CUDA Vision Lab Project Report

Vibhor Sharma and Mohammad Mehdi Deylamipour

Rheinische Friedrich-Wilhelms-Universitat Bonn
s58vshar@uni-bonn.de, s25mdeyl@uni-bonn.de
50010826,50009389

Abstract. In this project we tried an unique approach to video semantic segmentation[1], focusing on the dynamic and challenging environments of automotive scenes. With the objective of overcoming the limitations posed by frame-by-frame processing—such as flickering and ghosting, and the complexities introduced by ego-motion and occlusions—our method employs a recurrent semantic segmentation model. This model is built upon a robust architecture that includes a semantic segmentation model like UNet, along with a recurrent filter mechanism for structured state management[2]. The core innovation lies in the integration of two critical filters: an ego-motion filter that models the camera's movement, thereby ensuring the temporal consistency of the camera states across frames, and a geometry filter that captures the scene's geometric nuances, maintaining the temporal consistency of scene content and geometry. This dual-filter approach not only enhances the segmentation accuracy but also provides additional outputs related to scene geometry such as depth, invaluable for applications in autonomous driving, robotics, and agriculture. We train and validate our model on the CARLA dataset, designed for autonomous driving simulation, which includes semantic segmentation and depth estimation across diverse urban environments.

1 Introduction

The task of video semantic segmentation, which involves assigning a semantic category to every pixel in every frame of a video sequence, is a fundamental problem in computer vision with wide-ranging applications in autonomous driving[3], robotics, and agriculture[4]. This task is significantly more challenging than static image segmentation due to the additional dimension of time, which introduces dynamic changes and temporal dependencies within the scene. Traditional frame-by-frame processing methods often fail to capture these temporal dynamics, leading to inconsistencies such as flickering and ghosting effects. Moreover, the presence of ego-motion—the movement of the camera itself, typically in automotive contexts—complicates the segmentation task further by introducing significant changes in the scene from frame to frame. Our approach leverages the power of recurrent neural networks (RNNs) and specifically tailored filters to enhance the segmentation process. Our model is built upon a foundation of proven segmentation architectures such as UNet, incorporating a recurrent filter mechanism that facilitates structured state management across frames. This allows our model to maintain temporal consistency and accurately capture the evolution of the scene over time. A key innovation of our approach is the integration of two specialized filters: the ego-motion filter and the geometry filter. The ego-motion filter is designed to model the movement of the camera, providing a mechanism to account for changes in camera perspective between consecutive frames. By accurately modeling this motion, we can significantly reduce the impact of camera movement on segmentation accuracy. The geometry filter, on the other hand, captures abstract scene features such as the geometry and contents of the scene, ensuring that the temporal consistency of these features is maintained throughout the video sequence.

2 Data Preparation

Our project uses the CARLA dataset, an advanced simulation environment designed specifically for autonomous driving applications. This dataset provides a comprehensive framework for training and evaluating our video segmentation model, offering approximately 12,000 sequences that capture a wide array of urban driving scenarios across different towns.

2.1 Dataset Creation:

To facilitate seamless integration into our training pipeline, we devised a custom PyTorch Dataset class, named *SegmentationDataset*. This class is responsible for generating a sequence of 6 frames of images, segmentations and depth starting from a random start index. We also include logic to discard the sequences with irregular number of images.

Data Transformation: We applied a series of transformations to the loaded data to ensure compatibility with our model and that only valid samples are included in the batch. Specifically, we employed some custom transformation classes:

- ConvertToRGB: Converts RGBA images to RGB format by discarding the alpha channel.
- Corrupt Frames: Introduces corruption or noise into %10 of training frames.
- DepthImageTransform: Processes depth images to ensure they are within a normalized range suitable for depth training.
- Colorjitter: Used to randomly change the brightness, contrast, saturation, and hue of an image

Additionally, we employed standard PyTorch transformations such as resizing images to a fixed size and converting them to tensors.

Data Splitting: To facilitate model training, validation, and testing,the dataset is partitioned into three subsets: training, validation, and testing. For the training phase, sequences from Towns 01, 03, 04, 05, 06, and 07 are utilized.The validation phase employs sequences from Town 02. And for the testing phase, Town 10 is designated.

