SAPIENZA
UNIVERSITÀ DI ROMA

Report

Machine Learning Homework 2

Professor:
Luca Iocchi

Student:
De Sio Ilaria - 2064970

Academic Year 2023/2024

Contents

1 Introduction

This is my report for the second Machine Learning Homework, about an image classification problem to learn the behaviour of a racing car in a Gym environment.

1.1 Assigned Task

Dataset is divided into training and test sets, and contains a top-down 96x96 RGB image of the car and race track, labelled with one out of the 5 actions available for the control of the car.

Action Space in the CarRacing-v2 environment of OpenAI's Gym framework refers to the set of possible actions that an agent can take at any given step within the environment. In this case is discrete and there are 5 actions:

0: Do nothing; **1:** Steer left; **2:** Steer right; **3:** Gas; **4:** Brake;

2 Data Preprocessing

In the initial phase of the project, the focus is on the critical step of data preprocessing to ensure the reliability and quality of the results. This process involved initially the setting of the batch size to 128 for both TR (`trainingset`) and TS (`testset`).

2.1 Data Augmentation

It involves the adoption of techniques to artificially increase the variety of the training dataset through various transformations that modify the original images without altering their meaning or class. Given these premises, I decided to use it for the following reasons in the following table.

Table 1: Advantages of using Data Augmentation

Advantages	**Description**
Increased Dataset Variety	It helps to create a larger and more diverse training dataset without the need to collect new images.
Reduction of Overfitting	Allows the model to be exposed to a wider range of object variations, reducing the risk of it overfitting to the specifics of the training set.
Improved Generalization	Transformed images simulate different capture conditions, helping the model to better generalize to new images during the testing phase.

The following transformations were applied to the Training set images:

```
train_datagen = ImageDataGenerator(
    rescale = 1. / 255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.1,
```

```
    horizontal_flip=False,
    fill_mode='nearest',
    validation_split=0.2
)
```

- **Rescaling:** Each pixel of the images was rescaled by dividing by 255, normalizing the values in the range [0,1], which helps in faster convergence during neural network training.
- **Rotation:** This parameter specifies the range (in degrees) within which the images can be randomly rotated during training. In this case is setted to 10°.
- **Horizontal and Vertical Translations:** I applied random translations up to 10 percent horizontally and vertically. This simulates variations in object position in images, making the model less sensitive to the exact position of objects.
- **Shear Range:** Random image cuts by a value of 10% simulate slight distortion, helping the model recognize objects under different conditions.
- **Zoom Range:** Random 10% zoom allows the model to learn to recognize objects at different scales.
- **Fill Mode:** I chose 'nearest' as the fill method for the new pixels created during the transformation, maintaining the visual consistency of the images.
- **Validation Split:** A value of 0.2 means that 20% of the dataset will be used for validation and not be used in training.

The choice not to use horizontal flipping was dictated by the specific nature of the images, since when testing the environment the direction of the car turns was also wrong as a 'mirror' effect was created.

2.2 Data Exploration

Regarding the dataset, the size of the training set contains 5096 samples (20% of which are dedicated to validation and the rest for training), and 2749 samples for the test set.

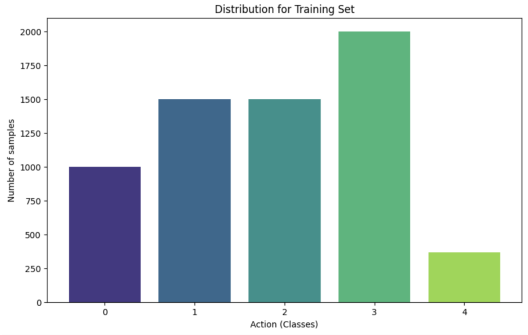


Figure 1: Distribution of Training Set

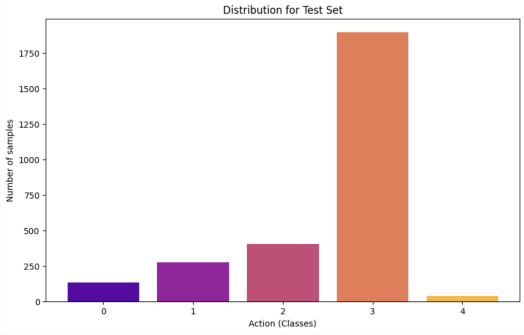


Figure 2: Distribution of Test Set

From the graphs of class distributions in both the training and test datasets, it is clear that class 3 is significantly overrepresented with over 1750 samples. In contrast, the other classes, particularly class 4, have much fewer samples. To prevent the model from overfitting due to this uneven distribution, the assignment of weights to classes and training parameters will be adjusted. There will also be

