LOK1 - Object Localization with the MPII Human Pose Dataset

Micha Birklbauer
University of Applied Sciences, Campus Hagenberg
Master Student, Data Science and Engineering
Hagenberg, Austria
micha.birklbauer@students.fh-hagenberg.at

Nicole Hölzl
University of Applied Sciences, Campus Hagenberg
Master Student, Data Science and Engineering
Hagenberg, Austria
nicole.hoelzl@students.fh-hagenberg.at

***Abstract*—Object localization is a critical step in subsequent identification and classification. This paper proposes and compares three different methods for localizing persons in different poses in images. To be more precise as a data basis the MPII Human Pose Dataset is used and Haar Cascade Localization as well as deep neural networks like YOLOv3 and EfficientDet are tested. The results show that those pre-trained networks not only perform best but also are more simple to implement and apply. Code for this paper is made available via**

https://github.com/nhoelzl/object_localisation.

***Index Terms*—computer vision, object localization, yolov3 net, haar cascade, efficientdet**

I. Introduction

The progress in hardware and telecommunication technologies has resulted in a rapid increase in multimedia information - namely videos and images predominately. Extracting semantic meaning and interpretation of that information falls into the domain of computer vision where object localization is a vital task and a requirement for subsequent identification and classification of objects [1].

A. Motivation

Currently many models trained on different image datasets already exist for object localization and classification. While research is pushing for better and faster models the focus of this paper is how one can train their own model or apply pre-trained models on a dataset featuring persons in different poses - that might not be found in standard image databases [2].

B. Problem Statement

The dataset at hand offers a wide variety of images with persons in different poses, different sizes and at various different activities. Localization of these persons is in many cases not trivial since their pose - meaning the position of body parts like legs and arms - may be completely abnormal and unobserved in general purpose image datasets. Finding the best enclosing bounding box is of vital interest for further image analysis.

C. State of the Art

At this point in time the current state of the art is going in the direction of deep neural networks like YOLOv3 which are able to draw precise bounding boxes as well as predictions for a great number of classes. Those networks perform well on a wide selection of images but are trained to be more general and not specifically to detect persons in different poses [2] [3].

D. Overview of the Solution

In this paper two approaches are proposed: Firstly manually training a Haar Cascade Classifier and evaluating it in terms of mean average precision on a test dataset. Furthermore the Haar Cascade Classifier should serve as a baseline for more complex models. Secondly two different state of the art pre-trained models are applied to the same test dataset to compare their performance to each other as well as the Haar Cascade Classifier.

II. Basics

The following approaches are realized using several different kinds of tools and and all notebooks, scripts and artifacts are made available via GitHub. Generally however, most implementations are done in python 3.8.5 using OpenCV natively as well as the python bindings. All neural networks are implemented in TensorFlow 2.4 or loaded via TensorFlow Hub [4] [5]. A complete specification of requirements, setting up an environment and hardware information can be found in the above mentioned repository.
Moreover at this point the general workflow should be outlined:

1) Everything starts with the data: Sampling, annotation and augmentation.
2) Training of models using a training split of the data. This is only done if the model needs to be trained of course, for pre-trained models this step is skipped.
3) Evaluation of each model on test split of the data.
4) Comparison of models by visual ”eye test” and more importantly in terms of quality metrics like mean average precision (mAP).

III. Material

This section covers the used data basis as well as important aspects of image annotation and transformations for image augmentation.

A. Data

The underlying data basis was the MPII Human Pose Dataset which features over 25 000 images extracted from YouTube videos. The dataset also includes over 40 000 different people and has annotated body joints. Furthermore 410 human activities are represented in the data [6]. Only a sample of the data was used for training and testing of the subsequently described models for two reasons: Firstly, the amount of images was too big for the setup and hardware that was available. Secondly, the bounding boxes had to be manually annotated to establish a ground truth because they cannot be automatically inferred from body joints e.g. even if a finger has the leftmost x-coordinate in the annotation it could still be that another body part which is not referenced in the data has a smaller x-value. Considering those aspects 267 images of single persons were sampled - 217 for training and 50 testing.

B. Annotation

Henceforth the 267 images needed to be annotated and for that purpose the python application "labelImg" was the tool of choice. "labelImg" is an open-source program and freely available via GitHub and pip [7]. Because it is implemented in python it is also platform independent and for ease of usage offers a convenient graphical user interface. Annotations are saved in XML files that can easily be parsed.

