Prediction with the saved CNN model

```
# Imports
import numpy as np
import cv2
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
import keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras import backend as K


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive
```

```
# Link Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive
```

```
model = load_model('/content/drive/MyDrive/model_cnn/save_NN_model.model')
model.summary()

img_width = 28
img_height = 28
```

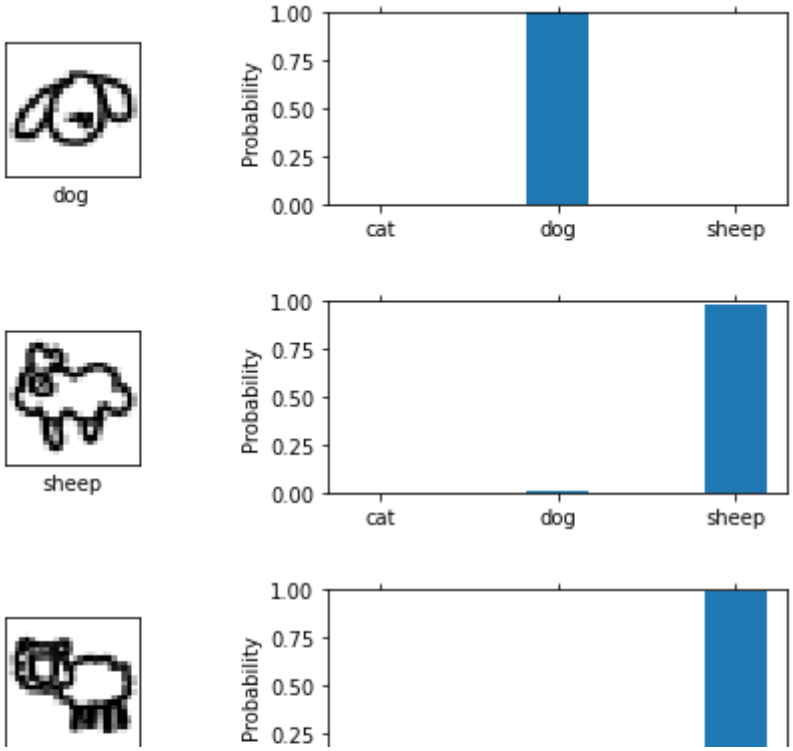
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 24, 24, 30)        780

max_pooling2d_2 (MaxPooling  (None, 12, 12, 30)        0
2D)

conv2d_3 (Conv2D)            (None, 10, 10, 15)        4065

max_pooling2d_3 (MaxPooling  (None, 5, 5, 15)          0
2D)

dropout_1 (Dropout)          (None, 5, 5, 15)          0
```

```
flatten_1 (Flatten)         (None, 375)               0

dense_3 (Dense)             (None, 128)               48128

dense_4 (Dense)             (None, 50)                6450

dense_5 (Dense)             (None, 3)                 153

=================================================================
Total params: 59,576
Trainable params: 59,576
Non-trainable params: 0
_________________________________________________________________
```

```
# Load the data
cat = np.load('/content/drive/MyDrive/1 2564/CP461 Computer Vision/Lab/Lab14/data_cat.npy')
dog = np.load('/content/drive/MyDrive/1 2564/CP461 Computer Vision/Lab/Lab14/data_dog.npy')
sheep = np.load('/content/drive/MyDrive/1 2564/CP461 Computer Vision/Lab/Lab14/data_sheep.npy')

# Add a column with labels, 0=cat, 1=dog, 2=sheep
cat = np.c_[cat, np.zeros(len(cat))]
dog = np.c_[dog, np.ones(len(dog))]
sheep = np.c_[sheep, 2*np.ones(len(sheep))]

# merge the cat and sheep arrays,
# and split the features (X) and labels (y).
# Convert to float32 to save some memory.
X = np.concatenate((cat[:6000,:-1], dog[:6000,:-1], sheep[:6000,:-1]), axis=0).astype('float32') # all columns b
y = np.concatenate((cat[:6000,-1], dog[:6000,-1], sheep[:6000,-1]), axis=0).astype('float32') # the last column

# train/test split (divide by 255 to obtain normalized values between 0 and 1)
# I will use a 50:50 split,
X_train, X_test, y_train, y_test = train_test_split(X/255.,y,test_size=0.5,random_state=0)

