

Bachelor Thesis  
Artificial Intelligence

**Radboud University**



Phosphene representation using  
Triplet Pose estimation

**Author**  
Yousif Eldaw  
S4437403  
y.eldaw@student.ru.nl

**Supervisor**  
dr. U. Güçlü  
Donders Institute,  
Radboud University  
u.guclu@donders.ru.nl  
**Daily supervisor**  
drs. J. de Ruyter  
van Steveninck  
Donders Institute,  
Radboud University,  
j.deruyter@donders.ru.nl



February 05, 2021

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Related work</b>	<b>4</b>
<b>3</b>	<b>Model design</b>	<b>5</b>
3.1	Architecture . . . . .	5
3.2	Layer explanation . . . . .	6
3.3	Training procedure . . . . .	7
<b>4</b>	<b>Dataset</b>	<b>8</b>
<b>5</b>	<b>Results</b>	<b>9</b>
<b>6</b>	<b>Discussion</b>	<b>12</b>
6.1	Training . . . . .	12
6.2	Images . . . . .	12
6.3	Future research . . . . .	13
<b>7</b>	<b>Conclusion</b>	<b>14</b>
<b>8</b>	<b>References/Bibliography</b>	<b>15</b>
<b>9</b>	<b>Appendix A</b>	<b>17</b>
<b>10</b>	<b>Appendix B</b>	<b>19</b>

## Abstract

With stimulation of the visual cortex, we can generate the perception of so-called phosphenes, which are points of light perceptible without any light entering the eye. Through the usage of these phosphenes we can partially restore the vision of blind people by showing images on their visual cortex. This phosphene vision, however, is limited compared to regular vision due to having lower resolution and no colour contrast. The limitations mean that the images we want to stimulate onto the visual cortex need to be pre-processed, this is done so we only express the most important information, making the information more understandable. For the human body, a representation in phosphene vision can be made using the locations of their joints to create a skeleton.

I present a model which uses triplet representation, a way of representing people using more coordinates per joint than a regular representation which provides more information about the dimensions of the human body. Using the triplet model, I was able to get more information about the shape of the human body. However, this information does not necessarily translate well into phosphene vision due to noise and research into the usability for blind people remains to be tested.

## 1 Introduction

People with impaired bodily senses partially lose ability to socially interact and participate in society as well as healthy individuals do, causing further complications in their life[1]. For this research, we are looking at those who have lost their vision and how we can help them using AI. Research has yielded solutions, like glasses or surgery, to help some who suffer from partial vision loss, however we have yet to do so reliably for those who have complete vision loss originating from extensive damage to their eyes or visual tract and cortex in the brain.

A promising technique for the restoration of vision is prosthetic vision through the usage of phosphenes[2][3]. A phosphene is the phenomenon of seeing light through stimulation of the visual cortex, without light entering the eyes. Such a technique allows blind people with a working visual cortex to see the stimulated light points. However, this technique has a low resolution and does not allow for colour-contrast, which means the images it can project onto a patient's brain are limited[4]. This causes a need to pre-process the images so unimportant factors are removed and only the most salient information is shown, which can be done using neural network models brought forth through artificial intelligence.

My research will focus on pose estimation and will try to find a more accurate portrayal of people than traditional pose estimation does when using phosphene vision. The aim of my research will be to implement a neural network model which estimates the human pose based on a picture and joint estimations and will output a triplet joint estimation. Using the predicted triplet points, a filter will be applied to turn the predicted human model into phosphenes. The accuracy of the triplet estimation will be considered against the estimate of the initially estimated joints.

## 2 Related work

Phosphene stimulation has been researched for a long time[5], and implementations seek to portray salient information from an image. In order to not oversaturate the stimulated image with unnecessary information, one would need to be able to recognise the objects in the image and choose the most important ones. The recognition of objects and selection of the important ones in an image can be done using object recognition. From the new image containing only the important objects the most salient information can be extracted to provide a less detailed but informative image that still allows people to recognise the objects. However, just object recognition is not enough to extract this information. For human objects, information is needed on important body parts and their position even when they overlap with one another or are hidden behind clothing or objects. Research into what type of pre-processing is best to acquire this information about humans in pictures is limited so far.

