Machine Learning Engineer Nanodegree

Capstone Project

Ahmed Awad
October 3rd, 2019

I. Definition

Project Overview

Autonomous vehicles had always been a dream. Imagining a car driving itself without any human interference was science-fiction and was first introduced in movies. Today, as we talk there are many companies out there building autonomous cars.

This project aims to make a car drive inside a lane without any human interference. This is a simpler problem than a full autonomous car. This project is related to computer vision Domain.

Using supervised learning, a machine learning algorithm is used to do the task. The dataset used is a labeled dataset of pairs (image, steering command).

Problem Statement

This project focuses on one task from the tasks of a full Autonomous vehicle. The task is **In-lane Driving** where the car should learn to identify the lane and stay driving within the lane.

The problem solution is a convolutional neural network (CNN) that will be trained on a labeled dataset in a supervised manner to do the task.

This will be viewed and treated as a classification problem instead of regression and the reason is that I will use my self-collected dataset which is collected using a monocam mounted on an RC toy car which doesn't steer with angle. Also, I made my own lanes using white paper sheets.

Note that: my RC car has only one angle of rotation to the left and to the right which justifies the classification approach with 3 classes.

The sampling rate that is used for collecting the images/dataset is 5 fps to avoid redundant frames. Also, tried to make the three classes balanced and avoid biasing towards the forward class. I collected 15k examples in my dataset.

Metrics

Accuracy score will be used as evaluation metric. Accuracy is calculated as the number of correctly classified examples over the number of all examples.

Of course, for accuracy to be a good metric, the collected dataset will be downsampled till all 3 classes contribute to 33% of the total dataset. Also, after splitting into train and test, the test set will be downsampled if it's note already balanced.

II. Analysis

Data Exploration

The **dataset** consists of 15700 examples exactly in the form of an input RGB image and the target is the class (Right, Left, Forward) which is used to control the car. All images are consistent in dimentions (256, 455, 3).
In the following figure, a sample from my collected dataset is provided.

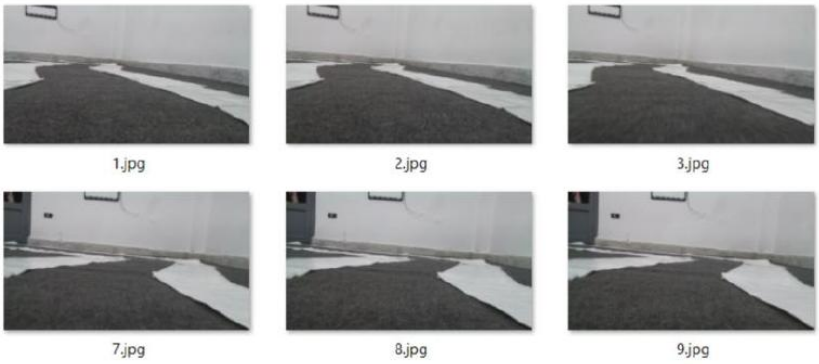

Fig (1)-Samples from my dataset

Further more, let's view some examples from each class to further understand the nature of the dataset.

