

Course Name: Intro. To Artificial Intelligence

CSC 4301 - 02

**Project4**

Student’s Name:Othman Bouchentouf

Omar Bouchentouf

Supervised by: Dr Tajjeeddine RACHIDI

Submission Date: 12/7/2023

Sommaire

[I. Introduction: 3](#_Toc152611438)

[II. Case 1 and 2: 3](#_Toc152611439)

[III. Case 3: 4](#_Toc152611440)

[IV. Case 4: 5](#_Toc152611441)

[1. Trying a different hyperparameter: 5](#_Toc152611442)

[2. Trying different filters in the convolutional layers: 7](#_Toc152611443)

[V. Youtube link: 8](#_Toc152611444)

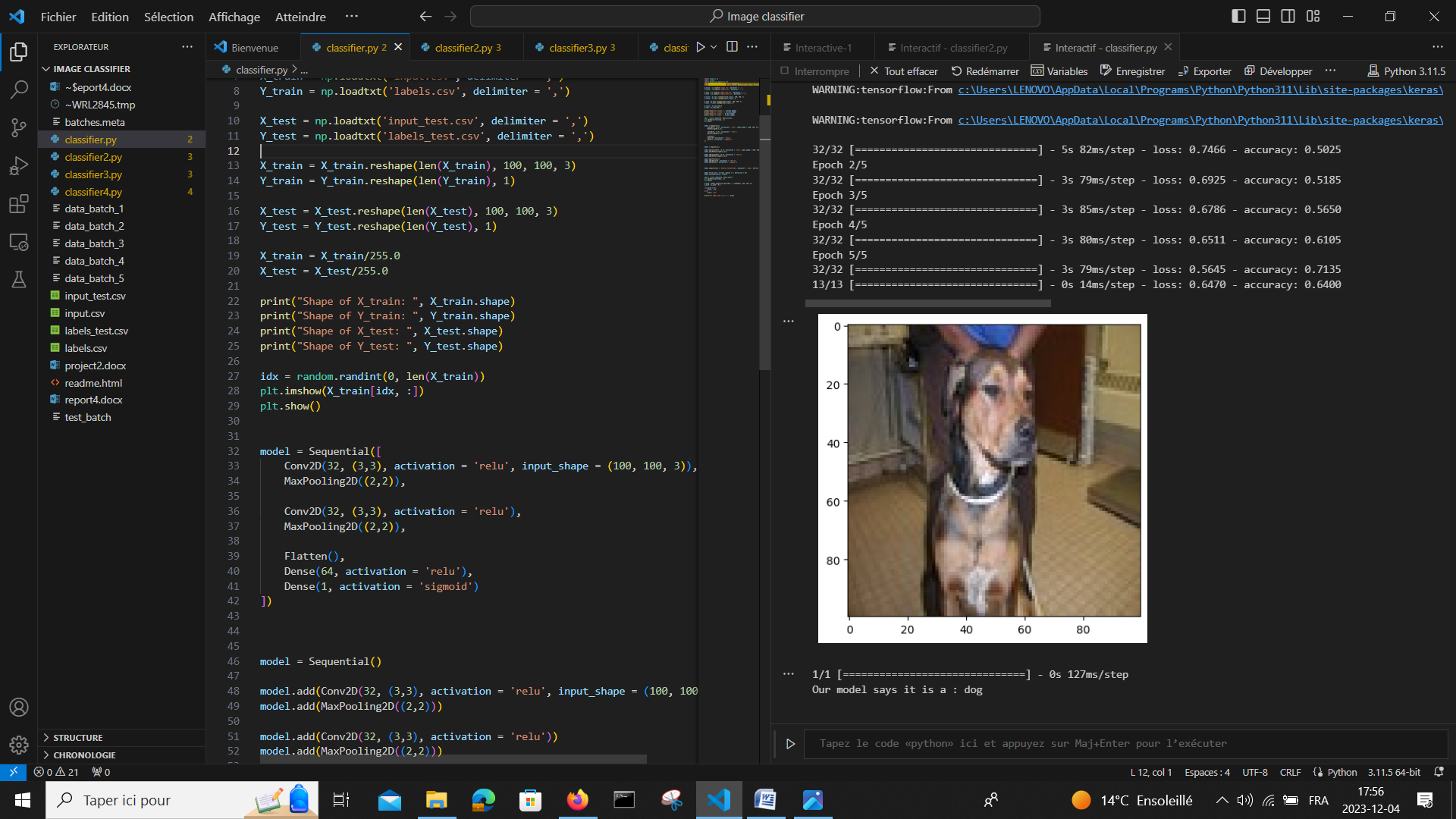
[VI. Conclusion: 8](#_Toc152611445)

# Introduction:

The pursuit of optimal performance in machine learning models is often a nuanced balance of architecture selection and hyperparameter tuning. This report delves into the latter, examining how different learning rates and batch sizes affect the training of a Convolutional Neural Network (CNN) applied to the CIFAR-10 dataset. The CIFAR-10 dataset, with its ten classes of 32x32 pixel images, provides a standard testbed for assessing model accuracy and generalization. We document the process of training the CNN with varying hyperparameters, analyze the impact on the loss and accuracy, and interpret the results to understand the model's convergence behavior. The findings aim to guide the selection of hyperparameters that facilitate effective learning and provide a framework for similar optimization challenges in machine learning tasks.

# Case 1 and 2:

The provided video shows how we can use deep learning using Tensoflow and Keras for image classification:

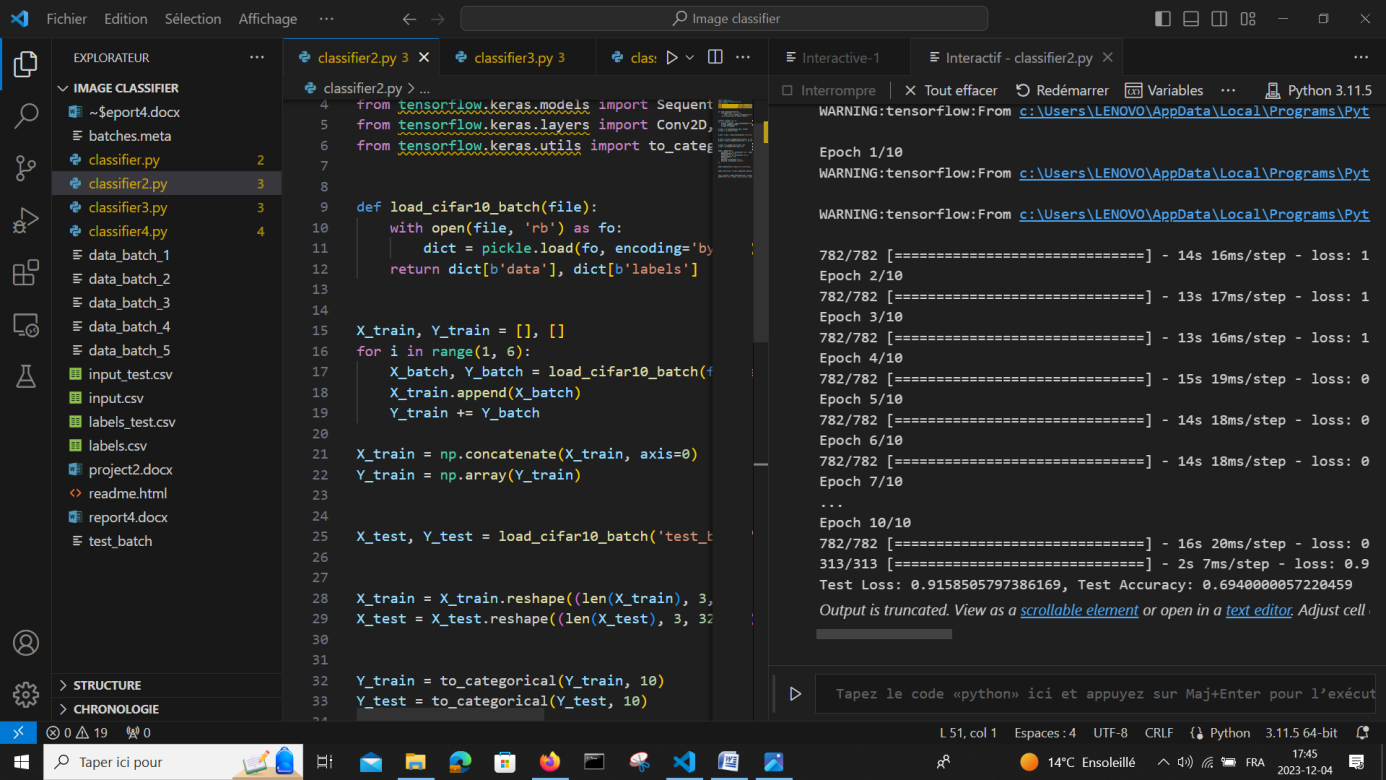


**Result:**

We can clearly see that the model is guessing correctly what animal is in the image (dog or cat) with an accuracy of 0.64

# Case 3:

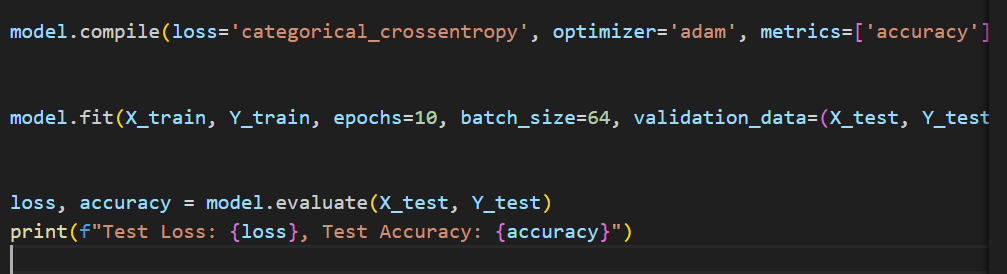
After loading all the files and training the model, we got this:



Test loss = 0.91 and test accuracy = 0.69

**Reflection:**

We used this to calculate the loss and accuracy:



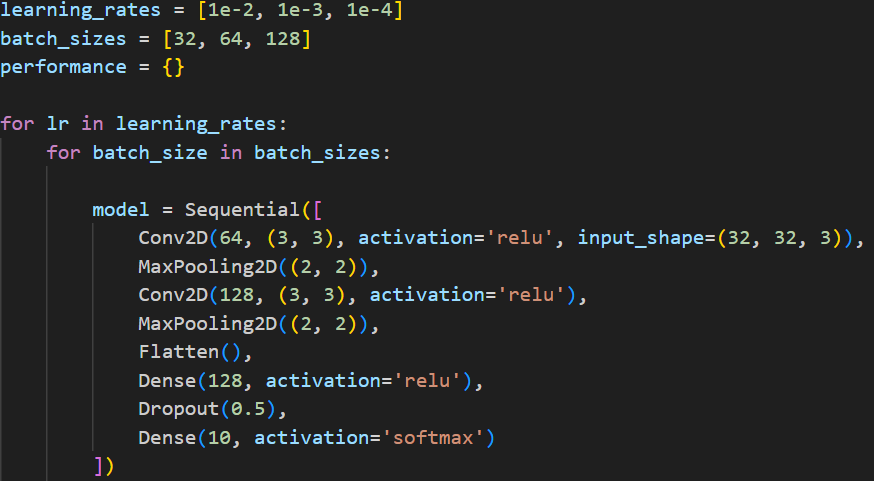
The result we got means that our model predicted correctly 70% of the images in the test and the loss of 0.88 means that there is still a significant error between the predicted values and the actual values.

