Universidade de São Paulo

Escola Politécnica - Engenharia de Computação e Sistemas Digitais

Convolutional Networks

Prof. Artur Jordão

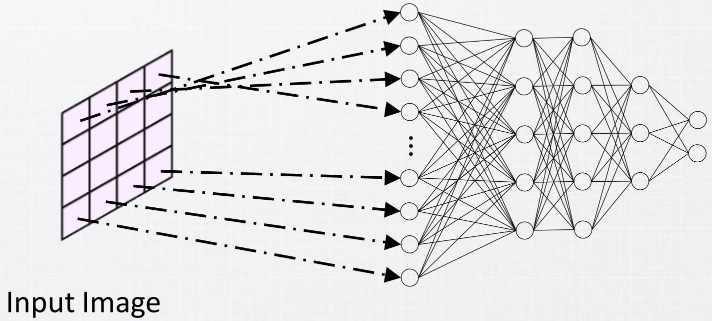
Introduction

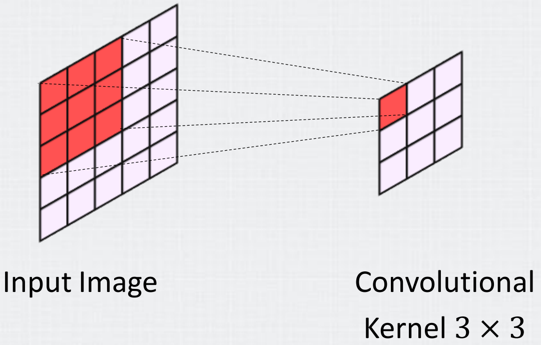
Definition

Introduction

- Convolutional networks are neural networks that use **convolution** in place of general matrix multiplication in at least one of their layers

- These networks are a specialized kind of neural network for processing data with a known grid-like topology. For example:
    - Time-series data (1D grid taking samples at regular time intervals)
    - Image data (2D grid of pixels)



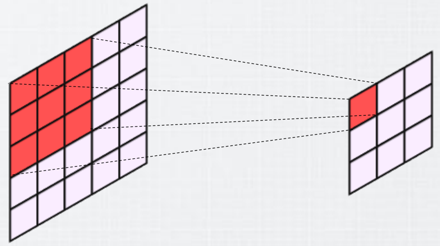


Motivation

Introduction

- Applying neurons directly to the image lacks the spatial location of the patterns

- Instead, we can organize neurons as elements within a window (filter/kernel)
  - Additionally, this strategy reduces the number of parameters
  - Weights are shared across “all” image locations



Architectural Details

Components

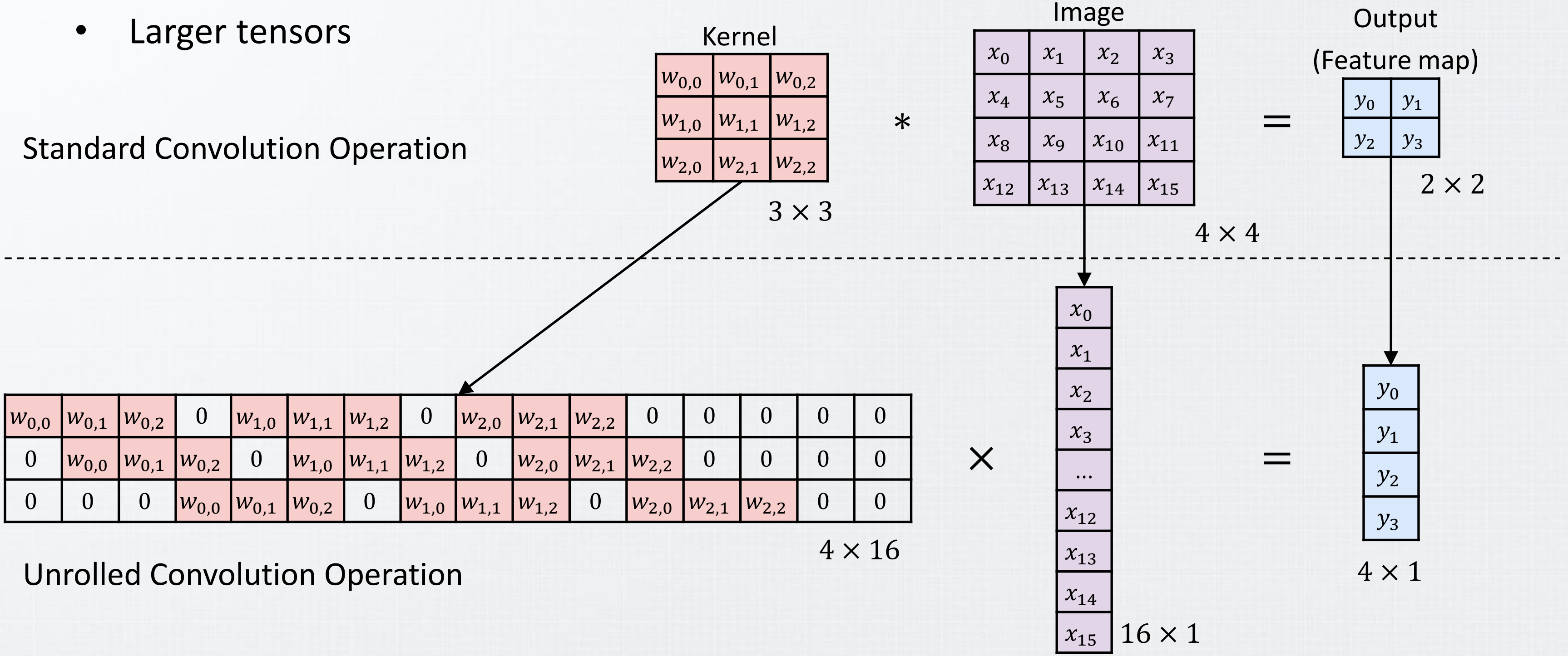
Architectural Details

- Convolutional Layers
- Downsampling Layers
  - Pooling Layers
- Classification Layers

Convolutional Layers

Introduction

- Unrolling convolutional operation
  - Transforms the problem into a (tensor) matrix multiplication
  - Larger tensors



Standard Convolution Operation

Kernel 3×3, Image 4×4, Output (Feature map) 2×2

$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} * \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} y_0 & y_1 \\ y_2 & y_3 \end{bmatrix}$$

Unrolled Convolution Operation

4×16, 16×1, 4×1

$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \dots \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

