Joint Optimization for DNN Model Compression and Corruption Robustness

Serin Varghese, Christoph Hümmer, Andreas Bär, Fabian Hüger, and Tim Fingscheidt

Abstract Modern deep neural networks (DNNs) are achieving state-of-the-art results due to their capability to learn a faithful representation of the data they are trained on. In this chapter, we address two insufficiencies of DNNs, namely, the lack of robustness to corruptions in the data, and the lack of real-time deployment capabilities, that need to be addressed to enable their safe and efficient deployment in real-time environments. We introduce hybrid corruption-robustness focused compression (HCRC), an approach that jointly optimizes a neural network for achieving network compression along with improvement in corruption robustness, such as noise and blurring artifacts that are commonly observed. For this study, we primarily consider the task of semantic segmentation for automated driving and focus on the interactions between robustness and compression of the network. HCRC improves the robustness of the `DeepLabv3+` network by 8.39% absolute mean performance under corruption (mPC) on the Cityscapes dataset, and by 2.93% absolute mPC on the Sim KI-A dataset, while generalizing even to augmentations not seen by the network in the training process. This is achieved with only minor degradations on undisturbed data. Our approach is evaluated over two strong compression ratios (30% and 50%) and consistently outperforms all considered baseline approaches. Additionally, we perform extensive ablation studies to further leverage and extend existing state-of-the-art methods.

S. Varghese (✉) · C. Hümmer · F. Hüger
Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany
e-mail: john.serin.varghese@volkswagen.de

C. Hümmer
e-mail: christoph.heummer@volkswagen.de

F. Hüger
e-mail: fabian.hueger@volkswagen.de

A. Bär · T. Fingscheidt
Institute for Communications Technology (IfN), Technische Universität Braunschweig, Schleinitzstr. 22, 38106 Braunschweig, Germany
e-mail: andreas.baer@tu-bs.de

T. Fingscheidt
e-mail: t.fingscheidt@tu-bs.de

T. Fingscheidt et al. (eds.), *Deep Neural Networks and Data for Automated Driving*,
https://doi.org/10.1007/978-3-031-01233-4_15

1 Introduction

Motivation: Image classification [KBK20], object detection [YXC20], machine translation [SVS19], and reading comprehension [ZHZ11] are just some of the tasks where deep neural networks (DNNs) excel at. They have proven to be an effective way to extract information from enormous amounts of data, and they are only expected to become more advanced over time. Despite their rapid progress, two insufficiencies of DNNs need to be addressed before deployment in real-time systems. First, in real-world applications, the edge devices on which these networks are deployed have limited capabilities in terms of the availability of memory and computational complexity (operations per second) that are required for neural network deployment. Second, the DNNs suffer from being not robust to even slight changes in the input (such as noise and weather conditions), which makes deployment in safety-critical applications challenging (Fig. 1).

Lack of DNN efficiency: To overcome the lack of efficiency, techniques such as pruning [HZS17, MTK+17, TKTH18], quantization [CLW+16, JKC+18, JGWD19] including quantization-aware trainings [GAGN15], knowledge distillation [HVD14], and encoding techniques [HMD16] are commonly used. All of these strategies seek to take advantage of the available redundancy in large DNNs to achieve run-time speedup.

Lack of DNN robustness: In addition to this insufficiency, recent studies [AMT18, HD19, BHSFs19] show that DNNs are not robust to even slight changes to the input image. These changes vary from carefully crafted perturbations called adversarial attacks [XLZ+18, DFY+20, BKV+20], to real-world augmentations such as snow, fog, additive noise, etc. [HD19]. The changes in the input image could vary from changes in just a few pixels [SVS19] to more global changes such as contrast and brightness [ZS18]. In the real world, such local or global changes are to be expected. For example, varying lighting conditions or foggy weather conditions can cause changes in the brightness and contrast of the input image.

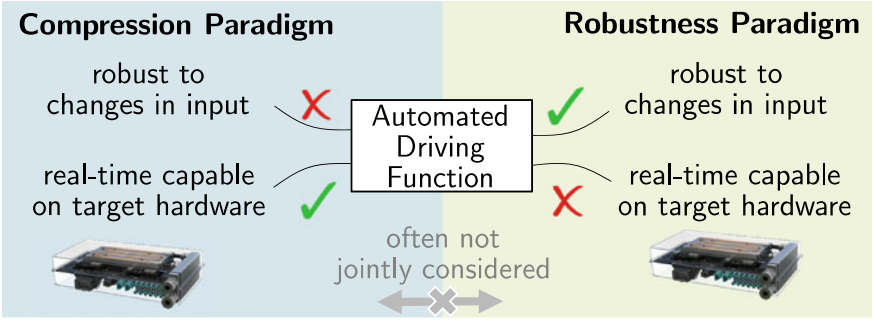

Fig. 1 Automated driving functions underly the two possibly opposing goals of compression and robustness. Approaches in the compression paradigm (*left*) are focused on enabling real-time efficient networks and rarely consider the effect of such a compression on the robustness properties of the network. Similarly, the robustness paradigm (*right*) typically does not consider real-time properties of the network

In this chapter, we tackle the insufficiencies that were mentioned above and introduce hybrid corruption-robustness focused compression (HCRC), an approach to jointly optimize a neural network for achieving network compression along with improvement in corruption robustness. By corruption, we refer to real-world augmentations, such as noise, blur, weather conditions, and digital effects, which are commonly occurring in the real world, and are therefore of significance. Our major contributions in this chapter are described below.

First, HCRC focuses on real-world corruption robustness and proposes a hybrid compression strategy, combining pruning and quantization approaches. We obtain a more robust and compressed network and also perform comparisons with sequential application of robustification and compression methods. Second, we approach the problem of robustness by training with augmentations in a controlled severity fashion. With our method, we show a further improvement under corruption (rPC) not only to the corruptions used during training but also to unseen corruptions including noise and blurring artifacts. Third, since all the methods discussed so far are only evaluated on small datasets for image classification, such as MNIST [LBBH98], CIFAR-10 [Kri09], and SVHN [NWC+11], there remains the question of their transferability to complex tasks, such as semantic segmentation [XWZ+17]. We, for the first time, perform such a study on two road-scenes datasets (Cityscapes [COR+16] and Sim KI-A) and a state-of-the-art semantic segmentation `DeepLabv3+` [CPK+18] network. sss

This chapter is structured as follows: In Sect. 3, we describe the individual components of such a system and our HCRC methodology in detail. In Sect. 4, we describe the corruptions that are used during training and evaluation, the datasets that are used, and the metrics used in the experiments. In Sect. 5, we present our experimental results and observations. Finally, in Sect. 6, we conclude our chapter.

2 Related Works

It is only recently that there have been studies to investigate the interaction between the two techniques, model compression and network robustness that tried to individually address the above-mentioned insufficiencies of DNNs.

Zhao et al. [ZSMA19] report one of the first investigations to empirically study the interactions between adversarial attacks and model compression. The authors observe that a smaller word length (in bits) for weights, and especially activations, makes it harder to attack the network. Building upon the alternating direction method of multipliers (ADMM) framework introduced by Ye et al. [YXL+19], Gui et al. [GWY+19] evaluated the variation of adversarial robustness (FGSM [GSS15] and also PGD [MMS+18]) with a combination of various compression techniques such as pruning, factorization, and quantization. In summary, so far we observe that network compression affects adversarial robustness, and a certain trade-off exists between them. The extent of the trade-off and the working mechanism behind it remains unsolved [WLX+20].

