

Comparing the Performance of CNN and MLP in Image Classification

Abhishek Damle

1. Abstract

This work develops an optimal Multilayer Perceptron(MLP) and Convolutional Neural Network(CNN) through hyperparameter optimization and compares their performance in image classification. The Fashion-MNIST dataset was used for training and testing. Since accuracy is not a comprehensive measure of performance, the two classifiers were evaluated using a combination of several performance metrics, in addition to 10-fold cross validation. The work evaluated the relative performance of the two neural networks using the performance metrics of precision, recall, f1-score, Cohen's Kappa, Matthews Correlation Coefficient, and Cross-Entropy. This resulted in a thorough comparison wherein the per-class performance and overall performance of the two neural networks could be evaluated. Of the two neural networks, CNN consistently performed better than MLP across all metrics.

2. Background

Neural networks are a collection of interconnected nodes, analogous to biological neural networks. Biological neural networks, composed of functionally associated neurons that are interconnected by synapses, play a critical role in the learning process for many animals that have an evolved nervous system. It is through these networks that the organisms with cognitive abilities develop a learned behavior that helps them better adapt to their environment. Drawing inspiration from the brain, artificial neural networks enable computers to solve cognitive tasks at which only humans excel. Just like a human brain, machines can be trained to perform certain tasks by analyzing training datasets. A perceptron, which is analogous to a biological neuron, is the fundamental unit of neural networks. A model of a perceptron that computes an output based on the weighted sum of its inputs is shown in figure 1. The weights allow the inputs to have differing levels of influence over the perceptron's output and the mathematical function used to compute the output is known as an activation function. The Multilayer Perceptron (MLP) is an example of a neural network which consists of layers of fully connected perceptrons.

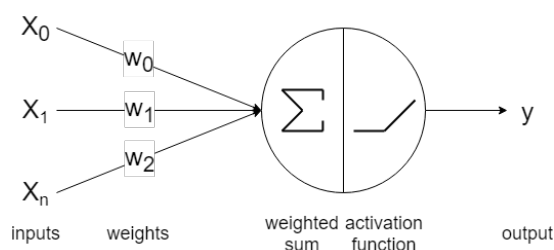


Figure 1: Model of a Perceptron

MLP's are formed by arranging perceptrons into layers as shown in figure 2. These neural networks are trained through a gradient descent process known as backpropagation where the error of the output of the network is minimized by adjusting the values of the weights.

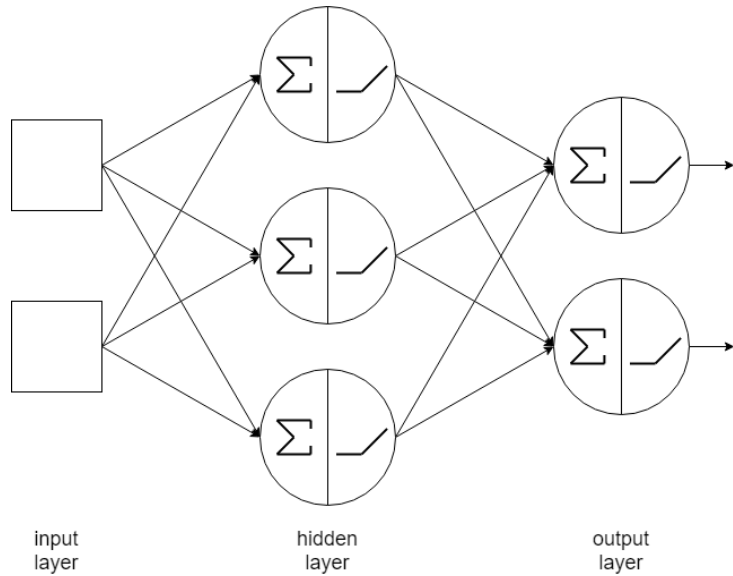


Figure 2: Example of a Multilayer Perceptron with a single hidden layer

With each layer further abstracting the input, MLP's are powerful because they can amplify aspects of the input that are important for classification while suppressing irrelevant information[6]. This property makes them particularly useful in image classification tasks where the complex task of identifying an object can be decomposed into simpler tasks such as identifying curves and edges. The key feature and advantage of MLP's is that this decomposition is not done manually but is rather learned automatically during training[6].

Convolutional neural networks (CNN's) evolved from MLP's and were found to be more flexible and easier to train[6]. Figure 3 shows an example of a general CNN.

