assignment06-2b_muley_tushar

January 8, 2022

Name: Tushar Muley

Assignment: Assignment 6-2b

Date:January 9, 2022

Assignment 6.2b 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.

[1]:
```
# download data
from keras.datasets import cifar10
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
```

[2]:
```
# libraries
import pandas as pd
import matplotlib.pyplot as plt
```

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

[4]:
```
# check the data volume for training
x_train.shape, y_train.shape
```

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

[5]:
```
# check the data volume for test
x_test.shape, y_test.shape
```

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

[6]:
```
# preprocess the data
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 10K samples for validation
x_val = x_train[-10000:]
y_val = y_train[-10000:]
partial_x_train = x_train[:-10000]
partial_y_train = y_train[:-10000]
```

[10]:
```
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(partial_x_train, partial_y_train,
                                     batch_size=32)

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

[11]:
```
# instantiate the model
from keras import models
from keras import layers
```

[12]:
```
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'))
```

[13]:
```
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
_________________________________________________________________
```

[14]:
```
from keras import optimizers
model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[15]:
```
history = model.fit_generator(train_generator,
                              steps_per_epoch=len(partial_x_train) / 32,
                              epochs=30,
                              validation_data=validation_generator,
                              validation_steps=len(x_val) / 32)
```

```
C:\Users\Tushar\AppData\Roaming\Python\Python38\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 1/30
1250/1250 [==============================] - 60s 47ms/step - loss: 2.2583 -
accuracy: 0.1430 - val_loss: 1.9999 - val_accuracy: 0.2568
Epoch 2/30
1250/1250 [==============================] - 62s 50ms/step - loss: 1.9972 -
accuracy: 0.2494 - val_loss: 1.8848 - val_accuracy: 0.3030
Epoch 3/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.9050 -
accuracy: 0.2826 - val_loss: 1.8627 - val_accuracy: 0.3095
Epoch 4/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.8612 -
accuracy: 0.3009 - val_loss: 1.8021 - val_accuracy: 0.3351
```

```
Epoch 5/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.8156 -
accuracy: 0.3192 - val_loss: 1.7502 - val_accuracy: 0.3557
Epoch 6/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.7806 -
accuracy: 0.3404 - val_loss: 1.7236 - val_accuracy: 0.3691
Epoch 7/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.7681 -
accuracy: 0.3456 - val_loss: 1.6977 - val_accuracy: 0.3761
Epoch 8/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.7446 -
accuracy: 0.3568 - val_loss: 1.6772 - val_accuracy: 0.3939
Epoch 9/30
1250/1250 [==============================] - 61s 49ms/step - loss: 1.7183 -
accuracy: 0.3696 - val_loss: 1.6555 - val_accuracy: 0.4047
Epoch 10/30
1250/1250 [==============================] - 61s 49ms/step - loss: 1.6886 -
accuracy: 0.3797 - val_loss: 1.6210 - val_accuracy: 0.4179
Epoch 11/30
1250/1250 [==============================] - 61s 49ms/step - loss: 1.6771 -
accuracy: 0.3872 - val_loss: 1.6175 - val_accuracy: 0.4104
Epoch 12/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.6608 -
accuracy: 0.3956 - val_loss: 1.5951 - val_accuracy: 0.4259
Epoch 13/30
1250/1250 [==============================] - 62s 50ms/step - loss: 1.6458 -
accuracy: 0.4024 - val_loss: 1.5915 - val_accuracy: 0.4257
Epoch 14/30
1250/1250 [==============================] - 63s 50ms/step - loss: 1.6321 -
accuracy: 0.4082 - val_loss: 1.5732 - val_accuracy: 0.4314
Epoch 15/30
1250/1250 [==============================] - 58s 47ms/step - loss: 1.6223 -
accuracy: 0.4096 - val_loss: 1.5489 - val_accuracy: 0.4376
Epoch 16/30
1250/1250 [==============================] - 54s 43ms/step - loss: 1.6057 -
accuracy: 0.4191 - val_loss: 1.5541 - val_accuracy: 0.4433
Epoch 17/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6026 -
accuracy: 0.4219 - val_loss: 1.5387 - val_accuracy: 0.4476
Epoch 18/30
1250/1250 [==============================] - 55s 44ms/step - loss: 1.5769 -
accuracy: 0.4283 - val_loss: 1.5222 - val_accuracy: 0.4552
Epoch 19/30
1250/1250 [==============================] - 53s 42ms/step - loss: 1.5702 -
accuracy: 0.4296 - val_loss: 1.5036 - val_accuracy: 0.4639
Epoch 20/30
1250/1250 [==============================] - 54s 43ms/step - loss: 1.5645 -
accuracy: 0.4354 - val_loss: 1.5043 - val_accuracy: 0.4595
```

```
Epoch 21/30
1250/1250 [==============================] - 56s 45ms/step - loss: 1.5555 -
accuracy: 0.4385 - val_loss: 1.4785 - val_accuracy: 0.4710
Epoch 22/30
1250/1250 [==============================] - 53s 42ms/step - loss: 1.5421 -
accuracy: 0.4427 - val_loss: 1.4754 - val_accuracy: 0.4779
Epoch 23/30
1250/1250 [==============================] - 50s 40ms/step - loss: 1.5369 -
accuracy: 0.4494 - val_loss: 1.4709 - val_accuracy: 0.4753
Epoch 24/30
1250/1250 [==============================] - 50s 40ms/step - loss: 1.5217 -
accuracy: 0.4532 - val_loss: 1.4534 - val_accuracy: 0.4818
Epoch 25/30
1250/1250 [==============================] - 50s 40ms/step - loss: 1.5191 -
accuracy: 0.4543 - val_loss: 1.4482 - val_accuracy: 0.4783
Epoch 26/30
1250/1250 [==============================] - 50s 40ms/step - loss: 1.5038 -
accuracy: 0.4554 - val_loss: 1.4608 - val_accuracy: 0.4853
Epoch 27/30
1250/1250 [==============================] - 50s 40ms/step - loss: 1.5010 -
accuracy: 0.4602 - val_loss: 1.4553 - val_accuracy: 0.4801
Epoch 28/30
1250/1250 [==============================] - 52s 42ms/step - loss: 1.4935 -
accuracy: 0.4627 - val_loss: 1.4310 - val_accuracy: 0.4837
Epoch 29/30
1250/1250 [==============================] - 56s 45ms/step - loss: 1.4863 -
accuracy: 0.4663 - val_loss: 1.4354 - val_accuracy: 0.4914
Epoch 30/30
1250/1250 [==============================] - 62s 50ms/step - loss: 1.4890 -
accuracy: 0.4674 - val_loss: 1.4205 - val_accuracy: 0.5007
```

