

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 [29]:

```
def main():  
    for j in range(1):  
        if j==0:  
            data = load_data('mnist')  
            size_hidden = [128, 128, 128]  
            learning_rate = 5e-4  
            weight_coeff = 1e-2  
  
        if j==1:  
            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_hidden, 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_hidden, 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 [30]:

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

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=: 0

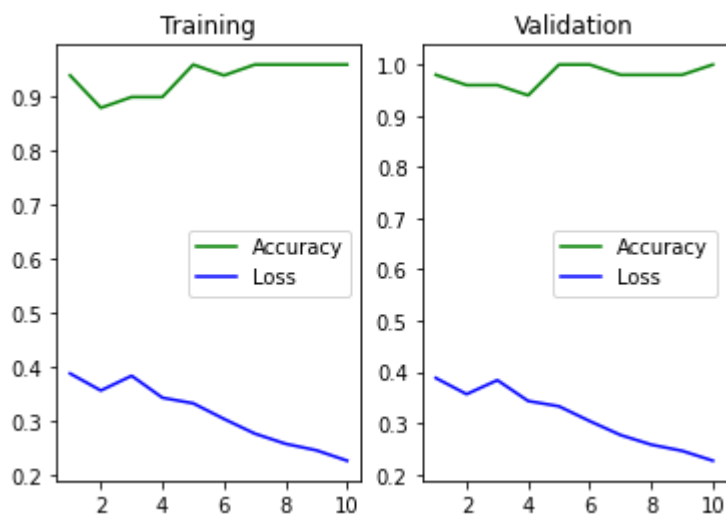
***** Training model: mlp_on_gpu_default with optimizer: RM
 Sprop and seed: 4817 *****

```
# Epoch:=1/10 - train loss:=0.0133 - val loss:=0.0078, train acc:=0.94 - val acc:=0.98
# Epoch:=2/10 - train loss:=0.0122 - val loss:=0.0133, train acc:=0.88 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0131 - val loss:=0.0104, train acc:=0.90 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0118 - val loss:=0.0100, train acc:=0.90 - val acc:=0.94
# Epoch:=5/10 - train loss:=0.0114 - val loss:=0.0107, train acc:=0.96 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0104 - val loss:=0.0091, train acc:=0.94 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0095 - val loss:=0.0123, train acc:=0.96 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0088 - val loss:=0.0121, train acc:=0.96 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0084 - val loss:=0.0099, train acc:=0.96 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0078 - val loss:=0.0092, train acc:=0.96 - val acc:=1.00
```

Total time taken (in seconds): 195.47

Finished training model: mlp_on_gpu_default

***** Testing *****
 mlp_on_gpu_default model accuracy = 95.84%



Count: 1, j=: 0

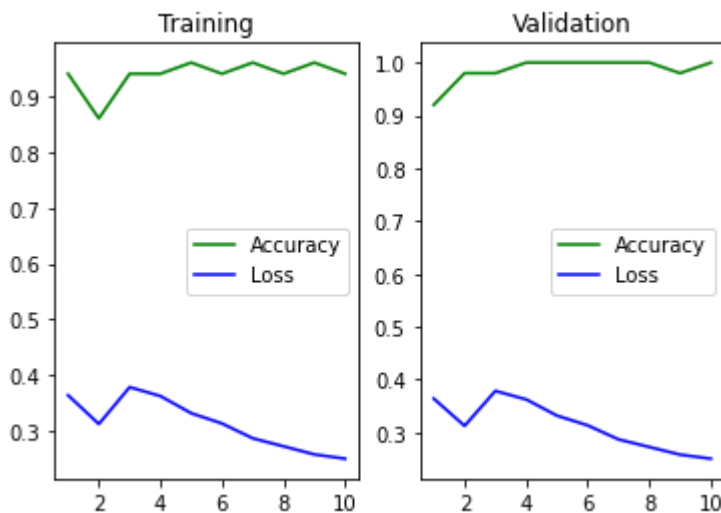
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 4908 *****

```
# Epoch:=1/10 - train loss:=0.0136 - val loss:=0.0083, train acc:=0.94 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0116 - val loss:=0.0161, train acc:=0.86 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0141 - val loss:=0.0134, train acc:=0.94 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0135 - val loss:=0.0128, train acc:=0.94 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0124 - val loss:=0.0116, train acc:=0.96 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0117 - val loss:=0.0097, train acc:=0.94 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0107 - val loss:=0.0157, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0102 - val loss:=0.0101, train acc:=0.94 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0096 - val loss:=0.0107, train acc:=0.96 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0093 - val loss:=0.0094, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 188.57

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 96.35%



Count: 2, j=: 0

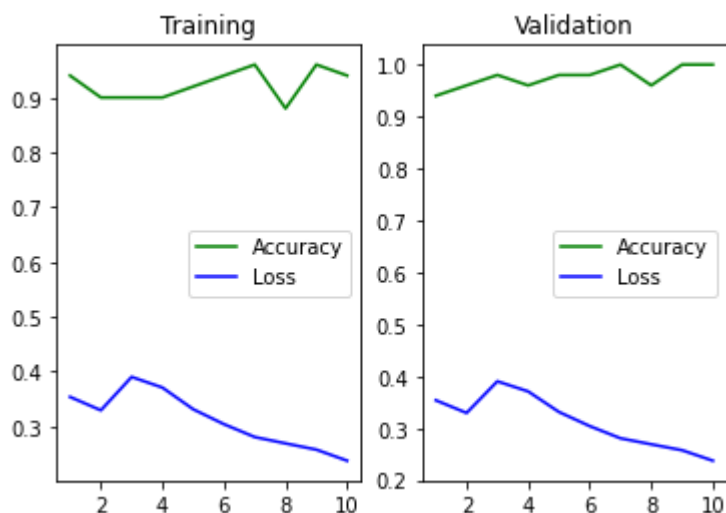
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 9149 *****

```
# Epoch:=1/10 - train loss:=0.0140 - val loss:=0.0087, train acc:=0.94 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0130 - val loss:=0.0179, train acc:=0.90 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0154 - val loss:=0.0124, train acc:=0.90 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0146 - val loss:=0.0116, train acc:=0.90 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0131 - val loss:=0.0108, train acc:=0.92 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0120 - val loss:=0.0114, train acc:=0.94 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0111 - val loss:=0.0102, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0106 - val loss:=0.0109, train acc:=0.88 - val acc:=0.96
# Epoch:=9/10 - train loss:=0.0102 - val loss:=0.0104, train acc:=0.96 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0094 - val loss:=0.0097, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 191.20

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 96.12%



Count: 3, j=: 0

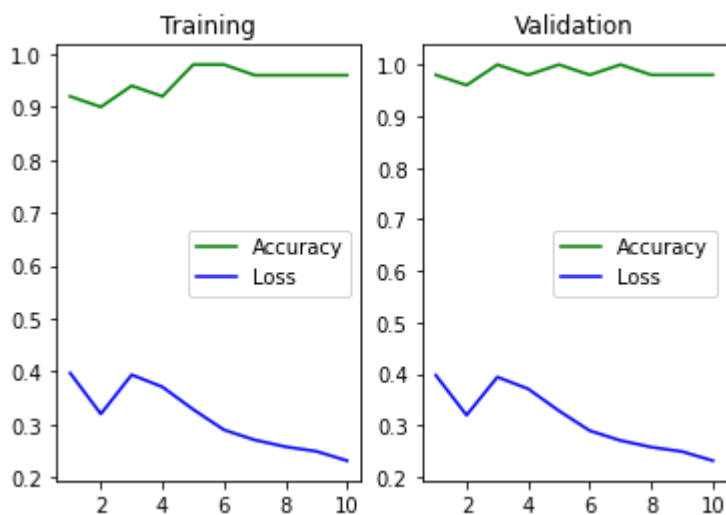
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 3528 *****

