

# Lecture 07 – CNN Applications and Tricks

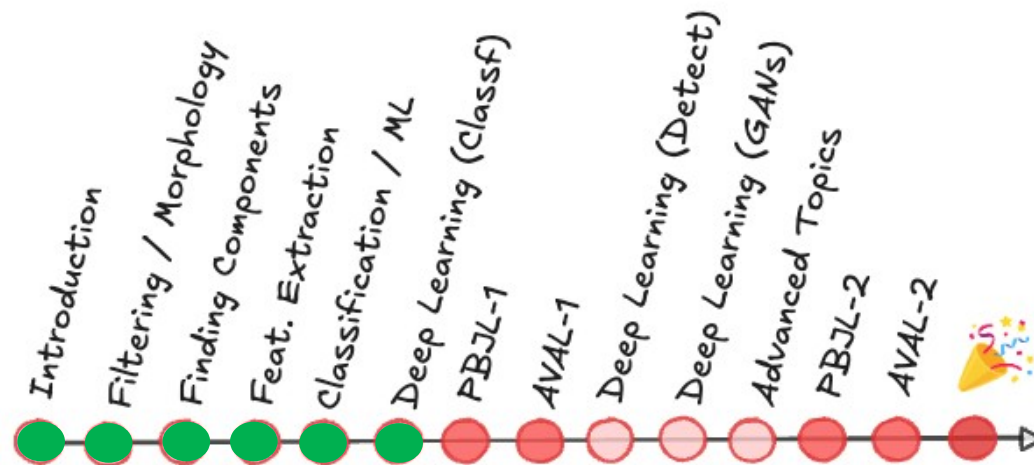
Prof. André Gustavo Hochuli

[gustavo.hochuli@pucpr.br](mailto:gustavo.hochuli@pucpr.br)

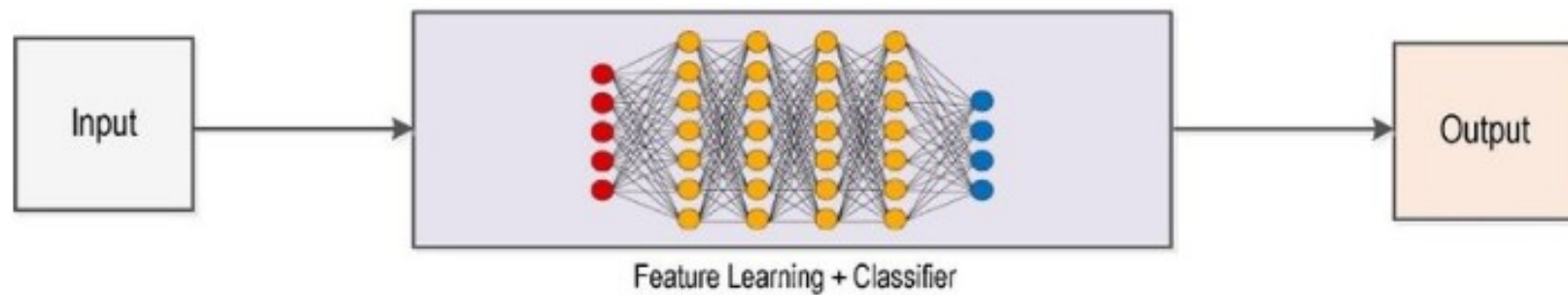
[aghochuli@ppgia.pucpr.br](mailto:aghochuli@ppgia.pucpr.br)

# Topics

- Convolutional Neural Network
  - Basic Concepts
  - Architecture and Hyper Parameters
  - Data Augmentation
  - Transfer-Learning
  - Applications
- Practice

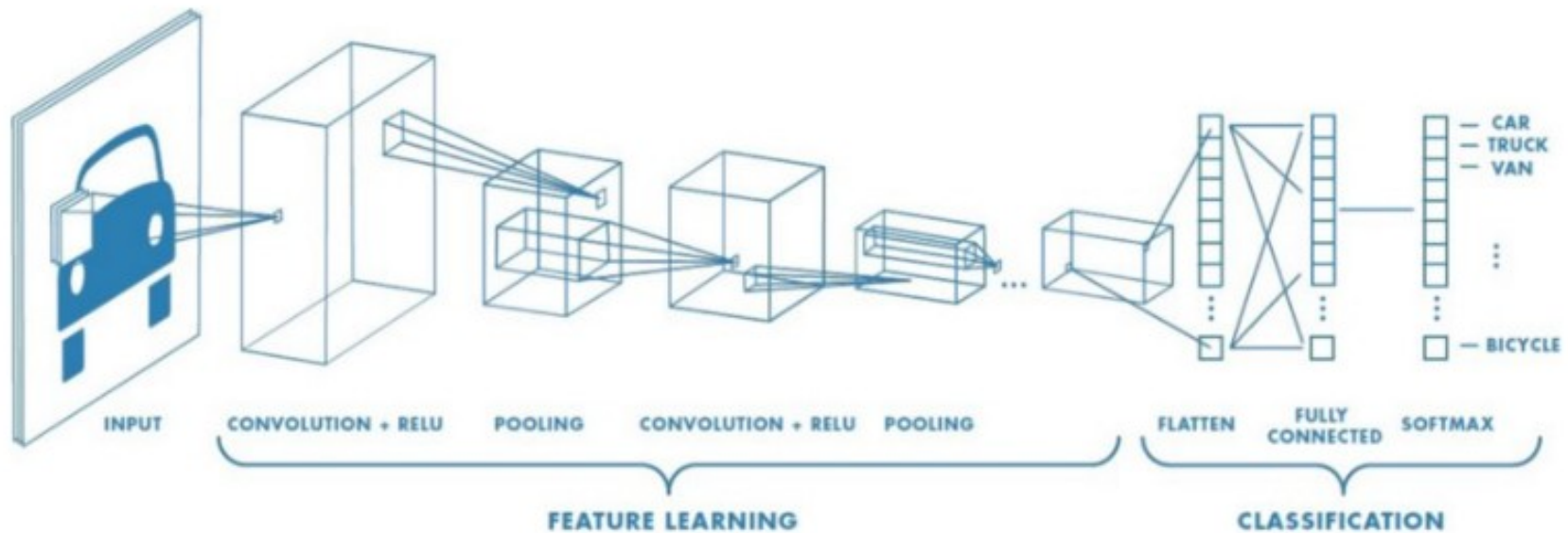


# Deep Learning Pipeline



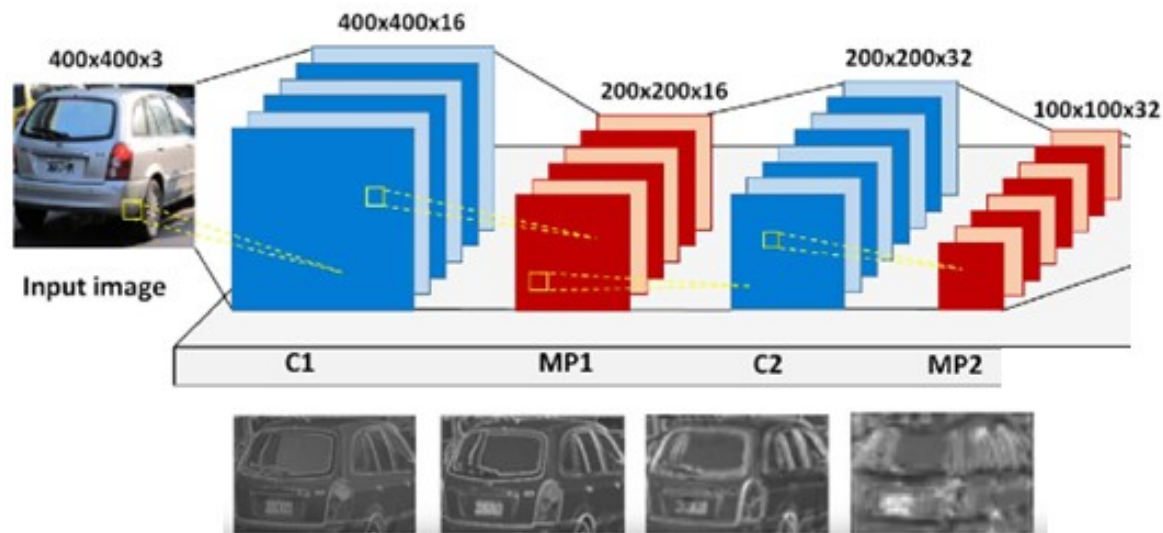
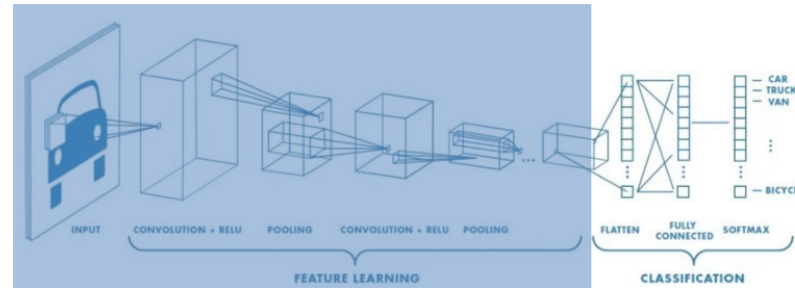
# Convolutional Neural Network

