IJHSR
ijhighschoolresearch.org
■ RESEARCH ARTICLE

Convolutional and Backpropagation Neural Networks On Image Classification

Jianqiao Song
St John's College, 27 Saint David Road, Johannesburg, Gauteng, 2198, South Africa; alexsongjianqiao@gmail.com

ABSTRACT: Nowadays, with the rapid development of Artificial Intelligence (AI) in our everyday lives, machine learning algorithms have been applied in a wide variety of fields, performing tasks that are unfeasible for conventional algorithms. Neural networks, inspired by the design of the biological nervous system, have become increasingly popular in fields that require recognizing relationships between vast amounts of data, such as facial recognition, stock market prediction, and signature verification. Different types of models each have their unique architectures. There are differences between the computational complexities of the models, influencing their performance of specific tasks. However, determining which architecture best suits a specific application, such as image classification, takes time and effort. This article compares the performance of two types of artificial neural networks when classifying images: the Convolutional Neural Network and the Fully Connected Neural Network. The results from experimentation lead to a better understanding of the two fundamental models and how their training and validation accuracies vary individually, both reaching a terminal point ultimately. It was clear that CNN had an advantage in terms of better accuracy but more time-consuming. It was also found that the number of convolutional layers does not necessarily improve the accuracies.

KEYWORDS: Computer Science; Artificial Intelligence; Machine Learning; Convolutional Neural Network; Fully Connected Neural Network; Image Classification.

■ Introduction

Background knowledge:

Machine Learning (ML) refers to the ability of a computer to learn automatically from past experiences or data, and its algorithms can construct mathematical models according to the training data provided, thus making predictions and decisions without being explicitly programmed. Neural networks are a subset of ML; they process information in a way inspired by the human brain and constantly improve with more training data, thus allowing them to perform comparably fewer specific tasks previously impossible for computer programs.

A biological neuron in the human brain receives input from other neurons, combines them, and performs a nonlinear operation to obtain the output. A node in a neural network performs similar operations. A neural network consists of three types of layers: the input layer, the hidden layer, and the output layer. The main operations within a neural network can be simplified into the following: The inputs are fed into the input layer, and specific weights and biases are assigned to the input layer. Then, a summation function is performed, which computes the product of each input and weight and adds all the products to calculate the weighted sum. After that, an activation function is performed, and the resulting value is passed to the next layer in the neural network. As compared to the linear functions with a non-restricted range, non-linear functions, such as sigmoid with a specified range from 0 to 1, are used to decide whether a neuron should be activated or not. Lastly, the errors are evaluated in the outputs by subtracting the desired output from the actual output. The errors are sent back to the hidden layer for adjusting the weights to lessen the error. This process is called Backpropagation and is iterated to achieve the desired result.[3]

Defining the problem

Neural Networks have become an indispensable tool in the realm of artificial intelligence, exhibiting remarkable capabilities in various domains ranging from computer vision to natural language processing. Among the numerous architectures available, Convolutional Neural Networks (CNN) and Fully Connected Neural Networks (FCNN) stand out as two common yet influential models, each with its own strengths and characteristics.[4-5] This research article aims to compare and analyze these two neural networks in the context of handwritten digit and character recognition, bringing out their similarities, differences, and performance in this fundamental dataset.

The utilization of both CNN and FCNN in a wide array of tasks and applications highlights their significance in the field of artificial intelligence. By comparing these two models, I aim to explore their respective advantages and limitations, providing valuable insights to researchers and programmers and enabling them to make informed decisions when selecting a neural network architecture for specific applications.

The Extended Modified National Institute of Standards and Technology database (EMNIST), a widely recognized dataset in the deep learning space, serves as an ideal criterion for our comparison.[6] It consists of images of handwritten digits and alphabetical letters. It enables the experimentation and testing of the two neural networks in tasks with significant practical implications, such as recognizing handwritten digits and characters. For instance, accurate recognition of handwritten digits

DOI: 10.36838/v5i6.2

is essential in postal services for efficient sorting and delivery. Similarly, the ability to convert handwritten text on tablets into digital text opens up possibilities for enhanced productivity and accessibility in work spaces – a simple example would be the marking of numerous exam papers written by students and scanned on a device. Moreover, utilizing the EMNIST dataset allows for easy comparison with existing research results since it is widely known and used by professionals in the field. Thus, this paper would provide a standardized evaluation framework, allowing other researchers to assess the effectiveness of their proposed techniques and algorithms by comparing their results with this paper and improving the theory. Additionally, the choice of EMNIST also offers the advantage of a relatively small dataset, which translates into faster training of the neural network models. Since less time is required for the training processes, a more comprehensive comparison can be achieved by allowing the article to explore a broader range of model configurations, hyperparameters, and optimization techniques within a reasonable time period.

Furthermore, recently, the importance of computer vision has been growing ever so rapidly in our society, as it holds vast potential for numerous applications across most of the working industries, even replacing human labor with AI in several jobs. The ability to accurately recognize and interpret visual data provides opportunities for advancements in autonomous vehicles, surveillance systems, medical imaging, and virtual reality, to name a few. By analyzing the performance of CNN and FCNN in the context of EMNIST, this paper contributes to the broader understanding of computer vision and its practical implications.

