Classification of Image Data

Ramin Akhavan, Ryan Sowa, Samuel Torres

Abstract

In this paper, we implemented a fundamental machine learning model: the multilayer perceptron (MLP). For this implementation, we implemented a MLP class which would take as input a list of Neural Net layers. We also implemented a optimizer which we used to train our MLP using gradient descent. To test our implementation, we examined a benchmark dataset: Fashion-MNIST. First, we vectorized our data to have the appropriate dimensions to train our MLP. Then, we proceeded to evaluate our implementation by classifying image data in the Fashion-MNIST dataset. We conducted experiments on the model, analyzing factors such as effects of different activation functions, adding $L2$ regularization, training with unnormalized images, and adding dropout. We also implemented a simple ConvNet and compared its accuracy to that of our MLPs. We discovered that the ConvNet outperformed our best MLP, scoring about 4% higher in accuracy.

1 Introduction

Image classification, a main topic in the computer vision discipline, is used in a myriad of applications across different areas such as medical imaging, autonomous driving, security, etc. Machine learning has proved to be a go-to approach to tackle these problems relevant in this practice. Specifically, deep neural networks have outperformed most of the classical machine learning methods when it comes to classification of unstructured data, such as images.

Multilayer feedforward perceptrons (MLPs) are the most basic form of a neural network. They consists of a finite number of successive layers, each layer with a finite number of units connected to the subsequent layer where information travels from one layer to the next. We call backpropagation, a gradient descent method [1], the learning algorithm used by these neural networks. These models have been widely used in the past [2] for image classification. However, Convolutional Neural Networks (ConvNets or CNNs) suggested a superior approach when LeNet [3] was introduced in 1998 for classifying handwritten digits, and then when AlexNet [4] was proposed in 2012, outperforming MLPs and wining the ImageNet challenge that year,

The objective of this study is to test and compare both MLP and ConvNet architectures in the Fashion-MNIST dataset. There has already been a large extent of research conducted on this dataset using ConvNets, such as [5] and [6]. Here, we implemented several MLP configurations and conducted similar experiments as described in this research to train the “best” (highest accuracy) MLP. We aimed to test the effect of hidden layers, hidden units, and activation functions on the performance of these models. We also studied the effect of regularization implementing $L2$ and dropout techniques as a way of preventing over-fitting in the models [7, 8]. Our results demonstrate the effect that the aforementioned parameters have on the model performance as well as the advantages of using ConvNets instead of MLPs for image classification tasks.

2 Datasets

The **Fashion-MNIST** dataset [9] contains 60,000 images as training samples and 10,000 samples for testing. All images are grayscale and 28x28 in size. Each training and test sample is assigned to one of the 10 labels/class displayed in Fig.1. The dataset is balanced, having the exact number of samples per label, 6000 per class for the training set and 1000 per class for the test set. As an example, we visualized the histograms of the pixel intensities in Fig.2 for each of the displayed classes. As expected, most of the values in these images were 0 (or black pixels), but we also observed that depending on the class, higher pixel intensities may appear more often. For example, we observed that a "Sandal" sample may have less higher intensities in comparison to a "T-shirt/top" or an "Ankle boot" image, as these could occupy the majority of the image, hence having more non-zero pixels. This is relevant as the pixel intensities will be the values/data that the models will to learn from.

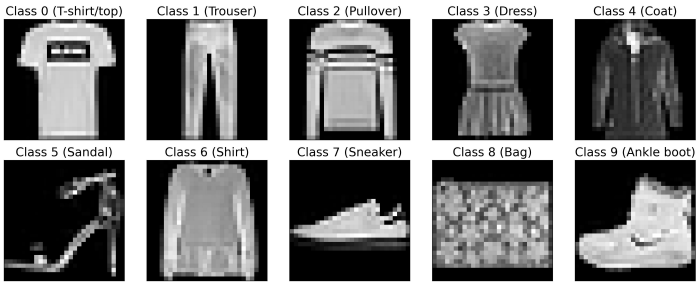


Figure 1: Fashion-MNIST labels/classes

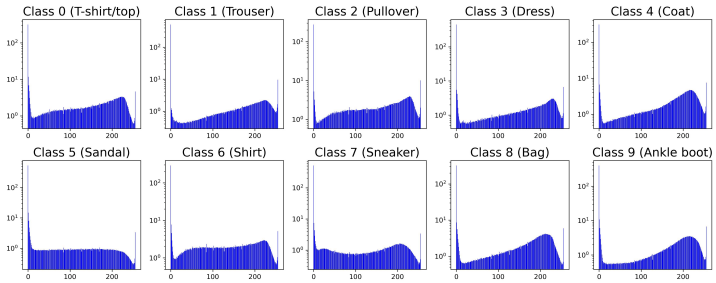


Figure 2: Fashion-MNIST histograms (log-scale) of examples in Fig. 1.

