Assignment 6.2b

author: Rachel Nelson

class: DSC650

Assignment 6.2.b

Using section 5.2 in Deep Learning with Python as a guide, create a ConvNet model that classifies images CIFAR10 small images classification dataset. This time includes dropout and data-augmentation. 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.

1
```
# Import packages
from keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
import pandas as pd
import matplotlib.pyplot as plt
```

2
```
#Loading the Data
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

3
```
# Training Shape
x_train.shape, y_train.shape
```

3
```
((50000, 32, 32, 3), (50000, 1))
```

4
```
# Testing Shape
x_test.shape, y_test.shape
```

4
```
((10000, 32, 32, 3), (10000, 1))
```

5
```
# Creating train and test sets
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

6
```
# Create samples for validation (using 10,000)
x_val = x_train[-10000:]
y_val = y_train[-10000:]
x_train_2 = x_train[:-10000]
y_train_2 = y_train[:-10000]
```

7
```
# Listing 5.14 Training the convnet using data-augmentation generators
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(x_train_2, y_train_2,  batch_size=32)

validation_generator = train_datagen.flow(x_val, y_val, batch_size=32)
```

8
```
from keras import models
from keras import layers

# Listing 5.2 Adding a classifier on top of the convnet, initiating the model
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.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
model.save('results/6.2b_Model.h5')
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 73,418
Trainable params: 73,418
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
10 # Listing 5.6 Configuring the model for training
   from tensorflow.keras import optimizers

   model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
```

```
D:\College\venv\lib\site-packages\keras\optimizer_v2\rmsprop.py:130: UserWarning: The `lr` argument is de
  super(RMSprop, self).__init__(name, **kwargs)
```

```
11 # Listing 5.8 Fitting the model using a batch generator
   history = model.fit_generator(train_generator,
                                 steps_per_epoch=len(x_train_2) / 32,
                                 epochs=30,
                                 validation_data=validation_generator,
                                 validation_steps=len(x_val) / 32)
```

```
D:\College\venv\lib\site-packages\ipykernel_launcher.py:5: UserWarning: `Model.fit_generator` is deprecat
  """

Epoch 1/30
1250/1250 [==============================] - 23s 18ms/step - loss: 2.1437 - accuracy: 0.1878 - val_loss:
Epoch 2/30
1250/1250 [==============================] - 23s 18ms/step - loss: 1.9689 - accuracy: 0.2581 - val_loss:
Epoch 3/30
1250/1250 [==============================] - 23s 18ms/step - loss: 1.9023 - accuracy: 0.2874 - val_loss:
Epoch 4/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.8593 - accuracy: 0.3092 - val_loss:
Epoch 5/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.8173 - accuracy: 0.3278 - val_loss:
Epoch 6/30
1250/1250 [==============================] - 23s 19ms/step - loss: 1.7873 - accuracy: 0.3364 - val_loss:
Epoch 7/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.7588 - accuracy: 0.3542 - val_loss:
Epoch 8/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.7316 - accuracy: 0.3661 - val_loss:
Epoch 9/30
1250/1250 [==============================] - 23s 19ms/step - loss: 1.7137 - accuracy: 0.3720 - val_loss:
Epoch 10/30
1250/1250 [==============================] - 23s 19ms/step - loss: 1.6898 - accuracy: 0.3808 - val_loss:
Epoch 11/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.6740 - accuracy: 0.3872 - val_loss:
Epoch 12/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.6523 - accuracy: 0.3941 - val_loss:
Epoch 13/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.6369 - accuracy: 0.4032 - val_loss:
Epoch 14/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.6225 - accuracy: 0.4098 - val_loss:
Epoch 15/30
1250/1250 [==============================] - 23s 19ms/step - loss: 1.6159 - accuracy: 0.4126 - val_loss:
Epoch 16/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.6000 - accuracy: 0.4198 - val_loss:
Epoch 17/30
1250/1250 [==============================] - 25s 20ms/step - loss: 1.5832 - accuracy: 0.4263 - val_loss:
Epoch 18/30
1250/1250 [==============================] - 24s 20ms/step - loss: 1.5797 - accuracy: 0.4278 - val_loss:
Epoch 19/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5651 - accuracy: 0.4346 - val_loss:
Epoch 20/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5573 - accuracy: 0.4376 - val_loss:
```

```
Epoch 21/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5450 - accuracy: 0.4442 - val_loss:
Epoch 22/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5333 - accuracy: 0.4452 - val_loss:
Epoch 23/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5238 - accuracy: 0.4500 - val_loss:
Epoch 24/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5126 - accuracy: 0.4555 - val_loss:
Epoch 25/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5128 - accuracy: 0.4540 - val_loss:
Epoch 26/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.5011 - accuracy: 0.4596 - val_loss:
Epoch 27/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.4936 - accuracy: 0.4615 - val_loss:
Epoch 28/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.4855 - accuracy: 0.4645 - val_loss:
Epoch 29/30
1250/1250 [==============================] - 24s 19ms/step - loss: 1.4741 - accuracy: 0.4693 - val_loss:
Epoch 30/30
1250/1250 [==============================] - 25s 20ms/step - loss: 1.4703 - accuracy: 0.4762 - val_loss:
```

