Assignment 3

By – Abhishek Kumar and Chinmay Katpatal

Part I

Income or Census Income dataset is a real-world dataset based on demographic data from the 1994 U.S. Census. For our analysis, we are using the updated Income dataset from UBlearns. There are 32561 observations and 15 features in the dataset. The continuous features are – 'age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss' and 'hours.per.week' while the rest are categorical which include 'workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country' and 'income'.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
```

Main statistics of the continuous variables are given below:

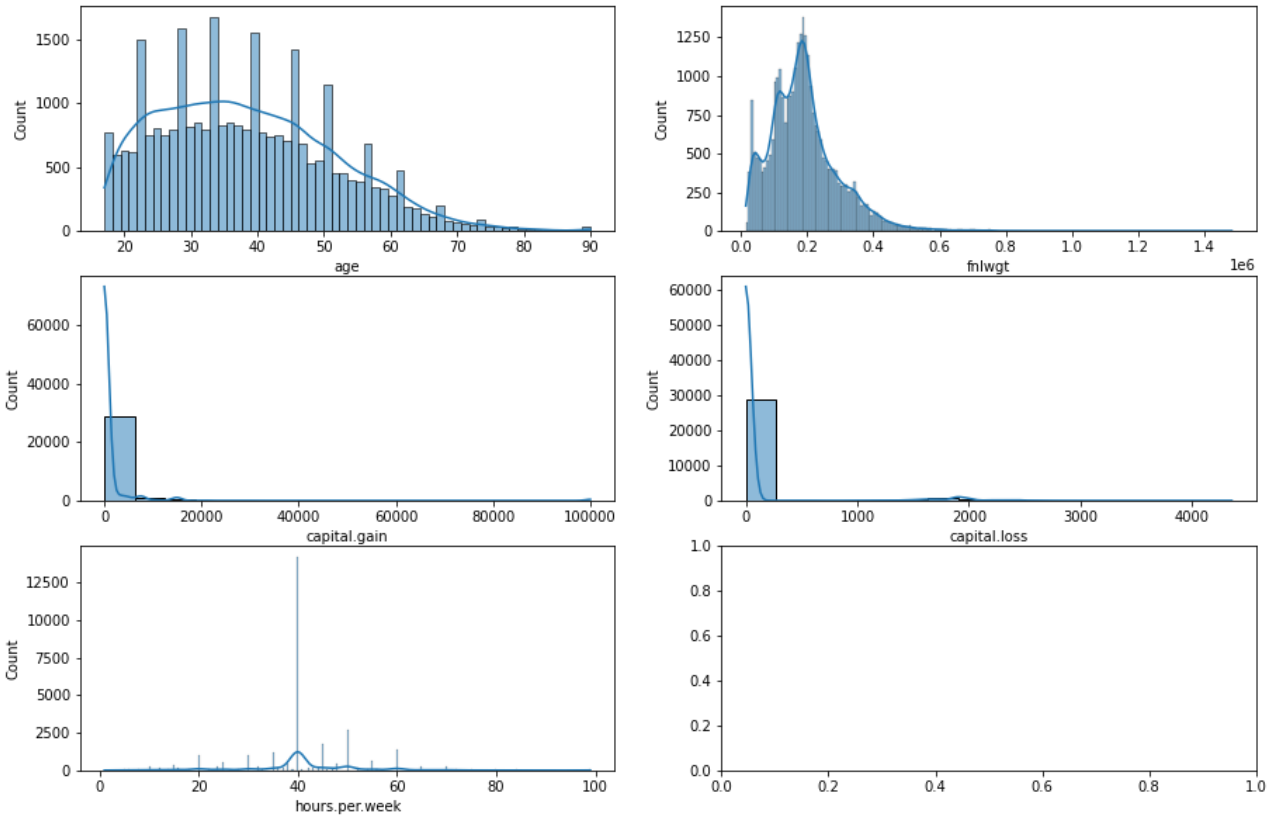
	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561	32561	32561	32561	32561	32561
mean	38.581647	189778.4	10.080679	1077.648844	87.303830	40.437456
std	13.640433	105550.0	2.572720	7385.292085	402.960219	12.347429
min	17.000000	12285.0	1.000000	0.000000	0.000000	1.000000
25%	28.000000	117827.0	9.000000	0.000000	0.000000	40.000000
50%	37.000000	178356.0	10.000000	0.000000	0.000000	40.000000
75%	48.000000	237051.0	12.000000	0.000000	0.000000	45.000000
max	90.000000	1484705	16.000000	99999.000000	4356.000000	99.000000

The dataset has missing values in some of the features which are represented by '?'. We have decided to drop these rows before further analysis.

EDA

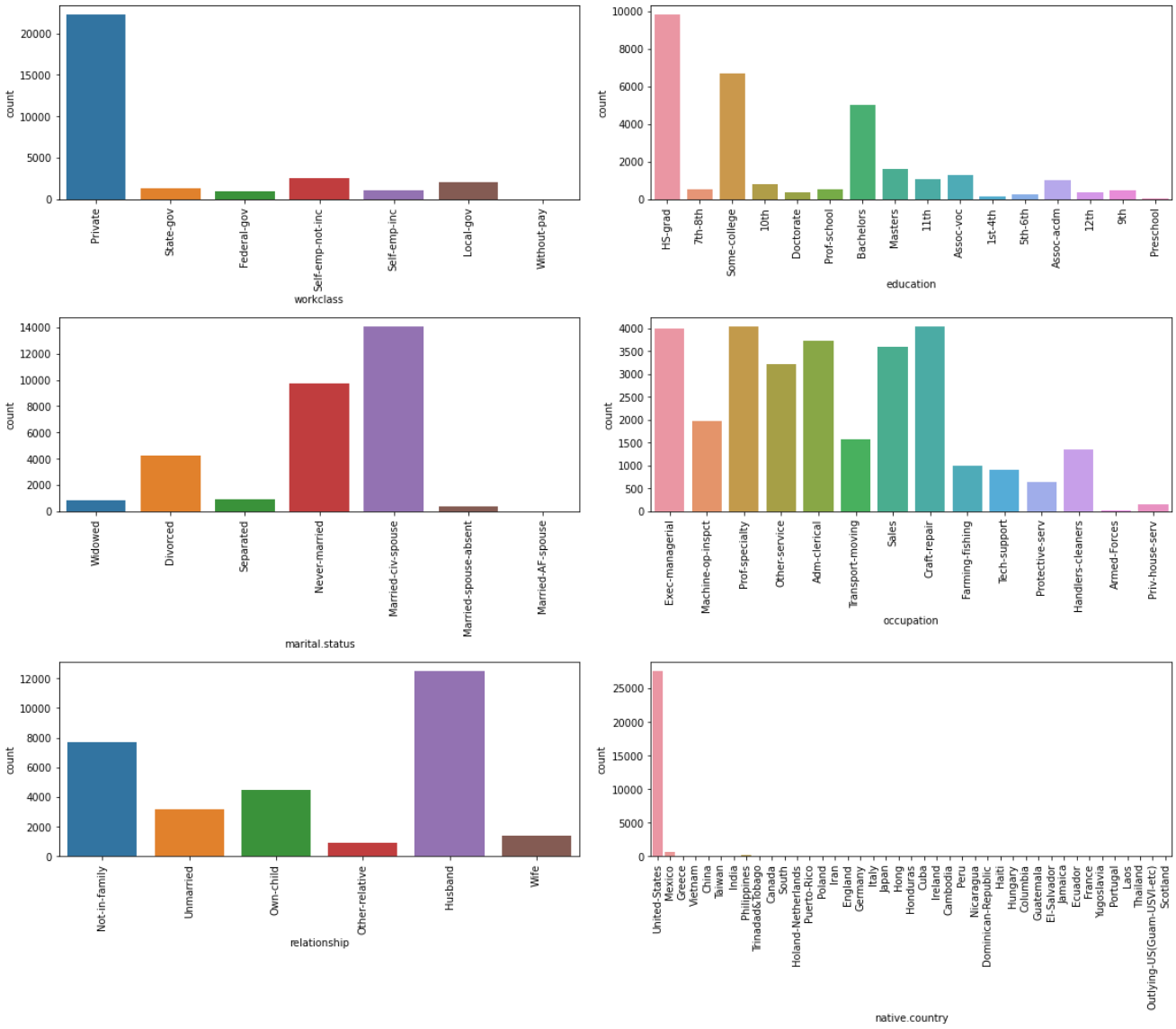
We performed EDA before normalizing the continuous variables (or one-hot encoding the categorical variables) so that we could observe any trends in the real data at the original scale. The results of EDA follow:
First, we've done a univariate analysis.

