Capstone Project — Raviteja Reddy

Machine Learning Engineer Nanodegree — Sep 30, 2019

1. Definition

Project Overview

Classifying objects into different classes is a very common thing we do every day. For example, consider an individual who is an ardent car lover. Whenever he sees a car on road, he will classify it into which brand it belongs to. So, tagging an object in to one of different classes is very common and we do it with out our knowledge.

But what if we would like computer to do that classification task without involvement of humans. In this project I will be creating a model capable of classifying dogs in to 133 different breeds. This approach falls into the domain of computer vision where we are making computer learn by looking(vision) at images of different breeds of dogs. I will be using dataset provided at this location
https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification

Problem Statement

Main goal of this project is to develop a model that is capable of classifying dogs in to 133 different dog breeds. If a human image is provided to the model it will come up with the dog breed that resembles that human. Since we are working with images, I will be using convolutional neural networks to model my algorithm. Steps followed are:

- Use pretrained face detectors to detect human faces in images. Assessing this detector on our images.
- Using pretrained VGG-16 model to detect dogs in images. Again, assessing this model on our images.
- Create a convolutional neural network from scratch to classify dog breeds.
- Create a convolutional neural network using transfer learning to classify dog breeds.
- Test our algorithms.

My final model will be able to display the input image with corresponding label saying whether it's a dog or human and what breed it belongs to.

Metrics

I will be using accuracy as my metric to measure performance of my model both on training, validation and on test sets.

Accuracy = Number of times our model correctly classifies/Total number of cases

Since my use case deals with multi class classification, accuracy is the most common metric unless there is imbalance in the data. But in my case, there is no imbalance, so I decide to go on with accuracy.

2. Analysis

Data Exploration

Data for the project is from two sources which contain both images related to dogs and images related to humans. Dogs dataset is further divided into training, validation and testing sets. Each of the three datasets contains 133 folders, each corresponding to different dog breeds. So, each folder has one class label. But human's dataset has 5749 folders, each corresponding to images of different humans.

Size of dog images in different datasets. All are RGB images.

Training data - 6680 images of size 300 * 400 * 3
Validation data – 835 images of size 650 * 432 * 3
Test data – 836 images of size 320 * 314 * 3

Size of human images – 13233 images of size 250 * 250 *3.
In our dataset labels for 133 different breeds are provided by the way of folders. Each folder has one class label.

Exploratory Visualization

Plot below shows how different classes of dogs are distributed among training data.

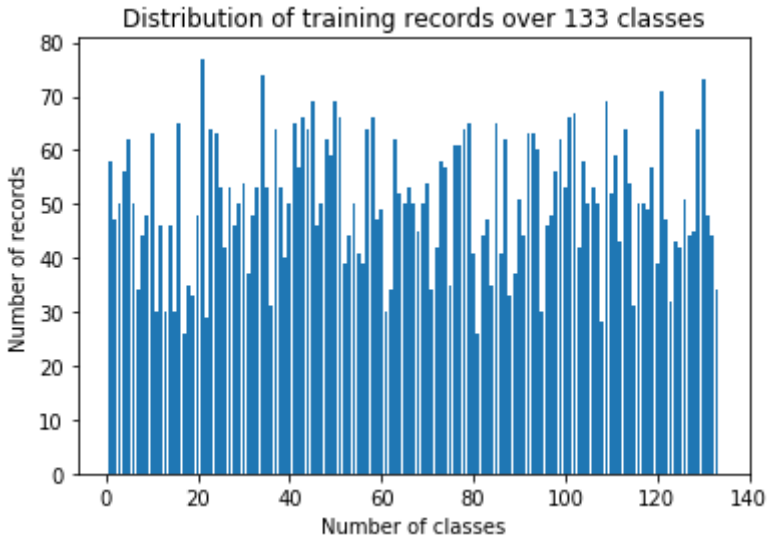

Fig. Shows how different dog breeds are distributed among training data

From above graph its very clear that data is not imbalanced. This is very important because if data is imbalanced, we cannot use our regular metrics like accuracy and most of the machine learning algorithms gets benefited from having balanced data. But one other thing which we can infer from above plot is number of images per class are limited. This will have some negative impact on performance as neural networks are data hungry algorithms.

Also, I see different dog breeds have minimum interclass variation. For example, Curly-Coated Retrievers and American water spaniels. Even though they are from different class they look alike. So, model will have hard time finding difference between them.

Fig. Shows pictures of Curly-Coated Retrieve and American Water Spaniel. They look alike

Algorithm and Techniques

I used Convolutional neural network architecture through out the project as I am dealing with Image data. Since my use case contains many sub projects, I will use more than one architecture.

