Memory Guided Road Detection

Praveen Venkatesh*
Electrical Engineering
Indian Institute of Technology
Gandhinagar, India

Rwik Rana*
Mechanical Engineering
Indian Institute of Technology
Gandhinagar, India

Varun Jain*
Electrical Engineering
Indian Institute of Technology
Gandhinagar, India

***Abstract*—In self driving car applications, there is a requirement to predict the location of the lane given an input RGB front facing image. In this paper, we propose an architecture that allows us to increase the speed and robustness of road detection without a large hit in accuracy by introducing an underlying shared feature space that is propagated over time, which serves as a flowing dynamic memory. By utilizing the gist of previous frames, we train the network to predict the current road with a greater accuracy and lesser deviation from previous frames.**

I. Introduction

Tracking roads is a very important task in self-driving vehicles, where it is important to know the location of the roads, and where the car is positioned with respect to the road. Several existing methods rely on either algorithmic models of the roads, or using segmentation based models for predicting roads like integrating RANSAC with CNNs, RBMs, CRFs etc. However, these methods fail to address the temporal consistency of the act of road detection, and treat the problem as independently predicting roads for each frame.

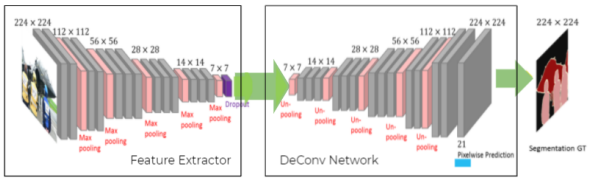


Fig. 1: FCN for Segementation [5]

In this paper, we use a combination of large and small feature extractors that learn to infer from a shared underlying flowing memory allowing for fast inference with a low accuracy dip. The proposed method takes inspiration from the human eye as to how a human can perceive complex scenes and infer a rich representation of the scenes, based on previously seen scenes.

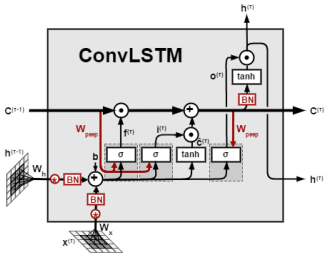


Fig. 2: Conv LSTM Model [11]

Convolutional LSTMs (Long Short Term Memory) can be used to remember and track images instead of applying object detection techniques will help save time, computational power and will be more efficient. Thus an LSTM, based memory model is proposed to detect an object in a single frame and then tracking the object using and hybrid of Resnet 18, Resnet 101, FCN 32 and Conv LSTM network. Thus this model makes the process computationally fast and is robust and thus can also be deployed on mobile devices for a smooth application of road detection. We primarily take inspiration from the paper Looking Fast and Slow: Memory-Guided Mobile Video Object Detection, where Liu et al [7] propose a method for fast object detection by using different capacity models.

II. Related Works

With the onset of machine learning, deep learning and computer vision techniques, the area of semantic segmentation has been increasingly studied. Early works such as those introduced by Long et al(2014) [10] are called FCN's (Fully Convolutional Networks). The main aim of the FCN was to replace fully connected layer by fully convolutional layers. This allows for a large reduction in the number of parameters while also introducing spatial consistency over localized patches of an image. Fisher et al(2015) [12] suggested a dilated convolutional method whose is based on the exponential expansion of the receptive field without loss of resolution or coverage.

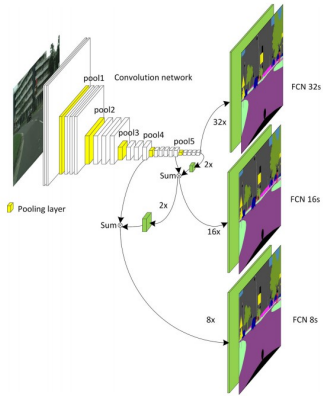


Fig. 3: A schematic of FCN implemented by Long et al

Given the volume of work done in this area, a large number of datasets such as the Microsoft COCO, PASCAL, ImageNet, SUN, CityScape have been created [3], [2], [1]. Moreover several simulators such as the CARLA, LGSVL, DeepDrive, AirSim etc. [4], [8], [9] have been extensively used to test autonomous agents safely using synthetic data. They include various functionalities ranging from semantic segmentation to depth data to help train algorithms for autonomous vehicles.