Histograms for continuous variables



We note that other than age, all variables are highly skewed. Age has a somewhat uniform distribution. However, there are multiple spikes at fixed intervals. This could be due to rounding off.
Both capital.gain and capital.loss have mostly zero values and a few extremely high values.
40 hours per week is the most common work duration per week.

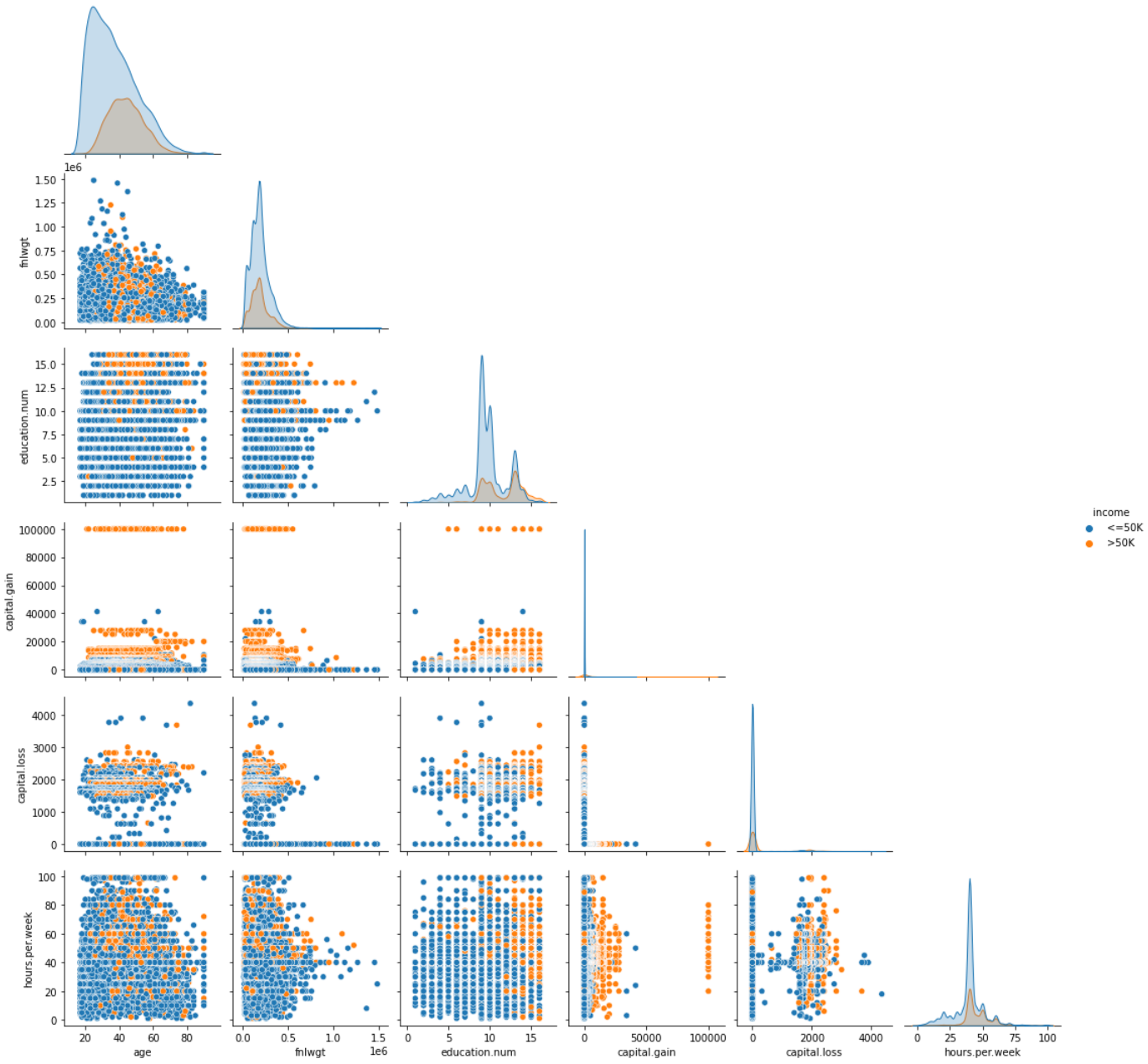
Countplots for categorical variables



We can note that the most common work class is ‘Private’ (60 to 70%). ‘Without pay’ work class is almost negligible.

Most of the working population has an HS degree, followed by a college degree and Bachelors.