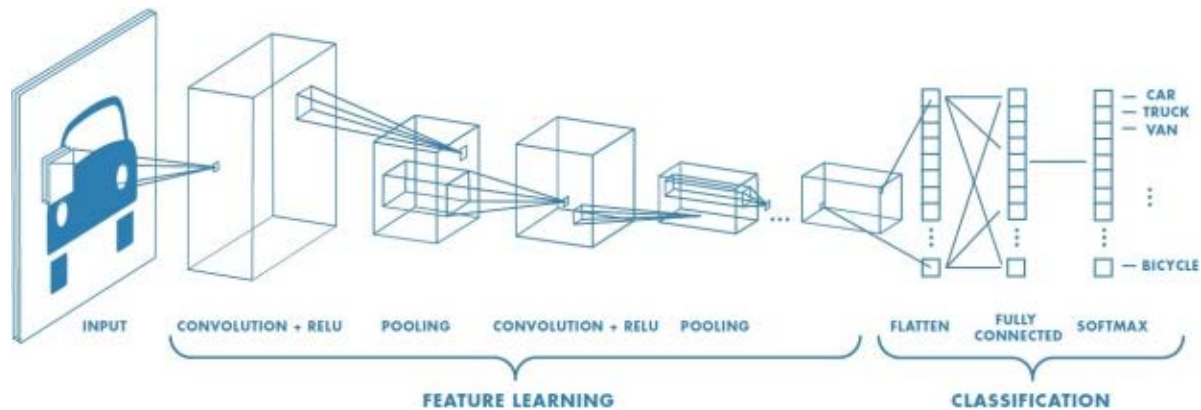


Figure 3: Generalized example of a Convolutional Neural Network[9]

The primary difference between CNN's and MLP's is that CNN's have an additional feature learning phase that consists of several alternating stacks of convolutional and pooling layers

before the fully connected neural network[6]. The convolutional layer convolves a filter matrix over the input while the pooling layer summarizes the input into a smaller output[7]. These layers are also important in other modern machine learning methods such as Fully Convolutional Networks(FCN) which fully omit the fully connected neural network and have found application in the task of semantic segmentation[8].

3. Related Works

This study was motivated by the need to rigorously cross examine the robustness of MLP and CNN as models of neural networks. While exploring an optimal dataset for comparing classifiers, a few studies were found that had evaluated the performance of these two neural networks. These studies tested the robustness of MLP and CNN models using classification accuracy. The Fashion-MNIST dataset Github page lists the classification accuracy of various classifiers in classifying its images [16]. Examples include the CNN designed by Dezhic [14] and MLP designed by Heitorrapela [15] which were able to achieve accuracies of .947 and .89, respectively. However, accuracy by itself is not an adequate measure of performance. Though it is very simple and intuitive, it is poor in assessing imbalanced data. Even for perfectly balanced classes, accuracy is over simplistic and does not capture the nuances of the performance of classifiers. Therefore, in order to perform a valid comparison of classifiers for the Fashion-MNIST dataset, multiple performance metrics must be used.

This work aims to rectify this shortcoming by using the metrics of precision, recall, f1-score, Cohen's Kappa, Matthews Correlation Coefficient, and Cross-Entropy to compare the performance of a MLP and CNN classifier. Furthermore, the various classifiers benchmarked on the Fashion-MNIST used a single training dataset and testing dataset. This work aims to further increase the validity of the performance metrics through performing 10-fold cross validation.






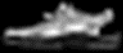




It is also unclear whether the classifiers listed on the Fashion-MNIST Github use the optimal hyperparameters. This work seeks to explore the optimal hyperparameters for CNN and MLP classifiers using hyperparameter optimization.

4. Methodology

A. Dataset

The Fashion-MNIST dataset was used for this project. It consists of 60,000 28x28 pixel grayscale images of clothing items that are assigned to a label from 10 different classes, as shown in Table 1. The Uniform Manifold Approximation and Projection (UMAP) visualization of the Fashion-MNIST dataset is shown in figure 4.

