Published at International Joint Conference on Neural Networks (IJCNN), 2015

Recurrent Convolutional Neural Networks for Object-Class Segmentation of RGB-D Video

Mircea Serban Pavel, Hannes Schulz, and Sven Behnke
Universität Bonn
Computer Science Institute VI
Friedrich-Ebert-Allee 144
53113 Bonn
pavel@cs.uni-bonn.de, schulzh@ais.uni-bonn.de, behnke@cs.uni-bonn.de

***Abstract*—Object-class segmentation is a computer vision task which requires labeling each pixel of an image with the class of the object it belongs to. Deep convolutional neural networks (DNN) are able to learn and exploit local spatial correlations required for this task. They are, however, restricted by their small, fixed-sized filters, which limits their ability to learn long-range dependencies. Recurrent Neural Networks (RNN), on the other hand, do not suffer from this restriction. Their iterative interpretation allows them to model long-range dependencies by propagating activity. This property might be especially useful when labeling video sequences, where both spatial and temporal long-range dependencies occur. In this work, we propose novel RNN architectures for object-class segmentation. We investigate three ways to consider past and future context in the prediction process by comparing networks that process the frames one by one with networks that have access to the whole sequence. We evaluate our models on the challenging NYU Depth v2 dataset for object-class segmentation and obtain competitive results.**

I. Introduction

Current deep neural network architectures achieve superior performance on a number of computer vision tasks, such as image classification, object detection and object-class segmentation. Most of these tasks focus on extracting information from a single image. Deep neural networks compute increasingly abstract features, which simultaneously become more and more semantically meaningful, and incorporate larger contexts.

A real-world vision system will have to deal with the time dimension as well. Content is increasingly generated in the form of videos by Internet users, surveillance cameras, cars, or mobile robots. Video information can be helpful, as looking at a whole sequence instead of single frames may enable the interpretation of ambiguous measurements.

Similar to increasingly abstract features on images, we are interested in neural networks which produce high-level features on sequences. In a recursive computation, these high-level features should help to interpret the next frame in a sequence. In addition to a semantically meaningful localized content description, such features should form high-level descriptions of motions with increasing temporal context.

In this paper, we introduce and compare recurrent convolutional neural network architectures which produce high-level localized sequence features. We evaluate them on the NYU Depth v2 dataset, an RGB-D object-class segmentation task, where every pixel of the input image must be labeled with the category of the object it belongs to. In this challenging and established benchmark, most methods focus on prediction based on single frames, while our method exploits image sequences.

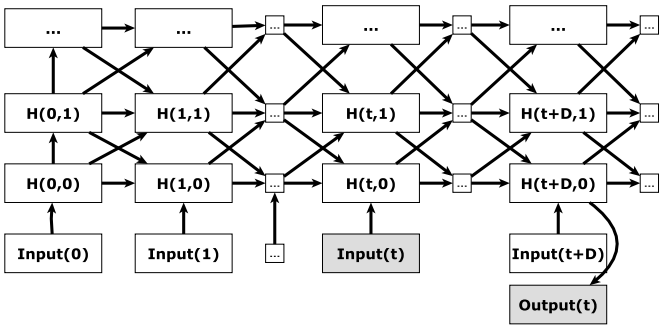


Fig. 1: Architecture of the unidirectional RNN. The layers are connected to each other with valid convolutions. Upward (*forward*) connections additionally include spatial max-pooling operations, while downward (*backward*) connections include a spatial upsampling. Delay D is number of time frames between an input and corresponding output frame.

In short, our contributions are as follows:

- We introduce three recurrent neural network models for processing image sequences.
- On toy datasets, we show that our recurrent models are able to keep an abstract state over time, track and interpret motion, and retain uncertainty.
- We show that our model can reach competitive results on RGB-D object class segmentation on the challenging NYU Depth v2 dataset.
- We compare the introduced models and find that unidirectional models with long sequences produce the best results.

The remainder of this paper is organized as follows. Section II introduces our architecture. In Section III, we discuss related work. Network training is described in Sec. IV. We evaluate our model in Section V, and discuss the results in Section VI.

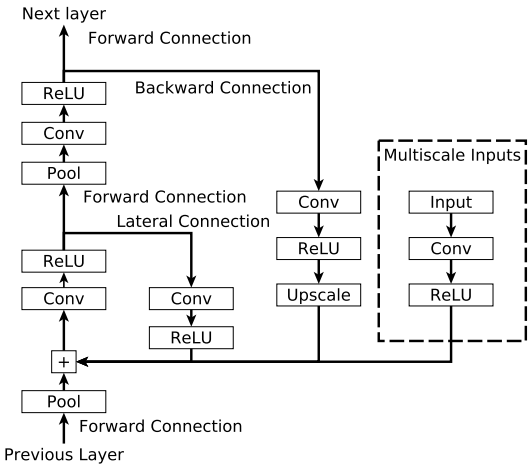


Fig. 2: Recurrent connections as viewed from a single hidden layer. Activations of a hidden layer are the result of forward connections from below, backward connections from above, and lateral connections from the same layer at previous time steps. Inputs may be provided at all scales.