Training scheme

We trained all of our MLP models using mini-batch gradient descent with a batch size of 100 samples over 100 epochs. The only exception is our ConvNet, where we used Adam optimization (stochastic gradient descent) [10] with a batch size of 64. Cross-entropy loss was used as target function in all of the experiments.

3 Results

Testing hidden layers (h)		
Model	Hidden layers (h)	Accuracy
1	0	0.8381
2	1	0.8650
3	2	0.8726

Table 1: Testing effect of hidden layers.

Testing learning rate (α)	
Learning rate (α)	Accuracy
0.1	0.8663
0.01	0.8726
0.001	0.8397

Table 2: Choosing learning rate.

We first tested with a different numbers of hidden layers using ReLU activations. We observed that the 2 hidden layer model produced the highest accuracy of 87.26%, as shown in Table 1. However, the difference of accuracy

between the models decreased when increasing the number of hidden layers. When switching from 0 hidden layers to 1 hidden layer, we observed a 2.69% increase in accuracy compared to a 0.76% increase when moving from 1 hidden layer to 2 hidden layers. In addition, we decided to test which learning rate had the highest accuracy. In this experiment (Table. 2), each model had 2 hidden layers with ReLU activations. The highest accuracy was found when using a 0.01 learning rate, and this was used for the remainder of our experiments.

Testing activation function (σ)	
Activation function (σ)	Accuracy
ReLU	0.8365
Leaky-ReLU	0.8131
tanh	0.8346
Sigmoid	0.7926

Table 3: Activation function

Testing regularization	
Regularization	Accuracy
L2 Regularization ($\lambda = 2$)	0.8434
Dropout (0.2)	0.7589

Table 4: Regularization

Subsequently, we tested out different activation functions on a 2 hidden layer MLP to find the best suited one by comparing model accuracies (see Table 3). We found that ReLU and tanh activations had similar performances (83%), in contrast to sigmoid which performed the worst (79%). We then explored the effect of regularization on the same networks by testing two methods: $L2$ regularization $\lambda||\mathbf{W}||_2^2$ and dropout layers. As shown in Table 4, we observed that the model performs better using $L2$ with $\lambda = 2$ in contrast to dropping out 0.2 units of each layer.

To understand the effect of using unnormalized data, we used a 2 hidden layer MLP with ReLU activations. We had to modify certain parameters provided to the linear layer and the mini batch training function in order to prevent overflow. While this showed a limitation of not normalizing the data, Fig.3 shows interesting characteristics of the model as well. Overall, the test accuracy was 84.43%, 2.83% lower than when we normalized the data. The graph not only shows volatile test and training accuracies but also some outliers in the cross entropy loss. These characteristics did not occur to this extent when normalizing the data.

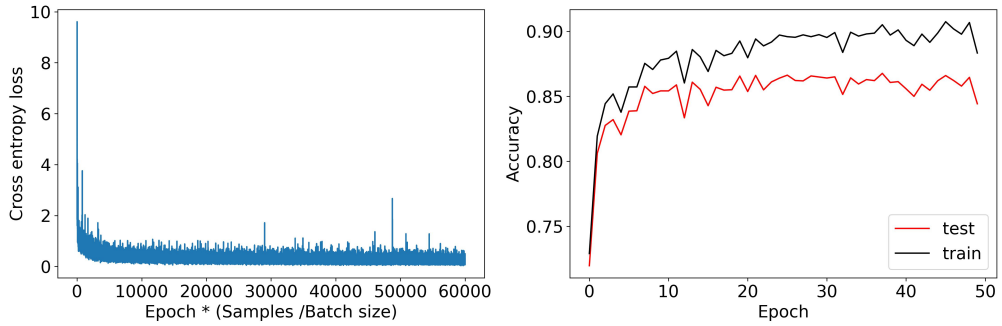
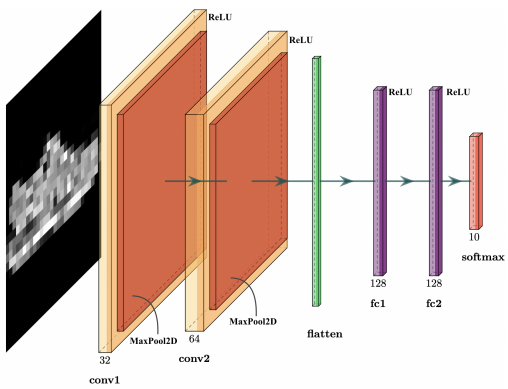