- CNN



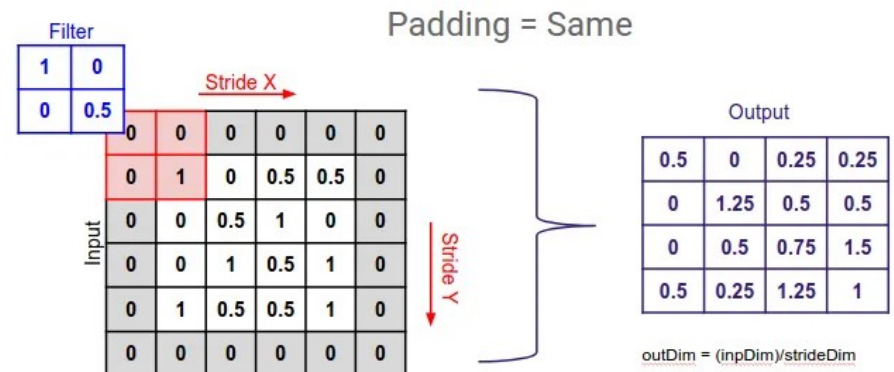
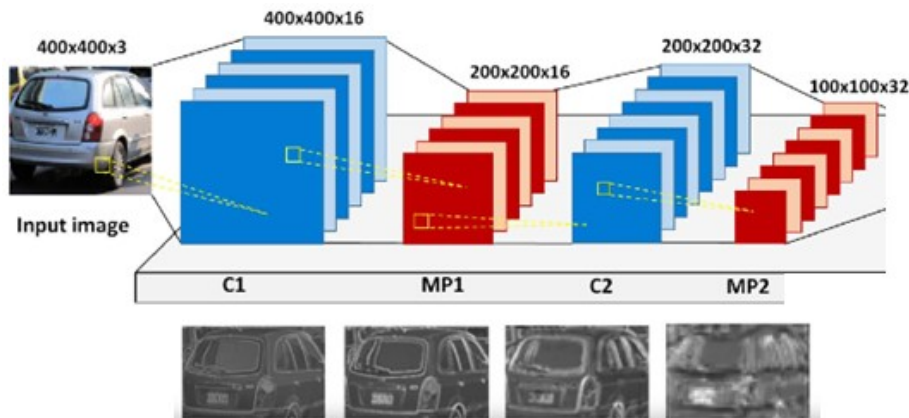
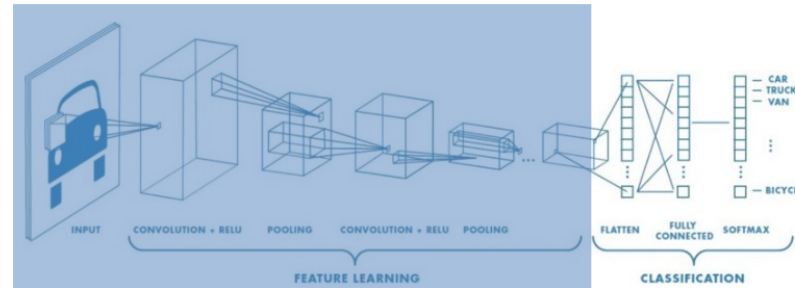
# Convolutional Neural Network

- Feature Extraction



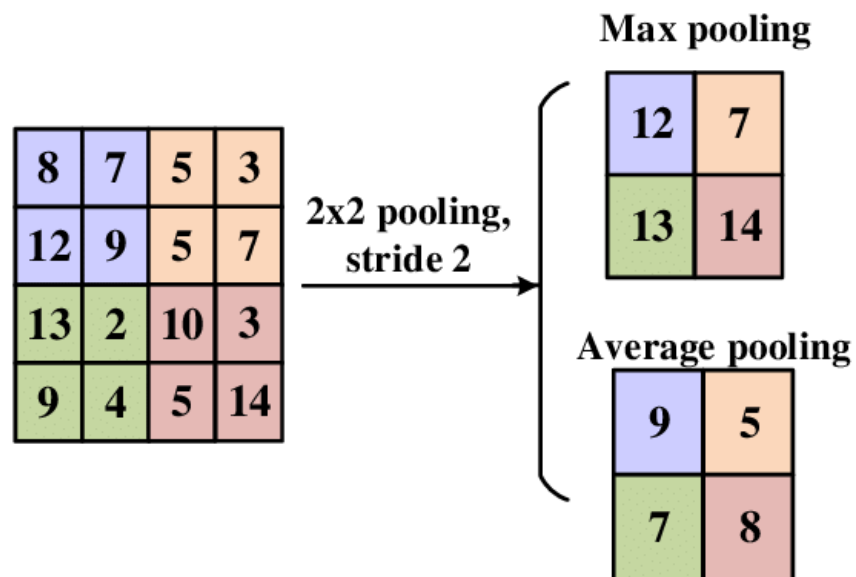
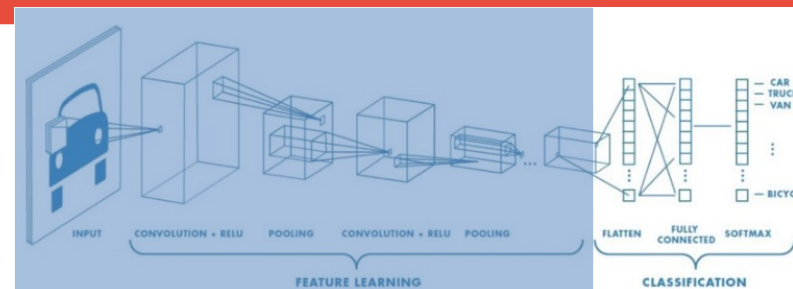
# Convolutional Neural Network

- Convolutional Layer (Learnable Filters)
  - Padding
  - Stride
  - Kernel Size
  - Number of Filters



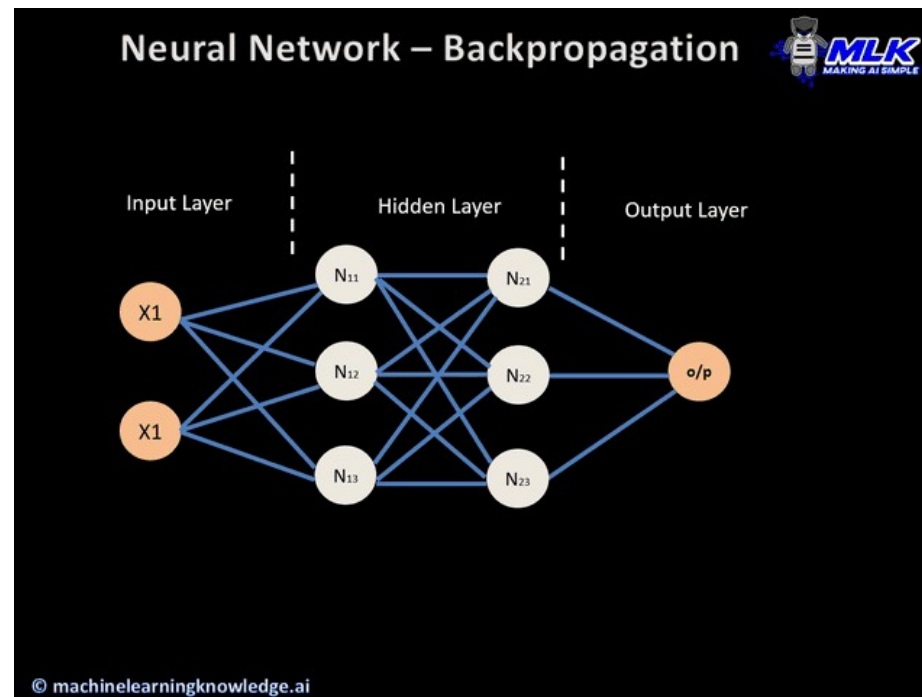
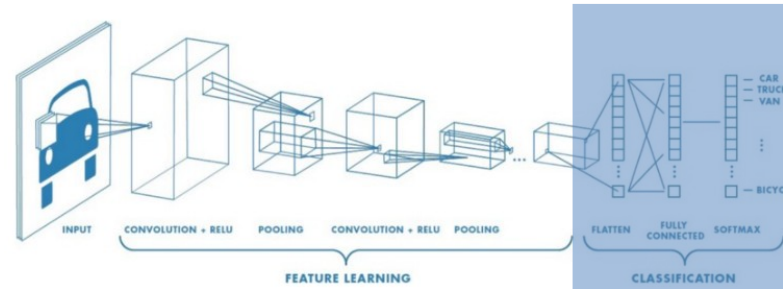
# Convolutional Neural Network

- Pooling Layer
  - Reduce Spatial Dimensions
  - Translation-Invariant
  - Common Filter
    - Max: Preserve the “strongest” features
    - Average: Smooth features, preserves general representations



