A BOTTOM-UP APPROACH FOR PIG SKELETON EXTRACTION USING RGB DATA.

Akif Quddus Khan *Salman Khan*

Department of Computer Science (IDI), Norwegian University of Science and Technology, Norway.

ABSTRACT

Animal behavior analysis is a crucial tasks for the industrial farming. In an indoor farm setting, extracting Key joints of animal is essential for tracking the animal for longer period of time. In this paper, we proposed a deep network that exploit transfer learning to trained the network for the pig skeleton extraction in an end to end fashion. The backbone of the architecture is based on hourglass stacked dense-net. In order to train the network, key frames are selected from the test data using K-mean sampler. In total, 9 Keypoints are annotated that gives a brief detailed behavior analysis in the farm setting. Extensive experiments are conducted and the quantitative results show that the network has the potential of increasing the tracking performance by a substantial margin.

Index Terms— Pig, behavior analysis, Hourglass, Stacked dense-net, K-mean sampler, .

1. INTRODUCTION

Automatic behavior analysis of different animal species is one of the most important tasks in computer vision. Due to variety of applications in the human social world like sports player analysis [1], anomaly detection [2], action recognition [3], crowd counting [4], and crowd behavior [5], humans have been the main focus of research. However, due to growing demands of food supplies, vision based behavior analysis tools are pervasive in the farming industry and demands for cheaper and systematic solutions are on the rise. From the algorithmic point of view, other than characterization of the problem, algorithm design for humans and the farm animals are similar. Essentially, behavior analysis is a high level computer vision task and consist of feature extraction, 3D geometry analysis, and recognition, to name a few. As far as the input data is concerned, it could be obtained thought smart sensors (Radio-frequency identification [6], gyroscope [7], GPS [8]). Depending on the precision of measurements, such sensors give acceptable results but they using such sensors has many drawbacks. For example, in most cases, it is required to remove the sensor from the animal to collect the data. Such process is exhausting for the animals and laborious for the human operator. Compared to this, video based automated behaviors analysis offers a non-invasive solution. Due to cheaper hardware, it is not only convenient for the animals but also cost effective for the industry. Automatic behavior analysis and visual surveillance [9, 10] has been used for the security of public places (airports, shopping malls, subways etc.) and turned into a mature field of computer vision.

In this regard, Hu et al. [11] proposed a recurrent neural network named MASK-RNN for the instance level video segmentation. The network exploit the temporal information from the long video segment in the form of optimal flow and perform binary segmentation for each object class. Ullah et al. [12] extracted low level global and Keypoint features from video segments to train a neural network in a supervised fashion. The trained network classify different human actions like walking, jogging, running, boxing, waving and clapping. Inspired from the human social interaction, a hybrid social influence model has been proposed in [13] that mainly focused on the motion segmentation of the moving entities in the scene. Lin et al. [14] proposed a features pyramid network that extract features at different level of a hierarchical pyramid and could potentially benefit several segmentation [11, 15], detection [16, 17], and classification [18, 19] frameworks. By addressing the problem of scale variability for object detection, Khan et al. [17] proposed a dimension invariant convolution neural network (DCNN) that compliment the performance of RCNN [16] but many other state-of-the-art object detector [4, 20] could take advantage of it. Inspired from the success of deep features, [21] proposed a two stream deep convolutional network where the first stream focused on the spatial features while the second stream exploit the temporal feature for the video classification. The open-source deep framework named OpenPose proposed by Cao et al. [22] focuses on the detection of Keypoints of the human body rather than detection of the whole body. Detection of Keypoints have potential applications in the pose estimation and consequently behavior analysis. Their architecture consist of two convolutional neural network where the first network extract features and gives the location of main joints of the human body in the form of heat map. While the second network is responsible for the associating the corresponding body joints. For the feature extraction, they used the classical VGG architecture. The frameworks like OpenPose are very helpful in skeleton extraction of the human body and potentially, it could be used in tracking framework. For example, the Bayesian framework proposed in [23] works as a Keypoint tracker. Where any Keypoints like the position of head, or neck or any other body organ

