CS764: Assignment 7b

Report

Siddharth Saha 170100025 **Parth Shettiwar** 170070021 **Parikshit Bansal** 170050040

Training Details

The main problem to be challenged here is weak supervision training. The model still outputs the 3d coordinate of the pose, but the issue here is that we don't have true 3d coordinates to enforce loss. There are multiple ways to go about solving this issue of imposing loss

1. The predicted 3d coordinates for image 1 is taken and along with the rotation and translation matrices given projected to get the 2d image joints in the second view. Here we need all the given paramters i.e. rotation translation and focal length of the second camera to get the projected image in 2d space. We use MPJPE loss in 2d to train the model
2. The predicted 3d coordinates for both image 1 and image 2 are predicted using our model. Then we transform image1 into image2 view and impose a MPJPE loss on the predicted 3d coordinates.

We coded up both the approaches but the first approach gave worse results than the second one. Hence, we use the second approach. A subtle point to be considered is that the the cameras are different and hence has different camera matrix i.e. focal length and optical centers. Hence they should ideally have different models to convert 2d to 3d, or have focal length/optical centers as input to the model.

Optical centers are considered to be (0,0) and the focal length of both the cameras was observed to be emperically so close ((1.149 1.148) and (1.149 1.147))to as to produce neglible difference in 3d coordinates if they were taken into account. Hence just a single model would work in our case.

We still experiment with 2 different models to account for this difference in focal length and any other distortion which might have been not specified in the problem statement. We tried out emperically that it didn't work out. It was expected as this method essentially doubles the number of parameters while halving the data available each model.

Note that we could also have predicted the rotation and translation matrix from view1 to view2 using a neural network, but the same would lead to collapse of the network to a trivial solution i.e. some constant outputs predicted irrespective of the input. Hence the given translation and rotation matrices play an important role here.

The transformation to go from view1 to view2 is given as

$$V_2 = RV_1 + T$$

where R is converted from (9,1) to (3,3) using view operation of pytorch

Validation Details

One tenth of the training data is kept aside for validation. Validation is done in the same way as testing, i.e. we take the loss with the true 3d coordinates. Testing also can be done in the same way.

Main Numbers

To train the model we use Adam optimiser with a learning rate of $1e-3$. The batch size is kept at 64. 10% of the data is kept as validation data and is not used for training and used to draw the validation plots. We train the model for 50 epochs. Model weights are stored in a file called 'checkpoint.pt' in the same directory as train.py. eval.py uses these weights to test the model. The validation loss is decreasing for this range of epochs. The code works on CPU by default but can be changed to work on GPU by changing a single line in training.py

We kept the number of blocks to be constant to be 2 and have the following plot for the same

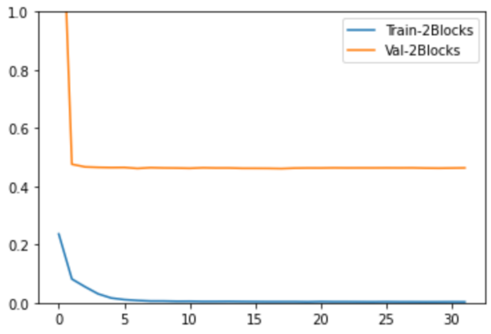

Figure 1: (b)

Note : To replicate the results copy the data_train.pkl file to the directory and run the training.py file followed by eval.py file. A jupyter notebook of the same is also submitted.

Other Numbers

Numbers for other experiments run mentioned are given as follows

3D, 2 models	0.94
3D projected to 2D	Validation Loss diverges from Epoch1
3D, 1 model (Ours)	0.46

Table 1: Caption

References

[1] https://drive.google.com/file/d/1dWd0G6WPVnnDkoKHSkin6d7lbobLWw4o/view

[2] https://pytorch.org/docs/stable/nn.html