

In [13]:

```
import os
import time
import shelve
import random
import numpy as np
import pandas as pd
import tensorflow as tf
from pandas import DataFrame
import matplotlib.pyplot as plt
```

In [14]:

```
def load_data(name):  
    if name == 'mnist':  
        (X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()  
    elif name == 'fashion_mnist':  
        (X_train, y_train), (X_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()  
    else:  
        print('Only mnist or fashion_mnist.')  
        return False  
  
    imageSize = X_train.shape[1]*X_train.shape[2]  
    numClasses = np.max(y_train)+1  
  
    X_train = np.reshape(X_train.astype(float)/255.0, (-1, 784))  
    X_test = np.reshape(X_test.astype(float)/255.0, (-1, 784))  
  
    y_train = tf.keras.utils.to_categorical(y_train, num_classes=numClasses)  
    y_test = tf.keras.utils.to_categorical(y_test, num_classes=numClasses)  
  
    X_val = X_train[-10000:]  
    y_val = y_train[-10000:]  
    X_train = X_train[:-10000]  
    y_train = y_train[:-10000]  
  
    print('Data Split: ')  
    print(f'X_train: {X_train.shape}, y_train: {y_train.shape}')  
    print(f'X_test : {X_test.shape }, y_test : {y_test.shape }')  
    print(f'X_val  : {X_val.shape }, y_val  : {y_val.shape }')  
  
    data = {}  
    data['X_train'] = X_train  
    data['y_train'] = y_train  
    data['X_val'] = X_val  
    data['y_val'] = y_val  
    data['X_test'] = X_test  
    data['y_test'] = y_test  
  
    data['imageSize'] = imageSize  
  
    return data
```

In [27]:

```

class MLP(object):

    def __init__(self, name, size_input, size_hidden, size_output, learning_rate=
0.01, optimizer='SGD', weight_coeff=1,\
                Reg=None, RegC=0, training=None, validation=None, accuracy=0, dev
ice=None):

        self.name            = name
        self.size_input      = size_input
        self.size_hidden     = size_hidden
        self.size_output     = size_output
        self.learning_rate   = learning_rate
        self.optimizer       = optimizer
        self.Reg             = Reg
        self.RegC            = RegC
        self.training        = training
        self.validation       = validation
        self.accuracy        = accuracy
        self.device          = device
        self.weight_coeff    = weight_coeff

        self.W1 = self.initWeights(self.size_input, self.size_hidden[0], self.weig
ht_coeff)
        self.b1 = self.initWeights(1, self.size_hidden[0], self.weight_coeff)

        self.W2 = self.initWeights(self.size_hidden[0], self.size_hidden[1], self.
weight_coeff)
        self.b2 = self.initWeights(1, self.size_hidden[1], self.weight_coeff)

        self.W3 = self.initWeights(self.size_hidden[1], self.size_hidden[2], self.
weight_coeff)
        self.b3 = self.initWeights(1, self.size_hidden[2], self.weight_coeff)

        self.W4 = self.initWeights(self.size_hidden[2], self.size_output, self.wei
ght_coeff)
        self.b4 = self.initWeights(1, self.size_output, self.weight_coeff)

        self.variables = [self.W1, self.b1, self.W2, self.b2, self.W3, self.b3, sel
f.W4, self.b4]

    def initWeights(self, rows, columns, multFactor=1):
        return tf.Variable(multFactor*tf.random.normal([rows, columns]))

    def forward(self, X):

        if self.device is not None:
            with tf.device('gpu:0' if self.device=='gpu' else 'cpu'):
                self.y = self.compute_output(X)
        else:
            self.y = self.compute_output(X)

        return self.y

```

```

def getRegLoss(self, X_train):
    if self.Reg=='L2':
        return (self.RegC/X_train.shape[0])*(tf.reduce_sum(tf.math.square(self
.W1)) +
                                                    tf.reduce_sum(tf.math.square(self
.W2)) +
                                                    tf.reduce_sum(tf.math.square(self
.W3)) +
                                                    tf.reduce_sum(tf.math.square(self
.W4)))

    elif self.Reg=='L1':
        return (self.RegC/X_train.shape[0])*tf.abs(tf.reduce_sum(self.W1) +
                                                    tf.reduce_sum(self.W2) +
                                                    tf.reduce_sum(self.W3) +
                                                    tf.reduce_sum(self.W4))

    elif self.Reg=='L1+L2':
        L2 = (self.RegC/X_train.shape[0])*(tf.reduce_sum(tf.math.square(self
.W1)) +
                                                    tf.reduce_sum(tf.math.square(self
.W2)) +
                                                    tf.reduce_sum(tf.math.square(self
.W3)) +
                                                    tf.reduce_sum(tf.math.square(self
.W4)))

        L1 = (self.RegC/X_train.shape[0])*tf.abs(tf.reduce_sum(self.W1) +
                                                    tf.reduce_sum(self.W2) +
                                                    tf.reduce_sum(self.W3) +
                                                    tf.reduce_sum(self.W4))

        return L1+L2

    else:
        return 0

def loss(self, y_pred, y_true):
    y_true_tf = tf.cast(tf.reshape(y_true, (-1, self.size_output)), dtype=tf.f
loat32)
    y_pred_tf = tf.cast(y_pred, dtype=tf.float32)

    loss = tf.keras.losses.CategoricalCrossentropy()(y_true_tf, y_pred_tf)
    return loss

def backward(self, X_train, y_train):
    if self.optimizer=='SGD':
        optimizer = tf.keras.optimizers.SGD(learning_rate=self.learning_rate)

    elif self.optimizer=='Adam':
        optimizer = tf.keras.optimizers.Adam(learning_rate=self.learning_rate)

    elif self.optimizer=='RMSprop':

```

```

optimizer = tf.keras.optimizers.RMSprop(learning_rate=self.learning_rate)

    else:
        pass

    if self.Reg is not None and self.RegC==0:
        print('Regularization coffecient argument was 0, seeting it to default
lamda=0.01')
        self.RegC = 0.01;

    with tf.GradientTape() as tape:
        predicted = self.forward(X_train)
        current_loss = self.loss(predicted, y_train)
        current_loss += self.getRegLoss(X_train)

    grads = tape.gradient(current_loss, self.variables)
    optimizer.apply_gradients(zip(grads, self.variables))

def compute_output(self, X):

    X_tf = tf.cast(X, dtype=tf.float32)

    w1Hat = tf.matmul(X_tf, self.W1) + self.b1
    h1Hat = tf.nn.relu(w1Hat)

    w2Hat = tf.matmul(h1Hat, self.W2) + self.b2
    h2Hat = tf.nn.relu(w2Hat)

    w3Hat = tf.matmul(h2Hat, self.W3) + self.b3
    h3Hat = tf.nn.relu(w3Hat)

    w4Hat = tf.matmul(h3Hat, self.W4) + self.b4
    output = tf.nn.softmax(w4Hat)

    return output

def getAccuracy(self, predictions, outputs):
    preds = np.argmax(predictions, axis=1)
    y_true = np.argmax(outputs, axis=1)

    return (preds==y_true).mean()