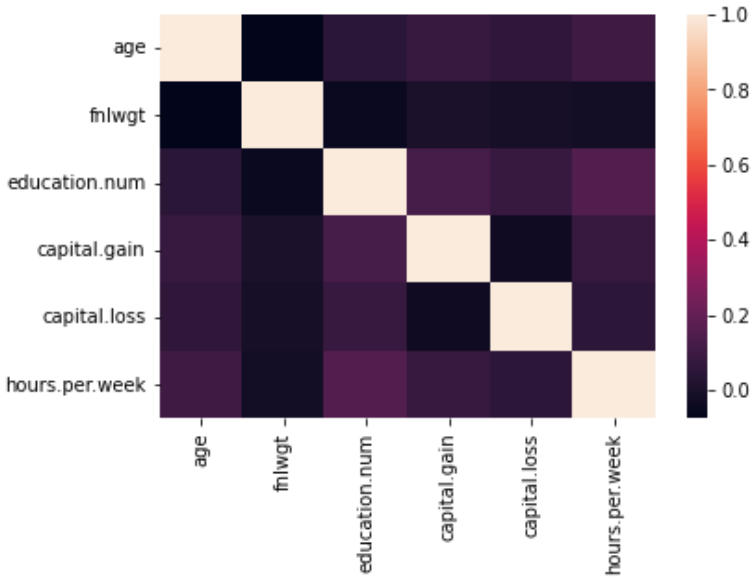
Bivariate analysis



All variables seem to be uniformly distributed for both levels of income. But we can still see some trends, for example, young people (<30) are predominantly in the <=50K income category, also, if we consider the combination of education.num and hours per week then clearly people with a lower level of education and working fewer hours are in the lower-income bracket.

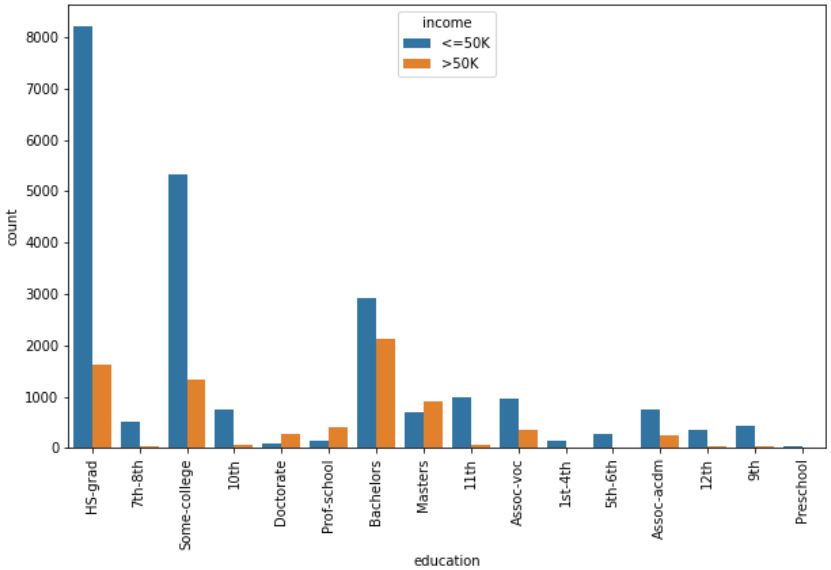
Some other graphs

Correlation Plot



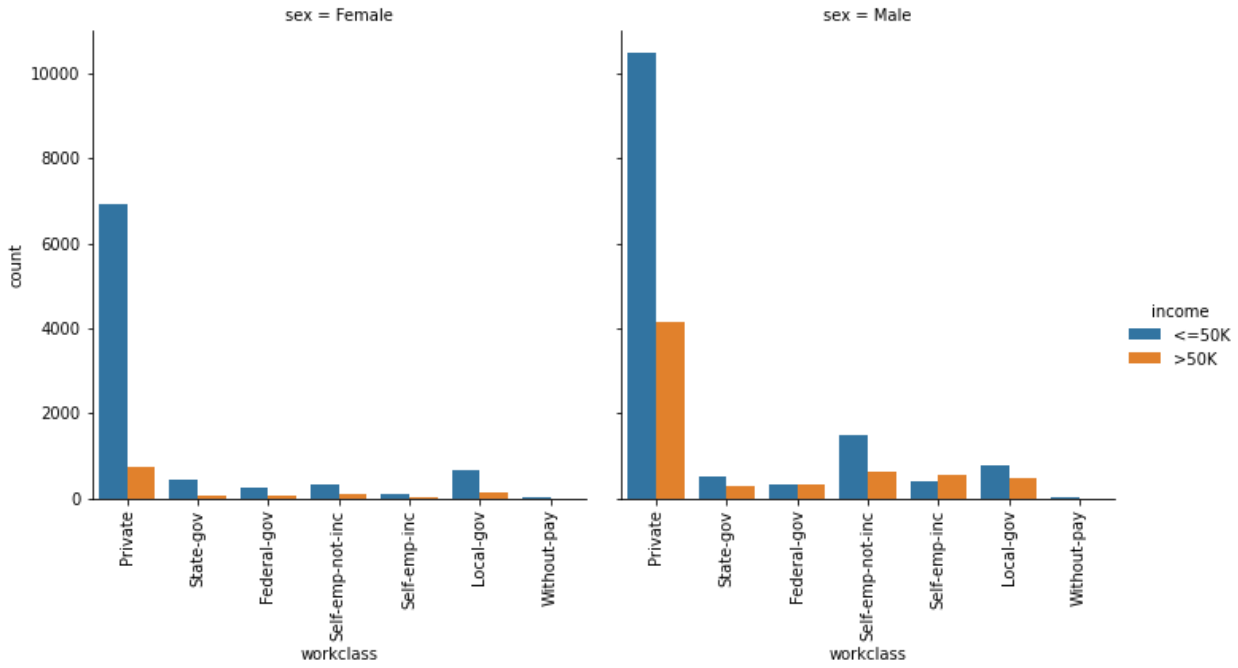
Clearly, there is a negligible correlation between each pair of continuous variables.

Let's explore the relationship between education and income level in detail



All levels of education have mostly lower-income populations. However, this discrepancy is least for Bachelors level education. And for Doctorate and Post-Doctorate level education this trend is reversed.

Let us also check whether the combination of work-class and sex has an influence on Income.



As we can see, both males and females have similar trends in the workclass. However, the total number of males in the workforce is much higher for all workclass except ‘without-pay’.

Preprocessing

We have already dropped the rows containing missing values.

We found that the column ‘education.num’ was already an encoded column for ‘education’. So, we removed that column.

We used one-hot encoding to encode the categorical columns. Thus, the total number of features increased to 81 (excluding the target variable ‘income’)

We also scaled the continuous features using min-max scaling.

Architecture Structure of NN

Q.) How many input neurons are there?
- There are 81 input neurons corresponding to 81 features

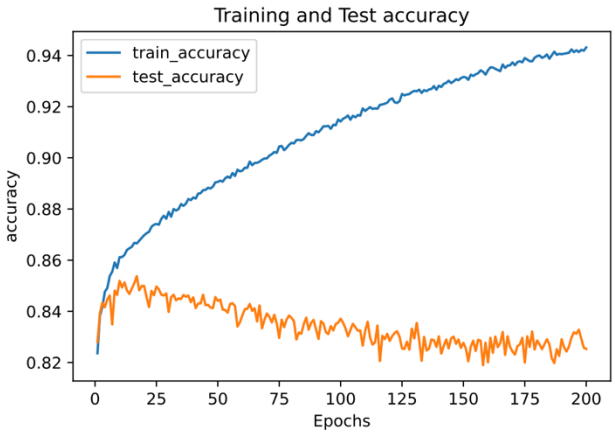
Q.) What activation function will you choose?
