ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Quantized Sparse Models for Video Semantic Segmentation

Master's Thesis

Ziqing Chang
Department of Information Technology and Electrical Engineering

Advisors: Dr. Yawei Li, Dr. Guolei Sun
Supervisor: Prof. Dr. Luca Benini

December 12, 2024

Abstract

Video Semantic Segmentation (VSS) is essential for applications like autonomous driving, augmented reality, and robotics, where understanding dynamic scenes in real-time is crucial. However, deploying VSS models on resource-constrained devices poses significant challenges due to high computational complexity and memory demands. While Extremely Low Bit Quantization (ELBQ) has shown promise in compressing large models, its application to small models suitable for edge devices often results in substantial performance degradation, as smaller models have less capacity to absorb quantization noise. Additionally, integrating quantization with N:M structured sparsity in a way that preserves model accuracy remains an open challenge. In this paper, we address these limitations by extending ELBQ techniques to small VSS models and closely coupling quantization with N:M structured sparsity, enhanced by a channel permutation strategy to preserve important weights during pruning. We introduce *QLinear* and *QConv2d* as drop-in replacements for standard linear and convolutional layers, enabling quantization-aware training from scratch with minimal code modifications. Our method significantly reduces computational complexity and memory usage while maintaining high segmentation accuracy. Experimental results on the VSPW dataset demonstrate that our approach enables efficient deployment of VSS models on resource-constrained devices without compromising performance, outperforming existing quantization and pruning methods applied to small VSS models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Video Semantic Segmentation (VSS) is a critical task in computer vision that involves assigning semantic labels to every pixel in a video sequence. It plays an important role in applications such as autonomous driving, augmented reality, surveillance, and robotics, where understanding dynamic scenes is essential for decision-making. However, deploying VSS models in real-world scenarios presents significant challenges due to their high computational complexity and substantial memory requirements. These models need to process high-resolution video frames in real time, which is demanding, especially on edge devices with limited computational resources and power constraints.

To address these challenges, quantization has emerged as a promising technique to reduce the computational burden and memory footprint of deep neural networks by representing weights and activations with lower precision [14, 12, 6, 24, 31, 58, 8, 20, 37, 36, 47, 54, 23, 7]. While quantization has been extensively studied in Large Language Models (LLMs) and image-based models, its application to VSS remains relatively unexplored. Moreover, recent advances in Extremely Low Bit Quantization (ELBQ) have demonstrated that models can be quantized to as low as 1-bit representations [55, 38, 63], offering significant efficiency gains. However, these methods are primarily designed for large models, benefiting from a scaling law where performance improves with an increasing number of parameters. Consequently, when applied to smaller models, such as those suitable for edge devices, ELBQ techniques like BitNet [55, 38] often experience substantial performance degradation. *Addressing the performance gap of ELBQ in small models is crucial for enabling efficient deployment in resource-constrained environments.*

Besides quantization, pruning techniques, particularly N:M structured sparsity [69], have been employed to reduce model size and accelerate inference. N:M sparsity imposes a fine-grained sparsity pattern that is compatible with modern hardware accelerators and allows for significant computational savings. Nevertheless, integrating quantization with N:M sparsity in a way that preserves model accuracy remains a complex challenge. *The interplay between reduced precision and structured sparsity requires careful balancing to ensure that the combined compression techniques do not excessively compromise the model's performance.*

To tackle these issues, we explore the synergy between extremely low-bit quantization and N:M structured sparsity, specifically focusing on small VSS models. Our goal is to develop an approach that significantly reduces the computational complexity and memory usage of VSS models while maintaining high segmentation accuracy, enabling their deployment on resource-constrained devices. By addressing the limitations of ELBQ in small models and effectively using the joint optimization combining quantization with structured pruning during training, we aim to bridge the performance gap and unlock the potential for efficient video semantic segmentation in practical applications.

We propose a novel method that integrates extremely low-bit quantization with N:M structured sparsity, enhanced by a channel permutation strategy to preserve important weights during pruning. Additionally, we introduce *QLinear* and *QConv2d* as drop-in replacements for the standard *nn.Linear* and *nn.Conv2d*

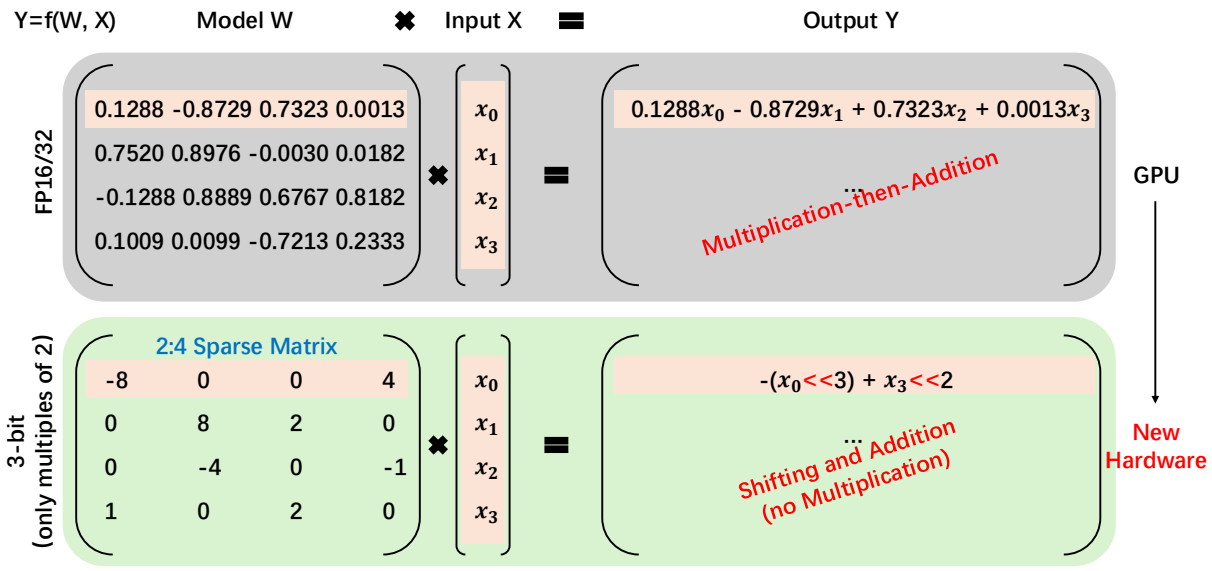

Figure 1.1: Our Method vs. Full Precision model.

layers. These quantization-aware layers enable the training of quantized weights from scratch and facilitate seamless integration into existing architectures with minimal modification. Furthermore, we employ knowledge distillation from a high-precision teacher model to mitigate the accuracy loss typically associated with aggressive quantization and pruning.

Our contributions can be summarized as follows:

- **Extending ELBQ to Small VSS Models:** We investigate the application of extremely low-bit quantization to small VSS models, addressing the performance degradation typically observed when quantizing smaller networks. Through quantization-aware training and carefully designed quantization schemes, we achieve significant efficiency gains with minimal loss in accuracy.
- **Integrating Quantization with N:M Structured Sparsity and Channel Permutation:** We introduce a method that closely couples quantization with N:M structured sparsity, augmented by a channel permutation strategy. This joint optimization ensures that important weights are preserved during pruning and that the structured sparsity pattern aligns well with hardware acceleration capabilities, effectively leveraging both quantization and pruning to maximize computational efficiency without sacrificing model performance.
- **Introducing *QLinear* and *QConv2d* as Drop-in Replacements:** We present *QLinear* and *QConv2d*, which serve as drop-in replacements for the standard *nn.Linear* and *nn.Conv2d* layers. These quantization-aware layers enable the training of quantized weights from scratch, allowing for easy integration into existing architectures with minimal code modifications. This plug-and-play design simplifies the adoption of our quantization strategy, making it accessible for a wide range of VSS models.

Chapter 2

Related Work

2.1 Quantization

Quantization reduces the computational complexity and memory footprint of deep neural networks by representing weights and activations with lower precision. Early works like Han *et al.* [14] combined quantization with pruning and Huffman coding to compress models significantly without substantial accuracy loss. Nowadays, two primary approaches exist: Post-Training Quantization (PTQ) [12, 6, 24, 31, 58, 8, 20, 37] and Quantization-Aware Training (QAT) [36, 47, 54, 23, 7].

PTQ applies quantization to a pre-trained model without additional retraining. This method is efficient and straightforward, making it suitable for rapid deployment. SmoothQuant [58] addresses activation outliers by transferring the quantization challenges from activations to weights. AWQ [31] propose to identify the optimal per-channel scaling that protects the sailent weights by monitoring activations. PTQ4SAM [37] quantized the Segment Anything Model (SAM) [25]. However, PTQ may lead to significant degradation in performance for models sensitive to quantization errors or when using extremely low bit-widths.

QAT integrates quantization effects into the training process, allowing the model to adapt to the quantization noise. QLoRA [7] is an efficient fine-tuning method that uses 4-bit quantization and memory optimizations to train large language models on a single GPU. LLM-QAT [36] trains a limited number of parameters when quantizing the model and utilizes knowledge distillation with data produced by the original model. I-ViT [30] is an integer-only quantization method for Vision Transformers (ViTs) [10] that enables the entire inference process to run using integer arithmetic and bit-shifting. By simulating low-precision arithmetic during both forward and backward passes, QAT helps in maintaining accuracy even at lower bit-widths.

Recently, Extremely Low Bit Quantization (ELBQ) [55, 38, 63] pushes the boundaries by reducing the precision to 4-bit, 2-bit, or even binary representations. BitNet [55] introduces BitLinear as a direct substitute for the nn.Linear layer, enabling the training of 1-bit weights from scratch. BitNet b1.58 [38] improves the performance of BitNet by making the weights ternary {-1, 0, 1} instead of binary {-1, 1}. While these methods offer significant reductions in model size and computational cost, they often suffer from accuracy degradation, especially in smaller models suitable for edge devices.

While quantization methods has been extensively studied in image-based models, its application to video semantic segmentation remains less explored, representing an opportunity for significant efficiency gains.

2.2 Pruning

Pruning techniques aim to eliminate redundant weights from neural networks to reduce their size and computational demands. There are mainly two types of pruning approaches: Post-training Pruning (PTP) [16, 28, 11, 53], which identifies and removes redundant parameters after the model has been trained, aiming

to simplify the model while maintaining its performance; and Pruning-Aware Training (PAT) [15, 32, 41], which incorporates pruning into the training process. In this approach, the network is trained with a pruning objective, allowing it to adapt to the sparsity pattern and maintain performance despite parameter reduction. For instance, Han *et al.* [15] removed connections with small weights, leading to sparse networks. While effective in reducing the number of parameters, unstructured sparsity is challenging to exploit for computational acceleration on standard hardware due to irregular memory access patterns.

N:M Sparsity introduces a fine-grained structured sparsity pattern where, in every group of M elements, only N elements are non-zero. This approach balances the benefits of unstructured and structured sparsity, allowing efficient acceleration on specialized hardware. For instance, NVIDIA's Ampere architecture supports $2{:}4$ sparsity, where in every block of four weights, two are zero. This specific sparsity pattern enables optimized matrix multiplication operations, leading to significant speed-ups during inference. Zhou *et al.* [69] demonstrated that training models with N:M sparsity constraints can maintain accuracy while benefiting from hardware acceleration. Additionally, Pool and Yu [45] proposed a method that integrates channel permutations with N:M sparsity. Their approach rearranges the channels before applying N:M sparsity to better preserve weight magnitude, which are further improved by Zhang *et al.* [67] by proposing a new pruning metric RIA.

Nonetheless, applying pruning techniques to video semantic segmentation remains relatively unexplored, presenting valuable opportunities for future research.

2.3 Distillation

Knowledge distillation [17] involves training a smaller "student" model to replicate the behavior of a larger "teacher" model. This technique has proven effective in transferring knowledge while reducing model size.

Structured distillation methods [34] focus on transferring not only the output logits but also intermediate representations, which is particularly beneficial for dense prediction tasks like segmentation. In video understanding, some studies [40, 57, 60] have explored distillation from models trained on large-scale video datasets to smaller models or different modalities.

Thus, by applying distillation to video semantic segmentation tasks can help training smaller models to capture more intricate temporal and spatial patterns learned by larger models.