```

In [28]:

```

def trainModel(model, data, NUM_EPOCHS=10, batchSize=50, seedVal=1234):

    X_train = data['X_train']
    y_train = data['y_train']
    X_val    = data['X_val']
    y_val    = data['y_val']

    training = np.zeros(shape=(NUM_EPOCHS, 3))
    validation = np.zeros(shape=(NUM_EPOCHS, 3))

    train_ds = tf.data.Dataset.from_tensor_slices((X_train, y_train)).batch(batchSize)
    val_ds    = tf.data.Dataset.from_tensor_slices((X_val, y_val)).batch(batchSize)

    print(f'\n\n***** Training model: {model.name} with optimizer: {model.optimizer} and seed: {seedVal} *****\n')
    time_start = time.time()
    for epoch in range(NUM_EPOCHS):
        train_loss = tf.zeros([1, 1], dtype=tf.float32)
        val_loss    = tf.zeros([1, 1], dtype=tf.float32)

        train_ds = tf.data.Dataset.from_tensor_slices((X_train, y_train)).shuffle(25, seed = epoch*(seedVal)).batch(batchSize)
        val_ds    = tf.data.Dataset.from_tensor_slices((X_val, y_val)).shuffle(25, seed = epoch*(seedVal)).batch(batchSize)

        for inputs, outputs in train_ds:
            train_pred = model.forward(inputs)
            train_loss = train_loss + model.loss(train_pred, outputs)
            model.backward(inputs, outputs)
            train_acc = model.getAccuracy(train_pred, outputs)

        for inputs, outputs in val_ds:
            val_pred = model.forward(inputs)
            val_loss = val_loss + model.loss(val_pred, outputs)
            val_acc = model.getAccuracy(val_pred, outputs)

        # train_loss = np.array(train_loss)
        # val_loss = np.array(val_loss)

        training[epoch] = [epoch+1, train_acc, np.sum(train_loss)/X_train.shape[0]]
        validation[epoch] = [epoch+1, val_acc, np.sum(val_loss)/X_val.shape[0]]

    print(f'# Epoch:={}/{} - train loss:={:.4f} - val loss:={:.4f}, train acc:={:.2f} - val acc:={:.2f}'.format(epoch+1, NUM_EPOCHS, np.sum(train_loss)/X_train.shape[0], np.sum(val_loss)/X_val.shape[0], train_acc, val_acc))

    time_taken = time.time()-time_start
    print(f'\nTotal time taken (in seconds): {time_taken: .2f}')
    print(f'\nFinished training model: {model.name}\n')

```

```
model.training = training
model.validation = validation

def testModel(model, data):

    X_test = data['X_test']
    y_test = data['y_test']

    preds = model.forward(X_test)

    pred = np.argmax(preds, axis=1)
    y_true= np.argmax(y_test, axis=1)

    model.accuracy = (pred==y_true).mean()*100

    print(f'***** Testing *****')
    print(f'{model.name} model accuracy = {model.accuracy:.2f}%')
    print(f'*****')

def plotAccuracyAndLoss(model):

    training = model.training
    validation = model.validation
    fig, (ax1, ax2) = plt.subplots(1, 2)
    training[:, -1] = training[:, -1]/np.linalg.norm(training[:, -1])
    ax1.plot(training[:,0], training[:,1], 'g')
    ax1.plot(training[:,0], training[:,2], 'b')
    ax1.set_title('Training')
    ax1.legend(["Accuracy", "Loss"])

    validation[:, -1] = validation[:, -1]/np.linalg.norm(validation[:, -1])
    ax2.plot(validation[:,0], validation[:,1], 'g')
    ax2.plot(validation[:,0], validation[:,2], 'b')
    ax2.set_title('Validation')
    ax2.legend(["Accuracy", "Loss"])
    plt.show()
```

In [31]:

```
def main():  
  
    for j in range(1):  
        j=1  
        if j==0:  
            data = load_data('mnist')  
            size_hidden = [128, 128, 128]  
            learning_rate = 5e-4  
            weight_coeff = 1e-2  
  
        if j==1:  
            print('fashion_mnist is running')  
            data = load_data('fashion_mnist')  
            size_hidden = [1024, 512, 256]  
            learning_rate = 5e-4  
            weight_coeff = 1e-3  
  
    for k in range(1):  
        if k==0:  
            opt = 'RMSprop'  
        elif k==1:  
            opt = 'Adam'  
        elif k==2:  
            opt = 'SGD'  
        else:  
            pass  
  
        imageSize = data['imageSize']  
  
        size_input = imageSize  
        size_output = 10  
  
        allModels = {}  
        allModels['mlp_on_gpu_default'] = {}  
        allModels['mlp_on_gpu_RegL1'] = {}  
        allModels['mlp_on_gpu_RegL2'] = {}  
  
        for model_name in allModels:  
            model = allModels[model_name]  
  
            cnt = -1  
  
            numEpochs = 10  
            batchSize = 50  
            numTrials = 10  
  
            seeds = random.sample(range(1000, 9999), numTrials)  
  
            # loss = np.zeros(shape=(numEpochs, 1))  
            accuracy = np.zeros(shape=(numTrials, 1))  
  
            for i in seeds:  
                cnt += 1
```



```

        np.random.seed(i)
        tf.random.set_seed(i)

        print(f'Count: {cnt}, j=: {j}')
        if model_name == 'mlp_on_gpu_default':
            model['name'] = MLP('mlp_on_gpu_default', size_input, size_
_hidden, size_output, learning_rate, opt, weight_coeff,\
                                device='gpu')

            elif model_name == 'mlp_on_gpu_RegL1':
                model['name'] = MLP('mlp_on_gpu_RegL1', size_input, size_h
idden, size_output, learning_rate, opt, weight_coeff,\
                                    'L1', 1e-2, device='gpu')

            elif model_name == 'mlp_on_gpu_RegL2':
                model['name'] = MLP('mlp_on_gpu_RegL2', size_input, size_h
idden, size_output, learning_rate, opt, weight_coeff,\
                                    'L2', 1e-2, device='gpu')

            else:
                pass

        trainModel(model['name'], data, numEpochs, batchSize, i)
        testModel(model['name'], data)

        accuracy[cnt] = model['name'].accuracy

        plotAccuracyAndLoss(model['name'])

        allModels[model_name][i] = model['name']
        allModels[model_name]['Accuracy'] = [np.mean(accuracy), np.var
(accuracy)]

    if j==0:
        mnist = allModels
    elif j==1:
        fashion_mnist = allModels
    else:
        pass

return mnist, fashion_mnist

