Capstone Project – Dog breed classifier

Saiful Haq | 22nd May 2020

Definition

1. Project overview:

Domain background:

Image classification refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human figure or not.

Today, it has classification has many applications like autonomous cars use it in object detection, Hospitals use it in medical diagnosis, security systems use it in burglar detection etc.

In this project we are solving a **Multiclass Image classification problem** using Deep Learning to identify the dog breed that resembles or is related to an image.

Datasets and inputs:

A part of ImageNet Database Provided by Udacity. It contains:

i. 8351 Dog Images: Used for training, validation and testing of the CNN.
    1. No. of training images = 6680

2. No. of testing images = 836
3. No. of validation images = 835

ii. 13233 Human images: Used for testing

2. Problem statement:

Problem statement:

Supervised ML problem where labelled dog images are used to train a convolution neural network that can classify the input image to the dog breed it belongs or closely relates to.

Goal:

Goal is to detect whether an image contains a human or a dog.

1. If dog**,** classify images of dogs to the breed they belong to.
2. If human, classify of images of humans to the dog breeds that they resemble the most.

Strategy to solve the problem:

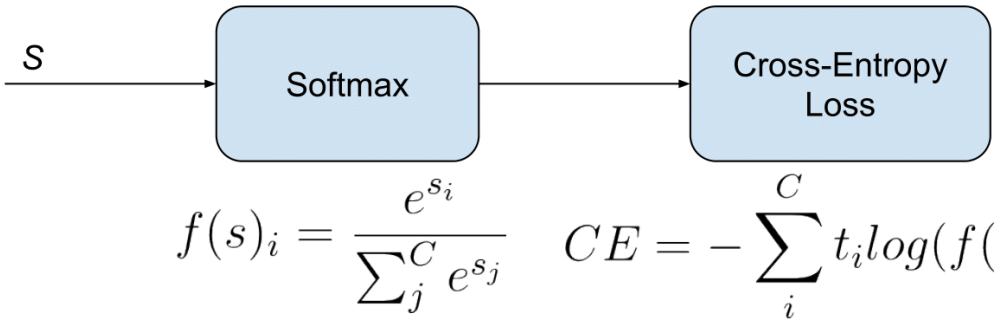
1. Find whether the image contains a human or dog:

2. Train a model using the dog images present in the dataset. This training will enable the model to classify a human image to the dog breed it closely resembles to or classify a dog image to the dog breed it belongs to, when the human/dog image is given as an input to the model.

3. Pass the image identified in step 1 to the model trained in step 2. The model will take this image as an input and will give the Dog breed that it belongs to or closely resembles as an output.

3. Metrics:

The Loss function that is used to **train** the model is the **Categorical Cross Entropy**. Cross-entropy calculates a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem. Cross-entropy loss increases as the predicted probability diverges from the actual label. **A perfect model would have a cross entropy of 0.** Categorical Cross Entropy Combines the Soft max function and the Cross-Entropy loss.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

After training the model, the **accuracy** of the model is calculated as follows :

$$\frac{\text{True positives+True Negatives}}{\text{True positives+True Negatives+False positives+False negatives}}$$

Analysis

1. Data exploration

Dog image dataset

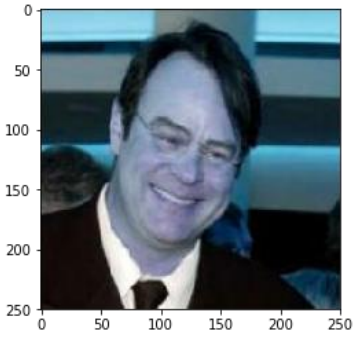
As Mentioned in datasets and inputs of **project overview section**, there are 8351 Dog Images. 6680 are used for training the model, 835 for validation and 836 for testing.

Class Imbalance: The number of labels for each breed varies. Label count of some of the dog breeds is listed below:

Dog Breed	Label Count
Alaskan malamute	77
Bullmastiff	69
Beagle	59
Kuvasz	49

Human image dataset

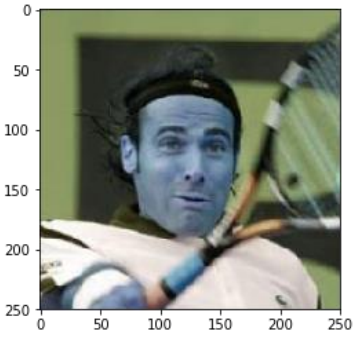
There are 13233 images in this dataset.



2. Data Visualization

Human image dataset

1. All images have the same size (250x250).
2. They have different backgrounds
3. The human faces are at different angles



Dog image dataset

1. Images are of different size.
2. They have different backgrounds
3. The dog faces are at different angles

3. Algorithms and techniques

Used OpenCV's implementation of **"Haar feature based cascade classifiers"** to detect human face in the images.

Used pretrained **VGG-16 Model** to detect Dog in the images.

Convolution neural network model is used to classify the image into the dog breed it closely resembles or belongs to.

1. A CNN model is designed from scratch.
2. A CNN model is designed using transfer learning based on RESNET50 architecture. **RESNET50** is selected using the comparison chart in **Fig. 2**, with **%top-1-error on the x-axis**, **Inference Time GPU on the y-axis** and the **model size**.

A Convolution neural network is made up of convolution, pooling, dropout and fully connected layers.

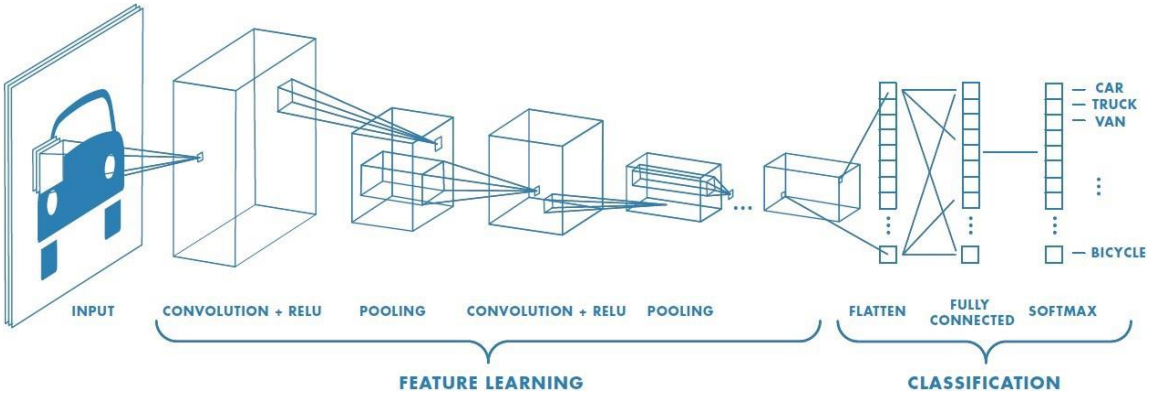


Fig 1. Convolution neural network [8]

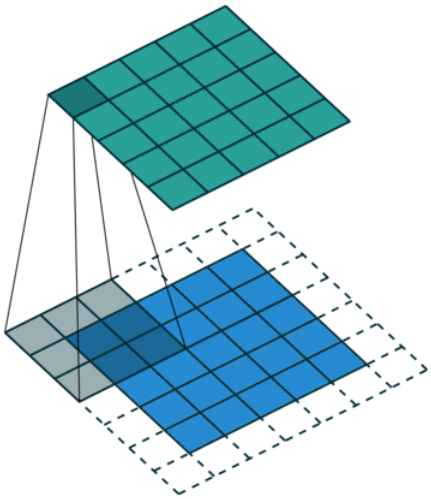


Fig 2. Input image (Blue), output(green), kernel (3x3 Box) [8]

The convolution layer consists of kernels/filters of finite size. As shown in the Fig. 2. Input image of size nxn when given to a kernel of size fxf the output image will be of the size (n-f+1) x(n-f+1).

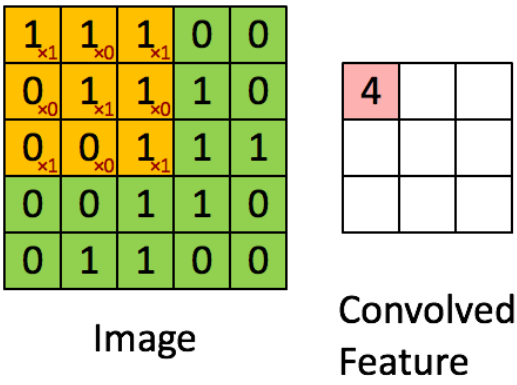


Fig 3. 5x5 Input image (green),3x3 kernel (elements in red), dot product of the kernel with subpart of the image(yellow), one of the features obtained after dot product in the 3x3 output (pink) [8]

CNN are good at the image problems as they also they consider the spatial information and can be used with image with large dimensions. For example, a simple neural network won't be able to distinguish between the image of human with his nose in place of his ears as it does not consider the spatial information. A CNN will be able to distinguish this image from a normal image as it also considers the spatial information.

Hyper parameters in a CNN:

1. Stride length: number of pixels shifts over the input matrix
2. Padding: number of pixels added to the kernel during convolution.
3. Dimension of the kernel

3. Benchmark

There are two models used in this project. One is created from scratch and the other is created using transfer learning. On both the models a benchmark

accuracy is set and our aim to is to achieve an accuracy greater than the benchmark accuracy.

1. CNN model designed from scratch has a benchmark accuracy of 10%.
2. CNN model created using transfer learning has a benchmark accuracy of 60%.