2.4 Video Semantic Segmentation

Video Semantic Segmentation (VSS) has grown its significance in computer vision research, primarily due to the increasingly large demand of real world applications in the new AI era, such as Augmented Reality (AR) and Virtual Reality (VR), Medical Imaging and Diagnostics, Autonomous Driving, Surveillance and Security, Content Creation and Video Editing. Prior to VSS, many great works of Image Semantic Segmentation (ISS) [48, 66, 4, 18, 2, 64, 46, 52, 1, 9] lay the foundation for VSS and can also be used to segment single frame of consecutive video frames. But by applying image segmentation models to videos frame-by-frame, ignored temporal information can result in flickering artifacts and degrade the segmentation performance. To leverage temporal dependencies, for example, CFFM [50] studies the static and motional contexts' relations in videos, whereas MRCFA [51] studies the relations among cross-frame affinities. Many great works of VSS [26, 49, 33, 22, 13, 42, 59, 62, 21, 29, 70, 35, 19, 39, 43, 44, 27, 50, 51, 56] are built upon ISS models and aim to consider adding different temporal relations of neighbor frames to VSS jobs for better segmentation results. VSS extends ISS to video sequences, aiming to assign semantic labels to every pixel in each frame while considering temporal consistency. Despite these advances, VSS models often face challenges in balancing accuracy and efficiency. High-performing models tend to be computationally intensive, making them unsuitable for real-time applications on devices with limited resources.

Chapter 3

Methodology

In this section, we present our approach for efficient video semantic segmentation, which integrates extremely low-bit quantization, N:M structured sparsity with channel permutation, and knowledge distillation. Our method aims to significantly reduce the computational complexity and memory footprint of semantic segmentation models, making them suitable for deployment on resource-constrained devices without substantial loss in segmentation accuracy.

3.1 Background

Quantization is a technique used to reduce the precision of weights and activations in neural networks, thereby decreasing model size and computational requirements. The key idea is to map continuous-valued parameters to a finite set of discrete levels, effectively compressing the model and enabling faster computations due to reduced bit-width operations.

Given a real-valued weight or activation x, quantization maps x to a quantized value $\hat{x}$ using a quantization function $Q(\cdot)$:

$$\hat{x} = Q(x). \tag{3.1}$$

One common approach is **uniform affine quantization**, where the quantization function is defined as:

$$\hat{x} = \text{clamp}\left(\text{round}\left(\frac{x - z}{s}\right), Q_n, Q_p\right), \tag{3.2}$$

where s is the scaling factor (step size), z is the zero-point offset, $\text{round}(\cdot)$ rounds to the nearest integer, Q_n and Q_p are the minimum and maximum quantization levels, and $\text{clamp}(\cdot)$ ensures the quantized value stays within the representable range.

For fixed-point quantization with n_{bits} bits, the number of quantization levels L is $2^{n_{\text{bits}}}$. The quantization levels can be represented as:

$$L = \{l \mid l = s \cdot k + z, \quad k \in [Q_n, Q_p]\}, \tag{3.3}$$

with $Q_n = 0$ and $Q_p = 2^{n_{\text{bits}}} - 1$ for unsigned quantization, or $Q_n = -2^{n_{\text{bits}-1}}$ and $Q_p = 2^{n_{\text{bits}-1}} - 1$ for signed quantization.

The scaling factor s and zero-point z are crucial for matching the dynamic range of x to the representable quantization levels. They are typically computed based on the minimum and maximum values of x:

$$s = \frac{x_{\text{max}} - x_{\text{min}}}{Q_p - Q_n}, \quad z = \text{round}\left(Q_n - \frac{x_{\text{min}}}{s}\right). \tag{3.4}$$

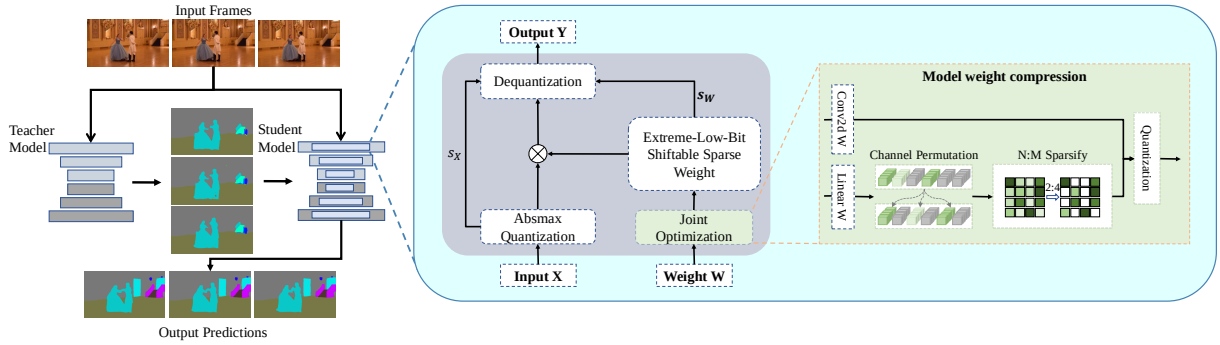

Figure 3.1: An overview of computation flow of our proposed QLinear and QConv2d.

However, in many applications, especially with symmetric quantization (zero-point $z = 0$), we simplify to:

$$s = \frac{\max(|x|)}{Q_p}. \tag{3.5}$$

Quantization can be applied to both weights and activations.

Aggressive quantization, such as using very low bit-widths (e.g., 1-bit weights), can lead to significant model compression and speed-up but may result in severe accuracy degradation due to the loss of information. To mitigate this, **quantization-aware training (QAT)** incorporates quantization effects during the training process, allowing the model to adjust its parameters to compensate for quantization errors.

3.2 QLinear and QConv2d Layers

To implement extremely low-bit quantization in our model, we introduce quantization-aware versions of linear and convolutional layers, termed *QLinear* and *QConv2d*, respectively. These layers incorporate quantization operations within their computations, enabling the model to learn quantized representations during training while maintaining computational efficiency.

Our quantization scheme focuses on extremely low-bit weights, specifically W3A8 (weights quantized to 3 bits, activations to 8 bits), W2A8, and W1A8. For each quantization bit-width, we define specific quantization levels suitable for shifting operations to appropriately represent the weight values after scaling. For instance, the quantization levels for 3-bit weights are $L = \{-8, -4, -2, -1, 1, 2, 4, 8\}$, for 2-bit weights $L = \{-2, -1, 1, 2\}$, and for 1-bit weights $L = \{-1, 1\}$.

To accommodate variations in weight distributions across different channels, we apply per-channel scaling factors. Given a weight tensor $\mathbf{W}$, the scaling factor $\mathbf{s_W}$ is computed as:

$$\mathbf{s_W} = \frac{1}{\text{mean}(|\mathbf{W}|,\ \text{dims}) + \epsilon}, \tag{3.6}$$

where $\epsilon = 10^{-5}$ prevents division by zero, and dims specifies the dimensions over which the mean is computed, such as per-output channel for linear layers or over kernel dimensions for convolutional layers.

The quantization process involves scaling the weights, quantizing them to the nearest level in the predefined set L, and incorporating the scaling factor during inference. Specifically, the scaled weights are computed as $\tilde{\mathbf{W}} = \mathbf{s_W} \odot \mathbf{W}$. Each element $\tilde{w}_i$ is then quantized to the nearest level in L:

$$\hat{w}_i = \arg\min_{l \in L} |\tilde{w}_i - l|. \tag{3.7}$$

For activations, we use 8-bit quantization with a global scaling factor based on the maximum absolute activation value:

$$s_{\mathbf{x}} = \frac{Q_p}{\max(|\mathbf{x}|) + \epsilon}, \tag{3.8}$$

where $Q_p = 2^{n_{\text{bits}}-1} - 1$. The quantized activations are then computed by clamping and scaling the activations accordingly:

$$\hat{\mathbf{x}} = s_{\mathbf{x}} \cdot \text{clamp}\left(\text{round}\left(\frac{\mathbf{x}}{s_{\mathbf{x}}}\right), Q_n, Q_p\right), \tag{3.9}$$

with $Q_n = -2^{n_{\text{bits}}-1}$ and $\text{clamp}(z, Q_n, Q_p) = \min\left(\max\left(z, Q_n\right), Q_p\right)$.

In the *QLinear* layer, we extend the standard linear layer by incorporating quantization and an optional channel permutation strategy to enhance the retention of significant weights during pruning. During the forward pass, the weights are first scaled and quantized as described above. If channel permutation is employed, we rearrange the weight matrix columns based on the magnitude of their corresponding channels to distribute important weights evenly, which is particularly beneficial when combined with N:M sparsity (discussed in Section 3.3).

The input activations are quantized using the computed scaling factor $s_{\mathbf{x}}$. The output is then computed using the quantized weights and activations:

$$\mathbf{y} = \hat{\mathbf{W}}\hat{\mathbf{x}} + \mathbf{b}, \tag{3.10}$$

where $\hat{\mathbf{W}}$ and $\hat{\mathbf{x}}$ are the quantized weights and activations.

Similarly, the *QConv2d* layer incorporates quantization into convolutional operations. The convolutional kernels are scaled and quantized in the same manner as the linear weights. The input feature maps are quantized using the global scaling factor $s_{\mathbf{x}}$. The convolution operation is then performed with the quantized weights and activations:

$$\mathbf{Y} = \text{QConv2D}\left(\hat{\mathbf{X}},\ \hat{\mathbf{W}}\right) + \mathbf{b}. \tag{3.11}$$

By integrating the quantization steps within these layers, we ensure that quantization effects are taken into account during training, allowing the model to adjust its parameters accordingly and improve performance despite the severely reduced precision. And during inference, both the scaling factors $\mathbf{s_W}$ and $s_{\mathbf{x}}$ can be applied after the matrix multiplication $\hat{\mathbf{W}}\hat{\mathbf{x}}$ for the *QLinear* layer, and the convolutional operation for the *QConv2d* layers, so the the matrix multiplications and the convolutional operations in our case can be done in shifting operations.

3.3 N:M Sparsity and Channel Permutation

To further reduce model complexity and enhance computational efficiency, we incorporate *N:M structured sparsity* into our network architecture, augmented with a *channel permutation* strategy to preserve important weights during pruning. This combination allows us to significantly compress the model while maintaining high performance in video semantic segmentation tasks.

In N:M pruning, within every group of M weights, at least N are zero. This fine-grained sparsity pattern is particularly amenable to hardware acceleration on modern GPUs that support sparse matrix operations, improving inference speed and reducing memory consumption.

Given a weight tensor $\mathbf{W}$, we first ensure that its size is compatible with the N:M sparsity pattern. If necessary, we pad $\mathbf{W}$ with zeros so that its total number of elements n becomes divisible by M. The padded tensor is then reshaped into $B = n'/M$ blocks of size M, where n' is the adjusted total number of elements after padding.

Within each block $\mathbf{w}_b \in \mathbb{R}^M$ for $b = 1, \ldots, B$, we identify the indices $\mathcal{I}_b$ of the $M - N$ elements with the largest magnitudes:

$$\mathcal{I}_b = \arg\operatorname{top}_{M-N}(|\mathbf{w}_b|). \tag{3.12}$$

A binary mask $\mathbf{m}_b \in \{0,1\}^M$ is created, where $m_{b,i} = 1$ if $i \in \mathcal{I}_b$ and $m_{b,i} = 0$ otherwise. The sparsified block $\mathbf{s}_b$ is obtained by element-wise multiplication:

$$\mathbf{s}_b = \mathbf{w}_b \odot \mathbf{m}_b, \tag{3.13}$$

where $\odot$ denotes the Hadamard (element-wise) product. The sparsified weight tensor $\mathbf{W}_{\text{sparse}}$ is reconstructed by concatenating all $\mathbf{s}_b$ and trimming any padded elements to restore the original shape.

While N:M sparsity effectively reduces the number of parameters, naïve application can inadvertently remove many important weights, leading to significant accuracy degradation. To address this, we incorporate our *channel permutation* strategy designed to maximize the retention of significant weights after pruning.

The channel permutation process begins by computing a magnitude score for each input channel (column in the weight matrix). For a scaled weight matrix $\tilde{\mathbf{W}}$, the magnitude of the j-th channel is calculated as:

$$c_j = \sum_i |\tilde{W}_{i,j}|. \tag{3.14}$$

Channels are then sorted in descending order based on c_j, yielding a permutation index π. By rearranging the columns of $\tilde{\mathbf{W}}$ according to π, we redistribute the significant weights evenly across the weight matrix. This rearrangement ensures that each block of M weights contains a mix of high-magnitude and low-magnitude weights, enhancing the effectiveness of the N:M sparsity pattern by preserving critical information in each block.

After permuting the channels, we apply the N:M sparsification process to the permuted weight matrix. The sparsification retains the $M - N$ largest-magnitude weights in each block, as before, but due to the permutation, these weights are more likely to be significant. Once the sparsification is complete, we apply the inverse permutation π^{-1} to restore the original channel order. This step is crucial for maintaining compatibility with subsequent layers and preserving the structural integrity of the network. This is called psedo-permutation when training the model, we remember the permutation index π in the last training step and apply the true permutation in the corresponding *QLinear* layers during inference.

