Assignment 6.2a

author: Rachel Nelson

class: DSC650

Assignment 6.2a

Using section 5.2 in Deep Learning with Python as a guide, create a ConvNet model that classifies images CIFAR10 small images classification dataset. Do not use dropout or data-augmentation in this part. Save the model, predictions, metrics, and validation plots in the dsc650/assignments/assignment06/results directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

23
```
from keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
import pandas as pd

from keras import layers
from keras import models
import pandas as pd
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import os, shutil
from keras.datasets import cifar10
import matplotlib.pyplot as plt
```

24
```
# Listing 5.5 Instantiating a small convnet for dogs vs. cats classification
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 32)        896       
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 15, 15, 32)       0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 13, 13, 64)        18496     
 max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 512)               0

 dense_6 (Dense)             (None, 512)               262656

 dense_7 (Dense)             (None, 10)                5130

=================================================================
Total params: 361,034
Trainable params: 361,034
Non-trainable params: 0
_________________________________________________________________
```

25
```
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
train_images.shape
test_images.shape
```

25
```
(10000, 32, 32, 3)
```

26
```
train_images = train_images.reshape((50000, 32, 32, 3))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 32, 32, 3))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

27
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=10, batch_size=64, validation_data=(test_images, t
```

```
Epoch 1/10
782/782 [==============================] - 21s 26ms/step - loss: 1.5785 - accuracy: 0.4277 - val_loss: 1.
Epoch 2/10
782/782 [==============================] - 19s 24ms/step - loss: 1.1618 - accuracy: 0.5901 - val_loss: 1.
Epoch 3/10
782/782 [==============================] - 20s 25ms/step - loss: 0.9733 - accuracy: 0.6591 - val_loss: 1.
Epoch 4/10
782/782 [==============================] - 19s 24ms/step - loss: 0.8404 - accuracy: 0.7065 - val_loss: 1.
Epoch 5/10
782/782 [==============================] - 19s 24ms/step - loss: 0.7414 - accuracy: 0.7417 - val_loss: 1.
Epoch 6/10
782/782 [==============================] - 20s 26ms/step - loss: 0.6546 - accuracy: 0.7718 - val_loss: 0.
Epoch 7/10
782/782 [==============================] - 21s 26ms/step - loss: 0.5776 - accuracy: 0.7981 - val_loss: 0.
Epoch 8/10
782/782 [==============================] - 20s 26ms/step - loss: 0.5127 - accuracy: 0.8228 - val_loss: 1.
Epoch 9/10
782/782 [==============================] - 20s 26ms/step - loss: 0.4515 - accuracy: 0.8424 - val_loss: 0.
Epoch 10/10
782/782 [==============================] - 20s 25ms/step - loss: 0.3995 - accuracy: 0.8622 - val_loss: 0.
```

34
```
model.save('results/6.2a_model.h5')
```

29
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
test_acc
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.9916 - accuracy: 0.7173□□□□□□□□□□□□□□□□□
```

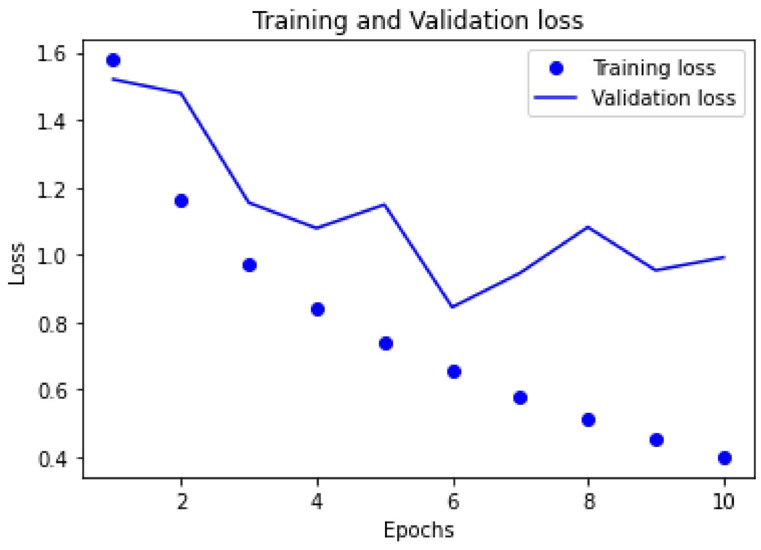
29
```
0.7172999978065491
```

33
```
history_dict = history.history
loss_values = history_dict['loss']
accuracy = history_dict['accuracy']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(accuracy) + 1)

#Plotting Training and Validation Loss on an image
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
plt.savefig('results/6.2a_LossPlot.png')
```

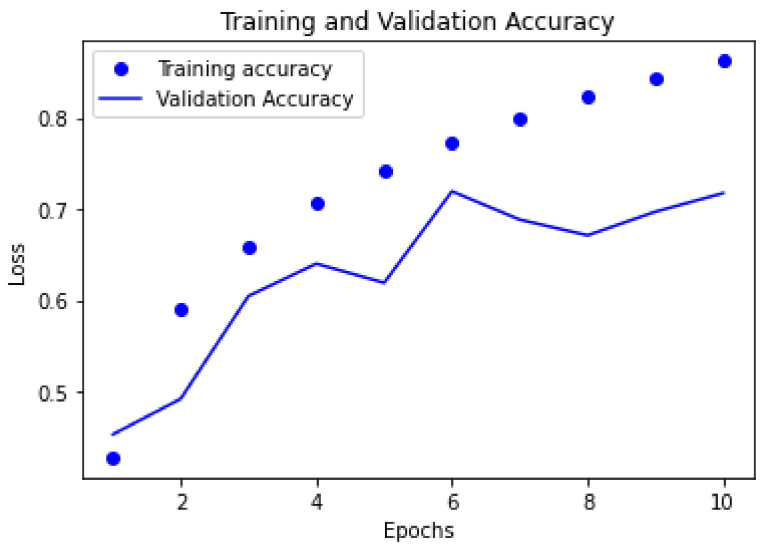


```
<Figure size 432x288 with 0 Axes>
```

32
```
plt.clf()
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']

plt.plot(epochs, acc_values, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc_values, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
plt.savefig('results/6.2a_AccuracyPlot.png')
```


Training and Validation Accuracy
Training accuracy
Validation Accuracy
Loss
0.5
0.6
0.7
0.8
2
4
6
8
10
Epochs

```
<Figure size 432x288 with 0 Axes>
```