23^{rd} *Computer Vision Winter Workshop*
Zuzana Kúkelová and Júlia Škovierová (eds.)
Český Krumlov, Czech Republic, February 5–7, 2018

The effect of different CNN configurations on textured-surface defect segmentation and detection performance

Domen Rački[1], Dejan Tomaževič[1,2], Danijel Skočaj[3]
[1]Sensum, Computer Vision Systems
[2]University of Ljubljana, Faculty of Electrical Engineering
[3]University of Ljubljana, Faculty of Computer and Information Science
{domen.racki@sensum.eu, dejan.tomazevic@fe.uni-lj.si, danijel.skocaj@fri.uni-lj.si}

Abstract. *In this paper we examine the effect of different CNN configurations on segmentation and detection performance. We identified criteria for network architectures designed for the use in specific problem domains such as textured-surface defect segmentation and detection. Evaluation is performed on a dataset consisting of diverse textured surfaces with variously-shaped weakly-labeled defects where we achieve state-of-the-art results in terms of defect segmentation as well as classification.*

1. Introduction

Deep learning based approaches have proven superior to non-deep learning based methods in a variety of different tasks, ranging from detection [10] to segmentation [8]. A more-or-less common practice is to adapt an existing architecture, such as VGG [14], AlexNet [7] or others, for a specific task in a way that utilizes the pre-trained weights of the lower layers on large datasets and adapts the weights of upper layers for new problem domain. In general this approach is cumbersome when dealing with specific problem domains such as defect detection on surfaces. This is due to the fact that large architectures implicitly exhibit the need for large training sets which are usually not available in industrial environments, where the acquisition of defective examples presents a costly and impractical task.

We argue that since the underlaying defect structures and patterns in specific problem domains are limited, contrary to objects in datasets such as ImageNet, large architectures are not necessarily needed in order to successfully learn underlying defect patterns. As such, certain criteria for the network architecture can be identified. A network designed for an inspection system should: (i) Be compact, i.e., be able to learn potential defect detection from a handful of defective examples. (ii) Be robust, i.e., within a similar problem domain a network should require merely slight, if any, hyperparameter adjustments. (iii) Be explainable, i.e., be able to provide visual localization and classification explanation to a human domain expert, as this increases the overall trust in the system and reduces the need to blindly rely on hidden system processes. In this paper we explore whether a general rule-of-thumb design schema can be estimated for designing convolutional-neural-networks for the usage in specific problem domains such as defect detection on surfaces.

The remainder of the paper is structured as follows. In Section 2 we provide a brief overview of related work, followed by the description of the proposed architecture for defect detection in Section 3. The experimental setup is described in Section 4 while results are presented in Section 5. We conclude with the discussion in Section 6.

2. Related work

Classical approaches in defect detection on object surfaces follow more or less the same paradigm, i.e., a classifier — such as SVM, LDA, PCA, Hough transform, decision trees or KNN — trained on feature descriptors obtained from preprocessed images. Here, the preprocessing stage is crucial as it ensures that the problem is well-conditioned for the process of hand-engineering suitable features. Commonly used non-deep-learning defect detection approaches include: (i) Filtering approaches based on Wavelet transform [3], independent component analysis [13] and Gabor filter [18]; (ii) Structural ap-

proaches based on morphological operators [9] and edge detection [12]; (iii) Model based approaches such as the Hidden Markov model [11] and autoregressive models [1]; (iv) Statistical approaches based on histograms [1], co-occurrence matrices [4] and autocorrelation [6]. Deep learning highly contrast these approaches by performing automated feature learning instead of hand-designing suitable and at times sub-optimal features.

Deep learning, i.e., convolutional neural networks have proven superior in tasks where hand-designing features proves a difficult task. Early work on utilizing a CNN for surface defect detection can be found in [10]. The motivation arises from the aforementioned difficulty, where even domain specialists struggle to devise accurate rules based on geometrical and shape features for certain defects. The classification error is reduced by half over the classical approach with a classifier trained on feature descriptors, which included a Multi Layer Perceptron (MLP) and SVM with RBF classifiers trained on features obtained via HOG, PHOG, rotation invariant measure of local variance, and Local Binary Patterns (LBP, LBP-Fourier). Taking new deep learning research insights into account, [16] present an overview of different design heuristics of CNN for industrial inspection. The paper examines the impact of different hyper-parameter settings with respect to the accuracy for defect detection. Evaluation is performed on an artificial dataset, as shown in Figure 4, comprised of diverse surfaces on which the goal is to detect defects. Although the dataset consists of artificially generated images, these imitate diverse textured surfaces with variously shaped defects.

