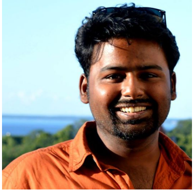
# Convolutional Neural Fabrics

Shreyas Saxena and Jakob Verbeek
INRIA, Grenoble, France



Neural Information Processing Systems 2016

# 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:

# Keys issues in practice: (1) Data

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

# Keys issues in practice: (1) Data

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

COCO Common Objects in Context

IMAGENET

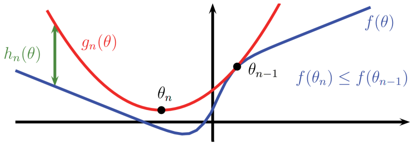
- Synthetic datasets “cloned” from real footage

Virtual KITTI dataset [Gaidon et al., 2016]

# Keys issues in practice: (2) Optimization

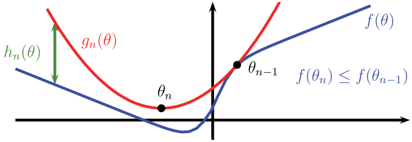
# Keys issues in practice: (2) Optimization

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

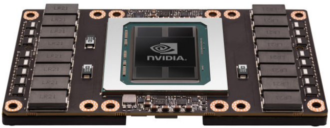


# Keys issues in practice: (2) Optimization

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




- Powerful compute platforms based on GPUs



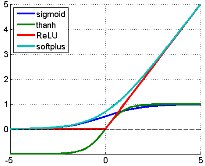

Nvidia P100

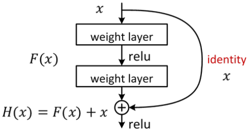

Nvidia DGX-1

# Keys issues in practice: (3) Network design

# Keys issues in practice: (3) Network design

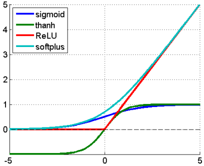
- Activations: Rectified linear unit [Nair and Hinton, 2010], residual units [He et al., 2016]

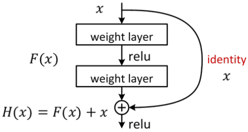


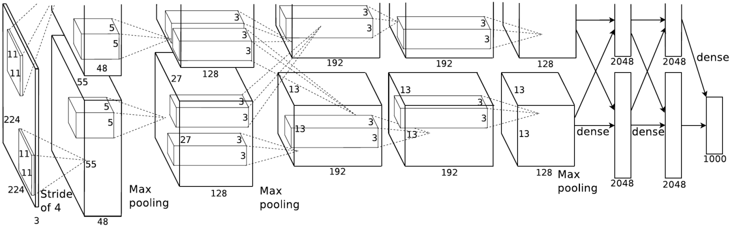
# Keys issues in practice: (3) Network design

- 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]

# Architecture design problem

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

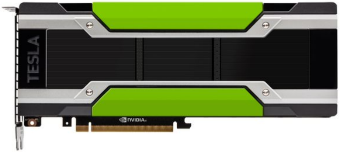


# Architecture design problem

- **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



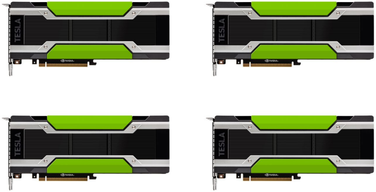
# Architecture design problem

- **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

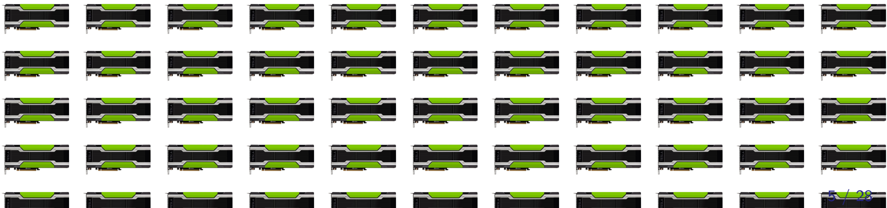


# Architecture design problem

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



- **Hard problem**: exponentially large architecture space
  - Pooling type, filter size, pool and convolve ordering, *etc.*
- Example: 19 Conv $\times$ 5 Pool layers $=$ 42,504 architectures
  - Training a single architecture takes weeks on a GPU



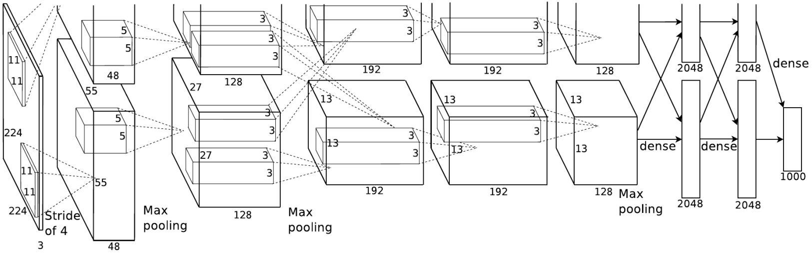
# Architecture design in practice: One size fits all ?!

- 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 ?!

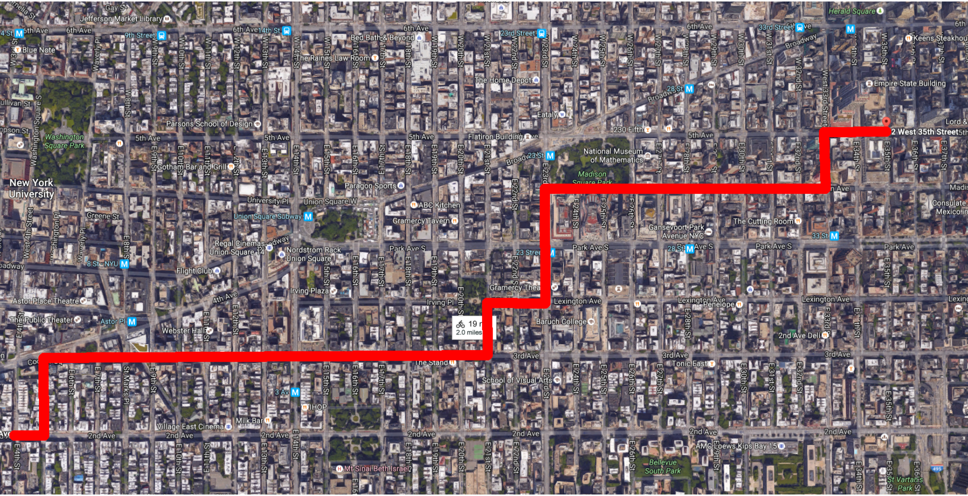
- Lack of systematic methods for architecture design
  - Lack of guiding theory
  - Naive exhaustive search way too expensive
  - Local search with parameter “recycling” [Chen et al., 2016]

- 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



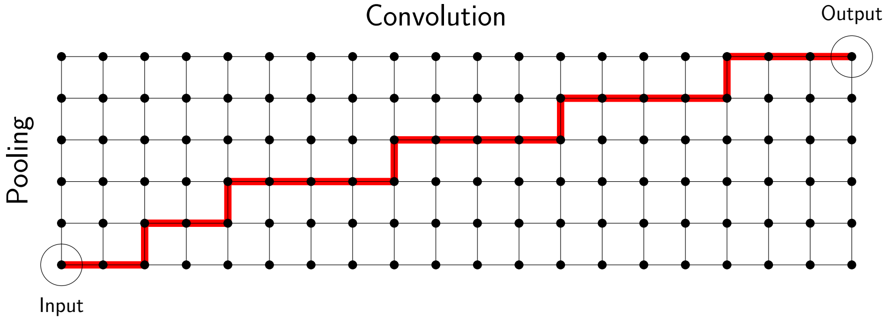
# Routing traffic in a city

- Combinatorially many routes in a city
- Example of how a 2D structure embeds many 1D sequences



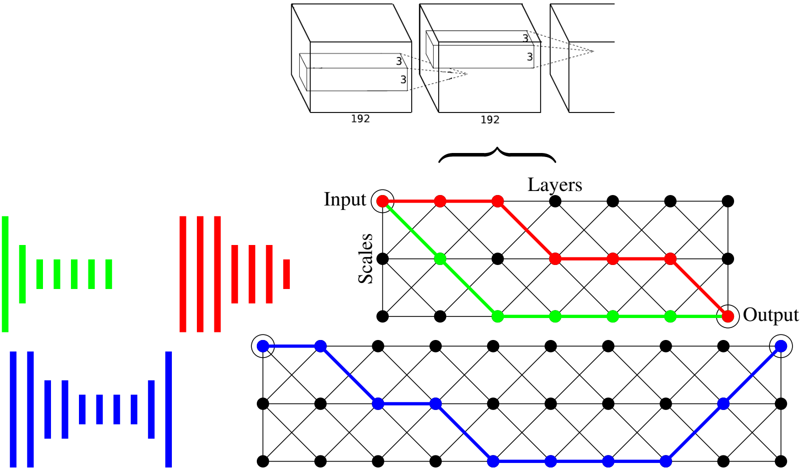
# Routing traffic in a city ≈ routing signal in a fabric

- 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

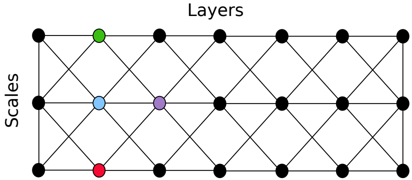
- Nodes are like conventional CNN layers

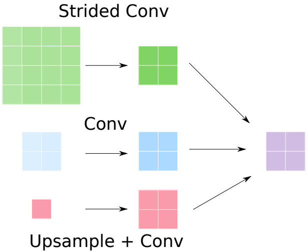



Classification and segmentation CNNs embedded in fabrics

# Convolutional neural fabrics

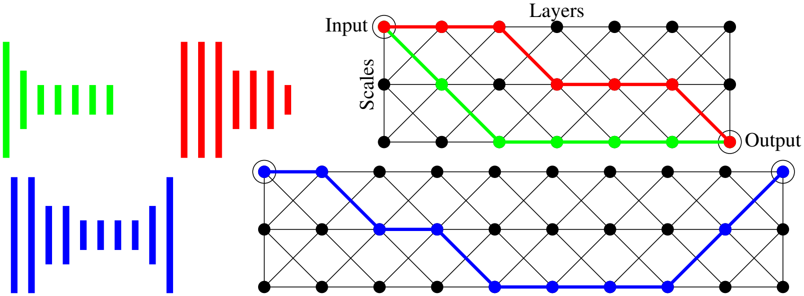
- Edges are $3 \times 3$ convolutions
- Diagonal edges use up/down sampling



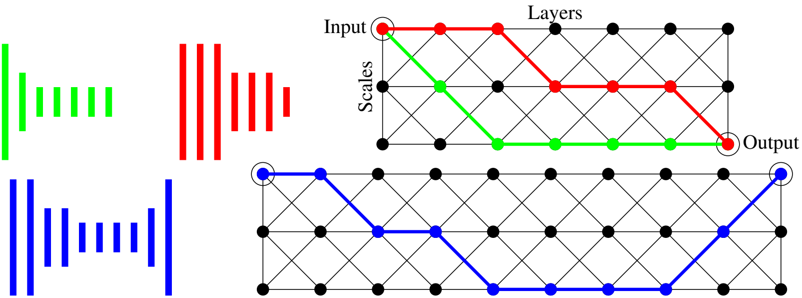
# Fabric properties

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



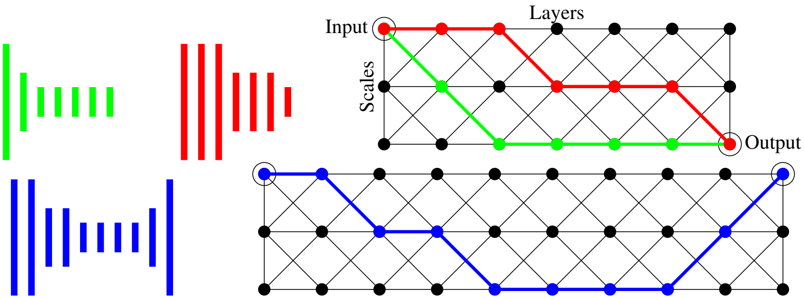
# Fabric structure

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



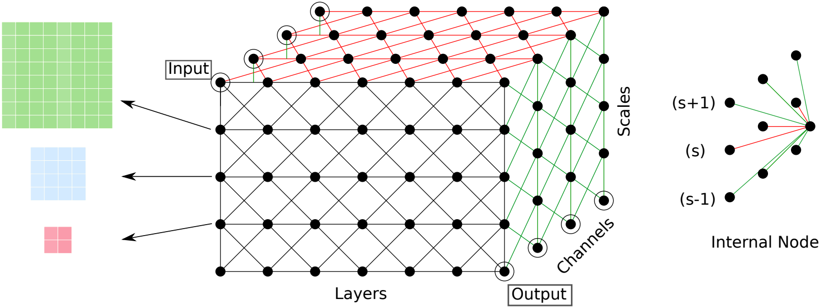
# Fabric structure

- 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

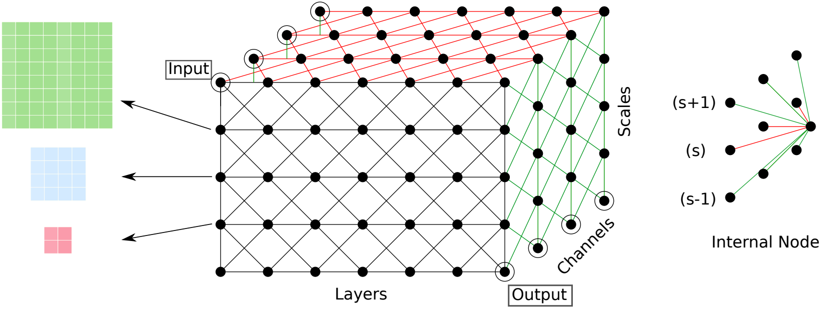


# Sparse fabrics: signal propagation

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

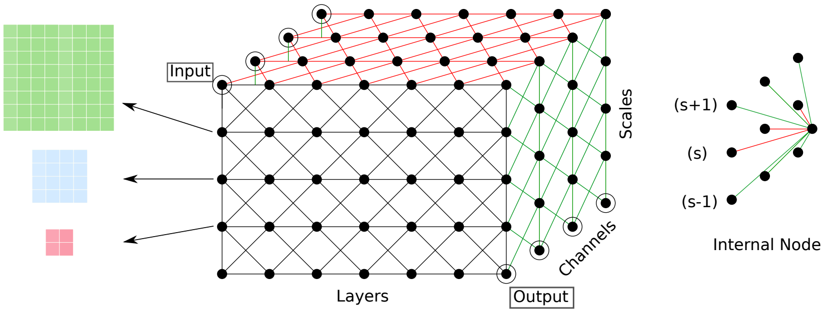
$$a(s, c, l+1) = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} \mathrm{Conv}\big(a(s+i, c+j, l); \theta^{ij}_{scl}\big) \quad (1)$$

- Units process $3 \times 3 \times 3 \times 3$ area, convolution in space only



# Learning instead of hand-crafting architectures

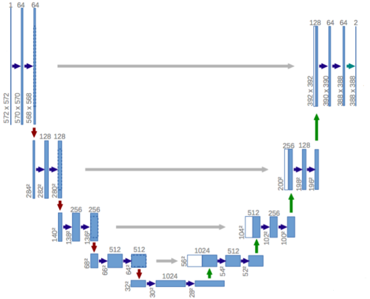
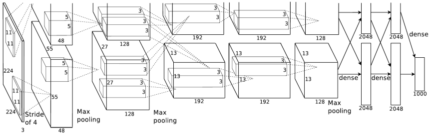
- Signals advance on layer axis: standard back-prop training
- Learning configures the fabric to implement one architecture, or as an ensemble of embedded architectures

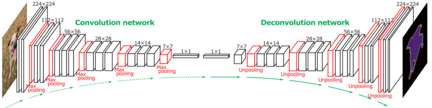


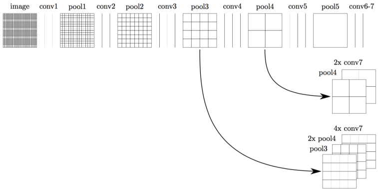
# Convolutional neural fabric is a “universal” architecture

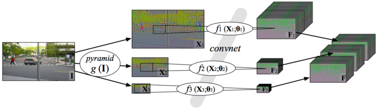
- Large enough fabrics can essentially implement any CNN

[Krizhevsky et al., 2012, Simonyan and Zisserman, 2015, Noh et al., 2015, Farabet et al., 2013, Ronneberger et al., 2015, Long et al., 2015]

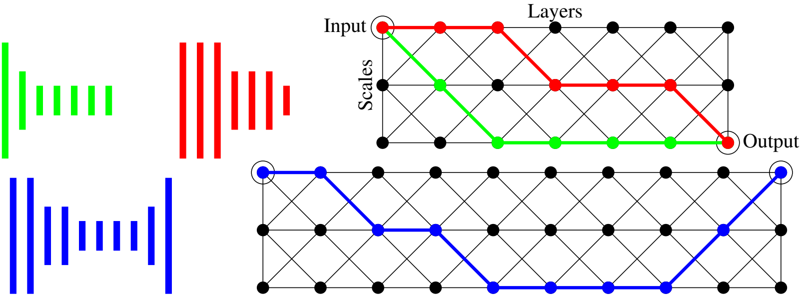






# Universality: (1) ordering of pooling and convolution

- Different paths across the fabric



# Universality: (2) down-sampling operators

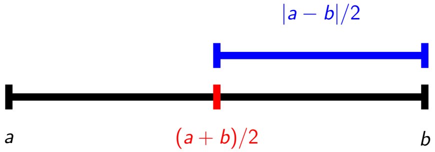
- Fabrics do **down-sampling by strided convolution**
  - Enough to “build-up” average and max pooling
- Average pooling: striding uniform filter along single channel

# Universality: (2) down-sampling operators

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

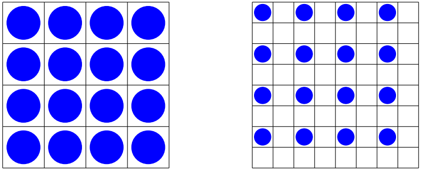
- Average pooling: striding uniform filter along single channel

- 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 $\max(a, b)$



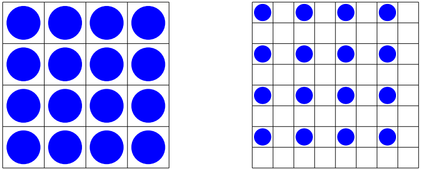
# Universality: (3) up-sampling operators

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



# Universality: (3) up-sampling operators

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



- Various interpolations by convolution with specific filters

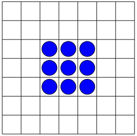
$$\text{Bi-linear:} \quad \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \qquad \text{Nearest neighbor:} \quad \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Fabric embeds these per-channel interpolations
- More general, e.g. cross-channel, interpolation as well

# Universality: (4) Filter sizes

- Fabrics use only $\mathbf{3 \times 3}$ **convolutions**
- Consider computing a $5 \times 5$ convolution over a single channel
  - Compute 9 "temporary" channels using 1-hot filters

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ldots$$

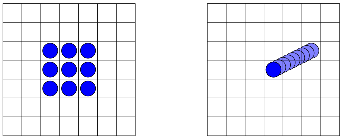


# Universality: (4) Filter sizes

- 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

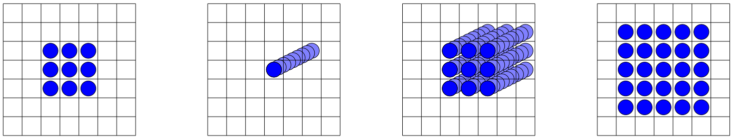


# Universality: (4) Filter sizes

- Fabrics use only **3 × 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

Layers

Channels

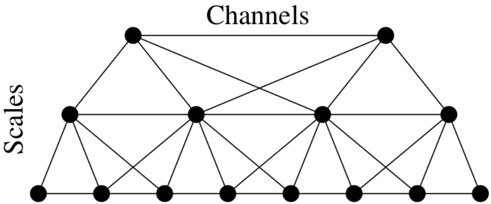
| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | | $a$ | | |
| $b$ | $\mathbf{a}$ | $b$ | $b$ | $b$ | $b$ | $b$ | $b$ | $b$ | $b$ | | $b$ | | |
| $c$ | $b$ | $\mathbf{a}$ | $c$ | $c$ | $c$ | $\mathbf{b}$ | $c$ | $c$ | $c$ | ... | $c$ | ... | ... |
| $d$ | $c$ | $c$ | $\mathbf{a}$ | $d$ | $d$ | $c$ | $\mathbf{b}$ | $d$ | $d$ | | $d$ | | |
| $e$ | $d$ | $d$ | $d$ | $\mathbf{a}$ | $e$ | $d$ | $d$ | $\mathbf{b}$ | $e$ | | $e$ | | |
| | $e$ | $e$ | $e$ | $e$ | $\mathbf{a}$ | $e$ | $e$ | $e$ | $\mathbf{b}$ | | $\mathbf{e}$ | — | — |
| | | | | | | $a$ | $a$ | $a$ | $a$ | | $d$ | $c+d+e$ | — |
| | | | | | | | | | | ... | $c$ | — | $a+b+c+d+e$ |
| | | | | | | | | | | | $b$ | $a+b$ | — |
| | | | | | | | | | | | $a$ | — | — |
| | | | | | | | | | | ... | $\vdots$ | ... | ... |

# 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:

# Experimental evaluation: Face Segmentation

| Part Labels | Year | # Params. | Accuracy |
|---|---|---|---|
| Tsogkas *et al.* [Tsogkas et al., 2015] | 2015 | >414M | 96.97% |
| Kae *et al.* [Kae et al., 2013] | 2013 | 0.7M | 94.95% |
| Convolutional Neural Fabric (sparse) | | 0.1M | 95.58% |
| Convolutional Neural Fabric (dense) | | 8.0M | 95.63% |

image

prediction

image

prediction

# Experimental evaluation: Face Segmentation

| Part Labels | Year | # Params. | Accuracy |
|---|---|---|---|
| Tsogkas *et al.* [Tsogkas et al., 2015] | 2015 | >414M | 96.97% |
| Kae *et al.* [Kae et al., 2013] | 2013 | 0.7M | 94.95% |
| Convolutional Neural Fabric (sparse) | | 0.1M | 95.58% |
| Convolutional Neural Fabric (dense) | | 8.0M | 95.63% |

- **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

image

prediction

image

prediction

# Experimental evaluation: digit classification

| MNIST | # Params. | # Error |
|---|---|---|
| [Chang and Chen, 2015] | 447K | 0.24% |
| [Wan et al., 2013] (Dropconnect) | 379K | 0.32% |
| [Goodfellow et al., 2013] (MaxOut) | 420K | 0.45% |
| Convolutional Neural Fabric (sparse) | 249K | 0.48% |
| Convolutional Neural Fabric (dense) | 5.3M | 0.33% |

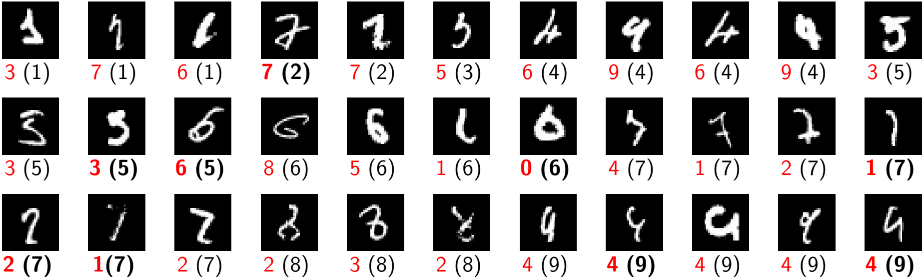
# Experimental evaluation: digit classification

| MNIST | # Params. | # Error |
|---|---|---|
| [Chang and Chen, 2015] | 447K | 0.24% |
| [Wan et al., 2013] (Dropconnect) | 379K | 0.32% |
| [Goodfellow et al., 2013] (MaxOut) | 420K | 0.45% |
| Convolutional Neural Fabric (sparse) | 249K | 0.48% |
| Convolutional Neural Fabric (dense) | 5.3M | 0.33% |

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

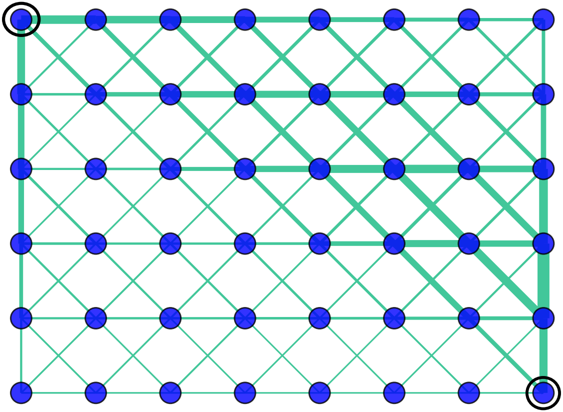
# Experimental evaluation: MNIST digit classification

- All 33 errors among 10,000 test samples
- Format: Prediction (True)



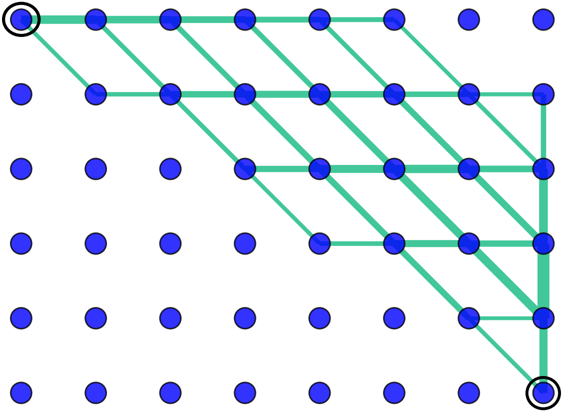
# Activated connections in trained fabric

- 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%



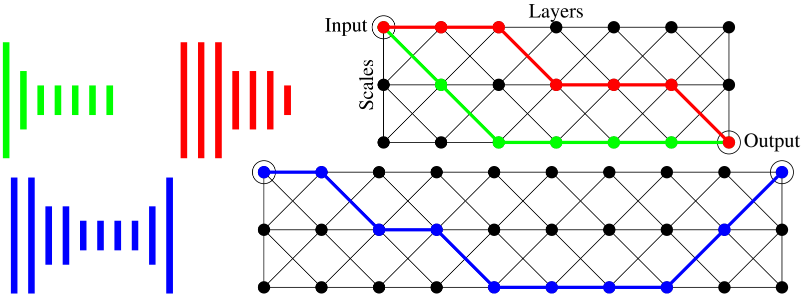
# Activated connections in trained fabric

- 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

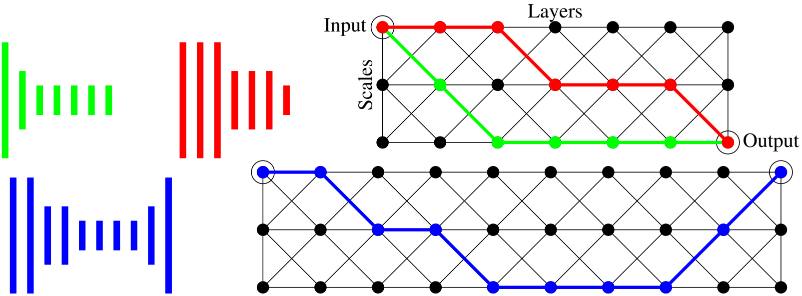
- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one



# Convolutional Neural Fabrics: Conclusion

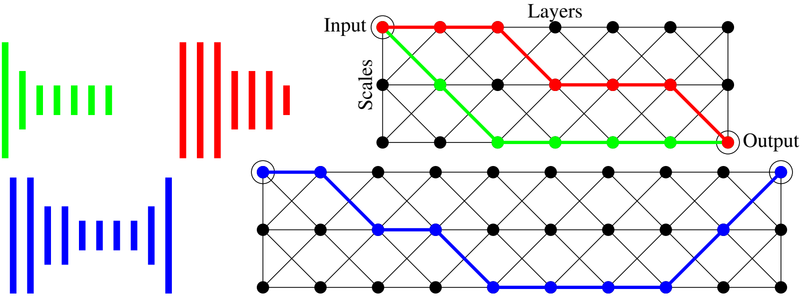
- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one

- From hand-crafted to learned features



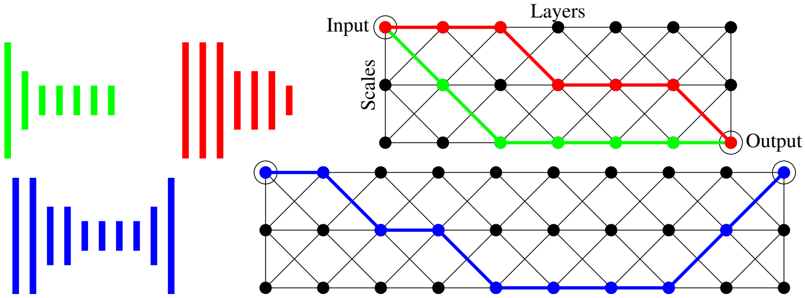
# Convolutional Neural Fabrics: Conclusion

- 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

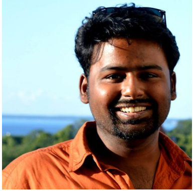
- Fabrics are universal architecture for conv nets
  - Fabric learns across architectures instead of selecting one
- From hand-crafted to learned ~~features~~ *architectures*
- Ongoing and future work
  - Scaling from hundreds to thousands of channels
  - Long-range connections across channels and layers
  - Scale invariance by convolution along scale axis



# Convolutional Neural Fabrics

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

# Thank you !



Neural Information Processing Systems 2016

# References I

[Chang and Chen, 2015] Chang, J.-R. and Chen, Y.-S. (2015).
Batch-normalized maxout network in network.
Arxiv preprint.

[Chen et al., 2016] Chen, T., Goodfellow, I., and Shlens, J. (2016).
Net2net: Accelerating learning via knowledge transfer.
In *ICLR*.

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013).
Learning hierarchical features for scene labeling.
*PAMI*, 35(8):1915–1929.

[Gaidon et al., 2016] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016).
Virtual worlds as proxy for multi-object tracking analysis.
In *CVPR*.

[Goodfellow et al., 2013] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013).
Maxout networks.
In *ICML*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Identity mappings in deep residual networks.
In *ECCV*.

[Kae et al., 2013] Kae, A., Sohn, K., Lee, H., and Learned-Miller, E. (2013).
Augmenting CRFs with Boltzmann machine shape priors for image labeling.
In *CVPR*.

[Kokkinos, 2016] Kokkinos, I. (2016).
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


[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012).
Imagenet classification with deep convolutional neural networks.
In *NIPS*.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015).
Fully convolutional networks for semantic segmentation.
In *CVPR*.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. (2010).
Rectified linear units improve restricted Boltzmann machines.
In *ICML*.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015).
Learning deconvolution network for semantic segmentation.
In *ICCV*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015).
U-net: Convolutional networks for biomedical image segmentation.
In *Medical Image Computing and Computer-Assisted Intervention*.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
In *ICLR*.

[Tsogkas et al., 2015] Tsogkas, S., Kokkinos, I., Papandreou, G., and Vedaldi, A. (2015).
Deep learning for semantic part segmentation with high-level guidance.
Arxiv preprint.

[Wan et al., 2013] Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. (2013).
Regularization of neural networks using DropConnect.
In *ICML*.

# Fabric analysis

- Comparing fabric variants: maps, parameters, activations

| # chan. / scale | # resp. maps | # parameters (sparse) | # parameters (dense) | # activations |
|---|---|---|---|---|
| constant | $C \cdot L \cdot S$ | $C \cdot L \cdot 3^{D+1} \cdot 3 \cdot S$ | $C \cdot L \cdot 3^{D+1} \cdot C \cdot S$ | $C \cdot L \cdot N^2 \cdot \frac{4}{3}$ |
| doubling | $C \cdot L \cdot 2^S$ | $C \cdot L \cdot 3^{D+1} \cdot 3 \cdot 2^S$ | $C \cdot L \cdot 3^{D+1} \cdot C \cdot 4^S \cdot \frac{7}{18}$ | $C \cdot L \cdot N^2 \cdot 2$ |