# Deep Learning Basics

**Feng Xu**

**Institute of EM Data Science and Remote Sensing Intelligence**

**Fudan University, Shanghai, China**

homepage.fudan.edu.cn/fengxu

github.com/fudanxu

復旦大學 电磁波信息科学教育部重点实验室

Key Laboratory of EMW Information,Fudan University

# Deep Learning Overview

# AI/Machine Learning History

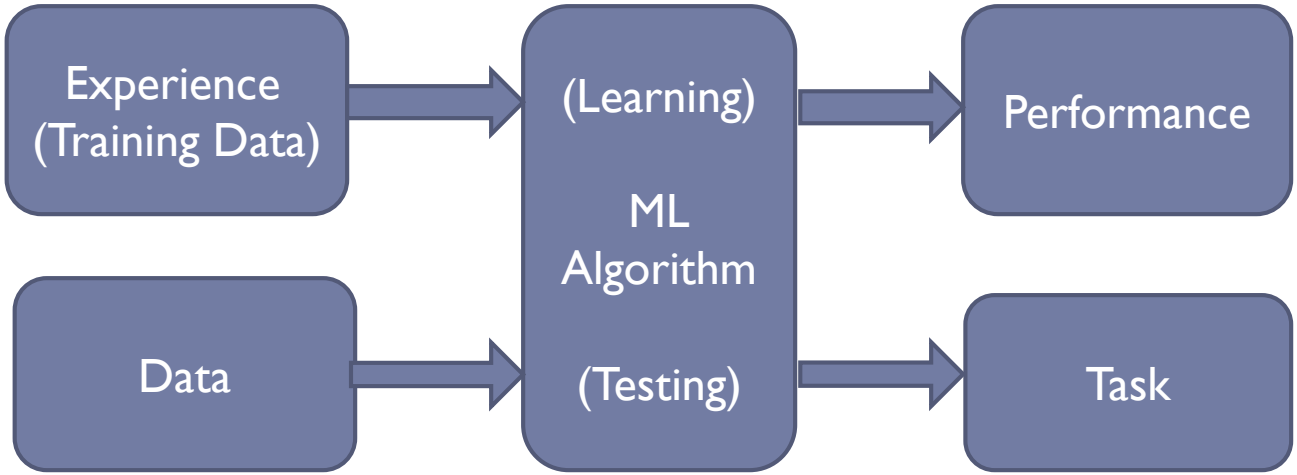
1949：Hebb Learning Rule
1950：Turing Test
1957：Perceptron
1969：Robot C
1970s：First Winter
1981：Multi-layer Perception and BP
1990s：AdaBoost、Decision Tree、SVM
1997：Schmidhuber LSTM
1998：LeCun CNN
2006：Hinton DBN
2012：AlexNet
2016：AlphaGo

# Artificial Intelligence (AI)

- AI is designed to allow machines to automate tasks that humans are good at, but cannot easily describe in math or computer language, such as speech recognition and image recognition.
- AI learns automatically from experiences, and different data drives the same algorithm to achieve different tasks.
- AI learns from human beings: AI can do what humans can; AI is unlikely to do what humans cant.
- AI will replace some of the human labor.
- AI can not think, be conscious, or have emotions.

# Machine Learning

- A computer program is said to learn from experience E with respect to task T and performance measure P, if its performance at task T, as measured by P, improves with experience E. – [Mitchell 1997]
- e.g. regression



# Machine Learning Tasks

- Regression
- Classification
- Translation
- Structured output
- Anomaly detection
- Generation
- Representation/Encoding
- Denoising
- Control/Interaction

Supervised

Unsupervised

Reinforcement

# How machine learns?

- Supervised learning
  - Learn to predict an output when given an input vector.
- Reinforcement learning
  - Learn to select an action to maximize payoff.
- Unsupervised learning
  - Discover a good internal representation of the input.

# How to do?

- 1.Design objective function
  - Supervised: Error Min（MSE,CE）
  - Unsupervised: Reconstruction error + Regularization (Sparsity)
  - Reinforcement: Reward Max

- 2.Optimization
  - Stochastic gradient descent
  - Backpropagation

# Supervised Learning

- Each training case consists of an input vector x and a target output t.
- Learns a mapping: y=f(x,w) to approximate t, by adjusting parameters w.

- Regression: The target output is a real number or a whole vector of real numbers. (Mean Square Error)
  - The price of a stock in 6 months time.
  - The temperature at noon tomorrow.

- Classification: The target output is a class label. (Cross-Entropy)
  - The simplest case is a choice between 1 and 0.
  - We can also have multiple alternative labels.

# Reinforcement Learning

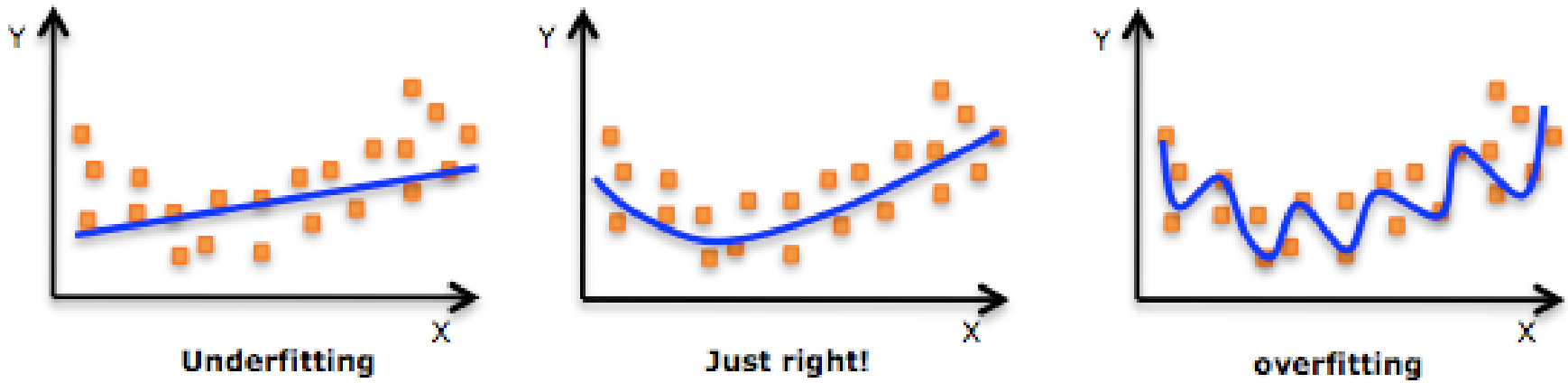
- In reinforcement learning, the output is an action or sequence of actions and the only supervisory signal is an occasional scalar reward.
  - The goal in selecting each action is to maximize the expected sum of the future rewards.
  - We usually use a discount factor for delayed rewards so that we don't have to look too far into the future.
- Reinforcement learning is difficult:
  - The rewards are typically delayed so its hard to know where we went wrong (or right).
  - A scalar reward does not supply much information.

# Unsupervised Learning

- The most natural way of learning.
- Many researchers thought that clustering was the only form of unsupervised learning.
- One major aim is to create an internal representation of the input that is useful for subsequent supervised or reinforcement learning

- It provides a compact, low-dimensional representation of the input.
  - High-dimensional inputs typically live on or near a low- dimensional manifold (or several such manifolds), e.g. PCA
- It provides an economical high-dimensional representation of the input in terms of learned features.
  - Binary features are economical.
  - So are real-valued features that are nearly all zero.
- Distributed representation.
  - Instead of one-hot, use combination of multiple dimension to represent different features.

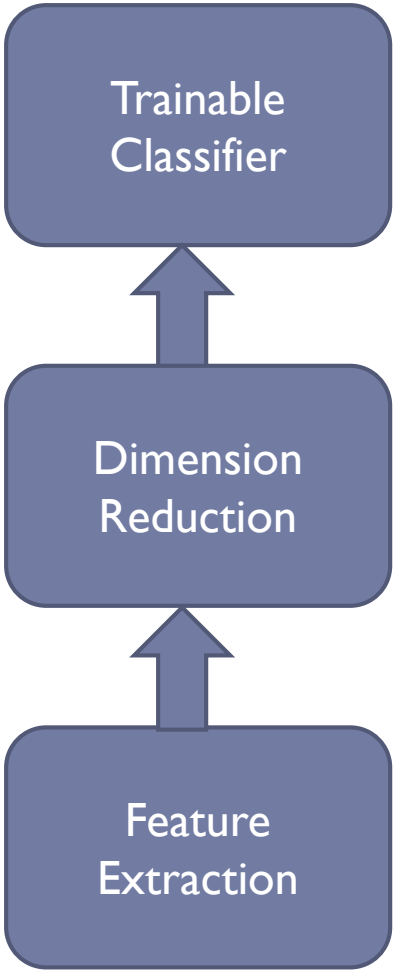
# Machine Learning's essence is … Fitting

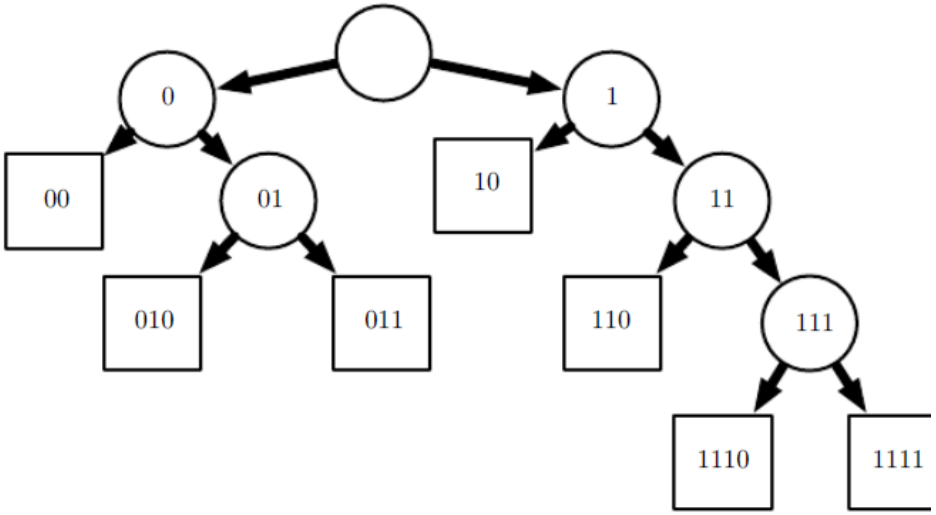


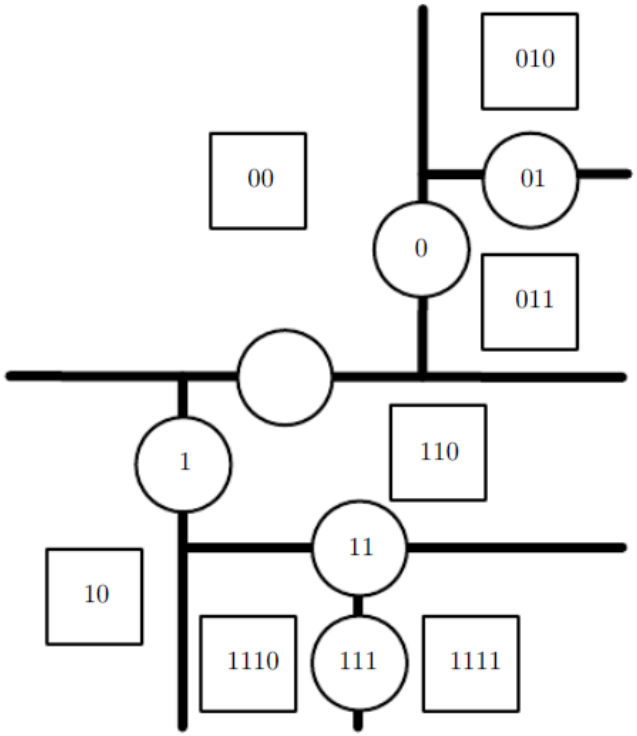

Balance between Data Size vs. Function Capacity

[Goodfellow, 2017]

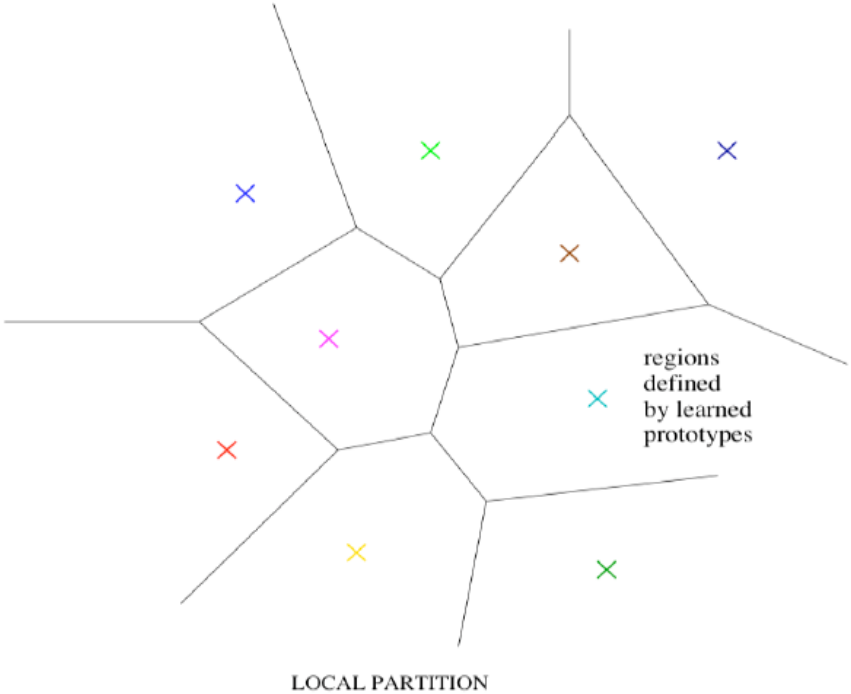
# Why Deep:

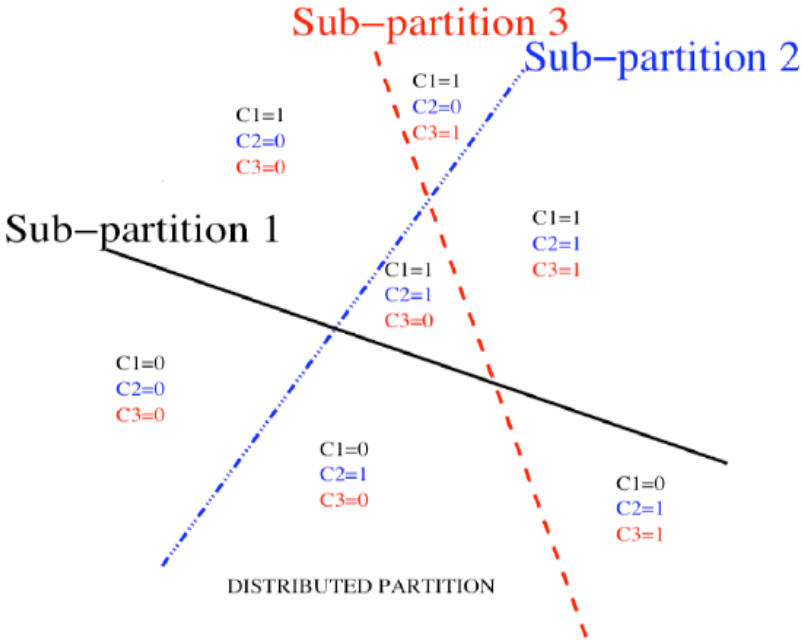

Trainable Classifier
Dimension Reduction
Feature Extraction


0
1
00
01
10
11
010
011
110
111
1110
1111


010
00
01
0
011
1
110
11
10
1110
111
1111

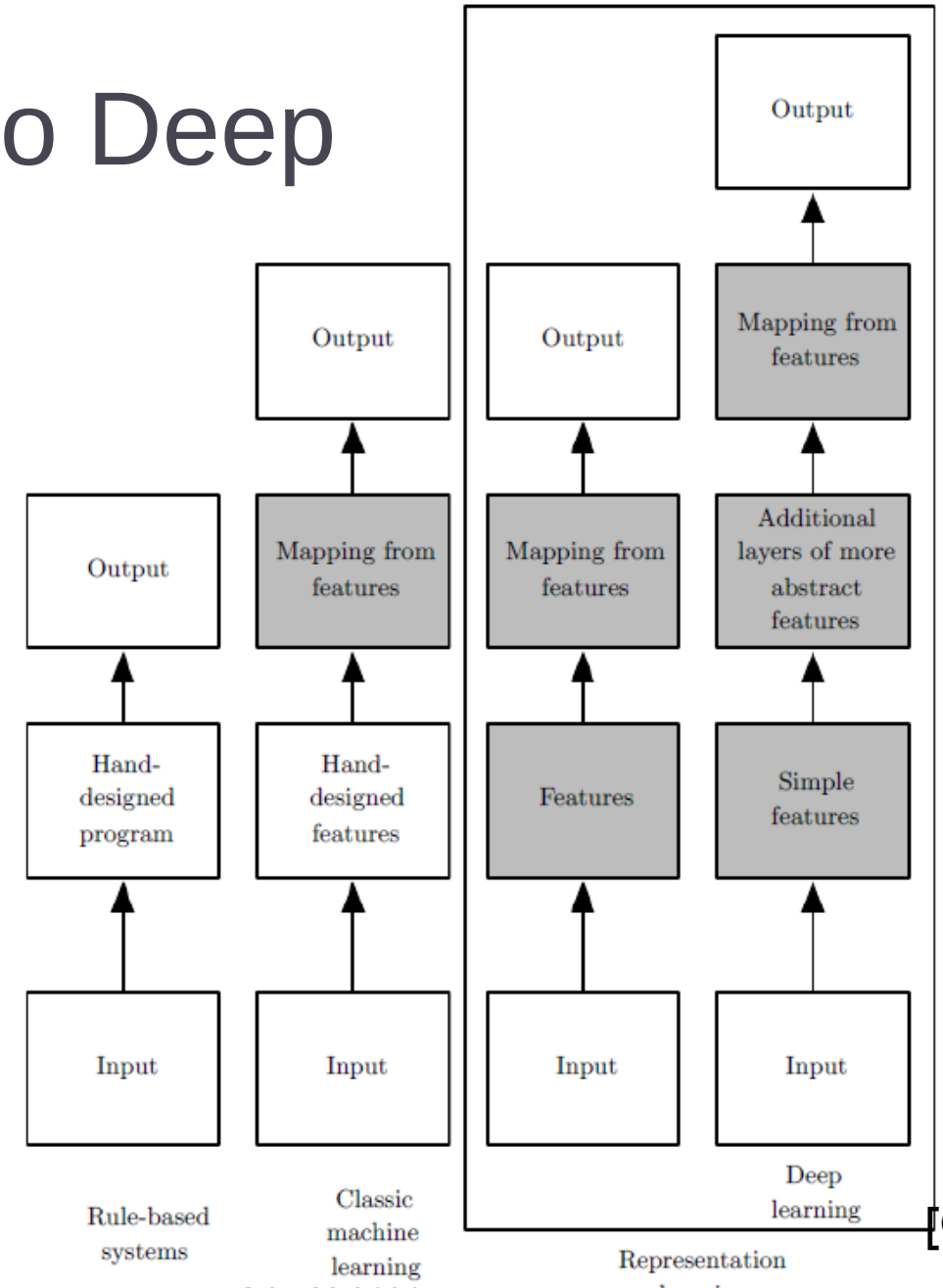
# Hierarchical Features: Exponential power of representation



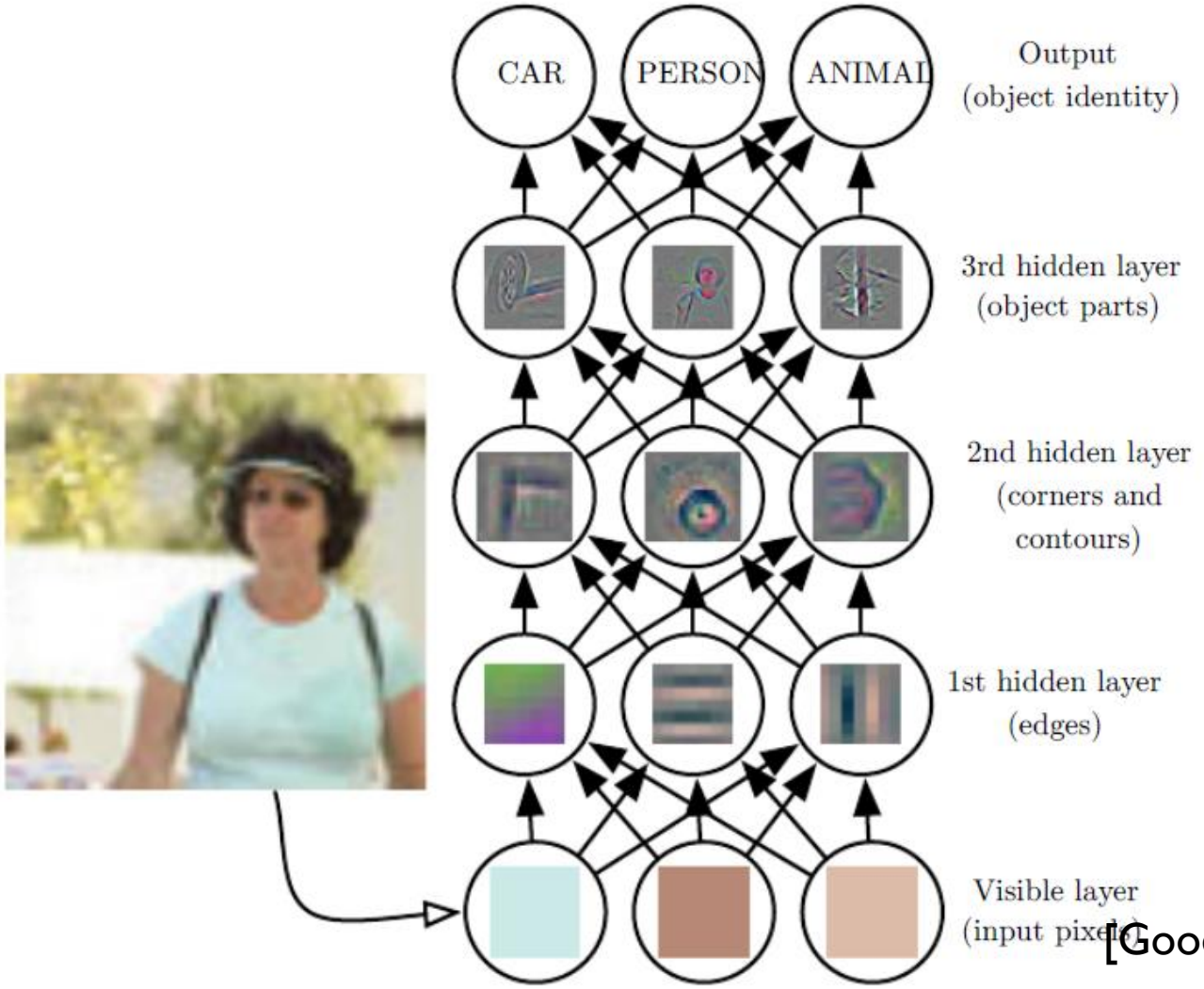

Learning a **set of parametric features** that are not mutually exclusive can be exponentially more statistically efficient than having nearest-neighbor-like or clustering-like models

# Shallow to Deep



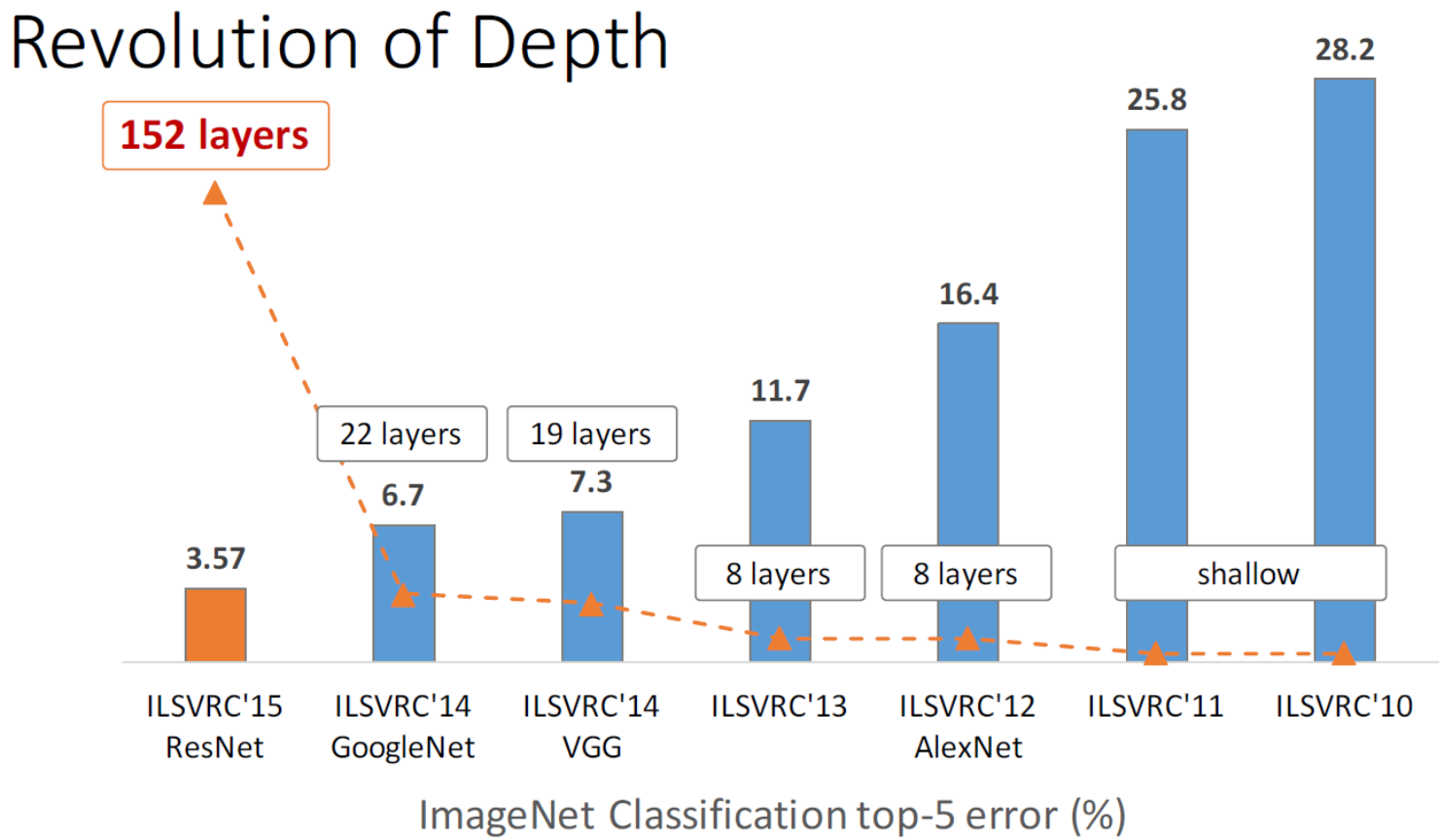

[Goodfellow, 2017]

# Hierarchical Representation




[Goodfellow, 2017]

# Performance vs. Capacity

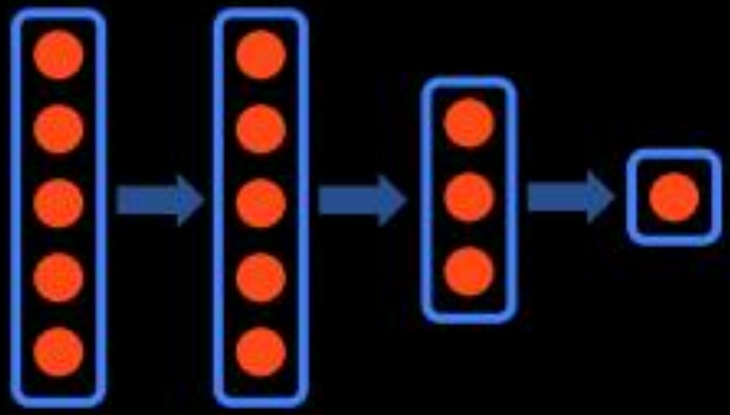



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

# Why is Deep Learning taking off?

Andrew Ng's Metaphor:

Engine



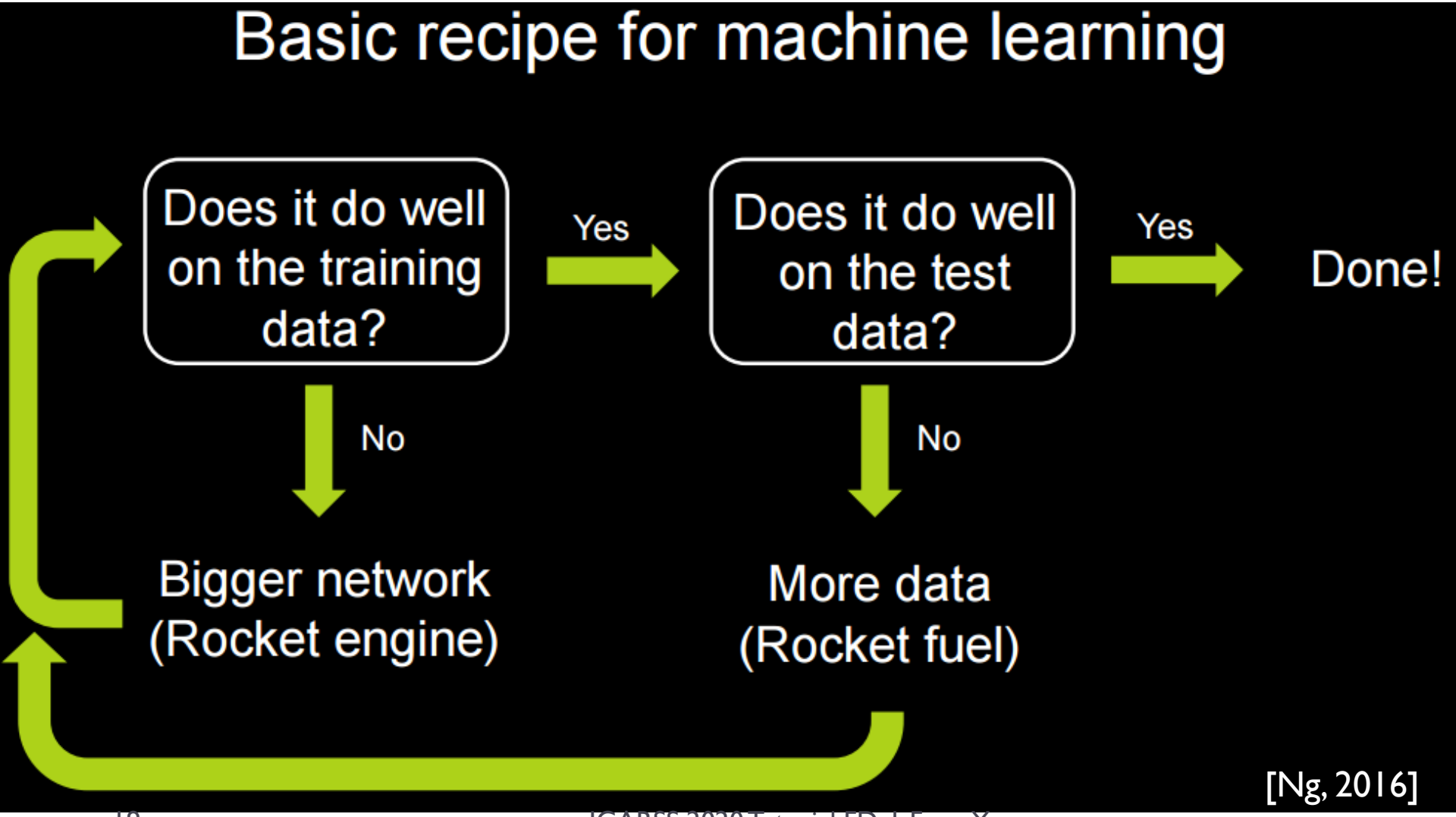
Large neural networks

Fuel

Data

[Ng, 2016]

# Andrew Ng's Recipe




[Ng, 2016]

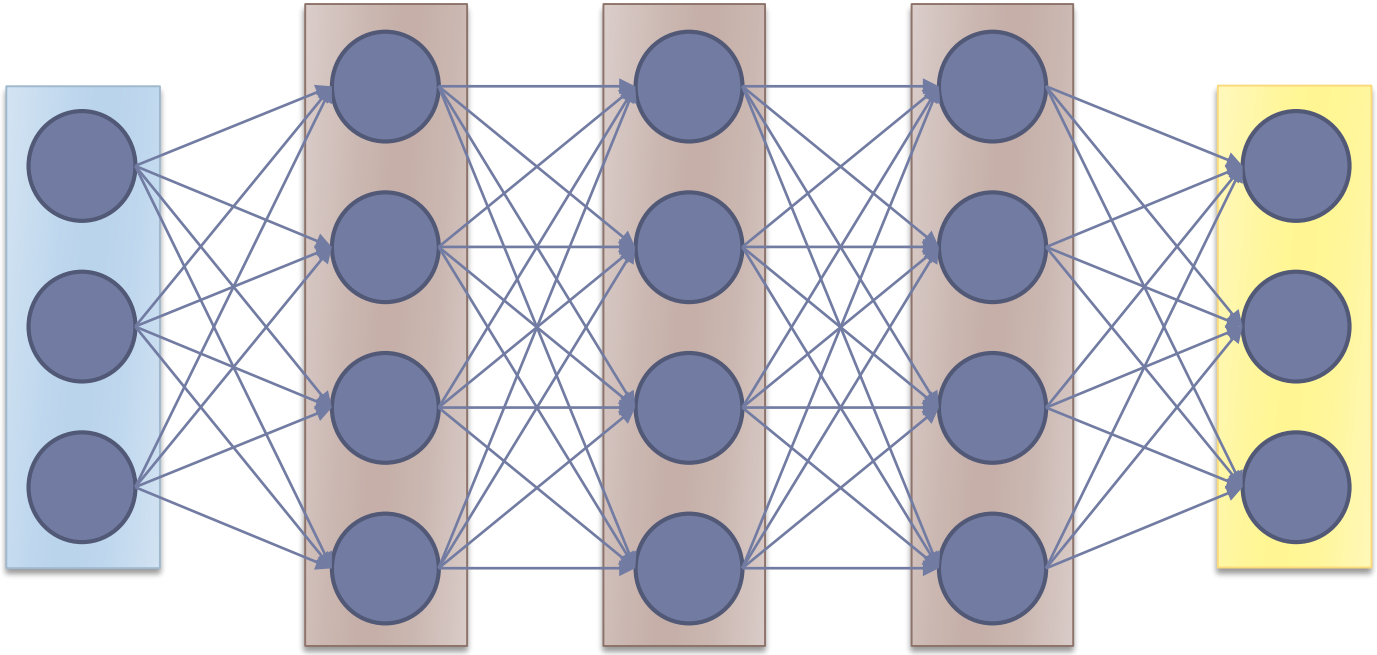
# Deep Learning

- Definition: Machine learning with Deep Architecture (automatically learn multi-layer representation）

- Deep neural network——Mainstream DL
  - Definition: NN with more than one hidden layer

- Arbitrary network: Directed Acyclic Graph

# Deep Neural Network (DNN)



**Input layer** **hidden layer 1** **hidden layer 2** **hidden layer 3** **Output layer**

- Multiple Layers of simple units
- Each units computes a weighted sum of its inputs simple units
- Weighted sum is passed through a non-linear function: $\boldsymbol{o}^{(l)} = \max\left(\boldsymbol{o}^{(l-1)} \cdot \boldsymbol{W}^{(l)}, 0\right)$
- The learning algorithm changes the weights: $\Delta \boldsymbol{W}^{(l)} = \epsilon \cdot \left(\boldsymbol{o}^{(l-1)}\right)' \boldsymbol{e}^{(l)}$

# Approach to Artificial Intelligence

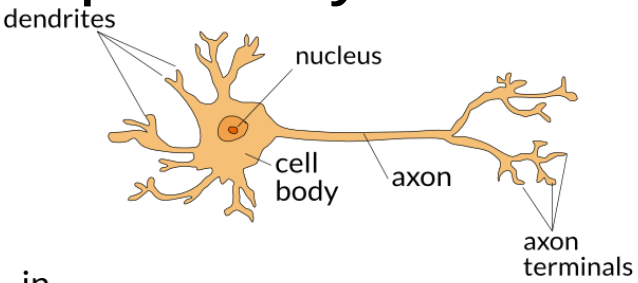
- Airplane inspired by birds, but do not flap wings
  - Get inspiration from brain, but not copy brain.
- They use the same basic aerodynamic principles for flight
  - Understand principles of learning and intelligence
- What is the equivalence of aerodynamics for intelligence?
  - Lecun: Artificial Neural Networks

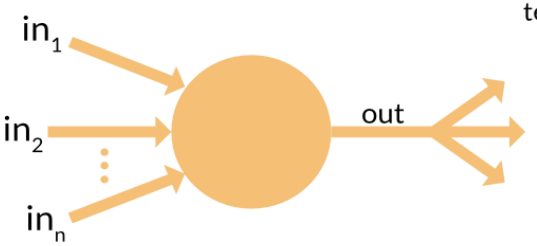


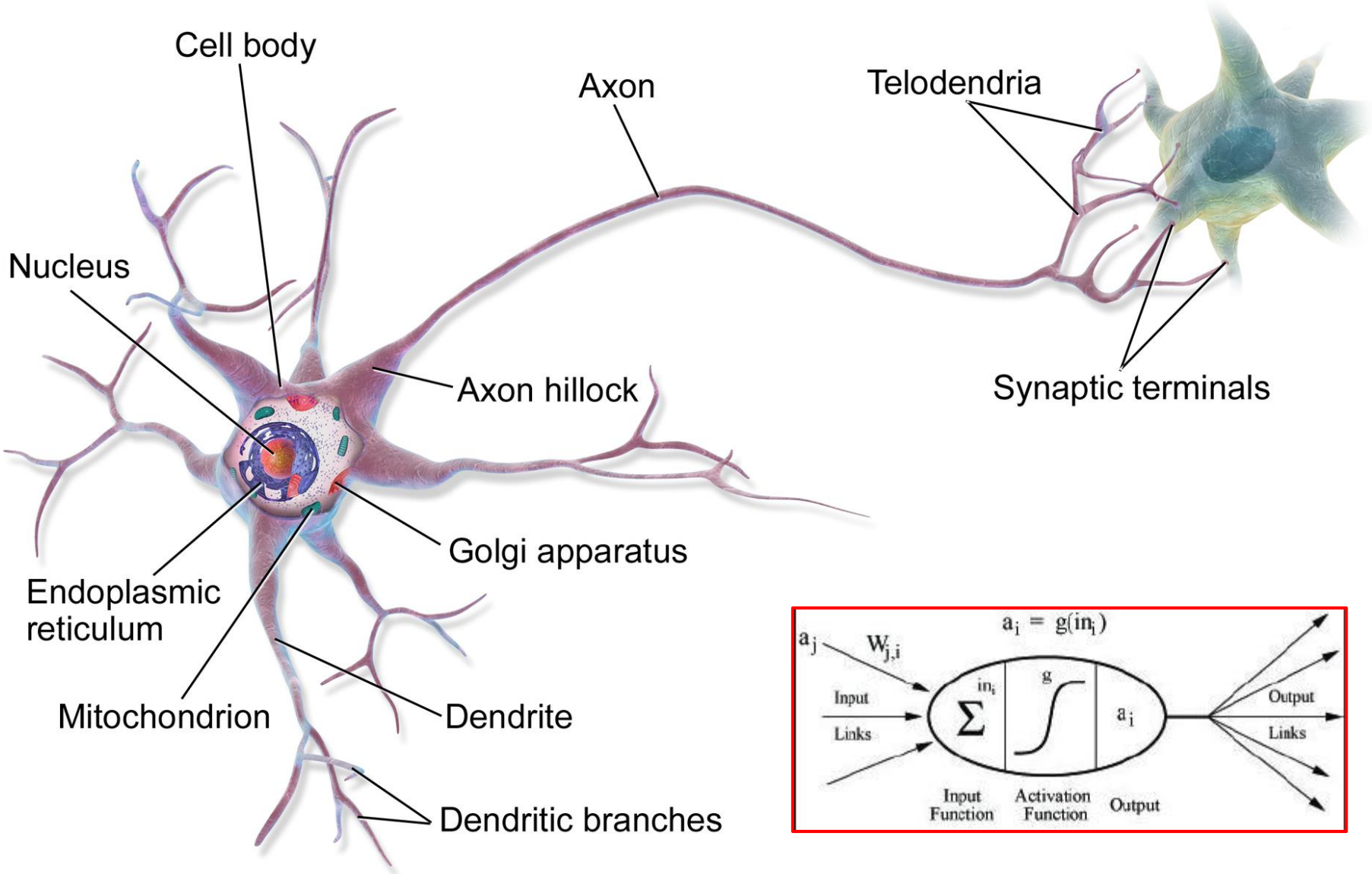
[LeCun, 2016]

# Biological basis of DNN

- Brain is composed of same matter, but can accomplish different functions
- Human brain has ~10^11 neurons
- Neurons communicates with spiking signal
- Synapses between neurons has plasticity
- Brain  vs. Computer
  - Parallel vs. Serial
  - Bain good at: Recognition
  - Computer good at: Arithmetic



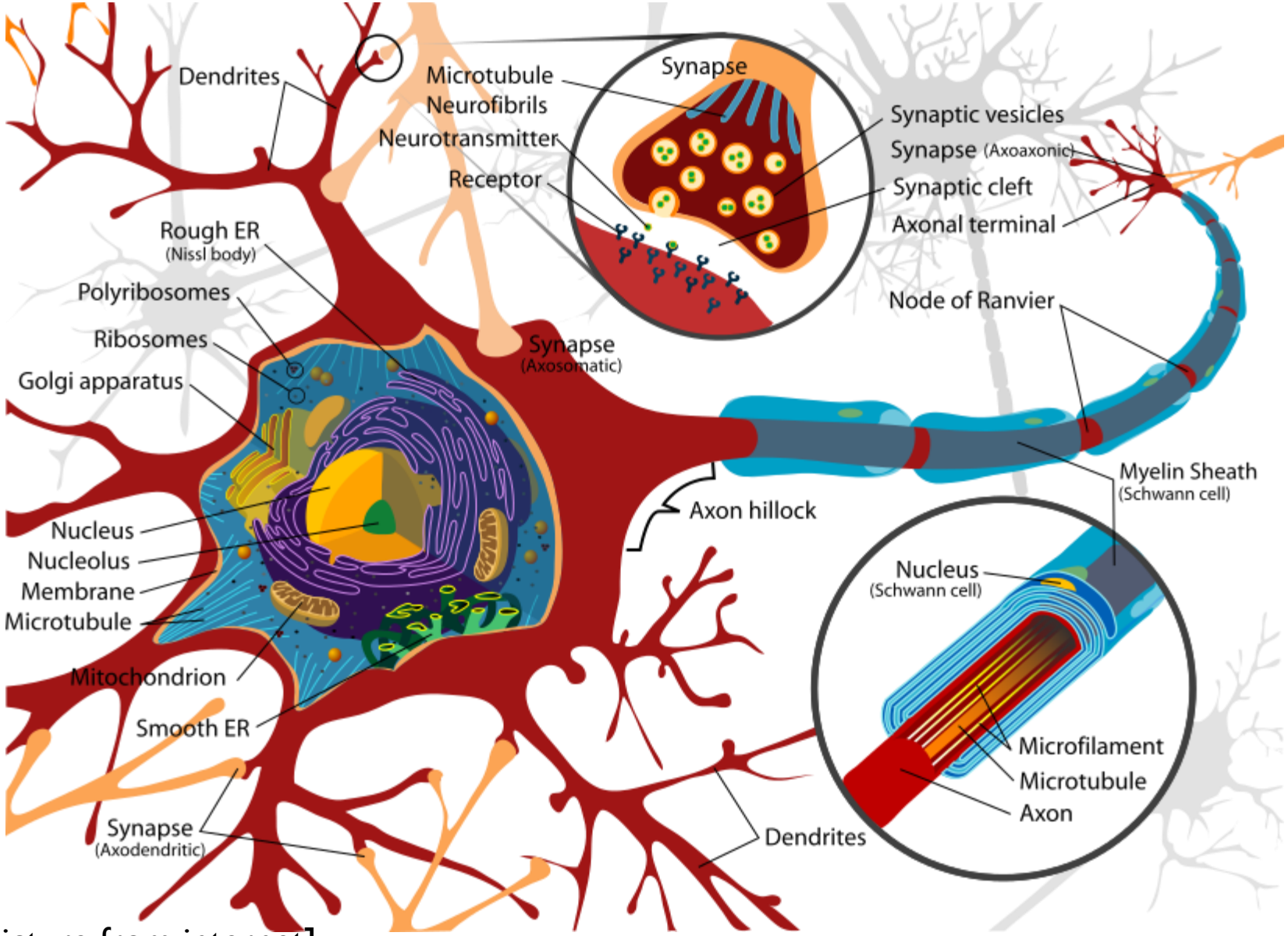









[picture from internet]



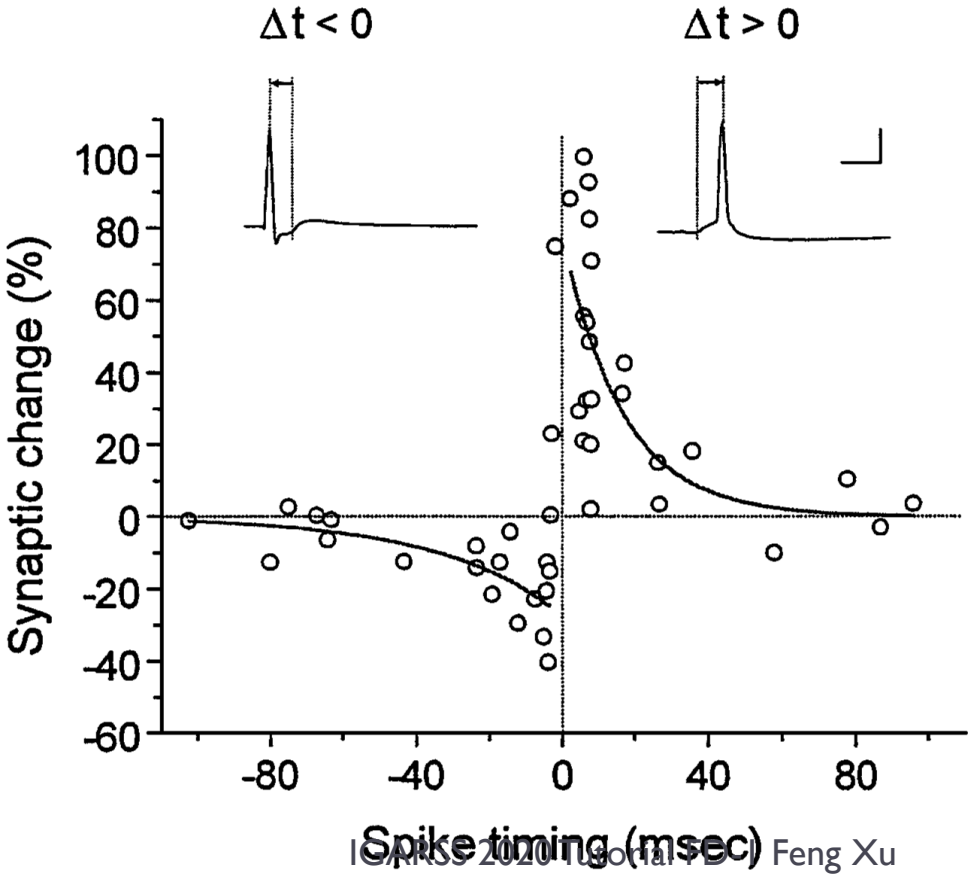

[picture from internet]

# How Brain Works?

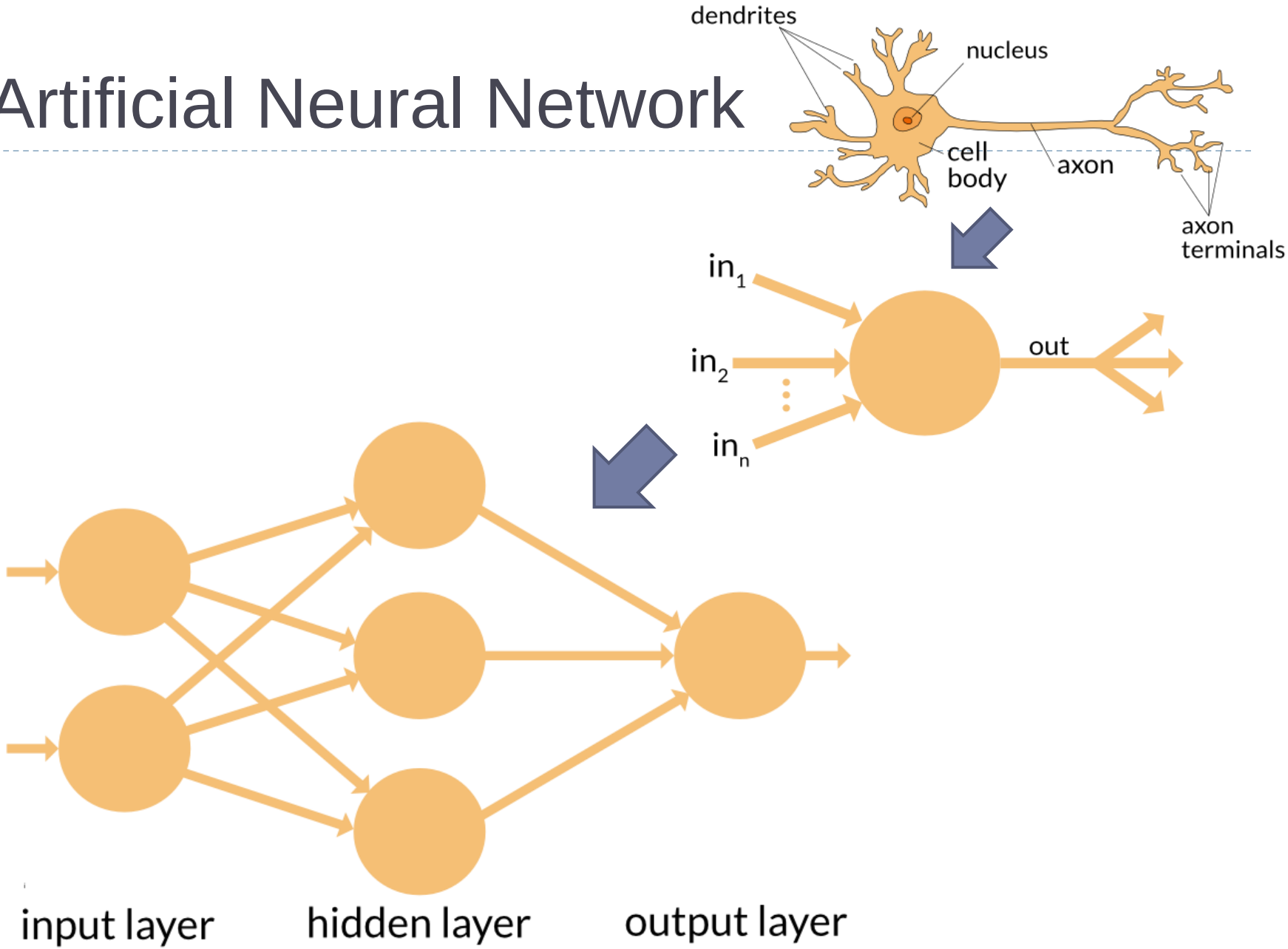
- Each neuron receives inputs from other neurons
  - Multiple inputs and multiple outputs.
  - Cortical neurons use spikes to communicate.
- The effect of each input line on the neuron is controlled by a synaptic weight
  - The weights can be positive or negative.
- The synaptic weights adapt so that the whole network learns to perform useful computations
  - Recognizing objects, understanding language, making plans, controlling the body.
- You have about 10^11 neurons each with about 10^4 weights.
  - A huge number of weights can affect the computation in a very short time. Much better bandwidth than a workstation.
- Cortex looks pretty much the same all over.
  - Early brain damage makes functions relocate.
- Different bits of the cortex do different things.
  - Local damage to the brain has specific effects.
  - Specific tasks increase the blood flow to specific regions.

# How Brain Learns?

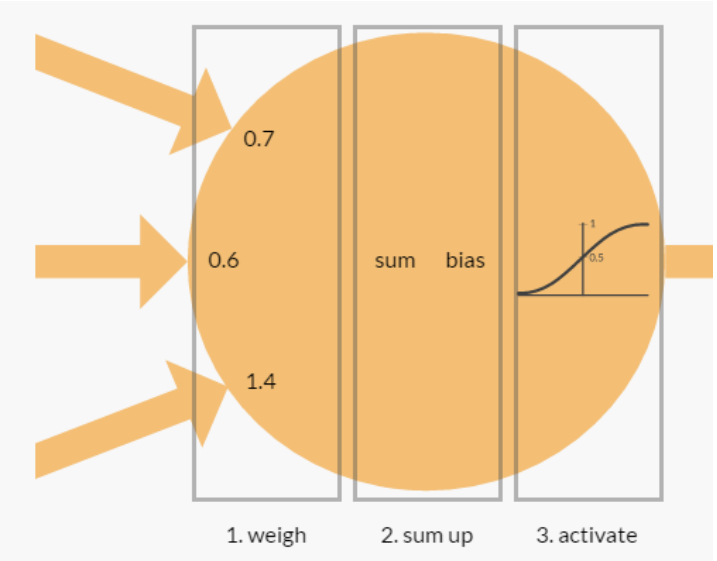
- Fire together, Wire together.
- Spiking-Time-Dependence-Plasticity



# Artificial Neural Network



# Artificial Neuron




$$y = b + \sum_i x_i w_i$$

bias

i th input

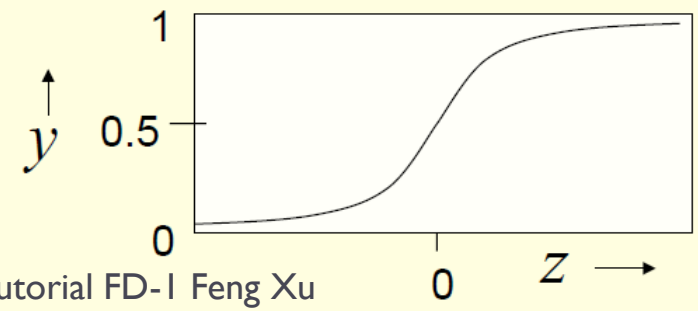
output

index over input connections

weight on i th input

$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$

Sigmoid Activation

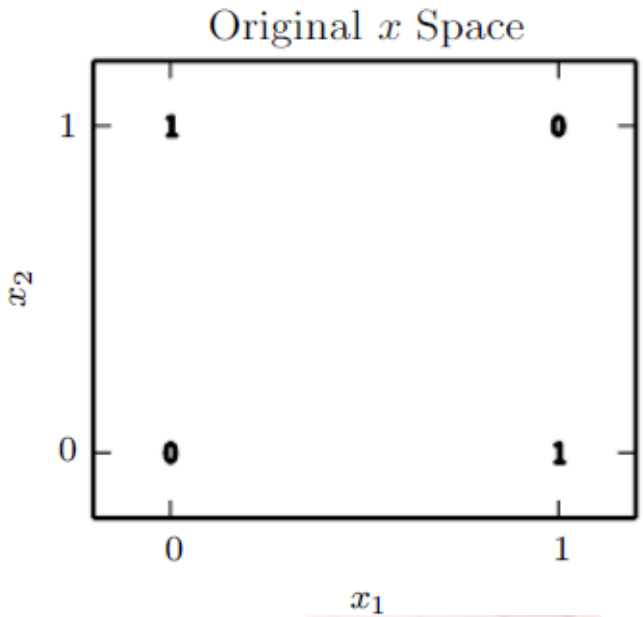


# The XOR Problem

- Linear Perceptron

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{x}^\top \boldsymbol{w} + b.$$

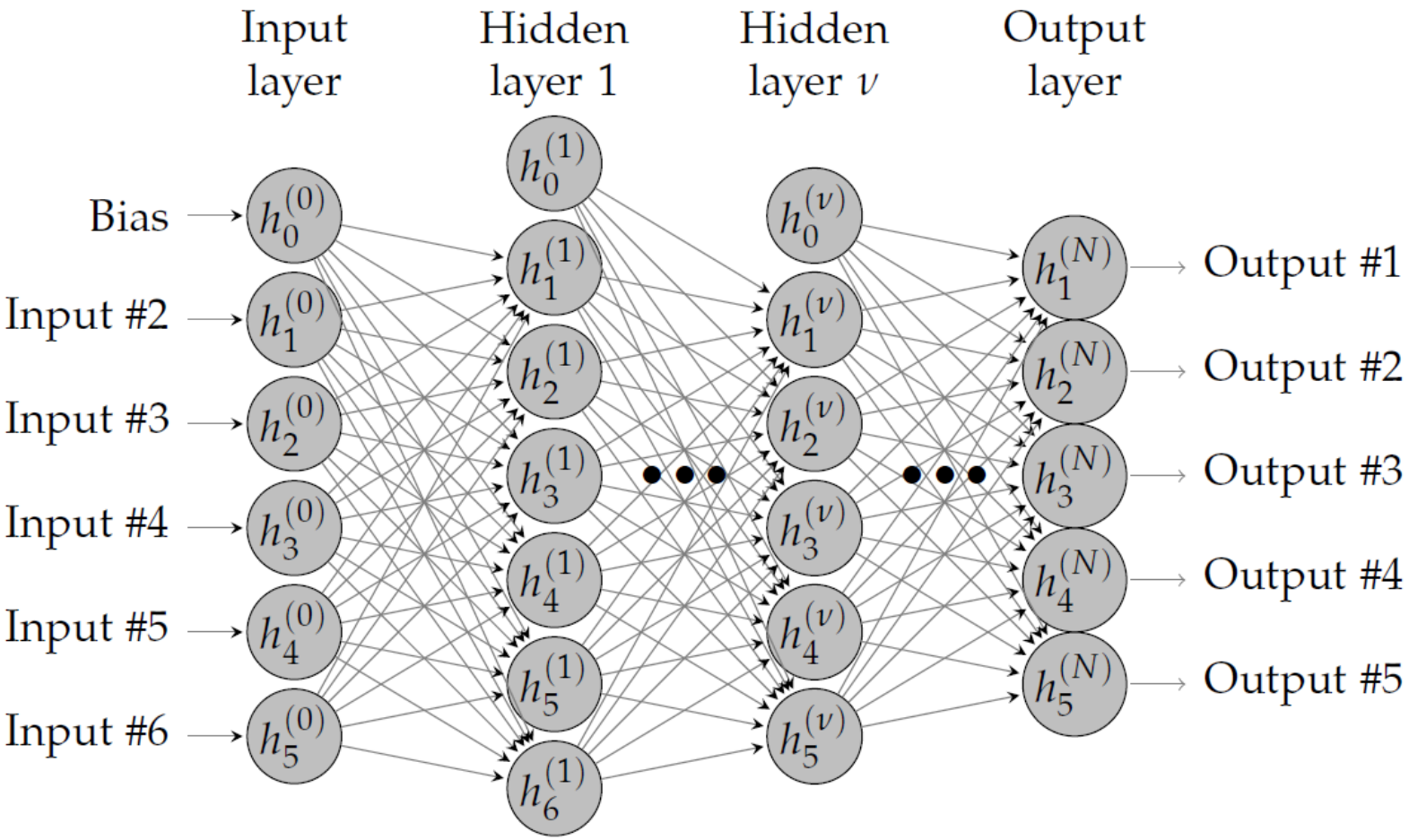
$$\boldsymbol{w} = \boldsymbol{0} \text{ and } b = \tfrac{1}{2}.$$




- The XOR Controversy
  - Minsky book
  - Single-Layer Perceptron
  - Multi-Layer Perceptron

Marvin Minsky

# Feedforward Neural Network



# Back-Propagation

## Learning representations by back-propagating errors

**David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

**We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].**

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, $x_j$, to unit $j$ is a linear function of the outputs, $y_i$, of the units that are connected to $j$ and of the weights, $w_{ji}$, on these connections

$$x_j = \sum_i y_i w_{ji} \tag{1}$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, $y_j$, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \tag{2}$$

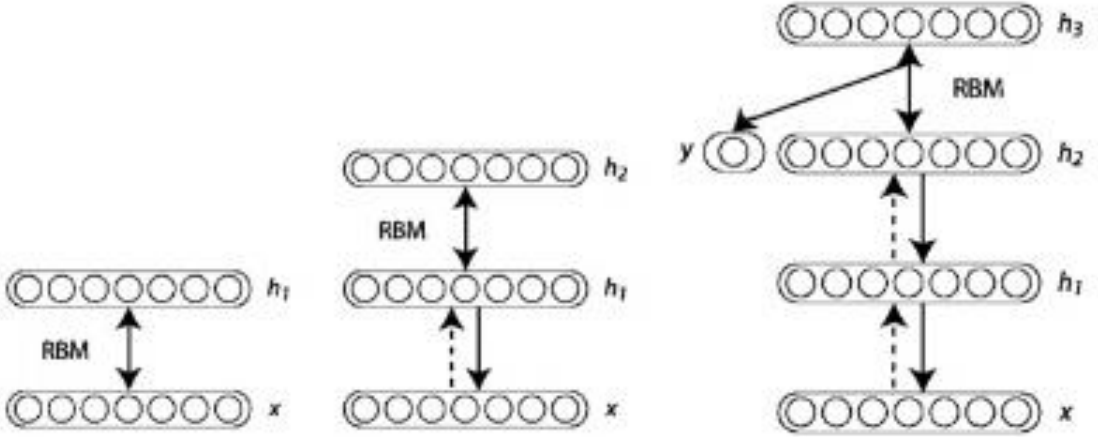
† To whom correspondence should be addressed.