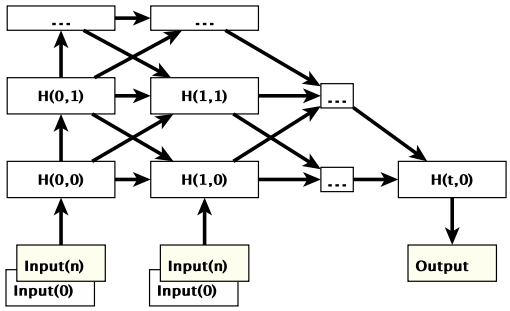


Fig. 3: Architecture of the simplified RNN. All time steps of the network have access to the whole image sequence.

II. Recurrent Convolutional Neural Networks for Image Labelling

In this section, we present our network architectures. All network architectures are inspired by the Neural Abstraction Pyramid of Behnke [1], a hierarchical recurrent neural network architecture for image processing. A schematic overview is shown in Figures 1 and 2. We use convolutional neural networks (CNN [2]), which retain the topological image structure and ensure localization of features. Our base configuration (without the ability to process sequences) contains $L = 3$ layers with 32 maps each, ReLU non-linearities, and interleaved spatial max-pooling (c.f. Hinton *et al.* [3]).

A. Connection Types

To process sequences, we replicate our model for T time steps and introduce connections between the temporal copies. Three types of connections exist: forward, lateral, and backward. Computationally, all connections are valid convolution operations followed by a half-rectifying point-wise non-linearity $f(x) = \max(0, x)$.

A hidden layer $H(t, l)$, at time step t and abstraction level l, is connected to layer $H(t+1, l+1)$ using a forward connection. These connections allow the vertical flow of information from the bottom of the network to the top and thus the construction of high level features based on the low level features. The non-linearity of the forward connections are followed by a spatial 2×2 maximum pooling operation with stride 2.

Lateral connections connect layers $H(t, l)$ and $H(t+1, l)$. These horizontal connections can incorporate immediate spatial context from the same activation level. The intermediate context is limited by the receptive field size of the convolution filters.

Backward connections connect layer $H(t, l)$ to layer $H(t+1, l-1)$, and can be interpreted as providing a high-level prior for the lower, more detailed layers. Since higher layers have a coarser spatial resolution, they also provide a convenient shortcut for information that needs to travel long distances. Backward connections are immediately followed by a spatial upsampling operation.

Due to padded convolutions and the opposing pooling and upsampling operations, all connections coinciding on a given hidden layer have the same size, and are simply summed elementwise.

All connections use temporal weight sharing, i.e. for all t and all $k \in \{-1, 0, 1\}$, the weights used in the convolution from $H(t, l)$ to $H(t+1, l+k)$ are identical across time steps.

B. Output

In contrast to common CNN models, the output of our network is always obtained in the lowest layer of the network, at the first time step which ensures that the activations were able to reach the highest layer and return back. This structural property allows us to produce detailed outputs at input resolution. The cross-entropy loss over the C object categories is measured on a subset of the maps in the lowest layer, where every map is responsible for one category. Apart from a simplified implementation, this allows the network to produce and reuse intermediate outputs, and refine them over time.

C. Multi-Scale Inputs

Inputs may be given on all scales of the network. When using multi-scale inputs, we additionally convolve a (downscaled) version of the input and add it to the result.

D. Unidirectional RNN

In the architecture described so far, we process the video images sequentially, one image per time step. The state (i.e., the activations) at time t containing information about its past is combined with the image at time t, producing an output and a new state. Since the last output benefits from learning from the whole sequence, it is natural to place the frame that we want to evaluate at the end.

The first temporal copy is special, since it contains regular feed forward connections. This allows us to produce activations in each layer such that all connection types can be used in the transition from t to $t + 1$.

