Assignment No. 5

Aim: Write a program to demonstrate the change in accuracy/loss/convergence time with change in optimizers like stochastic gradient descent, adam, adagrad, RMSprop and Nadam for any suitable application

Objectives:

1. To learn optimization algorithms

2. To learn and understand hyperparameters

Theory:

SGD, Adam, RMSprop, Nadam

The word '*stochastic*'means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy and less random manner, but the problem arises when our datasets gets big.

Adam is a replacement optimization algorithm for stochastic gradient descent for training **deep learning** models. **Adam** combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster. The difference between RMSprop and gradient descent is on how the gradients are calculated. The following equations show how the gradients are calculated for the RMSprop and gradient descent with momentum. The value of momentum is denoted by beta and is usually set to 0.9.

Nadam combines NAG and Adam. Nadam is employed for noisy gradients or for gradients with high curvatures. The learning process is accelerated by summing up the exponential decay of the moving averages for the previous and current gradient

Code:

```
import tensorflow as tf
from tensorflow.keras import layers,models,datasets
from tensorflow.keras.applications.vgg16 import VGG16
```

```
(train_images,train_labels),(test_images,test_labels)=datasets.cifar100.load_d
ata()
train_images=train_images/255
test_images=test_images/255


base=VGG16(include_top=False,input_shape=(32,32,3))
base.trainable=False
model=models.Sequential()
model.add(layers.Flatten())
model.add(layers.Dense(1200,activation="relu"))
model.add(layers.Dense(100,activation="softmax"))
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=
["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=200)

model.compile(optimizer="sgd",loss="sparse_categorical_crossentropy",metrics=[
"accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=50)

model.compile(optimizer="adagrad",loss="sparse_categorical_crossentropy",metri
cs=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=50)

model.compile(optimizer="rmsprop",loss="sparse_categorical_crossentropy",metri
cs=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=50)

model.compile(optimizer="sgd",loss="sparse_categorical_crossentropy",metrics=[
"accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=50)

model.compile(optimizer="nadam",loss="sparse_categorical_crossentropy",metrics
=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels),steps_per_epoch=50)
```

