Convolutional Neural Fabrics

Shreyas Saxena and Jakob Verbeek
INRIA, Grenoble, France

Deep learning breakthrough in computer vision

- Convolutional nets at ImageNet’12 [Krizhevsky et al., 2012]
- Learning instead of hand-crafting features
- State of the art for visual recognition and matching

Images from [Kokkinos, 2016]
Keys issues in practice:

(1) Data

- Collection of huge manually labelled datasets, e.g.
- Synthetic datasets "cloned" from real footage
- Virtual KITTI dataset [Gaidon et al., 2016]
Keys issues in practice: (1) Data

- Collection of huge manually labelled datasets, e.g.

[Images of COCO and IMAGENET datasets]
Keys issues in practice: (1) Data

- Collection of huge manually labelled datasets, e.g.
  
  [COCO](http://cocodataset.org)  
  Common Objects in Context

- Synthetic datasets “cloned” from real footage

  Virtual KITTI dataset [Gaidon et al., 2016]
Keys issues in practice: (2) Optimization
Keys issues in practice: (2) Optimization

- Efficient techniques for non-convex optimization
  - Back propagation, stochastic gradient descent, initialization

\[ f(\theta_n) \leq f(\theta_{n-1}) \]
Keys issues in practice: (2) Optimization

- Efficient techniques for non-convex optimization
  - Back propagation, stochastic gradient descent, initialization

- Powerful compute platforms based on GPUs

  - NVIDIA’s P40
  - Facebook’s Big Sur
Keys issues in practice: (3) Network design

Activations: Rectified linear unit [Nair and Hinton, 2010], residual units [He et al., 2016]

Network architecture

Pooling type, filter size, pool and convolve ordering, etc.

AlexNet architecture [Krizhevsky et al., 2012]
Keys issues in practice: (3) Network design

- Activations: Rectified linear unit [Nair and Hinton, 2010], residual units [He et al., 2016]
Keys issues in practice: (3) Network design

- Activations: Rectified linear unit [Nair and Hinton, 2010], residual units [He et al., 2016]

- Network architecture
  - Pooling type, filter size, pool and convolve ordering, etc.

AlexNet architecture [Krizhevsky et al., 2012]
Important problem: maximize performance given hardware
Architecture design problem

- **Important problem**: maximize performance given hardware

- **Hard problem**: exponentially large architecture space
  - Pooling type, filter size, pool and convolve ordering, etc.

- Example: 19 Conv $\times$ 5 Pool layers $= 42,504$ architectures
  - Training a single architecture takes weeks on a GPU
Architecture design problem

- **Important problem**: maximize performance given hardware

- **Hard problem**: exponentially large architecture space
  - Pooling type, filter size, pool and convolve ordering, etc.

- Example: 19 Conv $\times$ 5 Pool layers $= 42,504$ architectures
  - Training a single architecture takes weeks on a GPU
Architecture design problem

- **Important problem**: maximize performance given hardware

- **Hard problem**: exponentially large architecture space
  - Pooling type, filter size, pool and convolve ordering, etc.

- Example: 19 Conv × 5 Pool layers = 42,504 architectures
  - Training a single architecture takes weeks on a GPU
Architecture design in practice: One size fits all ?!

- Lack of systematic methods for architecture design
  - Lack of guiding theory
  - Naive exhaustive search way too expensive
  - Local search with parameter “recycling” [Chen et al., 2016]
Architecture design in practice: One size fits all ?!

- Lack of systematic methods for architecture design
  - Lack of guiding theory
  - Naive exhaustive search way too expensive
  - Local search with parameter “recycling” [Chen et al., 2016]

- Massive reliance on a handful of architectures
  - Re-purposing Alex-net, VGG-16/19 nets, residual nets
  - Can result in overkill, by lack of other designs
Routing traffic in a city

- Combinatorially many routes in a city
- Example of how a 2D structure embeds many 1D sequences
Routing traffic in a city \(\approx\) routing signal in a fabric

- Combinatorially many CNNs in a 2D grid network
- 2D network embeds VGG-19 [Simonyan and Zisserman, 2015]
- One path among 42,504 in fabric with 120 nodes
Convolutional neural fabrics

- Nodes are like conventional CNN layers

Classification and segmentation CNNs embedded in fabrics
Convolutional neural fabrics

- Edges are $3 \times 3$ convolutions
- Diagonal edges use up/down sampling
Fabric properties

- Embed exponentially many CNNs as paths
- Implicit ensembling of all models, sharing weights on overlap
- Multiple outputs possible: e.g. classification and segmentation
Fabric structure

- Homeogeneous local connectivity across nodes
- All channels interact between connected nodes
  - As in most CNNs, but exceptions exist, e.g. AlexNet
Fabric structure

