SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning Homework 2

Alex Parri - 2140961 - Master's Degree in Cybersecurity

A.Y. 2024/25

Indice

Abstract

This is the report for the **second homework** of the Machine Learning course directed by Luca Iocchi for the Academic Year 24/25 for the Master's Degree in **Artificial Intellingence and Robotics** at Sapienza University of Rome.

In this homework, the goal was to learn a model that would learn how to drive a **2D top-view** car in a road with sharp left and right turns taken from gymnasium's car racing environment. All models and their variations have been tested with the same seed, thus on the same exact race track. Results **may vary** for different seeds.

More specifically it is an **image classification** problem whose objective was to learn the following

$$f : X = \{\underbrace{(96 \times 96 \times 3)}_{\text{colored images}}\} \rightarrow Y = \{0, 1, 2, 3, 4\}$$

The (supervised) dataset for the problem is composed of colored screenshots each of size (96×96) pixels and has been provided by the **professor itself**, already split between training and test samples as shown in the following table

Class label	Training samples	Test samples
do nothing	1000	133
steer left	1500	275
steer right	1500	406
gas	2000	1896
brake	369	39
TOTAL	6369 (69.9%)	2749 (30.1%)

There were no attempts at trying to balance the dataset due to the **low amount** of samples for each class, especially *brake*. Of the above training samples $\frac{1}{5}$ where used for validation, and multiple **random states** of these splits have been tested for variability.

Moreover, it is important to state that the dataset is somewhat **noisy** due to the presence of misclassified images, which is a recurring trait in real world image classification. Anyhow, it is a crucial factor to consider when evaluating the models' **accuracy**.

Two main approaches based on **Convolutional Neural Networks - CNNs** have been trained to try and learn the the problem:

- **First model** will prioritize feature elaboration
- **Second model** will prioritize feature extraction

Both abovementioned models are equipped with **Sparse Categorical Cross-Entropy - SCCE** loss function, one of the best to consider when we have classification problems with **integer encoded** labels as we in fact have

$$SCCE(x_i) = -\frac{1}{N}\sum_{i=1}^{N}\log \hat{P}(y_i)$$

Additionally, the output layer will always have the **softmax** function, which behaves very well for multiclass classification problems

$$\sigma(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

The results that will be shown are **not guaranteed** to be the best achievable ones, as the homework's main objective was to enchance the student's knowledge about image classification and CNNs.

Note: Generative AI has been utilized in this project solely as inspiration for the code. The project in its entirety has been developed **individually** without any other external collaborators, including this report.

The **hardware** utilized to train the models is Nvidia RTX 3070 GPU (8GB VRAM) with driver version 560.35.03 and CUDA version 12.6 on Ubuntu 24.04.01 LTS x86_64 and 16GB of RAM.

Preprocessing and data augmentation

Preprocessing is performed in machine learning to transform data in a **more suitable format** for model training, it has a relevant role in improving result quality and model efficiency: in this case, it has been applied to inputs given to both proposed models by **normalizing** the data contained in each pixel between **0 and 1 included**, both in the training and prediction phase.

Data augmentation is an important approach towards improving model generalization, thus **reducing overfitting**. In our problem the decision was to apply the following augmentation on all samples (values are compared to the original)

Augmentation	Minimum	Maximum
random gamma	90%	110%
random constrast	90%	110%
random quality	75%	95%

Not all kinds of data augmentation make sense depending on the specific problem: for this case in fact **flipping** images left-to-right would confuse the model as it would misclassify left with right and viceversa more often. In practice, that would translate to a significantly worse model accuracy.

Preprocessing and augmenting all the data takes around **16.7 seconds**.

First proposed model

The first model that is focused on feature elaboration more than extraction. It is composed of three convolutional layers with exponentially increasing units and two dense layers, which in its best form, after manual hyperparameter searching, seems to be the following

Layer	Units	Activation	Padding	Kernel size	Out Shape	Parameters
Conv2D	32	ReLU	valid	(2, 2)	(95, 95, 32)	416
MaxPooling2D	-	-	valid	(2, 2)	(47, 47, 32)	-
Conv2D	64	ReLU	valid	(2, 2)	(46, 46, 64)	8,256
MaxPooling2D	-	-	valid	(2, 2)	(23, 23, 64)	-
Conv2D	128	ReLU	valid	(2, 2)	(22, 22, 128)	32,896
MaxPooling2D	-	-	valid	(2, 2)	(11, 11, 128)	-
Flatten	-	-	-	-	15488	-
Dense	256	ReLU	-	-	256	3,965,184
Dropout 0.5	-	-	-	-	256	-
Dense	64	ReLU	-	-	64	16,448
Output	5	softmax	-	-	5	325

The total number of trainable parameters is **4,023,525**.