performance evaluation strategies that compensate for the uneven representation of classes. These steps are essential to ensure good generalization of the model.

3 Approaches

In this section, I discuss in detail the two distinct approaches I used to address the problem at hand, each characterized by a unique set of layers and parameters. I will present a side-by-side tabular overview to clearly highlight the structural components and configurations of both architectures. This comparative visualization is intended to provide a clear and structured representation of the differences and specifics of each model, facilitating an understanding of how each is built and designed to function.

Layer	CNN1	CNN2
Input Shape	96×96×3	96×96×3
Convolutional #1	16 filters, 3×3, relu	64 filters, 3×3, relu
Regularization #1	L2 (if specified)	L2 (if specified)
Pooling #1	2×2 max pooling	3×3 max pooling
Convolutional #2	32 filters, 3×3, relu	128 filters, 3×3, relu
Regularization #2	L2 (if specified)	L2 (if specified), Batch Normalization
Pooling #2	2×2 max pooling	3×3 max pooling
(Additional Layers)	-	(If depth > 2, more conv layers follow the pattern)
Dropout	Dropout (0.4)	-
Flatten	Flatten	Flatten
Dense #1	-	48 neurons, relu
Output	num_classes neurons, softmax	num_classes neurons, softmax
Optimizer	Adam (default)	SGD (default), lr=0.001

Figure 3: CNN architectures

The key differences lie in the base width of the filters, the presence of batch normalization in CNN2, the max pooling size, the regularization strategies, and the choice of optimizers. CNN1 may train faster with the Adam optimizer and is likely more suitable for scenarios where computational resources are limited, given its simpler architecture. CNN2, with batch normalization and SGD, might train slower but could potentially offer better generalization due to more sophisticated regularization techniques.

3.1 First model

The CNN1 model is a fairly simple convolutional neural network designed for efficient processing. It starts with a basic width of 16 filters in the first convolutional layer, which allows the model to focus on capturing fine-grained patterns before expanding to more complex features. The use of a rectified linear activation function (ReLU) facilitates training time and helps mitigate the vanishing gradient problem.

A key feature of this architecture is the doubling of filters at each successive convolutional layer, which allows the network to progressively extract more abstract representations of the input data.

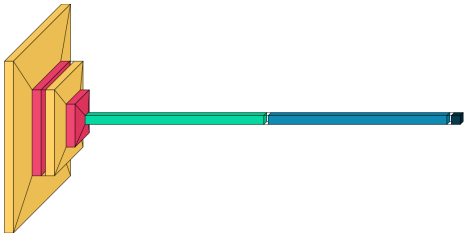


Figure 4: CNN1 architecture created with visualkeras

Regularization plays a crucial role in CNN1, with an optional L2 regularizer penalizing large weights, thus promoting a simpler and potentially more generalizable model. The inclusion of a dropout layer with a rate of 0.4 is particularly effective and was chosen to prevent overfitting.

The architecture concludes with a fully connected layer that employs a softmax activation function to produce probabilistic class predictions, making it suitable for multiclass classification problems. By default, CNN1 uses the Adam optimizer, known for its adaptive learning capabilities, which often leads to faster convergence.

3.2 Second model

In contrast, CNN2 is a more complex architecture that starts with a higher base width of 64 filters, enabling the model to capture a broader range of features from the initial layer. Each convolutional layer in CNN2 is followed by batch normalization, which standardizes the inputs to each layer, thereby stabilizing the learning process and often allowing for the use of higher learning rates.

