Semantic Segmentation of RGBD Videos with Recurrent Fully Convolutional Neural Networks

by

Ekrem Emre Yurdakul

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

29 September, 2017

Semantic Segmentation of RGBD Videos with Recurrent Fully Convolutional Neural Networks

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ekrem Emre Yurdakul

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Yücel Yemez

_____

Prof. Dr. Gözde Ünal

_____

Prof. Dr. A. Murat Tekalp

Date: _____

To my parents, Canan and Sebahattin...

ABSTRACT

Semantic segmentation of videos using neural networks is currently a popular task, however the work done in this field is mostly on RGB videos. The main reason for this is the lack of large RGBD video datasets, annotated with ground truth information at the pixel level. In this work, we use a synthetic and a real RGBD video dataset to investigate the contribution of depth and temporal information to the video segmentation task using fully convolutional and recurrent fully convolutional neural network architectures. Additionally, we employ weight transfer from fully convolutional neural networks to recurrent fully convolutional neural networks and investigate different depth encoding schemes. Our experiments show that the addition of depth information improves semantic segmentation results and exploiting temporal information results in higher quality output segmentations.

ÖZETÇE

Yapay sinir ağları kullanarak videoların semantik bölütlenmesi şu sıralar popüler bir görevdir, ancak bu alanda yapılan çalışmalar çoğunlukla RGB videolar üzerinedir. Bunun temel nedeni, piksel seviyesinde sınıf bilgisi içeren büyük RGBD video veri kümelerinin bulunmamasıdır. Bu tezde, tamamen konvolüsyonel ve özyinelemeli tamamen konvolüsyonel yapay sinir ağları mimarileriyle video semantik bölütlenmesine derinlik ve zaman bilgilerinin katkılarını araştırmak için bir sentetik ve bir gerçek RGBD video veri kümesi kullanıyoruz. Ek olarak, tamamen konvolüsyonel yapay sinir ağlarından özyinelemeli tamamen konvolüsyonel yapay sinir ağlarına ağırlık aktarımı yapıyoruz ve farklı derinlik kodlama metodları uyguluyoruz. Deneylerimiz, derinlik bilgisinin semantik bölütleme sonuçlarını geliştirdiğini ve zaman bilgisinden faydalanmanın daha yüksek kaliteli çıktı bölütlemeleriyle sonuçlandığını göstermektedir.

ACKNOWLEDGMENTS

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) Grants 114E628 and 215E201.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Semantic segmentation aims to assign an object class label to every pixel in a given image, hence it is a key task in computer vision towards scene understanding. Before deep neural network architectures had been proved to be useful and found practical applications in various vision tasks, semantic segmentation methods mostly relied on patch-wise training of handcrafted features with conventional classifiers. The focus has recently shifted to methods that use neural networks. In particular, deep convolutional neural networks, when trained on large datasets, can achieve pixel level segmentation on a given image and can solve the recognition and localization problems at the same time by globally analyzing the image as a whole.

Solving the semantic segmentation problem could enable many applications currently not possible and further improve performance of current systems in various fields such as robotics and autonomous driving. Robots could identify objects in their environments with near human accuracy and perform tasks on their own while navigating in their environments. In the field of autonomous driving, semantic segmentation could provide extra information not possible through other approaches such as a bounding box approach, resulting in a more secure driving experience by determining the locations of other vehicles and pedestrians more precisely.

With the introduction of fully convolutional neural networks [Long et al., 2014], the use of deep neural network architectures has become popular for the semantic segmenta-

tion task. Most of the work done in this field is on RGB images [Badrinarayanan et al., 2015, Long et al., 2014, Noh et al., 2015, Chen et al., 2014, Liu et al., 2015, Marmanis et al., 2016, Tsogkas et al., 2015] with a few extensions to RGB videos, which make use of recurrent neural networks so as to incorporate temporal information into the semantic segmentation task [Badrinarayanan et al., 2015, Valipour et al., 2016, Ren and Zemel, 2016, Pohlen et al., 2016]. Although there are numerous works on RGBD image segmentation [Couprie et al., 2013, Gupta et al., 2014, Deng et al., 2015, Hazirbas et al., 2016], there exists currently no work in the literature, except [Pavel et al., 2015], that addresses the problem of pixel level segmentation of RGBD videos. The primary reason for this is that while there exist a good number of RGB image and video datasets available with pixel level annotation, such RGBD video datasets are fairly limited. Besides, most of the existing semantic segmentation methods transfer weights from pretrained deep convolutional neural networks such as the VGGnet [Simonyan and Zisserman, 2014b] to train their architectures, whereas no such deep architectures pretrained on large datasets exist with RGBD data. We believe that the depth information contained in RGBD data can be exploited as an additional modality to improve the performance of the semantic segmentation task.

In this work, we address the problem of pixel level semantic segmentation of RGBD videos using both fully convolutional neural networks and recurrent fully convolutional neural networks. Our primary contribution is a fusion scheme based on existing recurrent and fully convolutional neural network architectures, which combines color and depth information in RGBD videos for semantic segmentation. We train our bimodal recurrent fully convolutional neural networks with a progressive strategy based on weight transfer. We first convert depth images to color images using linear and nonlinear encoding schemes, and initialize the convolution layers with the weights of VGGNet pretrained on ImageNet, and train two separate fully convolutional neural networks for color and depth modalities via backpropagation. We then transfer the weights of these unimodal architectures to the recurrent fully convolutional neural networks to train our final bimodal architectures. Our work has been accepted for publication [Yurdakul and Yemez, 2017] in *ICCV 2017 4th IEEE/ISPRS Joint Workshop on Multi-Sensor Fusion for Dynamic Scene Understanding*.

In Chapter 2, we review the related work on semantic segmentation, particularly using neural networks. Chapter 3 describes the datasets we use to test our neural network architectures, which are then explained in Chapter 4. The experiments we conduct are presented in Chapter 5, and the concluding remarks are finally given in Chapter 7.

Chapter 2

RELATED WORK

The ImageNet challenge [Russakovsky et al., 2014] has led to (very) deep convolutional neural network architectures such as AlexNet [Krizhevsky et al., 2012], VGGNet [Simonyan and Zisserman, 2014b] and ResNet [He et al., 2015]. Although these networks were originally trained with the goal of RGB based image classification, since the RGB image datasets on which they were trained are very large, they can easily be generalized to other datasets as well as to various other vision tasks. The most common approaches to transfer learning from these pretrained deep convolutional neural networks (CNNs) are either to use them as feature extractors or to transfer weights for network initialization [Yosinski et al., 2014]. This type of transfer learning from pretrained neural networks is commonly employed by the state of the art semantic segmentation methods.

2.1 Neural Networks for Semantic Segmentation

Semantic segmentation networks in the literature differ in their architectures. They are most commonly fully convolutional [Long et al., 2014], where all layers of the network are composed of convolution layers. They are either composed of a single deconvolution layer [Long et al., 2014] or multiple deconvolution layers [Badrinarayanan et al., 2015, Noh et al., 2015].

In the case of a network with a single deconvolution layer, inputs are first fed to the convolution and pooling layers which reduce their size. Lastly, they are fed to the deconvolution layer which increases their size in order to generate predictions at the original input resolution.

On the other hand, a network may be composed of multiple deconvolution layers by mirroring the convolutional section of the network with corresponding deconvolution and unpooling layers [Badrinarayanan et al., 2015, Noh et al., 2015]. In this type of networks, called encoder-decoder type in the literature, each convolution layer has a corresponding deconvolution layer and each pooling layer has a corresponding unpooling layer.

In order to combine multiple modalities in semantic segmentation networks, multiple processing streams can be present in a single network as in neural networks used in other fields for various other tasks [Simonyan and Zisserman, 2014a, Wu et al., 2015]. Each stream is responsible for a single modality and these streams are later fused to produce a single prediction. In this work, we use the depth and color modalities.

If temporal data is present, recurrent layers can be added to exploit temporal relationships. The addition of the time component adds another level of challenge to the semantic segmentation task, as predictions of networks must take temporal relationships of pixels into account. Convolutional recurrent layers are a crucial building block of semantic segmentation networks that exploit temporal data. Whereas recurrent layers accept vector inputs and produce vector outputs, convolutional recurrent layers take matrices as inputs and produce matrices as outputs. As a result, spatial information in convolutional recurrent layers is preserved and can be exploited as opposed to standard recurrent layers [Liang and Hu, 2015]. In our work, we use convolutional recurrent layers similar to [Shi et al., 2015, Valipour et al., 2016].

2.2 Modalities

Semantic segmentation methods most commonly employ color and depth modalities. Color is the most widely used modality as it is the easier modality to capture data for compared to depth. Hence, there exist many works that investigate the problem of semantic segmentation of RGB images such as [Badrinarayanan et al., 2015, Long et al., 2014, Noh et al., 2015, Chen et al., 2014, Liu et al., 2015, Marmanis et al., 2016, Tsogkas et al., 2015]. These works employ convolutional neural networks, except [Chen et al., 2014] which combines convolutional neural networks with conditional random fields and [Tsogkas et al., 2015] combines convolutional neural networks with conditional random fields and restricted Boltzmann machines.