To improve this model, we can follow some strategies ( as we will see in case 4) for example changing the hyperparameters like adjusting the learning rate, batch size, number of epochs, or the architecture of the neural network (like adding more layers or changing the number of neurons) can significantly impact performance.

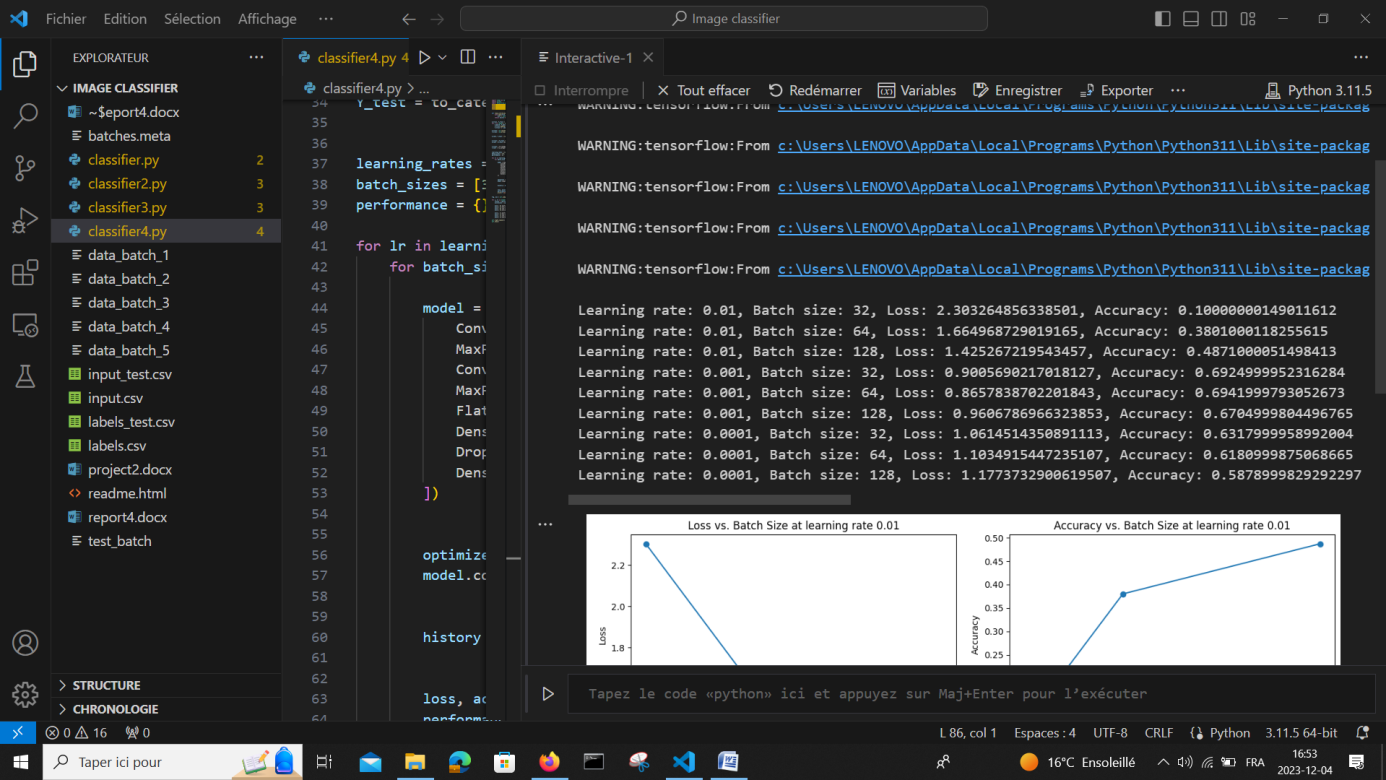
# Case 4:

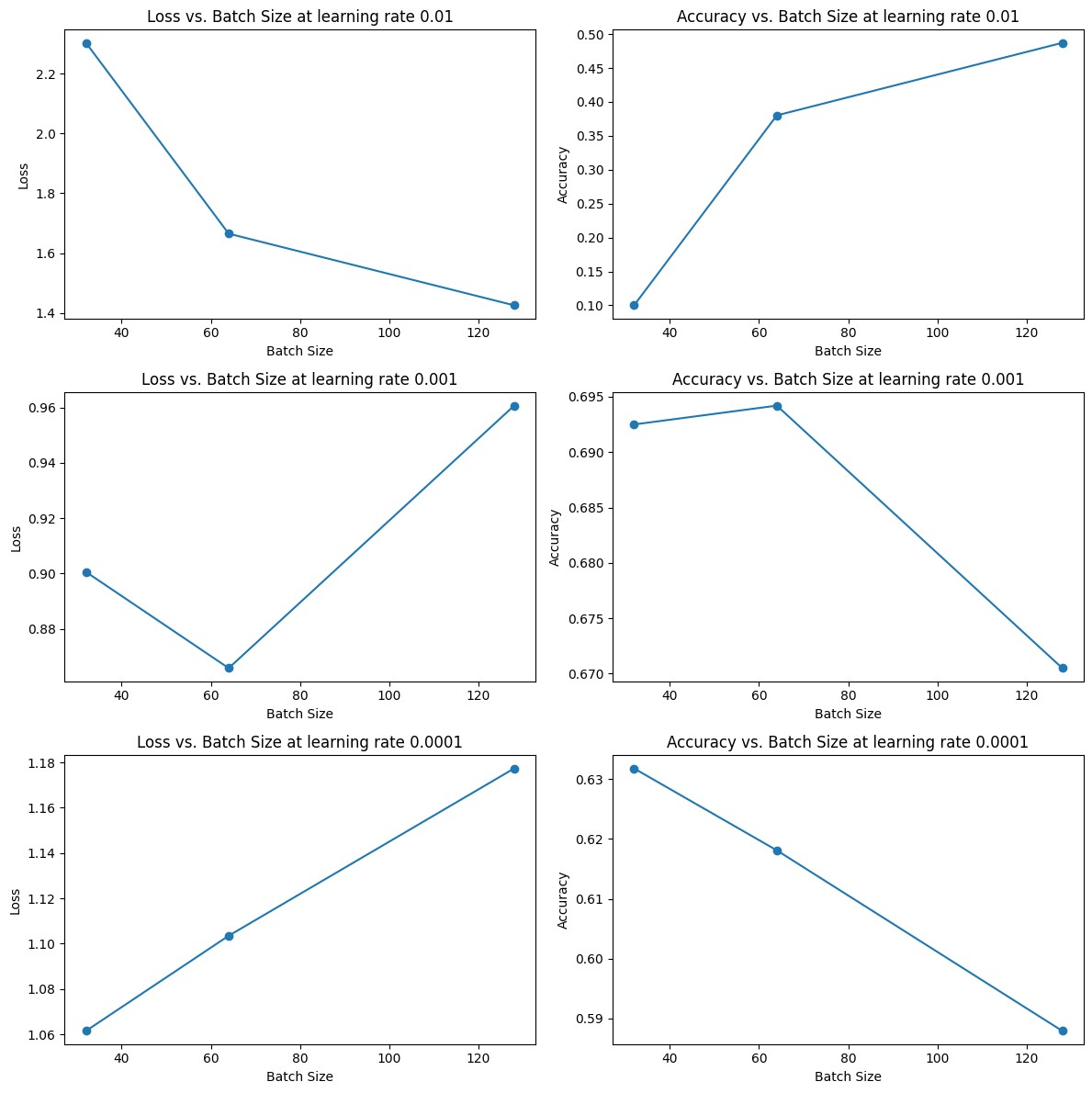
### Trying a different hyperparameter:

Here we tried different hyperparameters (different learning rates and batch sizes)



**The output:**



**Graphs:**  **Reflection:**

For the learning rate 0.01, we can see a clear trend where increasing the batch size from 32 to 128 improves both loss and accuracy. The loss decreases from 2.30 to 1.42, and the accuracy increases from 10% to 48.71%. However, an accuracy of around 48.71% suggests that the model might still be learning, but this learning rate could be too high, causing the model to not converge as well as it could.

