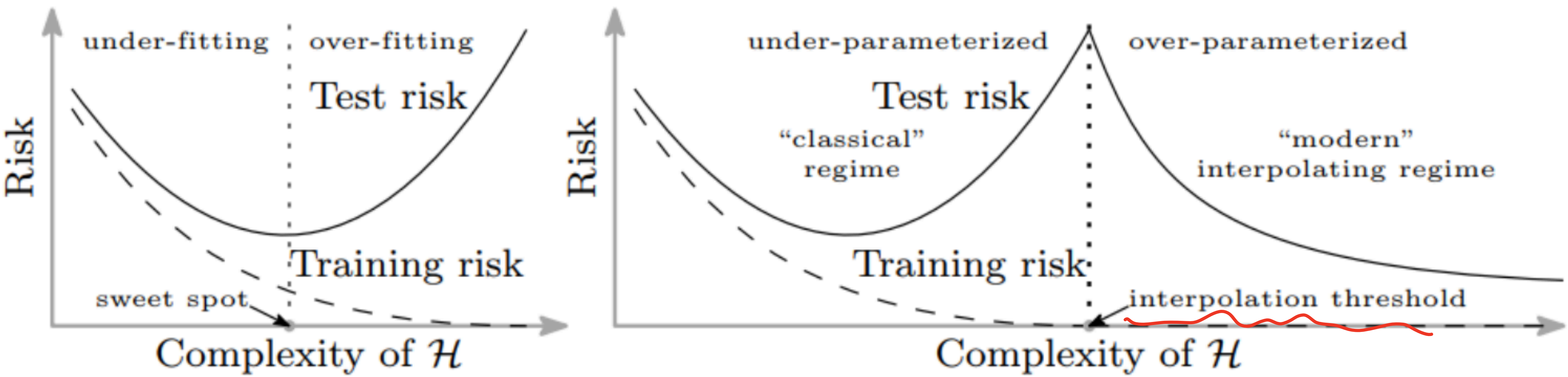
Deep Learning Generalization

Double descent



(a) U-shaped "bias-variance" risk curve

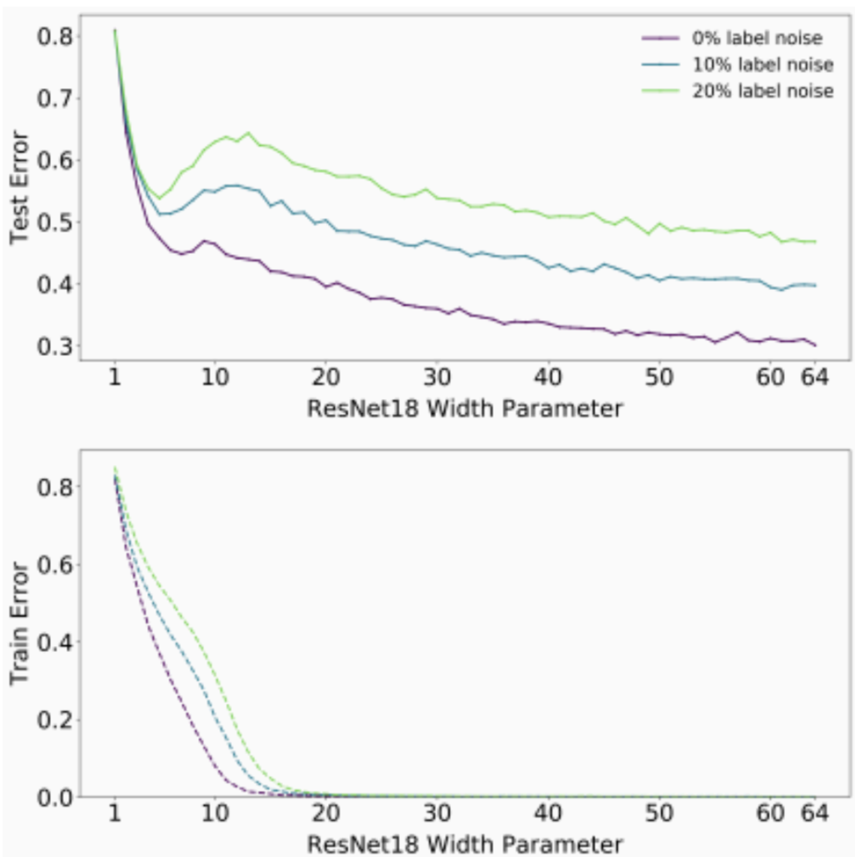
(b) "double descent" risk curve

Belkin, Hsu, Ma, Mandal '18

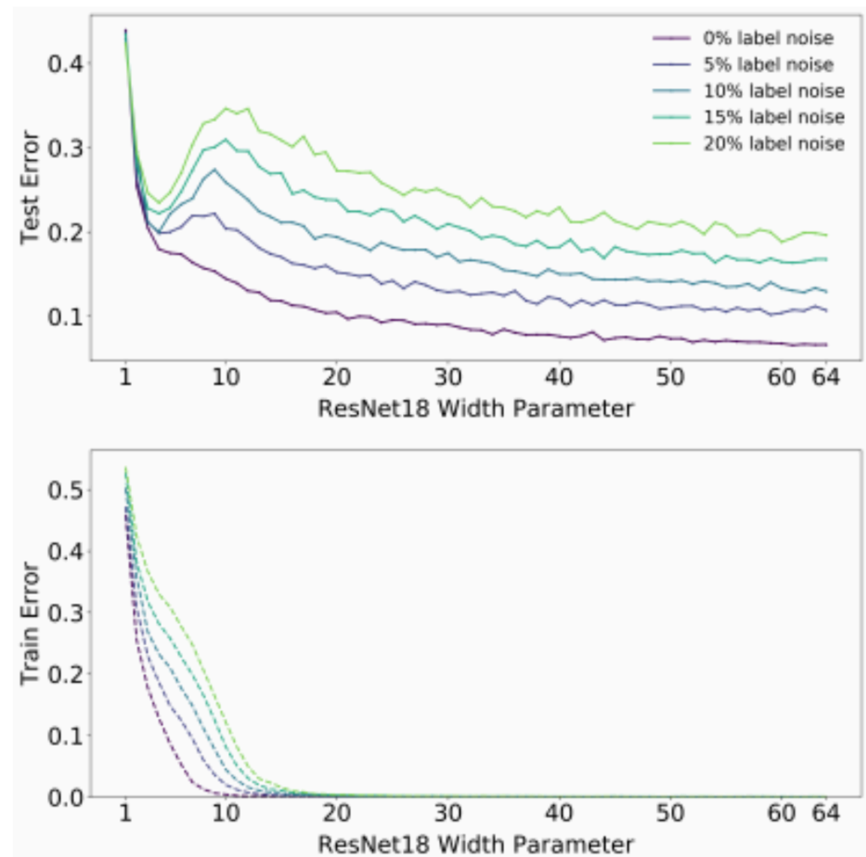
- There are cases where the model gets bigger, yet the (test!) loss goes down, sometimes even lower than in the classical "under-parameterized" regime.
- Complexity: number of parameters.

Double descent

Widespread phenomenon, across architectures (Nakkiran et al. ’19):



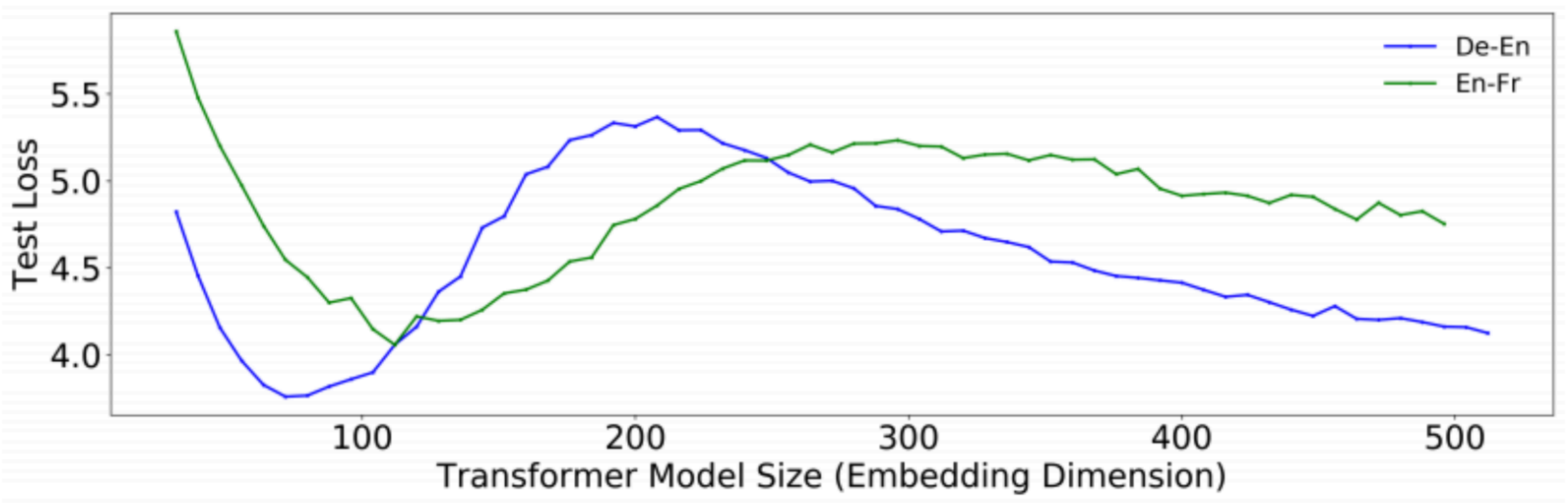
(a) **CIFAR-100.** There is a peak in test error even with no label noise.



(b) **CIFAR-10.** There is a “plateau” in test error around the interpolation point with no label noise, which develops into a peak for added label noise.

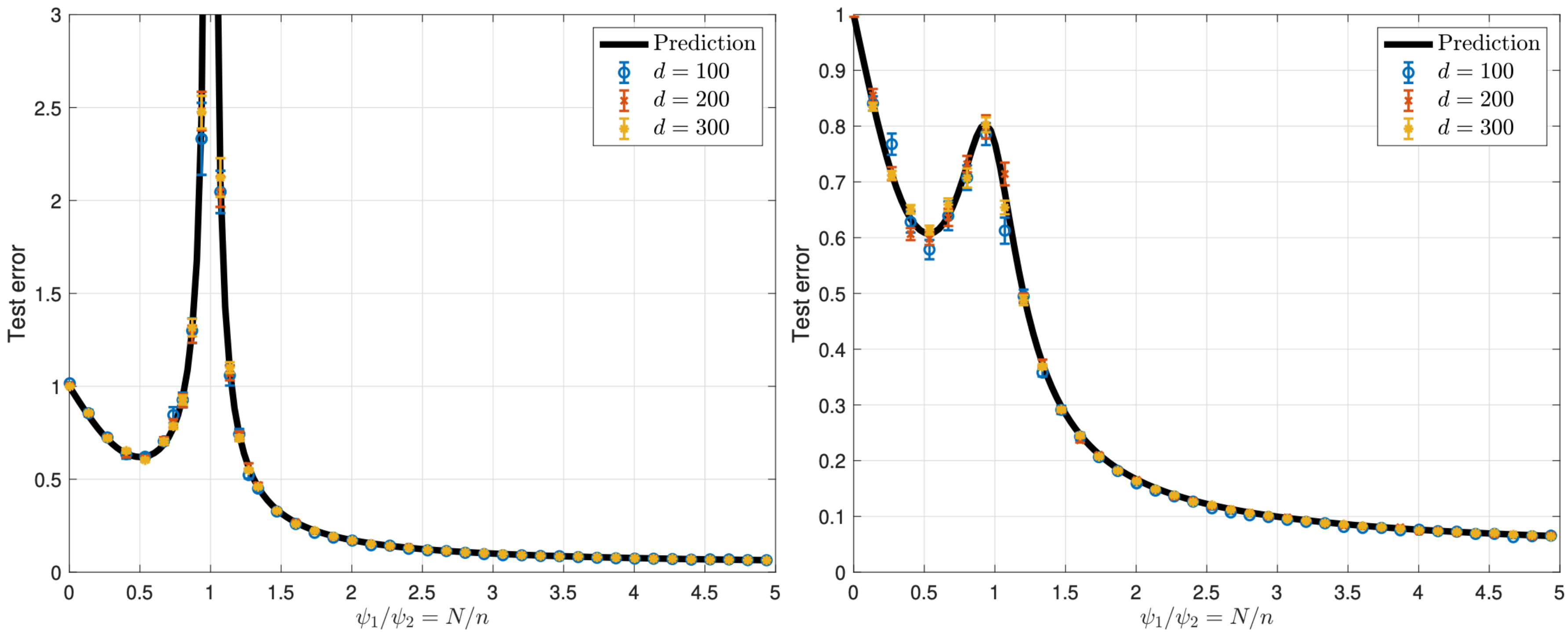
Double descent

Widespread phenomenon, across architectures (Nakkiran et al. ’19):



Double descent

Widespread phenomenon, also in kernels (can be formally proved in some concrete settings [Mei and Montanari ’20]), random forests, etc.