Table 1: Labels and descriptions of data from the Fashion-MNIST dataset

Label	Description	Example
0	T-shirt/top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandal	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boot	

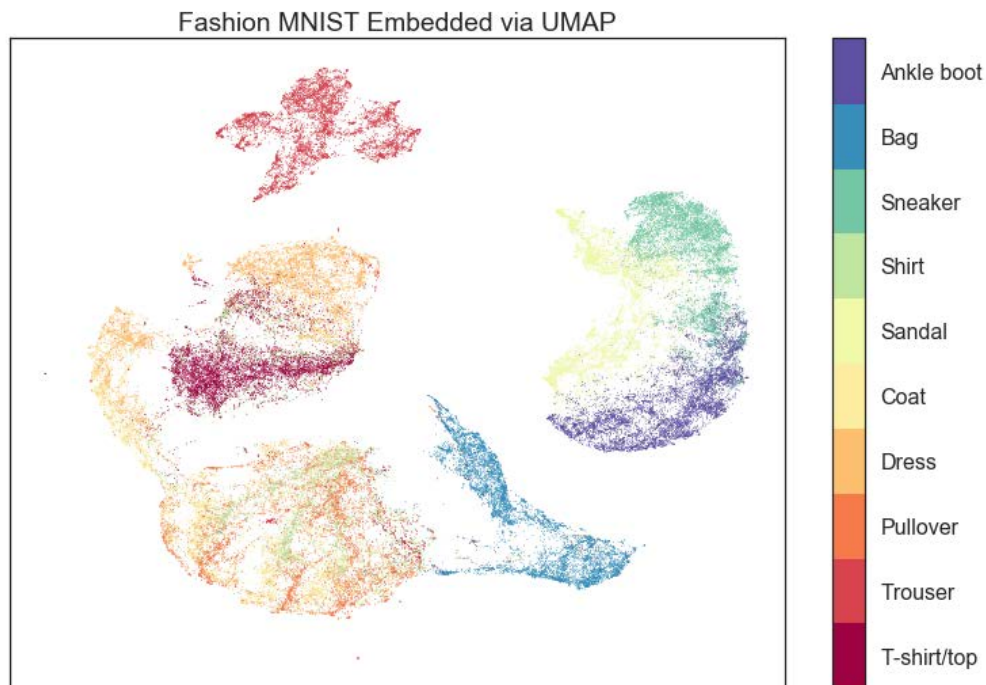


Figure 4: UMAP visualization of the Fashion-MNIST dataset[2]

The Fashion-MNIST dataset was developed as a replacement for the extremely popular MNIST dataset which had quite a few shortcomings. Primarily, the MNIST dataset is not complex enough since even simple machine learning algorithms can achieve an accuracy of 97% [1]. The Fashion-MNIST dataset was chosen with the rationale that its increased complexity would make the difference between the performance of MLP and CNN algorithms more distinguishable. Furthermore, Fashion-MNIST is already included in various machine learning libraries such as Keras, making it convenient to use for training.

B. Implementation of MLP and CNN Algorithms

The Keras Python library with the TensorFlow 2 backend was used to implement the CNN and MLP neural networks. Keras is a deep-learning API that allows the user to build and train neural networks[3]. Moreover, Keras is also highly configurable and enables the user with full control over hyperparameters. The combination of these features makes Keras ideal for implementing neural networks.

To allow for hyperparameter optimization, the CNN and MLP neural networks were built using functions that take hyperparameters as their input and output compiled Keras models. The MLP neural networks followed the architecture shown in figure 5 and consisted of three layers, namely the input layer, hidden layer, and the output layer. The input layer flattens the image to one dimension to allow it to be classified by the neural network. The variable number of hidden layers contain a variable number of neurons and use the rectified linear unit function(reLU) as their activation function. Lastly, the output layer contains ten neurons associated with each

clothing category and uses the softmax activation function to normalize the output to a probability distribution over predicted output classes.

