GAN-BASED SYNTHESIS OF DEEP LEARNING TRAINING DATA FOR UAV MONITORING

D. Langenkämper[1], R. van Kevelaer[1], T. Möller[1], T. W. Nattkemper[1]*

[1] Biodata Mining Group, Technical Faculty, Bielefeld University, 33615 Bielefeld, Germany - (daniel.langenkaemper,robin.van_kevelaer,torben.moeller,tim.nattkemper)@uni-bielefeld.de

KEY WORDS: deep learning, computer vision, inspection, Generative Adversarial Networks, low-shot learning

ABSTRACT:

Wind energy is a critical part of overcoming the use of fossil or nuclear energy usage. The price pressure on the renewable industry sector demands to cut the costs for costly regular inspections carried out by industrial climbers. Drone-based video-inspection reduces costs as well as increases the safety of inspection personal. To further increase the throughput, automatic or semi-automatic solutions to analyze these videos are needed. However, modern machine learning architectures need a lot of data to work reliably. This is by design a problem, as structural damage is rather rare in industrial infrastructure. Our proposed approach uses Generative Adversarial Networks to generate synthetic unmanned aerial vehicle imagery. This allows us to create a large enough training dataset ($> 10^3$) from a dataset, which is at least an order of magnitude smaller (approx. 10^2). We show that we can increase the classification accuracy of up to 6 percentage points.

1. INTRODUCTION

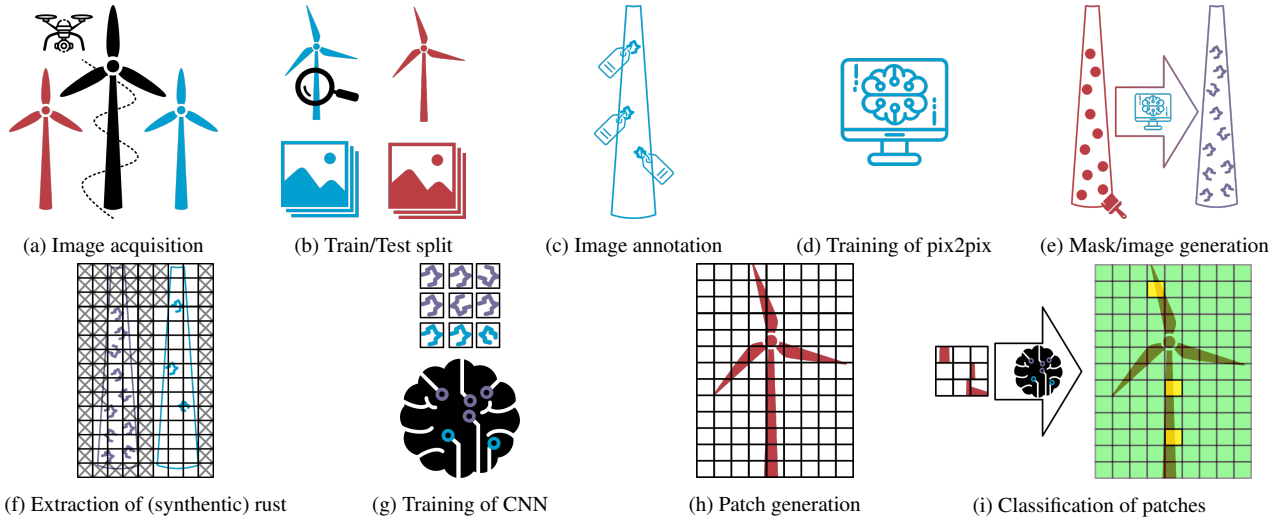
Onshore and especially offshore wind energy farms are crucial to overcome the use of fossil fuels or nuclear energy to generate electricity. Due to high price pressure on the energy market, ongoing costs have to be reduced. Regular inspection carried out by professional industrial climbers is essential to prevent structural damage and to assure optimal performance of the turbines. New technological approaches are needed to lower the costs of this labor-intensive and thus costly method. This applies to a wide range of other industries running hard-to-reach infrastructure as well. Our research project is using unmanned aerial vehicle-based video inspection to avoid highly trained and thus costly climbers. Furthermore, unmanned aerial vehicle (UAV) inspection improves the security of the inspection task as an error will not be fatal. UAV-based inspection is already used for a multitude of scenarios, e.g. inspection of bridges (Metni, Hamel, 2007, Hallermann, Morgenthal, 2014), industrial facilities (Nikolic et al., 2013), power lines (Jones, 2005, Deng et al., 2014), poles (Sa et al., 2015), buildings (Phung et al., 2017) and power facilities (Jordan et al., 2017). For a more in depth overview on possibilties and limitations of UAV-based inspection have a look at (Morgenthal, Hallermann, 2014) and for an extensive review of applications of UAV inspection at (Jordan et al., 2017). Although the process of acquiring the images or video can be now done by UAV operators, experts are still needed to review the UAV videos, especially concerning that the large majority ($> 99\%$) of the frames contain no damage at all. Thus semi- or fully automatic inspection is required in order to cope with the huge amounts of video data. A problem by design with machine learning-based structural-damage detection is that damage observations are quite rare, thus there is not much training data available. In computer vision this problem is called one-shot or low-shot learning, depending on the scarcity of data. Object detection, here damage detection, using modern deep learning architectures such as Mask R-CNN (He et al., 2017) or YOLO (Redmon, Farhadi, 2018) are quite advanced and show state of the art detection performance. However, these need a large amount of data samples to learn from (approx. $> 10^3$). Due to the very small number of damage observations in UAV-based inspections, these networks cannot be applied in this domain in a straightforward manner. To tackle this problem, algorithmic solutions, e.g. low-shot transfer detector (LSTD) (Chen et al., 2018) exchanging these classifiers exist. In most cases, an increase in training data and the use of these established state of the art detectors is favored. In addition, objects to learn from should not morphologically be too different from the objects to be detected in these approaches. Yet, damage patterns such as rust can morphologically be quite diverse, as it can form arbitrary shapes, thus a new solution is needed. Here we propose an alternative solution using Generative Adversarial Networks(GANs), based on the pix2pix architecture (Isola et al., 2017), to generate synthetic samples. The pix2pix network transfers the style of one image to that of another one. In their paper Isola et al. show the transfer from segmentation masks to a street scene or a facade image, from aerial imagery to a map, from a photo taken at day to one taken at night, from an edge image to a photo and they use it to colorize images. The basis for the generation of our synthetic samples is a segmentation mask of the image to be generated. The mask can either be created by altering the segmentation mask of an existing image, or by manually creating a new one. Altering means to introduce or remove features, such as rust, contaminations or oil spills, to be added or removed in the synthetic image. This process allows us to introduce damage into any image of an intact structure, which can be used as training data afterward. In addition, we can remove unwanted features like contaminations on the lens.

