IMAGE CLASSIFICATION ( CATS AND DOGS )

##### 15IT322E- PYTHON PROGRAMMING PROJECT REPORT

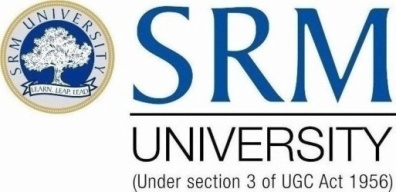
###### ***Submitted by***

##### RAKSHIT VARU- RA1511008010058

**DEVANSH GOEKA-**

*for the assessment of Semester V Minor Project*

# 



SRM UNIVERSITY

KATTANKULATHUR

##### October 2017

**DECLARATION**

I ………………………….(Name of the Student with Reg. No.) studying in III year B.Tech Information Technology program at, SRM University, Kattankulathur, Chennai, hereby declare that this project is an original work of mine and I have not verbatim copied / duplicated any material from sources like internet or from print media, excepting some vital company information / statistics and data that is provided by the company itself.

Signature of the Student

Date:

Place:

**Abstract**

A human can distinguish between cats and dogs very easily but a computer system will find it a bit difficult task.

We proposed a CNN- based recognition method on cats and dogs. Our algorithm uses **Keras** - a deep learning library for the neural networks. We performed on the dataset given by Kaggle on cats and dogs with accuracy of over 85.5%. We can also implement the algorithm to classify between many different objects which might help people with less knowledge about the details of the object.

The algorithm gives the predictions as either 1 or 0. 1 is considered to be dog and 0 as cat.

The image is first preprocessed and then sent as an input for the input layer of the convolution neural network. The data is processed in hidden layers of the CNNs.

The other python libraries used are scipy, matplotlib, PIL.

**TABLE OF CONTENTS**

1. **Introduction……………………………………………………5**
   1. Deep Learning……………………………..………………..5
   2. Data Sets…………………………..………………………...5
   3. Convolutional Neural Network..……………………………6
   4. Libraries……….……………………………………………7
      1. Keras...………………………………………………..7
      2. PIL...………………………………………………….7
      3. Scipy………………………...………………………..7
      4. Matplotlib……...…………………………………......8
   5. Accurac……………………………………………………..8
2. **Code…………………………………………………………….8**
3. **Output………………………………………………………….15**
4. **References……………………………………………………..17**
5. **Introduction**

The algorithm deals with the classification between the images of Cats and Dogs. The algorithm is based on Deep Learning.

**1.1. Deep Learning**

Deep learning (also known as deep structured learningor hierarchical learning) is part of a broader family of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) methods based on [learning data representations](https://en.wikipedia.org/wiki/Learning_representation), as opposed to task-specific algorithms. Learning can be [supervised](https://en.wikipedia.org/wiki/Supervised_learning), partially supervised or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning).

Deep learning architectures such as [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks), [deep belief networks](https://en.wikipedia.org/wiki/Deep_belief_network) and [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks) have been applied to fields including [computer vision](https://en.wikipedia.org/wiki/Computer_vision), [speech recognition](https://en.wikipedia.org/wiki/Automatic_speech_recognition), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), audio recognition, social network filtering, [machine translation](https://en.wikipedia.org/wiki/Machine_translation) and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics) where they produced results comparable to and in some cases superior to human experts.

Deep learning is a class of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [algorithms](https://en.wikipedia.org/wiki/Algorithm) that:

* use a cascade of multiple layers of [nonlinear processing](https://en.wikipedia.org/wiki/Nonlinear_filter) units for [feature extraction](https://en.wikipedia.org/wiki/Feature_extraction) and transformation. Each successive layer uses the output from the previous layer as input.
* learn in [supervised](https://en.wikipedia.org/wiki/Supervised_learning) (e.g., classification) and/or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning) (e.g., pattern analysis) manners.
* learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
* use some form of [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) for training via [backpropagation](https://en.wikipedia.org/wiki/Backpropagation" \o "Backpropagation).

**1.2 Data Sets**

A training data directory and validation data directory containing one subdirectory per image class, filled with .png or .jpg images:

data/

train/

dogs/

dog001.jpg

dog002.jpg

...

cats/

cat001.jpg

cat002.jpg

...

validation/

dogs/

dog001.jpg

dog002.jpg

...

cats/

cat001.jpg

cat002.jpg

...

We use two sets of pictures, which we got from Kaggle: 12,500 cats and 12,500 dogs. We also use 3200 additional samples from each class as validation data, to evaluate our models.

That is very few examples to learn from, for a classification problem that is far from simple. So this is a challenging machine learning problem, but it is also a realistic one: in a lot of real-world use cases, even small-scale data collection can be extremely expensive or sometimes near-impossible (e.g. in medical imaging). Being able to make the most out of very little data is a key skill of a competent data scientist.

**1.3 Convolutional Neural Network**

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

Convolutional networks were [inspired](https://en.wikipedia.org/wiki/Mathematical_biology) by [biological](https://en.wikipedia.org/wiki/Biological" \o "Biological)processes[[4]](https://en.wikipedia.org/wiki/Convolutional_neural_network" \l "cite_note-robust_face_detection-4) in which the connectivity pattern between [neurons](https://en.wikipedia.org/wiki/Artificial_neuron) is inspired by the organization of the animal [visual cortex](https://en.wikipedia.org/wiki/Visual_cortex). Individual [cortical neurons](https://en.wikipedia.org/wiki/Cortical_neuron) respond to stimuli only in a restricted region of the [visual field](https://en.wikipedia.org/wiki/Visual_field) known as the [receptive field](https://en.wikipedia.org/wiki/Receptive_field). The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other [image classification algorithms](https://en.wikipedia.org/w/index.php?title=Image_classification_algorithm&action=edit&redlink=1). This means that the network learns the [filters](https://en.wikipedia.org/w/index.php?title=Filter_(image_processing)&action=edit&redlink=1) that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

**1.4 Libraries**

**1.4.1 Keras**

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](https://github.com/tensorflow/tensorflow), [CNTK](https://github.com/Microsoft/cntk), or [Theano](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

* Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
* Supports both convolutional networks and recurrent networks, as well as combinations of the two.
* Runs seamlessly on CPU and GPU.

**1.4.2 PIL**

The Python Imaging Library (PIL) adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.

The [**Image**](http://pillow.readthedocs.io/en/3.4.x/reference/Image.html#module-PIL.Image) module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory functions, including functions to load images from files, and to create new images.

**1.4.3 Scipy**

**SciPy** (pronounced "Sigh Pie") is an [open source](https://en.wikipedia.org/wiki/Open_source) [Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language))library used for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing)and technical computing.

SciPy contains modules for [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)), [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), [integration](https://en.wikipedia.org/wiki/Integral), [interpolation](https://en.wikipedia.org/wiki/Interpolation), [special functions](https://en.wikipedia.org/wiki/Special_functions), [FFT](https://en.wikipedia.org/wiki/Fast_Fourier_transform), [signal](https://en.wikipedia.org/wiki/Signal_processing" \o "Signal processing)and [image processing](https://en.wikipedia.org/wiki/Image_processing), [ODE](https://en.wikipedia.org/wiki/Ordinary_differential_equation) solvers and other tasks common in science and engineering.

SciPy builds on the [NumPy](https://en.wikipedia.org/wiki/NumPy" \o "NumPy) array object and is part of the NumPy stack which includes tools like [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib" \o "Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)) and [SymPy](https://en.wikipedia.org/wiki/SymPy" \o "SymPy), and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as [MATLAB](https://en.wikipedia.org/wiki/MATLAB), [GNU Octave](https://en.wikipedia.org/wiki/GNU_Octave), and [Scilab](https://en.wikipedia.org/wiki/Scilab" \o "Scilab). The NumPy stack is also sometimes referred to as the SciPy stack.

**1.4.4 Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shell, the [jupyter](http://jupyter.org/index.html) notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail](https://matplotlib.org/gallery/index.html) gallery.

**1.5 Accuracy**

Our Algorithm works with an accuracy of around 83.34%. The accuracy can be increased by changing the algorithm or model. The loss is around 0.37 and over all validation accuracy is around 83.19 %.

1. **Code**

**cnn.py**

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras import backend as K

from keras.models import model\_from\_json

# Initialising the CNN

classifier = Sequential()

# Step 1 - Convolution #fliters - dimension#3x3

classifier.add(Convolution2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a convolutional layer

classifier.add(Convolution2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

#dataset/train\_set

training\_set = train\_datagen.flow\_from\_directory('E:/machinelearning/Data/train',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

#dataset/test\_set

test\_set = test\_datagen.flow\_from\_directory('E:/machinelearning/Data/test1',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

classifier.fit\_generator(training\_set,

samples\_per\_epoch = 800,

nb\_epoch = 25,

validation\_data = test\_set,

nb\_val\_samples = 2000)

# Saving the weights to a json file

model\_json= classifier.to\_json()

with open("model.json", "w") as json\_file:

json\_file.write(model\_json)

classifier.save\_weights("model.h5")

print("Saved model to disk")

**load cats and dogs.py**

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras import backend as K

from keras.models import model\_from\_json

from PIL import Image,ImageChops, ImageOps

import matplotlib.pyplot as plt

from keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from scipy.misc import imresize

# Initialising the CNN

classifier = Sequential()

# Convolution #fliters - dimension#3x3

classifier.add(Convolution2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

classifier.add(Convolution2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection power of 2

classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Part 2 - Fitting the CNN to the images

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True

test\_datagen = ImageDataGenerator(rescale = 1./255)

#dataset/train\_set

training\_set = train\_datagen.flow\_from\_directory('E:/machinelearning/Data/train',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

#dataset/test\_set

test\_set = test\_datagen.flow\_from\_directory('E:/machinelearning/Data/test1',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

# pretrained model loaded

json\_file= open('model.json', 'r')

loaded\_model\_json= json\_file.read()

json\_file.close()

loaded\_model= model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("model.h5")

print("Loaded model from disk")

loaded\_model.compile(loss='binary\_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

#image for prediction

path = "E:/machinelearning/Data/test1/3213.jpg"

img = load\_img(path)

plt.imshow(img)

img = imresize(img, (64, 64))

img = img\_to\_array(img)

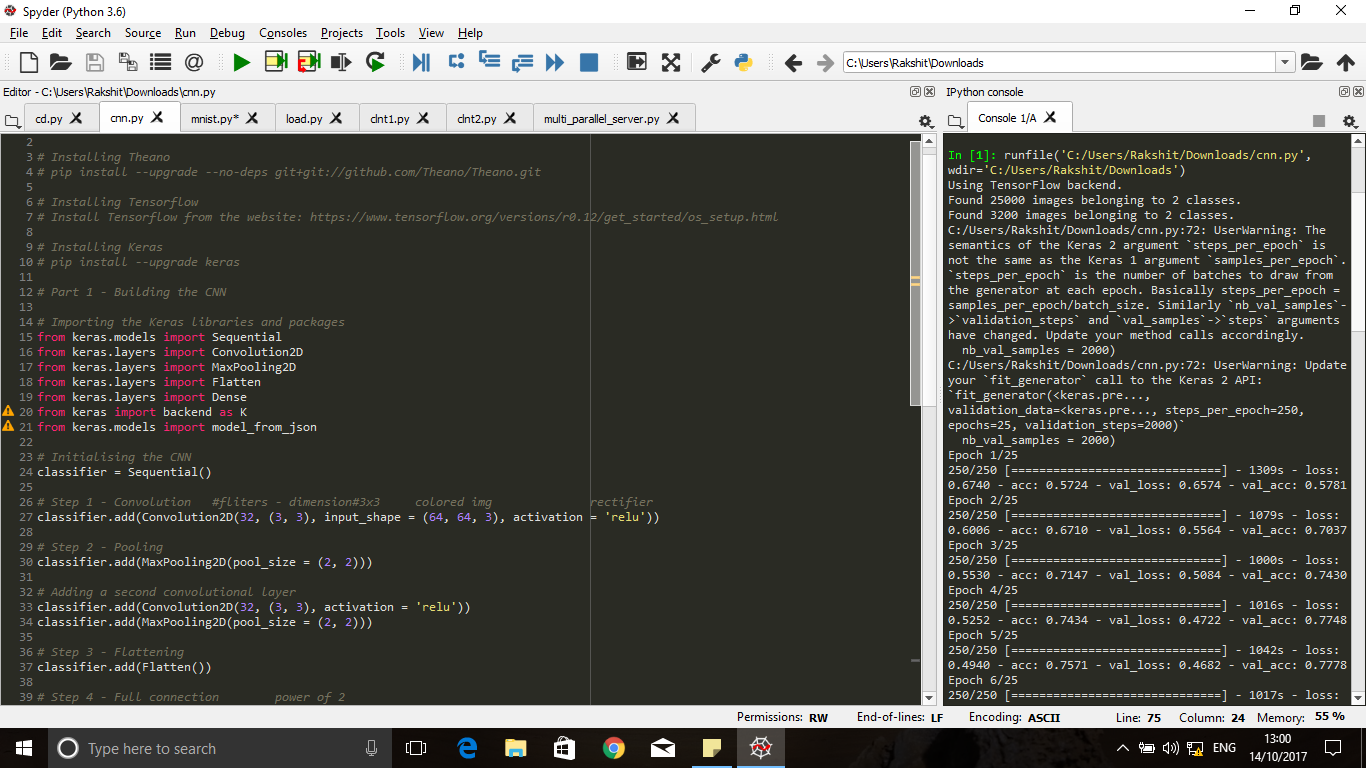
img = img.reshape(1, 64, 64, 3)

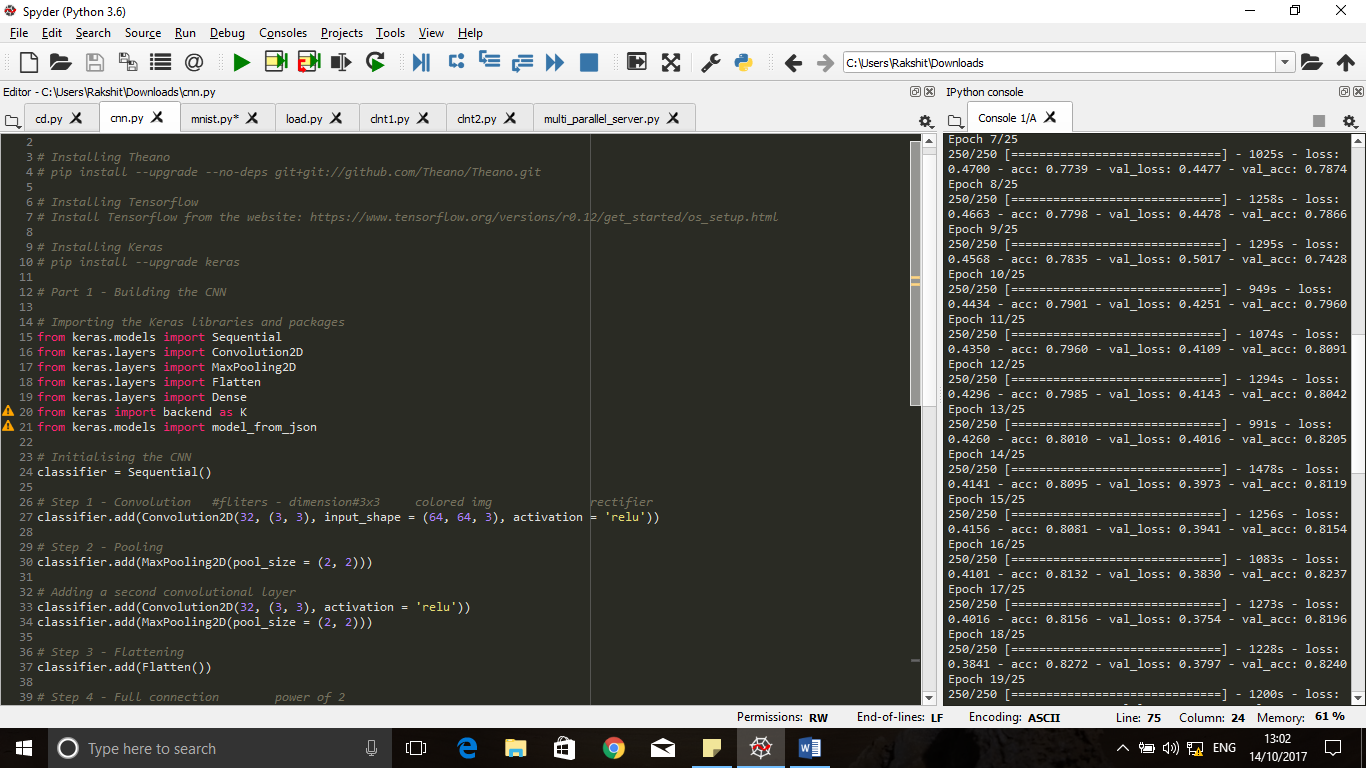
classes = ["dog", "cat"]

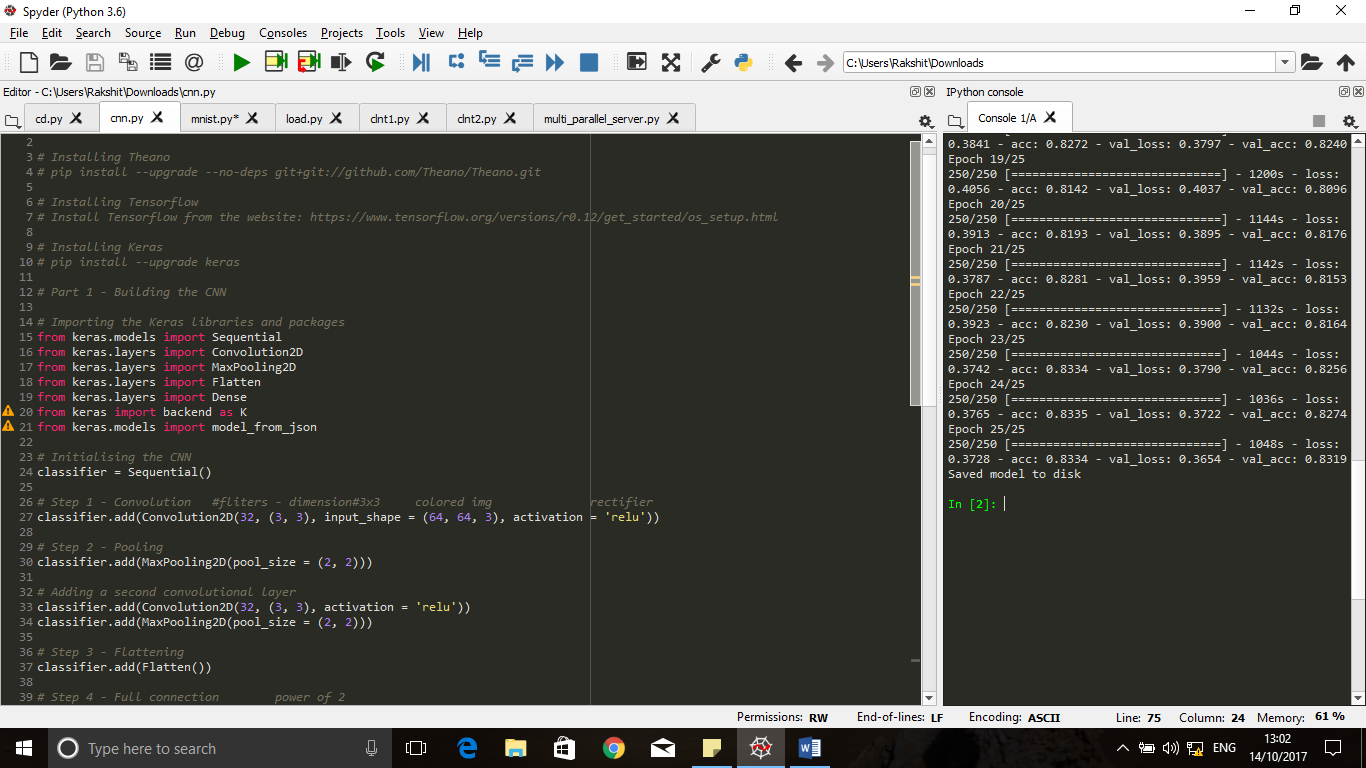
print(classifier.predict\_classes(img))

