

NEURAL NETWORK & DEEP LEARNING

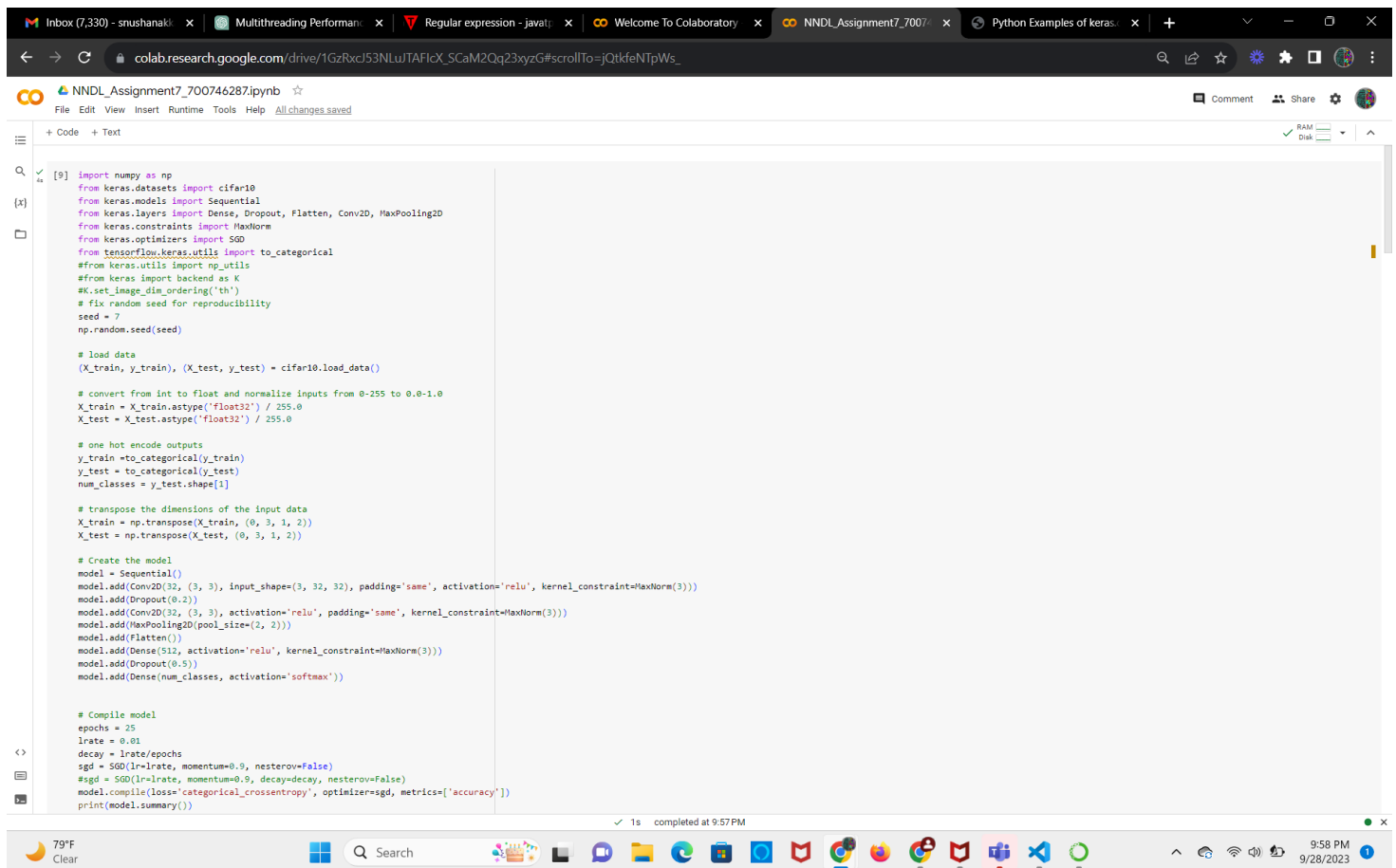
ASSIGNMENT 7

Name : SAI SNUSHA NAKKA

Student ID : 700746287

Git hub Link: https://github.com/NSnusha/NNDL_Assignment7

Video link: https://drive.google.com/file/d/1vAPMQuRB_fKzScIQ1fcu6ngx9wm-EO6R/view?usp=sharing



The screenshot displays a Google Colab notebook titled "NNDL_Assignment7_700746287.ipynb". The notebook contains a Python script for training a deep neural network on the CIFAR-10 dataset using Keras and TensorFlow. The script includes the following components:

- Imports:** Imports necessary libraries such as numpy, keras.datasets, keras.models, keras.layers, keras.constraints, keras.optimizers, tensorflow.keras.utils, and keras.backend.
- Seed Setting:** Sets a random seed for reproducibility.
- Data Loading:** Loads the CIFAR-10 dataset and splits it into training and testing sets.
- Data Preprocessing:** Converts the data to float32, normalizes it, and encodes the labels as one-hot vectors.
- Model Architecture:** Defines a Sequential model with the following layers:
 - Conv2D(32, (3, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3))
 - Dropout(0.2)
 - Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3))
 - MaxPooling2D(pool_size=(2, 2))
 - Flatten()
 - Dense(512, activation='relu', kernel_constraint=MaxNorm(3))
 - Dropout(0.5)
 - Dense(num_classes, activation='softmax')
- Compilation:** Compiles the model with the 'categorical_crossentropy' loss function, 'sgd' optimizer, and 'accuracy' metric.
- Training:** Trains the model for 25 epochs with a learning rate of 0.01, using SGD with a momentum of 0.9 and a decay of 0.9.
- Summary:** Prints the summary of the trained model.

The notebook interface shows the code in the left pane and the output in the right pane. The status bar at the bottom indicates that the code was completed at 9:57 PM.


```
Inbox (7,330) - snushanaki x Multithreading Performan Regular expression - java Welcome To Colaboratory x NNDL_Assignment7_7007 Python Examples of keras x +
colab.research.google.com/drive/1GzRxcJ53NLuTAFicX_SCaM2Qq23xyzG#scrollTo=jQtkfENTpWs_

NNDL_Assignment7_700746287.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[11] import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.constraints import MaxNorm
from keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
# from keras import backend as K
# K.tensorflow_backend.set_image_dim_ordering('th')
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# convert from int to float and 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 = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# transpose the dimensions of the input data
X_train = np.transpose(X_train, (0, 3, 1, 2))
X_test = np.transpose(X_test, (0, 3, 1, 2))

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), 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(Conv2D(64, (3, 3), padding='same', activation='relu', 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=(1, 1)))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu', 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=(1, 1)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

1s completed at 9:57 PM
```

79°F Clear Search 9:58 PM 9/28/2023

```
Inbox (7,330) - snushanaki x Multithreading Performan Regular expression - java Welcome To Colaboratory x NNDL_Assignment7_7007 Python Examples of keras x +
colab.research.google.com/drive/1GzRxcJ53NLuTAFicX_SCaM2Qq23xyzG#scrollTo=jQtkfENTpWs_

NNDL_Assignment7_700746287.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[11] # Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, nesterov=False)
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
history=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))

Total params: 2923466 (11.15 MB)
Trainable params: 2923466 (11.15 MB)
Non-trainable params: 0 (0.00 Byte)

None
Epoch 1/25
1563/1563 [=====] - 157s 100ms/step - loss: 2.0287 - accuracy: 0.2478 - val_loss: 1.8118 - val_accuracy: 0.3352
Epoch 2/25
1563/1563 [=====] - 157s 101ms/step - loss: 1.7049 - accuracy: 0.3719 - val_loss: 1.6106 - val_accuracy: 0.4142
Epoch 3/25
1563/1563 [=====] - 151s 97ms/step - loss: 1.5851 - accuracy: 0.4206 - val_loss: 1.4497 - val_accuracy: 0.4679
Epoch 4/25
1563/1563 [=====] - 154s 99ms/step - loss: 1.4944 - accuracy: 0.4531 - val_loss: 1.4285 - val_accuracy: 0.4827
Epoch 5/25
1563/1563 [=====] - 157s 100ms/step - loss: 1.4367 - accuracy: 0.4779 - val_loss: 1.3842 - val_accuracy: 0.5017
Epoch 6/25
1563/1563 [=====] - 153s 98ms/step - loss: 1.3850 - accuracy: 0.4961 - val_loss: 1.3408 - val_accuracy: 0.5060
Epoch 7/25
1563/1563 [=====] - 151s 97ms/step - loss: 1.3526 - accuracy: 0.5115 - val_loss: 1.2884 - val_accuracy: 0.5310
Epoch 8/25
1563/1563 [=====] - 151s 97ms/step - loss: 1.3200 - accuracy: 0.5231 - val_loss: 1.2326 - val_accuracy: 0.5571
Epoch 9/25
1563/1563 [=====] - 146s 93ms/step - loss: 1.2947 - accuracy: 0.5309 - val_loss: 1.2812 - val_accuracy: 0.5373
Epoch 10/25
1563/1563 [=====] - 142s 91ms/step - loss: 1.2659 - accuracy: 0.5439 - val_loss: 1.2461 - val_accuracy: 0.5559
Epoch 11/25
1563/1563 [=====] - 142s 91ms/step - loss: 1.2443 - accuracy: 0.5526 - val_loss: 1.2213 - val_accuracy: 0.5592
Epoch 12/25
1563/1563 [=====] - 145s 93ms/step - loss: 1.2286 - accuracy: 0.5561 - val_loss: 1.2474 - val_accuracy: 0.5621
Epoch 13/25
1563/1563 [=====] - 148s 95ms/step - loss: 1.2146 - accuracy: 0.5636 - val_loss: 1.2121 - val_accuracy: 0.5666
Epoch 14/25
1563/1563 [=====] - 147s 94ms/step - loss: 1.1936 - accuracy: 0.5685 - val_loss: 1.1933 - val_accuracy: 0.5757
Epoch 15/25
1563/1563 [=====] - 146s 94ms/step - loss: 1.1866 - accuracy: 0.5723 - val_loss: 1.1981 - val_accuracy: 0.5781
Epoch 16/25
1563/1563 [=====] - 147s 94ms/step - loss: 1.1684 - accuracy: 0.5793 - val_loss: 1.1962 - val_accuracy: 0.5761
Epoch 17/25
1563/1563 [=====] - 149s 95ms/step - loss: 1.1511 - accuracy: 0.5852 - val_loss: 1.1503 - val_accuracy: 0.5875

1s completed at 9:57 PM
```

