

Spring 2024: CS5720 Neural Networks & Deep Learning - ICP-8

Assignment- 7

Name : Afroz Mohammad Student Id : 700758012

GitHub Link: <https://github.com/Afrozmoammad19/Assignment7>

Video Link <https://drive.google.com/file/d/1tlf48OSveLaZunc-i2lNs5VJj2RcJG8I/view?usp=sharing>

Use Case Description:

LeNet5, AlexNet, Vgg16, Vgg19

1. Training the model
2. Evaluating the model

Programming elements:

1. About CNN
2. Hyperparameters of CNN
3. Image classification with CNN

In class programming:

1. Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy and reduce validation loss.
2. Provide logical description of which steps lead to improved response and what was its impact on architecture behavior.
3. Create at least two more visualizations using matplotlib (Other than provided in the source file)
4. Use dataset of your own choice and implement baseline models provided.
5. Apply modified architecture to your own selected dataset and train it.
6. Evaluate your model on testing set.
7. Save the improved model and use it for prediction on testing data
8. Provide plot of confusion matrix
9. Provide Training and testing Loss and accuracy plots in one plot using subplot command and history object.
10. Provide at least two more visualizations reflecting your solution.
11. Provide logical description of which steps lead to improved response for new dataset when compared with baseline model and enhance architecture and what was its impact on architecture behavior.

Home Page - Select or create a notebook | icp8_700758012 - Jupyter Notebook | +

localhost:8888/notebooks/icp8_700758012.ipynb

jupyter icp8_700758012 Last Checkpoint: 22 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.datasets import cifar100

from tensorflow.keras.optimizers import RMSprop
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, BatchNormalization

%matplotlib inline
```

Extract data and train and test dataset

In [2]:

```
#cifar100 = tf.keras.datasets.cifar100
(X_train,Y_train) , (X_test,Y_test) = cifar100.load_data()

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169001437/169001437 [=====] - 6s 0us/step
```

In [27]:

```
classes = ['apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle', 'b
```

Let's look into the dataset images

Home Page - Select or create a notebook | icp8_700758012 - Jupyter Notebook | +

localhost:8888/notebooks/icp8_700758012.ipynb

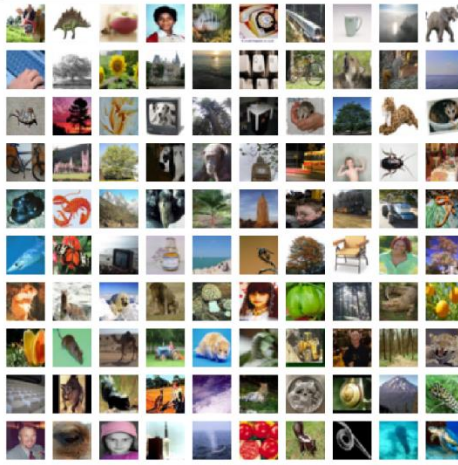
jupyter icp8_700758012 Last Checkpoint: 23 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Let's look into the dataset images

In [3]:

```
plt.figure(figsize=(10,10))
for i in range(100):
    plt.subplot(10,10,i+1)
    plt.axis('off')
    plt.imshow(X_train[i], cmap = 'gray')
```



Training, Validating and Splitting trained and tested data

In [4]:

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(X_train, Y_train, test_size=0.2)
```


Home Page - Select or create a ... icp8_700758012 - Jupyter Notebo ...

localhost:8888/notebooks/icp8_700758012.ipynb

jupyter icp8_700758012 Last Checkpoint 24 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Non-trainable params: 0

```
In [14]: model = tf.keras.Sequential()
model.add(vgg_model)
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1024, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(100, activation='softmax'))

model.summary()

Model: "sequential_1"
Layer (type) Output Shape Param #
-----
vgg19 (Functional) (None, 1, 1, 512) 20824384
flatten_1 (Flatten) (None, 512) 0
dense_4 (Dense) (None, 1024) 525312
batch_normalization_3 (Batch Normalization) (None, 1024) 4096
dense_5 (Dense) (None, 1024) 1049600
batch_normalization_4 (Batch Normalization) (None, 1024) 4096
dense_6 (Dense) (None, 256) 262400
batch_normalization_5 (Batch Normalization) (None, 256) 1024
dropout_1 (Dropout) (None, 256) 0
dense_7 (Dense) (None, 100) 25700
-----
Total params: 21,896,612
Trainable params: 21,892,004
Non-trainable params: 4,608
```

```
In [15]: optimizer = tf.keras.optimizers.SGD(learning_rate = 0.001, momentum = 0.9)
model.compile(optimizer=optimizer,
```

Home Page - Select or create a ... icp8_700758012 - Jupyter Notebo ...