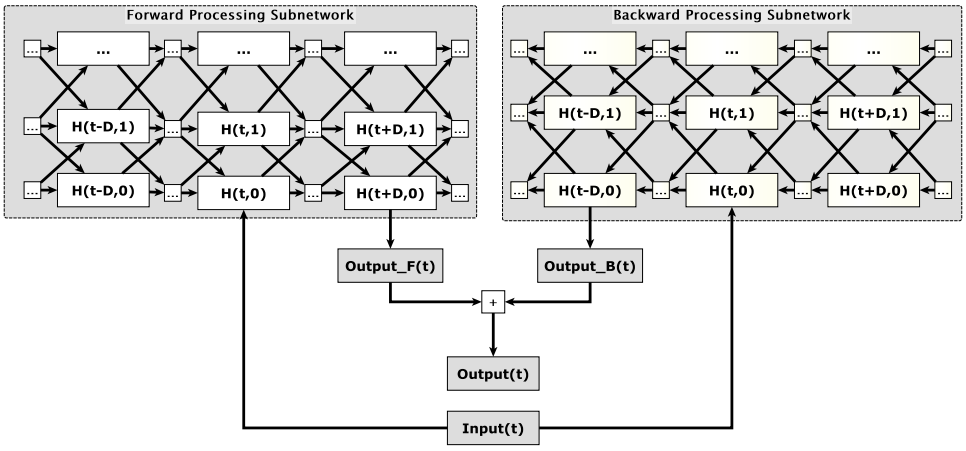


Fig. 4: Architecture of the bidirectional RNN. The final output at the center has access to both past and future context. Delay D is number of time frames between an input and corresponding output frame.

When processing input at time t, we allow $L-1$ time steps for the information to reach the top level of the network and the same amount for propagating back to the bottom layer, where the output corresponding to time t is evaluated. Note that the last temporal steps do not need all the hidden layers, since their activation would no longer propagate to the output.

E. Simplified RNN

In this architecture, shown in Figure 3, we concatenate inputs from all time steps and provide the whole sequence to the network at every time step. This is a simplified version of our first architecture, since the necessity of exploiting temporal dependencies is reduced. On the other hand, spatial information can still be aggregated over time, since we keep temporal connections and the pyramid structure of the network. Our intent for doing so is to demonstrate that learning from stacks of time frames is inefficient and that limiting the processing to a single image per time step does not result in a performance decline.

F. Bidirectional RNN

The third architecture, shown in Figure 4, is inspired by the bidirectional recurrent network [4], [5]. Here, connections exist in both directions of the temporal dimension. This network is able to exploit not only the past context, but also the future context. In a real-time setting, of course, this would require waiting for all the future frames to become available in order to produce the result for the current one. We implement this concept using two unidirectional networks, one in forward and one in backward direction. We combine their outputs in order to obtain the final output of the network. To keep the amount of context between the models constant, we use half of the context used by the other models in both directions.

III. Related Work

Several groups have used neural networks to process image sequences. Most works use stacks of frames to provide time context to the neural network. Le *et al.* [6] and Taylor *et al.* [7] learn hierarchical spatio-temporal features on video sequences using Gated Convolutional Restricted Boltzmann Machines (convGRBM) and Independent Subspace Analysis (ISA), respectively. Their image features are not learned discriminatively and the models do not allow localized predictions.

Simonyan and Zisserman [8] use a two-stream architecture for action recognition, which separately creates high-level features from single-frame content and motion. Again, motion is provided through a stack of optical flow images, so that the modeled complexity is limited by the stack size. In our paper, we find that increasing temporal context by providing frames consecutively yields improved performance.

More recently, Michalski *et al.* [9] introduced a model designed to explicitly represent and predict the movement of entities in images. The architecture is chosen in a way that higher layers represent invariances of derivatives of positions (motions, accelerations, jerk). Our models do not explicitly model motion. However, our models can make use of deep layers even in the case no high-level position invariances exist, since in addition to motion, they also encode static content. Furthermore, in our model, deep layers have a lower resolution and facilitate transport of information across longer distances.

Jung *et al.* [10] introduce a multiple timescale recurrent neural network for action recognition, which uses neurons with fixed time constants. The model uses leaky integrator neurons, which limits the rate at which higher layer activations can change. It is trained and evaluated on a simplified version of the Weizmann Human Action Recognition dataset.

Various architectures for processing video data are explored by Karpathy *et al.* [11]. The architecture most similar to our best model, *slow fusion*, uses weight sharing between time steps and merges them in higher layers. In their study, *slow fusion* yields best results. In contrast to classifying video sequences with a single label, we produce label output at pixel level.

Recurrent neural networks were successfully used for object-class segmentation by Graves [4] and Pinheiro and Collobert [12]. Both works use recurrence only to increase

spatial context, whereas we extend processing to the temporal domain.

Long-Short Term Memory (LSTM) units are capable of carrying information, at the original resolution, over long temporal distances. This is especially useful for tasks such as speech recognition [5] or language understanding [13], where e.g. a specific property of a distant word or sound might influence the semantics of the current context. In this paper, we opt for simple neural units instead. While we are also interested in learning long-range dependencies, we do not provide spatial or temporal context at the original resolution. Instead, our architecture maintains expressive low resolution context information in higher layers. This is more realistic for natural images, where correlations are stronger between nearby pixels than those between distant ones. Pascanu *et al.* [14] suggest that LSTM also addresses the problem of vanishing gradients. Here, we use RMSProp as gradient method, which in addition to preventing vanishing gradients also counteracts gradient explosion.

