapter 1

Introduction

Modern deep neural networks have over the years significantly improved their performance on common classification, segmentation, and object detection tasks. Unfortunately, as these models get bigger and more complex, their predictions of uncertainty seem to be less precise and they are often overconfident. This can lead to difficulties when employing these models in the real world. In the real world, the model can often encounter data that are very different from those it was trained on. In these situations, the model needs to provide a low confidence score so that it can for example be retrained using these unknown data samples, or a human operator can handle the situation.

Past works about uncertainty estimation methods usually focus mainly on the performance of one model architecture on one dataset using a limited set of evaluation metrics. Moreover, most of the time, these methods are only compared to baseline methods. This can cause problems when choosing the perfect method for a specific model architecture and problem settings.

Instead, this work focuses on multiple model architectures and evaluates them on various computer vision datasets for image classification, semantic segmentation, and object detection. The uncertainty estimation methods are evaluated not only on models trained from scratch but also when being employed on pretrained models. This is important since training large models from scratch can be computationally costly and time-consuming.

The modified models are also evaluated on datasets distorted by various strengths of image distortions, which simulate changes to images frequently encountered in the real world (noise, blur, change in brightness). These experiments provide valuable findings about predictive performance and quality of uncertainty estimations of these models when tasked with significantly different data than they were trained on.

Chapter 2 describes model architectures used in the experiments. Chapter 3 describes the used evaluation metrics to measure model performance and uncertainty estimation quality. Chapter 4 focuses on uncertainty estimation techniques that can improve the model‘s predictive performance and uncertainty estimation quality. Chapter 5 describes the structure of experiments, evaluates the models, and forms practical recommendations for employing these uncertainty estimation methods in practice.

Chapter 2

Used Deep Learning Models

This chapter describes commonly used deep learning models for image classification, semantic segmentation, and object detection. These models are evaluated in chapter 5.

2.1 LeNet-5

The LeNet-5 [21] is a simple convolutional neural network used for image classification. LeNet-5 has two convolution layers, after each convolution there's a max pooling layer and after the last max pooling layer there's a block of 3 fully connected layers. The tanh activation function is applied after every convolution and fully connected layer.

LeNet-5 was developed to classify handwritten zip code digits in 1989.

2.2 ResNet

ResNet [12] is a convolutional neural network that introduced skip connections. It can be used for image classification or as a backbone for semantic segmentation, instance segmentation, object detection, etc. Skip connections operate by jumping over one or more following layers. This helps with the vanishing gradient problem. Vanishing gradient problem can occur in deeper networks when backpropagation to earlier layers can make the gradient infinitely small.

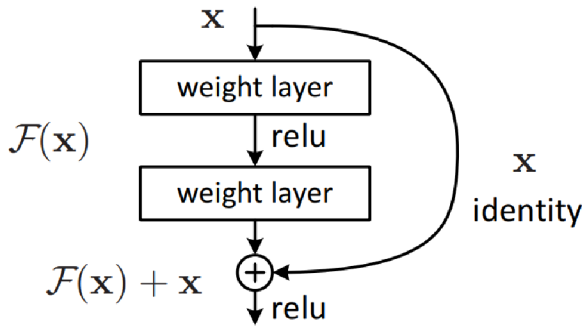


Figure 2.1: Example of a skip connection (taken from [12]).

There are number of variants of the ResNet networks. Authors suggested 18, 34, 50, 101 and 152 layer variants. While the 18 and 34 layer variants use the basic block, which

consists of two 3x3 convolutions the larger variants use a bottleneck block, which consists of two 1x1 convolutions and a one 3x3 convolution, which serves as a bottleneck and helps to reduce model complexity in deeper variants.

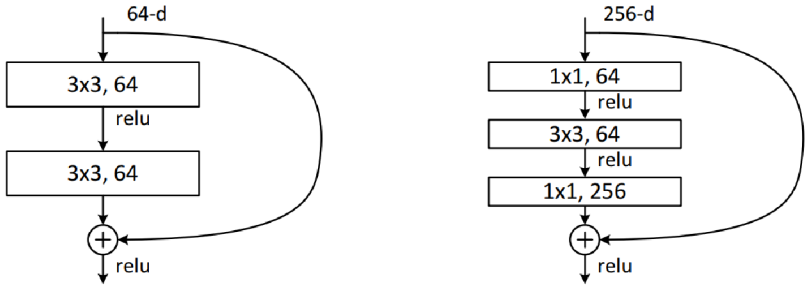


Figure 2.2: ResNet basic building block (left), used in the 18 and 34 layer variants, and the bottleneck block, used in the 50, 101 and 152 layer variants (taken from [12]).

ResNet won the ImageNet 2015 [6] image classification, detection, and localization competitions as well as the MS COCO 2015 [22] detection and segmentation competitions. Since then, skip connections have been used in many other neural network architectures such as DenseNet [16] or ResNext [36].

2.3 MobileNet

MobileNet is a family of small, efficient and fast neural networks designed to work on embedded devices. They can be used for image classification or as backbones for semantic segmentation, instance segmentation, object detection, etc. There are 3 versions of the MobileNet architectures: MobileNetV1 [15], MobileNetV2 [32], and MobileNetV3 [14].

2.3.1 MobileNetV1

Instead of standard convolution filters, MobileNetV1 splits the convolution operation into depthwise and pointwise convolutions. This approach produces features with same dimensions but needs less parameters.

Depthwise convolution applies a single filter to a single input channel, whereas standard convolution applies the filters to all of the input channels. Depthwise convolution basically filters the data from one channel. To combine multiple channels and to create new features a pointwise convolution is used. Pointwise convolution combines the output of depthwise convolution by a 1×1 filter. This approach significantly reduces computational cost while only slightly decreasing accuracy.

2.3.2 MobileNetV2

MobileNetV2 added a second pointwise convolution before depthwise convolution that expands the data before filtering them with depthwise convolution. Also, a jump connection spanning the block was added. This whole block was named Inverted Residual Block and helped to improve the performance.

2.3.3 MobileNetV3

MobileNetV3 added a squeeze and excite module after the second pointwise convolution. This helps to give unequal weights to different channels. Also Network Architecture Search [38] (NAS) and NetAdapt [37] algorithms were used to improve the architecture.

2.4 U-Net

U-Net [31] is a fully convolutional neural network, which means it does not containt any fully connected layers and therefore can accept image of any size. It was developed for use in medical image segmentation.

It consists of downsampling and upsampling paths. In the downsampling path, 3×3 convolutions and ReLU layers are followed by 2×2 max pooling layers. In the upsampling path, 3×3 convolutions and ReLU layers are followed by 2×2 transposed convolution layers which provide the upsampling. The output of every downsampling block is also cropped and concatenated to the corresponding upsampling block.

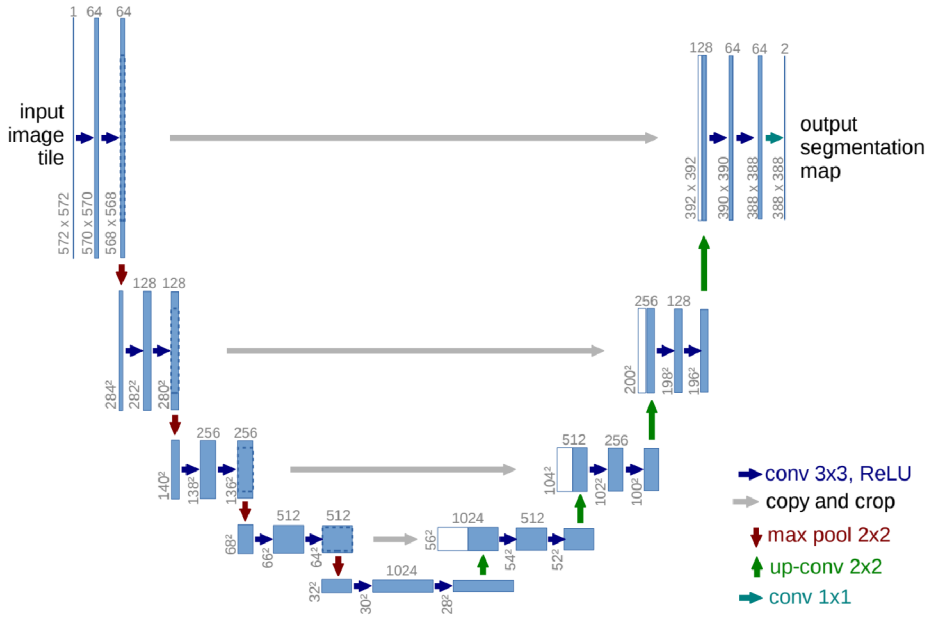


Figure 2.3: The U-Net architecture. Downsampling path (left) is being cropped and concatenated to the upsampling path (right). Taken from [31].

2.5 DeepLab

DeepLab is a family of convolutional neural networks used for semantic image segmentation on realistic scenes. There are 4 versions of these models: DeepLabV1 [2], DeepLabV2 [3], DeepLabV3 [4] and DeepLabV3+ [5]. These models successively improved the state of the art results on the PASCAL VOC-2012 semantic image segmentation task.

Each of these models consist of a backbone, which produces features, and a classifier head, which computes classification scores for each pixel.

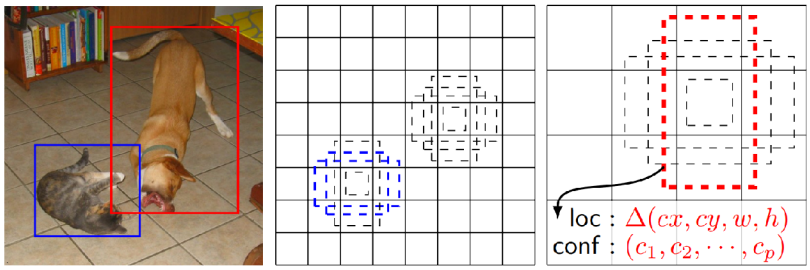
One novel idea that all of the DeepLab models share is called atrous convolution. While standard convolution always samples neighbouring pixels, atrous convolution samples only

every nth pixel. This helps to widen the receptive field of filters and to incorporate larger context.

2.6 Single Shot Detector (SSD)

Single Shot Detector (SSD) [23] is an object detection model. It uses a single shot approach, which is different from two step approach used by region proposal networks like RCNN [9] and Faster-RCNN [30]. The original architecture uses the VGG16 [34] network as the backbone.

It uses multiple predefined bounding box aspect ratios. These predefined boxes are placed on feature maps of different scales. This helps to detect objects of various sizes and shapes. For each of these predefined boxes, it outputs confidence scores for the presence of each object and box shape offsets to better match the predefined box to the object shape.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2.4: Example of the SSD prediction process. The detector uses predefined bounding boxes with different aspect ratios. And evaluates them at feature maps with different sizes. This helps the model to detect objects with different shapes and sizes. The model outputs classification scores and regression offsets for the location of each bounding box (image taken from [23]).

For training, the total MultiBox loss is calculated as a weighted sum of localization loss (Smooth L1 loss) and classification loss (Cross Entropy loss).

Chapter 3

Evaluation Metrics

This chapter describes commonly used evaluation metrics for deep learning models for image classification, semantic segmentation, and object detection tasks. These metrics evaluate both the predictive performance as well as quality of uncertainty estimation.

3.1 Prediction Accuracy

Prediction acuracy is defined as the number of correct predictions divided by the number of total predictions. Accuracy is commonly used in image classification and semantic segmentation problems. However, it can be misleading when used on imbalanced datasets, where the model can ignore the lower-represented classes and still achieve good overall accuracy. These imbalances are common in semantic image segmentation datasets.

3.2 Average Confidence

Average confidence is an average of confidence scores on all samples in the test set. These scores are usually obtained from the softmax function. Confidence should encapsulate how likely is the prediction correct. For a well calibrated model, its average confidence score should be close to its average accuracy.

Maximum possible confidence is 1. Minimal possible confidence confidence happens when the output probability distribution is flat (it has highest possible entropy). The lowest possible confidence value is $\frac{1}{N}$ where N is the number of classes.

3.3 Brier Score

Brier score [1] measures accuracy of probability predictions. Its definition is equivalent to the Mean Squared Error:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{3.1}$$

Where N is the number of classes, $\vec{x}$ is the probability output vector and $\vec{y}$ is the actual outcome vector of the events (0 = event did not occur, 1 = event did occur).

Brier score is a proper scoring rule, which means that there is no trivial solution which would yield perfect brier score. When the model achieves a perfect Brier score of 0, the model has a perfect calibration and accuracy.

3.4 Calibration Graph and Expected Calibration Error

Calibration measures how close are the model confidence scores to the real probability of being correct. Perfect calibration means that for example, of all the predictions with confidence score 0.5, 50 % of them are correct.

There are several ways to observe calibration. One of which is a calibration graph. An example of a calibration graph can be found in figure 5.2. Calibration graphs show accuracy as a function of confidence. Model predictions are grouped into equally spaced bins by their confidence levels. Accuracy is then computed for each bin. Perfect calibration occurs when the mean accuracy of the bin is the same as the mean confidence of the bin. The difference between the bin‘s mean accuracy and its mean confidence value is shown in orange. The bottom row of the graphs shows the confidence distribution of the samples.

Another way of measuring calibration is by using the Expected Calibration Error (ECE) [27], which is defined as:.

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{3.2}$$

It is basically a weighted average of the difference between accuracy and confidences of the bins (the orange bars in the calibration graph).

Although calibration graphs and ECE can be useful, they have shortcomings. Models can have perfect calibration on the dataset level but still be overconfident on some parts of the dataset and uncerconfident on other parts [17].

3.5 Receiver Operating Characteristic Curve

A receiver operating characteristic (ROC) curve measures the ability of a binary classifier to classify between positive and negative samples as its classification threshold is varied. It is created by plotting the true positive rate (TPR, also known as sensitivity or recall) against the false positive rate (FPR, also known as specificity) while increasing the classification threshold from 0 to 1.

The true positive rate measures how many of the positive samples were classified as positive.

$$TPr = \frac{TP}{TP + FN} \tag{3.3}$$

The false positive rate measures how many negative samples were classified as negative.

$$FPr = \frac{FP}{FP + TN} \tag{3.4}$$

In the context of uncertainty estimation models, ROC curves are often used to measure the model‘s ability to assign lower confidence scores to incorrectly classified (or out of distribution) samples and higher confidence scores to correctly classified samples.

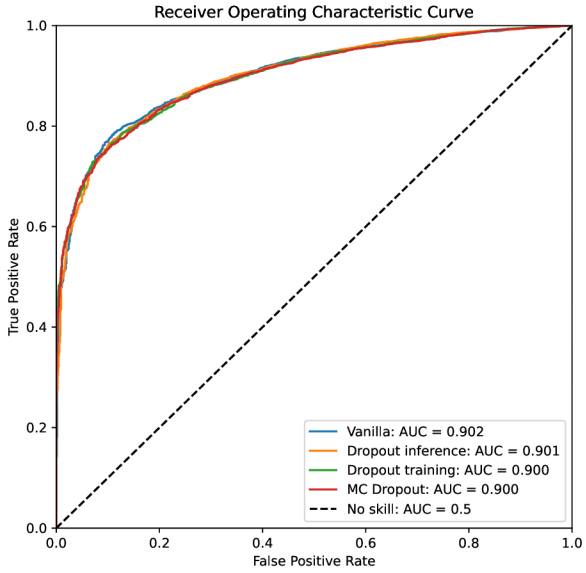


Figure 3.1: Receiver Operating Characteristic Curve of the LeNet-5 models on the FMNIST test set.

To better quantify the classifier performance, Area Under Curve (AUC) is often computed. The maximum possible AUC is 1 with the curve reaching the point (0.0, 1.0), which means 1.0 true positive rate and 0.0 false positive rate.

3.6 Precision Recall Curve

Similarly as the ROC curve, the Precision Recall (PR) curve measures the quality of model scores for positive (correct or in distribution) and negative (incorrect or out of distribution) samples. PR curve plots the classifier precision against recall while changing the classification threshold. PR curve is often preffered over ROC curve when the dataset is more imbalanced.

