UDACITY

Capstone Project
Machine Learning Engineer Nanodegree

December 20th, 2020
Mostafa Ashraf Mohamed
mostaphaashraf1996@gmail.com

Dog Breed Classifier Using CNN

Definition

- **Project Overview**

A canine's breed estimator exploring CNN principles for classifications. A model will be training with real-world data then after final training, the model will be supplied with dog images and predict its breed. If a model is supplied with a human image, the model will estimate a closet similarity for a dog breed.

- **Problem Statement**

A canine's breed as we discussed before, its used to classify breeds of dogs, and if it's supplied with a human image its give me closet similarity for a human face to an estimated breed dog, but let's talk more, we start using a cascade classifier provide from OpenCV it's a face detector pre-trained model it's trying to detect a human face and try to find it's accuracy on a chunk from our data humans and dogs it's given me high accuracy in human images, we start using a VGG-16 model in PyTorch it's trained on an ImageNet dataset it contains around 1000 categories we used it's to classify an image it's a dog or not after making some data preprocessing e.g. resize, cropping, normalization, etc... then also find it's accuracy and its give us a high accuracy on dogs images. Finally, we start to developed our CNN model from scratch using PyTorch and train - validate, and test this model. we also apply a concept of transfer learning to make a prediction. Finally, all models mentioned make some preprocessing for data.

- **Metrics**

According to the precision and recall concept when we used in binary class classification, will apply this concept into multiclass classification, a typical multi-class classification problem, we need to categorize each sample into 1 of N different classes, Similar to a binary case, we can define precision and recall for each of the classes.
So, Accuracy = correct / total size
Or = true positive + true negative / dataset size

Analysis

- **Data Exploration and Visualization**

Dataset I've used consists of 13233 human images and 8351 dog images, its supplied to CNN model I have created in Project, also in a transfer learning section I'm used a VGG16 Model and this is trained on ImageNet, it is a very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories.

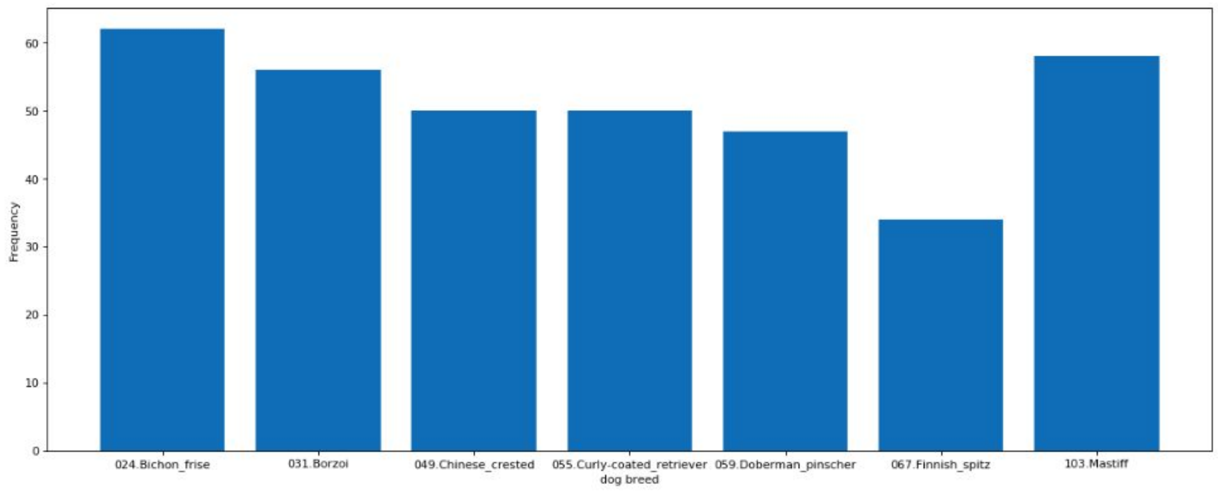


Fig. 1 Image from DataSet -sample from dog breed vs Frequency of each one

● Algorithms and Techniques

The classifier I have used it’s follows the principle of convolutional neural networks 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, its division into three main approaches, Pretranided haar cascade classifier model for face detection, VGG-16 pre-trained model for dog detector or it’s breed, CNN written from scratch, so let’s discuss CNN architecture.