Pose estimation[6][7] is used to estimate the pose and joint locations of people in an image. Using the information about someone's pose, you could better attempt to infer a person's current activity as well as occluded body part locations. As the application of pose estimation as a pre-processing technique for phosphene vision has not been researched so far, my idea is to apply human pose estimation to create a phosphene representation of people in an image. Most current implementations of human pose estimation are focused on estimating singular joint locations[8]. However, there are techniques which could help better estimate the body's shape. One could use edge detection to try and detect the exact contours of the body based on the estimated joint locations. However, for the problem of occlusion of body parts and body shape under clothes and this would still pose the same problem as it would without joint locations. Another way could be using stereoscopic images at eye's width to reconstruct a 3D model of the people in the image[9]. Using such a technique we would no longer need to predict the joint locations in the first place; but this would be more computationally expensive, and datasets are non-existent at this point. In my attempt to better represent the body shape I will use triplet joint estimation wherein additional datapoints are estimated around most joints to show the shape around it. My hope is that these extra datapoints could help provide more information about the body dimensions allowing me to create a more accurate exterior of the human body compared to the singular predicted joint locations and allow for a better phosphene representation.

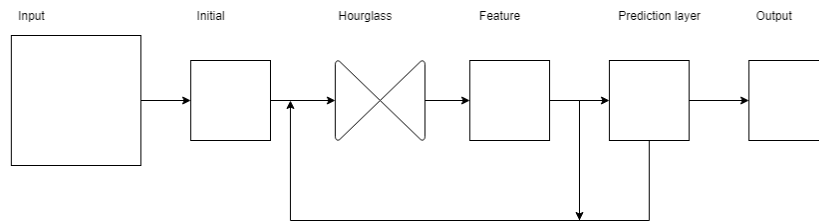
This leads me to my research question:

*Does an implementation of triplet joint representation provide a usable model for phosphene vision?*

## 3 Model design

### 3.1 Architecture

For the model I have made in this research<sup>1</sup> two sub-models had to be trained. The first is the joint estimation model which uses only an image as input and whose output is 16 estimated heatmaps showing the most likely joint locations, the second is the triplet estimation model where the input is the image as well as the 16 ground-truth heatmaps, its output is 26 estimated heatmaps. Due to the similarities between the usage of the two models both use the same implementation of the hourglass model[10], which for this research was based off a mixture of several recent implementations[11][12][13].



**Figure 1:** *The network layout.*  
*The input and output shapes are shown in appendix A*

The model uses a series of layers, which in my implementation will be referred to as the initial, hourglass, feature, and prediction layer, to reduce the dimensionality and predict the data. These layers create feature maps and predict joint locations and their underlying relationship. The learned relationship between joint locations can help ensure the estimated joint locations are connected in a normal manner. All of the aforementioned layers, except for the initial layer, are part of a stack of layers. The first layer of this stack is the hourglass layer, followed by a feature layer and finally a prediction layer. The model uses multiple stacks in a row where the successor stack takes the output of the feature and prediction layers from the previous stack and adds it to its input, using the added information to try and make better predictions. This leads to a model which has an initial layer followed by a series of stacks, the number of which depends on whether the model is used for joint estimation, in which case it uses 8 stacks, or triplet estimation, in which case 5 stacks are used.

---

<sup>1</sup>The full code is available at <https://github.com/yeldaw/image-to-phosphene>

### 3.2 Layer explanation

The model uses residual[14] layers extensively. They consist of three convolutions with each convolution being preceded by batch normalisation and a relu unit. The first and third convolutional layers are 1x1 convolutions, the second is a 3x3 convolution. The outputs are the features, which were present in the input, after background information has been removed. However, the removal of background information also removes spatial information about feature location, so at the end the input of the residual layer is added to the output value to retain this spatial information while still highlighting feature information.

The initial layer receives either a 3x256x256 for the joint estimation model or a 19x256x256 tensor for the triplet estimation model, it uses a 7x7 convolutional layer with a stride of 2, this outputs 64 units creating a 64x128x128 tensor which is batch normalised and passed through a relu unit. This is passed through a residual layer, a 2x2 max pooling layer and another two residual layers, creating a 256x64x64 tensor, which is a constant input and output size between all following layers except for the output of the prediction layer.

An hourglass layer consists of several layers, with one being a special layer. The hourglass layer receives both the data as well as a number which specifies whether it should recursively create hourglasses or not. If the number is greater than 1 it assigns a hourglass layer to its special layer and subtracts 1 from the number before passing it to that hourglass layer, if it is not greater than 1 it assigns a residual layer to the special layer. For this model, the first hourglass layer always has 4 as its supplied number. An hourglass layer starts with a residual layer, followed by 2x2 max pooling, a second residual layer, the special layer, a third residual layer and finally an upsampling layer. The output of the upsampling layer is appended to the value of the first residual layer.

