Classification on Image Data

Seraphin Bassas, Hays Johnson, Clemence Granade
McGill University

March 2022

Abstract

In this project, we compared the accuracies of neural network architectures on the benchmark dataset Fashion-MNIST. We compared seven different networks: a CNN; one-layer, two-layer, and three-layer MLPs with ReLU activation; three-layer MLPs with hyperbolic tangent and Leaky ReLU activation; and a three-layer MLP with dropout regularization. To train the models, we implemented mini-batch gradient descent with backpropagation. We noted that hyperparameter tuning significantly affected the accuracies obtained, observing differences of up to 13% in MLPs and 80% in the CNN between best and worst accuracy measured. Additionally, we noticed that the MLPs took longer to train than the CNN, particularly for the three-layer models. The addition of drop regularization and Leaky ReLU activation increased the models' generalization capabilities and permitted the three layer MLP to achieve accuracies of 70.84% and 72.81%, respectively.

Introduction

The objective of this project is to determine the optimal MLP network architecture for image classification. To this end, we implement and tune the learning rate and batch size of seven models on the Fashion-MNIST dataset for model selection. Subsequently, we perform a final training and testing phase on 10,000 data samples (in the interest of available computational resources). In order to create a model closest to the existing benchmarks, we built a CNN using functions from the Keras library. Then, to compare the effect of depth on the accuracy, we built one, two and three-layer MLPs with ReLU activation functions. We additionally wanted to compare the impact of the activation functions and avoiding vanishing gradients; thus, we implemented models with Leaky ReLU and tanh activation functions as well. Finally, knowing the impact of data preprocessing and high variance on a models, we tested our model on unnormalized data and with dropout regularization. The Fashion-MNIST dataset has been a highly popular ML dataset in recent years: Google Scholar lists over 7,000 papers citing the dataset since 2018. Much of this research has focused on developing CNN performance and methods, as these networks remain state-of-the-art (SOTA) for image classification. Tanaka and Tamukoh 2022 used Fashion-MNIST to demonstrate that a reservoir computing-based CNN, optimized using ridge regression, can achieve higher accuracies than other proposed models (Tanaka and Tamukoh 2022). The dataset has been used to develop a modified ThiNet pruning algorithm, which prunes CNNs in a fraction of the time as standard ThiNet while producing similar reductions in accuracy (Tofigh, Ahmad, and Swamy 2022). Fashion-MNIST has also been used in unsupervised image classification, such as in developing novel clustering techniques that do not require a previously-defined number of clusters (Chang and Leung 2022) and in demonstrating the efficacy of deep generative classifiers with variational autoencoding (Yang, Fan, and Bouguila 2022).

Dataset

A single dataset was used for this project: the Fashion-MNIST dataset. This dataset is composed of 60,000 training and 10,000 testing images over 10 categories, and serves as an alternative to the original MNIST dataset. Images are the same resolution as its precursor (28x28 grayscale pixels), but are generally thought to be more difficult to classify. The preprocessing involved is minimal, requiring just data centering, pixel normalization, and one-hot encoding of the labels. Generally speaking, non-zero value pixels appear more often towards the center of the image, placing less importance on fringe pixels. The advantage of using the Fashion-MNIST dataset is that it is a well-understood and well-known dataset in the Image Classification (IC) community. Further, loading and processing of the dataset has been streamlined in the APIs of popular IC libraries, accelerating that phase of model development. Finally, one last advantage of using this dataset is the ability to benchmark model performance compared to SOTA models with online resources such as Kaggle (Camporese 2020) or Papers With Code (Unknown 2022).

Results

In order to find the optimal model, we compared a Convolutional Neural Network to MultiLayer Perceptrons with varying architectures obtaining the following testing accuracies:

Model Architecture	Accuracy(%)	Training time (mins)
CNN	85.76	1
1 Layer ReLU MLP	75.14	18
2 Layer ReLU MLP	61.49	N/A
3 Layer ReLU MLP	66.19	18
3 Layer Leaky ReLU MLP	72.81	56
3 Layer Tanh MLP	51.01	27
3 Layer ReLU MLP + dropout	70.84	48
3 Layer ReLU MLP + dropout (256 units)	71.11	83
3 Layer ReLU MLP - data norm.	77.81	28

Table 1: Performance comparisons of different model architectures.

