VTSeg: Video Transformer for Semantic Segmentation

Vasile Lup
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
lupvasi97@gmail.com

Ion Giosan
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
ion.giosan@cs.utcluj.ro

***Abstract*—Semantic segmentation is a critical task for any system that requires scene understanding, as it computes the objects' class at pixel level. We present VTSeg, a novel video semantic segmentation framework that exploits spatio-temporal information between consecutive frames to achieve a higher segmentation quality. VTSeg leverages semi-supervised learning, eliminating the need for additional training data. The architecture employs a unique adaptation of Transformers to video semantic segmentation by making an analogy between image representation spaces and natural languages. The encoder and decoder incorporate an innovative video attention module and generate multiscale features. Additionally, we introduce a novel segmentation head that uses super-resolution techniques. Compared to other video semantic segmentation approaches that include optical flow, we achieve a 3% mIoU improvement on the Virtual KITTI 2 dataset, even when ground truth is used directly for optical flow. Compared to image segmentation networks, VTSeg keeps the same 3% mIoU distance despite using 75% fewer labels. On the Cityscapes validation set, we obtain 76% mIoU using half the input resolution, without including any of the usual quality enhancing techniques such as ImageNet pre-training, test set augmentation, or sliding window inference. VTSeg framework learns powerful representations from both labeled and unlabeled data, making it a suitable video adapter able to boost the results of any image-based semantic segmentation network.**

***Index Terms*—Semantic segmentation, Video semantic segmentation, Transformer, Super-Resolution, Deep Learning.**

I. Introduction

Images are the main modality through which we, as humans, perceive and interpret the world around us. The fast advancement of technology has enabled the inclusion of digital cameras in a plethora of devices, enabling them to noninvasively visualize the physical world.

Semantic segmentation, the task of labeling each pixel in an image with the corresponding class, is a foundational component for many systems seeking scene understanding. The applications span across areas such as autonomous driving, robotics, and medical imaging [1], [2]. Semantic segmentation is a challenging problem due to object scale variability, occlusions, unique object shapes, and intraclass variations. Moreover, the high cost of pixel-level ground truth annotations imposes severe constraints on deep architectures that require a vast training dataset.

As the Transformer architecture has raised the bar in natural language processing [3], there is a new focus on adapting this approach to image processing. Vision Transformer (ViT) [4] is introduced by Dosovitskiy et al. for image classification and used as a backbone for many works that employ Transformer architecture for semantic segmentation [5]–[7].

In practice, most image-processing systems work predominantly with video sequences, characterized by spatio-temporal relationships between frames. While we could use any image semantic segmentation architecture and segment the video frame-by-frame, all the invaluable information contained in the relationship between consecutive images would be lost.

There is an ongoing effort to develop architectures for video semantic segmentation. The redundant information between video frames can be used for increasing inference speed [8] or for attaining higher segmentation quality [9], [10].

We want to take a step back and address video semantic segmentation from a different angle. From a user's perspective, semantic segmentation networks simply transform a real-world image into another image that labels the real world. In the new image, all the colors have an associated semantic class. Essentially, RGB (Red Green Blue) image representation "language" is translated to another "language" that has semantics associated with each color. By making an analogy between words and images, phrases and video sequences, video semantic segmentation becomes a problem of translation between RGB and semantic spaces/languages.

Since the Transformer architecture has impressive results for language translation, we want to adapt it for video semantic segmentation while maintaining two key principles: the encoder receives input in the source language, while the decoder has as input the phrase in the destination language. This setup stands in contrast to existing works that incorporate Transformer architectures for video semantic segmentation. These prior approaches often only adopt certain components of the architecture, failing to maintain the integral encoder-decoder pair of Transformers.

Our proposed VTSeg framework introduces a novel approach to Transformer-based video semantic segmentation. It uses spatio-temporal information between consecutive frames to boost segmentation quality. VTSeg harnesses the power of semi-supervised learning, bypassing the need for extensive

labeled data, thereby reducing the overall cost and time of manual annotation.

As mentioned above, the input for VTSeg decoder needs to be in the destination language, which in our case is the semantic space. To obtain this input, we use a semantic segmentation network that works on images, also called *static semantic segmentation network*. Therefore, VTSeg can be viewed as a framework that improves the segmentation of any static network.

We base VTSeg's encoder and decoder on the encoder used in Segformer [7], by extending the modules to work in the time domain. Also, we introduce a new video cross-attention module in VTSeg's decoder.

VTSeg is tested using two datasets: Cityscapes [11], which contains images from real life, and Virtual KITTI 2 [12] composed of densely labeled video sequences. We show that VTSeg obtains better segmentation quality than the architectures it is based on, Segformer and the static segmentation network respectively. Even when VTSeg is using 75% fewer labels in training, it still manages to surpass the static segmentation network. Moreover, we demonstrate that VTSeg is superior to another video segmentation approach that uses the same static segmentation network but also includes optical flow. In the end, we provide a study on the effect of video attention span and segmentation results.

