Single Person 2-D Pose Estimation from RGB images using Convolutional Neural Networks

Akhil Bommadevara
abommad@ncsu.edu

Ishan Bhatt
ivbhatt@ncsu.edu

Aishwarya Seth
aseth@ncsu.edu

Code : https://github.ncsu.edu/ivbhatt/SinglePerson2DPoseEstimation-DLNN-ProjF

ABSTRACT

Human Pose Estimation (HPE) in two dimensions is usually done by locating a few key-points (usually joints) of the human-body and then connecting them appropriately to estimate the pose of the person. In single person HPE, the classical approach to finding key-points is to regress the key-point coordinates. Enhancements to this approach were made until Convolutional Pose Machines (CPMs) [1] showed that auto-encoders could be used to generate key-point heat-maps & then we could find the center-of-mass of those heatmaps to locate the corresponding keypoints. They also demonstrated how heatmaps can be improved by stacking auto-encoders on top of each other. In this work, we build a single-person pose estimation system taking inspiration from CPMs as well as other heat-map based approaches such as Stacked Hourglass Models (SHMs) [2][3]. We train our system on a subset of the MPII Human Pose dataset[4] and present our results in this report.

Keywords: Human Pose Detection

I. MOTIVATION

In human pose estimation (HPE), we try to develop systems to localize body key points to recognize the posture of an individual in an image. In general sense, an HPE system may be expected to report postures of multiple people (in a single frame), work with different imaging techniques (such as 3D depth-sensing cameras) and deal with occlusions as they may arise in the real world. However, for this project, we developed an HPE that:

- Takes RGB images as input.
- Assumes each image to have a subject person without significant occlusions.
- Detects the visible key points of the subject person.

HPE systems have found a wide range of applications in sectors such as:

- Commercial: Person tracking, Activity recognition [5]
- Entertainment: Animation, Virtual Reality[6]
- Medical: Physical Therapy, Assisted Living[7]

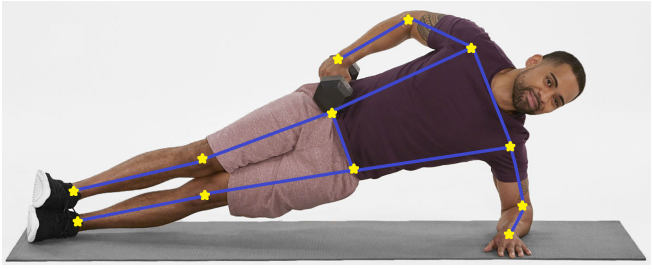

Fig 1: *Sample output (image for illustration only)* [Raw image: unsplash.com]

As shown in Figure 1, our system should be able to locate key points from a picture of a person. Using those key points, it should be able to trace the pose of the person.

The main challenges we faced include:

- Occluded body parts (We removed such cases from the dataset.)
- Truncated body parts (System is be able to handle cases where only one (or two) parts of the subject person are not visible in the picture)
- Other people in the background (If other background people are sufficiently blurred, our system is be able to filter them out)

II. DATA

We use a subset from the MPII Human Pose[4] dataset for training and testing our system. This dataset contains approximately 25K images of about 40K people. For each person, they have annotated 15 body key-points viz:
left -ankle, knee, hip, shoulder, elbow, wrist;
right -ankle, knee, hip, shoulder, elbow, wrist;
head,thorax(upper-neck),pelvis
From these 15 key-points, we drop the head, thorax and pelvis key-points because:

- We noticed that in some instances, the annotations said that the head-keypoint was not visible even though we could clearly see the person's face. Due to such discrepancies in the dataset, we had to drop the head key-point unwillingly.
- Thorax and Pelvis are generally centered between the shoulders and the hips respectively, hence we decided to remove them as somewhat redundant features.