Precision measures how many predicted positives are true positives.

$$P = \frac{TP}{TP + FP} \tag{3.5}$$

Recall measures how many true positives were predicted as positives.

$$R = \frac{TP}{TP + FN} \tag{3.6}$$

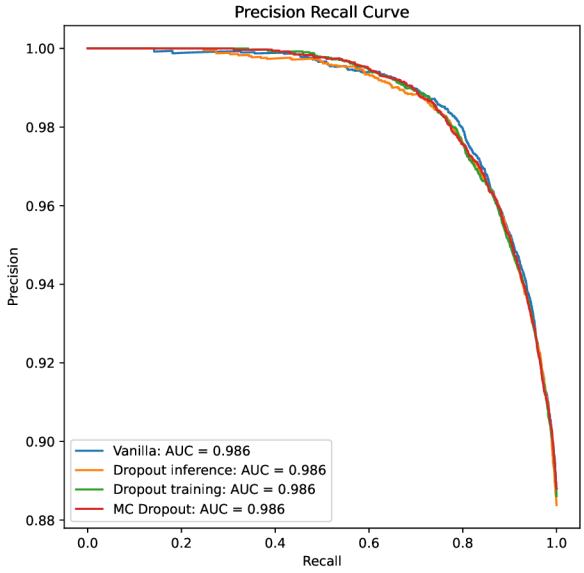


Figure 3.2: Precision Recall Curve of the LeNet-5 models on the FMNIST test set.

Similarly as with ROC curves, to better quantify the classifier performance on the PR curve, the area under the curve (AUC) is computed. The maximum possible AUC is 1 with the curve reaching the point (1.0, 1.0), which means 1.0 precision and 1.0 recall.

3.7 Intersection Over Union

Intersection Over Union (IOU, also known as Jaccard index) is a metric commonly used in segmentation and object detection. Measures how well the area of predictions fits the area of ground truth. It is defined as the area of intersection of the predictions and ground truth divided by the area of the union of the predictions and ground truth.

$$IOU = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)} \tag{3.7}$$

For semantic segmentation, IOU is computed for the predicted and ground truth pixelwise masks. Semantic segmentation, datasets are commonly very imbalanced, so a simple pixelwise accuracy is not enough to judge the performance of a model on all classes. In this case, mean class IOU is therefore much better metric since it weights mean IOU of every class equally.

For object detection, IOU is computed for the predicted and ground truth bounding boxes and it is used as a measurement on how tight do the predicted bounding boxes fit. It is also used to compute the Mean Average Precision metric, where only predictions with IOU larger than some threshold are considered true positives.

Semantic segmentation experiments in sections 5.3 and 5.4 use two kinds of IOU implementations. Mean Class IOU (mIOU) is defined as the mean of pixelwise IOUs for each class. And IOU is defined as the mean IOU of all foreground pixels.

3.8 Mean Average Precision

Mean Average Precision (mAP) is the most commonly used metric to evaluate object detectors. It draws a precision recall curve for a every class and a certain IOU threshold. This curve is then smoothed: The precision value for recall r is replaced with the maximum precision for any recall $\geq r$.

$$p_{interp}(r) = \max p(\hat{r}) \text{ for } \hat{r} \geq r. \tag{3.8}$$

The exact evaluation procedure for this smoothed precision recall curve depends on the dataset.

3.8.1 PASCAL-VOC 2007 Mean Average Precision

The PASCAL-VOC [7] 2007 detection competition evaluates the smoothed precision recall curve in 11 equally spaced recall values from 0.0 to 1.0. For each of these recall values, the precision is recorded. The Average Precision (AP) for each class is the mean of these values. To obtain the mAP, the AP for each class is averaged. This is the implementation used for evaluation in Section 5.5.

3.8.2 PASCAL-VOC 2008-2012 Mean Average Precision

The PASCAL-VOC 2008-2012 detection competitions evaluate the smoothed curve differently. The smoothed curve is sampled at each point and the Area Under Curve (AUC) is computed. This method is more precise since no approximation is done.

3.8.3 MS COCO Mean Average Precision

The MS COCO [22] dataset uses the term mAP differently. The PR curve is sampled at 101 points and the class average is called the AP. This would be traditionally called the mAP. To obtain the MS COCO mAP, the AP is computed for multiple IOU thresholds from 0.5 to 0.95 using 0.05 steps and multiple sizes of bounding boxes. Then, the MS COCO mAP is the average of all of these categories.

Chapter 4

Uncertainty Estimation Techniques

This chapter describes some of the commonly used methods to improve predictive uncertainty estimation and model calibration.

Bayesian Neural Networks (BNNs) have been commonly used to improve the uncertainty estimation of models. BNNs work by learning a probability distribution of the model parameters. However, they are difficult to train and often have worse performance than deterministic models. The Monte Carlo Dropout [8] and Deep Ensembles [20] methods have been created to also model distribution over their model parameters while being based on existing deterministic neural network architectures.

4.1 Baseline (softmax)

In a classification problem, neural networks output a vector which usually does not sum to 1 and can contain even negative numbers. To convert this vector into probability distribution, the softmax function is used. The softmax function is defined as:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{4.1}$$

Where K is the length of the output vector or, in other words, the number of classes in the classification problem and $\vec{z}$ is the output vector. Basically, it applies the exponential function to every element z_i of the output vector $\vec{z}$ and then normalizes these values by dividing them by the sum of these exponentials. The output of the softmax function is another vector which sums to 1 and only contains numbers from 0 to 1 and can therefore be interpreted as probability distribution.

The softmax function is a generalization of the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to multiple dimensions. Since the softmax function uses the exponential function, small changes in input around 0 can lead to large changes in output. This can make it difficult for the networks to output an uniform probability distribution, which would be interpreted as prediction with very low confidence.

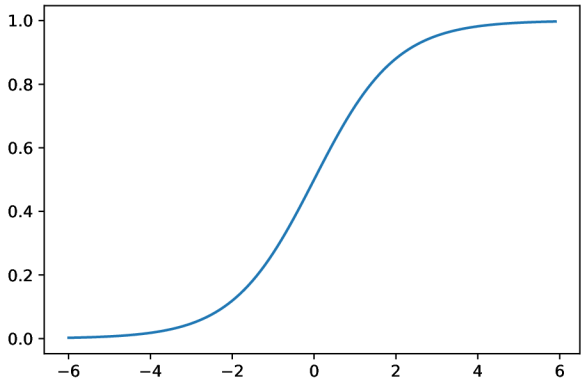


Figure 4.1: Plot of the sigmoid function.

Experiments in [13] show that while the softmax function outputs have very often poor correspondence to confidence, the maximum probability from the softmax function tends to be lower for out of distribution samples or incorrectly classified samples. This can make it sufficient to use the softmax outputs for the detection of incorrectly classified or out-of-distribution samples even though the prediction probabilities are often misleading.

4.2 Temperature Scaling

Temperature Scaling is a simple extension of the Platt scaling [29] technique. Instead of training a logistic regression model on the model output logits, temperature scaling uses a single scalar parameter $T > 0$, which divides the model output logits.

The T parameter is optimized with respect to the Negative Log Likelihood loss on the validation set. Increasing T beyond 1 flattens the probability distribution and increases entropy. In turn, decreasing T decreases entropy and sharpens the output distribution. With $T = 1$, the output distribution remains unchanged.

Temperature scaling is a simple technique which can be used on already trained models without needing to change their architecture. This method also has no effect on prediction accuracy since all logits are divided by the same number T and therefore the largest logit remains the largest.

One disadvantage may be that there needs to be a separate validation set apart from the training and test sets on which the T parameter is trained. From my experience, another limitation is that training the T parameter may be difficult if the neural network is already well calibrated. Despite its simplicity, the technique performed the best in experiments conducted in [10].

4.3 Monte Carlo Dropout

Monte Carlo Dropout works by using dropout at test time, performing multiple forward passes, and then averaging these forward passes to get the final model input. This method was introduced in [8] where it was shown that this technique approximates the distribution of the model parameters. This is similar to how a Bayesian neural network (BNN) operates.

Advantages to using Monte Carlo Dropout compared to BNNs are that there is no increase in training time, only small changes to existing model architectures are needed

and there should be no decrease in accuracy. These advantages and a relatively simple setup has made Monte Carlo Dropout a popular choice for uncertainty estimation.

One disadvantage of this method is that it generally requires a relatively large number of forward passes ($\geq$ 20) to provide optimal prediction accuracy and uncertainty estimation. Another disadvantage is that there are infinitely many possibilities to setup the dropout layers inside the model architecture. For example, in [24] `Dropout2d` layers are used after every `ReLU` activation of the ResNet model, while in [18] and [26] dropout layers are used only on deeper layers. After adding dropout to the model architecture, the model also has to be finetuned or retrained from scratch. To achieve the best results, the dropout probability p also needs to be set up properly. All of this makes Monte Carlo Dropout tricky to use to its full potential.

4.3.1 Dropout and Dropout2d layers

There are two types of dropout layers used in this work. While the standard dropout layers are used on 1D outputs of fully connected layers, the `Dropout2d` layers are used on outputs of 2D convolution layers. The standard dropout layers randomly zero out some of the elements of the input tensor with probability p. The `Dropout2d` zeroes out whole channels of the input tensor with probability p.

4.4 Deep Ensembles

Deep ensemble is a model which combines several independently trained deep neural networks. To obtain the ensemble‘s predictions, the probability distributions of its members are averaged. Deep ensembles have been long known to improve the accuracy of predictions. However, in [20] ensembles were also found out to improve calibration and robustness to dataset shift.

Classical machine learning algorithms used as ensemble members have been usually trained only on subsets of the whole training set (bagging). This introduces randomness and decreases correlation between ensemble members. However, since neural networks are initialized with random parameters and have multiple local optima, there is no need to introduce another source of randomness. Neural networks also generally achieve better results when trained on more data. Therefore, deep ensemble members are usually trained on the whole training set.

An advantage of deep ensembles compared to monte carlo dropout is that there is no need to modify the model architecture. Unfortunately, both the time and the memory needed for training and inferencing scale linearly with the number of ensemble members. However, both training and inferencing can be parallelized.

Experiments in [20] show that ensembles outperform vanilla models and monte carlo dropout models in terms of accuracy, calibration and robustness to dataset shift. Since then, deep ensembles have often been used to improve uncertainty estimation with good results [28] [11].

Chapter 5

Experiments, Findings and Practical Recommendations

This chapter describes conducted experiments comparing popular uncertainty estimation methods on multiple datasets for image classification, semantic segmentation, and object detection. In each section, uncertainty estimation methods are evaluated on the standard test set as well as on artificially transformed test set with increasing strength of shift to simulate out-of-distribution data. The methods are evaluated on metrics that measure predictive performance (accuracy, IOU) as well as metrics that measure quality of uncertainty estimations (ECE, Brier score).

The goal of these experiments is not to train models with the best possible predictive performance on the given dataset but to evaluate uncertainty estimation methods in real world conditions. Given this, usually very little (or none) hyperparameter tuning and data augmentation was done for each experiment.

5.1 Image Classification on the FMNIST dataset

First experiment was conducted using the FMNIST [35] dataset which consists of 60000 training images and 10000 test images of different types of clothing. Each image is 28x28 grayscale and has 1 of 10 classes. Since this is an image classification dataset, the goal of the model is to classify each image into one of the 10 classes.

The training set of 60000 samples was divided further into a validation set of 6000 samples, used for **Temperature scaling**, and an actual training set of 54000 samples. Figure 5.1 shows examples of the FMNIST dataset.

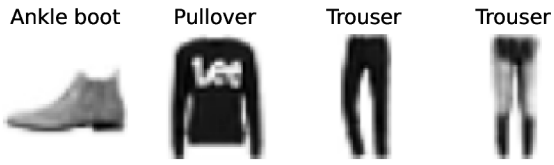


Figure 5.1: Examples of the FMNIST dataset

LeNet-5 [21] network architecture was used. This was done to see the effect of the methods on simple and small model.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Table 5.1: Classes in the FMNIST dataset

Training was done on the training set for 20 epochs using the Adam optimizer with Pytorch's default parameters and cross entropy loss. After each epoch, the model was evaluated on the test set and test loss was computed. If the current test loss was lower than the previous lowest test loss, the model checkpoint was saved. Therefore, for evaluation, models with the lowest possible validation loss were used.

In total, 6 models were evaluated:

- **Vanilla** is a standard LeNet-5 model.
- **Dropout inference** uses the same weights as **Vanilla** and the same architecture as the **Dropout training** model. Dropout is active in test time, and 20 forward passes are averaged to form the final prediction..
- **Temperature scaling** uses the same weights and architecture as **Vanilla** but the model logits are post-processed using the temperature scaling technique.
- **Dropout training** is a LeNet-5 model with added dropout. The dropout model has Dropout2D layers after second and third tanh activation and standard Dropout layer after the first fully connected layer. All dropout layers had probability $p = 0.2$. Dropout is turned off for test time.
- **MC Dropout** uses the same architecture and weights as the **Dropout training** model while also using dropout inference.
- **Ensemble** model is made up of 5 independently trained **Vanilla** models whose outputs are averaged.

5.1.1 Results on Standard Test Set

Table 5.2 shows evaluation results on the standard test set. The **Ensemble** model seems to perform the best in all metrics by a significant margin. There seems to be little difference between other models. Interestingly, models that use dropout inference (**Dropout inference** and **MC Dropout**) have higher ECE than models that do not use it. This is probably because the **LeNet-5** network used is relatively small and, therefore, is not as prone to overconfidence. Since the models are not overconfident, the dropout inference makes them underconfident. To achieve better calibration results, the dropout probability

p would need to be better tuned for each model that uses dropout inference (**Dropout inference** and **MC Dropout**).

Method	Accuracy	ECE	Avg. Conf.	Brier score	AUROC	AUPR
Vanilla	0.887	3.330	0.917	0.016	0.902	0.986
Dropout inference	0.887	4.536	0.862	0.016	0.901	0.987
Temperature scaling	0.887	2.958	0.887	0.016	0.903	0.987
Dropout training	0.886	2.956	0.901	0.016	0.900	0.986
MC Dropout	0.885	4.056	0.869	0.016	0.901	0.986
Ensemble	**0.902**	**2.834**	0.910	**0.014**	**0.908**	**0.989**

Table 5.2: Results on on standard FMNIST test set. ECE = Expected Calibration Error, Avg. Conf. = Average confidence score, AUROC = Area Under Receiver Operating Characteristic, AUPR = Area Under Precision Recall.

Figure 5.2 shows calibration graph for the **Vanilla**, **Dropout inference** and **Temperature scaling** models. The models achieve relatively low ECE. While the **Vanilla** and **Temperature scaling** models are very slightly overconfident, the **Dropout inference** is slightly underconfident. This may be because the dropout probability $p = 0.2$, which was used in the **Dropout training** architecture, may be too large for the **Vanilla** weights. **Temperature scaling** seems to only improve the calibration slightly because the **Vanilla** model is already calibrated relatively well.