Despite the good hardware capabilities, performing automatic hyperparameter searching on the models themselves would take a **significant amount** of time as there are many possible configurations to try.

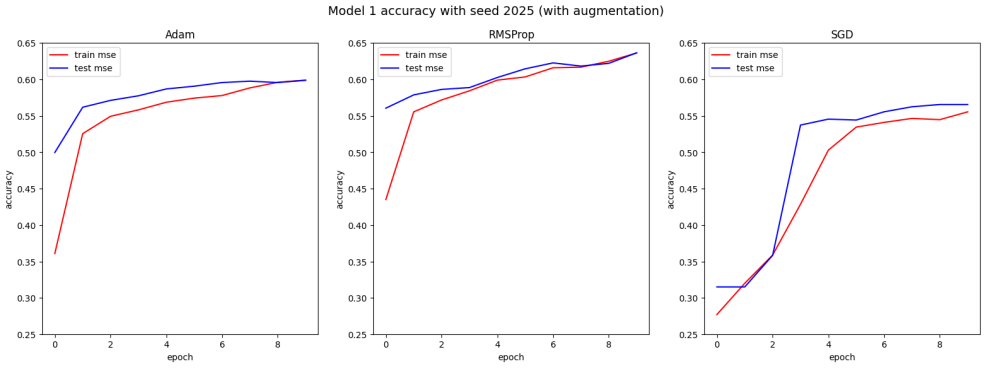
For practical reasons it was automatically performed only on the second model's **optimizers' hyper-parameters**.

Choosing the optimizer

The tested optimizers with manually searched optimal hyperparameters are the following

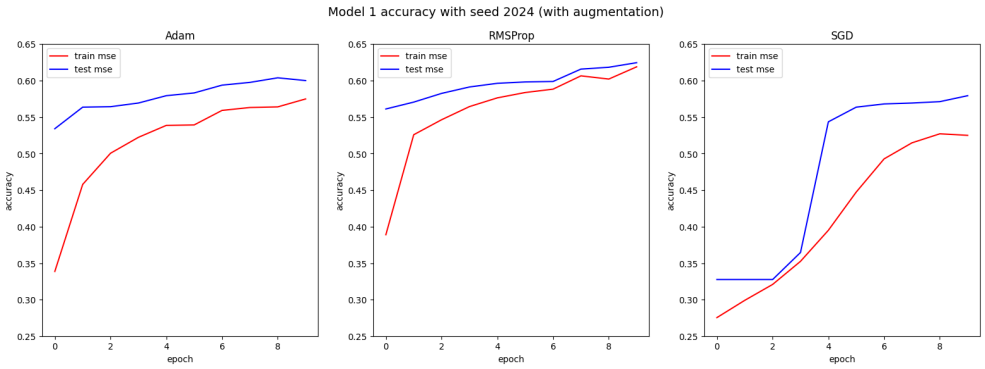
Optimizer	Hyperparameters	Training time	Batch size
Adam	$\eta = 0.001, \beta_1 = 0.6, \beta_2 = 0.8$	114.65s	64
RMSProp	$\eta = 0.0001, \mu = \rho = 0.9$	115.86s	64
SGD	$\eta = 0.001, \mu = 0.9$	105.63s	64

Training the three copies of first model with fixed weights with the above for only **10 epochs** reveals the following result

