# CSI 5340 Homework Exercise 2

Name: Jayshil Patel

Student Number 300312550

## Question 1: Backpropagation and Gradient Descent

Implementation of back-propagation for a given set of functions involving matrices. The primary objective is to compute partial derivatives of a loss function *L* with respect to *A, B,* and *C* using the chain rule. The report outlines the step-by-step manual computation, followed by using these gradients for gradient descent to optimize the matrices *A, B,* and *C.* Additionally, the results are compared with automatic differentiation to validate the accuracy of the manual implementation.

### 1.1 Computing Gradients

- To compute $\frac{\partial L}{\partial A}$, we had to use the chain rule. So $\frac{\partial L}{\partial A} = \frac{\partial L}{\partial w} \times \frac{\partial w}{\partial z} \times \frac{\partial z}{\partial u} \times \frac{\partial u}{\partial y} \times \frac{\partial y}{\partial A}$
- To compute $\frac{\partial L}{\partial B}$, the same way using the chain rule we have $\frac{\partial L}{\partial B} = \frac{\partial L}{\partial w} \times \frac{\partial w}{\partial z} \times \frac{\partial z}{\partial v} \times \frac{\partial v}{\partial B}$
- To compute $\frac{\partial L}{\partial C}$, same what we have $\frac{\partial L}{\partial C} = \frac{\partial L}{\partial w} \times \frac{\partial w}{\partial C}$

### 1.2 Trend in loss vs Epochs

The graph typically shows the training loss on the y-axis and the number of epochs on the x-axis.

Start of Training: At the beginning of training, the loss is usually relatively high as the model's parameters are randomly initialized, and the predictions are far from the actual target values.

Convergence: As the training progresses, the loss generally decreases. This is because the model is learning and adjusting its parameters to better fit the training data.

Stabilization: Over time, the loss stabilize or decrease at a slower rate. This is an indication that the model is converging and further significant improvements are becoming harder to achieve.

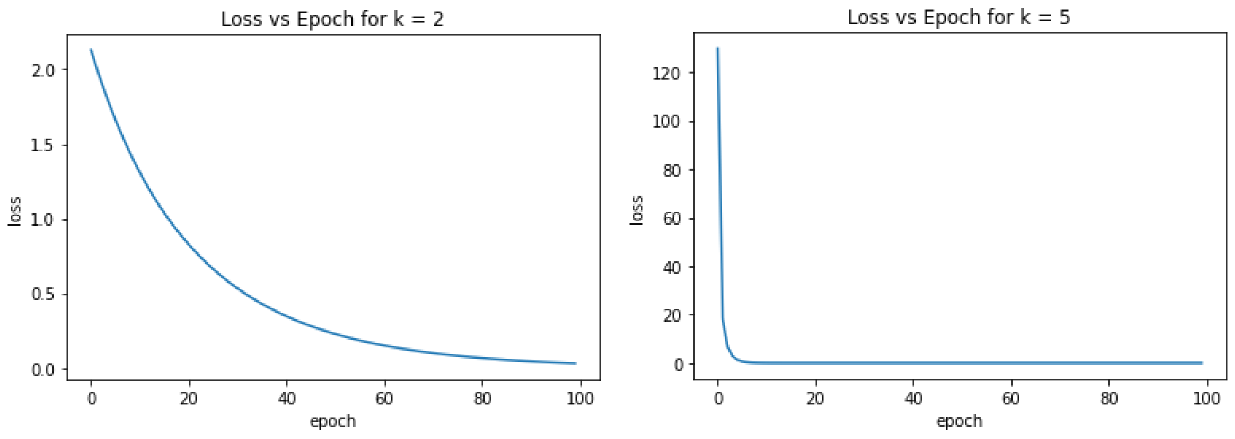


Figure 1 Loss vs Epoch for manual computation

Initially, at the beginning of training, we observed a faster reduction in loss compared to lower K because the model has more dimensions to capture information from the input. As the value of K increases we can observe a steeper initial part of the loss curve. Consequently, the loss curve takes more epochs to converge to a minimum. The lower complexity of the input matrix (A, B, C) results in a slower decrease in K but the initial loss is higher. As the value of k(dimension of input matrices) increases the initial loss value is very high, but decreases very rapidly compared to lower dimensions of input.

## 1.3 Automatic Differentiation Library VS manual comptation

Torch.autograd was used as the automatic differentiation library. The loss obtained for each iteration was overlapping with the loss obtained from manual backpropagation and gradient descent.

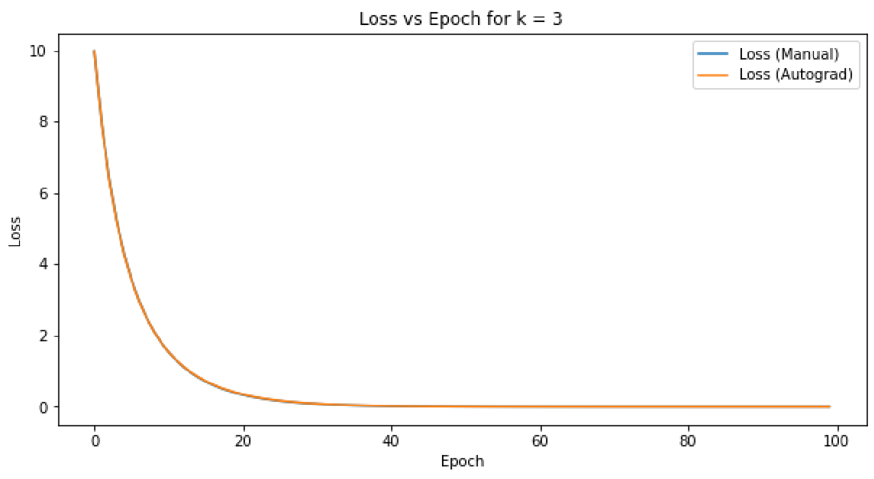


Figure 2 Loss vs Epoch comparing manual and autograd’s loss

Here we can observe that loss obtained for each Epoch for both manual computation of gradients and using Torch.autograd was completely similar.

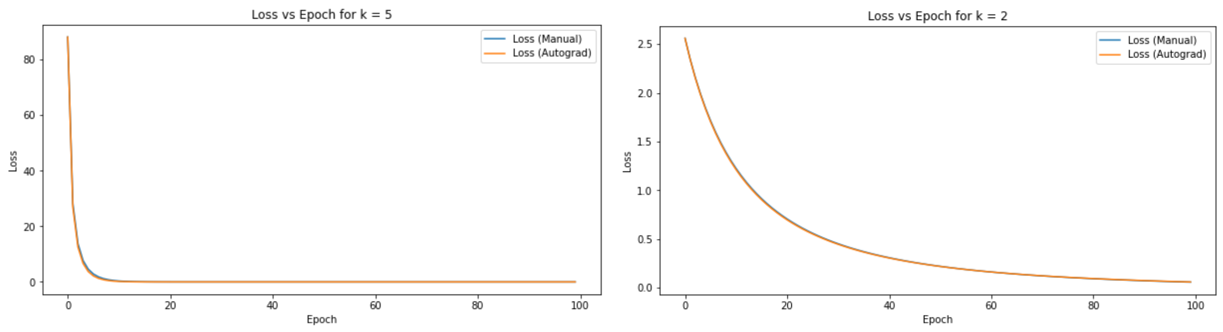
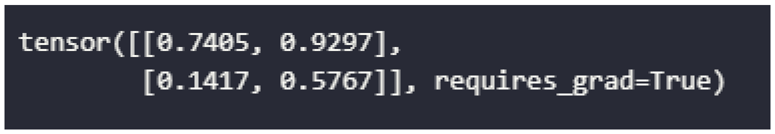


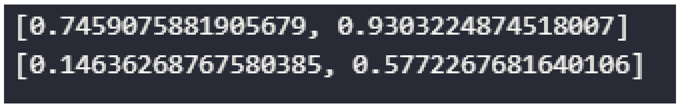
Figure 3 Comparing loss trends for manual and autograd for different values of K

Even for different values of K (dimension of input vector and dimension of input matrices), the loss curve overlaps with the output of Torch.autograd library which concludes that the manual implementation of the backpropagation was correct. Further from the images, we can observe that the new optimized matrices obtained using the automatic differentiation library were almost similar to the ones computed manually. A slight difference is observed because of approximations and rounding errors used by auto differentiation libraries built for speed and efficiency Different libraries may use slightly different floating-point arithmetic implementations, which can lead to small differences in the results. Numerical Stability: The sigmoid function and its derivatives can lead to numerical instability, especially for very small or very large inputs. Libraries may use optimized or stabilized versions of these functions, which could affect the results. Gradient Computation: Automatic differentiation libraries like PyTorch use advanced algorithms for gradient computation, which may differ from the manual gradient computation method. Small differences in the gradient computation can accumulate and lead to variations in the optimized matrices.
Optimization Algorithm: Even with the same loss function, the optimization algorithm (e.g., gradient descent) might have differences in implementation, convergence criteria, or step size adaptation, leading to slightly different optimization paths.



```
tensor([[0.7405, 0.9297],
        [0.1417, 0.5767]], requires_grad=True)
```

Figure 4 Optimized values of A using Autograd library



```
[0.7459075881905679, 0.9303224874518007]
[0.14636268767580385, 0.5772267681640106]
```

Figure 5 Optimized values of A using manual computation

# Question 2:- MNIST dataset

## 2.1 Base Classifiers

### 2.1.1 Softmax Regression Model (Normal):

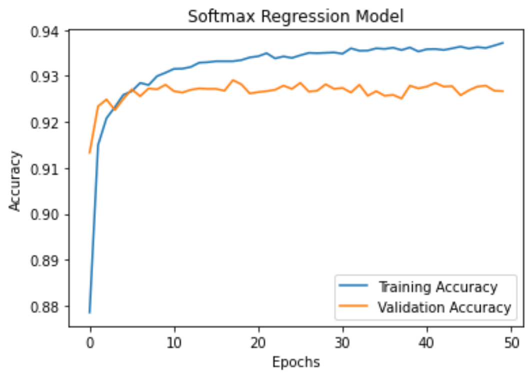


Figure 6 Softmax Regression Model

- The accuracy starts at approximately 87.98% and steadily increases over the epochs, reaching around 93.52% by the 30th epoch.
- The accuracy continues to increase but at a slower rate, indicating that the model is learning and refining its predictions as more epochs progress.
- The trend shows a consistent improvement in accuracy, suggesting that the model is effectively learning the features and patterns in the data.

### 2.1.2 MLP Model (Normal):

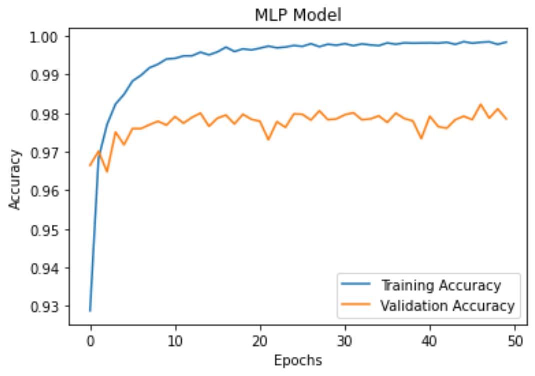


Figure 7 MLP model

- The accuracy starts at approximately 92.72% and demonstrates a rapid increase in accuracy within the initial epochs, reaching around 99.84% by the 30th epoch.
- The model quickly converges to a very high accuracy, suggesting that the architecture and learning process of an MLP model are effective for this dataset.
- The trend indicates that the model effectively learns the features and patterns in the data and achieves a high accuracy within a relatively small number of epochs.

### 2.1.3 CNN Model (Normal):

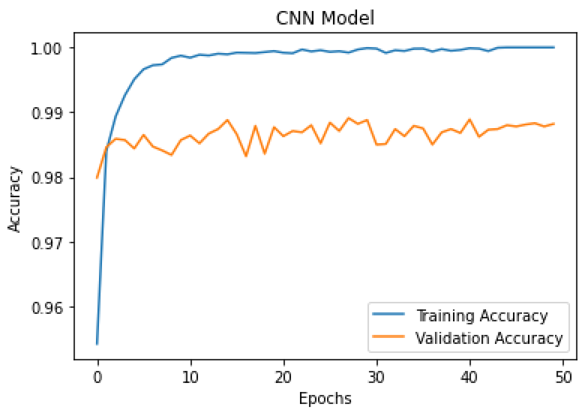


Figure 8 CNN

- The accuracy starts at approximately 95.37% and shows a rapid increase within the initial epochs, reaching around 99.97% by the 30th epoch.
- CNNs are highly effective for image-related tasks due to their ability to capture hierarchical features, resulting in a rapid and substantial improvement in accuracy.
- The trend indicates that the CNN model effectively learns complex features in the images and achieves a very high accuracy within a relatively small number of epochs.

## 2.2 Classifiers with dropout

### 2.2.1 Softmax Regression Model with Dropout:

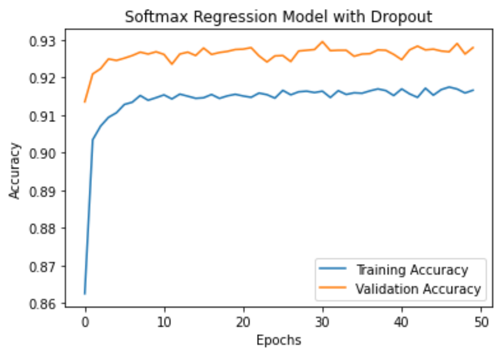


Figure 9 Softmax with Dropout

- The accuracy starts at approximately 86.20% and shows a similar increasing trend but with more fluctuations compared to the normal softmax model.
- Dropout helps reduce overfitting, and despite fluctuations, the accuracy steadily improves, reaching around 91.69% by the 30th epoch.
- The fluctuations may be due to dropout's stochastic nature, where different sets of neurons are dropped during each training batch, affecting the training process.

## 2.2.2 MLP Model with Dropout:

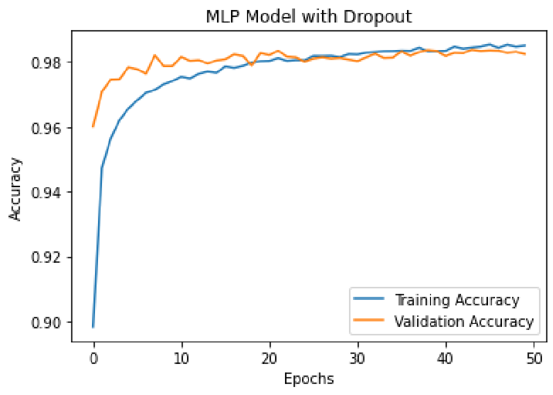


Figure 10 MLP with Dropout

- The accuracy starts at approximately 90.13% and exhibits a similar increasing trend to the normal MLP model but with more fluctuations and a slightly slower rate of increase.
- Dropout helps reduce overfitting, leading to a more generalizable model. However, this can introduce fluctuations in the accuracy as the dropout layers randomly "turn off" neurons during training.
- The fluctuations in accuracy indicate the stochastic nature of dropout, affecting the model's performance during training.

## 2.2.3 CNN Model with Dropout:

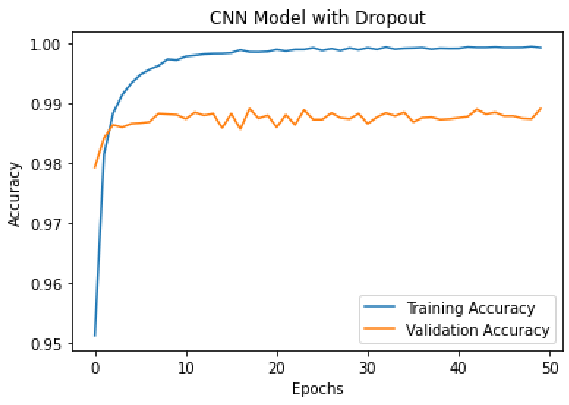


Figure 11 CNN with Dropout

- The accuracy starts at approximately 95.34% and follows a similar trend to the normal CNN model but with more fluctuations and a slightly slower rate of increase.
- Dropout helps reduce overfitting, but it may introduce fluctuations in accuracy as neurons are randomly dropped during training.
- The fluctuations in accuracy may be attributed to the stochastic nature of dropout, affecting the model's performance during training.

## 2.3 Classifiers with Batch Normalization

### 2.3.1 Softmax Regression Model with BatchNormalization:

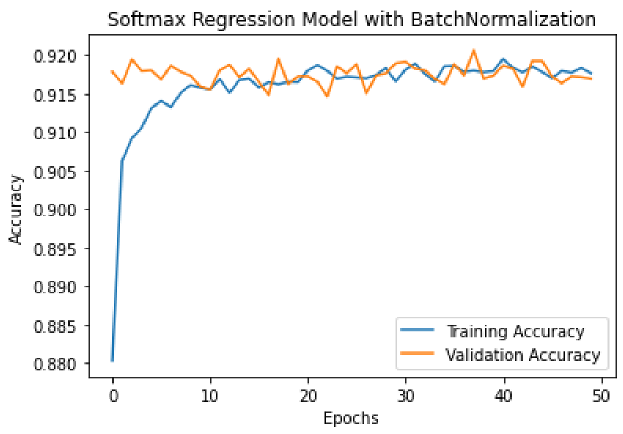


Figure 12 Softmax Regression with BatchNormalization

- The accuracy starts at approximately 88.06% and exhibits a smooth and steady increase throughout the epochs, reaching around 91.83% by the 30th epoch.
- Batch normalization helps in stabilizing and accelerating the learning process, leading to a more consistent and faster convergence compared to the normal softmax model.
- The trend suggests that batch normalization allows the model to learn more efficiently and achieve a higher accuracy within the same number of epochs.

### 2.3.2 MLP Model with BatchNormalization:

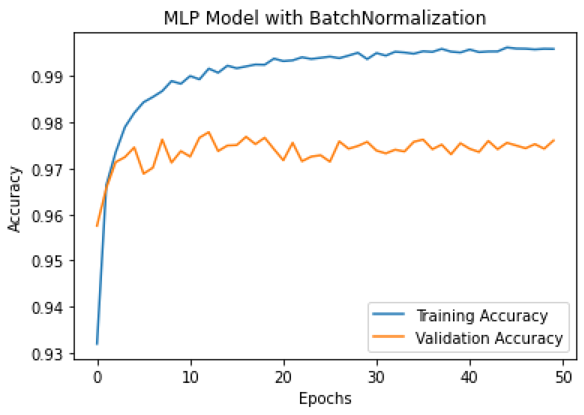


Figure 13 MLP Model with BatchNormalization

- The accuracy starts at approximately 93.02% and follows a smooth and steady increase, reaching around 99.48% by the 30th epoch.
- Batch normalization helps in stabilizing and accelerating the learning process, leading to a smoother accuracy curve and faster convergence compared to the normal MLP model.
- The trend suggests that batch normalization allows the model to learn more efficiently and achieve a high accuracy within the same number of epochs.

### 2.3.3 CNN Model with BatchNormalization:

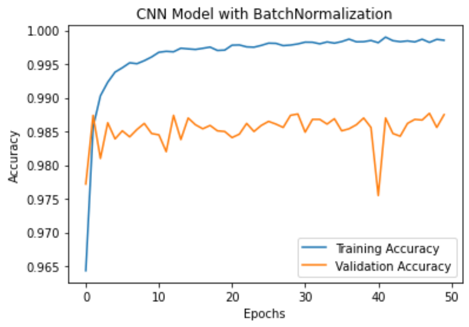


Figure 14 CNN with BatchNormalization

- The accuracy starts at approximately 96.45% and exhibits a smooth and steady increase throughout the epochs, reaching around 99.88% by the 30th epoch.
- Batch normalization helps in stabilizing and accelerating the learning process, resulting in a smoother accuracy curve and faster convergence compared to the normal CNN model.
- The trend suggests that batch normalization allows the model to learn more efficiently and achieve a high accuracy within the same number of epochs.

## 2.4 Difference in the same model with different Layers

### 2.4.1 Comparing all types of Softmax Classifiers.

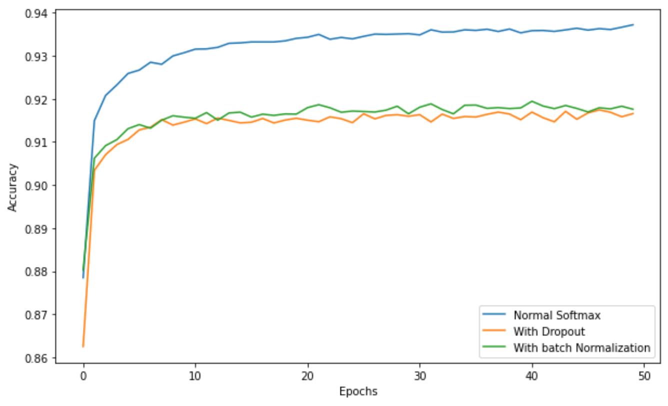


Figure 15 All types of Softmax Classifiers

- The normal softmax regression model shows a consistent and steady increase in accuracy over the epochs.
- The softmax regression model with dropout and batch normalization exhibits slightly more fluctuations but still demonstrates an overall increasing trend in accuracy.

## 2.4.2 Comparing all types of MLP Classifiers

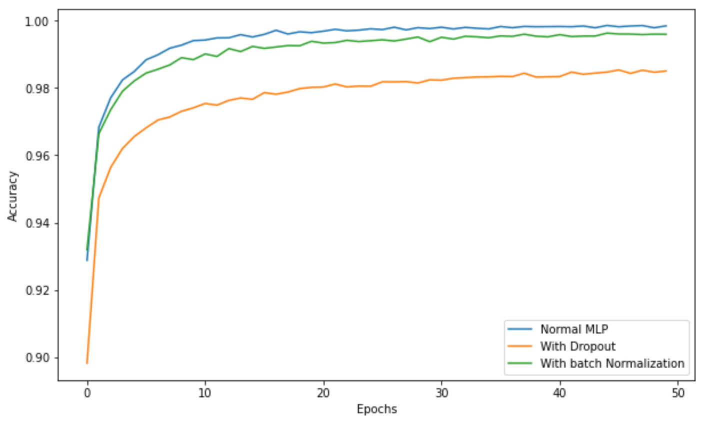


Figure 16 All types of MLP Classifiers

- The normal MLP model demonstrates a rapid and consistent increase in accuracy, converging quickly to a very high accuracy.
- The MLP model with dropout exhibits a similar increasing trend but with more fluctuations, highlighting the effect of dropout in reducing overfitting.
- The MLP model with batch normalization showcases a smooth and steady increase in accuracy, demonstrating the benefits of batch normalization in stabilizing and accelerating the learning process.

## 2.4.3 Comparing all types of CNN

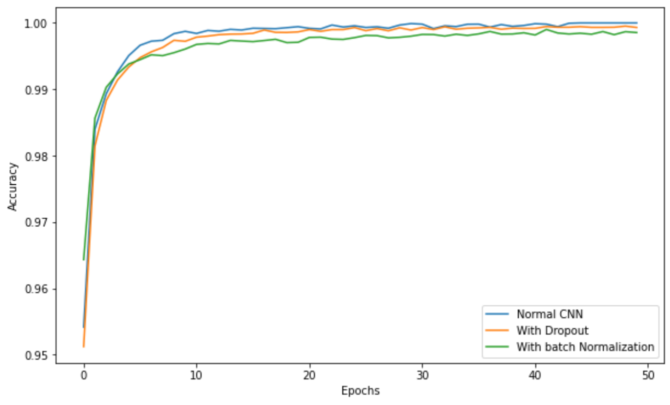


Figure 17 All types of CNN

- The normal CNN model demonstrates a rapid and consistent increase in accuracy, converging quickly to a very high accuracy, showcasing the effectiveness of CNNs for image-related tasks.
- The CNN model with dropout exhibits a similar increasing trend but with more fluctuations, highlighting the effect of dropout in reducing overfitting and potentially introducing more variability in the accuracy.

- The CNN model with batch normalization showcases a smooth and steady increase in accuracy, demonstrating the benefits of batch normalization in stabilizing and accelerating the learning process in CNNs.

These observations emphasize the importance of architectural choices, such as dropout and batch normalization, in enhancing the training dynamics and performance of CNNs for image classification tasks.

## 2.5 Comparing all Model's performance with the same layers

### 2.5.1 Comparison: Normal Softmax Regression vs. Normal MLP vs. Normal CNN

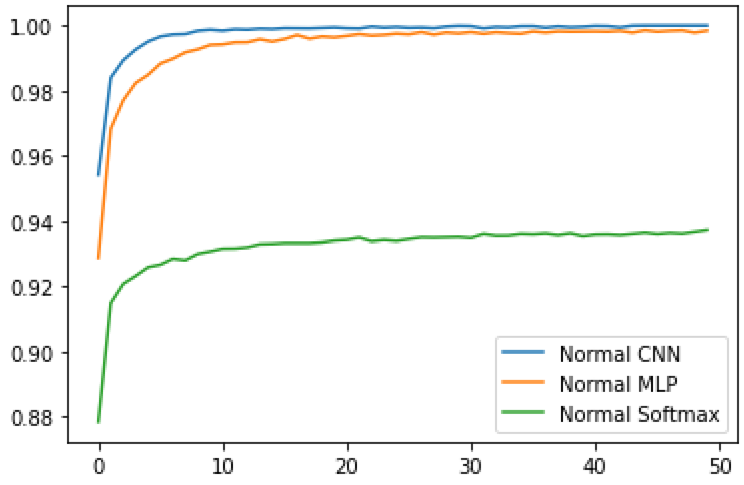


Figure 18 Comparing All base model performance

Considering the outputs and observed trends, the CNN model achieves the highest accuracy and demonstrates rapid convergence, making it superior for image classification tasks like the MNIST dataset. MLP also achieves high accuracy, showcasing its versatility for various tasks. Softmax Regression, while simpler and achieving a moderate accuracy, may be limited in handling more complex tasks compared to MLP and CNN.

In conclusion, for image-related tasks like MNIST digit classification, the CNN model stands out as the most effective and efficient choice based on the achieved accuracy and convergence speed.

### 2.5.2 Comparison: Softmax Regression with Dropout vs. MLP with Dropout vs. CNN with Dropout

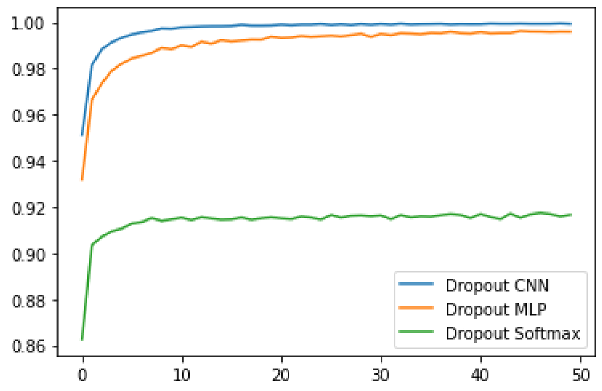
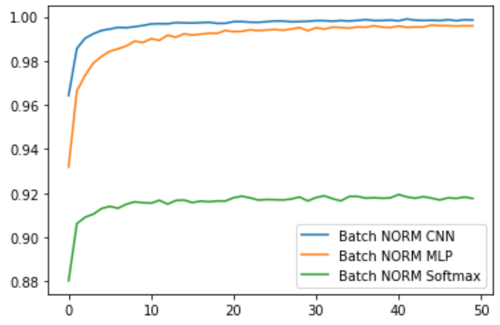


Figure 19 Comparing All models with dropout

In summary, when dropout is applied, CNN with dropout achieves the highest accuracy, making it the most effective choice for image classification tasks. MLP with dropout also achieves high accuracy, showcasing its effectiveness. Softmax regression with dropout improves accuracy compared to the regular softmax regression, making it a viable option, especially for simpler tasks. However, for image classification, the CNN model with dropout stands out as the most effective choice based on the achieved accuracy.

### 2.5.3 Comparison: Softmax Regression with BatchNormalizaiton vs. MLP with BatchNormalizaiton vs. CNN with BatchNormalizaiton



In summary, when batch normalization is applied, CNN with batch normalization achieves the highest accuracy, making it the most effective choice for image classification tasks. MLP with batch normalization also achieves a very high accuracy, showcasing its effectiveness. Softmax regression with batch normalization improves accuracy compared to regular softmax regression, making it a viable option, especially for simpler tasks. However, for image classification, the CNN model with batch normalization stands out as the most effective choice based on the achieved accuracy.

**Dropout Effect on CNN:**

Dropout is a powerful regularization technique that helps prevent overfitting by randomly dropping out (setting to zero) a fraction of the neurons during each training batch. This forces the model to learn more robust and generalizable features.
Even when applied to just the final layer, dropout can significantly enhance accuracy and convergence speed. This is due to the regularization effect that dropout has on the weights and the ability to prevent overfitting.

**Batch Normalization Effect on CNN:**

Batch normalization is another regularization technique that helps stabilize and accelerate the learning process by normalizing the activations of each layer in a mini-batch.
In CNNs, batch normalization can be applied after each convolutional or fully connected layer. This helps in addressing the internal covariate shift and stabilizing the learning process.
However, batch normalization in every layer might slows down the training process due to the extra computations involved in each layer. It's a trade-off between faster convergence and the computational cost.

**Comparing Dropout and Batch Normalization:**

The fact that dropout, even when applied only to the last layer, outperforms batch-normalized CNNs in terms of accuracy and convergence speed is a testament to the regularization power of dropout, especially in the context of CNNs.
Batch normalization being applied in every layer indeed adds computational overhead and slows down the training process compared to dropout, which is only applied to the last layer.