3.4 Knowledge Distillation

To mitigate the potential accuracy degradation caused by aggressive quantization and pruning, we employ *knowledge distillation* to transfer knowledge from a high-precision teacher model to our quantized and pruned student model. This technique enables the student model to learn from the teacher's output distributions, capturing essential information that might be lost due to reduced precision and sparsity.

In our framework, we use a high-precision model $\mathcal{T}$ as the teacher and the compressed model $\mathcal{S}$ as the student. During training, both models process the same input data $\mathbf{X}$, and the student is trained not only to predict the ground truth labels but also to mimic the output logits of the teacher model. This approach encourages the student to learn representations that are more similar to those of the teacher, improving its performance despite the limitations imposed by quantization and pruning.

We define the *distillation loss* $\mathcal{L}_{\text{distill}}$ as the Mean Squared Error (MSE) between the logits (pre-softmax outputs) of the student and teacher models:

$$\mathcal{L}_{\text{distill}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{z}_{\mathcal{S}}^{(i)} - \mathbf{z}_{\mathcal{T}}^{(i)} \right\|_2^2, \tag{3.15}$$

where N is the total number of output elements (e.g., pixels in segmentation), and $\mathbf{z}_{\mathcal{S}}^{(i)}$ and $\mathbf{z}_{\mathcal{T}}^{(i)}$ are the logits for the i-th element from the student and teacher models, respectively.

The overall training loss combines the standard segmentation loss $\mathcal{L}_{\text{seg}}$ with the distillation loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seg}} + \alpha \mathcal{L}_{\text{distill}}, \tag{3.16}$$

where α is a weighting factor that balances the influence of the distillation term. We set $\alpha = 0.15$ based on empirical observations to effectively combine the losses without overshadowing the primary segmentation objective.

During training, the student model processes the input data to produce output logits $\mathbf{z}_{\mathcal{S}} = \mathcal{S}(\mathbf{X})$, while the teacher model, operating in evaluation mode to maintain consistent outputs, generates logits $\mathbf{z}_{\mathcal{T}} = \mathcal{T}(\mathbf{X})$. The distillation loss $\mathcal{L}_{\text{distill}}$ is then computed using Equation (3.15), capturing the discrepancy between the student's and teacher's predictions.

In practice, knowledge distillation serves as an effective regularizer, guiding the learning process of the quantized and pruned model. It provides additional supervision that complements the ground truth labels, especially in regions where the teacher model's outputs offer informative gradients that the student can leverage. This synergy between the distillation loss and the segmentation loss leads to a more robust and performant model despite the aggressive compression techniques applied.

The integration of these techniques enables our model to achieve high efficiency in video semantic segmentation tasks while maintaining competitive accuracy compared to full-precision models.

Chapter 4

Experiments and Results

4.1 Experimental Setup

Datasets. We conduct our experiments on the Video Scene Parsing in the Wild (VSPW) dataset [39]. VSPW comprises 2,806 training clips with 198,244 frames, 343 validation clips with 24,502 frames, and 387 test clips with 28,887 frames. The dataset includes 124 semantic categories across diverse indoor and outdoor scenes, providing a comprehensive evaluation environment. Each video in VSPW is densely annotated at a high frame rate of 15 frames per second, ensuring detailed temporal information.

Implementation details. We implement our approach using the mmsegmentation codebase for CFFM and MRCFA models, and the Mask2Former codebase for Mask2Former and MPVSS, conducting experiments on NVIDIA GPUs. For CFFM and MRCFA, teacher models with MiT-B0, MiT-B1, and MiT-B2 backbones are trained on 4 GPUs with an initial learning rate of 6×10^{-5}, while quantized student models are trained on 8 GPUs using our proposed *QLinear* and *QConv2d* layers. The first encoder layer and the last decoder layer are kept in full precision, and for MRCFA, the final output layer of the SegFormer decoder is also left unquantized. Both student and teacher models are trained for 160k iterations using the AdamW optimizer with a "poly" learning rate schedule. For Mask2Former and MPVSS, teacher and student models with Swin-Tiny and Swin-Small backbones are trained on 8 GPUs, applying quantization similarly by keeping the first encoder layer and last decoder layer in full precision for Mask2Former. For MPVSS, we use the quantized Mask2Former while only quantizing the linear layers in the motion encoder, motion decoder, and flow head. The initial learning rate is set to 5×10^{-5}, and the models are trained for up to 90k iterations. Knowledge distillation is employed with a weighting factor of 0.15 to transfer knowledge from teacher to student models. We use data augmentations including random resizing, flipping, cropping, and photometric distortion. VSPW images are randomly cropped to the size of 480×480 to train the network, and for inference, the input images are resized to 480×853. We perform single-scale inference without post-processing.

Evaluation Metrics. We evaluate the segmentation results by adopting the commonly used metrics of Mean Intersection over Union (mIoU) and Weighted Intersection over Union (WIoU), following [48]. Additionally, we employ Video Consistency (VC) [39] to assess the temporal consistency of the predicted segmentation masks across adjacent frames. Specifically, for a sequence of C consecutive frames $\{\boldsymbol{I}_c\}_{c=1}^{C}$, the video consistency VC_n for n consecutive frames is computed as $\text{VC}_n = \frac{1}{C-n+1} \sum_{i=1}^{C-n+1} \frac{\left(\bigcap_{j=i}^{i+n-1} \boldsymbol{S}_j\right) \cap \left(\bigcap_{j=i}^{i+n-1} \boldsymbol{S}_j^{'}\right)}{\bigcap_{j=i}^{i+n-1} \boldsymbol{S}_j}$, where $\boldsymbol{S}_j$ and $\boldsymbol{S}_j^{'}$ denote the ground-truth mask and the predicted mask for the j^{th} frame, respectively, and $C \geq n$. We compute the mean video consistency mVC_n across all videos in the dataset by averaging VC_n for each video. Following [39], we calculate mVC_8 and mVC_{16} to evaluate the visual consistency of the predicted masks over longer temporal ranges. For further details on VC, please refer to [39].

Table 4.1: Comparison of state-of-the-art methods on the VSPW validation set [39]. **Ori.**: original model; **Cpr.**: compressed model. Lossless results; nearly lossless results.

Methods	Backbone	Compression Method	mIoU ↑		WIoU ↑		mVC_8 ↑		mVC_{16} ↑		Params ↓	Size Red. (%) ↑
			Ori.	Cpr.	Ori.	Cpr.	Ori.	Cpr.	Ori.	Cpr.		
DeepLabv3+ [3]	ResNet-101	None	34.7		58.8		83.2		78.2		62.7	
UperNet [59]	ResNet-101	None	36.5		58.6		82.6		76.1		83.2	
PSPNet [68]	ResNet-101	None	36.5		58.1		84.2		79.6		70.5	
OCRNet [65]	ResNet-101	None	36.7		59.2		84.0		79.0		58.1	
ETC [35]	OCRNet	None	37.5		59.1		84.1		79.1		58.1	
NetWarp [13]	OCRNet	None	37.5		58.9		84.0		79.0		58.1	
TCB [39]	ResNet-101	None	37.8		59.5		87.9		84.0		70.5	
SegFormer [61]	MiT-B0	None	32.9		56.8		82.7		77.3		3.8	
SegFormer [61]	MiT-B1	None	36.5		58.8		84.7		79.9		13.8	
SegFormer [61]	MiT-B2	None	43.9		63.7		86.0		81.2		24.8	
CFFM [50]	MiT-B0	BitNet b1.58 [38]	35.4	27.9	58.5	52.7	87.7	86.9	82.9	81.8	4.7 → 1.6	65.9%
CFFM [50]	MiT-B0	2 : 4 Pruning [69]	35.4	34.3	58.5	58.4	87.7	87.5	82.9	82.4	4.7 → 3.1	34.6%
CFFM [50]	MiT-B0	1 : 4 Pruning [69]	35.4	34.9	58.5	58.4	87.7	87.6	82.9	82.8	4.7 → 3.9	17.3%
CFFM [50]	MiT-B0	Ours	35.4	33.2	58.5	57.5	87.7	87.5	82.9	82.8	4.7 → 1.3	72.7%
CFFM [50]	MiT-B1	Ours	38.5	38.5	60.0	60.3	88.6	89.1	84.1	84.7	15.5 → 3.7	76.4%
CFFM [50]	MiT-B2	Ours	44.9	43.8	64.9	63.9	89.8	90.5	85.8	86.6	26.5 → 4.7	82.3%
MRCFA [51]	MiT-B0	Ours	35.2	32.8	57.9	57.1	88.0	87.2	83.2	82.2	5.2 → 1.3	74.8%
MRCFA [51]	MiT-B1	Ours	38.9	35.7	60.0	58.1	88.8	87.8	84.4	82.8	16.2 → 3.7	77.4%
MRCFA [51]	MiT-B2	Ours	45.3	42.3	64.7	63.4	90.3	89.5	86.2	84.9	27.3 → 4.7	82.7%
Mask2Former [5]	Swin-T	Ours	41.2	39.3	62.6	61.9	84.5	80.6	80.0	75.3	47.4 → 5.4	88.6%
Mask2Former [5]	Swin-S	Ours	45.5	44.8	64.9	64.6	84.7	81.7	79.3	76.4	68.9 → 7.5	89.2%
MPVSS [56]	Swin-T	Ours	39.9	38.1	62.0	60.5	85.9	85.9	80.4	80.3	114.0 → 48.6	57.3%
MPVSS [56]	Swin-S	Ours	43.7	43.3	63.7	63.1	87.2	87.1	81.9	81.8	108.0 → 38.6	64.3%

4.2 Main Results

We evaluate the effectiveness of our proposed method on the VSPW dataset [39] by quantizing and pruning several state-of-the-art video semantic segmentation models, using 3-bit quantization and $1:4$ sparsity. The results are summarized in Table 4.1. Our method demonstrates that it can significantly reduce the computational complexity and memory footprint of these models while maintaining high segmentation accuracy and temporal consistency.

For the CFFM model [50], our quantized and pruned version with the MiT-B0 backbone achieves an mIoU of 33.21%, closely matching the original full-precision model. When using larger backbones like MiT-B1 and MiT-B2, the mIoU increases to 38.50% and 43.83%, respectively. Similar trends are observed for the MRCFA model [51], where the quantized and pruned models achieve mIoU scores of 32.78%, 35.68%, and 42.34% for the MiT-B0, MiT-B1, and MiT-B2 backbones, respectively. Our method also maintains high temporal consistency, as evidenced by the mVC8 and mVC16 metrics. For instance, the quantized CFFM model with the MiT-B2 backbone attains mVC8 of 90.50% and mVC16 of 86.62%, indicating that the segmentation results are consistent across consecutive frames.

Moreover, we apply our method to Swin-Transformer-based architectures like Mask2Former [5] and MPVSS [56] with Swin-Tiny and Swin-Small backbones. The quantized Mask2Former models achieve mIoU scores of 39.29% and 44.77%, while the quantized MPVSS models reach 38.06% and 43.26% mIoU for the Swin-Tiny and Swin-Small backbones, respectively. These results demonstrate that our approach generalizes well across different architectures and backbones.