Results:

```python
model.compile(optimizer="rmsprop",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels),steps_per_epoch=50)
```

```
Epoch 1/10
50/50 [==============================] - 8s 165ms/step - loss: 3.2918 - accuracy: 0.2295 - val_loss: 3.4460 - val_accuracy: 0.2053
Epoch 2/10
50/50 [==============================] - 8s 163ms/step - loss: 3.1533 - accuracy: 0.2452 - val_loss: 3.3486 - val_accuracy: 0.2210
Epoch 3/10
50/50 [==============================] - 8s 159ms/step - loss: 3.1276 - accuracy: 0.2473 - val_loss: 3.3681 - val_accuracy: 0.2141
Epoch 4/10
50/50 [==============================] - 8s 163ms/step - loss: 3.1184 - accuracy: 0.2512 - val_loss: 3.3351 - val_accuracy: 0.2253
Epoch 5/10
50/50 [==============================] - 9s 174ms/step - loss: 3.1034 - accuracy: 0.2523 - val_loss: 3.4193 - val_accuracy: 0.2031
Epoch 6/10
50/50 [==============================] - 8s 167ms/step - loss: 3.0861 - accuracy: 0.2577 - val_loss: 3.3914 - val_accuracy: 0.2155
Epoch 7/10
50/50 [==============================] - 8s 152ms/step - loss: 3.0775 - accuracy: 0.2616 - val_loss: 3.3460 - val_accuracy: 0.2236
Epoch 8/10
50/50 [==============================] - 8s 158ms/step - loss: 3.0583 - accuracy: 0.2630 - val_loss: 3.3535 - val_accuracy: 0.2227
Epoch 9/10
50/50 [==============================] - 8s 156ms/step - loss: 3.0469 - accuracy: 0.2653 - val_loss: 3.3743 - val_accuracy: 0.2155
Epoch 10/10
50/50 [==============================] - 7s 149ms/step - loss: 3.0225 - accuracy: 0.2699 - val_loss: 3.4018 - val_accuracy: 0.2176

<tensorflow.python.keras.callbacks.History at 0x14900ffe588>
```

```python
model.compile(optimizer="adagrad",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels),steps_per_epoch=50)
```

```
Epoch 1/10
50/50 [==============================] - 6s 127ms/step - loss: 2.9720 - accuracy: 0.2847 - val_loss: 3.2421 - val_accuracy: 0.2382
Epoch 2/10
50/50 [==============================] - 7s 132ms/step - loss: 2.9712 - accuracy: 0.2851 - val_loss: 3.2420 - val_accuracy: 0.2389
Epoch 3/10
50/50 [==============================] - 6s 128ms/step - loss: 2.9706 - accuracy: 0.2854 - val_loss: 3.2418 - val_accuracy: 0.2390
Epoch 4/10
50/50 [==============================] - 6s 123ms/step - loss: 2.9700 - accuracy: 0.2852 - val_loss: 3.2417 - val_accuracy: 0.2383
Epoch 5/10
50/50 [==============================] - 7s 131ms/step - loss: 2.9695 - accuracy: 0.2851 - val_loss: 3.2415 - val_accuracy: 0.2386
Epoch 6/10
50/50 [==============================] - 7s 133ms/step - loss: 2.9690 - accuracy: 0.2853 - val_loss: 3.2413 - val_accuracy: 0.2384
Epoch 7/10
50/50 [==============================] - 7s 134ms/step - loss: 2.9685 - accuracy: 0.2850 - val_loss: 3.2411 - val_accuracy: 0.2385
Epoch 8/10
50/50 [==============================] - 6s 124ms/step - loss: 2.9680 - accuracy: 0.2857 - val_loss: 3.2410 - val_accuracy: 0.2376
Epoch 9/10
50/50 [==============================] - 7s 134ms/step - loss: 2.9675 - accuracy: 0.2855 - val_loss: 3.2408 - val_accuracy: 0.2381
Epoch 10/10
50/50 [==============================] - 6s 125ms/step - loss: 2.9670 - accuracy: 0.2855 - val_loss: 3.2407 - val_accuracy: 0.2387

<tensorflow.python.keras.callbacks.History at 0x149000f03c8>
```

```python
model.compile(optimizer="sgd",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels),steps_per_epoch=50)
```

```
Epoch 1/10
50/50 [==============================] - 7s 136ms/step - loss: 3.0500 - accuracy: 0.2682 - val_loss: 3.2709 - val_accuracy: 0.2341
Epoch 2/10
50/50 [==============================] - 6s 114ms/step - loss: 3.0152 - accuracy: 0.2766 - val_loss: 3.2599 - val_accuracy: 0.2336
Epoch 3/10
50/50 [==============================] - 6s 112ms/step - loss: 3.0031 - accuracy: 0.2784 - val_loss: 3.2541 - val_accuracy: 0.2349
Epoch 4/10
50/50 [==============================] - 5s 110ms/step - loss: 2.9955 - accuracy: 0.2802 - val_loss: 3.2503 - val_accuracy: 0.2384
Epoch 5/10
50/50 [==============================] - 6s 116ms/step - loss: 2.9901 - accuracy: 0.2808 - val_loss: 3.2479 - val_accuracy: 0.2389
Epoch 6/10
