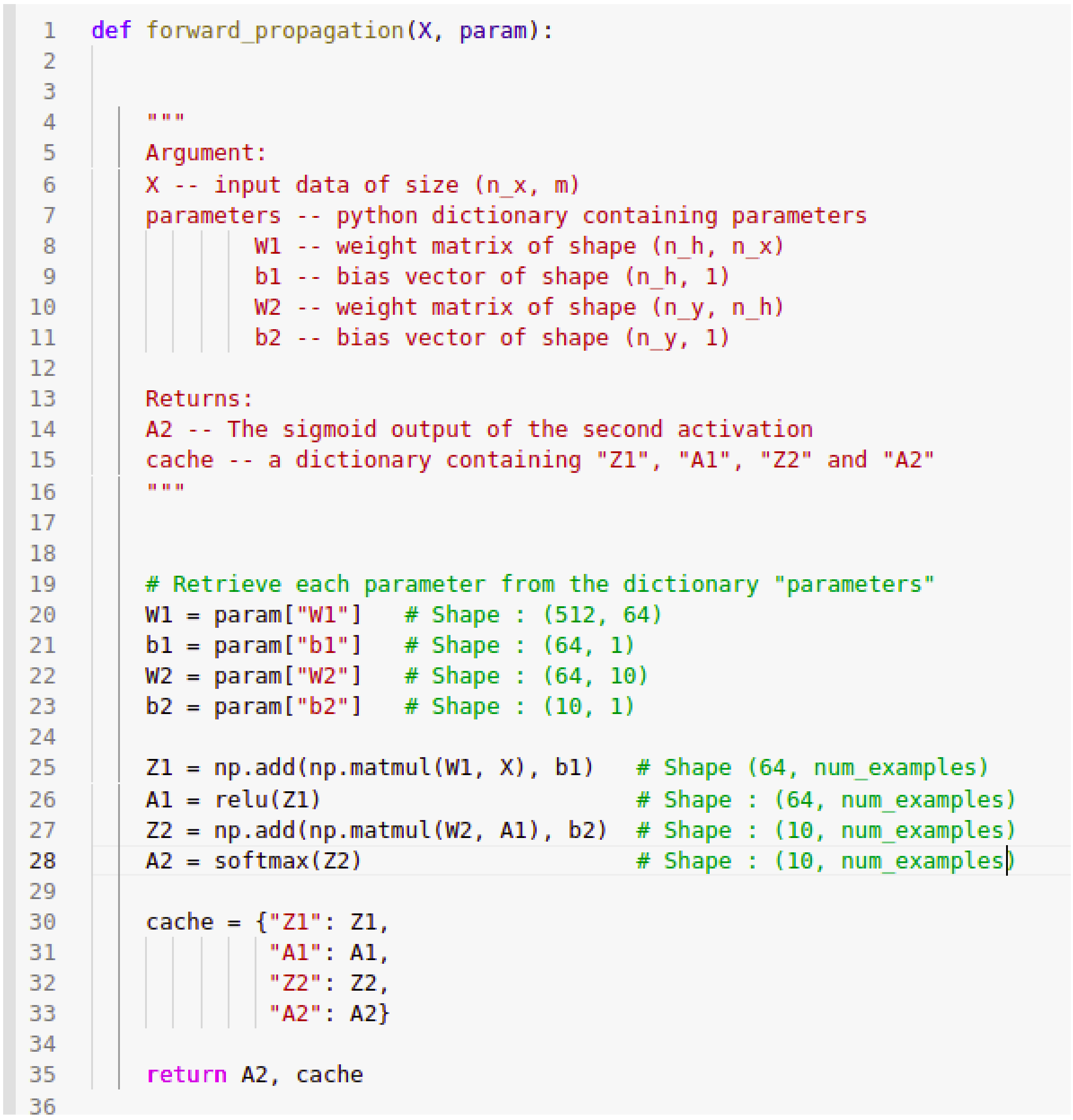
# Name : Subhajyoti Saha
# Roll no : 21111269
# 1st Year, PhD
# Deep Learning

- **Clear Instruction for running the model :**
  - I have submitted a .ipynb notebook, of which the first section i.e. the first few code cells have clearly written codes for checking the models, which read parameters values, extracted test images's feature values etc from local disk and run the inference and clearly side by side the classification report of the new runned model as well as the report which was earlier evaluated and which I shall be adding in the report. It is required to keep the below files in the same folder from where the notebook will be running. The files are as follows : 'aug_classification_report.npy', 'aug_model_params.pkl', ' class_labels.npy', 'unaug_model_params.pkl', 'unaugmented_classification_report.npy', 'x_test.npy', 'y_test.npy'.
- **Architecture of the model, epoch and learning rate :**