```

In []:

```
if __name__ == "__main__":  
    mnist, fashion_mnist = main()
```

fashion_mnist is running

Data Split:

X_train: (50000, 784), y_train: (50000, 10)

X_test : (10000, 784), y_test : (10000, 10)

X_val : (10000, 784), y_val : (10000, 10)

Count: 0, j=: 1

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 8349 *****

Epoch:=1/10 - train loss:=0.0250 - val loss:=0.0253, train acc:=0.6
6 - val acc:=0.62
Epoch:=2/10 - train loss:=0.0320 - val loss:=0.0244, train acc:=0.6
6 - val acc:=0.78
Epoch:=3/10 - train loss:=0.0319 - val loss:=0.0798, train acc:=0.8
2 - val acc:=0.60
Epoch:=4/10 - train loss:=0.0537 - val loss:=0.0755, train acc:=0.6
8 - val acc:=0.68
Epoch:=5/10 - train loss:=0.0602 - val loss:=0.0618, train acc:=0.7
8 - val acc:=0.80
Epoch:=6/10 - train loss:=0.0573 - val loss:=0.0639, train acc:=0.8
0 - val acc:=0.78
Epoch:=7/10 - train loss:=0.0581 - val loss:=0.0566, train acc:=0.7
2 - val acc:=0.88
Epoch:=8/10 - train loss:=0.0565 - val loss:=0.0591, train acc:=0.7
4 - val acc:=0.82
Epoch:=9/10 - train loss:=0.0584 - val loss:=0.0645, train acc:=0.7
8 - val acc:=0.72
Epoch:=10/10 - train loss:=0.0554 - val loss:=0.0551, train acc:=0.
80 - val acc:=0.84

Total time taken (in seconds): 191.03

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 81.77%



Count: 1, j=: 1

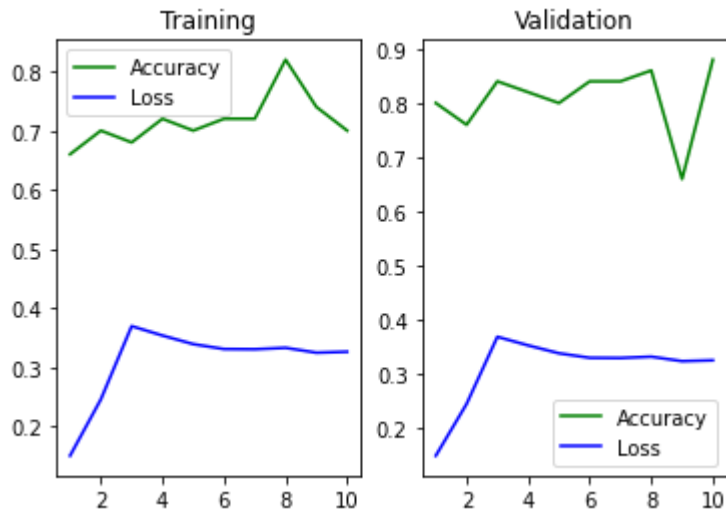
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 2196 *****

```
# Epoch:=1/10 - train loss:=0.0265 - val loss:=0.0213, train acc:=0.6
6 - val acc:=0.80
# Epoch:=2/10 - train loss:=0.0433 - val loss:=0.0653, train acc:=0.7
0 - val acc:=0.76
# Epoch:=3/10 - train loss:=0.0651 - val loss:=0.0634, train acc:=0.6
8 - val acc:=0.84
# Epoch:=4/10 - train loss:=0.0623 - val loss:=0.0606, train acc:=0.7
2 - val acc:=0.82
# Epoch:=5/10 - train loss:=0.0597 - val loss:=0.0569, train acc:=0.7
0 - val acc:=0.80
# Epoch:=6/10 - train loss:=0.0582 - val loss:=0.0635, train acc:=0.7
2 - val acc:=0.84
# Epoch:=7/10 - train loss:=0.0582 - val loss:=0.0584, train acc:=0.7
2 - val acc:=0.84
# Epoch:=8/10 - train loss:=0.0586 - val loss:=0.0563, train acc:=0.8
2 - val acc:=0.86
# Epoch:=9/10 - train loss:=0.0572 - val loss:=0.0711, train acc:=0.7
4 - val acc:=0.66
# Epoch:=10/10 - train loss:=0.0574 - val loss:=0.0565, train acc:=0.
70 - val acc:=0.88
```

Total time taken (in seconds): 188.32

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 81.36%



Count: 2, j=: 1

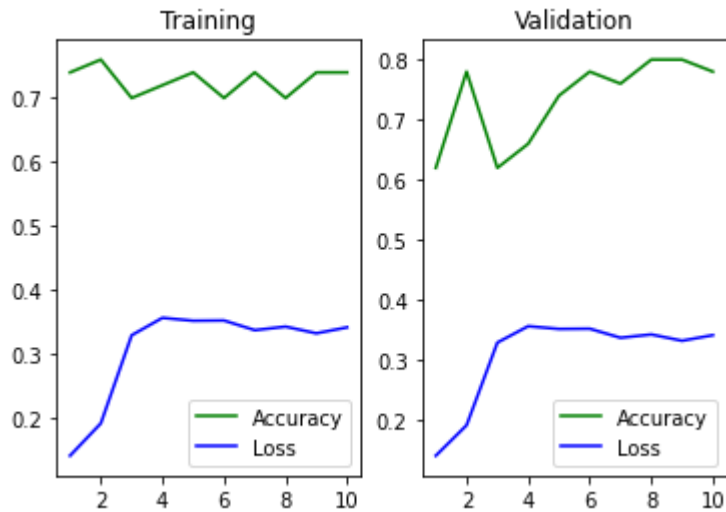
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 7304 *****

```
# Epoch:=1/10 - train loss:=0.0255 - val loss:=0.0313, train acc:=0.7
4 - val acc:=0.62
# Epoch:=2/10 - train loss:=0.0346 - val loss:=0.0323, train acc:=0.7
6 - val acc:=0.78
# Epoch:=3/10 - train loss:=0.0593 - val loss:=0.0868, train acc:=0.7
0 - val acc:=0.62
# Epoch:=4/10 - train loss:=0.0642 - val loss:=0.0857, train acc:=0.7
2 - val acc:=0.66
# Epoch:=5/10 - train loss:=0.0634 - val loss:=0.0697, train acc:=0.7
4 - val acc:=0.74
# Epoch:=6/10 - train loss:=0.0635 - val loss:=0.0589, train acc:=0.7
0 - val acc:=0.78
# Epoch:=7/10 - train loss:=0.0608 - val loss:=0.0610, train acc:=0.7
4 - val acc:=0.76
# Epoch:=8/10 - train loss:=0.0618 - val loss:=0.0568, train acc:=0.7
0 - val acc:=0.80
# Epoch:=9/10 - train loss:=0.0599 - val loss:=0.0670, train acc:=0.7
4 - val acc:=0.80
# Epoch:=10/10 - train loss:=0.0615 - val loss:=0.0759, train acc:=0.
74 - val acc:=0.78
```