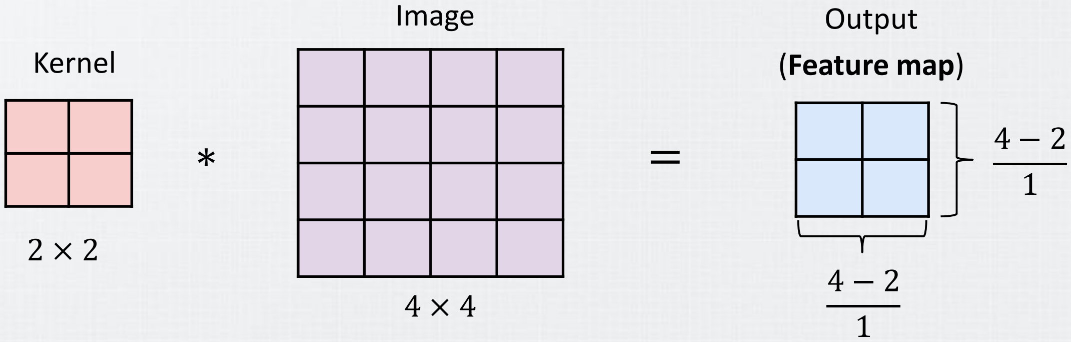
Convolutional Layers

Architectural Details

- Define W, H as the dimensions of the input image, and w, h as the dimensions of a convolutional kernel

- The convolution operation reduces the input in terms of
  - $\frac{W-w}{s_x} + 1, \frac{H-h}{s_y} + 1$, where s_i is the stride

- Most convolutional architectures employ 3×3 filters and a stride (s_x and s_y) of one



Kernel

2×2

$*$

Image

4×4

$=$

Output

(**Feature map**)

$\frac{4-2}{1}$

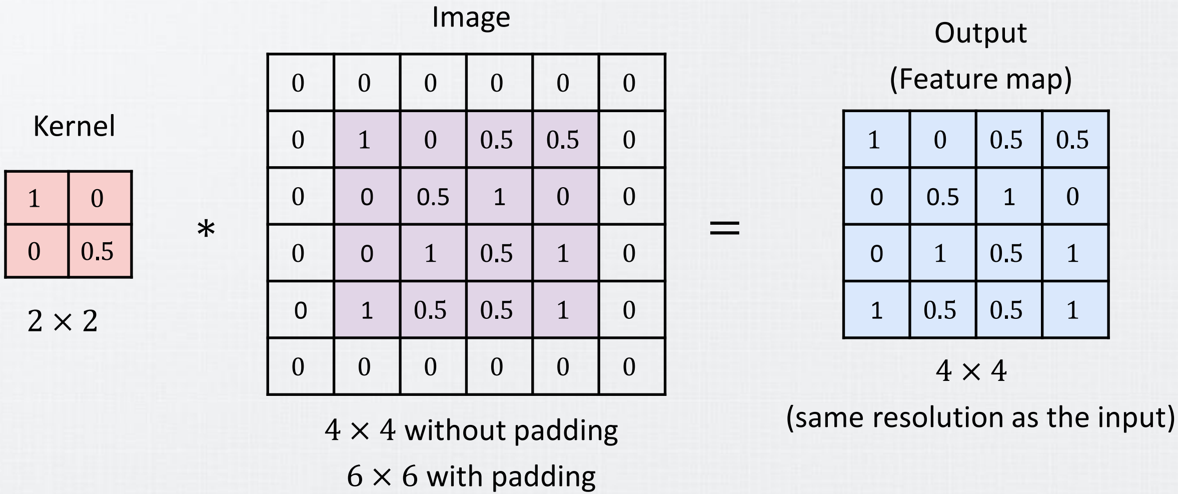
$\frac{4-2}{1}$

Convolutional Layers

Architectural Details

- Padding
  - It involves adding values on the input's edges to ensure that the input and output (after the convolution operation) **have the same spatial dimension**

- Most works employ **zero-padding**



Kernel

1	0
0	0.5

2×2

$*$

Image

0	0	0	0	0	0
0	1	0	0.5	0.5	0
0	0	0.5	1	0	0
0	0	1	0.5	1	0
0	1	0.5	0.5	1	0
0	0	0	0	0	0

4×4 without padding

6×6 with padding

$=$

Output

(Feature map)

1	0	0.5	0.5
0	0.5	1	0
0	1	0.5	1
1	0.5	0.5	1

4×4

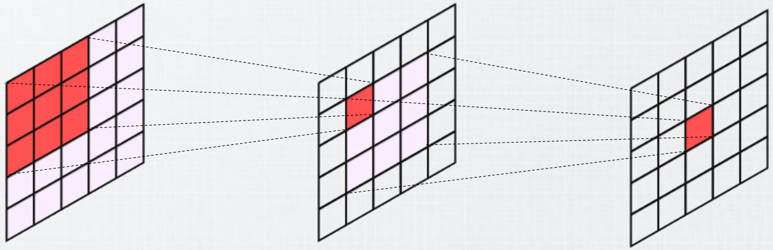
(same resolution as the input)

Convolutional Layers

Architectural Details

- Receptive field
  - Refers to all the elements (from all the previous layers) that may affect the calculation of x (a position in the feature map) during the forward propagation

- The local operation of the convolution kernel makes the model focus too much on local representations (e.g., texture) (Geirhos et al., 2019; Guo et al., 2023)



Layer i Layer $i+1$ Layer $i+2$

Geirhos et al. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. International Conference on Learning Representations (ICLR) 2019

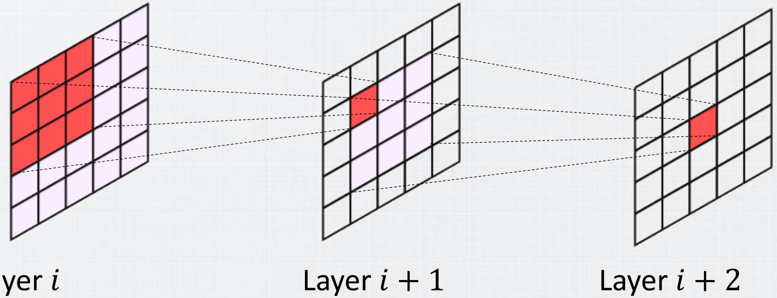
Guo et al. ALOFT: A Lightweight MLP-like Architecture with Dynamic Low-frequency Transform for Domain Generalization. Conference on Computer Vision and Pattern Recognition (CVPR), 2023

Downsampling Layers

Architectural Details

- An important role in learning new representations is to reduce the spatial dimensions of feature maps (Greff et al., 2017)
  - Downsampling the input increases the receptive field

- There are two distinct ways of achieving this reduction
  - Pooling operations
  - Convolutional layers with stride 2 × 2



Greff et al. *Highway and residual networks learn unrolled iterative estimation*. International Conference on Learning Representations (ICLR), 2017