- **Forward Proagation Equation :**



```python
def forward_propagation(X, param):


    """
    Argument:
    X -- input data of size (n_x, m)
    parameters -- python dictionary containing parameters
                    W1 -- weight matrix of shape (n_h, n_x)
                    b1 -- bias vector of shape (n_h, 1)
                    W2 -- weight matrix of shape (n_y, n_h)
                    b2 -- bias vector of shape (n_y, 1)

    Returns:
    A2 -- The sigmoid output of the second activation
    cache -- a dictionary containing "Z1", "A1", "Z2" and "A2"
    """


    # Retrieve each parameter from the dictionary "parameters"
    W1 = param["W1"]   # Shape : (512, 64)
    b1 = param["b1"]   # Shape : (64, 1)
    W2 = param["W2"]   # Shape : (64, 10)
    b2 = param["b2"]   # Shape : (10, 1)

    Z1 = np.add(np.matmul(W1, X), b1)   # Shape (64, num_examples)
    A1 = relu(Z1)                       # Shape : (64, num_examples)
    Z2 = np.add(np.matmul(W2, A1), b2)  # Shape : (10, num_examples)
    A2 = softmax(Z2)                    # Shape : (10, num_examples)

    cache = {"Z1": Z1,
             "A1": A1,
             "Z2": Z2,
             "A2": A2}

    return A2, cache

```

As the forward propagation equations are nicely written in the code, I have taken the screen shot from there and have attached it. It would work as a reference while deriving the back propagation equations.

## ❖ Derivation of Back Propagation Equations :

Let's assume the following forward prop equation :

$$Z2 = W2.X_j + b2$$

Let's assume Z2 denote the output of the second layer of the neural net without applying the softmax activation function for a single example.

Let’s consider A2 to be the softmax activation of the second layer corresponding to Z2 defined as follows:

$$A2 = \frac{e^{Z2}}{\sum_{j=1}^{10(i.e. num_classes)} e^{Z2_j}}$$

Now the corresponding cross-entropy loss can be defined as follows for a single example :

$$J = \sum_{j=1}^{10} y_j * \log_e A2_j$$

Now derivative of A2 w.r.t Z2 will be a 2D matrix of size 10*10, as we know that derivative of vector w.r.t a vector is always a 2D matrix, which denotes how one vector will behave while changing any position of another vector. Thus let’s assume, we denote i to index the row number of the 2D derivative matrix and j to denote the column number of the derivative matrix. And the(i,j)-th entry of the matrix will denote how much i-th component of A2 will change by tweaking the j-th component of Z2.

So we have two cases for calculating the matrix :

1. Case 1 : i = j :

$$\frac{\partial}{\partial Z2_i} A2_j$$

$$= \frac{\partial}{\partial Z2_j} \frac{\exp^{Z2_j}}{\sum_{k=1}^{10} \exp^{Z2_k}}$$

$$= \frac{\exp^{z2_j} (\sum_{k=1}^{10} \exp^{Z2_k} - \exp^{Z2_j})}{(\sum_{k=1}^{10} \exp^{Z2_k})^2}$$

$$= A2_i(1 - A2_j)$$

2. Case 2 : i not equal to j:

$$\frac{\partial}{\partial Z2_i} A2_j$$

$$= \frac{\partial}{\partial Z2_j} \frac{\exp^{Z2_i}}{\sum_{k=1}^{10} \exp^{Z2_k}}$$

$$= \frac{0 - \exp^{Z2_j} \exp^{Z2_i}}{\left(\sum_{k=1}^{10} \exp^{Z2_k}\right)^2}$$

$$= -A2_i A2_j$$

Thus, derivative of the cost function w.r.t Z2 will be as following via applying chain rule of derivative:

$$\frac{\partial L}{\partial Z2_i}$$

$$= -\sum_{k=1}^{10} y_k \frac{\partial \log_e A2_k}{\partial A2_k} \frac{\partial A2_k}{\partial Z2_i}$$

$$= -\sum_{k=1}^{10} \frac{1}{A2_k} \frac{\partial A2_k}{\partial Z2_j}$$

Thus,

$$\frac{\partial L}{\partial Z2_i}$$

$$= -y_i(1 - A2_i) + \sum_{k \neq i}^{y} {}_k A2_i$$

$$= A2_i(y_i + \sum_{k \neq i}^{y} {}_k) - y_i$$

$$= A2_i - y_i$$

Thus we can see that the derivative of the loss function w.r.t the i-th component of Z2 would be a scalar defined as $A2_i - y_i$

Thus the derivative of a scalar i.e. Loss function L w.r.t a vector Z2 will be a vector of the same size whose each position will denote how the scalar will behave in change of the corresponding position of vector. Thus ,

$$\frac{\partial L}{\partial Z2} = A2 - Z2$$

Thus in the above we show how to take the derivative of the cost function w.r.t the Z2.