Figure 3: Unormalized Cross Entropy and Accuracy Plot

ConvNets constitute most of the state-of-the-art methods for image classification [11, 12]. To create a ConvNet, we used Google's platform, TensorFlow [13], an API designed for implementing and executing machine learning algorithms. Here, we used an architecture consisting of 2 convolutional layers followed by two fully-connected layers as shown in Fig.4.a. The first and second convolutional layers use 32 and 64 filters, respectively, both with a 3x3 kernel size. After each convolution, we applied a max-pooling operation. Then, the second convolutional layer was flattened for the fully connected layers, each of which had 128 units each. The architecture ended with a softmax activation function. We then used this to train our model for 30 epochs in a GPU. Appendix Fig.11 shows the training and test accuracies as a function of the number of epochs and Fig.4.b the classification metrics for each class/label using `classification_report` from scikit-learn [14]. Overall, this model was able to achieve 91% classification accuracy.



(a) Convolutional Neural Network diagram.

	precision	recall	f1-score	support
Class 0 (T-shirt/top) :	0.87	0.86	0.86	1000
Class 1 (Trouser) :	0.99	0.98	0.98	1000
Class 2 (Pullover) :	0.88	0.82	0.85	1000
Class 3 (Dress) :	0.91	0.91	0.91	1000
Class 4 (Coat) :	0.85	0.89	0.87	1000
Class 5 (Sandal) :	0.98	0.98	0.98	1000
Class 6 (Shirt) :	0.73	0.76	0.74	1000
Class 7 (Sneaker) :	0.96	0.95	0.96	1000
Class 8 (Bag) :	0.97	0.98	0.97	1000
Class 9 (Ankle boot) :	0.95	0.97	0.96	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

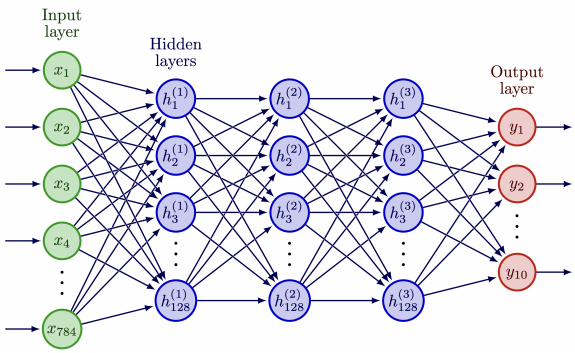
(b) Metrics.

Figure 4: Performance of a ConvNet with 2 convolutional and 2 fully conected layers.

In order to find the best architecture for the MLP, we conducted several experiments where we increased the number of hidden layers (h) and/or the number of hidden units (h_u). The outcomes of these experiments are presented in Table. 5. We found similar results for the first three models - increasing the number of hidden layers to 3 and using more hidden units (i.e., 300) did not increase the performance of the model. However, increasing the number of hidden layers to 4 affected its performance. Therefore, we selected model 1 (3 hidden layers with 128 hidden units each) as our best MLP model (see Fig.5.a) using runtime and representation (non-linearity) as determinants. We then visualized the model metrics using `classification_report` to see the model performance in each class (Fig.5.b). Additionally, we plotted the learning curve of this model (see Appendix Fig.13) by training the model using $60,000k, k \in \{0.1, 0.2, ..., 1\}$ of the data to analyze how the performance increases when training with larger datasets.

Testing MLP architectures			
Model	Hidden layers (h)	Units (h_u)	Accuracy
1	3	[128,128,128]	0.87
2	2	[300,100]	0.87
3	3	[300,128,100]	0.87
4	4	[128,128,128,128]	0.86

Table 5: Testing different configurations of MLP architectures.