1. **Output**

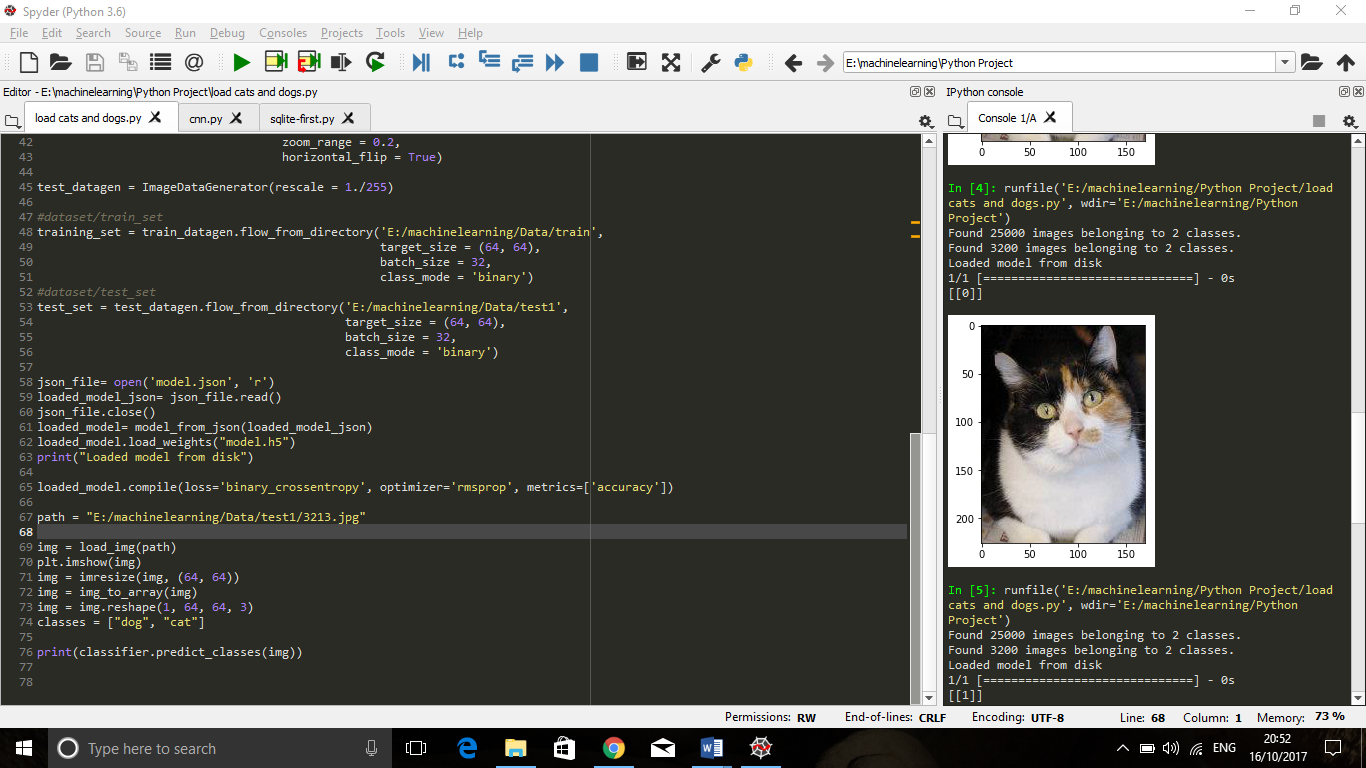
**Cnn.py**

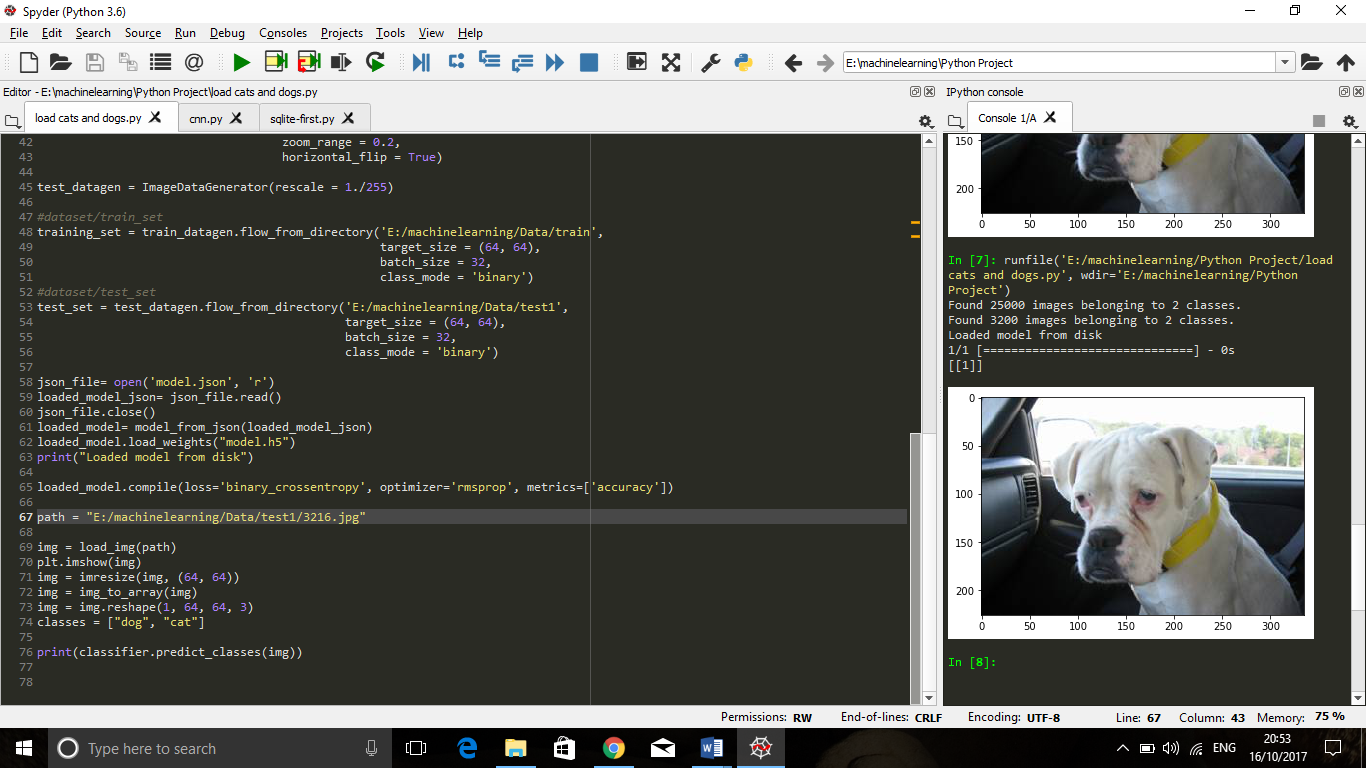






**load cats and dogs.py**





1. **References**
2. <https://www.kaggle.com/c/dogs-vs-cats/data>
3. [https://keras.io/models/sequential](https://keras.io/models/sequential/)