Motor AI Perception Challenge

Controlling MountainCar-v0 Gym Environment using Webcam

Abhishek Dinkar Jagtap

September 19, 2022

Abstract

Traditionally Gesture detection in real time is achieved by developing a detection module network that helps in detecting hands in a given frame followed by a classifier module to produce classification outputs. This yields state of the art performance but is computationally very heavy when deploying in real time. We solve the challenge by a more data centric approach using basic computer vision algorithms. To this end we develop a custom gesture modeling dataset and deploy this model in real time to not only classify the gesture but also control the MounatinCar-v0 environment.

1 Introduction

Open AI Gym environments are used widely to train reinforcement learning algorithms, In our given task we will be using MountainCar-v0 Gym Environment. The car is on a one-dimensional track, positioned between two mountains also called as Sinusoidal Valley. The goal is to drive up the mountain on the right and reach the flag. Generally the car is underpowered to climb the hill, we need to create a momentum before accelerating to reach the flag and end the episode. Inorder to control the car we will be using hand gestures performed infront of the webcam.

2 Pipeline

Inorder to control the mountain car using gesture we will build a custom dataset and train a classifier to classify those gestures and pass the resulting outputs to gym environment for taking appropriate actions in real time. As we impose ourselves with constraint such as not using any pre-trained model of face/hand nor using existing datasets for gesture classification. we will rely on core computer vision concepts such as background subtraction, motion detection and thresholding. We will breifly discuss the steps involved in a sequential order from data collection to building a classifier and thereby controlling the car.

2.1 Creating Custom Dataset

As we have three states/actions to control MountainCar-v0 [left, right and stop], we will choose 3 gestures accordingly that can map to a certain action. Furthermore, we have complete control over the process of generating data, This allows us to apply intuitive ideas and solve/complete the objective using data centric approaches rather than traditional model centric appraoches. We will design the dataset in a such a way that any underlying features can be easily learnt from even a traditional low computation machine learning algorithms.

2.1.1 Dataset Design Strategy

As the video feed from the webcam can be of fixed size, we will choose a region of interest of $200*200$ size where we can perform different gestures. This way we can reduce the traditional heavy computational part of working with the entire size of an image with unwanted information, thereby discarding the whole detection stage involved in real time gesture recognition. Our approach would be to place our hand in the region of interest (ROI) and then record the gestures observed in the corresponding space. Furthermore, we can observe that for a typical gesture dataset geometric shape/orientation of the object is more important than the pixel level features coming from a 3 channel RGB image. So we will convert our RGB image into a black and white pixelated image, where black corresponds to background and white corresponds to the gesture shape. Thereby making the classification part more feasible.

To achieve this pixaleted black and white image we will rely on core computer vision concepts such as background subtraction and motion tracking. Firstly we will calculate the running average of the image without introducing hand into the webcam feed, and then compute the difference between the stationary background and the frame with hand. Furthermore to obtain the a precise shape of the foreground object we will introduce a thresholding parameter. This thresholding depends on different conditions such as lighting, shadows and contrast, Since we are dealing with indoor lighting conditions for our application we carefully experiment with different values to obtain optimum segmentation of our hand.

2.1.2 Types of Gestures Used

As the action space in our environment include the movement for [left, right and stop], predominantly the important movement action would be [left and right] which can also be translated as [Move Back and Move Forward] as the car needs to accelearte in both directions to reach the flag. So Intuitively we will choose two gestures [left and right] in a such a way that the difference between this gesture features are high as shown in the Fig.1. The stop action is controllable but does not yield much importance in the grand scheme of our task. So we choose the gesture that is slightly similar to moving left. As a result even in a case of wrong detection our intended task would still not be hindered, allowing a higher task accuracy.

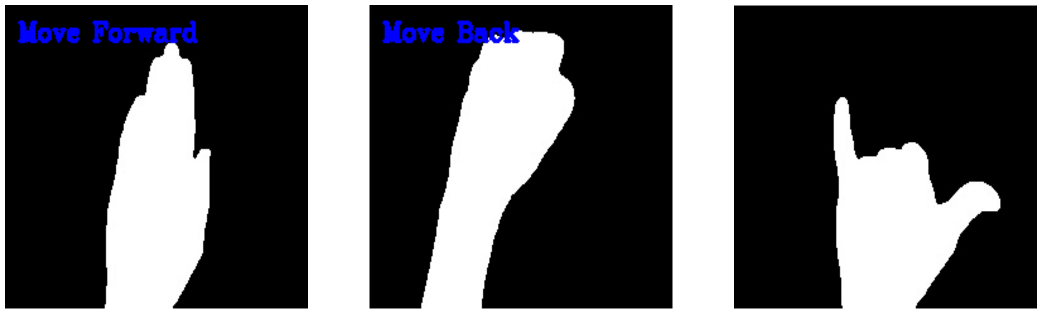

Figure 1: Gestures Mapped to Environment Action - Move Forward, Move Back and Stop

2.1.3 Plausible Risks in Dataset

Apart from data collection stage we are also preprocessing the frame during the inference, so our dataset might fail in the following conditions

1. If the lighting condition continously change
2. If shadows persits in Region of Interset (ROI)
3. If thresholding parameter is changed
4. Foreground mask can become inaccurate if camera is not calibrated at the beginning

   Reason: As we choose to use computer vision algorithms to obtain the mask for hand and gesture style, we incorporate basic pre-processing of the frames in real time. This pre.-processing steps becomes difficult to handle when we encounter various aforemention changes.

2.1.4 Properties and Statistical Info on the Dataset

As we sequentially discussed the process of generating custom dataset and the design strategy, we will look at some of the statistics involved in our dataset. We recorded nearly 600 images from each class as part of our training data.Furthermore, we will split the data into validation subset with an size of 200 images. As we are dealing with supervised learning, it is important to realize the data during the test holdout should include all the real world occurences. So the test holdout is carefully curated by selecting different thresolding value and lighting condition amounting to 350 images from all classes. Fig 2 shows a batch of test holdout with different threshold values and lighting conditions that can be observed in real time.

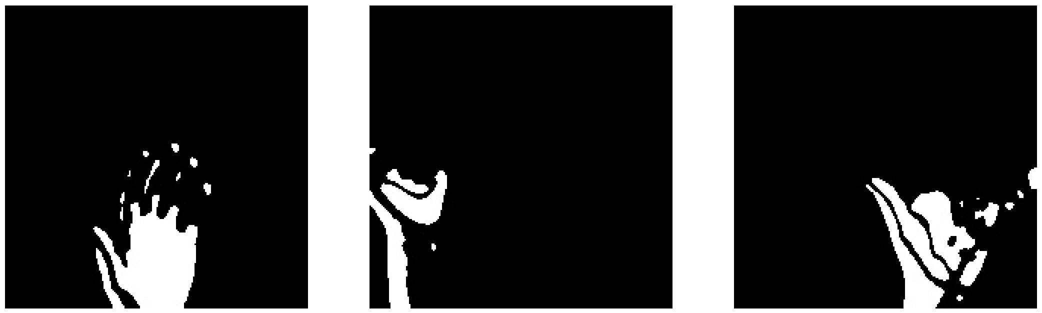

Figure 2: Test Holdout Under Varying Indoor Conditions

2.2 Building a Classifier

Now that we have a developed a gesture dataset with all 3 possible actions to incorporate in the gym environment. We will focus on building a CNN classifer that can predict these gestures in real time. As our inference is in real time and needs to keep up with the gym environment, we will build a basic Convolutional Neural Network which can efficently learn features of our dataset. By choosing a data centric approach to solve the problem, we have already developed a training dataset that can be easily classified by even using Support Vector Machines(SVM). Also using a pre-trained network would be an over kill as we are dealing with low-scale images with only one channel.

2.2.1 Implementation

We will stick to a baseline CNN model with just 4 convolution layers to better capture the scene context and a fully connected layer to produce classification outputs which is coupled with an intermediate max-pooling layer. We employ PyTorch deep learning framework in the implementations and training is done on a single NVIDIA 1660TI GPU with 6 GB memory. Our model is trained using Stochastic Gradient Descent optimizer with learning rate 1×102 and a batch size of 64. We use CometML Toolkit for visualizing the neccesary plots and keeping track of different hyper-parameters.

2.2.2 Results and Evaluation

As expected from a CNN, our training provides superior results of upto 95.81 % training accuracy and 93.7 % validation accuracy as shown in the Fig. 3. As our test holdout consist of various images with different lighting conditions and thresholding values as discussed in Section 2.1.4. Our models yeilds an classification accuracy of 77.3 on a test holdout. This drop is expected as we have not included any such variation of images in the training data. We do this because of a mere simple fact that a weak supervised classifier is better than a more rigid overfitted model trained on the whole data. The Fig. 3 also shows the training and validation loss over 30 epochs.

To Evaluate the performance of our model we will look into the confusion matrix during validation and testing as shown in the Fig. 4. As expected we can observe that our model fails to distinguish between moving left [Class 1] and stop[Class 2] on several occasion as the difference of features between left and stop are minimal.

Furthermore we can also use Recall and Precision to evaluate the performance of our classifier. F1 Score is widely used to measure the model performance for classification task.

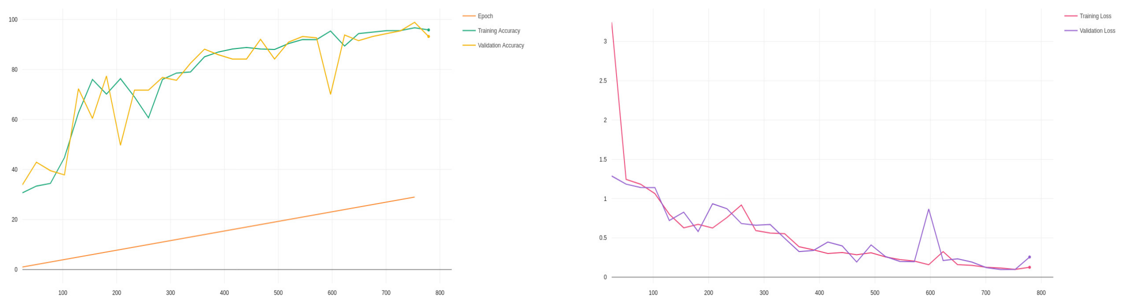

Figure 3: Illustration of Training and Validation Metrics

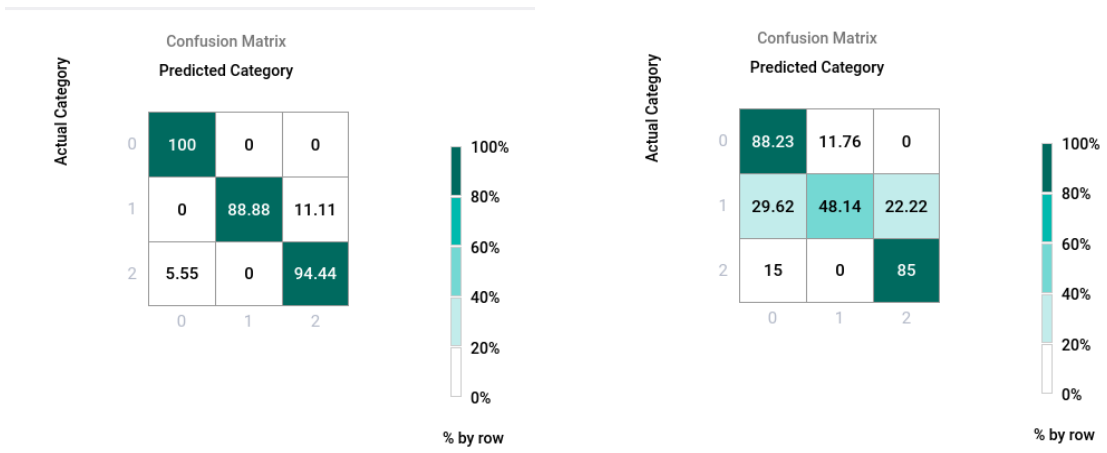

Figure 4: Left:Validation Confusion Matrix, Right:Test Confusion Matrix

2.3 Controlling MountainCar-v0 in Real Time

As we have saved our trained classifier, we will use this classifier to detect gestures in real time and simultaneously pass the outputs to mountainCar-v0 environment to control the actions. We will use the same pipeline that we used for collecting data as we need to pre-process the frames during inference to black and white pixelated images that can act as inputs to our model. The model predictions are mapped to a certain action and then forwarded to MountainCar-v0 gym environment. Then the gym environment takes this action and returns a new state. To obtain more precision in the control part we infuse a delay of 0.1-0.5 seconds between the prediction, action and the new state.

To evaluate the performance of the overall task in real time, we will store the probability scores of each predictions and display them on the webcam feed. We observe a slight drop in the classification performance when predicting in real time. This happens due to the drawback of pre-processing frames in real time. The Demo Video showing the consecutive successful control of the MountainCar is available at Google Drive.

2.4 Conclusion

In this work, we address the use of basic computer vision algorithms for gesture classification in real time. We can observe that even in a simple classification task how supervised training of a model hinders real world deployment. Few of the challenges encountered are handling the varying light conditions in an indoor setup. Furthermore, we can still optimize the model by choosing a network of more depth for better feature extraction and increasing the training data size by incorporating varying indoor conditions.