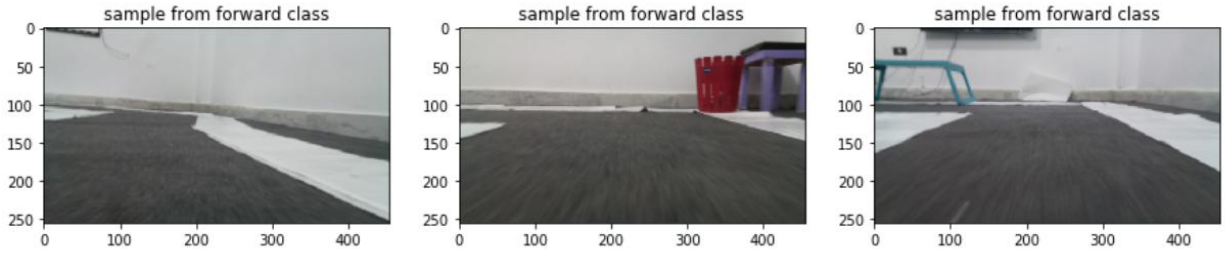

Fig (2)- Samples from forward class

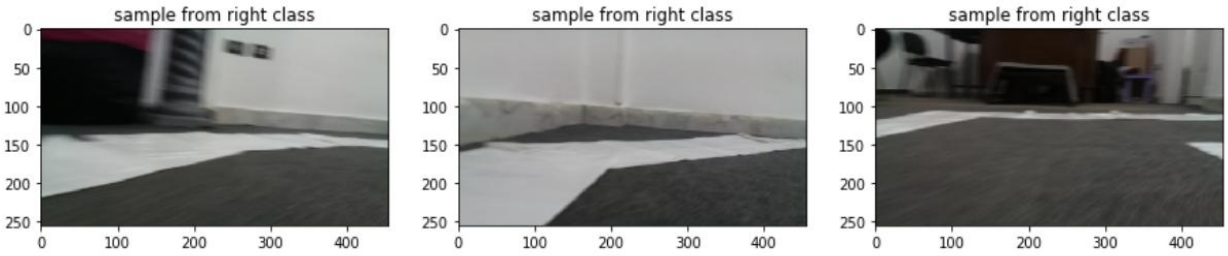

Fig (3) – samples from right class

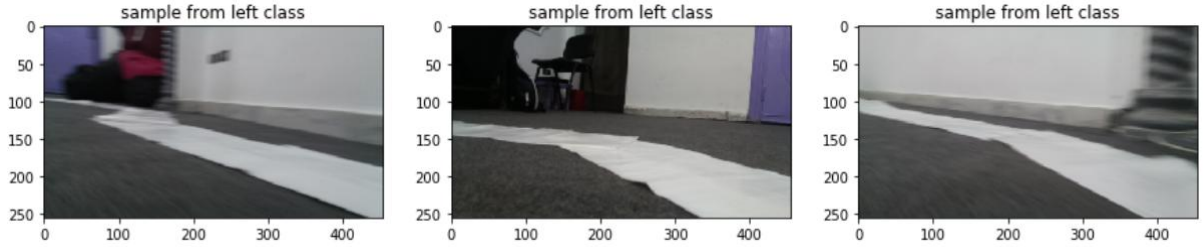

Fig (4) – samples from left class

Exploratory Visualization

After we had a better sense of the dataset nature and viewed some examples from each class, in the next figure we will get statistics about class ratios in our dataset.

Non-valid bar refers to some records that are corrupted and are not labeled with one of our 3 labels. These records will be removed in preprocessing.

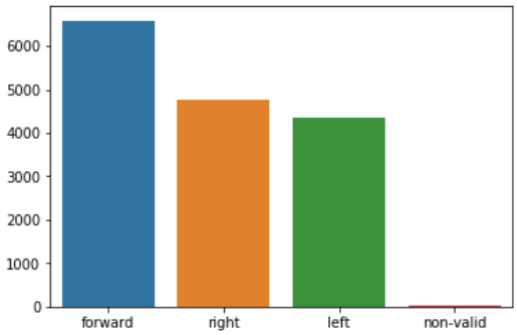

Fig (5) – class ratios in dataset

Algorithms and Techniques

For this problem, we will use **deep learning and convolutional neural nets**. The convolutional part of the network learns useful features from raw images and then these features are used by the later fully connected layers that will give the classification.

- Neural Network architecture:
    - Number of layers.
    - Layer types (conv, fc, pooling).
    - Kernel size in conv layers.
    - Number of nodes in fc layers.
- The learning algorithm used for training is **ADAM** optimizer which is a combination of **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems) and **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).
- During training, data is loaded into memory in batches one by one to avoid memory overload. Also, the validation set is loaded into memory to evaluate performance after each epoch of training.
- Implementing the model will be done in Keras framework.

Benchmark Model

For this task a naïve benchmarking is used where all images are classified as forward class.
This benchmarking was performed after making sure the test set classes are balanced and each contributing to 33.33% of the test set. This benchmarking gives an accuracy of 33.3% as well.

