Maths of AI - Backbone Networks

Instructor - Simon Lucey

THE UNIVERSITY of ADELAIDE

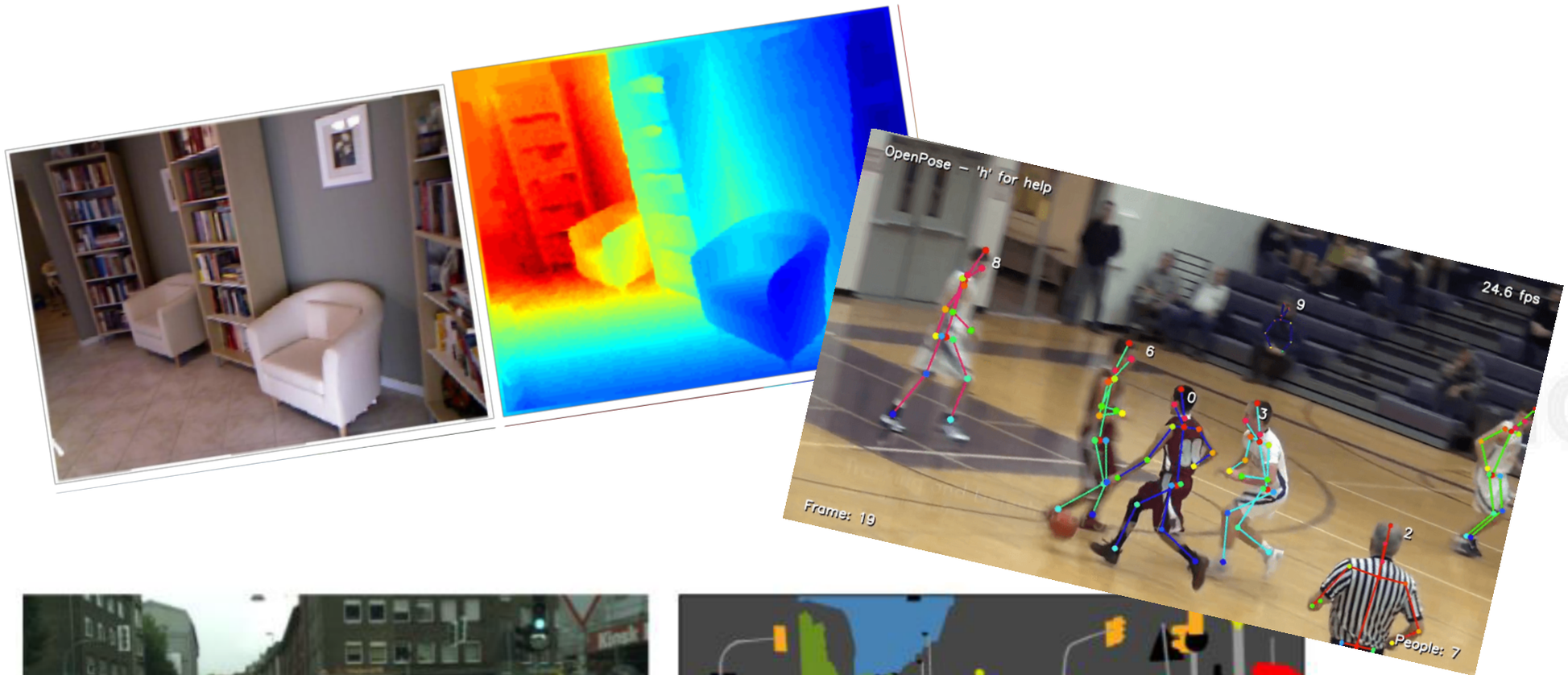
AUSTRALIAN INSTITUTE FOR MACHINE LEARNING

Today

- Backbone Networks
- VGGNet
- Residual Networks
- Attention versus Convolution

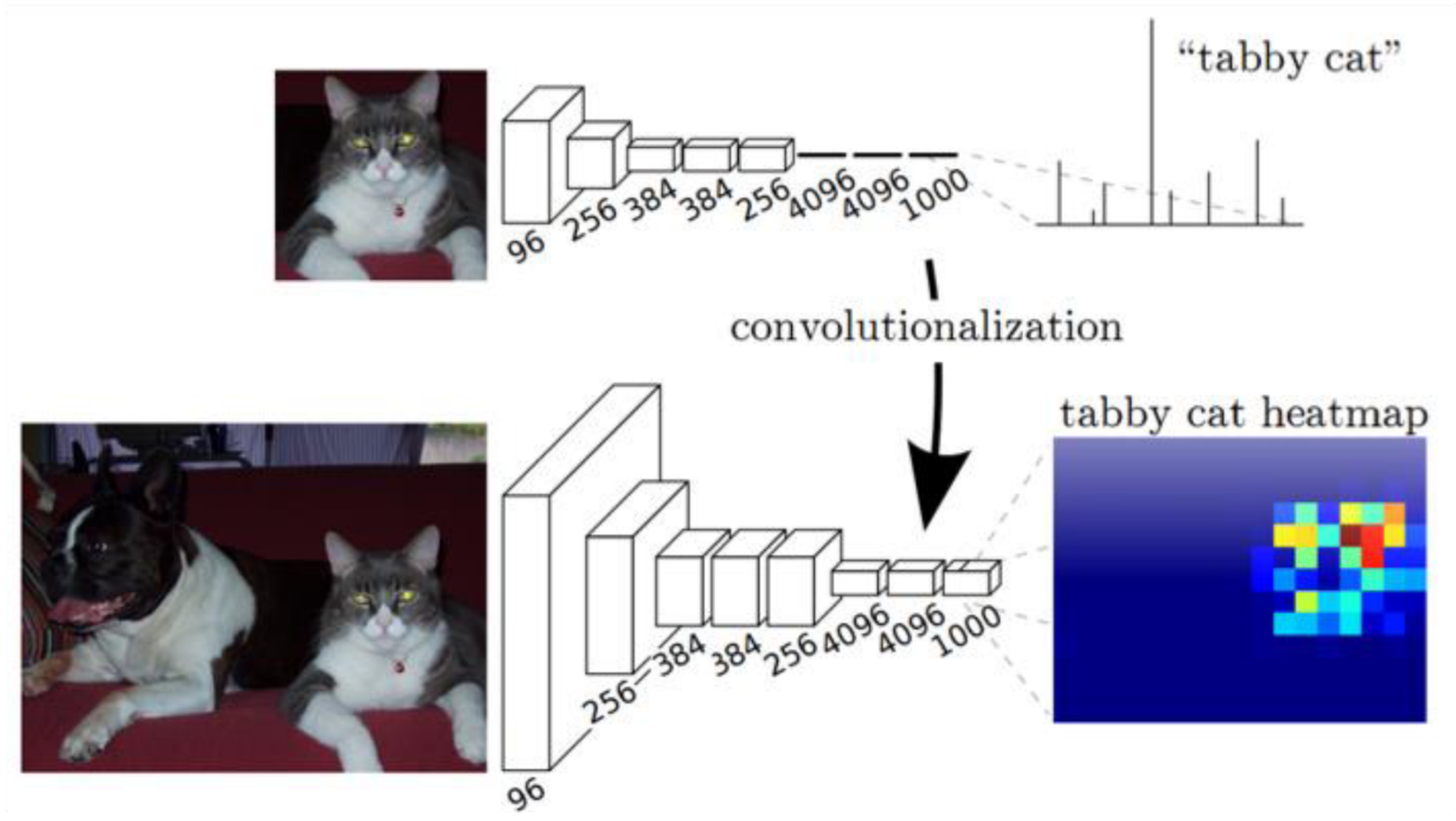
Backbone Networks

- Computer and robotic vision rely upon so-called “backbone” networks for many tasks.

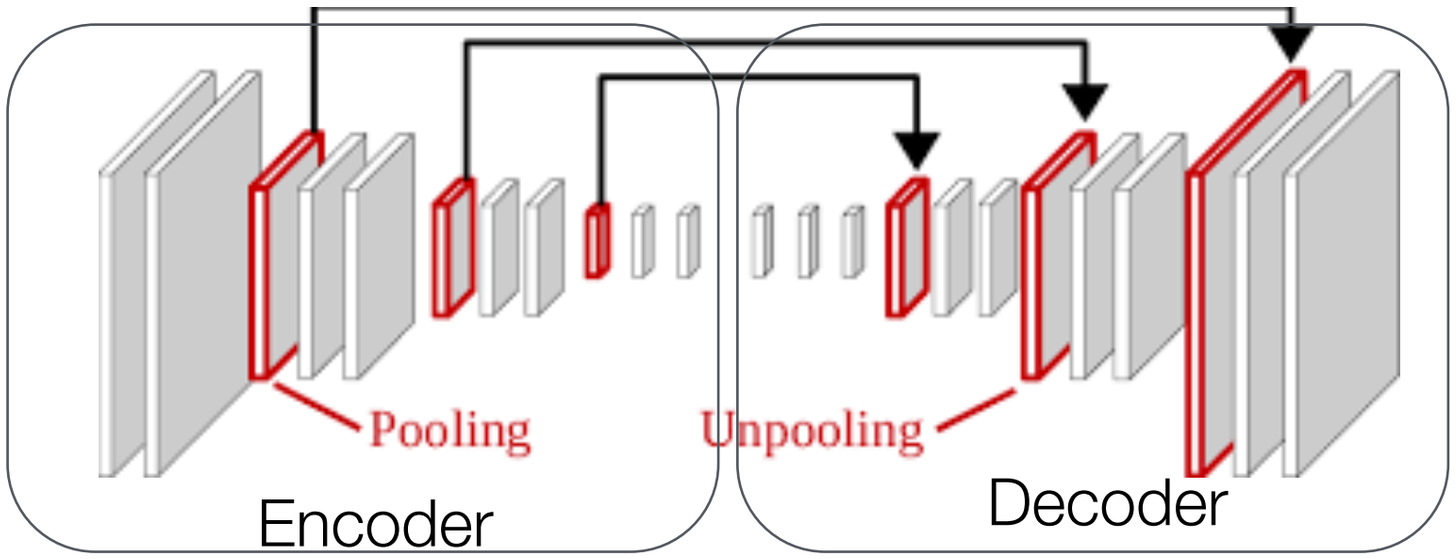


Fully Convolutional Network

- Converting fully connected layer to convolutions.
- Remember convolution = Sliding window detector.

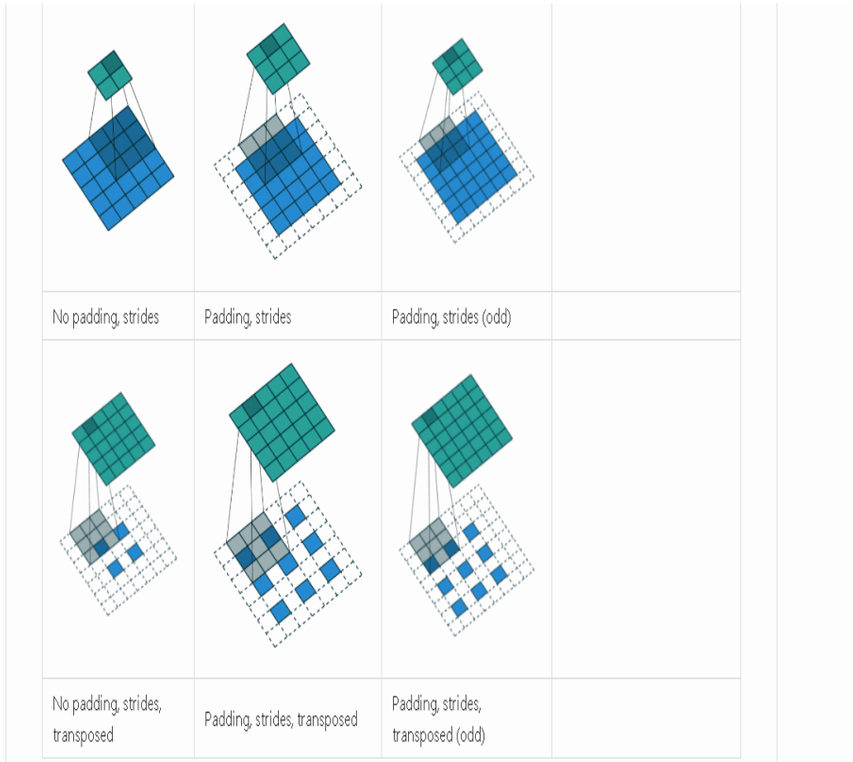


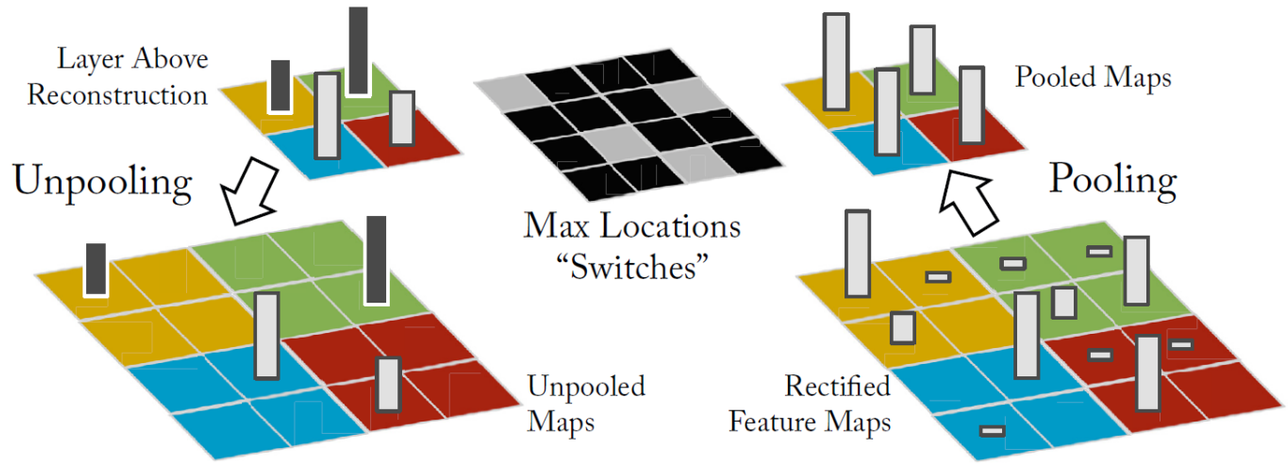
Too Coarse? Deconvolutional-Net



- Encoder: Create coarse abstract representation using successive convolutions and down-samplings.
- Decoder: Decodes features with Deconvolutions and Unpooling.
- Skip Connections : Reuse finer details that were lost in successive downsamplings of encoder.