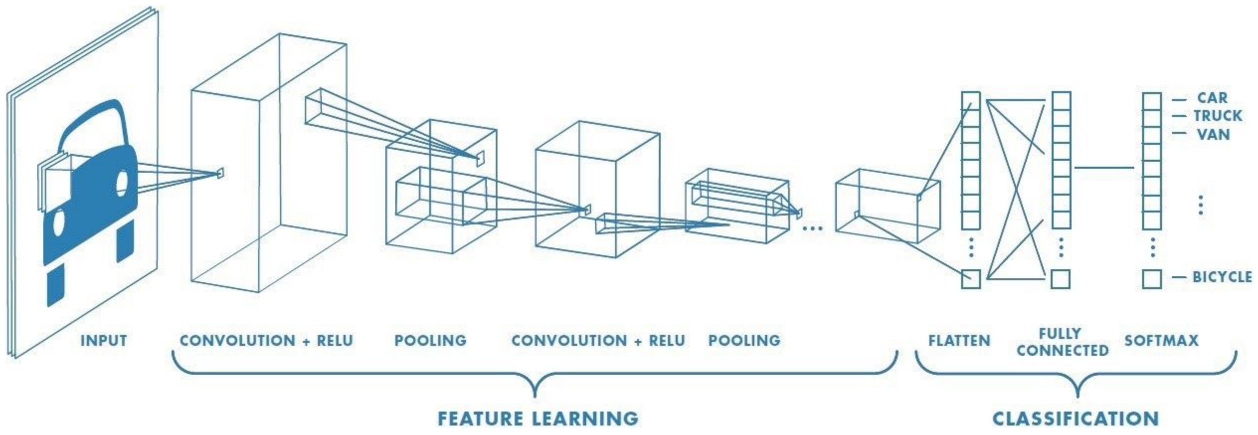


Fig. 2 CNN Architecture

● Benchmark

I’ll provide below some image preprocessing techniques, also In CNN created from scratch added three convolutional and pooling layers then make flatten and add them to a fully connected layer, also in transfer learning, I’ve edited the last layer by freezing the last weighted parameters.

Methodology

- **Data Preprocessing**

The preprocessing done in the “Prepare data” notebook consists of the following steps:
1. The list of images is randomized.
2. The images are divided into a training set and a validation set.
3. Make some transformation into images cropping.
4. Normalized images and convert them into tensors.
5. Pass images to data loader with batch size equal thirty then shuffle it.

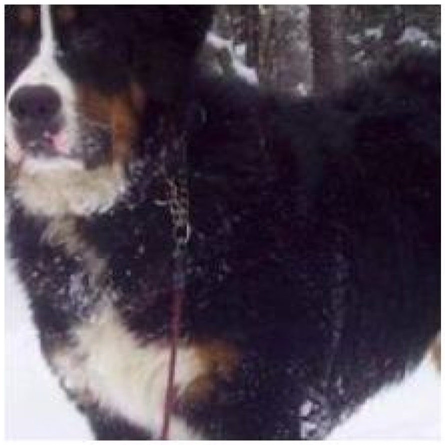


Fig. 3 Image processed sample

- **Implementation**

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and the reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction. It’s consists of a convolutional it has a kernel n * m dim with stride(step of movement), also it has a pooling layer and when a stride is higher than 2 it’s called downsample pooling can be max pooling or average pooling, etc...,

Fig. 4 The illustration on the right shows
My CNN layout I made,
The acronyms can be read as follow:

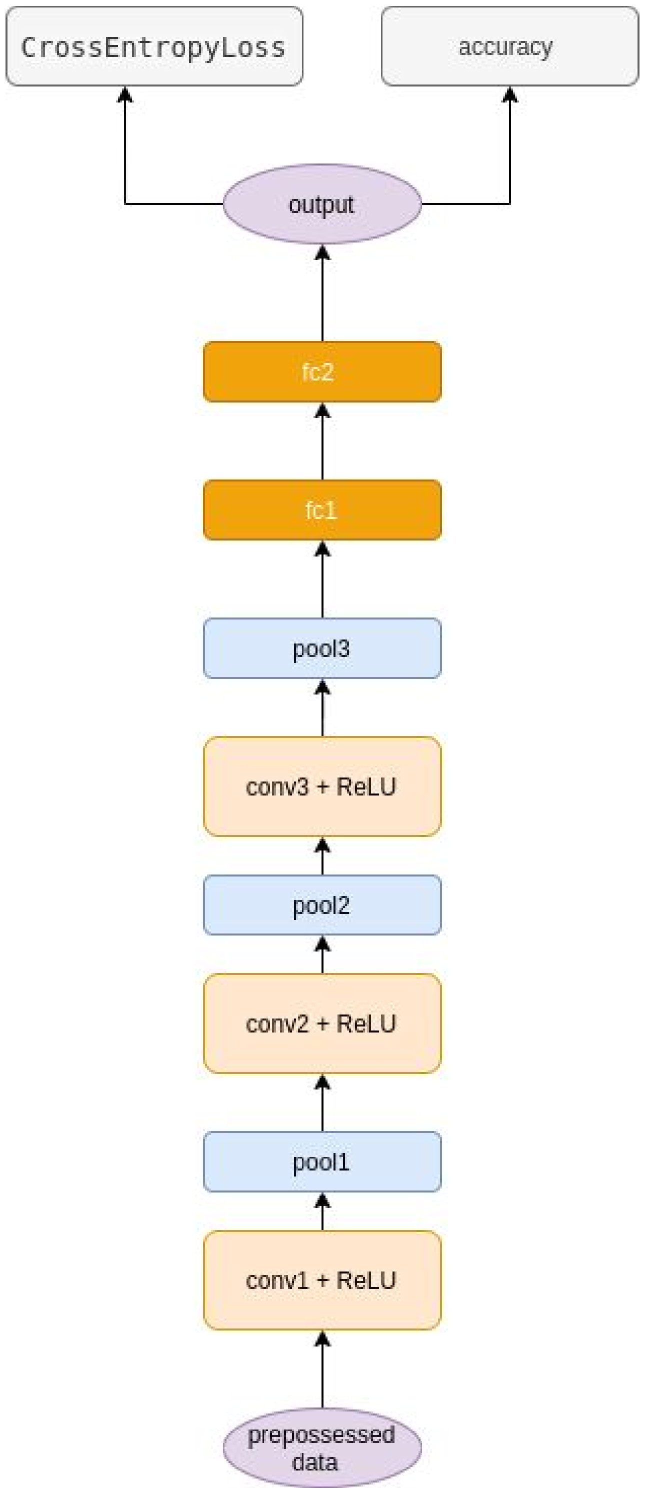
- Conv: Convolutional layer
- Pool: MaxPooling layer
- Fc: fully connected

The following can be seen by looking at the graph:

- The loss function is mainly composed of the cross-entropy loss and its combination of log softmax and negative log-likelihood loss” `NLLLoss”` In a single one class.

After saying layout, will make anatomy for my network:

- Pooling layer as mentioned above I’m used a max-pooling it’s 2*2 with stride 2 so it makes a downsampling by 2.
- I follow a VGG-16 CNN principle and make transformation to images and the final size is [3,224,224]
- The 1st conventional layer in_channel is 3-> RGB, out_channel 32, and make a stride by 2, so after this with pooling data will be [32, 56, 56]
- The 2nd conventional layer in_channel is 32, out_channel 64, and make a stride by 2, so after this with pooling data will be [64, 14, 14]
- The 3rd conventional layer in_channel is 64, out_channel 128, and make a stride by 1 not make downsample, so after this with pooling data will be [128, 7, 7]
- Finally, the final output from feature learning layers [128, 7, 7] will feed into a LinearLayer “dense layer” with ReLU activation function, to predict

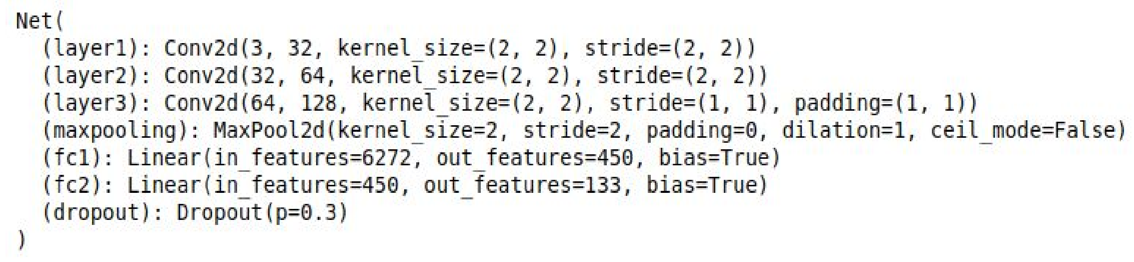


output and use a Cross entropy
loss and backpropagation to
enhance a model.

