Neural Network and Deep learning: Assignment -3

Mallika Mamidi

700746126

GitHub: <https://github.com/mallika76/NN-Assignment3.git>

Video Link: <https://drive.google.com/file/d/1q0KS_9Jxmw2g1Q7yx-_a52yF8Yufgfcl/view?usp=sharing>

1. First I have execute the code given in the assignment so that we can check both the performance after adding the layer given in the first question.

# Simple CNN model for CIFAR-10

import numpy

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from keras.constraints import maxnorm

from keras.optimizers import SGD

from keras.layers.convolutional import Conv2D

from keras.layers.convolutional import MaxPooling2D

from keras.utils import np\_utils

#from keras import backend as K

#K.set\_image\_dim\_ordering('th')

# fix random seed for reproducibility

seed = 7

numpy.random.seed(seed)

# load data

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

# normalize inputs from 0-255 to 0.0-1.0

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# one hot encode outputs

y\_train = np\_utils.to\_categorical(y\_train)

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# Create the model

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

# Compile model

epochs = 5

lrate = 0.01

decay = lrate/epochs

sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

print(model.summary())

# Fit the model

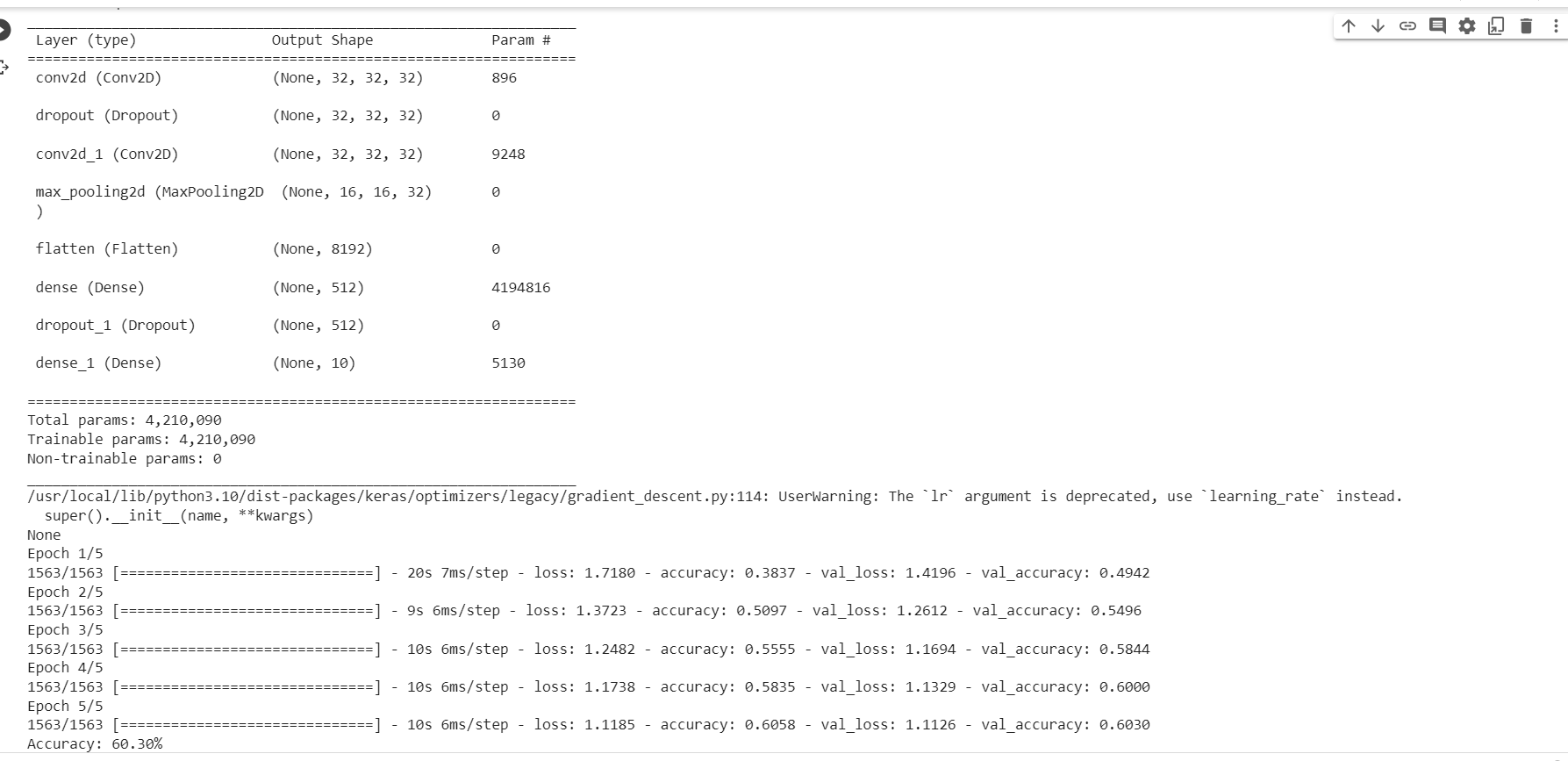
model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=32)

