VRANJAN_Week6.2b

April 26, 2021

[1]:
```
# Course DSC 650 - Data Mining
# Name - Vikas Ranjan
# Assignment - Assignment 6.2b - Create a ConvNet model that classifies images␣
 ↪CIFAR10 small images classification dataset.
#                               This time includes dropout and data-augmentation.
 ↪
```

[2]:
```
from keras.datasets import cifar10
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
from keras import models
from keras import layers
from keras import optimizers
import matplotlib.pyplot as plt
import pandas as pd
```

[3]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

[4]:
```
x_train.shape, y_train.shape
```

[4]:
```
((50000, 32, 32, 3), (50000, 1))
```

[5]:
```
x_test.shape, y_test.shape
```

[5]:
```
((10000, 32, 32, 3), (10000, 1))
```

[6]:
```
# Preprocess the data (these are NumPy arrays)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")

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

[7]:
```
# Reserve 10,000 samples for validation
x_val = x_train[-10000:]
y_val = y_train[-10000:]
x_train_2 = x_train[:-10000]
```

```
y_train_2 = y_train[:-10000]
```

[8]:
```
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)
```

[9]:
```
#instantiate 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.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: "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 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
```

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

[11]:
```
history = model.fit_generator(train_generator,
                              steps_per_epoch=len(x_train_2) / 32,
                              epochs=10,
                              validation_data=validation_generator,
                              validation_steps=len(x_val) / 32)
```

```
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/10
1250/1250 [==============================] - 50s 40ms/step - loss: 2.2648 -
accuracy: 0.1389 - val_loss: 2.0724 - val_accuracy: 0.2226
Epoch 2/10
1250/1250 [==============================] - 48s 39ms/step - loss: 2.0608 -
accuracy: 0.2241 - val_loss: 1.9252 - val_accuracy: 0.2807
Epoch 3/10
1250/1250 [==============================] - 53s 42ms/step - loss: 1.9486 -
accuracy: 0.2648 - val_loss: 1.8692 - val_accuracy: 0.3027
Epoch 4/10
1250/1250 [==============================] - 57s 45ms/step - loss: 1.8946 -
accuracy: 0.2900 - val_loss: 1.8183 - val_accuracy: 0.3214
Epoch 5/10
1250/1250 [==============================] - 59s 47ms/step - loss: 1.8463 -
accuracy: 0.3073 - val_loss: 1.7770 - val_accuracy: 0.3503
Epoch 6/10
1250/1250 [==============================] - 57s 46ms/step - loss: 1.7973 -
accuracy: 0.3280 - val_loss: 1.7426 - val_accuracy: 0.3615
Epoch 7/10
```

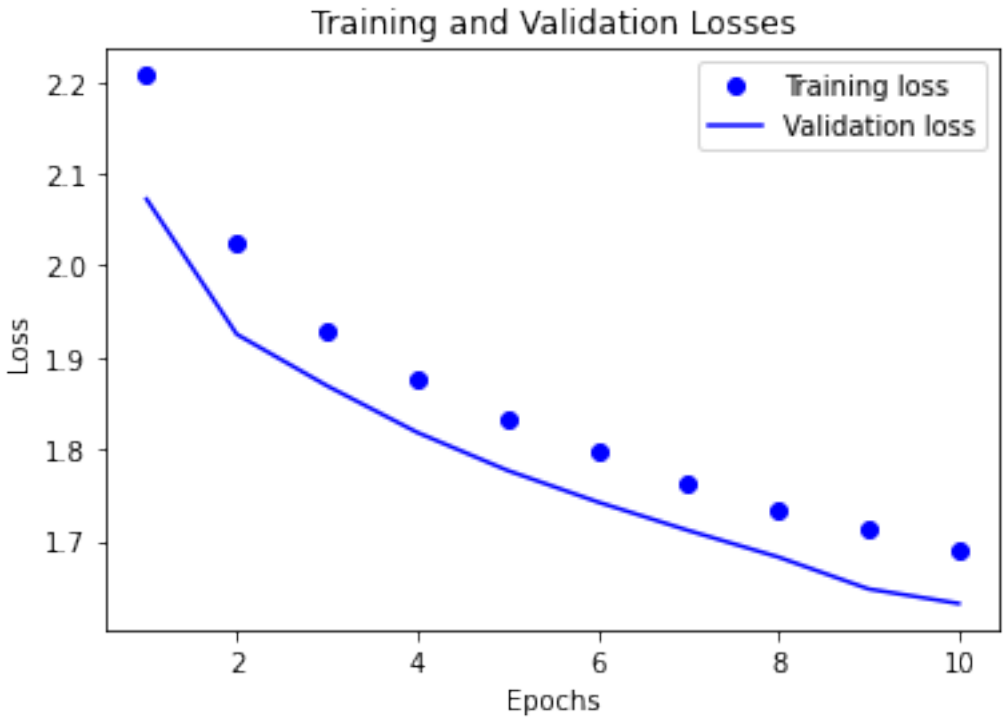
```
1250/1250 [==============================] - 63s 50ms/step - loss: 1.7680 -
accuracy: 0.3487 - val_loss: 1.7117 - val_accuracy: 0.3758
Epoch 8/10
1250/1250 [==============================] - 58s 46ms/step - loss: 1.7363 -
accuracy: 0.3590 - val_loss: 1.6827 - val_accuracy: 0.3841
Epoch 9/10
1250/1250 [==============================] - 55s 44ms/step - loss: 1.7163 -
accuracy: 0.3707 - val_loss: 1.6479 - val_accuracy: 0.4068
Epoch 10/10
1250/1250 [==============================] - 60s 48ms/step - loss: 1.7057 -
accuracy: 0.3734 - val_loss: 1.6321 - val_accuracy: 0.4098
```

[12]:
```
train_loss = history.history['loss']
val_loss = history.history['val_loss']

