Comparison of Convolutional Neural Networks

Rafael Gayahan
School of Computer Science
University of Texas at Dallas
Frisco, Texas
RMG180000@utdallas.edu

***Abstract*—Convolutional Neural Networks (CNNs) are deep learning network architectures that learn directly from data provided through data sets fed to the network. This paper will explore different convolutional neural networks and explore different hyper parameter tunings and see how it affects loss on the data set as well as the accuracy of the neural network. The networks being explored are a simple convolutional neural network provided to us, LeNet, and a modified version of LeNet. The data sets that will be used in this paper are MNIST, Fashion MNIST, and CIFAR10. Hyper parameter tunings will become much more important in the CIFAR10 data set as the data set includes 3 input channels as it has colored images while MNIST and Fashion MNIST are black and white images. The modified version of LeNet is similar in style but has two extra convolutional layers and one less linear layer.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Convolutional neural networks are deep learning network architectures that learn directly from data provided from data sets. These networks are modeled after the human body's biological nervous systems and specifically how the brain itself operates with information. For example, given pictures of different animals the human brain is able to make a connection and describe what kind of animal it is seeing. There are many variations of specific animals but the brain can figure them out due to the information the brain has obtained over time on what an animal generally looks like. Cats have many breeds and come in various shapes and sizes. These can include a Siamese, a British Shorthair, a Maine Coon, a Ragdoll, a Sphynx, and a Scottish Fold. Although the number of breeds far exceeds this list, the brain can easily recognize any of these breeds as a cat. Convolutional neural networks are used in this way as it is primarily used in recognizing patterns within images. These networks learn and self-optimize through receiving large quantities of data like the data sets that will be present further in this paper. Similar to how brains receive information, CNNs receive input data and split it into channels that it can understand. If it is a black and white image, there is only one input channel while coloured pictures generally have three channels due to it requiring the red, green, and blue colors that images are generally built from. These tasks are pretty easily solved by our brain and are able to understand pictures and label them. On the other hand, CNNs will be bound by how powerful the hardware of the computer is to make these computations on the pictures. Run time and accuracy become much more difficult as the load of information from a data set increases. This paper is motivated to explore different neural network architectures from the past and compare them using older and more modern data sets. LeNet-5 is an old neural network architecture that was raised back in 1998 [5]. LeNet-5 was originally designed with MNIST in mind so we will see a very good performance of LeNet-5 on the MNIST dataset. The main focus will be how a simple CNN architecture performs on the MNIST, Fashion-MNIST, and CIFAR10 data sets in comparison to LeNet-5 on the same data sets. After this comparison, we will continue to expand further and see how modifications to LeNet-5 can increase the accuracy of the network on the CIFAR10 dataset.

A. The MNIST Dataset

The MNIST data set is comprised of images of hand written numbers as shown below.

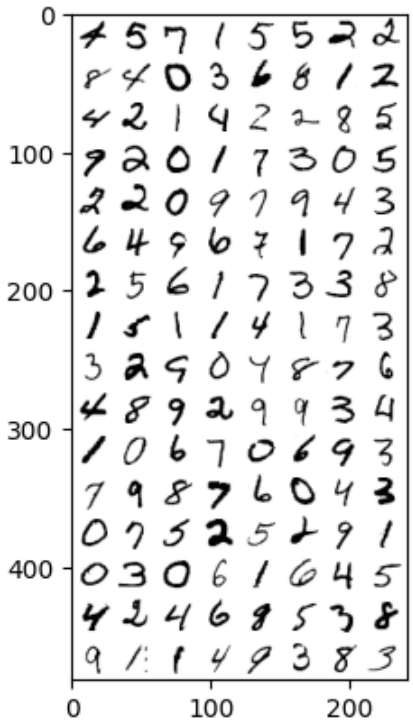

Fig. 1. MNIST images generated by code provided

The training set is 60,000 images at 28x28 resolution while the testing set is 10,000 images also at 28x28 resolution.

B. The Fashion-MNIST Dataset

The Fashion-MNIST data set is comprised of 28x28 images of different articles of clothing.

