Adaptive Architectures for Semantic Segmentation in the Field of Sensor-Based Sorting Systems

Felix Kronenwett[1], Roman Lehmann[2], Haoxiang Zheng[1], Georg Maier[1], Thomas Längle[1] and Wolfgang Karl[2]

[1] Fraunhofer IOSB, Institute of Optronics, System Technologies and Image Exploitation, Fraunhoferstr. 1, 76131 Karlsruhe, Germany
[2] Research Group Computer Architecture and Parallel Processing (CAPP), Institute of Computer Engineering (ITEC), Karlsruhe Institute of Technology (KIT), Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany

Abstract In recent years, the demand for efficient and accurate sorting solutions across various industries has surged due to the need for enhanced material recovery and sustainability. Sensor-based sorting systems have emerged as pivotal technologies. They employ visual inspection to ensure the precise classification and sorting of bulk materials. Several challenges hinder the potential of deep learning models in industrial systems for image data analysis in complex sorting tasks. Due to different hardware, traditional static deep learning models often fail to handle the dynamic requirements of varying material throughput and execution times, leading to inefficient sorting accuracy. This paper investigates the integration of adaptive architectures for semantic segmentation. These architectures dynamically adjust their computation pathways based on input complexity, optimizing performance and resource utilization. Implementing an architecture with early exit mechanisms improved accuracy, enabling sorting decisions regardless of hardware limitations. Experimental validation using real-world data from sorting plants demonstrates adaptive models' practical applicability and benefits for sensor-based sorting systems.

Keywords Adaptive architectures, semantic segmentation, sensor-based sorting, early-exit mechanisms

1 Introduction

Sensor-based sorting is a real-time application in the field of machine vision. Corresponding systems provide solutions for separating single particles from a material stream and are used in various industries such as recycling, food processing, and waste management [1].

The functional principle of a common kind of sensor-based sorting systems is as follows. Bulk material particles are transported on a conveyor belt. After being discharged from the belt, they pass through an inspection line and are detected by imaging sensors. The recorded data is analyzed to make sorting decisions for the physical separation of different material classes. Individually activated, fast-switching pneumatic valves, arranged orthogonally to the main transport direction of the conveyor belt, are used to eject individual objects. Assuming a constant speed of the bulk material due to conveyor belt transport, a fixed delay time between detection and ejection can be established, resulting in real-time requirements to initiate physical separation promptly [2].

1.1 Motivation

A crucial subsystem of sensor-based sorting systems is the visual inspection unit. For sorting tasks based on color and shape, color sensors are used. In this case, the aim of data evaluation is semantic segmentation, typically by assigning each pixel to a specific material class. However, in demanding sorting tasks with complex materials with similar optical properties, this is challenging.

Particularly in the area of image analysis, significant progress has been made thanks to innovative processes from the field of deep learning [3, 4]. Yet, sensor-based sorting systems face various challenges that require different aspects of adaptability from the neural networks employed that can be summarized as:

- Varying material throughput and load
- Varying hardware and execution times
- Wasteful computations on simple inputs

A particular challenge for the use of deep learning models in sensor-based sorting systems is the long execution time [5]. Generally, the

timing constraints depend on the distance between the sensor and the ejection unit. In addition, the model inference on a graphics processing unit (GPU) is subject to time jitter. The deployment of a deep learning model is therefore difficult, as not only the time constraints can change, but also the execution time due to the different hardware used. In standard neural networks, the execution time remains relatively constant due to static inference, where the compiotational graph and parameters remain fixed after training.

This leads to the concept of adaptive networks, which can adapt their structure or parameters to the input during inference. An adaptive network is often realized using early-exit architectures, where intermediate exits are placed within the neural network. These intermediate results can be utilized at different points during inference, and increasing computation time yields more accurate results. When used in sensor-based sorting, adaptive networks have the advantage that sorting decisions can be made regardless of hardware properties. The time between inspection and ejection can be optimally utilized by making results more accurate through more complex, therefore time-consuming evaluation, but still always reliably delivering timely results.

1.2 Contribution

This paper explores the use of adaptive architectures for semantic segmentation of sensor data, emphasizing their potential to enhance the use of deep learning models in sensor-based sorting systems. By using adaptive networks that adapt their structure depending on the situation, these architectures promise improved accuracy, flexibility, and robustness. They offer a significant advantage by optimizing the time between material inspection and rejection, ensuring timely and precise sorting decisions regardless of hardware limitations.