Fig. 4: A image from the Cityscape Dataset

III. Dataset

For our experiments, we use a dataset collected manually from the CARLA simulator. The dataset along with their splits can be found here: *Link*. The dataset is collected using the Town01 map, with the weather settings set to rainy. We collect the front facing RGB images, and the corresponding road segmentation using the available segmentation maps. In order to speed up the training process, we store the dataset in a single file as a list of images that is loaded entirely into memory during training. We also double the size of the dataset collected by using the flipped versions of the road and RGB images as inputs.

IV. Our Approach

Our approach consists of three main cascaded stages -

1) Feature Extraction
2) Shared Memory
3) Decoder

The memory module and the segmentation generator are shared between multiple models, whereas we utilize two feature extractors for our experiments.

A. Feature Extractors - The Interpreters

In our experiments, we use primarily use 2 variants of pretrained ResNet based architectures. Since the goal of the method is to maintain accuracy while increasing speed, we use 2 drastically different sized variants, namely the ResNet-18 and the ResNet-101 variants. The ResNet is used as a low capacity feature extractor that runs at almost 3x the speed of the ResNet-101 which is slower, but performs better. We use the weights downloaded from the Torch Model zoo which has been pretrained on the ImageNet dataset for classification tasks.

Since there are only 2 classes (no road / road), we use binary labelling where road is marked as 1 and no road is marked as 0. We use the Adam optimizer along with binary cross entropy loss for segmentation.

Initially, we perform our experiments on two VGG variants (VGG11 and VGG16). However, VGG models have a significantly larger number of parameters leading to lower speeds. However, we later switch to the two ResNet variants described above due to their ability to produce extremely high accuracy in classification techniques using considerably less training parameters. For example [6], produces an accuracy of 98.4%.

To extract features from the ResNet models, we remove the last classification layers, and use the intermediate feature vector that is generated (512 and 2048 dimensions for ResNet18 and ResNet 101). To maintain the same dimension of the feature vector, we add convolutional layers to reduce the dimensionality of the features to 512 dimensions for both the ResNet variants. The following images are the results corresponding to only Resnet18, Resnet101 and interleaving of Resnet18 & Resnet101.

B. Shared Memory - The Time Keeper

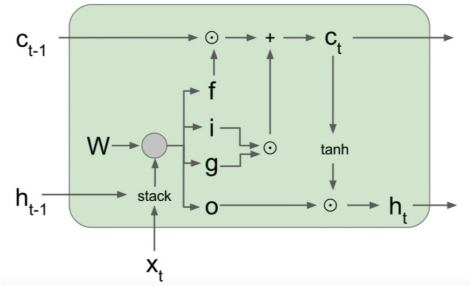


Fig. 5: A LSTM cell

The main aim of the model is to exploit the temporal consistency of time series data along with spatial data. A natural fit is hence the concept of the convolutional LSTMs, which is essentially an LSTM cell which operates on convolutional gates instead of connected gates. The LSTM stores the h_{i-1} hidden state and inputs the hidden states along with the ResNet input in the i^{th} time step. The 512 channel output of the ResNet block, becomes the input to the convolutional LSTM block. Our model employs a single LSTM cell (Fig : 5). The LSTM cell provides a 128 channel output C_i along with a cell state h_i, c_i which is the hidden state and current state of the ConvLSTM of the i^{th} timestep. The ConvLSTM is implemented as a stateful LSTM where the hidden states are propogated through frames, which thus giving a temporal feature to the enitre process.

C. Generating Segmentations - The Decoder

Once the information has passed through the ConvLSTM, the outputs of the LSTM model are passed through a decoder block which gives a segmented image of the inferred road. The decoder block is an up-sampling fully convolutional network made up of 5 transpose convolutional layers as shown in Fig.6. The up-sampling layers in later joined with a convolutional

layer to predict the final road mask. The FCN32 decreases the channels obtained from the LSTM from 128 to 3 and upsamples the image to the required size.

