The Blavatnik School
of Computer Science
The Raymond and Beverly Sackler
Faculty of Exact Sciences
Tel Aviv University

Mathematical foundations of machine learning

Comparison and Analysis of DNRF vs Simple CNN

0372-4009

Guy Hay

26/08/2020

Submitted to Professor Shai Dekel

Abstract:

This report analyzes the proposed DNRF model [6] and its result on MNIST and CIFAR10 databases as compared to simple CNN models. The DNRF model switches the final conventional SoftMax layer of a classification CNN model with a Random Forest layer. Therefore, this analysis will take simple CNN models as benchmarks and substitute the final layer with the proposed fully connected Random Forest layer. The implementation of DNRF was written in python[1], creating a new Keras layer and model. The models will be evaluated with a 5-fold cross validation with metrics of accuracy, precision, recall and top 2 accuracy. The results show an overall improvement in all metrics.

Introduction

Convolutional Neural Networks (CNN) [1,2] feature representations and classification have been found to outperform most conventional feature descriptors and classifier pipelines, even outperforming humans on the task of image classification.

Random Forest [3,4] have been found to outperform state-of-the-art learners in handling high dimensional data problems [5], and with their inherent ability to deal with multiclass problems makes them ideal learners.

The conventional final layer for CNN is a SoftMax layer. This layer while ideal for multiclass classification problems is simple and naive. Making the CNN feature detection work especially hard to obtain the proper results. For this reason, substituting the final SoftMax layer with a Random Forest which is ideal for high dimensional multiclass problems will easy on the need for CNN feature detections to find near perfect features, and in turn easy on the Random Forest need in learning special features. Therefore, CNN and Random Forests complement each other and should outperform both individual learners.

SoftMax

The SoftMax layer is the conventional final layer for a classification neural network. SoftMax described in Eq. 1 converts a vector z of K real numbers and normalizes it into a probability distribution of K probabilities. Due to its exponentials the input to the SoftMax function can be any real number vector whereas the largest number will get the highest probability. This creates a simple mapping of a non-normalized output of a neural network to a probability distribution over predicted output classes making it ideal for classification problems.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{n=1}^{k} e^{z_n}} \tag{1}$$

Stochastic Decision Trees – Random Forest Layer

The main problem in combining the CNN models to a Random Forest Layer is the non-differentiability of the Random Forest. This non-differentiability makes backpropagation impossible, therefore a stochastic differentiable tree was created.

The stochastic differentiable tree consists of nodes and layer just like a conventional tree, whereas each node in a stochastic differentiable tree, indexed by N, will not be a nonlinear function dictating left to right but a probability dictating left to right. Each node is described by Eq. 2 indicating the probability of going left and right respectively:

$$d_n(\boldsymbol{x};\boldsymbol{\theta}) = \sigma\big(f_n(\boldsymbol{x};\boldsymbol{\theta})\big);\ \bar{d}_n(\boldsymbol{x};\boldsymbol{\theta}) = 1 - \ d_n(\boldsymbol{x};\boldsymbol{\theta}) \tag{2}$$

[1] Github repository: https://github.com/guyhaytau/dnrf

Whereas σ is the sigmoid function, f_n is a real-valued function depending on the sample **x**, and the parametrization $\boldsymbol{\theta}$

Hence the probability of reaching each leaf node l is Eq. 3:

$$\mu_l(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{n\in N} d_n(\boldsymbol{x};\boldsymbol{\theta})^{I_{left}} \bar{d}_n(\boldsymbol{x};\boldsymbol{\theta})^{I_{right}} \quad (3)$$

I_{left} or I_{right} are indicator functions indicating if the leaf belongs to the left or right subtree of node n.

Finally, the probability given to each class is described by Eq. 4:

$$\mathrm{P}_T[y|\boldsymbol{x},\boldsymbol{\theta},\pi] = \sum_{l\in L} \pi_{ly}\mu_l(\boldsymbol{x}|\boldsymbol{\theta}) \quad (4)$$

Whereas L is the index of leaf nodes and π_l the probability distribution of leaf l.