3 Method

3.1 Model

As the first step towards the task, we made a baseline model which was a U-Net architecture[5] using Resnet-34[6] backbone. The U-net architecture(see Figure 1) offers a good balance between performance and efficiency which makes it suitable for various segmentation tasks.
It includes of the following components:

- Encoder: initializes a ResNet backbone, removes the average pooling and fully connected layers.
- Decoder: used to upsample feature maps. Each block consists of a transposed convolution layer followed by ReLU activation and batch normalization.
- Segmentation class: combines the encoder backbone and decoder blocks to form a fully convolutional network for different stages of downsampling and upsampling. The model provides methods to freeze and unfreeze the encoder's parameters, allowing fine-tuning of the decoder while keeping the encoder weights fixed.

We used an identical set of decoder blocks for the depth output head. Since the depth segmentation is a regression task we used 1 channel as output.

3.2 Training Methods

to Generate Segmentation images, the model takes an input image and produces an output tensor with multiple channels, where each channel represents the probability distribution of each class. In this case, the model is configured to have 23 output channels, which corresponds to the 23 classes that need to be segmented in the input image.

Cross-entropy loss [7] is calculated between the predicted segmentation map and the ground truth segmentation values.

For depth prediction, where the goal is to estimate the distance of objects in a scene the L1 loss is used to measures the absolute difference between the predicted depth values and the ground truth depth values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y_i}|$$

The Adam optimizer is chosen for training the model,it adapts the learning rates for each parameter individually based on estimates of the first and second

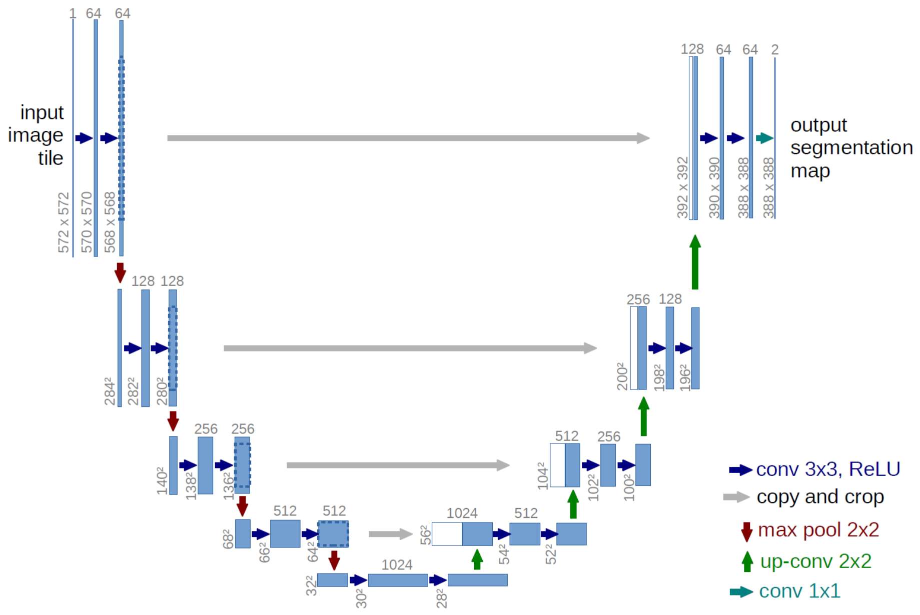

Fig. 1. A typical U-Net Architecture

moments of the gradients. The initial learning rate is set to 3e-4, which is a common choice for many deep learning tasks.

The ReduceLROnPlateau scheduler[8] is used to improve convergence and prevent overshooting. it reduces the learning rate by a factor of 0.1 when the loss stops decreasing and takes 10 epochs to wait before reducing the learning rate when the loss plateaus.

3.3 Evaluation Metrics

When evaluating the model in each epoch, IoU and mean accuracy are used for a comprehensive assessment of its segmentation capabilities. IoU offers insights into the spatial agreement between predicted and ground truth segmentations for individual classes:

$$IoU = \frac{Area\,of\,Overlap}{Are\,of\,Union}$$

Furthermore, mean accuracy provides an overall measure of pixel-wise classification accuracy across all classes.

4 Experimental Details

We tried training our model with various strategies including batch normalization, using combined and separate losses and training on single towns. To do batch normalization, we tried updating weights after 4 epochs but the intermediate validations were poor and there were no signs of model being trained quicker. For this approach we used only 1 town for training. For our second experiments, we ran model once with combined losses and other time with separate loss for segmentation and depth. We preferred the second approach more as it gave more freedom and visibility to see if one there is issues with which losses. In the end we could train for 12 epochs the baseline model, with batch size of 2 and no batch normalization and separate loss functions. Please look at the figures 2,3,4,5 from logs.

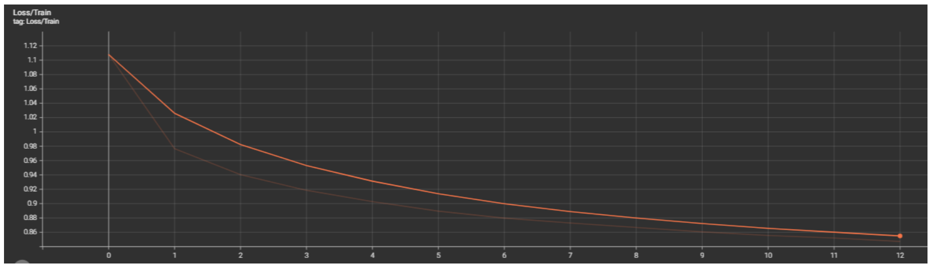

Fig. 2. Training loss

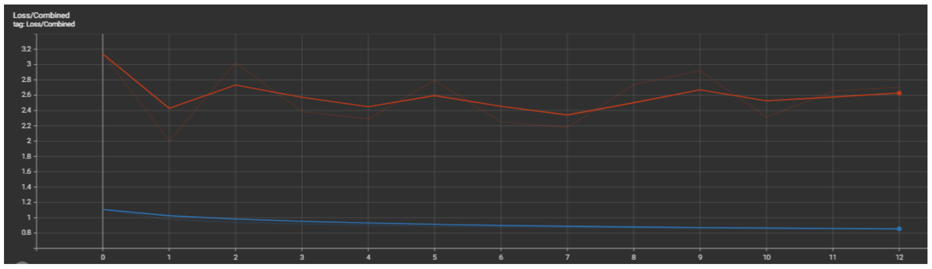

Fig. 3. Training and Validation loss

5 Results

The results of the baseline model can be seen in figures 6 and 7. The model has mAcc value of 0.43 and mIou at 0.075. The lower values can be accredited to factors like lower training epochs, some disturbances in dataset due to transforms.

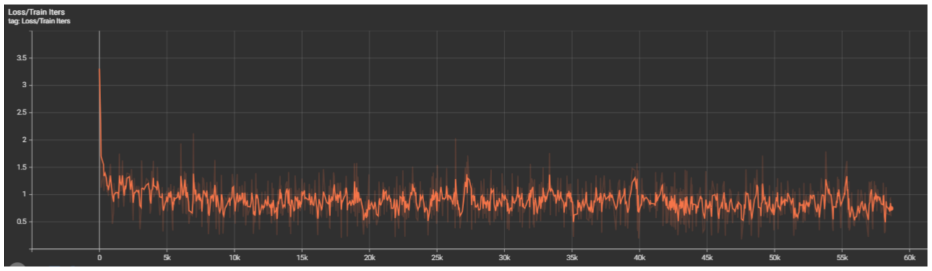

Fig. 4. Loss value for every 30 iterations

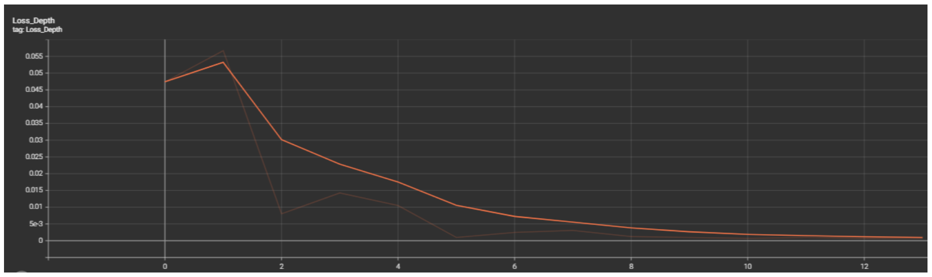

Fig. 5. Depth loss

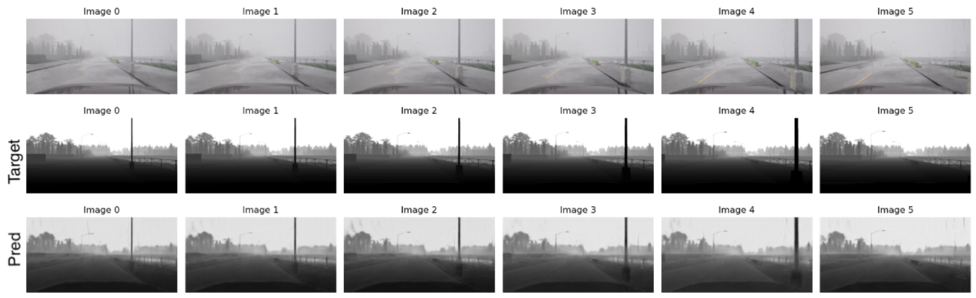

Fig. 6. Sample Depth Images

6 Challenges

During our project we faced challenges to get access to GPU hours. We managed to get access to a lower memory GPU towards the end of deadline, which ran into Out of Memory errors while using bigger batches. The training time for each epoch varied between 1 and 2 hours which made testing new models even more challenging. Towards the end it was impossible to formulate and train with Rnn approach, so we focused to perfect the baseline model.

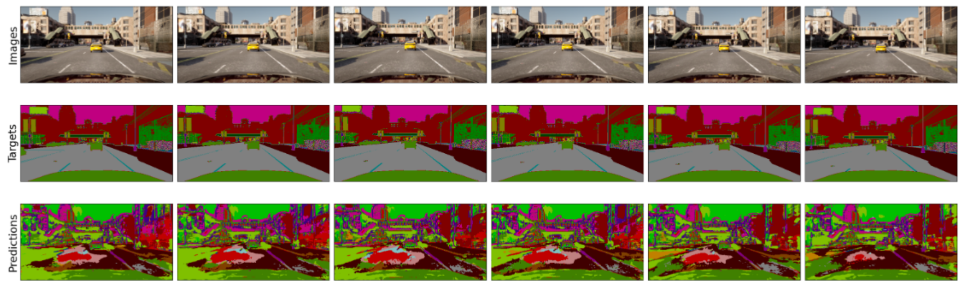

Fig. 7. Sample Segmentation Images

7 Conclusion

In conclusion, our research presents a comprehensive approach to structured video segmentation, specifically tailored for the intricate scenarios encountered in automotive environments. By leveraging a novel combination of recurrent neural networks, ego-motion filters, and geometry filters, integrated within a robust segmentation framework like UNet, we can successfully addressed several critical challenges inherent in video segmentation. These include the temporal inconsistencies, the complexities introduced by the camera's own movement, and the difficulty in capturing the geometric intricacies of urban scenes. The utilization of the CARLA dataset, with its diverse array of urban driving conditions spread across different towns for training, validation, and testing, has been instrumental in evaluating our model's performance. **Even though our implementation fails to capture the architecture proposed, we have done our research to support its validity.**

8 Contributions

Vibhor was responsible for dataset creation and model generation. Mehdi was responsible for visualizations, evaluation metrics and training the models. Both contributed equally to the report.

References

1. Manuel Rebol and Patrick Knöbelreiter. Frame-to-frame consistent semantic segmentation. *CoRR*, abs/2008.00948, 2020.
2. Imtiaz Hassan, Huma Zia, Hafiza Sundus Fatima, Syed Yusuf, and Muhammad Khurram. A lightweight convolutional neural network to predict steering angle for autonomous driving using carla simulator. *Modelling and Simulation in Engineering*, 2022:1–11, 08 2022.
3. Sujan Gannamaneni, Sebastian Houben, and Maram Akila. Semantic concept testing in autonomous driving by extraction of object-level annotations from carla. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1006–1014, 2021.

4. Jörg Wagner, Volker Fischer, Michael Herman, and Sven Behnke. Functionally modular and interpretable temporal filtering for robust segmentation. *CoRR*, abs/1810.03867, 2018.
5. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
6. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
7. Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.
8. Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua. Scheduling techniques for liver segmentation: Reducelronplateau vs onecyclelr, 2022.