# Convolutional Neural Fabrics – summary

Matěj Nikl

August 6, 2016

Convolutional Neural Fabrics (CNFs) is a architecture for embedding an exponentially large number of CNN architectures (via massive parameter sharing). It is intended as a step aside from the conventional CNN model architecture selection problem – it is not trying to solve it, instead it offers the opportunity for the fabrics to learn/select the most appropriate model (or even ensemble of models) within itself by standard back-propagation algorithm, leaving only the hyper-parameters of number of layers and channels up to us. Other hyper-parameters, which are being "set" automatically by embedding in the fabrics are:

- filter size per layer
- stride per layer
- number of pooling vs. convolutional layers
- type of pooling operator per layer
- size of the pooling regions
- ordering of pooling and convolutional layers
- channel connectivity pattern between layers
- type of activation (ReLU vs MaxOut)

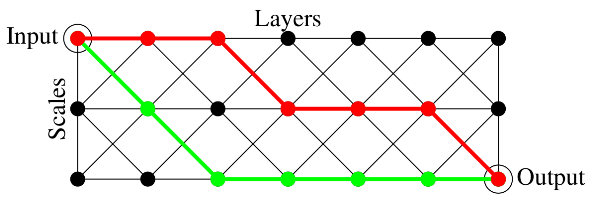

Figure 1: Trellis embedding of two seven-layer CNNs. Trellis nodes receiving the input and producing output are encircled. All edges are oriented to the right, down in the first layer, and towards the output in the last layer. The channel dimension of the 3D trellis is omitted for clarity.

## 1 Making sense of the Fabrics

The Fabrics consist of a 3D trellis that connects response maps at different layers, scales, and channels with a sparse homogeneous local connectivity pattern. Each node in the trellis represents a response map with the same number of dimensions as the input signal.

### 1.1 The trellis structure

It is spanned by three axes:

1. a **layer** axis – along which all edges advance, is analogous to the depth axis of a CNN
2. a **scale** axis – along which response maps of different resolutions are organized from fine to coarse
3. a **channel** axis – along which different response maps of the same scale and layer are organized

Each node is connected to a 3 x 3 scale-channel neighborhood in the previous layer, i.e. channel $c$ at scale $s$ receives input from channels $\{c-1, c, c+1\}$ at scales $\{s-1, s, s+1\}$. Input from a finer scale is obtained via strided convolution, and input from a coarser scale by convolution, after upsampling by padding zeros around the activations at the coarser level.

### 1.2 Embedded architectures

**Re-sampling operators** Fine-to-coarse change of the resolution of response maps is done using stride-two convolutions. Larger strides can be obtained by following multiple such edges.

*Average pooling* across small regions is also a strided convolution with uniform filter weights, thus available in the trellis. Average pooling over larger areas is also available.

*Bi-linear interpolation* is commonly used in deconvolutional networks. Factor-two interpolation can be represented on coarse-to-fine edges by using a filter with 1 in the center, 1/4 on corners, and 1/2 elsewhere. Larger powers of two can be obtained by repetition.

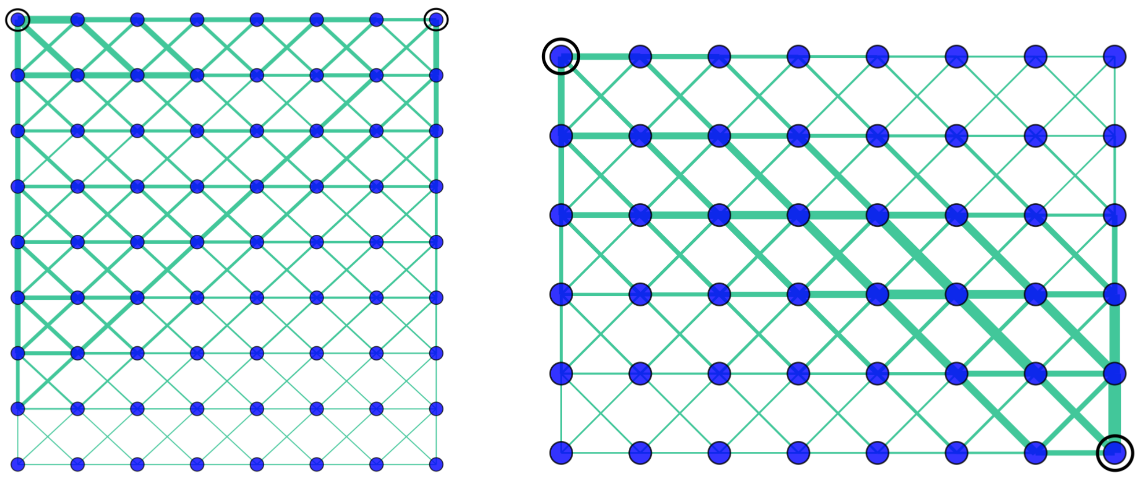

Figure 2: Visualization of mean-squared filter weights (mean along weights and channels) in models learned for Part Labels (left) and MNIST (right). Layers are laid out horizontally, and scales vertically.

| | Year | Augmentation | # Params. | Error (%) |
|---|---|---|---|---|
| Chang et al. | 2015 | N | 447 K | 0.24 |
| Lee et al. | 2015 | N | | 0.31 |
| Grid LSTM | 2016 | T | | 0.32 |
| Dropconnect | 2013 | T+F | 379 K | 0.32 |
| CKN | 2014 | N | 43 K | 0.39 |
| Maxout | 2013 | N | 420 K | 0.45 |
| Network in Network | 2013 | N | | 0.47 |
| Ours: CNF-sparse (L = 16, C = 32) | | T | 249 K | 0.48 |
| Ours: CNF-dense (L = 8, C = 64) | | T | 5.3 M | 0.33 |

Table 1: Comparison of results with the state of the art on MNIST. Data augmentation with translation and flipping is denoted by T and F respectively, N denotes no data augmentation.

**Filter sizes** A 5 x 5 filter can be implemented by computing nine intermediate channels to obtain a vectorized version of the 3 x 3 neighborhood at each pixel. A second 3 x 3 convolution can then aggregate values across the original 5 x 5 patch, and output the desired convolution. Repetition allows to implement every filter of any desired size.

**Ordering convolution and re-sampling** It is up to the back-propagation algorithm to choose the desired path(s) to use – by setting all, except one paths weights to zero, a particular sequence of convolutions and re-sampling operators is obtained.

**Channel connectivity pattern** This sparsely connected among the channel axis trellis suffices to emulate densely connected convolutional layers by copying channels, convolving them, and locally aggregating them.

## 2 Principles of modularization

The CNF architecture does not contain any standalone modules. The network itself can utilize some of its parts more, and by weight visualization it can be seen as modules being instationed, maybe even for certain purposes, however they still make up a integral, inseparable part of the network.

## 3 Principles of growing

The CNF architecture does not allow for any more growing as a conventional (C)NN does. Its purpose is not in growing, but in dynamically utilizing its trellis structure, whilst hiding away multiple hyper-parameters.