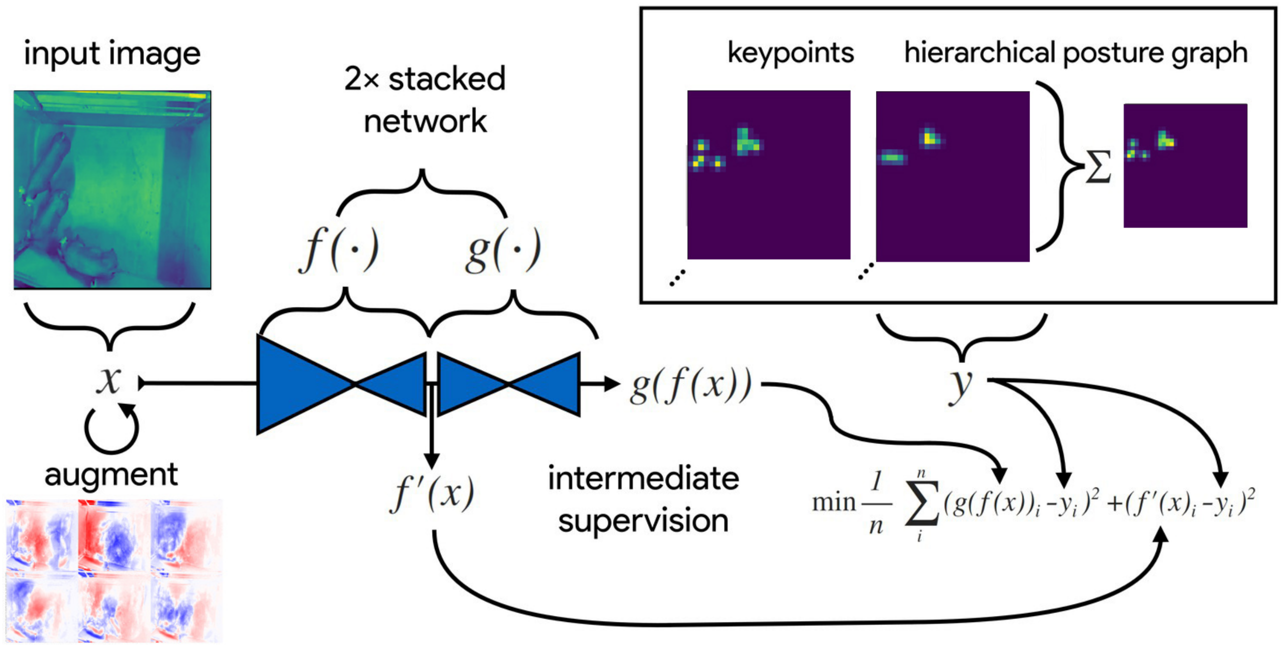

Fig. 1: Detailed Illustration of Model Training Process

can be used to do tracking for longer time. Such Keypoints can be obtained from a variety of human pose estimation algorithms. For example, Sun et al. [24] proposed a parallel multi-resolution subnetworks for the human pose estimation. The idea of parallel network helps to preserver high resolution and yield high quality features maps that results better spatial Keypoints locations. Essentially, in such a setting, the detection module is replaced by [22, 24]. In this regards, a global optimization approach like [25] could be helpful for the accurate tracking in offline setting. By focusing only on pose estimation of humans, Fang et al. [26] proposed a top down approach where first the humans are detected in the form of bounding boxes and later, the joints and Keypoints are extracted through a regional multi-person pose estimation framework. Such a framework is helpful in not only in the localization and tracking of tracking in the scene, but also getting the pose information of all the targets in sequential fashion. For a robust appearance model that could differentiate between different targets, a sparse coded deep features framework was proposed in [27] that accelerate the extraction of deep features from different layers of a pre-trained convolution neural network. The framework is helpful in handling the bottleneck phenomenon of appearance modeling in the tracking pipeline. Alexander et al. [28] used transfer learning and fine-tuned ResNet to detect 22 joints of horse for the pose estimation. They used the data collected from 30 horses for the within domain and out of domain testing. The work by Mathis et al. [29] analyzed the behavior of mice during the experimental tasks of tail-tracking, and reach & pull of joystick tasks. They also analyze the behavior of drosophila while it lays eggs. The classical way of inferring behavior is to perform segmentation and tracking [30] first, and based on the temporal evolution of the trajectories, perform behavior analysis. However, approaches like [31] can be use to directly infer predefined actions and behaviors from the visual data. In addition to the visual data, physiological signals [32, 33], and acoustic signals [34] can be used to identify different emotional states and behavioral traits in farm animals.

Compared to the existing methods, our proposed framework is focused on the extraction of the key joints of the pig in an indoor setting. The visual data is obtained from a head mounted Microsoft Kinect sensors. Our proposed framework is inspired by [35] where a fully-convolutional stacked hourglass-shaped network is proposed that converts the image into a 16-channel space representing detection maps. For the part detection, the thresholds are set from 0.10 to 0.90. These threshold are used while evaluating the recall, precision, and F-measure metrics for both the vector matching and euclidean matching results. Such an analysis provides a detailed overview of the trade-offs between precision and recall while maintaining an optimal detection threshold. The loss function, the optimizer and the training details are also given in 3. The qualitative results are mentioned in 4 and the remarks are given in 5 which concludes the paper.

