Visual Computing: Convolutional Neural Networks

Prof. Marc Pollefeys

ETH

inf | Informatik
Computer Science

Convolutional Neural Network

- Motivation for deep learning
- Linear classifier
- Activation functions
- Optimization
- Back propagation
- Motivation for CNN
- Convolution layer

Motivation

- Recall: handcrafted convolutional kernels

$$* \begin{pmatrix} -1 & 0 & 1 \\ -1 & [0] & 1 \\ -1 & 0 & 1 \end{pmatrix} \rightarrow$$

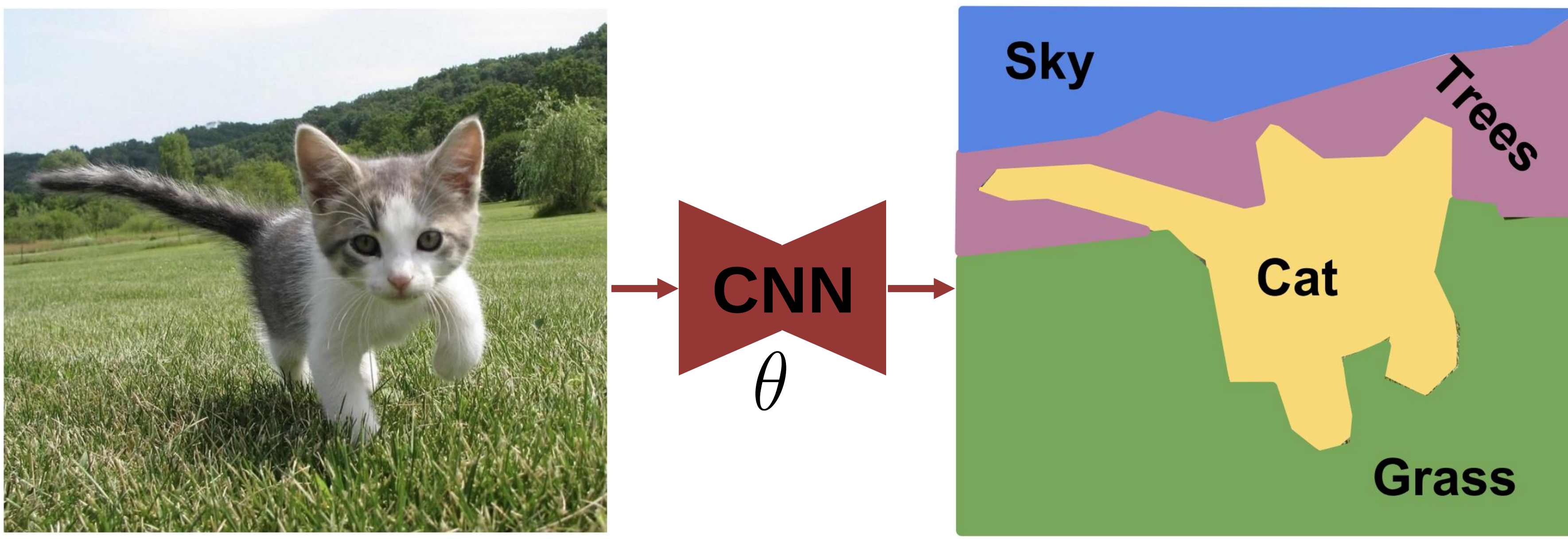
- What if we want to find more complex relation? Eg. Classify the image as a cat?

$$* \begin{pmatrix} ? \end{pmatrix} \rightarrow y$$

$$\mathcal{L}(y, \hat{y})$$

Deep Learning

- What we will see



x $f(\boldsymbol{x}, \theta)$

$$\underset{\theta}{\operatorname{argmin}} \mathcal{L}\big(\boldsymbol{y}, f(\boldsymbol{x}, \theta)\big)$$

Data-Driven Approach

- **Goal**: summarize the input – output relationship directly from a collection of data
- **Overview**

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}\big(\boldsymbol{y}, f(\boldsymbol{x}, \theta)\big)$$

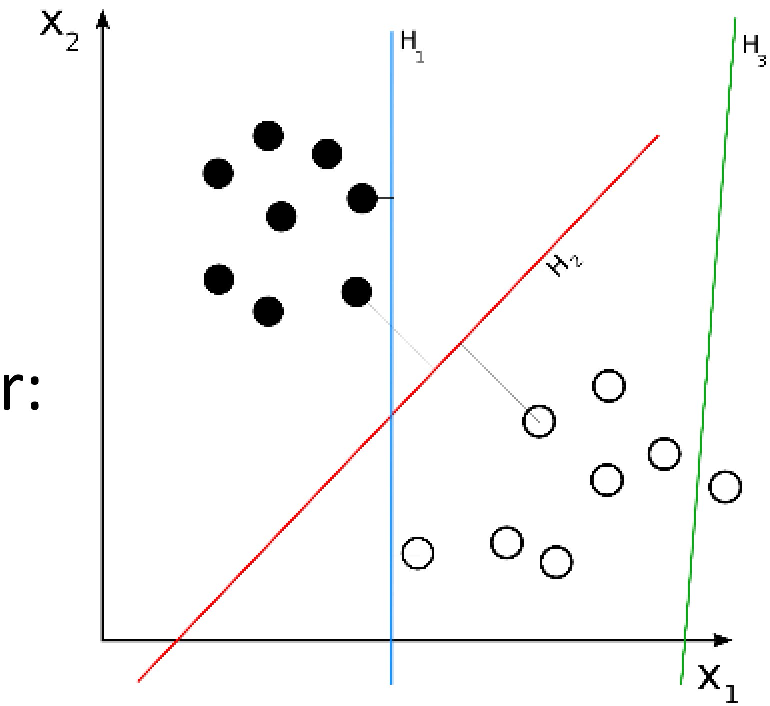
  – $\boldsymbol{x}$ input
  – θ kernel weights
  – $f(x, \theta)$ prediction
  – $\boldsymbol{y}$ learning target
  – $\mathcal{L}$ loss function

A Simplified Problem

- Task: separate black dots from white ones
- **Linear classifier:**

$$f(x, \theta) = Wx + \mathrm{b}$$

Called fully connected layer: weights interact with **all** dimension of data **simultaneously**

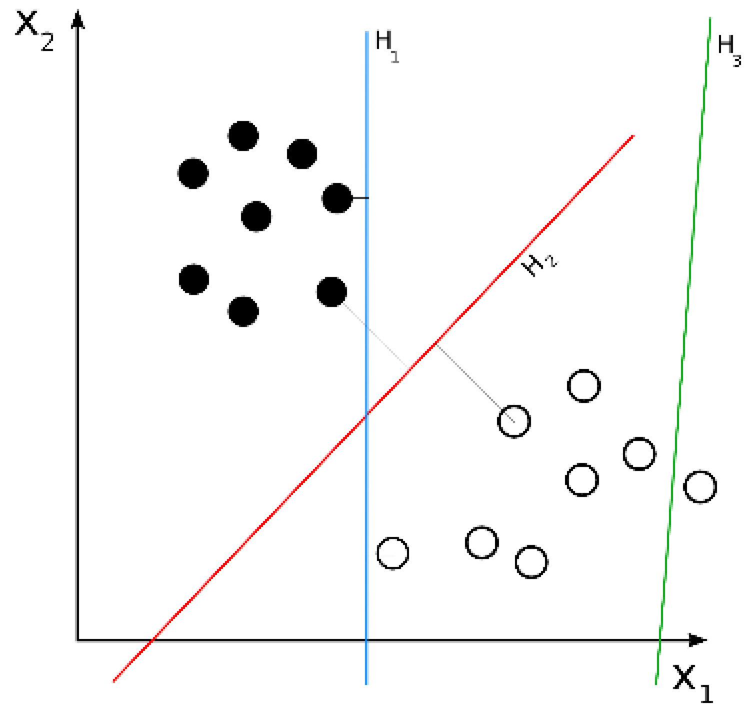


Credit: Wikipedia

- Model parameters $\theta = \{\mathrm{W}, \mathrm{b}\}$

Loss Function

- Three classifier H_1, H_2, H_3 , how to compare ?
  - Loss function!
- A **loss function** quantifies the quality of a classifier