Some works have used compression techniques such as pruning and quantization that were traditionally used to obtain network compression, to improve the

robustness of networks. For example, Lin et al. [LGH19] use quantization not for acceleration of DNNs, but to control the error propagation phenomenon of adversarial attacks by quantizing the filter activations in each layer. On a similar line, Sehwag et al. [SWMJ20] propose to select the filters to be pruned by formulating an empirical minimization problem by incorporating adversarial training (using the PGD attack [MMS+18]) in each pruning step. Very recently, in addition to proposing the new evaluation criterion AER (that stands for accuracy, efficiency, and robustness) for evaluating the robustness and compressibility of networks, Xie et al. [XQXL20] describe a blind adversarial pruning strategy that combines adversarial training along with weight pruning.

In this chapter, we focus on real-world corruption robustness as opposed to the robustness to adversarial attacks. Additionally, we focus on the study of the interactions between robustness, quantization, and pruning methods within our proposed approach, supported by ablation studies.

3 HCRC: A Systematic Approach

Our goal is to improve the robustness of common image corruptions and at the same time reduce the memory footprint of semantic segmentation networks in a systematic way. In this section, we describe our systematic hybrid corruption-robustness focused compression (HCRC) approach to achieve compressed models that are also robust to commonly occurring image corruptions. Our proposed system can be broadly divided into two objectives: the *robustness objective* and the *compression objective*. Both will be described in the following subsections.

3.1 Preliminaries on Semantic Segmentation

We define $\mathbf{x} \in \mathbb{I}^{H \times W \times C}$ to be a clean image of the dataset $\mathcal{X}$, with the image height H, image width W, $C = 3$ color channels, and $\mathbb{I} = [0, 1]$. The image $\mathbf{x}$ is an input to a semantic segmentation network $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ with network parameters $\boldsymbol{\theta}$. Further, we refer to a network layer using an index $\ell \in \mathcal{L} = \{1, \ldots, L\}$, with $\mathcal{L}$ being the set of layer indices. Within layer ℓ we can define $\boldsymbol{\theta}_{\ell,k} \in \mathbb{R}^{H_\ell \times W_\ell}$ to be the kth kernel, where $k \in \mathcal{K}_\ell = \{1, \ldots, K_\ell\}$, with the set of kernel indices $\mathcal{K}_\ell$ of layer ℓ. The image input $\mathbf{x}$ is transformed to class scores by

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{I}^{H \times W \times S}. \tag{1}$$

Each element in $\mathbf{y} = (y_{i,s})$ is a posterior probability $y_{i,s}(\mathbf{x})$ for the class $s \in \mathcal{S} = \{1, 2, \ldots, S\}$ at the pixel position $i \in \mathcal{I} = \{1, \ldots, H \cdot W\}$ of the input image $\mathbf{x}$, and S denoting the number of semantic classes. A segmentation mask $\mathbf{m} = (m_i) \in \mathcal{S}^{H \times W}$ can be obtained from these posterior probabilities with elements

$$m_i = \underset{s \in \mathcal{S}}{\text{argmax}}\ y_{i,s}, \tag{2}$$

by assigning a class to each pixel i. The accuracy of the prediction is evaluated by comparing this obtained segmentation mask $\mathbf{m}$ against the labeled (ground truth) segmentation mask $\overline{\mathbf{m}} \in \overline{\mathcal{M}}$, that has the same dimensions as the segmentation mask $\mathbf{m}$. Likewise, $\overline{\mathbf{y}} \in \{0, 1\}^{H \times W \times S}$ is the one-hot encoded vector ground truth in three-dimensional tensor format that can be retrieved from $\overline{\mathbf{m}}$.

3.2 Robustness Objective

Data augmentation: In Fig. 2, the green data augmentation block on the left depicts the image pre-processing method following Hendryks et al. [HMC+20]. Here, the input image is augmented by mixing randomly sampled corruptions. The key idea is to introduce some amount of randomness in both, the type and the superposition of image corruptions. To achieve this, the input image is first split into three parts and then passed as an input to the data augmenter sub-blocks. Within a data augmenter sub-block, initially, a uniformly sampled corruption $\mathbf{A}_n \in \mathcal{A}^{\text{train}}$ is applied to the input. Here, $\mathcal{A}^{\text{train}} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N\}$ denotes a set of N pre-defined corruption functions $\mathbf{A}_n()$ that are used during training. The corresponding corrupted image is computed as

$$\tilde{\mathbf{x}} = \mathbf{A}_n(\mathbf{x}, \Psi) \in \mathbb{I}^{H \times W \times C}, \tag{3}$$

where $\mathbf{A}_n(\mathbf{x}, \Psi)$ is the image corruption function and Ψ is a parameter controlling the strength of the applied augmentation. This random sampling and augmentation operation is repeated consequently $R = 4$ times within each of the data augmenter sub-blocks.

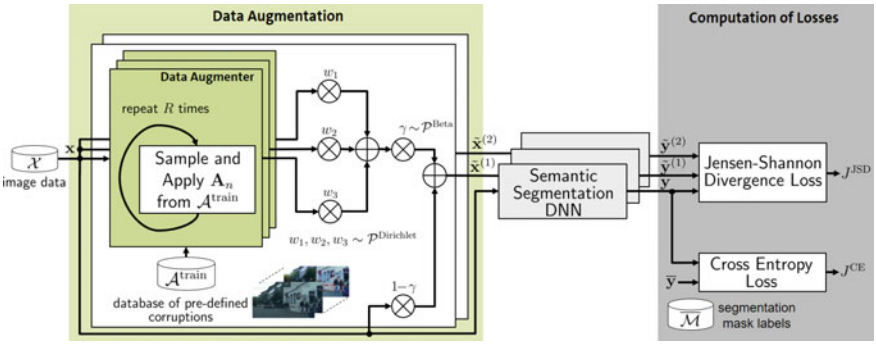

Fig. 2 Overview of the data augmentation strategy (left) and loss construction (right) for a semantic segmentation DNN

The output of each of the N data augmenter sub-blocks is, therefore, an augmented image

$$\tilde{\mathbf{x}}_n = \sum_{r=1}^{R} \tilde{\mathbf{x}}_n^{(r)} = \sum_{r=1}^{R} \mathbf{A}_n^{(r)}(\mathbf{x}, \Psi), \quad n \in \{1, 2, \ldots, N\}, \tag{4}$$

that is a combination of R applications of corruptions from $\mathcal{A}^{\text{train}}$. Choosing $N=3$, these outputs are first passed to multipliers with weights w_1, w_2, and w_3, which are sampled from a Dirichlet distribution $\mathcal{P}^{\text{Dirichlet}}$ and then added. Thereafter, the added output is multiplied by a factor γ that is sampled from a beta distribution $\mathcal{P}^{\text{Beta}}$ with parameters α=1 and β=1. Further, this is added to the input image $\mathbf{x}$, which is multiplied by a factor $1-\gamma$ to obtain the augmented image $\tilde{\mathbf{x}}^{(b)}$, with $b \in \mathcal{B} = \{1, 2, \ldots, B\}$ denoting the index among the B final augmented images being used in our proposed training method. Note that $B+1$ is our minibatch size, where one original image and B augmented images are being employed.