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

plt.plot(epochs, train_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_lossplot.png')
```



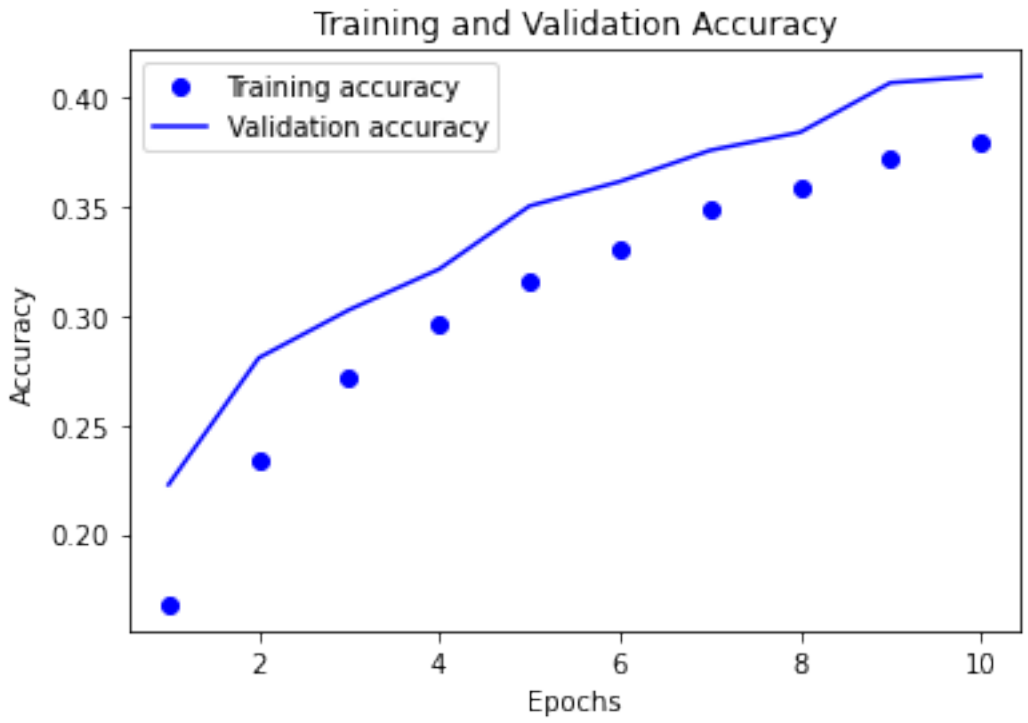
```
<Figure size 432x288 with 0 Axes>
```

[13]:
```
train_loss = history.history['accuracy']
val_loss = history.history['val_accuracy']

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

plt.plot(epochs, train_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_accplot.png')
```




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

[14]:
```
#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)
```

```
Epoch 1/16
1562/1562 [==============================] - 55s 35ms/step - loss: 1.6707 -
accuracy: 0.3881
Epoch 2/16
1562/1562 [==============================] - 58s 37ms/step - loss: 1.6496 -
accuracy: 0.4003
Epoch 3/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.6299 -
accuracy: 0.4097
Epoch 4/16
1562/1562 [==============================] - 53s 34ms/step - loss: 1.6135 -
accuracy: 0.4187
Epoch 5/16
1562/1562 [==============================] - 53s 34ms/step - loss: 1.5988 -
accuracy: 0.4226
Epoch 6/16
1562/1562 [==============================] - 55s 35ms/step - loss: 1.5733 -
accuracy: 0.4308
Epoch 7/16
1562/1562 [==============================] - 62s 39ms/step - loss: 1.5712 -
accuracy: 0.4333
Epoch 8/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.5525 -
accuracy: 0.4396
Epoch 9/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.5411 -
accuracy: 0.4522
Epoch 10/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.5293 -
accuracy: 0.4477
Epoch 11/16
1562/1562 [==============================] - 60s 39ms/step - loss: 1.5218 -
accuracy: 0.4491
Epoch 12/16
1562/1562 [==============================] - 58s 37ms/step - loss: 1.4960 -
accuracy: 0.4639
```

```
Epoch 13/16
1562/1562 [==============================] - 64s 41ms/step - loss: 1.5001 -
accuracy: 0.4621
Epoch 14/16
1562/1562 [==============================] - 65s 42ms/step - loss: 1.4826 -
accuracy: 0.4685
Epoch 15/16
1562/1562 [==============================] - 64s 41ms/step - loss: 1.4774 -
accuracy: 0.4712
Epoch 16/16
1562/1562 [==============================] - 53s 34ms/step - loss: 1.4679 -
accuracy: 0.4743
313/313 [==============================] - 2s 5ms/step - loss: 178.5847 -
accuracy: 0.4082
```

[15]:
```
model.save('results/6_2b_model.h5')
```

[16]:
```
prediction_results = model.predict(x_test)
```

[17]:
```
#write 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]))
```

[18]:
```
predictions = pd.DataFrame(prediction_results,␣
 ↪columns=['0','1','2','3','4','5','6','7','8','9'])
predictions.to_csv('results/6_2b_predictions.csv', index=False)
```