In Defense of Pre-trained ImageNet Architectures for Real-time Semantic Segmentation of Road-driving Images

Marin Oršić* Ivan Krešo* Petra Bevandić Siniša Šegvić
Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia
name.surname@fer.hr

Abstract

Recent success of semantic segmentation approaches on demanding road driving datasets has spurred interest in many related application fields. Many of these applications involve real-time prediction on mobile platforms such as cars, drones and various kinds of robots. Real-time setup is challenging due to extraordinary computational complexity involved. Many previous works address the challenge with custom lightweight architectures which decrease computational complexity by reducing depth, width and layer capacity with respect to general purpose architectures. We propose an alternative approach which achieves a significantly better performance across a wide range of computing budgets. First, we rely on a light-weight general purpose architecture as the main recognition engine. Then, we leverage light-weight upsampling with lateral connections as the most cost-effective solution to restore the prediction resolution. Finally, we propose to enlarge the receptive field by fusing shared features at multiple resolutions in a novel fashion. Experiments on several road driving datasets show a substantial advantage of the proposed approach, either with ImageNet pre-trained parameters or when we learn from scratch. Our Cityscapes test submission entitled SwiftNetRN-18 delivers 75.5% MIoU and achieves 39.9 Hz on 1024×2048 images on GTX1080Ti.

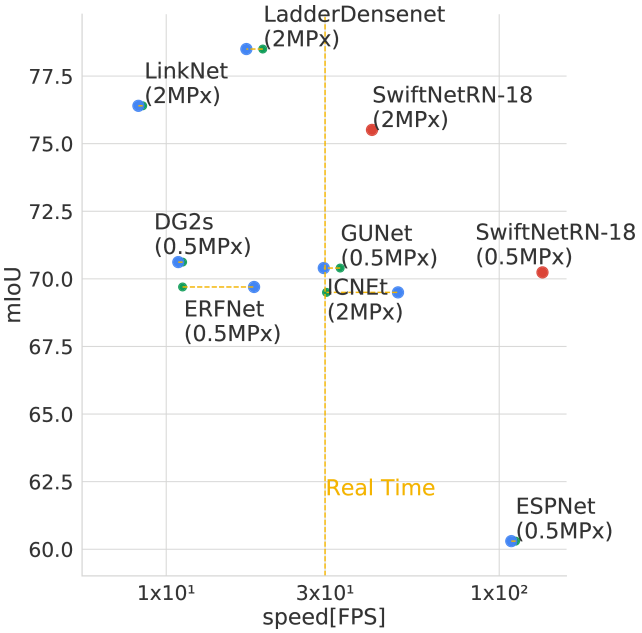

Figure 1. Speed-accuracy trade-off for different semantic semantic segmentation approaches on Cityscapes test on GTX1080Ti (except for DG2s which reports only validation performance). Red dots represent our method. Other methods are displayed in green, whereas blue dots show estimated frame rates of the corresponding methods on our GPU (please refer to subsection 4.2 for details). Our submissions achieve the best accuracy and the best speed among all approaches aiming at real-time operation.

1. Introduction

Semantic segmentation is an important dense prediction task in which the inference targets posterior distribution over a known set of classes in each image pixel [6, 20, 3]. Currently, the best results are achieved with deep fully convolutional models which require extraordinary computational resources. Many important applications such as autonomous navigation or driver assistance require inference on very large images in order to cover a wide field of view and perceive pedestrians at distances of over 200m. At the same time, these applications require a very low latency in order to be able to bring real-time decisions. The resulting requirements intensify computational strain and make real-time implementations a challenging research objective.

Many real-time semantic segmentation approaches [28, 40, 24, 32] address this goal by introducing custom lightweight architectures which are not suited for large-scale visual recognition. Most of these approaches initialize training from scratch, and thus miss a huge regularization

*equal contribution

opportunity offered by knowledge transfer [26] from larger and more diverse recognition datasets [30]. Consequently, these methods incur a comparatively large overfitting risk. Some approaches alleviate this shortcoming by pre-training on ImageNet [28]. However, our experiments suggest that the resulting benefits tend to be smaller than in architectures which are designed for competitive ImageNet performance.

A simple model for semantic segmentation starts with a fully convolutional encoder which gradually decreases the resolution and increases the number of feature maps of the resulting representation. Instead of performing global pooling (as we would do in image-wide classification) one can proceed by attaching a pixel-wise loss to obtain the predictions [6]. The resulting model would lead to a very fast evaluation on modern hardware, however its accuracy would be rather low due to the following problems. Firstly, small objects (e.g. distant traffic signs) would not be recognized due to low resolution of pixel-wise predictions, which is usually 32 times smaller than the input image. Secondly, the receptive field of such models would not be large enough to classify pixels at large objects (e.g. nearby buses or trucks). These problems can be alleviated with various techniques such as dilated convolutions [37], learned upsampling [20], lateral connections [27, 29, 19, 16] and resolution pyramids [6]. However, not all of these techniques are equally suited for real-time operation.