Other work on utilizing deep learning for defect detection can be found, such as learning from photometric stereo images of rail surface defects, where images depict differently colored light-sources illuminating the rail surfaces from different and constant directions, made visible in a photometric dark-field setup [15]. Or the usage of deep learning for non-trivial extraction of suitable features for the detection of rail surface defects from raw automated video recordings [2]. The aforementioned papers showcase the feasibility of utilizing deep learning for the problem of detecting defects on different surfaces.

3. Architecture and configurations

Considering the aforementioned criteria, we propose a compact convolutional network architecture

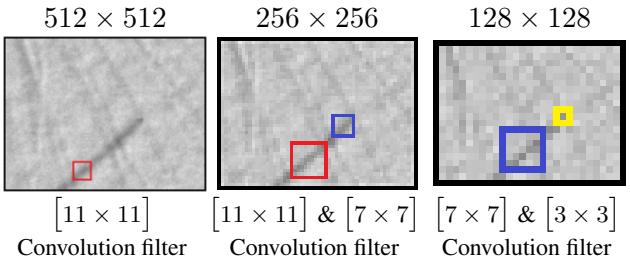

Figure 1. Convolutional filter dimensions with respect to the zoomed example containing a defect from Surface 4 show in Figure 4. The first image depicts the original input size of 512×512, while each subsequent image displays the size after subsampling. The red square represents a filter of size 11×11, the blue square a filter of size 7×7 and the yellow square a filter of size 3×3 pixels.

comprised of a segmentation and classification stage as shown in Figure 2. The task of the classification stage is to assign a given example a score which can be interpreted as the network's confidence that a given example contains a defect. The task of the segmentation stage is to provide visual defect localization and thereby visual classification explanation to a human domain expert. The segmentation part of the network consists of three blocks. Each block consists of three convolution layers, whereby the number of features is increased by a factor of two in each subsequent block as depicted in Figure 3. Depending on the filter configuration of these blocks the segmentation and detection performance is influenced. We examine six different configurations:

(i) *v1173* where the filter size in each subsequent block is decreased. When choosing a filter size, we can make some valid assumptions about the problem at hand and utilize these in the architecture design, i.e., the filters should enclose part of the defect that should be detected, as shown in Figure 1. Here we choose filters in such a way that the ratio between two subsequent filter sizes is kept constant during subsampling stages. This ensures that the filter continues to enclose part of the defect throughout the network. The actual representation within the network will differ from the depictions, however, this is merely to illustrate the reasoning. In general the coarse factor-of-two feature count step-size combined with a stride-of-two convolution at the beginning of a block, and a step-wise filter size reduction keeps the parameter count from increasing manifold, achieving compactness; (ii) *v3711* which is an inverse of configuration (i), meaning that the filter size is increased in each configuration block; (iii)

c1173 which follows the same principle as (i), however, the filters are configurated in a pyramid manner within each block, meaning that within each block filter sizes are varied from large to small; (iv) *c3711* which represents an inverse of configuration (iii); (v) *v333* which follows the VGG [14] design principle with a fixed filter size; (vi) *v111* which is similar to configuration (v) whereby no local context is taken into account.

The classification part of the network relies on the segmentation part. Classification scoring of a given example is achieved via a maximum and average global pooling combination from the segmentation layer (*SegLayer*) and compression layer (*CompLayer*) as illustrated in Figure 2. The segmentation layers is inspired by fully convolutional networks [8] and merely provides the segmentation output from one layer above. The compression layer hand on the other hand serves to compress the activation volume from one layer above. This reduces the number of parameters from which the classification score is estimated and robustifies the classification score. The usage of maximum and average global pooling proves robust in cases where for example a larger non-defective area would be segmented as a defective region. The maximum pooling alone would fail to see any difference between this and an example where merely a small defective area would be segmented. This is the underlying reason why we additionally perform average pooling, as this would exhibit a difference in the aforementioned example.