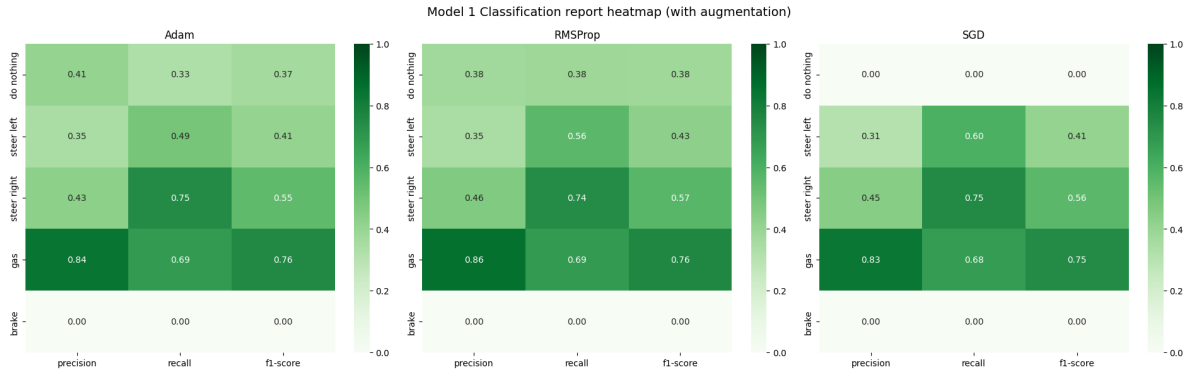


In terms of **validation set accuracy**, RMSProp seems to be the most performant with 63.65, with Adam slightly behind at 59.89 and SGD in last place with just 56.56. These seem like already good results, considering that predicting images at **random** would be just $\frac{1}{|classes|} = \frac{1}{5}$. These numbers are subject to change depending on the random weights with which the model is initialized with.

By testing a different validation split seed, the result is **the same** for all three optimizers, likely due to the small size of the dataset

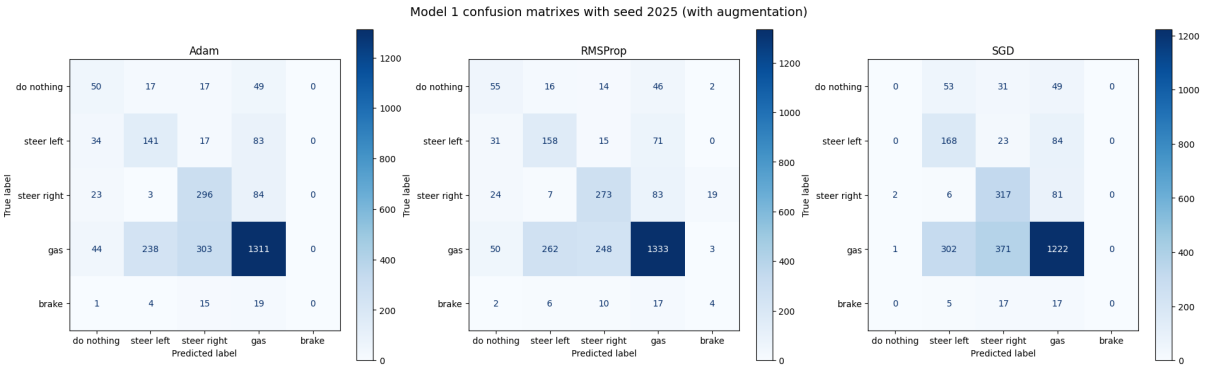


From the **classification report** below it is possible to see decent results across the board.



Adam and RMSProp present very similar results, with the *gas* class presenting most precision and f1-score due to its large amount of samples and the *brake* class with score zero.

All models are learning the problem **quite well**, as the following **confusion matrixes** are showing below



It is possible to notice the **same pattern** repeated throughout all optimizers

- *gas* is usually misclassified with *steer left* or *steer right*
- *brake* is practically never predicted

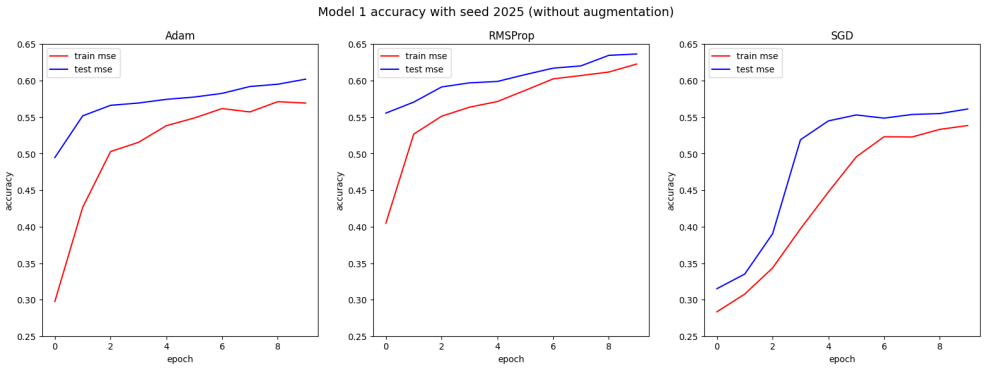
These are factors contributing to bringing the prediction accuracy down. However, if the end goal is just driving the car on the track and not **how gracefully** this driving is performed then it makes little difference: the above numbers **do not** tell the full story, in fact by utilizing the above on the simulation we obtain the following scores

Optimizer	Simulation Score	Remarks
Adam	729.91	Consistently and precisely on road
RMSProp	736.64	Offroads for a short period then saves itself