50/50 [==============================] - 6s 125ms/step - loss: 2.9859 - accuracy: 0.2823 - val_loss: 3.2465 - val_accuracy: 0.2368
Epoch 7/10
50/50 [==============================] - 6s 118ms/step - loss: 2.9828 - accuracy: 0.2820 - val_loss: 3.2453 - val_accuracy: 0.2391
Epoch 8/10
50/50 [==============================] - 6s 117ms/step - loss: 2.9800 - accuracy: 0.2834 - val_loss: 3.2443 - val_accuracy: 0.2390
Epoch 9/10
50/50 [==============================] - 6s 116ms/step - loss: 2.9776 - accuracy: 0.2835 - val_loss: 3.2436 - val_accuracy: 0.2372
Epoch 10/10
50/50 [==============================] - 6s 114ms/step - loss: 2.9756 - accuracy: 0.2843 - val_loss: 3.2427 - val_accuracy: 0.2385

<tensorflow.python.keras.callbacks.History at 0x1496eb90b08>
```

```
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels),steps_per_epoch=200)
```

```
Epoch 1/10
200/200 [==============================] - 22s 109ms/step - loss: 4.2006 - accuracy: 0.0757 - val_loss: 3.8973 - val_accuracy: 0.1107
Epoch 2/10
200/200 [==============================] - 23s 117ms/step - loss: 3.7537 - accuracy: 0.1343 - val_loss: 3.7244 - val_accuracy: 0.1415
Epoch 3/10
200/200 [==============================] - 21s 107ms/step - loss: 3.6068 - accuracy: 0.1607 - val_loss: 3.6081 - val_accuracy: 0.1604
Epoch 4/10
200/200 [==============================] - 22s 111ms/step - loss: 3.5108 - accuracy: 0.1801 - val_loss: 3.5507 - val_accuracy: 0.1779
Epoch 5/10
200/200 [==============================] - 23s 113ms/step - loss: 3.4321 - accuracy: 0.1927 - val_loss: 3.5164 - val_accuracy: 0.1831
Epoch 6/10
200/200 [==============================] - 23s 113ms/step - loss: 3.3603 - accuracy: 0.2066 - val_loss: 3.4489 - val_accuracy: 0.1994
Epoch 7/10
200/200 [==============================] - 21s 107ms/step - loss: 3.3046 - accuracy: 0.2150 - val_loss: 3.4260 - val_accuracy: 0.1993
Epoch 8/10
200/200 [==============================] - 20s 102ms/step - loss: 3.2435 - accuracy: 0.2288 - val_loss: 3.4075 - val_accuracy: 0.2041
Epoch 9/10
200/200 [==============================] - 18s 89ms/step - loss: 3.2154 - accuracy: 0.2325 - val_loss: 3.3717 - val_accuracy: 0.2109
Epoch 10/10
200/200 [==============================] - 14s 68ms/step - loss: 3.1621 - accuracy: 0.2409 - val_loss: 3.3724 - val_accuracy: 0.2089

<tensorflow.python.keras.callbacks.History at 0x1496e5bd848>
```

```
model.compile(optimizer="nadam",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels),steps_per_epoch=50)
```

```
Epoch 1/10
50/50 [==============================] - 10s 193ms/step - loss: 2.9010 - accuracy: 0.2938 - val_loss: 3.3019 - val_accuracy: 0.2345
Epoch 2/10
50/50 [==============================] - 9s 181ms/step - loss: 2.8911 - accuracy: 0.2963 - val_loss: 3.3082 - val_accuracy: 0.2314
Epoch 3/10
50/50 [==============================] - 9s 182ms/step - loss: 2.8688 - accuracy: 0.3027 - val_loss: 3.2694 - val_accuracy: 0.2442
Epoch 4/10
50/50 [==============================] - 9s 188ms/step - loss: 2.8464 - accuracy: 0.3037 - val_loss: 3.2894 - val_accuracy: 0.2374
Epoch 5/10
50/50 [==============================] - 10s 200ms/step - loss: 2.8303 - accuracy: 0.3091 - val_loss: 3.2336 - val_accuracy: 0.2472
Epoch 6/10
50/50 [==============================] - 10s 193ms/step - loss: 2.8139 - accuracy: 0.3141 - val_loss: 3.2741 - val_accuracy: 0.2391
Epoch 7/10
50/50 [==============================] - 10s 192ms/step - loss: 2.8008 - accuracy: 0.3150 - val_loss: 3.2407 - val_accuracy: 0.2472
Epoch 8/10
50/50 [==============================] - 10s 194ms/step - loss: 2.7790 - accuracy: 0.3185 - val_loss: 3.3214 - val_accuracy: 0.2366
Epoch 9/10
50/50 [==============================] - 10s 196ms/step - loss: 2.7694 - accuracy: 0.3205 - val_loss: 3.2575 - val_accuracy: 0.2456
Epoch 10/10
50/50 [==============================] - 10s 197ms/step - loss: 2.7341 - accuracy: 0.3288 - val_loss: 3.2599 - val_accuracy: 0.2506

<tensorflow.python.keras.callbacks.History at 0x149002eb108>
```

Conclusion:

Thus, we have understood the difference in performance of various optimisation algorithms