© 1986 Nature Publishing Group

# Deep Belief Network

Geoff Hinton

- Iterative pre-training construction of **Deep Belief Network (DBN)** (Hinton et al., 2006)




(a) RBM for $\mathbf{x}$ (b) RBM for $\mathbf{h}^1$ (c) RBM for $\mathbf{h}^2$ and $y$

# Convolutional Neural Network

Yann LeCun

- LeCun, 1998, LeNet-5

PROC. OF THE IEEE, NOVEMBER 1998

## Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

*Abstract*—
**Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful**

I. Introduction

Over the last several years, machine learning techniques,

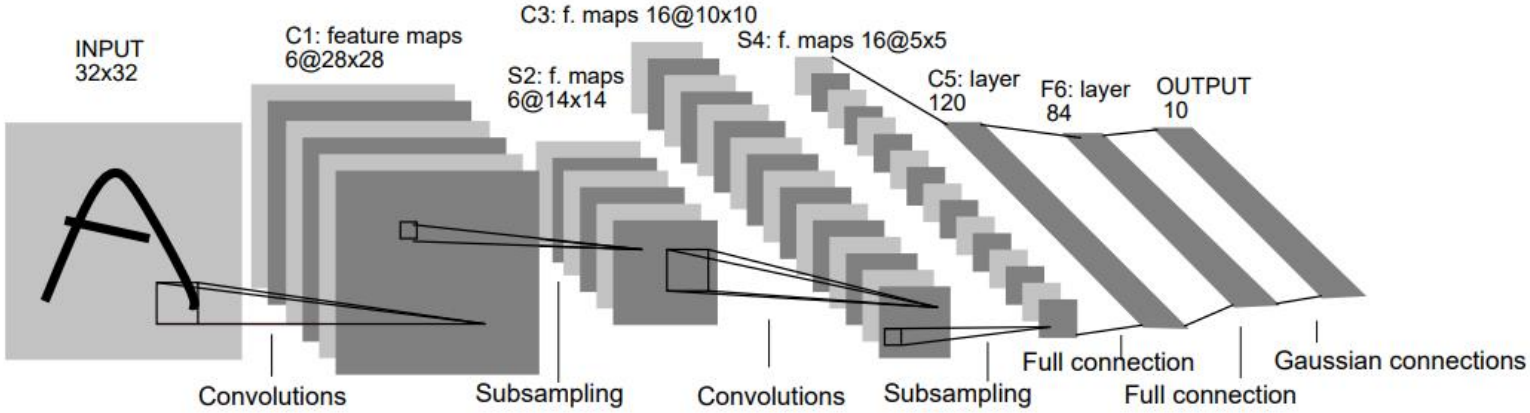
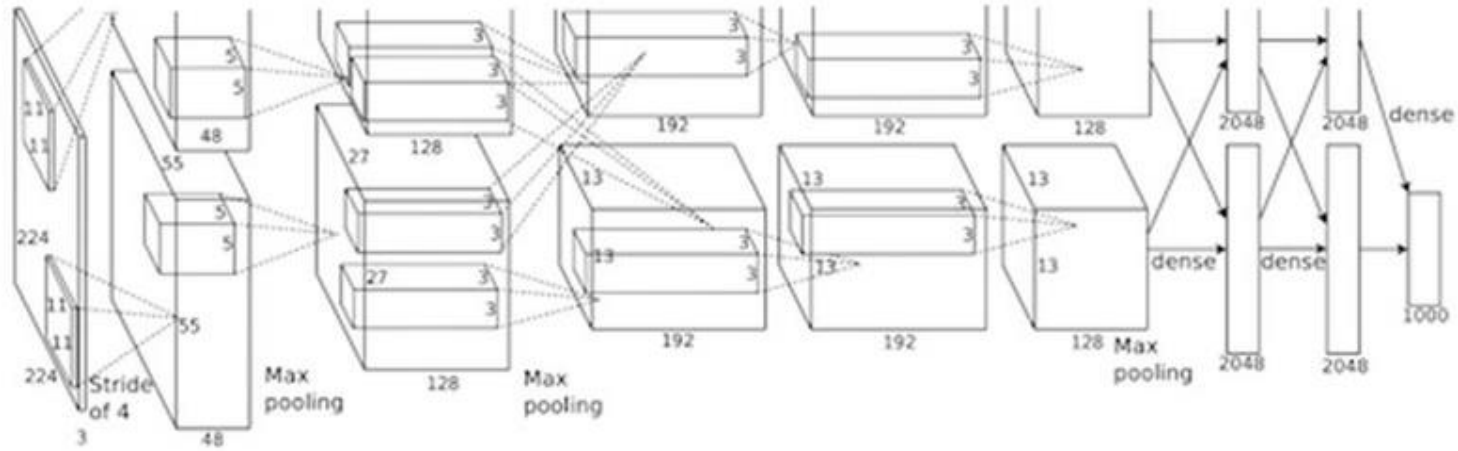


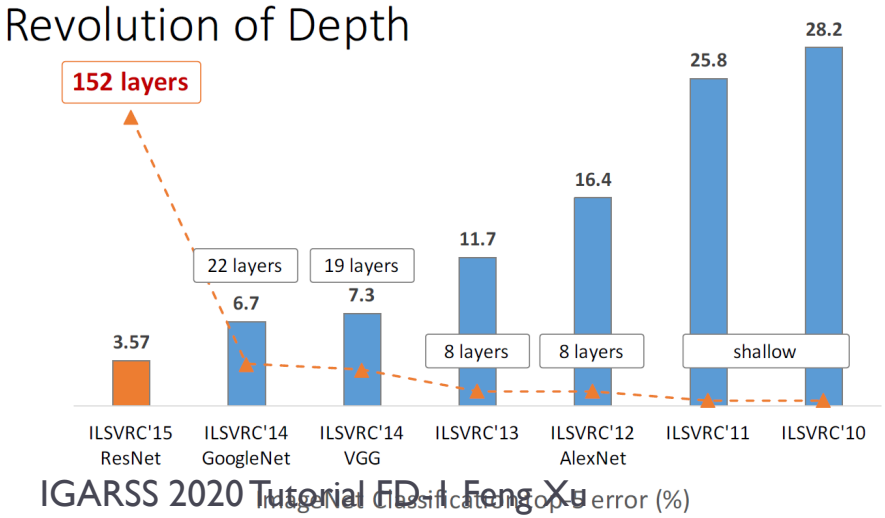

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# ImageNet




AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

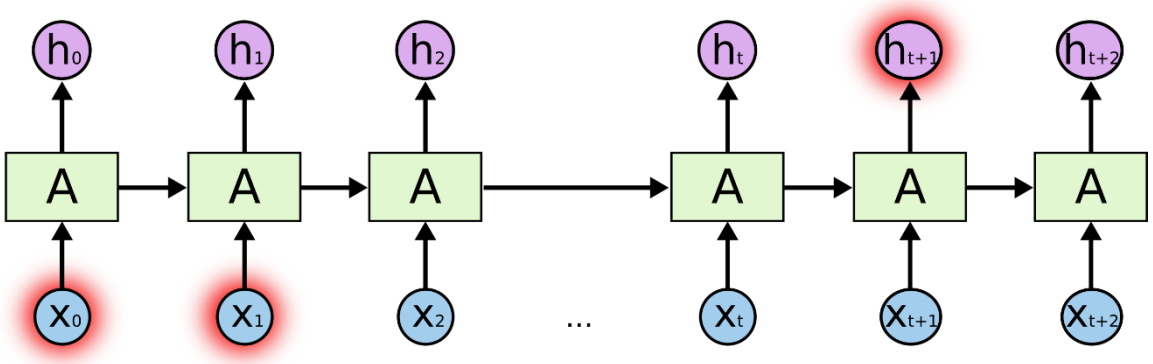
- AlexNet
- VGG
- GoogLeNet
- ResNet

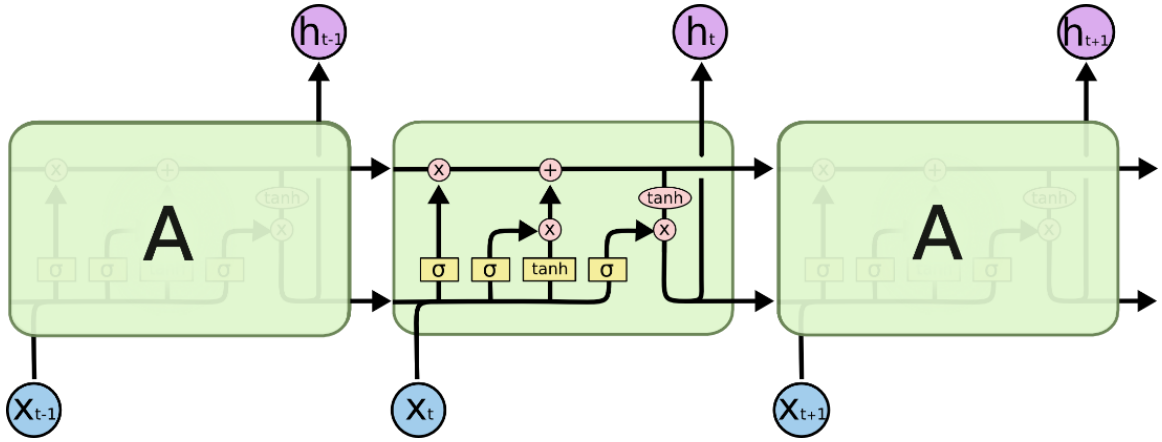



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

# Recurrent Neural Network

Jürgen Schmidhuber




Long Short-Term Memory (LSTM)

# Applications

Speech recognition

Image Search

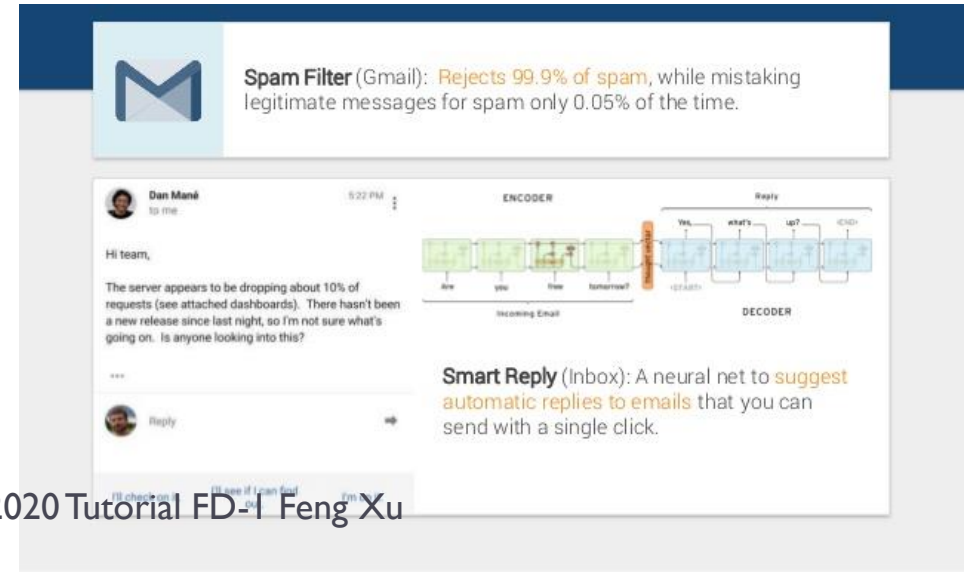
Ads; Web search

Andrew Ng

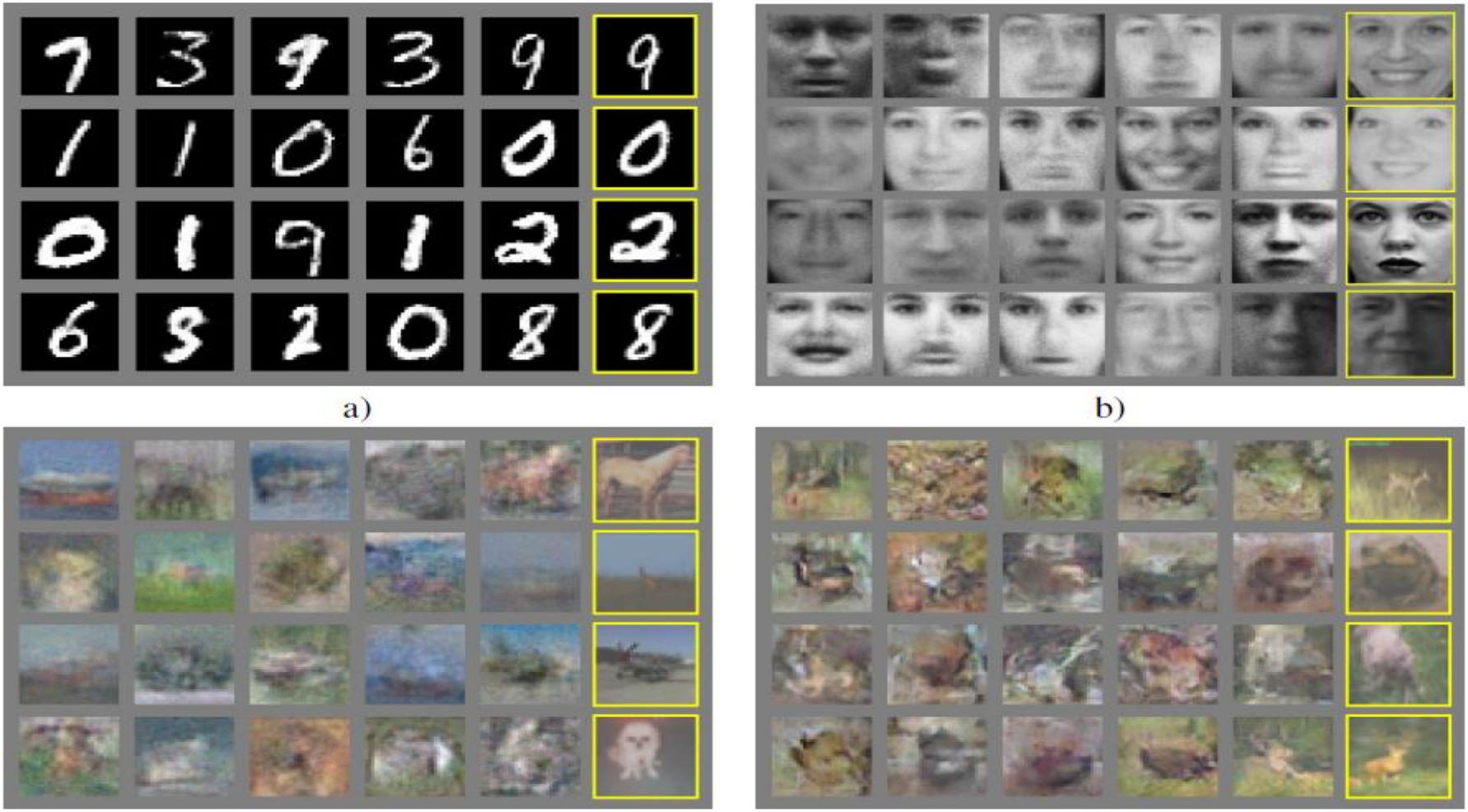
# Applications

## Google's Neural Machine Translation

...reduced translation errors by an average of 60% when compared to the prior Google Translate technology

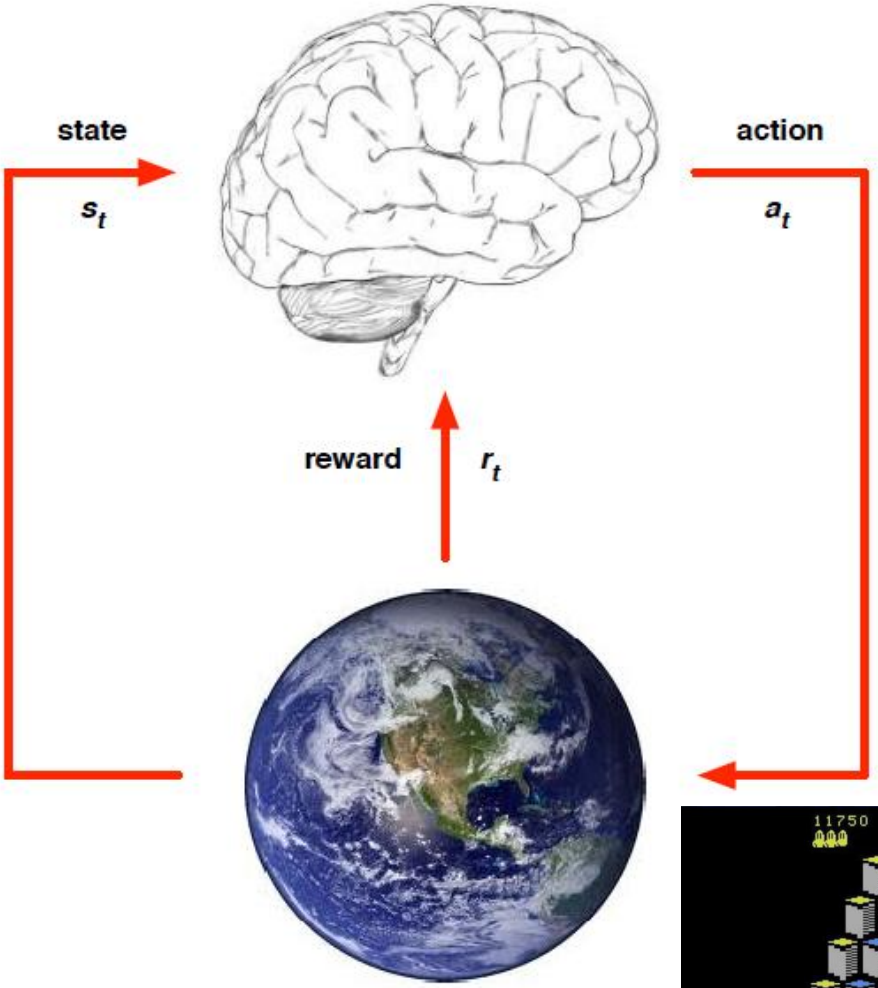


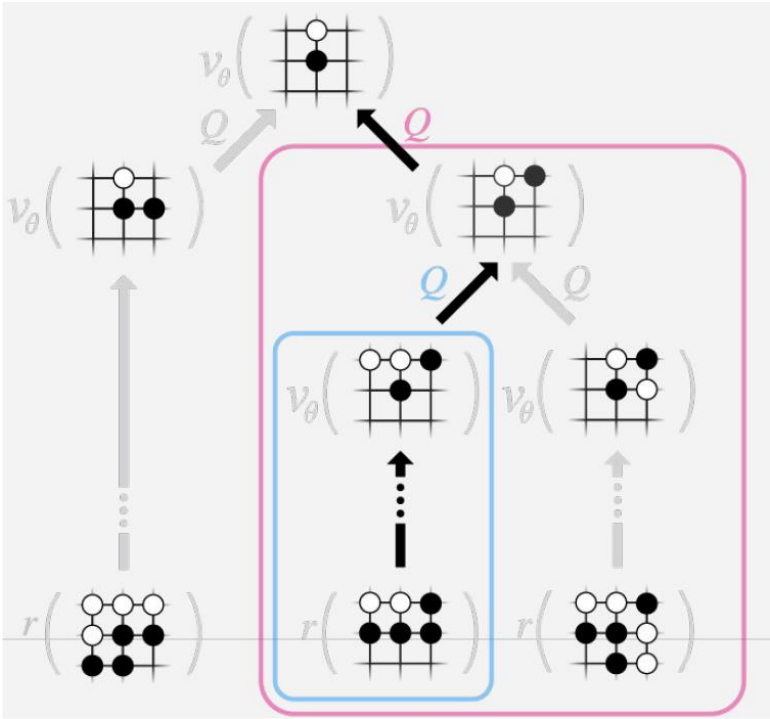
# Generative Adversary Nets



a) b) c) d)

# Deep Reinforcement Learning

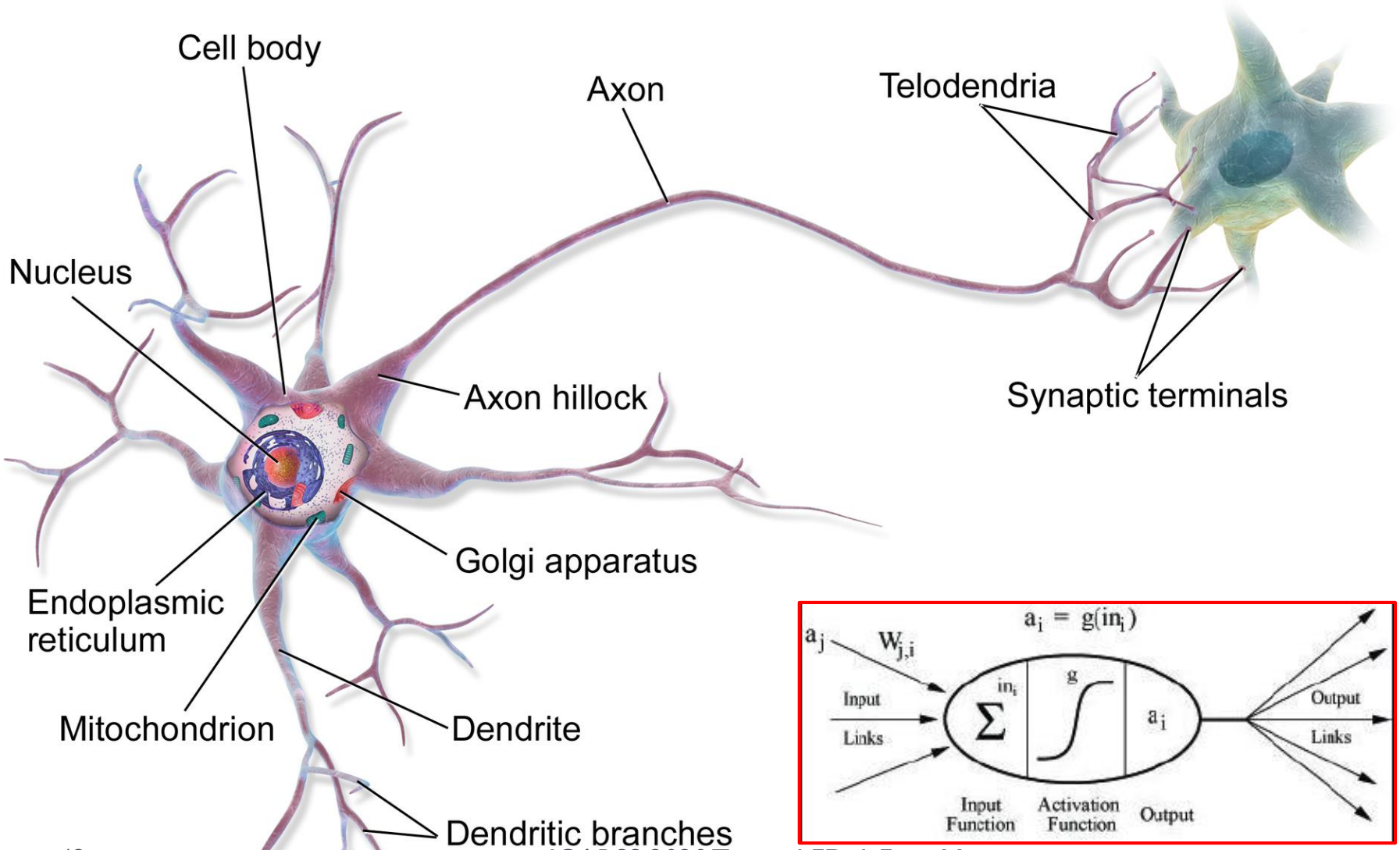


[Silver 2016]

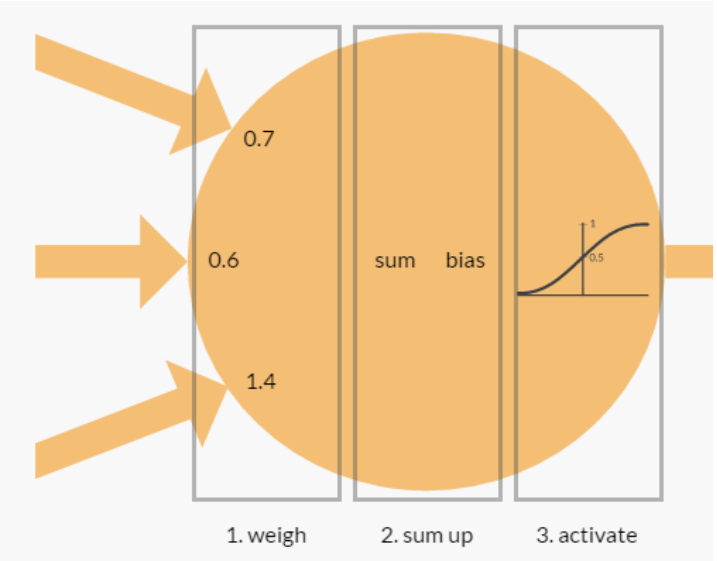
# Deep Neural Networks

# Neurons




[from wikipedia]

# Neuron Modeling




$$z = b + \sum_i x_i w_i$$

bias

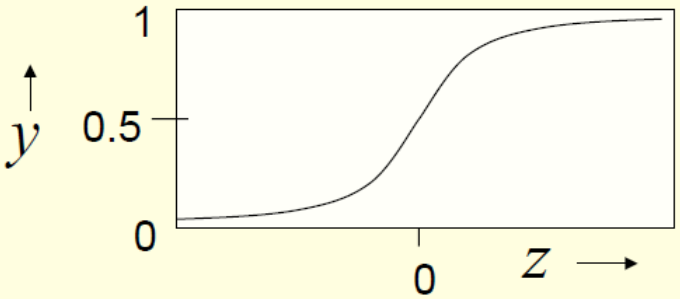
$i^{th}$ input

output

index over input connections

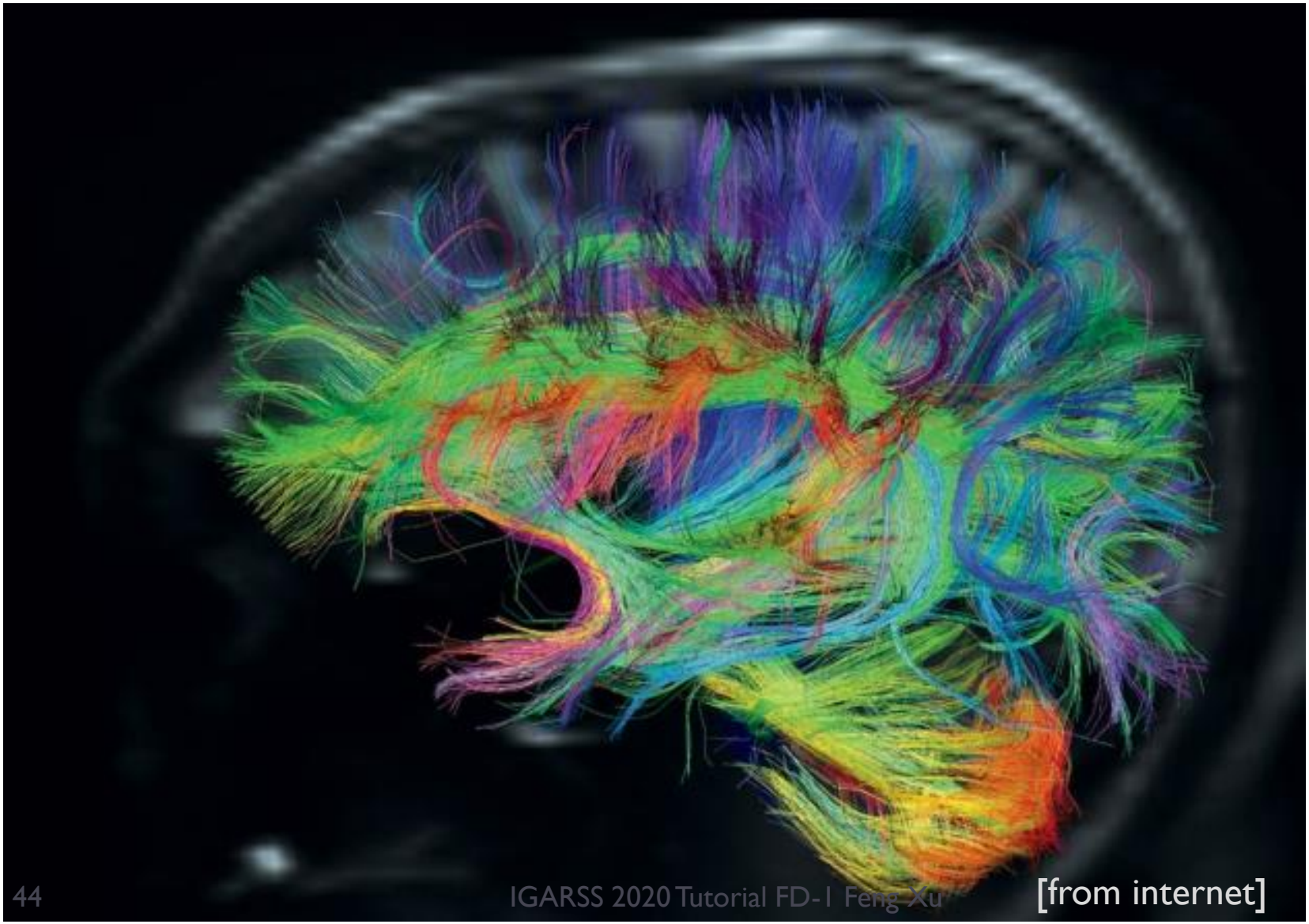
weight on $i^{th}$ input

$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1+e^{-z}}$$



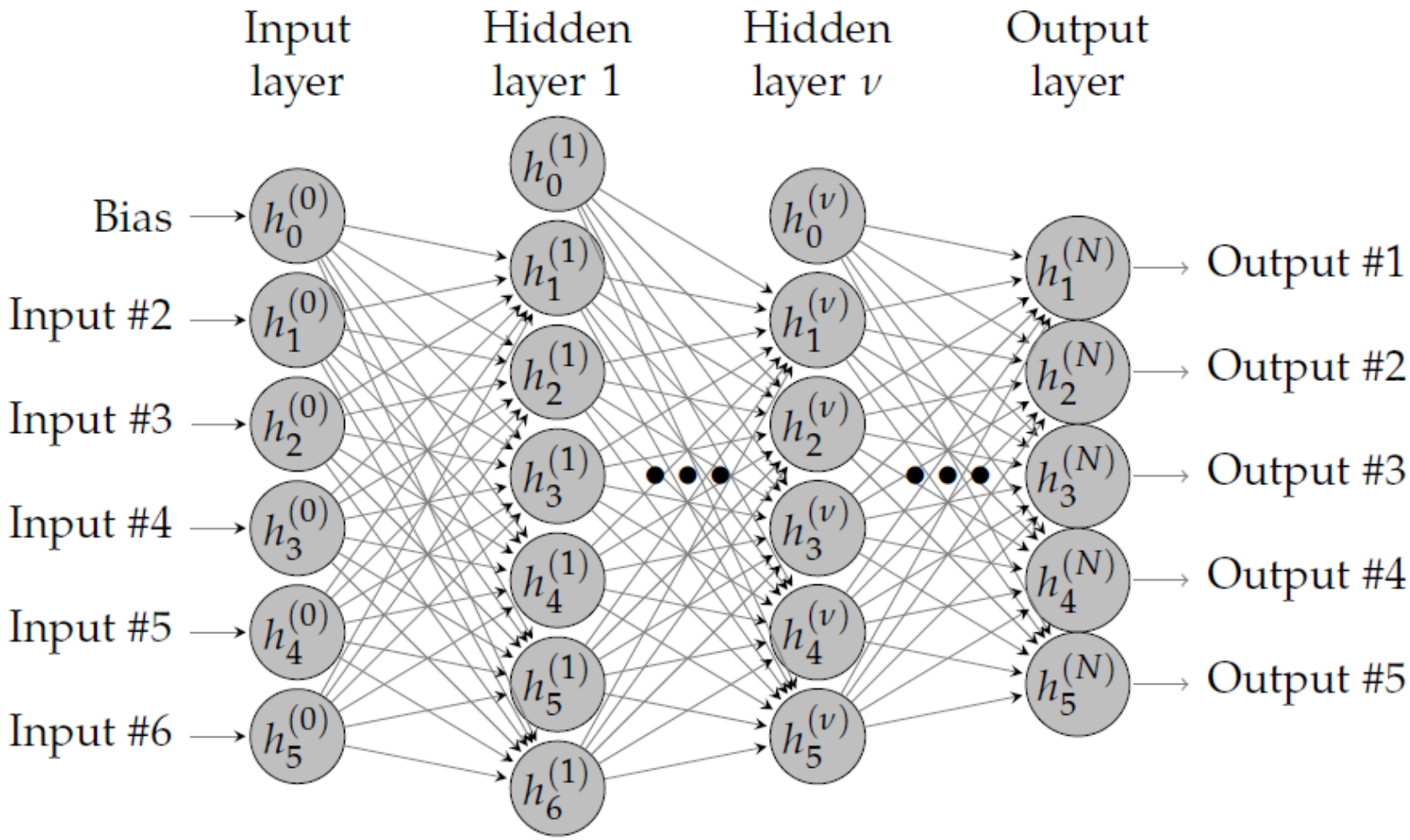

Sigmoid Activation

# Neural Network Modeling?



[from internet]

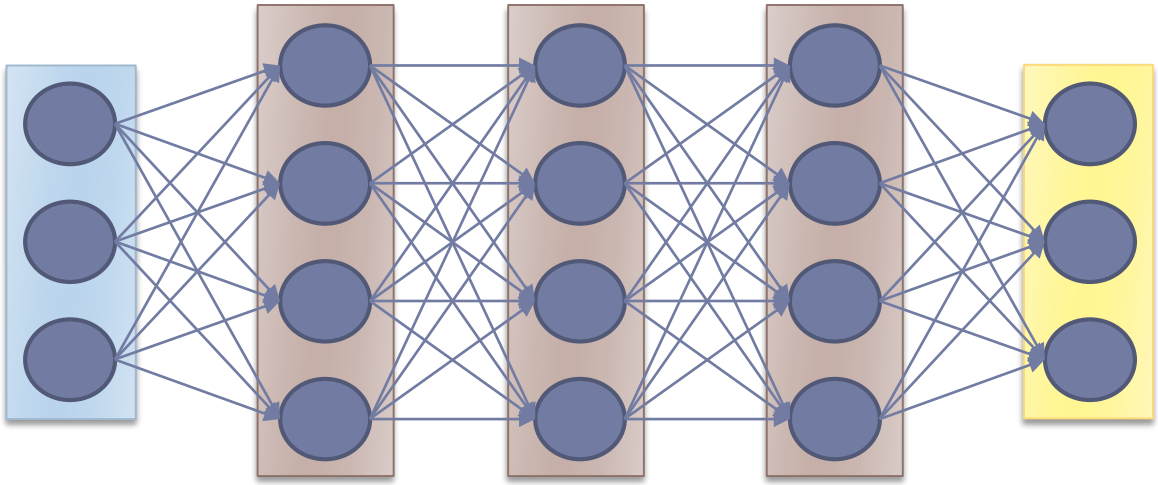
# Feedforward NN（Fully-connected）




[Epelbaum 2017]

# Feedforward NN（Fully-connected）

Data



Weights

Output

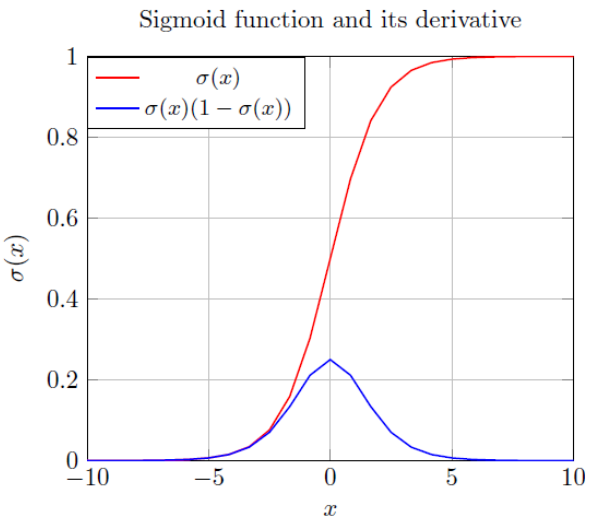
Label

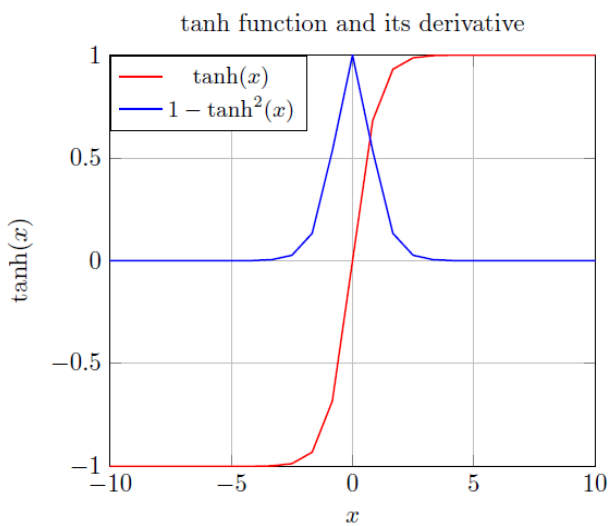
Objective

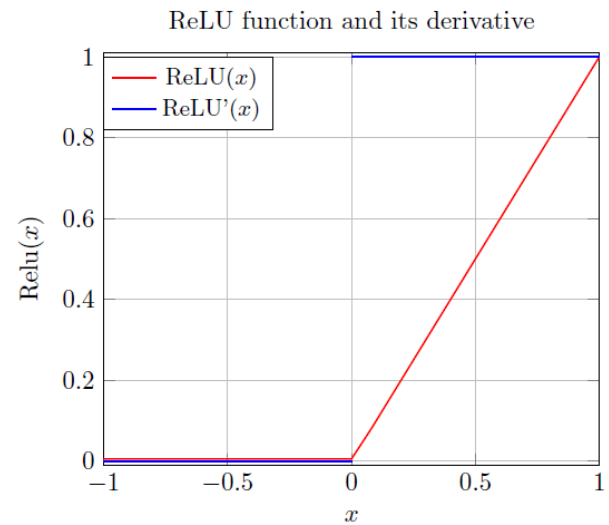
$$\begin{cases} y = f(x|w) \\ w = \arg\min_{w} L(y, z) \end{cases}$$

# Basic concepts

- Neuron
  - Activation Function




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

$$g(x) = \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

$$g(x) = \mathrm{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

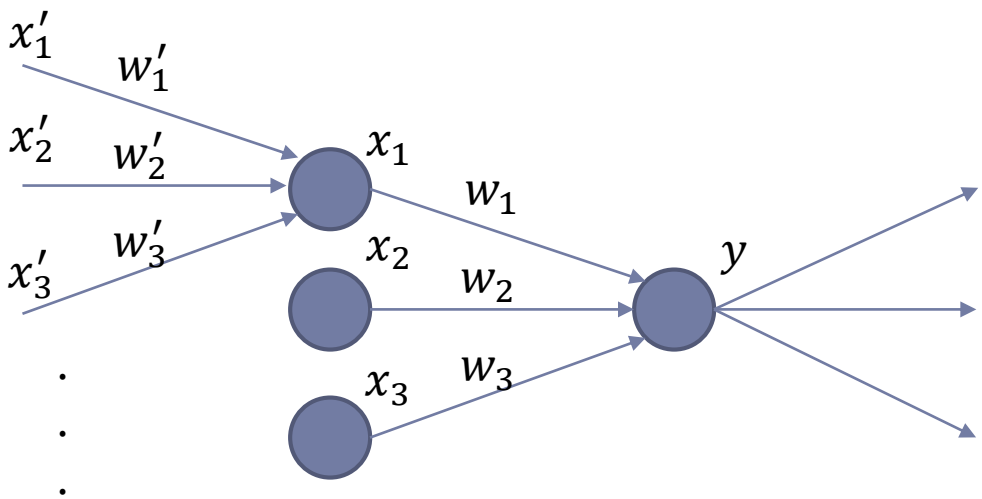
$$\sigma'(x) = \sigma(x)\,(1-\sigma(x))$$

$$\tanh'(x) = 1 - \tanh^2(x)$$

$$\mathrm{ReLU}'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

# Basic concepts

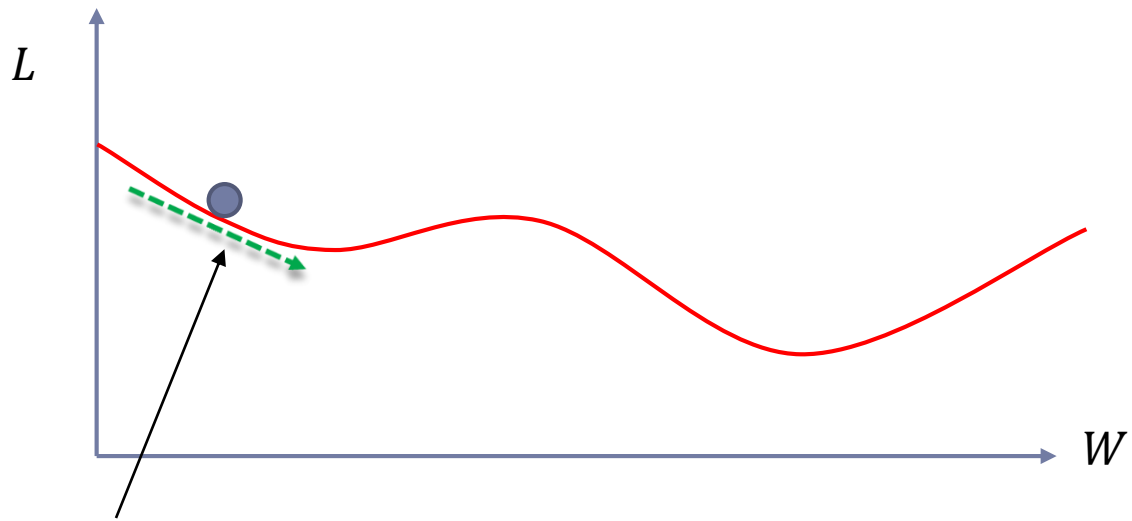
- Network Layer
  - Input Layer: Data Preprocessing
  - Hidden Layer: Computation Graph
  - Output Layer: Result (input to the loss function)




$$y = f(\vec{W} \cdot \vec{X}) = f\left(\vec{W} \cdot \overrightarrow{f(\vec{W}' \cdot \vec{X}')}\right) = \cdots$$

# Basic concepts

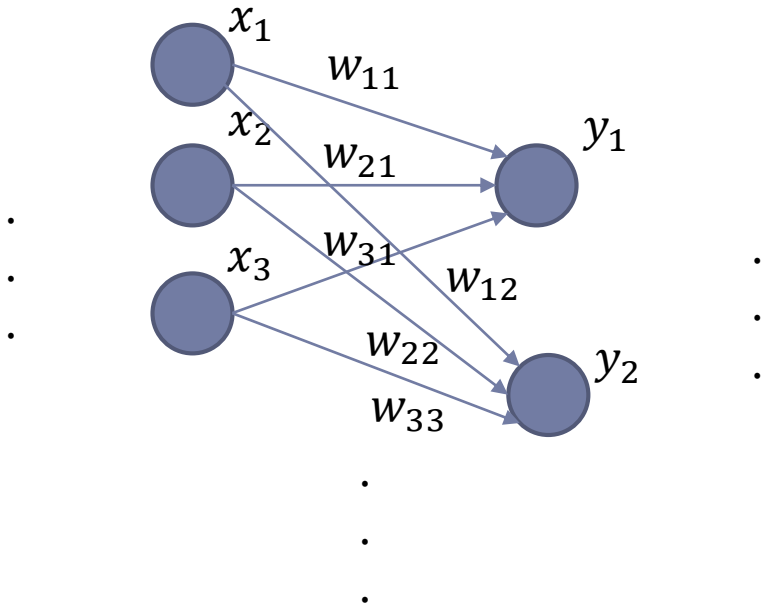
- Loss Function
  - $L(W, X, Y^*) = Distance(Y, Y^*) + Regularization(W)$
  - $Y = f(W, X)$
- Training Algorithm（Optimization）
  - Gradient Descent




How to compute the gradient：Backpropagation

# Fundamentals: Matrix Algebra

- Matrix Multiplication




$$y_j = f\left(\sum w_{ij} x_i\right)$$

$$\begin{bmatrix} \vdots \\ y_j \\ \vdots \end{bmatrix} = f\left(\begin{bmatrix} \ddots & \ddots & \ddots \\ \ddots & w_{ij} & \ddots \\ \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ x_i \\ \vdots \end{bmatrix}\right)$$

- Matrix Calculus
  （https://en.wikipedia.org/wiki/Matrix_calculus）

$d(\mathbf{x}^T\mathbf{A})/d(\mathbf{x}) = \mathbf{A}$

$d(\mathbf{A}\mathbf{x})/d(\mathbf{x}^T) = \mathbf{A}$

$d(\mathbf{a}^T\mathbf{X}\mathbf{b})/d(\mathbf{X}) = \mathbf{a}\mathbf{b}^T$

$d(\mathbf{x}^T\mathbf{A}\mathbf{x})/d(\mathbf{x}) = (\mathbf{A}+\mathbf{A}^T)\mathbf{x}$

$d(\mathbf{a}^T\mathbf{X}^T\mathbf{X}\mathbf{b})/d(\mathbf{X}) = \mathbf{X}(\mathbf{a}\mathbf{b}^T + \mathbf{b}\mathbf{a}^T)$

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

$\mathbf{J}=d(\mathbf{y})/d(\mathbf{x}^T)$

$d[d(f)/d(\mathbf{x})]/d(\mathbf{x})$

# Fundamentals: Chain Rule

$$\frac{\partial f[g(x)]}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Width

$$y_j = g([x_i]), z_k = f([y_j])$$

$$\frac{\partial z_k}{\partial x_i} = \sum_j \frac{\partial z_k}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

$$\left[\frac{\partial z_k}{\partial x_i}\right]_{k\times i} = \left[\frac{\partial z_k}{\partial y_j}\right]_{k\times j} \cdot \left[\frac{\partial y_j}{\partial x_i}\right]_{j\times i}$$

Depth

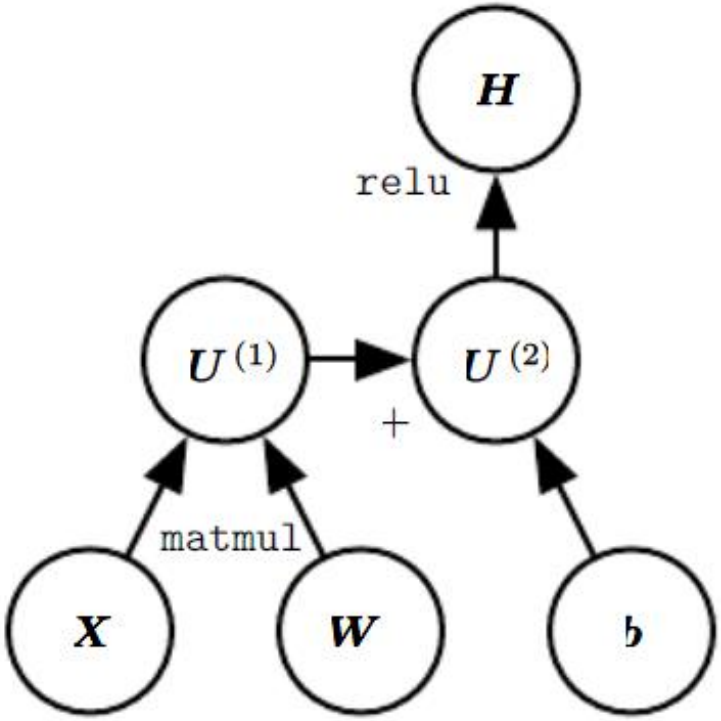
$$\frac{\partial h\{f[g(x)]\}}{\partial x} = \frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

$$y_j = g([x_i]), z_k = f([y_j]), v_n = h([z_k])$$

$$\left[\frac{\partial v_n}{\partial x_i}\right]_{n\times i} = \left[\frac{\partial v_n}{\partial z_k}\right]_{n\times k} \cdot \left[\frac{\partial z_k}{\partial y_j}\right]_{k\times j} \cdot \left[\frac{\partial y_j}{\partial x_i}\right]_{j\times i}$$

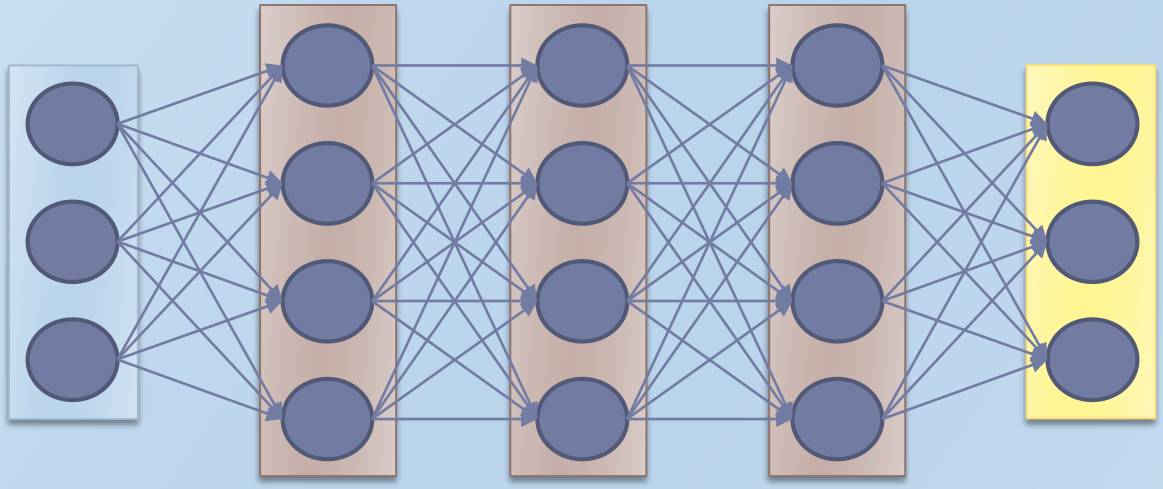
# Fundamentals: Computational Graph

- Use graph to represent sequential computation
- Node: Variables (Scale, Vector, Matrix, Tensor)
- Edge: Computation

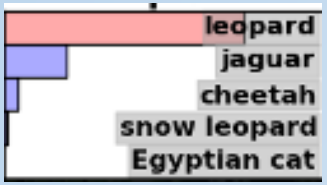



$$H = ReLU(W \cdot X + b)$$

Data



Weights

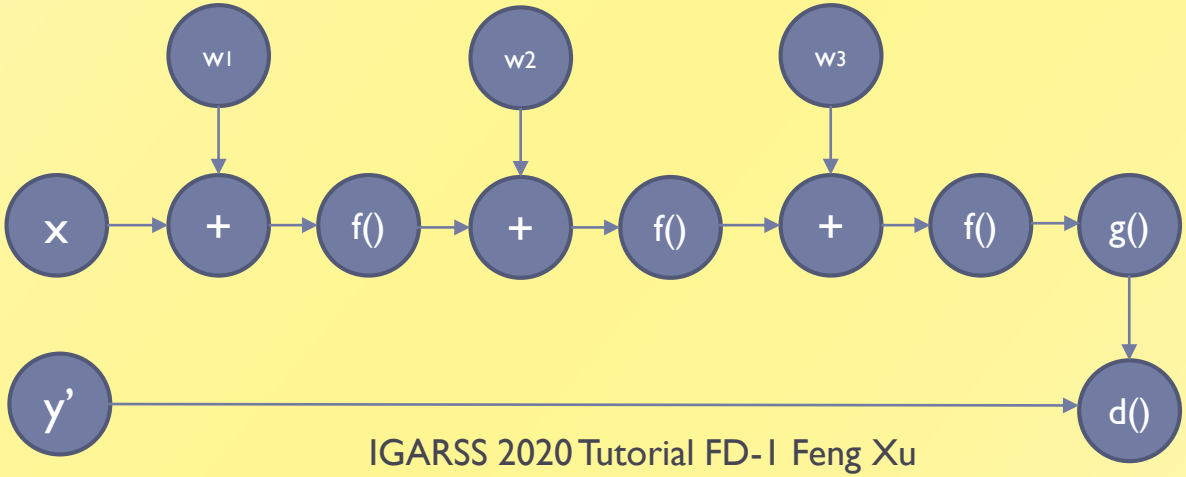



Output

Label

Objective

Weights




Data

Label

Error

# How to solve problems with DNN?

**Step1. Design DNN for specific Problems**

- CNN、RNN、FNN、R-CNN、GAN、ResNet
- Loss Function、Regularization

**Step2. Use data to train DNN**

- SGD+BP、Momentum、Adam、Dropout、BatchNorm

**Step3. Testing and analysis**

- Underfitting/Overfitting
- Data Augmentation
- Cross-Validation

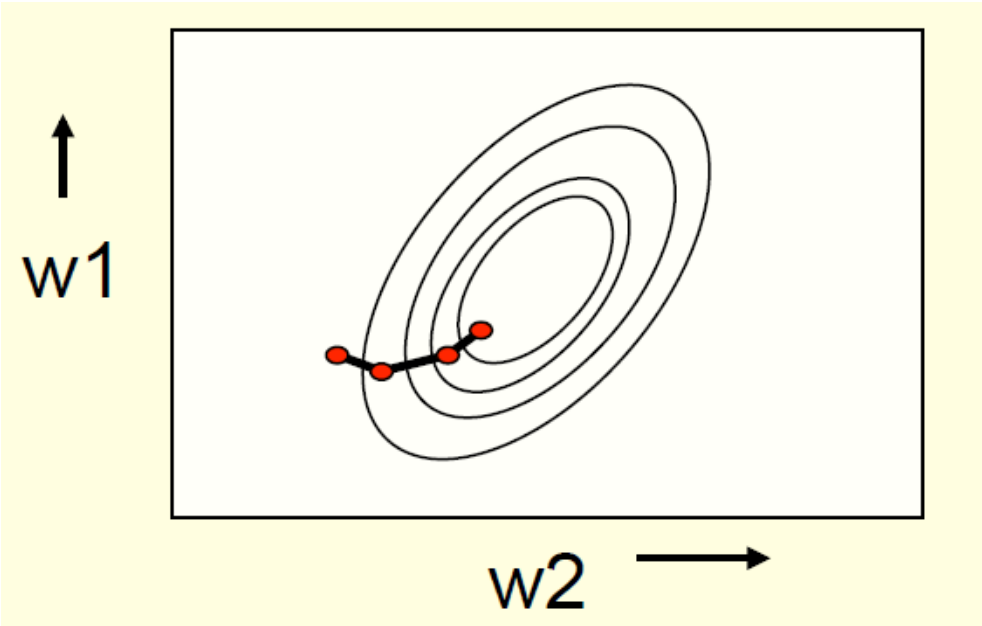
# Step2: Use data to train DNN

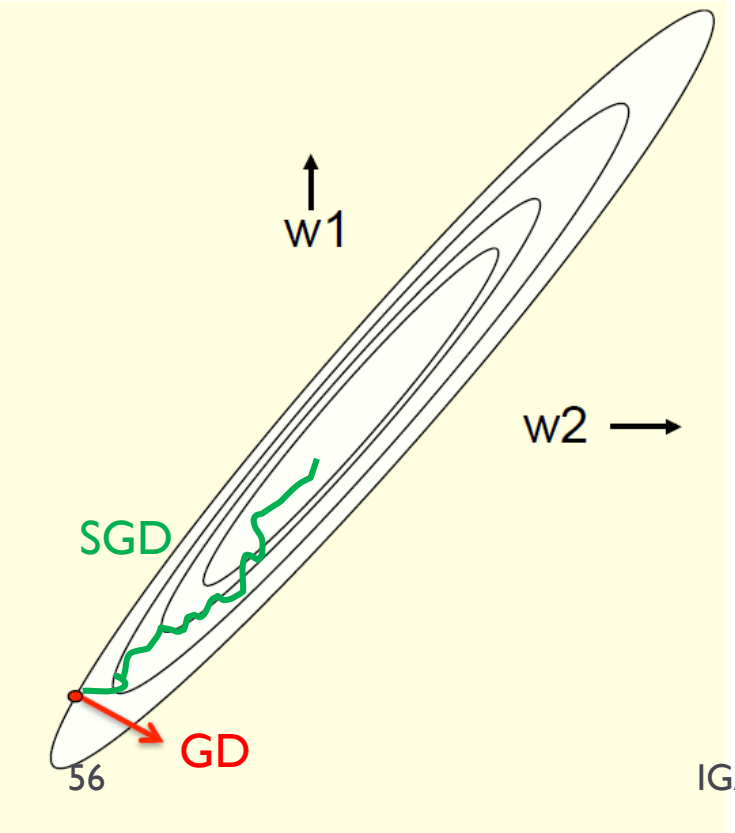
- Loss Function
  - Regression： Continuous value
    - $L(y,z) = \|y - z\|_2^2$
  - Classification： Discrete labels
    - $L(y,z) = \sum_i z_i \log y_i\,,\ 1 = \sum_i z_i = \sum_i y_i$
- Learning → Training → Fitting → Optimization

| Method | Optimum | Gradient | Converge |
|---|---|---|---|
| Gradient Descent | Local ✓ | 1st order ✓ | Middle ✓ |
| Newtonian | Local | 2nd order ✗ | Fast |
| Inspired: SA, GA | Global ✗ | None | Slow ✗ |

# Learning = Training = Optimization

- Gradient Descent
- $\Delta w_j = -\delta \mathrm{E}_x \left( \frac{\partial L}{\partial w_j} \right)$




- Stochastic GD
- $\Delta w_j = -\delta \mathrm{E}_{x'} \left( \frac{\partial L}{\partial w_j} \right)$
- 1.Low Complexity,
- 2.More Robust

# How to implement SGD?

- We have:
  - N training samples
  - A defined network
  - A defined loss function
- We have to:
  - Compute gradients
  - Update weights
- SGD：
  - Use mini-batch data to compute gradients and update weights

$$L(\boldsymbol{x}, \boldsymbol{y}^*, W) = \frac{1}{N}\sum |y(x_i) - y_i^*|^2 = \frac{1}{N}\sum L_i(x_i, y_i^*, W)$$

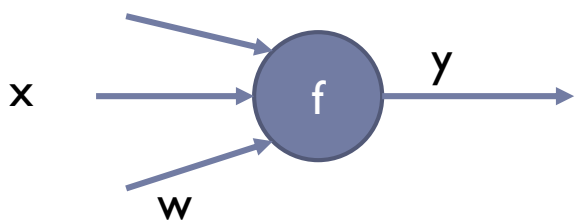
$$\frac{\partial L(\boldsymbol{x}, \boldsymbol{y}^*, W)}{\partial W} = \frac{1}{N}\sum \frac{\partial L_i(x_i, y_i^*, W)}{\partial W} \approx \frac{1}{N_{minibatch}} \sum_{minibatch} \frac{\partial L_i(x_i, y_i^*, W)}{\partial W}$$

# How to compute gradients?

- Chain Rule： $$\frac{\partial f[g(x)]}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

- Loss function： $$\frac{\partial L(y,z)}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w} = L'(y,z) \cdot \frac{\partial y}{\partial w}$$

- One layer： $$L' \cdot \frac{\partial y}{\partial w} = L' \cdot \frac{\partial f(w^{\mathrm{T}} \cdot x)}{\partial w} = L' \cdot f'\left(w^{\mathrm{T}} \cdot x\right) \cdot x$$



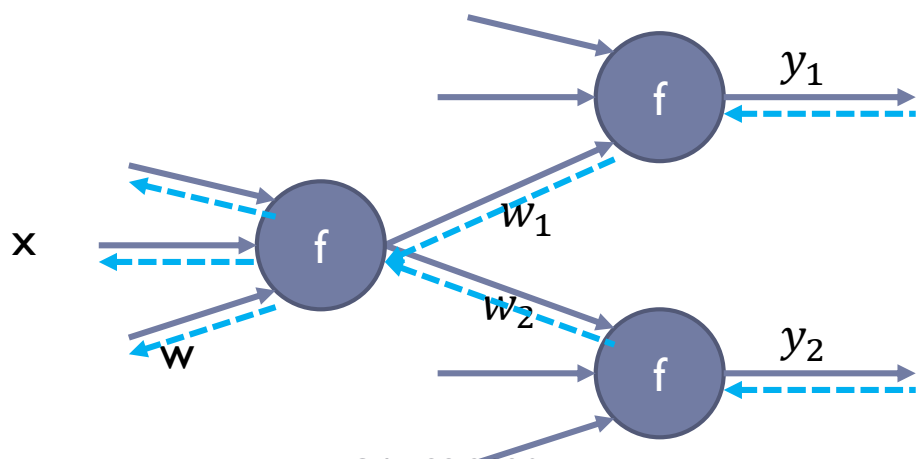
# How to compute gradients?