(a) MLP with 3 hidden layers diagram.

	precision	recall	f1-score	support
Class 0 (T-shirt/top) :	0.78	0.85	0.81	1000
Class 1 (Trouser) :	0.99	0.96	0.98	1000
Class 2 (Pullover) :	0.80	0.74	0.77	1000
Class 3 (Dress) :	0.84	0.90	0.87	1000
Class 4 (Coat) :	0.77	0.83	0.80	1000
Class 5 (Sandal) :	0.97	0.93	0.95	1000
Class 6 (Shirt) :	0.71	0.61	0.66	1000
Class 7 (Sneaker) :	0.92	0.95	0.93	1000
Class 8 (Bag) :	0.96	0.96	0.96	1000
Class 9 (Ankle boot) :	0.94	0.96	0.95	1000
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000

(b) Metrics.

Figure 5: Performance of a 3 layer deep neural network.

To test our model, we applied small perturbations to a random weight in the neural network. We used epsilons

of 0.0001 and 0.0002 to see if they would have an effect on the gradient. We observed very little change when we tested this. For example, we received a gradient of -0.0022634007643129728 and -0.002263400799423776, respectively. We only noticed a difference starting from the 12th significant number. Therefore, the small perturbation only slightly affects the gradient calculation of the model.

4 Discussion and Conclusion

As it relates to MLPs, we can see that increased network depth creates more non-linearity, increasing training accuracy as the model can fit the data better (as expected). However, if we look at the accuracy graphs in the Appendix Fig.6, we can observe that the increased network depth comes with a cost of over-fitting the data as bias increases. Therefore, an overall increase of test accuracy was less expected.

We observed that the MLPs trained with ReLU and Leaky-ReLU activations behaved very similarly, the first outperforming the rest in terms of test accuracy. However, the MLP trained with tanh activations behaved differently - cross-entropy loss was not as fast to converge and accuracy was about 5% lower (see Appendix Fig.8). Similarly, we can see the same behavior when using sigmoid activations, as shown in Appendix Fig.10. This is inline with what was presented by [8] as they observed that ReLU activations generally outperform sigmoid activations. Furthermore, we observed the $L2$ regularization did not make a big difference for the model performance, however, it converged slightly faster with the regularizing term (Appendix Fig.9). We also found that in this approach, regularizing the model through dropout prompted instability and affected its performance, significantly dropping the test accuracy about 10% .

It should be noted that using a ConvNet model could be used to increase our accuracies for these experiments. As mentioned by [5], ConvNets can be more accurate than other traditional machine learning models and simple MLP architectures like the ones we test in this study. This is proved in our experiments where we used a basic architecture with 2 convolutional and 2 fully connected layers. We achieved a test accuracy of 91%, 4% higher than our best MLP model (with 3 hidden layers). ConvNets are better than MLP models because they can acquire effective depictions of the original image, allowing them to recognize visual patterns directly [12] instead of the vectorized representation that MLPs offers. For this main reason, it is difficult for a MLP to outperform a ConvNet.

However, when we analyze the learning curve of our best MLP (Fig.5.a) we can argue that 60k samples are not enough to saturate the network, as it continues with an upward trend. Hence, it could be possible to come closer to the ConvNet performance if we had more training data available. Nevertheless, we can find similarities when comparing how both models perform with individual classes. Specifically, we see that both models have the same F1-score (0.98) for the class “Trouser”, and both models struggle to classify the class “Shirt” (0.78 for the ConvNet and 0.66 for the 3hl-MLP).

In conclusion, through our findings, we discovered that network depth in a MLP is correlated with higher accuracy, and our MLP performed the best using ReLU and Leaky-ReLU activation functions. However, we found that out Convnet performed even better than our best MLP. For our MLPs to reach a Convnet’s accuracy, we would need to have more training data available.

5 Statement of Contributions

The team decided to split up the various tasks. In the end, a mixture of all was used.

References

[1] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta Numerica* 8 (1999), pp. 143–195. DOI: 10.1017/S0962492900002919.

[2] Ioannis Kanellopoulos and Graeme G Wilkinson. “Strategies and best practice for neural network image classification”. In: *International Journal of Remote Sensing* 18.4 (1997), pp. 711–725.

