**Shruti Karpe**

**2213182**

**AIA-1-A**

**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)

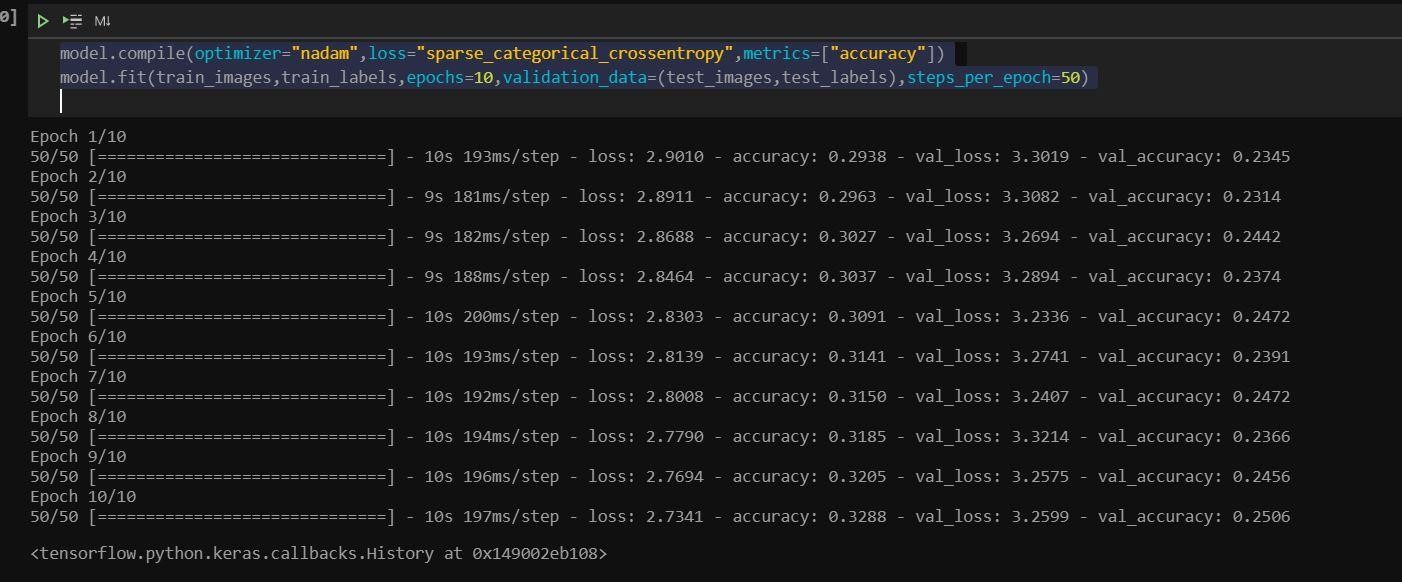
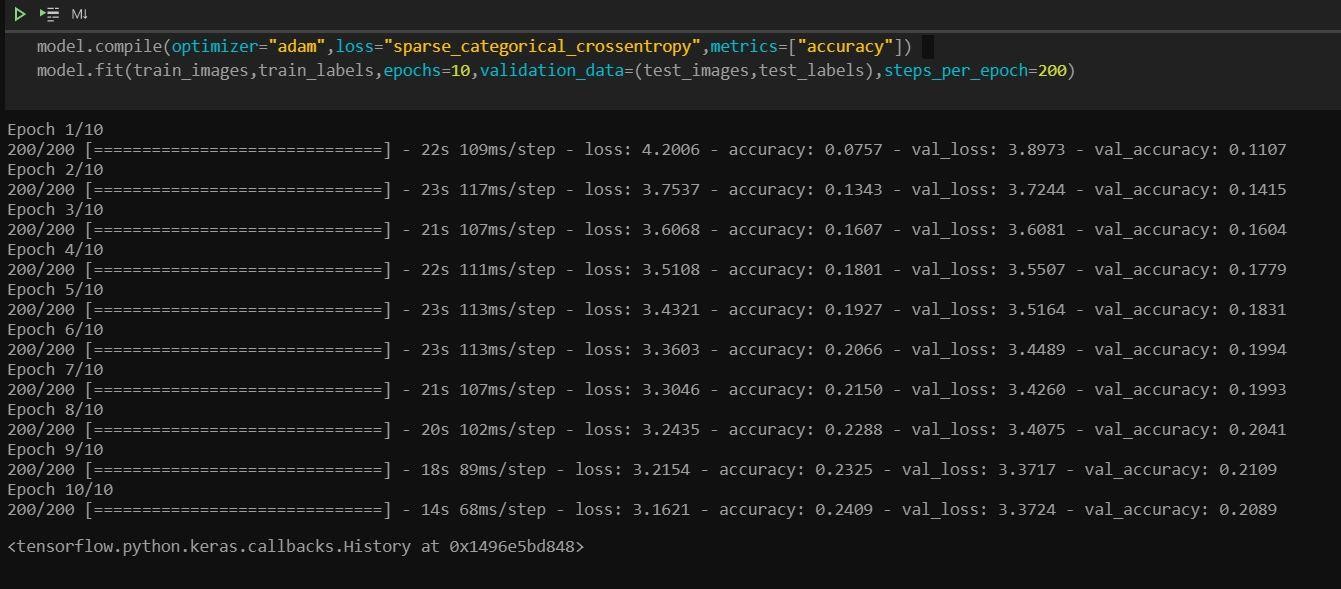
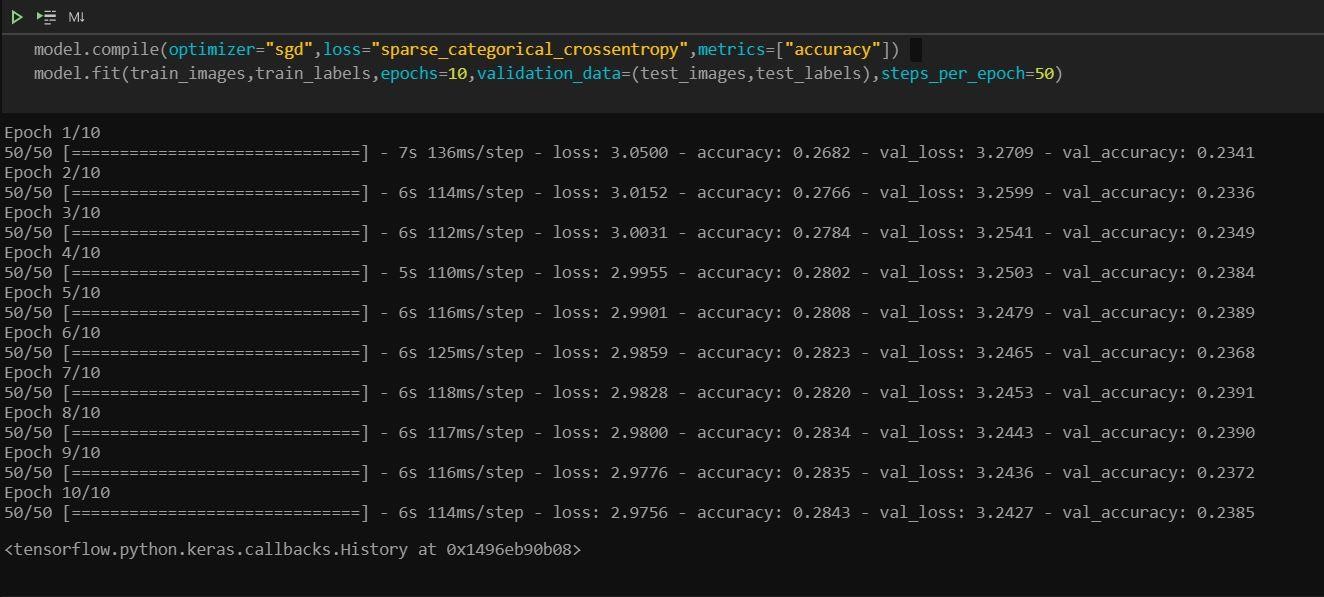
