applied sciences

MDPI

*Article*

# Real-Time Semantic Image Segmentation with Deep Learning for Autonomous Driving: A Survey

**Ilias Papadeas [1,*], Lazaros Tsochatzidis [1], Angelos Amanatiadis [2] and Ioannis Pratikakis [1]**

[1] Visual Computing Group, Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece; ltsochat@ee.duth.gr (L.T.); ipratika@ee.duth.gr (I.P.)
[2] Department of Production and Management Engineering, Democritus University of Thrace, 67100 Xanthi, Greece; aamanat@pme.duth.gr
* Correspondence: ipapadea@ee.duth.gr

**Abstract:** Semantic image segmentation for autonomous driving is a challenging task due to its requirement for both effectiveness and efficiency. Recent developments in deep learning have demonstrated important performance boosting in terms of accuracy. In this paper, we present a comprehensive overview of the state-of-the-art semantic image segmentation methods using deep-learning techniques aiming to operate in real time so that can efficiently support an autonomous driving scenario. To this end, the presented overview puts a particular emphasis on the presentation of all those approaches which permit inference time reduction, while an analysis of the existing methods is addressed by taking into account their end-to-end functionality, as well as a comparative study that relies upon a consistent evaluation framework. Finally, a fruitful discussion is presented that provides key insights for the current trend and future research directions in real-time semantic image segmentation with deep learning for autonomous driving.

**Keywords:** semantic image segmentation; real time; deep learning; autonomous driving



**Citation:** Papadeas, I.; Tsochatzidis, L.; Amanatiadis, A.; Pratikakis, I. Real-Time Semantic Image Segmentation with Deep Learning for Autonomous Driving: A Survey. *Appl. Sci.* **2021**, *11*, 8802. https://doi.org/10.3390/app11198802

Academic Editor: Francesco Bianconi

Received: 2 August 2021
Accepted: 11 September 2021
Published: 22 September 2021



## 1. Introduction

Semantic segmentation is the task of assigning each pixel of an image to a corresponding class label from a predefined set of categories [1]. Although it can be considered to be a pixel-level classification problem in the pixels of an image, it is a much more complex procedure, compared to the standard classification which targets predicting the label of the entire image.

The enormous success of deep learning has made a huge impact in semantic segmentation methods, improving their performance in terms of accuracy. This promising progress has attracted the interest of many technological and research fields that require high-end computer vision capacities. Such an application is autonomous driving, in which self-driving vehicles must understand their surrounding environment, i.e., other cars, pedestrians, road lanes, traffic signs or traffic lights. Semantic segmentation based on deep learning is a key choice for accomplishing this goal, due to the phenomenal accuracy of deep neural networks in detection and multi-class recognition tasks.

Nevertheless, in applications, such as autonomous driving, that require low-latency operations, the computational cost of these methods is still quite limiting. Autonomous driving belongs to these applications because of the crucial need to take decisions in precise intervals. It is, therefore, necessary to improve the design of segmentation models towards achieving efficient architectures that will be able to perform in real time with the appropriate precision. To this end, in this paper, we review the best semantic segmentation architectures in terms of speed and accuracy. For the best and most complete evaluation of the examined architectures, all the models are compared based on their performance in a consistent evaluation framework.

The contribution of this paper is three-folded: (1) it presents in a consistent way an exhaustive list of the most efficient methods for designing real-time semantic segmentation models aiming at achieving both high accuracy and low latency. (2) it provides a comparative study of the state-of-the-art real-time semantic segmentation models based on their accuracy and inference speed. (3) it presents a fruitful discussion on current issues and improvements that results in key insights in the field of real-time semantic image segmentation.

This paper is organized as follows: in Section 1, an introduction for our survey of real-time semantic image segmentation with deep learning for autonomous driving is presented, while in Section 2, the basic approaches that reduce the inference time of real-time models are presented. In Section 3, we present an exhaustive list of the state-of-the-art models. In Section 4, the most popular datasets and metrics used in evaluation are summarized. Section 5 presents a discussion including our findings and key insights. In Section 6, promising future research issues are presented aiming to improve the performance of real-time semantic segmentation, and conclusions are drawn in Section 7.

## 2. Approaches for Inference Time Reduction

Although semantic segmentation models based on deep learning have achieved great accuracy in recent years, the need for efficiency that requires less inference time is still vital, especially for applications such as autonomous driving. In the following, we present all existing approaches that can be used in the deep neural network architecture design aiming to achieve a reduced response time in semantic image segmentation models.

### *2.1. Convolution Factorization—Depthwise Separable Convolutions*

It is known that for most deep-learning models the convolutional layers are vital structural components. Therefore, transforming the convolutions that are performed in the layers of the network into more computationally efficient actions is an excellent way to improve the model's performance in terms of speed. A popular design choice for improving convolutions is the use of depthwise separable convolutions, which is a type of factorized/decomposed convolutions [2]. Standard convolution performs channel-wise and spatial-wise computation in a single step. On the contrary, depthwise separable convolution breaks the computation process into two steps. In the first step, a single convolutional filter per each input channel is applied (depthwise convolution), while in the second step, a linear combination of the output of the depthwise convolution is considered by means of a pointwise convolution [3]. These two different procedures are shown in Figure 1. It is important to compare the computational burden of these two tasks. More specifically, for $N$ filters of size $D \times D$, the ratio of computational complexity between depthwise separable convolutions and standard convolutions equals to $Ratio = 1/N + 1/D^2$. For example, given 100 filters of size $256 \times 256$, the ratio of complexity is 0.01, which means that a series of depthwise separable convolution layers execute 100 times less multiplications than a corresponding block of standard convolutional layers.

This approach was first used in Xception [4], where the designers replaced Inception module [5] with depthwise separable convolutions. Likewise, MobileNets [6] comprise depthwise separable convolutions to improve efficiency. Overall, the efficiency by design offered by this approach is a strong argument for being used in any new efficiency-oriented network implementations.

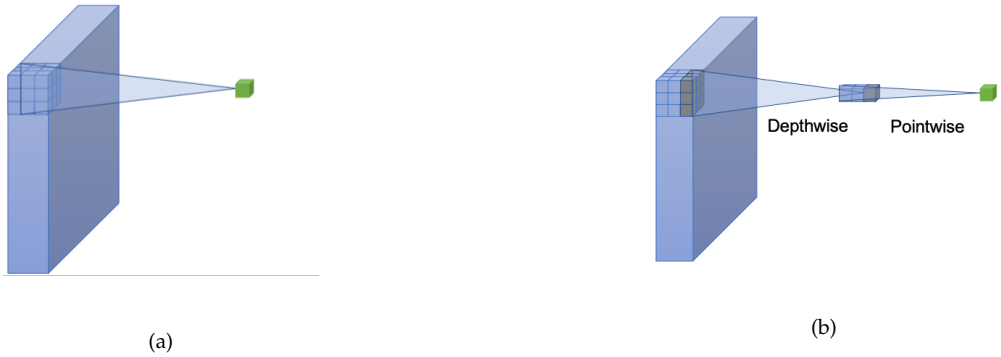



**Figure 1.** (**a**) Standard convolution. (**b**) Depthwise separable convolution (adapted from [3]).

### *2.2. Channel Shuffling*

Another way to reduce computational cost significantly while preserving accuracy is channel shuffling. This approach was introduced in ShuffleNet [7]. Although in standard group convolution every input channel is associated with only a single output channel, in the case of channel shuffling, a group convolution acquires data from different input groups. In particular, every input channel will correlate with every output channel, as shown in Figure 2. Specifically, for the feature map generated from the previous group layer, we can first divide the channels in each group into several subgroups and then feed each group in the next layer with different subgroups.

This can be efficiently implemented by a channel shuffle operation that is described as follows: suppose a convolutional layer with $g$ groups whose output has $g \times n$ channels; we first reshape the output channel dimension into $(g, n)$, transposing and then flattening it back as the input of next layer. Channel shuffle is also differentiable, which means it can be embedded into network structures for end-to-end training. Overall, this approach comprises appealing characteristics, but one should be careful with the selection of the parameter $g$ which defines the number of groups. For optimal results, experimentation is required to achieve the best $g$ value.

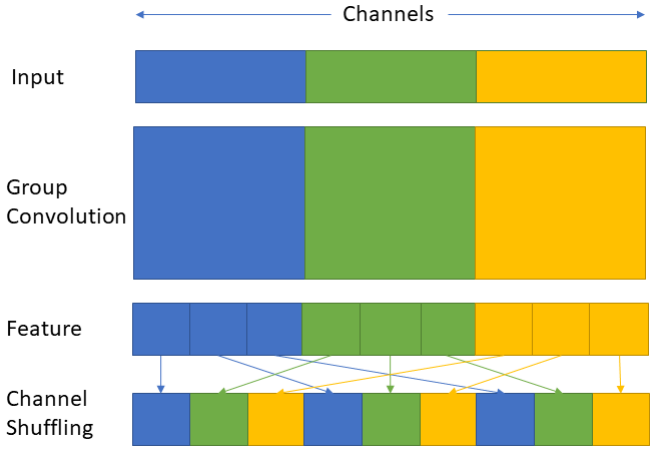



**Figure 2.** Channel shuffling.

### *2.3. Early Downsampling*

In ENet [8] model architecture, it is stated that since processing large input frames is very expensive, a good way to solve this is to downsample these frames in the early stages of the network, resulting in the use of only a small set of feature maps.

The primary role of the initial network layers should be feature extraction and the preprocessing of the input data for the following parts of the architecture, rather than contributing to the classification stage [8].

This approach has been employed by the ENet model architecture so that spatial information loss due to downsampling is prevented. ENet relied on the SegNet [9] approach of saving indices of elements chosen in max-pooling layers and using them to produce sparse upsampled maps in the decoder, as shown in Figure 3. This approach allows reducing memory requirements while recovering spatial information. Overall, since the proposed approach relies upon spatial downsampling, it is not recommended for applications where the initial image contains fine image details that have the potential to disappear after the corresponding max-pooling operation.

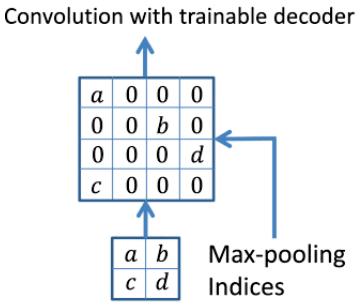



**Figure 3.** SegNet decoder uses the max-pooling indices to upsample (without learning) the feature maps (adapted from [9]).

### *2.4. The Use of Small Size Decoders*

The encoder–decoder network is one of the most standard architectures of semantic segmentation. As a result, it is crucial to optimize the performance of this architecture as it influences plethora of models. In [8], it is suggested that the architecture of an encoder–decoder model can be simplified by reducing the decoder's size, in order to save computational cost. In particular, they introduced an architecture in which the encoder is larger than the decoder [8], following a different approach from the symmetric encoder–decoder architecture [9]. This novelty is based on the idea that encoder should process input data with smaller resolution of the input image. On the contrary, the one and only decoder's role is to upsample the output of the encoder by perfecting its details. Thus, reducing the decoder's size results in computational cost savings. Overall, this approach is appealing since in most of the times the reduction in the decoder's size does not affect the effectiveness.

### *2.5. Efficient Reduction of the Feature Maps' Grid Size*

In CNNs, the reduction of the feature maps' grid size is achieved by the application of pooling operations. A problem that occurs is that the pooling operations can lead to representational bottlenecks, which can be avoided by expanding the activation dimension of the network filters. However, this process increases the computational cost. To remedy this, Szegedy et al. [10] suggested a pooling operation with a convolution of stride 2 performed in parallel, followed by the concatenation of the resulting filter banks. This technique of reducing the grid size of the feature maps (shown in Figure 4) is proven in the works of [8,10] that can achieve a significant improvement in inference time. Overall, several approaches exist that result in reducing the features map size which has been proven to not only achieve efficiency but to also achieve state-of-the-art effectiveness [11].