- We've chosen ReLu activation function for the input layer

Q.) What is the number of hidden layers? (1, 2, or 3?)
- 3 Hidden layers

Q.) What is the size of each hidden layer?
- 128 neurons in the first layer and 256 neurons in the next two

Q.) What activation function is used for the hidden and output layer?
- ReLu for Hidden layers, Sigmoid for the Output layer

Graphs for Training and Testing Accuracy and Loss



From the above graphs, we can note that the test accuracy maximized (equivalently the test loss minimized) at about 20 epochs. While the training accuracy continued increasing, it only resulted in overfitting to the training data.

Part II

Hyperparameter Tuning

Dropout Tuning

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1	75.00	0.2	74.93	0.4	75.00
Optimizer	Adam Learning rate = 0.1		Adam Learning rate = 0.1		Adam Learning rate = 0.1	
Activation Function	ReLu		ReLu		ReLu	
Initializer	Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02	

Optimizer Tuning

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1	84.06	0.1	75.00	0.1	75.02
Optimizer	Adamax Learning rate = 0.001		FTRL Learning rate = 0.001		AdaDelta Learning rate = 0.001	
Activation Function	ReLu		ReLu		ReLu	
Initializer	Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02	

Activation Function Tuning

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1	80.74	0.1	75.00	0.1	75.02
Optimizer	Adamax Learning rate = 0.001		Adamax Learning rate = 0.001		AdaDelta Learning rate = 0.001	
Activation Function	Sigmoid		GeLu		Softplus	
Initializer	Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02		Random Normal Mean = 0.0, Std Dev = 0.02	

Reasoning for trying the above NN setups

We tried out different random values for Drop out tuning which had no specific effect but instead reduced the accuracy of the model. So, we decided not to use a drop-out hyperparameter in the model.

For optimizer tuning I tried 3 different optimizing functions which then gave different accuracies, Adamax gave the highest accuracy whereas Ftlr and AdaDelta gave similar accuracies, so we finally settled with AdaMax. AdaMax is an extension to the Adam version of gradient descent that generalizes the approach to the infinite norm (max) and may result in a more effective optimization on some problems.