Downsampling Layers

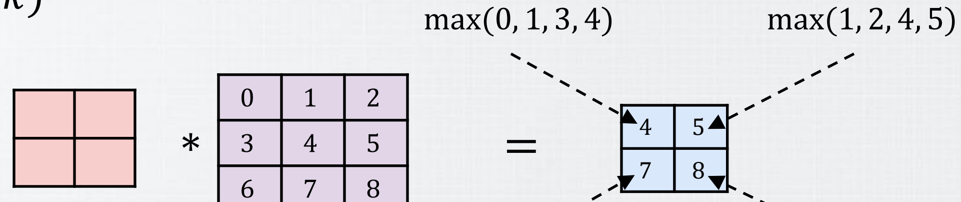
Architectural Details

- Pooling layers (also known as pooling operators or functions) are a fixed-shape window that slides over all regions in the input, computing a single output for each location traversed by the window
    - Pooling **summarizes the responses over a whole neighborhood**
- Pooling are deterministic operations
    - It contains kernel size and stride but has **no (learnable) parameters**
- Pooling helps the representation become approximately invariant to small translations of the input
    - Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change

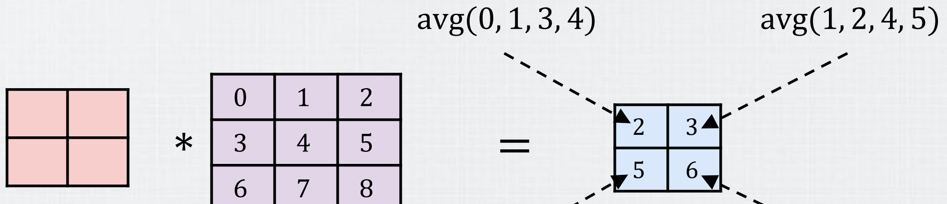
Downsampling Layers

Architectural Details

- Common pooling operations
  - Max-pooling $(k \times k)$
  - Average-pooling $(k \times k)$