- Multiple layers

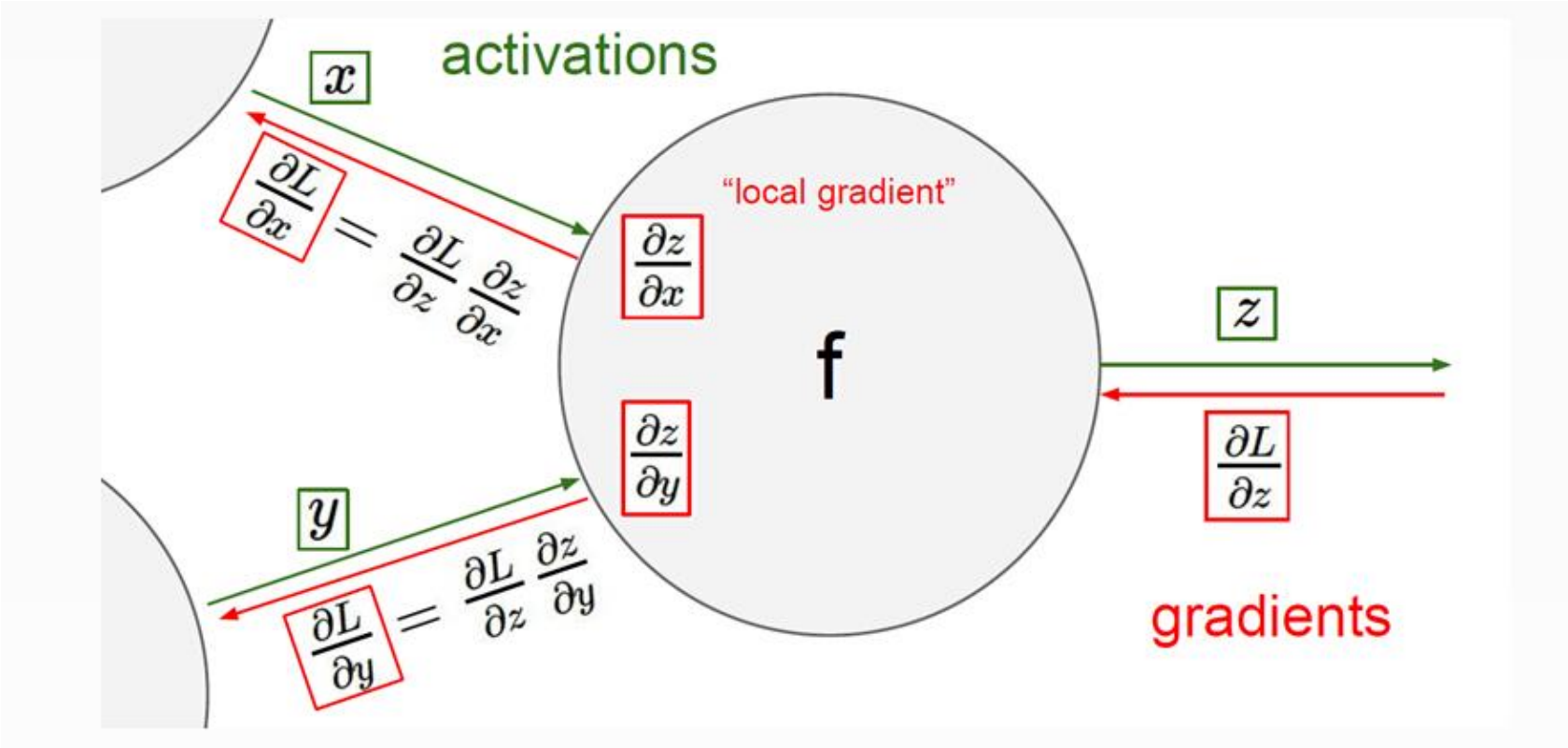
$$L(y_1, y_2) = L\{f\left(w_1 f(w^T x)\right), f\left(w_2 f(w^T x)\right), \}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y_1} \cdot f_1' \cdot w_1 \cdot f' \cdot x + \frac{\partial L}{\partial y_2} \cdot f_2' \cdot w_2 \cdot f' \cdot x$$

$$\frac{\partial L}{\partial x} = \left\{\frac{\partial L}{\partial y_1} \cdot f_1' \cdot w_1 + \frac{\partial L}{\partial y_2} \cdot f_2' \cdot w_2\right\} \cdot f' \cdot x$$

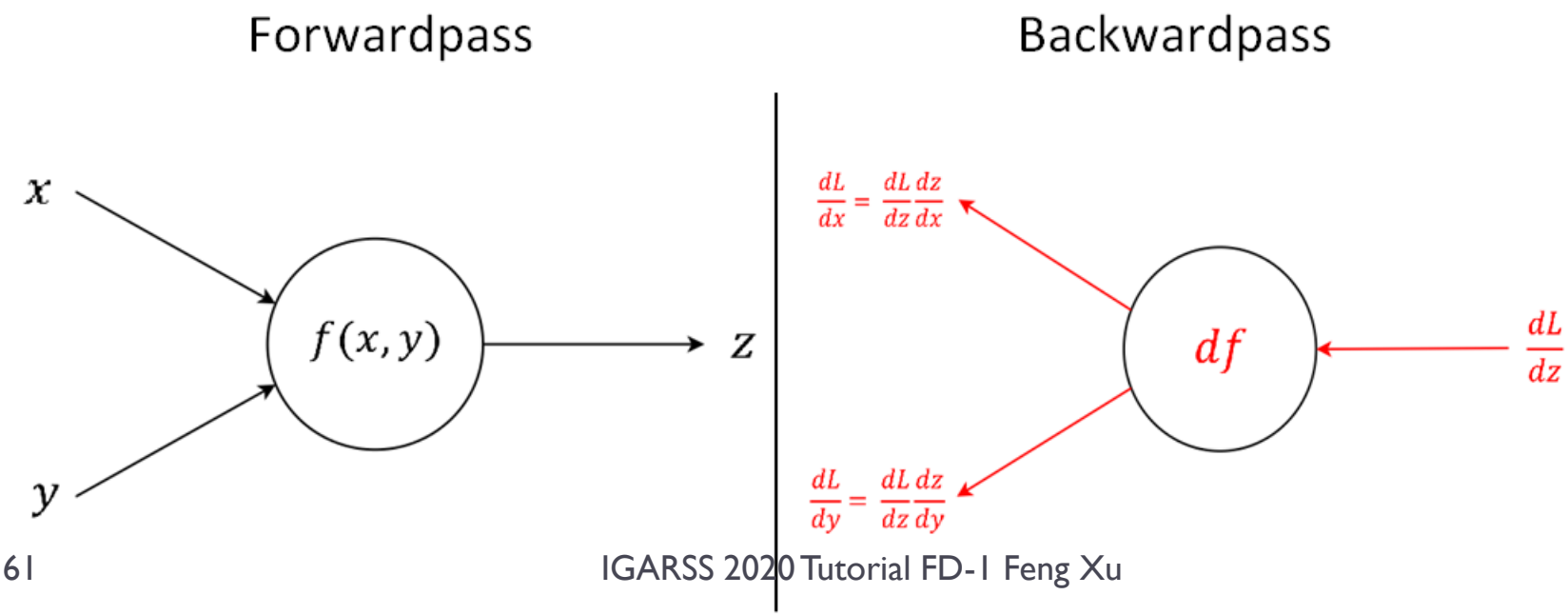


# Error back-propagation



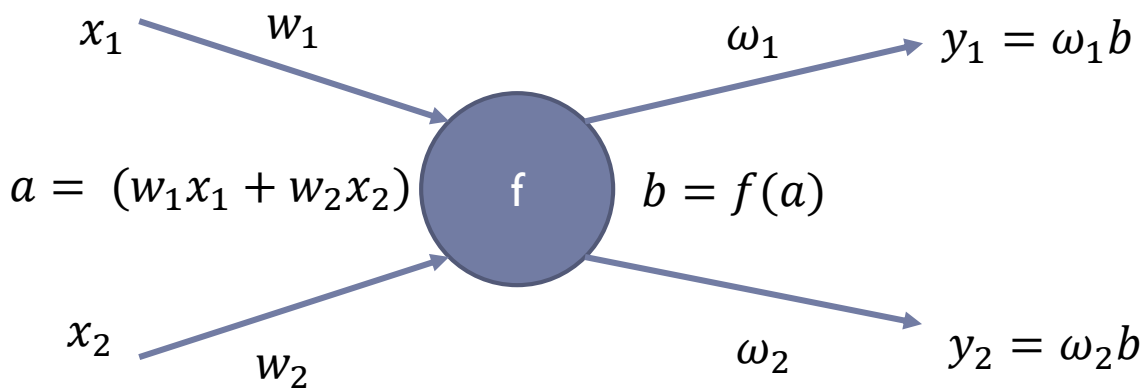
# Backpropagation

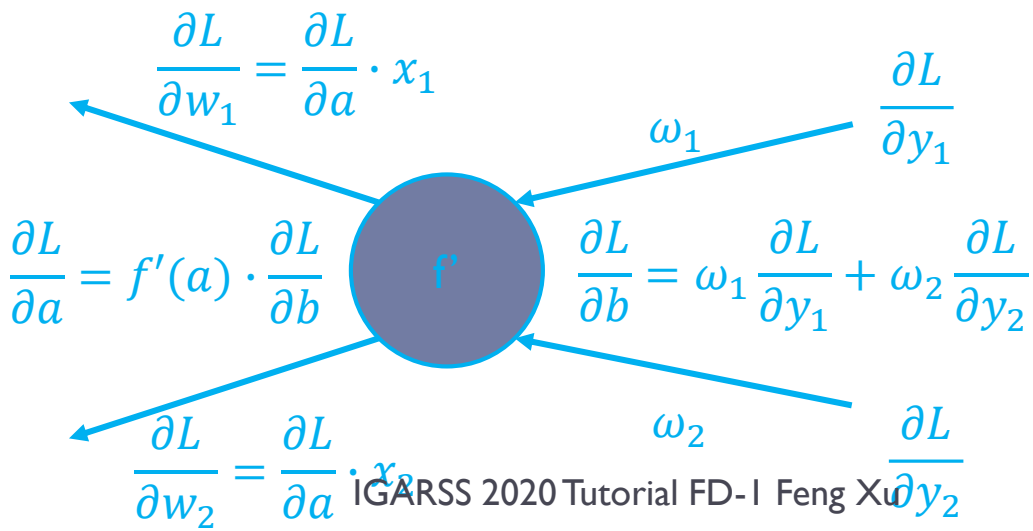
- Signal forward propagate: Forward-Pass
  - Forward computation
- Error back propagate: Backward-Pass
  - Error backpropagate, Adjust weight along the way
- Efficient gradient descent



# Backpropagation

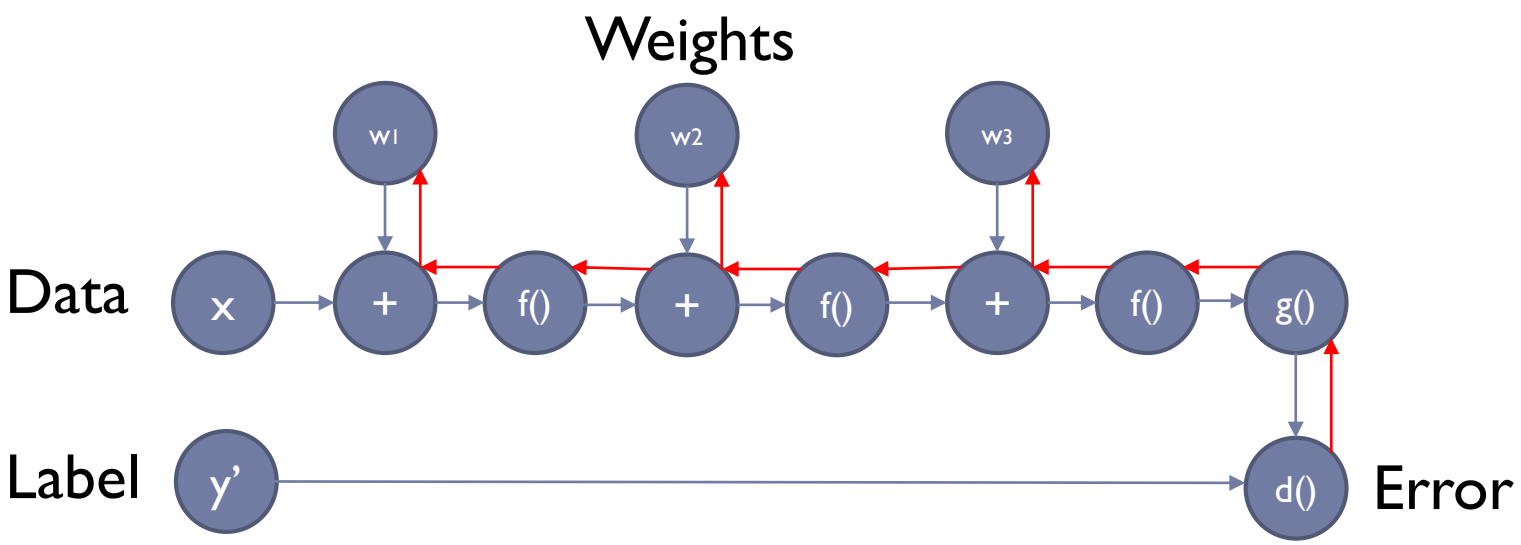
- Each neuron has to implement both forward pass and backpropagate function



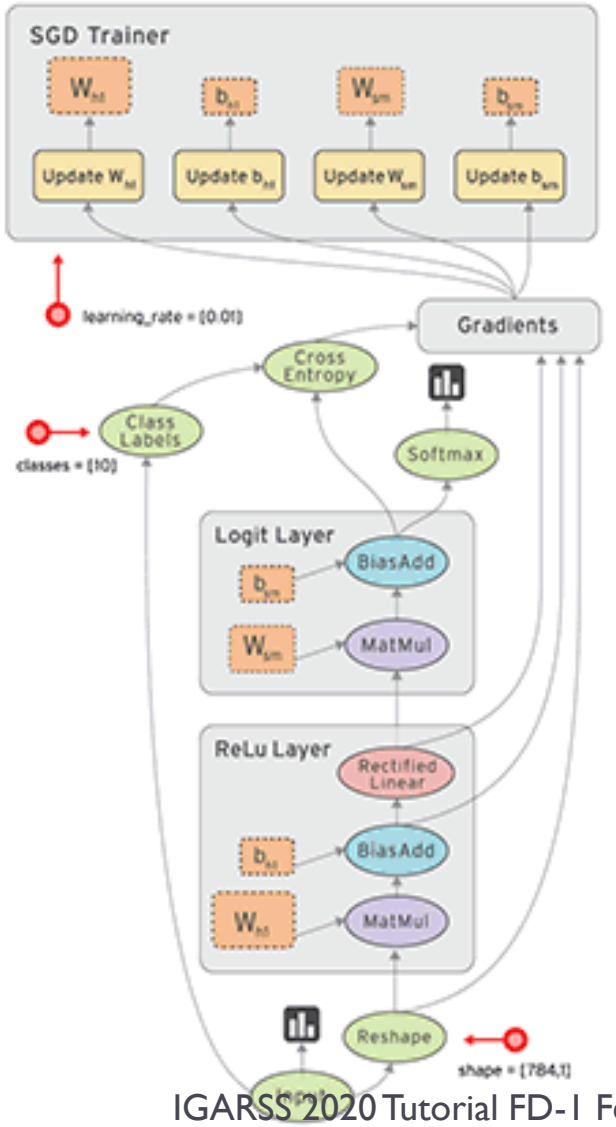



# Computational Graph

- Matrix Algebra + Computation Graph + Chain Rule



# “Tensor Flow”


SGD Trainer
Update W
Update b
Update W
Update b
learning_rate = [0.01]
Gradients
Cross Entropy
Class Labels
classes = [10]
Softmax
Logit Layer
BiasAdd
MatMul
ReLu Layer
Rectified Linear
BiasAdd
MatMul
Reshape
shape = [784,1]

# In-depth understanding of DNN

# How to solve problems with DNN?

**Step1. Design DNN for specific Problems**

- CNN、RNN、FNN、R-CNN、GAN、ResNet
- Loss Function、Regularization

**Step2. Use data to train DNN**

- SGD+BP、Momentum、Adam、Dropout、BatchNorm

**Step3. Testing and analysis**

- Underfitting/Overfitting
- Data Augmentation
- Cross-Validation

# Step1. How to design DNN?

- CNN, RNN, DQN, ResNet, etc…
  - Problem Specific
  - Incorporating Prior Knowledge of Human
  - Will talk in later chapters
- Must Be Deep!!!

# Why Deep Learning is taking off?

- Big Data

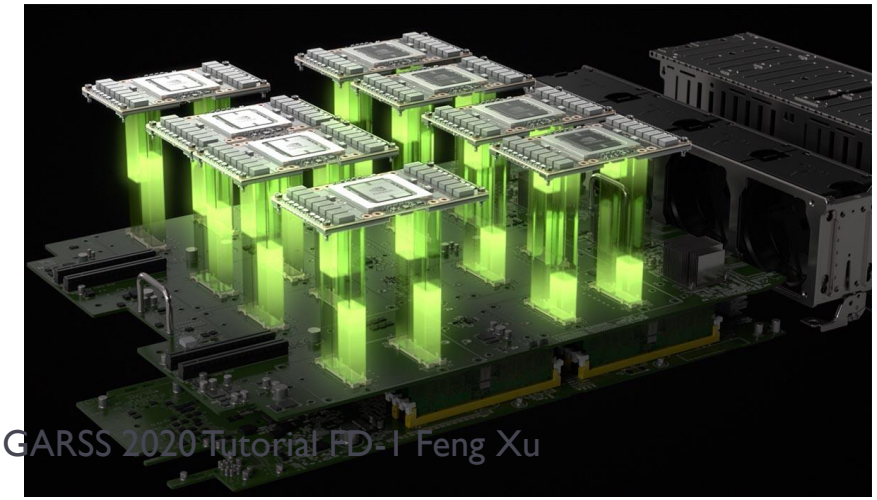
**ImageNet Dataset**

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*. [web]

3

- Powerful Computer

**Nvidia DGX-1**



- Algorithm Improvement

# Algorithmic Evolution

Perceptron (1950s)

Multi-Layer Perceptron (Feedfowar d)

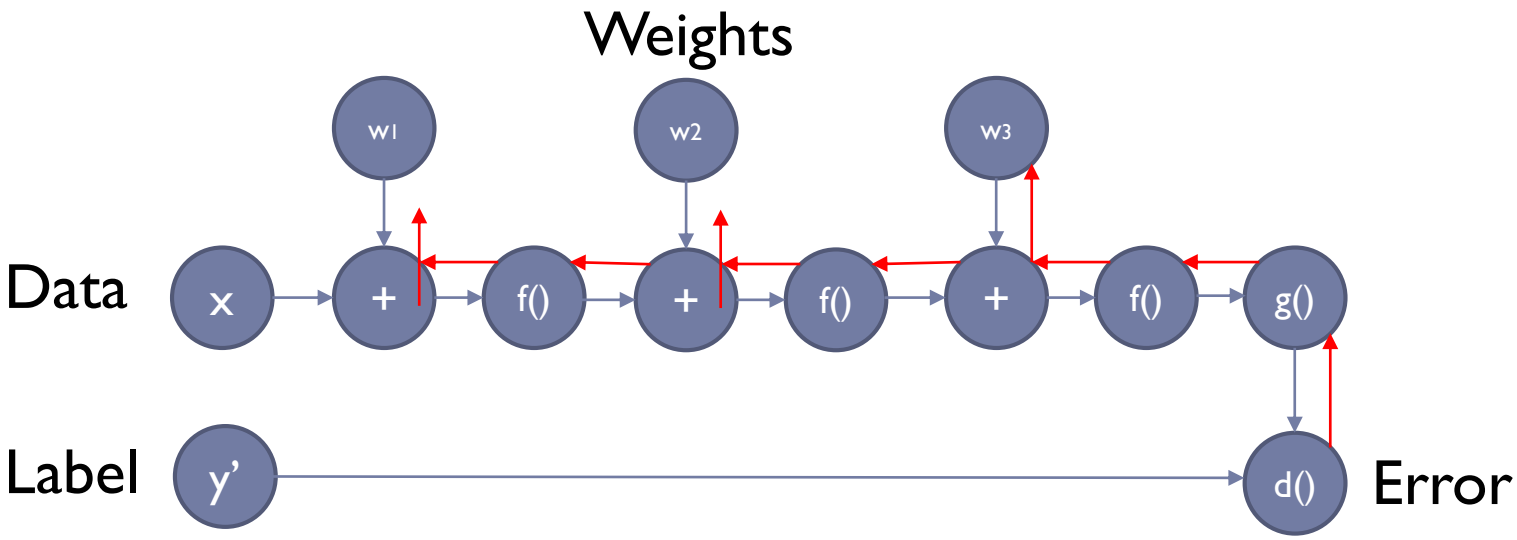
BackProp in Neural Networks (1986)

Deep Neural Networks (2006, 2012, 2016)

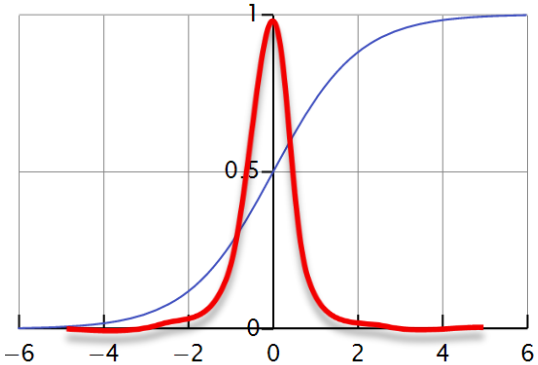
- Deep Representation
- Activation Function
- Loss Function
- Network Architecture

# Deep Architecture

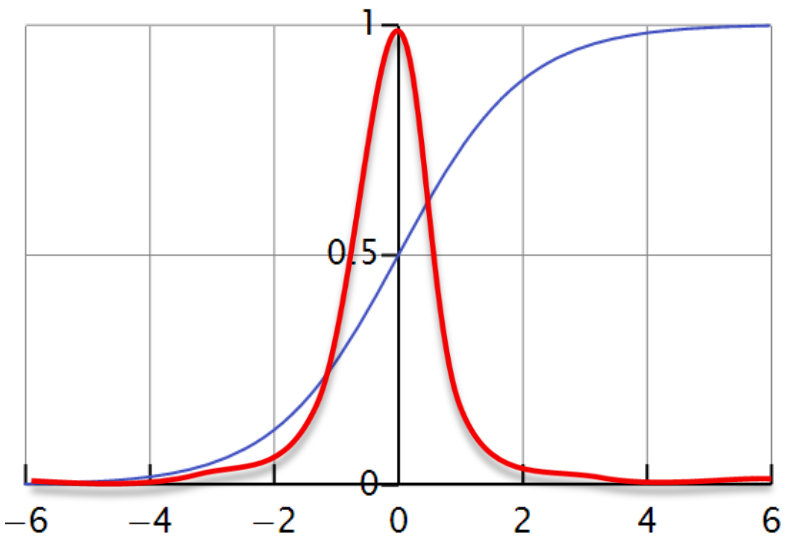
- Why not deep at the beginning?
- Gradient becomes unstable after backpropagation

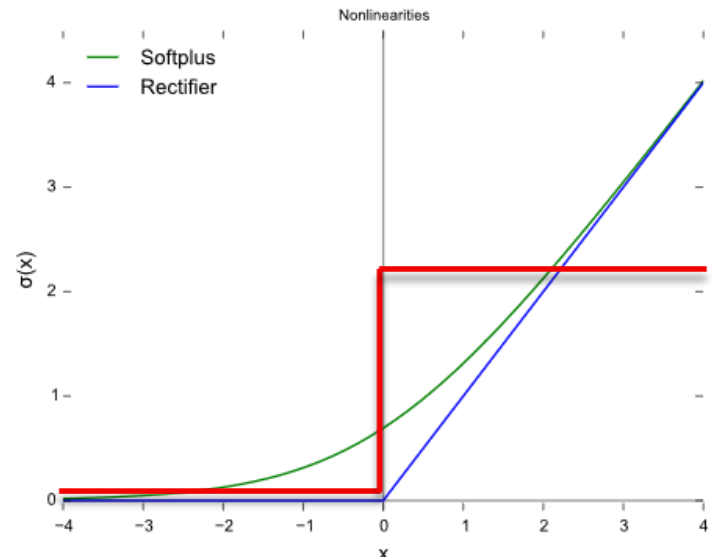



$$\frac{\partial L}{\partial w_1} \propto g' \cdot f' \cdot f' \cdot f'$$



# ReLU: Improved Activation Function




Sigmoid

- Continuous Differentiable
- Nonlinearity

- Gradient Saturation
- Gradient Exaggeration

ReLU

- Non-differentiable
- Approximate Nonlinearity

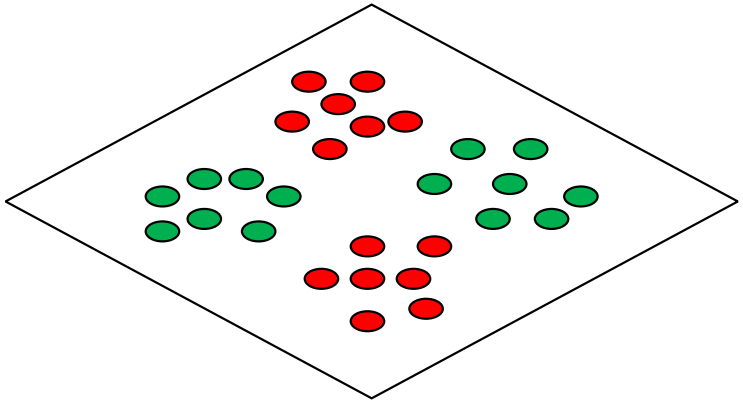
- Stable Gradient
- Sparsity

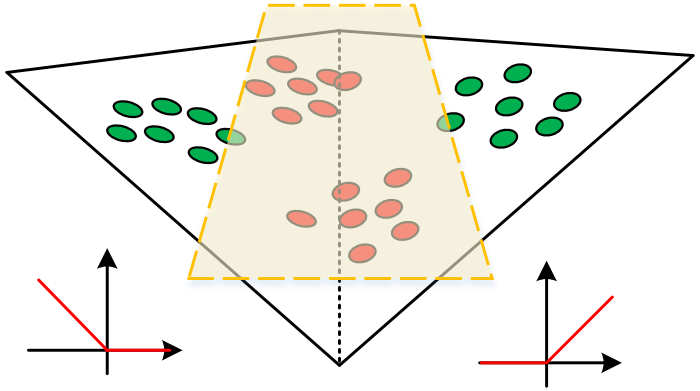
# On ReLU: Necessity of Nonlinearity

- Mutual Terms

$$f(W \cdot X) \approx f_0 + f_0' \cdot W \cdot dX + f_0'' \cdot \underbrace{(W \cdot dX)^2 + \cdots}_{\text{High-order Terms}}$$

- ReLU: Piecewise Linear





# On ReLU： how it was invented?

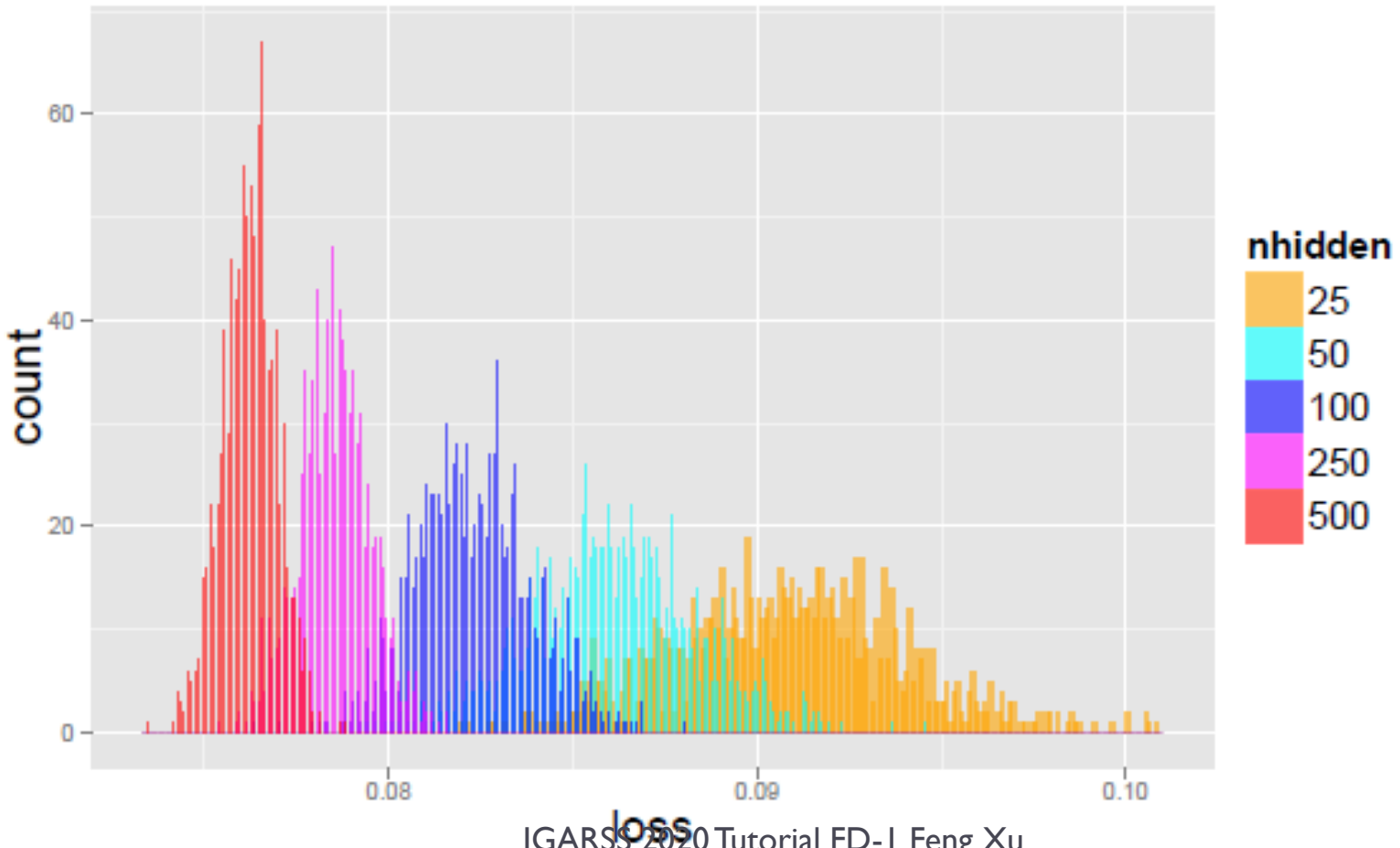
- Rectifications such as max(0,x) was proposed early Fukushima (1975)
- Rectifications replaced by sigmoid in 1980s.
- In 2000s, it is believed that non-differentiable points should be avoided.
- Jarrett et al. (2009) observed that rectification is the most important factor of deep networks.
- Glorot et al. (2010, 2011) showed that rectified linear unit (ReLU) should be used in deep networks.
- AlexNet (2012) achieved milestone success based on ReLU
- Choromanska et al. (2015) proved that deep rectified network has a band of 'optimal' local optimum, and can be easily recovered by SGD.

# On ReLU：Biological Evidences

- Glorot et al. (2011) : ReLU bears similarity with neurons
  - Neurons not respond to some input at all
  - Neurons respond to some input proportionally
  - Sparse Activation: most neurons are not activated at any particular moment

# On ReLU：
# Local Optimum vs. Global Optimum

- Choromanska et al. 2015: “The Loss Surfaces of Multilayer Networks”



# On ReLU：
# Local Optimum vs. Global Optimum

- Choromanska et al. 2015: “The Loss Surfaces of Multilayer Networks”
  - The lowest critical values of the random loss function located in a band lower-bounded by the global minimum.
  - The number of local minima outside that band diminishes exponentially with the size of the network.
  - All critical points are local minima of high quality measured by the test error.
  - Both simulated annealing and SGD converge to the band of low critical points.
  - Recovering the global minimum becomes harder as the network size increases but it is not necessary as global minimum often leads to overfitting.

# Improved Loss Function：Softmaxloss

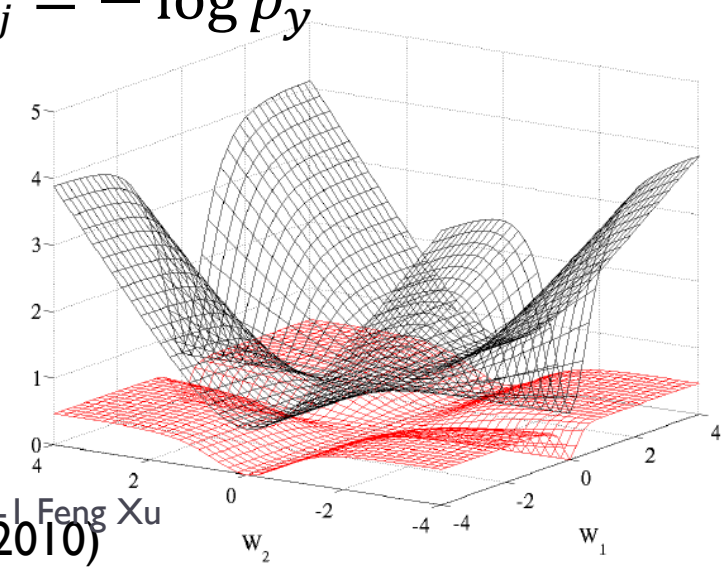
- For Classification Task
- Output A Posterior Probability:

$$p_i = \frac{\exp\left(O_i^{(L)}\right)}{\sum_{j=1}^{K} \exp\left(O_j^{(L)}\right)} \quad (i = 1, \cdots, K)$$

- Cross-Entropy: minimizing the discrepancy between the ground truth $y$ and the network prediction $\boldsymbol{p}$

$$L(w) = \sum_{j=1}^{K} -\delta(y-j)\log p_j = -\log p_y$$

Gradient of Loss Function： $\delta(y-j) - p_j$

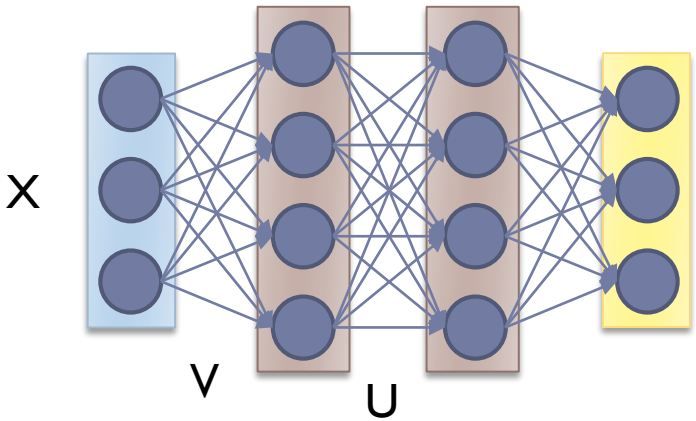



Glorot et al. (2010)

# Improved Network Architecture

- Special Layers（Conv、Pooling、LSTM）
- Special Structure（ResNet）
- Depth：Must Be Deep!!!

# Why Deep?




$$Y = f(U \cdot f(V \cdot X))$$

$$Y \approx U \cdot V \cdot X$$

$$Y = W \cdot X$$

$$W_{p\times n} = U_{p\times q} \cdot V_{q\times n}$$

$$p \cdot n > q(p+n)$$

- q<n、p: Low-rank matrix factorization
- Structured high-dimensional data is always distributed on a low-dimension manifold

# Why Deep?

- Compositionality
  - Just as human languages exploit compositionality to give representations and meanings to complex ideas
  - (1) Distributed representations / embeddings: feature learning
  - (2) Deep architecture: multiple levels of feature learning

# Why Deep?

- Use Prior Knowledge to Reduce Unknowns
- For Specific Applications, Use Domain Knowledge Enhance Deep Learning Algorithms
- This includes Network Designing, Data Preparation and Algorithm Designing
- The essence of DL algorithmic improvement is: Using prior knowledge!

# Prior Knowledge used in CNN

- We can put our prior knowledge about the task into the network by designing appropriate:
  - Connectivity. →e.g. Locally connected
  - Weight constraints. →e.g. Weight sharing
  - Neuron activation functions →e.g. Max pooling
- Alternatively, we can use our prior knowledge to create a whole lot more training data.
  - This may require a lot of work
  - It may make learning take much longer.
  - E.g. Cropping, Translation, Mirror, Scaling, Rotation, Noise Contamination, Distortion

# Basic Practice

# Entry-case MNIST

- MatConvnet
- Caffee
- TensorFlow
- Keras (TF/Theano)

# MNIST

- Modified National Institute of Standards and Technology database
- 0-9 digit recognition
- 50000 training sample
- 10000 test sample

| | |
|---|---|
| train-images-idx3-ubyte.gz: | training set images (9912422 bytes) |
| train-labels-idx1-ubyte.gz: | training set labels (28881 bytes) |
| t10k-images-idx3-ubyte.gz: | test set images (1648877 bytes) |
| t10k-labels-idx1-ubyte.gz: | test set labels (4542 bytes) |

# MNIST Ranking

| Type | Classifier | Distortion | Preprocessing | Error rate (%) |
|---|---|---|---|---|
| Linear classifier | Pairwise linear classifier | None | Deskewing | 7.6[9] |
| K-Nearest Neighbors | K-NN with non-linear deformation (P2DHMDM) | None | Shiftable edges | 0.52[17] |
| Boosted Stumps | Product of stumps on Haar features | None | Haar features | 0.87[18] |
| Non-linear classifier | 40 PCA + quadratic classifier | None | None | 3.3[9] |
| Support vector machine | Virtual SVM, deg-9 poly, 2-pixel jittered | None | Deskewing | 0.56[19] |
| Neural network | 2-layer 784-800-10 | None | None | 1.6[20] |
| Neural network | 2-layer 784-800-10 | elastic distortions | None | 0.7[20] |
| Deep neural network | 6-layer 784-2500-2000-1500-1000-500-10 | elastic distortions | None | 0.35[21] |
| Convolutional neural network | 6-layer 784-40-80-500-1000-2000-10 | None | Expansion of the training data | 0.31[14] |
| Convolutional neural network | 6-layer 784-50-100-500-1000-10-10 | None | Expansion of the training data | 0.27[15] |
| Convolutional neural network | Committee of 35 CNNs, 1-20-P-40-P-150-10 | elastic distortions | Width normalizations | 0.23[8] |
| Convolutional neural network | Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10 | None | Expansion of the training data | 0.21[16] |

# Keras

If you want to learn theano:

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html

TensorFlow or theano

Very flexible

Need some effort to learn

Interface of TensorFlow or Theano

keras

Easy to learn and use

(still have some flexibility)

You can modify it if you can write TensorFlow or Theano

# Convolutional Neural Network

# How to solve problems with DNN?

**Step1. Design DNN for specific Problems**

- CNN、RNN、FNN、R-CNN、GAN、ResNet
- Loss Function、Regularization

**Step2. Use data to train DNN**

- SGD+BP、Momentum、Adam、Dropout、BatchNorm

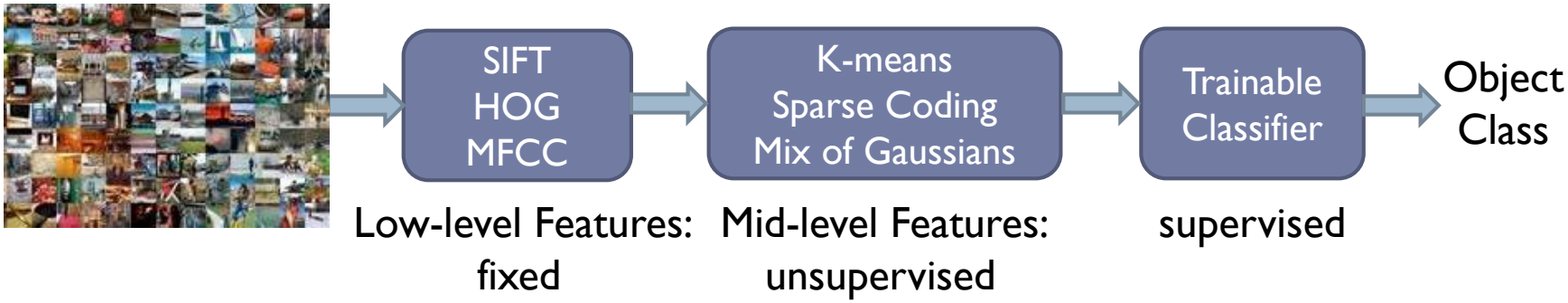
**Step3. Testing and analysis**

- Underfitting/Overfitting
- Data Augmentation
- Cross-Validation

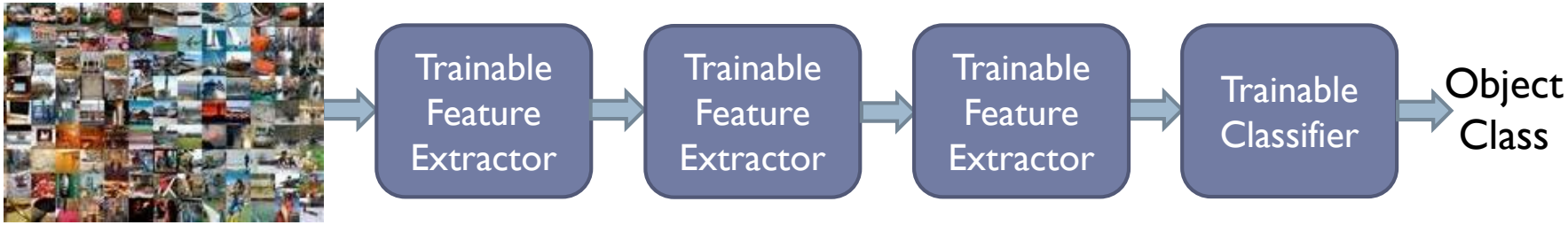
# CNN: Revaluation in CV

- Image/Speech Recognition

Before 2012

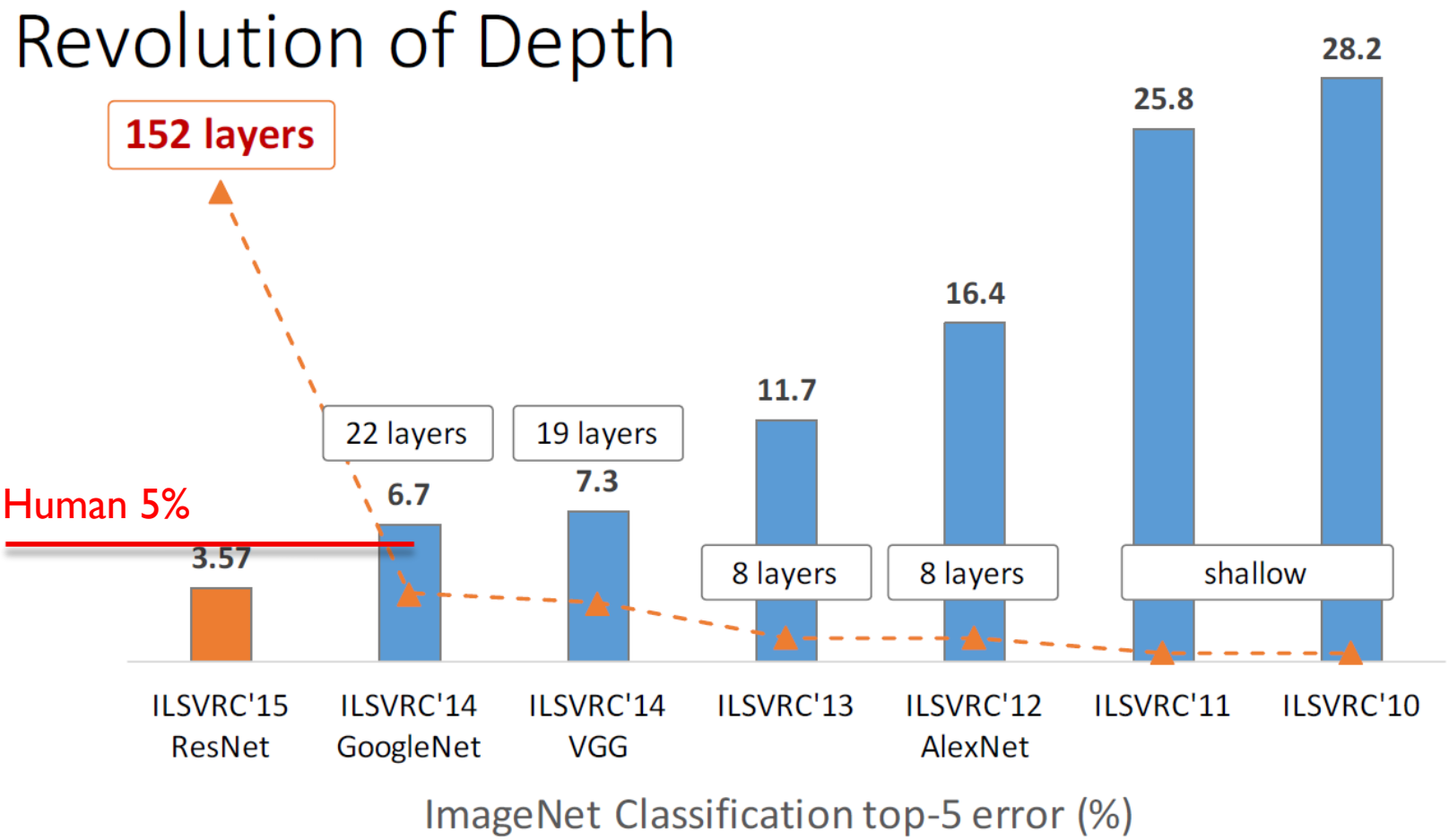



2012-Now: Deep Learning



# Rapid Development

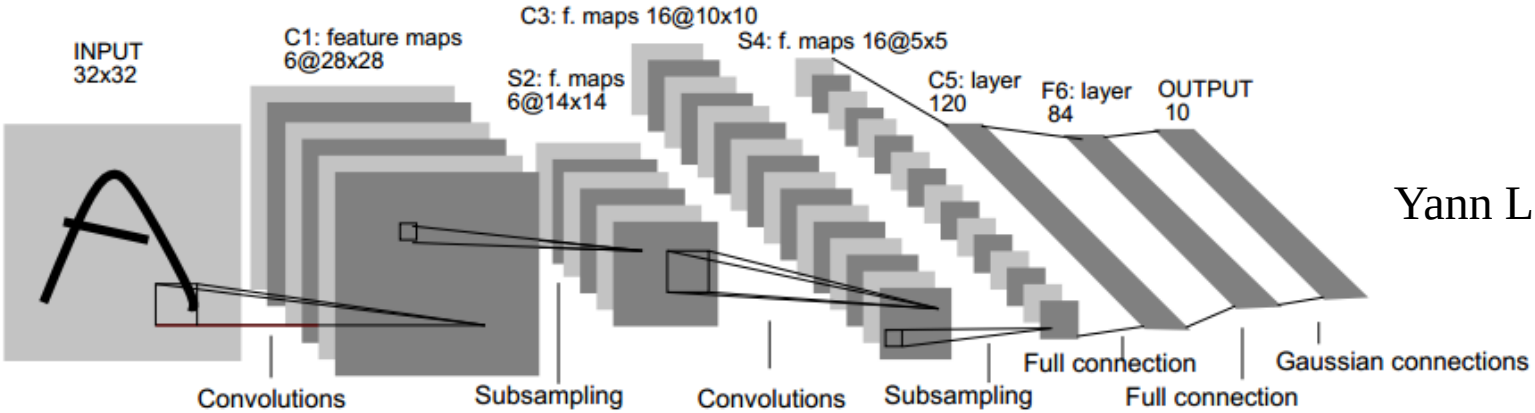
ImageNet Challenge



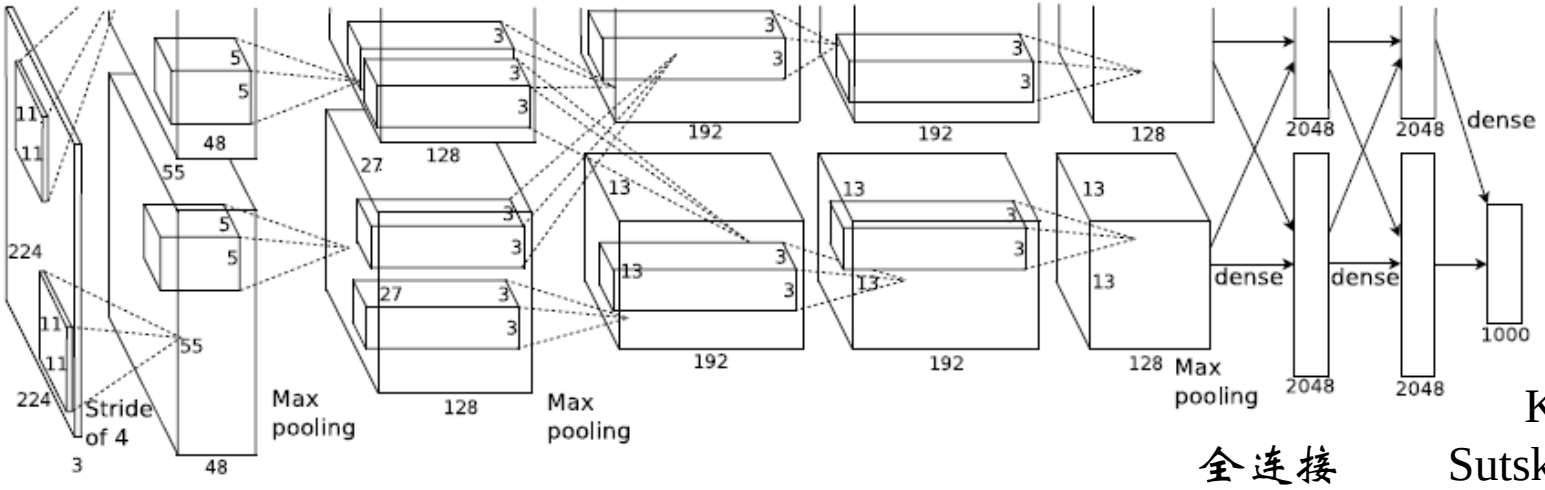

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

# CNN

- Alternating convolutions and pooling + fully-connected layer




Yann Lecun et al. 1998




卷积+池化

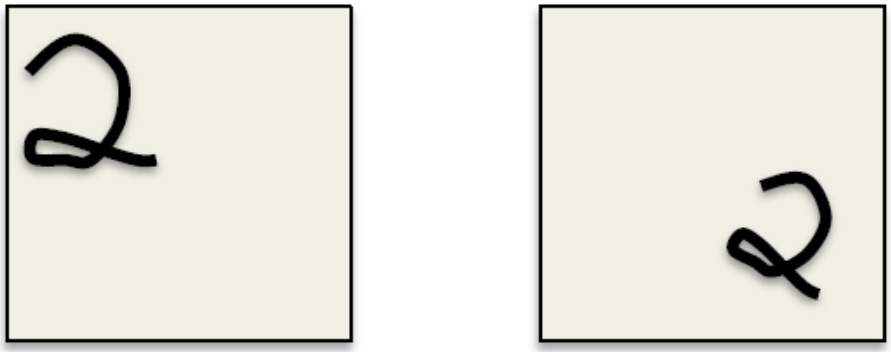
全连接

Krizhevsky, Sutskever & Hinton 2012

# CNN: Visual Cortex Process

- Invariance

Conv



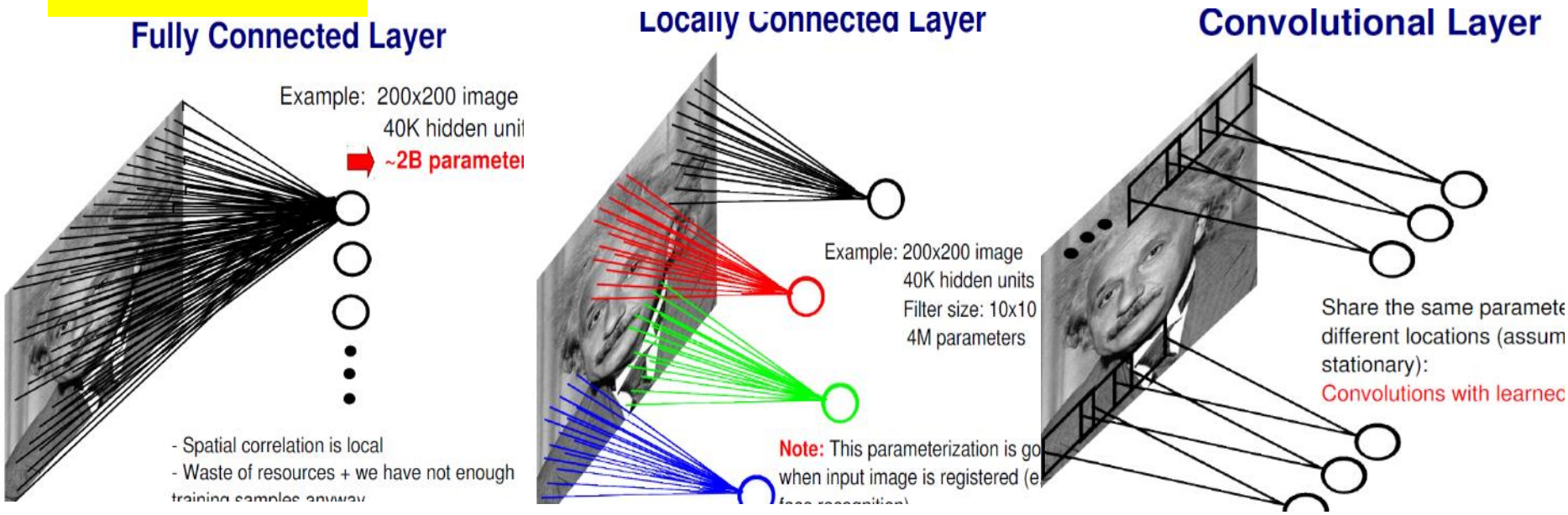
- Compositionality

Deep

[Lake et al. 2015]

# Reduce DoF Using Prior Knowledge

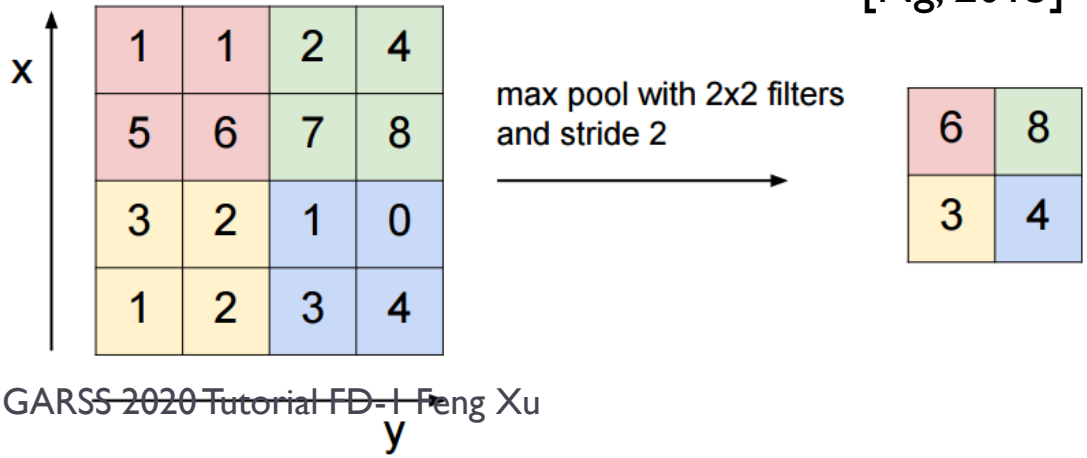
## Convolution



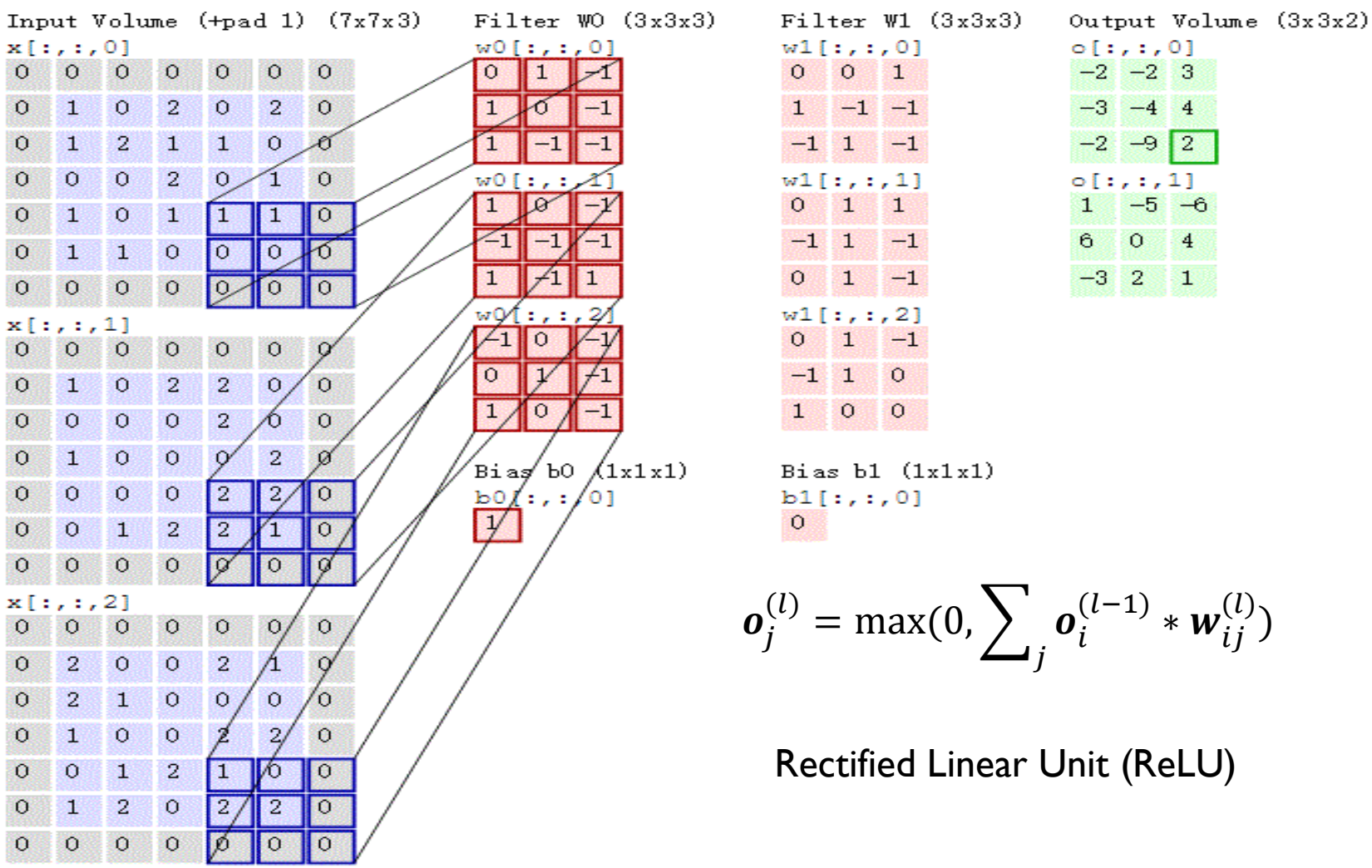

[Ng, 2015]

## Pooling

- Dimension Reduction
- Translational Invariance



# Convolution




$$\boldsymbol{o}_j^{(l)} = \max(0, \sum_j \boldsymbol{o}_i^{(l-1)} * \boldsymbol{w}_{ij}^{(l)})$$

Rectified Linear Unit (ReLU)

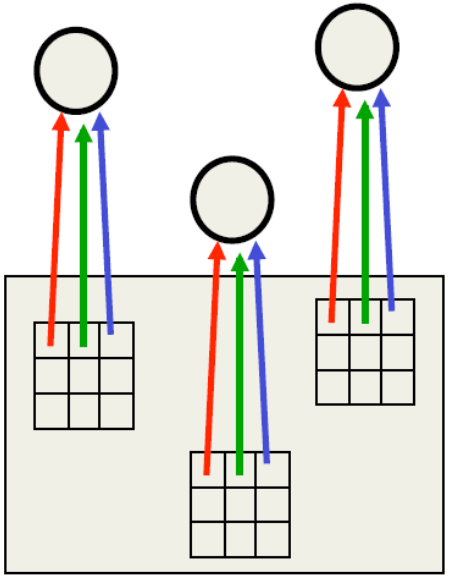


[Ng, 2015]

# Use prior knowledge to reduce unknowns

- Use many different copies of the same feature detector with different positions.
  - Could also replicate across scale and orientation (tricky and expensive)
  - Replication greatly reduces the number of free parameters to be learned.
- Use several different feature types, each with its own map of replicated detectors.
  - Allows each patch of image to be represented in several ways.

The red connections all have the same weight.



[He, 2016]

shallower

pixels → classifier → "bus"?

→ edges → classifier → "bus"?

SIFT/HOG

deeper

→ edges → histogram → classifier → "bus"?

→ edges → histogram → K-means/ sparse code → classifier → "bus"?

Specialized components, domain knowledge required

→ edges → histogram → K-means/ sparse code → classifier → "bus"?

Generic components ("layers"), less domain knowledge

→ [ ] → [ ] → [ ] → [ ] → "bus"?

Repeat elementary layers => Going deeper

→ [ ] → [ ] → [ ] → [ ] → [ ] → [ ] → "bus"?