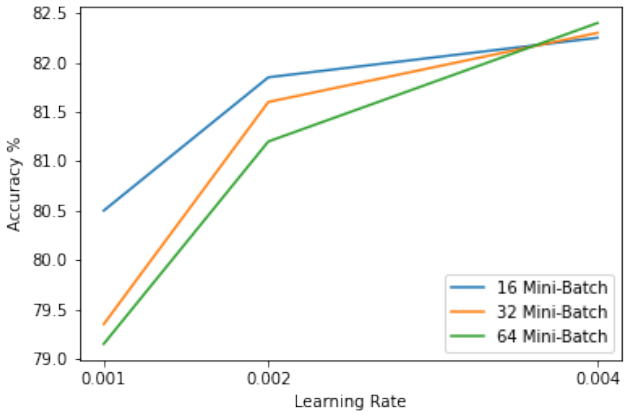


Figure 1: One-Layer MLP Accuracy with respect to Hyperparameters

MLPs

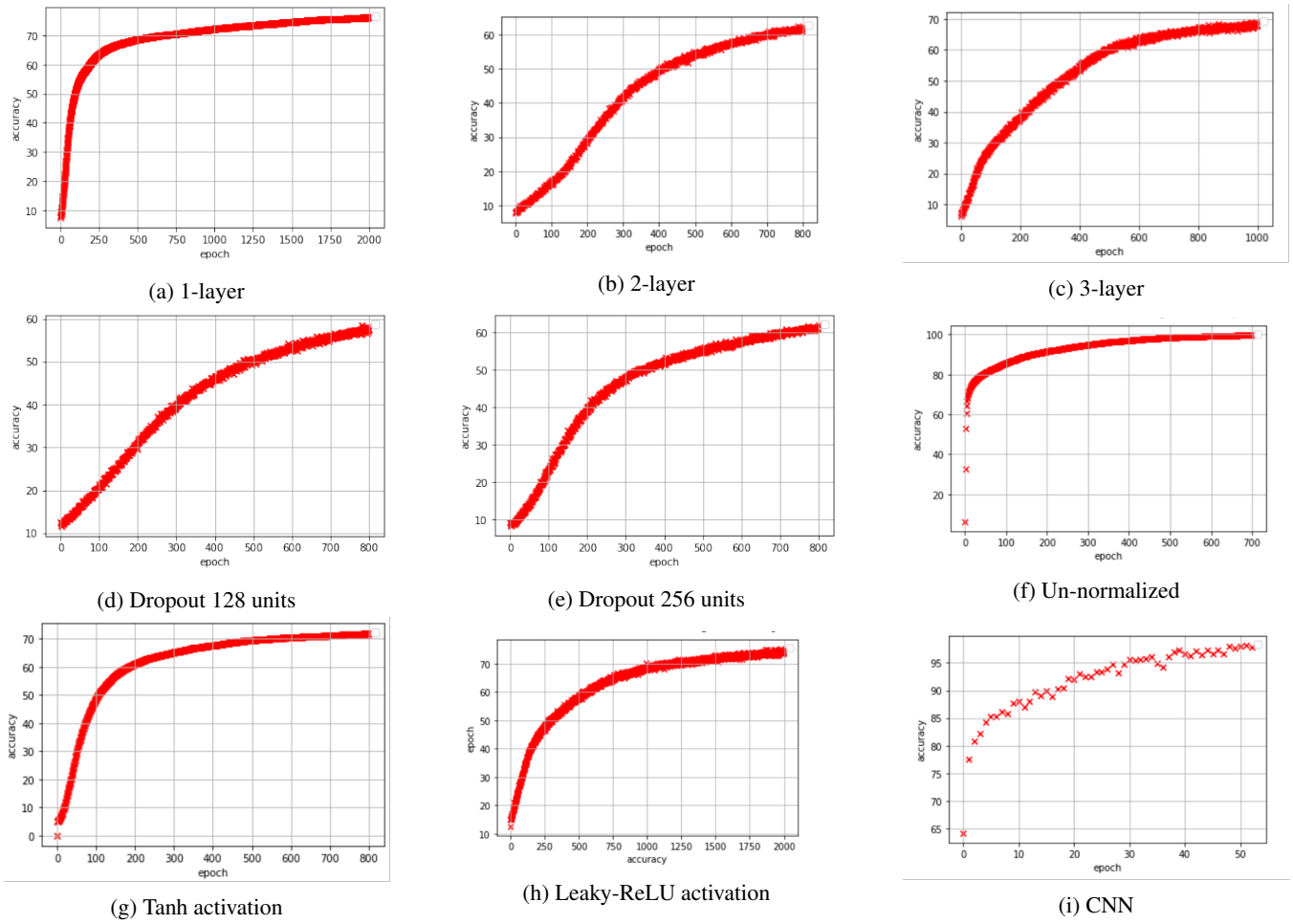
The project consisted in implementing seven MLPs: a three layer ReLU MLP as baseline, two models to compare depth, two models to analyse the impact of the activation function, one model to test the impact of regularisation, and finally another to test the impact of limited data pre-processing. Each hidden layer contained 128 units, and the output activation function for each of these models was a softmax layer, as adapted for multi-class classification. Each model was trained using a mini-batch SGD, implemented from scratch. Using sklearn k-fold cross validation on a small subsample of the dataset, we tuned the learning rate and batch size of the gradient descent.

