

Capstone Project

# **Machine Learning Engineer Nanodegree**

---

R Aravind

13th May, 2020

## Definition

### Project Overview

This is a Udacity specific project and came with pre-stated goals and a jupyter notebook to complete. The project is to use Convolutional Neural Networks to Identify Dog Breeds

Dog breed classification is really challenging due to the minimal inter-class variations. A Convolutional Neural Network (CNN) model can be used for this task. Due to the existence of a huge variety of species and organisms, research on technology to classify them is crucial. Hsu, David from Stanford University published a paper about using convolutional neural networks (CNN) to classify dog breeds. The final project will be able to identify the breed from 133 given breeds given an image of a dog or if given an image of a human it will identify the breed which is similar to the human.

### Problem Statement

The aim of the project is to build a pipeline to process real-world, user-supplied images. A CNN will be used to predict the breed of a dog from the input image. Given an image of a dog, the model will identify an estimate of the canine's breed. If supplied an image of a human, it will identify the resembling dog breed. The workflow for this project can be described as follows:

1. Import Datasets
2. Detect Humans using OpenCV's Haar feature-based cascade classifiers.
3. Detect Dogs using a pre-trained VGG-16 model.
4. Create a CNN to Classify Dog Breeds (from Scratch)
5. Create a CNN to Classify Dog Breeds (using Transfer Learning)
6. Write an Algorithm to return the breed if a dog is detected or the resembling breed if a human is detected
7. Test the Algorithm

### Metrics

Accuracy will be used to evaluate both the solution model and benchmark model. Accuracy will be calculated with the help of the confusion matrix.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

The accuracy requirements of the project are as follows:

1. The CNN built from scratch was required to have an accuracy greater than 10%.
2. The CNN built using transfer learning was required to have an accuracy of 60% at least.

## Analysis

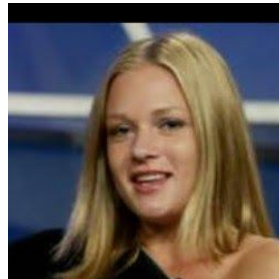
### Data Exploration

The Datasets were provided along with the jupyter notebook mentioned earlier. It was properly labelled and already split into train, validation and test sets. The dogImages dataset consists of 8351 images of 133 breeds, 8 images from each breed. The human images were from the Labeled Faces in the Wild (LFW) dataset. It consisted of 13233 images of humans labelled according to their names. The data from both datasets are evenly distributed.

Datasets:

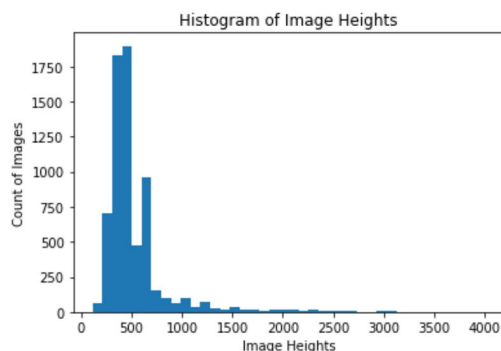
1. DogImages: <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip>
2. LFW: <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip>

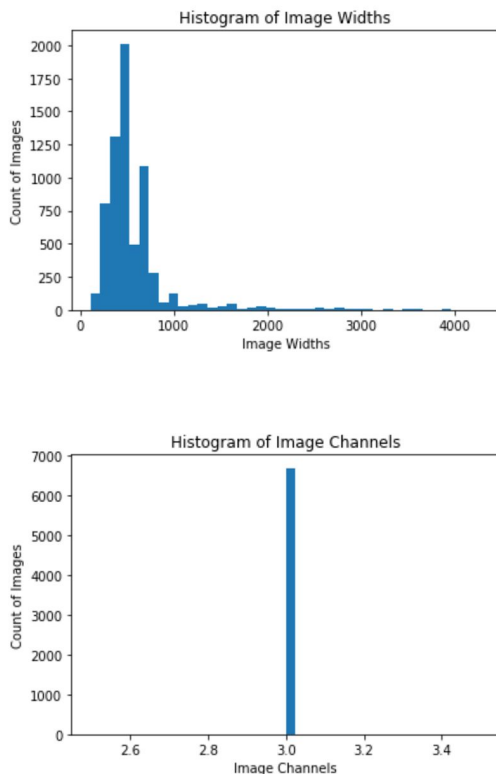
Example images:



### Exploratory Visualization

I visualised the heights, widths and channels of the images in the dataset to get an idea of the variations in their shapes.






