



CLASSIFICATION USING CONVOLUTION NEURAL NETWORK

Supervised Learning



Prepared By-

Abhinav Rai

Harshitha S

Submission Date: 8th Dec,2018



Prepared By-

Abhinav Rai

Harshitha S

Course: Computational Machine Learning (Aug 2018- Dec 2018)

Faculty:

Dr Vijaya Kumar Beekanahalli

Abhijith Chandrababhu

Contents

NEURAL NETWORK	3
PROJECT OBJECTIVE	3
TECHNOLOGY USED	3
CONVOLUTION NEURAL NETWORK.....	3
LEARNING ALGORITHM/NEURAL NETWORK	4
SAMPLE OUTPUT	8
CONCLUSION	9
FUTURE SCOPE	9

NEURAL NETWORK

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example.

An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.




PROJECT OBJECTIVE

The objective of this project is to solve the classification problem on the identification of Cats and Dogs from by image processing. The training sample contains the images of either Cats or Dogs.

The classification problem will be solved by using the Convolutional Neural Network technology which will be used via learning algorithm in order classify the given Images.

The training sample taken to perform the initial training the Neural Network.

TECHNOLOGY USED

-  PYTHON 3.0
-  TENSOR FLOW 1.0
-  OPEN CV 3.4.4

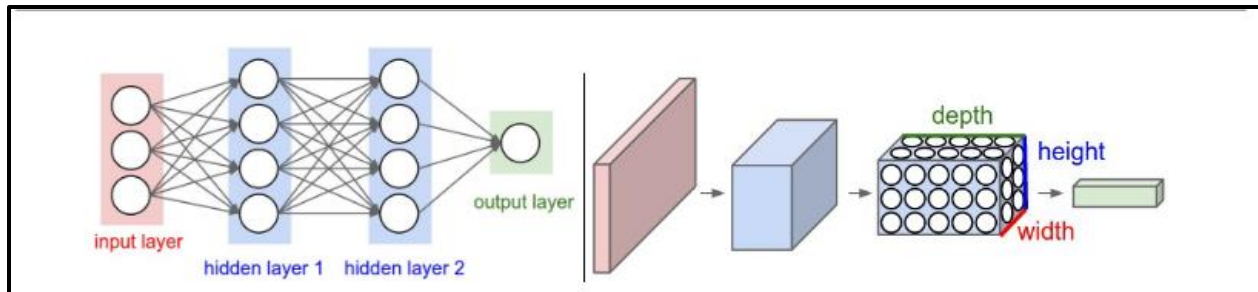
CONVOLUTION NEURAL NETWORK

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the 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. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.



(CNN ARCHITECTURE)

LEARNING ALGORITHM/NEURAL NETWORK

```
import cv2                # working with, mainly resizing, images
import numpy as np        # dealing with arrays
import os                 # dealing with directories
from random import shuffle

# mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm     # a nice pretty percentage bar for tasks.

TRAIN_DIR = 'data/input/train'
TEST_DIR = 'data/input/test'
IMG_SIZE = 50
LR = 1e-3

MODEL_NAME = 'dogsvscats-{}-{}.model'.format(LR, '2conv-basic')

# just so we remember which saved model is which, sizes must match
#-----

def label_img(img):
    word_label = img.split('.')[0]

    # conversion to one-hot array [cat,dog] # [much cat, no dog]

    if word_label == 'cat': return [1,0]

    # [no cat, very doggo]
```

```
elif word_label == 'dog': return [0,1]

#-----

def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR, img)
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        training_data.append([np.array(img), np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data

#-----

def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR, img)
        img_num = img.split('.')[0]
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        testing_data.append([np.array(img), img_num])
    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data

train_data = create_train_data()

# If you have already created the dataset:
# train_data = np.load('train_data.npy')

#-----
```

```
import tflearn

from tflearn.layers.conv import conv_2d, max_pool_2d

from tflearn.layers.core import input_data, dropout, fully_connected

from tflearn.layers.estimator import regression

#-----

#Convolution Neural Network

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

#Layer 1

convnet = conv_2d(convnet, 32, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)

#Layer 2

convnet = conv_2d(convnet, 64, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)

#Layer 3

convnet = conv_2d(convnet, 32, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)

#Layer 4

convnet = conv_2d(convnet, 64, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)

#Layer 5

convnet = conv_2d(convnet, 32, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)

#Layer 6

convnet = conv_2d(convnet, 64, 5, activation='relu')

convnet = max_pool_2d(convnet, 5)


convnet = fully_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
```

```
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy',
name='targets')
```

```
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

```
#-----
```

```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
```

```
    model.load(MODEL_NAME)
```

```
    print('model loaded!')
```

```
train = train_data[:500]
```

```
test = train_data[500:]
```

```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
```

```
Y = [i[1] for i in train]
```

```
test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
```

```
test_y = [i[1] for i in test]
```

```
model.fit({'input': X}, {'targets': Y}, n_epoch=4, validation_set=({'input': test_x}, {'targets': test_y}),
```

```
    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
model.save(MODEL_NAME)
```

```
#-----
```

```
import matplotlib.pyplot as plt
```

```
# if you need to create the data:
```

```
test_data = process_test_data()
```

```
# if you already have some saved:
```

```
#test_data = np.load('test_data.npy')
```

```
fig=plt.figure()
```

```
for num,data in enumerate(test_data[:12]):
```

```
    # cat: [1,0]
```

```
    # dog: [0,1]
```

```
    img_num = data[1]
```

```
    img_data = data[0]
```



```

y = fig.add_subplot(3,4,num+1)

orig = img_data

data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)

#model_out = model.predict([data])[0]

model_out = model.predict([data])[0]

if np.argmax(model_out) == 1:str_label='Dog'
else: str_label='Cat'

y.imshow(orig,cmap='gray')

plt.title(str_label)

y.axes.get_xaxis().set_visible(False)

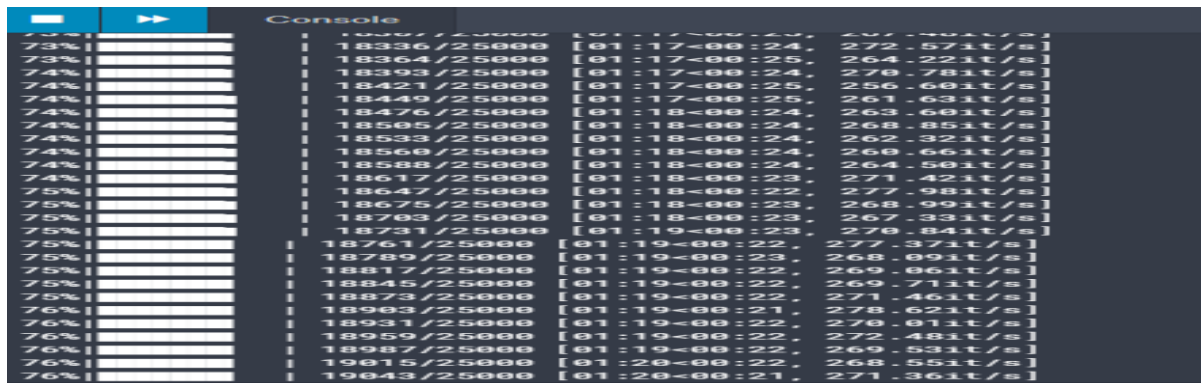
y.axes.get_yaxis().set_visible(False)

plt.show()

```

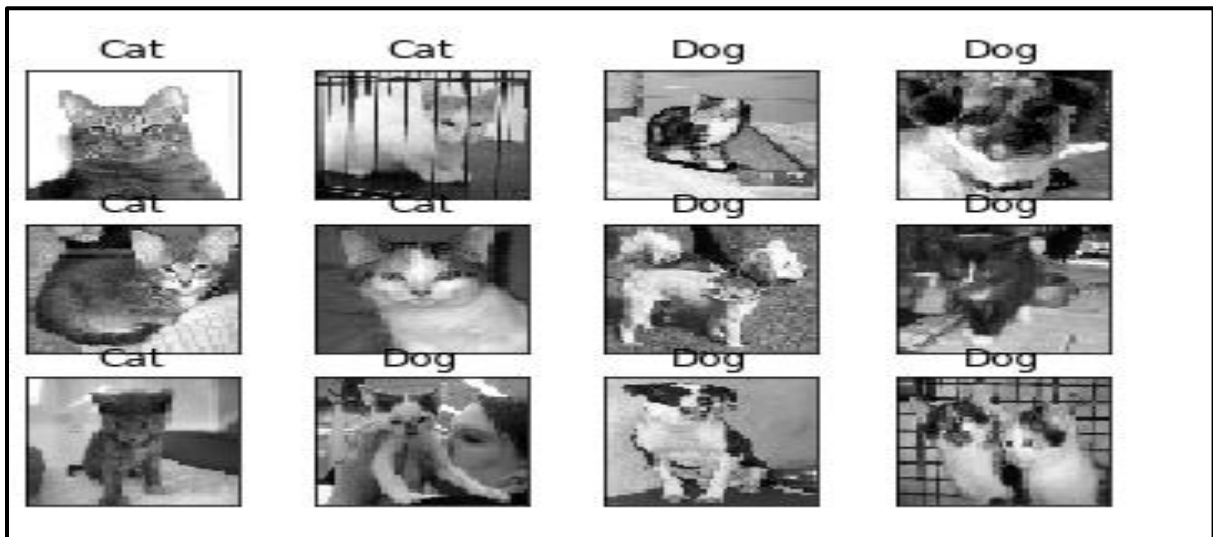
SAMPLE OUTPUT

🚦 Training/Learning Stage



37% | 9223/25000 [00:17<00:28, 550.78it/s]

🚦 Sample Output



CONCLUSION

Neural Network was designed which can classify Dogs and Cats in the given Images was achieved by using the Convolution Neural Network. However, there are some inconsistency was observed in classification.

FUTURE SCOPE

The below are some future scope of this project:

- 🚦 Increase the accuracy.
- 🚦 Classify other objects like Human, Street etc in the given images
- 🚦 Classifying the objects while processing the live capturing of the surrounding environment.