The feature layer uses a residual layer, followed by a 1x1 convolutional layer, batch normalisation and a relu unit. Finally the prediction layer uses a 1x1 convolution to output either 16x64x64 or 26x64x64, based on the model used.

If this is not the final stack in the model, both the features and the predictions are passed through another 1x1 convolutional layer and then added to the output of the initial layer, after which it is used as input by the next stack. If it is the final stack in the model all the predictions made for each stack are returned and the loss is calculated over each stack prediction individually compared to the ground-truth heatmaps.

### 3.3 Training procedure

For the training 500 images were chosen, with batch size 4, at random from the training images per epoch to reduce memory constraints. For the test set 50 images were used for every epoch. After every epoch 500 more random images were chosen for the training set, and 50 for the test set.

The weights for the models were initialised using Xavier initialisation, the parameters were updated using the adam optimiser. For the loss the MSE-error was used. The model training was done on a computer running MXNet version 1.5.0, using an NVIDIA RTX 2060 with CUDA driver 9.2.

Initially the output for the joint location estimation model was simply the joint locations. However, initial testing had shown that simply training the model to predict the joint was not successful. It could learn the training data well but would overfit and would not perform well at all on new data. The idea was to instead use heatmaps. Heatmap generation was done by assigning weighted values on a 64x64 grid according to their location in relation to the actual joint's location, with no more values being assigned for any blocks more than 5 pixels away. This encouraged the model to find locations close to or on the model without punishing close guesses as harshly as ones further away.

For the triplet model, the input is also the image, however it also uses the heatmaps used as the ground truth of the joint model to facilitate the learning of the relationship between joint positions and body width. This means that the input for the triplet model is a 19x256x256 tensor. For its output, it tries to estimate 26 heatmaps to compare to the ground-truth heatmaps generated in the same way as with the joint model.

Due to a low loss at low heatmap values a vanishing gradient problem occurred. A sigmoid activation layer was attempted to fix the values between 0 and 1 however this caused the neural network to have problems learning. A relu activation layer was used instead like suggested in the papers, and the weighted values of the heatmaps were artificially increased a thousandfold to help with learning and to avoid the vanishing gradient problem.

## 4 Dataset

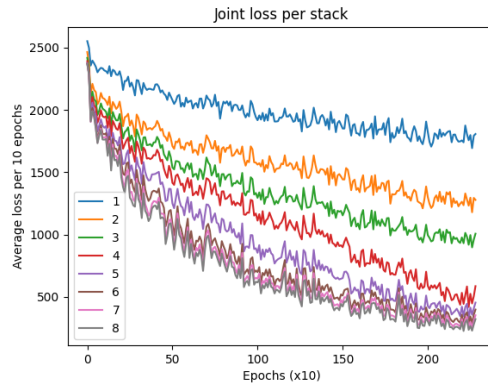
To be able to train the joint estimation properly a dataset was required that annotated the joints in the body, for the triplet estimation a dataset was required that had triplet joint annotation that fit those of the first dataset. The dataset used for the joint estimation was the MPII Human Pose dataset[15], a dataset containing pictures of people in many different poses and performing different activities. This dataset supplied joint locations for all of its images as well as information about which of them were visible or were not visible which was included as a visibility tag per joint. The dataset for the triplet estimation was the augmented dataset of the MPII Human Pose dataset[16], this dataset was adjusted to contain two additional locations for most joint points that could be linked to the joint locations of the original dataset.

<b>MPII</b>	<b>Augmented MPII-1</b>	<b>Augmented MPII-2</b>
Head top		
Upper neck	Right neck	Left neck
Shoulder(left/right)	Medial shoulder	Lateral shoulder
Elbow(left/right)	Medial elbow	Lateral elbow
Wrist(left/right)	Medial wrist	Lateral wrist
Thorax		
Pelvis		
Hip(left/right)	Medial hip	Lateral hip
Knee(left/right)	Medial knee	Lateral knee
Ankle(left/right)	Medial ankle	Lateral ankle