Total time taken (in seconds): 188.59

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 75.19%



Count: 3, j=: 1

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 2637 *****

```
# Epoch:=1/10 - train loss:=0.0253 - val loss:=0.0244, train acc:=0.7
0 - val acc:=0.64
# Epoch:=2/10 - train loss:=0.0462 - val loss:=0.0850, train acc:=0.7
4 - val acc:=0.62
# Epoch:=3/10 - train loss:=0.0670 - val loss:=0.0787, train acc:=0.7
2 - val acc:=0.80
# Epoch:=4/10 - train loss:=0.0653 - val loss:=0.0629, train acc:=0.7
4 - val acc:=0.82
# Epoch:=5/10 - train loss:=0.0629 - val loss:=0.0704, train acc:=0.7
0 - val acc:=0.78
# Epoch:=6/10 - train loss:=0.0641 - val loss:=0.0616, train acc:=0.7
0 - val acc:=0.86
# Epoch:=7/10 - train loss:=0.0617 - val loss:=0.0796, train acc:=0.6
0 - val acc:=0.74
# Epoch:=8/10 - train loss:=0.0617 - val loss:=0.0853, train acc:=0.6
2 - val acc:=0.76
# Epoch:=9/10 - train loss:=0.0612 - val loss:=0.0682, train acc:=0.7
0 - val acc:=0.68
# Epoch:=10/10 - train loss:=0.0595 - val loss:=0.0690, train acc:=0.
78 - val acc:=0.80
```

Total time taken (in seconds): 199.29

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 77.32%



Count: 4, j=: 1

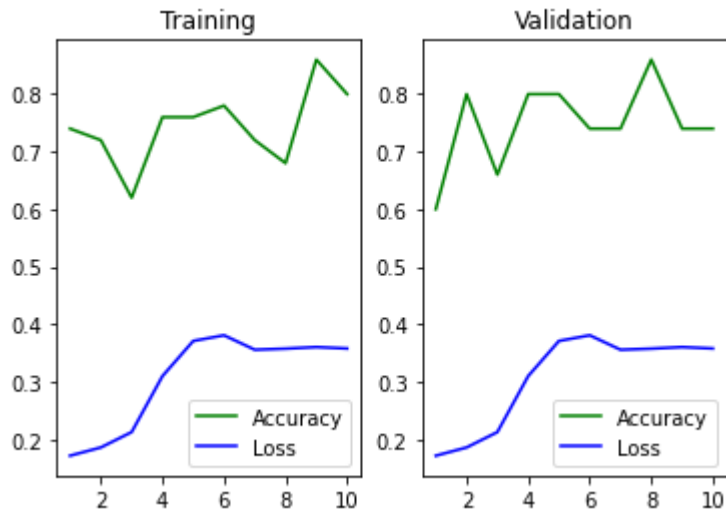
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 4664 *****

```
# Epoch:=1/10 - train loss:=0.0261 - val loss:=0.0372, train acc:=0.7
4 - val acc:=0.60
# Epoch:=2/10 - train loss:=0.0282 - val loss:=0.0261, train acc:=0.7
2 - val acc:=0.80
# Epoch:=3/10 - train loss:=0.0323 - val loss:=0.0403, train acc:=0.6
2 - val acc:=0.66
# Epoch:=4/10 - train loss:=0.0471 - val loss:=0.0587, train acc:=0.7
6 - val acc:=0.80
# Epoch:=5/10 - train loss:=0.0564 - val loss:=0.0553, train acc:=0.7
6 - val acc:=0.80
# Epoch:=6/10 - train loss:=0.0579 - val loss:=0.0638, train acc:=0.7
8 - val acc:=0.74
# Epoch:=7/10 - train loss:=0.0541 - val loss:=0.0606, train acc:=0.7
2 - val acc:=0.74
# Epoch:=8/10 - train loss:=0.0544 - val loss:=0.0559, train acc:=0.6
8 - val acc:=0.86
# Epoch:=9/10 - train loss:=0.0548 - val loss:=0.0700, train acc:=0.8
6 - val acc:=0.74
# Epoch:=10/10 - train loss:=0.0544 - val loss:=0.0686, train acc:=0.
80 - val acc:=0.74
```

Total time taken (in seconds): 192.32

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 78.34%



Count: 5, j=: 1

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 3945 *****

```
# Epoch:=1/10 - train loss:=0.0261 - val loss:=0.0156, train acc:=0.64 - val acc:=0.62
# Epoch:=2/10 - train loss:=0.0290 - val loss:=0.0256, train acc:=0.72 - val acc:=0.68
# Epoch:=3/10 - train loss:=0.0534 - val loss:=0.0671, train acc:=0.74 - val acc:=0.76
# Epoch:=4/10 - train loss:=0.0643 - val loss:=0.0629, train acc:=0.76 - val acc:=0.88
# Epoch:=5/10 - train loss:=0.0630 - val loss:=0.0650, train acc:=0.80 - val acc:=0.76
# Epoch:=6/10 - train loss:=0.0635 - val loss:=0.0720, train acc:=0.76 - val acc:=0.68
# Epoch:=7/10 - train loss:=0.0608 - val loss:=0.0620, train acc:=0.72 - val acc:=0.76
# Epoch:=8/10 - train loss:=0.0612 - val loss:=0.0565, train acc:=0.74 - val acc:=0.86
# Epoch:=9/10 - train loss:=0.0608 - val loss:=0.0654, train acc:=0.70 - val acc:=0.78
# Epoch:=10/10 - train loss:=0.0598 - val loss:=0.0620, train acc:=0.78 - val acc:=0.84
```

