Application of convolutional neural networks to the problem of pose detection in images

Erlantz Calvo **Daniel Cañadillas**

Abstract

Pose estimation involves determining the location of parts of a body and joints in an image (Munea et al. [6]). This is one of the most treated problems in the field of computer vision because of its usefulness for the classification of human activities. One of the most recent approaches to pose estimation is based on the use of deep neural networks. Pose estimation can be used for activity recognition, augmented reality and training robots among other uses.

Contents

1 Description of the problem **1**

2 Description of our approach **2**

2.1 Head detection 3

2.2 Pose estimation 3

2.3 Models 4

2.3.1 Convolutional Neural Network 4

2.3.2 Residual Neural Network 6

3 Results **7**

3.1 Head detection 7

3.2 Pose estimation 9

4 Conclusion **11**

1 Description of the problem

Our goal in this project was to implement a deep neural network that, given a picture with people in it, the network has to be able to detect the person's pose. This task can be handled using the MPII Human Pose dataset (Andriluka et al. [1]) as a way of using the current state of the art for evaluating articulate human pose estimators. The dataset includes around 25K images containing over 40K people with annotated body joints, besides, covers 410 human activities. This amount of variability inside the dataset provides the possibility to learn a robust model in order to make good predictions in any situation. Images from the dataset correspond to different situations, captured in different sizes, that one or more people are involved.

For this dataset we are provided with different annotations. These annotations are made in such a way that each person that appears in each image has his or hers information labeled. From all the annotations, we used for each image the following:

- Head Rectangle (figure 1): (x_{start}, y_{start}) (x_{end}, y_{end})

Figure 1: Ground truth for head rectangle coordinates

- Joints coordinates (figure 2):

0 - Right ankle	1 - Right knee	2 - Right hip	3 - Left hip
4 - Left knee	5 - Left ankle	6 - Pelvis	7 - Thorax
8 - Upper neck	9 - Head top	10 - Right wrist	11 - Right elbow
12 - Right shoulder	13 - Left shoulder	14 - Left elbow	15 - Left wrist

Table 1: Joints enumeration

Figure 2: Ground truth for joint coordinates

Due to this variety, we decided to select as tasks, head detection and pose estimation. More precisely a regression task where given a image will predict the coordinates of the head rectangle, in the case of head detection, or the coordinates of all the joints, in the case of pose estimation.

2 Description of our approach

To tackle this problem, we decided to remove all the pictures with more than one person in it so the model only has to detect one person. This decision was made by taking into account the computational cost and the limited resources we had.

The dataset's images don't have a fixed dimension, so each of them may have different width and height and that generates some problems in order to pass the images to the neural network, hence, before the images are passed to the model, they are resized to 220x220 pixels, so the output coordinates are also converted to the new range.

Our solution has been carried out using two different types of deep neural networks, a solution based on Convolutional Neural Network (CNN) (Toshev and Szegedy [9]) and another one based on Residual Neural Network (ResNet) (He et al. [3]).

In order to solve this task we went through two processes: Detecting the head of the person in the picture and estimating the pose of the person in the picture.

2.1 Head detection

The dataset includes coordinates of people's heads in the images. These coordinates are represented by four decimal real numbers that indicates the coordinates of the top-left and bottom-right corners of a box containing the head. After resizing the image, we pass them to the two models in the same way, so they start training taking as input one image and giving as output a vector of four real values as a regression task.

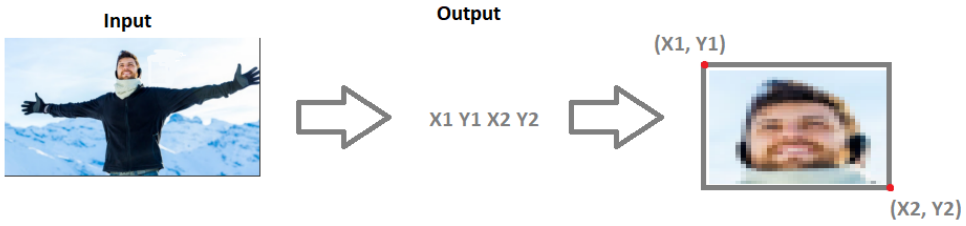


Figure 3: Input/output format.