localhost:8888/notebooks/icp8_700758012.ipynb

jupyter icp8_700758012 Last Checkpoint 25 minutes ago (autosaved)

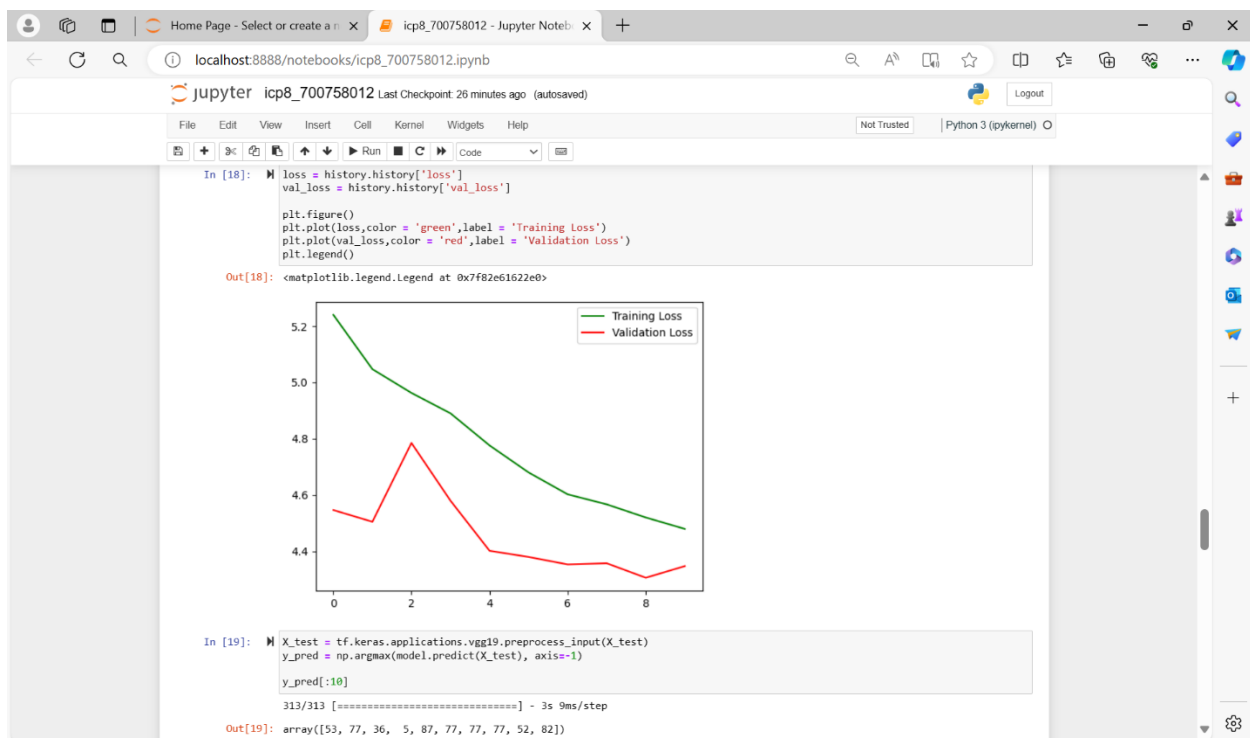
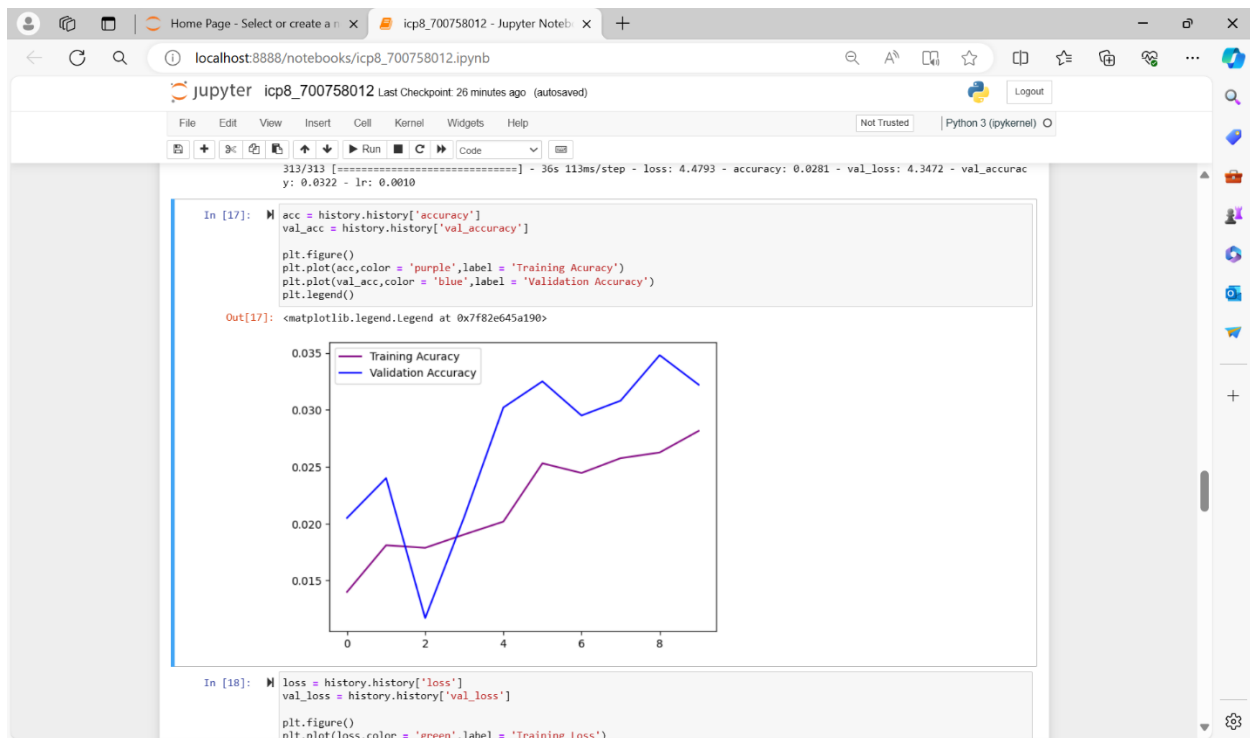
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [15]: optimizer = tf.keras.optimizers.SGD(learning_rate = 0.001, momentum = 0.9)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

