Defect Detection in Tunnel Images using Random Forests and Deep Learning

Guillaume Décor, Mamadou Dian Bah, Philippe Foucher, Pierre Charbonnier, Fabrice Heitz

Defect Detection in Tunnel Images using Random Forests and Deep Learning

G. Decor[1,2], M. D. Bah[1,3], P. Foucher[1], P. Charbonnier[1], F. Heitz[2]

[1] Cerema, Project-team ENDSUM, Strasbourg, France
[2] ICube, UMR 7357, Université de Strasbourg, CNRS, France
[3] Centre d'étude des tunnels (CETU), Bron, France
email: {guillaume.decor,philippe.foucher,pierre.charbonnier}@cerema.fr, fabrice.heitz@unistra.fr, m-dian.bah@univ-orleans.fr

Keywords: Defect detection, Tunnel inspection, Supervised classification, Deep learning, Random forests.

Abstract

Tunnel maintenance requires a complicated and constraining visual inspection. In order to automate this task, we propose to evaluate and compare three statistical learning algorithms, a random forest and two convolutional networks, dedicated to the detection of defects (e.g. cracks) on tunnel linings. Each model is trained on datasets of our own, consisting of images of concrete walls and masonry walls. We show that these learning-based approaches are competitive with the state of the art on this application domain.

1 Introduction

Maintaining tunnels (road, rail or navigable) in service is essential to ensure the safety of goods and users. This requires a periodic inspection of the tunnel to locate surface defects (e.g. cracks) that can lead to more serious deterioration (e.g. concrete splitting exposing reinforcement) or even endangering the structure. The deterioration phenomena that can be observed on tunnel linings are diverse, but often due to the undesirable presence of water, which is therefore also important to identify. Figures 1 and 3 show some examples of defects (cracks, water ingress, lining spalling) on concrete and masonry tunnel linings.
Tunnel inspections are currently carried out by specialized operators directly on-site, which is constraining. This is time consuming, both for on-site intervention and for subsequent data analysis. In this context, Cerema seeks to ease the visual inspection of tunnels through the development of high-performance image acquisition systems, supplemented by image processing algorithms. From an operational point of view, this aims to reduce the downtime of infrastructures as well as inspection time by the operators.
Automatically detecting defects using images of tunnel linings is a particularly complex task, due to the variability of the textures, shapes, colours and objects of interest to be detected. Statistical learning methods are good candidates for solving this type of problem. In this paper, we propose to evaluate the performance of three learning-based methods for detecting defects on images of concrete or masonry linings (see figure 1). We assess two convolutional networks and one random forest algorithm [1] that are trained and tested on our annotated databases. Some preliminary results using random forests have been published in [2] (in French).
The paper is organized as follows: after an overview of the different pattern recognition algorithms tested for tunnel inspection purposes (section 2), we present the experimental data and algorithms that have been implemented (section 3). The results of our experiments are detailed and discussed in section 4. Finally, we conclude and give some perspectives in the last section (section 5).

Figure 1: Examples of images of the sidewall of a masonry tunnel (left) and the vault of a concrete tunnel (right)

2 Related work

2.1 Traditional algorithms

Most traditional methods for tunnel inspection (i.e. without learning) are dedicated to crack detection, see for example [3]. They use relatively basic approaches (grey level filtering and thresholding, combinations of morphological operators, techniques by successive subdivisions and midpoint adjustment). More advanced methods are based on deformable models and the propagation of minimal paths (for a state-of-the-art of these techniques, see [4]), or on the so-called *Percolation* [5] method which consists of ex-

amining the geometry of the level set propagated from a seed. These methods are restrictive in the sense that they are only designed to detect cracks on concrete linings. Adapting them to detect more varied defects (or to generalize to more diverse linings) would require a significant effort.

2.2 Machine Learning algorithms

Several machine learning algorithms have been experimented in recent years. Most of them focuses on crack detection. Each algorithm presented below has been trained on different databases, which makes it difficult to compare methods.

The authors of [6] suggest the use of a convolutional architecture in which the images are previously enriched with additional, non-convolutional descriptors. From a grayscale image $w \times h \times 1$, they build an image $w \times h \times 6$ by adding features such as HOG [7], contour maps, or local entropy. The method reached an accuracy of 88.6% on a test basis of 10,000 images (90,000 images being used for learning and validation).

In [8], several types of defects (cracks, water leakage, deposit) are detected. The authors use a variant of the ResNet [9] architecture on patches extracted from the images and detect the defects using a sliding-window paradigm. The method reaches an accuracy of 87.5% on a basis of 603 images, of which 20% are dedicated to testing.

For sewer pipes inspection, a Faster R-CNN architecture [10] has also been tested in [11] to detect anomalies using a fully convolutional way. 12000 images (training: 7500, validation: 2500, test: 2000) have been annotated in different categories (e.g. cracks and roots) and an accuracy of 86.2% is obtained.

These articles show that learning algorithms (and, more specifically, convolutional networks algorithms) perform well for detecting anomalies in engineering structures. We also notice a huge variability of the data in the learning datasets from one model to another, which further supports the thesis that learning methods are adapted to our problem.

3 Methodology

3.1 Data

To create datasets, Cerema has made its own acquisitions on-site. Tunnel images were recorded using a system composed of 6 cameras mounted on a vehicle (see figure 2). Each image is in 1080p (i.e. 1920×1080 pixels) and covers about $5m^2$ of lining (vault and sidewalls). Two successive images have an overlap of about 95% in canal tunnels and 80% in road tunnels.

Two datasets, from a concrete road tunnel and a navigable masonry tunnel, were used in our experiments. Given the differences in the nature of the defects sought, they were processed separately. More precisely, we are interested here in four types of defects for concrete linings (cracks, water ingresses, exposed irons, pebble nests) while for masonry linings, we only consider water ingresses (see figure 3). In order to address the problem of rare anomalies, these defects are classified in a binary form (healthy/defects).

Figure 2: Imaging system mounted on a car (left) or a boat (right) for image acquisition in road or canal tunnels

		Learning	Test	Total
Concrete	Healthy	594	77	671
	Defect	600	201	801
	Total	1194	278	**1472**
Masonry	Healthy	508	302	810
	Defect	508	268	776
	Total	1016	570	**1586**

Table 1: Composition (number of examples) of the two datasets

3.2 Preprocessing

In order to precisely locate the defects, we extract subimages from the original images. The images are divided into patches of size 101×101 pixels for concrete surfaces and 251×251 for masonry ones. We use a larger format in masonry tunnels to cover several rubble stones in the same sample. Table 1 shows some details about the two datasets.

Labeling is carried out by annotating each sample manually as shown on figure 3. Note that a sample is labelled as "defect" even if a small part of a defect is located in the sample. We observe a variability on the aspect on both categories (intra- and inter-classes interlacing). Moreover, it may be noticed that the labeling operation is not an easy task, even for a human operator. Indeed, it may be quite difficult to determine the label of an area without any information of its surroundings.

3.3 Classification

3.3.1 Random Forest

The random forest algorithm [1] is a meta-classifier, consisting of a set of decision trees. In this *ensemble learning* approach, each tree classifies a sample, represented by its feature vector, into a category. The final classification of the sample by the forest is obtained from the majority

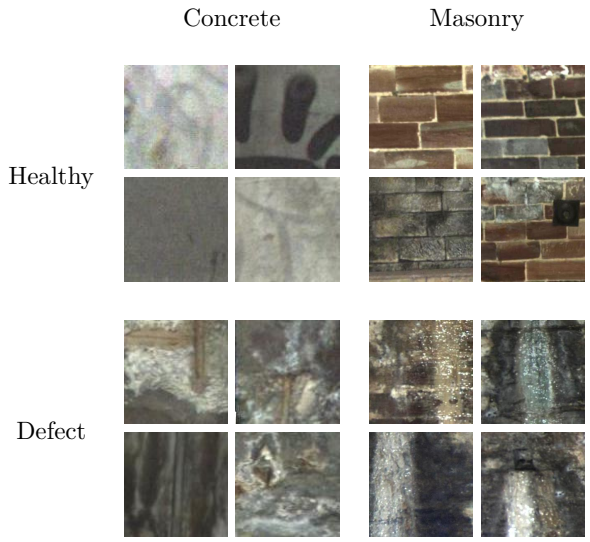


Figure 3: Samples from healthy and defect classes (concrete and masonry linings), illustrating the intra-class variability