MLP depth In order to analyse the impact of depth on our MLPs we built a one layer and a two layer MLP, both utilizing the ReLU activation function and softmax for classification. We adjusted the gradient and backpropagation computations accordingly for each, and compared the classification results with those of the 3-layer model. We hypothesized that the depth of the model would increase its expressive power and accuracy would increase proportionally. Surprisingly, the best performing model was the 1-layer (75.14%), followed by the 3-layer (66.19%) (Table 1).

Increasing depth for neural networks is usually a double-edged sword in the sense that more-depth means more parameters, but also the possibility of extracting higher level features, at the expense of higher chance of overfitting the data. While, we expected the model to take longer to train for a gain in accuracy, our results suggested otherwise; the one-layer model took as long to train as the three-layer (18 mins) for more than twice the number of training iterations (around 2000) (Figure 2a), but predicted the correct class almost 9% more often (see Discussion). Since it seems that the two- and three-layer models could have used some more training iterations (Figures 2b, 2c), an interesting future experiment would be to compare testing accuracy when more computational resources are available, as perhaps letting them train longer would yield different results.

Activation Functions on MLPs To investigate the impact of activation function choice on model performance, we conserved one model architecture and all hyperparameters, controlling solely for activation function. The chosen architecture was the three-layer MLP (learning rate = 0.002, batch size = 32, hidden units =128) and the three activation functions were ReLU, Leaky-ReLU ($\alpha = 0.1$), and Tanh. From the baseline ReLU model which achieved 66.19% accuracy, the Leaky-ReLU model notched a 6% boost (72.81%), while tanh represented a 15% decrease (51.01%) (Table 1). This highlights the importance of activation function choice on model performance. Here, the performance degradation we observe may be due to tanh awarding a gradient of 1 only when the input has value 0, leading to dead neurons. Further, tanh is also prone to the vanishing gradient problem. However, there are different architectures and tasks in which different activation functions may be useful. For instance, tanh is widely used in Natural Language Processing in the context of Recurrent Neural Networks (Nwankpa et al. 2018). And finding out whether dead neurons or vanishing gradients are at the root of this performance degradation would require further investigation into the model weights and gradients. Moreover, while the choice of Leaky-ReLU over other activation function may be justified in this case, these experiments do not suggest our results would extend to other models and architectures. Figure 2g shows that tanh trains well reaching above 70% training accuracy, but test time accuracies tell a different story. This is the same pattern one could expect from an overfitting model, but tanh generalizes too well for that to be the case.