Construction of losses: In Fig. 2, the gray block on the right shows the strategy to construct losses from the predictions of the semantic segmentation network. Following (1), $\tilde{\mathbf{y}}^{(b)}$ denotes the class scores for an augmented input image $\tilde{\mathbf{x}}^{(b)}$. In addition to the aforementioned data augmentation strategy in the pre-processing stage, a loss function with an auxiliary loss term is introduced to enforce regularization between the responses of the semantic segmentation network to clean and augmented images in the training stage. The total loss is defined as

$$J = J^{\text{CE}} + \lambda J^{\text{JSD}}, \tag{5}$$

where J^{CE} is the cross-entropy loss and J^{JSD} is the auxiliary loss, also called the Jenson-Shannon divergence (JSD) loss [MS99]. The λ term is a hyper-parameter introduced to adjust the influence of J^{JSD} on the total loss J. The cross-entropy loss J^{CE} is computed between the posterior probabilities $\mathbf{y}$ of the network conditioned on input $\mathbf{x}$ and its corresponding labels $\overline{\mathbf{y}}$. It is defined as

$$J^{\text{CE}} = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}} \alpha_s \overline{y}_{i,s} \cdot \log(y_{i,s}), \tag{6}$$

by taking a mean over all pixels for the posterior probability $\mathbf{y}$, where α_s are the weights assigned to each class during training, following [WSC+20]. The auxiliary loss, or the Jenson-Shannon Divergence (JSD) loss, is defined as

$$J^{\text{JSD}} = \frac{1}{B+1} \cdot (\text{KL}(\mathbf{y}, \mathring{\mathbf{y}}) + \sum_{b \in \mathcal{B}} \text{KL}(\tilde{\mathbf{y}}^{(b)}, \mathring{\mathbf{y}})). \tag{7}$$

It is computed between the posterior probabilities $\mathbf{y}$ and $\mathring{\mathbf{y}}$ or $\tilde{\mathbf{y}}^{(b)}$, where $b \in \mathcal{B} = \{1, \ldots, B\}$. Note that $\mathring{\mathbf{y}} = \frac{1}{B+1} \cdot (\mathbf{y} + \sum_{b \in \mathcal{B}} \tilde{\mathbf{y}}^{(b)})$ being the mixtures of the probabilities, and $\tilde{\mathbf{y}}^{(b)} = \mathbf{F}(\tilde{\mathbf{x}}^{(b)}, \boldsymbol{\theta})$. The auxiliary JSD loss is introduced to reduce the

variation in the probability distributions of the predictions between a clean input and an augmented input. To do this, two Kullback-Leibler (KL) divergence terms are introduced in (7), e.g.,

$$\mathrm{KL}(\tilde{\mathbf{y}}^{(b)}, \mathring{\mathbf{y}}) = \sum_{i \in \mathcal{I}} \tilde{\mathbf{y}}_i^{(b)} \log \left(\frac{\tilde{\mathbf{y}}_i^{(b)}}{\mathring{\mathbf{y}}_i} \right), \tag{8}$$

defining a distribution-wise measure of how one probability distribution (here: $\tilde{\mathbf{y}}^{(b)}$) differs from the reference mixture distribution (here: $\mathring{\mathbf{y}}_i$).

3.3 Compression Objective

Network pruning: We define a neural network as a particular parameterization of an architecture, i.e., $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ for specific parameters $\boldsymbol{\theta}$. Neural network pruning entails taking as input a model $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta})$ and producing a new network $\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \tilde{\boldsymbol{\theta}})$. Here $\tilde{\boldsymbol{\theta}}$ is the set of parameter values that may be different from $\boldsymbol{\theta}$, but both sets are of the same size $|\boldsymbol{\theta}| = |\tilde{\boldsymbol{\theta}}|$, and $\mathbf{M} \in \{0, 1\}^{|\tilde{\boldsymbol{\theta}}|}$ is a binary mask that forces certain parameters to be 0, while $\odot$ is the element-wise product operator. In practice, rather than using an explicit mask, pruned parameters of $\boldsymbol{\theta}$ are fixed to zero and are removed entirely.

We focus on producing a pruned network $\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \tilde{\boldsymbol{\theta}})$ from a network $\mathbf{F}(\mathbf{x}, \boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_0$ is either sampled from an initialization distribution, or retrieved from a network pretrained on a particular task. Most neural network pruning strategies build upon [HPTD15], where each parameter or structural element in the network is issued a score, and the network is pruned based on these scores. Afterward, as pruning reduces the accuracy of the network, it is trained further (known as fine-tuning) to recover this lost accuracy. The process of pruning and fine-tuning is often iterated several times (iterative pruning) or performed only once (one-shot pruning).

In this chapter, we adopt the magnitude-based pruning approach [HPTD15] that is described in Algorithm 1. Although there exists a large body of more sophisticated scoring algorithms, the gain with such algorithms is marginal, if at all existing [MBKR18]. Based on the number of fine-tuning iterations F^{iter}, the number of filter weights to be pruned F^{pruned} (see Algorithm 1), the total number of prunable filter weights F^{total}, and the type of pruning (see Algorithm 1, Iterative Pruning), the function returns a sparser network $\boldsymbol{\theta}^{\text{pruned}}$ and the binary weight mask $\mathbf{M}$.

Algorithm 1 Magnitude-Based Filter Pruning and Fine-Tuning (Iterative Pruning)

Input: F^{iter}, the number of iterations of fine-tuning,
$\mathcal{X}^{\text{train}}$, the dataset to train and fine-tune,
F^{total}, the total number of prunable filter weights,
F^{pruned}, the number of filter weights to be pruned, and
`IterativePruning`, a boolean. If `true`: iterative pruning, if `false`: one-shot pruning.
$\boldsymbol{\theta} \leftarrow$ `initialize()` ▷ Random/ImageNet pretrained weights initialization
$\boldsymbol{\theta} \leftarrow$ `trainToConvergence`$(\mathbf{F}(\mathbf{x}, \boldsymbol{\theta}))$ ▷ Standard network training
$\mathbf{M} \leftarrow$ `rank`$(\boldsymbol{\theta}, F^{\text{pruned}})$ ▷ Filter weight rank computation (one-shot pruning)
for i in 1 to F^{iter} **do**
  if `IterativePruning` **then**
    $F^{\text{current}} \leftarrow 1 - (F^{\text{pruned}}/F^{\text{total}})^{i/F^{\text{iter}}}$ ▷ Adapt rule of pruning to iterative pruning
    $\mathbf{M} \leftarrow$ `rank`$(\boldsymbol{\theta}^{\text{pruned}}, F^{\text{current}})$ ▷ Updating $\mathbf{M}$
  end if
  $\boldsymbol{\theta}^{\text{pruned}} \leftarrow$ `fineTune`$(\mathbf{F}(\mathbf{x}, \mathbf{M} \odot \boldsymbol{\theta}))$ ▷ Network fine-tuning with sparsed weights
end for
Output: $\boldsymbol{\theta}^{\text{pruned}}$, the pruned network,
$\mathbf{M}$, the binary weight mask vector

Quantization: Low precision fixed-point representations replace floating-point number representations in fixed-point scalar quantization methods. Fixed-point scalar quantization operates on single weights $\theta_{\ell,k,j}$ of the network parameters, where the floating-point format weights are generally replaced by Q-bit fixed-point words [GAGN15], with the extreme case of binarization ($Q = 1$) [CBD15]. We focus on this uniform rounding scheme instead of other non-uniform schemes because it allows for fixed-point arithmetic with implementations in `PyTorch`. Quantization of network weights contributes to a large reduction in the model size and gives possibilities for acceleration on target hardware. In-training quantization (ITQ) refers to training a network by introducing quantization errors (12) in the network weights, and post-training quantization (PTQ) refers to quantizing the weights of a network after the training process by calibrating on the $\mathcal{X}^{\text{train}}$ and/or the $\mathcal{X}^{\text{val}}$ set. Figure 3 gives an overview of the in-training quantization (ITQ) and post-training quantization (PTQ) methods that are used in this chapter. Here, $\boldsymbol{\theta}$ corresponds to the neural network which is the input to the quantization methods, and $\boldsymbol{\theta}^{\text{quant}}$ refers to the fixed-point quantized codewords with lower precision. To do so, in the first block, the statistics for the scale factor

$$\rho_{\ell,k} = \frac{\max_j \theta_{\ell,k,j} - \min_j \theta_{\ell,k,j}}{2^Q - 1}, \tag{9}$$

which defines the spacing between bins, and the bias

$$\delta_{\ell,k} = \text{round}\left(\frac{\min_j \theta_{\ell,k,j}}{\rho_{\ell,k}}\right) \tag{10}$$