Credit: Wikipedia

Softmax (Logistic) Classifier

- scores = unnormalized log probabilities of different classes ➔ maximize the probability

$$P(Y = k \,|X = \boldsymbol{x_i}) = \frac{e^{\boldsymbol{s}_{i,k}}}{\sum_j e^{\boldsymbol{s}_{i,j}}} = P_{i,k}, \boldsymbol{s_i} = f(\boldsymbol{x_i}, \theta)$$

- (**Softmax**) Loss $\mathcal{L}\big(\boldsymbol{y}, f(\boldsymbol{x}, \theta)\big) = -\sum_{i=1}^{N} \log \frac{e^{\boldsymbol{s}_{i,y_i}}}{\sum_j e^{\boldsymbol{s}_{i,j}}} \;, y_i \in \mathbb{N}$
- Minimize the negative log likelihood of the correct class
- If only two class $y_i \in \{0,1\}$ and one score: **logistic** loss

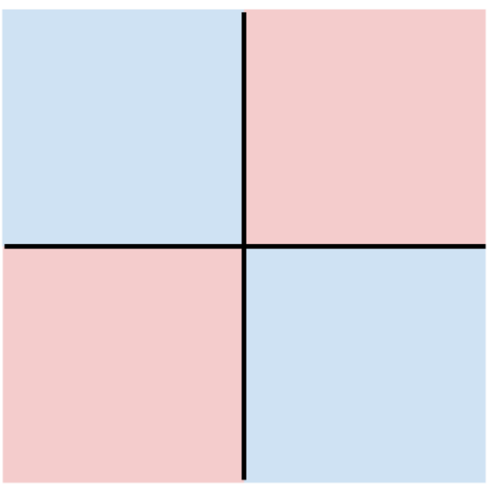
$$\mathcal{L}\big(y, f(x, \theta)\big) = \frac{1}{N} \sum_{i=1}^{N} y_i \log \frac{e^s}{1 + e^s} + (1 - y_i) \log \frac{1}{1 + e^s}$$

Limitations for Linear Classifier

- Not all classes are linear separable

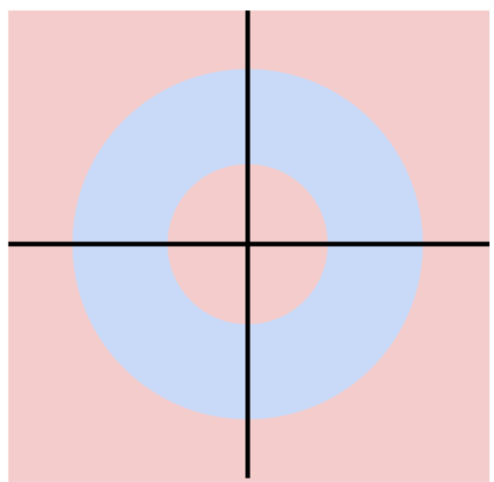
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



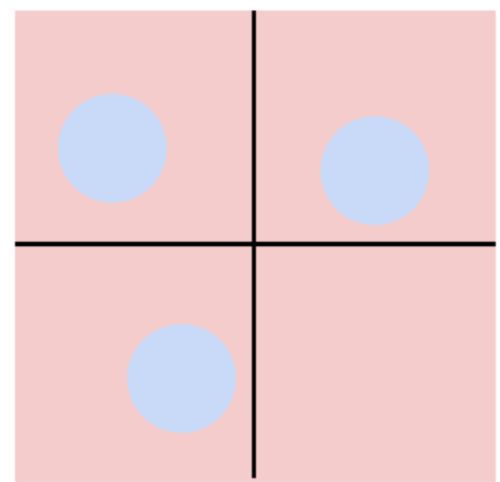
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

First Trial

- Address the limitation by stacking more layers

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = W_2(W_1\boldsymbol{x} + b_1) + b_2$$
$$= W_2W_1\boldsymbol{x} + (W_2b_1 + b_2)$$

- Collapse to the single layer case, not working
- Non-linearity is necessary:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = W_2\phi(W_1\boldsymbol{x} + b_1) + b_2$$

$\phi(x) \rightarrow$ non-linear, scalar "activation" function

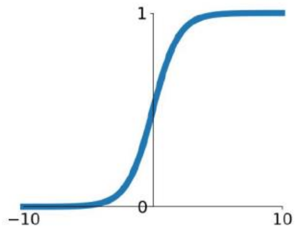
- Q: What is a good activation function?

Activation Functions

- Introduce non-linearity by activation functions

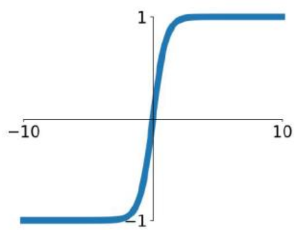
Sigmoid

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



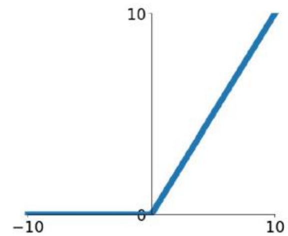
tanh

$\tanh(x)$



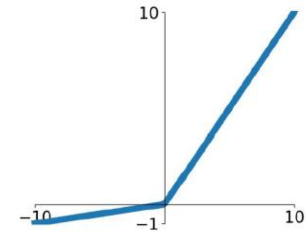
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

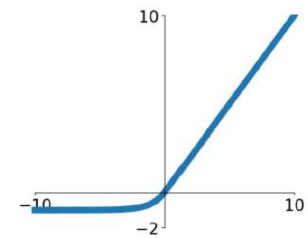


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

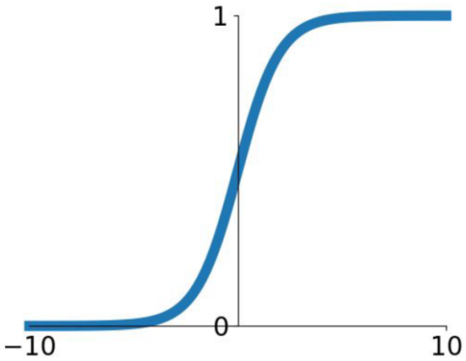
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Activation Functions

Activation Functions



Sigmoid

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

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

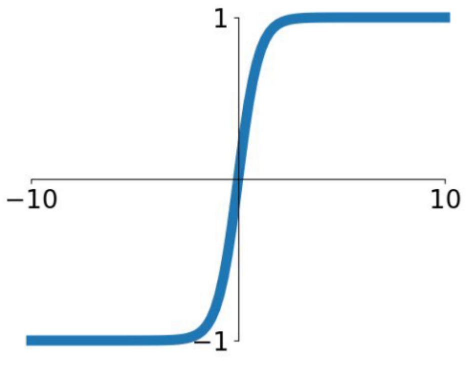
3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Activation Functions

Activation Functions



tanh(x)

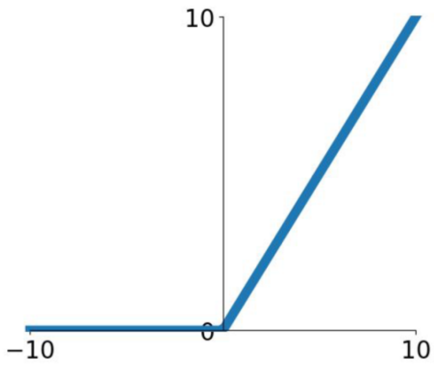
- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Activation Functions

Activation Functions



ReLU
(Rectified Linear Unit)

- Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

- Not zero-centered output
- An annoyance:

hint: what is the gradient when $x < 0$?