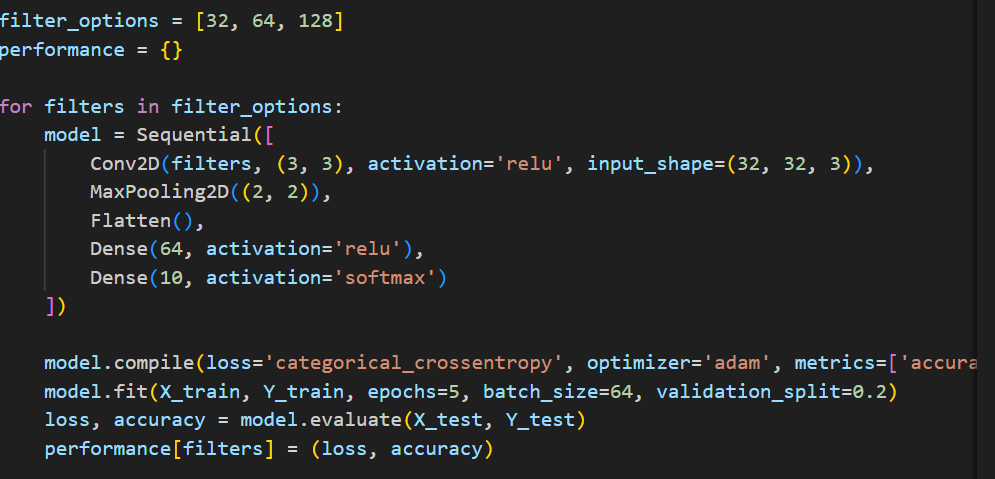
For the learning rate 0.001 it shows the best performance among the ones you tested. The model achieves the highest accuracy of 69.42% with a batch size of 64. This suggests that a learning rate of 0.001 is more suitable for this problem and model architecture. However, increasing the batch size to 128 shows a decrease in performance. This could be due to the model having fewer updates per epoch.

For the learning rate 0.0001, the performance here is worse than for 0.001, with the highest accuracy being 63.18% for a batch size of 32. The loss also increases as the batch size increases, which could indicate that the learning rate is too low, causing the model to learn too slowly and possibly get stuck in a suboptimal convergence.

