In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-driving Images - Supplementary

1. Lighter upsampling

As mentioned in the paper, we have experimented by replacing the 3×3 convolution of the upsampling module using a variety of lighter alternatives. Here we show our results using the following variants: grouped convolution, depthwise separable convolution, inverted residual block, 1×1 convolution. We replaced the 3×3 convolution in the upsampling module and kept the rest of the model and training specification the same. Results are reported in Table 1. ResNet-18 was used as the model encoder. All of the models were trained on Cityscapes train subset. mIoU is measured on Cityscapes val. The input image size is 2048×1024. The number of floating point operations is shown for the same input size. Also, the total number of randomly initialized parameters is reported (all parameters not trained on ImageNet). For comparison, when using a 3×3 convolutions in the upsampling, the total number of randomly initialized parameters is 636 thousand at 104 GFLOPs.

Method	mIoU	GFLOPs	params
grouped conv (8 groups)	75.73	84.1	252K
depthwise separable	73.76	84.0	250K
inverted residual	74.87	92.7	405K
1×1 conv	74.18	83.8	247K
3×3 conv	75.35	104	636K

Table 1. Evaluation of convolution alternatives in the upsampling path of a model with a ResNet-18 encoder. The *GFLOPs* column shows the number of floating point operations when the input image is full Cityscapes resolution, i.e. 2048×1024. The *params* column displays the number of parameters in the decoder path (without ImageNet pre-trained parameters).

On CamVid, we used the MobileNet V2 encoder to see the effect of using lighter convolution alternatives. Table 2 shows effects of using lighter convolution alternatives in combination with the MobileNet V2 encoder. The models were trained on CamVid train subset and the reported results are on CamVid val subset. In these experiments we witnessed drops in accuracy for all methods except for grouped convolution. However, we chose not to use grouped convolution instead of 3×3 convolutions in the upsampling path as the full 3×3 convolution yields faster inference time despite having more GFLOPs. This is due to 3×3 convolution having more optimized library support inside cuDNN.

Method	mIoU	GFLOPs	params
grouped conv(8 groups)	72.83	5.9	180K
depthwise separable	71.13	5.8	178K
inverted residual	71.98	8.6	332K
1×1 conv	71.69	5.7	174K
3×3 conv	72.05	13.1	567K

Table 2. Evaluation of convolution alternatives in the upsampling path of a model with MobileNet V2 encoder. The *GFLOPs* column shows the number of floating point operations when the input image is full CamVid resolution, i.e. 960×720. *params* column displays the number of parameters in the decoder path (without ImageNet pre-trained parameters).

2. Profiling on the Jetson TX2

Nearest neighbour upsampling. We have found out that PyTorch models with bilinear interpolation can not be implemented on the Jetson TX2 SoC due to exhaustion of CUDA cores. Instead, we had to use nearest neighbour interpolation. To see the effect of replacing the bilinear interpolation using nearest neighbour, we trained the single scale model with the aforementioned change in the upsampling path. Using nearest neighbour, we measured a drop from 75.35% to 73.69% mIoU. Both models were trained on Cityscapes train and the reported accuracy is measured on Cityscapes val.

Half precision. Jetson TX2 SoC supports half precision floating point operations. These may be beneficial for decreasing the memory throughput as well as raising inference speed. We measured the impact of using 16 bit precision on segmentation accuracy and witnessed drops of around 0.01% for various trained models. We also studied the effect of using half precision accuracy on inference speed. Compared to using 32 bit floating point arithmetic, we measured a speed-up for both ResNet-18 and MobileNet V2 backbones. These speed-ups were spanning from 10 to 15%. All our experiments were performed under CUDA 9.0, CUDNN 7.1, and PyTorch v1.0rc1. We expect that expressing our models in TensorRT would result in greater speed-ups. We visualize differences in inference speeds in

Figure 1 for the ResNet-18 backbone, and in Figure 2 for the MobileNet V2 backbone.

Runtime measurements. We report measured inference speed for all methods used in the paper. This means using MobileNet V2 and ResNet-18 as encoders and single scale and pyramid decoders. Reported inference speeds are shown in Figure 3.

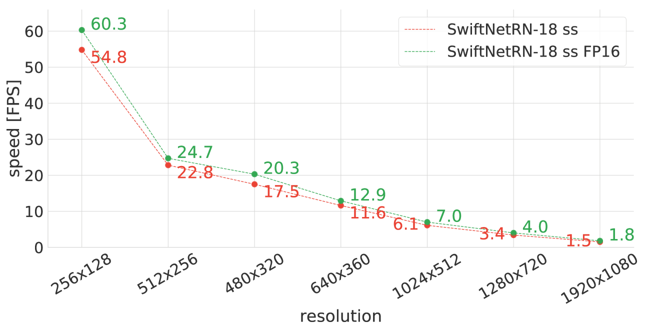

Figure 1. Comparison of inference speeds on the Jetson TX2. The same model, ResNet-18 single scale model, was ran using 32 and 16 bit floating point precision.

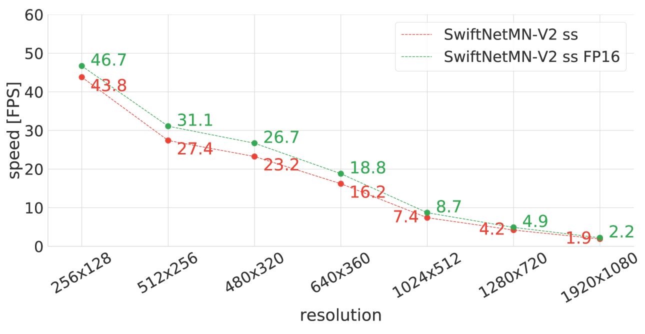

Figure 2. Comparison of inference speeds on the Jetson TX2. The same model, MobileNet V2 single scale model, was ran using 32 and 16 bit floating point precision.

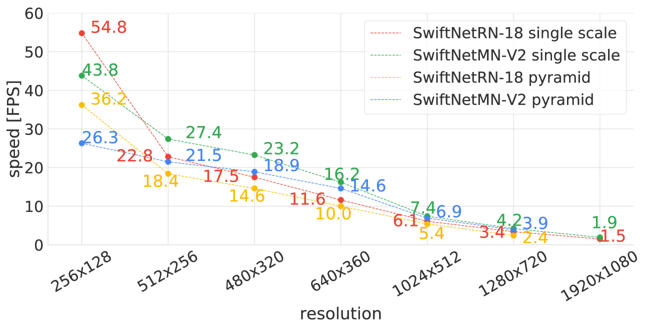

Figure 3. Inference speed of both pyramid and single scale methods for ResNet-18 and MobileNet V2 backbones. Reported FPS are measured using single precision floating point arithmetic.

3. Additional ablation and validation

Benefits of SPP To demonstrate the importance of SPP, we train a single scale model without it, which is very similar to FPN. That model achieves 71.47% mIoU on Cityscapes val, a 4 percent point drop compared to our single scale model with SPP.

Impact of ImageNet init Table 3 presents ablation experiments which explore benefits of ImageNet pre-training. In each experiment, we use ImageNet init in one additional residual block to find out where this regularization helps the most.

Conv-BN 1	RB 1	RB 2	RB 3	RB 4	mIoU
					68.50
✓					69.21
✓	✓				69.98
✓	✓	✓			71.74
✓	✓	✓	✓		73.08
✓	✓	✓	✓	✓	**75.35**

Table 3. Impact of partial pre-training on Cityscapes val. Checked residual blocks were initialized on ImageNet. All models were trained through 250 epochs on Cityscapes train at full resolution.

More pyramid levels A pyramid model with 3 pyramid levels achieves 74.9% mIoU on Cityscapes val mIoU, around 0.5pp drop with respect to the 2 level pyramid.

Single scale vs pyramid To find differences between single scale and pyramid models, we measure segmentation accuracy with respect to stereoscopic disparity. Although the pyramid and single scale model perform the same on Cityscapes val, we find that on 5% of closest pixels, the pyramid model is 1.4% more accurate. On 5% of the farthest pixels, the pyramid model performs 3.2% worse.

Ensembling pyramid and single scale models We compare the single scale SPP model with the pyramid model by evaluating the following two ensembles on Cityscapes val: i) one SPP model and one pyramid model, and ii) two SPP models. Table 4 shows that most improvement over the single SPP model is achieved with a heterogeneous ensemble. This indicates that the two approaches learn different representations.

models	first	second	ensemble
SPP1 + SPP2	75.35	75.43	76.45
pyramid + SPP1	75.45	75.35	77.12
pyramid + SPP2	75.45	75.43	**77.29**

Table 4. Experiments of segmentation mIoU of ensembled models on Cityscapes val. The combination of one SPP model and one pyramid model surpasses the ensemble of two SPP models.

Summary of Cityscapes results Table 5 briefly presents all of our Cityscapes val results for the single scale model.

backbone	res	IN	mIoU	GFLOPS	FPS
RN-18	full	✓	**75.4**	104.0	39.9
MN V2			75.3	41.0	27.7
RN-18	full		70.4	104.0	39.9
MN V2			69.4	41.0	27.7
RN-18	half	✓	70.2	26.0	**134.9**
MN V2			70.6	**10.3**	95.1

Table 5. Our Cityscapes val results expressed in *mIoU*. *res* column shows wether the input resolution is 2048×1024(full) or 1024×512(half). *IN* indicates ImageNet pre-training. *GFLOPS* and *FPS* show the total number of floating point operations and number of processed frames per second on GTX 1080Ti.

Multi-scale inference We demonstrate the effectiveness of using a resolution pyramid at test time. For multi-scale inference we employ the following approach. We create a resolution pyramid of input images by re-scaling the original image by the following factors $[0.5, 0.75, 1.0, 1.25, 1.5]$. Next, we perform forward passes on each of the images and resize the probability maps to original resolution. The final output tensor is an average value of all five outputs. The results from Table 6 show that both the single scale and pyramid model benefit for multi-scale inference.

model	mIoU	mIoU $\triangle$
SwiftNetRN-18 SPP	75.4	76.3
SwiftNetRN-18 pyramid	75.5	76.4

Table 6. Results on Cityscapes val with and without multi-scale inference. *mIoU* $\triangle$ shows results when performing multi-scale inference. *mIoU* shows results when using a single image as input.