vote of all trees. To ensure that the votes of each tree are independent, a different set of data and features is presented to each tree. These subsets of data and features are drawn randomly with replacement, from the learning samples and from the set of features, respectively. In our application, the number of trees in the forest has been empirically set at 200 and the minimum number of elements in each leaf of the tree is set at 1. The selected features should best represent the characteristics observed on the samples of each class. In the case of tunnel images, three types of features were used. The Haralick texture features [12], calculated in 4 orientations of the co-occurrence matrix, provide texture information. Haralick features were adapted to account for colour information. Histograms of Oriented Gradients (HOG) [7] highlight the distribution of contour orientations in the gray level image. Finally, the binary relationships between a pixel and its 8 neighbors are extracted using the Census [13] transform. The statistical distribution of Census values is then calculated to form the mCentrist descriptor [14], in which color information is taken into account. In the end, all features are gathered into a single vector of dimension 3161 (95 values for Haralick features, 18 components for HOG, and 3048 values in the mCentrist descriptor).

3.3.2 Convolutional Networks

Two families of architectures were used. The first one, composed of rather shallow architectures, is based on the LeNet model [15] which was chosen for its simplicity and its speed of learning. The second one is an adaption of the deeper network VGG19 [16], which represents a good compromise between the expressiveness of the model and the computing power needed for training. In both cases, the number of neurons in the multilayer perceptron was adjusted according to the size of the images and the number of classes, in order to maintain the same level of expressiveness. In addition, the LeNet-based network for masonry has been enriched with several convolution layers to better reflect the complexity of this type of lining, compared to concrete whose surface is generally smoother. These architectures are detailed in figure 4. Note that they were not pretrained, but learned end-to-end, *from scratch*.

Input: $(101, 101, 3)$
conv5-6
avgpool-2
conv5-16
avgpool-2
conv1-120
fc-84
dropout (0.5)
fc-2
Softmax

(a) LeNet-based Network (concrete)

Input: $(251, 251, 3)$
conv5-8
avgpool-2
conv5-16
avgpool-2
conv5-32
avgpool-2
conv5-64
avgpool-2
fc-512
dropout (0.5)
fc-512
dropout (0.5)
fc-2
Softmax

(b) LeNet-based Network (masonry)

Input: $(101, 101, 3)$ or $(251, 251, 3)$
Conv. layers identical to VGG19 [16]
fc-32
fc-32
fc-2
Softmax

(c) VGG-based Networks (Concrete and masonry)

conv$k$**-**m	convolutional layer with m filters and a $k \times k$ kernel
avgpool-k	average pooling with a $k \times k$ kernel
fc-n	fully-connected layer with n neurons
dropout (p)	dropout with a probability of p

Figure 4: Convolutional architectures used for the detection of defects (the convention used for the notations is taken from [16])

Prior to the learning phase, the image samples are normalized by mapping each pixel linearly into the cube $[-1; 1]^3$. To this end, the function $x \mapsto \frac{2}{255}x - 1$ is applied to each component.
Stochastic gradient descent is used for training. The learning step l_i at time $i \in \mathbb{N}^*$ is set to:

$$l_i = \frac{l_0}{1 + d \times i}$$

where the parameters l_0 and d were empirically set to 0.01

Model	LeNet-based Network	VGG-based Network
Concrete	181	250
Masonry	250	912

Table 2: Number of epochs

and 0.3 for concrete and to 0.01 and 1 for masonry.
The number of epochs in the learning algorithm is fixed according to the different architecture (see table 2). Initially set at 250, this number was manually adjusted upwards or downwards depending on the learning state. For example, the LeNet-based network for concrete tunnels converged after 181 epochs whereas the VGG-based network for masonry linings did not finish converging at epoch 912, when we stopped it.

3.4 Detection algorithm

The classifiers were trained and tested on the collected datasets (table 1). In the detection step, the classification algorithms are assessed on the original full-frame images (these test images are not present in the training databases). A sliding window algorithm (at the size of the learned sub-image samples) is applied, starting from the top left corner of the full-frame image. Some pixels at the bottom and at the right side on the image are not processed, since there is no overlapping between windows. The result is a block-wise binary segmentation of the full-frame images, that highlights the defects.

4 Results

4.1 Classification

The performances of the different classification algorithms are measured using accuracy and precision/recall metrics on the test sets. Accuracy refers to the true classification rate. Formally, if M is the confusion matrix associated with the model on the test set, we have the following relationship:

$$\text{accuracy} = \frac{\text{Tr}(M)}{\|M\|_1}$$

where $\text{Tr}(M)$ denotes the trace of the matrix M and $\|M\|_1 = \sum_{i,j}|M_{i,j}|$.
For our application domain, we seek to maximize accuracy by focusing on recall over precision. Thus, we prefer having some false alarms than to omit anomalies.
The results are presented in figure 5. It can be seen that, on concrete linings, convolutional networks have higher accuracy than random forests in detecting anomalies, regardless of the depth of the architecture. We also observe that precision/recall values are significantly higher for neural networks than random forest algorithm. In addition to that, these values are better balanced for both neural models.

On masonry surfaces, LeNet-based network and Random Forest alogrithm provide quite similar accuracies (with a slight advantage to the first model). The VGG-based network has more difficulties to learn the discriminating features. Its accuracy is noticeably lower compared to the other two models. As mentioned in section 3.3.2, the training of this model has been stopped before convergence. As for the precision and recall, it can be noted that LeNet-based network is barely better than the random forest algorithm. The same statement does not hold for the VGG-based model, for which results are significantly worse for these two parameters. This is clearly shown through the confusion matrix (cf. figure 5f) where 71 samples were wrongly classified as healthy samples.

$$\begin{array}{c} \\ \text{H (g.t.)} \\ \text{D (g.t.)} \end{array} \begin{array}{c} \begin{array}{cc} \text{H (d.)} & \text{D (d.)} \end{array} \\ \begin{pmatrix} 71 & 6 \\ 17 & 184 \end{pmatrix} \end{array}$$

(a) Random Forest (concrete)

$$\begin{array}{c} \\ \text{H} \\ \text{D} \end{array} \begin{array}{c} \begin{array}{cc} \text{H} & \text{D} \end{array} \\ \begin{pmatrix} 284 & 18 \\ 23 & 245 \end{pmatrix} \end{array}$$

(b) Random Forest (masonry)

$$\begin{array}{c} \\ \text{H} \\ \text{D} \end{array} \begin{array}{c} \begin{array}{cc} \text{H} & \text{D} \end{array} \\ \begin{pmatrix} 73 & 4 \\ 5 & 196 \end{pmatrix} \end{array}$$

(c) LeNet-based Network (concrete)

$$\begin{array}{c} \\ \text{H} \\ \text{D} \end{array} \begin{array}{c} \begin{array}{cc} \text{H} & \text{D} \end{array} \\ \begin{pmatrix} 278 & 24 \\ 12 & 256 \end{pmatrix} \end{array}$$

(d) LeNet-based Network (masonry)

$$\begin{array}{c} \\ \text{H} \\ \text{D} \end{array} \begin{array}{c} \begin{array}{cc} \text{H} & \text{D} \end{array} \\ \begin{pmatrix} 73 & 4 \\ 5 & 196 \end{pmatrix} \end{array}$$

(e) VGG-based Network (concrete)

$$\begin{array}{c} \\ \text{H} \\ \text{D} \end{array} \begin{array}{c} \begin{array}{cc} \text{H} & \text{D} \end{array} \\ \begin{pmatrix} 294 & 8 \\ 71 & 197 \end{pmatrix} \end{array}$$

(f) VGG-based Network (masonry)

Confusion matrices of the models (H: healthy, D: defect, Rows: ground truths. Columns: detected categories)

Model	Random forest	LeNet-based Network	VGG-based Network
Concrete	91.73%	**96.76%**	**96.76%**
Masonry	92.81%	**93.68%**	86.14%

Accuracy reached by the different models

Model	Random Forest	LeNet-based Network	VGG-based Network
Concrete	(88.76, 91.88)	(95.79, 96.16)	(95.79, 96.16)
Masonry	(92.83, 92.73)	(93.64, 93.79)	(88.32, 85.43)

(Precision,Recall) couples obtained by the different models (expressed in %)

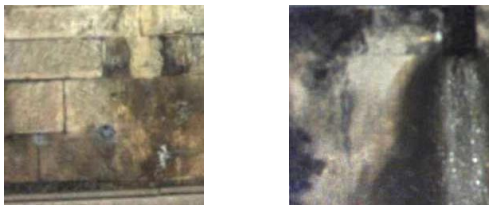
Figure 5: Results of the experimentation

In figure 6, we illustrate some examples of samples misclassified by both neural networks. The samples 6b and 6d have been classified as healthy (ground truth: defect). In these examples, the defect are located at the edge of the sample and hard to identify. The examples 6a and 6c are considered as defects by the algorithms. For the first one, the presence of a graffiti may explain the error. For

the second one, the variability in the rubble stones may have fooled the networks.

(a) Healthy (b) Defect

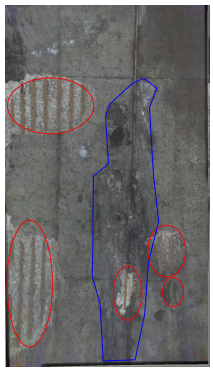
Concrete



(c) Healthy (d) Defect

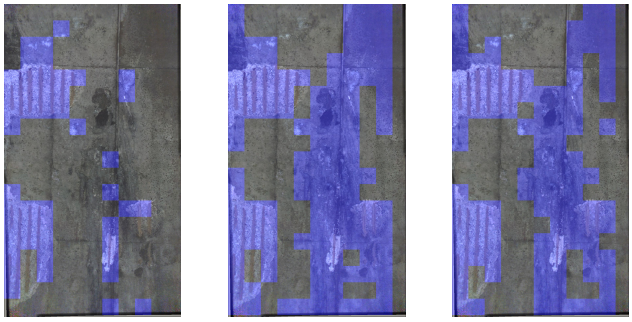
Masonry

Figure 6: Examples of samples simultaneously misclassified by both neural networks. The examples in the first column have been wrongly classified as defects and vice versa for the second column. The caption corresponds to the corresponding ground truth.



◯ Exposed iron □ Water ingress

(a) Test image with manually labeled anomalies



(b) Random Forest (c) LeNet-based Network (d) VGG-based Network

Figure 7: Detection of anomalies on an example of concrete linings (anomalies are highlighted in blue)

4.2 Detection

For the moment, we do not have ground truths for our full-frame image sequences, so an evaluation of the detection algorithm can only be qualitative. In figure 7, we present a representative result of the detection algorithm on a concrete lining. It can be seen that all three models provide correct results for exposed irons defects whereas the water ingress is poorly detected by the random forest. Neural networks perform well on that type of defect, at the expense of a higher number of false alarms. The high rate of false alarms may be explained by the small size of the learning dataset. Indeed, neural networks are known to need large datasets to be learned efficiently. We can also notice that the two neural networks, which have the same accuracy, yield close results.

5 Conclusion

The proposed learning methods have allowed us, within a limited setup (size of the learning and test database, two-class classification) to obtain competitive results in comparison with the state of the art. However, to conclude further on this point, it will be necessary to compare with models of the literature on common datasets and benchmarks.

The introduction of more complex models, combined with significantly expanded learning datasets, could address several limitations of our current implementation.

The first point is the limited representativeness of the data collected on a small number of tunnels, with a number of examples limited to one thousand for each type of lining. The data currently available are therefore only partially representative of the linings that can be observed in other tunnels or on structures involving other materials. With such a limited learning dataset, we can expect a quite poor capacity of generalization of our models.

A second limitation is the fact that we put all the defects in the same class for the classification process (*i.e.* binary classification). It would be desirable to obtain a more detailed characterization of the different types of defects or the severity level of the detected anomaly.

A third point of improvement is to move towards a more precise location of anomalies, avoiding the use of computationally expensive sliding window methods, leading to a direct detection or segmentation of defects. Recent neural architectures, such as [10], [17] or [18] could be employed for that purpose.

To address these different points, a priority will be to increase the size of the learning datasets, diversifying acquisitions in other tunnels, exploiting temporal redundancies in acquired sequences (a defect being seen from several

points of view) and using data augmentation methods. New acquisition campaigns are already planned. It will also be necessary to work on the deep models to increase their capacity of representation, while avoiding overfitting. Finally, we will develop different strategies to carry out learning transfers from pretrained models, adapted to the nature and volume of our new collected data [19].

References

[1] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[2] P. Foucher, M. D. Bah, P. Charbonnier, C. Boulogne, and C. Larive, "Classification automatique de défauts sur des images de tunnels par forêts d'arbres aléatoires," in *Congrès national sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA)*, (Clermont-Ferrand, France), pp. 1–2, 2016.

[3] P. Wang and H. Huang, "Comparison analysis on present image-based crack detection methods in concrete structures," in *3rd International Congress on Image and Signal Processing*, vol. 5, (Yantai, China), pp. 2530–2533, IEEE, 2010.

[4] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection," *IEEE Trans. Intell. Transportation Systems*, vol. 17, no. 10, pp. 2718–2729, 2016.

[5] T. Yamaguchi, S. Nakamura, and S. Hashimoto, "An efficient crack detection method using percolation-based image processing," in *3rd IEEE Conference on Industrial Electronics and Applications*, (Singapore, Singapore), pp. 1875–1880, IEEE, 2008.

[6] K. Makantasis, E. Protopapadakis, A. Doulamis, N. Doulamis, and C. Loupos, "Deep Convolutional Neural Networks for efficient vision based tunnel inspection," in *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, (Cluj-Napoca, Romania), pp. 335–342, IEEE, 2015.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, (San Diego, CA, USA, USA), pp. 886–893, IEEE, 2005.

[8] C. Feng, M.-Y. Liu, C.-C. Kao, and T.-Y. Lee, "Deep Active Learning for Civil Infrastructure Defect Detection and Classification," *Computing in Civil Engineering 2017*, pp. 298–306, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 770–778, IEEE, 2016.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[11] J. C. P. Cheng and M. Wang, "Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques," *Automation in Construction*, vol. 95, pp. 155–171, 2018.

[12] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[13] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *European Conference on Computer Vision (ECCV)*, (Berlin, Heidelberg), pp. 151–158, Springer Berlin Heidelberg, 1994.

[14] Y. Xiao, J. Wu, and J. Yuan, "mCENTRIST: A Multi-Channel Feature Generation Mechanism for Scene Categorization," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 823–836, 2014.

[15] Y. LeCun, "Generalization and network design strategies," in *Connectionism in perspective*, pp. 1–20, R. Pfeifer and Z. Schreter and F. Fogelman and L. Steels, elsevier ed., 1989.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, pp. 1–14, 2015.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–12, 2018.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–10, 2018.

[19] J. Yosinki, J. Clune, Y. Bengio, and H. Lispon, "How Transferable are Features in Deep Neural Networks?," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 2, (Montreal, Canada), pp. 3320–3328, MIT Press, 2014.