Now we shall be taking the derivative of the cost function, i.e. a scalar variable wrt all the trainable parameters of our model. Note that the derivative of scalar w.r.t any vector would be a vector of the same size whose each of the corresponding positions would denote how the scalar will behave in response to the tweak of the corresponding position of the vector.

$$\frac{\partial L}{\partial W2}$$
$$Z2 = W2.A1 + b2$$

$$= \frac{\partial L}{\partial Z2}\frac{Z2}{W2}$$

$$= \frac{1}{samples}\sum_{k=1}^{samples}\frac{\partial L}{\partial Z2_i}A1_i^T$$ Here, samples denotes num of examples.

And derivative of the cost function w.r.t the vector b2 would be vector derive as follows:

$$\frac{\partial L}{\partial b2}$$

$$= \frac{\partial L}{\partial Z2}\frac{Z2}{b2}$$

$$= \frac{1}{samples}\sum_{k=1}^{samples}\frac{\partial L}{\partial Z2_i}$$

Now we need to find out the derivative of the cost function w.r.t A1 i.e.

$$\frac{\partial L}{\partial A1}$$

$$= \frac{\partial L}{\partial Z2}\frac{\partial Z2}{\partial A1}$$

$$= W2^T.\frac{\partial L}{\partial Z2}$$

We also know that A1 = relu(Z1) = np.maximum(0, Z1)

Therefore, $\frac{\partial A1}{\partial Z1}$ = 1 if Z1>0, otherwise 0.

Thus, $\frac{\partial L}{\partial Z1} = \frac{\partial L}{\partial A1} * \frac{\partial A1}{\partial Z1}$ (where, * denotes elmentwise multiplication.)

In the similar way we can write the following equations as follows :

$$\frac{\partial L}{\partial W1}$$

$$= \frac{\partial L}{\partial Z1}\frac{\partial Z1}{\partial W1}$$

$$= \frac{1}{samples} \sum_{k=1}^{samples} \frac{\partial L}{\partial Z1} X^T$$

$$\frac{\partial L}{\partial b1}$$

$$= \frac{\partial L}{\partial Z1} \frac{Z1}{b1}$$

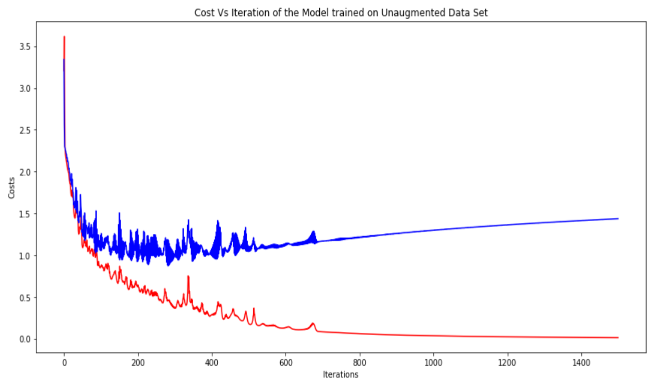
$$= \frac{1}{samples} \sum_{k=1}^{samples} \frac{\partial L}{\partial Z1_i}$$