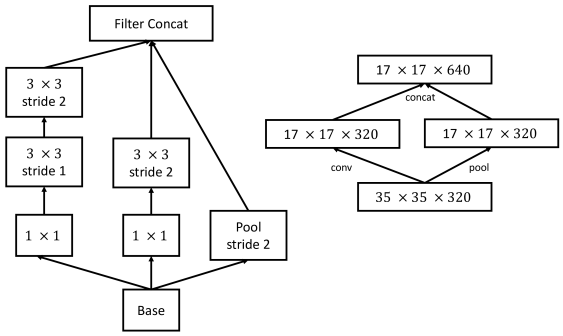



**Figure 4.** Efficient reduction of grid size of the feature maps (adapted from [10]).

### *2.6. Increasing Network Depth While Decreasing Kernel Size*

Research in [12] suggested that the use of very small ($3 \times 3$) convolutional filters shows improvement on the standard configurations of CNNs. Smaller convolutional filters can permit an increase in the depth of the network by adding more convolutional layers and at the same time reducing the number of parameters of the network. This technique reduces computational cost, and at the same time, increases the accuracy of the network [13].

### *2.7. Two-Branch Networks*

The trade-off between accuracy and inference time, has been addressed using two-branch networks: the one branch captures spatial details and generate high-resolution feature representation, while the other branch obtains high-level semantic context. Two-branch networks manage to achieve a beneficial balance between speed and accuracy because one of the two pathways is used to be a lightweight encoder of sufficient depth and the other pathway is a shallow, still wide branch consisting of a few convolutions [14–16]. At the same time, unlike the encoder–decoder architecture, two-branch networks preserve partial information that is lost after downsampling operations [16]. A standard two-branch network of [15] is shown in Figure 5. Overall, the proposed approach relies upon a lightweight architecture which achieves efficiency that is coupled with the possibility of both low- and high-level features representation learning that results in improved effectiveness.

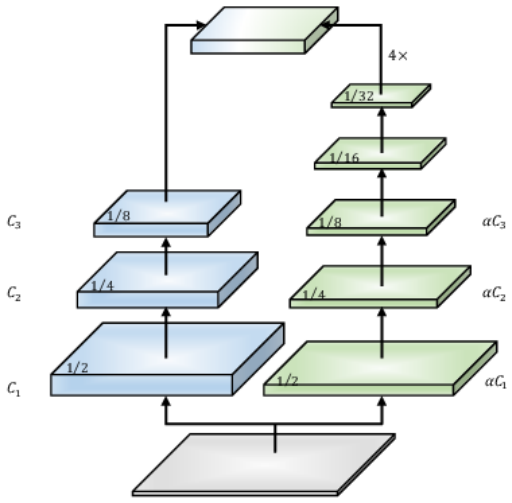



**Figure 5.** Example of two-branch networks (adapted from [15]).

### 2.8. Block-Based Processing with Convolutional Neural Networks

A novel and innovative method to speed up inference time is block-based processing, where as in [17] an image is split into blocks and adjusts the resolution of each block by downsampling the less important. This reduction of the processing resolution results in the reduction of the computational burden and the memory consumption.

### 2.9. Pruning

Pruning is a method suitable for producing models that perform more accurately, while being faster with less memory cost. The visual information is highly spatially redundant, and thus can be compressed into a more efficient representation. Pruning is distinguished in two categories: weight pruning and filter (channel) pruning. Both weight pruning and filter pruning are shown in Figure 6.

In weight pruning, individual parameters (connections), and hence the weights, are being removed, generating a sparse model that preserves the high-dimensional features of the original network. The work of [18] suggested weight pruning with a three-step method in which first the network is being trained to learn important connections. In the second step, unessential connections are being pruned. Lastly, the network is trained again to fine-tune the weights of the remaining connections. This way, network pruning can reduce the number of connections by $9x$ to $13x$ without reducing its effectiveness [18,19].

Filter (or channel) pruning is also a very successful type of method which leads to successful results. By removing filters that have negligible effect on the output accuracy alongside with their corresponding feature maps, the computation costs are significantly reduced [20]. He et al. [21] proposed an efficient method based on channel pruning which improves inference time, while preserving the accuracy of the network. Overall, this approach puts emphasis on the efficiency with the cost of losing information used in the network at the spatial and functional level resulting in reduced effectiveness.

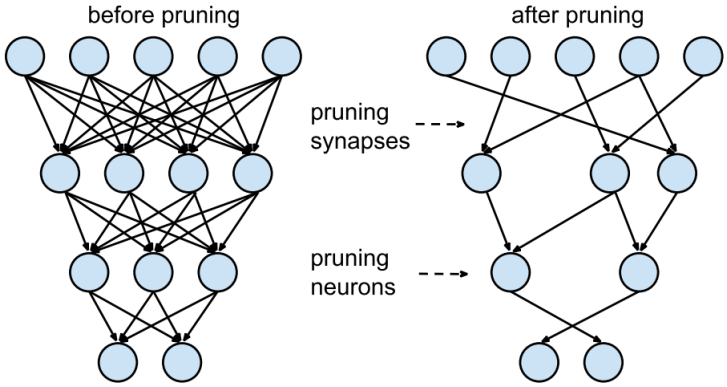



**Figure 6.** Synapses and neurons before and after pruning (adapted from [18]).

### 2.10. Quantization

An additional method to ameliorate the efficiency of semantic segmentation is to use quantization to reduce the number of bits needed for the weight representation of the network. A standard way to represent each weight is to employ 32 bits. However, 32-bit operations are slow and have large memory requirements [22]. Han et al. [19] proposed a quantization approach that reduces the number of bits that represent each connection from 32 to 5 and at the same time restrict the number of effective weights by sharing the same weights between multiple connections, and then fine-tune them.

## 3. State-of-the-Art Deep-Learning Models

In this Section, we present the current state-of-the-art models of real-time semantic segmentation, based on their performance in terms of accuracy and inference time for

"Cityscapes" and "CamVid" datasets. In our presentation, we follow a grouping of the models based on the approaches that have been used to achieve efficiency.

In the case of using a two-branch network, several models appear in the state-of-the-art. SS [23] belongs to the multi-branch networks as it uses two branches, one for receiving the input image and the other for receiving a half-resolution version of that image. Moreover, SS introduces spatial sparsity in this type of architecture, managing to reduce the computational cost by a factor of 25, alongside with using in-column and cross-column connections and removing residual units. Its architecture is shown in Figure 7.

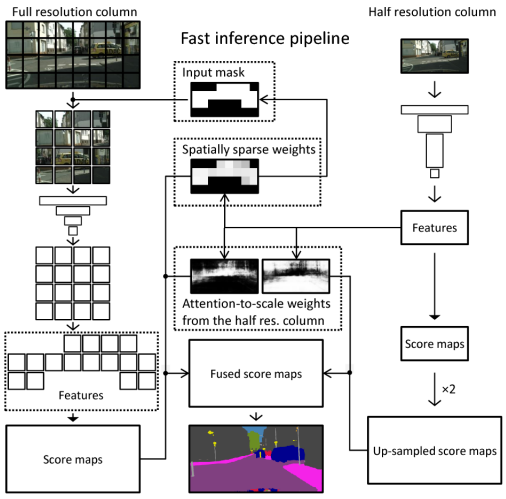



**Figure 7.** The inference pipeline of SS (adapted from [23]).

Another model that constitutes of two branches is ContextNet [24]. Taking a deeper look at Figure 8, the first branch achieves cost efficient and accurate segmentation at low resolution, while the second combines a sub-network at high resolution to provide detailed segmentation results. ContextNet also uses depthwise separable convolutions for speeding up inference time and bottleneck residual blocks. Very important factor for its performance is network compression and the pyramid representation to segment in real time with low memory cost. Apart from these approaches, ContextNet uses also pruning to decrease the parameters of the network.

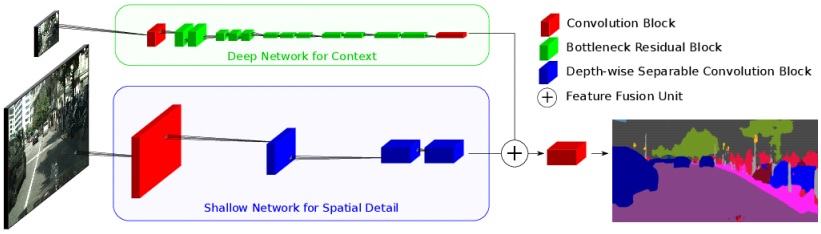



**Figure 8.** The inference pipeline of ContextNet (adapted from [24]).

Fast-SCNN [25] proposes a "learning to downsample" module for computing low-level features for multiple resolution branches in parallel, as shown in Figure 9. Fast-SCNN also features a global feature extractor for capturing the global context for semantic segmentation. Finally, Fast-SCNN uses depthwise separable convolutions and residual bottleneck blocks [26] to increase speed and reduce the number of parameters and the computational cost.

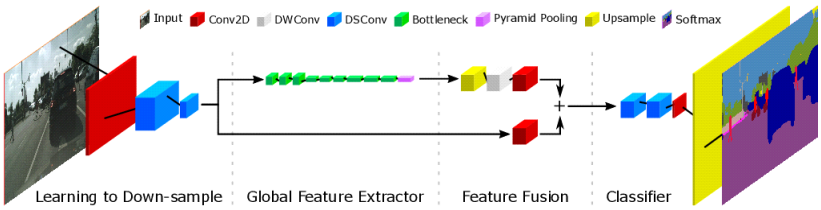



**Figure 9.** The framework of Fast-SCNN (adapted from [25]).

BiSeNet (Bilateral Segmentation Network) [14] introduces a feature fusion module used for the efficient combination of the features and an attention refinement module to filter the features of each stage. In that fashion, it improves precision while upsampling operations are being avoided and thus the computational cost is kept low. BiSeNet V2 [15] is the evolution of BiSeNet, which presents good trade-off between speed and accuracy. It features a Guided Aggregation Layer to fuse the features extracted from the Detail and the Semantic branch, and proposes booster training strategy. Creators use fast-downsampling in the Semantic Branch to advance the level of the feature representation and increase the receptive field rapidly. The structure of both networks are presented in Figures 10 and 11, respectively.

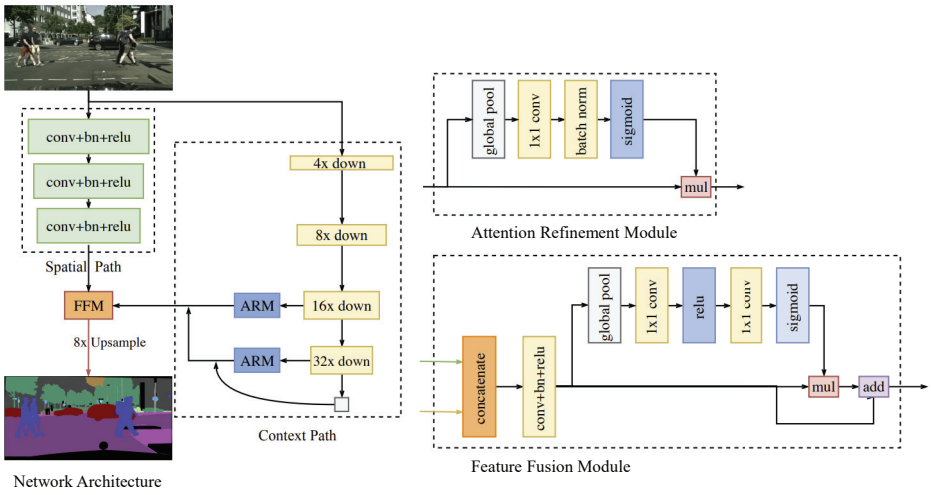



**Figure 10.** An overview of BiSeNet Network Architecture including the Attention Refinement and the Feature Fusion Module (adapted from [14]).