From the above visualisations we can infer that all the images have 3 channels, but their heights and widths vary. We can also infer that the majority of the images have their heights and widths less than 1000 pixels and a few outliers in the range 1000-4000 pixels.

## Algorithms and Techniques

CNNs are a kind of deep learning model that can learn to do things like image classification and object recognition. They keep track of spatial information and learn to extract features like the edges of objects in something called a convolutional layer. The convolutional layer is produced by applying a series of many different image filters called convolutional kernels to the input image.

Other than the CNNs, OpenCV's implementation of Haar feature cascade classifier was used to perform face detection of the human images.

On tasks involving computer vision CNNs outperforms other Neural Networks like MLP by a far. This project requires two CNNs ( model from scratch and model from transfer learning ) mainly for classifying the breed and one for detecting dogs.



For detecting dogs a pre-trained VGG-16 model was used, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

The model from scratch was implemented with 5 convolution layers for extracting the features and 3 fully connected layers to act as the classifier.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. For transfer learning, I picked the pre-trained Resnet-50 model and froze its layers and replaced its fully connected layer with one that suits our purpose and trained the model to update the weights of the replaced layer.

## **Benchmark**

The benchmark was pre-stated as the model from scratch must achieve a test accuracy of at least 10% and the model from transfer learning must achieve a test accuracy of at least 60%.

My implementation of the model from scratch had a test loss of 3.644251 and a test accuracy of 12%. My implementation of the model from transfer learning had a significant improvement from the model from scratch and it had a test loss of 0.473873 and test accuracy of 85%.

# Methodology

## Data Preprocessing

The images were resized and cropped to 224x224 pixels to achieve an input tensor of shape 224x224 which suits the pre-trained model and the model from scratch.

Images for the training set had undergone data augmentation to prevent overfitting and to get more generalized training data. Augmentation was done by randomly rotating, flipping, resizing and cropping the data. The validation and testing data was exempted from augmentation, but they were resized and cropped to 224x224 like the training set.

After resizing and cropping, the image data were converted to tensors and they were normalized.

## Implementation

The implementation of this project involved 3 major parts: the CNN from scratch, the CNN from transfer learning and the final app.

The code snippet below shows the architecture of the CNN from scratch.

```
# define the CNN architecture
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)

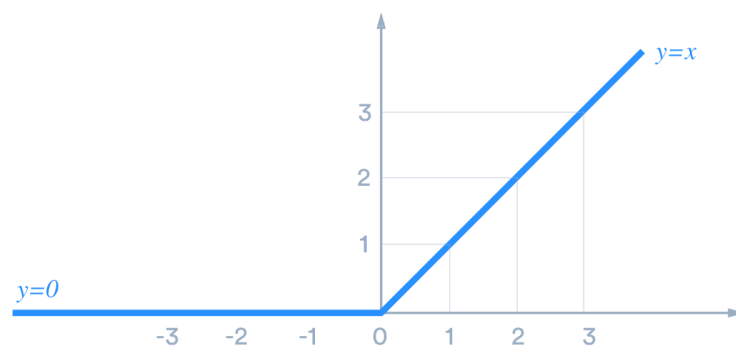
        x = F.relu(self.fc2(x))
        x = self.dropout(x)

        x = self.fc3(x)

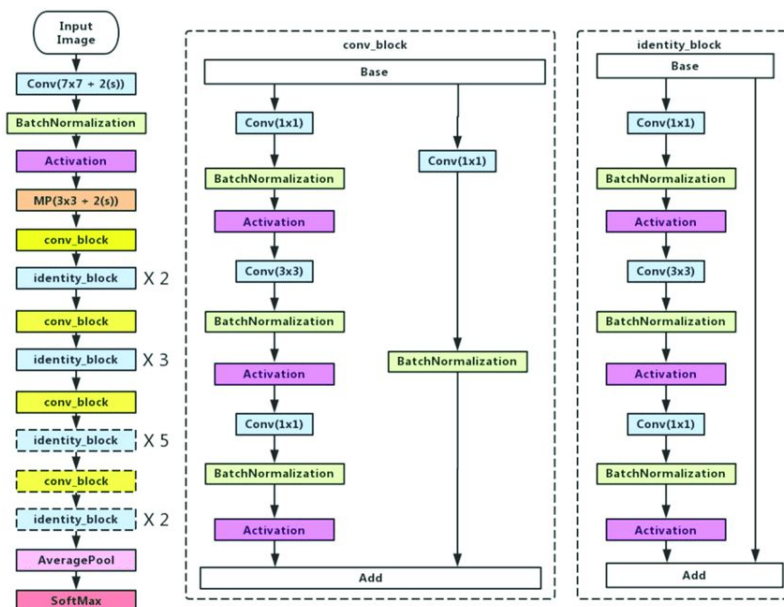
        return x
```