Eq. 2,3,4 creates a stochastic differentiable tree and a log-loss term can be used for training:

$$L(\boldsymbol{\theta},\pi;\boldsymbol{x},y) = -\log(P[y|\boldsymbol{x},\boldsymbol{\theta},\pi]) \quad (5)$$

The probability and loss for a forest of k trees will be Eq.6 and 7 respectively:

$$P_F[y|\boldsymbol{x},\boldsymbol{\theta},\pi] = \frac{1}{k}\sum_{j=1}^{k} P_{T_j}[y|\boldsymbol{x},\boldsymbol{\theta},\pi] \quad (6)$$

$$R(\boldsymbol{\theta},\pi,B) = \frac{1}{|B|}\sum_{(\boldsymbol{x},y)\in B} L(\boldsymbol{\theta},\pi;\boldsymbol{x},y) \quad (7)$$

Whereas B is a mini-batch or all the training set, and T_j is the j-th tree of the forest.

In the used implementation, each tree will be fully connected to the final CNN layer and each mini-batch will train separately for each tree resembling the basic idea of Dropout[7], which is found to allow learning algorithms to generalize better and therefore, help with overfitting. All weights except the leaf probability distribution will be trained via a Stochastic Gradient Descent as described in Eq. 8:

$$\Delta\theta^{(t+1)} = m\Delta\theta^{(t)} - \eta\frac{\partial R}{\partial\theta}\left(\theta^{(t)},\pi;B\right); \quad \theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t+1)} \quad (8)$$

Whereas η the learning rate, and m the momentum.

During each epoch for each tree an accumulative value will be remembered. At the end of each epoch the value will be normalized and substituted for the leaf probability distribution as in Eq. 9.

$$\pi_{ly}^{(t+1)} = \frac{1}{Z_l^{(t)}}\sum_{(\boldsymbol{x},y')\in B} \frac{I_{y=y'}\pi_{ly}^{(t)}\mu_l(\boldsymbol{x}|\boldsymbol{\theta})}{P_T\left[y\middle|\boldsymbol{x},\boldsymbol{\theta},\pi^{(t)}\right]} \quad (9)$$

Whereas $Z_l^{(t)}$ is the normalizing factor that ensures that $\sum_y \pi_{ly} = 1$, and B the set of all training examples trained on the specific tree. In the used implementation, the leaf probability distribution was initialized as $\pi_{ly}^{(0)} = |y|^{-1}$, whereas y is the number of predicted classes.

VGG Models

VGG models are CNN models inspired by the VGG-16 architecture. Each VGG block is comprised of three layers: two convolutional layers and one MaxPooling layer. In each block the convolutional layers will have the same amount of 3x3 filters with the first block having 32 filters and each subsequent block doubling the number of filters as the previous block.

Benchmark models

To analyze how the complexity of the CNN model impacts the difference between the proposed Random Forest layer to the conventionally SoftMax layer, 3 models of 1, 2 and 3 blocks of VGG models were created and tested on the MNIST database. Due to relatively high metric values with low overfitting as can be seen in Appendix A, no dropout was used.

For the CIFAR models 3-block VGG where used and due to the high overfitting rate as can be seen in Fig. 2 another 3-layer VGG model with dropout of 25% was used. 25% was a rate tested to give low overfitting as can be seen from Fig. 1.

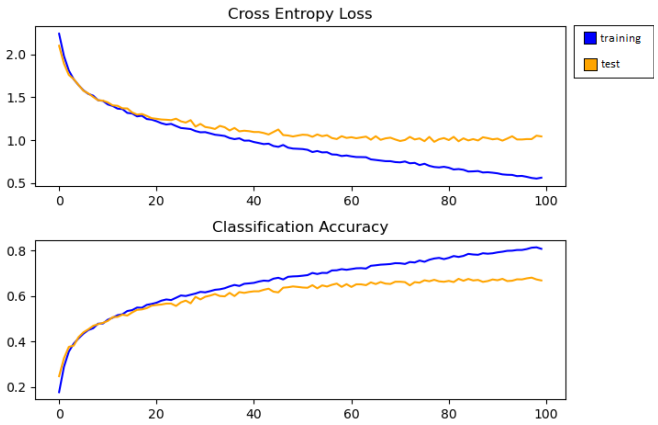

Figure 2: 3-layer VGG CNN benchmark on CIFAR10 databases loss and accuracy, relative high overfitting can be seen from the divergence of the training lines from the test lines