Max-Pooling
2×2

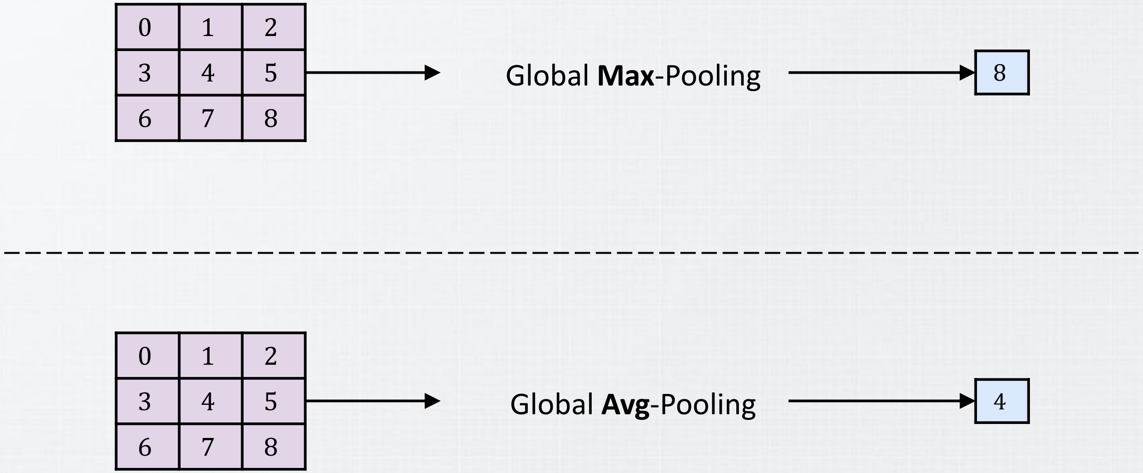


Avg-Pooling
2×2

Downsampling Layers

Architectural Details

- Global-pooling operations
  - Global max-pooling
  - Global average-pooling



0	1	2
3	4	5
6	7	8

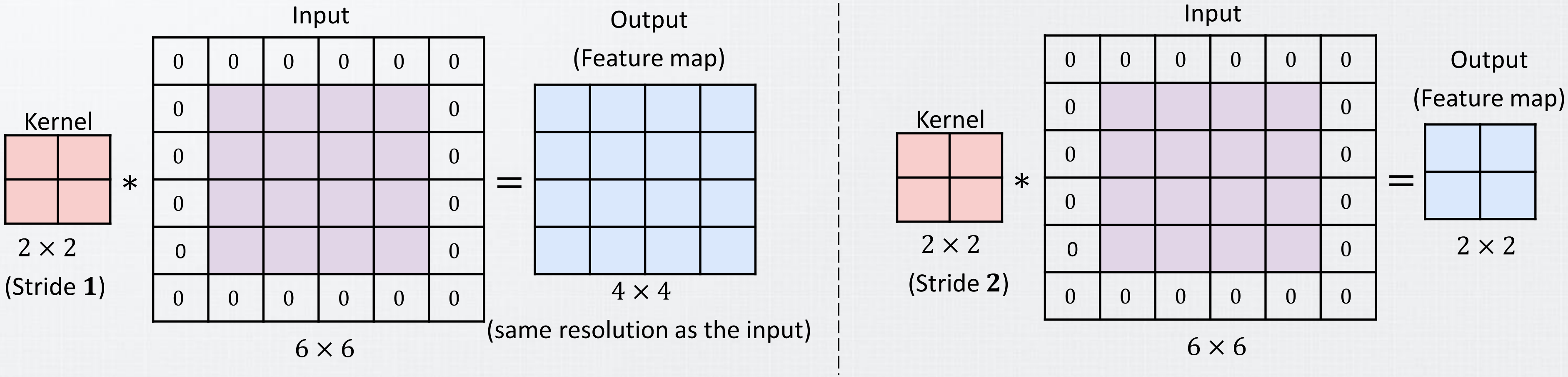
0	1	2
3	4	5
6	7	8

Downsampling Layers

Architectural Details

- Convolution with stride $(s_i) = 2$
    - **Learnable parameters**

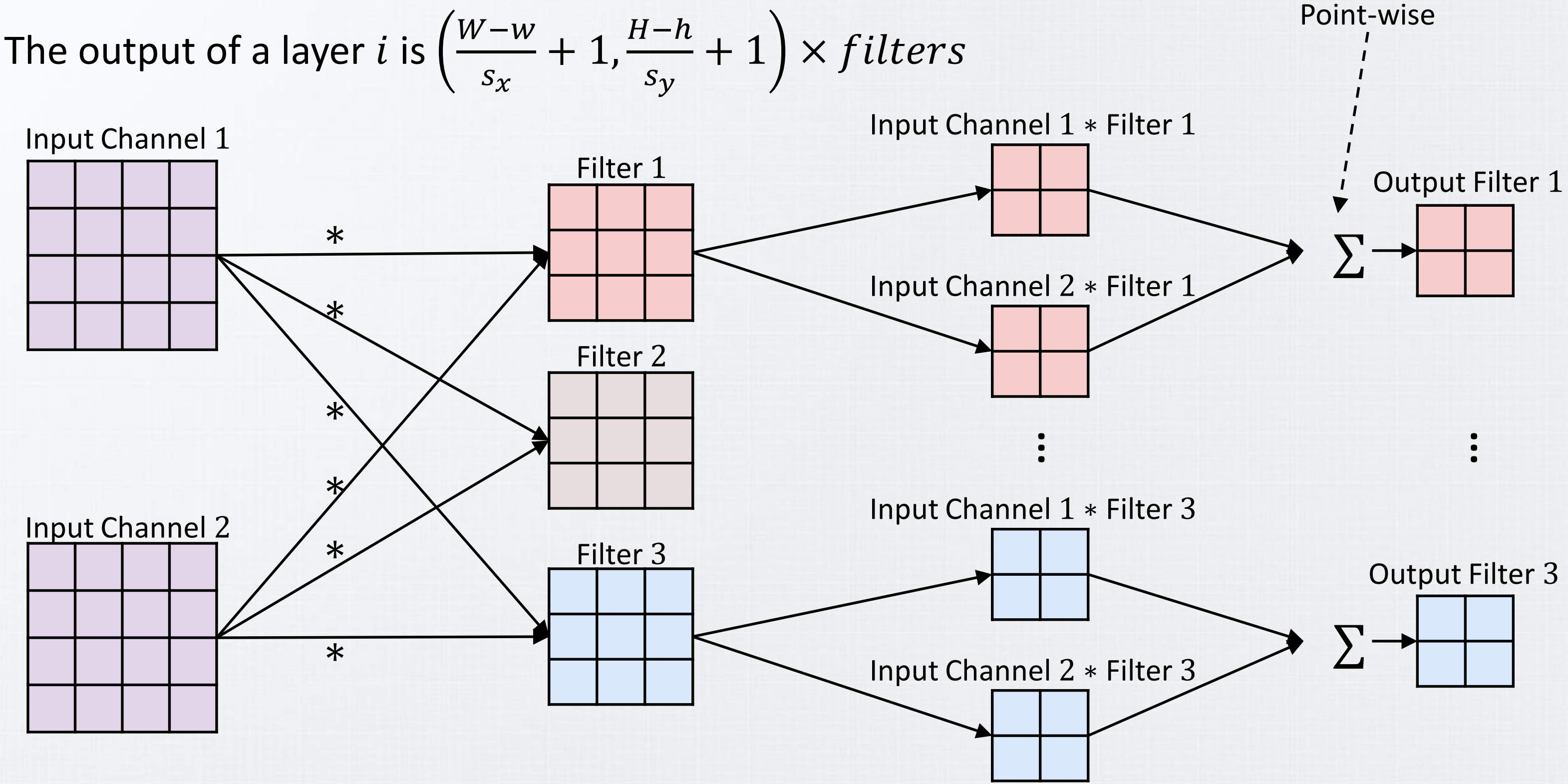
- Remember that the output of a convolution is $\left(\frac{W-w}{s_x} + 1, \frac{H-h}{s_y} + 1\right)$
    - By setting the stride higher than 1, we obtain downsampled feature maps



Multiple Filters

Architectural Details

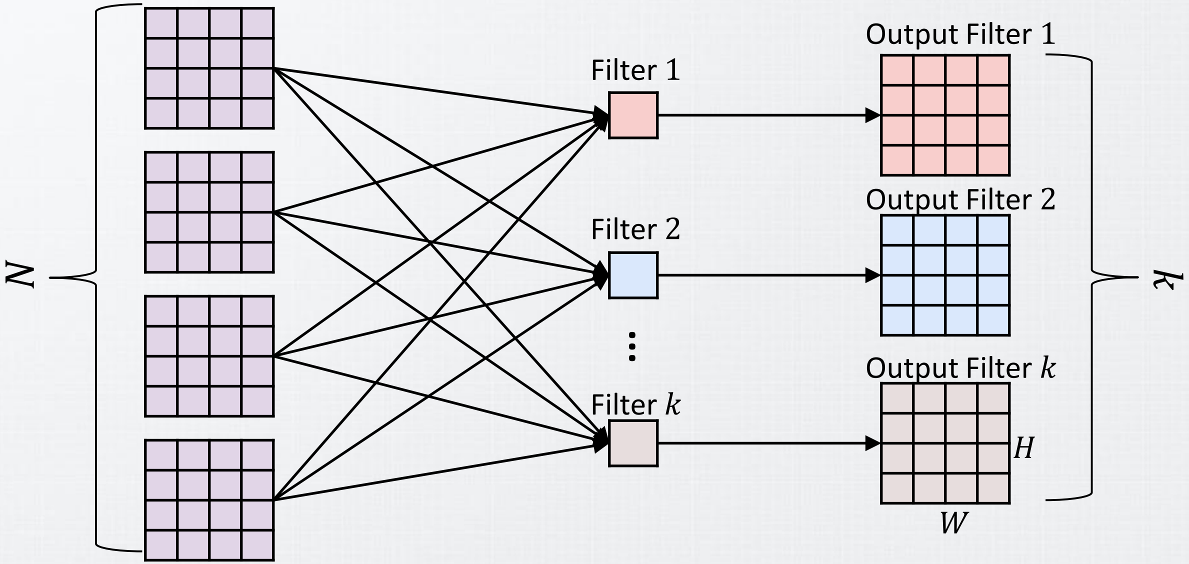
- Each filter from a layer i convolves all input dimensions
  - The output of a layer i is $\left(\frac{W-w}{s_x}+1, \frac{H-h}{s_y}+1\right) \times filters$



1×1 Convolutional Layers

Architectural Details

- Convolutions with kernel of size 1
  - Such configuration preserves the spatial resolution of the input
  - It projects an input of size (W, H, N) into (W, H, k), where k is the number of filters
  - Therefore, this type of convolution enables to reduce the dimension of channels



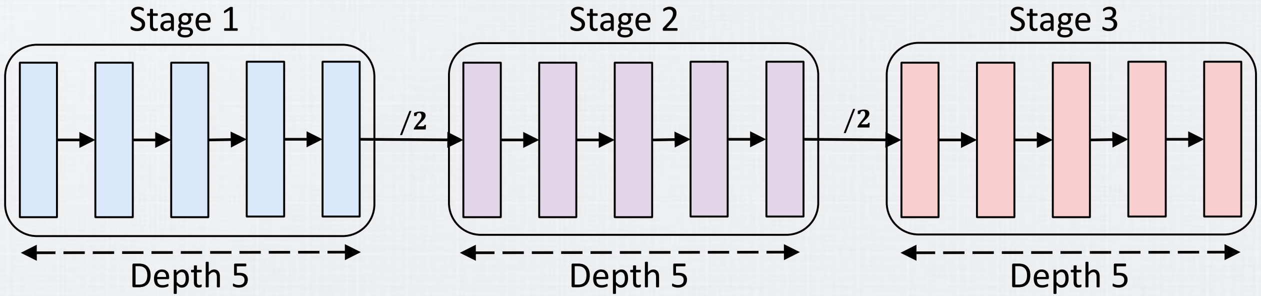
Stages

Architectural Details

- A Stage (or module) is a group of layers that operate on representations (feature maps) at the **same resolution**
  - From stage i to $i + 1$, the common strategy is to reduce the spatial resolution. For this purpose, we employ a downsampling layer (/2)

