# WAVELET FEATURE MAPS COMPRESSION FOR LOW BANDWIDTH CONVOLUTIONAL NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Quantization is one of the most effective techniques for compressing Convolutional Neural Networks (CNNs), which are known for requiring extensive computational resources. However, aggressive quantization may cause severe degradation in the prediction accuracy of such networks, especially in image-to-image tasks such as semantic segmentation and depth prediction. In this paper, we propose Wavelet Compressed Convolution (WCC)—a novel approach for activation maps compression for $1 \times 1$ convolutions (the workhorse of modern CNNs). WCC achieves compression ratios and computational savings that are equivalent to low bit quantization rates at a relatively minimal loss of accuracy. To this end, we use a hardware-friendly Haar-wavelet transform, known for its effectiveness in image compression, and define the convolution on the compressed activation map. WCC can be utilized with any $1 \times 1$ convolution in an existing network architecture. By combining WCC with light quantization, we show that we achieve compression rates equal to 2-bit and 1-bit with minimal degradation in image-to-image tasks.

## 1 INTRODUCTION

Over the past years, Convolutional Neural Networks (CNNs) have brought significant improvement in processing images, video, and audio (LeCun et al., 2015; Krizhevsky et al., 2017). However, CNNs require significant computational and memory costs, which makes the usage of CNNs difficult in applications where computing power is limited, e.g., on edge devices. To address this limitation, several approaches have been proposed to reduce the computational costs of neural networks. Among the popular ones are weight pruning (Han et al., 2015; Guo et al., 2016), architecture search (Howard et al., 2019), and quantization (Li et al., 2017; Banner et al., 2018a). In principle, all these approaches can be applied simultaneously on top of each other to reduce the computational costs of CNNs.

In this work, we focus on quantization. This approach relieves the computational cost of CNNs by quantizing their weights and activation (feature) maps using low numerical precision so that they can be stored and applied as fixed point integers (Hubara et al., 2017; Banner et al., 2018a). In particular, it is common to apply aggressive quantization (less than 4-bit precision) to compress the activation maps (Esser et al., 2020). Yet, it is known that compressing natural images using uniform quantization is sub-optimal. Indeed, applying aggressive quantization in certain CNNs can lead to significant degradation in the accuracy of the network. The impact is especially evident for the image-to-image tasks such as semantic segmentation (Tang et al., 2019) and depth prediction (Lee et al., 2020), where each pixel has to be assigned a value. In a recent work that targets quantized U-Nets (Tang et al., 2019), activations bit rates are kept relatively high (about 8 bits) while the weight bit rates are lower (down to 2 bits). Beyond that, we note that the majority of the quantization works are applied and tested on image classification (Esser et al., 2020; Li et al., 2020, and references therein).

This work aims to improve the compression of the activation maps by introducing Wavelet Compressed Convolution (WCC) layers. These layers utilize the Haar-wavelet transform (Daubechies, 1992) to compress the activation maps before applying convolutions. This wavelet transform and its inverse can be applied efficiently in linear complexity for each channel, using additions and subtractions, thanks to the simplicity of the Haar basis. The main idea of our approach is to keep the same top $k$ entries (in magnitude) of the transformed activations maps with respect to *all channels* (dubbed as joint shrinkage) and perform the convolution in the wavelet domain on the *compressed* signals, saving significant computations. We show that the transform and shrinkage operations

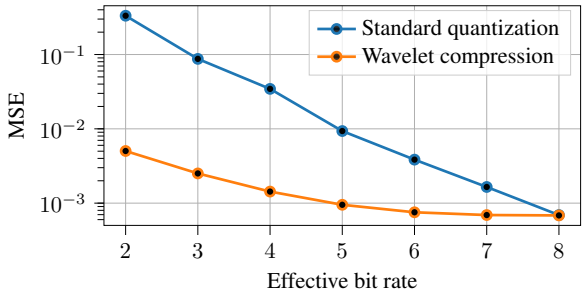



Figure 1: Comparison between standard quantization and our (joint-channel) wavelet compression. The plot shows MSE between the quantized and original activation maps based on a batch of 1000 activation maps from the second hidden layer of MobileNetV3 (small) using the ImageNet data set.

commute with the $1 \times 1$ convolution, the heart of modern CNNs. This procedure is applied along with modest quantization to reduce the computational costs further.

We demonstrate the effectiveness and flexibility of our WCC by applying it in popular network architectures: DeeplabV3plus (Chen et al., 2018) with MobileNetV2 encoder (Sandler et al., 2018) for semantic segmentation, and BTS (Lee et al., 2020) with ResNeXt50 encoder (Xie et al., 2017) for depth prediction. We show that using WCC dramatically improves the results over aggressive quantization for the same compression rates while retaining the baseline network architecture.

## 2 Related work

**Quantized Neural Networks:** Quantized neural networks have been quite popular recently, and are exhibiting impressive progress in the goal for true network compression and efficient CNN deployment. Quantization methods include Zhou et al. (2018); Zhang et al. (2018); Banner et al. (2018b), and in particular Li et al. (2020); Esser et al. (2020); Choi et al. (2019), which show that the clipping parameters—a highly important parameter in quantization schemes—can be learned through optimization. Beyond that, there are more sophisticated methods to improve the mentioned schemes. For example, dynamic quantization schemes utilize different bit allocations at every layer (Dong et al., 2020; Cai & Vasconcelos, 2020; Uhlich et al., 2020). Non-uniform methods can improve the quatization accuracy (Yamamoto, 2021) but require a look-up-table, which reduces hardware efficiency. Quantization methods can also be enhanced by combination with pruning (Tung & Mori, 2018) and knowledge distillation for better training (Kim et al., 2019).