Our architecture choices are motivated by the Neural Abstraction Pyramid of Behnke [1], which performs pixel-wise classification tasks at input resolutions as well. In contrast to our work, Behnke did not train on video sequences, but only on stationary patterns, which in some cases were corrupted by temporally changing noise. We also include modern architectural features, such as max-pooling and ReLU non-linearities, and use RMSProp to increase learning speed.

IV. Learning

We initialize the weights biases from a Gaussian distribution. It is important to ensure that the activations do not explode or vanish early during training. Ideally, activations in the first forward pass should have similar magnitudes. This is difficult to control, however. Instead, we choose the variance of the weights and the mean of the bias such that the average of the activations in every point of our network is positive and slightly decreasing over time.

We learn the parameters of our network using Root Mean Square Propagation (RMSProp), which is a variant of Resilient Propagation (RProp) suitable for mini-batch learning [15]. RProp considers only the sign of the gradient, thus being robust against vanishing and exploding gradients, phenomena that occur when training recurrent neural networks.

We apply dropout [3] during learning, which we found improves our results in almost all cases. Applying dropout in RNNs is delicate, however. It should not affect recurrent connections, which would otherwise loose their ability to learn long-range dependencies [16]. Thus, when using dropout, we apply it to add a final convolution applied to the bottom layer to extract the output.

V. Experiments

We first conduct experiments on handcrafted datasets, which allow us to demonstrate important characteristics of our model. In a second step, we use our architectures for object-class segmentation on a challenging RGB-D dataset.

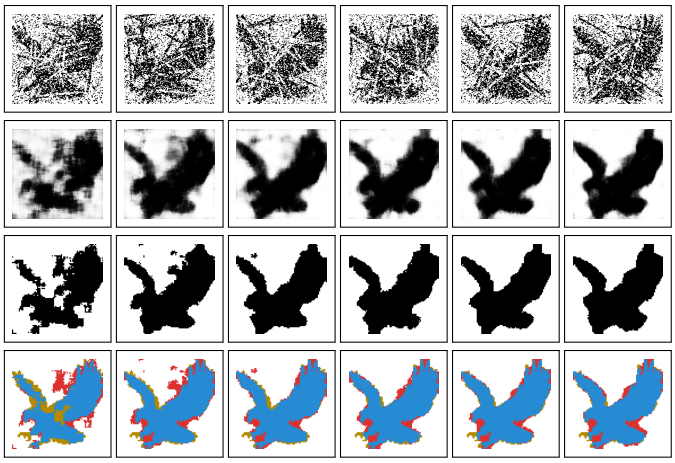


Fig. 5: Toy Experiment: Denoising. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network, the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The last output represents the final output of the network and we use it for evaluation.

A. Toy Experiments

We present three toy experiments, showing that our network is able to learn 1) filtering noisy information over time, 2) tracking and interpreting motion, and 3) retaining an internal state including uncertainty.

1) Denoising: In this experiment, we simply feed different degraded versions of the same binary image to the network. We use salt and pepper noise, uniformly distributed over the whole image. We also draw random black or white lines, to make the task more difficult. The task is to obtain the original image without noise. One way the network could solve this task would be to learn to average the image over time. In addition, denoising filters learned by the neural network can remove high frequency noise.

To ensure that the network is able to generalize instead of learning an input by heart, we use different objects for training, validation and testing. Every split contains 100 independently generated sequences.

Since the task has a reduced complexity, we opt for a simple convolutional model of only one hidden layer with 32 maps. A small filter size of 5×5 provides sufficient spacial context. There is no specific order in such a sequence of noised images, thus we only test the unidirectional architecture on this task.

We use $T=6$ temporal copies. During training, we optimize a weighted sum of the losses at all time steps, with a ten times larger weight placed on the final output. In all toy examples, we train for 12,000 iterations with minibatches of size 16.

Figure 5 shows an example from the test set. Our model is able to improve its prediction step by step, accumulating over time information even from the areas which are more affected by noise. After only two steps, the network is able to remove most of the false positives and to assemble together almost all features of the object.

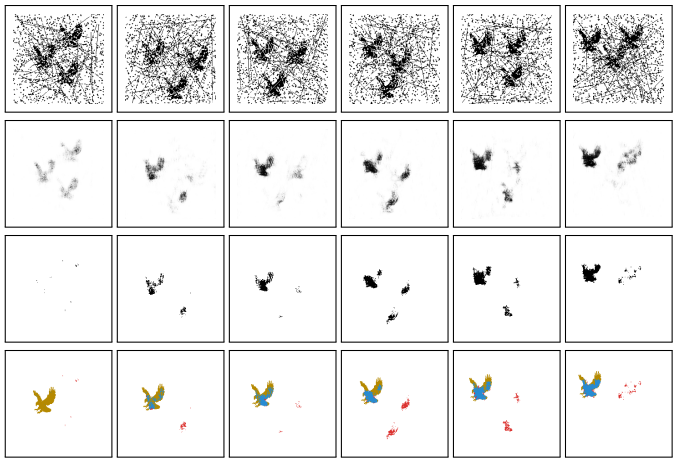


Fig. 6: Toy Experiment: Detecting movement. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network and the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The last output represents the final output of the network and we use it for evaluation.

We also train a convolutional model without recurrent connections (not shown). Here the result is less clear, the model only reduces noise in areas where the structure of the object is clearly visible, resulting in a classification accuracy of 84.7% network compared to 93.2% of the recurrent model.

2) Detecting Movement: In this experiment, we test the capabilities of the network to track a foreground object while the object is moving with constant speed through a noisy image. To ensure that motion is the cue for tracking, we add two randomly placed distractor objects of the same shape and size in a random position at every time step. These distractors should be classified as background. To prevent the network from overfitting on motion direction and speed, we generate several sequences, each moving the object from a random position to another, with varying speed.

Figure 6 shows the results obtained on this task using the unidirectional network. In the first time step, the network cannot decide which object is moving continuously. Already at $t = 2$, however, the network detects a slight positional change from one of the objects, while the others are further away from their initial position. The softmax layer activations show that the certainty of the hypothesis increases step by step. Also, one can notice that more details are added to the representation. Some false positives still exist when the new random position of an distractor object is close to its former position.

3) Retaining Uncertainty: While in the previous experiments, we showed that the network is able to track a moving object, we now consider if a regular movement can be inferred from temporally distant information. We use a bi-directional model and provide only the first and the last input, so that the initial positions have to be remembered until information from the past and future converges at the center time step. Since denoising is not an essential component of this task, we do not add noise.

Figure 7 depicts a sample sequence from the test set. As

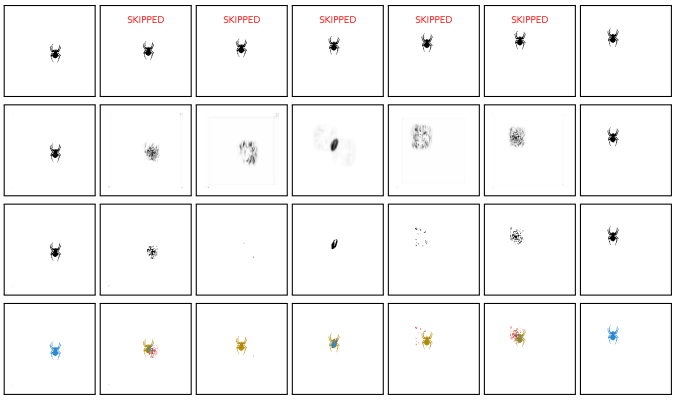


Fig. 7: Toy experiment: Retaining uncertainty. Rows represent, in order: the RGB input of the network for each time-step, the output of the softmax layer, the final outputs of the network and the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives). The middle output represents the final output of the network and we use it for evaluation.

no motion information is provided, the best strategy of the network is to create a circular expanding hypothesis from the seen location, which would then collapse as both timelines are combined. This is what we observe in the output maps of Figure 7. While the position is correctly identified, the shape of the object is largely lost.

B. RGB-D Object-Class Segmentation

The NYU-Depth v2 (NYUD [17]) dataset is comprised of video sequences taken from 464 indoor scenes, annotated with a total of 894 categories. We use the popular relabeling into four high-level semantic categories, small objects that can be easily carried ("prop"), large objects that cannot be easily carried ("furniture"), non-floor parts of the room: walls, ceiling, columns ("structure"), and the floor of the room ("ground").

Although the NYUD dataset was recorded as a video sequence, the actual dataset consists of a subset of 1449 frames which were preprocessed and manually labeled. The remainder of 407,024 frames are the raw output of the RGB-D cameras.

To transform the dataset into an image sequence dataset, but at the same time use the labeled frames for evaluation, we extracted the past and future context of each labeled frame from the video stream and preprocessed it. For evaluation, we compare the output corresponding to the labeled frame with the ground truth, retaining the same training/testing split as in the literature.

The temporal distance between frames should be kept short enough to ensure that the translation stays within the 7×7 filter size, such that each abstraction layer of the network can track the changes between two frames. Although the speed of the camera movement may vary, a fixed interval of $0.1\,\mathrm{s}$ between frames allows us to both have a smooth transition, but also to cover a significant time span. It is necessary to synchronize the RGB and depth frames, since the RGB and the depth sensors work independently of each other.