4. Experimental details

All experiments, that is, for all network configurations and all surfaces, are ran with a fixed set of learning parameters and network hyperparameters. The network architecture used is depicted in Figure 2 while the different network configurations are depicted in Figure 3. Given an input image of size 512×512 pixels, our network outputs a segmentation map of size 128×128 pixels. For all layers within the network the ReLU activation function is used after which batch normalization is applied. The only exceptions are the *SegLayer* and *S-neuron* where we use the linear and sigmoid activation function respectively. All network weights are initialized with a normal distribution centered around zero, as proposed in [5].

Network training is performed in two stages. In the first stage, i.e., *Segmentation stage* we train the defect segmentation step of the network for 25 epochs. In the second stage, i.e., *Classification stage* we train the classification step of the network for 10 epochs. The two stage training is essential. In the first stage all classification layers are frozen and merely the segmentation layers are learned. Conversely, in the second stage all the segmentation layers are frozen and merely the classification layers are trained. This ensures that the classification will be trained on meaningful segmentation representations. The two training stages are outlined in Figure 2.

In both cases the network is learning a regression value from either $[-1, 1]$, which is assigned to each pixel in the segmentation step, or $[0, 1]$ which is assigned to a single example in the classification step. During the segmentation stage training we minimize the mean squared-error loss function, i.e.,

$$\mathcal{S} = \frac{1}{np} \sum_{i=1}^{n} \sum_{j=1}^{p} ||x_i^{\langle j \rangle} - \hat{x}_i^{\langle j \rangle}||^2 \tag{1}$$

where n denotes the number of examples, p denotes the number of pixels, x_i the annotated pixel value and $\hat{x_i}$ the predicted pixel value. In the classification stage the binary cross-entropy loss function, i.e,

$$\mathcal{C} = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i}) \right] \tag{2}$$

is minimized, where n denotes the number of examples, y_i the ground truth example annotation and $\hat{y_i}$ the example regression prediction. Both functions are minimized using the Adadelta optimizer [17] with the parameters left at default values as suggested in the paper. All experiments were ran on a Gigabyte GeForce GTX 1080 Ti graphics card.

4.1. The DAGM dataset

The dataset for Industrial Optical Inspection [1] consists of artificially generated textured surfaces. As can be seen in Figure 4 the dataset consists of ten diverse surface classes with diverse defects emulating smudges, cracks, dents and impurities, each generated by a different texture and defect model. We refer to a given example as *positive* if it contains a defect and *negative* if it contains no defect.

Table 1 depicts the distribution of training and testing examples over the dataset. The entire dataset consists of 8050 train examples of which 1046 contain defects, and 8050 test examples of which 1054

[1] https://hci.iwr.uni-heidelberg.de/node/3616

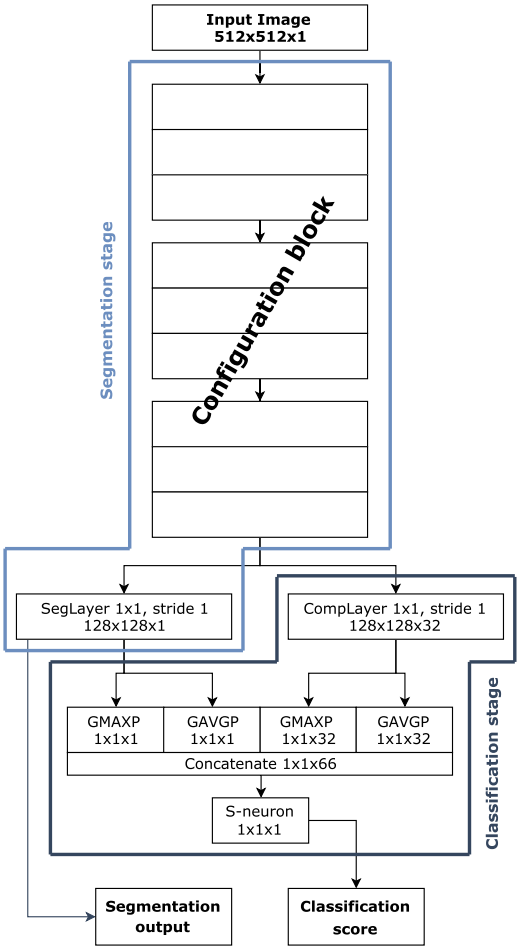

Figure 2. The proposed CNN-architecture which outputs a segmentation map and a probability value for a given example. The abbreviations ConvLayer, SegLayer and CompLayer stand for convolutional, segmentation and compression layer respectively. Abbreviations GMAXP and GAVGP stand for global maximum pooling and global average pooling, and S-neuron stands for scoring neuron. Different configuration blocks used are depicted in Figure 3.

contain defects. If a given surface of size 512x512 pixels contains a defect, it contains exactly one weakly labeled defect on the background texture. Weak labels are provided in form of ellipses which roughly indicate a defective are on a given example but also include defect-free areas to some extent, as shown in Figure 4. Although the entire defect is contained within the encircling ellipse, a significant portion of the regular surface is encircled as well. Consequently, many image pixels are falsely labelled, which can affects the learning process. This problem is however to be expected in many real world situations since very precise annotations of surface defects are very difficult and costly to obtain.

Surface	Train examples		Test examples	
	Positive	Negative	Positive	Negative
1	79	496	71	504
2	66	509	84	491
3	66	509	85	490
4	82	493	68	507
5	70	505	81	494
6	83	492	67	508
7	150	1000	150	1000
8	150	1000	150	1000
9	150	1000	150	1000
10	150	1000	150	1000

Table 1. Distribution of train and test examples over the DAGM dataset.

To our knowledge, this is the only publicly available annotated surface-defect dataset. The underlying reason of why such datasets are hard to come by are non disclosure agreements which serve to prevent the disclosure of company secrets such as image acquisition or other crucial processes which ensure a competitive advantage.

5. Results

We evaluate the performance of our network in terms of the true positive rate (TPR), i.e., positives that are correctly identified as positives and true negative rate (TNR), i.e., negatives that are correctly identified as negatives. We also evaluate the performance of our network in terms of the absolute number of misclassified test examples. Table 3 depicts the performance of our network with different configurations. As can be seen, the network when trained with configuration *v1173* fails to detect merely one positive test example from Surface 4, while maintaining a high negative, i.e., defect-free example detection accuracy. Conversely, configuration *v111*, which due to filters of size one-pixel captures no local context during the learning process, is as such unable to distinguish defects from the background. Table 2 shows the performance of our network with configuration *v1173* compared to a state of the art deep learning based approach proposed in [16]. Our network architecture outperforms the latter in terms of the the detection of defective examples, while maintaining a high defect-free example detection.

Although the high-level performance aspect of the network configurations seem promising, a qualitative analysis of the segmentation outputs provides further insights. Figure 6 depicts the segmentation outputs

