Binary Quantization for Semantic Segmentation

Geraerds, M.F.A.M.

Award date:
2021

Report of the Graduation Project

Project phase

Binary Quantization for Semantic Segmentation

Master:	Automotive Technology
Department:	Mechanical Engineering
Research group:	Signal Processing Systems

Student:	M.F.A.M. Geraerds
Identity number:	0715419
Thesis supervisor:	P. Jancura & F. de Putter
Date:	03-06-2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

03/06/2021

Name

M.F.A.M. Geraerds

ID-number

0715419

Signature

[illegible]

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Binary Quantization for Semantic Segmentation

Graduation project paper [DRAFT]

M.F.A.M. Geraerds
Department of Mechanical Engineering
Eindhoven University of Technology
Eindhoven, Netherlands
m.f.a.m.geraerds@student.tue.nl

***Abstract*—Semantic segmentation is a vital task for self-driving vehicles. Current state-of-the-art neural networks are able to attain sufficient accuracy, but are computationally expensive in terms of required energy and memory storage for real-time inference on embedded devices. To allow these networks to be used on hardware that can be placed in vehicles, the networks will have to be compressed. Binary quantization is one of the avenues that has been researched to obtain this compression. The majority of contemporary state-of-the-art methods for quantization of neural networks have been focused on classification tasks. The research described in this paper shows how the methodology used in ReActNet can be extended to perform a semantic segmentation task. This is achieved by quantizing the DeepLabv3 neural network architecture, as well as FCN32 and FCN8 decoder structures. The resulting performance on cityscapes is shown to be on par with Group-Net, without the need to parallelize the network into multiple branches. As a result the network has roughly a quarter of the parameters of Group-Net, while achieving comparable performance.**

I. Introduction

Semantic segmentation is among the most vital challenges in the development of autonomously driving vehicles. By enabling a vehicle to obtain a complete understanding of its surroundings at a pixel level, the vehicles computer can accurately identify lanes, determine which way to go, when to brake, and perform various other tasks required for autonomous driving.
In order for a neural network to run in real time, significant computing power is required. Current state of the art networks require a Graphics Processing Unit (GPU) to run and a significant amount of memory to store the network parameters. To be able to use these networks without excessive hardware requirements for energy-constrained applications, such as self-driving vehicles, quantization is used to reduce the size of the weights and activations by converting them from 32-bit floating point values to e.g. 8- or 5-bit integer values. This conversion decreases the memory required to load the network, as well as the computational complexity, as it replaces the computationally heavy 32-bit multiply-accumulate (MAC) operations with less complex integer MAC operations.
Binary quantization takes the quantization process a step further and replaces the 32-bit floating point values with binary values set to either -1 or +1. One of the major challenges in quantization is the trade off between accuracy and computational efficiency. Binary arithmetic is less complex, and thus faster, but 1-bit values only indicate whether a weight is larger than or smaller than a given base value. To resolve this issue, several techniques have been proposed that aim to reduce the quantization error while maintaining the high speed of a binary network.
While binary quantization has been applied to several classification networks, not many papers have been published that research the application of binary quantization on networks for semantic segmentation.

A. Problem Definition

The purpose of this paper was to investigate the viability of binarized neural networks for semantic segmentation tasks. Two implementations of binarized neural networks have been made that are able to perform semantic segmentation tasks on the cityscapes dataset. The classification backbone of both networks is based on the ReActNet paper [1]. The baseline is given by the Group-Net semantic segmentation network [2].
To adapt the existing classification networks for semantic segmentation, the ReActNet architectures can be extended to the decoder and atrous spatial pyramid pooling (ASPP) of DeepLabV3 [3]. This architecture is chosen to enable a fair comparison between the new implementation based on the ReActNet methodology, and the existing semantic segmentation version of Group-Net. To implement the ReActNet method on the DeepLab network, the network has to be modified to run with a MobileNet V1 backbone. Then the parallel tracks that constitute ASPP are binarized according to the ReActNet methodology.
The accuracy of all networks is tested on cityscapes. The cityscapes dataset is a benchmark for semantic segmentation in an automotive context, and allows for a comparison to be made to current state-of-the-art networks. Furthermore, the number of trainable parameters of the saved networks as well as the number of operations required will be taken into consideration.
The contributions of this research are as follows:

1) Presenting binary quantized versions of DeepLabV3 with 2 different backbones, using the ReActNet methodology.
2) Comparing these networks against the current state of the art Group-Net network on the cityscapes dataset

3) Demonstration that the methods from ReActNet can also be applied to other structures by binarizing FCN8s and FCN32 decoders.

II. Related Work

A. Semantic Segmentation

The first steps in using convolutional neural networks for semantic segmentation were made in the research presented in the paper on fully convolutional networks [4]. By replacing the final (fully connected) layers of a classification network with 1x1 convolutions, the network produces a heat map that shows where in the image a certain class is located. Although the use of fully convolutional networks significantly improved state of the art performance, the observed output was not as accurate as one would hope. The lowered accuracy was due to the down- and upsampling by a factor 32 over the course of the network. The factor 32 between the output of the "encoder" and the output of the decoder meant that fine details were lost.
To improve on this FCN-32 architecture, FCN-16 and FCN-8 were proposed, which include information from previous pooling layers, meaning the network upsamples 16 and 8 times respectively, rather than 32 times.
To improve the accuracy, U-Net [5], which was developed to detect and locate tumours in the lungs or brain from scans, adds connections between the encoder (downsampling) and decoder (upsampling). This aids in providing context during the upsampling stage, allowing for more accurate segmentation.
Google's DeepLab architecture [6] added several more improvements to the network architecture: Atrous convolutions, Atrous spatial pyramid pooling (ASPP) and the use of conditional random fields to post-process the output.
Atrous convolution increases the size of a filter by adding dilation. Dilation consists of spacing out the filter parameters and filling the "holes" with zeroes. A 3x3 filter for example, is expanded to a 5x5 filter by adding rows and columns of zeroes between the 3x3 filter values. This allows the network to get the context of the larger filter size, while having the number of parameters of the smaller filter.
ASPP is an extension of spatial pyramid pooling, presented in the paper on SPPNet [7]. It combines the information from different dilation rates to get more information on the original image. As conditional random fields are a post-processing step they will not be expanded upon in this paper.
DeepLabV3 [3] added batch normalization and suggested new dilation rates for each layer in a ResNet block [8]. Additionally, image-level features were added to the ASPP module. The image-level features offer the network the context of the full image by pooling, as opposed to the detail-level features from the convolution filters. DeepLabV3+ [9] suggested using a decoder structure instead of plain bilinear upsampling. This added the connections between encoder and decoder as presented in U-Net, which further improved accuracy. Fig. 1 shows the DeepLabV3 architecture, when using a ResNet backbone. In the figure, blocks 1-3 are the first three basic blocks of ResNet. Block 4 is based on the fourth basic block, but uses dilated convolution in stead of downsampling.

