university of groningen / faculty of science and engineering

Transfer Learning with Uncertain Features

Bachelor's Project Thesis

Samuele Milanese, s3725294, s.milanese@student.rug.nl,
Supervisors: Dr Matias Valdenegro Toro

Abstract: As Deep Learning is used in an increasing number of (sensitive) scientific fields and real-world applications, Transfer Learning is being used in those areas where acquiring data is expensive and/or difficult. Although Transfer Learning is a useful technique to counter scarcity of data, it carries DL problems such as over- or under-confidence and uncalibrated predictions in general. In standard settings, Uncertainty Quantification methods can be used to achieve safe and reliable models with calibrated confidence scores (Leibig et al., 2017). This research aims at using different UQ methods to extract uncertain features (with mean and variance) instead of point features (only mean) in order to carry the advantages of UQ to Transfer Learning. This is explored by building and comparing feature extractors in three different setups: different UQ methods applied to different architectures; models with only some of the layers implementing uncertainty quantification; uncertain features are sampled to generate new data points. The quality of the features is evaluated by feeding them to an SVM (Ho & Kim, 2021). The results of the experiments did not show any improvement in performance when using uncertain rather than point features, neither in terms of accuracy nor expected calibration error. However, generating new data through sampling uncertain features could be suggested as a valid Data Augmentation technique

1 Introduction

The field of Deep Learning (DL) is becoming more and more important in Machine Learning and is making its way into almost any other field of technology (Leibig et al., 2017; Gawlikowski et al., 2022). This is due to the astounding results a State of the Art technique can deliver in tasks such as classification, regression and prediction given enough data, no matter its origin. As a consequence of this huge interest and advancement, Neural Networks are now employed as a standard for plenty of tasks in a plethora of different scientific fields and real-world applications. Some of these can be very critical in terms of importance and safety (Dong et al., 2021).

A fitting example is Computer Vision. This is one of the fields where Deep Neural Networks thrived the most. Because of their efficiency, DNNs are also being used for critical applications such as medical imaging and diagnosis (Leibig et al., 2017). As Deep Learning is increasingly involved in such applications a new concern arises. This concern is the one of reliability and confidence. As Leibig et al. (2017) point out, these techniques have been applied without any risk management. In fact, the predictions of a regular DNN do not give any insight on how confident the DNN is about them. It follows that a particularly difficult case might end up being classified wrongly but with a high confidence (Valdenegro-Toro, 2021; Abdar et al., 2021). This is very different from how a human examiner would behave. Confronted with an edge case, a human examiner would refuse to give a prediction and suggest further analysis. In order to make Neural Networks more reliable and transparent, they should be able to do the same. This is the aim of Uncertainty Quantification methods.

However important reliability is, it is not the only reason why Uncertainty Quantification is a topic on which research should focus more. In fact, modeling uncertainty in a Neural Network can help with over- and under-confidence of DNNs. Furthermore, modeling uncertainty better reflects the real world, where data is unbalanced, noisy and might shift (Abdar et al., 2021; Gawlikowski et al., 2022). As

Gawlikowski et al. (2022) suggest, this can be useful in tasks where data is scarce as well.

The current trend in situations where a model needs to be trained on scarce data is to use Transfer Learning (Tan et al., 2018). This technique consists in pre-training a model on a rich and general dataset to extract features that may benefit subsequent training on the target dataset.

The aim of this research is to introduce Uncertainty Quantification in the Transfer Learning setting. Generally, models that implement Uncertainty Quantification assume their weights to be probability distributions instead of point features (Abdar et al., 2021; Mobiny et al., 2019; Gawlikowski et al., 2022). This would enable feature extractors to output probability distributions which are more informative than point values. Uncertain features might give a better insight on the true distribution of the features and could be sampled to obtain new data points under the assumed distribution. It is the objective of the research to assess whether these advantages can be carried to a target domain through feature extraction and (positively) affect performance.

Formally, this research will answer this question: Does extracting Uncertain instead of Point intermediate features of a CNN lead to better results in a Transfer Learning Image Classification Task in terms of Accuracy and Calibration?

Given the lack of previous research on this specific challenge, the present research is purely exploratory. It is trying to give initial insights on the results that could be obtained by using Uncertain features in Transfer Learning tasks. For this purpose, different Neural Network models with different Uncertainty Quantification methods will be built and used as feature extractors. Said features will be fed to a non-parametric Machine Learning model to evaluate Transfer Learning performance (Ho & Kim, 2021). Calibration will be the main ground of comparison as reliability is the main motivation of this research. Furthermore, also accuracy will be measured and compared, in order to establish whether producing uncertainty in the features might lead to a performance improvement in Transfer Learning Tasks.

In the following section the reader will be introduced to the broader background behind this research.

2 Background

2.1 Transfer Learning

An important problem in Supervised Learning is that of scarcity of data. This problem plagues many domains where acquiring data is expensive or just difficult (Zhuang et al., 2020).

Transfer Learning is one of the techniques employed to solve this problem. Its aim is defined by Zhuang et al. (2020) as using the knowledge acquired in a domain to improve or facilitate learning in a target domain.

A formal definition of the Transfer Learning task can be found in Tan et al. (2018):

"Given a learning task $\mathcal{T}_t$ based on $\mathcal{D}_t$, and we can get the help from $\mathcal{D}_s$ for the learning task $\mathcal{T}_s$. Transfer learning aims to improve the performance of predictive function $f_{\mathcal{T}}(\cdot)$ for learning task $\mathcal{T}_t$ by discover and transfer latent knowledge from $\mathcal{D}_s$ and $\mathcal{T}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$ and/or $\mathcal{T}_s \neq \mathcal{T}_t$. In addition, in the most case, the size of $\mathcal{D}_s$ is much larger than the size of $\mathcal{D}_t$, $N_s \gg N_t$."

This same survey gives an overview of how this task is carried out. There are multiple approaches to achieve Transfer Learning. The one used in this project is Network-based deep Transfer Learning. In this approach a first Network is trained on a general and rich dataset to extract useful features. In a second moment, a partial of this model, the *Feature Extractor*, is used as part of a second (usually smaller) model that will be trained and evaluated on the scarce dataset from the target domain.