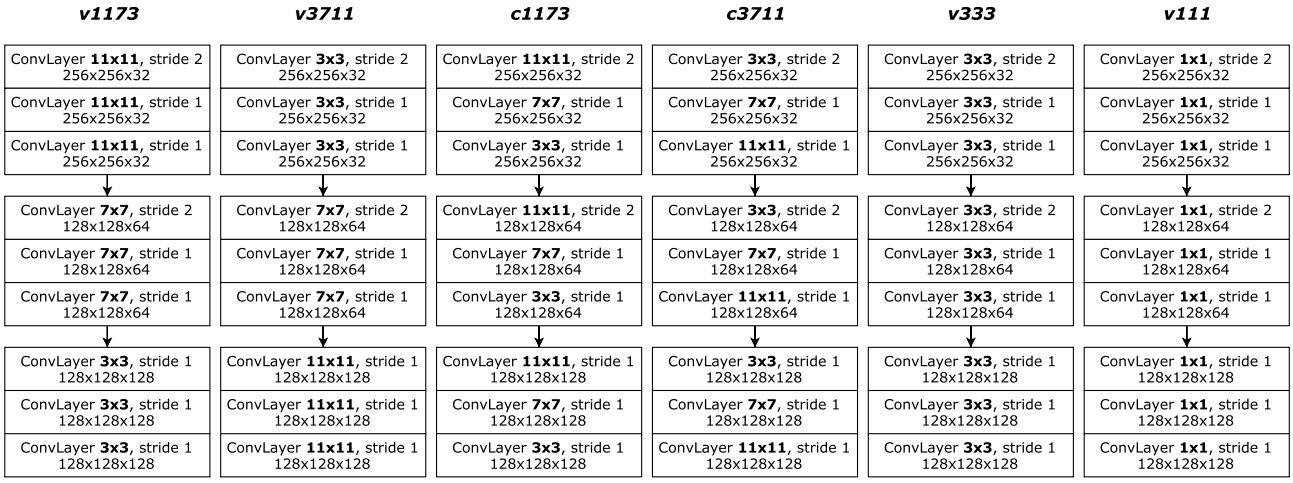

Figure 3. Different configuration blocks used with the network architecture depicted in Figure 2.

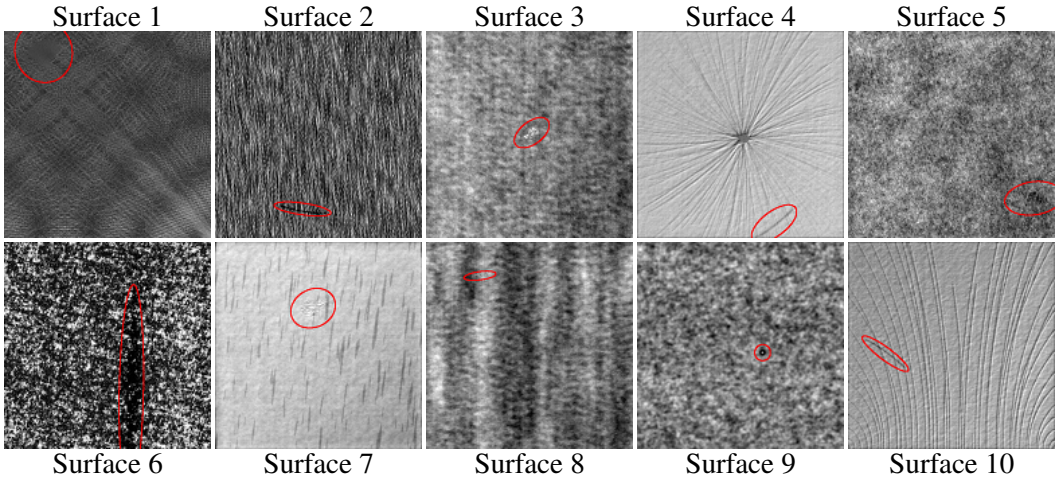

Figure 4. A snapshot of the diverse textured-surfaces in the dataset. Each surface class exhibits additional intra-class variation of the background texture. Red ellipses present coarse surface defect labeling, i.e., weakly labeled ground truth annotations as these include areas which do not correspond to defects.

of the network for a few given examples. From these it is clear that configuration *v111* is unable to separate defects from the background, due to the fact that no local context was considered in the learning stage. Configuration *v333* takes local context into account, however does so in a small scale and as a result false artifact segmentations remain on the background texture. Configurations *v1173* and *c1173* both account for local context in a pyramid fashion, where as configurations *v3711* and *c3711* account for local context in an inverse-pyramid fashion, either globally within the network architecture or locally within each configuration block.

Table 4 gives the ratio of noisy background segmentations for each configuration. The ratio is computed by binarizing the background segmentations at the value off 0.15, and computing the relative number of falsely segmented background pixels w.r.t. the number of all background pixels for a given example. The ratio is computed separately for and averaged over positive, i.e., *POS* and negative, i.e., *NEG* examples for each surface. As can be seen configurations *v1173* and *v333* exhibit least noisy background segmentations — this can also be seen in Figure 6 for an example from Surface 1.

Figure 5 provides a performance summary of evaluations in Table 3 and Table 4. As can be seen, the configuration *v1173* exhibits the highest detection performance and the lowest ratio of falsely segmented background pixels in negative images, meaning that the filter configuration is quite robust to textured-backgrounds, and activations are most likely to occur merely on defect regions.