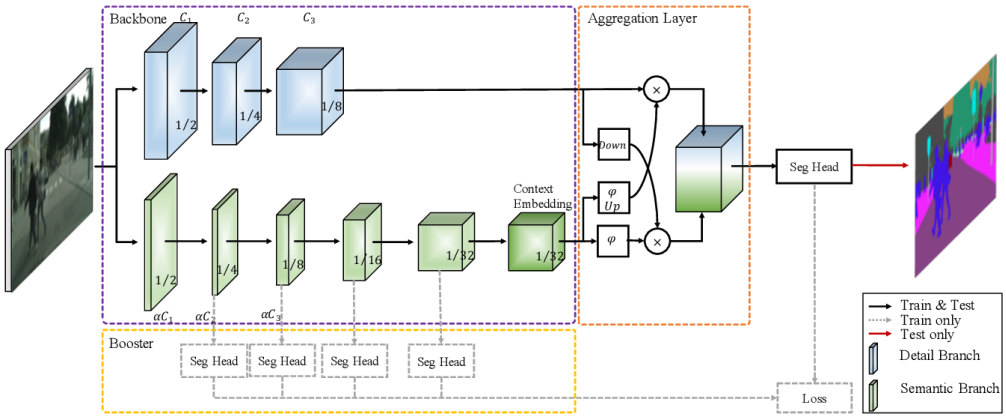



**Figure 11.** An overview of BiSeNet V2 (adapted from [15]).

FasterSeg [27] employs neural architecture search [28,29] and proposes a decoupled and fine-grained latency regularization to balance the trade-off between high inference speed and low accuracy. The scope of neural architecture search is the design of deep-learning architectures in an automatic fashion, thus reducing the involvement of the human

factor [30]. Another contribution is the use of knowledge distillation [31] and specifically, the introduction of a co-searching teacher-student framework that improves accuracy. Knowledge distillation aims to transfer the features learned from a large and complex network (teacher network) to a smaller and lighter network (student network). In the proposed co-searching teacher-student framework the teacher and student networks share the same weights, working as a supernet, thus not creating extra burden in terms of memory usage and size. The general structure is shown in Figure 12.

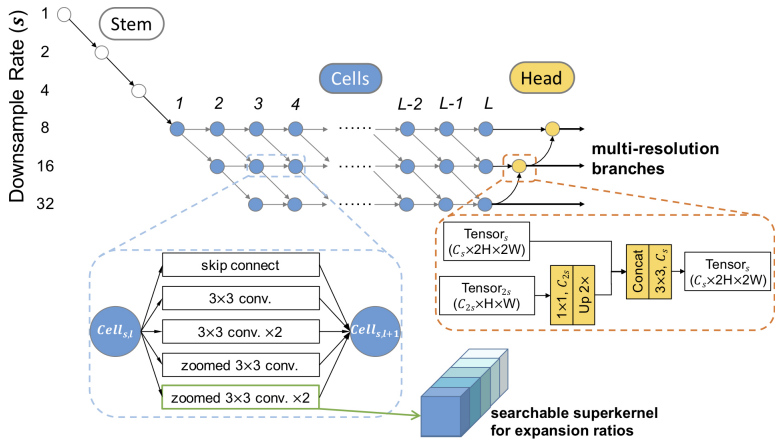



**Figure 12.** Network architecture of FasterSeg (adapted from [27]).

ESNet [32] follows a symmetric encoder–decoder architecture, as shown in Figure 13. It involves a parallel factorized convolution unit module with multiple branch parallel convolutions, multi-branch dilated convolution and pointwise convolutions. The symmetry of ESNet's architecture reduces network's complexity and as a result leads to a reduction of the inference time. The parallel factorized convolution unit module with multiple branch parallel convolutions manage to learn imparallel feature representations in a powerful manner, without increasing the computational complexity.

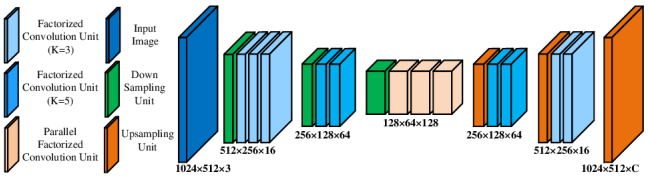



**Figure 13.** Overall symmetric architecture of ESNet (adapted from [32]).

ShelfNet18 [33] structure is presented in Figure 14. More specifically, it is composed of multiple encoder–decoder branches, uses shared-weights and a residual block. To decrease inference time, ShelfNet18 proposes channel reduction to efficiently reduce to computation burden. The use of different encoder–decoder branches ameliorates the computational process, increasing the segmentation accuracy. Weights are being shared between convolutional layers of the same residual block, in order to decrease the number of network parameters without decreasing accuracy.

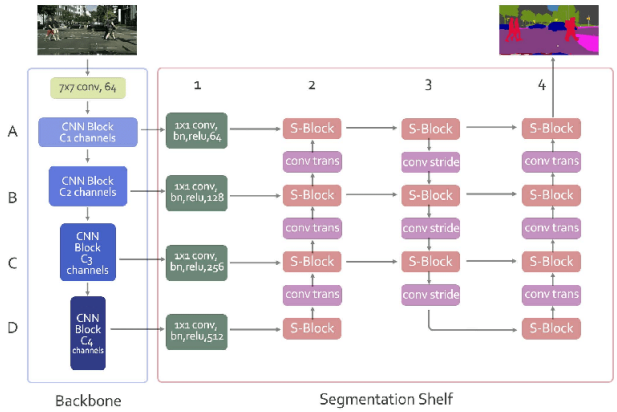



**Figure 14.** Network architecture of ShelfNet (adapted from [33]).

ICNet [34] proposes a framework for saving operations in multiple resolutions and features a cascade feature fusion unit. Its architecture is shown in Figure 15. It processes, simultaneously, semantic information from low resolution branches, along with details from high-resolution images in an efficiently manner.

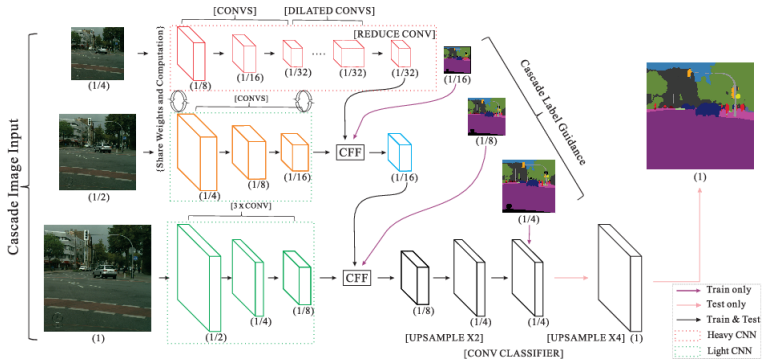



**Figure 15.** Network architecture of ICNet (adapted from [34]).

In the case of using factorized convolutions to reduce latency the efficiency is achieved by modifying the way convolutions work in the network architecture to boost speed [2,35]. In the literature, several models employ such an approach. First, ESPNet [36] introduces a convolutional module, called efficient spatial pyramid, which functions efficiently in terms of computational cost, memory usage and power. ESPNet depends on a principle of convolution factorization. More analytically, convolutions are resolved into two steps: (1) pointwise convolutions and (2) spatial pyramid of dilated convolutions. ESPNet's network structure, shown in Figure 16, is fast, small, capable of handling low power, and low latency while preserving semantic segmentation accuracy.

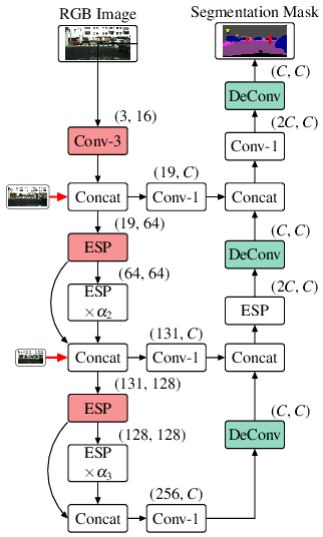



**Figure 16.** Network architecture of ESPNet (adapted from [36]).

ESPNetv2 [37] based on ESPNet, outperforms the latter by 4–5% and has 2–4× fewer FLOPs on the PASCAL VOC and the "Cityscapes" dataset. ESPNetv2 modifies efficient spatial pyramid module and introduces an Extremely Efficient Spatial Pyramid unit by replacing pointwise convolutions with group pointwise convolutions, computationally expensive 33 dilated convolutions with depthwise dilated separable convolution. Lastly, it fuses the feature maps employing a computationally efficient hierarchical feature fusion method. According to [37], pruning and quantization are complementary methods for ESPNetv2. A structural unit of its architecture is presented in Figure 17.

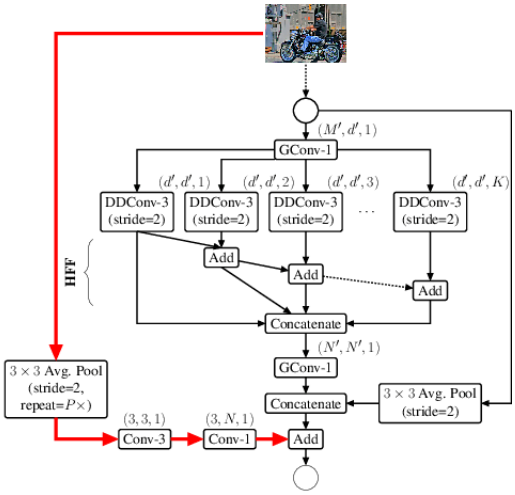



**Figure 17.** Strided Extremely Efficient Spatial Pyramid unit with shortcut connection to an input image (highlighted in red) for downsampling (adapted from [37]).

ESSGG [38] improves runtime of ERFNet by over a factor of 5*X*, by replacing its modules with more efficient ones, such as the aforementioned depthwise separable convolutions, grouped convolutions and channel shuffling. In this manner the inference time is reduced efficiently. Another contribution of this work is the training method, called gradual grouping where dense convolutions are transformed to grouped convolutions optimizing the function of gradient descent. Furthermore, it is important to refer that ESSGG uses a small decoder's size. Finally, pruning and quantization are proposed as secondary options used in the later stages of the network to improve efficiency. The detailed network's architecture is shown in Table 1.

**Table 1.** Network architecture of ESSGG [38].

| Layer | Type | Output Channel | Output Resolution |
|---|---|---|---|
| 1 | Downsampling block | 16 | 512 × 256 |
| 2 | Downsampling block | 64 | 256 × 128 |
| 3–5 | 3 × Non-bt-1D | 128 | 128 × 64 |
| 5–7 | 2 × Conv-module | 64 | 256 × 128 |
| 8 | Downsampling block | 128 | 128 × 64 |
| 9 | Non-bt-1d (dilated 2) | 128 | 128 × 64 |
| 10 | Non-bt-1d (dilated 4) | 128 | 128 × 64 |
| 11 | Non-bt-1d (dilated 8) | 128 | 128 × 64 |
| 12 | Non-bt-1d (dilated 16) | 128 | 128 × 64 |
| 13 | Conv-module (dilated 2) | 128 | 128 × 64 |
| 14 | Conv-module (dilated 4) | 128 | 128 × 64 |
| 15 | Conv-module (dilated 8) | 128 | 128 × 64 |
| 16 | Conv-module (dilated 16) | 128 | 128 × 64 |
| 17 | Deconvolution (upsampling) | 64 | 256 × 128 |
| 18–19 | 2 × Non-bt-1D | 64 | 256 × 128 |
| 20 | Deconvolution (upsampling) | 16 | 512 × 256 |
| 21–22 | 2 × Non-bt-1D | 16 | 5125 × 256 |
| 23 | Deconvolution (upsampling) | C | 1024 × 512 |

DABNet [39] introduces a Depthwise Asymmetric Bottleneck module, which extracts combined local and contextual information and reduces the number of parameters of the network. This specific module uses depthwise asymmetric convolution and dilated convolution. DABNet is build on the idea of creating a sufficient receptive field which aids to the dense use of the contextual information. DABNet's design, shown in Figure 18, achieves an efficient and accurate architecture with reduced number of parameters compared to other real-time semantic segmentation models. It is of high importance to note that DABNet uses convolution factorization by choosing depthwise separable convolutions to speed up and inference time. DABNet presents 70.1% mIoU on "Cityscapes" test set, involving solely 0.76 million parameters, and can run at 104 fps on 512 × 1024 high-resolution images.

