Dog Breed Identification

Theresa Shevenock

https://www.kaggle.com/c/dog-breed-identification (https://www.kaggle.com/c/dog-breed-identification)

Overview

The dog breed identification problem tasks challengers with creating a neural networ that will correctly identify the breed of dog out of 120 choices.

Data

The dataset that Kaggle provided for the dog breed identification problem consisted of: 10222 training images, 10222 validation images, and one .csv file containing the labels of the images.

For this neural network, the images in the training file provided by kaggle was split up into the training set and the validation set. This was done by using the file names, for example0a1f8334a9f583cac009dc033c681e47 in the .csv file, searching though the trainign folder and placing it in the proper category in the new training folder. The same process was follwed to categorize and fill the validation set. The files were renamed according to the number that corresponded with breed and the number of times the code has repeated over the code with an underscore(_) seperating the two.

The file structure for the training and validation sets consisted of 120 folders labelled 1 - 20. Each breed was given a number by the code and is used to determine where the images would be placed.

Training: 8877 images

Validation: 1808 images

Example of training and Validation images:

Bull Terrier

data_generators were used to create the training and validation sets

Training

A convolutional neural network was used in the training of the data.

model.compile(optimizer = optimizers.SGD(lr=0.01, momentum=0., decay=0., nesterov=False),

```
            loss='categorical_crossentropy',

            metrics=['accuracy'])
```

Since data_generators were used, to train the network, model.fit_generator() was used

image.png

Accuracy

Training Accuracy: 99% (Highest)

Validation Accuracy: 87% (One epoch reach 100%)

image.png

Problems

A major issue I faced in completing this project was a bug that I encountered using Google Drive. After i had downloaded the data for the nueral network, google drive started to send images to the trash without my permission. I had gone down from 10222 images to 6000 images in one minute. The code required a certain amount of images to be present and I had to continuously recover the deleted images.

When i finally got the program to populate the folders i encountered another issue. The code did not save the file with the extension .jpg so the program was not able to train. The data had to be redownloaded and the code to move had to be rerun in order to populate the data folder once again.

The last issue was discovered when validation accuracy did not rise above 20%. The validation dataset turned out to be incorrectly labelled. The problem was resolved by relabelling the data with the fixed categorizing code. The same thing happened with the training dataset, though on a lesser level. This was fixed by following the steps for the validation dataset to clear up the noise in the dataset.

In [3]:
```
import keras
keras.__version__

from google.colab import drive
drive.mount('/gdrive')

from PIL import Image
import os
import shutil
import numpy as np
import pandas as pd
from keras import models
from keras import layers
from keras.layers import Dense, BatchNormalization
from keras.layers import Dropout
from sklearn.model_selection import train_test_split
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras import backend
base_dir = '/gdrive/My Drive/Dog_Breed_Identification'
dogs = pd.read_csv("/gdrive/My Drive/Dog_Breed_Identification/labels.csv")
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client
_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&
redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2F
www.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fau
th%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=
code

Enter your authorization code:
··········
Mounted at /gdrive
```

In [0]:
```
##only run code between the '##' once to seprate files into categories. All ot
her runs, ignore.
```

In [0]:
```
## https://github.com/kjaisingh/dog-breed-classifier/blob/master/cnn.py used l
ines 13 - 53 for labeling images from the csv file ##
## change code to point to correct directories in gdrive as well as sending to
 new directories for the labelled images
##only need to run once to sepreate the images uncomment to run again

## changes denoted by '###' to the right of code line
references = np.array(dogs.iloc[:, :])
references = references.astype(str)
```

In [0]:
```
breeds = []
breedCateogry = []
j = 1
for i in range(0, 10222): ### was 10222, changed due to loss of images in gdri
ve (encountered a bug is gdrive)
    if references[i][1] not in breeds:
        breeds.append(references[i][1])
        breedCateogry.append(j)
        j += 1

newCategories = [0] * 10222  ### was 10222, changed due to loss of images in g
drive (encountered a bug is gdrive)
```