As loss function and evaluation metric we have used Mean Squared Error in order to measure how far the coordinates have been predicted from the ground truth ones.

$$Mean\ squared\ error\ MSE = \frac{1}{n}\sum_{t=1}^{n}(y - \widehat{y})^2$$

2.2 Pose estimation

For the pose estimation task, this one has as input one image and returns as output a vector of thirty-two real values representing each joint of person's body following the same organization as the mentioned in table 1 (sixteen X coordinates and sixteen Y coordinates).

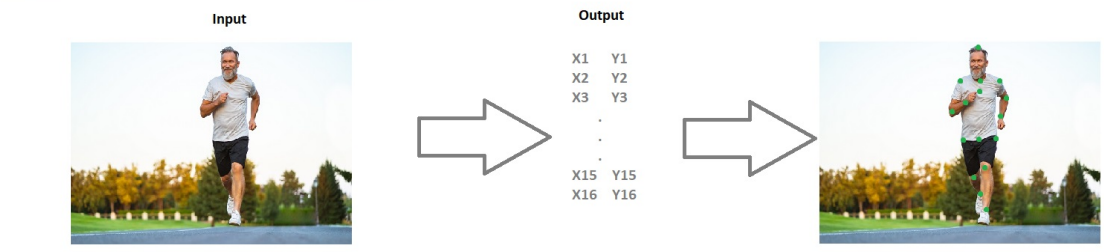


Figure 4: Input/output format.

As loss function the MSE is used for the same reason as in the previous task, measuring how far are the coordinates from the ground truth coordinates. In this task the evaluation metric changes to the probability of correct keypoint (PCK) (Yang and Ramanan [11]) that uses the matching threshold as 50% of the head segment length (PCKh @ 0.5) (Andriluka et al. [1]). This metric measures the accuracy of the locations of the body joints. This metric works in the following way:

1. The L2 norm (euclidean distance) between the predicted and the ground truth point is calculated.

$$d(y, \widehat{y}) = ||y - \widehat{y}||_2 = \sqrt{\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2}$$

Where y is the coordinates of the ground truth joint (x_{joint}, y_{joint}) and $\widehat{y}$ is the coordinates of the predicted joint $(\widehat{x_{joint}}, \widehat{y_{joint}})$

2. Once the distance is calculated, it has to be normalized by the head bone length, in this case we had defined the head bone as the length of the vector that goes from the Head top joint to the Upper neck joint.

$$d_{head\ bone}(y, \widehat{y}) = \frac{||y, \widehat{y}||_2}{||j_{head\ top} - j_{upper\ neck}||_2}$$

Where $d_{head\ bone}(y, \widehat{y})$ is the distance of the predicted vector and the ground truth normalized by the head bone length.

3. If the calculated value is lower than a threshold of 0.5 (if has a distance from its ground truth of half the head bone length) the model has made a good prediction, otherwise, the prediction is not correct.

$$f(y, \widehat{y}) = \begin{cases} 1 & , d_{head\ bone}(y, \widehat{y}) <= 0.5 \\ 0 & , otherwise \end{cases}$$

$$PCKh(y, \widehat{y}) = \frac{1}{N}\sum_{i=1}^{N} f(y_i, \widehat{y_i})$$

Then the accuracy is calculated by the mean of the PCKh for each prediction each training step.

2.3 Models

As Deep Neural Network approach we decided to use two different Deep Neural Network models and analyze their results between each other. One was based on Convolutional Neural Networks (LeCun et al. [5]) and another on Residual Neural Networks (He et al. [3]).