In addition to evaluating our proposed quantization and pruning method, we compare it with other compression techniques such as BitNet b1.58 [38] and structured pruning with $2:4$ or $1:4$ sparsity. For the CFFM model with the MiT-B0 backbone, BitNet b1.58 reduces the model size significantly by quantizing

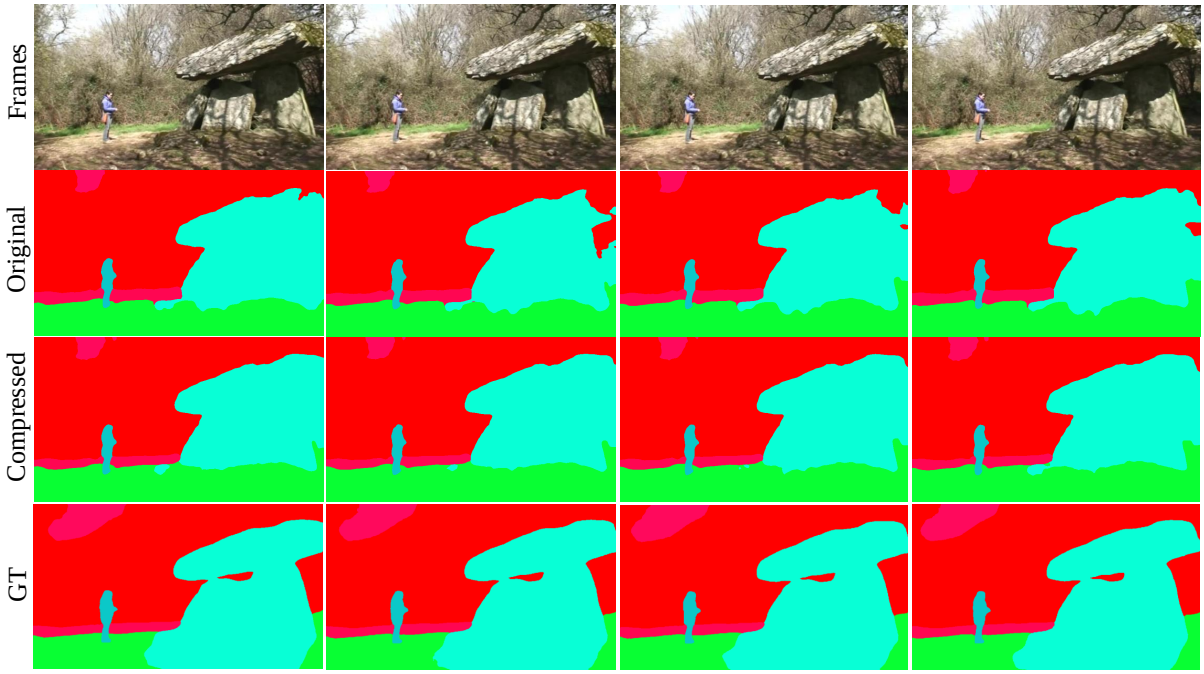

Figure 4.1: Qualitative results of **CFFM-B2** compressed by our method using 3-bit quantization and $1:4$ sparsity.

Table 4.2: Ablation study on the influence of N:M sparsity level.

Methods	N	M	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
CFFM [50]	3	4	27.27	86.00	80.54
	2	4	31.32	87.74	83.02
	1	4	33.21	87.53	82.79
	0	4	33.71	88.26	83.74
MRCFA [51]	3	4	27.50	85.02	80.20
	2	4	31.10	86.17	80.67
	1	4	32.78	87.22	82.17
	0	4	32.54	87.48	82.38

Table 4.3: Ablation study on excluding zero in quantization levels after N:M sparsification.

Methods	Quantization Levels	N	M	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
MRCFA [51]	Including Zero	2	4	30.75	85.85	80.43
		1	4	32.50	87.05	81.90
	Excluding Zero	2	4	31.10	86.17	80.67
		1	4	32.78	87.22	82.17

linear modules to 1.58 bit but results in an mIoU drop to 27.9%. Similarly, applying $2:4$ and $1:4$ structured pruning alone yields mIoU scores of 34.3% and 34.9%, respectively, with less reduction in model size. In contrast, our approach achieves a better balance between model size reduction and segmentation accuracy, obtaining an mIoU of 33.21% and mVC_8 of 87.53% at a higher compression ratio. These comparisons indicate that while other methods can compress models, they often lead to more significant performance degradation or less efficient compression, whereas our method more effectively preserves accuracy and

Table 4.4: Ablation study on the influence of quantization level.

Methods	Bits	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
MRCFA [51]	1	27.80	85.33	80.28
	2	30.18	85.77	80.47
	3	32.78	87.22	82.17
CFFM [50]	1	27.68	86.81	81.75
	2	32.52	87.75	82.99
	3	33.21	87.53	82.79

temporal consistency.

To evaluate the efficiency of our compression method, we calculated the size reduction by determining the original model size as the total number of parameters multiplied by 32 bits (FP32), computing the quantized model size by summing the bit-lengths of quantized layers (3 bits per parameter) and non-quantized layers (32 bits per parameter). And for the N:M sparse matrix, we calculated the size reduction by multiplying the fraction of nonzero parameters by the bit-length of each parameter.

Overall, the experimental results confirm that our method effectively compresses video semantic segmentation models without significant loss in accuracy or temporal consistency, making it suitable for deployment on resource-constrained devices.

4.3 Ablation Studies

We conduct ablation studies to examine the effects of various components in our method, including N sparsity levels, exclusion of zero in quantization levels, quantization bit-width, distillation weight α, training iterations, and teacher model selection. Unless stated otherwise, we use the same training settings as described earlier.

N:M Sparsity Level. We explore how different N sparsity configurations affect model performance by varying N from 0 to 3 (with $M = 4$) using 3-bit quantization and the MiT-B0 backbone. Table 4.2 shows that for CFFM, decreasing N improves mIoU from 27.27% at $N = 3$ to 33.71% at $N = 0$. MRCFA shows a similar pattern, with mIoU peaking at 32.78% for $N = 1$. Lower N generally preserves more weights, enhancing accuracy and temporal consistency at the cost of reduced sparsity, highlighting a trade-off between compression and accuracy.

Excluding Zero in Quantization Levels. After applying N sparsity, excluding zero from quantization levels dedicates all quantization levels to representing non-zero weights. As shown in Table 4.3, excluding zero consistently improves performance for MRCFA. With a $1:4$ sparsity pattern, mIoU increases from 32.50% to 32.78% and mVC_8 from 87.05% to 87.22%. These results indicate that excluding zero allows better use of quantization levels for meaningful weights, enhancing model expressiveness and performance.

Quantization Bit-Width. Table 4.4 evaluates the impact of different quantization bit-widths for weights using $1:4$ sparsity. For CFFM, reducing precision from 3-bit to 2-bit reduces mIoU from 33.21% to 32.52%. MRCFA exhibits a similar trend. With 1-bit quantization, performance degrades significantly, suggesting extremely low-bit quantization may not suit smaller models with limited capacity to handle quantization noise.

Distillation Weight α. We investigate how the distillation weight α affects performance, as detailed in Table 4.5. For MRCFA, $\alpha = 0.15$ achieves the highest mIoU of 32.78%. Larger α values, such as 0.5, reduce mIoU, indicating that excessive emphasis on the distillation loss can negatively impact learning of the primary segmentation objective.

Training Iterations. We analyze the influence of training duration by training the CFFM and MRCFA models for both 160k and 250k iterations under 3-bit quantization with $1:4$ sparsity and the MiT-B0 backbone.

Table 4.5: Ablation study on the influence of distillation weight α.

Methods	α	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
MRCFA [51]	0.01	32.34	87.41	82.41
	0.02	32.38	87.38	82.40
	0.05	32.17	87.15	82.15
	0.1	32.31	87.30	82.21
	0.15	32.78	87.22	82.17
	0.2	32.19	87.10	82.14
	0.5	30.86	85.60	79.96

Table 4.6: Ablation study on the influence of training iterations.

Methods	iters	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
MRCFA [51]	160k	32.78	87.22	82.17
	250k	31.71	87.10	82.09
CFFM [50]	160k	33.21	87.53	82.79
	250k	33.35	87.90	83.18

Table 4.7: Ablation study on the influence teacher model selection.

Methods	Student	Teacher	mIoU ↑	mVC_8 ↑	mVC_{16} ↑
CFFM [50]	MiT-B0	MiT-B0	33.21	87.53	82.79
	MiT-B0	MiT-B1	34.00	88.35	83.86
	MiT-B0	MiT-B2	33.83	88.30	83.85
	MiT-B1	MiT-B1	38.50	89.09	84.69
	MiT-B1	MiT-B2	39.23	89.13	84.77
	MiT-B2	MiT-B2	43.83	90.50	86.62
MRCFA [51]	MiT-B0	MiT-B0	32.78	87.22	82.17
	MiT-B0	MiT-B1	31.92	87.00	81.64
	MiT-B0	MiT-B2	31.63	86.63	81.24
	MiT-B1	MiT-B1	36.9	87.84	82.78
	MiT-B1	MiT-B2	36.47	88.09	83.35
	MiT-B2	MiT-B2	42.34	89.50	84.91
Mask2Former [5]	Swin-T	Swin-T	39.29	80.63	75.27
	Swin-T	Swin-S	38.43	80.00	74.61
	Swin-S	Swin-S	44.77	81.72	76.40
MPVSS [56]	Swin-T	Swin-T	38.06	85.94	80.32
	Swin-T	Swin-S	37.85	85.70	79.97
	Swin-S	Swin-S	43.26	87.08	81.75

Table 4.6 shows that for MRCFA, extending training from 160k to 250k iterations slightly reduces mIoU from 32.78% to 31.71%, indicating potential overfitting. For CFFM, increasing iterations improves mIoU from 33.21% to 33.35%. These results suggest that optimal training duration may differ across architectures, necessitating careful tuning to prevent underfitting or overfitting in quantized and pruned models.

Teacher Model Selection. Table 4.7 explores how choosing different teacher models affects student model performance. For CFFM, using a larger backbone as a teacher can improve performance. For instance, a MiT-B0 student model's mIoU increases from 33.21% (with a MiT-B0 teacher) to 34.00% (with a MiT-B1 teacher). In other cases, improvements are less pronounced, suggesting that the effectiveness of knowledge distillation may depend on the compatibility between teacher and student architectures.

Chapter 5

Discussion

In this work, we integrated extremely low-bit quantization, N:M structured sparsity, and channel permutation to significantly reduce model size and computational demands in VSS. By minimizing precision to as low as ± 1, these techniques drastically reduce memory overhead and enable efficient, accelerated computations using simpler bitwise operations on general-purpose hardware like CPUs. This approach not only facilitates real-time segmentation on resource-limited devices such as PCs and smartphones but also reduces energy consumption and chip area requirements, making it highly suitable for deployment in edge devices and other constrained environments.

Chapter 6

Conclusion

We present a novel approach that combines quantization-aware training, N:M sparsity, and channel permutation strategies to create compact and efficient models for VSS. Our experiments on the VSPW dataset showed that this integrated method substantially reduces model complexity while maintaining high segmentation accuracy and temporal consistency. This balance between efficiency and performance is crucial for deploying state-of-the-art segmentation models in resource-constrained environments, enabling faster inference without significant loss in accuracy. Future research can explore further optimizations and the integration of additional compression techniques tailored for specialized hardware to enhance both the speed and accuracy of VSS models.

Appendix A

The First Appendix

A.1 Overview

In this appendix, we provide additional details to further support our main paper. Section A.2 presents a detailed description of our algorithm, including key steps and implementation specifics. In Section A.3, we offer a more in-depth analysis of our ablation studies to validate the effectiveness and robustness of our proposed method. Finally, Section A.4 showcases additional visual results, including qualitative comparisons and extra examples, to visually demonstrate the performance of our method on video semantic segmentation tasks.

A.2 Algorithm

In this section, we provide detailed descriptions of the algorithms employed in our approach to efficient neural network quantization. These algorithms are crucial for reducing the computational complexity and memory footprint of deep learning models, making them suitable for deployment in resource-constrained environments such as mobile devices and embedded systems.

N:M Sparsification. Algorithm 1 introduces the `N:M Sparsify` function, which enforces a structured sparsity pattern on a given tensor. By retaining only $M - N$ significant elements out of every M in the tensor, we reduce the number of non-zero parameters, leading to computational efficiency during inference.

Activation Quantization. Algorithm 2 presents the `activation_quant` function used for quantizing activations during both training and inference. The function computes a scaling factor based on the maximum absolute value of the activation tensor and quantizes the activations to a specified bit-width b. During training, activations are dequantized after quantization to allow gradient computation. In contrast, during inference, the function returns the quantized activations along with the scaling factor for efficient computation.

***QConv2d* Weight Quantization.** Algorithm 3 details the weight quantization process for convolutional layers using the `weight_quant` function. The weights are scaled and quantized to powers of two, which simplifies the hardware implementation by replacing multiplication operations with bit-shift operations.

***QLinear* Weight Quantization with N:M Sparsification and Permutation.** Algorithm 4 describes the `weight_quant` function for linear layers, incorporating N:M sparsification and optional permutation. The function computes the scaling factor, applies permutation to evenly distribute important weights, performs N:M sparsification to enforce structured sparsity, and quantizes the remaining non-zero weights.

***QLinear* Forward Pass During Training.** Algorithm 5 outlines the forward pass of the *QLinear* layer during training. If quantization is enabled, both the weights and activations are quantized using their respective

quantization functions. The output is then computed using these quantized values, ensuring that the model learns to operate effectively with quantized parameters.

***QConv2d* Forward Pass During Training.** Algorithm 6 presents the forward pass of the *QConv2d* layer during training. Similar to the linear layer, the weights and activations are quantized if quantization is enabled. The convolution operation is performed using the quantized weights and activations, facilitating the training of models that are robust to quantization effects.