Total time taken (in seconds): 192.91

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 80.20%



Count: 6, j=: 1

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 8249 *****

```
# Epoch:=1/10 - train loss:=0.0263 - val loss:=0.0204, train acc:=0.6
6 - val acc:=0.68
# Epoch:=2/10 - train loss:=0.0292 - val loss:=0.0309, train acc:=0.6
6 - val acc:=0.80
# Epoch:=3/10 - train loss:=0.0587 - val loss:=0.0840, train acc:=0.6
6 - val acc:=0.70
# Epoch:=4/10 - train loss:=0.0588 - val loss:=0.0797, train acc:=0.7
8 - val acc:=0.68
# Epoch:=5/10 - train loss:=0.0585 - val loss:=0.0694, train acc:=0.7
4 - val acc:=0.76
# Epoch:=6/10 - train loss:=0.0593 - val loss:=0.0593, train acc:=0.7
4 - val acc:=0.84
# Epoch:=7/10 - train loss:=0.0575 - val loss:=0.0659, train acc:=0.7
4 - val acc:=0.78
# Epoch:=8/10 - train loss:=0.0577 - val loss:=0.0614, train acc:=0.7
6 - val acc:=0.82
# Epoch:=9/10 - train loss:=0.0574 - val loss:=0.0642, train acc:=0.6
6 - val acc:=0.84
# Epoch:=10/10 - train loss:=0.0575 - val loss:=0.0547, train acc:=0.
70 - val acc:=0.86
```

Total time taken (in seconds): 192.47

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 81.89%



Count: 7, j=: 1

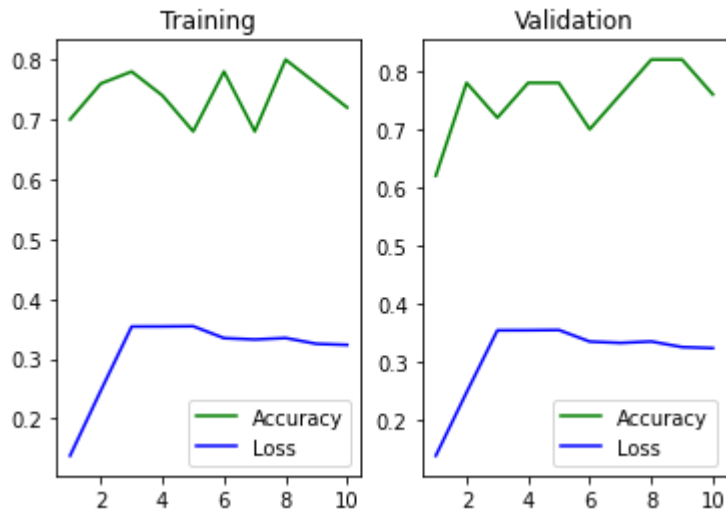
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 3899 *****

```
# Epoch:=1/10 - train loss:=0.0251 - val loss:=0.0290, train acc:=0.7
0 - val acc:=0.62
# Epoch:=2/10 - train loss:=0.0449 - val loss:=0.0712, train acc:=0.7
6 - val acc:=0.78
# Epoch:=3/10 - train loss:=0.0643 - val loss:=0.0739, train acc:=0.7
8 - val acc:=0.72
# Epoch:=4/10 - train loss:=0.0643 - val loss:=0.0724, train acc:=0.7
4 - val acc:=0.78
# Epoch:=5/10 - train loss:=0.0644 - val loss:=0.0810, train acc:=0.6
8 - val acc:=0.78
# Epoch:=6/10 - train loss:=0.0608 - val loss:=0.0634, train acc:=0.7
8 - val acc:=0.70
# Epoch:=7/10 - train loss:=0.0603 - val loss:=0.0747, train acc:=0.6
8 - val acc:=0.76
# Epoch:=8/10 - train loss:=0.0608 - val loss:=0.0616, train acc:=0.8
0 - val acc:=0.82
# Epoch:=9/10 - train loss:=0.0590 - val loss:=0.0555, train acc:=0.7
6 - val acc:=0.82
# Epoch:=10/10 - train loss:=0.0587 - val loss:=0.0664, train acc:=0.
72 - val acc:=0.76
```

Total time taken (in seconds): 185.15

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 78.02%



Count: 8, j=: 1

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 8714 *****

```
# Epoch:=1/10 - train loss:=0.0324 - val loss:=0.0168, train acc:=0.5
8 - val acc:=0.72
# Epoch:=2/10 - train loss:=0.0574 - val loss:=0.0745, train acc:=0.7
4 - val acc:=0.76
# Epoch:=3/10 - train loss:=0.0687 - val loss:=0.0761, train acc:=0.7
0 - val acc:=0.66
# Epoch:=4/10 - train loss:=0.0684 - val loss:=0.0721, train acc:=0.7
6 - val acc:=0.78
# Epoch:=5/10 - train loss:=0.0694 - val loss:=0.0639, train acc:=0.7
0 - val acc:=0.82
# Epoch:=6/10 - train loss:=0.0662 - val loss:=0.0615, train acc:=0.6
8 - val acc:=0.84
# Epoch:=7/10 - train loss:=0.0653 - val loss:=0.0798, train acc:=0.7
0 - val acc:=0.70
# Epoch:=8/10 - train loss:=0.0626 - val loss:=0.0723, train acc:=0.7
2 - val acc:=0.82
# Epoch:=9/10 - train loss:=0.0602 - val loss:=0.0660, train acc:=0.7
2 - val acc:=0.70
# Epoch:=10/10 - train loss:=0.0606 - val loss:=0.0552, train acc:=0.
76 - val acc:=0.88
```

Total time taken (in seconds): 184.22

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 81.98%



Count: 9, j=: 1

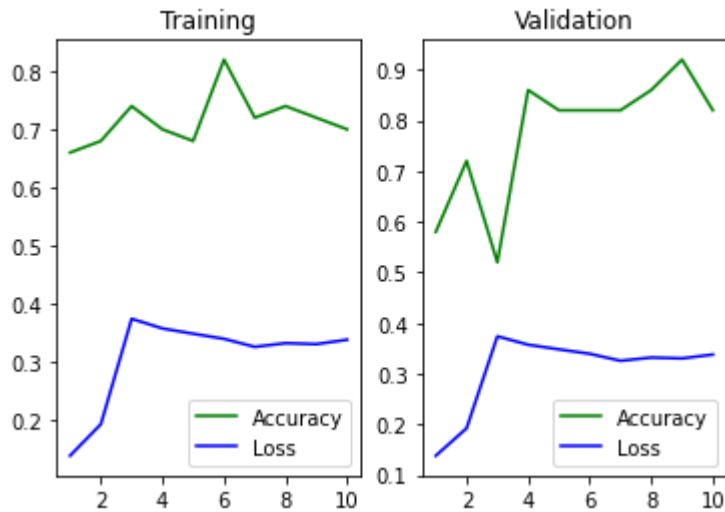
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 6686 *****

```
# Epoch:=1/10 - train loss:=0.0254 - val loss:=0.0286, train acc:=0.6
6 - val acc:=0.58
# Epoch:=2/10 - train loss:=0.0354 - val loss:=0.0704, train acc:=0.6
8 - val acc:=0.72
# Epoch:=3/10 - train loss:=0.0687 - val loss:=0.0952, train acc:=0.7
4 - val acc:=0.52
# Epoch:=4/10 - train loss:=0.0657 - val loss:=0.0603, train acc:=0.7
0 - val acc:=0.86
# Epoch:=5/10 - train loss:=0.0640 - val loss:=0.0576, train acc:=0.6
8 - val acc:=0.82
# Epoch:=6/10 - train loss:=0.0624 - val loss:=0.0567, train acc:=0.8
2 - val acc:=0.82
# Epoch:=7/10 - train loss:=0.0599 - val loss:=0.0617, train acc:=0.7
2 - val acc:=0.82
# Epoch:=8/10 - train loss:=0.0610 - val loss:=0.0640, train acc:=0.7
4 - val acc:=0.86
# Epoch:=9/10 - train loss:=0.0607 - val loss:=0.0601, train acc:=0.7
2 - val acc:=0.92
# Epoch:=10/10 - train loss:=0.0621 - val loss:=0.0648, train acc:=0.
70 - val acc:=0.82
```