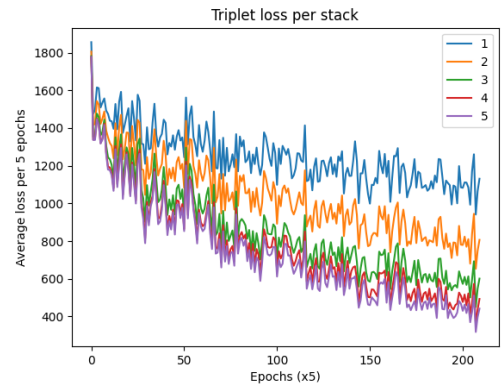
To make use of this dataset in my model, the images had to be preprocessed. Only images supplying all 16 different joint locations of a person as an annotation were kept, to extract the people in these images the images were cropped around a person's body. Due to the cropping, only one person was in a picture. This was not a problem as the idea of the model was to only apply to a single person at a time. As joint locations do not show the extremities of the body, a padding of 50 pixels had to be applied around every person's smallest and largest dimensions to ensure it would capture as much as possible of their body. The images were scaled so the largest dimension was 256 pixels, while the other dimension was scaled appropriately. Once this was done, the images were rotated and moved around in the grid to augment the dataset and simulate more data. The joint locations were all adjusted accordingly so they still showed the correct location, as was done for the triplet dataset. In the end a white-padding was applied to ensure the images were 256x256 regardless of the location of the image.

## 5 Results

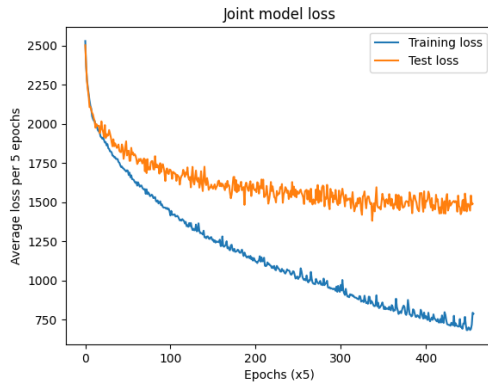
These graphs show the loss, unless otherwise specified, each point is the average over 5 epochs.



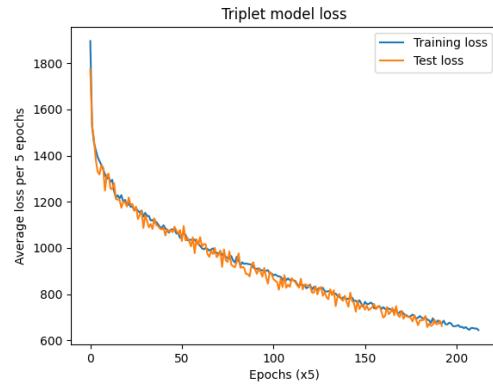
**Figure 2:** The loss for every stack of the joint estimation model over the course of the training. The average of every 10 epochs is taken here.