Ans: Dead ReLU will never activate ➔ usually initialize with slightly positive biases

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Activation Functions

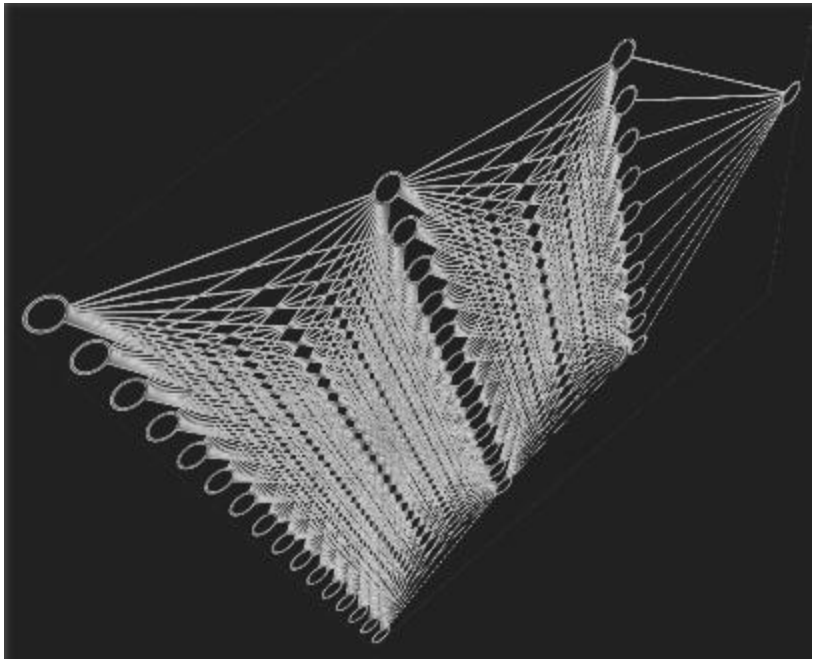
TLDR: In practice:

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don't expect much
- Don't use sigmoid

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Multilayer Perceptron (MLP)

- Stack several linear classifiers
  - One or more more "hidden" layers
- Add activation function between layers
- Can distinguish data that is not linearly separable
- "Universal approximator"