# Convolutional Neural Network

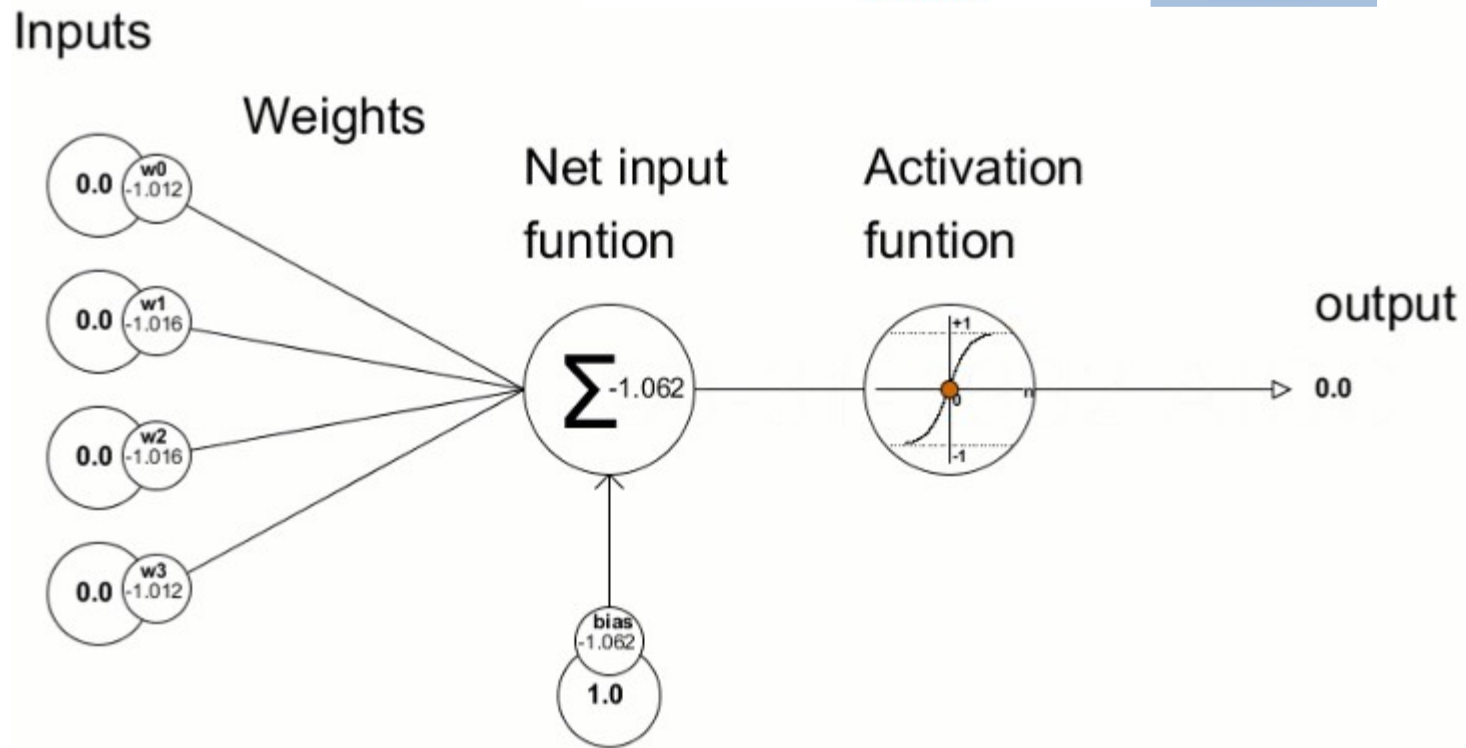
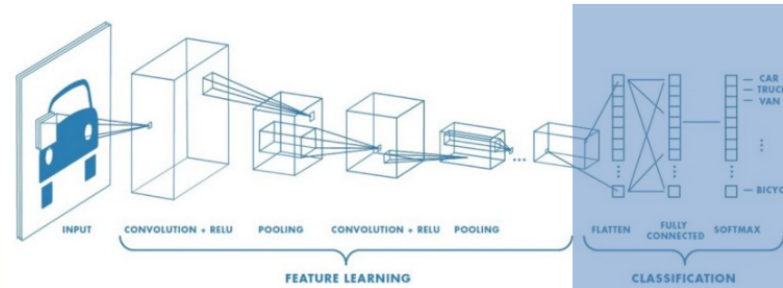
- Classification
- Forward and Back Propagation





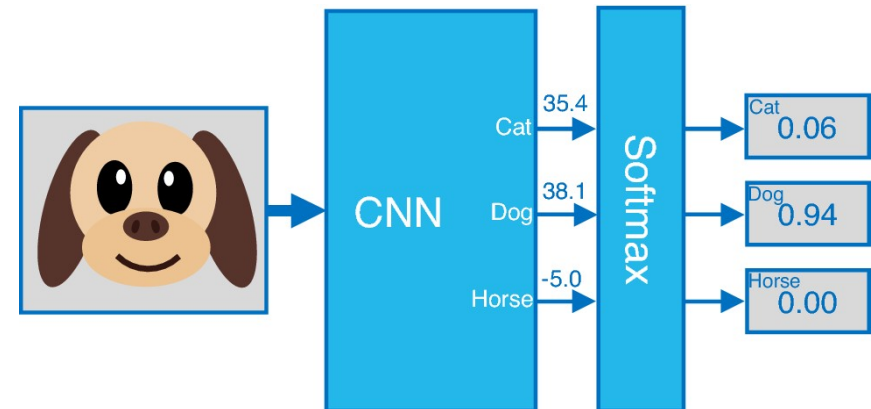
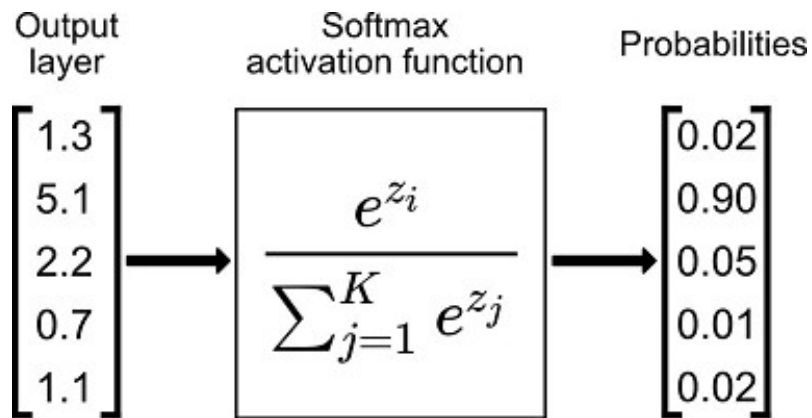
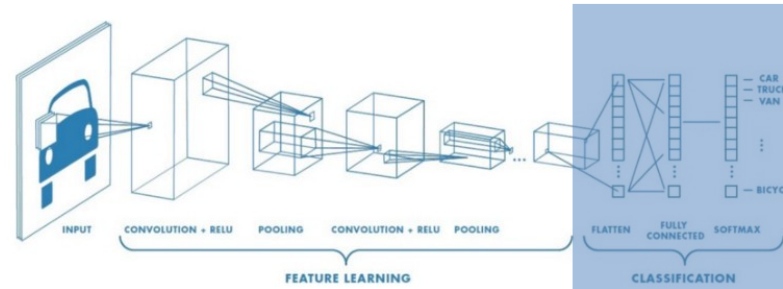
# Convolutional Neural Network

- Forward and Back Propagation



# Convolutional Neural Network

- Softmax



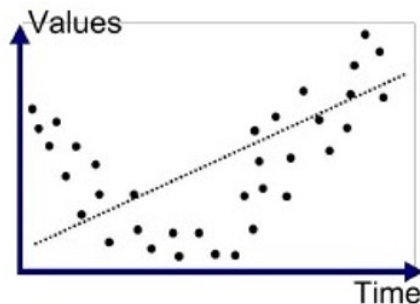
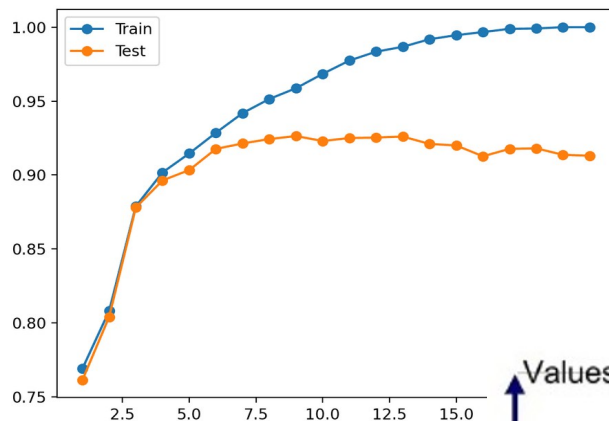
# Convolutional Neural Network

- Lets code our first CNN from scratch

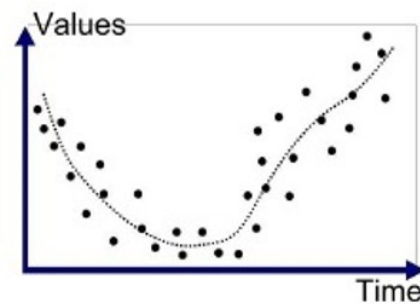
[Lecture 07 - CNN Architecture](#)

# Overfitting

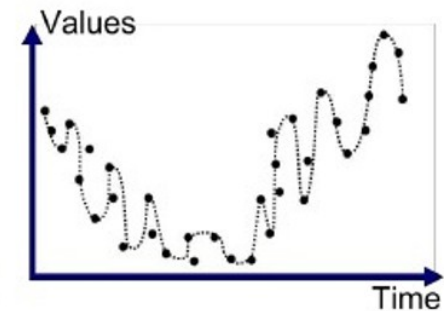
- Overfitting occurs when a model captures noise or specific patterns in the training data, impairing its ability to generalize to unseen data. Strategies such as regularization, dropout, data augmentation, and transfer learning help mitigate this by controlling model complexity and leveraging pre-learned features.



Underfitted



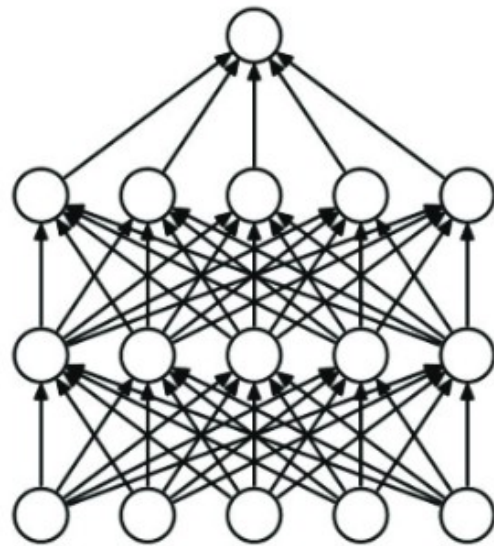
Good Fit/Robust



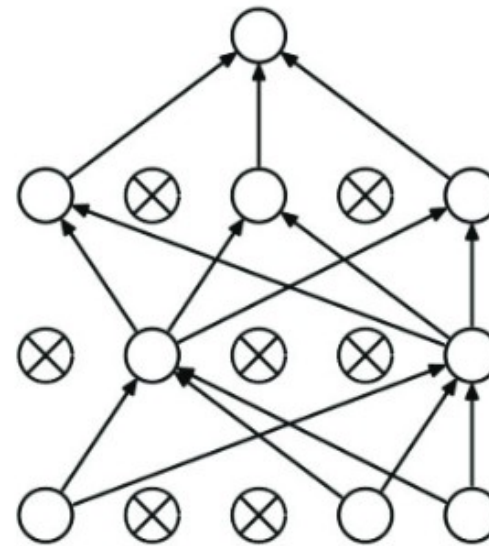
Overfitted

# Dropout

- Dropout is a regularization technique that randomly deactivates a fraction of neurons during training, forcing the model to learn redundant representations and reducing overfitting.