First goal is to detect humans face in images. For this I will be using OpenCV's pretrained face detector. I opted this method mainly because my main goal is not detecting faces even though face detector will be helpful whenever my model sees humans face.

Second goal is to detect dogs in images. For this I will be using pretrained VGG-16 model which was trained on huge ImageNet data. Again, I opted this pretrained method mainly because my main goal is not just detecting dog images even though dog detector will be useful whenever my model sees dogs face.

For classifying dog breeds, I will be writing algorithm from scratch using different layers of CNN's and also, I will be using transfer learning. I opted to follow this approach so that I can see performance of CNN's when little data with many numbers of classes are passed. Finally, for same reason of not having enough data to train our model, I want to leverage the help of transfer learning.

Techniques: Some of parameters I will be tuning when building the architecture.

- Number of layers
- Layer types (ex: Max pooling layers, Batch normalization etc.)
- Number of epochs
- Learning rate.

Benchmark

I couldn't find a similar project where I could compare my results with. So, I will be comparing my results with VGG16 model built on ImageNet data. Even though VGG16 model is not just classifying dog breeds but since its trained on around 1000 categories of which some are dogs, I decided this would be the best approach. VGG16 has got the accuracy of 91% on ImageNet data. But my goal of this project is to get at least 10% accuracy with model built from scratch and at least 60% accuracy with transfer learning. So, these will be my actual benchmarks even though I am going to keep VGG16 accuracy in mind.

3.Methodology

Data Pre-processing

Following pre-processing steps are applied to images in training, validation and testing data of dog images.
Training and validation data pre-processing transformations in below order:

- Images were resized to the size 256 * 256
- Randomly changed the brightness, contrast and saturation of image
- Horizontally flip the images randomly with probability of 0.5
- Finally make image to 224 * 224 size using both random cropping and resizing.
- Normalize the image with mean and standard deviation.

Test data:

- Images were resized to 224 * 224.
- Normalize the image.

All the images were resized to 224*224 mainly because most of famous architectures like VGG16 resize the images to 224 * 224 before training. Also, after some trial and error found that horizontal flip and changing brightness, contrast and saturation are giving better performance.

Implementation

Implementation was carried in 4 stages.

1) Detect humans.
2) Detect Dogs.
3) Create a CNN to classify Dog breeds (from scratch)
4) Create a CNN to classify Dog breeds (using Transfer learning)
5) Algorithm to use above Transfer learning model.

1) Detect humans:

- For detecting humans used OpenCV pre-trained classifier to which an image was passed, and it gives number of faces detected and bounding box around the image.
- Implemented face_detector(…) function which takes image path as input and returns true if human face is detected in the image.

2) Detect Dogs:

- Used VGG-16 pretrained model on ImageNet for this task.
- Implemented VGG16_predict(…) which takes image path as input and returns index corresponding to VGG-16 models' prediction i.e. outputs one of 1000 categories.
- Implemented dog_detector(…) which takes image path as input and return true if dog is detected in image.

3) Create a CNN to classify Dog breeds (from scratch)

- Define the CNN architecture.
- Define loss function and optimizer with learning rate.
- Train the model for n-number of epochs on training data and validating on validation data at same time printing both training loss and validation loss.
- If the accuracy on validation data after n-epochs is not good enough return to first or second point and change architecture or optimizer or learning rate or number of epochs.
- Save the model parameters.

When training my CNN model from scratch I had to do some trial and error initially in selecting number of layers, learning rate. Batch normalization in every layer really helped in improving performance. But once I got some stable model just had to tune parameters like number of epochs. Initial process of trial and error to get to decent architecture is the only complication I faced in whole process.

4) Create a CNN to classify Dog breeds (using transfer learning)

- Define the CNN architecture. Used VGG16 for transfer learning. VGG16 has 6 fully connected layers at the end with last layer outputting one of 1000 categories.
- Freeze VGG16 model weights.
- Change the final layer in VGG16 to accommodate for our use case so it outputs 133 categories.
- Make only last layer trainable.
- Define loss function and optimizer with learning rate.
- Train the model for n-number of epochs on training data and validating on validation data at same time printing both training loss and validation loss.
- If the accuracy on validation data after n-epochs is not good enough return to first or second point and change architecture or optimizer or learning rate or number of epochs.
- Save the model parameters.

5) Algorithm to use above transfer learning model.
    - Implemented predict_breed_transfer(…) function that takes image path as input and predicts the class label using transfer learning model.
    - Implemented run_app(…) function which again take image path as input and outputs the image and class label in required format(can be specified by user).

