Advanced Computer Vision

Naeemullah Khan

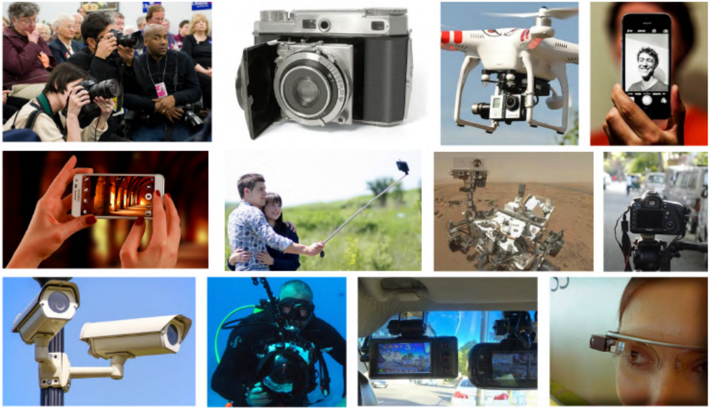
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

June 15, 2023

Building artificial systems that process, perceive, and reason about visual data

Computer Vision is Everywhere



Some Applications

أكاديمية كاوست
KAUST ACADEMY

Image Classification

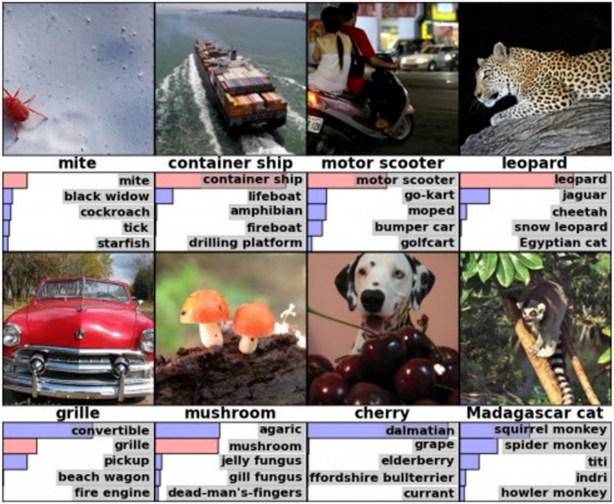


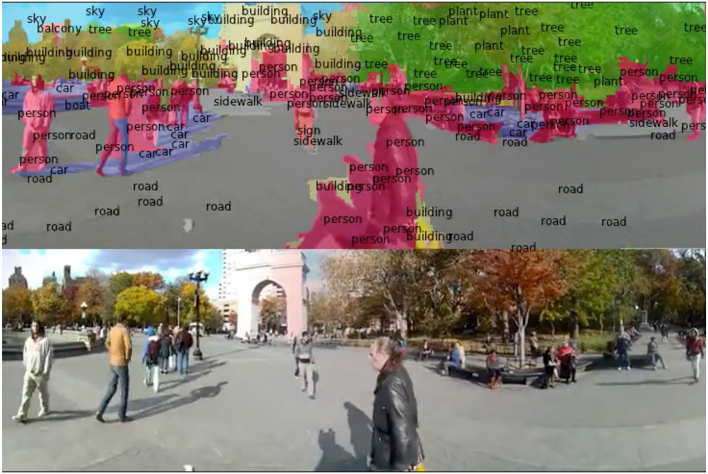
Image Retrieval

Object Detection

Ren, He, Girshick, and Sun, 2015

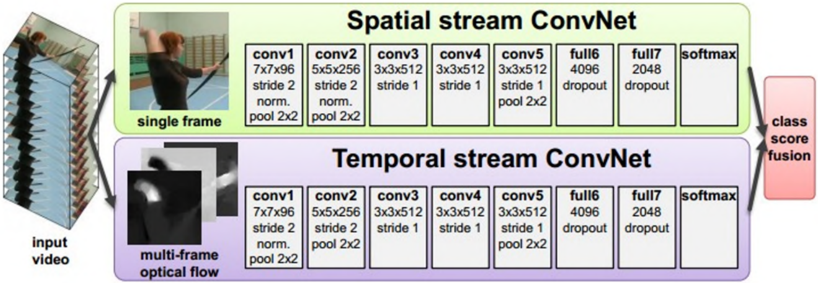
Some Applications (cont.)

Image Segmentation



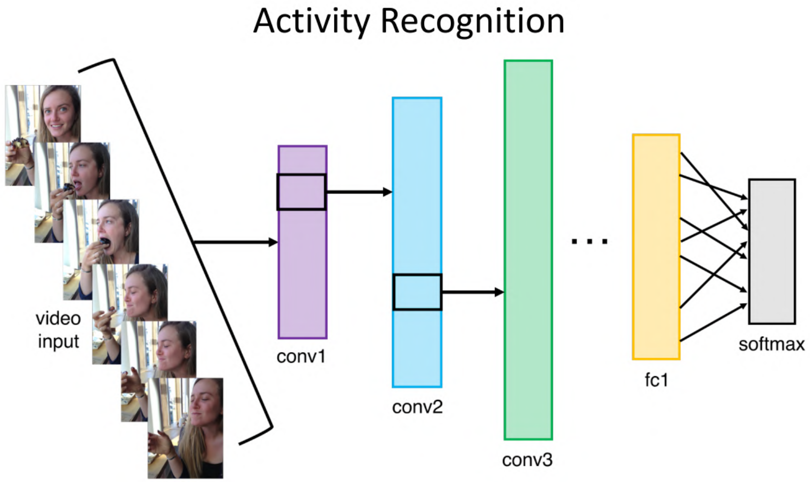
Fabaret et al, 2012

Video Classification



Simonyan et al, 2014

Some Applications (cont.)



Some Applications (cont.)

Pose Recognition (Toshev and Szegedy, 2014)

Some Applications (cont.)

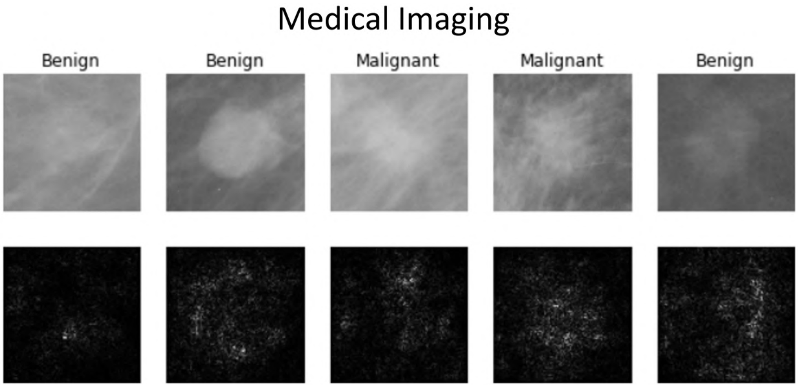


Image Captioning

Vinyals et al, 2015

Karpathy and Fei-Fei, 2015

A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Image Generation

"Teddy bears working on new AI research underwater with 1990s technology"

DALL-E 2

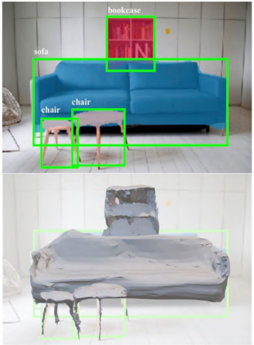
Some Applications (cont.)

Style Transfer

3D Vision

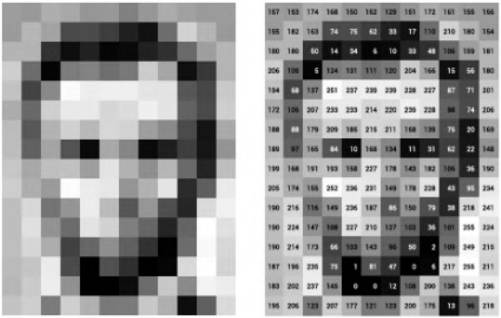
Choy et al., 3D-R2N2: Recurrent Reconstruction Neural Network (2016)

Zhou et al., 3D Shape Generation and Completion through Point-Voxel Diffusion (2021)



Gkioxari et al., "Mesh R-CNN", ICCV 2019

- Images are represented as Matrices with elements in $[0, 255]$
- Grayscale images have one channel while RGB images have 3 channels



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
206	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

[0] https://www.v7labs.com/blog/image-recognition-guide

Fully-Connected Neural Networks

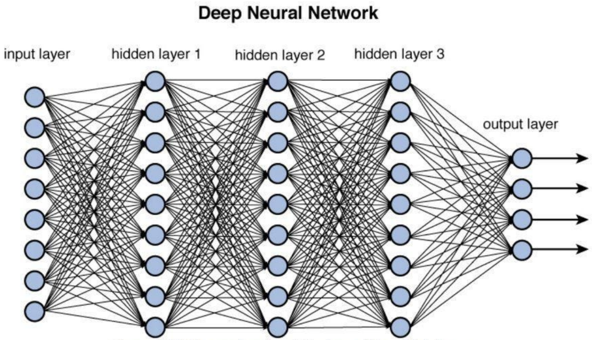
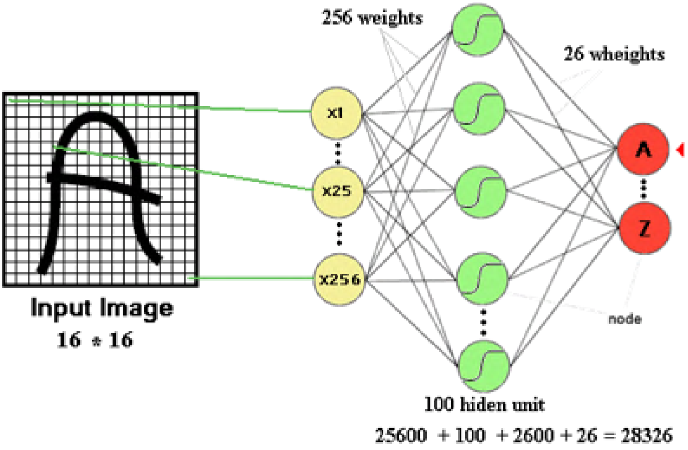


Figure 12.2 Deep network architecture with multiple layers.

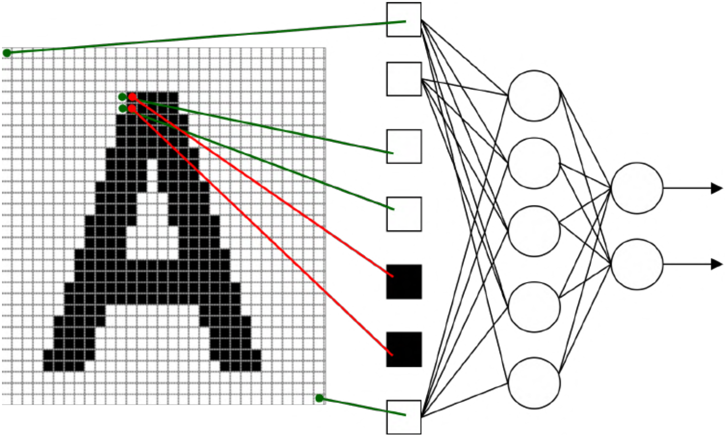
$$z = W_1 x_1 + W_2 x_2 + \cdots + W_n x_n + b$$

[0] https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

- The number of trainable parameters becomes extremely large

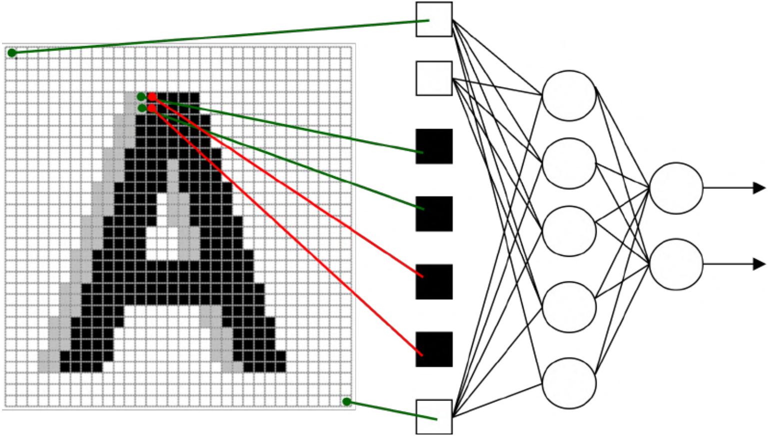


- Little or no invariance to shifting, scaling, and other forms of distortion



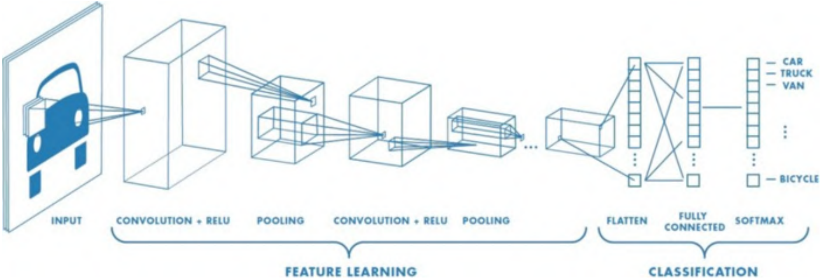
Drawbacks of Fully-Connected Neural Networks (cont.)

- Little or no invariance to shifting, scaling, and other forms of distortion



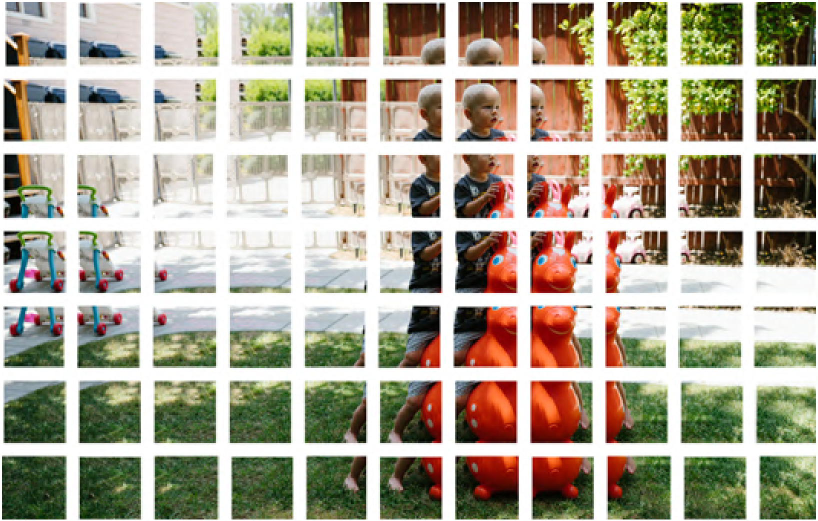
Drawbacks of Fully-Connected Neural Networks (cont.)

- The topology of the input data is completely ignored
- For a 32×32 image, we have
  - Black and white patterns: $2^{32*32} = 2^{1024}$
  - Grayscale patterns: $256^{32*32} = 256^{1024}$

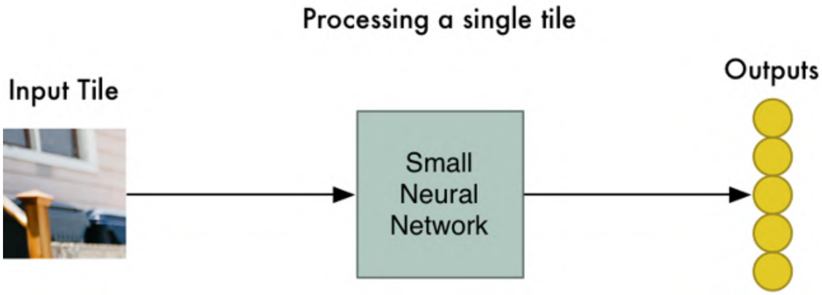


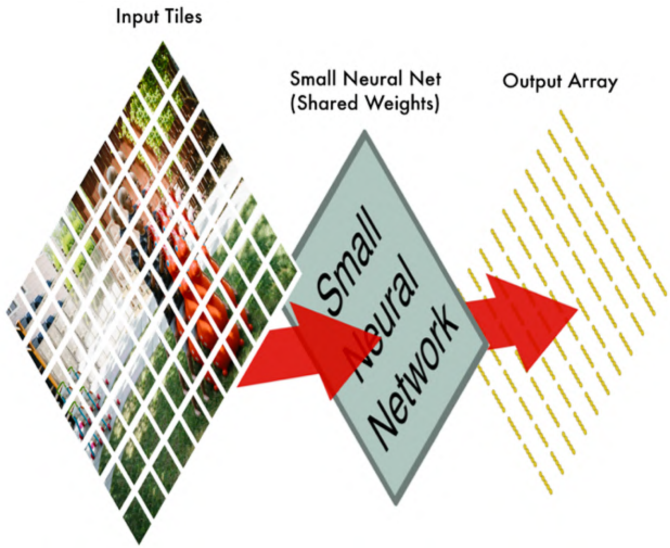
$$z = W * x_{i,j} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} W_{ab} x_{(i+a)(j+b)}$$

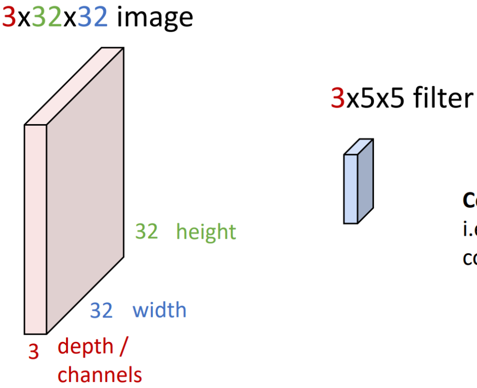
How Convolution Works?



How Convolution Works? (cont.)

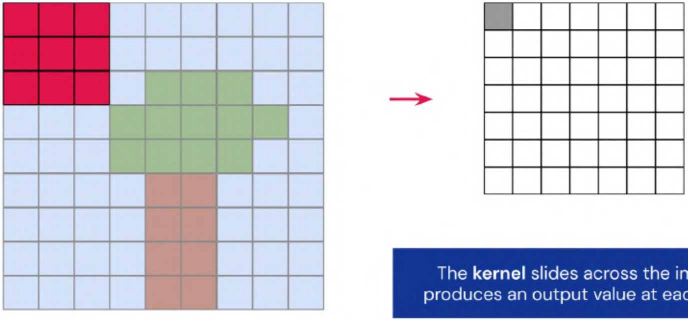



Input Tiles
Small Neural Net
(Shared Weights)
Output Array
Small
Neural
Network

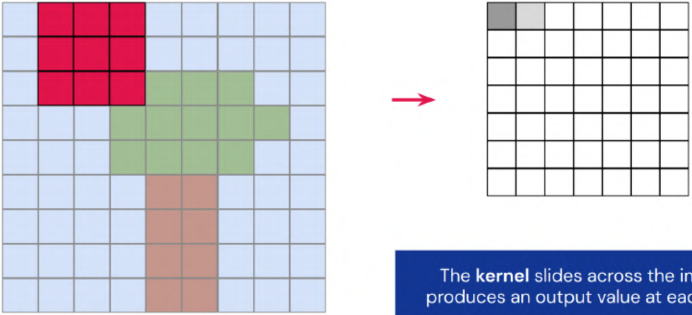


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

How Convolution Works? (cont.)

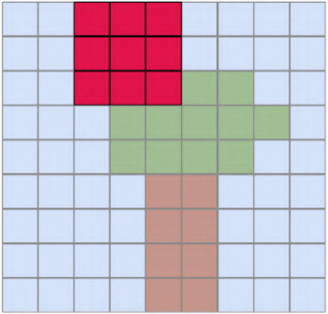


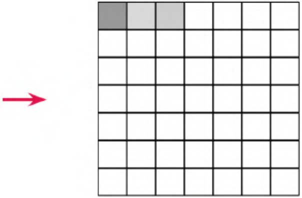
The **kernel** slides across the image and produces an output value at each position



The **kernel** slides across the image and produces an output value at each position

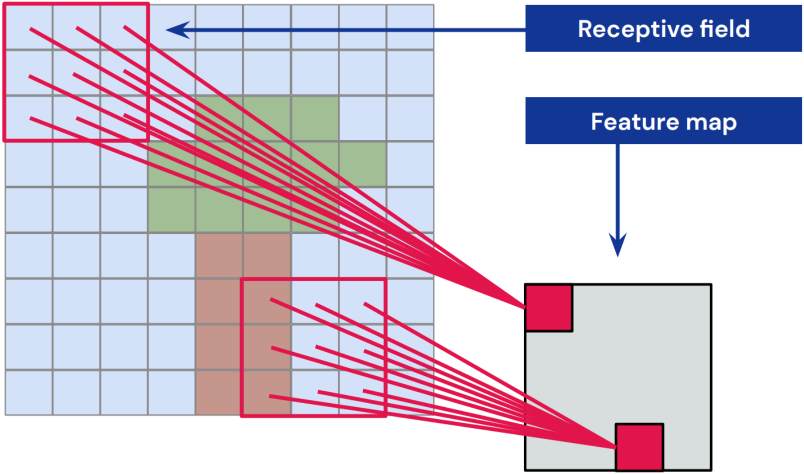
How Convolution Works? (cont.)

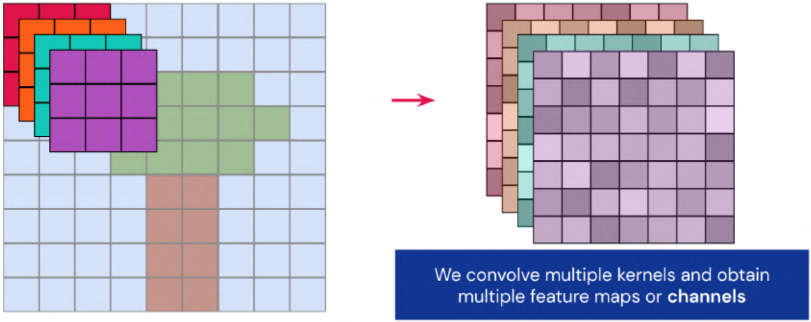




The **kernel** slides across the image and produces an output value at each position

How Convolution Works? (cont.)





We convolve multiple kernels and obtain multiple feature maps or **channels**

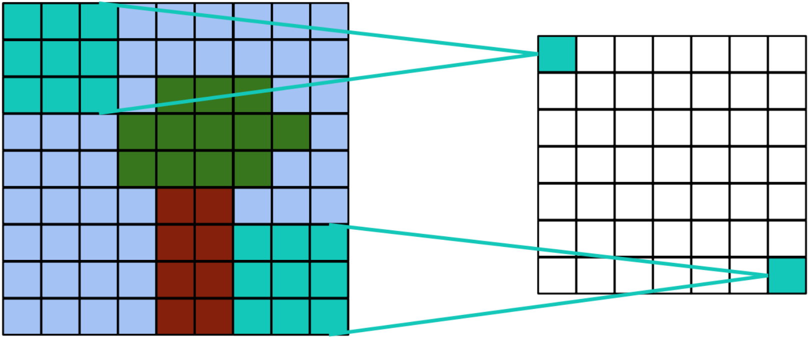
How Convolution Works? (cont.)

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

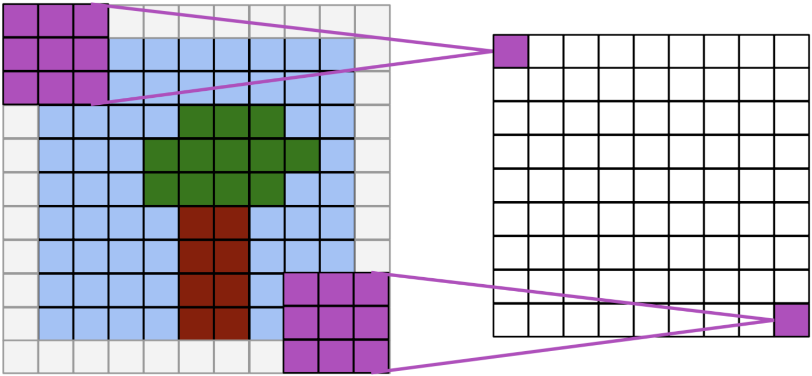
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Padding

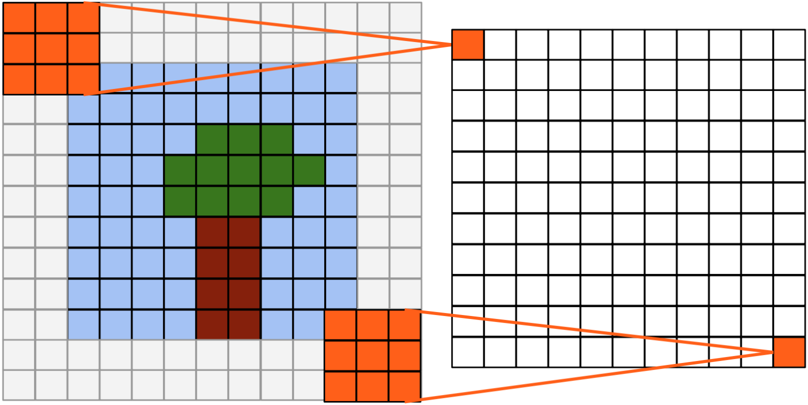
- Applying Convolution as such reduces the size of the borders.
- Sometimes this is not desirable.
- We can pad the border with zeros.



- Same Convolution: Output is the same size as input

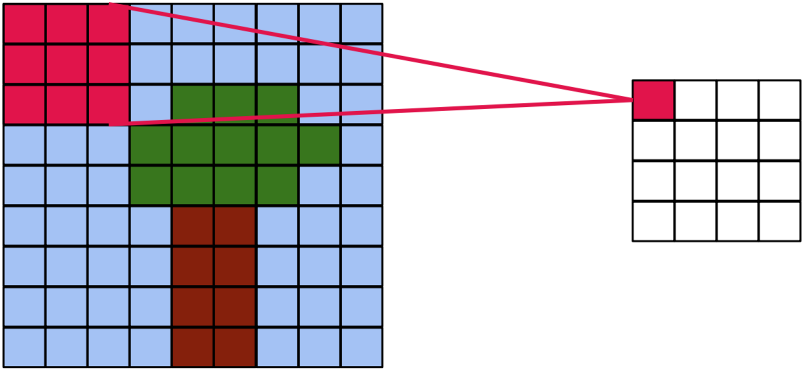


- Full Convolution: output size = input size + kernel size - 1

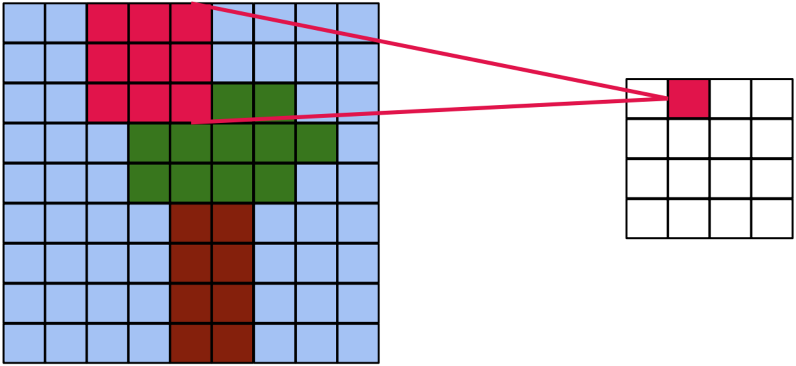


Strided Convolution

- Kernel slides along the image with a step > 1

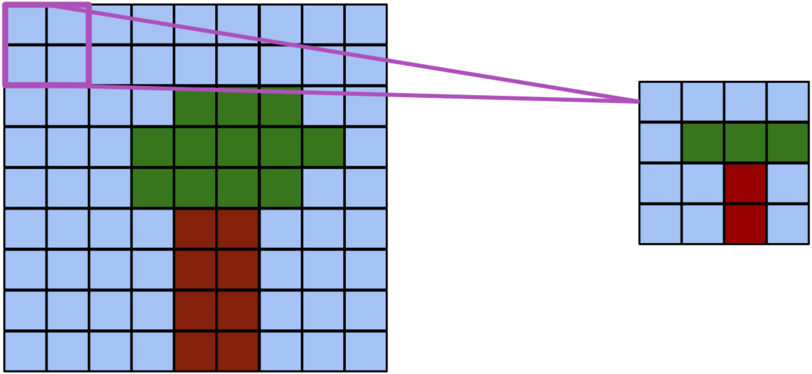


- Kernel slides along the image with a step > 1

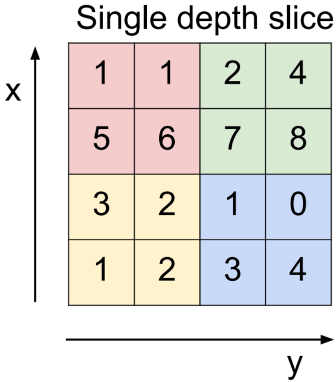


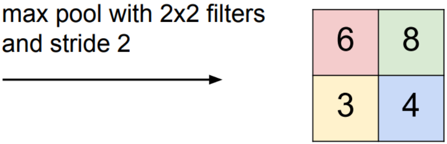
Pooling

- Compute mean or max over small windows to reduce resolution



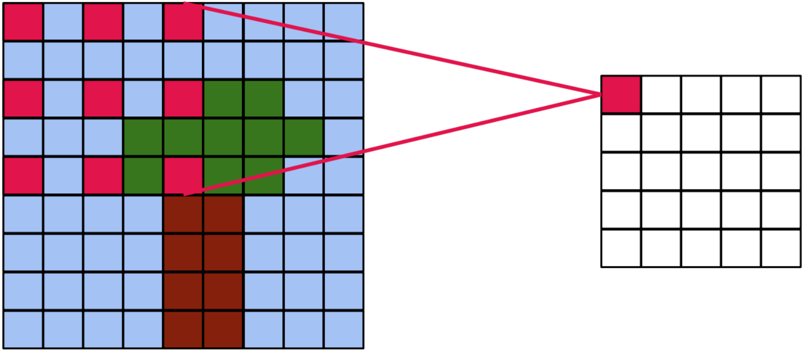
Pooling (cont.)





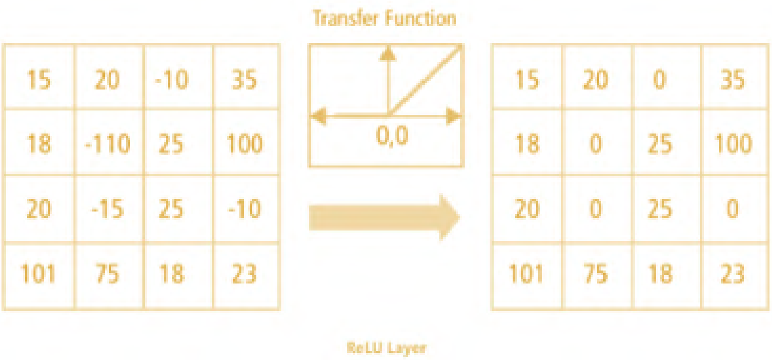
- No learnable parameters
- Introduces spatial invariance

- Kernel is spread out, step >1 between kernel elements



Activation

- Just like Fully-Connected Neural Networks, we can apply an activation over convolutional layer outputs
- It helps break linearity
- For example, Rectified Linear Unit (ReLU): $\sigma(x) = max(0, x)$



Batch Normalization

- Consider a single layer $y = Wx$
- The following could lead to tough optimazation
  - Inputs x are not centered around zero (need large bias)
  - Inputs x have different scaling per element (entries in W will need to vary a lot)

Batch Normalization

- Consider a single layer $y = Wx$
- The following could lead to tough optimazation
  - Inputs x are not centered around zero (need large bias)
  - Inputs x have different scaling per element (entries in W will need to vary a lot)
- **Idea:** Force inputs to be "nicely scaled" at each layer!

Batch Normalization

- Consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

- Consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

- **Problem:** What if zero-mean, unit variance is too hard of a constraint?

Batch Normalization

Input: $x : N \times D$

Learnable scale and shift parameters:

$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Estimates depend on minibatch; can't do this at test-time!

Input: $x : N \times D$

Learnable scale and shift parameters:

$$\gamma, \beta : D$$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$$\boxed{\begin{aligned}\mu_j &= \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \\ \sigma_j^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2\end{aligned}}$$

Per-channel mean, shape is D

Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

Batch Normalization

Input: $x : N \times D$

Learnable scale and shift parameters:

$$\gamma, \beta : D$$

During testing batchnorm becomes a linear operator! Can be fused with the previous fully-connected or conv layer

$\mu_j =$ (Running) average of values seen during training — Per-channel mean, shape is D

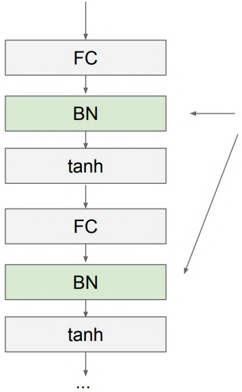
$\sigma_j^2 =$ (Running) average of values seen during training — Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\boxed{\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}}$$

Batch Normalization

Batch Normalization for **fully-connected** networks

$$\mathbf{x}: \; N \times D$$

Normalize

$$\boldsymbol{\mu}, \boldsymbol{\sigma}: \; 1 \times D$$

$$\boldsymbol{\gamma}, \boldsymbol{\beta}: \; 1 \times D$$

$$\mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \beta$$

Batch Normalization for **convolutional** networks (Spatial Batchnorm, BatchNorm2D)

$$\mathbf{x}: \; N \times C \times H \times W$$

Normalize

$$\boldsymbol{\mu}, \boldsymbol{\sigma}: \; 1 \times C \times 1 \times 1$$

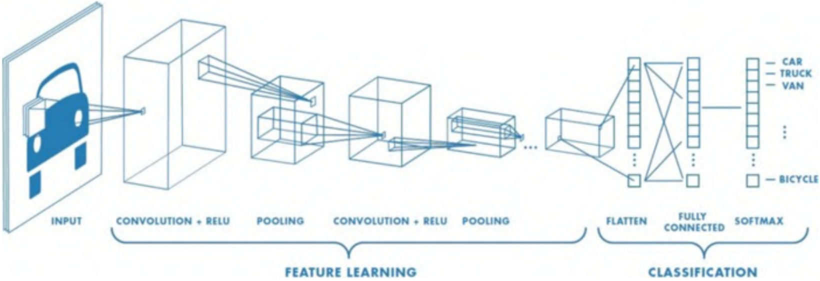
$$\boldsymbol{\gamma}, \boldsymbol{\beta}: \; 1 \times C \times 1 \times 1$$

$$\mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \beta$$

- Advantages:
  - Makes deep networks much easier to train!
  - Improves gradient flow
  - Allows higher learning rates, faster convergence
  - Networks become more robust to initialization
  - Acts as regularization during training
  - Zero overhead at test-time: can be fused with conv!

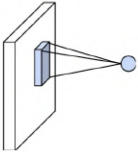
- Advantages:
  - Makes deep networks much easier to train!
  - Improves gradient flow
  - Allows higher learning rates, faster convergence
  - Networks become more robust to initialization
  - Acts as regularization during training
  - Zero overhead at test-time: can be fused with conv!
- Disadvantages:
  - Behaves differently during training and testing: this is a very common source of bugs!

Convolutional Neural Networks

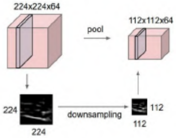


Components of a CNN

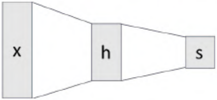
Convolution Layers



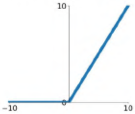
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

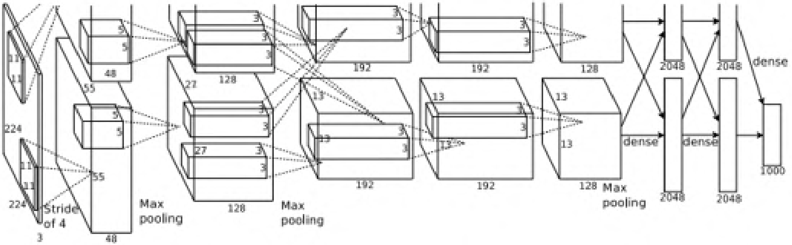
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Most Notable CNNs

- AlexNet *[Krizhevsky et al. 2012]*
- VGGNet *[Simonyan and Zisserman, 2014]*
- InceptionNet (GoogLeNet) *[Szegedy et al., 2014]*
- ResNet *[He et al., 2015]*

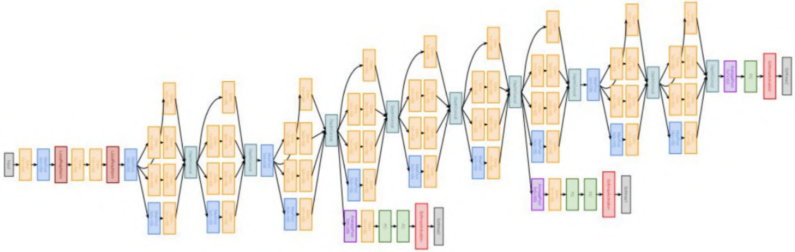
KAUST ACADEMY

- First big improvement in image classification
- Made use of CNN, pooling, dropout, ReLU and training on GPUs.
- 5 convolutional layers, followed by max-pooling layers; with three fully connected layers at the end

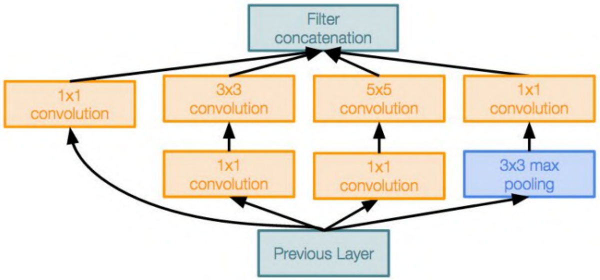


- Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer
- But deeper, more non-linearities and lesser parameters
- 13 or 16 conv layers with 3 fully-connected layers. Most params in the fully connected layer