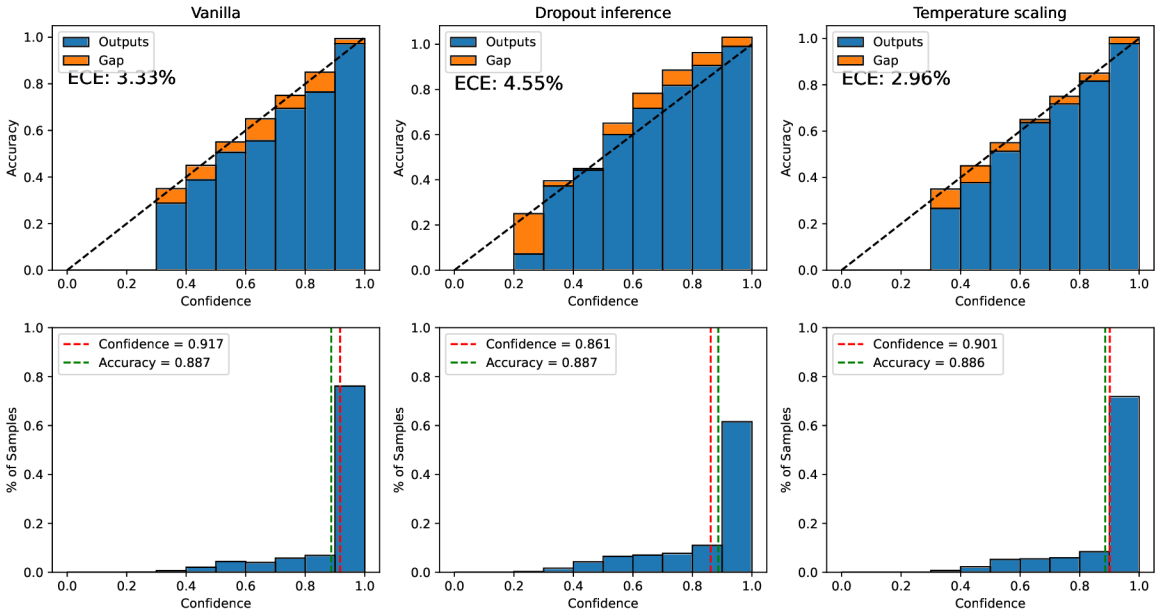


Figure 5.2: Calibration graphs for the LeNet-5 **Vanilla**, **Dropout inference** and **Temperature scaling** models on the FMNIST test set.

Figure 5.3 shows the calibration graph for the **Dropout training**, **MC Dropout**, and **Ensemble** models. The **MC Dropout** model has higher ECE than the **Dropout training** model. This is probably because the **Dropout training** model is already well calibrated and dropout inference made the model underconfident (same as the **Vanilla** and

Dropout inference models). In addition to having the highest accuracy, the **Ensemble** model has also the lowest ECE.

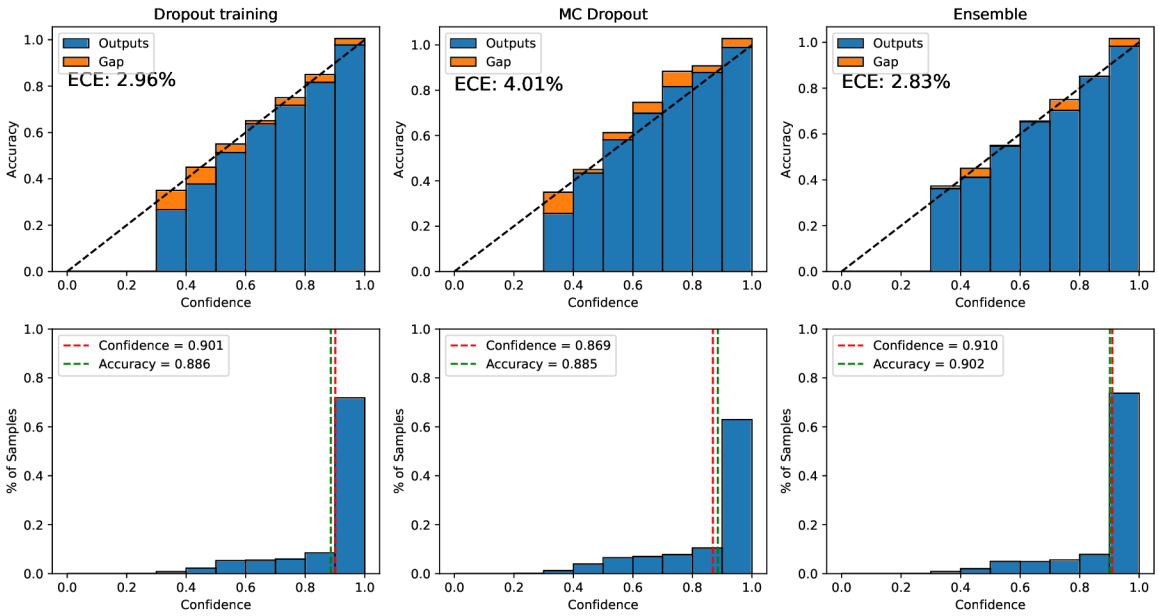


Figure 5.3: Calibration graphs for the LeNet-5 **Dropout training**, **MC Dropout** and **Ensemble** models on the FMNIST dataset.

5.1.2 Results on Rotated Test Set

In this section test set is rotated from 0 to 60 degrees using 5 degree steps. At each step, the models are evaluated on all metrics. This experiment shows how much are the models resistant to dataset shift. Figure 5.4 shows examples of the FMNIST dataset rotated by 20 degress.

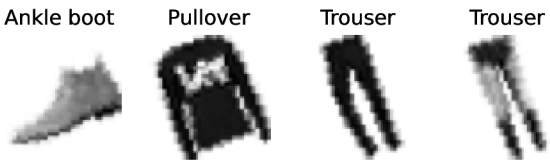


Figure 5.4: Examples of the FMNIST dataset using 20 degrees of rotation

Figure 5.5 shows the evaluation results with increasing rotation. Accuracy decreases with more rotation similarly for all of the models. Both Brier score and ECE increases fastest for the **Vanilla** model. While the **Ensemble** model has the lowest ECE on the normal test set, it is outperformed on more shifted samples by the **Dropout inference** and **MC Dropout** models. This hints at possible advantages of using dropout sampling when encountering out-of-distribution data.

When looking at the **AUROC** and **AUPR** metrics, all the models perform similarly. However, the **Ensemble**, **Dropout inference** and **MC Dropout** models perform the best.

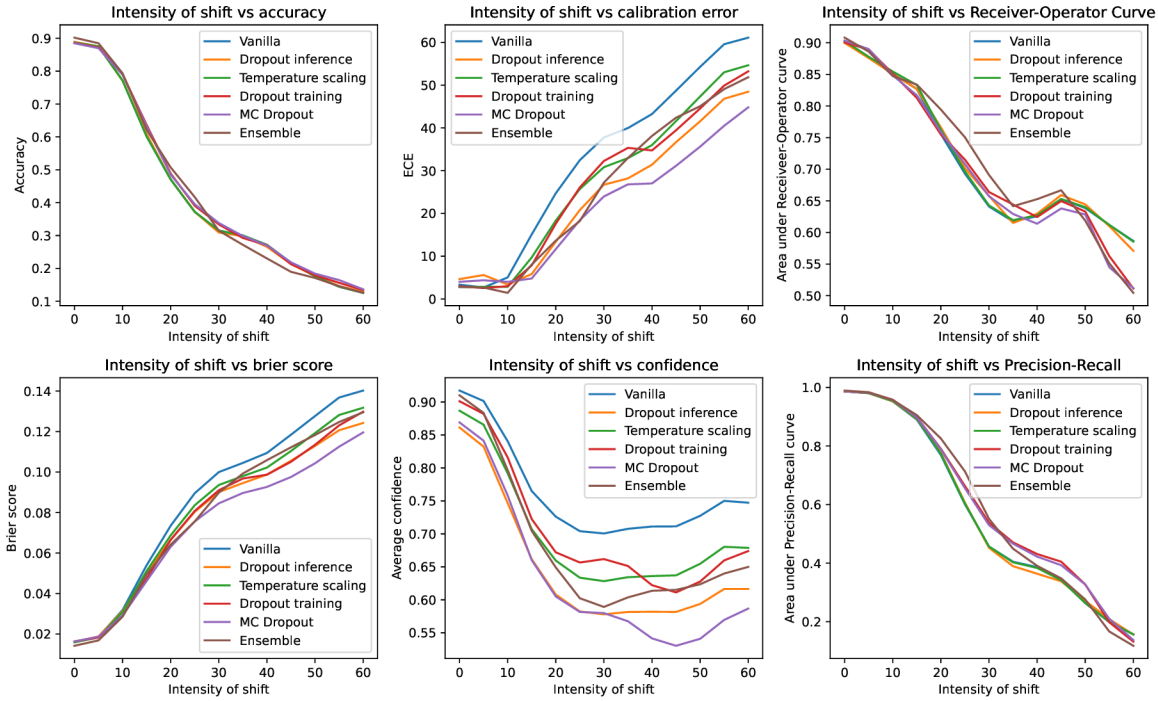


Figure 5.5: Graph of evaluation metrics on the FMNIST test set with increasing degrees of rotation, using the LeNet-5 network.

5.1.3 Results on Test Set with Additive Gaussian Noise

In this section, the models are evaluated on test set with additive Gaussian noise added. Equation 5.1 shows the definition of random additive Gaussian noise transformation. Where x is the input image, *rand()* is standard normal distribution generator and *std* is the standard deviation.

$$x + rand() * std \tag{5.1}$$

In this experiment, the *std* parameter is increased from 0 to 0.6 using 0.05 steps. Figure 5.6 shows examples of the FMNIST dataset transformed with additive Gaussian noise using $std = 0.2$.

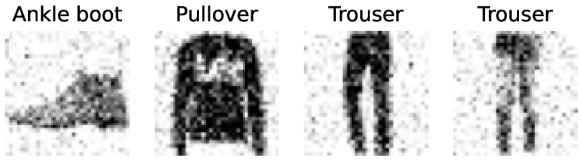


Figure 5.6: Examples of the FMNIST dataset with additive Gaussian noise using $std = 0.2$.

Figure 5.7 shows evaluation results with increasing parameter *std*. Although accuracy is fairly similar on the standard test set, the models trained with dropout (**Dropout training** and **MC Dropout**) have significantly lower accuracy on noisier data than the other models. This is surprising because, intuitively dropout during training should make models more resistant to noise in data.

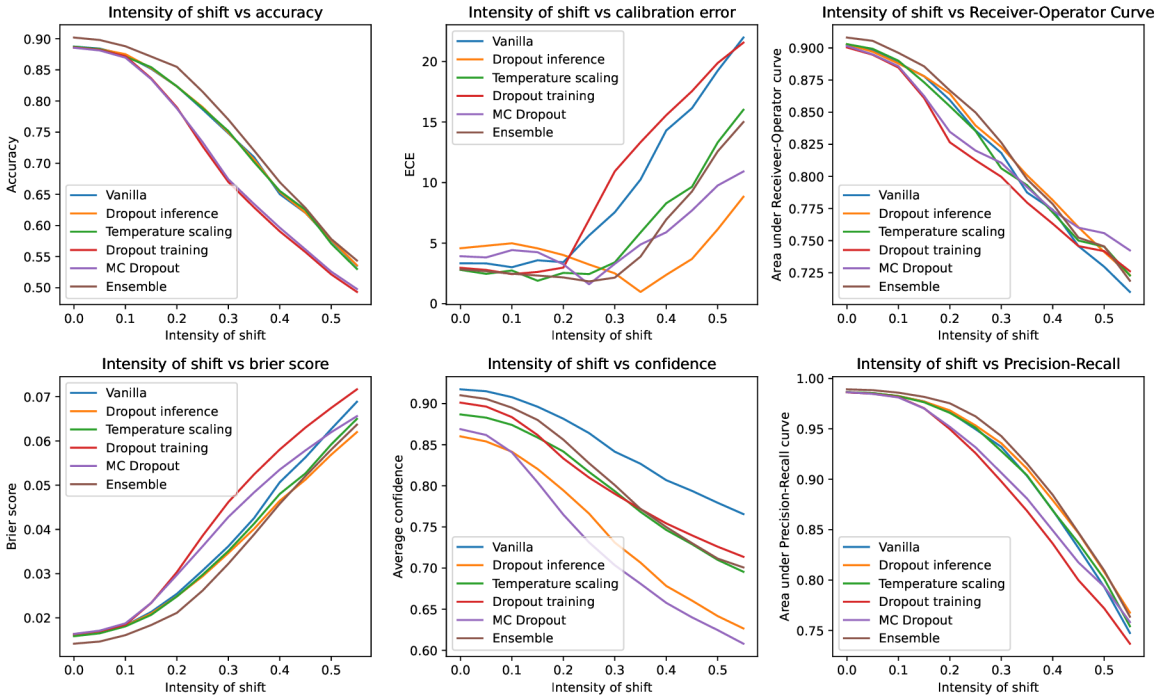


Figure 5.7: Graph of evaluation metrics on the FMNIST test set with increasing strength of Gaussian noise, using the LeNet-5 network.

Dropout training also seems to have the worst Brier score and ECE on noisier samples. In general **Dropout inference** achieves the best ECE on data with more noise. We can see that the ECE curve even goes down before going up again. This is probably because the chosen probability $p = 0.15$ is too high and, therefore, the model is underconfident on less shifted data. As the shift strength increases, the model becomes overconfident, the same as the other models.

5.1.4 Findings

Experiments using both transformations show that **Ensemble** improves predictive performance and uncertainty estimation even on noisier samples. The **Dropout training** and **MC Dropout** models improve accuracy on rotated test set but surprisingly achieve the worst accuracy on test set with Gaussian noise. The **Temperature scaling** and **Dropout inference** models offer better quality uncertainty estimates, compared to the **Vanilla** model, on both transformed test sets.

5.2 Image Classification on the CIFAR-100 Dataset

The CIFAR-100 dataset [19] contains 60000 32x32 color images split into 100 classes, each class containing 600 images. There are 500 training images and 100 testing images per class. Since this is an image classification dataset, the goal of the model is to classify each image into one of the 100 classes.

These 50000 training images were randomly split into 45000 images used for training and 5000 images used as the validation set for the **Temperature scaling** method. The

split was done using the `torch.utils.data.random_split` method using manual seed with value 0.

This dataset was chosen because it is fairly challenging while still being relatively small and easy to train on. Figure 5.8 shows examples of the CIFAR-100 dataset.

Figure 5.8: Examples of the CIFAR-100 dataset.

For evaluation, the ResNet-18[1] [12] and MobileNetV2[2] [32] architectures were used. Training was done for 200 epochs using the Adam optimizer with cross entropy loss and batch size of 64. Checkpoints were saved when the models reached the lowest validation loss. In total, 8 models are evaluated:

- **Vanilla** is a standard ResNet-18 or MobileNetV2 network.
- **Dropout inference** uses the same weights as **Vanilla** model and the same architecture as the **Dropout training** model. Dropout is active in test time, and 20 forward passes are being averaged to form the final prediction.
- **Temperature scaling** uses the same weights and architecture as **Vanilla** model but the model logits are post-processed using the temperature scaling technique.
- **Dropout training** is a ResNet-18 or MobileNetV2 network with added dropout. The ResNet-18 models with dropout have a single `Dropout2d` layer with $p = 0.1$ inserted at the end of each `Residual Block`. This architecture is the same as the one used in [24]. Only changing the probability p to 0.1 yielded slightly better results. Similarly, the MobileNetV2 models with dropout have a single `Dropout2d` layer with $p = 0.1$ inserted at the end of each `Inverted Residual Block`.
- **MC Dropout** uses the same architecture and weights as **Dropout training** model while also using dropout inference.
- **Ensemble** model is meade up of 5 independently trained **Vanilla** models whose outputs are averaged.
- **Finetuned** (or **Vanilla finetuned with dropout**) uses the **Dropout training** model architecture and the **Vanilla** model weights. The model with added dropout layers is then finetuned on the training set for 10 epochs. This is done to see if it is possible to add dropout to pretrained model.
- **Finetuned with MCD** (or **Vanilla finetuned with dropout with MCD**) is the same model as **Finetuned** model but also uses dropout inference.

[1] ResNet and ResNet Dropout implementations were taken from https://github.com/mattiasegu/uncertainty_estimation_deep_learning

[2] MobileNetV2 implementation was taken from https://github.com/NoUnique/MobileNet-CIFAR100.pytorch

5.2.1 Results on Standard Test Set

Table 5.3 shows the evaluation results on the standard test set for the ResNet-18 model. The **Ensemble** model achieves the best results on all metrics except ECE, where it achieves the third best result. This is consistent with the results in Section 5.1 where the **Ensemble** model also performed the best on the standard test set.

All models using dropout inference achieve better ECE than the models using normal inference. However, AUROC and AUPR do not seem to increase when using dropout inference. This is consistent with the results in Section 5.1.

