Please note this kernel is for practice purposes only.

In this kernel I will be using AlexNet (https://en.wikipedia.org/wiki/AlexNet) for multiclass image classification.

Inferences from the given dataset description:

- There are 20,580 dogs images divided into 120 different categories (i.e., 120 breeds of dogs)

Steps followed in this kernel:

- Pick different categories of dog images for training the CNN model.
- After the training is done we will check the accuracy of the CNN model in predicting the dog's breed correctly given an image of the dog.

1. Load all the libraries

In [1]:
```
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
from keras.layers.normalization import BatchNormalization

print("Loaded all libraries")
```

```
Loaded all libraries


Using TensorFlow backend.
```

2. Data loading and exploration

In [2]:
```
fpath = "../input/images/Images/"
random_seed = 42

categories = os.listdir(fpath)
categories = categories[:20]
print("List of categories = ",categories,"\n\nNo. of categories = "
, len(categories))
```

```
List of categories =  ['n02105162-malinois', 'n02094258-Norwich_ter
rier', 'n02102177-Welsh_springer_spaniel', 'n02086646-Blenheim_span
iel', 'n02086910-papillon', 'n02093256-Staffordshire_bullterrier',
```

```
'n02113624-toy_poodle', 'n02105056-groenendael', 'n02109961-Eskimo_
dog', 'n02116738-African_hunting_dog', 'n02096177-cairn', 'n0209658
5-Boston_bull', 'n02100735-English_setter', 'n02102973-Irish_water_
spaniel', 'n02099429-curly-coated_retriever', 'n02088364-beagle',
'n02101006-Gordon_setter', 'n02108089-boxer', 'n02097130-giant_schn
auzer', 'n02112137-chow']

No. of categories =  20
```

In [3]:

```
def load_images_and_labels(categories):
    img_lst=[]
    labels=[]
    for index, category in enumerate(categories):
        for image_name in os.listdir(fpath+"/"+category):
            img = cv2.imread(fpath+"/"+category+"/"+image_name)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            img_array = Image.fromarray(img, 'RGB')

            #resize image to 227 x 227 because the input image resol
ution for AlexNet is 227 x 227
            resized_img = img_array.resize((227, 227))

            img_lst.append(np.array(resized_img))

            labels.append(index)
    return img_lst, labels

images, labels = load_images_and_labels(categories)
print("No. of images loaded = ",len(images),"\nNo. of labels loaded
= ",len(labels))
print(type(images),type(labels))
```

```
No. of images loaded =  3337
No. of labels loaded =  3337
<class 'list'> <class 'list'>
```

In [4]:

```
images = np.array(images)
labels = np.array(labels)

print("Images shape = ",images.shape,"\nLabels shape = ",labels.sha
pe)
print(type(images),type(labels))
```

```
Images shape =  (3337, 227, 227, 3)
Labels shape =  (3337,)
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

- Check few random images and labels by displaying them in a graph

In [5]:

```
def display_rand_images(images, labels):
    plt.figure(1 , figsize = (19 , 10))
    n = 0
```

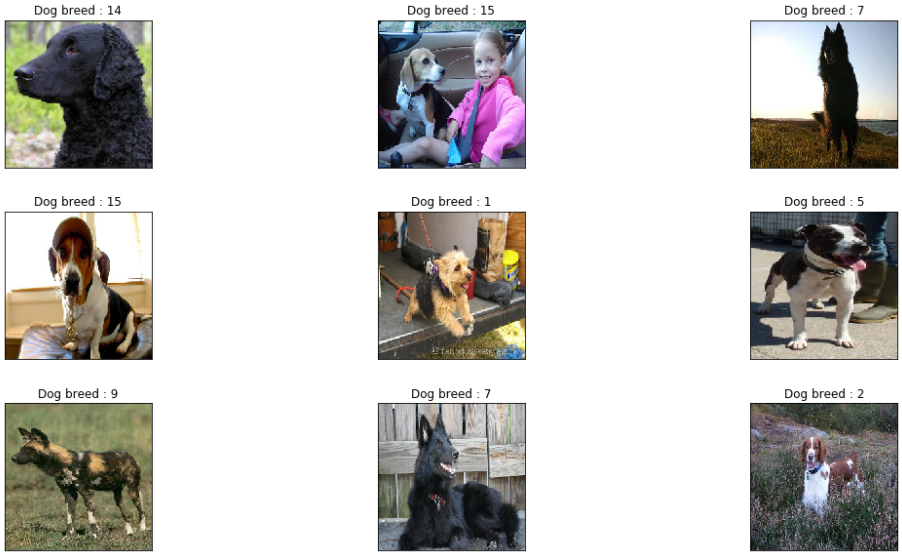
```
    for i in range(9):
        n += 1
        r = np.random.randint(0 , images.shape[0] , 1)

        plt.subplot(3 , 3 , n)
        plt.subplots_adjust(hspace = 0.3 , wspace = 0.3)
        plt.imshow(images[r[0]])

        plt.title('Dog breed : {}'.format(labels[r[0]]))
        plt.xticks([])
        plt.yticks([])

    plt.show()

display_rand_images(images, labels)
```



3. Prepare data for training the CNN model

- For training the CNN model we have to shuffle all the data that is loaded in images, labels list.

In [6]:

```
#1-step in data shuffling

#get equally spaced numbers in a given range
n = np.arange(images.shape[0])
print("'n' values before shuffling = ",n)

#shuffle all the equally spaced values in list 'n'
np.random.seed(random_seed)
np.random.shuffle(n)
print("\n'n' values after shuffling = ",n)
```

```
'n' values before shuffling =  [   0    1    2 ... 3334 3335 3336]

'n' values after shuffling =  [ 321 3095  727 ... 1294  860 3174]
```

In [7]:
```
#2-step in data shuffling

#shuffle images and corresponding labels data in both the lists
images = images[n]
labels = labels[n]

print("Images shape after shuffling = ",images.shape,"\nLabels shap
e after shuffling = ",labels.shape)
```

```
Images shape after shuffling =  (3337, 227, 227, 3)
Labels shape after shuffling =  (3337,)
```

- Data normalization

In [8]:
```
images = images.astype(np.float32)
labels = labels.astype(np.int32)
images = images/255
print("Images shape after normalization = ",images.shape)
```

```
Images shape after normalization =  (3337, 227, 227, 3)
```

- Display few random images after data normalization

In [9]:
```
display_rand_images(images, labels)
```

- Split the loaded dataset into train, test sets

In [10]:
```
x_train, x_test, y_train, y_test = train_test_split(images, labels,
test_size = 0.2, random_state = random_seed)

print("x_train shape = ",x_train.shape)
print("y_train shape = ",y_train.shape)
```

```
print("\nx_test shape = ",x_test.shape)
print("y_test shape = ",y_test.shape)
```

```
x_train shape =  (2669, 227, 227, 3)
y_train shape =  (2669,)

x_test shape =  (668, 227, 227, 3)
y_test shape =  (668,)
```

In [11]:

```
display_rand_images(x_train, y_train)
```

4. Define AlexNet CNN model

- Define all layers in the AlexNet CNN model

In [12]:

```
model=Sequential()

#1 conv layer
model.add(Conv2D(filters=96,kernel_size=(11,11),strides=(4,4),paddi
ng="valid",activation="relu",input_shape=(227,227,3)))

#1 max pool layer
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

model.add(BatchNormalization())

#2 conv layer
model.add(Conv2D(filters=256,kernel_size=(5,5),strides=(1,1),paddin
g="valid",activation="relu"))

#2 max pool layer
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

model.add(BatchNormalization())
```

```
model.add(BatchNormalization())

#3 conv layer
model.add(Conv2D(filters=384,kernel_size=(3,3),strides=(1,1),paddin
g="valid",activation="relu"))

#4 conv layer
model.add(Conv2D(filters=384,kernel_size=(3,3),strides=(1,1),paddin
g="valid",activation="relu"))

#5 conv layer
model.add(Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),paddin
g="valid",activation="relu"))

#3 max pool layer
model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))

model.add(BatchNormalization())


model.add(Flatten())

#1 dense layer
model.add(Dense(4096,input_shape=(227,227,3),activation="relu"))

model.add(Dropout(0.4))

model.add(BatchNormalization())

#2 dense layer
model.add(Dense(4096,activation="relu"))

model.add(Dropout(0.4))

model.add(BatchNormalization())

#3 dense layer
model.add(Dense(1000,activation="relu"))

model.add(Dropout(0.4))

model.add(BatchNormalization())

#output layer
model.add(Dense(20,activation="softmax"))

model.summary()
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tens
orflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/kera
s/backend/tensorflow_backend.py:3445: calling dropout (from tensorf
low.python.ops.nn_ops) with keep_prob is deprecated and will be rem
oved in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `ra
```

```
te = 1 - keep_prob`.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 55, 55, 96)        34944
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 27, 27, 96)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 27, 27, 96)        384
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 23, 23, 256)       614656
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 256)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 9, 384)         885120
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 384)         1327488
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 5, 5, 256)         884992
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              4198400
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_4 (Batch (None, 4096)              16384
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_5 (Batch (None, 4096)              16384
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dropout_3 (Dropout)          (None, 1000)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1000)              4000
_________________________________________________________________
dense_4 (Dense)              (None, 20)                20020
=================================================================
Total params: 28,883,132
Trainable params: 28,863,532
Non-trainable params: 19,600
_________________________________________________________________
```

- Compile the CNN model

In [13]:

```
model.compile(optimizer="adam", loss="sparse_categorical_crossentro
py", metrics=["accuracy"])
```

5. Train the model

- Fit the model using training data

In [14]:
```
%%time
model.fit(x_train, y_train, epochs=100)
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/tens
orflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.pytho
n.ops.math_ops) is deprecated and will be removed in a future versi
on.
Instructions for updating:
Use tf.cast instead.
Epoch 1/100
2669/2669 [==============================] - 8s 3ms/step - loss: 3.
4869 - acc: 0.1034
Epoch 2/100
2669/2669 [==============================] - 3s 1ms/step - loss: 3.
1226 - acc: 0.1278
Epoch 3/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
9775 - acc: 0.1487
Epoch 4/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
8394 - acc: 0.1712
Epoch 5/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
7636 - acc: 0.1888
Epoch 6/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
7037 - acc: 0.2016
Epoch 7/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
6212 - acc: 0.2068
Epoch 8/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
5784 - acc: 0.2196
Epoch 9/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
4898 - acc: 0.2308
Epoch 10/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
4467 - acc: 0.2375
Epoch 11/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
3674 - acc: 0.2698
Epoch 12/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
3088 - acc: 0.2825
Epoch 13/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
2494 - acc: 0.2900
Epoch 14/100
```