```
# Epoch:=1/10 - train loss:=0.0144 - val loss:=0.0075, train acc:=0.92 - val acc:=0.98
# Epoch:=2/10 - train loss:=0.0116 - val loss:=0.0150, train acc:=0.90 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0143 - val loss:=0.0111, train acc:=0.94 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0135 - val loss:=0.0107, train acc:=0.92 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0119 - val loss:=0.0117, train acc:=0.98 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0105 - val loss:=0.0092, train acc:=0.98 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0098 - val loss:=0.0111, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0094 - val loss:=0.0104, train acc:=0.96 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0091 - val loss:=0.0094, train acc:=0.96 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0084 - val loss:=0.0089, train acc:=0.96 - val acc:=0.98
```

Total time taken (in seconds): 192.24

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.98%



Count: 4, j=: 0

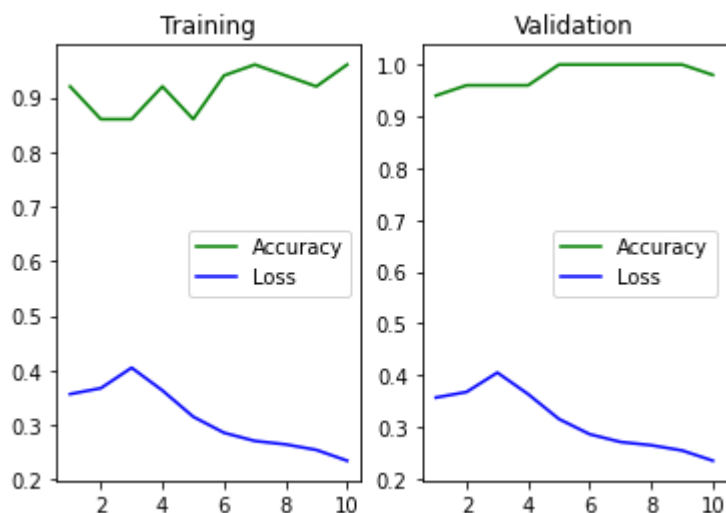
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 7931 *****

```
# Epoch:=1/10 - train loss:=0.0138 - val loss:=0.0094, train acc:=0.92 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0142 - val loss:=0.0130, train acc:=0.86 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0157 - val loss:=0.0127, train acc:=0.86 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0141 - val loss:=0.0136, train acc:=0.92 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0122 - val loss:=0.0098, train acc:=0.86 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0111 - val loss:=0.0116, train acc:=0.94 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0105 - val loss:=0.0107, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0102 - val loss:=0.0109, train acc:=0.94 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0098 - val loss:=0.0101, train acc:=0.92 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0091 - val loss:=0.0107, train acc:=0.96 - val acc:=0.98
```

Total time taken (in seconds): 191.19

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.79%



Count: 5, j=: 0

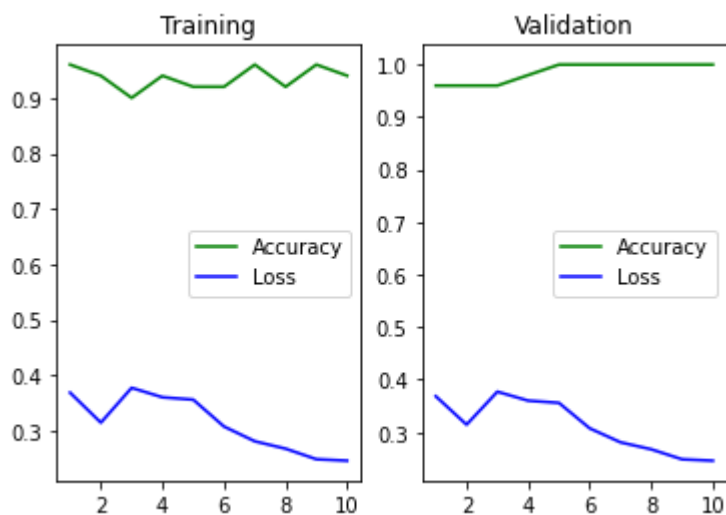
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 4282 *****

```
# Epoch:=1/10 - train loss:=0.0136 - val loss:=0.0076, train acc:=0.96 - val acc:=0.96
# Epoch:=2/10 - train loss:=0.0116 - val loss:=0.0143, train acc:=0.94 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0139 - val loss:=0.0112, train acc:=0.90 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0133 - val loss:=0.0123, train acc:=0.94 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0131 - val loss:=0.0116, train acc:=0.92 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0113 - val loss:=0.0112, train acc:=0.92 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0104 - val loss:=0.0093, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0099 - val loss:=0.0096, train acc:=0.92 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0092 - val loss:=0.0096, train acc:=0.96 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0091 - val loss:=0.0101, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 191.34

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 96.31%



Count: 6, j=: 0

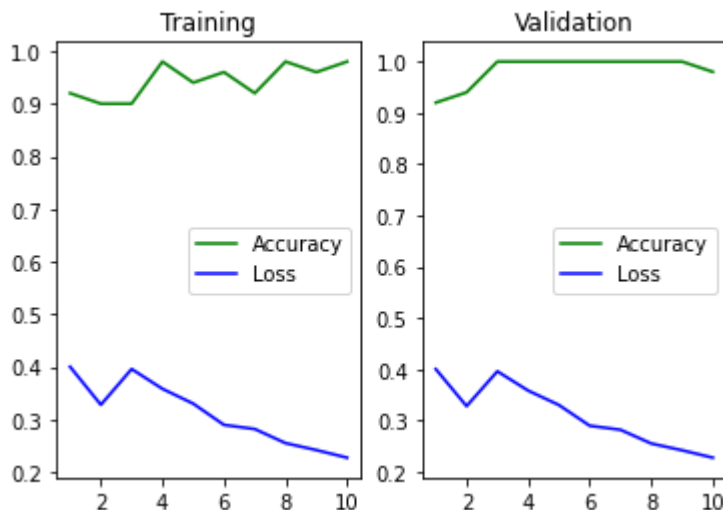
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 6406 *****

```
# Epoch:=1/10 - train loss:=0.0141 - val loss:=0.0074, train acc:=0.92 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0116 - val loss:=0.0180, train acc:=0.90 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0140 - val loss:=0.0101, train acc:=0.90 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0127 - val loss:=0.0111, train acc:=0.98 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0117 - val loss:=0.0103, train acc:=0.94 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0102 - val loss:=0.0098, train acc:=0.96 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0100 - val loss:=0.0092, train acc:=0.92 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0090 - val loss:=0.0082, train acc:=0.98 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0085 - val loss:=0.0086, train acc:=0.96 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0080 - val loss:=0.0113, train acc:=0.98 - val acc:=0.98
```

Total time taken (in seconds): 191.36

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.53%



Count: 7, j=: 0

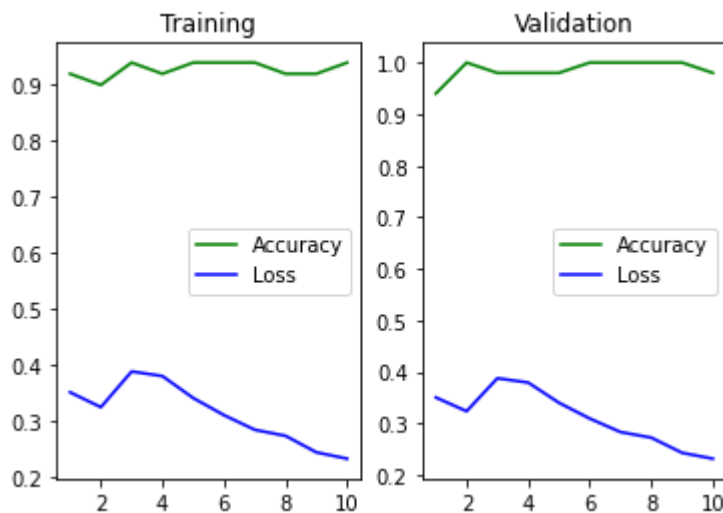
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 7930 *****

```
# Epoch:=1/10 - train loss:=0.0148 - val loss:=0.0091, train acc:=0.9
2 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0137 - val loss:=0.0133, train acc:=0.9
0 - val acc:=1.00
# Epoch:=3/10 - train loss:=0.0164 - val loss:=0.0139, train acc:=0.9
4 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0161 - val loss:=0.0117, train acc:=0.9
2 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0144 - val loss:=0.0117, train acc:=0.9
4 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0131 - val loss:=0.0119, train acc:=0.9
4 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0120 - val loss:=0.0099, train acc:=0.9
4 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0115 - val loss:=0.0100, train acc:=0.9
2 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0103 - val loss:=0.0116, train acc:=0.9
2 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0098 - val loss:=0.0098, train acc:=0.
94 - val acc:=0.98
```