CNN2's use of max pooling with a 3x3 window, as opposed to CNN1's 2x2, reduces the spatial dimensions of the feature maps more aggressively, which can be beneficial for capturing the most salient features while reducing computational load. The absence of a dropout layer is compensated by the use of L2 regularization (where specified) and batch normalization, which can provide a form of regularization through noise introduction during the training phase.

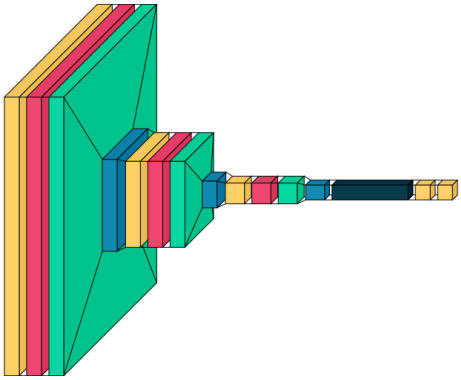


Figure 5: CNN2 architecture created with visualkeras

The model features a fully connected dense layer with 48 neurons before the final classification layer, adding an additional level of abstraction for the features learned by the convolutional layers. The default SGD optimizer, with a chosen learning rate of 0.001, that ensures a steady and controlled update of the weights, which can be advantageous for achieving a well-generalized model.

4 Training

In the development process, I employed a methodology of experimentation and refinement to accurately define the parameters of two distinct CNN architectures. Before arriving at an optimal configuration, a series of preliminary experiments were conducted. This approach allowed the development of two architectures that, while sharing a common goal, were refined in different specific manners to better suit different challenges within the same problem context to be solved.

For both architectures, training was done in two distinct phases: initially, each network was trained for 50 epochs, followed by an extension to 100 epochs. This stepwise progression in training facilitated a detailed analysis of the evolution of the performance of each architecture, allowing to evaluate the effectiveness of parameter changes made during the experimental phase.

4.1 Training Time Analysis

Configuration for the tests All tests, and all graphs obtained within this project report were performed on a MacBook Pro M1 machine (so times are rightly affected), with the following configuration:

- RAM: 8 GB unified memory;
- CPU: Apple M1 chip with an 8-core CPU;
- GPU: Integrated 8-core GPU;
- Neural Engine: 16-core NE;

It's also important to note that the machine's performance can be influenced by factors beyond the hardware specifications, such as system load during testing, the specific software versions used for machine learning libraries, and background processes running on the machine.

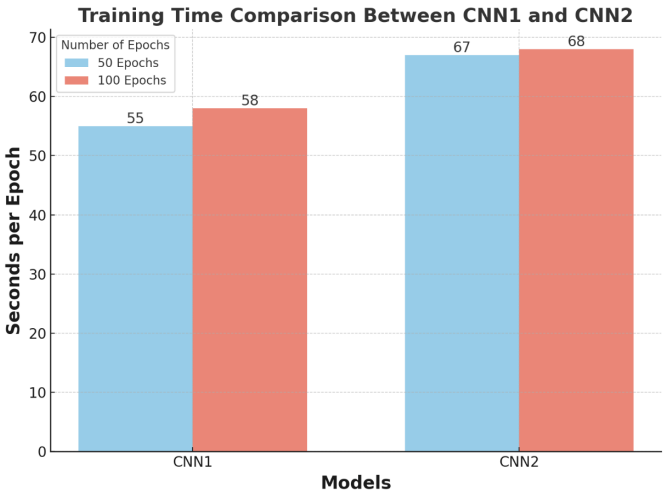


Figure 6: Time Comparison

5 Evaluation

5.1 Metrics chosen

The metrics chosen for evaluation are :

- **Cross-Entropy Loss Function:** It measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label and the main goal is generally to get the model as close to 0 as possible. For multi-class classification problems, the cross-entropy loss is a generalization of the binary case:

$$\text{Cross-Entropy Loss} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

  M Number of classes (in this case, the 5 actions).

  log The natural logarithm.

  y Binary indicator (0 or 1) if class label c is the correct classification for observation o.

  p Predicted probability that observation o is of class c.