**Figure 3:** The loss for every stack of the triplet estimation model over the course of the training.

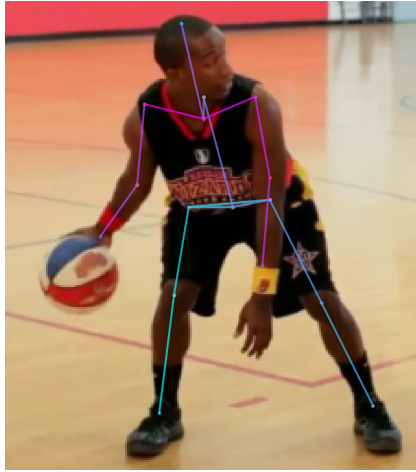


**Figure 4:** The training and test losses of the joint estimation model.

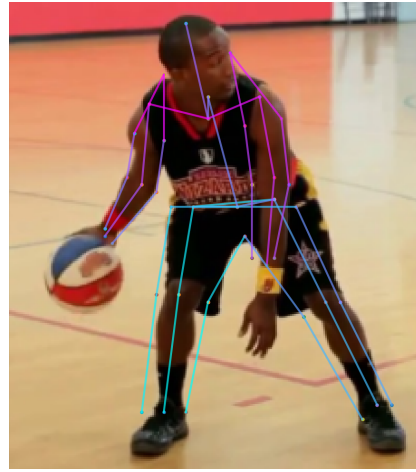


**Figure 5:** The training and test losses of the triplet estimation model.

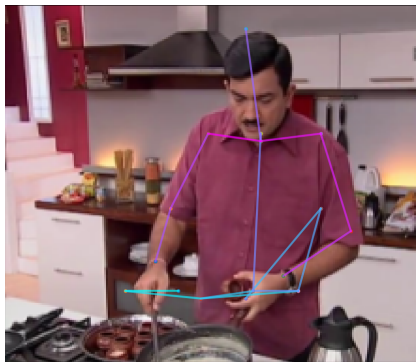
For data representation, data points in a limb were connected together. The pelvis to the thorax, the thorax to the neck, and the neck to the head. The hips connected one another, as well as to the pelvis, while the shoulders connected to the neck directly. For the triplet representation the same logic was followed, although there were no equivalent of a head or thorax and pelvis, as such those points did not connect. The datasets did not contain a phosphene representation as this was not part of the models. Phosphene results were generated by passing the predicted locations through an activation mask.



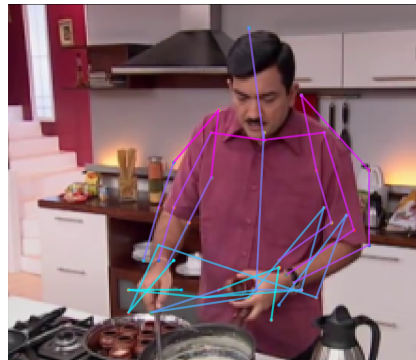
**Figure 6:** This image shows the joint estimation model applied to a validation image.



**Figure 7:** The triplet representation using the estimates of the joint estimation model.



**Figure 8:** Performance of the joint model on occluded body parts.



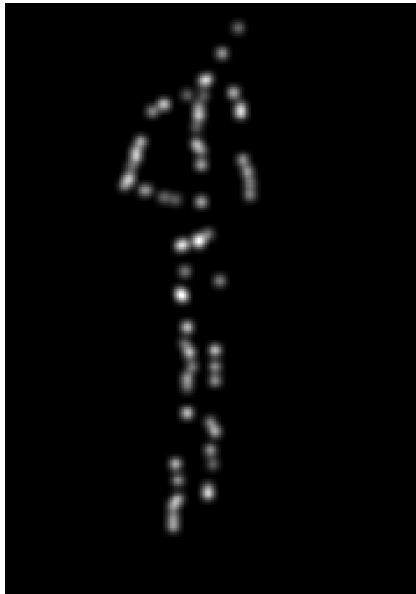
**Figure 9:** Triplet model's performance when using incorrect joint location.



**Figure 10:** The phosphene representation of the estimated joint points.



**Figure 11:** The phosphene representation of the estimated triplet points.



**Figure 12:** Phosphene representation of the joint locations of an image



**Figure 13:** Phosphene representation of the triplet locations of the same image

## 6 Discussion

### 6.1 Training

Figure 2 shows that the usage of several stacks benefits the joint estimation model’s ability to learn. There is a considerable gain in performance with the first few stacks, however, it seems that while stack 7 and 8 initially separate from stack 6, over time this does diminish and stack 6 starts performing almost equally well. This early separation shows that more stacks is beneficial to the model, or at least early on, but the later convergence suggests that stack 7 and 8 are less capable of learning new things after a while. This could be due to improper hyperparameter settings or local minima, which prevents them from learning well after a while, but could also mean more training data is required so the layers can learn how to generalise better. Figure 4 suggests that the trained model might not translate well to unseen data as the test loss does drop, but compared to the training loss it is a much slower curve. This could be due to an insufficient amount of data as well, which prevents the model from being able to predict well on unseen data. Improvements that would lead to the model’s ability to generalise better to new data might also make its 7th and 8th stack results more discernible from the 6th.

Figure 3 shows the loss per stack of the triplet estimation model. Here it shows that over time the stacks clearly separate. Stacks 4 and 5 seem to perform almost equal for the entire training sequence. This might suggest that more than 4 stacks is not significantly better and only increases training time. Figure 5 shows the triplet model’s training loss versus the test loss and it shows that the triplet model is really good at learning. It manages to constantly perform as well as the joint model, however, as both are averaged over 5 epochs each, it does not show that the real performance of the test set is less stable than that of the training set, as can be seen in appendix B.

### 6.2 Images

Between figure 6 and 7 the triplet estimation model seems to show the body’s shape better. However, due to its reliance on data from the joint estimation model, it seems to underperform when the joint estimates are not accurate. This is especially obvious in figure 8 and 9, where the joint estimation model does not know how to handle occlusion and is incapable of predicting the position of the legs. The triplet estimation model, which uses the joint estimation model’s locations, is incapable of correcting this at all using just the image.