Total time taken (in seconds): 188.16

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.99%



Count: 8, j=: 0

***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 9173 *****

```
# Epoch:=1/10 - train loss:=0.0139 - val loss:=0.0086, train acc:=0.94 - val acc:=0.96
# Epoch:=2/10 - train loss:=0.0131 - val loss:=0.0127, train acc:=0.90 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0147 - val loss:=0.0205, train acc:=0.92 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0140 - val loss:=0.0105, train acc:=0.92 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0126 - val loss:=0.0109, train acc:=0.96 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0115 - val loss:=0.0108, train acc:=0.96 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0108 - val loss:=0.0104, train acc:=0.96 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0105 - val loss:=0.0119, train acc:=0.98 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0097 - val loss:=0.0099, train acc:=0.94 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0088 - val loss:=0.0097, train acc:=1.00 - val acc:=0.98
```

Total time taken (in seconds): 187.11

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.72%



Count: 9, j=: 0

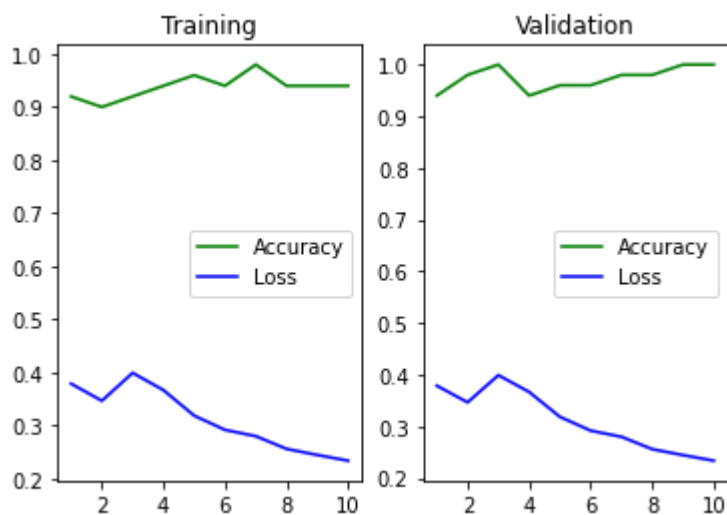
***** Training model: mlp_on_gpu_default with optimizer: RM
Sprop and seed: 7992 *****

```
# Epoch:=1/10 - train loss:=0.0146 - val loss:=0.0071, train acc:=0.92 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0133 - val loss:=0.0129, train acc:=0.90 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0154 - val loss:=0.0121, train acc:=0.92 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0141 - val loss:=0.0128, train acc:=0.94 - val acc:=0.94
# Epoch:=5/10 - train loss:=0.0122 - val loss:=0.0128, train acc:=0.96 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0112 - val loss:=0.0115, train acc:=0.94 - val acc:=0.96
# Epoch:=7/10 - train loss:=0.0108 - val loss:=0.0092, train acc:=0.98 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0098 - val loss:=0.0110, train acc:=0.94 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0094 - val loss:=0.0100, train acc:=0.94 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0090 - val loss:=0.0109, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 189.56

Finished training model: mlp_on_gpu_default

***** Testing *****
mlp_on_gpu_default model accuracy = 95.92%



Count: 0, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1938 *****

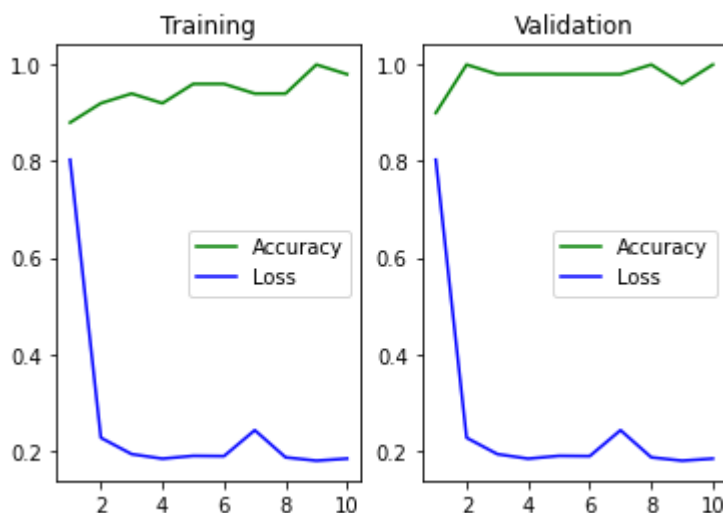
```
# Epoch:=1/10 - train loss:=0.0209 - val loss:=0.0062, train acc:=0.88 - val acc:=0.90
# Epoch:=2/10 - train loss:=0.0059 - val loss:=0.0046, train acc:=0.92 - val acc:=1.00
# Epoch:=3/10 - train loss:=0.0050 - val loss:=0.0038, train acc:=0.94 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0048 - val loss:=0.0055, train acc:=0.92 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0050 - val loss:=0.0051, train acc:=0.96 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0049 - val loss:=0.0061, train acc:=0.96 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0063 - val loss:=0.0055, train acc:=0.94 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0049 - val loss:=0.0055, train acc:=0.94 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0047 - val loss:=0.0074, train acc:=1.00 - val acc:=0.96
# Epoch:=10/10 - train loss:=0.0048 - val loss:=0.0055, train acc:=0.98 - val acc:=1.00
```

Total time taken (in seconds): 200.73

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.60%



Count: 1, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 3329 *****

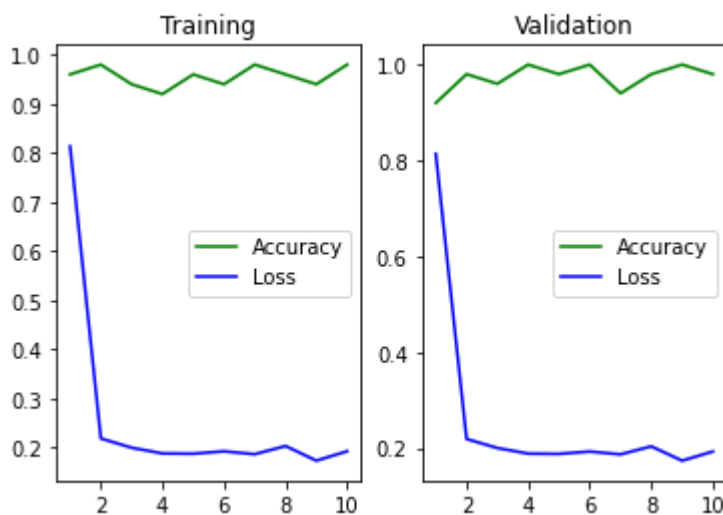
```
# Epoch:=1/10 - train loss:=0.0229 - val loss:=0.0076, train acc:=0.96 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0062 - val loss:=0.0050, train acc:=0.98 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0056 - val loss:=0.0063, train acc:=0.94 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0053 - val loss:=0.0060, train acc:=0.92 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0053 - val loss:=0.0049, train acc:=0.96 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0054 - val loss:=0.0053, train acc:=0.94 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0053 - val loss:=0.0080, train acc:=0.98 - val acc:=0.94
# Epoch:=8/10 - train loss:=0.0057 - val loss:=0.0058, train acc:=0.96 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0049 - val loss:=0.0040, train acc:=0.94 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0054 - val loss:=0.0060, train acc:=0.98 - val acc:=0.98
```