B. Quantization

As performance on neural network tasks increased, so did the requirements for hardware, both computation power and memory. Since it is not feasible to use powerful GPU's and large data storage devices for low-energy applications, such as driver-assistance functions, steps were taken to reduce these requirements, while trying to maintain accuracy and throughput. There are several techniques that can be used for compression of deep neural networks [10], but this paper focuses on quantization.
Quantization is the process of reducing the number of bits used to represent a number. The idea of using quantization for neural networks is not new, in the early 90's quantization was proposed as a way to make neural network implementation feasible on the hardware available at the time [11] [12]. The predominant format to store weights, biases and activation functions used in neural networks is the 32-bit floating point. This means that every value takes up 4 bytes of memory. To put this into perspective, ResNet-50 has over 23 million trainable parameters, meaning it takes over 736 million bits (circa 91.5 MB) to store these values alone. As all this data gets moved around during computation, reducing the number of bits per parameter can significantly reduce the amount of memory required to load the network, as well as the energy consumption. In addition, the multiplications required for convolution when using 32-bit floating point weights require a 32-bit multiplier. These multipliers are complex structures, which use a significant amount of energy to perform the multiplications (roughly 4 pJ for a single 32-bit FP multiplication [13]). When using a neural network on a battery-powered device such as a mobile phone, this will have a significant impact on battery life.
Various techniques have been suggested to quantize neural networks, and while they all reduce the number of bits used for weights and activations in the network, the methods used to limit the quantization error vary.
One of the early modern papers on Quantization for neural networks was the one that presented DoReFa-Net [14]. The DoReFa-Net paper aimed to only quantize the gradients used in the backwards pass. The research showed that, while weights and activations can be quantized deterministically, the gradients had to be stochastically quantized. DoReFa-Net achieved comparable results to full-precision networks, while reducing the memory and computational power required for backpropagation.
A year after the publication of DoReFa-Net, the paper on weighted-entropy based quantization [15] was based on the premise that most weights in convolutional or fully connected layers are concentrated near zero, implying the distribution of the weights is a bell-shape. The authors suggest using more quantization levels around the area where the concentration of weights is highest, or weighted quantization. This means that, rather than a linear quantization, where quantization levels are

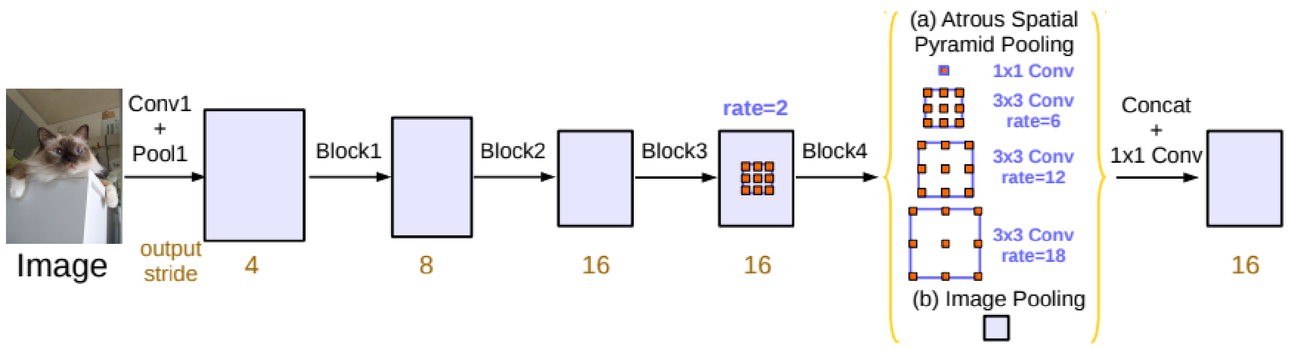


Fig. 1: DeepLabV3, figure from [3]

equidistant, or a log quantization, where the levels are highly concentrated around 0, the quantization levels are determined based on the original weight distribution, and the influence of the weights. Weights near 0 are numerous, but have relatively little impact on the quality of the output. Large weights are few, but have a relatively high impact. By concentrating the quantization levels between these two groups of weights, the weights that are relatively common, and have a reasonable impact on the outcome, have a smaller quantization error than the weights that are either infrequent or low-impact. This approach outperformed several state-of-the-art networks when applied to AlexNet.
Incremental Network Quantization [16] aimed to reduce the quantization error by introducing two innovations and applying these to a pretrained full-precision network.
Firstly, three operations are introduced that are interdependent. By partitioning the weights, quantizing in groups, and re-training the network, the weights of each layer are split into two groups: a low-precision base and a group to be re-trained to compensate for quantization loss.
Secondly, by repeating this process until all weights are converted to low-precision, the incremental network quantization enhances the accuracy of the network as a whole.
The main goal of INQ is to convert all 32-bit floating point weights to either a power of two or zero, while minimizing loss of accuracy. The resulting performance when using a bit-width of 5 is a decrease of 1.59% in top-1 accuracy and a 1.21% decrease in top-5 accuracy when INQ is applied to ResNet-50 and a 0.15% and 0.23% decrease respectively when applied to AlexNet.

C. Binary Quantization

When quantization is set to only use 1 bit per value, we speak of binary quantization. In order to maintain the possibility of positive and negative values, the binary value is interpreted as either +1 or -1.
XNOR-Net [17] was one of the first binarized networks to achieve state-of-the-art performance, using single bit values for both the weights and activations of convolutional layers. This resulted in 32x savings in memory and 58x faster convolutional operations on a CPU compared to the full-precision network. To approximate the values of the full-precision weights with binary values, a scaling factor was used. The scaling factor is a positive real scalar that was set at the average of the absolute weight values. During training, the forward pass used binarized weights, while the parameter update was done using the full-precision weights. Convolutions, when binary values are used, consist of a shift operation followed by a dot product. For binary values, this dot product can be replaced with an XNOR-gate followed by a bitcount. This allows for inference to be done in real-time on a CPU and uses less memory than a full-precision network. As experiments showed that the increase in speed was not very large on small channel and filter sizes, the authors decided to not binarize the first and last layers.
Around the same time as XNOR-Net, a paper on quantized neural networks (QNN) [18] was presented, that used a similar approach to binarization, but did include the first and last layers in the binarization process. In addition, stochastic binarization was considered, which, rather than taking the sign of the value to be binarized, takes the sign of the value minus a uniform random variable z. The performance of this network, when using 2-bits to represent the activations, is better than that of XNOR-Net. While XNOR-Net achieves 44.2% and 69.2% top-1 and top-5 accuracy respectively on ImageNet, QNN reaches 51.03% and 73.67%.

D. Group-Net

The paper on Group-Net [2] presents a different approach and aims to find a middle ground between binary and high bit-width quantization. Rather than the value approximation used by other methods, Group-Net aims to approximate the structure of the network. To that end, the full-precision network is divided into groups. By using a set of binary bases, the floating-point structure is approximated. Fig. 2 shows this process: Fig. 2a shows the full-precision residual blocks, where each convolution block is replaced with a binarized version (value approximation). Fig. 2b shows a basic group-wise decomposition where each full-precision residual block is approximated by several binary blocks in a similar structure. Finally, Fig. 2c shows the Group-Net approach, where the whole group is approximated by multiple identical parallel binary network branches. Group-wise decomposition considers

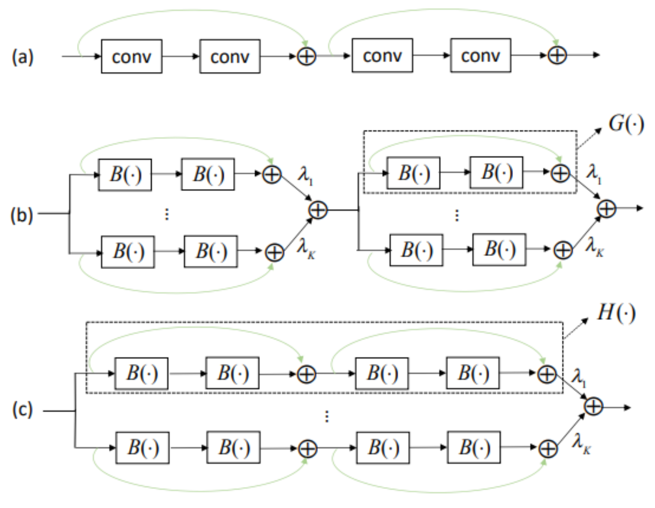


Fig. 2: Example of group-wise binary decomposition from group-net. From [2]

each convolution as a linear combination of binary groups. By combining these groups across layers, the structure of the original network can be approximated, while avoiding the error accumulation that occurs when decomposing the network layer by layer. While value approximation approaches show promising performance on classification tasks, the authors of Group-net [2] claim that performance degrades on more challenging tasks like semantic segmentation.

For the semantic segmentation task, the authors absorb ASPP into the feature extraction stage, rather than run it on top of the extracted features. When absorbing the ASPP into the network each branch gets its own dilation rate, essentially making ASPP a by-product of the network branch decomposition. The authors named this structure binary parallel atrous convolution (BPAC) and it is one of the innovations that came with this paper.

While most of the papers on quantization mentioned above only address classification tasks, Group-Net also includes an implementation for a semantic segmentation network. By applying Group-Net on DeepLabV3 and testing on the PASCAL VOC 2012 validation set, the Group-Net version of DeepLabV3 with a ResNet-34 backbone achieved a mIoU of 73.6% when only the backbone was modified, and 70.2% when both the backbone and ASPP were modified. The full-precision baseline was 76.9%, meaning that the binarized versions scored 3.3% and 6.7% lower respectively.

E. ReActNet

While Group-Net achieves higher accuracy through extending the network into multiple parallel paths, the authors of the paper on ReActNet [1] go back to value approximation. This means that the network can be a factor K smaller than that of Group-Net, where K is the number of bases used.

The goal behind the development of ReActNet was to close the gap between binary neural networks and real-valued neural networks. The network uses a MobileNet V1 [19] backbone, as the authors believe binarizing a compact model is of more practical use. The main contributions introduced in the paper are the RSign and RPReLU functions, as well as using a knowledge distillation learning method for a quantized network.

The ReAct-Sign (RSign) and ReAct-PReLU (RPReLU) functions are new generalizations of the Sign and PReLU functions respectively. While the classical sign function is a step function that has value -1 for values up to and including zero and +1 for values above 0, the RSign function adds a learnable parameter α that shifts the function over the x-axis. The parameter can have different values for diffent channels, which allows for further optimization. The resulting formula is shown in Equation 1.

$$x_i^b = h(x_i^r) = \begin{cases} +1, & \text{if } x_i^r > \alpha_i \\ -1, & \text{if } x_i^r \leq \alpha_i \end{cases} \tag{1}$$

The RPReLU function similarly shifts the PReLU function by 2 learnable parameters γ and ζ, along the x- and y-axis respectively. The formula for the RPReLU function is shown in equation 2.

$$f(x_i) = \begin{cases} x_i - \gamma_i + \zeta_i, & \text{if } x_i > \gamma \\ \beta_i(x_i - \gamma_i) + \zeta_i, & \text{if } x_i \leq \gamma \end{cases} \tag{2}$$

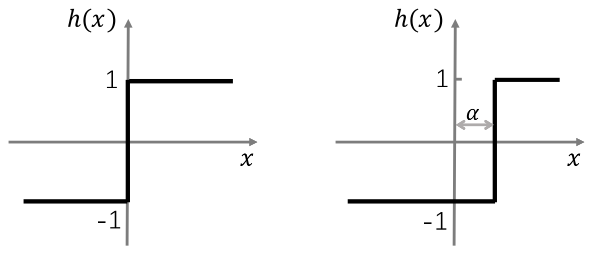
To demonstrate the effect of these parameters on the behaviour of the function outputs, Fig. 3 shows the graphs of Sign, RSign, PReLU and RPReLU.

The training method of ReActNet is a two-stage process. The network uses a full-precision teacher network and trains a student network with binary activations only during the first stage. During the second stage the teacher network remains the same, but the student network is replaced by a fully binarized network that uses the (quantized) values of the first stage as a starting point. Both stages use the same custom loss function that takes the cross-entropy loss of the teacher network and student network. This means that, rather than using the ground truth of the dataset, the student network is trained to approach the outputs of the teacher network.

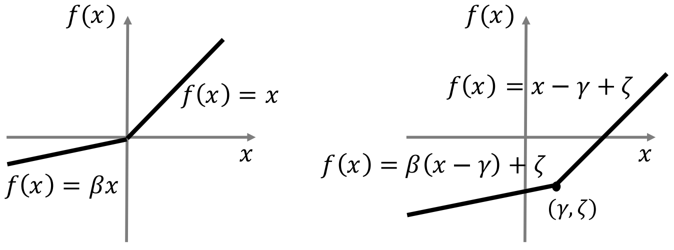
ReActnet achieves 69.4% top-1 accuracy on ImageNet, surpassing the ResNet-18 benchmark, while achieving more than 22x computational complexity reduction.

III. Dataset

To train and test the extended ReActNet networks, the Cityscapes dataset [20] is used. This dataset is widely used as a benchmark for semantic segmentation which allows for comparison of results to current state of the art (non-quantized) networks. Cityscapes consists of 5000 finely annotated images of daytime driving scenery, divided between 2975 images for training, 500 for validation, and 1525 for testing. The latter does not contain any ground truths and can thus not be used to determine metrics. The images in the dataset are video stills from 50 cities in Germany, spread out over several months, in good to medium weather conditions. The images have been selected to ensure a large number of



(a) Sign vs. RSign



(b) PReLU vs. RPReLU

Fig. 3: Comparison of RSign and RPReLU and traditional Sign and PReLU functions, figure from: [1]

dynamic objects, varying scene layouts, and backgrounds are present in the data.