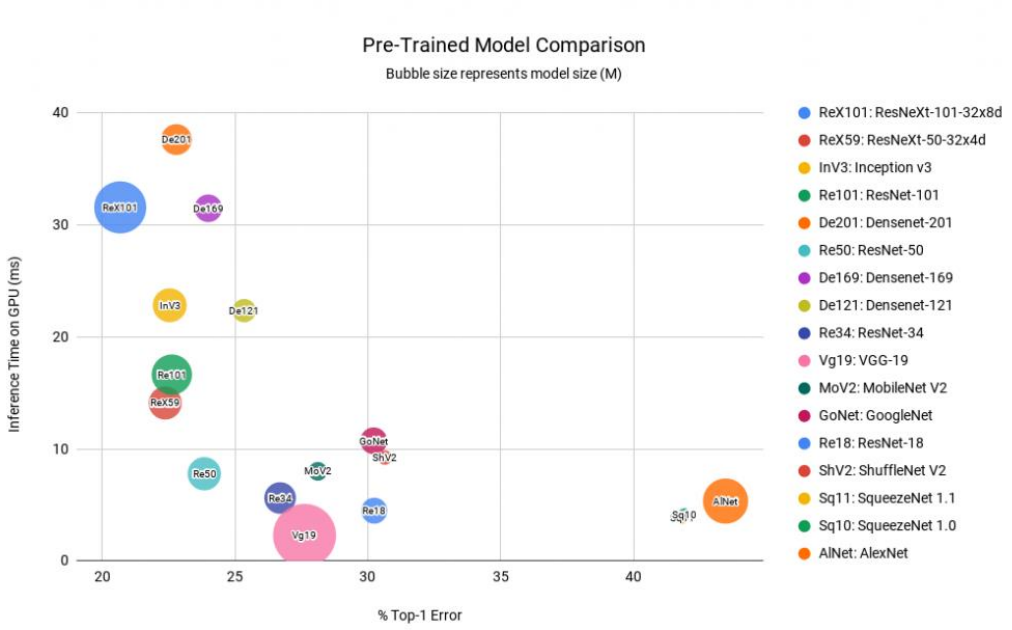


Fig 4. Comparison chart [7]

Methodology

1. Data preprocessing

Training data:

1. Images are resized to 224x224.
2. Augmentation (Horizontal Flip, Rotation and Change in the brightness) is done to all the images
3. Normalization is also done to all the images.

Validation and Testing data:

1. Images are resized to 224x224.
2. Normalization is also done to all the images.

2. Implementation

CNN model from scratch

Architecture tested

1. Convolution layer -> pooling layer -> Convolution layer -> pooling layer -> Fully Connected layer -> dropout layer -> Fully connected layer
2. Convolution layer -> pooling layer -> Convolution layer -> pooling layer -> Convolution layer -> pooling layer -> Fully Connected layer -> dropout layer -> Fully connected layer

2nd Architecture gives the best accuracy and was selected.

Number of Input-Output Channels

Convolution Layer

S I. No	**1st Convolution Layer (input channel x output channel)**	**2nd Convolution Layer (input channel x output channel)**	**3rd Convolution Layer (input channel x output channel)**
1	3 x24	3x36	3x48
2	3x24	3x48	3x96
3	3x36	3x72	3x108

The number of input x output channels given in the 3rd serial gives the best results and were selected.

Pooling Layer

The number of output channels are kept same as the number of input channels.

Fully connected layer

1st Fully connected layer: 84,672 input channel; 512 output channels

2nd Fully Connected layer: 512 input channels; 133 output channels

Other specifications

Convolution Layer

Strides = 1, Kernel size = 3x3, padding = 0

Pooling layer

Strides = 2, Kernel size = 2x2, padding = 0

No. of epochs used for training the CNN = 20

Activation Function: ReLu Function

A **Dropout Layer 0.25** is added to prevent overfitting.

CNN model using Transfer Learning

The last fully-connected layer of RESNET50 is removed and one custom fully-connected is added at the last which is intended to output a 133 sized vector.

3. Refinement

The CNN created from scratch gives an accuracy of 16%. Although it meets the benchmarking accuracy, the results are further improved using transfer learning. RESNET50 CNN architecture gave an accuracy of 82% with 20 epochs.

Results

1. Model evaluation and validation

Human face detector

It detects 98 human faces out of 100 human images,

True Positive = 98, False Positive = 0

True Negative = 0, False Negative = 2

Accuracy is 98% on human data

It detects 17 human faces out of 100 dog images.

True Positive = 0, False positive = 17

False Negative = 0, True Negative = 83

Accuracy is 83% on dog data.

Dog face detector

It detects 100 dog faces out of 100 dog images.

True Positive = 100, False Positive = 0

True Negative = 0, False Negative = 0

Accuracy is 100% on dog data

It detects 3 dog faces out of 100 human faces.

True Positive = 0, False Positive = 3

True Negative = 97, False Negative = 0

Accuracy is 97% on dog data

CNN Model from scratch

It gives an **accuracy of 16% on test data**.

CNN Model using Transfer Learning

It gives an **accuracy of 82% on test data**.

2. Justification

The CNN models created passed the benchmarking accuracy. The model created from transfer learning performed quite well. It gave an accuracy of 82% on the test data.

References:

[1] GitHub Repository: *https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification*

[2] Torch vision models: *https://pytorch.org/docs/stable/torchvision/models.html*

[3] Pytorch Transformations: *https://pytorch.org/docs/stable/torchvision/transforms.html*

[4] CNN: *https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/*

[5] CNN architectures: *https://www.jeremyjordan.me/convnet-architectures/*

[6] CNN using pytorch: *https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/*

[7] *https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/*

[8] *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*