Double descent

Also in other quantities such as train time, dataset, etc (Nakkiran et al. '19):

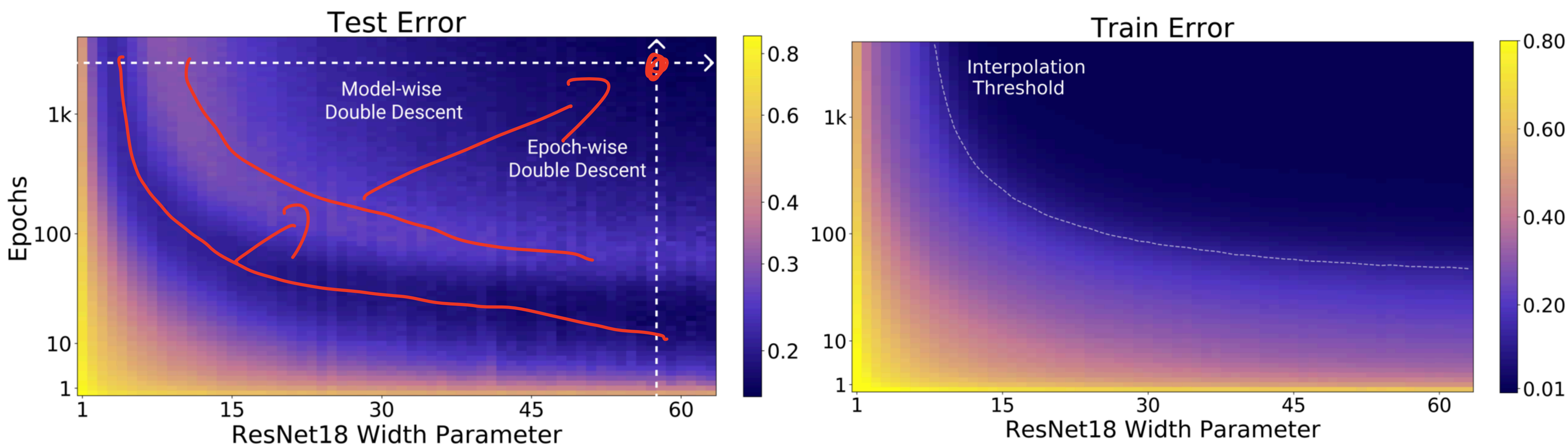
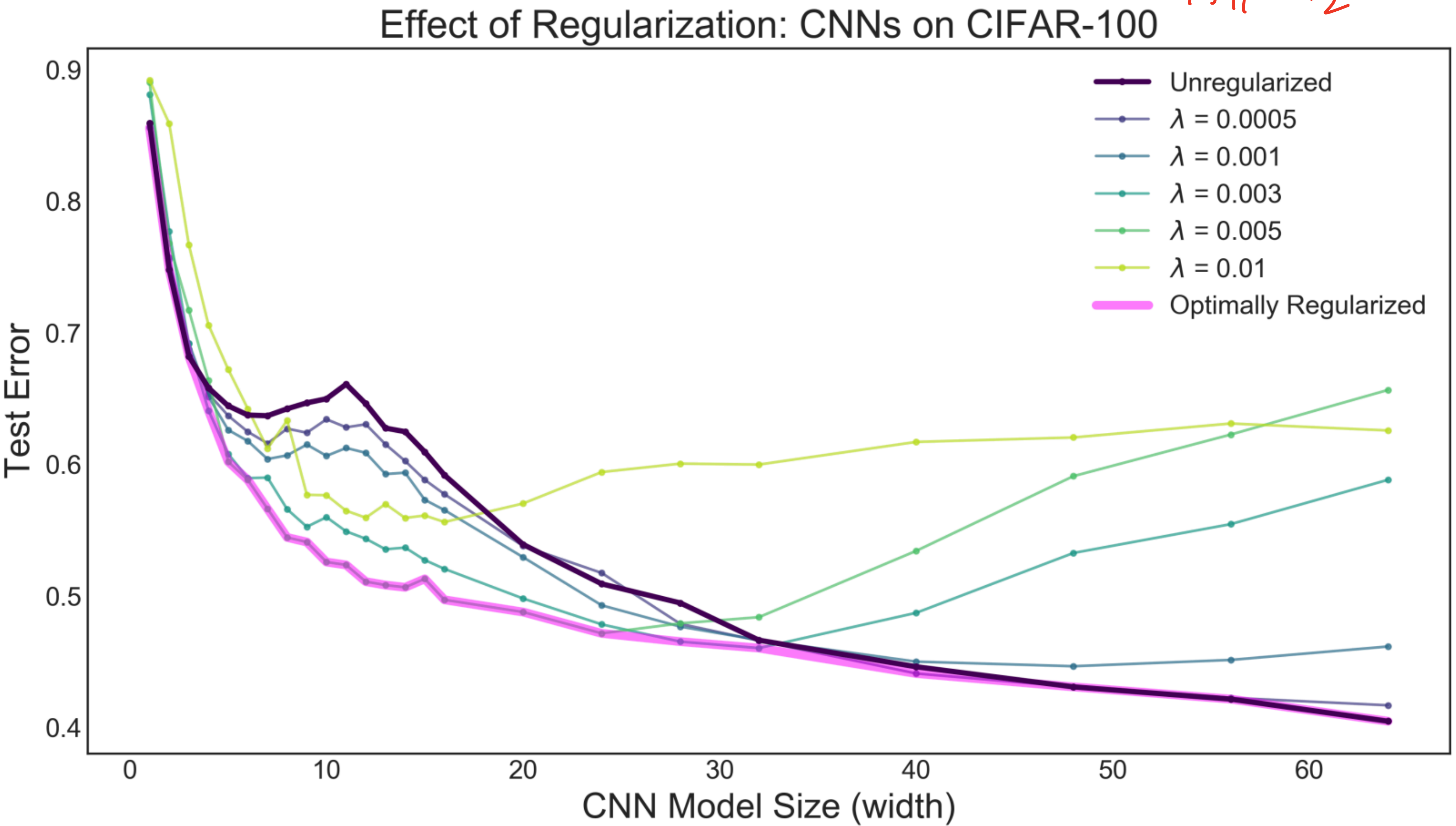


Figure 2: **Left:** Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent–varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as train time increases. **Right** Train error of the corresponding models. All models are Resnet18s trained on CIFAR-10 with 15% label noise, data-augmentation, and Adam for up to 4K epochs.

Double descent

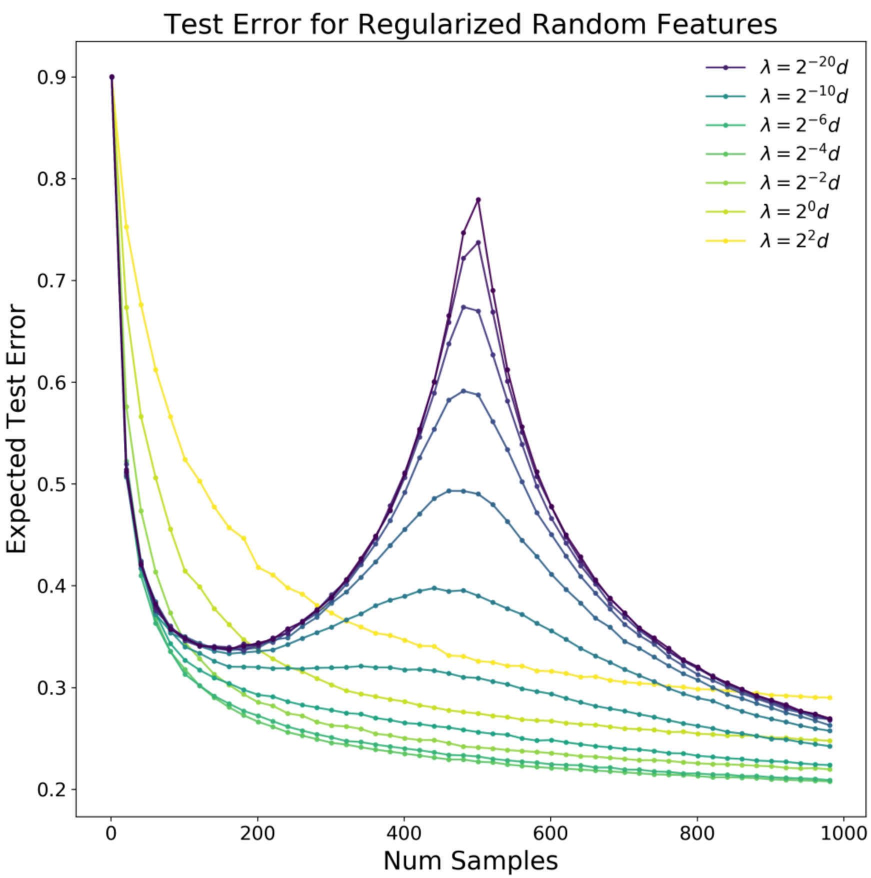
Optimal regularization can mitigate double descent [Nakkiran et al. '21]:

$\lambda \|w\|_2^2$

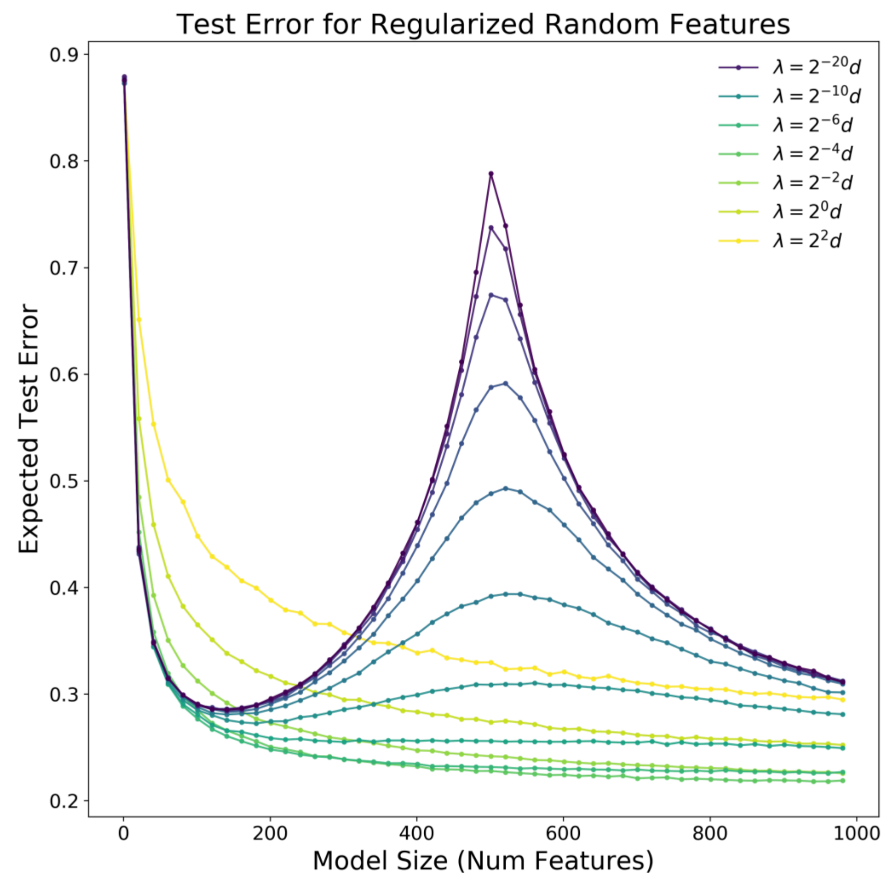


Double descent

Optimal regularization can mitigate double descent [Nakkiran et al. ’21]:



a) Test Classification Error vs. Number of Train-
ng Samples.