2. METHOD

For an overview of the methodology used please have a look at Figure 1.

a) Image acquisition We trained and evaluated our approach on 310 images shot by a UAV, each showing a part of a wind turbine. The wind turbine is an on-shore model located in Bremerhaven on the German North-Sea coast.

* Corresponding author



(a) Image acquisition (b) Train/Test split (c) Image annotation (d) Training of pix2pix (e) Mask/image generation

(f) Extraction of (synthetic) rust (g) Training of CNN (h) Patch generation (i) Classification of patches

Figure 1. Experiment overview: We show the different steps of our experimental setup. The upper row a)-e) shows the generation of synthetic data samples using GANs. The lower row f)-i) shows the patch classification experiment used for validation of our method. More information concernig each step is provided in the text (cmp. section 2). Training data and derived data and models are colored blue, test data are colored red and synthetic data are colored purple.

b) Train/Test split The images were split into 290 training images and 20 test images to avoid data leakage.

c) Image annotation The annotations were marked by hand, i.e. labeled polygon annotations marking the most important features, such as tower, oil, rust, or contamination on the lens. Of the 290 training images, 145 showed rust of different sizes and qualities. The annotations were automatically analyzed and the largest connected annotations were selected as area of interest, i.e. the wind turbine, whereas the rest was considered background. A segmentation mask for the labeled polygon annotations was created (without background).

d) Training of pix2pix With the segmentation masks a network based on the pix2pix architecture (Isola et al., 2017) was trained. This network learns to transfer from one representation of a scene to another. In our case, this is from the segmentation mask of an image to the actual color image. To our knowledge, we are the first to propose the use of pix2pix in the context of inspection and monitoring. The pix2pix network was trained for 7500 epochs.

e) Mask/image generation Segmentation mask images of the test set were altered in a way that the masks for contaminations on the lens were removed and additional masks for rust were introduced by hand. Furthermore, for each segmentation mask 6 copies were created using shifts of [-300, -150, -50, 50, 150, 300] on the image is x direction and an additional 6 copies by random cropping and resizing to the original size, i.e. zooming in on the image. The pix2pix network can now be applied to the masks to generate synthetic images of wind turbines. As the background is not important to create training data and it is quite hard to generate a realistic-looking environment, we omitted the generation of it and replaced it by an excerpt of an image showing background. The resulting images are 512×768 px in size.