Refinement

For building CNN from scratch initially I started with very simple architecture with decent number of epochs. When I trained it on my data and compared both train loss and validation loss, I found that my model is heavily underfitting. I got around 1% accuracy on both train and test data. Then I went on for some complex architecture and found that model is overfitting because our data size is small. In this case my train accuracy is better, but test is not.
So, performance was improved by using following techniques.

1) Adding batch normalization in ever layer.
2) Using optimal number of layers and filters in architecture.
3) Training with a range of learning rates and selecting the one with best validation loss.
4) Drop out layer did not help much so opted out of using it.
5) Augmenting the dataset with flips and changing saturation, brightness and contrast levels.

Final model was trained in iterative fashion adjusting above parameters (learning rate, number of epochs). The final model has accuracy of 12%.

Same approach is followed when building CNN from transfer learning. By adjusting learning rate, number of epochs, and just training last layer of VGG16 model accuracy has gone up to 75%. Since CNN with transfer learning yielded best results I will be using that for model evaluation and validation.

4. Results

Model evaluation and validation

VGG16 with last layer trained with our data will be used as final model. One of the main reasons for selecting this is since we have very less data, it is very difficult to build the model from scratch and get decent accuracy. So, leveraging features which are trained on bigger dataset like ImageNet will help a lot when we have less training data. Since my goal is to get at least 60% accuracy on test and my final model delivered 75% accuracy its aligning with my solution expectations.

To verify robustness of my final model, tested my final model with around 6 images of dogs. I see that even though my model was able to correctly classify dog breeds in most cases, but it is getting confused when there is minimum interclass variation for example it confuses between ‘Mastiff’ and ‘Bull Mastiff’. For example

Fig. Model predicting it as Bull mastiff when it is Mastiff.

Training for a greater number of epochs would definitely improve performance. I stopped at 15 epochs because it is taking too long to run.

Justification

Even though I mentioned that VGG16 as my benchmark model, my benchmark accuracy I desired to get was 60% mainly because dog breed classification is very difficult task with me having very less data (around 50 records for each class).

VGG16 has accuracy of 91% on 1000 categories when it was trained on around 14 million images. So, we getting accuracy of 75% with around 6500 images is decent job. But as seen above my model has problem when we have minimum interclass variation (in other words when two breeds look alike). But overall model does a decent job.

5. Conclusion

Free-Form visualization

Main quality of the project which I want to emphasize is that even with very small training set of 6500 images and with the help of transfer learning I was able to 75% accuracy on test data. Not only my model classifies different dog breeds, but when an image of a human is supplied to it, it can return resembling dog breed. Refer to below image.

Fig: This human when provided to model thinks it as Irish water spaniel. Right is the pic of actual Irish water spaniel.

Good things aside my model also suffers from minimum interclass variation.

Fig. Model predicting it as Bull mastiff when it is Mastiff.

Reflection

The process followed in the project can be summarised as:

- Data was downloaded which are part of public datasets
- Data pre-processing was done.
- Benchmark was created for final classifier. Benchmark accuracies were also locked.
- Using OpenCV pre-trained model face detector was created.
- Using pre-trained VGG-16 model dog detector was created.
- CNN was built from scratch and trained using data multiple times until good set of parameters were found.
- CNN using transfer learning was built and trained with data until good set of parameters and optimal accuracy on test data is found.
- Simple algorithm was written to use the above CNN model.

Of all the above steps I found building CNN from scratch the most difficult part, as I had to come up with architecture, learning rates, optimizer, epochs and all of these can be tuned.

So, coming up with perfect values for all these is most difficult part. At same time this is also my most interesting part. Understanding how all these factors effect final accuracy, what things need to be tuned to make model not underfit/overfit are most exciting part of the project.

Improvement

I see main improvement that can be made to the model is to train not just the last layer but last two layers of VGG16 model. I opted to train just one layer so that it can train faster. One advantage of training last 2 layers is that VGG16 model will adapt more to the data at hand so it will generalize well to dog breed images. Also trying ResNet50 or Inception could result in better accuracies which can also be tried.

Getting more training data will help as model will have more data to learn difficult patterns for example like differentiating between Bull mastiff and mastiff.

One of the things which could be done is to do training on more powerful GPU's so

1) we can train for more epochs.
2) Grid search over large range of learning rates and get the one with best validation accuracy in small amount of time.

I definitely feel If I made above changes and run on powerful GPU's for longer amount of time better solution exists.