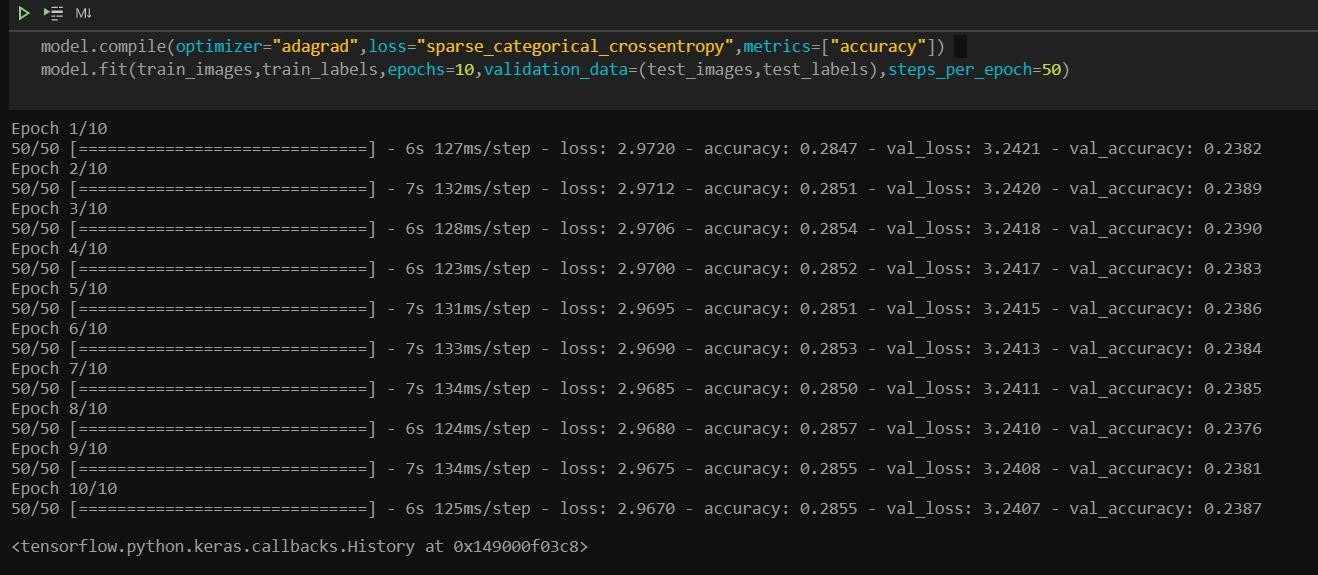
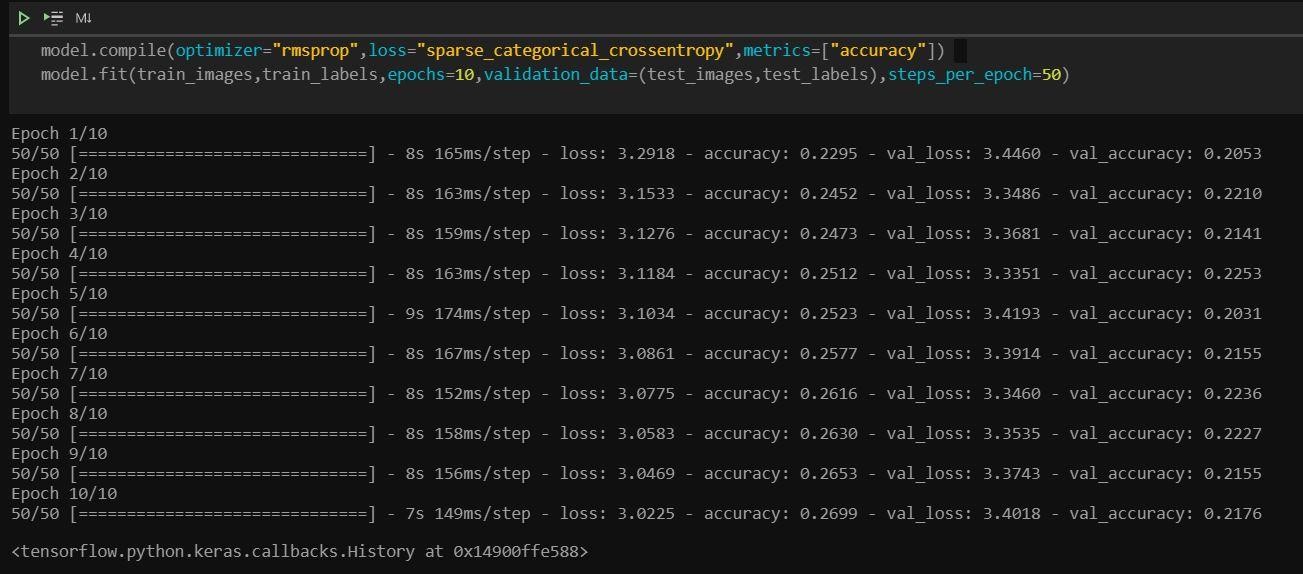
model.compile(optimizer="rmsprop",loss="sparse\_categorical\_crossentropy",metri cs=["accuracy"]) model.fit(train\_images,train\_labels,epochs=10,validation\_data=(test\_images,tes t\_labels),steps\_per\_epoch=50)

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:**



**Conclusion:**

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