DOI: https://doi.org/10.30837/IST-2024.P1.169

Semantic Image Segmentation on Autonomic Devices: Prospects for Implementing Deep Learning Models

Mykhailo Trusov[1*†] та Dmitro Uzlov[1†]

[1] *V.N. Karazin Kharkiv National University*

Abstract

The implementation of deep learning models for semantic segmentation on autonomic devices is a promising direction in developing intelligent systems capable of analyzing visual information independently. The present study explores the possibilities and challenges of using such models, employing theoretical analysis, systematization, and generalization of their usage in autonomic devices. Our research reveals significant potential for this technology, identifying key parameters influencing model efficiency on resource-constrained devices and proposing implementation recommendations. Furthermore, we highlight contemporary approaches to model encryption, emphasizing the need for a comprehensive security approach combining hardware and software solutions. The study concludes that advancements in this field will contribute to more efficient autonomic systems across various applications, including computer vision and robotics. By addressing technical challenges and security considerations, this research paves the way for robust autonomic visual processing systems, potentially revolutionizing industries reliant on intelligent image analysis. The relevance of this study lies in its practical insights into implementing deep learning models for semantic segmentation in resource-constrained environments, contributing to the evolving field of autonomic intelligent systems.

Keywords

Semantic segmentation, deep learning, computer vision, model optimization, autonomic devices

1. Introduction

Computer vision is a rapidly evolving field of artificial intelligence (AI) focused on creating technologies that replicate human visual perception. It combines elements from computer science, optics, mechanics, and neuroscience to develop algorithms for analyzing and interpreting visual data [1]. A key area within this field is semantic image segmentation, which classifies each pixel in an image according to specific categories, providing deeper insights as compared to simple object detection [2]. Semantic segmentation models generate a segmentation map where each pixel is color-coded based on its semantic class, allowing for the creation of masks that highlight distinct areas of an image [3]. This technique is crucial for applications in satellite imagery, UAV data analysis, robotics, medicine, and automotive industries. Current approaches to semantic segmentation include traditional methods that rely on feature extraction and pixel classification, which can be time-consuming and dependent on domain expertise [4]. To overcome these limitations, deep learning techniques using neural networks have been developed, significantly enhancing accuracy and adaptability. Prominent architectures for semantic segmentation include Fully Convolutional Networks (FCN), U-Net, DeepLab, and PSPNet. However, integrating these neural networks into autonomic devices with limited or no Internet access represents the challenging task. This work aims to explore the

Information Systems and Technologies (IST-2024), November 26-28, 2024, Kharkiv, Ukraine
* Corresponding author.
† These authors contributed equally.

mykhailo.trusov@karazin.ua (M. Trusov); dmytro.uzlov@karazin.ua (D. Uzlov)
0009-0001-4390-5307 (M. Trusov); 0000-00Q3-3308-424X (D. Uzlov)

application of deep learning models in such devices and analyze effective integration methods without constant connectivity.

2. Utilization of deep learning models in autonomic devices for semantic image segmentation

Recent research in semantic segmentation demonstrates that deep learning models, particularly Fully Convolutional Networks (FCNs) and other neural networks, can be effectively deployed on autonomic devices without Internet access. This is achieved through the on-device inference, where trained models are deployed directly on the device for local predictions [5]. This approach reduces latency, enhances privacy, and allows operation in areas with limited or no Internet connectivity.

Edge computing is another concept that places computational resources closer to data sources, enabling local processing. For instance, smart cameras can analyze video streams directly on the device using pre-trained models, reducing dependence on constant internet connections and facilitating real-time data processing [6].

Specialized embedded systems like NVIDIA Jetson, Google Coral, and Intel Movidius are widely used for efficient neural network execution on autonomic devices. These platforms are optimized for neural network operations, significantly accelerating computations compared to standard processors while maintaining low power consumption [7].

Various optimization techniques are employed to ensure efficient neural network operation on resource-constrained devices. These include model compression methods such as quantization, pruning, and knowledge distillation, which reduce computational complexity while maintaining accuracy. Hardware acceleration using GPUs, TPUs, or AI-specific chips is another crucial optimization approach, leveraging specialized computing devices optimized for neural network operations [8].