In the activation function tuning, we used Sigmoid, Gelu, and Softplus where Sigmoid gave the highest accuracy and Gelu and Softplus gave similar accuracies. The main reason why we use the sigmoid function is that it exists between (0 to 1). Therefore, it is especially used for models where we must predict the probability as an output (Just like for our dataset target variable). Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

Final (base) Model

The final model we chose was the one without Dropout and Initializer because it gave the highest accuracy.

Model Improvisation Methods

Early Stopping
Learning Rate Scheduler
L1 and L2 regularization

These are the three methods we used for model improvisation as per our knowledge but none of them had an improvement in the accuracy as such.

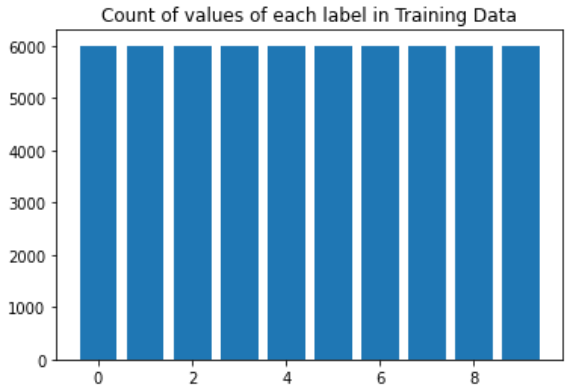
Part III

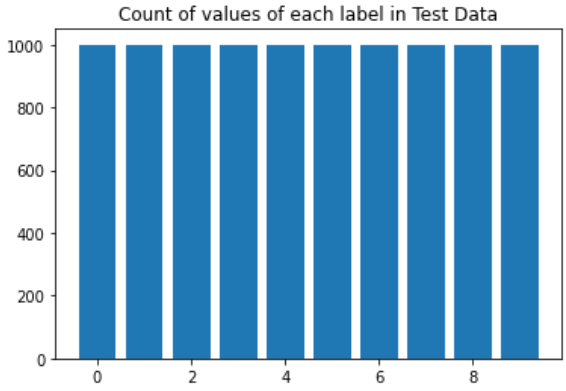
We have worked on the Fashion-MNIST dataset which comprises 60,000 training and 10,000 test examples of images of clothing items. There are 10 classes in which the examples are divided. Each image is a 28 pixels x 28 pixels (total 784 pixels) grayscale image. The pixel value is an integer between 0 and 255 with higher numbers corresponding to darker pixels.

The classes are mentioned below:

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

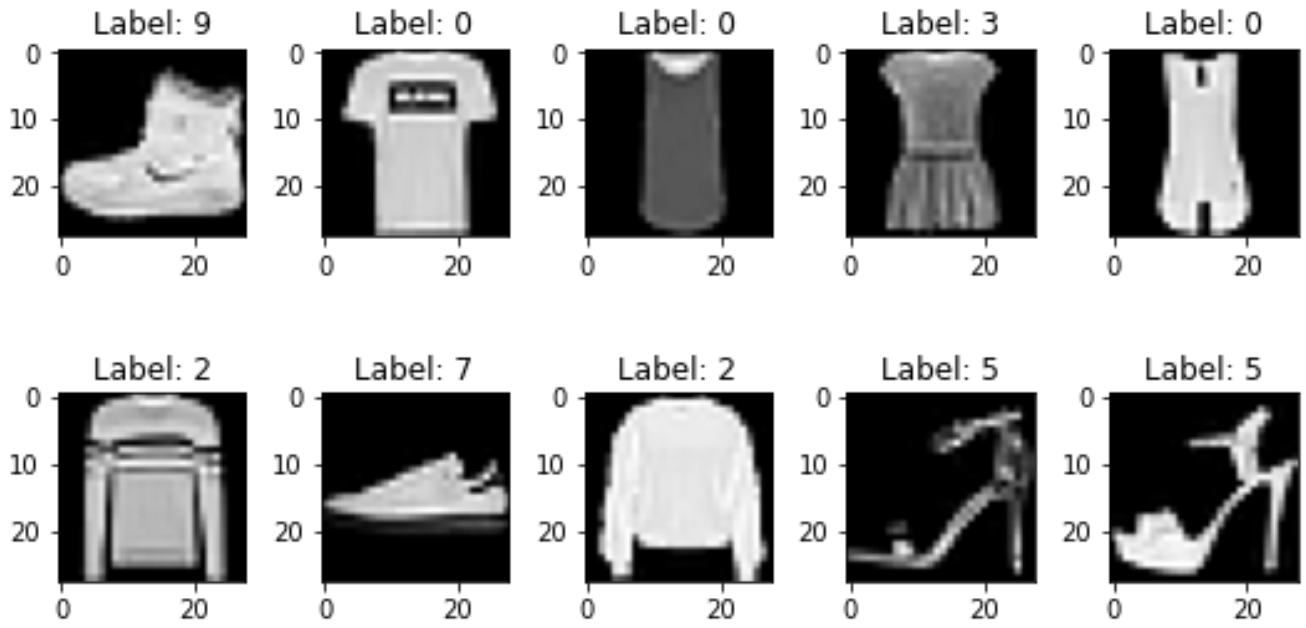
The numbers of training and test images for each class are shown in the following plots:





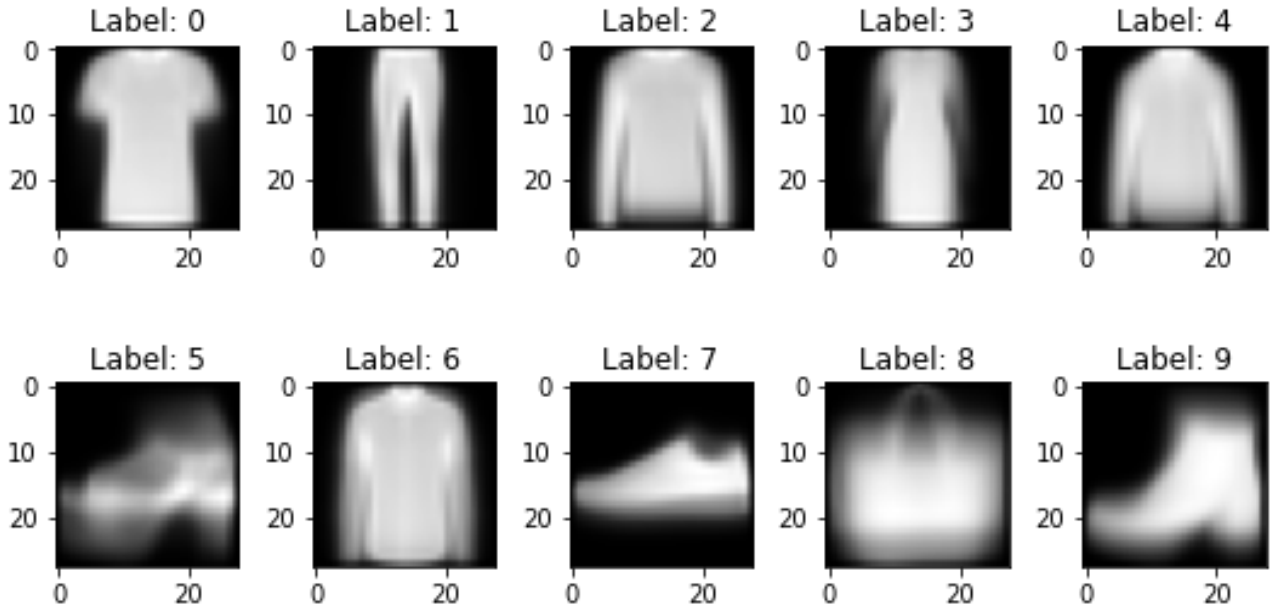
Both the training and test datasets are well-balanced.

We can visualize and look at the first 10 examples in the dataset using implot.



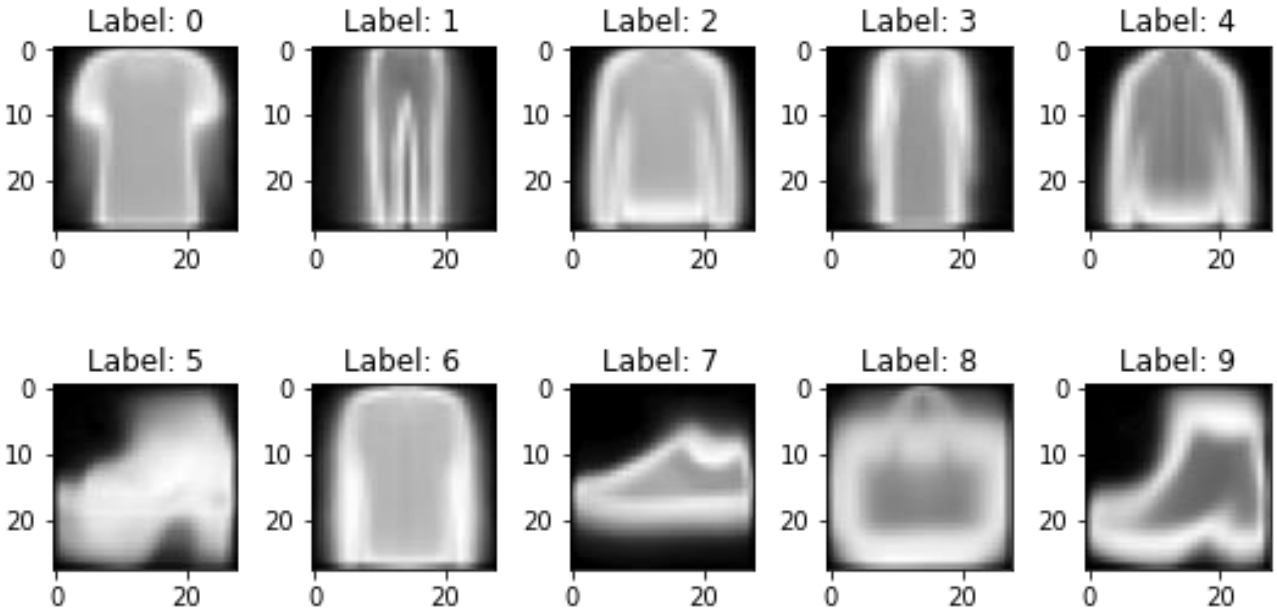
Since, the observations are images, we're displaying the mean and standard deviation of objects in each class using 28 x 28 images as well

<u>Mean</u>



The mean of the classes helps us visualize the average position of the pixels. Basically, all the images except Label 5 are mostly well-centered. Some images like Label 3 and Label 8 are (on an average) less bright than the others.

Standard Deviation



The standard deviation helps in visualizing the spread of the pixels in each class. Most of the images don't have a huge variance except Label 5, Label 6 and Label 8.

Architecture Structure of CNN

- How many input neurons are there? - 6

- What activation function will you choose?- **ReLu and SoftMax(output layer)**

- What is the number of hidden layers? (1, 2 or 3?)- **3**

- What is the kernel size, number of filters, strides, paddings, etc.? - Filters=6, kernel_size=5, strides=1, activation='ReLu', kernel_initializer='he_normal', input_shape=(28,28,1), padding='same'

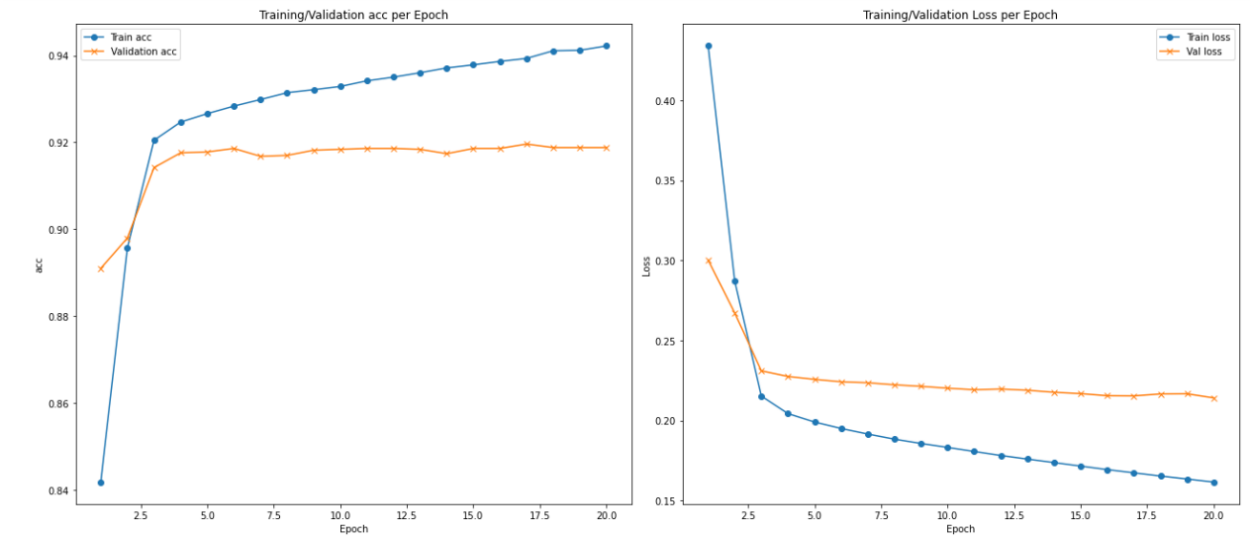
- What activation function is used for the hidden and output layer?- **ReLu and SoftMax**

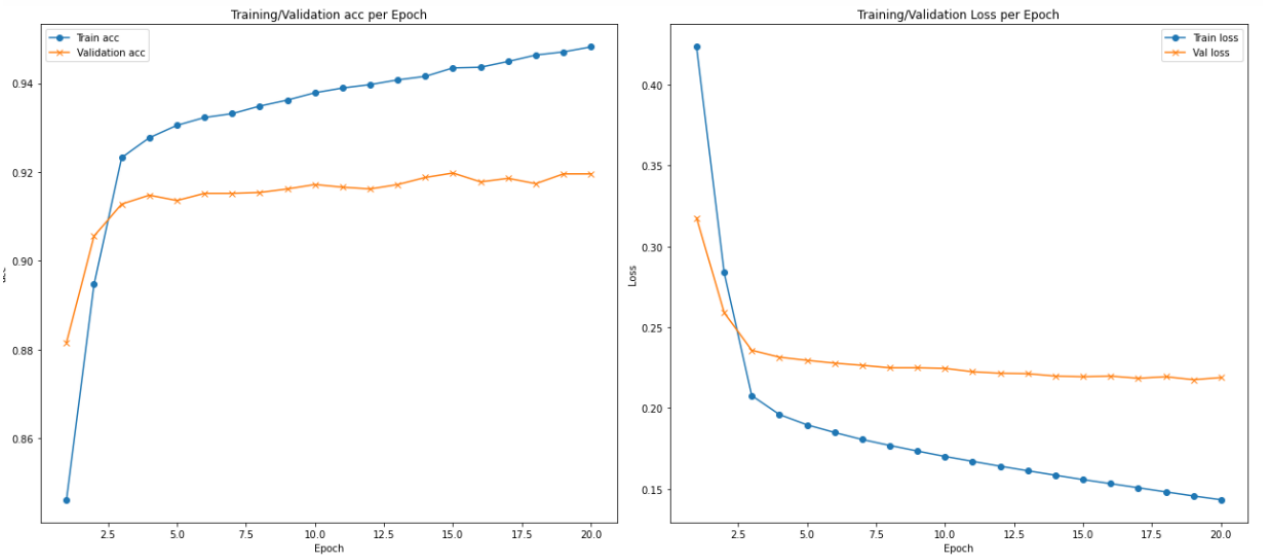
How the improvement told work on CNN architectures

I used early stopping which gave maximum accuracy.

91.6899

Graph for Training and Test - Accuracy and Loss





Part IV

Hyperparameter Tuning

1. Kernel Size Tuning

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Optimizer	Adam	90.99	Adam	91.39	Adam	91.53
Activation Function	ReLu		ReLu		ReLu	
Initializer	he_normal		he_normal		he_normal	
Kernel Size	3,3		5,5		7,7	
Padding	same		same		same	

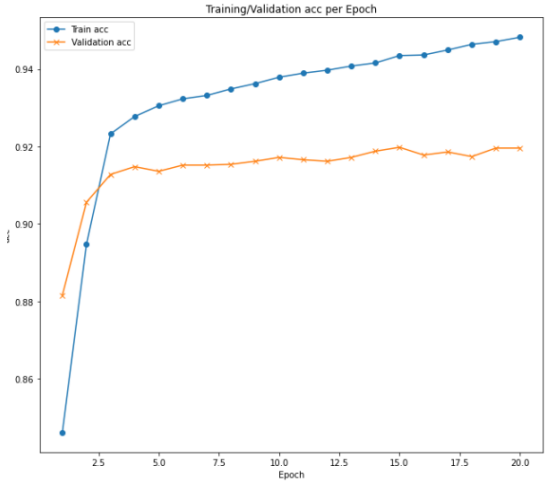
2. Kernel Initializer Tuning

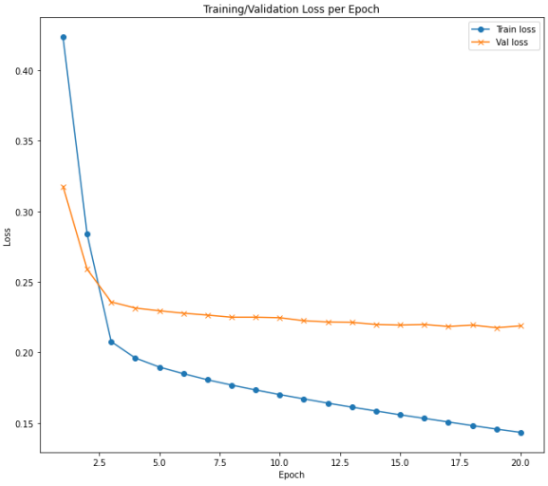
	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Optimizer	Adam	90.03	Adam	90.57	Adam	90.329
Activation Function	ReLu		ReLu		ReLu	
Initializer	random_normal		GlorotNormal		HeUniform	
Kernel Size	3,3		3,3		3,3	