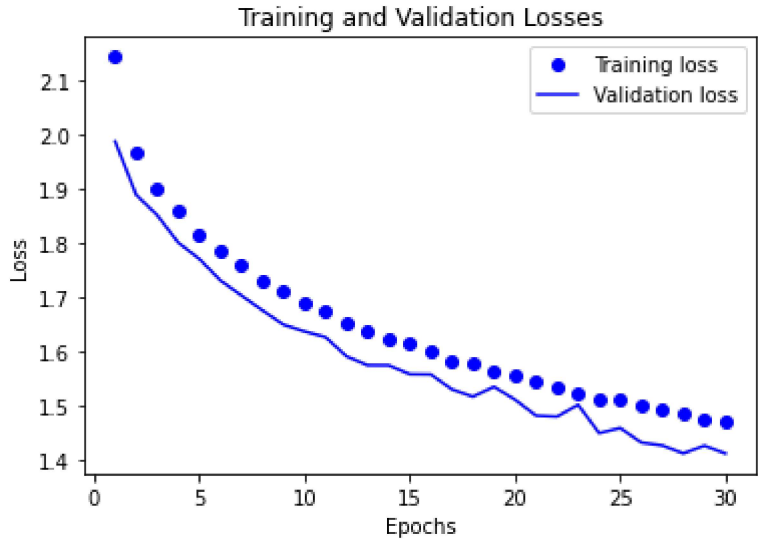
12
```
# Listing 5.9 Saving the Model
model.save('results/Assignment6.2b_Model.h5')
```

13
```
# Plotting Results for Loss
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(history.history['loss']) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Losses')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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




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

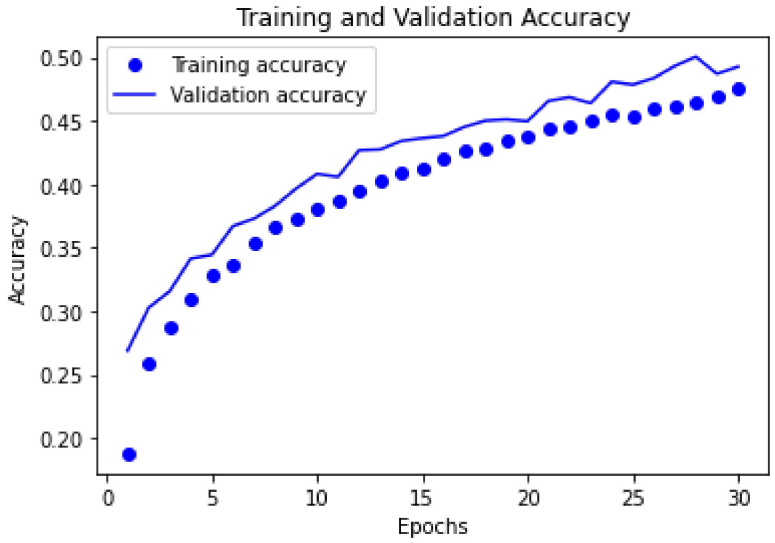
14
```
# Plotting Results for Accuracy
```