Combining model compression and hardware acceleration methods achieves a synergistic effect, significantly enhancing neural network efficiency on embedded systems. This comprehensive optimization approach opens up broad prospects for implementing complex AI algorithms in mobile and embedded devices, expanding their application scope and increasing the autonomy and intelligence of edge devices.

Offline data processing is another aspect of modern embedded systems that allows devices to perform complex tasks without constant internet connectivity. This approach is particularly useful in situations where devices collect and process data in batches, performing segmentation tasks autonomicly and only occasionally connecting to the internet for updates or additional data transmission.

3. Key determinants of trained model storage requirements

The memory requirements for deploying trained models on devices are influenced by several key factors. Model complexity plays a significant role, with more intricate architectures generally demanding more storage space. The specific neural network architecture chosen also impacts memory usage, as some designs are inherently more efficient than others.

Optimization techniques are crucial in managing memory consumption. Quantization, which reduces the precision of model weights, can substantially decrease memory requirements, often compressing models by up to 75%. Pruning, another effective method, removes less critical weights or neurons, further reducing model size while maintaining performance. Knowledge distillation, where a smaller model learns to emulate a larger one, can result in compact models with comparable effectiveness.

The choice of model is also pivotal. Lightweight architectures like MobileNetV2, designed specifically for mobile applications, typically occupy only 10-15 MB after optimization [9]. In contrast, more complex models such as full-sized Fully Convolutional Networks (FCNs) or advanced architectures like ResNet can require hundreds of megabytes or even several gigabytes [10].

Additional considerations include the memory needed for runtime environments (e.g., TensorFlow Lite, ONNX Runtime) and auxiliary data such as class labels or configuration files. These components contribute to the overall memory footprint and must be accounted for in deployment planning.

Broadly speaking, memory requirements can be categorized as follows: 10-50 MB for simple tasks and lightweight models, 50-200 MB for moderately complex models with some optimization, and 200 MB to several GB for sophisticated, high-performance models.

By leveraging efficient architectures and applying appropriate optimization techniques, it is possible to deploy neural networks on devices with limited memory resources. This capability enables many automated devices to operate autonomicly, even in scenarios with limited or no Internet connectivity.

4. Implementation of trained models in custom software products

To use trained models in custom applications, specialized libraries or frameworks are necessary [11]. Here are some popular options for various platforms:

- TensorFlow Lite which is designed for mobile and embedded devices, supporting Python, C++, Java, and Swift. It offers model conversion, inference APIs, and hardware acceleration support.
- ONNX Runtime which represents a cross-platform environment for ONNX format models, supporting multiple languages and optimized for various hardware accelerators.
- PyTorch Mobile which is useful for deploying PyTorch models on mobile devices, supporting Python, Java, and C++.
- Core ML which stands for the framework for iOS and macOS applications, primarily used with Swift and Objective-C.
- NVIDIA TensorRT which represents a toolkit for high-performance deep learning inference on NVIDIA GPUs, supporting Python and C++.
- OpenCV which is characterized as an open-source computer vision and machine learning library, supporting C++, Python, Java, and MATLAB.

The choice of library or framework depends on project specifics, target platform, and performance requirements. Here's a brief example of integrating with TensorFlow Lite in Python:

```
import tensorflow as tf
import numpy as np

# Load the TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Prepare input data
input_data = np.array(your_input_data, dtype=np.float32)

# Run inference
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()

# Get output data
output_data = interpreter.get_tensor(output_details[0]['index'])
print(output_data)
```

5. Security of deep learning models for autonomic devices

In the context of using deep learning models for semantic image segmentation on autonomic devices, ensuring the security of trained models is crucial. Modern approaches to security in this domain focus on hardware-based solutions and comprehensive encryption strategies.

One key component of this security framework is the use of Hardware Security Modules (HSMs) [12]. These specialized physical devices play a vital role in protecting deep learning models by performing cryptographic processing and providing secure key storage. By integrating HSMs into autonomic devices for semantic image segmentation, encryption and decryption operations can be conducted in a secure environment, preventing unauthorized access to trained models and protecting intellectual property.

Complementing HSMs, Trusted Platform Modules (TPMs) offer another layer of hardware-based security [13]. These secure cryptoprocessors store cryptographic keys and perform cryptographic operations, creating a reliable hardware foundation for protection. TPMs ensure that models can only be decrypted through authorized operations, significantly enhancing security even if physical access to the device is obtained.