Padding	same		same		same	

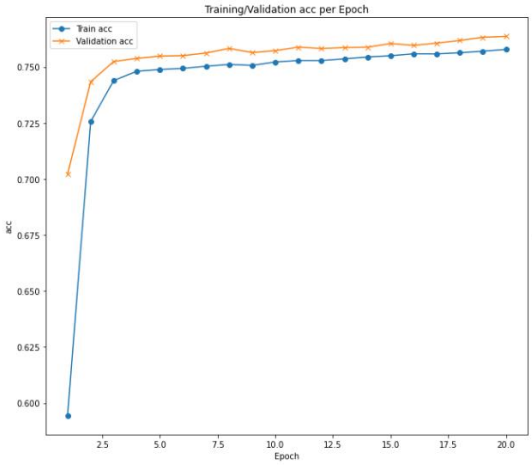
3. Optimizer Tuning

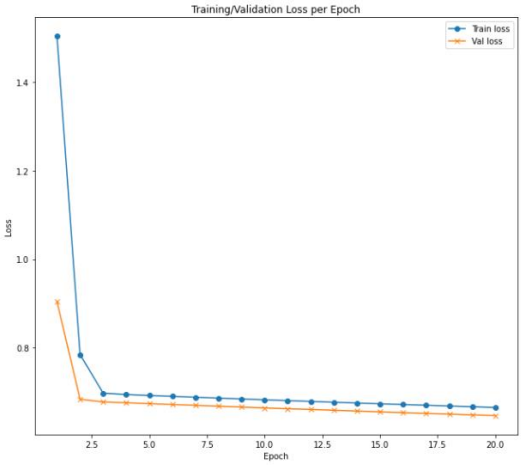
	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Optimizer	Adamax	89.179	SGD	75.309	Adagrad	78.329
Activation Function	ReLu		ReLu		ReLu	
Initializer	he_normal		he_normal		he_normal	
Kernel Size	3,3		3,3		3,3	
Padding	same		same		same	

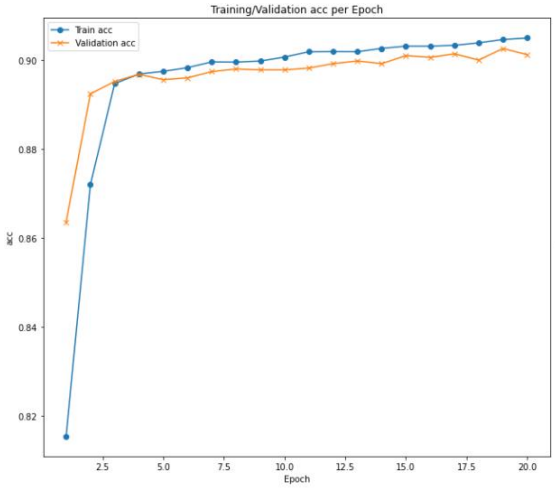
Graphs for Hyperparameter Tuning

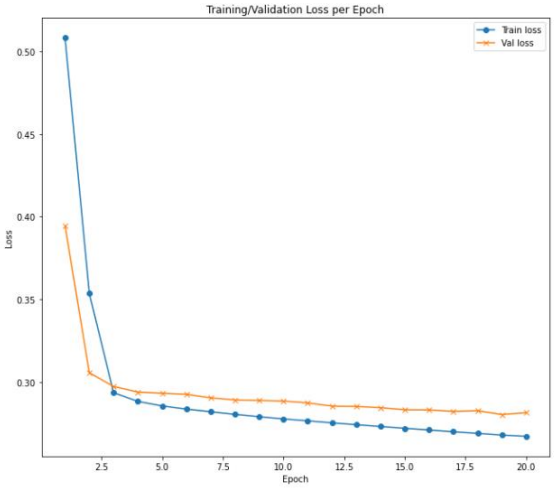


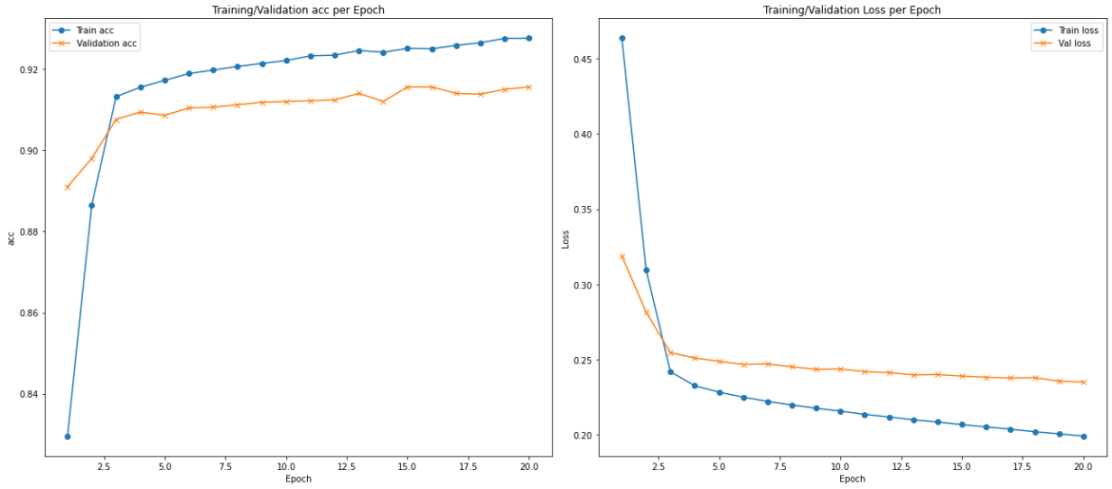




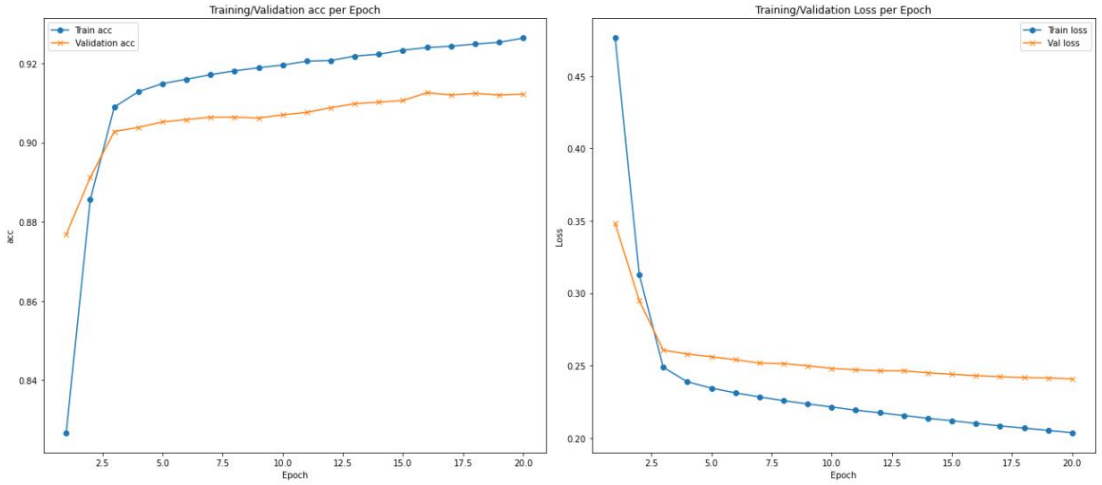


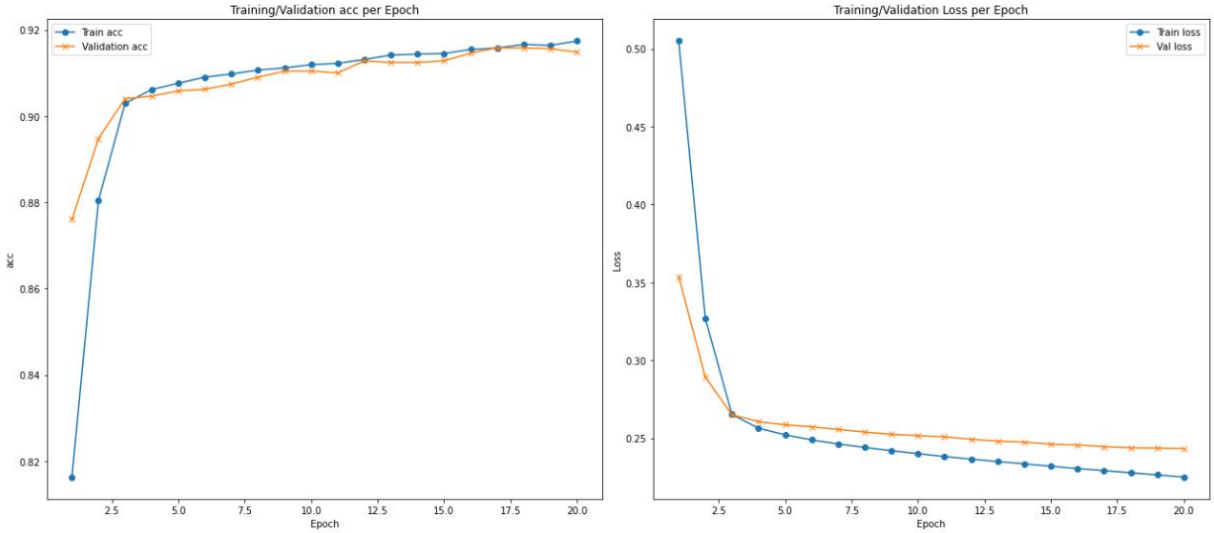


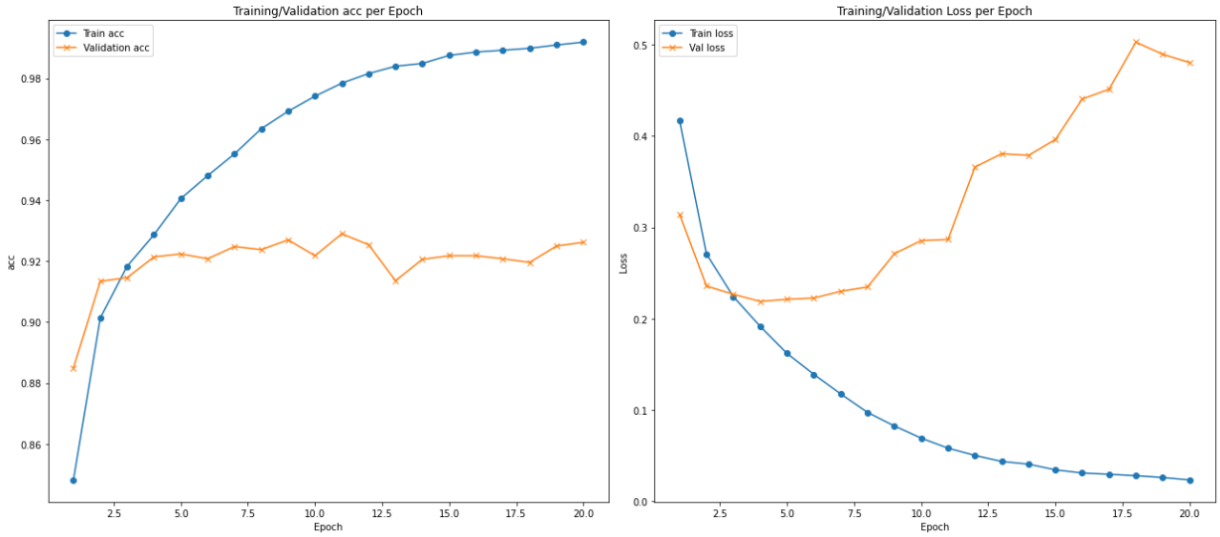



Training/Validation acc per Epoch
Train acc
Validation acc
acc
Epoch
Training/Validation Loss per Epoch
Train loss
Val loss
Loss
Epoch

```
plt.show()
```


Training/Validation acc per Epoch
Train acc
Validation acc
acc
Epoch
Training/Validation Loss per Epoch
Train loss
Val loss
Loss
Epoch


Training/Validation acc per Epoch
Train acc
Validation acc
acc
Epoch
Training/Validation Loss per Epoch
Train loss
Val loss
Loss
Epoch



Analysis and Reasoning for trying the above CNN setups

Kernel Size = Specifies the height and width of the 2D convolution window, so we play around with the size because it increases the parameters size and the run time of the model and sometimes might increase the accuracy.

Kernel Initializers = Regularize function applied to the kernel weights matrix, these are basically math functions which are applied to the matrix, so we have to try out which ones give out maximum accuracy. In a big dataset like ours either the functions won't run, or they will give about same accuracy. The three different initializers we tried gave around same accuracy.

Optimizer Tuning = We have mostly used Adam as our primary optimizer, Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first order and second-order moments. SGD gave the least accuracy. We used Adagrad which is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. Even Adagrad gave less accuracy. And at last, **Adamax** is sometimes superior to Adam, especially in models with embeddings.

Data Augmentation methods used and their impact on accuracy

Random Flip, Random Rotation, Resizing, Rescaling were the data augmentation techniques I used but data augmentation techniques are only used when the data is less, or the model is overfitting. In our model neither was the case so data augmentation cannot increase the accuracy of the model. So, the model gave an accuracy of 69.88%.

Bonus

Explanation written in python file.

Contributions

	Abhishek Kumar	Chinmay Katpatal
Part I	60%	40%
Part II	40%	60%
Part III	40%	60%
Part IV	40%	60%
Bonus	70%	30%
Total	50%	50%