Credit: Wikipedia

Optimization

- Find the best weights (θ) that minimize the loss function

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Optimization

Strategy #1: A first very bad idea solution: **Random search**

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (trunctated: continues for 1000 lines)
```

15.5% accuracy vs SOTA >95%

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Optimization

Strategy #2: **Follow the slope**

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Optimization

Strategy #2: **Follow the slope**

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the **gradient** is the vector of (partial derivatives) along each dimension

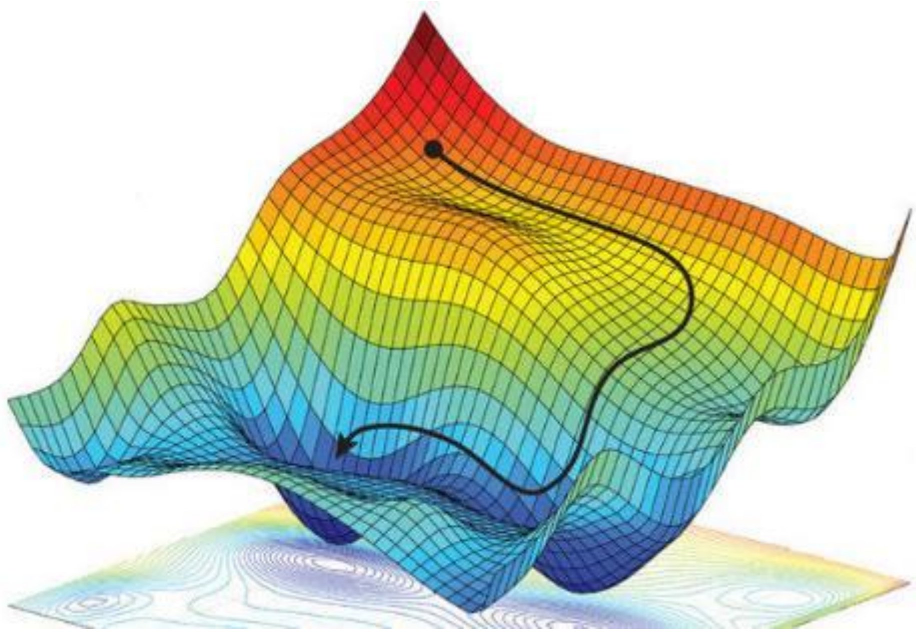
The slope in any direction is the **dot product** of the direction with the gradient
The direction of steepest descent is the **negative gradient**

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Gradient Descent

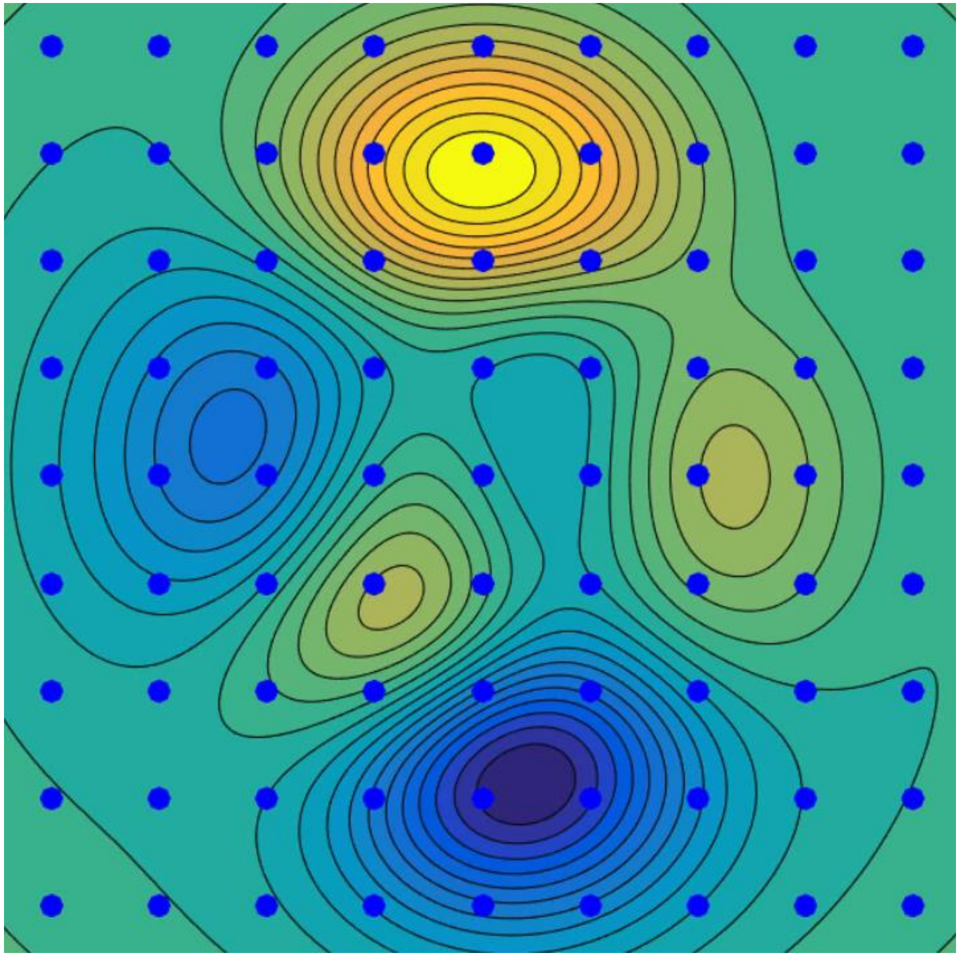
$$\theta_{t+1} = \theta_t + \lambda \nabla \mathcal{L}_\theta$$

- $\nabla \mathcal{L}_\theta$ gradient of $\mathcal{L}(y, f(x, \theta_t))$ with respect to θ.
- λ step size, control how far each step goes → "learning rate"



Credit: Alexander Amini et al.

Gradient Descent



Credit: Wikipedia

Stochastic Gradient Descent (SGD)