Figure 10 and 11 show the difference in the phosphene representation between the figure 6 and 7. The triplet estimation model seems to capture the body’s width much better, however, this does not necessarily mean that all body parts are captured better in the phosphene representation. Due to the left arm’s overlap with the rest of the body and predicted points, it is more difficult to tell where it is in the triplet phosphene representation than it is in the joint

phosphene representation. More processing could be required where predicted points from underlying body parts are removed so as to not interfere with the ability to comprehend where the overlapping body parts are. Due to the lack in complexity in the joint estimations, figure 12 seems to be more easily readable than figure 13. The arms are more visible while in the triplet model they seem to blend in with the body.

### 6.3 Future research

Due to the triplet model's reliance on the joint model's estimations, future research should look into better joint estimation models and how this affects the triplet estimation model. Further training of this model while using a larger dataset might also prove useful. Furthermore, while it was outside of the scope of this research, the performance of the models when facing occlusion or multiple people in a picture is an important aspect, as it could add confusion and make parts of the human models difficult to identify in phosphene vision. While this research has shown some promising results, it was not exhaustive and only looked into whether triplet models could be viable for phosphene research. It was not an objective research into whether either is a better representation than the other. Future research could look into the benefits of each representation and attempt to mediate between them, creating phosphene representations that offer more information than the joint representation alone, while also not adding confusion due to the noise. A model that takes phosphene representation into account and attempts to add as little noise as possible to maintain clarity could help in solving the noise problem.

A combination of the two sub-models into one larger model is also possible as it could potentially have a positive effect on performance as well as resource usage than running two models back to back. A reduction in resource usage could increase its viability for real-life usage as it could mean a reduction in price as well as weight and bulkiness.

A better loss function and heatmap generation function needs to be considered. The current heatmap generation works, however, it is only an estimate and not an accurate probability map of the joint locations. While it is possible to generate such a map, the model performed worse under it due to lower loss values and a vanishing gradient problem. While the L2-loss was useful for this application, once a sigmoid layer was considered with the new heatmap generation to ensure the values would stay between 0 and 1, it would no longer converge or learn properly.

## 7 Conclusion

The research has shown that triplet models can be a viable representation of a human. They are able to add information that regular joint representation does not contain. However, it also can add confusion due to overlap of triplet joint locations. For phopsphene vision, more research is required into whether it is a viable alternative due to the added noise. Furthermore, more research is required into how this information can be used and whether it is more preferable than the traditional joint representation.

## 8 References/Bibliography

### References

- [1] Brunes A, Hansen M, Heir T (2019)  
Loneliness among adults with visual impairment: prevalence, associated factors, and relationship to life satisfaction  
<https://hqlo.biomedcentral.com/articles/10.1186/s12955-019-1096-y>
- [2] Brindley GS, Lewin WS (1968)  
The sensations produced by electrical stimulation of the visual cortex  
479-493
- [3] Bollen C, Güçlü U, Van Wezel R, Van Gerven M, Güçlütürk Y (2019)  
Simulating neuroprosthetic vision for emotion recognition.
- [4] Kime S, Simon-Shane C, Sabatier Q, Galluppi F, Benosman R (2018)  
High Temporal Sub-millisecond Time Resolution Stimulation  
Increases Performances of Retina Prosthetic Vision
- [5] Van Steveninck JR, Güçlü U, Van Wezel R, van Gerven M (2020)  
End-to-end optimization of prosthetic vision  
<https://www.biorxiv.org/content/10.1101/2020.12.19.423601v1.full.pdf>
- [6] Papandreou G, Zhu T, Chen LC, Gidaris S, Tompson J, Murphy K (2018)  
PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model  
<https://arxiv.org/pdf/1803.08225.pdf>
- [7] Wei SE, Ramakrishna V, Kanade T, Sheikh Y (2016)  
Convolutional Pose Machines  
<https://arxiv.org/pdf/1602.00134v4.pdf>
- [8] Fieraru M, Khoreva A, Pischulin L, Schiele B (2018)  
Learning to Refine Pose Estimation  
<https://arxiv.org/pdf/1804.07909v1.pdf>
- [9] Lallemand J, Szczot M, Illic S (2014)  
Human Pose Estimation in Stereo Images
- [10] Zhao H, Shi J, Qi X, Wang X, Jia J (2017)  
Pyramid Scene Parsing Network  
<https://arxiv.org/pdf/1612.01105v2.pdf>
- [11] Newell A, Yang K, Deng J (2016)  
Stacked Hourglass Networks for Human Pose Estimation  
<https://arxiv.org/pdf/1603.06937v2.pdf>