- The **accuracy** measures the percentage of correct predictions made by the model out of all predictions made. It is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2}$$

- The **precision** is the proportion of positive identifications that were actually correct, relative to the total number of positive identifications made. It is particularly useful when the cost of false positives is high.

- The **recall**, or sensitivity, measures the proportion of actual positives that were identified correctly. It is important when it is crucial to detect all positive instances.

- The **F1-score** is the harmonic mean of precision and recall and provides a single score that balances both metrics. It is defined as:

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- The **support** refers to the number of occurrences of each class in the true dataset. It is useful for assessing the model's performance on classes with an unequal number of examples.

5.2 Training with 50 epochs

CNN1 shows a rapid reduction in loss and increase in accuracy in the early epochs, stabilizing thereafter, suggesting effective learning and minimal overfitting. The accuracy remains constant over the epochs with a slight discrepancy between training and testing, indicative of good generalization of the model.

Turning to the CNN2 model, a different behavior is observed. The loss on the training set decreases very rapidly, indicating a strong fit to the training data, while on the test set the loss shows a tendency to stabilize earlier, suggesting a lower ability to generalize than CNN1 perhaps given the few epochs.

A key observation is that both models show high average accuracy. In particular, the CNN1 model shows remarkable learning ability even with only 50 epochs, maintaining satisfactory performance. In contrast, CNN2, while achieving a similar level of accuracy as CNN1, encounters difficulties in classifying minority classes. It is also relevant to note that, for both models, the most complex class to identify turns out to be class 4.

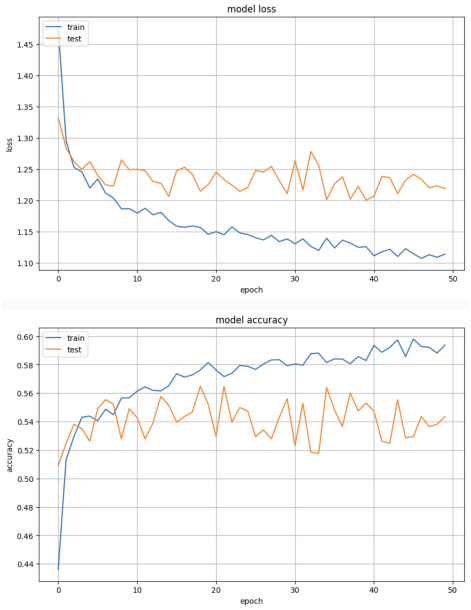


Figure 7: CNN1 Loss and Accuracy

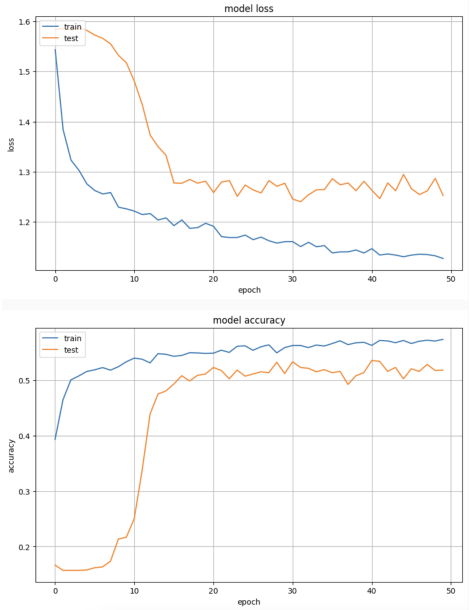


Figure 8: CNN2 Loss and Accuracy

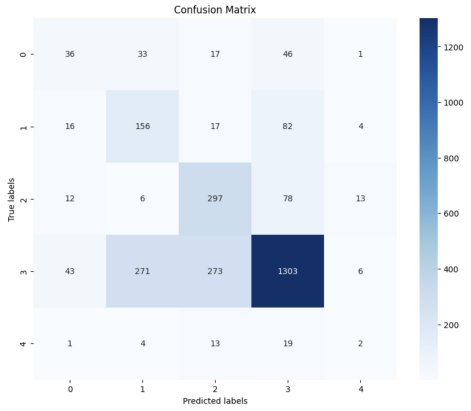
Figure 9: Performances on 50 epochs

	precision	recall	f1-score	support
0	0.333	0.271	0.299	133
1	0.332	0.567	0.419	275
2	0.481	0.732	0.581	406
3	0.853	0.687	0.761	1896
4	0.077	0.051	0.062	39
accuracy			0.653	2749
macro avg	0.415	0.462	0.424	2749
weighted avg	0.710	0.653	0.668	2749