C. Augmentation

To further enrich the training data the 217 annotated images where augmented in python using the package "albumentations". In "albumentations" augmentation pipelines can be defined to automatically generate new images and corresponding bounding boxes [8]. The following transformations were applied to the training images:

1) Horizontal flipping of the image.
2) Vertical flipping of the image.
3) Rotation of the image randomly between -90 and +90 degrees.
4) Transforming the image to gray scale.
5) Transforming the image to sepia colours.
6) Inverting the image.
7) Posterizing the image e.g. reducing the bits per channel.
8) Normalizing the image using the mean and standard deviation of all channels of all training images.

Transforming the images resulted in 1953 available samples, meaning almost 10 fold of the original sample size.

IV. Methodology

A. Haar Cascade

Haar Cascade Classifiers are prominently used for face detection but can be trained to detect arbitrary objects. During training so called Haar features like convolution kernels of different scale, orientation and location are applied to the images' region of interest. This way the Haar kernels detect different characteristics of the desired object e.g. for a person this could be arms or legs. Due to the fact that many features need to be evaluated at all positions and scale Haar Cascade training is a computationally power-hungry approach that needs a lot of resources and time for calculation. To leverage that the Adaboost algorithm is applied and features are checked by significance in a multi-stage approach (cascade) hence the name [9].

Training of a Haar Cascade Classifier requires annotated positive images e.g. pictures of humans in different poses with corresponding bounding boxes, but also negative images without the object that is to be recognized - in this case without persons. For this purpose another image dataset was used that features mostly images of nature and landscaping. The sample size of this dataset was 300 images [16].

The Haar Cascade Classifiers were trained using OpenCV and the corresponding bindings for python. Moreover two different approaches were tried: Firstly training the Haar Cascade Classifier with the original 217 training images and secondly, training it with the 1953 augmented images.

B. YOLO

YOLOv3, so called - "You only look once" - is a fast and accurate deep learning network using an object detection approach in the third generation. The model applies a single neural network to the full 416 x 416 image and divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities using Logistic Regression. YOLOv3 looks at the whole image at test time so its predictions are performed on the whole image context [15]. The latest model version predicts bounding boxes on images with dimension clusters as anchor boxes. For each dimension, there are 3 anchors which can be adjusted manually. The network predicts 4 coordinates for each box [3]. This approach got adapted in the following versions of YOLO [15].

The backbone of the YOLOv3 model is a variant of Darknet which uses 53 layers trained on ImageNet as shown in Figure 1. Additionally 53 layers help to detect objects in images. In total, 106 fully convolutional layers build the underlying architecture of YOLOv3. The model predicts at three scales, which are given by downsampling the dimensions of the input image by 32, 16 and 8 [3]. See also Figure 2 for reference.

The model is 3x faster than RetinaNet on the COCO 50 Benchmark dataset [17]. The previous version of the Darknet-19 architecture achieved a performance of 74.1% in the top-1 category whereas Darknet-53 improved slightly to 77.2% on ImageNet. On the COCO 75 benchmark dataset, RetinaNet detects bounding boxes more precisely and accurate than YOLOv3. The YOLOv3 model is pre-trained on the COCO dataset and available at [14] and [15] and applied without adaption of the anchors.

C. EfficientDet

EfficientDet is a relatively new deep learning approach to image detection that focuses on efficiency, meaning not only smaller network architecture but also faster inference time.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 1. Darknet-53 Architecture - An overview of the design of the Darknet-53 network [3].

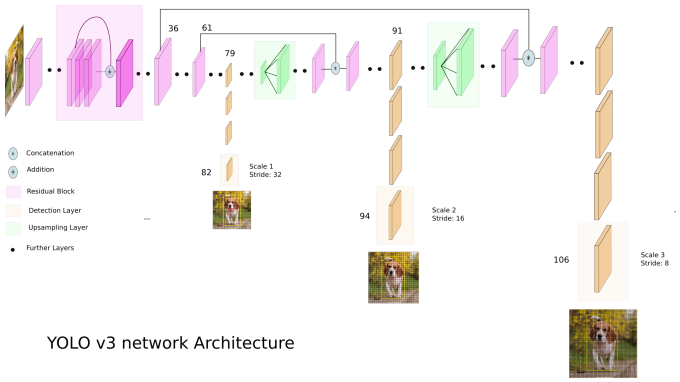

Fig. 2. YOLOv3 Architecture - An overview of the design of the YOLOv3 model [3].

EfficientDet follows an one-stage or single shot detection approach where objects in images are detected in a single stage rather than following a regional proposal approach which need two stages to detect objects [10].

The backbone of the EfficientDet is the EfficientNet, a convolutional neural network (CNN) that encapsulates uniformly scaled depth, width and resolution by a compound coefficient. This network architecture achieves 84.3% top-1 accuracy on ImageNet, while simultaneously being 8.4 times smaller and 6.1 times faster on inference than the best existing CNN [11].