In this paper, we argue that a competitive blend of efficiency and prediction accuracy can be achieved by models based on lightweight ImageNet-grade classification architectures [8, 31]. Additionally, we propose a novel approach to increase the receptive field of deep model predictions, based on a resolution pyramid with shared parameters [6]. The proposed approach incurs a very modest increase of the model capacity and is therefore especially suited for datasets with large objects and few annotated images. Finally, we show that the resolution of the predictions can be efficiently and accurately upsampled by a lightweight decoder with lateral connections [27, 29]. We show that the resulting semantic segmentation models can be evaluated under various computing budgets, and that they are feasible even on embedded GPU platforms. We present experiments both with ImageNet pre-training and learning from scratch on different road driving datasets. Our experiments achieve state-of-the art semantic segmentation accuracy among all existing approaches aiming at real-time execution.

2. Related Work

As described in the introduction, semantic segmentation models have to face two major problems: restoring the input resolution and increasing the receptive field. The simplest way to restore input resolution is to avoid downsampling. This is usually achieved by replacing stride-2 poolings with non-strided poolings, and doubling the dilation factor in subsequent convolutions [4, 38]. However, this approach increases the resolution of deep latent representations, which implies a large computational complexity. Furthermore, dilated convolutions incur significant additional slow-down due to necessity to rearrange the image data before and after calling optimized implementations.

Another way to achieve dense image prediction relies on trained upsampling [20], which results in an encoder-decoder architecture. The idea is to perform recognition on subsampled latent representation to reduce complexity and then to restore the resolution by upsampling the representation (or the predictions). This setup can be naturally augmented by introducing lateral connections [27, 29, 19, 16] to blend semantically rich deep layers with spatially rich shallow layers. The upsampling path has to be as lean as possible (to achieve efficiency and prevent overfitting) but no leaner (to avoid underfitting). It turns out that the sweet spot is computationally inexpensive, which makes this setup especially well-suited for real-time operation.

Early approaches to enlarge the receptive field of logit activations were based on dilated convolutions [37, 4, 38]. A more involved approach is known as spatial pyramid pooling (SPP) [7]. SPP averages features over aligned grids with different granularities [17]. We use a convolutional adaptation of that idea, in which the feature pyramid is upsampled to the original resolution [41] and then concatenated with the input features. Thus, subsequent convolutions obtain access to broad spatial pools and that increases their receptive field. The combination of dilated convolutions and SPP is known as à trous SPP, or ASPP for short [3]. However, SPP and ASPP may hurt generalization due to large capacity. In this paper, we study resolution pyramids as an alternative way to increase the receptive field and at the same time promote scale invariance [34, 15, 18, 4]. Most previous pyramidal approaches to semantic segmentation [6, 40, 23] fuse only the deepest representations extracted at different resolutions. Different from them, we combine representations from different abstraction levels before joining the upsampling path within the decoder. This results in a better gradient flow throughout the pyramidal representation which is advantageous when the objects are large and the training data is scarce.

Efficient recognition architectures leverage optimized building blocks which aim at reducing computational load while preserving accuracy. Grouped convolutions reduce the number of floating point operations and the number of parameters by enclosing the information flow within smaller groups of feature maps. Various methods have been proposed to discover prominent inter-group connections. ShuffleNet [39] uses channel shuffling to pass information across convolutional groups. CondenseNet [11] incorporates a training strategy which locates important connections in grouped convolutions and prunes those which

are redundant. Neural architecture search [42] is an approach that leverages reinforcement learning to jointly learn the model architecture and the corresponding parameters. The resulting architectures achieve competitive ImageNet performance when the computational budget is restricted. Depthwise separable convolutions [33, 36] decrease computational complexity by splitting a regular convolution in two. Firstly, a $k \times k$ convolution is separably applied to each input channel. This can be viewed as a group convolution where the number of groups corresponds to the number of channels C. In other words, there are C kernels k×k×1. Secondly, a 1×1 convolution is applied to propagate inter-channel information. Replacing standard convolutions with depthwise separable convolutions lowers the number of parameters and increases the inference speed at the cost of some drop in performance [10]. Strong regularization effect of depthwise separable convolutions can be relaxed by inverted residual blocks [31] which lead to compact residual models suitable for mobile applications.

Most semantic segmentation approaches aiming at real-time operation refrain from using encoders designed for competitive ImageNet performance. ICNet [40] proposes a custom encoder which processes an image pyramid with shared parameters and fuses multi-scale representations before entering the decoder which restores the resolution. ERFNet [28] redefines a residual block as a composition of a 3×1 followed by a 1×3 convolution, which yields a 33% reduction in parameters. Vallurupalli et al. [35] propose the DG2s approach as an efficient ERFNet variant with the following modifications in residual blocks: i) depthwise separable convolutions, and ii) channel shuffling operation before pointwise convolutions. ESPNet [24] factorizes convolutions in a similar manner and refrains from shared parameters across the image pyramid in order to produce a fast and compact architecture.