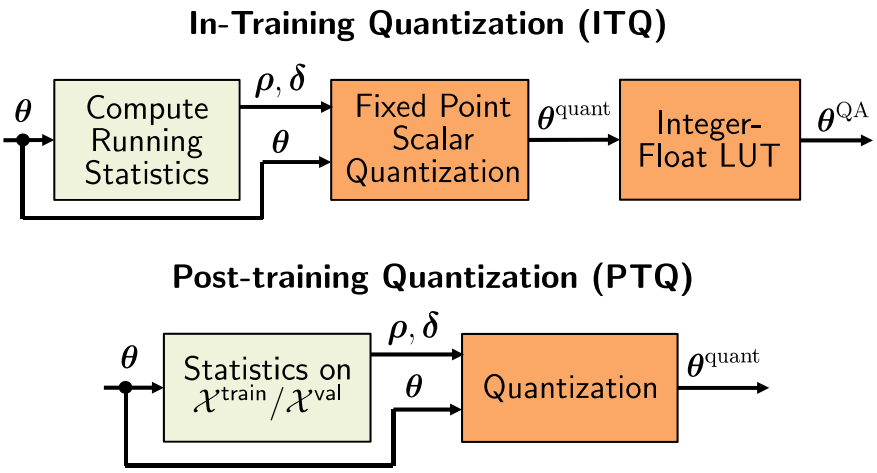

Fig. 3 Overview of quantization methods used in this work. Top: Within the in-training quantization (ITQ), for each iteration, based on the computed scale $\boldsymbol{\rho} = (\rho_{\ell,k})$ (9) and bias $\boldsymbol{\delta} = (\delta_{\ell,k})$ (10) terms, the network parameters $\boldsymbol{\theta}$ are first quantized to $\boldsymbol{\theta}^{\text{quant}}$ (11). Thereafter, each element in $\theta^{\text{quant}}_{\ell,k,j}$ is converted back to its floating-point representation based on a LUT (12). Bottom: Within the post-training quantization (PTQ), similar statistics $(\boldsymbol{\rho}, \boldsymbol{\delta})$ for a trained network are computed on the training and validation set and the network is quantized to $\boldsymbol{\theta}^{\text{quant}}$ (11)

by which the codewords are shifted, are computed. Here, $j \in \mathcal{J}_{\ell,k}$, with $\mathcal{J}_{\ell,k}$ is the set of parameter indices of kernel k in layer ℓ. Thereafter, for both ITQ and PTQ, each weight $\theta_{\ell,k,j}$ is mapped to its closest codeword $\theta^{\text{quant}}_{\ell,k,j}$ by quantizing $\theta_{\ell,k,j}$ using

$$\theta^{\text{quant}}_{\ell,k,j} = \min(\mathrm{q}^{\max}, \max(\mathrm{q}^{\min}, \mathrm{round}(\theta_{\ell,\mathrm{k},\mathrm{j}}/\rho_{\ell,\mathrm{k}} + \delta_{\ell,\mathrm{k}}))). \tag{11}$$

Here, $q^{\min}$ and $q^{\max}$ correspond to the minimum and maximum of the range of quantization levels depending on the chosen Q- bit quantization. For example, for an 8-bit quantization, $q^{\min} = 0$ and $q^{\max} = 255$. For ITQ training, the quantized parameters $\theta^{\text{quant}}_{\ell,k,j}$ are converted back to floating-point representation $\theta^{\text{QA}}_{\ell,k,j}$ based on an integer-float lookup table (LUT) following

$$\theta^{\text{QA}}_{\ell,k,j} = (\theta^{\text{quant}}_{\ell,k,i} - \delta_{\ell,k}) \cdot \rho_{\ell,k}. \tag{12}$$

This means that quantization errors are introduced within the network parameters $\boldsymbol{\theta}^{\text{QA}}$, which are then used within the training process. For PTQ, the quantized parameters $\boldsymbol{\theta}^{\text{quant}}$ are directly used in the evaluation of the semantic segmentation network.

In this chapter, we focus on the uniform rounding scheme instead of other non-uniform schemes, because it allows for fixed-point arithmetic with implementations in `PyTorch`. Throughout this chapter, we use a strong quantization of $Q = 8$ bits to enable higher acceleration on edge devices.

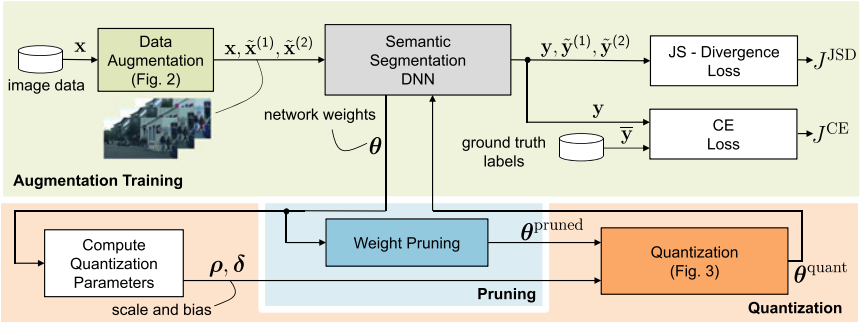

Fig. 4 Overview of our training strategy to co-optimize for corruption robustness along with network compression by the use of augmentation training, weight pruning, and quantization methods

3.4 HCRC Core Method

Within the HCRC framework, we systematically combine the robustness (Sect. 3.2), pruning, and quantization (Sect. 3.3) methods to co-optimize both robustness and compression objectives. Figure 4 gives an overview of our training strategy. The green block on the top depicts the augmentation strategy of the input image data and the consequent construction of losses. For each input image $\mathbf{x}$, the augmented images $\tilde{\mathbf{x}}^{(b)}$ are initially computed and passed to the semantic segmentation network. The total loss (5) is then computed based on the clean and corrupted image predictions. The orange block on the bottom depicts the quantization of the network weights and activations and the blue block contains the pruning module. We start by initializing the network parameters. In each training iteration, the scale factor (9) and the bias (10) are computed, and the network parameters are quantized (11). Additionally, in each training epoch, we use iterative pruning which continually prunes a certain percentage of the weights of the network (see blue block in Fig. 4).

4 Experimental Setup

In this section, the details of the semantic segmentation network, the road-scenes datasets, and the semantic segmentation networks that have been used in this chapter are initially described. The image corruptions that are applied in both phases, training and evaluation, are then introduced. Finally, the evaluation metrics are described.

4.1 Datasets and Semantic Segmentation Network

Our dataset splits are summarized in Table 1. For Cityscapes [COR+16], the baseline networks are trained with the 2,975 images of the training set $\mathcal{X}_{\text{CS}}^{\text{train}}$. Due to the Cityscapes test set upload restrictions, we split the official validation set into two sets-a mini validation set $\mathcal{X}_{\text{CS}}^{\text{val}}$ (Lindau, 59 images) and a mini test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ (Frankfurt and Münster, 441 images). The images have a resolution of 2,048×1,024. The Sim KI-A dataset is an artificially generated dataset with 4,257 training ($\mathcal{X}_{\text{Sim}}^{\text{train}}$), 387 validation ($\mathcal{X}_{\text{Sim}}^{\text{val}}$) and 387 test ($\mathcal{X}_{\text{Sim}}^{\text{test}}$) images. The images have a resolution of 1,920×1,080.