After this, we are left with 12-body key-points (yellow) after dropping the 3 key-points (red) as shown in Figure 2:

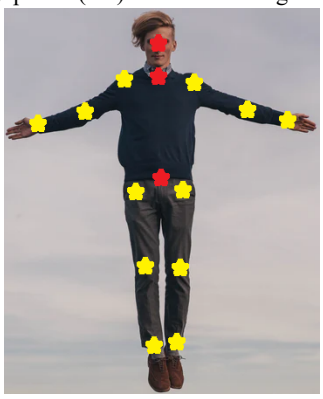

Fig 2: *Keypoint selection (red = dropped)* [Raw image: unsplash.com]

It is not guaranteed that each image has a clear subject person. It is also not guaranteed that each person is clearly visible. Therefore, we must perform some cleaning of this data before we can start training our models.

Out of the 40K people present in the images, only 24K are sufficiently separated from each other. The curators provide bounding-boxes to help locate such instances.

To ensure that we only keep the bounding boxes where the subject person is properly visible, we remove all the bounding boxes where at least twelve of our key-points are not visible. After this we were left with only about 4K bounding boxes where the subject-person was reasonably visible.

However, even after this, we found that some images had multiple overlapping that could potentially confuse our system. Hence, we manually deleted those images where the subject-person was not clear. Figure 3 loosely illustrates the kind of images we kept versus the kind of images we dropped.

Fig 3: *Manually Cleaning of images (images in the lower row were dropped)*

After cleaning the images, we were left with only about 2.5K high-quality images to work with. We performed data-augmentation with the following operations in random-order:

- 50% Flip Horizontal
- Upto 10% random-scaling
- Upto 5% random-translation
- Upto 5° random-rotation

We applied this augmentation technique to get about 8 copies for each of the 2.5K images. During the augmentation, some of the key points went out of the frame and hence we had to drop those augmented images. Therefore, instead of having about 20K images, we had only 16K images in our final dataset. Figure 4, illustrates the augmentation scheme.

Fig 4: *Illustrating Data Augmentation scheme*

We use about 75% of this data (~12K images) for training while the rest 4K images are reserved for testing. **In summary, we use about 12K images each containing a reasonably-visible and clear subject-person to train (and cross-validate) our system.** Another ~4K such images are reserved for testing.

III. MODEL & TRAINING

To generate the annotations (heatmaps) for our models, we put a gaussian distribution at each key-point coordinate with a peak = 1 and a standard deviation of 15. We have resized each image to (128, 128) since that was the smallest size we could use while still preserving the image resolution. Figure 5 illustrates our annotations. We have 13 heatmaps (1 for each key-point and 1 for background). We combine the 12 keypoint-heatmaps to better visualize the annotations.

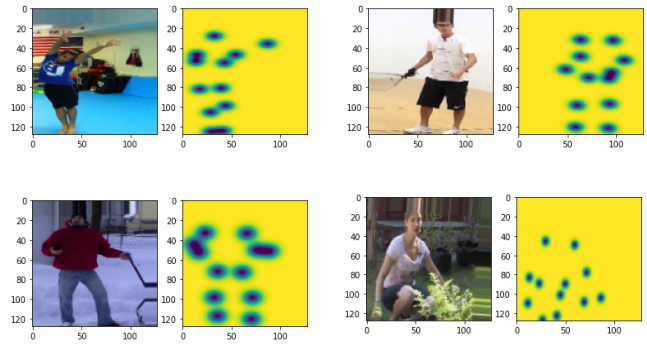

Fig 5: *Sample data-points (12 keypoint heatmaps compressed into 1)*

Our system was inspired by the success of CPMs[1] and SHMs[2].

We describe our model in the following bullets:

- Each hourglass module is essentially an autoencoder that takes an X-channel input and produces a Y-channel output. Y = 13 (12 key points + background) for each of our hourglass modules. For the first hourglass module, we have set X = 3 (rgb image) and for all subsequent ones, X = 13 + 3 (previous heatmaps + rgb image)
- Each hourglass module can have k downsampling steps and k upsampling steps. Value of k is the same for each of the hourglass modules. We found that k=2 worked best in our tests.
- Each hourglass module produces a 13 channel output heatmap.
- The first hourglass module is given the image as an input and all other subsequent hourglass models are given a concatenation of the input image and the heat maps generated from the last hourglass model.
- We find the center of mass for each channel to find the coordinates for each keypoint.

The schematic for our model is presented in the following figure.