Our method is most related to semantic segmentation approaches which use lightweight encoders trained on ImageNet and benefit from such initialization. Similar to our work, Nekrasov et al. [25] rely on MobileNet V2 [31] and NASNet [42] encoders and feature a thick decoder with lateral connections. This is similar to our single scale model, however, our decoder has much less capacity, which allows us to report a half of their number of floating point operations without sacrificing recognition accuracy on road driving datasets. LinkNet [2] uses a small ResNet-18 backbone and a lightweight decoder to achieve satisfying performance-speed ratio. Our single scale model is similar to LinkNet, however we omit convolutional layers at full resolution in order to substantially reduce memory imprint and greatly increase the processing speed. Mazzini et al. [23] use a dilated encoder initialized from the DRN-D-22 model [38], and a decoder with one lateral connection. They also learn nonlinear upsampling to improve accuracy at object boundaries. Instead of using dilated convolutions, our decoder upsamples predictions by exclusively relying on lateral connections, which results in a 4-fold speed-up. Additionally, we succeed to leverage full resolution images during training which achieves an improvement of 5 percentage points on Cityscapes test.

3. The proposed segmentation method

Our method assumes the following requirements. The model should be based on an ImageNet pre-trained encoder in order to benefit from regularization induced by transfer learning. The decoder should restore the resolution of encoded features in order for the predictions to retain detail. The upsampling procedure must be as simple as possible in order to maintain real-time processing speed. Gradient flow should be promoted throughout the network to support training from random initialization in an unusual event that ImageNet pre-training turns out not to be useful.

3.1. Basic building blocks

The proposed segmentation method is conceived around three basic building blocks which we briefly describe in the following paragraphs. These building blocks are going to be used in our two models which we propose in subsequent subsections.

Recognition encoder We advocate the usage of compact ImageNet pre-trained model as segmentation encoders. We propose to use ResNet-18 [8] and MobileNet V2 [31] for a number of reasons. These models are a good fit for fine tuning due to pre-trained parameters being publicly available. They are also suitable for training from scratch due to moderate depth and residual structure. Finally, these models are compatible with real-time operation due to small operation footprint. Computationally, ResNet-18 is around six times more complex than MobileNet V2. However, MobileNet V2 uses depthwise separable convolutions which are not directly supported in GPU firmware (the cuDNN library). Therefore, MobileNet V2 tends to be slower than ResNet-18 in most experimental setups. Note that the same issue disqualifies usage of the DenseNet architecture [12], since it requires efficient convolution over a non-contiguous tensor, which is still not supported in cuDNN.

Upsampling decoder The recognition encoder transforms the input image into semantically rich visual features. These features must have a coarse spatial resolution in order to save memory and processing time. The purpose of the decoder is to upsample these features to the input resolution. We advocate a simple decoder organized as a sequence of upsampling modules with lateral connections [27, 29]. The proposed ladder-style upsampling modules have two

inputs: the low resolution features (which should be upsampled), and the lateral features from an earlier layer of the encoder. The low resolution features are first upsampled with bilinear interpolation to the same resolution as the lateral features. Upsampled input features and lateral encoder features are then mixed with elementwise summation and finally blended with a 3×3 convolution.

We propose to route the lateral features from the output of the elementwise sum within the last residual block at the corresponding level of subsampling, as shown in Figure 2. Note that routing the lateral features from the output of the subsequent ReLU leads to a considerable drop in validation accuracy. Replacing the standard 3×3 convolution with either a 1×1 convolution, or a depthwise separable convolution also decreases the validation accuracy.

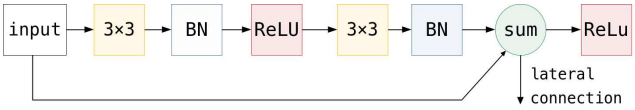

Figure 2. Structural diagram of the last residual unit within a convolutional block operating on common spatial resolution. We do not use pre-activation [9] since we could not find a pre-trained parameterization for ResNet-18. The lateral connection is taken from the output of the elementwise sum after the last residual block. The output of the ReLU node is forwarded to the next residual block.

Module for increasing the receptive field As discussed before, there are two viable possibilities for increasing the receptive field while maintaining real-time speed: i) spatial pyramid pooling, and ii) pyramid fusion. The SPP block gathers the features produced by the encoder at several pooling levels and produces a representation with a varying level of detail. We demonstrate the use of SPP in our single scale model. Our SPP block is a simplified version of the pyramid pooling module from PSPNet [41]. The pyramid fusion produces genuine multi-scale representations which need to be carefully fused within the decoder in order to avoid overfitting to unsuitable level of detail. We propose a pyramid pooling approach which blends representations at different levels of abstraction and thus enlarges the receptive field without sacrificing spatial resolution.

3.2. Single scale model

The proposed single scale model transforms the input image into dense semantic predictions throughout a downsampling recognition encoder and upsampling decoder, as shown in Figure 3. Yellow trapezoids designate convolution groups, that is, parts of the encoder which operate on the same spatial resolution. All considered encoders consist of four such convolution groups. The first convolution group produces features at the H/4×W/4 resolution, while each following group increases the subsampling by the factor of 2. Thus the features at the far end of the encoder are H/32×W/32. These features are fed into the spatial pyramid pooling layer (designated by a green diamond) in order to increase the model receptive field. The resulting tensor is finally routed to the decoder whose upsampling modules are shown in blue.

