CS 763/CS 764: Lab 7(a)

Imputing 3D from 2D

- Announced: 03/26. Due: 03/31 5PM
- Note: This is an individual assignment

In an earlier lab, you went through the classic way of inferring 3D points from 2D images. This was done explicitly using an Augmented Reality application.

In this lab, we will use the tools of neural network learning to implicitly infer several 3D coordinates of human skeleton based on two dimensional information.

1 Problem Overview

The goal of the assignment is to predict a 3D skeleton given a 2D skeleton of a human as shown in Fig. 1. The 2D skeleton itself is obtained from running another "detector" that outputs "significant corners" of the skeleton given an RGB image containing a human being in an abitrary pose.

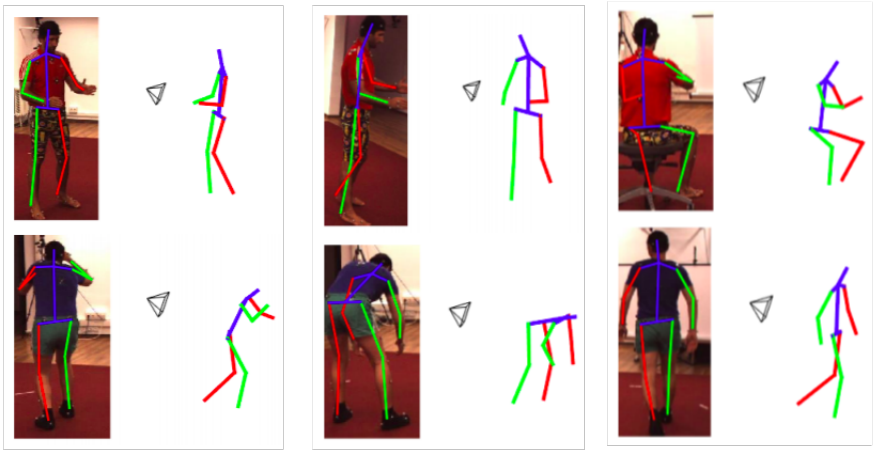

Figure 1: 2D input skeletons are overlaid on RGB images. Shown adjacent to each image are (a) 3D rendering of the corresponding skeleton, and (b) a camera configuration. If the center of the camera is as shown, the resulting RGB image would be the one shown.

2 The data and the model

Note: Some of the questions below are best answered after the experiments are conducted.

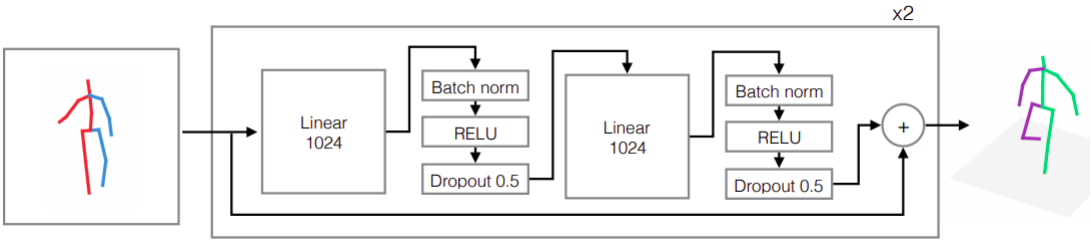

Figure 2: The network is composed of two so-called residual blocks made up of linear layers. The figure shows one such residual. Missing in this picture at the input stage is a linear layer that takes the input skeleton to the 1024 node linear layer. Also missing is another linear layer at the output stage that maps 1024 nodes to the 3D skeleton.

1. Observe the network architecture (Fig. 2) that is to be used.

   Q1. Explain the purpose and location of the batch normalization later, and the dropout layer. What is the reason for duplicating the layer. What if we used three blocks?

2. Any such neural network architecture typically requires labeled data. The data is to be retrieved from this location (please take care to keep the data in your private space and do not circulate).

You will observe that the data is organized in the form of a dictionary with the usual (key, value).

- `'joint_3d'`: This key points to the "true" "3D" of several skeletons. Each skeleton has 15 joints and a root joint (not included in the array) with coordinates $(0, 0, -1)$. The values of the joints are real numbers in some unknown unit. The coordinates are w.r.t to the camera center which is to be treated as the origin.
- `'joint_2d_1'`: This key points to several 2D-skeletons each of which are projections of the corresponding 3D skeleton. Each skeleton has 15 joints and the position of the joints are a pair of real numbers.

2.1 Tasks

There are two tasks below.

2.1.1 Lifting

In the first, you are to use only the data described above, that is, in this part, you use only the (2D, 3D) pair to train the data.

1. Model. Create a model with a signature such as

   ```
   class LiftModel(n_blocks=2, hidden_layer=1024, dropout=0.1, output_nodes=15*3)
   ```

2. Report. We are interested in minimizing the so-called mean per joint position error (`mpjpe`). This is the average Euclidean distance between the ground truth and prediction for all joints. An example is provided simply for reference.

   ```
   def cal_mpjpe(pose_1, pose_2, avg=True)
   ```

   Report the error you have obtained using your training. We will evaluate this for the test data set.

3. Train. Use the data to train the neural network.

```
def run_epoch(epoch_no, data, model, optimiser, scheduler, batch_size=64, split='train')
```

Q2. Provide details of your training. What experiments did you conduct? Show charts.

4. Evaluation script. Save your model in the file `liftModel`. Something like (for `pytorch`)

   ```
   torch.save(model.state_dict(), filepath)
   ```

   Provide a script `eval.py` which takes your model weights, and from the current working directory, a test datafile `data_test_lift.pkl` (similar format as train), runs the model and the average mpjpe (and also average loss if you are doing something different). Submit this along with `training.py` plus the usual reflection essay.

2.1.2 Weak supervision

Often it is difficult to get ground truth 3D data and it can also be more error-prone. In this part, you will pretend you do not have any 3D data for training (but you have it only for computing the mean joint error).

This part will be revealed in Lab 7(b)