(b) Test Classification Error vs. Model Size (Number of Random Features).

explicit regularization:

$\lambda \|w\|_2^2$, $\lambda \|w\|_1$

dropout

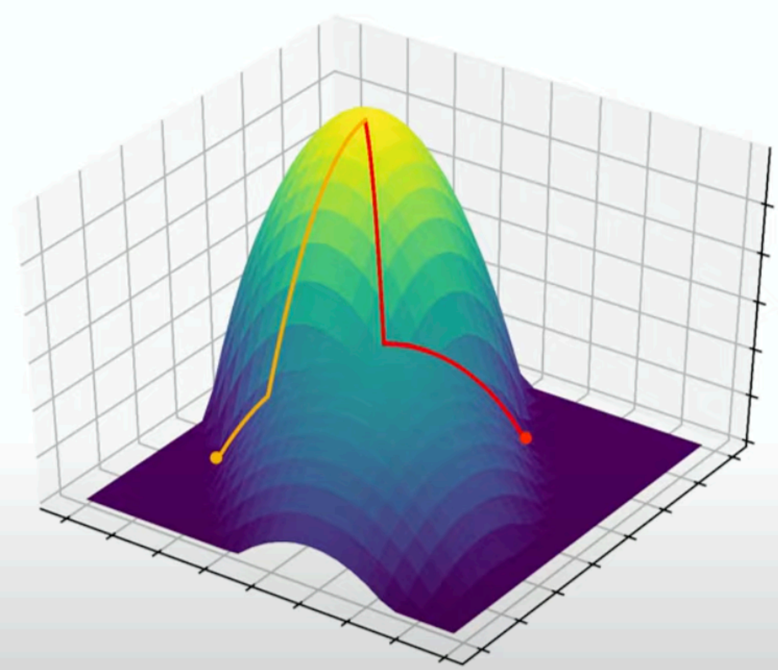
Implicit Regularization

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties

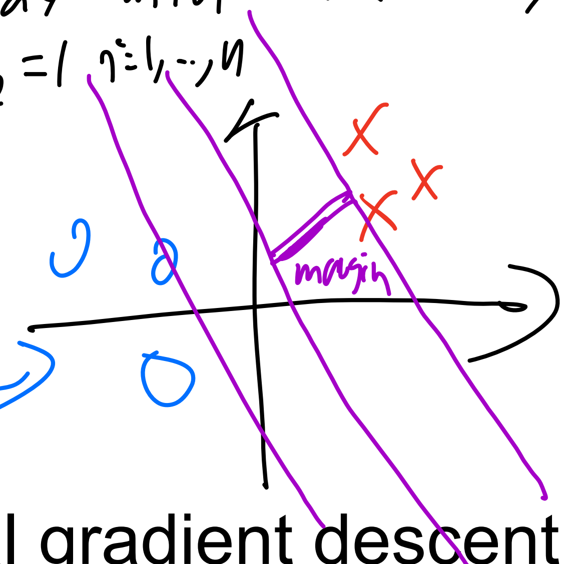


Implicit Bias

linear $\quad \bar{u} = \arg\max_{\|u\|_2=1} \min_{i=1,\dots,n} y_i\langle w, x_i\rangle$

Margin: $\quad m_i \stackrel{\Delta}{=} \min(W) = y_i \cdot f(W, x_i)$

$(H+1)$-homogeneous $\quad$ margin: $\bar{u} = \dfrac{\min_i m_i}{\|W\|^{H+1}}$



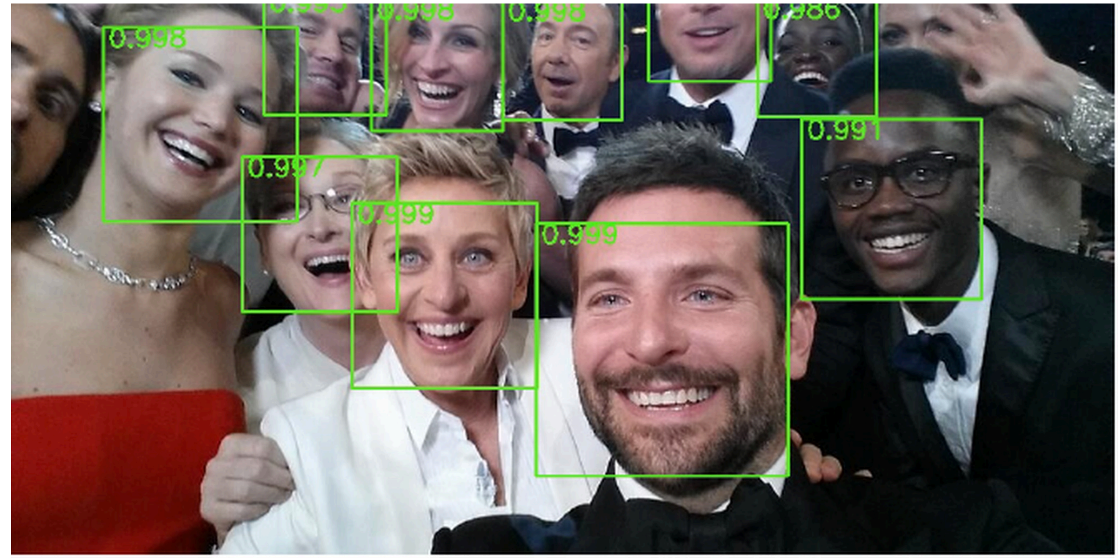
- Linear predictors: on logistic loss
  - Gradient descent, mirror descent, natural gradient descent, steepest descent, etc maximize margins with respect to different norms. $\|w\|_p = 1$

- Non-linear:
  - Gradient descent maximizes margin for homogeneous neural networks.
  - Low-rank matrix sensing: gradient descent finds a low-rank solution.

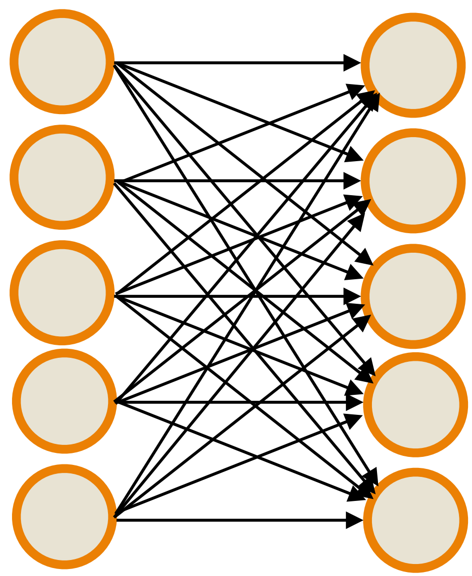
Convolutional Neural Networks

Neural Network Architecture

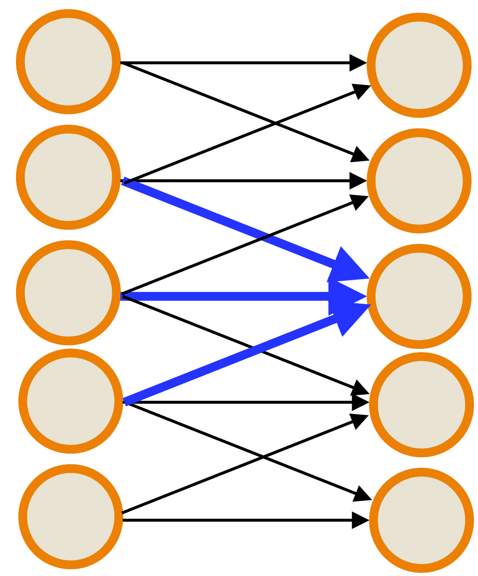
Objects are often **localized in space** so to find the faces in an image, not every pixel is important for classification—makes sense to drag a window across an image.



Similarly, to identify edges or other local structure, it makes sense to only look at **local information**

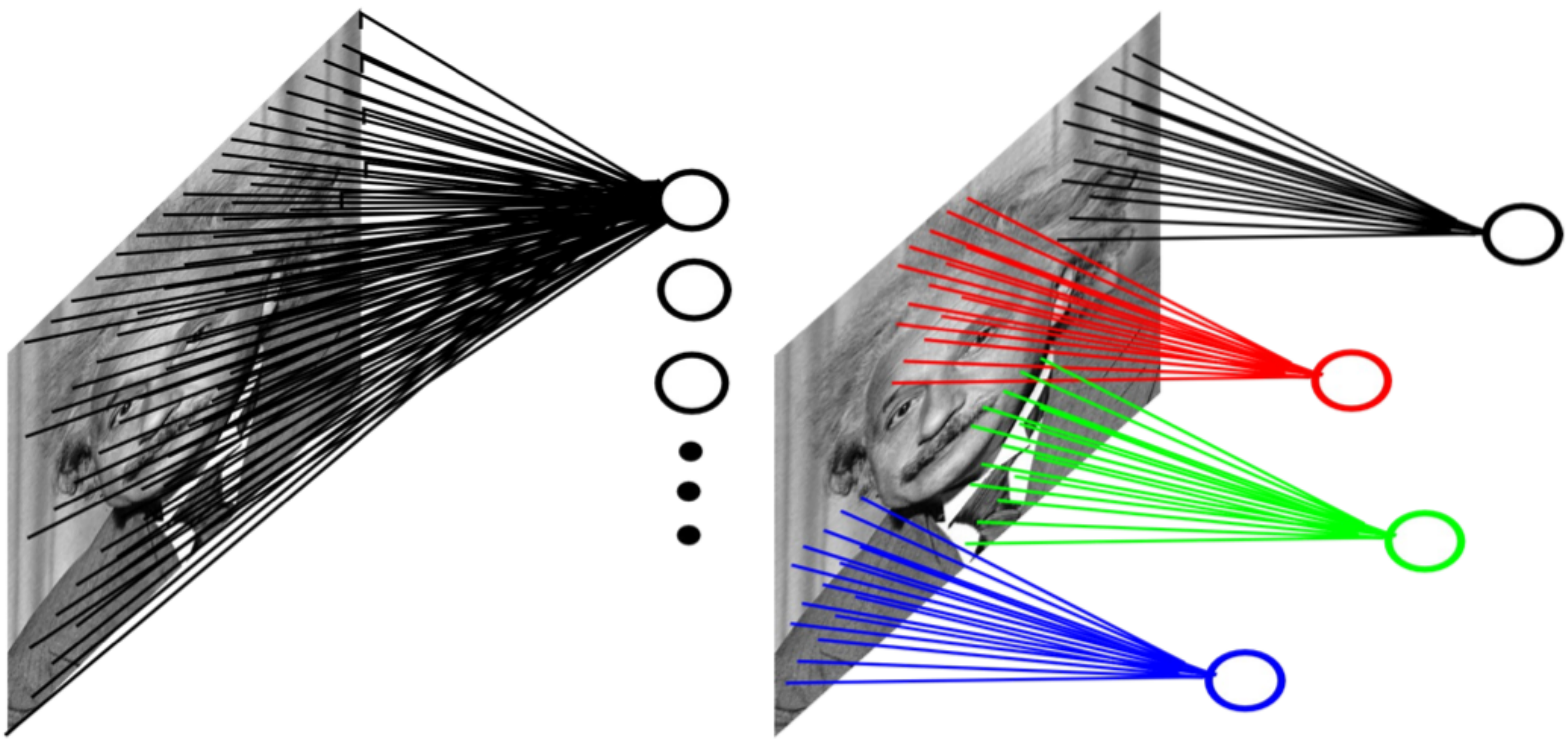


vs.



2d Convolution Layer

- **Example: 200x200 image**
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
  - Local connections capture local dependencies



Convolution of images (2d convolution)

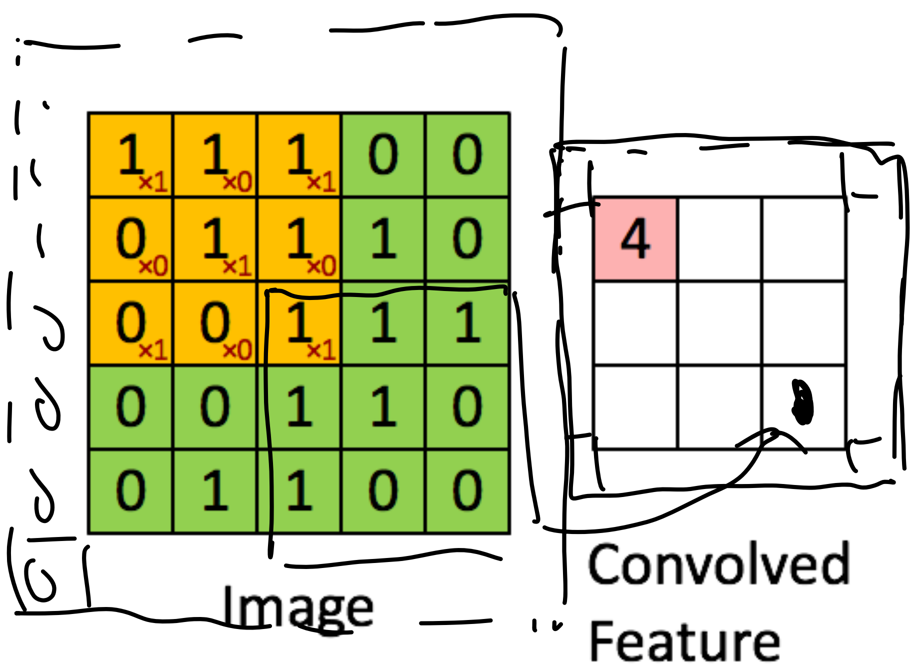
$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image I

1	0	1
0	1	0
1	0	1

Filter K

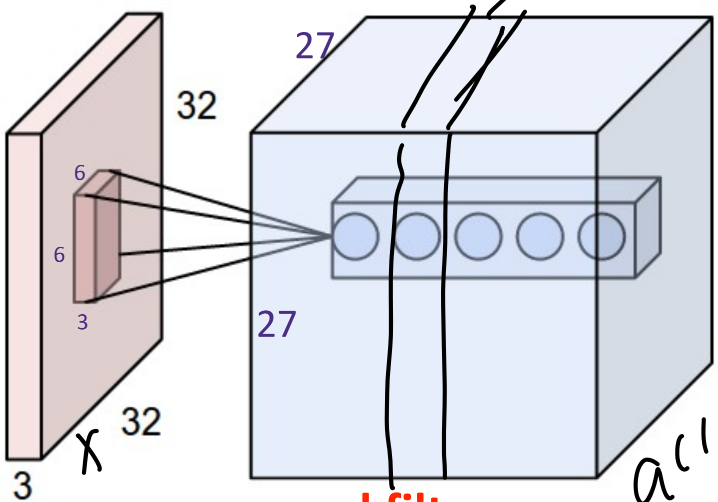


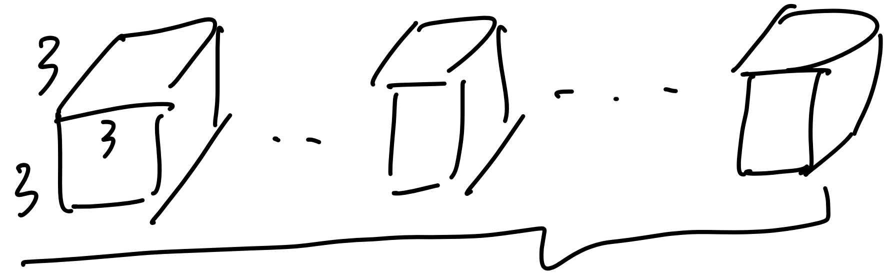
$I * K$

Stacking convolved images

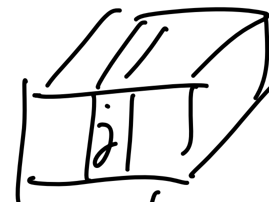
g: activation function

$a_j^{(1)}$





Repeat with d filters!