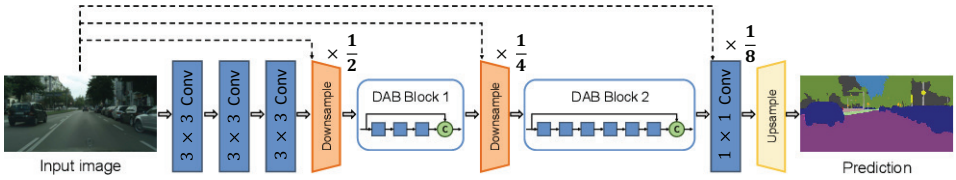



**Figure 18.** Architecture of Depthwise Asymmetric Bottleneck Network. C means concatenation, dashed lines indicate average pooling operation. (adapted from [39]).

DFANet [40] focuses on deep feature aggregation using several interconnected encoding paths to add high-level context into the encoded features. Its structure is shown in Figure 19.

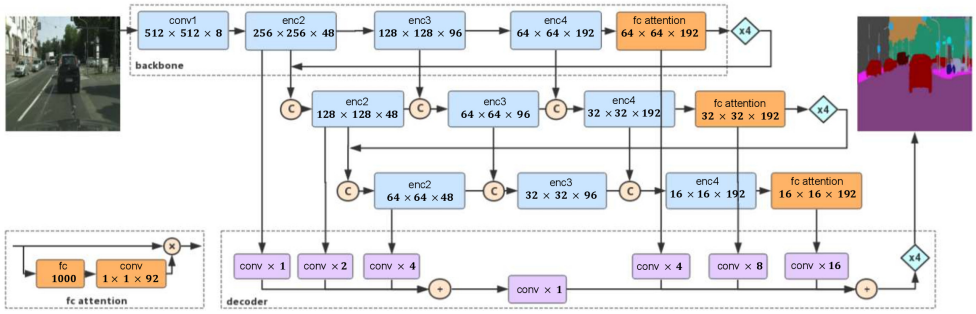



**Figure 19.** Overview of Deep Feature Aggregation Network: sub-network aggregation, sub-stage aggregation, and dual-path decoder for multi-level feature fusion. In the Figure, C denotes concatenation, xN is N upsampling operation (adapted from [40]).

ShuffleSeg [41] achieves satisfactory levels of accuracy by employing higher resolution feature maps. A deeper analysis of Figure 20, puts emphasis that in its encoder features grouped convolutions and channel shuffling which ameliorate performance. Shuffleseg was one of the first works that involved these two approaches for decreasing inference time.

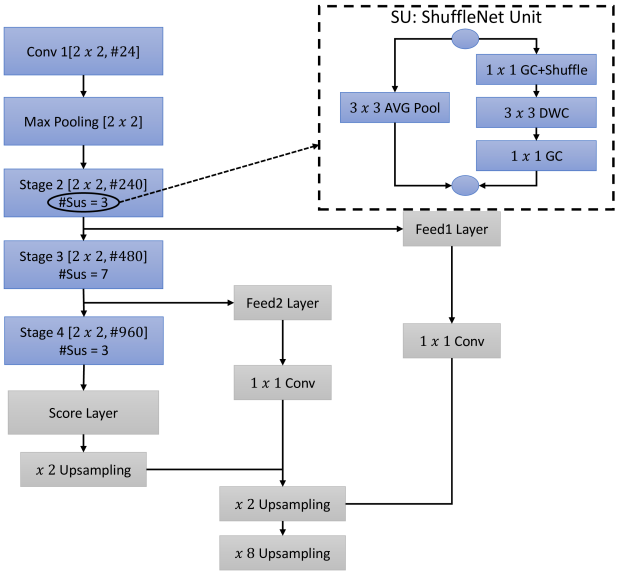



**Figure 20.** Network Architecture of ShuffleSeg (adapted from [41]).

HarDNet [42] introduces dynamic random-access memory traffic for feature map access and proposes a new metric, Convolutional Input/Output. It also features a Harmonic Dense Block and depthwise separable convolution. HarDNet also states that inference latency is highly correlated with the DRAM traffic. HarDNet achieves a high accuracy-over-Convolutional Input/Output and a great computational efficiency by improving the computational density (MACs (number of multiply-accumulate operations or floating-point operations) over Convolutional Input/Output). HarDNet's architecture is shown in Figure 21.

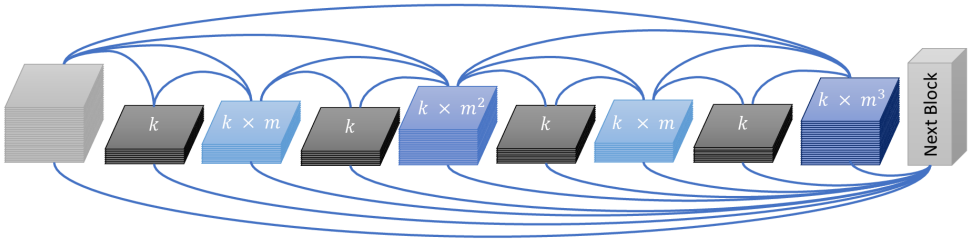



**Figure 21.** Network Architecture of HarDNet (adapted from [42]).

ERFNet [43] is based on the encoder–decoder architecture. Taking a deeper look at Figure 22, it constitutes of a layer that features residuals connections and factorized convolutions aiming to sustain efficiency while being accurate enough. To speed up the processing time, designers chose a small decoder size and deconvolutions to simplify memory and computational costs.

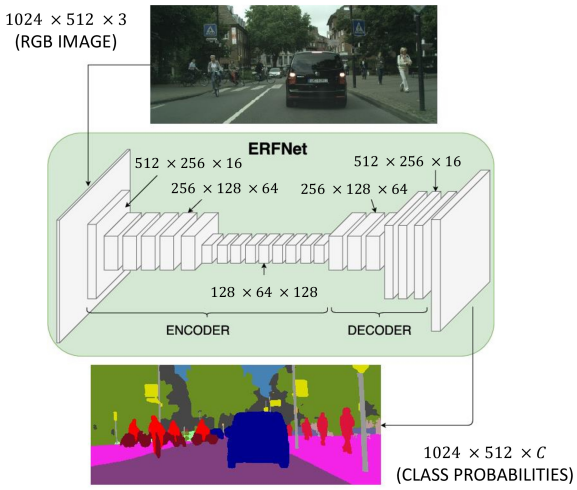



**Figure 22.** ERFNet Network Architecture (adapted from [43]).

In the case of channel shuffling, a considerable number of state-of-the-art models employ this approach to increase efficiency. LEDNet [44] is a novel lightweight network (shown in Figure 23) that focuses on reducing the amount of network parameters. It follows an asymmetric encoder–decoder architecture and uses channel shuffling for boosting inference speed. ESSG and ShuffleSeg also use channel shuffling for improving efficiency. Furthermore, LEDNet's decoder involves an attention pyramid network to enlarge the receptive fields, while alleviating the network from extra computational complexity. Moreover, the asymmetric encoder–decoder architecture indicates the efficiency-oriented approach of the small decoder's size to improve performance in terms of speed.

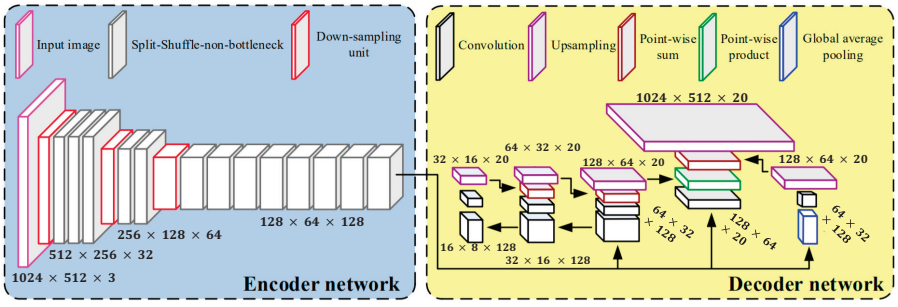



**Figure 23.** Overall asymmetric architecture of the LEDNet (adapted from [44]).

In the case of early downsampling, several semantic segmentation models use this approach to boost efficiency. EDANet [45] follows an asymmetric convolution structure. Asymmetric convolution decays a standard 2D convolution into two 1D convolutions. In this fashion, the parameters are reduced without sacrificing accuracy. EDANet uses early downsampling and dense connectivity to improve efficiency, while keeping low computational cost. It is important to note that BiSeNet V2 is also included in the networks which use early downsampling. Finally, EDANet does not use a classic decoder module to upsample the feature maps to reduce computational costs. On the contrary, bilinear interpolation is used to upsample feature maps by a factor of 8 to the size of input images, as shown in Figure 24 between the block of the Precision Layer and the Output block. This method is based on the efficiency-oriented approach mentioned in Section 2. Although this approach reduces accuracy, the trade-off between accuracy and inference speed is still satisfactory.

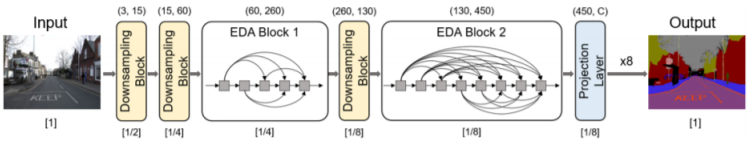



**Figure 24.** EDANet Network Architecture (adapted from [45]).

ENet [8] is an optimized deep neural network designed for fast inference and high accuracy. It follows a compact encoder–decoder architecture; however, as shown in Table 2, ENet uses a small size decoder for reducing computation cost and increasing inference speed. Furthermore, ENet introduces early downsampling to achieve low-latency operation. More specifically, early downsampling is the process of downsampling the input image at the first layers of the network. The reason behind this technique is that a downsampled version of the input image can be much more effective without losing vital information and thus without sacrificing accuracy. Last but not least, [8] was one of the first semantic image segmentation models that aimed at real-time performance and a milestone for the following research attempts.