# BP of Convolution Layer

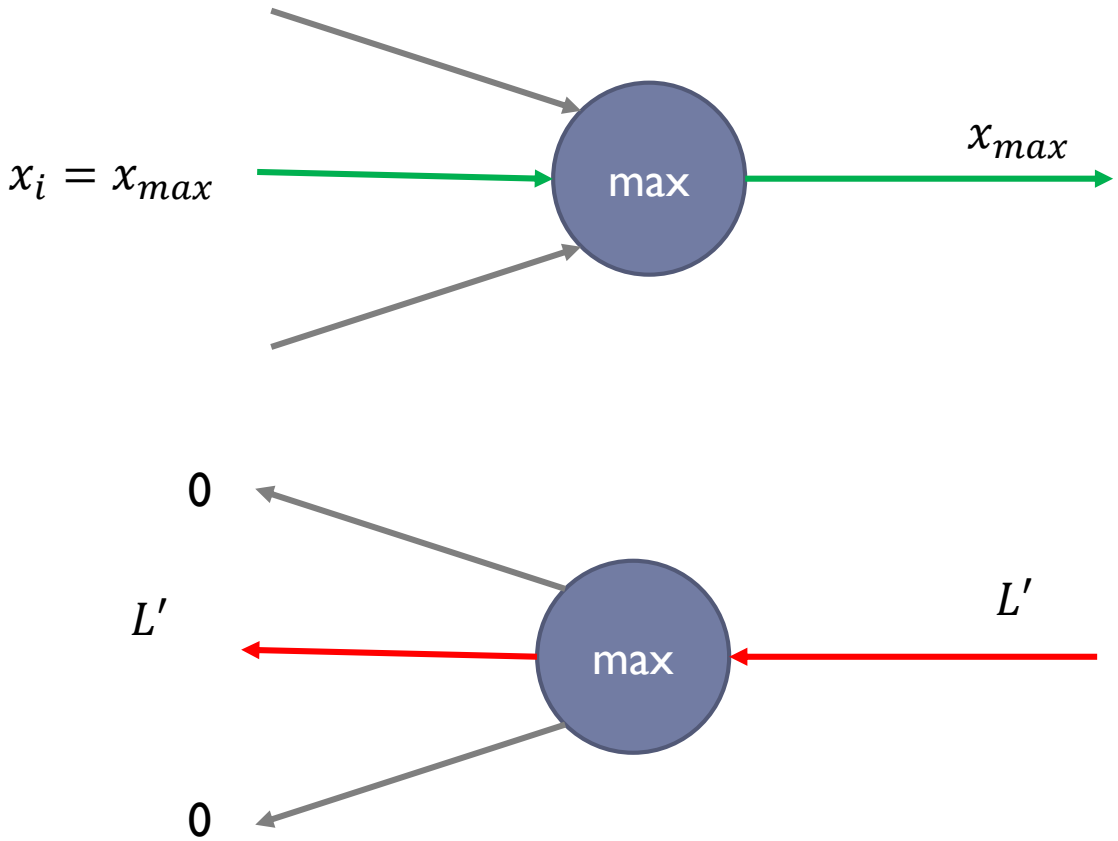
- Constraints: Weights to shared
- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.
  - So if the weights started off satisfying the constraints, they will continue to satisfy them.

- ReLU Gradient
- $f' = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$

$$\begin{aligned} &\textit{To constrain}: \quad w_1 = w_2 \\ &\textit{we need}: \quad \Delta w_1 = \Delta w_2 \\ &\textit{compute}: \quad \frac{\partial E}{\partial w_1} \quad \textit{and} \quad \frac{\partial E}{\partial w_2} \\ &\textit{use} \quad \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \quad \textit{for } w_1 \textit{ and } w_2 \end{aligned}$$

# BP of Pooling Layer

- Save the maximum input information

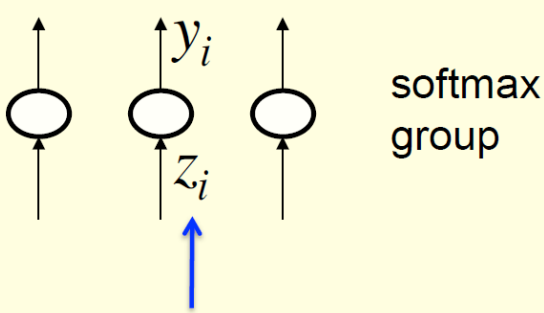


# Softmax

- Applies to the output layer in the case of multi-class classification
- Estimates the posterior probabilities over each class:

$$p_i = \frac{\exp\left(O_i^{(L)}\right)}{\sum_{j=1}^{K} \exp\left(O_j^{(L)}\right)} \quad (i = 1, \cdots, K)$$

The output units in a softmax group use a non-local non-linearity:




$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

$$\frac{\partial y_i}{\partial z_i} = y_i\,(1 - y_i)$$

this is called the "logit"

# Cross-Entropy (Softmaxloss)

- Cross-entropy loss function: minimizing the discrepancy between the ground truth $y$ and the network prediction $\boldsymbol{p}$:

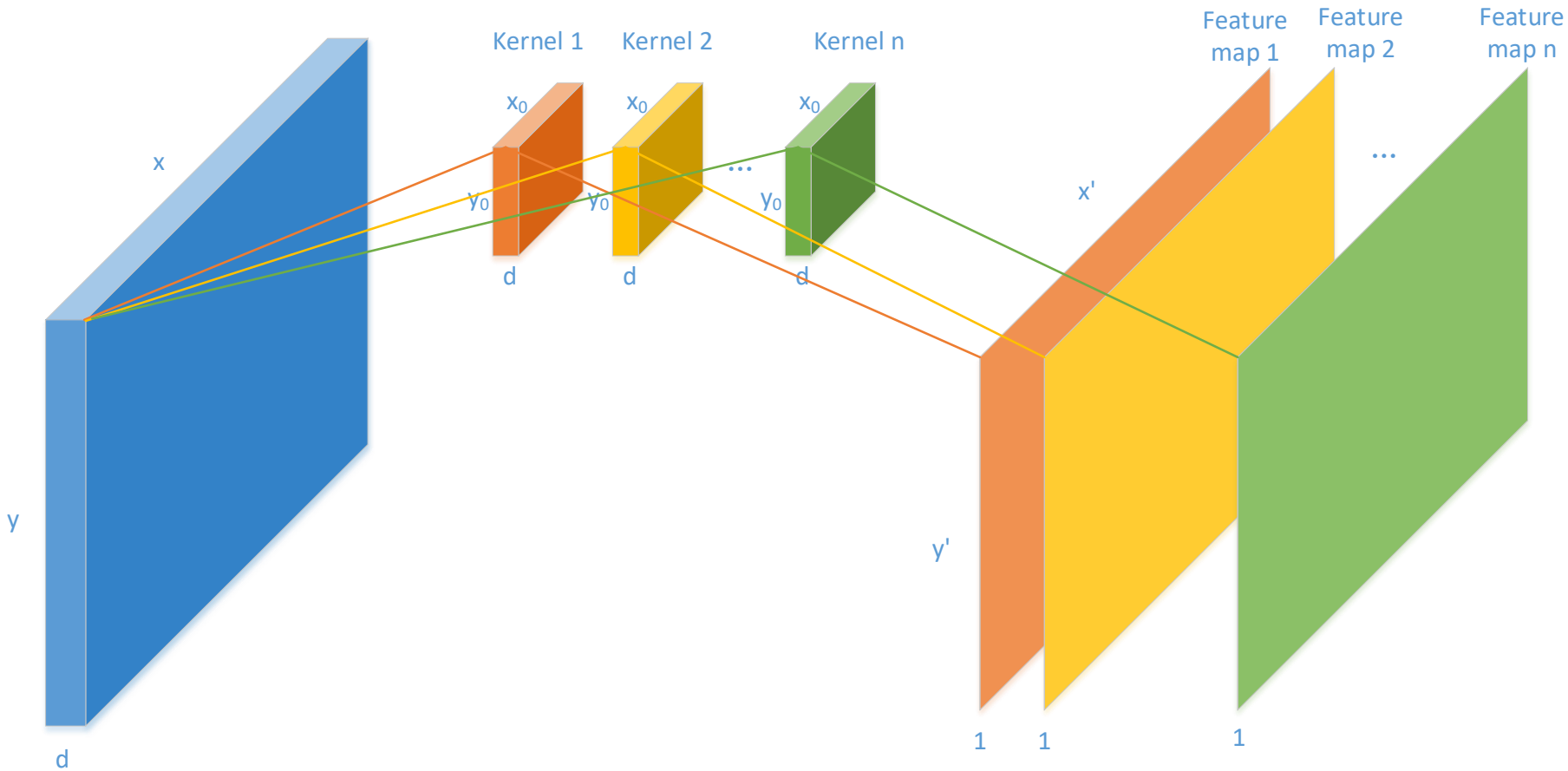
$$L(w) = \sum_{j=1}^{K} -t_j \log p_j = -\log p_y$$

- Gradient of Softmaxloss is balanced out between softmax and cross-entropy

$$\frac{\partial L}{\partial O} = \frac{\partial L}{\partial p} \cdot \frac{\partial p}{\partial O} = t - p$$

- Minimum loss while error backpropagation!

# CNN Parameters




x x y x d

$x_0$ x $y_0$ x d x n

x' x y' x n

# CNN Parameters

- Padding
- Stride

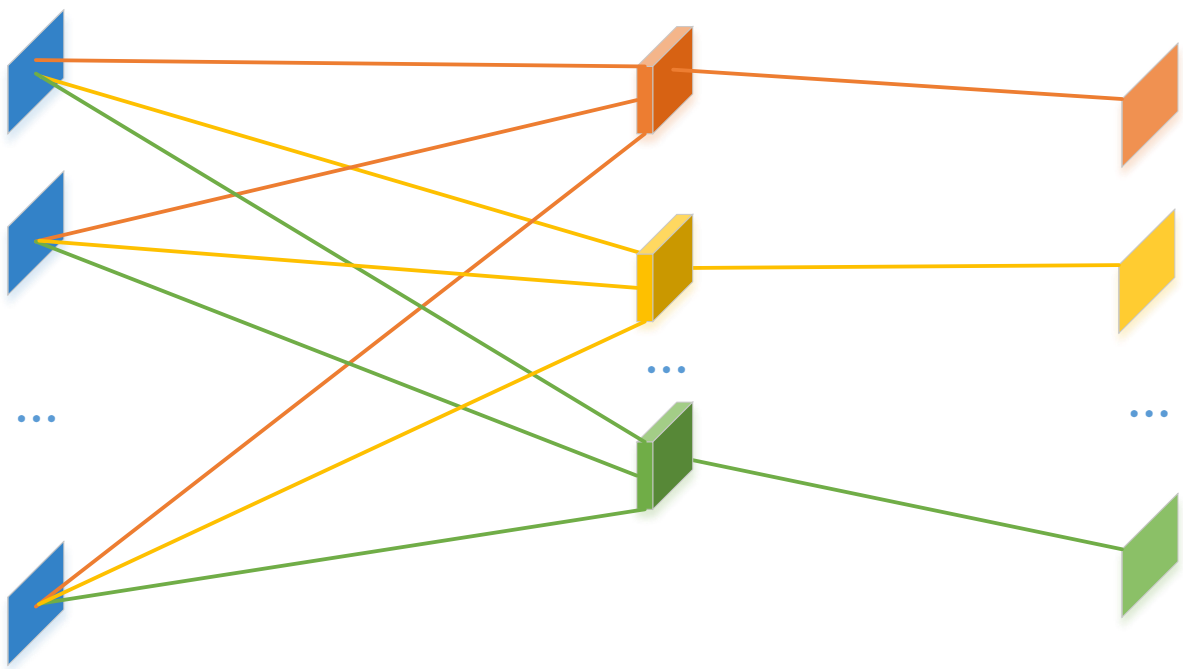
- $x' = \frac{(x+2p)-x_0}{s} + 1$
- $y' = \frac{(y+2p)-y_0}{s} + 1$

- applies to both Conv and Pooling layers.

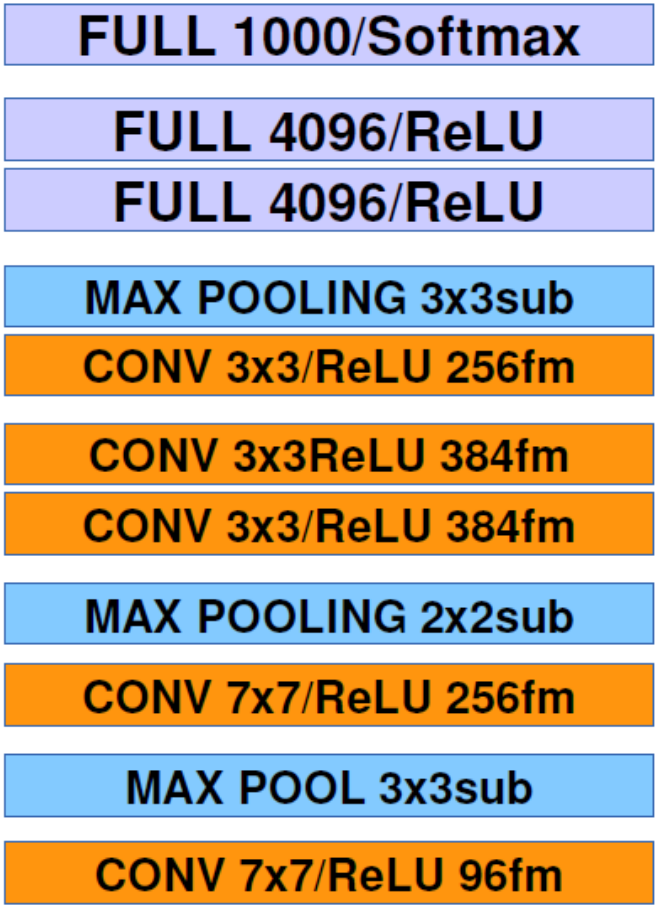
- Once parameters of input (previous) layer and conv/pool layer are given, the size of output (next) layer are automatically given.

# Fully-connected Layer

- Input Layer is 1x1xd， Kernel is 1x1xdxn
- Output is 1x1xn
- Just a special case of convolution layer

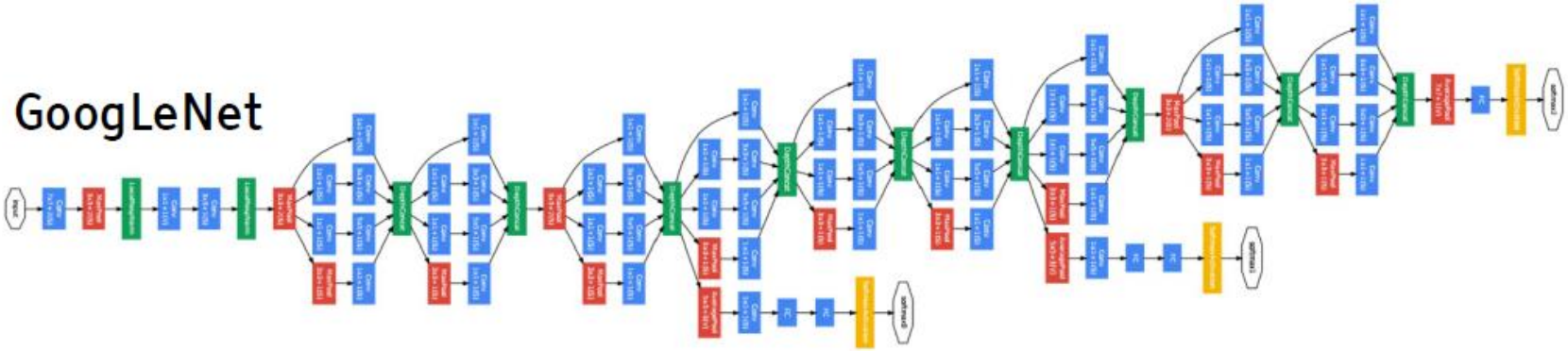


# Typical CNN Architecture

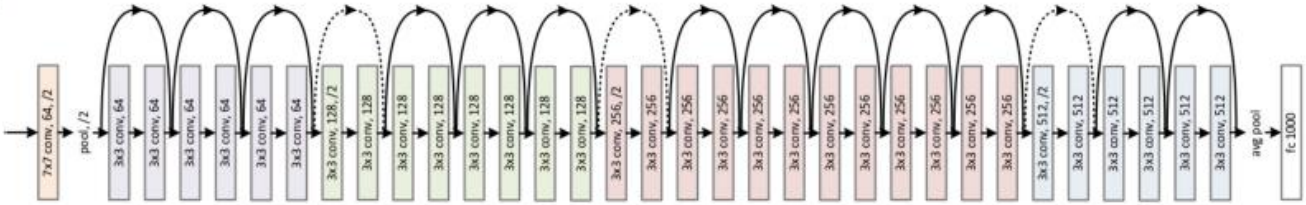


VGG

GoogLeNet



ResNet



# References


- Goodfellow: Deep Learning, MIT Press
- Nielsen: neuralnetworksanddeeplearning.com
- Epelbaum: Deep learing technical introduction, arxiv
- Ng: CS229 of Stanford
- Zeiler: PhD Thesis, New York University
- Silver: Deep reinforcement learning tutorial
- LeCun: Deep Learning, talk at CERN, 2016
- Lee, Deep Learning Tutorial, 2016

- Vishal Maini: Machine Learning for Humans (https://medium.com/machine-learning-for-humans)
- **Vahid Kazemi:** Effective Tensorflow (https://github.com/vahidk/EffectiveTensorflow?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more)

# Thank you

**fengxu@fudan.edu.cn**

homepage.fudan.edu.cn/fengxu

github.com/fudanxu

復旦大學 电磁波信息科学教育部重点实验室

Key Laboratory of EMW Information,Fudan University

# Deep Learning as Applied in SAR Imagery Interpretation

**Feng Xu**

**Institute of EM Data Science and Remote Sensing Intelligence**

**Fudan University, Shanghai, China**

homepage.fudan.edu.cn/fengxu

github.com/fudanxu

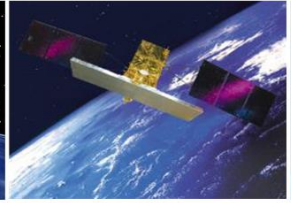
復旦大學 电磁波信息科学教育部重点实验室

Key Laboratory of EMW Information,Fudan University

# Outline

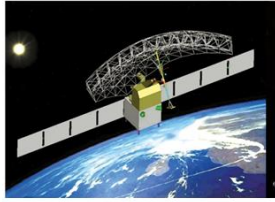
- SAR Information Retrieval
- DL Applications in SAR
  - POLSAR Classification
  - Automatic Target Recognition
  - Representation Learning
  - Speckle Reduction
  - Few-Shot Learning
- Outlook

ALOS

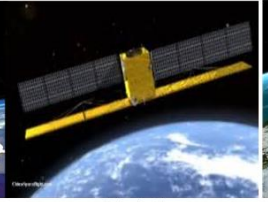


Cosmo-Skymed

TerraSAR-X



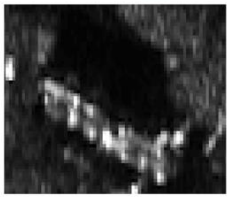
HJ-1C



GF-3

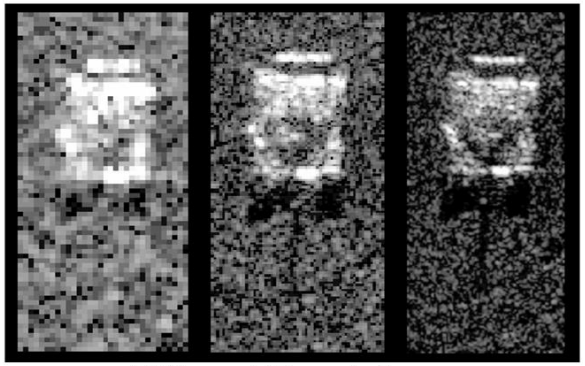
Tandem-L

## SAR Technology Advancement



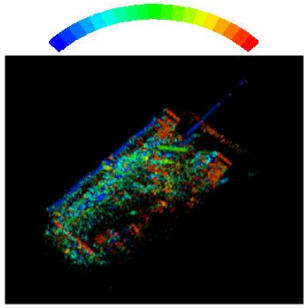
Meter-Res

Submeter-Res



Different Resolutions

Higher-Res

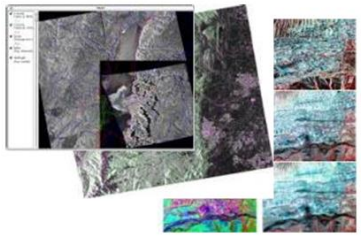


Multi-aspect

Higher-Dim

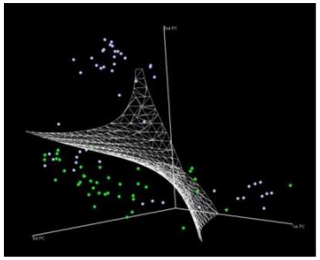
**Multi-Pol**
**Multi-Temp**
**Multi-Band**
**...**

## Multi-Dimensional SAR Data

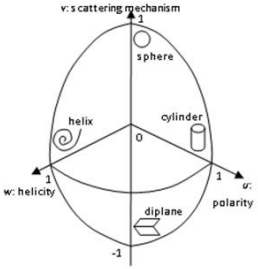


Challenge

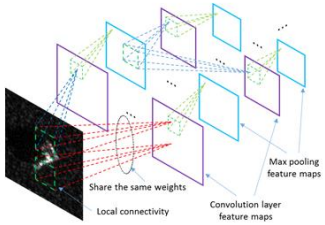
Human Interpreter



Statistical



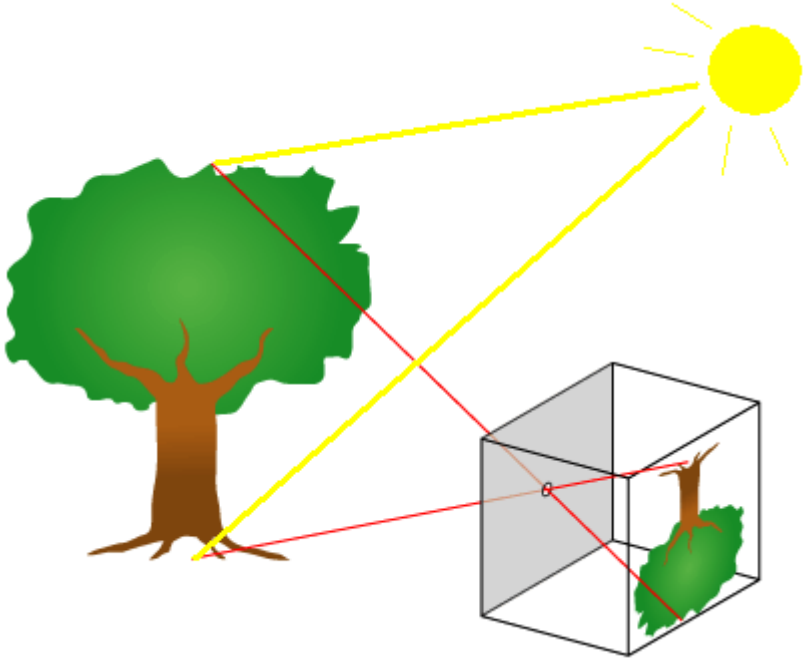

Physical-based

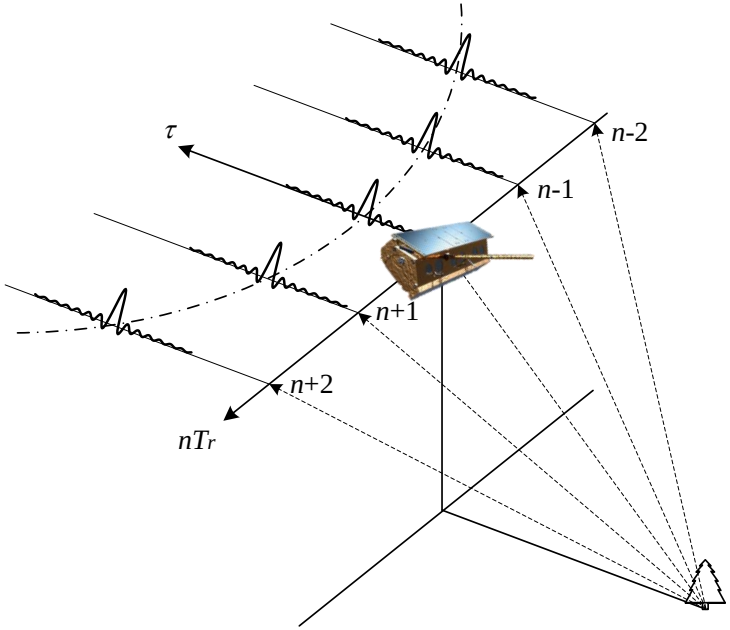



Synergy

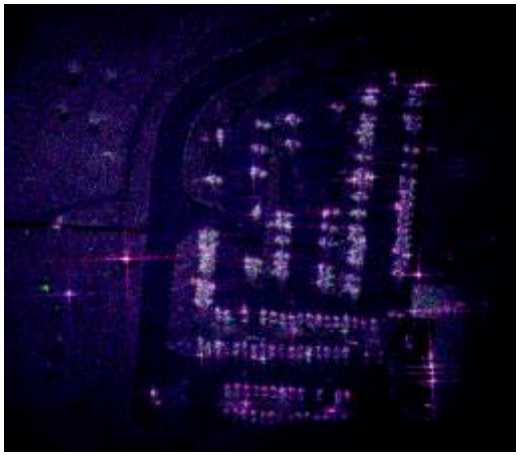
## SAR Imagery Interpretation

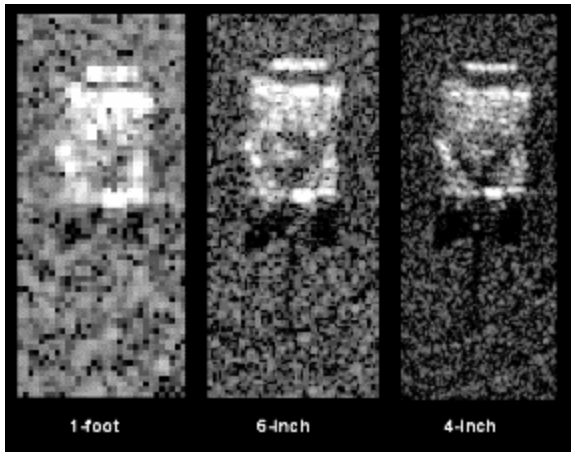
# SAR VS. Camera

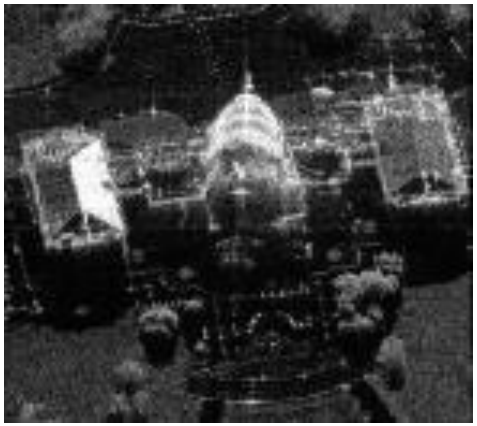




# SAR Image VS. Optical Image



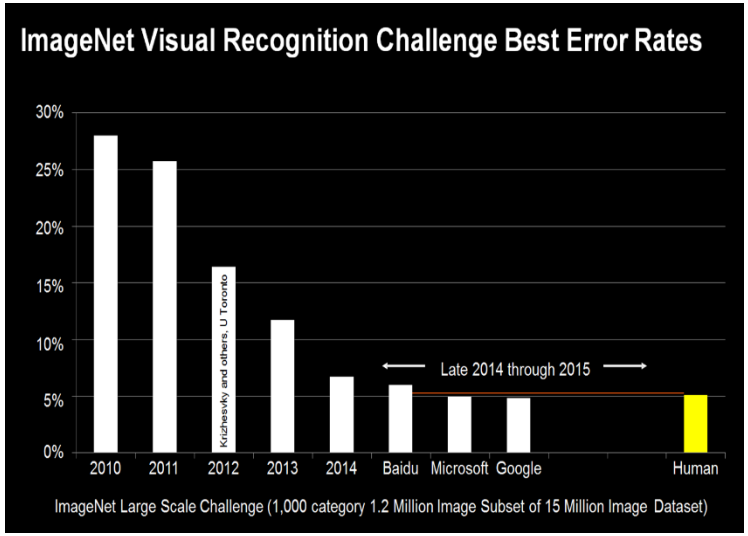


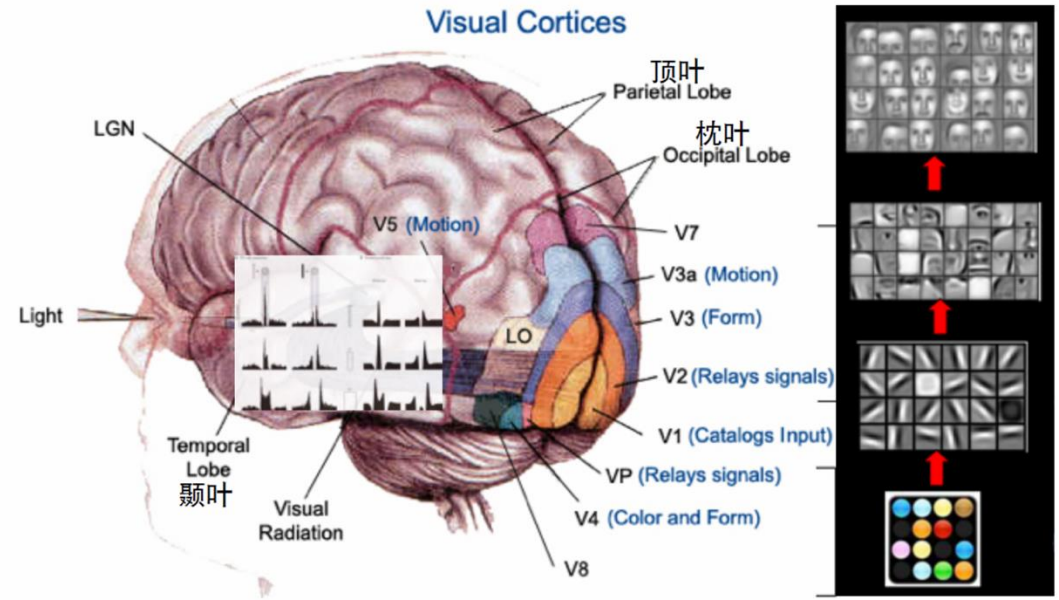


| | SAR Image | Optical Image |
|---|---|---|
| **Frequency Band** | Microwave | Optical |
| **Imaging Mechanism** | Phase-coherent Summation | Energy Focusing |
| **Projection Scheme** | Range vs. Azimuth | Elevation vs. Azimuth |
| **Resolution** | Constant Spatial Resolution (Independent of Distance) | Constant Angular Resolution (Proportional to Distance） |
| **Typical Viewpoint** | From Aircraft/Satellite | From Human |
| **Data Formats** | Complex, Multi-Dimension, Multi-Modality | RGB,Video |

# Challenges of DL

- Deep learning has achieved great success in computer vision, object detection, speech recognition, translation etc.
- DL Application to Remote Sensing (SAR) is booming
- DL is in essence… fitting… heavily relying on training samples



# POLSAR Classification

# PolSAR Classification

- Sliding Window
- Multi-channel
- Ground Truth Label

$$A = 10\log_{10}\left(T_{11} + T_{22} + T_{33}\right)$$

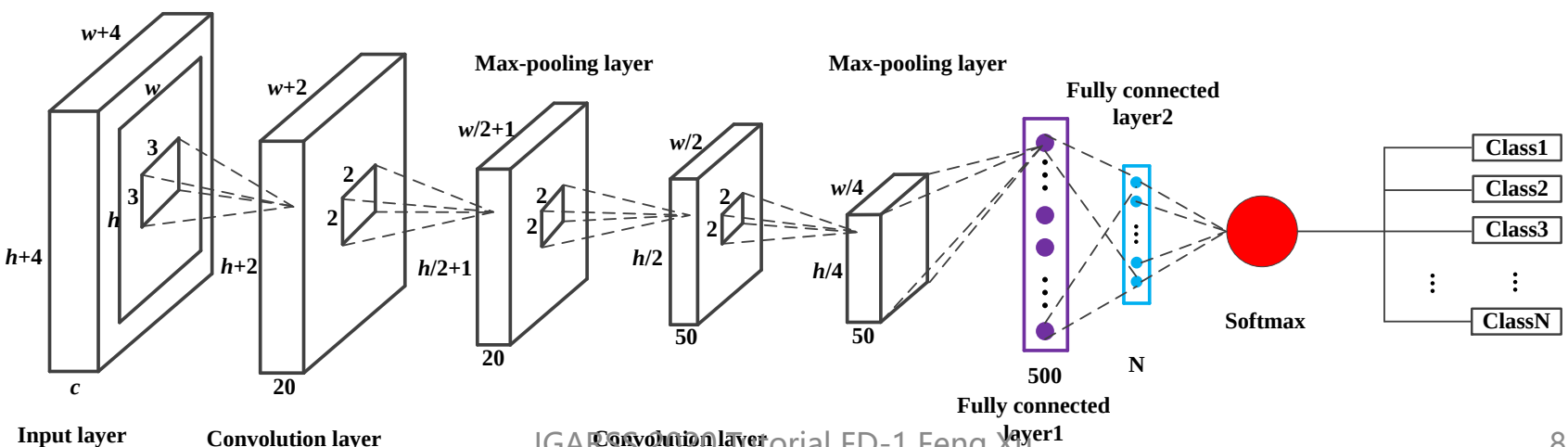
$$B = T_{22} / \left(T_{11} + T_{22} + T_{33}\right)$$

$$C = T_{33} / \left(T_{11} + T_{22} + T_{33}\right)$$

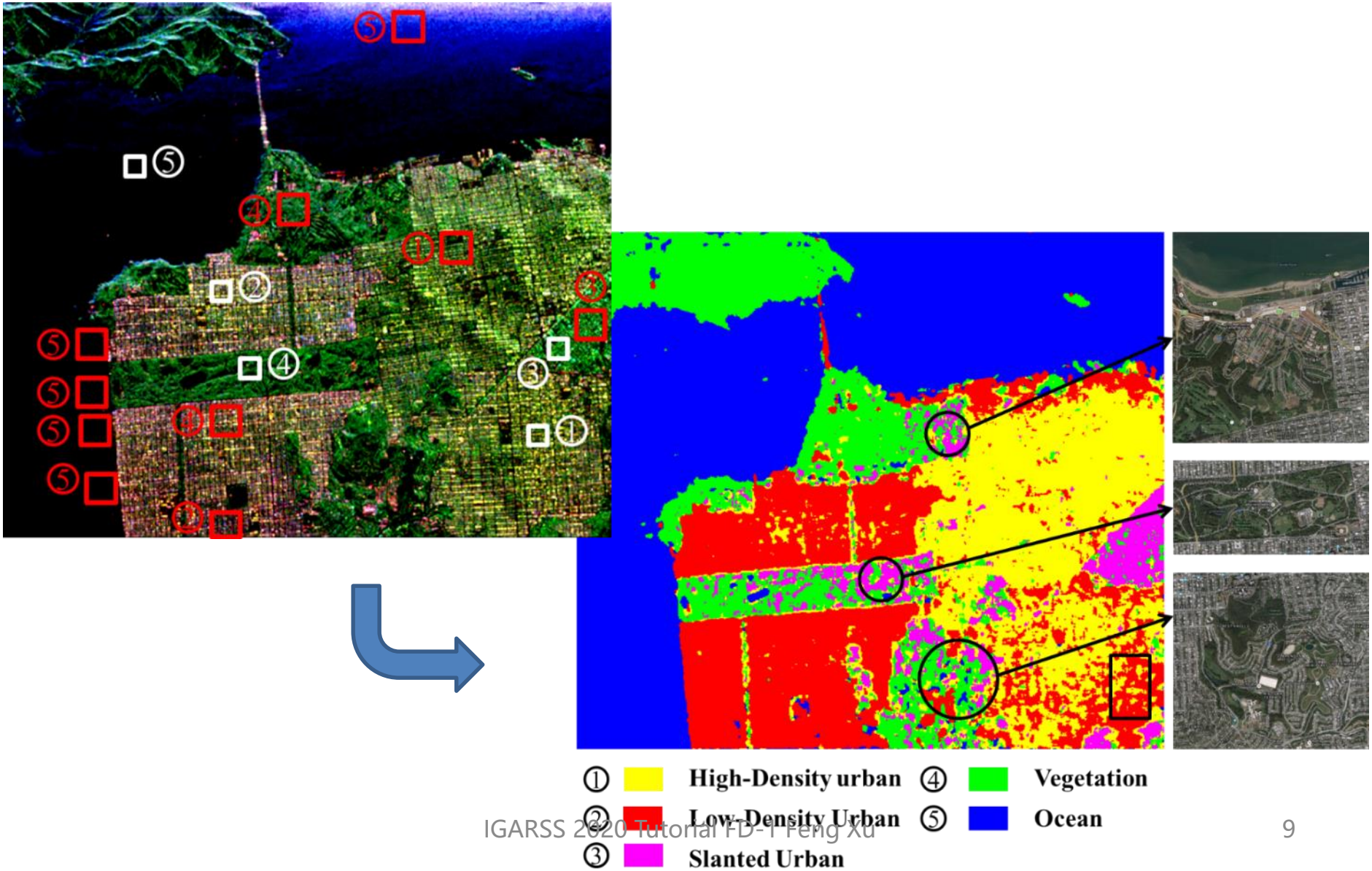
$$D = \left|T_{12}\right| / \sqrt{T_{11} \cdot T_{22}}$$

$$E = \left|T_{13}\right| / \sqrt{T_{11} \cdot T_{33}}$$

$$F = \left|T_{23}\right| / \sqrt{T_{33} \cdot T_{22}}$$

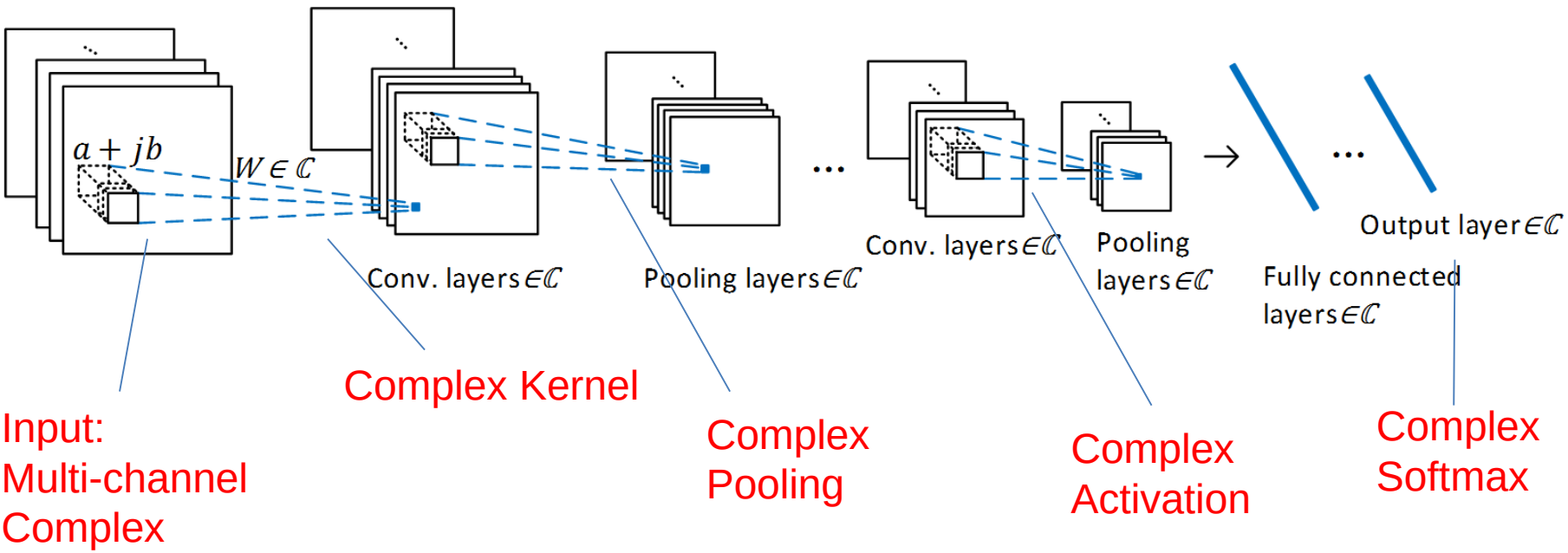


# AirSAR San Francisco



# Complex Valued - CNN

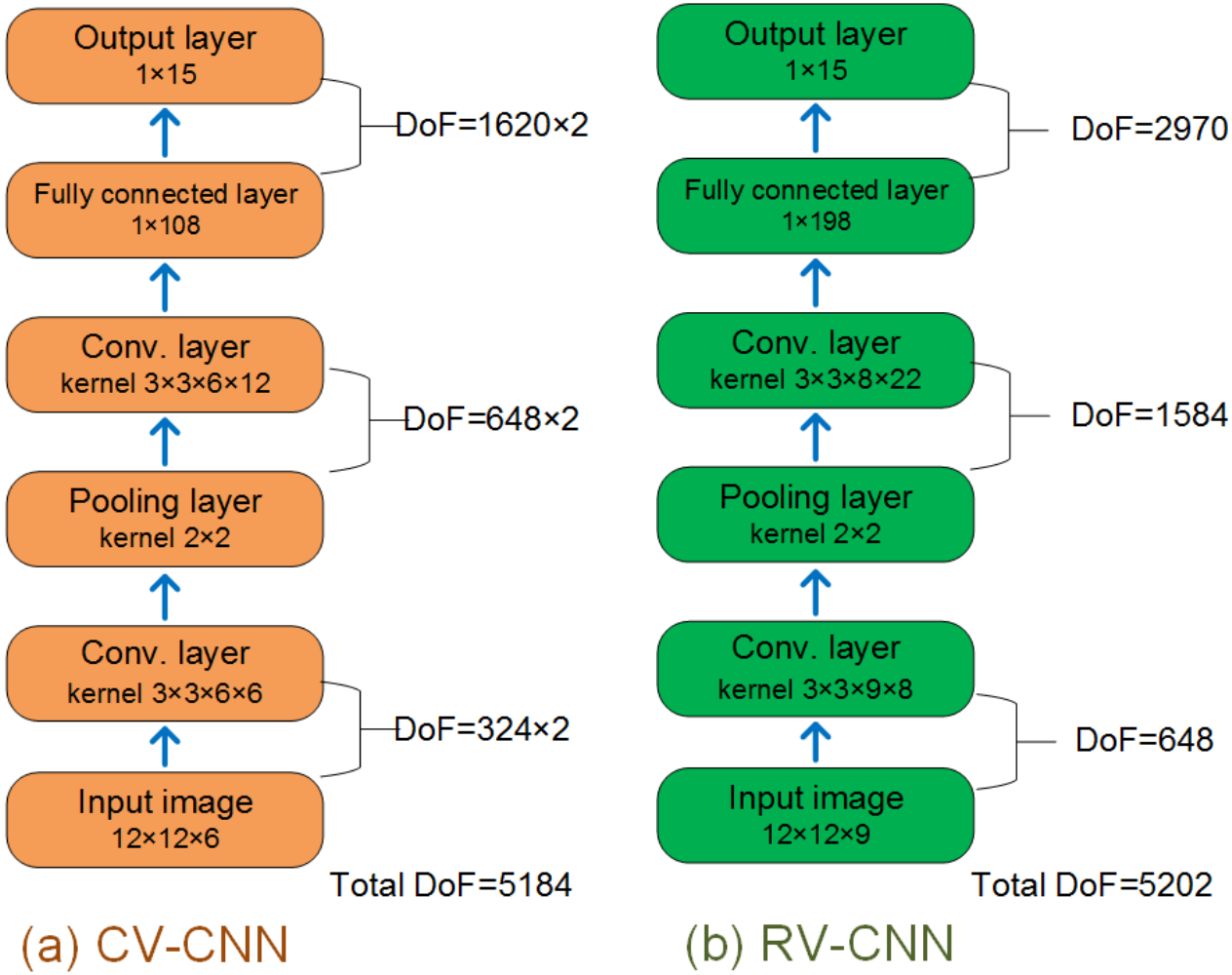
- Phase Information is Useful
- Extend CNN to Complex Domain



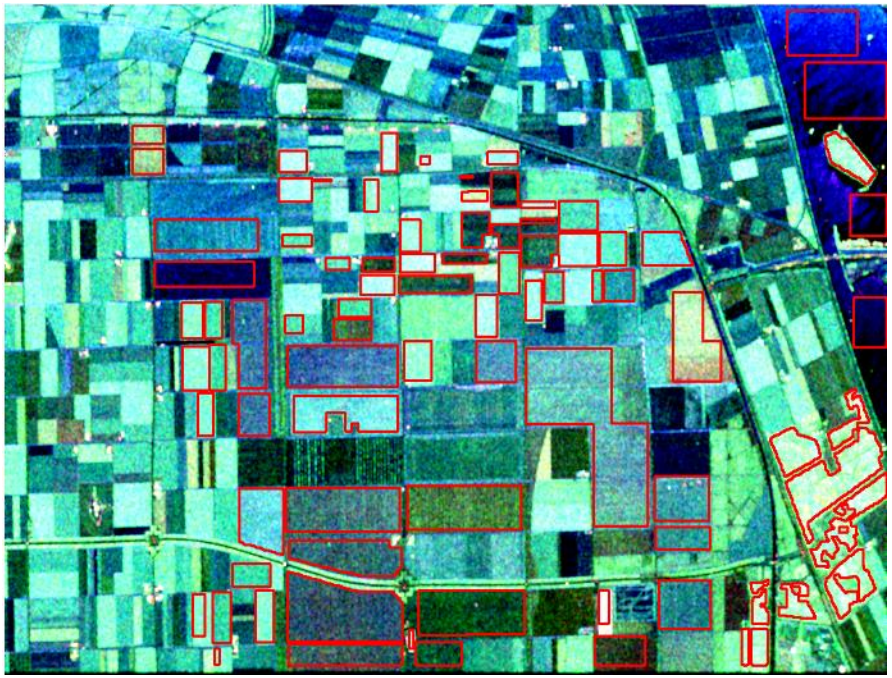

BP, SGD etl. all in Complex Domain

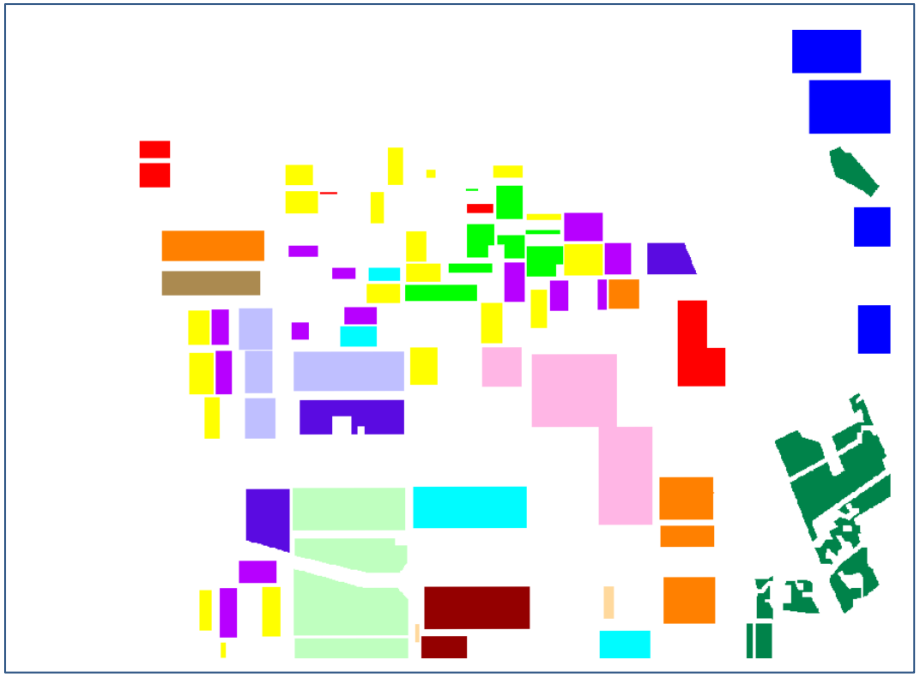
- **Zhang et al., "Complex-Valued Convolutional Neural Network and Its Application in Polarimetric SAR Image Classification", *IEEE TGRS*, 2017.**
- **Zhou et al., "Polarimetric SAR Image Classification Using Deep Convolutional Neural Networks", IEEE Geosci. Remote Sens. Lett., 2016**
- **[code: https://github.com/fudanxu/CV-CNN]**

# CV-CNN vs. RV-CNN



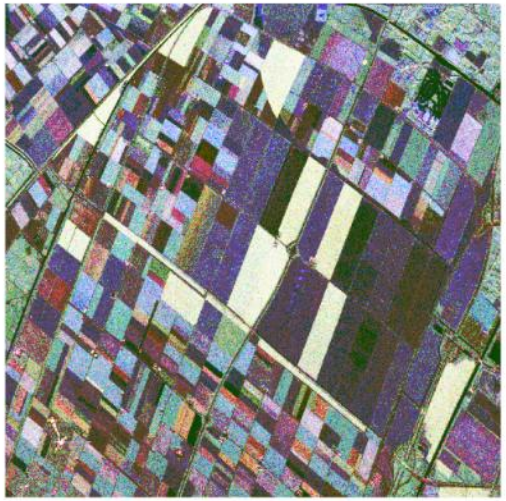
# AirSAR: Flevoland

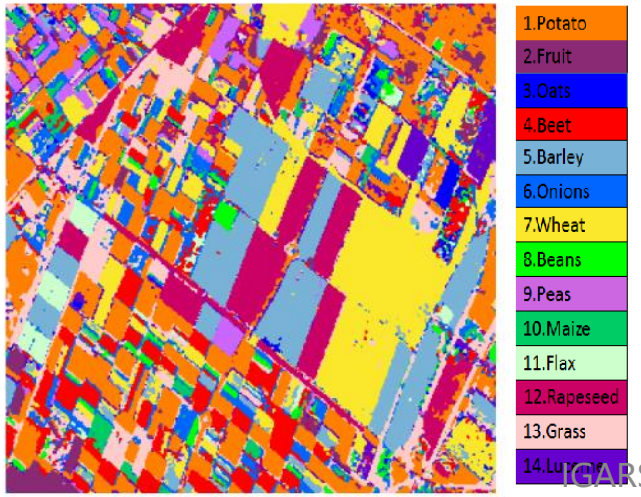




| | | | | |
|---|---|---|---|---|
| Stembeans | Peas | Forest | Lucerne | Wheat |
| Beet | Potatoes | Bare soil | Grasses | Rapeseed |
| Barley | Wheat 2 | Wheat 3 | Water | Buildings |

# Flevoland2

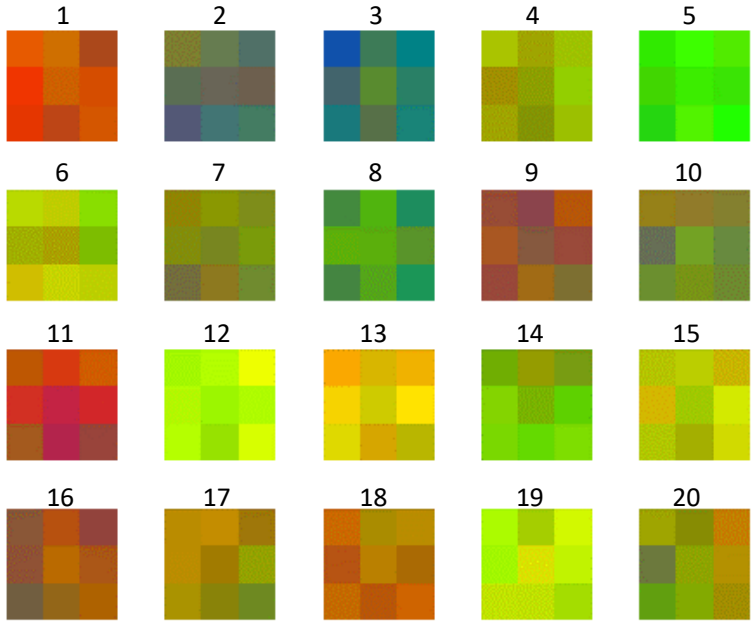





| Class | CV-CNN | RV-CNN | Gao et al. [31] Using L band | Liu et al. [32] Using PLC band |
|---|---|---|---|---|
| 1Potato | 99.71 | 92.71 | 89.4 | 99.6 |
| 2Fruit | 97.30 | 88.62 | 91.7 | 99.7 |
| 3Oats | 98.78 | 93.40 | 92.0 | 99.7 |
| 4Beet | 93.51 | 85.27 | 82.1 | 95.9 |
| 5Barley | 99.83 | 94.15 | 84.2 | 99.4 |
| 6Onions | 97.70 | 73.52 | 65.0 | 93.4 |
| 7Wheat | 99.48 | 83.72 | 91.5 | 99.5 |
| 8Beans | 92.76 | 60.34 | 86.8 | 99.7 |
| 9Peas | 99.81 | 95.88 | 95.6 | 100 |
| 10Maize | 98.60 | 83.02 | 77.2 | 89 |
| 11Flax | 95.88 | 92.14 | 96.9 | 97.3 |
| 12Rapeseed | 99.52 | 94.33 | 90.1 | 99.6 |
| 13Grass | 98.88 | 93.43 | 86.7 | 98.2 |
| 14Lucerne | 99.36 | 92.48 | 94.8 | 99.9 |
| Overall Accuracy | 98.84 | 90.32 | 87.4 | 98.9 |

# Learned Conv. Kernels

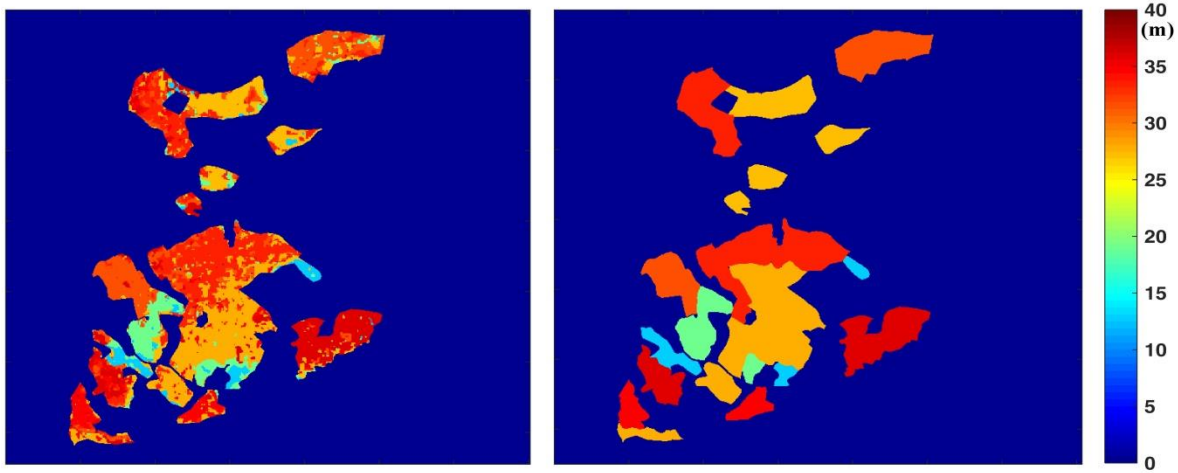
First Layer Kernels:
3x3 (Pauli RGB)




| Class | Responsive Kernels |
|---|---|
| Steambeans | 13,11,13,11,13 |
| Peas | 14,2,14,2,2 |
| Forest | 6,6,6,6,6 |
| Lucerne | 13,13,13,13,13 |
| Wheat | 7,7,2,7,11 |
| Beet | 14,14,14,14,14 |
| Potatoes | 14,14,6,10,14 |
| Bare soil | 3,3,3,3,3 |
| Grasses | 1,13,1,1,3 |
| Rapeseed | 3,2,3,14,3 |
| Barley | 1,6,13,3,13 |
| Wheat2 | 2,3,16,2,2 |
| Wheat3 | 16.16,1,16,16 |
| Water | 3,3,3,3,3 |
| Buildings | 18,18,18,18,18 |

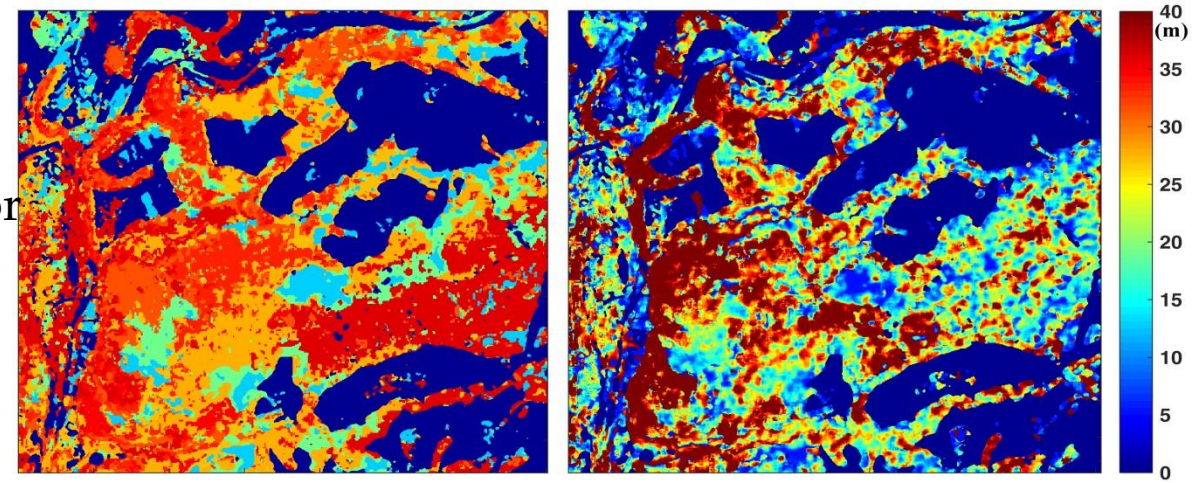
# Extension to POLINSAR Tree Height Estimation

Mapping heights for ground truth area




CV-CNN Ground Truth

Mapping heights for whole forest area

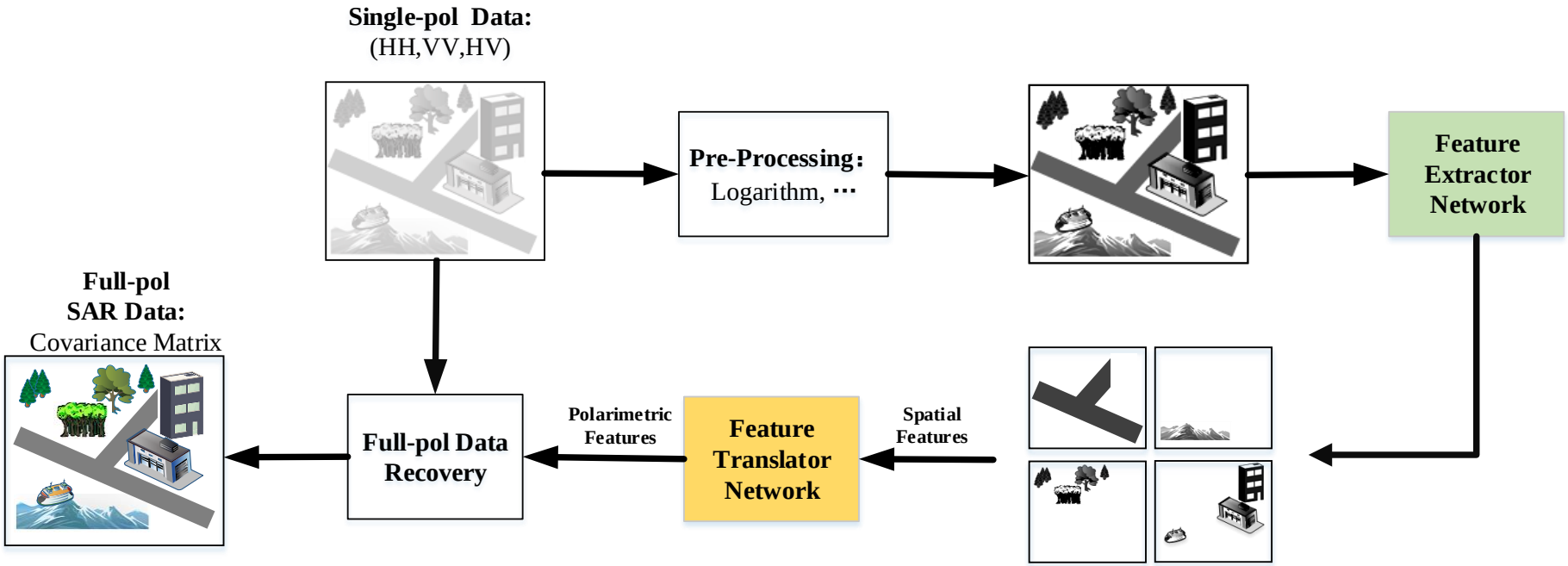



CV-CNN RVoG

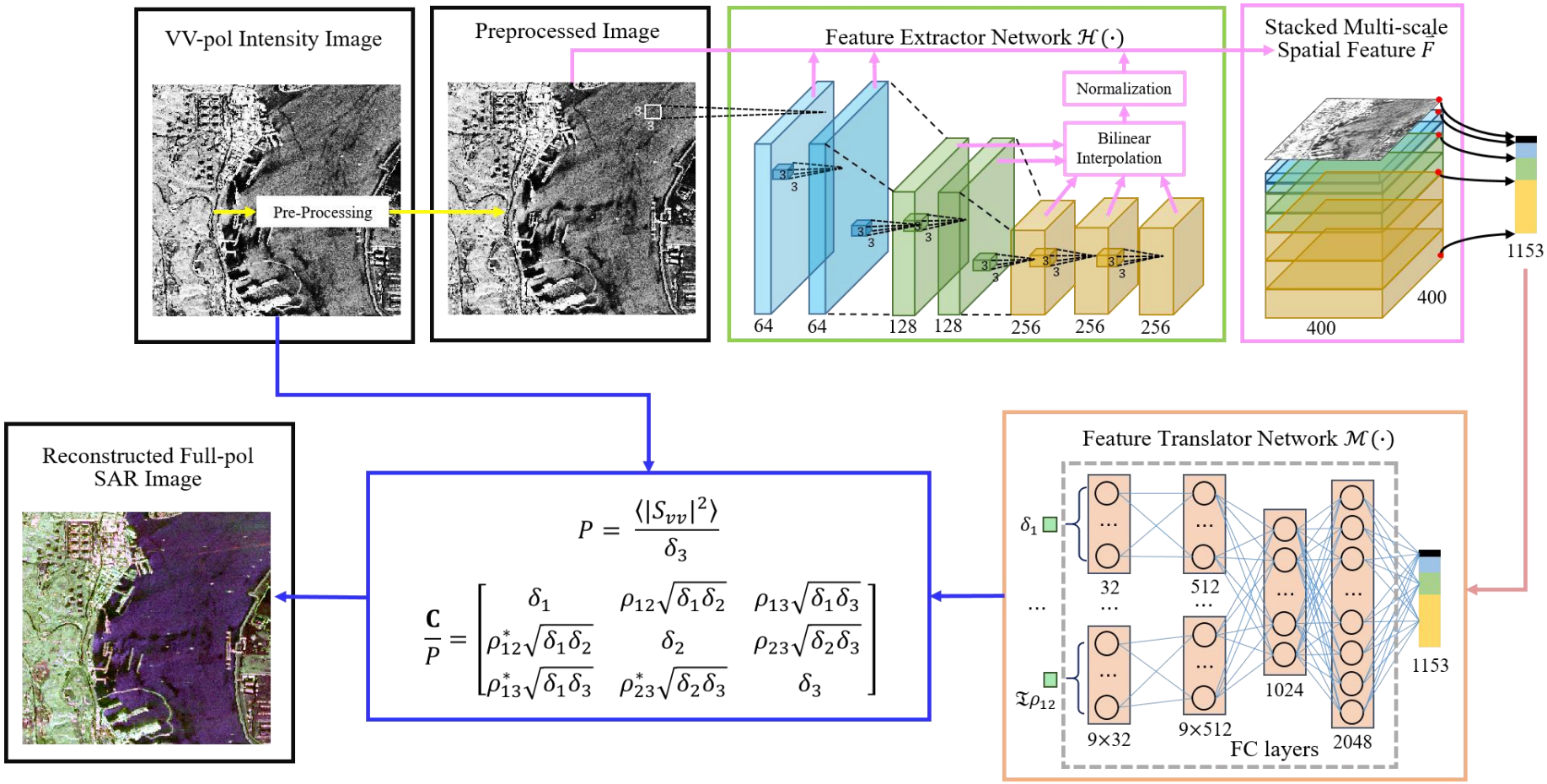
- **X. Wang and H. Wang, "Forest Height Mapping Using Complex-Valued Convolutional Neural Network," in IEEE Access, vol. 7, pp. 126334-126343, 2019**