In [0]:
```
file_names = os.listdir(base_dir +"/train")

for i in range(0,10222): ### was 10222, changed due to loss of images in gdriv
e (encountered a bug is gdrive)
    substring = file_names[i].split('.')[0]
    print(i)
    for j in range(0,10222): ### was 10222, changed due to loss of images in g
drive (encountered a bug is gdrive)
        if(substring == references[j][0]):
            cat = references[j][1]
            for k in range(0, 120):
                if(breeds[k] == cat):
                    newCategories[i] = breedCateogry[k]
```

In [0]:
```
finalFileNames = []
finalCategories = []
for i in range(0,10222):  ### was 10222, changed due to loss of images in gdri
ve (encountered a bug is gdrive)
    finalFileNames.append(file_names[i])
    finalCategories.append(newCategories[i])
```

In [0]:
```
X_train, X_cv, y_train, y_cv = train_test_split(finalFileNames, finalCategorie
s, test_size = 0.2)

for i in range(0, 8222): ### was 8177, changed due to loss of images in gdrive
 (encountered a bug is gdrive)
    shutil.move(base_dir +"/train/" + X_train[i], base_dir +"/trainNEW/" + str
(y_train[i]) + "/" + str(y_train[i]) + "_" + str(i) + ".jpg") ### originally w
ithout '.jpg'
    print(i)
                                                                ### so it did no
t save as image

                                                                ### generators w
ould then detect 120 classes with no images
z = 1
for i in range(0, 2000): ### was 2045, changed due to loss of images in gdrive
 (encountered a bug is gdrive)
    z = i
    shutil.move(base_dir +"/train/" + X_cv[i], base_dir +"/testNEW/" + str(y_c
v[i])+ "/" + str(y_cv[i]) + "_" + str(i) + ".jpg")### added in to fix issue
    print(i)


### Additional note for above ranges. In order to deal with missing images the
 '0' was manually hanged after each error
### in order to get the maximum amount of data for the sets as possible
```

In [4]:
```
if backend.image_data_format() == 'channels_first':
    input_shape = (3, 255, 255)
else:
    input_shape = (255, 255, 3)

## end use of github for image labeling ##
train_dir = os.path.join('/gdrive/My Drive/Dog_Breed_Identification/trainNEW')
test_dir = os.path.join( '/gdrive/My Drive/Dog_Breed_Identification/testNEW')

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_from_directory(
        '/gdrive/My Drive/Dog_Breed_Identification/trainNEW',
        target_size=(255, 255),
        batch_size=28#,
        #class_mode='categorical'
)

validation_generator = test_datagen.flow_from_directory(
        '/gdrive/My Drive/Dog_Breed_Identification/testNEW',
        target_size=(255, 255),
        batch_size=300#79#,
       #class_mode='categorical'
)

for data_batch, labels_batch in train_generator:
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
    break

for data_batch, labels_batch in validation_generator:
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
    break
```

```
Found 8177 images belonging to 120 classes.
Found 1808 images belonging to 120 classes.
data batch shape: (28, 255, 255, 3)
labels batch shape: (28, 120)
data batch shape: (300, 255, 255, 3)
labels batch shape: (300, 120)
```

In [11]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=input_shape))
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.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(layers.Dense(120, activation='softmax')) #softmax due to multiple ou
tcomes

model.compile(optimizer = optimizers.SGD(lr=0.01, momentum=0., decay=0., neste
rov=False),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit_generator(
      train_generator,
      steps_per_epoch=317, # training sample # / batchsize
      epochs = 50,
      validation_data=validation_generator,
      validation_steps=1, # validation sample #
      verbose=1)
```