The works above focus on image classification. When considering image-to-image tasks like semantic segmentation, networks tend to be more sensitive to quantization of the activations. In the work of AskariHemmat et al. (2019), targeting segmentation of medical images, the lowest bit rate for the activations is 4 bits, and a significant degredation in the perfomance is evident compared to 6 bits. This is consistent with the work of Tang et al. (2019) that was mentioned earlier, which uses a higher bit rate for the activations than for the weights. The work of Xu et al. (2018) uses weight (only) quantization for medical image segmentation as an attempt to remove noise, and not for computational efficiency. The recent work of Liu et al. (2021) shows both a sophisticated post training quantization scheme and also includes fine-tuned semantic segmentation results. Again a significant degradation is observed when going from 6 to 4 bits. One exception is the work of Heinrich et al. (2018) that uses ternary networks (equivalent to 2 bits here), and segment one medical data set relatively well compared to its full precision baseline.

In this work, we focus on the simplest possible quantization scheme: uniform (across all weights and activations), quantization aware training, with per-layer clipping parameters. That is to ensure hardware compatibility and efficient use of available training data. In principle, one can run our platform regardless of the quantization type and scenario (e.g., non-uniform/dynamic quantization). Also, since we target the compression of the activations, then any method that is focused on the weights (e.g., pruning (Tung & Mori, 2018)) can be combined here as well.

**Wavelet transforms and compression:** Wavelet transforms are widely used in image processing (Porwik & Lisowska, 2004). For example, the JPEG2000 format (Rabbani, 2002) uses the wavelet domain to represent images as highly sparse feature maps. Recently, wavelet transforms have been used to define convolutions and architectures in CNNs for various of imaging tasks. For example, the work of Huang et al. (2017) presents a network architecture for super-resolution, where the wavelet coefficients are predicted and used to reconstruct the high-resolution image. The works of Duan et al. (2017); Williams & Li (2018) use wavelet transforms in place of pooling operators to improve CNNs performance. The former uses dual-tree complex wavelet transform (Kingsbury, 1998), and the latter learns the wavelet basis as part of the network optimization. In these cases, the wavelet transform is not used for compression, but to better preserve information with its low-pass filters.

The work of Wolter et al. (2020) proposes a wavelet-based approach to learn basis functions for the wavelet transform to compresses the weights of linear layers, as opposed to compression of the activation as we apply here. We use the Haar transform (as opposed to a learned one) for its hardware efficiency. Using a different transform (known or learned) is also possible, at the corresponding computational cost. The work of Sun et al. (2021) introduces quantization in the wavelet domain, somewhat similarly to this work. However, the authors suggest to improve the quantization scheme by learning a different clipping parameter per wavelet component, but without the feature shrinkage stage, which is the heart of our approach (we use the same clipping parameter for the whole layer using 8 bits for hardware efficiency). As mentioned before, one can use our WCC together with different types of quantization schemes, in particular including Sun et al. (2021), taking the additional hardware complexity of using different clipping parameter per component into account.

Lastly, Liu et al. (2018) suggest using a modified U-Net architecture for image-to-image translation. There, wavelet transforms are used for down-sampling, and the inverse is used for up-sampling. This work is architecture-specific, and the method can not be easily integrated into other architectures. In contrast, here we propose the WCC layer to easily replace $1 \times 1$ convolutions regardless of the network architecture. Hence our framework is, in principle, also suitable for post-training quantization (Banner et al., 2019), where the data is not available, and the original network is not retrained.

## 3 BACKGROUND

**Quantization-aware training** Quantization schemes can be divided into post- and quantization-aware training schemes. Post training schemes perform model training and model quantization separately, which is most suitable when the training data is not available during the quantization phase (Banner et al., 2019; Nagel et al., 2019; Cai et al., 2020). On the other hand, quantization aware training schemes are used to adapt the model's weights as an additional training phase. Such schemes do require training data, but generally provide better performance. Quantization schemes can also be divided into uniform vs non-uniform methods, where the latter is more accurate but the former is more hardware friendly (Li et al., 2020; Jung et al., 2019). Lastly, quantization schemes can utilize quantization parameters per channel within each layer, or for utilize these parameters only per layer (where all the channels share the same parameters). Similarly to before, per-channel methods are difficult to exploit in hardware, while per-layer methods are less accurate but are more feasible for deployment on edge devices. In this paper we focus on per-layer and uniform quantization aware training for both weights and activations, and aim to improve this with wavelet transforms. Other quantization schemes can be applied within our wavelet compression framework as well.

In quantization-aware training, we set the values of the weights to belong to a small set, so that after training, the application of the network can be carried out in integer arithmetic, i.e., activation maps are quantized as well. Even though that we use discontinuous rounding functions throughout the network, quantization-aware training schemes utilize gradient-based methods to optimize the network's weights (Han et al., 2015; Yin et al., 2019). In a nutshell, when training, we iterate on the floating-point values of the weights. During the forward pass, both the weights and activation maps are quantized, while during the backward pass, the Straight Through Estimator (STE) approach is used (Bengio et al., 2013), where we ignore the rounding function, whose exact derivative is zero.

