Master in
Computer Vision
Barcelona

Image Classification

Module: C5

Group: 7

Students: Cristian Gutiérrez
Iñaki Lacunza
Marco Cordón
Merlès Subirà

Index

1. Used Frameworks
2. C3 Model
3. PyTorch Lightning
4. PyTorch Lightning vs PyTorch
5. Final remarks
6. Summary Slide

Used Frameworks

First of all we had to make a change in the content of this week's task because in C3 we had some problems with the GPU and the solution that we found was using PyTorch instead of Keras.

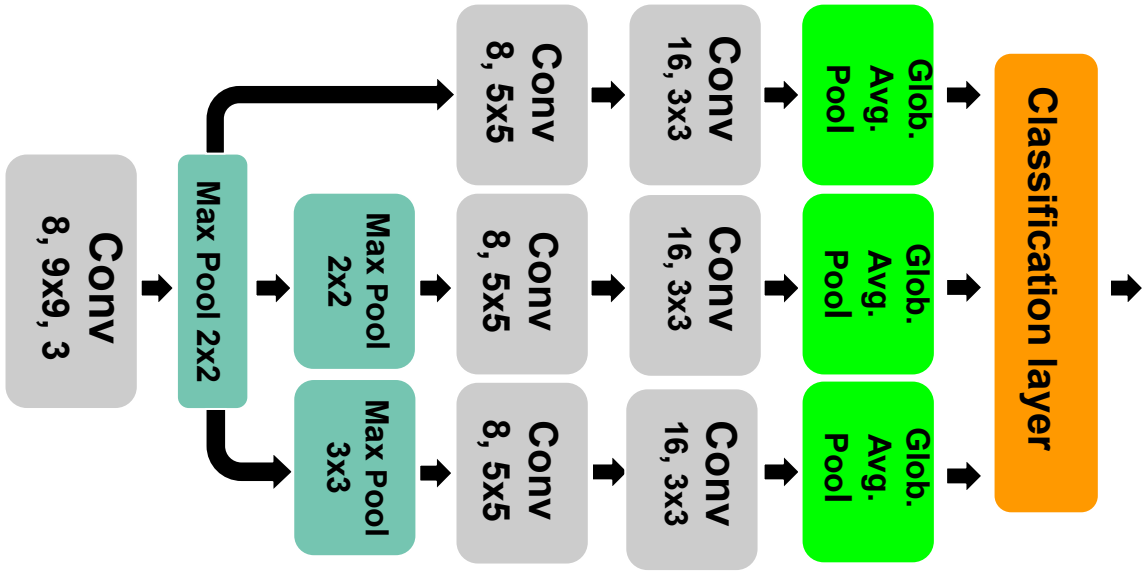
After asking the teacher about this, he invited us to compare **PyTorch vs PyTorch Lightning.**

PyTorch Lightning was created to simplify the training and deployment of models by providing a lightweight and a standardized interface. It is useful for the academic students since they can experiment with deep learning and machine learning models by avoiding the repetitive tasks and creating a more structured and organized approach to development.

VS

C3 Model: Architecture

Just 10k parameters and 79% of accuracy!!



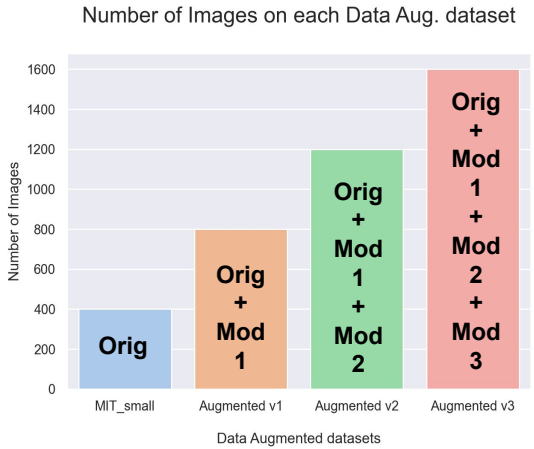
The architecture that we created in the last week of C3. It is based in the Inception block of GoogLe Net, analyzing different scales at once.