# Final evaluation of the model

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))

Accuracy for above code: **60.30 %** on the validation set after 5 epochs



QUESTION 1.

import numpy as np

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.constraints import maxnorm

from keras.utils import np\_utils

from keras.optimizers import SGD

# Fix random seed for reproducibility

seed = 7

numpy.random.seed(seed)

# Load data

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

# Normalize inputs from 0-255 to 0.0-1.0

X\_train = X\_train.astype('float32') / 255.0

X\_test = X\_test.astype('float32') / 255.0

# One hot encode outputs

y\_train = np\_utils.to\_categorical(y\_train)

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

# Create the model

model = Sequential()

#Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.

model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3))) #

model.add(Dropout(0.2))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

#Convolutional input layer, 64 feature maps with a size of 3×3 and a rectifier activation function.

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

#Convolutional input layer, 128 feature maps with a size of 3×3 and a rectifier activation function.

model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

#Flatten layer

model.add(Flatten())

model.add(Dropout(0.2))

#Fully connected layer with 1024 units and a rectifier activation function.

model.add(Dense(1024, activation='relu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

#Fully connected layer with 512 units and a rectifier activation function.

model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))

model.add(Dropout(0.2))

#Fully connected output layer with 10 units and a Softmax activation function

model.add(Dense(num\_classes, activation='softmax'))

# Compile model

epochs = 5

learning\_rate = 0.01

decay\_rate = learning\_rate / epochs

sgd = SGD(lr=learning\_rate, momentum=0.9, decay=decay\_rate, nesterov=False)

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

print(model.summary())