[3] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[5] VALDERI LEITHARDT. “Classifying garments from fashion-MNIST dataset through CNNs”. In: *Advances in Science, Technology and Engineering Systems Journal* 6.1 (2021), pp. 989–994.

[6] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed. “Classification of garments from fashion MNIST dataset using CNN LeNet-5 architecture”. In: *2020 international conference on innovative trends in communication and computer engineering (ITCE)*. IEEE. 2020, pp. 238–243.

[7] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[8] Michael McKenna. “A comparison of activation functions for deep learning on Fashion-MNIST”. In: *arXiv preprint arXiv:1708.07747* (2017).

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[10] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.698* (2014).

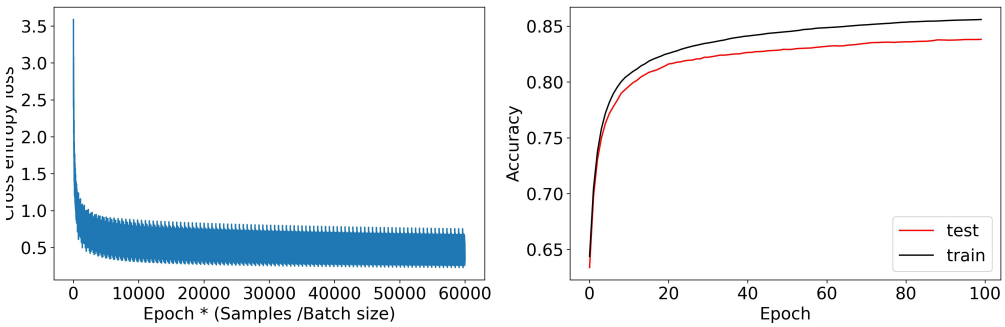
[11] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.

[12] Jiuxiang Gu et al. “Recent advances in convolutional neural networks”. In: *Pattern recognition* 77 (2018), pp. 354–377.

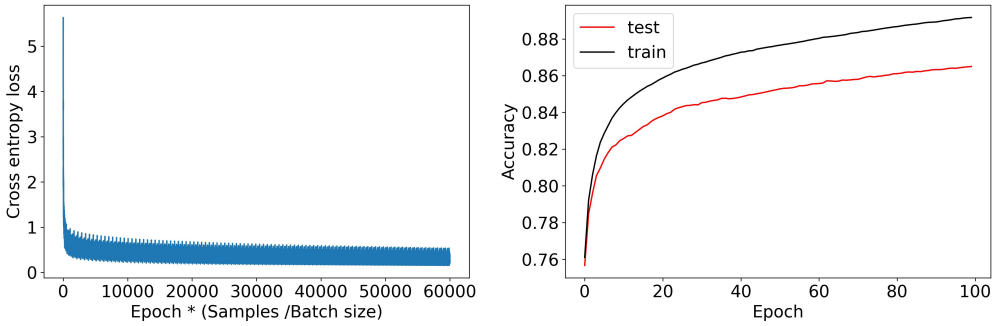
[13] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[14] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

