Capstone Project

Randheer Ramesh K

Machine Learning Engineer Nanodegree

May 14th, 2020

Definition

Project OverView

Welcome to the Convolutional Neural Networks (CNN) project! In this project, I will build a pipeline to process real-world, user-supplied images. Given an image of a dog, my algorithm will identify an estimate of the canine's breed. If supplied with an image of a human, the code will identify the resembling dog breed.

Along with exploring state-of-the-art CNN models for classification, we have to make important design decisions about the performance of the model. Our goal is that by completing this lab, you understand the challenges involved in piecing together a series of models designed to perform various tasks in a data processing pipeline.

Problem Statement

The goal is to create a dog breed classification model using CNN. The task involved are the following:

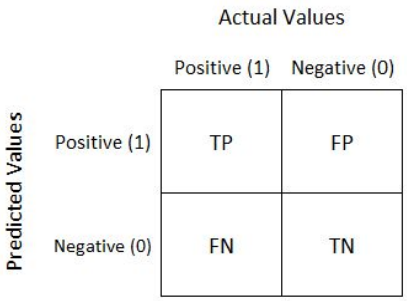
1. Import Datasets
2. Detect Humans
3. Detect Dogs
4. Create a CNN to Classify Dog Breeds (from Scratch)
5. Create a CNN to Classify Dog Breeds (using Transfer Learning)
6. Write your Algorithm
7. Test Your Algorithm

The final model is expected to detect the dog breed from the input image. If the image contains a human face, then output the dog breed that is most similar to the face.

Metrics

The evaluation metrics that can be used to evaluate the performance of the machine learning models are:

- Accuracy: The ratio of correct predictions to the total size of the data (i.e. (TP+TN)/Data Size)
- Recall: The ratio of true positives to the true positive and false negative (i.e. TP/(TP+FN))
- Precision: The ratio of true positives to the true positive and false positive (i.e. TP/(TP+FP))



$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

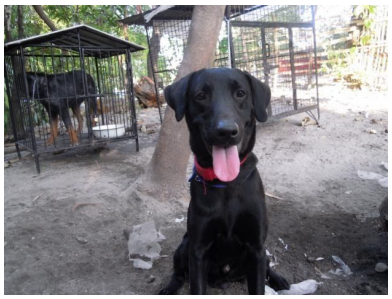
$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

In our case, we will be using the accuracy as the metric of measurement to evaluate the performance of the model.
There are cases where the metrics such as accuracy might be misleading in the case of imbalance data. One such example is the fraudulent data in which we dealt in the earlier studies of this course. We have seen the percentage of fraud cases being drastically low compared to non fraudulent cases in the data. However, in this data of dog breeds, as we observe in the visualization part, the data in each class is beyond certain threshold and looks ideally balanced to carry the classification work. Therefore, the accuracy can be used as the metric of evaluation in our work.

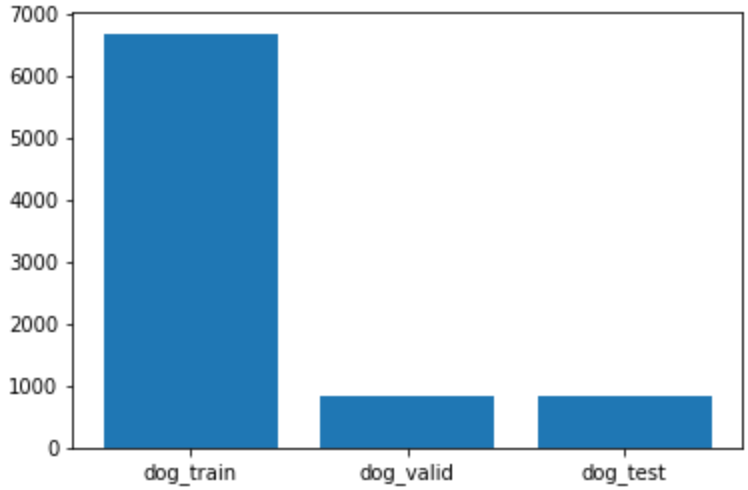
Analysis

Data Exploration

Here, in the Dog Breed Classification, the dataset contains the images of Dogs and Humans. There are a total of 133 breeds, 8351 images for dogs. Using these images as data, it has to be processed according to our needs and a model has to be designed to train our machine. On making the analysis on the data, we see that the resolution of the images are not the same for all images of dogs in their respective breeds. The images have a varied resolution and they need to be resampled based on the requirement of our model. Here is one example of the images discussed in terms of resolution:

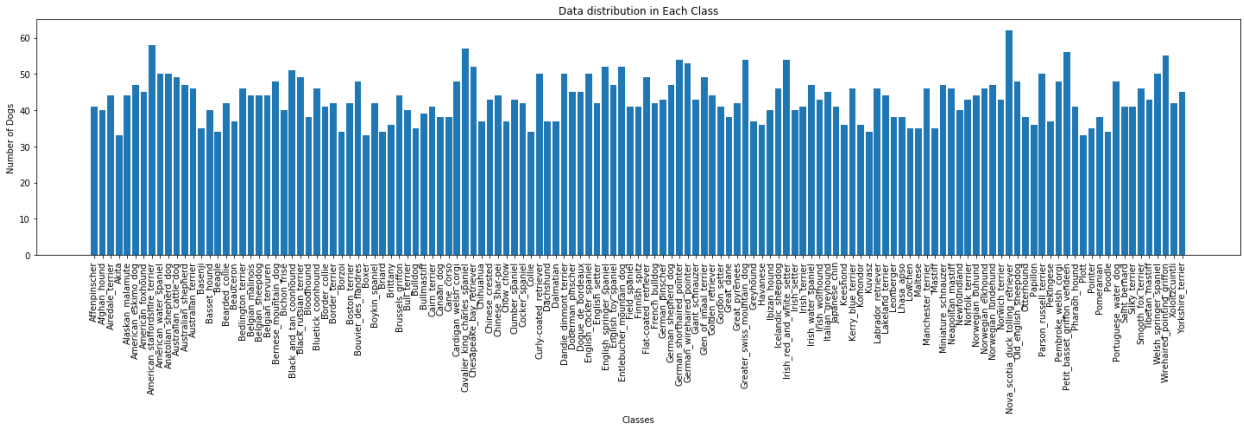


In our case, we observe that the split of train and test data is 90%-10%, i.e. 90% for training and 10% for testing purposes. In the training data, we have reserved another 10% for validation. The resultant split of data can be observed in the below graph. From the below plot we can observe that a total of 6680 images will be used to train our machine, to further fine-tune the parameters we use another 835 images for validating it. And, lastly, we will be using 836 images to test our model's performance for the evaluation of metric i.e. Accuracy.



Exploratory Visualization

The study on the distribution of data gives us the information on balance or imbalance in the data. If there is an imbalance in the data beyond a certain threshold, we must see that the data is balanced by adding relevant images. If the balance in the data is comparatively near to the threshold, it is good to carry forward with the operation. Let us see how it works with our data in the below figure. The plot shows a clear description on breed class with the number of dogs.



Algorithms and Techniques

The classifier is a Convolutional Neural Network, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; fortunately, the COCO and COCO-Text datasets are big enough. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives using a threshold. (The tradeoff is that this increases the number of false negatives.)

The following parameters can be tuned to optimize the classifier:

❖ Classification threshold (see above)

❖ Training parameters

- Training length (number of epochs)
- Batch size (how many images to look at once during a single training step)
- Solver type (what algorithm to use for learning)
- Learning rate (how fast to learn; this can be dynamic)
- Weight decay (prevents the model being dominated by a few “neurons”)
- Momentum (takes the previous learning step into account when calculating the next one)

❖ Neural network architecture

- Number of layers
- Layer types (convolutional, fully-connected, or pooling)

- Layer parameters (see links above)

❖ Preprocessing parameters (see the Data Preprocessing section)

During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing. The training is done using the Mini-batch gradient descent algorithm (with momentum).

In contrast to this, inference (that is, inference in the Android app) is done using only the CPU, because TensorFlow doesn't support smartphone GPUs.

Benchmark

To tackle such data, we use a benchmark model to build a basic pipeline and a well-versioned model to improvise our classification rate. Such a methodology is carried to tune our model for better prediction of results. The benchmark model helps us to make a comparison and reduce the overfitting or underfitting condition and tune the model for a better outcome. Logistic Regression, KNN are such examples of the benchmark. We can also use the predefined image classifiers such as ImageNet, ResNet, VGG16, etc. to classify our images and later optimize our pipeline for better evaluation of metrics.

In the works of Dog Identification in Kaggle, we see that "Mohamed Sheik Ibrahim" used the VGG19, a predefined base model and carried various processing techniques such as data augmentation to improvise the results obtained from the predefined model. He also used logistic regression to classify the images of dogs and achieved an accuracy of 68%.

Considering another work performed on the same data, using the inception v3, the pre-trained model for image classification, Carey B achieved an accuracy of 83%, which is considered to be a good classification rate based on the performance of the model.

Using the above understandings, We will be using VGG16 for our data for classifying the breeds of the Dogs, Later we build a Convolutional Neural Network and tune the parameters, Using transfer learning through these models, make a comparative study and analyze the performance of the model.

Methodology

Data Preprocessing

The image we given as input to the model process the image before passing it to the network. In this step, the image is normalized, resize, crop the image to speed up the training. Then the image is converted into tensors.

In this step, there are some parameters we can choose, like the parameter for resizing, center crop, and normalize methods.

Implementation

In this step, the model is designed. We will be using 5 convolutional layers for this project and 3 fully connected layers. A Dropout layer is added at the end so that it will reduce the complexity of the network.
5 convolutional layers are created with max-pooling layers in between them to learn a hierarchy of high-level features. Max pooling layer is added to reduce the dimensionality. Dropout was used along with the flattening layer before using the fully connected layer to reduce overfitting and ensure that the network generalizes well. The number of nodes in the last fully connected layer was set up as 133. Relu activation function was used for most of the layers.

The below given are the steps followed in the implementation part.

- Load both the training and validation images into memory, preprocessing them as described in the previous section.
- Define network architecture and training parameters.
- Define the loss function, accuracy.
- Train the network, logging the validation/training loss and validation accuracy.
- If the accuracy is not high enough, retry these steps, else save and freeze the trained network.

```
trainLoader = torch.utils.data.DataLoader(train_dataset,
                                          batch_size=128,
                                          shuffle=True,
                                          num_workers=0)

validLoader = torch.utils.data.DataLoader(valid_dataset,
                                          batch_size=128,
                                          shuffle=True,
                                          num_workers=0)

testLoader = torch.utils.data.DataLoader(test_dataset,
                                          batch_size=64,
                                          shuffle=False,
                                          num_workers=0)
```

This is how the images are loaded.

```
class Net(nn.Module):
      ### TODO: choose an architecture, and complete the class
      def __init__(self):
      super(Net, self).__init__()
      ## Define layers of a CNN
      self.conv1 = nn.Conv2d(3, 64, kernel_size=3, padding=1)
      self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
      self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
      self.conv4 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
      self.conv5 = nn.Conv2d(512, 512, kernel_size=3, padding=1)
      self.pool = nn.MaxPool2d(2, 2)
      self.fc1 = nn.Linear(25088, 512)
      self.fc2 = nn.Linear(512, 512)
      self.fc3 = nn.Linear(512, 133)
      self.dropout = nn.Dropout(0.5)

      def forward(self, x):
      ## Define forward behavior
      x = self.pool(F.relu(self.conv1(x)))
      x = self.pool(F.relu(self.conv2(x)))
      x = self.pool(F.relu(self.conv3(x)))
      x = self.pool(F.relu(self.conv4(x)))
      x = self.pool(F.relu(self.conv5(x)))
      x = x.view(-1, 25088)
      x = F.relu(self.fc1(x))
      x = self.dropout(x)
      x = F.relu(self.fc2(x))
      x = self.dropout(x)
      x = self.fc3(x)
      return x
```

This is the model architecture.

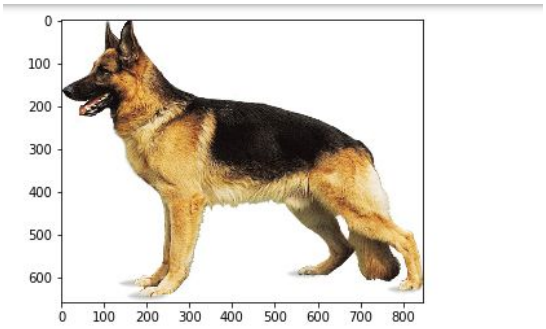
Model Evaluation and Validation

During development, a validation set was used to evaluate the model. The final architecture and hyperparameters were chosen because they performed the best among the tried combinations. The following list:

- The details about the network can be obtained from the above figure.

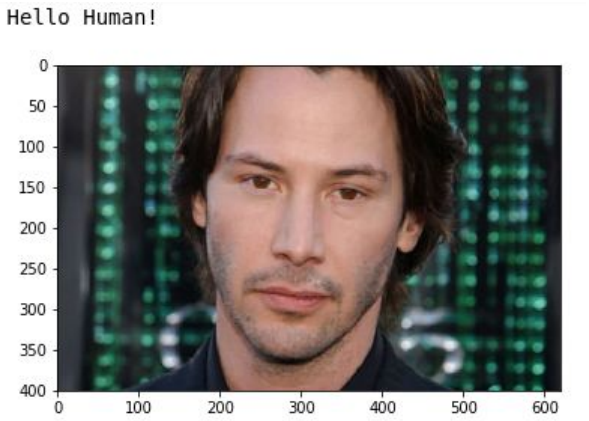
The validation set is used for hyperparameter tuning and the test set is used to test the performance of the model. After completing the training, and testing of the model we got an accuracy of 77%

At last, I used 6 images to test the model and found that the model correctly classified the images

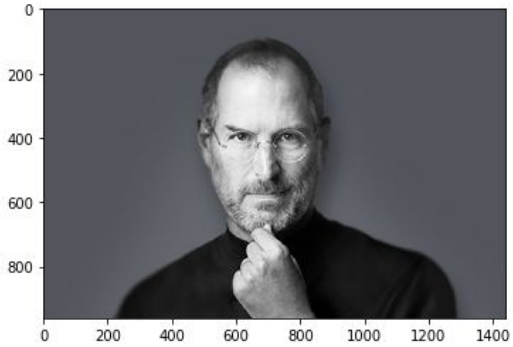


This is a picture of a ... German shepherd dog

Hello Human!

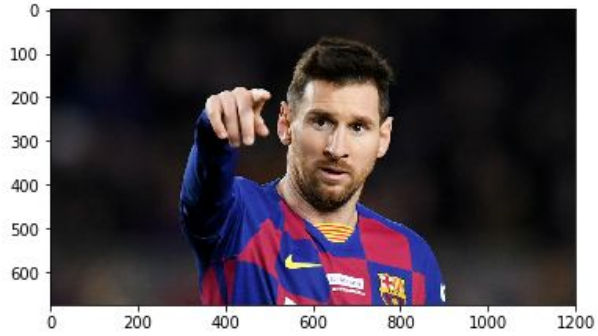


Hello Human!

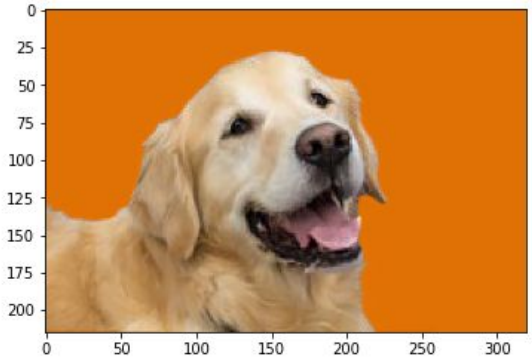


You look like a ... Italian greyhound

Hello Human!



You look like a ... Dachshund



This is a picture of a ... Golden retriever

This is a picture of a ... Flat-coated retriever

Results

In this section, we observe the performance evaluation and results of the final implemented model. Initially, we observe the study made on the detection algorithm and compared their results to see their behavior. On implementing the haar cascade and local binary pattern, we observe there were 98 faces detected for haar cascade and 95 faces in local binary pattern of human images. The results show the haarcascade has a better performance compared to the

local binary pattern.

In the implementation phase of the pre-trained model VGG 16, we test the prediction of the model and we also observe that the results show the right prediction value of the dog. On implementing the custom Convolutional Neural Network model, we construct out the relative stack of layers required for our purpose and tune the necessary hyperparameters of the model.

On testing the performance of the custom model through the metric evaluation i.e. accuracy in our case, we observe that the results give us a result of 18% in terms of accuracy. The project had the condition to achieve the results of more than 10% accuracy.

After the implementation and training of the transfer learning model for a total of 7 epochs, we need to evaluate the performance of the given test data of dog images so that the model should predict the breed of the dog with utmost accuracy. Therefore, getting the model ready for evaluating the performance on the test data, we used the test function that was previously defined for the custom model and evaluated the performance of the transfer learning model. We see that the model achieved an accuracy of 77%, and we see that there is a considerable rise in the performance of the model after combining the predefined model with our custom model.

Reflection

The process used for this project can be summarized using the following steps:

- An initial problem and relevant, public datasets were found
- The data was downloaded and preprocessed
- The classifier was trained using the data (multiple times, until a good set of parameters, were found)

Out of these steps, the 3rd step bit hard compared to other steps since it contains network construction and training. But after many tries, I found a good pair of parameters to train the model and got a good model at the last.

Improvement

Even though I trained the model to predict good results, It doesn't mean that we can't further improve the model. Below are some steps that could result in a better model.

- Increase the number convolutional layer
- Adding more dropout layer
- Convolutional Network Parameters
- Altering the Multi-Layer Feed-Forward NN at the end of the network

Conclusion

The model for detecting Dog breed from the input image if constructed with an accuracy of 77%