- Going Deep: 22 layers
- Only 5 million parameters! (12x less than AlexNet and 27x less than VGGNet)
- Introduced efficient "Inception module"
- Introduced "bottleneck" layers that use 1x1 convolutions to reduce feature channel size and computational complexity



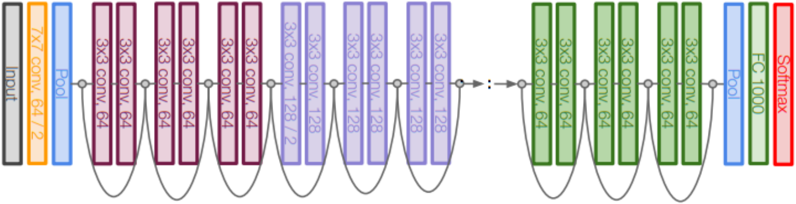
- **Inception module:** design a good local network topology (network within a network) and then stack these modules on top of each other



Inception module

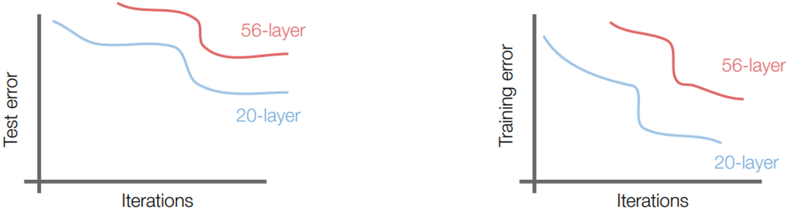
ResNet

- Very deep networks using residual connections
- 152-layer model for ImageNet
- Stacked Residual Blocks



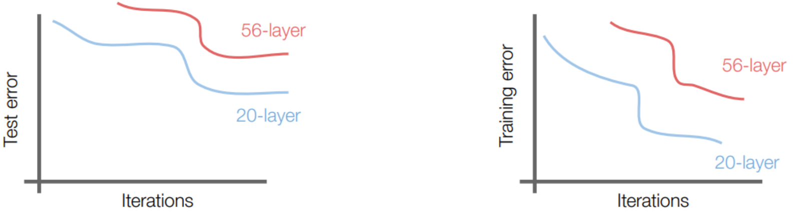
- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?

- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?



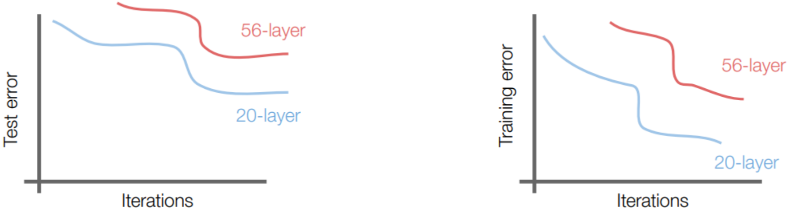
ResNet

- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?



- 56-layer model performs worse on both test and training error

- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?



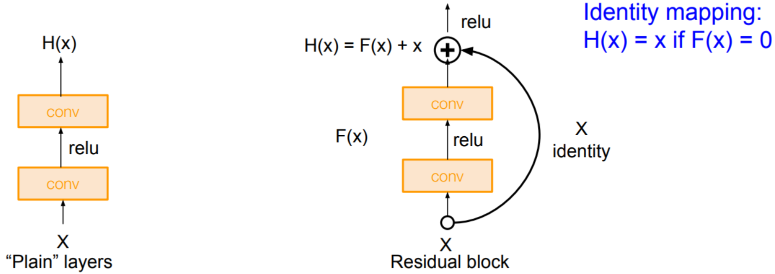
- 56-layer model performs worse on both test and training error
- The deeper model performs worse, but it's not caused by overfitting!

- **Fact:** Deep models have more representation power (more parameters) than shallower models.

- **Fact:** Deep models have more representation power (more parameters) than shallower models.
- **Hypothesis:** The problem is an optimization problem, deeper models are harder to optimize

- **Fact:** Deep models have more representation power (more parameters) than shallower models.
- **Hypothesis:** The problem is an optimization problem, deeper models are harder to optimize
- **Solution:** Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

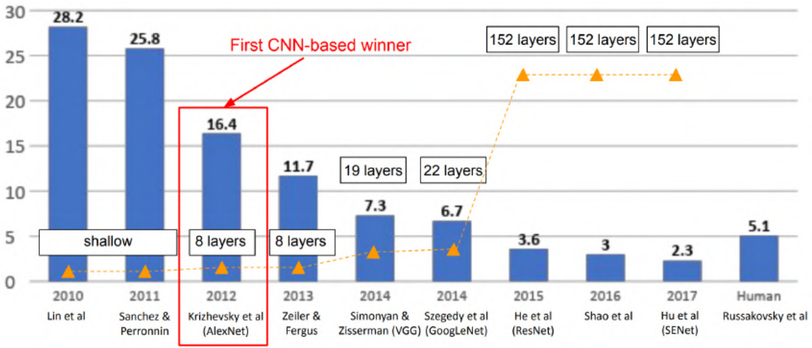
- **Fact:** Deep models have more representation power (more parameters) than shallower models.
- **Hypothesis:** The problem is an optimization problem, deeper models are harder to optimize
- **Solution:** Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



ImageNet

- The most extensive data for Image Classification
- 3 RGB channels from 0 to 255
- 14,197,122 images
- 1000 classes

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



References

These slides have been adapted from

- Fei-Fei Li, Yunzhu Li & Ruohan Gao, Stanford CS231n: Deep Learning for Computer Vision
- Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
- Justin Johnson, UMich EECS 498.008/598.008: Deep Learning for Computer Vision
- Sander Dieleman, Deepmind: Deep Learning Lecture Series 2020