II. Related work

A. Image Semantic Segmentation

Traditional techniques for semantic segmentation include thresholding, clustering, Support Vector Machines (SVM) or decision trees [13]. After convolutional neural networks (CNN) were introduced in the LeNet architecture [14], they were adapted for semantic segmentation by Long et al. [15] who proposed a fully convolutional network (FCN).

After the introduction of FCN, a series of successive improvements followed. Many architectures use encoder-decoder model, such as U-Net [2]. PSPNet [16] and DeeplabV2 [17] take a step further and employ dilated convolutions or pyramid pooling. Kayming et al. [18] ease gradient propagation in deep networks with a novel block that uses residual connections. DeeplabV3+ [19] uses Atrous Spatial Pyramid Pooling, while Wang et al. [20] achieve state-of-the-art performance by leveraging deformable convolutions. Other architectures, such as ERFNet [21] or SegNet [22] are focused on reducing the inference time.

Chen et al. [5] develop a dense prediction adapter for ViT [4]. Other adaptations of Transformer architecture for semantic segmentation are presented in the following: Segmenter [23], SETR [6], Swin [24] and Twins [25]. Segformer [7] uses a hierarchical Transformer encoder.

B. Video Semantic Segmentation

We split architectures for video semantic segmentation into two categories: those including optical flow and those that do not incorporate it. The proposed network, VTSeg, does not use optical flow.

In the optical flow inclusion category we note NetWarp [26], which translates the internal representation across temporal dimension. In [9] a recurrent module for video semantic segmentation improvement is proposed, being further refined and studied on video datasets in [10].

In [8] is presented an architecture that does not use optical flow and updates the network's layers using different clock speeds. TDNet [27] is a time-distributed network for fast video segmentation that employs an attention propagation module (APM). TMANet [28] uses self-attention to aggregate the relations between consecutive video frames.

C. Image-to-image translation

Image-to-image translation networks specialize in translating an image from one domain to another. They are generally used for the operation inverse to semantic segmentation, i.e. for transitioning from an image's label to a realistic image. Even if the task is different, we mention them because they have the same principle as our approach: an image is translated into another image.

An example is [29], which includes conditioned adversarial networks that, in addition to the mapping between images, also learn a cost function. The authors test the method for semantic segmentation, but the results are inferior to regular segmentation approaches.

III. Approach

A. Core idea

In image semantic segmentation we begin with images in RGB space and generate predictions with the number of channels equal to the number of object classes C, where each channel contains the pixel probability of belonging to the respective class. The final class of a pixel is given by the channel with the highest probability for the respective position.

Starting from the RGB space with 3 channels, we end up in another space with C channels, which we will refer to as the *Semantic* space. If we have a video sequence, starting from a sequence of frames represented in the RGB space, we will generate a sequence of frames in the *Semantic* space.

Considering the images as the equivalents of words and the representation space as the equivalent of a language, a video sequence implicitly becomes a sentence in a language. Our task is to translate a sentence from the RGB "language" into another sentence from the *Semantic* "language". Thus, we have equated the problem of semantic segmentation with a translation problem. This is the core idea of the paper.

As the Transformer is currently the state-of-the-art translation architecture, we decided to adapt this approach to video semantic segmentation. Of course, there will be differences when compared to the architecture presented in [3], but the basic ideas remain unchanged.

B. VTSeg overview

We named the proposed architecture **VTSeg**: Video Transformer for Semantic Segmentation. Figure 1 presents the main modules. The network's input and output are video

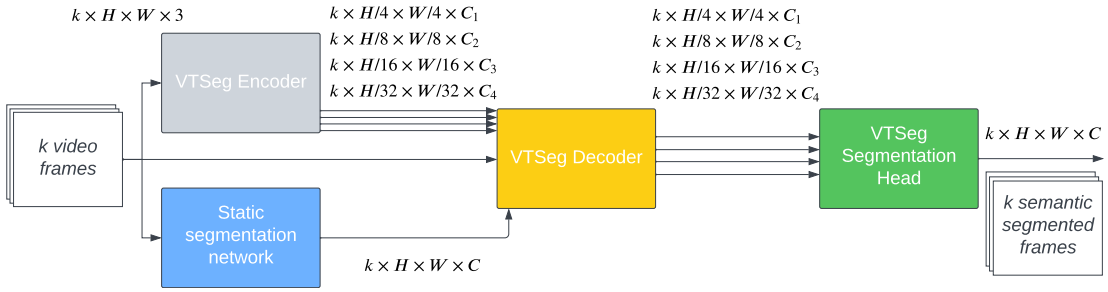

Fig. 1. Video Transformer for Semantic Segmentation (**VTSeg**) framework. It consists of four modules: hierarchical encoder and decoder with video attention, a segmentation head with pixel shuffle upsampling, and a static segmentation network. C represents the number of semantic classes, while $C_{1..4}$ are network hyperparameters.