III. Methodology

Data Preprocessing

In order for the data to be ready for feeding to the model, multi-stages of preprocessing were performed on the data. In this section we will discuss each step of preprocessing in details.

❖ **Data Augmentation**

➢ **Flipping**

- All images were flipped horizontally.
- The label of the new images is also changed such that it goes well with the new flipped image.

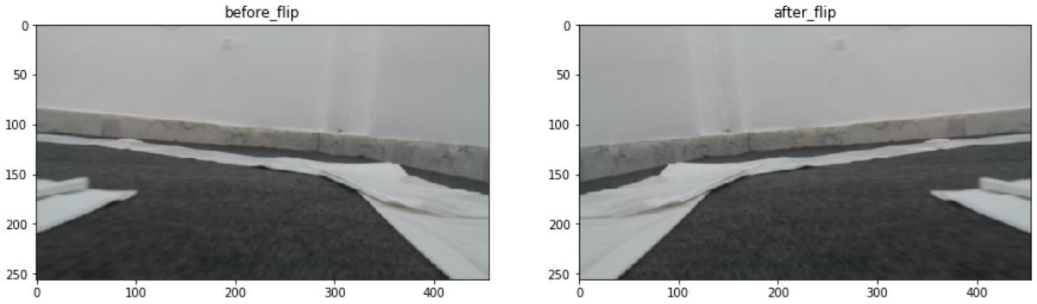

Fig (6) – shows the result of flipping augmentation

➢ **Contrast**

- Contrast enhancement is applied to all images.
- The label was kept unchanged.

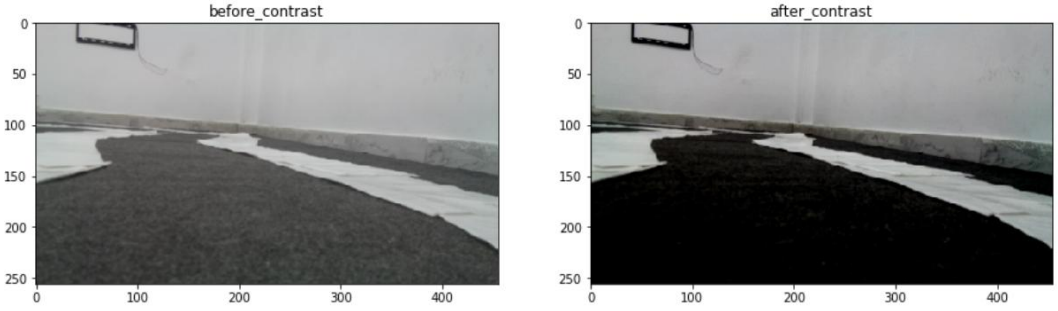

Fig (7) – shows the result of contrast augmentation

❖ **Shuffling**

- The csv file which contains image names in addition to the label is shuffled after augmentation using a pandas dataframe.

	img	forward	right	left
0	17546.jpg	0	1	0
1	48191.jpg	1	0	0
2	2977.jpg	0	1	0
3	51541.jpg	0	0	1
4	13236.jpg	1	0	0
5	39230.jpg	0	0	1
6	57295.jpg	0	0	1
7	31387.jpg	0	1	0
8	25567.jpg	1	0	0
9	127.jpg	0	1	0

Fig (8) – shows the dataset csv file after shuffling

❖ **Downsampling**

- The csv file was split into three dataframes, each one containing the examples from one class, after that each dataframe was downsampled then the three dataframes were concatenated again and shuffled once more.
- During splitting the data into three dataframes, the non-valids records were ignored thus, when concatenating the three dataframes, the produced dataframe is clean of non-valid records.

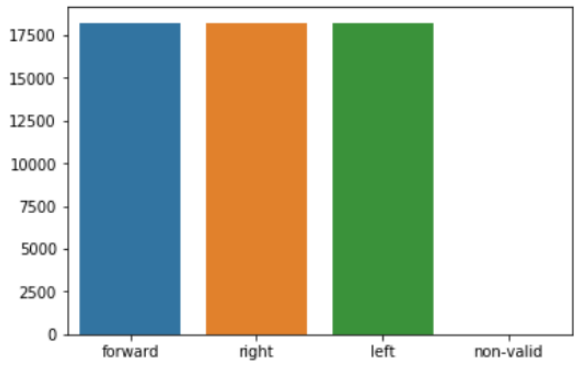

Fig (9) – shows the result of downsampling the dataset.

- **Splitting**
  - After downsampling and reshuffling, the dataset was split into train, validation and test sets with ratios 75:5:15 respectively. Fortunately, the test set was found to be class-balanced after splitting so there was no need for further downsampling.

- **Cropping and resizing**
  - While generating batches fom data, the loaded images are cropped such that the lower region which is the region of interest (the track part of the image) is used then the image is resized to be (3x66x200).

- **Input Normalization**
  - The image pixels are divided all by 255 as a normalization step before being fed to the neural network. This normalization step helps for faster and better convergence of our model.

Implementation

- The Project was carried out using google colab.
- The project is divided into two parts, each part has it's own jupyter notebook.

  - **Part (1):**
    - Data visualization.
    - Data Augmentation and shuffling.
    - Then, the augmented data version was uploaded to google drive so I can load it fast instead of re-executing augmentation code every time the session timeouts.

  - **Part (2):**
    - Continued data preprocessing (downsampling, splitting, cropping/resizing and normalization).
    - Model implementation
    - Model Training and evaluation metrics
    - Model refinement through transfer learning on VGG16 model.

Model Architecture

The model used is a typical CNN architecture of stacked conv layers for feature extraction followed by a fully-connected layer for classification. Regularization techniques were used to avoid overfitting such as dropout and L2 regularization. This model was trained for 30 epochs.

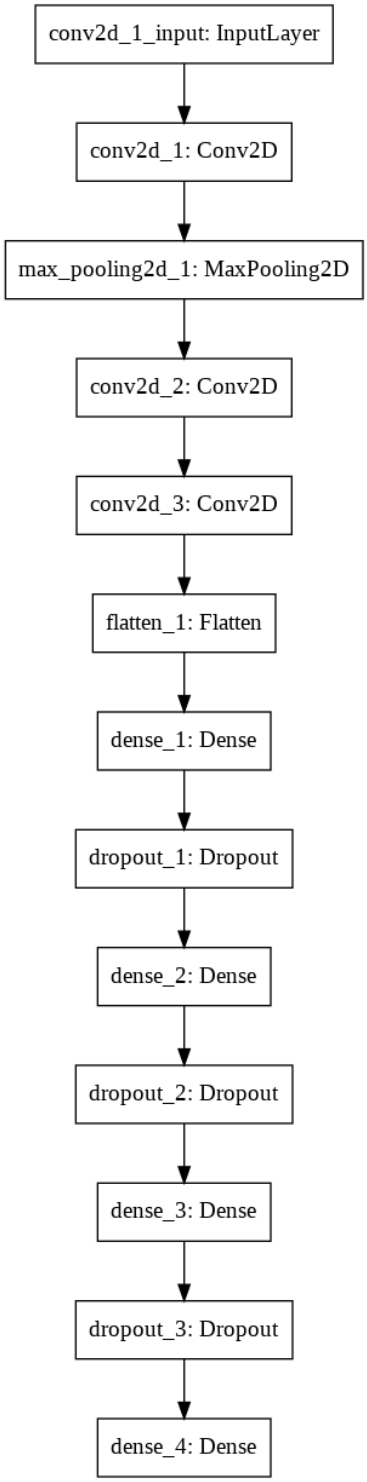

Fig (10) – shows model architecture

Model parameters and hyperparameters

- **Conv layers:**
  - Kernel size (5,5) for the first and third conv layers.

- Kernel size (3,3) for the second conv layer.
- All layers have stride (2,2).
- The first layer has 12 filters, the second has 16 filters and the third one has 24 filters.

➢ **Fully connected layers:**

- 4 Dense layers of 1000, 100, 10,3 nodes respectively.
- A dropout layer is used between each two successive dence layers.