Deconvolution and Unpooling

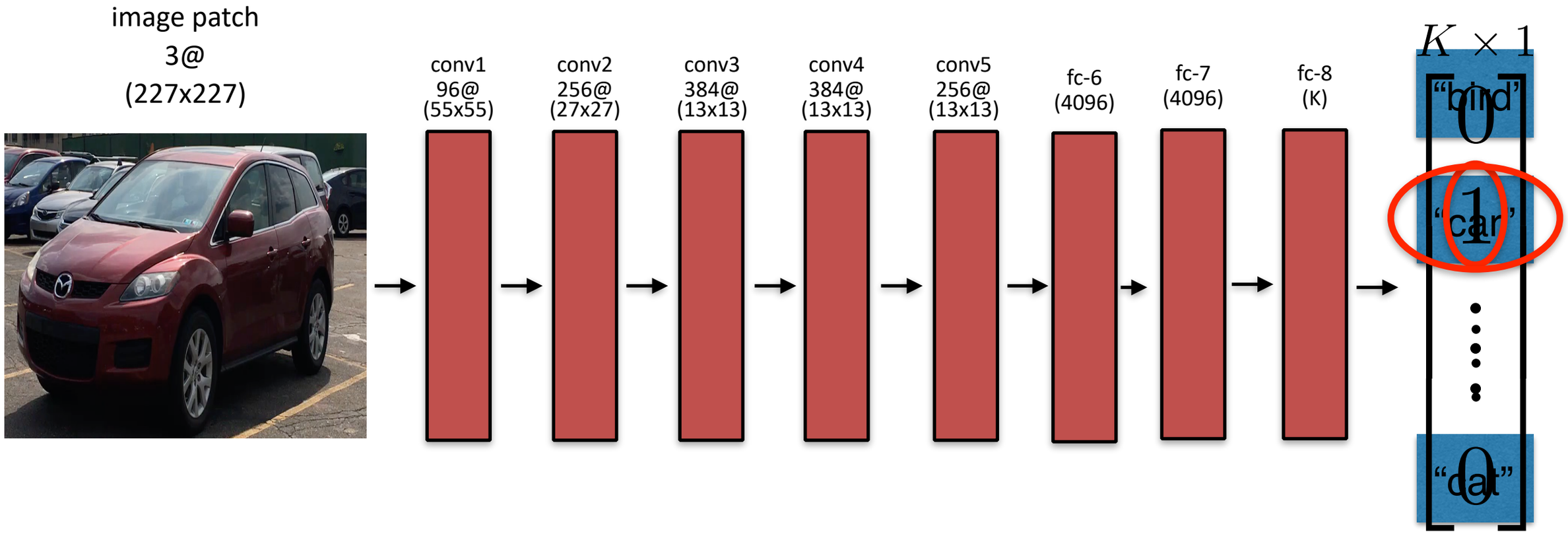




- Intuitively, invert the Convolutions and Subsampling operations.
  - Convolution to Transpose Convolution
  - Pooling to Unpooling

AlexNet - The Original Backbone

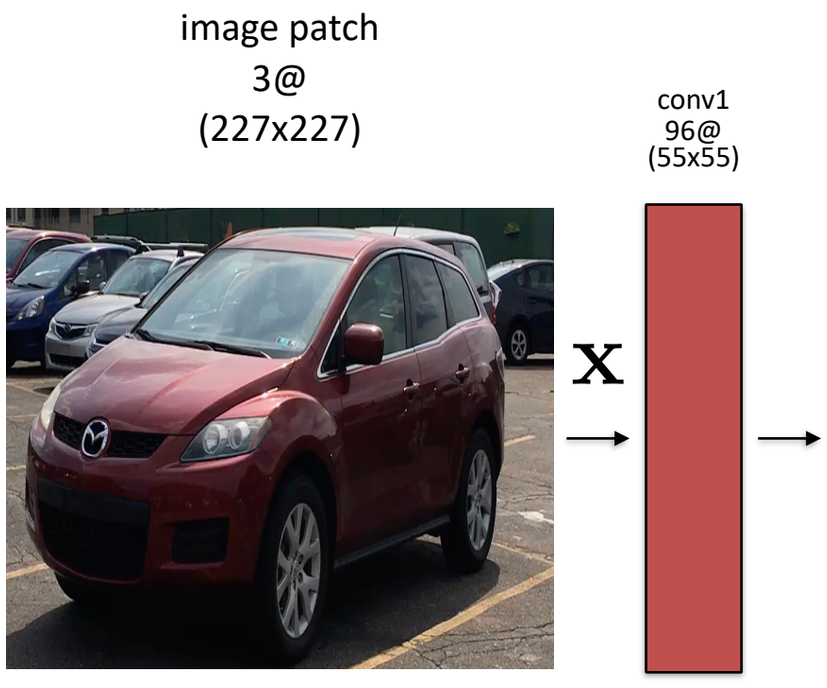
- AlexNet won the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%. (Second best was 26.2 %).
- Network has 25 layers, but only 8 layers with learnable weights.
  - 5 convolutional weights.
  - 3 fully connected weights.



A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

AlexNet Drawbacks

- AlexNet gave amazing performance but had some drawbacks.
- Fundamentally, it is extremely expensive to evaluate & learn.



- 55x55x96 = 290,400 hidden states
- each has 11x11x3 = 363 weights and 1 bias
- 96x364 = 34,944 parameters on the first layer of the AlexNet alone!

Today

- Backbone Networks
- VGGNet
- Residual Networks
- Attention versus Convolution

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

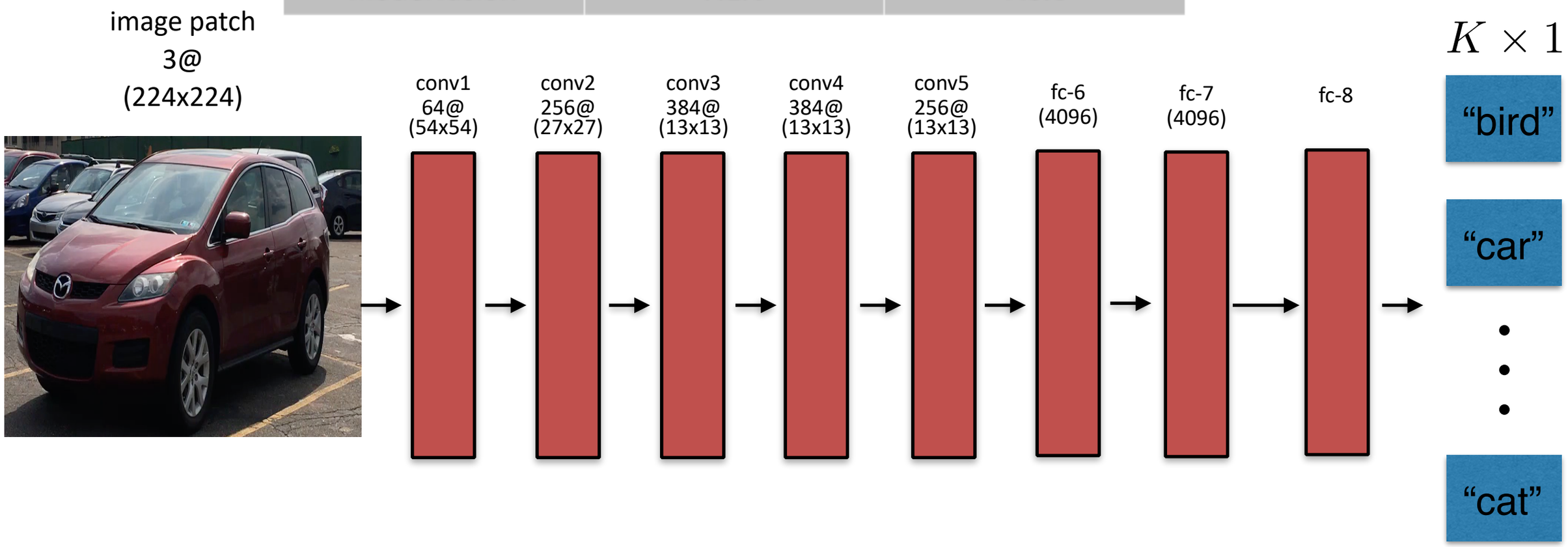
In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

VGGNet

- Two versions VGG-16 & VGG-19.
  - VGG-16 has 41 layers, with 16 learnable layers (13 convolutional & 3 fully connected).
  - VGG-19 has 47 layers, with 19 learnable layers (16 convolutional & 3 fully connected).

Model	top-5 classification error on ILSVRC-2012 (%)	
	validation set	test set
16-layer	7.5%	7.4%
19-layer	7.5%	7.3%
model fusion	7.1%	7.0%

"AlexNet = 15.3%"



K. Hatfield and A. Zisserman. "Very Deep Networks for Large Scale Image Recognition." In ICLR, 2015.

VGGNet

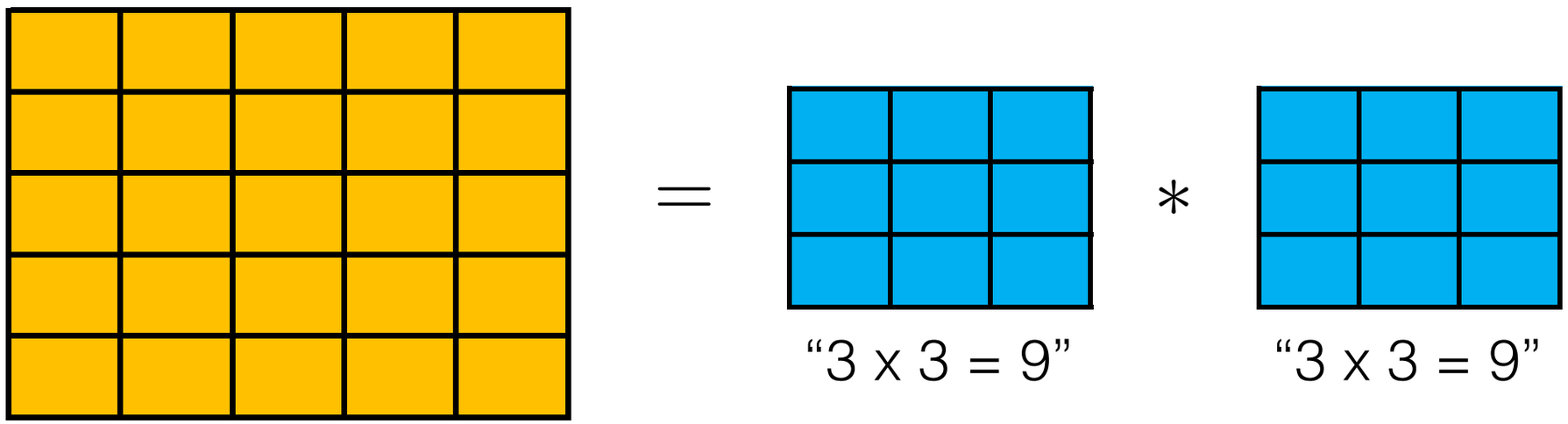
- VGGNet proposed by the Vision Geometry Group (VGG) at Oxford.
- Overall idea is to replace complex layers with multiple simple layers.
- In particular, explored a neat idea around the associative properties of convolution.

$$\mathbf{g} * (\mathbf{h} * \mathbf{x}) = (\mathbf{g} * \mathbf{h}) * \mathbf{x}$$ **(associative)**

$$\underbrace{\begin{bmatrix} 6 \\ -12 \\ -48 \end{bmatrix}}_{(\mathbf{g} * \mathbf{h})} = \underbrace{\begin{bmatrix} 2 \\ -8 \end{bmatrix}}_{\mathbf{g}} * \underbrace{\begin{bmatrix} 3 \\ 6 \end{bmatrix}}_{\mathbf{h}}$$

```
>>> from scipy.signal import convolve as conv
>>> conv(g,h,'full')
    array([  6, -12, -48])
```

VGGNet

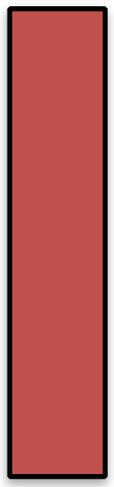


$$25 > 9 + 9$$

VGGNet

image patch
3@
(224x224)

conv1
64@
(54x54)



K. Hatfield and A. Zisserman. "Very Deep Networks for Large Scale Image Recognition." In ICLR, 2014.

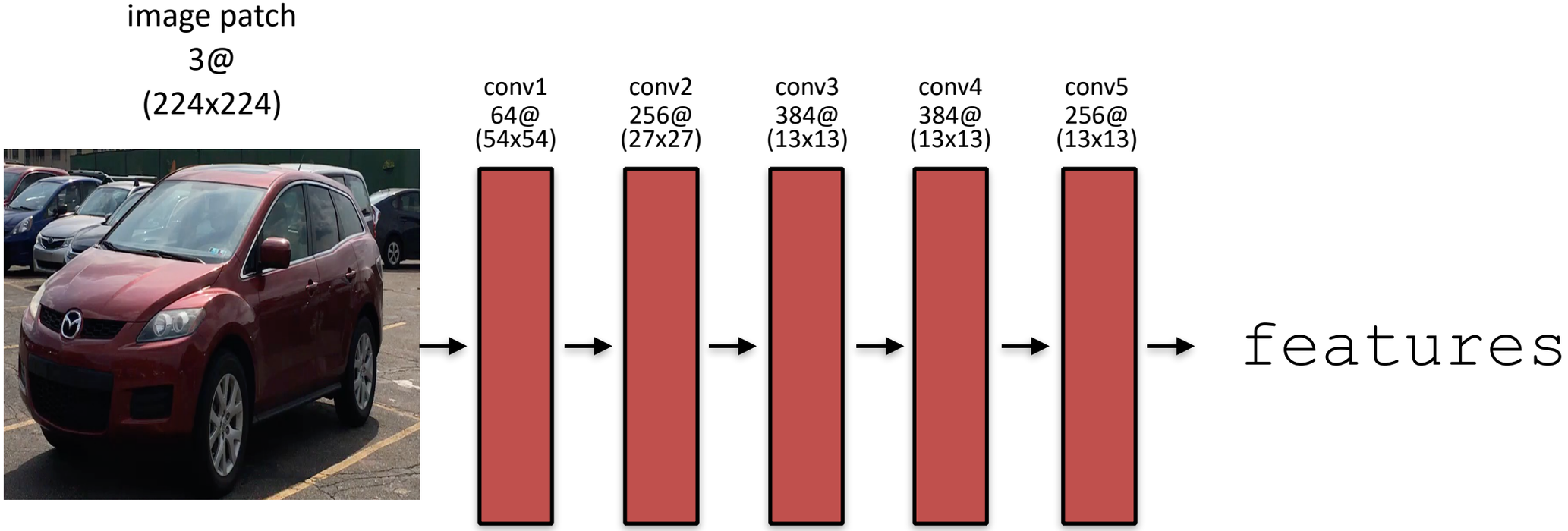
VGG in PyTorch

```
>>> net = models.vgg16(pretrained=True)
>>> net.features
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace)
  (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (18): ReLU(inplace)
  (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace)
  (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (25): ReLU(inplace)
  (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (27): ReLU(inplace)
  (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (29): ReLU(inplace)
  (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

VGG in PyTorch