In this chapter, we use the `DeepLabv3+` [CBLR18] semantic segmentation network with `ResNet-101` backbone [HZRS16]. For both datasets, the baseline network, that is the network without any augmentation or compression, is trained with a crop size of 513 × 513 and a batch size of 4 on an `Nvidia Tesla V100` GPU. The class frequency-weighted cross-entropy loss J^{CE} (6) in combination with stochastic gradient descent (SGD) are used as optimization criterion and optimizer, respectively. During training, a polynomial learning rate scheme with an initial learning rate of 0.01 and a power of 0.9 is applied. The network is trained to convergence for 100 epochs on the Cityscapes dataset and 50 epochs for the Sim KI-A dataset. For a fair comparison, all the networks are evaluated on an `Intel(R) Xeon(R) Gold 6148` CPU.

4.2 Image Corruptions

The images corruptions used in this chapter are described in Table 2. These corruptions are split into two different categories depending on their usage, i.e., either

Table 1 Details of the road-scenes datasets used in the experiments. The image resolution of the dataset images and split into training, validation, and test sets are described

Dataset	Resolution	$\mathcal{X}^{\text{train}}$	$\mathcal{X}^{\text{val}}$	$\mathcal{X}^{\text{test}}$
Cityscapes [COR+16]	2,048 × 1,048	2,975	59	441
Sim KI-A	1,920 × 1,080	4,257	387	387

Table 2 Types of image corruptions used in this work that are arranged in two categories, based on their usage in either the training $\mathcal{A}^{\text{train}}$ or test $\mathcal{A}^{\text{test}}$ phases

Category	Corruption type
$\mathcal{A}^{\text{train}}$	Auto-contrast, equalize, posterize, color, sharpness, Gaussian blur, spatter, saturation
$\mathcal{A}^{\text{test}}$	Gaussian noise, shot noise, impulse noise, defocus blur, frosted glass blur, motion blur, zoom blur

Table 3 Corruptions and their parameterization used during training are listed. A dash (-) indicates that the corruption function is image-dependent and does not need any parameterization. An interval [a, b] indicates that the respective parameter is a real number $\mathbb{R}$ sampled uniformly from this interval

Corruption type	Auto-contrast	Equalize	Posterize	Color
Parameterization	–	–	[0, 4.0]	[0.25, 4.0]
Corruption type	Sharpness	Gaussian blur	Spatter	Saturation
Parameterization	[0.25, 4.0]	8	{0.65, 0.5, 0.3, 0.7, 0.65}	{1.5, 0.1}

during training or during test. The corruptions $\mathcal{A}^{\text{test}}$ in the test process are adopted from the neural network robustness benchmark[1] from [HD19]. The corruptions $\mathcal{A}^{\text{train}}$ used in the training process are adopted following a large body of work [HMC+20, CZM+19, TPL+19, SK19] that use these corruptions in different ways for training with data augmentation. Table 3 gives an overview of the parameterization of each corruption within $\mathcal{A}^{\text{train}}$. For spatter corruption, the list of parameters corresponds to the location, scale, two sigma, and threshold values, respectively. For saturation corruption, the list of parameters corresponds to the amount of saturation and the scale. For posterize, color, and sharpness corruptions, the parameter is sampled from within the given interval.

Definition of severities: Various kinds of data augmentations exist, and it is rather difficult to compare between different augmentation types, although first attempts are known [KBFs20]. Let us take an example of brightness and contrast augmentations. We can increase or decrease the brightness and contrast values for a given input image by manipulating the image pixel values. An increase in the brightness and an increase in the contrast do not necessarily correspond to the same effect on the input image. To standardize the method of measuring the strength of augmentations irrespective of the augmentation type, the structural similarity (SSIM) metric [WBSS04] is used. To do this, SSIM is computed between the clean input image $\mathbf{x}$ and the augmented image $\tilde{\mathbf{x}}^{(b)}$. Here, $\text{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}^{(b)}) = 0$ indicates that the image $\mathbf{x}$ and the corresponding augmented image $\tilde{\mathbf{x}}^{(b)}$ are completely dissimilar. Similarly, $\text{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}^{(b)}) = 1$ indicates that the image $\mathbf{x}$ and the corresponding augmented image $\tilde{\mathbf{x}}^{(b)}$ are identical, or no augmentation is applied. We define severity levels (V) to indicate the strength of the augmentation. Severity level $V = 0$ indicates that no type of augmentation is applied to the input image and severity level $V = 10$ indicates that the input image is completely dissimilar after the augmentation. This means that for every increase in level in V, the SSIM between the clean input image and the augmented image reduces by 0.1. To control the severity of the data augmentation during training, the parameters (α, β) of the γ-function are varied (see Fig. 2) by keeping the corruption parameters constant following Table 2.

[1] https://github.com/hendrycks/robustness.

4.3 Metrics

Mean intersection-over-union (mIoU) between the predictions of the semantic segmentation network and the human-annotated ground truth labels is commonly used for evaluating semantic segmentation networks. The mIoU is defined as

$$\text{mIoU} = \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{\text{TP(s)}}{\text{TP(s)} + \text{FP(s)} + \text{FN(s)}} = \text{mIoU}(\mathbf{y}, \bar{\mathbf{y}}), \tag{13}$$

where TP(s), FP(s), and FN(s) are the class-specific true positives, false positives, and false negatives, respectively, computed between segmentation output $\mathbf{y}$ and ground truth one-hot encoded segmentation $\bar{\mathbf{y}}$.

Mean performance under corruption (mPC) has been introduced by [HD19] for evaluating the robustness of neural networks under varying corruptions and varying strengths. For this purpose, the individual augmentations $\mathbf{A}_n \in \mathcal{A}^{\text{test}}$ are further sub-divided with respect to the strength of the augmentations. We use the augmentations $\mathcal{A}^{\text{test}}$ (see Table 2) for the computation of mPC. This is computed by

$$\text{mPC} = \frac{1}{|\mathcal{A}^{\text{test}}|} \sum_{c=1}^{|\mathcal{A}^{\text{test}}|} \frac{1}{N_c} \sum_{V=1}^{N_c} \text{mIoU}_{c,V}, \tag{14}$$

with corruption index c and N_c denoting the amount of severity conditions for a corruption c. Here, $\text{mIoU}_{c,V}$ denotes the mIoU (13) of the model under the corruption c and severity V. The key factor here is choosing the severities, as this can vary on different datasets, and even models, depending on the selection criteria. In this chapter, we use the SSIM metric [HD19] as a means of finding severity thresholds. Using this metric allows for standardized severities across a dataset, as it is task- and model-agnostic. Thus, different robustness improvement methods can be benchmarked and compared easily using the mPC metric.

Relative performance under corruption (rPC) is simply the ratio of the mPC and the mIoU of the semantic segmentation under the corruptions during evaluation, and is defined as

$$\text{rPC} = \frac{\text{mPC}}{\text{mIoU}}. \tag{15}$$

4.4 Training Framework

For the task of achieving robust and compressed semantic segmentation networks, one can envision various different ways to approach it. An overview of all possible approaches is given in Fig. 5. For all the reference models, we start from the pre-trained checkpoint weights of the `ResNet-101` backbone for the `DeepLabv3+` architecture.