Transfer Learning

Another CNN model was used which is VGG16 trained on Imagnet. The conv layers of the VGG16 model where used as a feature extractor and their weights are fixed and the fc layers were discarded. A new fully-connected network was added after the conv part and this fc part is left trainable. This custom model was trained for 30 epochs with the same optimizer.

IV. Results

Model Evaluation and Validation

➢ **The first model:**

- Evaluating the final model on the balanced test set after training ends yields an accuracy score of 88.8%.
- This model has some symptoms of overfitting even after using regularization techniques as it gets accuracy 94% on training data which is larger than test data.
- This model is more suitable for real-time inference as it has much lower number of parameters than the VGG16 model thus have less computations done to get the output.

➢ **VGG16 customized model:**

- This model yields an accuracy score of 87.96% which is close to the first model.
- However, this model doesn't suffer from overfitting symptoms.

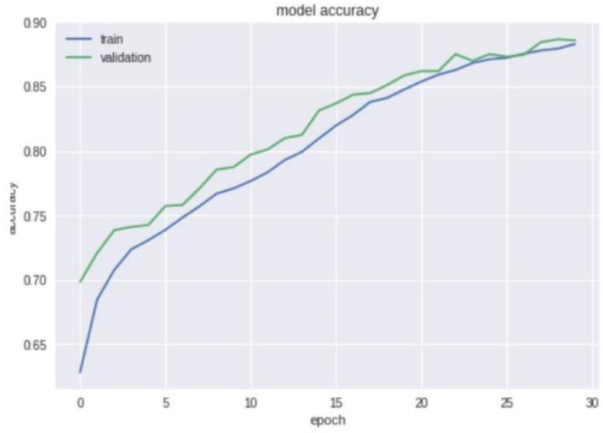

Fig (11) – shows training curves for train and validation on the VGG16 custom model

Justification

Using deep learning and neural networks, we have gone far away with accuracy from the benchmark. Our deep learning solution gives 88.8% and 87.96% accuracies on unseen data which really outperforms the benchmarks that gives 33.3% accuracy only.

This is due to the nature of concolutional nural networks that they can extract the really useful features that can determine the output significantly instead of using a statistical machine learning classifier which will need a lot of manual feature extraction. On the other hand, neural nets are data hungry and they really perform well when they have a huge amount of training data.

I believe we can further increase the accuracy of the model by collecting more examples and feeding more new data to the model. However, these results are acceptable taking into consideration that controlling the RC car manually is tricky and difficult to drive it inside the lane due to the absence of servo motors for steering. It only goes right or left with a fixed angle.

V. Conclusion

Free-Form Visualization

One characteristic of the problem is the simulation track manually made by white plastic sheets to serve as a simulation for a real road with a lane. Also, the car used for collecting the Dataset is an RC car which doesn’t have a continuous steering angle. The whole environment was set up indoors.

Reflection

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

1. An initial problem was proposed.
2. A dataset was collected from scratch using an RC car.
3. The data were preprocessed and split into train, validation and test sets.
4. A benchmark model was created and evaluated.
5. Two CNN models were used for solving the problem.
6. The two models were evaluated and compared to the benchmark model.

I found step 2 the most difficult as it was my first time collecting a dataset. While collecting the data I had to determine a good sampling rate to avoid redundant frames that are very similar. Also the track had to be designed carefully that it has a lot of turns so the forward class doesn't dominate the dataset.

Improvement

As said before, the model results can be further improved by collecting more training data as neural nets are data hungry.

Also, another solution can be used if I used my solution as a benchmark is using LSTMS and I think it can perform better than my solution where the frame and its previous frames are processed to give the output decision instead of depending on one frame only.

References:

[1] Nvidea's paper "End to End learning for self-driving cars":
https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

[2] Slides about Autonomous vehivles history:
https://www.mcca.com/wp-content/uploads/2018/04/Autonomous-Vehicles.pdf

[3] Wikipedia article about history of self-driving cars

https://en.wikipedia.org/wiki/History_of_self-driving_cars