Method	Accuracy	ECE	Avg. Conf.	Brier score	AUROC	AUPR
Vanilla	0.627	8.281	0.721	0.005	0.836	0.904
Dropout inference	0.619	2.102	0.605	0.005	0.836	0.899
Temperature scaling	0.627	2.461	0.609	0.005	0.838	0.904
Dropout training	0.639	8.256	0.734	0.005	0.845	0.915
MC Dropout	0.636	**1.645**	0.631	0.005	0.842	0.912
Ensemble	**0.734**	2.352	0.755	**0.004**	**0.868**	**0.950**
Finetuned	0.662	6.989	0.745	0.005	0.849	0.923
Finetuned with MCD	0.655	3.900	0.622	0.005	0.845	0.918

Table 5.3: ResNet-18 results on the CIFAR-100 test set. Avg. Conf. = Average confidence, AUROC = Area Under Precision Recall curve, AUPR = Area Under Precision Recall curve.

MobileNetV2 model results are in the table 5.4. Similarly as when using the ResNet-18 model, the **Ensemble** method seem to achieve the best results in most of the metrics. Also, like the ResNet-18 model, dropout inference seems to improve the ECE metric compared to standard inference. Interestingly, this is not the case with the **Finetuned** models, where dropout inference has a negative effect on ECE.

Method	Accuracy	ECE	Avg. Conf.	Brier score	AUROC	AUPR
Vanilla	0.593	9.036	0.692	0.005	0.831	0.889
Dropout inference	0.595	**1.916**	0.598	0.005	0.829	0.887
Temperature scaling	0.593	2.663	0.570	0.005	0.833	0.890
Dropout training	0.613	7.432	0.698	0.005	0.838	0.901
MC Dropout	0.613	2.884	0.596	0.005	0.844	0.904
Ensemble	**0.668**	4.780	0.630	**0.004**	**0.854**	**0.926**
Finetuned	0.593	5.316	0.653	0.005	0.829	0.888
Finetuned with MCD	0.596	6.566	0.532	0.005	0.827	0.885

Table 5.4: MobileNetV2 results on the CIFAR-100 test set. Avg. Conf. = Average confidence, AUROC = Area Under Precision Recall curve, AUPR = Area Under Precision Recall curve.

Figure 5.9 shows the calibration graph for the **Vanilla**, **Dropout inference**, **Temperature scaling**, and **Ensemble** models using the ResNet-18 architecture. The **Vanilla** model seems to be overconfident in its predictions and using **Dropout inference** corrects this. This is probably because the probability $p = 0.1$ well chosen and not big enough to make the model underconfident (as happened with the **Dropout inference** method in pre-

vious section 5.1). The **Temperature scaling** and **Ensemble** methods are also relatively well calibrated.

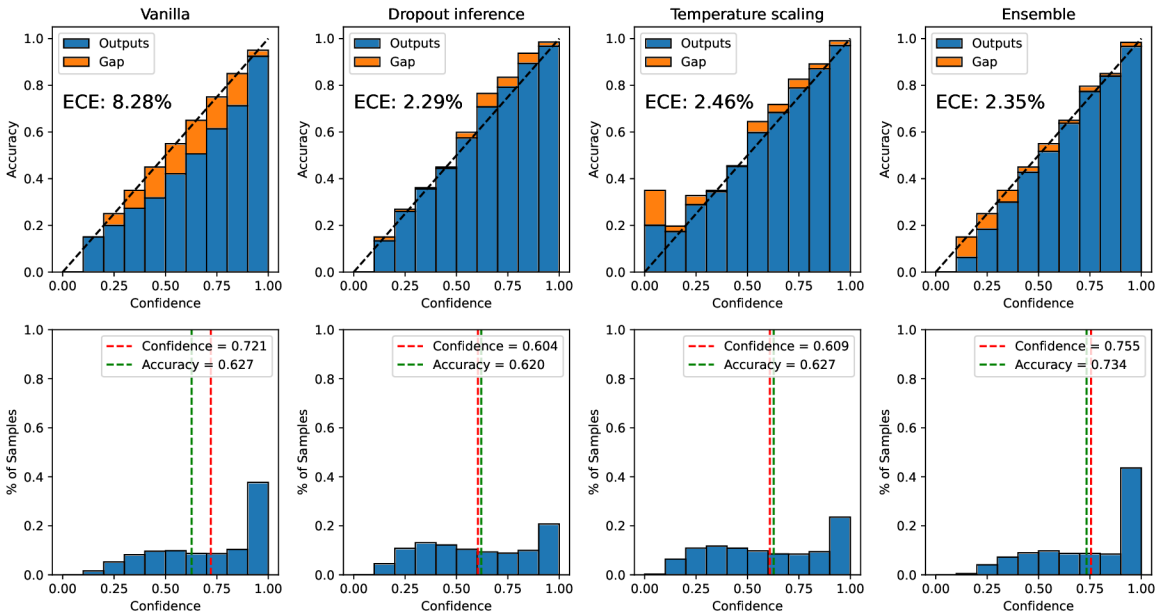


Figure 5.9: Calibration graphs on the standard CIFAR-100 test set for the ResNet-18 **Vanilla**, **Dropout inference**, **Temperature scaling** and **Ensemble** models.

Figure 5.10 shows the calibration graph for the **Dropout training**, **MC Dropout**, **Finetuned** and **Finetuned with MCD** models using the ResNet-18 architecture. Curiously, the **Dropout training** model seems suffer the same level of overconfidence as the **Vanilla** method. And using dropout inference (**MC Dropout** model) seems to help. The **Finetuned** model seems to be better calibrated than the **Vanilla** and **Dropout training** models. However, this makes the **Finetuned with MCD** model underconfident. This example shows that finding the optimal probability p for dropout inference heavily depends on how overconfident or underconfident is the base model.

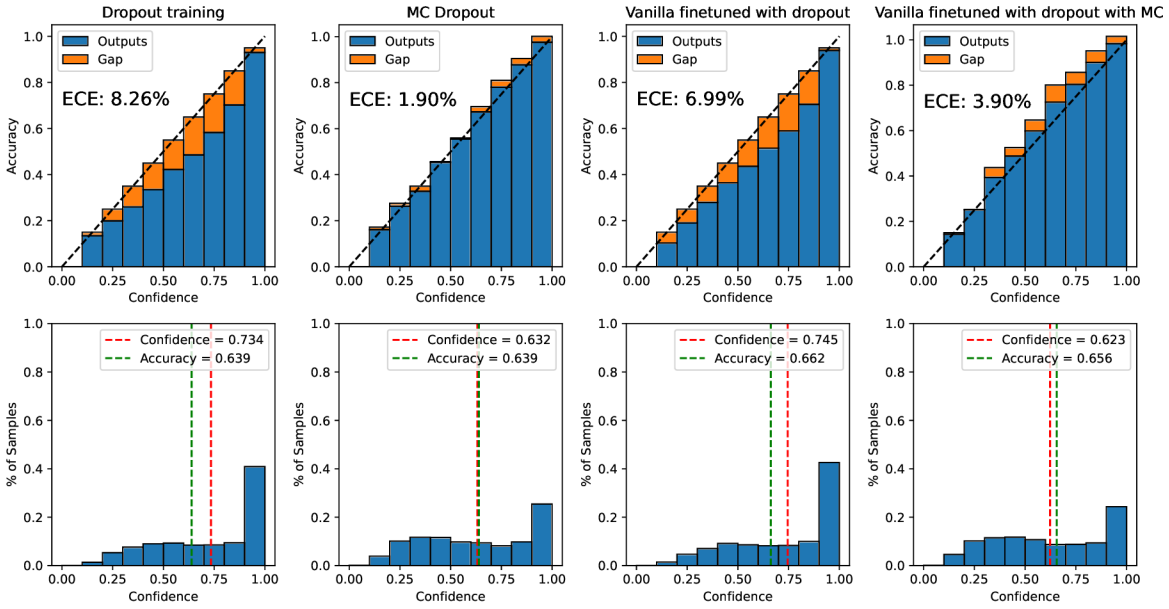


Figure 5.10: Calibration graphs on the standard CIFAR-100 test set for the ResNet-18 **Dropout training**, **MC Dropout**, **Vanilla finetuned with dropout** and **Vanilla finetuned with dropout with MCD** models.

The calibration graph for MobileNetV2 shown in Figure 5.11 shows very similar results as for the ResNet-18 network. The **Vanilla** model is overconfident, while **Dropout inference** seems to correct this issue. **Temperature scaling** and **Ensemble** methods seem to do similarly well as with the ResNet-18 network.

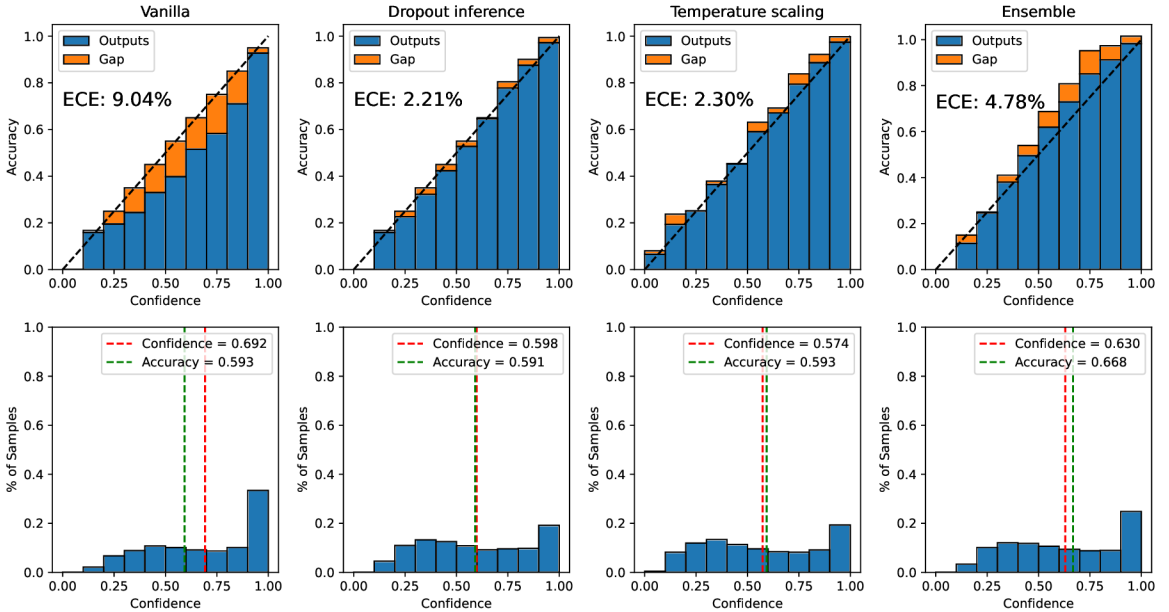


Figure 5.11: Calibration graphs on the standard CIFAR-100 test set for the MobileNetV2 **Vanilla**, **Dropout inference**, **Temperature scaling** and **Ensemble** models.

Figure 5.12 shows the calibration graph for methods that use dropout during training. The results are almost the same as when using ResNet-18 in Figure 5.10. Similarly as with ResNet-18, using dropout for training yields better accuracy than the **Vanilla** model. Interestingly, it does not seem to have much effect on ECE. Unlike the results from using the ResNet-18 network, training **Vanilla** model and then adding dropout and finetuning it for 10 epochs (**Finetuned** model) does not improve the accuracy of the MobileNetV2 model and only improves the ECE slightly. When using dropout inference the ECE actually increased. This probably an indication that the MobileNetV2 network is more sensitive to dropout. Maybe because it is smaller than the ResNet-18 network.

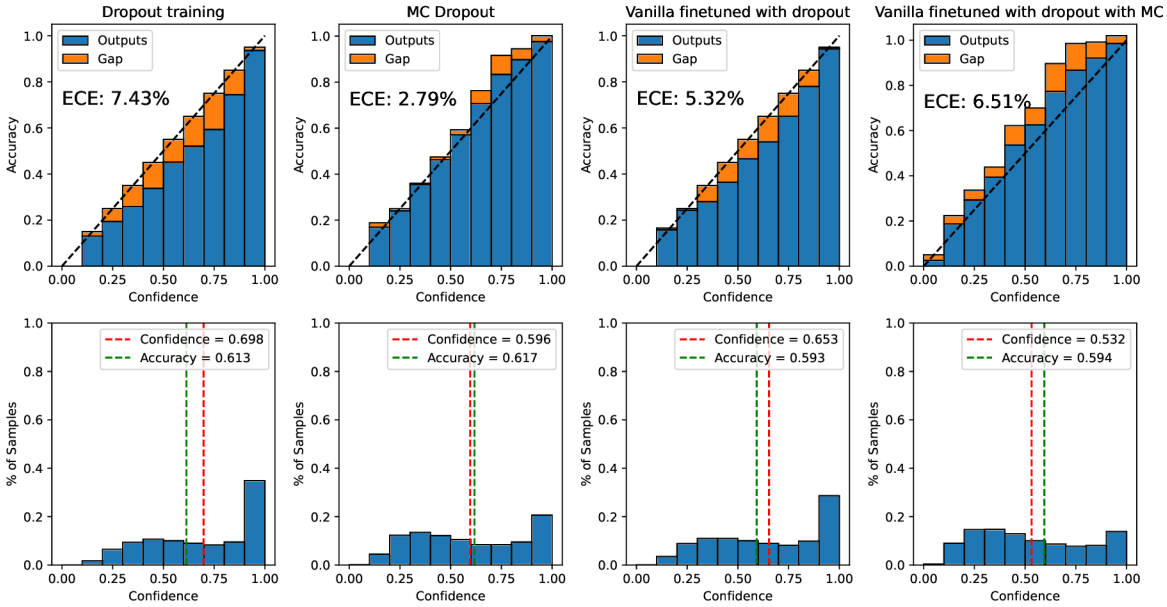


Figure 5.12: Calibration graphs on the standard CIFAR-100 test set for the MobileNetV2 **Dropout training**, **MC Dropout**, **Vanilla finetuned with dropout** and **Vanilla finetuned with dropout with MCD** models.

Overall, we can see that the results are almost the same for both of the networks used. **Ensemble** models achieve the best **Accuracy** and very good ECE. Using dropout for training seems to improve **Accuracy** considerably, but only slightly improves ECE. It is interesting that the **Finetuned** model performs very well with the ResNet-18 network but not with the MobileNetV2 network. This shows that not all architectures respond to dropout the same.

5.2.2 Results on Test Set with Random Additive Gaussian Noise

In this section, the models are evaluated using the same methodology as with the FMNIST dataset in Section 5.1.3. That means increasing the *std* parameter from 0 to 0.6 using 0.05 steps. Figure 5.13 shows examples of the CIFAR-100 transformed with additive Gaussian noise with $std = 0.5$.

Figure 5.13: Examples of the CIFAR-100 dataset transformed with additive Gaussian noise with $std = 0.5$.

Figure 5.14 shows the results of ResNet-18 network on test set with added gaussian noise. While the **Ensemble** model has the best accuracy and Brier score on the standard test set, the **Dropout training** and **MC Dropout** models achieve the best accuracy on samples with more noise. Using dropout inference also seems to improve the calibration on noisier samples. Models trained with dropout also seem to achieve higher **AUROC** and **AUPR** on noisier data.

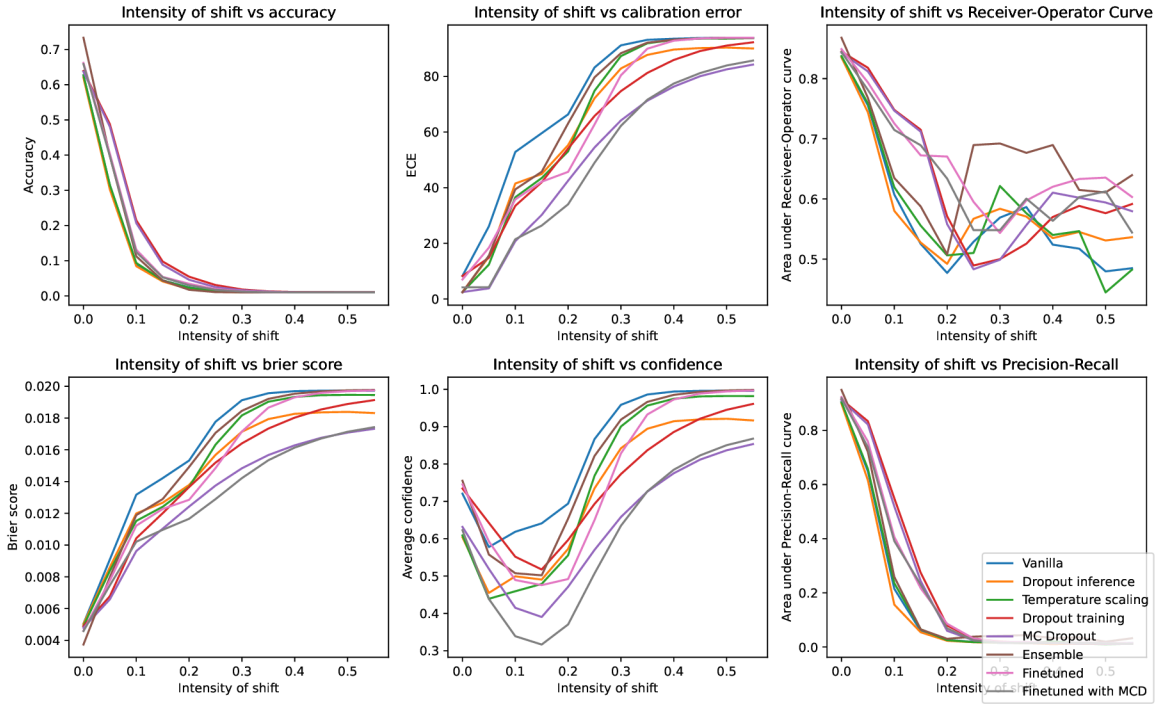


Figure 5.14: Graph of evaluation metrics on the CIFAR-100 test set with additive gaussian noise with increasing parameter *std*, using the ResNet-18 network.

Figure 5.15 shows the evaluation results while using the MobileNetV2 network. The **Accuracy** on noisier samples is almost the same in all the methods. Interestingly, the **Ensemble**, **Dropout training** and **MC Dropout** models achieve a worse ECE than even the **Vanilla** model. It is not clear why.

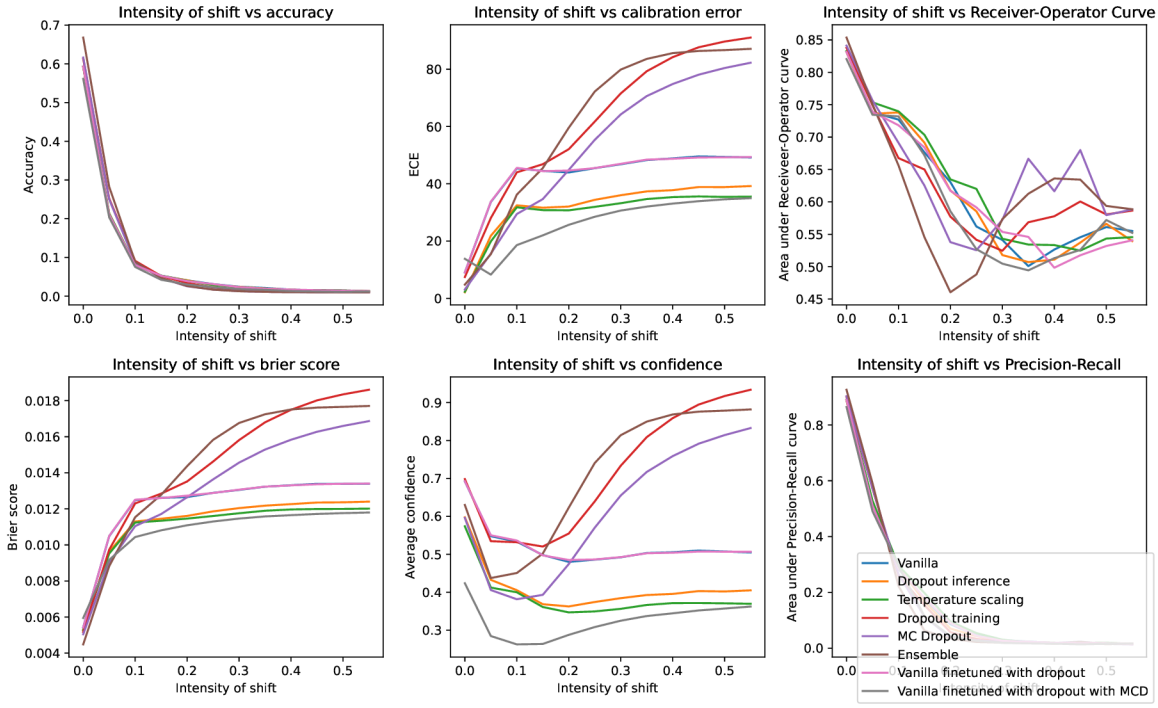


Figure 5.15: Graph of evaluation metrics on the CIFAR-100 test set with additive gaussian noise with increasing parameter *std*, using the MobileNetV2 network.

While the **Ensemble** methods achieve the best **Accuracy** for both networks and achieve relatively good calibration, it does not seem to improve calibration on noisier samples very much. It is also interesting that while using dropout training helps with calibration on noisier samples on the ResNet-18 network, it does not have the same effect on the MobileNetV2 network.

5.2.3 Results on Test Set with Color Jitter

In this section, the models are evaluated on test set with applied color jitter to it. Color jitter randomly changes the brightness, contrast, and saturation of the image. While changing contrast or saturation usually does not make the image unrecognizable, changing the brightness too much can make it unrecognizable (mountain image in figure 5.16). This is why here, the brightness is changed only half as much as contrast and saturation (strength 1.0 means strength 0.5 for brightness and strength 1.0 for contrast and saturation). Figure 5.16 shows examples of the CIFAR-100 dataset transformed with color jitter using $strength = 1.0$.

Figure 5.16: Examples of the CIFAR-100 dataset with color jitter using $strength = 1.0$.

The results of the ResNet-18 network models are shown in Figure 5.17. **Ensemble** achieves the best accuracy even on samples with a lot of jitter. **Finetuned** and **Dropout training** also achieve better accuracy results than the **Vanilla** model. However, the margin between model accuracies stays the same for every point of **Accuracy** chart. This means that technically **Ensemble** and **Finetuned** models achieve better **Accuracy** on shifted data, but only because they also achieve better **Accuracy** on the standard test set and not because the models would be more immune to dataset shift. **Vanilla**, **Dropout training** and **Finetuned** models are all outperformed by other models in terms of ECE. Especially the **Finetuned with MCD** model achieves good ECE even on very shifted samples.

Figure 5.18 shows results for the MobileNetV2 network. Similarly as with the ResNet-18 network, **Ensemble** achieves the best accuracy across all the shifted datasets. However, the **Finetuned** model achieves substantially worse accuracy than the **Finetuned** Resnet-18 model. This probably means that the MobileNetV2 network cannot be finetuned with adding dropout layers as well as the ResNet-18 network. Other metrics show similiar results as with the ResNet-18 network.

Experiments on both networks show that **Ensemble** models achieve the best accuracy, while also being reasonably well calibrated. And while using **Dropout training** can help achieve better accuracy, it does not seem to improve calibration. Using **Dropout training** or **Ensemble** in conjunction with **Dropout inference** (resulting in **MC Dropout**) or **Temperature scaling** can improve both accuracy and calibration.

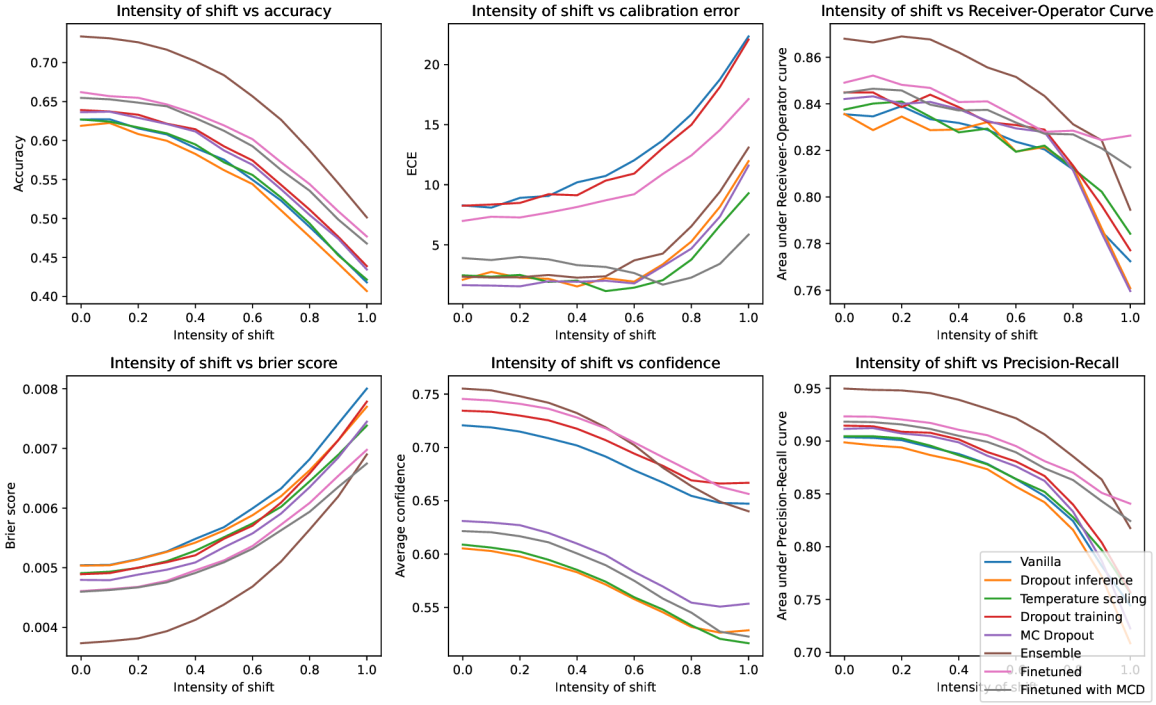


Figure 5.17: Graph of evaluation metrics on the CIFAR-100 test set with color jitter with increasing strength, using the ResNet-18 network.

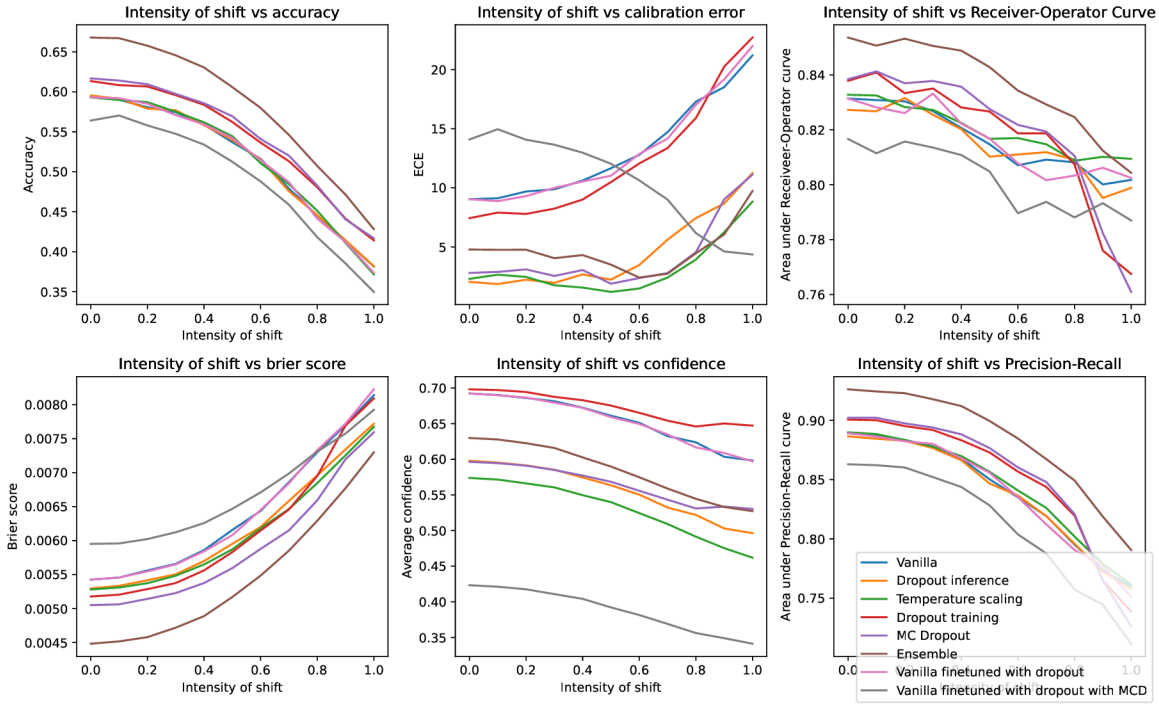


Figure 5.18: Graph of evaluation metrics on the CIFAR-100 test set with color jitter with increasing strength, using the MobileNetV2 network.

5.2.4 Results on Test Set with Gaussian Blur

This section focuses on evaluating the methods on data set with Gaussian blur applied. The Gaussian blur transformation expects `kernel_size` and `sigma` parameters. The parameter `kernel_size` sets the size of the Gaussian kernel, and the parameter `sigma` sets the standard deviation of the Gaussian blur kernel. The `kernel_size` parameter was experimentally chosen to be 5 and `sigma` is being increased from 0.1 to 1.4 using 0.1 steps. Figure 5.19 shows examples of the CIFAR-100 dataset transformed with Gaussian blur using `kernel_size` of 5 and $strength = 0.8$.

Figure 5.19: Examples of the CIFAR-100 dataset with gaussian blur using $strength = 0.8$.

Figure 5.20 shows results for the ResNet-18 network. Although the **Ensemble** model achieves the best accuracy on the standard test set, models that were trained with dropout outperform the **Ensemble** on samples with stronger blur. **MC Dropout** and **Finetuned with MCD** models achieve the best calibration on noisier samples. While the **Temperature scaling** and **Dropout inference** models are well calibrated on standard test set, their ECE increases rapidly on blurier samples.

The results in Figure 5.21, which shows the results for the MobileNetV2 network, are similar to the results from the ResNet-18 models. **Ensemble** achieves best accuracy on

the standard test set, but is outperformed on blurrier images by the **Dropout training** and **MC Dropout** models. **Finetuned with MCD** actually has the best calibration on more blurred images, but this is probably because the model uses too high dropout probability p and is therefore always underconfident. Other than that, the **MC Dropout** and **Ensemble** models are well calibrated across all shift strengths.

On both networks we can see that while **Ensemble** models achieve best accuracy on standard test set however, **Dropout training** and models based on it outperform **Ensembles** on noisier data in both accuracy and calibration.

