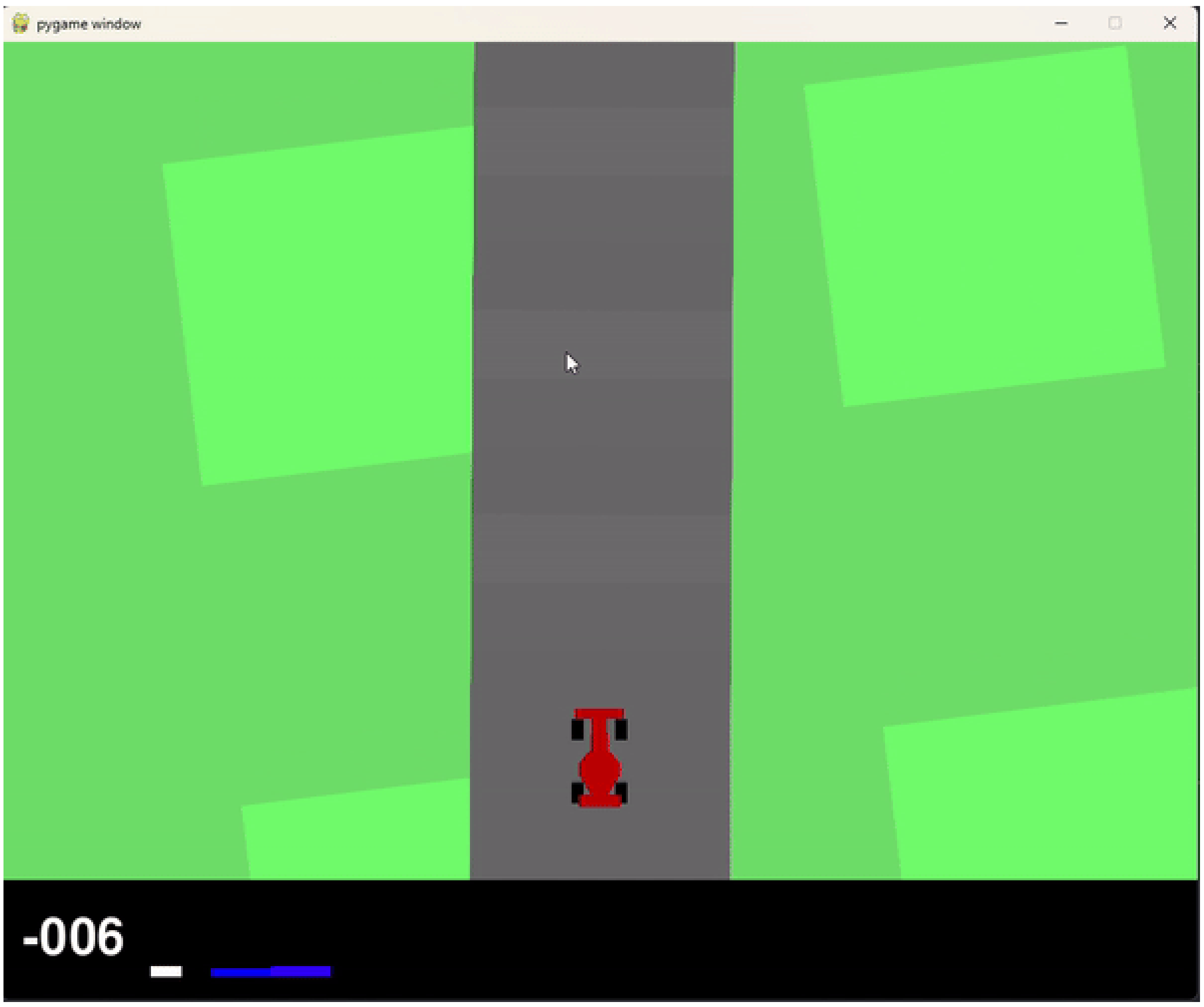