2.3.1 Convolutional Neural Network

This CNN's architecture is based on the DeepPose Human pose estimation model (Toshev and Szegedy [9]), with the architecture shown in figure 5.

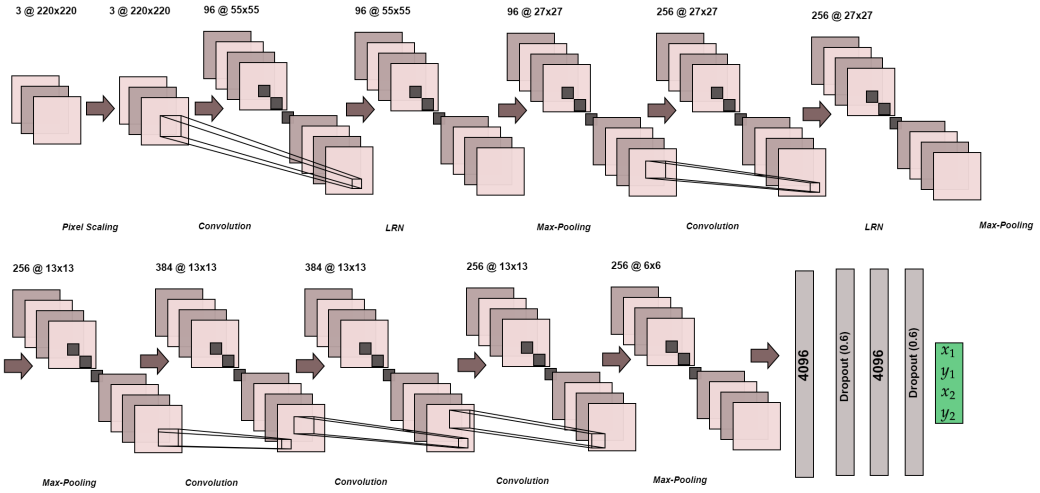


Figure 5: CNN based model

The input is an image of 220x220 dimensions with a channel size of 3 (RGB). As we have mentioned before, this project has a high computational cost so, in order to reduce it, the second layer is a pixel scaling layer. This layer divides all the pixels of the image by 255, causing the image to only have values between 0 and 1, this process can be called as scaling to the range of $[0, 1]$ and it is useful in order to speed up the convergence of the model. We have tried to train the same model with this pixel scaling layer and without it, and, without this layer, the training time is much higher (around 9 - 10 times higher), and the convergence is also slower.

The next layer is the first convolutional layer of the CNN. This layer makes a convolution of size 11x11, for each of the 96 filters (figure 6b), with a stride of 4x4 and padding in order to avoid information loss during convolutions.

The following layer is called Local Response Normalization layer (LRN) (Robinson et al. [7]). Its purpose is to simulate the human's neurobiological concept called "lateral inhibition". This layer is useful when a Rectified Linear Unit (ReLU) activation function is being used, as in our case, all activation functions are ReLU. This is because the ReLU activation function may output big values, so the local response normalization layer normalizes all the outputs, making the most important neurons (the ones that have more relevance over the image) getting more impact than their neighbours.

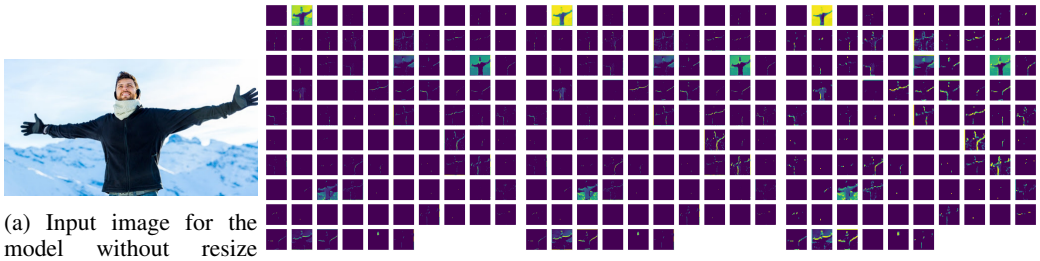
The model now reduces the input layer dimensions by half using a Max-Pooling layer with a pooling size of 2x2, outputting for each input from the previous LRN layer a half sized output with the max value for each operation, of 2x2 receptive field, during the pooling process (figure 6d).

