DEEP SUB-ENSEMBLES FOR DIABETIC RETINOPATHY DIAGNOSIS

Stephen Parsons, Kristina Gessel, Sairam Sri Vatsavai
University of Kentucky
{srpa226, kmge224, ssr226}@uky.edu

ABSTRACT

Understanding the uncertainty of deep learning models is critical in a number of applications, but methods so far have been developed outside of tight time constraints. Uncertainty prediction often involves expensive calculations such as the use of Deep Ensembles. A recent work, Deep Sub-Ensembles for Fast Uncertainty Estimation in Image Classification, aims to develop an uncertainty prediction method that nearly matches the performance of slower methods while improving the execution time. This work aims to reproduce the findings of that paper, and to apply them to a new dataset: an image classification dataset of diabetic retinopathy diagnosis.

Index Terms— Bayesian Machine Learning, Uncertainty, Ensembles, Computer Vision, Entropy

1. INTRODUCTION

Deep Neural Networks (DNNs) are proven to be very effective in solving machine learning tasks [5] efficiently. They are used in a broad spectrum of tasks like computer vision [7], speech recognition[8] and natural language processing [6]. These tasks may include high risk fields like autonomous vehicle control, robotics, financial, medical and legal fields where cost of an error is high. Thus, DNNs not only require high accuracy but also reliability and robustness. In order to achieve that DNNs uncertainty needs to be estimated during model training and inference to avoid any mishaps. Conventionally, Bayesian approaches which model uncertainty by modeling parameters of DNN with prior probability distribution and the posterior distribution over the parameters to estimate the uncertainty. Often adding these Bayesian approaches leads to an increase in implementation complexity, cost of training and decline in accuracy compared to standard DNNs. In addition, these require changes in the traditional training pipeline. Recently, few Non Bayesian approaches like Monte Carlo dropout [9] and Deep Ensembles [2] were proposed that perform on par uncertainty estimation for the DNNs at the same time requiring little modifications. However, along with quality of estimation, rate of estimation plays a vital role in applications especially like robotics. Even though Deep Ensembles quantify uncertainty well they are also computationally expensive and can be slow at inference. To speed up inference time Deep Sub-Ensembles [4] an enhancement to [2] was proposed to achieve fast uncertainty estimation. This motivated us to analyze existing uncertainty estimators and apply Deep Sub- Ensembles to a new dataset to verify the performance gain .

1.1. Project Objectives

The main goals of our work are to develop a better understanding of uncertainty in DNNs which is needed for establishing safer Machine Learning systems. We analyse various uncertainty estimators by reproducing benchmark results on diabetic retinopathy to improve our understanding. Then, for the first time we successfully evaluated the performance of Deep Sub Ensembles and Deep Ensembles on diabetic retinopathy diagnosis dataset. We compare their performance metrics like accuracy, AUC and inference time/speed up against well known state of the art uncertainty estimators [10-11].

1.2. Division of Work

Kristina implemented the training and inference steps of the deep sub-ensembles algorithm and also trained some of the first models that were used to obtain the final results, in addition to other miscellaneous coding work.
Stephen performed the initial dataset preparation and sharing, and implemented the benchmark methods to evaluate against.
Sairam adapted the reusable plotting features from the benchmark repository for the project, trained multiple

models used for algorithm implementation and created figures.

2. BACKGROUND

The need for deep networks to make decisions under tight time constraints is well documented. One example of this would be object detection in video frames. For a real world agent, it is important to process the incoming data quickly enough to make relevant decisions, as the environment is constantly changing. The tradeoff between accuracy and execution time is captured by recently developed benchmarks such as Streaming Perception [3].

It is also important to understand the uncertainty of deep models during their execution, to improve our understanding of the data they are processing and the decisions being made. This is the subject of a field of research to itself, discussed further in Section 3.

Thus far, uncertainty prediction methods have to our knowledge been considered independently of the aforementioned tradeoff between accuracy and execution time. However, an agent operating in real time may very well wish to make use of uncertainty information. In this case, it is critical that the uncertainty predictions be performed fast enough to be of use to the agent before the environment has moved on completely. The application to fields such as robotics and autonomous vehicles is immediately apparent, but there are others as well. Cheap uncertainty prediction could be useful in game playing agents, for image classification on mobile devices with limited battery, and in many other applications.

3. RELATED WORK

3.1. Deep Ensembles

Deep ensembles use the aggregated predictions of multiple trained models to make predictions. The high-level architecture of a deep ensemble is shown in Fig. 1.

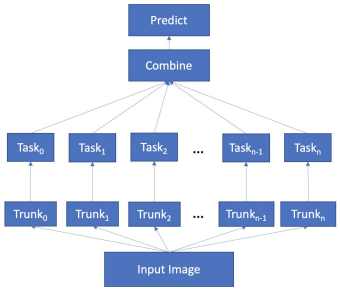

Fig. 1. Deep ensemble architecture, as described in [2].

Each model that makes up an ensemble must be trained on the training set. To perform inference, each model is given the same input and their predictions are combined in some way to produce a final prediction that is essentially the most likely prediction given the outputs of all the different models. The efficiency of this approach degrades as more models are added to the ensemble. Deep ensembles are cost-prohibitive in some domains such as robotic. This has motivated researchers to find ways to make ensembling more tractable.

3.2. Deep Sub-Ensembles

Deep sub-ensembles were proposed as an efficiency improvement on deep ensembles [4]. Deep sub-ensembles reduce the time taken to perform inference by maintaining a single 'trunk' network and training multiple small 'task' networks to obtain the ensemble effect. The image is passed into the trunk network, then the trunk network's output is passed into each of the task networks. The task networks' outputs are combined to produce a final prediction. The high-level architecture is shown in Fig. 2.

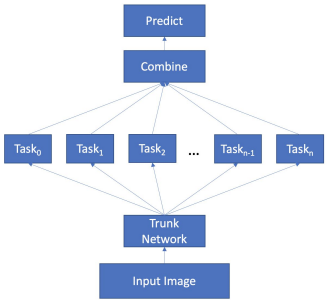

Fig. 2. Deep sub-ensemble architecture, described in [4].

To train a deep sub-ensemble, first an entire network must be trained with the dataset of interest. This full network is then divided into a trunk and a task network. Typically, the trunk network contains the majority of the layers. The task network layers of this initial network are saved as the first task network.

Further task networks can be trained by creating a new full network, copying all of the trunk network's weights into the new network and leaving the task layers randomly initialized. The trunk layers must be frozen, so they do not change during training. The new network is trained on the training set to change the task network's weights. Once training is complete, only the task network's weights are saved. This process is repeated as many times as necessary to get the desired number of task networks.

To perform inference, first the trunk model is evaluated on the input image. Then each task network is evaluated based on the trunk model's output. The task network outputs are combined to produce a final prediction.

Deep sub-ensembles are more efficient than their deep ensemble counterparts because the large trunk network is only evaluated once and the result is passed into each of the smaller task networks to obtain a final result.

4. METHODS

All code for this experiment was written in Python using the TensorFlow and Keras libraries. All code was consolidated into a Colab notebook to make access and editing easy for all team members.

4.1. Data Preparation

The diabetic retinopathy dataset was downloaded from Kaggle and prepared on a local machine similar to the approach in [1]. The dataset was uploaded to Google Drive to make sharing the dataset among all team members straightforward. The "250K" dataset--containing images with roughly 250,000 pixels each--was used for all experiments.

Due to resource constraints, the images were augmented minimally. While transformations such as slight translation, rotation, or zooming would help make the trained network more robust to variations, each of these transformations caused the training time to increase dramatically. To avoid Colab usage limits, the images were only resized. The images originally were labeled from zero to four, indicating the severity of diabetic retinopathy. These labels were converted to binary labels. Labels zero and one were mapped to the 'not diseased' label, while labels two through four were mapped to the 'diseased' label.

4.2. Training

A neural network similar to the one used in the existing benchmark repository was used [1]. This network is a VGG-like deep convolutional neural network, shown in Fig. 3. This network was initialized with a dropout rate of 0.1, a base filter size of 32, a learning rate of 0.0004 and l2-regularization for the convolutional kernels set at 0.0005. Every convolutional layer uses a 3x3 kernel with a stride of one pixel and same padding. ReLU activation functions were used between convolutional layers. A dropout layer was included after the ReLU activation functions between every convolutional layer. The final output runs through global pooling, then a dense layer and finally a sigmoid activation function to generate a prediction.

The network was trained for 70 epochs on the training set of images. Weights were randomly initialized each time. A total of ten models were trained. One additional model was trained for only 50 epochs on the same training set.

4.3. Implementing Deep Sub-Ensembles

One of the ten trained models was selected to be used as the trunk network in the deep sub-ensemble method. The trunk layers were copied from the trained network into a newly-initialized network, and all trunk layers were frozen. Only the layers that made up the task network remained newly initialized. The results shown below test a three convolutional layer task network and a one convolutional layer task network. The remaining layers were part of the trunk network. The division of the trunk network and the (one convolutional layer) task network is illustrated in Fig. 3. The task network was trained for 5 epochs in the case of the one layer task network, and 10 epochs in the case of the three layer task network. Training converged quickly. Dropout layers were also present in the trunk network to improve robustness of the model. All hyperparameters were set to the full trained network's

hyperparameters, as described in Section 4.2, for consistency.

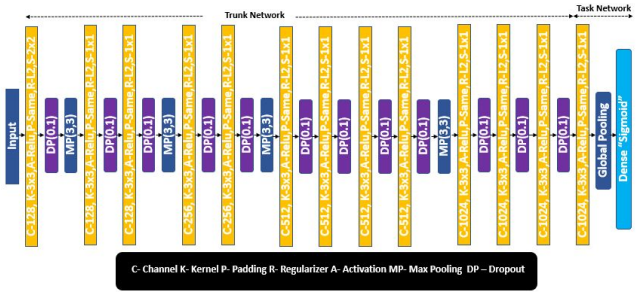

Fig. 3. Neural network architecture with labeled (one convolutional layer) task and trunk networks

4.4. Evaluation Strategies

Four existing diabetic retinopathy benchmarks were implemented to directly compare the performance of deep sub-ensembles to methods with known performance [1]. The benchmark algorithms used in this study were deep ensembles, MC dropout, a deterministic network, and ensemble MC dropout.

All ten of the trained models described in Section 4.2 were used in a deep ensemble to compare the performance of the deep ensemble to the deep sub-ensemble approach. These same models were used for ensemble MC dropout as well. A single model was randomly chosen from the ten trained models for MC dropout and the deterministic approach.

5. RESULTS

5.1. Predictive Ability

In our experiments, Deep Sub-Ensembles do allow for a tradeoff between predictive power and execution time. This confirms the original claims of the Deep Sum-Ensembles paper. In the original work, the method was tried on the MNIST and CIFAR datasets. The lower error rates on those datasets make it not worthwhile to directly compare those error rates with our results. Instead, we look only at the relative performance of Deep Sub-Ensembles against other methods on the diabetic retinopathy diagnosis dataset.

Figure 4 shows the results of the various uncertainty prediction methods on 100 batches (of size 128) of the test dataset. For this experiment, the trunk network was trained for 50 epochs and 5 sub-ensembles were trained for 10 epochs each. The sub-ensembles had 3 convolutional layers, plus the final fully connected layer as their task networks. The rest of the VGG network was the trunk network.

It can be seen that the sub-ensembles do indeed offer a tradeoff that allows one to determine how accurate the uncertainty predictions should be, and how much execution time should be spent to do this. The sub-ensembles in this configuration outperform both other methods that do not involve full ensembles, but as expected do not perform as well as a full ensemble.

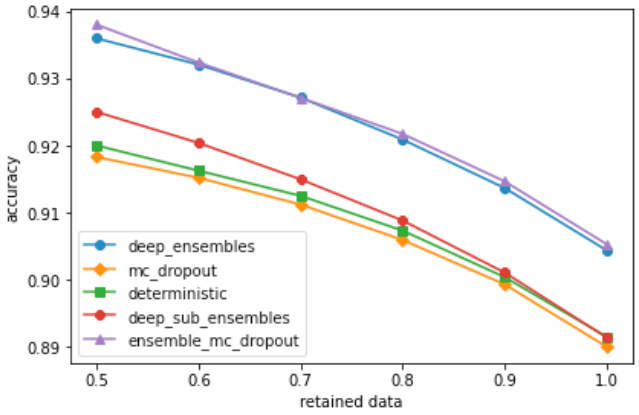

Fig. 4. Accuracy of various methods on 100 batches of the test dataset, with varying referral rates. Deep sub-ensembles have 3 convolutional layers in the task networks.

The ability of one to explore this tradeoff is best seen by viewing results from another configuration, where the sub-ensembles had smaller task networks. Figure 5 shows the results from this setup, where the sub-ensembles had only one convolutional layer (plus the final fully connected layer) as their task networks. The other difference in this plot is that the evaluation was performed on 20 batches of the test dataset. In this experiment, the performance of the sub-ensembles worsens to below that of Monte Carlo Dropout.

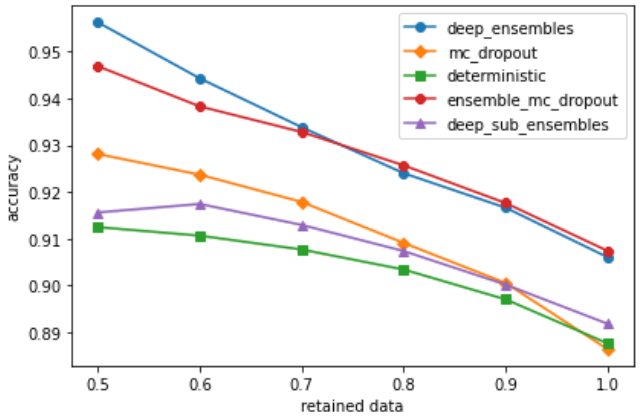

Fig. 5. Accuracy of various methods on 20 batches of the test dataset, with varying referral rates . Deep sub-ensembles have 1 convolutional layer in the task networks.

Figure 6 shows the results of the Area Under the receiver operator Curve (AUC) for the same evaluations seen in Figure 4.

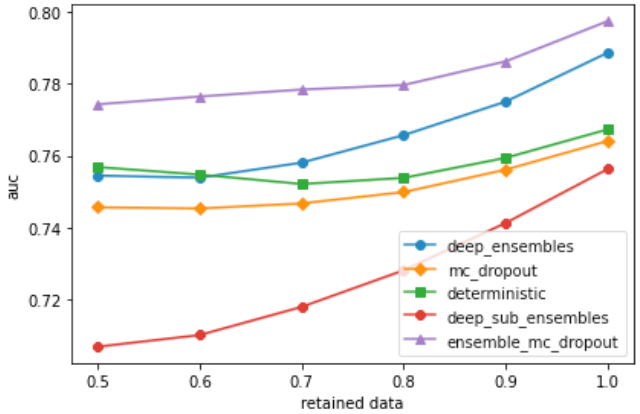

Fig. 6. Area Under the receiver operator Curve of various methods on 100 batches of the test dataset, with varying referral rates. Deep sub-ensembles have 3 convolutional layers in the task networks.

5.2. Execution Time

Deep Sum-Ensembles produce an uncertainty prediction faster than the full ensemble methods (Deep Ensembles and Ensemble Monte Carlo Dropout) in both cases. In our experiments, even when varying the size of the task networks, Deep Sub-Ensembles always offered a 3-10x speedup over Deep Ensembles. Varying the size of the task network does affect this number, but for any configuration where the trunk network is larger than the task network, the performance improvement is considerable.

Figure 7 shows example runtimes for evaluating each of the methods tested against 100 batches (of size 128) of the test dataset. Notably, Ensemble Monte Carlo Dropout is the slowest by a wide margin. This is expected as the number of predictions it must generate is multiplied twice: once for the number of ensembles, and then again for the number of dropout samplings tested with each ensemble.

```
deep_ensembles: 259.09s
mc_dropout: 270.61s
deterministic: 62.86s
ensemble_mc_dropout: 2687.87s
deep_sub_ensembles: 88.56s
```

Figure 7. Representative runtimes for evaluating the reviewed methods on 100 batches (of size 128) of the test dataset.

5.3. Discussion

The accuracy and runtime evaluations of the methods tested support the assertion of the original paper, that Deep Sub-Ensembles allow one to choose a tradeoff between execution time and uncertainty prediction ability. It is clear that increasing the size of the task networks, as expected, improves the uncertainty prediction. In the limit, increasing the size of the task networks to their maximum would be equivalent to the Deep Ensembles method, and performance would be the same.

The Area Under the receiver operator Curve (AUC) plots do not tell as clear of a story. Further discussion is reserved for Section 6.

6. CHALLENGES

The primary challenge in this work was the training of various methods that rely on ensembles. While it was tractable for us to train many different models and methods in different configurations, anywhere ensembles are introduced increases the time required by an order of magnitude. In practice, we found that this locked us into early versions of models that we

had started training. In order to create a complete and valid ensemble, we had to continue training models like the first one we had trained, even if in the meantime we had elsewhere implemented improvements that we would have wished to apply to the first trained models.

This was most evident with respect to the class imbalances present in the diabetic retinopathy diagnosis dataset. The dataset contains a ratio of roughly 4:1 healthy to unhealthy images. We started training some models before addressing this class imbalance, thinking that we would do so and then restart the training with the improvements. Unfortunately, we found that we did not have enough time to actually start over with the ensembles, since to produce one consistent evaluation plot across these methods requires training many independent models.

Though we are not certain, we believe this class imbalance may explain the strange results where AUC worsens (across all methods) as the referral rate increases and accuracy improves. This pattern was consistent across all of our tests. Future work could more closely inspect these results, as well as redo the evaluations after training the models with the dataset imbalance taken into consideration.

7. CONCLUSION

This work reproduced the findings of Deep Sub-Ensembles for Fast Uncertainty Estimation in Image Classification [4] on a new dataset. The diabetic retinopathy diagnosis dataset is more challenging than the baseline datasets tested in the original paper, so error rates cannot be compared directly. However we were able to compare the methods against each other on this dataset and observe their relative performance.

We were successfully able to observe the pattern proposed and observed in the original work, where Deep Sub-Ensembles allow one (by varying the size of their task networks) to explore the tradeoff between execution time and uncertainty prediction strength. It was true in our experiments that increasing the size of the task networks corresponded to improved accuracy not only of the model performance, but an improved ability of the uncertainty value to provide useful information. I.e. not only was accuracy higher, the slope of accuracy with respect to referral rate was also higher as the task network size increased.

The speedup observed was also consistent with the original work. In our experiments it varied between 3-10x compared to Deep Ensembles.

Whether or not this particular speedup is actually worth the cost of worsened uncertainty prediction is difficult to evaluate given this type of dataset. One would need either a performance metric or a dataset that inherently require a robust compromise between speed and accuracy. We are aware of such datasets for evaluating the classification ability itself, such as Streaming Perception [3]. However, we are not aware of a streaming dataset or metric that has to do with model uncertainty. The development of such a dataset or metric could be a unique contribution to the field.

One additional idea, suggested by Dr. Samson Cheung, would be to perform the task network trainings with different dropout samples of the trunk network. This would hopefully increase the variation between the task networks, and therefore their ensembling power, without any added compute cost. We agree this would be a good next step for work on this method.

8. REFERENCES

[1] Angelos Filos, Sebastian Farquhar, Aidan N. Gomez, Tim G. J. Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon and Yarin Ga, “A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks,” *Bayesian Deep Learning NeurIPS Workshop*, Dec. 2019.

[2] Balaji Lakshminarayanan, Alexander Pritzel and Charles Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems 30* pp. 6402-6413, 2017.

[3] Mengtian Li, Yu-Xiong Wang and Deva Ramanan, "Towards Streaming Perception," *European Conference on Computer Vision*, Springer, Cham, 2020.

[4] Matias Valdenegro-Toro, "Deep Sub-Ensembles for Fast Uncertainty Estimation in Image Classification," *Bayesian Deep Learning NeurIPS Workshop*, Dec. 2019.

[5] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015).

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, IEEE, 29(6): 82–97, 2012.

[9] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In ICML, 2016.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.

[11] Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.