$a^{(2)}$

d filters

$a^{(1)}: 27 \times 27 \times d$

R G B

$$a_j^{(1)} = g\left(\sum_{\alpha=1}^{3} X[:,:,\alpha] * K_j^{(1)}[:,:,\alpha]\right)$$

6

$$a_j^{(2)} = g\left(\sum_{i=1}^{d} a^{(1)}[:,:,i] * K_j^{(2)}[:,:,i]\right)$$

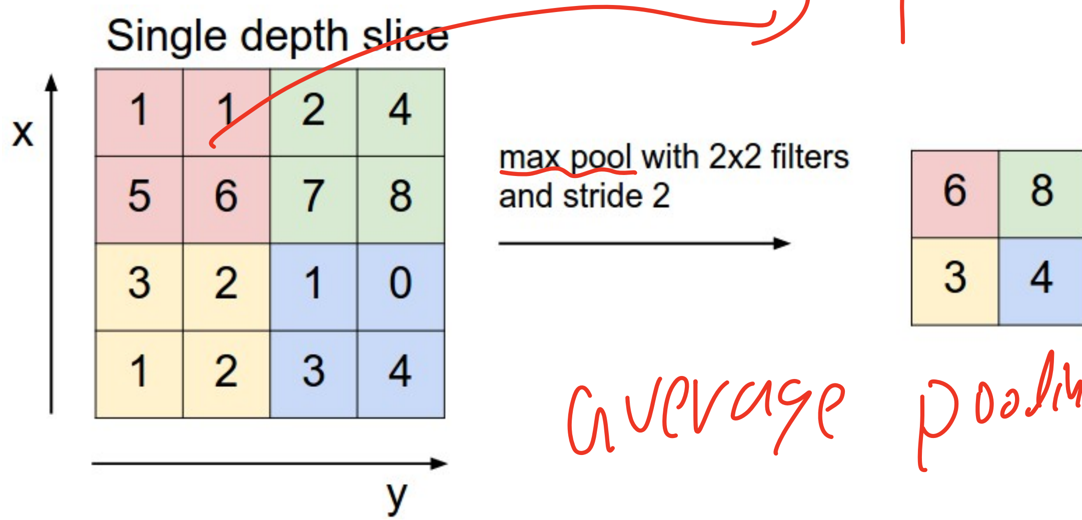
Second layer

$K_j^{(2)}$

kernel

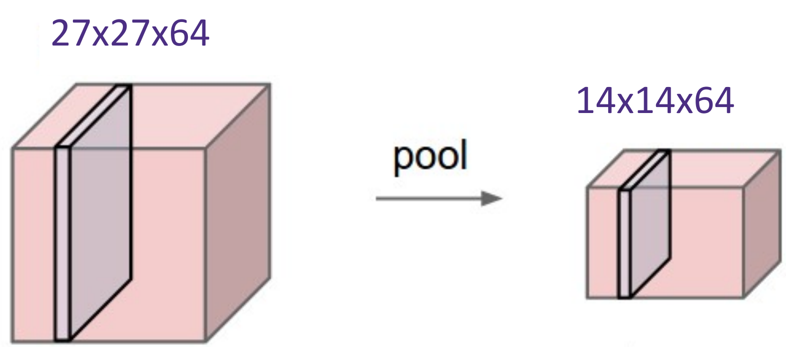
Pooling

Pooling reduces the dimension and can be interpreted as “This filter had a high response in this general region”

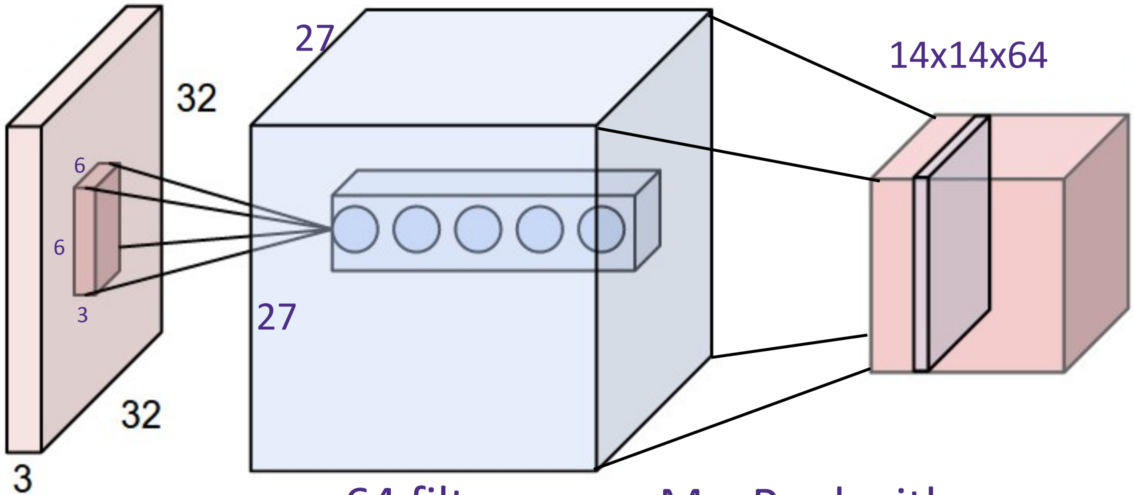


$\frac{13}{4}$

average pooling



Pooling Convolution layer

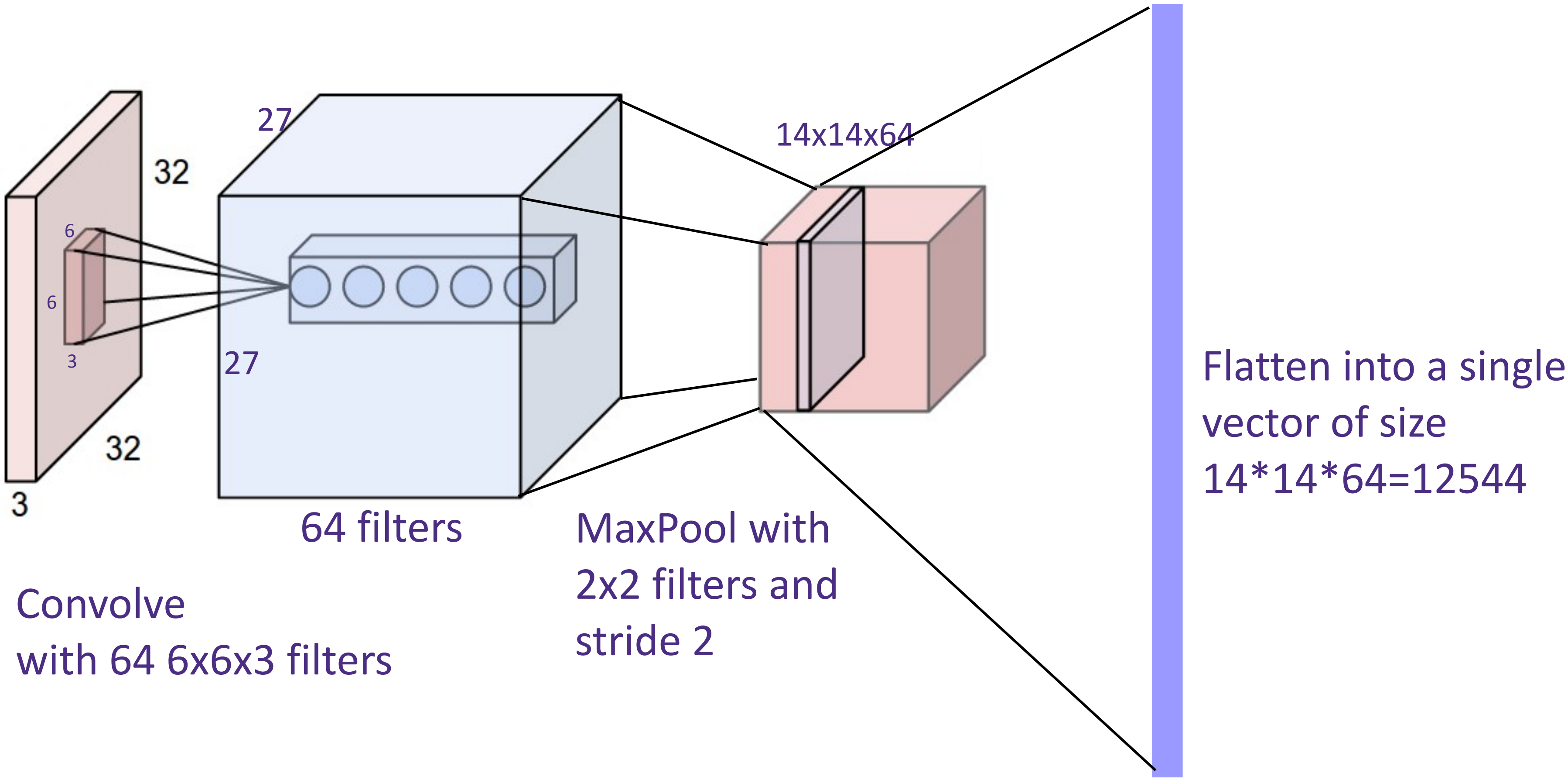


64 filters

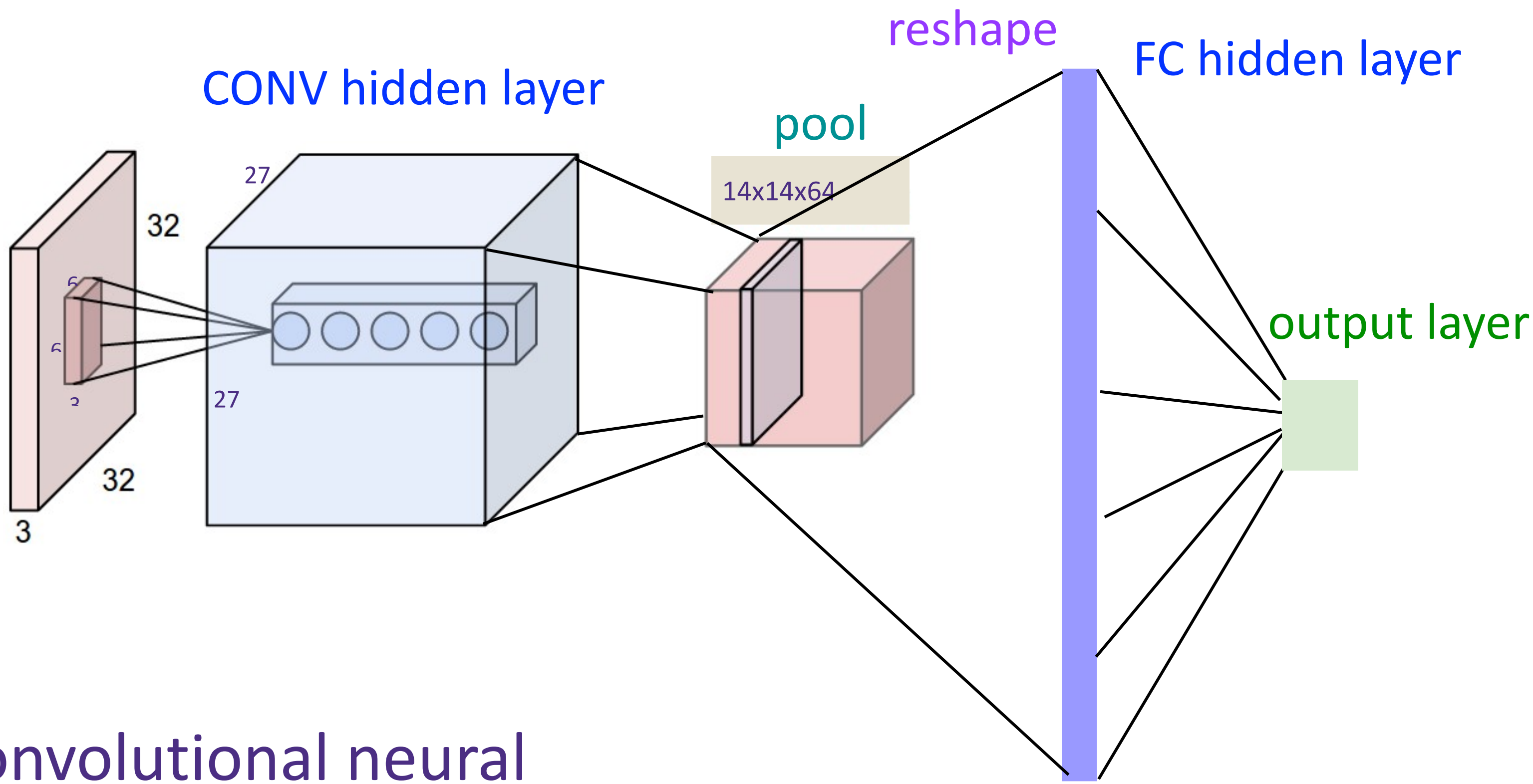
MaxPool with
2x2 filters and
stride 2

Convolve
with 64 6x6x3 filters

Flattening

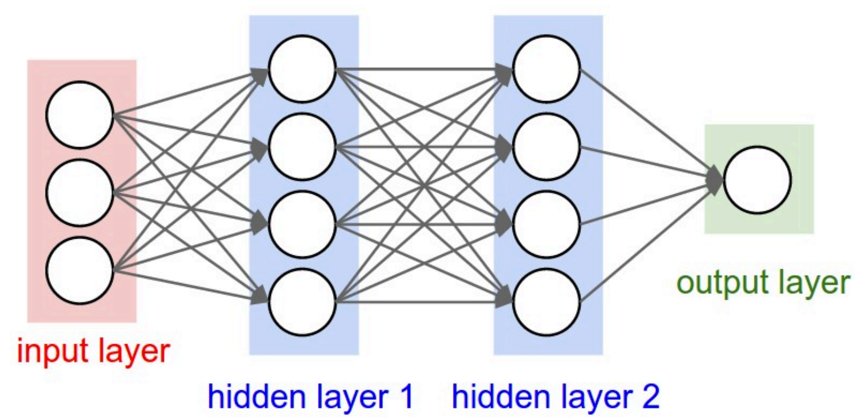

27
32
14x14x64
6
6
3
27
32
3
64 filters
Flatten into a single vector of size 14*14*64=12544
MaxPool with 2x2 filters and stride 2
Convolve with 64 6x6x3 filters

Training Convolutional Networks

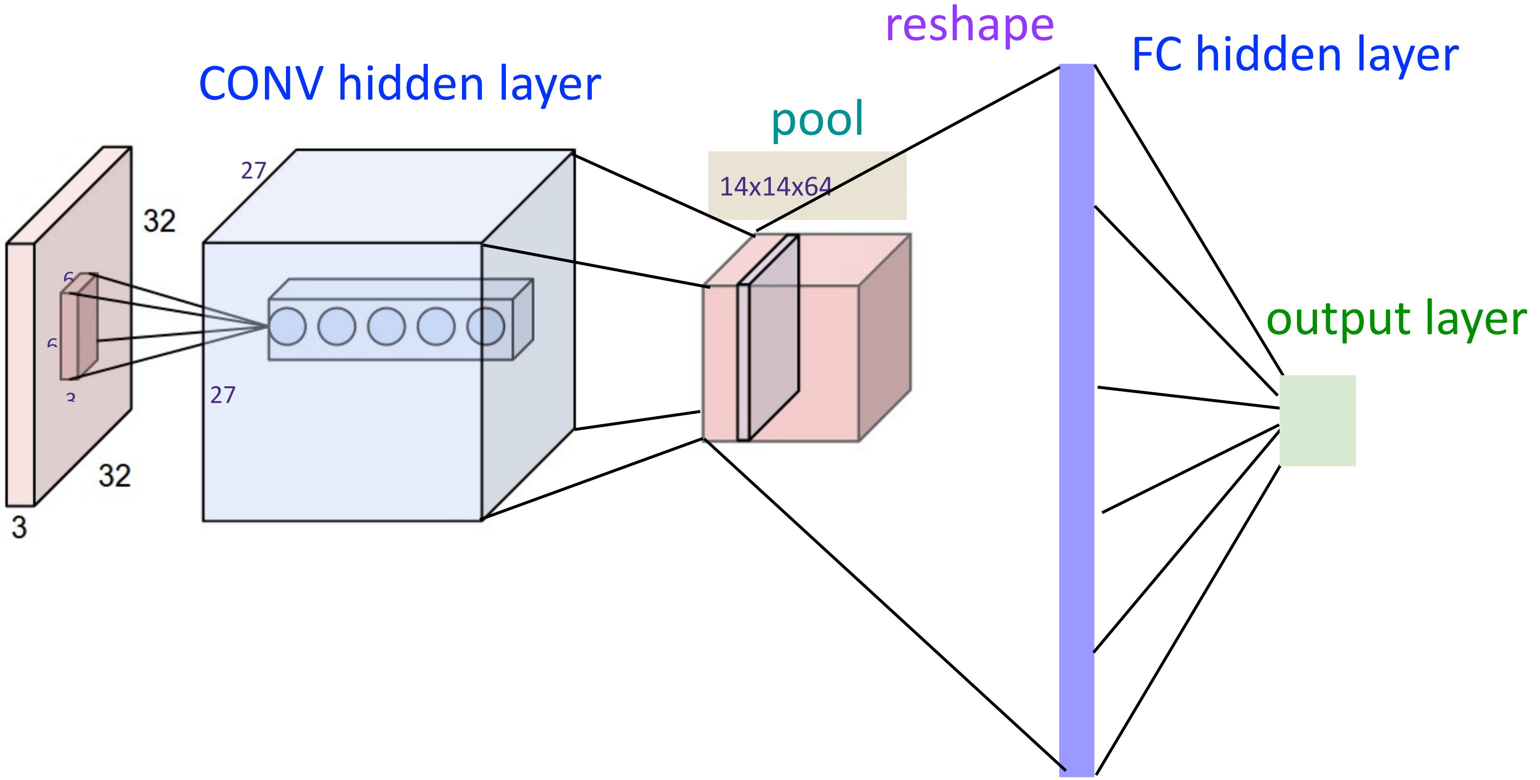


Recall: Convolutional neural networks (CNN) are just regular fully connected (FC) neural networks with some connections removed.

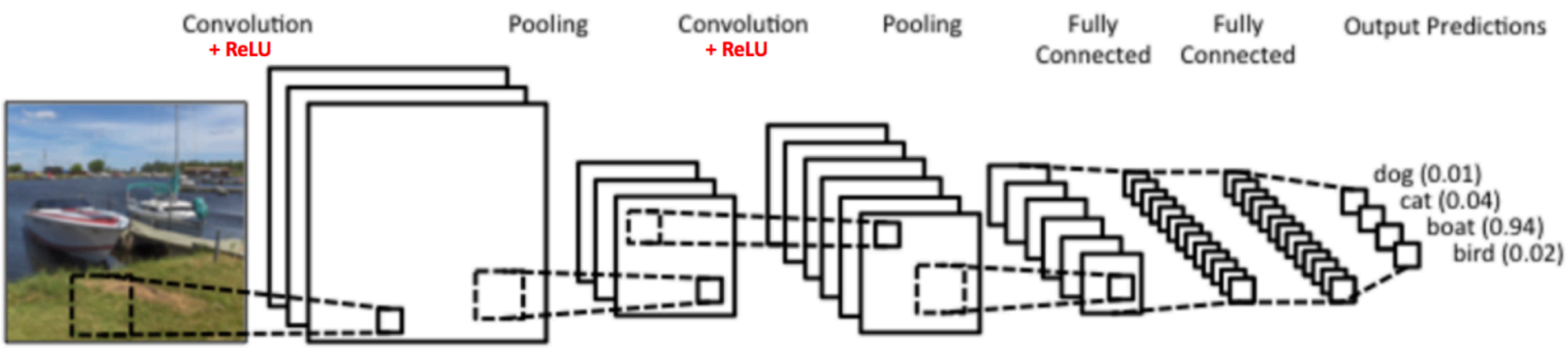
Train with SGD!

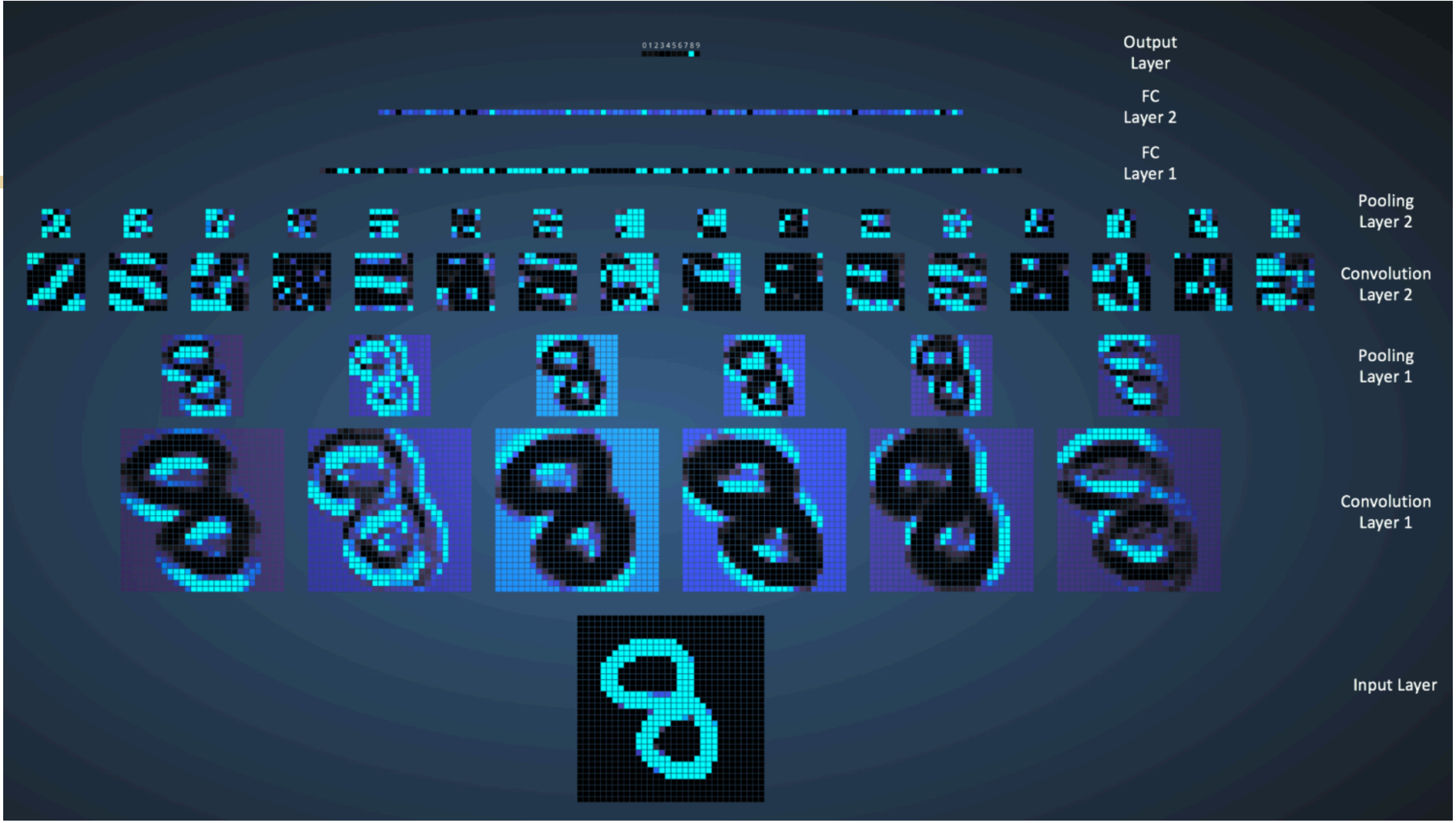


Training Convolutional Networks

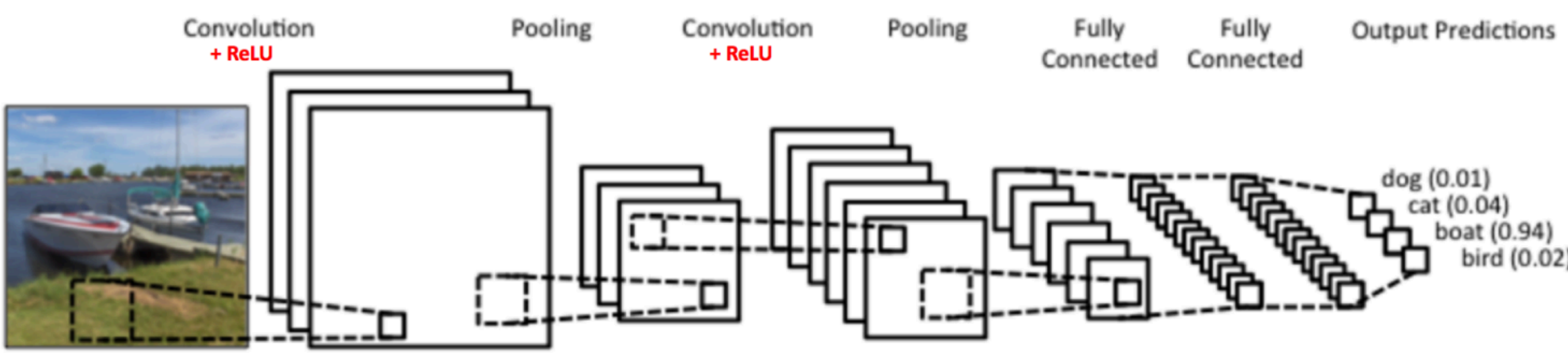


Real example network: LeNet





Real example network: LeNet



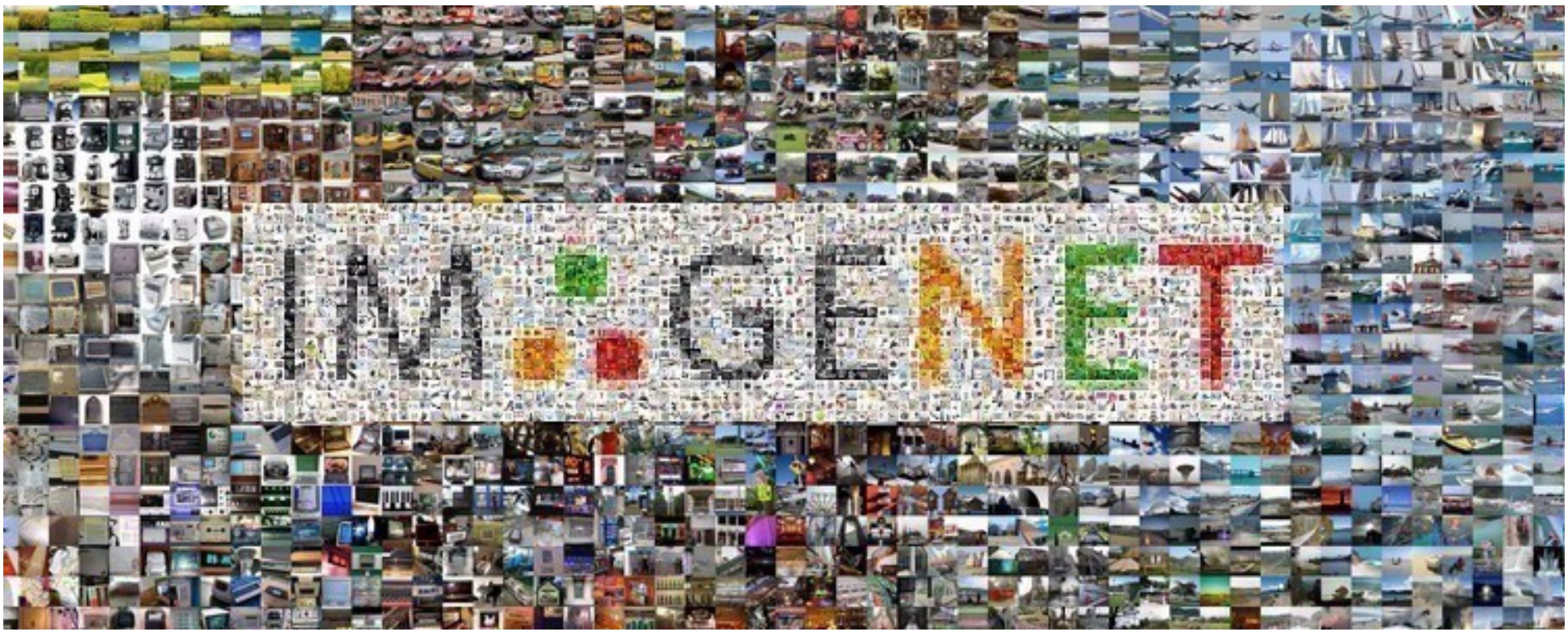
Famous CNNs

ImageNet Dataset

top 5

top 1

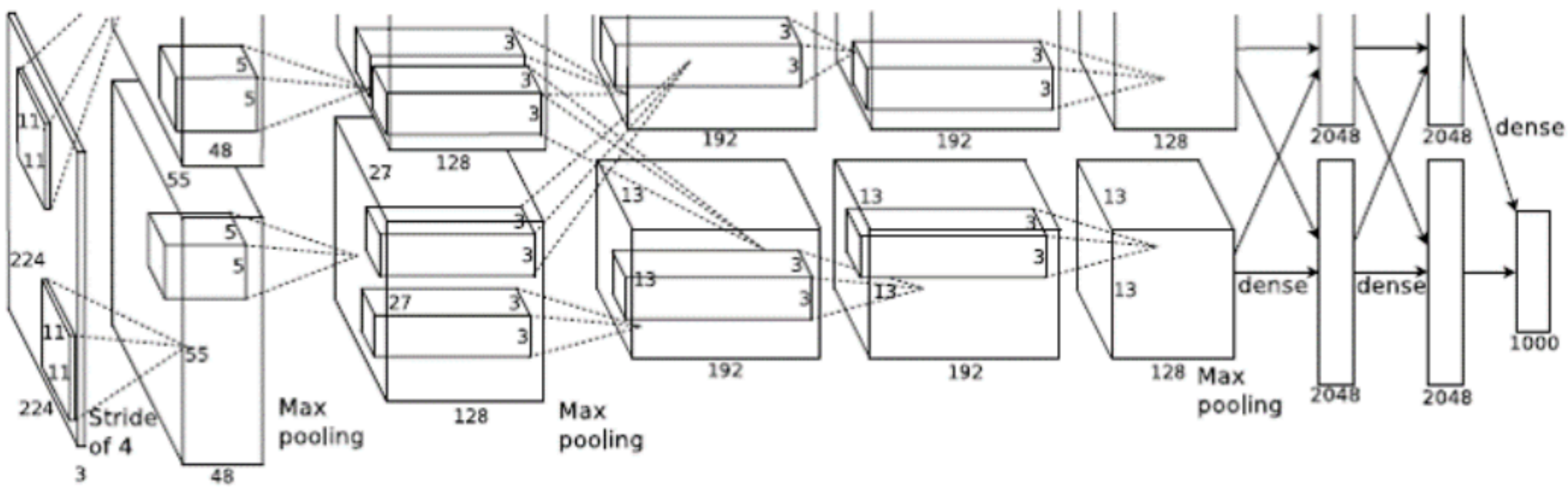
~14 million images, 20k classes



Deng et al. “Imagenet: a large scale hierarchical image database” ‘09

AlexNet

Breakthrough on ImageNet: ~the beginning of deep learning era



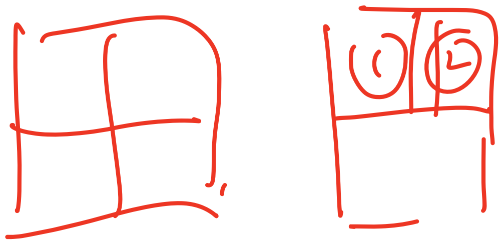
NeurIPS 2022

Krizhevsky, Sutskever, Hinton "ImageNet Claasification with Deep Convolutional Neural Networks", NIPS 2012. Test of time award

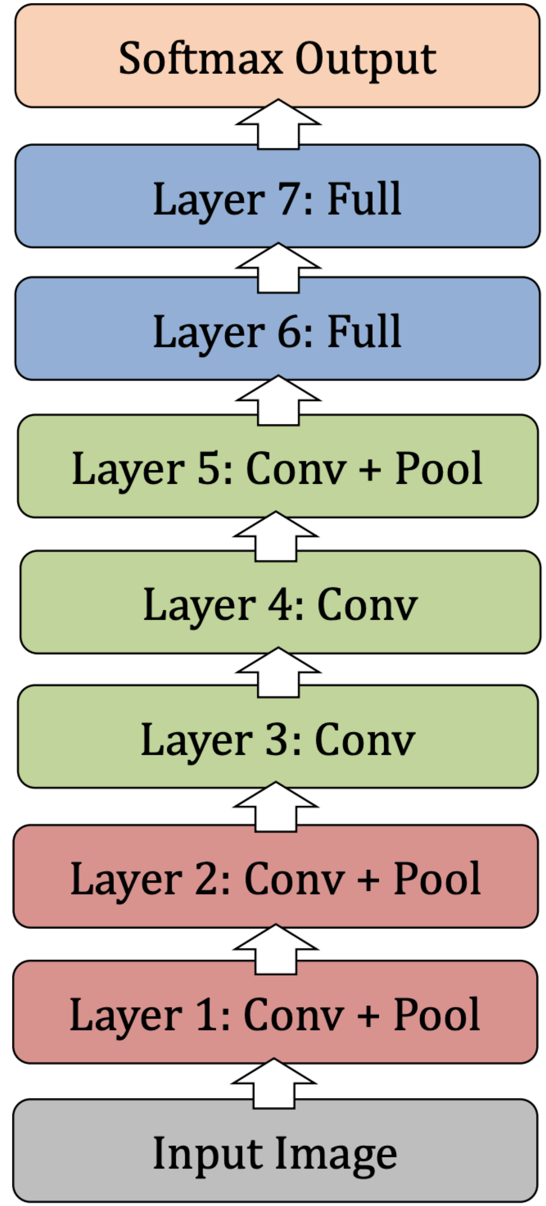
AlexNet

8 layers, ~60M parameters

Top5 error: 18.2%



Techniques used:
ReLU activation, overlapping pooling, dropout, ensemble (create 10 patches by cropping and average the predictions), data-augmentation (intensity of RGB channels)



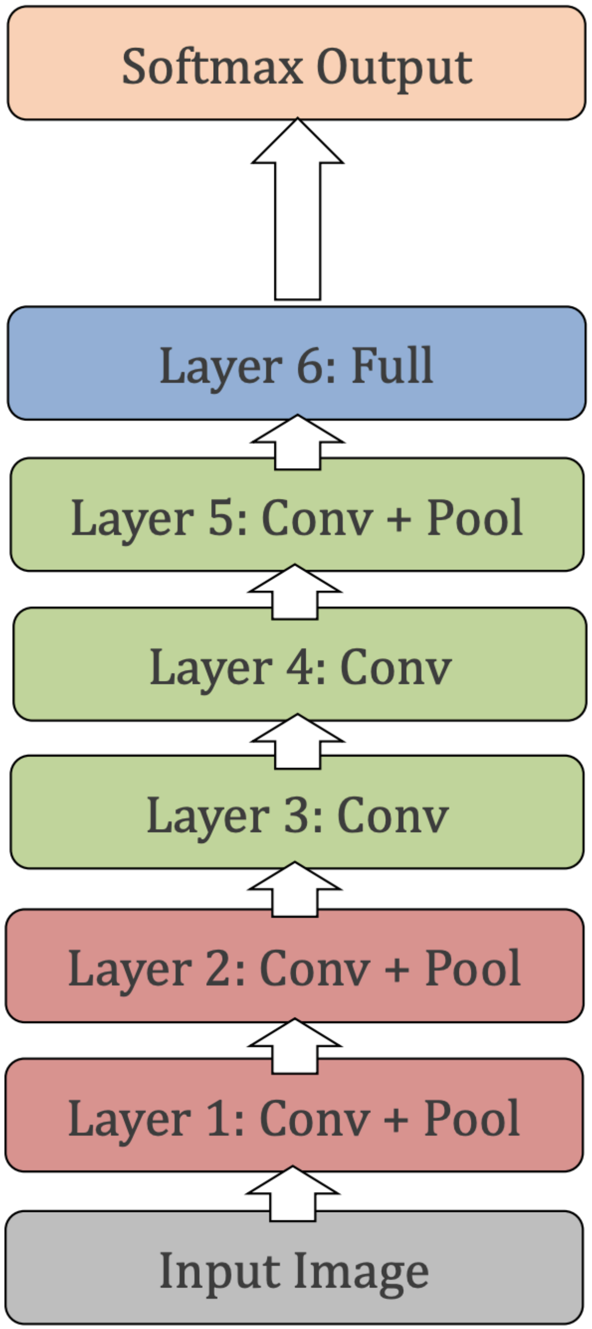
[From Rob Fergus' CIFAR 2016 tutorial]

AlexNet

Remove top fully-connected layer 7

Drop ~16 million parameters

1.1% drop in performance



[From Rob Fergus' CIFAR 2016 tutorial]

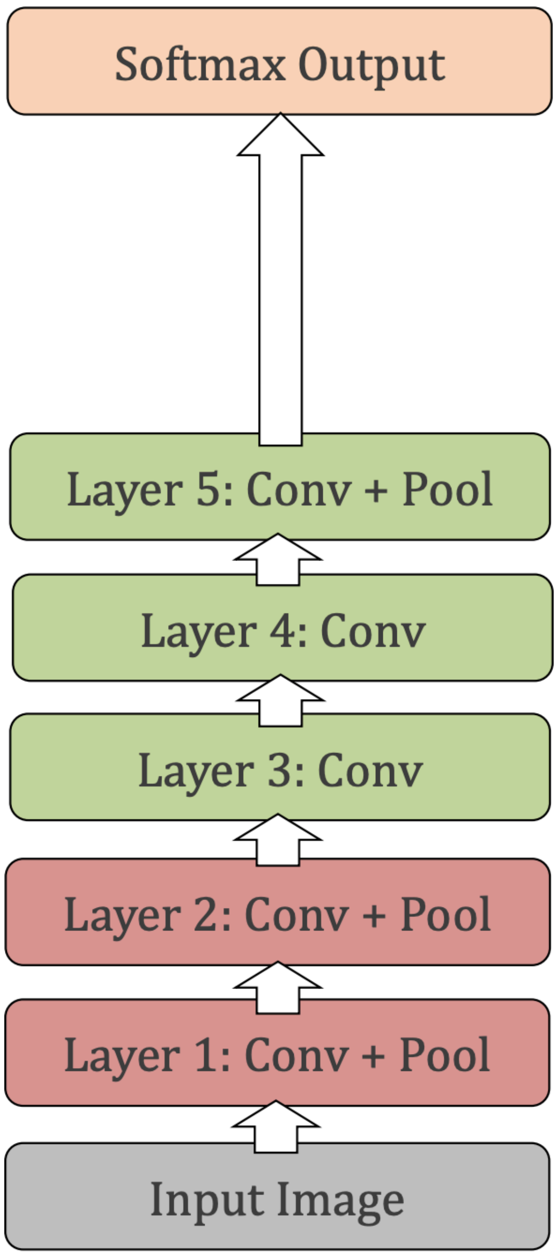
AlexNet

Remove both fully connected layers 6 and 7

Drop ~50 million parameters

5.7% drop in performance

[From Rob Fergus' CIFAR 2016 tutorial]

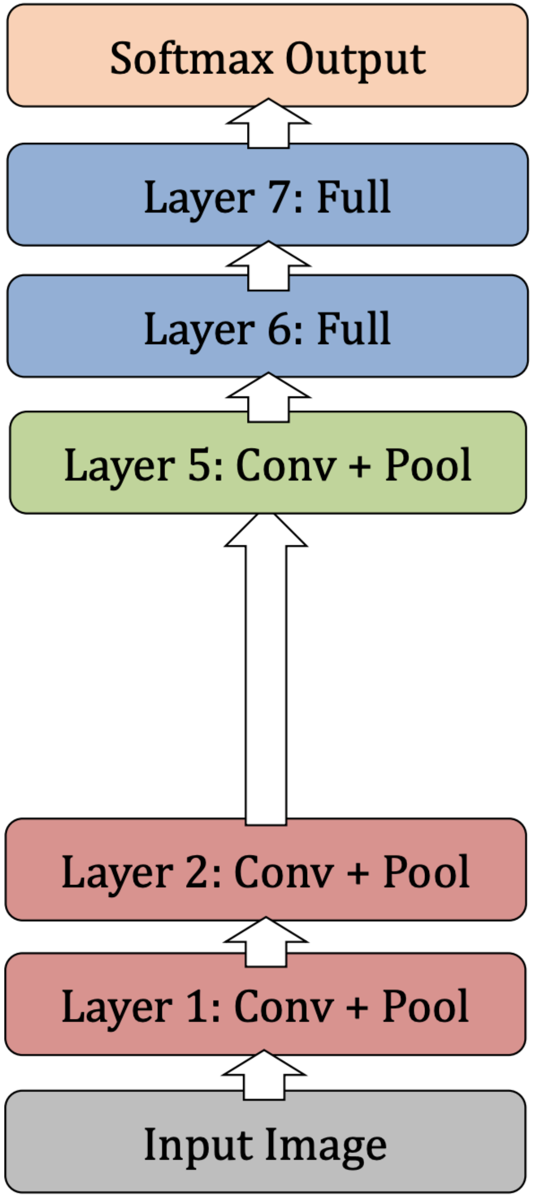


AlexNet

Remove upper convolutio / feature extractor layers (layer 3 and 4)

Drop ~1 million parameters

3% drop in performance



[From Rob Fergus' CIFAR 2016 tutorial]

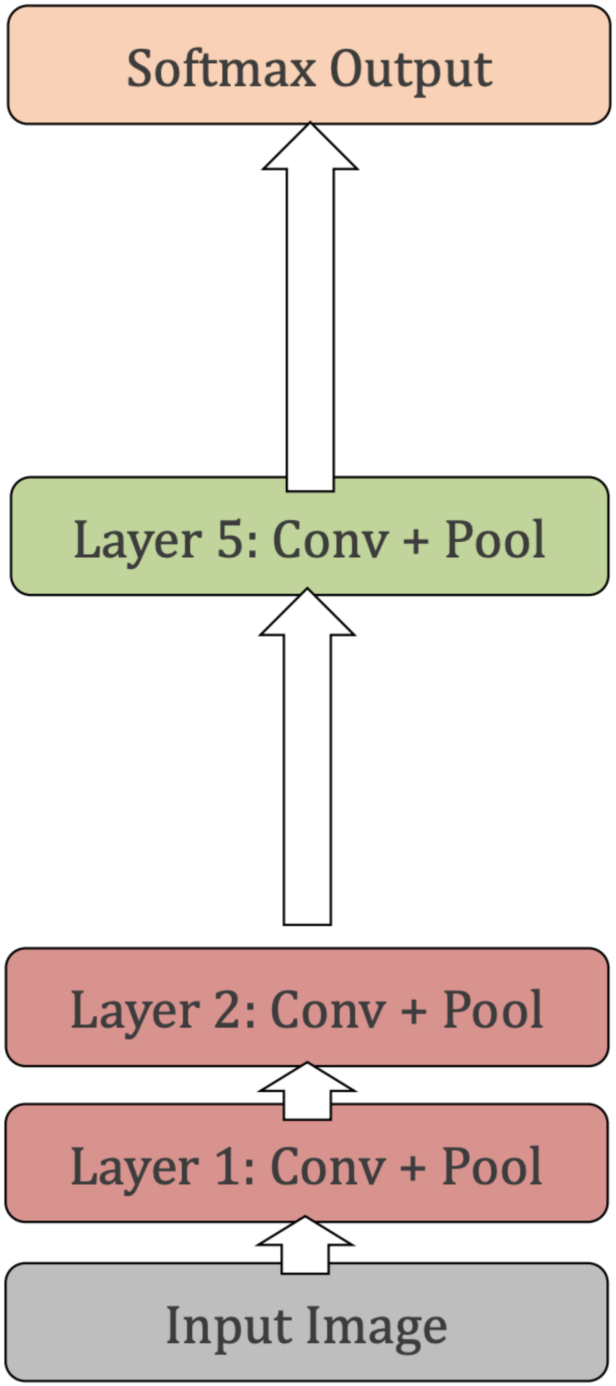
AlexNet

Remove top fully connected layer 6,7 and upper convolution layers 3,4.

33.5% drop in performance.

Depth of the network is the key.

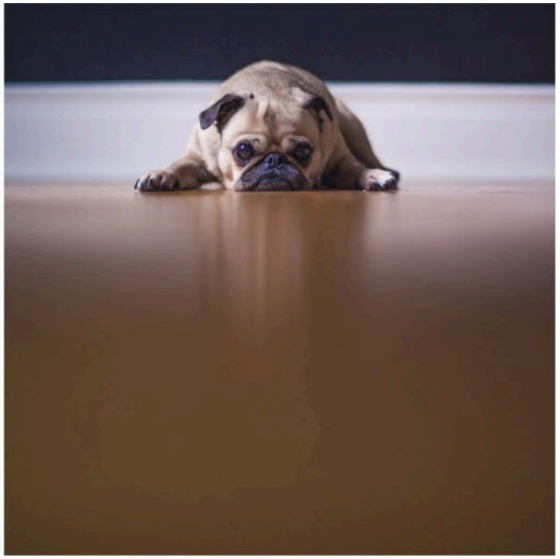
[From Rob Fergus' CIFAR 2016 tutorial]



GoogLeNet

Motivation: multiscale nature of images





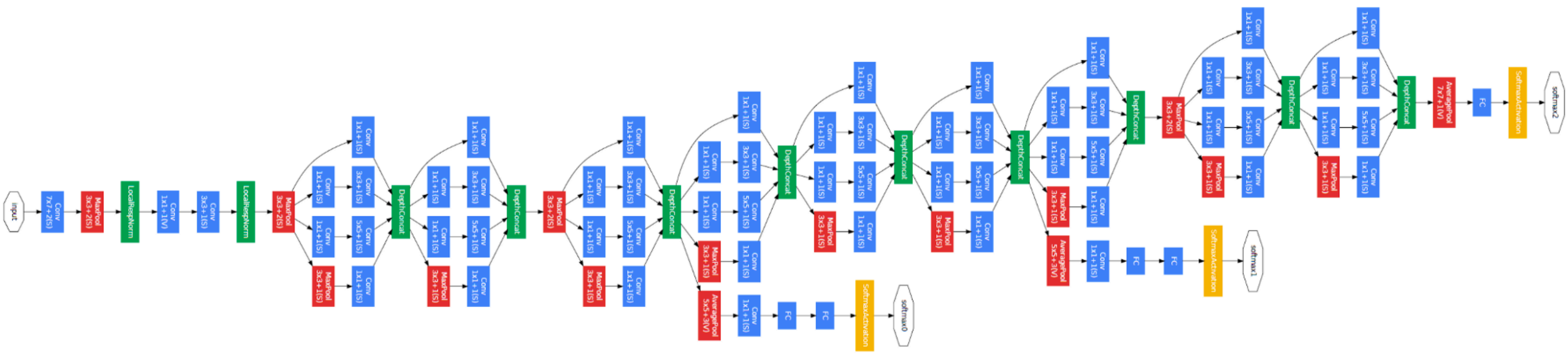
5X5 1X1

Large kernel for global features, and **smaller kernel** for local features.

Idea: have multiple different-size kernels at any layer.

[Going Deep with Convolutions, Szegedy et al. '14]

GoogLeNet

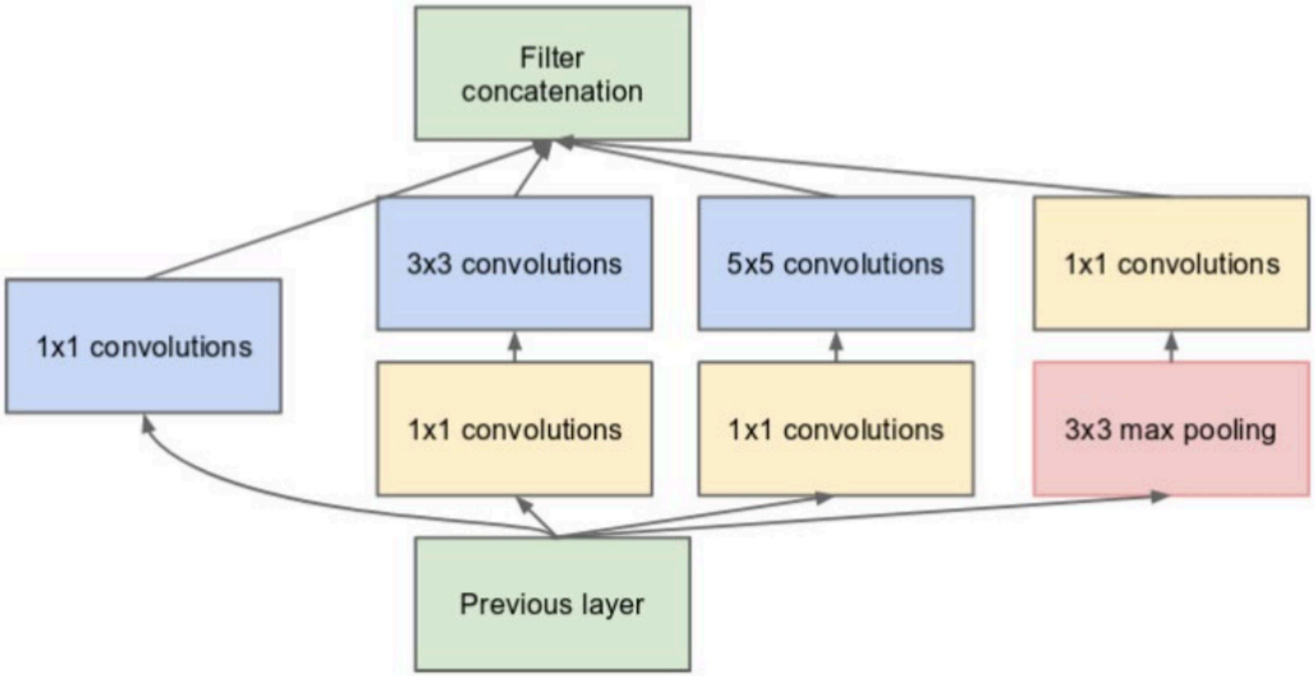


Large kernel for global features, and **smaller kernel** for local features.

Idea: have multiple different-size kernels at any layer.

[Going Deep with Convolutions, Szegedy et al. ’14]

Inception Module



Multiple filter scales at each layer

Dimensionality reduction to keep computational requirements down

[Going Deep with Convolutions, Szegedy et al. ’14]

Residual Networks

Motivation: extremely deep nets are hard to train (gradient explosion/ vanishing)

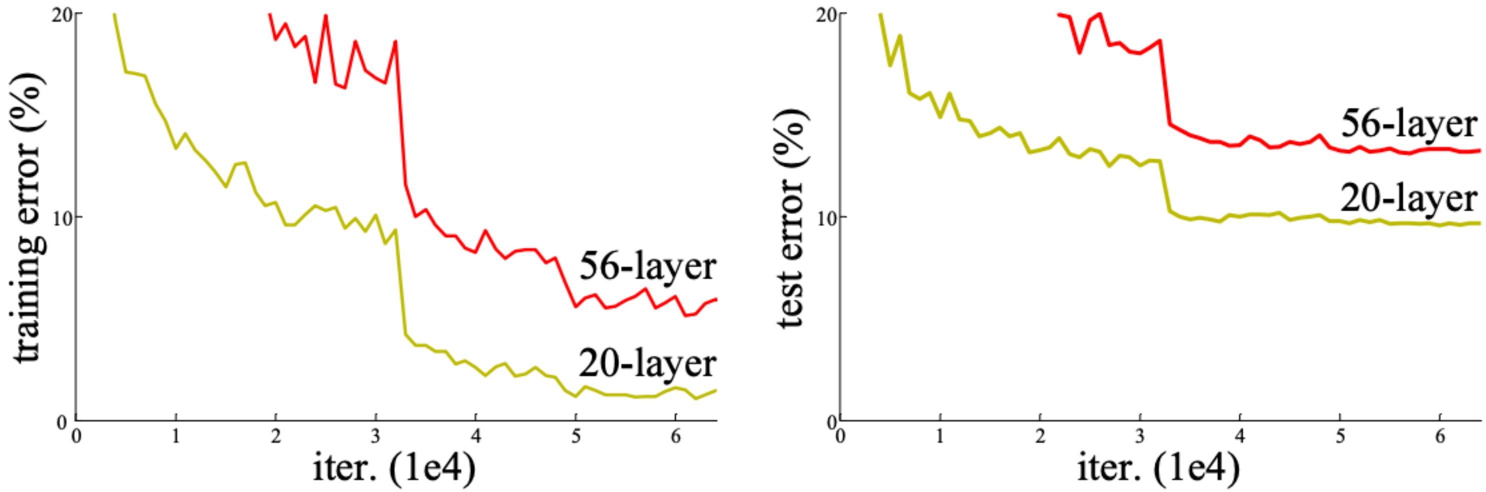


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

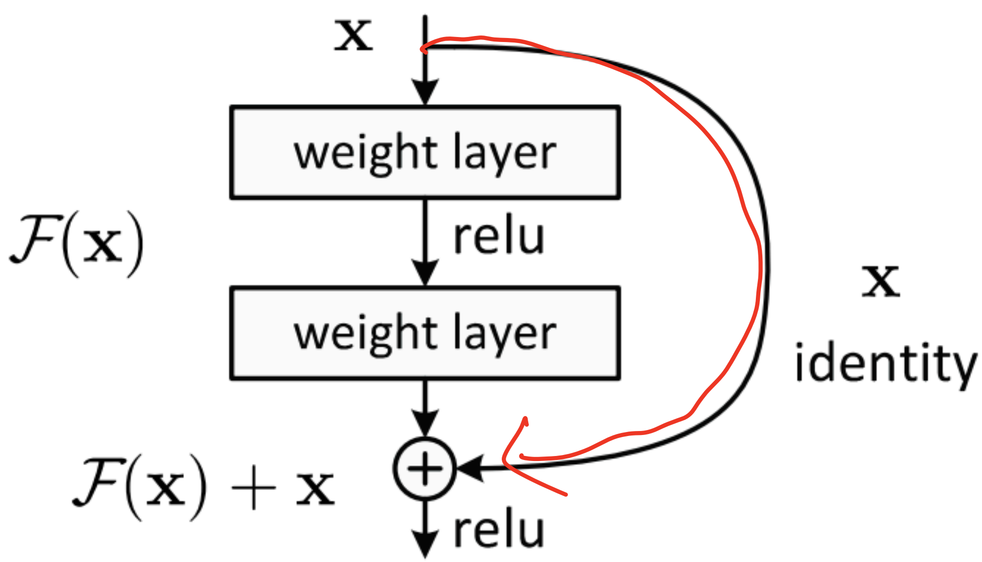
[He, Zhang, Ren, Sun, '16]

Residual Networks

Idea: identity shortcut, skip one or more layers.

Justification: network can easily simulate shallow network ($F \approx 0$), so performance should not degrade by going deeper.

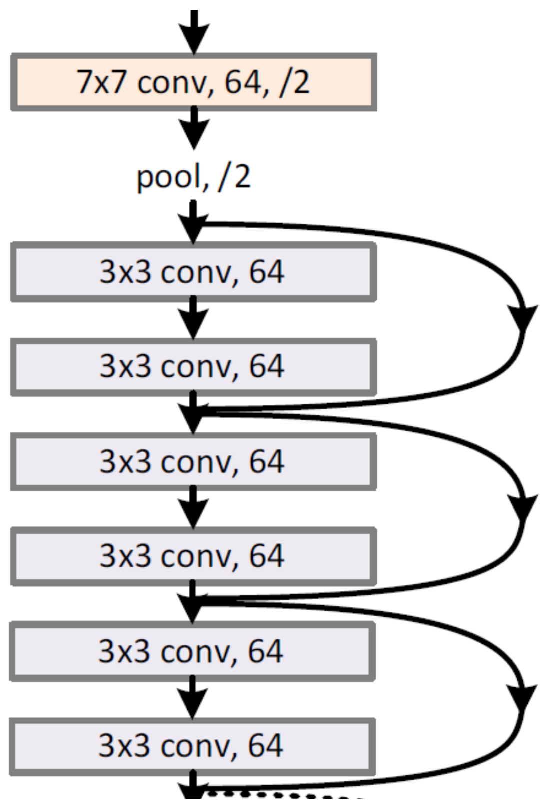
one block

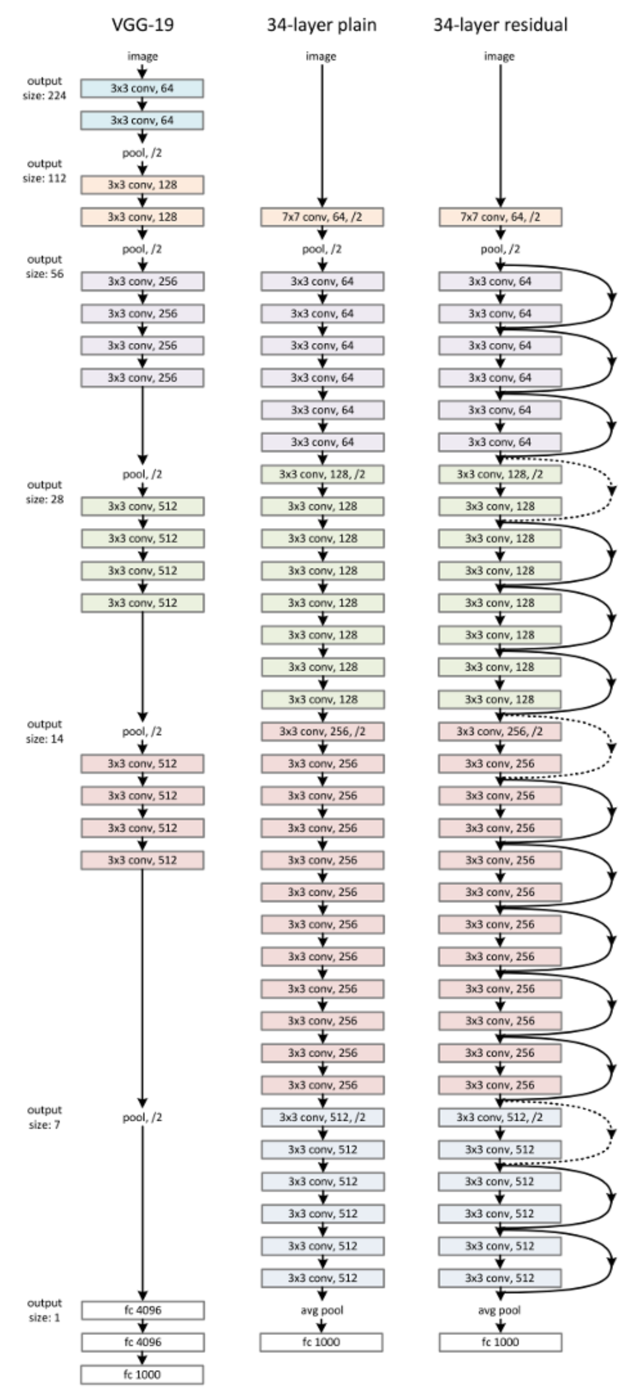


[He, Zhang, Ren, Sun, '16]

Residual Networks

- 3.57% top-5 error on ImageNet
- First deep network with > 100 layers.
- Widely used in many domains (AlphaGo)

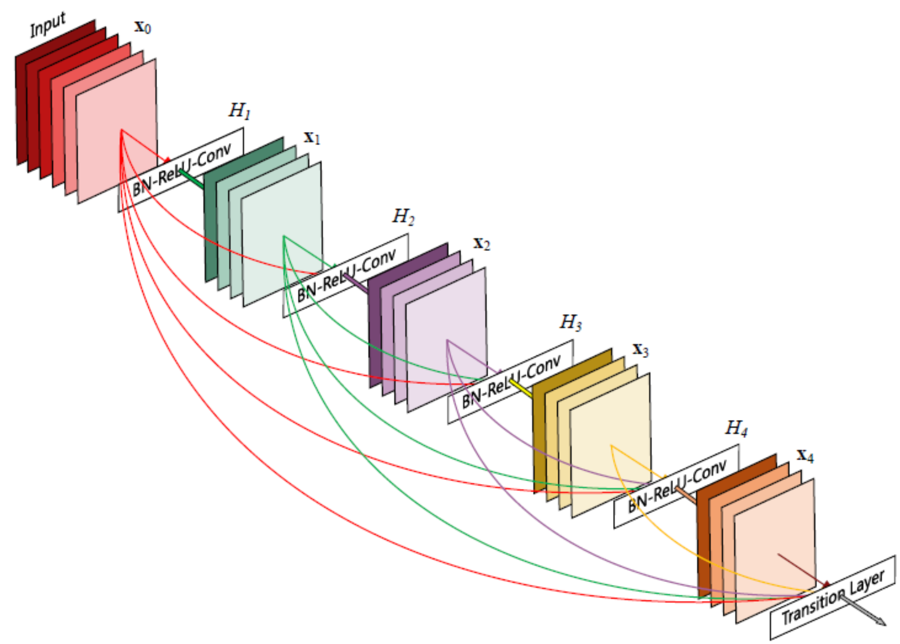




[He, Zhang, Ren, Sun, '16]

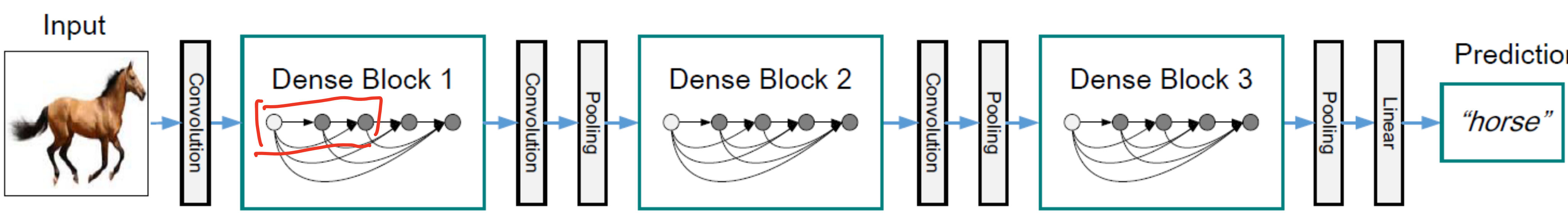
Densely Connected Network

Idea: explicit forward output of layer to all future layers (by concatenation)



Intuition: helps vanishing gradients, encourage reuse features (reduce parameter count)

Issues: network maybe too wide, need to be careful about memory consumption



[He, Zhang, Ren, Sun, '16]

Neural Architecture / Hyper-Parameter Search

Many design choices:
- Number of layers, width, kernel size, pooling, connections, etc.
- Normalization, learning rate, batch size, etc.

Strategies:
- Grid search
- Random search [Bergestra & Bengio '12]
- Bandit-based [Li et al. '16]
- Gradient-based (DARTS) [Liu et al. '19]
- Neural tangent kernel [Xu et al. '21]
- …

NN archi $\longrightarrow$ Kernel

n : # of para $n' \ll n$