$$\nabla_\theta \mathcal{L}\big(y, f(x,\theta)\big) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{L}\big(y_i, f(x_i,\theta)\big)$$

- When N is large, estimating the full gradient is expensive
- Approximate sum using a minibatch of examples

$$\nabla_\theta \mathcal{L}\big(y, f(x,\theta)\big) \approx \frac{1}{B}\sum_{i=1}^{B} \nabla_\theta \mathcal{L}\big(y_i, f(x_i,\theta)\big), B < N$$

  – B = 32 / 64 / 128 common
- Make a step per minibatch → repeat with next batch

Back Propagation

- For linear classifier $f(x, \theta) = Wx + b$:

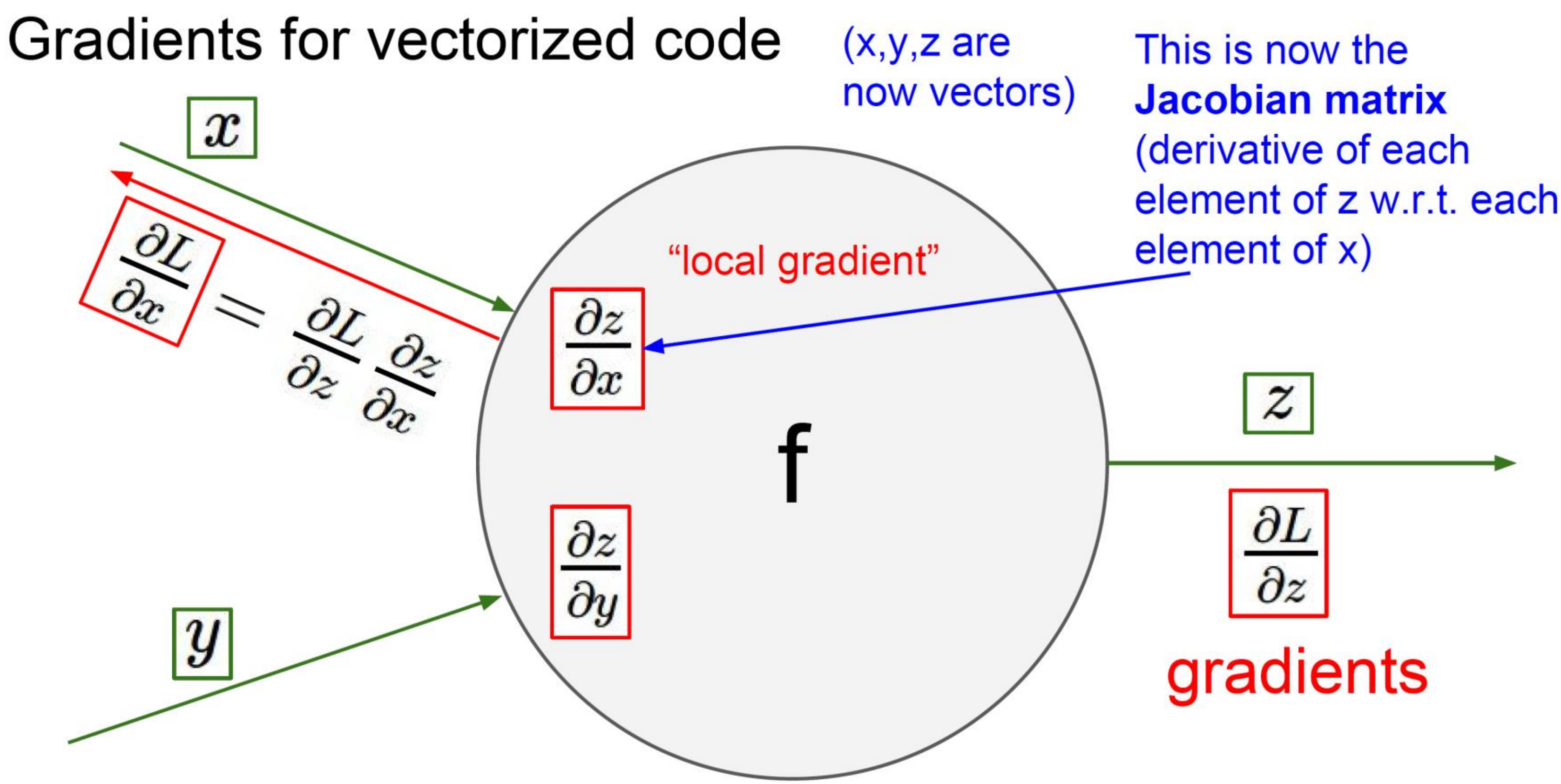
$$\nabla_\theta \mathcal{L}\big(y_i, f(x_i, \theta)\big) == \frac{\partial \mathcal{L}}{\partial f} x_i$$

- For MLP, use chain rule

$$\nabla_\theta \mathcal{L}\big(y_i, f(x_i, \theta)\big) = \frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial \theta}$$

- **Back propagation**: recursive application of the chain rule to compute the gradients

Back Propagation



Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Back Propagation

Credit

Scaling Up

- So far (fully connected layer)

  $$f(x,\theta) = g_n(W_n \cdots g_1(W_1 x + b_1) + \cdots b_2)$$

- Dimension of weights
  - $W_1 \in \mathbb{R}^{D \times k}$ where D is the dimension of input data k the dimension intermediate layers