1.3 Related Work

A deep neural network (DNN) can be formulated as a function $y = f(x) = f_L(f_{L-1}(\dots f_1(x))$, where x is the input data, L is the number of layers in the DNN and f_i is the differentiable operator at layer i. The output of a layer i can be denoted as $h_i = f_i(h_{i-1})$, where h_{i-1} is the output of the previous layer $i-1$.

To convert a DNN to an early-exit architecture, an early-exit branch $c_b(h_b) = y_b$ is placed at every selected branch location $b \in B \subseteq \{1, \ldots, L\}$. c_b can be an intermediate classifier or regressor, producing the early result y_b. We call the original network $f(x)$ the backbone.

Early works for early exits were introduced in classification tasks [6–8]. Later, the idea was further developed and improved. Depending on the goal required, the quality of the decision, the speed of the decision, or both were improved [9]. The field of application has also been extended to various other areas. For example, NLP [10–12] and image segmentation [13–17] have been investigated using this technique.

For image segmentation, MLSNet [14] implements an early exit mechanism by incorporating a hierarchically connected decoder with ResNet [18] as the backbone and multiple classifiers at different layers. Experimental results in the complex Cityscapes [19] and BDD100K [20] datasets, two datasets based on various street scene images with pixel-level annotations, demonstrate that MLSNet surpassed the baseline models in terms of both accuracy and inference speed.

On the other hand, ADP-C (Anytime Dense Prediction with Confidence) [15] uses HRNet [21] as its backbone. When confidence is high enough, further computation for pixels is minimized. Once a pixel reaches sufficient confidence, its prediction is retained for subsequent exits without additional processing. ADP-C shows better performance on the Cityscapes dataset compared to the baseline.

The above works show that the networks were tested on well-known benchmark datasets (Cityscapes, BDD100K, etc.). We want to extend these ideas to real-world sorting system scenarios. In this work, we show how we adjust the network architecture to fulfill the given task and test the networks on real data.

2 Methods and Datasets

2.1 Datasets

Collecting real, labeled data from a sensor-based sorting system is expensive and challenging, and no large open datasets exist. Therefore, the datasets used for training and validation in our study are generated synthetically. Firstly, we record real data for the materials to be sorted in the lab. Individual instances are then algorithmically cut out from

actual recordings using basic image processing algorithms. Instances are randomly selected, rotated, and arranged on a background. Using this synthesis method, the labels required for training can be generated automatically. The synthesized images contain complex situations, i.e., highly occupied images with numerous object overlaps [5]. For the material types to be sorted, we select three groups, and each group consists of two kinds of material: raisins and coffee beens, bricks and sand-lime bricks, and peanuts and hibiscus, as shown in Figure 1. We use real data captured from the device for testing, and the labels are manually generated.

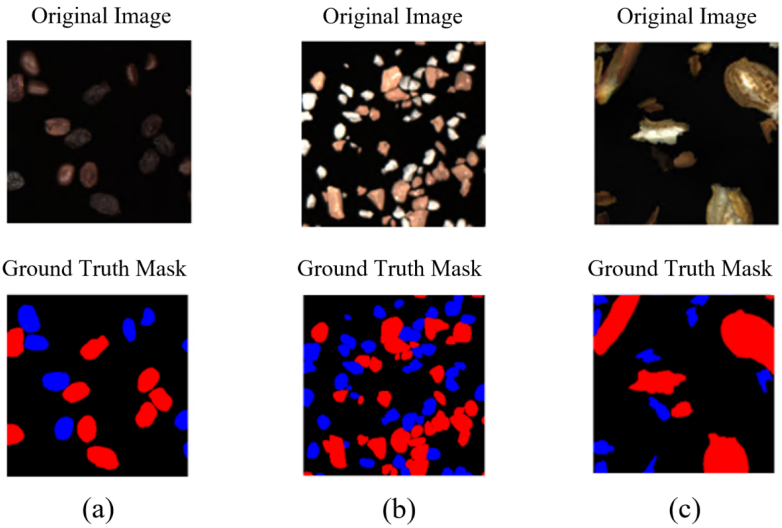

Figure 1: Synthetically generated training dataset: (a) raisins and coffee beens, (b) bricks and sand-lime bricks, (c) peanuts and hibiscus.

2.2 Adaptive Architectures

The difference between our datasets and the datasets of other domains in the literature is their complexity. Our data sets usually consist of very few classes, which means that the neural network supposed to

make the decisions does not have to be as complex as in the literature. We adapt the work from [14] by reducing the complexity. They use an encoder-decoder structure for segmentation. We replaced the backbone (encoder), originally consisting of a ResNet-18 [18], with a much narrower backbone. Now, the backbone consists of only a few layers. We further specify that the final architecture should have three exits. Every exit can be seen as a decoder to fulfill the encoder-decoder structure.