- **Refinement**

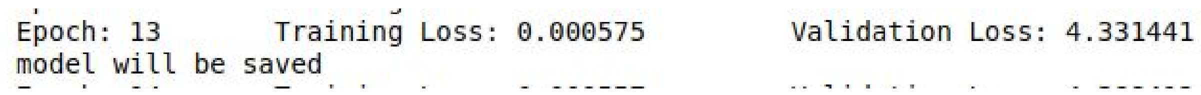
In the CNN model created from scratch, I think it's very complicated to guess the good depth for convolutional and pooling layers I need with stride and kernel size and some other parameters, so I'm starting with the Adhoc principle,

1st trail I get these values



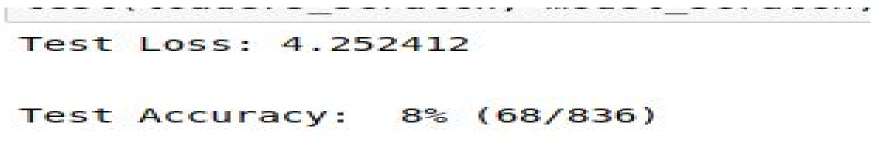
```
Net(
  (layer1): Conv2d(3, 32, kernel_size=(2, 2), stride=(2, 2))
  (layer2): Conv2d(32, 64, kernel_size=(2, 2), stride=(2, 2))
  (layer3): Conv2d(64, 128, kernel_size=(2, 2), stride=(1, 1), padding=(1, 1))
  (maxpooling): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=6272, out_features=450, bias=True)
  (fc2): Linear(in_features=450, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

Model architecture



```
Epoch: 13       Training Loss: 0.000575        Validation Loss: 4.331441
model will be saved
```

Final saved model



```
Test Loss: 4.252412

Test Accuracy:  8% (68/836)
```

Test accuracy!!

Make repeated for guessing values until results mentioned in model and validation section.

Results

- **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.
  For a complete description of the final model and the training process, refer to Figure 4 along with the following list:
  1. The shape of the filters of the convolutional layers is 2*2.
  2. The first convolutional layer learns 32 filters, the second learns 64 filters.
  3. The first and second convolutional layers have a stride of 2, so the resolution of the output matrices is half the resolution of the input matrices.
  4. Like the convolutional layers, the pooling layers halve the resolution 2 too.
  5. The first fully connected layer has 400 outputs

- **Justification**
  I'm using the model and check it on multiple image tests overall it's good but failed in some images because it needs some enhancement, also can use this model as a web app or mobile or real-time prediction. In summary project, it's useful and gives insights about creating a good Deep learning model and passing on some benefits and useful topics.

Conclusion

- **Free-Form Visualization**
  - **Face detector and dog detector examples**

```
Number of faces detected: 1
```

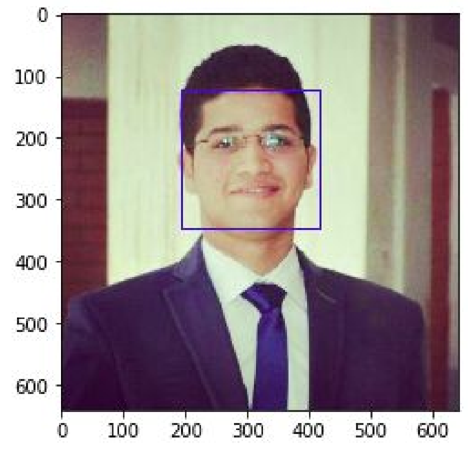


Fig. haar cascade results

```
image below is a dog
```

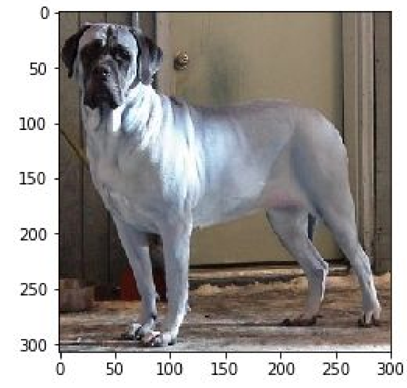
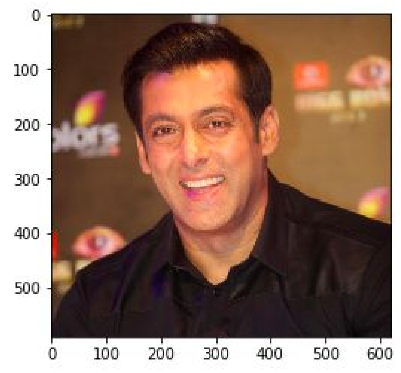


Fig. VGG-16 results

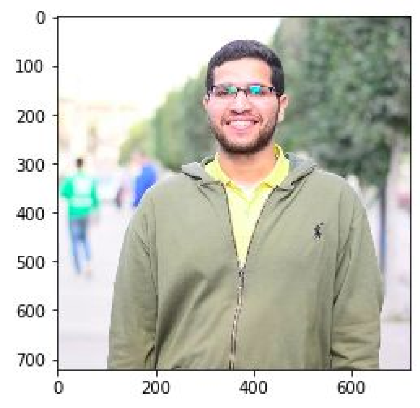
  - **CNN breed classifier results**