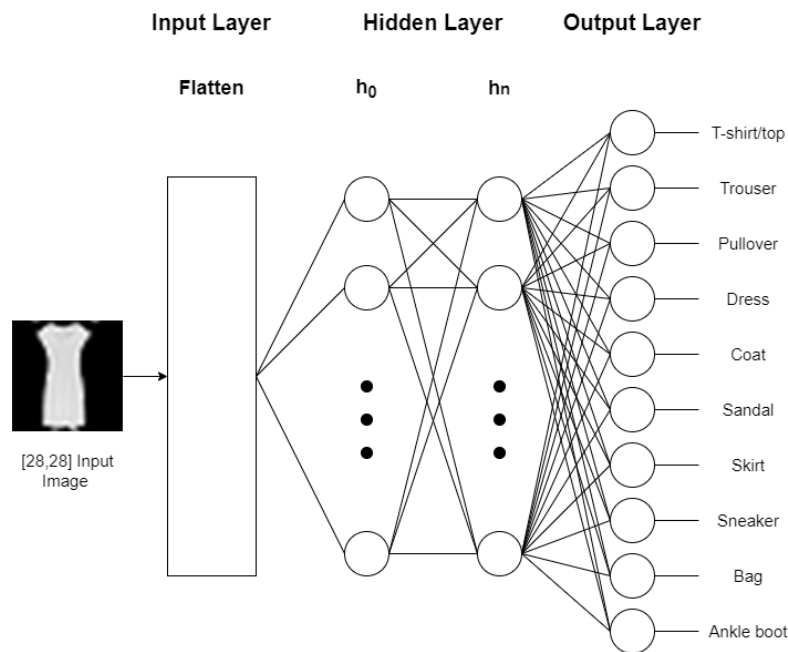


Figure 5: Architecture of MLP neural network

The architecture of the CNN is as shown in figure 6. Essentially, it contains the three layers of the MLP architecture from figure 5 preceded by convolution and pooling layers. Each convolution layer contains a variable number of filters that have a size of 3×3 , and use the ReLU activation function. Meanwhile, the pooling layers have a size of 2×2 . The number of alternating convolution and pooling layers in addition to the number of hidden layers, and the number of neurons in each hidden layer are variable.

Both the CNN and MLP neural networks used sparse categorical cross entropy for their loss function and used stochastic gradient descent for training. Furthermore, the learning rates for both neural networks were also variable.

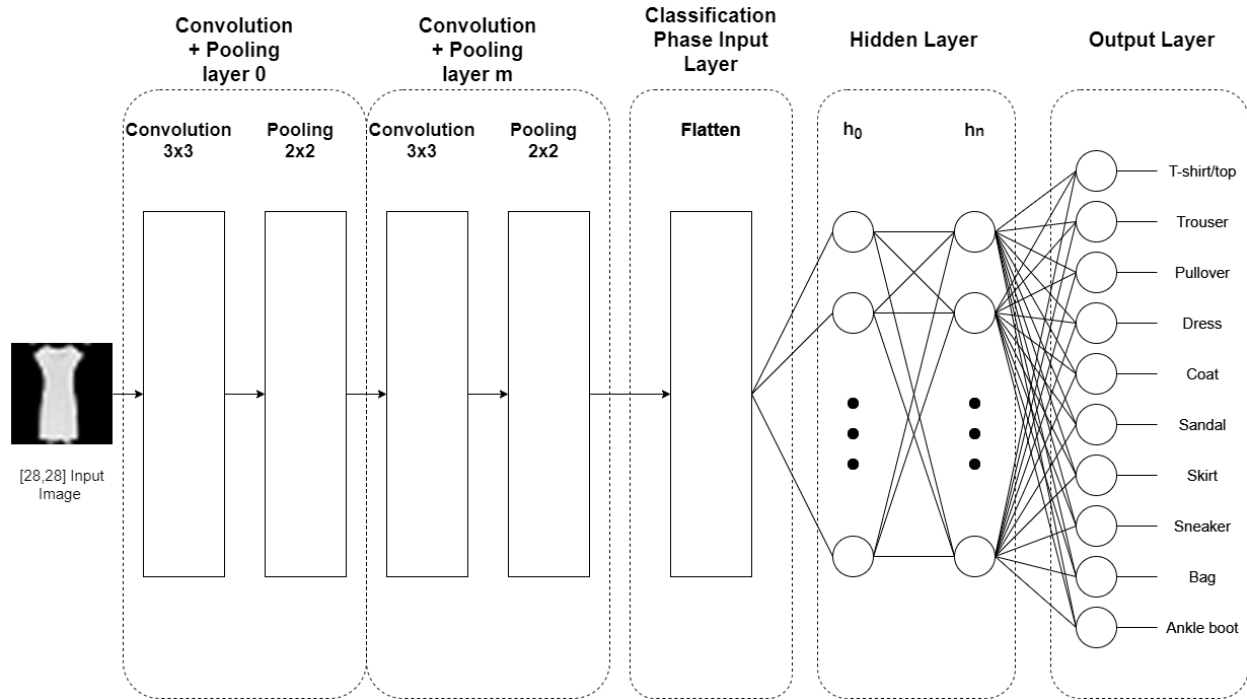


Figure 6: Architecture of MLP neural network

C. Hyperparameter Optimization

The CNN and MLP neural networks were implemented using Keras and their hyperparameters were optimized to ensure that the two machine learning algorithms could be fairly compared. Namely, the number of hidden layers, number of neurons, and learning rate hyperparameters were optimized for the MLP. On the other hand, the same hyperparameters in addition to the numbers of the convolution and pooling layers, and the number of filters in the convolution layers were optimized for the CNN. The sets of the optimized hyperparameters are listed in Table 2.

Table 2: Range of hyperparameters used for optimization

Hyperparameter	Set used for MLP	Set used for CNN
Number of hidden layers	[1,10]	[1,10]
Number of neurons in each hidden layer	[1,100]	[1,100]
Learning rate	{3e-2, 3e-3, 3e-4}	{3e-2, 3e-3, 3e-4}
Number of convolution and pooling layers	NA	[1,4]
Filters in convolution layer	NA	[32,256] with step size of 32

Hyperparameter optimization is still a nascent field of research and as a result, the best method to perform the optimization is unclear. Based on the literature review, Random Search was deemed suitable for this problem, as research has shown it to be more effective than Grid Search [4]. The Keras Tuner Library includes a random search function for hyperparameter optimization and was used for this project. A total of 100 random hyperparameters for both of the neural networks were tested using Keras Tuner and the metric of accuracy was used to select the best hyperparameters.

D. Performance Evaluation

After selecting the best hyperparameters, the CNN and MLP neural networks were trained and evaluated using 10-fold cross validation. The Scikit-learn Python library was used to evaluate the performance of each model. The performance metrics of precision, recall, f1-score, Cohen’s Kappa, Matthews Correlation Coefficient (MCC), and Cross-Entropy were used to compare the performance of both of the neural networks.

5. Results

The optimal hyperparameters of the CNN and MLP neural networks after 100 trials of random search are shown in Tables 3 and 4, respectively.

Table 3: Optimal hyperparameters for CNN

Hyperparameter	Optimal Value
Number of convolution and pooling layers	2
Filters in convolutional layer 1	224
Filters in convolutional layer 2	192
Number of hidden layers	2
Neurons in hidden layer 1	52
Neurons in hidden layer 2	11
Learning rate	0.03

Table 4: Optimal hyperparameters for MLP

Hyperparameter	Optimal Value
Number of hidden layers	6
Neurons in hidden layer 1	61
Neurons in hidden layer 2	30
Neurons in hidden layer 3	59
Neurons in hidden layer 4	19
Neurons in hidden layer 5	85
Neurons in hidden layer 6	66
Learning rate	0.0003

The classification reports for the CNN and MLP neural networks which consist of the precision, recall, f1-score, and support for each class are shown in tables 5 and 6, respectively. The values in Tables 5 and 6 represent an average of the values that were obtained from 10-fold cross validation.

Table 5: Classification report for CNN

Classification Report for CNN				
Class	precision	recall	f1-score	support
T-shirt/top	0.896	0.831	0.86	700
Trouser	0.993	0.987	0.989	700
Pullover	0.884	0.849	0.866	700
Dress	0.908	0.934	0.921	700
Coat	0.846	0.863	0.85	700
Sandal	0.988	0.983	0.987	700
Shirt	0.749	0.781	0.754	700
Sneaker	0.958	0.975	0.964	700
Bag	0.991	0.987	0.99	700
Ankle boot	0.977	0.964	0.97	700

Table 6: Classification report for MLP

Classification Report for MLP				
Class	precision	recall	f1-score	support
T-shirt/top	0.834	0.837	0.83	700
Trouser	0.991	0.977	0.982	700
Pullover	0.818	0.793	0.797	700
Dress	0.882	0.914	0.897	700
Coat	0.795	0.801	0.793	700
Sandal	0.948	0.972	0.955	700
Shirt	0.736	0.676	0.692	700
Sneaker	0.944	0.935	0.937	700
Bag	0.97	0.977	0.974	700
Ankle boot	0.967	0.925	0.94	700

The average accuracy, Cohen's Kappa, MCC, and cross entropy performance metrics for both of the neural networks are shown in table 6.

