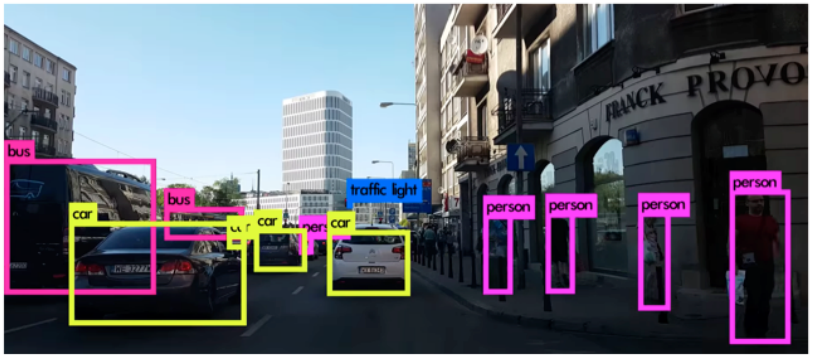
Introduction to Deep Learning and Adversarial Robustness

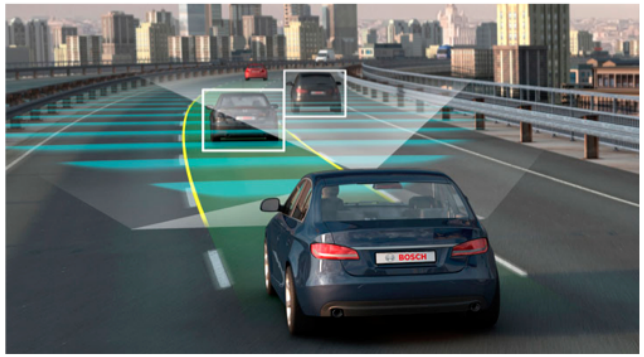
Yao Li
Department of Statistics and Operations Research
UNC Chapel Hill

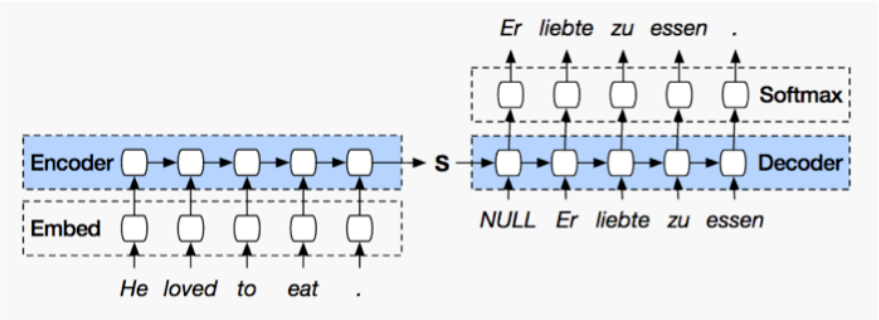
Instructor

- Name: Yao Li
- Personal Website: `https://liyao880.github.io/yaoli/`
- Research interest: security of machine learning system, large-scale recommendation systems, model compression

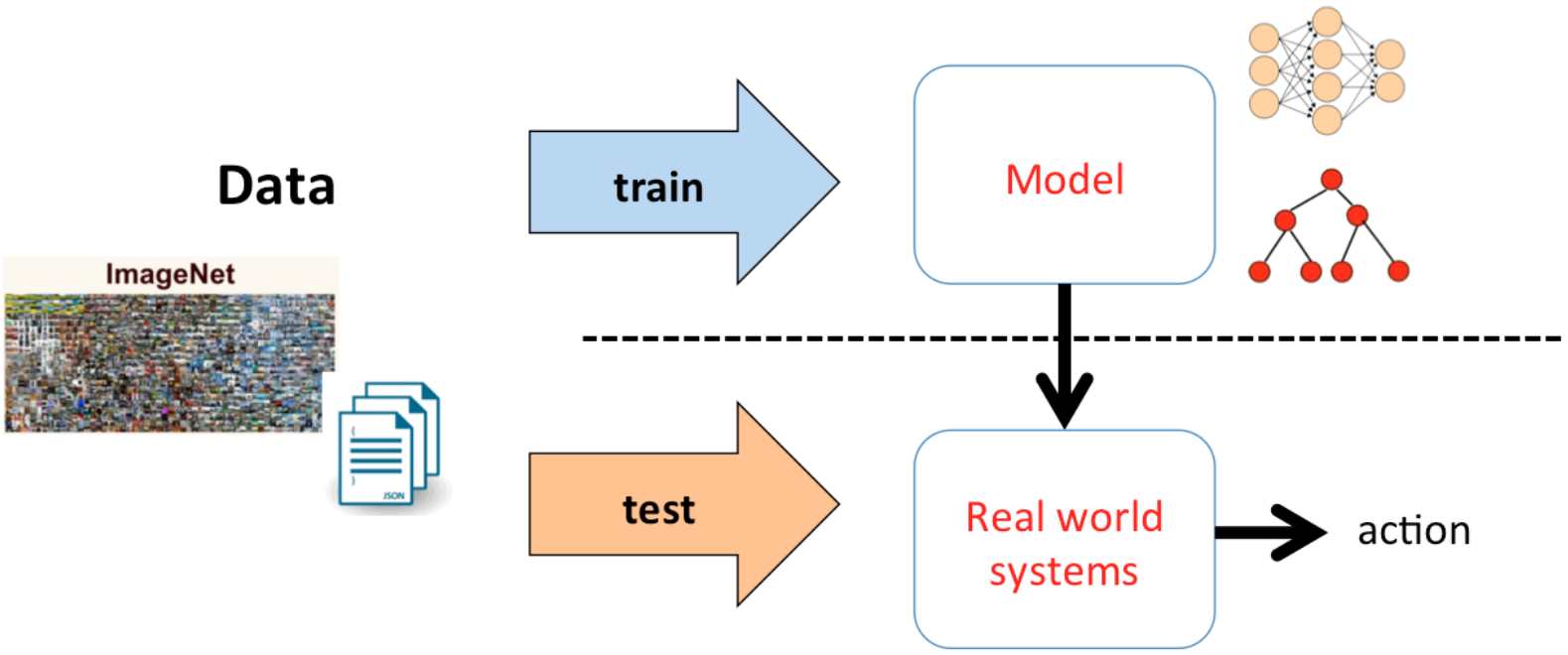
Machine Learning



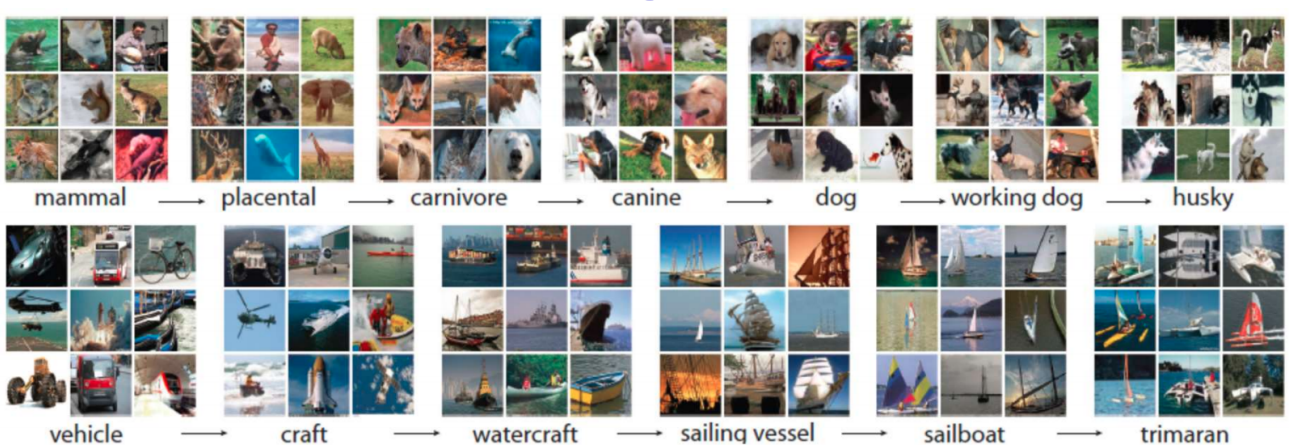




Machine Learning Systems

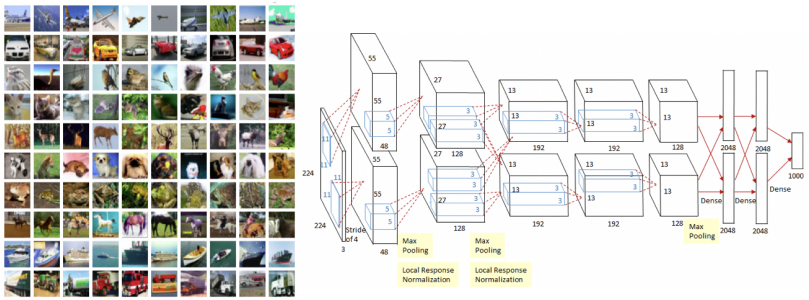


ImageNet Data



- ILSVRC competition: 1000 classes
  - training: about 1.0 million images
  - validation: 50,000 images
  - testing: 150,000 images
- Full imagenet: $> 20,000$ categories, each with about a thousand images.

Deep Neural Network



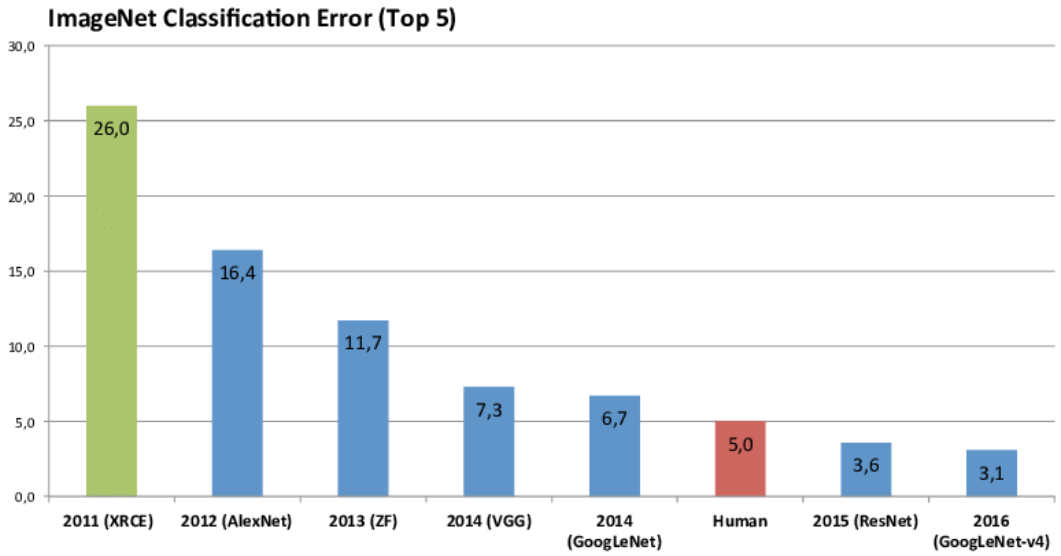


Figure: Winner results of the ImageNet large scale visual recognition challenge (LSVRC) of the past years on the top-5 classification task

Supervised Learning

$$f^*(\boldsymbol{X}, \boldsymbol{w}) = \underset{f}{\operatorname{argmin}}\, E(f, \boldsymbol{y}) = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i, \boldsymbol{w}), y_i),$$

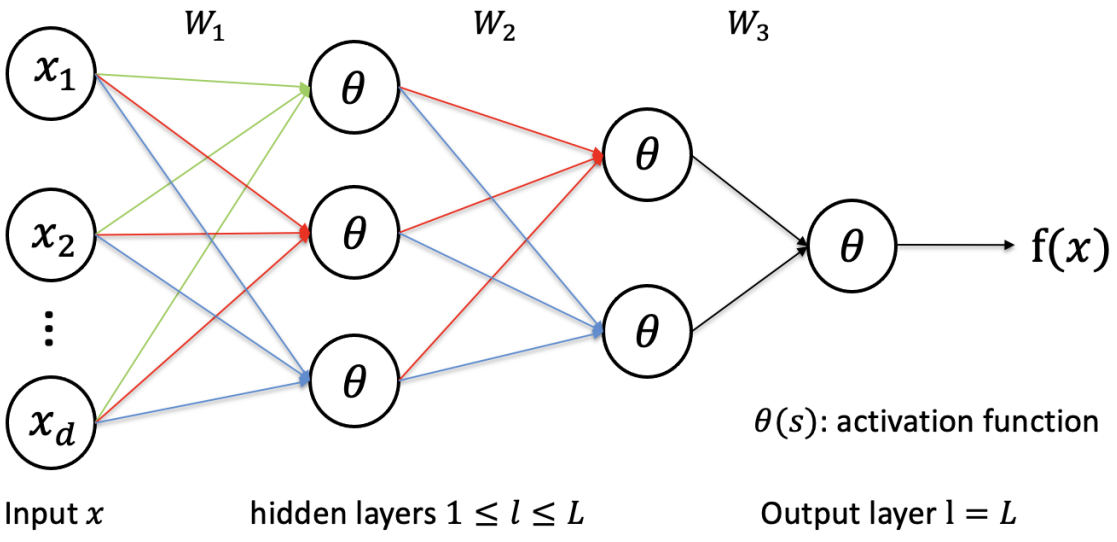
- Training data: $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$, $\boldsymbol{y} = (y_1, ..., y_n)$, $\boldsymbol{x}_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$
- Function to learn: $f : \mathbb{R}^p \to \mathbb{R}$
- Error function: $E(f, \boldsymbol{y})$
- Loss function: $\ell(\cdot, \cdot)$
- Parameters of the function: $\boldsymbol{w}$

Feedforward Neural Network

- Input layer: d neurons (input features)
- Neurons from layer 1 to L: Linear combination of previous layers + activation function

$$\theta(\boldsymbol{w}^T\boldsymbol{x}), \theta : \text{activation function}$$

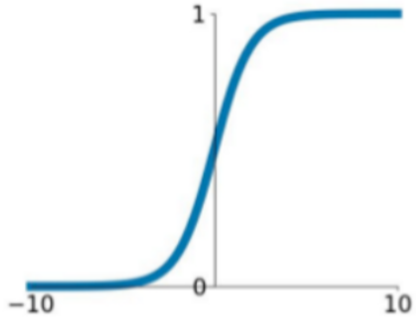
- Final layer: one neuron $\longrightarrow$ output $f(\boldsymbol{x}) = \theta(W_3\theta(W_2\theta(W_1\boldsymbol{x})))$
- All layers are: Fully connected layers



Activation Function

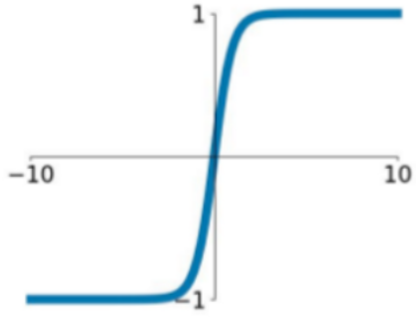
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



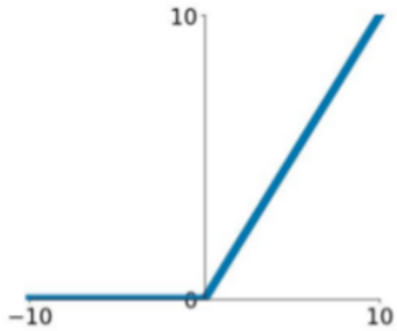
tanh

$$\tanh(x)$$

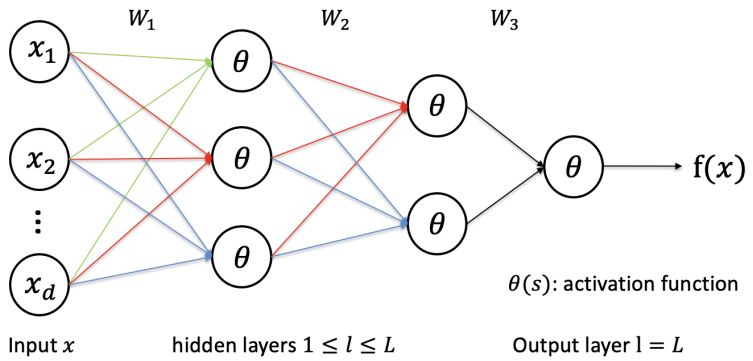


ReLU

$$\max(0, x)$$



Example: Forward Pass Computation



- Input data: $\boldsymbol{x} = (1.5, -1.0, 1.3)^T$, $y = 0.5$
- Activation: ReLU ($\theta(x) = \max(0, x)$)
- Weights:

$$W_1 = \begin{pmatrix} 0.3 & 0.4 & 0.2 \\ 0.3 & 0.5 & 0.2 \\ 0.8 & 1.0 & -1.0 \end{pmatrix}, W_2 = \begin{pmatrix} 0 & -1.2 & 0.5 \\ 0.9 & 1.0 & 0 \end{pmatrix}$$

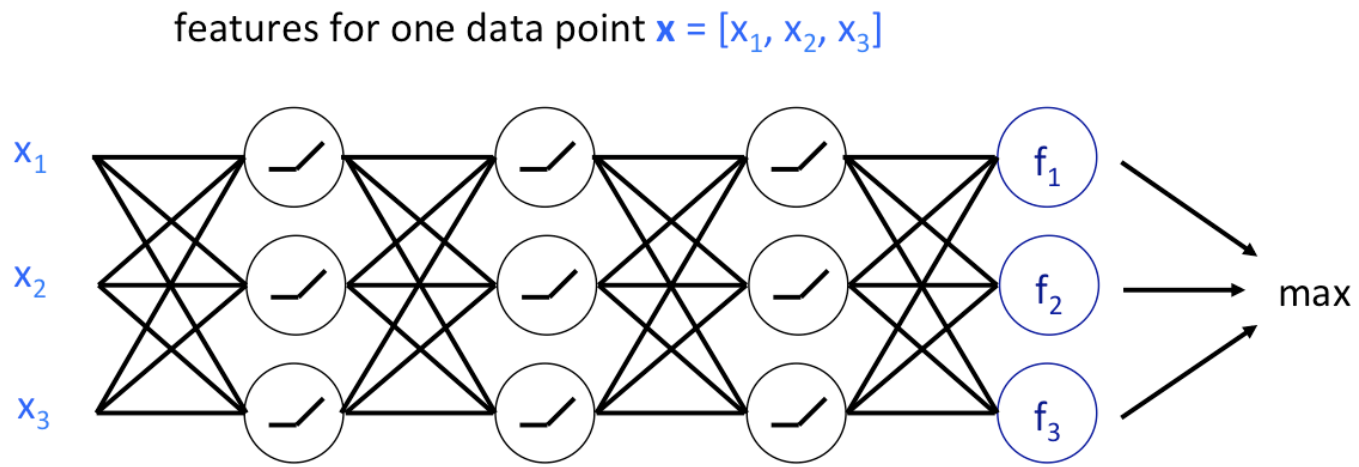
$$W_3 = (-1.0, 1.0)$$

- Please compute $f(\boldsymbol{x})$.
- Reminder: $f(\boldsymbol{x}) = \theta(W_3\theta(W_2\theta(W_1\boldsymbol{x})))$

Multiclass Classification

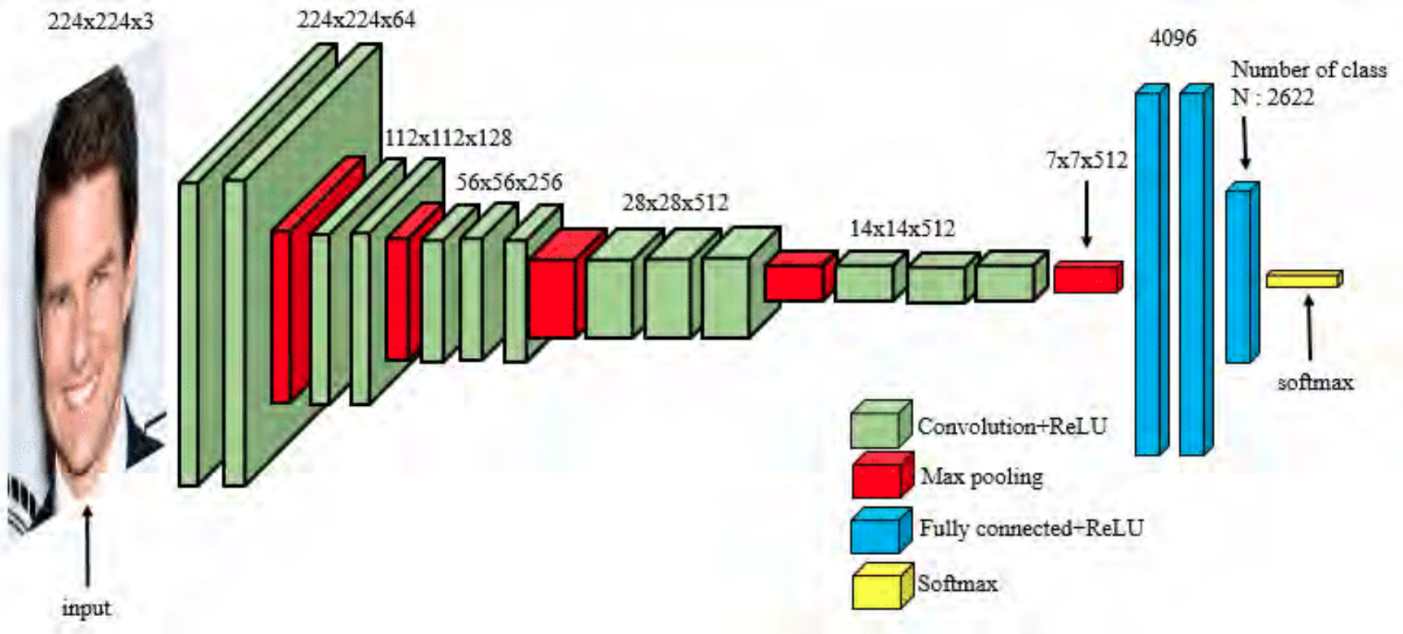
- K classes: K neurons in the final layer
- Output of each f_i is the score of class i

Taking $\text{argmax}_i f_i(\boldsymbol{x})$ as the prediction



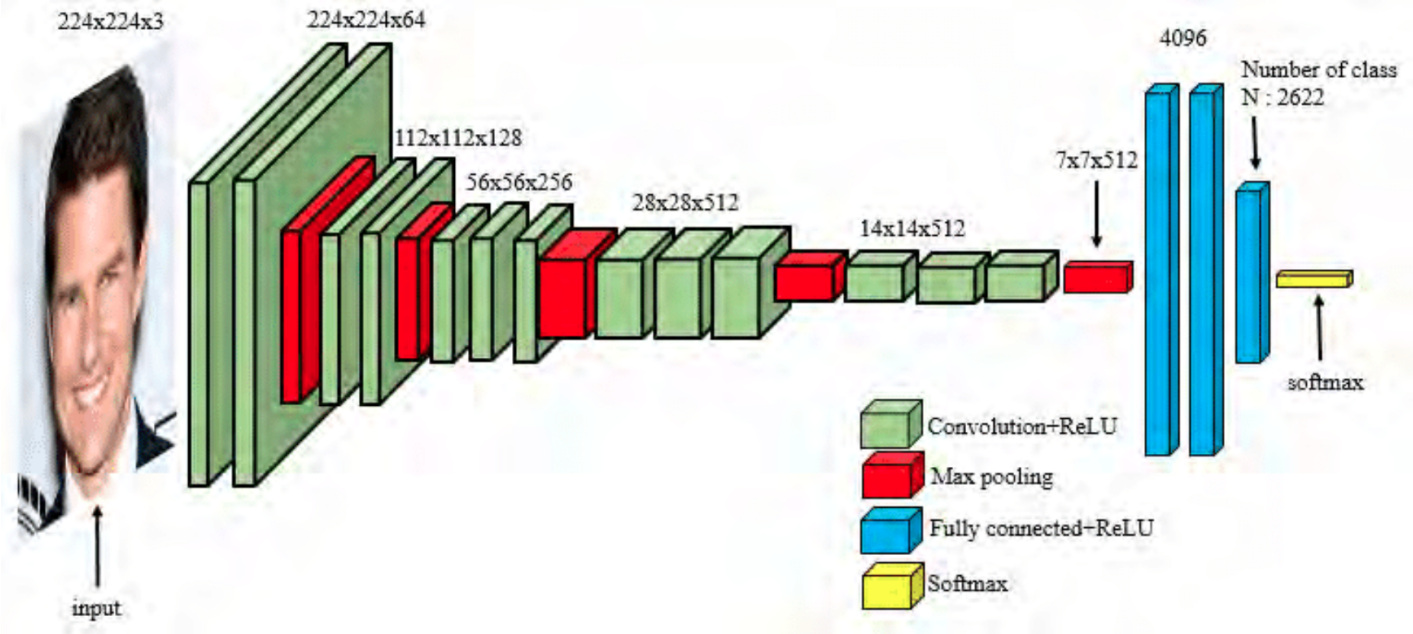
Convolutional Neural Network

- Structure of VGG



Convolutional Neural Network

- Structure of VGG



- Two important layers:
  - Convolution
  - Pooling

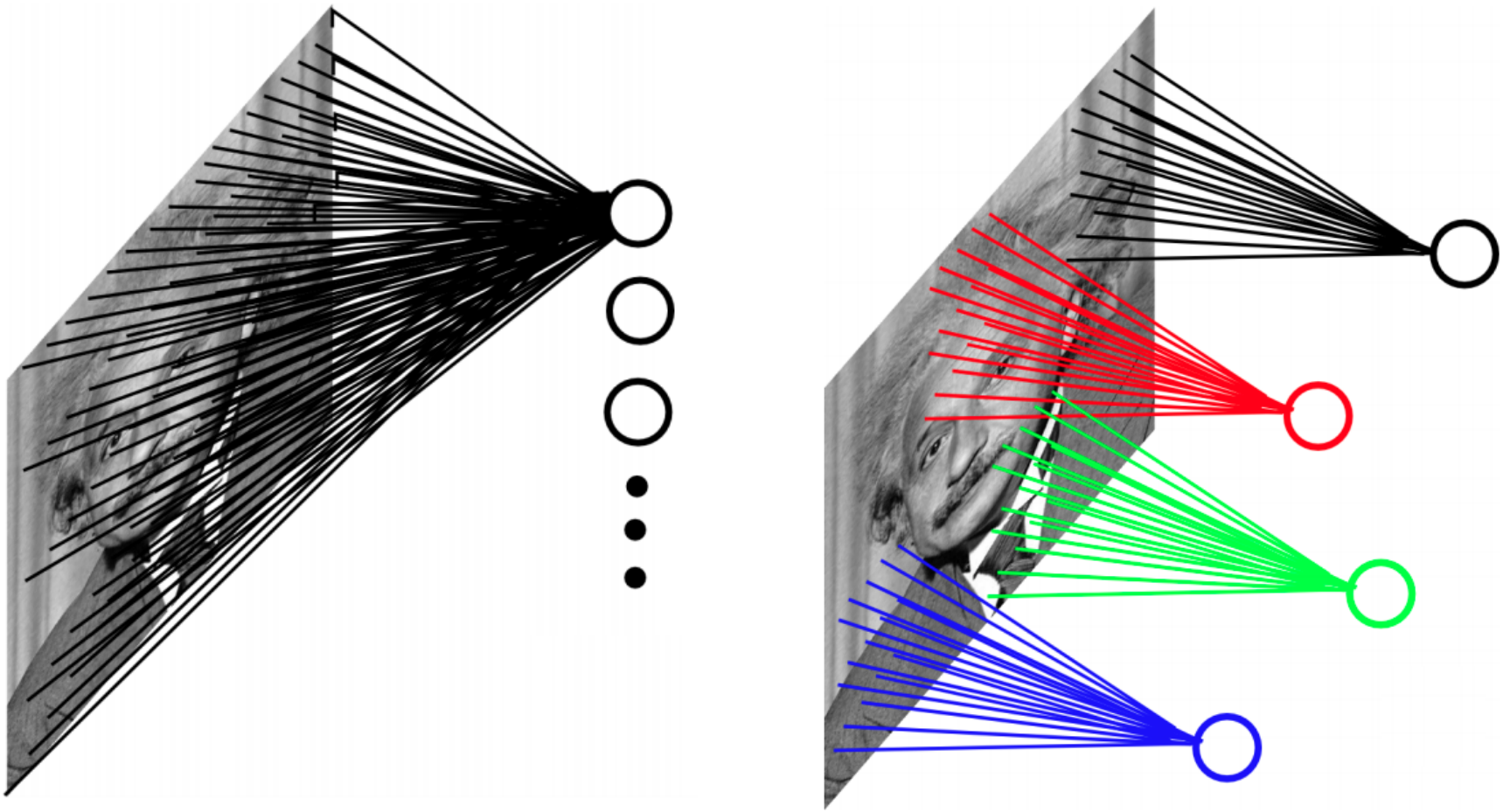
Convolution Layer

- Fully connected layers have too many parameters

  $\Rightarrow$ poor performance
- Example: VGG first layer
  - Input: $224 \times 224 \times 3$
  - Output: $224 \times 224 \times 64$
  - Number of parameters: $(224 \times 224 \times 3) \times (224 \times 224 \times 64) = 483$ billion
- Convolution layer leads to:
  - Local connectivity
  - Parameter sharing

Local connectivity

- Each hidden unit is connected only to a sub-region of input



(Figure from Salakhutdinov 2017)

Parameter Sharing

- Making one reasonable assumption:

If one feature is useful to compute at some spatial position (x, y), then it should also be useful to compute at a different position (x_2, y_2)

- Using the convolution operator

Convolution

- The convolution of an image x with a kernel k is computed as

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{p,q}$$

1	0.5	20
0.25	0	0
0	0	20

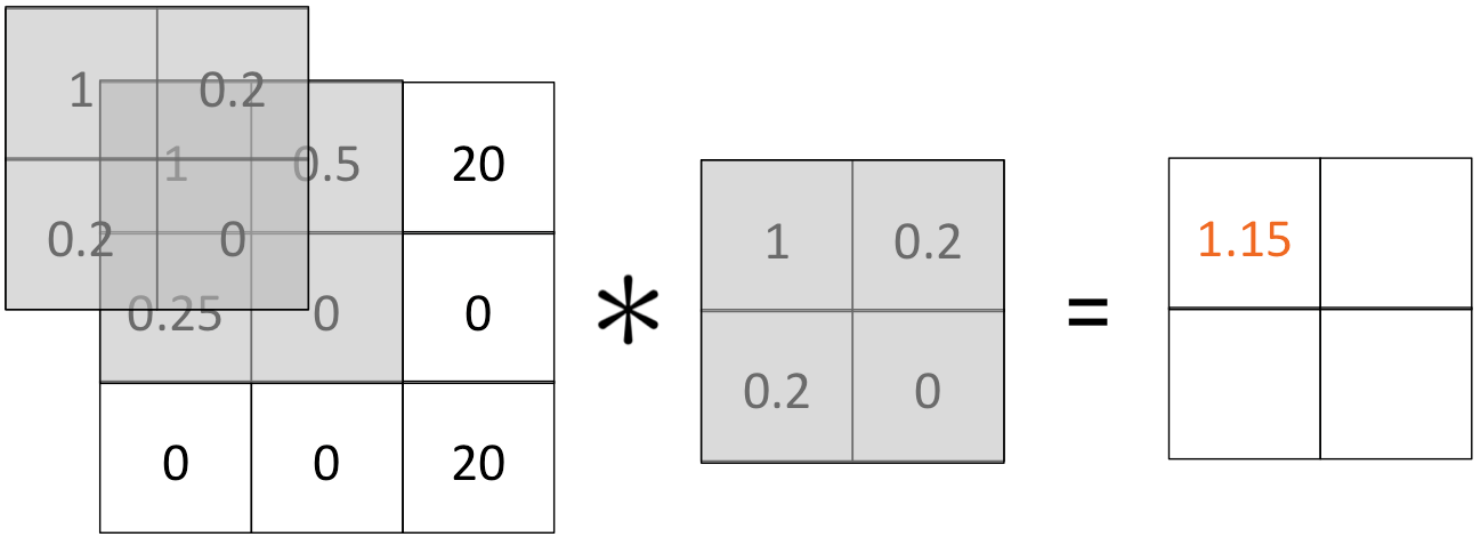
$*$

1	0.5
0.25	0

$=$

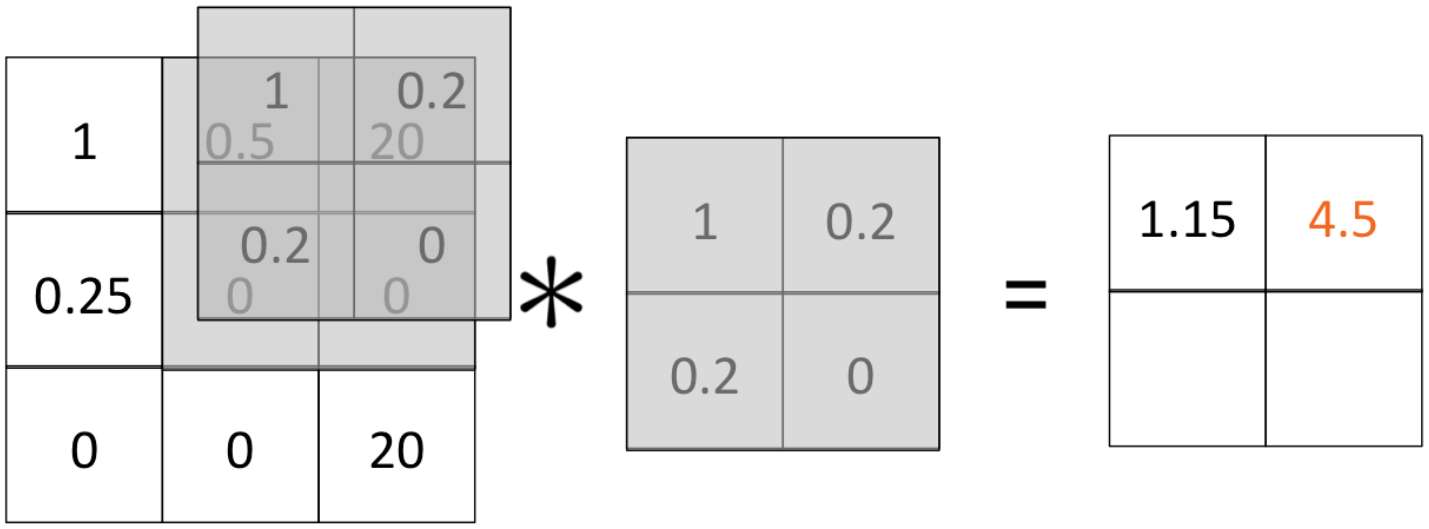
Convolution Process

1*1 + 0.5*0.2 + 0.25*0.2 + 0*0 = 1.15



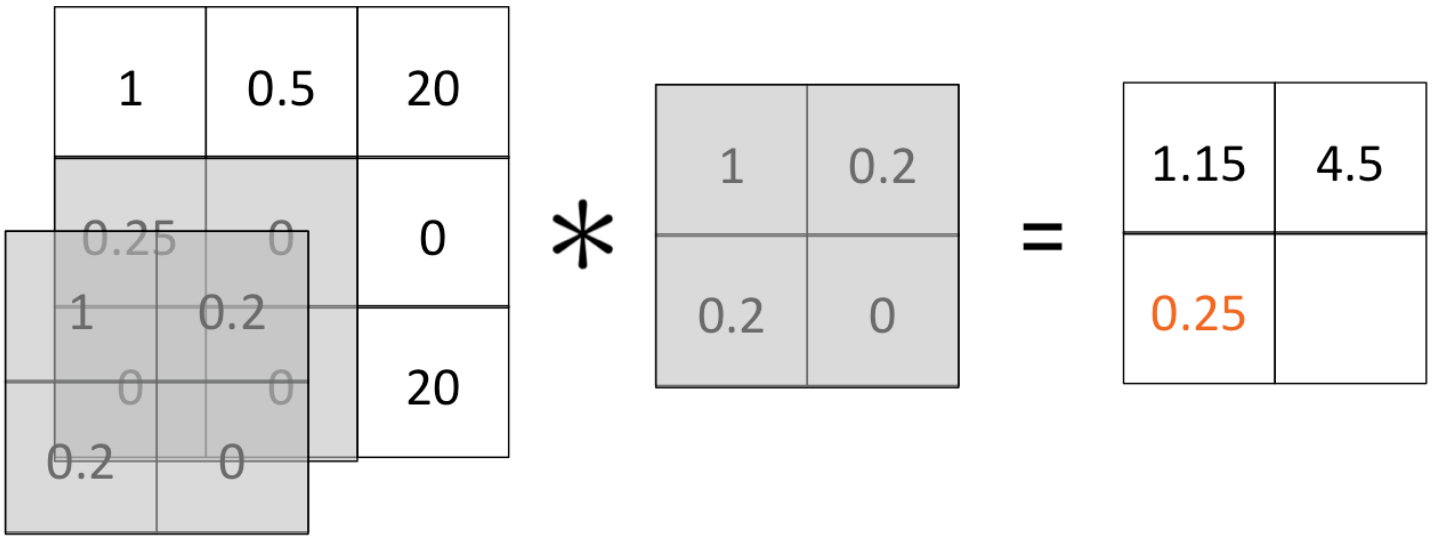
Convolution Process

0.5*1 + 20*0.2 + 0*0.2 + 0*0 = 4.5



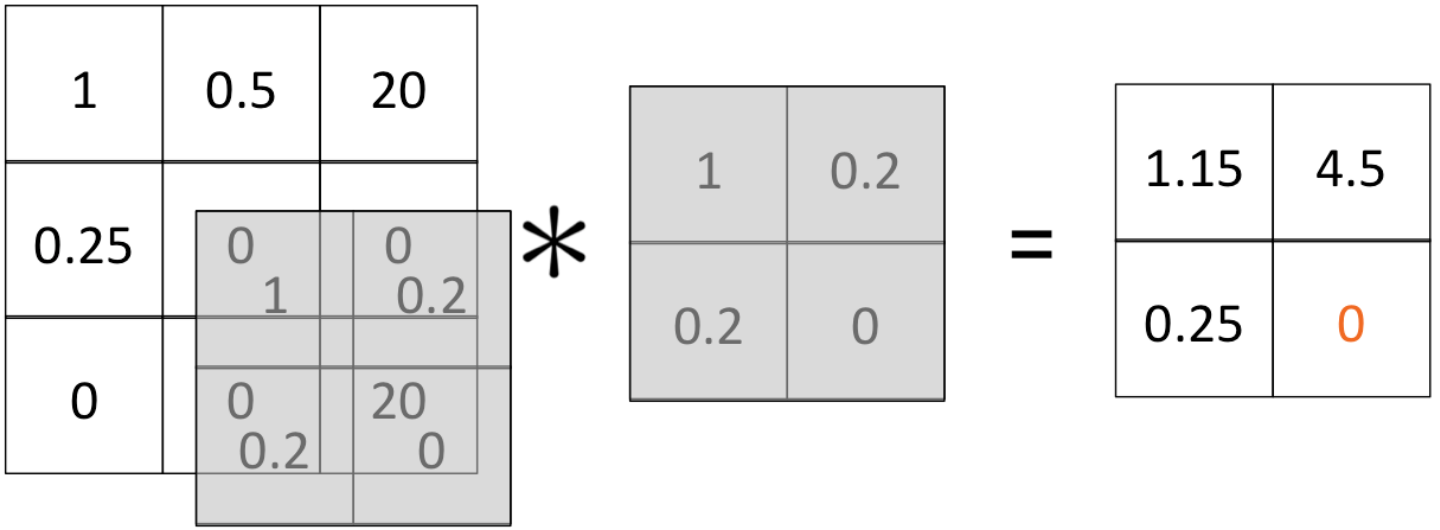
Convolution Process

0.25*1 + 0*0.2 + 0*0.2 + 0*0 = 0.25

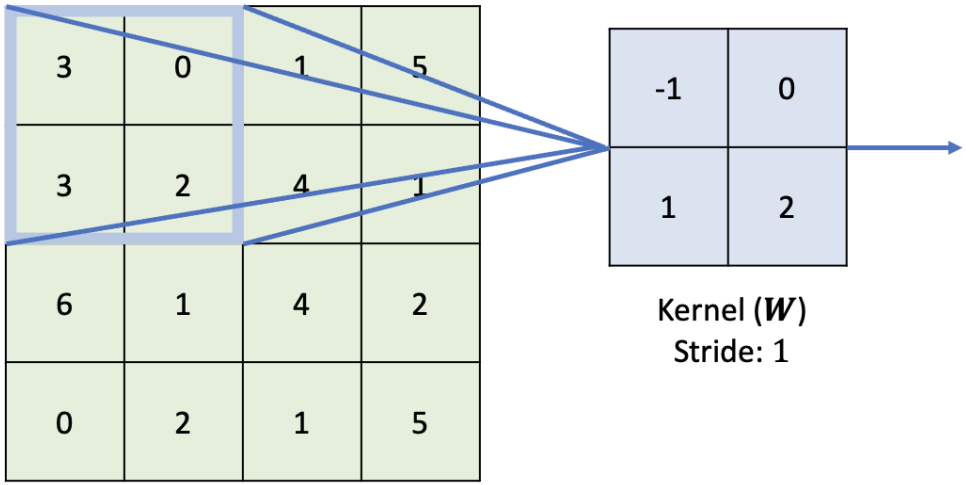


Convolution Process

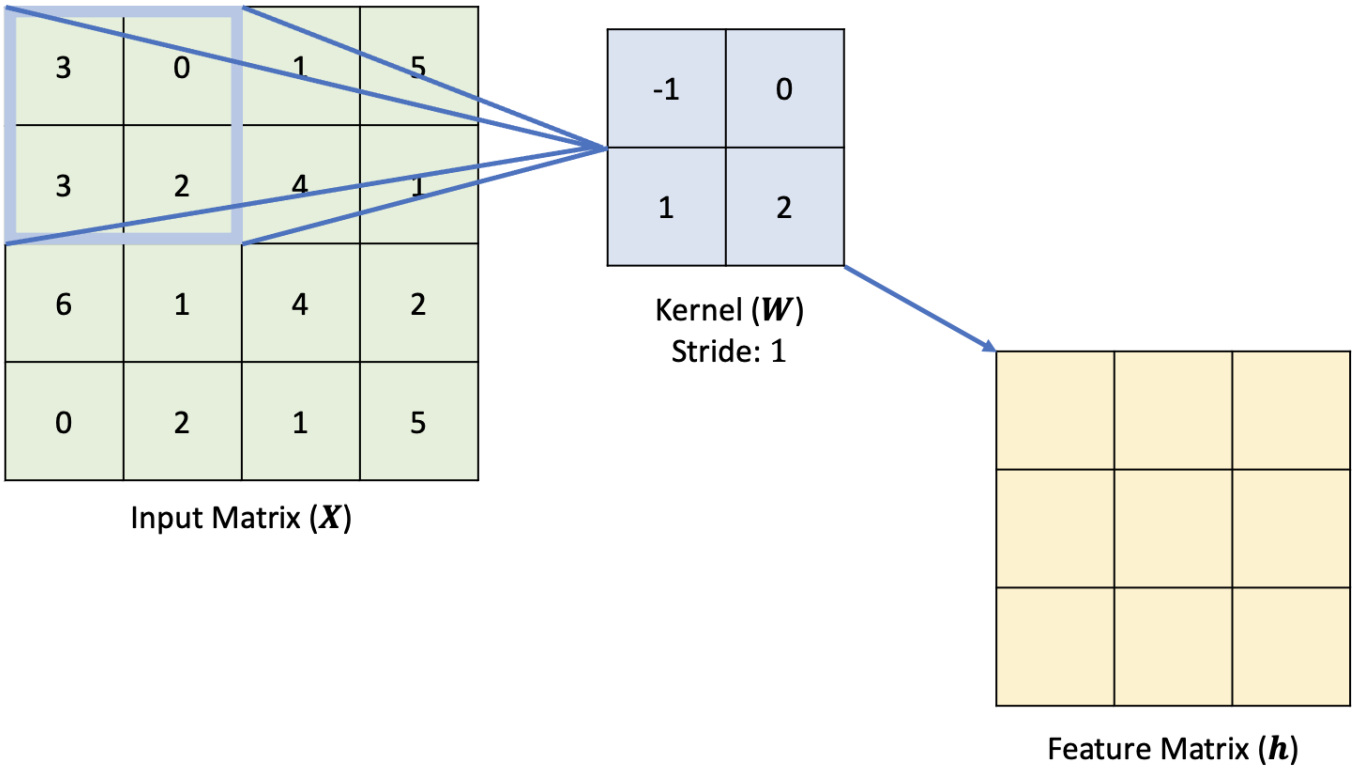
$$0*1 + 0*0.2 + 0*0.2 + 20*0 = 0$$



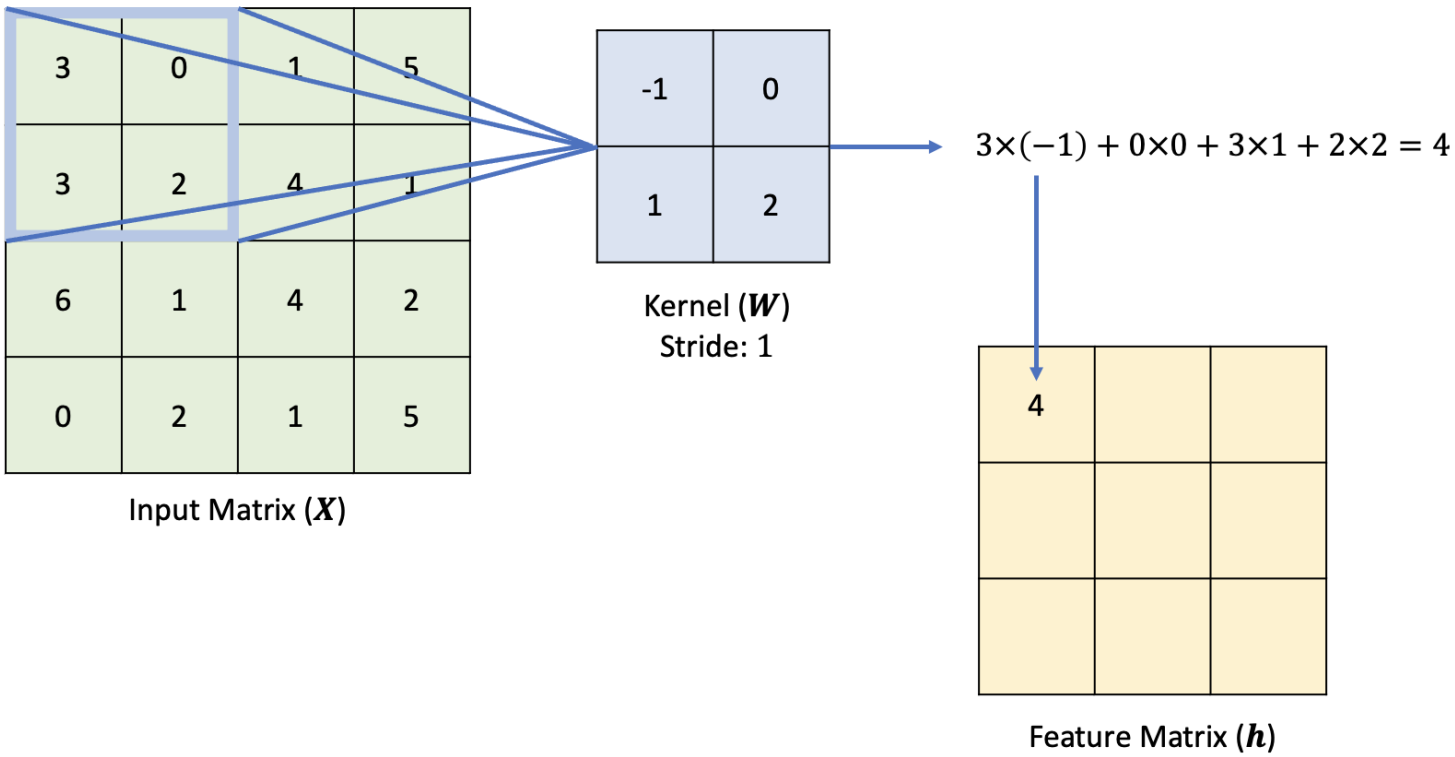
Example: Forward Pass Convolution



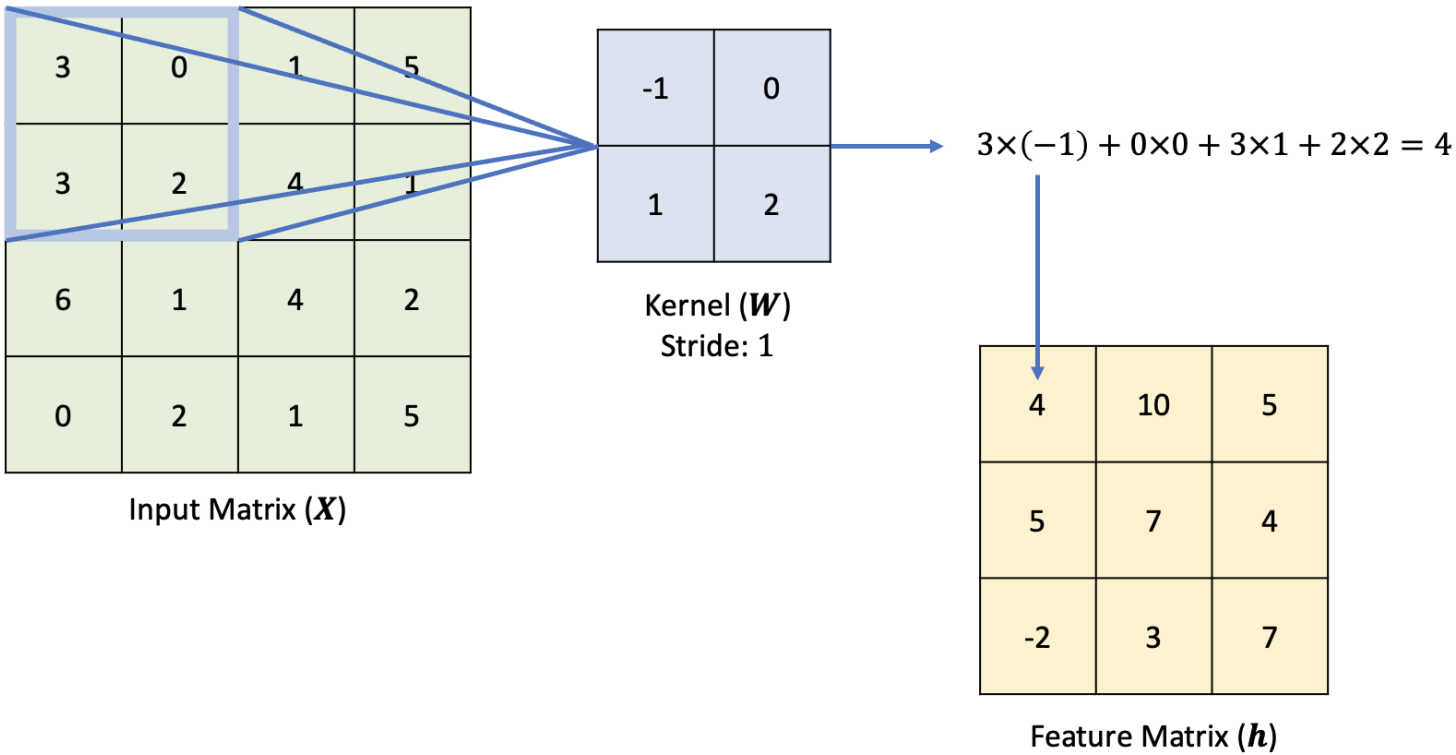
Example: Forward Pass Convolution



Example: Forward Pass Convolution

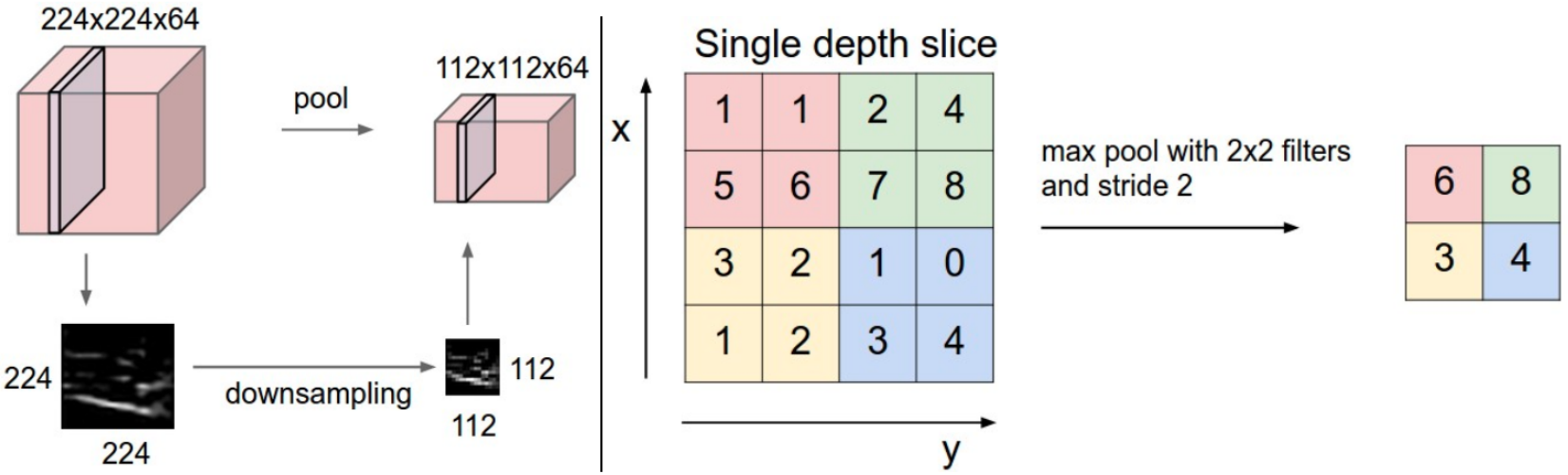


Example: Forward Pass Convolution

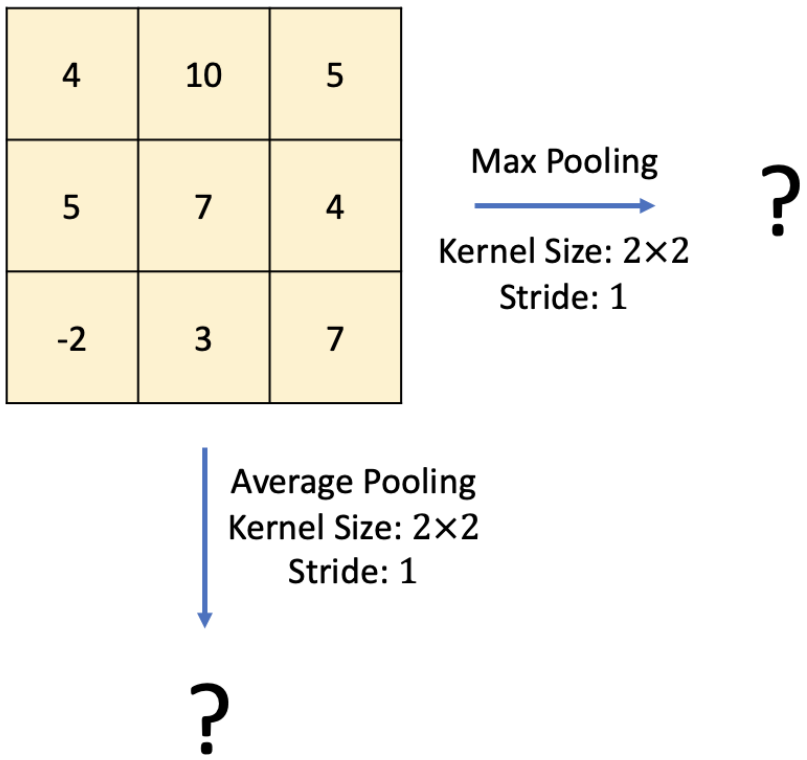


Pooling

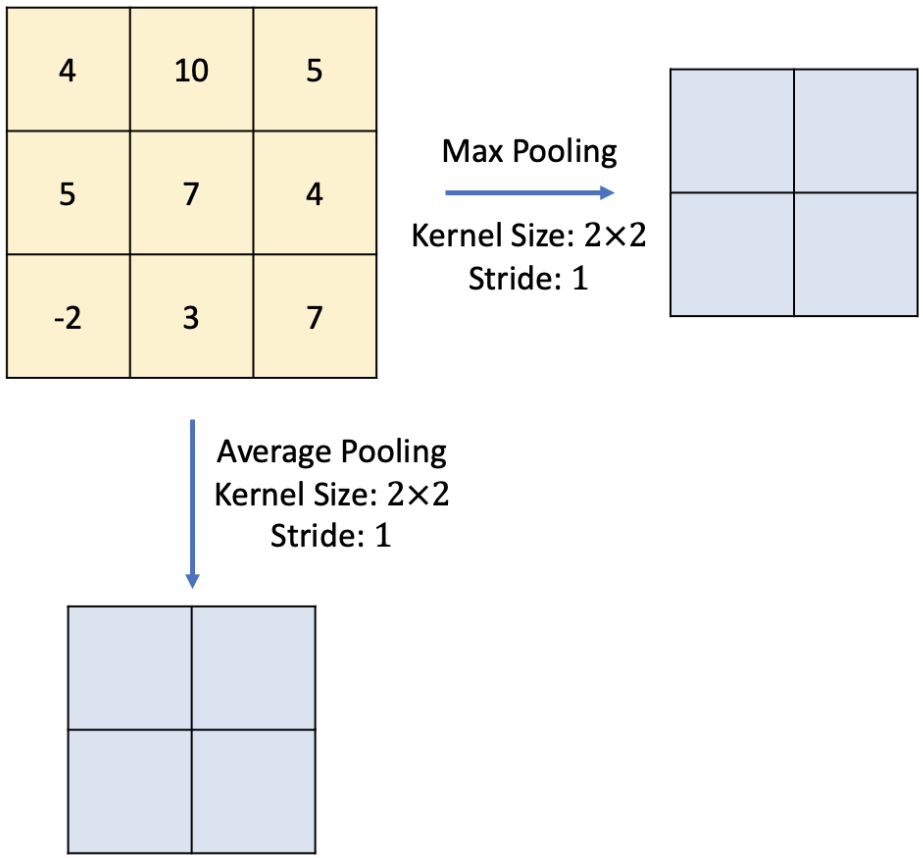
- Reduce the size of representation, down-sampling
- Example: Max Pooling



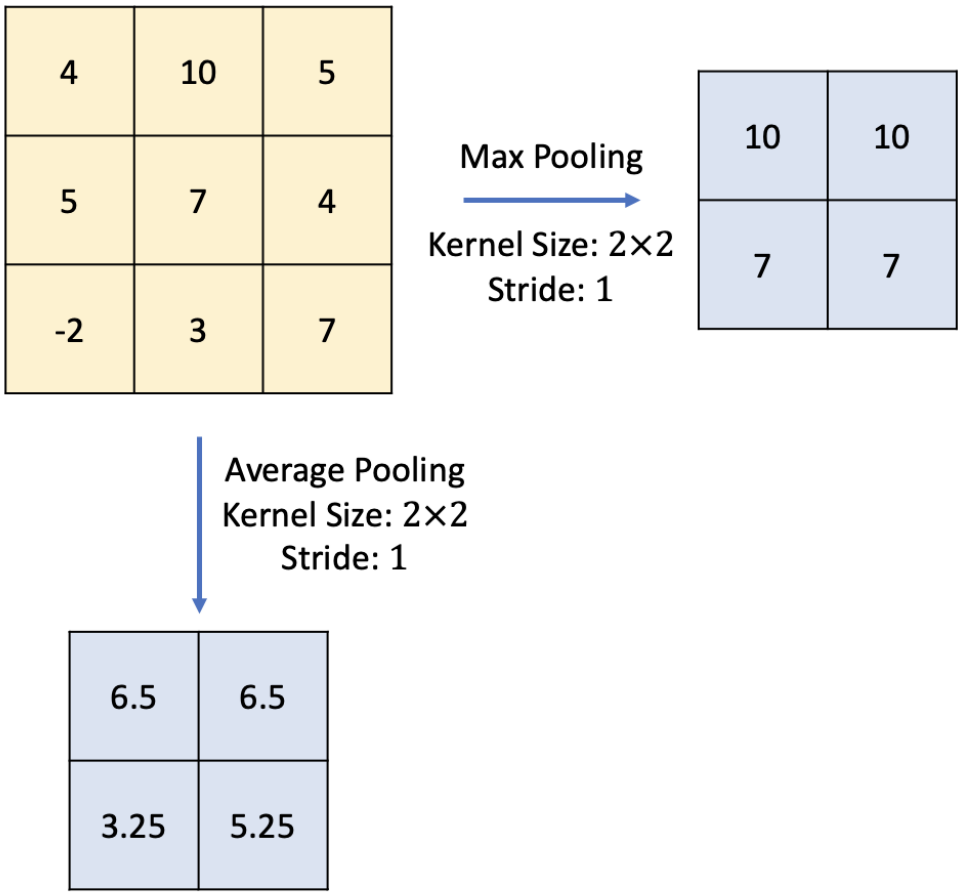
Example: Forward Pass Pooling



Example: Forward Pass Pooling



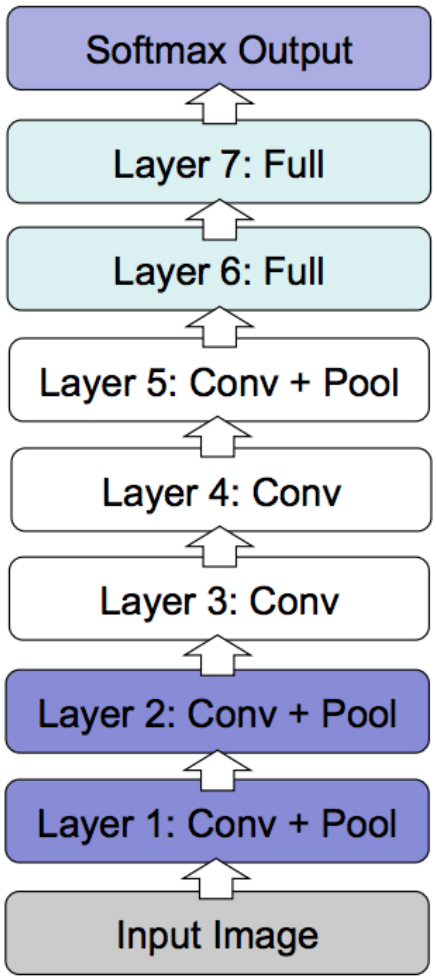
Example: Forward Pass Pooling



CNN: AlexNet

- 8 layers in total, about 60 million parameters and 650,000 neurons.
- Trained on ImageNet dataset
- 18.2% top-5 error (validation set)

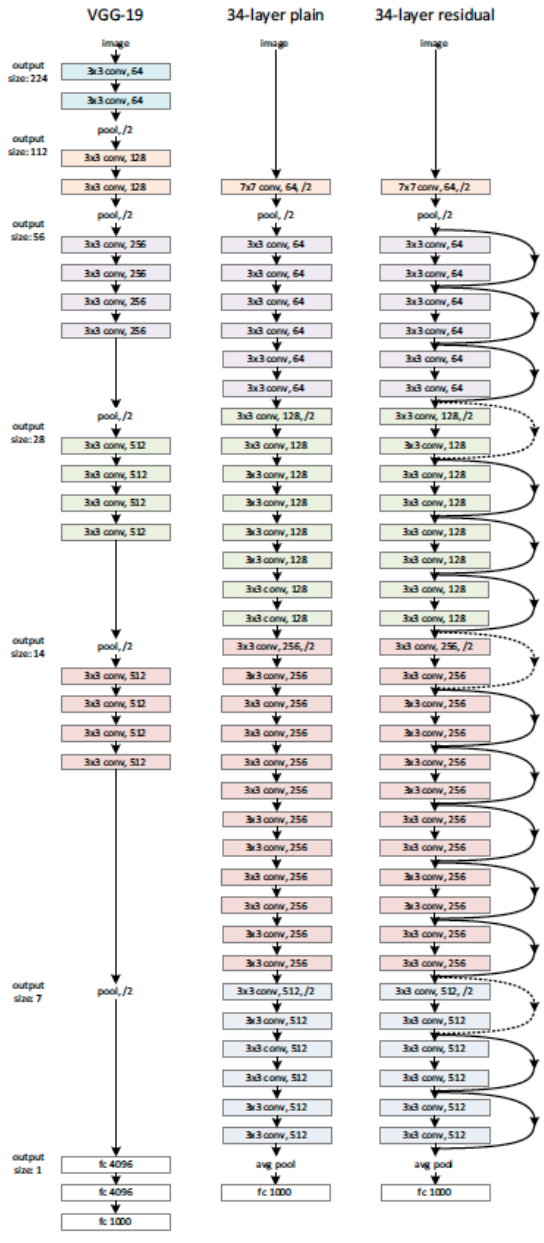
  "ImageNet Classification with Deep Convolutional Neural Networks", by Krizhevsky, Sustskever and Hinton, NIPS 2012.



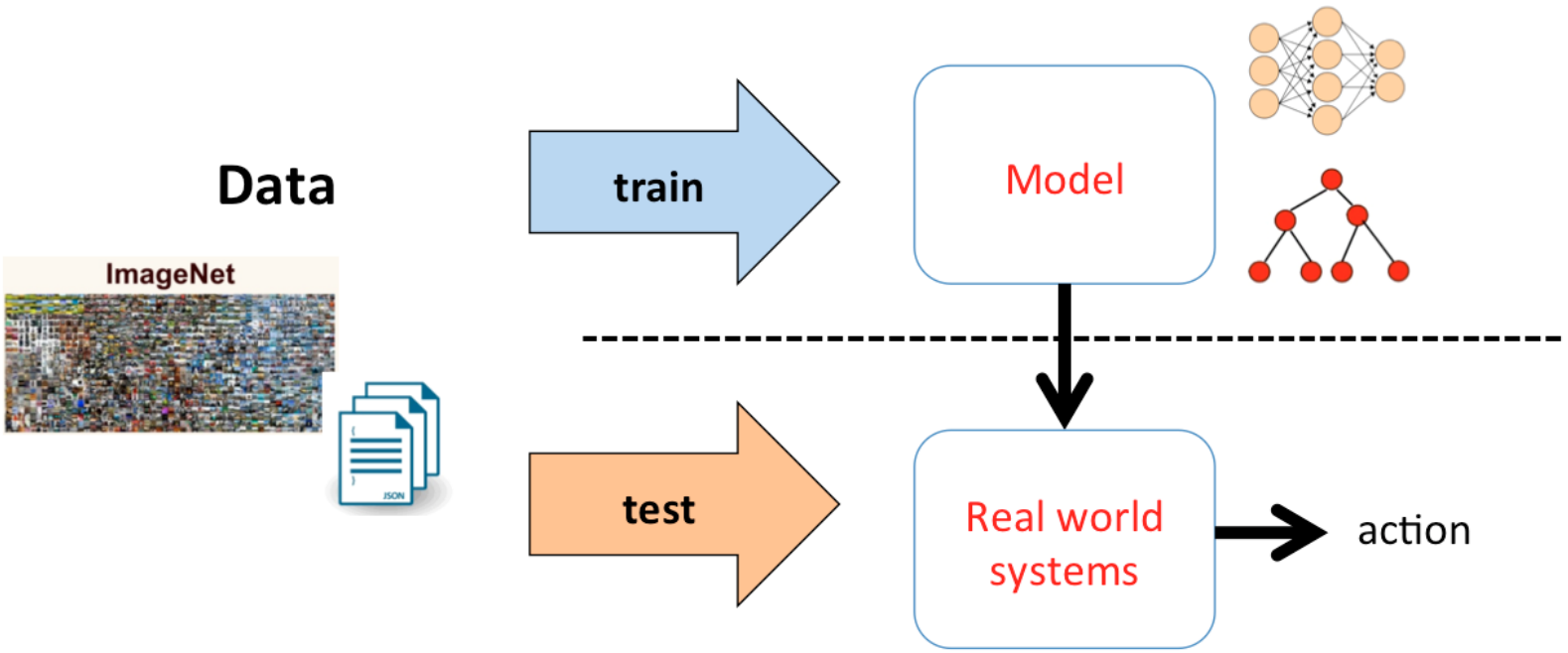
CNN: VGG, ResNet

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

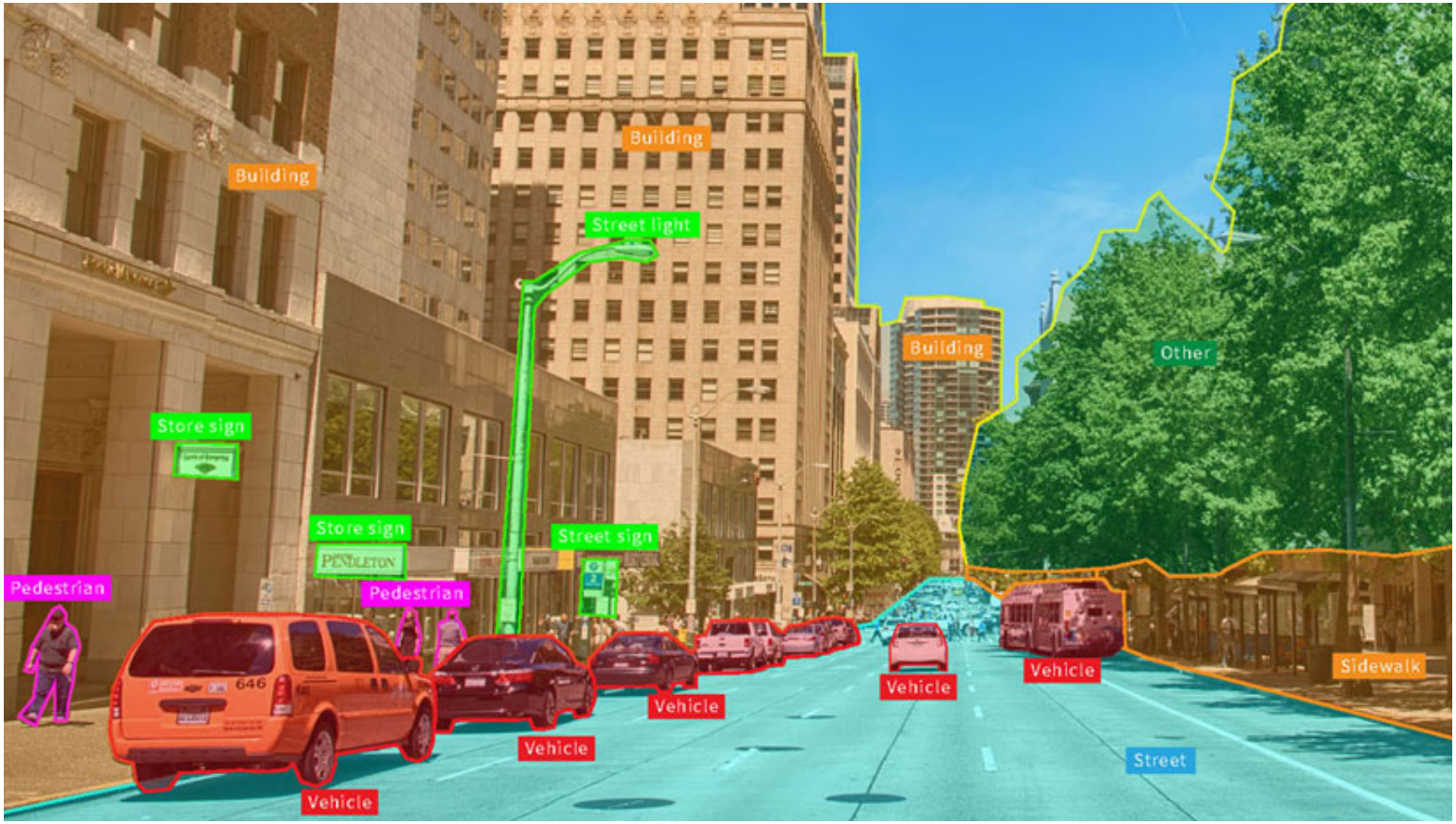
Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).



Machine Learning Systems



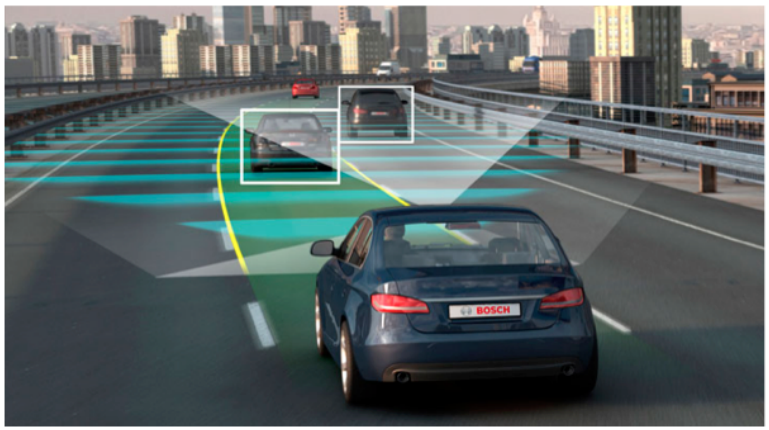
Limitation: Data Size



- Need large amount of annotated data.

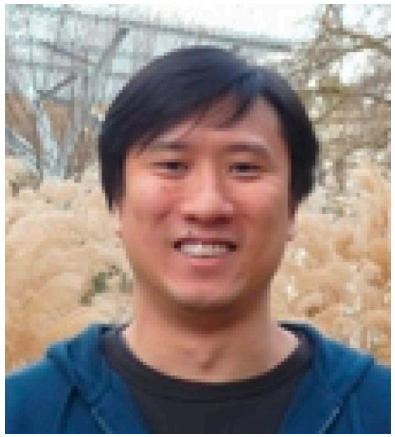
Prediction: Robustness and Safety

ML systems need to interact with real world

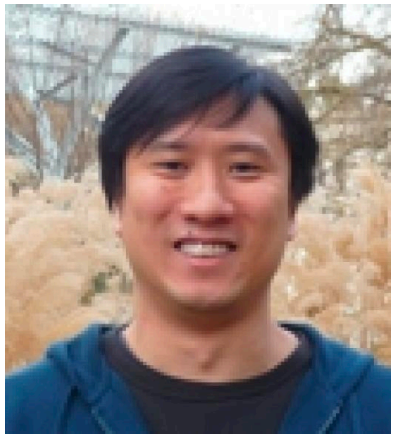


- Robustness and Safety

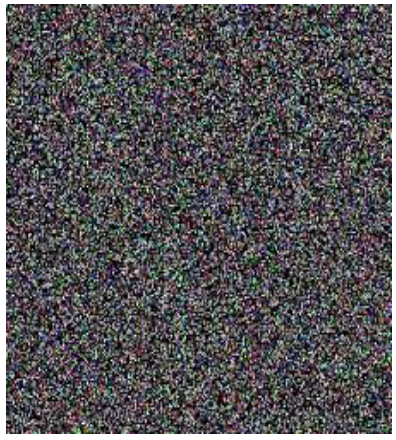
Adversarial Examples



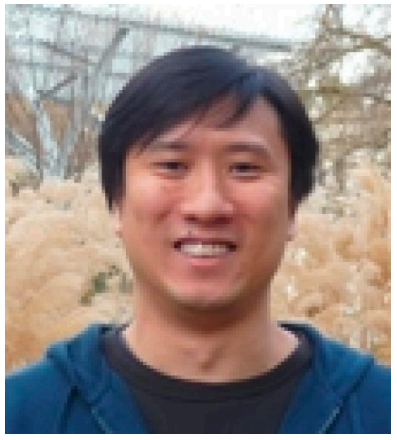
Adversarial Examples



$+0.0001\times$



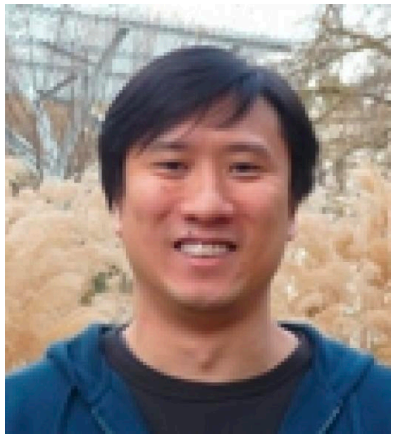
$=$



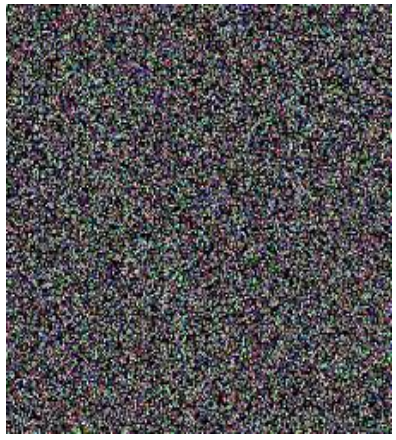
DNN

DNN

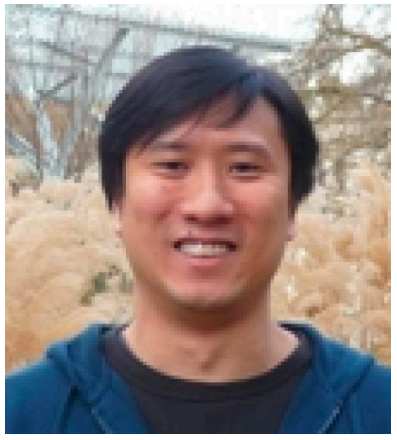
Adversarial Examples



$+0.0001\times$



$=$

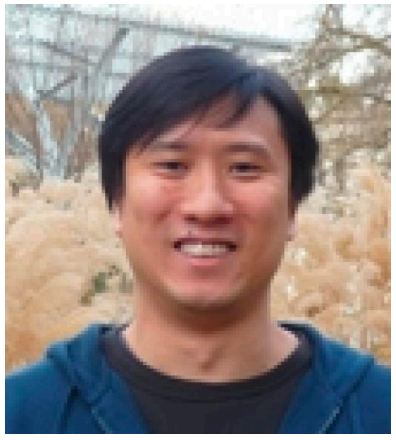


DNN

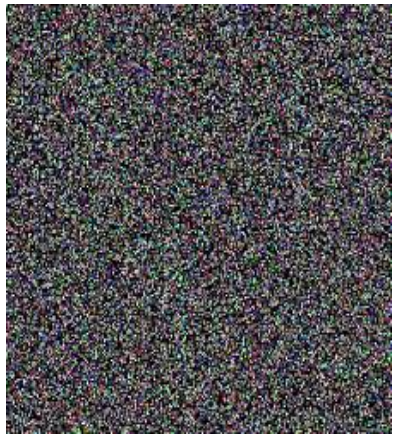
DNN

Thomas

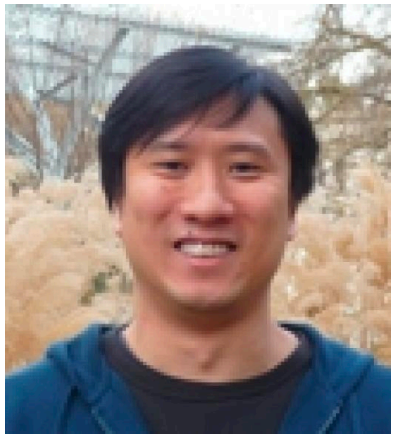
Adversarial Examples



$+0.0001\times$



$=$



DNN

↓

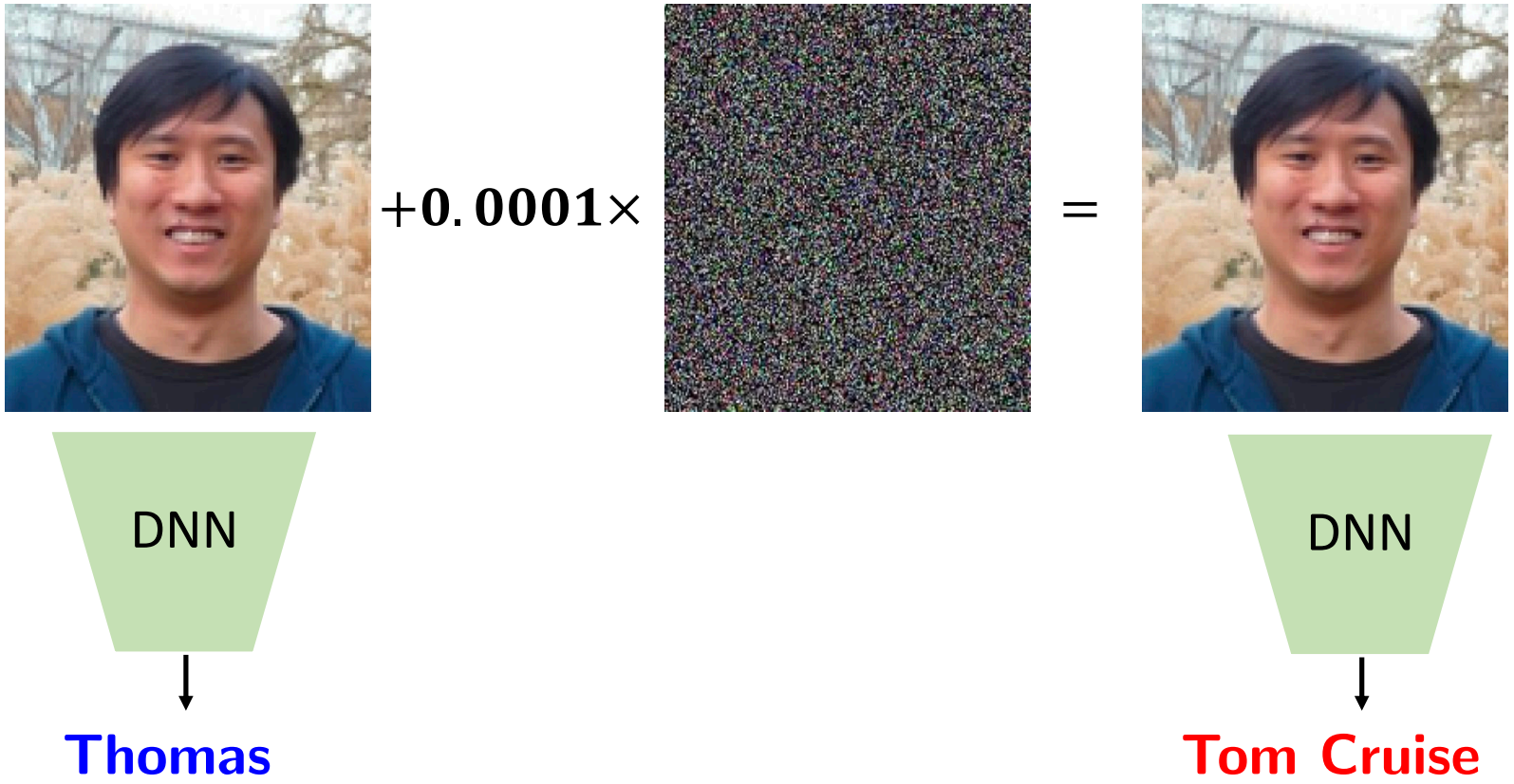
Thomas

DNN

↓

Tom Cruise

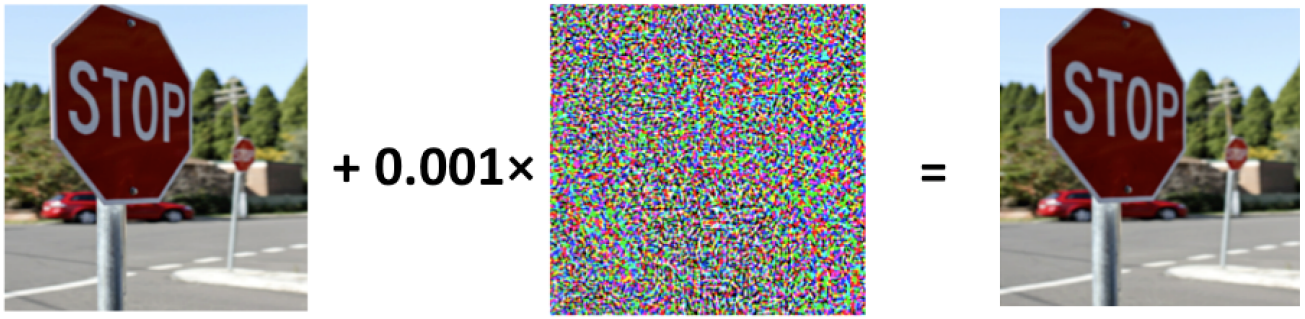
Adversarial Examples



A carefully crafted adversarial example can easily fool a deep network

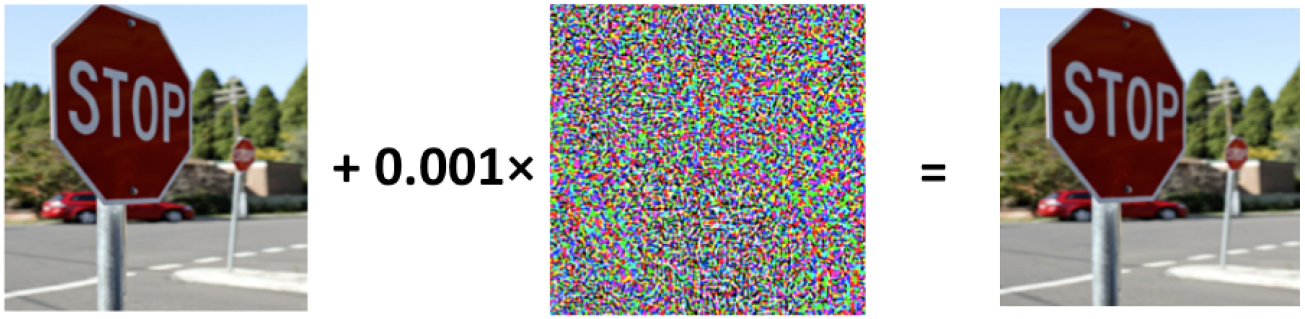
Adversarial Examples

- Robustness is critical in real systems



Adversarial Examples

- Robustness is critical in real systems



Not safe!

Research Questions

Attack: How to craft adversarial example?

Defense: How to **improve** robustness?

Notations and Attack Procedure

Model f

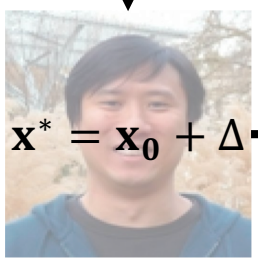
$\mathbf{y_0} = \mathbf{f}(\mathbf{x_0})$

Attack: small perturbation

$+$ Δ

$\neq$

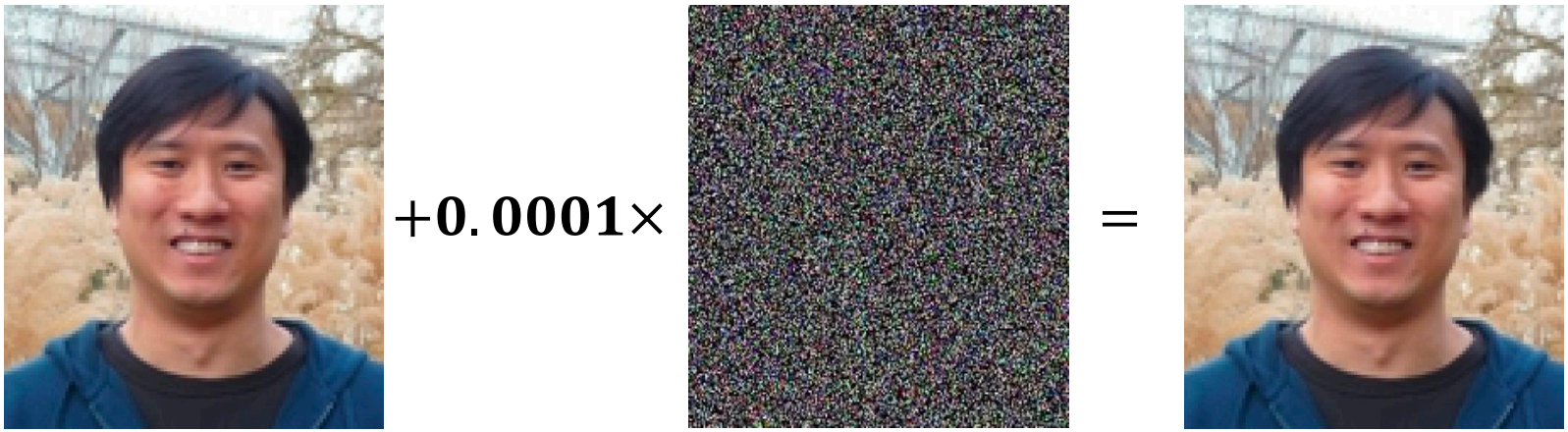
Adversarial example



Model f

$\mathbf{y}^* = \mathbf{f}(\mathbf{x}^*)$

Type of Attacks



1. Untargeted attack: $y^* \neq$Thomas
2. Targeted attack: For target class $t =$Tom Cruise, the attacker wants $y^* =$Tom Cruise

Attack vs. Defense

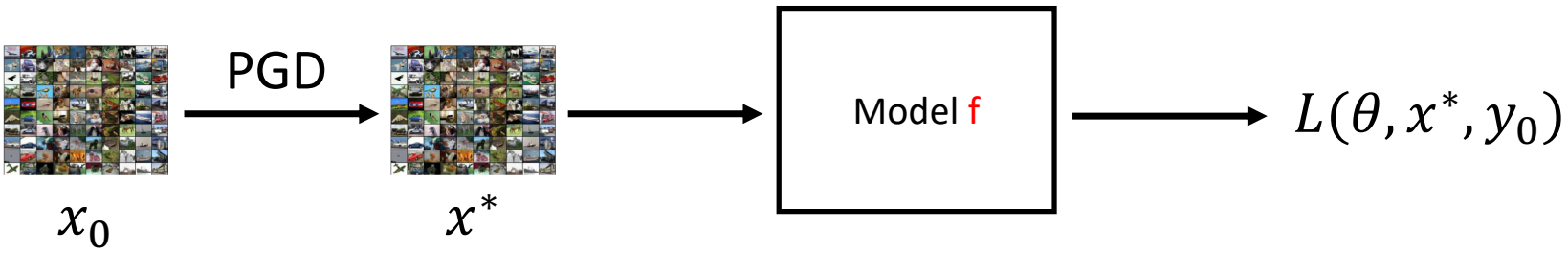
Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods

Nicholas Carlini David Wagner
University of California, Berkeley

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

Anish Athalye [*1] **Nicholas Carlini** [*2] **David Wagner** [2]

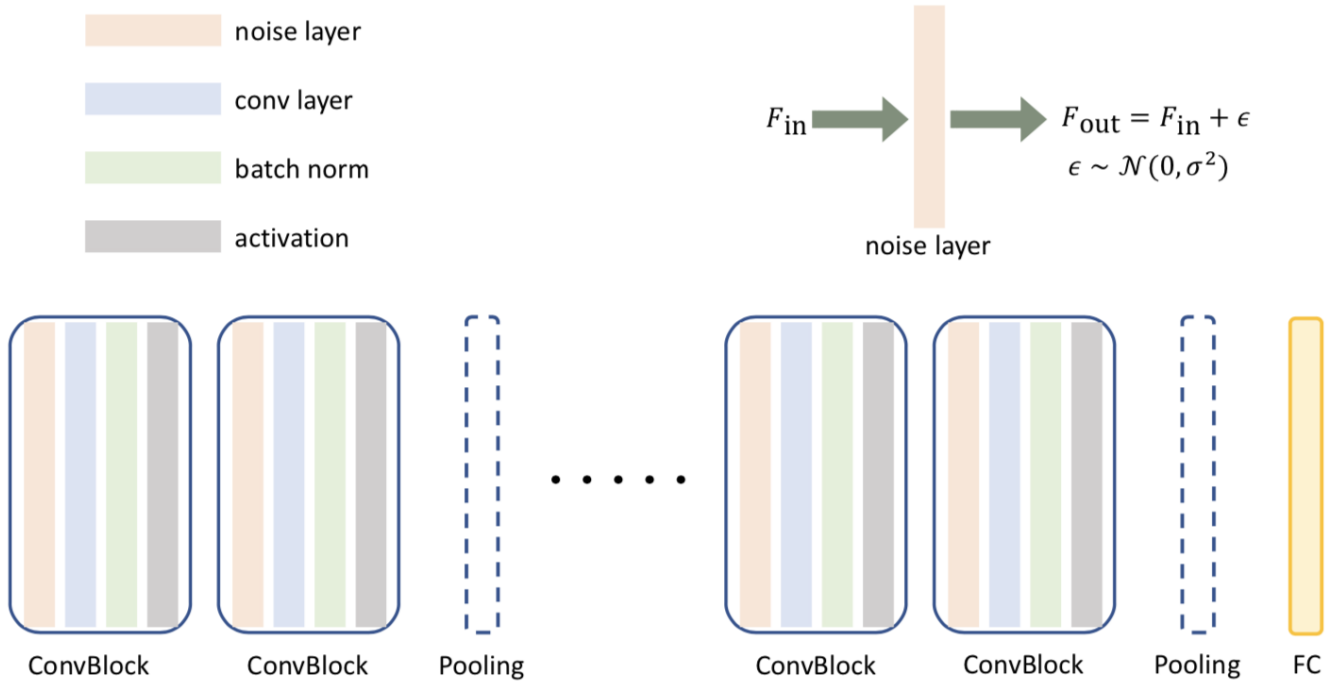
Previous Method 1: Madry's adversarial training



- **Madry's adversarial training:** Madry et al. (2018) proposed to incorporate the adversarial search inside the training process, by solving the following robust optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left\{ \max_{\|\boldsymbol{\delta}\|_\infty \leq \epsilon} L(\boldsymbol{\theta}, \boldsymbol{x}+\boldsymbol{\delta}, y) \right\}$$

Previous Method 2: Random Self-Ensemble



- **Random Self-Ensemble:** Liu et al. proposed a "noise layer", which fuses output of each layer with Gaussian noise.

Question

Questions?