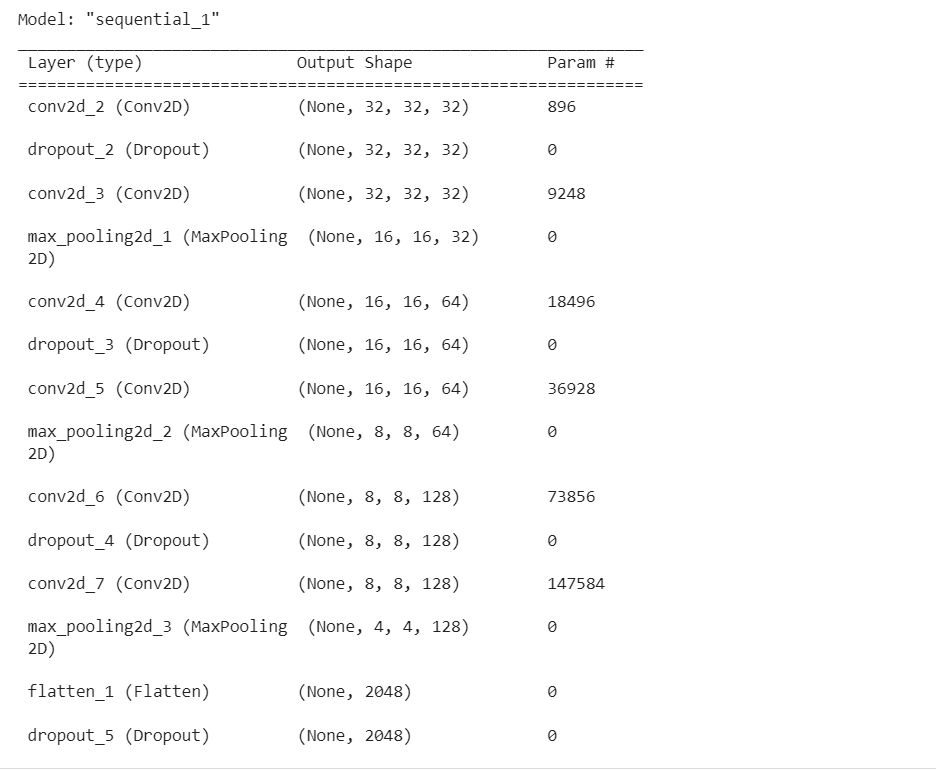
# Fit the model

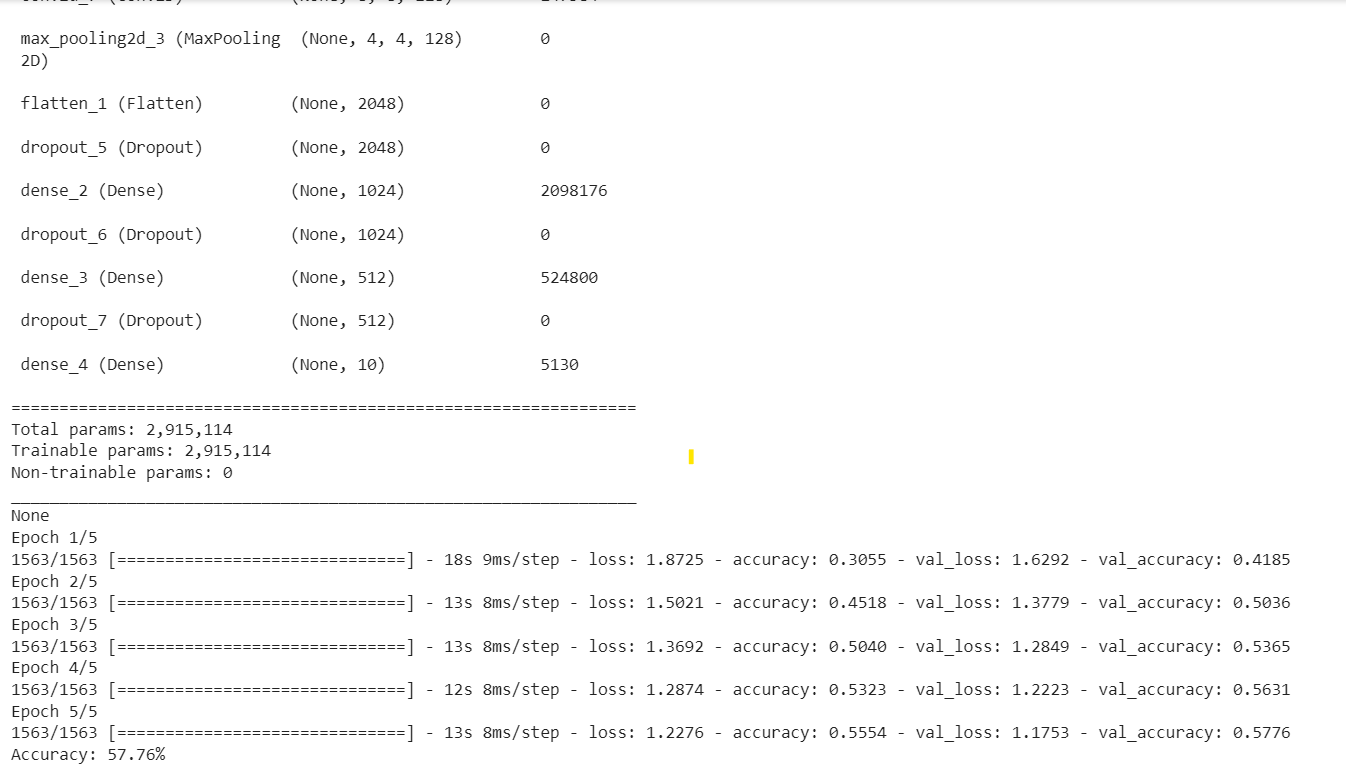
history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=32)

# Evaluate the model

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1] \* 100))





Did the performance change?

Yes, the performance of the model changed after the update. The initial model had an accuracy of 60.30% on the validation set after 5 epochs, while the updated model achieved an accuracy of 57.76% on the validation set after the same number of epochs.