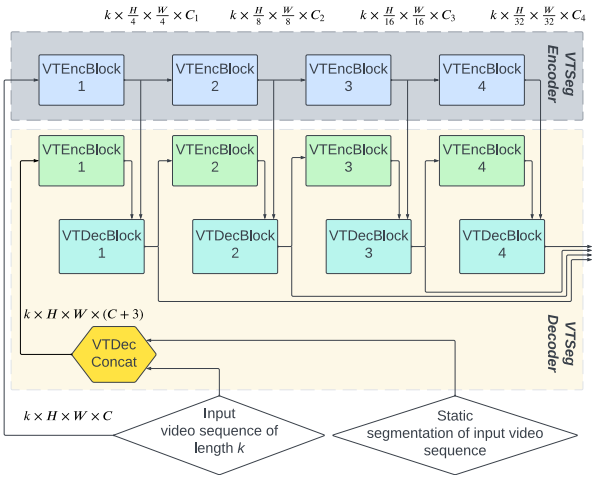

Fig. 2. VTSeg Encoder and Decoder. The main blocks are VTEncBlock and VTDecBlock, which use self-video attention and cross-video attention respectively. Above the arrows we have the feature maps' sizes for a video sequence of length k and with C semantic classes.

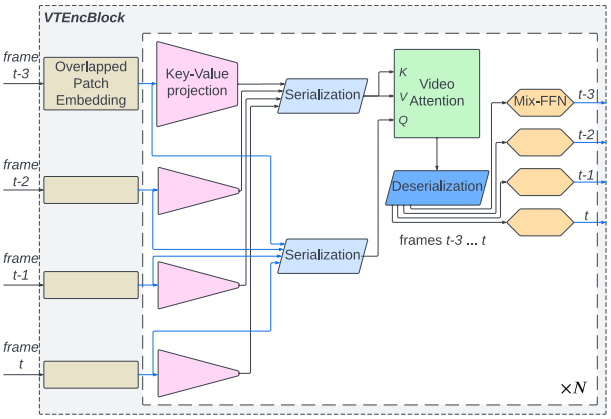

Fig. 3. VTEncBlock (Video Transformer Encoder Block) for video sequences with length $k = 4$. It uses efficient self-video attention.

sequences of length k, with the latter containing semantic segmentation of the frames. Both the encoder and decoder generate hierarchical feature maps for each frame in the input video sequence. This allows capturing of fine and more general details of the frames. The VTSeg segmentation head uses the pixel shuffle technique [30] to scale the semantic segmentation to the original resolution of the frames. The static semantic segmentation network can be replaced with any existing semantic segmentation network. Thus, VTSeg becomes a generic framework that can be used to improve semantic segmentation on video sequences of any semantic segmentation network that works just with images.

In the following, we present the role of each module and establish the relationship with Transformer [3] architecture. For simplicity, we did not include normalization and dropout layers.

C. VTSeg Encoder

This module plays the role of the encoder in the Transformer architecture, by using self-video attention layers. It receives as input the video sequence and will generate hierarchical feature maps for all the frames.

VTSeg Encoder is presented in figure 2, having gray background. It's composed of multiple blocks of type *VTEncBlock*, each one generating feature maps at a successively lower resolution. $C_1 \dots C_4$ are hyperparameters for the depth of each VTEncBlock, set to [32, 64, 160, 256] in experiments.

This encoder is based on the Segformer's encoder [7]. The main difference is that in the case of Segformer, the encoder works on individual images, while the VTSeg encoder represents an extension able to process video sequences by applying self-attention across time.

VTEncBlock (Video Transformer Encoder Block) is presented in figure 3 considering video sequences of length 4. The part with white background is cloned N times, by using outputs of Mix-FFN modules as the inputs for the next clone. We set $N = 2$ for the experiments. Overlapped Patch Embedding modules are used just for the first clone. The main components are the following:

- *Overlapped Patch Embedding*. Following Segformer's approach [7], we split each frame into overlapping patches and then extract embeddings that preserve local continuity between neighboring patches. This is achieved by a convolutional layer with kernel size = patch size = K, stride S, padding P, and C_i output feature maps. We use $(K = 7, S = 4, P = 3)$ for the first VTEncBlock and $(K = 3, S = 2, P = 1)$ for the rest.