***QLinear* Forward Pass During Inference.** Algorithm 7 describes the inference-time forward pass of the *QLinear* layer. It utilizes precomputed quantized weights and scaling factors. If permutation is required and has not been done yet, it permutes the weights and biases of both the current and previous layers. The activations are quantized in real-time, and the final output is computed using efficient integer arithmetic followed by scaling.

***QConv2d* Forward Pass During Inference.** Algorithm 8 illustrates the inference-time forward pass of the *QConv2d* layer. The quantized weights and scaling factors are precomputed, and the input activations are quantized on-the-fly. The convolution operation is then performed using these quantized values, and the result is scaled appropriately to obtain the final output.

Training Procedure. Algorithm 9 provides an overview of the training procedure for efficient video semantic segmentation. The student model $\mathcal{S}$, which consists of quantized layers, is trained using a combination of segmentation loss and distillation loss from a pre-trained teacher model $\mathcal{T}$. This approach ensures that the student model not only learns to perform the segmentation task effectively but also mimics the performance of the larger teacher model, leading to improved accuracy.

By integrating these algorithms, our method achieves a balance between computational efficiency and model performance. The quantization and sparsification techniques reduce the resource requirements, while the training procedure ensures that the accuracy remains competitive.

Algorithm 1 `N:M Sparsify` Function

Require: Tensor $\mathbf{t}$, integers N, M
Ensure: Sparse tensor $\mathbf{s}$

original_shape $\leftarrow$ shape of $\mathbf{t}$
$n \leftarrow$ total number of elements in $\mathbf{t}$
if $n \bmod M \neq 0$ **then**
    padding $\leftarrow M - (n \bmod M)$
    Pad $\mathbf{t}$ with zeros to length $n +$ padding
end if
Reshape $\mathbf{t}$ to matrix $\mathbf{T}$ of shape $\left(\frac{n+\text{padding}}{M},\ M\right)$
for each row $\mathbf{t}_i$ in $\mathbf{T}$ **do**
    Compute indices $\mathcal{I}_i$ of the $M - N$ largest $|t_{i,j}|$
    Initialize mask $\mathbf{m}_i \in \{0, 1\}^M$ with zeros
    for each index j in $\mathcal{I}_i$ **do**
        Set $m_{i,j} \leftarrow 1$
    end for
    Compute sparse row $\mathbf{s}_i \leftarrow \mathbf{t}_i \odot \mathbf{m}_i$
end for
Reshape $\{\mathbf{s}_i\}$ back to original_shape to obtain $\mathbf{s}$
return $\mathbf{s}$

Algorithm 2 `activation_quant` Function

Require: Activation tensor $\mathbf{x}$, quantization bits b, mode (`training` or `inference`)

Ensure: Quantized activation tensor $\mathbf{q}$ (during training), or quantized activation tensor $\mathbf{q}$ and scaling factor $s_{\mathbf{x}}$ (during inference)

1: Compute scaling factor:

$$Q_n = -(2^{b-1}), \quad Q_p = 2^{b-1} - 1 \tag{A.1}$$

$$s_{\mathbf{x}} = \frac{Q_p}{\max\left(\max\left(|\mathbf{x}|\right),\ \epsilon\right)} \tag{A.2}$$

2: Quantize activations:

$$\mathbf{q} = \text{clip}\left(\text{round}\left(\mathbf{x} \times s_{\mathbf{x}}\right),\ Q_n,\ Q_p\right) \tag{A.3}$$

3: **if** mode is `training` **then**

4: Dequantize activations for gradient computation:

$$\mathbf{q} = \frac{\mathbf{q}}{s_{\mathbf{x}}} \tag{A.4}$$

5: **return** $\mathbf{q}$

6: **else**

7: **return** $\mathbf{q}$, $s_{\mathbf{x}}$

8: **end if**

Algorithm 3 *QConv2d*'s `weight_quant` Function

Require: Weight tensor $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K_h \times K_w}$, quantization bits b

Ensure: Quantized weight tensor $\mathbf{Q}$, scaling factor s

1: Compute scaling factor:

$$\mathbf{s_W} = \frac{1}{\max\left(\text{mean}\left(|\mathbf{W}|\right),\ \epsilon\right)} \tag{A.5}$$

2: Scale weights:

$$\mathbf{S} = \mathbf{W} \times \mathbf{s_W} \tag{A.6}$$

3: Define quantization levels:

$$\mathcal{L} = \left\{ \pm 2^k \;\middle|\; k = 0, 1, \ldots, b-1 \right\} \tag{A.7}$$

4: Initialize quantized weights:

$$\mathbf{Q} = \mathbf{0} \quad \text{(same shape as } \mathbf{S}\text{)} \tag{A.8}$$

5: **for** each element (i, j, k, l) in $\mathbf{S}$ **do**

6: Quantize weight:

$$\mathbf{Q}_{i,j,k,l} = \arg\min_{q \in \mathcal{L}} |\mathbf{S}_{i,j,k,l} - q| \tag{A.9}$$

7: **end for**

8: **return** $\mathbf{Q}$, s

Algorithm 4 *QLinear*'s `weight_quant` Function with N:M Sparsification and Optional Permutation

Require: Weight tensor $\mathbf{W}$, quantization bits b, integers N, M, boolean flag `permute`
Ensure: Quantized weight tensor $\mathbf{Q}$, scaling factor s, permutation indices $\mathbf{p}$ (if applicable)

Compute scaling factor:

$$\mathbf{s_W} = \frac{1}{\max\left(\text{mean}\left(|\mathbf{W}|\right),\ \epsilon\right)} \tag{A.10}$$

Scale weights:

$$\mathbf{S} = \mathbf{W} \times \mathbf{s_W} \tag{A.11}$$

if `permute` is **True then**
    Compute importance scores:

$$\mathbf{c} = \sum_i |\mathbf{S}_{i,j}| \tag{A.12}$$

    Sort indices $\mathbf{p}$ such that:

$$\mathbf{c}_{\mathbf{p}(1)} \geq \mathbf{c}_{\mathbf{p}(2)} \geq \dots \tag{A.13}$$

    Permute weights:

$$\mathbf{S} = \mathbf{S}[:, \mathbf{p}] \tag{A.14}$$

else
    Set $\mathbf{p}$ as identity permutation
end if
Flatten and apply N:M sparsification to $\mathbf{S}$
Reshape sparsified weights back to original shape
if `permute` is **True then**
    Inverse permute to restore original order:

$$\mathbf{S} = \mathbf{S}[:, \mathbf{p}^{-1}] \tag{A.15}$$

end if
Define quantization levels (excluding zero):

$$\mathcal{L} = \left\{ \pm 2^k \,\middle|\, k = 0, 1, \dots, b-1 \right\} \tag{A.16}$$

Quantize non-zero weights:

$$\mathbf{Q}_{i,j} = \begin{cases} 0, & \text{if } \mathbf{S}_{i,j} = 0 \\ \arg\min_{l \in \mathcal{L}} |\mathbf{S}_{i,j} - l|, & \text{if } \mathbf{S}_{i,j} \neq 0 \end{cases} \tag{A.17}$$

return $\mathbf{Q}$, $\mathbf{s_W}$, $\mathbf{p}$

Algorithm 5 `QLinear`'s Forward Pass During Training

Require: Input tensor $\mathbf{x}$, weight matrix $\mathbf{W}$, bias vector $\mathbf{b}$ (if any), quantization bits b_w, b_x, boolean flag `quantize`

Ensure: Output tensor $\mathbf{y}$

if `quantize` is **True then**
  Quantize weights using `weight_quant` **function**:
  $\mathbf{Q_W}, \mathbf{s_W}, \mathbf{p} \leftarrow \texttt{weight\_quant}(\mathbf{W}, b_w)$
  Compute scaled quantized weights:
  $\mathbf{W}_{\text{quant}} \leftarrow \dfrac{\mathbf{Q_W}}{\mathbf{s_W}}$
  Quantize activations using `activation_quant` **function**:
  $\mathbf{x}_{\text{quant}} \leftarrow \texttt{activation\_quant}(\mathbf{x}, b_x)$
  Compute output:
  $\mathbf{y} \leftarrow \mathbf{x}_{\text{quant}} \times \mathbf{W}_{\text{quant}}^{\top}$
else
  Compute output without quantization:
  $\mathbf{y} \leftarrow \mathbf{x} \times \mathbf{W}^{\top}$
end if
if bias $\mathbf{b}$ exists **then**
  Add bias:
  $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{b}$
end if
return $\mathbf{y}$

Algorithm 6 `QConv2d`'s Forward Pass During Training

Require: Input tensor $\mathbf{x}$, weight tensor $\mathbf{W}$, bias tensor $\mathbf{b}$ (if any), quantization bits b_w, b_x, boolean flag `quantize`

Ensure: Output tensor $\mathbf{y}$

if `quantize` is **True then**
  Quantize weights using `weight_quant` **function**:
  $\mathbf{Q_W}, \mathbf{s_W} \leftarrow \texttt{weight\_quant}(\mathbf{W}, b_w)$
  Compute scaled quantized weights:
  $\mathbf{W}_{\text{quant}} \leftarrow \dfrac{\mathbf{Q_W}}{\mathbf{s_W}}$
  Quantize activations using `activation_quant` **function**:
  $\mathbf{x}_{\text{quant}} \leftarrow \texttt{activation\_quant}(\mathbf{x}, b_x)$
  Compute convolution output:
  $\mathbf{y} \leftarrow \texttt{Conv2D}(\mathbf{x}_{\text{quant}}, \mathbf{W}_{\text{quant}}, \mathbf{b})$
else
  Compute convolution output without quantization:
  $\mathbf{y} \leftarrow \texttt{Conv2D}(\mathbf{x}, \mathbf{W}, \mathbf{b})$
end if
return $\mathbf{y}$

Algorithm 7 *QLinear*'s Forward Pass During Inference

Require: Input tensor $\mathbf{x}$, precomputed quantized weight tensor $\mathbf{Q_W}$, scaling factor $\mathbf{s_W}$, bias $\mathbf{b}$ (if any), permutation indices $\mathbf{p}$ (if applicable), previous layer quantized weights $\mathbf{Q}_{\text{prev}}$ and bias $\mathbf{b}_{\text{prev}}$ (if any), boolean flags `permute`, `permutation_done`, quantization bits b_x

Ensure: Output tensor $\mathbf{y}$

if `permute` is **True** and `permutation_done` is **False** and previous layer is also *QLinear* **then**
    Permute previous layer weights and biases:
    $\mathbf{Q}_{\text{prev}} \leftarrow \mathbf{Q}_{\text{prev}}[\mathbf{p}, :]$
    If bias $\mathbf{b}_{\text{prev}}$ exists, permute: $\mathbf{b}_{\text{prev}} \leftarrow \mathbf{b}_{\text{prev}}[\mathbf{p}]$
    Permute current layer weights:
    $\mathbf{Q_W} \leftarrow \mathbf{Q_W}[:, \mathbf{p}]$
    Bias $\mathbf{b}$ does not require permutation
    Set `permutation_done` $\leftarrow$ **True**
end if
Quantize activations using `activation_quant` **function**:
$\mathbf{Q_x}, s_{\mathbf{x}} \leftarrow \texttt{activation\_quant}(\mathbf{x}, b_x, \texttt{inference})$
Compute output:

$$\mathbf{y} = \frac{\mathbf{Q_x} \times \mathbf{Q_W^\top}}{\mathbf{s_W} \times s_{\mathbf{x}}} \tag{A.18}$$

if bias $\mathbf{b}$ exists **then**
    Add bias:
    $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{b}$
end if
return $\mathbf{y}$

Algorithm 8 *QConv2d*'s Forward Pass During Inference

Require: Input tensor $\mathbf{x}$, quantized weight tensor $\mathbf{Q_W}$, weight scaling factor $\mathbf{s_W}$, bias $\mathbf{b}$ (if any), quantization bits b_x

Ensure: Output tensor $\mathbf{y}$

Quantize activations using `activation_quant` **function**:
$\mathbf{Q_x}, s_{\mathbf{x}} \leftarrow \texttt{activation\_quant}(\mathbf{x}, b_x, \texttt{inference})$
Compute convolution output:

$$\mathbf{y} = \frac{\texttt{Conv2D}(\mathbf{Q_x}, \mathbf{Q_W}, \mathbf{b})}{\mathbf{s_W} \times s_{\mathbf{x}}} \tag{A.19}$$

return $\mathbf{y}$

A.3 More Analysis on Ablation Studies

In this section, we provide a more in-depth analysis of the ablation studies presented in the main paper. We delve deeper into the effects of various components of our method, including N:M sparsity levels, the exclusion of zero in quantization levels, quantization bit-width, the distillation weight α, training iterations, and teacher model selection. Our goal is to offer a more comprehensive understanding of how these factors influence the performance of our quantized sparse models for video semantic segmentation.