After this 3 layers, the model continues applying the same structure of all the 3 explained layers with different parameters, the convolution now has a kernel size of 5x5 by 256 filters with padding and without stride , next the LRN layer is applied and finally the same pooling layer is applied (2x2).

Then, a 384 filters convolution with a kernel size of 3x3 is applied twice, both of them with padding and no stride, after this two convolutions, a last one is added, 256 filters convolution with a kernel size of 3x3 with padding and no stride followed by the last Max-Pooling layer (2x2).

Finally, we use the output from the last max-pooling layer as the first input for the Fully Connected layer by flattening all the applications of the max-pooling. At that point, we connected it with a fully connected layer of 4096 neurons and this fully connected layer with another one of the same size. Subsequently, we connected our fully connected layer with the output layer, a 4 neurons output layer in the head detection model and a 32 neurons output layer in the case of pose estimation model, in both cases with a ReLU as activation function. During these fully connected layers, a dropout is applied to each of them except for the output one, forcing the neurons to take more responsibility and depending less on the neighbours (Srivastava et al. [8]). These dropouts use a

60% probability of ignoring the neuron during training process causing the remaining neurons to take more responsibility on their outputs.



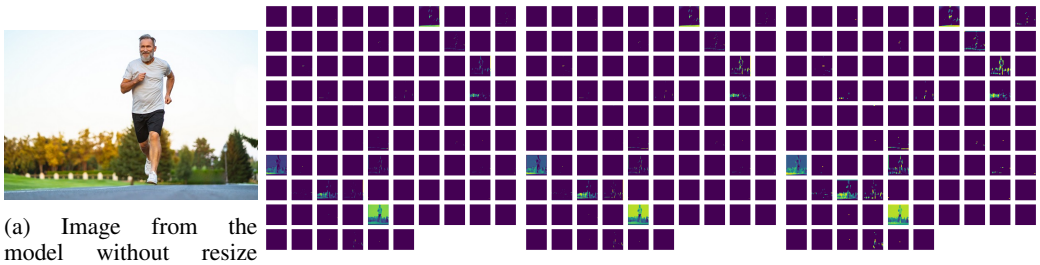
(a) Input image for the model without resize (512x275x3)

(b) First Convolutional layer with scaled input (kernel 11x11@96)

(c) Local Response Normalization Layer (with input shown in figure 6b)

(d) Max-Pooling process having as input the LRN layer results from figure 6c

Figure 6: First 3 layers: (Head Detection) Convolution - LRN - Max-pooling



(a) Image from the model without resize (512x275x3)

(b) First Convolutional layer (kernel 11x11@96)

(c) Local Response Normalization layer

(d) Max-Pooling layer

Figure 7: First 3 layers: (Pose Estimation) Convolution - LRN - Max-pooling

2.3.2 Residual Neural Network

A Residual neural network (ResNet) works in a similar way as a CNN but with a main difference, the ResNet can skip connections to jump over some layers (Zaeemzadeh et al. [12]). The meaning of this mechanic is that the typical ResNet models are implemented with double or triple layer skips that contain nonlinearities (ReLU) and batch normalization in between.

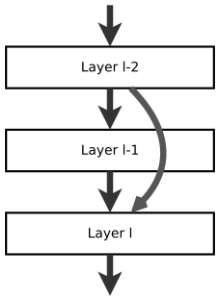


Figure 8: Typical Residual neural network architecture.

