U-N.o.1T: A U-Net exploration, in Depth

John Luke Chuter, Geoffrey Boris Boullanger, Manuel Nieves Saez
{jchuter, gbakker, mnievess}@stanford.edu

December 18, 2018

Abstract

In this paper, we are exploring the generation of depth-maps from a sequence of images. Compared to similar projects in the field, we have decided to incorporate both spatial (CNN) and temporal (LSTM) aspects in our model, by creating convLSTM cells. These are used in a U-Net encoder-decoder architecture. The results indicate some potential in such an approach.

1. Introduction

Hardware progress has enabled solutions which were historically computationally intractable. This is particularly true in video analysis. This technological advance has opened a new frontier of problems. Within this expanse, we have chosen the classic problem of depth inference from images. Specifically, given a sequence of images captured over time, we output depth maps corresponding one-to-one with the input sequence. As a spatiotemporal problem, we were motivated to model it with convolutions (spatial) and LSTMs (temporal).

The input to our algorithm is a sequence of images. We then use a neural network U-Net encoder-decoder architecture, with bi-ConvLSTM cells for encoding and convolutions and transconvolutions to decode, to output a predicted depth map sequence. As we deal with sequences of images, this process is many-to-many, where for each input image we output one depth map. Solutions to the above problem would enable 3D world generation from simple video input with applications from VR to robotics. While there are hardware approaches to depth-determination problems, such as LIDAR or multiple lenses, software solutions provide flexibility in their application.

1.1. In Depth

After researching this initial problem in depth, we became familiar with literature on depth maps, their algorithms and datasets. This presented itself as a sensible path forward, as it seemed simpler and better scoped. This area is a classic one, with not only history but ongoing and recent progress. Concerning depth maps, there are various families of problems; single image to depth map, depth map alignments, from sparse to dense - but given the background research we'd done on the image+depth map sequence, we were naturally drawn to the most similar problem: from a sequence of images, generate a sequence of depth maps.

There are many reasons to be excited about such a problem, especially as the interest for spatiotemporal models is booming. For us, however, we wanted to learn about RNNs and CNNs, and as space-time lends itself to natural conceptions of convolutions and recurrent networks, we proceeded down that path.

Quite excited to apply modern RNN and CNN techniques, we were both disappointed and relieved to find extremely relevant literature: 'DepthNet' [3], 'Spatiotemporal Modeling for Crowd Counting in Videos' [19], 'Bidirectional Recurrent Convolutional Networks for Multi-Frame Super-Resolution' [9], 'Cross-scene Crowd Counting via Deep Convolutional Neural Networks' [20], and 'Pyramid Dilated Deeper ConvLSTM for Video Salient Object Detection' [16]. All these papers address spatiotemporal problems with RNNs and convolutions.

While there some people praise CNN to the detriment of RNN, we wanted to explore this avenue further. In pursuit of this approach we have our own opinion, as will be discussed at the end.

2. Related Work

It is fitting to begin with paper that introduced the core unit of our model, "Convolutional LSTM: A Machine Learning Approach for Precipitation Nowcasting" [15]. This paper details the convolutional LSTM cell, wherein a typical LSTM cell performs a convolution at its gates. This enables encoding of spatial information (from the convolution) while benefiting from the LSTM. The authors then detail stacking of such convLSTM layers, to create a deep convLSTM network for encoding. The next notable paper, "DepthNet" [3] presents the most similar model to our own. Specifically, they explore the combination of U-Net architecture with convLSTM layers in an encoder-decoder framework for purposes of depth estimation. Our variations from there explore how to implement bi-directionality, as a natural and common expansion to most LSTM models, which we detail in the following Model section. "Spatiotemporal Modeling for Crowd Counting in Videos" [19] demonstrates one method of implementing bidirectionality in a spatiotemporal setting. "Pyramid Dilated Deeper ConvLSTM for Video Salient Object Detection" [16] combines multiple advanced techniques, but tackles a highly different problem. In this realm, there were several closely related problems:

We chose DepthNet [3] as a baseline model to iterate from. First, a brief description of this baseline: 8 convLSTM layers are stacked in the encoding phase of U-Net encoder-decoder network. These provide connections and skip connections to the decoding phase, which is made of 4 convolutional and trans-convolutional pairs. For details we cannot do justice to here, refer to the DepthNet paper.
The DepthNet authors themselves propose several possibilities for alteration, and we came up with a few ourselves. Alternative models include: Implement an explainability mask to better predict depth maps for individual objects, Attention mechanism, or Bi-directionality. It was this third option we chose to explore, as in a network like this there are surprisingly many ways to try to incorporate the forward and backward passes. While this remains an object of experimentation, there are three principle categories of variation: Full-communication, Sparse-communication, Mediation. We chose full-communication between a left and right pass over the input image sequence.

There are many great people and great ideas, [22] [9] [4] [2] [11] [10] [14] [7] [18] [8] [1] [17] [12] [5] but we have continued to develop our own.

3. Dataset and Features

3.1. Descriptive Overview

In the search for a dataset with both picture and depth map, we have decided to use the KITTI dataset [6], which was originally created from a Volkswagen station wagon for use in mobile robotics and autonomous driving research. A few hours of traffic scenarios have been recorded, using various sensors like a high resolution color camera or a Velodyne 3D laser scanner. Even if our project is on a different subject compared to the dataset's target audience, we were attracted by the large amount of recordings of paired video/depth map that the KITTI dataset offered. We are not using the other measurements provided by KITTI - e.g GPS, timestamp, etc.
The features for an image and depthmap pair are the pixels therein; i.e. the RGB values and depth values. These depthmap groundtruths are generated with LIDAR.
We are using the full raw dataset from KITTI containing 180GB worth of data, divided into categories: Road, City, Residential, Campus and Person. As we are conscious it would be impractical on either of the two machines we are using for training, we had to reduce the amount of data we would use.

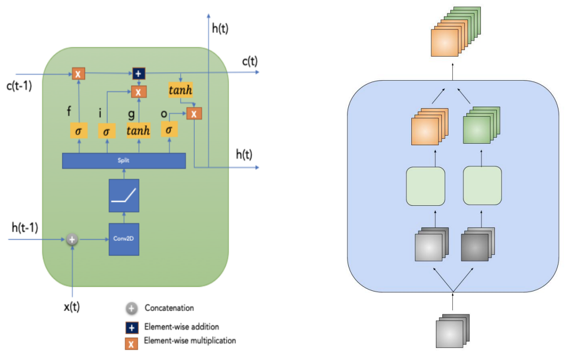

Figure 1. a. Conv-LSTM b. bi-ConvLSTM Cell

3.2. Preprocessing

First, we organized the data and store image sequences in subfolders as it seems to simplify and speed up the training [22]. Second, we reduced the quality of the images so that we could run the best model on our GPUs without needing extremely small batch sizes (i.e. 64x18px). Finally, we split our dataset into train, valid and test sub-datasets. These train (c. 35,000 images), valid/test (c. 10,000 images) sets are optimally preselected by KITTI and allow comparability to various models applied to this dataset. The two first were used to try different versions of models and calibrate the hyper-parameters for the most successful model, while the test set will only be used once, at the end, in order to report the performance of our best model on the final report. We created a bespoke data loader due to the unusual nature of our dataset (i.e. images stored by sequences in subfolders and depth maps linked to the sequence). This dataloader includes preprocessing such as Unity Normalization transformation, to quicken training. We experimented with cropping and random flipping, but such data augmentation techniques proved counterproductive, as we have a surplus of data for our computational means, and actively made a sequence of data inconsistent through time, working against the LSTM. Now on to the methods of what was to be trained, and how.

4. Methods

4.1. ConvLSTM, bi-ConvLSTM

ConvLSTMs are more than a convolutional layer into an LSTM layer; they convolve on the hidden state and the input together. This functional difference has led to some speculation as to the merits of one over the other, where convLSTMs sometimes prove more effective (as in *Very Deep Convolutional Networks for End-to-End Speech Recognition* [21]. We were curious to explore the more recent development of convLSTMS. Additionally, DepthNet achieved good results with convLSTMs, which indicated potential.

The specific math for a ConvLSTM is:

$$\begin{aligned}
i_t &= \sigma(ReLU(W_{xi} * X_t + W_{hi} * H_{t1} + W_{ci} \circ C_{t1} + b_i)) \\
f_t &= \sigma(ReLU(W_{xf} * X_t + W_{hf} * H_{t1} + W_{cf} \circ C_{t1} + b_f)) \\
g_t &= tanh(ReLU((W_{xg} * X_t + W_{hg} * H_{t1} + b_g)) \\
C_t &= f_t \circ C_{t1} + i_t \circ g_t \\
o_t &= \sigma(ReLU(W_{xo} * X_t + W_{ho} * H_{t1} + W_{co} \circ C_t + b_o)) \\
H_t &= o_t \circ tanh(C_t)
\end{aligned}$$

where $*$ refers to a convolution operation and $\circ$ to the Hadamard product.

4.2. Architecture - U-Net Encoder-Decoder

We picked this model to optimize for simplicity of approach, while maintain sophistication of the model's capacity. The U-Net structure is symmetrical and conceptually simple, and the main complexity is within its subcomponent bi-ConvLSTM. This subcomponent could be tinkered without alteration to the whole.
The inputs to our network are 5D tensors, *(B, N, C, H, W)*, where *B* refers to batch size, *N* to sequence length, *C* to channels, *H* to height and *W* to width. Per layer, the number of filters, and therefore output channels of that layer increase during the encoding phase (starting from 3; RGB), and decrease during the decoding phase (finishing at 1; depth). We use relu activation functions for each encoding layer, and at the last step of the decoding phase. Skip connections in the U-Net structure pass forward outputs to later layers, concatenating with the output of the directly previous layer. See figures 2,3 for greater details.

5. Experiments/Results/Discussion

5.1. Experiments

For this project, we are using two separate machines with both a recent NVidia GPU (1080Ti and P100). Our current implementation of the model uses Pytorch 0.4.1 [13] and Cuda 9.1. We have run various models for nontrivial hours, to test different sequence lengths (1,3,6), different image sizes (from 416x128 pixels, then 128x36, to 64x18). The final model was trained for 10 hours on 64x18 images.

5.2. Metrics

We used multiple metrics for training and evaluation purposes, based on properties distinct to each function. Specifically, RMSE, iRMSE, MAE, iMAE, and a custom scale invariant loss.
RMSE

$$L(y, y^*) = \sqrt{\frac{\sum_{t=1}^{n}(y - y^*)^2}{n}}$$

MAE

$$L(y, y^*) = \frac{1}{N}\sum_{t=1}^{N}|y_i^* - y_i|$$

Custom Loss

$$L(y, y^*) = \frac{1}{n}\sum_i d_i^2 - \frac{\lambda}{n^2}\left(\sum_i d_i\right)^2$$

where $d_i = \log(y_i)\log(y_i^*)$ for the i'th pixel, n the number of pixels, and $\lambda = 0.5$.

Succinctly, RMSE is standard and easy to implement, while providing comparison against other models. It is distinct from MAE, in that RMSE significantly larger than MAE indicates large variance of error distribution frequency.

Finally, A1, A2, and A3 metrics are *accuracies* that represent the percent of pixels that fall within a threshold ratio of inferred depth value, and ground truth depth value. a refers to a base, and 1,2,3 refer to powers of that base; ie A3 is the most lenient and A1 the strictest. These accuracies are independent of image size, and therefore ideal for baseline comparison. Also, whereas losses provide an unintuitive metric of "goodness" and progress, accuracy is more comprehensible. Multiple a-values indicate the distribution of inferences.

5.3. Baseline Measures

We are comparing ourselves most directly to DepthNet and other KITTI competitors, with the corresponding loss measures. The current two leaders are DL-61(DORN) and DL-SORD-SQ :

	SILog	**sqErrorRel**	**absErrorRel**	**iRMSE**
DL-61(DORN)	11.77%	2.23%	8.78%	12.98%
DL-SORD-SQ	13.00%	2.95%	10.38%	13.78%

5.4. Our Results

	SILog	**RMSE**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
6 Image Sequence	0.44	6.37	0.74	0.87	0.93
1 Image Sequence	0.54	7.4	0.705	0.84	0.91

5.5. Discussion

Comparing sequences of length 1 to 6, we see that the 6 outperformed the 1 on every metric. This implies that the LSTM does provide utility to analysis, and that over-time information holds clues to depth. This somewhat justifies the theory behind the model, and is consistent with the results of previous teams, such as DepthNet.

6. Conclusion and Future Work

We presented a different approach to image to depth-map implementation. Using jointly a U-Net architecture and convLSTM cells, we tried to incorporate both spacial

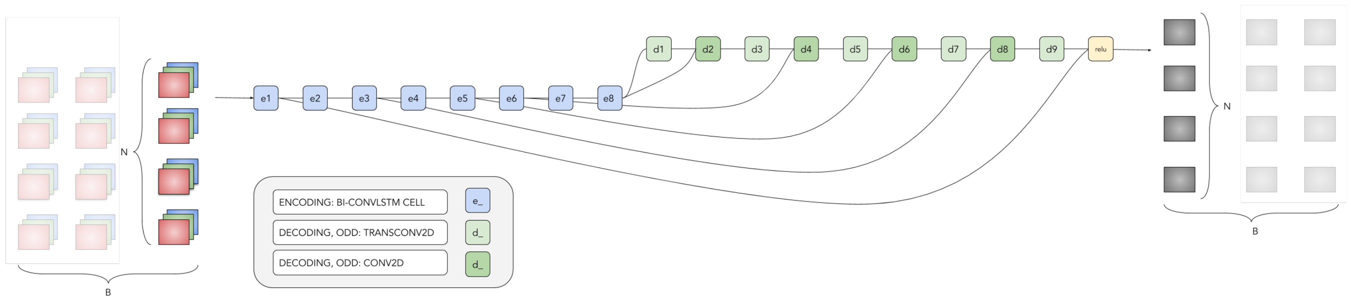

Figure 2. UNoIT Detail

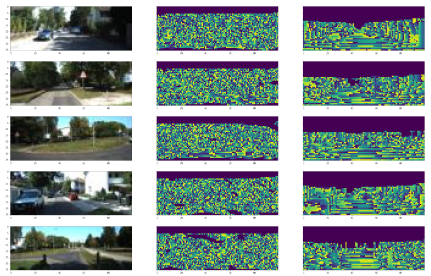

Figure 3. x (LHS), y*, y (RHS)

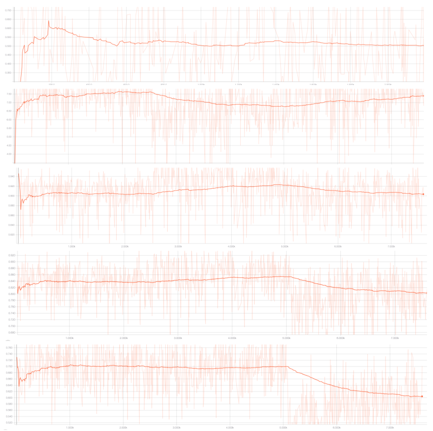

Figure 4. Sequence-1: SILog, RMSE, a1, a2, a3 (from top to bottom)

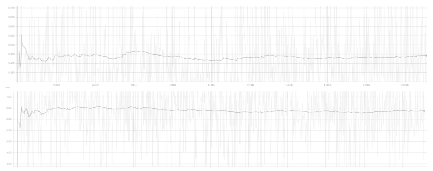

Figure 5. Sequence-6: SILog, RMSE (from top to bottom)

and temporal elements to insure consistency when generating depth-maps for sequences of images - i.e. video.

There are several areas to continue our work here. We'd first like to train for an extra week and see if we continue to progress towards convergence. First, increasing sequence length. While we limited ourselves to a length of 6, we are curious as to the impact a sequence of 600 would compare. Second, data processing. There are many transformation alternatives to be played with. We especially would like to train on bigger image sizes, if we had more time and compute. There are also multiple ways to iterate over the data As for loss functions, we used many of them for evaluation, and it'd be interesting to explore if any of those is better than our custom loss for guiding the gradient and training. Beyond that, we would like to play with kernel size, and the number of filters per layer, as there are interesting questions in the optimal number per layer. Expanding in creativity, how would increasing the number of encoding-decoding layers affect performance? We did not nearly approach overfitting problems. More dramatically, what are the effects of U-Net? If we were to remove the skip connections, how would we perform? Other advanced techniques invite exploration. Pyramids for convolutions, attention for LSTM; drop the RNN and just use a 3D convolution; perhaps use a 3D convolution inside a convLSTM (multiple timesteps to each lstm "timestep"). There are many possibilities.

7. Appendices

Contributions

While everyone has done some of everything, the largest contributions from each to any particulars may be described as follows. John has researched the literature, designed the models, and assisted Manuel with several of the models and metrics. Manuel implemented the final model and metrics, and ran multiple training runs. Geoffrey has set up the infrastructure for preprocessing data with various transforms, saving and loading models, and worked on metrics.

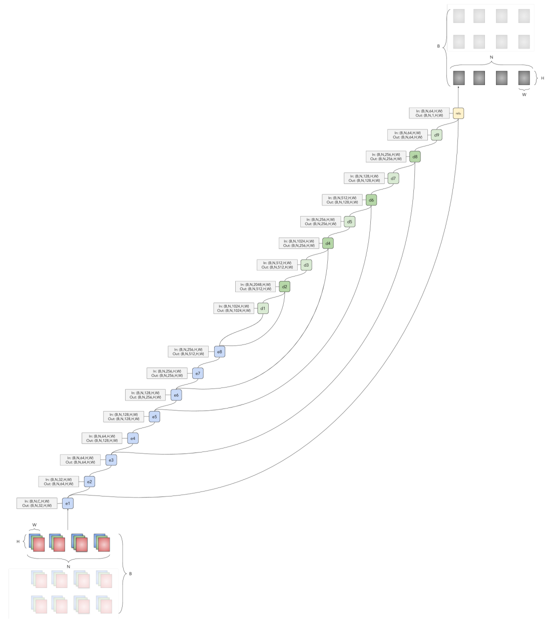

Figure 6. U-Net Encoder-Decoder

References

[1] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[2] J. T. Barron and B. Poole. The fast bilateral solver. *CoRR*, abs/1511.03296, 2015.

[3] A. CS Kumar, S. M. Bhandarkar, and M. Prasad. Depthnet: A recurrent neural network architecture for monocular depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[4] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014.

[5] M. Elbayad, L. Besacier, and J. Verbeek. Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction. *ArXiv e-prints*, Aug. 2018.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016.

[8] L. He, G. Wang, and Z. Hu. Learning depth from single images with deep neural network embedding focal length. *CoRR*, abs/1803.10039, 2018.

[9] Y. Huang, W. Wang, and L. Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 235–243. Curran Associates, Inc., 2015.

[10] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. *CoRR*, abs/1606.00373, 2016.

[11] W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104, 2016.

[12] F. Ma, G. Venturelli Cavalheiro, and S. Karaman. Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera. *ArXiv e-prints*, July 2018.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[14] G. Riegler, M. Rüther, and H. Bischof. Atgv-net: Accurate depth super-resolution. *CoRR*, abs/1607.07988, 2016.

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

[16] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam. Pyramid dilated deeper convlstm for video salient object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[17] R. Wang, J. Frahm, and S. M. Pizer. Recurrent neural network for learning densedepth and ego-motion from video. *CoRR*, abs/1805.06558, 2018.

[18] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. *CoRR*, abs/1709.08429, 2017.

[19] F. Xiong, X. Shi, and D. Yeung. Spatiotemporal modeling for crowd counting in videos. *CoRR*, abs/1707.07890, 2017.

[20] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 833–841, June 2015.

[21] Y. Zhang, W. Chan, and N. Jaitly. Very deep convolutional networks for end-to-end speech recognition, 2018.

[22] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.