We employed 3 different paths with different pooling layers to analyse on different scales simultaneously and we combined the results at the end of the block.

C3 Model: Training Data

Since we only had 400 images we applied a data augmentation to create 3 new datasets of new 400 images and combined them to obtain bigger datasets in order to obtain better results.

3 new datasets were created:



Original image

Mod. 1:

- Rotation: 15º
- Desp x : 0.1
- Desp y : 0.1
- Zoom : 0.9
- H flip : True
- Brightness = 0.9

Mod. 2:

- Rotation: -15º
- Desp x : 0.1
- Desp y : 0.1
- Zoom : 0.95
- H flip : True
- Brightness = 0.9

Mod. 3:

- Rotation: 25º
- Desp x : -0.1
- Desp y : -0.1
- Zoom : 0.95
- H flip : False
- Brightness = 1.1

C3 Model: Selecting an Optimizer

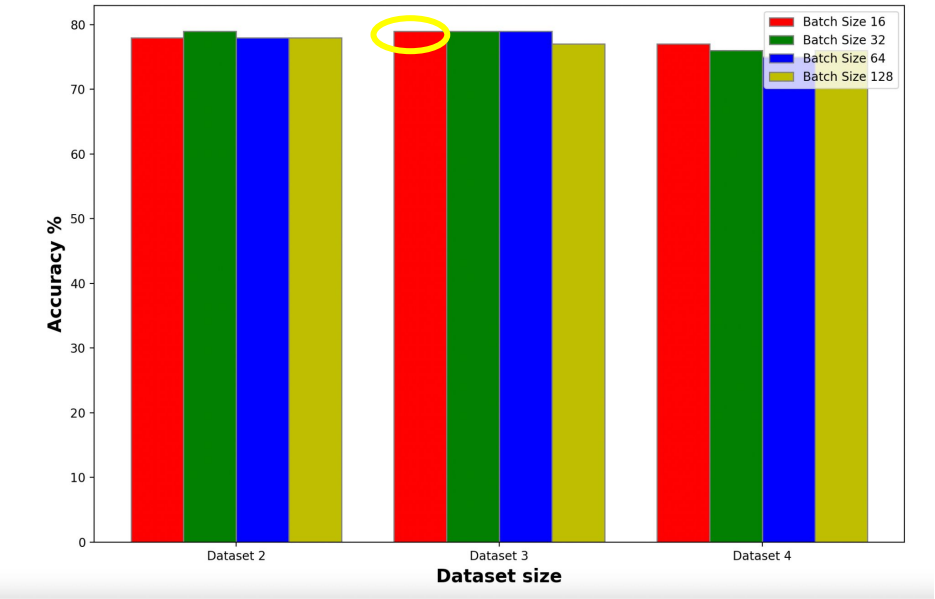
We tried different optimizers to know which one works better.

As we can see all the adaptive learning methods works very similar, meanwhile the loss of the SGD method is a bit higher.

Finally, we selected Adam as optimizers due to his stability with respect to the other methods.

C3 Model: Test Accuracy

Different datasets and batch sizes to find the best result
(Bigger datasets were created using Data Augmentation)



After implement our architecture in PyTorch we made different tries changing the parameters. We used several values of batch size (16, 32, 64, 128) and we proved it on the different datasets that we had created with the data augmentation.

Analysing our results we can see that all the values are very similar, but the highest one was obtained with **16 epochs** and using the **Dataset 3 (79% of accuracy)**.

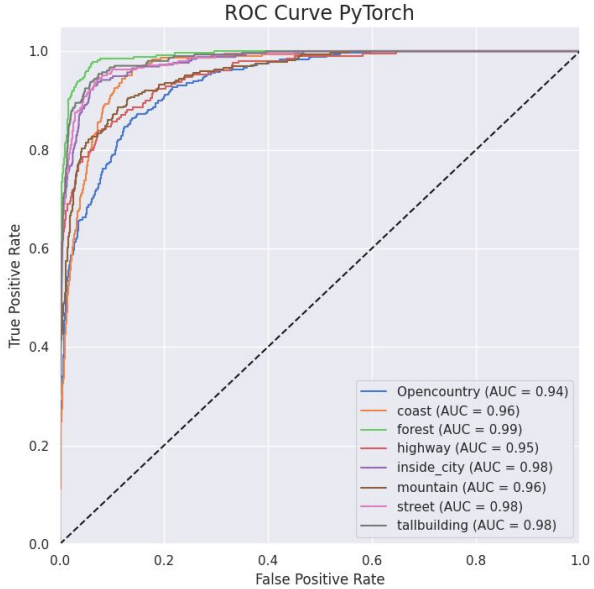
C3 Model: Training Curve

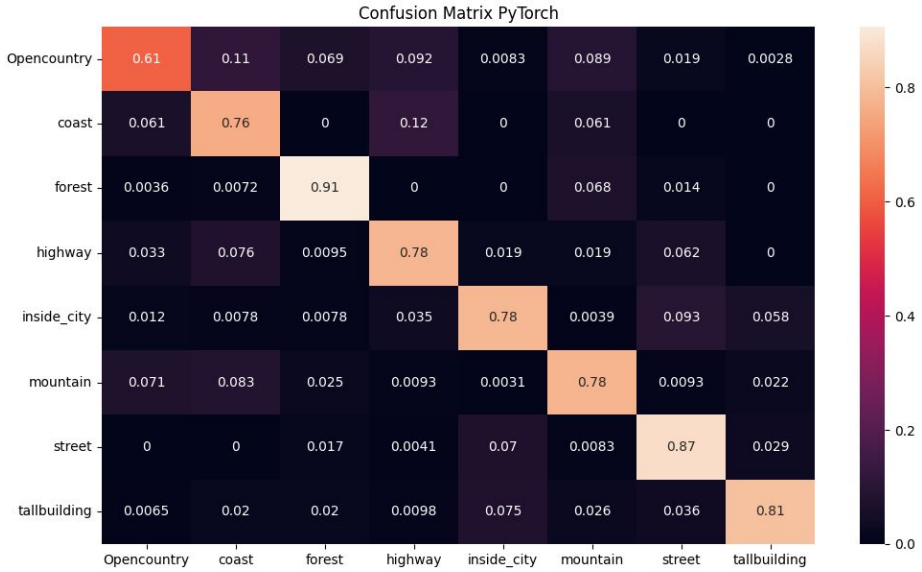
If we select the best hyperparameters and compare the test results obtained, we can see the improvement of the model in every epoch, but it is not able to surpass 80% of accuracy.

The model overfits the used training set. Using data augmentation and other regularization techniques, such as Batch Normalization, was not enough.The main reason is the limited amount of training instances, since by using such a small dataset the model is not able to generalize correctly.

C3 Model: Accuracy by classes

If we evaluate the accuracy class by class we can notice that there is one class than reduce the final accuracy mean. The class is opencountry and is mainly confused with coast, highway and mountain. That has a lot of sense since in opencountry images there are a highly marked horizont line and this is also a feature of the other classes.





PyTorch Lightning: Pros vs Cons

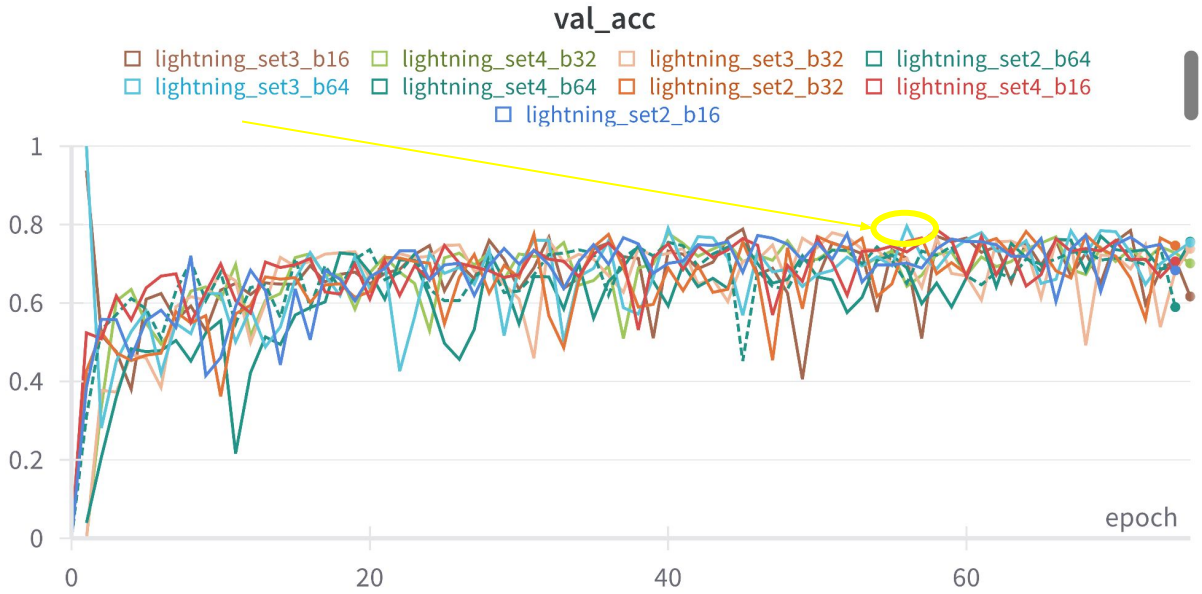
Pros:

- Reduces the repetitive tasks, simplifying the code.
- Easy scaling on different hardwares and also easy to reproduce.
- Can help to optimize our code to reduce the computational cost.

Cons:

- Since it is more recent, there is less documentation, and it is less extended among the community.
- It is difficult to change training hyperparameters once the training has started because we have not total control on the training cycle.

PyTorch Lightning: Best results



If we execute our architecture using Lightning PyTorch and evaluate the validation accuracy results, we obtain very similar values than the obtained with Standard PyTorch.

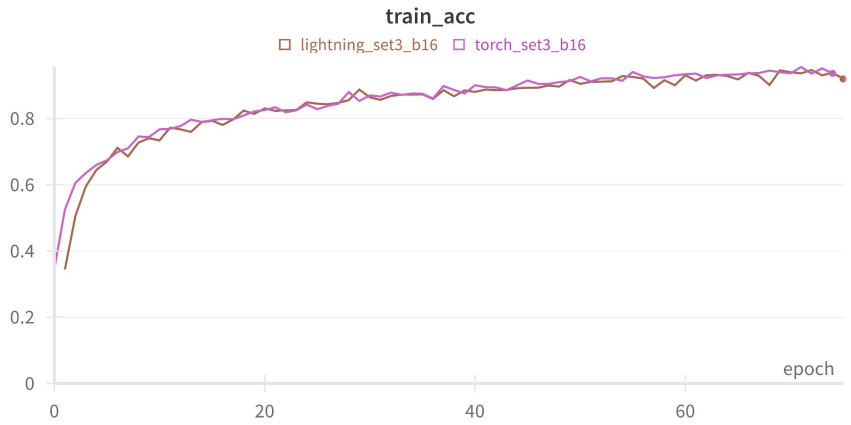
But this time the highest accuracy was obtained with batch 64 and again the Dataset 3. This makes sense because the accuracy value of these hyperparameters was very near also in PyTorch.