# SAR Image Colorization

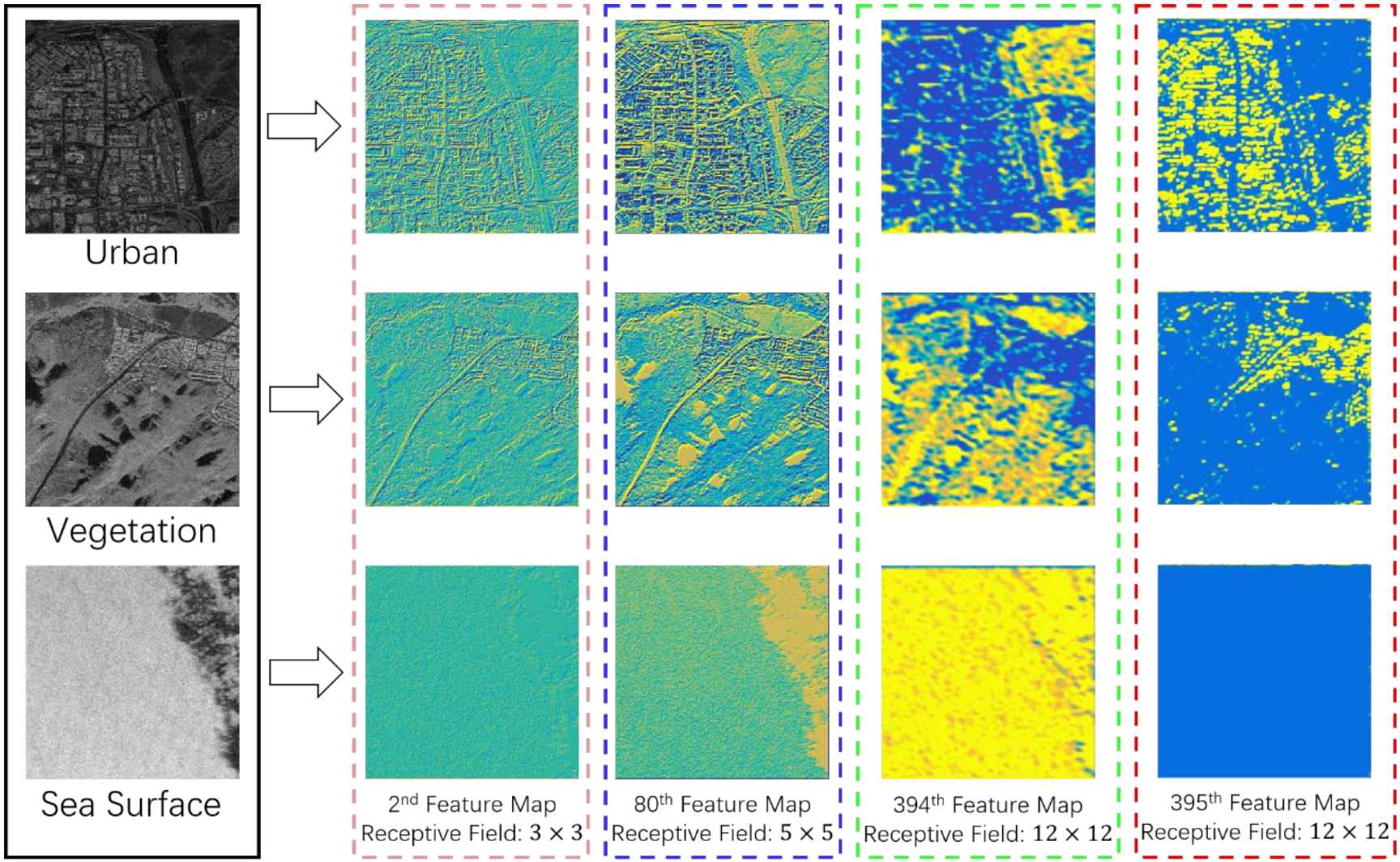
- Grayscale optical image can be colorized
- Single-Pol (compact-pol) SAR image can be converted to fully polarimetric



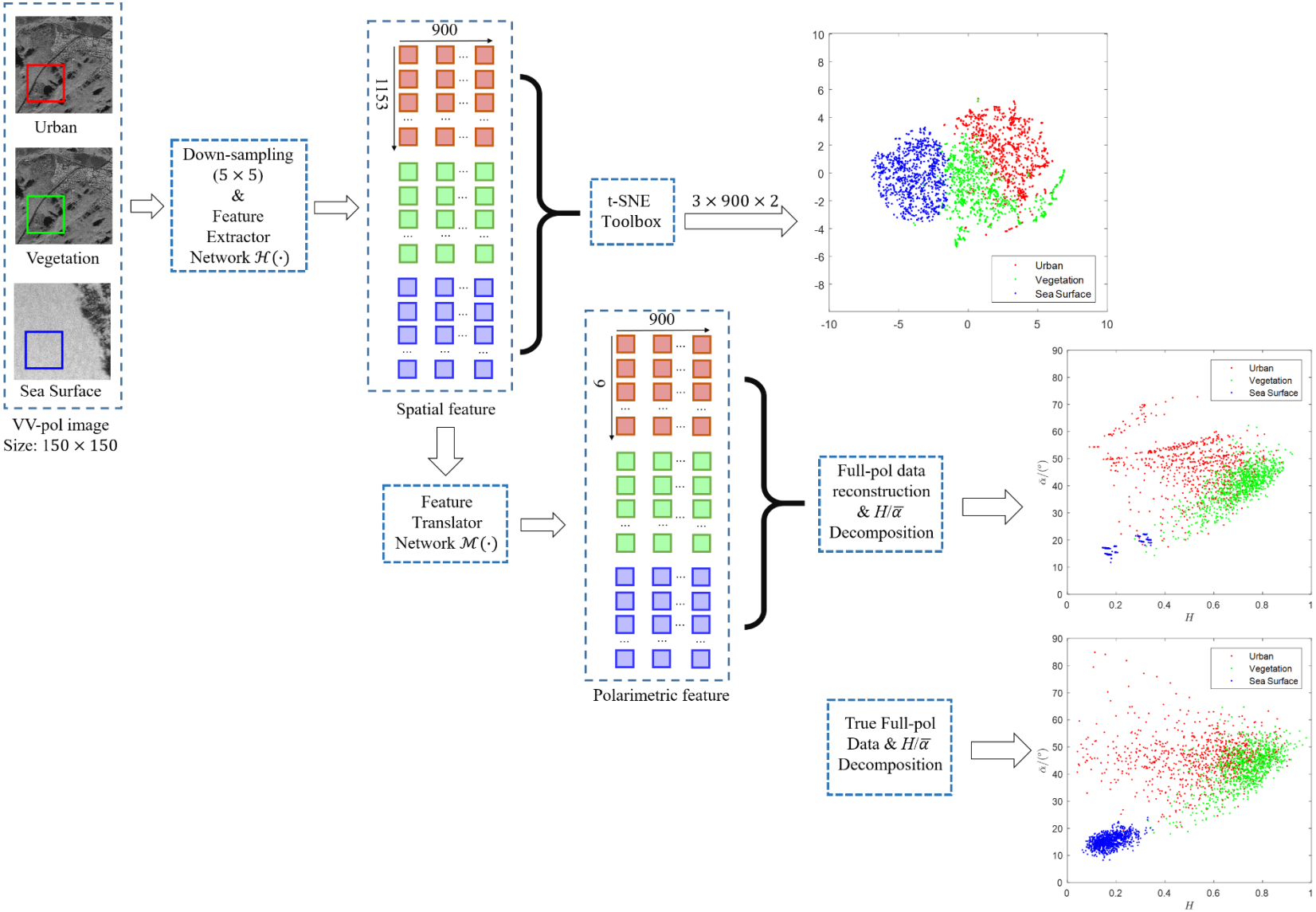
# SAR Image Colorization



# Features Extracted from VV SAR Image

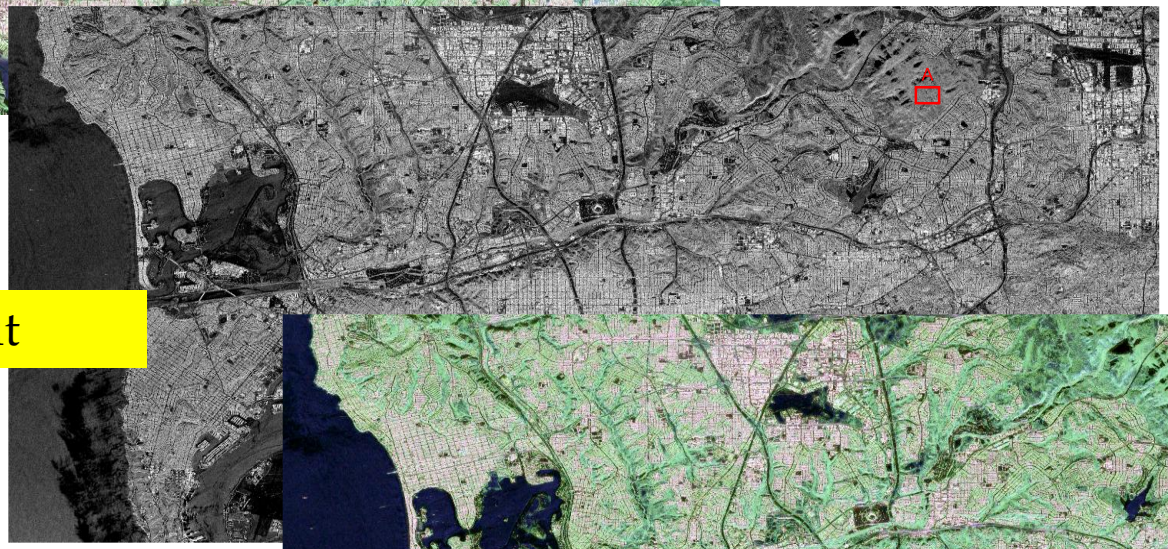


# Polarimetric Feature Reconstruction

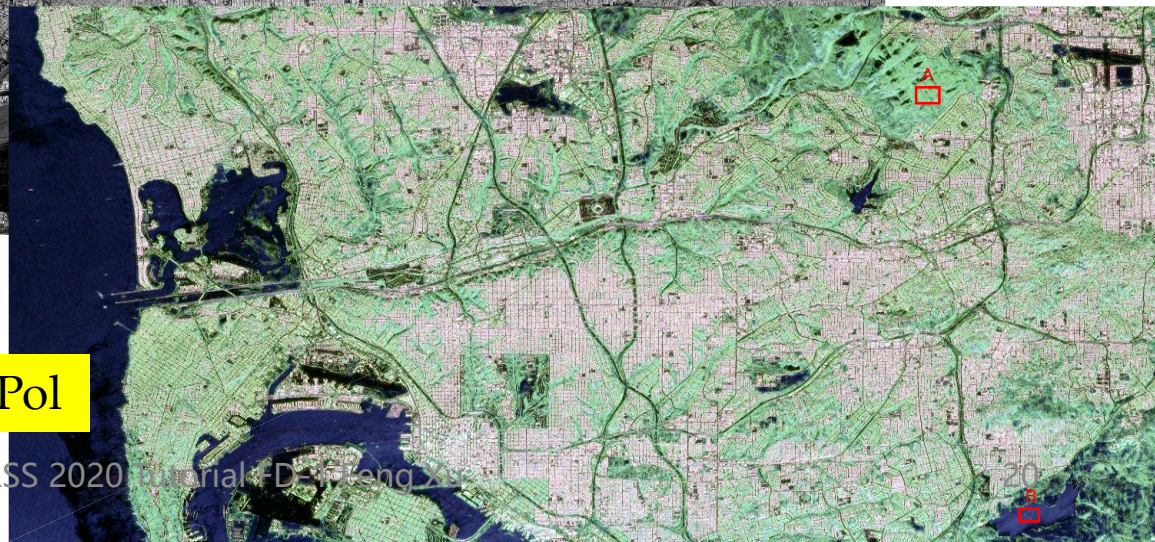


# SAR Image Colorization

Original True Full-Pol

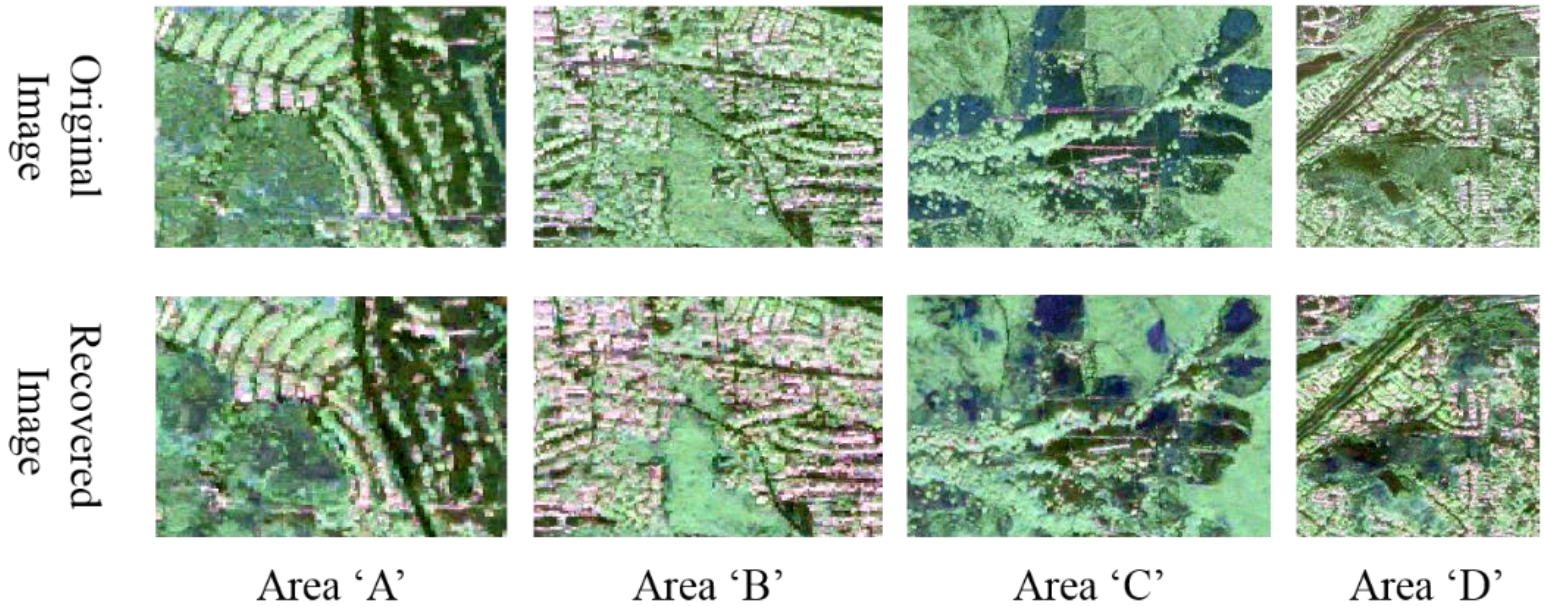


VV-Pol as Input



Reconstructed Full-Pol

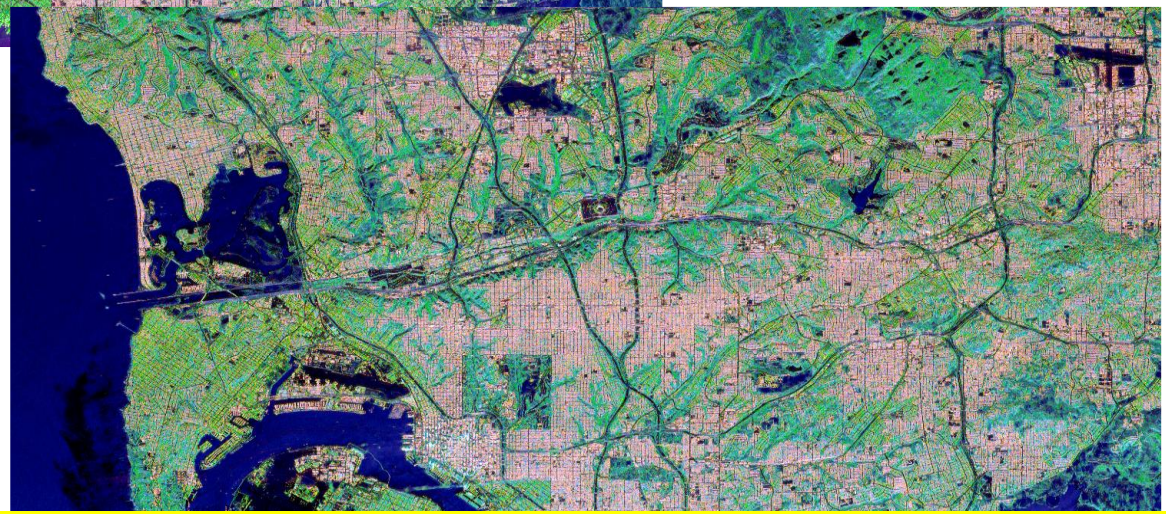
# Performance




| C Matrix | C11 | C22 | C33 | C13 | C23 | C12 |
|---|---|---|---|---|---|---|
| Coherency | 0.74 | 0.31 | 1 | 0.55 | 0.49 | 0.29 |

# Performance: Freeman Decomposition

True PolSAR Data

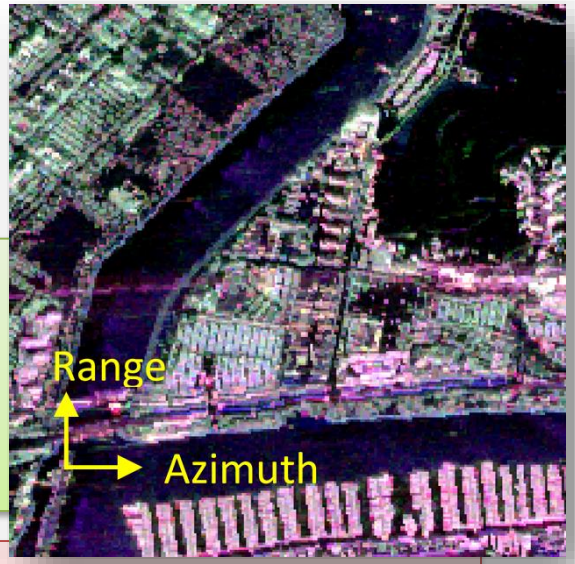


Reconstructed PolSAR Data

- **[Q. Song, Feng Xu, "Radar Image Colorization: Converting Single-Polarization to Fully Polarimetric Using Deep Neural Networks", IEEE Access, 6, 2018].**
- **Code:** https://github.com/fudanxu/SAR-Colorization-v2]

# POLSAR Image Factorization

$$\mathbf{T}(n) = \sum_i \sigma_i^2(n)\mathbf{T}_i + \boldsymbol{\epsilon}$$



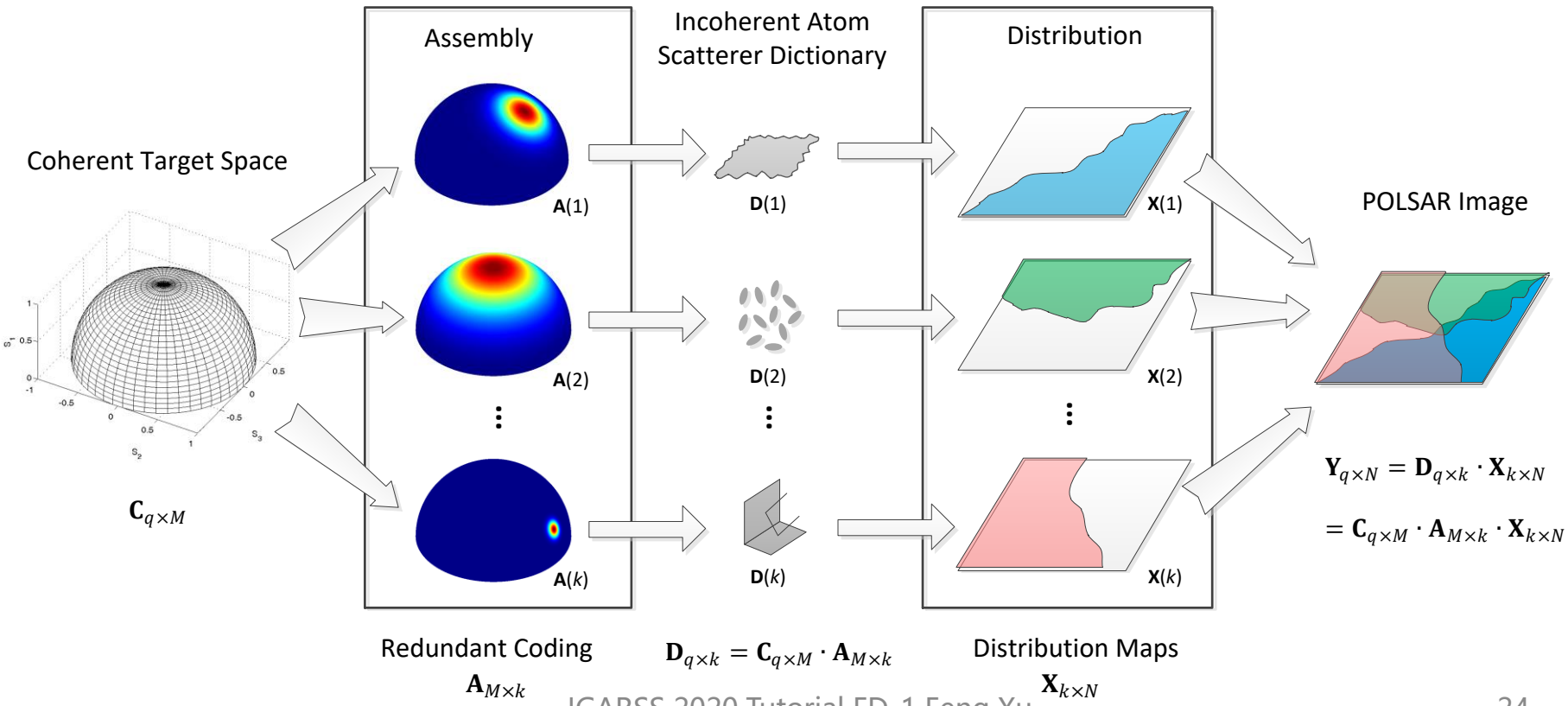

- Incoherent target decomposition
  - to solve for each pixel independently.

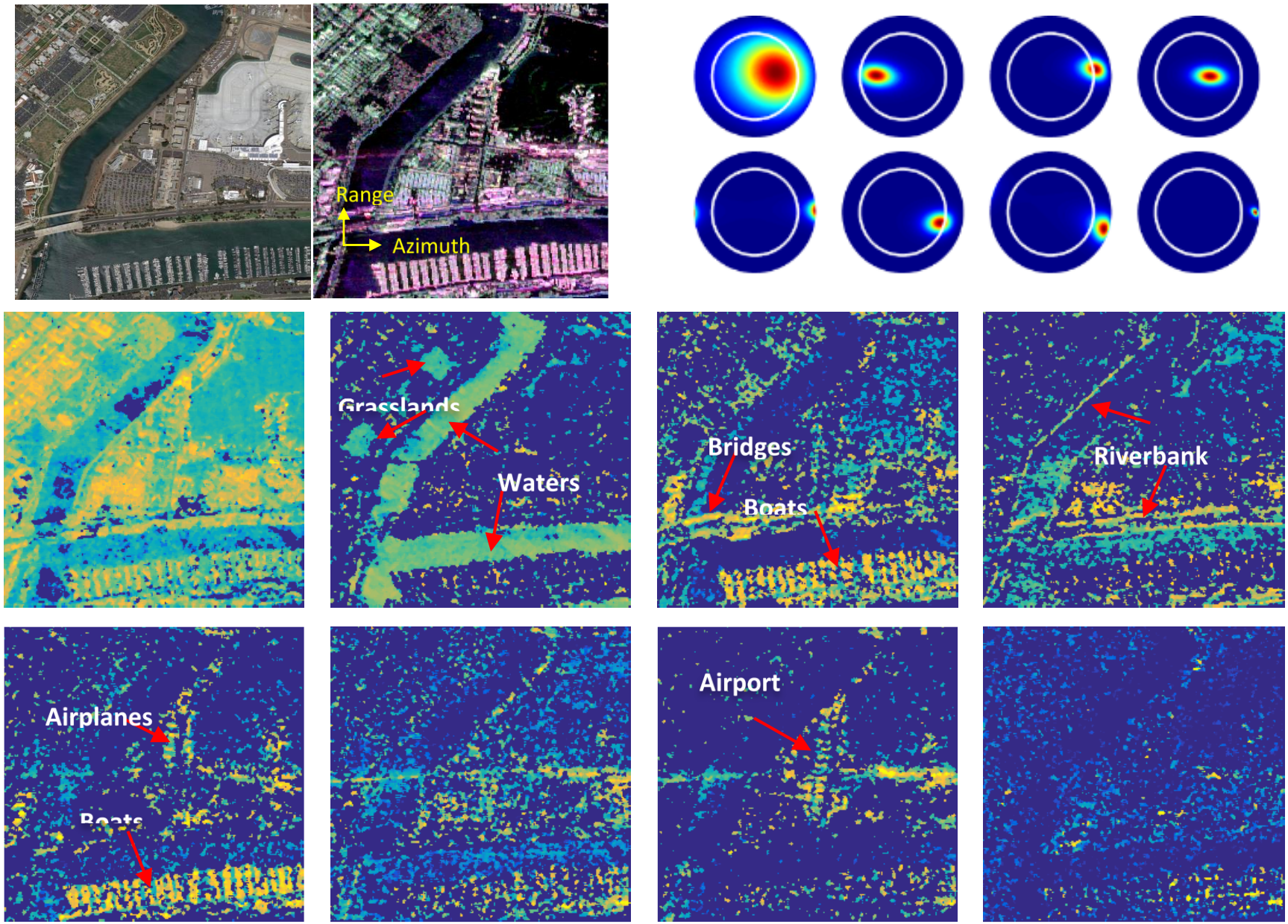
- POLSAR Image Factorization
  - To explore spatial correlation, we can solve it for a group of pixel collectively.
  - e.g. the whole image or a region of interest or a cluster of pixels/superpixels
  - Learn a dictionary of scatterers from the image

# POLSAR Image Factorization

$$\mathbf{Y}_{q\times N} = \mathbf{C}_{q\times M} \cdot \mathbf{A}_{M\times k} \cdot \mathbf{X}_{k\times N} + \epsilon$$

$$\text{s.t. } \operatorname{rank}\left[\mathcal{V}^{-1}\left(\mathbf{C}(i)_{q\times 1}\right)\right] = 1,\ \left\|\mathbf{C}(i)_{q\times 1}\right\|_2 = 1,\ \sum_m A_{m,i} = 1,\ i = 1,2,..$$

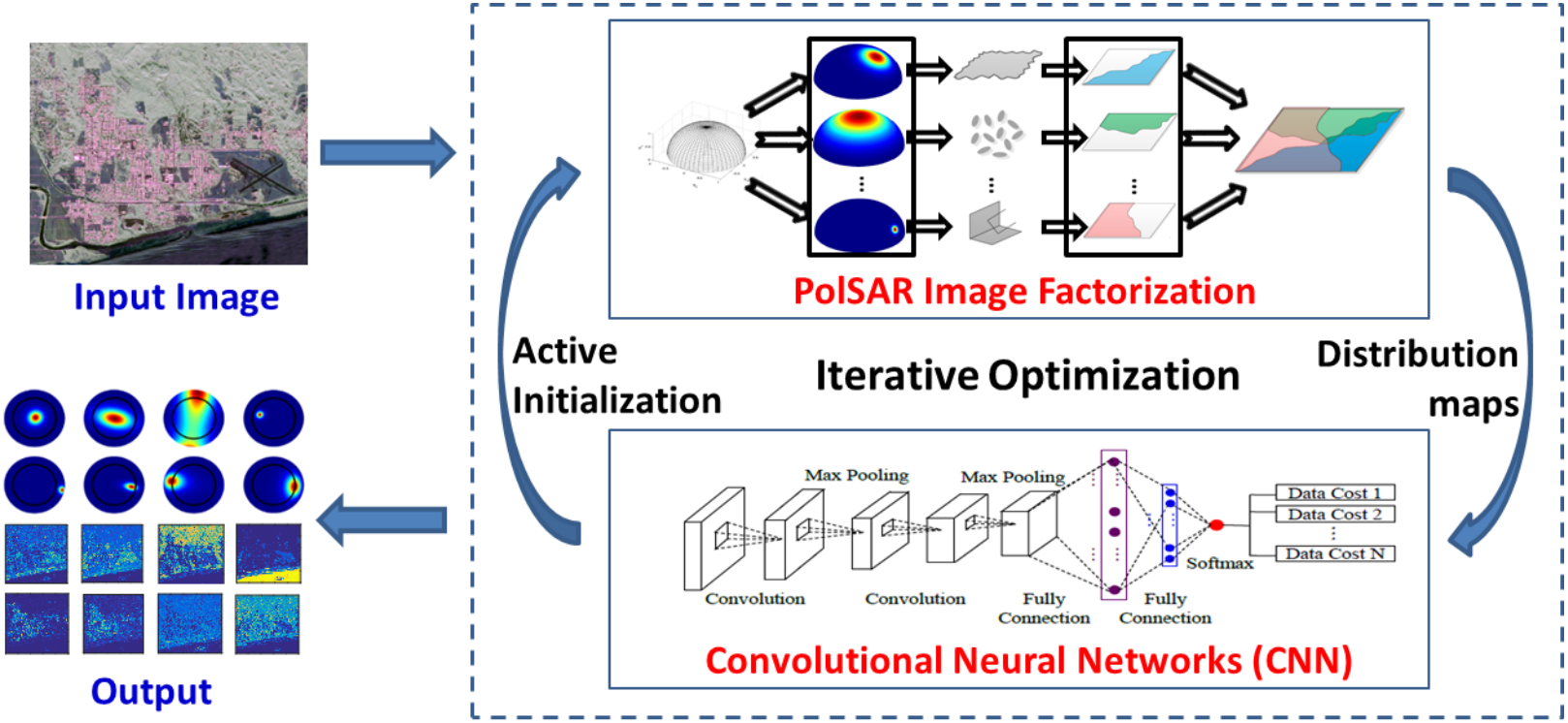





[Xu et al. "Polarimetric SAR Image Factorization", IEEE TGRS, 55(9) 2017. https://github.com/fudanxu/PolSARFactorization]

# Unsupervised PolSAR Image Factorization with Deep Convolutional Networks

- **DeepFactorization**

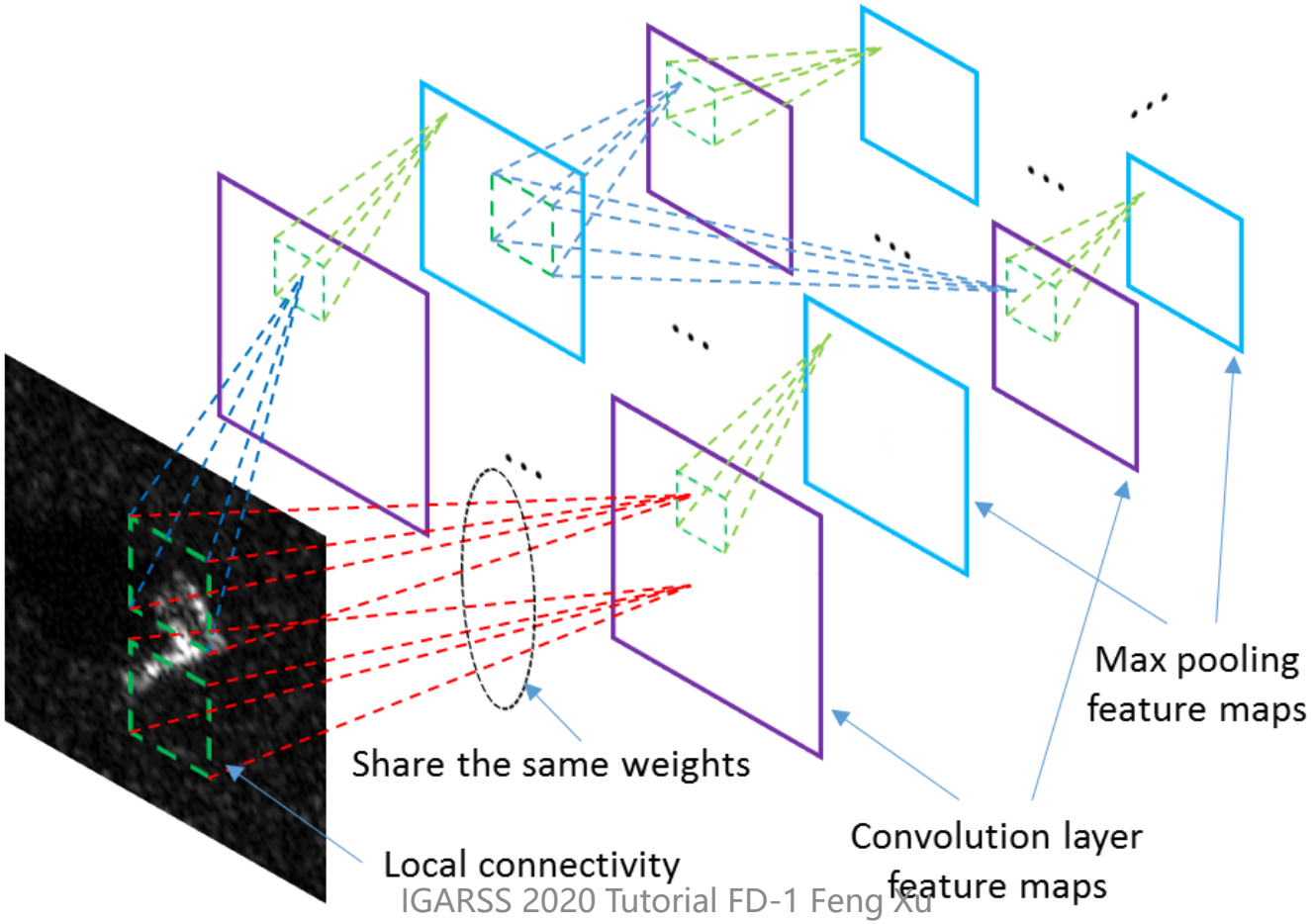



- [H. Bi et al., "Unsupervised PolSAR Image Factorization with Deep Convolutional Networks," IGARSS 2019 ]

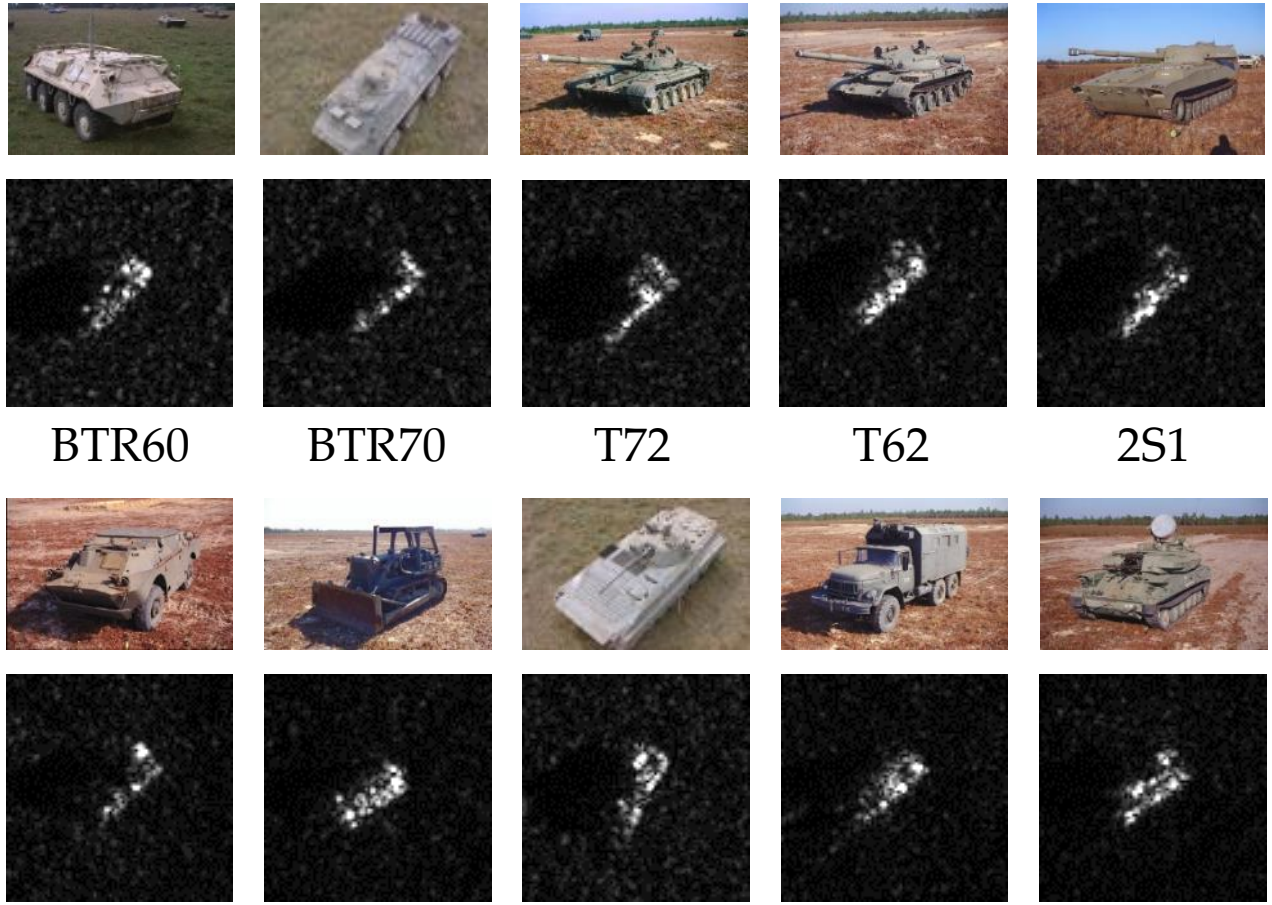
# SAR Target Recognition

# SAR ATR

- CNN Directly Applied on Grayscale SAR Image
- Issue: No sufficient Training Samples



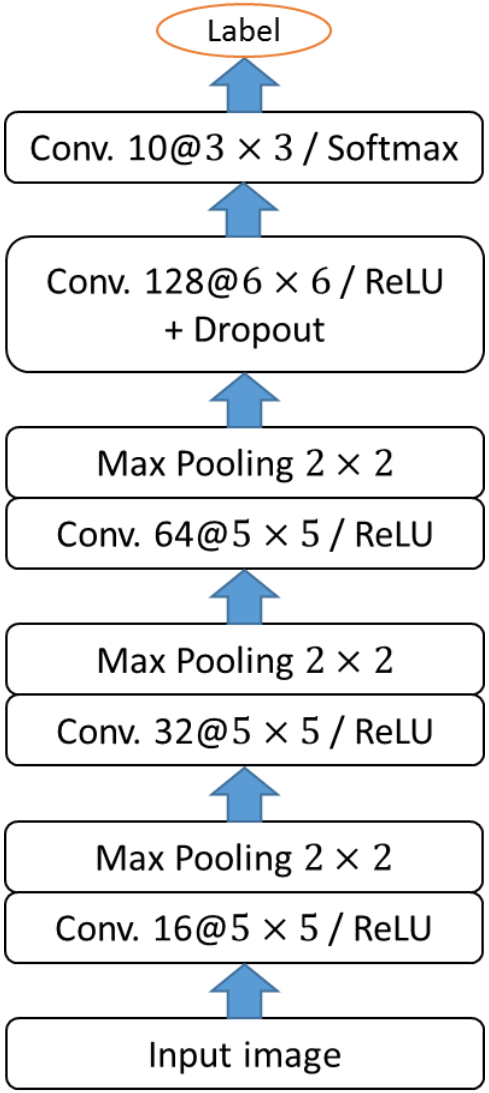
# The MSTAR Dataset




- From LL/SNL
- 0.3m× 0.3m resolution
- X-band, spotlight mode
- 0°~360° aspect angle coverage
- 17° depression angle for training 15° depression angle for testing

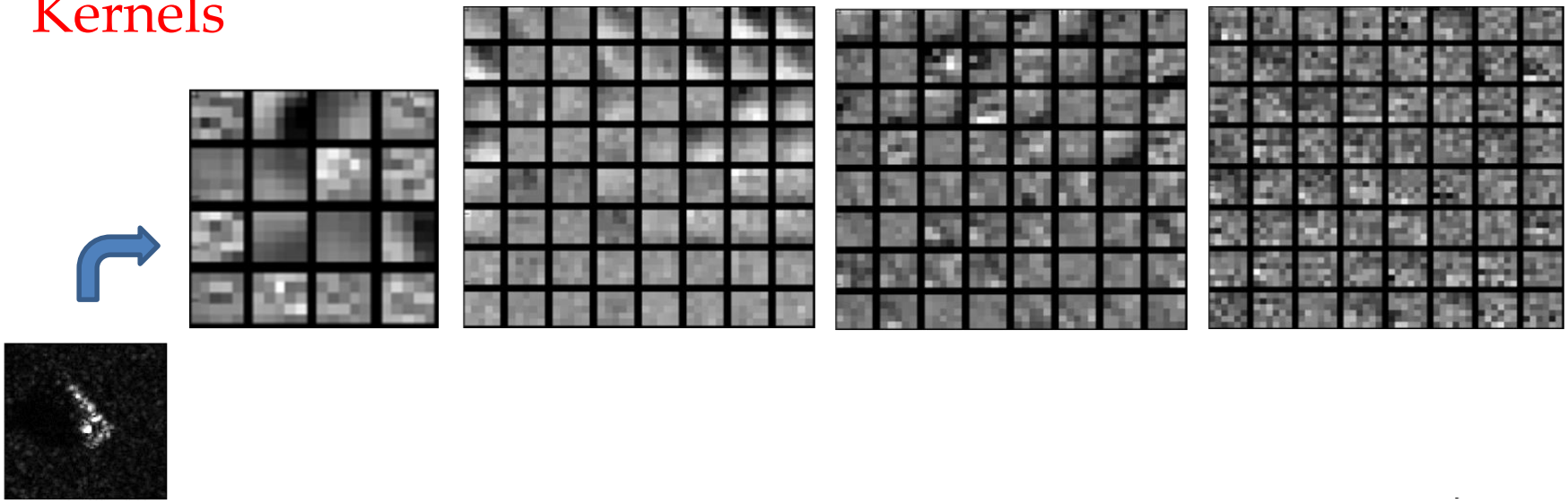
# All-Convolutional Neural Nets

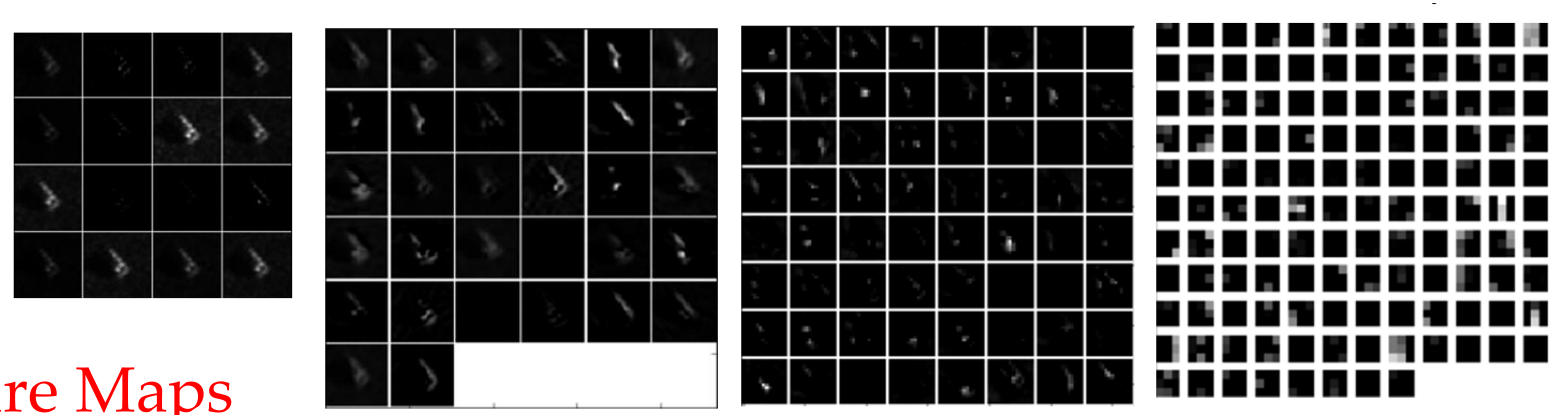
- To Avoid Overfitting
- Reduce Trainable Parameters
  - Replace Fully-Connected Layers with Conv. Layers
  - Keep Network Depth
- Data Augmentation
  - Translation/Distortion
- Architecture
  - Input 88x88x1
  - 3 Layers of Conv/Pool
  - 2 Conv +Dropout/Softmax
  - Output 10 Classes



# Learned Kernels and Feature Maps

Kernels





Feature Maps

# MSTAR Result

- Standard operating conditions (SOC):

| | | Train | | Test | |
|---|---|---|---|---|---|
| Class | Serial No. | Depression | No. Images* | Depression | No. Images |
| BMP-2 | 9563 | 17° | 233 | 15° | 196 |
| BTR-70 | c71 | 17° | 233 | 15° | 196 |
| T-72 | 132 | 17° | 232 | 15° | 196 |
| BTR-60 | k10yt7532 | 17° | 256 | 15° | 195 |
| 2S1 | b01 | 17° | 299 | 15° | 274 |
| BRDM-2 | E-71 | 17° | 298 | 15° | 274 |
| D7 | 92v13015 | 17° | 299 | 15° | 274 |
| T-62 | A51 | 17° | 299 | 15° | 273 |
| ZIL-131 | E12 | 17° | 299 | 15° | 274 |
| ZSU-234 | d08 | 17° | 299 | 15° | 274 |

*~2500 after augmentation

# Confusion Matrix

| Class | BMP-2 | BTR-70 | T-72 | BTR-60 | 2S1 | BRDM -2 | D7 | T62 | ZIL131 | ZSU23 4 | $P_{cc}(\%)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BMP-2 | 194 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 98.98 |
| BTR-70 | 0 | 195 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 99.49 |
| T-72 | 0 | 0 | 196 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| BTR-60 | 1 | 0 | 0 | 188 | 0 | 0 | 0 | 1 | 1 | 4 | 96.41 |
| 2S1 | 0 | 0 | 0 | 0 | 269 | 4 | 0 | 0 | 0 | 1 | 98.18 |
| BRDM -2 | 0 | 0 | 0 | 0 | 0 | 272 | 0 | 0 | 0 | 2 | 99.27 |
| D7 | 0 | 0 | 0 | 0 | 0 | 0 | 272 | 1 | 1 | 0 | 99.27 |
| T-62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 272 | 1 | 0 | 99.64 |
| ZIL-131 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 273 | 1 | 99.64 |
| ZSU-234 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 273 | 99.64 |
| Total | | | | | | | | | | | 99.13 |

# Generalization Capability Test 1

- Extended operating conditions (EOC):

- EOC-1： Incident Angle (30deg)

| Class | Serial No. | Depression | No. Images |
|---|---|---|---|
| 2S1 | b01 | 30° | 288 |
| BRDM-2 | E-71 | 30° | 287 |
| T-72 | A64 | 30° | 288 |
| ZSU-234 | d08 | 30° | 288 |

| Class | 2S1 | BRDM-2 | T-72 | ZSU-234 | $P_{cc}(\%)$ |
|---|---|---|---|---|---|
| 2S1 | 269 | 19 | 0 | 0 | 93.40 |
| BRDM-2 | 1 | 285 | 0 | 1 | 99.30 |
| T-72 | 3 | 18 | 263 | 4 | 91.32 |
| ZSU-234 | 9 | 9 | 13 | 257 | 89.24 |
| Total | | | | | 93.31 |

# Generalization Capability Test 2

- EOC-2: Configuration Change

T72-A04

T72-A05

T72-A07

T72-A10

| Class | Serial No. | BMP-2 | BRDM-2 | BTR-70 | T-72 | $P_{cc}(\%)$ |
|---|---|---|---|---|---|---|
| BMP-2 | 9566 | 412 | 1 | 2 | 13 | 96.26 |
| | c21 | 423 | 1 | 0 | 5 | 98.60 |
| T-72 | 812 | 16 | 0 | 0 | 410 | 96.24 |
| | A04 | 6 | 2 | 0 | 565 | 98.60 |
| | A05 | 1 | 0 | 0 | 572 | 99.83 |
| | A07 | 3 | 0 | 0 | 570 | 99.48 |
| | A10 | 0 | 0 | 0 | 567 | 100 |
| Total | | | | | | 98.60 |

# Generalization Capability Test 3

- EOC-3: Sub-types

T72-A32

T72-A62

T72-A63

T72-A64

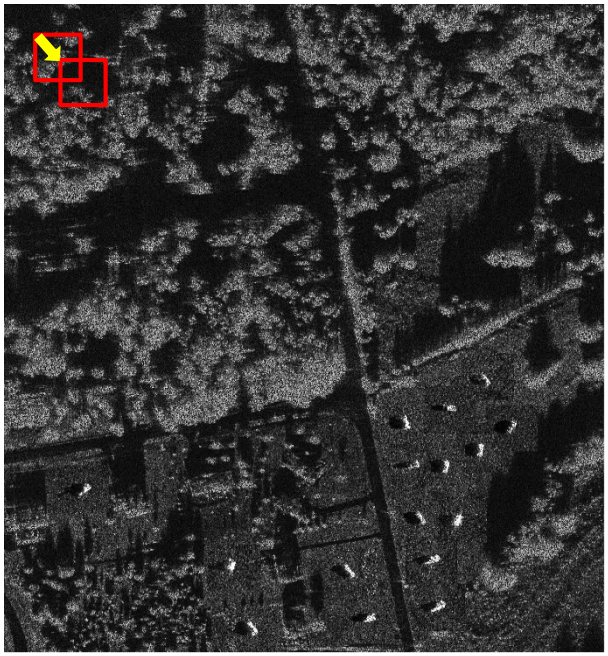
| Class | Serial No. | BMP-2 | BRDM-2 | BTR-70 | T-72 | $P_{cc}(\%)$ |
|---|---|---|---|---|---|---|
| T-72 | S7 | 7 | 0 | 0 | 412 | 98.33 |
| | A32 | 11 | 0 | 0 | 561 | 98.08 |
| | A62 | 1 | 0 | 0 | 572 | 99.83 |
| | A63 | 6 | 0 | 0 | 567 | 98.95 |
| | A64 | 4 | 0 | 0 | 569 | 99.30 |
| Total | | | | | | 98.93 |

# Comparisons

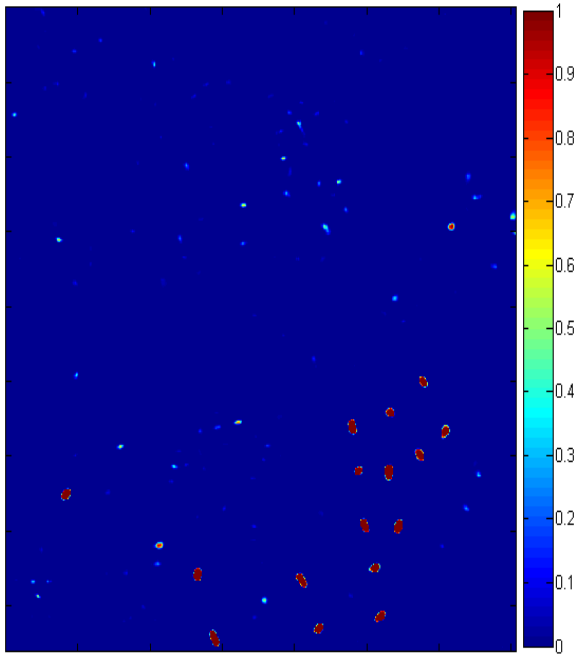
| Method | SOC (%) | EOC-1 (%) | EOC-2 (%) |
|---|---|---|---|
| EMACH | 88 | 77 | 68 |
| SVM | 90 | 81 | 75 |
| Cond Gauss | 92 | 80 | 76 |
| AdaBoost | 92 | 82 | 78 |
| BCS | 92 | 89.6 | - |
| MSRC | 93.6 | 98.49 | - |
| IGT | 95 | 85 | 80 |
| M-PMC | 98.81 | - | 97.31 |
| A-ConvNets | 99.13 | 93.31 | 98.93 |

# End-to-End Demonstration

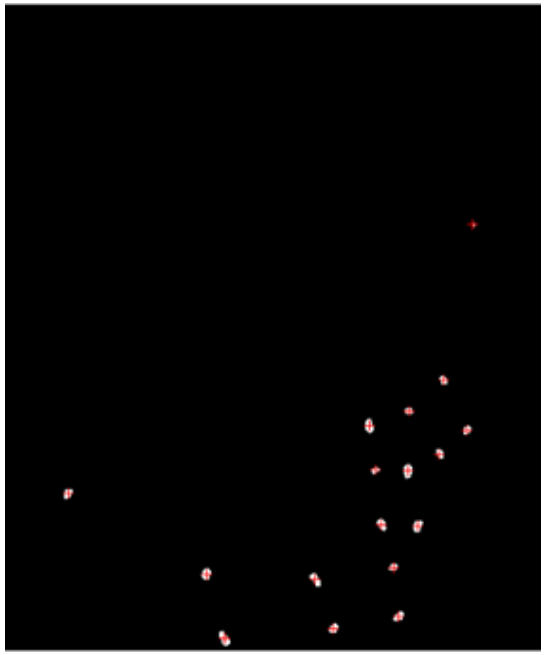
Detect → Discriminate → Classification



(a) Sliding Window

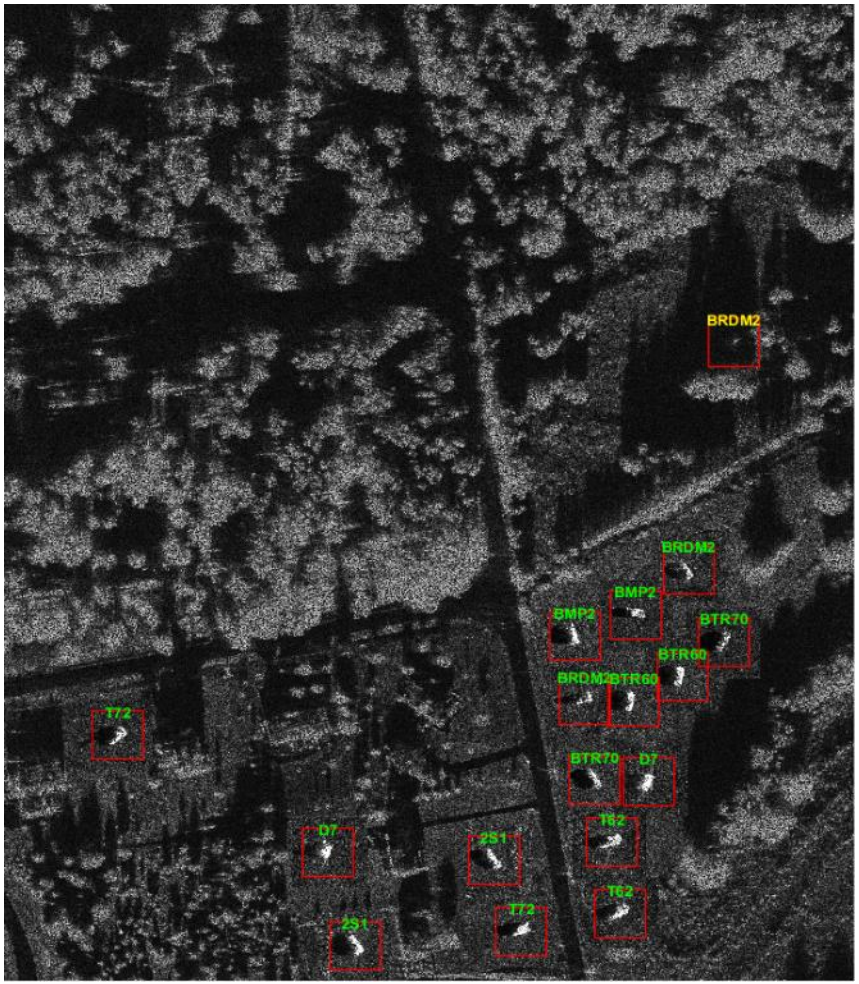


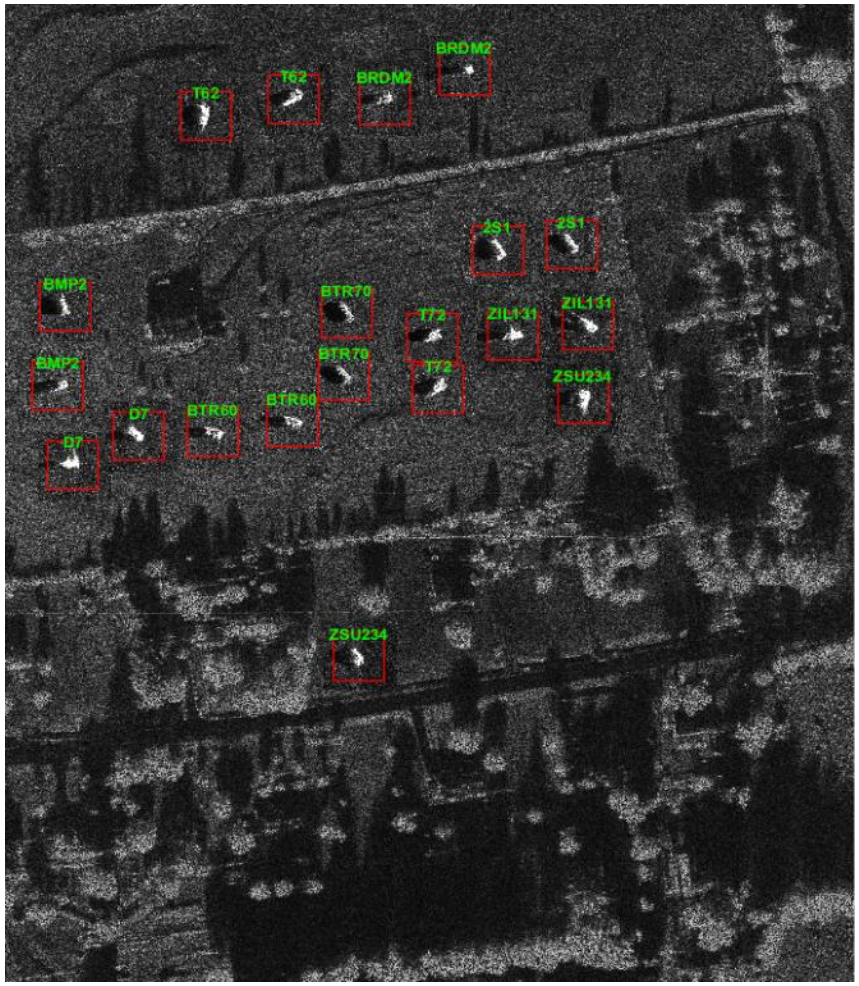

(b) Target/Background Classifier



(c) Connected Region Extraction

# SAR ATR: Recognition

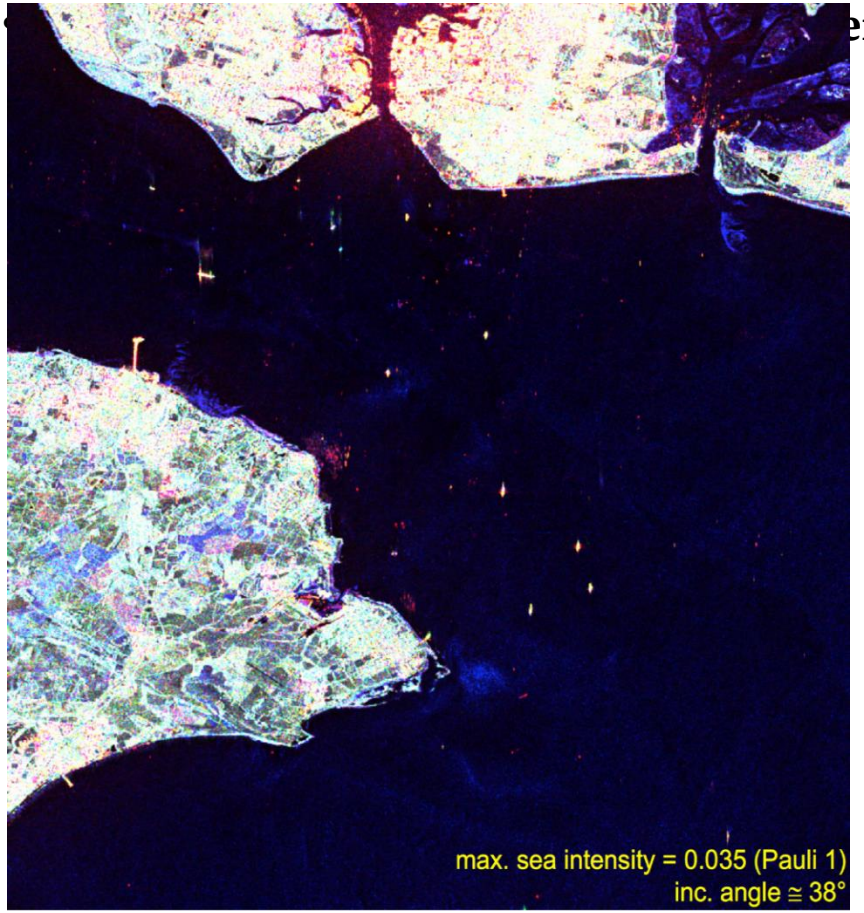


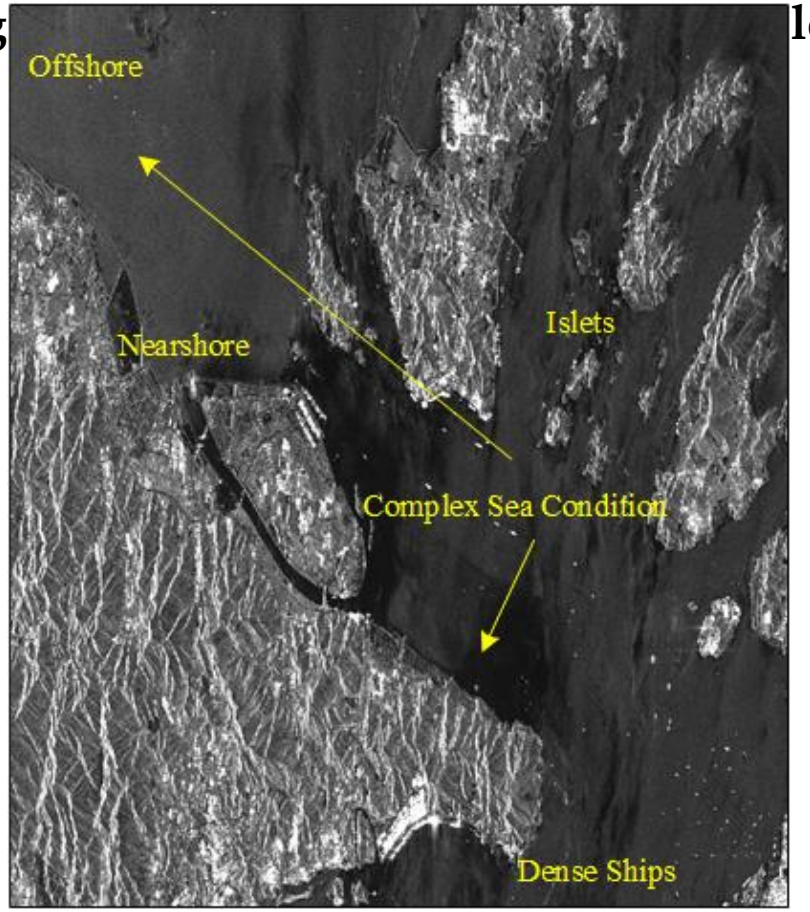




- **S. Chen, H. Wang, F. Xu, Y.Q. Jin, "Target Classification Using the Deep Convolutional Networks for SAR Images", *IEEE Trans Geosci. Remote Sens.*, vol. 54, no. 8, 2016.**
- [code: https://github.com/fudanxu/MSTAR-AConvNet]

# Ship Detection

- **Monitoring: sea target recognition is of critical importance for search and rescue etc.**



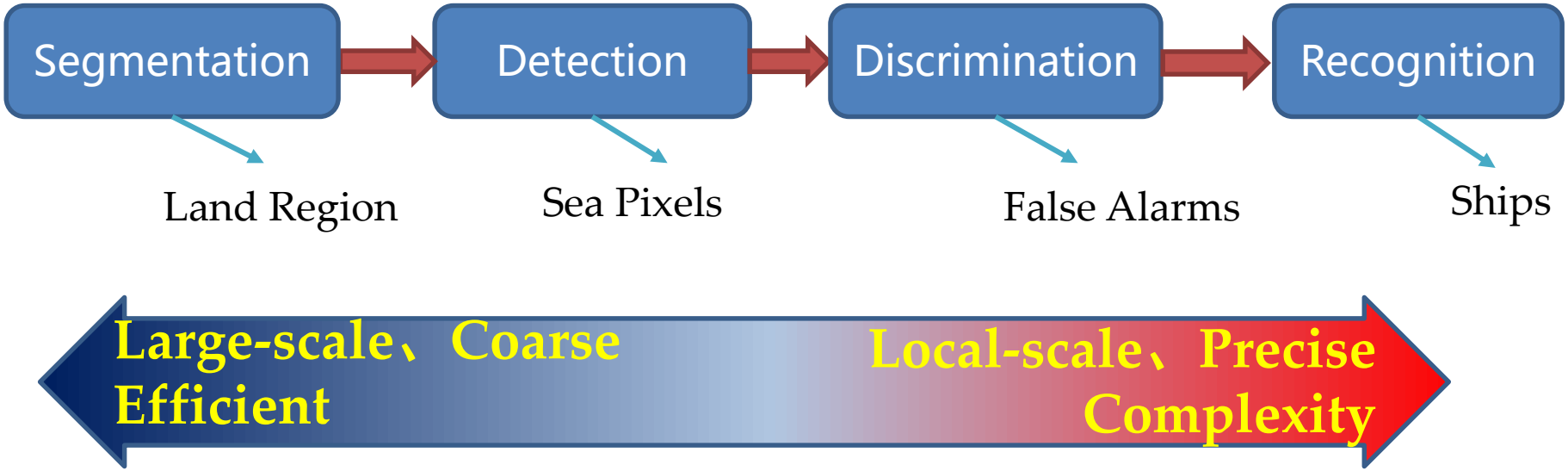




©MDA 2011 – All Rights Reserved.