IV. Methodology

For both ReActNet and Group-Net, github repositories are available [21] [22]. As mentioned in the related work section, Group-net is already implemented on a DeepLabV3 framework. To enable a fair comparison, the ReActNet implementation is extended from the MobileNet V1 backbone to include the decoder structure of DeepLab.
Training and inference of both networks is primarily done on an Nvidia GTX 1070 ti. In addition the MPS research group made a server available that offers an Nvidia RTX 2080 ti. The latter card offers 12 GB of VRAM compared to the 8 GB of the 1070 ti, which allowed training with slightly increased batch sizes. As neither setup allows for training with large batch sizes, a learning rate that maximizes the network performance will have to be determined. To that end, a ReActNet-DeepLabV3 network, using the ResNet-based version of ReActNet is implemented and run for 25 epochs at varying learning rates.

A. Extending ReActNet for semantic segmentation

The first step of extending ReActNet to a network for semantic segmentation was to find a teacher network that was pretrained on the Cityscapes dataset. While the training methodology from ReActNet does not require any similarity in network architecture between the teacher and student network, the choice was made to use a DeepLabV3+ network with a MobileNetV2 backbone [23] as the teacher. As both the backbone and decoder of the teacher are newer versions of the binarized student network, the basic architecture is similar. In addition to the pretrained network, the repository also contained scripts from the cityscapes github repository to handle the void class. By combining the training script from ReActNet with the dataloaders from the other repository, the train script allowed the network to be trained according to the ReActNet methodology on the Cityscapes dataset. The mIoU and class IoU metrics were determined by finding the intersection and union of the ground truth and student network output.
ReActNet uses a custom loss function that compares the outputs of the student and teacher networks. The function takes the Kullback-Leibler (KL) divergence between the softmax outputs p_c of the binary network B_θ and real-valued network R_θ. Subscripts c and n denote the classes and batch size respectively.

$$\mathcal{L}_{Dist} = -\frac{1}{n}\sum_{c}\sum_{i=1}^{n} p_c^{R_\theta}(X_i) log \frac{p_c^{B_\theta}(X_i)}{p_c^{R_\theta}(X_i)} \tag{3}$$

However, this loss function was designed for a classification task, and therefore only accepts inputs of $n \times c$. To allow for evaluation on a semantic segmentation task, which has $n \times h \times w \times c$ inputs, the loss function had to be modified. This was done by taking the knowledge distillation (KD) loss function from ReActNet and treating each pixel as a separate classification task. The resulting loss function then becomes the following:

$$\mathcal{L}_{Dist} = -\frac{1}{nhw}\sum_{c}\sum_{i=1}^{h}\sum_{j=1}^{w}\sum_{k=1}^{n} p_c^{R_\theta}(X_{i,j,k}) log \frac{p_c^{B_\theta}(X_{i,j,k})}{p_c^{R_\theta}(X_{i,j,k})} \tag{4}$$

After setting up the training environment, the ReActNet network was extended with various decoder structures to allow for semantic segmentation testing.

B. Binarization of the DeepLab decoder

To extend the ResNet and MobileNet backbones to a DeepLabV3 network, two major steps are required. The decoder structure itself needs to be added, which consists of the ASPP structure, a 1x1 convolution to reduce the dimensionality of the ASPP output, and a bilinear upsampling to get the output back to the same size as the network input. Additionally, the backbone needs to be modified so that atrous convolutions are used instead of the downscaling of the original network. The architecture of the network is comparable to that shown in Fig. 1, albeit with all but the first layer binarized, every block starting with an RSign activation, and the (P)ReLU activations in the original network having been replaced with RPReLU activations.

1) ASPP: To binarize ASPP, each separate path is implemented using the ReActNet binarization. In the ASPP block, the input is run through five paths in parallel:

- a 1x1 convolution
- a 3x3 dilated convolution with rate 6
- a 3x3 dilated convolution with rate 12
- a 3x3 dilated convolution with rate 18
- a pooling block

The 5 resulting outputs are then concatenated and, through the use of a 1x1 convolution, projected back down to the same dimensions as the input of the ASPP block.
The 1x1 convolution path (the block marked 1x1 conv in Fig. 1) consists of a sequence of a binarized 1x1 convolution, batch norm, and RPReLU operation. The RPReLU operation is implemented by putting learnable bias functions before and after the PReLU operation. This approach is the same as the one used in the original ReActNet paper [1] and shows how the two parameters act as a shift on the in- and output. The three dilated convolutions are implemented as a separate module, consisting of a 3x3 dilated convolution, batch norm and RPReLU layer, the latter of which is again implemented using two learnable biases around a PReLU layer. The (Binary)ASPPConv module takes the number of input channels, number of output channels, and dilation rate as inputs during network initialization, which allows for a single module to describe various dilation rates. The pooling path is also implemented as a module, and consists of an adaptive average pool layer with a 1x1 output, a 1x1 convolution layer, batch norm, and RPReLU, implemented in the same way as the other RPReLU layers in the network. To ensure the output has the same size as that of the other paths, bilinear interpolation is applied to ensure the height and width of the data are back to the same size as before the adaptive average. The outputs of the five paths are then concatenated and fed through a projection module that reduces the number of layers in the data by a factor five. This module consists of a sequence of a 1x1 convolution, batch norm, RPReLU and dropout layers, as in the DeepLabV3 architecture.
Fig. 4 shows the top-level view of the DeepLabV3 architecture as implemented.

2) Atrous Convolution: For the Atrous Convolution to replace the downsampling in both ResNet and MobileNetV1, the Basic block modules need to be modified. The Basic block modules contain the repeating elements of each layer of the network. In the ResNet architecture this consists of the binary activation function (RSign), a convolution, batch normalisation, and RPReLU function. The MobileNetV1 Basic block module contains some additional functions to fit the MobileNetV1 architecture, but operates along the same principle. To extend the functionality, dilation is added as an additional input argument to the Basic block module, with a default value of 1. This allows the code to remain functional for architectures that do not include the dilation. When the value is larger than 1, the 3x3 convolution in the block is replaced by a dilated version. In addition, the function that creates the layers when the network is built is changed. This change ensures that when a layer needs to apply dilation instead of the regular downsampling, three steps are taken. Firstly the dilation rate of the previous block is stored in a variable. Then the new dilation rate is set to the product of the previous dilation rate and the stride input. Finally the stride value is set to 1. The previous dilation rate is then used for the first block in each layer, and the new dilation rate for the remaining ones.
For ResNet-18 each layer of the network consists of 4 blocks, where the last 2 layers are replaced with dilated versions. For MobileNetV1, the general approach is the same, but where ResNet has two layers of 4 blocks with the same data dimensions, MobileNetV1 has a layer of 6 blocks and a layer of 2 blocks with the same data dimensions. The result is that while the last 8 blocks are replaced with dilated versions, the number of blocks with each dilation rate is different.