  - $D = 2$ for the point separation
  - $D = ?$ for image separation

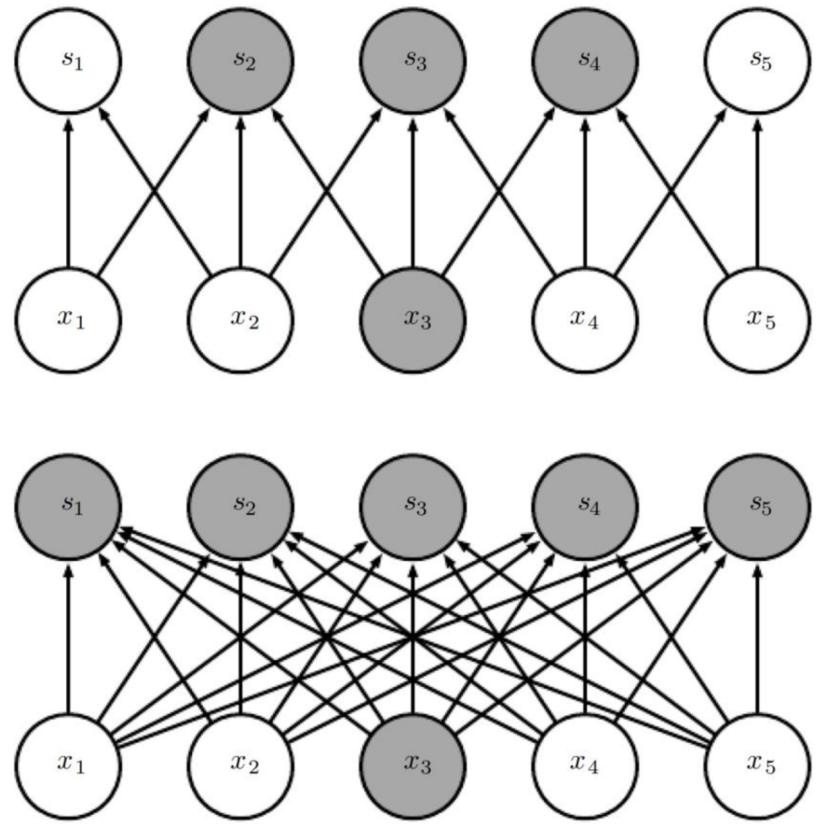
ETH

Scaling Up

- So far (fully connected layer)
  $$f(x, \theta) = g_n(W_n \cdots g_1(W_1 x + b_1) + \cdots b_2)$$
- Dimension of weights
  - $W_1 \in \mathbb{R}^{D \times k}$ where D is the dimension of input data k the dimension intermediate layers
  - $D = 2$ for the point separation
  - $D = 3 \times 10^6$ for image (1000×1000 px) separation
  - Expensive!

Motivation for Convolution Layer

- Sparse interactions
  - Also called sparse connectivity or sparse weights
  - Making the kernel smaller than input



Credit: Goodfellow et al, Deep Learning (2017)

Motivation for Convolution Layer

- Parameter sharing

Credit: Goodfellow et al, Deep Learning (2017)

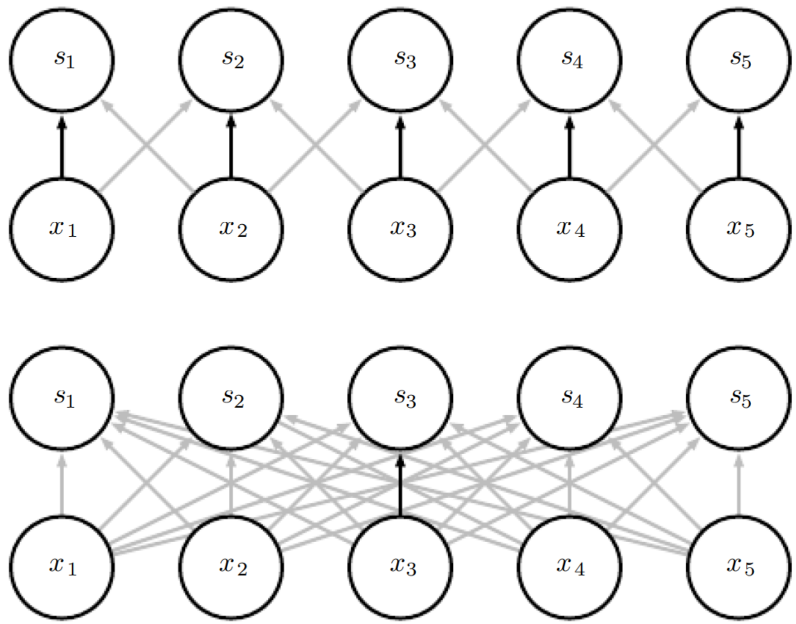


Figure 9.5: Parameter sharing: Black arrows indicate the connections that use a particular parameter in two different models. *(Top)*The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Due to parameter sharing, this single parameter is used at all input locations. *(Bottom)*The single black arrow indicates the use of the central element of the weight matrix in a fully connected model. This model has no parameter sharing so the parameter is used only once.

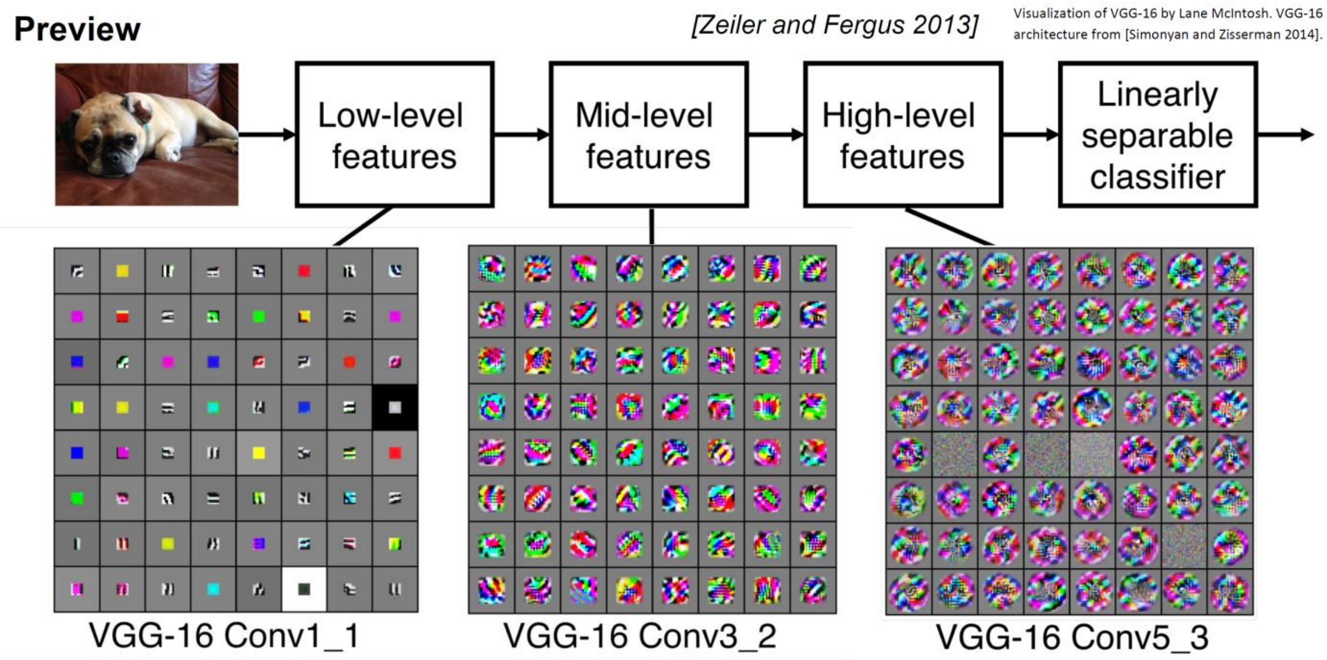