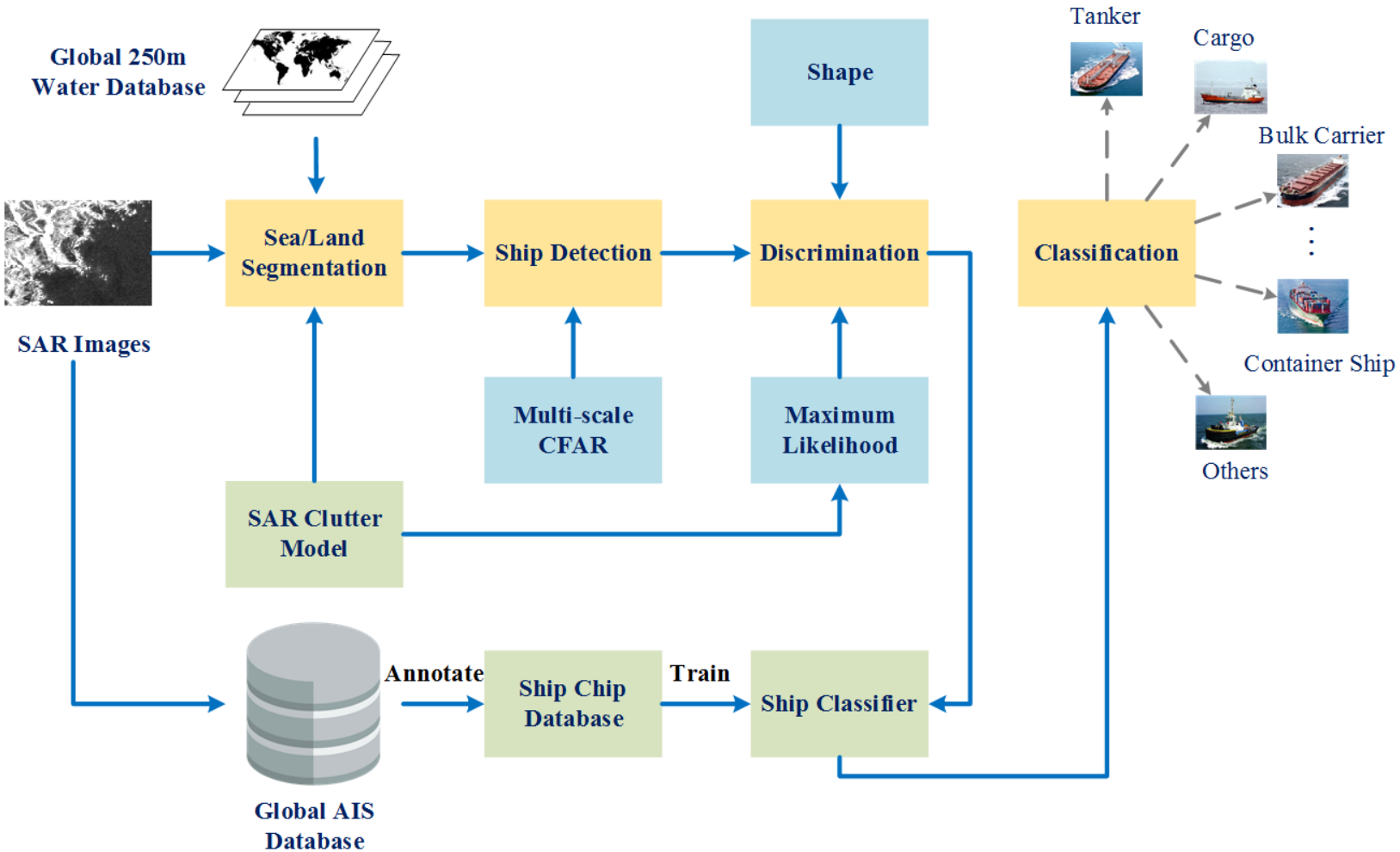
# Ship Target ATR

**Four main steps for ship detection:**

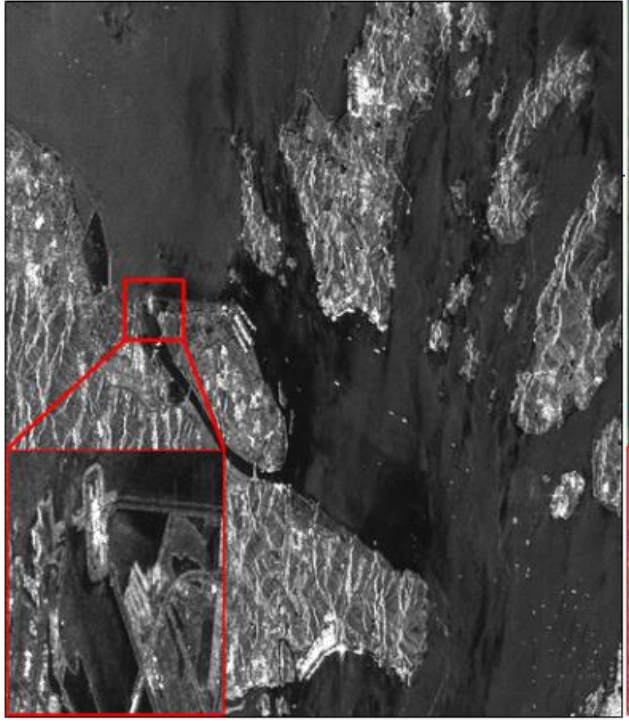
- **Sea/Land Segmentation： Global 250m water mask-aided segmentation**
- **Ship detection： Multi-scale CFAR detection**
- **Discrimination of false alarm： Shape-based Maximum Likelihood**
- **Classification: Convolutional Neural Network (CNN)**

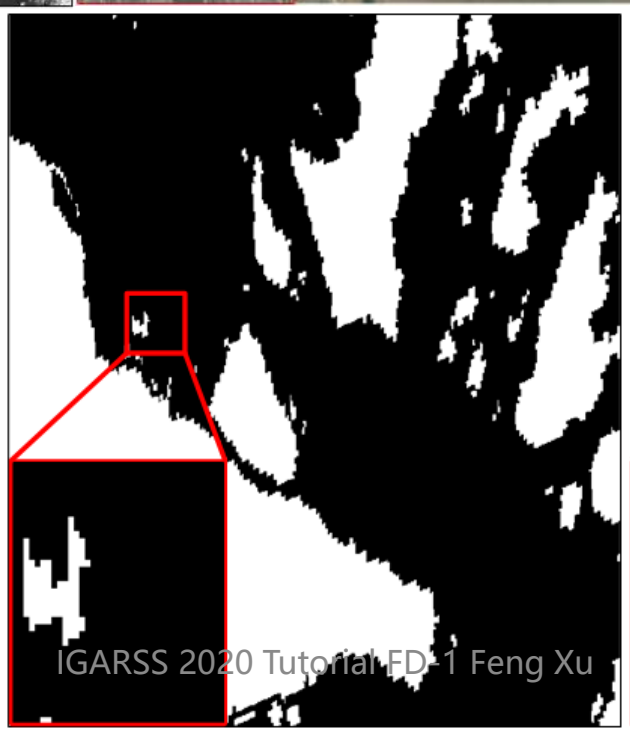


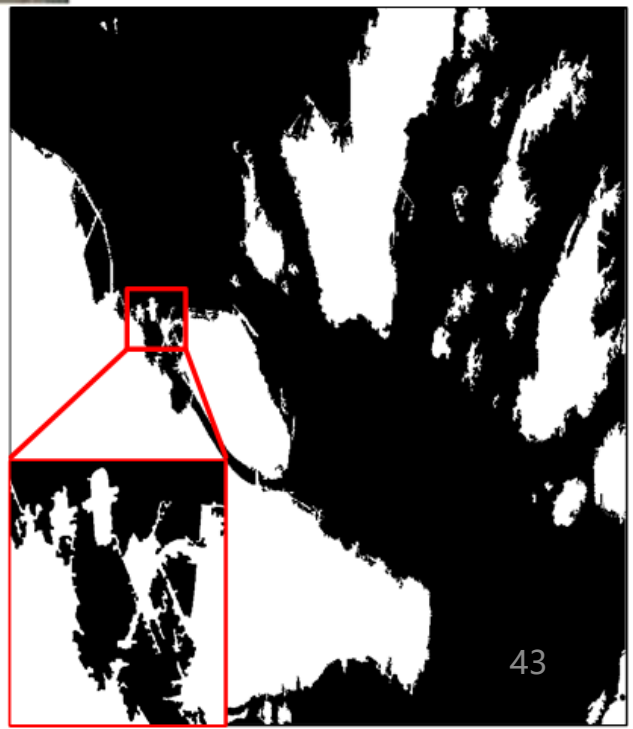
# SAR Ship ATR Framework



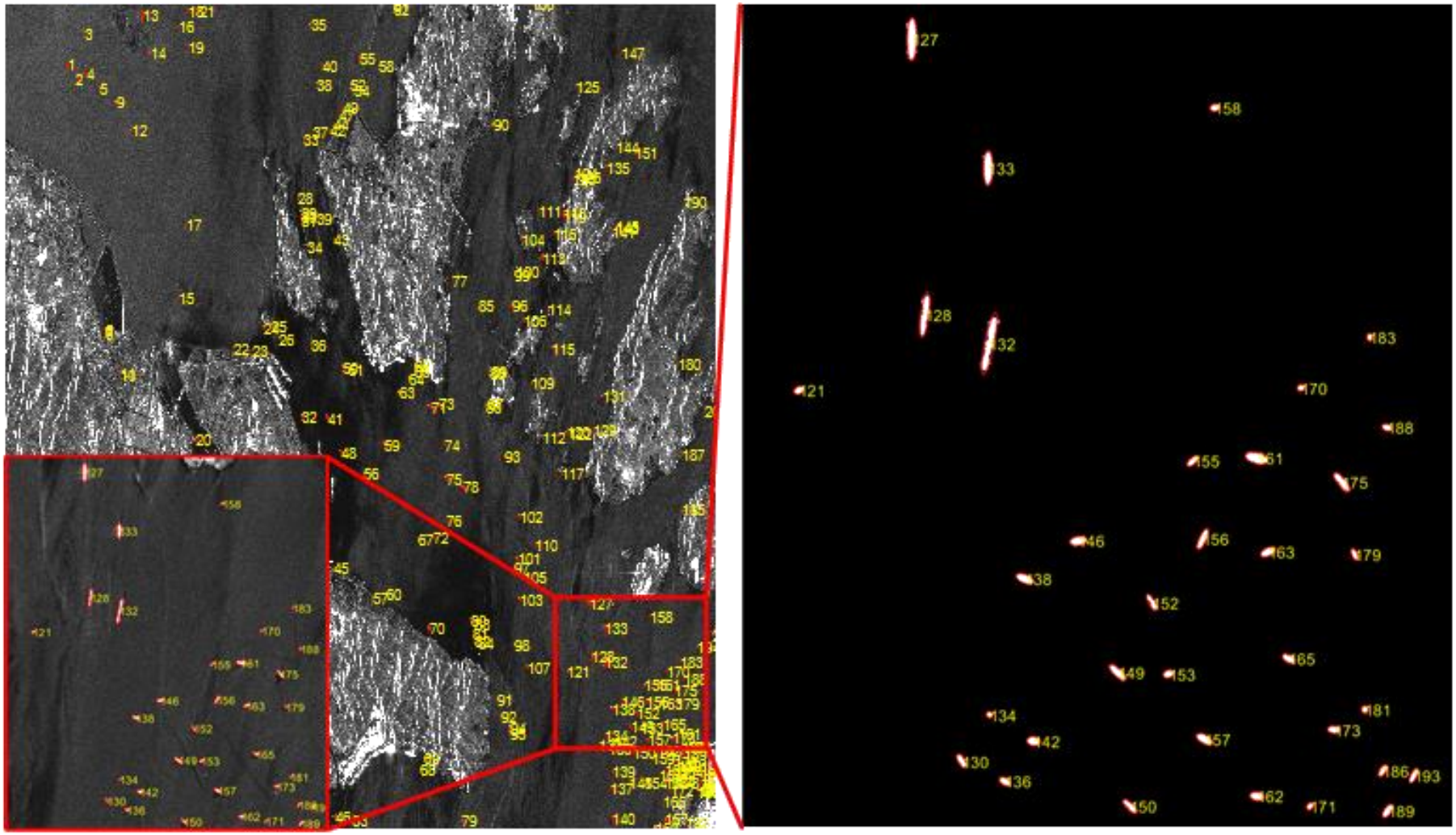

- **W. Ao, Feng Xu, Y. Li, H. Wang, "Detection and Discrimination of Ship Targets in Complex Background from Spaceborne ALOS2 SAR Image", IEEE JSTARS, 11( 2), pp. 536-550, 2018**







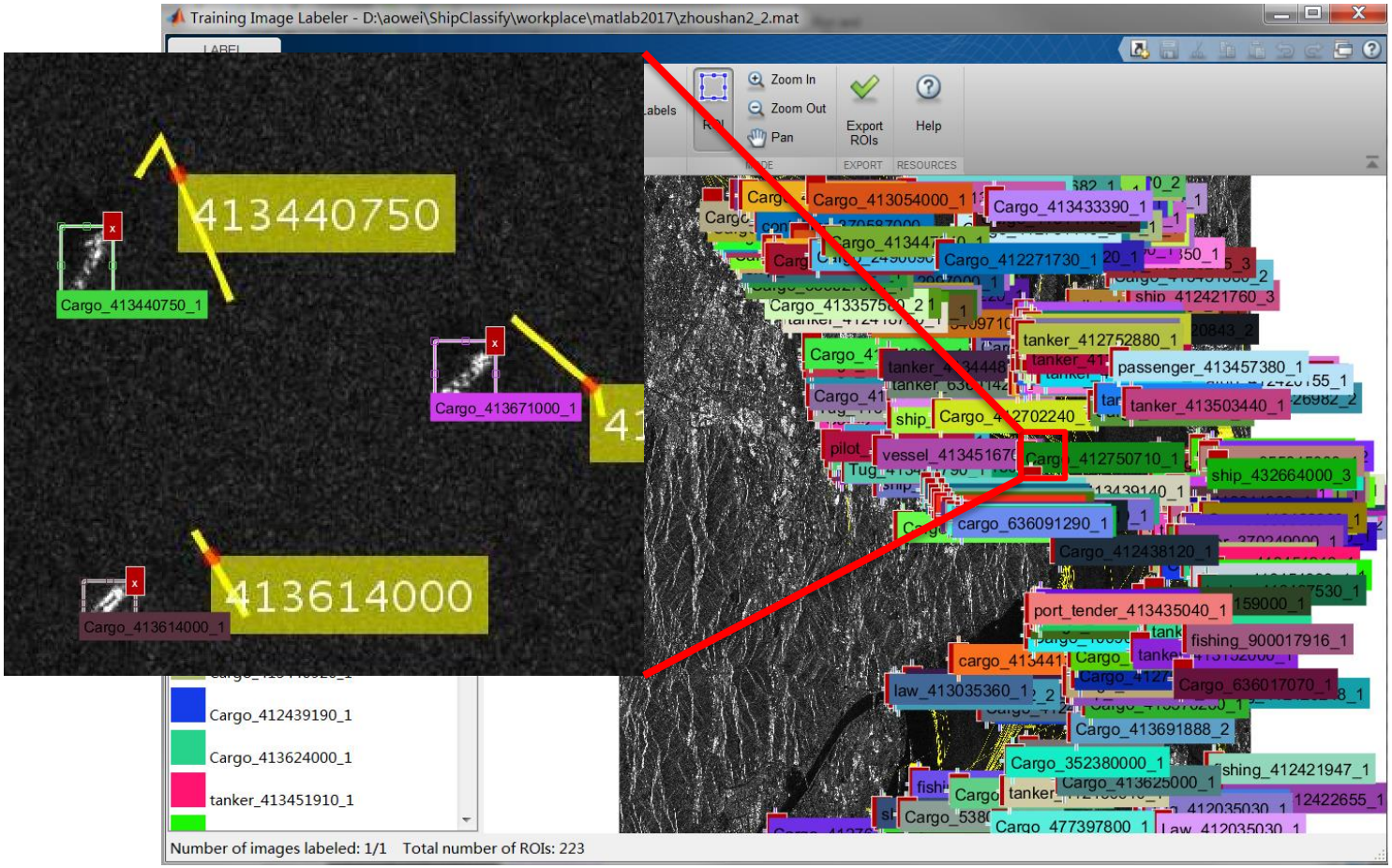
# Ship Detection Result



**Detection result** **Enlarged binary image**

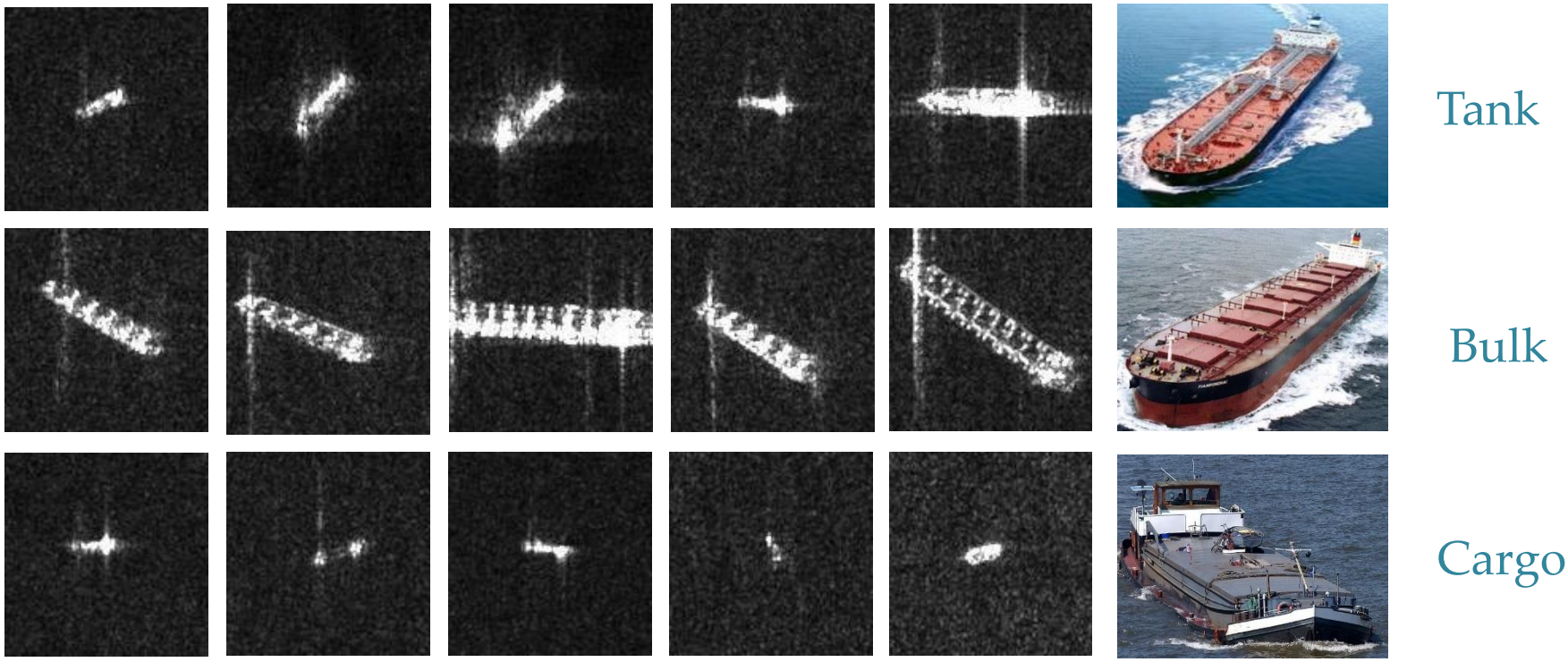
# Ship Target Labeling

**Matchup databased include manual labeling of detected ship and the corresponding track of a know MMSI.**



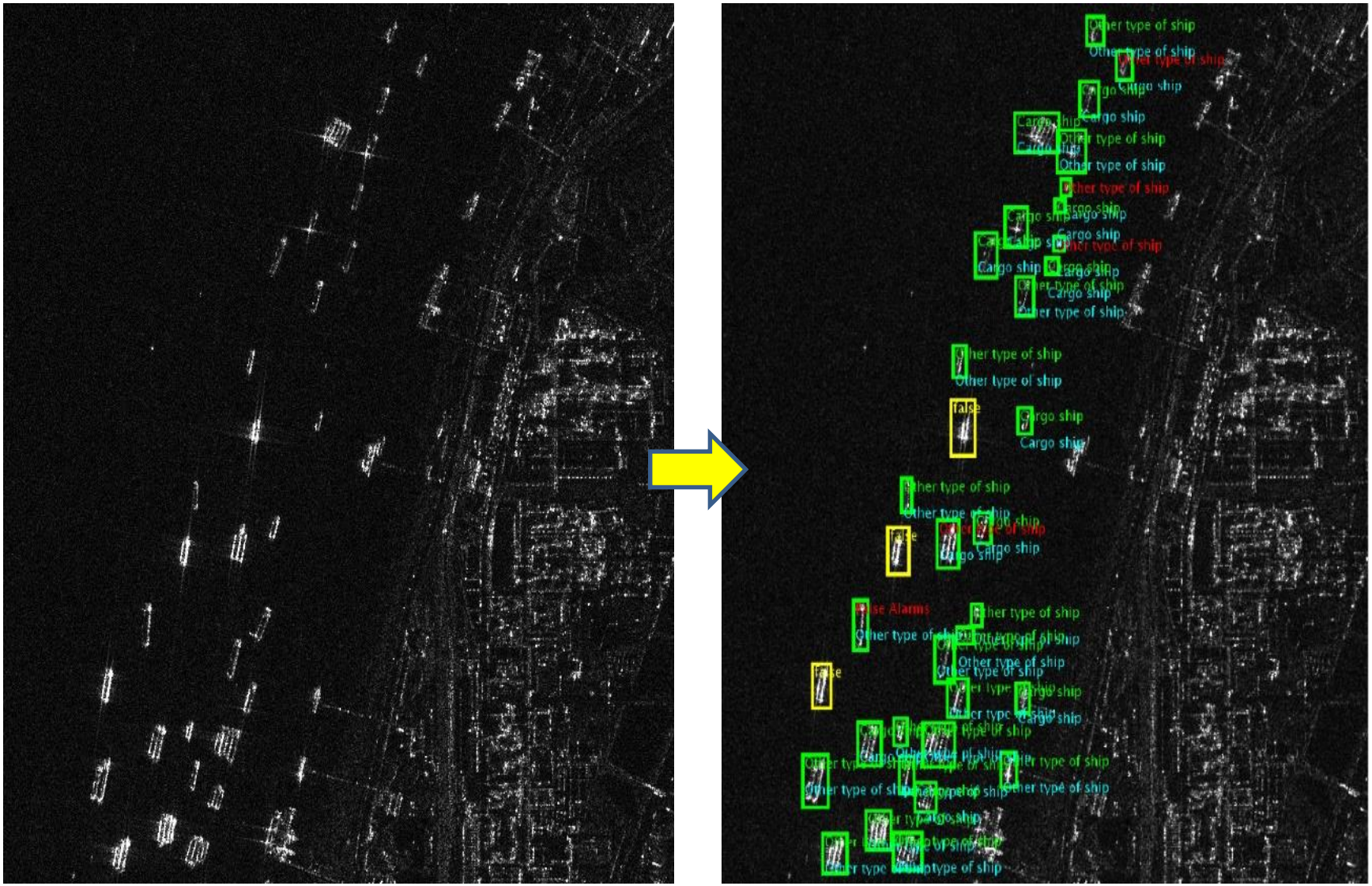
# High-Resolution SAR Ship Database

**Some samples of ALOS2 database**



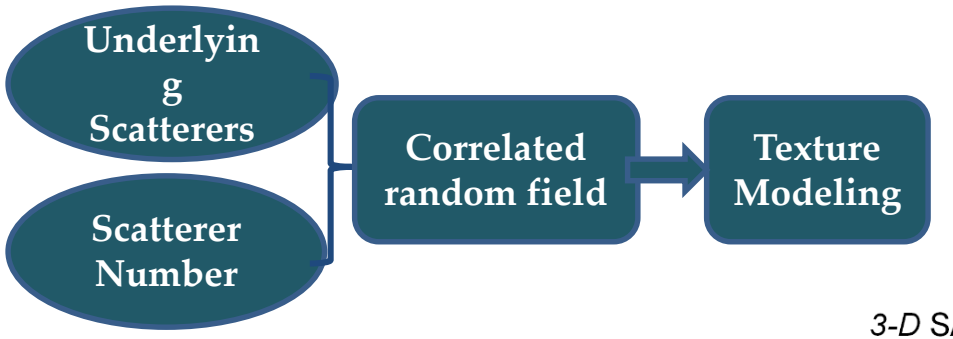

- **Hou X. et al., "FUSAR-Ship: Building a High-resolution SAR-AIS Matchup Dataset of Gaofen-3 for Ship Detection and Recognition", Science of China, Information Series, 2020 under review**

# Ship ATR



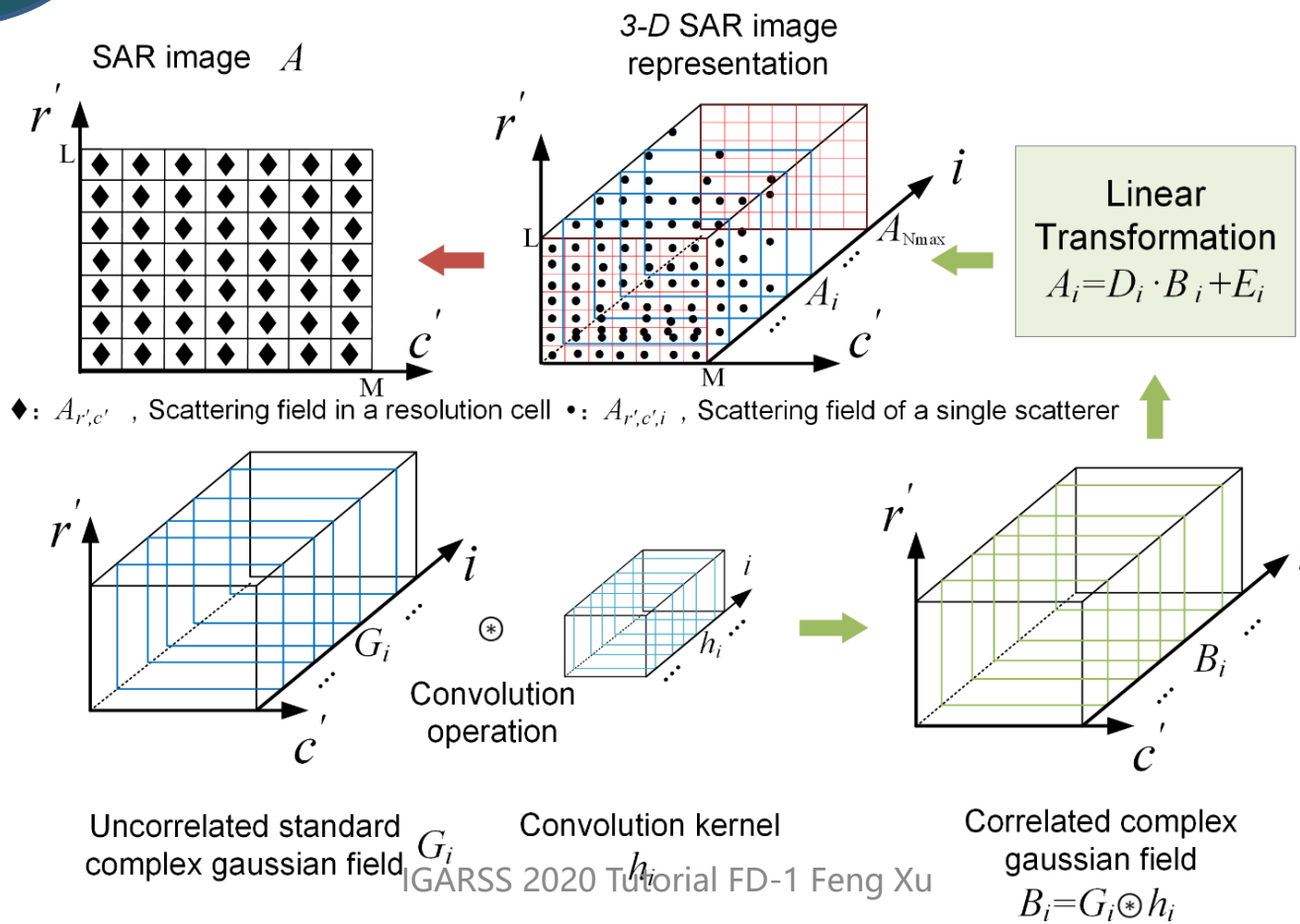
# Representation Learning

# Speckle Representation Model



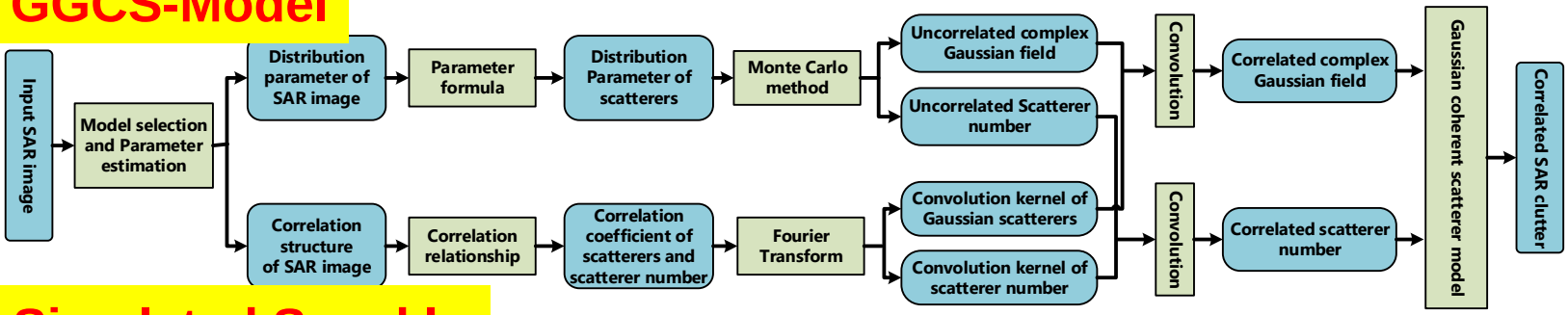

$$\rho_{RR}(\tau) = \frac{E[min\{N(t), N(t-\tau)\}]}{\mu_N} E[\rho_{xx}(i,\tau)]$$

$$\rho_{II}(\tau) = \frac{E(min\{N_0, N_\tau\} min\{N_0, N_\tau\})\rho_{xx}^2(\tau) + \rho_{NN}(\tau)\sigma_N^2}{2\sigma_N^2 + \mu_N^2}$$

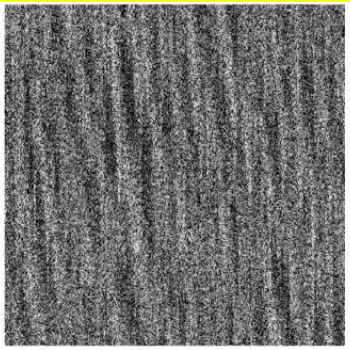


# GGCS：Generalized Gaussian Coherent Scatterer

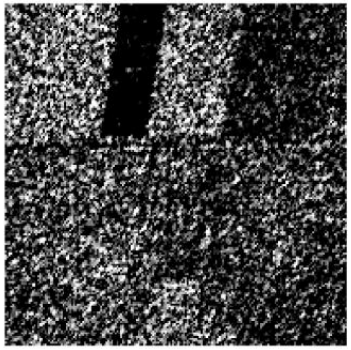
## GGCS-Model



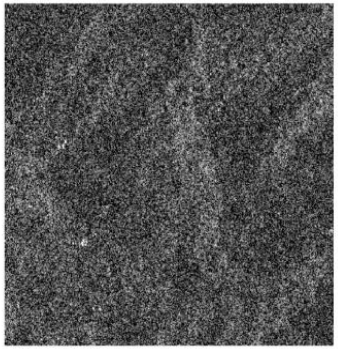

## Simulated Speckle



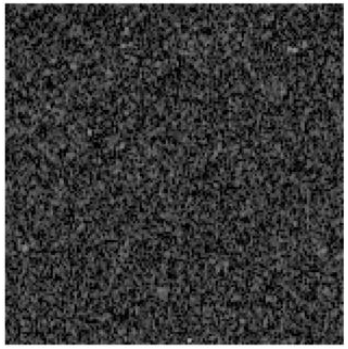
Sea3-real



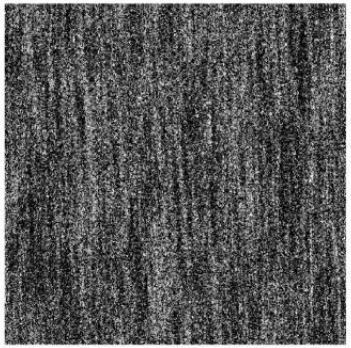
Farmland-real



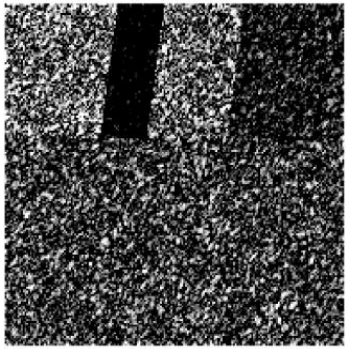
Forest2-real



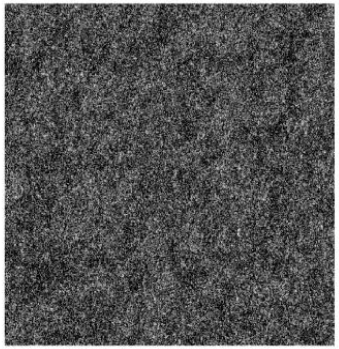
Road-real



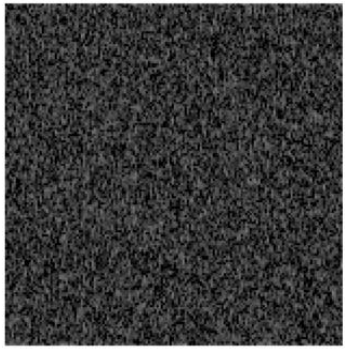
Sea3-simulated



Farmland-simulated

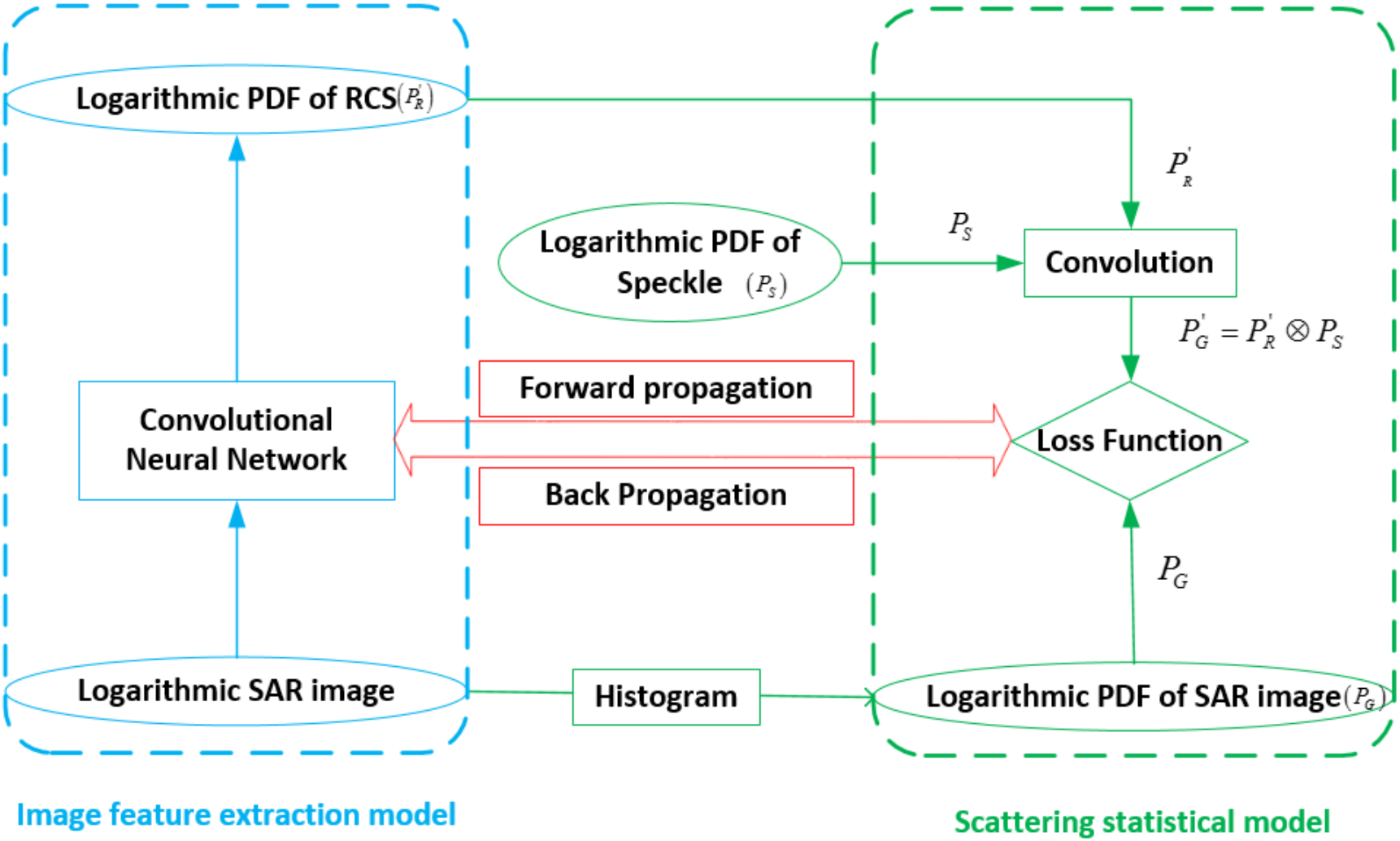


Forest2-simulated

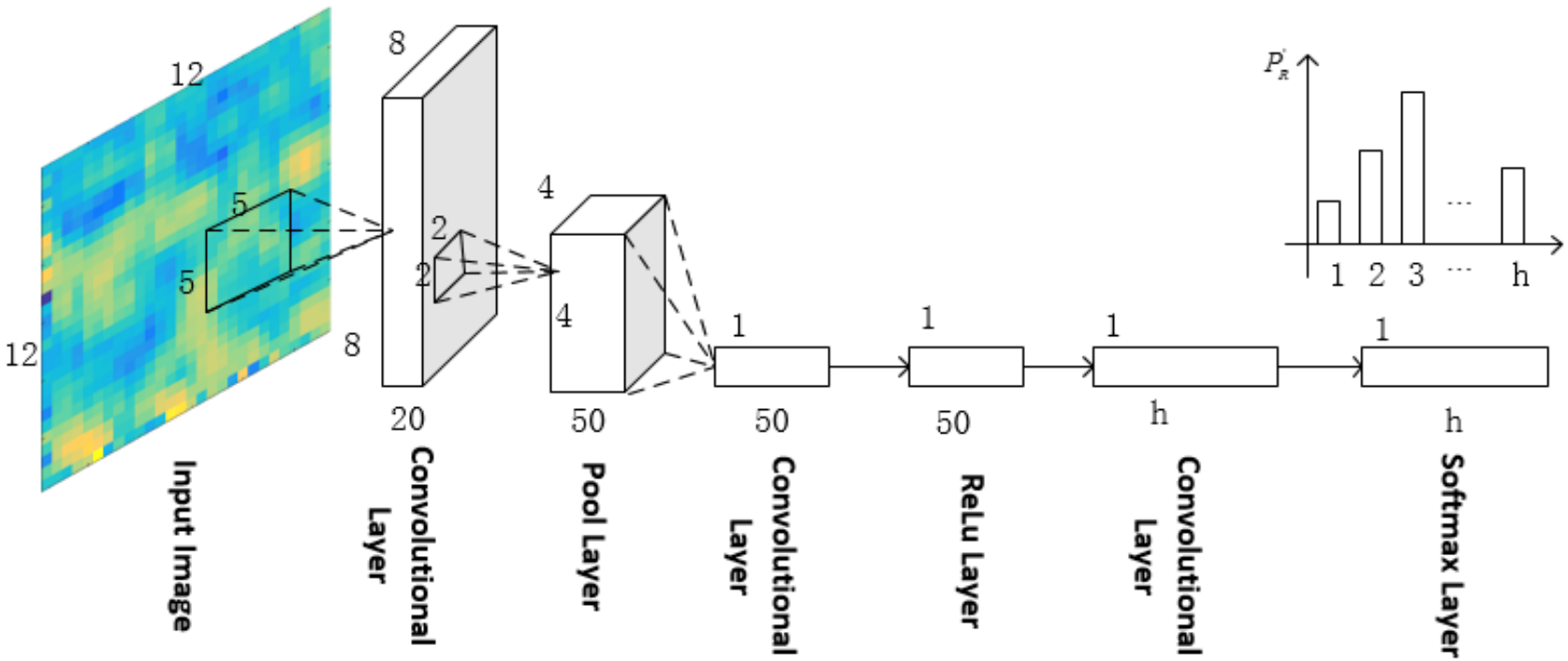


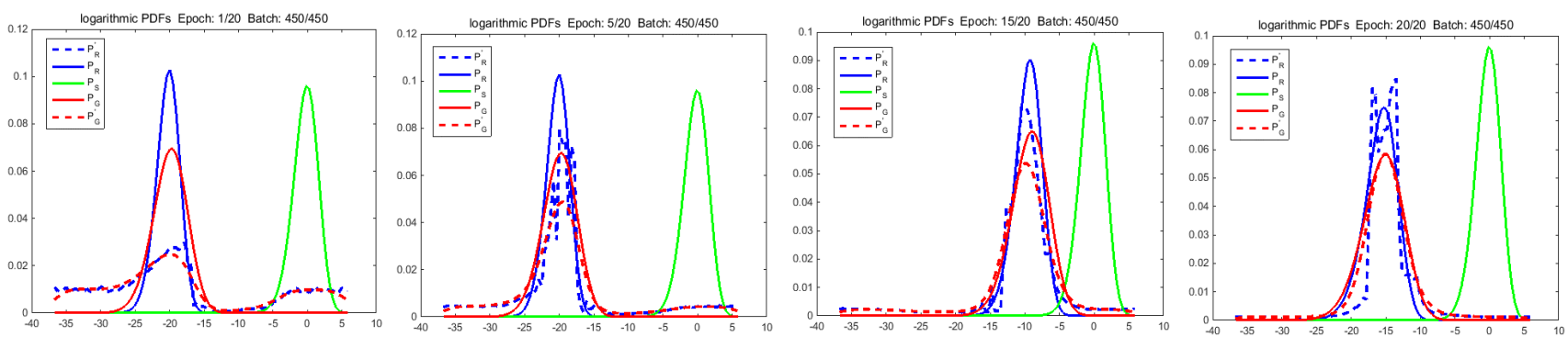
Road-simulated

# Despeckling Neural Networks



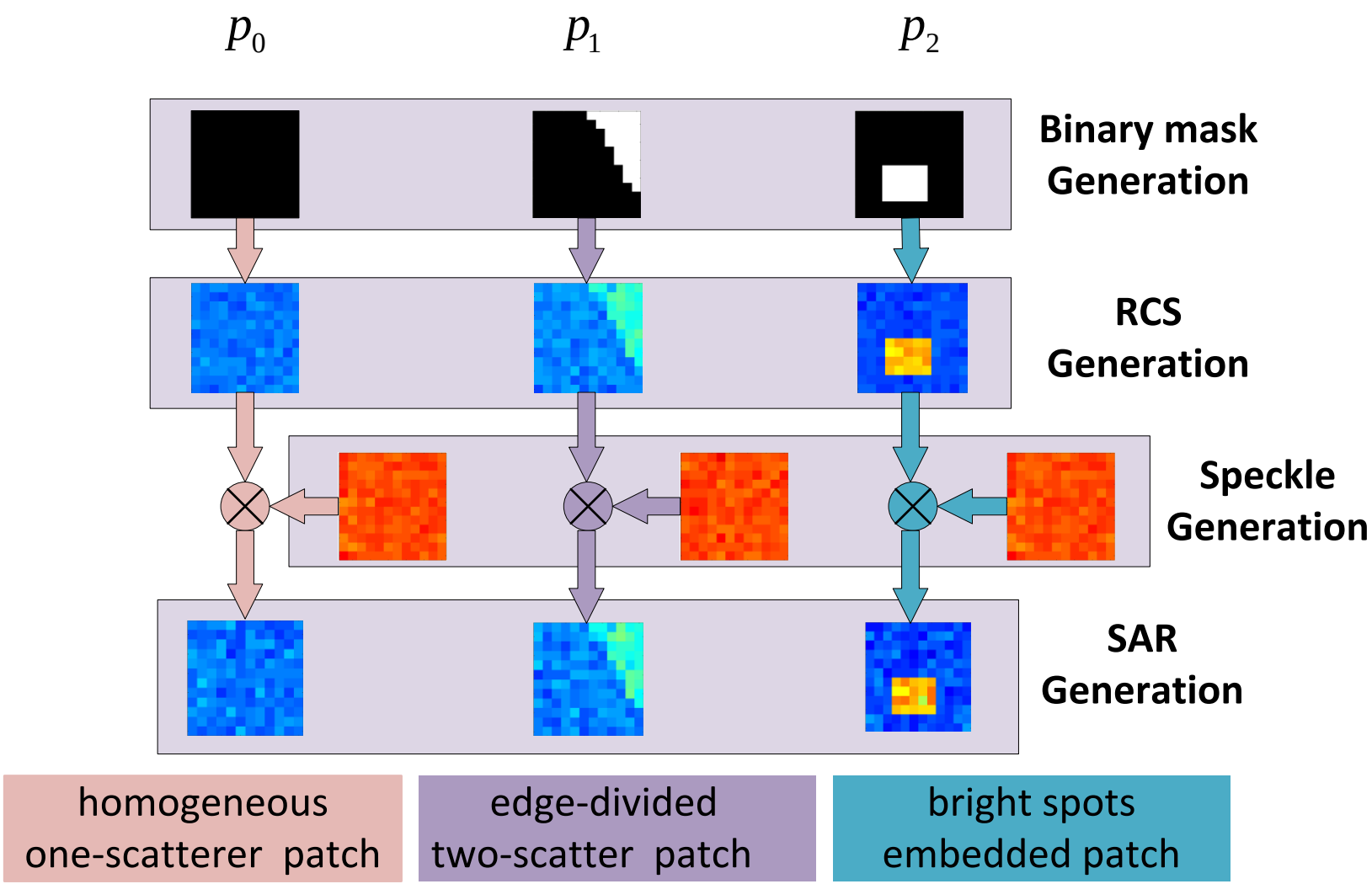
# Train CNN to estimate RCS distribution



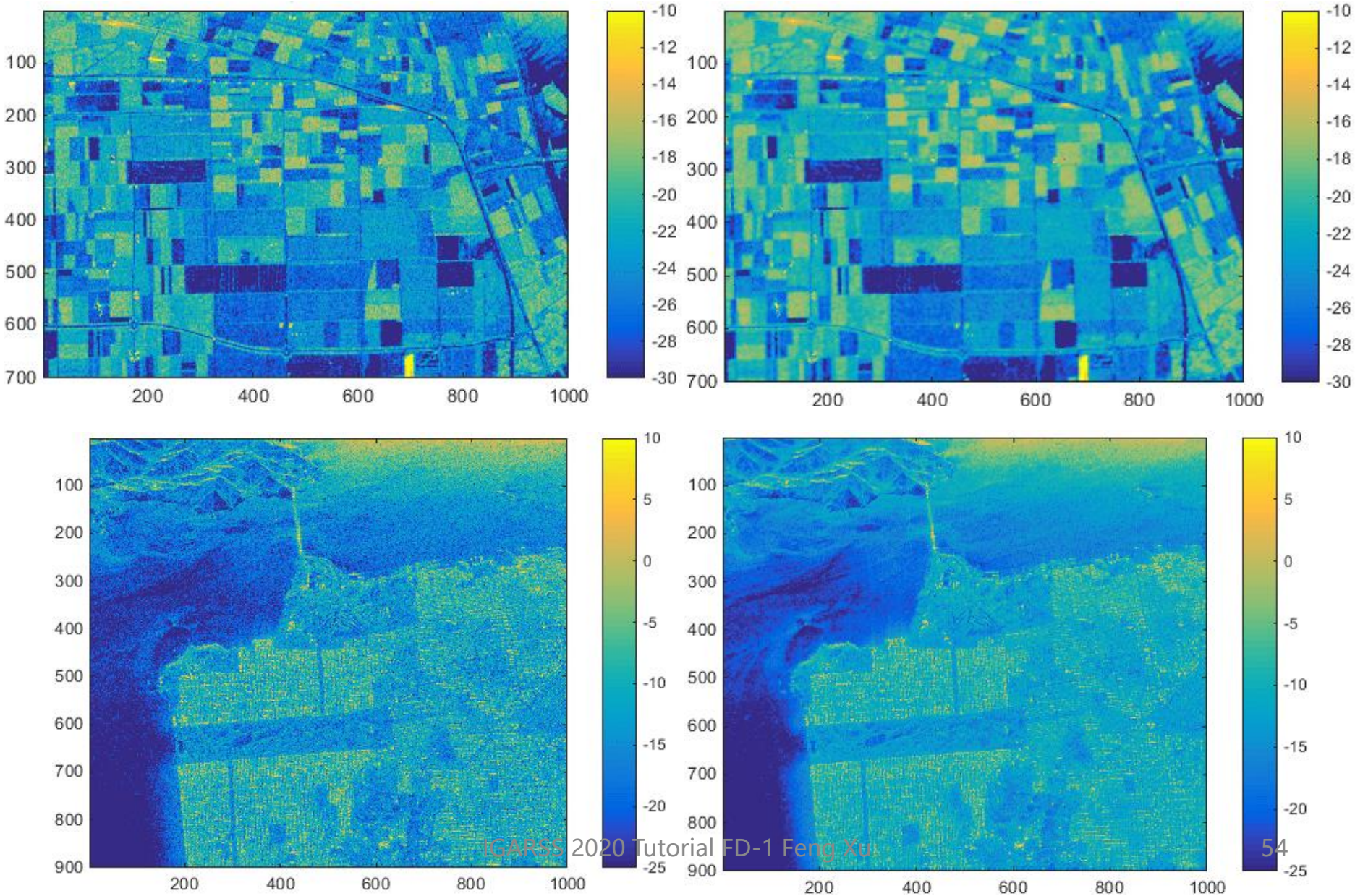




epoch 1 epoch 5 epoch 15 epoch 20

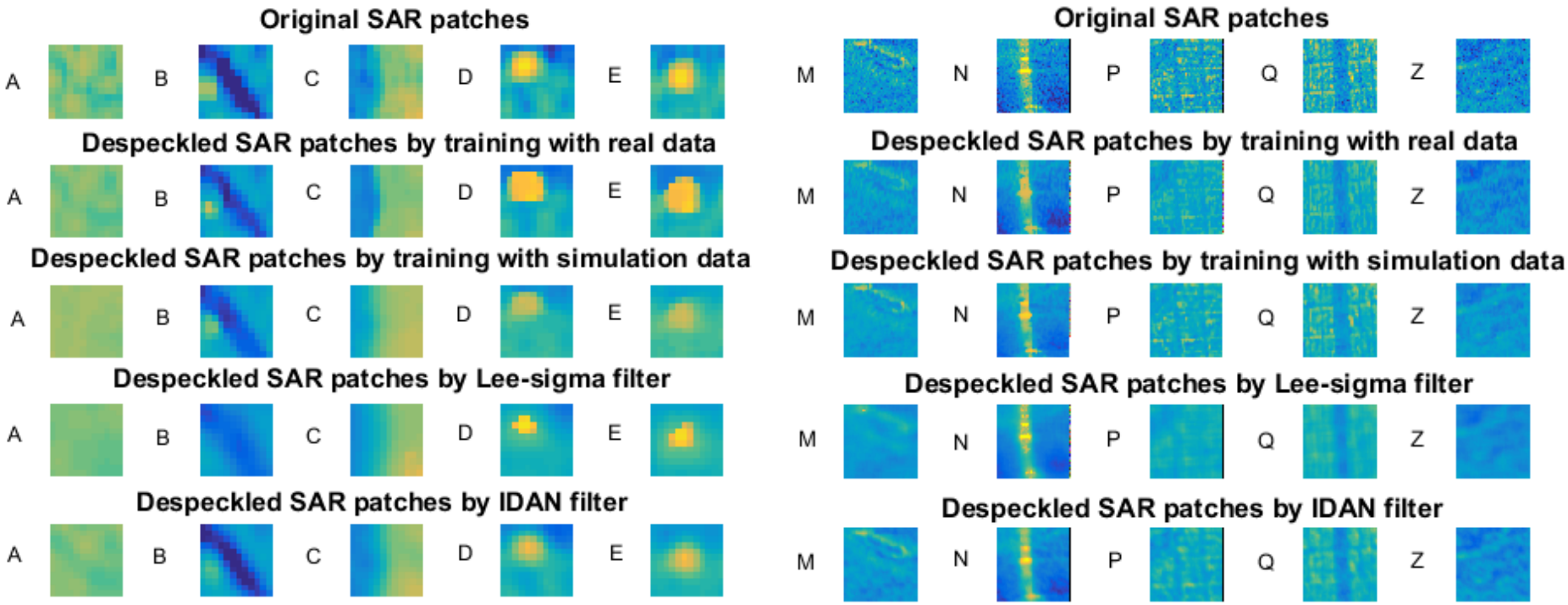
# Training with simulated data



# Speckle Reduction



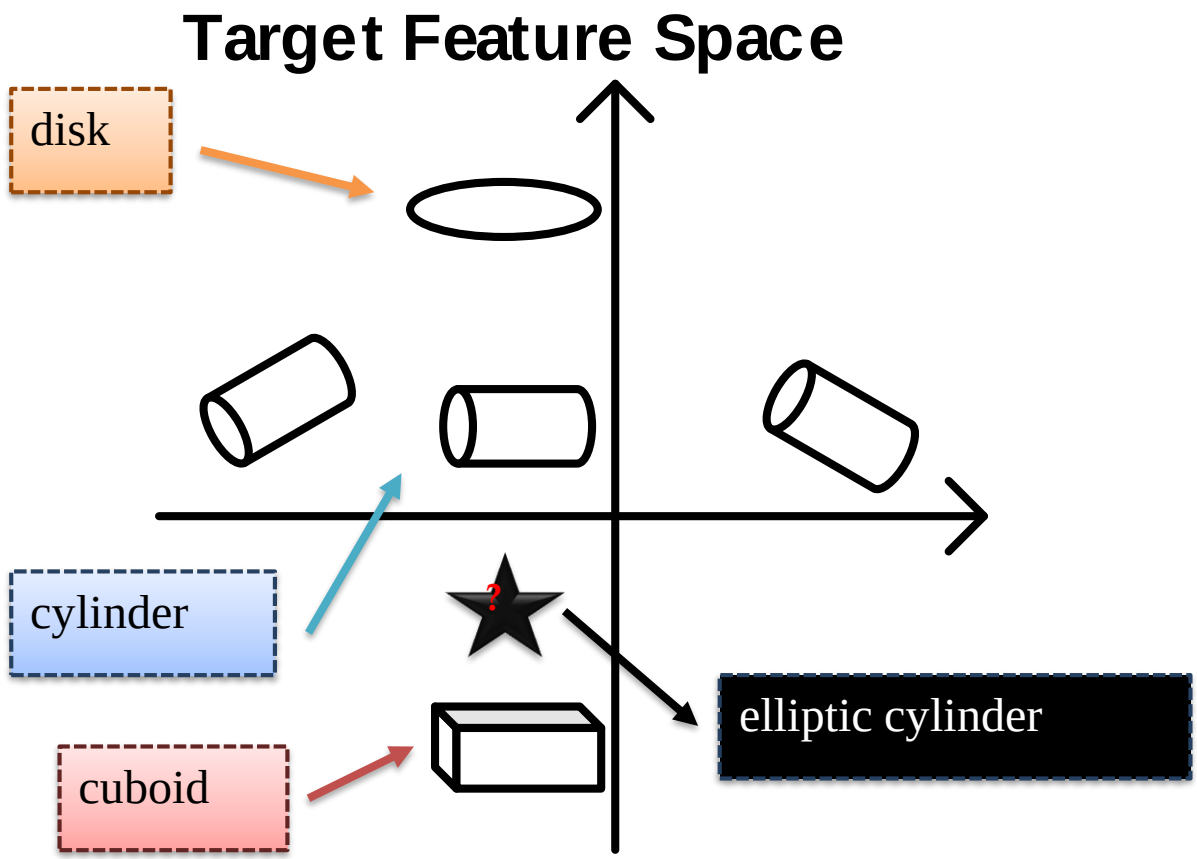
# Comparison




- Yue et al. 'Deep Despeckling Neural Network for SAR Images', IJRS, 2018
- Yue et al. 'A Generalized Gaussian Coherent Scatterer Model for Correlated SAR Texture', IEEE TGRS 2019 under review
- Code: https://github.com/fudanxu/SAR-GGCS