Note that decoder and encoder are asymmetric: the encoder has many convolutions per convolution group while decoder has only one convolution per upsampling module. Furthermore, the dimensionality of encoder features increases along the downsampling path, while the dimensionality of the decoder features is constant. Therefore, lateral connections have to adjust dimensionality with 1×1 convolutions designated with red squares. Upsampling modules operate in three steps: i) the low resolution representation is bilinearly upsampled, ii) the obtained representations is summed with the lateral connection, iii) the summation is blended using a 3×3 convolution.

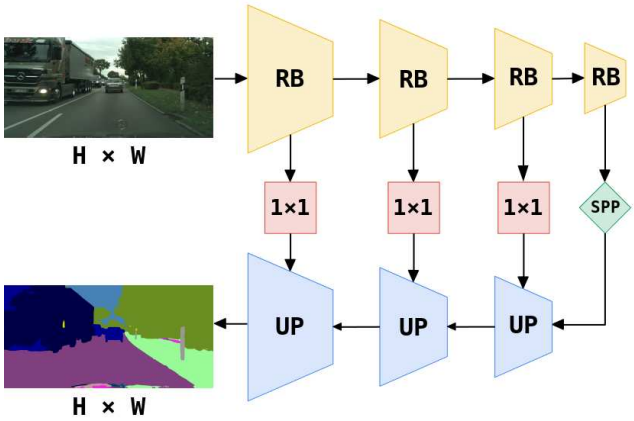

Figure 3. Structural diagram of the proposed single scale model. Yellow trapezoids designate convolution groups within the encoder which may be pre-trained on ImageNet. The green diamond designates the spatial pyramid pooling layer, the red squares designate bottleneck layers, and blue trapezoids designate lightweight upsampling modules. Logits are upsampled to original image resolution with bilinear interpolation.

3.3. Interleaved pyramid fusion model

While using a compact encoder is beneficial for fast inference, this also results in a decreased receptive field and a smaller capacity compared to general purpose convolutional models for visual recognition. To counteract these drawbacks, we propose to exploit image pyramids to enlarge the receptive field of the model and reduce the model capacity requirements.

The proposed model is shown in Figure 4. Two encoder instances (yellow) are applied to the input image at different levels of the resolution pyramid. This results in increased receptive field of the activations which sense the lowest res-

olution of the image pyramid. Furthermore, shared parameters enable recognition of objects of different sizes with the common set of parameters, which may relax the demand for model capacity. In order to enforce lateral connections and improve the gradient flow throughout the encoder, we concatenate the feature tensors from neighbouring levels of different encoders (we can do that since they have equal spatial resolution). This concatenation is designated with green circles. After concatenation, interleaved encoder features are projected onto the decoder feature space by 1×1 convolutions designated with red squares. The decoder (blue) operates in the same manner as in the single-scale model, however now we have an additional upsampling module for each additional level of the image pyramid.

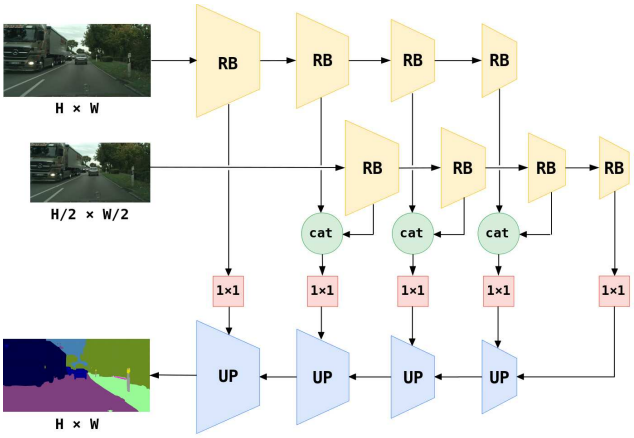

Figure 4. Structural diagram of the proposed model with interleaved pyramidal fusion. Encoder parameters (yellow) are shared across all pyramid levels and may be pre-trained on Imagenet. Features of the same resolutions are concatenated (green circles), fed into a 1×1 bottleneck convolution (red squares) and blended within the decoder (blue).

4. Experiments

We conduct semantic segmentation experiments on two datasets: Cityscapes [5] and CamVid [1]. We report mIoU accuracy, computational complexity and the execution speed of the trained models. The speed measurements are performed on a desktop GPU (GTX 1080Ti) and on an embedded System on a chip module (Jetson TX2). We also present ablation and validation experiments which provide a more detailed insight into the impact of various design choices. Please note that additional experiments can be found in the supplement.

4.1. Training details

We train all our models with the Adam [14] optimizer with the learning rate set to $4 \cdot 10^{-4}$. We decay the learning rate with cosine annealing [21] to the minimum value of $1 \cdot 10^{-6}$ in the last epoch (we do not perform any warm restarts). The weight decay is set to $1 \cdot 10^{-4}$. In experiments with ImageNet pre-training, we update pre-trained parameters with 4 times smaller learning rate and apply 4 times smaller weight decay. We train on jittered square crops with batch size 12. The jittering consists of random horizontal flipping, and scaling with random factors between 0.5 and 2. We use 768×768 crops for full Cityscapes resolution, and 448×448 crops for half Cityscapes resolution and CamVid. We train for 200 epochs on Cityscapes and 400 epochs on CamVid. We train for additional 200 epochs in experiments without ImageNet pre-training.