- The **depth (number of convolutional layers)** of these stages is defined either uniformly (e.g., ResNet20–110) or empirically (e.g., ResNet50–101)
  - A common practice is to double the number of filters as we decrease the resolution

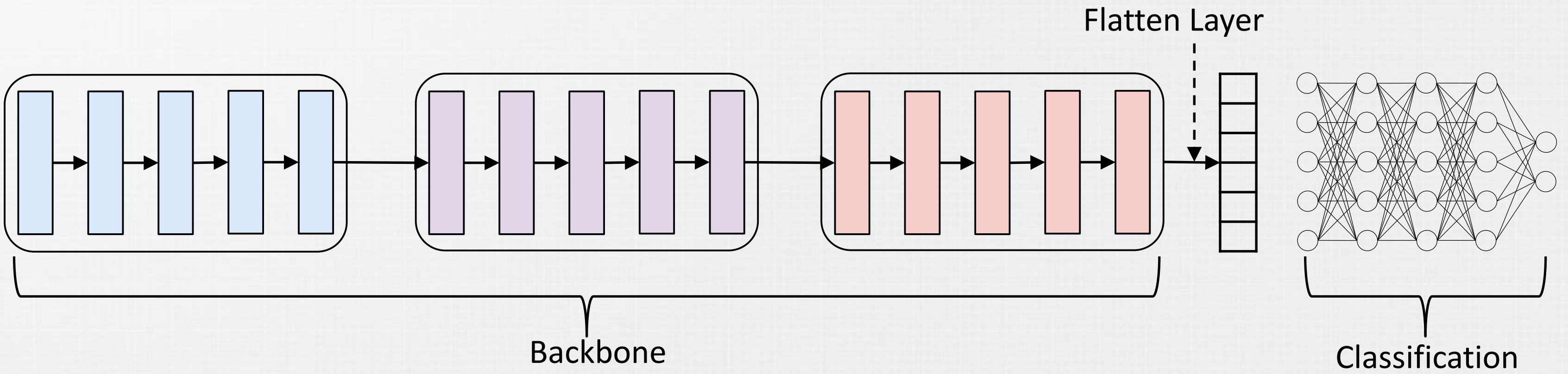
Stage	Resolution
1	32×32
2	16×16
3	8×8



Classification Layers

Architectural Details

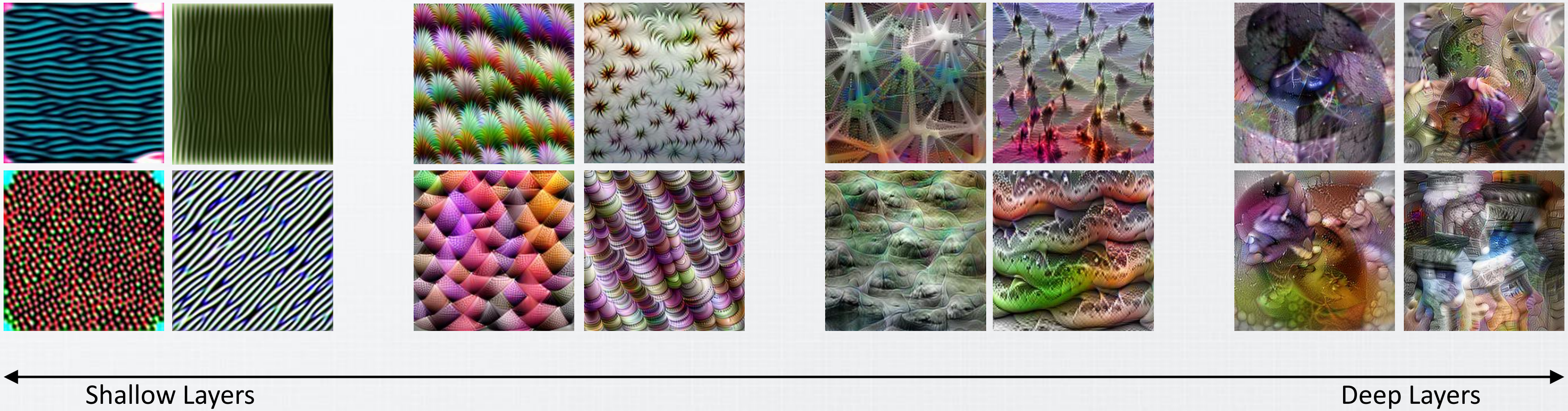
- Fully connected (FC) layers
    - MLPs
- Modern architectures often incorporate **global average pooling** before the classification layer
    - It significantly reduces the number of parameters
    - It helps to hand inputs of varying size



Representation Learned Across Layers

Architectural Details

- Neuron representations from https://microscope.openai.com/models
  - ResNet50



Popular Architectures

Architectural Details

- VGG (Depth 16 and 19)
- ResNet (Depth 50, 101 and 152)
- MobileNet
- NASNet
- EfficientNet
- All these architectures employ 3×3 convolutional kernels

Historical Trends

Architectural Details

- CNNs have been the de-facto standard in computer vision since the AlexNet model surpassed prevailing approaches based on hand-crafted image features (Krizhevsky et al., 2012)

- Simonyan and Zisserman (2015) demonstrated that one can train state-of-the-art models using only convolutions with small $\mathbf{3 \times 3}$ **kernels**

- He et al. (2016) introduced skip connections, which enable the training of ultra-deep neural networks and further improve performance
    - Since then, many advances in the field of deep learning have been made using skip connections

Krizhevsky et al. *ImageNet classification with deep convolutional neural networks*. Neural Information Processing Systems (NeurIPS), 2012

Simonyan and Zisserman. *Very deep convolutional networks for large-scale image recognition*. International Conference on Learning Representations (ICLR), 2015

He et al. *Deep residual learning for image recognition*. Conference on Computer Vision and Pattern Recognition (CVPR), 2016

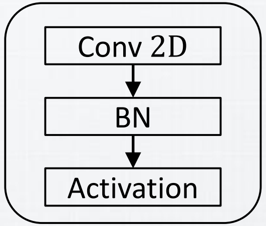
Building Blocks

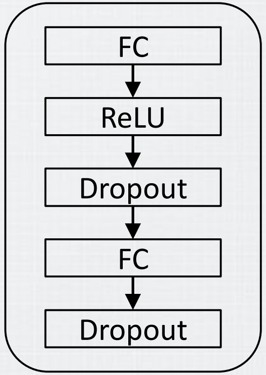
Introduction

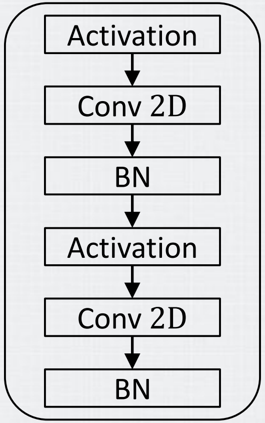
Building Blocks

- Modern neural network architectures often adopt a **modular approach**
    - Design a layer (building blocks) and replicate it to build the model