In Figure 2, the backbone is marked with a blue box, and each exit's level is highlighted differently. The feature maps of each level are computed sequentially, starting from level 1. The decoders consist of convolution layers with kernel size 3×3 followed by batch normalization, a ReLU-activation, and max-pooling. For upsampling in the decoder, bilinear interpolation is used, concatenated to the previous activations. The stride indicates the feature map dimension ratio to the final output resolution. The final classification is done by a 1 convolution to produce the multiscale segmentation outputs.

We use FLOP (Floating-Point Operation) as a measurement of the complexity of a neural network. FLOPs indicate how many floating-point operations are needed to calculate the result of the neural network to a given input. The advantage of FLOPs compared to a time measurement is the independence of the underlying hardware. Our architecture consists of two early exits (1 + 2) and the final exit (3) at the End. The first exit needs $0.038G$ FLOPs to calculate the desired output. The second exit needs $0.096G$ FLOPs, and the final (third) exit needs $0.18G$ FLOPs. In Table 1 you can see the difference in FLOPs of our model compared to [14].

Table 1: Comparison of FLOPs between the established and adapted model architecture in gigaflops.

Exit	New Model (FLOPs in G)	Old Model (FLOPs in G)
Exit 1	0.038	3.4
Exit 2	0.096	4.7
Exit 3	0.180	6.5

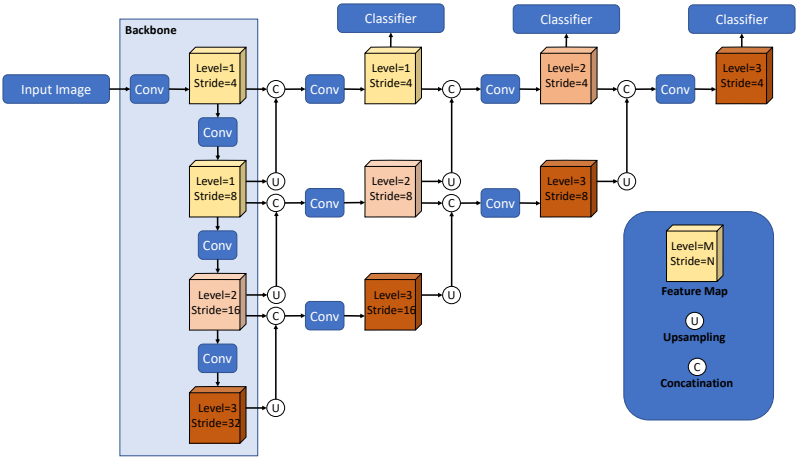

Figure 2: Our adapted MLSNet with a smaller backbone. „Conv“consists of a convolution operator with a kernel size of 3×3, followed by a batch normalization and a ReLU.

3 Experimental Results

In order to better understand the results of our own model, we first train and evaluate three established static architectures for semantic segmentation on the three datasets. This is followed by training and evaluation with the customized, own model. We trained all models on an NVIDIA Tesla T4 GPU with a batch size of 16. We used the ADAM optimizer [22] with a learning rate of 10^{-3} and a weight loss of 10^{-3} for further improvement. The PyTorch Lightning Framework was used to streamline training and monitoring. In particular, we use the mean intersection over union (mIoU) as a metric for our evaluation's semantic segmentation accuracy. With the number of classes n and $p_{x,y}$ representing the number of pixels of class x, that are predicted to class y, we can define mIoU as

$$mIoU = \frac{1}{n} \sum_{i=1}^{n} \frac{p_{i,i}}{\sum_{j=1}^{n} p_{i,j} + \sum_{j=1}^{n} p_{j,i} - p_{i,i}}. \tag{1}$$

The achieved accuracy is analyzed at various exits. The correlation between Accuracy and FLOPs, as a measure of possible interference speed, is specifically considered.

3.1 Baseline Semantic Segmentation Models

Three established semantic segmentation models were trained on the three described datasets to determine achievable segmentation accuracies. Especially regarding the fact that synthetic training data is used, but testing is done with real, manually labelled data The results of the mIoU metric with FCN32s [23], U-Net [24], and SegNet [25] models are shown in Table 2. U-Net is the largest model, with 45 GFLOPs, and it achieves the highest mIoU on the test data for all three datasets. The other two models each performed slightly worse. The results on the datasets also differ; the peanut/hibiscus dataset achieves very high mIoUs, over 90% in some cases. The brick/sand-lime brick data set seems to be the most difficult. This may be due to a high degree of similarity between the two classes, a high difference between training and actual data, or synthetic errors in generating synthetic data.