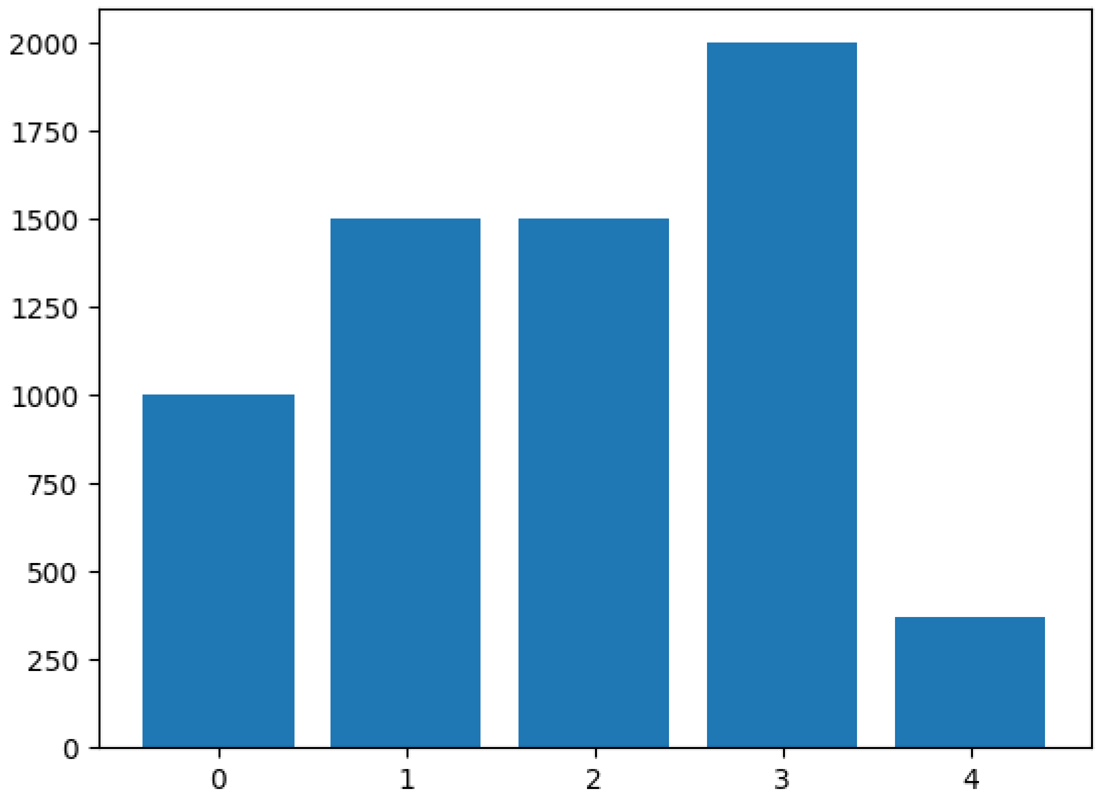
Introduction