```
Hello, human!
```



```
You look like Dachshund
Hello, human!
```

```
Hello, human!
```

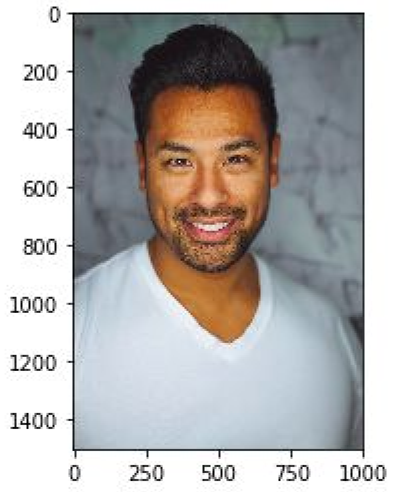


```
You look like Wirehaired pointing griffon
```

Fig. human result

Fig. human result

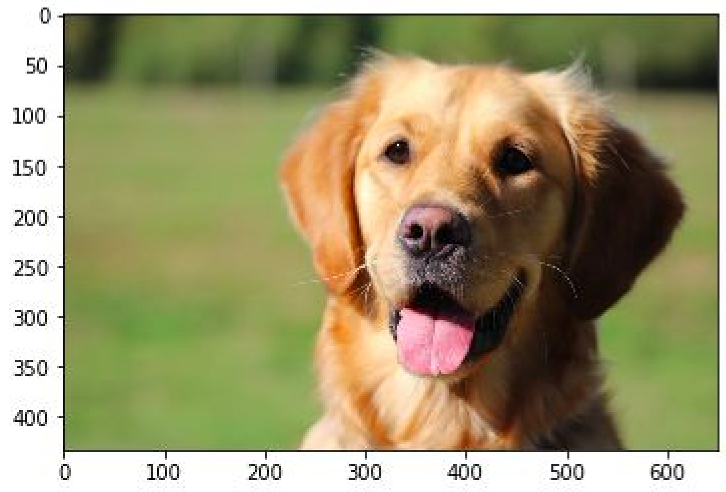
```
Hello, human!
```



```
You look like German pinscher
```

Fig. human result

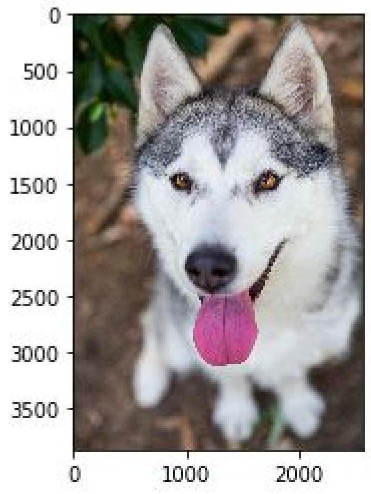
```
Hello, dog!
```



```
You look like ...
Golden retriever
```

Fig. dog result

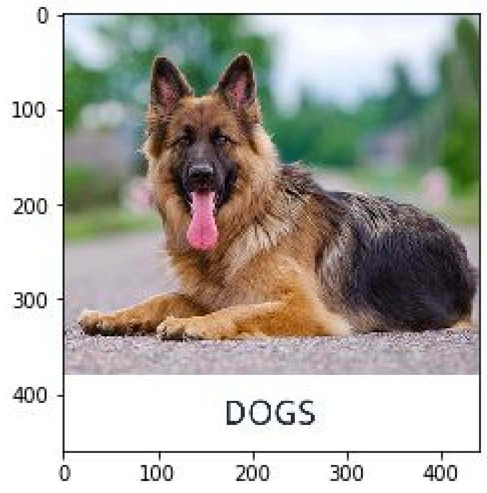
```
Hello, dog!
```



```
You look like ...
Alaskan malamute
```

Fig. dog result

```
Hello, dog!
```



```
You look like ...
German shepherd dog
```

Fig. dog result

- **Reflection**

  The process used for this project follow an end to end machine learning pipeline and can be summarized using the following steps:
  - Image acquisition
  - Image preprocessing, cropping, normalization, …
  - Image transformation, convert to format compatible with training algorithms.
  - Model creation. Training and hyperparameter tuning.
  - Model evaluation.
  - Model deploying.
- **Improvement**

  To improve this I think to use an appropriate model for this use good hardware that's enhance a training process, also can we try to enhance a classification accuracy and make deploy for the model.

  - Project Github repo: https://github.com/Mostafa-ashraf19/CNN_Dog_Breed_Classifier.git