Michael Xavier Ledesma

A Convolution Neural Network identifying a Dog's Breed

Capstone Proposal

Udacity Machine Engineering Nanodegree Program

May 2021

Domain Background

As we are moving into the 21st century, Data Scientists has turn towards Artificial intelligence(AI) that assists machines to mimic human behavior. Machines needs human data to be given a task to restructure that data. The machine needs to learn on emulating the data by combining a model with that data. Therefore, we can say that Machine Learning is a subset of AI techniques.

Branching off from Machine Learning, we can discuss one technique Deep Learning which uses Neural Networks. Similarly, the human brain has neurons. Each neuron has inputs that brings in information to learn. This same practice can be done to machines that uses Neural Networks which are given a set of inputs to train an AI to predict outputs such as the output image of a dog breed.

Problem Statement

How can we assist a machine to classify a dog's breed?

This project will use PyTorch Convolutional Neural Network (CNN) which is a Deep Learning Algorithm that can take an input image such as a dog. The inputs will have its weights and bias that can identify an estimate of the dog's breed. If supplied an image of a human, the code will identify the resembling dog breed. if the detection of the input image is neither a dog or a human, the output will indicate an error.

Datasets and Inputs

From Udacity, two large datasets compressed in zip forms are provided. Once the dataset is unzipped, the datasets consist of 8351 of dog image contents and 13233 of human image contents, respectively. The image contents are needed in providing the training and test sets. The goal is to create an image classifier capable of determining a dog's breed from a dog’s image or a resembling a dog breed from a human image.

Solution Statement

Once the datasets have been imported, method functions on human face detector and dog image detector are implemented. To classify the dog’s breed, the CNN algorithm is coded from scratch. Three separate data loaders load the data and method functions of training, validation and testing are used to model the data. Applying the CNN algorithm with transfer learning improves the generalization of the dog’s breed.   The machine keeps learning by taking features from previous tasks and transferring those features to new input images to classify the dog’s breed. The image classification algorithm can be applied to a mobile or web application.

Benchmark Model

My benchmark model will be a simple CNN model coded from scratch using TensorFlow and Keras. The results will be compared to the CNN Main Architecture with PyTorch via with transfer learning.

Evaluation Metrics

To evaluate the training and test set model, the accuracy score will be implemented.

The accuracy score is by dividing the number of correct predictions by the number of total predictions.

Accuracy score = correct predictions / total predictions.

Project Design

Datasets of humans and dogs was provided by Udacity. A recommendation of facial detection is from OpenCV's, the implementation of Haar feature-based cascade classifiers. The ption code written is the histogram of oriented gradients technology (HOG) on object detection. I believe it had a better accuracy than the OpenCV’s version.

To assist on dog detection, several software gurus from the Oxford University designed a pre-trained VGG16 Model with trained weights from ImageNet to classify dog images. For better performance, Cuda enabled GPU was available.

The VGG16 Model architecture will be used for Transfer Learning which will be describe in more detail.

Project Design

The Preprocessing Data steps are with the pre-trained VGG-16 model:

* Start with a dog image
* Convert the dog image to a PIL Image (Pillow Image) for VGG-16 Modeling
* Convert to "RGB" (Red, Green, Blue)
* Reshape the size of the image, if not it will slow down the pre-process
* Transform the image by tensor flow, normalization, and crop centering
* Transform the image to batch dimension and unsqueeze the NumPy array to get ready to classify the prediction through VGG16 modeling.
* Check if the pre-trained model predicts an index between 151 and 268 (inclusive). A Return value of True or False is given.

Project Design

* From the transform, the size of the original image is (224,224) pixels.

* I selected a stack of 3 convolution layers (RGB), a 3x3 filter convolutional Kernel with a paddling of 1 that gives the filter one more space to move in either direction, to find patterns in an image.

2nd and 3rd convolutional layer are used to discover patterns within a pattern.

* Next, is the Max pooling layer with a kernel size of (2x2) so the dimensionality of the input arrays that make up the image resolution has been reduced to 4x4. The filter size and stride will be (2,2). The original depth is 64.

* Next is to build the class for the Neural Network. The inputs needs to go to two input hidden layer. The 2nd nn.linear has the output of 30.

* To prevent overfitting, a dropout rate of 0.5 of is used.

Project Design

* For the forward behavior of the neural network, a ReLu activation, which as an output of 0 or 1, is applied to the output of these filters to standarized their output values. This is done on each of the three convolution layer.

* Two pool layers of 2x2 are used from self.pool(x)

* The image needs to be flatten by the view method for 1 dimension.

* Drop outs and relu fuction are used before the output.

* Train and validate the model with an EPOCH = 100

* Analyze results by accuracy

* Face recognition using Transfer Learning with VGG16 model.

    * Collect the dataset
    * Trained the model using the VGG16 architecture
    * Test and run the model
    * Check for accuracy