Despite the existence of many works in the literature on semantic segmentation of RGB images, there exist relatively few works that address the problem of semantic segmentation of RGB videos [Pohlen et al., 2016, Valipour et al., 2016, Ren and Zemel, 2016, Shi et al., 2015] with experiments conducted on various datasets such as [Perazzi et al., 2016, Cordts et al., 2016, Geiger et al., 2012, Li et al., 2013]. This task is more challenging compared to RGB images, since it involves incorporating the time component. Whereas methods that perform semantic segmentation on RGB images make use of convolutional neural networks, recurrent neural networks are used on RGB videos to exploit temporal data.

In [Pohlen et al., 2016], a neural network architecture similar to ResNet [He et al., 2015] is used and two processing streams are present in the architecture. The residual stream stays at the full image resolution, whereas the pooling stream does not. The pooling stream includes pooling and unpooling layers, therefore has a lower resolution than the full image resolution. However, the information in the pooling stream is combined with the residual stream through full-resolution residual units.

An online semantic segmentation approach by sliding a window over temporal data is proposed in [Valipour et al., 2016]. A recurrent neural network method is used for RGB video segmentation, with the proposed network being composed of convolution layers and a recurrent unit sliding a window over temporal data.

The literature includes numerous works on semantic segmentation of RGBD images such as [Couprie et al., 2013, Gupta et al., 2014, Deng et al., 2015, Hazirbas et al., 2016], on various datasets as *NYUv2* [Silberman et al., 2012], *SUN3D* [Xiao et al., 2013] and *SUN RGB-D* [Song et al., 2015]. Whereas [Gupta et al., 2014] adopt a decision forest and [Deng et al., 2015] a conditional random field approach, [Couprie et al., 2013] and [Hazirbas et al., 2016] train convolutional neural networks, with [Hazirbas et al., 2016] preferring an encoder-decoder type of convolutional neural network.

Although there are numerous works on semantic segmentation of RGBD images, to the best of our knowledge, there exists no semantic segmentation method of RGBD videos, except [Pavel et al., 2015], which can be viewed as a preliminary work in this field. [Pavel et al., 2015] use the *NYUv2* dataset [Silberman et al., 2012], which is not a suitable dataset for training recurrent neural networks since it is a small dataset of only 1449 frames and does not include ground truth annotations for all video frames. The network architecture in [Pavel et al., 2015] is an ad-hoc simplistic network architecture with no transfer learning and additionally takes handcrafted features as input. Furthermore, in [Pavel et al., 2015], the ground truth segmentation for all frames of the *NYUv2* dataset is estimated from the available sparse annotation based on optical flow.

We also note that there exist few other segmentation methods, such as [He et al., 2016], which take RGBD videos as input, but then use the extracted temporal information to enhance segmentation on sparsely distributed individual frames (for which annotations are available). However, instead of a recurrent neural network, a convolutional neural network is proposed, which produces segmentations for a single image using different images of the same scene.

Since producing RGBD video datasets with pixel level annotation is a much harder task compared to RGB video datasets, RGBD video segmentation remains as an unexplored field. There exist however very recent RGBD video datasets [Gaidon et al., 2016, Ros et al., 2016, Ruiz-Sarmiento et al., 2017], which provide ground truth annotations at the pixel level and has hence enabled us to conduct research on this problem.

Chapter 3

DATASETS

We perform experiments on three datasets, which include a synthetic RGBD video dataset, a real RGB video dataset and a real RGBD video dataset. All datasets provide pixel level ground truth annotation, however differ in the resolution of video frames, number of video frames and object classes present, their environment types (indoor/outdoor).

3.1 Virtual KITTI

We use the *Virtual KITTI* dataset [Gaidon et al., 2016], which contains photo-realistic synthetic RGBD videos with pixel accurate ground truth information for scene- and instance-level segmentation, and depth. Aimed at providing data for autonomous driving, this dataset contains video sequences of cars navigating roads. There are 5 different worlds, each rendered in 10 variations under different camera angles, weather conditions and time of day; *30-degrees-left, 30-degrees-right, 15-degrees-left, 15-degrees-right, clone; overcast, fog, rain; morning, sunset*. Each world has a different number of video frames, however the number of video frames remains the same in a world. The number of video frames in each world is 447, 233, 270, 339 and 837 respectively. We regroup all of the objects in the dataset into a total of 13 number of classes, ignoring the instance level categorization of the original annotation, where for example the ”car” class has several instances. Ground truth segmentations in the dataset are provided as RGB color images, which we convert into the one-hot matrix representation compatible with the pixel-wise softmax classifier.

The images in this dataset have a resolution of 1242×375 pixels, however we resize all of the images to a resolution of 224×224 pixels without preserving their aspect ratios. This downsampling enables us to train our networks more efficiently in regards to memory and computation, therefore we perform this downsampling operation on all of the datasets we use.

From the gray-scale depth images provided in the dataset, we generate additional data which is colorized depth images by applying a jet colormap ranging from red (near) over green to blue (far) as in [Eitel et al., 2015] to be able to benefit from the pretrained VGGNet weights. The standard jet colormap is however a linear encoding scheme, hence does not fully exploit the color spectrum. Thus, we also adopt a nonlinear version of the jet colormap. For this version, we quantize the available range of depth values nonuniformly and assign the same color to multiple depth values. This nonlinear assignment is achieved by first building a histogram of the available depth values and then merging bins such that the new histogram has roughly the same number of occurrences in all of the bins. We find the depth value that has the most number of occurrences and use this number as the bin size, which results in a total of 412 bins for the *Virtual KITTI* dataset. Hence, each depth image in the dataset is eventually represented by using 412 depth levels, i.e. colors. The number of depth levels is a result of the bin size, since the number of occurrences of depth values are unchanged. What changes is their distribution. The newly built histogram has roughly the same number of occurrences in each bin, however they can not all be the same, since the number of occurrences of all depth values is a discrete quantity and depth values of the same value can not be placed in different bins. Algorithm 1 shows how bins are merged through psuedocode. We note that this nonlinear encoding scheme is dataset specific and computed once only over the training set. We do not process the RGB images in any way.

Algorithm 1 Bin Merging in the Nonlinear Depth Encoding

```
1:  function ENCODE(oldHistogram, newHistogram, binSize, depthLevels)
2:      i = 0;
3:      for k = 0; k < oldHistogram.NumberOfBins; k++ do
4:          if oldHistogram.bins[k] + newHistogram.bins[i] < binSize then
5:              mergeBins(oldHistogram.bins[k], newHistogram.bins[i]);
6:          else if oldHistogram.bins[k] + newHistogram.bins[i] == binSize then
7:              mergeBins(oldHistogram.bins[k], newHistogram.bins[i]);
8:              i++;
9:          else if i == depthLevels then
10:             mergeBins(oldHistogram.bins[k], newHistogram.bins[i]);
11:         else if oldHistogram.bins[k] + newHistogram.bins[i] > binSize then
12:             i++;
13:             mergeBins(oldHistogram.bins[k], newHistogram.bins[i]);
14:         end if
15:     end for
16: end function
```

3.2 DAVIS

We use the *DAVIS* dataset [Perazzi et al., 2016] to show that our method works on real RGB videos. There are 50 Full HD videos totaling in 3455 frames with challenges such as occlusion, motion blur and appearance changes. This dataset provides pixel level ground truth annotation into 2 classes, which are *background* and *foreground*.

3.3 Robot@Home

Lastly, we use the *Robot@Home* dataset [Ruiz-Sarmiento et al., 2017] to show our method performs well on real RBGD videos, which contains video sequences of indoor scenes captured with 4 RGBD sensors. The dataset is organized into *sessions*, with each session storing captured data of rooms of an apartment. Rooms in the dataset feature daily household objects and present room types in the dataset range from bathrooms to bedrooms. The ground truth information in this dataset is produced by performing three steps for each video sequence. The first step is to build a 3D reconstruction of the scene using a point cloud representation, followed by the second step, which is to label the 3D reconstruction of the scene manually by fitting bounding boxes to all objects present in the scene. The final step is the propagation of the ground truth information to all frames in the video sequence; as the correspondence between pixels of a video frame and points in the point cloud representation is known, labeling is achieved automatically by computing intersections of the points in the point cloud with the bounding boxes and assigning class labels accordingly. Each depth image in this dataset is represented by 960 depth levels using the nonlinear depth encoding scheme.

Chapter 4

METHOD

We use custom layers in our recurrent convolutional networks, which we call C-RNN, C-LSTM and C-GRU. Simply put, these layers are extensions of the standard RNN [Elman, 1990], LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014] layers; rather than taking vectors as inputs and outputting vectors, our custom layers take matrices as inputs and produce matrices as outputs. This extension from vectors to matrices is achieved by replacing multiplication operations with convolution operations.

4.1 Recurrent Convolutional Layers

4.1.1 Convolutional RNN Layer (C-RNN)

C-RNN is the convolutional counterpart of the RNN layer [Elman, 1990]. It is the most simple convolutional recurrent layer and is dictated by Equation 4.1,

$$\boldsymbol{H}_t = \mathrm{relu}(\boldsymbol{W}_1 * \boldsymbol{X} + \boldsymbol{W}_2 * \boldsymbol{H}_{t-1} + \boldsymbol{b}) \tag{4.1}$$

where $*$ denotes the convolution operation, $\boldsymbol{H}_t$ is the hidden state matrix at time *t*, *relu* is the rectified linear unit (ReLU) [Nair and Hinton, 2010], $\boldsymbol{X}$ is the input matrix, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the parameter matrices and $\boldsymbol{b}$ is the bias vector.

4.1.2 Convolutional LSTM Layer (C-LSTM)

C-LSTM is the convolutional counterpart of the LSTM layer [Hochreiter and Schmidhuber, 1997] and calculates its gates and state according to the system of equations 4.2,

$$\begin{aligned}
\boldsymbol{I} &= \mathrm{sigm}(\boldsymbol{W}_1 * \boldsymbol{X} + \boldsymbol{W}_2 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_1) \\
\boldsymbol{F} &= \mathrm{sigm}(\boldsymbol{W}_3 * \boldsymbol{X} + \boldsymbol{W}_4 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_2) \\
\boldsymbol{O} &= \mathrm{sigm}(\boldsymbol{W}_5 * \boldsymbol{X} + \boldsymbol{W}_6 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_3) \\
\boldsymbol{J} &= \tanh(\boldsymbol{W}_7 * \boldsymbol{X} + \boldsymbol{W}_8 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_4) \\
\boldsymbol{C}_t &= \boldsymbol{C}_{t-1} \circ \boldsymbol{F} + \boldsymbol{I} \circ \boldsymbol{J} \\
\boldsymbol{H}_t &= \tanh(\boldsymbol{C}_t) \circ \boldsymbol{O}
\end{aligned} \tag{4.2}$$

where $\circ$ denotes the Hadamard product, $\boldsymbol{I}$ is the input gate matrix, $\boldsymbol{F}$ is the forget gate matrix, $\boldsymbol{O}$ is the output gate matrix, $\boldsymbol{C}_t$ is the cell state matrix at time *t*, $\boldsymbol{H}_t$ is the hidden state matrix at time *t*, $\boldsymbol{X}$ is the input matrix, *sigm* is the sigmoid function, *tanh* is the hyperbolic tangent function, $\boldsymbol{W}_1$ to $\boldsymbol{W}_8$ are the parameter matrices and $\boldsymbol{b}_1$ to $\boldsymbol{b}_4$ are the bias vectors.

4.1.3 Convolutional GRU Layer (C-GRU)

C-GRU is the convolutional counterpart of the GRU layer [Cho et al., 2014], defined by the system of equations 4.3,

$$\begin{aligned}
\boldsymbol{Z} &= \mathrm{sigm}(\boldsymbol{W}_1 * \boldsymbol{X} + \boldsymbol{W}_2 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_1) \\
\boldsymbol{R} &= \mathrm{sigm}(\boldsymbol{W}_3 * \boldsymbol{X} + \boldsymbol{W}_4 * \boldsymbol{H}_{t-1} + \boldsymbol{b}_2) \\
\boldsymbol{H}_h &= \tanh(\boldsymbol{W}_5 * \boldsymbol{X} + \boldsymbol{W}_6 * (\boldsymbol{R} \circ \boldsymbol{H}_{t-1}) + \boldsymbol{b}_3) \\
\boldsymbol{H}_t &= \boldsymbol{Z} \circ \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}) \circ \boldsymbol{H}_h
\end{aligned} \tag{4.3}$$

where $\boldsymbol{Z}$ is the update gate matrix, $\boldsymbol{R}$ is the reset gate matrix, $\boldsymbol{H}_t$ is the hidden state matrix at time *t*, $\boldsymbol{X}$ is the input matrix, *sigm* is the sigmoid function, *tanh* is the hyperbolic tangent function, $\boldsymbol{W}_1$ to $\boldsymbol{W}_6$ are the parameter matrices and $\boldsymbol{b}_1$ to $\boldsymbol{b}_3$ are the bias vectors.

C-RNN is the easiest layer to train, however has the least number of parameters. In order to increase the learning capacity of networks, C-LSTM is a more preferable choice, since it has more parameters than C-RNN with internal memory, but increases the training time. To speed up training, C-GRU can be used, which provides similar performance compared to C-LSTM with less number of parameters than C-LSTM and more number of parameters than C-RNN [Chung et al., 2014]. In the semantic segmentation task we focus on, we expect C-LSTM to perform better than C-RNN and C-GRU, since it has the most number of parameters and possesses internal memory that is potentially capable of learning more information from successive video frames in dealing with challenging conditions such as occlusions and uncovered regions.

Although the parameters of the convolution operations (stride, window size and padding) in our custom layers are tunable, we set them the same as the parameters of the convolution layers in our networks, since the results of our initial experiments presented the most performance boost with this configuration.

4.2 Network Architectures

Our networks have different architectures to accommodate various input data types as shown in Table 4.1.

Network	Input Type	Input Channels
D	Colorized Depth	3
RGB	Color	3
RGBD	Color + Colorized Depth	3 + 3

Table 4.1: Input types accepted by our networks. *-D* and *-RGB* networks accept a single input, whereas *-RGBD* networks accept two inputs.

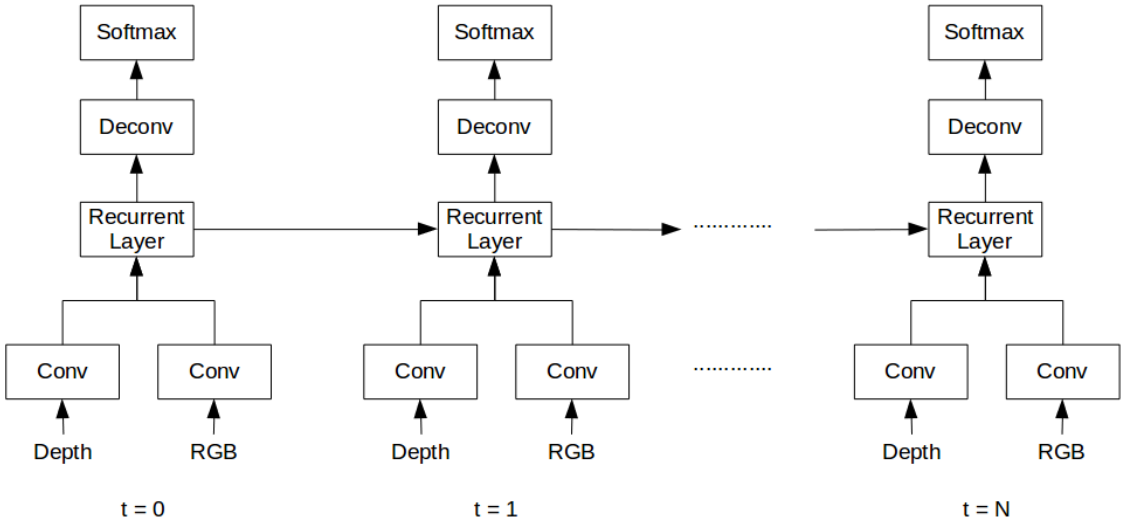

Figure 4.1: Block diagram of the recurrent fully convolutional *-RGBD* networks unrolled in time, from *t=0* to sequence length *t=N*.

We base our network architectures on the VGG-19 network (Model E) [Simonyan and Zisserman, 2014b] by mirroring its first 12 convolution and 3 pooling layers. *-D* and *-RGB* networks share the same architecture; accept a single input of size 224×224×3, however their input types are different. *-RGBD* networks on the other hand, accept two inputs of the same size of 224×224×3, where each input is processed by a different stream in the network. Whereas convolution layers produce finer outputs than their inputs, deconvolution layers produce coarser outputs than their inputs. In this context, the deconvolution operation is the inverse of the convolution operation, in the sense that $deconv(conv(x)) \approx x$. The deconvolution operation is implemented by switching the forward and backward passes of the convolution operation, such that the forward pass of the deconvolution operation is the backward pass of the convolution operation and the backward pass of the deconvolution operation is the forward pass of the convolution operation. Tables 4.3 and 4.4 show the architectures of our networks layer by layer, and Figure 4.1 visualizes the structure of our recurrent fully convolutional *-RGBD* networks. Only the *Conv-* networks are fully convolutional, whereas the *RNN-, LSTM-* and *GRU-* networks are recurrent fully convolutional networks. In the *-RGBD* networks, colorized depth and RGB are processed by two different streams of convolution layers. If the network is fully convolutional, the two streams are fused together before the deconvolution layer [Long et al., 2014] by element-wise addition and then pixel-wise softmax is applied, which determines the final output. On the other hand, if the network is recurrent fully convolutional, the two streams are fused together by element-wise addition and fed to the convolutional recurrent layer, which is either a *C-RNN*, *C-LSTM* or *C-GRU* layer. The output of the convolutional recurrent layer is then fed to the deconvolution layer [Long et al., 2014] and pixel-wise softmax is applied.

In all of our networks, all convolution and deconvolution layers are followed by ReLU activations, which are not shown for brevity. All convolution layers have padding=1, stride=1 and window=3. Deconvolution layers have padding=2 and stride=4. Pooling operations are maximum pooling with stride=window=2. We call the first 15 layers of the *VGG-19* network *TVGG-19* (Trimmed VGG-19) and use it as a building block in our networks. The architecture of *TVGG-19* is given in Table 4.2.

TVGG-19
conv3-64
conv3-64
pool
conv3-128
conv3-128
pool
conv3-256
conv3-256
conv3-256
conv3-256
pool
conv3-512
conv3-512
conv3-512
conv3-512

Table 4.2: Layers of *TVGG-19*.

Conv-	RNN-	LSTM-	GRU-
Input	Input	Input	Input
TVGG-19	TVGG-19	TVGG-19	TVGG-19
	C-RNN	**C-LSTM**	**C-GRU**
deconv	deconv	deconv	deconv
Softmax	Softmax	Softmax	Softmax

Table 4.3: Architectures of the *-D* and *-RGB* networks, layer by layer.

Conv-RGBD		RNN-RGBD		LSTM-RGBD		GRU-RGBD	
Depth Input	RGB Input	Depth Input	RGB Input	Depth Input	RGB Input	Depth Input	RGB Input
TVGG-19	TVGG-19	TVGG-19	TVGG-19	TVGG-19	TVGG-19	TVGG-19	TVGG-19
		C-RNN		**C-LSTM**		**C-GRU**	
deconv		deconv		deconv		deconv	
Softmax		Softmax		Softmax		Softmax	

Table 4.4: Structure of the *-RGBD* networks layer by layer. In each network, there are two different streams (TVGG-19), one for colorized depth and one for RGB data.

4.3 Initialization and Weight Transfer

Initial weights of the *Conv-D* and *Conv-RGB* networks are picked as the weights of the convolution layers of the VGG-19 network (Model E) pretrained on ImageNet. With this initialization, we benefit from the data learned by the VGG-19 network, since our data is composed of RGB and colorized depth. In training the *Conv-RGBD* network, we use the trained weights from the *Conv-D* and *Conv-RGB* networks as the initial weights of the convolutional layers in the corresponding streams. This transfer of weights reduces training time by increasing the convergence rate and improves pixel-wise accuracy. The recurrent fully convolutional networks are then initialized with the trained weights of the fully convolutional networks. This second transfer makes it easier for the recurrent fully convolutional networks to learn. For example, after having trained the *Conv-D* and *Conv-RGB* networks, *LSTM-D* and *LSTM-RGB* can be trained. It is possible to train *LSTM-RGBD* only after the trainings of *LSTM-D* and *LSTM-RGB* are completed. This requirement is the same for the *RNN-* and *GRU-* networks. As opposed to no weight transfer at all, our weight transfer scheme provided about a 10% accuracy boost in our initial experiments. In all of our networks, we initialize the weights of the deconvolution layers with the bilinear distribution [Long et al., 2014], the weight matrices and biases of the C-RNN, C-LSTM and C-GRU layers with the Xavier distribution [Glorot and Bengio, 2010] and zeros, respectively.

Chapter 5

EXPERIMENTS

We employ end-to-end training by minimizing the pixel-wise negative log-likelihood with Adam [Kingma and Ba, 2014] and use pixel-wise accuracy as our quantitative evaluation metric, defined as in Equation 5.1. In all settings, the learning rate is 10^{-5}. We train *Conv-D* and *Conv-RGB* networks for 100, *Conv-RGBD* for 50 and the remaining networks for 25 epochs. *Conv-D* and *Conv-RGB* networks are trained with a minibatch of size 24, whereas *Conv-RGBD* with 16. The remaining networks are trained without minibatches. The recurrent networks are trained with the backpropagation through time (BPTT) algorithm [Werbos, 1990], where the number of time steps *D* is 1. These settings apply to all datasets except stated otherwise.

$$PixelwiseAccuracy\ (\%) = CorrectlyPredictedPixels/TotalPixels \times 100 \quad (5.1)$$

5.1 Virtual KITTI

We have two different setups for the *Virtual KITTI* dataset. In all setups, we use all of the worlds in the dataset. **Setup 1** is designed to measure the variance of the networks to the angle of the camera. *30-deg-left, 30-deg-right, clone* variations are used for training; *15-deg-left, 15-deg-right* variations are used for testing. **Setup 2** is more challenging than the first setup; the first halves of all the variations are used for training and the second halves are used for testing. In the first setup there are 6378 train and 4252 test images, and in the second setup there are 10630 train and 10630 test images. We perform model selection by choosing the best performing network in the first setup, according to pixel-wise accuracy, and continue the rest of our experiments with that network, which is the *LSTM-RGBD* network as shown in Table 5.1. Therefore, we no longer train the *-RNN* and *-GRU* networks in the remaining experiments. Figure 5.1 and Figure 5.2 plot the pixel-wise negative log likelihood loss of *Conv-* and *LSTM-* networks in *Setup 1* and Figure 5.3 displays the corresponding pixel-wise accuracies, respectively. Note that we do not perform class balancing in our loss function, [Long et al., 2014] do not as well, therefore pixel-wise loss and pixel-wise accuracy are not fully related to each other, in the sense that a high loss value may correspond to a high accuracy value. This occurrence is best seen in the case of *LSTM-D* in Figure 5.2, where different multiple high loss values result in similar high accuracy values. Although there may be a slight case of overfitting, this potential issue is overcome with the fusion network, *LSTM-RGBD*.

Network	Test Accuracy (%)
Conv-D	73.73
Conv-RGB	84.87
Conv-RGBD	85.81
RNN-D	69.58
RNN-RGB	84.09
RNN-RGBD	85.81
LSTM-D	68.68
LSTM-RGB	85.26
LSTM-RGBD	**86.23**
GRU-D	69.81
GRU-RGB	85.14
GRU-RGBD	85.89

Table 5.1: Model selection on the *Virtual KITTI* dataset in *Setup 1* with linear depth encoding.

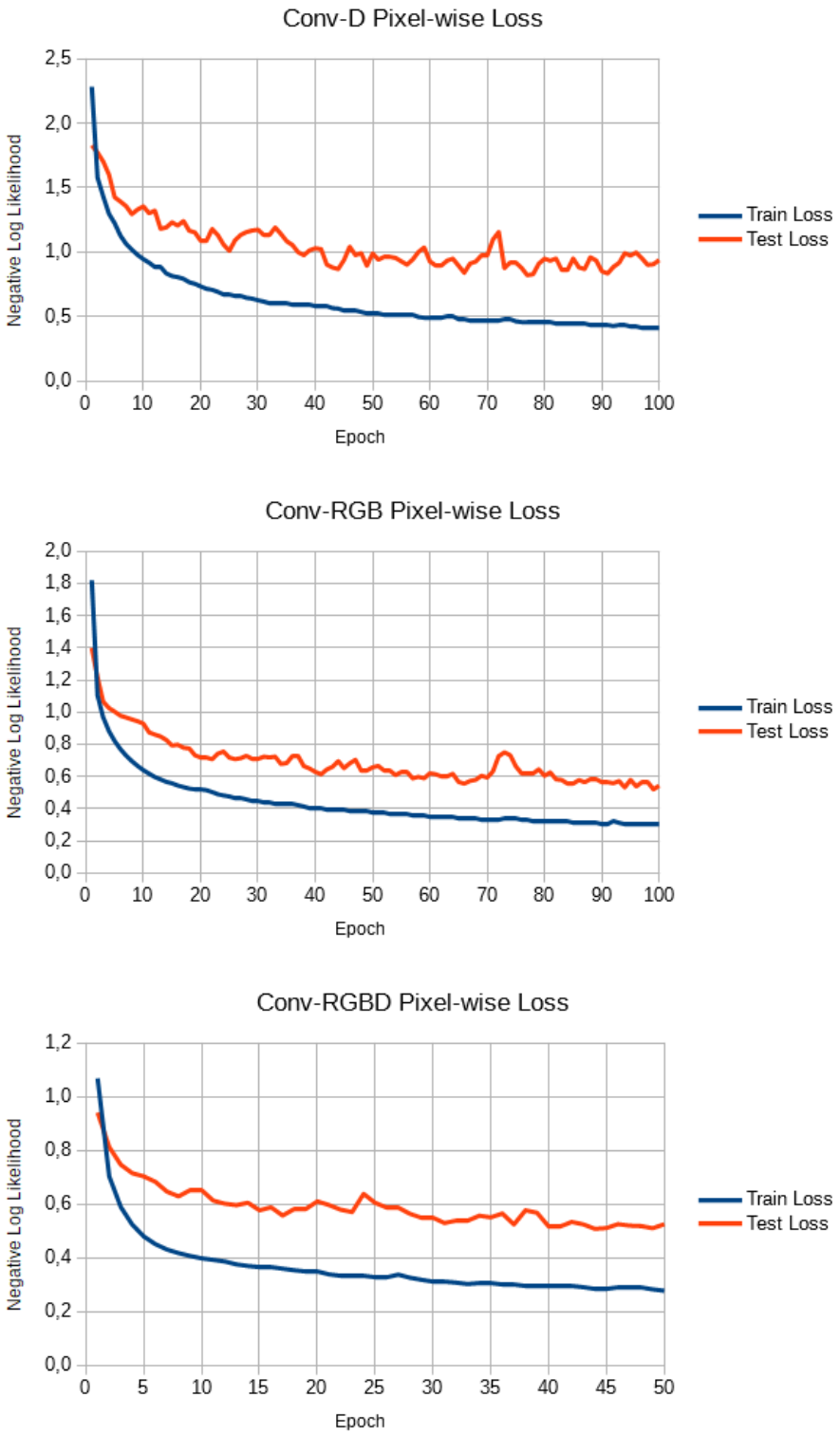

Figure 5.1: Pixel-wise negative log likelihood loss of *Conv-* networks in *Setup 1*.

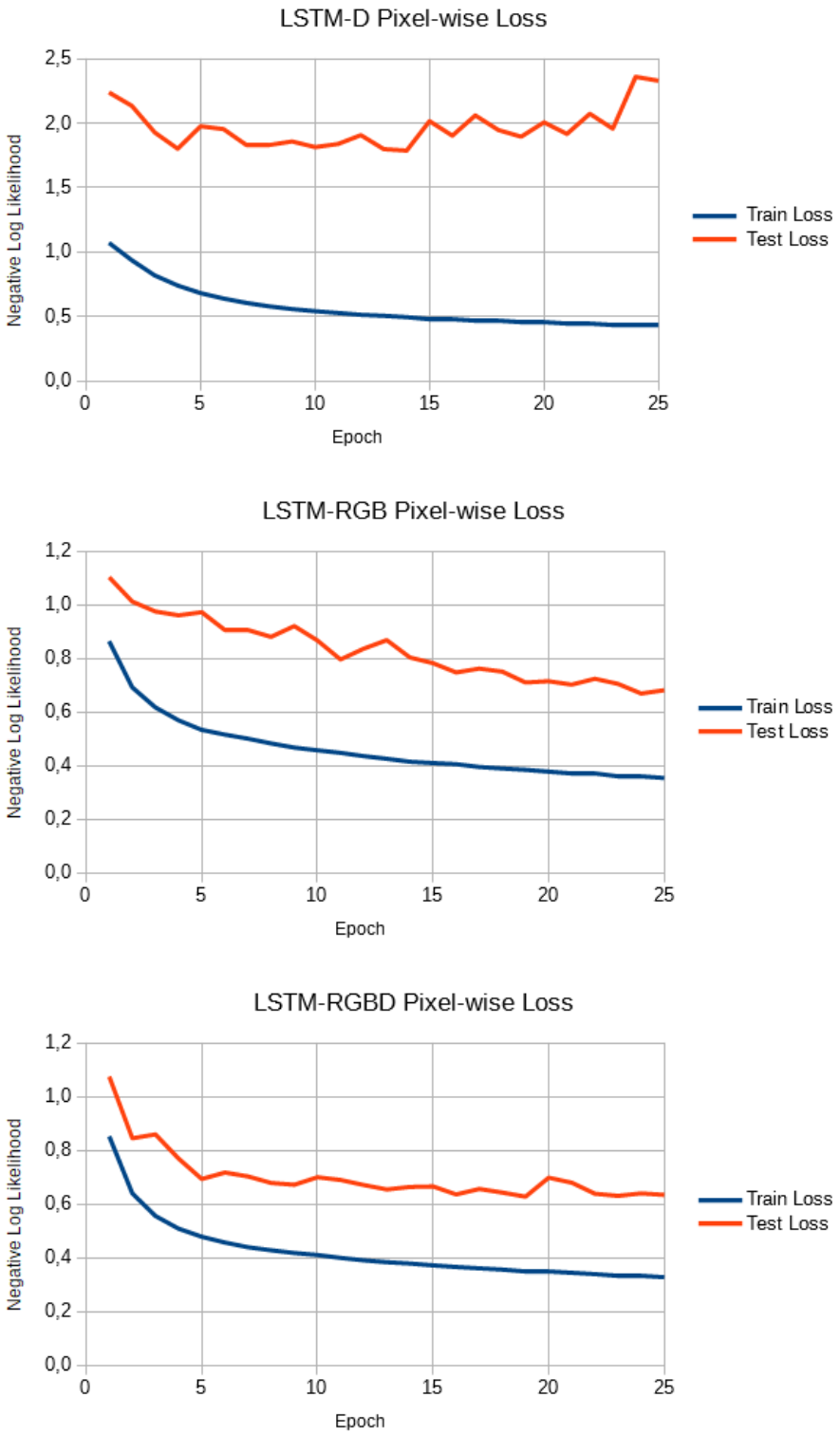

Figure 5.2: Pixel-wise negative log likelihood loss of *LSTM-* networks in *Setup 1*.

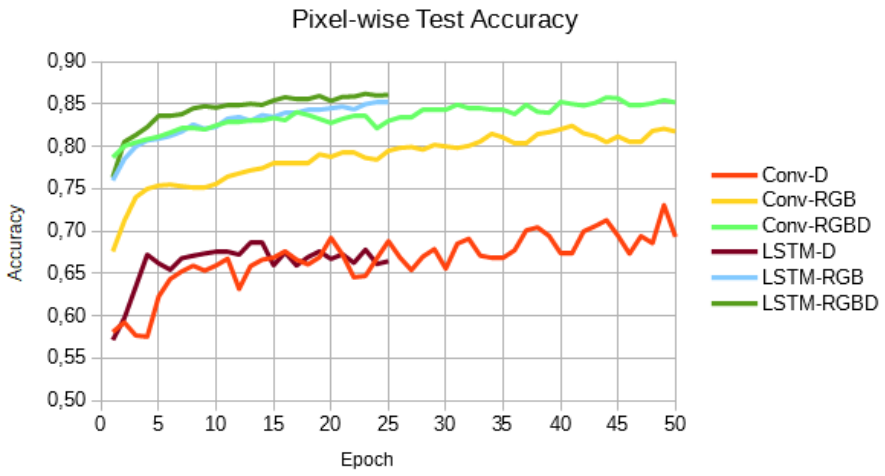

Figure 5.3: Pixel-wise accuracy of *Conv*- and *LSTM*- networks in *Setup 1*.

Results acquired in *Setup 2* are provided in Table 5.2. We test both the linear and the nonlinear depth encoding schemes in *Setup 2*, while in *Setup 1* we experiment only using the linear depth encoding scheme. We first observe that *LSTM-RGBD* and *Conv-RGBD* outperform all other networks in the first and second setups, respectively. Although we expected *LSTM-RGBD* to be the winner also in the second setup, this is not the case. This might be due to the fact that the number of training images in *Setup 2* is much higher than in *Setup 1* and the *LSTM-RGBD* network may not be able to fully exploit the temporal information on a scale this large. Despite slightly lower accuracy, the *LSTM-RGBD* network outputs higher quality segmentations as in Figure 5.4, where the boundaries around the cars are sharper and finer details are much more visible. Also in Figure 5.5, the *Conv-RGBD* network can not differentiate between the boundaries of different cars, whereas the *LSTM-RGBD* can. We also observe that in *Setup 2*, the nonlinear depth encoding scheme increases the accuracy of depth only *LSTM-D* by 5.43% and reduces the gap between *Conv-RGBD* and *LSTM-RGBD* from 0.48% to 0.14%. In any case, *-RGBD* networks perform better in test accuracy and show the addition of depth information improves semantic segmentation results.

Network	Test Accuracy (%)	
	Linear Encoding	Nonlinear Encoding
Conv-D	73.75	76.99
Conv-RGB	80.01	80.01
Conv-RGBD	**82.70**	**82.76**
LSTM-D	71.49	76.92
LSTM-RGB	79.95	79.95
LSTM-RGBD	82.22	82.62

Table 5.2: Comparison of the linear and nonlinear depth encodings on the *Virtual KITTI* dataset in *Setup 2*.

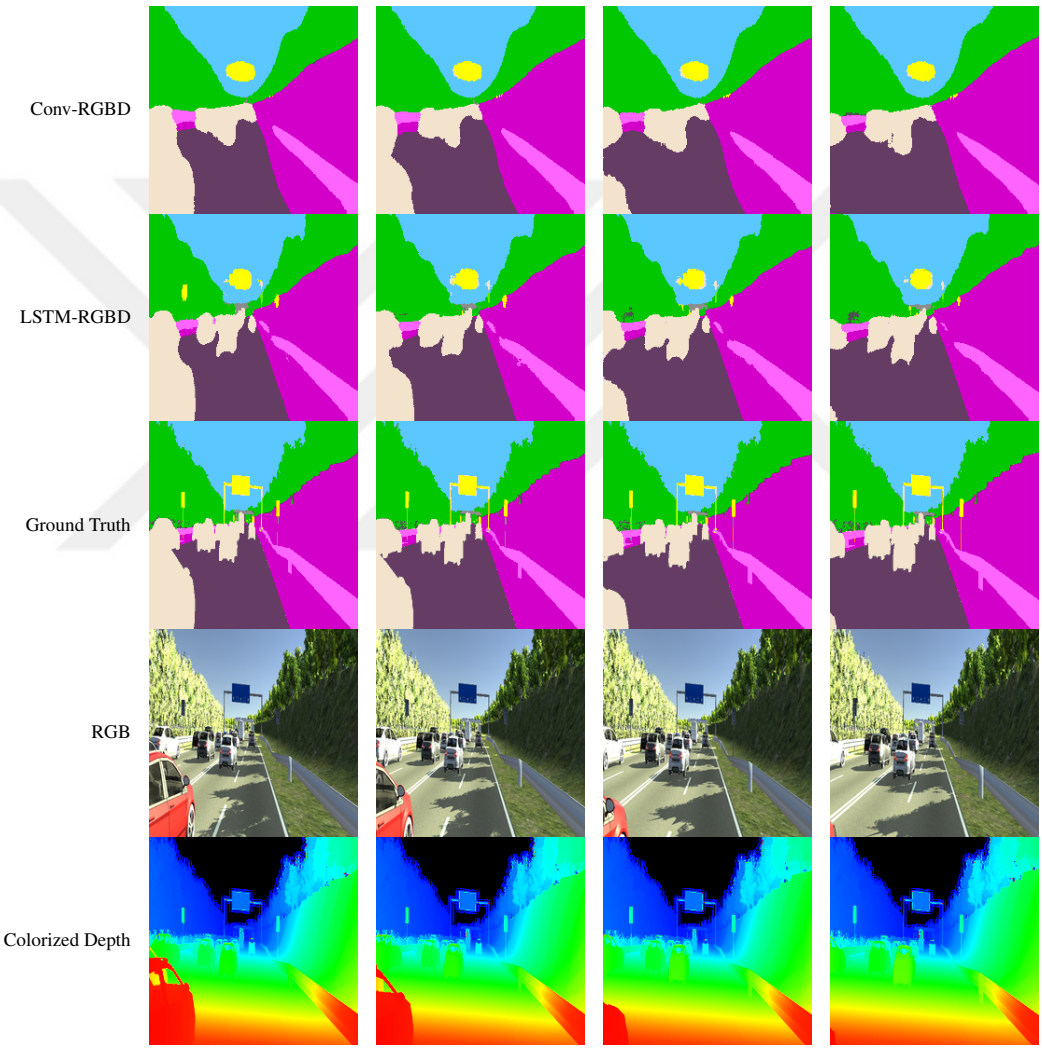

Figure 5.4: Comparison of semantic segmentations of the *Conv-RGBD* and *LSTM-RGBD* networks obtained on the *clone* sequence of world *0020* in the *Virtual KITTI* dataset in *Setup 2* with nonlinear depth encoding. The last two rows are inputs to the networks at time t. The horizontal axis (from left to right) depicts time.

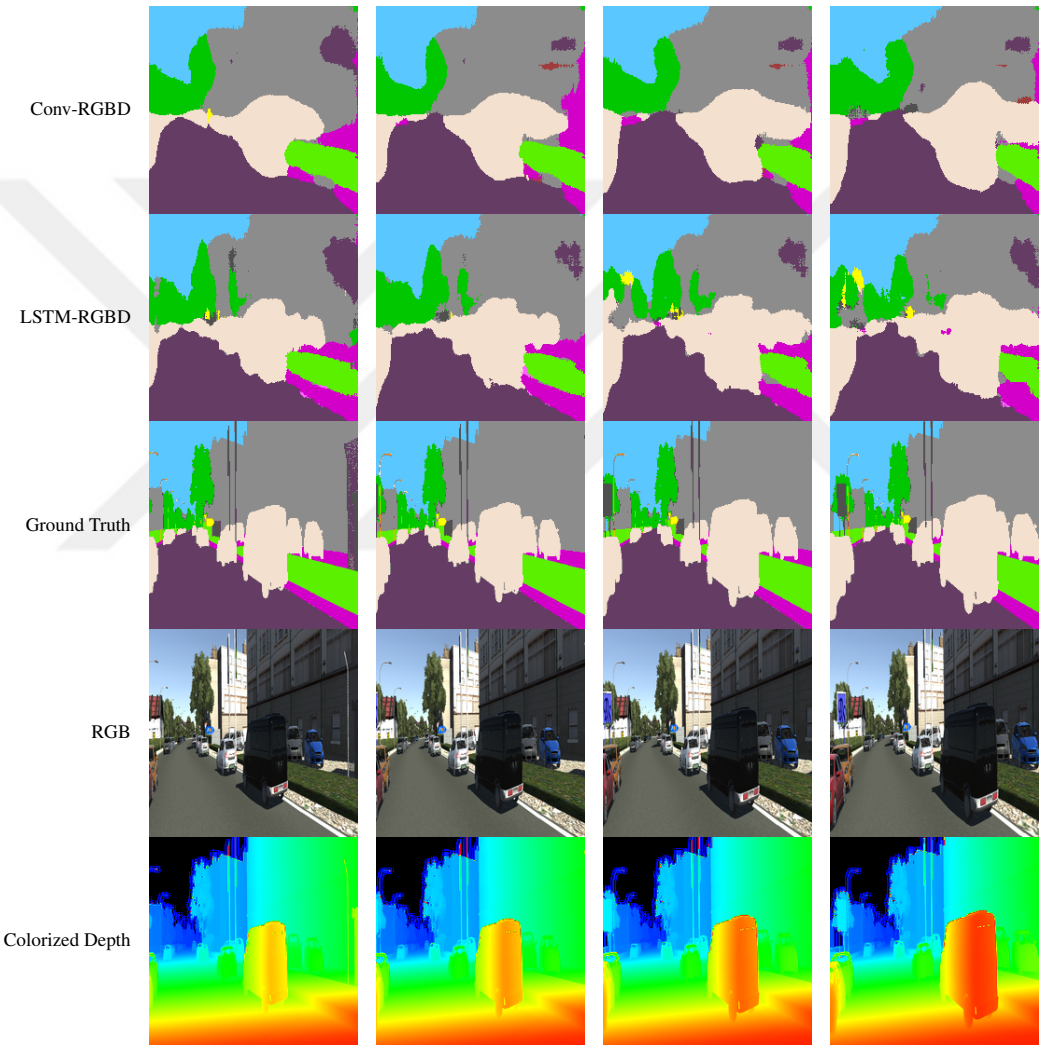

Figure 5.5: Comparison of semantic segmentations of the *Conv-RGBD* and *LSTM-RGBD* networks obtained on the *clone* sequence of world *0001* in the *Virtual KITTI* dataset in *Setup 2* with nonlinear depth encoding. The last two rows are inputs to the networks at time t. The horizontal axis (from left to right) depicts time.

5.2 DAVIS

For the experiments on the *DAVIS* dataset, we use all of the video sequences provided in the dataset and train our networks with the first halves of the sequences. The remaining second halves are used for testing. Since there is no depth data provided in this dataset, we can not train *-D* and *-RGBD* networks and only train *-RGB* networks. Table 5.3 presents our results on the *DAVIS* dataset. *LSTM-RGB* network architecture performs better in test accuracy and provides higher quality visual segmentation outputs compared to the *Conv-RGB* network as in Figure 5.6. The boundaries are sharper and the general shape of the objects are much more precise. Finer details are captured better by the *LSTM-RGB* network, while the *Conv-RGB* network generates coarser segmentations.

Network	Test Accuracy (%)
Conv-RGB	94.56
LSTM-RGB	**95.22**

Table 5.3: Results acquired on the *DAVIS* dataset.

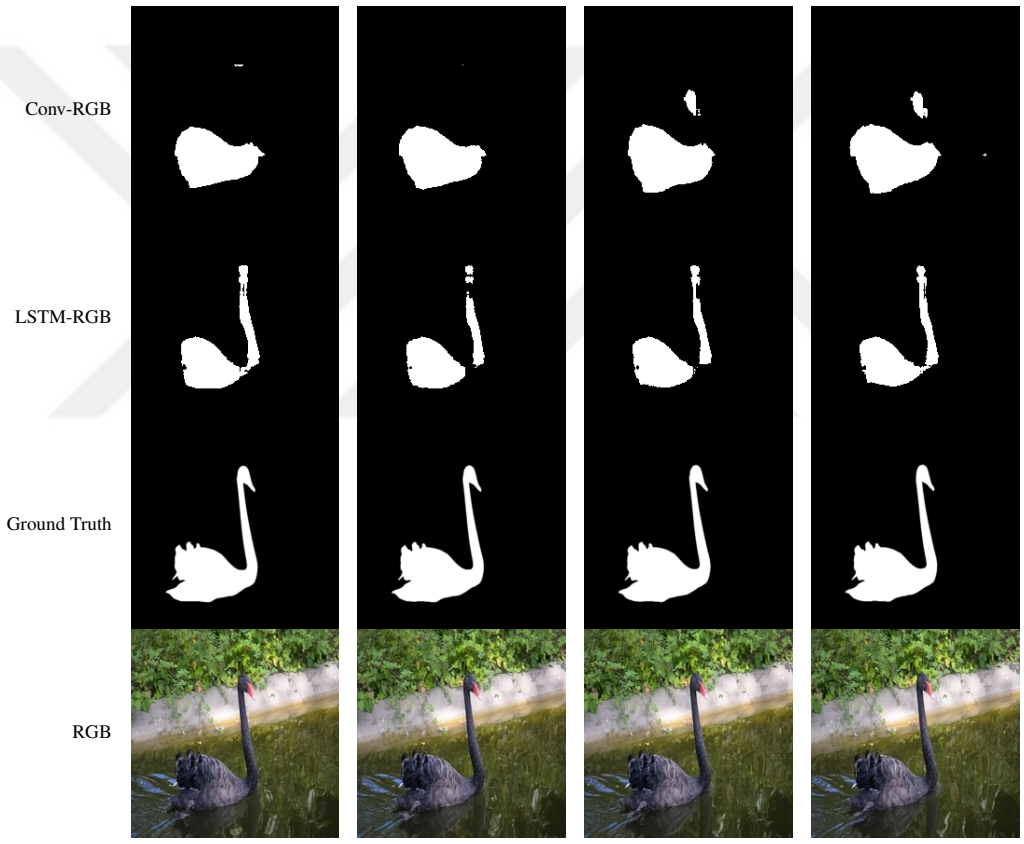

Figure 5.6: Comparison of semantic segmentations of the *Conv-RGB* and *LSTM-RGB* networks on the *blackswan* sequence in the *DAVIS* dataset. The last row is the input to the networks at time t. The horizontal axis (from left to right) depicts time.

5.3 Robot@Home

As the nonlinear depth encoding performs better, we solely use it in the experiments on the *Robot@Home* dataset. In these experiments, we use the following sessions: *alma, anto, pare, rx2*. We merge the independently captured images by the 4 RGBD sensors into a single image and regroup object classes into 5 major classes (unknown, prop, ground, wall, furniture), similar to *NYUv2* [Silberman et al., 2012]. Each sequence is split into 4 quarters temporally and training is accomplished with the 1^{st}, 2^{nd} and 4^{th} quarters, whereas testing with the 3^{rd}, to improve the overlap of objects present in the scenes, since some objects in the scenes are small and the movement of the camera is nonuniform, which significantly increase the difficulty of the task. 16 and 8 minibatches are used in the trainings of *Conv-D*, *Conv-RGB* and *Conv-RGBD* respectively. The *LSTM-* networks are trained for 50 epochs and there are 5196 train and 1717 test images in total. *LSTM-RGBD* is the best performing network, slightly better than *Conv-RGBD* as shown in Table 5.4. Sample segmentations are displayed on Figure 5.7 and 5.8, where *LSTM-RGBD* generates finer semantic segmentations.

Network	Test Accuracy (%)
Conv-D	65.05
Conv-RGB	67.07
Conv-RGBD	74.28
LSTM-D	59.93
LSTM-RGB	66.03
LSTM-RGBD	**74.37**

Table 5.4: Results acquired on the *Robot@Home* dataset with the nonlinear depth encoding.

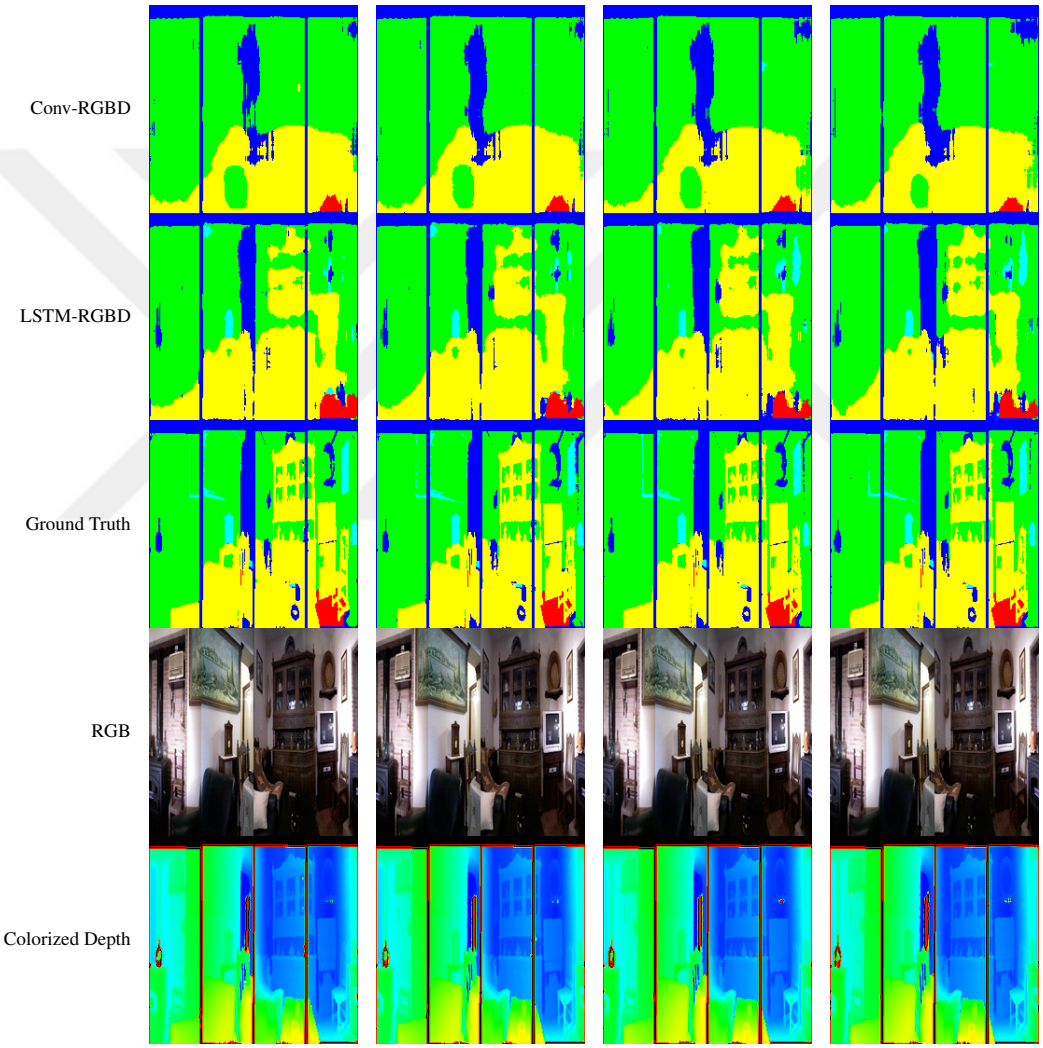

Figure 5.7: Sample semantic segmentations of the *Conv-RGBD* and *LSTM-RGBD* networks on the *Robot@Home* dataset. The last two rows are inputs to the networks at time t. The horizontal axis (from left to right) depicts time.

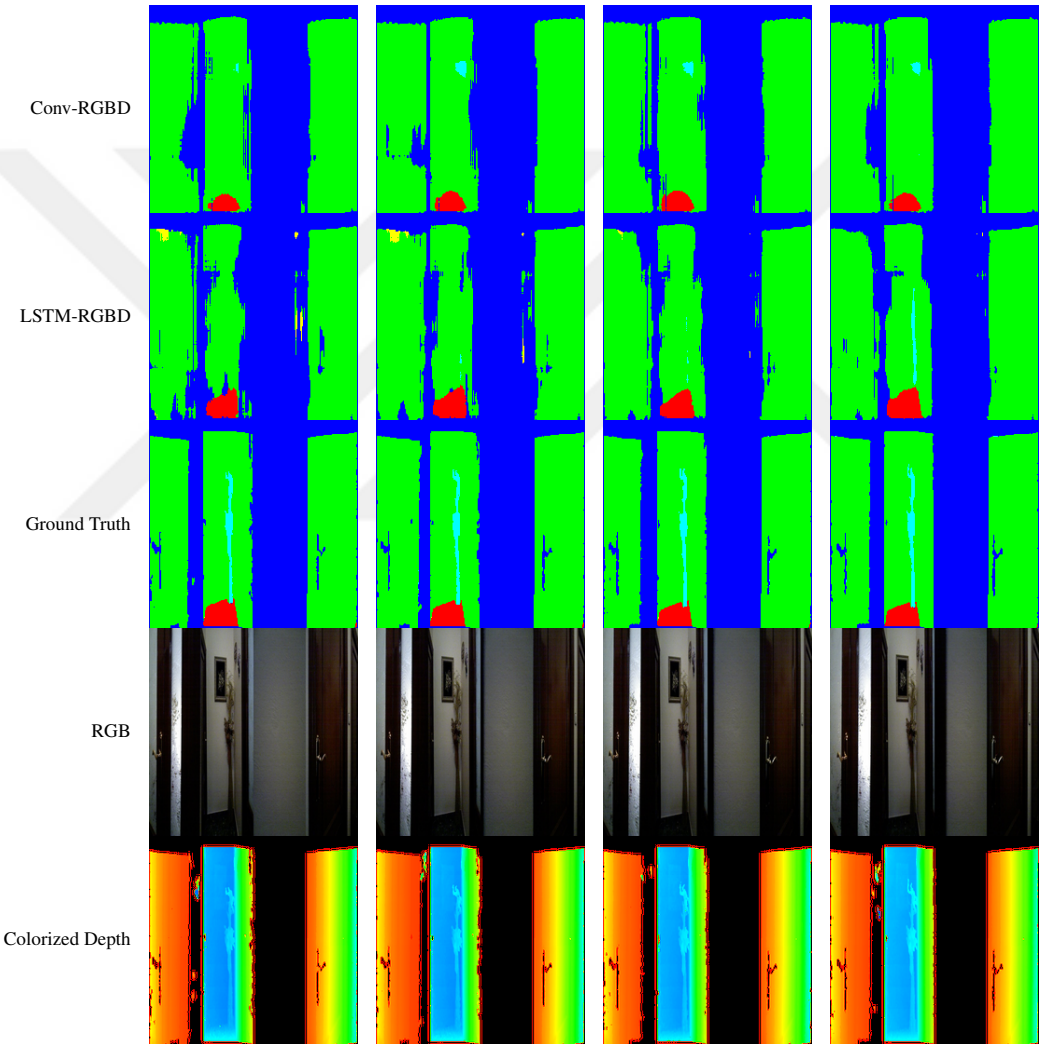

Figure 5.8: Sample semantic segmentations of the *Conv-RGBD* and *LSTM-RGBD* networks on the *Robot@Home* dataset. The last two rows are inputs to the networks at time t. The horizontal axis (from left to right) depicts time.

Chapter 6

IMPLEMENTATION

We implemented all of our networks in the programming language Julia [Bezanson et al., 2014] with the deep learning package Knet [Yuret, 2016], which provides GPU support for common operations used in the neural network field, such as convolution. We used a single NVIDIA K80 in a high performance cluster environment to train each network, which took about between 5 hours and 2.5 days for a single network, depending on the network configuration and amount of training data. At test time, our *Conv-RGBD* and *LSTM-RGBD* networks perform semantic segmentation at 18 and 13 FPS (frames per second) on average for a single frame respectively, as shown in Table 6.1. Our code, data and pretrained models are publicly available on GitHub at `https://github.com/ekyurdakul/MScThesis`.

Environment	Network	Average FPS
Laptop	Conv-D	19
	Conv-RGB	19
	Conv-RGBD	7
	LSTM-D	10
	LSTM-RGB	10
	LSTM-RGBD	5
HPC	Conv-D	47
	Conv-RGB	47
	Conv-RGBD	18
	LSTM-D	25
	LSTM-RGB	25
	LSTM-RGBD	13

Table 6.1: Performance of the *Conv-* and *LSTM-* networks at test time in terms of frames per second (FPS). The laptop has the following specifications: Intel Core i7 4700MQ, Nvidia GeForce GTX 860M, 8 GB RAM

Chapter 7

CONCLUSION

In this work, we explored the performance of recurrent fully convolutional neural networks in the domain of RGB and RGBD videos on the semantic segmentation task. Through experiments, we showed that fully convolutional neural networks possess the potential to benefit from depth information when combined with RGB data. In adding convolutional recurrent layers to fully convolutional neural networks, we observe further improvement in quantitative results and achieve better quality in visual semantic segmentation results. By transferring weights from fully convolutional neural networks to recurrent fully convolutional neural networks, we manage to reduce the training time of recurrent fully convolutional neural networks and also improve pixel-wise accuracy. Lastly, we observed that employing a nonlinear depth encoding scheme further increases performance.

A future goal is to explore the effects of increasing the number of time steps used in the backpropagation through time algorithm when training our recurrent fully convolutional neural networks. We believe this increase could further improve both quantitative and qualitative results, since the networks would have access to frames further back in time. Although the networks could exploit more temporal information, training the networks would become computationally more time consuming.

We also aim to improve the input resolution, however this would greatly increase computational complexity as more number of parameters would be required to train the networks effectively at a higher resolution. This increase in number of parameters in turn would require more GPU memory, which is a limited resource. Multiple GPUs could be used to overcome this memory limitation by distributing minibatches across GPUs, while keeping the other training parameters the same.

It is also possible to deepen the networks by adding more convolution or convolutional recurrent layers. Despite increasing computational complexity, it could enable the networks to learn more information and thus perform better on the semantic segmentation task. It would be interesting to observe the results of replacing every convolution layer with a convolutional recurrent layer, however this might not be computationally feasible, since even adding a single convolutional recurrent layer greatly increased training time compared to a network without any convolutional recurrent layers.

Although we used a simple method for colorized depth and RGB fusion, there exist many alternatives such as matrix concatenation and weighted matrix fusion. Whereas we fuse colorized depth and RGB before the deconvolution and convolutional recurrent layer for our fully convolutional and recurrent fully convolutional networks respectively, these modalities could be fused at different depths of the networks.

Lastly, various other transfer learning methods could be applied to depth from RGB, rather than applying a jet colormap, resulting in different expressions of the same depth data.

BIBLIOGRAPHY

[Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561.

[Bezanson et al., 2014] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2014). Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607.

[Chen et al., 2014] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

[Chung et al., 2014] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685.

[Couprie et al., 2013] Couprie, C., Farabet, C., Najman, L., and LeCun, Y. (2013). Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572.

[Deng et al., 2015] Deng, Z., Todorovic, S., and Latecki, L. J. (2015). Semantic segmentation of rgbd images with mutex constraints. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1733–1741.

[Eitel et al., 2015] Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M. A., and Burgard, W. (2015). Multimodal deep learning for robust RGB-D object recognition. *CoRR*, abs/1507.06821.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

[Gaidon et al., 2016] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. *CoRR*, abs/1605.06457.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*.

[Gupta et al., 2014] Gupta, S., Girshick, R. B., Arbelaez, P., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736.

[Hazirbas et al., 2016] Hazirbas, C., Ma, L., Domokos, C., and Cremers, D. (2016). Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *ACCV*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[He et al., 2016] He, Y., Chiu, W., Keuper, M., and Fritz, M. (2016). RGBD semantic segmentation using spatio-temporal data-driven pooling. *CoRR*, abs/1604.02388.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Li et al., 2013] Li, F., Kim, T., Humayun, A., Tsai, D., and Rehg, J. M. (2013). Video segmentation by tracking many figure-ground segments. In *ICCV*.

[Liang and Hu, 2015] Liang, M. and Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Liu et al., 2015] Liu, Z., Li, X., Luo, P., Loy, C., and Tang, X. (2015). Semantic image segmentation via deep parsing network. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Long et al., 2014] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038.

[Marmanis et al., 2016] Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., and Stilla, U. (2016). Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 473–480.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.

[Pavel et al., 2015] Pavel, M. S., Schulz, H., and Behnke, S. (2015). Recurrent convolutional neural networks for object-class segmentation of rgb-d video. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[Perazzi et al., 2016] Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L. V., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*.

[Pohlen et al., 2016] Pohlen, T., Hermans, A., Mathias, M., and Leibe, B. (2016). Full-resolution residual networks for semantic segmentation in street scenes. *CoRR*, abs/1611.08323.

[Ren and Zemel, 2016] Ren, M. and Zemel, R. S. (2016). End-to-end instance segmentation and counting with recurrent attention. *CoRR*, abs/1605.09410.

[Ros et al., 2016] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. (2016). The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*.

[Ruiz-Sarmiento et al., 2017] Ruiz-Sarmiento, J. R., Galindo, C., and González-Jiménez, J. (2017). Robot@home, a robotic dataset for semantic mapping of home environments. *International Journal of Robotics Research*.

[Russakovsky et al., 2014] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2014). Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575.

[Shi et al., 2015] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214.

[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

[Simonyan and Zisserman, 2014a] Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199.

[Simonyan and Zisserman, 2014b] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Song et al., 2015] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Tsogkas et al., 2015] Tsogkas, S., Kokkinos, I., Papandreou, G., and Vedaldi, A. (2015). Semantic part segmentation with deep learning. *CoRR*, abs/1505.02438.

[Valipour et al., 2016] Valipour, S., Siam, M., Jägersand, M., and Ray, N. (2016). Recurrent fully convolutional networks for video segmentation. *CoRR*, abs/1606.00487.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Wu et al., 2015] Wu, Z., Jiang, Y., Wang, X., Ye, H., Xue, X., and Wang, J. (2015). Fusing multi-stream deep networks for video classification. *CoRR*, abs/1509.06086.

[Xiao et al., 2013] Xiao, J., Owens, A., and Torralba, A. (2013). SUN3D: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1625–1632.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *CoRR*, abs/1411.1792.

[Yurdakul and Yemez, 2017] Yurdakul, E. E. and Yemez, Y. (2017). Semantic Segmentation of RGBD Videos with Recurrent Fully Convolutional Neural Networks. In *ICCV 2017 4th IEEE/ISPRS Joint Workshop on Multi-Sensor Fusion for Dynamic Scene Understanding*.

[Yuret, 2016] Yuret, D. (2016). Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS 2016*.