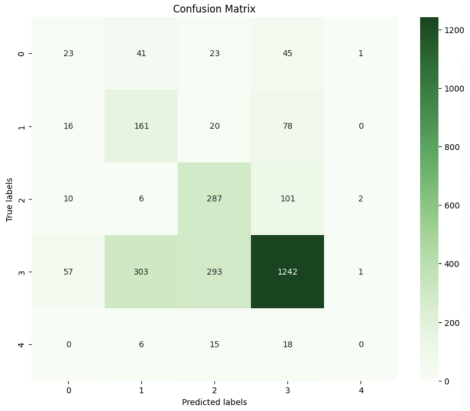
(a) CNN1 Classification Report

	precision	recall	f1-score	support
0	0.217	0.173	0.192	133
1	0.311	0.585	0.407	275
2	0.450	0.707	0.550	406
3	0.837	0.655	0.735	1896
4	0.000	0.000	0.000	39
accuracy			0.623	2749
macro avg	0.363	0.424	0.377	2749
weighted avg	0.685	0.623	0.638	2749

(b) CNN2 Classification Report



(c) CNN1 Confusion Matrix



(d) CNN2 Confusion Matrix

Figure 10: Classification rep. and Confusion Matrix on 50 epochs

5.3 Training with 100 epochs

Examining the 100-epoch training performance of the models, it can be observed that CNN1 has a propensity for overfitting, evident by the substantial gap between the training and test loss curves, accompanied by alternating peaks and troughs. This pattern suggests that CNN1, while effectively

learning the training data, struggles to maintain this performance on the unseen data, and after several experiments was found to be the absolute best performer, assuming a small dataset is the problem.