The rest of the paper is organized in the following order. In section 2 the proposed method is briefly explained including the Keypoints used in the experiment, the data filtration and annotation, and the augmentation. Model architecture is elaborated in section 3. The loss function, the optimizer and the training details are also given in 3. The qualitative results

Table 1: Key Points Names

name	parent	swap
snout		
head	snout	
neck	head	
forelegL1	neck	forelegR1
forelegR1	neck	forelegL1
hindlegL1	tailbase	hindlegR1
hindlegR1	tailbase	hindlegL1
tailbase		
tailtip	tailbase	

are mentioned in 4 and the remarks are given in 5 which concludes the paper.

2. PROPOSED APPROACH

The block diagram of the network is given in Fig. 1. It mainly consist of two encoder-decoder stacked back to back. The convolution neural network used in each encoder-decoder network is based on dense net. The network takes the input as the visual frame. In order to train the model, we annotate the data by first converting videos into individual frames, and then annotating each frame separately by specifying the important key points on the animal's body. After sufficient training, the model returns a 9x3 matrix for each frame, where each row corresponds to one keeping, the first two columns specify the x and y coordinates of the detected point, and the third column contains the confidence score of the model. After obtaining the x and y coordinates for each frame, we visualize these key points on each frame and stitch all the individual frames into a single video. A total of nine key points is being focused for each pig. Key points are as follows: Nose, Head, Neck, Right Foreleg, Left Foreleg, Right Hind leg, Left Hind Leg, Tail base, and tail tip. Key points and their connection is written in the CSV file for the annotator to read. Exact key points and their relations are shown in Table 1.

In Fig. 1, input images x (top-left) are augmented (bottom-left) with various spatial transformations such as rotation, translation, scale, etc. In addition to that, they are followed by noise transformations such dropout, additive noise, blurring, contrast, etc. These transformations are applied to improve the robustness and generalization of the model. The ground truth explanations are then changed with coordinating spatial growths and used to draw the certainty maps y for the key points and various hierarchical posture graph (top-right). The images x are then passed through the network to produce a multidimensional array $g(f(x))$ a stack of images corresponding to the key point and posture graph confidence maps for the ground truth y. Mean squared error between the outputs for both networks $g(f(x))$ and $f'(x)$ and the ground truth data y is then minimized (bottom-right), where $f'(x)$ indicates a subset of the output from $f(x)$. only those feature maps being optimized to reproduce the confidence maps for the purpose of intermediate supervision. The loss function is minimized until the validation loss stops improvingindicating that the model has converged or is starting to over fit to the training data [36].

In the subsequent subsections, the data filtration, data augmentation and data annotation techniques used to produce the training data is explained.

2.1. Data Filtration

The given data is in RGB format had different categories as per the number of Pigs. For example, there is a separate dataset for three pigs, six pigs, and 10 pigs. Each dataset has 2880 images. To get better and more accurate results, a larger dataset was required, hence more images need to be annotated. To overcome this problem, a video is created from the image data. After that, the frames were generated from the video. In order to save time and effort, while keeping in mind the original goal of accuracy, K-Means filter is applied to the frames, by which only 280 images are extracted from the larger dataset. The count is approximately 10 percent of the size of the original dataset. K-Means sample configuration is as follows:

- Batch size = 100
- clusters size = 100
- Reassignment Ratio = 0.01
- Tol = 0.0
- verbose = True

Skeleton from the CSV file and Image data extracted after applying the K-Means sampler, combined data is stored in .h5 format. Complete dataset after diving into 100 unique clusters is shown in Fig. 2.

2.2. Data Annotation

Data annotator developed by Jake Graving is used to annotate the dataset. It provides a simple graphical user interface that reads key points data from the CSV file and saves the data in .h5 format once the annotation is completed. The data annotator is show in Figure 3. DeepPoseKit works with augmenters from the imgaug package. We are using spatial augmentations with axis flipping and affine transforms. deepposekit.augment.FlipAxis takes the DataGenerator as an argument to get the key point swapping information defined in the annotation set. When the images are mirrored key points for left and right sides are swapped to avoid ”confusing” the model during training.

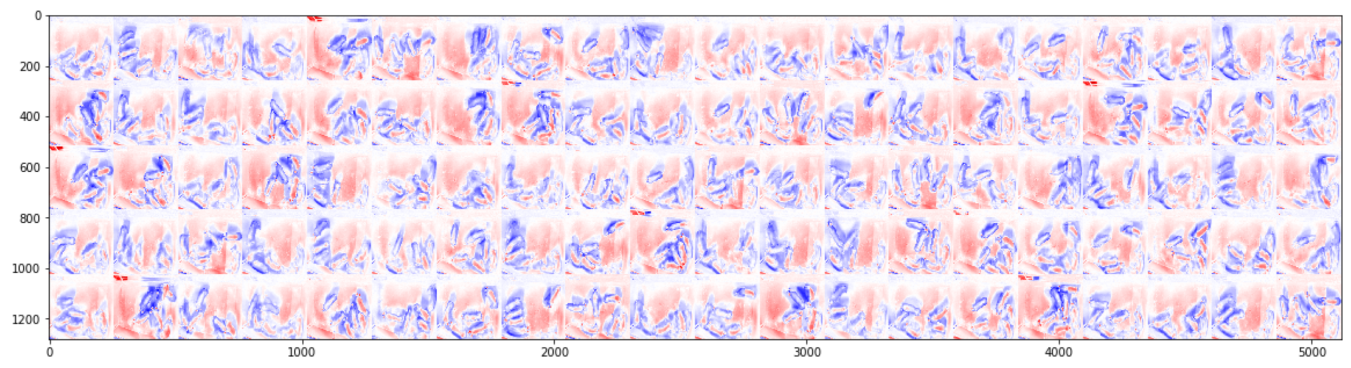

Fig. 2: Image Data after K-Means Sampler

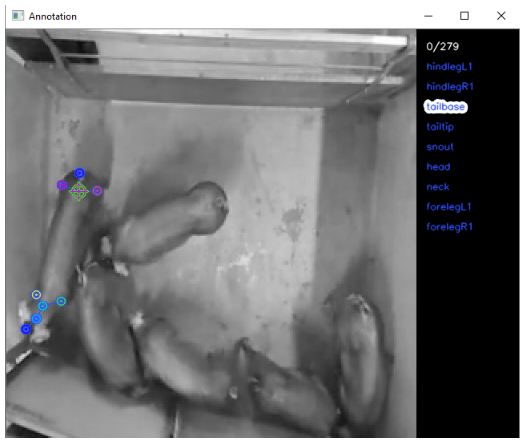

Fig. 3: Data Annotator

3. MODEL ARCHITECTURE

The proposed framework is based on Stacked densenet, an efficient multi-scale deep-learning model. A quick GPU-based pinnacle identification calculation for evaluating Keypoint areas with sub-pixel exactness. These advances improve preparing speed greater than 2x with no accuracy in exactness contrasted with at present accessible strategies [36]. Densenet is a densely Connected Convolutional Networks [37]. DenseNet can be seen as the next generation of convolutional neural network that are capable of increasing the depth of model with every decreasing the number of parameter.

3.1. Loss function & Optimizer

We have used the callback function using *ReduceLROnPlateau*. ReduceLROnPlateau automatically reduces the learning rate of the optimizer when the validation loss stops improving. This helps the model to reach a better optimum at the end of training. Following are the parameters for this function:

- Monitor: *val_loss*
- Factor: 0.2
- Verbose: 1
- Patience: 20

Mathematically, the loss function is:

$$L(x,y) = \frac{1}{n}\sum_{i}^{n}((g(f(x))_i - y_i)^2 + (f'(x) - y_i)^2) \quad (1)$$

3.2. Training

3.2.1. Training Model

While training a model on three pigs data, first a test was run for data generators. Figure 4 is the image of the very first image from the dataset.

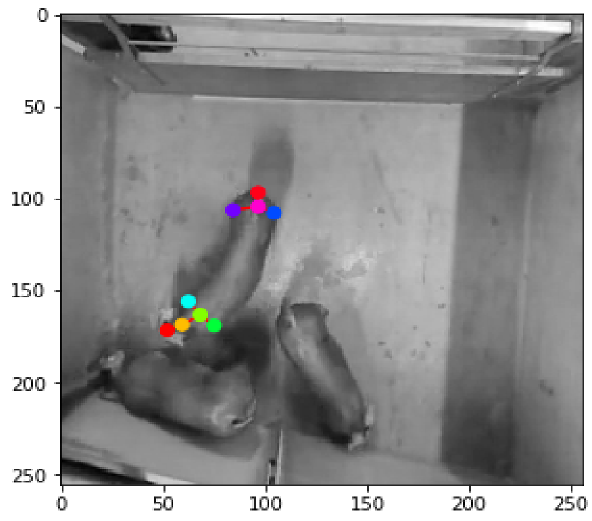

Fig. 4: Data Generator

3.2.2. Training Generator

Creating a TrainingGenerator from the DataGenerator for training the model with annotated data is an important factor. The TrainingGenerator uses the DataGenerator to load image-keypoints pairs and then applies the augmentation and draws the confidence maps for training the model. Figure 5 shows the training generator for one randomly picked frame.

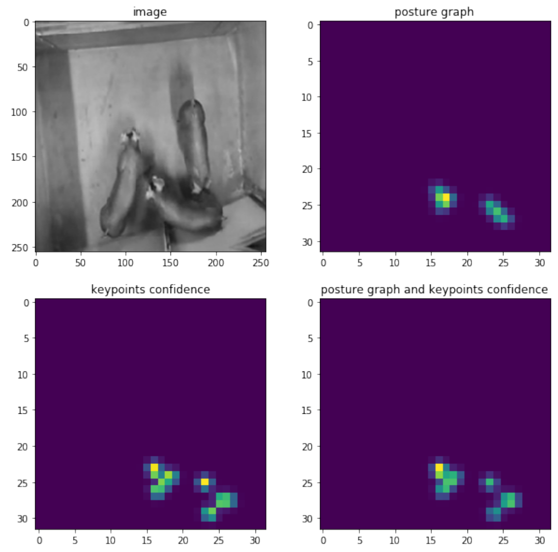

Fig. 5: Training Generator

The **validationSplit** argument defines how many training examples to use for validation during training. If a dataset is small (such as initial annotations for active learning), we can set this to **validationSplit=0**, which will just use the training set for model fitting. However, when using callbacks, we made sure to set monitor="loss" instead of monitor="valloss". To make sure the Training Generator is working, the following are some output visuals.

3.2.3. Reduce Learning Rate

Reduce learning rate parameters saves useless resource utilization and model overfeeding. For this particular reason, the parameter is set to reduce the learning rate by 0.2 if the loss does not improve after 20 iterations.

3.2.4. Early Stopping

Another parameter that is used to prevent resource exploitation is Early Stopping. Patience is set 100 iterations that are, training would stop automatically if the loss does not improve after 100 iterations.

3.2.5. Training Results

Training started at a loss of 220, after running 400 iterations, the loss stopped showing improvement at 4.5. In the test case, when given the same video from which dataset was generated, very accurate results are produced. Below are some results of the iterations after running 50 iterations on the dataset.

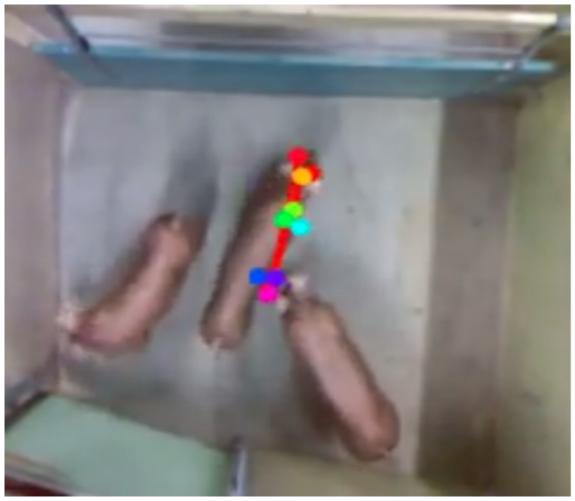

Fig. 6: Results after 50 Iterations

In Fig. 6, since the model is not properly trained, model is detecting very rough key points. Hardly the key points on head section is detected. On the tail section key points are not detected at all. Since the model showed a great tendency to learn, hence after 400 iterations, the results are shown in Fig. 7 and Fig. 8. Clearly difference between the accuracy can be seen between Fig. 6 and Fig. 7 as the result is generated on exactly the same frame.

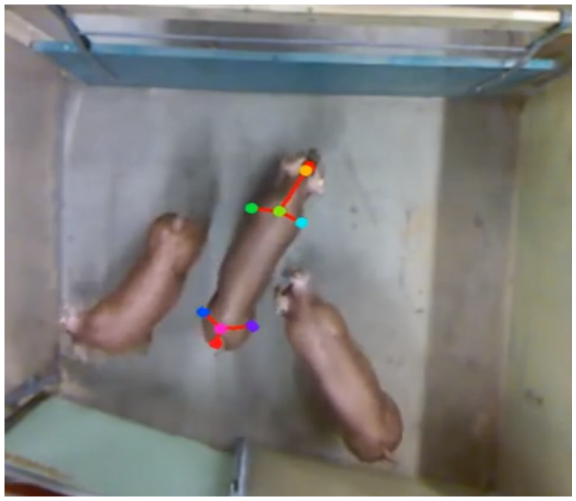

Fig. 7: Results after 400 Iteration

Result was generated on total of 2880 frames, in Figure 8 is showing pigs with different locations, yet very perfect key points detection.

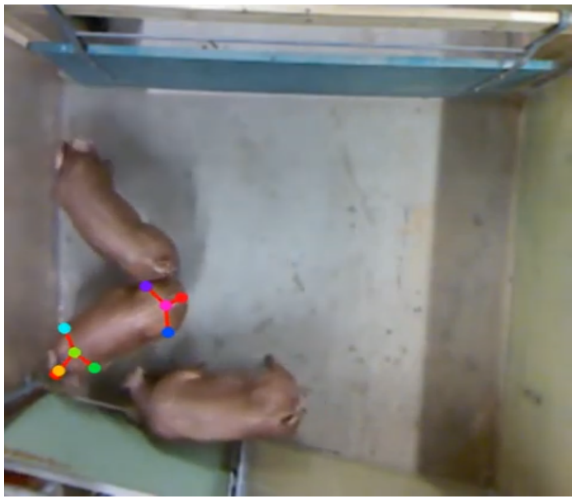

Fig. 8: Results after 50 Iterations

3.3. Outlier Frame

Using the confidence scores and the temporal derivatives, we detected the potential outliers and added them to the annotation set. The confidence difference is show in Fig. 9. Using the *scipy.signal.find_peaks* package, we detected the outliers and the generated output is show in Fig. 10.

4. EXPERIMENTS

The proposed framework is implemented in Python with the support of Keras backend by Tensorflow. The processing is performed on Nvidia P-100 with 32 GB RAM.

5. CONCLUSION

Since we had a constraint environment in the data set, we had almost similar backgrounds, camera angels, and color of objects (pigs), hence the model showed larger tendency to learn as compare to training the model on the dataset gathered for the objects in the wild. By applying the K-Means sampler, we were able to train the model on a bigger data distribution, hence eliminating redundant data frames. This helped us to filter out unique data frames. By using StackedDensenet pretrained weights to train our model, even providing the training set of just 280 images, we got very promising results.

6. APPENDICIES

In this section, we mentioned some resources that we used and the techniques we implied that did not work out as expected. Following the summary of the process and the workdone that resulted in a deadlock.

6.1. Environment

The first step we did was creating a Conda environment on Ubuntu operating system. The second step was to install all the dependencies and packages that were needed in the environment for running the pose estimation tensor flow. The code we used was from this Github repository [38]. We were able to do pose estimation on human images and human videos using the CMU model by running this very code-base.

6.2. Keypoints

A total of 5 key points were being tracked for the pigs. Key points that were being tracked were Nose, Neck, Tail, Left Ear, Right Ear. We had data of 2800 images of pigs. So the next step was to annotate the data.

6.3. Data Annotations

We used an online annotator for annotating the dataset. The annotator can be found at this link [39]. We had to upload one image at a time in the annotator. We had to select all the key points and set the bounding box on every pig in the image. The annotator then saved all the labels in the COCO JSON format.

6.4. Training Model

We started to train the model on the data we gathered from the annotator. We trained the model after doing necessary customization in the code-base available at the Github repository [40]. At the start of the training, the loss was at 270 and by the end of 140th iteration, the loss was at 22. Hence the model showed a great tendency of learning. After the training, the results were in .ckpt format which we could not use in the existing codebase. Although we were able to generate Protobuf format due to the difference in the model architecture, we could not generate any results while using it.

7. REFERENCES

[1] Sultan Daud Khan, Habib Ullah, Mohammad Uzair, Mohib Ullah, Rehan Ullah, and Faouzi Alaya Cheikh, “Disam: Density independent and scale aware model for crowd counting and localization,” in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4474–4478.

[2] Habib Ullah, Ahmed B Altamimi, Muhammad Uzair, and Mohib Ullah, “Anomalous entities detection and localization in pedestrian flows,” *Neurocomputing*, vol. 290, pp. 74–86, 2018.

[3] Jun Yang, Zhongke Shi, and Ziyan Wu, “Vision-based action recognition of construction workers using dense trajectories,” *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 327–336, 2016.

[4] Sultan Daud Khan, Habib Ullah, Mohib Ullah, Nicola Conci, Faouzi Alaya Cheikh, and Azeddine Beghdadi, “Person head detection based deep model for people counting in sports

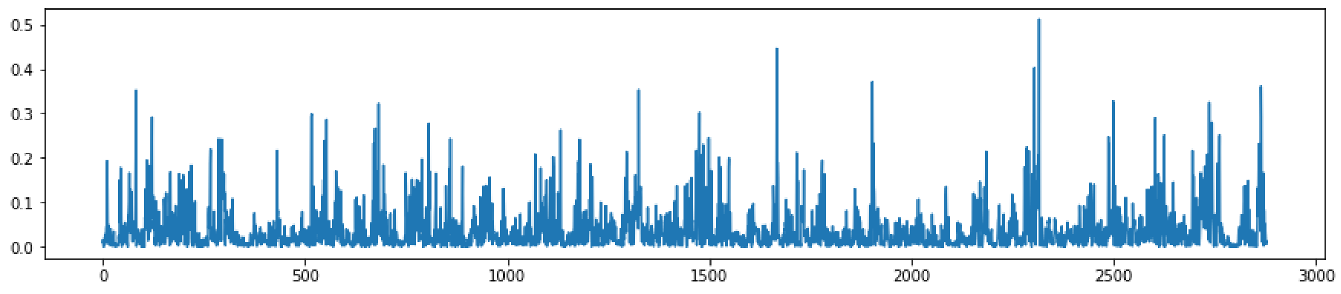

Fig. 9: Confidence Difference

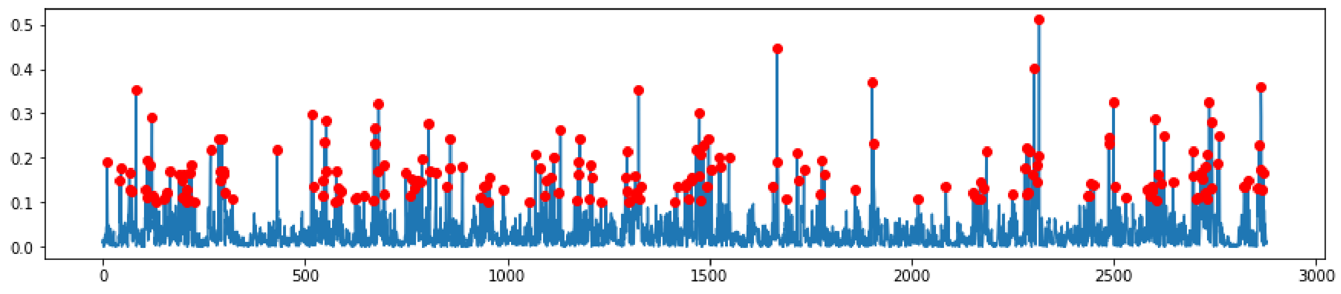

Fig. 10: Outliers

videos," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–8.

[5] Mohib Ullah, Habib Ullah, Nicola Conci, and Francesco GB De Natale, "Crowd behavior identification," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1195–1199.

[6] Jarissa Maselyne, Ines Adriaens, Tjebbe Huybrechts, Bart De Ketelaere, Sam Millet, Jurgen Vangeyte, Annelies Van Nuffel, and Wouter Saeys, "Measuring the drinking behaviour of individual pigs housed in group using radio frequency identification (rfid)," *animal*, vol. 10, no. 9, pp. 1557–1566, 2016.

[7] Mohib Ullah, Habib Ullah, Sultan Daud Khan, and Faouzi Alaya Cheikh, "Stacked lstm network for human activity recognition using smartphone data," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2019, pp. 175–180.

[8] Ian W Pray, Dallas J Swanson, Viterbo Ayvar, Claudio Muro, Luz M Moyano, Armando E Gonzalez, Hector H Garcia, Seth E ONeal, Cysticercosis Working Group in Peru, et al., "Gps tracking of free-ranging pigs to evaluate ring strategies for the control of cysticercosis/taeniasis in peru," *PLoS neglected tropical diseases*, vol. 10, no. 4, pp. e0004591, 2016.

[9] Jianguo Chen, Kenli Li, Qingying Deng, Keqin Li, and S Yu Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[10] Habib Ullah, *Crowd Motion Analysis: Segmentation, Anomaly Detection, and Behavior Classification*, Ph.D. thesis, University of Trento, 2015.

[11] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing, "Maskrnn: Instance level video object segmentation," in *Advances in Neural Information Processing Systems*, 2017, pp. 325–334.

[12] Mohib Ullah, Habib Ullah, and Ibrahim M Alseadonn, "Human action recognition in videos using stable features," 2017.

[13] Habib Ullah, Mohib Ullah, and Muhammad Uzair, "A hybrid social influence model for pedestrian motion segmentation," *Neural Computing and Applications*, pp. 1–17, 2018.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[15] Habib Ullah, Muhammad Uzair, Mohib Ullah, Asif Khan, Ayaz Ahmad, and Wilayat Khan, "Density independent hydrodynamics model for crowd coherency detection," *Neurocomputing*, vol. 242, pp. 28–39, 2017.

[16] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[17] Sultan Daud Khan, Habib Ullah, Mohib Ullah, Faouzi Alaya Cheikh, and Azeddine Beghdadi, "Dimension invariant model for human head detection," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2019, pp. 99–104.

[18] Wei Yu, Xiaoshuai Sun, Kuiyuan Yang, Yong Rui, and Hongxun Yao, "Hierarchical semantic image matching using cnn feature pyramid," *Computer Vision and Image Understanding*, vol. 169, pp. 40–51, 2018.

[19] Mohib Ullah, Habib Ullah, and Faouzi Alaya Cheikh, "Single shot appearance model (ssam) for multi-target tracking," *Electronic Imaging*, vol. 2019, no. 7, pp. 466–1, 2019.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[21] Habib Ullah, Sultan Daud Khan, Mohib Ullah, Faouzi Alaya Cheikh, and Muhammad Uzair, "Two stream model for crowd video classification," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2019, pp. 93–98.

[22] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[23] Mohib Ullah, Faouzi Alaya Cheikh, and Ali Shariq Imran, "Hog based real-time multi-target tracking in bayesian framework," in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2016, pp. 416–422.

[24] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang, "Deep high-resolution representation learning for human pose estimation," *arXiv preprint arXiv:1902.09212*, 2019.

[25] Mohib Ullah and Faouzi Alaya Cheikh, "A directed sparse graphical model for multi-target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1816–1823.

[26] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu, "Rmpe: Regional multi-person pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2334–2343.

[27] Mohib Ullah, Ahmed Kedir Mohammed, Faouzi Alaya Cheikh, and Zhaohui Wang, "A hierarchical feature model for multi-target tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2612–2616.

[28] Alexander Mathis, Mert Yksekgnl, Byron Rogers, Matthias Bethge, and Mackenzie W. Mathis, "Pretraining boosts out-of-domain robustness for pose estimation," 2019.

[29] Alexander Mathis, Pranav Mamidanna, Taiga Abe, Kevin M Cury, Venkatesh N Murthy, Mackenzie W Mathis, and Matthias Bethge, "Markerless tracking of user-defined features with deep learning," *arXiv preprint arXiv:1804.03142*, 2018.

[30] Mohib Ullah and Faouzi Alaya Cheikh, "Deep feature based end-to-end transportation network for multi-target tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3738–3742.

[31] Abozar Nasirahmadi, Sandra A Edwards, and Barbara Sturm, "Implementation of machine vision for detecting behaviour of cattle and pigs," *Livestock Science*, vol. 202, pp. 25–38, 2017.

[32] Saira Kanwal, Muhammad Uzair, Habib Ullah, Sultan Daud Khan, Mohib Ullah, and Faouzi Alaya Cheikh, "An image based prediction model for sleep stage identification," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 1366–1370.

[33] Lorre S Atlan and Susan S Margulies, "Frequency-dependent changes in resting state electroencephalogram functional networks after traumatic brain injury in piglets," *Journal of neurotrauma*, 2019.

[34] Alexandra F da S Cordeiro, Irenilza de A Nääs, Felipe da Silva Leitão, Andréia CM de Almeida, and Daniella Jorge de Moura, "Use of vocalisation to identify sex, age, and distress in pig production," *Biosystems engineering*, vol. 173, pp. 57–63, 2018.

[35] Eric T Psota, Mateusz Mittek, Lance C Pérez, Ty Schmidt, and Benny Mote, "Multi-pig part detection and association with a fully-convolutional network," *Sensors*, vol. 19, no. 4, pp. 852, 2019.

[36] Jacob M Graving, Daniel Chae, Hemal Naik, Liang Li, Benjamin Koger, Blair R Costelloe, and Iain D Couzin, "Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning," *eLife*, vol. 8, pp. e47994, 2019.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[38] "tf-pose-estimation," `https://imglab.in/#`, Accessed: 2019-11-20.

[39] "Img Lab annotation tool," `https://github.com/ildoonet/tf-pose-estimation`, Accessed: 2019-11-02.

[40] "Simple baselines for human pose estimation and tracking," `https://github.com/mks0601/TF-SimpleHumanPose`, Accessed: 2019-12-04.