# Assignment 2: 2D Human Pose Estimation

CPSC 533R Visual AI
by Helge Rhodin

This assignment focuses on computer vision aspects of deep learning. The goal is to infer the 2D human pose in a given image, the pixel location of the human body parts as shown in Figure 1. This assignment will compare three approaches. Straight regression with a black-box neural network, per-pixel classification in form of heatmap detection, and a combination of detection and regression via so called *integral pose regression* [1].

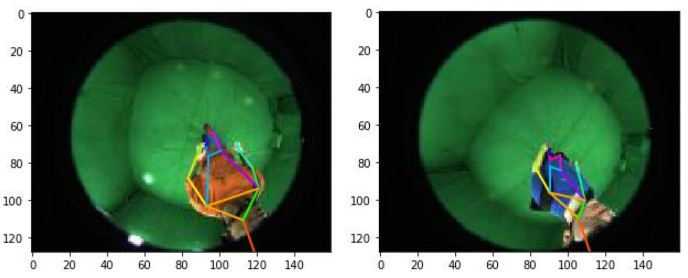

Figure 1: Egocentric images and corresponding pose annotation.

**Preliminaries.** While PyTorch supports most python expressions for defining a neural network (including control flow), using tensor operations instead of `for` loops is favorable, to facilitate parallel execution on the GPU and CPU (SIMD operations). We prepared the `assignment2_preliminaries.ipynb` notebook to exercise tensor operations. This part is not graded but highly recommended to prepare for the main task.

**Dependencies.** Install the conda package manager following the attached `readme.md`.

**Main tasks.** We provide a dataset of egocentric images and corresponding 2D pose in form of a pytorch dataset, a pre-defined neural network, and plotting utility functions. The data stems from the publication [2]. We also provide training code for regressing 2D pose directly from the image. Its training is very similar to what you implemented in Assignment 1. Make sure that the code we provide runs on your setup and make yourself familiar with the code we provide before moving on to the subsequent tasks.

- **Task I:** Implement human 2D pose detection through heatmap estimation [3]. The network must output a stack of $N = 18$ heatmaps, one for each joint of the human skeleton. It is your task to train the network by matching the output with reference heatmaps. Use tensor operations instead of for loops whenever possible. The notebook provides code for the heatmap display and for reading out the predicted keypoint location as the arg max at inference time. Proceed step-by-step, e.g., start with the processing of a single heatmap (as in the preliminaries notebook), later a stack of them, and then a mini batch of stacks. Entry points for your code are marked with `# TODO`.
- **Task II:** Implement human 2D pose regression through heatmap integration [2]. In this case, the heatmap prediction must be followed by a custom integration layer that computes the expected location over the heatmap. The loss on the resulting 2D pose can then simply be the MSE to the ground truth joint location, as for the direct regression methods we provide. The notebook contains a skeleton for every

step. Note, the integration layer must not use a for loop to gain full points. Make sure that your network trains from scratch, not from the result of **Task I**.

- **Task III:** Compare the three approaches (regression, classification, integral regression) in terms of the mean squared joint position error (MSE) on the validation set. Which of them attains the lowest error? You don't have to train for ages, but make sure that you train all models for at least five epochs. Comment on whether convergence speed (attaining a decent result early on) or overall accuracy (best result after training all methods for sufficient time) is the main factor in your setup.

**Hints and comments.**

- The lecture slides provide instruction of how to use pytorch with GPU support on UBC servers. Google colab with GPU acceleration enabled works too.
- It is easy to get lost when dealing with high-dimensional tensors. Break your implementation into individual steps and make sure that each of them works by printouts of tensor dimensions and tensor values as an image.
- By default, tensor operations are element wise, e.g., `A + B` generally assumes the same dimensions for `A` and `B`. So called *broadcasting* is a powerful operation that kicks in if the dimensions differ. Make yourself familiar with that concept.
- If your are unfamiliar with tensor operations, implement your function first using loops and turn it into equivalent tensor operations once your loop version works.
- Aggregation functions, such as `torch.sum(mat)`, compute a single value for an entire tensor. One or multiple dimensions can be specified to apply the operation only along that axis, e.g., `torch.sum(mat, dim=-1)`. Positive and negative indices are permitted, with negative ones counting 'from the right'. While this operation returns a tensor with dimension `dim` removed, it is often useful to specify `keepdim=True` to maintain that dimension in the output (with size 1).
- `torch.linspace` and `torch.meshgrid` are useful tools for creating sequences of numbers.
- We utilize a helper function `dict_to_device` that moves all elements of a dictionary to and from the cuda GPU.

**Bonus.** Solutions to the bonus task give up to one bonus point that can offset points deduced in the main tasks. The maximum number of points is still capped at 10.

- Implement DensityNetworks (see lecture on probabilistic modeling of regression) on top of integral pose regression. Estimate the conditional variance as the variance of the predicted heatmap.
- Optimize your model and train it for exactly 10 epochs. Report your MSE number on Piazza before the deadline, your last reply counts (we will create a piazza post). You get one bonus point if within a 3% margin on the leader. The top-three will receive one additional bonus point.

**Submission.** Once finished, submit your jupyter notebook on Canvas. The jupyter notebook that you submit must contain the cell output from a clean execution (restart kernel and run all cells sequentially) to ease grading. Do not submit downloaded datasets and other temporary files.

## References

[1] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei *Integral Human Pose Regression*, ECCV, 2018.

[2] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt, *EgoCap: Egocentric Marker-less Motion Capture with Two Fisheye Cameras*, SIGGRAPH Asia, 2016.

[3] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. *Efficient object localization using convolutional networks.* CVPR, 2015