Total time taken (in seconds): 184.10

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 79.54%



Count: 0, j=: 1

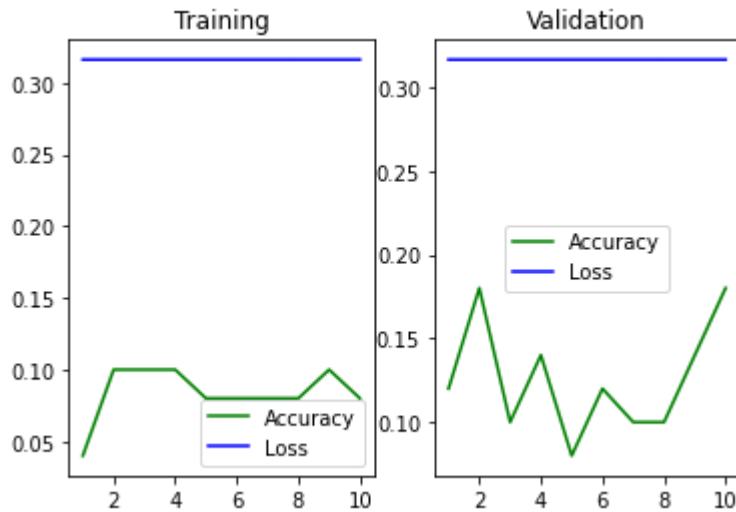
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1838 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.12
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.18
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.10
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.08
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.10
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.10
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.18
```

Total time taken (in seconds): 194.08

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 1, j=: 1

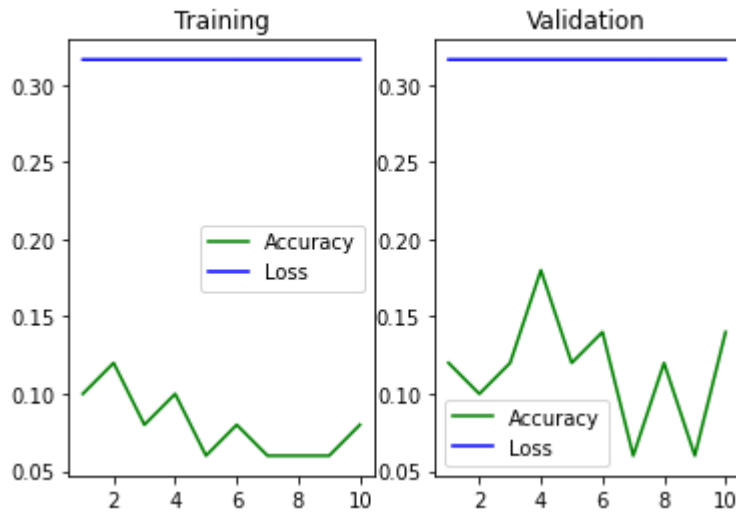
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1667 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.12
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.10
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.18
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.12
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.06
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.12
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.06
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
```

Total time taken (in seconds): 194.62

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 2, j=: 1

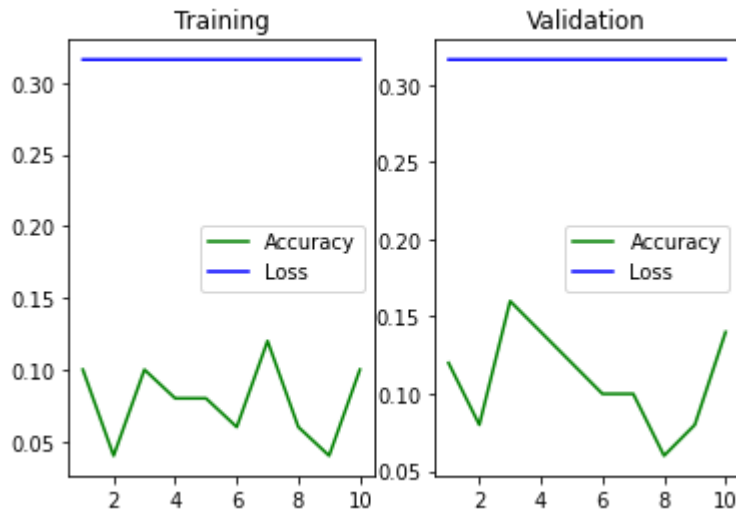
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 5517 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.12
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.08
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.16
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.10
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.10
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.06
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.08
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
```

Total time taken (in seconds): 194.06

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 3, j=: 1

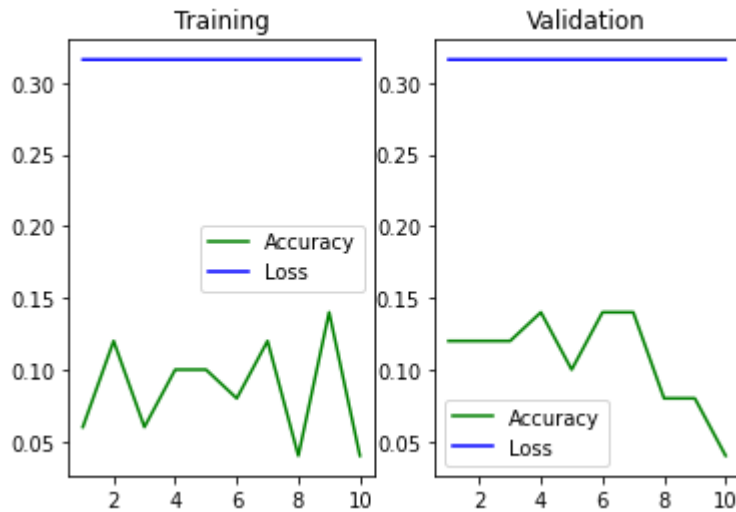
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1710 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.12
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.12
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.12
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.10
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.14
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.08
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.14 - val acc:=0.08
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.04
```