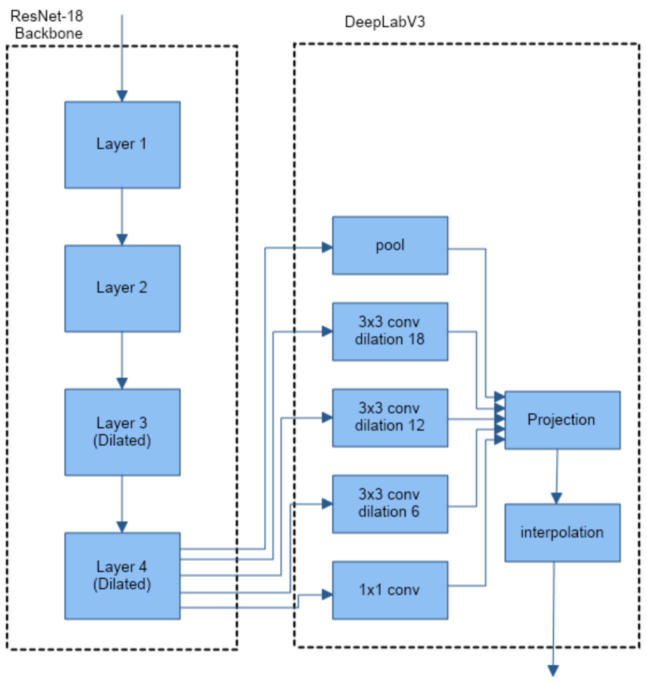


Fig. 4: Architecture of the ResNet-18 backbone and DeepLabV3 decoder

C. Binarization of FCN 32/8s decoders

To investigate whether the ReActNet methodology can also be used for other decoder structures, a FCN8s decoder was implemented. Where DeepLabV3 requires modification of the backbone, the FCN decoders use the output of the backbone before the final softmax layer, and, depending on the version of the decoder, intermediate outputs of the backbone. The least complicated version of the FCN decoder is FCN32, which only uses the final output and upscales it. Rather than a single 32x upscaling, the choice was made to upscale the output in 3 steps. The first two steps consist of a transpose convolution

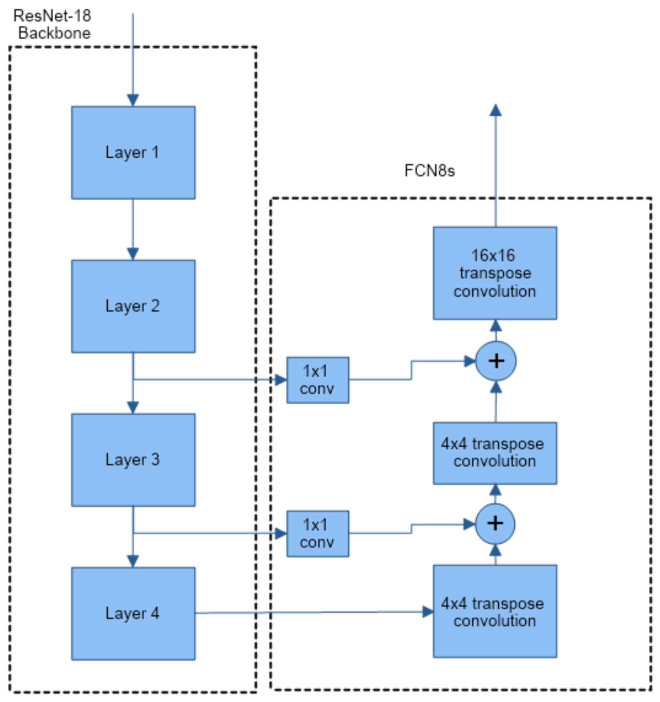

ResNet-18 Backbone
Layer 1
Layer 2
Layer 3
Layer 4
FCN8s
16x16 transpose convolution
1x1 conv
4x4 transpose convolution
1x1 conv
4x4 transpose convolution

Fig. 5: Architecture of the ResNet-18 backbone and FCN8s decoder. The FCN32 decoder does not have the connections with the 1x1 convolutions between the outputs of layers 2 and 3 and the decoder.

with a 4x4 kernel and stride 2. The last step is a transpose convolution with a 16x16 kernel and stride 8. By using these layers, the influence of the cross-connections between the encoder and decoder becomes more clear, as it is the only distinction between the FCN32 and FCN8s networks.
In order to ensure that the transpose convolutions are binarized the same way as the convolutions in ReActNet, the *HardBinaryConvolution* module was modified to create a transpose convolution with binarized weights and biases.
Fig. 5 shows the architecture used for the FCN8s decoder when paired with the ResNet-18 based backbone. As with the DeepLabV3 architecture, the network is in essence the same as the full-precision network, with the addition of RSign and RPReLU functions, and binary convolutions throughout.

D. Group-Net

To run the Group-Net network, the code from the authors of the paper was integrated into the framework used to train the ReActNet networks, while taking into account the change in training methodology. Group-Net uses a single ”step” to train the network, and trains directly off the ground truth. As such, the regular cross-entropy loss can be used. As the backbone architecture is not directly specified in the code, but extracted from a pretrained version, a pretrained version of the backbone is used. The network is then set to train for a maximum of 400 epochs, which was interrupted as soon as the validation loss curve plateaued.

Learning rate	train loss after 25 epochs
0.00001	0.7906
0.00005	0.5960
0.0005	0.5704
0.0002	0.5283
0.0001	0.5394

TABLE I: Train loss after 25 epochs for various learning rates

E. Evaluation

To determine the accuracy of the networks on the cityscapes images the mean intersection-over-union (mIoU) metric will be used. To determine the mIoU, the class IoU’s are calculated during validation and then averaged. The performance of the binarized models will be compared to that of the teacher network and Group-Net. In addition, a computational cost analysis is done on the networks, where the number of trainable parameters, as well as the number of operations. To determine the number of operations required, similar methodology as in the ReActNet paper [1] is used. As the majority of operations comes from the convolutional layers, these are the only ones taken into account. Binary operations (BOPs) and floating point operations (FLOPs) are counted separately. The total number of operations (OPs) is determined through equation 5.

$$OPs = BOPs/64 + FLOPs \tag{5}$$

.

V. Results

A. Learning Rate

To find the ideal learning rate for the training setup, several learning rates were tested for the DeepLabV3 network with ResNet-18 based backbone and binary activations as in step 1 of the training process. The network was trained for 25 epochs per learning rate. The resulting graph in Fig. 6 and the values of the train loss after 25 epochs in Table I show that a learning rate of 0.0002 gives the best results for a batch size of 2. As learning rate seems directly related to the batch size, the networks that are trained with batch size 8 on the server use a learning rate of 0.0008. As the learning rate in the ReActNet methodology is the same in both steps of the process, the assumption was made that the found optimum training rate from step 1 also approximated the optimum rate in step 2.

B. Influence of binarization

To determine the influence of binarizing each part of the network, partially binarized versions of the ResNet-based ReActNet-FCN8s network were trained. Four versions of the network were defined, to cover all possible combinations of binary and full-precision encoder and decoder.