Table 2: Class-level mIoU (%) of baseline models (numbers in parentheses indicate FLOPs) on various datasets.

Model	Datasets		
	Raisins and Coffee	Bricks and Sand-Lime	Peanuts and Hibiscus
FCN32s (19G)	87.2	73.3	91.2
U-Net (**45G**)	**90.1**	**85.4**	**95.1**
SegNet (11G)	88.0	81.3	93.0

3.2 Early Exit Model

In the following, our own modified MLSNet was trained. All three early exists were trained simultaneously, and the model was then analyzed based on its performance on the test data. Particular attention is paid to the discrepancy between the different outputs. The results can be found in Table 3. FLOPs also provide the number of required computing operations and thus an estimate of the runtime.

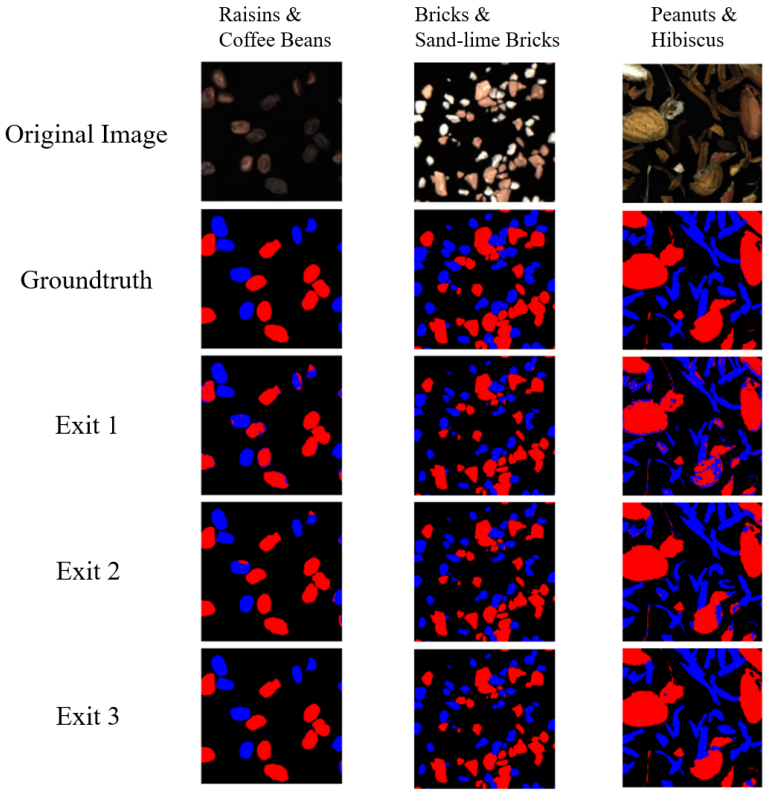

Figure 3: Comparison of the visual quality of the results of the different exits of our model for the three datasets.

Firstly, we look at the performance achieved at different points in time at the various outputs of the models. As expected, the Exit 3 output has the highest segmentation accuracy on all datasets. The achieved mIoUs are slightly lower than the performance of the baseline models, but this can be explained by the significantly smaller size of the architecture due to the lower depth and frequency of the convolutional layers. While the Exit 3 output has 0.18 GFLOPs, the U-Net has 45 GFLOPs.

In addition, increased segmentation performance can be seen with increasing output. The difference between the first and last exit is in the range of Δ7%. However, the FLOPs of the two exits have already more than quadrupled (factor 4.7).

The visual observation of the sorting results at the three exits in Figure 3 shows the improvement in segmentation performance. However, it can be argued that sorting would already be possible with the result at Exit 1 output in a sensor-based sorting scenario.

Table 3: Exit-level mIoU (%) of the i-th exit classifier (numbers in parentheses indicate FLOPs) of our Model on various datasets.