Total time taken (in seconds): 197.64

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 95.89%



Count: 2, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 8234 *****

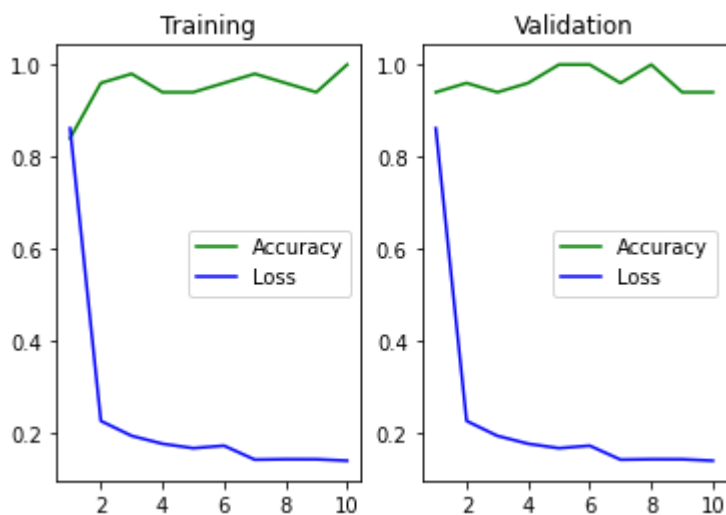
```
# Epoch:=1/10 - train loss:=0.0263 - val loss:=0.0081, train acc:=0.84 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0069 - val loss:=0.0056, train acc:=0.96 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0059 - val loss:=0.0051, train acc:=0.98 - val acc:=0.94
# Epoch:=4/10 - train loss:=0.0054 - val loss:=0.0048, train acc:=0.94 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0051 - val loss:=0.0065, train acc:=0.94 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0053 - val loss:=0.0040, train acc:=0.96 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0043 - val loss:=0.0045, train acc:=0.98 - val acc:=0.96
# Epoch:=8/10 - train loss:=0.0044 - val loss:=0.0061, train acc:=0.96 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0044 - val loss:=0.0048, train acc:=0.94 - val acc:=0.94
# Epoch:=10/10 - train loss:=0.0043 - val loss:=0.0061, train acc:=1.00 - val acc:=0.94
```

Total time taken (in seconds): 198.94

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.33%



Count: 3, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 7425 *****

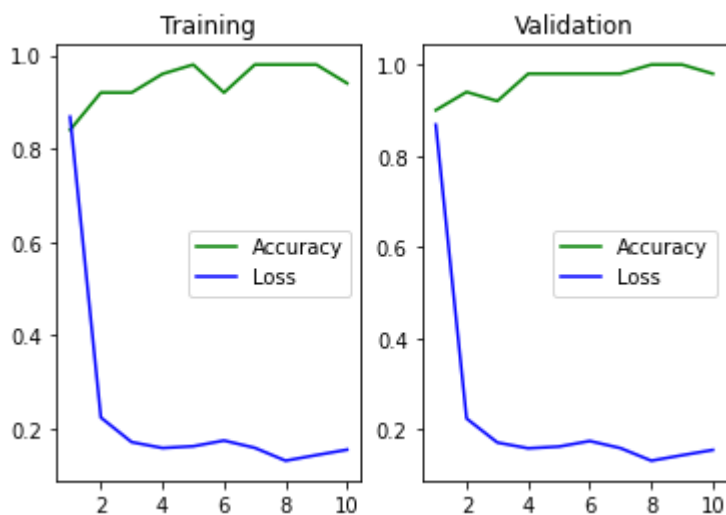
```
# Epoch:=1/10 - train loss:=0.0268 - val loss:=0.0094, train acc:=0.84 - val acc:=0.90
# Epoch:=2/10 - train loss:=0.0069 - val loss:=0.0053, train acc:=0.92 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0053 - val loss:=0.0053, train acc:=0.92 - val acc:=0.92
# Epoch:=4/10 - train loss:=0.0049 - val loss:=0.0043, train acc:=0.96 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0050 - val loss:=0.0064, train acc:=0.98 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0054 - val loss:=0.0046, train acc:=0.92 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0049 - val loss:=0.0044, train acc:=0.98 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0040 - val loss:=0.0049, train acc:=0.98 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0044 - val loss:=0.0051, train acc:=0.98 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0048 - val loss:=0.0047, train acc:=0.94 - val acc:=0.98
```

Total time taken (in seconds): 201.87

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.56%



Count: 4, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 1391 *****

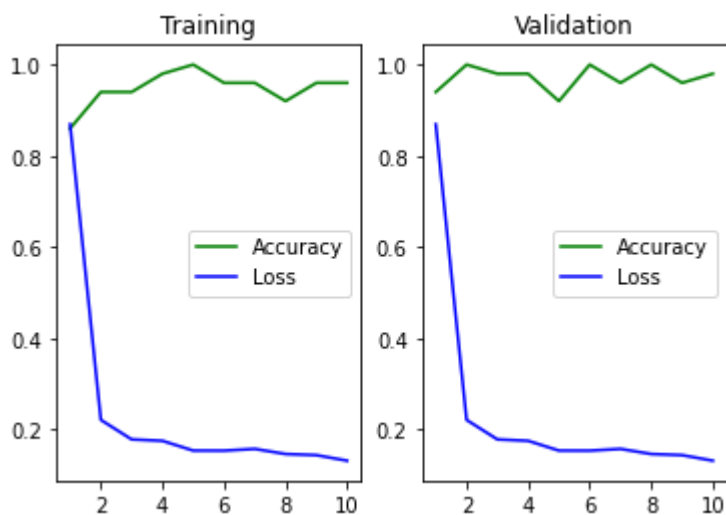
```
# Epoch:=1/10 - train loss:=0.0273 - val loss:=0.0091, train acc:=0.86 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0070 - val loss:=0.0051, train acc:=0.94 - val acc:=1.00
# Epoch:=3/10 - train loss:=0.0056 - val loss:=0.0051, train acc:=0.94 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0055 - val loss:=0.0082, train acc:=0.98 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0048 - val loss:=0.0053, train acc:=1.00 - val acc:=0.92
# Epoch:=6/10 - train loss:=0.0048 - val loss:=0.0045, train acc:=0.96 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0050 - val loss:=0.0055, train acc:=0.96 - val acc:=0.96
# Epoch:=8/10 - train loss:=0.0046 - val loss:=0.0052, train acc:=0.92 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0045 - val loss:=0.0048, train acc:=0.96 - val acc:=0.96
# Epoch:=10/10 - train loss:=0.0042 - val loss:=0.0053, train acc:=0.96 - val acc:=0.98
```

Total time taken (in seconds): 198.74

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 95.89%



Count: 5, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 7963 *****

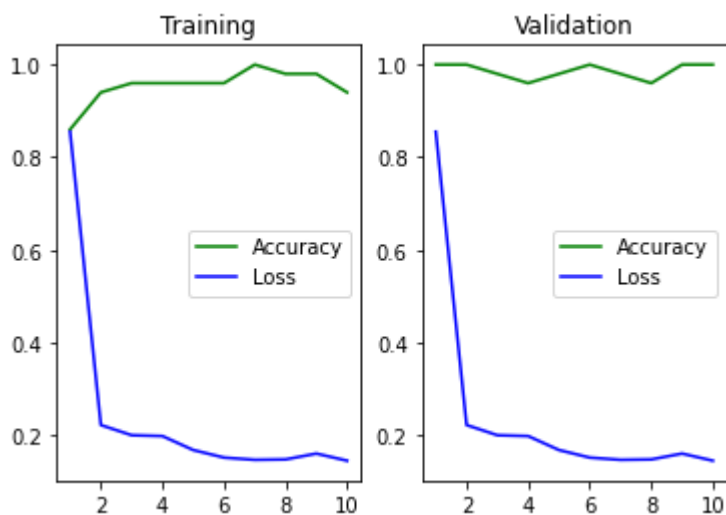
```
# Epoch:=1/10 - train loss:=0.0246 - val loss:=0.0072, train acc:=0.86 - val acc:=1.00
# Epoch:=2/10 - train loss:=0.0064 - val loss:=0.0053, train acc:=0.94 - val acc:=1.00
# Epoch:=3/10 - train loss:=0.0057 - val loss:=0.0047, train acc:=0.96 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0057 - val loss:=0.0054, train acc:=0.96 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0048 - val loss:=0.0042, train acc:=0.96 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0044 - val loss:=0.0050, train acc:=0.96 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0042 - val loss:=0.0050, train acc:=1.00 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0042 - val loss:=0.0069, train acc:=0.98 - val acc:=0.96
# Epoch:=9/10 - train loss:=0.0046 - val loss:=0.0060, train acc:=0.98 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0042 - val loss:=0.0052, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 200.83