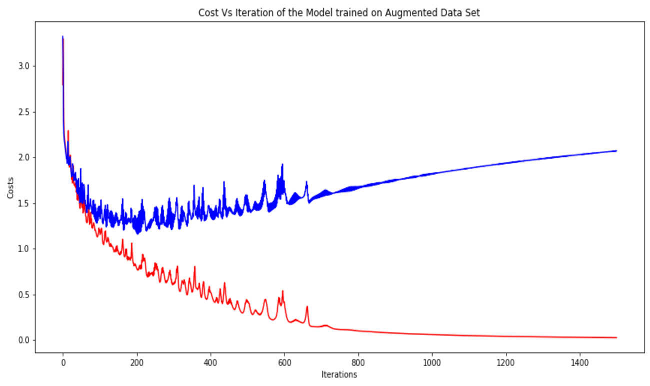
- **Epoch and Learning Rate :**

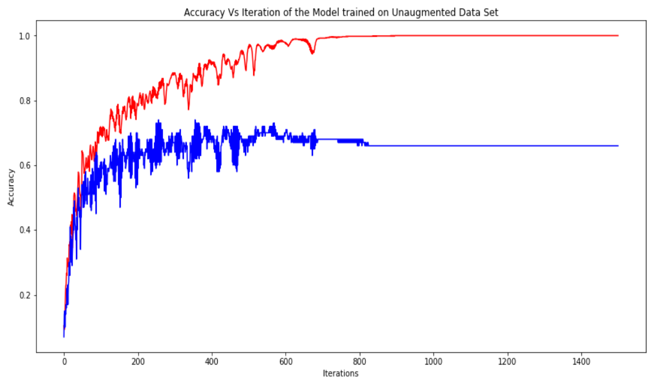
  In our case we saw that our model is nicely converging within 3000 epochs, and we were using the learning rate of 0.01 for all the data set.

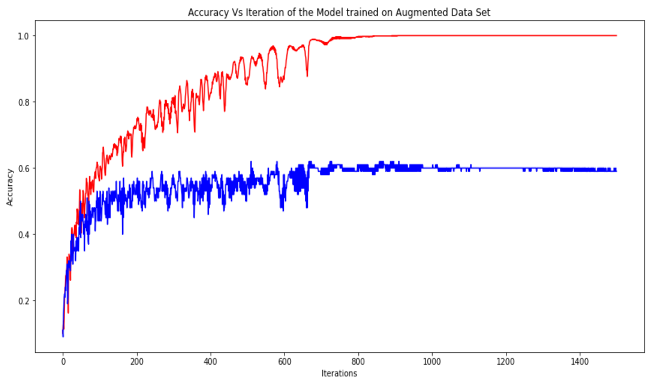
- **Comparison of the performance of the model trained on augmented and unaugmented data set.**

Below we are attaching the graph of the performance of the model on augmented and unaugmented data set.

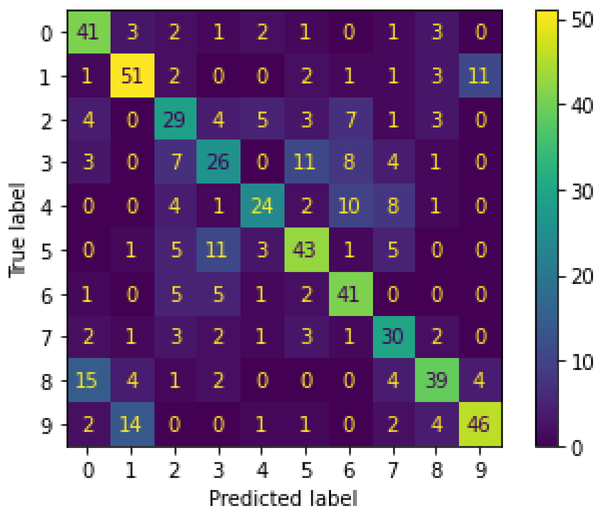






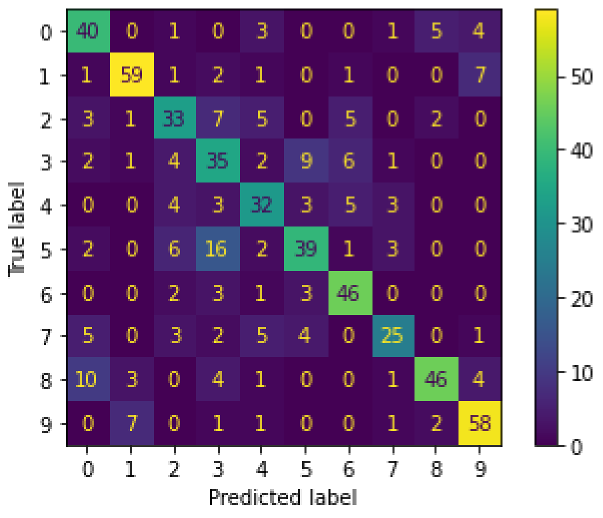


As we can see the above plot represents the loss and accuracy of our model trained on unaugmented and augmented data sets.Here, the red denotes the training set performance and the blue line denote the development set performance. As the amount of augmented data set used was almost similar to the unaugmented data set, due to the limitation of computation, we can not see the expected behaviour of augmented model's performance to be better than the unaugmented model's performance. However, we can see that the unaugmented model has reached the optimum level of accuracy at an earlier epoch than the augmented data set, due to the fact that augmentation of the data set has made the classification job more difficult than the unaugmented one, with same number of data set.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 41 | 3 | 2 | 1 | 2 | 1 | 0 | 1 | 3 | 0 |
| 1 | 1 | 51 | 2 | 0 | 0 | 2 | 1 | 1 | 3 | 11 |
| 2 | 4 | 0 | 29 | 4 | 5 | 3 | 7 | 1 | 3 | 0 |
| 3 | 3 | 0 | 7 | 26 | 0 | 11 | 8 | 4 | 1 | 0 |
| 4 | 0 | 0 | 4 | 1 | 24 | 2 | 10 | 8 | 1 | 0 |
| 5 | 0 | 1 | 5 | 11 | 3 | 43 | 1 | 5 | 0 | 0 |
| 6 | 1 | 0 | 5 | 5 | 1 | 2 | 41 | 0 | 0 | 0 |
| 7 | 2 | 1 | 3 | 2 | 1 | 3 | 1 | 30 | 2 | 0 |
| 8 | 15 | 4 | 1 | 2 | 0 | 0 | 0 | 4 | 39 | 4 |
| 9 | 2 | 14 | 0 | 0 | 1 | 1 | 0 | 2 | 4 | 46 |

The above is the confusion matrix on the test data set on the augmented model.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 40 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 5 | 4 |
| 1 | 1 | 59 | 1 | 2 | 1 | 0 | 1 | 0 | 0 | 7 |
| 2 | 3 | 1 | 33 | 7 | 5 | 0 | 5 | 0 | 2 | 0 |
| 3 | 2 | 1 | 4 | 35 | 2 | 9 | 6 | 1 | 0 | 0 |
| 4 | 0 | 0 | 4 | 3 | 32 | 3 | 5 | 3 | 0 | 0 |
| 5 | 2 | 0 | 6 | 16 | 2 | 39 | 1 | 3 | 0 | 0 |
| 6 | 0 | 0 | 2 | 3 | 1 | 3 | 46 | 0 | 0 | 0 |
| 7 | 5 | 0 | 3 | 2 | 5 | 4 | 0 | 25 | 0 | 1 |
| 8 | 10 | 3 | 0 | 4 | 1 | 0 | 0 | 1 | 46 | 4 |
| 9 | 0 | 7 | 0 | 1 | 1 | 0 | 0 | 1 | 2 | 58 |

The above is the confusion matrix on the test data set on the unaugmented data set

By observing the diagonal terms, we can see that the model trained on the unaugmented data set (below matrix), is performing slight better than the augmentyed one. However, one nice observation is that both the model are performing almost same sort of performance in the same class of the data set, i.e. the class which is difficult to be predicted, both the model are performing less accurately on those, whereas the simpler are easily classified by both the model. This denotes the consistency of our model on the prediction jo0b.

Next the classification report of the augmented model is shown below :

```
The classification report on Augmeneted Model is as follows on
Unaugmeneted Original Test Set :


              precision    recall  f1-score   support

  Air Planes       0.59      0.76      0.67        54
 Auto-mobile       0.69      0.71      0.70        72
        bird       0.50      0.52      0.51        56
         cat       0.50      0.43      0.46        60
        deer       0.65      0.48      0.55        50
         dog       0.63      0.62      0.63        69
        frog       0.59      0.75      0.66        55
       horse       0.54      0.67      0.59        45
        ship       0.70      0.57      0.62        69
       truck       0.75      0.66      0.70        70

    accuracy                           0.62       600
   macro avg       0.61      0.62      0.61       600
weighted avg       0.62      0.62      0.61       600
```

```
The classification report on Unaugmeneted Model is as follows on
Unaugmeneted Original Test Set :


              precision    recall  f1-score   support

  Air Planes       0.63      0.74      0.68        54
 Auto-mobile       0.83      0.82      0.83        72
        bird       0.61      0.59      0.60        56
         cat       0.48      0.58      0.53        60
        deer       0.60      0.64      0.62        50
         dog       0.67      0.57      0.61        69
        frog       0.72      0.84      0.77        55
       horse       0.71      0.56      0.63        45
        ship       0.84      0.67      0.74        69
       truck       0.78      0.83      0.81        70
```

```
    accuracy                           0.69       600
   macro avg       0.69      0.68      0.68       600
weighted avg       0.70      0.69      0.69       600
```

The classification report also says the same thing that our unaugmented model's performance is slightly better.