- Building blocks (or modules) are combinations of different components (sets of layers)
    - The input to the ith building block is the output of the $i - 1th$ block







Scalability and Complex Architectures

Building Blocks

- Building Blocks are fundamental for designing complex architectures
  - Once created, we can repeat the building block to compose the final architecture
- We can create **wider** models by increasing the embedding dimension or the number of channels in each block
- We can create **deeper** models by stacking more layers/blocks
- Isotropic architectures
  - Each block maintains a consistent and uniform layerwise design
- Hierarchical architectures
  - Consists of **stages** with varying scales and embedding dimensions

Scalability and Complex Architectures

Building Blocks

- Building blocks became popular after VGG and ResNet architectures

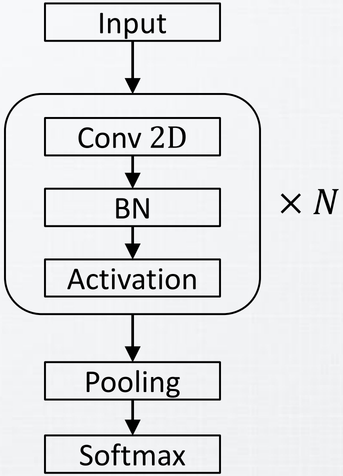
Layer Name	Output Size	18-Layer	50-Layer	152-Layer
Conv1	112×112	$7 \times 7, 64$, **Stride 2**		
Conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
Conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
Conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	Average Pooling, $1000 - d$ FC, Softmax		

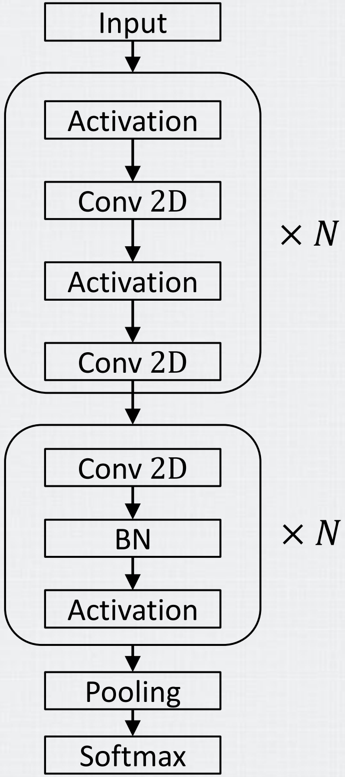
He et al. *Deep Residual Learning for Image Recognition*. Computer Vision and Pattern Recognition (CVPR), 2016

Final Architecture

Building Blocks

- Building blocks facilitate the construction and representation (i.e., illustration) of modern architectures





Residual Networks (Skip Connection)

Motivation

Residual Networks

- Practical experience in deep learning suggests that deeper models can significantly improve their predictive performance
  - Unfortunately, as the network becomes **deeper**, training becomes **harder**

- The degradation problem (He et al., 2016)
  - As the network **depth increases**, **accuracy saturates** and then **rapidly degrades**
  - Unexpectedly, such degradation is not caused by **overfitting**

	Plain	Residual
18 layers	27.94	27.88
34 layers	28.54	25.03

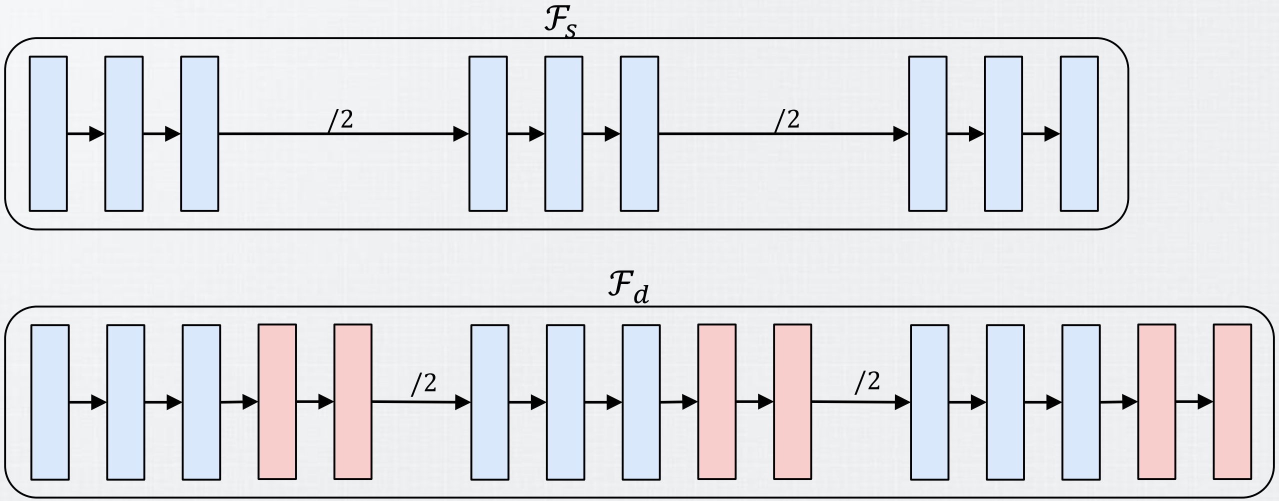
He et al. *Deep Residual Learning for Image Recognition*. Computer Vision and Pattern Recognition (CVPR), 2016

Problem Formulation

Residual Networks

- Consider a shallow architecture $\mathcal{F}_s$ and its deeper counterpart $\mathcal{F}_d$
  - $\mathcal{F}_d$ is essentially $\mathcal{F}_s$ with more layers (building blocks)
  - Informally: $\mathcal{F}_s \subset \mathcal{F}_d$

- The deeper model should not produce higher training error than its shallower counterpart



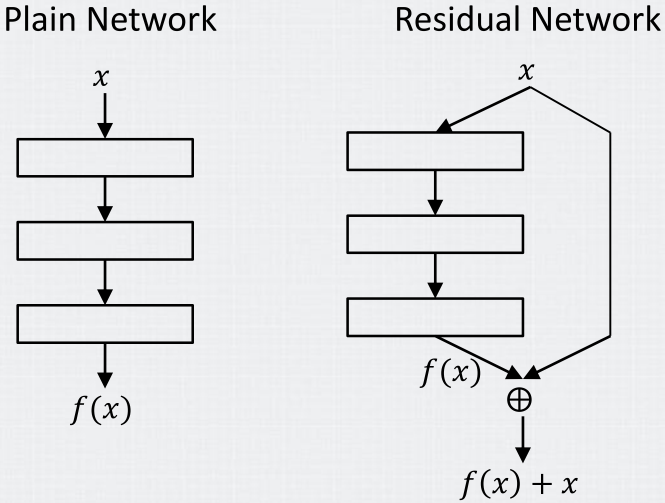
Overview

Residual Networks

- The idea consists of connecting layer i with a subsequent layer $i + j, j > 1$
  - This connection (skip-connection) is done by adding (element-wise) the feature maps of layer i and $i + j$