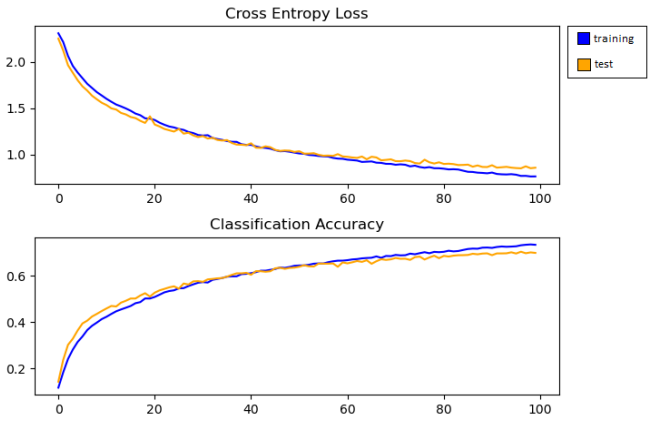

Figure 1: 3-layer VGG CNN benchmark on CIFAR10 databases loss and accuracy, relative low overfitting can be seen from the divergence of the training lines from the test lines

DNRF Analysis Models

The learning algorithm for each mini-batch in an epoch, chooses at random a different tree to train on and sums the contribution to the tree leaf parameters which are updated and normalized at the end of each epoch. This strategy resembles the basic idea of Dropout so an improvement with overfitting is expected on top of the general increase in accuracy because of multiple classifiers. Therefore, three different number of trees were tested with 5, 10, 15 trees each.

Due to the learning algorithm updating after each epoch the leaf parameters for each tree, with the sum of the contribution from all mini-batches trained on a specific tree, it is expected that mini-batch sizes will have an impact on convergence rate and overall accuracy. This is because the smaller the mini-batch the more times a tree will be trained and the final update at each epoch to the leaf parameters will be more polarized and subsequently will converge faster. Therefore, two different mini-batch sizes were used 100 and 500 on 3-block VGG with 10 trees and 5 layers.

Results

Due to computational limits all experiments were limited to 100 epochs.

Table 1 holds results of comparison of the 1,2,3 VGG blocks each with 10 trees and 5 layers compared to their benchmarks trained on MNIST dataset.

	1 VGG block with RF	2-block VGG with RF	3-block VGG with RF	1-block VGG benchmark	2-block VGG benchmark	3-block VGG benchmark
Accuracy %	98.493 ± 0.076	98.736 ± 0.045	**99.070 ± 0.036**	98.141 ± 0.069	98.683 ± 0.044	98.720 ± 0.067
Precision %	98.753 ± 0.055	98.912 ± 0.063	**99.169 ± 0.041**	98.421 ± 0.057	98.772 ± 0.043	98.791 ± 0.068
Recall %	98.227 ± 0.093	98.826 ± 0.083	**99.009 ± 0.036**	97.933 ± 0.069	98.607 ± 0.051	98.653 ± 0.057
Top 2 Accuracy %	99.616 ± 0.048	99.733 ± 0.046	**99.761 ± 0.041**	99.442 ± 0.020	99.723 ± 0.048	99.719 ± 0.045

All 3 models achieved better results than their corresponding benchmarks, implying that the substitution will improve the results regardless of the CNN complexity. This of course is yet to be determined for more complex models but can be said to very simple and shallow CNN models.

Table 2 holds results of comparison of the same model, a 3-block VGG with 10 trees and 5 layers trained on different batch sizes 500 and 100.