While hardware solutions provide a strong foundation, a truly comprehensive approach to protecting deep learning models combines robust encryption with stringent access control. Models should be encrypted using advanced algorithms like AES-256, with encryption keys stored securely in HSMs or TPMs. This ensures data integrity even if models are extracted from the device. Building upon this encryption, access control should implement multi-factor authentication (MFA) and a multi-layered authentication and authorization system, restricting access to verified users only [14].

6. Conclusions

Overall, the present study highlights the potential of deep learning models for semantic image segmentation on autonomic devices, enabling visual analysis without constant external computing resources. Key to implementation is the optimization of model size and efficiency through techniques like quantization, pruning, and knowledge distillation, making them suitable for resource-constrained devices. Deployment requires specialized frameworks such as TensorFlow Lite or PyTorch Mobile, with optimization for specific hardware crucial. Security is paramount, necessitating a comprehensive approach combining hardware and software solutions. Future developments focus on improving neural network architectures, developing new optimization methods, and creating specialized hardware accelerators. While challenges persist, the field promises significant advancements in intelligent visual processing systems, opening new possibilities for autonomic devices across various applications.

References

[1] R. Szeliski, Computer vision: algorithms and applications. Springer Nature, 2022. 925 p. doi: 10.1007/978-3-030-34372-9.

[2] Hao S., Zhou Y., Guo Y. A brief survey on semantic segmentation with deep learning. Neurocomputing 406 (2020) 302-321. doi:10.1016/j.neucom.2019.11.118.

[3] Guo Y., Liu Y., Georgiou T., Lew M. A review of semantic segmentation using deep neural networks. Int. J. Multimed. Info Retr. 7 (2018) 87-93. doi: 10.1007/s13735-017-0141-z.

[4] O'Mahony N., Campbell S., Carvalho A., et al. Deep learning vs. traditional computer vision, Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Vol. 11, Springer International Publishing, 2020. P. 128-144. doi:10.1007/978-3-030-17795-9.

[5] Mairittha N., Mairittha T., Inoue S. On-device deep learning inference for efficient activity data collection. Sensors (Basel) 19 (2019) 3434. doi: 10.3390/s19153434.

[6] Cui T. Review of deep learning and mobile edge computing in autonomous driving. Вісник Львівської політехніки 12 (2022) 208-218. doi:10.23939.

[7] Zhang Z., Li J. A review of artificial intelligence in embedded systems. Micromachines 14 (2023) 897. doi:10.3390/mi14050897.

[8] Merone M., Graziosi A., Lapadula V., Petrosino L., d'Angelis O., Vollero L. A practical approach to the analysis and optimization of neural networks on embedded systems. Sensors 22 (2022) 7807. doi:10.3390/s22207807.

[9] Dong K., Zhou C., Rian Y., Li Y. MobileNetV2 model for image classification. 2nd International Conference on Information Technology and Computer Application (ITCA). IEEE, 2020. P. 476-480. doi:10.1109/ITCA52113.2020.00106.

[10] Berthelier A., Chateau T., Duffner S., et al. Deep model compression and architecture optimization for embedded systems: a survey. J. Signal Processing Systems 93 (2021) 863-878. doi:10.1007/s11265-020-01596-1.

[11] Hadidi R., Cao J., Xie Y., et al. Characterizing the deployment of deep neural networks on commercial edge devices. IEEE International Symposium on Workload Characterization (IISWC), IEEE. 2019. P. 35-48. doi:10.1109/IISWC47752.2019.9041955.

[12] Mavrovouniotis S. Ganley M. Hardware security modules. Secure Smart Embedded Devices, Platforms and Applications. New York, NY: Springer New York, 2013. P. 383-405. doi:10.1007/978-1-4614-7915-4_17.

[13] Ezirim K., Khoo W., Koumantaris G., et al. Trusted platform module – a survey. The Graduate Center of The City University of New York, 11. 2012.

[14] Köylü T.Ç., Wedig Reinbrecht C.R., Gebregiorgis A., et al. A survey on machine learning in hardware security. ACM J. Emerg. Techn. Comput. Syst. 19 (2023) 1-37. doi:10.1145/3589506.