```
loss = history.history['accuracy']
val_loss = history.history['val_accuracy']

epochs = range(1, len(history.history['accuracy']) + 1)

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

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




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

15
```
#retrain the model and evaluate on test
train_generator = train_datagen.flow(x_train, y_train,  batch_size=32)

model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

#16 epochs chosen based on graphs above
history = model.fit_generator(train_generator,
                              steps_per_epoch=len(x_train) / 32,
                              epochs=16)
results = model.evaluate(x_test, y_test)
```

```
D:\College\venv\lib\site-packages\keras\optimizer_v2\rmsprop.py:130: UserWarning: The `lr` argument is de
  super(RMSprop, self).__init__(name, **kwargs)
D:\College\venv\lib\site-packages\ipykernel_launcher.py:11: UserWarning: `Model.fit_generator` is depreca
  # This is added back by InteractiveShellApp.init_path()

Epoch 1/16
1562/1562 [==============================] - 25s 16ms/step - loss: 1.4652 - accuracy: 0.4743□□□□□□□□□□□□□□□
Epoch 2/16
1562/1562 [==============================] - 27s 17ms/step - loss: 1.4572 - accuracy: 0.4806□□□□□□□□□□□□□□□
Epoch 3/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.4479 - accuracy: 0.4823□□□□□□□□□□□□□□□
Epoch 4/16
1562/1562 [==============================] - 27s 17ms/step - loss: 1.4445 - accuracy: 0.4817□□□□□□□□□□□□□□□
Epoch 5/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.4343 - accuracy: 0.4885□□□□□□□□□□□□□□□
```

```
Epoch 6/16
1562/1562 [==============================] - 26s 17ms/step - loss: 1.4289 - accuracy: 0.4907□□□□□□□□□□□□□□□
Epoch 7/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.4182 - accuracy: 0.4909□□□□□□□□□□□□□□□
Epoch 8/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.4102 - accuracy: 0.4975□□□□□□□□□□□□□□□
Epoch 9/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.4078 - accuracy: 0.4971□□□□□□□□□□□□□□□
Epoch 10/16
1562/1562 [==============================] - 27s 17ms/step - loss: 1.3970 - accuracy: 0.5015□□□□□□□□□□□□□□□
Epoch 11/16
1562/1562 [==============================] - 26s 17ms/step - loss: 1.3952 - accuracy: 0.5020□□□□□□□□□□□□□□□
Epoch 12/16
1562/1562 [==============================] - 29s 18ms/step - loss: 1.3899 - accuracy: 0.5038□□□□□□□□□□□□□□□
Epoch 13/16
1562/1562 [==============================] - 28s 18ms/step - loss: 1.3873 - accuracy: 0.5051□□□□□□□□□□□□□□□
Epoch 14/16
1562/1562 [==============================] - 29s 18ms/step - loss: 1.3778 - accuracy: 0.5109□□□□□□□□□□□□□□□
Epoch 15/16
1562/1562 [==============================] - 29s 18ms/step - loss: 1.3697 - accuracy: 0.5128□□□□□□□□□□□□□□□
Epoch 16/16
1562/1562 [==============================] - 29s 19ms/step - loss: 1.3710 - accuracy: 0.5108□□□□□□□□□□□□□□□
313/313 [==============================] - 1s 4ms/step - loss: 263.2042 - accuracy: 0.3272□□□□□□□□□□□□□□□□□□
```

17
```
# Saving the model
model.save('results/6.2b_model2.h5')
```

18
```
# Running model in predict mode and saving file
predictions = model.predict(x_test)
predictions_df = pd.DataFrame(predictions)
predictions_df.to_csv('results/6.2b_predictions.csv', index=False)
```

19
```
# Metrics to file
with open('results/6_2b_metrics.txt', 'w') as f:
    f.write('Training Loss: {}'.format(str(history.history['loss'])))
    f.write('\nTraining Accuracy: {}'.format(str(history.history['accuracy'])))
    f.write('\nTest Loss: {}'.format(results[0]))
    f.write('\nTest Accuracy: {}'.format(results[1]))
```