4.2. Measuring the computational complexity

We express the computational complexity of the trained models with two metrics: i) billions of floating point operations (GFLOP), and ii) number of processed frames per second (FPS). The GFLOP metric provides the number of fused multiply-add operations required to evaluate the model. Such platform-agnostic measure of the computational complexity is suitable for CPUs where all multiplications require roughly equal processing time. Unfortunately, the GFLOP metric poorly corresponds with the actual processing time on GPU platforms, since efficient implementations are available only for a small subset of all building blocks used to express current deep models. Consequently, both metrics are required for a complete description of algorithm suitability for real-time operation.

The FPS metric directly corresponds to the processing time on a particular hardware platform. Such metric does not necessarily correlate across platforms, although rough estimations can be done, as we show below. We simulate real-time applications by setting batch size to 1. We measure the time elapsed between transferring the input data to the GPU, and receiving the semantic predictions into RAM as shown in the code snippet shown in Figure 5.

```
device = torch.device('cuda')
model.eval()
model.to(device)
with torch.no_grad():
  input = model.prepare_data(batch).to(device)
  logits = model.forward(input)
  torch.cuda.synchronize()
  t0 = 1000 * perf_counter()
  for _ in range(n):
    input = model.prepare_data(batch).to(device)
      logits = model.forward(input)
      _, pred = logits.max(1)
      out = pred.data.byte().cpu()
      torch.cuda.synchronize()
  t1 = 1000 * perf_counter()
fps = (1000 * n) / (t1 - t0)
```

Figure 5. Measurement of the processing time under PyTorch.

We conduct all measurements on a single GTX1080Ti with CUDA 10.0, CUDNN 7.3 and PyTorch 1.0rc1. We exclude the batch normalization layers [13] from measurements since in real-time applications they would be fused with preceding convolutional layers. We report mean FPS values over 1000 forward passes. Results are shown in Table 1. The column *FPS norm* provides a rough estimate on how would other methods perform on our hardware. The scaling factors are: 1.0 for GTX1080Ti, 0.61 for TitanX Maxwell, 1.03 for TitanX Pascal, and 1.12 for Titan XP. These scaling factors were calculated using publicly available benchmarks: `goo.gl/N6ukTz`, `goo.gl/BaopYQ`. The column *GFLOPs@1MPx* shows an estimated number of FLOPs for an input image of 1MPx, as a resolution-agnostic metric of computational complexity.

4.3. Cityscapes

The Cityscapes dataset is a collection of high resolution images taken from the driver's perspective during daytime and fine weather. It consists of 2975 training, 500 validation, and 1525 test images with labels from 19 classes. It also provides 20000 coarsely labeled images which we do not use during our experiments. Table 1 evaluates the accuracy (class mIoU) and efficiency (GFLOP, FPS) of our methods and compares them to other real-time methods. Our single scale method based on the ResNet-18 encoder achieves 75.5% test mIoU, and delivers 39.9 FPS on full Cityscapes resolution (1024×2048 pixels). To the best of our knowledge, this result outperforms all other approaches aiming at real-time operation. The corresponding submission to the Cityscapes evaluation server is entitled SwiftNetRN-18. Table 1 also presents experiments in which our models are trained from scratch. The accuracy decreases for 5 mIoU percentage points (pp) with respect to the corresponding experiments with ImageNet pre-trained initialization. This shows that ImageNet pre-training represents an important ingredient for reaching highly accurate predictions. We notice that custom encoders like ERFNet [28] get less benefits from ImageNet pre-training: only 1.7% pp as shown in Table 1. Figure 7 presents examples of segmentations on Cityscapes val images. We show examples for both single scale and pyramid models. We did not achieve measurable improvements with the pyramid model over the single scale model on the Cityscapes dataset.

4.4. CamVid

The CamVid dataset contains 701 densely annotated frames. We use the usual split into 367 train, 101 validation and 233 test images. We train on combined train and validation subsets and evaluate semantic segmentation into 11 classes on the test subset. Table 2 shows that we obtain an improvement of roughly 1.5 pp mIoU when using the pyramid model with pre-trained ResNet-18 and MobileNetV2 backbones. Figure 8 shows frames from the CamVid test subset where the pyramid model performed better.

Table 2 also shows that ImageNet pre-training contributes more on CamVid than on Cityscapes (7-9pp of mIoU performance). This is not surprising since CamVid has almost 20 times less training pixels.

A small size of the dataset poses a considerable challenge when training from scratch due to high overfitting risk. Table 2 shows that the pyramid model achieves better

model	subset	mIoU	FPS	FPS norm	GPU	resolution	GFLOPs	GFLOPs@1Mpx	# params
D* [35]	val	68.4	-	-	TitanX M	1024x512	5.8	11.6	0.5M
DG2s [35]	val	70.6	-	-	TitanX M	1024x512	19.0	38	1.2M
Ladder DenseNet†[16]	val	72.8	31.0	30.1	TitanX	1024x512	-	-	9.8M
ICNet [40]	test	69.5	30.3	49.7	TitanX M	2048x1024	-	-	-
ESPNet [24]	test	60.3	112	108.7	TitanX	1024x512	-	-	0.4M
ERFNet [28]	test	68.0	11.2	18.4	TitanX M	1024x512	27.7	55.4	20M
GUNet†[23]	test	70.4	37.3	33.3	TitanXP	1024x512	-	-	-
ERFNet†[28]	test	69.7	11.2	18.4	TitanX M	1024x512	27.7	55.4	20M
SwiftNetRN-18	val	70.4	39.9	39.3	GTX 1080Ti	2048x1024	104.0	52.0	11.8M
SwiftNetMN V2	val	69.4	27.7	27.7		2048x1024	41.0	20.5	2.4M
SwiftNetRN-18†	val	70.2	134.9	134.9	GTX 1080Ti	1024x512	26.0	52.0	11.8M
SwiftNetRN-18 pyr†	val	74.4	34.0	34.0		2048x1024	114.0	57.0	12.9M
SwiftNetMN V2†	val	75.3	27.7	27.7		2048x1024	41.0	20.5	2.4M
SwiftNetRN-18†	val	75.4	39.9	39.3		2048x1024	104.0	52.0	11.8M
SwiftNetRN-18 pyr†	test	75.1	34.0	34.0	GTX 1080Ti	2048x1024	114.0	57.0	12.9M
SwiftNetRN-18†	test	**75.5**	39.9	39.3		2048x1024	104.0	52.0	11.8M
SwiftNetRN-18 ens†	test	**76.5**	18.4	18.4		2048x1024	218.0	109.0	24.7M

Table 1. Results of semantic segmentation on Cityscapes. We evaluate our best result on the online test benchmark and compare it with relevant previous work, where possible. We also report the computational complexity (GFLOP, FPS) GPU on which the inference was performed, and the image resolution on which the training and inference were performed. The column *GFLOPs@1Mpx* shows the GFLOPs metric when the input resolution is 1MPx. The column FPS norm shows or estimates the FPS metric on GTX 1080Ti. The default SwiftNet configuration is the single scale model presented in 3.2. Label pyr denotes the pyramid fusion model presented in 3.3. Label ens denotes the ensemble of the single scale model and the pyramid model. The symbol † designates ImageNet pre-training.

results than the single scale model. This supports our choice of sharing encoder parameters across pyramid levels.

backbone	model	mIoU†	mIoU
ResNet-18	single scale	72.58	63.33
	pyramid	73.86	65.70
MobileNet V2	single scale	71.56	64.01
	pyramid	73.08	65.01

Table 2. Semantic segmentation accuracy on CamVid test using ImageNet pre-training (mIoU†) and training from scratch (mIoU).

4.5. Validation of the upsampling capacity

The number of feature maps along the upsampling path is the most important design choice of the decoder. We validate this hyper-parameter and report the results in Table 3. The results show that the model accuracy saturates at 128 dimensions. Consequently, we pick this value as a sensible speed-accuracy trade-off in all other experiments.

model	upsampling features	mIoU
SwiftNetRN-18	64	69.50
	128	70.35
	192	70.26
	256	70.63

Table 3. Validation of the number of feature maps in the upsampling path. The models were trained on Cityscapes train subset at 512×1024 while the evaluation is performed on Cityscapes val. All models use ImageNet initialization.

4.6. Ablation of lateral connections

To demonstrate the importance of lateral connections between the encoder and the decoder, we train a single scale model without lateral connections. For this experiment, we discard the 1×1 convolution layers located on the skip connections and abandon the elementwise summations in upsampling modules. Training such a model on full Cityscapes train images causes the validation accuracy to drop from 75.35% to 72.93%.

4.7. Execution profile

To obtain a better insight into the execution time of our models, we report separate processing times and the GFLOP metrics for the downsampling path (encoder and SPP), and the upsampling path (decoder). Table 4 shows the results for the single scale model and input resolution of 2048×1024. Note the striking discrepancy of time and GFLOPs for the two downsampling paths. ResNet-18 is almost twice as fast than MobileNet v2 despite requiring 6 times more multiplications. Note also that our decoder is twice as fast as the ResNet-18 encoder.

model	dn time	up time	dn FLOPs	up FLOPs
RN-18	15.0ms	8.1ms	76.1B	30.9B
MN-V2	26.7ms	7.5ms	12.1B	28.9B

Table 4. Inference speed along the downsampling (encoder and SPP) and the upsampling (decoder) paths for the single scale model. The columns *dn time* and *up time* display the execution times, while the columns *dn FLOPs* and *up FLOPs* show the number of floating point operations for 2048×1024 images.

4.8. Size of the receptive field

We estimate the effective receptive field of our models by considering the central pixel in each image $\mathbf{X}$ of Cityscapes val. The estimate is based on gradients $\frac{\partial y_i}{\partial \mathbf{X}}$ [22] where $\mathbf{y}$ are the logits for the central pixel while i is $argmax(\mathbf{y})$. Table 5 expresses the effective receptive fields as standard deviations of pixels with top 5% gradient $\frac{\partial y_i}{\partial \mathbf{X}}$. The results show that both SPP and interleaved pyramidal fusion substantially increase the receptive field.

model	ERF horizontal	ERF vertical
RN-18 no SPP	84.47	84.78
RN-18 SPP	154.07	**153.55**
RN-18 pyramid	**185.22**	140.24

Table 5. Effective receptive fields (ERF) expressed as standard deviation of pixels with top 5% image gradients with respect to the dominant class of the central pixel, measured on Cityscapes val.

4.9. Processing speed on Jetson TX2

We evaluate the proposed methods on NVIDIA Jetson TX2 module under CUDA 9.0, CUDNN 7.1, and PyTorch v1.0rc1. Due to limited number of CUDA cores, all bilinear interpolations had to be replaced with nearest neighbour interpolation. Results are reported in Figure 6. The MobileNet V2 backbone outperforms ResNet-18 for 20-30% on most resolutions due to lower number of FLOPs. However, ResNet-18 is faster on the lowest resolution. Note that our implementations do not use TensorRT optimizations.

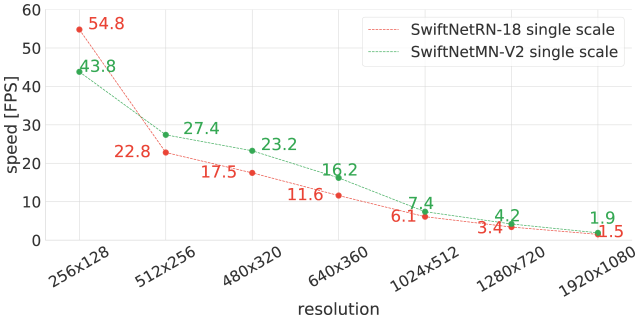

Figure 6. Processing speed in frames per second of the proposed architecture on NVIDIA Jetson TX2 module for two different backbones and various input resolutions.

5. Conclusion

Real-time performance is a very important trait of semantic segmentation models aiming at applications in robotics and intelligent transportation systems. Most previous work in the field involves custom convolutional encoders trained from scratch, and decoders without lateral skip-connections. However, we argue that a better speed-accuracy trade-off is achieved with i) compact encoders designed for competitive ImageNet performance and ii) lightweight decoders with lateral skip-connections. Additionally, we propose a novel interleaved pyramidal fusion scheme which is able to further improve the results on large objects close to the camera. We provide a detailed analysis of prediction accuracy and processing time on Cityscapes and CamVid datasets for models based on ResNet-18 and MobileNetv2. Our Cityscapes test submission achieves 75.5% mIoU by processing 1024×2048 images at 39.9 Hz on a GTX1080Ti. To the best of our knowledge, this result outperforms all previous approaches aiming at real-time application. The source code is available at `https://github.com/orsic/swiftnet`.

Acknowledgment

This work has been supported by the European Regional Development Fund under the project System for increased driving safety in public urban rail traffic (SafeTRAM) under grant KK.01.2.1.01.0022, and by European Regional Development Fund (DATACROSS) under grant KK.01.1.1.01.0009, and Microblink Ltd. We would like to thank Josip Krapac for helpful discussions. The Titan Xp used to train some of the evaluated models was donated by NVIDIA Corporation.

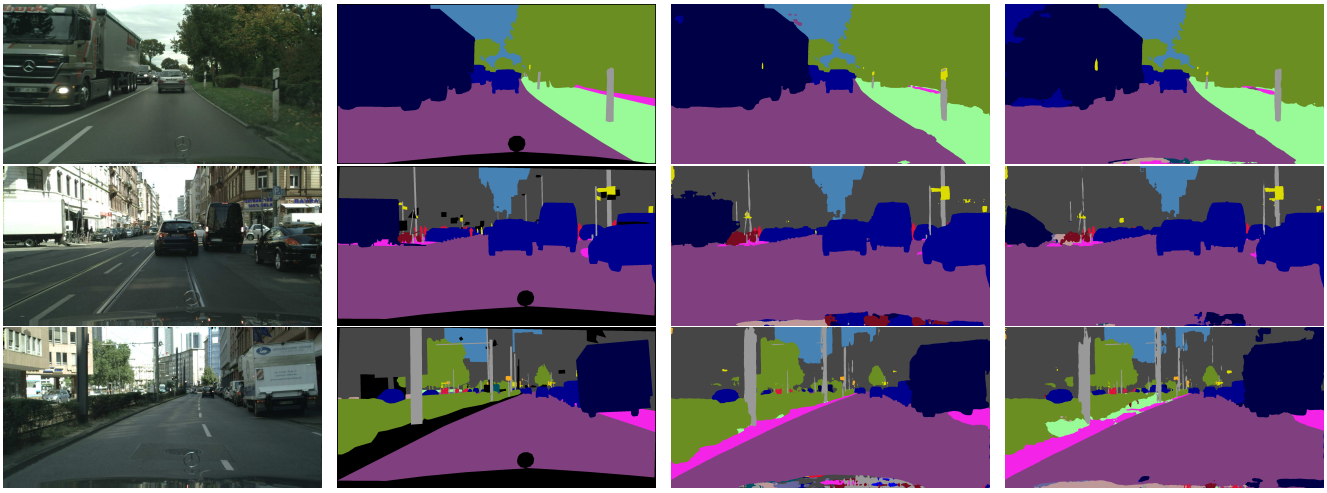

Figure 7. Semantic segmentation results on Cityscapes val. The columns correspond to input image, ground truth annotation, the output of the pyramid model, and the output of the single scale model. The most significant improvements occur on pixels of the class truck.