Total time taken (in seconds): 193.35

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 4, j=: 1

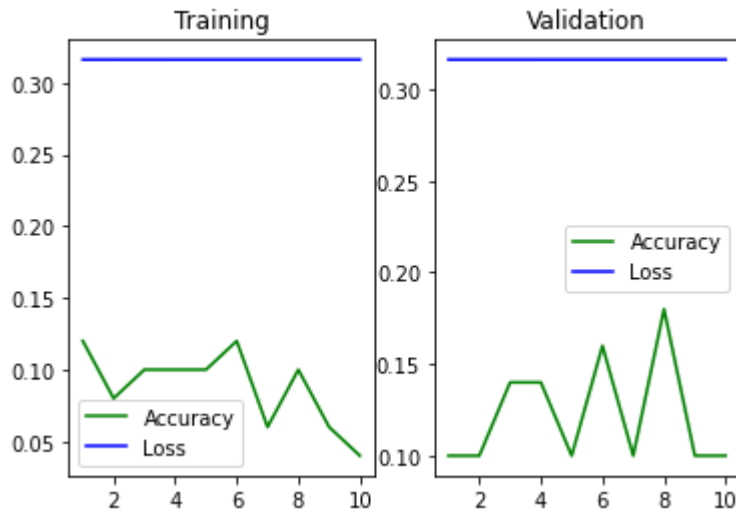
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 3010 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
2 - val acc:=0.10
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
8 - val acc:=0.10
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.14
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.14
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.10
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
2 - val acc:=0.16
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
6 - val acc:=0.10
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.18
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
6 - val acc:=0.10
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
4 - val acc:=0.10
```

Total time taken (in seconds): 196.55

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 5, j=: 1

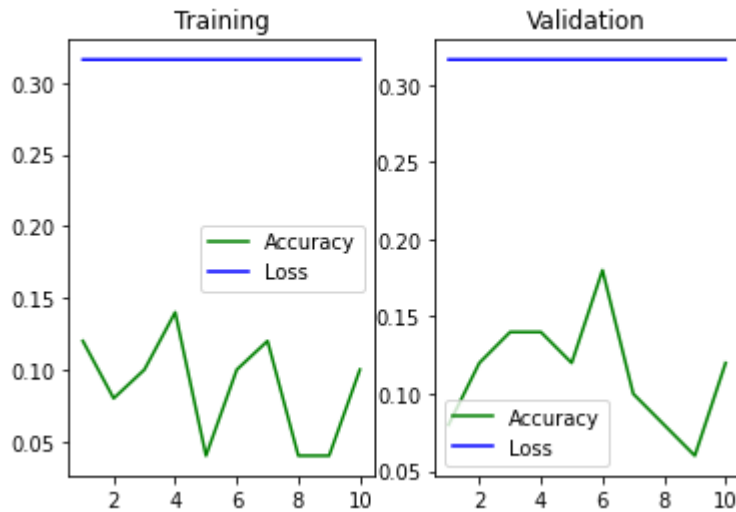
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1886 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.08
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.14 - val acc:=0.14
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.12
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.18
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.10
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.08
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.06
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.12
```

Total time taken (in seconds): 195.19

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 6, j=: 1

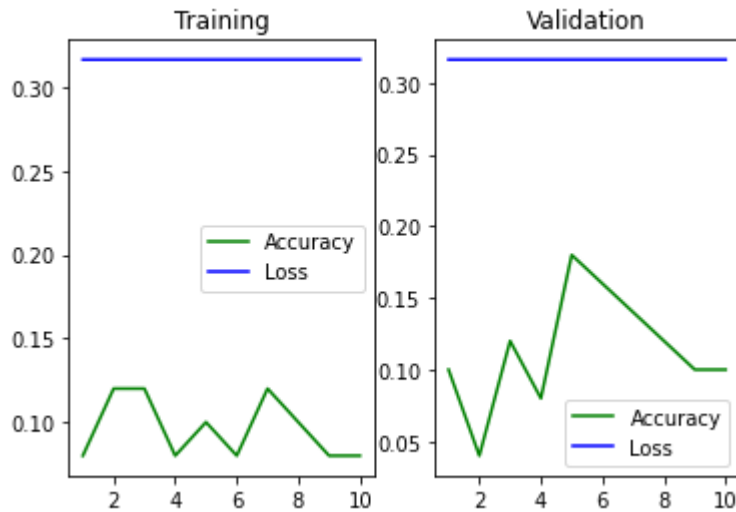
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 3712 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.10
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.04
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.12
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.08
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.18
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.16
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.14
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.12
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.10
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.10
```

Total time taken (in seconds): 195.08

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 7, j=: 1

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 3952 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
4 - val acc:=0.08
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.14
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
2 - val acc:=0.20
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
4 - val acc:=0.12
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
4 - val acc:=0.06
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
2 - val acc:=0.16
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.0
6 - val acc:=0.06
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
0 - val acc:=0.14
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.1
6 - val acc:=0.04
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.
08 - val acc:=0.12
```