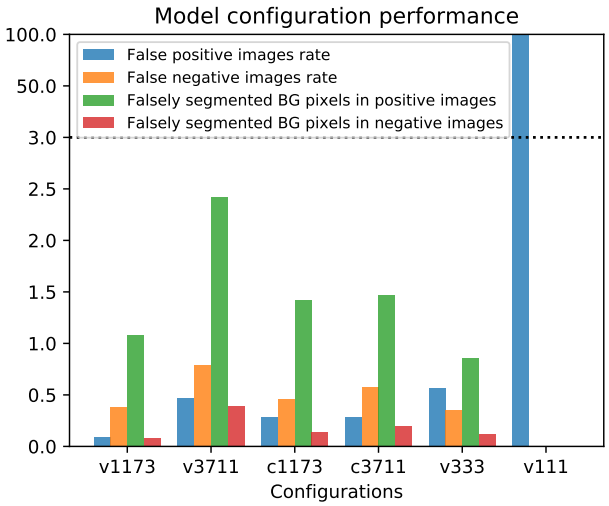

Figure 5. A summary of the performance evaluation in Table 3 and Table 4.

Surface	*Ours*		*Weimer et. al. [16]*	
	TPR	TNR	TPR	TNR
1	100	98.8	100	100
2	100	99.8	97.3	100
3	100	96.3	100	95.5
4	98.5	99.8	98.7	100
5	100	100	100	98.8
6	100	100	99.5	100
7	100	100	-	-
8	100	100	-	-
9	100	99.9	-	-
10	100	100	-	-

Table 2. Classification performance of our proposed CNN architecture with configuration *v1173* vs Weimer et. al. [16].

6. Conclusion

We examined the effect of different CNN configurations on segmentation and detection performance. We identified criteria for network architectures designed for the use in specific problem domains such as textured-surface defect segmentation and detection. We evaluated different network configurations on a dataset consisting of diverse textured surfaces with variously-shaped weakly-labeled defects.

We achieve state-of-the-art results in terms of defect segmentation as well as classification with the configuration *v1173*. The latter accounts for local context in a global pyramid-fashion by decreasing the filter size in each subsequent convolution block. The proposed configurations meets the identified criteria; (i) compactness in terms of parameter count; (ii) robustness across diverse surface textures; (iii) explanation as it provides a segmentation and classification score output.

References

[1] H.-G. Bu, X.-B. Huang, J. Wang, and X. Chen. Detection of fabric defects by auto-regressive spectral analysis and support vector data description. *Textile Research Journal*, 80(7):579–589, 2010. 2

[2] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, and B. De Schutter. Deep convolutional neural networks for detection of rail surface defects. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 2584–2589. IEEE, 2016. 2

[3] M. Ghazvini, S. Monadjemi, N. Movahhedinia, and K. Jamshidi. Defect detection of tiles using 2d-wavelet transform and statistical features. *World Academy of Science, Engineering and Technology*, 49:901–904, 2009. 1

[4] R. M. Haralick, K. Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 3(6):610–621, 1973. 2

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 3

[6] E. Hoseini, F. Farhadi, and F. Tajeripour. Fabric defect detection using auto-correlation function. *International Journal of Computer Theory and Engineering*, 5(1):114, 2013. 2

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. 1

[8] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1, 3

[9] K.-L. Mak, P. Peng, and K. Yiu. Fabric defect detection using morphological filters. *Image and Vision Computing*, 27(10):1585–1592, 2009. 2

[10] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout. Steel defect classification with max-pooling convolutional neural networks. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2012. 1, 2

[11] A. Mukherjee, S. Chaudhuri, P. K. Dutta, S. Sen, and A. Patra. An object-based coding scheme for frontal surface of defective fluted ingot. *ISA transactions*, 45(1):1–8, 2006. 2