The specific quantization scheme that we use is based on Li et al. (2020). First, the pointwise quantization operator is defined by:

$$q_b(t) = \frac{\text{round}((2^b-1)\cdot t)}{2^b-1}, \tag{1}$$

where $t \in [-1, 1]$ or $t \in [0, 1]$ for signed or unsigned quantization, respectively[1]. The parameter $b$ is the number of bits that are used to represent $t$. During the forward pass, each number undergoes three steps: scale, clip and round. That is, to get the quantized version of any number we apply:

$$\text{If signed: } x_b = \alpha q_{b-1}(\text{clip}(\frac{x}{\alpha}, -1, 1)); \quad \text{If unsigned: } x_b = Q_b(x) = \alpha q_b(\text{clip}(\frac{x}{\alpha}, 0, 1)). \tag{2}$$

Here, $x, x_b$ are the real-valued and quantized tensors, and $\alpha$ is the clipping parameter. The parameters $\alpha$ in (2) play an important role in the error generated at each quantization and should be chosen carefully. The works of Li et al. (2020); Esser et al. (2020) introduced an effective gradient-based optimization to find the clipping values $\alpha$ for each layer, again using the STE approximation. This enables the quantized network to be trained end-to-end manner with backpropagation. To further improve the optimization, weight normalization is also used before each quantization.

**Haar wavelet transform and compression.** In this section we describe the Haar-wavelet transform in deep learning language, and its usage for compression. Given an image channel $\mathbf{x}$, the one-level Haar transform can be achieved by a strided seperable 2D convolution with the following kernel:

$$\mathbf{W} = \frac{1}{2}\left[\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\right]. \tag{3}$$

The result of the convolution $[\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4] = \text{Conv}(\mathbf{W}, \mathbf{x})$ has four channels, each of them has half the resolution of $\mathbf{x}$. The leftmost kernel in (3) is an averaging kernel, and the three right kernels are edge-detectors. This, together with the fact that images are piece-wise smooth, leads to relatively sparse images $\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$. Hence, if we retain only the few top-magnitude entries in these vectors, we keep most of the information, as most of the entries we drop are zeros. That is the main idea of wavelet compression. We denote the Haar-wavelet transform by $\mathbf{y} = \text{HWT}(\mathbf{x})$, where $\mathbf{y}$ is defined as the concatenation of the vectors $\mathbf{y}_1, ..., \mathbf{y}_4$ into one. Since the kernels in (3) form an orthonormal basis, applying the inverse transform is obtained by applying the transposed convolution of (3):

$$\mathbf{x} = \text{iHWT}(\mathbf{y}) = \text{Conv-transposed}(\mathbf{W}, [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4]).$$

However, unlike $\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$, the averaged image $\mathbf{y}_1$ is not sparse, and is just the down-sampled original $\mathbf{x}$. Hence, in the multilevel wavelet transform, we apply the kernel in (3) on $\mathbf{y}_1$ to generate further down-sampled sparse channels. For example, a 2-level Haar transform can be summarized as

$$\left[\mathbf{y}_1^1, \mathbf{y}_2^1, \mathbf{y}_3^1, \mathbf{y}_4^1\right] = \text{Conv}(\mathbf{W}, \mathbf{x}); \quad \left[\mathbf{y}_1^2, \mathbf{y}_2^2, \mathbf{y}_3^2, \mathbf{y}_4^2\right] = \text{Conv}(\mathbf{W}, \mathbf{y}_1^1), \tag{4}$$

and the resulting transformed image with 2-levels can be written as the concatenated vector

$$\mathbf{y} = \left[\mathbf{y}_1^2, \mathbf{y}_2^2, \mathbf{y}_3^2, \mathbf{y}_4^2, \mathbf{y}_2^1, \mathbf{y}_3^1, \mathbf{y}_4^1\right] = \text{HWT}(\mathbf{x}). \tag{5}$$

In this work we use 3 levels in all the experiments. To apply compression we define the operator $\mathbf{T}$ that extracts the top $k$ entries in magnitude from the vector $\mathbf{y}$: $\mathbf{y}^{compressed} = \mathbf{T} \cdot \text{HWT}(\mathbf{x})$. To de-compress this vector we first zero-fill $\mathbf{y}^{compressed}$ (multiply with $\mathbf{T}^\top$) and apply the inverse Haar transform, which is the transposed convolutions in the opposite order.

## 4 Wavelet Compressed Convolution

In this work, we aim to reduce the memory bandwidth and computational cost associated with convolutions performed on intermediate activation maps. To this end, we apply the Haar-wavelet transform to compress the activation maps, in addition to light quantization of 8 bits. Our method is most efficient for scenarios with high-resolution feature maps (i.e., large images, whether in 2D or 3D), where the wavelet compression is most effective. Such cases are mostly encountered in image-to-image tasks like semantic segmentation, depth prediction, image denoising, in-painting, super-resolution, and more. Typically, in such scenarios, the size and memory bandwidth used for the weights are relatively small compared to those used for the features (e.g., point cloud segmentation). In addition, since we have to predict each pixel, typical architectures do not aggressively down-sample the images, as opposed to multi-class image classification, where activation maps become small and the number of channels grows towards the end of the network.

[1] In most standard CNNs, the ReLU activation is used, hence the activation feature maps are non-negative, and can be quantized using an unsigned scheme. If a different activation function that is not non-negative is used, or, as in our case, the wavelet coefficients are quantized, signed quantization should be used instead.