While the field of deep learning continues to advance and mature daily, there remains a lack of consensus on the definitive approach to determine an optimal model. Unfortunately, I do not have access to sufficient computational power to test various models and parameters extensively. Consequently, the analysis will primarily concentrate on the obtained results.

The scenario involves a car operating within a simulated environment. The primary goal is to address an image classification problem, aiming to understand the behavior of a racing car within a Gym environment. Here's am example of the environment.



A thing worth noting is that the dataset is unbalanced



Addressing the imbalance within the dataset is crucial. However, it's important to note that the environment utilizes a technique called parallax scrolling, rendering the car as a stationary image. This, coupled with the absence of variations in brightness and frame rotations, indicates that implementing data augmentation techniques would provide minimal utility or improvement, so no actual trasformations are perfomed on the dataset.

Models Architecture

Model1 and Model2 have distinct architectures:

Model1 comprises two convolutional layers with 12 hidden units each, utilizing Rectified Linear Unit (ReLU) as the activation function. It incorporates max-pooling with kernel dimensions of 3 and strides of 1 and 3, respectively.

In contrast, Model2 employs a more intricate architecture, featuring four convolutional layers with 20 hidden units. Similar to Model1, the convolutional layers use ReLU as the activation function. However, it diverges in the max-pooling strategy, utilizing a kernel dimension of 2 for the first three layers and 3 for the last layer. Moreover, Model2 implements an incremental stride of 1, 2, 5, and 7 across its layers.

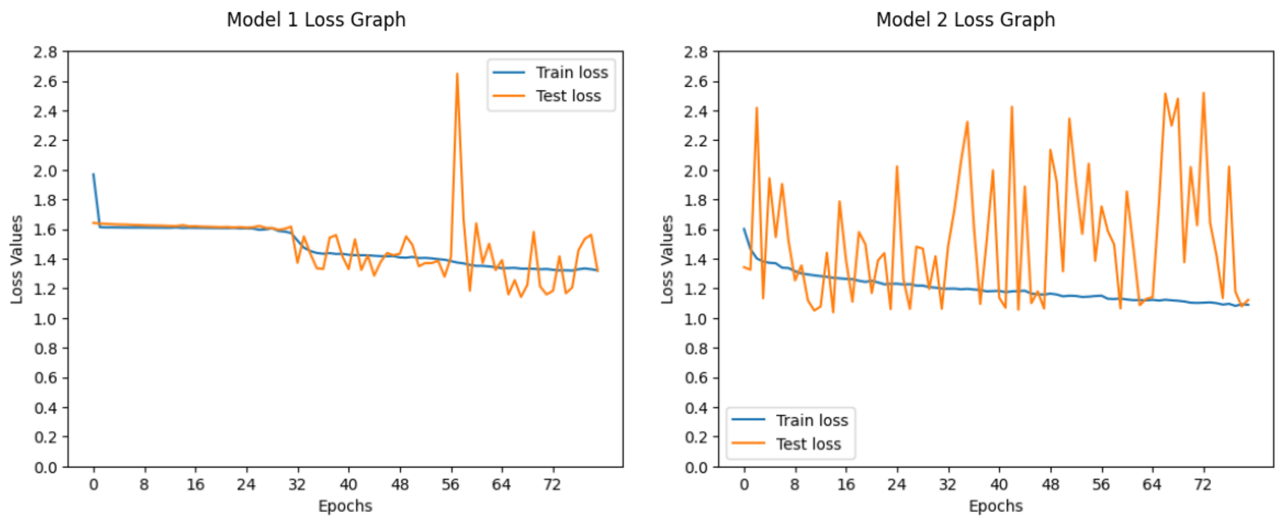
Both models end with a final linear layer. During the prediction phase, the output undergoes a softmax transformation, and the prediction is determined based on the highest probability obtained.

Models evaluation

To begin, I initiated the training of the two models across 80 epochs, with minibatches of size 32. To evaluate the model's quality, the initial step involved assessing the loss. PyTorch offers modularity in creating and training neural networks, granting comprehensive control over both the optimizer and the loss function.

For the loss function, I opted for CrossEntropy loss due to its flexibility in assigning weights to individual classes, considering variations in the number of examples per class.

Regarding the optimizer, I employed stochastic gradient descent (SGD) due to its fundamental nature and typically reliable performance. Here are the obtained results.