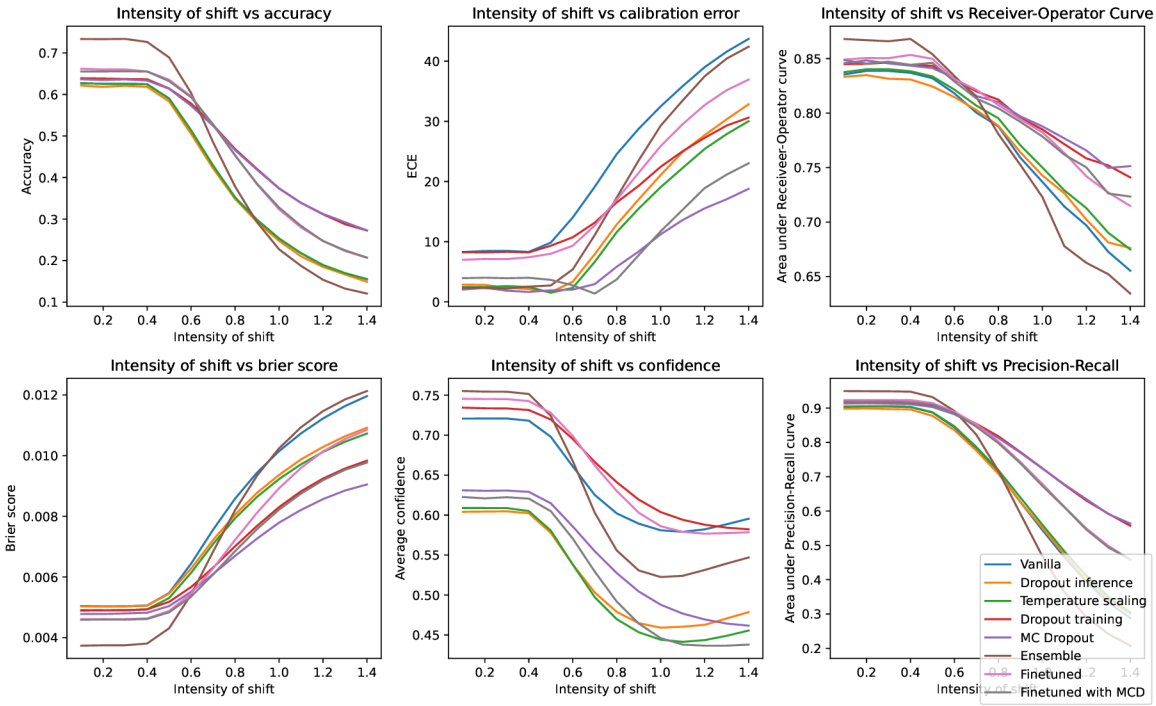


Figure 5.20: Graph of evaluation metrics on the CIFAR-100 test set with Gaussian blur with increasing strength, using the ResNet-18 network.

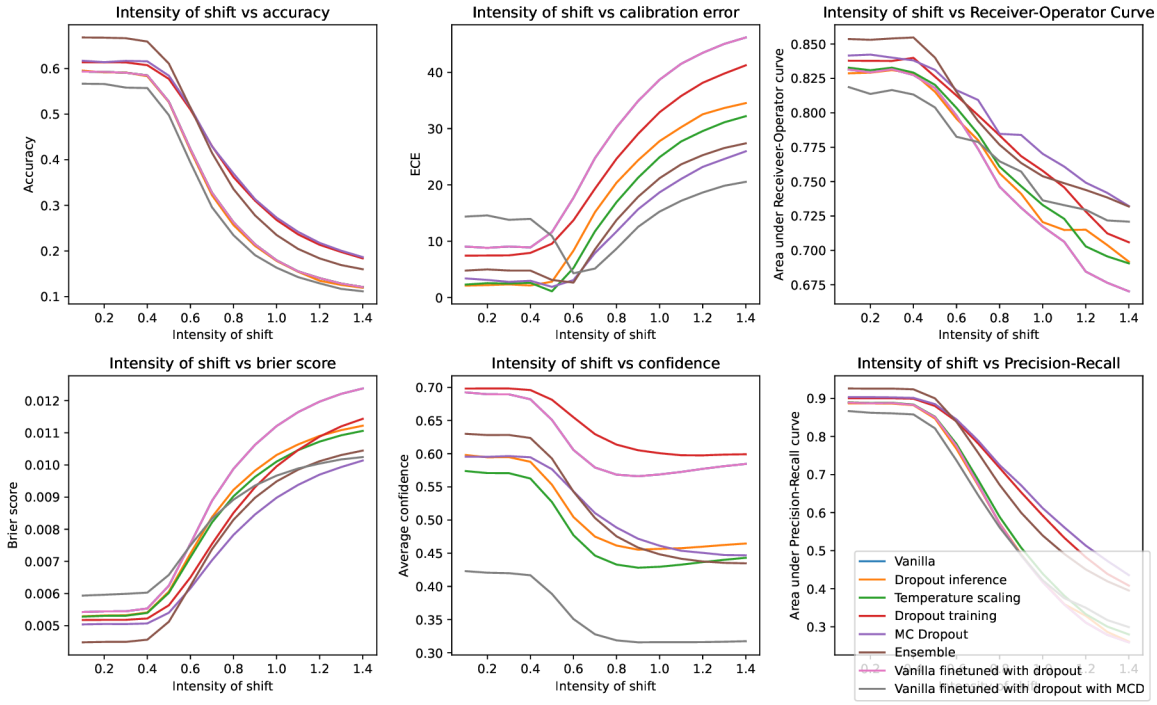


Figure 5.21: Graph of evaluation metrics on the CIFAR-100 test set with Gaussian blur with increasing strength, using the MobileNetV2 network.

5.2.5 Findings

Ensemble models perform the best across all the shift strengths while using color jitter but are outperformed on noisier data while using additive gaussian noise and gaussian blur by models trained with dropout. Using **Dropout inference** or **Temperature scaling** achieves better calibration even on noisier data regardless of the transformation used. However, **Dropout inference** is more complex to set up than **Temperature scaling** and slows down inference depending on how many forward passes are used. This possibly makes a case for training an **Ensemble** model with dropout to achieve the best possible accuracy and resistance to transformations and then using **Temperature scaling** to ensure good calibration.

5.3 Semantic Segmentation on the MedSeg Covid Dataset

In this section, the models are evaluated on the MedSeg Covid Dataset [25]. It is a medical dataset consisting of 100 CT slices of more than 40 patients with COVID-19. The images were manually segmented by a radiologist into 3 foreground labels: 1 = ground-glass, 2 = consolidation and 3 = pleural effusion. The remaining areas are classified as 0 = background. This is a semantic segmentation dataset, meaning that the model classifies each input image pixel into 3 foreground classes or the background class. This dataset was chosen because it was previously solved with a U-Net in [33]. The CT slices are compiled into NIFTI files[3]. Examples of the dataset are shown in figure 5.22.

[3]This work uses images exported into `.jpg` files taken from https://github.com/adnan-saood/COVID19-DL

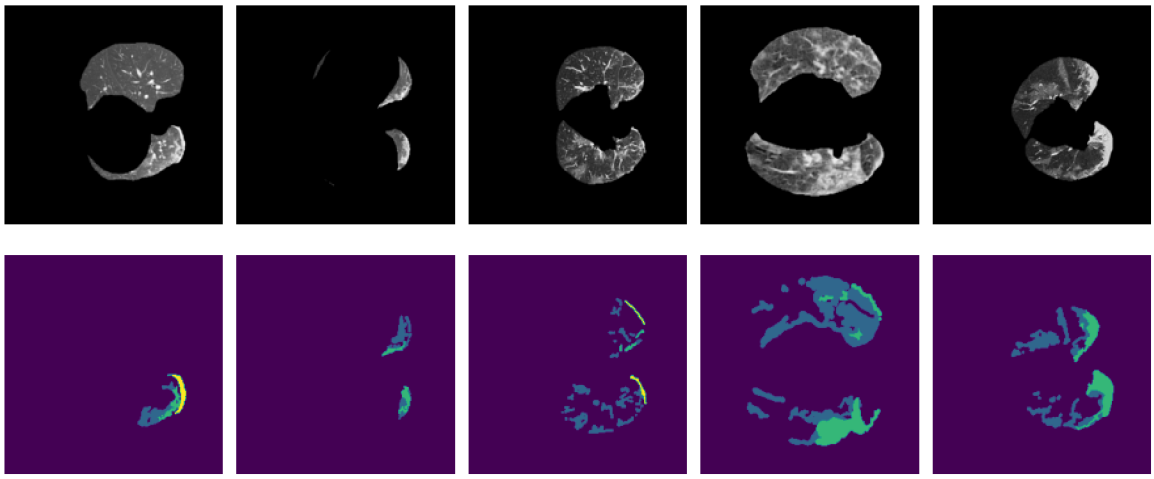


Figure 5.22: Examples of the Covid19 dataset. Input images are at the top, ground truth annotation in the bottom row. Violet = background and healthy tissues, blue = ground-glass, green = consolidation and yellow = pleural effusion.

The 100 images were randomly split into 60 images used for training, 30 for testing and 10 as a validation set for the **Temperature scaling** method. The split was performed using the `torch.utils.data.random_split` method using manual seed with value 0.

Neural network architecture used for this experiment is a standard U-Net[4] [31]. U-Net was chosen because it is small and easy to train, while achieving good results on medical datasets.

Every model was trained for 200 epochs using the Adam optimizer while saving checkpoints with the highest Intersection Over Union. The five models evaluated are as follows:

- **Vanilla** is a standard U-Net.
- **Temperature scaling** uses the same weights and architecture as **Vanilla** model, but the model logits are postprocessed using the temperature scaling technique.
- **Dropout training** is a U-Net network with additional dropout layers. `Dropout2d` layers with $p = 0.1$ are added after each `ReLU` activation.
- **MC Dropout** uses the same architecture and weights as the **Dropout training** model while also using dropout inference.
- **Ensemble** is an ensemble of 5 U-Nets.

5.3.1 Results on Standard Test Set

The results on the standard test set are shown in table 5.5. We can see that the **Dropout training**, **MC Dropout**, and **Ensemble** models achieve similar accuracy and outperform the **Vanilla** and **Temperature scaling** models by about 1%. These models also achieve better IOU and mIOU than the **Vanilla** and **Temperature scaling** models. All of these models are relatively well calibrated and achieve similar ECE. Their AUROC and AUPR is also very similar.

[4]U-Net implementation was taken from https://github.com/milesial/Pytorch-UNet

Method	Accuracy	ECE	Brier	AUROC	AUPR	IOU	mIOU
Vanilla	0.957	**4.162**	0.015	0.960	0.998	0.449	0.489
Temperature scaling	0.957	4.365	0.016	0.960	0.998	0.449	0.489
Dropout training	0.965	4.522	**0.013**	**0.968**	**0.999**	**0.493**	0.506
MC Dropout	0.965	5.142	**0.013**	0.966	**0.999**	0.490	0.505
Ensemble	**0.966**	4.303	**0.013**	0.967	**0.999**	0.485	**0.518**

Table 5.5: U-Net results on the COVID-19 test set. ECE = Expected Calibration Error, Brier = Brier score, AUROC = Area Under Receiver Operating Characteristic curve, AUPR = Area Under Precision Recall curve, IOU = pixelwise Intersection Over Union (excluding background), mIOU = Mean class Intersection Over Union.

Figure 5.23 shows the calibration graph for the models used. All of the models are well calibrated and achieve very similar ECE. It is interesting, that all of the models are slightly underconfident. Usually, models tend to be overconfident. This is probably because the dataset is very small. In this case, it means that the **MC Dropout** method actually makes the model even more uncerconfident and therefore worsens its ECE.

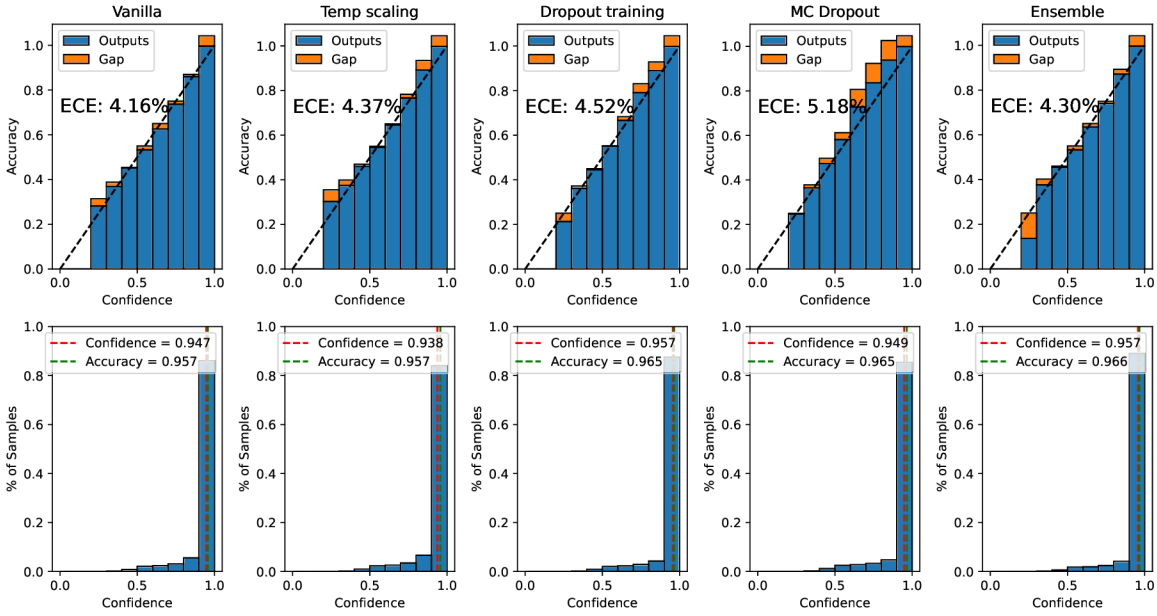


Figure 5.23: Calibration graph for the U-Net models on the COVID-19 dataset.

5.3.2 Results on Test Set with Additive Gaussian Noise

In this section, the models are evaluated on the test set transformed with additive Gaussian noise. The same methodology as in Sections 5.1.3 and 5.2.2 is used. The `std` parameter is increased from 0 to 0.6 using 0.05 steps. Figure 5.24 shows examples of the COVID-19 dataset transformed with additive Gaussian noise using $std = 0.2$.

Figure 5.25 shows the evaluation results with increasing strength of additive Gaussian noise. The **Ensemble** model achieves the best accuracy on the standard test set by a small margin but more importantly keeps good accuracy even on noisier data. Between strengths 0 and 0.1 the IOU and mean class IOU are very similar between the **Ensemble**, **Dropout**

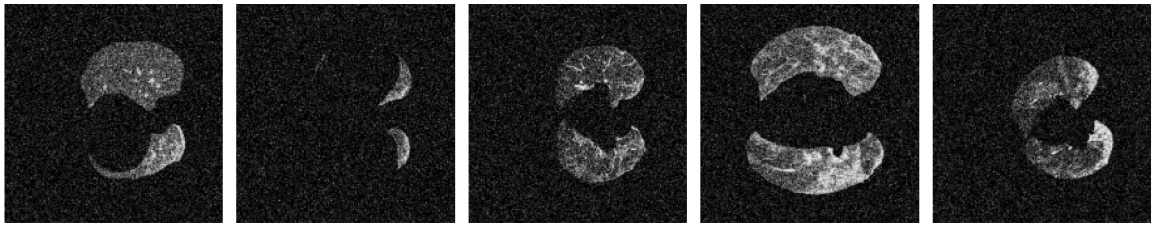


Figure 5.24: Examples of the COVID-19 dataset with additive Gaussian noise using $std = 0.2$.

training, and **MC Dropout** models, but on noisier data the **Ensemble** model achieves the best results by a huge margin. The **Dropout training** and **MC Dropout** models also show better resistance to noise than the **Vanilla** and **Temperature scaling** models but not to as high of a degree.