Surface	*v1173*		*v3711*		*c1173*		*c3711*		*v333*		*v111*	
	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
1	100 [0]	98.8 [6]	100 [0]	93.1 [35]	100 [0]	95 [25]	97.2 [2]	93.5 [33]	98.6 [1]	98.4 [8]	0 [71]	100 [0]
2	100 [0]	99.8 [1]	100 [0]	100 [0]	100 [0]	99.8 [1]	100 [0]	100 [0]	100 [0]	100 [0]	0 [84]	100 [0]
3	100 [0]	96.3 [18]	97.6 [2]	100 [0]	98.8 [1]	99 [5]	100 [0]	99.2 [4]	100 [0]	99.8 [1]	0 [85]	100 [0]
4	98.5 [1]	99.8 [1]	95.6 [3]	96.3 [19]	98.5 [1]	100 [0]	100 [0]	99.4 [3]	92.6 [5]	96.8 [16]	0 [68]	100 [0]
5	100 [0]	100 [0]	100 [0]	99.8 [1]	100 [0]	99.8 [1]	100 [0]	100 [0]	100 [0]	100 [0]	0 [81]	100 [0]
6	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	0 [67]	100 [0]
7	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	0 [150]	100 [0]
8	100 [0]	100 [0]	100 [0]	100 [0]	99.3 [1]	100 [0]	99.3 [1]	100 [0]	100 [0]	100 [0]	0 [150]	100 [0]
9	100 [0]	99.9 [1]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	0 [150]	100 [0]
10	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	100 [0]	0 [150]	100 [0]
Σ	[1]	[27]	[5]	[55]	[3]	[32]	[3]	[40]	[6]	[25]	[1056]	[0]
params	1,133,732		5,488,292		2,501,796		3,759,780		491,172		64,932	

Table 3. Classification performance of the proposed CNN architecture (Figure 2) with different configurations as depicted in Figure 3. The results are obtained by taking the TPR and TNR at the highest classification accuracy on the training set, and classifying the examples in the testing set. Numbers in the square brackets denote the absolute number of examples that were misclassified. The bottom numbers indicate the total number of misclassified examples w.r.t. the total number of examples in the testing set as shown in Table 1, which amount to 1056 for TPR and 6994 for TNR. The number of parameters for each configuration is denoted by *params*.

Surface	*v1173*		*v3711*		*c1173*		*c3711*		*v333*		*v111*	
	POS	NEG	POS	NEG	POS	NEG	POS	NEG	POS	NEG	POS	NEG
1	0.035	0.008	0.097	0.037	0.055	0.014	0.065	0.020	0.050	0.012	0.000	0.000
2	0.002	0.000	0.003	0.000	0.004	0.000	0.005	0.000	0.002	0.000	0.000	0.000
3	0.018	0.000	0.035	0.000	0.009	0.000	0.011	0.000	0.003	0.000	0.000	0.000
4	0.003	0.000	0.007	0.002	0.005	0.000	0.005	0.000	0.001	0.000	0.000	0.000
5	0.001	0.000	0.034	0.000	0.001	0.000	0.005	0.000	0.002	0.000	0.000	0.000
6	0.039	0.000	0.034	0.000	0.046	0.000	0.038	0.000	0.019	0.000	0.000	0.000
7	0.004	0.000	0.009	0.000	0.008	0.000	0.007	0.000	0.002	0.000	0.000	0.000
8	0.003	0.000	0.019	0.000	0.006	0.000	0.004	0.000	0.003	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.001	0.000	0.002	0.000	0.001	0.000	0.000	0.000
10	0.003	0.000	0.004	0.000	0.007	0.000	0.005	0.000	0.003	0.000	0.000	0.000

Table 4. The ratio of noisy background segmentations for each configuration. The ratio is computed by binarizing the background segmentations at the value off 0.15, and computing the relative number of falsely segmented background pixels w.r.t. the number of all background pixels for a given example.

[12] F. S. Najafabadi and H. Pourghassem. Corner defect detection based on dot product in ceramic tile images. In *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*, pages 293–297. IEEE, 2011. 2

[13] W. Polzleitner. Defect detection on wooden surface using gabor filters with evolutionary algorithm design. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 1, pages 750–755. IEEE, 2001. 1

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 3

[15] D. Soukup and R. Huber-Mörk. Convolutional neural networks for steel surface defect detection from photometric stereo images. In *International Symposium on Visual Computing*, pages 668–677. Springer, 2014. 2

[16] D. Weimer, B. Scholz-Reiter, and M. Shpitalni. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals-Manufacturing Technology*, 2016. 2, 4, 6

[17] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 3

[18] H. Zheng, L. X. Kong, and S. Nahavandi. Automatic inspection of metallic surface defects using genetic algorithms. *Journal of materials processing technology*, 125:427–433, 2002. 1

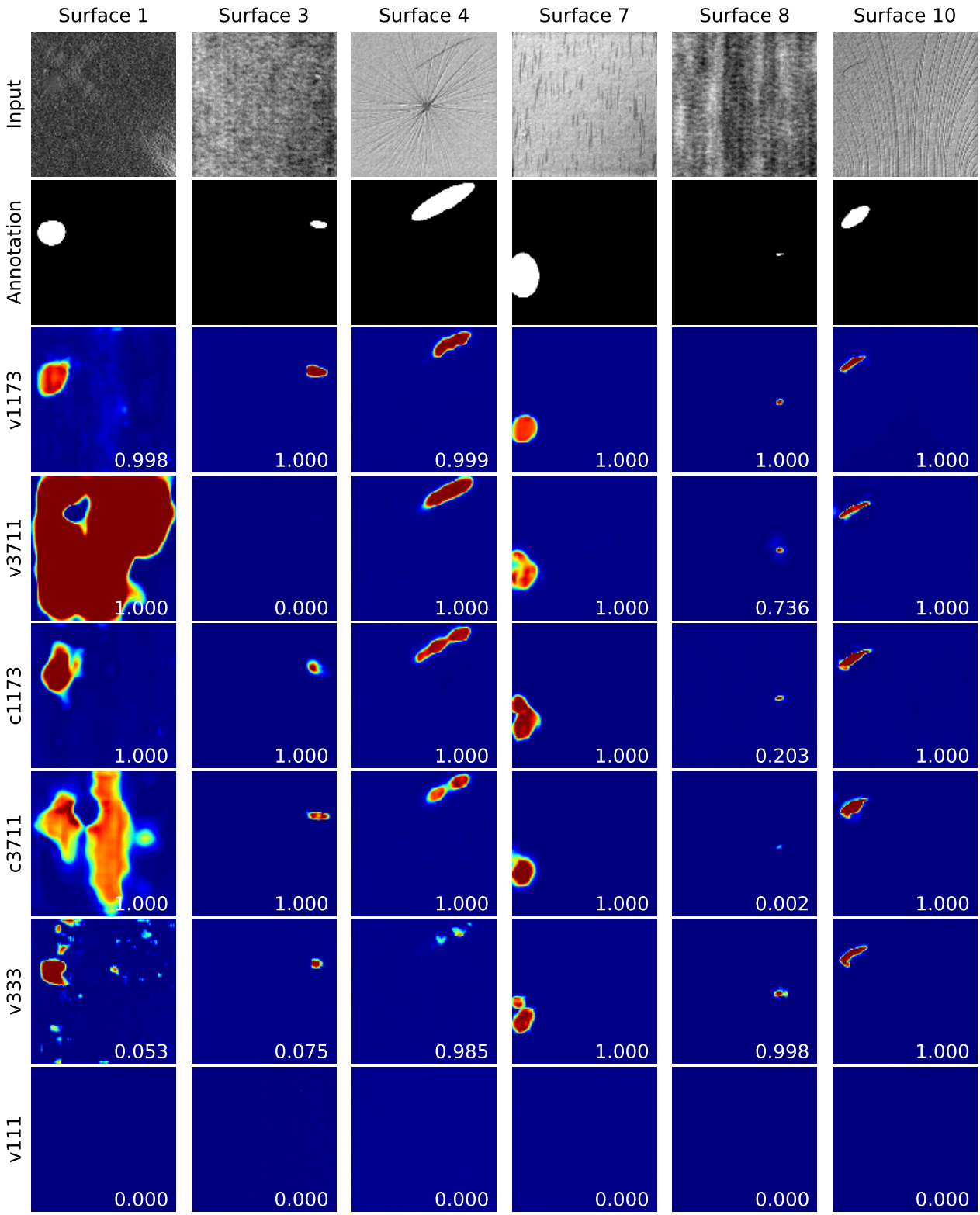

Figure 6. Segmentation results on different examples. The individual numbers in the bottom-right corners indicate the assigned scores to each segmentation output.