Total time taken (in seconds): 194.95

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 8, j=: 1

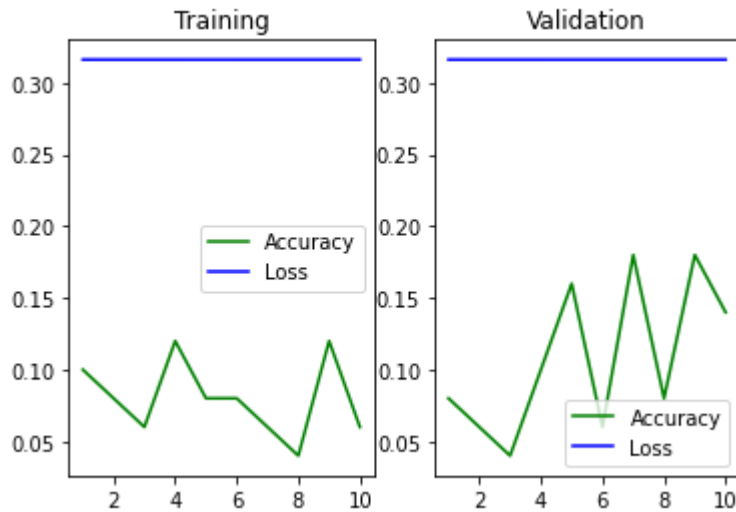
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 7026 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.08
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.06
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.04
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.10
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.16
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.06
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.18
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.04 - val acc:=0.08
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.18
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.14
```

Total time taken (in seconds): 201.90

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 9, j=: 1

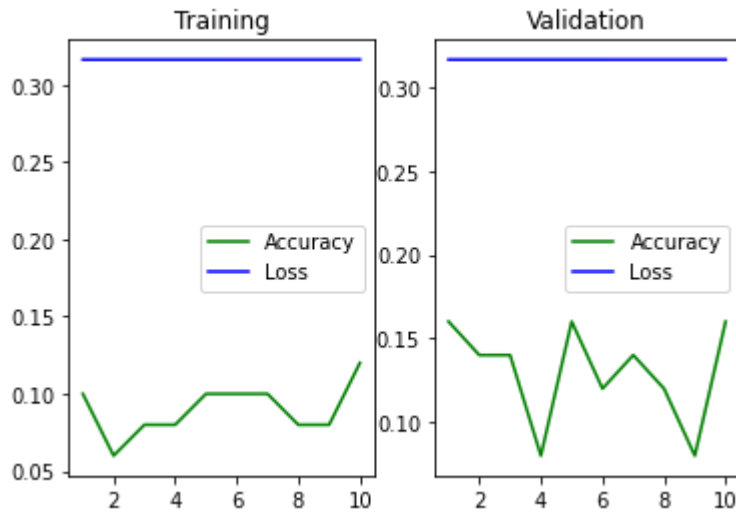
***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 9294 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.16
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.14
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.08
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.16
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.12
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.08
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.16
```

Total time taken (in seconds): 194.08

Finished training model: mlp_on_gpu_RegL1

***** Testing *****
 mlp_on_gpu_RegL1 model accuracy = 10.00%



Count: 0, j=: 1

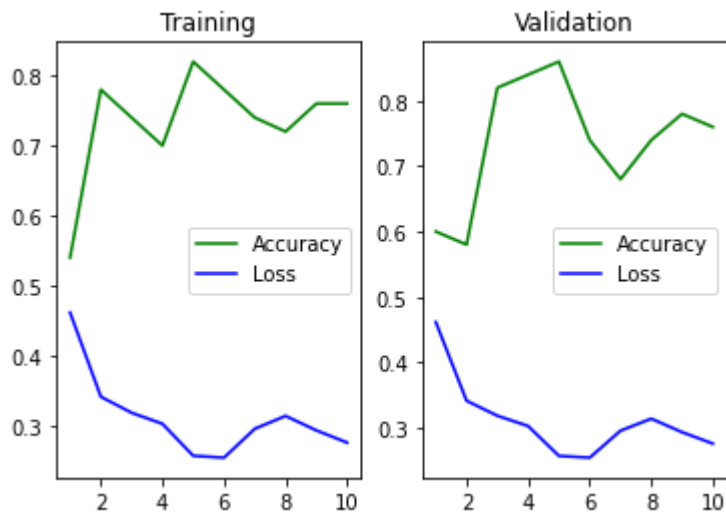
***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 9596 *****

```
# Epoch:=1/10 - train loss:=0.0246 - val loss:=0.0173, train acc:=0.54 - val acc:=0.60
# Epoch:=2/10 - train loss:=0.0182 - val loss:=0.0244, train acc:=0.78 - val acc:=0.58
# Epoch:=3/10 - train loss:=0.0170 - val loss:=0.0134, train acc:=0.74 - val acc:=0.82
# Epoch:=4/10 - train loss:=0.0162 - val loss:=0.0127, train acc:=0.70 - val acc:=0.84
# Epoch:=5/10 - train loss:=0.0137 - val loss:=0.0112, train acc:=0.82 - val acc:=0.86
# Epoch:=6/10 - train loss:=0.0136 - val loss:=0.0187, train acc:=0.78 - val acc:=0.74
# Epoch:=7/10 - train loss:=0.0158 - val loss:=0.0213, train acc:=0.74 - val acc:=0.68
# Epoch:=8/10 - train loss:=0.0168 - val loss:=0.0139, train acc:=0.72 - val acc:=0.74
# Epoch:=9/10 - train loss:=0.0156 - val loss:=0.0147, train acc:=0.76 - val acc:=0.78
# Epoch:=10/10 - train loss:=0.0147 - val loss:=0.0165, train acc:=0.76 - val acc:=0.76
```

Total time taken (in seconds): 196.16

Finished training model: mlp_on_gpu_RegL2

***** Testing *****
 mlp_on_gpu_RegL2 model accuracy = 77.53%



Count: 1, j=: 1

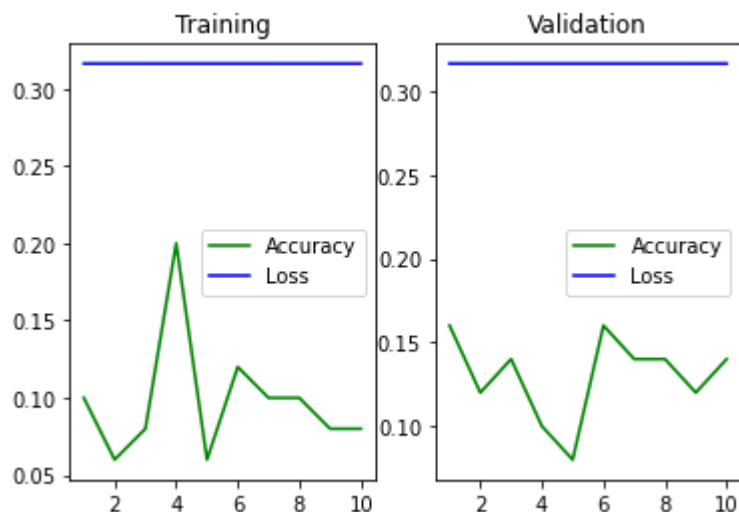
***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 9285 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.16
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.12
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.20 - val acc:=0.10
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.08
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.16
# Epoch:=7/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=8/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.10 - val acc:=0.14
# Epoch:=9/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
# Epoch:=10/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.14
```

Total time taken (in seconds): 196.98

Finished training model: mlp_on_gpu_RegL2

***** Testing *****
 mlp_on_gpu_RegL2 model accuracy = 10.00%



Count: 2, j=: 1

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 5981 *****

```
# Epoch:=1/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.12 - val acc:=0.12
# Epoch:=2/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.08
# Epoch:=3/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.16
# Epoch:=4/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.08
# Epoch:=5/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.06 - val acc:=0.08
# Epoch:=6/10 - train loss:=0.0461 - val loss:=0.0461, train acc:=0.08 - val acc:=0.12
```

In []: