Testing Resport 03/05-05/05

Memory

I had added more recordings to the ones I was using for my models (from the panoptic studio dataset) -I am not using all of them yet-, and I found out that some of the new additions had some invalid frames at the end of them that generated padding the way I processed them. I fixed that and I started to run tests. I used 20 videos for training, 8 for validation and 8 for testing.

My architecture consists in a LSTM with N-layers followed by a fully connected layer with ReLU activation. I run tests on three different variations of this architecture:

1. “Small”- 2 layer LSTM with hidden size 512 (3,273,754 trainable parameters).
2. “Medium”- 2 layer bidirectional LSTM with hidden size 512 (8,644,634 trainable parameters).
3. “Large”- 2 layer bidirectional LSTM with hidden size 1024 (34,066,458 trainable parameters).

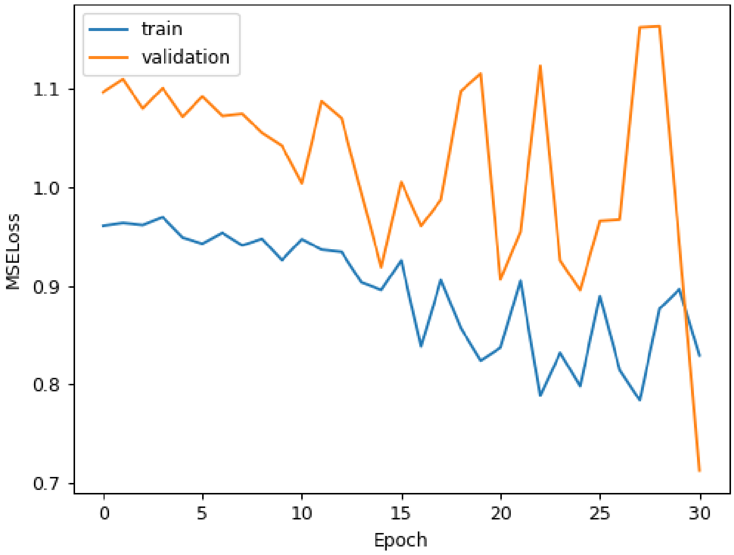
Since I wasn’t getting any results I decided to first try to overfit the models to the training data and then apply some regularization methods to prevent overfitting: dropout, weight decay and apply a schedule to the learning rate following the 1-cycle policy. As you will see, the results aren’t good, but I will discuss them on the sections below. For the “large” architecture I couldn’t get it to learn appropriately so I didn’t put the results.

Results

For every model in every section, I will put the results with the following format: first, on the left there will be the training and validation loss over the epochs and on the right the number of zero frames in each video prediction for training data; below, there will be the number of zero frames for validation data and after that, the test loss and test MPJPE; finally, I will plot a sample predicted skeleton against the groundtruth.

Body keypoints estimation

Small



Zero frames / Total frames:

```
1: 2496/8751
2: 2209/7836
3: 3785/8045
4: 4118/8752
5: 4098/8751
6: 2442/5155
7: 1/8272
8: 1/7326
9: 0/6012
10: 0/7141
11: 4058/8751
12: 4058/8751
13: 1502/8272
14: 934/5176
15: 2/6012
16: 2/8272
17: 0/8272
18: 0/8752
19: 0/5952
20: 0/8751
```

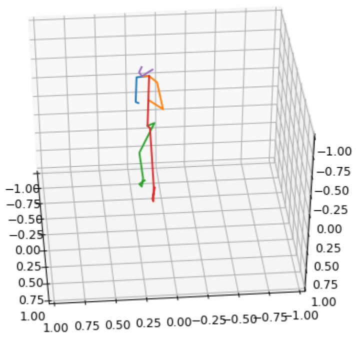
Zero frames / Total frames(Valid):

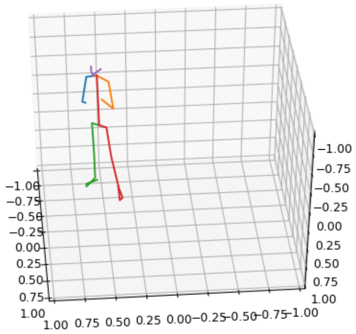
```
1438/5155
1444/5176
3354/7141
3439/7326
2811/5952
5861/8574
1/7611
1249/8574
```

Test:

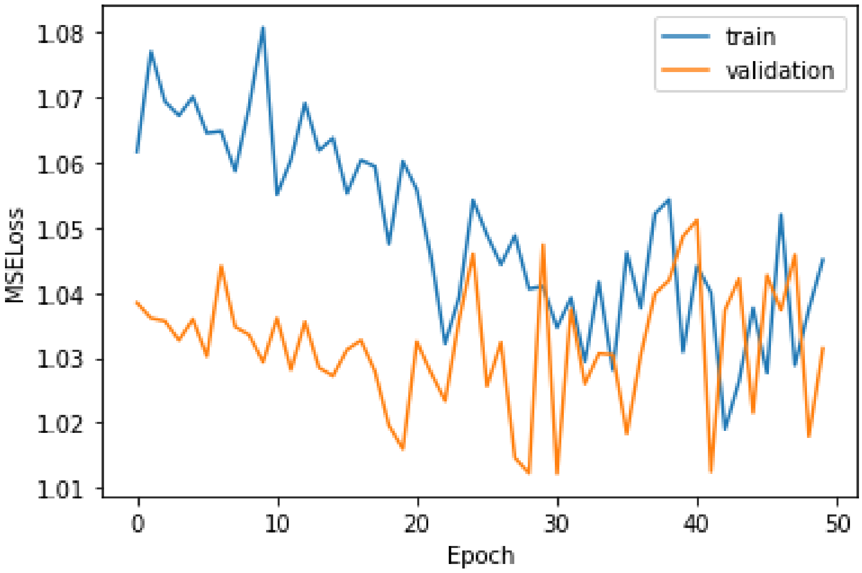
```
MPJPE: 0.0625 Test loss: 1.0686
```

Training plots [predicted | groundtruth]





Medium



Zero frames / Total frames:

```
0/5176
0/8751
0/7141
0/7141
0/8574
0/8574
0/7611
0/8751
0/6953
0/8272
0/5952
0/8751
0/7611
0/5155
0/8272
0/8751
0/7326
0/5155
0/6012
0/8751
```

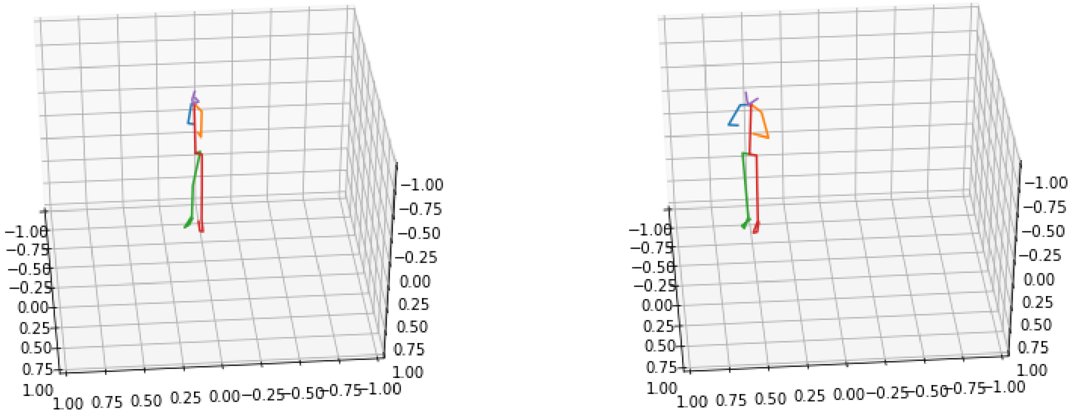
Zero frames / Total frames(Valid):

2869/8045
0/8271
1131/8272
0/6953
0/8272
0/7836
225/7836
0/6012

Test:

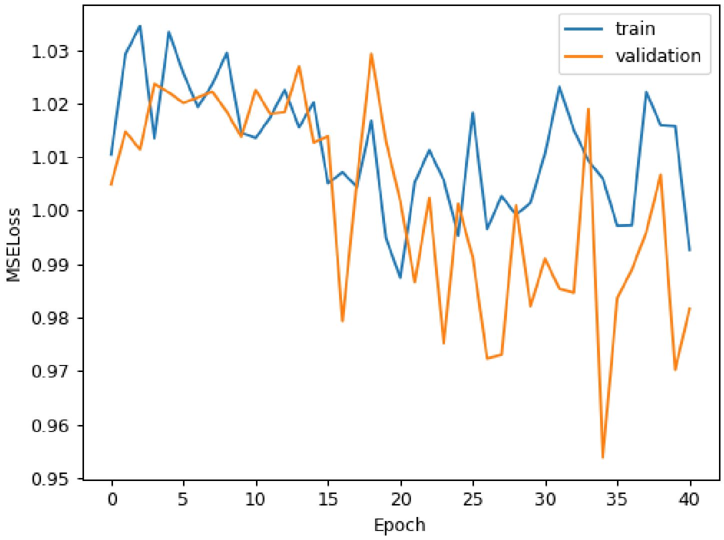
```
MPJPE: 0.0486  Test loss: 0.8034
```

Training plots [predicted | groundtruth]



Hands keypoints estimation

Small



Zero frames / Total frames:

```
2136/8751
1999/8272
0/6012
0/5176
0/5952
0/8751
2128/8575
2172/8752
10/5952
11/6953
0/8575
0/8751
1919/8272
1900/8045
0/8272
0/7326
0/8271
0/7836
1000/5155
1487/7141
```

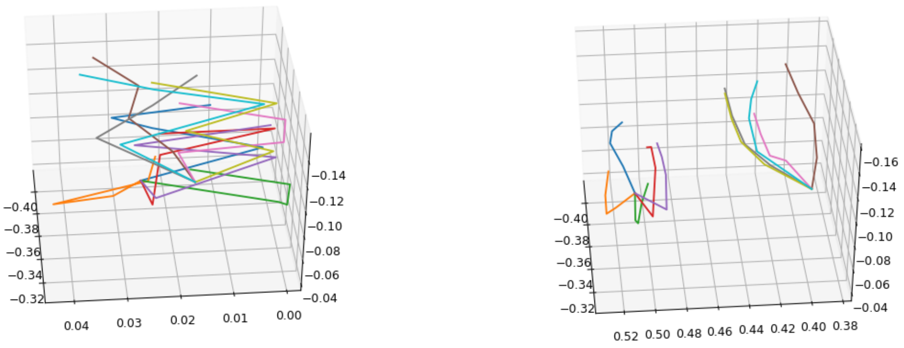
Zero frames / Total frames(Valid):

```
1738/7326
2478/8751
0/5155
2435/7611
1001/6953
1/8752
1993/8045
1884/7611
```

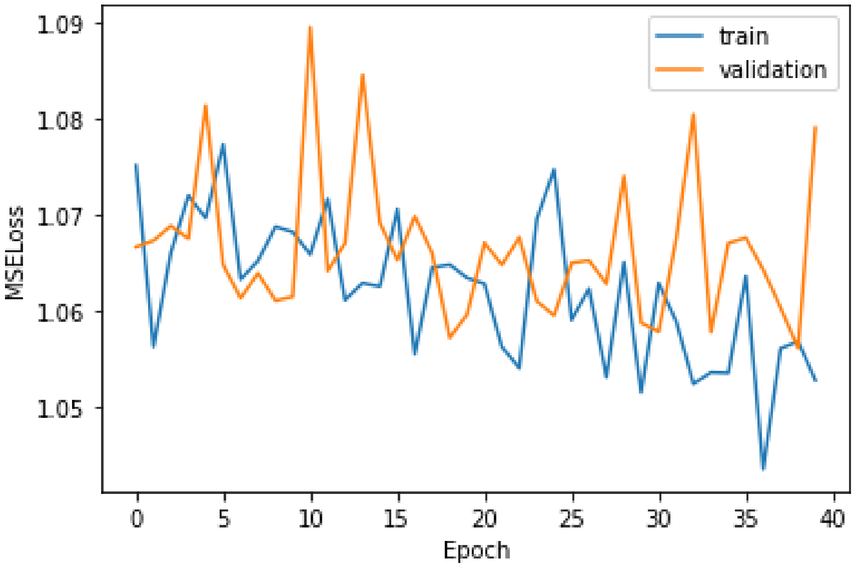
Test:

```
MPJPE: 0.0243 Test loss: 0.9396
```

Training plots [predicted | groundtruth]



Medium



Zero frames / Total frames:

```
0/8272
0/8752
0/5952
0/8272
0/6012
0/8045
0/8272
0/7141
0/7611
0/8751
0/7836
0/8271
0/8271
0/7611
0/5155
0/5952
0/8751
0/8575
0/8751
0/8751
```

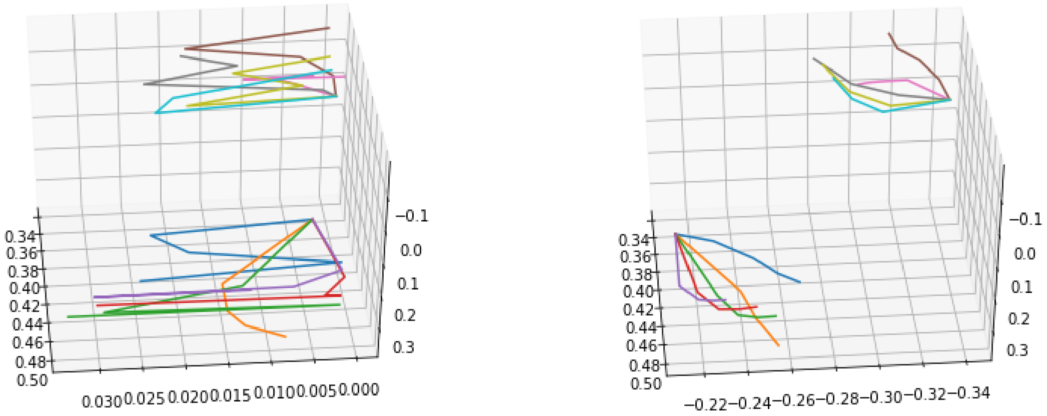
Zero frames / Total frames(Valid.):

0/5155
0/6012
0/5176
0/6953
2260/8272
0/7326
0/8045
0/5176

Test:

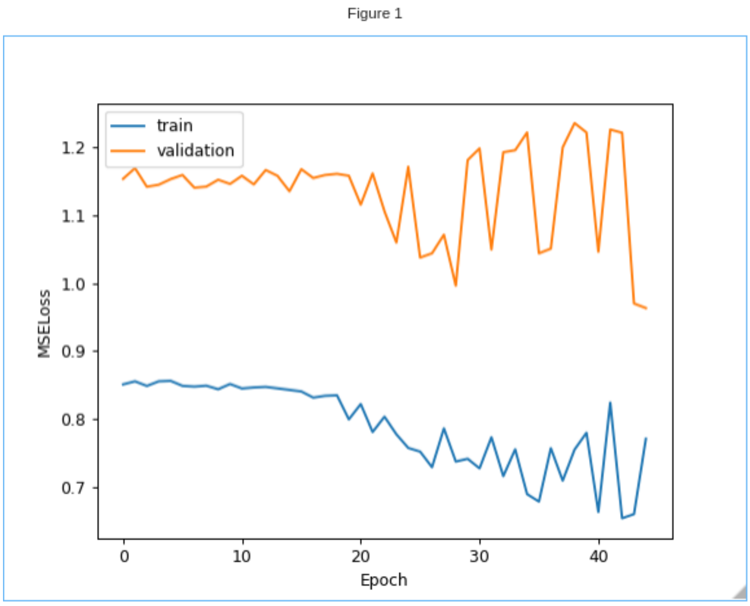
```
MPJPE: 0.0266 Test loss: 0.7840
```

Training plots [predicted | groundtruth]



Face keypoints estimation

Small



Zero frames / Total frames:

```
0/8272
0/7326
3878/8752
3562/8045
2265/6953
2506/8575
3584/8045
3684/8272
2/8751
2/8271
6900/7836
4563/5155
1/5155
1/7141
3748/8751
3749/8752
3285/7611
3285/7611
3199/7326
3837/8751
```

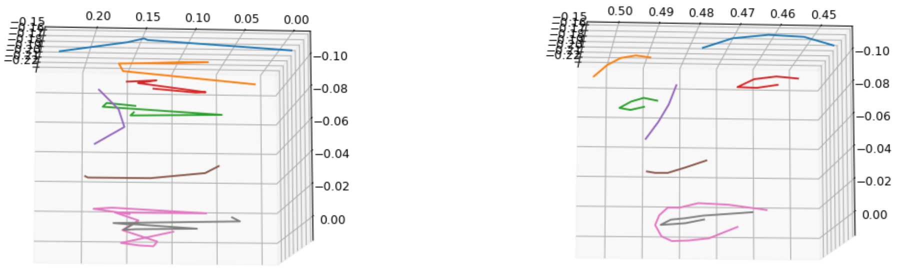
Zero frames / Total frames(Valid.):

0/8272
0/6953
2637/5952
2662/6012
1965/6012
2682/8751
3183/7141
2665/5952

Test:

```
MPJPE: 0.0258 Test loss: 1.3394
```

Training plots [predicted | groundtruth]



Medium

Similar to previous

Analysis

As you can see, the results are quite bad. The losses get really noisy, though they have a decreasing tendency. I didn’t show the results of the “Large” architecture because I couldn’t make the training loss decrease. More capacity doesn’t seem to improve the results as it is now.

Also, these are some of the best results I could get, but every time I load the keypoints I shuffle them randomly, and I noticed that without changing parameters it differed a lot from a run to another. Then, there is a high variance on the data I am using.

Difficulties

The project is much more time consuming than I expected. Since I am not getting results, I have to test over and over changing some parameters and it takes ~30 s to train an epoch if I train for 40-60 epochs, that makes 20-30 min each run. Although I use 2-3 computers simultaneously, it is still a slow process.

Furthermore, I am not using the full dataset because it takes too much time to download a new recording, copy it to my working directory, remove the blank frames and load it to my runtime.

I have been working as much as I planned and more, but it is taking me much longer than expected to get good results, so I don’t know whether I will be able to accomplish the objective with good performance.

Conclusions

I. I need to use more data to increase the batch size to reduce the noise and variance. Since I have few videos but each has a lot of frames, I thought of splitting each video in, for example, 4 parts and take them as a video each. I was avoiding to do that, because I could lose temporal information of the sequence, but I think it’s worth giving it a try. Of course, I will also add a couple of recordings more (but it’s still too much time consuming).

II. Network still have some tendency to predict zero, despite knowing when there is padding on the input and where. I may need then to tackle the problem with a different approach.

III. I want to try other solutions, such thinking about the problem as a classification task (currently working on it), or try a transformer (encoder-decoder) architecture. But since the final report due date is 18th May I won't have time to try much.