- Layers in Plain network
  - Receives x and outputs $f(x)$

- Layers in Residual network
  - Receives x and outputs $f(x) + \boldsymbol{x}$



Theoretical Issues

Residual Networks

- Two groups of believers
  - Skip connections to avoid the vanish gradient problem
  - Skip connections extend beyond addressing the vanishing gradient problem (I, Prof. Artur, belong to this category)

> We argue that this optimization **difficulty is unlikely to be caused by vanishing gradients**. These plain networks are trained with BN [16], which ensures forward propagated signals to have non-zero variances. **We also verify that the backward propagated gradients exhibit healthy norms with BN** […]

Source: He et al., 2016

> [...] In this paper, we explore the interaction between depth and the loss geometry. We first establish that gradient explosion or **vanishing is not responsible for the slowing down of training**, as is **commonly believed**.
>
> [...] The most prevalent explanation for why very deep networks are hard to train is that the gradient explodes or vanishes as the number of layers increase [5]; **this explanation has been infrequently challenged** [...]
>
> [...] Firstly, **there is no exponential increase or decrease in gradient norms** (i.e., we would see vastly different gradient norm scales), as hypothesised in gradient explosion explanations. Secondly, **residual connections do not consistently increase or decrease the gradient norms. In Figure 1, 49.4% of variables have lower gradient norm in residual networks (in comparison to a baseline of non-residual networks), making the exploding/vanishing gradient explanation untenable in this case.**

Source: Ghorbani et al., 2019

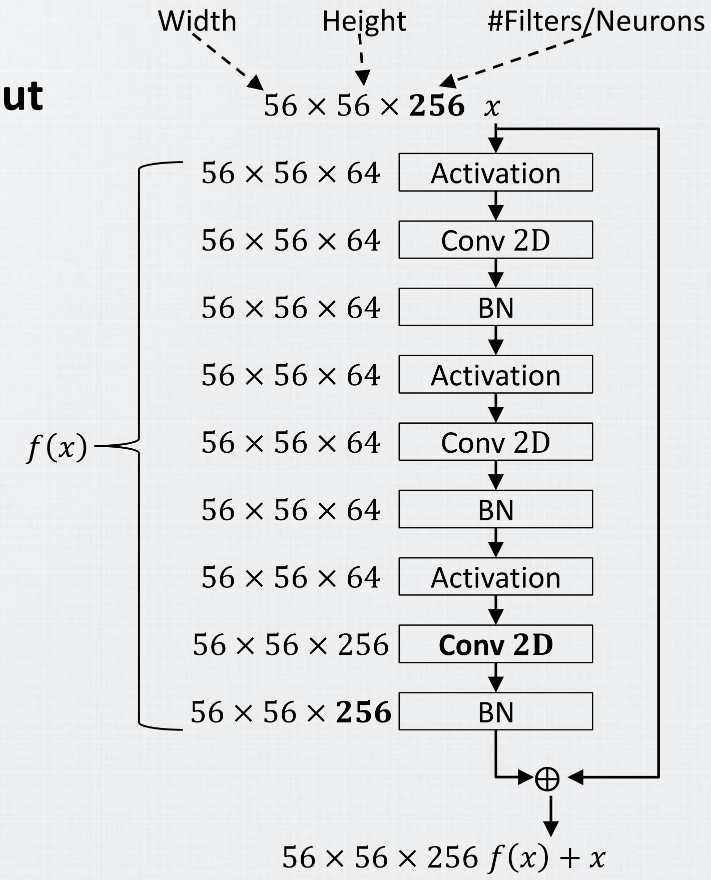
He et al. *Deep Residual Learning for Image Recognition*. Computer Vision and Pattern Recognition (CVPR), 2016

Ghorbani et al. *The Effect of Network Depth on the Optimization Landscape*. International Conference on Machine Learning (ICML), 2019

Technical Issues

Residual Networks

- Within a building block the **input and output must match**



Width

Height

#Filters/Neurons

$56 \times 56 \times \mathbf{256}$ x

$f(x)$

$56 \times 56 \times 64$ Activation

$56 \times 56 \times 64$ Conv 2D

$56 \times 56 \times 64$ BN

$56 \times 56 \times 64$ Activation

$56 \times 56 \times 64$ Conv 2D

$56 \times 56 \times 64$ BN

$56 \times 56 \times 64$ Activation

$56 \times 56 \times 256$ **Conv 2D**

$56 \times 56 \times \mathbf{256}$ BN

$\oplus$

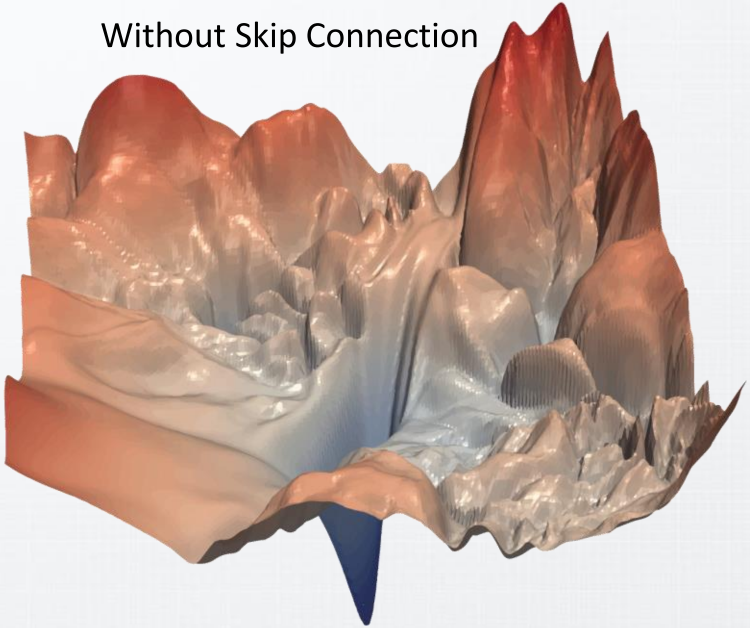
$56 \times 56 \times 256$ $f(x) + x$

Modern Architectures

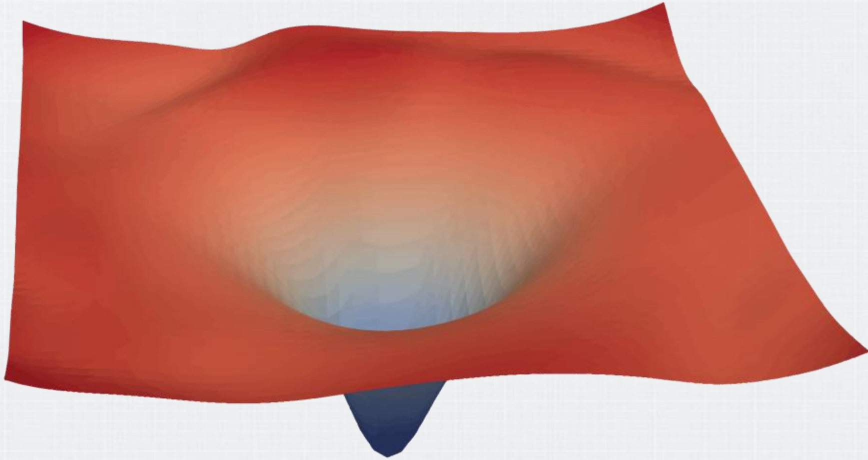
Residual Networks

- Due to the success and simplicity of residual networks, modern architectures are predominantly based on residual learning