**Table 2.** Network architecture of ENet [8] .

| Layer | Type | Output Channel | Output Resolution |
|---|---|---|---|
| 1 | Downsampling block | 16 | 512 × 256 |
| 2 | Downsampling block | 64 | 256 × 128 |
| 3–5 | 3 × Non-bt-1D | 128 | 128 × 64 |
| 5–7 | 2 × Conv-module | 64 | 256 × 128 |
| 8 | Downsampling block | 128 | 128 × 64 |
| 9 | Non-bt-1d (dilated 2) | 128 | 128 × 64 |
| 10 | Non-bt-1d (dilated 4) | 128 | 128 × 64 |
| 11 | Non-bt-1d (dilated 8) | 128 | 128 × 64 |
| 12 | Non-bt-1d (dilated 16) | 128 | 128 × 64 |
| 13 | Conv-module (dilated 2) | 128 | 128 × 64 |
| 14 | Conv-module (dilated 4) | 128 | 128 × 64 |
| 15 | Conv-module (dilated 8) | 128 | 128 × 64 |
| 16 | Conv-module (dilated 16) | 128 | 128 × 64 |
| 17 | Deconvolution (upsampling) | 64 | 256 × 128 |
| 18–19 | 2 × Non-bt-1D | 64 | 256 × 128 |
| 20 | Deconvolution (upsampling) | 16 | 512 × 256 |
| 21–22 | 2 × Non-bt-1D | 16 | 5125 × 256 |
| 23 | Deconvolution (upsampling) | C | 1024 × 512 |

The presented state-of-the-art models are listed in Table 3, along with their backbone and the efficiency-oriented approaches they share in common. The role of Table 3 is to summarize the ameliorative features that are mostly used in real-time semantic image segmentation, with the intention of clearing the way of designing efficient real-time semantic image segmentation models. Furthermore, Table 4 presents a collection of links to the implementation code of the state-of-the-art models.

**Table 3.** Common efficiency-oriented approaches used in real-time Semantic Segmentation models.

| Networks | Backbone | Efficiency-Oriented Approaches |
|---|---|---|
| DDRNet [16] | ResNet | two-branch network |
| STDC1-50 [11] | STDC | feature map size reduction<br>single branch efficient decoder |
| U-HarDNet-70 [42] | DenseNet | depthwise separable convolutions |
| HyperSeg [46] | EfficientNet-B1<br>PSPNet<br>ResNet18 | pointwise convolutions<br>depthwise convolutions<br>small decoder size |
| SwiftNetRN-18 [47] | ResNet-18<br>MobileNet V2 | small decoder size (lightweight decoder) |
| BiSeNet V2 [15] | VGGnet<br>Xception<br>MobileNet | two-branch network<br>fast-downsampling |
| TD4-BISE18 [48] | TDNet | Grouped convolutions |
| ShelfNet18 [33] | ResNet<br>Xception<br>DenseNet | multi-branch network<br>channel reduction |
| BiSeNet [14] | Xception39<br>ResNet18 | two-branch network |
| SegBlocks [17] | - | block-based processing |
| FasterSeg [27] | FasterSeg | multi-branch network |
| ESNet [32] | ESNet | multi-branch network<br>pointwise convolutions (factorized convolutions) |
| LEDNet [44] | ResNet | channel shuffling |
| ICNet [34] | image cascade network | multi-branch network |
| SQ [49] | SqueezeNet | decreased kernel size |
| ERFNet [43] | ERFNet | small decoder size |
| LinkNet [50] | ResNet18 | bypassing spatial information |
| SS [23] | ResNet18 | two-branch network |
| ContextNet [24] | ContextNet | two-branch network<br>depthwise separable convolutions<br>pruning |
| DSNet [51] | DSNet | channel shuffling |
| ESPNetv2 [37] | ESPNet | group pointwise convolutions<br>depthwise separable convolutions |
| ESSGG [38] | ERFNet | depthwise separable convolutions<br>channel shuffling |

**Table 3.** *Cont.*

| Networks | Backbone | Efficiency-Oriented Approaches |
|---|---|---|
| LWRF [52] | ResNet<br>MobileNetV2 | decreasing kernel's(receptive field) size<br>small decoder size |
| DABNet [39] | DABNet | depthwise separable convolutions |
| DFANet [40] | Xception | depthwise separable convolutions |
| Fast-SCNN [25] | Fast-SCNN | two-branch network<br>depthwise separable convolutions |
| ShuffleSeg [41] | ShuffleNet | Grouped convolutions<br>Channel shuffling |
| Template-Based-NAS-arch1 [13] | MobileNetV2 | separable convolutions<br>decreased kernel size |
| LiteSeg [53] | MobileNet<br>ShuffleNet<br>DarkNet19 | depthwise separable convolutions |
| Template-Based-NAS-arch0 [13] | MobileNetV2 | separable convolutions<br>decreased kernel size |
| ENet [8] | Enet<br>ResNets | early downsampling<br>small decoder size |
| ENet + Lovász-Softmax [54] | - | early downsampling<br>small decoder size |
| SegNet [9] | VGG16 | reuse of max-pooling indices |
| EDANet [45] | EDA | early downsampling<br>factorized convolutions |

**Table 4.** Links to source code of real-time Semantic Segmentation models.

| Networks | Github Repositories (All Accessed on 2 August 2021) | Ownership |
|---|---|---|
| DDRNet (ResNet) | https://github.com/ydhongHIT/DDRNet | Official code |
| HyperSeg | https://github.com/YuvalNirkin/hyperseg | Official code |
| STDC1-50 | https://github.com/MichaelFan01/STDC-Seg | Official code |
| SegBlocks | https://github.com/thomasverelst/segblocks-Segmentation-pytorch | Official code |
| SQ | https://github.com/klickmal/speeding_up_semantic_Segmentation | Third-party code |
| ERFNet | https://github.com/Eromera/erfnet | Official code |
| LinkNet | https://github.com/e-lab/LinkNet | Third-party code |
| ContextNet | https://github.com/klickmal/ContextNet | Third-party code |
| DSNet | https://github.com/s7ev3n/DSNet | Third-party code |
| ESPNetv2 | https://github.com/sacmehta/ESPNetv2 | Official code |
| LWRF | https://github.com/DrSleep/light-weight-refinenet | Third-party code |
| DABNet | https://github.com/Reagan1311/DABNet | Official code |
| DFANet | https://github.com/huaifeng1993/DFANet | Third-party code |
| Fast-SCNN | https://github.com/Tramac/Fast-SCNN-pytorch | Third-party code |
| ShuffleSeg | https://github.com/MSiam/TFSegmentation | Official code |
| U-HarDNet-70 | https://github.com/PingoLH/Pytorch-HarDNet | Official code |
| SwiftNetRN-18 | https://github.com/orsic/swiftnet | Official code |
| TD4-BISE18 | https://github.com/feinanshan/TDNet | Official code |
| ShelfNet18 | https://github.com/juntang-zhuang/ShelfNet | Official code |
| BiSeNet | https://github.com/osmr/imgclsmob | Third-party code |
| BiSeNet V2 | https://github.com/CoinCheung/BiSeNet | Third-party code |
| FasterSeg | https://github.com/VITA-Group/FasterSeg | Official code |
| ESNet | https://github.com/osmr/imgclsmob | Third-party code |
| LEDNet | https://github.com/xiaoyufenfei/LEDNet | Third-party code |
| ICNet | https://github.com/hszhao/ICNet | Official code |
| Template-Based-NAS-arch1 | https://github.com/drsleep/nas-segm-pytorch | Official code |
| LiteSeg | https://github.com/tahaemara/LiteSeg | Official code |
| Template-Based-NAS-arch0 | https://github.com/drsleep/nas-segm-pytorch | Official code |
| ENet | https://github.com/iArunava/ENet-Real-Time-Semantic-Segmentation | Third-party code |
| ENet + Lovász-Softmax | https://github.com/bermanmaxim/LovaszSoftmax | Official code |
| SegNet | https://github.com/alexgkendall/caffe-segnet | Third-party code |
| EDANet | https://github.com/shaoyuanlo/EDANet | Official code |

## 4. Evaluation Framework

### 4.1. Datasets

In this Section, we present the most popular datasets used in the field of semantic segmentation aiming for autonomous driving. The choice of a suitable dataset is of great importance for the training and the evaluation of the created models. The challenging task of dataset selection is one of the first major steps in research, especially for a difficult and demanding scientific field, such as autonomous driving, in which the vehicle exposure environment can be complex and varied. Each of the following datasets have been used for training and evaluation of real-time semantic segmentation models. Example images from the following datasets are shown in Figure 25.

#### 4.1.1. Cityscapes

Cityscapes [55] is one of the most popular datasets in the field of semantic segmentation and autonomous driving. At first, it was recorded as a video, thus the images are especially selected frames captured from 50 different cities The selection was based upon the need for a great number of objects, variety of scenes and variety of backgrounds. Totally, 30 individual classes are provided grouped into 8 categories. The Cityscapes dataset contains around 5000 images of fine annotation and 20,000 images of coarse annotation. The contained urban street scenes were captured over several months of spring, summer and fall during daytime with good weather conditions.

#### 4.1.2. CamVid

CamVid [56] is an image dataset containing road scenes. At first, it was recorded as a video of five sequences. The resolution of the images that CamVid is consisted of is $960 \times 720$. The dataset provides in total 32 classes. Some of the most important ones for road-scene understanding are: car, pedestrian, motorcycle, traffic light, traffic cone, lane markings, sign, road, truck/bus and child.

#### 4.1.3. MS COCO—Common Objects in Context

COCO [57] is an extensive dataset suitable for tasks such as object detection and semantic image segmentation. It contains 328,000 images. From the total amount of images, more than 82,873 are specified for training, around 41,000 for validation and over 80,000 for testing.

#### 4.1.4. KITTI

KITTI [58] is a hallmark dataset in the field of autonomous driving. It consists of a large-scale amount of traffic scenes. The data were collected with a diverse set of different sensors such as RGB and grayscale cameras and a 3D laser scanner..

#### 4.1.5. KITTI-360

KITTI-360 [59] is a wide-reaching dataset consisting of well-crafted annotations and great scene information. Data were captured from various suburbs of Karlsruhe, Germany. In total, it contains more than 320,000 images and 100,000 laser scans in a driving distance of 73.7 km. Designers annotated both static and dynamic 3D scene elements with rough bounding primitives. The definition of the labels is consistent with the Cityscapes dataset. Finally, it employs 19 classes for evaluation.

#### 4.1.6. SYNTHIA

SYNTHIA dataset [60] contains 9400 road scenes captured from a simulation of a city environment. It employs 13 classes. The resolution of the images is $1280 \times 960$.