# one hot encode outputs
y_train_cnn = np_utils.to_categorical(y_train)
y_test_cnn = np_utils.to_categorical(y_test)
num_classes = y_test_cnn.shape[1]

# reshape to be [samples][pixels][width][height]
# X_train_cnn = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
# X_test_cnn = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')

#"tensorflow"
X_train_cnn = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test_cnn = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')



import random

# store the label codes in a dictionary
label_dict = {0:'cat', 1:'dog', 2:'sheep'}
```

```
# CNN predictions
cnn_probab = model.predict(X_test_cnn, batch_size=32, verbose=0)

# Plotting the X_test data and finding out the probabilites of prediction
fig, ax = plt.subplots(figsize=(7,15))

for i in list(range(6)):

    rand_i = random.randint(0,cnn_probab.shape[0])

    print("The drawing is identified as --> ", label_dict[y_test[rand_i]], " <-- with a probability of ", max(cnn_pro

    # plot picture:
    ax = plt.subplot2grid((6, 5), (i, 0), colspan=1);
    plt.imshow(X_test[rand_i].reshape((28,28)),cmap='gray_r', interpolation='nearest');
    plt.xlabel(label_dict[y_test[rand_i]]); # get the label from the dict
    plt.xticks([])
    plt.yticks([])

    # plot probabilities:
    ax = plt.subplot2grid((6, 5), (i, 2), colspan=4);
    plt.bar(np.arange(3), cnn_probab[rand_i], 0.35, align='center');
    plt.xticks(np.arange(3), ['cat', 'dog', 'sheep'])
    plt.tick_params(axis='x', bottom='off', top='off')
    plt.ylabel('Probability')
    plt.ylim(0,1)
    plt.subplots_adjust(hspace = 0.5)
```

```
The drawing is identified as -->  dog  <-- with a probability of  99.99595880508423
The drawing is identified as -->  sheep  <-- with a probability of  98.62594604492188
The drawing is identified as -->  sheep  <-- with a probability of  98.63422513008118
The drawing is identified as -->  dog  <-- with a probability of  99.65910911560059
The drawing is identified as -->  cat  <-- with a probability of  99.99388456344604
The drawing is identified as -->  cat  <-- with a probability of  99.99979734420776
```



Test from your drawing

```
img = cv2.imread('/content/drive/MyDrive/model_cnn/Capture.JPG', 0)
#ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
img = cv2.resize(img, (img_width, img_height))
# cv2.resize()
plt.imshow((img.reshape((28,28))), cmap='gray_r')

#print img, "\n"
arr = np.array(img-255)
#print arr
arr = np.array(arr/255.)
#print arr

# new_test_cnn = arr.reshape(1, 1, 28, 28).astype('float32')
new_test_cnn = arr.reshape(1, 28, 28, 1).astype('float32')
print(new_test_cnn.shape)
```

```
(1, 28, 28, 1)
```

Predict, what is your drawing

```
import operator
# CNN predictions
new_cnn_predict = model.predict(new_test_cnn, batch_size=32, verbose=0)

pr = model.predict(arr.reshape((1, 28, 28, 1)))

# Plotting the X_test data and finding out the probabilites of prediction
fig, ax = plt.subplots(figsize=(8,3))

# Finding the max probability
max_index, max_value = max(enumerate(new_cnn_predict[0]), key=operator.itemgetter(1))

print("The drawing is identified as --> ", label_dict[max_index], " <-- with a probability of ", max_value*100)

for i in list(range(1)):
    # plot probabilities:
    ax = plt.subplot2grid((1, 5), (i, 2), colspan=4);
    plt.bar(np.arange(3), new_cnn_predict[i], 0.35, align='center');
    plt.xticks(np.arange(3), ['cat', 'dog', 'sheep'])
    plt.tick_params(axis='x', bottom='off', top='off')
    plt.ylabel('Probability')
    plt.ylim(0,1)
    plt.subplots_adjust(hspace = 0.5)

     # plot picture:
    ax = plt.subplot2grid((1, 5), (i, 1), colspan=1);
    plt.imshow((img.reshape(28,28)), cmap='gray_r')
    plt.xticks([])
    plt.yticks([])
```

```
The drawing is identified as -->  cat  <-- with a probability of  89.26838040351868
```