Algorithm 9 Training Procedure for Efficient Video Semantic Segmentation

Require:
- Pre-trained teacher model $\mathcal{T}$
- Student model $\mathcal{S}$ with *QLinear* and *QConv2d* layers
- Training data $\mathcal{D}$
- Weighting factor α

Ensure: Trained student model $\mathcal{S}$

1: **for** each mini-batch $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{D}$ **do**
2: **Student Forward Pass**: Compute $\mathbf{z}_{\mathcal{S}} = \mathcal{S}(\mathbf{X})$
3: **Segmentation Loss**: Compute $\mathcal{L}_{\text{seg}}$ between $\mathbf{z}_{\mathcal{S}}$ and ground truth $\mathbf{Y}$
4: **Teacher Forward Pass**: Compute $\mathbf{z}_{\mathcal{T}} = \mathcal{T}(\mathbf{X})$
5: **Distillation Loss**: Compute

$$\mathcal{L}_{\text{distill}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{z}_{\mathcal{S}}^{(i)} - \mathbf{z}_{\mathcal{T}}^{(i)} \right\|_2^2 \tag{A.20}$$

6: **Total Loss**: Compute

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seg}} + \alpha \mathcal{L}_{\text{distill}} \tag{A.21}$$

7: **Backpropagation**: Compute gradients and update $\mathcal{S}$'s parameters
8: **end for**

A.3.1 Impact of N:M Sparsity Levels

We evaluate how different N:M sparsity levels affect the performance of our quantized models. Specifically, we varied the value of N from 3 to 0 while keeping M fixed at 4, effectively adjusting the sparsity pattern from highly sparse ($N = 3$) to dense ($N = 0$) configurations. The experiments were performed using the CFFM [50] and MRCFA [51] models with the MiT-B0 backbone, trained with 3-bit quantization for 160k iterations.

Table 2 presents the results of this study. For the **CFFM** model, we observe that as N decreases (i.e., the model becomes less sparse), the Mean Intersection over Union (mIoU) improves significantly. When $N = 3$, the model achieves an mIoU of 27.27%. Reducing N to 2 increases the mIoU to 31.32%, indicating that retaining more non-zero weights per block enhances the model's ability to learn meaningful representations. Further decreasing N to 1 results in an mIoU of 33.21%, and at $N = 0$ (no enforced sparsity), the mIoU slightly increases to 33.71%. This trend suggests that while reducing sparsity (lower N) generally benefits model accuracy, the gains diminish as the model becomes denser.

Similarly, for the **MRCFA** model, the mIoU improves from 27.50% at $N = 3$ to 31.10% at $N = 2$. The highest mIoU of 32.78% is achieved at $N = 1$, but interestingly, the mIoU slightly decreases to 32.54% when N is further reduced to 0. This indicates that for the MRCFA model, an optimal balance between sparsity and performance is achieved at $N = 1$, and making the model denser does not provide additional benefits and may even slightly degrade performance.

The temporal consistency metrics, mVC_8 and mVC_{16}, exhibit similar patterns for both models. For the CFFM model, mVC_8 increases from 86.00% at $N = 3$ to 88.26% at $N = 0$, and mVC_{16} improves from 80.54% to 83.74%. For the MRCFA model, both mVC metrics peak at $N = 1$, with mVC_8 at 87.22% and mVC_{16} at 82.17%, before slightly decreasing at $N = 0$.

These results highlight a trade-off between model sparsity and performance. While reducing N (leading to denser models) generally improves accuracy and temporal consistency, the improvements become marginal beyond a certain point, and in some cases, excessive density may not yield additional benefits. For

instance, the negligible difference between $N = 1$ and $N = 0$ for the CFFM model suggests that enforcing a 1:4 sparsity pattern ($N = 1, M = 4$) can effectively compress the model without significant loss in performance.

In practical terms, selecting $N = 1$ offers a good balance between model size reduction and accuracy preservation. It allows for substantial compression due to the structured sparsity, facilitating efficient computation and memory usage, while maintaining high segmentation performance and temporal consistency. This balance is crucial for deploying models on resource-constrained devices where both efficiency and accuracy are important.

A.3.2 Effect of Excluding Zero in Quantization Levels

We investigate the impact of excluding zero from the quantization levels after applying N:M structured sparsity. In our approach, after enforcing N:M sparsity, a fixed pattern of zeros is introduced into the weight matrices. When performing quantization, including zero as one of the quantization levels may not be the most efficient use of the limited quantization levels, since zeros are already enforced by the sparsity pattern. By excluding zero from the quantization levels, we allocate all available quantization levels to represent the non-zero weights, potentially enhancing the expressiveness of the quantized model.

Table 3 presents the results of this study using the MRCFA [51] model with the MiT-B0 backbone, trained with 3-bit quantization for 160k iterations. We evaluated two sparsity patterns: $N = 2, M = 4$ and $N = 1, M = 4$. For each pattern, we compared the performance of models where zero is included in the quantization levels versus models where zero is excluded.

For the sparsity pattern $N = 2, M = 4$, excluding zero from the quantization levels resulted in an improvement in mIoU from 30.75% to 31.10%. The temporal consistency metrics also improved, with mVC_8 increasing from 85.85% to 86.17% and mVC_{16} from 80.43% to 80.67%. Similarly, for the $N = 1, M = 4$ sparsity pattern, excluding zero led to an increase in mIoU from 32.50% to 32.78%. The mVC_8 metric improved from 87.05% to 87.22%, and mVC_{16} increased from 81.90% to 82.17%.

These consistent improvements across both sparsity patterns indicate that excluding zero from the quantization levels allows for better utilization of the limited quantization levels to represent meaningful non-zero weights. This enhances the model's capacity to capture important features, leading to improved segmentation accuracy and temporal consistency. The rationale behind this observation is that by not allocating a quantization level to zero (which is already enforced by sparsity), we effectively increase the resolution of quantization for non-zero weights. This finer quantization granularity helps preserve more precise weight values, which is crucial for maintaining model performance, especially in low-bit quantization scenarios.

The results suggest that excluding zero from the quantization levels is beneficial for model performance in the context of N:M sparsity. By dedicating all quantization levels to non-zero weights, we enhance the representational capacity of the quantized weights without increasing the bit-width. This is particularly important in extremely low-bit quantization, where each quantization level represents a significant portion of the weight range.

In practice, this means that when applying quantization after N:M sparsification, it is advantageous to adjust the quantization scheme to exclude zero, thereby allocating the limited quantization levels more efficiently to the non-zero weights that contribute to the model's predictions. This strategy can be easily integrated into the quantization process and does not introduce additional computational overhead.

A.3.3 Influence of Quantization Bit-Width

We examine the effect of different quantization bit-widths on the performance of our quantized sparse models. Specifically, we evaluated 1-bit, 2-bit, and 3-bit quantization levels while using a fixed 1:4 sparsity

pattern (i.e., $N = 1$, $M = 4$). The experiments were conducted on both the MRCFA [51] and CFFM [50] models, using the MiT-B0 backbone for both student and teacher models, trained for 160k iterations.

The results are summarized in Table 4. For the **MRCFA** model, we observe a clear trend where increasing the bit-width leads to improved performance. With 1-bit quantization, the model achieves an mIoU of 27.80%, which increases to 30.18% with 2-bit quantization and further to 32.78% with 3-bit quantization. Similarly, the temporal consistency metrics, mVC_8 and mVC_{16}, improve from 85.33% and 80.28% at 1-bit to 87.22% and 82.17% at 3-bit, respectively.

The **CFFM** model exhibits a comparable pattern. The mIoU increases from 27.68% at 1-bit quantization to 32.52% at 2-bit, and reaches 33.21% at 3-bit quantization. The mVC_8 metric increases from 86.81% at 1-bit to 87.75% at 2-bit, then slightly decreases to 87.53% at 3-bit. Similarly, mVC_{16} improves from 81.75% at 1-bit to 82.99% at 2-bit, then slightly decreases to 82.79% at 3-bit.

These results indicate that increasing the quantization bit-width generally enhances model performance in terms of both accuracy and temporal consistency. The improvement is particularly significant when moving from 1-bit to 2-bit quantization, suggesting that extremely low-bit quantization (e.g., 1-bit) may not provide sufficient representational capacity for the models to perform effectively on the video semantic segmentation task.

At 3-bit quantization, the models achieve performance levels that are much closer to their full-precision counterparts, while still benefiting from significant reductions in model size and computational complexity. This suggests that 3-bit quantization offers a good balance between compression and performance for our quantized sparse models.

In practice, selecting an appropriate quantization bit-width involves balancing the trade-off between model size and performance. While lower bit-widths lead to higher compression ratios and potentially faster inference on specialized hardware, they may also introduce more quantization noise, adversely affecting model accuracy. Our findings suggest that using 3-bit quantization with 1:4 sparsity strikes an effective balance, achieving substantial model compression without significant loss in performance.

A.3.4 Effect of Distillation Weight α

We examine how varying the distillation weight α affects the performance of our quantized sparse models. The distillation weight α balances the contribution of the distillation loss from the teacher model and the primary segmentation loss during training. By adjusting α, we aim to find an optimal balance that leverages the teacher's knowledge without overshadowing the learning of the student model.

Using the MRCFA [51] model with the MiT-B0 backbone, we trained models with α values ranging from 0.01 to 0.5. All models were trained under 3-bit quantization with a $1:4$ sparsity pattern for 160k iterations. The results are summarized in Table 5.

From the table, we observe that the mIoU and temporal consistency metrics (mVC_8 and mVC_{16}) are sensitive to changes in α. When α is set to 0.01 or 0.02, the model achieves mIoU values of 32.34% and 32.38%, respectively. As α increases to 0.05 and 0.1, the mIoU slightly decreases or remains relatively stable, indicating that small variations in α within this range do not significantly impact performance.

Interestingly, the highest mIoU of 32.78% is achieved at $\alpha = 0.15$. This suggests that incorporating the teacher's knowledge with moderate emphasis enhances the student's learning. However, when α is increased beyond 0.15, we observe a decline in performance. At $\alpha = 0.2$, the mIoU drops to 32.19%, and further decreases to 30.86% at $\alpha = 0.5$.

The temporal consistency metrics follow a similar trend. The highest mVC_8 and mVC_{16} values are observed around $\alpha = 0.15$, indicating better consistency in segmentation across video frames at this distillation weight. As α increases beyond this point, both mVC_8 and mVC_{16} decrease, reflecting diminished temporal stability.

These results highlight the importance of choosing an appropriate distillation weight α. A moderate value of $\alpha = 0.15$ provides the best balance between leveraging the teacher's knowledge and allowing the student model to learn effectively from the data. Setting α too low may underutilize the benefits of distillation, while setting it too high can cause the student to over-rely on the teacher's outputs, potentially leading to overfitting or diminished generalization. In practice, we recommend tuning α within a moderate range (e.g., 0.1 to 0.2) to achieve optimal performance when applying knowledge distillation in conjunction with quantization and sparsity techniques.

A.3.5 Influence of Training Iterations

We evaluate how the number of training iterations influences the performance of our quantized sparse models. Specifically, we trained the MRCFA [51] and CFFM [50] models with the MiT-B0 backbone under 3-bit quantization with a $1:4$ sparsity pattern for 160k and 250k iterations. The results are presented in Table 6.

For the **MRCFA** model, we observed that increasing the training iterations from 160k to 250k led to a decrease in mIoU from 32.78% to 31.71%. The temporal consistency metrics mVC_8 and mVC_{16} also showed slight declines, from 87.22% to 87.10% and from 82.17% to 82.09%, respectively. This suggests that extending the training duration caused the model to overfit the training data, resulting in reduced generalization performance on the validation set.

In contrast, the **CFFM** model exhibited a slight improvement in performance with increased training iterations. The mIoU increased from 33.21% at 160k iterations to 33.35% at 250k iterations. Similarly, the temporal consistency metrics mVC_8 and mVC_{16} improved from 87.53% to 87.90% and from 82.79% to 83.18%, respectively. This indicates that the CFFM model benefits from additional training, potentially due to its architecture or learning dynamics, allowing it to further refine its feature representations without overfitting.

These observations highlight that the optimal number of training iterations can vary between models. While longer training may help some models achieve better convergence and performance, it can lead to overfitting in others. Therefore, it is crucial to monitor the validation performance during training to determine the appropriate number of iterations for each model. Adjusting the training schedule based on the model's behavior can help achieve a balance between sufficient learning and generalization.