For feature fusion a Bi-directional Feature Pyramid Network (BiFPN) is proposed where the weights are learned by a pyramid attention network contrary to the popular approach of simply resizing and summing the features up [2]. See also Figure 3.

Finally the classes and bounding boxes are predicted by two separate convolutional networks that share the same input features [2]. An overview of the complete EfficientDet architecture can be found in Figure 4.

The EfficientDet model is pre-trained and available via TensorFlow Hub where it can be downloaded and applied without adaptation.

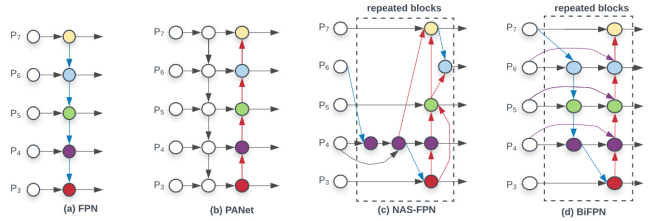

Fig. 3. Feature Network Design - Feature networks of different object detection architectures in comparison. The rightmost (d) BiFPN is applied to EfficientDet. P denotes the feature space at the subscripted level e.g. if the input resolution is 640x640 pixels then P_3 represents feature space level 3 with resolution 80x80 ($640/2^3 = 80$) [2].

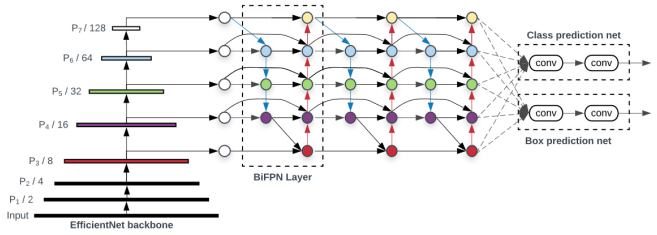

Fig. 4. EfficientDet Architecture - An overview of the design of the EfficientDet network [2].

D. Mean Average Precision

To compare the Haar Cascade Classifiers, YOLO and EfficientDet the mean average precision (mAP) was calculated for each model. To be more precise, the mAP is the mean of all average precision (AP) values for all classes. Since the scope of this work is to detect only one class (persons) AP and mAP are the same. The AP is calculated as the number of true positive (TP) samples over the sum of true positives and false positives (FP). Moreover a prediction is considered TP if the Jaccard Index of that prediction is greater than 0.5 and in terms of bounding boxes that means that the ratio of intersection over union is greater than 0.5 [13].

V. Results

The following section covers a sample of the manual ”eye test” as well as an accurate description of the result in terms of mAP for the test data.

A. Visual Comparison

The pictures in Figure 5 to Figure 8 show the bounding boxes for the manually annotated image (ground truth) in comparison to the different models. The Haar Cascade Classifier that was trained on augmented images is not shown since it performed significantly worse than the Classifier that was trained on the original images. Even though that judging from the "eye test" there is some common ground between the models and the ground truth, one can still already notice some differences like that the Haar Cascade Classifier underestimates the borders of the bounding box while the YOLOv3 net overestimates the borders a bit. From the visual inspection we concluded that the EfficientDet comes closest to the ground truth.

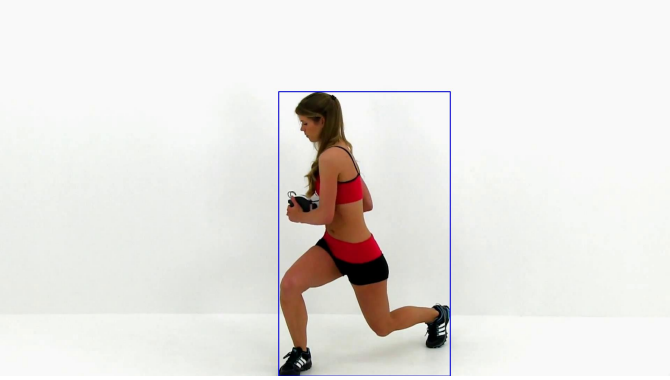

Fig. 5. Ground Truth - Bounding box as annotated manually.

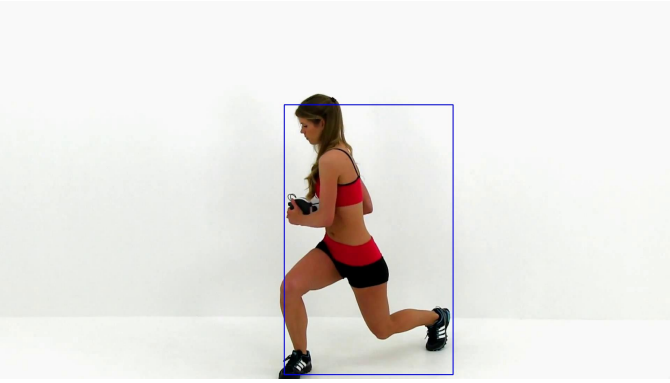

Fig. 6. Haar Cascade Classifier - Bounding box prediction of the Haar Cascade Classifier that was trained on the original manually annotated images.

B. Mean Average Precision

Even though a visual comparison is not necessarily wrong, an unbiased comparison in terms of mAP is a lot more scientific. Table I shows the values for mean average precision for all the considered models. It is immediately visible that Haar Cascade performs significantly worse than both deep neural network approaches and that even though that EfficientDet looked better in the "eye test" YOLOv3 performs exactly as good on the established test partition of the dataset.

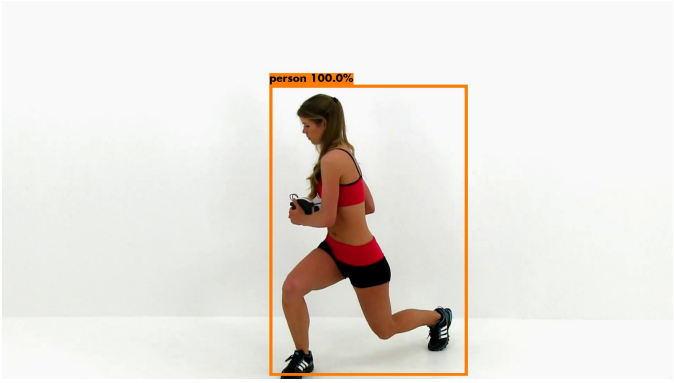

Fig. 7. YOLOv3 - Bounding box and class prediction of YOLOv3.

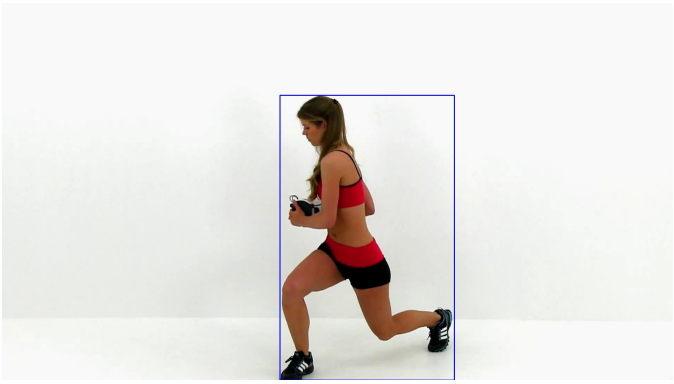

Fig. 8. EfficientDet - Bounding box prediction of EfficientDet.

VI. Discussion

Over the course of this project several challenges as well as surprises were encountered that we want to address in the following.

A. Annotation Challenges

Manual annotation of the images proved to be no trivial task since the images are ambiguous in several aspects:

1) A lot of activities happen in an environment that features mirrors and so it happens that a lot of the images in the MPII Human Pose Dataset contain not only the person but at the same time a mirror image of that person. We were unsure if we should annotate the mirror image as a person or not.
2) Often pictures are of poor quality, blurred or simply screenshots between two frames were even we were unable to draw an exact bounding box.

TABLE I
Model Comparison - Mean Average Precision

Model	**mAP**
EfficientDet	54.55
YOLOv3	54.55
Haar Cascade	03.26
Haar Cascade with Data Augmentaion	01.82

3) Many pictures have an exorbitant amount of persons in them which makes it not only impossible to annotate but we were also unsure of how we should compare found bounding boxes in those cases. This was also one of the reasons why we sampled images of single persons as mentioned in the third chapter of this paper.
4) Most of the activities also include some kind of sports equipment and it is questionable if one sees that as part of the body/person or not. For instance, a bicycle would not count as an extension of that body but what a baseball glove count as such? That would be debatable.

B. Haar Cascade Results

Contrary to expectation and also not really visible in the sample image shown in Figure 6 both of the Haar Cascade Classifiers yielded a really bad mAP. Furthermore the Classifier that had more training data available (the original images plus the augmented ones) even performed worse than the Classifier that only had those 217 manually annotated images for training. Our guesses to explain that behaviour are that it was simply not enough data to train a good model. Roughly 200 positive samples and 300 negative samples is most probably not sufficient to train a good Haar Cascade Classifier for detecting persons in different poses. Moreover it could also be that the negative samples are not diverse enough as they only show pictures of nature and landscapes and exclude any kind of urban or home environment. Last but not least it seems that augmenting the pictures with the transformations that we used did not help in training the Haar Cascade model, contrary is the case as the mAP - which was already small - dropped almost by half.

C. Comparison of COCO mAP to MPII mAP

On the other hand the pre-trained models performed really well with mAP of over 50% for both. Since both of those models were trained on the COCO dataset one can compare the mAP of COCO to that of our human pose dataset. YOLOv3 had a mAP of 55.3% on the COCO dataset, meaning in comparison it performed worse on our data. Again we are left guessing why that is the case but possible explanations could be that the COCO dataset is just not as representative of different human poses that are found in our images. It could also be that the bounding box annotations for COCO are slightly different to those we used in our manual annotation or it could also be just noise. However both of these aspects are more or less contradicted by the fact that EfficientDet was also trained on COCO (45.4% mAP) and performed better on our data. Which finally leads us to the conclusion that it seems that EfficientDet is more general that YOLOv3 and therefore over-performing on our data while YOLOv3 is under-performing on our data [17] [2] [3].

Conclusion and Outlook

Training a model for object localization from zero is no trivial task. Manually annotating images is not only tedious but can also be challenging if one does not set clear rules for drawing the bounding boxes. Even with data augmentation training and test results do not necessarily get better and transformation methods possibly have to be adapted depending on the model that is to be trained. However many "out-of-the-box" and state of the art solutions already exist in the form of pre-trained models that are freely available and can be adapted with minimal effort. The results show that even for a dataset that features humans in all kinds of poses and different body orientations the predictions of bounding boxes of said models still work fairly well. Last but not least one could also try to not only use pre-trained models for prediction but also for transfer-learning and fine tuning on one's own images - however this requires a lot of data, computational resources and time, which, to conclude this paper, was unfortunately not feasible for us and the scope of this project.

References

[1] Dasiopoulou, Stamatia, Mezaris, Vasileios, et al. "Knowledge-Assisted Semantic Video Object Detection." IEEE Transactions On Circuits And Systems For Video Technology, 2005.
[2] Tan, Mingxing, Pang , Ruoming and Quoc V. Le. "EfficientDet: Scalable and Efficient Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.
[3] Farhadi, Ali, and Joseph Redmon. "Yolov3: An incremental improvement." Computer Vision and Pattern Recognition, 2018.
[4] Bradski, Gary. "The OpenCV Library." Dr Dobb's Journal of Software Tools, 2000.
[5] Abadi, Martín, Agarwal, Ashish, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems." Google LCC, 2015. Software available from tensorflow.org.
[6] Andriluka, Mykhaylo, Pishchulin, Leonid, et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis." IEEE Conference on Computer Vision and Pattern Recognition, 2014.
[7] tzutalin (2020) labelImg [Source code]. https://github.com/tzutalin/labelImg.
[8] Albumentations Team (2020) Albumenations [Source code]. https://github.com/albumentations-team/albumentations.
[9] Viola, Paul, Jones, Michael. "Rapid Object Detection using a Boosted Cascade of Simple Features." Computer Vision and Pattern Recognition, 2001.
[10] Liu, Wei, Anguelov, Dragomir, et al. "SSD: Single Shot MultiBox Detector." Computer Vision and Pattern Recognition, 2016.
[11] Tan, Mingxing, Le, Quoc. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." International Conference on Machine Learning, 2019.
[12] Lin, Tsung-Yi, Goyal, Priya, et al. "Focal Loss for Dense Object Detection." Computer Vision and Pattern Recognition, 2018.
[13] Everingham, Mark, Van Gool, Luc, et al. "The PASCAL Visual Object Classes Challenge." Springer Science + Business Media LLC, 2009.
[14] shahkaran76 (2018) [Yolov3 Source Code]. https://github.com/shahkaran76/yolo_v3-tensorflow-ipynb.
[15] pjreddie (2020) [Yolo Blog]. https://pjreddie.com/darknet/yolo/.
[16] Bleys, Joris and Tony Belpaeme. "Datasets of natural and urban images." VUB Artificial Intelligence Lab Brussels, 2006. Dataset available from: arti.vub.ac.be/research/colour/data/imagesets.zip
[17] Lin, Tsung-Yi, Maire, Michael, et al. "Microsoft COCO: Common Objects in Context." Computer Vision and Pattern Recognition, 2015.