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.00%



Count: 6, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 4112 *****

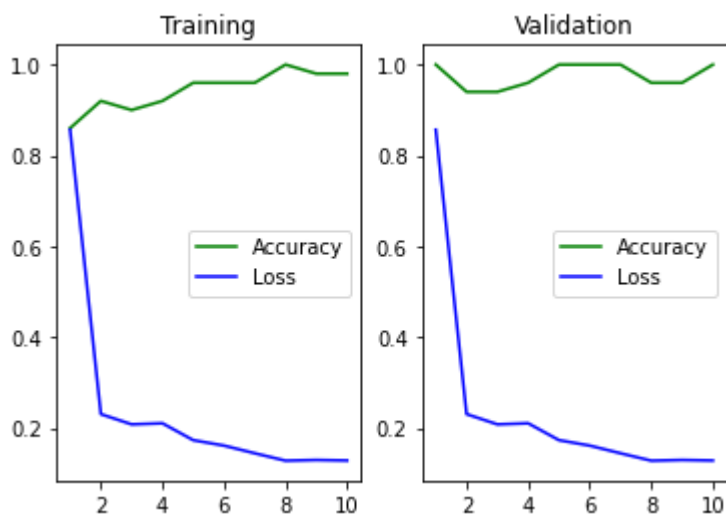
```
# Epoch:=1/10 - train loss:=0.0228 - val loss:=0.0075, train acc:=0.86 - val acc:=1.00
# Epoch:=2/10 - train loss:=0.0061 - val loss:=0.0051, train acc:=0.92 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0055 - val loss:=0.0050, train acc:=0.90 - val acc:=0.94
# Epoch:=4/10 - train loss:=0.0056 - val loss:=0.0049, train acc:=0.92 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0046 - val loss:=0.0057, train acc:=0.96 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0043 - val loss:=0.0055, train acc:=0.96 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0038 - val loss:=0.0043, train acc:=0.96 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0034 - val loss:=0.0049, train acc:=1.00 - val acc:=0.96
# Epoch:=9/10 - train loss:=0.0034 - val loss:=0.0051, train acc:=0.98 - val acc:=0.96
# Epoch:=10/10 - train loss:=0.0034 - val loss:=0.0051, train acc:=0.98 - val acc:=1.00
```

Total time taken (in seconds): 195.87

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.65%



Count: 7, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 9988 *****

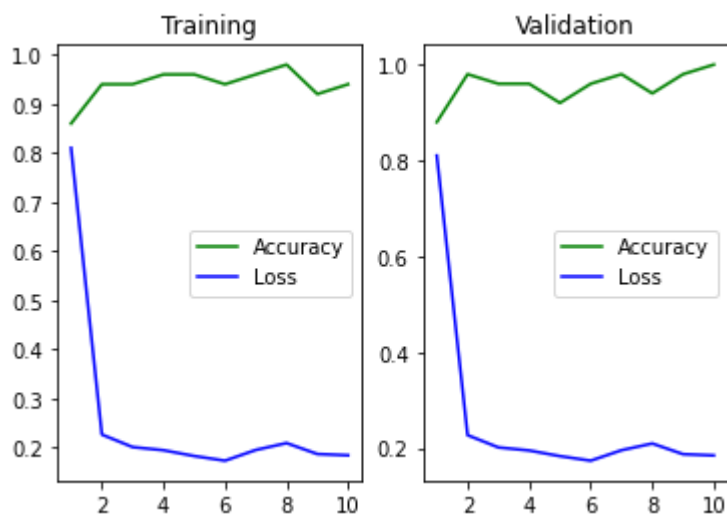
```
# Epoch:=1/10 - train loss:=0.0222 - val loss:=0.0074, train acc:=0.86 - val acc:=0.88
# Epoch:=2/10 - train loss:=0.0062 - val loss:=0.0060, train acc:=0.94 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0055 - val loss:=0.0048, train acc:=0.94 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0053 - val loss:=0.0079, train acc:=0.96 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0050 - val loss:=0.0050, train acc:=0.96 - val acc:=0.92
# Epoch:=6/10 - train loss:=0.0047 - val loss:=0.0045, train acc:=0.94 - val acc:=0.96
# Epoch:=7/10 - train loss:=0.0053 - val loss:=0.0075, train acc:=0.96 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0057 - val loss:=0.0059, train acc:=0.98 - val acc:=0.94
# Epoch:=9/10 - train loss:=0.0051 - val loss:=0.0052, train acc:=0.92 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0050 - val loss:=0.0055, train acc:=0.94 - val acc:=1.00
```

Total time taken (in seconds): 193.44

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.24%



Count: 8, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 9321 *****

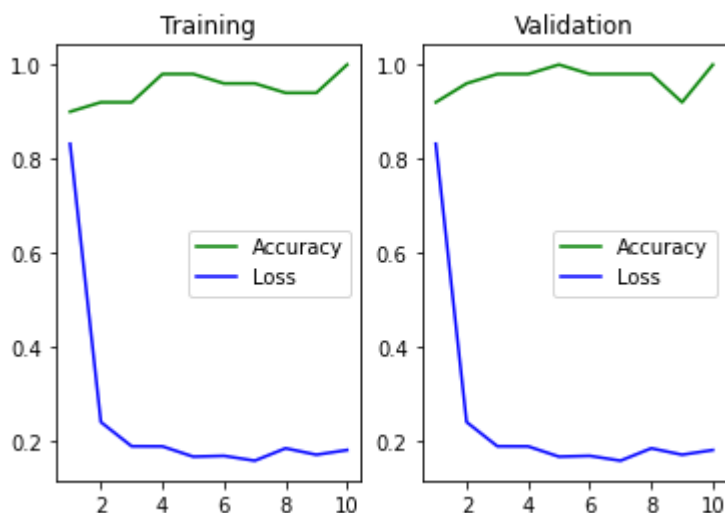
```
# Epoch:=1/10 - train loss:=0.0239 - val loss:=0.0079, train acc:=0.90 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0069 - val loss:=0.0050, train acc:=0.92 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0054 - val loss:=0.0048, train acc:=0.92 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0054 - val loss:=0.0045, train acc:=0.98 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0048 - val loss:=0.0048, train acc:=0.98 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0049 - val loss:=0.0043, train acc:=0.96 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0046 - val loss:=0.0058, train acc:=0.96 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0053 - val loss:=0.0051, train acc:=0.94 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0049 - val loss:=0.0076, train acc:=0.94 - val acc:=0.92
# Epoch:=10/10 - train loss:=0.0052 - val loss:=0.0073, train acc:=1.00 - val acc:=1.00
```

Total time taken (in seconds): 195.66

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 94.81%



Count: 9, j=: 0

***** Training model: mlp_on_gpu_RegL1 with optimizer: RMSprop and seed: 9558 *****

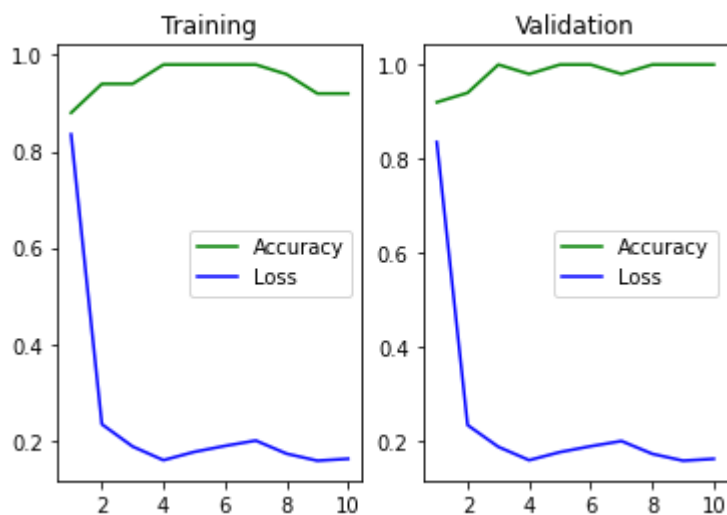
```
# Epoch:=1/10 - train loss:=0.0230 - val loss:=0.0077, train acc:=0.88 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0064 - val loss:=0.0060, train acc:=0.94 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0052 - val loss:=0.0040, train acc:=0.94 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0044 - val loss:=0.0040, train acc:=0.98 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0049 - val loss:=0.0051, train acc:=0.98 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0052 - val loss:=0.0048, train acc:=0.98 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0055 - val loss:=0.0052, train acc:=0.98 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0048 - val loss:=0.0067, train acc:=0.96 - val acc:=1.00
# Epoch:=9/10 - train loss:=0.0044 - val loss:=0.0048, train acc:=0.92 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0045 - val loss:=0.0050, train acc:=0.92 - val acc:=1.00
```