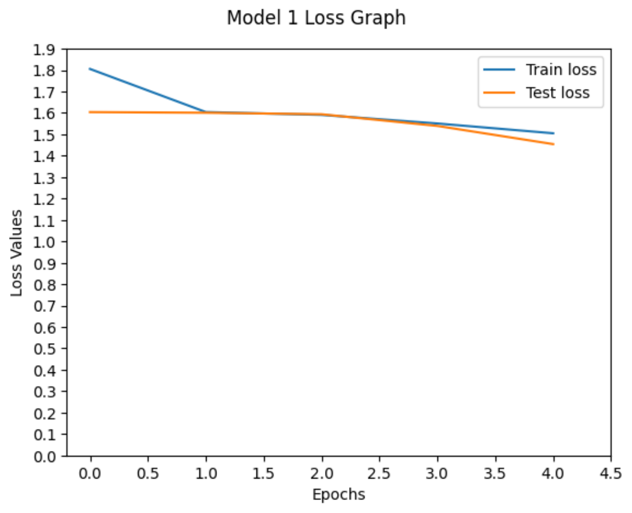


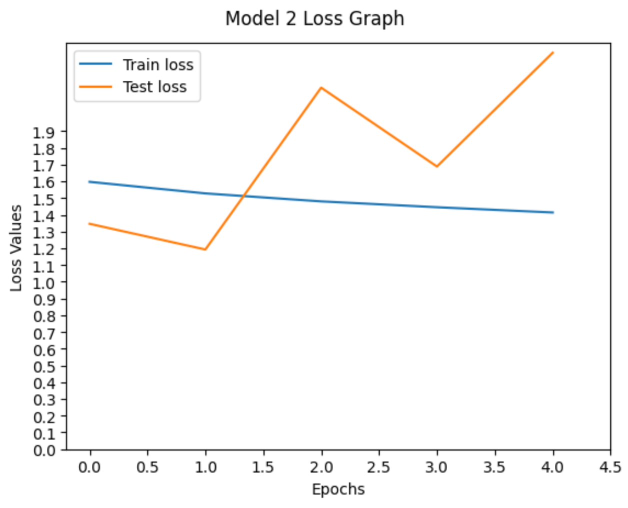
An important observation lies in the substantial fluctuation observed in the test loss, notably more pronounced in the second model. This aspect will be elaborated on further in subsequent sections.

While the high loss provides an indication of potential issues with the models, we will consider additional metrics to comprehensively assess the situation.

However, it's evident from the loss graphs that utilizing 80 epochs for training might be excessive, especially considering the significant time investment. The earlier mention of hardware limitations was crucial, as training these two models for this duration took approximately 3 hours. Therefore, a strategy to mitigate this involves reducing the number of epochs and implementing a technique known as early stopping.