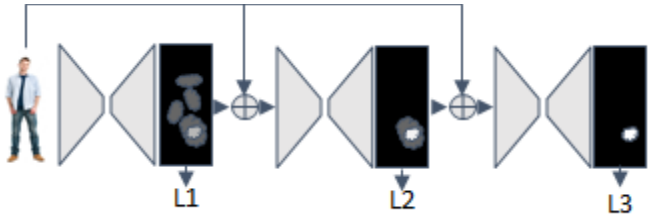

Fig 6: *Model Architecture (Inspired from CPMs[1] and SHMs[2])*

We use 12K train images and use 10% of them for cross-validation. We use keras with tensorflow backend to implement our model. We present the typical architecture of an hourglass module and our model in the following figures.

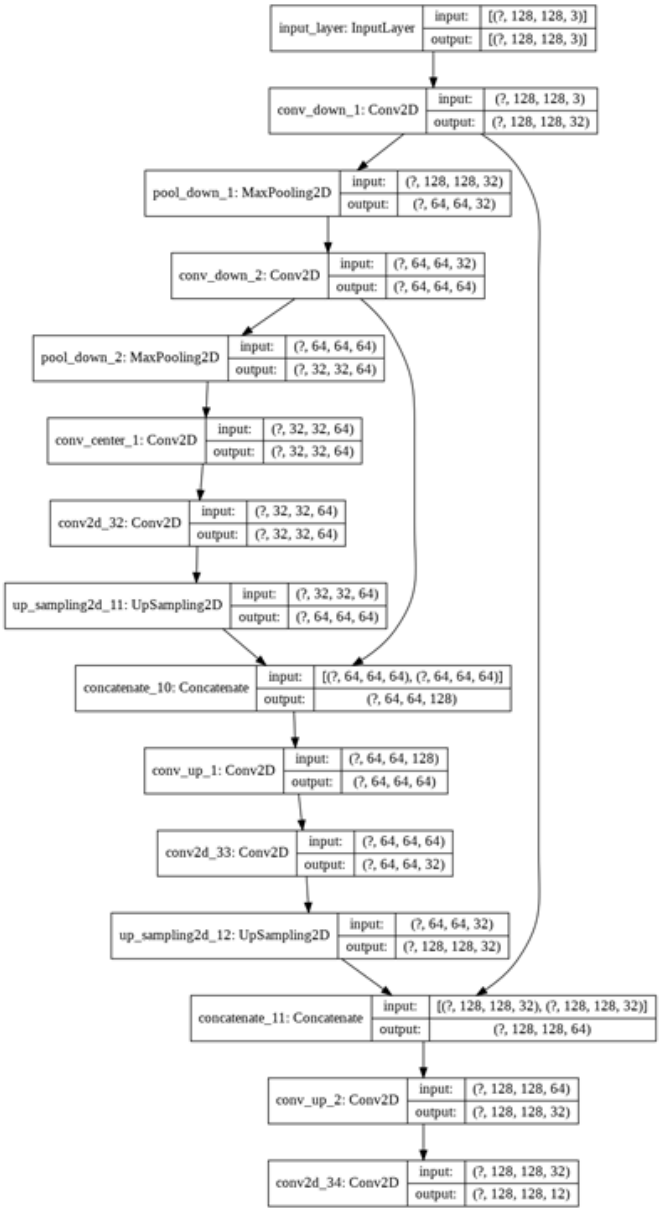

Fig 7: *A typical convolutional hourglass*

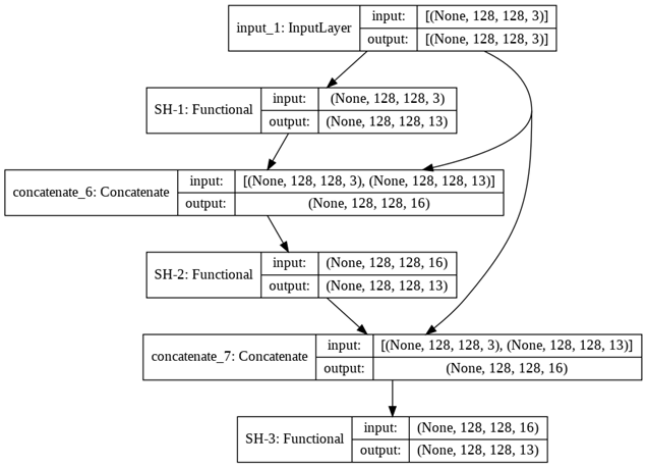

Fig 8: *Our model (with the Hourglass abstracted as SH-1, SH-2 & SH-3)*

Our final model had about 250K trainable parameters. It took about 2 hours to train for approximately 100 epochs on Nvidia Tesla T4.