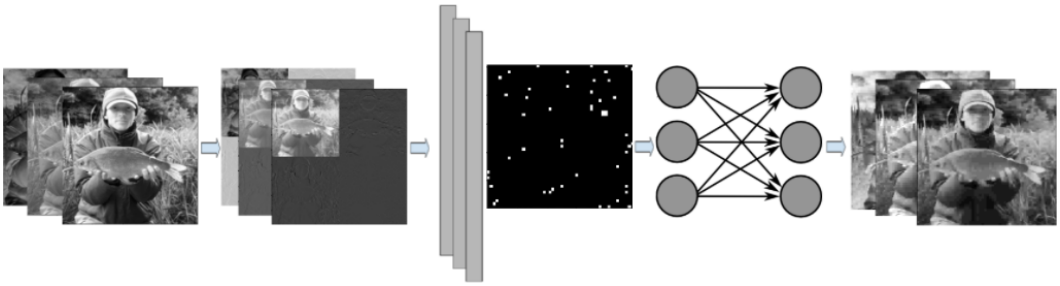


Figure 2: The workflow of WCC. From Left to right: the three input channels, their Haar transform, the joint shrinkage into 1D vectors and a single bit-map (or a list of indices), the application of the $1 \times 1$ convolution on the 1D compressed vectors, and lastly on the right: the inverse transform back to the spatial domain. In this example, the shrinkage ratio is set to $0.1$.

**Convolution in the wavelet domain:** Since we focus on computational efficiency, in this work, we use the Haar transform, as it is the simplest and most computationally efficient wavelet variant. Indeed, the Haar transform involves binary pooling-like operations, which include only additions, subtractions, and a bit-wise shift. We focus on the compression of fully-coupled $1 \times 1$ convolutions, as these are the workhorse of lightweight and efficient architectures like MobileNet (Sandler et al., 2018), ShuffleNet (Ma et al., 2018), EfficientNet (Tan & Le, 2019), and ResNeXt (Xie et al., 2017). All these modern architectures rely on $1 \times 1$ convolutions in addition to grouped or depthwise spatial convolutions (i.e., with $3 \times 3$ or larger kernels), which comprise a small part of the computational effort in the network—the $1 \times 1$ operations dominate the inference cost. The main idea of our method is to transform and compress the input in the wavelet domain prior to the $1 \times 1$ convolution and apply the convolution in the wavelet domain on a fraction of the original input size. Since the wavelet compression is applied separately on each channel, it commutes with the $1 \times 1$ convolution. Hence, applying the convolution in the wavelet domain is equivalent to applying it in the spatial domain.

**Joint hard shrinkage:** Prior to the $1 \times 1$ convolution, the spatial wavelet transform is applied, and we get sparse feature maps. Since the different channels result from the same input image propagated through the network, they typically include patterns at similar locations, and hence the sparsity pattern of their wavelet-domain representation is relatively similar. This idea of redundancy in the channel space is exploited in the lightweight networks GhostNet (Han et al., 2020) and MGIC (Eliasof et al., 2020), where part of the channels are used in both cases to represent the others via a spatially local convolution. Therefore, we perform a joint shrinkage operation between all the channels, in which we zero and remove the entries with the smallest feature norms across channels, resulting in a compressed representation of the activation maps[2]. The locations of the non-zeros in the original image are kept in a *single* index list or a bit-map for all the channels in the layer, as they are needed for the inverse transform back to the spatial domain. Lastly, we also apply light 8-bit quantization to the transformed images to further improve the compression rate. The wights and wavelet-domain activations are quantized using the symmetric scheme as described in section 3.

More precisely, the advantage of the wavelet transforms is their ability to compress images. Denote the Haar transform matrix as $\mathbf{H}$, i.e., $\mathbf{H}\mathbf{x} = \text{HWT}(\mathbf{x})$. Then, for most natural images we have that

$$\mathbf{x} \approx \mathbf{H}^\top \mathbf{T}^\top \mathbf{T}\mathbf{H}\mathbf{x} \tag{6}$$

where $\mathbf{T}$ is the shrinkage operator described above, and because of its orthogonality, $\mathbf{H}^\top$ is the inverse transform IHWT. Our WCC layer is defined by:

$$\text{WCC}(\mathbf{K}_{1\times1}, \mathbf{x}) = \mathbf{H}^\top \mathbf{T}^\top \mathbf{K}_{1\times1} \mathbf{T}\mathbf{H}\mathbf{x}, \tag{7}$$

where $\mathbf{K}_{1\times1}$ is the learned convolution matrix. The workflow is illustrated in Figure 2. Note that the convolution operates on the compressed domain, hence, if $\mathbf{T}$ can significantly reduce the dimensions of the channels, this leads to major savings in computations.

[2]Regardless of the joint sparsity assumption of the channels, some compression approaches suggest taking the left-upmost part of the wavelet transform for any image, so in principle, the joint sparsity approach may work relatively well for a general set of images as well.