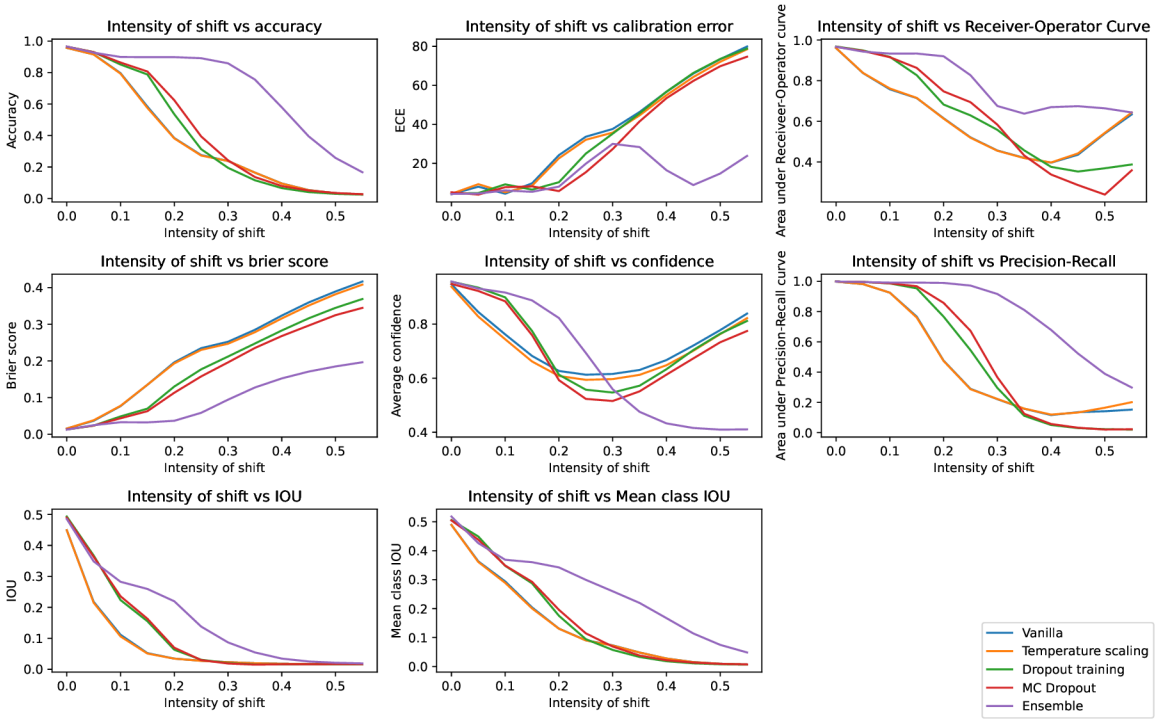


Figure 5.25: Evaluation results for the U-Net architecture on the COVID-19 test set with additive Gaussian noise.

5.3.3 Findings

The results show that even though there is only a small benefit for using **Ensemble**, **Dropout training** and **MC Dropout** on the standard test set. However, the **Ensemble** model performs much better than the other models on noisier data. This huge performance gap may be attributed to the independent learning of each component of the ensemble. In this way, each component can learn to segment pixels based on a different set of characteristics, which can lead to better generalization. This is also probably because the dataset is very small.

5.4 Semantic Segmentation on the PASCAL-VOC Segmentation Dataset

This section focuses on evaluation of the models on the PASCAL-VOC 2012 [7] segmentation dataset. The dataset consists of 20 object classes in realistic scenes. Since this is a semantic segmentation dataset, the goal of the model is to classify each image pixel into 20 object classes or the background class. The train set is used for training, and the val set is used for evaluation. In total, 1464 images are used for training and 1449 images are used for evaluation. Pytorch implementation of this dataset is used. The 20 foreground classes are grouped into 4 categories:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

Examples of the PASCAL-VOC segmentation dataset are shown in Figure 5.26.

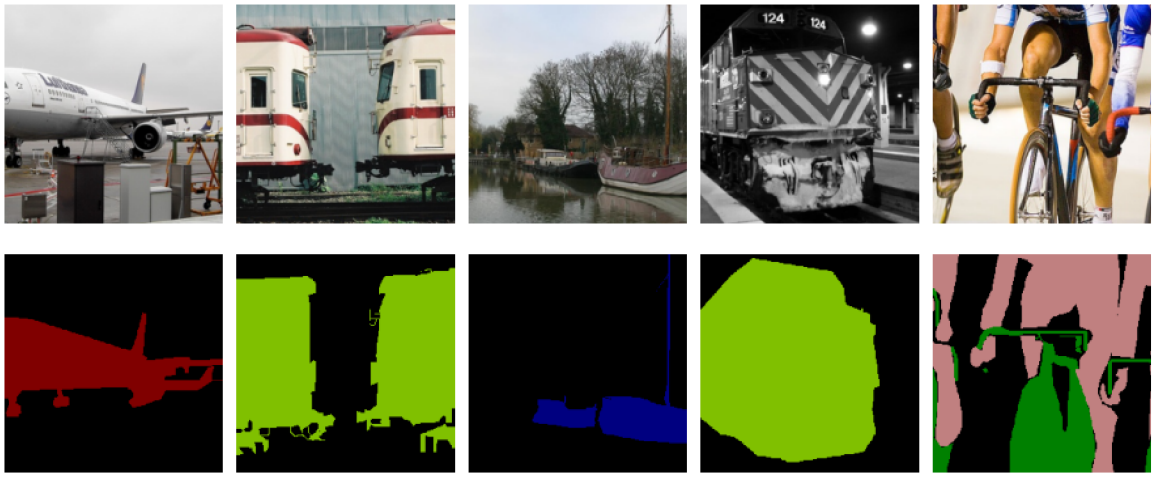


Figure 5.26: Examples of the PASCAL-VOC segmentation dataset. Input images are at the top, ground truth annotations are in the bottom row.

For evaluation, the DeepLabV3 [4] architecture with ResNet-50 backbone is used. The models are pre-trained on subset of the MS COCO dataset which shares the same classes as the PASCAL-VOC dataset used. The pretrained Pytorch implementation is used. The models are then trained for 20 epochs using the Adam optimizer. While training, the model backbone is frozen and only the classifier part of the network is trained. Only training the classifier yielded significantly better results than also training the backbone.

Three uncertainty estimation methods are evaluated:

- **Vanilla** is a Standard ResNet-50 DeeplabV3 model.
- **Dropout training** ResNet-50 DeeplabV3 model with added dropout layers before finetuning on the PASCAL-VOC dataset for 20 epochs. The model uses `Dropout2d`

layers with $p = 0.2$ inserted after the last 2 `Bottleneck` blocks of the backbone. This architecture was chosen experimentally. For example, adding dropout layers to the classifier part of the model did not have a huge effect.

- **MC Dropout** uses the same architecture and weights as the **Dropout training** model while also using dropout inference.

5.4.1 Results on Standard Test Set

Table 5.6 shows result on the standard test set. The **MC Dropout** model achieves the best accuracy and IOU by a small margin, but surprisingly achieves the worst mIOU. The IOU is computed only from foreground classes, while the mIOU is computed from all classes, including background. This probably means that the **MC Dropout** model prioritizes foreground classes more. The **MC Dropout** model also achieves the best AUROC and AUPR but also has the worst ECE. However, all of the models are reasonably well calibrated. Brier score is not computed in this section due to memory limitations.

Method	Accuracy	ECE	AUROC	AUPR	IOU	mIOU
Vanilla	0.894	**2.299**	0.878	0.983	0.692	0.647
Dropout training	0.894	2.546	0.882	0.984	0.698	**0.652**
MC Dropout	**0.899**	4.034	**0.883**	**0.985**	**0.699**	0.644

Table 5.6: DeepLabV3 results on the PASCAL-VOC segmentation test set. ECE = Expected Calibration Error, AUROC = Area Under Receiver Operating Characteristic curve, AUPR = Area Under Precision Recall curve, IOU = pixelwise Intersection Over Union (excluding background), mIOU = Mean class Intersection Over Union.

The calibration graph for the models is shown in Figure 5.27. While the **Vanilla** and **Dropout training** models achieve very good calibration but are very slightly overconfident, the **MC Dropout** model is underconfident and has worse ECE. For the standard test set, fewer dropout layers or a lower probability p would probably yield better ECE. However, as we have seen in Section 5.2, more dropout and being underconfident on the standard test set may yield a better calibration on noisier data.

While **MC Dropout** achieves marginally better accuracy and IOU, the difference between models is too small to justify using Monte Carlo Dropout in production, if the test set is close to the training set and if the **Vanilla** model is already well calibrated.

5.4.2 Results on Test Set with Additive Gaussian Noise

In this section, the models are evaluated on the test set shifted with additive Gaussian noise. Same methodology as in previous sections is used. The `std` parameter is increased from 0 to 0.6 using 0.05 steps. Examples of the dataset transformed with additive Gaussian noise with $std = 0.2$ are shown in Figure 5.28.

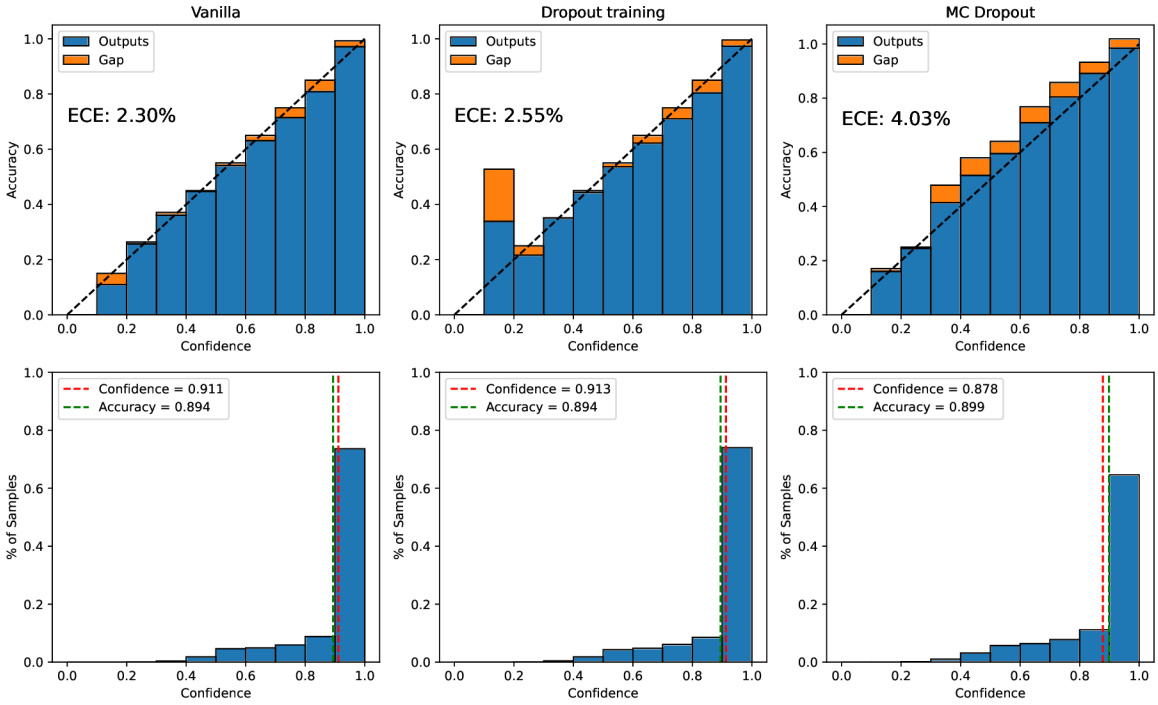


Figure 5.27: Calibration graph for the Pascal-VOC segmentation dataset using the DeepLabV3 network.

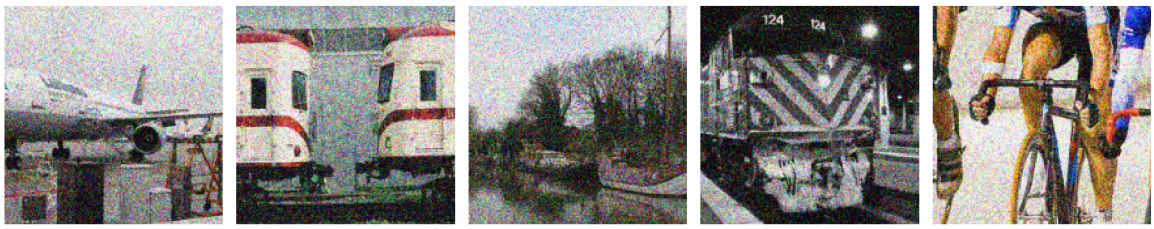


Figure 5.28: Examples of the PASCAL-VOC segmentaion dataset transformed with additive Gaussian noise using $std = 0.2$.

Figure 5.29 shows evaluation results while increasing the random Gaussian noise standard deviation. The IOU chart is not shown because the results are very similar to the mIOU chart. While the accuracies are very similar on the standard test, the **Dropout training** and **MC Dropout** models achieve better accuracy on noisier samples. Despite this, the mIOU stays very close between all the models. The **Dropout training** and **MC Dropout** models also achieve better AUROC and AUPR on noisier data. While the **MC Dropout** model achieves the best ECE for moderate amount of noise, the **Vanilla** model has the lowest ECE for data with a lot of noise.

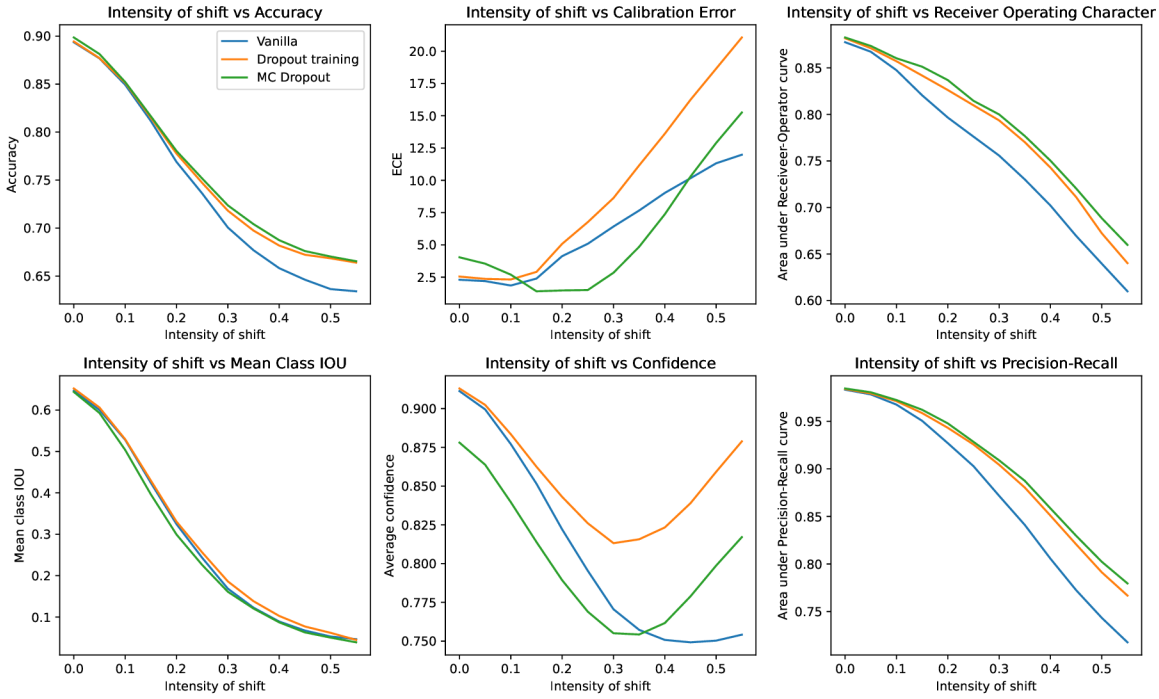


Figure 5.29: Evaluation results on the PASCAL-VOC detection val set with additive Gaussian noise using the DeepLabV3 network. The `std` parameter is increased from 0 to 0.6 using 0.05 steps.

5.4.3 Findings

We can see that adding dropout even to a pre-trained model can yield better results on out of distribution data. However, there seems to be little merit for also using dropout inference. The only metric where the **MC Dropout** model achieves significantly better results than the **Dropout training** model is ECE and only on samples with moderate amount of noise. As the samples become noisier, the ECE increases the same for both **MC Dropout** and **Dropout training** models.

5.5 Object Detection on the PASCAL-VOC Detection Dataset

In this section, the models are evaluated on the PASCAL-VOC [7] detection dataset. The foreground classes are the same as in the PASCAL-VOC segmentation dataset used in section 5.4. However, since this is a detection dataset, the model predicts bounding boxes for each object in the image. The 2012 trainval and 2007 trainval sets are used for training and 2007 test set is used for evaluation. Examples of the PASCAL-VOC detection dataset are shown in Figure 5.30.