```
>>> net.classifier
Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace)
  (2): Dropout(p=0.5)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace)
  (5): Dropout(p=0.5)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

VGG Features in PyTorch

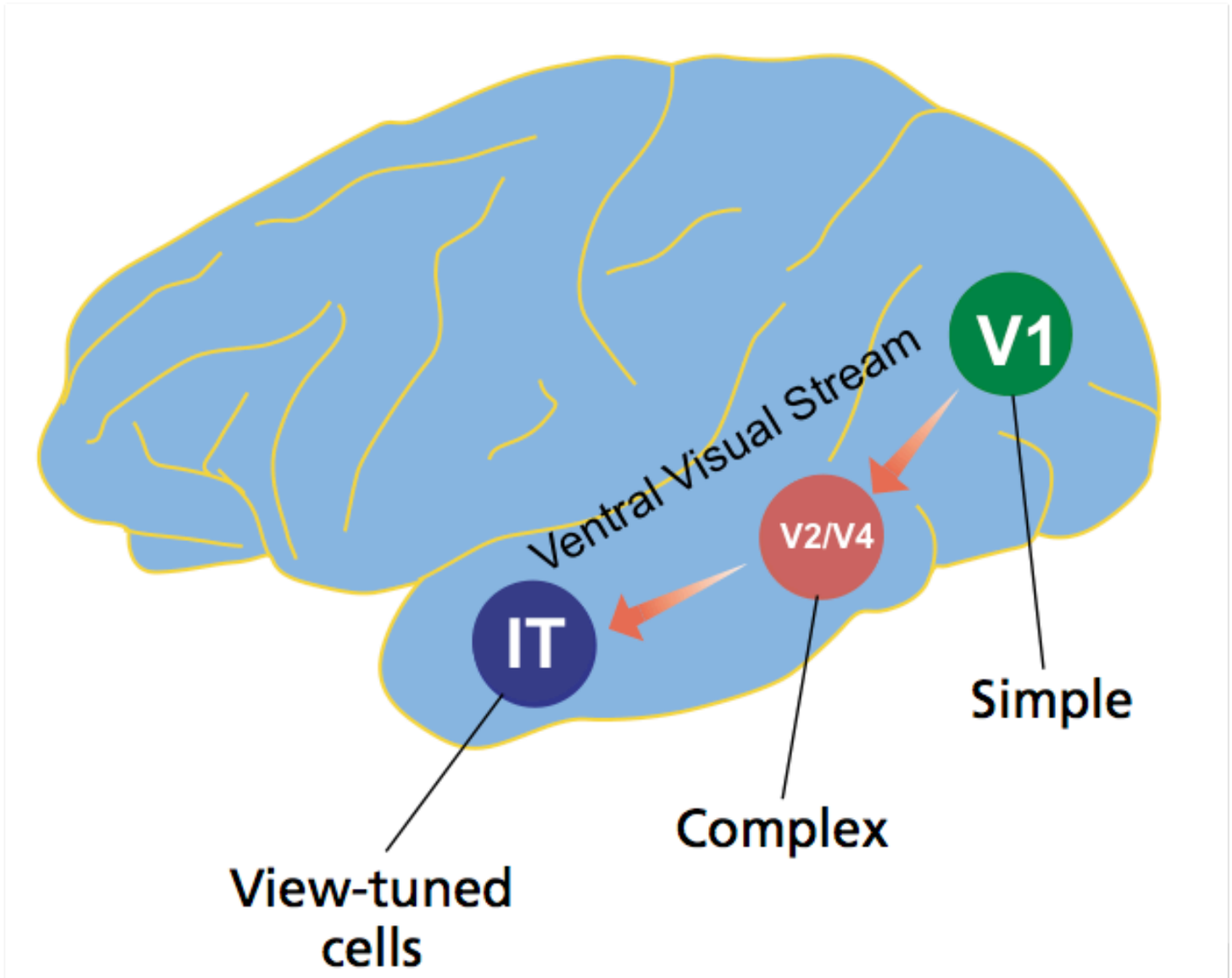



```
>>> rgb = torch.randn(1,3,224, 224)
>>> features = net.features(rgb)
>>> features.shape
    torch.Size([1, 512, 7, 7])

>>> features.flatten().shape
    torch.Size([25088])
```

Generalizability?

- How is it that a network learned for one type of task (e.g. object recognition) can generate representations that work for many different visual tasks?



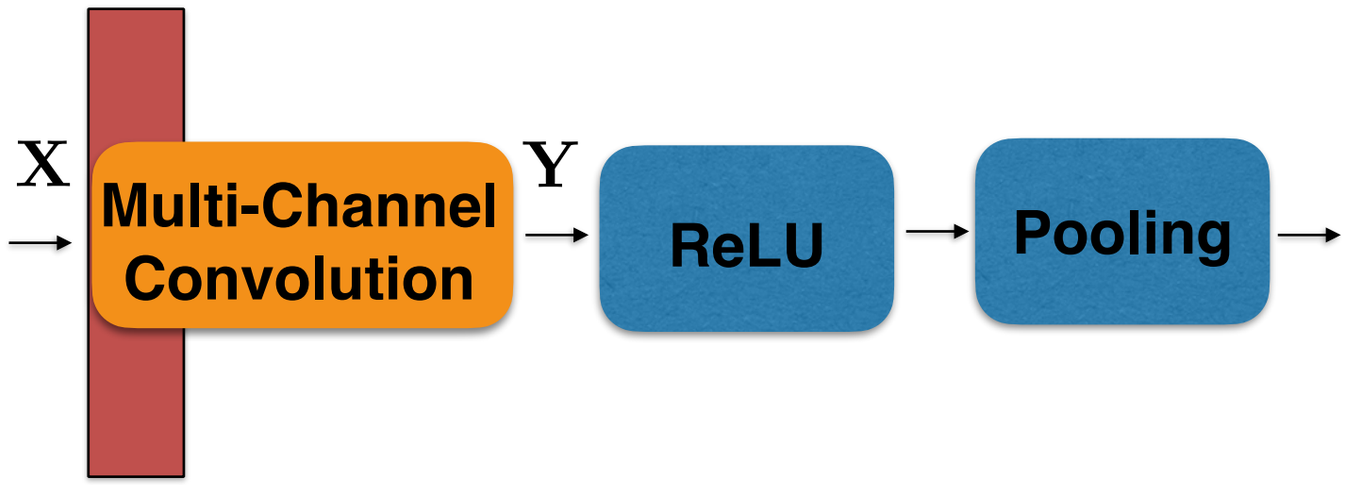
Reminder: Multi-Channel Convolution

image patch
3@
(227x227)

conv1
96@
(55x55)

Assuming: $\mathbf{X} \in \mathbb{R}^{M \times N}$

$\mathbf{H} \in \mathbb{R}^{D \times D}$



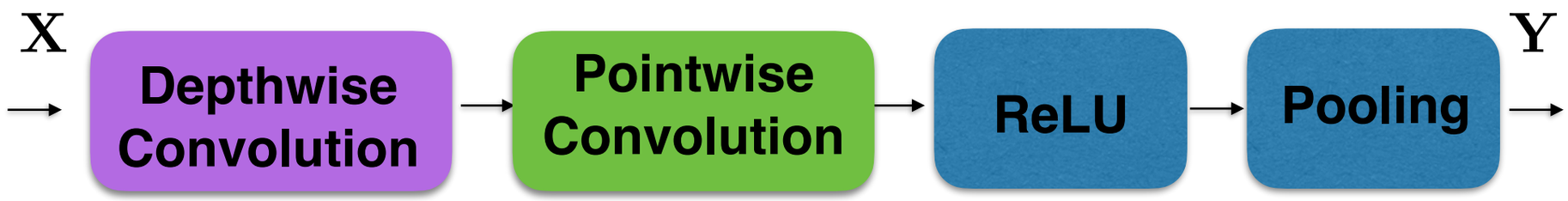
$$\mathbf{Y}^{(j)} = \sum_{k=1}^{K} \mathbf{X}^{(k)} * \mathbf{H}^{(j,k)}$$

Naive computational cost - $\mathcal{O}(M \cdot N \cdot D^2 \cdot K \cdot J)$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Trends: Depthwise Convolution

image patch
3@
(227x227)

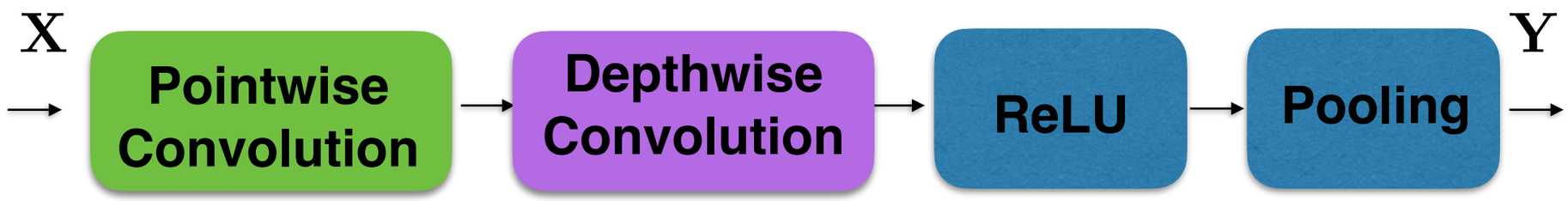


$$\mathbf{Y}^{(j)} = \sum_{k=1}^{K} \underbrace{\mathbf{W}[j,k]}_{\text{“1x1 Pointwise Convolution”}} \cdot [\underbrace{\mathbf{X}^{(k)} * \mathbf{H}^{(k)}}_{\text{“Depthwise Convolution”}}]$$

$$\mathcal{O}(M \cdot N \cdot D^2 \cdot J \cdot K) \rightarrow \mathcal{O}(M \cdot N \cdot (D^2 + J) \cdot K)$$

Trends: Depthwise Convolution

image patch
3@
(227x227)



$$\mathbf{Y}^{(j)} = \underbrace{\mathbf{H}^{(j)}}_{\text{“1x1 Pointwise Convolution”}} * [\sum_{k=1}^{K} \underbrace{\mathbf{W}[j,k] \cdot \mathbf{X}^{(k)}}_{\text{“Depthwise Convolution”}}]$$

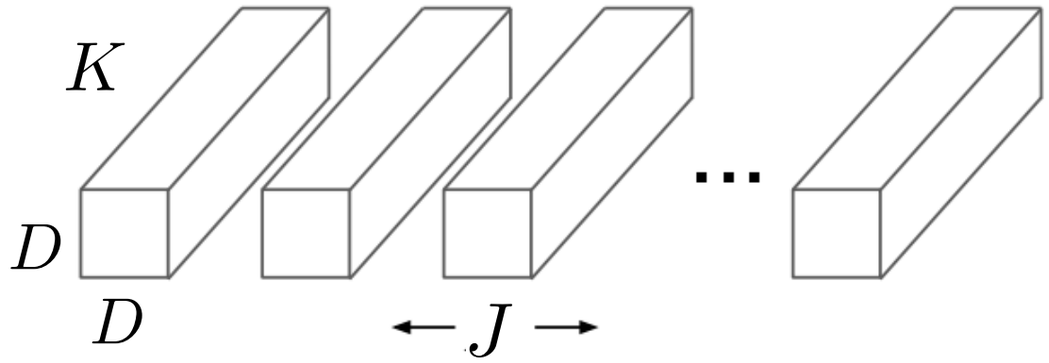
$$\mathcal{O}(M \cdot N \cdot D^2 \cdot J \cdot K) \rightarrow \mathcal{O}(M \cdot N \cdot (D^2 + J) \cdot K)$$

MobileNet

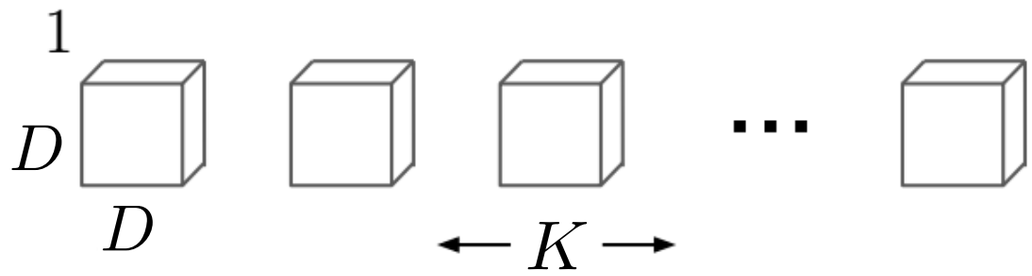
- MobileNet: (a) is divided into (b) and (c).
- Computation reduction

$$\frac{b+c}{a} = \frac{D^2 + J}{D^2 \cdot J}$$

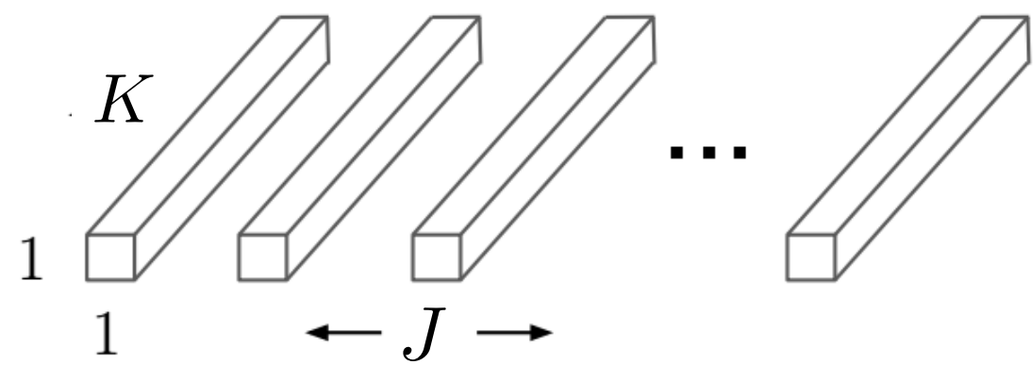
- D = 3: 8~9 times less computation.



(a) Standard convolution filters



(b) Depthwise convolution filters



(c) 1x1 pointwise convolution

Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

MobileNet

- MobileNet with similar accuracy but less computation and fewer parameters than VGG16 and GoogleNet
- $\alpha = 0.5$, input 160×160: better than AlexNet while being 45 times smaller and 9.4 times less compute than AlexNet

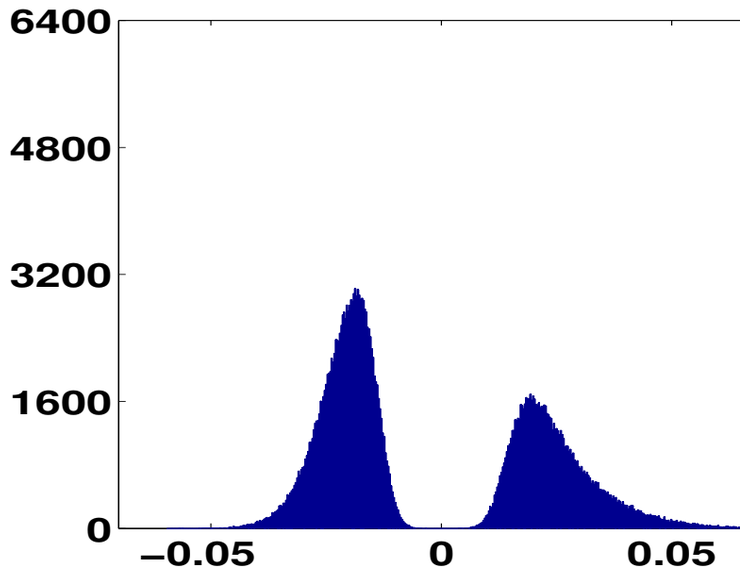
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Trends - Lower Precision

Reducing Precision

- Saving Memory
- Saving Computation

[Han et al. 2016]



$$\mathbb{R} \rightarrow \mathbb{I} \rightarrow \mathbb{B} \in \{-1, +1\}$$

32-bit 8-bit 1-bit

{-1,+1}	{0,1}
MUL	XNOR
ADD, SUB	Bit-Count (popcount)

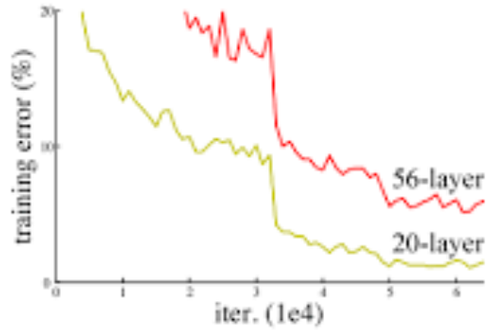
Rastegari, Mohammad, et al. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks." ECCV 2016

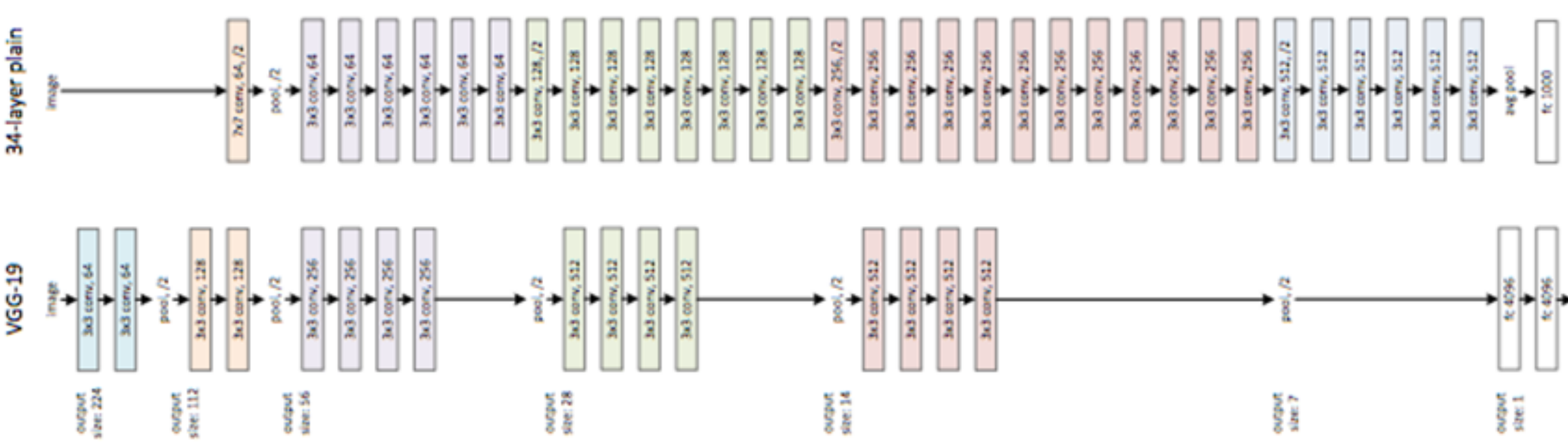
Today

- Backbone Networks
- VGGNet
- Residual Networks
- Attention versus Convolution

Neural Net : Deeper is Better?

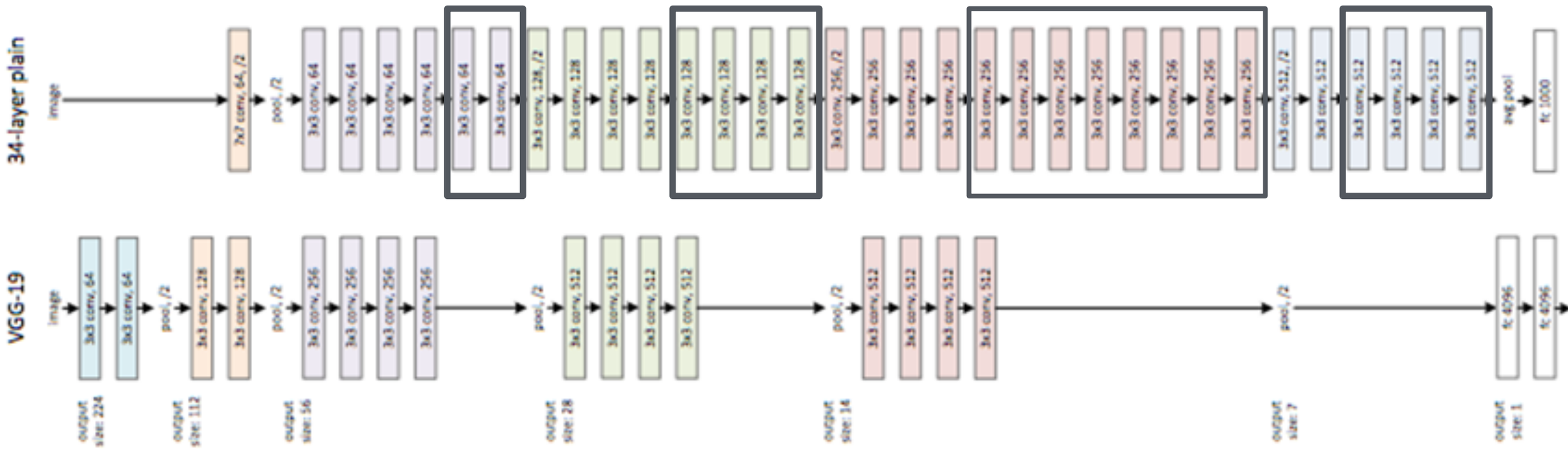
- Very deep network -- training error increases.





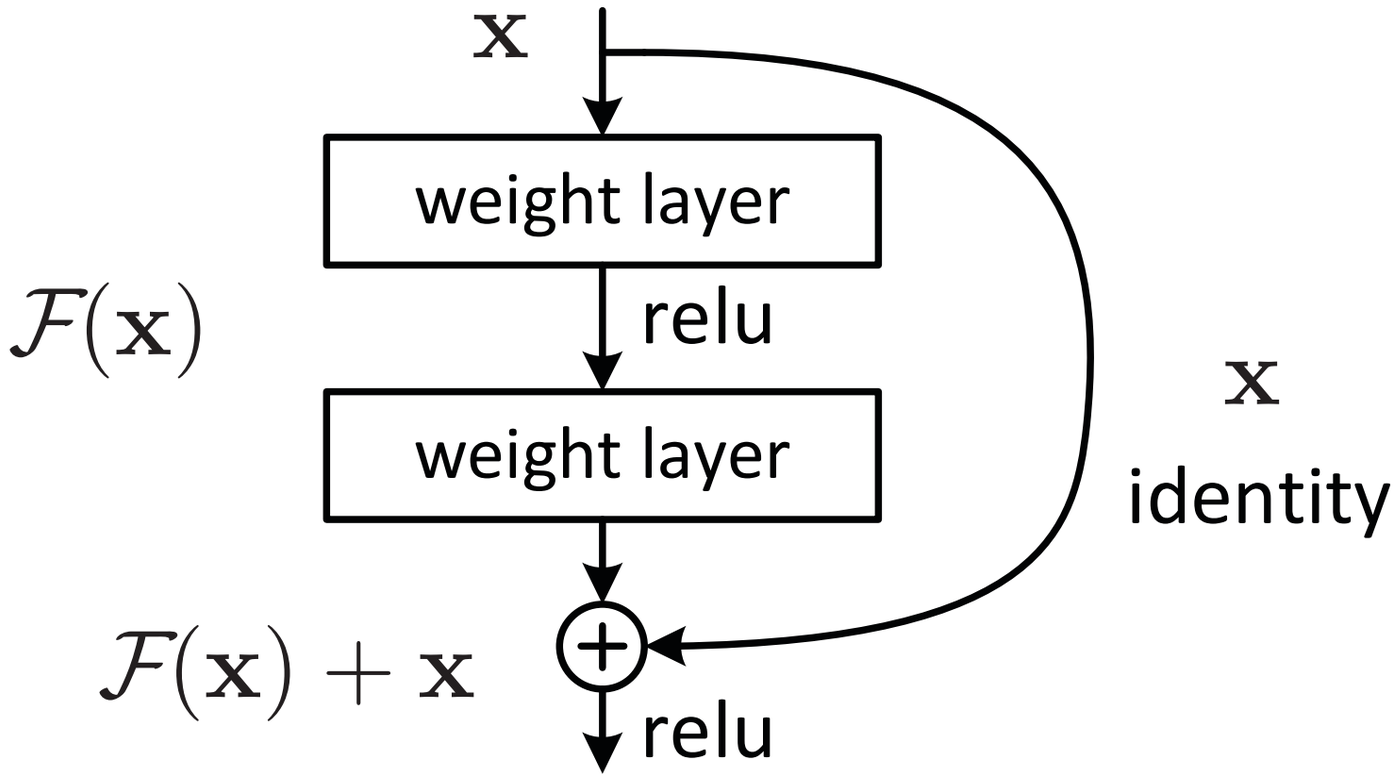
Neural Net : Deeper is Better?

- Very deep network -- training error increases.
- What if I just learn to do NOT A THING in some layer?
- 34 layer net then is just a 19 layer net with less loss!



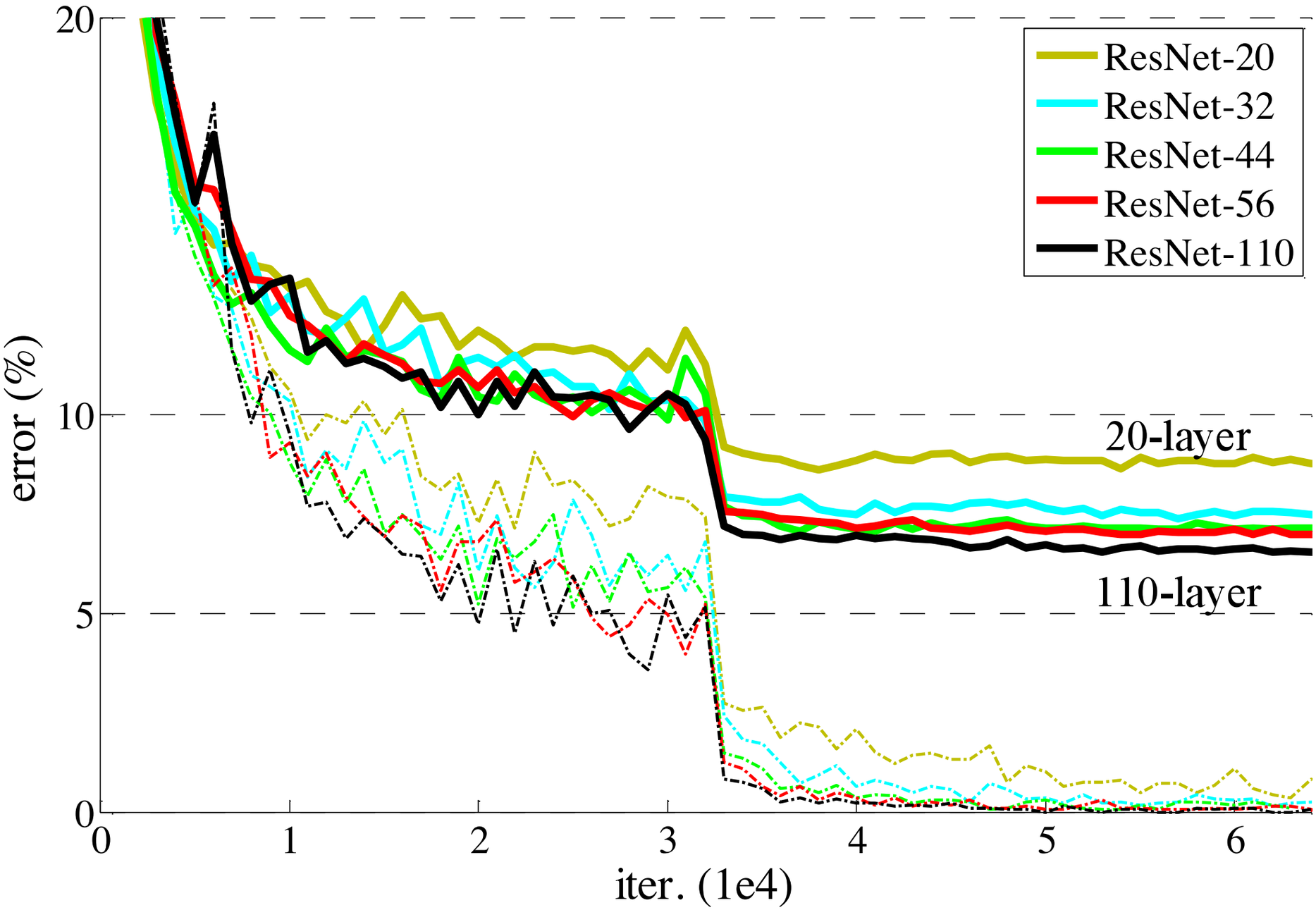
How Deep?

- Recent work has suggested that network depth is crucial for good performance (e.g. ImageNet).
- Counter intuitively, naively trained deeper networks tend to have higher train error than shallow networks.
- Innovation of residual learning has greatly helped with this.



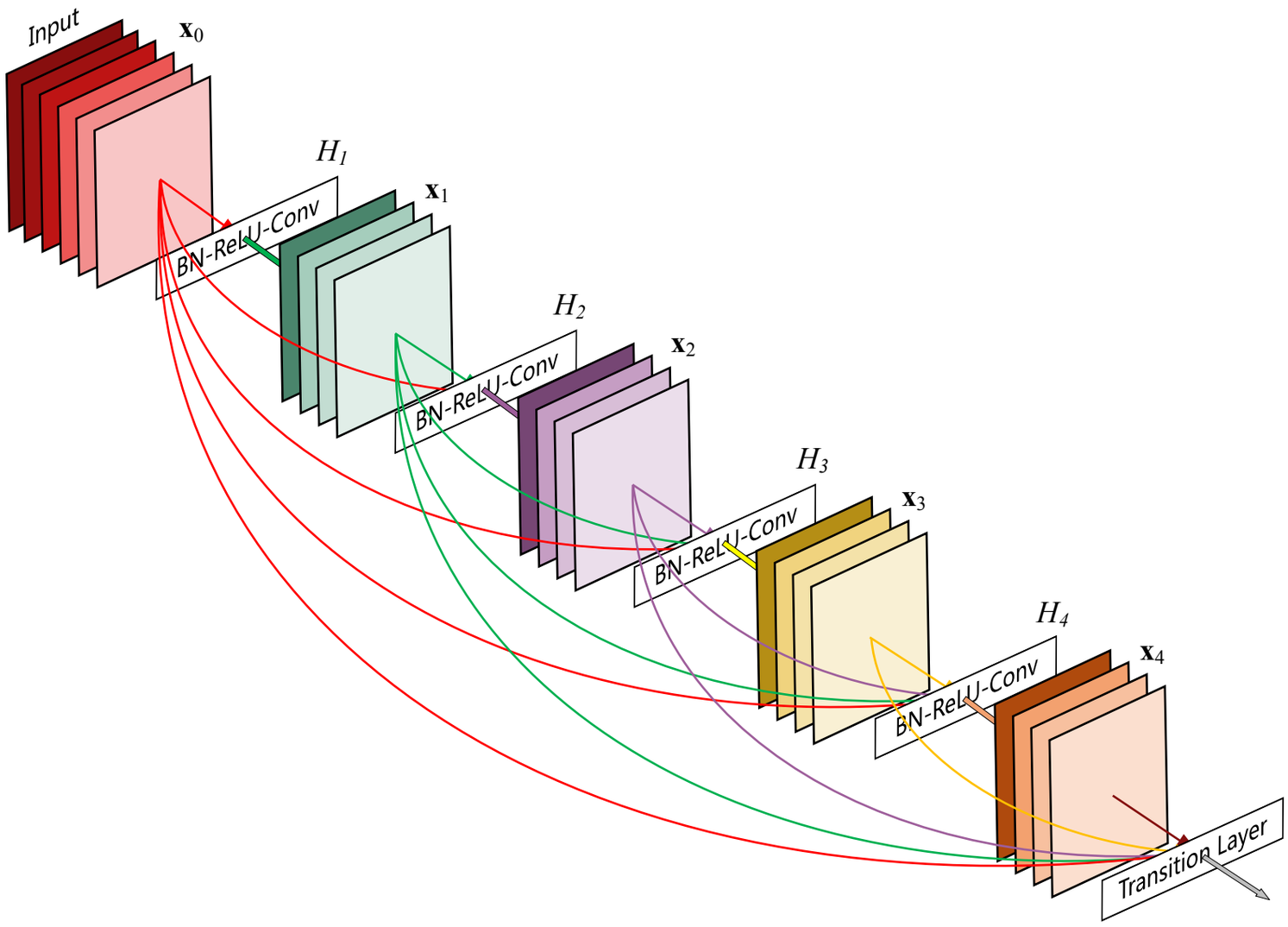
K. He, et al. "Deep residual learning for image recognition", CVPR 2016.

How Deep?



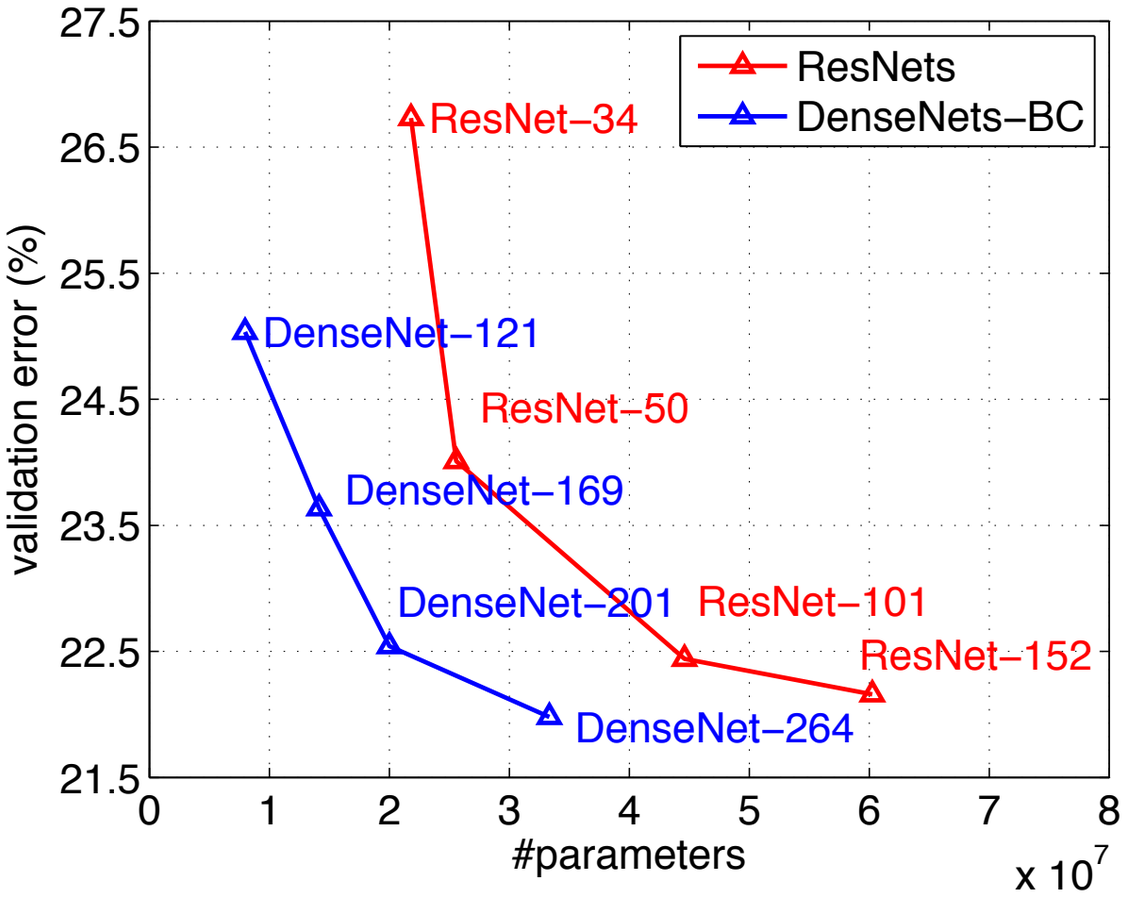
K. He, et al. "Deep residual learning for image recognition", CVPR 2016.

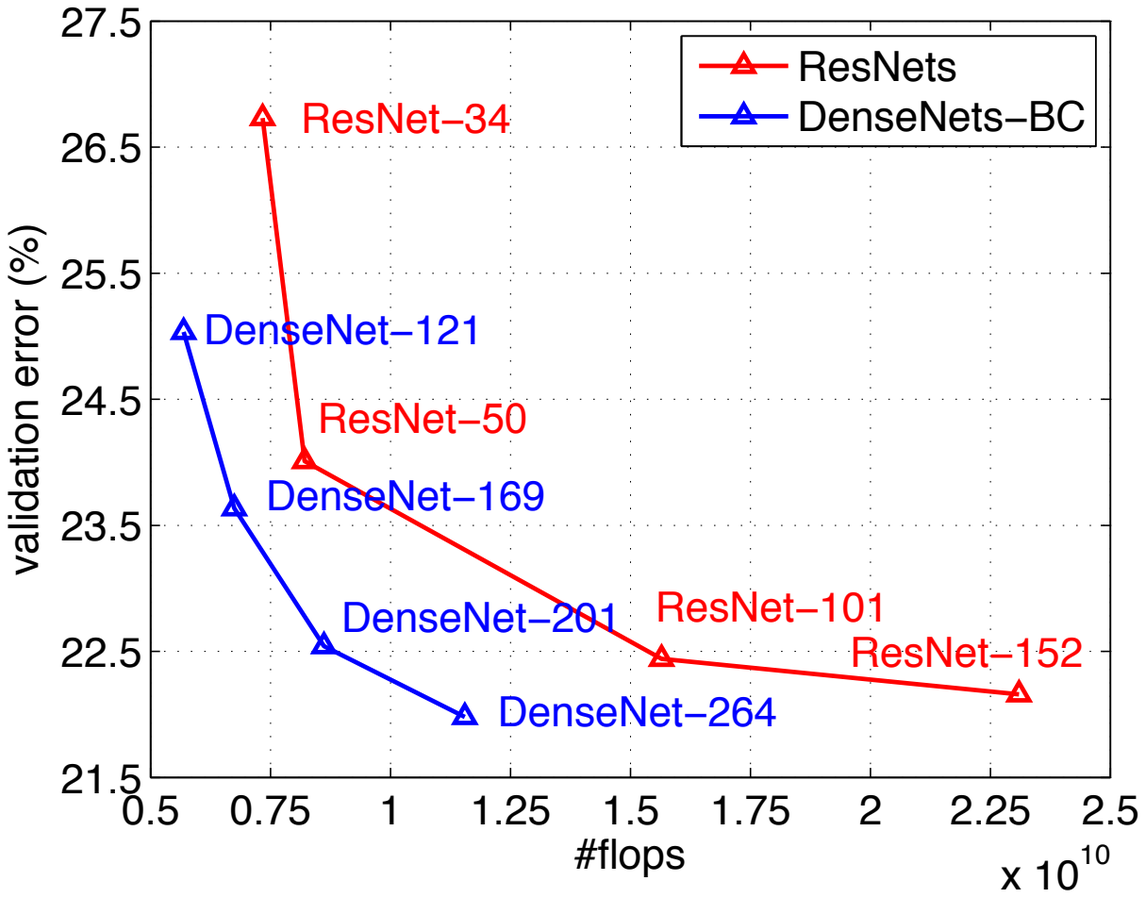
Densely Connected Networks



G. Huang et al. "Densely Connected Convolutional Networks", CVPR 2017.

Densely Connected Networks





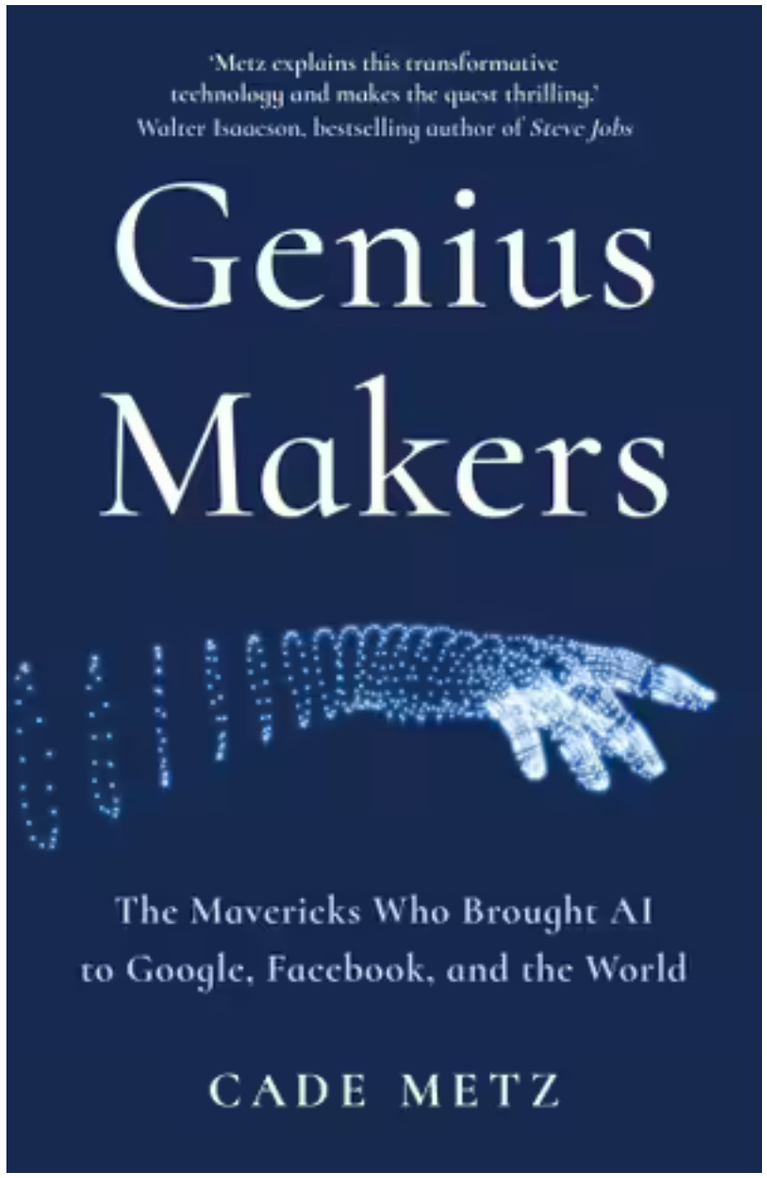
G. Huang et al. "Densely Connected Convolutional Networks", CVPR 2017.

Today

- Backbone Networks
- VGGNet
- Residual Networks
- Attention versus Convolution

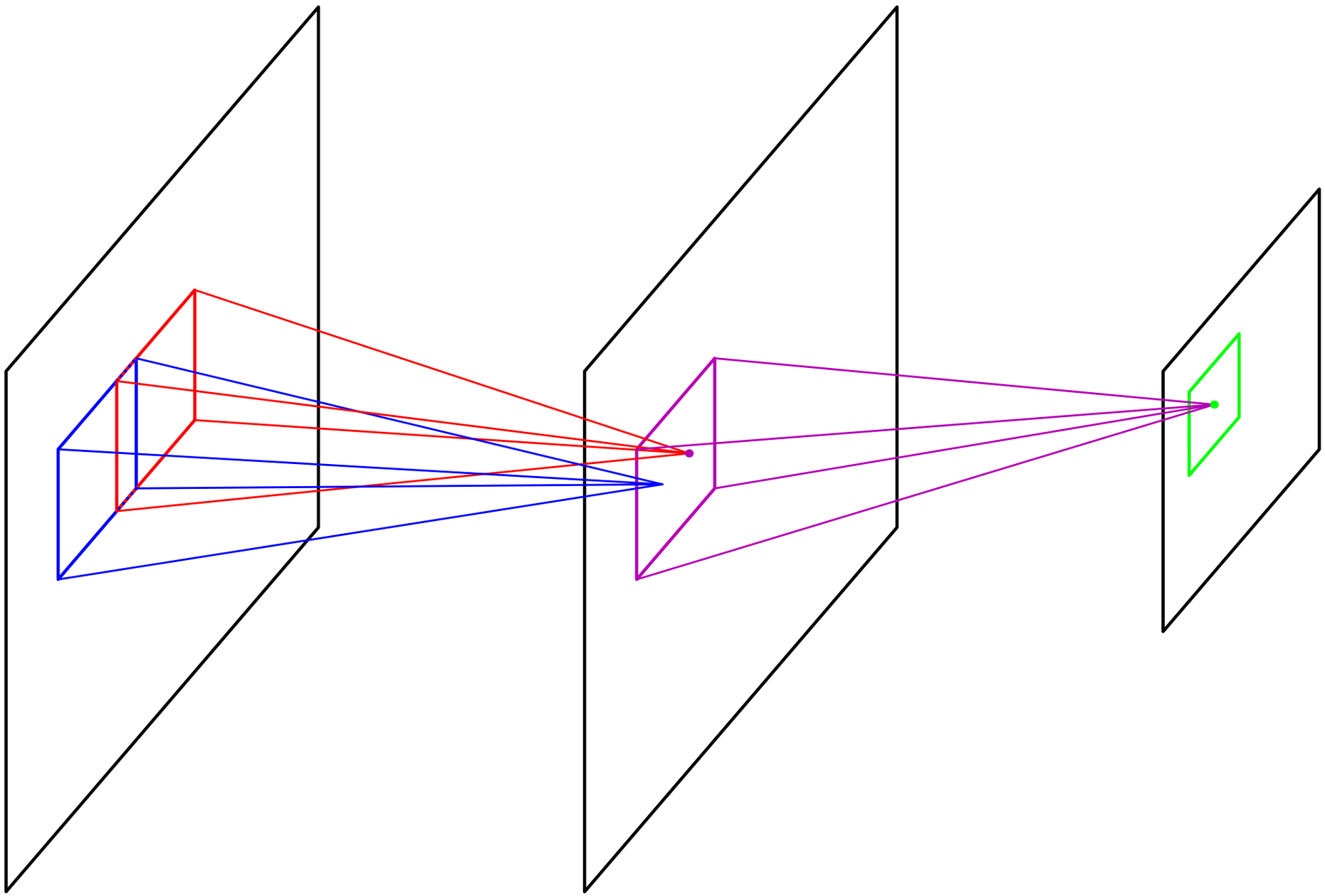
Reminder: Innate vs. Experience

Yoshua Bengio



Gary Marcus

Convolutional = A Type of Attention??



LeCun 1980

Attention is all you need!!

- Convolution is hard-coded attention.
- Could there be any way to learn it from data?

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention

A. Vaswani et al. “Attention is all you need”, NeurIPS 2017.

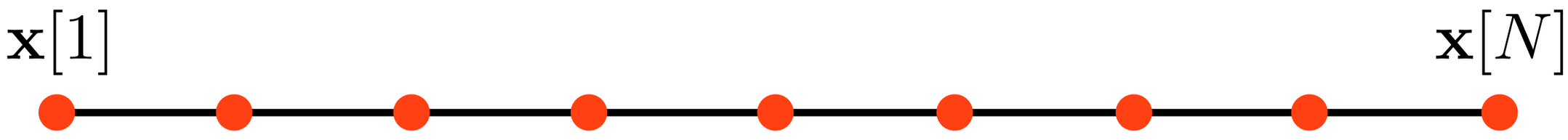
Trends: Depthwise Convolution

image patch
3@
(227x227)

Trends: Depthwise Convolution

image patch
3@
(227x227)

Convolution - Graph Laplacian



$$\mathbf{y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T = \mathbf{L} * \mathbf{x}$$

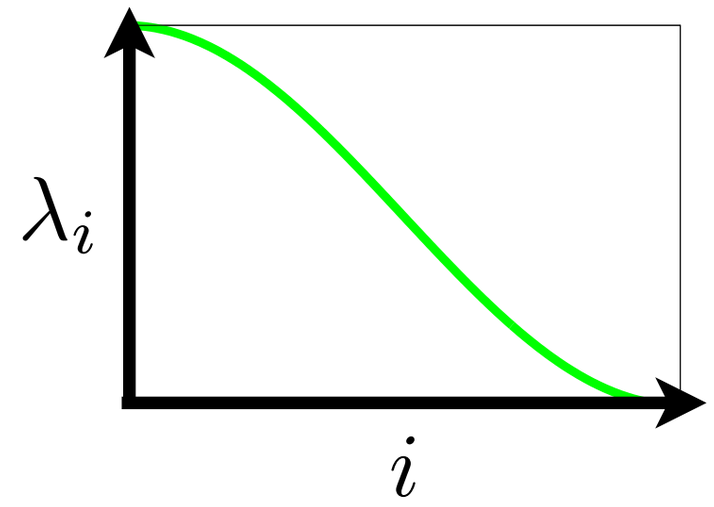
Graph Laplacian

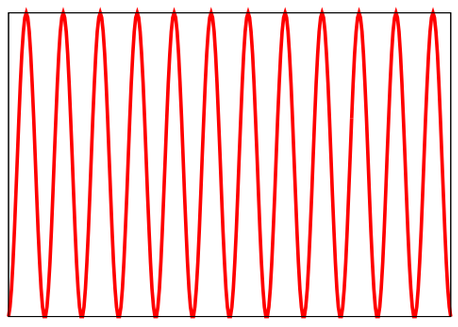
$$\mathbf{L} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T = \sum_i \lambda_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T$$

Graph Laplacian

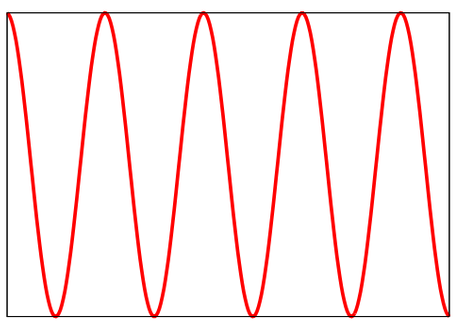
$$\mathbf{L} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\boldsymbol{T}} = \sum_i \lambda_i \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

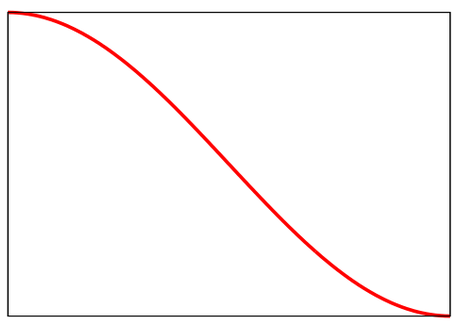




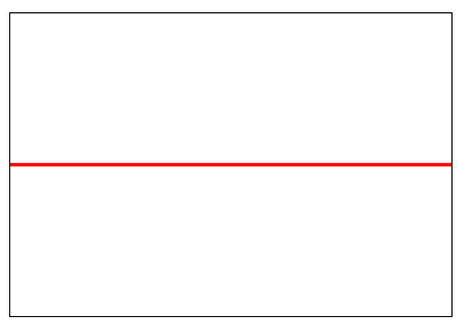
$\mathbf{\Phi}_1$



$\mathbf{\Phi}_2$



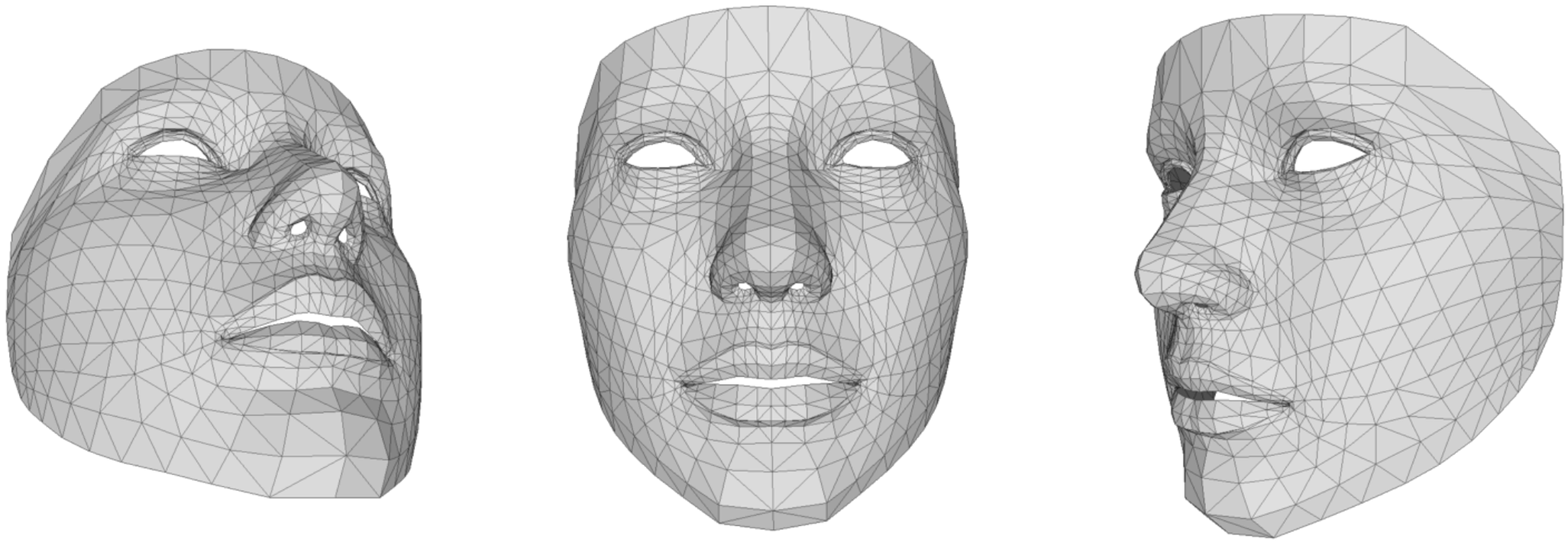
$\mathbf{\Phi}_{N-1}$



$\mathbf{\Phi}_N$

General Topology

What if $\mathbf{x} \neq$ grid?



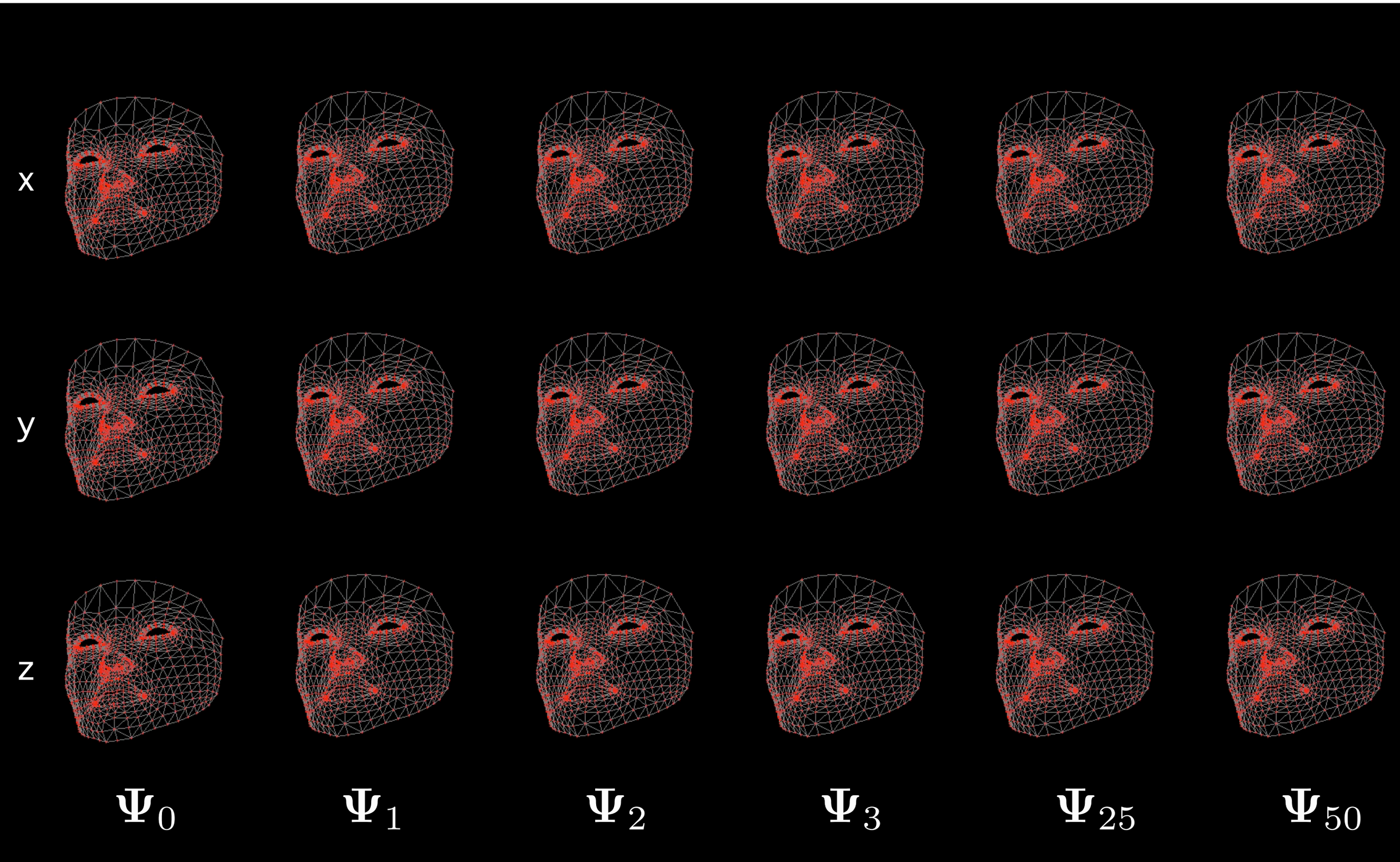
$$\mathbf{L}[i,j] = \begin{cases} |\mathcal{N}(i)| & \text{if } i = j \\ -1 & \text{if } j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

$$\mathbf{L}' = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$$

Normalized Graph Laplacian

Smooth Deformation Basis



Normalized Graph Laplacian

```
A =      ([[0., 0., 1., 1., 1.],
           [1., 0., 1., 0., 1.],
           [1., 0., 0., 0., 1.],
           [0., 1., 0., 0., 0.],
           [0., 0., 1., 1., 0.]])

L_norm = ([[1.0000, 1.0000, 0.5918, 0.4226, 0.5918],
           [0.6667, 1.0000, 0.5918, 1.0000, 0.5918],
           [0.5918, 1.0000, 1.0000, 1.0000, 0.5000],
           [1.0000, 0.4226, 1.0000, 1.0000, 1.0000],
           [1.0000, 1.0000, 0.5000, 0.2929, 1.0000]])
```

Graph Attention Networks

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

ABSTRACT

We present graph attention networks (GATs), novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, we address several key challenges of spectral-based graph neural networks simultaneously, and make our model readily applicable to inductive as well as transductive problems. Our GAT models have achieved or matched state-of-the-art results across four established transductive and inductive graph benchmarks: the *Cora*, *Citeseer* and *Pubmed* citation network datasets, as well as a *protein-protein interaction* dataset (wherein test graphs remain unseen during training).

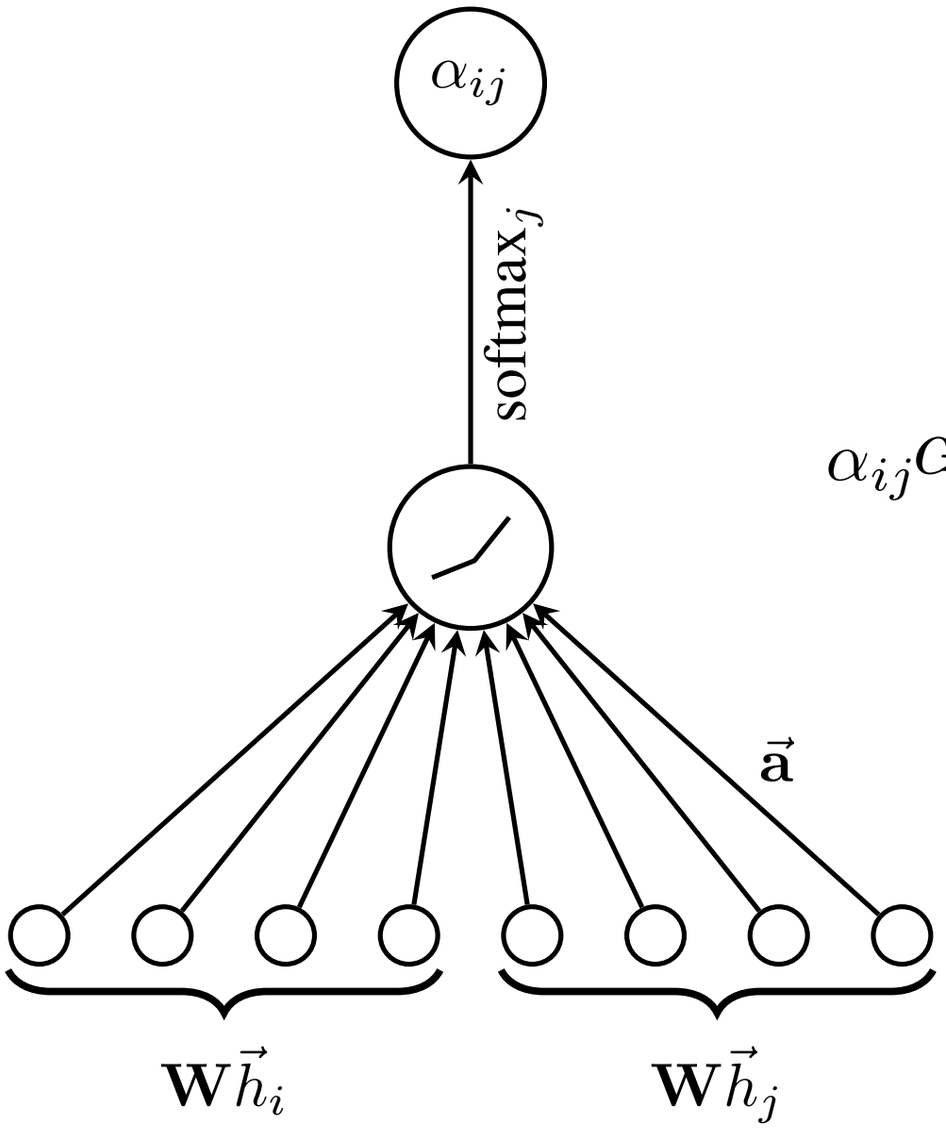
A. Velickovic et al. “Graph Attention Networks”, ICLR 2018.

What is Self Attention

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

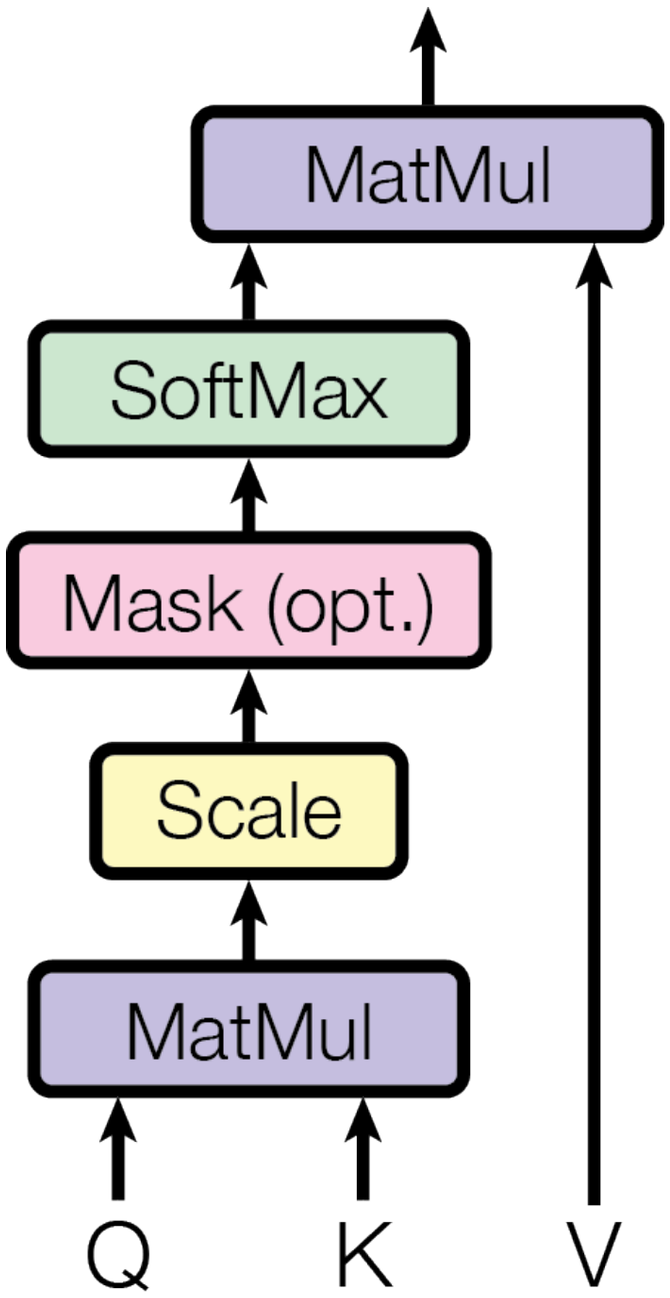
$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right)$$

A. Velickovic et al. “Graph Attention Networks”, ICLR 2018.

What is Attention?



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Problem: Attention scales quadratically with the number of pixels!!!

A. Vaswani et al. “Attention is all you need”, NeurIPS 2017.

Vision Transformer (ViT)

Published as a conference paper at ICLR 2021

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

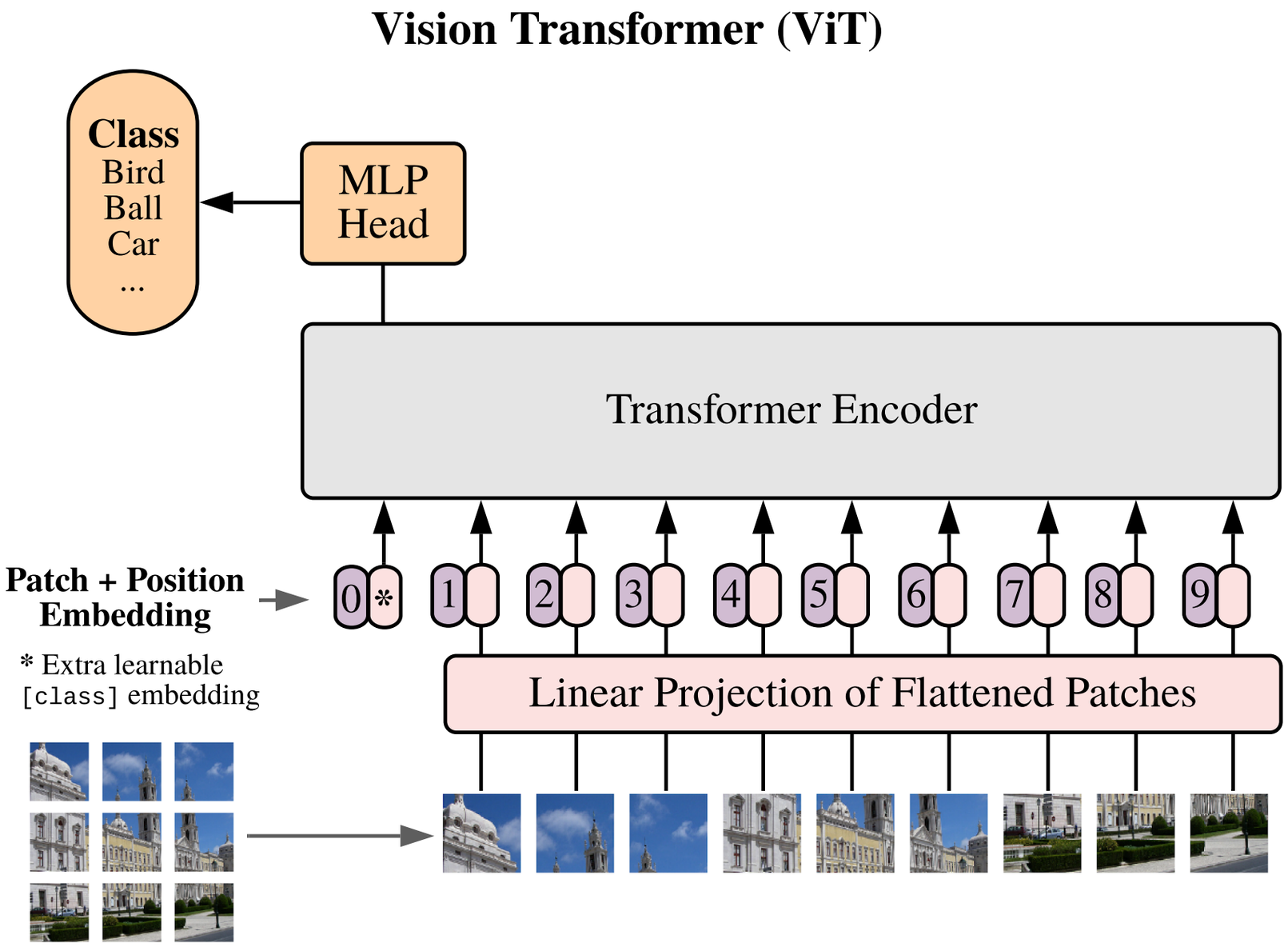
Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

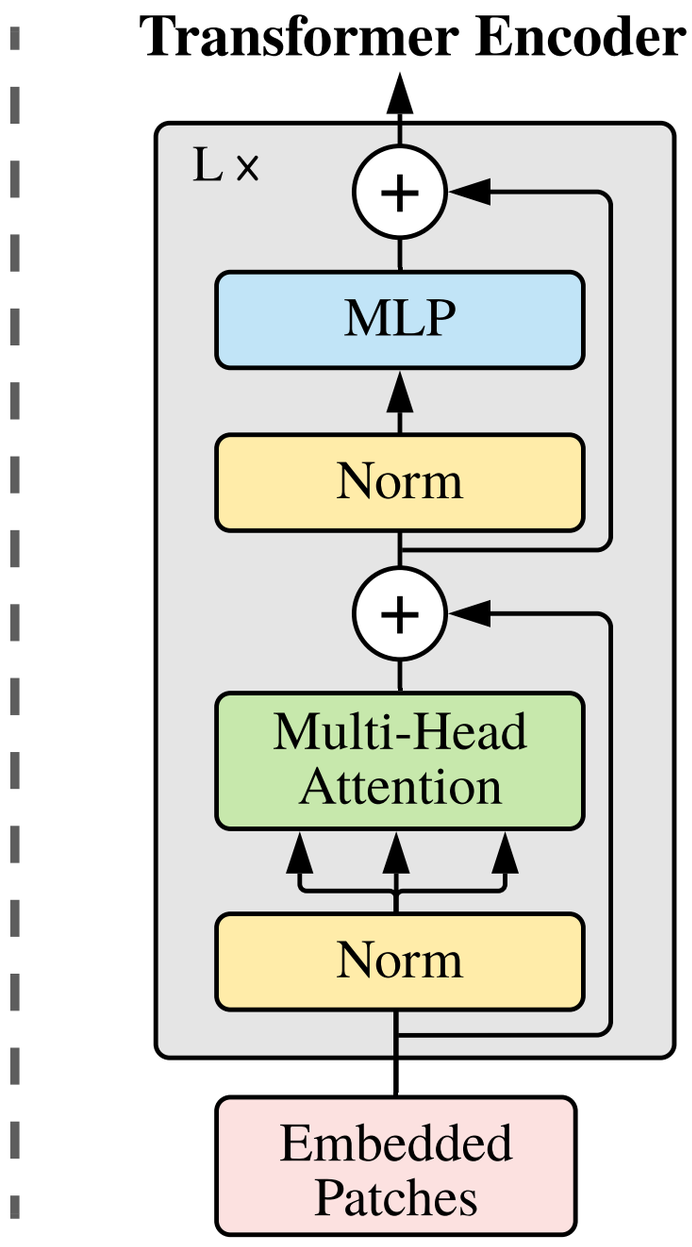
[*]equal technical contribution, [†]equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

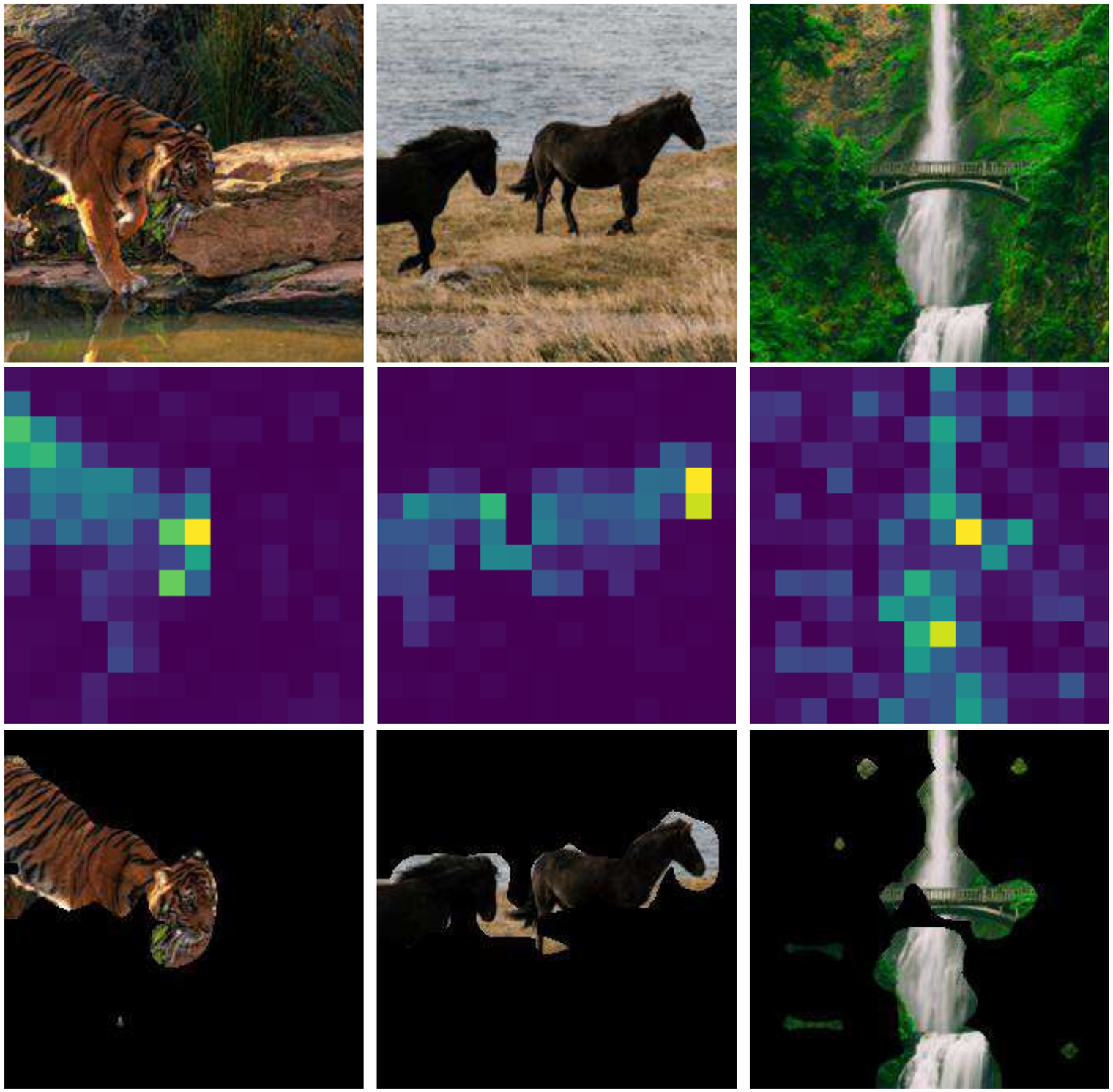
While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent

Vision Transformer (ViT)

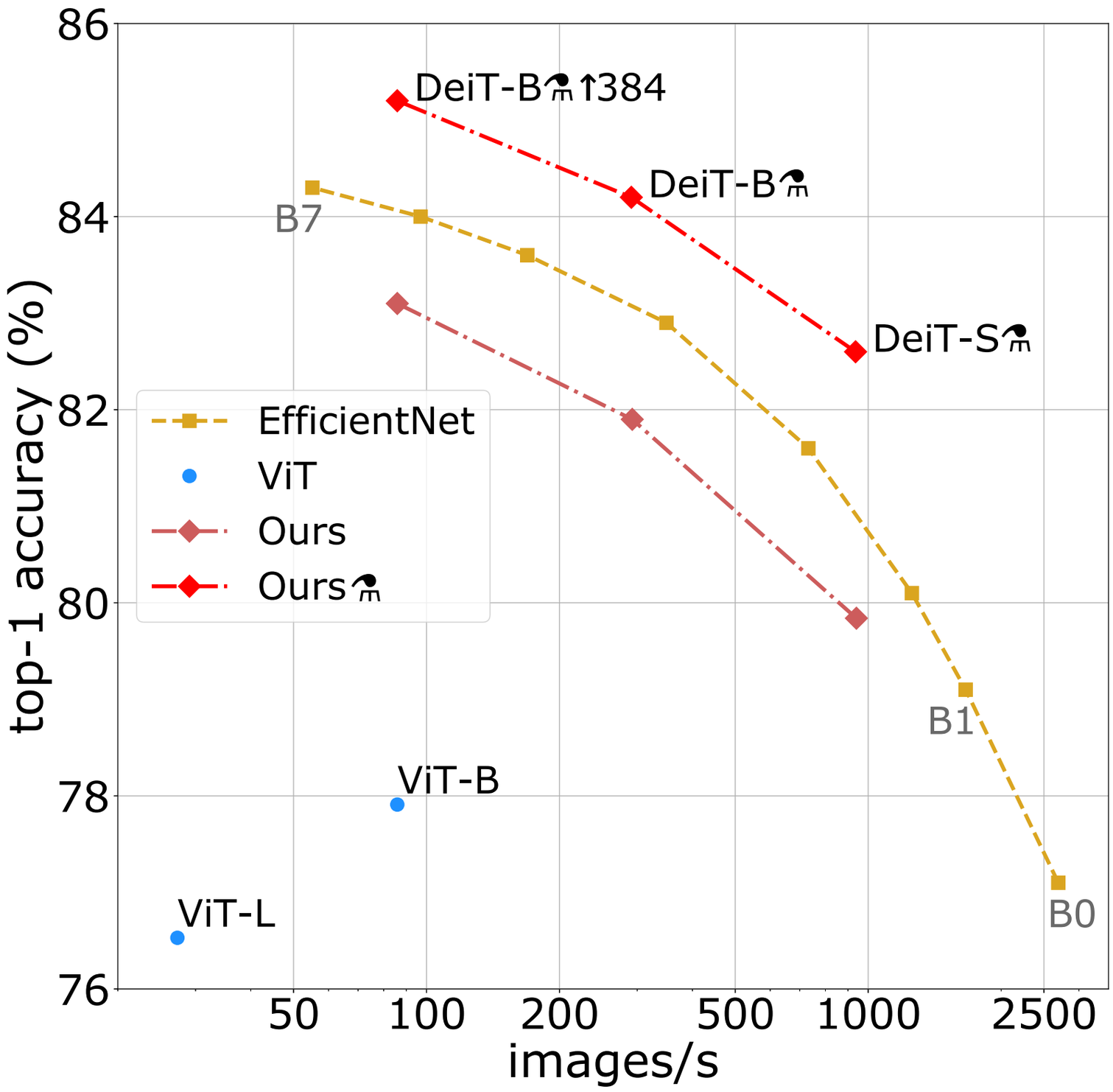




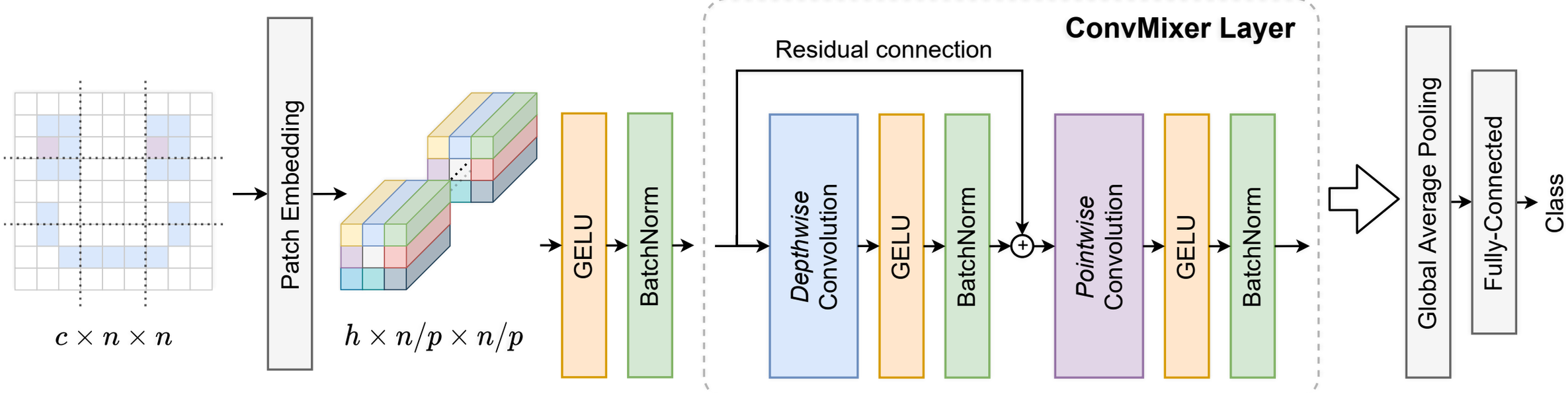
Visualizing Attention



Data-efficient image Transformer (DeiT)

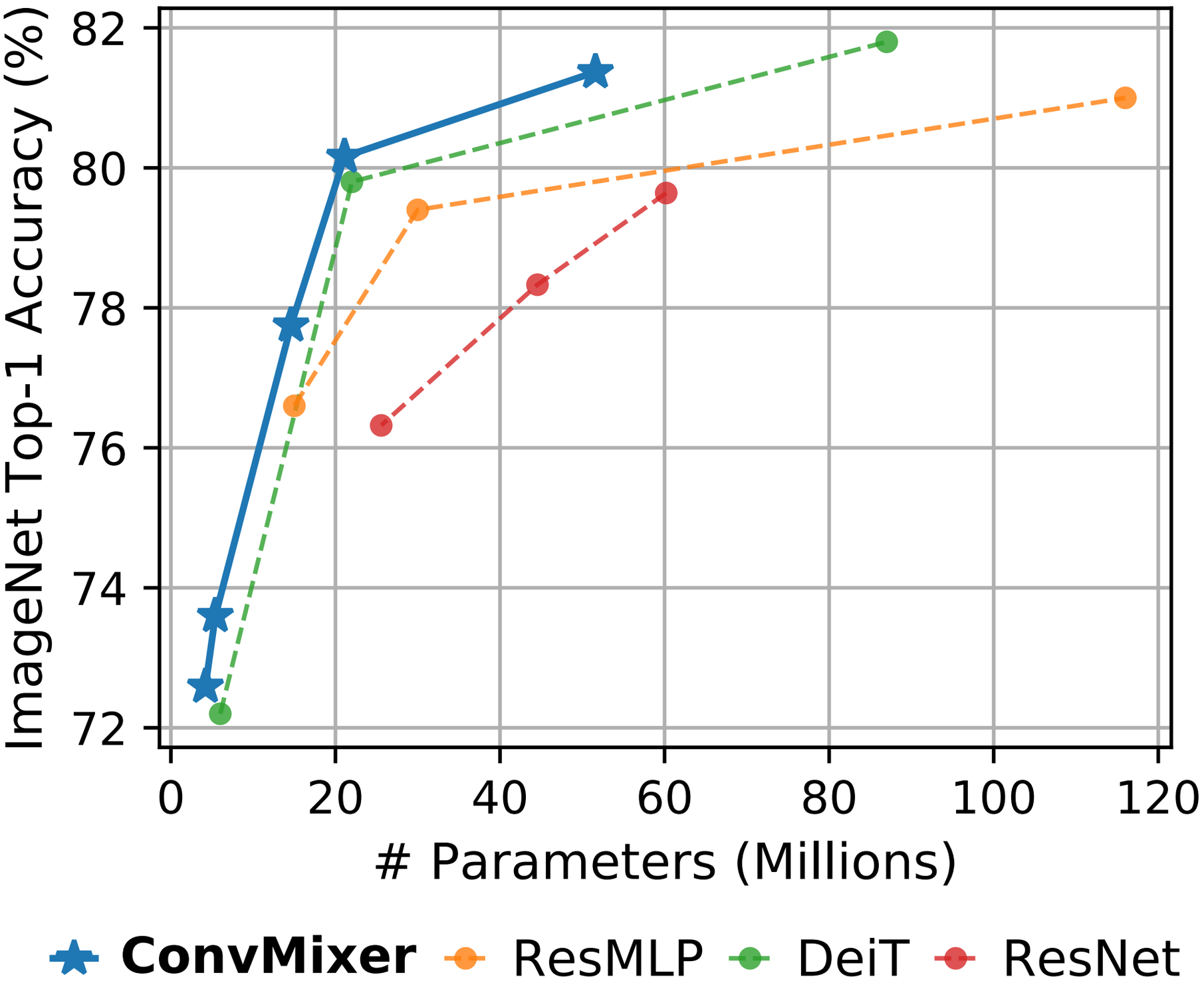


ConvMixer - Patches are all you need



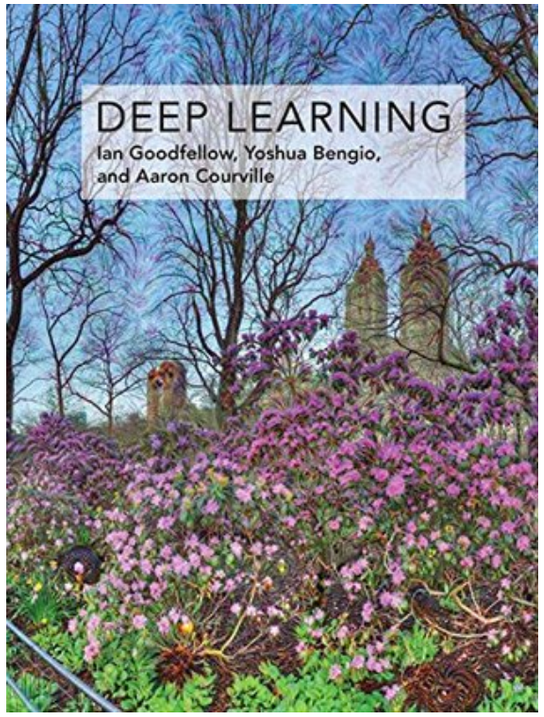

```python
def ConvMixer(h, depth, kernel_size=9, patch_size=7, n_classes=1000):
    Seq, ActBn = nn.Sequential, lambda x: Seq(x, nn.GELU(), nn.BatchNorm2d(h))
    Residual = type('Residual', (Seq,), {'forward': lambda self, x: self[0](x) + x})
    return Seq(ActBn(nn.Conv2d(3, h, patch_size, stride=patch_size)),
        *[Seq(Residual(ActBn(nn.Conv2d(h, h, kernel_size, groups=h, padding="same"))),
          ActBn(nn.Conv2d(h, h, 1))) for i in range(depth)],
         nn.AdaptiveAvgPool2d((1,1)), nn.Flatten(), nn.Linear(h, n_classes))
```

ConvMixer - Patches are all you need



A. Trockman, J. Z. Kolter. "Patches Are All You Need?", ICML 2022.

More to read…



- Goodfellow et al.
  - Chapter 9.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS 2012.
- K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." ICLR 2015.

arXiv:1409.1556v6 [cs.CV] 10 Apr 2015

Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman***
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

1 INTRODUCTION

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image and video recognition (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; Sermanet et al., 2014; Simonyan & Zisserman, 2014) which has become possible due to the large public image repositories, such as ImageNet (Deng et al., 2009), and high-performance computing systems, such as GPUs or large-scale distributed clusters (Dean et al., 2012). In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014), which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings (Perronnin et al., 2010) (the winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al., 2012) (the winner of ILSVRC-2012).

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of Krizhevsky et al. (2012) in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014). In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3×3) convolution filters in all layers.

As a result, we come up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when used as a part of a relatively simple pipelines (e.g. deep features classified by a linear SVM without fine-tuning). We have released our two best-performing models[1] to facilitate further research.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and the

*current affiliation: Google DeepMind *current affiliation: University of Oxford and Google DeepMind
[1] http://www.robots.ox.ac.uk/~vgg/research/very_deep/

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto krizh@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.