Table 6: Average accuracy, Cohen's Kappa, MCC, and cross entropy for MLP and CNN

	CNN	MLP
Accuracy	0.915	0.880
Cohen's Kappa	0.906	0.868
MCC	0.906	0.869
Cross-Entropy	0.227	0.324

The average normalized confusion matrices for the CNN and MLP are shown in figures 7, and 8, respectively.

Normalized CNN Confusion Matrix										
	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.83	0.00	0.01	0.03	0.00	0.00	0.12	0.00	0.00	0.00
Trouser	0.00	0.99	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
Pullover	0.01	0.00	0.85	0.01	0.07	0.00	0.07	0.00	0.00	0.00
Dress	0.01	0.00	0.00	0.93	0.03	0.00	0.02	0.00	0.00	0.00
Coat	0.00	0.00	0.05	0.03	0.86	0.00	0.06	0.00	0.00	0.00
Sandal	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.01	0.00	0.00
Shirt	0.08	0.00	0.05	0.02	0.07	0.00	0.78	0.00	0.00	0.00
Sneaker	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.97	0.00	0.02
Bag	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00
Ankle boot	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.96

Figure 7: Normalized confusion matrix for the CNN

Normalized MLP Confusion Matrix										
	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.84	0.00	0.01	0.03	0.00	0.00	0.11	0.00	0.01	0.00
Trouser	0.00	0.98	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
Pullover	0.01	0.00	0.80	0.01	0.11	0.00	0.07	0.00	0.00	0.00
Dress	0.02	0.01	0.01	0.91	0.03	0.00	0.02	0.00	0.00	0.00
Coat	0.00	0.00	0.09	0.04	0.80	0.00	0.06	0.00	0.00	0.00
Sandal	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.02	0.00	0.01
Shirt	0.13	0.00	0.08	0.03	0.07	0.00	0.68	0.00	0.01	0.00
Sneaker	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.94	0.00	0.03
Bag	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.98	0.00
Ankle boot	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.04	0.00	0.92

Figure 8: Normalized confusion matrix for the MLP

6. Discussion

Based on the classification report, both the methods had better precision, recall and f1-scores when it came to identifying items such as trouser, sandal, sneaker, bag, and ankle boot but had slightly low scores for items such as t-shirt/top, pullover, dress, coat, and shirt. This result is intuitive based on the visual similarity of the t-shirt/top, pullover, dress, coat, and shirt classes as exhibited by their proximity in the UMAP visualization from figure 4. Among the two neural networks, CNN consistently had higher precision, recall, and f1-score than the MLP. The metric of Precision is calculated, as shown by equation 1, as the ratio of true positives (TP) divided by the sum of true positives and false positives (FP). In other words, a higher precision value indicates the proportion of positive identifications that were correct. Since the CNN had higher precision values for its classes, it can be inferred that the CNN produced fewer false positives and more true positives than MLP.

$$Precision = TP / (TP + FP) \quad \text{Equation 1}$$

On the other hand, Recall, as given by equation 2, is a ratio of true positives divided by a sum of true positives and false negatives. It measures the proportion of the actual positives that were identified correctly. Therefore, the fact that the CNN has higher recall values than the MLP is indicative that the CNN had fewer false negatives and more true positives.

$$Recall = TP / (TP + FN)$$

Equation 2

Ideally, a model should have both high precision and high recall. The f1-score, as shown in equation 3, is the harmonic mean of precision and recall and substantiates that a model has both high precision and recall. Since the CNN had better f1-scores than the MLP, it can be concluded that its combined precision and recall scores are also better than the MLP.

$$F1\ Score = 2 * Precision * Recall / (Precision + Recall)$$

Equation 3

Based on the confusion matrices from figures 7 and 8, the trouser, sandal, sneaker, bag, and ankle boot classes were the most correctly identified classes while the t-shirt/top, pullover, dress, coat, and shirt classes were sometimes confused. This result is commensurate with the relative lower precision, recall, and f1-scores for these classes and reflects their visual similarity as exhibited by the UMAP visualization from figure 4.

In addition to the per class performance metrics, the CNN also had better Cohen's Kappa, Cross-Entropy, and MCC values. Cohen's Kappa is a metric that measures how closely the output of the classifier matches the validation set and corrects for how often the two might match by chance[10]. Therefore, the CNN's higher Cohen's Kappa score exhibits that it acts as a better classifier than MLP due to its output more closely matching the correct output.