Regularization Multiple methods exist to reduce overfitting and variance, but dropout is particularly adapted to MLP, which we therefore implemented on the hidden activation layers of a 3-layer model with ReLU. We experimented with models that had 0.5 and 0.2 probability of dropping a node, respectively, and found that $p = 0.2$ balanced generalization with variance nicely, maximizing accuracy. We ob-



(a) 1-layer (b) 2-layer (c) 3-layer

(d) Dropout 128 units (e) Dropout 256 units (f) Un-normalized

(g) Tanh activation (h) Leaky-ReLU activation (i) CNN

Figure 2: Training accuracy per epoch for different architectures.

served an increase in classification accuracy over the vanilla three-layer model, but we reasoned that dropout would have more substantial effects on more complex models. As such, we also tested dropout regularization on a 256 hidden unit model. Like increasing depth, increasing hidden-layer width can lead to overfitting and higher model variance. We hoped to see dropout counter these effects, allowing the higher expressiveness to ultimately improve testing error. While our test accuracy did improve to 71.11% (from 70.84%) (Table 1), demonstrating an overall improvement and lack of overfitting, this gain is negligible, and when weighted against the increase in runtime; overall, probably not justifiable. An interesting future study to do would be to see whether dropout has the same effects on the CNN, however any difference would require further testing as CNNs also have parameter sharing which might be a confounding variable of any results.

Data Preprocessing Surprisingly, the 3-layer MLP model that was trained on the same batch of unnormalized data (i.e. features were not scaled to [0:1] range) performed better (77.81%) (Table 1) than the same model that was trained on normalized data. In a different scenario, we could infer that this difference is due to the different scalings of each feature. However, a property of image data is that each pixel is bounded by the same upper and lower values, therefore, features scale identically with respect to each other whether they all fall in the [0:255] or [0:1] ranges. Perhaps the fault lies in the numerical properties of each range (e.g. [0:255] and [0:1]) as larger values interact differently with the model.

Similarly to the Tanh activation model, the unnormalized data model does very well in training (almost 100% accuracy) (Figure 2f), but as mentioned above, it does significantly worse during testing. Nevertheless, it still achieves the highest testing accuracy among all MLPs tested.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of neural network architecture. CNNs build on the MLP architecture by introducing the added network properties of extracting location information (vis-à-vis the inputs) and the sharing of weights between nodes. Further, as Graphical Processing Units (GPUs) improved and matrix operations were optimized, the use of convolution operators and the learn-

ing of convolutional filter weights to extract location information became trivial. Put together, these properties have made CNNs dominant in object recognition and image classification competitions, such as Image Large Scale Visual Recognition Challenge (Russakovsky et al. 2015), and represent the current SOTA. As such, we want to investigate how our MLP model competes with a CNN on the same image classification task. In that light, we tuned and trained a neural network on the Fashion MNIST dataset using the Keras API from the TensorFlow python library (Chollet and others 2015).

Using a sequential model with two Convolutional layers, interleaved with Max-Pooling layers, and followed by two densely connected layers, we create a model with over 623 thousand parameters. We train the CNN with categorical cross entropy loss and the Adam optimizer, letting it converge until the validation accuracy stops improving over a period of 10 epochs. We performed hyperparameter tuning of the CNN using Keras Tuner (O'Malley et al. 2019) and used the optimal combination of parameters: learning rate = 0.0001, number of filters = 64, and kernel size = (7,7). Then, mimicking the training and testing setups for the MLPs, we train the network on 10 thousand training examples and set the batch size to 32. After 43 iterations in just over a minute of training with Google Colab GPUs (Figure 2i), the model converged and yielded a 85% classification accuracy on unseen data (Table 1). This remarkable efficiency and speed highlights the optimization of the Keras library, and the power of being able to harness hardware accelerators. Further, when all the training data was available, the model yielded even better test results (90.01%), showing that this model architecture really is more powerful than MLPs, and it can even train faster.