f) Extraction of (synthetic) rust For training a classifier, we need training data. Thus a grid is overlaid over the synthetic images and all patches of size 64×64 containing rust are extracted. For the wind turbine photographs the same grid is used, however, this time all patches are extracted, but divided into containing either rust or background.

g) Training of CNN A convolutional neural network (CNN) is trained with the said patches to classify patches to either contain rust or belong to the background. For more details on the CNN have a look at the text below. We trained two classifiers with either the GAN generated rust patches added or without them. In the case without the GAN samples we trained the classifier with 7344 background and 498 rust patches. When adding the GAN generated rust samples there are 2874 rust patches and still 7344 background samples.

h) Patch generation The test images are annotated completely, thus marking all the rust spots. All images of the test set are cut into patches using the grid approach described above.

i) Classification of patches These patches are classified using the trained CNN. Using the annotations we can evaluate the images using precision, recall and (macro) F_1-score. There were 140 test patches containing rust and 5236 test patches containing background.

2.1 CNN

We tested different recent classifier architectures with this problem. These are Alexnet (Krizhevsky, 2014), VGG (Simonyan, Zisserman, 2014), Squeezenet (Iandola et al., 2016), Inception V3 (Szegedy et al., 2016), GoogLeNet (Szegedy et al., 2015), Shufflenet (Ma et al., 2018), Resnet (He et al., 2016), Resnext (Xie et al., 2017), Densenet (Huang et al., 2017), Mobilenet v2 (Sandler et al., 2018). If not stated otherwise the networks were trained from scratch, however, some networks performed better using pretraining with ImageNet data. The number of epochs was 64 and a batch size of 32 was used. PyTorch (Paszke et al., 2019) was used for the patch classification experiment. For the pix2pix network the Tensorflow implementation was used (Abadi et al., 2016).

2.2 Evaluation metrics

For evaluation precision $\mathcal{P}$, recall $\mathcal{R}$ and the F_1-score F_1 are computed.

$$\mathcal{P} = \frac{TP}{TP + FP} \tag{1}$$

$$\mathcal{R} = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 * \frac{\mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{3}$$

where TP are the true positives, e.g. the samples containing rust and being classified as rust, FP are the false positives, e.g. the samples which are classified to contain rust, but there is no rust on the patch and FN are the false negatives, e.g. the samples which contain rust but are classified to not contain rust. These are per class measures. To yield one value for an experiment, one can compute the measure for each class and average it. Thus yielding the macro scores. The macro scores are invariant to class abundance, thus are quite relevant for our case as patches showing background are much more abundant than patches showing rust. For completeness we also provide the weighted scores, which are more commononly used. Here the class-wise scores are averaged using a weighting scheme proportional to the abundance of classes.

3. RESULTS

A resulting synthetic image is shown in Figure 2 a) alongside the original image 2 b), the original segmentation mask 2 c) and the altered segmentation mask 2 d). We show that rust can be successfully added, which looks realistic and that the lens contamination was removed. The whole images, along with the segmentation masks, can be used to train an object detection algorithm, such as Mask R-CNN or YOLO. In addition, the rust spots can be extracted to train a convolutional neural network (CNN), such as ResNet (He et al., 2016), to classify rust in images showing wind turbines. In addition, the generation of rust training samples for classification, i.e. images showing only rust, can directly be generated to train classification networks. Although pix2pix works deterministically, thus resulting in one output per segmentation mask, differently looking segmentation masks and thus resulting images can be generated by manipulating the masks with random flips of background pixels to rust pixels at the border of the already existing segmentation masks for rust. In addition, by moving either all segmentations, i.e. moving the tower including the rust annotations or the rust annotations alone, differently looking results can be achieved, thus introducing more training samples. In Table 1 the results of the patch classification experiment are shown. If multiple parametrizations of the algorithm have been tested only the best performing ones, i.e. the ones with the highest macro F_1-score is chosen. The corresponding result (with or without added GAN samples) with the same network parametrization is then presented as well, even though there might be other configurations where the corresponding exeriment performed better. All tested networks, except resnext50_32x4d and squeezenet1_1, provide better results with the GAN generated test samples added. However, the decrase in performance for resnext50_32x4d and squeezenet1_1 is not that big. The biggest increase is that of vgg16_bn. However, for this network as well as for Alexnet the increases are a bit misleading as the version without added GAN samples performed very poorly by every time predicting background, thus achieving a classification performance of 49.34. With added GAN samples vgg16_bn performed rather competitively with the other networks. Apart from these trivial improvements resnet101 with 4.42 percentage points improvements has the biggest gain. The best overall accuracy and the only example above 90% is shufflenet_v2_x0_5, which also features the second biggest improvement. More detailed results are shown in the Appendix.

CNN	F_1	F_1^{GAN}	Δ
shufflenet_v2_x0_5	88.67	91.52	2.85
resnext50_32x4d	89.73	88.88	-0.85
resnet101*	85.16	89.58	4.42
densenet121*	87.99	89.51	1.52
squeezenet1_1*	86.07	85.27	-0.80
googlenet	84.75	86.00	1.25
vgg16_bn*	49.34	84.24	34.9
mobilenet_v2	83.16	83.91	0.75
inception_v3	83.22	83.46	0.24
Alexnet	49.34	57.34	8.00

Table 1. Results of the patch classification experiment sorted by highest score. All F_1-scores are macro averaged (cmp. section 2.2). F_1 are the results using the classifier trained only on the training data. For F_1^{GAN} the GAN generated samples are added to the training set additionally. Δ is the difference $(F_1^{GAN} - F_1)$ between the two. Networks with an * have been pretrained on ImageNet. All others were trained from scratch.

4. CONCLUSION

We have shown that our approach is able to create arbitrarily large training data sets for deep learning-based detection or classification based on few training samples. This approach allows another solution to the low-shot learning problem in inspection based projects. We achieved improvements of up to 4.42 percentage points and could increase the absolute macro F_1-score to over 90% only with our methodology. With 91.52% shufflenet with added GAN samples is the best performing network. Except of two examples adding GAN samples improved the classification performance of all networks (cmp. Table 1). Our approach allows to compensate for the scarcity of image samples of damaged structures. The overall results allow for a semi-automatic inspection process where attention is guided by the classification algorithm. This allows the inspection teams to check huge amounts of data acquired by drones today.

ACKNOWLEDGEMENTS

DL, RK, TM and TWN have received funding by the Federal Ministry for Economic Affairs and Energy (grant number 0324254D).

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Chen, H., Wang, Y., Wang, G., Qiao, Y., 2018. Lstd: A low-shot transfer detector for object detection. *Thirty-Second AAAI Conference on Artificial Intelligence*.

Deng, C., Wang, S., Huang, Z., Tan, Z., Liu, J., 2014. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *J. Commun*, 9(9), 687–692.

