Comparing Boosted Cascades to Deep Learning Architectures for Fast and Robust Coconut Tree Detection in Aerial Images

Steven Puttemans*, Kristof Van Beeck* and Toon Goedemé
KU Leuven, EAVISE Research Group, Jan Pieter De Nayerlaan 5, Sint-Katelijne-Waver, Belgium

Keywords: Object Detection, Boosted Cascades, Deep Learning.

Abstract: Object detection using a boosted cascade of weak classifiers is a principle that has been used in a variety of applications, ranging from pedestrian detection to fruit counting in orchards, and this with a high average precision. In this work we prove that using both the boosted cascade approach suggest by Viola & Jones and the adapted approach based on integral or aggregate channels by Dollár yield promising results on coconut tree detection in aerial images. However with the rise of robust deep learning architectures for both detection and classification, and the significant drop in hardware costs, we wonder if it is feasible to apply deep learning to solve the task of fast and robust coconut tree detection and classification in aerial imagery. We examine both classification- and detection-based architectures for this task. By doing so we prove that deep learning is indeed a feasible alternative for robust coconut tree detection with a high average precision in aerial imagery, keeping attention to known issues with the selected architectures.

1 INTRODUCTION

Getting a robust and accurate location of any object in a given input image, combined with a correct label, is a key part in solving many automation tasks. In most cases the localization is only a small part of the complete pipeline, thus requiring a very high accuracy, in order to reduce the propagated error through the remaining pipeline as much as possible.

Large companies use human annotators in low-cost countries to manually analyse every single image, locating and labelling each instance of specific object classes, by manually augmenting the given image with bounding boxes of the object instances. In our application of coconut tree detection in aerial images *(see Figure 1)*, the human annotators are asked to click on the coconut tree centres, after which a circle with a predefined average diameter *(which can be defined due to the fixed height at which the images are captured using aerial photography)* is placed as annotation on top of the coconut tree. An example of such a labelled image can be seen in Figure 2.

This manual annotation is a cumbersome, time-consuming and expensive process. Furthermore, to avoid bias introduced by a single annotator, we need to incorporate annotation redundancy, by forcing multiple annotators to process the same image and then averaging out on the annotation locations. Additionally, manual annotation is very prone to mistakes when performing these repetitive tasks *(e.g. missing coconut trees, selecting wrong locations, . . .)*.

Many of these tasks could be automated given the possibilities of state-of-the-art object detection algorithms. These systems can, given pre-trained models on labelled training data, robustly locate objects in a

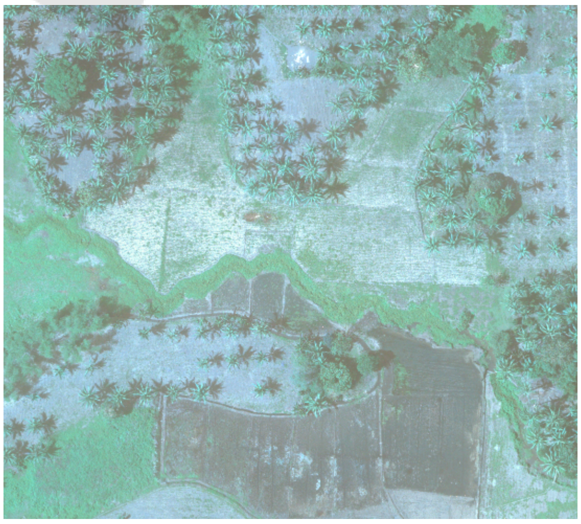

Figure 1: Example aerial image captured through remote sensing equipment in which we need to detect coconut trees.

[*] Both authors equally contributed to this work.

Puttemans, S., Beeck, K. and Goedemé, T.
Comparing Boosted Cascades to Deep Learning Architectures for Fast and Robust Coconut Tree Detection in Aerial Images.
DOI: 10.5220/0006571902300241
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 5: VISAPP*, pages 230-241
ISBN: 978-989-758-290-5

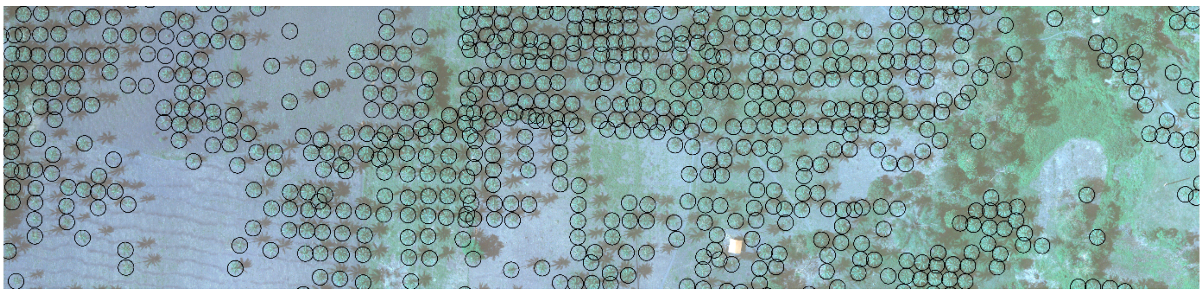
Figure 2: Example of an aerial image containing manually annotated coconut trees.

given input image with human-like accuracy using the power of machine learning. Compared to human annotators these repetitive tasks are just perfect for automated systems, which can heavily parallelize these tasks and look for multiple object instances at once.

The challenging part lies in finding the correct algorithm for training these accurate machine-learned object detection models. This is where we position this research on finding the optimal solution for automated and robust coconut tree detection. We are convinced that several object detection algorithms based on the principle of a boosted cascade of weak classifiers (Viola and Jones, 2001; Dollár et al., 2009; Dollár et al., 2010) are a perfect solution for this task. On the other hand, deep learning algorithms have introduced a new wave of state-of-the-art object detectors, capable of achieving top-notch accuracy results. Combined with the fact that the required GPGPU hardware is becoming affordable and the fact that many pre-trained networks already exist *(e.g. caffe model zoo (Jia et al., 2014))*, it seems a valid alternative to the boosted cascade classifier based approaches.

In this work we compare the well-known cascade classifier object detection algorithms to these new and powerful deep-learned object detection algorithms. We evaluate the trained detection models both in achieved accuracy and execution speed, keeping in mind that achieving real-time performance is in many cases a hard constraint for the actual application. Furthermore we give recommendations on how to efficiently use deep learning algorithms in the context of object detection in aerial images and propose some general rules to keep in mind.

Initial reading on the topic of deep learned object detection shows us that training models from scratch without large amounts of training data and expensive hardware is infeasible, thus we examine the subject of transfer learning, where we adapt existing deep learned models, trained on larger datasets *(e.g. Pascal VOC (Everingham et al., 2010), Microsoft COCO (Lin et al., 2014))* to suit our specific object detection needs, by fine-tuning the weights of all the convolutional layers onto new object classes.

The remainder of this paper is organized as follows. Section 2 discusses the state-of-the-art in object detection and takes a look at similar research performing object detection in aerial imagery. This is followed by section 3 where the collected data for training and validating our object detection solutions is discussed. Section 4 and 5 discuss the different approaches we suggest for coconut tree detection. The achieved results are subsequently discussed in section 6, followed by conclusions in section 7.

2 RELATED WORK

The principle of a boosted cascade of weak classifiers is introduced by (Viola and Jones, 2001), where it is used to obtain a robust face detection algorithm based on the very simple and weak HAAR-wavelet like features. By letting the boosting process decide which features are good at separating object from non-object patches, they obtain a robust classifier with a limited set of features. Furthermore the concept of a cascade, which allows for the early rejection of non-object patches, increases execution speeds of this algorithm drastically.

(Dollár et al., 2009) argues that dropping colour information and sticking to a single invariant feature representation might limit the possibilities of boosted cascade classifiers. The paper suggests using multiple feature representations, called integral channels (ICF), ranging from Gabor filters, to edge filters, colour filters, ...all with the aim to improve the detection quality of boosted cascades. By doing so they significantly improve the accuracy of pedestrian detection. In (Dollár et al., 2010) they introduce a set of extensions and the concept of aggregated channel features (ACF), pushing the achieved accuracy on pedestrian detection even further.

Even though algorithms based on a boosted cascade of weak classifiers are already a bit older, there is ongoing research showing that these techniques are still valid solutions for existing industrial problems (Puttemans et al., 2016a; Puttemans et al., 2016b; Zheng et al., 2016; Shaikh et al., 2016). These solutions offer high accuracies, by smartly using scene- and application-specific constraints to improve the efficiency of the boosted cascade algorithms, while maintaining real-time processing speeds.

Since 2015 deep learning frameworks are taking over the state-of-the-art in both object classification and object detection tasks. Due to the rise of enormous datasets and the drop in hardware cost, deep learning becomes a valid alternative for any classic machine learning task. With the introduction of pre-trained networks like AlexNet (Krizhevsky et al., 2012), InceptionV3 (Szegedy et al., 2016), DenseNet (Iandola et al., 2014), ResNet (Targ et al., 2016), ...classification results on challenging datasets like ImageNet (Deng et al., 2009) rise to the top.

On the detection part, interesting approaches are suggested, based on combining the above classification networks with a multi-scale sliding window based approach (Szegedy et al., 2013). However, the downside of these approaches is the tremendous amount of region proposals due to the multi-scale sliding window approach, which results in computationally expensive algorithms. To tackle this issue, region proposal networks are proposed as a pre-filter for the CNN classification pipeline (Ren et al., 2015). Even though several parts of the region proposal network can be shared with the subsequent classification network, this approach still needs two separate architectures, which need to be tuned individually.

Finally, the current state-of-the-art in object detection is found in single-pass deep learned object detectors. These algorithms integrate the region proposal approach directly as a layer inside the classification pipeline, thus only requiring a single pass of the network for detecting all objects in the given image. Examples of these state-of-the-art algorithms are the ‘Single Shot Multibox’-detector (Liu et al., 2016) and the ‘You-Only-Look-Once’-detector (Redmon et al., 2016). These efficient networks reduce the amount of region proposals so drastically that obtaining real-time performance is achievable, even rising to up to 120 FPS at VGA resolution.

3 DATASET AND FRAMEWORK

As training and validation data for our suggested approaches we make use of a 10.000×10.000 pixel aerial image covering a partial coconut plantage provided in RGBA format. Inside the image all coconut trees are manually annotated using their centre position. The average size of a patch covering the whole tree is 100×100 pixels, so this size is used as annotation patch around the centre position. The image contains 3798 coconut tree patches, while the remaining image parts are used as background information.

In all cases we split the provided data into parts, using one part for model learning and the other part for model validation, to ensure the detector is not validated on actual training data. Specific data splits are discussed at each separate technique.

To train our object detection models we use three publicly available frameworks. The first boosted cascade approach, based on the principle of (Viola and Jones, 2001) combined with LBP features, is trained using the implementation of the OpenCV3.2 framework (Bradski and Kaehler, 2000). The second boosted cascade approach, based on the principle of (Dollár et al., 2010), using the aggregate channel features approach, is trained using the authors own MATLAB toolbox (Dollár, 2005). For our deep learned models we start by using an implementation of the *InceptionV3* architecture in Tensorflow (Abadi et al., 2015), but quickly switched to the C and CUDA based framework Darknet (Redmon, 2013), which includes both the classification *(Darknet19,Densenet201)* and detection architectures *(YOLOv2)* we further test in our paper.

4 APPROACHES WITH BOOSTED CASCADES

In this section we discuss the different boosted cascade based approaches we used for training our coconut tree detectors, combined with the specific number of training and validation samples used.

4.1 An Adaptive Boosted Cascade of Weakly Trained Classifiers

Our first approach (Viola and Jones, 2001) is a boosted cascade of weak classifiers using the adaptive boosting algorithm (Margineantu and Dietterich, 1997) for learning the weak classifiers, based on the local binary pattern (LBP) feature representation (Ahonen et al., 2004). This invariant feature representation ignores colour information and works directly on a grayscale image, focusing on local differences in pixel intensities. We choose to work on a grayscale image since no obvious separation between coconut trees and background *(e.g. grasslands, other vegetation types, ...)* seems achievable using colour information. If however we want to detect another object

Table 1: Training data for the Viola&Jones based detection models and the model complexity *(expressed by the number of stages and weak stump classifiers).*

	#pos	#neg	#weak	#feats
Model 1	1000	2500	16	126
Model 2	1000	5000	15	123
Model 3	1000	10000	15	142
Model 4	2000	8000	16	221

class, with distinct colour based features, we would first apply a colour-based transformation on our input data, as suggested by (Puttemans et al., 2016b).

For training the detection model we split the source image in four equal parts. The annotations of the top left image part are used as positive training samples, while the remaining image parts are used for validation. As background training patches we randomly sample patches at the model size, from the image, not containing actual coconut trees. We increase the number of negative samples with each model, to obtain a more accurate detector with less false positive detections, achieving a higher average precision with each step, as discussed in section 6.

On top of the gathered training samples we apply data augmentation for our final model, by randomly flipping the training patches around their vertical or horizontal axis. The amount of training samples used for each model can be seen in Table 1, together with the amount of stages of weak classifiers and the actual number of weak classifiers, somewhat indicating the models complexity. All weak classifiers are represented as single depth binary decision trees on top of the selected LBP features.

4.2 An Aggregate Channel Features based Approach

In comparison to our first approach, the algorithm of (Dollár et al., 2010) proposes to add multiple invariant feature representations to the adaptive boosting process, as aggregate feature channels. By adding colour, gradient filters, Gabor wavelets, . . . the accuracy of the trained detectors increases compared to using a single feature channel.

We first of all train a model using a similar amount of positive training data *(2000 samples)* to the best performing model of the previous technique. However we notice that the negative data might be gathered from patches that are also validated afterwards, since the single top left corner did not contain enough background patches to use in the Viola&Jones based approach. Therefore, two extra ACF-based models are trained, splitting the dataset into a lower *(1.741 training samples)* and an upper *(1.914 training samples)* image half. We train a model using one half and validated the model using the other half of the image. In general the ACF algorithm uses a lot more negative training samples gathered from the same image as the positive training, leading up to 150.000 patches.

5 APPROACHES WITH DEEP LEARNING

After training our boosted cascade models, we switch to the deep learned models. We first try learning a complete architecture from scratch without initialized weights, then apply several transfer learning approaches, where existing weights of a pre-trained deep model are fine-tuned towards application-specific weights, resulting in a model that can detect the new object class. Finally we investigate the difference between classification and detection architectures in deep learning and their applicability on coconut tree detection in aerial imagery.

5.1 Learning a Complete New Deep Learned Architecture

Although literature advises not to do this, we train a completely new model on the available case-specific data, where no initialization of the weights, based on a previous training, is done. This model seems to converge, looking at the loss-rate over the number of training iterations, but in subsection **??** we discuss why this converged model is misleading and usable in our application.

5.2 Freezing (n-1) Layers and Fine-tuning Final Layer Weights

A second approach is to freeze the weights of the pre-trained convolutional layers, and only re-train the final layer and its connections. This forces the deep learned model to make new constellations of existing features for a new object class. We apply this approach on the existing *InceptionV3* model inside TensorFlow and try to fine-tune the final layer to be able to classify coconut trees in aerial image patches, while freezing all the other convolutional layers.

One major advantage of this approach is that the amount of data needed for this kind of transfer-learning is very small. Sample cases in the TensorFlow framework prove that only 75 samples per class can already be enough for obtaining satisfying results.

This approach only works if the object class to be detected is somehow related to the data contained in the initial dataset on which the model was trained. If the data is however drastically different, like in the case of aerial imagery, then obtaining satisfying results using this approach is quite hard, as illustrated in subsection 6 and other approaches should be considered.

5.3 Fine-tuning Weights of All Layers

Instead of freezing the weights of all the pre-trained layers, we can also tolerate slight adaptations of the pre-trained weights of the convolutional layers. This allows to change the learned features to be more specific to our desired detection task and then learn a constellation of those new fine-tuned features on top of that. When doing so, setting a small learning rate is mandatory, else the initial weights will be changed too drastically too fast, prohibiting the model to converge on an optimal solution.

Using Darknet, we apply transfer learning using this fine-tuning approach on both the *Darknet19* and the *Densenet201* architecture, trained on ImageNet, with the goal of obtaining a deep learned classifier for our new case of coconut tree localisation and classification, using a smaller set of case-specific manually annotated image patches.

5.4 From Classification towards Detection Architectures

Since we are aware that using a classification network implies that we need to provide a multi-scale sliding-window based approach for gathering image patches, we try training a single pass detection based model *(YOLOv2 architecture)*. Literature clearly states that these single pass networks are much faster than their sliding-window counterparts.

Unfortunately due to the coarse grid-based region proposals, the proposed architecture is not able to cope with dense object-packed scenes, where object instances are closely together and slightly overlapping. This triggers final output detections that cover multiple object instances, instead of retrieving single object instances and furthermore doesn't allow the model to converge to an optimal configuration. This is a major problem in our application of coconut tree detection in aerial imagery and thus this approach was abandoned.

6 RESULTS

This section discusses the various results we obtain with the different object detection approaches focusing on our case of robust and accurate coconut tree detection.

6.1 Viola&Jones-based Object Detection

Figure 3 displays the obtained precision-recall curves for the Viola&Jones boosted cascades of weak classifiers using local binary pattern features. For each detector we also report the average precision (AP), which is calculated as the area-under-the-curve for the given precision-recall curve.

The closer the precision-recall curve lies to the top-right corner, the better the detector. Increasing the number of negative samples, which gives the model a better descriptive power for its negative class, seems to work well. This should in principle also mean a higher average precision, but we reckon our graph does not directly represent this. Our OpenCV based implementation does not allow to generate more precision-recall points for the given data, and since we do not want to extrapolate the unknown data points, we do not take the area under this non-existing part into account. This gives a wrong impression on the achieved average precision.

We conclude that given a fairly limited set of case-specific annotated training data, and a limited training time of only two hours, we obtain a detector that is able to detect coconut trees with 90% precision at a recall of 80%. Furthermore we notice that applying data augmentation helps boosting the generalization properties of our boosted cascade models. The final model performs detections on a 10.000×10.000 pixel image within ten minutes.

6.2 ACF-based Object Detection

Figure 4 shows the obtained precision-recall curves for the ACF boosted cascades. We immediately notice that this framework is able to draw power from multiple feature channels and is thus able to obtain higher average precisions. Our best scoring model, trained on the bottom half of our dataset image and validated on the top part, achieves an average precision of 94.55%.

The optimal point of the best model, and thus the optimal setting of our detector, achieves 96% precision at a recall of 90% which is quite amazing given the very limited training time of only 30 minutes. The best performing model performs detections

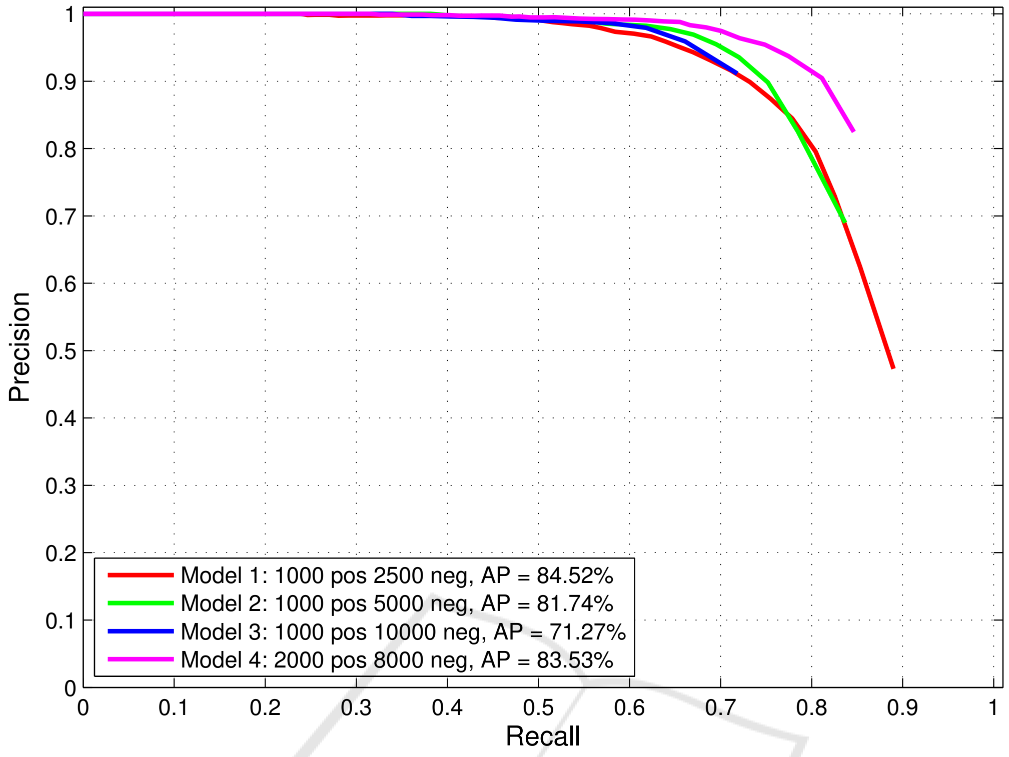

Figure 3: Precision-recall curves for the different Viola&Jones-based detection models accompanied by the number of training samples used and the achieved average precision.

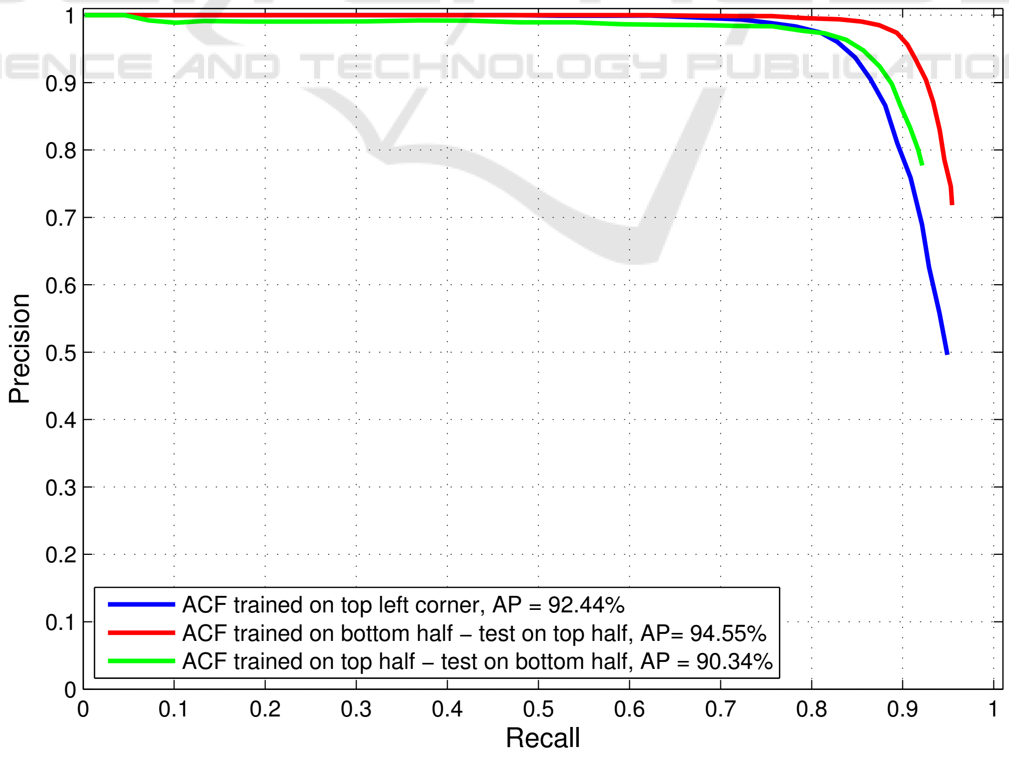

Figure 4: Precision-recall curves for the different ACF-based detection models with their achieved average precision.

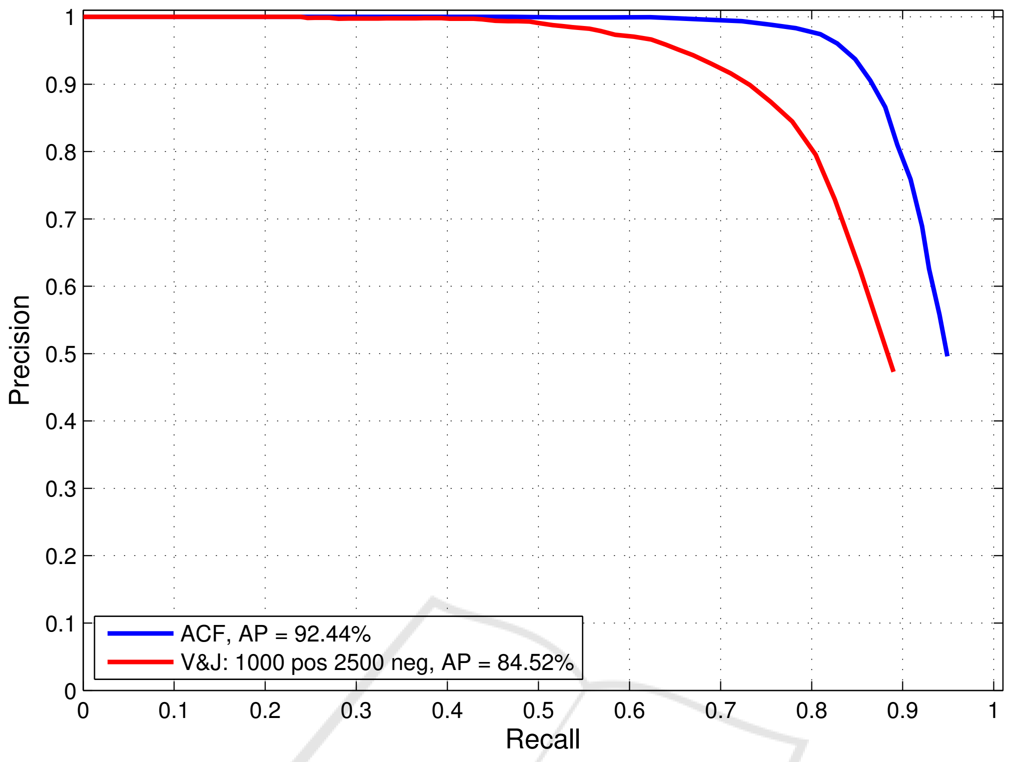

Figure 5: Precision-recall comparison between VJ and ACF models on the same validation dataset.

on the 10.000 × 10.000 pixel image within five minutes.

6.3 Viola&Jones versus ACF

Since both detectors of subsection 6.1 and 6.2 are validated on different amounts of data, we decide to perform an additional comparative study.

Figure 5 shows the result of validating the best performing Viola&Jones and ACF detector trained on the top left quarter and then validated on the same remaining image as validation. This clearly shows that ACF outperforms Viola&Jones with a 7.5% higher average precision.

6.4 Deep Learned Object Classification

Our initial attempt at training a complete deep learning classification model from our limited set of training data, did not produce usable results, although the model seems to converge. With a top1-accuracy, a validation metric used in large scale classification benchmarks, for classification of only 33% given a two-class problem *(coconut tree or background)*, this trained model performs worse than random guessing on the class label, which given a large enough dataset, should eventually converge to 50% top1-accuracy.

6.4.1 Transfer Learning with Frozen Layers

The transfer learning using TensorFlow is done with only 75 coconut tree samples and 75 background samples, randomly sampled from the dataset, because retraining the final convolutional layer is computationally less demanding. All other layers are frozen in this set-up, meaning their weights cannot be changed. The remaining image content is used for validation and compared to the ground truth annotations. The trained model achieves a 77% top1-accuracy.

To be able to compare this accuracy to the accuracy of our previously trained boosted cascades we calculate the precision and recall at pixel level. This results in a precision of 72% at a recall of 52%. Compared to the results obtained with our boosted cascades we decide that his approach does not yield satisfying results, and thus this approach was abandoned.

6.4.2 Transfer Learning by Fine-tuning All Layers

Following the frozen-layer-model approach we suggest using pre-trained weights as initialization for model fine-tuning. However, instead of freezing the weights of all but the last convolutional layers, we al-

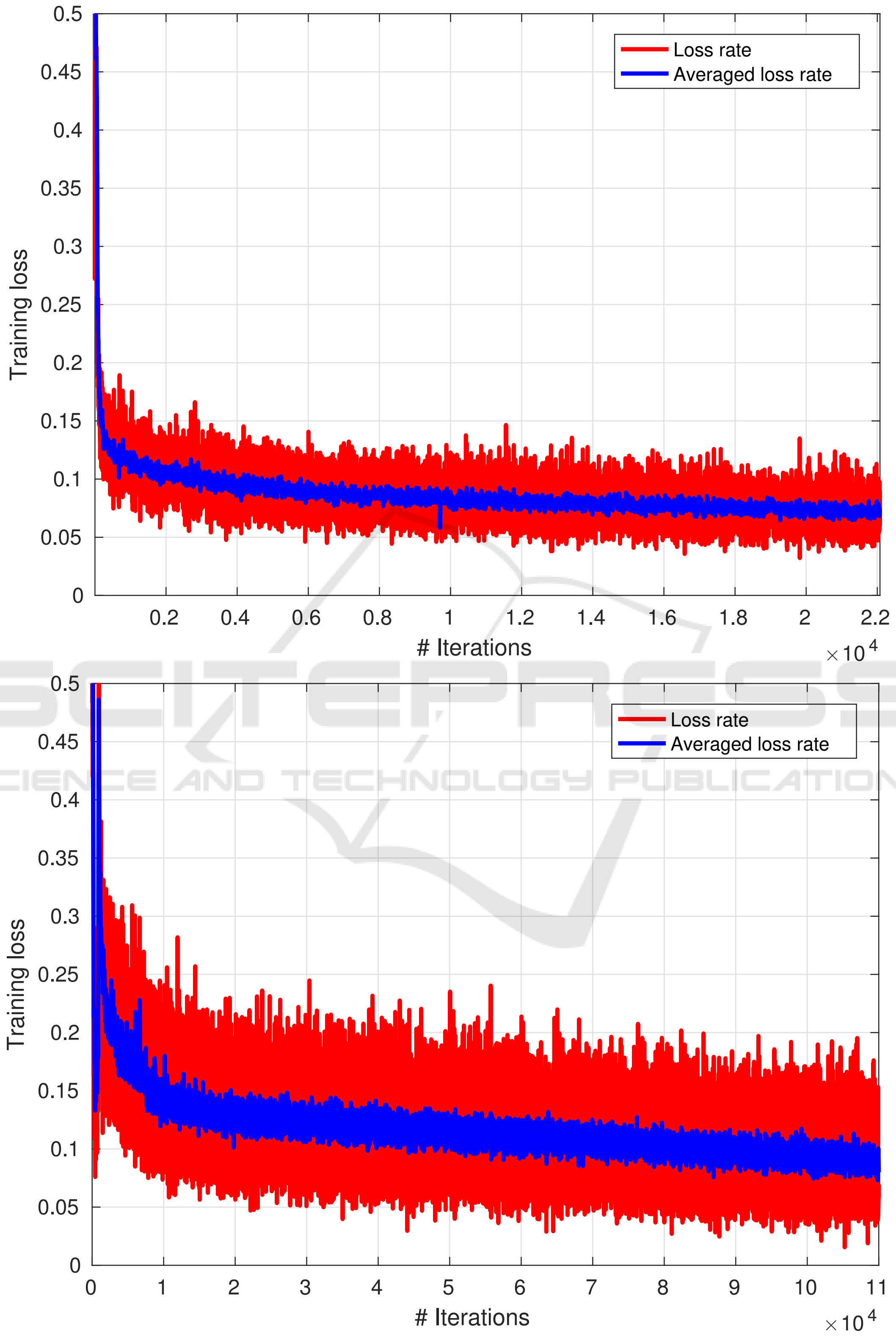

Figure 6: Loss-rate and average loss-rate curves during training for both *(top)* Darknet19 and *(bottom)* Densenet201 models.

low the complete network to fine-tune its weights.

We started with the default *Darknet19* network, existing of nineteen convolutional layers and then tried a similar approach with the more complex *Densenet201* network, containing two-hundred-and-one convolutional layers. The reason of testing both architectures is the fact that the author of the Darknet framework illustrated that using an even deeper network achieves higher top1-accuracy while being slower at inference time (Redmon, 2013). We decided to verify if this behaviour was reproducible using our coconut tree dataset.

Figure 6 displays the loss rate versus the number of training iterations for both bodels. As seen both models seem to be able to converge to a stable model given enough iterations. In order to avoid overfitting to our training data we evaluated our deep learned classification models at several iteration intervals to determine the best model weights for our coconut tree classification task. The fast drop in loss rate is explained by the fact that we increase the batch size for training these models. This allows to take larger learning rate steps and at once is some sort of safety measurement against outliers. For our fine-tuned *Darknet19* model we find that using 10.000 iterations seems optimal at a top1-accuracy of 95.2%, while for the fine-tuned *Densenet201* model using 20.000 iterations gives us the best performance at a top1-accuracy of 97.4%.

6.4.3 Execution Speeds and Memory Footprints of Deep Learning Approaches

Since we are using the classification architectures of Darknet instead of using detection architectures, we are aware that we need to apply a sliding-window based evaluation on our large input image to perform coconut tree localisation.

We evaluate our models using a single NVIDIA TitanX GPU. The *Darknet19* model evaluates our 100×100 pixel input patches at 265 FPS while our *Densenet201* model evaluates patches at 52 FPS. Table 2 gives an overview of the step size, used for the sliding-window, in relation to the execution time of both of our models. Due to the nature of our images, performing multi-scale analysis is quite useless, since images are captured on a constant flight height. Combined with the fact that deep learning is quite robust to slight size changes, we stick to 100×100 pixels.

Considering a 50 pixel overlap between patches in both x and y directions and thus at a 50 pixel step size, the complete image can be evaluated in only two and a half minutes using our *Darknet19* model. While this does increase when a smaller step size is selected, one can argue if this smaller step size is actually needed, since there is already a 50% overlap of patches in both dimensions. Given the fact that there are several more optimization possibilities *(e.g. using multiple GPUs)* makes us believe that we can achieve even faster processing speeds. If we compare this to our boosted cascade based approaches, as shown in Table 3, our Viola&Jones model takes 10 minutes for a 10.000×10.000 pixel image, while the ACF model takes 5 minutes for the same resolution. Given the high top-1 accuracy obtained with the deep learned models, one could definitely consider switching to these more advanced algorithms.

Finally, taking a look at the memory footprint of our deep learning models might be interesting for future research. For training on our NVIDIA TitanX instance, we made the batch sizes as large as possible to fill the complete 12GB of dedicated memory. However, at run time, we process image patch per image patch and thus the footprint is only 400MB for both models, which means the model can also be run on a low-end GPU, albeit slower.

6.5 Visual Results

Precision-recall curves or top1-accuracy results give a quantitative evaluation of the trained models, but for customers, it is always interesting to see visual results of the trained models. Therefore we developed a visualisation tool that allows to visualise the output detections of any given model with a specific colour code as an overlay on top of the original input image, as seen in Figure 7. Here we see the output of our VJ and ACF boosted cascade algorithms and for our deep learning classification output. For visualisation purposes we need to select a fixed point on the precision-recall curve. This threshold is set at a precision of 90.46% and a recall of 81.12% for the Viola&Jones model, a precision of 90.55% and a recall of 86.43% for the ACF model and a precision of 97.31% and a recall of 88.58% for the deep learning approach. Green patches are true positive detections *(patches classified as coconut tree by the model and actually containing a coconut tree)*, red patches are false positives *(patches classified as coconut tree by the model but not containing a coconut tree)* and purple patches are false negatives *(patches classified as*

Table 2: Execution speeds for a full 10.000×10.000 pixel image for both deep learning models (*Darknet19* and *Densenet201*), at different step sizes.

step	#patches	Darknet19	Densenet201
5px	3.924.361	4h	20h30m
25px	157.609	9m5s	50m20s
50px	39.601	2m30s	12m35s

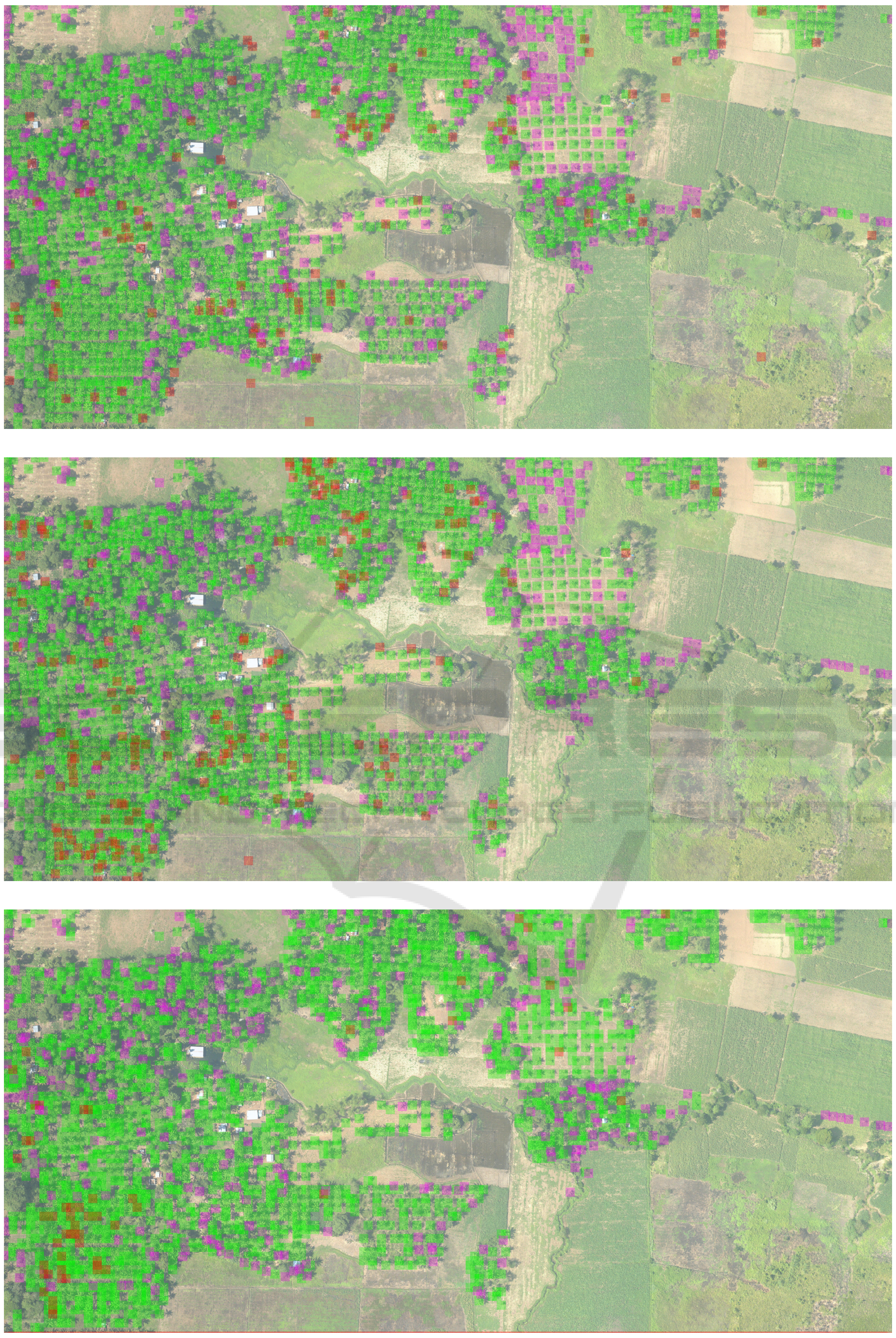

Figure 7: Visual results for the **(top)** VJ boosted cascade model *[P=90.46%,R=81.12%]*, the **(middle)** ACF boosted cascade model *[P=90.55%,R=86.43%]* and the **(bottom)** deep learned Darknet19 model *[P=97.31%,R=88.58%]* showing: *(green)* TP *(red)* FP *(purple)* FN.

Table 3: Configurations for the visual output, including precision, recall, training and inference time (for a 10.000 × 10.000 pixel image).

Model	Precision	Recall	Train	Infer
V&J	90.64%	81.12%	2h	10m
ACF	90.55%	86.43%	30m	5m
DN19	97.31%	88.58%	24h	2m30s

background by the model but actually containing a coconut tree).

Comparing the different output images, we clearly see some expected behaviour. The VJ model suffers from a higher false positive rate than the ACF model. This can be explained by the fact that VJ does not take into account colour information and thus triggers several detections on coconut tree shadows, whereas ACF is more robust to this. Comparing the ACF model to the Darknet19 model, we see that the Darknet19 model has almost no false positive detections, hence the high precision at a high recall rate. However the approach still suffers from false negative detections. We are convinced that this is partly due to the step size of 50 pixels, used for this evaluation. Decreasing the step size towards 25 or even 10 pixels, should further reduce the number of false negative detections.

7 CONCLUSIONS

With this research we have proven both the capabilities of boosted cascade as well as deep learned detection models for coconut tree localisation in aerial images. Our best boosted cascade performs at an average precision of 94.56% while our best deep learning model achieves a top1-accuracy of 97.4%. Although our deep learning pipeline evaluates two times as fast, we reckon that boosted cascades are still in the race, especially given the lower computational complexity demands, but the high classification accuracy and speed of deep learning can simply not be ignored.

As future work we suggest taking a look at region proposal networks, to combine with our classification deep learning networks. This would reduce the amount of image patches drastically and make the complete pipeline even faster. On top of that we also notice that more recent research, focusses on combining the best of both worlds, as described in (Ouyang et al., 2017; Zhang et al., 2017). While using the principle of a boosted cascade, to benefit from the early rejection principle, the weak classifiers are built using convolutional neural network architectures, which guarantees a higher average precision in the long end.

ACKNOWLEDGEMENTS

This work is supported by the KU Leuven, Campus De Nayer and the Flanders Innovation & Entrepreneurship (AIO).

REFERENCES

Abadi, M., Agarwal, A., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face recognition with local binary patterns. *Proceedings of the European Conference on Computer Vision*, pages 469–481.

Bradski, G. and Kaehler, A. (2000). The opencv library. *Doctor Dobbs Journal*, 25(11):120–126.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dollár, P. (2005). Piotr's Computer Vision Matlab Toolbox (PMT). https://github.com/pdollar/toolbox.

Dollár, P., Belongie, S., J, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *Proceedings of the British Machine Vision Conference*, volume 2.

Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 5–12.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., and Keutzer, K. (2014). Densenet: Implementing efficient convnet descriptor pyramids. *arXiv*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the International Conference on Multimedia*, pages 675–678. ACM.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer.

Margineantu, D. D. and Dietterich, T. G. (1997). Pruning adaptive boosting. In *Proceedings of the International*

Conference on Machine Learning, volume 97, pages 211–218.

Ouyang, W., Wang, K., Zhu, X., and Wang, X. (2017). Chained cascade network for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*.

Puttemans, S., Van Ranst, W., and Goedemé, T. (2016a). Detection of photovoltaic installations in RGB aerial imaging: a comparative study. In *Proceedings of GEOBIA*.

Puttemans, S., Vanbrabant, Y., et al. (2016b). Automated visual fruit detection for harvest estimation and robotic harvesting. In *Proceedings of the International Conference on Image Processing Theory, Tools and Applications*.

Redmon, J. (2013). Darknet: Open source neural networks in c. http://pjreddie.com/darknet/.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Shaikh, F., Sharma, A., Gupta, P., and Khan, D. (2016). A driver drowsiness detection system using cascaded AdaBoost. *Imperial Journal of Interdisciplinary Research*, 2(5).

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Targ, S., Almeida, D., and Lyman, K. (2016). Resnet in resnet: generalizing residual architectures. *arXiv preprint arXiv:1603.08029*.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511.

Zhang, K., Zhang, Z., Wang, H., Li, Z., Qiao, Y., and Liu, W. (2017). Detecting faces using inside cascaded contextual cnn. In *Proceedings of the IEEE International Conference on Computer Vision*.

Zheng, Y., Merkulovb, A., and Bandarib, M. (2016). Early breast cancer detection with digital mammograms using haar-like features and AdaBoost algorithm. In *Scientific Sensing and Imaging*.