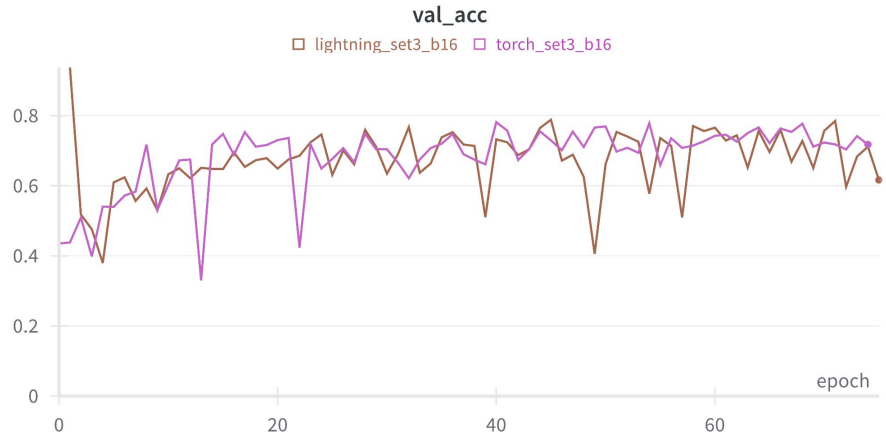
From now on, we will use the values obtained with batch 16 and Dataset 3 in order to compare the data with the same hyperparameters.

PyTorch Lightning vs PyTorch: Accuracy

If we analyse the accuracy curves of Lightning and PyTorch we can see how they are very similar, specially the train curve where the unique difference may be that the Lightning one is a little more unstable on the final epochs.

Something similar occurs on the validation ones, where the values are very near but again the Lightning is more unstable specially at the final epochs.

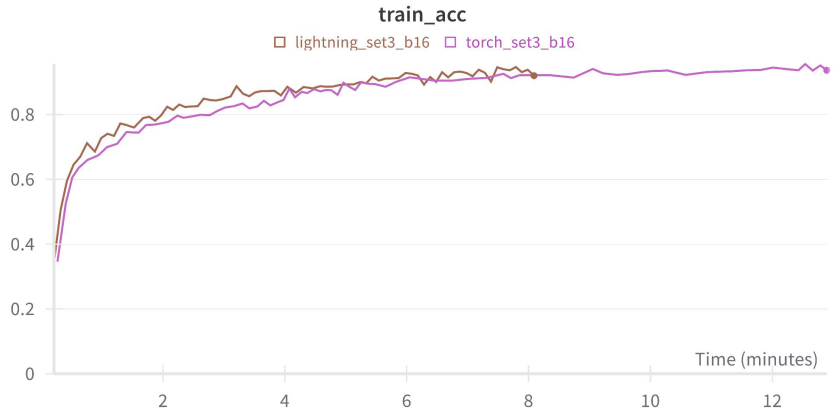


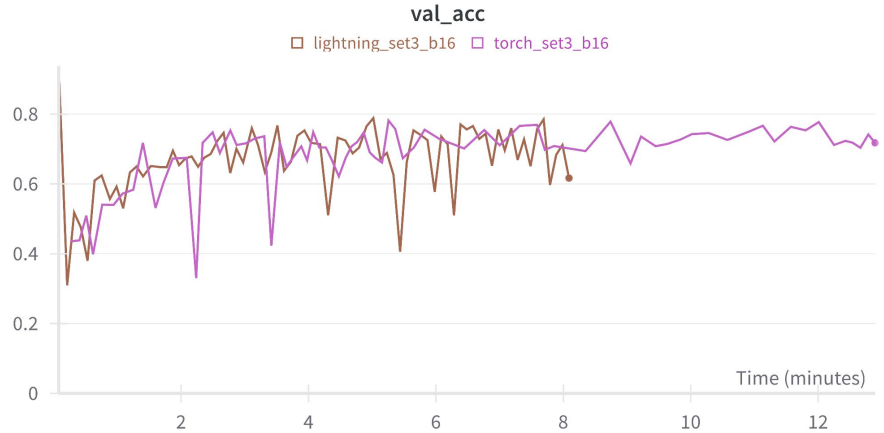


PyTorch Lightning vs PyTorch: Accuracy

But if we compare them by putting the processing time in the x-axis there are big differences. We can see how faster is the Lightning method compared with the standard framework. There is a difference of 4 minutes between them.

The main reason of that could be that since we are using a framework that is specially created to be simpler and standardized, we have created a more optimum version of our old code implemented with Standard PyTorch. So as we have optimized our code, we may have eliminated some repetitive structures that made the computation slower. (But we have to take into account that if the original code used on PyTorch is already optimized, the velocity will be very similar).



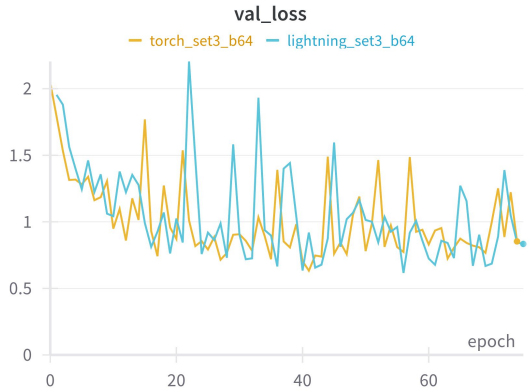


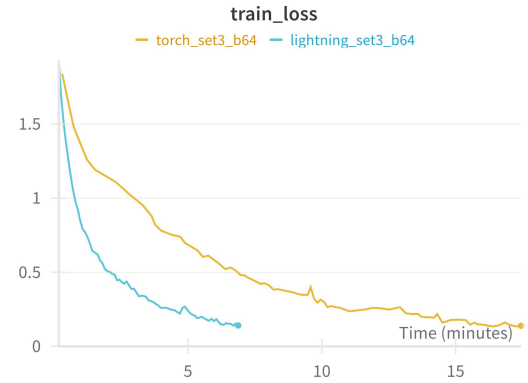
PyTorch Lightning vs PyTorch: Loss

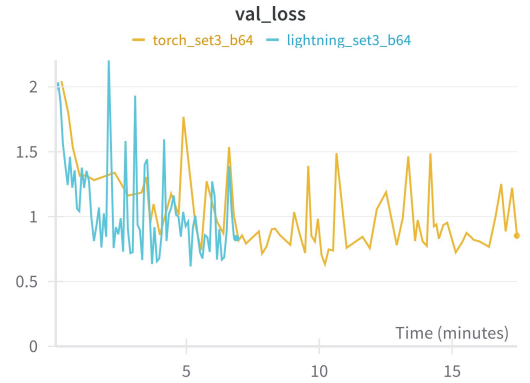
Analysing the loss curves, we reach the same conclusions.

First, both methods converge to the same values, but this time Standard PyTorch is more unstable for the loss values. This makes sense, since it preserves more stability for the accuracy.

Second, we can also see that training is faster by visualizing the loss over the time on the x-axis.

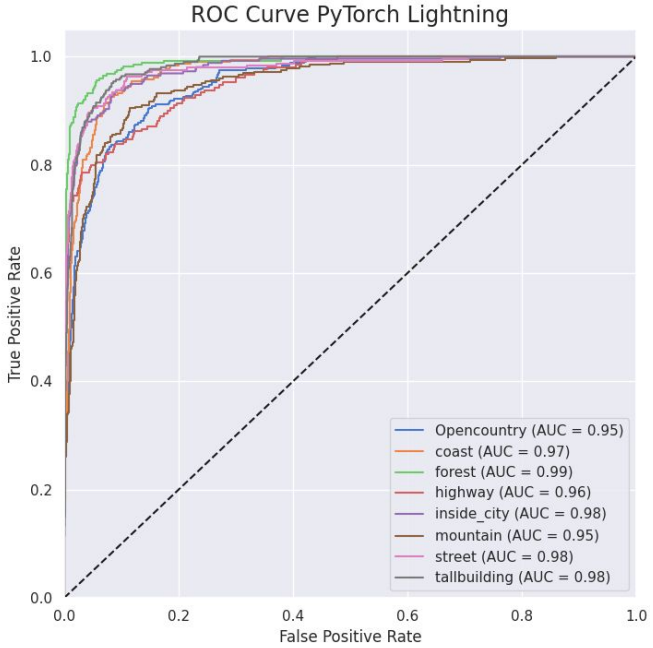


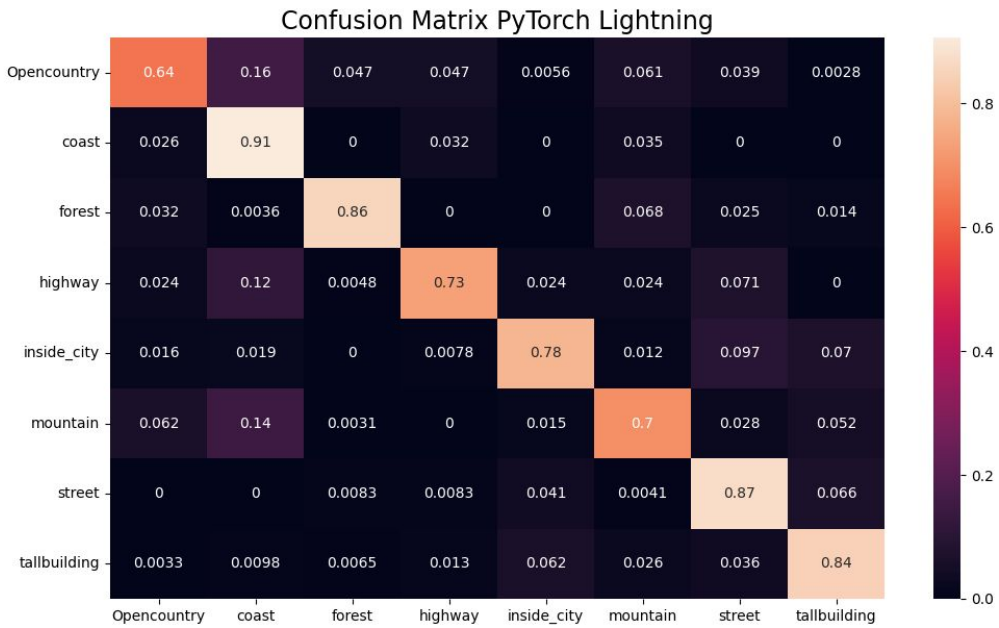




PyTorch Lightning vs PyTorch: Accuracy by Classes

As we stated in the original confusion matrix, both models have problems with the OpenCountry label. With PyTorch Lightning the mountain class performs worse than with the original model.





PyTorch Lightning vs PyTorch: Bad Predictions

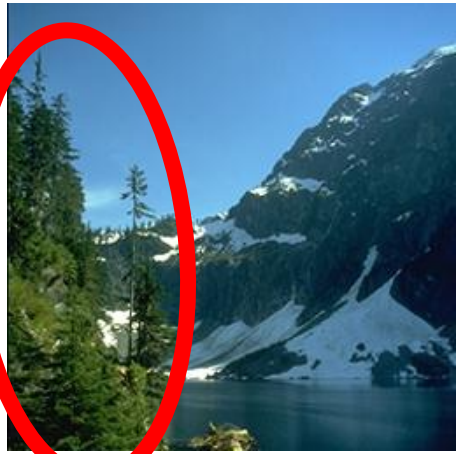
Pred: coast
GT: OpenCountry

The presence of forest should be a disciminant factor.

Pred: highway
GT: OpenCountry

The river assembles a high-way with a smooth gray texture.

Pred: forest
GT: mountain



The sample contains parts of a forest.

Final Remarks

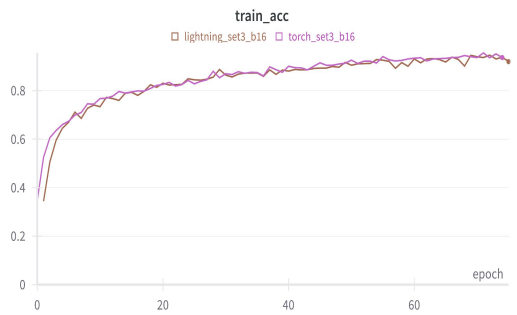
PyTorch vs PyTorch Lightning

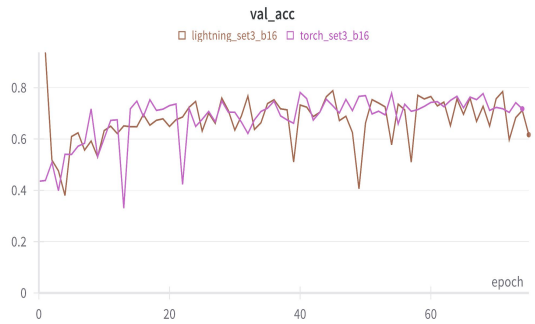
Problems:

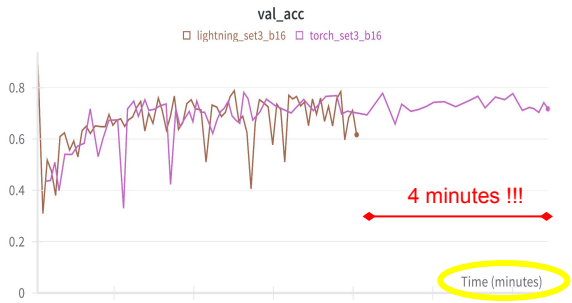
- Understanding the differences between PyTorch and Lightning.
- Code transformation specially when working with tensor values that we have to transform them using the CPU.

Conclusions:

- Lightning is simpler and more standardized so can help us to avoid code loops and repetitions that increase computational cost.
- Although the architecture is simpler provides very similar results than Standard PyTorch, maybe the only difference is that the Lightning curves are a bit more unstable.





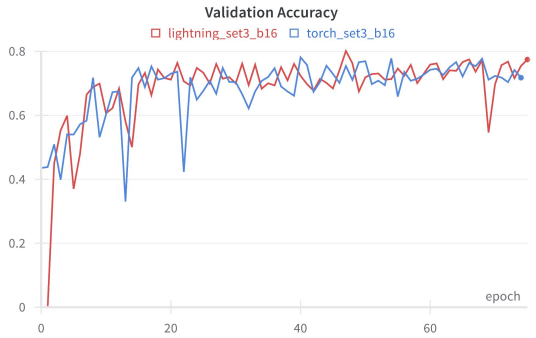


Summary Slide (G7)

PyTorch vs PyTorch Lightning

Results

Lightning achieved slightly better results than PyTorch

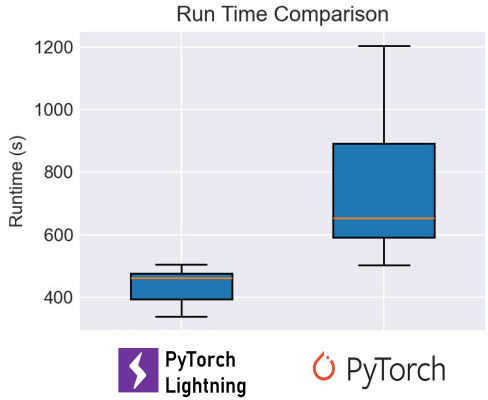


Best Val	80.16%	77.84%

Both achieve similar results

Performance

12 Runs for each framework



Avg	7.23 mins	12.49 mins

Lightning is **faster**
Lightning is more **stable**

Code

PyTorch let's you code

```
optimizer = torch.optim.Adam(model.paramete
for batch in dataloader:
    optimizer.zero_grad()
    inputs, labels = batch
    outputs = model(inputs)
    loss = loss_function(outputs, labels)
```

Lightning has a convention

```
class LitModel(pl.LightningModule):
  def training_step(self, batch, batch_idx):
    inputs, labels = batch
    outputs = self(inputs)
    loss = self.loss_function(outputs, labels)
    return loss
```

PyTorch is more **flexible**
Lightning is more **opinionated**