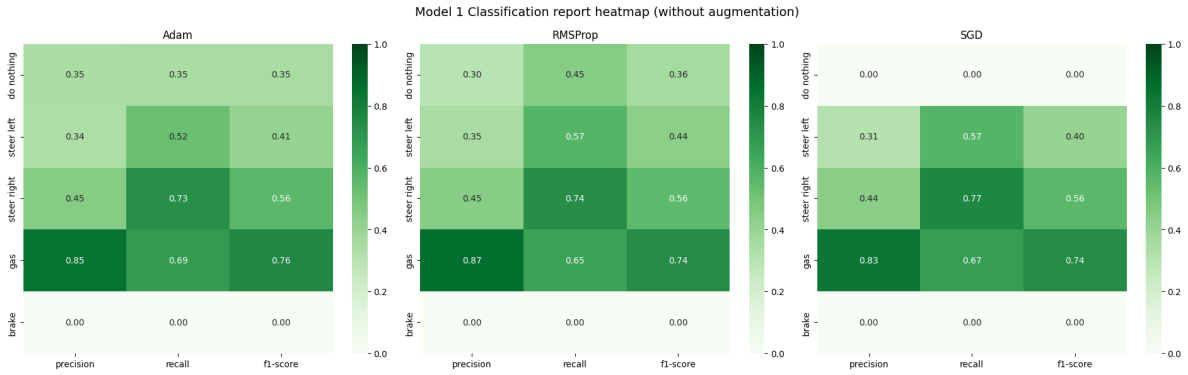
Optimizer	Simulation Score	Remarks
SGD	662.58	Turns on straight paths, drives too fast

Despite what the simulation scores say, Adam performs best when it comes to the **time spent** on the road and how many corners were cut.

If we **did not use** augmentation, kept the same hyperparameter values and split seed the validation accuracy values would pretty much not change, as shown below



Classification report also shows **practically identical numbers** as before, some parameters are better some are worse but the average is the same across the all optimizers.



This would **hint** at a simulation result that is exactly the same, however this is not the case at all as it is possible to see in the table below

Optimizer	Simulation Score	Remarks
Adam	303.55	Goes compltely offroad almost right away
RMSProp	659.21	Offroads for a short period then saves itself

Optimizer	Simulation Score	Remarks
SGD	780.42	Cuts many corners

It is rather curious to notice that SGD seems to perform best when no data augmentation was performed, but from a **stability point of view** it would be fair to say that Adam with augmentation performs best for the provided seed, although the simulation score isn't representative of that.

Second proposed model

This second model is instead focused on extracting more features, where an additional convolutional layer was added (for a total of four) and only one big dense layer. The result is a **latent space** that is half as big as it was before

Layer	Units	Activation	Padding	Kernel size	Out Shape	Parameters
Conv2D	32	tanh	valid	(2, 2)	(96, 96, 32)	416
MaxPooling2D	-	-	valid	(2, 2)	(48, 48, 32)	-
Conv2D	64	tanh	valid	(2, 2)	(48, 48, 64)	8,256
MaxPooling2D	-	-	valid	(2, 2)	(24, 24, 64)	-
Conv2D	128	tanh	valid	(2, 2)	(24, 24, 128)	32,896
MaxPooling2D	-	-	valid	(2, 2)	(12, 12, 128)	-
Conv2D	256	tanh	valid	(2, 2)	(12, 12, 256)	131,328
MaxPooling2D	-	-	valid	(2, 2)	(6, 6, 256)	-
Flatten	-	-	-	-	9216	-
Dense	512	tanh	-	-	512	4,719,104
Dropout 0.7	-	-	-	-	512	-
Output	5	softmax	-	-	5	2,565

Whose total number of trainable parameters amounts to **4,894,565**.

Choosing the optimizer

The same optimizers were tested, just with manually tuned hyperparameters

Conclusions

Image classification is **difficult**, in fact higher validation accuracy does not necessarily correspond to an overall better model. It mostly depends on the **context** and on the specific definition of “accuracy” when it comes to using the model to predict.

In this specific context it could be:

- driving stability
- driving speed
- simulation score
- …
- a mix of the above

The problem’s difficulty was additionally **exacerbated** by noise and unbalance of the dataset, considering the low amount of samples it is not that trivial to obtain a high accuracy.

In conclusion, both homeworks were very useful and insightful for practicing machine learning concepts hands-on. With machine learning there is so much power at our (literal) fingertips, and it is only going to get better in the future. Exciting times ahead.