```
Epoch 1/50
317/317 [==============================] - 88s 279ms/step - loss: 4.7319 - ac
c: 0.0186 - val_loss: 5.2175 - val_acc: 0.0233
Epoch 2/50
317/317 [==============================] - 75s 236ms/step - loss: 4.5350 - ac
c: 0.0354 - val_loss: 4.9267 - val_acc: 0.0200
Epoch 3/50
317/317 [==============================] - 76s 241ms/step - loss: 4.3557 - ac
c: 0.0526 - val_loss: 4.3333 - val_acc: 0.0500
Epoch 4/50
317/317 [==============================] - 76s 241ms/step - loss: 4.2253 - ac
c: 0.0701 - val_loss: 4.8442 - val_acc: 0.0400
Epoch 5/50
317/317 [==============================] - 75s 238ms/step - loss: 4.1157 - ac
c: 0.0909 - val_loss: 4.1969 - val_acc: 0.0000e+00
Epoch 6/50
317/317 [==============================] - 76s 240ms/step - loss: 3.9972 - ac
c: 0.1078 - val_loss: 3.9966 - val_acc: 0.0833
Epoch 7/50
317/317 [==============================] - 75s 238ms/step - loss: 3.8760 - ac
c: 0.1289 - val_loss: 4.0709 - val_acc: 0.0900
Epoch 8/50
317/317 [==============================] - 74s 235ms/step - loss: 3.7280 - ac
c: 0.1533 - val_loss: 4.0459 - val_acc: 0.1200
Epoch 9/50
317/317 [==============================] - 73s 230ms/step - loss: 3.5709 - ac
c: 0.1924 - val_loss: 3.8288 - val_acc: 0.1367
Epoch 10/50
317/317 [==============================] - 73s 231ms/step - loss: 3.3942 - ac
c: 0.2288 - val_loss: 4.2459 - val_acc: 0.0967
Epoch 11/50
317/317 [==============================] - 74s 234ms/step - loss: 3.2081 - ac
c: 0.2707 - val_loss: 3.4540 - val_acc: 0.2133
Epoch 12/50
317/317 [==============================] - 73s 230ms/step - loss: 2.9571 - ac
c: 0.3338 - val_loss: 3.5712 - val_acc: 0.2500
Epoch 13/50
317/317 [==============================] - 73s 231ms/step - loss: 2.7140 - ac
c: 0.4046 - val_loss: 3.1531 - val_acc: 0.2967
Epoch 14/50
317/317 [==============================] - 73s 229ms/step - loss: 2.4272 - ac
c: 0.4791 - val_loss: 2.7991 - val_acc: 0.3833
Epoch 15/50
317/317 [==============================] - 73s 231ms/step - loss: 2.0620 - ac
c: 0.5889 - val_loss: 2.3375 - val_acc: 0.4533
Epoch 16/50
317/317 [==============================] - 72s 226ms/step - loss: 1.7221 - ac
c: 0.6824 - val_loss: 2.5791 - val_acc: 0.4300
Epoch 17/50
317/317 [==============================] - 73s 229ms/step - loss: 1.3735 - ac
c: 0.7749 - val_loss: 1.9019 - val_acc: 0.6500
Epoch 18/50
317/317 [==============================] - 73s 231ms/step - loss: 1.0405 - ac
c: 0.8596 - val_loss: 1.7845 - val_acc: 0.6233
Epoch 19/50
317/317 [==============================] - 72s 226ms/step - loss: 0.7252 - ac
c: 0.9277 - val_loss: 1.2568 - val_acc: 0.7500
```