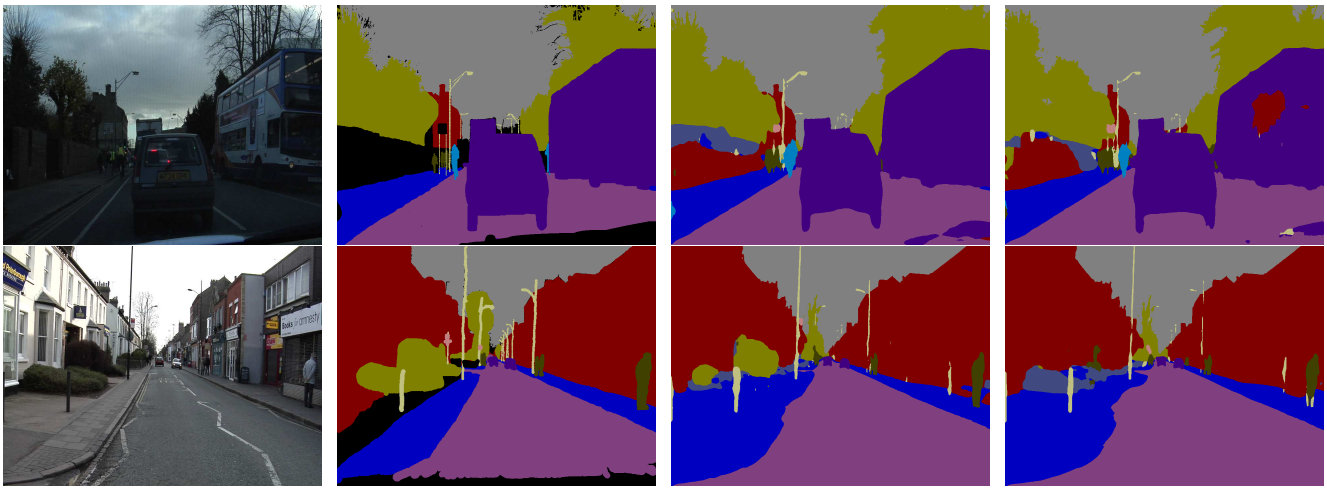

Figure 8. Semantic segmentation results on CamVid test. The columns correspond to input, ground truth, the output of the pyramid model, and the output of the single scale model. The most significant improvements occur on pixels of classes bus (top) and tree (bottom).

References

[1] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx, 2008.

[2] A. Chaurasia and E. Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing, VCIP 2017, St. Petersburg, FL, USA, December 10-13, 2017*, pages 1–4, 2017.

[3] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[4] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3640–3649, 2016.

[5] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *CVPRW*, 2015.

[6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *PAMI*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[11] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. *arXiv preprint arXiv:1711.09224*, 2017.

[12] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269, 2017.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] I. Kreso, D. Causevic, J. Krapac, and S. Segvic. Convolutional scale invariance for semantic segmentation. In *GCPR*, pages 64–75, 2016.

[16] I. Kreso, J. Krapac, and S. Segvic. Ladder-style densenets for semantic segmentation of large natural images. In *ICCVW CVRSUAD*, pages 238–245, 2017.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006.

[18] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5168–5177, 2017.

[19] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[22] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NIPS*, pages 4898–4906, 2016.

[23] D. Mazzini. Guided upsampling network for real-time semantic segmentation. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 117, 2018.

[24] S. Mehta, M. Rastegari, A. Caspi, L. G. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, pages 561–580, 2018.

[25] V. Nekrasov, C. Shen, and I. D. Reid. Light-weight refinenet for real-time semantic segmentation. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 125, 2018.

[26] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1717–1724, 2014.

[27] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3546–3554, 2015.

[28] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, pages 234–241, 2015.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[31] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[32] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, H. Zhang, N. Vallurupalli, S. Annamaneni, G. Varma, C. Jawahar, et al. A comparative study of real-time semantic segmentation for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 587–597, 2018.

[33] L. Sifre and S. Mallat. Rigid-motion scattering for texture classification. *CoRR*, abs/1403.1687, 2014.

[34] B. Singh and L. S. Davis. An analysis of scale invariance in object detection–snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3587, 2018.

[35] N. Vallurupalli, S. Annamaneni, G. Varma, C. Jawahar, M. Mathew, and S. Nagori. Efficient semantic segmentation using gradual grouping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[36] M. Wang, B. Liu, and H. Foroosh. Factorized convolutional neural networks. In *ICCV Workshops*, pages 545–553, 2017.

[37] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[38] F. Yu, V. Koltun, and T. A. Funkhouser. Dilated residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 636–644, 2017.

[39] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[40] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. *arXiv preprint arXiv:1704.08545*, 2017.

[41] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *ICCV*, 2017.

[42] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.