Total time taken (in seconds): 193.60

Finished training model: mlp_on_gpu_RegL1

***** Testing *****

mlp_on_gpu_RegL1 model accuracy = 96.15%



Count: 0, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 7130 *****

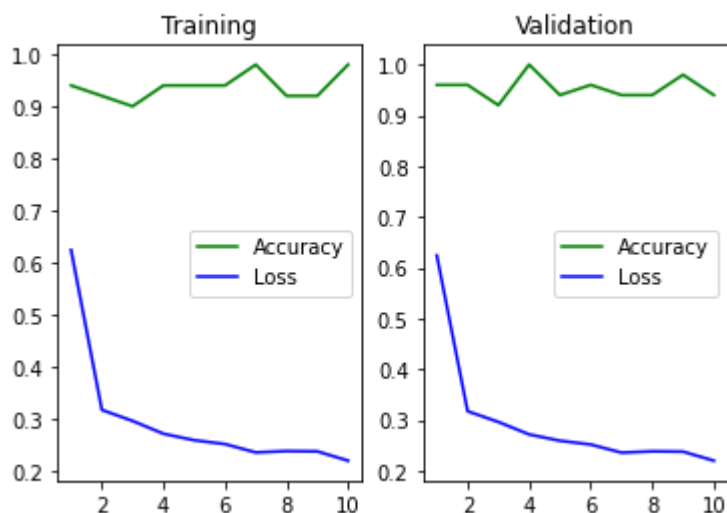
```
# Epoch:=1/10 - train loss:=0.0150 - val loss:=0.0078, train acc:=0.94 - val acc:=0.96
# Epoch:=2/10 - train loss:=0.0076 - val loss:=0.0060, train acc:=0.92 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0071 - val loss:=0.0068, train acc:=0.90 - val acc:=0.92
# Epoch:=4/10 - train loss:=0.0065 - val loss:=0.0050, train acc:=0.94 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0062 - val loss:=0.0072, train acc:=0.94 - val acc:=0.94
# Epoch:=6/10 - train loss:=0.0060 - val loss:=0.0063, train acc:=0.94 - val acc:=0.96
# Epoch:=7/10 - train loss:=0.0057 - val loss:=0.0074, train acc:=0.98 - val acc:=0.94
# Epoch:=8/10 - train loss:=0.0057 - val loss:=0.0070, train acc:=0.92 - val acc:=0.94
# Epoch:=9/10 - train loss:=0.0057 - val loss:=0.0048, train acc:=0.92 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0053 - val loss:=0.0061, train acc:=0.98 - val acc:=0.94
```

Total time taken (in seconds): 194.76

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 92.26%



Count: 1, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 3984 *****

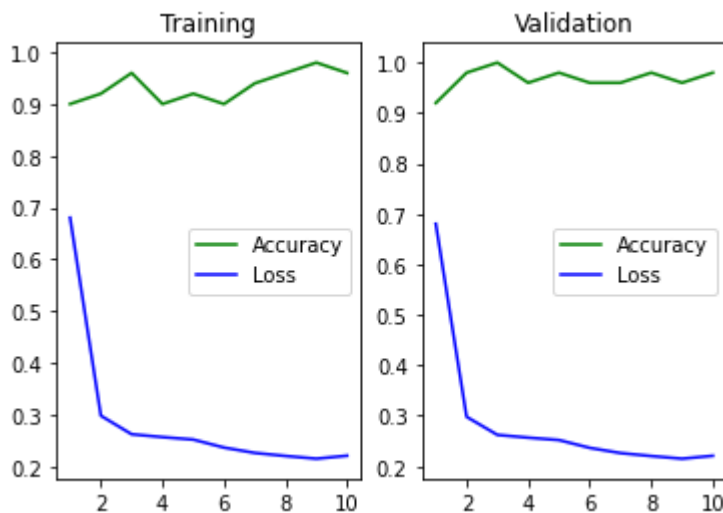
```
# Epoch:=1/10 - train loss:=0.0165 - val loss:=0.0085, train acc:=0.90 - val acc:=0.92
# Epoch:=2/10 - train loss:=0.0072 - val loss:=0.0065, train acc:=0.92 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0063 - val loss:=0.0049, train acc:=0.96 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0062 - val loss:=0.0055, train acc:=0.90 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0061 - val loss:=0.0053, train acc:=0.92 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0057 - val loss:=0.0051, train acc:=0.90 - val acc:=0.96
# Epoch:=7/10 - train loss:=0.0055 - val loss:=0.0051, train acc:=0.94 - val acc:=0.96
# Epoch:=8/10 - train loss:=0.0053 - val loss:=0.0055, train acc:=0.96 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0052 - val loss:=0.0063, train acc:=0.98 - val acc:=0.96
# Epoch:=10/10 - train loss:=0.0053 - val loss:=0.0049, train acc:=0.96 - val acc:=0.98
```

Total time taken (in seconds): 195.02

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 93.76%



Count: 2, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 2366 *****

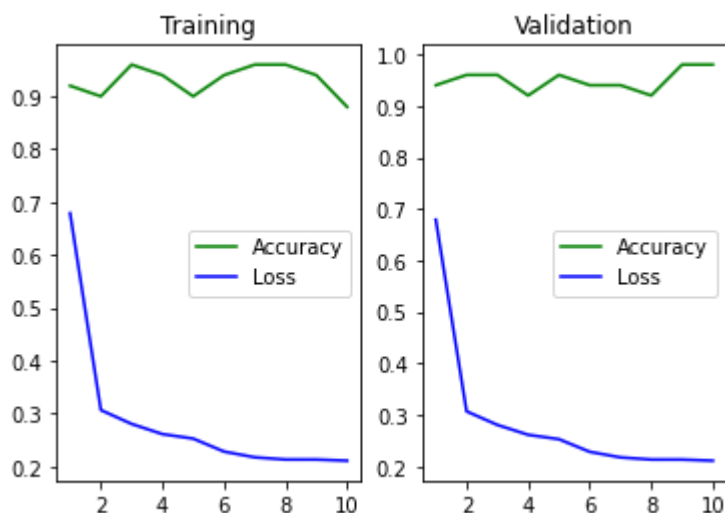
```
# Epoch:=1/10 - train loss:=0.0163 - val loss:=0.0077, train acc:=0.92 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0074 - val loss:=0.0065, train acc:=0.90 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0067 - val loss:=0.0072, train acc:=0.96 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0063 - val loss:=0.0056, train acc:=0.94 - val acc:=0.92
# Epoch:=5/10 - train loss:=0.0061 - val loss:=0.0050, train acc:=0.90 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0055 - val loss:=0.0049, train acc:=0.94 - val acc:=0.94
# Epoch:=7/10 - train loss:=0.0052 - val loss:=0.0047, train acc:=0.96 - val acc:=0.94
# Epoch:=8/10 - train loss:=0.0051 - val loss:=0.0052, train acc:=0.96 - val acc:=0.92
# Epoch:=9/10 - train loss:=0.0051 - val loss:=0.0045, train acc:=0.94 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0051 - val loss:=0.0051, train acc:=0.88 - val acc:=0.98
```

Total time taken (in seconds): 194.65

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 93.50%



Count: 3, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 1850 *****

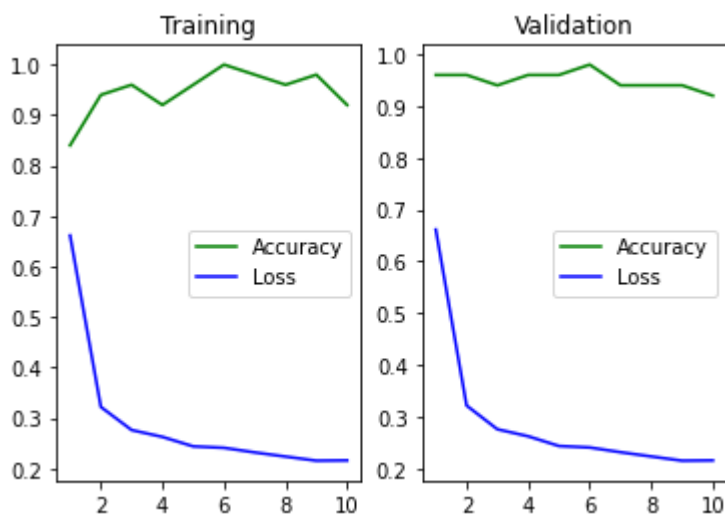
```
# Epoch:=1/10 - train loss:=0.0157 - val loss:=0.0091, train acc:=0.84 - val acc:=0.96
# Epoch:=2/10 - train loss:=0.0076 - val loss:=0.0072, train acc:=0.94 - val acc:=0.96
# Epoch:=3/10 - train loss:=0.0065 - val loss:=0.0051, train acc:=0.96 - val acc:=0.94
# Epoch:=4/10 - train loss:=0.0062 - val loss:=0.0056, train acc:=0.92 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0058 - val loss:=0.0049, train acc:=0.96 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0057 - val loss:=0.0057, train acc:=1.00 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0055 - val loss:=0.0046, train acc:=0.98 - val acc:=0.94
# Epoch:=8/10 - train loss:=0.0053 - val loss:=0.0049, train acc:=0.96 - val acc:=0.94
# Epoch:=9/10 - train loss:=0.0051 - val loss:=0.0061, train acc:=0.98 - val acc:=0.94
# Epoch:=10/10 - train loss:=0.0051 - val loss:=0.0060, train acc:=0.92 - val acc:=0.92
```