79°F Clear Search 9:58 PM 9/28/2023

Colab interface showing code execution for a neural network model. The code includes training history output, predictions, and plotting of accuracy and loss.

```
[11] 1563/1563 [=====] - 146s 94ms/step - loss: 1.1454 - accuracy: 0.5891 - val_loss: 1.1824 - val_accuracy: 0.5839
Epoch 19/25
1563/1563 [=====] - 147s 94ms/step - loss: 1.1399 - accuracy: 0.5877 - val_loss: 1.2279 - val_accuracy: 0.5703
Epoch 20/25
1563/1563 [=====] - 149s 95ms/step - loss: 1.1252 - accuracy: 0.5960 - val_loss: 1.1551 - val_accuracy: 0.5871
Epoch 21/25
1563/1563 [=====] - 148s 94ms/step - loss: 1.1229 - accuracy: 0.5960 - val_loss: 1.1578 - val_accuracy: 0.5920
Epoch 22/25
1563/1563 [=====] - 154s 98ms/step - loss: 1.1093 - accuracy: 0.6008 - val_loss: 1.1799 - val_accuracy: 0.5838
Epoch 23/25
1563/1563 [=====] - 150s 96ms/step - loss: 1.1076 - accuracy: 0.6024 - val_loss: 1.1777 - val_accuracy: 0.5818
Epoch 24/25
1563/1563 [=====] - 148s 94ms/step - loss: 1.1011 - accuracy: 0.6033 - val_loss: 1.2040 - val_accuracy: 0.5760
Epoch 25/25
1563/1563 [=====] - 149s 95ms/step - loss: 1.0967 - accuracy: 0.6068 - val_loss: 1.1685 - val_accuracy: 0.5884
Accuracy: 58.84%
```

```
predictions = model.predict(X_test[:4])
print(predictions)
print(np.argmax(predictions, axis=1))
print(y_test[:4])
```

```
1/1 [=====] - 0s 182ms/step
[[6.6421755e-02 4.8251629e-02 4.9000204e-02 3.7323502e-01 8.7610923e-02
 2.3808683e-01 3.8983859e-02 2.4645276e-02 4.1188173e-02 3.2576293e-02]
 [4.0154178e-02 9.1406219e-02 1.3929105e-03 1.1444904e-03 9.6673996e-04
 1.9689640e-03 5.8818251e-04 1.2016086e-03 6.4895992e-01 1.2296793e-02]
 [5.3523266e-01 3.0359907e-02 3.6236208e-02 1.0163108e-02 1.4053379e-02
 1.1074324e-02 5.2310782e-04 6.3204574e-03 3.2090539e-01 3.5131443e-02]
 [7.7109390e-01 1.6146995e-02 1.9118270e-02 1.4980440e-02 4.4592754e-03
 5.2238037e-04 4.0119587e-04 1.6665384e-01 2.7209385e-03 3.9031005e-03]]
[[3 8 0]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0.]]
```

```
import matplotlib.pyplot as plt

# Plot training and validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training and validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
```