In [16]: history = model.fit(
    train_datagen.flow(x_train, y_train, batch_size = 128),
    validation_data = val_datagen.flow(x_val, y_val, batch_size = 128),
    epochs = 10,
    verbose = 1,
    callbacks = [learning_rate_reduction]
)

Epoch 1/10
313/313 [=====] - 44s 124ms/step - loss: 5.2395 - accuracy: 0.0140 - val_loss: 4.5463 - val_accu
racy: 0.0205 - lr: 0.0010
Epoch 2/10
313/313 [=====] - 38s 121ms/step - loss: 5.0465 - accuracy: 0.0181 - val_loss: 4.5047 - val_accu
racy: 0.0240 - lr: 0.0010
Epoch 3/10
313/313 [=====] - 36s 116ms/step - loss: 4.9621 - accuracy: 0.0179 - val_loss: 4.7844 - val_accu
racy: 0.0117 - lr: 0.0010
Epoch 4/10
313/313 [=====] - 36s 115ms/step - loss: 4.8896 - accuracy: 0.0191 - val_loss: 4.5794 - val_accu
racy: 0.0206 - lr: 0.0010
Epoch 5/10
313/313 [=====] - 36s 116ms/step - loss: 4.7757 - accuracy: 0.0202 - val_loss: 4.4016 - val_accu
racy: 0.0302 - lr: 0.0010
Epoch 6/10
313/313 [=====] - 36s 116ms/step - loss: 4.6797 - accuracy: 0.0253 - val_loss: 4.3797 - val_accu
racy: 0.0325 - lr: 0.0010
Epoch 7/10
313/313 [=====] - 35s 113ms/step - loss: 4.6021 - accuracy: 0.0245 - val_loss: 4.3531 - val_accu
racy: 0.0295 - lr: 0.0010
Epoch 8/10
313/313 [=====] - 36s 115ms/step - loss: 4.5665 - accuracy: 0.0258 - val_loss: 4.3573 - val_accu
racy: 0.0308 - lr: 0.0010
Epoch 9/10
313/313 [=====] - 36s 116ms/step - loss: 4.5199 - accuracy: 0.0262 - val_loss: 4.3059 - val_accu
racy: 0.0340 - lr: 0.0010
Epoch 10/10
313/313 [=====] - 36s 113ms/step - loss: 4.4793 - accuracy: 0.0281 - val_loss: 4.3472 - val_accu
racy: 0.0322 - lr: 0.0010

In [17]: acc = history.history['accuracy']
```



```
jupyter ip8_700758012 Last Checkpoint 26 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
In [19]: array([[24, 11, 20, 5, 18, 11, 24, 24]])

In [20]: from sklearn.metrics import confusion_matrix, accuracy_score
print('Testing Accuracy : ', accuracy_score(Y_test, y_pred))
Testing Accuracy : 0.0303

In [21]: cm = confusion_matrix(Y_test, y_pred)
cm
Out[21]: array([[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]])

In [30]: import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=True,
                          title='Confusion matrix',
                          cmap=plt.cm.Greens):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting 'normalize=True'.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=30)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    #print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(i, j, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