Our method aims at compression only. Hence, we show that a $1 \times 1$ convolution kernel can be applied both in the spatial and in the compressed wavelet domain. By its definition, we can write a $1 \times 1$ convolution as a summation over channels. That is:

$$\mathbf{y} = \mathbf{K}_{1\times 1}\mathbf{x} \Rightarrow \mathbf{y}_i = \sum_j k_{ij}\mathbf{x}_j, \tag{8}$$

where $k_{ij} \in \mathbb{R}$ are the weights of the convolution tensor. Now, suppose we wish to compress the result $\mathbf{y}$, through (6). Because $\mathbf{T}$ and $\mathbf{H}$ are separable and spatial, we get by simple linear arithmetic:

$$\mathbf{y}_i \approx \mathbf{H}^\top\mathbf{T}^\top\mathbf{T}\mathbf{H}(\mathbf{K}_{1\times 1}\mathbf{x})_i = \mathbf{H}^\top\mathbf{T}^\top\sum_j k_{ij}\mathbf{T}\mathbf{H}\mathbf{x}_j \Rightarrow \mathbf{y} \approx \mathbf{H}^\top\mathbf{T}^\top\mathbf{K}_{1\times 1}\mathbf{T}\mathbf{H}\mathbf{x}. \tag{9}$$

Hence, the wavelet compression operator, which is known to be highly efficient for natural images, commutes with the $1 \times 1$ convolution, so the latter can be applied on the compressed signal and get the same result as compressing the result directly. This introduces an opportunity to save computations on the one hand and use more accurate compression on the other.

The description above is suited for $1 \times 1$ convolutions only, while many CNN architectures involve spatial convolutions with larger kernels, strides, etc. The main concept here is that fully coupled convolutions that mix all channels are expensive, inefficient, and redundant when used with large spatial kernels (Ephrath et al., 2020). The spatial mixing can be obtained using separable convolutions at less cost without losing efficiency (Chen et al., 2018). Since we aim at saving computations, separating the kernels in the architecture is recommended even before discussing any type of lossy compression. Furthermore, separable convolutions (and the Haar transform) can be applied separately in chunks of channels or together as part of the $1 \times 1$ convolution using specialized implementation. Hence the memory bandwidth can be reduced. The $1 \times 1$ convolution, on the other hand, involves all channels at once; hence it is harder to reduce the bandwidth in this case.

### 4.1 Computational Costs in Bit Operations (BOPs)

To evaluate the computational cost involved in WCC we use the measure of Bit-Operations (BOPs) (Wang et al., 2020; Louizos et al., 2018). First, the number of Multiply-And-Accumulate (MAC) operations in a convolutional layer is given by

$$\text{MAC(conv)} = C_{\text{in}} \cdot C_{\text{out}} \cdot N_W \cdot N_H \cdot K_W \cdot K_H \cdot \frac{1}{S_W \cdot S_H}, \tag{10}$$

where $C_{\text{in}}$ and $C_{\text{out}}$ are the number of input and output channels, $(N_W, N_H)$ is the size of the input, $(K_W, K_H)$ is the size of the kernel, and $(S_W, S_H)$ is the stride value. The BOPs count is then

$$\text{BOPs(conv)} = \text{MAC(conv)} \cdot b_w \cdot b_a, \tag{11}$$

where $b_w$ and $b_a$ denote the number of bits used for weight and activations.

As described in section 3, the Haar transform is separable between the input channels, and can be viewed as four $2 \times 2$ convolutions with stride $(2, 2)$ and binary weights. Hence, the one-level transform requires $4 \cdot C_{\text{in}} \cdot W \cdot H \cdot b_a$ BOPs. The transform can be used with more levels of compression explained in section 3, on down-scaled inputs, resulting in a total of

$$\sum_{l=1}^{L} 4 \cdot C_{\text{in}} \cdot N_W \cdot N_H \cdot \frac{1}{4^{l-1}} \cdot b_a \tag{12}$$

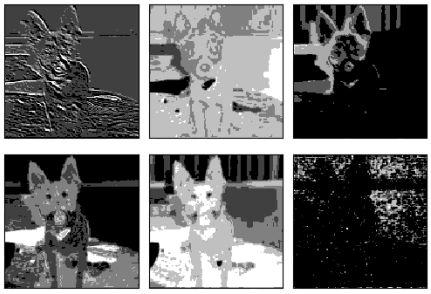
BOPs, where $L$ is the level of compression. Similarly, the inverse-transform result in the same calculation, only with $C_{\text{out}}$ in place of $C_{\text{in}}$. To demonstrate the relatively small cost of the compression, consider a $1 \times 1$ convolution with $C_{\text{in}} = 160$, $C_{\text{out}} = 960$, input size of $(34, 34)$, and quantization $b_w = b_a = 8$ (which is part of a network used later in section 5). This layer costs $11,364M$ BOPs. Using a 3 levels wavelet transform and its inverse for this layer results in $54M$ BOPs, a negligible cost which allows for better compression, as we demonstrate next.

## 5 Results

In this section, we evaluate our WCC layer's performance in image-to-image tasks—semantic segmentation and depth estimation—where the feature maps are relatively large and suitable for wavelet compression. We compare our method to quantization aware training based on the work of Li et al. (2020), which we also use in our method for 8-bit quantization of the shrunk signals. Our code is implemented in PyTorch (Paszke et al., 2017) based on Torchvision implementations, including the same data augmentation. We ran our experiments on NVIDIA 24GB RTX 3090 GPU. We detail our experimental setup for each section separately.

(a) Original input image from ImageNet

(b) Feature maps from ResNet50 layers.



(c) 2-bit quantized layers (unsigned).

(d) 2-bit equivalent wavelet compressed.

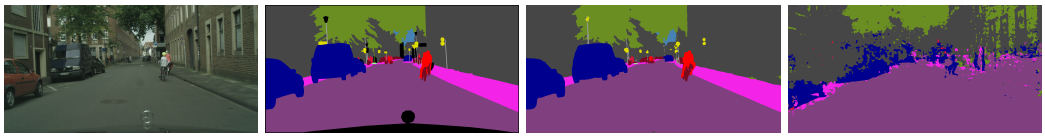
Figure 3: Feature maps from layers 2 and 3 (top and bottom triplets, respectively) of a pretrained ResNet50 on ImageNet. The maps are compressed with uniform quantization (2-bit) and wavelet compression (25% shrinkage + 8-bit quantization, equivalent to 2-bit quantization in terms of size). It is clear that wavelet compression loses much less information than aggressive quantization.

### 5.1 Qualitative Compression Assessment

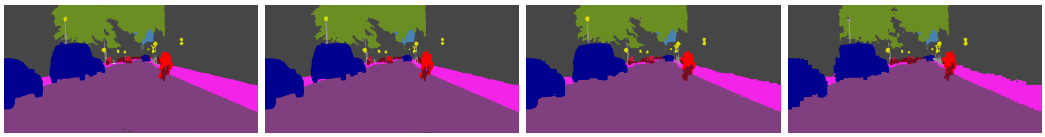
Before describing experiments with common datasets and tasks, we first qualitatively demonstrate the advantage of our approach. First, we compare the MSE of standard quantization vs our approach on a feature map from a pretrained MobileNetV3 applied on a batch of 1000 images from the ImageNet dataset (Krizhevsky et al., 2017). We consider an effective bit rate range of $[2, 8]$. The quantization of the shrunk wavelet coefficients is 8 bit, and multiplying it with the compression ratio yields the effective bit rate. E.g., 8 bit and 25% shrinkage is effectively 2 bits. This also agrees with the BOPs measure in Eq. (11). Figure 1 shows the MSE comparison per effective bit rate. One can see that the information loss based on the MSE is more significant in standard quantization. We note that the quantization parameters in this experiment were chosen by an exhaustive search for minimizing the MSE for every dot in the plot. Furthermore, in Figure 3 we can see a visualization comparison of a typical activation map compressed by a standard quantization and our method. It is obvious that the perceptive degradation in the image approximation is more significant in the standard quantization. This is not surprising since wavelet image compression is more advanced than uniform quantization.

### 5.2 Semantic Segmentation

Semantic segmentation is the task of assigning a label to every pixel in the input image. CNNs have shown significant improvements over traditional methods but at a high computational cost. Our model of choice here is the popular DeeplabV3plus (Chen et al., 2018) with MobileNetV2 backbone (Sandler et al., 2018). We evaluated the proposed method on the Cityscapes and Pascal VOC. The Cityscapes dataset (Cordts et al., 2016) contains images of urban street scenes. The images are of size 1024x2048 with pixel-level annotation of 19 classes. During training, we used a random crop of size 768x768 and no crop for the validation set. The Pascal VOC (Everingham et al., 2015) dataset contains images of size 513x513 with pixel-level annotation of 20 foreground object classes and a background class. We augmented the dataset similarly to Chen et al. (2018).



(a) Input image (b) Ground truth (c) Quantization 8/8 (d) Quantization 8/4



(e) WCC 8/8 100% (f) WCC 8/8 50% (g) WCC 8/8 25% (h) WCC 8/8 12.5%

Figure 4: Cityscapes segmentation results. All the networks use weight quantization of 8-bits. (a) input image. (b) ground truth. (c), (d) normal quantization with 8- and 4-bits activations respectively. (e)-(h) WCC with 8-bits activations and shrinkage rate of 100%, 50%, 25% and 12.5% respectively.

| Precision (W / A) | Wavelet shrikage | Cityscapes BOPs (B) | Cityscapes mIoU | Pascal VOC BOPs (B) | Pascal VOC mIoU |
|---|---|---|---|---|---|
| Full precision | None | 36,377 | 0.717 | 4,534 | 0.715 |
| 8bit / 8bit | None | 2,273 | 0.701 | 283 | 0.712 |
| 8bit / 6bit | None | 1,705 | 0.683 | 212 | 0.678 |
| 8bit / 4bit | None | 1,136 | 0.173 | 141 | 0.095 |
| 8bit / 8bit | 100% | 2,292 | 0.697 | 285 | 0.695 |
| 8bit / 8bit | 50% | 1,213 | 0.681 | 150 | 0.675 |
| 8bit / 8bit | 25% | 673 | 0.620 | 82 | 0.611 |
| 8bit / 8bit | 12.5% | 403 | 0.552 | 48 | 0.519 |
| 4bit / 8bit | None | 1,136 | 0.682 | 141 | 0.675 |
| 4bit / 6bit | None | 852 | 0.669 | 106 | 0.657 |
| 4bit / 4bit | None | 568 | 0.190 | 70 | 0.099 |
| 4bit / 8bit | 100% | 1,156 | 0.672 | 144 | 0.678 |
| 4bit / 8bit | 50% | 616 | 0.667 | 76 | 0.661 |
| 4bit / 8bit | 25% | 346 | 0.621 | 42 | 0.583 |
| 4bit / 8bit | 12.5% | 211 | 0.549 | 24 | 0.515 |

Table 1: Validation results for semantic segmentation task using DeepLabV3 with MobileNetV2 as the backbone. Segmentation performance is measured by mean intersection over union (mIoU).

We used two configurations for the weights—4 and 8 bits—and for each of them we used different compression rates for the activations, both for the standard quantization and our WCC layer. Like before, our WCC layer with a shrinkage rate of 50% has nearly equal BOPs to reducing the activation bits from 8 to 4. Likewise, a shrinkage rate of 25% nearly equals to 2 bits.

The training scheme used is similar to Li et al. (2020). We first trained a network in full precision and then gradually reduce the bit rates (this was used for the WCC as well). For all datasets and model optimization, we used SGD with momentum 0.9, weight decay $10^{-4}$, learning rate decay 0.9, and set the batch size to 16. For Cityscapes, we trained the full precision model for 160 epochs with a base learning rate of 0.1, and each of the retrains was trained for 50 epochs with a base learning rate of 0.01. For Pascal VOC, we trained the full precision model for 50 epochs with a base learning rate of 0.01, and each of the retrains was trained for 25 epochs with a base learning rate of 0.002.

Table 1 shows the performance and BOPs of each model, from which we can see that our model achieves superior performances at high compression configurations, outperforming the 4-bit activation even with compression rates equivalent to 2- and 1-bits. Figure 4 shows a visual example.

| Precision (W / A) | Wavelet shrinkage | BOPs (B) | AbsRel | RMSE |
|---|---|---|---|---|
| Full precision | None | 156,882 | 0.061 | 2.774 |
| 8bit / 8bit | None | 9,805 | 0.068 | 2.889 |
| 8bit / 8bit | 50% | 4,910 | 0.071 | 3.063 |
| 8bit / 4bit | None | 4,902 | 0.070 | 3.065 |
| 8bit / 8bit | 25%-50% | 4,549 | 0.075 | 3.228 |
| 8bit / 2bit-4bit | None | 4,541 | 0.106 | 4.082 |
| 8bit / 8 bit | 25% | 4,297 | 0.08 | 3.352 |

Table 2: Validation results on BTS, using ResNeXt50 as backbone encoder, measured by absolute relative difference (AbsRel) and root mean squared error (RMSE), in both lower is better.

Comparison to Liu et al. (2021): This work deals with zero-shot quantization but also shows results on quantization-aware training (fine-tuned) for the Cityscapes dataset, albeit using $256 \times 256$ input images. In this paper, the 8W8A fine-tuned result has an mIoU of 61.25 compared to about 70 here, in both the standard and wavelet cases. Our standard 4W6A setting achieves 0.67 vs. 0.6 for 6W6A by Liu et al. (2021). Furthermore, our results with the wavelet compression using 4W with all percentages (up to 12.5%) outperform or are on par with the 4W4A result of 0.56 by Liu et al. (2021).

### 5.3 Monocular Depth Estimation

In this section, we apply our compression technique on BTS (Lee et al., 2020), a network architecture for monocular depth estimation—a task of estimating scene depth using a single image. We evaluated the results on the KITTI dataset (Geiger et al., 2013), containing images of autonomous driving scenarios, each of size $\sim$1241x376 pixels. The dataset splits are as in BTS, using the strategy of Eigen et al. (2014). As in BTS, we use a maximum value of 80 meters for prediction when testing, and the performance evaluation is based on the cropping scheme of Garg et al. (2016). When applying compression to BTS, we did not modify the initial convolution in the backbone network, nor did we modify the final convolutions and depth prediction layer in the decoder. Applying any compression technique to these layers resulted in a significant degradation in the results.

In our experiments, we focus on a ResNeXt50 backbone. The compressed models are fine-tuned from a 8bit/8bit quantized network, trained from scratch. In local planar guidance layers, we use 8bit/4bit quantization instead of our wavelet scheme. When training, we use the same configuration presented in the BTS repository. We use the AdamW optimizer (Loshchilov & Hutter, 2019), a learning rate of 0.00014, a weight decay of 0.001, a batch size of 4 images, and train for 50 epochs.

Table 2 shows a comparison between the standard quantization and WCC for BTS. Using our WCC layer with 50% compression resulted in comparable scores to the alternative method of 8bit/4bit quantization (with similar BOPs). Moreover, when applying a compression factor of 25% (50% on low spatial-sized tensors), we achieve superior results to the quantized alternative. This comparison is presented qualitatively in Figure 5 in the appendix. Finally, in the full 25% compression experiment, we see that even when aggressively compressing the low spatial-sized tensors, we achieve better results than using low-precision standard quantization.

## 6 Conclusions

In this work, we presented a new approach for feature map compression, aiming to reduce the memory bandwidth and computational cost of image-to-image CNNs, where typically the resolution is high. Our approach is based on the classical Haar-wavelet image compression, which has been used for years in standard devices and simple hardware, and in real-time. We save the computational cost by applying $1 \times 1$ convolutions on the shrunk wavelet domain. We show that this approach surpasses aggressive quantization using the same bit-rate at minimal additional costs of the transform.

## REPRODUCIBILITY STATEMENT

The code for running and reproducing the experiments reported in this paper is attached as supplementary material.

## REFERENCES

MohammadHossein AskariHemmat, Sina Honari, Lucas Rouhier, Christian S. Perone, Julien Cohen-Adad, Yvon Savaria, and Jean-Pierre David. U-net fixed point quantization for medical image segmentation. In *Medical Imaging and Computer Assisted Intervention (MICCAI), Hardware Aware Learning Workshop (HAL-MICCAI)*. Springer, 2019.

R. Banner, Y. Nahshan, and D. Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *The Conference on Neural Information Processing Systems (NeurIPS)*, pp. 7948–7956, 2019.

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 5145–5153, 2018a.

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Aciq: Analytical clipping for integer quantization of neural networks. *ArXiv*, abs/1810.05723, 2018b.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Zhaowei Cai and Nuno Vasconcelos. Rethinking differentiable search for mixed-precision neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2349–2358, 2020.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. Accurate and efficient 2-bit quantized neural networks. In *MLSys*, 2019.

