CS 763/CS 764: Lab 7(a)

Imputing 3D from 2D

- Announced: 04/06. Due: 04/13 5PM
- Note: This is a group 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.

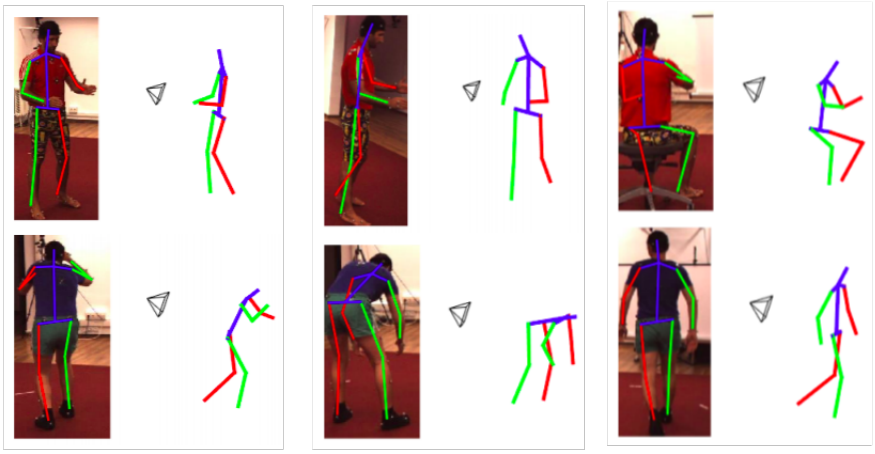

Figure 1: 2D input skeletons are overlaid on RGB images. Shown adjacent to each image are (a) 3D rendering of the corresponding skeleton, and (b) a camera configuration. If the center of the camera is as shown, the resulting RGB image would be the one shown.

2 The data and the model

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

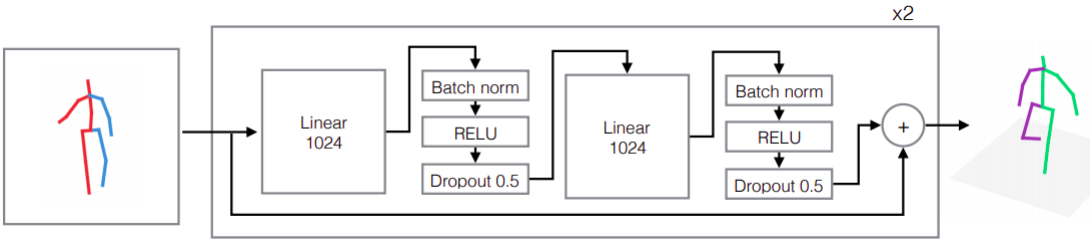

Figure 2: The network is composed of two so-called residual blocks made up of linear layers. The figure shows one such residual. Missing in this picture at the input stage is a linear layer that takes the input skeleton to the 1024 node linear layer. Also missing is another linear layer at the output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

Observe that the data set provided here has more information compared to earlier.

- `'joint_2d_2'`: This key points to several 2D skeletons each of which are projections of the same 3D skeleton (as earlier). Unlike `'joint_2d_1`, this pertains to a different view, and possibly a different camera.
- `'rot', 'transl'`: How are the views related? The rotation is specified by a reshaped 1×9 vectors corresponding to a standard 3×3 matrix). Also provided is the translation as a 1×3 vectors.
- `'focal_len_2'`: This key points to the focal length of the camera corresponding to the projection in the second view.

As before

1. Model.
2. Train. Use `'joint_2d_1`, `'joint_2d_2`, `'rot'`, and `'focal_len_2'` to train. Write the function

   ```
   def run_epoch_weak()
   ```

   as done earlier. All training related code is in a **single** file called `training.py`

   Q. Provide details of your training. What loss function did you use? What experiments did you conduct? Show charts.

3. Report the mean joint error in your reflection essay as well as the standard output.
4. Reflection essay. Discuss all aspects of this experiment lucidly. This component is central for marks purposes. Comment on the deep learning method of obtaining 3D information versus your earlier lab.
5. Evaluation script. As before but see file structure below!

3 Submission Guidelines

1. The top assignment directory should include the lab submission as detailed below.

   (a) Do include a `readme.txt` (telling me whatever you want to tell me including any external help that you may have taken). Don't forget to include your honor code here (or your name and roll number). All members of a group are expected to (electronically) sign the honor code and the percentages (see below). This is a text file, not `pdf`, not `docx`.
   The `readme.txt` will contain individual contributions of each of the team members. If the assignment is finally worth 80 marks as graded by the TA, then a contribution of the form `80, 100, 60` (in sorted order of roll numbers) will result in (respectively) marks `64, 80, 48`. Do this for each question separately. A person claiming 100% is basically saying that (s)he can reproduce the entire assignment without the help of the other team members.
   Many of you are using colab. **The readme.txt will also contain a link to a share worthy colab file**.

   (b) `ReflectionEssay.pdf`: Should contain the explanation for all the questions implicitly and explicitly raised. Provide an output of a sample run. Explain what you learnt in this assignment, and how this assignment improved your understanding. What will someone who reads this gain? Can this be a blog post, which if read end-to-end someone not in your class (but in your batch) will understand?

   (c) A directory called `code` which contains all source files, and only source files (no output junk files). The mapping of code file to questions should be obvious and canonical.

   (d) A directory called `results` to store output that needs to be saved (this will typically be explicitly stated in the question).

   (e) A directory called `data` on similar lines to code, whenever relevant. Note that your code should read your data in a relative manner.

   (f) Source files, and only source files (no output junk files). Mac users please don't include junk files such as `.DS_store`. We are not using MacOS.

   (g) Create a directory called `convincingDirectory` which contains anything else you want to share to convince the grader that you have solved the problem. We don't promise to look at this (especially if the code passes the tests) but who knows? This is your chance.

2. Once you have completed all the questions and are ready to make a submission, prepend the roll numbers of all members in your group to the top assignment directory name and create a submission folder that looks like (for group but you get the idea) this
   `130010009_140076001_150050001_lab0X_description.tgz`

   Please stick to `.tar.gz`. Do not use `.zip`. Do not use `.rar`

3. Your lab submission folder should look something like this:

```
130010009_140076001_150050001_labXX/
├── ReflectionEssay.pdf
├── code
│   ├── training.py
│   └── eval.py
├── data
│   ├── data_train.pkl
│   └── data_test.pkl
├── results
│   └── bestModel.pt
├── convincingDirectory
└── readme.txt
```

4. Submission.

   (a) Submit on **Moodle** at the course **CS-763**.

   (b) The lexicographic smallest roll number in the group should submit the entire payload (with all the technical stuff).

   (c) All other roll numbers submit only `readme.txt` as discussed above.