Motivation for Convolution Layer

- Equivariant representations
  - Change the position of an object should not change the classification of it

Credit: Sofa, Cat

Motivation for Convolution Layer

- Hierarchical perception
  - From low-level features to high-level concepts
  - Motivated by perception systems

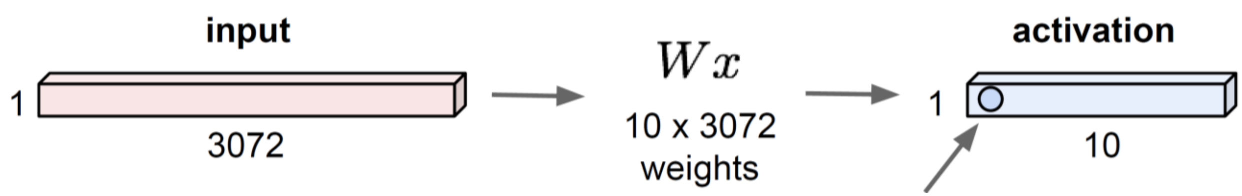


Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

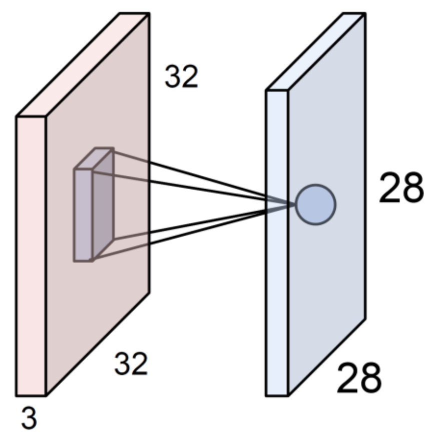
Convolution Layer

- Preservation of spatial structure
  - Fully connected layer stretched an image into 1D vector

Fully connected layer

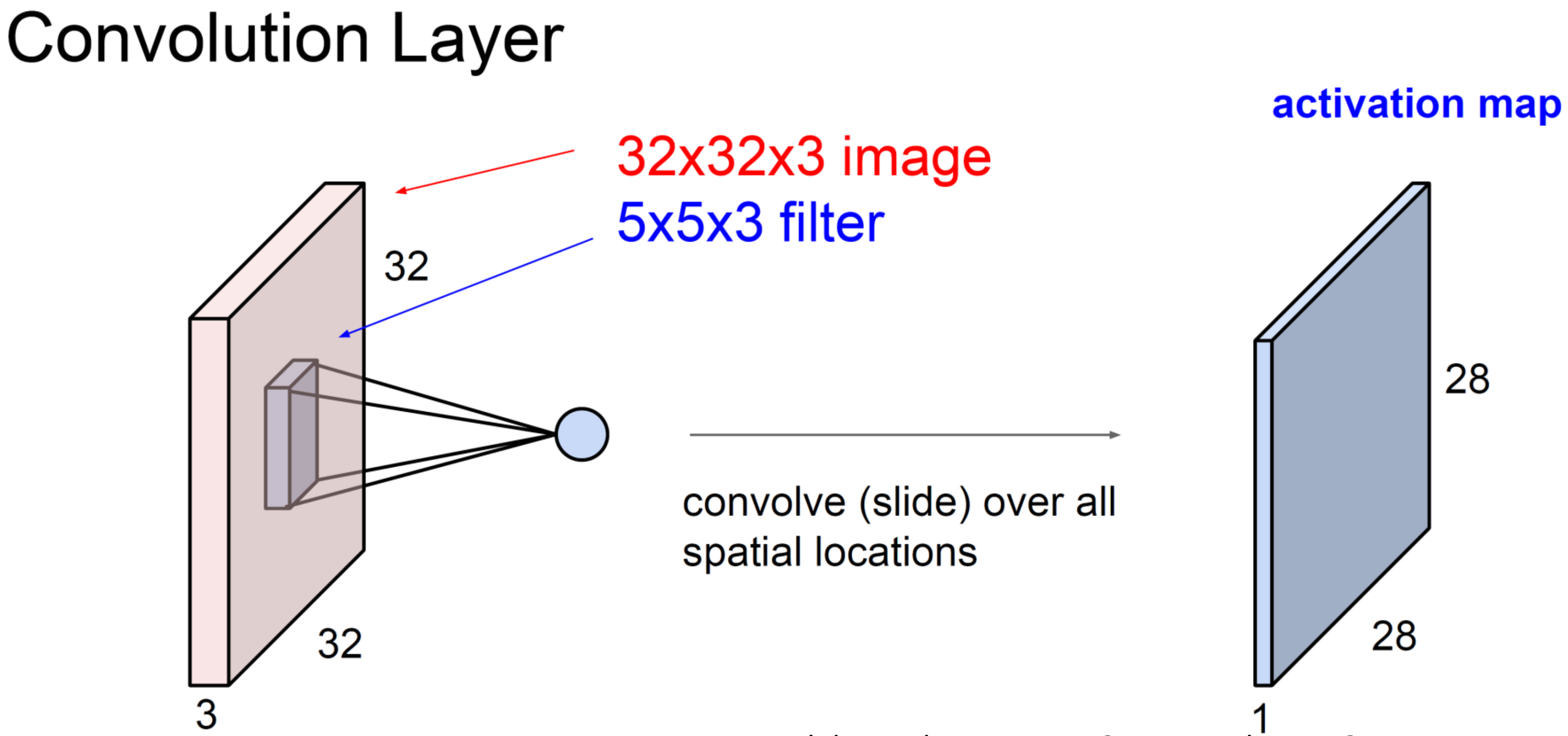


Convolution layer



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Convolution Layer

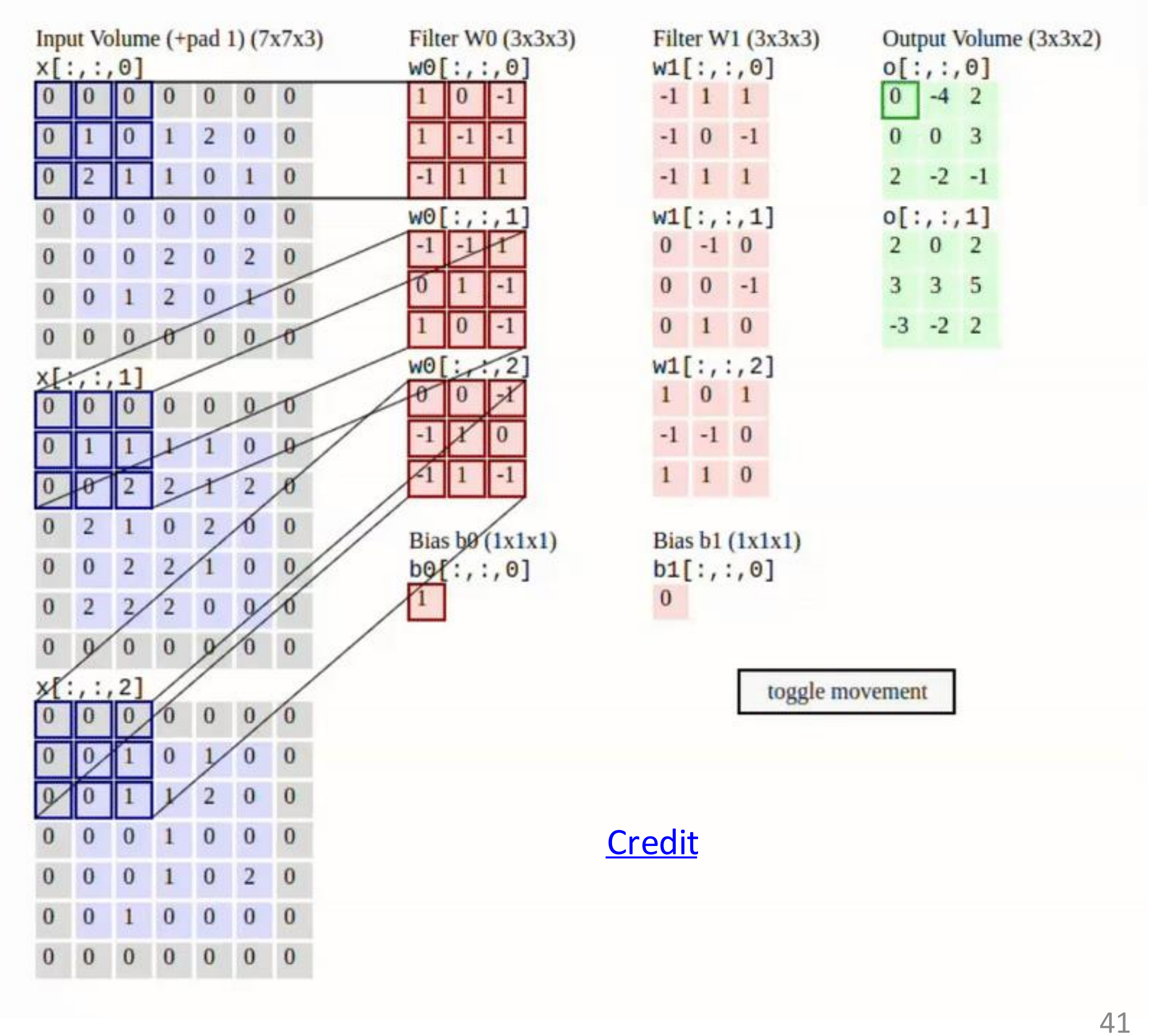


Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Q: How many parameters has a convolutional filter if the input image has N channels and the output feature map has D channels?

Convolution Layer

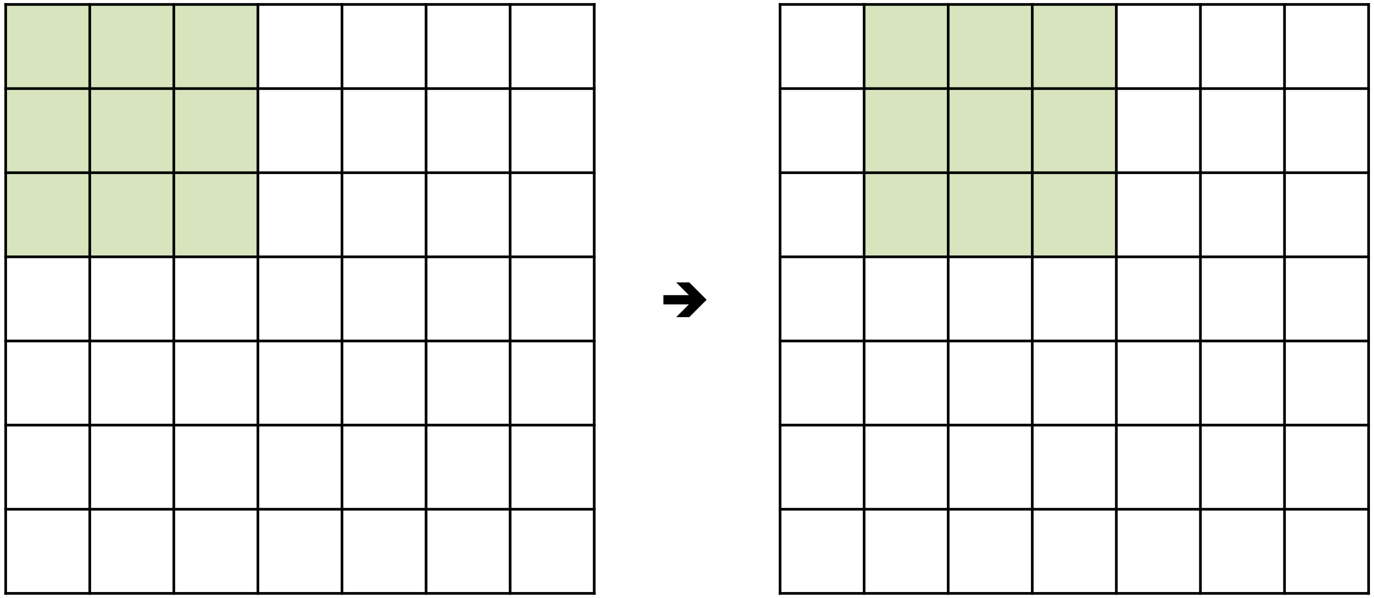
Convolve over all spatial locations



Credit

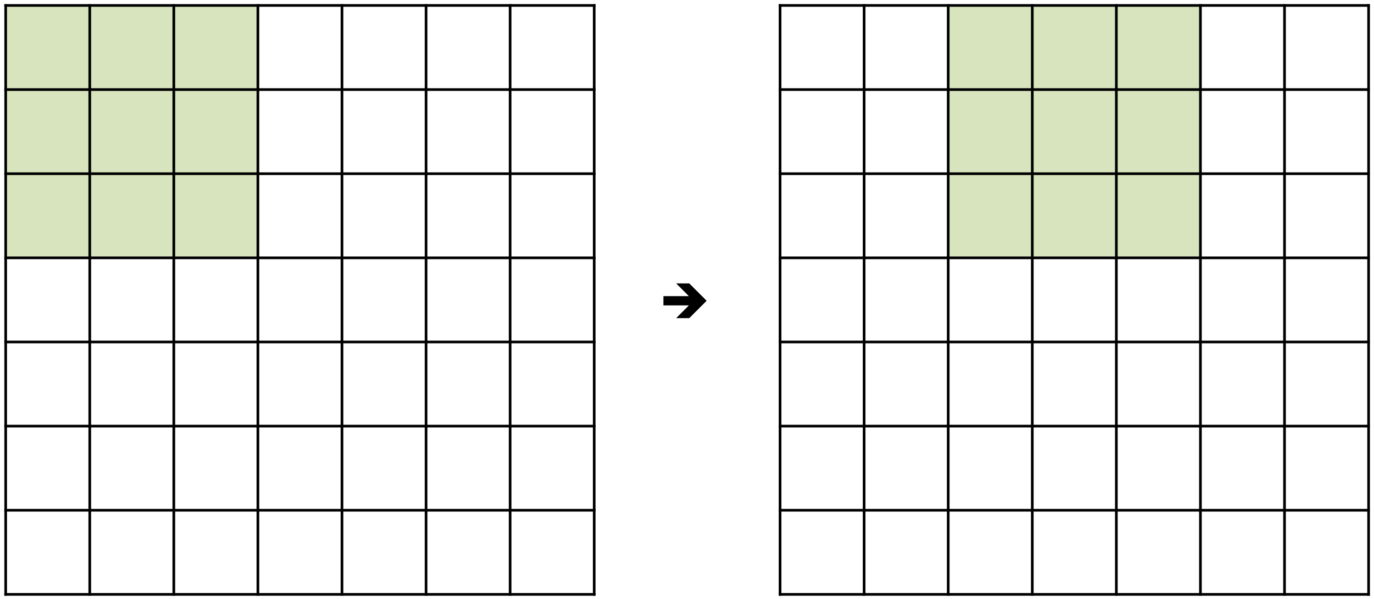
Convolution Layer

- Kernel size: dimension of the weights
- Stride: the step size of applying kernel
- Applying 3×3 kernel on 7×7 grid with stride 1

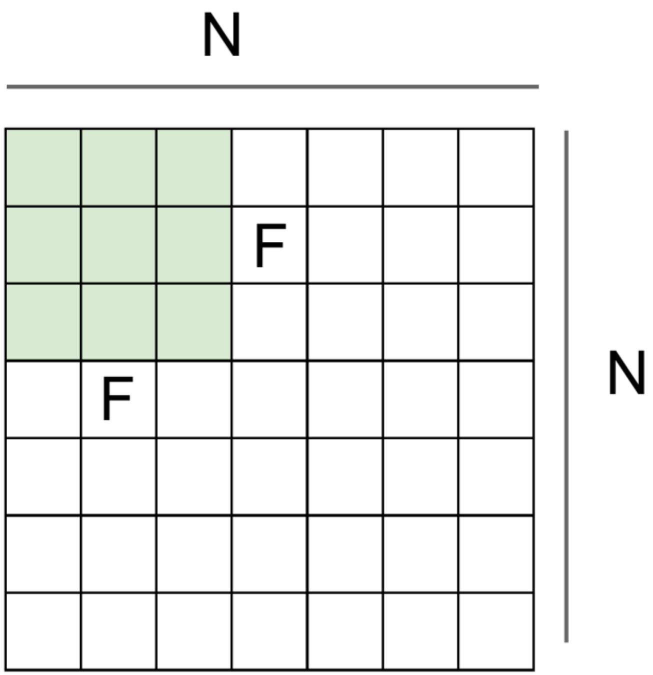


Convolution Layer

