

Neural Network and Deep Learning – ICP4

Name : Prajwala Nalluri

Student ID: 700766230

Github Link: <https://github.com/Prajwalanalluri/Neural-Assignment-4.git>

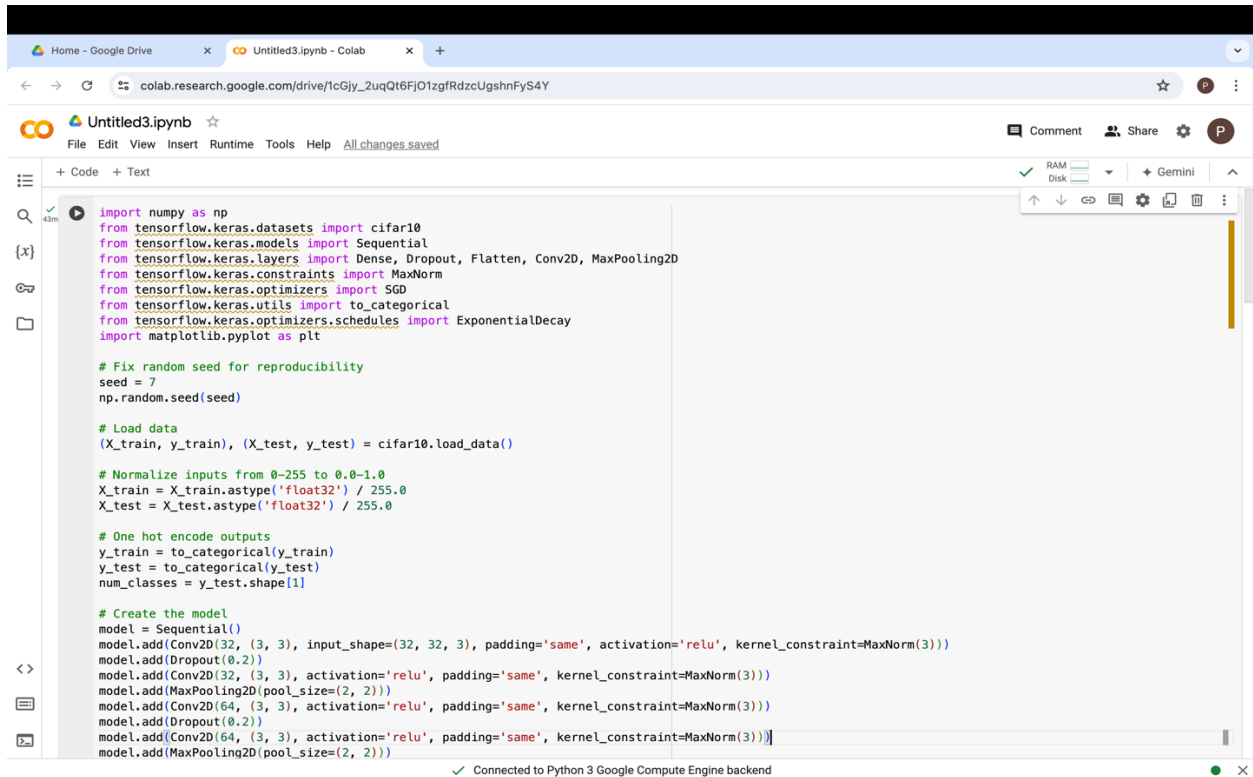
Video link:

https://drive.google.com/file/d/1SewfCE8WYSyE9x7MVrKfhj7lH_ndxLge/view?usp=drive_link

1. Follow the instruction below and then report how the performance changed.(apply all at once)
 - Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2 .
 - Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2 .
 - Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2 .
 - Flatten layer.
 - Dropout layer at 20%.
 - Fully connected layer with 1024 units and a rectifier activation function.
 - Dropout layer at 20%.
 - Fully connected layer with 512 units and a rectifier activation function.
 - Dropout layer at 20%.
 - Fully connected output layer with 10 units and a Softmax activation function Did the performance change?

2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.
3. Visualize Loss and Accuracy using the history object.