(a) Standard Neural Network



(b) Neural Net with Dropout

# Data Augmentation

- Enlarge the dataset with synthetic samples
  - Rotation
  - Crop
  - Brightness

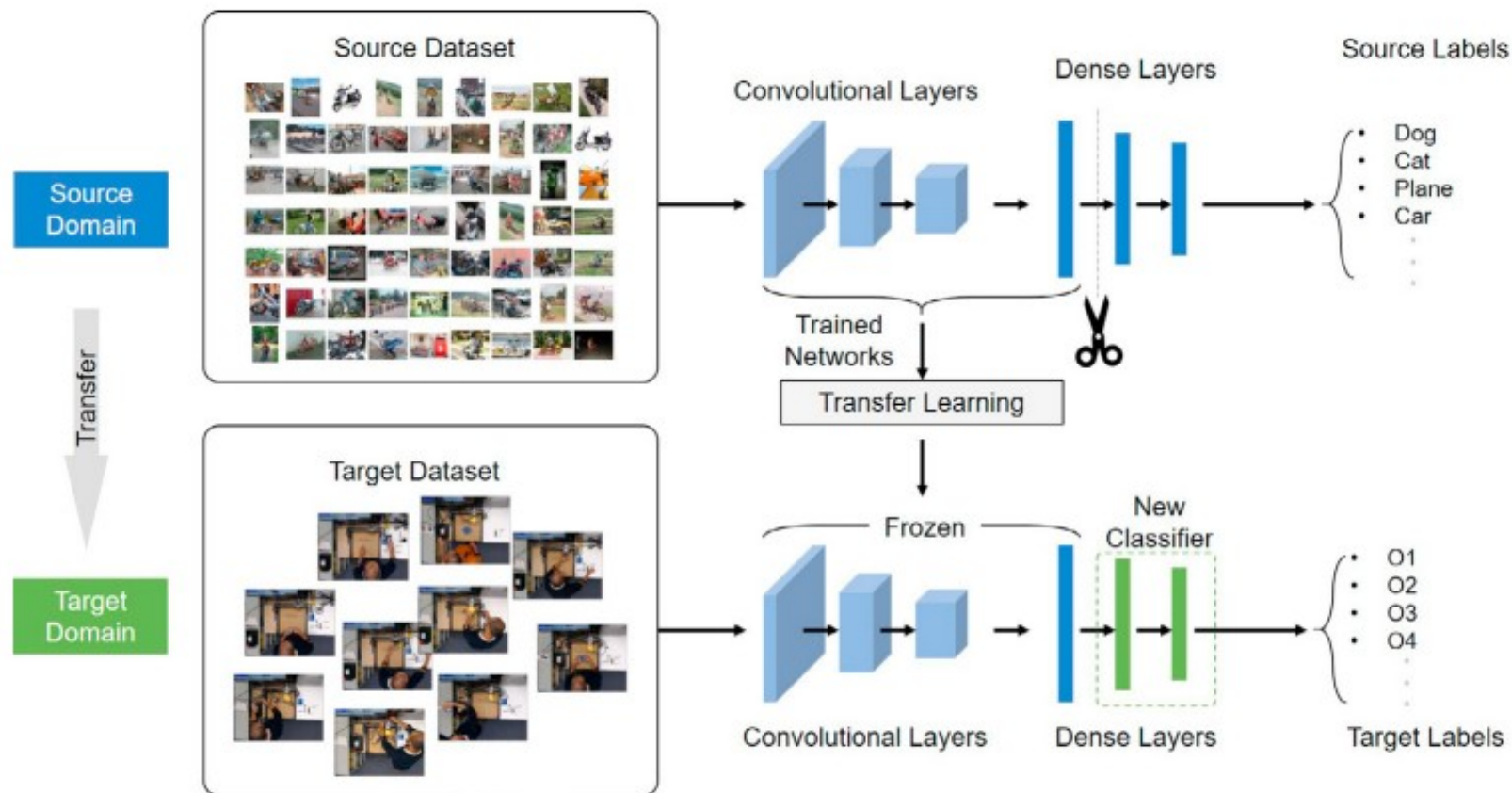


**Image  
Augmentation**



# Transfer Learning

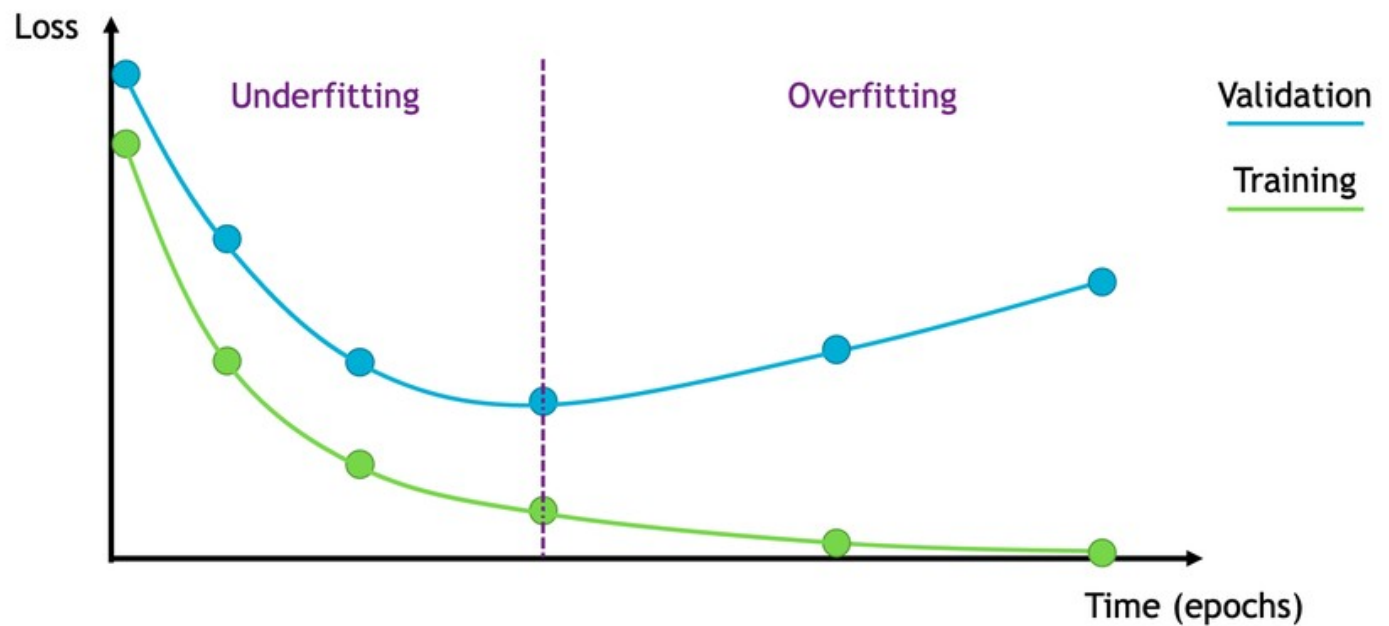
- Weight Sharing
- Feature Extraction weights are frozen (or not...) during learning





# Miscellaneous

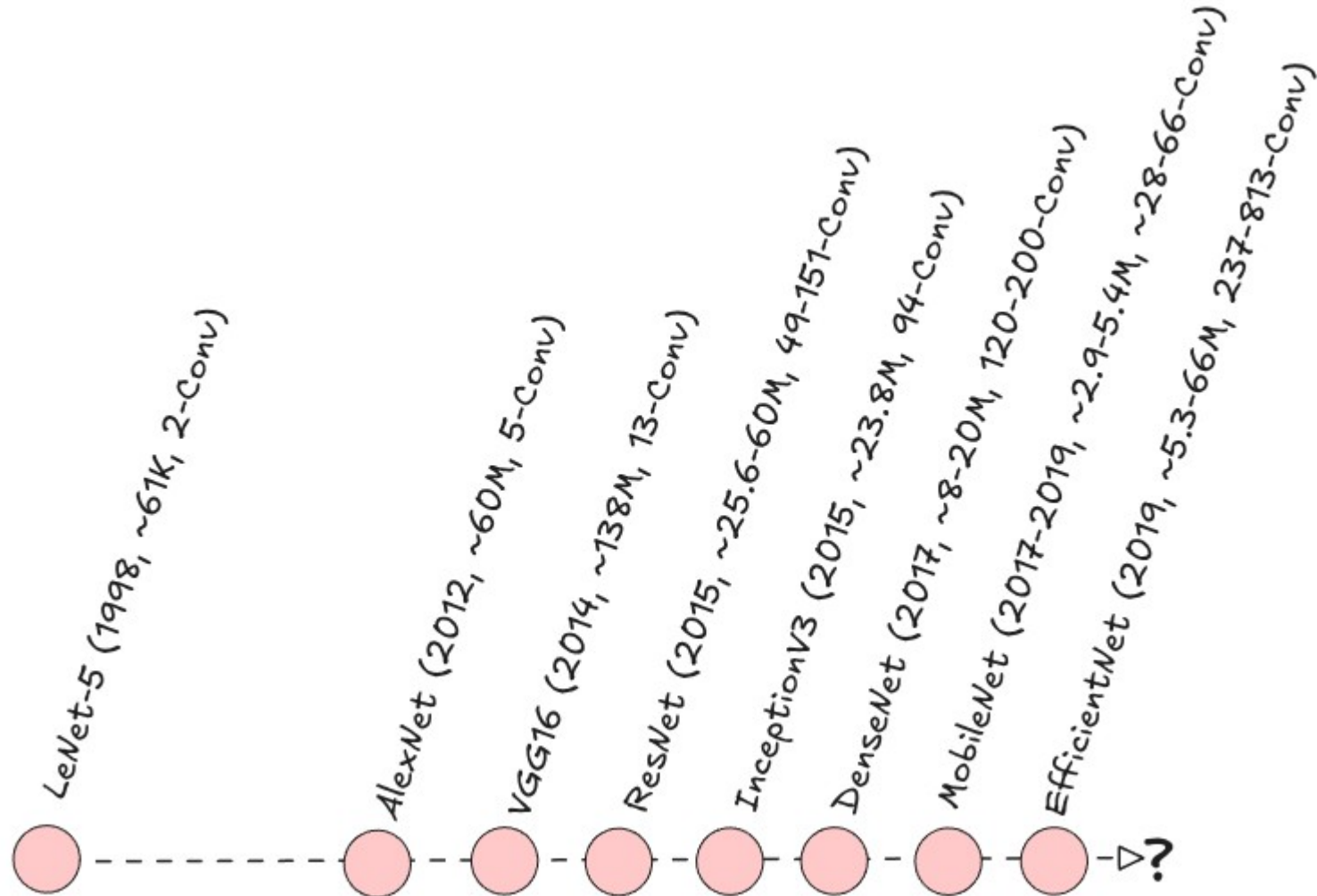
- Save and Load Weights
- Model Checkpoint
- Resuming Training
- Early Stopping