A major example of how Transfer Learning is used in real-world application is the medical field. AI-aided Medical Imaging is a useful tool for diagnosis. However, medical images are generated by specialized equipment and labeling is carried out manually by specialists (Zhuang et al., 2020). For these reasons, data is usually scarce. Hence, Transfer Learning becomes vital.

Medical Imaging is also a good source of motivation for interest in Transfer Learning by the researchers in Uncertainty Quantification. As a matter of fact, Transfer Learning is a useful technique to cope with scarcity of data but still drags the same concerns of reliability and confidence of standard Neural Networks (Leibig et al., 2017)

2.2 Uncertainty Quantification

In order to address the concerns of reliability and confidence in critical fields, research is being conducted on Uncertainty Quantification methods.

The way these concerns are addressed is by using techniques to estimate the predictive uncertainty correctly. A model able to estimate uncertainty can accompany its predictions with reliable confidence scores, generally in the form of probabilities (Gawlikowski et al., 2022). In classification, the probabilities outputted by the Softmax activation in the last layer of standard CNNs could be taken as confidence scores. However, these are usually not well calibrated and thus might lead to overconfidence in making the wrong prediction (Valdenegro-Toro, 2021; Abdar et al., 2021).

Predictive uncertainty refers to the lack of knowledge about the outcome associated with a partial observation; that is how much can we say about a prediction. Estimating this, per instance, is much more powerful than giving an overall score for precision and reliability to the whole model (Hüllermeier & Waegeman, 2021). Predictive Uncertainty conventionally comprises two kinds of uncertainties: Aleatoric uncertainty, inherent to the data and generated by the noise or stochastic processes involved in data acquisition; and Epistemic uncertainty, inherent to the model itself, generated by Out-of-distribution or missing data etc.. (Valdenegro-Toro, 2021).

The objective of a model is to learn the probability of some output given the input and the model weights: $P(y|x, w)$. In training, the model tries to learn the best weights given the data $\mathcal{D}$: $P(w, \mathcal{D})$. Standard Neural Networks training employs MLE (Maximum Likelihood Estimation) which ignores uncertainty altogether because it just tries to maximize the probabilities of the labels to be the same as in training. To overcome this problem, point-wise weights are replaced by distributions (often Gaussian) (Abdar et al., 2021; Mobiny et al., 2019).

Bayesian statistics offer a useful framework to model uncertainty in Neural Networks by reformulating the training objectives and optimizations in terms of probability distributions. For this reason, many Uncertainty Quantification methods use this framework and are called Bayesian Neural Networks (BNN).

BNNs re-formulate learning the distribution of the output as computing the predictive posterior distribution

$$P(y|x) = \int_w P(y|w, x)P(w|x)dw \tag{2.1}$$

This equation marginalizes over all possible weights of any model. Obviously this is intractable. Moreover, the posterior distribution of the weights $P(w|x)$ cannot be computed analytically (Mobiny et al., 2019).

Variational Inference is generally used to approximate BNNs. The goal is to find a Variational distribution on the weights $q_\theta(w)$ which minimizes the Kullback-Leibler Divergence $KL(q_\theta(w)||p(w|\mathcal{D}))$. KL Divergence measures the relative distance between two distributions. However, $p(w|\mathcal{D})$ is still unknown such that another approximation is needed. As Mobiny et al. (2019) explain, minimizing KL divergence is equivalent to minimizing the negative evidence lower bound (ELBO).

Among the methods used in this research, Monte-Carlo Dropout and DropConnect use sampling to approximate Equation 2.1. In the Monte-Carlo approximation of the predictive posterior distribution, the number of samples, obviously, directly impacts the quality of the approximation. On the other hand, Deep Ensembles take the Frequentist approach.

In the case of classification tasks, the output of BNNs are categorical probabilities that can be taken as confidence scores. The quality of these confidences can be evaluated using calibration. The essential concept of calibration is that a prediction made with, say, 90% confidence should be correct 90% of the times, and incorrect 10% of the times. In other words: *"A predictor is called well-calibrated if the derived predictive confidence represents a good approximation of the actual probability of correctness"* as Gawlikowski et al. (2022) state.

2.2.1 Uncertainty quantification methods

Monte-Carlo Dropout Monte-Carlo (MC) Dropout is part of the sampling methods for estimating uncertainty. These methods can be computationally very expensive, but generally give good approximates of uncertainty as long as a large number of forward passes is executed. On a practical level, MC-Dropout is implemented by enabling dropout at inference time. Dropout is a

common layer added to networks as a regularization technique in order to avoid over-fitting. It works by multiplying a mask drawn by a Bernoulli distribution with the input activations of the layer, effectively making some of them 0.

Gal & Ghahramani (2016) proved that using dropout at inference time means that a prediction from a different model is taken at each forward pass, which in turn approximates to taking samples of the predictive posterior distribution. This is achieved by using the Monte-Carlo version of the predictive posterior distribution (see Equation 2.2 where M is the number of forward passes).

$$p(y|x) \sim M^{-1} \sum_{i}^{M} p(y|x, \theta_i) \text{ where } \theta_i \sim \Theta \quad (2.2)$$

Monte-Carlo DropConnect DropConnect is a method proposed by Mobiny et al. (2019). It is very similar to MC-Dropout, but the mask is applied to the weights of a layer instead of its input activations.

Being another sampling method, DropConnect presents similar advantages and disadvantages as MC-Dropout.

Deep-Ensemble Lakshminarayanan et al. (2017) proposed to use Deep Ensembles to estimate uncertainty. Practically this is done by training independently multiple instances of the same architecture initialized randomly in a different way. The predictions made by the estimators are then averaged to obtain the model prediction.

According to the reference paper, this method has nothing to envy to BNNs in terms of performance. On the other hand, it brings the advantages of being computationally less expensive and readily parallelizable and scalable.

2.3 Evaluation of Transfer Learning

Putting it all together, when applying Network-based Deep Transfer Learning technique on a standard CNN point features are obtained. When applying this technique on a model that estimates uncertainty, uncertain features, which can be interpreted as probability distributions are obtained.

Since the goal of this research is to compare which of the two types of features would be more informative and lead to better performance, a methodology to assess the quality of the features would be needed. Ho & Kim (2021) gives such methodology that is being followed in this research to evaluate this. All the models were used as feature extractors to produce "off-the-shelf" features from the input images. The features, which are the output of an intermediate layer are then fed as input to a neutral classifier, in this case a Support-Vector-Machine (SVM). The quality of the classifier's predictions can be interpreted as the quality of the features produced.