- *Key-value projection*. In order to make the video attention module efficient, we reduce the size of key-value pairs with a factor of R in spatial dimension, obtaining a computational complexity for attention of $O(H^2 \cdot W^2 / R^2)$ instead of $O(H^2 \cdot W^2)$. Details can be consulted in [7]. The 4 VTEncBlocks use values [8, 4, 2, 1].
- *Serialization*. For video sequences, our aim is to distribute attention across patches from all frames simultaneously. Serialization of all the frame patches allows us to utilize fragments from frames originating at different time steps. This enables us to leverage important information for segmentation across time.
- *Deserialization*. This is the inverse operation of the previous module. After applying the attention mechanism, we reconstruct the separate feature maps for each frame of the video sequence by taking the fragments in the order of their serialization.
- *Video Attention*. This is a classic attention module (ecuation 1). For queries (Q), we directly use the fragments from the embedding, while for key-value pairs (K, V), we use their reduced projections.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{head}}}\right)V \quad (1)$$

- *Mix-FFN*. This module is introduced in [7] to replace the positional encoding module from ViT [4]. Mix-FFN learns positional information from the input data. For our use case, this module is a perfect fit since the learned positional information is not just spatial, but also temporal. The module is described by equation 2, where MLP stands for Multi-Layer Perceptron.

$$x_{out} = MLP(GELU(Conv_{3\times3}(MLP(x_{in})))) + x_{in} \quad (2)$$

D. VTSeg Decoder

This is the equivalent of the decoder in Transformer architecture [3]. It receives the encoder's output, static segmentation of the video frames, and the original video input. With self- and cross-video attention modules it will generate hierarchical output feature maps for each frame.

VTSeg Decoder is presented in figure 2, using light yellow background. It is composed of 3 types of blocks: *VTEncBlock* (section III-C), *VTDecBlock*, and *VTDec Concat*. Regarding the data flow, we have triplets ($VTEncBlock_i$ (from Encoder), $VTEncBlock_i$ (from Decoder), $VTDecBlock_i$), $i = 1 \dots 4$. In the decoder, $VTEncBlock_i$ takes input from $VTDecBlock_{i-1}$ for $i \geq 2$ and from VTDec Concat for $i = 1$. $VTDecBlock_i$ has two inputs, one for Encoder's and one for Decoder's $VTEncBlock_i$. For the same value of i, all blocks in the triplet will have the same dimensions for the input and output feature maps. The only exception with different dimensions is the input to the first VTEncBlock in both the encoder and decoder. In experiments, VTSeg Decoder uses the same hyperparameters as VTSeg Encoder.

VTDecBlock (Video Transformer Decoder Block) is presented in figure 4 considering video sequences of length 4. The

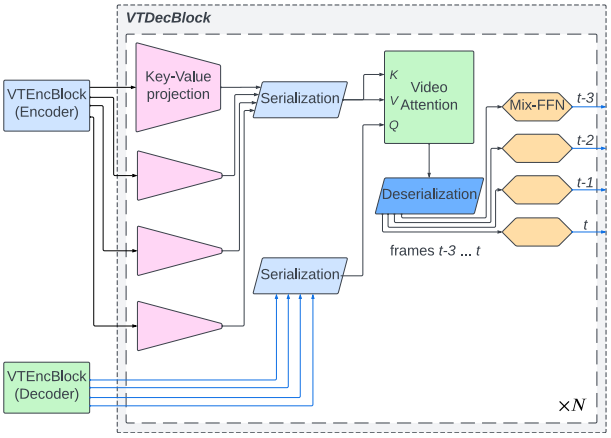

Fig. 4. VTDecBlock (Video Transformer Decoder Block) for video sequences with length $k = 4$. It uses efficient cross-video attention.

part with white background is cloned N times, by using outputs of Mix-FFN modules instead of VTEncBlock(Decoder) inputs. We use $N = 2$ in experiments. It is very similar to VTEncBlock, but has two key differences:

- Overlapped Patch Embedding is missing since it is included in the VTEncBlocks used as input.
- Cross-Video Attention is used. The outputs of corresponding $VTEncBlock_i$ from VTSeg Encoder are projected, serialized, and then used for K, V input of the video attention module. The outputs of $VTEncBlock_i$ from VTSeg Decoder are serialized and used for Q input of the video attention module.

VTDec Concat. In Transformer's [3] training the decoder receives as input the correctly translated text, i.e. the ground truth. In the case of VTSeg, the training datasets do not have the label available for each frame of the video sequence, so we use the static semantic segmentation network as an alternative. The static segmentation network generates an initial segmentation which will be further improved by VTSeg, but it could be far from ground truth. Therefore, to include appropriate queries in the decoder, we add the original frames as input. More specifically, we concatenate the 3 RGB channels of the original frames to the C segmentation channels.

The outputs of the static segmentation network are unscaled values while RGB inputs are in the $[0, 1]$ range. To eliminate scaling discrepancies between channels we normalize the segmentation with LayerNorm and apply GELU activation function before RGB channel concatenation. Layer normalization preserves the semantics of the segmentation, as it maintains the relative order of values across channels. GELU prevents large negative values, and the final segmentation values will be approximately in the same value range as the RGB input.