```
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
1742 - acc: 0.3080
Epoch 15/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
1548 - acc: 0.3245
Epoch 16/100
2669/2669 [==============================] - 3s 1ms/step - loss: 2.
0757 - acc: 0.3421
Epoch 17/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
9860 - acc: 0.3720
Epoch 18/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
8994 - acc: 0.3810
Epoch 19/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
8457 - acc: 0.4061
Epoch 20/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
7737 - acc: 0.4215
Epoch 21/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
8074 - acc: 0.4125
Epoch 22/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
6133 - acc: 0.4792
Epoch 23/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
5159 - acc: 0.5051
Epoch 24/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
4338 - acc: 0.5257
Epoch 25/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
3129 - acc: 0.5643
Epoch 26/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
2373 - acc: 0.5995
Epoch 27/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
0869 - acc: 0.6373
Epoch 28/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
9776 - acc: 0.6767
Epoch 29/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
0310 - acc: 0.6557
Epoch 30/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
8480 - acc: 0.7123
Epoch 31/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
6950 - acc: 0.7628
Epoch 32/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
7164 - acc: 0.7636
Epoch 33/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
5980 - acc: 0.8048
Epoch 34/100
```

```
Epoch 34/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
5337 - acc: 0.8213
Epoch 35/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
4588 - acc: 0.8449
Epoch 36/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
4066 - acc: 0.8587
Epoch 37/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
4646 - acc: 0.8471
Epoch 38/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3716 - acc: 0.8741
Epoch 39/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3064 - acc: 0.9003
Epoch 40/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3934 - acc: 0.8685
Epoch 41/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
9287 - acc: 0.4706
Epoch 42/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
9472 - acc: 0.6875
Epoch 43/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
9284 - acc: 0.6965
Epoch 44/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
8388 - acc: 0.7257
Epoch 45/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
4792 - acc: 0.8456
Epoch 46/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3194 - acc: 0.9045
Epoch 47/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2686 - acc: 0.9090
Epoch 48/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2564 - acc: 0.9202
Epoch 49/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2178 - acc: 0.9281
Epoch 50/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2147 - acc: 0.9314
Epoch 51/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1998 - acc: 0.9314
Epoch 52/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1756 - acc: 0.9371
Epoch 53/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1945 - acc: 0.9329
```

```
Epoch 54/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2205 - acc: 0.9183
Epoch 55/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2696 - acc: 0.9030
Epoch 56/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2580 - acc: 0.9153
Epoch 57/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
7208 - acc: 0.8262
Epoch 58/100
2669/2669 [==============================] - 3s 1ms/step - loss: 1.
0705 - acc: 0.6782
Epoch 59/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3509 - acc: 0.8820
Epoch 60/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2033 - acc: 0.9363
Epoch 61/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1612 - acc: 0.9479
Epoch 62/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1332 - acc: 0.9569
Epoch 63/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1242 - acc: 0.9580
Epoch 64/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1117 - acc: 0.9659
Epoch 65/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1334 - acc: 0.9528
Epoch 66/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1038 - acc: 0.9652
Epoch 67/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1410 - acc: 0.9543
Epoch 68/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1381 - acc: 0.9558
Epoch 69/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2320 - acc: 0.9292
Epoch 70/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1810 - acc: 0.9404
Epoch 71/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1304 - acc: 0.9554
Epoch 72/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1587 - acc: 0.9505
Epoch 73/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1215 - acc: 0.9599
```

```
Epoch 74/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1347 - acc: 0.9520
Epoch 75/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1161 - acc: 0.9644
Epoch 76/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1489 - acc: 0.9513
Epoch 77/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1324 - acc: 0.9592
Epoch 78/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1210 - acc: 0.9580
Epoch 79/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
0931 - acc: 0.9682
Epoch 80/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1053 - acc: 0.9674
Epoch 81/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
0885 - acc: 0.9726
Epoch 82/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1057 - acc: 0.9618
Epoch 83/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2130 - acc: 0.9281
Epoch 84/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1264 - acc: 0.9599
Epoch 85/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1514 - acc: 0.9520
Epoch 86/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1286 - acc: 0.9573
Epoch 87/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
0922 - acc: 0.9700
Epoch 88/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
0664 - acc: 0.9771
Epoch 89/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
0865 - acc: 0.9741
Epoch 90/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
3682 - acc: 0.8973
Epoch 91/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
2416 - acc: 0.9217
Epoch 92/100
2669/2669 [==============================] - 3s 1ms/step - loss: 0.
1733 - acc: 0.9475
Epoch 93/100
```