A Support-Vector-Machine is a supervised machine learning model that tries to find a hyperplane in a N-dimensional space that best separates the data points, where N is the dimensionality of a feature. Such model was chosen for this purpose because, being a non-parametric model, it needs no assumption on the data making it a neutral option, fit to assess the quality of the features.

3 Methods

As mentioned in the last section, the core idea of this research project is to have different models complete a Transfer Learning image classification task. However, it was already pointed out that the focus of the research is not on the model completing the task but on the features it is provided with.

For this reason, different experimental setups were devised to produce these features. This would allow to carry out different analyses and explore different aspects of what uncertain features could achieve.

In the following subsection the general task and the different experimental setups are described.

3.1 Procedure

The task consists in classifying samples from the fashion MNIST dataset using a SVM. However, the focus is on what is fed to the SVM, rather than the classifier itself.

Much of what is done in this research happens in the pre-training phase. Here, all the models are created by combining the base architectures and UQ methods that will be described in section 3.2. All these models are trained on MNIST, in order

to obtain the weights that will output meaningful features when given the input.

The trained models are then truncated to obtain the feature extractors. The feature extractors are the trunk models whose output layer was discarded. By running these truncated networks in inference mode, the features are obtained. In this research, the second-to-last layer was taken to output the intermediate features. In the case of models with uncertainty, we obtain the means and variances that describe each of the weights in the layer.

Input images for both training and testing are ran through the feature extractors and become features that are the input to the SVM.

Before feeding the features to the SVM, these are passed through Principal Component Analysis (PCA). This is a technique which applies a linear transformation to the features in order to obtain lower-dimensional features that can still explain part of the variance of the original data. In this research, 90% of the variance must be explained when applying PCA. This means that the dimensionality of the features varied at each run. PCA was used because, although it causes some information to be lost, it eases much of the computational complexity of the procedure.

In Appendix A the reader can find visual representations of the features extracted by all the models. These are the data-points the SVM needs to separate. Figure 3.1 shows a diagram that summarizes the procedure.

3.1.1 Experimental setups

The task is carried out similarly for all the experiments, with only slight changes. These can be noted also in figure 3.1 and will be explained in detail in the following paragraphs.

UQ comparison The first experiment explores the performance of the different methods applied to the different base architectures. For this, only the mean of the uncertain features is used. That means that the data points that the SVM will classify are the mean values of the features. In this experiment, 1000 samples from fashion MNIST are classified.

Partial UQ application In the second experiment, only those methods that can be applied partially are taken into exam. Applying a UQ method partially means that only a number of layers implement custom functions that estimate uncertainty. This condition aims at showing the gradual impact of Uncertainty Quantification on the quality of the features.

Similarly to the previous condition, a 1000 samples from fashion MNIST are classified.

Sampling new features The last experiment is somewhat different from the previous two. The aim of this condition is to realize the potential of having uncertain features over point features. That is, using both the mean and the variance of the uncertain features.

Using the variance directly as data points to feed into the SVM (as is done with the mean) does not lead to any results. In this case, the classifier would not perform better than chance. The reason why this happens is that the value of variance does not have a proper sense when not in the context of a mean value it refers to.

Uncertain features, yielding mean and variance can be used to sample new data from the assumed distribution, a Gaussian. By doing so, we obtain N new data points from the distribution which supposedly approximate the true distribution of the features.

A classifier such as an SVM generally works better if it has more data to fit during training. In these terms, sampling uncertain features can be seen as a Data Augmentation technique (Perez & Wang, 2017). Therefore, it can be expected that the SVM will perform better when fitted on more data as long as the new data is informative. This can be traced back to the quality of the features.

Since in this experiment, the goal is to augment the data, a much smaller subset of the fashion MNIST dataset is going to be used. Namely, only 90 samples will be taken for the SVM to fit. However, the sampling process will be repeated 10 times, for each sample, yielding a final batch that will be used for training which will be 10 times bigger than the initial subset.

3.1.2 Evaluation protocol

As described in section 2.3, the quality of the features is assessed by measuring the performance of the SVM to which they are fed. The SVM predictions are compared on the grounds of two different

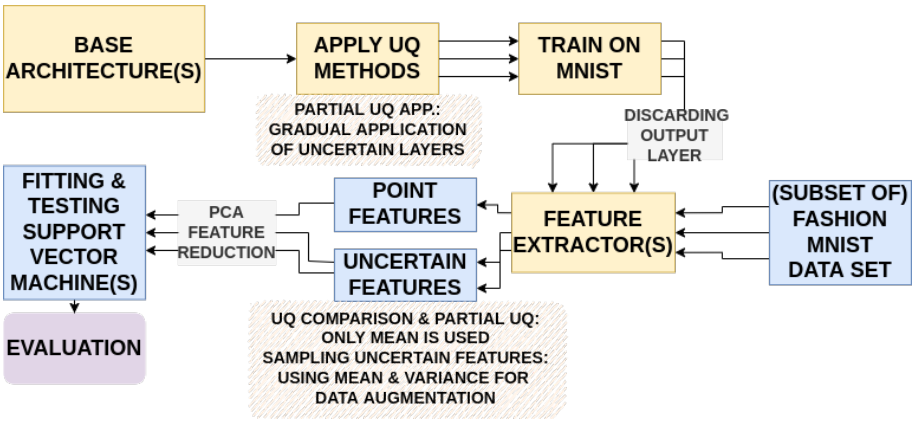

Figure 3.1: Diagram of the experiments procedure

measures that give insight on the predictive quality and reliability of the predictions.

Accuracy The first metric is the simple accuracy score. It is the ratio of correct predictions over the total. While it is not a proper scoring rule and is seldom used alone, it can give a general idea of whether the model is working correctly, especially when working with small datasets that are known to be balanced and well-annotated.

Expected Calibration Error This metric was briefly presented in section 2.2 and is discussed in detail in Gawlikowski et al. (2022). Calibration is computed by dividing the confidence score outcomes in bins and computing the accuracy of each bin. The error is obtained by summing all the differences between the accuracy of the bin and the confidence score represented by the bin.

This metric is based on the observation that the confidence scores should be a good approximation of the accuracy of the predictions. Thus, by computing the error we get an idea of how reliable the predictions are.