1) No Binarization: Both encoder and decoder are full-precision

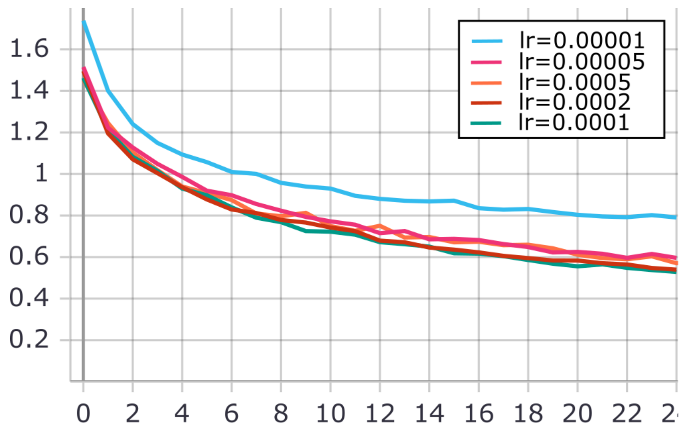


Fig. 6: Training loss as a function of the number of epochs trained for the DeepLabV3 network with ResNet-18 backbone at various learning rates.

Binarization	mIoU
No Binarization	56.07%
Encoder	54.86%
Decoder	49.14%
Encoder and Decoder	47.59%

TABLE II: Mean Intersection-over-Union for partially binarized networks as well as full precision and fully binarized

2) Encoder: The encoder uses binarized weights and activations, the decoder is full-precision.
3) Decoder: The encoder is full-precision, the decoder uses binarized weights and activations.
4) Encoder and Decoder: Both encoder and decoder use binarized weights and activations.

By comparing performance of these four combinations, the influence of binarizing the encoder and decoder can be shown. Table II shows the mIoU for each of the networks after training. Each network was trained in 2 steps, with 200 epochs per step. While the network without binarization did not require a two-step process to handle binarization, the same approach was used, which meant the learning rate had the same sawtooth shape over the course of the training as the other networks. As expected, the network with no binarization has the highest mIoU, and the network with binary encoder and decoder has the lowest. Table II also shows that the combination of a binarized encoder and full-precision decoder outperforms the network with a binarized decoder and full-precision encoder. This implies that, in the context of semantic segmentation, the relative importance of the data in the decoder is higher than that in the encoder.

C. Performance of binarized networks

Table III shows the mIoU for each of the networks. As expected, the FCN8s and FCN32s networks were outperformed by the DeepLabV3 networks. This is due to the difference in decoder structure. The DeepLabV3 decoder is able to get a better fit on the data due to the combination of the number of parameters and architectural choices that were made. Table III also shows that the DeepLabV3-based networks achieve comparable performance to Group-Net, despite the small batch size and limited training duration. In addition, examples from the dataset and the predictions for each image are shown in Fig. 8 and Fig. 9. It is immediately clear that the networks with FCN32 and FCN8s decoders are unable to distinguish between multiple smaller objects in close proximity such as the traffic signs in the middle of Fig. 8. Group-Net and the two versions of ReActNet with a DeepLabV3 decoder perform comparably on both examples. It is however noticeable that Group-Net fills the ego-vehicle with various other class labels, while the DeepLabV3 networks simply classify it as being part of the road. This may stem from having a backbone that was pretrained on ImageNet.

Backbone	Network (epochs)	mIoU
Group-Net	DeepLabV3 (313)	66.76%
ReActNet (ResNet-18)	FCN32 (2×200)	48.11%
ReActNet (ResNet-18)	FCN8s (2×200)	48.96%
ReActNet (ResNet-18)	DeepLabV3 (2×200)	61.15%
ReActNet (MobileNetV1)	FCN32 (2×200)	43.15%
ReActNet (MobileNetV1)	FCN8s (2×200)	45.63%
ReActNet (MobileNetV1)	DeepLabV3 (2×200)	57.38%
Teacher Network (MobilenetV2)	DeepLabV3 (pretrained)	72.10%

TABLE III: Achieved mean Intersection-over-Union on the cityscapes dataset

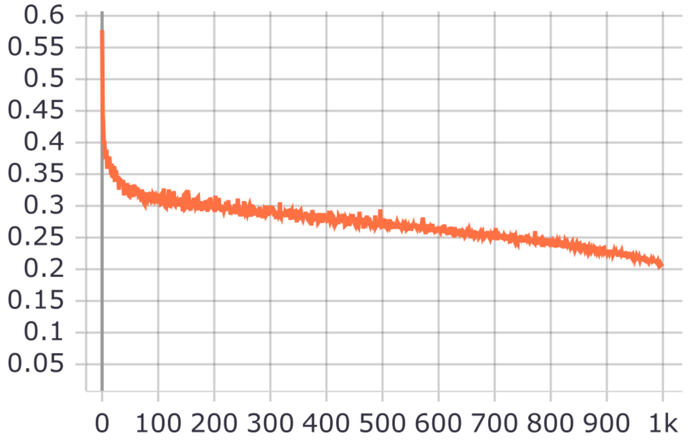
Network (backbone)	epochs	mIoU
ReActNet-DeepLabV3 (ResNet-18)	2×200	61.15%
ReActNet-DeepLabV3 (ResNet-18)	2×400	64.15%
ReActNet-DeepLabV3 (ResNet-18)	2×600	65.03%
ReActNet-DeepLabV3 (ResNet-18)	2×1000 (batchsize 8)	65.76%

TABLE IV: ReActNet mIoU increasing as the network is trained longer

While the training and validation loss of Group-Net were plateauing after 313 epochs, the ReActNet-DeepLabV3 networks still had a significant slope. Hence, additional training runs with 400 and 600 epochs per step were done. The results of these runs are shown in table IV. As these runs still showed a declining slope on the validation loss at the end of training, a final experiment was run with 1000 epochs per step. The hardware for the last run allowed larger batch sizes, so the batch size was increased to 8 and the learning rate accordingly to 0.0008. Although this long run still did not reach a plateau in the validation loss, as can be seen in Fig. 7 the network accuracy is only 1% under the result from Group-Net.