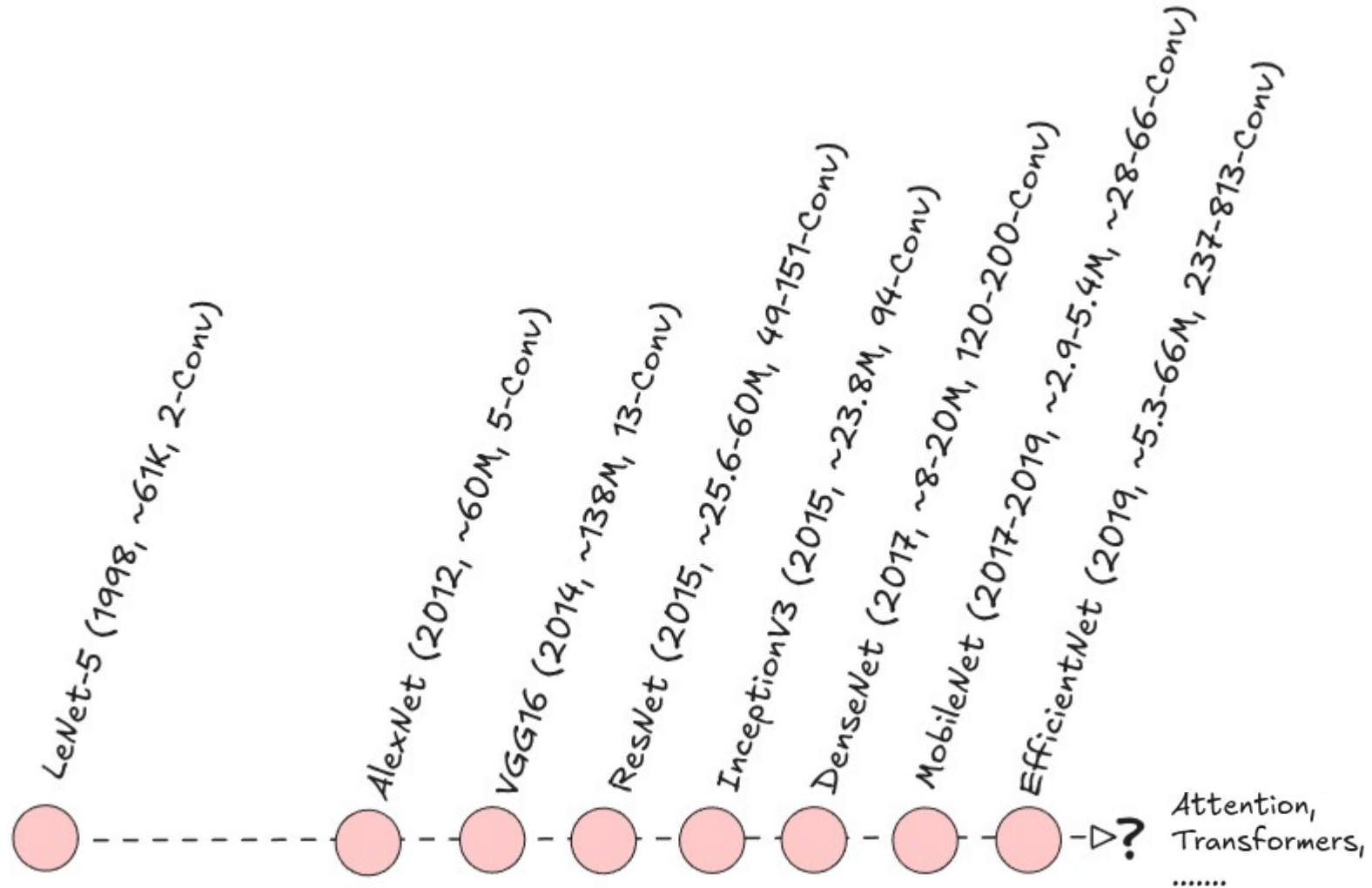


# State of The Art Architectures

# CNN Architectures Timeline



# CNN Architectures Timeline

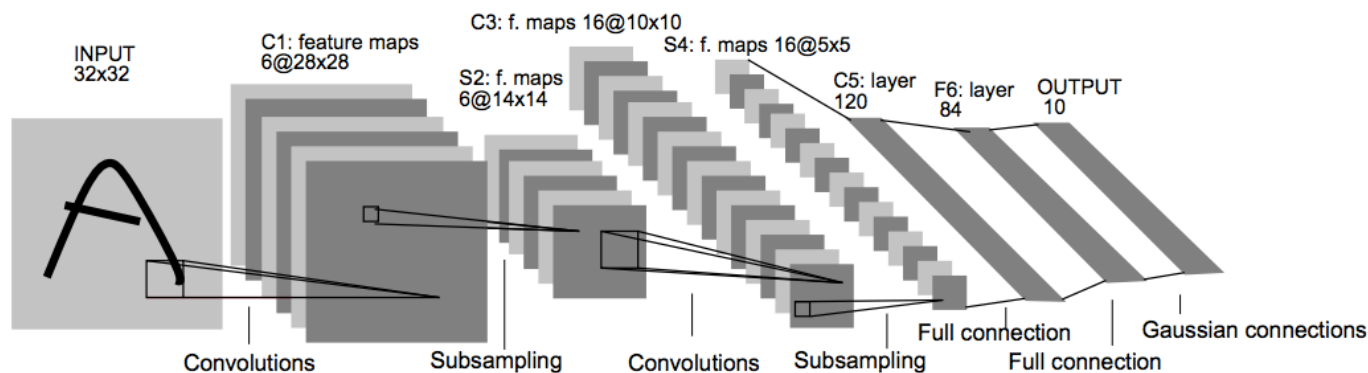


# Lenet-5

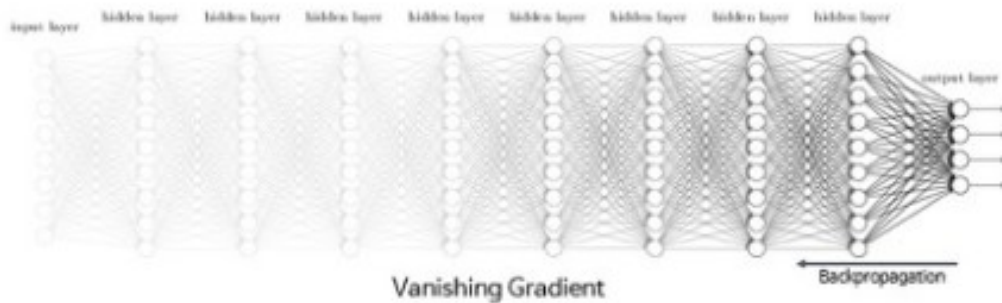
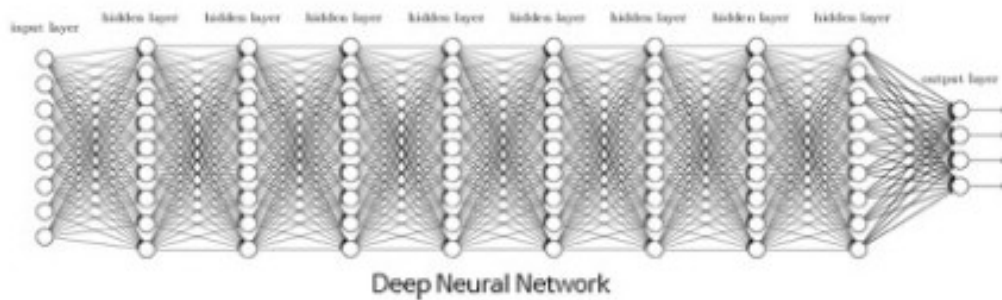
## Gradient-Based Learning Applied to Document Recognition

YANN LECUN, MEMBER, IEEE, LÉON BOTTOU, YOSHUA BENGIO, AND PATRICK HAFFNER

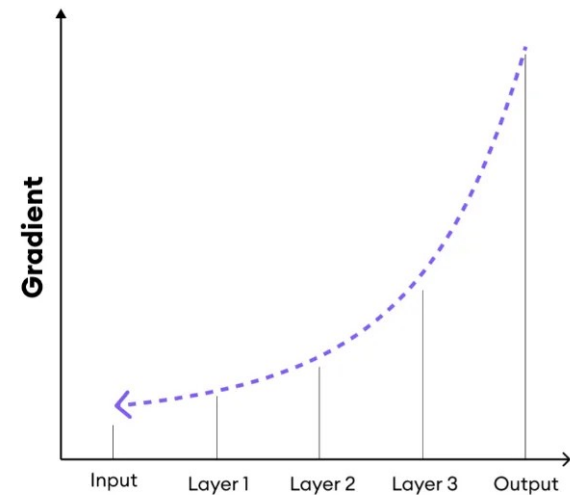
- Architecture: 2 Conv + Pool layers + 3 Fully Connected layers.
- MNIST digit recognition.
- Learned convolutional filters for feature extraction.
- Parameters: ~60k trainable weights.
- First widely cited CNN demonstrating effectiveness for vision tasks.