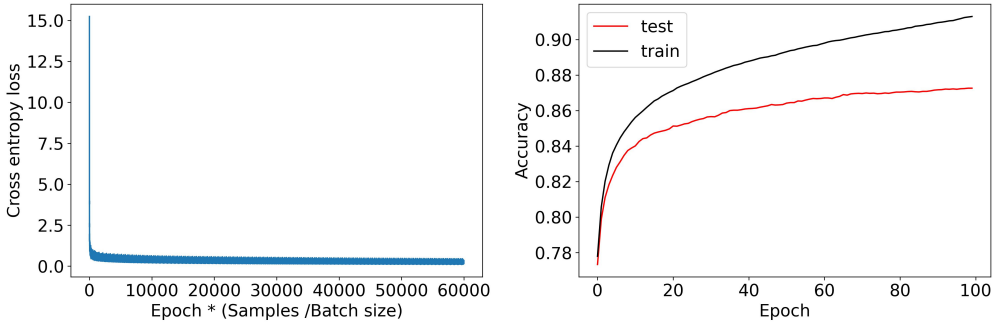
Appendix



(a) Loss and Accuracy per Epoch Using ReLU Activation and 0 hidden layers

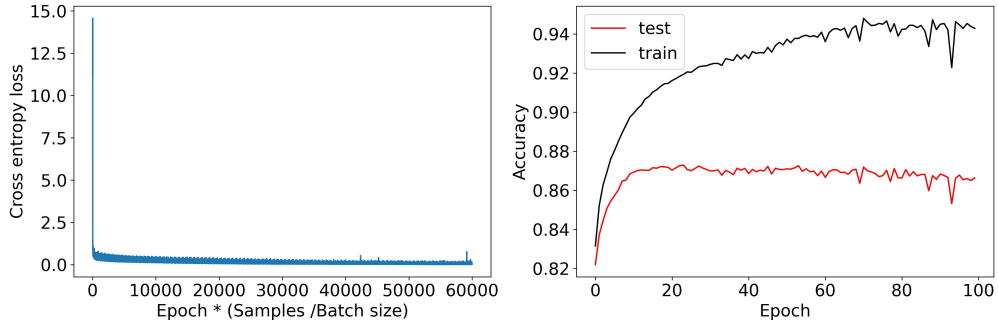


(b) Loss and Accuracy per Epoch Using Relu Activation and 1 hidden layer

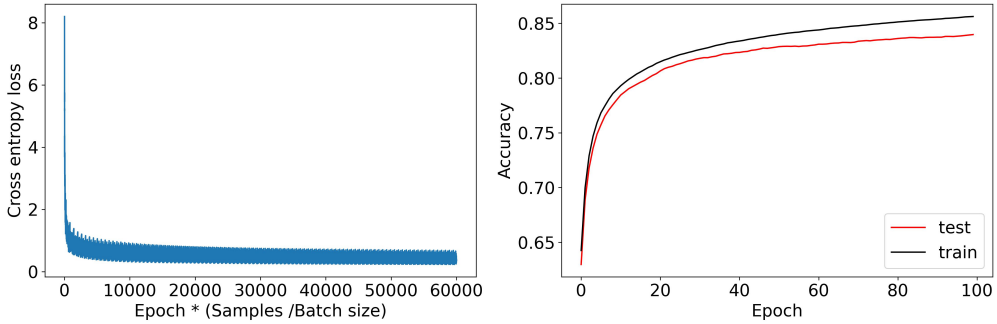


(c) Loss and Accuracy per Epoch Using Relu Activation and 2 hidden layers

Figure 6: Experiment 1 Results Using Relu Activation with 0, 1 and 2 hidden layers