```
Epoch 20/50
317/317 [==============================] - 73s 230ms/step - loss: 0.4940 - ac
c: 0.9602 - val_loss: 1.1485 - val_acc: 0.7933
Epoch 21/50
317/317 [==============================] - 72s 228ms/step - loss: 0.3081 - ac
c: 0.9866 - val_loss: 1.2045 - val_acc: 0.8100
Epoch 22/50
317/317 [==============================] - 73s 231ms/step - loss: 0.2049 - ac
c: 0.9928 - val_loss: 1.1262 - val_acc: 0.8133
Epoch 23/50
317/317 [==============================] - 71s 224ms/step - loss: 0.1416 - ac
c: 0.9945 - val_loss: 1.0611 - val_acc: 0.8067
Epoch 24/50
317/317 [==============================] - 73s 230ms/step - loss: 0.1029 - ac
c: 0.9958 - val_loss: 1.0356 - val_acc: 0.8133
Epoch 25/50
317/317 [==============================] - 72s 226ms/step - loss: 0.0880 - ac
c: 0.9956 - val_loss: 0.8285 - val_acc: 0.8400
Epoch 26/50
317/317 [==============================] - 71s 225ms/step - loss: 0.0879 - ac
c: 0.9927 - val_loss: 0.3934 - val_acc: 0.8750
Epoch 27/50
317/317 [==============================] - 73s 230ms/step - loss: 0.0644 - ac
c: 0.9958 - val_loss: 1.1607 - val_acc: 0.7867
Epoch 28/50
317/317 [==============================] - 73s 230ms/step - loss: 0.0633 - ac
c: 0.9955 - val_loss: 0.9743 - val_acc: 0.8400
Epoch 29/50
317/317 [==============================] - 74s 235ms/step - loss: 0.0574 - ac
c: 0.9956 - val_loss: 0.8983 - val_acc: 0.8167
Epoch 30/50
317/317 [==============================] - 74s 232ms/step - loss: 0.0568 - ac
c: 0.9952 - val_loss: 1.0610 - val_acc: 0.8167
Epoch 31/50
317/317 [==============================] - 75s 235ms/step - loss: 0.0494 - ac
c: 0.9958 - val_loss: 0.9718 - val_acc: 0.8267
Epoch 32/50
317/317 [==============================] - 72s 229ms/step - loss: 0.0482 - ac
c: 0.9954 - val_loss: 0.8627 - val_acc: 0.8500
Epoch 33/50
317/317 [==============================] - 73s 231ms/step - loss: 0.0495 - ac
c: 0.9952 - val_loss: 0.0201 - val_acc: 1.0000
Epoch 34/50
317/317 [==============================] - 73s 230ms/step - loss: 0.0413 - ac
c: 0.9961 - val_loss: 0.9230 - val_acc: 0.8367
Epoch 35/50
317/317 [==============================] - 74s 233ms/step - loss: 0.0445 - ac
c: 0.9952 - val_loss: 1.0341 - val_acc: 0.8100
Epoch 36/50
317/317 [==============================] - 73s 230ms/step - loss: 0.0402 - ac
c: 0.9956 - val_loss: 1.1718 - val_acc: 0.7933
Epoch 37/50
317/317 [==============================] - 72s 226ms/step - loss: 0.0416 - ac
c: 0.9955 - val_loss: 0.9174 - val_acc: 0.8300
Epoch 38/50
317/317 [==============================] - 73s 230ms/step - loss: 0.0396 - ac
c: 0.9955 - val_loss: 1.2227 - val_acc: 0.7967
```

```
Epoch 39/50
317/317 [==============================] - 76s 240ms/step - loss: 0.0526 - ac
c: 0.9922 - val_loss: 0.7622 - val_acc: 0.8700
Epoch 40/50
317/317 [==============================] - 75s 237ms/step - loss: 0.0355 - ac
c: 0.9958 - val_loss: 1.0930 - val_acc: 0.8750
Epoch 41/50
317/317 [==============================] - 75s 237ms/step - loss: 0.0375 - ac
c: 0.9955 - val_loss: 1.1297 - val_acc: 0.8033
Epoch 42/50
317/317 [==============================] - 74s 233ms/step - loss: 0.0368 - ac
c: 0.9952 - val_loss: 1.0864 - val_acc: 0.8033
Epoch 43/50
317/317 [==============================] - 74s 232ms/step - loss: 0.0333 - ac
c: 0.9958 - val_loss: 0.8499 - val_acc: 0.8467
Epoch 44/50
317/317 [==============================] - 72s 226ms/step - loss: 0.0386 - ac
c: 0.9951 - val_loss: 1.1965 - val_acc: 0.8067
Epoch 45/50
317/317 [==============================] - 74s 234ms/step - loss: 0.0326 - ac
c: 0.9958 - val_loss: 1.0014 - val_acc: 0.8233
Epoch 46/50
317/317 [==============================] - 74s 234ms/step - loss: 0.0331 - ac
c: 0.9955 - val_loss: 0.9954 - val_acc: 0.8400
Epoch 47/50
317/317 [==============================] - 75s 236ms/step - loss: 0.0335 - ac
c: 0.9957 - val_loss: 0.4123 - val_acc: 0.8750
Epoch 48/50
317/317 [==============================] - 76s 240ms/step - loss: 0.0323 - ac
c: 0.9956 - val_loss: 0.8799 - val_acc: 0.8533
Epoch 49/50
317/317 [==============================] - 76s 239ms/step - loss: 0.0321 - ac
c: 0.9956 - val_loss: 0.9068 - val_acc: 0.8533
Epoch 50/50
317/317 [==============================] - 76s 239ms/step - loss: 0.0303 - ac
c: 0.9956 - val_loss: 1.3155 - val_acc: 0.7933
```

In [0]:

```
model.save('DogBreedID.h5')
```

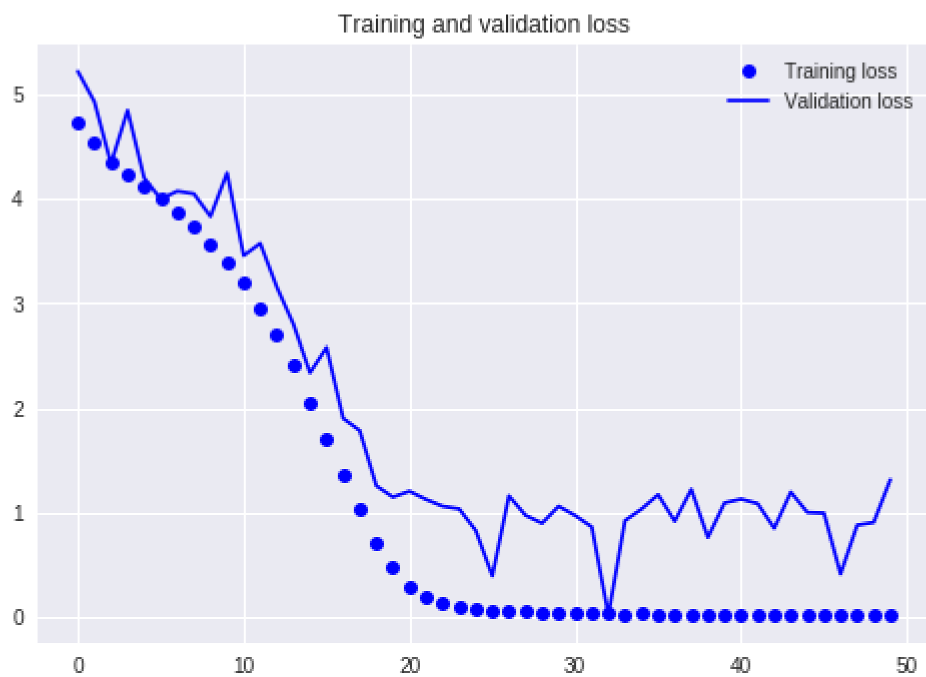
In [19]:

```
results = model.evaluate_generator(validation_generator,1)
print("Accuracy = ", results[1])
```

```
Accuracy =  0.8666666746139526
```

In [14]:
```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```


Training and validation accuracy
Training acc
Validation acc
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50


Training and validation loss
Training loss
Validation loss
5
4
3
2
1
0
0
10
20
30
40
50

In [15]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_10 (Conv2D)           (None, 253, 253, 32)      896       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 126, 126, 32)      0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 124, 124, 64)      18496     
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 62, 62, 64)        0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 60, 60, 64)        36928     
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 30, 30, 64)        0         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 128)       73856     
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 14, 14, 128)       0         
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 12, 12, 256)       295168    
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 6, 6, 256)         0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 4, 4, 256)         590080    
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 2, 2, 256)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0         
_________________________________________________________________
dense_5 (Dense)              (None, 256)               262400    
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024      
_________________________________________________________________
dense_6 (Dense)              (None, 120)               30840     
=================================================================
Total params: 1,309,688
Trainable params: 1,309,176
Non-trainable params: 512
_________________________________________________________________
```