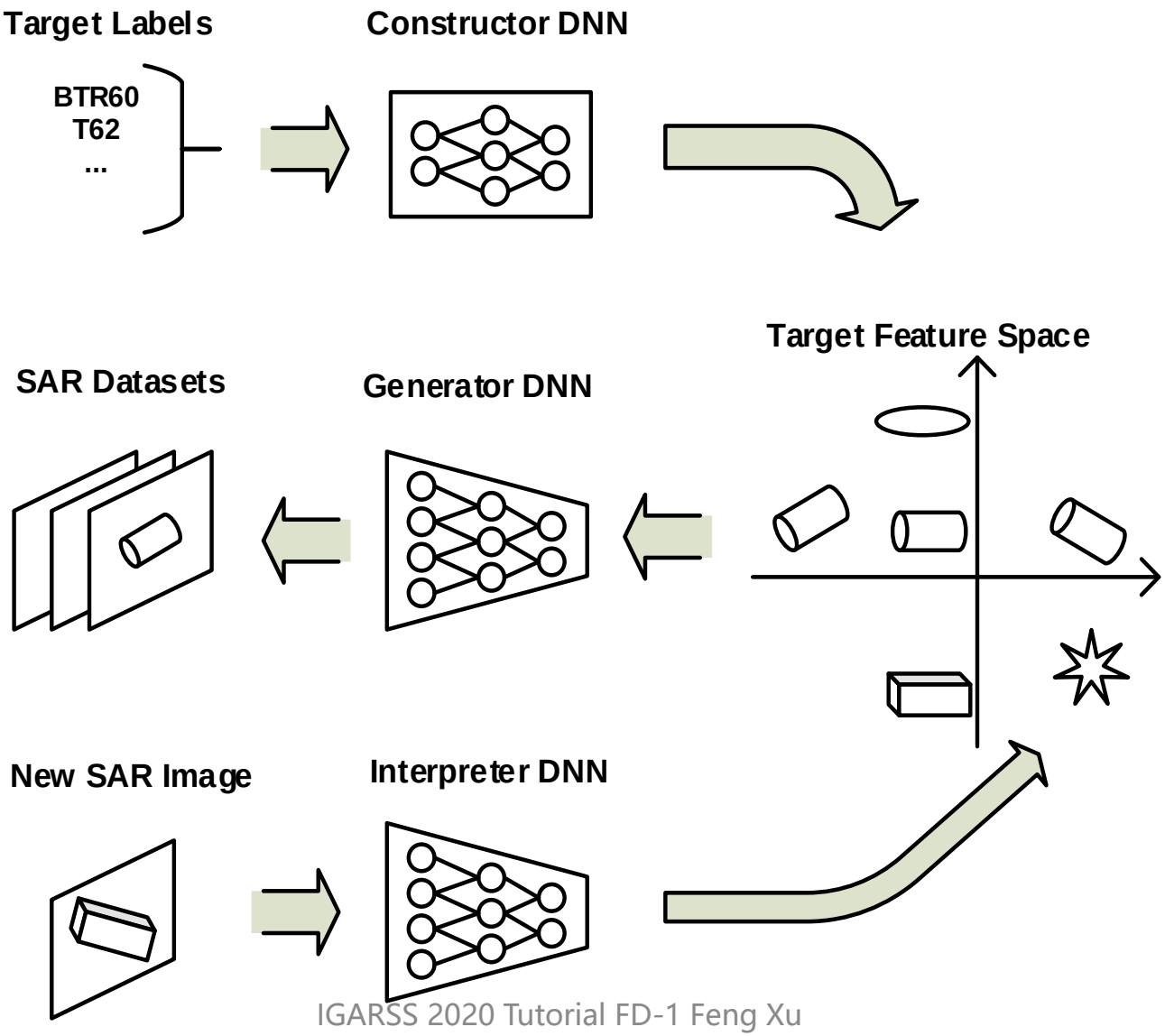
# Target Representation Learning

- **Human can understand/create new concepts based on the known concepts**
- **Construct target feature space by learning from known samples**

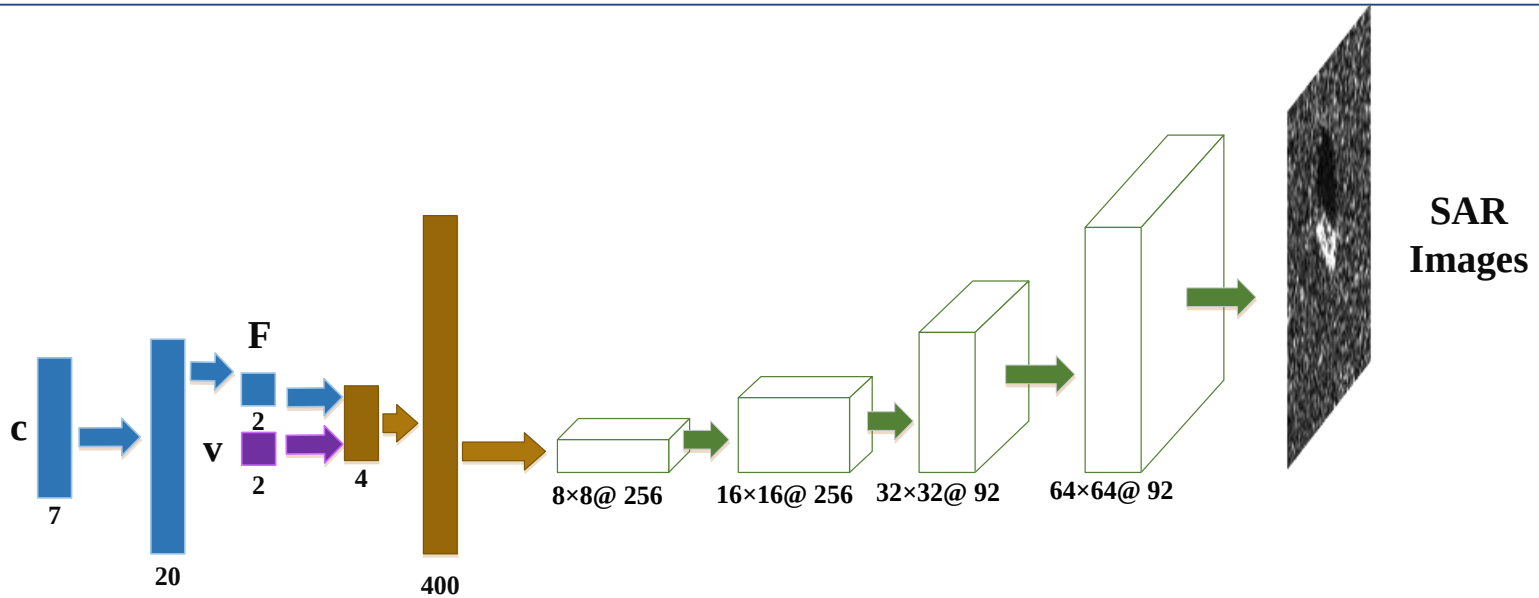


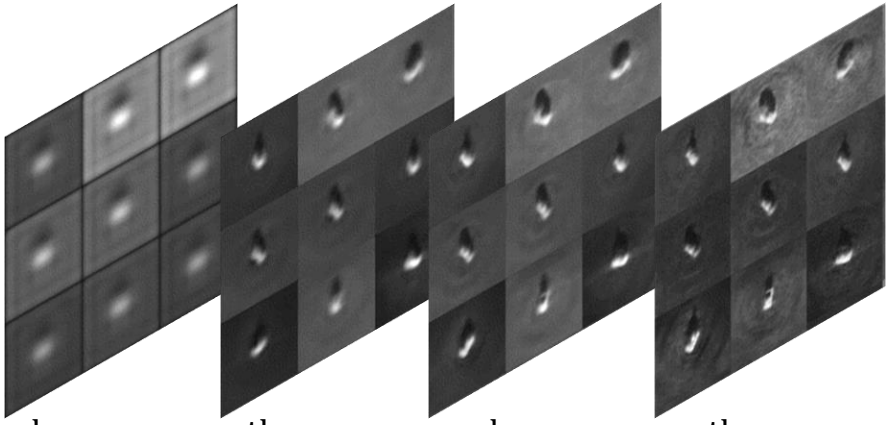

**Continuous target feature space**

# Generative Neural Network for SAR

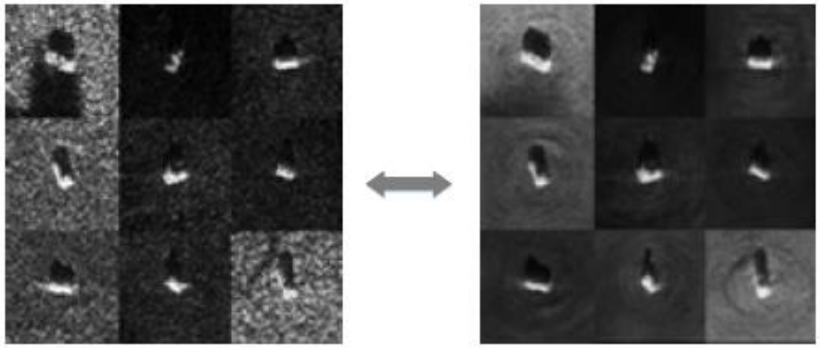


# Deep SAR Generative Neural Network



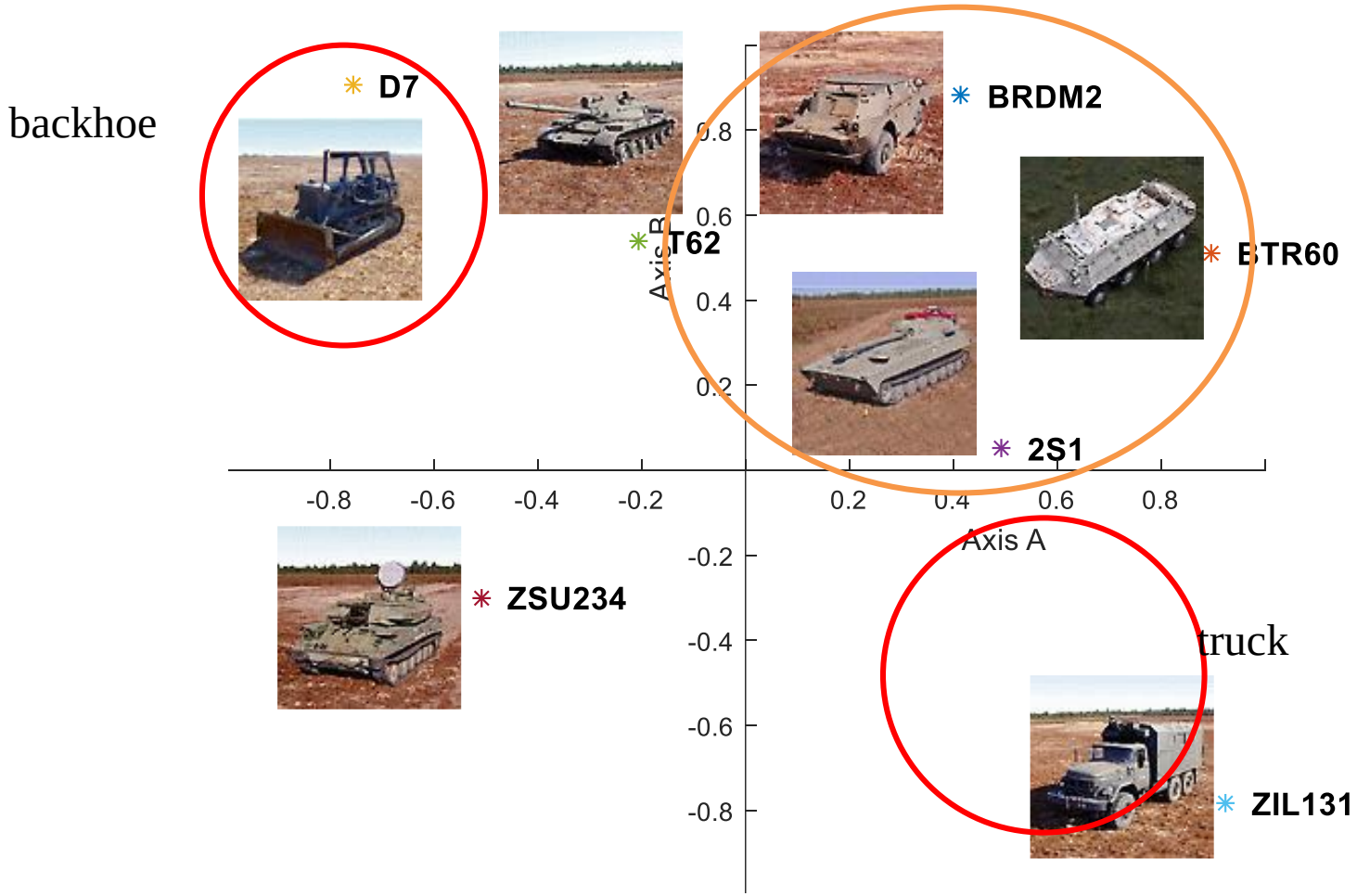



2nd epoch 37th epoch 72nd epoch 177th epoch



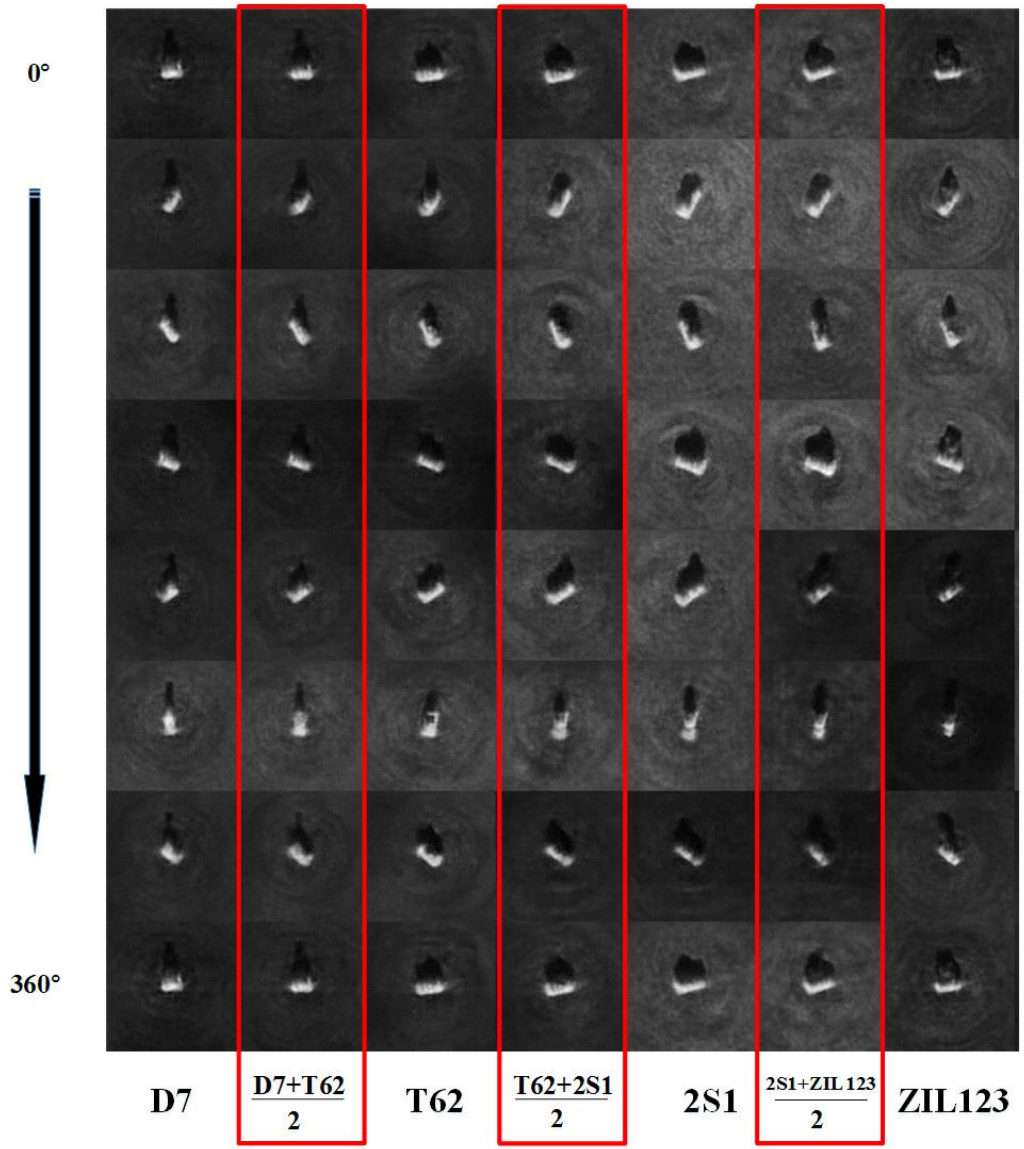
True SAR Image Generated SAR Image

# The Learned Target Space



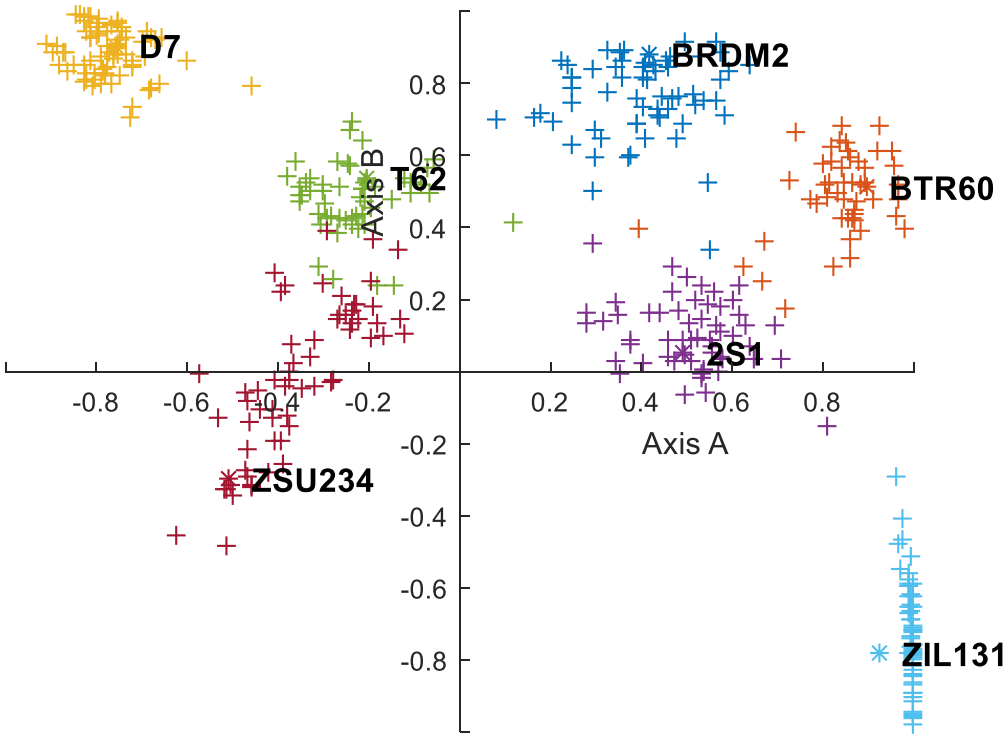
# Generating New Sample Images

- Interpolating between aspect angles
- Interpolating between different target types



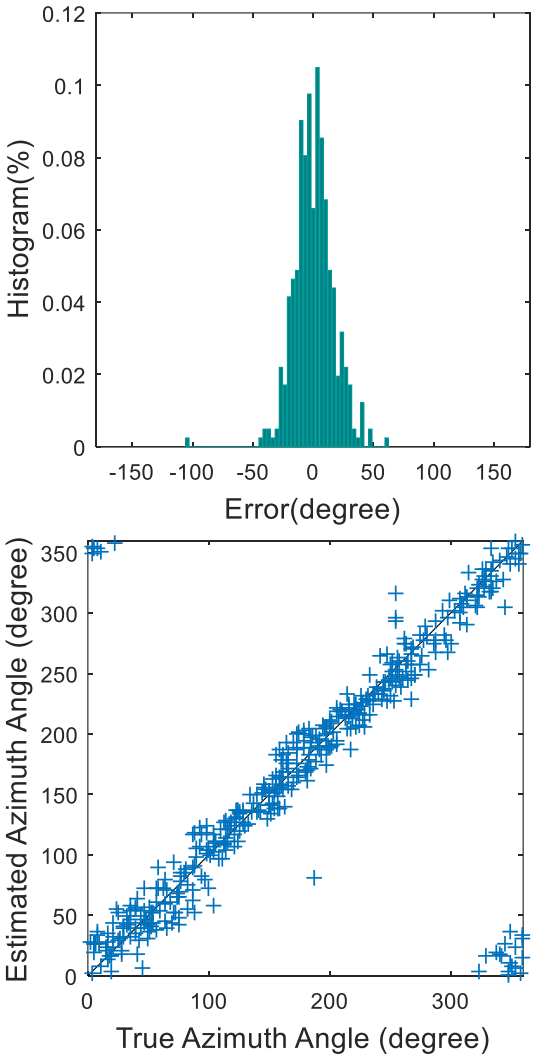
# SAR Interpreter CNN

## Feature Estimation




**Distribution of test samples in feature space.**

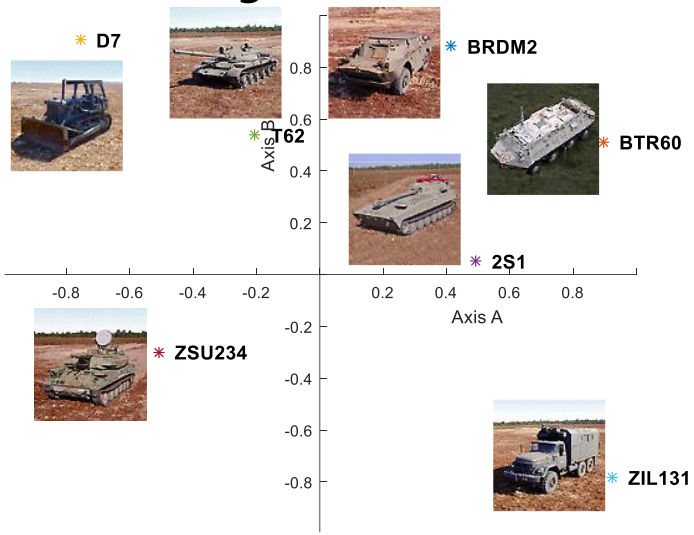
## Orientation Estimation

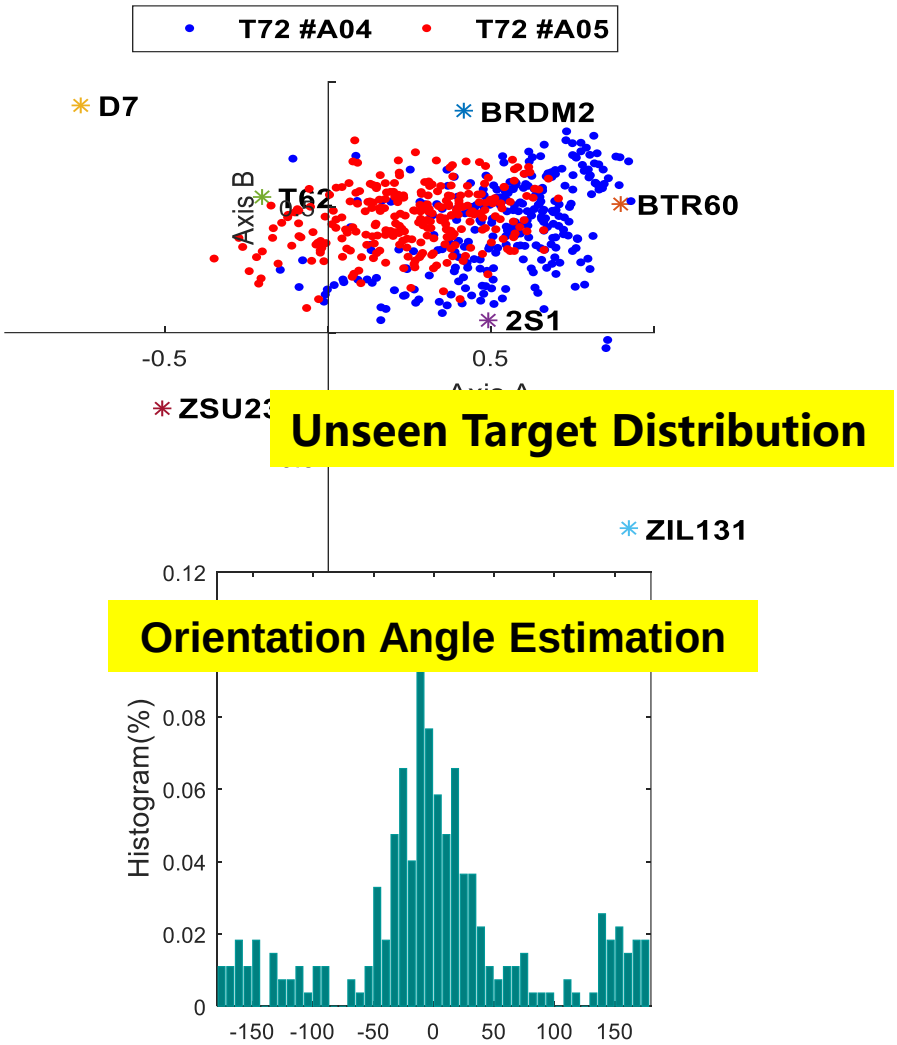



**Orientation angle estimation of test samples.**

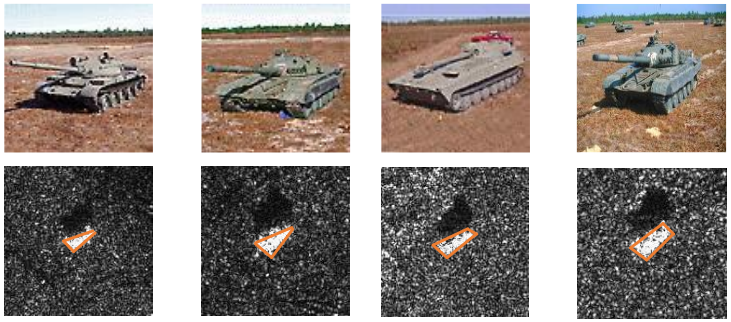
# SAR ATR without Training Sample

- **Known Targets**




- **Unseen Targets**



T62 T72 #A05 2S1 T72 #A04

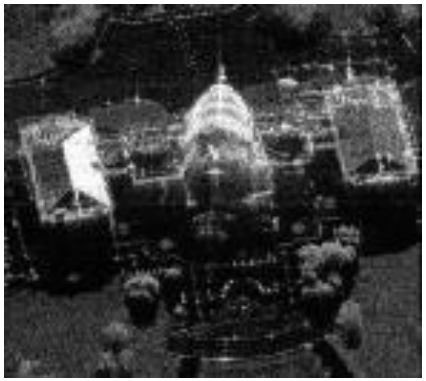
- Song et al. 'Zero-shot learning of SAR target feature space with deep generative neural networks, IEEE Geosci Remto Sens Lett, 2017.
- code: https://github.com/fudanxu/SAR-ZSL_for_featureSpace

# Few-Shot Learning

# Bottlenecks of DL in SAR

- DL as applied to SAR target recognition, can achieve superior performance on standard dataset, but has weak generalization capability, because:
  - SAR image is distinct from optical
  - SAR image training samples are order of magnitude less
  - SAR image is sensitive to observation configuration (angles)

◆ Distinct image features

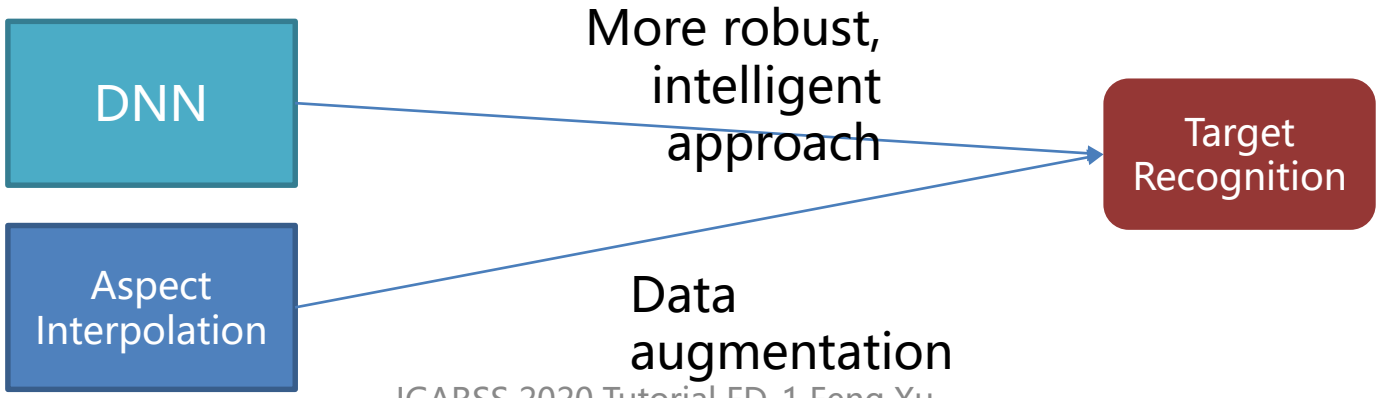


◆ Scattering mechanism

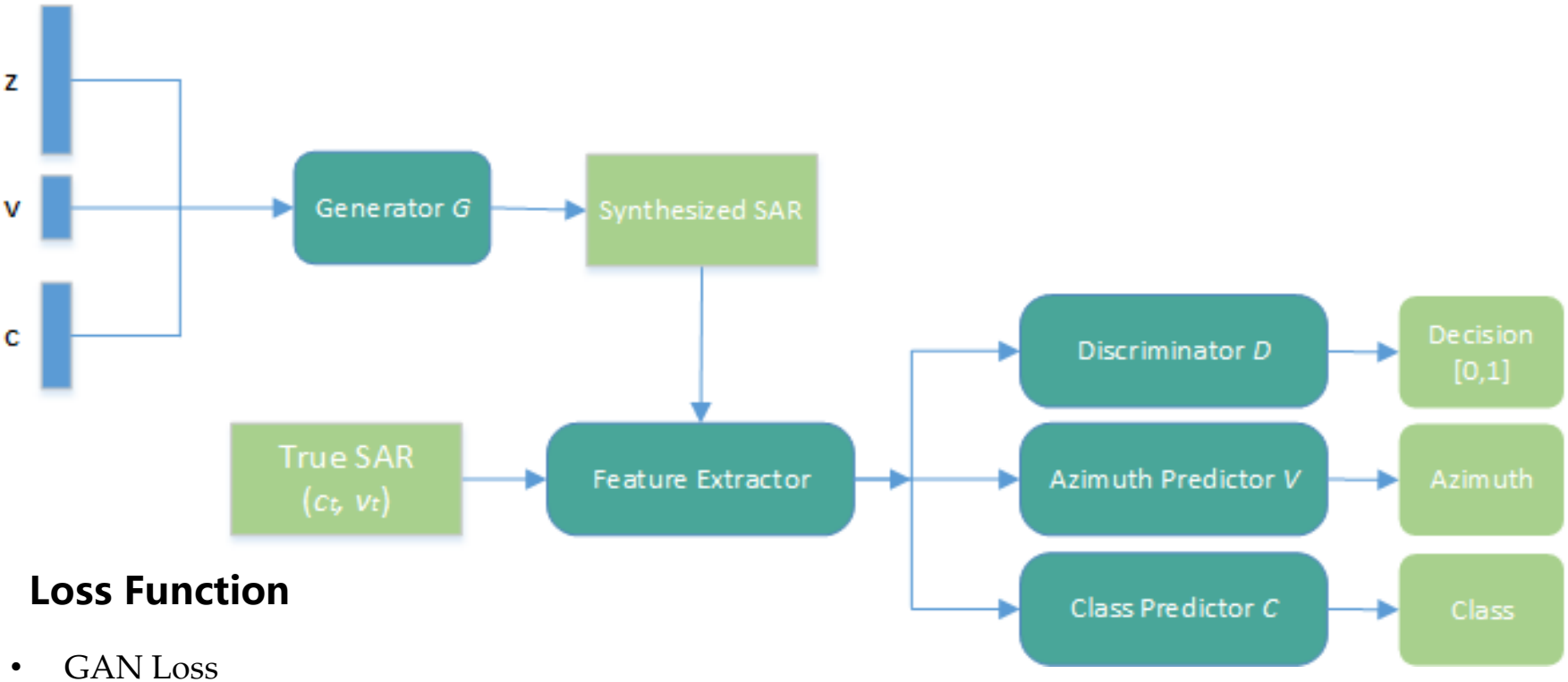
# Tackle small sample & over sensitivity?

- Type I challenge: Missing observation angle

| Aspect | 0° | 5° | 15° | 20° | 25° | 30° | 35° | … |
|---|---|---|---|---|---|---|---|---|
| BTR60 | × | √ | × | × | × | √ | × | |
| BRDM | √ | × | × | √ | × | × | × | |
| ZSU | × | × | √ | × | × | √ | × | |
| T72 | × | × | × | √ | × | × | √ | |



# Aspect Interpolation using GAN




## Loss Function

- GAN Loss

$$\mathcal{L}_{GAN} = \min_{G} \max_{D} \mathbb{E}_X[D(X)] - \mathbb{E}_{z\sim p(data)}[D(G(z,v,c))]$$

- Angle estimation

$$\mathcal{L}_v(v'|v) = \mathbb{E}_v\,[\|v - v'\|_2^2]$$

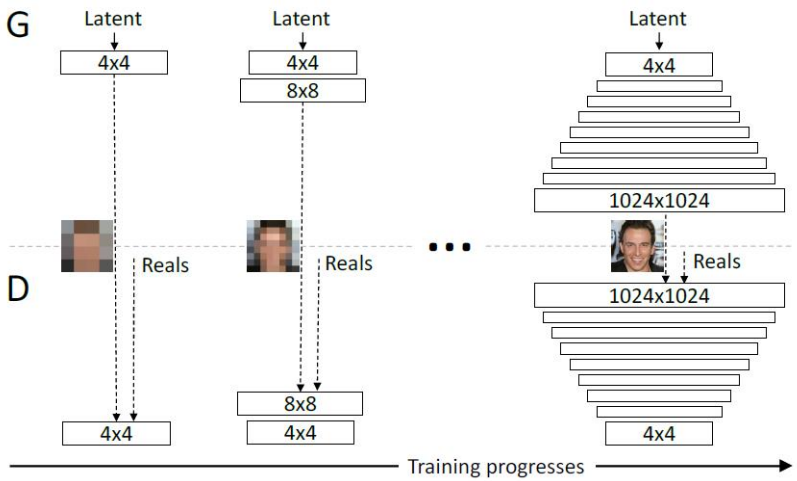
- Classification

$$\mathcal{L}_c(c'|c) = D_{entropy}(c'|\,c) = \mathbb{E}_c[c \log c']$$

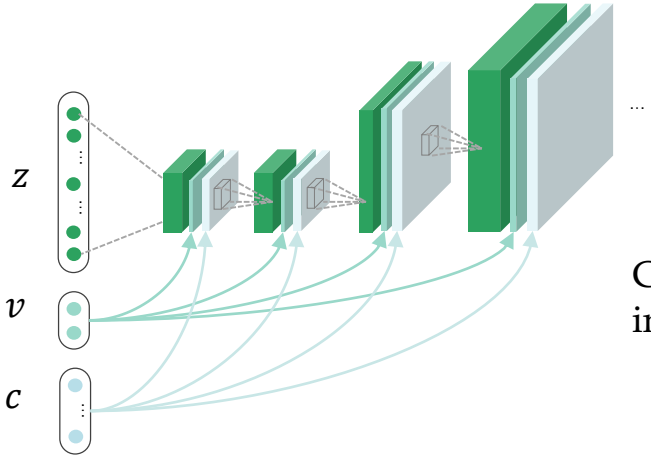
- Generation

$$\mathcal{L}_g(I|X) = \mathbb{E}_X\,[\|I - X\|_1]$$

# Network Architecture



PG-GAN: Progressively train multi-scale generation 4*4、8*8.. of image, finally output 128x128 image

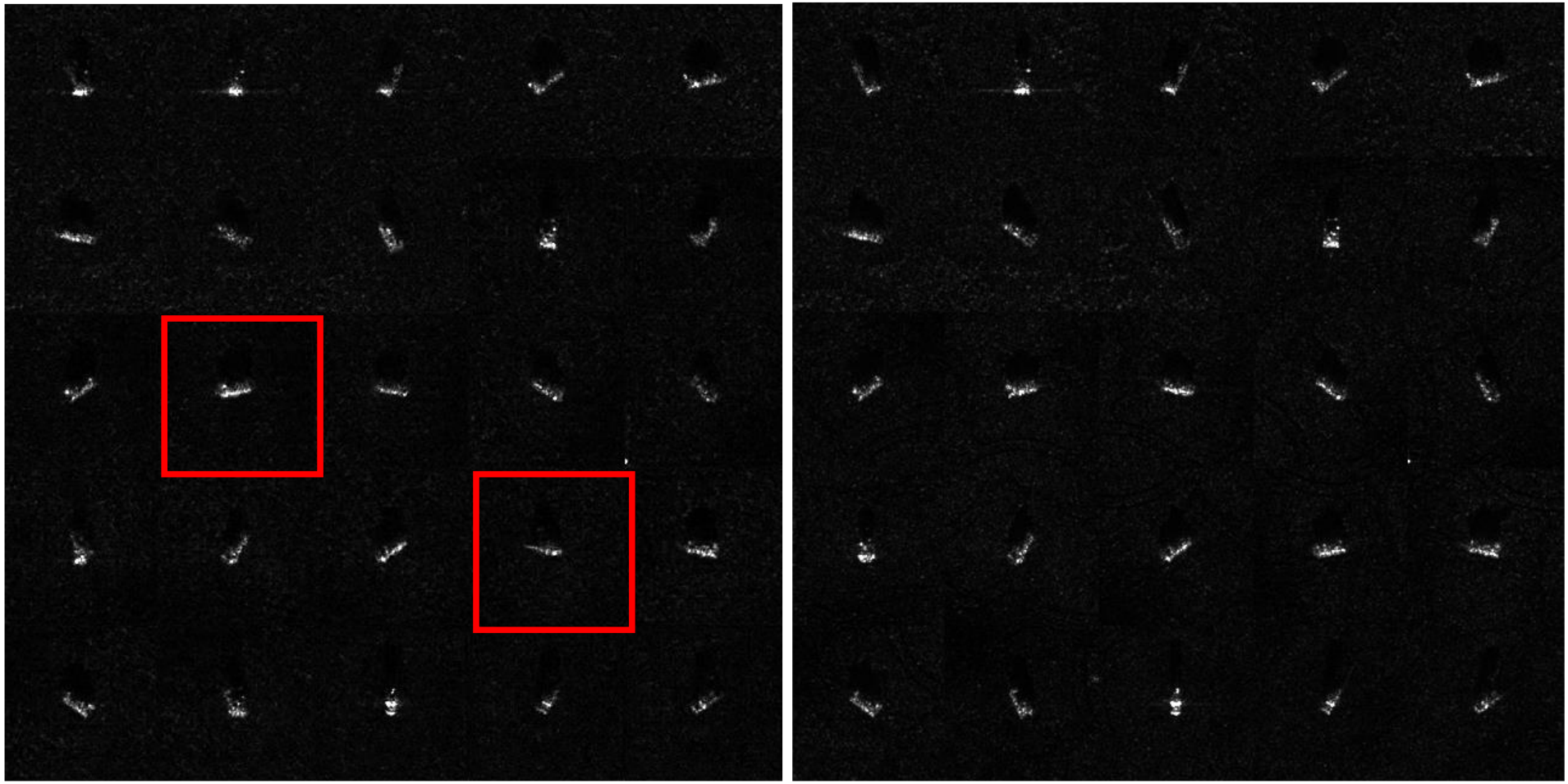



Cascaded skip connections: input label/angle information to each scale after convolution



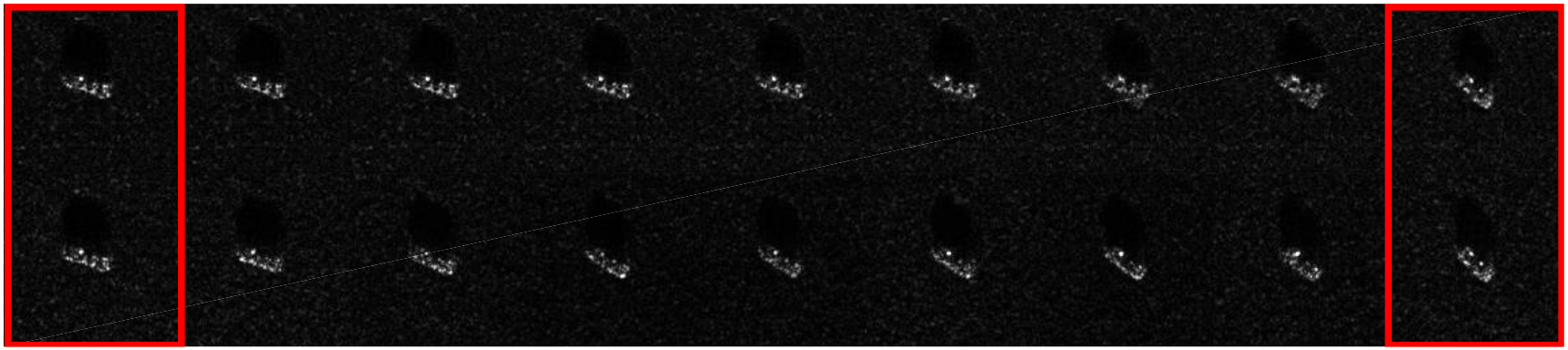
Tero Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation," ICLR 2018.

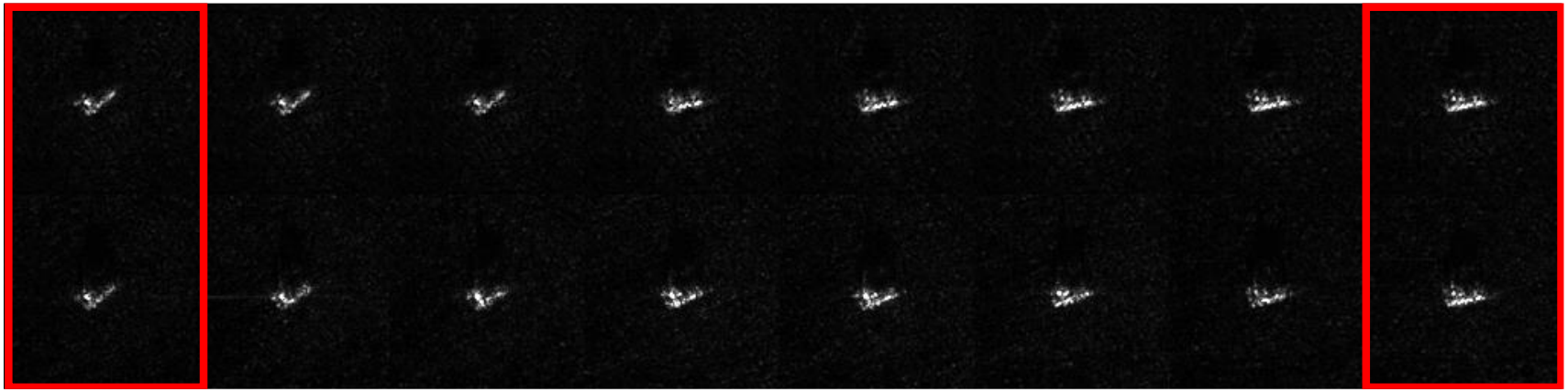
# Experiment Results



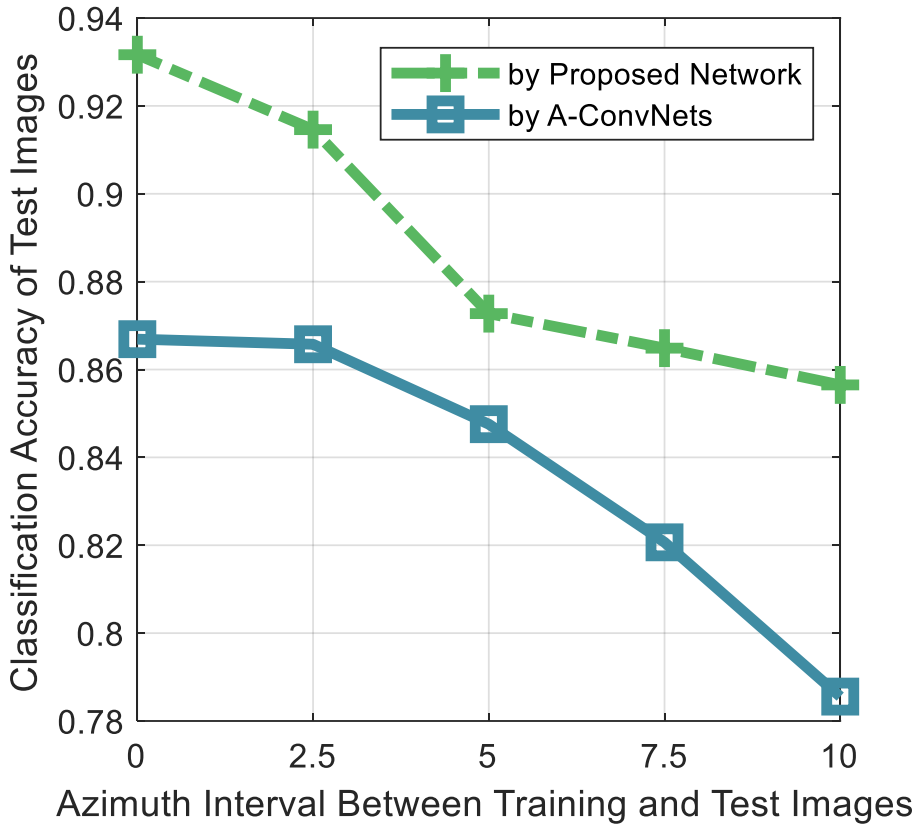
**Generated samples（left）vs. True samples（right）**
**Red boxes indicates the incorrectly-classified sample**

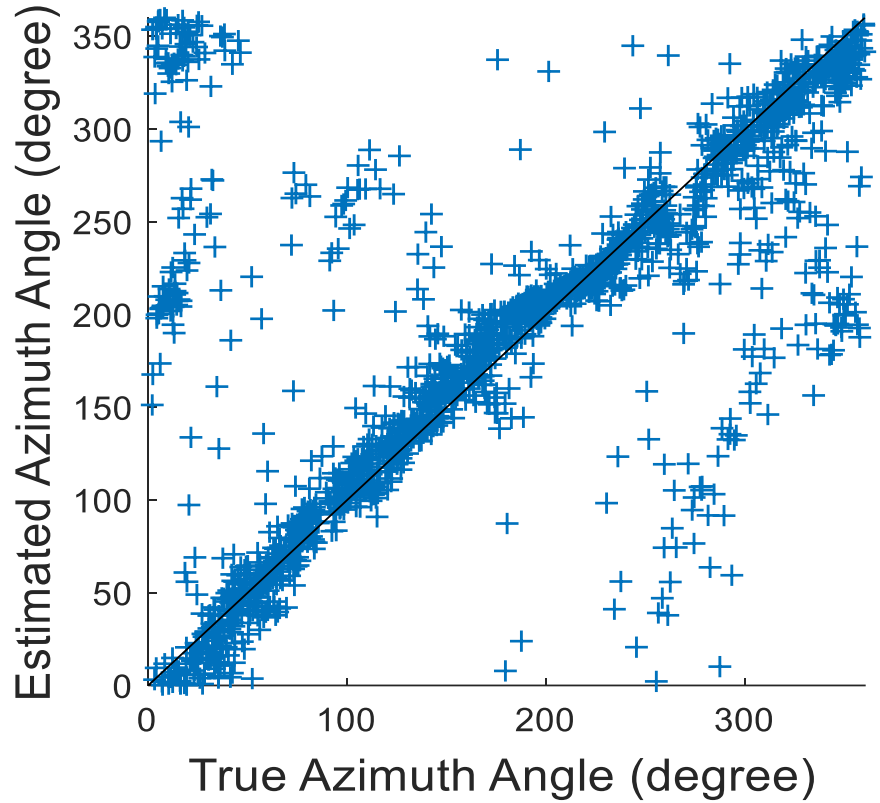
# GAN Aspect-angle Interpolation





# GAN-Augmented: accuracy improvement

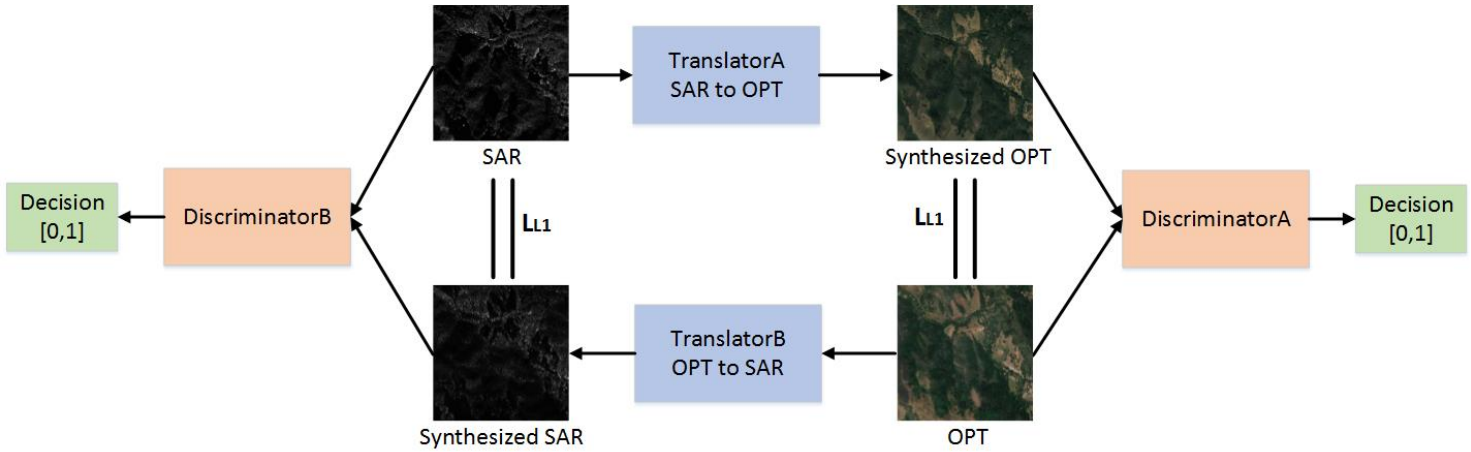


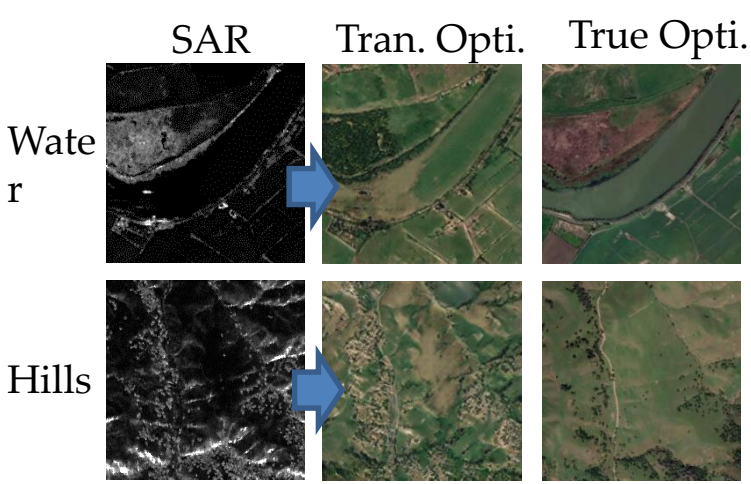



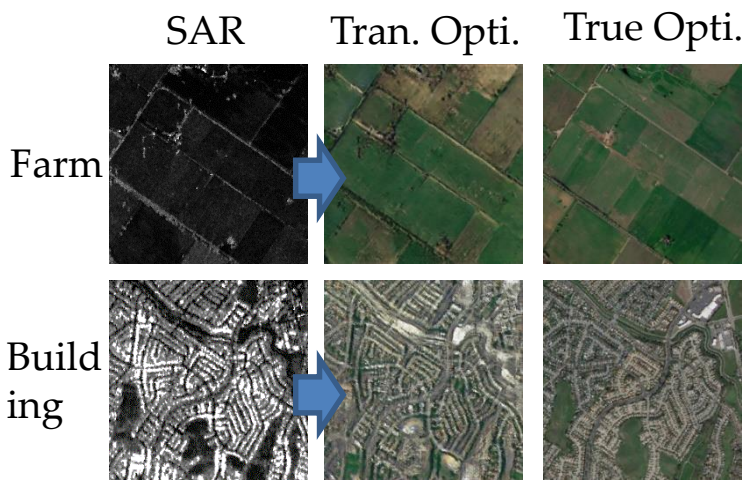

**Classification accuracy：88.53%；Angle estimation error：22.17°。**

- **Qian Song, F.Xu, YQ Jin, "SAR Image Representation Learning with Adversarial Autoencoder Networks", IGARSS 2019**

# Translate SAR to Optical for assisted interpretation



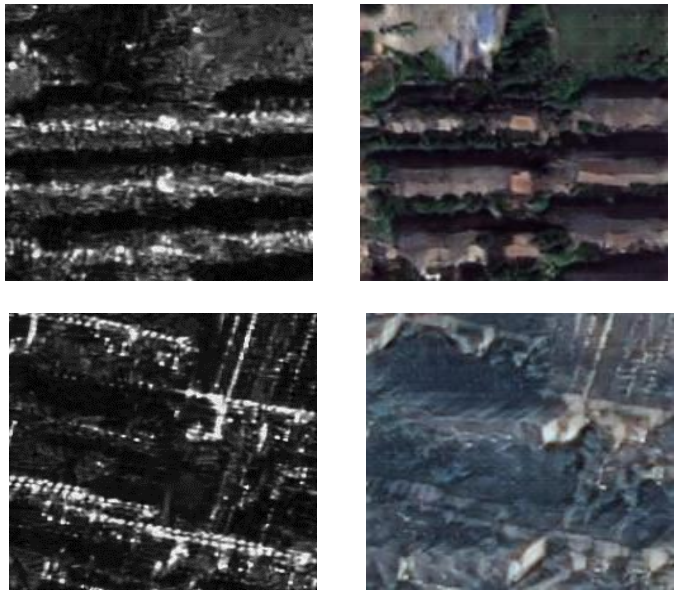



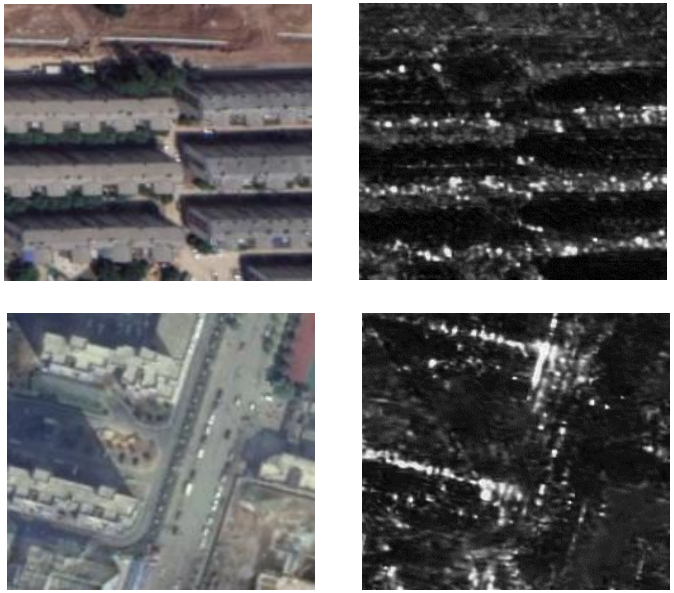

- Shilei Fu, F.Xu, YQ Jin, "Reciprocal Translation between SAR and Optical Remote Sensing Images with Cascaded-Residual Adversarial Networks", ISPRS Journal of Photogrametry and Remote Sensing, 2019 under review

# Translation of Hi-res SAR (GF3, 0.5m)

SAR ➡ Tran. Opti.