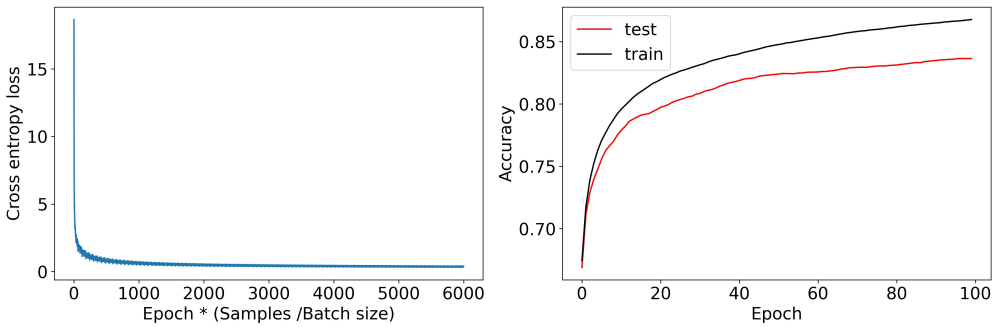


(a) Loss and Accuracy per Epoch Using ReLU Activation and 0.1 learning rate

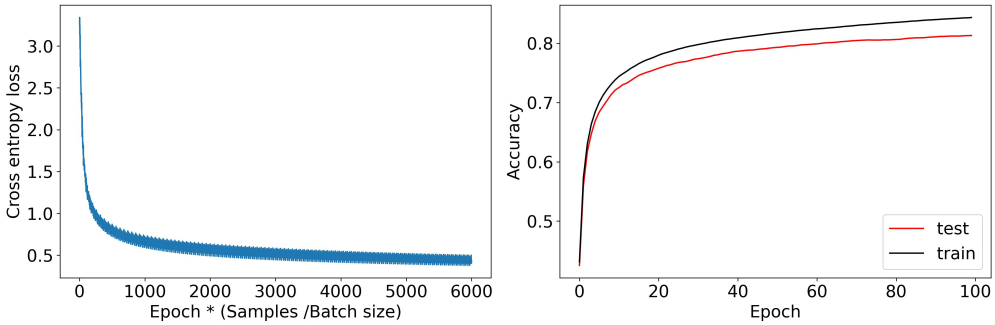


(b) Loss and Accuracy per Epoch Using Relu Activation and 0.001 learning rate

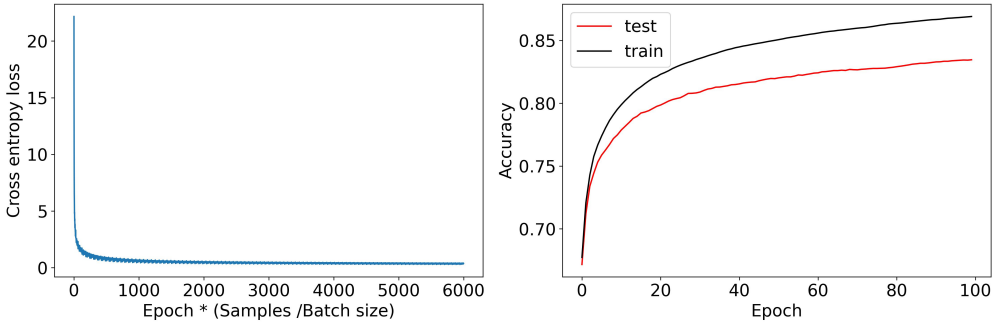
Figure 7: 2 alternative learning rate tests with 2 hidden layers



(a) Loss and Accuracy per Epoch Using ReLU Activations



(b) Loss and Accuracy per Epoch Using Tanh Activations



(c) Loss and Accuracy per Epoch Using Leaky-ReLU Activations

Figure 8: Experiment 2 Results Using Relu, Tanh, and Leaky-ReLU Activations

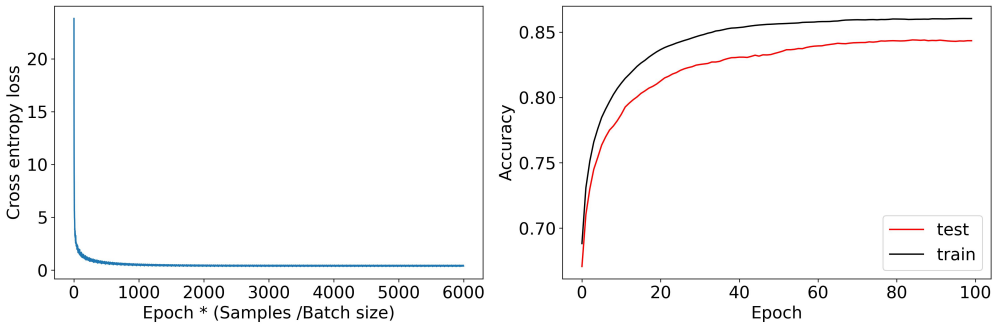
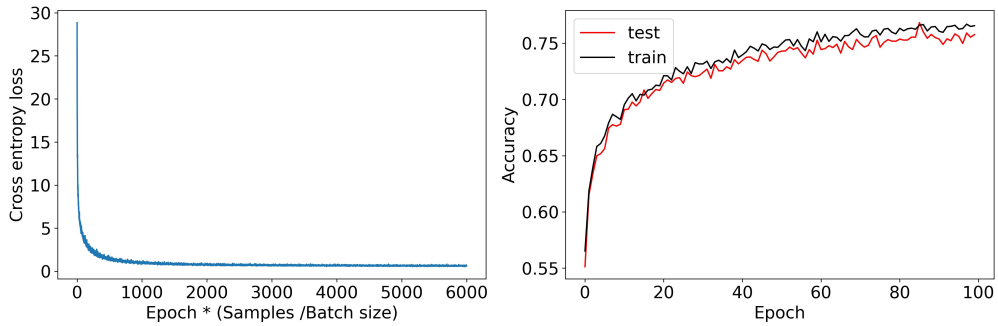
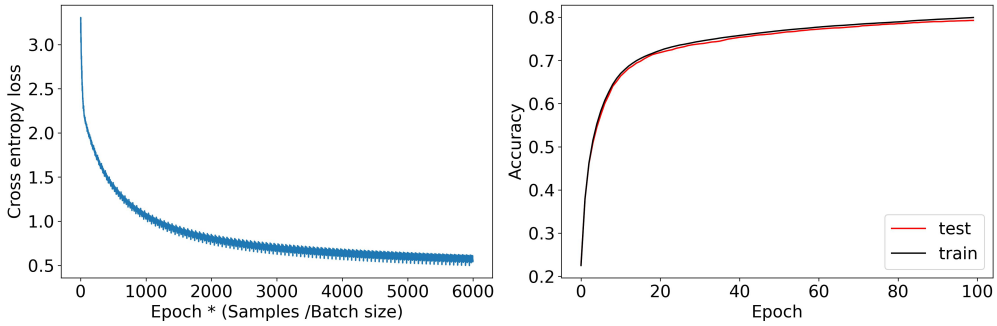