TABLE I: NYUD dataset test results of the best models for our three architectures. The unidirectional network was additionally run with multi-scale inputs (MS), slightly improving the results.

Method	Average Accuracy (%)	
	Class	Pixel
Simplified Network	59.2	60.0
Bidirectional Network	62.2	62.5
Unidirectional Network	62.3	62.7
Unidirectional Network + MS	62.4	63.1

TABLE II: Comparison with two non-recurrent CNN models with similar architecture. Recurrent networks perform better than non-recurrent architectures for both multi-scale input (MS) and depth-normalized sliding windows (SW).

Method	Class Accuracies (%)				Average (%)	
	ground	struct	furnit	prop	Class	Pixel
Höft *et al.* [19]	77.9	65.4	55.9	49.9	62.0	61.1
Unidirectional + MS	73.4	66.8	**60.3**	49.2	62.4	63.1
Schulz *et al.* [20] (no height)	87.7	70.8	57.0	53.6	67.3	65.5
Unidirectional + SW	**90.0**	**76.3**	52.1	**61.2**	**69.9**	**67.5**

The next step is the preprocessing of the RGB and depth images. Here, we follow standard procedure, sequentially applying lens correction, projecting the depth onto the RGB sensor perspective and filling-in the depth. Lens correction attempts to fix the barrel distortion typical to wide angle cameras and is computed by the "Camera Calibration and 3D Reconstruction" package of OpenCV with the camera parameters provided by the creators of the dataset. Depth and RGB measurements are not taken from the same viewpoint. Thus, the depth is projected to the RGB camera viewpoint. This is done by a rotation and a translation using the matrices provided by the creators of the dataset. The final step is to fill-in the missing depth measurements, which are caused by certain material properties (e.g. black or shiny surfaces) or by occlusions. We fill in missing depth values based on color information using the colorization algorithm of Levin *et al.* [18].

Finally, we produce ground truth for unlabeled frames by propagating labels along the optical flow direction. We use OpenCV to compute optical flow on RGB image pairs and write special *ignore* labels when label information is unavailable or ambiguous. Locations with *ignore* labels are excluded from the computation of the loss and its gradient.

1) Learning: We train the network with depth $L = 3$, an input resolution of 160×160 pixels, again using minibatches of size 16, and a temporal context of 8 frames. As input, we use Histogram of Oriented Gradients (HOG) and Histogram of Oriented Depth (HOD) channels and the whitened version of the images, as described by Höft *et al.* [19]. We use randomly chosen 10% of the training set for validation (early stopping and model selection). Ground truth is provided at times $t = 3$, 6, and 8 (cf. Figure 9). Training continues for 12,000 iterations, with an initial learning rate of $3 \cdot 10^{-4}$. The learning rate was decreased once the validation error failed to improve.

2) Comparison of the three architectures: We tested various hyper-parameter settings of the network architectures. The best result that we obtained for each architecture is listed in Table I. While the unidirectional model wins slightly over the bidirectional model, we were unable to produce similarly good results from the simplified network. Inspection of the weight matrices suggests that the learning problem becomes too difficult, i.e. the huge number of input maps at all time steps prevents the network from assigning credit to time frames correctly.

The slightly worse performance of the bidirectional network is likely insignificant, but could be caused by the smaller temporal context of the two subnetworks or the increased number of weights due to the lack of weight sharing between the networks.

A sample segmentation by the unidirectional model is depicted in Figure 8.

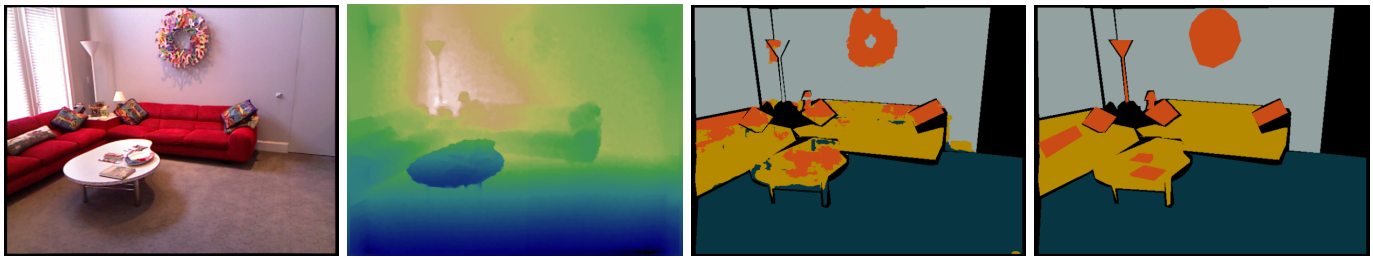
3) Exploiting the temporal context: To check whether our network can take advantage of a temporal context, we perform a static frame experiment. Here, we use the same frame as input at all time steps in the unidirectional model during training and prediction. In this setting, the recurrent architecture is still able to learn long-range spatial dependencies, but cannot exploit temporal context, which results in accuracy reduction in both class accuracy and pixel-wise accuracy (0.4 and 0.8 percentage points, respectively) relative to the model which has access to temporal context.