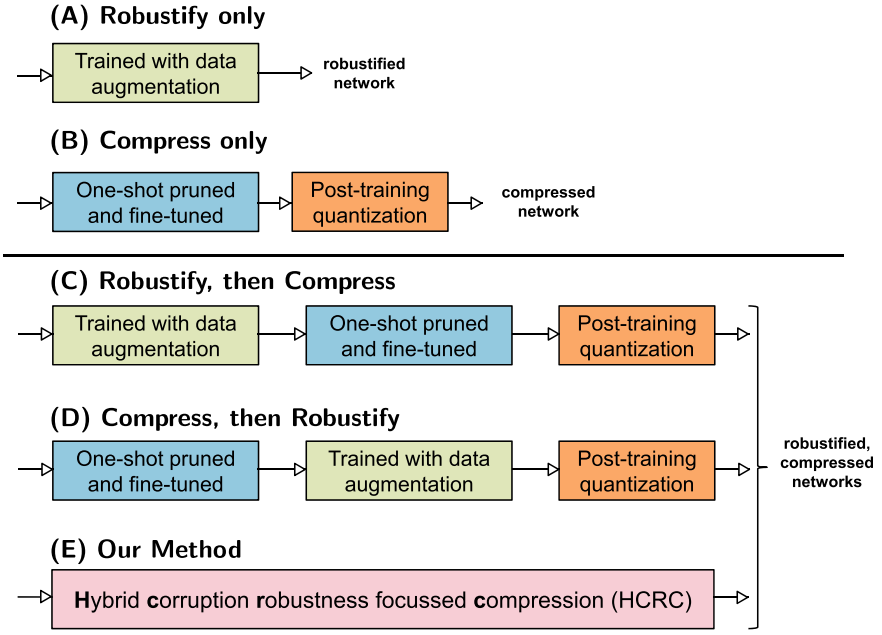

Fig. 5 The training approaches used in this work are depicted. In addition to the simple baselines (Reference A, and Reference B), we compare our HCRC approach against sequential applications (Reference C, and Reference D) of the individual steps in the training framework

Reference A: In this approach, the `DeepLabv3+` network with the `ResNet-101` backbone is trained for improving its corruption robustness. Here, no compression techniques are applied. The network is trained using the protocol defined in Sect. 4.1 with the total loss (5) and $\lambda = 10^{-6}$.

Reference B: Here, the `DeepLabv3+` undergoes one-shot pruning (see Algorithm 1). First, the network is trained using the protocol defined in Sect. 4.1 with the class frequency weighted cross-entropy loss (6). Next, the statistics (9), (10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3). No robustness-related training is enforced.

Reference C: In this configuration, we perform sequential application of the robustness and compression goals. In the first step, the `DeepLabv3+` is trained using the protocol defined in Sect. 4.1 with the total loss (5) and $\lambda = 10^{-6}$ (see also Reference A) in combination with the data augmentation strategy described in Sect. 3.2 (see Fig. 2). Next, the network undergoes iterative pruning (see Algorithm 1) following again the protocol defined in Sect. 4.1, this time with the class frequency weighted cross-entropy loss (6). Finally, statistics (9), (10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3).

Reference D: In this setup, the `DeepLabv3+`, the network is first one-shot pruned (see Algorithm 1) and then fine-tuned following the protocol defined in Sect. 4.1 using

the class frequency weighted cross-entropy loss (6). In the next step, the network is trained with the data augmentation strategy described in Sect. 3.2 (see Fig. 2) following again the protocol defined in Sect. 4.1, however, this time the total loss J (5) is used as the optimization criterion. Finally, statistics (9, 10) are computed on the $\mathcal{X}^{\text{train}}$ and $\mathcal{X}^{\text{val}}$ set and the network undergoes PTQ (see Fig. 3).

5 Experimental Results and Discussion

5.1 Ablation Studies

In-training quantization (ITQ) vs. post-training quantization (PTQ) We hypothesize that training the network with quantization errors is better than quantizing the network after training.

In Table 4, we compare these two approaches of achieving quantized networks. The baseline `DeepLabv3+` network has an mIoU of 69.78% and mPC of 44.03%. On one hand, we observe that mIoU drops by 5.35% and mPC by 2.76% (both: absolute) after PTQ. On the other hand, the drop in mIoU is only 1.8% after ITQ, within no change in mPC. This result supports the abovementioned hypothesis on quantization, that in-training quantization is superior to post-training quantization. Additionally, we increased the size of the calibration set used within PTQ by also including $\mathcal{X}^{\text{val}}$ along with $\mathcal{X}^{\text{train}}$. This, however, resulted in no significant changes in the performance of the quantized networks.

Controlled severity training: The semantic segmentation network is evaluated over various corruptions and various severity levels. From our initial experiments, we observed that the data corruptions used during the training process [HMC+20] have a mean severity level of $V = 1$. It is intuitive that a network trained on data augmentations of higher severity should be, in theory, more robust to higher severity corruptions during test. To study the effect of the training severity on the robustness of the trained semantic segmentation network, we train the `DeepLabv3+` network with three different severities of corruption. To do this, we vary the parameters of the

Table 4 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation of mIoU, mPC, and rPC comparing the non-quantized `DeepLabv3+` network, and the corresponding PTQ and ITQ networks that are trained to convergence for 100 epochs. Note that the inference times are computed on the `Intel(R) Xeon(R) Gold 6148` CPU. Best numbers reported in **bold**

Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Time (s)
`DeepLabv3+`	69.78	44.03	63.09	5.23
with PTQ	64.43	41.27	64.05	**2.66**
with ITQ	**67.98**	**44.04**	**64.78**	**2.66**

Table 5 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation based on mIoU, mPC, and rPC comparing the three DeepLabv3+ networks trained on augmentations of three different severity levels. Note that the networks are not subjected to any kind of compression. Best numbers reported in **bold**

Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)
DeepLabv3+	69.78	44.03	63.09
Trained with severity level $V = 1$	**69.98**	53.65	76.66
Trained with severity level $V = 2$	69.54	56.14	80.73
Trained with severity level $V = 3$	69.44	**56.27**	**81.03**

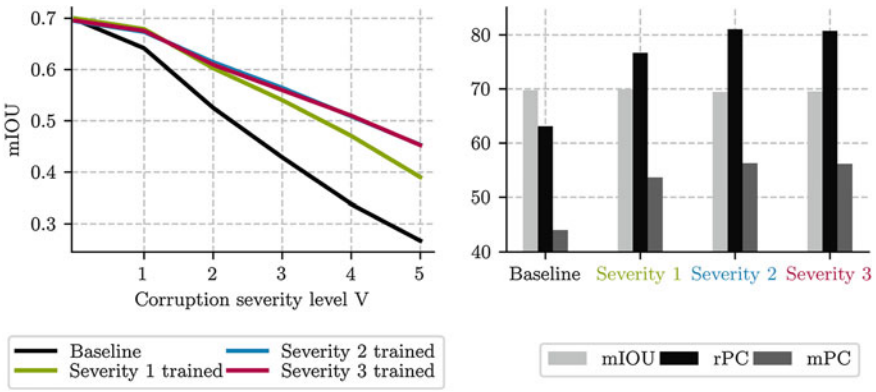

Fig. 6 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation for the DeepLabv3+ network trained with different corruption severities. Left: The four networks are evaluated over the augmentations $\mathcal{A}^{\text{test}}$ with six severity levels (x-axis, severities $V = 0, 1, \ldots, 5$). Right: The bar chart shows the mIoU on clean data ($V = 0$), as well as mPC and rPC, computed over the same six severity levels ($V = 0, 1, \ldots, 5$), for four networks trained with severity $V = 0$ (baseline), 1, 2, and 3

beta distribution to increase the influence of the individual corruptions. The results are shown in Table 5.

We generally observe that training with higher severities leads to higher robustness in terms of the mPC. The DeepLabv3+ network trained with a severity level of 3 has an increase of 2.62% absolute mPC and 4.36% absolute rPC when evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$. In Fig. 6, we show the results of evaluating these networks on six different severity values. The networks trained with higher severities show higher robustness, especially when evaluated on higher severities ($V \geq 3$). Training with a higher severity ($V \geq 4$) did not show any further improvements. A drop in the mIoU indicates a certain trade-off between an increase in the generalization (to unseen corruptions) capability of the network to a decrease in its performance on the clean (or vanilla) input.

