Question 1 :

PointMLP	ModelNet10_PLY	ModelNet40_PLY
25 epochs	22,0 %	15,2 %
250 epochs	17,2 %	25,1 %

We can see that using the PointMLP model isn't efficient at all, even when we use a lot of epochs to train it. Indeed, this model isn't well fitted to this use : we only have fully connected layers, which means that the pointcloud is processed as a data field, and no geometrical features are taken into account. For example, two symmetric pointclouds wouldn't be processed the same way, which is done in the data preprocessing (when using rotations).
Thus, the accuracy of the network cannot be very high.

Question 2 :

PointNet Mini	ModelNet10_PLY	ModelNet40_PLY
25 epochs	88,5 %	81,6 %
250 epochs	90,2 %	86,6 %

On the ModelNet40 dataset, we achieve approximately 87% of accuracy, which is way better than the previous model. This is expected, because the maxpooling layer will re-orient the input vectors in an orientation-independant way.
The results with this "mini" PointNet version is already a significant improvement.

Question 3 :

PointNet Full	ModelNet10_PLY	ModelNet40_PLY
25 epochs	89,8 %	84,4 %
250 epochs	90,7 %	86,7 %

We can see that the results are a little bit better than with the previous model, but the gap isn't as high as expected. This can be due to the fact that the data augmentation isn't very significant : the T-Net network allows the network to be more resilient to affine transformations for example. We should then try to see if such data augmentations can have an impact on the performance (see Question 4).

Question 4:
We can augment the data by using random scaling and random translations as well to see if we can improve the results.

PointNet Full – augmentation	ModelNet10_PLY	ModelNet40_PLY
25 epochs	84,7 %	81,5 %

We can see that the results are worse than the previous one, however there are still over 80 % accurate, and we lost only about 5 % of accuracy when the data was changed by a scale and translation factor.

However, when comparing to the PointNet Basic network (without the T-Net), we can see that the results got significantly worse :

PointNet Basic – augmentation	ModelNet40_PLY
25 epochs	73,4%

So the PointNet Full network is more resilient to the preprocessing, compared to the Basic version. The T-Net is thus fulfilling its job.

Question Bonus :
I wasn’t able to find the relevant file in the GitHub repository, and wasn’t even able to install the Minkowski engine on my computer : the error message was « segmentation fault : core dumped », and after a lot of tinkering, and finding out that even the aur package for the minkowski engine isn’t up to date, I decided to let it go.