- Homeogeneous local connectivity across nodes
- All channels interact between connected nodes
  - As in most CNNs, but exceptions exist, e.g. AlexNet
- Minimal “infrastrucutre” to implement CNNs?
Fabrics with sparse channel connectivity

- Each node contains a single response map
- Channels organized along a third dimension
Sparse fabrics: signal propagation

- Activations computed as $3 \times 3$ convolutions of neighboring scales and channels in previous layer

\[
a(s, c, l + 1) = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} \text{Conv}(a(s + i, c + j, l); \theta_{scl}^{ij})
\]  

- Units process $3 \times 3 \times 3 \times 3$ area, convolution in space only
Learning instead of hand-crafting architectures

- Signals advance on layer axis: standard back-prop training
- Learning configures the fabric to implement one architecture, or as an ensemble of embedded architectures
Convolutional neural fabric is a “universal” architecture

- Large enough fabrics can essentially implement any CNN

[Krizhevsky et al., 2012, Simonyan and Zisserman, 2015, Noh et al., 2015, Farabet et al., 2013, Ronneberger et al., 2015, Long et al., 2015]
Universality: (1) ordering of pooling and convolution

- Different paths across the fabric
Universality: (2) down-sampling operators

- Fabrics do **down-sampling by strided convolution**
  - Enough to “build-up” average and max pooling

- Average pooling: striding uniform filter along single channel

\[
\begin{align*}
\text{average pooling:} & \quad \frac{(a + b)}{2}, \\
\text{max pooling:} & \quad \max(a, b)
\end{align*}
\]
Universality: (2) down-sampling operators

- Fabrics do **down-sampling by strided convolution**
  - Enough to “build-up” average and max pooling

- Average pooling: striding uniform filter along single channel

- Max-pooling: consider computing \( \max(a, b) \)
  - Compute three terms via convolution
    \[ \{ (a + b)/2, (a - b)/2, (b - a)/2 \} \]
  - Apply ReLU activation: \( x \leftarrow \max(0, x) \)
  - At most two non-zero terms remain
  - Summing all three terms by convolution gives \( \max(a, b) \)
Universality: (3) up-sampling operators

- Fabrics do **up-sampling by zero-padding** + convolution

<table>
<thead>
<tr>
<th>Bi-linear:</th>
<th>Nearest neighbor:</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\] |

- Fabric embeds these per-channel interpolations
- More general, e.g. cross-channel, interpolation as well
Universality: (3) up-sampling operators

- Fabrics do **up-sampling by zero-padding** + convolution

- Various interpolations by convolution with specific filters
  
  **Bi-linear:**
  \[
  \frac{1}{4} \begin{pmatrix}
  1 & 2 & 1 \\
  2 & 4 & 2 \\
  1 & 2 & 1 \\
  \end{pmatrix}
  \]

  **Nearest neighbor:**
  \[
  \begin{pmatrix}
  1 & 1 & 0 \\
  1 & 1 & 0 \\
  0 & 0 & 0 \\
  \end{pmatrix}
  \]

- Fabric embeds these per-channel interpolations
- More general, e.g. cross-channel, interpolation as well
Universality: (4) Filter sizes

- Fabrics use only $3 \times 3$ convolutions
- Consider computing a $5 \times 5$ convolution over a single channel
  - Compute 9 “temporary” channels using 1-hot filters
    
    \[
    \begin{pmatrix}
    1 & 0 & 0 \\
    0 & 0 & 0 \\
    0 & 0 & 0 \\
    \end{pmatrix}, \begin{pmatrix}
    0 & 1 & 0 \\
    0 & 0 & 0 \\
    0 & 0 & 0 \\
    \end{pmatrix}, \ldots
    \]

- Stores vectorized version of $3 \times 3$ patch in 9 channels
- Convolution with 3 filter can now access $5 \times 5$ patch
Universality: (4) Filter sizes

- Fabrics use only $3 \times 3$ convolutions
- Consider computing a $5 \times 5$ convolution over a single channel
  - Compute 9 “temporary” channels using 1-hot filters
    \[
    \begin{pmatrix}
    1 & 0 & 0 \\
    0 & 0 & 0 \\
    0 & 0 & 0
    \end{pmatrix},
    \begin{pmatrix}
    0 & 1 & 0 \\
    0 & 0 & 0 \\
    0 & 0 & 0
    \end{pmatrix}, \ldots
    \]
- Stores vectorized version of $3 \times 3$ patch in 9 channels
Universality: (4) Filter sizes

- Fabrics use only $3 \times 3$ convolutions

- Consider computing a $5 \times 5$ convolution over a single channel
  - Compute 9 “temporary” channels using 1-hot filters
    \[
    \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ldots
    \]

- Stores vectorized version of $3 \times 3$ patch in 9 channels
- Convolution with 3 filter can now access $5 \times 5$ patch
Universality: (5) Channel connectivity

- Fabrics with **sparse channel connectivity** suffice to implement networks with dense channel connectivity
Universality: (5) Channel connectivity

- Fabrics with **sparse channel connectivity** suffice to implement networks with dense channel connectivity

- Demonstration by explicit construction
  - Create multiple copies of input channels
  - Aggregate input with corresponding filters
  - Fiddle with biases to remove negative intermediate results

<table>
<thead>
<tr>
<th>Channels</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>a a a a a</td>
<td>a a a a a</td>
</tr>
<tr>
<td>b a b b b</td>
<td>b b b b b</td>
</tr>
<tr>
<td>c b a c c</td>
<td>b c c c c</td>
</tr>
<tr>
<td>d c c a d</td>
<td>c b d d d</td>
</tr>
<tr>
<td>e d d d a</td>
<td>e d d b e</td>
</tr>
<tr>
<td>e e e e a</td>
<td>e e e b</td>
</tr>
<tr>
<td></td>
<td>e</td>
</tr>
<tr>
<td>a a a a a</td>
<td>d c+d+e</td>
</tr>
<tr>
<td>... c</td>
<td>a+b+c+d+e</td>
</tr>
<tr>
<td></td>
<td>a+b</td>
</tr>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Extension: Channel doubling across scales

- Channels are “cheaper” at coarser resolutions
  - Grow the number of channels when down-sampling
  - Commonly used in densely connected networks
  - Computation constant per layer for dense connect

- How about channel doubling in sparse fabrics?
Extension: Channel doubling across scales

- Channels are “cheaper” at coarser resolutions
  - Grow the number of channels when down-sampling
  - Commonly used in densely connected networks
  - Computation constant per layer for dense connect

- How about channel doubling in sparse fabrics?
  - As before: total of 9 incoming channels per node
  - 4 from coarser, 2 from finer, 3 from same resolution
Experimental evaluation:

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Parameters</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsogkas et al.</td>
<td>2015</td>
<td>414M</td>
<td>96.97%</td>
</tr>
<tr>
<td>Kae et al.</td>
<td>2013</td>
<td>0.7M</td>
<td>94.95%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (sparse)</td>
<td></td>
<td>0.1M</td>
<td>95.58%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (dense)</td>
<td></td>
<td>8.0M</td>
<td>95.63%</td>
</tr>
</tbody>
</table>

- Competitive with the best hand-crafted architectures
- Without structured prediction model: CRF, RBM, etc
- Upto $4000 \times$ fewer params. than re-purposed VGG net
- Trained from scratch with $500 \times$ fewer images
## Experimental evaluation: Face Segmentation

<table>
<thead>
<tr>
<th>Part Labels</th>
<th>Year</th>
<th># Params.</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsogkas et al. [Tsogkas et al., 2015]</td>
<td>2015</td>
<td>&gt;414M</td>
<td>96.97%</td>
</tr>
<tr>
<td>Kae et al. [Kae et al., 2013]</td>
<td>2013</td>
<td>0.7M</td>
<td>94.95%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (sparse)</td>
<td></td>
<td>0.1M</td>
<td>95.58%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (dense)</td>
<td></td>
<td>8.0M</td>
<td>95.63%</td>
</tr>
</tbody>
</table>

- Competitive with the best hand-crafted architectures
- Without structured prediction model: CRF, RBM, etc
- Upto 4,000 × fewer params. than re-purposed VGG net
- Trained from scratch with 500 × fewer images
Experimental evaluation: Face Segmentation

<table>
<thead>
<tr>
<th>Part Labels</th>
<th>Year</th>
<th># Params.</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsogkas et al. [Tsogkas et al., 2015]</td>
<td>2015</td>
<td>&gt;414M</td>
<td>96.97%</td>
</tr>
<tr>
<td>Kae et al. [Kae et al., 2013]</td>
<td>2013</td>
<td>0.7M</td>
<td>94.95%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (sparse)</td>
<td></td>
<td>0.1M</td>
<td>95.58%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (dense)</td>
<td></td>
<td>8.0M</td>
<td>95.63%</td>
</tr>
</tbody>
</table>

- Competitive with the best hand-crafted architectures
- Without structured prediction model: CRF, RBM, etc.
- Upto 4,000× fewer params. than re-purposed VGG net
- Trained from scratch with 500× fewer images
Experimental evaluation: digit classification

<table>
<thead>
<tr>
<th>MNIST</th>
<th># Params.</th>
<th># Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Chang and Chen, 2015]</td>
<td>447K</td>
<td>0.24%</td>
</tr>
<tr>
<td>[Wan et al., 2013] (Dropconnect)</td>
<td>379K</td>
<td>0.32%</td>
</tr>
<tr>
<td>[Goodfellow et al., 2013] (MaxOut)</td>
<td>420K</td>
<td>0.45%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (sparse)</td>
<td>249K</td>
<td>0.48%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (dense)</td>
<td>5.3M</td>
<td>0.33%</td>
</tr>
</tbody>
</table>
## Experimental evaluation: digit classification

<table>
<thead>
<tr>
<th>MNIST</th>
<th># Params.</th>
<th># Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Chang and Chen, 2015]</td>
<td>447K</td>
<td>0.24%</td>
</tr>
<tr>
<td>[Wan et al., 2013] (Dropconnect)</td>
<td>379K</td>
<td>0.32%</td>
</tr>
<tr>
<td>[Goodfellow et al., 2013] (MaxOut)</td>
<td>420K</td>
<td>0.45%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (sparse)</td>
<td>249K</td>
<td>0.48%</td>
</tr>
<tr>
<td>Convolutional Neural Fabric (dense)</td>
<td>5.3M</td>
<td>0.33%</td>
</tr>
</tbody>
</table>

- **Competitive with the best hand-crafted architectures**
- Sparse versus densely connected fabric
  - $20 \times$ fewer parameters
  - Error increased by 0.15%
Experimental evaluation: MNIST digit classification

- All 33 errors among 10,000 test samples
- Format: Prediction (True)
Activated connections in trained fabric

- Effective multi-path network is recovered by training CIFAR10
- Can be used to prune fabric to fit hardware requirements
  - Cutting 67% of connections increases error from 7.4% to 8.1%
Activated connections in trained fabric

- Effective multi-path network is recovered by training CIFAR10
- Can be used to prune fabric to fit hardware requirements
  - Cutting 67% of connections increases error from 7.4% to 8.1%
Convolutional Neural Fabrics: Conclusion

- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one

- Ongoing and future work
  - Scaling from hundreds to thousands of channels
  - Long-range connections across channels and layers
  - Scale invariance by convolution along scale axis
Convolutional Neural Fabrics: Conclusion

- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one

- From hand-crafted to learned features
Convolutional Neural Fabrics: Conclusion

- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one

- From hand-crafted to learned features architectures
Convolutional Neural Fabrics: Conclusion

► Fabrics are universal architecture for conv nets
  ▶ Fabric learns across architectures instead of selecting one

► From hand-crafted to learned features architectures

► Ongoing and future work
  ▶ Scaling from hundreds to thousands of channels
  ▶ Long-range connections across channels and layers
  ▶ Scale invariance by convolution along scale axis
Convolutional Neural Fabrics

Shreyas Saxena and Jakob Verbeek
firstname.lastname@inria.fr

Thank you!

Batch-normalized maxout network in network.
Arxiv preprint.

Net2net: Accelerating learning via knowledge transfer.
In ICLR.

Learning hierarchical features for scene labeling.

Virtual worlds as proxy for multi-object tracking analysis.
In CVPR.

Maxout networks.
In ICML.

Identity mappings in deep residual networks.
In ECCV.

Augmenting CRFs with Boltzmann machine shape priors for image labeling.
In CVPR.

Ubernet: Training a ‘universal’ convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory.
arXiv 1609.02132.
References II

Imagenet classification with deep convolutional neural networks.
In *NIPS*.

Fully convolutional networks for semantic segmentation.
In *CVPR*.

Rectified linear units improve restricted Boltzmann machines.
In *ICML*.

Learning deconvolution network for semantic segmentation.
In *ICCV*.

U-net: Convolutional networks for biomedical image segmentation.
In *Medical Image Computing and Computer-Assisted Intervention*.

Very deep convolutional networks for large-scale image recognition.
In *ICLR*.

Deep learning for semantic part segmentation with high-level guidance.
Arxiv preprint.

Regularization of neural networks using DropConnect.
In *ICML*.
Fabric analysis

- Comparing fabric variants: maps, parameters, activations

<table>
<thead>
<tr>
<th># chan. / scale</th>
<th># resp. maps</th>
<th># parameters (sparse)</th>
<th># parameters (dense)</th>
<th># activations</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>$C \cdot L \cdot S$</td>
<td>$C \cdot L \cdot 3^{D+1} \cdot 3 \cdot S$</td>
<td>$C \cdot L \cdot 3^{D+1} \cdot C \cdot S$</td>
<td>$C \cdot L \cdot N^2 \cdot \frac{4}{3}$</td>
</tr>
<tr>
<td>doubling</td>
<td>$C \cdot L \cdot 2^S$</td>
<td>$C \cdot L \cdot 3^{D+1} \cdot 3 \cdot 2^S$</td>
<td>$C \cdot L \cdot 3^{D+1} \cdot C \cdot 4^S \cdot \frac{7}{18}$</td>
<td>$C \cdot L \cdot N^2 \cdot 2$</td>
</tr>
</tbody>
</table>