Total time taken (in seconds): 205.48

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 92.02%



Count: 4, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 1562 *****

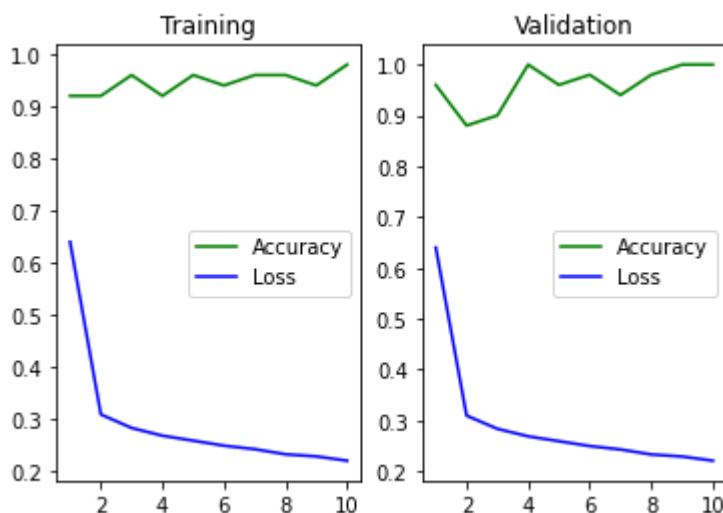
```
# Epoch:=1/10 - train loss:=0.0149 - val loss:=0.0077, train acc:=0.92 - val acc:=0.96
# Epoch:=2/10 - train loss:=0.0072 - val loss:=0.0090, train acc:=0.92 - val acc:=0.88
# Epoch:=3/10 - train loss:=0.0066 - val loss:=0.0060, train acc:=0.96 - val acc:=0.90
# Epoch:=4/10 - train loss:=0.0062 - val loss:=0.0072, train acc:=0.92 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0060 - val loss:=0.0065, train acc:=0.96 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0058 - val loss:=0.0072, train acc:=0.94 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0056 - val loss:=0.0050, train acc:=0.96 - val acc:=0.94
# Epoch:=8/10 - train loss:=0.0054 - val loss:=0.0047, train acc:=0.96 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0053 - val loss:=0.0051, train acc:=0.94 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0051 - val loss:=0.0048, train acc:=0.98 - val acc:=1.00
```

Total time taken (in seconds): 202.82

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 93.50%



Count: 5, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 4183 *****

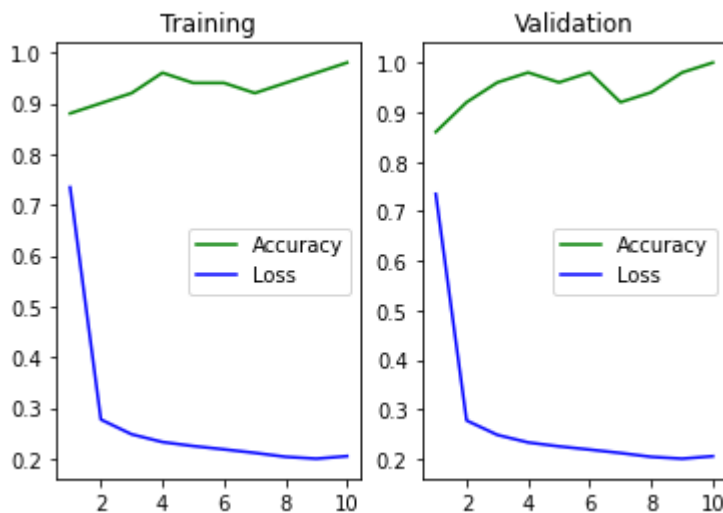
```
# Epoch:=1/10 - train loss:=0.0200 - val loss:=0.0084, train acc:=0.88 - val acc:=0.86
# Epoch:=2/10 - train loss:=0.0076 - val loss:=0.0059, train acc:=0.90 - val acc:=0.92
# Epoch:=3/10 - train loss:=0.0068 - val loss:=0.0073, train acc:=0.92 - val acc:=0.96
# Epoch:=4/10 - train loss:=0.0064 - val loss:=0.0054, train acc:=0.96 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0061 - val loss:=0.0058, train acc:=0.94 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0060 - val loss:=0.0065, train acc:=0.94 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0058 - val loss:=0.0057, train acc:=0.92 - val acc:=0.92
# Epoch:=8/10 - train loss:=0.0056 - val loss:=0.0050, train acc:=0.94 - val acc:=0.94
# Epoch:=9/10 - train loss:=0.0055 - val loss:=0.0058, train acc:=0.96 - val acc:=0.98
# Epoch:=10/10 - train loss:=0.0056 - val loss:=0.0049, train acc:=0.98 - val acc:=1.00
```

Total time taken (in seconds): 205.89

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 93.75%



Count: 6, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 7845 *****

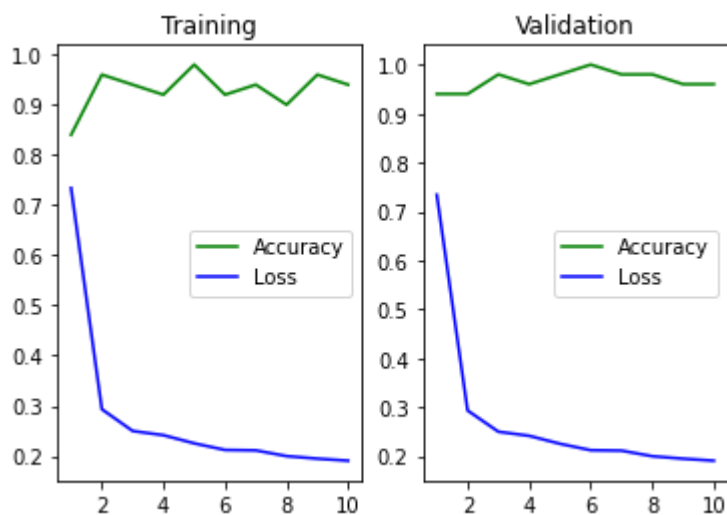
```
# Epoch:=1/10 - train loss:=0.0192 - val loss:=0.0077, train acc:=0.84 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0077 - val loss:=0.0058, train acc:=0.96 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0065 - val loss:=0.0059, train acc:=0.94 - val acc:=0.98
# Epoch:=4/10 - train loss:=0.0063 - val loss:=0.0059, train acc:=0.92 - val acc:=0.96
# Epoch:=5/10 - train loss:=0.0059 - val loss:=0.0044, train acc:=0.98 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0055 - val loss:=0.0055, train acc:=0.92 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0055 - val loss:=0.0045, train acc:=0.94 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0052 - val loss:=0.0047, train acc:=0.90 - val acc:=0.98
# Epoch:=9/10 - train loss:=0.0051 - val loss:=0.0049, train acc:=0.96 - val acc:=0.96
# Epoch:=10/10 - train loss:=0.0050 - val loss:=0.0055, train acc:=0.94 - val acc:=0.96
```