To achieve the benchmark of at least 10% test accuracy, I decided to go with the above model. It consists of 5 convolutional layers, each with a convolutional kernel of size 3x3 and padding of 1. After each convolutional layer was a Max Pooling layer of window size 2x2 and stride of 2. The input was a tensor of shape 3x224x224 which had undergone pre-processing and augmentation ( in the case of training data ). After passing through these 5 convolutional layers, we get an output tensor in the shape of 512 7x7 feature maps. It is then flattened into a vector of length 25088 and fed into the fully connected layers. The fully connected layers use a Rectified Linear Unit (ReLU) activation function and I made use of dropout which randomly freezes nodes to prevent overfitting and improve validation/testing accuracy. The final fully connected layer has an output shape of 133 corresponding to the number of breeds.

The graph below depicts the ReLU activation function.



For the transfer learning model, I picked a Resnet-50 model pre-trained on the ImageNet dataset. The figure below illustrates the architecture of the Resnet-50 model.



Then, I froze all the layers in the pre-trained model and replaced its final fully connected layer with a new layer which output features equal to the number of classes in our task, i.e. number of breeds.

For both the above models, I used Cross entropy loss as criterion and Adam as the optimizer with a learning rate of 0.001. I picked 0.001 after some trial and error, the model was not converging on learning rates such as 0.3 and 0.03.

Finally, an algorithm was needed to bring the face detector, dog detector and CNN ( model from transfer learning ) together. If the face detector ( OpenCV's Haar cascade classifier ) detects a human face the algorithm would return the breed which is identical to the face. Otherwise, if the dog detector ( pre-trained VGG-16 model ) detects a dog then the algorithm would predict the breed of that dog. If both detectors returns false, the algorithm would print out an error message.

The code snippet below illustrates the final app.

```
def run_app(img_path):  
    ## handle cases for a human face, dog, and neither  
  
    plt.imshow(Image.open(img_path))  
    plt.show()  
  
    human_found = face_detector(img_path)  
    dog_found = dog_detector(img_path)  
  
    if human_found:  
        print('Hello! You look like a ' + predict_breed_transfer(img_path))  
    elif dog_found:  
        print('This dog breed is ' + predict_breed_transfer(img_path))  
    elif (not human_found) & (not dog_found):  
        print('No Dog or Human found!!')
```

## Refinement

The performance of both the models were below par initially and did not satisfy the benchmark requirements. Initially the model from scratch was overfitting and overall accuracy was not satisfactory, so I introduced a dropout layer between the fully connected layers. This increased the accuracy considerably, then I tried increasing the dropout probability from 0.25 to 0.5 and the results were good. As for the model's architecture , I had undergone a few iterations with different numbers of fully connected layers and variations in their number of input features and output features.

