# CNN as a Classifier and Transfer Learning

# Problem Definition

Image classification is the process of identifying or predicting the class or category of an image after extracting the features necessary for the classification. These features can be extracted manually, using feature extraction algorithms or, automatically, using Deep Learning techniques.

**Dataset Used**

We created our own dataset of 2514 images. It is divided into 3 classes i.e., cats, dogs and panda.They are of various size (height and width). We standardize the dimension of the image in the preprocessing phase.

**Techniques Used**

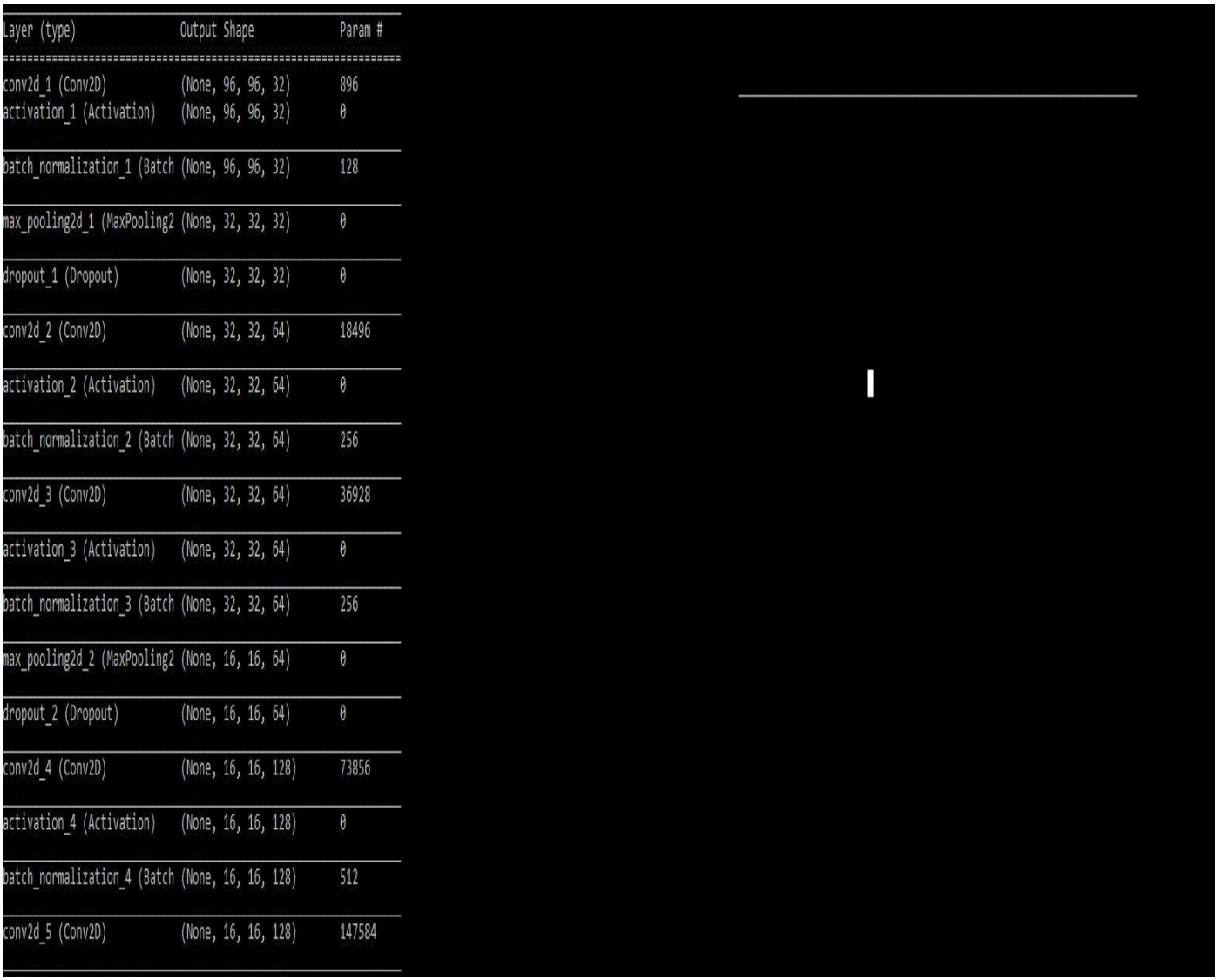
1. CNN as a classifier :
   We trained a 27 layer CNN. We implemented our own VGGNet using keras.
2. Visualizing the activation maps :
   We were able to generate activation maps of every layer. It helps understand the output of convolutional layer, activation layer and max pooling layer.
3. Using CNN to extract CNN codes :
   We were able to extract CNN code from the trained model of our own. These CNN codes can be as feature vectors. Classifier such as SVM, Knn etc. can be trained on it.

# Design

The project has been designed in a very modular manner. All the images undergo  A smaller version of VGG net was implemented in keras. It was trained for 70 epochs. All the images were resized to 96 x 96 dimension. The model consists of series of Convolutional Layer, Activation Layer, Batch Normalization and Max Pooling Layer. Flattening is done to obtain a feature vector followed by Dense layer to change the dimension of the feature vector.
The trained model was used for two purpose :

1. For classification using CNN.

2. To extract CNN code from the trained model. These CNN codes are stored in a csv files along with the labels.



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 96, 96, 32) | 896 |
| activation_1 (Activation) | (None, 96, 96, 32) | 0 |
| batch_normalization_1 (Batch | (None, 96, 96, 32) | 128 |
| max_pooling2d_1 (MaxPooling2 | (None, 32, 32, 32) | 0 |
| dropout_1 (Dropout) | (None, 32, 32, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 32, 32, 64) | 18496 |
| activation_2 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_2 (Batch | (None, 32, 32, 64) | 256 |
| conv2d_3 (Conv2D) | (None, 32, 32, 64) | 36928 |
| activation_3 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_3 (Batch | (None, 32, 32, 64) | 256 |
| max_pooling2d_2 (MaxPooling2 | (None, 16, 16, 64) | 0 |
| dropout_2 (Dropout) | (None, 16, 16, 64) | 0 |
| conv2d_4 (Conv2D) | (None, 16, 16, 128) | 73856 |
| activation_4 (Activation) | (None, 16, 16, 128) | 0 |
| batch_normalization_4 (Batch | (None, 16, 16, 128) | 512 |
| conv2d_5 (Conv2D) | (None, 16, 16, 128) | 147584 |

# Algorithm

train.py

- Load the network
- Load the dataset
- Preprocess the images suitable for the implemented network
- Train the model
- Save the model
- Save the labels

cnn_as_a_classifier.py

- Load the trained model along with the labels
- Load the image to be classified
- Preprocess the image
- Pass the image through the model
- Get the index of element having the highest probability in the softmax vector
- Get the label corresponding to that index

extract_feature_vector.py

- Load the trained model along with the labels
- Load the entire dataset again.
- Preprocess all the images in the dataset.
- Pass the images through the trained model.
- Extract the feature vector for each of the images and store it in the csv file.

smallerVGGNetForTransferLearning.py (For visualizing cnn layers)

- The image is provided as input.
- Load the trained model.
- Pass the image through the trained model.
- Activation maps for this particular image is written to the disk.