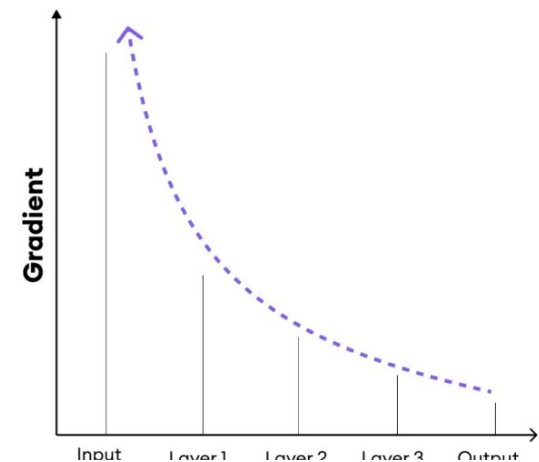
# Vanish Gradient



Vanishing Gradient

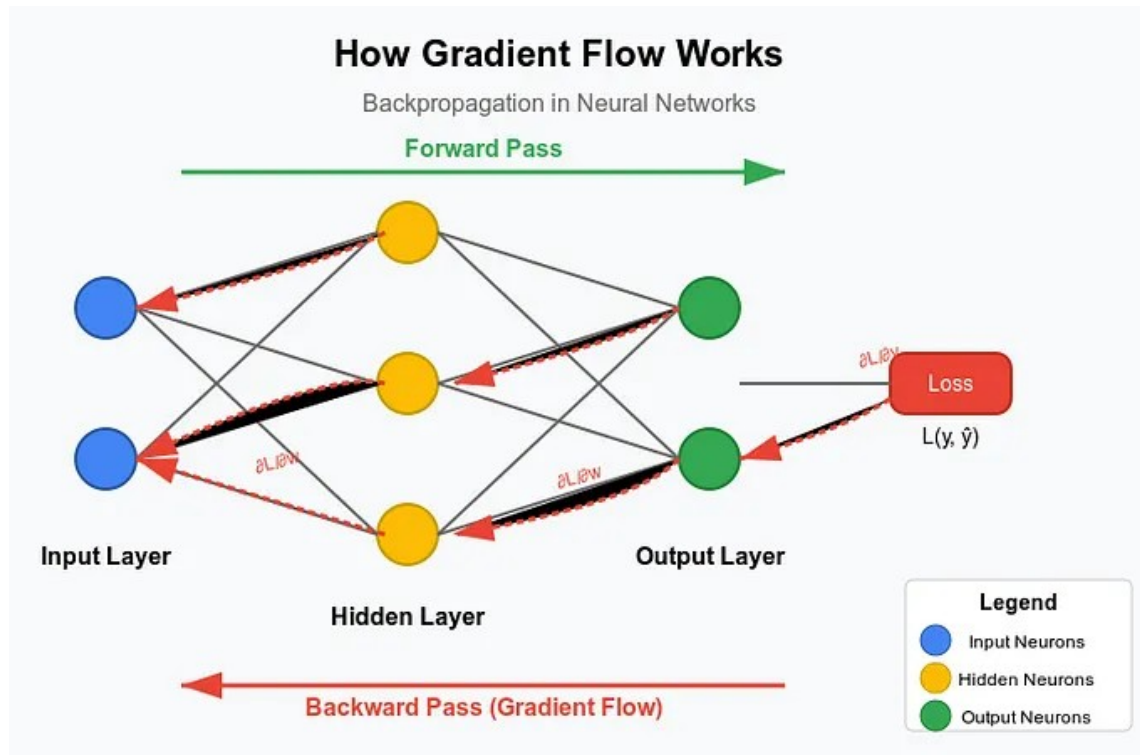


Exploding Gradient



# Vanish Gradient

- Backpropagation multiplies gradients layer by layer using the chain rule.



Artificial Neuron

$$z = \sum_{i=1}^n w_i x_i + b$$

Activation

$$y = f(z) = f\left(\sum_{i=1}^n w_i x_i + b\right)$$

$$\frac{d}{dx} [f(g(x))] = f'(g(x))g'(x)$$

# Vanish Gradient

Letter | Published: 09 October 1986

## Learning representations by back-propagating errors

[David E. Rumelhart](#), [Geoffrey E. Hinton](#) & [Ronald J. Williams](#)

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}$$

Gradient of the loss with respect to the weights

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} * \frac{\partial z}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} = a - y$$

Gradient of the loss with respect to the bias

$$\begin{aligned} \frac{\partial \mathcal{L}(y, a)}{\partial a} &= \frac{\partial \mathcal{L}}{\partial a} = \frac{\partial}{\partial a} \left[ - (y * \log(a) + (1 - y) * \log(1 - a)) \right] = \\ &= - \frac{\partial}{\partial a} \left( y * \log(a) \right) - \frac{\partial}{\partial a} \left( (1 - y) * \log(1 - a) \right) = \\ &= - \left( 0 * \log(a) + \frac{y}{a} \right) - \left( 0 * \log(1 - a) - \frac{(1 - y)}{(1 - a)} \right) = - \frac{y}{a} + \frac{1 - y}{1 - a} \end{aligned}$$

Gradient of the lost function with respect to the predicted value

$$w := w - \alpha \frac{\partial \mathcal{L}}{\partial w} = w - \alpha * (a - y)x$$

$$b := b - \alpha \frac{\partial \mathcal{L}}{\partial b} = w - \alpha * (a - y)$$

Weight and bias update

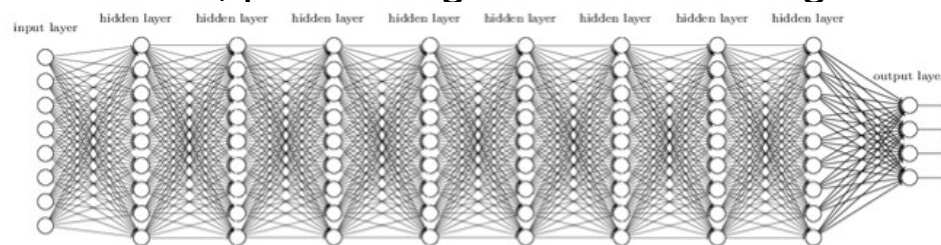
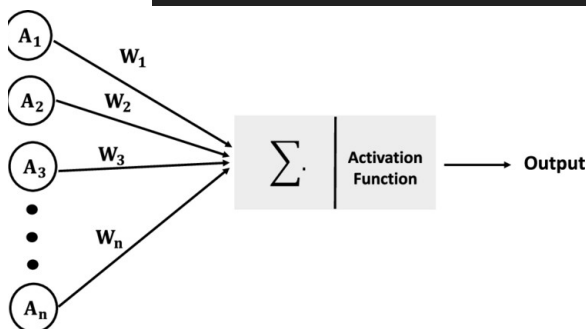


# Vanish Gradient

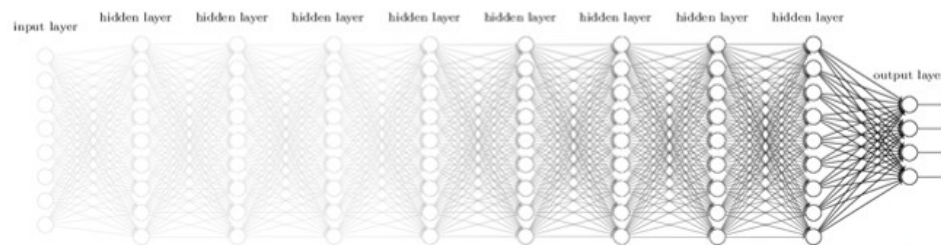
- Backpropagation multiplies gradients layer by layer using the chain rule.
- If activation functions (like sigmoid or tanh) squash inputs into small ranges, their derivatives are less than 1.
- Multiplying many numbers less than 1 across deep layers causes the gradient to shrink exponentially toward zero.
- As a result, weight updates in early layers vanish, preventing effective training.

Activation

$$y = f(z) = f\left(\sum_{i=1}^n w_i x_i + b\right)$$



Deep Neural Network



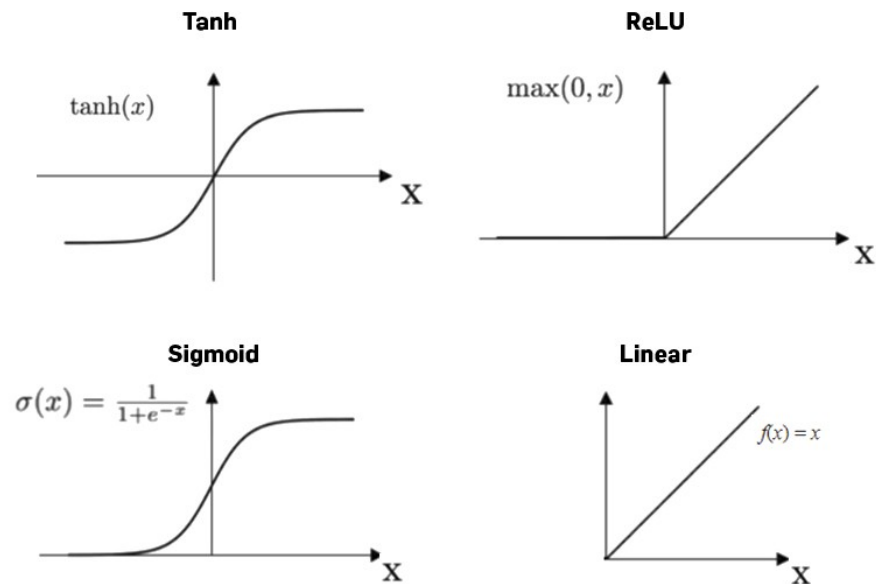
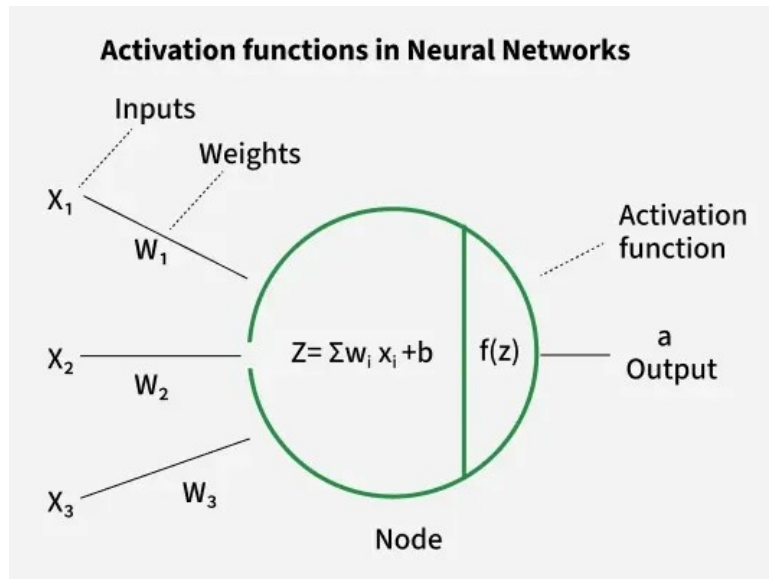
Vanishing Gradient