A.3.6 Effect of Teacher Model Selection

We conducted an ablation study to examine how the selection of the teacher model influences the performance of the student model in our quantized sparse video semantic segmentation framework. Table 7 presents the results of this study across four methods: CFFM [50], MRCFA [51], Mask2Former [5], and MPVSS [56]. All models are quantized to 3 bits with a $1:4$ sparsity pattern.

For the **CFFM** method, we observe that using a teacher model with a larger backbone improves the student model's performance. When both the student and teacher use the MiT-B0 backbone, the mIoU is 33.21%. Upgrading the teacher backbone to MiT-B1 while keeping the student at MiT-B0 increases the mIoU to 34.00%, and using MiT-B2 as the teacher further maintains a high mIoU of 33.83%. This demonstrates that a more capable teacher can provide better guidance, leading to improved segmentation accuracy for the student model. When both the student and teacher are upgraded to MiT-B1, the mIoU significantly increases to 38.50% with a MiT-B1 teacher and to 39.23% with a MiT-B2 teacher. The best performance is achieved when both the student and teacher use the MiT-B2 backbone, resulting in an mIoU of 43.83%.

In contrast, for the **MRCFA** method, the benefits of using a teacher with a larger backbone are less evident. With the student backbone fixed at MiT-B0, increasing the teacher backbone from MiT-B0 to MiT-B1 or MiT-B2 actually leads to a slight decrease in mIoU from 32.78% to 31.92% and 31.63%, respectively.

This indicates that for MRCFA, the architectural compatibility between the student and teacher is crucial, and a significant mismatch may hinder effective knowledge transfer. However, when both the student and teacher use the MiT-B1 backbone, the mIoU improves to 36.90% with a MiT-B1 teacher and slightly decreases to 36.47% with a MiT-B2 teacher. The highest mIoU of 42.34% is achieved when both the student and teacher use the MiT-B2 backbone.

For the **Mask2Former** and **MPVSS** methods, similar trends are observed. Using matching backbones for the student and teacher models yields better performance. For instance, in Mask2Former, the mIoU increases from 39.29% with Swin-T backbones to 44.77% with Swin-S backbones. In MPVSS, the mIoU improves from 38.06% to 43.26% when both the student and teacher use Swin-S backbones instead of Swin-T backbones.

These findings suggest that while a more powerful teacher can enhance the student model's performance, the effectiveness of knowledge distillation is also dependent on the architectural alignment between the student and teacher models. Matching or closely related architectures appear to facilitate better feature representation learning and more efficient knowledge transfer, leading to improved segmentation accuracy and temporal consistency in quantized sparse models.

A.4 Visual Results

In this section, we present qualitative visual results to demonstrate the effectiveness of our proposed compression method. Figures are provided to illustrate the robustness and generalization capabilities of our method across different scenes and conditions. These visual examples confirm that the proposed approach effectively preserves essential features and details in the segmentation output, even under aggressive compression settings.

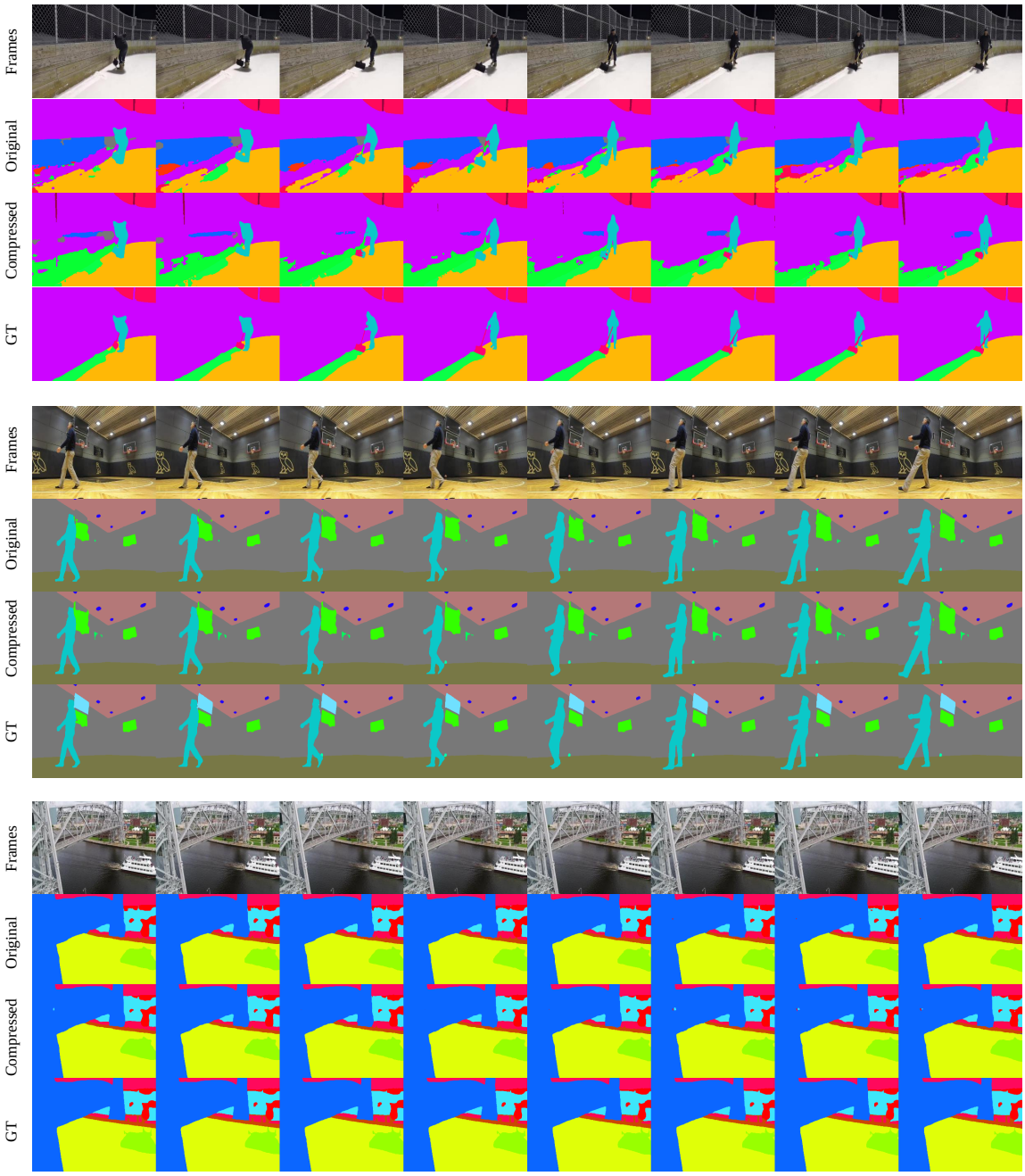

Figure A.1: Qualitative results of compressed **CFFM-B2** model using 3-bit quantization and $1:4$ sparsity.

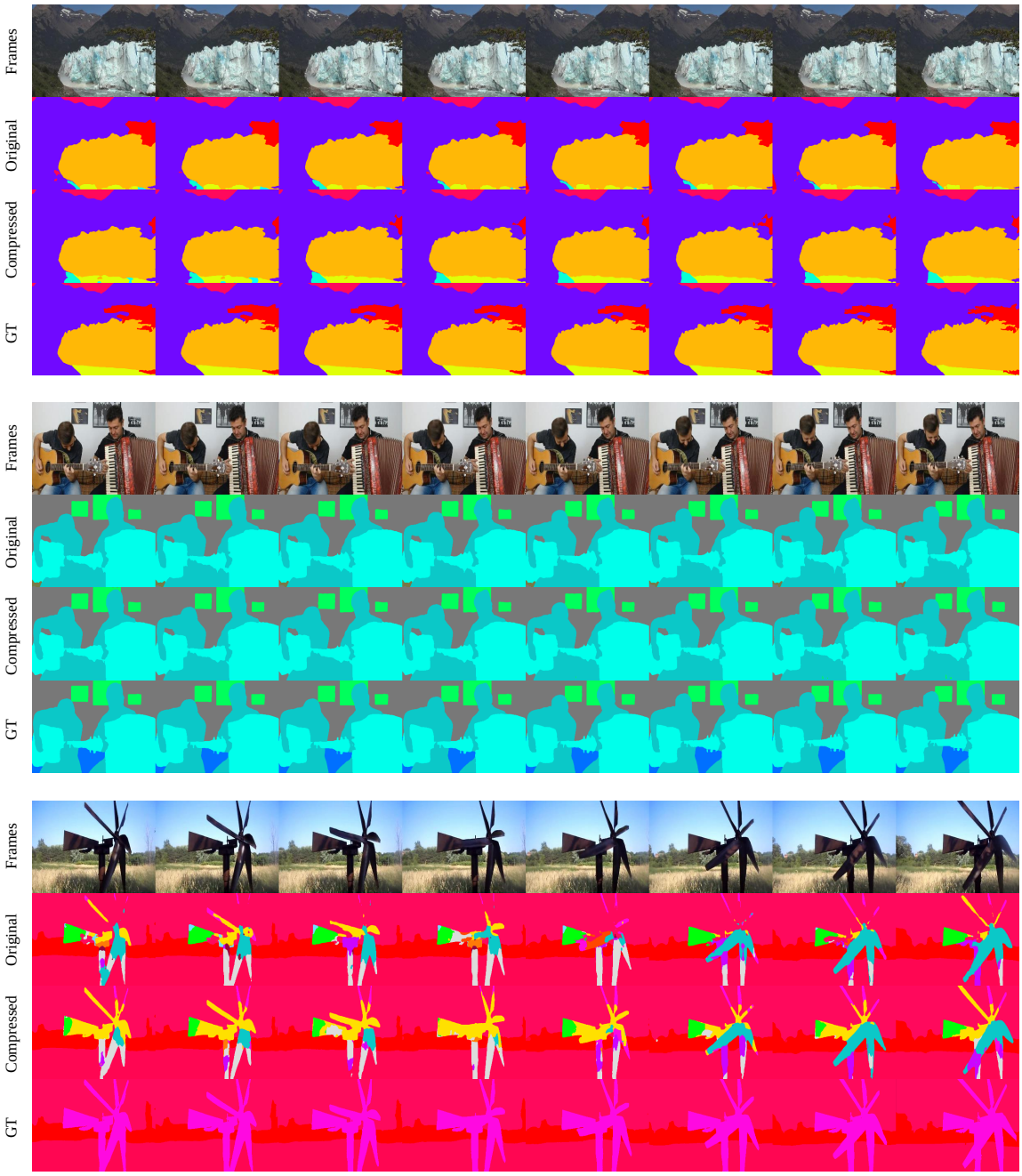

Figure A.2: Qualitative results of compressed **MRCFA-B2** model using 3-bit quantization and $1:4$ sparsity.