D. Model Size

Since the models are stored as though they were full-precision networks, and thus each parameter is stored as a 32-bit float, the size of the saved checkpoints does not give a complete picture of the reduction in network size that the quantization achieves. To resolve this, the number of trainable parameters in each network architecture has been determined. As can be seen in table V, the networks that have been binarized based on the ReActNet architecture have significantly less trainable parameters than the Group-Net network. The



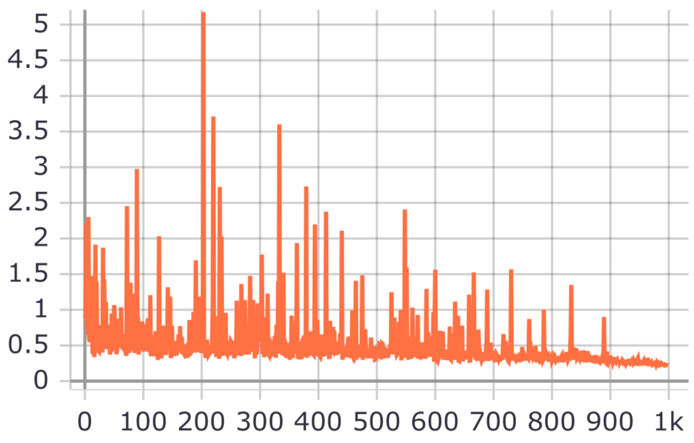


Fig. 7: Train loss (left) and validation loss (right) of the DeepLabV3 network with ResNet-18 backbone over the second set of 1000 epochs

networks with a MobileNetV1 backbone have significantly more parameters than those with a ResNet backbone. This stems from the fact that the MobileNetV1 backbone has more layers than the ResNet-18 backbone, and thus more trainable parameters. While none of the models have solely binary trainable parameters, as they all use a full-precision first convolution, the percentage of full-precision and binary parameters is assumed to be roughly equal. The result is thus that the ReActNet-based implementations take up roughly a quarter of the space, when using a ResNet-18 based backbone, and half of the space, when using a MobileNetV1-based backbone, when compared to Group-Net.

E. Number of Operations

To further determine the efficiency of the ReActNet-based networks compared to the Group-Net baseline, the number of floating point and binary operations is determined. Since the convolution layers are the biggest contributor to the number of operations in the network, and to simplify the calculations, only the convolution layers are taken into account. Since all networks use a full-precision first layer, and only binary convolutions after that, the number of floating-point operations (FLOPs) is only determined by the first layer. The remaining convolution layers in the network determine the number of binary operations (BOPs), and the total number of operations (OPs) is determined through the equation 5.
The number of operations for each convolution is determined by taking the kernel size and multiplying it with the output size. This means that for e.g. a 3x3 kernel with 3-dimensional input that has a 256x256x32 output, the number of operations is $(3 * 3 * 3) * (256 * 256 * 32) = 56,623,104$.
Table VI shows the number of FLOPs, BOPs and OPs for each network. While both Group-Net and the new networks with a ResNet-18 backbone have a 7x7 kernel on the first layer, Mobilenet only uses a 3x3 kernel. It is noticeable that the number of operations is significantly higher than the values mentioned in the ReActNet paper [1]. This is due to the change in size of the input image, The ImageNet images were cropped to 224x224, while the cityscapes images were cropped to 512x512. This means that the image size is a factor 5.2 larger, and the number of operations per convolution is increased by this factor. Additionally, extending the network with the decoder structures also increased the number of operations.
While Group-Net uses more operations in the network, the decrease by switching to ReActNet is not the factor 4 one would expect from going from Group-Nets 4 branches to the single branch in ReActNet. This is due to the BPAC structure mentioned in section II-D. The number of binary operations in the network is dominated by the convolutions in the decoder. By absorbing the ASPP paths into the branches of the encoder, rather than running them 4 times in parallel, the number of operations is reduced significantly.

VI. Discussion

While the results support the viability of using ReActNet for semantic segmentation, hardware and time-restrictions on the research made that some avenues have gone unexplored. Although the performance of the binarized DeepLabV3 networks reaches comparable results to Group-Net after 400 epochs, the validation loss had not yet stabilized, which is shown by the increase in performance after 600 and 1000 epochs. In addition, larger batch sizes are shown to improve performance, but the upper limit of what was possible on the hardware available to the author was at 8 images per batch, while e.g. the used teacher network was trained at 16 images per batch. The teacher network itself also poses options for further research, as the upper limit of the network performance will be at the performance of the teacher network. Since the KD-loss function used does not require the student and teacher network to have the same architecture, any network trained on cityscapes can be used. As such, using networks higher on the cityscapes leaderboards could also potentially improve performance. Each of these options could further improve the network performance, without requiring additional calculations at runtime, and would therefore be worth investigating in future research.

Decoder \Backbone	ReActNet (ResNet-18)	ReActNet (MobileNetV1)	Group-Net
DeepLabV3	15,923,840	36,852,096	60,473,002
FCN8s	11,312,416	28,456,608	N/A
FCN32s	11,305,120	26,442,016	N/A

TABLE V: number of trainable parameters

Network	BOPs	FLOPs	OPs
Group-Net	5.74×10^{10}	6.17×10^{8}	1.51×10^{9}
FCN32 (ResNet-18)	3.37×10^{10}	6.17×10^{8}	1.14×10^{9}
FCN8 (ResNet-18)	3.38×10^{10}	6.17×10^{8}	1.14×10^{9}
DeepLabV3 (ResNet-18)	2.90×10^{10}	6.17×10^{8}	1.07×10^{9}
FCN32 (MobileNetV1)	4.99×10^{10}	0.57×10^{8}	0.84×10^{9}
FCN8 (MobileNetV1)	5.00×10^{10}	0.57×10^{8}	0.84×10^{9}
DeepLabV3 (MobileNetV1)	6.00×10^{10}	0.57×10^{8}	0.99×10^{9}

TABLE VI: Number of operations per network

VII. Conclusions

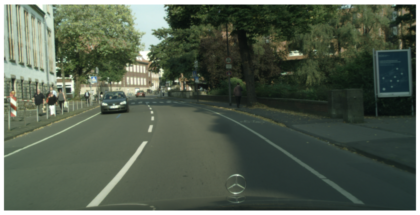
In this paper, I have shown that the ReActNet approach can be successfully extended to and applied on semantic segmentation tasks, as shown in the results section. The network achieves comparable performance to Group-Net, and the loss graphs show there is still room for improvement, despite the reduced number of trainable parameters. The loss graphs show the validation loss is not yet plateaued, and thus the network performance could likely be improved when trained for more epochs, or with a larger batch size. The ReActNet approach also works when used with the FCN decoder structure. These architectures were used as a proof-of-concept to demonstrate how other architectures can be quantized with relatively small losses in accuracy.

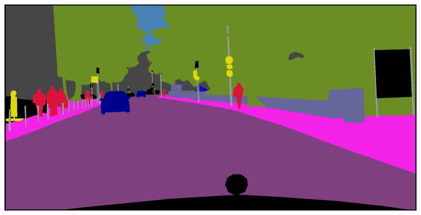
Acknowledgment

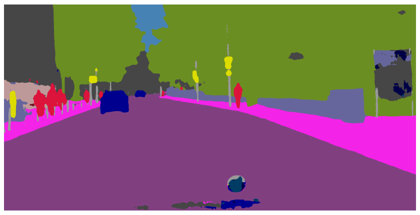
I would like to thank dr. Pavol Jancura, for the aid in finding a topic and supervising the project, and Floran de Putter MSc. for his expertise on binarized networks and his insightful feedback.
I would like to acknowledge the MPS group and Anweshan Das MSc. in particular, for setting up the server and making it available to me so the networks could be trained with an increased batch size.