Model	Datasets		
	Raisins and Coffee	Bricks and Sand-Lime	Peanuts and Hibiscus
Exit 1 (0.038G)	80.6	76.8	78.0
Exit 2 (0.096G)	85.1	80.0	91.5
Exit 3 (0.18G)	**87.6**	**82.9**	**93.5**

3.3 Training Strategy Evaluation

For a better understanding of the model architecture, the existing model was split into three individual models, each of which was reduced to just one of the three outputs. The three models were then trained separately to enable a comparison between the separate and end-to-end training of the mixed model. The results in Table 4 show that the combined model with all exits together performs better than the results obtained at the same exit of the individual model. The previous exits can therefore benefit from the later layers and exits during training and enable better generalization. This behavior fits the known phenomenon called *Knowledge Distillation* [26]. As a result, the model is not only more flexible in use but also performs better than individual models of the same size that have been specifically customized.

4 Conclusion and Future Work

This paper shows how early-exiting neural networks as adaptive architecture can perform on sensor-based sorting systems. Such techniques

Table 4: Exit-level mIoU (%) of the i-th exit classifier using separate training strategies (numbers in parentheses indicate FLOPs).

Model	Datasets		
	Raisins and Coffee	Bricks and Sand-Lime	Peanuts and Hibiscus
Exit 1 (0.038G)	72.3	71.4	77.9
Exit 2 (0.096G)	84.5	79.4	90.9
Exit 3 (0.18G)	87.3	80.8	93.0

offer advantages in accuracy and the ability to adjust resource needs. Furthermore, with later exits, earlier exits can profit in training such that these exits can outperform networks with similar FLOPs without early exits. Due to this, training a big network with an early exit has to be considered, even if the big network can't be used in production.

In the future, we are eager to investigate more neural architectures and datasets. We also want to prove this technique in production with a real sorting system.

References

1. Y. Zhao and J. Li, "Sensor-Based Technologies in Effective Solid Waste Sorting: Successful Applications, Sensor Combination, and Future Directions," *Environmental Science & Technology*, Nov. 2022, publisher: American Chemical Society.
2. G. Maier, R. Gruna, T. Längle, and J. Beyerer, "A survey of the state of the art in sensor-based sorting technology and research," *IEEE Access*, vol. 12, pp. 6473–6493, 2024.
3. F. Kronenwett, G. Maier, N. Leiss, R. Gruna, V. Thome, and T. Längle, "Sensorgestützte Charakterisierung von Bauschutt bei hohen Belegungsdichten mittels Deep Learning," in *Vorträge-Konferenzband Zur 16. Recy & DepoTech-Konferenz*. Leoben, Österreich: Montanuniversität Leoben, Lehrstuhl für Abfallverwertungstechnik und Abfallwirtschaft, 2022.
4. F. Kronenwett, G. Maier, N. Leiss, R. Gruna, V. Thome, and T. Längle, "Sensor-based characterization of construction and demolition waste at high occupancy densities using synthetic training data and deep learning," *Waste Management & Research*, p. 0734242X241231410, 2024.
5. F. Kronenwett, G. Maier, and T. Längle, "Potential of deep learning methods for image processing in sensor-based sorting: data generation, train-

ing strategies and model architectures," *10th Sensor-Based Sorting & Control 2024*, p. 45, 2024.

6. E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo, "Big/little deep neural network for ultra low power inference," in *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2015, pp. 124–132.
7. S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," *CoRR*, vol. abs/1709.01686, 2017.
8. Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3301–3310.
9. G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," *CoRR*, vol. abs/1703.09844, 2017.
10. J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "Deebert: Dynamic early exiting for accelerating BERT inference," *CoRR*, vol. abs/2004.12993, 2020.
11. W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, "Fastbert: a self-distilling BERT with adaptive inference time," *CoRR*, vol. abs/2004.02178, 2020.
12. J. Wang, K. Chen, G. Chen, L. Shou, and J. McAuley, "Skipbert: Efficient inference with shallow layer skipping," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 7287–7301.
13. X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
14. S. Yokoo, S. Iizuka, and K. Fukui, "Mlsnet: Resource-efficient adaptive inference with multi-level segmentation networks," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 1510–1514.
15. Z. Liu, T. Darrell, and E. Shelhamer, "Confidence adaptive anytime pixel-level recognition," *CoRR*, vol. abs/2104.00749, 2021.
16. C.-H. Kung and C.-R. Lee, "Add: A fine-grained dynamic inference architecture for semantic image segmentation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4792–4799.

17. A. Kouris, S. I. Venieris, S. Laskaridis, and N. Lane, "Multi-exit semantic segmentation networks," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 330–349.

18. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

19. M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

20. F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell *et al.*, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

21. J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021.

22. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

23. 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 (CVPR)*, June 2015.

24. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

25. V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

26. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.