Table 6 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation based on mIoU, mPC, and rPC comparing the one-shot and iterative pruning approaches for the `DeepLabv3+` network. All the networks are trained with augmentations of severity level 2. A $Q = 8$ bits quantization is applied to all the HCRC trainings. Best numbers reported in **bold**

Method	Pruning Ratio [%]	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)
`DeepLabv3+`	0	69.78	44.03	63.09
HCRC with one-shot pruning	30	66.06	50.80	76.89
HCRC with iterative pruning	30	**68.12**	**52.50**	**77.07**
HCRC with one-shot pruning	50	64.58	49.45	76.57
HCRC with iterative pruning	50	**66.84**	**51.95**	**77.72**

Sensitivity of the pruning algorithm: We perform an ablation study to analyze the effect of the types of pruning methodology within our HCRC approach. To this end, we train the `DeepLabv3+` network in a combined fashion (see Sect. 3.4) with two different types of pruning, namely, one-shot pruning and iterative pruning. In Table 6, we provide the evaluation results of this study over two different pruning ratios (30% and 50%). A pruning ratio of 30% indicates that 30% of the prunable weights are removed from the network, while 70% are remaining. For quantization, within all our experiments, we have used a strong quantization of $Q = 8$ bits.

For 30% pruning ratio, we observe that the iterative pruning method shows an (absolute) increase in mIoU (2.06%), mPC (1.7%), and rPC (0.18%), when compared to the one-shot pruning method, evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$. For 50% pruning ratio, we observe similar (absolute) improvements for iterative pruning in its mIoU (2.26%), mPC (2.5%), and rPC (1.15%), computed over $\mathcal{X}_{\text{CS}}^{\text{test}}$.

5.2 Comparison With Reference Baselines

In this section, we compare our HCRC method (with iterative pruning) with the reference methods (see Sect. 4.4). In particular, we compare our HCRC method against Reference C and Reference D, which also aim to achieve robust and compressed segmentation networks.

For the Cityscapes dataset, the results of the evaluation are shown in Table 7. We observe that our HCRC method outperforms all the relevant reference methods for both pruning ratios. The reference A network with a pruning ratio of 0% shows an improvement of 11.62% absolute mPC over the `DeepLabv3+` baseline network with a slight improvement in the mIoU. For 30% pruning, the HCRC shows significant improvements over the reference methods B, C, and D. The HCRC method shows an improvement of 3.29% absolute mIoU and 9.14% absolute mPC over the best reference (Reference D). Additionally, HCRC improves the robustness of the `DeepLabv3+` network by 8.47% absolute mPC with a 77.67% reduction in the

Table 7 Test set $\mathcal{X}_{\text{CS}}^{\text{test}}$ evaluation comparing HCRC to reference methods A–D. A quantization with $Q = 8$ bits quantization is applied to all trainings where compression is applied. Best numbers reported in **bold**

Pruning Ratio [%]	Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Model Size (MB)
0	`DeepLabv3+`	69.78	44.03	63.10	237.38
	Reference A	**69.98**	**53.65**	**76.67**	237.38
30	Reference B	62.56	35.79	57.21	52.33
	Reference C	64.28	37.13	57.76	52.33
	Reference D	64.83	43.36	66.88	52.33
	HCRC (ours)	**68.12**	**52.50**	**77.07**	52.33
50	Reference B	64.48	37.50	58.16	47.47
	Reference C	66.05	36.52	55.29	47.47
	Reference D	66.03	48.52	73.48	47.47
	HCRC (ours)	**66.16**	**51.95**	**77.72**	47.47

model size. For 50% pruning ratio, we observe similar improvements in HCRC over the reference methods B, C, and D. The HCRC method shows an improvement in mIoU (2.13%) and mPC (2.91%) when evaluated on $\mathcal{X}_{\text{CS}}^{\text{test}}$ and when compared to the best reference (reference D). Overall, HCRC with pruning ratio of 50% improves the robustness of the `DeepLabv3+` network by 7.92% absolute mPC with an almost 80% reduction in the model size.

For the Sim KI-A dataset, we similarly observe that our HCRC method outperforms all the relevant reference baselines for both the pruning ratios, see Table 8. The reference A network with a pruning ratio of 0% shows an improvement of 12.98% absolute mPC over the `DeepLabv3+` baseline network with a slight improvement in the mIoU. For 30% pruning, the HCRC shows significant improvements over the reference methods B, C, and D. The HCRC method shows an improvement of 2.33% absolute mIoU and 5.09% absolute mPC over the best reference (Reference D). For 50% pruning ratio, we observe similar improvements in HCRC over the reference methods B, C, and D. The HCRC method shows an improvement in mIoU (1.04%) and mPC (3.68%) when evaluated on $\mathcal{X}_{\text{Sim}}^{\text{test}}$ and when compared to the best reference (reference D). Overall, HCRC with pruning ratio of 50% improves the robustness of the `DeepLabv3+` network by 7.60% absolute mPC with an almost 80% reduction in the model size.

Interestingly, the clean performance of our compressed HCRC network is nearly the same as the uncompressed `DeepLabv3+` baseline, albeit with much improved robustness. We also show qualitative results in Fig. 7 for impulse noise of severity level $V = 3$, where we observe a significant improvement over the simpler reference B baseline. In summary, our proposed HCRC approach to co-optimize for corruption robustness and model compression outperforms all possible reference baselines and produces a network that is heavily compressed and robust to unseen and commonly occurring image corruptions.

Table 8 Test set $\mathcal{X}_{\text{Sim}}^{\text{test}}$ evaluation comparing HCRC to reference methods A–D. A quantization with $Q = 8$ bits is applied to all trainings where compression is applied. Best numbers reported in **bold**

Pruning Ratio [%]	Method	mIoU [%] (13)	mPC [%] (14)	rPC [%] (15)	Model Size (MB)
0	`DeepLabv3+`	**77.57**	54.42	70.16	237.38
	Reference A	77.05	**67.40**	**87.48**	237.38
30	Reference B	74.19	49.78	67.10	52.33
	Reference C	72.44	51.56	71.17	52.33
	Reference D	73.90	58.70	79.43	52.33
	HCRC (ours)	**76.23**	**63.79**	**83.68**	52.33
50	Reference B	75.65	47.60	62.92	47.47
	Reference C	75.08	47.82	63.69	47.47
	Reference D	74.38	58.34	78.43	47.47
	HCRC (ours)	**76.12**	**62.02**	**81.48**	47.47

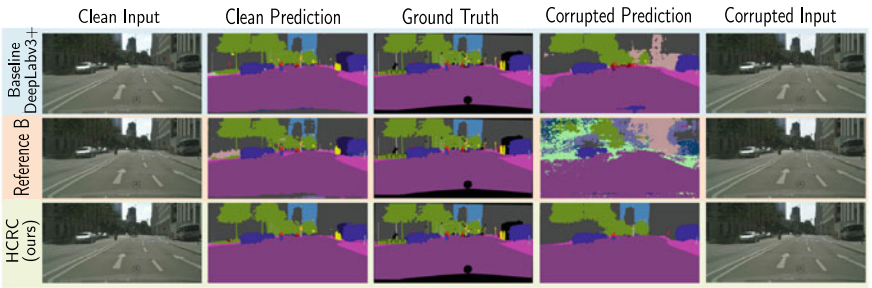

Fig. 7 Example segmentations on the Cityscapes dataset. We show a snippet from $\mathcal{X}_{\text{CS}}^{\text{test}}$, where the differences in the robustness of the compressed networks are more pronounced. We observe that our HCRC method is compressed and has superior robustness to the `DeepLabv3+` baseline and the compressed network of reference B, in this example, for impulse noise corruption

6 Conclusions

In this chapter, we introduce hybrid corruption-robustness focused compression (HCRC), an approach to jointly optimize a neural network for achieving network compression along with improvement in corruption robustness, such as noise and blurring artifacts, which are commonly observed. For this study, we consider the task of semantic segmentation for automated driving and look at the interactions between robustness and compression of networks. HCRC improves the robustness of the `DeepLabv3+` network by 8.47% absolute mean performance under corruption (mPC) on the Cityscapes dataset and 7.60% absolute mPC on the Sim KI-A dataset and generalizes even to augmentations not seen by the network in the training process. This is achieved with only minor degradations on undisturbed data.

Our approach is evaluated over two strong compression ratios and consistently outperforms all considered baseline approaches. Additionally, we perform extensive ablation studies to further leverage and extend existing state-of-the-art methods.

Acknowledgements The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI-basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI Absicherung)". The authors would like to thank the consortium for the successful cooperation.

References

[AMT18] A. Arnab, O. Miksik, P.H.S. Torr, On the robustness of semantic segmentation models to adversarial attacks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 888–897

[BHSFs19] A. Bär, F. Hüger, P. Schlicht, T. Fingscheidt, On the robustness of redundant teacher-student frameworks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (Long Beach, CA, USA, 2019), pp. 1380–1388

[BKV+20] A. Bär, M. Klingner, S. Varghese, F. Hüger, P. Schlicht, T. Fingscheidt, Robust semantic segmentation by redundant networks with a layer-specific loss contribution and majority vote, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, virtual conference (2020), pp. 1348–1358

[CBD15] M. Courbariaux, Y. Bengio, J-P. David, Binary connect: training deep neural networks with binary weights during propagations, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2015), pp. 3123–3133

[CBLR18] Z. Chen, V. Badrinarayanan, C-Y. Lee, A. Rabinovich, GradNorm: gradient normalization for adaptive loss balancing in deep multitask networks, in *Proceedings of the International Conference on Machine Learning (ICML)* (Stockholm, Sweden, 2018), pp. 794–803

[CLW+16] Y. Cao, M. Long, J. Wang, H. Zhu, Q. Wen, Deep quantization network for efficient image retrieval, in *Proceedings of the AAAI Conference on Artificial Intelligence* (Phoenix, AZ, USA, 2016), pp. 3457–3463

[COR+16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The Cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 3213–3223

[CPK+18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)

[CZM+19] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q.V. Le, AutoAugment: learning augmentation strategies from data, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA, USA, 2019), pp. 113–123

[DFY+20] Y. Dong, Q-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, J. Zhu, Benchmarking adversarial robustness on image classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 1322–1330

[GAGN15] S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, in *Proceedings of the International Conference on Machine Learning (ICML)* (Lille, France, 2015), pp. 1737–1746

[GSS15] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in *Proceedings of the International Conference on Learning Representations (ICLR)* (San Diego, CA, USA, 2015), pp. 1–11

[GWY+19] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, J. Liu, Model compression with adversarial robustness: a unified optimization framework, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Vancouver, BC, Canada, 2019), pp. 1283–1294

[HD19] D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–15

[HMC+20] D. Hendrycks, N. Mu, E.D. Cubuk, B. Zoph, J. Gilmer, B. Lakshminarayanan, Augmix: a simple data processing method to improve robustness and uncertainty, in *Proceedings of the International Conference on Learning Representations (ICLR)*, virtual conference (2020), pp. 1–15

[HMD16] S. Han, H. Mao, W.J. Dally, Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding, in *proceedings of the international conference on learning representations (ICLR)* (2016), pp. 1–14

[HPTD15] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2015), pp. 1135–1143

[HVD14] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops (Montréal, QC, Canada, 2014)*, pp. 1–9

[HZRS16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA, 2016), pp. 770–778,

[HZS17] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, 2017), pp. 1398–1406

[JGWD19] S. Jain, A. Gural, Mi. Wu, C. Dick, Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware (2019), pp. 1–17, arXiv:1903.08066

[JKC+18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A.G. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT, USA, 2018), pp. 2704–2713

[KBFs20] M. Klingner, A. Bär, T. Fingscheidt, Improved noise and attack robustness for semantic segmentation by using multi-task training with self-supervised depth estimation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* , virtual conference (2020), pp. 1299–1309

[KBK20] I. Kim, W. Baek, S. Kim, Spatially attentive output layer for image classification, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 9533–9542

[Kri09] A. Krizhevsky, *Object Classification Experiments* (Technical report, Canadian Institute for Advanced Research, April 2009)

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. IEEE **86**(11), 2278–2324 (1998)

[LGH19] J. Lin, C.Gan, S. Han, Defensive quantization: when efficiency meets robustness, in *Proceedings of the International Conference on Learning Representations (ICLR)* (New Orleans, LA, USA, 2019), pp. 1–15

[MBKR18] D. Mittal, S. Bhardwaj, M. Khapra, B. Ravindran, Recovering from random pruning: on the plasticity of deep convolutional neural networks, in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)* (Lake Tahoe, NV, USA, 2018), pp. 848–857

[MMS+18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Vancouver, BC, Canada, 2018), pp. 1–10

[MS99] D. Christopher, *Manning and Hinrich Schütze* (MIT Press, Foundations of Statistical Natural Language Processing, 1999)

[MTK+17] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in *Proceedings of the International Conference on Learning Representations (ICLR)* (Toulon, France, 2017), pp. 1–17

[NWC+11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS) Workshops* (Granada, Spain, 2011), pp. 1–9

[SK19] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning. J. Big Data **60**(6), 1–48 (2019)

[SVS19] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks. IEEE Trans. Evolut. Comput. (TEVC) **23**(5), 828–841 (2019)

[SWMJ20] V. Sehwag, S. Wang, P. Mittal, S. Jana, HYDRA: pruning adversarially robust neural networks (2020), pp. 1–22. arXiv:2002.10509

[TKTH18] L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster Gaze Prediction With Dense Networks and Fisher Pruning (2018), pp. 1–18. arXiv:1801.05787

[TPL+19] Z. Tang, X. Peng, T. Li, Y. Zhu, D. Metaxas, Adatrans form: adaptive data transformation, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, 2019), pp. 2998–3006

[WBSS04] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

[WLX+20] S. Wang, N. Liao, L. Xiang, N. Ye, Q. Zhang, Achieving Adversarial Robustness via Sparsity (2020), pp. 1–9, arXiv:2009.05423

[WSC+20] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, M. Yadong, M. Tan, X. Wang et al., Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **43**(10), 3349–3364 (2020)

[XLZ+18] C. Xiao, B. Li, J-Y. Zhu, W. He, M. Liu, D. Song, Generating adversarial examples with adversarial networks (2018), pp. 1–8, arXiv:1801.02610

[XQXL20] H. Xie, L. Qian, X.g Xiang, N. Liu, Blind adversarial pruning: balance accuracy, efficiency and robustness (2020), pp. 1–12. arXiv: 2004.05913

[XWZ+17] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, A. Yuille, Adversarial examples for semantic segmentation and object detection, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy, 2017), pp. 1369–1378

[YXC20] M. Ye, S. Xu, T. Cao, HVNet: hybrid voxel network for LiDAR based 3d object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, virtual conference (2020), pp. 1631–1640

[YXL+19] S. Ye, K. Xu, S. Liu, H. Cheng, J.H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, X. Lin, Adversarial robustness vs. model compression, or both? in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Seoul, Korea, October 2019), pp. 111–120

[ZHZ11] M. Zhou, M. Huang, X. Zhu, Robust reading comprehension with linguistic constraints via posterior regularization. IEEE/ACM Trans. Audio, Speech, Lang. Process. **20**(4), 1096–1108 (2011)

[ZS18] Z. Zhang, M.R. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS/NeurIPS)* (Montréal, QC, Canada, 2018), pp. 8792–8802

[ZSMA19] Y. Zhao, I. Shumailov, R. Mullins, R. Anderson, To compress or not to compress: understanding the interactions between adversarial attacks and neural network compression, in *Proceedings of the Conference on Machine Learning and Systems (MLSys)* (Stanford, CA, USA, 2019), pp. 230–240