Opti. ➡ Tran. SAR

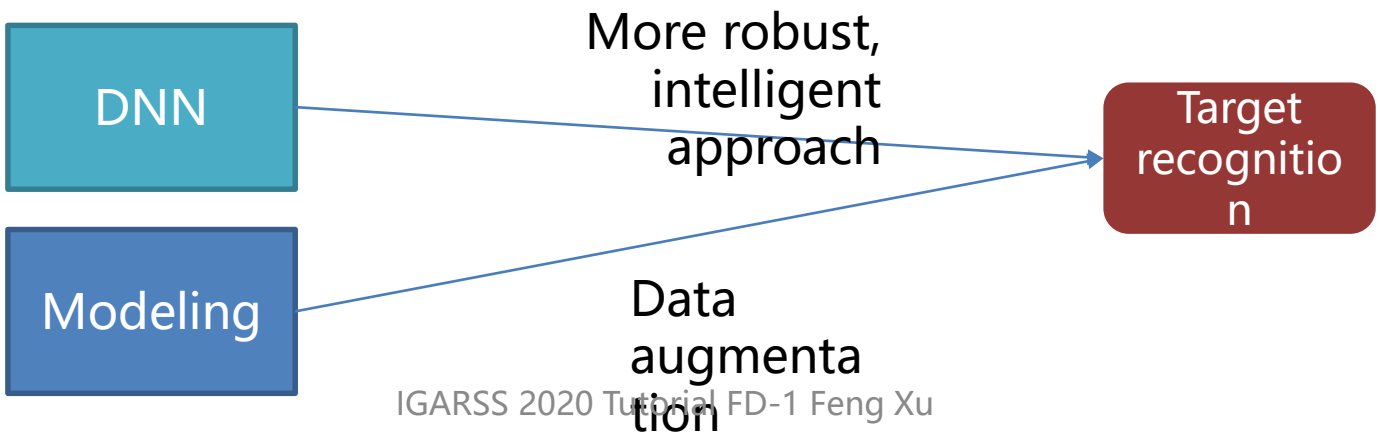


| metric | Tran. | CycleGAN | Pix2Pix | Proposed |
|---|---|---|---|---|
| PSNR | To optic. | 14.9675 | 14.4686 | **15.5268** |
| | To SAR | 15.7171 | 15.4978 | **15.8425** |
| SSIM | To optic. | 0.2656 | 0.2317 | **0.2782** |
| | To SAR | 0.2535 | 0.2194 | **0.2583** |
| FID | To optic. | 185.3181 | 212.5304 | **154.7532** |
| | To SAR | 62.1420 | 77.6901 | **53.0067** |

# Tackle small sample & over sensitivity?

- Type II Challenge： missing target

| aspect | 0° | 5° | 15° | 20° | 25° | 30° | 35° | … |
|---|---|---|---|---|---|---|---|---|
| BTR60 | × | √ | × | × | × | √ | × | |
| BRDM | √ | × | × | √ | × | × | × | |
| ZSU | × | × | √ | × | × | √ | × | |
| T72 | × | × | × | × | × | × | × | |



# EM Simulation-aided zero-shot learning

## Problem Setup

| Target | No. sample | Training | Testing |
|---|---|---|---|
| BRDM2 | 274 | 随机选取 80% | 随机选取 20% |
| BTR60 | 195 | 随机选取 80% | 随机选取 20% |
| D7 | 274 | 随机选取 80% | 随机选取 20% |
| 2S1 | 274 | 随机选取 80% | 随机选取 20% |
| T62 | 273 | 随机选取 80% | 随机选取 20% |
| ZIL131 | 274 | 随机选取 80% | 随机选取 20% |
| ZSU234 | 274 | 随机选取 80% | 随机选取 20% |
| BMP2 | 196 | 随机选取 80% | 随机选取 20% |
| BTR70 | 196 | 随机选取 80% | 随机选取 20% |
| 仿真T72 | 359 | 359 | 0 |
| 真实T72 | 196 | 0 | 196 |

**9 known targets**

**1 unknown target**

BRDM2

BTR60

D7

2S1

T62

ZIL131

ZSU234

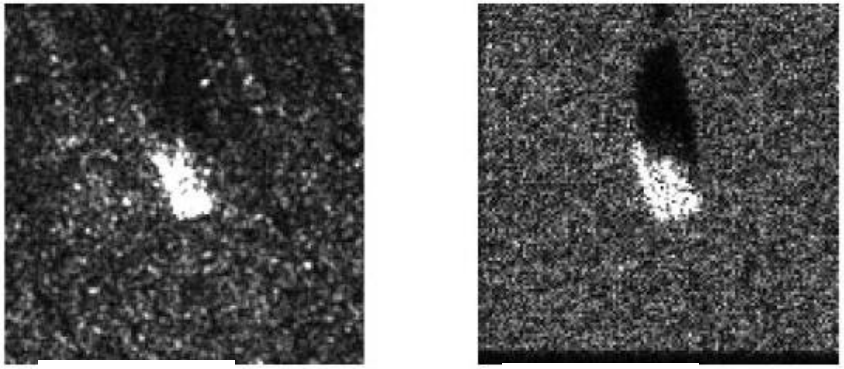
T72

X-band
0.3m resolution
Incident angle 15°

# EM Simulation-aided zero-shot learning

- Distinction of simulated image
- Lacks of randomness, variety

How to avoid overfitting?

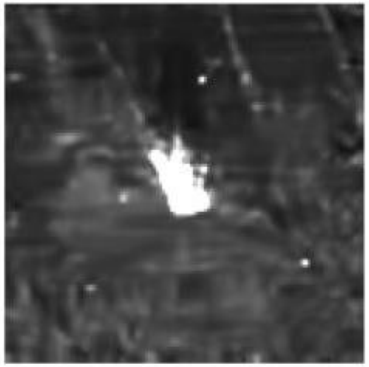


Before True (左)；



After (右)

L=1



After True (左)；

After (右)

使用相干斑抑制
算法BM3D

# EM Simulation-aided zero-shot learning

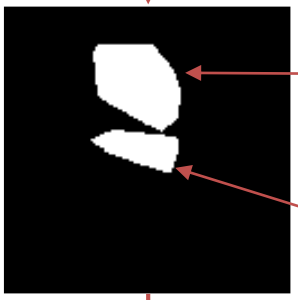
## Suppression of non-essentials

BM3D filtering



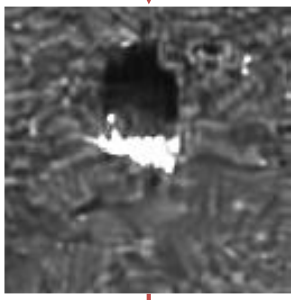
Segmentation

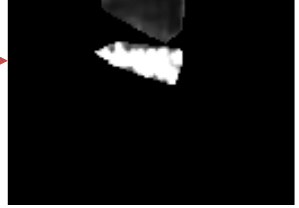
Adjusting



Shadow

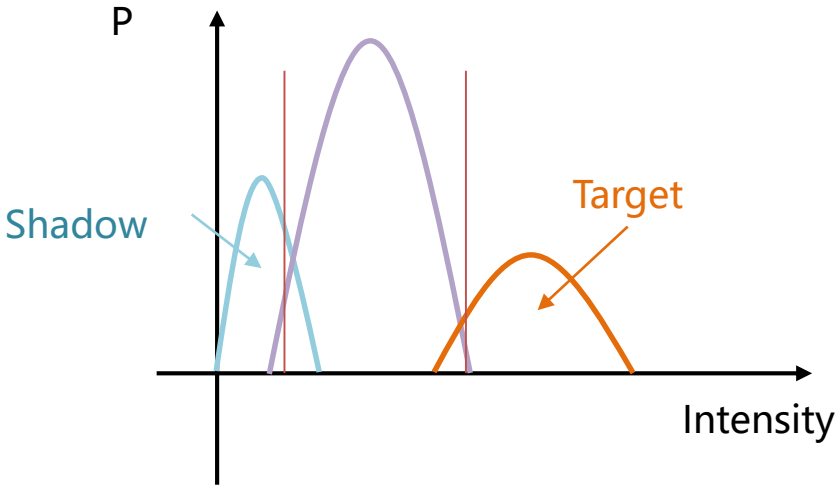
Target





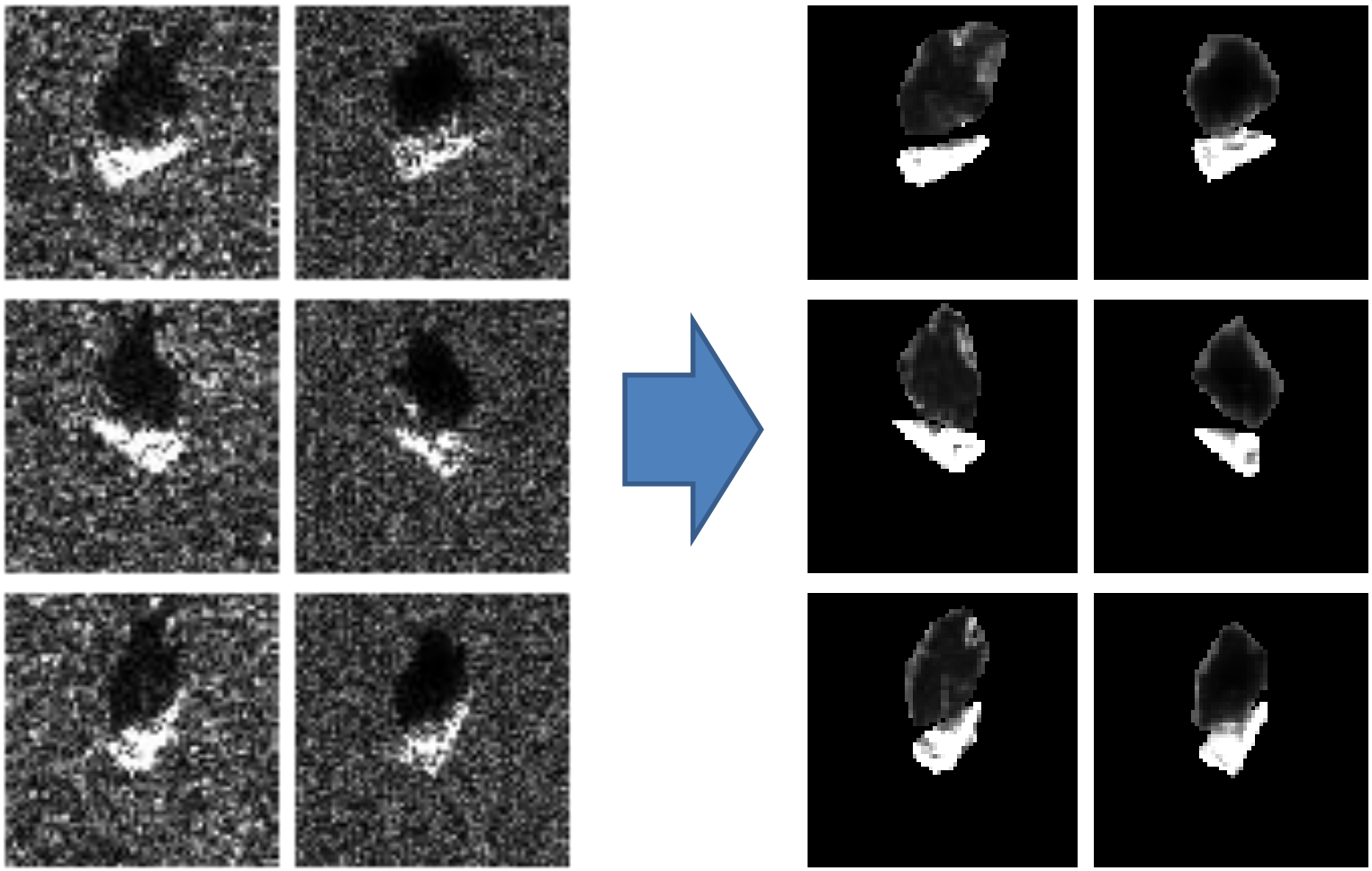
Preprocessed

### Segmentation




D. Cozzolino, S. Parrilli, G. Scarpa, G. Poggi and L. Verdoliva. "Fast Adaptive Nonlocal SAR Despeckling," IEEE Geoscience and Remote Sensing Letters, 2013.

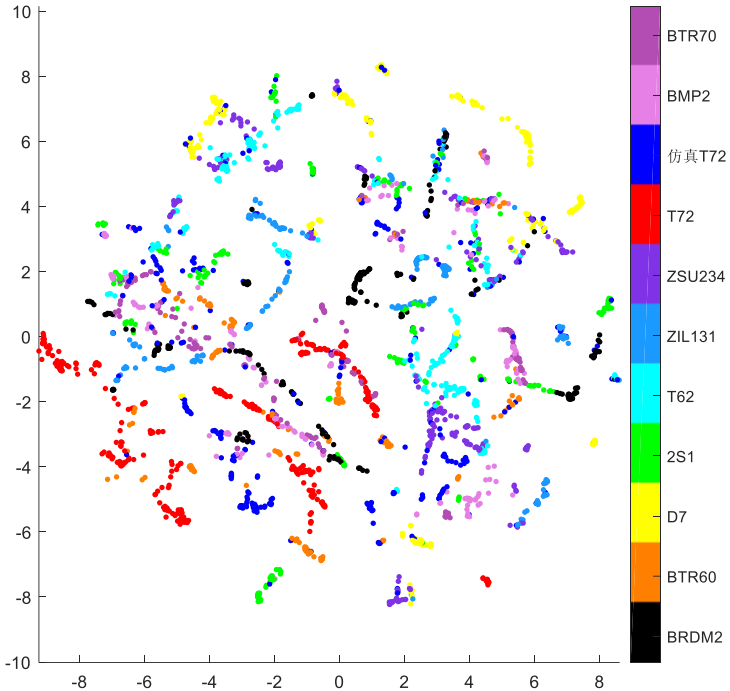
# Before and after preprocessing



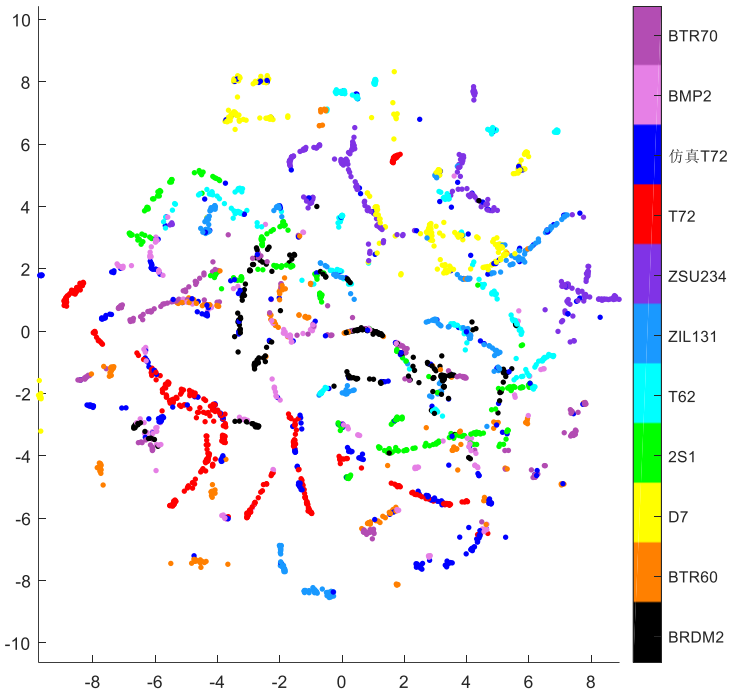
True Simulated True Simulated

# EM Simulation-aided zero-shot learning

## Visualization of feature space with t-SNE



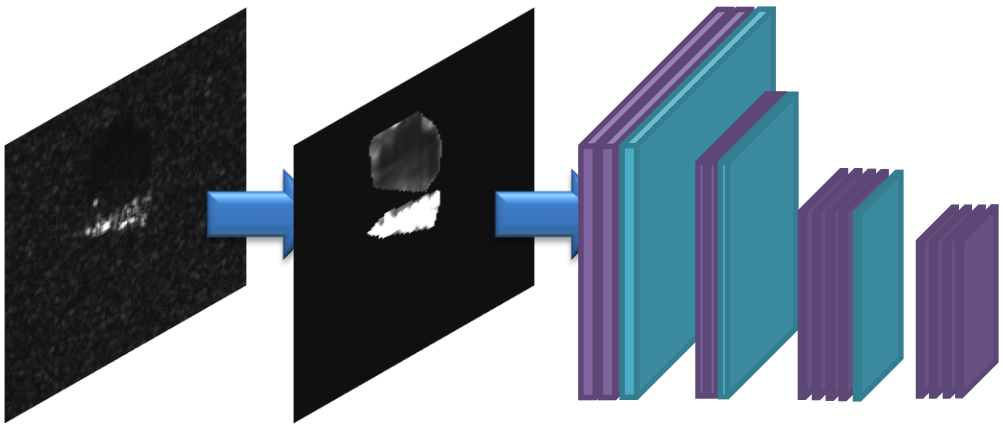

Before



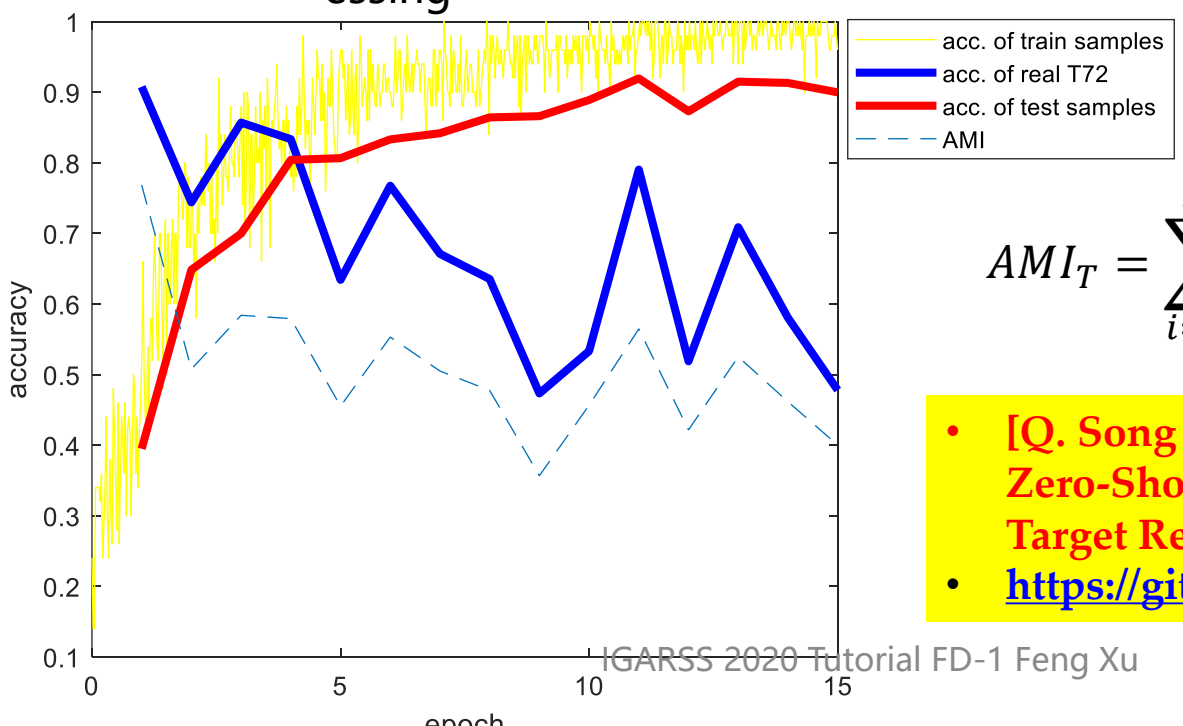

After

L. J. P. van der Maaten and G. E. Hinton. “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 25792605, Nov. 2008.

# EM Simulation-aided zero-shot learning



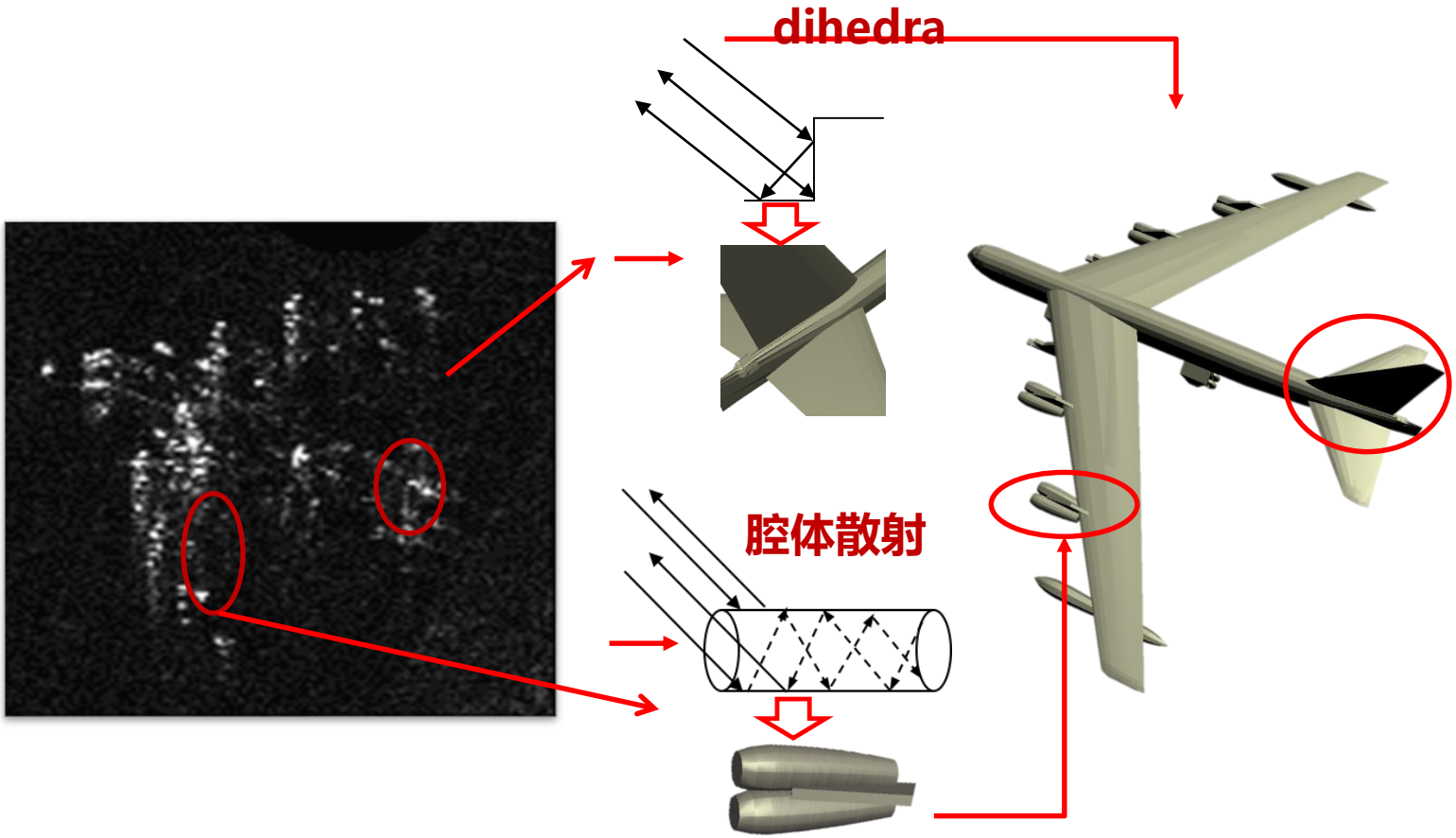

| 9 known target | T72 |
|---|---|
| 89.6% | 81.7% |




$$AMI_T = \sum_{i=1}^{N} MI_{T_i - T}$$

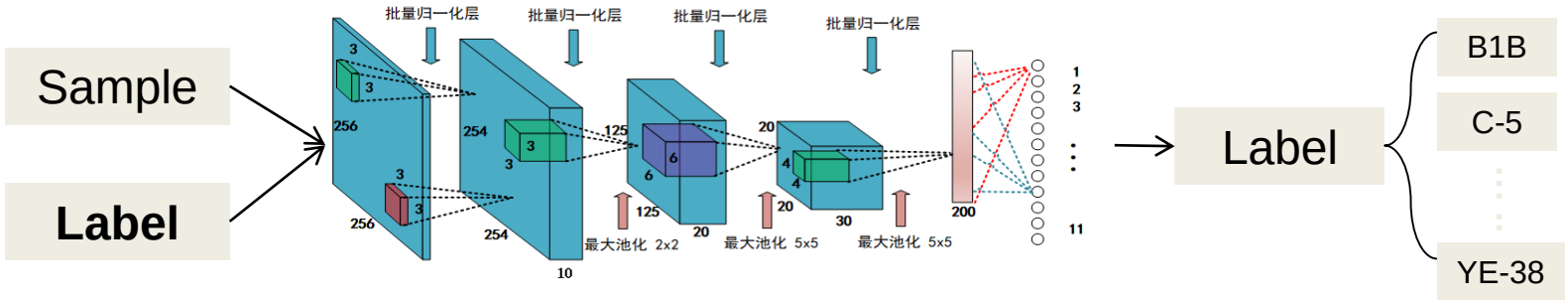
- [Q. Song et al., "EM Simulation-Aided Zero-Shot Learning for SAR Automatic Target Recognition", IEEE GRSL 2020]
- https://github.com/fudanxu/SAR-ZSL

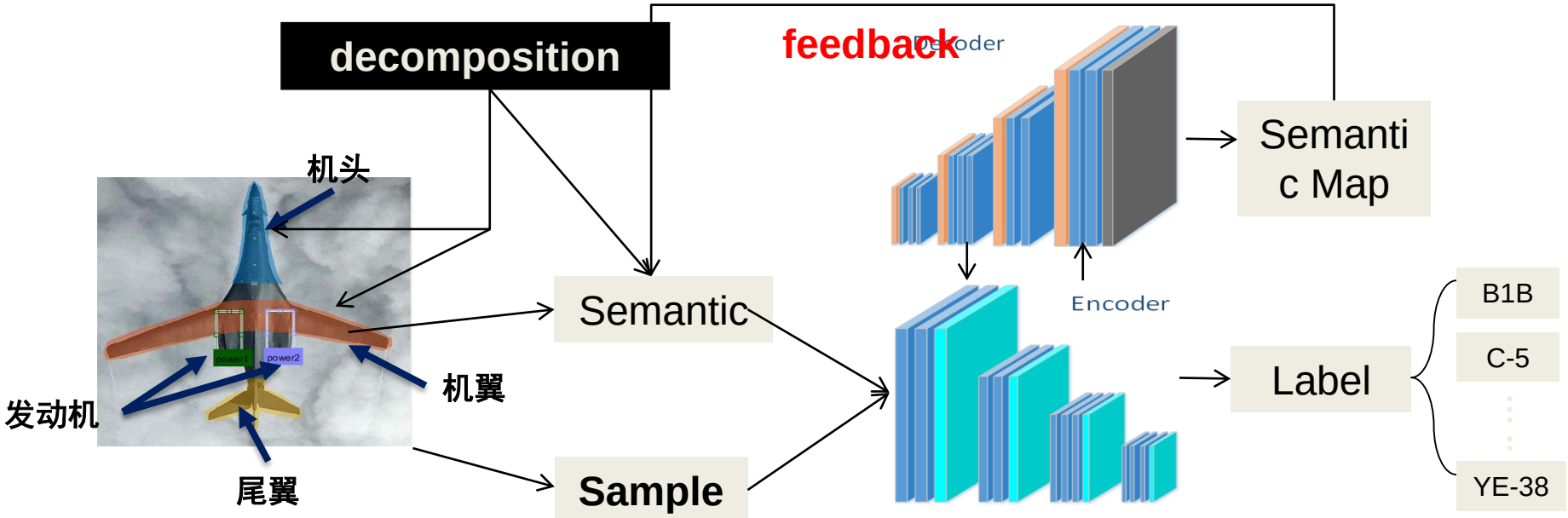
# Dive into scattering mechanism



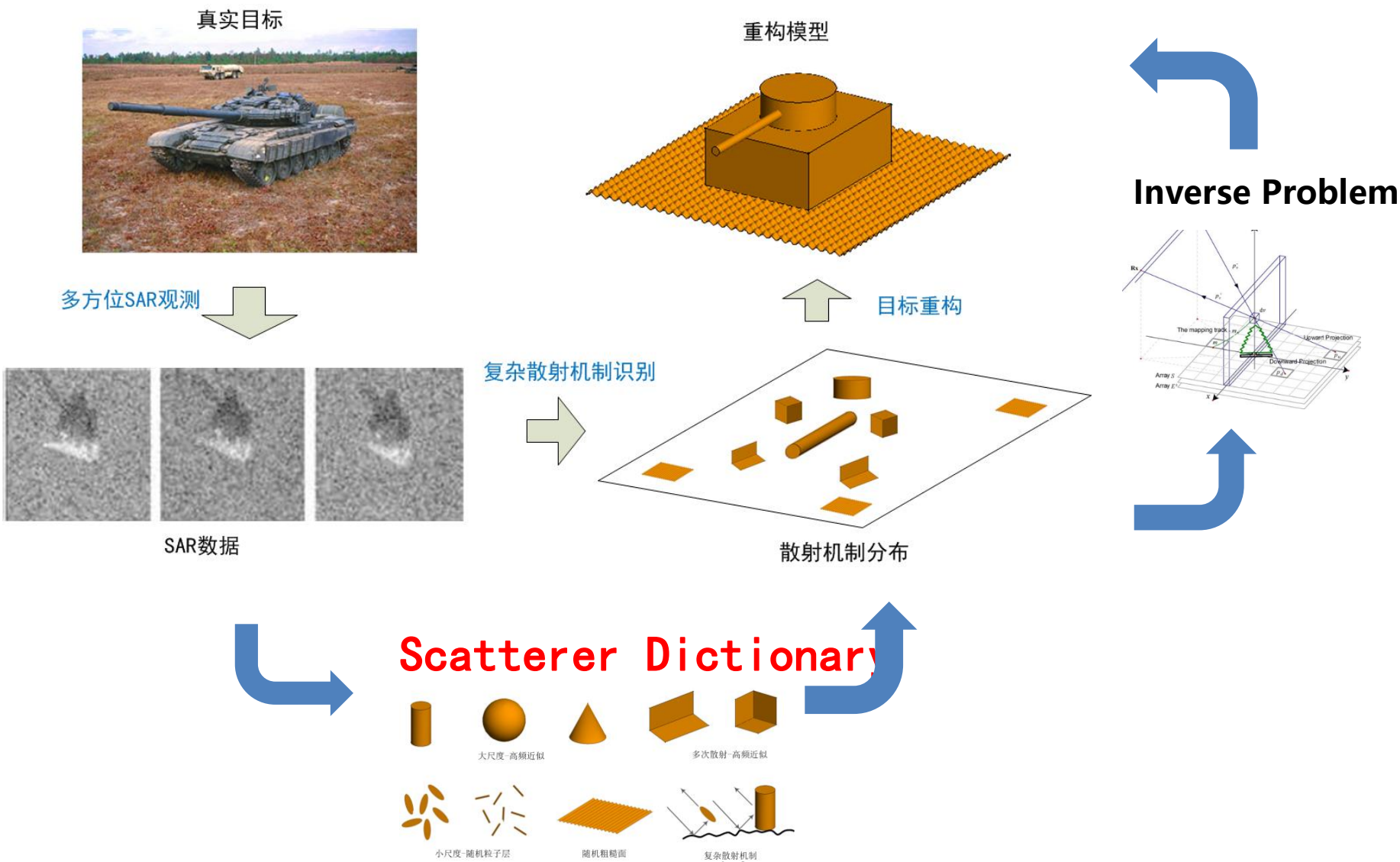

SAR Intelligent Interpretation should base on EM physics

# Structural annotation

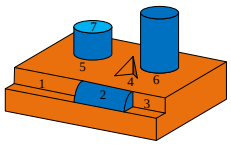


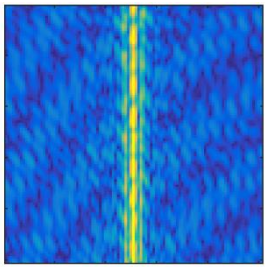
# Target Reconstruction



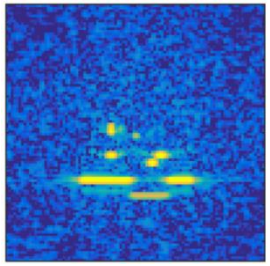
# Scatterer extraction with MCMC



散射

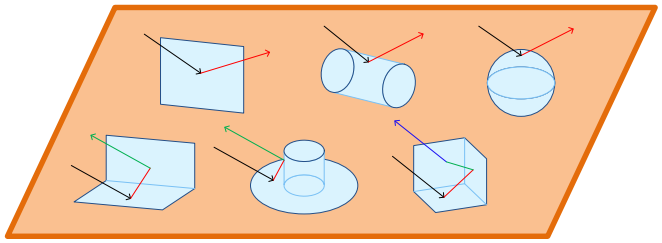


成像



传统SAR成像

几何散射特征集



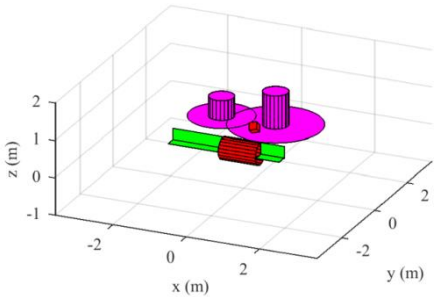
参数模型 $S(\boldsymbol{\theta}), \boldsymbol{\theta} = \{A, \alpha, L, \phi_o, x, y\}$

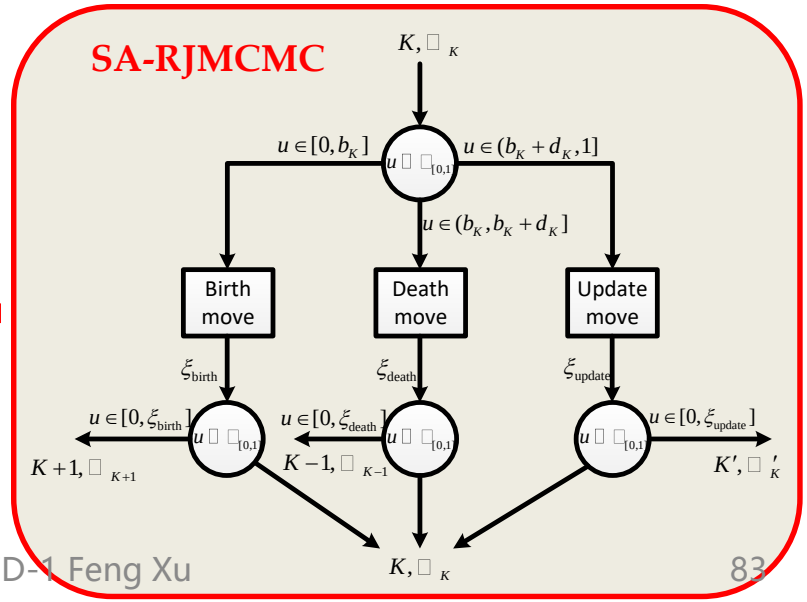
$$\mathbf{y} = \sum_{i=1}^{K} S(\boldsymbol{\theta}_i) + \mathbf{n}$$

Bayesian估计: SA-RJMCMC

Bayesian模型 $\pi(K, \mathrm{M}_K \mid \mathbf{y}), \mathrm{M}_K = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K\}$

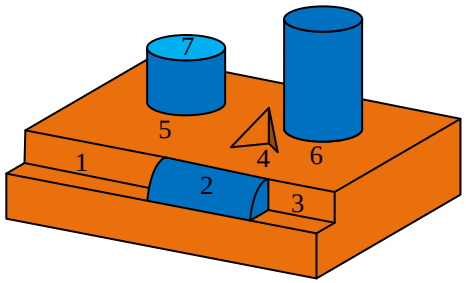
散射特征的几何重构



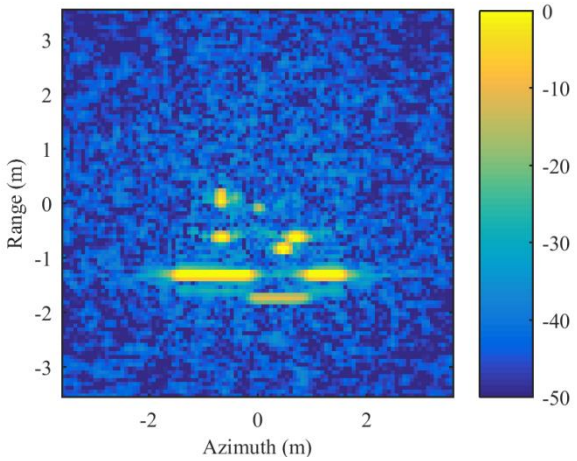



# Simple case: SLICY Model

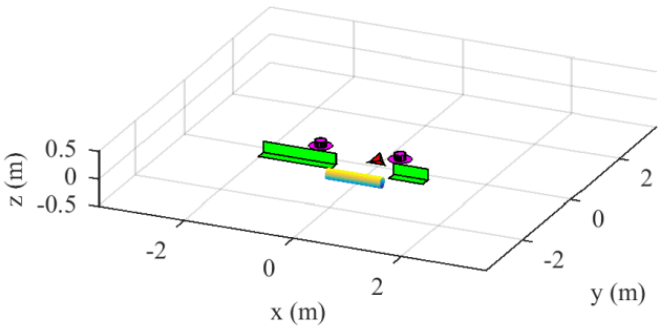
1. 二面角
2. 圆柱
3. 二面角
4. 三面角
5. 帽顶
6. 帽顶
7. 腔体



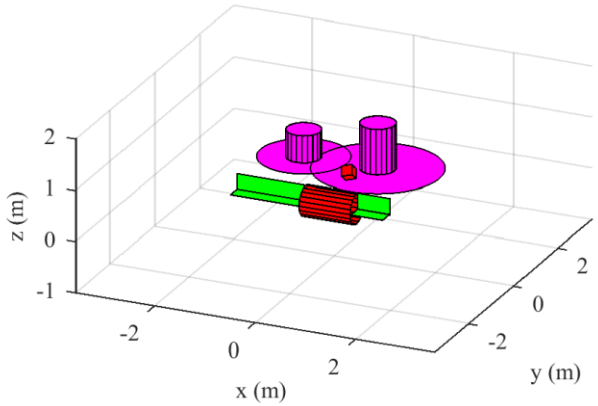

SLICY模型




散射成像

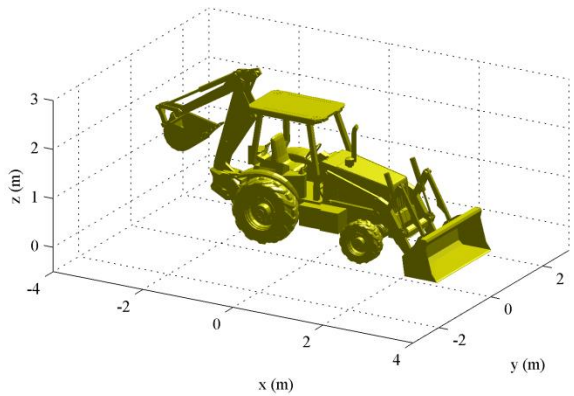



散射体提取

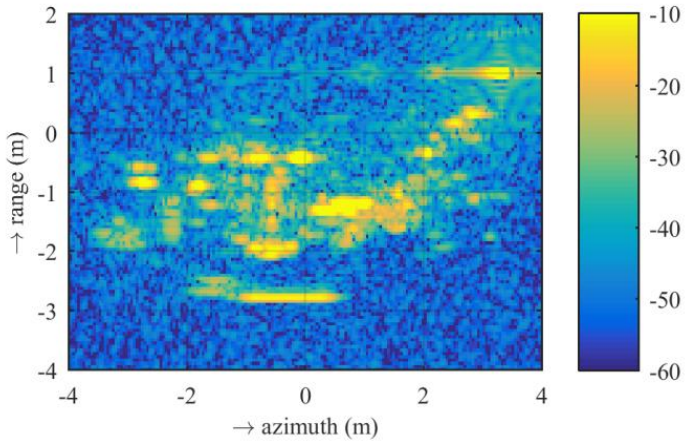



几何散射体重构

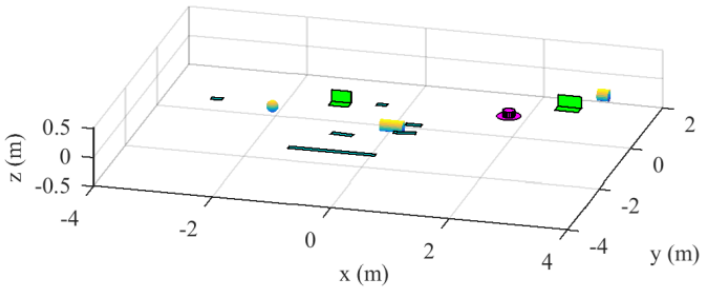
# Case study: Backhoe



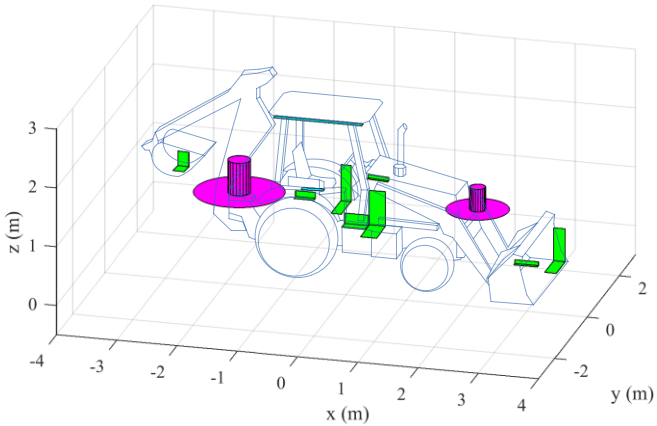

铲车模型




散射成像

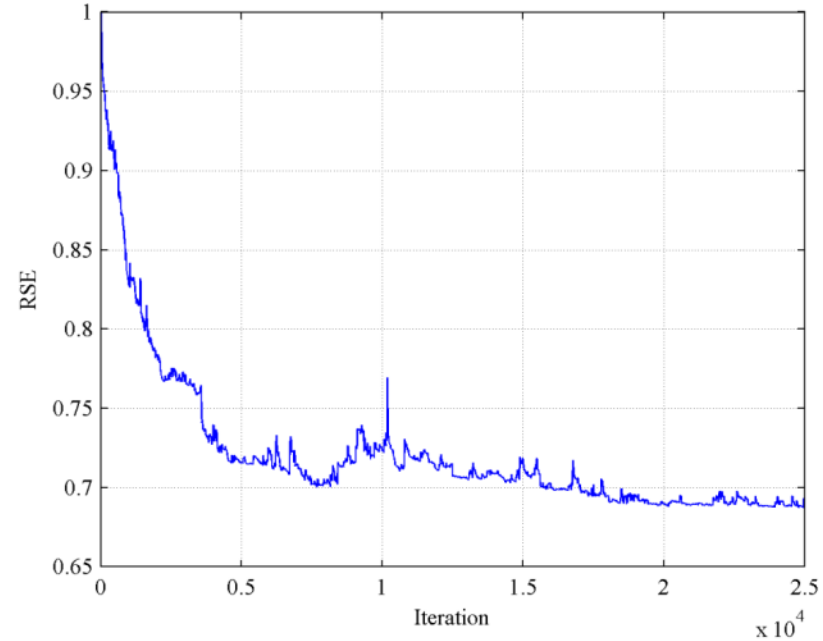


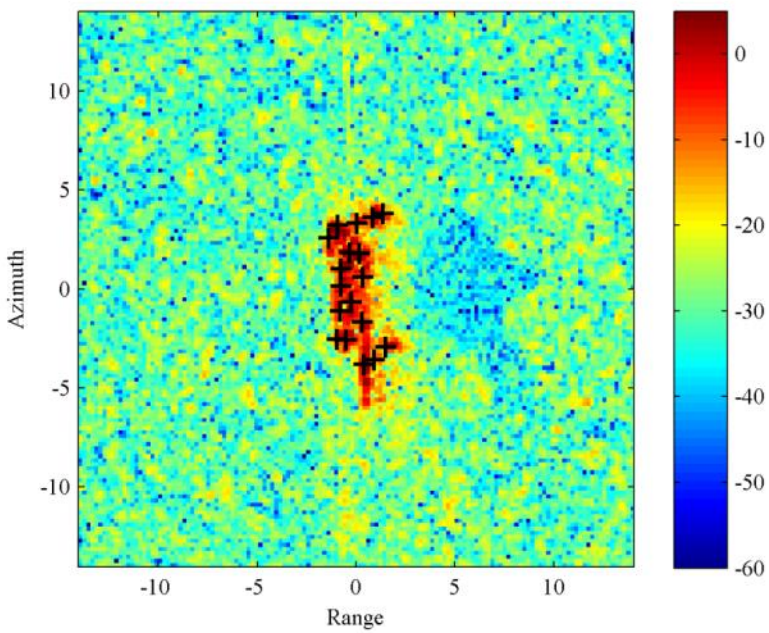

散射体提取

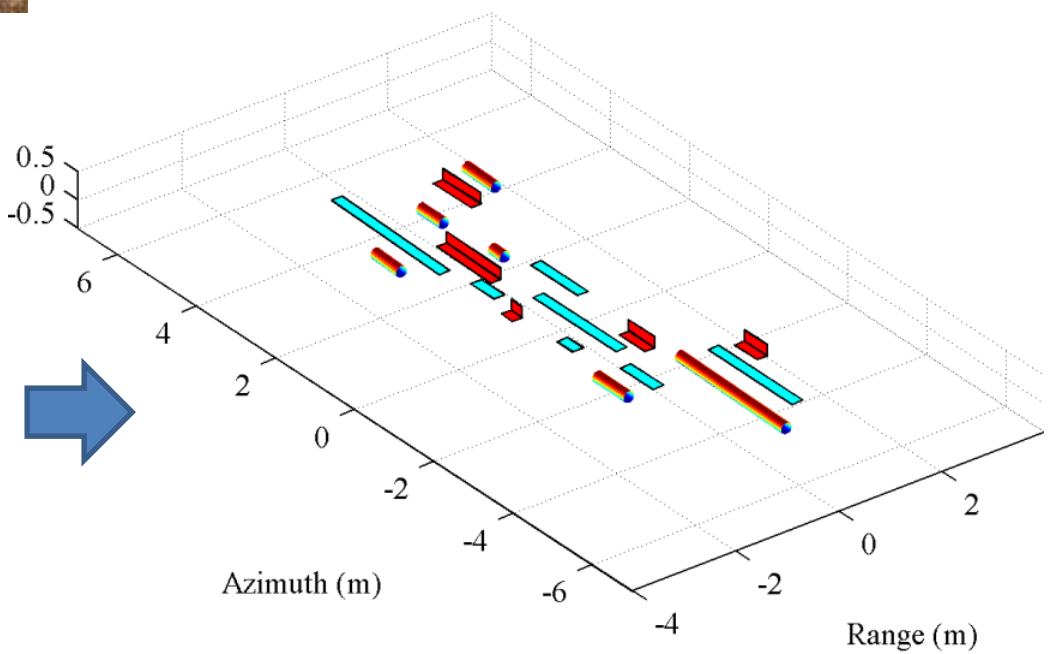



几何散射体重构

# MSTAR



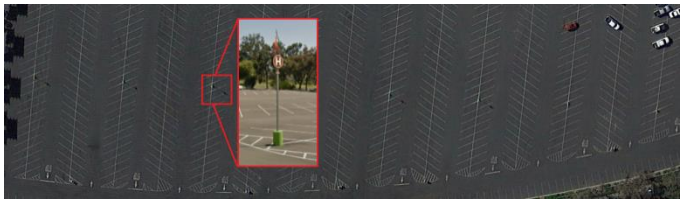



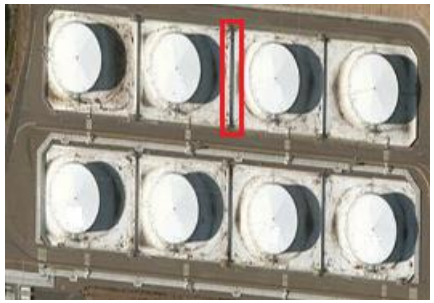




MSTAR (X-band, 0.3m)

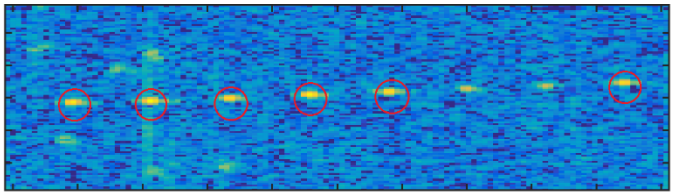
# UAVSAR Data

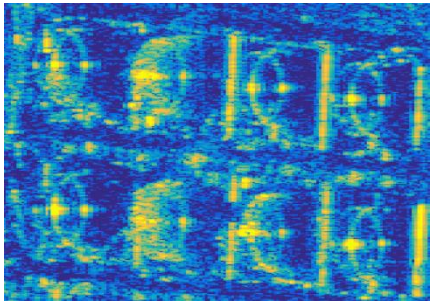
光学图像



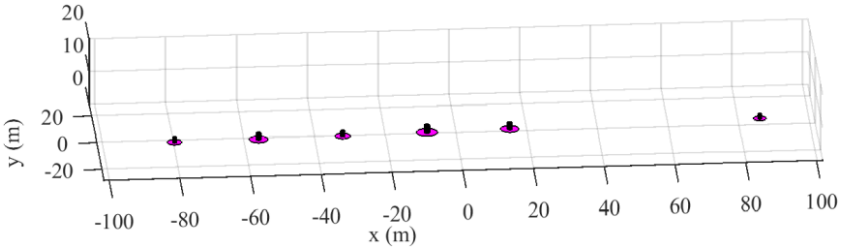


SAR成像

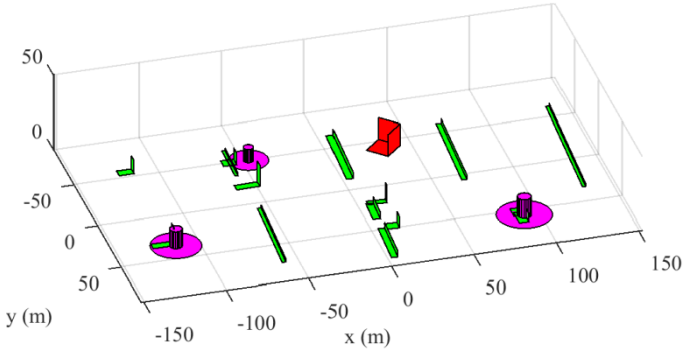




重构结果



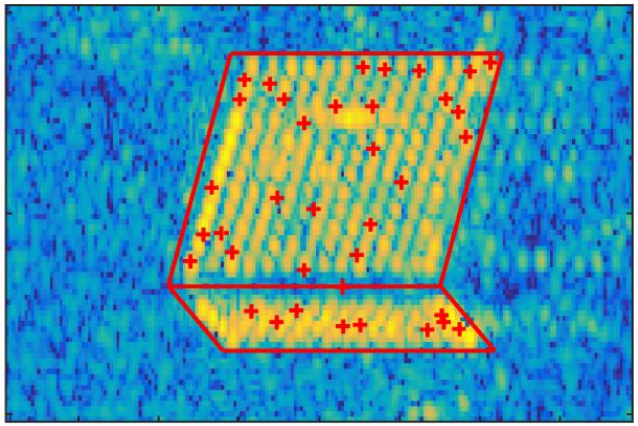

停车场指示牌的几何重构



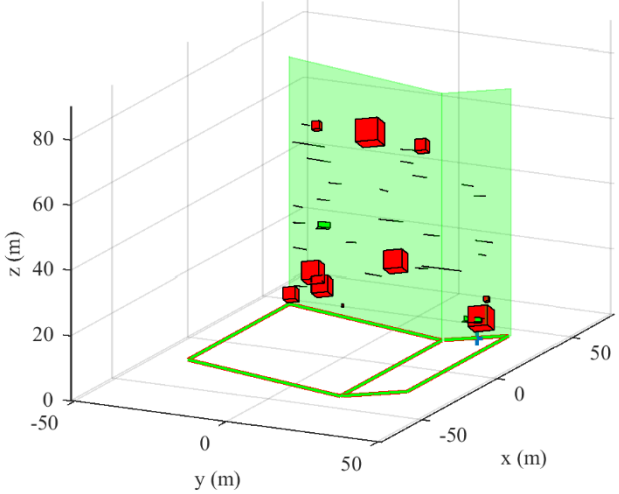

油库区域的几何散射体重构

# TerraSAR-X

光学图像



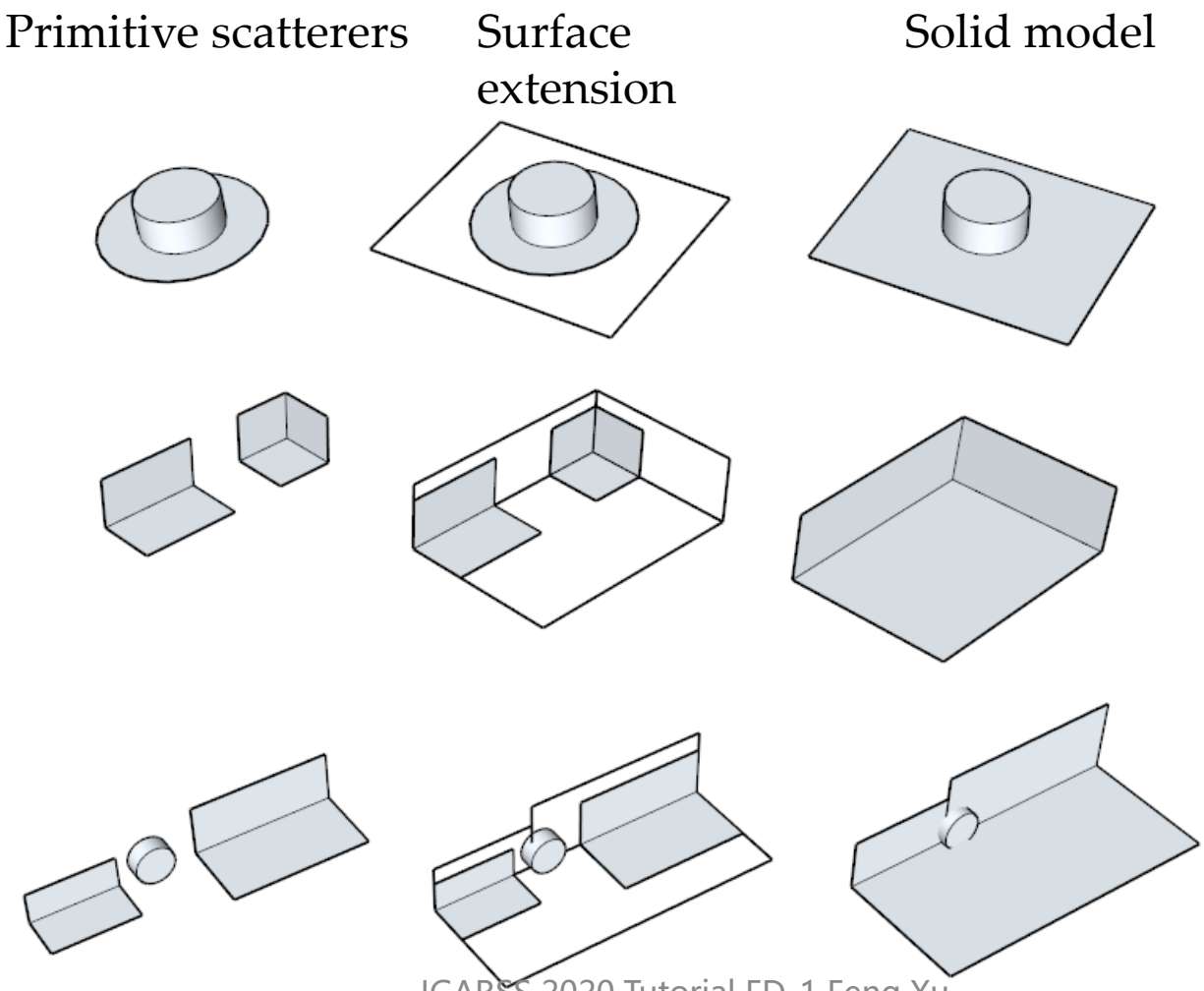
SAR成像




几何散射体重构结果

# Toward 3D Reconstruction

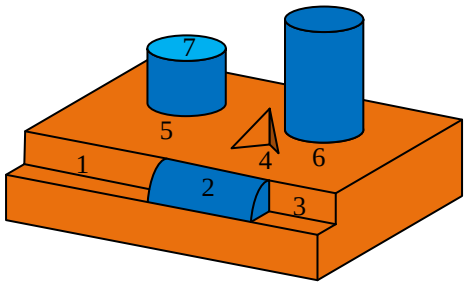
- Surface extension



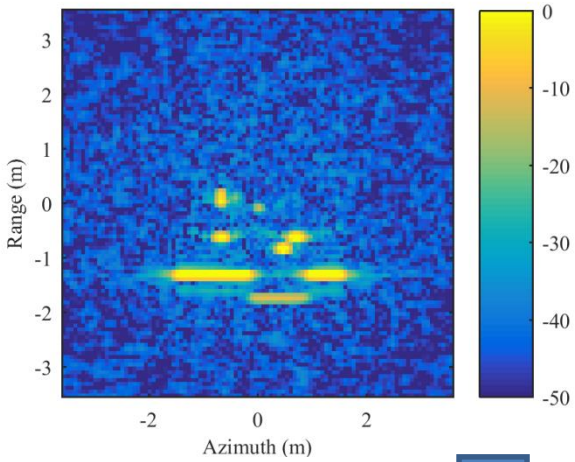
# Reconstruction of SLICY Model

- **From 2D SAR to 3D Visualization**

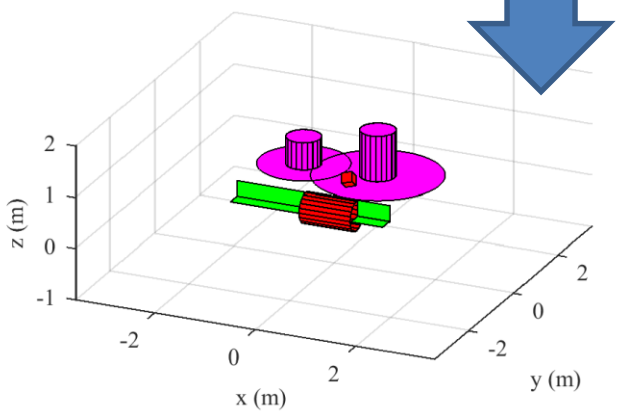
1. 二面角
2. 圆柱
3. 二面角
4. 三面角
5. 帽顶
6. 帽顶
7. 腔体



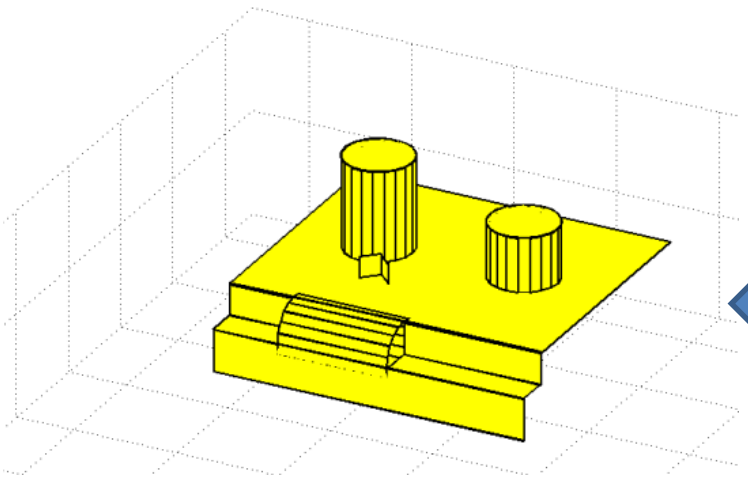

SLICY Model




SAR Image



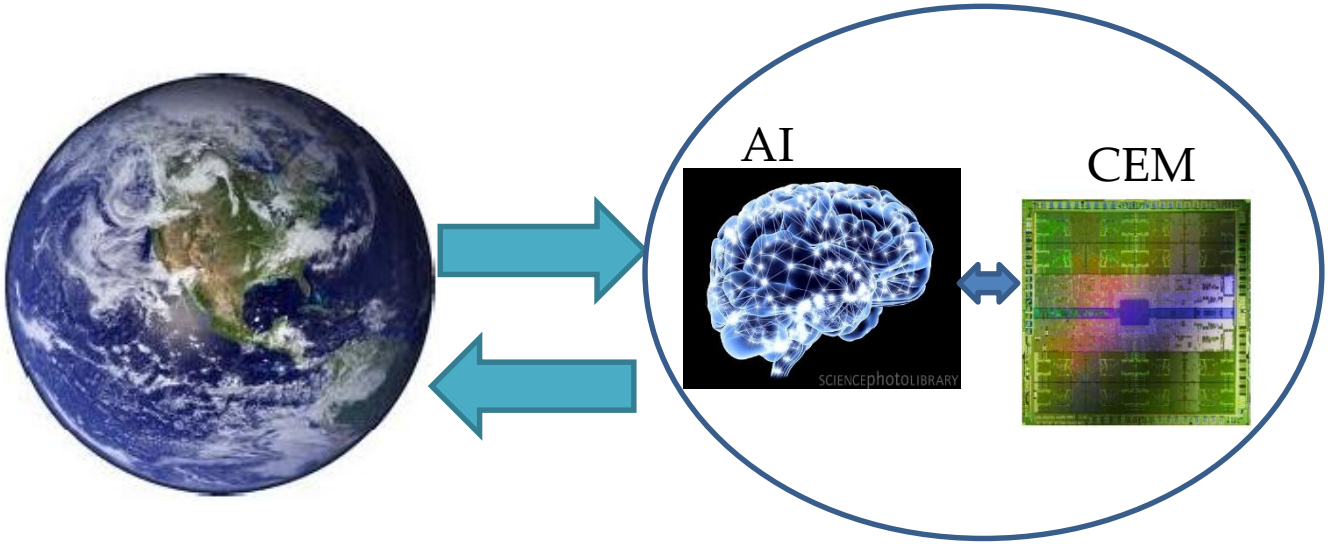

Scatterer Extraction



- [F. Xu et al., “A Preliminary Study on SAR Advanced Information Retrieval and Scene Reconstruction”, IEEE GRSL 2016]

# Microwave Vision

- Science revolution enable human to precisely describe the world.
- Technology revolution enable human to efficiently predict the world.
- Empower AI with Computational Physics.




[徐丰，金亚秋．从物理智能到微波视觉[J]．科技导报，2018，36(10)：30-44.]

# Thank you

**fengxu@fudan.edu.cn**
homepage.fudan.edu.cn/fengxu
github.com/fudanxu

徐丰、王海鹏、金亚秋
《合成孔径雷达图像智能解译》
科学出版社，2020年

復旦大學 电磁波信息科学教育部重点实验室
Key Laboratory of EMW Information,Fudan University