# Mitigating The Vanish

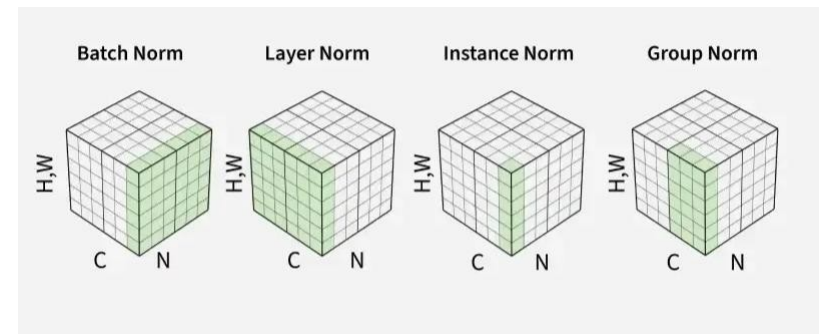
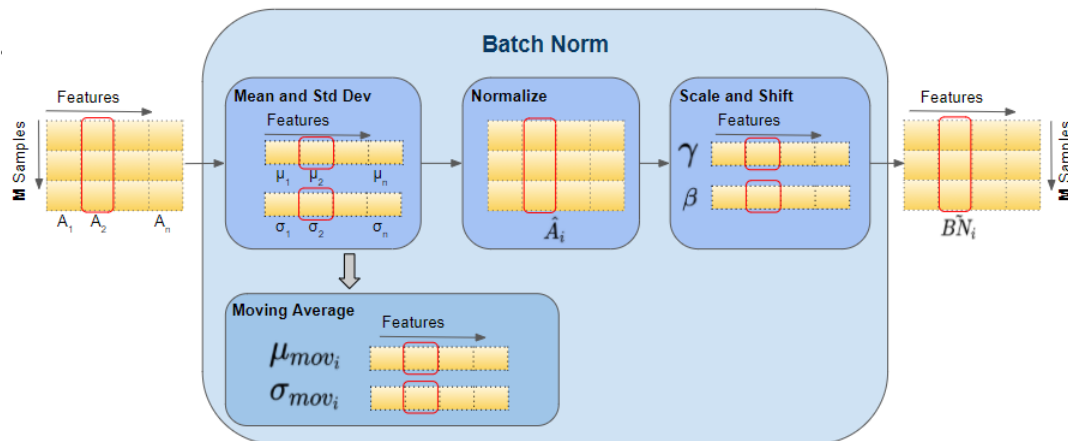
- ReLu Activation

- Prevents saturation: Sigmoid/tanh saturate (gradient  $\approx 0$ ) for large or small values; ReLU does not.
- Maintains gradient flow: Constant derivative (1) for  $x > 0$  preserves the backpropagation signal.
- Computational simplicity: Direct calculation, no exponentials required.



# Mitigating The Vanish

- Batch Normalization (BN)
  - BN normalizes the output of a layer before applying the activation function
  - For a given mini-batch, it computes the mean and variance of each feature/channel and scales the activations to have zero mean and unit variance.
  - BN keeps the distribution of activations stable, which prevents gradients from shrinking too much in deep networks.
  - It reduces the dependency of gradient magnitude on the scale of previous layers' weights.
  - Helps gradients propagate more effectively.



# AlexNet

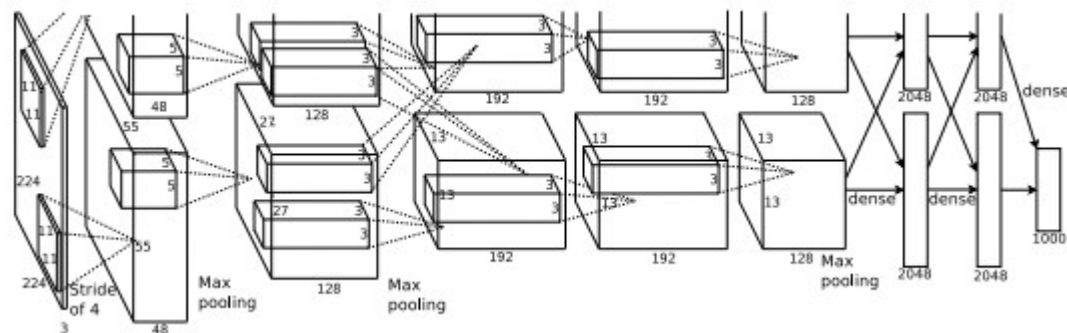
## ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca

- Architecture: 5 Conv layers + 3 Fully Connected layers, ReLU activations, Dropout.
- IMAGENET Challenge 2012 – Error 16.4%
- Introduced ReLU, GPU training, data augmentation, and dropout.
- Parameters: ~61M.
- Sparked modern deep learning revolution in computer vision.



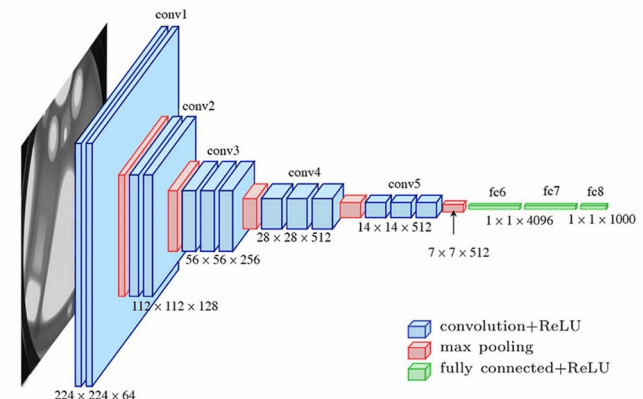
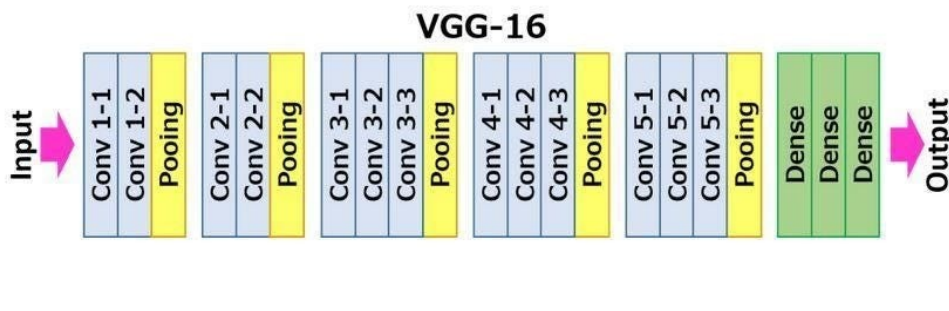
# VGG16

## VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan\* & Andrew Zisserman\*

Visual Geometry Group, Department of Engineering Science, University of Oxford  
{karen, az}@robots.ox.ac.uk

- Architecture: 13 Conv layers + 3 Fully Connected layers, small 3×3 filters.
- IMAGENET Challenge 2012 – Error 6.7%
- Key Features: Deep network using uniform architecture with small filters.
- Parameters: ~138M.
- Showed depth improves performance; standard baseline for many tasks.



# ResNet

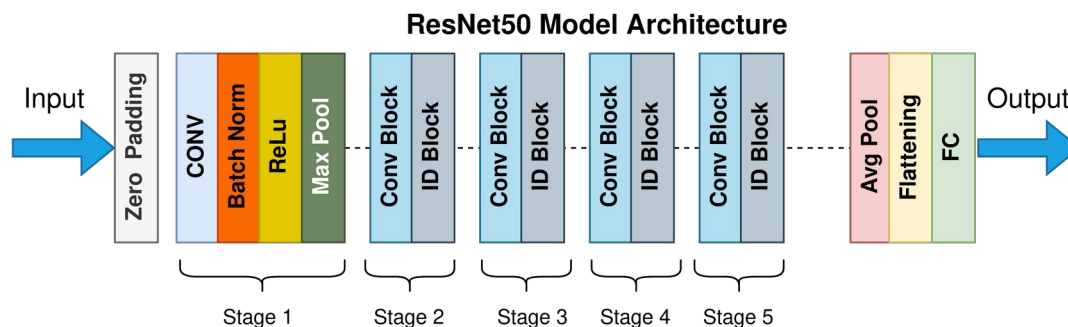
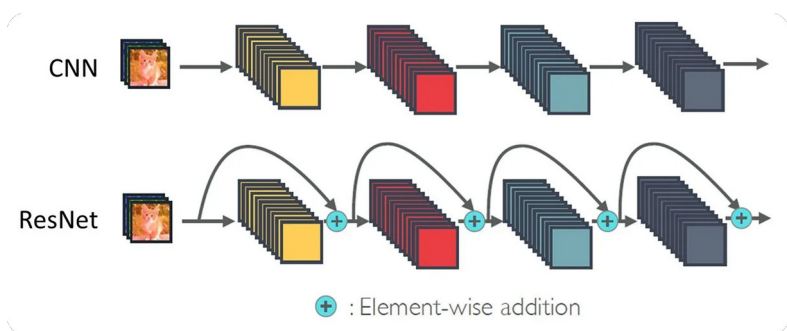
## Deep Residual Learning for Image Recognition

Kaiming He    Xiangyu Zhang    Shaoqing Ren    Jian Sun

Microsoft Research

{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

- Architecture: 50~152 layers with residual (skip) connections.
- IMAGENET Challenge 2012 – Error 3.6%
- Residual blocks enable training very deep networks (\*).
- Parameters: ~25M to ~60M.
- Mitigates vanishing gradient problem; enabled ultra-deep networks.