Without Skip Connection



With Skip Connection



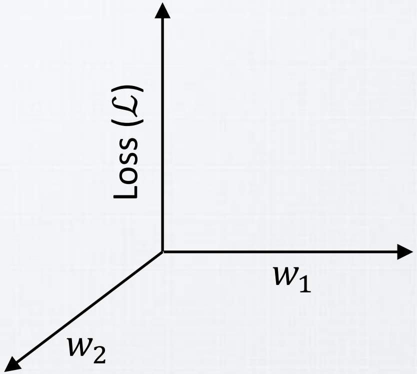
Li et al. *Visualizing the Loss Landscape of Neural Nets*. Neural Information Processing Systems (NeurIPS) 2018

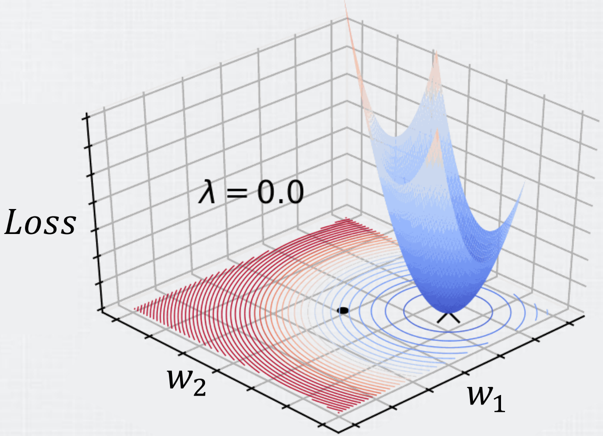
Loss LandScape

Modern Architectures

Loss LandScape

- Due to the high number of parameters (i.e., dimensions in space), visualizing the loss landscape of the parameters in overparameterized models is hard





Li et al. *Visualizing the Loss Landscape of Neural Nets*. Neural Information Processing Systems (NeurIPS) 2018

Definitions

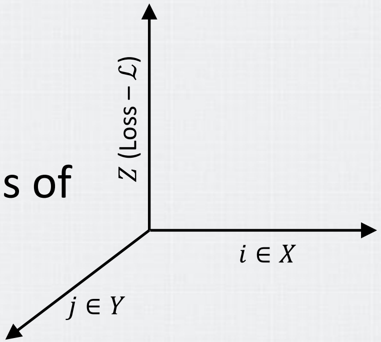
Loss Landscape

- Li et al. (2018) proposed a novel visualization scheme
    - The method samples two Gaussian random vectors (δ and β) that normalizes the filter norms in the neural network
    - Then compute the loss across different combinations of the two filter-normalized Gaussian random vectors from the minimizer

- Define $\delta \sim \mathcal{N}(0,1)$ and $\beta \sim \mathcal{N}(0,1)$

- Consider a equally spaced $3D$ grid (X, Y, Z)

- The method by Li et al. (2018) estimates the loss landscape in terms of
    - $Z_{i,j} = \mathcal{L}(\theta^* + i * \delta + j * \beta)$



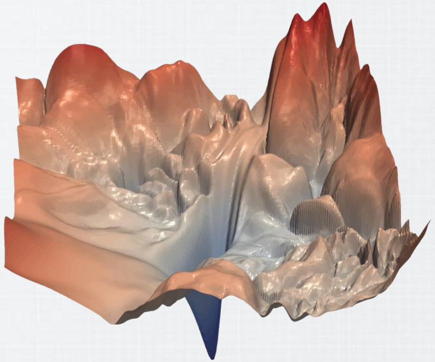
Li et al. *Visualizing the Loss Landscape of Neural Nets*. Neural Information Processing Systems (NeurIPS) 2018

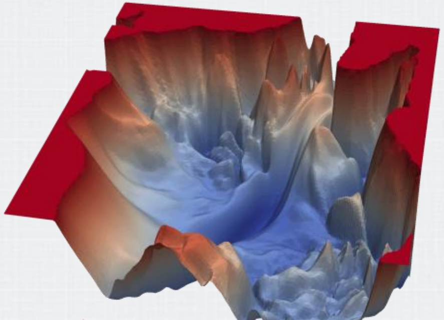
Loss Landscape

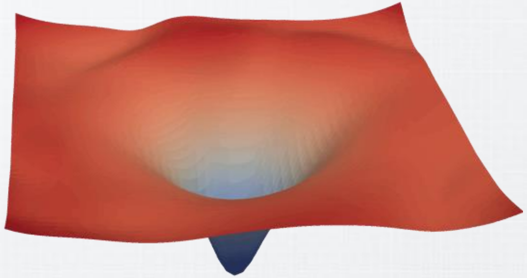
Loss Landscape

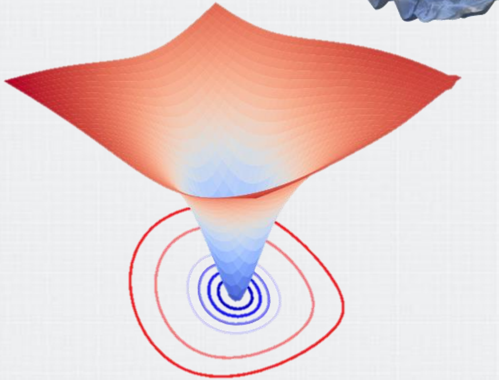
- http://www.telesens.co/loss-landscape-viz/viewer.html

- https://losslandscape.com/









Li et al. *Visualizing the Loss Landscape of Neural Nets*. In Neural Information Processing Systems (NeurIPS), 2018

Bibliography

Bibliography

- He et al. *Deep Residual Learning for Image Recognition*. Computer Vision and Pattern Recognition (CVPR), 2016

- Geirhos et al. *Imagenet-trained CNNs are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness*. International Conference on Learning Representations (ICLR) 2019

- Veit et al. *Residual Networks Behave Like Ensembles of Relatively Shallow Networks*. In Neural Information Processing Systems (NeurIPS), 2016

- Li et al. *Visualizing the Loss Landscape of Neural Nets*. In Neural Information Processing Systems (NeurIPS), 2018