4) Extensions: We have investigated two extensions to the model described so far. As described in Section II-C, we provide direct access to downscaled versions of the input maps to higher abstraction layers. This improves our results by an additional 0.4% and 0.1% in pixel-wise accuracy and class accuracy, respectively.

We further integrate depth-normalized sliding windows of Schulz *et al.* [20] into our learning algorithm. This approach evaluates the model on image patches at a spatial resolution which is dependent on the distance of the center pixel to the observer, effectively building scale invariance into the model. Similarly to Schulz *et al.* [20], the use of sliding windows strongly improved both the class and pixel-wise accuracy by 7.2% and 4.1%, respectively. Apart from the increased resolution at which outputs can be generated, the network is able to focus on dynamics that take place within a smaller spatial context. Thus, it is able to track smaller objects easier. Since the network then processes image patches instead of images, we could reduce the size of the input the network receives from 160×160 to 80×80 pixels. This allows faster training at the cost of prediction time, since the network has to process patches that fully cover the input and combine their prediction afterwards.

5) Comparison with state of the art: We compare the results of our network with other results obtained from the literature. Due to the large impact the sliding window approach had on the results, we consider it separately.

Our recurrent neural networks use the same convolutional



(a) RGB frame (b) Depth (c) Prediction (d) Ground truth

Fig. 8: Prediction for one of the NYUD dataset frames. Images (a) and (b) show RGB and depth, respectively, after being preprocessed. (c) and (d) represent the prediction and ground truth, respectively, where color codes "floor" (), "prop" (), "furniture" (), "structure" () and "unknown" (). The network detects most of the pixels correctly, even some wrongly labeled ones (e.g the third object on the table and the center of the wall-mounted piece).

TABLE III: Comparison of NYUD classification performance with state of the art. Our recurrent net with depth-normalized sliding windows (SW) performs similar to explicit spatial aggregation of random forest features [21].

	Class Accuracies (%)				Average (%)	
Method	ground	struct	furnit	prop	Class	Pixel
Unidirectional + SW	90.0	76.3	52.1	**61.2**	69.9	67.5
Schulz *et al.* [20]	93.6	80.2	66.4	54.9	**73.7**	**73.4**
Müller and Behnke [22]	**94.9**	78.9	**79.7**	55.1	71.9	72.3
Stückler *et al.* [21]	90.8	81.6	67.9	19.9	65.0	68.3
Couprie *et al.* [23]	87.3	**86.1**	45.3	35.5	63.5	64.5
Höft *et al.* [19]	77.9	65.4	55.9	49.9	61.1	62.0
Silberman *et al.* [17]	68	59	70	42	59.6	58.6

base model as Höft *et al.* [19] and Schulz *et al.* [20][1], where the main difference is due to the introduction of the depth-normalized sliding window. The results in Table II show that we were able to improve on both baseline results in both pixel and class accuracy.

Table III shows our depth-normalized sliding window result together with state-of-the-art results on the same dataset. Our method is still behind the state-of-the-art, but shows promising results. In particular, it performs similar to Stückler *et al.* [21], who explicitly accumulated predictions in 3D. It is likely that results will improve significantly when the neural network has access to the height above ground [20] and predictions are post-processed with conditional random fields, which strongly improved object-class segmentation results of random forests [22] and neural networks [20].

Note, however, that except for Stückler *et al.* [21], none of the listed publications made use of temporal context to determine class labels.

VI. Conclusion

In this work, we introduced recurrent convolutional neural network architectures, which in addition to learning spatial relations are also able to exploit temporal relations from video. We started with a series of toy examples that showed that our networks are able to solve tasks that require denoising, detecting movement, and retaining uncertainty.

We further carried out experiments on sequences of indoor RGB-D video sequences from the NYU Depth v2 dataset. We tested three architectures: *simplified*, where the whole temporal context is available at all processing steps, *unidirectional*, which has access to past frames by recursive processing, and *bidirectional*, which also allows access to future context. In our experiments, the unidirectional and bidirectional networks obtained better results than the simplified model. This suggests that explicitly modeling time in the network architecture was beneficial.

Our proposed model improves on non-recurrent baseline models with similar architecture and obtains close to the state of the art RGB-D segmentation results.

[1] Note that we compare to their model without height-above-ground inputs, which were also not used in the present study.

References

[1] S. Behnke, *Hierarchical Neural Networks for Image Interpretation*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2766.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012. arXiv: 1207.0580 [abs].

[4] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 385.

[5] A. Graves, M. Abdelrahman, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[6] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Computer Vision and Pattern Recognition (CVPR), Conference on*, 2011, pp. 3361–3368.

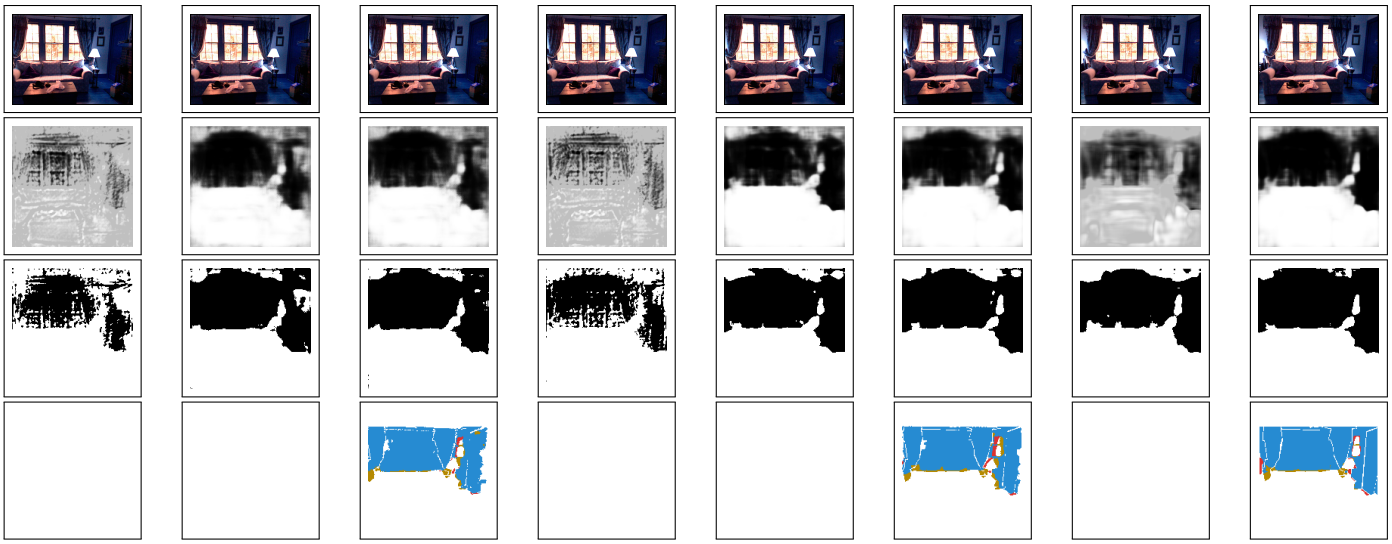


Fig. 9: Prediction for one sample of NYUD test dataset. Rows represent, from top to bottom: the RGB input, the softmax layer output, the output of the network; and the evaluation (● True Positives ○ True Negatives ● False Positives ● False Negatives) for the class structure.

.

[7] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European Conference on Computer Vision (ECCV)*, Springer, 2010, pp. 140–153.

[8] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NIPS)*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., 2014, pp. 568–576.

[9] V. Michalski, R. Memisevic, and K. Konda, "Modeling deep temporal dependencies with recurrent grammar cells," in *Advances in Neural Information Processing Systems (NIPS)*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., 2014, pp. 1925–1933.

[10] M. Jung, J. Hwang, and J. Tani, "Multiple spatio-temporal scales neural network for contextual visual recognition of human actions," in *International Conference on Development and Learning and on Epigenetic Robotics (ICDL)*, 2014.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), Conference on*, 2014, pp. 1725–1732.

[12] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," 2014.

[13] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Interspeech*, 2012.

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks.," *Journal of Machine Learning Research*, vol. 28, pp. 1310–1318, 2013.

[15] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization," *arXiv preprint arXiv:1502.04390*, 2015.

[16] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[17] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision (ECCV)*, 2012.

[18] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

[19] N. Höft, H. Schulz, and S. Behnke, "Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks," in *German Conference on Artificial Intelligence (KI)*, 2014.

[20] H. Schulz, N. Höft, and S. Behnke, "Depth and height aware semantic RGB-D perception with convolutional neural networks," in *European Symposium on Artificial Neural Networks (ESANN)*, 2015.

[21] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from RGB-D video," *Journal of Real-Time Image Processing*, 2013.

[22] A. C. Müller and S. Behnke, "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images," in *International Conference on Robotics and Automation (ICRA)*, 2014.

[23] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," in *International Conference on Learning Representations (ICLR)*, 2013.