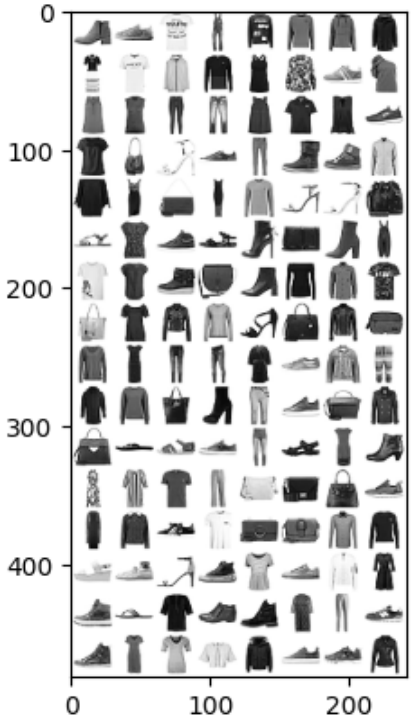

Fig. 2. Fashion-MNIST images generated by code provided

The training set is 60,000 images at 28x28 resolution while the testing set is 10,000 images also at 28x28 resolution.

C. The CIFAR-10 Dataset

The CIFAR-10 data set is comprised of 32x32 images of different animals and other objects like airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks

The training set is 60,000 images at 28x28 resolution while the testing set is 10,000 images also at 32x32 resolution with three channels as it has colored images. This dataset will be modified to use a grayscaled version as well as the original colored version.

II. Related Work

A. Comparing Previous Experiments to our experiment

Similar work has been done to improve the LeNet-5 network such as the work done by Yifan Wang, Fenghou Li, Hai Sun, Wenbo Li, Cheng Zhong, Xuelian Wu, Hailei Wang, and PingWang [3]. They also used the MNIST dataset to analyze how well the LeNet-5 network architecture performs [3]. In a traditional LeNet-5 architecture, it uses sigmoid activations while in the LeNet-5 architecture that we used

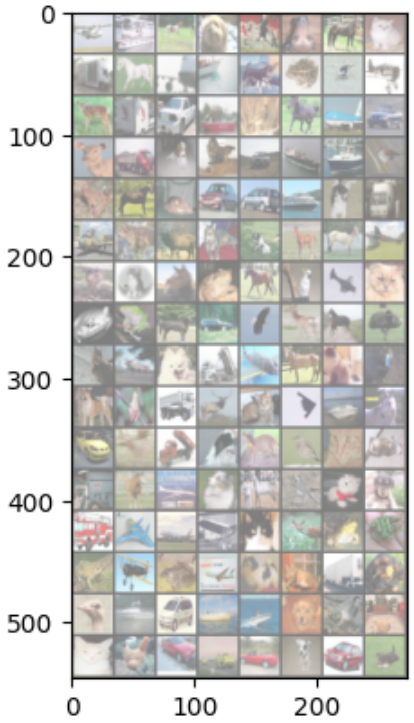

Fig. 3. CIFAR-10 images generated by code provided

for this paper was Tanh activations [3]. In general, Sigmoid activations have a smaller derivative as compared to Tanh. Sigmoid activations have inputs that are centered from (0, 1) while a Tanh function will be centered around 0 as its inputs are pushed from (-1, 1). The vanishing gradient problem is a problem that occurs when the activation functions product keeps decreasing nearring zero until it fully hits zero making the gradient completely vanish. Having a larger derivative helps it cope better with the vanishing gradient problem [3]. Other work has been done to try and overcome the vanishing gradient problem as discussed in Yuhuang Hu, Adrian huber, Jithendar Anumula, and Shih-Chii Lu's paper however we will not be covering network architectures that defeat the vanishing gradient problem like ResNet or Residual Networks. In the other paper's work on improving LeNet-5, they went from using the traditional version of Sigmoid activations to testing ReLU activations instead. ReLU activations are more effective than Tanh and Sigmoid activations as it has a much faster convergence speed thus making the model more accurate [3]. In the case of this paper's experiments, the LeNet-5 architecture we will be comparing is the difference between Tanh activations and ReLU activations. The ReLU activation is described as follows: if the ReLU function is greater than 0 it remains, otherwise it is 0 [3]. ReLU ignores any negative signals and is very similar to the reflection of signals by human neurons [3]. In general, their results showed that ReLU had a much higher accuracy much faster than the Sigmoid

activations which will be similar to results in comparing Tanh and ReLU activations. Another experiment was LeNet-5 being applied to the CIFAR-10 data set [5]. In their performance figures, they used a similar set up as the test this paper will apply but got a much higher accuracy in comparison to our implementation. In their idea about improving accuracy on the LeNet-5 network, they suggested increasing the depth of the layer [5]. Although they did not follow through with this implementation, this paper will implement and add more layers to the LeNet-5 network to see whether more depth will increase the accuracy of the LeNet-5 network architecture.

B. Sigmoid Activation Function

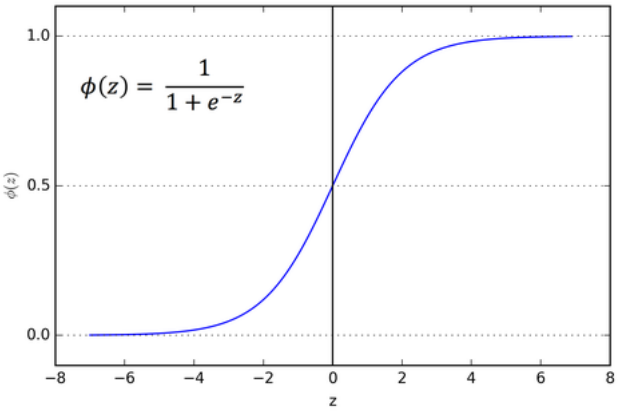

Fig. 4. Example graph of a sigmoid activation [6]

Sigmoid function here is pictured with the input being pushed to being only between (0, 1)

C. Sigmoid in Comparison to ReLU

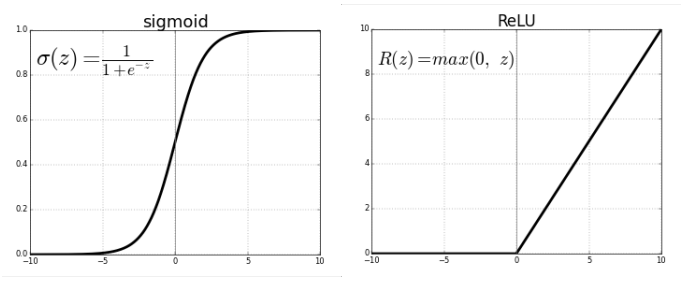

Fig. 5. Graph comparing a sigmoid activation and a ReLU activation [6]

In comparison, the ReLU function is the same or zero.

III. Proposed Approach

A. Simple Convolutional Neural Network Given By The Professor

The first CNN architecture that we will be implementing is one given by the professor. An image of the neural network is provided below.

This architecture is similar to LeNet-5 in that it takes only 1 input layer. The first convolutional layer has one input layer,

```
Net(
  (cnn_layers): Sequential(
    (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear_layers): Sequential(
    (0): Linear(in_features=196, out_features=10, bias=True)
  )
)
```

Fig. 6. CNN introduced in the course project guidelines

four output layers, a filter size of three, a stride of one, and a padding of one. After the first convolutional layer, there is a layer that applies normalization on the input followed by an ReLU activation. After the ReLU activation, a 2D max pooling is applied to the output of the batch normalization with a filter size of two, a stride of two, zero padding, and a dilation of one. That is followed up by another convolutional layer accepting four input channels and outputting four input channels with a very similar batch normalization and ReLU activation to follow. The linear layers have 196 in-features and ten out-features which will give us the output of this convolutional neural network. This architecture theoretically should run well with MNIST and Fashion-MNIST as those two data sets only have one channel as they are black and white images. The drawback of this however is that it should not work well with the CIFAR-10 architecture due to the data set having a start of three channels as it is an RGB image rather than a black and white image.

B. The LeNet-5 Architecture

The LeNet-5 architecture is the main focus of this research paper. The LeNet-5 architecture that was implemented used Tanh activations, three convolutional layers, two average pooling layers, and two fully connected layers. The hyperparameters in the first convolutional layer is one input channel with six output channels with a filter size of five, a stride of one, and a padding of two. The padding of two is required due to LeNet-5 normally accepting a 32x32 image while our images are 28x28. Both average pooling layers will have a filter size of two and a stride of two. The second convolutional layer has 6 input channels, 16 output channels, and a filter size of 5, a stride of 1, and a padding of 0 as we have already modified the input layer. Finally the fully connected layers have different starting in-features while the final out-features of the last linear layer always have 10 out-features. This architecture is more well known and so should give us a much better accuracy score with well known data sets. The drawback of this architecture is similar to the CNN provided by the professor as it also starts out with one input channel which will cause trouble for the CIFAR-10 data set.

C. A Modified Version Of LeNet-5

The modified version of LeNet-5 that we have changed implements both Tanh activations in the initial version and

```
def Net():
    model = nn.Sequential(
        #nn.Conv2d(in channels, out channels, kernel size, stride, padding)
        #our input has 1 channel as it is a grayscale image
        #need padding because our images are 28x28 while the LeNet takes in 32x32 inputs
        nn.Conv2d(in_channels = 1,out_channels = 6, kernel_size = 5, stride = 1, padding  = 2),
        nn.Tanh(),
        nn.AvgPool2d(kernel_size = 2, stride = 2),

        #next convolutional layer
        #no longer needs padding since weve adjusted the input
        nn.Conv2d(in_channels = 6, out_channels = 16, kernel_size = 5, stride = 1, padding = 0),
        nn.Tanh(),
        nn.AvgPool2d(kernel_size = 2, stride = 2),

        nn.Flatten(),
        nn.Linear(576, 128),
        nn.Tanh(),
        nn.Linear(128, 84),
        nn.Tanh(),
        nn.Linear(84, 10)
    )
    return model
```

Fig. 7. LeNet-5 architecture as described by an article [1]

then ReLU activations in the final version. It is a similar structure with two average pooling layers but instead has four convolutional layers and two linear layers. The first two convolutional layers are similar, but continues on to the third convolutional layer that has 16 input channels, 64 output channels while the rest of the hyperparameters stay the same. The next convolutional layer has 64 input channels, 120 output channels, and then also the same hyper parameters as the previous layers. Due to the additional layers, the amount of linear layers was able to be dropped as the output of the fourth linear layer was giving a smaller output than the original version of LeNet-5. The positive of this version of LeNet is that it allows for three input channels at the input layer which should result in a higher accuracy score as compared to using a black and white version of the CIFAR-10 data set.

```
def Net():
    model = nn.Sequential(
        #nn.Conv2d(in channels, out channels, kernel size, stride, padding)
        #our input has 1 channel as it is a grayscale image
        #need padding because our images are 28x28 while the LeNet takes in 32x32 inputs
        nn.Conv2d(in_channels = 1,out_channels = 6, kernel_size = 5, stride = 1, padding  = 2),
        nn.Tanh(),
        nn.AvgPool2d(kernel_size = 2, stride = 2),

        #next convolutional layer
        #no longer needs padding since weve adjusted the input
        nn.Conv2d(in_channels = 6, out_channels = 16, kernel_size = 5, stride = 1, padding = 0),
        nn.Tanh(),

        #additional convolutional layer 1
        nn.Conv2d(in_channels = 16, out_channels = 64, kernel_size = 5, stride = 1, padding = 0),
        nn.Tanh(),

        #Last additional convolutional layer
        nn.Conv2d(in_channels = 64, out_channels = 120, kernel_size = 5, stride = 1, padding = 0),
        nn.Tanh(),
        nn.AvgPool2d(kernel_size = 1, stride = 2),

        nn.Flatten(),
        nn.Linear(120, 84),
        nn.Tanh(),
        nn.Linear(84, 10)
    )
    return model
```

Fig. 8. LeNet-5 architecture described by an article which has been modified to have more convolutional layers and less fully connected layers [1]

D. Cross Entropy Loss Function

The loss function used in all of the implemented convolutional networks is the Cross Entropy Loss function. Cross entropy loss is described by pytorch as the criterion in which it computers the cross entropy between input logits and target [7]. It is useful as we did not use normalization in the Lenet-5 and modified LeNet-5 implementations and it is expected to take in unnormalized logits [7].

$$\ell(x, y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n} \cdot 1\{y_n \neq \text{ignore\_index}\}} l_n, & \text{if reduction} = \text{'mean'}; \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{'sum'}. \end{cases}$$

Fig. 9. Cross entropy loss formula given by pytorch [7]

IV. Experiment

A. Experiments On The CNN Given By The Professor

The first experiment is with the MNIST dataset. We used the Net class as given and tested the training loss, validation loss, testing loss, and accuracy over 25 epochs. All sets tested gave us an expected output of loss being reduced with every epoch while accuracy was being maintained between 95 percent on the first epoch and 98 percent for a majority of the run.

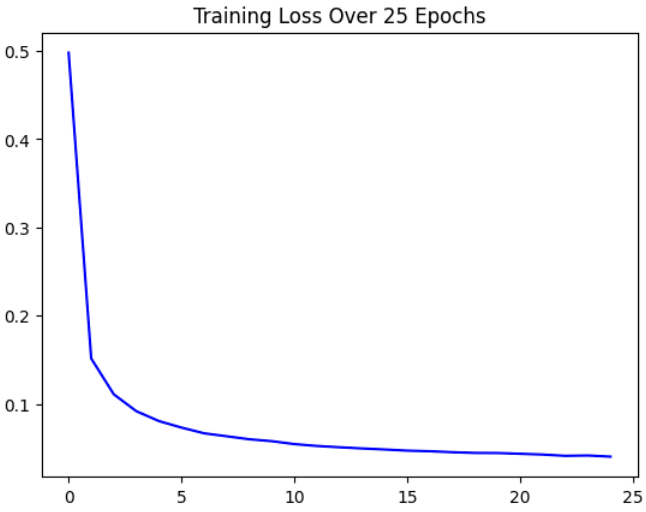

Fig. 10. MNIST Testing set loss using the given CNN over 25 epochs

Further figures will not be shown for other sets unless there is a large contrast in the data given and is not being trained as expected. However, there are more figures available in the code that was used to report this data. On the Fashion-MNIST dataset, the testing loss was very similar in nature while the maximum accuracy only managed to hit about 78 percent after 25 epochs. In similar fashion, the CNN performed relatively poor on the CIFAR-10 dataset only managing 33 percent accuracy by the end of the 25 epochs.

B. Experiments with the LeNet-5 Architecture

In a similar fashion, we tested the LeNet-5 architecture on the MNIST, Fashion-MNIST, and CIFAR-10 data set. With the MNIST dataset, it was going to be difficult to beat 98 percent accuracy by the end of the 25 epochs. LeNet-5 performed similarly levelling out at a max of 98 percent accuracy but actually started at a higher percentage of accuracy. The given

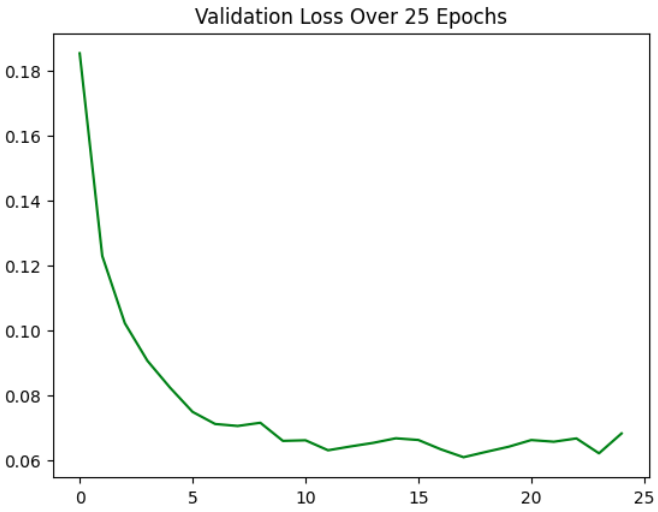

Fig. 11. MNIST Validation set loss using the given CNN over 25 epochs

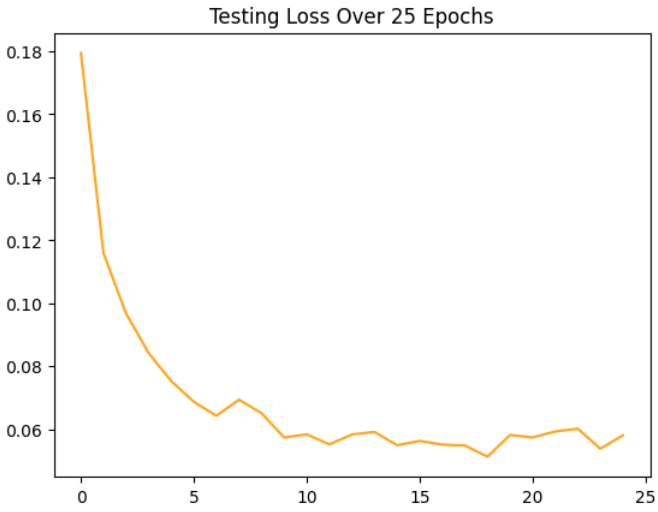

Fig. 12. MNIST Testing set loss using the given CNN over 25 epochs

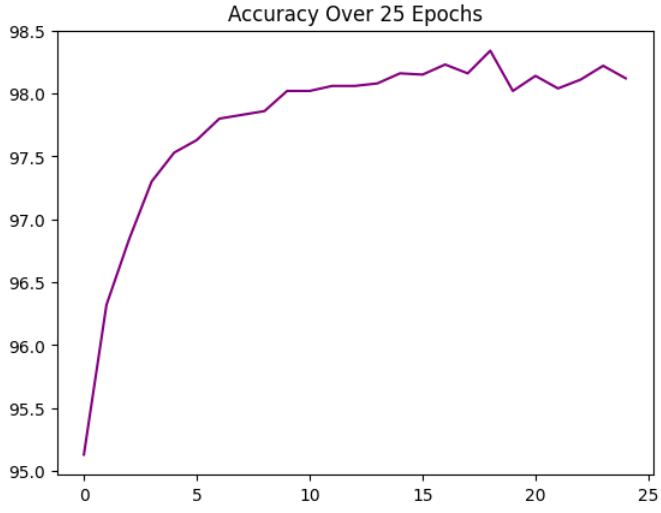

Fig. 13. MNIST Accuracy of the CNN over 25 epochs

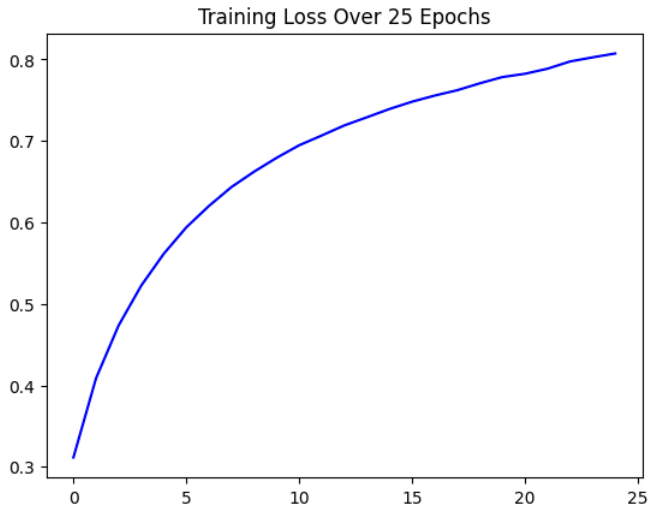

Fig. 14. MNIST Training Loss of LeNet-5 over 25 epochs surprisingly increasing

CNN architecture begun at around 95 percent accuracy while the LeNet-5 architecture started at around 96 almost 97 percent by the first epoch. However, the training set loss when using LeNet-5 actually increased in figure 14 while the validation and testing set loss followed the curve similarly to the given CNN.

Suprisingly, LeNet-5 struggled more with the Fashion-MNIST dataset than the sample CNN given by the professor. The accuracy of the given CNN was at 78 percent while the LeNet-5 implementation only managed to get up to about 75 percent. The accuracy was still similar, but it still showed that LeNet-5 performed worse than the sample CNN. This is highlighted even more when we get to the results of the CIFAR-10 testing. Using the same unmodified architectures for both showed similar performance results. The given CNN managed to score 5 percent higher accuracy than LeNet-5 with both maintaining similar curves in their set losses. With LeNet-5 performing worse than the sample CNN, it was decided that there would be modifications made to LeNet-5 which were described previously. One of the first modifications made to the LeNet-5 architecture was just changing the first input layer from one input channel to three input channels. These netted a 23 percent increase in accuracy alone. Changing the activations from Tanh to ReLU further netted a four percent increase changing the original accuracy on the CIFAR-10 from 23 percent all the way to 58 percent accuracy. From here we decided that we would modify the LeNet-5 architecture to see if adding more depth makes it more capable of handling data sets with images that have more data.

C. Experiments with the modified LeNet-5 Architecture

These experiments on the modified LeNet-5 architecture are tested with multiple different changes. This first version adds two convolutional layers, two extra Tanh activations, and one less linear layer. With MNIST, it is no suprise that it is still performing very well at 98 percent accuracy for a majority of the epochs. However, the same problem with training loss still occurs to this version of LeNet-5 as shown in figure 16, but does not occur to any other set even with other changes.

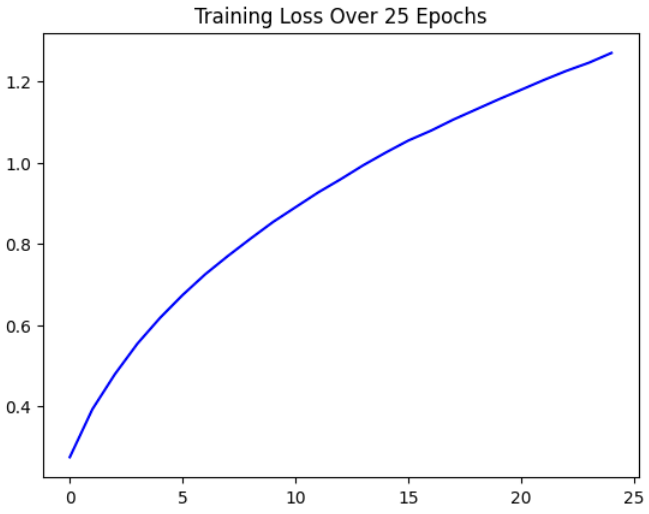

Fig. 15. MNIST Training Loss of Modified LeNet-5 over 25 epochs surprisingly increasing

Continuing on with Fashion-MNIST, the modified layers increased the accuracy of the network to 78 percent, matching the given CNN model. Finally, With CIFAR-10 we did see an increase in accuracy by 1 percent in the grayscaled version of the images. When we move to changing the input channels to three and increasing the learning rate from 1e-05 to 1e-3 , we further increase the accuracy of the model to 56 percent. Further modifications like changing the Tanh activations to ReLU activations further increased the accuracy of the model to 58 percent. Attempting to reduce the batch size only made the accuracy worse and changing the optimizer from Adam to SGD only made the loss worse. The last modification made was increasing the batch size back to 128, keeping the ReLU activations, and then multipling all input and output channels by three rather than only changing the first input channel to three. These modifications changed the accuracy from 58 percent max all the way to 68 percent on the best trained version of the model and down to 66 percent after all 25 epochs.

V. Conclusion

From the data gathered from all the different changes, it can be seen that LeNet-5 is very good at more simple data sets as it was specifically built for the MNIST data set. However, modifications to the original LeNet-5 architecture increases the accuracy but not enough to make it a viable option for modern day data sets. However, it does give a good idea on how convolutional neural networks work and how they can be modified to create better neural network models for the future.

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first . . .”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] S. Saxena, “The architecture of lenet-5,” Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/ (accessed May 10, 2023).

[2] K. O’Shea and R. Nash, “An introduction to Convolutional Neural Networks,” arXiv.org, https://arxiv.org/abs/1511.08458 (accessed May 10, 2023).

[3] Y. Wang, “Improvement of MNIST image recognition based on CNN - iopscience,” https://iopscience.iop.org/article/10.1088/1755-1315/428/1/012097/pdf, https://iopscience.iop.org/article/10.1088/1755-1315/428/1/012097 (accessed May 11, 2023).

[4] Y. Hu, A. Huber, J. Anumula, and S.-C. Liu, “A arxiv:1801.06105v3 [cs.NE] 5 jul 2019,” OVERCOMING THE VANISHING GRADIENT PROBLEM IN PLAIN RECURRENT NETWORKS, https://arxiv.org/pdf/1801.06105.pdf (accessed May 11, 2023).

[5] X. Zhang, ”The AlexNet, LeNet-5 and VGG NET applied to CIFAR-10,” 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering (ICBASE), Zhuhai, China, 2021, pp. 414-419, doi: 10.1109/ICBASE53849.2021.00083.

[6] A. Kathuria, “How to chose an activation function for your network,” Paperspace Blog, https://blog.paperspace.com/vanishing-gradients-activation-function/ (accessed May 10, 2023).

[7] Pytorch, “Cross Entropy Loss,” CrossEntropyLoss - PyTorch 2.0 documentation, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html (accessed May 10, 2023).