One motivation for skipping over layers is to avoid the problem of vanishing gradients (Veit et al. [10]), by reusing activations from a previous layer until the adjacent layer learns its weights. The vanishing gradient problem causes to a neural network difficulties when updating its weights or usually, make that task impossible.

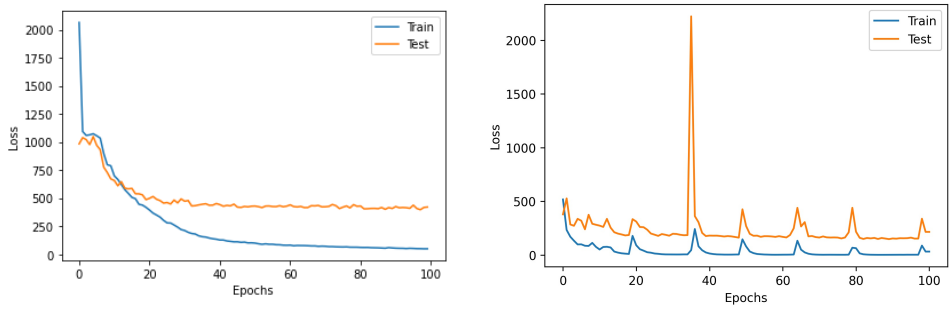
In our case, we decided to use the ResNet50V2 (He et al. [3]) pretrained for the ImageNet task (Deng et al. [2]) as initial weights for our tasks, head detection and pose estimation. Although ImageNet is a classification task, we think the internal learning process could help in our regression tasks due to the variations between inputs in both applications. Now, instead of having around 10 layers, this network provides a depth around 50 layers, although it was modified in the input and output layer in order to achieve our tasks (regression instead of classification) and being able to train this network within a time and computational cost we could assume.

3 Results

During the project we trained both networks in different ways for each of the tasks but maintaining the same training parameters in both of them for a fair comparison between them. 11503 train instances and 6908 for test.

3.1 Head detection

For the head detection task we decided to establish a batch size of 128 images for the DeepPose based model and 32 images for the ResNet. For each of them we set 100 epochs, having as loss function the mean squared error and Adam (Kingma and Ba [4]) as the optimizer with a learning rate of 0.0005. As we can see in figure 9a during training the loss was decreasing continuously while the validation loss reduced until a certain point ($\sim$30th epoch) and then it got stable until the end of training process.



(a) Head detection DeepPose based CNN

(b) Head detection ResNet based

Figure 9: Training process for Head detection with DeepPose and ResNet based models

We can see in figure 9 that the ResNet's performance is quite better than the CNN, although, the ResNet has around 32 million parameters while the CNN has got around 58 million parameters which is a lot more in terms of computational cost. The DeepPose based model is continuously decreasing as we mentioned before, but, ResNet has a faster convergence as shown in figure 9b. Both of them has not a big generalization error so we can estimate that they can perform well their task in an unknown case.

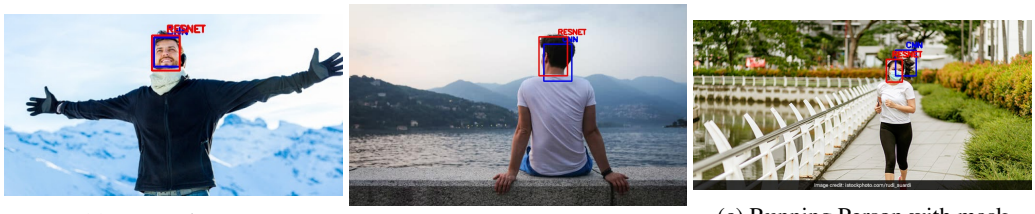
In order to test this model we think that showing how their evolution is during epochs is a good point of view on how they perform in new cases.

Figure 10: Prediction of DeepPose based model in epochs 10, 40, 60 and 100

Figure 11: Prediction of ResNet based model in epochs 10, 40, 60 and 100

In figure 11 we can see how in the first few epochs the ResNet's error is still visible but, in the epoch 40 the model achieves it's best performance with the given image. On the contrary we can see that the DeepPose based model does not improve as much as ResNet during epochs but still making a good prediction.

After analysing their predictions during epoch we decided to test them with different images that reflect in different situations which is the best model. For example, we applied both models in a person who is backwards and we can still see how models detect their heads (figure 12b), or how they detect the head in another picture with a person running with a mask in her face (figure 12c).



(a) Forward Person

(b) Backward Person

(c) Running Person with mask

Figure 12: DeepPose and ResNet based model predictions

After doing some visualizations we realized that the models can detect not only human faces in the pictures but also faces of humanoid entities in the images. With the will of showing this belief, in figure 13 we present how the models can detect the face of the David (Michelangelo) statue, of a surikata (animal) or a cartoon character.

(a) Surikata

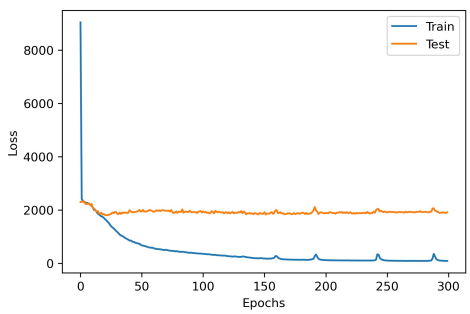
(b) Cartoon character

(c) David (Michelangelo)

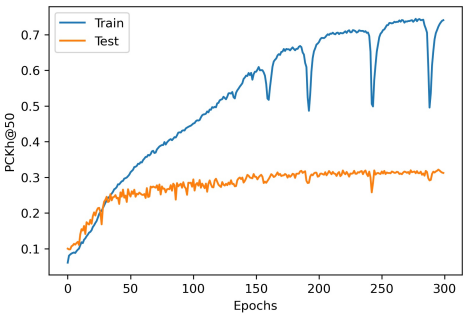
Figure 13: DeepPose and ResNet based model predictions

3.2 Pose estimation

For the pose estimation task we decided to establish a batch size of 128 images for the DeepPose based model and 32 images for the ResNet. For each of them we set 300 epochs, having as loss function the mean squared error and Adam (Kingma and Ba [4]) as the optimizer with a learning rate of 0.0005. In addition, we added the PCKh@50, explained before, as the metric for this task.



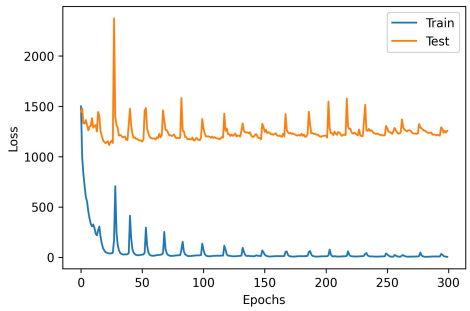
(a) Pose Estimation DeepPose based CNN mean squared error



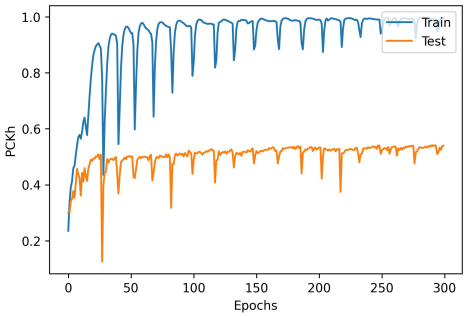
(b) Pose Estimation DeepPose based CNN PCKh@50

Figure 14: Training process for Pose Estimation with DeepPose based model

As we can see in figure 14 and figure 15, as the loss decreases the PCKh@50 gets higher until its convergence to some value depending on the model, from that moment both models improve very slowly almost imperceptible.



(a) Pose Estimation ResNet based mean squared error



(b) Pose Estimation ResNet based PCKh@50

Figure 15: Training process for Pose Estimation with ResNet based model

We can see in figure 14 and in figure 15 that the ResNet's performance is quite better than the CNN, although, the ResNet's number of parameters have risen from 32M to 35M while the CNN still has 58M parameters.

The DeepPose based model decreases the training error continuously but the test error keeps stable until the end. The PCKh@50 value for measuring the joint's accuracy improves a lot over the epochs for the training set, but for the testing set at some point the error gets stuck.

ResNet gets a higher performance than CNN in training and testing but having a look on different situations and images we think that it may have some overfitting compared on CNN's predictions.

So, as we measured, CNN reached an accuracy of 30% based on PCKh@50, and ResNet reached an accuracy of 50% with the same measure.

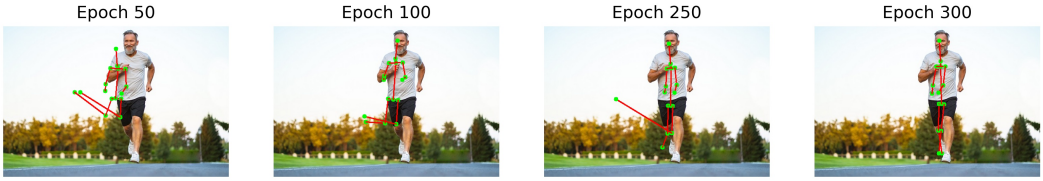


Figure 16: Prediction of DeepPose based model in epochs 50, 100, 250 and 300

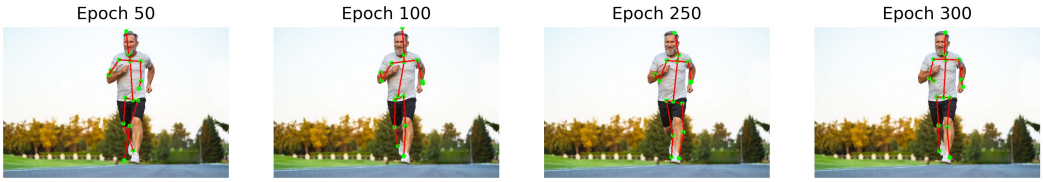


Figure 17: Prediction of ResNet based model in epochs 50, 100, 250 and 300

Taking a look at figures 17 it is easy to notice how faster the ResNet converges to the best pose estimation and, from the epoch 50 to epoch 300 the joint estimation changes very little, showing that the model is polishing the prediction. On the other hand, the DeepPose based model, at figure 16, takes time to converge and, when it does, it is not as good as the ResNet model.

(a) Running person (b) Standing up person (c) Barman (d) Person practising yoga

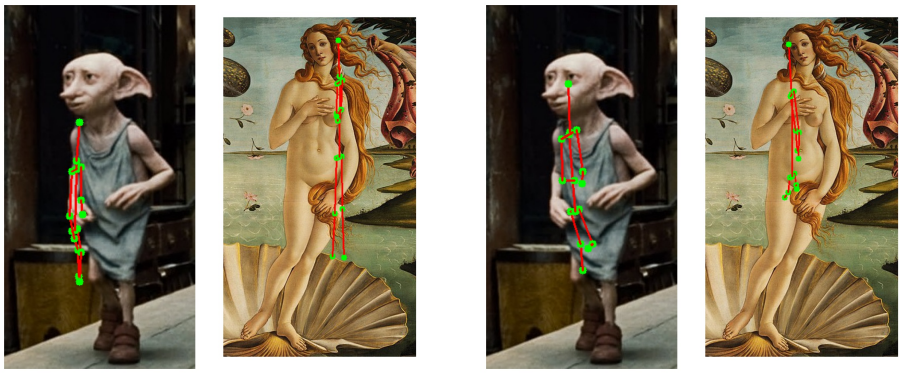
Figure 18: DeepPose based model predictions

(a) Running person (b) Standing up person (c) Barman (d) Person practising yoga

Figure 19: ResNet based model predictions

To remark some differences between the two models presented we show some important differences in the figures 19 and 18. The DeepPose based model has predicted some wrong joints for the barman's legs while the ResNet has not predicted any joints for the same image. Regarding the barman's legs example, we have done that, when the model predicts that a person does not show any joint in the images, the model sets those joint's coordinates out of the image, this is, it predicts values out of the image's dimensions range, so they are not taking into account when the result is plotted.