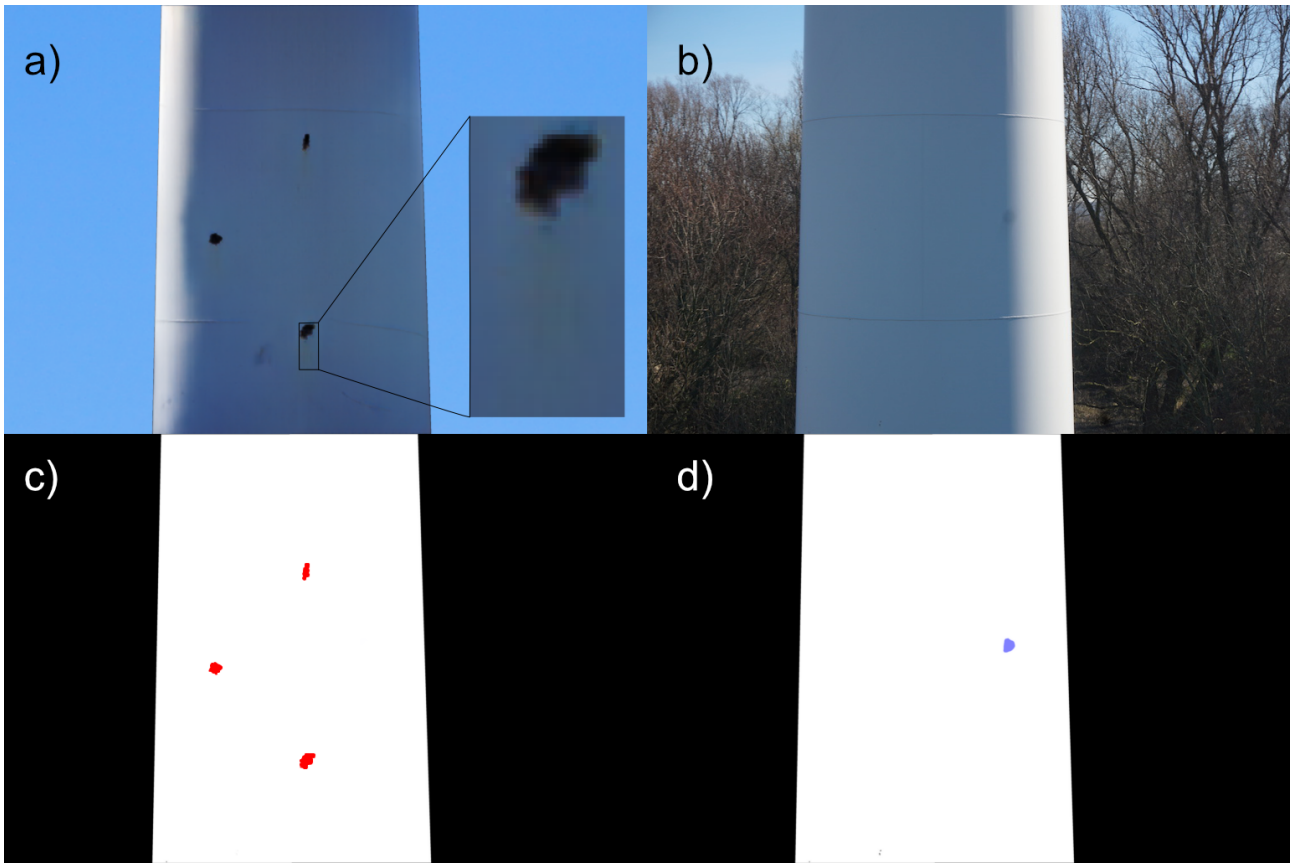


Figure 2. Synthetic image generation from segmentation masks. a) generated image, b) original image from test set, c) segmentation mask for image to be generated (see a), d) segmentation mask for original image (see b)

CNN	Recall	Precision	F_1	Recall^{GAN}	Precision^{GAN}	F_1^{GAN}
shufflenet_v2_x0_5	90.69	86.86	88.67	93.97	89.34	91.52
resnext50_32x4d	90.42	89.06	89.73	91.05	86.95	88.88
resnet101*	90.42	81.18	85.16	91.78	87.61	89.58
densenet121*	92.35	84.49	87.99	90.75	88.35	89.51
squeezenet1_1*	90.84	82.35	86.07	90.43	81.35	85.27
googlenet	91.41	80.04	84.75	91.51	81.85	86.00
vgg16_bn*	50.00	48.70	49.34	88.65	80.78	84.24
mobilenet_v2	87.20	79.96	83.16	88.28	80.49	83.91
inception_v3	89.24	78.90	83.22	91.63	78.13	83.46
alexnet	50.00	48.70	49.34	56.37	58.76	57.34

Table 2. Results of the patch classification experiment sorted by highest F_1-score. All metrics are macro averaged (cmp. section 2.2).

Hallermann, N., Morgenthal, G., 2014. Visual inspection strategies for large bridges using unmanned aerial vehicles (uav). *Proc. of 7th IABMAS, International Conference on Bridge Maintenance, Safety and Management*, 661–667.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q., 2017. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

Isola, P., Zhu, J.-Y., Zhou, T., Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Jones, D., 2005. Power line inspection-a uav concept. *2005 The IEE Forum on Autonomous Systems (Ref. No. 2005/11271)*, IET, 8–pp.

Jordan, S., Moore, J., Hovet, S., Box, J., Perry, J., Kirsche, K., Lewis, D., Tse, Z. T. H., 2017. State-of-the-art technologies for UAV inspections. *IET Radar, Sonar & Navigation*, 12(2), 151–164.

Krizhevsky, A., 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*.

Ma, N., Zhang, X., Zheng, H.-T., Sun, J., 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European Conference on Computer Vision (ECCV)*, 116–131.

Metni, N., Hamel, T., 2007. A UAV for bridge inspection: Visual servoing control law with orientation limits. *Automation in construction*, 17(1), 3–10.

Morgenthal, G., Hallermann, N., 2014. Quality assessment of unmanned aerial vehicle (UAV) based visual inspection of structures. *Advances in Structural Engineering*, 17(3), 289–302.

Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., Siegwart, R., 2013. A uav system for inspection of industrial facilities. *2013 IEEE Aerospace Conference*, IEEE, 1–8.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al., 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 8024–8035.

Phung, M. D., Quach, C. H., Dinh, T. H., Ha, Q., 2017. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Automation in Construction*, 81, 25–33.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Sa, I., Hrabar, S., Corke, P., 2015. Outdoor flight testing of a pole inspection uav incorporating high-speed vision. *Field and Service Robotics*, Springer, 107–121.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

APPENDIX

The appendix contains additional results of the patch classification experiment (see Table 2), which are not directly relevant but might be of interest to the reader.