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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *The Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Yiping Duan, Fang Liu, Licheng Jiao, Peng Zhao, and Lu Zhang. SAR image segmentation based on convolutional-wavelet neural network and markov random field. *Pattern Recognition*, 64:255–267, 2017.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network, 2014.

Moshe Eliasof, Jonathan Ephrath, Lars Ruthotto, and Eran Treister. Multigrid-in-channels neural network architectures. *arXiv preprint arXiv:2011.09128*, 2020.

Jonathan Ephrath, Moshe Eliasof, Lars Ruthotto, Eldad Haber, and Eran Treister. Leanconvnets: Low-cost yet effective convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):894–904, 2020.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *The International Conference on Learning Representations (ICLR)*, 2020.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue, 2016.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493*, 2016.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1580–1589, 2020.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

Mattias P Heinrich, Max Blendowski, and Ozan Oktay. Ternarynet: faster deep model inference without gpus for medical 3d segmentation using sparse and binary convolutions. *International journal of computer assisted radiology and surgery*, 13(9):1311–1320, 2018.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1689–1697, 2017.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187:1–187:30, 2017.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. Qkd: Quantization-aware knowledge distillation. *arXiv preprint arXiv:1911.12491*, 2019.

Nick Kingsbury. The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement. In *9th European Signal Processing Conference (EUSIPCO 1998)*, pp. 1–4. IEEE, 1998.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation, 2020.

Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5813–5823, 2017.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *The International Conference on Learning Representations (ICLR)*, 2020.

Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 773–782, 2018.

Yuang Liu, Wei Zhang, and Jun Wang. Zero-shot adversarial quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1512–1521, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. *arXiv preprint arXiv:1810.01875*, 2018.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *ICCV*, pp. 1325–1334, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*, 2017.

Piotr Porwik and Agnieszka Lisowska. The haar-wavelet transform in digital image processing: its status and achievements. *Machine graphics and vision*, 13(1/2):79–98, 2004.

Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Qigong Sun, Yan Ren, Licheng Jiao, Xiufang Li, Fanhua Shang, and Fang Liu. MWQ: Multiscale wavelet quantized neural networks. *arXiv preprint arXiv:2103.05363*, 2021.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Zhiqiang Tang, Xi Peng, Kang Li, and Dimitris N Metaxas. Towards efficient u-nets: A coupled and quantized approach. *IEEE transactions on pattern analysis and machine intelligence*, 42(8): 2038–2050, 2019.

Frederick Tung and Greg Mori. Deep neural network compression by in-parallel pruning-quantization. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):568–579, 2018.

S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J.A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura. Mixed precision DNNs: All you need is a good parametrization. *The International Conference on Learning Representations (ICLR)*, 2020.

Ying Wang, Yadong Lu, and Tijmen Blankevoort. Differentiable joint pruning and quantization for hardware efficiency. In *European Conference on Computer Vision*, pp. 259–277. Springer, 2020.

Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *International Conference on Learning Representations*, 2018.

Moritz Wolter, Shaohui Lin, and Angela Yao. Neural network compression via learnable wavelet transforms. In *International Conference on Artificial Neural Networks*, pp. 39–51. Springer, 2020.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Xiaowei Xu, Qing Lu, Lin Yang, Sharon Hu, Danny Chen, Yu Hu, and Yiyu Shi. Quantization of fully convolutional networks for accurate biomedical image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8300–8308, 2018.

Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5029–5038, 2021.

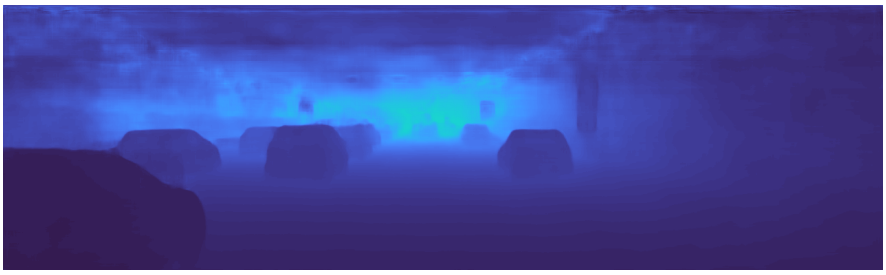
Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *International Conference on Learning Representations, (ICLR)*, 2019.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

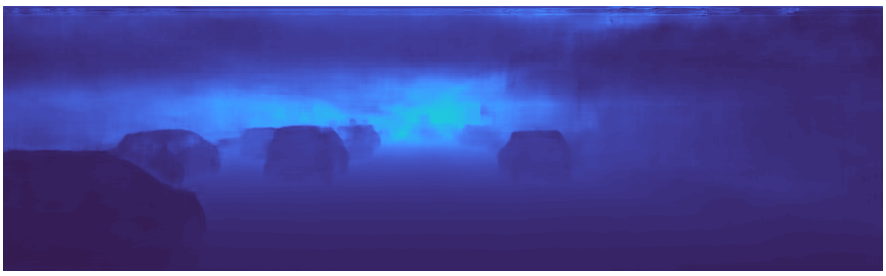
Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *The International Conference on Machine Learning (ICML)*, 2018.

# A APPENDIX

(a) Input Image



(b) Wavelet Compression 8/8 25% (50% in small tensors)



(c) Quantization 8/2 (8/4 in small tensors)

Figure 5: Kitti depth estimation prediction examples. We compare two compressed networks with the same BOP magnitude, the first uses our wavelet compression method and the second uses low-bit quantization.