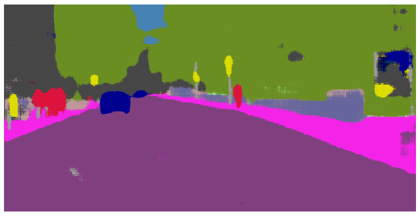
References

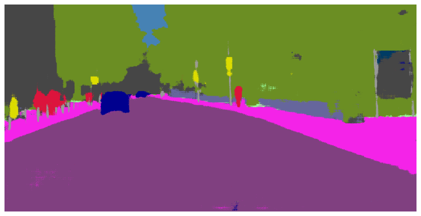
[1] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions, 2020.
[2] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Peng Chen, Lingqiao Liu, and Ian Reid. Structured binary neural networks for image recognition, 2020.
[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.
[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.
[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018.
[10] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jag Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 02 2020.
[11] Fiesler Choudry, E. Fiesler, A. Choudry, and H. J. Caulfield. A weight discretization paradigm for optical neural networks. In *in Proceedings of the International Congress on Optical Science and Engineering*, pages 164–173. SPIE, 1990.
[12] Wolfgang Balzer, Masanobu Takahashi, Jun Ohta, and Kazuo Kyuma. Weight quantization in boltzmann machines. *Neural Networks*, 4:405–409, January 1991.
[13] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014.
[14] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.
[15] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
[16] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044, 2017.
[17] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016.
[18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18(1):6869–6898, January 2017.
[19] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
[20] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.
[21] Zechun Liu. Reactnet. https://github.com/liuzechun/ReActNet, 2020.
[22] Bohan Zhuang. Group-net-image-classficiation. https://github.com/bohanzhuang/Group-Net-image-classification, 2020.
[23] Gongfan Fang. Deeplabv3plus-pytorch. https://github.com/VainF/DeepLabV3Plus-Pytorch, 2020.

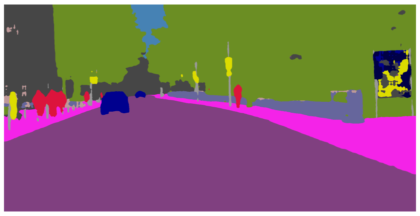

(a) input image

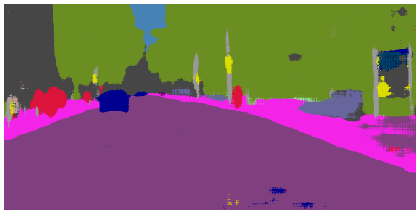

(b) ground truth

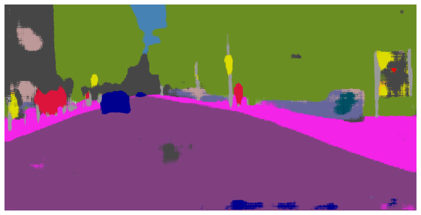

(c) Group-Net prediction


(d) ReActNet-FCN32 (ResNet-18) prediction


(e) ReActNet-FCN8s (ResNet-18) prediction


(f) ReActNet-DeepLabV3 (ResNet-18) prediction


(g) ReActNet-FCN32 (MobileNetV1) prediction


(h) ReActNet-FCN8s (MobileNetV1) prediction

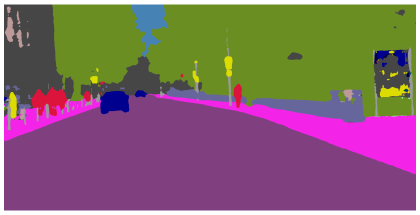
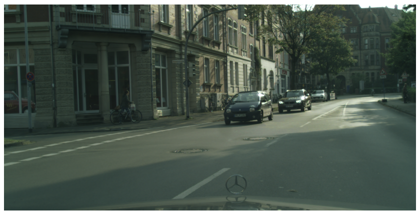
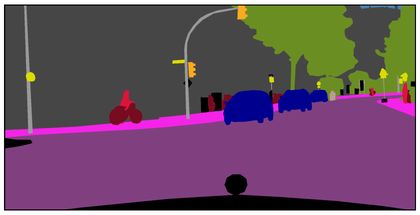
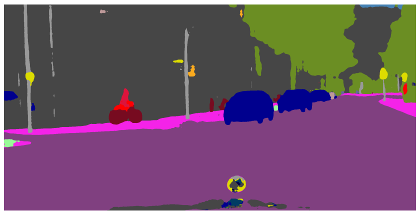
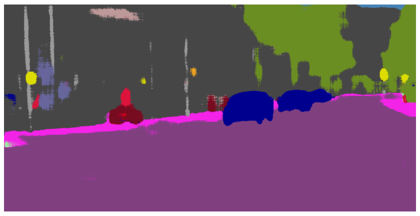

(i) ReActNet-DeepLabV3 (MobileNetV1) prediction

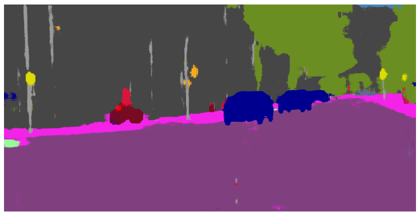
Fig. 8: Example from the cityscapes dataset

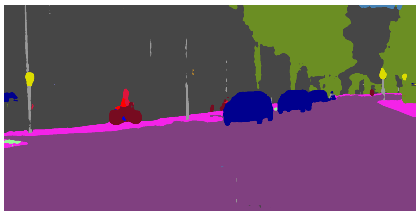

(a) input image

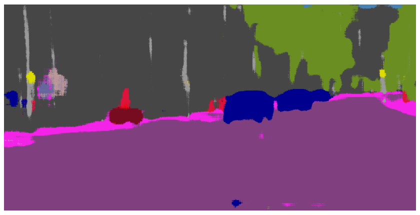

(b) ground truth

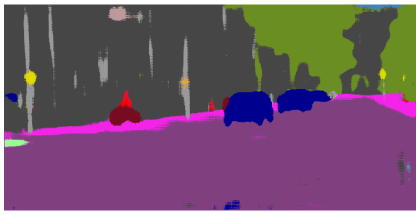

(c) Group-Net prediction


(d) ReActNet-FCN32 (ResNet-18) prediction


(e) ReActNet-FCN8s (ResNet-18) prediction


(f) ReActNet-DeepLabV3 (ResNet-18) prediction


(g) ReActNet-FCN32 (MobileNetV1) prediction


(h) ReActNet-FCN8s (MobileNetV1) prediction

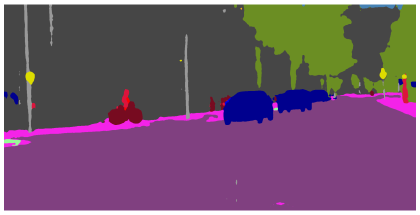

(i) ReActNet-DeepLabV3 (MobileNetV1) prediction

Fig. 9: Example from the cityscapes dataset