The expected calibration error (ECE) differs from the calibration error in that the error of each bin is weighted by the number of predictions in the bin, so to obtain a normalized score.

3.2 Implementation details

The experiments were implemented using the Keras framework (Chollet et al., 2015) for the datasets and the baseline architectures. In order to implement the custom layers and models needed for uncertainty quantification methods, the library keras_uncertainty (Valdenegro-Toro, 2018) was used. Both these libraries are open-source and publicly available.

3.2.1 Datasets and pre-processing

For all the experiments the same two datasets were used: MNIST and fashion-MNIST. The former was used as the informative dataset on which all the models were pre-trained while the latter was taken as the target dataset on which the validation performance was measured.

These two datasets were preferred for their simplicity and availability. Since this research aims at being an exploration of the subject, there is no interest in correlating the results to a specific dataset or field.

MNIST and fashion-MNIST come conveniently packaged as objects in Keras modules. The preprocessing was therefore simple. All that needed to be done was scaling and fixing the shape as shown in the examples on the Keras website.

3.2.2 Baseline architectures

The two baseline architectures used for comparison for this experiment are ConvNet, inspired by the example present on the Keras website, and a simple implementation of a Multi-layer Perceptron (MLP). The MNIST data-set, in fact, does not pose

much of a challenge and thus, such simple architectures are an easy and a sufficient baseline for this task. A random search was performed on the Hyper-parameters using keras_tuner, in order to find the best parameters for the models

The exact configurations used are the following:

- **ConvNet** Conv2D(64, 3×3, ReLU) - MaxPool(2×2) - Conv2D(96, 3×3, ReLU) - MaxPool(2×2) - Flatten() - Dropout($p = 0.3$) - Dense(10, softmax).
- **MLP** Dense(8, ReLU) - Dropout($p = 0.3$) - Dense(8, ReLU) - Dropout($p = 0.3$) - Dense(8, ReLU) - Dropout($p = 0.3$) - Flatten() - Dense(10, softmax).
- **pre-training** Number of epochs: 60; batch-size: 32; optimizer: Adam(lr = 0.001); callbacks: early stopping.

3.2.3 Uncertainty quantification methods

The ConvNet and MLP architectures were not only used as baseline for comparison but also served as the base on which models with uncertainty are built. That means that models with uncertainty follow the same architecture as the baselines but apply slight changes in the structure. The methods used are MC-Dropout, MC-DropConnect and Deep-Ensembles as described in section 2.2.1.

For sampling methods, custom layers from keras_uncertainty replaced the standard implementations of Dense, Dropout and Convolutional layers. For ensembles, on the other hand, keras_uncertainty offers ad-hoc models to implement them starting from an existing architecture.

3.2.4 Parameters

The parameters of the experiments can be divided into two groups: those inherent to the models and those inherent to the procedure.

Some parameters inherent to the models were already described in section 3.2.2. For the uncertainty quantification methods other important parameters are the dropout rate and number of forward passes for DropConnect and MC-Dropout. The dropout rate was set to 0.11. This values was found by exploring different values and selecting the one that yield best ECE and accuracy. The number of forward passes, as already pointed out, is a parameter that should only be maximized to obtain better performance. However, to suit the hardware capabilities, this was set to 100 for this research.

The parameters inherent to the procedure comprised the number of samples to fit the SVM with and the amount of variance to be explained when applying PCA. The former was mentioned in section 3.1.1, the latter in section 3.1.

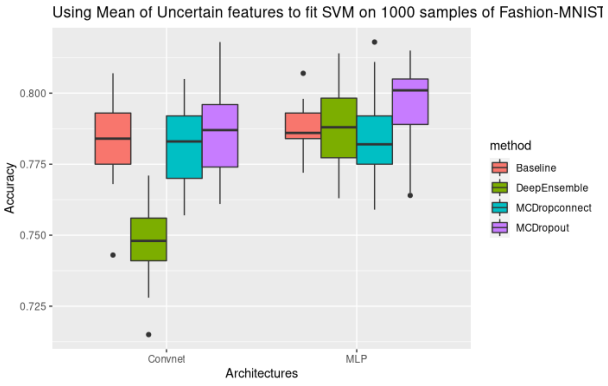

Figure 4.1: Comparison of the accuracy of SVMs fitted on features extracted by models with different architectures and different UQ methods applied to them. (UQ comparison experiment condition)

4 Results

This section will walk the reader through the main results obtained from the experiments described in the previous section and the analyses carried out.

4.1 UQ methods comparison

The first experimental setup aimed at comparing the accuracy and expected calibration error obtained by the SVMs trained on the features obtained by different methods applied to different architectures as described in Section 3.1.1.

Accuracy The main findings on accuracy are summarized by Figure 4.1.

Across architectures, the mean accuracy for each method is shown in Table 4.1.

Table 4.1: Mean Accuracy of each method in comparison experiment condition.

Method	Accuracy (%)
Baseline	78.57%
Deep-Ensemble	76.69%
MC-DropConnect	78.37%
MC-Dropout	79.13%

Table 4.2: Mean Expected Calibration Error per Method in comparison experiment condition.

Method	ECE
Baseline	0.0303
DeepEnsemble	0.0423
MCDropconnect	0.0283
MCDropout	0.0315

ANOVA test was ran to confirm any statistical significance among the differences that could be observed in Figure 4.1 and Table 4.1. Since the p-value is very small (7.98×10^{-7}) , it follows that the method used can be a predictor of the resulting accuracy.

In order to find out what groups mostly led to the previous finding, a Tukey test was ran as well. The Tukey test, in essence, runs a multiple pairwise means comparison among the groups.

For this condition, the Tukey test only highlighted one comparison with a small enough p-value to be significant (1.65×10^{-4}). This is the difference between Deep-Ensemble and Baseline. The difference is negative (-0.02) since applying this technique actually hinders performance.

Expected Calibration Error The same analyses were carried out for the expected calibration error. The box-plot which gives a visual insight into the difference in ECE is shown in Figure 4.2. The values of the mean ECE per method are reported in Table 4.2.

ANOVA, again, shows a significant difference (p-value is 4.16×10^{-8}). Upon further investigation using the Tukey test, it was ascertained that there is a significant difference (p-value $= 6 \times 10^{-6}$) between the Deep-Ensemble models and the baseline of 0.01. It follows from the definition of ECE that this is a negative result.

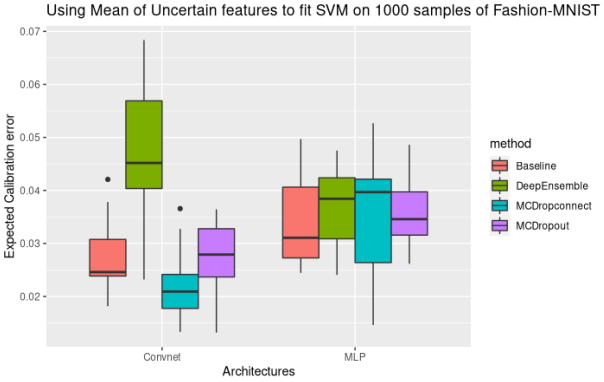

Figure 4.2: Comparison of the ECE of SVMs fitted on features extracted by models with different architectures and different UQ methods applied to them. (UQ comparison experiment condition)

Table 4.3: Mean Accuracy per number of layers in partial UQ application experiment condition.

N# of layers	Accuracy (%)
0	79.99%
1	79.67%
2	78.98%
3	79.38%

4.2 Partial UQ application

The second experimental setup aimed at showing the gradual impact of (partially) applying more layers that estimate uncertainty as described in Section 3.1.1.

Accuracy The results on accuracy are summarized by Figure 4.3. The mean value of accuracy across methods per number of uncertain layers is reported in Table 4.3.

Running ANOVA highlighted a significant difference (p-value $= 1.2 \times 10^{-3}$).

Further analysis running the Tukey test, showed a mean difference of -0.01 (p-value $= 7.6 \times 10^{-4}$) between models with 2 layers and models with 0 layers (baseline). This and the other pairwise mean comparisons are plotted in Figure 4.4.

Expected Calibration Error Figure 4.5 shows the results of this experiment in terms of expected

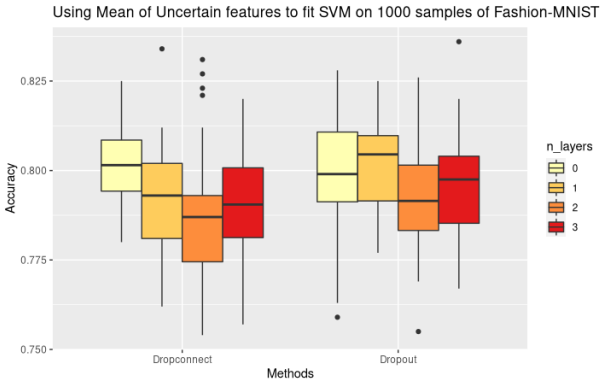

Figure 4.3: Comparison of the accuracy of SVMs fitted on features extracted by models with a different number of MC-DropConnect or MC-Dropout layers. (partial UQ application experiment condition)

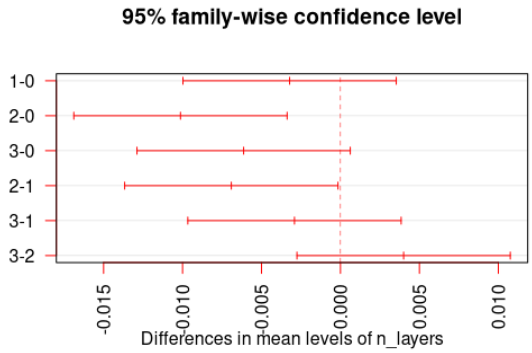

Figure 4.4: Pairwise mean comparison (performing Tukey test) in accuracy among different numbers of uncertain layers in the feature extractor of the partial UQ application experiment setup.

Table 4.4: Mean ECE per number of layers in Partial UQ application experiment condition.

N# of layers	ECE
0	0.0352
1	0.0343
2	0.0355
3	0.0351

Table 4.5: Mean Accuracy per method in sampling experiment condition.

Method	Accuracy (%)
Baseline	58.95%
Deep-Ensemble	58.64%
MC-DropConnect	63.17%
MC-Dropout	61.48%

calibration error. The mean values of ECE per number of uncertain layers is reported in Table 4.4.

Running ANOVA did not show any statistically significant difference (p-value of n_layers as a predictor is 0.89) which rendered unnecessary to carry out further analysis with the Tukey test.

4.3 Sampling new features

In the third experiment, the distributions of uncertain features were sampled to obtain new datapoints, as described in Section 3.1.1.

Accuracy The results on accuracy are summarized by Figure 4.6. The mean value of accuracy across architectures per uncertainty quantification method is reported in Table 4.3.

Running ANOVA highlighted a significant difference (p-value $= 9.97 \times 10^{-5}$).

The Tukey test confirmed that MC-DropConnect and MC-Dropout perform better than the baseline. The mean differences with the baseline are respectively 0.04 (p-value $= 3.54 \times 10^{-3}$) for the former, and 0.04 (p-value 1.16×10^{-2}) for the latter.

Expected Calibration Error The results of this experiment in terms of expected calibration error are shown in Figure 4.7. The mean values of ECE per method are reported in Table 4.6.

The very small p-value in the ANOVA results (3.26×10^{-12}) indicates that there is a statistical significance in the differences among the groups. These differences are found by running the Tukey test. A summary of the pairwise mean differences are shown in Table 4.7

5 Conclusions

In general, the experiments seem to answer the research question negatively: uncertain features do not seem to lead to better results. Nonetheless, the

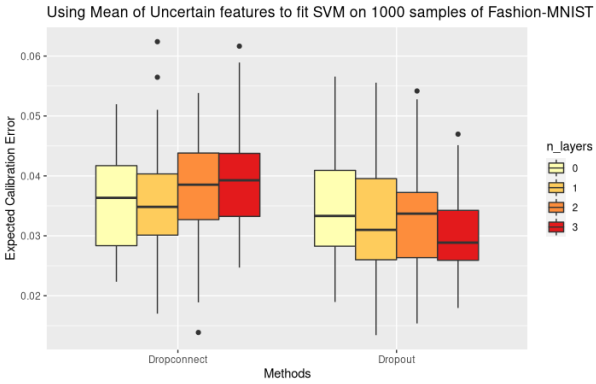

Figure 4.5: Comparison of the ECE of SVMs fitted on features extracted by models with a different number of MC-DropConnect or MC-Dropout layers. (partial UQ application experiment condition)

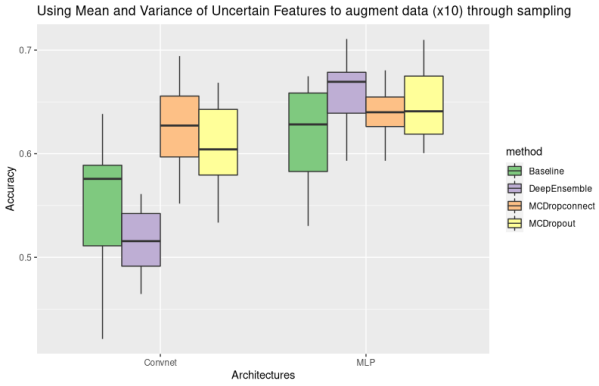

Figure 4.6: Accuracy of SVMs trained on baseline data vs SVMs trained on data augmented by sampling features obtained from different models. (sampling new features experiment condition)

Table 4.6: Mean ECE per method in sampling experiment condition.

Method	ECE
Baseline	0.1196
DeepEnsemble	0.2059
MCDropconnect	0.1604
MCDropout	0.1616

Table 4.7: Pairwise Mean comparisons (performing Tukey test) of the expected calibration error between each of the methods and the baseline (sampling new features experiment condition). Diff is the mean difference among the groups

	diff	p-value
DeepEnsemble-Baseline	0.09	0.00
MCDropConnect-Baseline	0.04	0.01
MCDropout-Baseline	0.04	0.01

results in Section 4 shed some light on this topic and allowed for some interesting observations to be made.

5.1 Mean of uncertain features

The first two experimental setups used only the mean of uncertain features. This way what was tested was the quality of the approximation of the distribution of the features. This tends to vary with the number of forward passes executed (Gawlikowski et al., 2022). With the parameters used in this research, the mean of the uncertain features did not lead to better results in transfer learning. In fact, no significant results were found. On the contrary, in some cases, uncertain features seemed to be performing worse.

As reported in Section 4.1, the Bayesian Neural Networks performed as good as the baseline or slightly better, although not significantly. Since there was no background literature on the extraction of uncertain features, finding no improvement at all was a plausible hypothesis. What came as a bit of a surprise were the results of uncertain features extracted from Deep Ensembles. These features performed significantly worse in both accuracy and ECE, despite Lakshminarayanan et al. (2017) proving that probabilistic Ensemble techniques are comparable to Bayesian Neural Networks.

Appendix A shows the data-points in a 2-dimensional space. Although the SVM works in a higher dimensional space, these plots give a visual insight on the task the classifier has to perform. The plots look fairly similar for equal architectures, across different methods. It is clear from the figures that the data-points mostly overlap. This would explain why the performance of the SVMs did not reach optimal results, as the points are very difficult

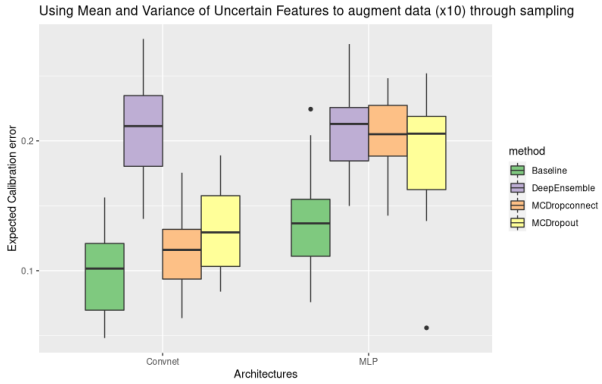

Figure 4.7: ECE of SVMs trained on baseline data vs SVMs trained on data augmented by sampling features obtained from different models. (sampling new features experiment condition)

to distinctly separate. This is even more obvious when observing the plots of the features obtained from the Deep-Ensembles (Figures A.7; A.8), which explains why these models in particular performed so poorly.

Applying MC-Dropout and MC-DropConnect layers partially (see Section 4.2) led to some more uninspiring results. It seems to be the case that the more uncertain layers a model implements, the worse it performs in terms of accuracy. On the other hand, this is not the case for ECE, such that modeling uncertainty does not improve nor hinder reliability. This can also come as a bit of a surprise since this research started from the assumption that modeling uncertainty is needed for improving reliability (Gawlikowski et al., 2022; Leibig et al., 2017; Abdar et al., 2021).

Given the lack of similar studies, the best explanation that can be produced is that these results stem from the fact that ML and DL involve a lot of craft. Transfer Learning techniques in particular need a lot of trial and error and do not generalize to every architecture, technique and/or dataset (Tan et al., 2018). This renders studying such methods sensitive to design choices and to the range of architectures, methods and datasets explored.

Evidence for these conclusions can be also observed in the data collected. Although no significant results could be drawn, it can be observed how accuracy is sensitive to design choices. For instance, performance plummets when using Deep-Ensembles but only when applying this technique to Convnet, whereas performance is similar to baseline for MLP Deep-Ensembles. A similar phenomenon happens in the partial UQ application condition, where the differences can be observed mainly when using MC-DropConnect rather than MC-Dropout.

5.2 Sampling features as a data augmentation technique

Using only the mean of uncertain features did not turn out to be fruitful. However, as mentioned in the introduction, the main advantage of having uncertain rather than point features is that uncertain features can be interpreted as probability distributions, since the weights of uncertain layers can be seen as probability distributions (Abdar et al., 2021; Mobiny et al., 2019). When sampling from uncertain features we obtain new data from the distribution of the features which, presumably, approximates on some level the distribution of the features of real-world data.

Data augmentation is another strategy used to counter scarcity of data instead or along Transfer Learning (Perez & Wang, 2017). It is general knowledge that more data should lead to better performance of Deep Networks. However, data should still be in-distribution to avoid the model underfitting. Other well-known Data Augmentation techniques such as manipulation or generative adversarial networks (GANs) have this precise goal (Perez & Wang, 2017). These techniques, in fact aim at creating new data points by transforming the available data.

Good Data Augmentation techniques are therefore expected to boost a model's accuracy. As reported in Section 4.3, this seems to apply to sampling new data from uncertain features produced with MC-DropConnect and MC-Dropout extractors. Hence, this procedure could be considered as a Data Augmentation technique for future experiments.

On the other hand, the findings were not as positive as far as reliability is concerned. Section 4.3 shows that all UQ methods scored a worse ECE. It could have been expected to have an opposite result as UQ methods are applied to improve reliability of Deep Learning models (Leibig et al., 2017).

However, the negative results can be explained by the way SVMs work. As stated by Pedregosa et al. (2011), the probabilities outputted by SVMs do not accurately represent confidence when fitting very small batches of data, as it happens to be the case for the sampling experiment (see Section 3.1.1).

5.3 Limitations

In order to have a complete picture of this study it is very important to discuss its limitations. These mainly involve computational and time constraints which limited the design and execution of the experiments.

Deep Learning models can be computationally very expensive to train. Applying Uncertainty Quantification methods makes these models many times more expensive to train and run. This heavily factored into experiment design and the choice of parameters.

Given the exploratory nature of the research, it would have been ideal to test most if not all popular benchmark Networks tested in other, similar studies. Some candidates would have been Deep Networks such as VGG-19, ResNet etc. However, these models contain a number of weights in the order of millions, rendering it unrealistic to train Uncertainty Quantification models based on them with the available hardware and time frames. Using Convnet and MLP as base architectures was the viable option but, possibly, not the best one in terms of completeness.

Many compromises had to be made on parameter choices as well, for the same reasons. A major example is the number of forward passes for Bayesian NNs or the number of instances in the ensembles. As mentioned before, this has a direct impact on the quality of the approximation of the distribution of the weights, but it had to be kept rather low to keep the procedure feasible.

Using SVMs as target models also posed some limitations on the experiments. These ML models, in fact, struggle with larger datasets and too highly-dimensional data. For this reason the full fashion-MNIST dataset could not be used and PCA was needed to be applied on the features, which costed some loss of information.

5.4 Future research

Although this research did not produce particularly exciting results and needed to be rather limited, it could still be considered as a suggestion to investigate this topic more. Resourceful research could immediately be conducted by overcoming the limitations just discussed. Given the important implications of using Uncertainty Quantification in high-stakes fields where Transfer Learning is employed, further exploring the methods discussed can potentially lead to game-changing findings.

If uncertain features really do have the potential to improve Transfer Learning, all it takes to find out is a more exhaustive research conducted with powerful hardware.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... Nahavandi, S. (2021, December). A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges. *Information Fusion*, *76*, 243–297. Retrieved 2022-07-29, from `http://arxiv.org/abs/2011.06225` (arXiv:2011.06225 [cs]) doi: 10.1016/j.inffus.2021.05.008

Chollet, F., et al. (2015). *Keras.* GitHub. Retrieved from `https://github.com/fchollet/keras`

Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*, 100379. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1574013721000198` doi: https://doi.org/10.1016/j.cosrev.2021.100379

Gal, Y., & Ghahramani, Z. (2016, October). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* arXiv. Retrieved 2022-07-10, from `http://arxiv.org/abs/1506.02142` (Number: arXiv:1506.02142 arXiv:1506.02142 [cs, stat])

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., ... Zhu, X. X. (2022, January). A survey of uncertainty in deep neural networks. *arXiv:2107.03342 [cs, stat]*. Retrieved

2022-03-28, from http://arxiv.org/abs/2107.03342 (arXiv: 2107.03342)

Ho, N., & Kim, Y.-C. (2021, December). Evaluation of transfer learning in deep convolutional neural network models for cardiac short axis slice classification. *Scientific Reports*, *11*(1), 1839. Retrieved 2022-02-18, from http://www.nature.com/articles/s41598-021-81525-9 doi: 10.1038/s41598-021-81525-9

Hüllermeier, E., & Waegeman, W. (2021, March). Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning*, *110*(3), 457–506. Retrieved 2022-05-20, from http://arxiv.org/abs/1910.09457 (arXiv:1910.09457 [cs, stat]) doi: 10.1007/s10994-021-05946-3

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017, November). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles.* arXiv. Retrieved 2022-07-10, from http://arxiv.org/abs/1612.01474 (Number: arXiv:1612.01474 arXiv:1612.01474 [cs, stat])

Leibig, C., Allken, V., Ayhan, M. S., Berens, P., & Wahl, S. (2017, December). Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, *7*(1), 17816. Retrieved 2022-03-28, from http://www.nature.com/articles/s41598-017-17876-z doi: 10.1038/s41598-017-17876-z

Mobiny, A., Nguyen, H. V., Moulik, S., Garg, N., & Wu, C. C. (2019, June). *DropConnect Is Effective in Modeling Uncertainty of Bayesian Deep Networks.* arXiv. Retrieved 2022-05-29, from http://arxiv.org/abs/1906.04569 (Number: arXiv:1906.04569 arXiv:1906.04569 [cs, stat])

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

Perez, L., & Wang, J. (2017, December). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning.* arXiv. Retrieved 2022-07-23, from http://arxiv.org/abs/1712.04621 (Number: arXiv:1712.04621 arXiv:1712.04621 [cs])

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, August). *A Survey on Deep Transfer Learning.* arXiv. Retrieved 2022-05-13, from http://arxiv.org/abs/1808.01974 (Number: arXiv:1808.01974 arXiv:1808.01974 [cs, stat])

Valdenegro-Toro, M. (2018). *keras-uncertainty.* GitHub. Retrieved from https://github.com/mvaldenegro/keras-uncertainty

Valdenegro-Toro, M. (2021, April). *I Find Your Lack of Uncertainty in Computer Vision Disturbing.* arXiv. Retrieved 2022-05-20, from http://arxiv.org/abs/2104.08188 (Number: arXiv:2104.08188 arXiv:2104.08188 [cs])

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... He, Q. (2020, June). *A Comprehensive Survey on Transfer Learning.* arXiv. Retrieved 2022-05-19, from http://arxiv.org/abs/1911.02685 (Number: arXiv:1911.02685 arXiv:1911.02685 [cs, stat])

A Data Visualization

In this appendix are included the plots showing the (high-dimensional) features obtained from the feature extractors mapped to a 2-dimensional space using Principal Component Analysis (PCA). The data is obtained from one run of the UQ comparison experiment setup.

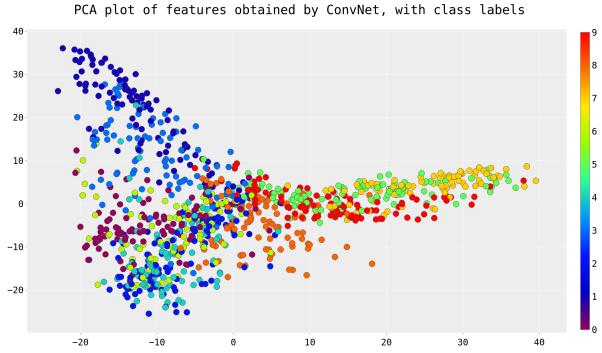

Figure A.1: Features extracted from the baseline model ConvNet mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

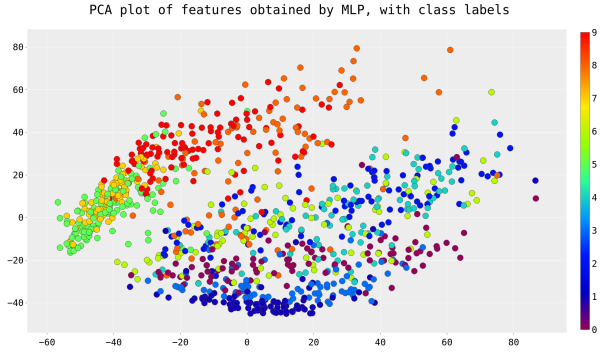

Figure A.2: Features extracted from the baseline model MLP mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

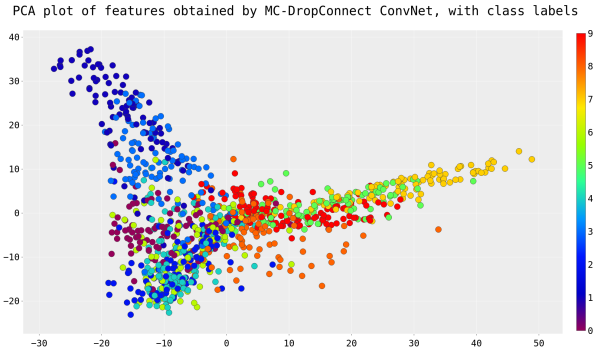

Figure A.3: Features extracted from the DropConnect model ConvNet mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

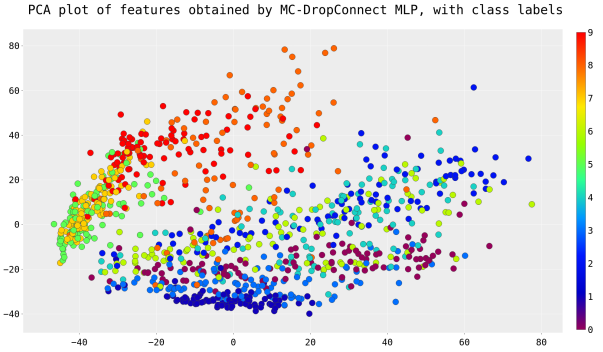

Figure A.4: Features extracted from the DropConnect model MLP mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

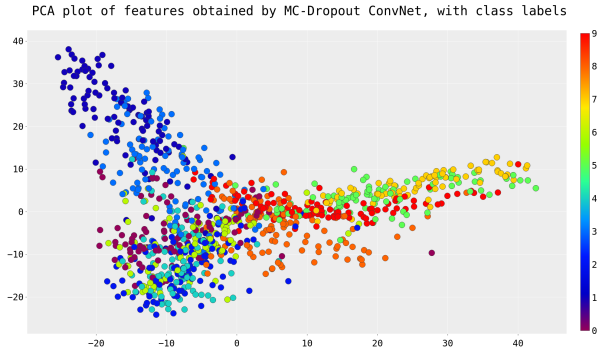

Figure A.5: Features extracted from the Dropout model ConvNet mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

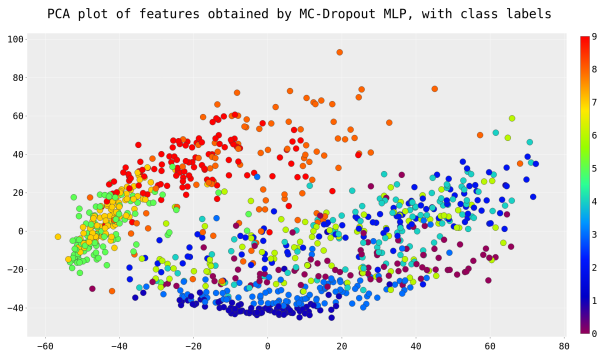

Figure A.6: Features extracted from the Dropout model MLP mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

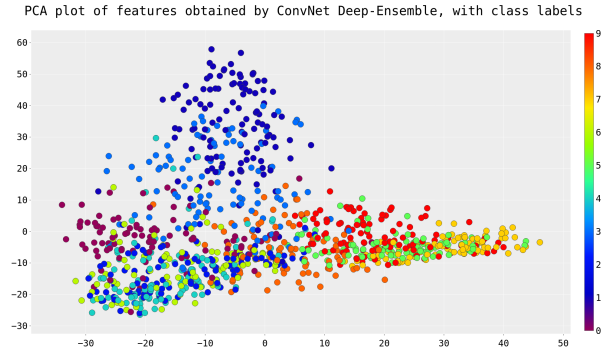

Figure A.7: Features extracted from the Deep-Ensemble of ConvNet models mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.

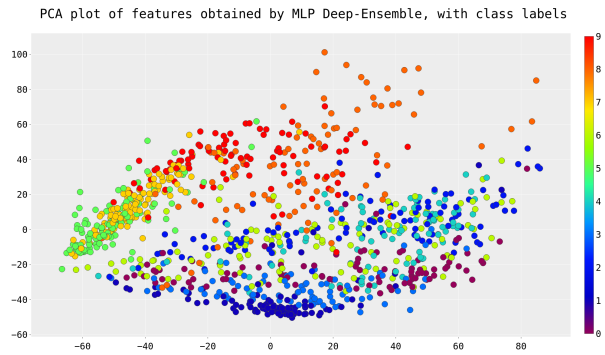

Figure A.8: Features extracted from the Deep-Ensemble of MLP models mapped to a 2-dimensional space using PCA. The color-scale indicates the class to which the data-point belongs.