Figure 9: Experiment 3 Results Using L2 Regularization on Leaky-ReLU MLP



(a) Loss and Accuracy per Epoch Using Dropout



(b) Loss and Accuracy per Epoch Using Sigmoid Activations

Figure 10: Additional Experiments Conducted with Dropout and Sigmoid Activations

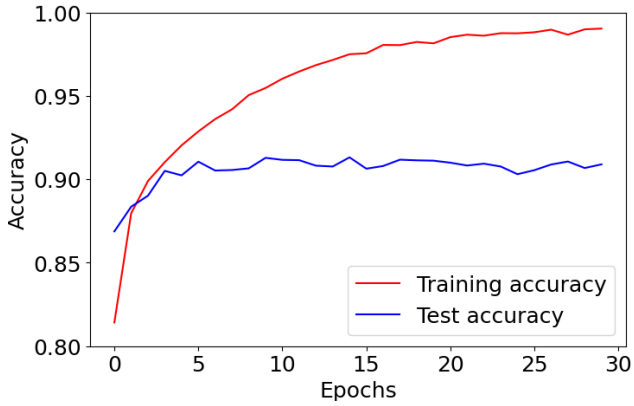


Figure 11: Train/test accuracy of ConvNet (Task 5).

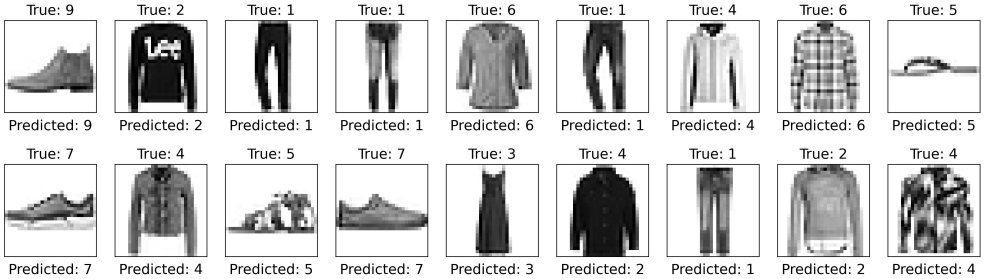


Figure 12: Prediction examples of ConvNet (Task 5).

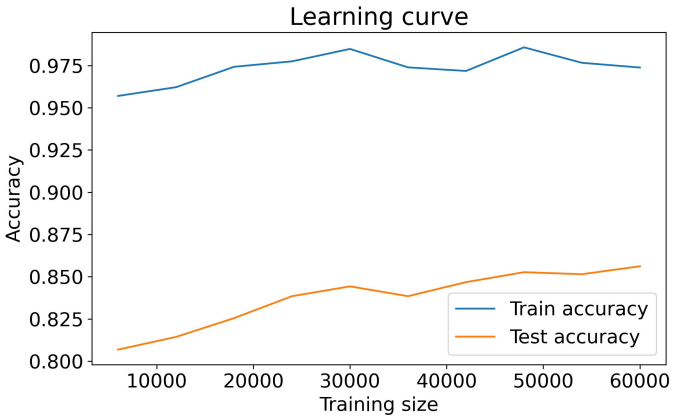


Figure 13: Learning curve for MLP with 3 hidden layers (Task 6).