Dynamic Filter Networks for Predicting Unobserved Views

Xu Jia[3,1]
xu.jia@esat.kuleuven.be
Bert De Brabandere[3,1]
bert.debrabandere@esat.kuleuven.be
Tinne Tuytelaars[1]
tinne.tuytelaars@esat.kuleuven.be
Luc Van Gool[2]
vangool@vision.ee.ethz.ch

[1] ESAT-PSI, KU Leuven
[2] D-ITET, ETH Zurich
[3] equal contribution to this work

Abstract

In a traditional convolutional layer, the learned filters stay fixed after training. In contrast, we introduce a new framework, the *Dynamic Filter Network*, where filters are generated dynamically conditioned on input. We show that this architecture is a powerful one, with increased flexibility thanks to its adaptive nature, yet without an increase in the number of model parameters. We demonstrate the effectiveness of the dynamic filter network on the tasks of video and stereo prediction, and reach state-of-the-art performance on the moving MNIST dataset with a much smaller model. By visualizing the learned filters, we illustrate that the network has learned flow information by only looking at unlabelled training data. This suggests that the network can be used to pretrain networks for various supervised tasks in an unsupervised way, like optical flow and depth estimation.

1 Introduction

The vision community has realized that endowing machines with capabilities of predicting another view from related views would be rewarding. Several papers have already addressed the generation of an image conditioned on given image(s). [10, 11] and [6] learn to rotate a given face to another pose. The authors of [3, 5, 7, 8] train a deep neural network to predict subsequent video frames. Flynn *et al.* [1] use a deep network to interpolate between views separated by a wide baseline. Yet all these methods apply the exact same set of filtering operations on each and every input image. This seems suboptimal for the tasks at hand. For example, for video prediction, there are different motion patterns within different video clips. The main idea behind our work is to generate the future frames with parameters adapted to the motion pattern within a particular video. Therefore, we propose a learnable parameter layer that provides custom parameters for different samples, which is similar in spirit to the Spatial Transformer Network [2]. Our method is also related to that of multiplicative interactions in neural networks [4, 9].

Our *dynamic filter module* consists of two parts: a *filter-generating network* and a *dynamic filtering layer* (see Figure 1). The filter-generating network dynamically generates sample-specific filter parameters conditioned on the network's input. Note that these are not fixed after training, like regular model parameters. The dynamic filtering layer then applies those sample-specific filters to the input. Both components of the dynamic filter module are differentiable with respect to the model parameters such that gradients can be backpropagated throughout the network. In particular, we propose a special kind of dynamic filtering layer which we coin *dynamic local filtering layer*, which is not only sample-specific but also position-specific. The filters in that case vary from position to position and from sample to sample, allowing for more sophisticated operations on the input. We demonstrate the effectiveness of the proposed dynamic filter module on the tasks of video prediction and stereo prediction. We also show that the computed dynamic filters can be visualised as an image similar to an optical flow or stereo map. Moreover, they are learned in a self-supervised way, i.e. without a need for manual annotations.

2 Dynamic Filter Networks

In this section we describe our dynamic filter framework. A dynamic filter module consists of a filter-generating network that produces filters conditioned on an input, and a dynamic filtering layer that applies the generated filters to another input. Both components of the dynamic filter module are differentiable. The two inputs of the module can be either identical or different, depending on the task. The general architecture of this module is shown schematically in Figure 1.

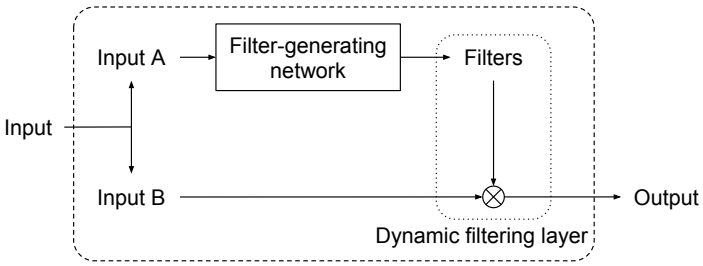

Figure 1: The general architecture of a Dynamic Filter Network.

2.1 Filter-Generating Network

The filter-generating network takes an input $I_A \in \mathbb{R}^{h\times w\times c_A}$, where h, w and c_A are height, width and number of channels of the input A respectively. It outputs filters $\mathcal{F}_\theta$ parameterized by parameters $\theta \in \mathbb{R}^{s\times s\times c_B\times n\times d}$, where s is the filter size, c_B the number of channels in input B and n the number of filters. d is equal to 1 for dynamic convolution and $h\times w$ for dynamic local filtering, which we discuss below. The filters are applied to input $I_B \in \mathbb{R}^{h\times w\times c_B}$ to generate an output $G = \mathcal{F}_\theta(I_B)$, with $G \in \mathbb{R}^{h\times w\times n}$. The filter size s determines the receptive field and is chosen depending on the application. The size of the receptive field can also be increased by stacking multiple dynamic filter modules. This is for example useful in applications that may involve large local displacements.

The filter-generating network can be implemented with any differentiable architecture, such as a multilayer perceptron or a convolutional network.

2.2 Dynamic Filtering Layer

The dynamic filtering layer takes images or feature maps I_B as input and outputs the filtered result $G \in \mathbb{R}^{h\times w\times n}$. For simplicity, in the experiments we only consider a single feature map ($c_B = 1$) filtered with a single generated filter ($n = 1$), but this is not required in a general setting. The dynamic filtering layer can be instantiated as a dynamic convolutional layer or a dynamic local filtering layer.

Dynamic convolutional layer. A dynamic convolutional layer is similar to a traditional convolutional layer in that the same filter is applied at every position of the input I_B. But different from the traditional convolutional layer where filter weights are fixed, in a dynamic convolutional layer filter weights θ are dynamically generated by a filter-generating network:

$$G(i,j) = \mathcal{F}_\theta(I_B(i,j)) \tag{1}$$

The filters are sample-specific and conditioned on the input of the filter-generating network.

Dynamic local filtering layer. An extension of the dynamic convolution layer that proves interesting, as we show in the experiments, is the dynamic local filtering layer. In this layer the filtering operation is not translation invariant anymore. Instead, different filters are applied to

[1] An extended version of this work was accepted by NIPS 2016

Model	Moving MNIST # params	bce
FC-LSTM [8]	142,667,776	341.2
Conv-LSTM [7]	7,585,296	367.1
Baseline (ours)	637,443	432.5
DFN (ours)	**637,361**	**285.2**

Table 1: Quantitative results on Moving MNIST: number of model parameters and average binary cross-entropy (bce).

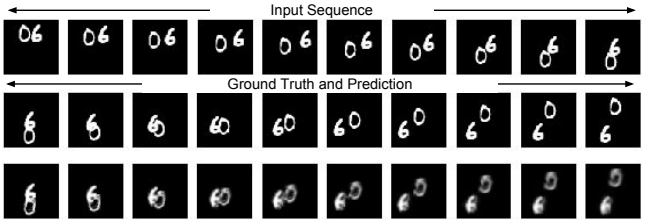

Figure 2: Qualitative results on moving MNIST. Note that the network has learned the bouncing dynamics and separation of overlapping digits.

different positions of the input I_B similarly to the traditional locally connected layer: for each position (i, j) of the input I_B, a specific local filter ${\mathcal{F}_\theta}^{(i,j)}$ is applied to the region centered around $I_B(i, j)$:

$$G(i, j) = {\mathcal{F}_\theta}^{(\mathbf{i},\mathbf{j})}(I_B(i, j)) \tag{2}$$

The filters used in this layer are not only sample specific but also position specific. The dynamic local filtering layer can perform not only a single transformation like the dynamic convolutional layer, but also position-specific transformations like local deformation. Before or after applying the dynamic local filtering operation we can add a dynamic pixel-wise bias to each element of the input I_B to address situations like photometric changes. This dynamic bias can be produced by the same filter-generating network that generates the filters for the local filtering.

3 Experiments

3.1 Video prediction

Given a sequence of frames, the task is to predict a sequence of future frames that follow the input frames. The architecture of our model is shown in Table 1 (right). We use a convolutional encoder-decoder as the filter-generating network where the encoder consists of several strided convolutional layers and the decoder consists of several unpooling layers and convolutional layers. To exploit the temporal correlation between frames we add a recurrent connection inside the filter-generating network. A softmax layer is applied such that each generated filter is encouraged to have only a few high magnitude elements. The generated filters are applied on the previous frame to make the prediction.

Moving MNIST We first evaluate the method on the moving MNIST dataset [8]. We use the average binary cross-entropy over the 10 generated frames as loss function. We also compare with a baseline consisting of only the filter-generating network, followed by a 1×1 convolution layer. The quantitative results are shown in Table 1 (left). Our method outperforms the state-of-the-art [7, 8] with a much smaller model.

Highway Driving We also evaluate our method qualitatively on real-world data of a car driving on the highway. We visualize the dynamically generated filters of the trained model in a flow-like manner, see Figure 4 (left). This byproduct is learned in an unsupervised way by only training on unlabeled video data.

3.2 Stereo prediction

The Dynamic Filter Network can also be applied to the task of stereo prediction, which we define as predicting the right view given the left view of two horizontal-disparity cameras. We recycle the architecture from the previous section without the recurrent connection and replace the square filters with horizontal filters. As shown in Figure 4 (right) the network has effectively learned to estimate depth information from a single image. The results suggest that it is possible to use the proposed dynamic filter network architecture to pre-train networks for optical flow and disparity map estimation in an unsupervised manner using only unlabeled data.

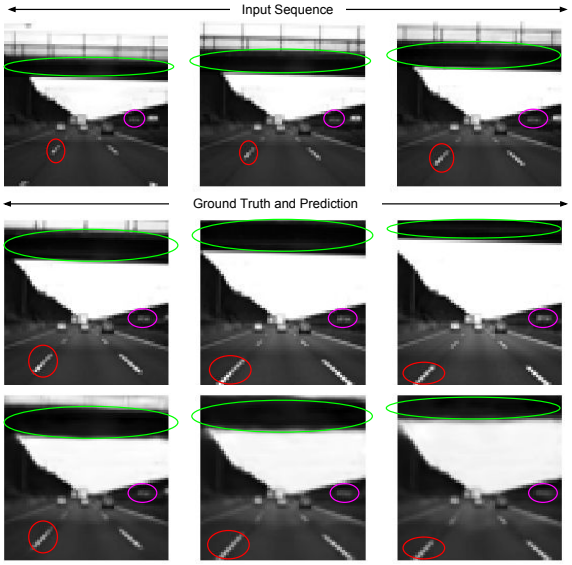

Figure 3: Qualitative results of video prediction on the Highway Driving dataset. Note the good prediction of the lanes (red), bridge (green) and a car moving in the opposite direction (purple).

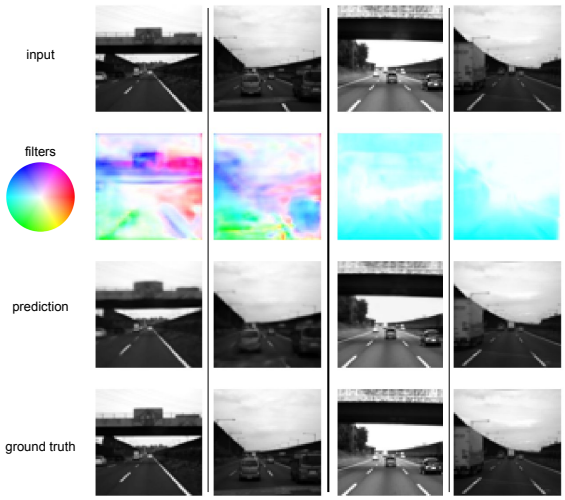

Figure 4: Some samples for video (*left*) and stereo (*right*) prediction and visualization of the dynamically generated filters.

Acknowledgement This work was supported by FWO through the project G.0696.12N "Representations and algorithms for captation, visualization and manipulation of moving 3D objects, subjects and scenes", the EU FP7 project Europa2, the iMinds ICON project Footwork and bilateral Toyota project.

[1] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2015.

[2] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[3] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[4] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.

[5] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, abs/1412.6604, 2014.

[6] S. Reed, K. Sohn, Y. Zhang, and H. Lee. Learning to disentangle factors of variation with manifold interaction. In *ICML*, 2014.

[7] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

[8] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[9] G. Taylor and G. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *ICML*, 2009.

[10] J. Yang, S. Reed, M.-H. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.

[11] J. Yim, H. Jung, B. Yoo, C. Choi, D.-S. Park, and J. Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015.