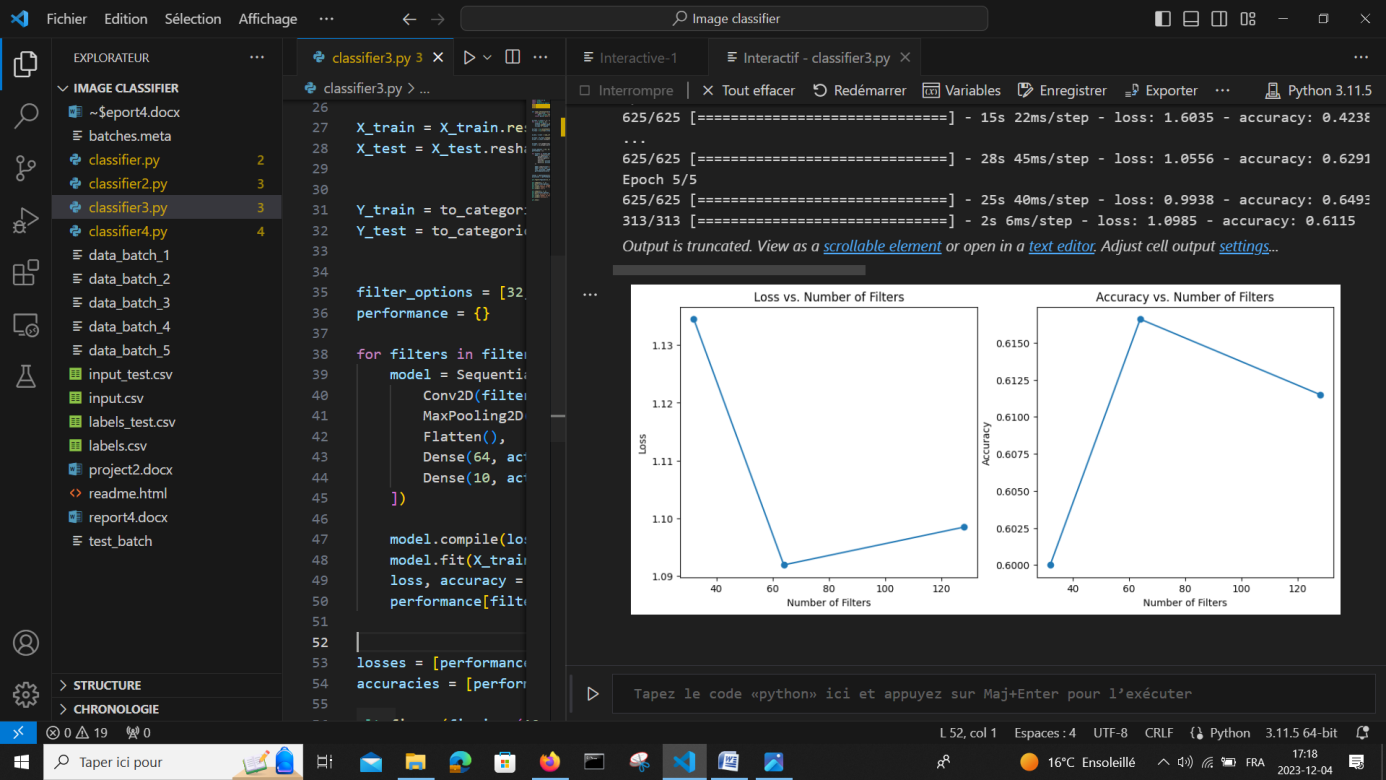
From these results we can say that a learning rate too high can prevent the model from finding an optimized solution and having it too low cause the model to converge too slowly. Also the **batch size** can affect the granularity of the model updates. Smaller batch sizes offer more frequent updates and can lead to better generalization but might also cause more noise in the updates. Larger batch sizes provide smoother updates but may miss out on some nuances in the data.

### Trying different filters in the convolutional layers:

Here we varied the number of filters:



**The output:**



Accuracy = 0.61

**Reflection:**

We can see that the loss is lowest with a number of filters (around 64), suggesting that with this particular architecture and dataset, 64 filters may be striking a good balance between model complexity and its ability to generalize. But, the loss increases when the number of filters is increased to 128. This could indicate overfitting. The same for accuracy, it peaks with a number of filters (again around 64), which aligns with the lowest loss observed and there is a decrease if the number of filters is more up to 128.

From these results, we can infer that increasing the model's complexity does not always lead to better performance and can actually degrade performance if the model becomes too complex.

# Youtube link:

https://www.youtube.com/watch?v=mhvqfcguLCw

# Conclusion:

The experimental exploration of hyperparameter tuning on the CIFAR-10 dataset has yielded insightful conclusions about the behavior of CNN models under varying conditions. Our investigation into learning rates and batch sizes has demonstrated that these parameters significantly influence model accuracy and convergence. A learning rate of 0.001 with a batch size of 64 emerged as the most effective combination, striking a balance between efficient learning and model stability. Notably, while higher learning rates led to faster but less stable convergence, lower rates resulted in slower learning that did not necessarily culminate in better performance.

These results underscore the importance of hyperparameter optimization in the model training process. They suggest that careful, empirical hyperparameter selection can lead to substantial improvements in model performance without the need for more complex architectures or additional training data. Future work may expand upon this foundation by exploring a broader range of hyperparameters, implementing learning rate schedulers, and employing regularization techniques to further enhance model generalization and performance.