Discussion and Conclusion

The CNN achieved accuracies similar to the SOTA IC errors, while the MLP did slightly less well. While the hyperparameter tuning and architecture choice added 15% accuracy relative to the average obtained across tuned CNNs, at an accuracy of 85.76% the CNN out performed our best MLP at 77.81%. Additionally, while increasing complexity did generally increase performance, the deeper MLPs took much longer to train than the CNN. Further, the increased complexity does not account for the results of the one-layer MLP. We speculate that the one-layer model outperformed its counterparts as a realization of the manifold hypothesis; the one-layer model was able to pick up on a data manifold that was lower dimensional than the patterns the more complex networks were picking up on. From our readings, current CNNs compose most SOTA models (Unknown 2022; Camporese 2020), but there exist alternative computational methods to chose between the efficiency and accuracy tradeoff.

Further interesting results are our final test accuracies for the tanh activation MLP and the MLP trained on unnormalized data. These two models are the only two models whose maximum training accuracies were significantly higher than their test accuracies ($> 20\%$ difference for both). In the tanh activation function case, the model also achieved a final accuracy much lower than the other three-layer MLPs; we speculate this is the result of vanishing gradients. The vanishing gradient problem is a well-known neural network issue, and has been known to arise when using activation functions such as logistic sigmoid and hyperbolic tangent (Nwankpa et al. 2018). Vanishing gradients hamper the model's ability to learn, render the training process inefficient, and can even make whole layers virtually useless (Liu et al. 2021). As such, we believe the model did not learn the data well at all, and while it was able to obtain reasonable training accuracies, it ultimately could not generalize on the test data.

In the un-normalized case, the model did perform quite well; in fact, it had the highest accuracy out of all MLPs. We hypothesize that despite the overfitting demonstrated by the drop in test accuracy, the model was overall still able to learn the data better when it was not normalized. It is possible that the model was using irrelevant features to better learn the data, such as the absolute magnitude of the pixel gray-scale values. Despite performing well in IC tasks, MLPs have a tendency to learn the data in undesirable ways, based not on feature detection but other factors and properties of the data, as mentioned in lecture. For this result and the previous one, more experiments would need to be conducted to identify exactly what is occurring in these models.

To conclude, our experiments yielded interesting results, notably that the effect of activation function on model accuracy is large, despite minor changes in the function. Further, whether we consider the number of layers or weights in the model, the one-layer MLP proved that model complexity does not necessarily mean better performance. Finally, while MLPs can be tuned and modified to increase classification accuracy, it seems that CNNs are just better suited models for this task; even still, the one we used could have been further improved.

Statement of Contribution

For this project, Séraphin implemented the CNN and the minibatch function while adapting the gradient descent, Clemence wrote the gradient descent functions and cleared the implementation of the tanh and Leaky ReLU and Hays helped with the gradient descent and implemented the dropout. All three wrote the report and ran the models.

References

Camporese, G. 2020. Machine learning project - image classification. https://www.kaggle.com/competitions/image-classification-fashion-mnist/leaderboard.

Chang, J.-H., and Leung, Y.-C. 2022. Dynamic image clustering from projected coordinates of deep similarity learning. *Neural Networks*.

Chollet, F., et al. 2015. Keras. https://github.com/fchollet/keras.

Liu, M.; Chen, L.; Du, X.; Jin, L.; and Shang, M. 2021. Activated gradients for deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Nwankpa, C.; Ijomah, W.; Gachagan, A.; and Marshall, S. 2018. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L.; et al. 2019. Keras Tuner. https://github.com/keras-team/keras-tuner.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Tanaka, Y., and Tamukoh, H. 2022. Reservoir-based convolution. *Nonlinear Theory and Its Applications, IEICE* 13(2):397–402.

Tofigh, S.; Ahmad, M. O.; and Swamy, M. 2022. A low-complexity modified thinet algorithm for pruning convolutional neural networks. *IEEE Signal Processing Letters*.

Unknown. 2022. Image classification for fashion mnist. https://paperswithcode.com/sota/image-classification-on-fashion-mnist.

Yang, L.; Fan, W.; and Bouguila, N. 2022. Robust unsupervised image categorization based on variational autoencoder with disentangled latent representations. *Knowledge-Based Systems* 108671.