(\*) At this point the focus shifts from parameter-heavy networks to efficient architectures with better feature representation.

# Inception

## Going Deeper with Convolutions

Christian Szegedy<sup>1</sup>, Wei Liu<sup>2</sup>, Yangqing Jia<sup>1</sup>, Pierre Sermanet<sup>1</sup>, Scott Reed<sup>3</sup>,  
Dragomir Anguelov<sup>1</sup>, Dumitru Erhan<sup>1</sup>, Vincent Vanhoucke<sup>1</sup>, Andrew Rabinovich<sup>4</sup>

<sup>1</sup>Google Inc. <sup>2</sup>University of North Carolina, Chapel Hill

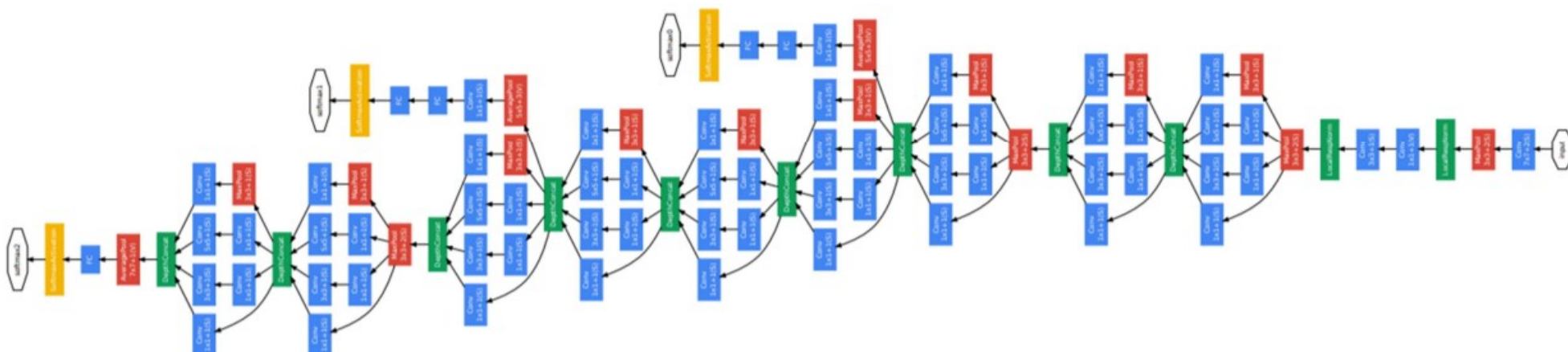
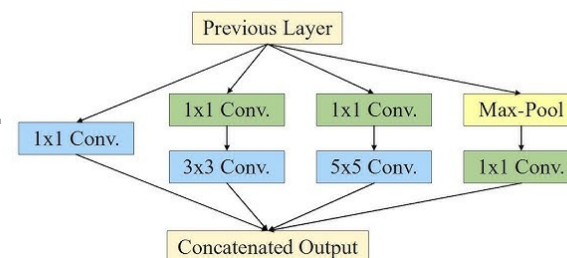
<sup>3</sup>University of Michigan, Ann Arbor <sup>4</sup>Magic Leap Inc.

<sup>1</sup>{szegedy, jiaayq, sermanet, dragomir, dimitru, vanhoucke}@google.com

<sup>2</sup>wliu@cs.unc.edu, <sup>3</sup>reedscott@umich.edu, <sup>4</sup>arabinovich@magic Leap Inc.

- Architecture: Inception modules with multi-scale convolutions.
- ImageNet classification
- **Efficient computation** via factorized convolutions and dimension reduction.
- Parameters: ~23.9M.
- Combines depth and width efficiently; high accuracy with moderate compute

## INCEPTION MODULE





# DenseNet

## Densely Connected Convolutional Networks

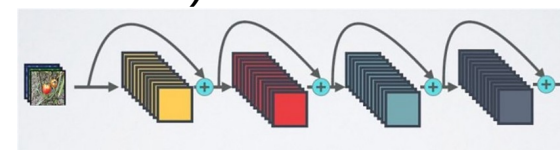
Gao Huang\*  
Cornell University  
gh349@cornell.edu

Zhuang Liu\*  
Tsinghua University  
liuzhuang13@mails.tsinghua.edu.cn

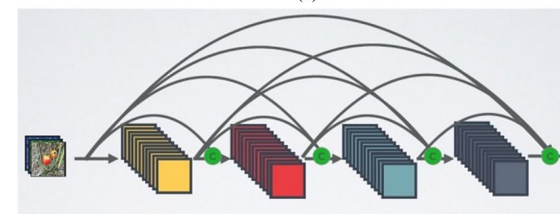
Laurens van der Maaten  
Facebook AI Research  
lvdmaaten@fb.com

Kilian Q. Weinberger  
Cornell University  
kqw4@cornell.edu

- Architecture: 121 ~201 layers with dense connections (feature reuse). 1X1 Convs
- ImageNet classification.
- Each layer receives inputs from **all previous layers**.
- Parameters: ~8M ~20M.
- Significance: Reduces parameters while maintaining high performance; encourages feature reuse.



(a)



(b)

Layers	Output Size	DenseNet-121( $k = 32$ )	DenseNet-169( $k = 32$ )	DenseNet-201( $k = 32$ )	DenseNet-161( $k = 48$ )
Convolution	$112 \times 112$	$7 \times 7$ conv, stride 2			
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2			
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$	$1 \times 1$ conv			
	$28 \times 28$	$2 \times 2$ average pool, stride 2			
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$	$1 \times 1$ conv			
	$14 \times 14$	$2 \times 2$ average pool, stride 2			
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	$14 \times 14$	$1 \times 1$ conv			
	$7 \times 7$	$2 \times 2$ average pool, stride 2			
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	$1 \times 1$	$7 \times 7$ global average pool			
		1000D fully-connected, softmax			

# MobileNet

## MobileNetV2: Inverted Residuals and Linear Bottlenecks

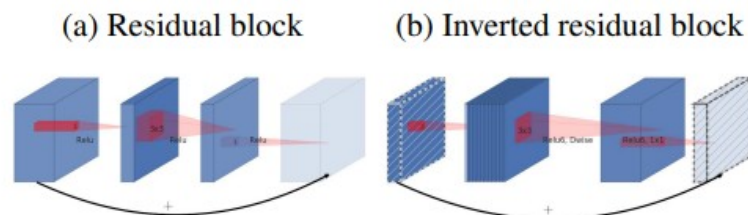
Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen  
Google Inc.

{sandler, howarda, menglong, azhmogin, lcchen}@google.com

- Architecture: 53 depthwise separable convolutions.
- Mobile/embedded vision applications.
- **Inverted residual blocks, linear bottlenecks.**
- Parameters: ~3.4M.
- Optimized for low-latency and low-memory devices.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d, ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1: *Bottleneck residual block* transforming from  $k$  to  $k'$  channels, with stride  $s$ , and expansion factor  $t$ .



Input	Operator	$t$	$c$	$n$	$s$
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-



# EfficientNet

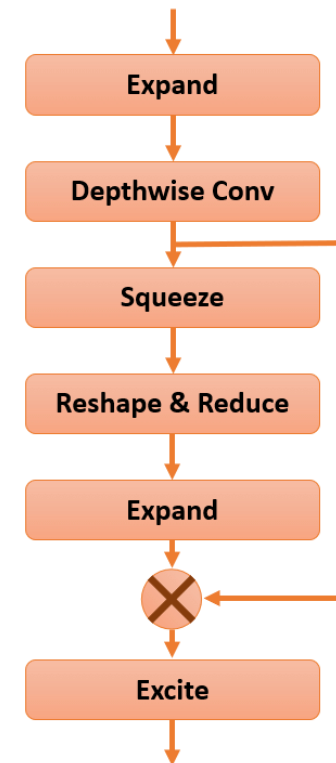
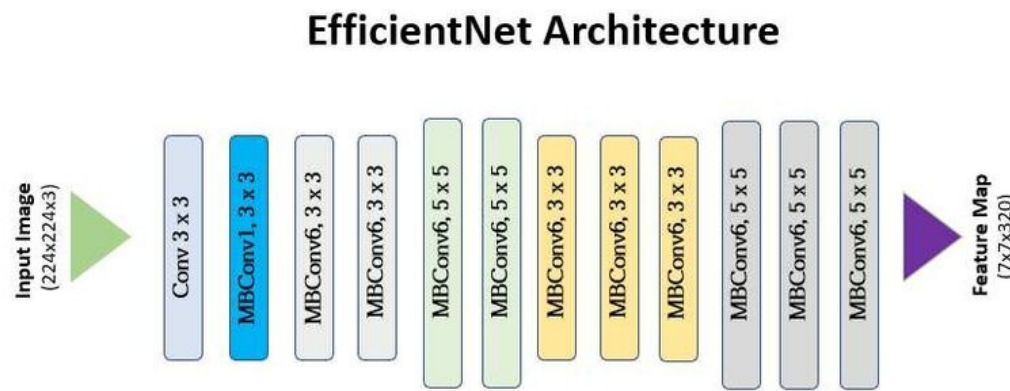
## EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan<sup>1</sup> Quoc V. Le<sup>1</sup>

<sup>1</sup>Google Research, Brain Team, Mountain View, CA. Correspondence to: Mingxing Tan <tanmingxing@google.com>.

Proceedings of the 36<sup>th</sup> International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

- Architecture: 16 ~81(\*) MBConv blocks (depthwise + pointwise convs).
- Dataset/Application: ImageNet classification.
- Compound **scaling of depth, width, and resolution.**
- Parameters: ~5.3M ~ 66M .
- State-of-the-art efficiency; high accuracy with minimal compute.



# Let's Code

[Lecture 07 - CNN Architecture](#)