Cross-Entropy is a measure of how closely a classifier's predicted probability matches the actual probability of the dataset[11]. Since a smaller cross-entropy value represents a smaller difference between the predicted and actual probabilities, the CNN's lower Cross-Entropy makes it a better classifier than MLP.

Matthew's correlation coefficient (MCC) is a metric that considers the true and false positive and negative rates to measure the correlation between a classifier's predicted classes and the actual classes[12]. Since the MCC of the CNN is closer to +1 than the MLP, the CNN is a better classifier.

Overall, it was found that the CNN performed better than the MLP based on the metrics of precision, recall, f1-score, Cohen's Kappa, Matthews Correlation Coefficient (MCC), and Cross-Entropy. In a broader context, the performance of this CNN was similar to the CNN developed by the Tensorflow documentation which was able to achieve an accuracy of .916 with two convolution and pooling layers in addition to a single fully connected layer[13]. While the CNN from the Tensorflow documentation [13] had similar architecture to the CNN developed in this project, other CNN's were able to achieve better results with different architectures. For example, Dezhic [14] was able to achieve a significantly greater accuracy of .947 by using shortcut connections. Meanwhile, other MLP's were also able to achieve better results using different architectures. For example, Heitorrapela [15] was able to achieve an accuracy of .89 using three hidden layers with dropout.

In the future, more architectures could be explored through hyperparameter optimization. However, such a task requires large amounts of computational power and time. Computational limitations had a significant impact on this current project and consequently only limited neural network architectures with a few hyperparameters could be considered. For example, there were

more than $3 \cdot 10^{20}$ possible combinations of hyperparameters for the simple MLP described in figure 5 and the 100 trials used for hyperparameter optimization were very unlikely to result in the most optimal results. Furthermore, even this limited search required multiple hours. This problem was further exacerbated for CNN which had a more complex design than the MLP and therefore had a much larger search space for hyperparameters. Increasing the hyperparameter search space further, to account for alternate architectures, would require significantly more trials for successful hyperparameter optimization. In order to solve this issue, hyperparameter optimization could be performed over the course of several days on a dedicated computer with a better hardware to allow for the evaluation of thousands of hyperparameters.

7. References

- [1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv*, Aug. 2017.
- [2] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform Manifold Approximation and Projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.
- [3] "Keras documentation," *Keras*. [Online]. Available: <https://keras.io/about/>. [Accessed: 18-Oct-2020].
- [4] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, Feb. 2012.
- [5] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. Sebastopol, CA: O'Reilly Media, Incorporated, 2019.
- [6] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [7] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
- [8] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965.
- [9] "Convolutional Neural Network," *MathWorks*. [Online]. Available: <https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html>. [Accessed: 18-Oct-2020].
- [10] K. Pykes, "Understanding Cohen's Kappa coefficient," *Towards Data Science*, 23-Nov-2020. [Online]. Available: <https://towardsdatascience.com/cohens-kappa-9786ceceab58>. [Accessed: 05-Dec-2020].
- [11] H. Goonewardana, "Evaluating Multi-Class Classifiers," *Apprentice Journal*, 14-Jan-2019. [Online]. Available: <https://medium.com/apprentice-journal/evaluating-multi-class-classifiers-12b2946e755b>. [Accessed: 05-Dec-2020].
- [12] "sklearn metrics Matthews correlation coefficient," *scikit learn*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html. [Accessed: 05-Dec-2020].
- [13] Tensorflow, "fashion-mnist CNN," *GitHub*, 25-Aug-2017. [Online]. Available: <https://github.com/zalandoresearch/fashion-mnist/blob/master/benchmark/convnet.py>. [Accessed: 05-Dec-2020].

- [14] Dezhic, “fashion-classifier,” *GitHub*, 2017. [Online]. Available: <https://github.com/Dezhic/fashion-classifier>. [Accessed: 05-Dec-2020].
- [15] Heitorrapela, “fashion-mnist mlp,” *GitHub*, 2017. [Online]. Available: <https://github.com/heitorrapela/fashion-mnist-mlp>. [Accessed: 05-Dec-2020].
- [16] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST,” *GitHub*, 28-Aug-2017. [Online]. Available: <https://github.com/zalando-research/fashion-mnist>. [Accessed: 09-Dec-2020].