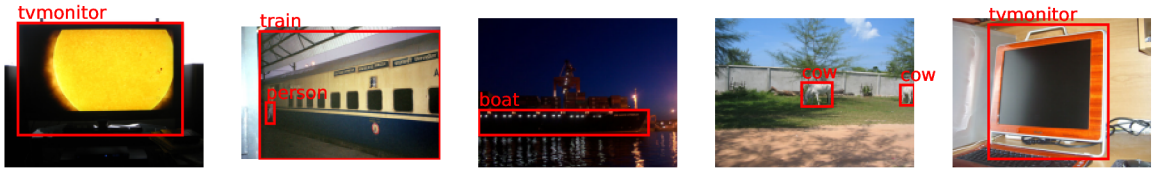


Figure 5.30: Examples of the PASCAL-VOC detection dataset.

The SSD300[5] [23] was chosen as the network architecture. The model architecture is relatively simple compared to another object detectors and allows for straightforward addition of dropout layers. This model was also used with Monte Carlo Dropout in [26].

The training procedure for the **Vanilla** and **Dropout training** models follows the original paper. The models were trained for 231 epochs using a batch size of 8. Stochastic Gradient Descent was used as optimizer with initial learning rate of $1e-3$, momentum of 0.9, and $5e-4$ weight decay. Also, a number of training data augmentations were used:

- Color Jitter with 50% chance.
- Zoom out with 50% chance.
- Random crop.
- Horizontal flip with 50% chance.

The models were evaluated using non-max suppression with IOU threshold of 0.5 and score threshold of 0.2. This setting had much lower number of false positives compared to score threshold of 0.01, which was used for evaluation in the original SSD paper, while still achieving good mAP. In total, 3 versions of models were evaluated:

- **Vanilla** is a standard SSD300 model.
- **Dropout training** SSD300 model with added `Dropout2d` layers with $p = 0.3$ after last two layers of the VGG16 backbone. This follows the same placement of dropout layers as in [26].
- **MC Dropout** uses the same architecture and weights as **Dropout training** model while also using dropout inference.

5.5.1 Dropout Inference Techniques for Object Detection Models

There is a number of approaches to dropout inference with object detection models. In [26] the model is treated as a black box and all of the bounding boxes from multiple forward passes are clustered based on their IOU. The whole procedure is as follows:

1. Make N forward passes for each input image.
2. Bounding box predictions for multiple forward passes are grouped into `Observations`. Bounding boxes from multiple forward passes are grouped if their predicted label is the same and their IOU is bigger than 0.9. If a bounding box cannot be added to an existing `Observation`, a new `Observation` is created.

[5]Implementation of the SSD300 model and detection utilities was taken from https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection.

3. Only use observations with more than D bounding boxes.

4. Compute the mean of confidences and locations of all bounding boxes in an observation.

However, this approach did not provide good results. Either having a large number of false positives (when using $D = 1$, as was done in [26]), or having a large number of false negatives (when using larger D).

Since this approach did not work very well, I simply averaged the model outputs before the post-processing step that detects the bounding boxes.

5.5.2 Results on Standard Test Set

Table 5.7 shows evaluation results on the PASCAL-VOC 2007 test set. **Vanilla** model achieves the best mAP when using IOU threshold 50 but not with IOU threshold 75. In this case, dropout inference may very slightly improve bounding box localization performance. The **Dropout training** model detects the most true positives and least false negatives, but in turn also detects the most false positives. All of the metrics are very similar for all of the models and there seems to be no reason to prefer **Dropout training** or **MC Dropout** models over the **Vanilla** model on the standard test set.

Method	mAP 50	mAP 75	AUROC	AUPR	TP	FP	FN
Vanilla	**0.732**	0.520	**0.879**	**0.934**	10463	**5904**	4513
Dropout training	0.726	0.520	0.876	0.928	**10467**	6300	**4509**
MC Dropout	0.727	**0.521**	0.875	0.930	10412	5981	4564

Table 5.7: SSD300 results on the PASCAL-VOC 2012 segmentation val set. mAP 50 = mean Average Precision at 50 IOU, mAP 75 = mean Average Precision at 75 IOU, AUROC = Area Under Receiver Operating Characteristic curve, AUPR = Area Under Precision Recall curve, TP = number of True Positives, FP = number of False Positives, FN = number of False Negatives. All of the metrics apart from mAP 75 are computed at IOU threshold 50 and score threshold 0.2.

5.5.3 Results on Test Set with Additive Gaussian Noise

Models in this section were evaluated on the PASCAL-VOC 2007 test set transformed with random additive Gaussian noise. This transformation was chosen because it was not used for training. The models performed very well even on very noisy data when using the color jitter transformation because the models were trained with it.

Figure 5.31 shows the evaluation results on the test set with increasing levels of noise. The models achieve very similar results to each other in most of the metrics. However, on noisier samples, we can see that the **Dropout training** and **MC Dropout** models have a significantly lower number of false positives than the **Vanilla** model. The **Dropout training** and **MC Dropout** models also achieve better AUROC with noisier samples. However, the **Dropout training** and **MC Dropout** models also have a lower number of true positives and a higher number of false negatives.

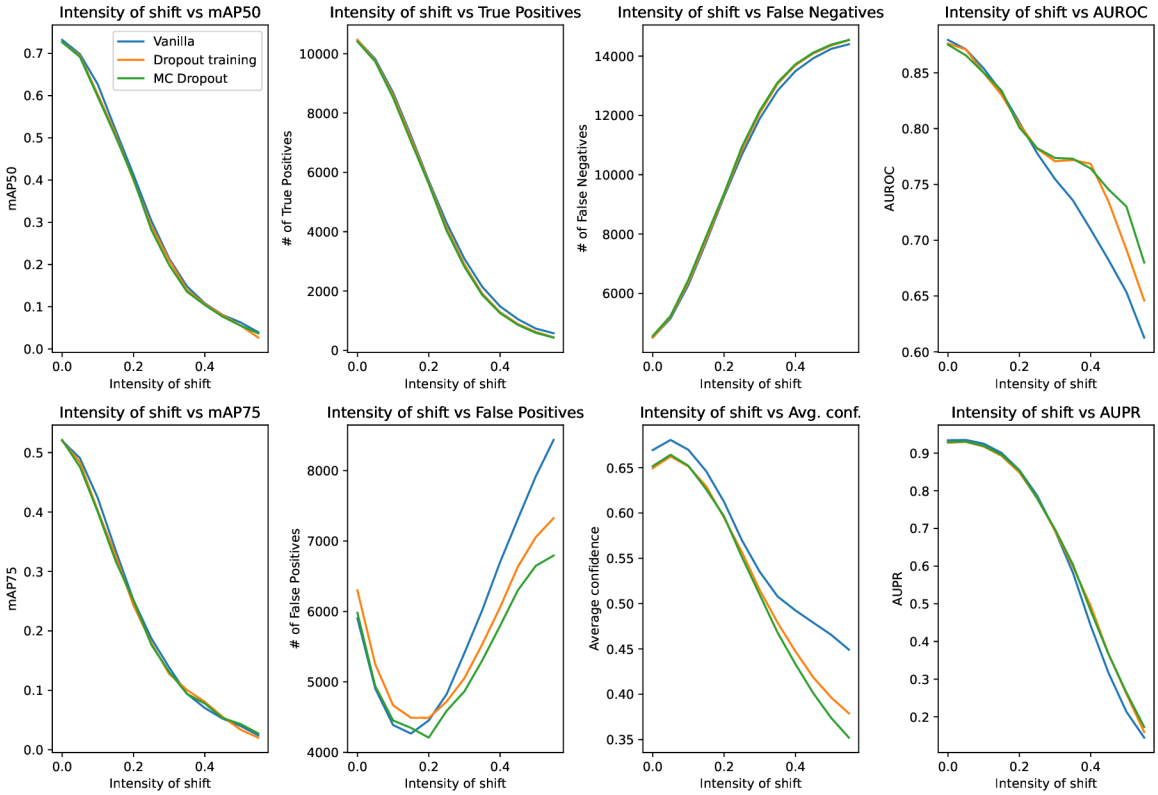


Figure 5.31: Evaluation results on the PASCAL-VOC detection test set with additive Gaussian noise. The `std` parameter is increased from 0 to 0.6 using 0.05 steps.

5.5.4 Findings

The results show that, while employing **Dropout training** or **MC Dropout** can lead to a lower number of false positives and a higher AUROC on noisier predictions. It also reduces the number of true positives and introduces more false negatives. Therefore, the pros are not large enough to outweigh the cons and to recommend the use of **Dropout training** or **MC Dropout** in this setting.

5.6 Practical Recommendations

This section summarizes practical recommendations based on the results of the experiments.

- **Ensembles** achieved the best predictive performance on all of the standard test sets. But on transformed test sets they can be outperformed by models trained with dropout.
- **Ensembles** dominated the COVID-19 dataset. This dataset has much fewer samples than others. **Ensembles** may offer the biggest benefits compared to other methods when used on very small datasets.
- **Temperature scaling** can be very effective in reducing the ECE if the base model has bad calibration. However, it is not very effective on already well calibrated models and does not improve the model‘s resistance to dataset shift. Also, while it improves the ECE it does not improve AUROC or AUPR.

- **Dropout training** improved predictive performance for all of the models except for the SSD300. It also had the best resistance to dataset shift on most of the transformations. This is surprising since nowadays, dropout is not used very often. However, choosing the placement and the probability p of the dropout layers is very important. Also, the strategy for adding dropout layers heavily depends on the model architecture.

- **Dropout inference** alone can improve the models‘ calibration if the base model is overconfident. However, if the base model is well calibrated, dropout inference can make the model underconfident. Moreover, while it can improve calibration, it does not improve AUROC or AUPR.

- **MC Dropout** models have improved predictive performance and uncertainty estimation quality. But this stems from the base model being trained with dropout layers and not from dropout inference.

- Adding dropout layers to a pretrained model and finetuning it may or may not improve the predictive performance and uncertainty estimation quality. When used on the ResNet-18 network, improved the prediction accuracy by almost 4% and improved the uncertainty estimation quality and resistance to dataset shift, even surpassing the **Dropout training** model. However, on the MobilNetV2 network, it had virtually no effect.

- The resistance to dataset shift depends more on the base model used that on the uncertainty estimation method.

- The experiments showed mostly positive effects of **Dropout training** and **Ensembles** on the predictive performance and uncertainty estimation quality. It would be interesting to combine several models, trained with dropout layers into one ensemble. To improve the calibration, **Temperature scaling** offers the benefits of the **Dropout inference** without introducing additional computing costs.

Chapter 6

Conclusion

This work focused on common uncertainty estimation methods. Modifications to widely used convolutional neural network architectures were proposed to employ these methods. A set of experiments for each computer vision task was created for evaluation of model performance on in distribution as well as out of distribution data. The proposed modified models achieved significantly better results on both in-distribution as well as out-of-distribution data. Especially ensemble models achieved very good results on in-distribution data, while architectures with added dropout layers displayed better resistance to dataset shift.

In total, this work explored uncertainty estimations on six widely used model architectures ranging from small and simple to the current state of the art. Evaluation was done on 5 datasets for image classification, semantic segmentation, and object detection. The models were evaluated on standard test sets, as well as artificially distorted test sets with varying strength of distortion and different types of distortions. These experiments provide useful knowledge about the effectiveness of these methods on in-distribution as well as out-of-distribution data. To my knowledge, this work is also one of the first to touch on the subject of uncertainty estimation in deep object detectors, especially on shifted data.

Based on the results of the experiments conducted, practical recommendations were formed for employing these uncertainty estimation methods in practice.

This work could be further expanded by evaluating different strategies for adding dropout layers and employing Monte Carlo Dropout on existing models. Another direction for future research could be on new and emerging uncertainty estimation techniques that do not induce additional computational or memory costs.

Bibliography

[1] Brier, G. W. et al. Verification of forecasts expressed in terms of probability. *Monthly weather review.* 1950, vol. 78, no. 1, p. 1–3.

[2] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *ArXiv preprint arXiv:1412.7062.* 2014.

[3] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence.* IEEE. 2017, vol. 40, no. 4, p. 834–848.

[4] Chen, L.-C., Papandreou, G., Schroff, F. and Adam, H. Rethinking atrous convolution for semantic image segmentation. *ArXiv preprint arXiv:1706.05587.* 2017.

[5] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *Proceedings of the European conference on computer vision (ECCV).* 2018, p. 801–818.

[6] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 2009, p. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[7] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results* [http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html]. 2012.

[8] Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: PMLR. *International conference on machine learning.* 2016, p. 1050–1059.

[9] Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014, p. 580–587.

[10] Guo, C., Pleiss, G., Sun, Y. and Weinberger, K. Q. On calibration of modern neural networks. In: PMLR. *International Conference on Machine Learning.* 2017, p. 1321–1330.

[11] Gustafsson, F. K., Danelljan, M. and Schön, T. B. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).* 2020, p. 1289–1298.

[12] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, p. 770–778.

[13] Hendrycks, D. and Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *ArXiv.* 2017, abs/1610.02136.

[14] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B. et al. Searching for mobilenetv3. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, p. 1314–1324.

[15] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv preprint arXiv:1704.04861.* 2017.

[16] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017, p. 4700–4708.

[17] Jungo, A. and Reyes, M. Assessing reliability and challenges of uncertainty estimations for medical image segmentation. In: Springer. *International Conference on Medical Image Computing and Computer-Assisted Intervention.* 2019, p. 48–56.

[18] Kendall, A., Badrinarayanan, V. and Cipolla, R. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *ArXiv preprint arXiv:1511.02680.* 2015.

[19] Krizhevsky, A., Nair, V. and Hinton, G. CIFAR-100 (Canadian Institute for Advanced Research). Available at: http://www.cs.toronto.edu/~kriz/cifar.html.

[20] Lakshminarayanan, B., Pritzel, A. and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems.* 2017, vol. 30.

[21] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E. et al. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation.* 1989, vol. 1, no. 4, p. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[22] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P. et al. Microsoft coco: Common objects in context. In: Springer. *European conference on computer vision.* 2014, p. 740–755.

[23] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. et al. Ssd: Single shot multibox detector. In: Springer. *European conference on computer vision.* 2016, p. 21–37.

[24] Loquercio, A., Segu, M. and Scaramuzza, D. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*. IEEE. 2020, vol. 5, no. 2, p. 3153–3160.

[25] MedSeg, Jenssen, H. B. and Sakinis, T. MedSeg Covid Dataset 1. january 2021. DOI: 10.6084/m9.figshare.13521488.v2. Available at: https://figshare.com/articles/dataset/MedSeg_Covid_Dataset_1/13521488.

[26] Miller, D., Nicholson, L., Dayoub, F. and Sünderhauf, N. Dropout sampling for robust object detection in open-set conditions. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, p. 3243–3249.

[27] Naeini, M. P., Cooper, G. and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[28] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D. et al. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*. 2019, vol. 32.

[29] Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*. Cambridge, MA. 1999, vol. 10, no. 3, p. 61–74.

[30] Ren, S., He, K., Girshick, R. and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*. 2015, vol. 28.

[31] Ronneberger, O., Fischer, P. and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: Springer. *International Conference on Medical image computing and computer-assisted intervention*. 2015, p. 234–241.

[32] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 4510–4520.

[33] Saood, A. and Hatem, I. COVID-19 lung CT image segmentation using deep learning methods: U-Net versus SegNet. *BMC Medical Imaging*. Feb 2021, vol. 21, no. 1, p. 19. DOI: 10.1186/s12880-020-00529-5. ISSN 1471-2342. Available at: https://doi.org/10.1186/s12880-020-00529-5.

[34] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv preprint arXiv:1409.1556*. 2014.

[35] Xiao, H., Rasul, K. and Vollgraf, R. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017.

[36] Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K. Aggregated residual transformations for deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 1492–1500.

[37] Yang, T.-J., Howard, A., Chen, B., Zhang, X., Go, A. et al. Netadapt: Platform-aware neural network adaptation for mobile applications. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 285–300.

[38] Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *ArXiv preprint arXiv:1611.01578*. 2016.