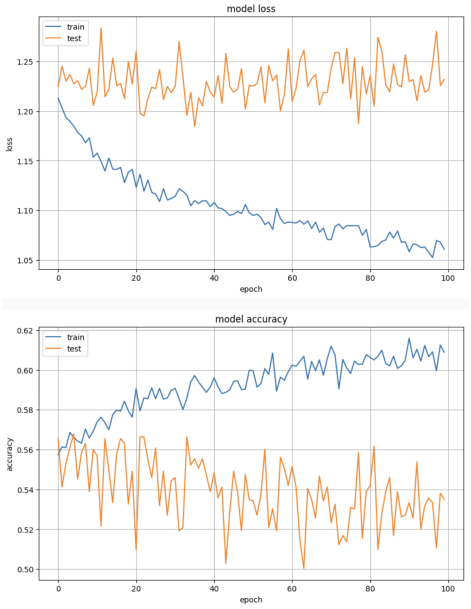


Figure 11: CNN1 Loss and Accuracy

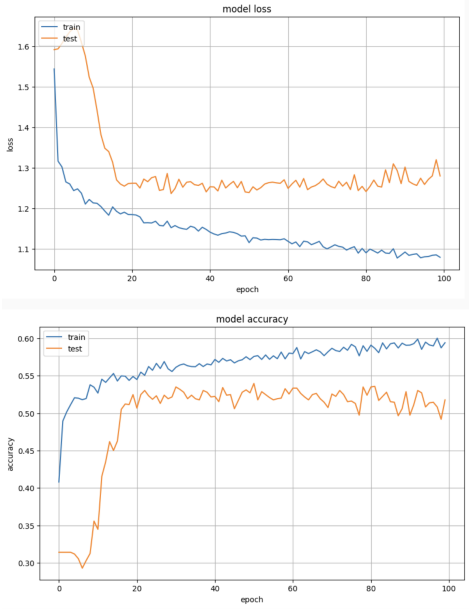


Figure 12: CNN2 Loss and Accuracy

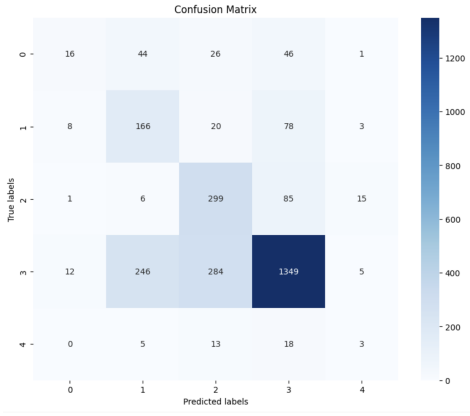
Figure 13: Performances on 100 epochs

	precision	recall	f1-score	support
0	0.432	0.120	0.188	133
1	0.355	0.604	0.447	275
2	0.466	0.736	0.571	406
3	0.856	0.711	0.777	1896
4	0.111	0.077	0.091	39
accuracy			0.667	2749
macro avg	0.444	0.450	0.415	2749
weighted avg	0.717	0.667	0.675	2749

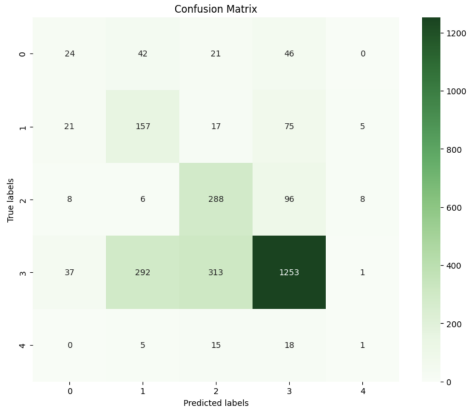
(a) CNN1 Classification Report

	precision	recall	f1-score	support
0	0.267	0.180	0.215	133
1	0.313	0.571	0.404	275
2	0.440	0.709	0.543	406
3	0.842	0.661	0.741	1896
4	0.067	0.026	0.037	39
accuracy			0.627	2749
macro avg	0.386	0.429	0.388	2749
weighted avg	0.691	0.627	0.642	2749

(b) CNN2 Classification Report



(c) CNN1 Confusion Matrix



(d) CNN2 Confusion Matrix

Figure 14: Classification rep. and Confusion Matrix on 100 epochs

In contrast, CNN2 demonstrates remarkable recovery from initial training difficulties, showing performance graphs that indicate excellent data generalization. CNN2's loss curves are closely aligned between training and testing, reflecting a more balanced and generalizable learning process. The ability of the model to perform consistently on both data sets implies the robustness of its architecture with the parameters set.

6 Racing scores

The table presents the rewards, obtained from two convolutional neural network models (CNN1 and CNN2) with different seed initializations (2000, 3000 and 4000) after running 50 and 100 training epochs in the context of running CarRacing-v2.

Seed	CNN1_50ep	CNN1_100ep	CNN2_50ep	CNN2_100ep
2000	752	771	755	820
3000	745	801	812	816
4000	753	809	812	816

Figure 15: Car Racing Scores

In general, all models show a tendency to improve with increasing training epochs. This may be due to finer optimization of the weights, allowing the model to better learn the characteristics of the environment. The CNN2 model appears to be more consistent in achieving high scores regardless of the initial seed, suggesting that it may have a more robust architecture or optimal parameters for this particular task.

7 Conclusion and other considerations

In conclusion, analysis of the results obtained from the models clearly shows that both models have excellent learning ability and adapt effectively to the game environment. As the training epochs increase with different parameter settings, significant improvements in performance can be seen, suggesting that the neural networks have benefited from the additional experience, refining their ability to make decisions and maximize total reward.

Looking ahead, the implementation of other reinforcement learning techniques, such as Q-learning, could further increase the capacity of these models. Q-learning, in particular, with its value-based approach, could provide a more direct and fine-grained mechanism for estimating future rewards, leading to even more optimal decisions and acceleration of the learning process.