- [12] Yang W, Li S, Ouyang W, Li H, Wang X (2017) Learning Feature Pyramids for Human Pose Estimation  
<https://arxiv.org/abs/1708.01101>
- [13] Ferdinand N (2020)  
Using Hourglass Networks To Understand Human Poses  
<https://towardsdatascience.com/using-hourglass-networks-to-understand-human-poses-1e40e349fa15>
- [14] He K, Zhang X, Ren S, Sun J (2015)  
Deep Residual Learning for Image Recognition
- [15] MPII Dataset  
<http://human-pose.mpi-inf.mpg.de/>
- [16] Triplet Dataset  
<https://github.com/KRMKarr/Triplet-Representation-of-human-Body>

## 9 Appendix A

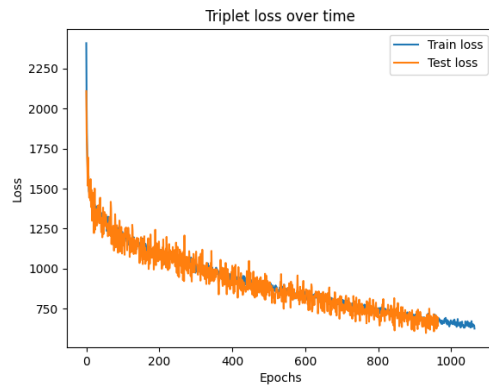
This is the joint model architecture using only one stack, also only one additional hourglass layer is used which then uses a residual layer.

The input of each layer is the output of the previous layer. Due to this being the joint model architecture, the input and output are different from that of the triplet model. The triplet joint model has input(Input) shape (4, 19, 256, 256) and output(MySequential-129) shape (4, 26, 64, 64).

Layer (type)	Output shape	Layer (type)	Output shape
Input	(4, 3, 256, 256)	Activation-31	(4, 128, 64, 64)
Conv2D-1	(4, 64, 128, 128)	Conv2D-32	(4, 128, 64, 64)
BatchNorm-2	(4, 64, 128, 128)	BatchNorm-33	(4, 128, 64, 64)
Conv-3	(4, 64, 128, 128)	Activation-34	(4, 128, 64, 64)
Conv2D-4	(4, 128, 128, 128)	Conv2D-35	(4, 256, 64, 64)
BatchNorm-5	(4, 64, 128, 128)	Residual-36	(4, 256, 64, 64)
Activation-6	(4, 64, 128, 128)	Initial-37	(4, 256, 64, 64)
Conv2D-7	(4, 64, 128, 128)	BatchNorm-38	(4, 256, 64, 64)
BatchNorm-8	(4, 64, 128, 128)	Activation-39	(4, 256, 64, 64)
Activation-9	(4, 64, 128, 128)	Conv2D-40	(4, 128, 64, 64)
Conv2D-10	(4, 64, 128, 128)	BatchNorm-41	(4, 128, 64, 64)
BatchNorm-11	(4, 64, 128, 128)	Activation-42	(4, 128, 64, 64)
Activation-12	(4, 64, 128, 128)	Conv2D-43	(4, 128, 64, 64)
Conv2D-13	(4, 128, 128, 128)	BatchNorm-44	(4, 128, 64, 64)
Residual-14	(4, 128, 128, 128)	Activation-45	(4, 128, 64, 64)
MaxPool2D-15	(4, 128, 64, 64)	Conv2D-46	(4, 256, 64, 64)
BatchNorm-16	(4, 128, 64, 64)	Residual-47	(4, 256, 64, 64)
Activation-17	(4, 128, 64, 64)	MaxPool2D-48	(4, 256, 32, 32)
Conv2D-18	(4, 64, 64, 64)	BatchNorm-49	(4, 256, 32, 32)
BatchNorm-19	(4, 64, 64, 64)	Activation-50	(4, 256, 32, 32)
Activation-20	(4, 64, 64, 64)	Conv2D-51	(4, 128, 32, 32)
Conv2D-21	(4, 64, 64, 64)	BatchNorm-52	(4, 128, 32, 32)
BatchNorm-22	(4, 64, 64, 64)	Activation-53	(4, 128, 32, 32)
Activation-23	(4, 64, 64, 64)	Conv2D-54	(4, 128, 32, 32)
Conv2D-24	(4, 128, 64, 64)	BatchNorm-55	(4, 128, 32, 32)
Residual-25	(4, 128, 64, 64)	Activation-56	(4, 128, 32, 32)
Conv2D-26	(4, 256, 64, 64)	Conv2D-57	(4, 256, 32, 32)
BatchNorm-27	(4, 128, 64, 64)	Residual-58	(4, 256, 32, 32)
Activation-28	(4, 128, 64, 64)	BatchNorm-59	(4, 256, 32, 32)
Conv2D-29	(4, 128, 64, 64)	Activation-60	(4, 256, 32, 32)
BatchNorm-30	(4, 128, 64, 64)	Conv2D-61	(4, 128, 32, 32)