Output:



The screenshot shows a Google Colab notebook titled 'Untitled3.ipynb'. The code in the notebook is as follows:

```
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers.schedules import ExponentialDecay
import matplotlib.pyplot as plt

# Fix random seed for reproducibility
seed = 7
np.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 = to_categorical(y_train)
y_test = 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(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)))
```

At the bottom of the notebook, a status bar indicates 'Connected to Python 3 Google Compute Engine backend'.

Home - Google Drive x Untitled3.ipynb - Colab x +

colab.research.google.com/drive/1cGjy_2uqQt6FjO1zgFRdzUgshnFyS4Y

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings

+ Code + Text

```
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
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)))
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'))

# Compile model
epochs = 5
lr_rate = 0.01
lr_schedule = ExponentialDecay(
    initial_learning_rate=lr_rate,
    decay_steps=epochs * len(X_train) // 32,
    decay_rate=0.1
)
sgd = SGD(learning_rate=lr_schedule, momentum=0.9, 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))

# Predict the first 4 images of the test data
predictions = model.predict(X_test[:4])
predicted_classes = np.argmax(predictions, axis=1)
actual_classes = np.argmax(y_test[:4], axis=1)
```

Connected to Python 3 Google Compute Engine backend

Home - Google Drive x Untitled3.ipynb - Colab x +

colab.research.google.com/drive/1cGjy_2uqQt6FjO1zgFRdzUgshnFyS4Y

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings

+ Code + Text

```
predicted_classes = np.argmax(predictions, axis=1)
actual_classes = np.argmax(y_test[:4], axis=1)

# Print the predictions and actual labels
print("Predicted classes: ", predicted_classes)
print("Actual classes: ", actual_classes)

# Plot the first 4 test images, predicted labels, and actual labels
fig, axes = plt.subplots(1, 4, figsize=(15, 3))
for i in range(4):
    axes[i].imshow(X_test[i])
    axes[i].set_title(f"Pred: {predicted_classes[i]}, Actual: {actual_classes[i]}")
    axes[i].axis('off')
plt.show()

# Visualize Loss and Accuracy
plt.figure(figsize=(12, 4))

# Plot Loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper right')

# Plot Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='upper left')

plt.show()
```

Connected to Python 3 Google Compute Engine backend

Chrome File Edit View History Bookmarks Profiles Tab Window Help Wed Jul 10 20:22

colab.research.google.com/drive/1cGjy_2uqQt6FjO1zgfRdzcUgshnFyS4Y

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

43m

Trainable params: 2915114 (11.12 MB)
Non-trainable params: 0 (0.00 Byte)

None

Epoch 1/5
1563/1563 [=====] - 524s 334ms/step - loss: 1.8998 - accuracy: 0.2948 - val_loss: 1.5525 - val_accuracy: 0.4338
Epoch 2/5
1563/1563 [=====] - 535s 342ms/step - loss: 1.4456 - accuracy: 0.4731 - val_loss: 1.3263 - val_accuracy: 0.5190
Epoch 3/5
1563/1563 [=====] - 492s 315ms/step - loss: 1.2549 - accuracy: 0.5457 - val_loss: 1.2085 - val_accuracy: 0.5637
Epoch 4/5
1563/1563 [=====] - 503s 322ms/step - loss: 1.1270 - accuracy: 0.5963 - val_loss: 1.0850 - val_accuracy: 0.6104
Epoch 5/5
1563/1563 [=====] - 481s 308ms/step - loss: 1.0360 - accuracy: 0.6271 - val_loss: 1.0352 - val_accuracy: 0.6334
Accuracy: 63.34%
1/1 [=====] - 0s 301ms/step
Predicted classes: [3 8 8 0]
Actual classes: [3 8 8 0]

Pred: 3, Actual: 3

Pred: 8, Actual: 8

Pred: 8, Actual: 8

Pred: 0, Actual: 0

Model Loss

Model Accuracy

0.65

train_loss

train_accuracy

Connected to Python 3 Google Compute Engine backend

Home - Google Drive x x +

colab.research.google.com/drive/1cGjy_2uqQt6FjO1zgfRdzcUgshnFyS4Y

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

43m

Pred: 3, Actual: 3

Pred: 8, Actual: 8

Pred: 8, Actual: 8

Pred: 0, Actual: 0

Model Loss

Model Accuracy

Loss

Accuracy

Epoch

Epoch

Connected to Python 3 Google Compute Engine backend