README

File Structure and description:

Feed_Forward_NN.py	*Contains Feed Forward Neural Network class which has both forward and backpropagation as its member functions.*
activation_fns_and_grads.py	*Contains activations functions and their respective gradients.* *Implemented:* 1. *sigmoid* 2. *relu* 3. *tanh and* 4. *softmax.*
loss_fns_with_regularization.py	*Contains loss functions with regularization compatibility.* *Implemented:* 1. *cross entropy softmax* 2. *mean squared error and* 3. *cross entropy sigmoid (for 2-class classification and single output node)*
optimizers_with_regularization.py	*Contains optimizers with regularization compatibility. Every optimizer can run for any batch size.* *Implemented:* 1. *SGD (stochastic gradient descent also runs for mini – batches of any size)* 2. *Momentum gradient descent* 3. *Nesterov gradient descent* 4. *RMSProp* 5. *Adam* 6. *Nadam*
utils.py	*Contains various helper functions:* 1. *Class to one hot vector conversion* 2. *One hot to class conversion* 3. *Train - Val split function (from Sklearn)* 4. *Function to calculate accuracy and loss* 5. *Function to create mini batches* 6. *Function for loss grad with respect to last layer.*
train_fn.py	*Contains train function.*

Important Functions and their parameters:

1. *FeedForward_NN class in Feed_Forward_NN.py :*

```
FeedForward_NN( input_features,  # Number of input features.
        output_nodes,  # Number of output nodes (generally equal to number of classes)
        hidden_layers_dims, # List of hidden layer nodes for each layer in same order eg. [32,16]
```

```
        act_fn, # Activation function for all layers except last layer
        dropout, # Probability with which we want a node to be on
        initialization # Initialization type eg. xavier or random
)
```

---

2. *train function in train_fn.py:*

```
train(model,
    X_train,
    Y_train,
    X_val,
    Y_val,
    output_activation, # Last layer activation eg. Softmax for classification
    output_loss_fn, # Loss function eg. cross_entropy
    epochs, # Number of epochs
    optimizer_name, # Name of optimizer eg. adam
    l2_lambd, # L2 regularization constant (lambda)
    learning_rate, # Learning rate
    lr_schedule, # Learning rate schedule (this value is multiplied to learning rate ever 10 epochs)
    batch_size, # Batch size
    print_cost, # To print cost and accuracy every epoch
    val # To calculate and print val cost and accuracy
)
```

String parameters (exact names to be used) in above functions are:

Optimizers: *'sgd', 'momentum', 'rmsprop', 'nesterov', 'adam', 'nadam'*

Output activations: *'softmax', 'linear'*

Activation functions: *'relu', 'tanh', 'sigmoid'*

Initialization: *'xavier', 'random'*

To run a custom model:

- Open the **train_model.ipynb** file and change the parameters as required. (Already all required functions are imported)

To reproduce the results:

- To load single image per class for Fashion MNINST in wandb: Run **load_classes.ipynb**
- For Sweep1: Run **Sweep1.ipynb**
- For Sweep2: Run **Sweep2.ipynb**
- For Best model for Fashion MNIST: Run **Best_model_FMNIST.ipynb**
- For MNIST best 3 hyperparameters: Run **MNIST.ipynb**