The main difference between Convolutional Neural Networks (CNNs) and Fully Connected Neural Networks (FCNNs) is the presence of a convolutional layer. The layer performs a dot product between two matrices, where one matrix is the set of learnable parameters called "kernel," and the other matrix is the restricted portion in the image. It takes the sum of the products of the corresponding numbers in each pixel of the two matrices. The kernel is spatially smaller than an image but is more in-depth. During the forward pass, the kernel slides across the height and width of the image, producing the image representation of that region. This produces a two-dimensional representation of the image known as an activation map that gives the kernel's response at each spatial position of the image. The sliding size of the kernel is called a stride. The pooling layer comes after the convolution layer, which replaces the network's output at specific locations by deriving a summary statistic of the nearby outputs. This helps reduce the representation's spatial size, which decreases the required computation and weights. The pooling operation is processed on every kernel individually. There are several pooling functions, such as the average pool and L2 norm, but the most popular process is max pooling, which reports the maximum output from the rectangular neighborhood.

Both neural network models contain a series of fully connected layers of nodes that link every node in one layer to every node in the next, and so on. CNN also contains several convolutional and pooling layers, thus considering each pixel's spatial properties when analyzing the image. Each model is coded using Python. The primary goals are to determine which of the two models is more accurate in classifying a specific set of images and why the model is suitable for the task. The CNN will likely be able to classify images with a higher accuracy than the FCNN. This article explains the structure and coding of the CNN and FCNN, the EMINST dataset used for training and testing, the comparison and analysis of results, and the reasons behind the phenomenon are discussed.

Literature Review

The idea of neural networks first came to be when the Cybenko paper was published in 1989.[7] The main statement of the paper was that a finite linear combination of compositions of a fixed function with a single variable and a set of affine functionals could uniformly approximate any continuous function of n real variables with support in the hypercube with n dimensions. And the combinations of functions discussed in the paper include a continuous sigmoidal function, which is one of the activation functions used in most neural networks today. The paper led to researchers conducting experiments on the performance of FCNN for approximating any function. However, the CNNs have only gained interest in recent years since 2010, when they outperformed the FCNN in classifying images. This article investigates the effect of the convolutional layer in the CNN on the accuracy when tested and whether the CNN is a better choice for image classification.

The investigation of the use of CNN for identifying images has always been a popular topic in recent years among experts in the field of deep learning. The first modern convolutional network is LeNet, created by Yann LeCun. It was trained using the MNIST dataset, which consists of images of handwritten digits paired with the labels of their real values.[8-9] Over time, CNNs have been improved so that they can be trained with larger, more complex datasets. The research around CNNs gained much more attention since Alex Krizhevsky introduced AlexNet in 2012.[10] The model's architecture was modified to achieve state-of-the-art performance, proven by labeling images in the ImageNet with an error rate of merely 15.3%.[11] Then, in 2014, Szegedy made further improvements by significantly decreasing the number of parameters involved and introducing the GoogLeNet.[12] It had an error rate of 6.67% when tested on ImageNet. In the same year, the VGGNet, with an error rate of 7.32% on the same test, was also published, focusing on increasing the depth of the network to improve performance.[13] The latest model until now is the ResNet, which was introduced in 2015.[14] It makes use of "skip connections" and batch normalization and was able to achieve the lowest error rate on the ImageNet test of only 5.71%.

By observing the evolution of these neural network architectures, I noticed a prominent trend toward increasing the models' depth by adding significant layers. This is evident in many notable models, such as ResNet, which achieved the top in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2015. ResNet has a depth of approximately 20 times greater than AlexNet and eight times

DOI: 10.36838/v5i6.2

greater than VGGNet. This trend towards deeper networks offers advantages such as adding more nonlinearity to approximate a desired function more accurately and improving feature representations. However, the increased depth has its downsides in terms of network complexity – more complex models make optimization more difficult and increase the risk of overfitting. Consequently, more recent researchers have proposed various methods to address these issues, coming up with diverse network design and optimization aspects. To avoid the issues mentioned caused by the high complexity of neural networks, this paper uses two rather fundamental model designs and a simple but well-known dataset for experimentation and comparison.

In 2010, Tiled CNNs were introduced, multiplying feature maps to learn rotational and scale-invariant features.15 It effectively eliminates the limitations of models unable to learn different kinds of invariances other than weight without diminishing the advantage of the weight-sharing mechanism, significantly decreasing the number of parameters. In 2015, Wang performed several experiments on the Tiled CNN and found that it performs better than traditional CNN on small time series datasets.[16] Furthermore, from 2014 to 2016, various researchers used Transposed Convolution, also known as Deconvolution, for better performances in the fields of visualization and recognition.[17,18] Unlike the traditional convolution that connects multiple input activations to a single activation, deconvolution connects a single activation with multiple output activations.

Despite the significant work investigating the performance of CNNs when classifying images, few papers are dedicated to researchers trying to implement FCNNs for image classification and analyzing the performance. One of the main aims of this article is first to investigate the performance of an FCNN when used to classify images and attempt to explain the observed results (for example, accuracy). The comparison of these results with that of a CNN with the same complexity is also made to show which one is more suitable for such a task. The advantage of such a model for image classification is explained as well. Thus, this article improves other researchers' understanding of the differences and similarities between the structure and performance of CNNs and FCNNs. It helps them make better decisions when using one of the two for a specific task.

Methods

The objective of the experiments is to determine the changes in the accuracy of classifying images when convolutional layers are added to the FCNNs. The experiments should be split into two separate sections: One for adjusting the architecture of the FCNN and another for adding convolutional layers to a fully connected layer to observe any changes in the accuracy. Initially, the CNN and FCNN must be programmed in Python using the appropriate imports from TensorFlow. It must be ensured that the complexities of each model are the same. Therefore, the number of fully connected layers should be equivalent in both.

The input image size will be 28 * 28 pixels since they are chosen from EMNIST. The class used is "letters" from the EMNIST dataset, which consists of handwritten uppercase letters. The training set contains 124,800 images, and the testing set contains 20,800 images. When training, for both models, the batch size is always set to be 128. For the FCNN, firstly, the number of epochs and the accuracy will be compared to show how the model learns through the training set. This model will have one hidden layer with 256 neurons. The total number of epochs is set to 30. The accuracy includes training accuracy and validation accuracy. The validation split is set to 0.1, meaning that 10% of the training data will be randomly selected as validation data, and the model will be tested on these data after each epoch to get the validation accuracy.

Then, the number of neurons in the layer will be adjusted from a range of 2 to 1024, while the number of layers is kept constant (1 hidden layer) to observe how the number of neurons affects both the training accuracy and the testing accuracy. The number of epochs in the training is set to 10 for all. Each hidden layer will use the activation function "tanh." The final output layer will always have 27 neurons, 26 alphabetical letters, and one blank sample, with the activation "SoftMax."

For the second section of the experiments, the convolutional (conv) layer will be added to the fully connected layer of 256 neurons and the output layer, with the same activations as used in the FCNN for both. The conv layer will have 32 kernels, all with the size of (3, 3), followed by a max pool with size (2,2). The epoch number will be set to 30 at first, the same as the initial FCNN, to compare the two regarding their training and validation accuracies. The accuracy will be determined by plotting both separately; therefore, the performance of both can be compared. Moreover, to add an additional layer of analysis regarding the comparison of the two models, their respective number of trainable and non-trainable parameters will be compared by constructing a table of the data and plotting a bar graph to show the difference in magnitude and relate it to the difference in accuracies. This is achieved using a "model.summary()" function in Python. A "time.perf_counter()" function is also added to time the total training time from the start of the first epoch to the end of the 30th epoch of each model for comparison.

Then, the number of conv layers will be increased to observe any changes in the model's performance. The respective trainable and non-trainable parameters of each of the varied models and their time taken for training will also be compared. Lastly, the number of kernels will be varied from 2 to 500 to see how that variable also affects the performance of the CNN; for all of the experiments, tables, and graphs will be created in Excel to represent and visualize the relationship between the independent/adjusted variable and accuracy.

Results and Discussion

Comparing different FCNNs:

For the experiments conducted on the FCNN, a model of 2 layers (1 hidden layer + 1 output layer) is constructed with 256 neurons in the hidden layer. The graph is used to help visualize the relationship between the three parameters and observe how they are changing with one another (Figure 1.1).

Table 1.1: Data collected with each of the 30 epochs and its corresponding training and validation accuracies as the model is being trained.

Epoch	Training Accuracy	Validation Accuracy
1	0.7275	0.8126
2	0.8444	0.8606
3	0.8785	0.8796
4	0.8972	0.886
5	0.9075	0.8954
6	0.9162	0.895
7	0.9224	0.8988
8	0.9276	0.8998
9	0.9326	0.8992
10	0.9365	0.9045
11	0.9416	0.9029
12	0.9432	0.9025
13	0.9468	0.905
14	0.9493	0.9034
15	0.9523	0.9031
16	0.9543	0.9032
17	0.9559	0.9021
18	0.9589	0.9014
19	0.9608	0.9035
20	0.9627	0.9029
21	0.964	0.9024
22	0.9653	0.8983
23	0.9668	0.8996
24	0.9675	0.9019
25	0.9691	0.9003
26	0.9705	0.8986
27	0.9712	0.898
28	0.9727	0.899
29	0.9731	0.8975
30	0.9745	0.8954

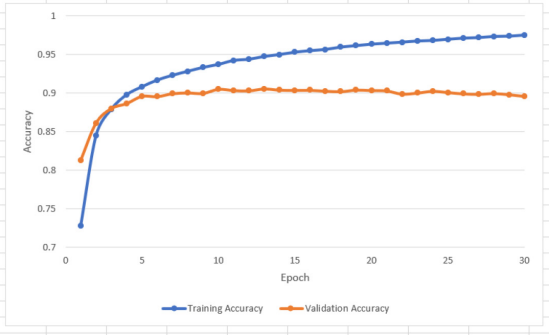

Figure 1.1: Graph plotted based on data points from table 1.1, with epoch on the x-axis and accuracy on the y-axis. Two different lines are used to represent training and validation accuracy.

An obvious observation that can be made from the data is that with the increasing number of epochs, the change in both the training and validation accuracies also decreases (Table 1.1). Since the gradient of both curves constantly decreases, the model is experiencing diminishing gains with an increasing number of epochs. This trend is shown most significantly for the training accuracy, which from epoch 1 to 2 increased by 12%, and from epoch 2 to 3 increased by only 3.5%. Moreover, the rates of accuracy increase from the last few epochs, namely from 20 to 30, are gradually approaching 0, which implies that there is a maximum training accuracy that can be approached with increasing number of epochs, and once it is reached, increasing the number of epochs would not have an effect on the accuracy anymore. This corresponds with the theory of the loss function and gradient descent. The max accuracy can be seen as a representation of the local minimum where the loss is at the lowest. It can be seen that the last few epochs have already reached the local minimum area, but the values are still randomly jumping left and right of the minimum. The first few epochs can be seen as initial jumps where the gradient is comparably a lot steeper, thus resulting in a significant increase in accuracy.

Furthermore, looking now at the validation accuracy, it is seen that the difference between validation and training accuracy (validation - training) changes from decreasing positive values to 0 then to negative values of increasing magnitude. In other words, the validation accuracy is initially higher than the training accuracy, but as the epoch number increases, it becomes comparably lower. The trend of the validation accuracy can be explained by considering how the model is being trained as time goes on: In the first few epochs, the model is learning the general characteristics of the data that are common to both training and validation datasets. Thus, earlier on, both performances are improving. Initially, validation accuracy is higher than training accuracy because the validation is performed after the model is fully trained with the set of data in the specific epoch. As the number of epochs increases to a certain amount (in this case, approximately 10), the model starts learning specific qualities in the training dataset only, losing its ability to generalize. Therefore, the validation accuracy stagnates, whereas the training accuracy continuously increases. This phenomenon is defined as "overfitting". To avoid overfitting, one can end the training earlier, train with more data, or use data augmentation.

Then, the number of neurons in the hidden layer is adjusted from 8 to 1024. The number of epochs is set to 10 for all models. The graph shows the effect of increasing the number of neurons on the final training and testing accuracy (Figure 2.1).

Table 2.1: Data collected for the different number of neurons and their corresponding training and testing accuracy. The training accuracy here refers to the training accuracy in the last epoch.

Neuron number	Training Accuracy	Testing Accuracy
1024	0.945	0.8964
768	0.9479	0.8993
640	0.9487	0.9032
576	0.9495	0.9045
512	0.9485	0.9033
448	0.9479	0.903
384	0.9463	0.9025
320	0.9431	0.9017
256	0.9396	0.8957
224	0.9361	0.8934
192	0.9322	0.8925
160	0.9252	0.8877
128	0.9177	0.8854
96	0.9034	0.8728
64	0.8832	0.8624
32	0.8293	0.8174
16	0.7553	0.7475
8	0.6171	0.619

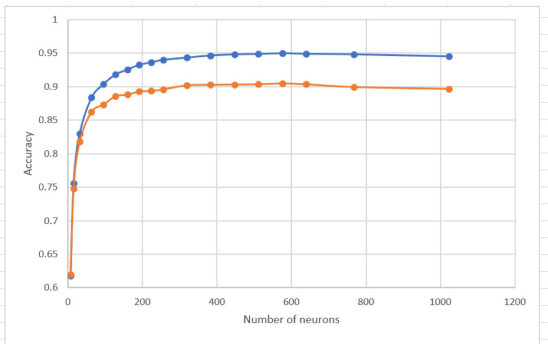

Figure 2.1: Graph plotted based on the data points from table 2.1, with the number of neurons on the x-axis and accuracy on the y-axis. Two different lines are used.

DOI: 10.36838/v5i6.2

Looking at the testing accuracy from the data collected, it can be reasonably concluded that increasing the number of neurons from 8 to 256 results in an obvious increase in the model's accuracy. The changes dropped significantly from 256 to 576 neurons, and the accuracy seems to stay at around 0.9, barely moving at all (Table 2.1). At this point, it appears that the limit of such a 2-layer FCNN has been found, which achieves approximately 0.90 accuracy on the test data. However, when the neuron number continues to go up to 1024, the accuracy begins to drop slightly to 0.89. This phenomenon occurs due to the number of neurons exceeding the data presented in the input, hence resulting in the data not being able to assign enough values to each neuron to make use of every neuron available.

Moreover, when the comparison between training accuracy and testing accuracy is being made, it is shown that the difference/drop between the training and testing accuracy increases with the increasing number of neurons. In fact, for the model with an 8-neuron layer, the testing accuracy is actually higher than the training accuracy.

Comparing FCNN and CNN:

The second part of the experiment is conducted with CNN. Firstly, a model of 1 convolutional layer consisting of 32 3 by 3 kernels with a stride of 1 is added to 2 fully connected layers, with 256 and 27 neurons, the same structure as the FCNN used for the initial experiment. The input of that layer is 5408 nodes since it comes from the flattened data of the results of conv layers.

Table 3.1: Data collected for the different number of epochs and their corresponding training accuracy for both FCNN and CNN as the model is being trained.

Epochs	Training Accuracy	
	FCNN	CNN
1	0.7275	0.8364
2	0.8444	0.9191
3	0.8785	0.9354
4	0.8972	0.9462
5	0.9075	0.9541
6	0.9162	0.9601
7	0.9224	0.9649
8	0.9276	0.9679
9	0.9326	0.9714
10	0.9365	0.9736
11	0.9416	0.9762
12	0.9432	0.9777
13	0.9468	0.98
14	0.9493	0.9811
15	0.9523	0.9807
16	0.9543	0.9826
17	0.9559	0.9831
18	0.9589	0.9844
19	0.9608	0.9853
20	0.9627	0.9857
21	0.964	0.9879
22	0.9653	0.987
23	0.9668	0.987
24	0.9675	0.9893
25	0.9691	0.9899
26	0.9705	0.9899
27	0.9712	0.9908
28	0.9727	0.9919

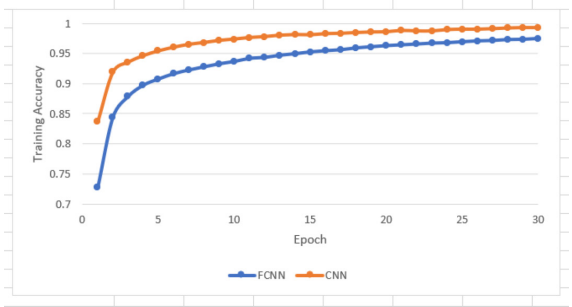

Figure 3.1: Graph plotted based on Table 3.1, Training Accuracy versus Epoch. Two lines are used to represent the FCNN data and the CNN data.

Table 3.2: Data collected for the different number of epochs and their corresponding validation accuracy for both FCNN and CNN after the model has been trained.

Epochs	Validation Accuracy	
	FCNN	CNN
1	0.8126	0.9014
2	0.8606	0.9179
3	0.8796	0.9232
4	0.886	0.9287
5	0.8954	0.9312
6	0.895	0.9252
7	0.8988	0.9324
8	0.8998	0.93
9	0.8992	0.9282
10	0.9045	0.9284
11	0.9029	0.9261
12	0.9025	0.9298
13	0.905	0.9245
14	0.9034	0.9258
15	0.9031	0.9261
16	0.9032	0.9288
17	0.9021	0.9265
18	0.9014	0.9236
19	0.9035	0.9256
20	0.9029	0.9295
21	0.9024	0.9244
22	0.8983	0.926
23	0.8996	0.9252
24	0.9019	0.9274
25	0.9003	0.925
26	0.8986	0.9248
27	0.898	0.9242
28	0.899	0.9288
29	0.8975	0.9266
30	0.8954	0.9252

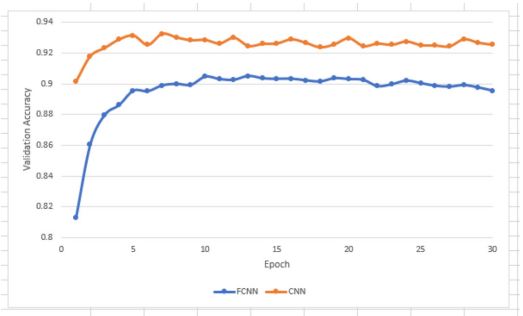

Figure 3.2: Graph plotted based on Table 3.2, Validation Accuracy versus Epoch. Two different lines are used.

Table 3.3: Data collected for the two types of neural networks and their corresponding trainable parameters.

	Trainable parameters
CNN	1391963
FCNN	207899

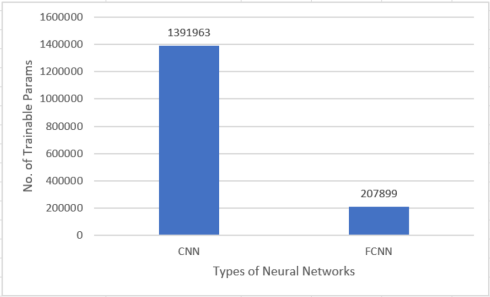

Figure 3.3: Bar graph plotted based on Table 3.3, Number of training parameters versus Types of Neural Networks.

Table 3.4: Data collected for the two types of neural networks and the corresponding time taken for each to train from epoch 1 to epoch 30.

	Time taken for training
CNN	124.0681
FCNN	94.7396

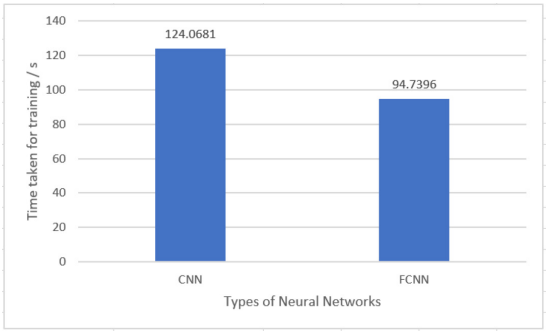

Figure 3.4: Bar graph plotted based on Table 3.4, Time taken for complete training (unit in seconds) versus Types of Neural Networks.

The data clearly indicates that the CNN has achieved better accuracy than the FCNN (Figure 3.1 & 3.2). Both the initial training and validation accuracy for CNN in epoch 1 is already at a much higher starting point than the FCNN (Table 3.1 & 3.2). However, this also resulted in a lower change/increase in accuracy between epochs 1 and 2 for the CNN. Similar to the FCNN, the CNN's training accuracy also seems to reach a maximum point of at the last few epochs, but that point seems to be 100%. This fact also reflects that the CNN can reach the optimum training accuracy faster than the FCNN. Both models seem to be moving up and down around a fixed level after epoch 10 for validation accuracy. It is also clear, from Figure 3.2, that the CNN can achieve a higher level of validation accuracy than the FCNN. All in all, the CNN outperforms the FCNN. Thus, the convolutional layer added improved the model's overall performance in classifying images.

Moreover, this supremacy of CNNs in terms of better accuracies can be verified and explained by comparing the trainable parameters between the two models (Table 3.3). The total trainable parameters for a given neural network between 2 layers is a sum of total weights and total biases that exist between them. Weights control the signal, or the strength of the connection, between two neurons. That is, a weight will decide how much influence the input will have on the output. Biases, which are constant, are an additional input into the next layer that will always have the value of 1. The more weights and biases a model has, the more accurate it'll be for a fixed dataset. In this case, since the number of non-trainable parameters is 0 for both models, it is not considered. The bar graph shows a clear difference in the magnitude of the number of trainable parameters between the CNN and FCNN (Figure 3.3). The CNN has almost seven times (6.7) as many trainable parameters as that of the FCNN. This corresponds with the theory and definition of trainable parameters since CNN has greater accuracy for both training and validation.

However, CNN also has its disadvantages compared to FCNNs. For instance, the data collected for the time taken for complete training of both models demonstrate one of the downsides of using CNNs for image classification (Table 3.4). As shown in the bar graph, FCNN takes noticeably less amount of time to train from epoch 1 to 30 compared to the CNN (Figure 3.4). It's around three-quarters of CNN's training time. This reflects the fact that CNNs may offer better accuracy, but it also takes longer to train with the same number of epochs.

Comparing different CNNs:

For the next part of the experiment with CNN, convolutional layers with the same structure, namely 32 (3,3) kernels with a stride of 1 and a max pool of (2,2), are continuously added to the FCNN of 256 + 27 nodes to observe the change on both training and validation accuracy.

Table 4.1: : Data collected for training accuracy of CNN models with 1, 2, and 3 conv layers with increasing epochs from 1 to 30.

Epochs	Training Accuracy		
	1 conv layer	2 conv layers	3 conv layers
1	0.8364	0.8405	0.6985
2	0.9191	0.9164	0.8339
3	0.9354	0.9309	0.8626
4	0.9462	0.9382	0.8785
5	0.9541	0.9446	0.8852
6	0.9601	0.9494	0.8936
7	0.9649	0.9541	0.8983
8	0.9679	0.958	0.9033
9	0.9714	0.9622	0.9074
10	0.9736	0.9644	0.9095
11	0.9762	0.9678	0.9116
12	0.9777	0.9702	0.914
13	0.98	0.972	0.9158
14	0.9811	0.974	0.9186
15	0.9807	0.9759	0.9198
16	0.9826	0.9764	0.9222
17	0.9831	0.9778	0.9228
18	0.9844	0.9789	0.924
19	0.9853	0.9795	0.9259
20	0.9857	0.9803	0.9266
21	0.9879	0.9811	0.9269
22	0.987	0.9834	0.9291
23	0.987	0.9817	0.9299
24	0.9893	0.9825	0.9309
25	0.9899	0.9836	0.9312
26	0.9899	0.9851	0.9338
27	0.9908	0.9841	0.9345
28	0.9919	0.9832	0.9346
29	0.9926	0.986	0.9356
30	0.9925	0.988	0.9368

DOI: 10.36838/v5i6.2

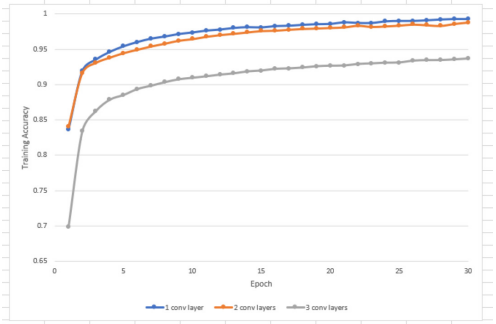

Figure 4.1: Graph plotted using the data in Table 4.1, Training Accuracy versus Epoch. Three different lines are used to represent the 3 CNN models.

Table 4.2: : Data collected for training accuracy of CNN models with 1, 2, and 3 conv layers with increasing epochs from 1 to 30.

Epochs	Validation Accuracy		
	1 conv layer	2 conv layers	3 conv layers
1	0.9014	0.9078	0.8096
2	0.9179	0.9177	0.8548
3	0.9232	0.9254	0.8674
4	0.9287	0.9324	0.8803
5	0.9312	0.9344	0.884
6	0.9252	0.9328	0.8889
7	0.9324	0.9338	0.8888
8	0.93	0.9348	0.8942
9	0.9282	0.9375	0.9013
10	0.9284	0.9362	0.8985
11	0.9261	0.9384	0.9014
12	0.9298	0.9344	0.9008
13	0.9245	0.9363	0.901
14	0.9258	0.9337	0.9018
15	0.9261	0.9369	0.9038
16	0.9288	0.9341	0.9044
17	0.9265	0.9334	0.9078
18	0.9236	0.9348	0.9062
19	0.9256	0.9341	0.9037
20	0.9295	0.9345	0.9093
21	0.9244	0.933	0.905
22	0.926	0.9318	0.9052
23	0.9252	0.9295	0.9069
24	0.9274	0.9324	0.9036
25	0.925	0.9325	0.9066
26	0.9248	0.9335	0.9085
27	0.9242	0.933	0.9066
28	0.9288	0.9323	0.9075
29	0.9266	0.9328	0.9115
30	0.9252	0.9337	0.905

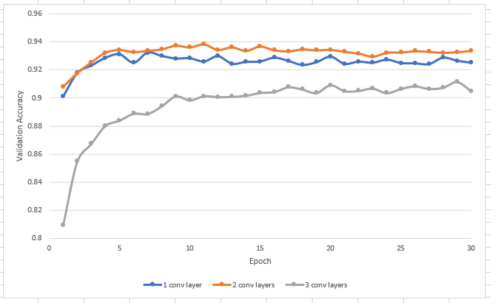

Figure 4.2: Graph plotted using the data in Table 4.1, Validation Accuracy versus Epoch, 3 different lines are used.

Starting with observations made with training accuracy, it can be seen that the training accuracies of the CNN with two conv layers are slightly lower than that of CNN with one conv layer from epoch 3 (Table 4.1). Whereas a noticeable decrease in training accuracy can be seen for the CNN with three layers from epoch 1 (Figure 4.1). However, it is also clear that, in the last ten epochs, the rate of increase of training accuracy (gradient of the graph) for both CNNs with 1 and 2 conv layers begins to decrease and settle down. While the same trend is not seen for the CNN with three conv layers – the rate of increase is consistent in the last ten epochs. This suggests that the CNN with three conv layers will eventually reach the maximum training accuracy achieved by the other two models with more training epochs. The data also corresponds with the theory that a CNN with more convolutional layers takes more time to train. Thus, more epochs are needed to achieve the optimal training accuracy.

Then, looking at the validation accuracy, the line plotted for the model with two conv layers is slightly higher than that of the model with one conv layer (Figure 4.2). This indicates that the CNN with two conv layers would outperform the CNN with 1 when both were tested on the testing dataset, even though the training accuracies are slightly lower for the former. A similar trend to the training accuracy is observed with the validation accuracy of the CNN, with three conv layers being lower than that of the other 2 (Figure 4.1). However, the difference is much lower than the training accuracy (Table 4.2). The model with three conv layers performs better because of the models' architectural design. Since a max pool is applied at the end of every layer instead of only one at the end of all conv layers, there might be too much data loss, thus lowering the performance. The number of kernels is also kept constant for all conv layers, which may be a factor preventing optimization as well, but further investigation is needed.

The last experiment is performed on the CNN with one conv layer of 32 (3,3) kernels, a stride of 1, and a max pool of (2,2). Here, the number of kernels is varied from 2 to 512 to observe the effect on both the training and testing accuracy.

Table 5.1: Data collected for both training and testing accuracy of CNN models with an increasing number of kernels. The training accuracy here refers to after all training epochs have been completed.

Number of kernels	Training Accuracy	Testing Accuracy
2	0.9391	0.9041
4	0.9472	0.9147
8	0.964	0.9186
16	0.9698	0.923
32	0.974	0.9235
64	0.9782	0.9258
96	0.9794	0.9312
128	0.9804	0.9248
192	0.9822	0.9262
256	0.9825	0.923
384	0.9845	0.9244
512	0.9843	0.927

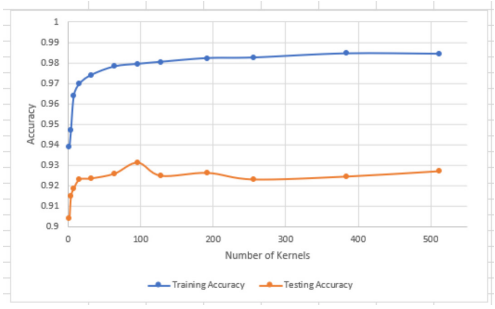

Figure 5.1: Graph plotted using the data in Table 4.1, Validation Accuracy versus Epoch, 3 different lines are used.

The trends observed from the data are similar to the results of the first experiment conducted with FCNN (Table 5.1).

The first few increases in kernels (from 2 to 16) result in a comparably large increase in both accuracies and as the number of kernels gets larger, the rate of increase gets closer to 0 (Figure 5.1). This phenomenon is because the images are merely black and white, their contents being handwritten letters, with 784 pixels in total. There needs to be more complexity in the image; in other words, the specific features or qualities that differentiate one classification from another are rather simple. Thus, the excess number of kernels is unnecessary and will be wasted after a certain number of kernels is reached. This can also be seen in the data when the training and testing accuracy stagnates after around 180 kernels.

Conclusion

In this paper, we first investigated the performance of both the CNN and the FCNN on image classification and compared the two to see which is better suited for the task. The first observation made is that in the training of both models, the rate of increase in accuracy decreases with the increase in number of epochs. The phenomenon of overfitting is also seen, where the validation accuracy lies below the training accuracy. The effect of varying the number of neurons in the hidden layer in the FCNN with two layers (hidden + output) is also seen, which shows that more neurons result in an increase in accuracy at the start until a maximum value of accuracy is reached, which indicates the data of the image is not enough. When comparing the performance of FCNN and CNN, the main conclusion can be made that CNN outperforms the FCNN in image classification since both the training and validation accuracy are higher in the former. This conclusion is also validified by the data collected for the number of trainable parameters related to weights and biases. It is concluded that the more weights and biases there are for a model, the more accurate it will be on a fixed dataset. Then, the disadvantage of the CNN is also discovered by timing the training from epoch 1 to 30. It was found that the FCNN takes around three-quarters of the time of CNN for training.

Moreover, the experiments performed on CNN prove that optimizing the model is complex. Adding layers of the same structure does not necessarily improve the model's performance. In the future, experiments can be done with the location and number of max pools as a parameter to compare against accuracy or using different numbers of kernels in more than one layer, switching between a decreasing or increasing number as the conv layers get closer to the fully connected ones. These future experiments can be done to investigate further the optimization of the CNN for image classification and aim to increase the validation accuracy and, thus, the testing accuracy.

Lastly, we have also observed that changing the number of kernels in a CNN with a single conv layer while keeping other parameters constant can have a similar effect on both the training and testing accuracies of the model compared to varying the number of neurons in the FCNN. The image is not complex enough for the excess number of kernels to be necessary. In the future, these experiments can be conducted using a more complex dataset, such as ImageNet, to observe clearer trends in the results and more significant differences between the performance of FCNN and CNN when classifying images.

Acknowledgments

This paper was written by the author Jianqiao Song with the help and tutoring of his mentor, Tim Adamson. Tim spent much time deepening my understanding of the fundamental concepts of neural networks and guiding me through designing the experiments, including advanced coding using Python.

References

1. Mitchell, T. M. Machine Learning; McGraw Hill: New York, 1997
2. Gurney Gevin. An Introduction to Neural Networks; Ucl Press, C op: London, 1997.
3. Werbos, P. J. Backpropagation through Time: What It Does and How to Do It. *Proceedings of the IEEE* **1990**, *78* (10), 1550–1560 . https://doi.org/10.1109/5.58337.
4. Nebauer, C. Evaluation of Convolutional Neural Networks for Vis ual Recognition. IEEE Transactions on *Neural Networks* **1998**, *9* (4), 685–696. https://doi.org/10.1109/72.701181.
5. Hecht-Nielsen, R. Theory of the Backpropagation Neural Networ k. *Neural Networks* **1988**, *1*, 445. https://doi.org/10.1016/ 0893-6 080(88)90469-8.
6. Cohen, G.; Afshar, S.; Tapson, J.; van Schaik, A. EMNIST: Exten ding MNIST to Handwritten Letters. 2017 *International Joint C onference on Neural Networks (IJCNN)* **2017**. https://doi. org/10. 1109/ijcnn.2017.7966217.
7. Cybenko, G. Approximation by Superpositions of a Sigmoidal Fu nction. *Mathematics of Control, Signals, and Systems* **1989**, *2* (4),303 –314. https://doi.org/10.1007/bf02551274.
8. LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L. D. Backpropagation Applied to Handwrit ten Zip Code Recognition. *Neural Computation* **1989**, *1* (4), 541-5 51. https://doi.org/10.1162/neco.1989.1.4.541.
9. Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Pro cessing Magazine* **2012**, *29* (6), 141–142. https://doi.org/10.1109/ msp.2012.2211477.
10. Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classifica tion with Deep Convolutional Neural Networks. *Communication s of the ACM* **2012**, *60* (6), 84–90. https://doi. org/10.1145/30653 86.
11. Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classificati on with Deep Convolutional Neural Networks. *Communications of the ACM* **2012**, *60* (6), 84–90. https://doi.org/10.1145/306538 6.
12. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D. ; Dumitru Erhan; Vanhoucke, V.; Rabinovich, A. Going Deeper w ith Convolutions. *arXiv (Cornell University)* **2014**. https://doi.org /10.48550/arxiv.1409.4842.
13. Simonyan, K. and Zisserman, A. Very Deep Convolutional Netw orks for Large-Scale Image Recognition. *arXiv (Cornell University) **2015**. https://doi.org/10.48550/arXiv.1409.1556.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for I mage Recognition. *arXiv (Cornell University)* **2015**. https://doi.or g/10.48550/arxiv.1512.03385.
15. Jiquan Ngiam; Chen, Z.; Chia, D.; Pang Wei Koh; Le, Q. V.; Ng, A. Y. Tiled Convolutional Neural Networks. **2010**, *23*, 1279–1287.
16. Wang, Zhiguang; Oates, Tim. Encoding Time Series as Images f or Visual Inspection and Classification Using Tiled Convolutional Neural Networks. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Workshops* **2015**.
17. Zeiler, M. D.; Fergus, R. Visualizing and Understanding Convolu

DOI: 10.36838/v5i6.2

tional Networks. *arXiv (Cornell University)* **2013**. https://doi.org/10.48550/arxiv.1311.2901.
18. Zhang, Y.; Lee, K.; Lee, H. Augmenting Supervised Neural Networks with Unsupervised Objectives for Large-Scale Image Classification. *arXiv (Cornell University)* **2016**. https://doi.org/10.48550/arxiv.1606.06582.

Author

Jianqiao is a high school student studying A levels at St John's College in South Africa. He has received several prominent certifications for the math and physics competitions he participated in, both his passions. He plans to study at a UK university and major in mathematics with physics.