Total time taken (in seconds): 200.64

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 92.59%



Count: 7, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 1700 *****

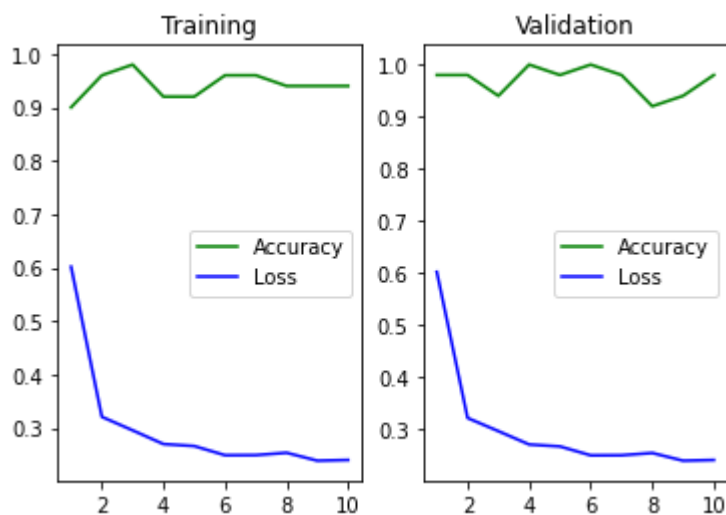
```
# Epoch:=1/10 - train loss:=0.0138 - val loss:=0.0067, train acc:=0.9
0 - val acc:=0.98
# Epoch:=2/10 - train loss:=0.0074 - val loss:=0.0055, train acc:=0.9
6 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0068 - val loss:=0.0061, train acc:=0.9
8 - val acc:=0.94
# Epoch:=4/10 - train loss:=0.0062 - val loss:=0.0058, train acc:=0.9
2 - val acc:=1.00
# Epoch:=5/10 - train loss:=0.0061 - val loss:=0.0053, train acc:=0.9
2 - val acc:=0.98
# Epoch:=6/10 - train loss:=0.0057 - val loss:=0.0054, train acc:=0.9
6 - val acc:=1.00
# Epoch:=7/10 - train loss:=0.0057 - val loss:=0.0049, train acc:=0.9
6 - val acc:=0.98
# Epoch:=8/10 - train loss:=0.0058 - val loss:=0.0055, train acc:=0.9
4 - val acc:=0.92
# Epoch:=9/10 - train loss:=0.0055 - val loss:=0.0060, train acc:=0.9
4 - val acc:=0.94
# Epoch:=10/10 - train loss:=0.0055 - val loss:=0.0067, train acc:=0.
94 - val acc:=0.98
```

Total time taken (in seconds): 200.24

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 92.01%



Count: 8, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 2161 *****

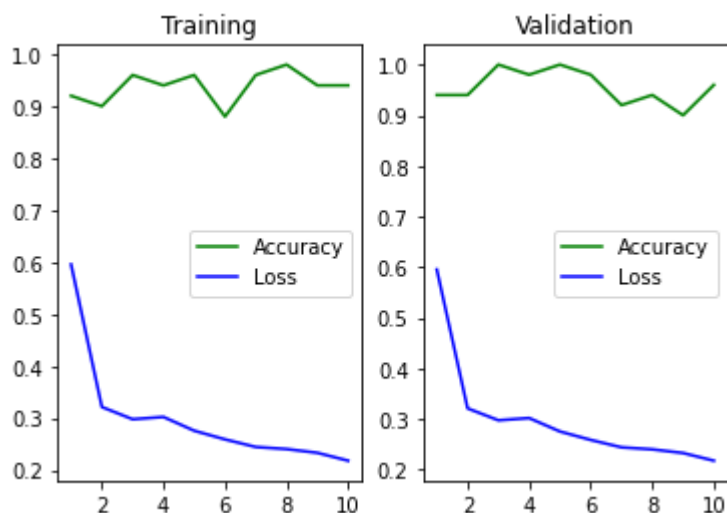
```
# Epoch:=1/10 - train loss:=0.0136 - val loss:=0.0063, train acc:=0.92 - val acc:=0.94
# Epoch:=2/10 - train loss:=0.0073 - val loss:=0.0066, train acc:=0.90 - val acc:=0.94
# Epoch:=3/10 - train loss:=0.0068 - val loss:=0.0056, train acc:=0.96 - val acc:=1.00
# Epoch:=4/10 - train loss:=0.0069 - val loss:=0.0055, train acc:=0.94 - val acc:=0.98
# Epoch:=5/10 - train loss:=0.0063 - val loss:=0.0055, train acc:=0.96 - val acc:=1.00
# Epoch:=6/10 - train loss:=0.0059 - val loss:=0.0054, train acc:=0.88 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0056 - val loss:=0.0059, train acc:=0.96 - val acc:=0.92
# Epoch:=8/10 - train loss:=0.0055 - val loss:=0.0055, train acc:=0.98 - val acc:=0.94
# Epoch:=9/10 - train loss:=0.0053 - val loss:=0.0043, train acc:=0.94 - val acc:=0.90
# Epoch:=10/10 - train loss:=0.0050 - val loss:=0.0046, train acc:=0.94 - val acc:=0.96
```

Total time taken (in seconds): 196.67

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 93.53%



Count: 9, j=: 0

***** Training model: mlp_on_gpu_RegL2 with optimizer: RMSprop and seed: 6409 *****

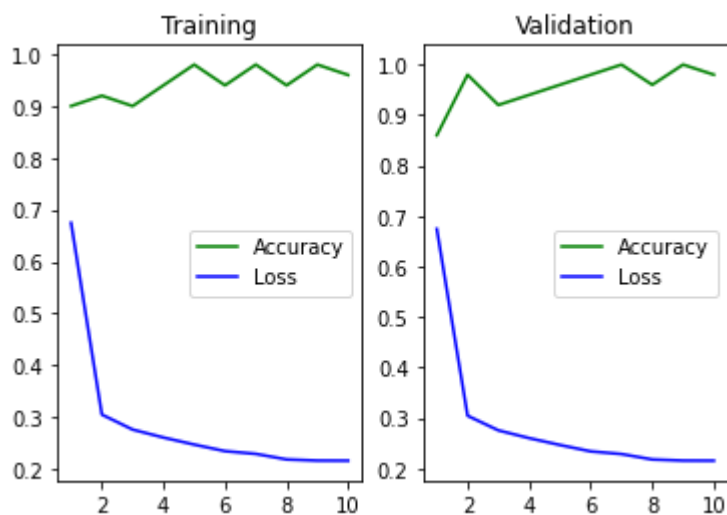
```
# Epoch:=1/10 - train loss:=0.0166 - val loss:=0.0097, train acc:=0.90 - val acc:=0.86
# Epoch:=2/10 - train loss:=0.0075 - val loss:=0.0067, train acc:=0.92 - val acc:=0.98
# Epoch:=3/10 - train loss:=0.0068 - val loss:=0.0076, train acc:=0.90 - val acc:=0.92
# Epoch:=4/10 - train loss:=0.0064 - val loss:=0.0079, train acc:=0.94 - val acc:=0.94
# Epoch:=5/10 - train loss:=0.0061 - val loss:=0.0056, train acc:=0.98 - val acc:=0.96
# Epoch:=6/10 - train loss:=0.0058 - val loss:=0.0059, train acc:=0.94 - val acc:=0.98
# Epoch:=7/10 - train loss:=0.0056 - val loss:=0.0056, train acc:=0.98 - val acc:=1.00
# Epoch:=8/10 - train loss:=0.0054 - val loss:=0.0058, train acc:=0.94 - val acc:=0.96
# Epoch:=9/10 - train loss:=0.0053 - val loss:=0.0044, train acc:=0.98 - val acc:=1.00
# Epoch:=10/10 - train loss:=0.0053 - val loss:=0.0053, train acc:=0.96 - val acc:=0.98
```

Total time taken (in seconds): 195.41

Finished training model: mlp_on_gpu_RegL2

***** Testing *****

mlp_on_gpu_RegL2 model accuracy = 92.91%



UnboundLocalError Traceback (most recent call last)

/tmp/ipykernel_4762/2116875660.py in <module>

```
1 if __name__ == "__main__":  
----> 2     mnist, fashion_mnist = main()
```

/tmp/ipykernel_4762/2267604138.py in main()

```
89  
90  
---> 91     return mnist, fashion_mnist
```

UnboundLocalError: local variable 'fashion_mnist' referenced before assignment

In []: