# 11-695: AI Engineering
# Convolution II

LTI/SCS

Spring 2020

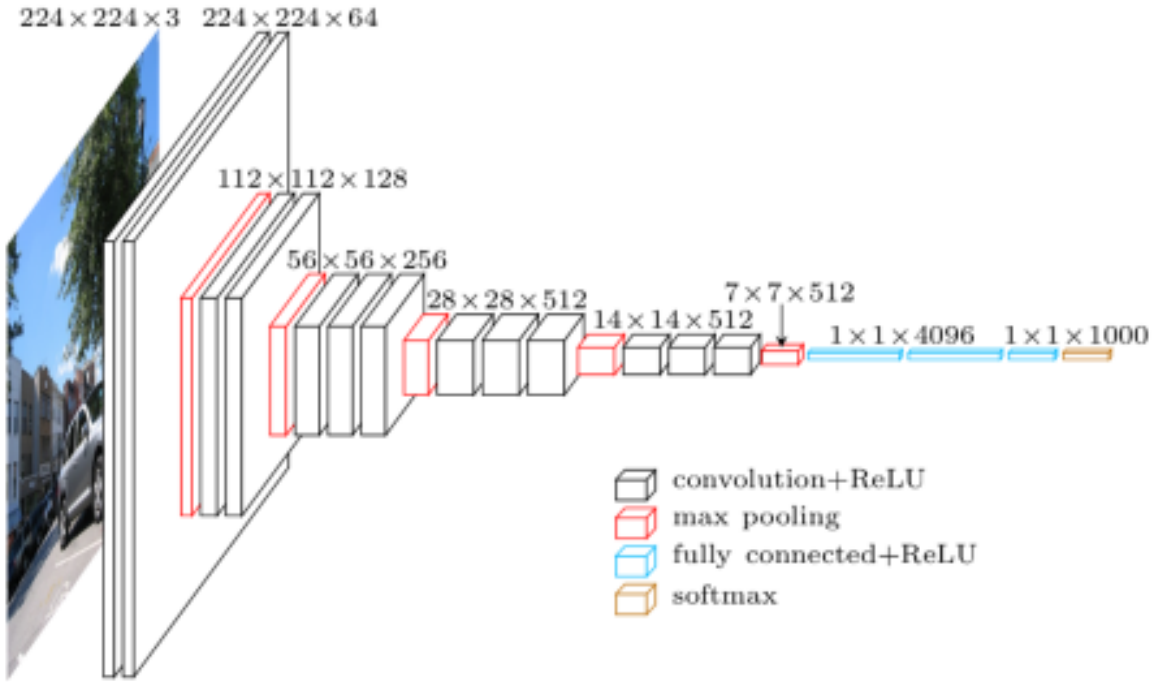
# Outline



1. Recap
2. Convolution Architectures (cont'd)
3. What CNNs Learn?
4. Graph CNN

# Where are we at so far






- Motivation of Convolution
- Operation of Convolution
- Convolutional Layers
- Pooling Layers
- LeNet-5 and AlexNet and VGG

VGG Image credit: David Frossard

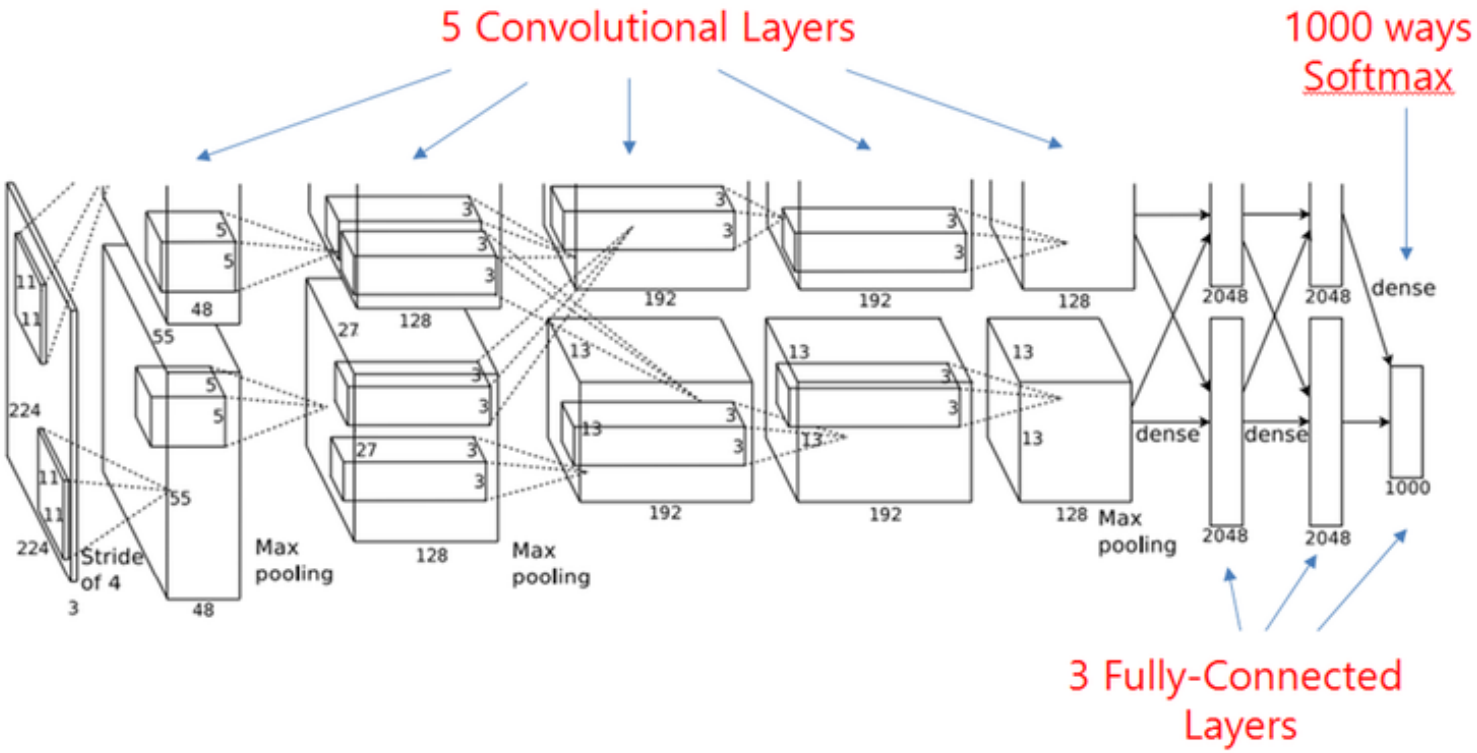
# Outline



1. Recap
2. Convolution Architectures (cont'd)
3. What CNNs Learn?
4. Graph CNN

# Recap: AlexNet



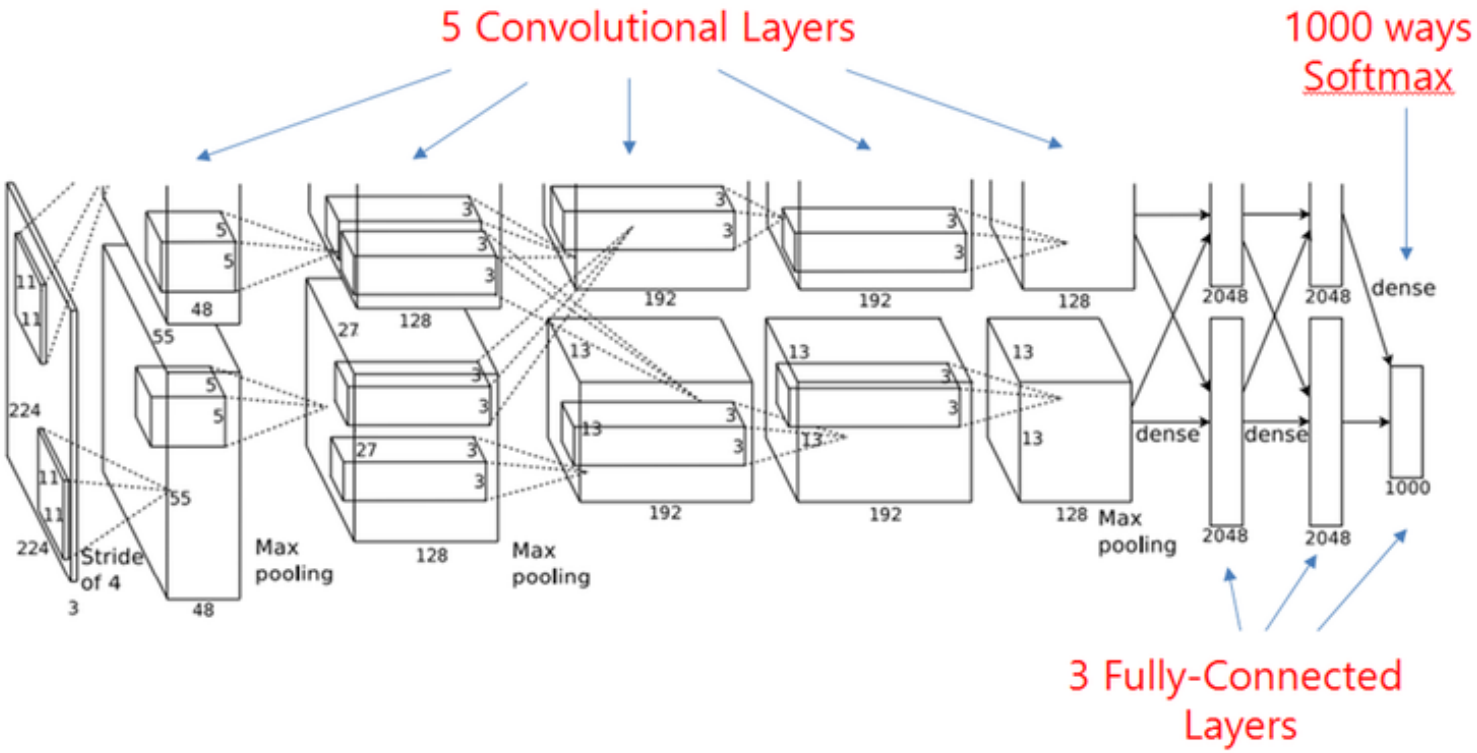



- Total params: 62,367,776 (check the module I slides)

Image credit: Krizhevsky et. al 2012

# Recap: AlexNet



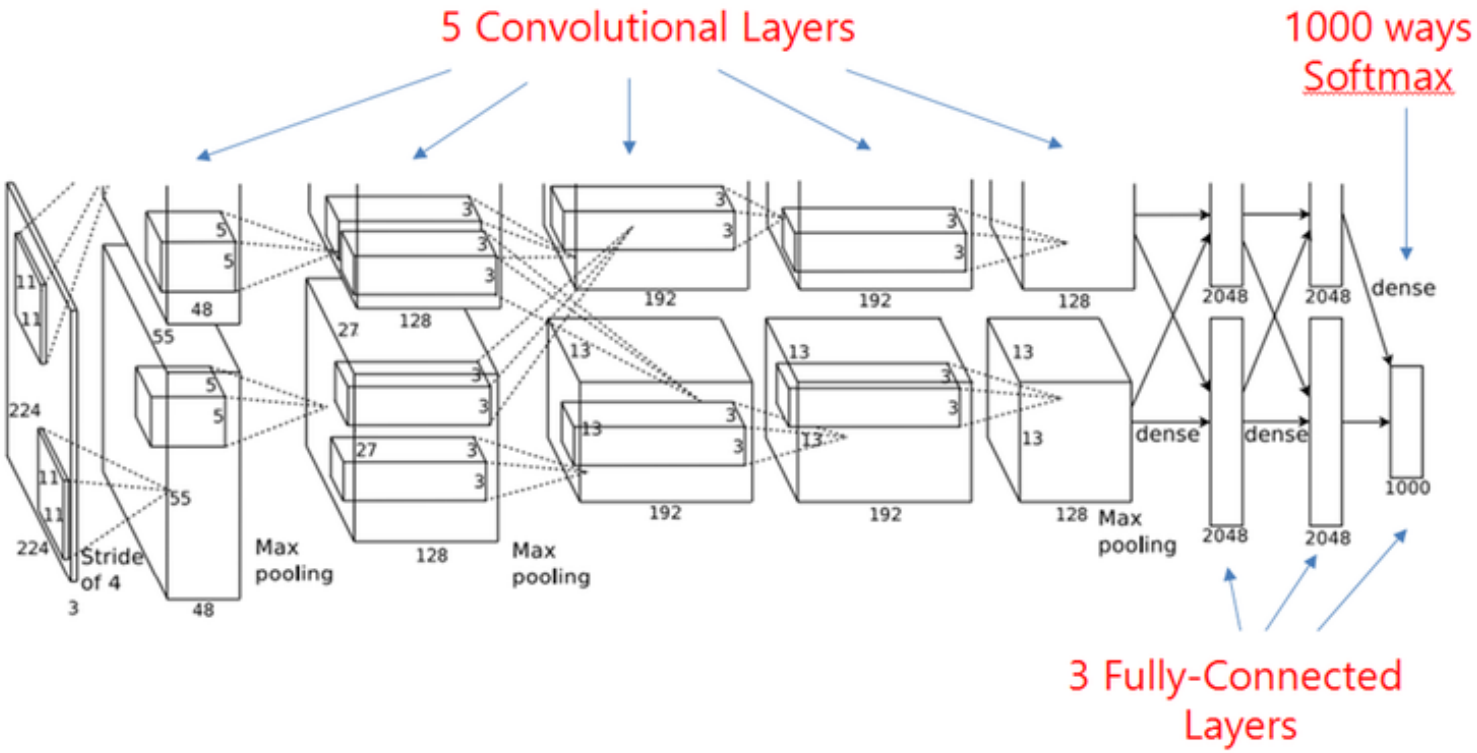



- Total params: 62,367,776 (check the module I slides)
- 58,621,952 params in FC layers ($\approx 94\%$) $\rightarrow$ a pain!

Image credit: Krizhevsky et. al 2012

# Recap: AlexNet



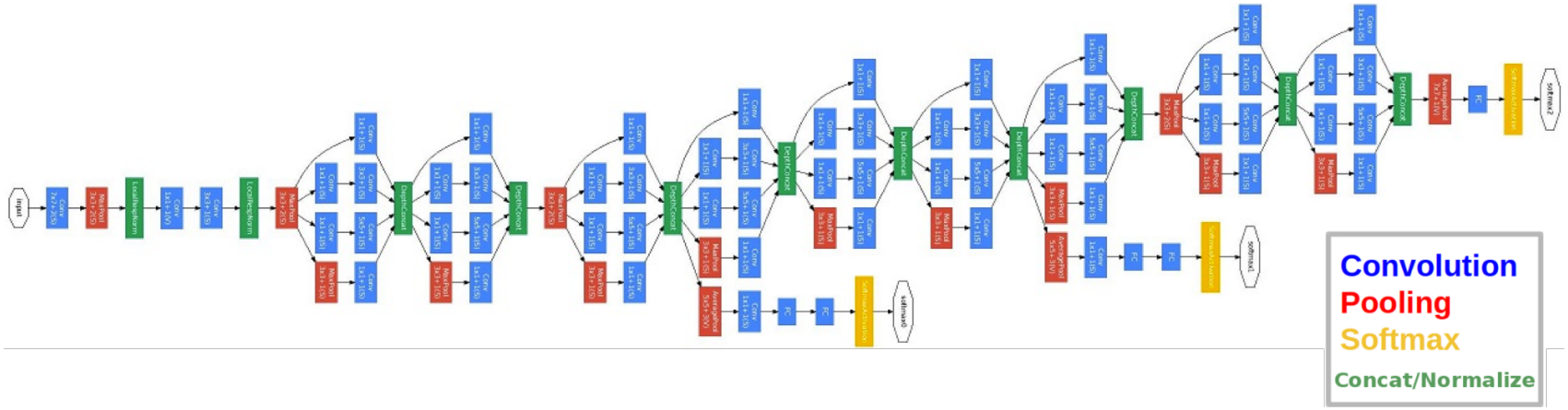



- Total params: 62,367,776 (check the module I slides)
- 58,621,952 params in FC layers ($\approx 94\%$) $\rightarrow$ a pain!
- First use: ReLU, norm layer, heavy augmentation, dropout, momentum

Image credit: Krizhevsky et. al 2012

# GoogLeNet[1]






- Inception module: *concatenate* $1 * 1$, $3 * 3$, $5 * 5$ conv and a pooling
- Auxiliary losses with discounted factor of 0.3
- Use of average pooling and conv $1 * 1$ to reduce parameters $\rightarrow$ reduce params and hence overfitting at FC layers

[1] https://arxiv.org/pdf/1409.4842.pdf

Image credit: Google

# Deep Trend?

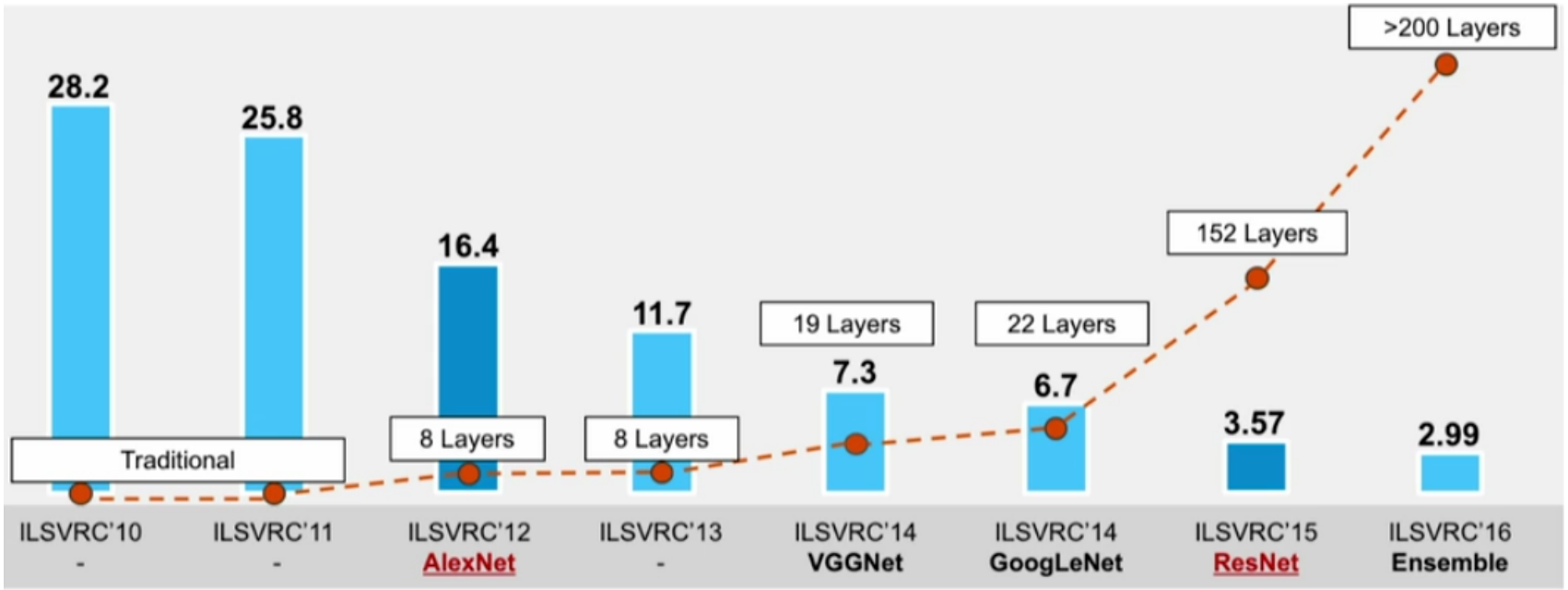
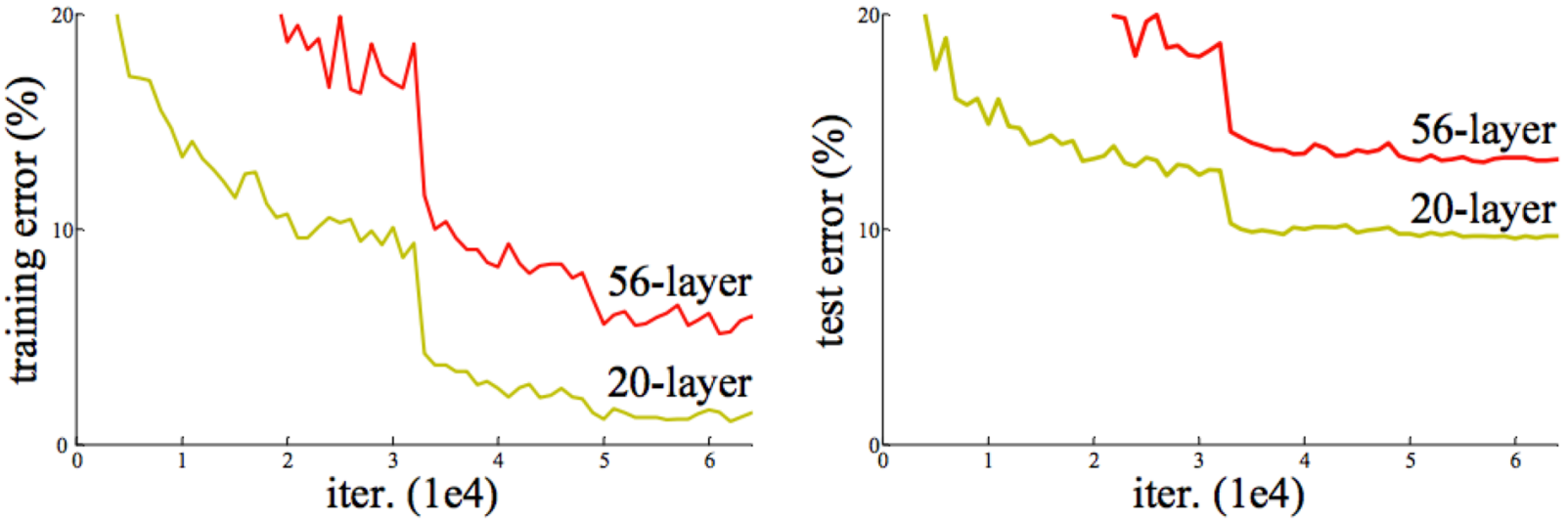



Image credit: Google

# Is more layers always better?



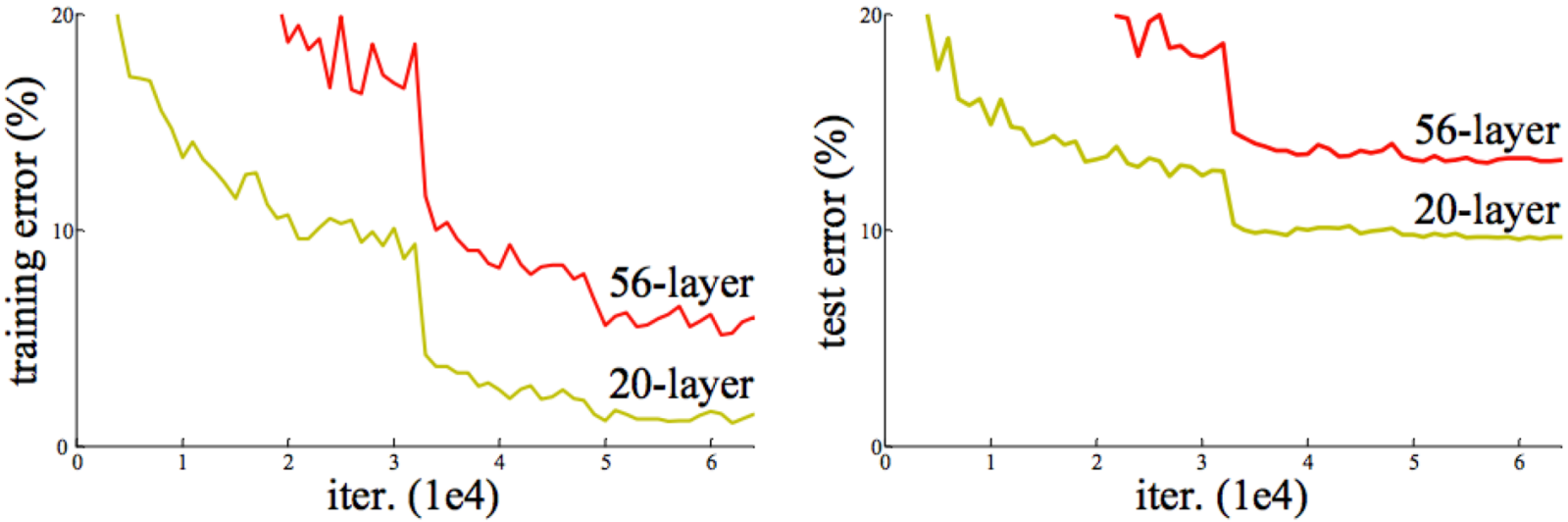



- LeNet 5 layers → AlexNet 8 layers → VGG 16 layers → GoogLeNet 22 layers
- Until it seems to saturate... (why?).

Image credit: Kaiming He

# Is more layers always better?

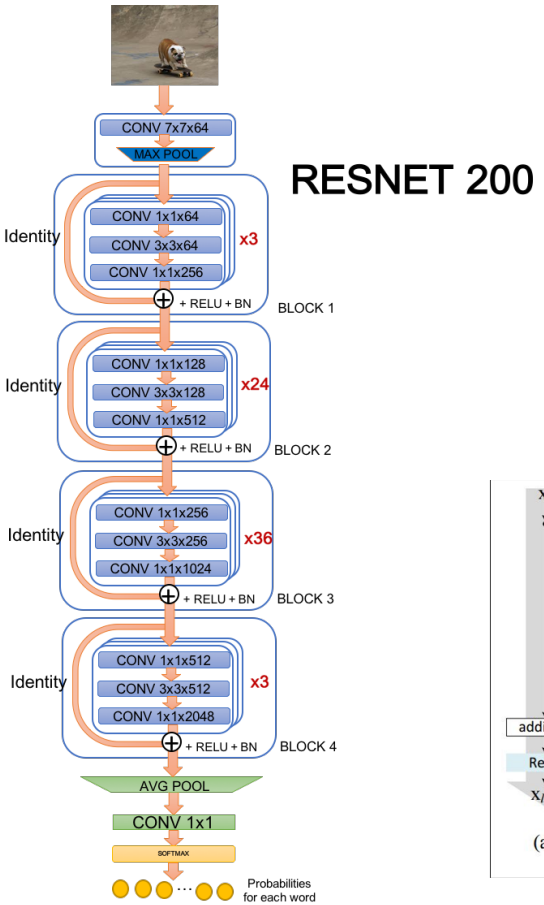


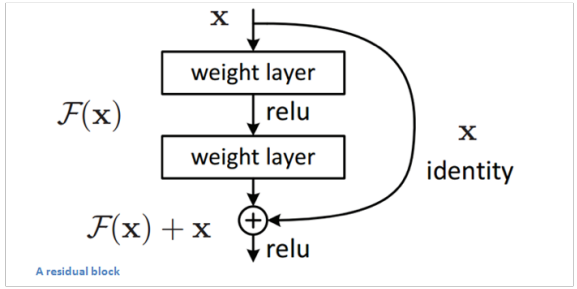



- LeNet 5 layers → AlexNet 8 layers → VGG 16 layers → GoogLeNet 22 layers
- Until it seems to saturate... (why?). We need something more than just increasing no. of layers (and resources)

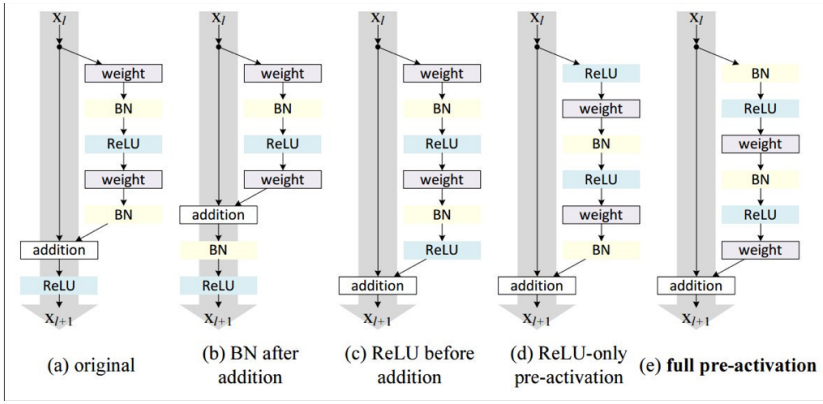
Image credit: Kaiming He

# ResNet[2] (2015)




TYPES of RESNET BLOCKS



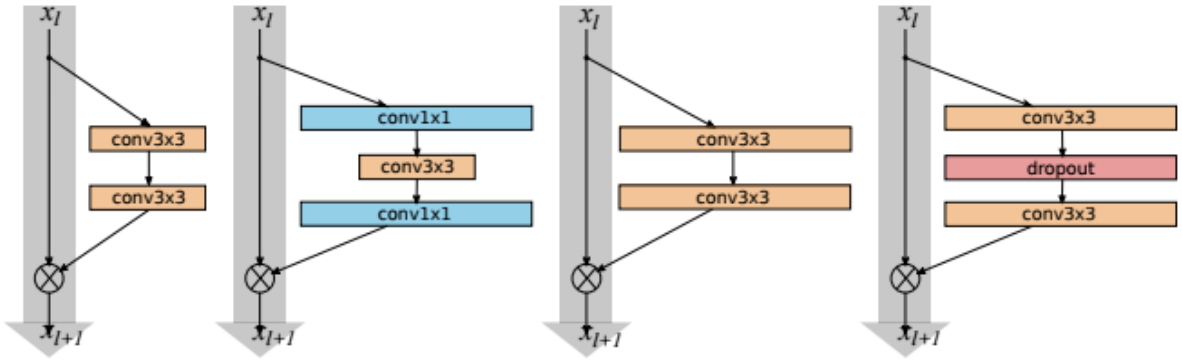

- Use all $3 * 3$ convs, only 2 poolings (MAX and AVG)
- Identity mapping of either Padding or Trainable type

[2] https://arxiv.org/pdf/1512.03385.pdf and https://arxiv.org/pdf/1603.05027.pdf

Image credit: Kaiming He

# WideResNet[3] (2016)

Carnegie Mellon



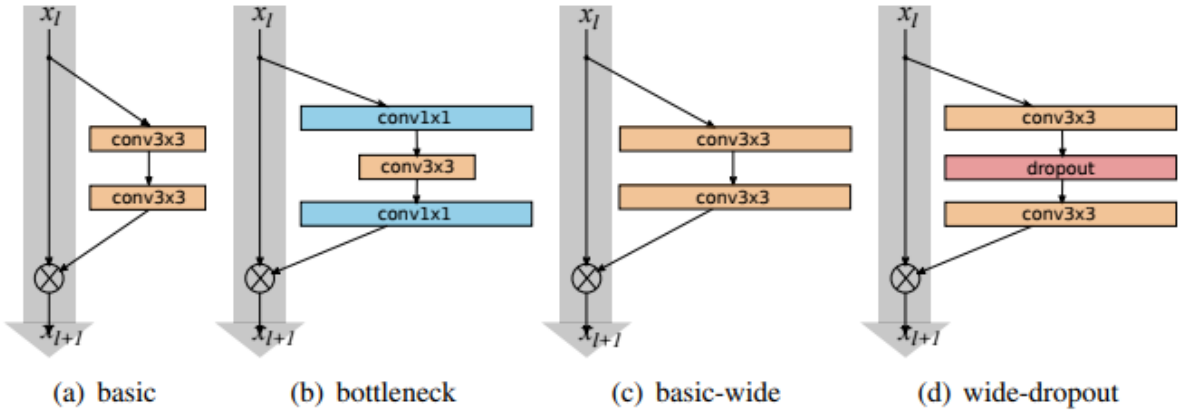

- Going wide instead of deeper, to?

[3] https://arxiv.org/pdf/1605.07146.pdf

Image credit: Sergey Zagoruyko and Nikos Komodakis

# WideResNet[3] (2016)






(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

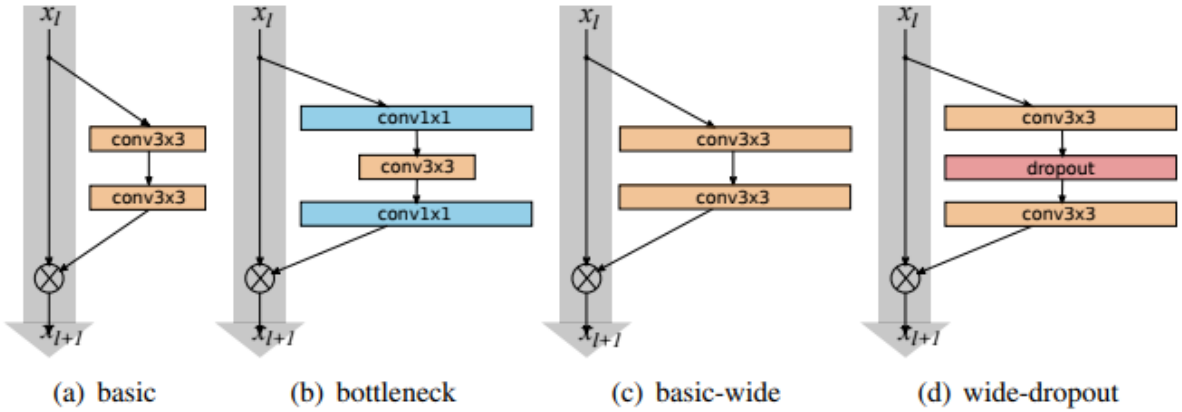
- Going wide instead of deeper, to?
  - exploit parallelization advantages of GPU,

[3] https://arxiv.org/pdf/1605.07146.pdf

Image credit: Sergey Zagoruyko and Nikos Komodakis

# WideResNet[3] (2016)






(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

- Going wide instead of deeper, to?
  - exploit parallelization advantages of GPU,
  - increase feature reuse.

[3] https://arxiv.org/pdf/1605.07146.pdf

Image credit: Sergey Zagoruyko and Nikos Komodakis

# WideResNet (2016)

Carnegie Mellon

| group name | output size | block type = $B(3,3)$ |
|---|---|---|
| conv1 | $32 \times 32$ | [3×3, 16] |
| conv2 | 32×32 | $\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$ |
| conv3 | 16×16 | $\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$ |
| conv4 | 8×8 | $\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$ |
| avg-pool | $1 \times 1$ | $[8 \times 8]$ |

Table 1: Structure of wide residual networks. Network width is determined by factor $k$. Original architecture [13] is equivalent to $k = 1$. Groups of convolutions are shown in brackets where $N$ is a number of blocks in group, downsampling performed by the first layers in groups `conv3` and `conv4`. Final classification layer is omitted for clearance. In the particular example shown, the network uses a ResNet block of type $B(3,3)$.

- Going wide instead of deeper, to?
  - exploit parallelization advantages of GPU,
  - increase feature reuse.
- Same as ResNet, two important uses of:

Image credit: Sergey Zagoruyko and Nikos Komodakis

# WideResNet (2016)

Carnegie Mellon

| group name | output size | block type = $B(3,3)$ |
|---|---|---|
| conv1 | $32 \times 32$ | [3×3, 16] |
| conv2 | 32×32 | $\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$ |
| conv3 | 16×16 | $\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$ |
| conv4 | 8×8 | $\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$ |
| avg-pool | $1 \times 1$ | $[8 \times 8]$ |

Table 1: Structure of wide residual networks. Network width is determined by factor $k$. Original architecture [13] is equivalent to $k = 1$. Groups of convolutions are shown in brackets where $N$ is a number of blocks in group, downsampling performed by the first layers in groups `conv3` and `conv4`. Final classification layer is omitted for clearance. In the particular example shown, the network uses a ResNet block of type $B(3,3)$.

- Going wide instead of deeper, to?
  - exploit parallelization advantages of GPU,
  - increase feature reuse.
- Same as ResNet, two important uses of:
  - Residual connection to alleviate gradient vanishing and create ensembling effect

Image credit: Sergey Zagoruyko and Nikos Komodakis

# WideResNet (2016)

Carnegie Mellon

| group name | output size | block type = $B(3,3)$ |
|---|---|---|
| conv1 | $32 \times 32$ | [3×3, 16] |
| conv2 | 32×32 | $\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$ |
| conv3 | 16×16 | $\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$ |
| conv4 | 8×8 | $\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$ |
| avg-pool | $1 \times 1$ | $[8 \times 8]$ |

Table 1: Structure of wide residual networks. Network width is determined by factor $k$. Original architecture [13] is equivalent to $k = 1$. Groups of convolutions are shown in brackets where $N$ is a number of blocks in group, downsampling performed by the first layers in groups `conv3` and `conv4`. Final classification layer is omitted for clearance. In the particular example shown, the network uses a ResNet block of type $B(3,3)$.

- Going wide instead of deeper, to?
  - exploit parallelization advantages of GPU,
  - increase feature reuse.
- Same as ResNet, two important uses of:
  - Residual connection to alleviate gradient vanishing and create ensembling effect
  - Use of $1 * 1$ conv layers to reduce dimensionality

Image credit: Sergey Zagoruyko and Nikos Komodakis

# ResNeXt[4] (2016)



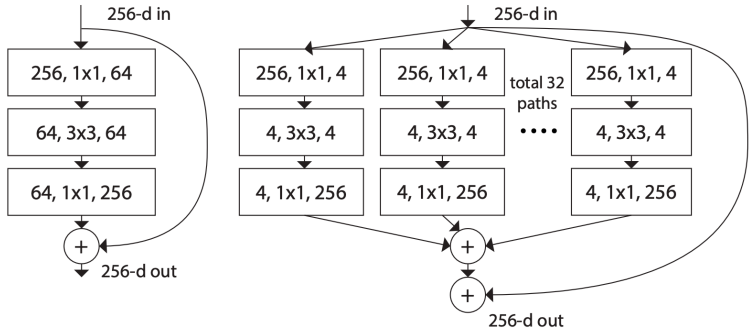



Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

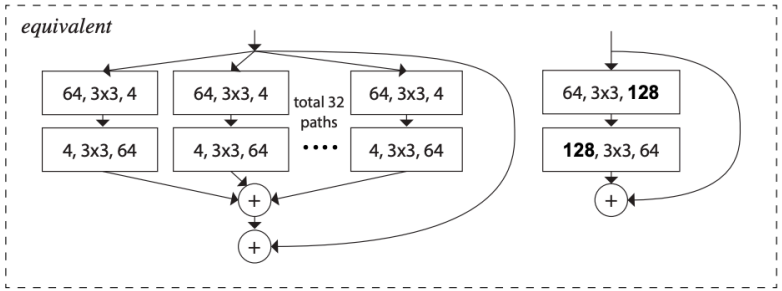



Figure 4. (**Left**): Aggregating transformations of depth = 2. (**Right**): An equivalent block, which is trivially wider.

| stage | output | ResNet-50 | **ResNeXt-50 (32×4d)** |
|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | 7×7, 64, stride 2 |
| conv2 | 56×56 | 3×3 max pool, stride 2 | 3×3 max pool, stride 2 |
| | | [1×1, 64<br>3×3, 64<br>1×1, 256] ×3 | [1×1, 128<br>3×3, 128, $C$=32<br>1×1, 256] ×3 |
| conv3 | 28×28 | [1×1, 128<br>3×3, 128<br>1×1, 512] ×4 | [1×1, 256<br>3×3, 256, $C$=32<br>1×1, 512] ×4 |
| conv4 | 14×14 | [1×1, 256<br>3×3, 256<br>1×1, 1024] ×6 | [1×1, 512<br>3×3, 512, $C$=32<br>1×1, 1024] ×6 |
| conv5 | 7×7 | [1×1, 512<br>3×3, 512<br>1×1, 2048] ×3 | [1×1, 1024<br>3×3, 1024, $C$=32<br>1×1, 2048] ×3 |
| | 1×1 | global average pool<br>1000-d fc, softmax | global average pool<br>1000-d fc, softmax |
| # params. | | **25.5**×10$^6$ | **25.0**×10$^6$ |
| FLOPs | | **4.1**×10$^9$ | **4.2**×10$^9$ |

Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "$C$=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

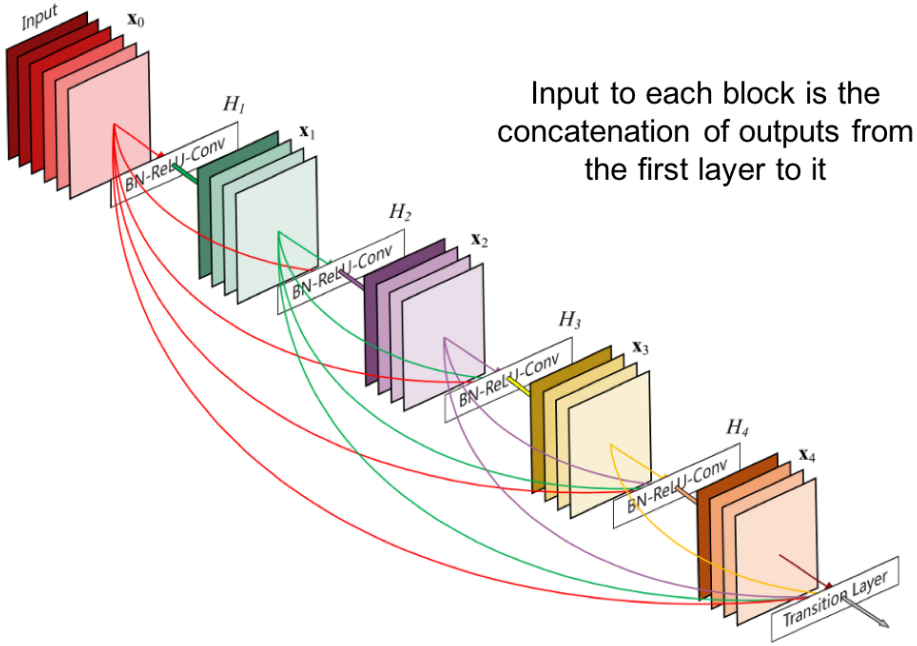
- Also going wider, but with *cardinality* (group)

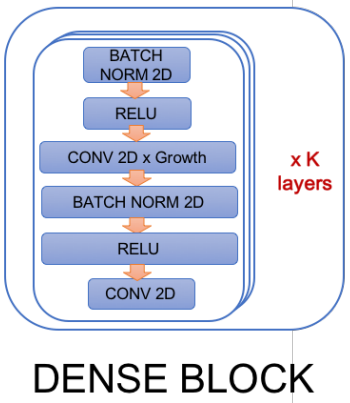
[4] https://arxiv.org/pdf/1611.05431.pdf

Image credit: Saining Xie *et al.*

# DenseNet[5] (2017)

Carnegie Mellon



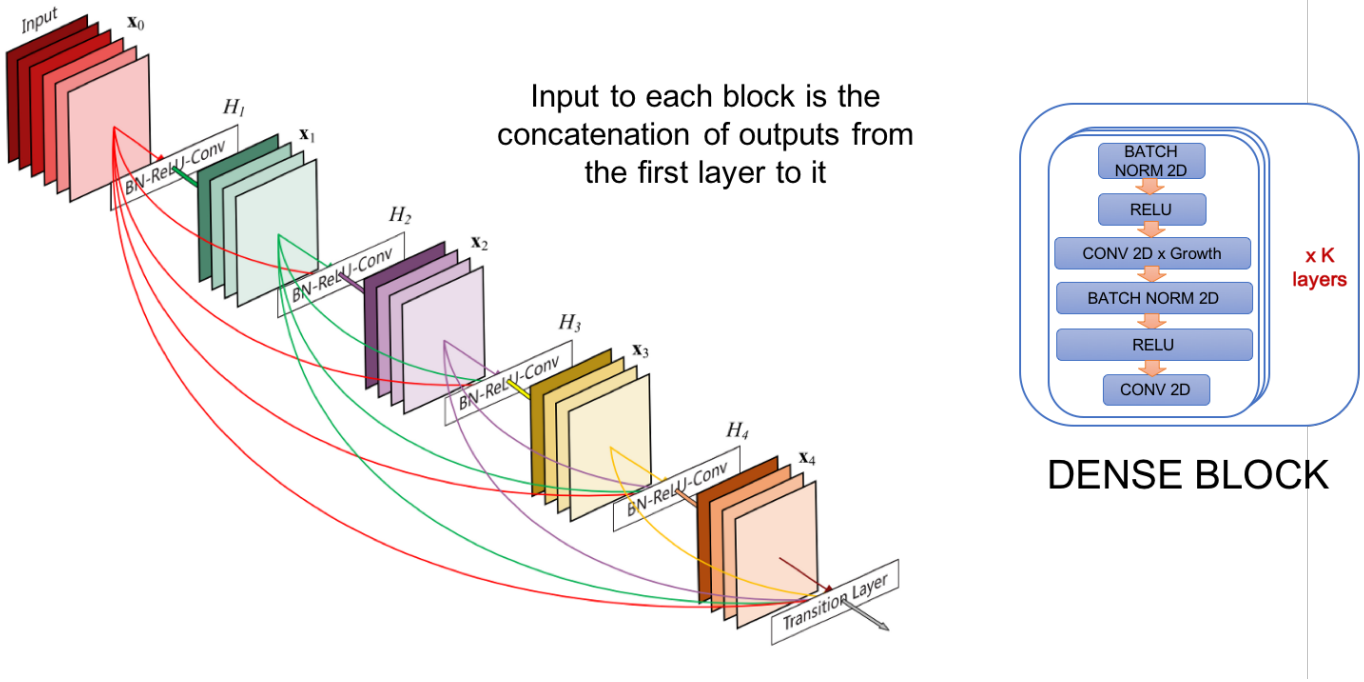

- Same motivation of feature reuse, with a different implementation
- It grows with layers

[5] https://arxiv.org/abs/1608.06993

Image credit: Gao Huang et al.

# DenseNet[5] (2017)

Carnegie Mellon



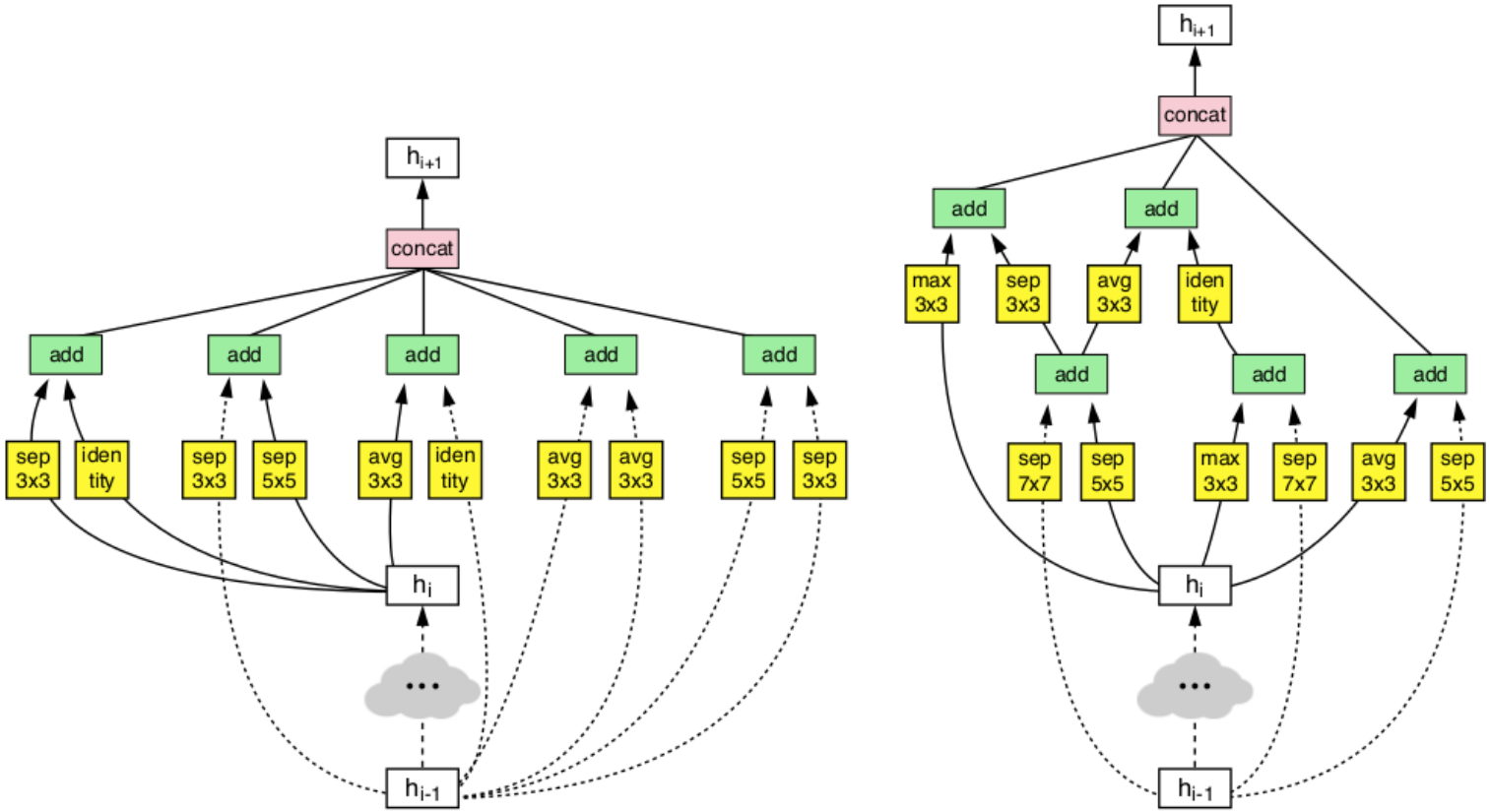

- Same motivation of feature reuse, with a different implementation
- It grows with layers
- But it also has a measure to reduce it by a compressing transition layer which outputs $\theta * m$ layers $(0 < \theta < 1)$

[5] https://arxiv.org/abs/1608.06993

Image credit: Gao Huang et al.