Layer (type)	Output shape	Layer (type)	Output shape
BatchNorm-62	(4, 128, 32, 32)	BatchNorm-96	(4, 128, 16, 16)
Activation-63	(4, 128, 32, 32)	Activation-97	(4, 128, 16, 16)
Conv2D-64	(4, 128, 32, 32)	Conv2D-98	(4, 256, 16, 16)
BatchNorm-65	(4, 128, 32, 32)	Residual-99	(4, 256, 16, 16)
Activation-66	(4, 128, 32, 32)	UpSample-100	(4, 256, 32, 32)
Conv2D-67	(4, 256, 32, 32)	Hourglass-101	(4, 256, 32, 32)
Residual-68	(4, 256, 32, 32)	BatchNorm-102	(4, 256, 32, 32)
MaxPool2D-69	(4, 256, 16, 16)	Activation-103	(4, 256, 32, 32)
BatchNorm-70	(4, 256, 16, 16)	Conv2D-104	(4, 128, 32, 32)
Activation-71	(4, 256, 16, 16)	BatchNorm-105	(4, 128, 32, 32)
Conv2D-72	(4, 128, 16, 16)	Activation-106	(4, 128, 32, 32)
BatchNorm-73	(4, 128, 16, 16)	Conv2D-107	(4, 128, 32, 32)
Activation-74	(4, 128, 16, 16)	BatchNorm-108	(4, 128, 32, 32)
Conv2D-75	(4, 128, 16, 16)	Activation-109	(4, 128, 32, 32)
BatchNorm-76	(4, 128, 16, 16)	Conv2D-110	(4, 256, 32, 32)
Activation-77	(4, 128, 16, 16)	Residual-111	(4, 256, 32, 32)
Conv2D-78	(4, 256, 16, 16)	UpSample-112	(4, 256, 64, 64)
Residual-79	(4, 256, 16, 16)	Hourglass-113	(4, 256, 64, 64)
BatchNorm-80	(4, 256, 16, 16)	BatchNorm-114	(4, 256, 64, 64)
Activation-81	(4, 256, 16, 16)	Activation-115	(4, 256, 64, 64)
Conv2D-82	(4, 128, 16, 16)	Conv2D-116	(4, 128, 64, 64)
BatchNorm-83	(4, 128, 16, 16)	BatchNorm-117	(4, 128, 64, 64)
Activation-84	(4, 128, 16, 16)	Activation-118	(4, 128, 64, 64)
Conv2D-85	(4, 128, 16, 16)	Conv2D-119	(4, 128, 64, 64)
BatchNorm-86	(4, 128, 16, 16)	BatchNorm-120	(4, 128, 64, 64)
Activation-87	(4, 128, 16, 16)	Activation-121	(4, 128, 64, 64)
Conv2D-88	(4, 256, 16, 16)	Conv2D-122	(4, 256, 64, 64)
Residual-89	(4, 256, 16, 16)	Residual-123	(4, 256, 64, 64)
BatchNorm-90	(4, 256, 16, 16)	Conv2D-124	(4, 256, 64, 64)
Activation-91	(4, 256, 16, 16)	BatchNorm-125	(4, 256, 64, 64)
Conv2D-92	(4, 128, 16, 16)	Conv-126	(4, 256, 64, 64)
BatchNorm-93	(4, 128, 16, 16)	Features-127	(4, 256, 64, 64)
Activation-94	(4, 128, 16, 16)	Conv2D-128	(4, 16, 64, 64)
Conv2D-95	(4, 128, 16, 16)	MySequential-129	(4, 16, 64, 64)

## 10 Appendix B



**Figure 14:** While the loss is close to the train loss, it seems to be less stable