E. VTSeg Segmentation Head

Normally, this module would be part of the VTSeg Decoder, but in order to reduce the decoder's complexity we decided to keep it separate. The semantic segmentation head takes hierarchical feature maps generated by the decoder and generates

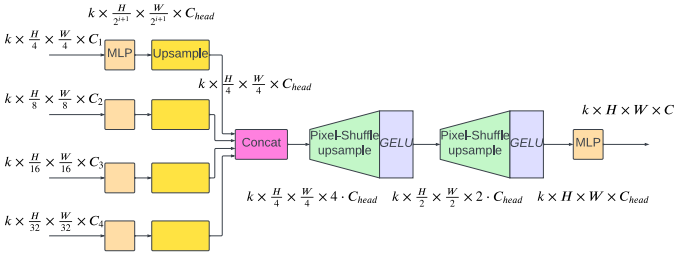

Fig. 5. VTSeg Segmentation Head. Based on the VTSeg Decoder's hierarchical output it generates the semantic segmentation at full resolution for each of the k video frames. C is the number of semantic classes, while C_{head} is a hyperparameter.

the semantic segmentation of each of the k input frames, using their original resolution.

VTSeg semantic segmentation head is presented in figure 5. It updates the lightweight decoder from [7] with super-resolution techniques. The frames at each resolution are processed independently and then concatenated. Next, the spatial resolution is increased using upsampling based on Pixel-Shuffle modules [30]. In the end, the output is adapted to C semantic classes. We use $C_{head} = 256$ in experiments.

F. Static Semantic Segmentation Network

In the Transformer architecture, the input to the decoder needs to be in the target translation language. Following the analogy from subsection III-A, the input for the VTSeg decoder should be in the semantic space. To obtain initial segmentation we use a separate image semantic segmentation architecture.

We choose Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation (ERFNet) [21] as the static segmentation network since it provides a good balance between speed and segmentation quality. ERFNet uses factorized convolutions in the residual blocks that have the same receptive field but require less computations than their non-factorized counterpart. Also, it includes dilated convolutions to capture multiscale context information without increasing the computational cost.

IV. Experiments

A. Datasets

Cityscapes [11] is a driving semantic segmentation dataset captured in the real world. Images have a resolution of 2048×1024 and 19 semantic classes. The dataset contains 5000 labeled images, which are split as follows: 2975 images for training, 500 for validation, and 1525 for testing. Ground truth for the test set is not publicly available.

The labeled images do not form a contiguous video sequence, but for each labeled image the 19 preceding frames and 10 subsequent frames are available. Therefore, Cityscapes contains video sequences in which just one frame is labeled.

Virtual KITTI 2 [12] is a synthetic dataset for autonomous driving that can be used for semantic segmentation, object detection, or optical flow tasks. It aims to replicate the real-world KITTI dataset [31] in a virtual space and consists of 5 video sequences, each having 10 variations. In this project, we include *clone, overcast, fog, 15-deg-left, sunset*, and *15-deg-right* variations from the left camera. The images have a resolution of 1242×375, and the semantic segmentation ground truth contains 15 classes (including *undefined* class).

Virtual KITTI provides labels for each frame in the video sequences. However, in the training set, we will only use the label of each 4th frame to simulate the scenario of real-world datasets with costly ground truth generation. Evaluation will use the labels for all the frames. We use *70/20/10* data split for the training/validation/testing sets, following a technique [10] that prevents information leakage across splits.

The main advantage of Virtual KITTI is that it allows us to evaluate the accuracy of the model on a video dataset with dense labels for each frame.

B. Training

We will use video sequences of length $k = 4$ which provide ground truth only for the last frame. VTSeg will use semi-supervised learning to leverage information from both labeled and unlabeled frames. Cityscapes frames are scaled to a resolution of 1024×512, while in the Virtual KITTI case we use a resolution of 640×192.

We use an AdamW optimizer with a two-phase OneCycleLR [32] learning rate scheduler, weight decay 1e-2, and Cross-Entropy loss. The scheduler uses a maximum learning rate of 6e-5. Batch size is 2, i.e. 2 sequences of 4 frames each. Data augmentation includes random flipping, scaling, cropping, rotation, blurring, and modifications of brightness, saturation, and hue. We do not adopt widely used tricks such as OHEM, auxiliary losses, or class balance loss. Moreover, we do not use pre-training on other datasets and do not train the encoder separately.

Our training has three stages:

1) Training of the static segmentation network, ERFNet. We skipped this stage as we used the weights made available by authors of [21] for Cityscapes and [10] for Virtual KITTI. Important note: since we use a different input interpolation, ERFNet reported results might differ slightly from the original work.
2) Training of VTSeg with frozen static segmentation network for 100 epochs (one epoch = one pass through the whole dataset).
3) End-to-end training of VTSeg and the static segmentation network for 200 epochs.

In each epoch, in order to train VTSeg using all generated outputs, for 80% of the sequences we will derive a new video sequence using random augmentation of the labeled frame. By scaling and cropping we can crudely simulate camera motion in space and generate densely labeled video sequences. The remaining 20% sequences will discard the segmentation of the first three frames, as no ground truth is available.

C. Evaluation

VTSeg is trained exclusively on the training set. Then, we report the performance on the validation or test sets. The best-

TABLE I
PER-CLASS AND MEAN IOU ON CITYSCAPES VALIDATION SET. ERFNET AND SEGFORMER-B0 ARE STATIC SEMANTIC SEGMENTATION NETWORKS, WHILE GRU(VCN) IS AN OPTICAL FLOW VIDEO SEMANTIC SEGMENTATION ARCHITECTURE.

Class	ERFNet	Segformer-B0	GRU(VCN)	VTSeg
road	97.61	97.82	97.73	98.17
sidewalk	80.93	82.16	81.58	84.77
building	90.63	90.84	90.86	91.88
wall	46.69	55.52	49.31	58.30
fence	55.78	51.19	55.20	59.67
pole	60.16	54.18	60.07	61.85
traffic-light	61.76	61.08	63.89	65.01
traffic-sign	71.89	70.53	72.93	75.19
vegetation	91.25	91.32	91.46	91.42
terrain	60.02	61.63	61.45	61.61
sky	93.39	93.97	93.77	94.37
person	76.36	74.82	76.91	78.29
rider	53.88	52.26	53.91	58.96
car	92.85	92.63	93.13	94.26
truck	71.56	67.76	73.27	79.91
bus	77.65	76.49	79.36	85.13
train	61.54	65.02	63.77	77.09
motorcycle	46.92	55.93	46.05	55.45
bicycle	71.10	70.78	72.45	72.71
mIoU	71.68	71.89	72.48	**76.00**

performing model on the validation set is run on the test set only once.

In evaluation we use the same input resolution as in training, but we upscale the output to the official dataset's resolution using bicubic interpolation before computing the results. We report semantic segmentation performance using mean Intersection over Union (mIoU). We do not use sliding window inference or test set augmentation.

For a fair comparison, we included architectures that use the same training and inference resolution as VTSeg and do not use the performance-enhancing tricks mentioned above. We compare VTSeg with the following:

- ERFNet [21]. This is the network used for static segmentation. The provided weights for Cityscapes include Imagenet [33] pretraining.
- Segformer-B0 [7]. Segformer family includes many variants, but we use the same hyperparameters as this model. Also, this is the only variant trained on the same resolution as VTSeg. Segformer-B0 also has Imagenet pretraining.
- GRU(VCN) and GRU(GT) [10]. These are video semantic segmentation networks that improve segmentation results using optical flow. They include ERFNet as a component, so we can compare VTSeg with an optical flow framework that uses the same static segmentation network. GRU(VCN) uses VCN [34] as the optical flow source, while GRU(GT) directly uses Virtual KITTI ground truth optical flow.

D. Cityscapes results

Table I presents the IoU values for VTSeg and the three architectures mentioned above on the Cityscapes validation set. VTSeg and GRU use unlabeled frames from the video sequence, while ERFNet and Segformer-B0 are image segmentation networks that only use labeled frames.

VTSeg outperforms the image-based segmentation networks by more than 4% mIoU. Specifically, VTSeg shows significant improvements in classes such as train (+12%), bus (+8%), truck (+8%), and rider (+5%). Therefore, VTSeg enhances the accuracy for dynamic and underrepresented classes by leveraging unlabeled frames.

VTSeg exhibits superior performance (+3.5% mIoU) over the GRU video semantic segmentation with optical flow. The latter requires separate training on a different dataset for the optical flow network, while VTSeg learns the temporal relationships between frames directly through semi-supervised learning.

TABLE II
PER-CLASS AND MEAN IOU ON VIRTUAL KITTI TEST SET. GRU(GT) USES GROUND TRUTH OPTICAL FLOW. IN TRAINING JUST 25% LABELS ARE USED, WHILE EVALUATION IS DONE WITH ALL AVAILABLE LABELS.

Class	ERFNet	GRU(VCN)	VTSeg
terrain	95.50	95.47	96.44
sky	94.19	94.22	95.44
tree	94.26	94.28	95.31
vegetation	94.56	94.57	96.32
building	94.94	95.09	96.08
road	98.34	98.34	98.77
guardrail	92.36	92.57	94.87
traffic-sign	90.02	90.94	94.16
traffic-light	82.12	83.56	87.02
pole	70.36	70.20	77.34
misc	65.14	66.54	73.19
truck	89.33	90.59	93.77
car	93.82	94.38	95.53
van	60.13	65.03	70.05
mIoU	86.79	87.56	**90.31**

TABLE III
PER-CLASS AND MEAN IOU ON VIRTUAL KITTI VALIDATION SET. ERFNET(4), GRU(4), VTSEG(4) USE 25% LABELS IN TRAINING (EVERY 4TH LABEL), WHILE ERFNET(1) USES ALL THE AVAILABLE LABELS.

Class	ERFNet (4)	ERFNet (1)	GRU (4, VCN)	GRU (4, GT)	VTSeg (4)
terrain	95.75	95.98	95.69	95.77	96.33
sky	93.76	94.22	93.75	93.79	94.74
tree	93.98	94.32	94.01	94.04	95.04
vegetation	91.24	92.40	90.93	91.13	93.43
building	87.93	88.67	88.47	88.40	90.81
road	98.25	98.38	98.24	98.28	98.70
guardrail	86.66	86.87	86.86	86.86	88.98
traffic-sign	71.71	66.69	76.17	76.33	84.82
traffic-light	75.33	79.88	79.54	80.34	82.24
pole	52.17	54.59	52.79	53.41	61.69
misc	52.57	63.90	54.76	55.34	62.74
truck	51.94	62.11	67.76	70.00	77.22
car	94.89	95.60	94.88	94.98	94.86
van	85.64	88.60	88.42	88.82	88.11
mIoU	80.84	83.02	83.02	83.39	**86.41**

E. Virtual KITTI results

Table II presents the IoU values on the test set of Virtual KITTI. Compared to ERFNet, VTSeg achieves a mIoU increase of over 3.5%. At individual class level, the largest

TABLE IV
MEAN IOU ON VALIDATION SETS WHEN THE VIDEO SEQUENCE LENGTH IS VARIED. k_{inf} DENOTES THE NUMBER OF FRAMES USED IN INFERENCE.

Dataset	**VTSeg** ($k_{inf} = 1$)	**VTSeg** ($k_{inf} = 2$)	**VTSeg** ($k_{inf} = 3$)	**VTSeg** ($k_{inf} = 4$)
Virt. KITTI	86.32	86.29	86.40	**86.41**
Cityscapes	75.37	75.53	75.82	**76.00**

improvements are observed for van (+10%), misc (+8%), pole (+7%), traffic-light (+5%), and traffic-sign (+4%). Once again, we observe that the improvements occur for classes that are underrepresented, with the dynamic class "van" showing the largest increase.

In comparison to GRU(VCN), VTSeg achieves an improvement of 2.75% mIoU. At class level, the underrepresented classes, such as pole, misc (+7%), and van (+5%) have the biggest IoU boost.

Table III presents the IoU values on the validation set of Virtual KITTI 2. VTSeg, GRU and ERFNet(4) use 25% of the labels for training, while ERFNet(1) uses all the labels.

When the same number of labels is used, VTSeg achieves a mIoU increase of over 5.5% compared to the static segmentation network. Compared to GRU(VCN) and GRU(GT), the improvements are 3.4% and 3%. Even when GRU directly uses the ground truth optical flow from Virtual KITTI, it is outperformed by VTSeg. The gains are again observed in the case of underrepresented classes, such as traffic-sign and pole.

Another interesting comparison is with ERFNet(1), which uses all the ground truth during training. VTSeg achieves a 3.4% increase in mIoU, even though it is trained with 75% fewer labels.

F. Effects of video sequence length

We use the networks previously trained on sequences of length 4, but we vary the length of video sequences between 1 and 4 in evaluation. Table IV presents the results. On Virtual KITTI dataset the difference between using 4 frames (VTSeg(4)) and 3 frames (VTSeg(3)) is minimal, but when using a single frame, the mIoU decreases by 0.1%. Virtual KITTI is not as complex as Cityscapes and has few dynamic classes, therefore previous frames might not contain information that is not present in the current frame. On Cityscapes, as expected, the mIoU values decrease proportionally with the number of frames used. Sequences with a single frame obtain a mIoU that is over 0.6% lower than 4-frame sequences. Since the difference between using 3 and 4 frames is 0.2% mIoU, VTSeg might benefit from using even larger video sequences.

G. Qualitative results

Figure 6 shows the semantic segmentation results on sample 2-frame video sequences. On the Virtual KITTI example (left), ERFNet has the most errors for the white van and the overhead traffic sign. The video segmentation network GRU(VCN) resolves the errors for the van but still segments the traffic sign incorrectly. In contrast, VTSeg eliminates the ERFNet errors and generates a semantic segmentation very close to the ground truth. On Cityscapes (right) we observe that ERFNet and Segformer-B0 ignore the pole of the parking sign (orange rectangle), while VTSeg correctly segments it, even though it is barely visible. The white rectangle shows how VTSeg correctly segments both the bicycle and the cyclist, unlike the other architectures.

V. CONCLUSION

In this paper, we present the VTSeg framework, a powerful Transformer architecture inspired by a novel correlation between video semantic segmentation and natural language translation. VTSeg innovatively exploits spatio-temporal information between consecutive frames with no additional training data, yielding improved segmentation quality. The framework introduces a new video attention module in the multiscale encoder and decoder. Moreover, we develop a segmentation head that leverages super-resolution techniques.

We test VTSeg on Cityscapes and Virtual KITTI datasets, where it easily surpasses image and video segmentation networks trained in similar conditions. VTSeg is superior even when ground truth optical flow is used directly by other approaches, or when it uses 75% less training labels than image segmentation architectures.

The semi-supervised training of VTSeg successfully exploits both labeled and unlabeled data, indicating its potential as a versatile video adapter that can significantly enhance the results of any image-based semantic segmentation network. Future work may explore the further adaptability of this model to other video-based tasks, potentially expanding our understanding and utilization of Transformer architecture in video processing.

REFERENCES

[1] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 10 2017, pp. 1–8.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, p. 234–241, 2015.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *9th International Conference on Learning Representations, {ICLR}*, 2021.

[5] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision Transformer Adapter for Dense Predictions," in *The Eleventh International Conference on Learning Representations*, 2023.

[6] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. S. Torr, and L. Zhang, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*. Computer Vision Foundation / IEEE, 2021, pp. 6881–6890.

[7] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 12 077–12 090.

[8] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, "Clockwork convnets for video semantic segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 852–868.

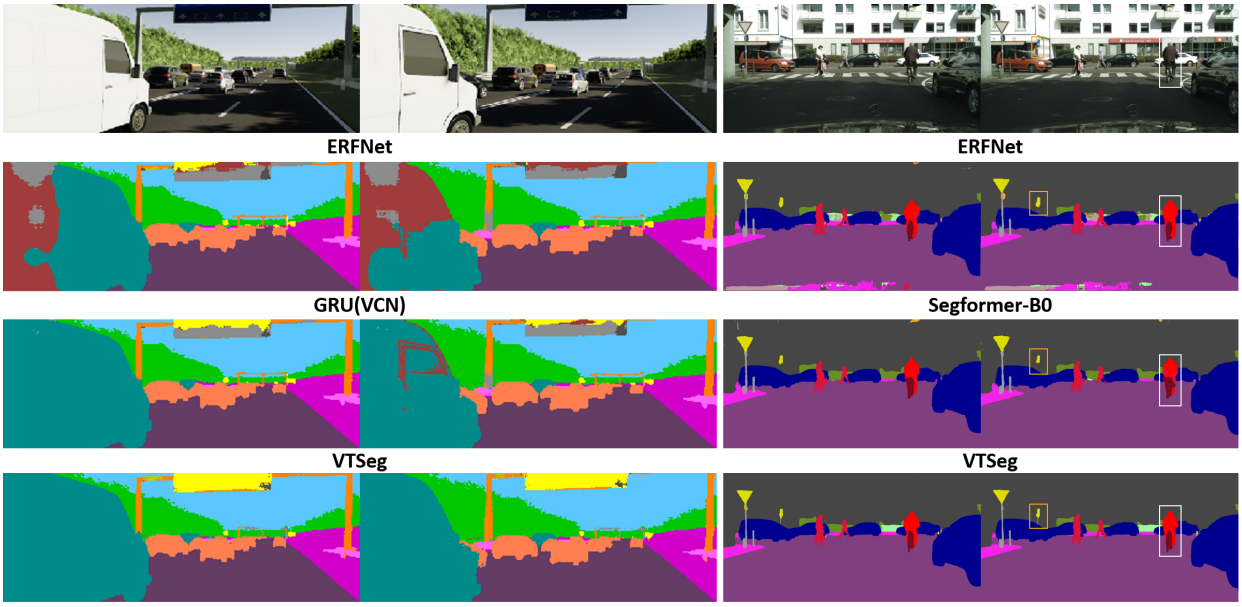

Fig. 6. Qualitative results on Virtual KITTI (left) and Cityscapes (right) using video sequences of length 2. The first row is the input video sequence.

[9] D. Nilsson and C. Sminchisescu, "Semantic Video Segmentation by Gated Recurrent Flow Propagation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6819–6828.

[10] V. Lup and S. Nedevschi, "Semantic Segmentation on Video Sequences leveraging Dense Optical Flow," in *International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2020.

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Y. Cabon, N. Murray, and M. Humenberger, "Virtual KITTI 2," *arXiv preprint arXiv: 2001.10773*, 2020.

[13] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *International Journal of Computer Vision*, vol. 81, pp. 2–23, 2009.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 12 1989.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6230–6239.

[17] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4 2018.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." in *ECCV (7)*, 2018.

[20] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, X. Wang, and Y. Qiao, "InternImage: Exploring Large-Scale Vision Foundation Models With Deformable Convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 14 408–14 419.

[21] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[22] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 2017.

[23] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for Semantic Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7262–7272.

[24] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10 002, 2021.

[25] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting spatial attention design in vision transformers," *ArXiv*, vol. abs/2104.13840, 2021.

[26] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic Video CNNs Through Representation Warping," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4463–4472.

[27] P. Hu, F. C. Heilbron, O. Wang, Z. L. Lin, S. Sclaroff, and F. Perazzi, "Temporally distributed networks for fast video semantic segmentation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8815–8824, 2020.

[28] H. Wang, W. Wang, and J. Liu, "Temporal memory attention for video semantic segmentation," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 2254–2258.

[29] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[30] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, jun 2016, pp. 1874–1883.

[31] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061–3070.

[32] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2018.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, p. 211–252, 4 2015.

[34] G. Yang and D. Ramanan, "Volumetric Correspondence Networks for Optical Flow," in *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2019, pp. 794–805.