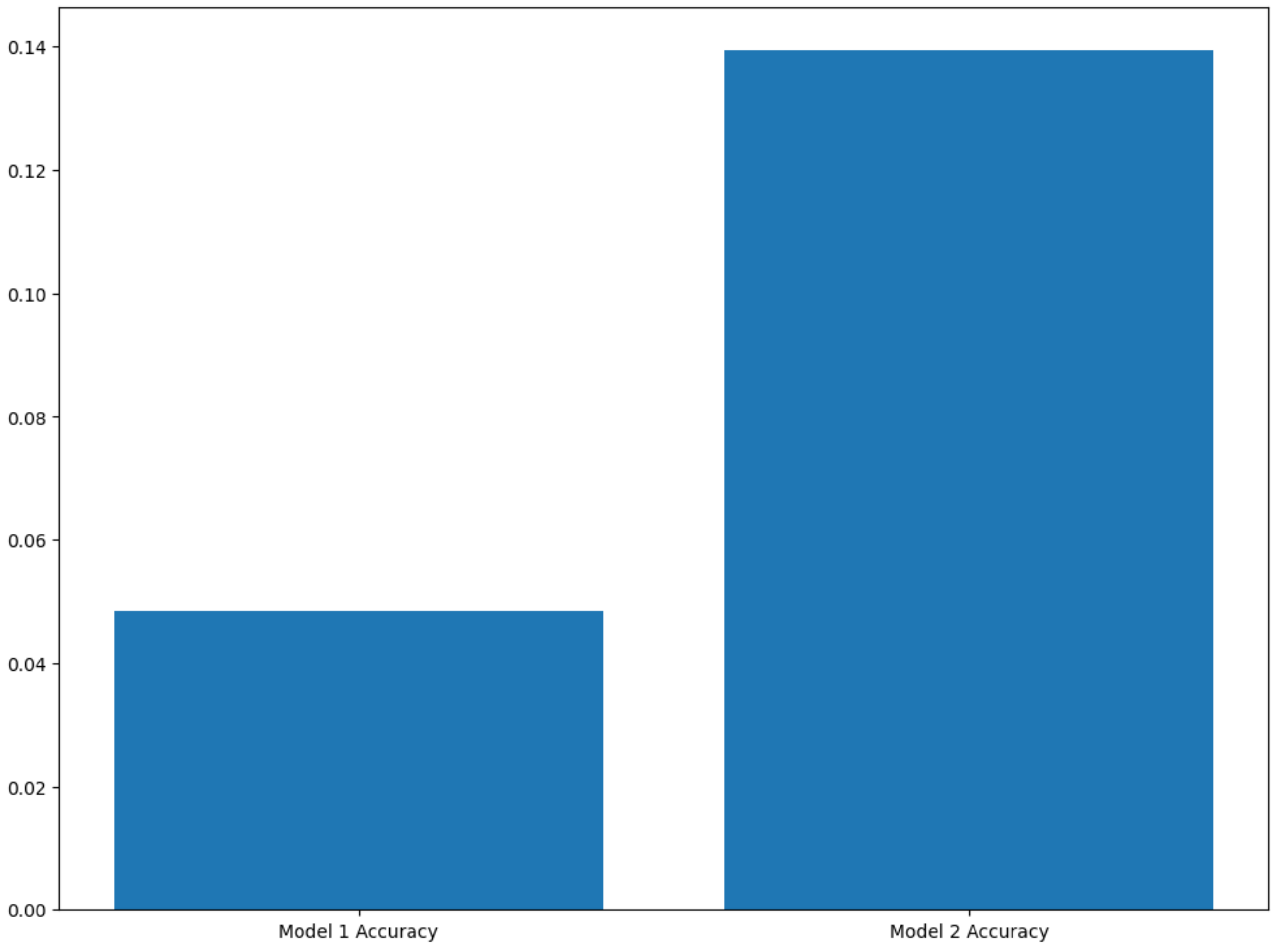
Preliminary analysis suggests that an optimal number of epochs might hover around 5. Thus, the plan is to retrain the models with this reduced epoch count and assess the outcomes.





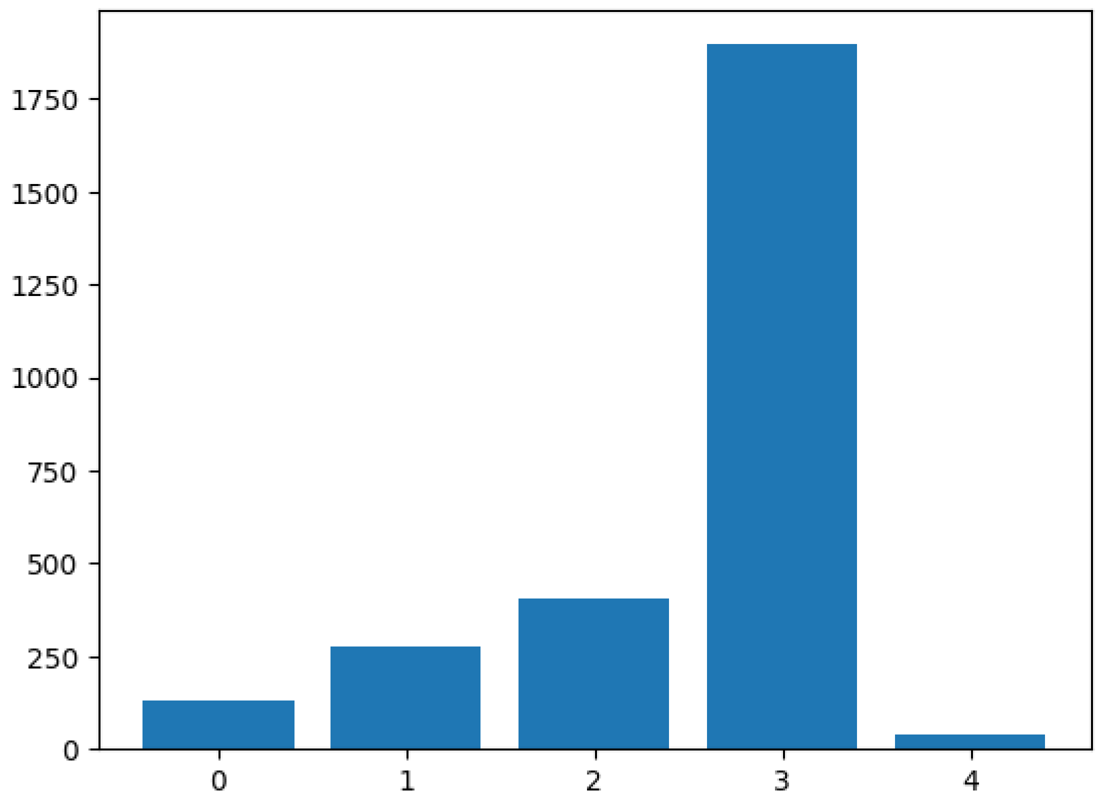
Accuracy

Despite achieving stability in the training loss for model 1, it remains important to establish a quantitative measure of its performance. Initially, let's focus on obtaining a snapshot of its accuracy.



In terms of accuracy, it appears that model 2 outperforms model 1 significantly, contrary to our expectations based on the progression of the models' losses. To understand this discrepancy, let's delve into the underlying reasons.

As highlighted in the introduction, we previously acknowledged the highly imbalanced nature of the training dataset. However, this imbalance becomes even more apparent upon closer inspection of the training data.



To gain a more comprehensive understanding of the data distribution, let's analyze the percentage of elements divided by class and keep these numbers in mind.

% of elements of Class 0	% of elements of Class 1	% of elements of Class 2	% of elements of Class 3	% of elements of Class 4
4.84	10.0	14.77	68.97	1.42

The observation that class 3 constitutes around 70% of the total dataset raises suspicions. One possible explanation for what we're observing could be as follows:

Model 1, being a basic CNN, exhibits an accuracy of approximately 20%, which suggests that it might be making random guesses among the five available classes.