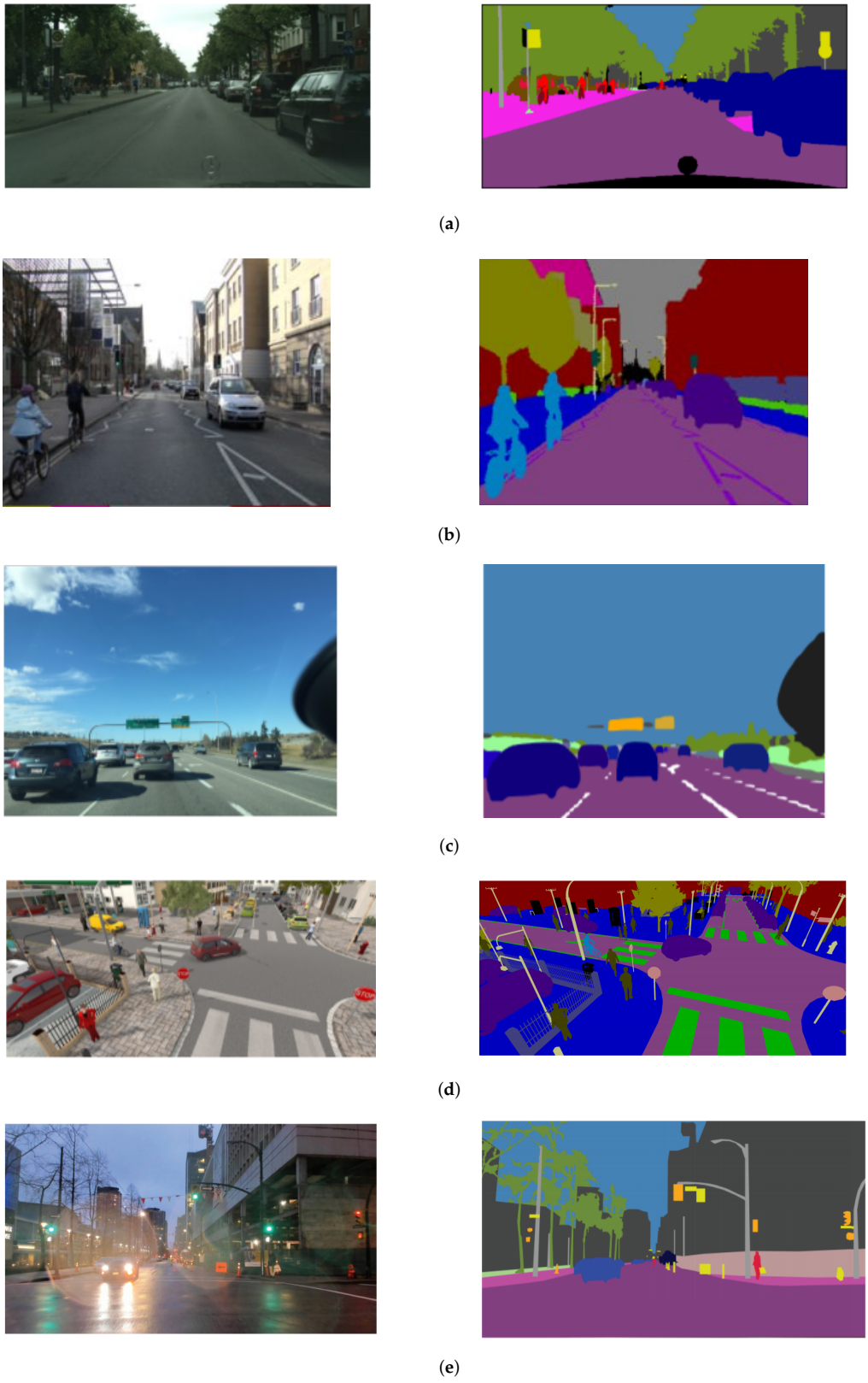


**Figure 25.** Examples of the original image and the corresponding ground truth image for the datasets: (**a**) Cityscapes (**b**) CamVid (**c**) Mapillary Vistas (**d**) SYNTHIA (**e**) RaidaR.

#### 4.1.7. Mapillary Vistas

Mapillary Vistas Dataset [61] is an exhaustive dataset of road scenes with human-crafted pixel-wise annotations. It is designed for road-scene understanding from images

captured globally. It features 25,000 high-resolution images, 124 semantic object categories, 100 instance-specifically annotated categories. It covers scenes from 6 continents, and it provides a diversity of weather, season, time of day, camera, and viewpoint.

#### 4.1.8. ApolloScape

ApolloScape [62] is a large dataset consisting of over 140,000 video frames (73 street scene videos) from various locations in China under varying weather conditions.

#### 4.1.9. RaidaR

RaidaR [63] is a large-scale dataset of rainy road scenes, specifically designed for autonomous driving. RaidaR comprises 58,542 images of rainy weather. From this amount, a subset of 5000 is annotated with semantic segmentation. Moreover, 4085 sunny images were also annotated with semantic segmentations. Thus, RaidaR is one of the extensive datasets. Finally, it is one of the most promising ones due to the challenges it presents because of the rainy weather conditions.

### *4.2. Metrics*

In this Section, we summarize some of the most popular metrics used for evaluating the performance of semantic segmentation models. The evaluation of these models and especially, those who are designed for real-time semantic segmentation depends on two key factors: effectiveness and efficiency.

#### 4.2.1. Metrics Related to Effectiveness

For $k+1$ classes (+1 class corresponds to background) and $p_{ij}$ the number of pixels of class i predicted/presumed as belonging to class $j$ we define the following metrics [64]:

- *Pixel Accuracy:* is defined as the ratio of correctly classified pixels divided by their total number.

$$PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}} \tag{1}$$

- *Mean Pixel Accuracy:* is an extension of Pixel Accuracy, which calculates the ratio of correct pixels in a per-class basis and then averaged over the total number of classes.

$$MPA = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}} \tag{2}$$

- *Intersection over Union (IoU):* is a very popular metric used in the field of semantic image segmentation. IoU is defined the intersection of the predicted segmentation map and the ground truth, divided by the area of union between the predicted segmentation map and the ground truth.

$$IoU = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij} + \sum_{i=0}^{k} \sum_{j=0}^{k} p_{ji} - \sum_{i=0}^{k} p_{ii}} \tag{3}$$

- *mean Intersection over Union (mIoU):* is the most widely used metric for semantic segmentation. It is defined as the average IoU over all classes.

$$mIoU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}} \tag{4}$$

#### 4.2.2. Metrics Related to Efficiency

To evaluate the efficiency of a semantic image segmentation model is vital to use metrics that define the processing time of the models and their computational and memory burden.

- *Frames per second:* A standard metric for evaluating the time needed for a deep-learning model to process a series of image frames of a video is "Frames per second". Especially in real-time semantic segmentation applications, such as autonomous driving, it is crucial to know the exact number of frames a model can process below the time of a second. It is a very popular metric, and it can be really helpful for comparing different segmentation methods and architectures.
- *Inference time:* is another standard metric for evaluating the speed of semantic segmentation. It is the inverse of FPS (Frame Rate), and it measures the execution time for a frame.
- *Memory usage:* it is also a significant parameter to be taken into consideration when comparing deep-learning models in terms of speed and efficiency. Memory usage can be measured in different ways. Some researchers use the number of parameters of the network. Another way is to define the memory size to represent the network and lastly, a metric used frequently is to measure the number of floating-point operations (FLOPs) required for the execution.

## 5. Discussion

In this Section, we present our findings and key insights after considering the comprehensive analysis of the state-of-the-art networks presented in Section 3. It constitutes a fruitful discussion that comprises the emergence of a common operational pipeline and a comparative performance analysis enriched by a discussion of the dependencies on the used hardware along with the limitations and influence addressed by the current benchmarking datasets.

### 5.1. Common Operational Pipeline

The description of the state-of-the-art models for real-time semantic segmentation, which has been presented in Section 3, has shown sound evidence that most models share a common operational pipeline.

In particular, the first type of semantic segmentation networks which achieved real-time performance is based upon the encoder–decoder architecture. Representative example is ENet [8]. The encoder module uses convolutional and pooling layers to perform feature extraction. On the other hand, the decoder module recovers the spatial details from the sub-resolution features, while predicts the object labels (i.e., the semantic segmentation) [25]. A standard choice for the encoder module is a lightweight CNN backbone, such as GoogLeNet [5] or a revised version of it, namely Inception-v3 [10]. The design of the decoder module usually consists of upsampling layers based on bipolar interpolations or transposed convolutions.

In the effort to design efficient and at the same time accurate models, two-branch and multi-branch networks have been proposed. Instead of a single branch encoder, a two-branch network uses a deep branch to encode high-level semantic context information and a shallow branch to encode substantial/rich spatial details of higher resolution. Under the same concept, multi-branch architectures integrate branches handling different resolutions of the input image (high, low and medium). However, the features extracted from the different branches must be merged to proceed to the segmentation map. To this end, two-branch networks introduce a fusion module to combine the output of the encoding branches. The fusion module can be a Feature Fusion module in which the output features are joined by concatenation or addition, an Aggregation Layer (BiSeNet V2), a Bilateral Fusion module (DDRNet), or a Cascade Feature Fusion Unit (ICNet).

In a nutshell, there are three types of modules used in the operational pipeline of real-time semantic segmentation: an encoding, a decoding and a fusion module. There are several configurations where the fusion module is part of the decoder. Figure 26 shows the three modules and the corresponding standard yet dominant, design choices.

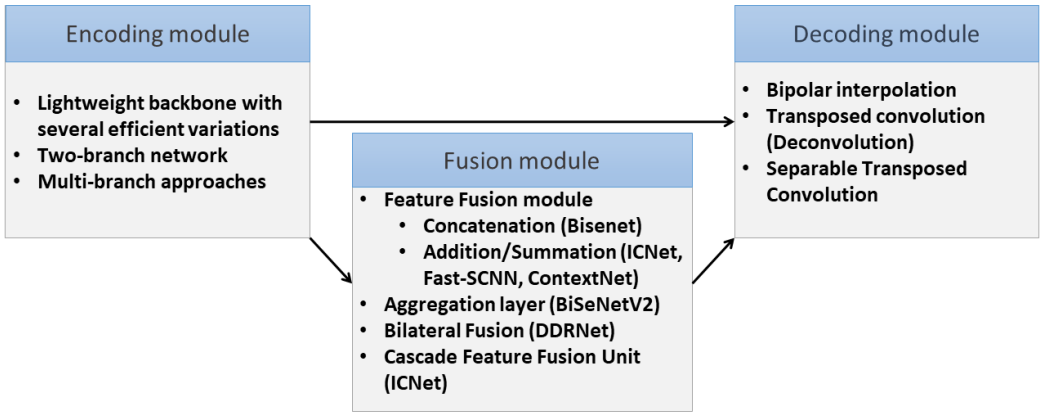



**Figure 26.** Basic modules of efficient Semantic Segmentation.

### *5.2. Comparative Performance Analysis*

At Table 5, a detailed listing of the performance of state-of-the-art-models in terms of effectiveness (expressed by the mIoU metric) and efficiency (expressed by the FPS metric) for both "Cityscapes" and "CamVid" datasets. Concerning the results of the "Cityscapes" dataset, STDC1-50 presents the best result in terms of inference speed with 250.4 FPS and 71.9% mIoU in terms of accuracy. Moreover, FasterSeg achieves 163.9 FPS in terms of inference speed and 71.5% mIoU in terms of accuracy. Finally, BiSeNet V2 and Fast-SCNN achieve 156 FPS and 123.5 FPS with 72.6% and 68% mIoU, respectively. The results of these models are exceptional and really promising about the future of real-time semantic segmentation. As far as the results on the "CamVid" dataset are concerned, FasterSeg presents the outstanding results of 398.1 FPS in terms of inference speed and 71.1% mIoU in terms of accuracy. Moreover, DDRNet-23-slim achieve 230 FPS and 78% mIoU, while STDC1-Seg achieved 197.6 FPS and 73% mIoU.

After the aforementioned objective evaluation, we examined the presented models by taking into account the trade-off between speed and accuracy which can be shown in Figure 27a,b. In particular, in Figure 27a, where experimental results for the "Cityscapes" dataset are considered, we can observe that STDC1-50 [11] presents a satisfactory trade-off between mIoU and FPS, compared to the other models. Although other models such as U-HardNet70, BiSeNet v2 Large, SwiftNetRN-18 and ShelfNet18 achieve better performance in terms of accuracy, they lag behind in terms of frame rate. Moreover, in Figure 27b, where experimental results for the "CamVid" dataset are considered, it is shown that FasterSeg [27] can be considered to be an appealing choice for real-time semantic image segmentation, by taking into account the trade-off between accuracy and efficiency compared to all other models.

#### 5.2.1. Dataset-Oriented Performance

Another issue concerning the finding of the best suited model for a problem at hand is the dataset selection for training. This is a critical issue in the field of deep learning in particular, as the data used to train a model is one of the most influential factors for the performance of the neural network. Therefore, in autonomous driving where the models work in real time, they should be trained in an earlier stage with the most suitable data. Moreover, the comparison of the semantic segmentation models should be made for the same dataset. Thus, research papers use the same dataset to compare different models. Nonetheless, in the area of real-time semantic segmentation, most of the models are being compared in the "Cityscapes" and "CamVid" datasets. Although this fact simplifies the searching for the fastest, yet most accurate, segmentation model it may not provide the most objective decision for the choice of the most suitable method. In particular, in the area of autonomous driving where every mistake may cause irreversible consequences, there is a practical necessity for training and testing the deep-learning models used in a

wide variety of situations of the external environment (day, light, traffic, etc.). For example, "Cityscapes" and "CamVid" which are one of the most popular and most used datasets for benchmark tests, lack some vital features, regarding their diversity. In detail, they contain images with good/medium weather conditions on daytime. However, an efficient and safe autonomous driving system must have the ability to function under adverse weather conditions, such as snowfall and of course at night-time, especially in the case of emergency. To this end, one could adopt transfer learning techniques as will be discussed in Section 6 to remedy this limitation raised by currently available datasets.