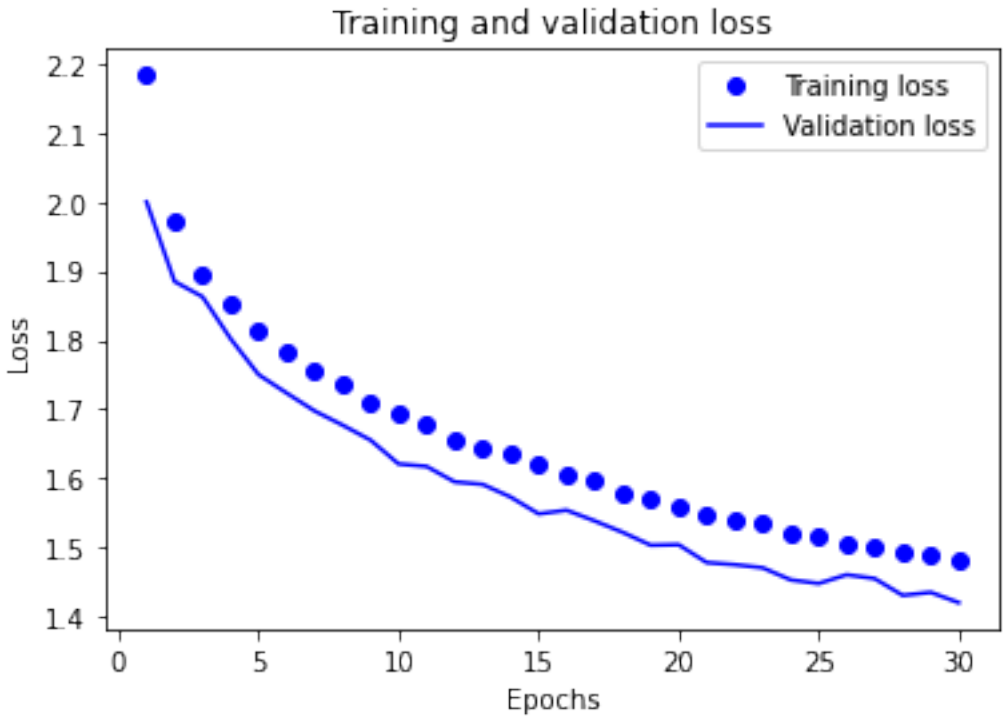
[16]:
```
history_dict = history.history
history_dict.keys()
```