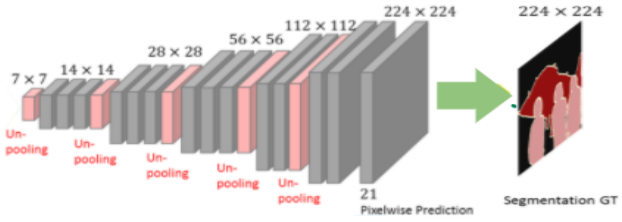


Fig. 6: Decoder for FCN32s

D. Pipeline

Integrating all of the parts, we show that 3 stages are cascaded

- Feature Extraction -The Interpreter
- Shared Memory - The Time keeper
- Segmentation Generation - The Decoder

The block diagram of the architecture is shown in Figure 7.

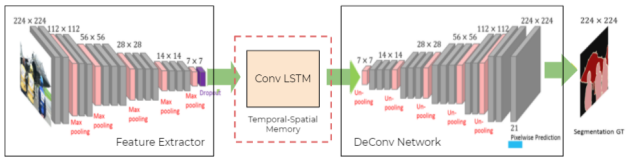


Fig. 7: Model with the Temporal-Spatial Memory

V. Interleaving - Training Pipeline

In order to train the ConvLSTM and the deconvolutional network, we use the following approach:

- At the beginning of a sequence of 6 images, randomly choose a feature extractor and pass the image through it.
- Pass the extracted features through the ConvLSTM and the Deconv block.
- Randomly sample a feature extractor and pass the next frame through it.
- Pass the previous hidden states along with the current feature through the LSTM and Deconv block.
- Repeat for the sequence of 6 frames.

An illustrative diagram of this approach is show in Fig 9. We set the extractor selector to choose the ResNet101 extractor more often than the ResNet18 extractor as it has greater accuracy. This is achieved by randomly generating a number and checking if it is greater than a threshold epsilon and thus, a feature extractor is selected.

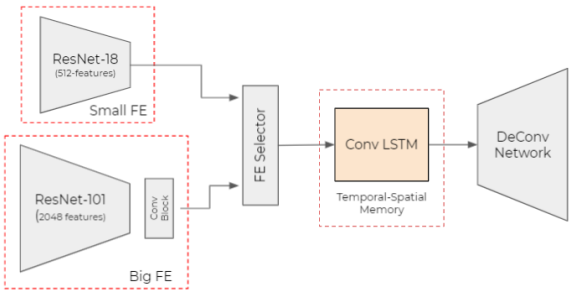


Fig. 8: Training Pipeline

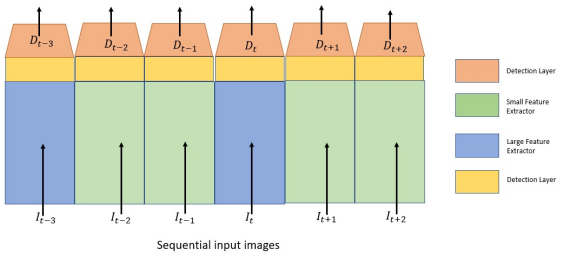


Fig. 9: Interleaving

In our experiments, we utilize two different methods for introducing temporal consistency -

- **Batched Interleaving** - The sequence of 6 images are generated by duplicating the same image 6 times.
- **Sequential Interleaving** - The+ sequence of 6 images are taken from the actual sequence.

VI. Results

For baseline comparisons, we train FCN32s deconvolution block, preceeded by different bare features extractors. We observe that using a small Feature extractor like the Resnet18, gives us a faster inference time trading off our accuracy, while using a more intricate feature extractor like ResNet-101, we get better results at the cost of inference time.

Now that our baselines are set, we setup our two training pipelines on the various data loading techniques as discussed in the earlier section.

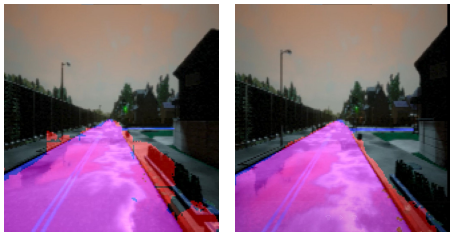
Batched & Interleaved model is the model that trains over similar images to create the lstm memory. For training the model, we replicate the images and pass them through the feature extractor and the ConvLSTM, to prepare the temporal memory. Using the memory, we infer the last image and calculate the loss after decoding the segmentation map with ground truth.

Sequential & Interleaved model is the model that trains over 6 sequential images. The LSTM memory is generated using the 5 sequential images. Using this temporal memory we infer the 6th image and calculate the loss. The weights are updated using the obtained loss.

During inference, unlike the training pipeline, we infer models using strategies that are predecided. In our experiments,

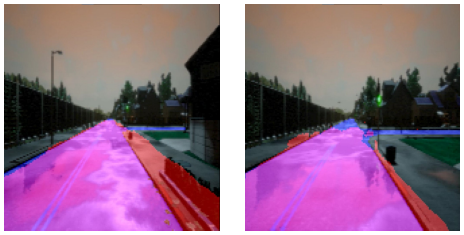
we try making choices depending on the order (Inference the large model for 1 in n frames, or by using thresholding of a random number generator). In our pipeline, we clear the weights whenever the large feature extractor is run. We do this as the results accumulate over time if the weights are not cleared.

A. Qualitative Results



(a) (b)

Fig. 10: (a) Vanilla ResNet18 (b) Vanilla ResNet101



(a) (b)

Fig. 11: (a) Batched Interleaved (b) Sequential Interleaved

Comparing results between different models, we see that the Vanilla ResNet models and interleaved models perform similarly at each given frame. We notice that occasionally the prediction moves into the pavement as well. However, it is evident from the videos that the temporal consistency of the interleaved models is far superior than the vanilla models which simply treat each frame as an independent prediction.

In the videos, we notice that predictions immediately after clearing the weights of the LSTM are quite poor. This can be attributed to the loss being computed only for the last frame in a given sequence, which does not enforce the first few frames to have good predictions. We solve this problem by passing the features through the ConvLSTM before the features are propagated through the randomized feature extractor.

We also notice that the predictions are the worse when there are sharp/sudden changes in the video. This is to be expected since we are trying to use the temporal consistency of the video to make predictions.

Videos corresponding to each model can be found here: Link

B. Quantitative Results

Name	Strategy	Avg IoU	Avg FPS
Resnet 18	Vanilla	0.852	**155.4**
Resnet 101	Vanilla	**0.915**	46.18
Batched interleaved	Randn >0.9	0.878	132.58
Batched interleaved	1 in 6	0.877	135.73
Batched interleaved	1 in 10	**0.88**	**146.02**
Batched interleaved	1 in 12	0.879	147.92
Sequential Interleaved	Randn >0.7	0.872	117.91
Sequential Interleaved	Randn >0.8	**0.872**	132.12
Sequential Interleaved	Randn >0.9	0.871	**146.53**
Sequential Interleaved	1 in 6	**0.871**	132.42
Sequential Interleaved	1 in 10	0.872	**141.31**
Sequential Interleaved	1 in 12	0.868	139.64

TABLE I: Avg IoU and Avg FPS of Various Trained Models

As seen from the table, we notice that the Vanilla ResNet18 performs the fastest, whereas the vanilla Resnet 101 is the most accurate. However, the batched interleaved and the sequqential interleaved have IOUs almost similar to Resnet 101 with the FPS achieved is nearly equal to that of vanilla Resnet18.

We notice that the batched interleaved model performs better than the sequential interleaved model.

All of the experiments were performed on a laptop on an NVIDIA GTX 1650 GPU, and AMD Ryzen 4600H CPU with 8GB RAM.

VII. Novelty

Previous approaches to road detection employ methods that treat frames without having any temporal context. To the best of our knowledge, this is the first work that utilizes an interleaved suite of feature extractors for the problem of road detection. Apart from the interleaved model, we have also implemented the following, but have not included them in the comparison due to their weak performance:

- DeepLabv3-Mobilenet based road segmentation
- Direct Convolutional LSTM based road segmentation
- Upsampling Resnet18 features to Resnet101 features
  - Here we try to replicate the features extracted by the Resnet101 by appending extra layers at the end of the ResNet18 model and upsampling the feature vector size.
  - By essentially allowing the ResNet18 to behave similar to the ResNet101, we hypothesize that the predictions would improve. However, we were unable to train the model completely due to time and resource constraints.

VIII. Observations & Conclusion

We summarize some key observations as follows:

- The interleaved model performs almost 2.5x as fast as the slowest extractor while having a performance that averages the slow and the fast extractors.
- There is a high temporal consistency between successive predictions due to the LSTM cell present in the architecture.

- The strategy employed to use a particular feature extractor affects the performance of the interleaved model. Although there is not a very wide variation in the performance, we can optimize the performance by trying to clear the weights and using the larger extractor depending on changes in the frames themselves.
- If the weights are not cleared at appropriate times, the prediction of the lanes get accumulated and lead to poor predictions.

We choose pretrained ResNet models as they dont have to be fully trained on a large database from scratch. Since the ResNets are large models, training them from scratch will take a lot of time and processing power. Hence, in the future, we can obtain better results by training the model specifically for the task of segmentation.

In the future, different strategies can be implemented for interleaving. Furthermore, we can explore the possibility of using reinforcement learning based policies to choose the extractors to alleviate the problem of slowly changing lanes / fast changing lanes.

The number of LSTM cells can also be increased to improve the temporal consistency.

IX. Individual Contributions

1) Vanilla Resnet 18, Vanilla Resnet 101 - Rwik Rana
2) VGG Implementations - Rwik Rana
3) Sequential interleaving - Praveen Venkatesh
4) Batch interleaving - Varun Jain
5) Dataloading and making the dataset - Praveen Venkatesh
6) Implementing FCN32s - Varun Jain
7) Conv LSTM - Praveen Venkatesh
8) DeepLabv3 - Rwik Rana
9) Direct convolutional LSTM - Varun Jain

X. Links

1) Dataset : Link
2) Colab File : Link
3) Videos of Inference : Link
4) Trained Models : Link
5) Github Repo: Link

References

[1] Microsoft COCO: Common Objects in Context | SpringerLink.

[2] The Pascal Visual Object Classes (VOC) Challenge | International Journal of Computer Vision.

[3] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[4] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. CARLA: An Open Urban Driving Simulator. *arXiv:1711.03938 [cs]* (Nov. 2017). arXiv: 1711.03938.

[5] Kim, Y. Two-step recurnet - younghyun.

[6] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big Transfer (BiT): General Visual Representation Learning. *arXiv:1912.11370 [cs]* (May 2020). arXiv: 1912.11370 version: 3.

[7] Liu, M., Zhu, M., White, M., Li, Y., and Kalenichenko, D. Looking Fast and Slow: Memory-Guided Mobile Video Object Detection. *arXiv:1903.10172 [cs]* (Mar. 2019). arXiv: 1903.10172.

[8] Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., Agafonov, E., Kim, T. H., Sterner, E., Ushiroda, K., Reyes, M., Zelenkovsky, D., and Kim, S. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. *arXiv:2005.03778 [cs, eess]* (June 2020). arXiv: 2005.03778.

[9] Shah, S., Dey, D., Lovett, C., and Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *arXiv:1705.05065 [cs]* (July 2017). arXiv: 1705.05065.

[10] Shelhamer, E., Long, J., and Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE transactions on pattern analysis and machine intelligence 39*, 4 (Apr. 2017), 640–651.

[11] Xavier, A. An introduction to convlstm, Apr 2019.

[12] Yu, F., and Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122 [cs]* (Apr. 2016). arXiv: 1511.07122.