We settled on stacking three hourglass models as we found very diminishing returns as we went on stacking more and more hourglass models after three. We present the training graphs of the best mode in figure 9.

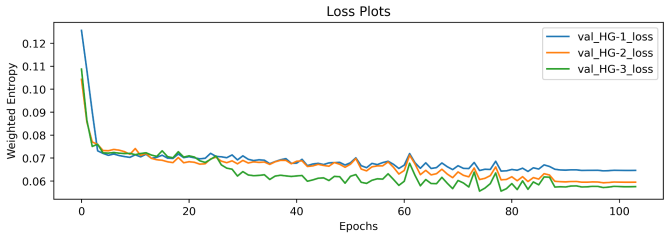

Fig 9: *Model Training logs*

We report the validation-losses calculated for the heat-maps predicted at each of the hourglass modules. We notice how there is a large difference between the performance when we go from hourglass 1 to hourglass 2. However, going from hourglass 2 to hourglass 3, there is a very small gain in performance. The performance gain obtained as we go from hourglass 3 to hourglass four is even smaller (almost indistinguishable). For that reason, we stopped our training at a model with 3 hourglasses stacked on top of each other.

We notice that, in the heatmaps, there is a severe class-imbalance between the key-point classes and the background. To address this problem, we also implement a custom loss function. Our custom loss function is weighted categorical cross-entropy which gives slightly less importance to the keypoint classes than the background class. In short, we weigh a misclassification among keypoint classes half the importance as we give to a misclassification in the background class.

$L_o(h, y) = -w*y*\log(h)$ | where

w = [1 1 1 … 1 0.5]

h = one-hot predictions

y = one-hot labels

Also, we weight the losses produced at the different depths in the stack. We weight the output at the last level twice as much as the intermediate losses.

IV. RESULTS & EVALUATION

Our final system manages to detect about 82% of the key points within half the length of the head of the person whose keypoints are being detected. **This metric is known as PCKh@0.5 (read: Percent of head-normalized Correct Key Points @ 0.5).**

The state of the art on this dataset was achieved in 2019 with a similar stacked autoencoder model [3], which is about 10% better than ours. However, it should be noted that we have significantly constrained the scope of this project & therefore our results are not directly comparable to the other methods we mention here.

Many authors present their results by breaking their results in parts, that is they report the PCKh for each part individually. Such an approach helps the reader understand if a particular model is better at detecting a particular part. A comparison between our model and other notable models is shown in Table 1 given on the next page.

From Table 1, we notice that our model outperforms the SotA in locating hips! We think this is because the other models also have scenarios where the hip of the person is not visible whereas we only consider the scenarios where the hip of the subject person is clearly visible in addition to the entire person. This requires that the hips are almost always found in the center of the image & the model may have easily learnt to over-fit those particular heatmaps.

Table 1: *Result Comparison with baseline (PCKh@0.5)*

	Ours*	CPMs[1]	SHMs[2]	SotA[3]
Head	-	0.97	0.98	0.98
Wrist	0.63	0.84	0.87	0.89
Elbow	0.79	0.88	0.91	0.93
Shoulder	0.87	0.95	0.96	0.97
Hip	0.95	0.88	.90	0.92
Knee	0.89	0.83	0.87	0.89
Ankle	0.77	0.79	0.83	0.87
Total	0.82	0.89	0.91	**0.92**

* Our results cannot be directly compared with these numbers because our scope is limited to images where the entire subject person is clearly visible. None of the other works make that simplifying assumption.

From Table 1, we can see that **our final best performance falls about 10% short of the state-of-the-art model** which uses a much deeper model with more trainable parameters trained for a longer time.

