Comparision of Optimizers

Fashion MNIST

The datset we will use to start is the Fashion MNIST dataset. This dataset contains 60000 images of different clothing items.

In [ ]:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```
data = keras.datasets.fashion_mnist #loading the dataset
```

In [ ]:

```
(train_images, train_labels), (test_images, test_labels) = data.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-labels-idx1-ubyte.gz
16384/5148 [===============================================================
================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

In [ ]:

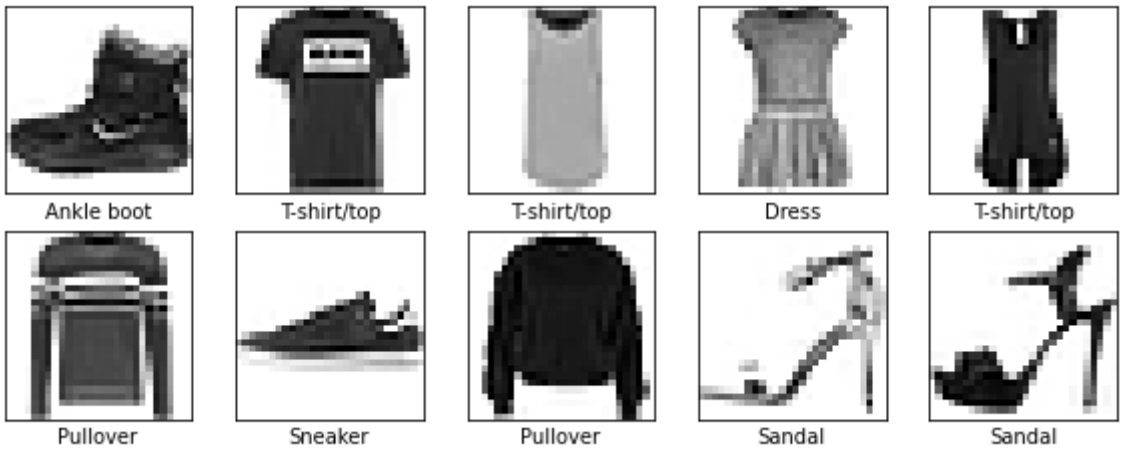
```
#assigning classes

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Pre-processing images

We do this by dividing each image by 255. Since each image is greyscale we are simply scaling the pixel values down to make computations easier for our model.

In [ ]:

```
train_images = train_images/255.0
test_images = test_images/255.0

plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```



Creating Models

Adam Optimizer

In [ ]:

```
modeladm = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

RMS optimizer

In [ ]:

```
modelrms = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

Adagrad optimizer

In [ ]:

```
modeladgrad = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

SGD optimizers

In [ ]:

```
modelsgd = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

SGDM optimizer

In [ ]:

```
modelsgdm = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10)
])
```

Compiling the model with diffrent optimizers

In [ ]:

```
modeladm.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
modelrms.compile(optimizer='RMSprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
modeladgrad.compile(optimizer='adagrad',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
modelsgd.compile(optimizer='SGD',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
opt = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9)
modelsgdm.compile(optimizer=opt,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Fitting the model

SGD

In [ ]:

```
hist4=modelsgd.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7362 -
accuracy: 0.7665
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5079 -
accuracy: 0.8269
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4645 -
accuracy: 0.8404
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4400 -
accuracy: 0.8487
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4215 -
accuracy: 0.8548
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4079 -
accuracy: 0.8596
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3960 -
accuracy: 0.8634
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3869 -
accuracy: 0.8655
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3785 -
accuracy: 0.8679
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3711 -
accuracy: 0.8708
```

SGDM

In [ ]:

```
hist5=modelsgdm.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7853 -
accuracy: 0.7133
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7553 -
accuracy: 0.7110
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7148 -
accuracy: 0.7350
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6829 -
accuracy: 0.7499
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7145 -
accuracy: 0.7428
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6809 -
accuracy: 0.7466
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7253 -
accuracy: 0.7313
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6448 -
accuracy: 0.7670
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6561 -
accuracy: 0.7565
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6462 -
accuracy: 0.7710
```

Adgrad

In [ ]:

```
hist3=modeladgrad.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 5s 2ms/step - loss: 1.0645 -
accuracy: 0.6781
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7186 -
accuracy: 0.7706
Epoch 3/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.6421 -
accuracy: 0.7951
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6014 -
accuracy: 0.8071
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5744 -
accuracy: 0.8154
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5550 -
accuracy: 0.8207
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5396 -
accuracy: 0.8247
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.5277 -
accuracy: 0.8280
Epoch 9/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.5178 -
accuracy: 0.8305
Epoch 10/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.5092 -
accuracy: 0.8326
```

RMS prop

In [ ]:

```
hist2=modelrms.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.5065 -
accuracy: 0.8194
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3788 -
accuracy: 0.8645
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3446 -
accuracy: 0.8782
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3302 -
accuracy: 0.8829
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3153 -
accuracy: 0.8897
Epoch 6/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3076 -
accuracy: 0.8931
Epoch 7/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2967 -
accuracy: 0.8970
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2930 -
accuracy: 0.9000
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2856 -
accuracy: 0.9016
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2787 -
accuracy: 0.9033
```

Adam

In [ ]:

```
hist1=modeladm.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.4985 -
accuracy: 0.8242
Epoch 2/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.3745 -
accuracy: 0.8652
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3377 -
accuracy: 0.8774
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3145 -
accuracy: 0.8844
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2955 -
accuracy: 0.8909
Epoch 6/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2820 -
accuracy: 0.8960
Epoch 7/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2693 -
accuracy: 0.9005
Epoch 8/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2577 -
accuracy: 0.9047
Epoch 9/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2479 -
accuracy: 0.9078
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2406 -
accuracy: 0.9098
```

Plotting accuracies with respect to epochs

In [ ]:

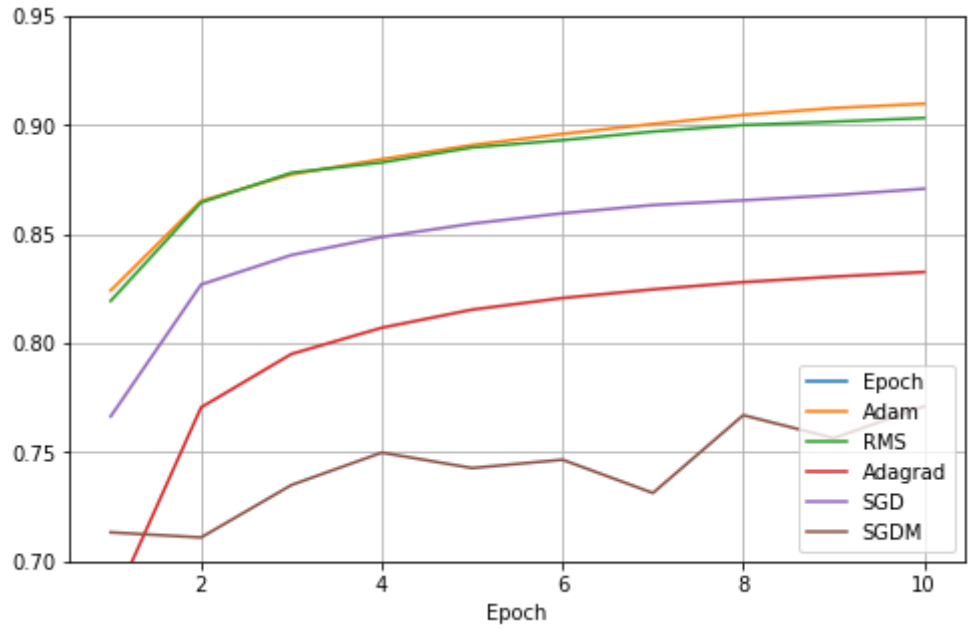
```
ep=np.arange(1,11,1)

acc1=hist1.history['accuracy']
acc2=hist2.history['accuracy']
acc3=hist3.history['accuracy']
acc4=hist4.history['accuracy']
acc5=hist5.history['accuracy']
list_of_tuples = list(zip(ep,acc1,acc2,acc3,acc4,acc5))
```

In [ ]:

```
df = pd.DataFrame(list_of_tuples, columns = ['Epoch', 'Adam','RMS','Adagrad','SGD','SGD
M'])
```

In [ ]:

```
df.index = df['Epoch']
df.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(.7, .95) # set the vertical range to [0-1]
plt.show()
```



Validation using test set

In [ ]:

```
modeladm.evaluate(test_images, test_labels)
modelrms.evaluate(test_images, test_labels)
modeladgrad.evaluate(test_images, test_labels)
modelsgd.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3500 - ac
curacy: 0.8747
313/313 [==============================] - 1s 2ms/step - loss: 0.3881 - ac
curacy: 0.8815
313/313 [==============================] - 1s 2ms/step - loss: 0.5320 - ac
curacy: 0.8174
313/313 [==============================] - 1s 3ms/step - loss: 0.4090 - ac
curacy: 0.8554
```

Out[ ]:

```
[0.40901100635528564, 0.855400025844574]
```