	Mini-Batch Size of 500	Mini-Batch Size of 100
Accuracy %	60.4 ± 1.4	**73.38 ± 0.67**
Precision %	**78.54 ± 0.92**	75.23 ± 0.88
Recall %	40.9 ± 1.6	**72.36 ± 0.61**
Top 2 Accuracy %	79.44 ± 0.85	**86.71 ± 0.52**

Table 1: results for 3-block VGG with 10 trees and 5 layers trained on CIFAR10 dataset while using 500 and 100 mini-batch size

The results in Table 2 show a clear advantage for the model trained with a mini-batch of size 100 in all metrics except in precision. As explained in DNRF Analysis Models, with smaller mini-batches the leaf parameters polarize faster and therefor the model gives a more curtain prediction. This more curtain prediction results in a higher accuracy, recall and top 2 accuracy but in a lower precision. Through all the DNRF experiments the precision decreased and at some point, started to raise a bit, displaying a shifted parabolic shape. Hence, these results align with the theory that smaller mini-batches allowed the algorithm to converge faster. Another indication for the faster convergence, are Fig. 3,4 which show the loss and accuracy during training. Both figures clearly show that while the model trained on 100 mini-batch size converged and even overfitted, the 500 mini-batch size's slope is still substantially greater than 0 indicating it still has not converged.

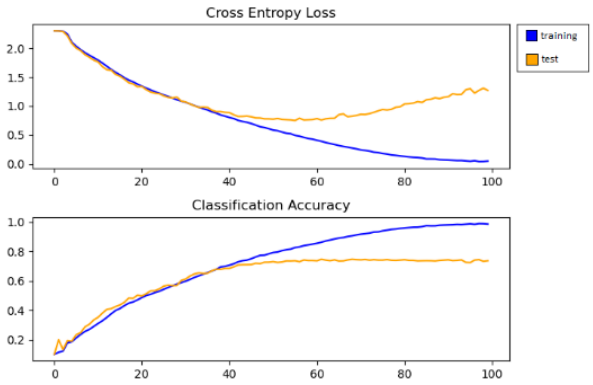

Figure 4: 3-block VGG with 10 trees and 5 layer trained on CIFAR10 with mini-batch size of 100. The test accuracy and loss converge.

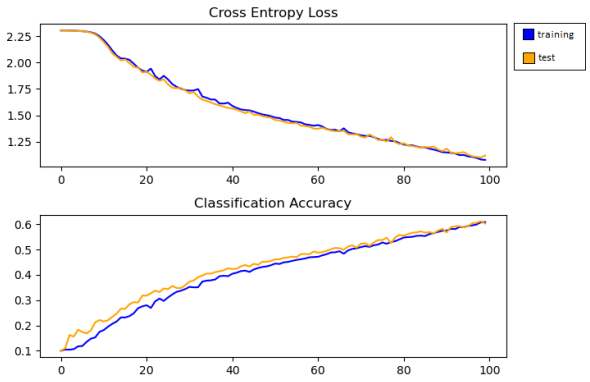

Figure 3: 3-block VGG with 10 trees and 5 layer trained on CIFAR10 with mini-batch size of 500. The test accuracy and loss still tend higher and hasn't converged yet.

CIFAR10 Model Results:

Table 2 holds results of the 3 DNRF models with 5,10 and 15 trees without a Dropout layer, 2 DNRF models with 10 and 15 trees with a Dropout layer, compared to each other and their benchmark, trained with mini-batch size of 100.

	5 trees	10 trees	15 trees	10 trees with Dropout	15 trees with Dropout	3-block VGG	3-block VGG with Dropout
Accuracy %	70.65 ± 0.51	73.38 ± 0.67	72.28 ± 0.64	77.61 ± 0.68	**78.43 ± 0.72**	66.53 ± 0.59	70.35 ± 0.37
Precision %	72.07 ± 0.51	75.23 ± 0.88	74.23 ± 0.68	81.12 ± 0.62	**82.85 ± 0.51**	72.25 ± 0.56	79.86 ± 0.61
Recall %	69.78 ± 0.51	72.36 ± 0.61	71.10 ± 0.61	75.13 ± 0.86	**74.94 ± 0.72**	61.45 ± 0.70	61.11 ± 0.96
Top 2 Accuracy %	84.45 ± 0.12	86.71 ± 0.52	86.11 ± 0.46	90.17 ± 0.53	**90.57 ± 0.63**	82.89 ± 0.31	85.72 ± 0.31

Table 2: Metric results for 3-block VGG benchmarks and DNRF on CIFAR10 dataset with varing number of trees with and without a Dropout layer

The first element that stands out is that 15 trees achieved lower results than 10 trees. This result suggests that there is an upper limit to the number of trees that can be added and still have a positive impact on the results. With the Dropout layer 15 trees has achieved the best result over all models and specifically compared to 10 trees with Dropout, which achieved the second highest results from the models. Also, this analysis is not comprehensive enough to fully determine, but it does suggest that the optimal number of trees is dependent on the CNN architecture.

As can be seen from Table 2 and more clearly from Fig. 3,4,5,6,7 that despite a strong Dropout like feature and improvement from 5 trees to 10 trees, a higher improvement was achieved using Dropout rather than adding more trees. This implies that more trees will not necessarily have a positive outcome for overfitting and more conventional methods should be used.

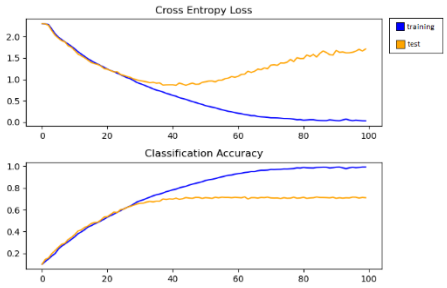

Figure 6: 3-block VGG with 5 trees loss and accuracy trained on CIFAR10, visible overfitting around epoch 30

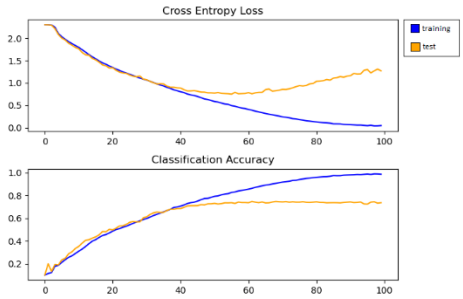

Figure 5: 3-block VGG with 10 trees loss and accuracy trained on CIFAR10, visible overfitting around epoch 40

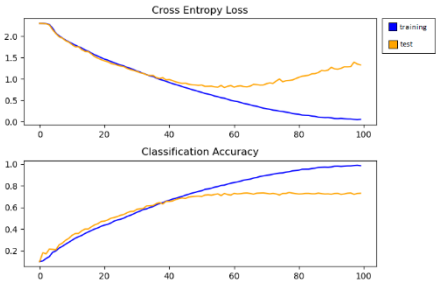

Figure 7: 3-block VGG with 15 trees loss and accuracy trained on CIFAR10, visible overfitting around epoch 40

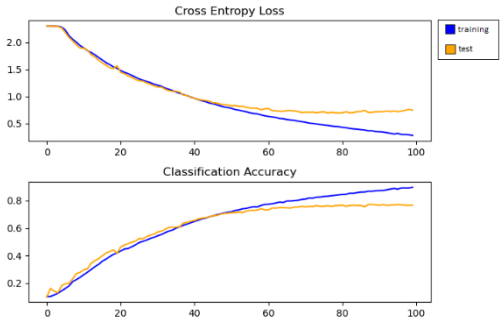

Figure 9: 3-block VGG with Dropout and 10 trees loss and accuracy trained on CIFAR10, visible overfitting around epoch 50

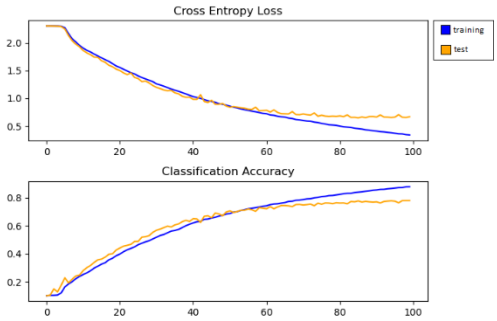

Figure 8: 3-block VGG with Dropout and 15 trees loss and accuracy trained on CIFAR10, visible overfitting around epoch 55

Analyzing Wrong Predictions:

From the comparison of the wrong predictions of the 3-layer VGG with Dropout and 15 trees, and its benchmark on the CIFAR10 dataset it was shown that 72% of the classifications that the DNRF model got wrong the benchmark also got wrong, indicating that both architectures created a similar classification function. This increases the idea that switching the final layer to Random Forest has a positive effect on the classification function and does not just create a better function for certain problems.

In Table 2 it was shown that there is more than a 12 and 15 percent difference between the accuracy and the top 2 accuracy of DNRF and its benchmark, respectively. Indicating that some images, due to the low resolution, are difficult to be distinguished between 2 classes. Fig.11 and 10 are examples of such images that were classified wrong. Figure 11 is a cat that was classified as a dog, predicting dog and cat with the following certainties respectively: 43%, 42%. Figure 10 is a truck that was classified as car, predicting car and truck with the following certainties respectively: 60%, 37%.

Figure 11: A cat that was predicted as a dog with 43% and as a cat with 42%

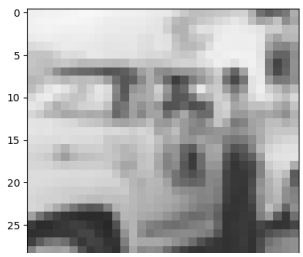

Figure 10: A truck that was classified as a car with 60% and as a truck with 37%

	Percentage of wrong labeling from total wrong [%]	Label that the algorithm wrongly labeled the most [%]	Second most wrongly labeled [%]	Third most wrongly labeled [%]
Cats	26	Dog – 46	Deer – 16	Horse – 11
Birds	18	Deer – 34	Cat - 14	Dog – 12
Dogs	15	Cat – 44	Horse - 26	Deer – 18

Table 3: main classification errors made by the algorithm with the three most common labels that were mislabeled and the corresponding most common labels that the algorithm classified them. After the first three all wrong labels per label are below 7%

From Table 4 it is shown that the algorithm main problem is the classification between dogs and cats were cats were misclassified 26% wrong with 46% of the times as dogs. With no surprise dogs were as well classified wrong with 15% wrong from the total wrong predictions with 44% of the times as cats. This indicates that the algorithm did not put enough weight or could not extract enough features that have to do with faces or minor details that are used to determine between dogs and cats. Birds misclassification is also an indication to the same problem that the algorithm is not enough detail oriented.

Unsurprisingly, because the high common errors, the same misclassifying problem occurred on the benchmark with cats, deer, birds, and dogs misclassified 17%, 16%, 14%, 12% from all wrong predictions, respectively. Showing the same misclassifying relations as with the DNRF model with 43% of cats misclassified as dogs and 40% of dogs misclassified as cats.

From the comparison of the wrong predictions of the 3-layer VGG with Dropout 10 trees and its benchmark on MNIST dataset it was shown that only 27% that the DNRF model got wrong the benchmark also got wrong, indicating that those images are harder to create a function that accurately predict them. But despite the similarity between the classification functions trained on the CIFAR10 dataset, the classification functions trained on the MNIST dataset did not display the same errors. Due to the high accuracy values on both models the low common wrong predictions do not indicate a different classification functions with different strength and weakness but more of somewhat random errors. There is not a strong correlation with the misclassification error between the two models but there are some similarities, that also occur in our human experience, as can be shown with Fig. 9, like misclassification 4 as 9 where in the DNRF model it accounted for 59% of the 9 misclassification and in the benchmark 63%.

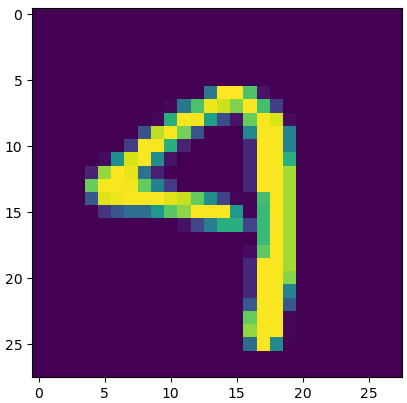

Figure 12: A 4 misclassified in both DNRF and benchmark models as a 9

Conclusion

It was shown that the proposed substitution improved results in all metrics in all VGG models tested, indicating that it can be used freely with shallow and deep CNN models, also this analysis is not comprehensive enough to reach an absolute conclusion. An indication to a limitation that an optimal number of trees exists, was found and moreover this optimal number is correlated to the CNN model's complexity, while a more comprehensive test should be carried out. Another limitation is convergence rates which were found to be highly sensitive to mini-batch sizes. In addition, although an improvement with overfitting was expected due to how the algorithm updates his final leaf parameters, a minimal improvement was observed, and more conventional methods should be used.

Appendix

Appendix A – Loss and Accuracy Graphs for 3-Layer VGG CNN on MNIST database showing low overfitting:

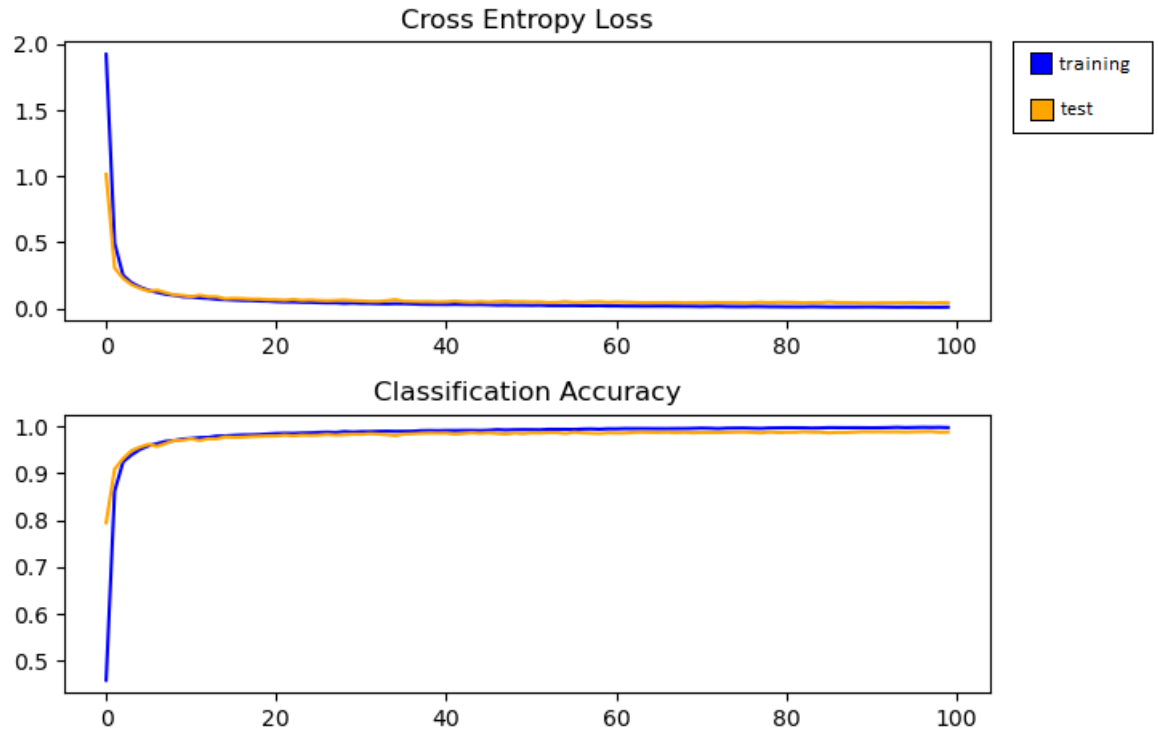

Figure 13: Loss and Accuracy for 3-Layer VGG CNN on MNIST database, closeness of training accuracy and loss to test accuracy and loss indicates low overfitting

References

[1] Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arxiv1409.1556

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

[3] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. (NC), 9(7):1545–1588, 1997

[4] L. Breiman. Random forests. Machine Learning, 45(1):5– 32, 2001

[5] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In (ICML), pages 96–103, 2008

[6] P. Kontschieder, M. Fiterau, A. Criminisi, S. Rota Bulo, Deep neural decision forests, ICCV 2015

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15:1929–1958, 2014