# NASNet[6]

Carnegie Mellon



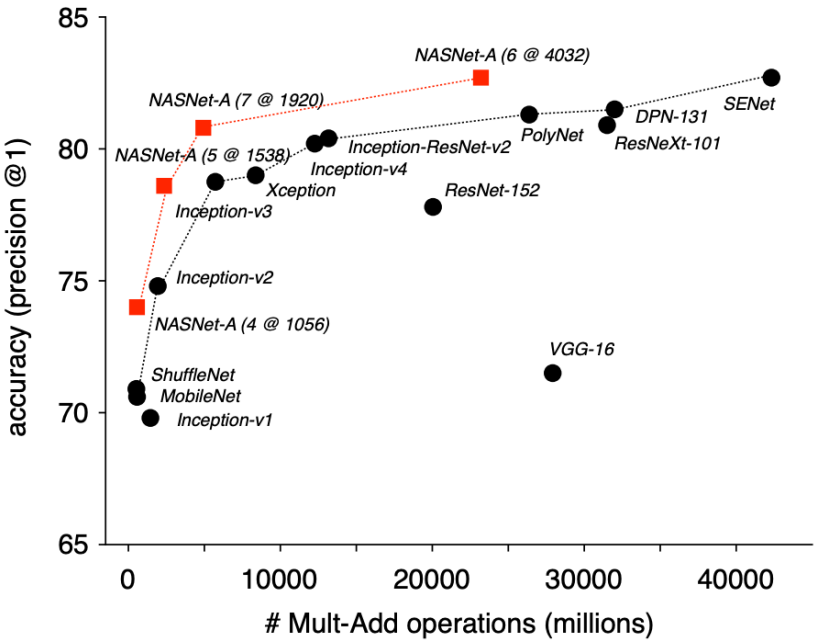

- As a result from Neural Architecture Search

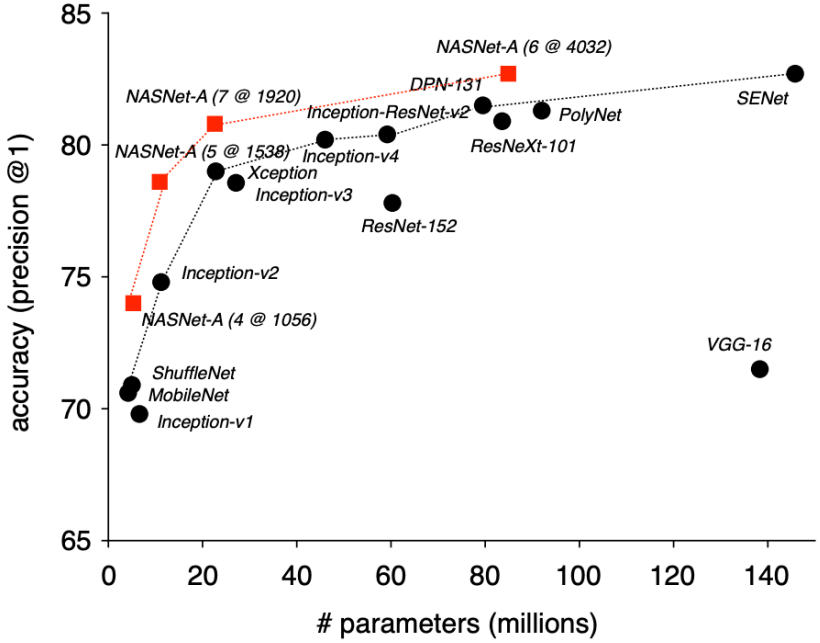
[6] https://arxiv.org/pdf/1707.07012.pdf

Image credit: Barret Zoph - Google AI

# NASNet

Carnegie Mellon




- As a result from Neural Architecture Search
- Attain state-of-the-art results in several datasets

Image credit: Barret Zoph - Google AI

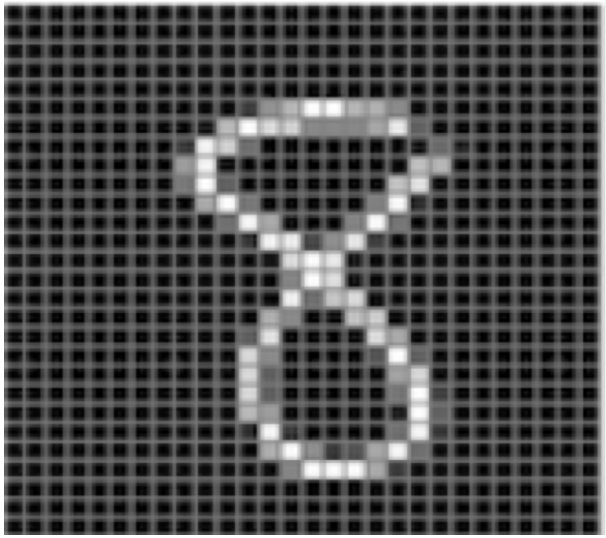
# Outline

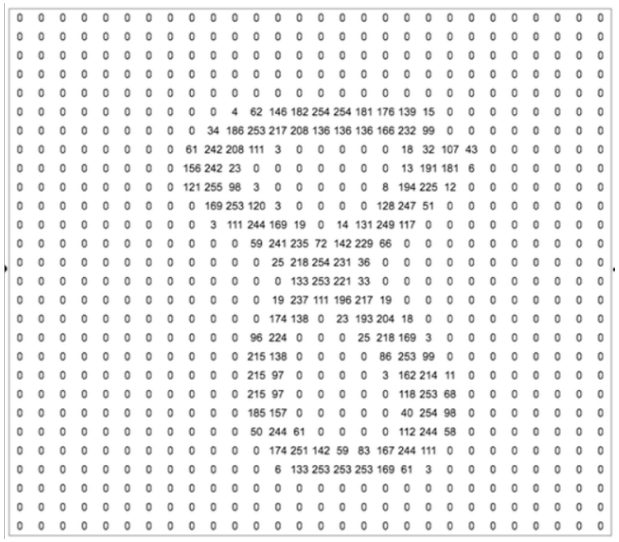


1. Recap
2. Convolution Architectures (cont'd)
3. What CNNs Learn?
4. Graph CNN

# Is CNN a blackbox to human?

Carnegie Mellon



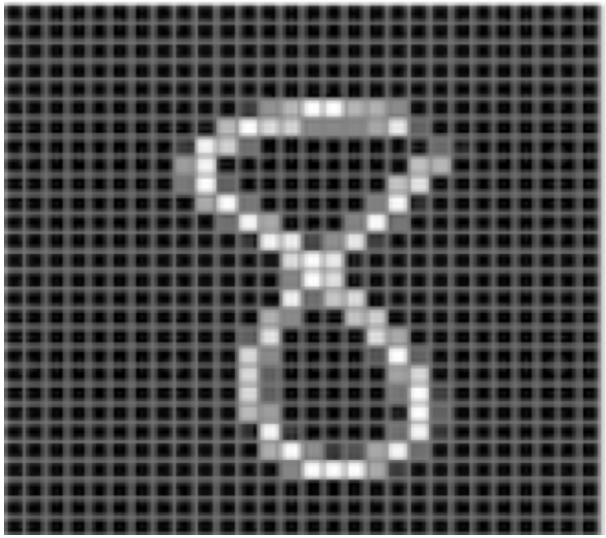


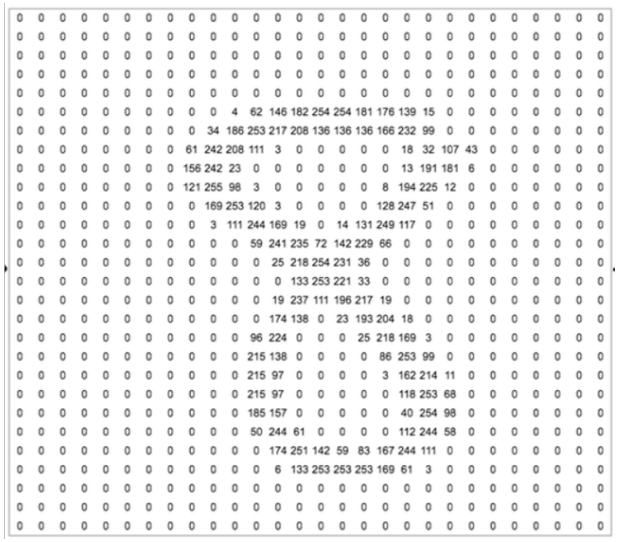
- Ugly truth: Human's perception and machine's are different
- Human struggled to make machine understand vision (the way it thinks useful)

Image credit: https://ml4a.github.io/ml4a/neural_networks/

# Is CNN a blackbox to human?

Carnegie Mellon



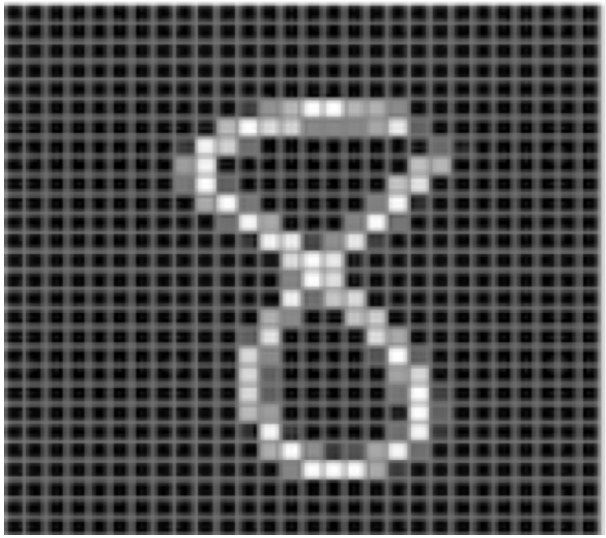


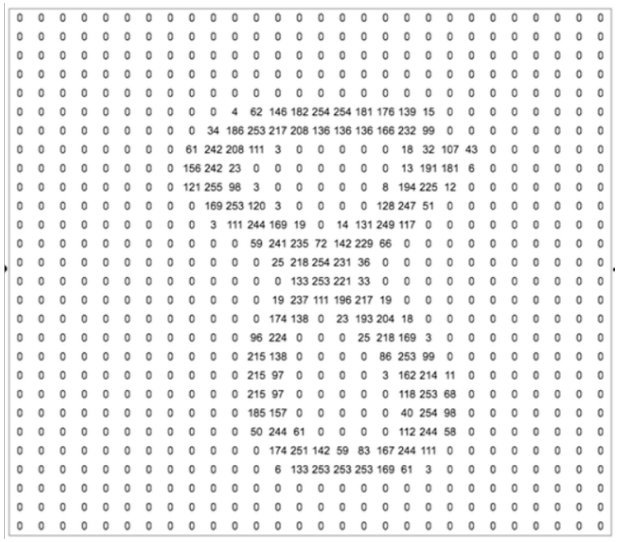

- Ugly truth: Human's perception and machine's are different
- Human struggled to make machine understand vision (the way it thinks useful)
- Now human makes effort to understand more about machine's language/perception of vision

Image credit: https://ml4a.github.io/ml4a/neural_networks/

# Is CNN a blackbox to human?

Carnegie Mellon



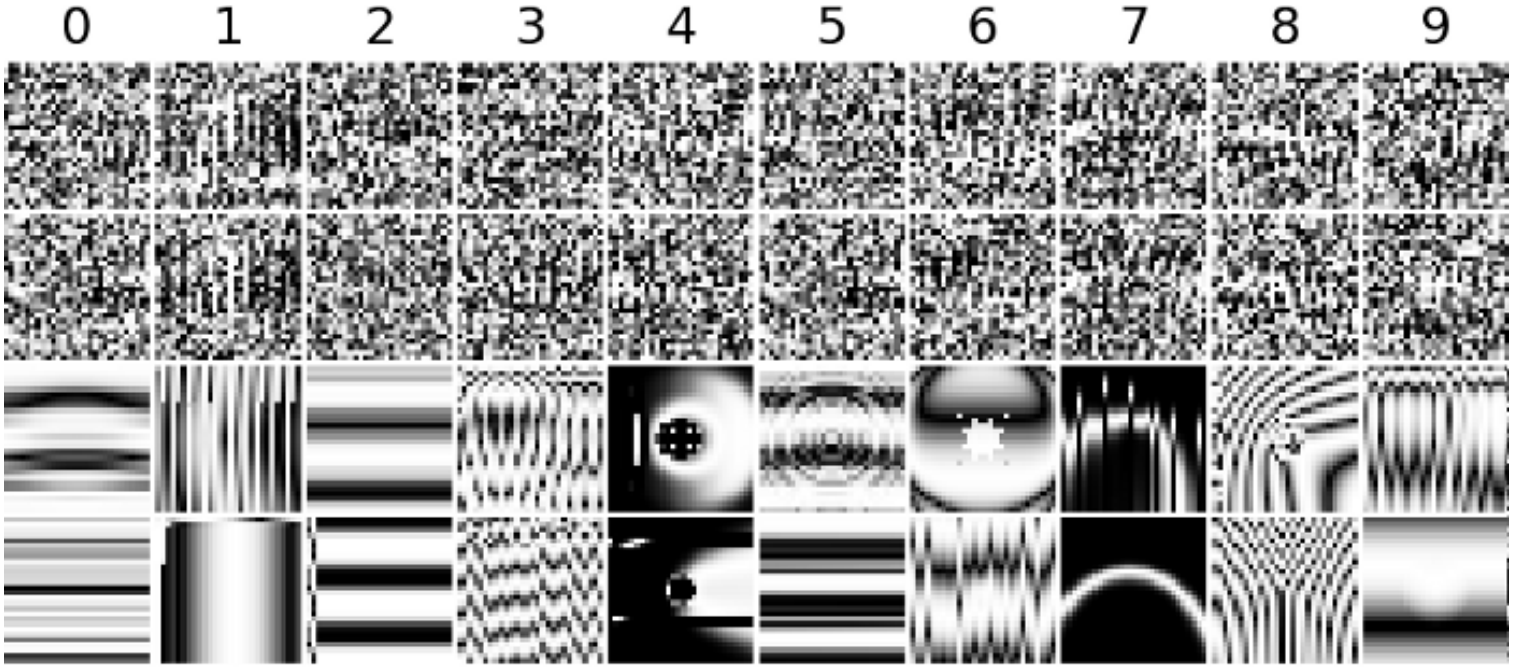



- Ugly truth: Human's perception and machine's are different
- Human struggled to make machine understand vision (the way it thinks useful)
- Now human makes effort to understand more about machine's language/perception of vision
- Although how NN works in general is still a myth...

Image credit: https://ml4a.github.io/ml4a/neural_networks/

# Human vs. Machine

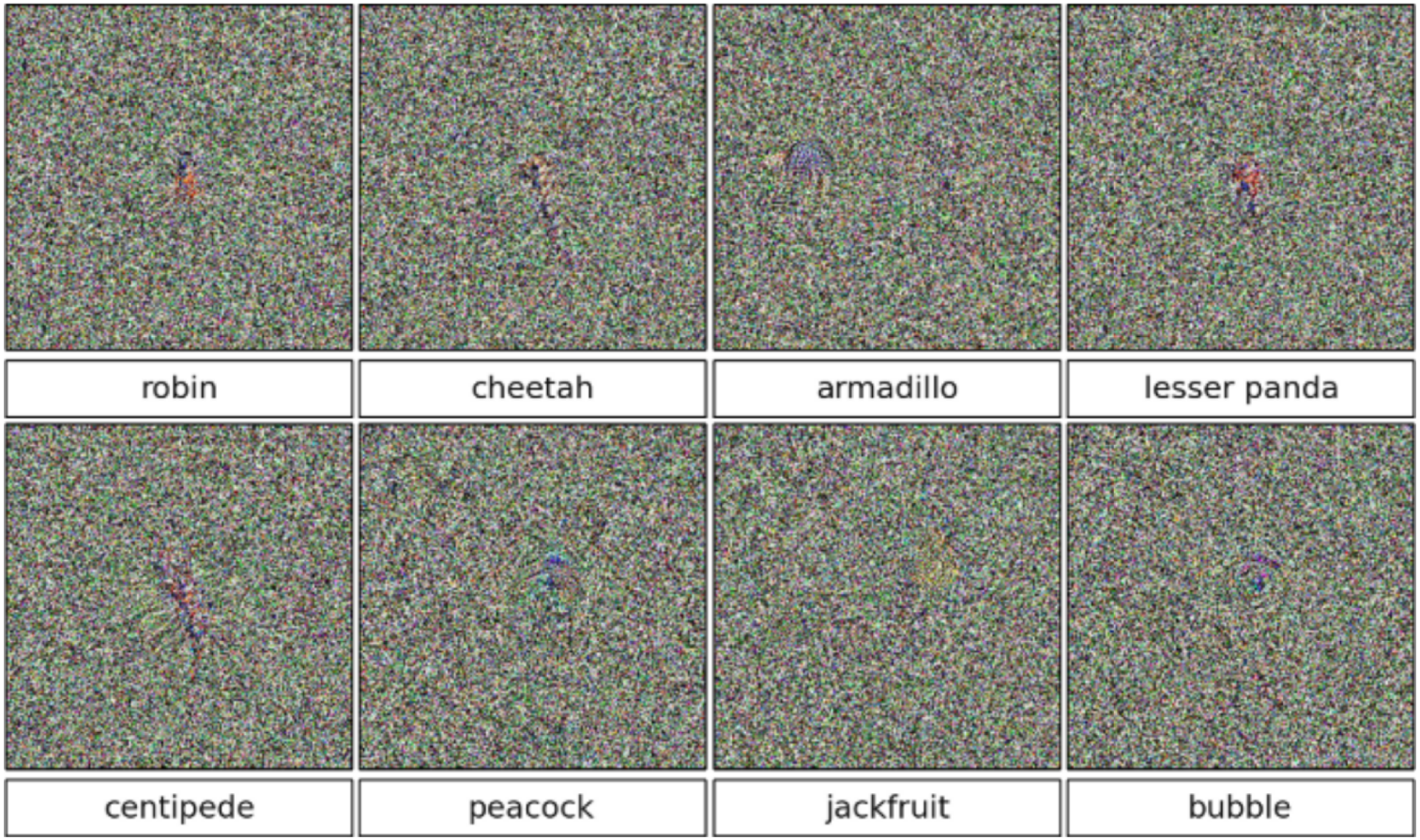





- Machine: 99.99% vs. Human: $\approx 0\%$

Image credit: Anh Nguyen et al.

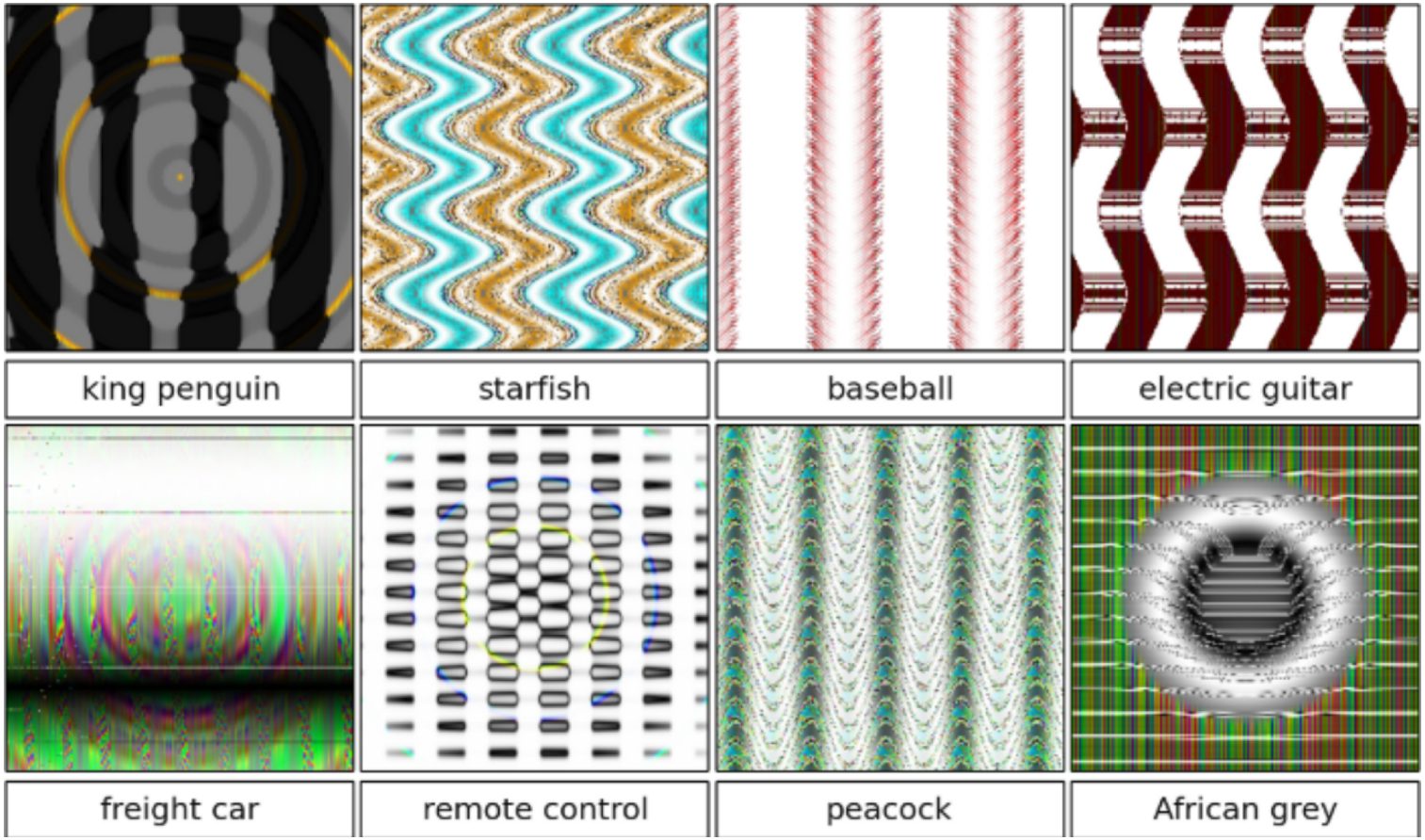
# Human vs. Machine




- Machine: 99.99% vs. Human: $\approx 0\%$

Image credit: Anh Nguyen et al.

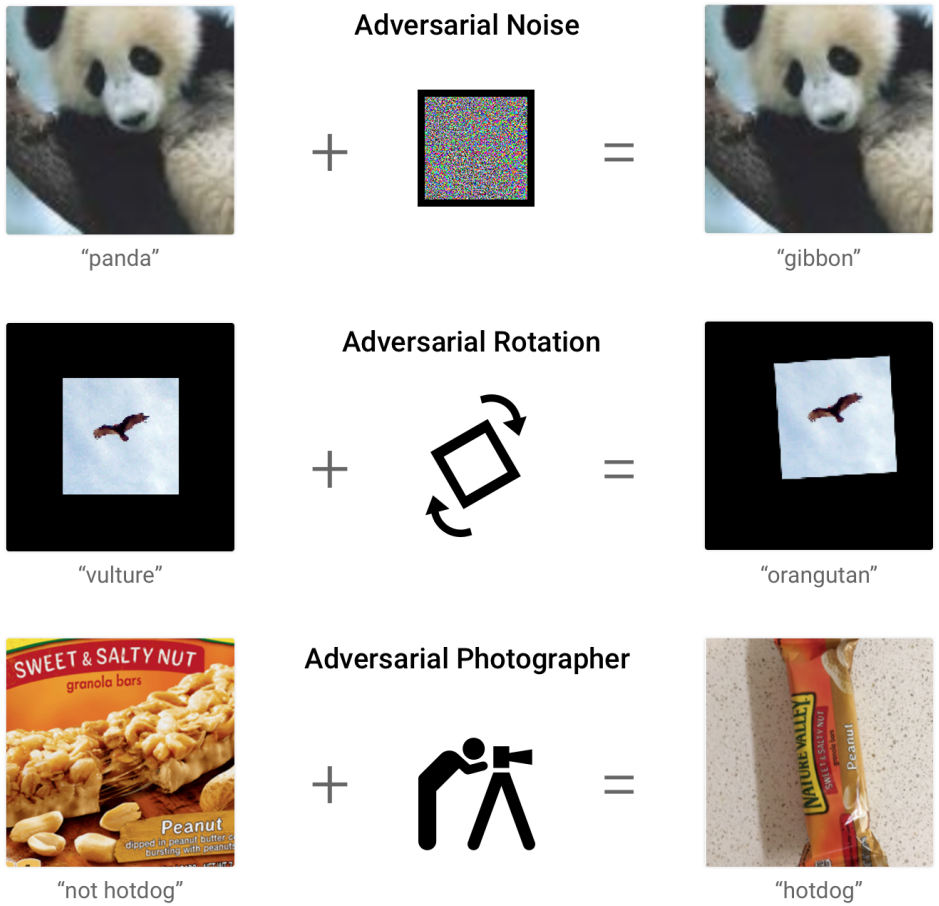
# Human vs. Machine




- Machine: 99.99% vs. Human: $\approx 0\%$

Image credit: Anh Nguyen et al.

# Human vs. Machine

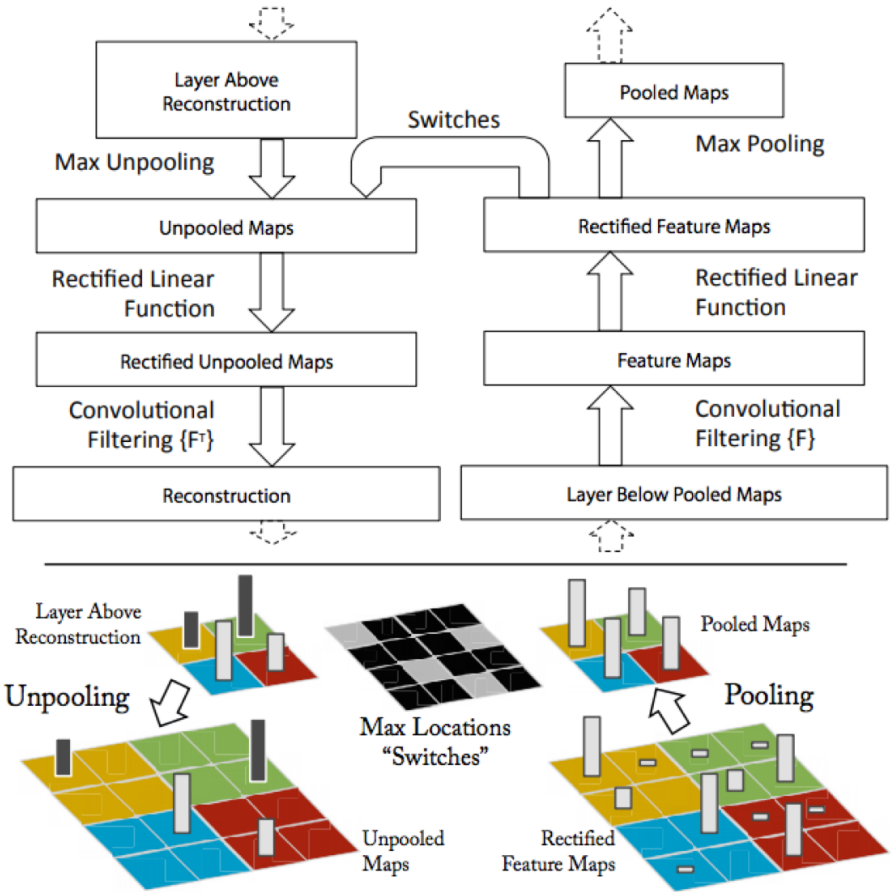



- Panda + 0.07 Noise = Gibbon

Image credit: Google AI Blog

# Translate back from machine to human?

Carnegie Mellon

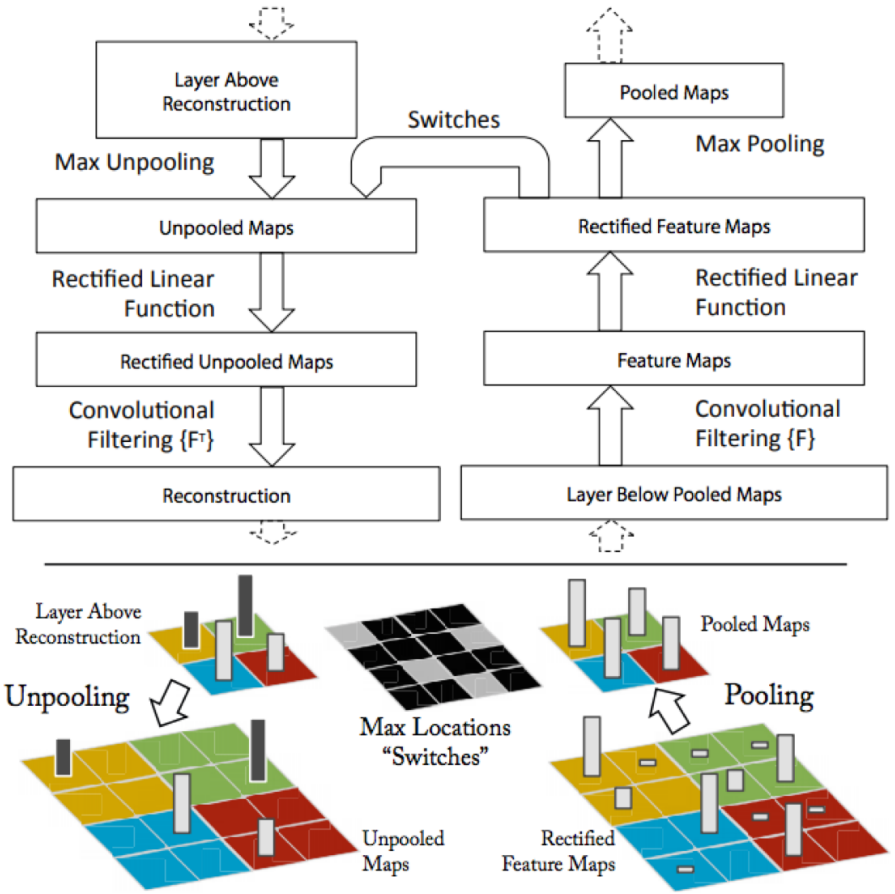



- Convolution $\iff$ Deconvolution, but

Image credit: Zeiler and Fergus

# Translate back from machine to human?






- Convolution $\iff$ Deconvolution, but
- Is CNN an invertable transformation?

Image credit: Zeiler and Fergus

# CNN Visualization



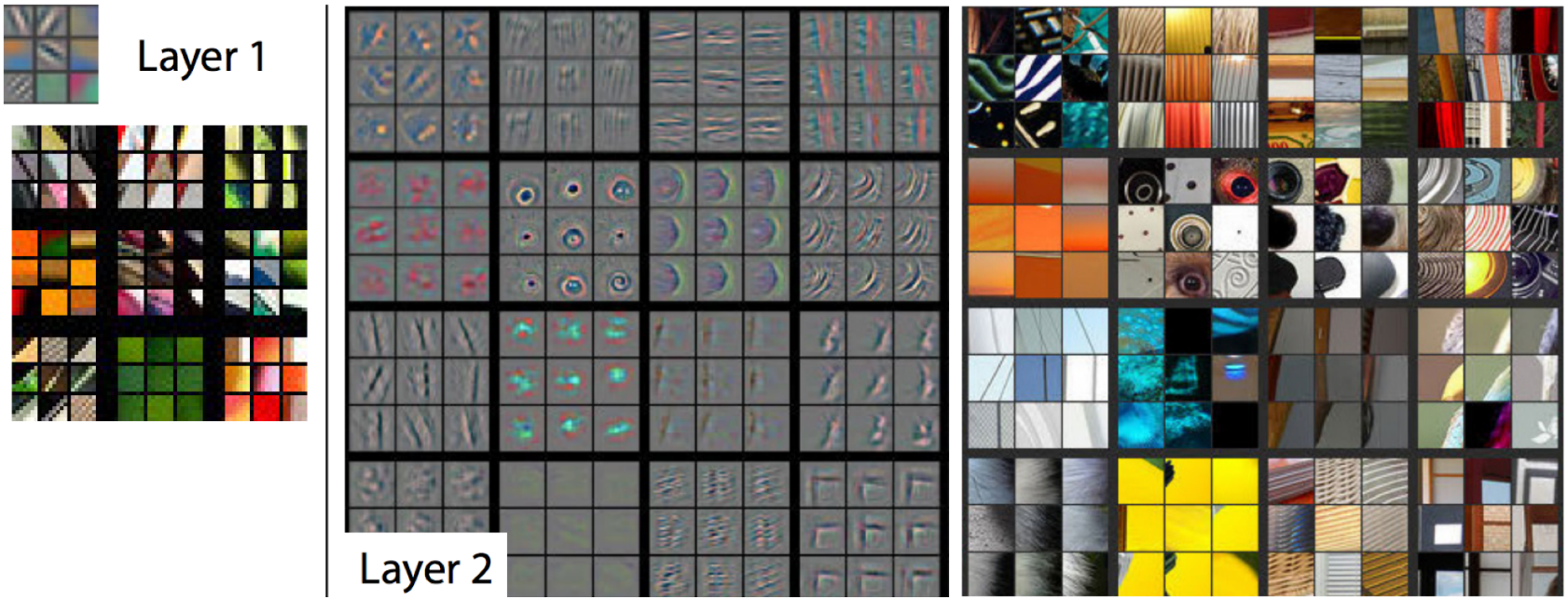



Image credit: Zeiler and Fergus

# CNN Visualization

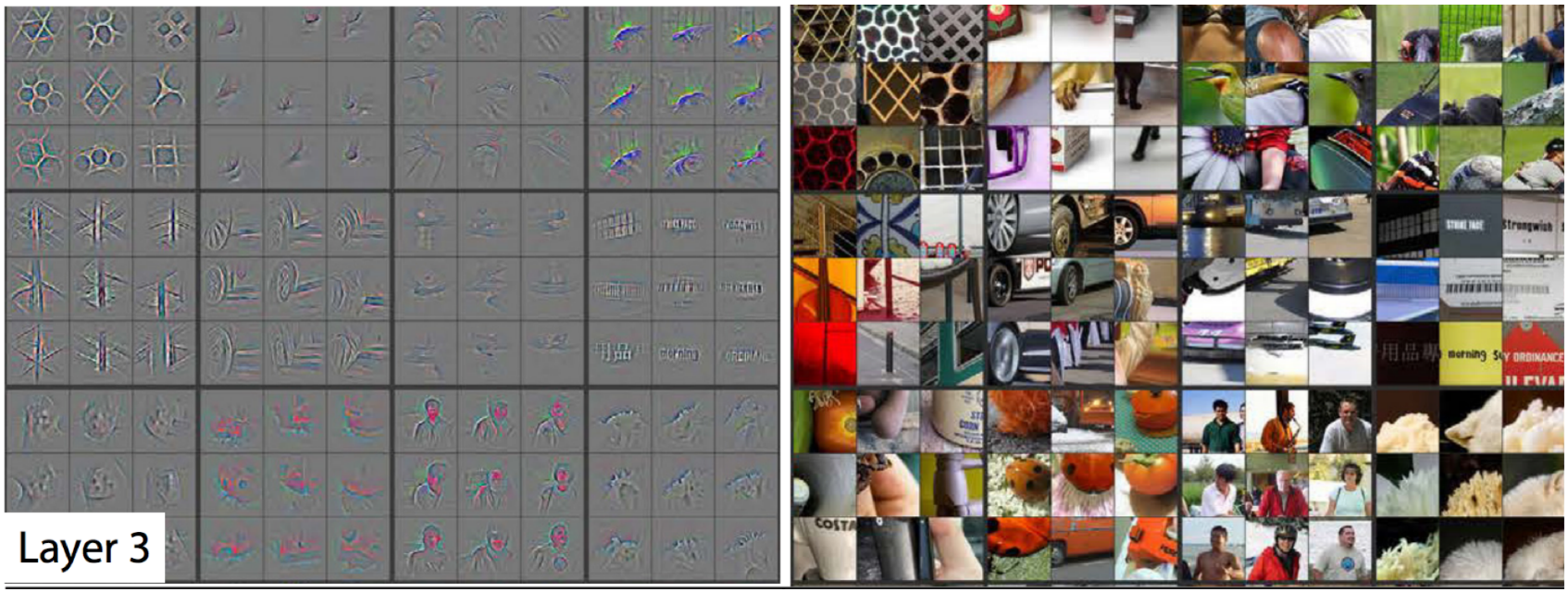



Image credit: Zeiler and Fergus

# CNN Visualization

Demo

Carnegie Mellon

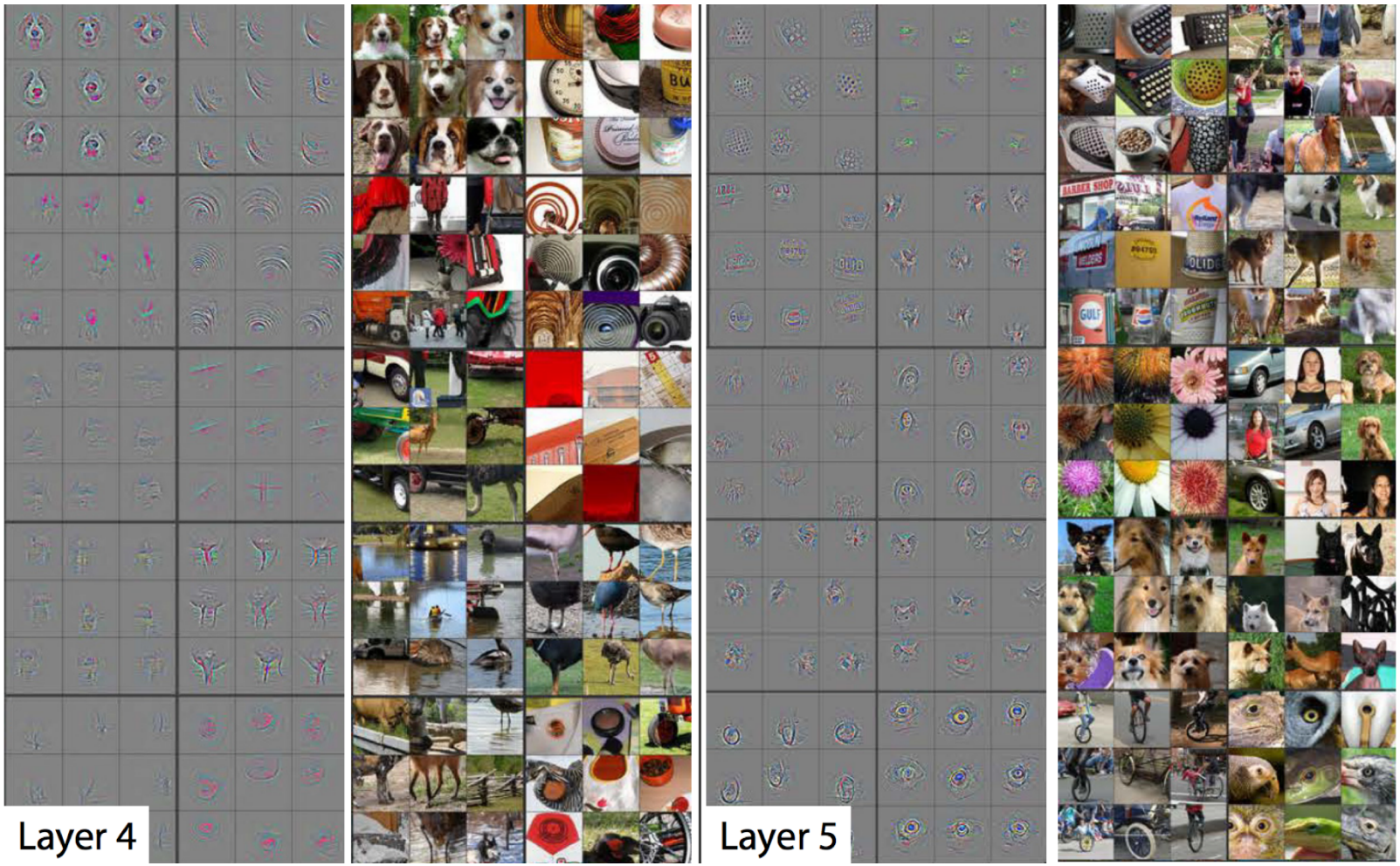
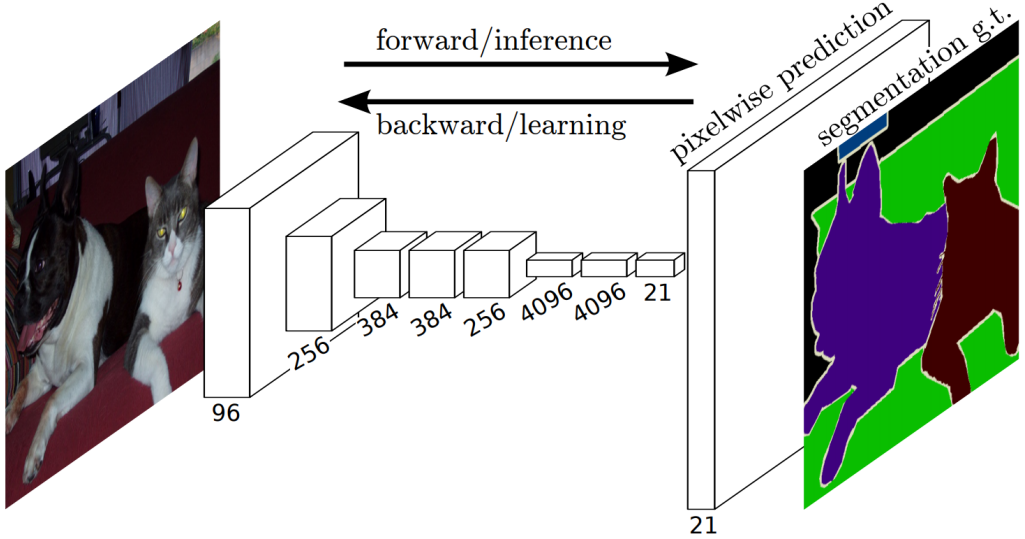
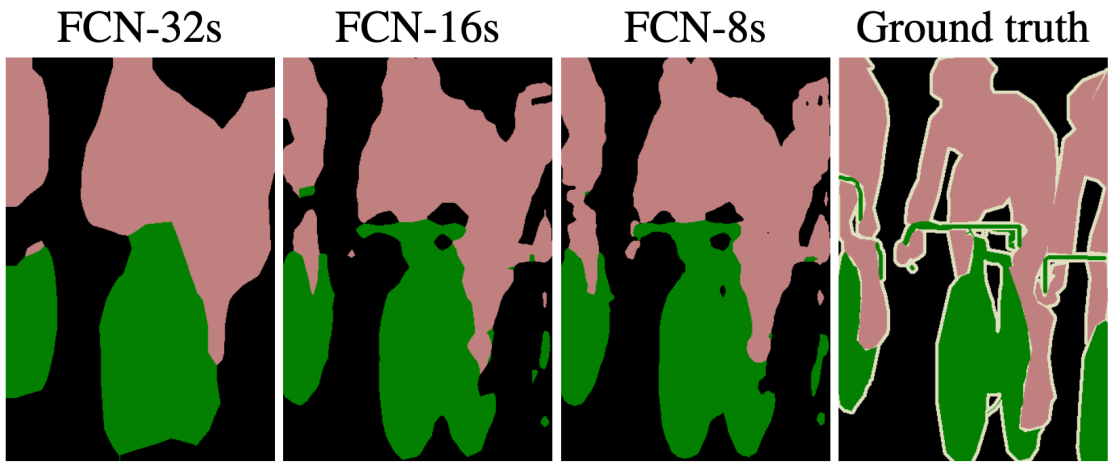



Image credit: Zeiler and Fergus

# FCN for Semantic Segmentation[7]

Carnegie Mellon



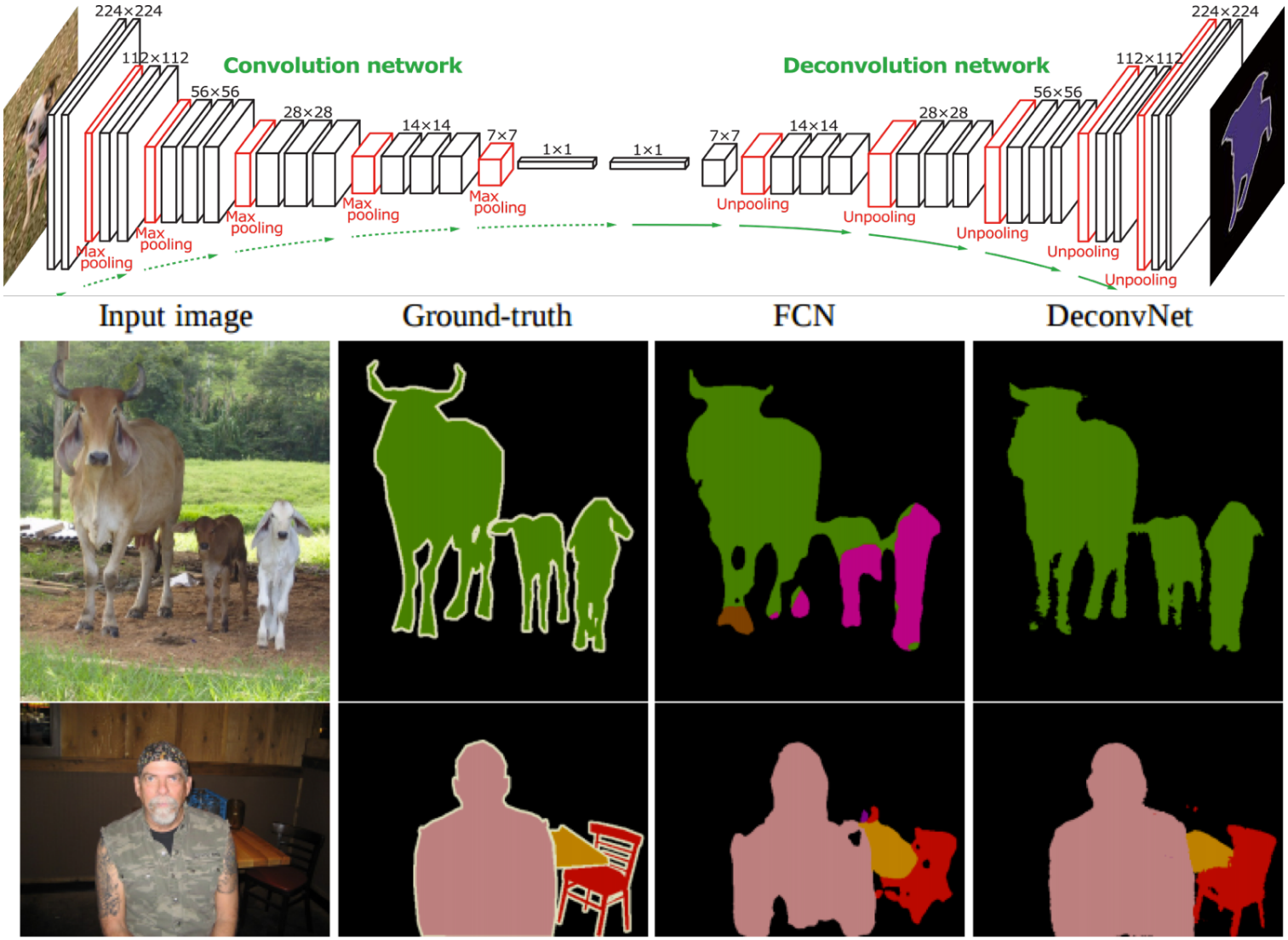




[7] https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Image credit: Jonathan Long et al.

# Deconv for Semantic Segmentation[8]

Carnegie Mellon




[8] https://arxiv.org/pdf/1505.04366.pdf

Image credit: Hyeonwoo Noh et al.

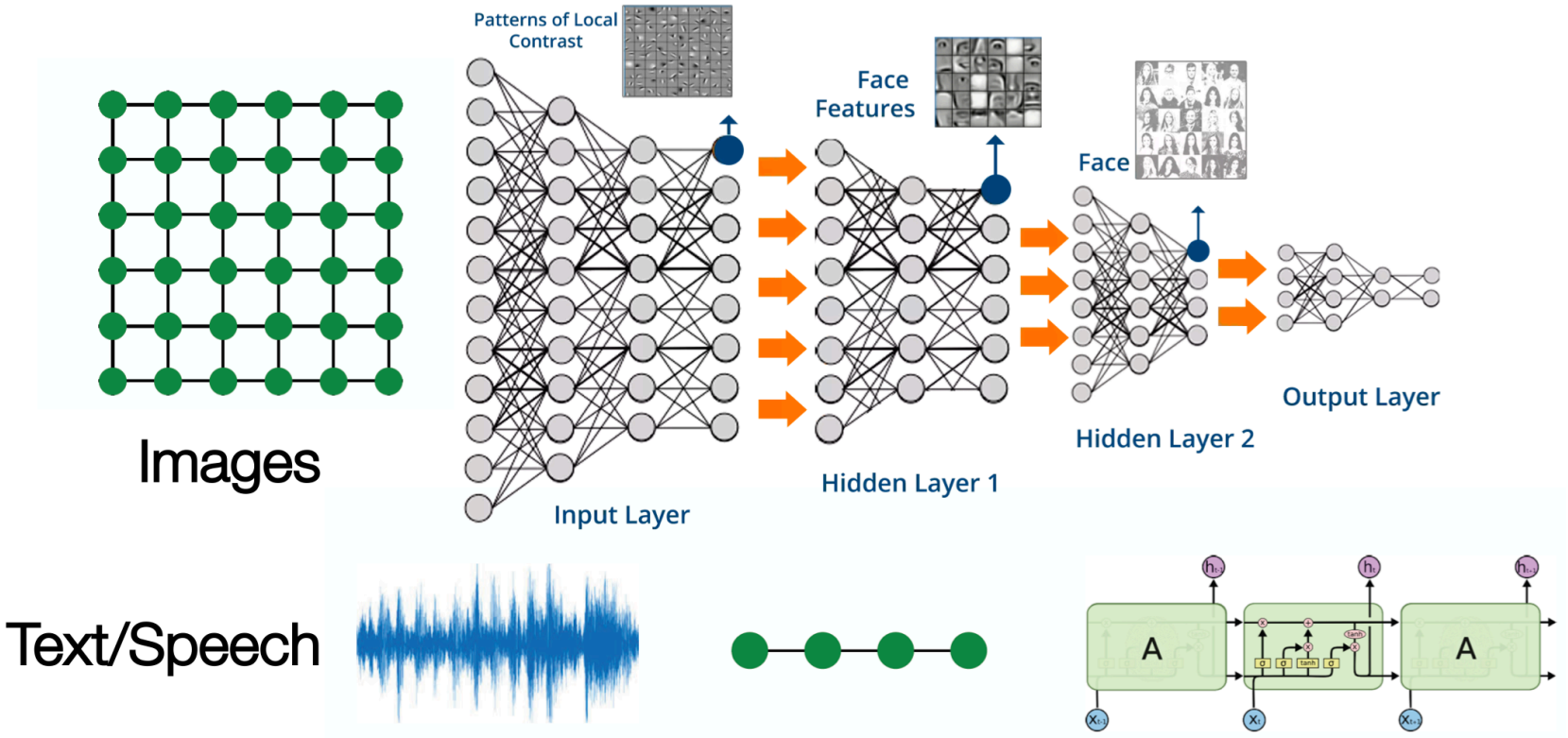
# Outline



1. Recap
2. Convolution Architectures (cont'd)
3. What CNNs Learn?
4. Graph CNN

# What we've got so far?

Carnegie Mellon



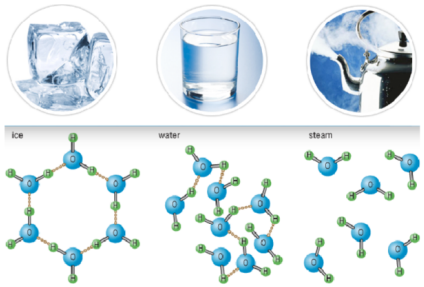

- So far we are familiar with well-structured data: grids
- Another popular type we will learn soon: sequence

Image credit: Jure Leskovec

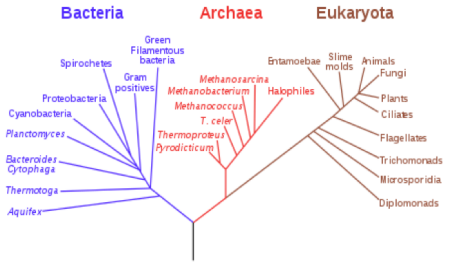
# But: lots of complex structured data

Carnegie Mellon

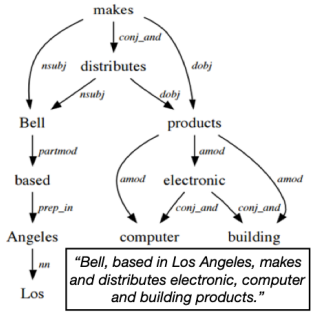
Molecules



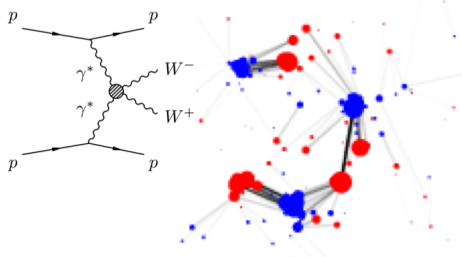

Biological species



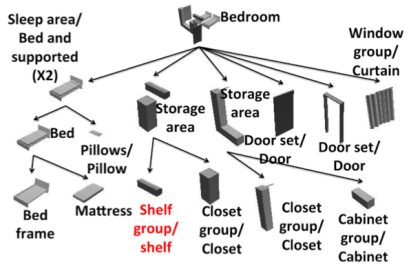

Natural language



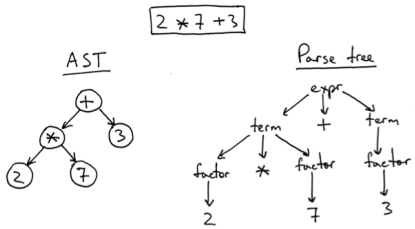

Sub-atomic particles




Everyday scenes




Code

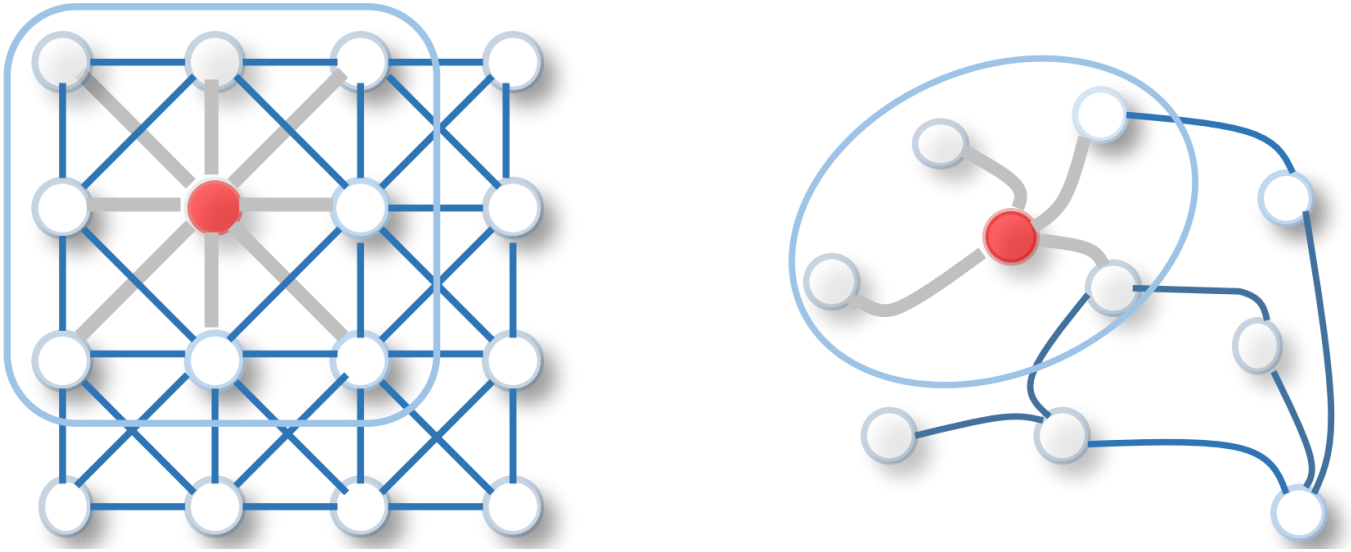



- Current tools are not suited for modeling such complex relations
- They are also usually arbitrary in size, and dynamic

Image credit: Peter Battaglia

# Similarity to grids



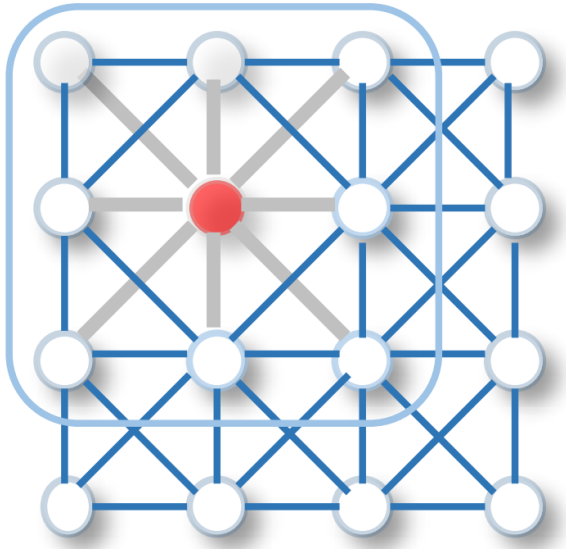


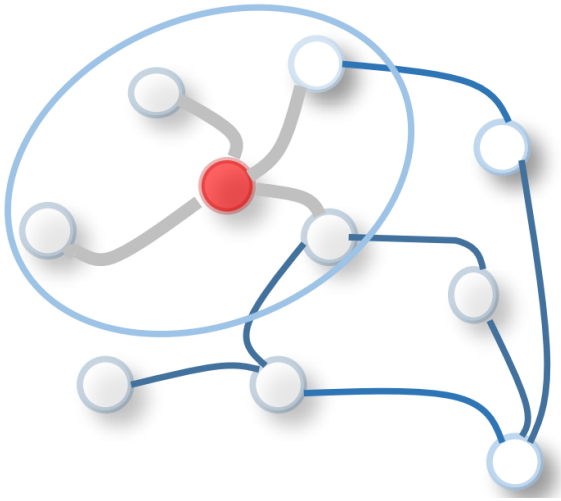
- Think grid type as a special case of graphs
- CNN node features: weighted average over itself and neighbors

Image credit: Zonghan Wu *et al.*

# Similarity to grids





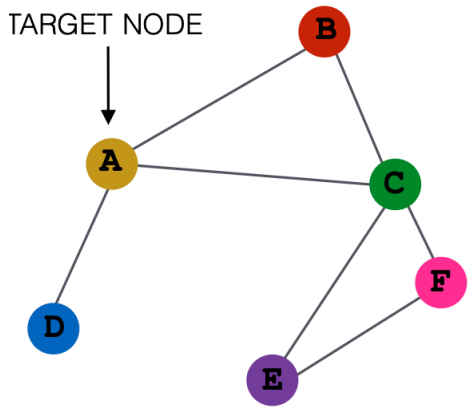


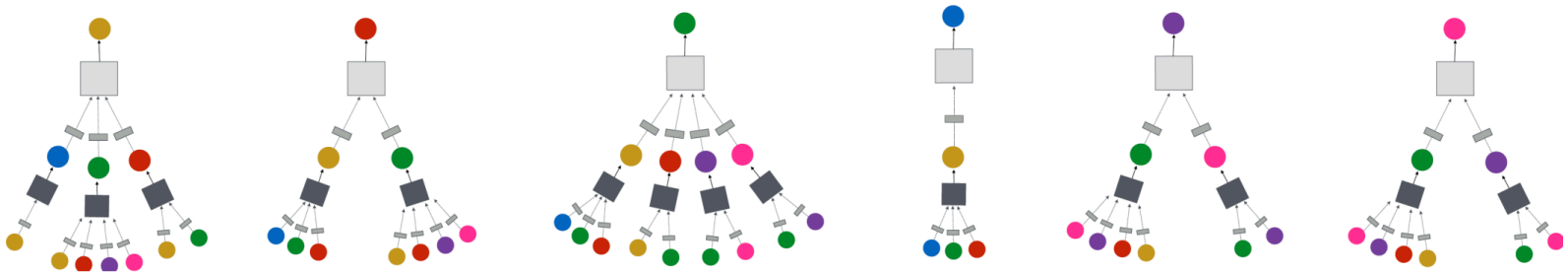
- Think grid type as a special case of graphs
- CNN node features: weighted average over itself and neighbors
- Graphs: knowing neighbors, maybe we can do a similar thing

Image credit: Zonghan Wu *et al.*

# Similarity to grids, but more complex

Carnegie Mellon



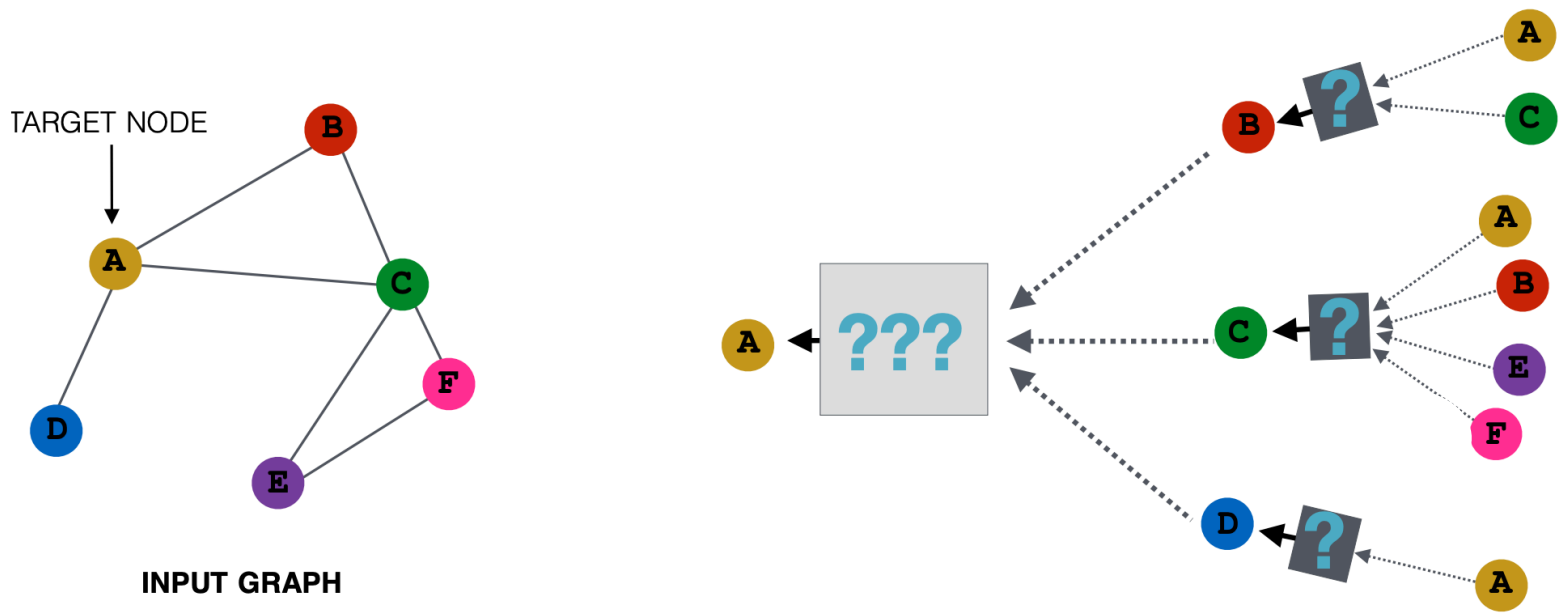



- Each node has a different, dynamic set of variable-sized neighbors
- And so has its own computational graph

Image credit: Jure Leskovec

# Idea: Aggregation



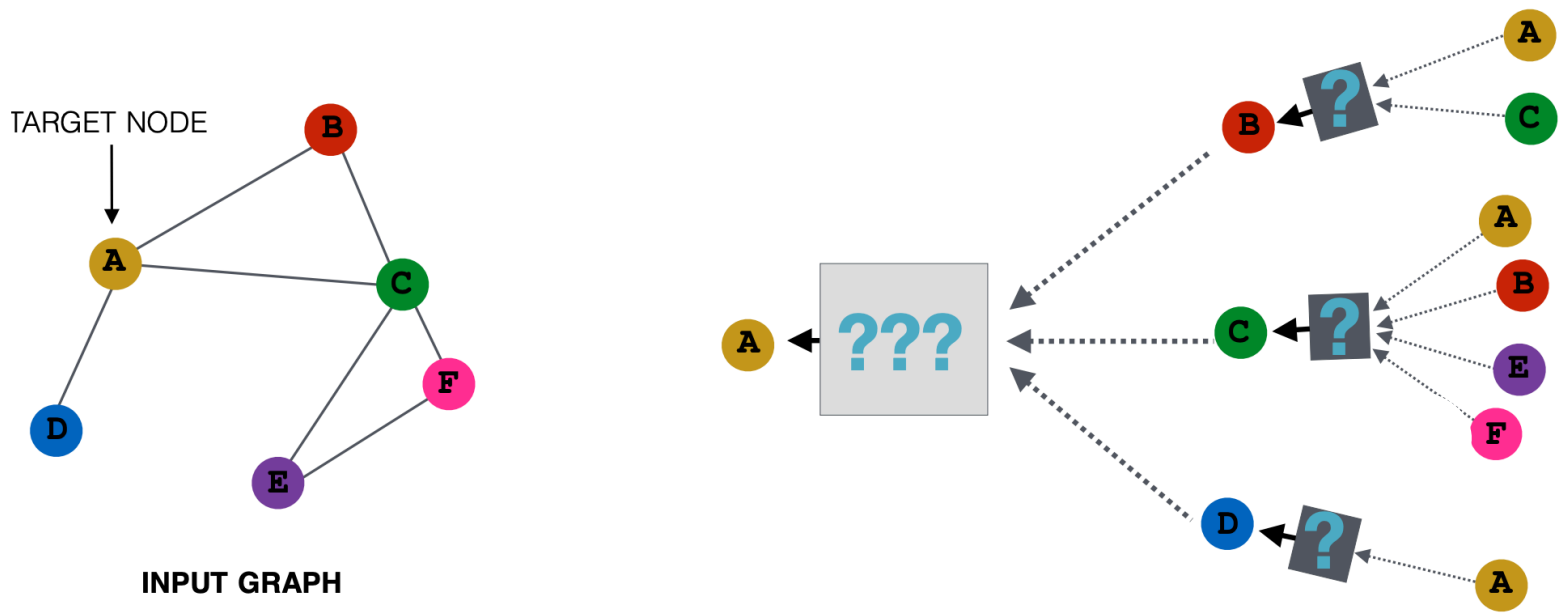



- Aggregate from its neighbors for a node's representation

Image credit: Jure Leskovec

# Idea: Aggregation






- Aggregate from its neighbors for a node's representation
- Depth (hop): chosen $k$

Image credit: Jure Leskovec

# Notations



- Graph $\mathbf{G} = (V, E)$ for sets of $V$(ertices) and $E$(dges) where $|V| = n$
- Neighborhood representation: adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
  - Let $v_i, v_j \in V$
  - Edge $e_{ij} = (v_i, v_j) \in E$ then $\mathbf{A}_{ij} = 1$ else 0
  - Some cases, $\mathbf{A}$ is weighted (not having binary values)
  - Also denote $\mathcal{N}(v)$ for the set of neighbors of $v$

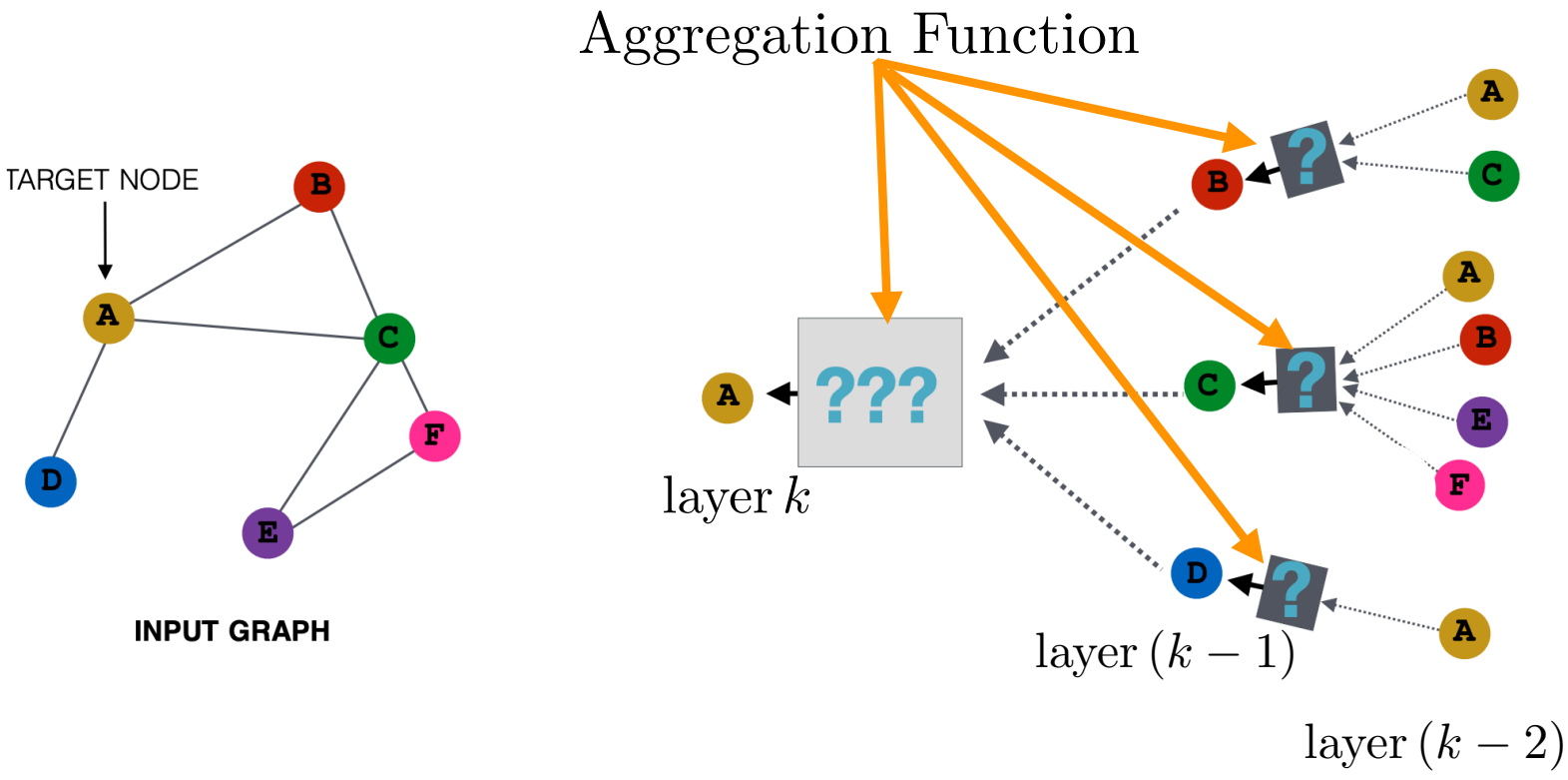
# Notations

Carnegie Mellon

- Graph $\mathbf{G} = (V, E)$ for sets of $V$(ertices) and $E$(dges) where $|V| = n$
- Neighborhood representation: adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
  - Let $v_i, v_j \in V$
  - Edge $e_{ij} = (v_i, v_j) \in E$ then $\mathbf{A}_{ij} = 1$ else 0
  - Some cases, $\mathbf{A}$ is weighted (not having binary values)
  - Also denote $\mathcal{N}(v)$ for the set of neighbors of $v$
- Rich information: each node $v \in V$ has a feature vector $\mathbf{X}_v \in \mathbb{R}^d$

# Idea: Aggregation






- At layer $k$, representation of $v$ is $z_v^{(k)}$

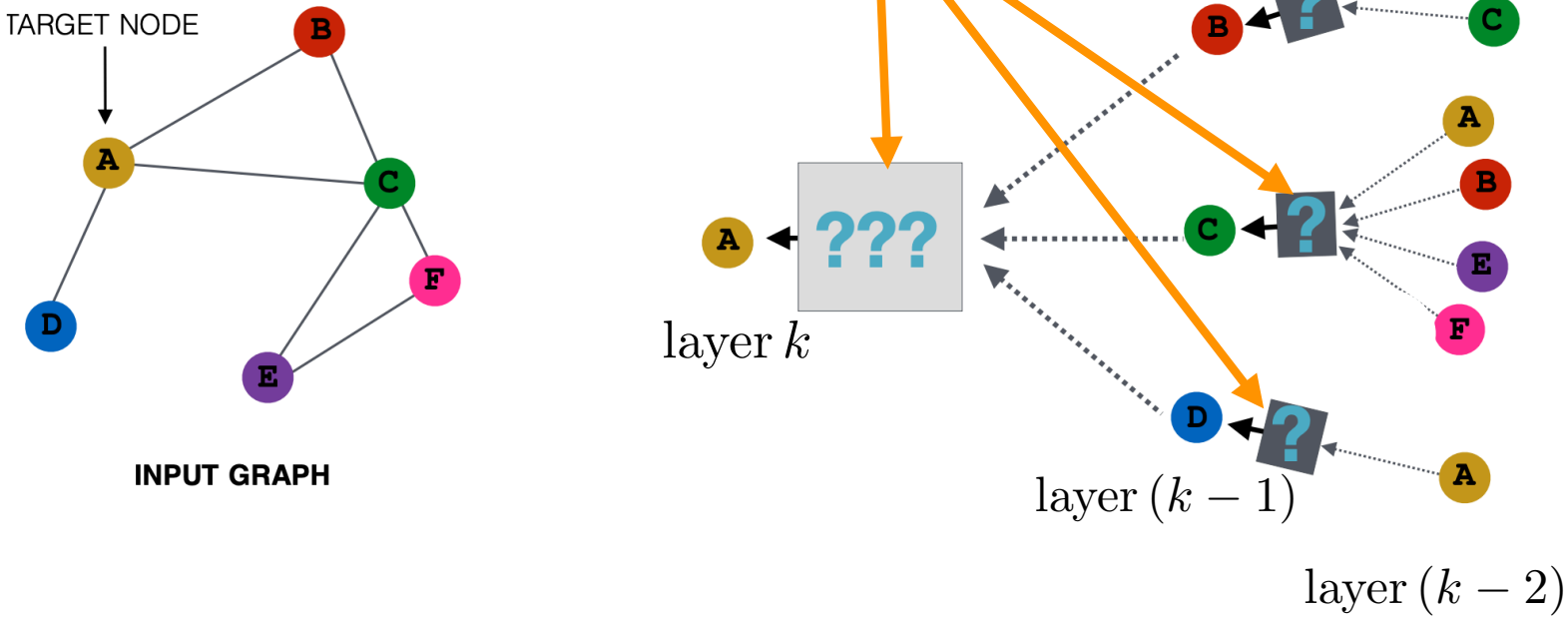
$$z_v^{(k)} = \mathbf{f}_{\mathbf{W}^{(k)}}(z_v^{(k-1)}, z_{v\_neighbor}^{(k)}) \quad \text{(update from prev step)}$$

$$z_{v\_neighbor}^{(k)} = \mathbf{g}_{\mathbf{C}^{(k)}}(\mathcal{N}(v)) \quad \text{(aggregation)}$$

Image credit: Jure Leskovec

# Idea: Aggregation






- At each layer $k$, we have learnable matrix-weights $\mathbf{W}^{(k)}, \mathbf{C}^{(k)}$

Image credit: Jure Leskovec

# More on Aggregation

Carnegie Mellon

- Initialization at layer 0: $z_v^{(0)} = \mathbf{X}_v \in \mathbb{R}^d$
- There are various choices for update function $\mathbf{f}_{\mathbf{W}^{(k)}}$ and aggregation function $\mathbf{g}_{\mathbf{C}^{(k)}}$
  - Graph Convolutional Networks[9], they're combined as:

$$z_v^{(k)} = \text{ReLU}\left(\mathbf{W}^{(k)} \sum_{u \in \mathcal{N}(v) \cup v} \frac{z_v^{(k-1)}}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}\right)$$

  - GraphSAGE[10], they're:

$$z_v^{(k)} = \mathbf{f}_{\mathbf{W}^{(k)}}(z_v^{(k-1)}, z_{v\_neighbor}^{(k)}) = \mathbf{W}^{(k)}\,[z_v^{(k)}, z_{v\_neighbor}^{(k)}]$$

$$z_{v\_neighbor}^{(k)} = \mathbf{g}_{\mathbf{C}^{(k)}}(\mathcal{N}(v)) = \max\left(\text{ReLU}\left(\mathbf{C}^{(k)} \sum_{u \in \mathcal{N}(v)} \frac{z_v^{(k-1)}}{|\mathcal{N}(u)|}\right)\right)$$

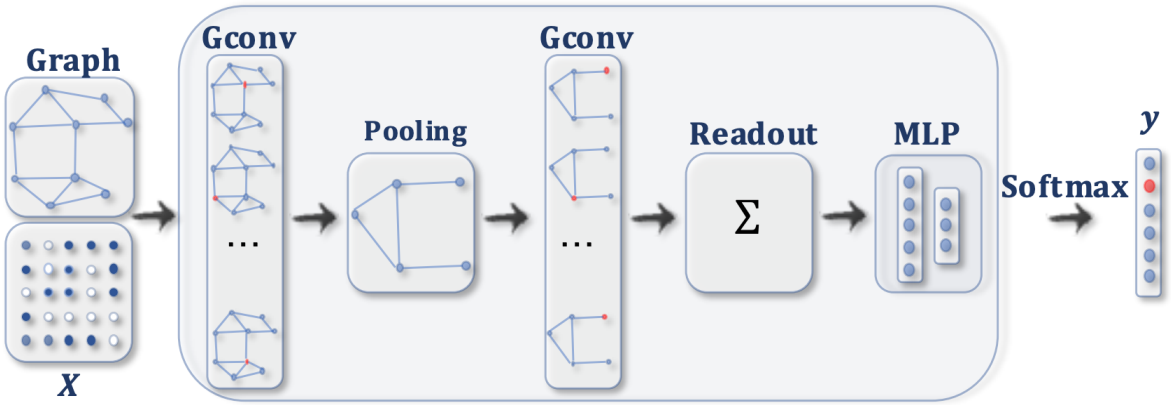
[9] https://arxiv.org/pdf/1609.02907.pdf

[10] https://arxiv.org/pdf/1709.05584.pdf

# Graph Convolutional Tasks

Carnegie Mellon

- Node classification (identify each type of node): $z_v^{(k)}$ to classify node $v$
- Graph classification: given a set of sub-graphs with labels $\{(G_i, y_i) : G_i \in \mathbf{G}, y_i \in \mathbf{Y}\}$, learn the label for the entire graph $\mathbf{G}$




`Readout` function can be simple as avg or complex as pooling[11]

[11] `https://arxiv.org/pdf/1806.08804.pdf`

Image credit: Zonghan Wu *et al.*

# Model Params

Carnegie Mellon

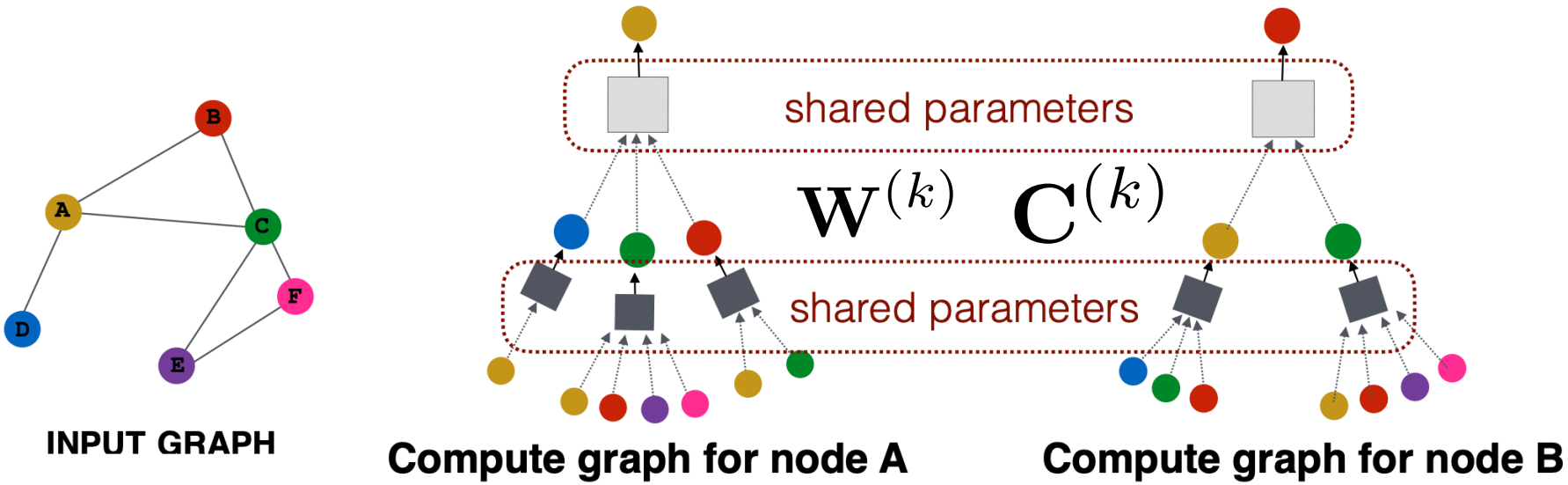
- Each node has its own computational graph, so we have too many params?

Image credit: Jure Leskovec

# Model Params

Carnegie Mellon

- Each node has its own computational graph, so we have too many params?




- No, each layer has shared params

Image credit: Jure Leskovec

# Shared Params → Inductive Capability

Carnegie Mellon

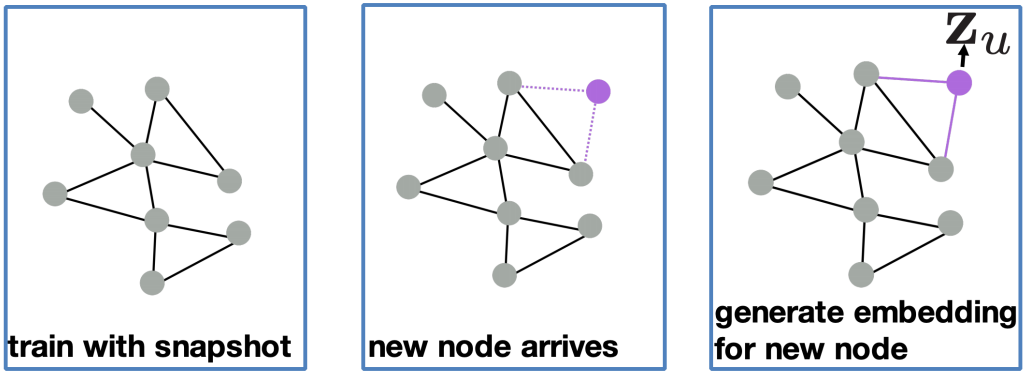
- Generate representation for new nodes on the fly, *e.g.* Twitter, Facebook, GoogleScholar, ...

Image credit: Jure Leskovec

# Shared Params → Inductive Capability

Carnegie Mellon

- Generate representation for new nodes on the fly, *e.g.* Twitter, Facebook, GoogleScholar, ...




- Generate representation for the new graph, *e.g.* protein interactions for a new organism

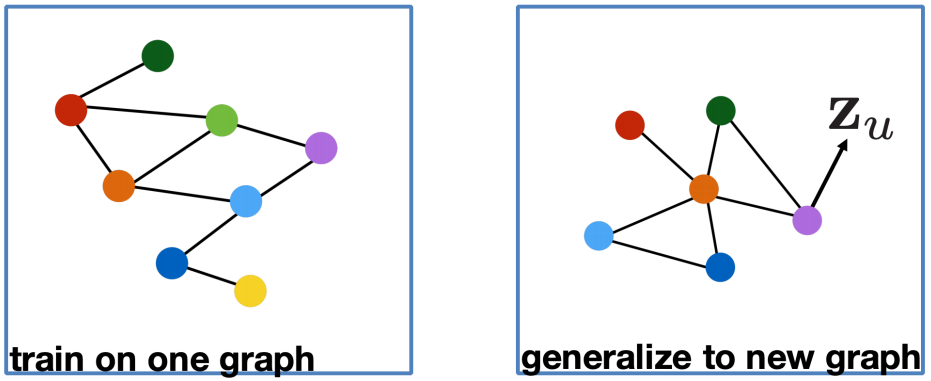



Image credit: Jure Leskovec

End of Module II.