**Table 5.** Performance of real-time Semantic Segmentation models in "Cityscapes" and "CamVid" datasets.

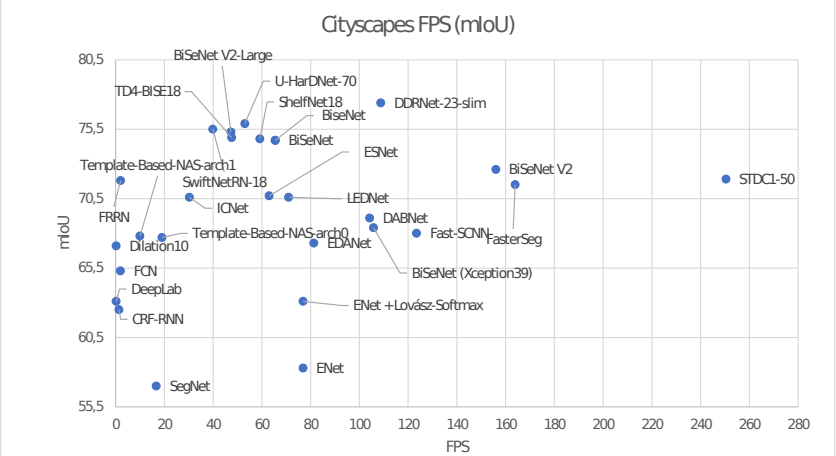
| Networks (Backbone) | Cityscapes | | | CamVid | | |
|---|---|---|---|---|---|---|
| | mIoU (%) | FPS | Hardware | mIoU (%) | FPS | Hardware |
| SQ (SqueezeNet) | **84.3** | 16.7 | Jetson TX1 | - | - | - |
| DDRNet-23 | 79.4 | 38.5 | GTX 2080Ti | **79.9** | 94 | GTX 2080Ti |
| HyperSeg-S (EfficientNet-B1) | 78.1 | 16.1 | GTX 1080TI | 78.4 | 38.0 | GTX 1080TI |
| DDRNet-23-slim | 77.4 | 108.8 | GTX 2080Ti | 78.0 | 230 | GTX 2080Ti |
| LinkNet (ResNet18) | 76.4 | 18.7 | Titan X | - | - | - |
| U-HarDNet-70 (DenseNet) | 75.9 | 53 | GTX 1080Ti | 67.7 | 149.3 | Titan V |
| HyperSeg-M (EfficientNet-B1) | 75.8 | 36.9 | GTX 1080Ti | - | - | - |
| SwiftNetRN-18 (ResNet-18, MobileNet V2) | 75.5 | 39.9 | GTX 1080Ti | 73.86 | 43.3 | GTX 1080Ti |
| TD4-BISE18 (TDNet) | 74.9 | 47.6 | Titan Xp | 74.8 | 59.2 | Titan Xp |
| ShelfNet18 (ResNet, Xception, DenseNet) | 74.8 | 59.2 | GTX 1080Ti | - | - | - |
| BiSeNet (ResNet18) | 74.7 | 65.5 | Titan Xp | 68.7 | - | Titan Xp |
| SegBlocks-RN18 (t = 0.4) | 73.8 | 48.6 | GTX 1080Ti | - | - | - |
| SS (ResNet18) | 72.9 | 14.7 | GTX 980 | - | - | - |
| BiSeNet V2 (VGGnet, Xception, MobileNet, ShuffleNet) | 72.6 | 156 | GTX 1080 Ti | 72.4 | 124.5 | GTX 1080 Ti |
| STDC1-50 (STDC) | 71.9 | **250.4** | GTX 1080 Ti | - | - | - |
| FasterSeg (FasterSeg) | 71.5 | 163.9 | GTX 1080Ti | 71.1 | **398.1** | GTX 1080Ti |
| DFANet (Xception) | 71.3 | 100 | Titan X | 64.7 | 120 | Titan X |
| ESNet (ESNet) | 70.7 | 63 | GTX 1080Ti | - | - | - |
| ICNet (image cascade network) | 70.6 | 30.3 | Titan X | 67.1 | 27.8 | Titan X |
| LEDNet (ResNet) | 70.6 | 71 | GTX 1080Ti | - | - | - |
| STDC1-Seg (STDC) | - | - | - | 73.0 | 197.6 | GTX 1080Ti |
| ERFNet (ERFNet) | 69.7 | 41.6 | Titan X | - | - | - |
| DSNet (DSNet) | 69.3 | 36.5 | GTX 1080Ti | - | - | - |
| DABNet (DABNet) | 69.1 | 104.2 | GTX 1080Ti | - | - | - |
| BiSeNet (Xception39) | 68.4 | 105.8 | Titan Xp | 68.7 | - | Titan Xp |
| ESSGG (ERFNet) | 68 | - | - | - | - | - |
| Fast-SCNN (Fast-SCNN) | 68 | 123.5 | Nvidia Titan Xp | - | - | - |
| Template-Based-NAS-arch1 (MobileNetV2) | 67.8 | 10 | GTX 1080Ti | 63.2 | - | GTX 1080Ti |
| LiteSeg (MobileNetV2) | 67.8 | 22 | GTX 1080Ti | - | - | - |
| Template-Based-NAS-arch0 (MobileNetV2) | 67.7 | 19 | GTX 1080Ti | 63.9 | - | GTX 1080Ti |
| EDANet (EDA) | 67.3 | 81.3 | Titan X | 66.4 | - | GTX 1080Ti |
| ESPNetv2 (ESPNet) | 66.2 | 5.55 | GTX 1080 Ti | - | - | - |
| ContextNet (ContextNet) | 66.1 | 23.86 | Titan X | - | - | - |
| ENet + Lovász-Softmax | 63.1 | 76.9 | | - | - | - |
| ShuffleSeg (ShuffleNet) | 58.3 | 15.7 | Jetson TX2 | - | - | - |
| ENet (Enet, ResNets) | 58.3 | 46.8 | Titan X | 51.3 | - | - |
| LWRF (ResNet, MobileNetV2) | 45.8 | - | GTX 1080Ti | - | - | - |

#### 5.2.2. The Influence of Hardware

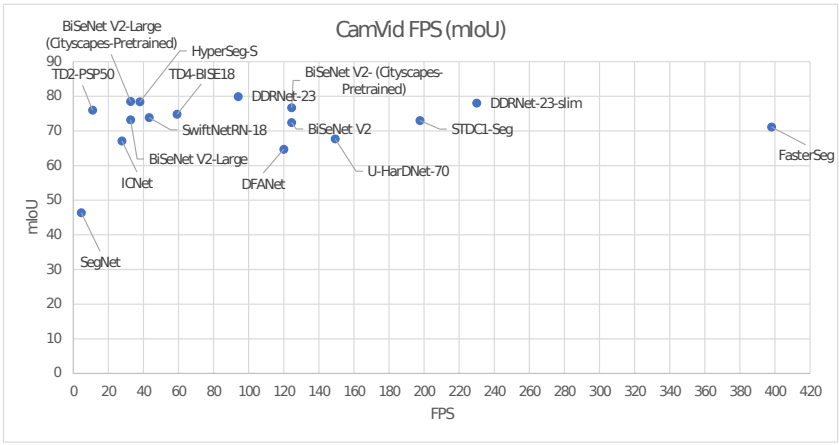
For a concise comparative study, it is crucial that the experiments should be performed under the same conditions. In particular, the inference time is measured for a particular hardware configuration. As a result, the comparisons between the different architectures should be taken under deep examination, as the specific GPU model, the ram and other parameters play an important role in the efficiency which the models present. For this reason, the vast majority of the examined research papers provide information about the hardware specifications and the experiment conditions under which their proposed models

are being evaluated. To this end, at Table 5 the performance of each model is coupled with the GPU used during the experimental work.

Beyond modern GPUs, fast inference could be achieved with other powerful computer devices. In fact, on the subject of the hardware choice [65] provide a thorough and concise list of the commercially available edge devices. Edge TPU of Google, Neural Compute $2^2$ of Intel, Jetson Nano, TX1, AGX Xavier of NVidia, AI Edge of Xilinx and Atlas 200 DK of Huawei are some of the modern commercially available edge computing devices for mobile and aerial robots deep-learning inference.




(**a**)




(**b**)

**Figure 27.** Performance in terms of FPS in relation to mIoU for the state-of-the-art models in (**a**) "Cityscapes" dataset and (**b**) "CamVid" dataset.

## 6. Future Research Trends

In this Section, we present some promising techniques towards improving the performance of semantic image segmentation that can be adopted in future research efforts.

- Transfer learning: Transfer learning transfers the knowledge (i.e., weights) from the source domain to the target domain, leading to a great positive effect on many domains that are difficult to improve because of insufficient training data [66]. By the same token, transfer learning can be useful in real-time semantic segmentation by reducing the amount of the needed training data, therefore the time required. Moreover, as [47] proposes, transfer learning offers a greater regularization to the parameters of a pre-trained model. In [67,68], the use of transfer learning improved the semantic segmentation performance in terms of accuracy.

- Domain adaptation: Domain adaptation is a subset of transfer learning. Domain adaptation's goal is to ameliorate the model's effectiveness on a target domain using the knowledge learned in a different, yet coherent source domain [69]. In [70] the use of domain adaptation, achieved a satisfactory increase in mIoU on unseen data, without the adding extra computational burden, which is one of the great goals of real-time semantic segmentation. Thus, domain adaptation might be a valuable solution for the future of autonomous driving, by giving accurate results on unseen domains while functioning in low latency.
- Self-supervised learning: Human-crafting large-scale data has a high cost, is time-consuming and sometimes is an almost impracticable process. Especially in the field of autonomous driving, where millions of data are required due to the complexity of the street scenes, many hurdles arise in the annotation of the data. Self-supervised learning is a subcategory of unsupervised learning introduced to learn representations from extensive datasets without providing manually labeled data. Thus, any human actions (and involvements) are avoided, reducing the operational costs [71].
- Weakly supervised learning: Weakly supervised learning is related to learning methods which are characterized by coarse-grained labels or inaccurate labels. As reported in [71], the cost of obtaining weak supervision labels is generally much cheaper than fine-grained labels for supervised methods. In [72], a superior performance compared to other methods has been achieved in terms of accuracy, for a benchmark that uses the "Cityscapes" and "CamVid" datasets. Additionally, ref. [73] with the use of classifier heatmaps and a two-stream network shows greater performance in comparison to the other state-of-the-art models that use additional supervision.
- Transformers: ref. [74] allows the modeling of a global context already at the first layer and throughout the network, contrary to the ordinary convolutional-based methods. Segmenter approach reaches a mean IoU of 50.77% on ADE20K [75], surpassing all previous state-of-the-art convolutional approaches by a gap of 4.6%. Thus, transformers appear to be promising methods for the future of semantic segmentation.

## 7. Conclusions

In this paper, we present an overview of the best methods and architectures for real-time semantic segmentation. The collection of the possible methods used for real-time segmentation aims at helping researchers find the most suitable techniques for boosting speed of deep-learning models while preserving their accuracy. Furthermore, tables listing the most accurate and efficient state-of-the-art real-time models are provided. The selection of the chosen models is based on experiments made using "Cityscapes" and "CamVid" datasets, depending on their mIoU, FPS and inference time they achieved. For studying purposes, a list of extensively used datasets and metrics of real-time semantic segmentation was described. Finally, this survey discusses current issues, thus showing areas of improvement in real-time semantic segmentation regarding the needs of autonomous driving and other high-end technological fields.