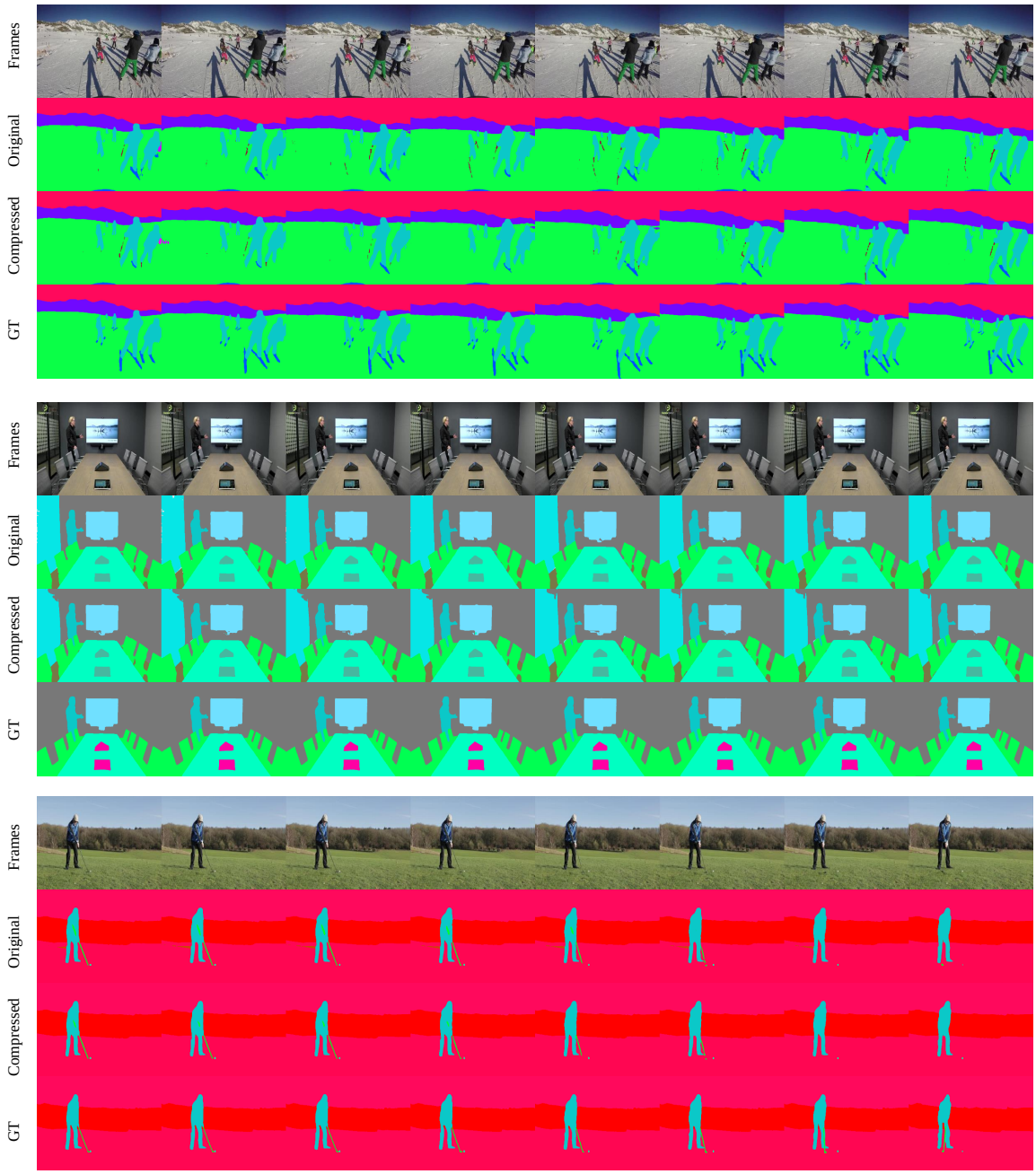

Figure A.3: Qualitative results of compressed **Mask2Former-Swin-S** model using 3-bit quantization and $1:4$ sparsity.

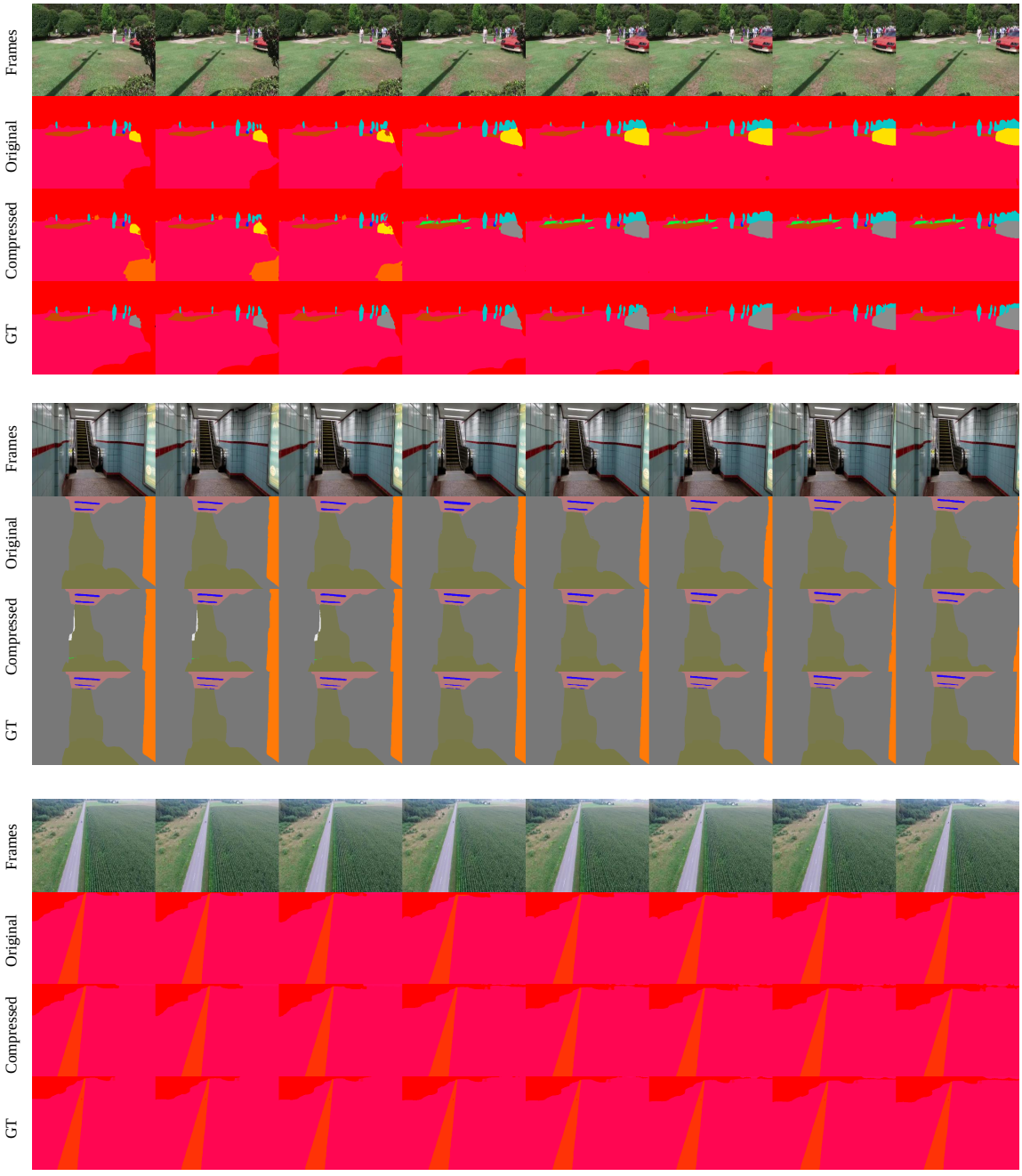

Figure A.4: Qualitative results of compressed **MPVSS-Swin-S** model using 3-bit quantization and $1:4$ sparsity.

Bibliography

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *CVPR*, pages 2209–2218, 2019.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE TPAMI*, 40(4):834–848, 2018.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[4] Wanli Chen, Xinge Zhu, Ruoqi Sun, Junjun He, Ruiyu Li, Xiaoyong Shen, and Bei Yu. Tensor low-rank reconstruction for semantic segmentation. In *ECCV*, pages 52–69, 2020.

[5] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2022.

[6] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing System (NeurIPS)*, 35:30318–30332, 2022.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing System (NeurIPS)*, 36:10088–10115, 2023.

[8] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless llm weight compression. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[9] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. In *CVPR*, pages 8885–8894, 2019.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[11] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

[12] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[13] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video CNNs through representation warping. In *ICCV*, pages 4453–4462, 2017.

[14] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[15] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[16] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[18] Chi-Wei Hsiao, Cheng Sun, Hwann-Tzong Chen, and Min Sun. Specialize and fuse: Pyramidal output representation for semantic segmentation. In *ICCV*, pages 7137–7146, 2021.

[19] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *CVPR*, pages 8818–8827, 2020.

[20] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms, 2024.

[21] Samvit Jain, Xin Wang, and Joseph E Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *CVPR*, pages 8866–8875, 2019.

[22] Xiaojie Jin, Xin Li, Huaxin Xiao, Xiaohui Shen, Zhe Lin, Jimei Yang, Yunpeng Chen, Jian Dong, Luoqi Liu, Zequn Jie, et al. Video scene parsing with predictive feature learning. In *ICCV*, pages 5580–5588, 2017.

[23] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing System (NeurIPS)*, 36:36187–36207, 2023.

[24] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 23901–23923, 2024.

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[26] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, pages 3168–3175, 2016.

[27] Shih-Po Lee, Si-Cun Chen, and Wen-Hsiao Peng. GSVNet: Guided spatially-varying convolution for fast semantic segmentation on video. In *ICME*, pages 1–6, 2021.

[28] Jiajun Li and Ahmed Louri. Adaprune: An accelerator-aware pruning technique for sustainable cnn accelerators. *IEEE Transactions on Sustainable Computing*, 7(1):47–60, 2021.

[29] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *CVPR*, pages 5997–6005, 2018.

[30] Zhikai Li and Qingyi Gu. I-vit: Integer-only quantization for efficient vision transformer inference, 2023.

[31] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[32] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration, 2022.

[33] Si Liu, Changhu Wang, Ruihe Qian, Han Yu, Renda Bao, and Yao Sun. Surveillance video parsing with single frame supervision. In *CVPR*, pages 413–421, 2017.

[34] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2604–2613, 2019.

[35] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. Efficient semantic video segmentation with per-frame inference. In *ECCV*, pages 352–368, 2020.

[36] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL)*, pages 467–484, 2024.

[37] Chengtao Lv, Hong Chen, Jinyang Guo, Yifu Ding, and Xianglong Liu. Ptq4sam: Post-training quantization for segment anything, 2024.

[38] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024.

[39] Jiaxu Miao, Yunchao Wei, Yu Wu, Chen Liang, Guangrui Li, and Yi Yang. VSPW: A large-scale dataset for video scene parsing in the wild. In *CVPR*, pages 4133–4143, 2021.

[40] Roy Miles, Mehmet Kerim Yucel, Bruno Manganelli, and Albert Saà-Garriga. Mobilevos: Real-time video object segmentation contrastive learning meets knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10480–10490, 2023.

[41] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[42] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *CVPR*, pages 6819–6828, 2018.

[43] Matthieu Paul, Martin Danelljan, Luc Van Gool, and Radu Timofte. Local memory attention for fast video semantic segmentation. In *IROS*, pages 1102–1109. IEEE, 2021.

[44] Matthieu Paul, Christoph Mayer, Luc Van Gool, and Radu Timofte. Efficient video semantic segmentation with labels propagation and refinement. In *WACV*, pages 2873–2882, 2020.

[45] Jeff Pool and Chong Yu. Channel permutations for n: m sparsity. *Advances in neural information processing systems*, 34:13316–13327, 2021.

[46] Soroush Seifi and Tinne Tuytelaars. Attend and segment: Attention guided active semantic segmentation. In *ECCV*, pages 305–321, 2020.

[47] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[48] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE TPAMI*, 39(4):640–651, 2017.

[49] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *ECCV*, pages 852–868, 2016.

[50] Guolei Sun, Yun Liu, Henghui Ding, Thomas Probst, and Luc Van Gool. Coarse-to-fine feature mining for video semantic segmentation. In *CVPR*, pages 3126–3137, 2022.

[51] Guolei Sun, Yun Liu, Hao Tang, Ajad Chhatkuli, Le Zhang, and Luc Van Gool. Mining relations among cross-frame affinities for video semantic segmentation, 2022.

[52] Guolei Sun, Wenguan Wang, Jifeng Dai, and Luc Van Gool. Mining cross-image semantics for weakly supervised semantic segmentation. In *ECCV*, pages 347–365. Springer, 2020.

[53] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[54] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 48630–48656, 2024.

[55] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models, 2023.

[56] Yuetian Weng, Mingfei Han, Haoyu He, Mingjie Li, Lina Yao, Xiaojun Chang, and Bohan Zhuang. Mask propagation for efficient video semantic segmentation, 2023.

[57] Meng-Chieh Wu, Ching-Te Chiu, and Kun-Hsuan Wu. Multi-teacher knowledge distillation for compressed video action recognition on deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2202–2206. IEEE, 2019.

[58] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 38087–38099, 2023.

[59] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018.

[60] Zeyu Xiao, Xueyang Fu, Jie Huang, Zhen Cheng, and Zhiwei Xiong. Space-time distillation for video super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2113–2122, 2021.

[61] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.

[62] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *CVPR*, pages 6556–6565, 2018.

[63] Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. Onebit: Towards extremely low-bit large language models, 2024.

[64] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. DenseASPP for semantic segmentation in street scenes. In *CVPR*, pages 3684–3692, 2018.

[65] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, pages 173–190, 2020.

[66] Fan Zhang, Yanqin Chen, Zhihang Li, Zhibin Hong, Jingtuo Liu, Feifei Ma, Junyu Han, and Errui Ding. ACFNet: Attentional class feature network for semantic segmentation. In *ICCV*, pages 6798–6807, 2019.

[67] Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[68] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[69] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch, 2021.

[70] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, pages 8856–8865, 2019.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

⊗ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Quantized Sparse Models for Video Semantic Segmentation

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Chang	Ziqing

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
15.12.2024	常子清 Ziqing Chang

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard