Carlos Marzal Romon

DLCV-Homework 4

1. Trimmed action recognition w/o RNN

1.1. Strategies and implementations

First of all we process the videos with the given reader file but changing the code so that it makes the video frames squared (adding black lines on the top and bottom of the videos), resizing them to 200x200(otherwise it would give a CUDA out of memory error) and normalizing it using the typical mean and std values. We also only get one of every 12 frames (basically videos at 2 fps).

To extract the features of the frames of the videos I am using the Resnet50 pretrained model using the ImageNet dataset and taking away the last layer to have the output of size 2048.

Once I obtain the video frames features, I obtain the mean of the features to have a final 2048 tensor for every video, no matter how big it is.

Finally, this mean video feature is trained using a simple model that uses linear functions. The overall model of the system is the following:

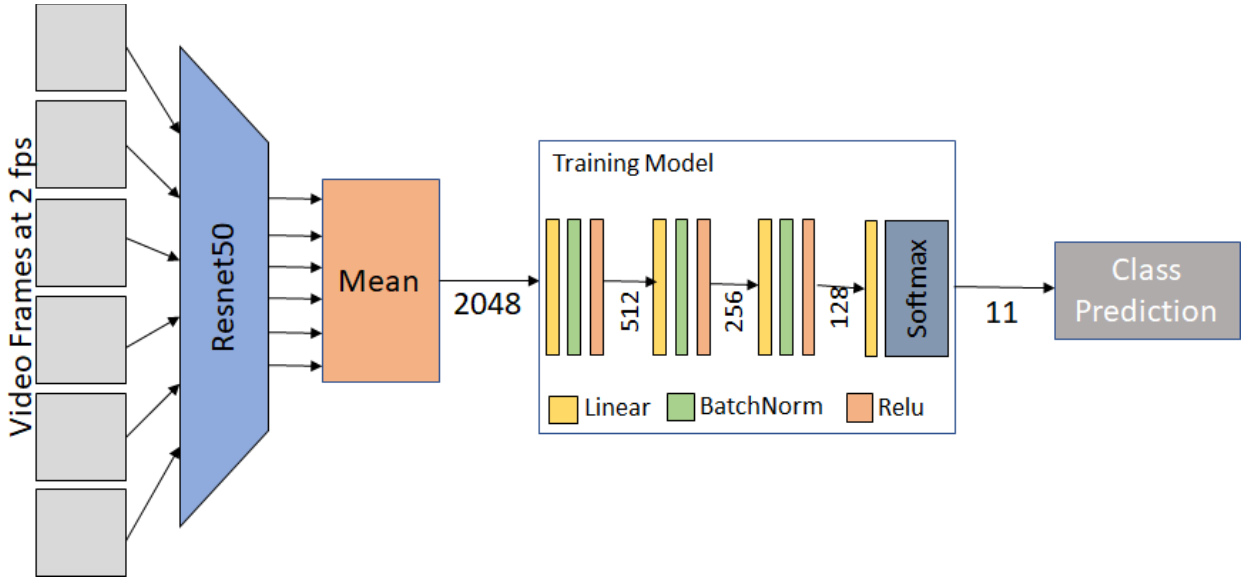


Figure 1 CNN model

In this architecture we are only training everything inside the training model. The features are obtained beforehand and stored in a file to then be used later to save time in having to pass all the video frames though the resnet every time we run the program.

Loss and accuracy curves:

We store the loss and accuracy of each epoch in an array to then save the following files; we can see how we reach equilibrium at around 70 epochs, where validation fluctuates between 35 and 41%.

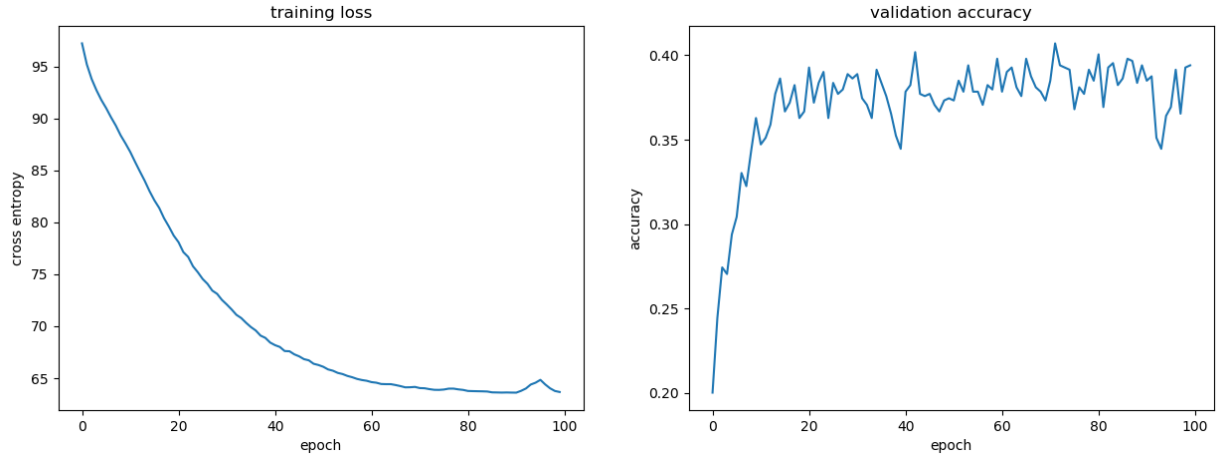


Figure 2 CNN Loss and Accuracy per epoch

1.2. Recognition performance

The performance obtained in the CNN without RNN is the following:

Accuracy: 0.4070221066319896 (40.7%).

Note that when running the program this accuracy might slightly change (<0.5%) because the Resnet50 has a slight randomness to it, but in general it should stay as 40.7%.

1.3. CNN-based video feature visualization

The next image shows the video features in a graph after passing said features through TSNE in two dimensions (the video features only come from the mean of the frames features coming from Resnet50, so in this case there was no training involved.

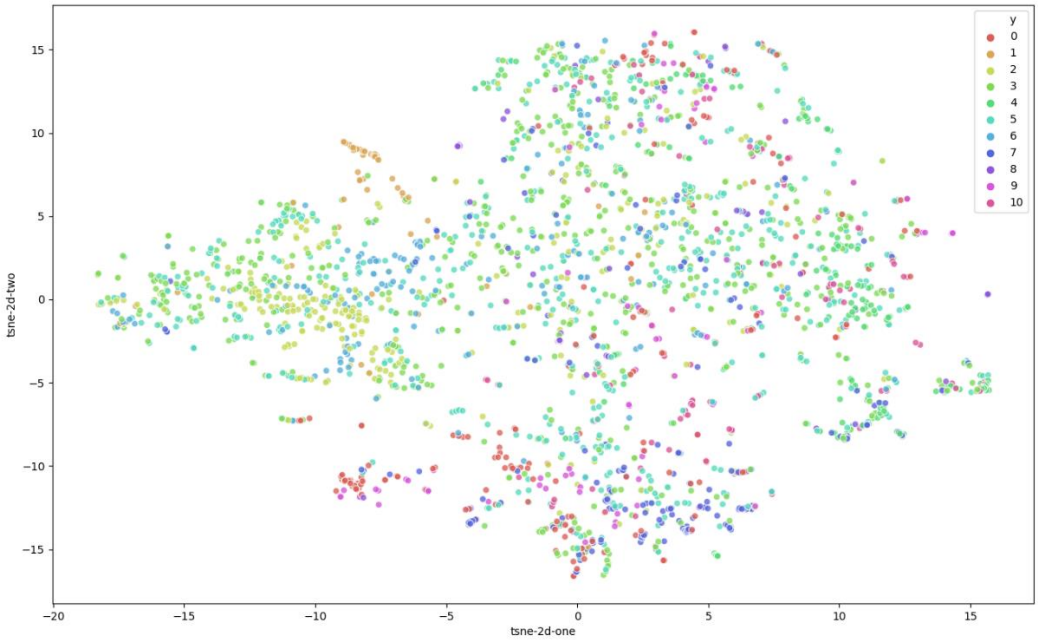


Figure 3 Video features CNN after TSNE

2. Trimmed action recognition w/ RNN

2.1. Strategies and implementations

In this problem we are also using the Resnet50 model to obtain the features of the frames and the whole preprocess of the images is the same as before.

The difference in the RNN architecture is the use of pack_padded_sequence to obtain the RNN features and get better results. For this we have two LSTM layers with a hidden size of 512 (that we will consider the features that then appear on the graph in the following section). Before passing though the model we have to zero-pad the different videos for them to have the same shape and size (by adding 0s when necessary).

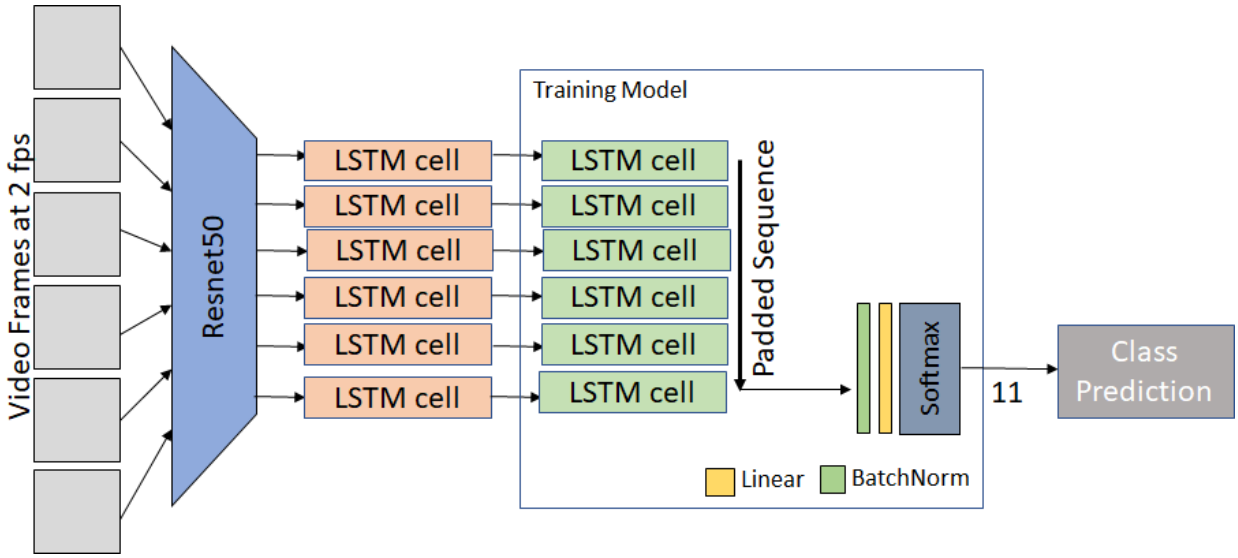


Figure 4 RNN Arquitecture

Loss and accuracy curves:

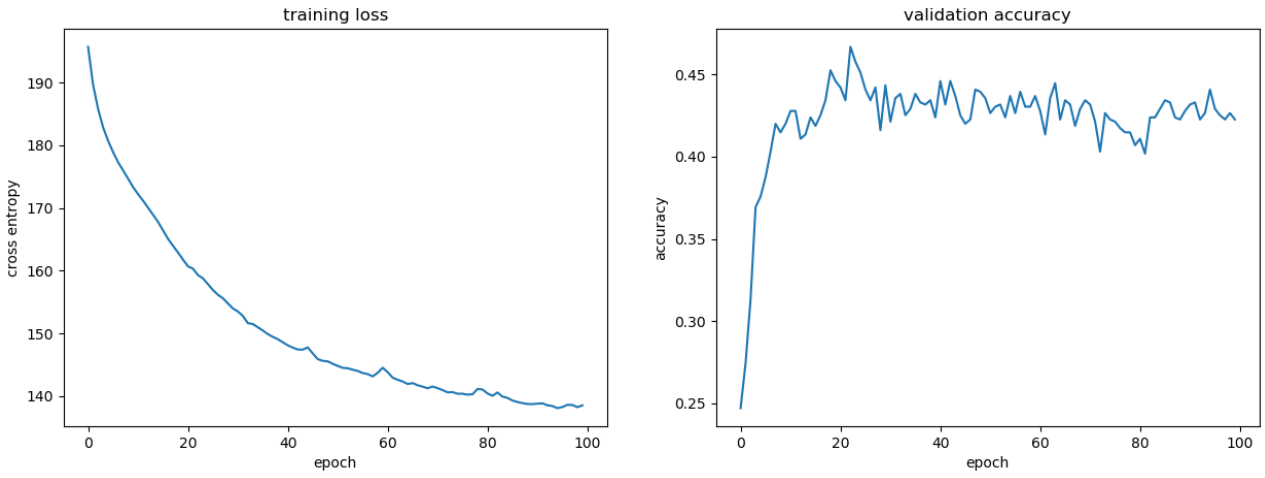


Figure 5 RNN loss and validation curves

The peak accuracy is obtained in epoch 22 and the loss starts to converge at around 90 epochs.

2.2. Recognition Performance

The accuracy of the model trained in the RNN architecture is:

Accuracy: 0.4668400520156047 (46.7%)

This passes the baseline asked for in the problem slides of 45%.

As mentioned in the previous accuracy, this accuracy might slightly change due to slight variations in the randomness of resnet, but it's not worth it to save the resnet model (>100MB) for a <0.5% difference.

2.3. RNN-based video feature visualization

These features are obtained from the hidden output in the trained model, so therefore they should be a bit better than the ones observed in exercise 1.

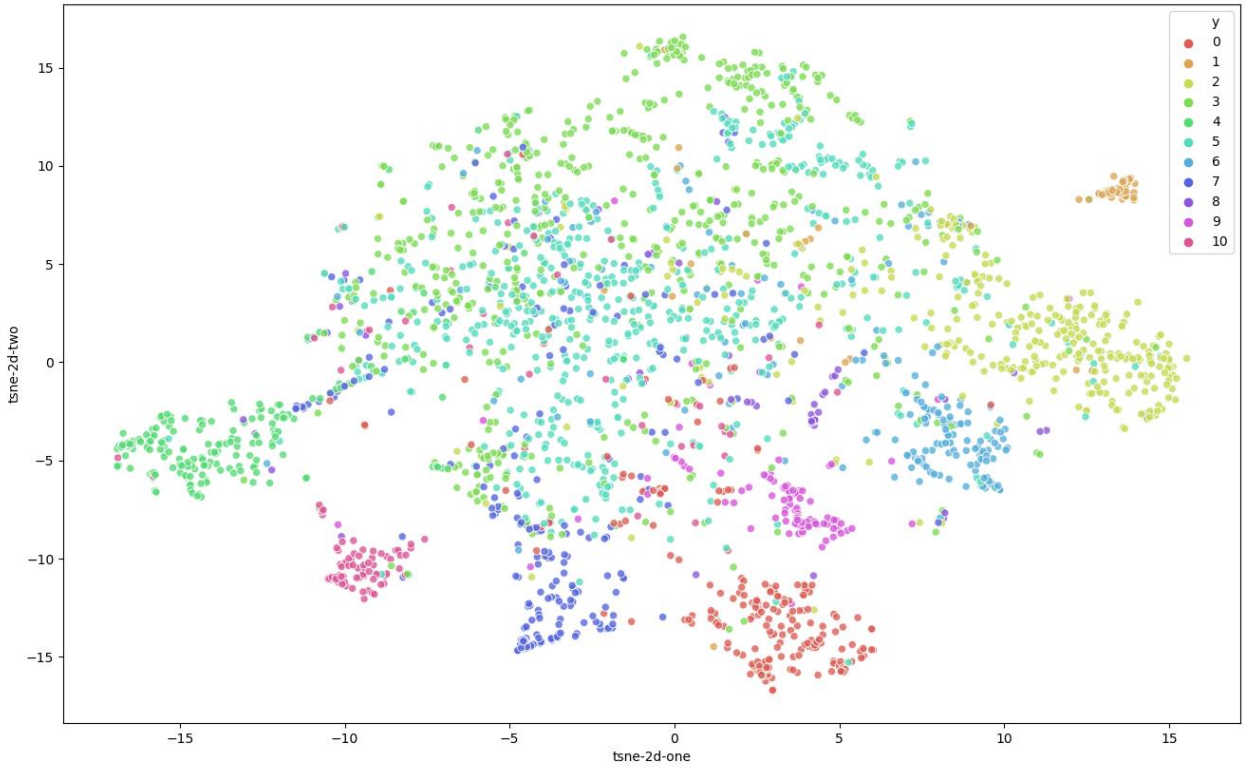


Figure 6 Video features of RNN architecture

As we can see each action seems a lot more differentiated than in the CNN-based video features. This is because RNN does a better job at getting features of a video than simply applying the average value of the frames like we did before. Also, these features come from a trained model in our specific dataset, so it will naturally have better results than before.

3. Temporal action segmentation

3.1. Strategies and implementations

For this exercise we are using the same model as we used in the previous one, but this time making the model return an output every time fragment. For this we are creating time fragments of 300 frames with overlapping frames with the next/previous time frame so that we can use some frames more than once in the training (overlap of 30 frames).

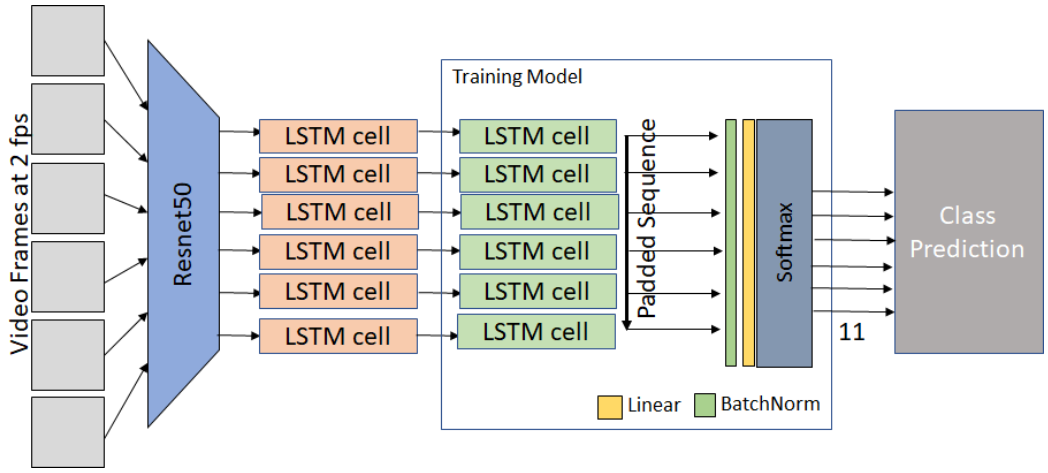


Figure 7 Sequence RNN architecture

3.2. Validation accuracy

For this validation accuracy I divided it into three different ways to calculate them: mean accuracy of videos, accuracy counting each frame the same amount and individual accuracy of the videos.

The values obtained are:

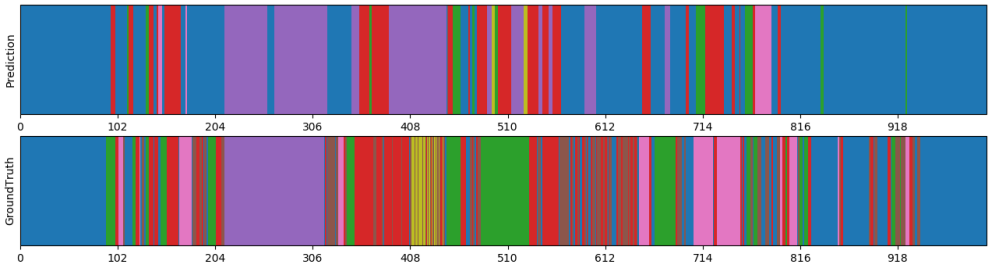
Mean Accuracy: 0.5894962024635669 (58.9%)
Total Accuracy: 0.6029313045424032 (60.3%)
Accuracy of video OP01-R02-TurkeySandwich.txt = 0.4575098814229249 (45.8%)
Accuracy of video OP01-R04-ContinentalBreakfast.txt = 0.6252545824847251 (62.5%)
Accuracy of video OP01-R07-Pizza.txt = 0.6317280453257791 (63.2%)
Accuracy of video OP03-R04-ContinentalBreakfast.txt = 0.5343082114735658 (53.4%)
Accuracy of video OP04-R04-ContinentalBreakfast.txt = 0.6433179723502304 (64.3%)
Accuracy of video OP05-R04-ContinentalBreakfast.txt = 0.5580168776371308 (55.8%)
Accuracy of video OP06-R03-BaconAndEggs.txt = 0.6763378465506125 (67.6%)

3.3. Visualization of Prediction

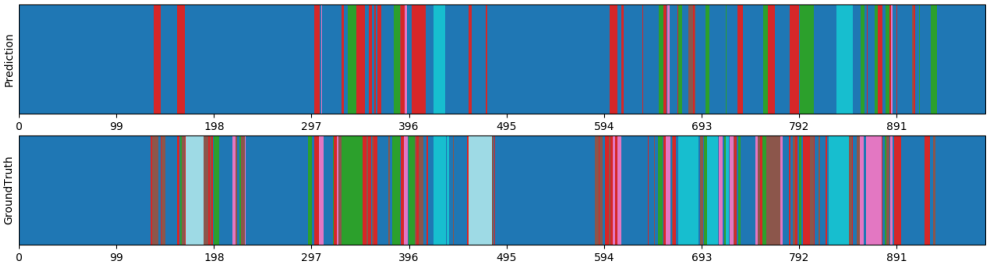
Here is the visualization of the 6 videos in the validation dataset. The colors represent the different class predicted in that area. The numbers on the bottom represent the frame inside the video and the top colors are the prediction and the bottom the ground truth. In general, other (blue) and cut(purple) seem to have better performances because they have to information to them and more training.

Carlos Marzal Romon

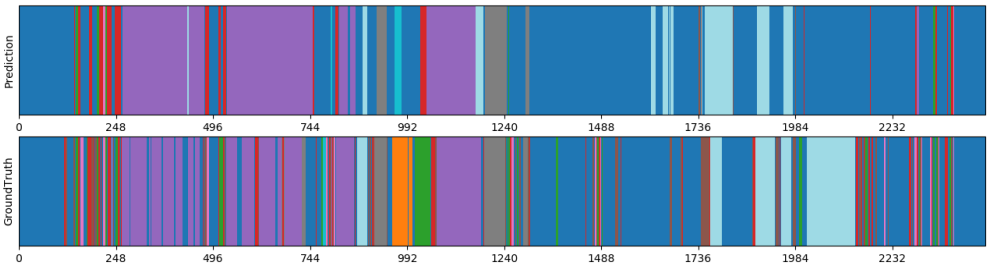
Video 0:



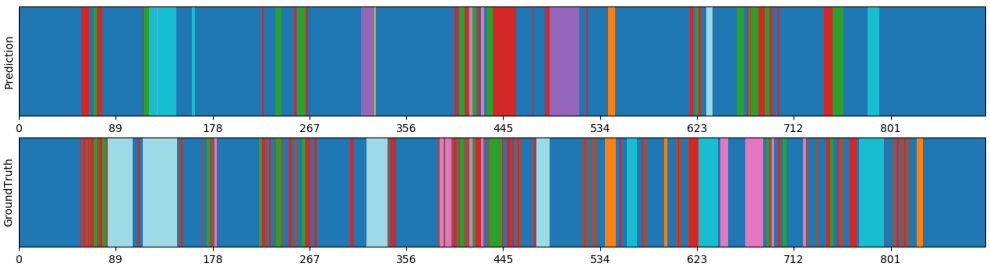
Video 1:



Video 2:

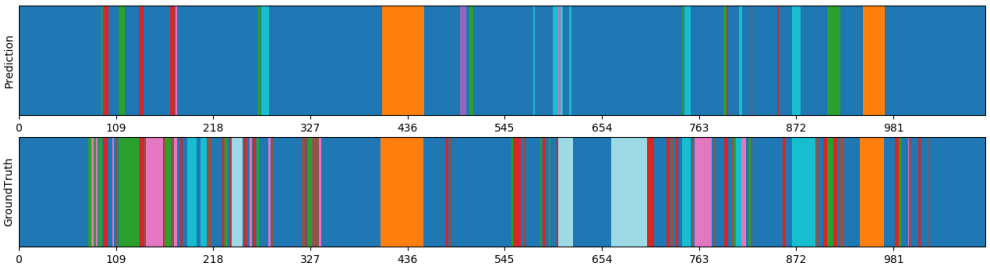


Video 3:

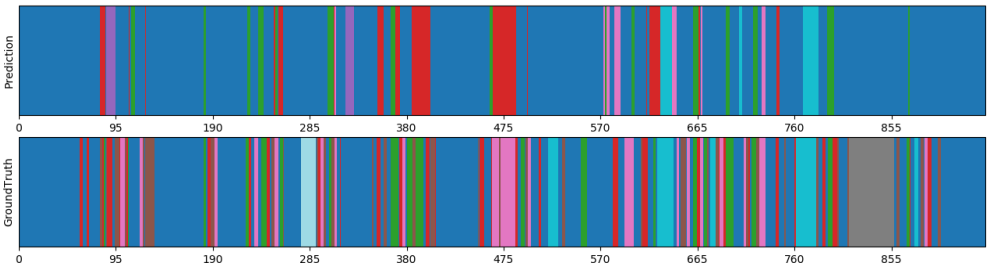


Carlos Marzal Romon

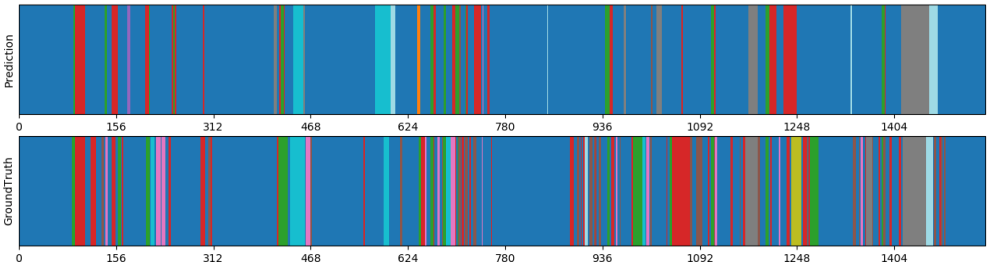
Video 4:



Video 5:



Video 6:



Carlos Marzal Romon

Bibliography

[1] RNN example, https://github.com/thtang/ADLxMLDS2017/tree/master/hw1.
[2] RNN explanation, https://zhuanlan.zhihu.com/p/34418001.
[3] Torchvision Resnet model, https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html.
[4] Torch CNN features for videos, https://github.com/kenshohara/video-classification-3d-cnn-pytorch.
[5] Example of homework for similar problem, https://github.com/thtang/DLCV2018SPRING/tree/master/hw5.
[6] Torch training a classifier, https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.
[7] CNN in Pytorch, https://algorithmia.com/blog/convolutional-neural-nets-in-pytorch.
[8] Other stackoverflow relevant links.
[9] Useful website, https://www.google.com.tw/

Students I collaborated with:
Ricardo Manzanedo-R08942139
Javier Sanguino-T08901105
Celine Nauer-A08922116
Julia Maricalva-A08922107