**Author Contributions:** Supervision, A.A. and I.P. (Ioannis Pratikakis); Writing—original draft, I.P. (Ilias Papadeas) and L.T.; Writing—review and editing, I.P. (Ilias Papadeas) and L.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code:T2EDK-02743).

**Acknowledgments:** This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code:T2EDK-02743). We would also like to thank NVIDIA Corporation, which kindly donated the Titan X GPU, that has been used for this research.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Janai, J.; Güney, F.; Behl, A.; Geiger, A. Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *Found. Trends Comput. Graph. Vis.* **2020**, *12*, 85. [CrossRef]
2. Wang, M.; Liu, B.; Foroosh, H. Factorized Convolutional Neural Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 545–553. [CrossRef]
3. Guo, Y.; Li, Y.; Feris, R.; Wang, L.; Simunic, T. *Depthwise Convolution Is All You Need for Learning Multiple Visual Domains*; AAAI: Honolulu, HI, USA, 2019.
4. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807.
5. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
6. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:abs/1704.04861.
7. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv* **2017**, arXiv:cs.CV/1707.01083.
8. Paszke, A.; Chaurasia, A.; Kim, S.; Culurciello, E. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv* **2016**, arXiv:abs/1606.02147.
9. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
10. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. [CrossRef]
11. Fan, M.; Lai, S.; Huang, J.; Wei, X.; Chai, Z.; Luo, J.; Wei, X. Rethinking BiSeNet For Real-time Semantic Segmentation. *arXiv* **2021**, arXiv:cs.CV/2104.13188.
12. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
13. Nekrasov, V.; Shen, C.; Reid, I. Template-Based Automatic Search of Compact Semantic Segmentation Architectures. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*; IEEE: Los Alamitos, CA, USA, 2020.
14. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. *BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation*; Computer Vision–ECCV 2018; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 334–349.
15. Yu, C.; Gao, C.; Wang, J.; Yu, G.; Shen, C.; Sang, N. BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation. *arXiv* **2020**, arXiv:abs/2004.02147.
16. Hong, Y.; Pan, H.; Sun, W.; Jia, Y. Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes. *arXiv* **2021**, arXiv:abs/2101.06085.
17. Verelst, T.; Tuytelaars, T. SegBlocks: Block-Based Dynamic Resolution Networks for Real-Time Segmentation. *arXiv* **2020**, arXiv:abs/2011.12025.
18. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both Weights and Connections for Efficient Neural Network. *arXiv* **2015**, arXiv:abs/1506.02626.
19. Han, S.; Mao, H.; Dally, W. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:1510.00149.
20. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. *arXiv* **2017**, arXiv:cs.CV/1608.08710.
21. He, Y.; Zhang, X.; Sun, J. Channel Pruning for Accelerating Very Deep Neural Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1398–1406.
22. Takos, G. A Survey on Deep Learning Methods for Semantic Image Segmentation in Real-Time. *arXiv* **2020**, arXiv:abs/2009.12942.
23. Wu, Z.; Shen, C.; van den Hengel, A. Real-time Semantic Image Segmentation via Spatial Sparsity. *arXiv* **2017**, arXiv:cs.CV/1712.00213.
24. Poudel, R.P.K.; Bonde, U.D.; Liwicki, S.; Zach, C. *ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-Time*; BMVC: Newcastle, UK, 2018.
25. Poudel, R.P.K.; Liwicki, S.; Cipolla, R. Fast-SCNN: Fast Semantic Segmentation Network. *arXiv* **2019**, arXiv:cs.CV/1902.04502.
26. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
27. Chen, W.; Gong, X.; Liu, X.; Zhang, Q.; Li, Y.; Wang, Z. FasterSeg: Searching for Faster Real-time Semantic Segmentation. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
28. Elsken, T.; Metzen, J.H.; Hutter, F. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
29. Elsken, T.; Metzen, J.H.; Hutter, F. Neural Architecture Search. In *Automated Machine Learning: Methods, Systems, Challenges*; Hutter, F., Kotthoff, L., Vanschoren, J., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 63–77.

30. Liu, C.; Chen, L.C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.; Li, F.-F. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 82–92.
31. Gou, J.; Yu, B.; Maybank, S.; Tao, D. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
32. Wang, Y.; Zhou, Q.; Wu, X. *ESNet: An Efficient Symmetric Network for Real-Time Semantic Segmentation*; PRCV: Xi'an, China, 2019.
33. Zhuang, J.; Yang, J.; Gu, L.; Dvornek, N. ShelfNet for Fast Semantic Segmentation. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019; pp. 847–856.
34. Zhao, H.; Qi, X.; Shen, X.; Shi, J.; Jia, J. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. In *Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 418–434.
35. Bergman, A.W.; Lindell, D.B. *Factorized Convolution Kernels in Image Processing*; Stanford University: Stanford, CA, USA, 2019.
36. Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. In *Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switerland, 2018; pp. 561–580.
37. Mehta, S.; Rastegari, M.; Shapiro, L.; Hajishirzi, H. ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 9182–9192. [CrossRef]
38. Vallurupalli, N.; Annamaneni, S.; Varma, G.; Jawahar, C.V.; Mathew, M.; Nagori, S. Efficient Semantic Segmentation Using Gradual Grouping. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018; pp. 711–7118.
39. Li, G.; Yun, I.; Kim, J.; Kim, J. *DABNet: Depth-Wise Asymmetric Bottleneck for Real-Time Semantic Segmentation*; BMVC: Cardiff, UK, 2019.
40. Li, H.; Xiong, P.; Fan, H.; Sun, J. DFANet: Deep Feature Aggregation for Real-Time Semantic Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 9514–9523.
41. Gamal, M.; Siam, M.; Abdel-Razek, M. ShuffleSeg: Real-time Semantic Segmentation Network. *arXiv* **2018**, arXiv:cs.CV/1803.03816.
42. Chao, P.; Kao, C.; Ruan, Y.; Huang, C.; Lin, Y. HarDNet: A Low Memory Traffic Network. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 3551–3560. [CrossRef]
43. Romera, E.; Álvarez, J.M.; Bergasa, L.M.; Arroyo, R. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 263–272. [CrossRef]
44. Wang, Y.; Zhou, Q.; Liu, J.; Xiong, J.; Gao, G.; Wu, X.; Latecki, L. Lednet: A Lightweight Encoder-Decoder Network for Real-Time Semantic Segmentation. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 1860–1864.
45. Lo, S.Y.; Hang, H.; Chan, S.; Lin, J.J. Efficient Dense Modules of Asymmetric Convolution for Real-Time Semantic Segmentation. In Proceedings of the ACM Multimedia Asia, Beijing, China, 15–18 December 2019.
46. Nirkin, Y.; Wolf, L.; Hassner, T. HyperSeg: Patch-wise Hypernetwork for Real-time Semantic Segmentation. *arXiv* **2020**, arXiv:abs/2012.11582.
47. Oršic, M.; Krešo, I.; Bevandic, P.; Šegvic, S. In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 12599–12608. [CrossRef]
48. Hu, P.; Caba, F.; Wang, O.; Lin, Z.; Sclaroff, S.; Perazzi, F. Temporally Distributed Networks for Fast Video Semantic Segmentation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 8815–8824. [CrossRef]
49. Treml, M.; Arjona-Medina, J.; Unterthiner, T.; Durgesh, R.; Friedmann, F.; Schuberth, P.; Mayr, A.; Heusel, M.; Hofmarcher, M.; Widrich, M.; et al. Speeding up Semantic Segmentation for Autonomous Driving. In Proceedings of the 2016 Machine Learning for Intelligent Transportation Systems (MLITS) in Conjunction with the Thirtieth Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016.
50. Chaurasia, A.; Culurciello, E. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4. [CrossRef]
51. DSNet for Real-Time Driving Scene Semantic Segmentation. *arXiv* **2018**, arXiv:abs/1812.07049.
52. Nekrasov, V.; Shen, C.; Reid, I.D. Light-Weight RefineNet for Real-Time Semantic Segmentation. *arXiv* **2018**, arXiv:abs/1810.03272.
53. Emara, T.; Munim, H.E.A.E.; Abbas, H.M. LiteSeg: A Novel Lightweight ConvNet for Semantic Segmentation. In Proceedings of the 2019 Digital Image Computing: Techniques and Applications (DICTA), Perth, Australia, 2–4 December 2019; pp. 1–7. [CrossRef]
54. Berman, M.; Triki, A.; Blaschko, M.B. The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4413–4421.

55. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223.
56. Brostow, G.J.; Fauqueur, J.; Cipolla, R. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.* **2009**, *30*, 88–97. [CrossRef]
57. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755.
58. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
59. Xie, J.; Kiefel, M.; Sun, M.T.; Geiger, A. Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
60. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3234–3243. [CrossRef]
61. Neuhold, G.; Ollmann, T.; Bulò, S.R.; Kontschieder, P. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5000–5009. [CrossRef]
62. Wang, P.; Huang, X.; Cheng, X.; Zhou, D.; Geng, Q.; Yang, R. The ApolloScape Open Dataset for Autonomous Driving and Its Application. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2702–2719. [CrossRef] [PubMed]
63. Jin, J.; Fatemi, A.; Lira, W.P.; Yu, F.; Leng, B.; Ma, R.; Mahdavi-Amiri, A.; Zhang, H.R. RaidaR: A Rich Annotated Image Dataset of Rainy Street Scenes. *arXiv* **2021**, arXiv:abs/2104.04606.
64. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Martinez-Gonzalez, P.; Garcia-Rodriguez, J. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput. J.* **2018**, *70*, 41–65. [CrossRef]
65. Faniadis, E.; Amanatiadis, A. Deep Learning Inference at the Edge for Mobile and Aerial Robotics. In Proceedings of the 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Abu Dhabi, United Arab Emerites, 4–6 November 2020; pp. 334–340. [CrossRef]
66. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A Survey on Deep Transfer Learning. In *Artificial Neural Networks and Machine Learning—ICANN 2018*; Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 270–279.
67. Ye, J.; Lu, C.; Xiong, J.; Wang, H. Semantic Segmentation Algorithm Based on Attention Mechanism and Transfer Learning. *Math. Probl. Eng.* **2020**, *2020*, 1–11.
68. Sharma, S.; Ball, J.E.; Tang, B.; Carruth, D.W.; Doude, M.; Islam, M.A. Semantic Segmentation with Transfer Learning for Off-Road Autonomous Driving. *Sensors* **2019**, *19*, 2577. [CrossRef]
69. Csurka, G. Deep Visual Domain Adaptation. In Proceedings of the 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 1–4 September 2020; pp. 1–8. [CrossRef]
70. Sankaranarayanan, S.; Balaji, Y.; Jain, A.; Lim, S.N.; Chellappa, R. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3752–3761.
71. Jing, L.; Tian, Y. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [CrossRef] [PubMed]
72. Wang, X.; Ma, H.; You, S. Deep clustering for weakly-supervised semantic segmentation in autonomous driving scenes. *Neurocomputing* **2020**, *381*, 20–28. [CrossRef]
73. Saleh, F.S.; Aliakbarian, M.S.; Salzmann, M.; Petersson, L.; Alvarez, J.M. Bringing Background into the Foreground: Making All Classes Equal in Weakly-Supervised Video Semantic Segmentation. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2125–2135. [CrossRef]
74. Strudel, R.; Garcia, R.; Laptev, I.; Schmid, C. Segmenter: Transformer for Semantic Segmentation. *arXiv* **2021**, arXiv:cs.CV/2105.05633.
75. Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; Torralba, A. Scene Parsing through ADE20K Dataset. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5122–5130. [CrossRef]