Another interesting thing which we notice in our result comparisons is how all models (including our) struggle to locate the edge-joints of the human body. In other words, we notice how each model struggles in locating the ankles and the wrists; do a little better when detecting the elbows and knees; find the hips and shoulders relatively easily!

Even though we would like to improve performance across the board, these kinds of errors imply that the neural networks are indeed trying to figure out the pose of the person & not overfitting on some artifacts present in the data.

Another interesting thing to notice about these stacked auto-encoder models is the relative improvement of the heatmaps as we go deeper into the stack. Figure 10 shows the relative improvement of the heatmaps as we go deeper into the hourglass-stack.

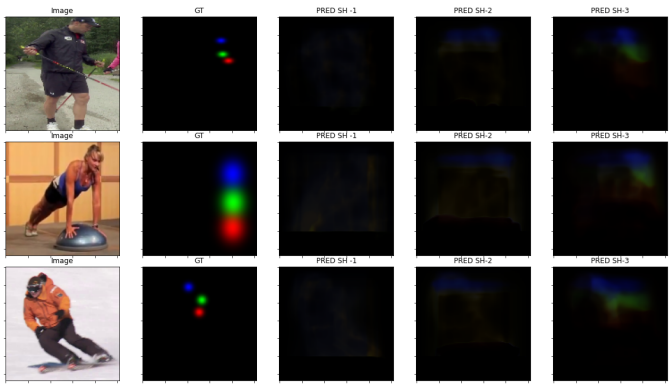

Fig 10: *Relative improvement of the heatmaps as we go deeper*

We notice how heatmaps predicted by the first hourglass are refined by the second hourglass and the third hourglass refines the heat maps produced by the second hourglass.

Finally, Figure 11 shows some sample predictions generated by our system:

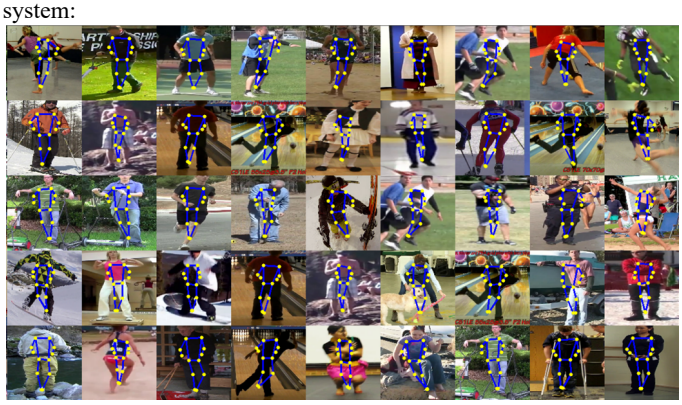

Fig 11: *Sample results*

V. CONCLUSION

This team has been able to successfully build the system it set out to build. The resultant system can estimate poses reasonably well. The discrepancy between the SotA results and ours is justifiable since we trained our model with about a quarter of the data and for about one-sixth the time as the State-of-the-Art model.

Our method could manage to detect over 80% of the key points in unseen images and that is indicative of good generalizable performance.

We noticed that most of our errors are arising in images where the subject person is not directly facing the camera, which is an area of improvement for us.

Also, to convert this system from single-person to multi-person would require to have a light-weight person-detection module that could feed this model with person bounding-boxes.

REFERENCES

[1] S.-E. Wei, et. al.. “Convolutional pose machines” in CVPR, 2016

[2] A. Newell, ‘Stacked Hourglass Networks for Human Pose Estimation”, arxiv, July 2016

[3] Zhang, Hong, et al. "HPE with spatial contextual information." arxiv, 2019

[4] M. Andriluka, et. al., “2D Human Pose Estimation: New Benchmark and State of the Art Analysis”, CVPR, 2014

[5] A. Yao, et. al., “Coupled action recognition and pose estimation from multiple views,” Int. J. Comput. Vis., Oct 2012

[6] M. B. Holte, et. al., ‘‘Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments,” IEEE J. Sel. Topics Signal Process., Sep 2012

[7] Spagnolo and Paolo, ‘‘Proceedings IEEE conference on advanced video and signal based surveillance. AVSS 2003,’’ in Proc. IEEE Conf. Adv. Video Signal Based Surveill., Jul 2003