For the transfer learning model, I tried variations in the new layer replacing the fully connected layer of resnet-50. Changing the optimizer from stochastic gradient descent to adam improved the performance and I decided to go with a learning rate of 0.001 after 0.3 and 0.03 were showing poor performance.



## Results

### Model Evaluation and Validation

It was already recommended in the given content to use the model from transfer learning in the final app. It was an obvious choice as the model from transfer learning outperformed the model from scratch by far. The models were selected during training, during each epoch the validation accuracy of the model was compared to the one with the minimum validation accuracy. And if it was lower, the present model would be saved. Then the models were tested against previously unseen data from a testing dataset and testing accuracy was calculated. This testing accuracy was compared with the benchmark requirements. Both the models passed the benchmark tests. The model from transfer learning performed really well and proved robust enough to solve the task of breed classification.

### Justification

The model from transfer learning outperformed the benchmark by 25% and It seems good enough to be used in the application.

Model from scratch:

- Benchmark requirement: 10% accuracy
- Testing accuracy: 12% (107/836)

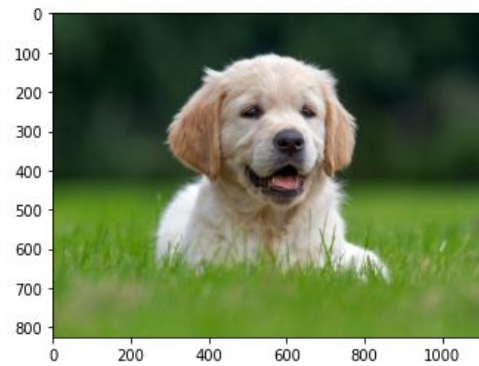
Model from transfer learning:

- Benchmark requirement: 60% accuracy
- Testing accuracy: 85% (717/836)

## Conclusion

### Free-Form Visualization

The output of the final application is shown below:



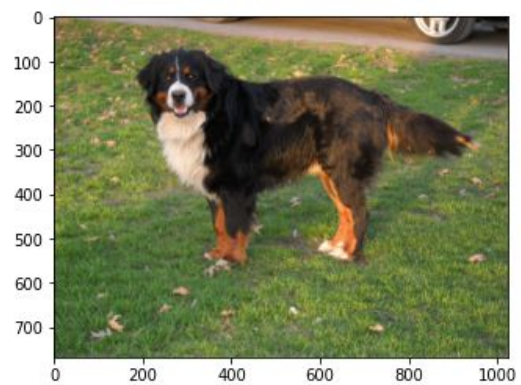
This dog breed is Golden retriever



Hello! You look like a Portuguese water dog



No Dog or Human found!!



This dog breed is Bernese mountain dog

As you can see above, the application correctly classifies dogs and humans. It also identifies the breeds of the dogs in the top-left, bottom-right correctly.



## Reflection

This project was quite a journey of learning from the initial proposal to the final report at hand. This capstone project really illustrates how to solve a real-world machine learning problem and the pipeline in solving a problem. I picked this particular project of dog breed classification ( using CNNs ) as I was really interested in Deep Learning and I saw this as an opportunity to work on a real world problem with a deep learning algorithm ( i.e. convolutional neural networks ) and the notion of it excited me.

I learned that most parts of the pipeline in solving a machine learning problem is expensive in terms of computing resources and time. The model took a lot of time to train even with a GPU and CUDA which highlighted the importance of making smart decisions over trial and error in terms of tuning architectures and parameters. I also learned the importance of documentation in peer-reviewed projects and journals.

## Improvement

I could improve my final model from transfer learning and model from scratch by training it on more data which had undergone augmentation as this would generalize the model. For a model from scratch, I could try different architectures but this would require more computing power especially more powerful GPUs. For the transfer learning model, I could experiment with other pre-trained models and see if they perform better than my Resnet-50 one.

It would be good to implement the final app as a mobile or web application that detects dog breeds in real time. It would be welcomed by the dog-lovers community. Or we could expand the application and implement models which would detect different species of birds.etc. Such an application would be a boon to a number of people from observers to wildlife photographers.

