

## A Dynamic Convolutional Layer for Short Range Weather Prediction

Benjamin Klein, Lior Wolf, Yehuda Afek  
The Blavatnik School of Computer Science, Tel Aviv University

Deep learning and specifically Convolutional Neural Networks (CNNs) [1] are becoming increasingly popular in solving various computer vision applications. What sets CNNs apart from other neural networks is the use of the convolutional layer. This layer computes the output feature maps by convolving the feature maps of the previous layer with a set of filters which are the only parameters of the convolutional layer. Motivated by the success of CNNs, we decided to examine their performance in the task of short range weather prediction. In this task, one receives a sequence of rain radar images (Figure 2) such that each two consecutive images were taken 10 minutes apart and the goal is to predict the next image in the sequence. Observing that a radar image in the sequence can be usually approximated as a translation of the previous image in the sequence, suggested the need for a new layer that will translate the last image in the sequence according to the motion behavior of the all sequence.

Motivated by this observation, we present a new deep network layer called the "Dynamic Convolutional Layer", which generalizes the convolutional layer. Similar to the convolutional layer, the dynamic convolutional layer takes the feature maps from the previous layer and convolves them with filters. In contrast to the convolution layer, the dynamic convolution layer receives two inputs. The first input is the features maps from the previous layer and the second is the filters. The feature maps are obtained from the input by following a sub-network A. The filters are the result of applying a separate convolutional sub-network B on the input. The output of the layer is computed by convolving the filters across the features maps from the previous layer in the same way as in the convolution layer but here the filters are a function of the input and therefore vary from one sample to another during test time. The whole system is a directed acyclic graph of layers and therefore the training is done by using the back-propagation algorithm [2].

**Forward Pass** In the forward pass, network A computes the feature maps that will be given to the dynamic convolution layer as the first input and the separated sub convolution network B computes the filters that will be given to the dynamic convolution network as the second input (Figure 1). Let  $x_i^t$  be the  $i$ -th input feature map of sample  $t$ , let  $k_{ij}^t$  be the  $ij$  input kernel of sample  $t$  and let  $y_j^t$  be the  $j$ -th output feature map of sample  $t$ , then in the forward pass of the dynamic convolution network, the output feature maps are calculated as follows:

$$y_j^t = \sum_i k_{ij}^t * x_i^t$$

Notice that in contrast to the conventional convolution layer, in the dynamic convolution layer every sample has a different kernel  $k_{ij}^t$ .

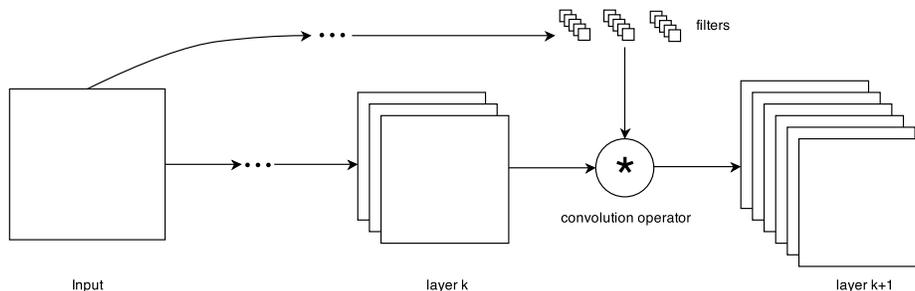


Figure 1: The dynamic convolutional layer. Layer  $k + 1$  is computed by convolving the filters across the feature maps in the  $k$  layer. The filters themselves are the result of applying a convolutional network on the input.

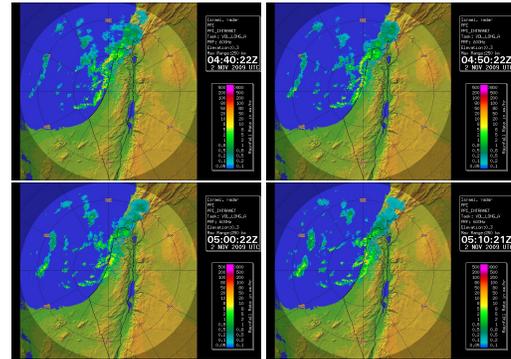


Figure 2: A sequence of 4 radar images from the Tel Aviv Dataset. Each two consecutive images were taken 10 minutes apart.

**Backward Pass** In the backward pass, the dynamic convolution layer computes the gradient of the loss function  $l$  with respect to  $x_i^t$ :

$$\frac{\partial l}{\partial x_i^t} = \sum_j \left( \frac{\partial l}{\partial y_j^t} \right) * (k_{ij}^t)$$

The values of the gradient  $\frac{\partial l}{\partial x_i^t}$  are passed to the layer in network A that produced  $x_i^t$ . Additionally, and similarly to the conventional convolutional layer, the gradient of the loss function with respect to  $k_{ij}^t$  is computed:

$$\frac{\partial l}{\partial k_{ij}^t} = \left( \frac{\partial l}{\partial y_j^t} \right) * (\tilde{x}_i^t)$$

In contrast to the convolution layer,  $k_{ij}^t$  are not parameters of the layer - they are a function of the input  $t$  that are passed from a previous layer in network B. Therefore, the values of the gradient  $\frac{\partial l}{\partial k_{ij}^t}$  are passed to the layer that computed  $k_{ij}^t$  as part of the back-propagation algorithm.

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] David E. Rumelhart, Geoff E. Hinton, and R. J. Wilson. Learning representations by back-propagating errors. pages 533–536, 1986.