In the yoga image, both models did not predict how the legs are crossed, and, in general, the extremities of the person. On the other hand, in the standing up person picture we think due to the color palette both of the models were not able to predict as well as in other cases.



(a) CGI character (DeepPose)

(b) Birth of Venus (DeepPose)

(c) DeepPose based model predictions

(d) CGI character(ResNet)

(e) Birth of Venus (ResNet)

(f) ResNet based model prediction

Figure 20: Some Singular predictions

As we did before with the Head Detection task, we tried to analyse how the models would work with humanoid pictures, in this case both of them, DeepPose and ResNet based models, were not able to predict the joints neither from the CGI character nor the painting, although ResNet was able to predict where the humanoid is located in the picture.

4 Conclusion

After developing this project and seeing the results we obtained by comparing these proposed models, we made up some conclusions.

During pose estimation, after seeing the results and investigating the dataset, we think that there are some data that is wrong labeled, mainly we realized that due to pictures that have not all the joints. When some joints are missing the models get in some way "crazy", this happens because some points are not represented in the image but still have ground truth values, but this values were labeled poorly by making some extrapolations o directly converted to 0's (there are some more information of the visibility of the joints, but they are also wrong labeled). We can see in figure 19c or during epochs (figure 16) that leg's joints are tending to the up-left corner, this is probably originated by the representation of the information the model learnt, "if I cannot detect the leg's joints probably they are in the $(0, 0)$ coordinates like the images from the dataset I have been learning from".

Another issue that is easy to see is that the head prediction works way better than the joint estimation one for both of the models (DeepPose based(Toshev and Szegedy [9]) and ResNet(Veit et al. [10])). The reason of this is that, for the head detection task the model only has to predict 4 continuous values and also is easier to get filters that can generalize people faces since usually they follow a similar shape pattern. In the pose estimation task 32 continuous values have to be predicted and the filter may be harder to be created since the human pose may vary a lot. This is, it does not matter so much how a person is behaving in an image, as its face is going to follow the usual pattern (eyes, nose, mouth, head shape...) but, on the other hand, when the models have to predict body's joints the pose or many other factors can change completely the pattern, i.e., a knee can be completely folded or completely stretched and it's difficult for a filter to predict both of the scenarios.

Finally, we think our approach (predicting a vector of coordinates) is a good approach in tasks that the complexity of the context allows us to condense information in a not too large vector, but, as we have seen in pose estimation, they make a sufficiently good approach, but for a real point of view they are about to mature. So we think the next step from here to a more powerful tool for

both of the tasks would be to use heat maps or segmentation maps. This would allow to detect more efficiently the head rectangle and the joints by, for example, converting the regression task in a pixel classification task and with the use of some more computational costly models like U-Net and some centroid estimation, better results should be achieved.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[6] Tewodros Legesse Munea, Yalew Zelalem Jembre, Halefom Tekle Weldegebriel, Longbiao Chen, Chenxi Huang, and Chenhui Yang. The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8:133330–133348, 2020.

[7] Alan E Robinson, Paul S Hammon, and Virginia R de Sa. Explaining brightness illusions using spatial filtering and local response normalization. *Vision research*, 47(12):1631–1644, 2007.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[9] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[10] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29:550–558, 2016.

[11] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2878–2890, 2012.

[12] Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. Norm-preservation: Why residual networks can become extremely deep? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.