[16]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

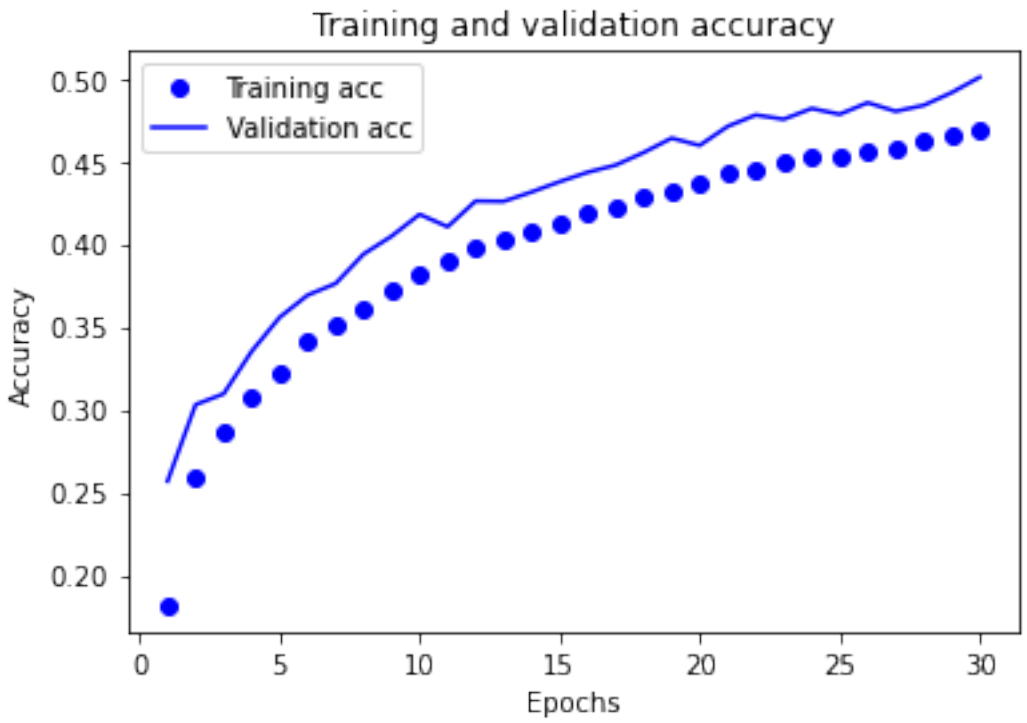
[17]:
```
# plot the training and validation loss

history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
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()
```



[18]:
```
# plot the training and validation accuracy
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[19]:
```
# retrain the model
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'])

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 [==============================] - 61s 38ms/step - loss: 1.4690 -
accuracy: 0.4749
Epoch 2/16
1562/1562 [==============================] - 61s 39ms/step - loss: 1.4774 -
accuracy: 0.4704
Epoch 3/16
1562/1562 [==============================] - 59s 38ms/step - loss: 1.4617 -
accuracy: 0.4783
Epoch 4/16
1562/1562 [==============================] - 59s 38ms/step - loss: 1.4572 -
accuracy: 0.4773
Epoch 5/16
```

```
1562/1562 [==============================] - 60s 39ms/step - loss: 1.4420 -
accuracy: 0.4835
Epoch 6/16
1562/1562 [==============================] - 61s 39ms/step - loss: 1.4284 -
accuracy: 0.4907
Epoch 7/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.4361 -
accuracy: 0.4879
Epoch 8/16
1562/1562 [==============================] - 66s 42ms/step - loss: 1.4229 -
accuracy: 0.4887
Epoch 9/16
1562/1562 [==============================] - 68s 43ms/step - loss: 1.4268 -
accuracy: 0.4926
Epoch 10/16
1562/1562 [==============================] - 63s 40ms/step - loss: 1.4205 -
accuracy: 0.4928
Epoch 11/16
1562/1562 [==============================] - 63s 41ms/step - loss: 1.4097 -
accuracy: 0.4974
Epoch 12/16
1562/1562 [==============================] - 61s 39ms/step - loss: 1.3999 -
accuracy: 0.5010
Epoch 13/16
1562/1562 [==============================] - 62s 40ms/step - loss: 1.4002 -
accuracy: 0.5011
Epoch 14/16
1562/1562 [==============================] - 60s 38ms/step - loss: 1.3925 -
accuracy: 0.5030
Epoch 15/16
1562/1562 [==============================] - 60s 38ms/step - loss: 1.3854 -
accuracy: 0.5091
Epoch 16/16
1562/1562 [==============================] - 60s 39ms/step - loss: 1.3960 -
accuracy: 0.5014
313/313 [==============================] - 3s 9ms/step - loss: 237.8977 -
accuracy: 0.3425
```

[20]:
```
# print results
results
```

[20]:
```
[237.89772033691406, 0.3425000011920929]
```

[21]:
```
# generate predictions on new data
model.predict(x_test)
```

```
[21]: array([[1., 0., 0., …, 0., 0., 0.],
             [0., 1., 0., …, 0., 0., 0.],
             [1., 0., 0., …, 0., 0., 0.],
             …,
             [0., 0., 0., …, 0., 0., 0.],
             [1., 0., 0., …, 0., 0., 0.],
             [0., 0., 0., …, 1., 0., 0.]], dtype=float32)
```