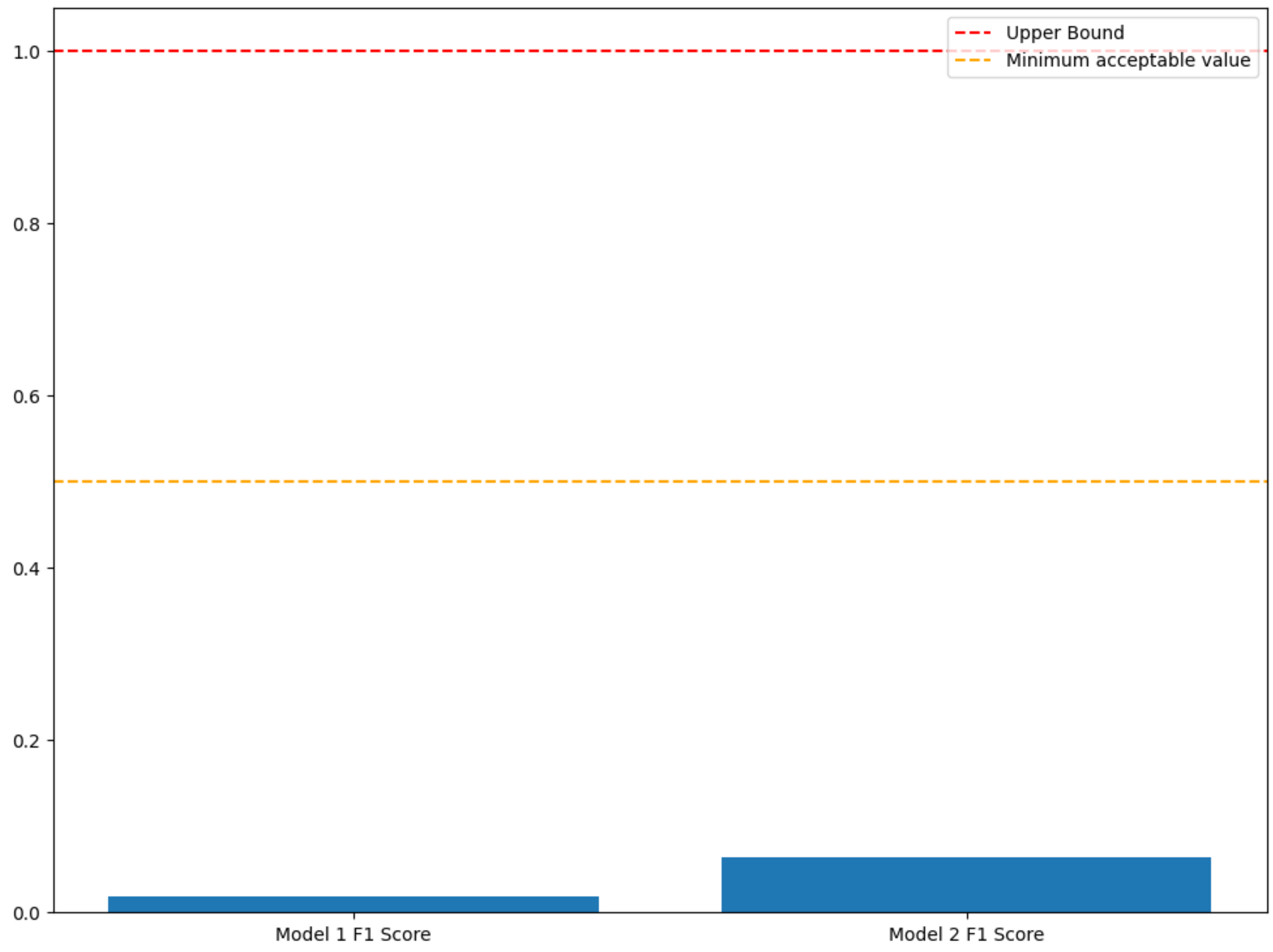
Conversely, Model 2 is more complex and deeper, implying a higher number of parameters and overall complexity. This complexity might result in overfitting, where the model essentially memorizes the training data, leading it to predominantly predict class 3 to increase accuracy. To illustrate what I'm trying to say, if we consistently predict class 3 for 100 examples, we would achieve around 70% accuracy, the model misclassifies only the remaining classes, which constitute a mere 30.

This situation highlights a well-known issue: accuracy is not a reliable metric for imbalanced datasets. The example provided serves as a demonstration of this limitation.

F1 Score

When handling an imbalanced dataset, precision and recall become preferred metrics. Precision represents the model's capability to avoid false positives, while recall represents its ability to avoid false negatives. An ideal model would display high values for both metrics, yet there typically exists a trade-off between precision and recall. The choice between the two depends on the specific application.

In this scenario, lacking clarity on which metric holds more significance to minimize, I have opted for another metric known as the F1 score, which combines both. Specifically, we will calculate the F1 score using macro averaging, thereby disregarding the number of support instances. This decision is rooted in the limitations of other averaging methods: the micro average aligns closely with accuracy, while the weighted average may excessively emphasize classes with a surplus of examples, such as class 3 in this case.



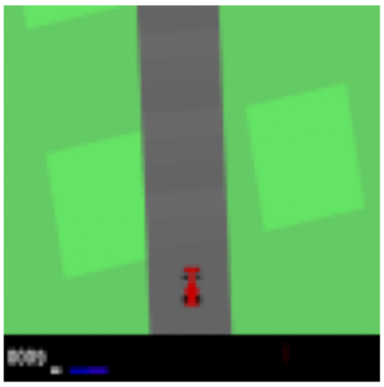
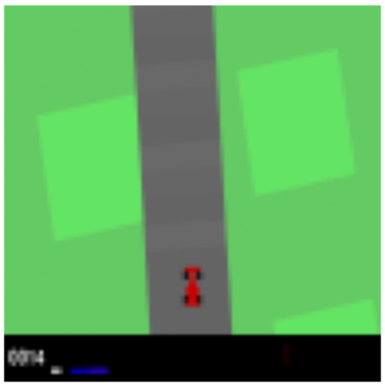
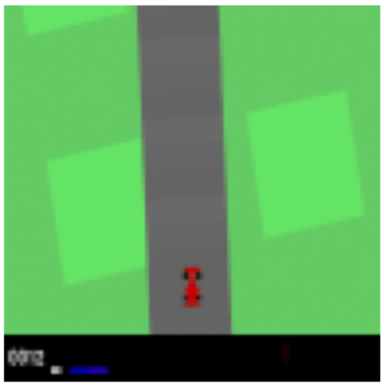
As depicted in the histogram, our achieved F1 score falls significantly below an acceptable minimum threshold, let alone reaching a good standard. This aligns with our previous assertions that these models' performance is notably inadequate.

Conclusions

But why might this be the case?

One prominent reason is the relatively small size of the dataset concerning the complexity of deep learning models. Consequently, it becomes easier for these models to escalate in complexity and overfit the data, a situation particularly noticeable in model 2.

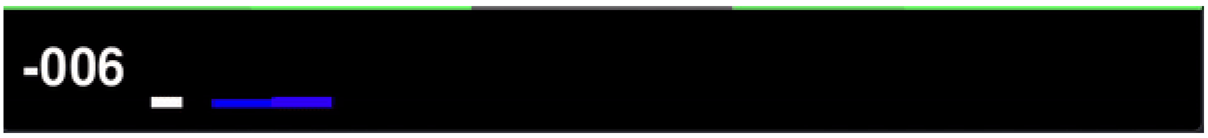
Moreover, the visual similarity among the photos doesn't let trivial models, like model 1, to generalize well. For instance, here are three sample photos extracted from the training dataset, each representing class 0, 1, and 3.

Upon examining these images ('data/test/3/0040.png', 'data/test/0/0048.png', and 'data/test/2/0003.png'), it's apparent that distinguishing among them is challenging, as I struggled to identify which image corresponds to which class. Considering these challenges, it's plausible that the models' limitations persist regardless of the extent of their training or specialization.

This leads me to believe that the dataset itself might not be suitable for the intended task. Allow me to elaborate:

In the introduction, I included a frame illustrating the environment, if we focus on the lower part of that frame.



The movements of the elements in the bar is evidently linked to the state of the preceding action (even though I could not find anything in the documentation), indicating that knowledge of the optimal next action is not only reliant on the current frame but also dependent on the previous frame and its state.

To illustrate this further, the car has three levels of acceleration. Decelerating by a level reduces the acceleration, indicating that regardless of the subsequent action, braking isn't a viable choice when the vehicle is stationary. Additionally, when the car is motionless, it's impossible to steer left or right, and idling results in point deductions. Consequently, the only viable option is to accelerate.

This example clarifies two aspects. Firstly, the predominance of class 3 (gassing) instances can be attributed to the fact that often, accelerating is the only feasible action. Secondly, the information conveyed by the bar is crucial. Unfortunately, due to the small size of the images, extracting information from the lower bar becomes exceedingly challenging, given its relatively small portion of the frame.

In my spare time, I attempted to train a model to address this issue, locating where the bar is (see frame_dataset.ModifiedFramesDataset) but unfortunately, I was unsuccessful (mainly due to the fact that the images convey no information and are mostly black).

Resources:

- What is parallax scrolling
- Why I choose 32 minibatches
- CrossEntropyLoss
- Explanation on why the loss is so volatile in model2
- Another Explanation on why the loss is volatile in model2
- Yet another explanation on why the loss is volatile in model2
- How to read F1 Score
- Why i choose macro averaged F1 Score-,Macro%20Average,-Macro%20averaging%20is)
- Why I choose Accuracy and F1 score as metrics
- Why I choose F1 score over precision and recall and precision and recall tradeoff