- Kernel size: dimension of the weights
- Stride: the step size of applying kernel
- Applying 3×3 kernel on 7×7 grid with stride 2



Output Dimension



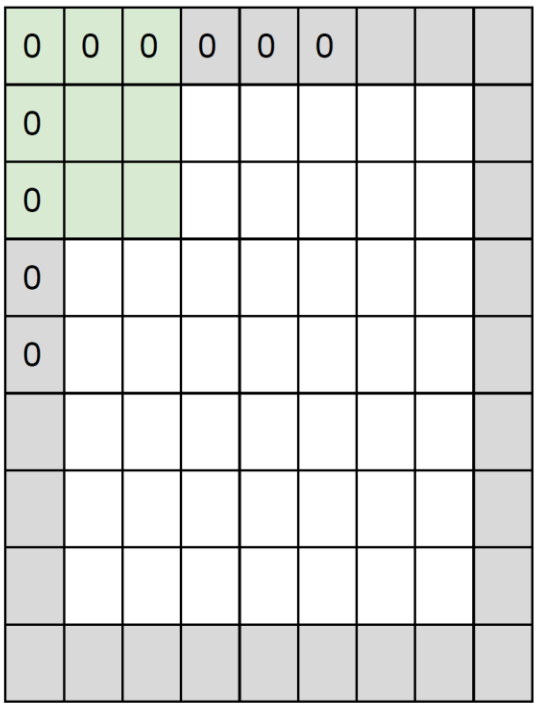
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Zero Padding

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

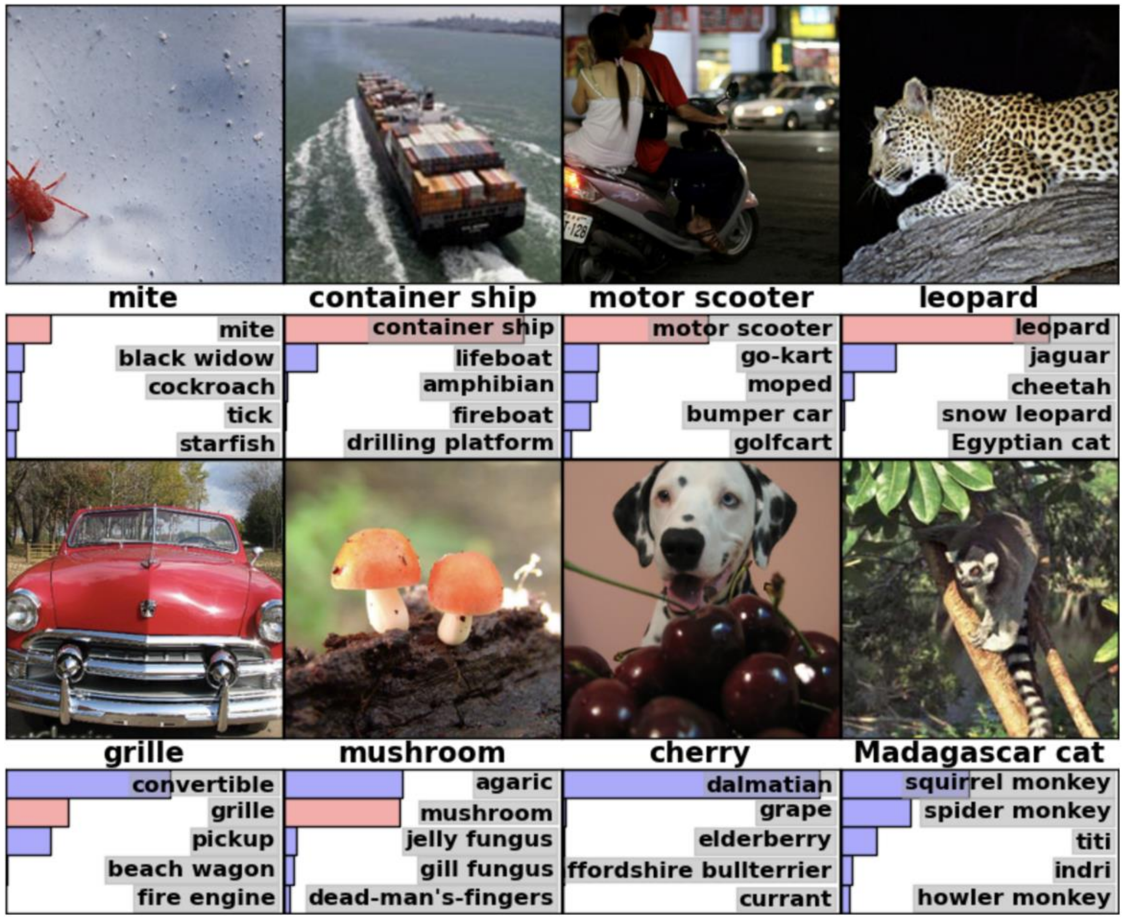
Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

Classification VS Regression

- Classification
  - $f(x_1, \theta)$ as the score
  - take the class with larger score
  - $\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\theta})) = -\sum_{i=1}^{N} \log \frac{e^{s_{i,y_i}}}{\sum_j e^{s_{i,j}}}$, $y_i \in \mathbb{N}, s_i = f(x_i, \theta)$
- Regression
  - $f(x_1, \theta)$ as the value
  - can be used for classification by comparing value
  - $\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\theta})) = \sum_{i=1}^{N} ||y_i - s_i||^2$, $y_i \in \mathbb{R}^n, s_i = f(x_i, \theta)$

ETH

Image Classification

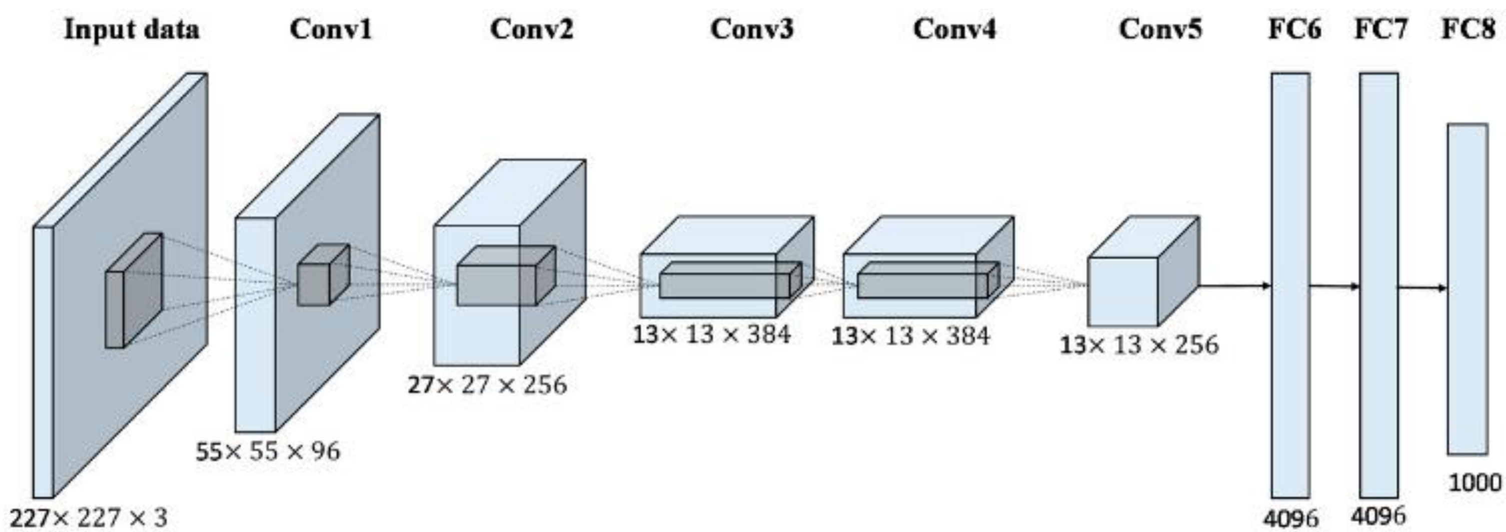


Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks ‘12

ETH

CNN Success Stories

- Early options: Ensemble, boosting, SVM, decision trees, MLP, …
- 2012: AlexNet revolutionizes the field of Computer Vision
- CNN reduces classification error on ImageNet: 26% -> 16.4% error

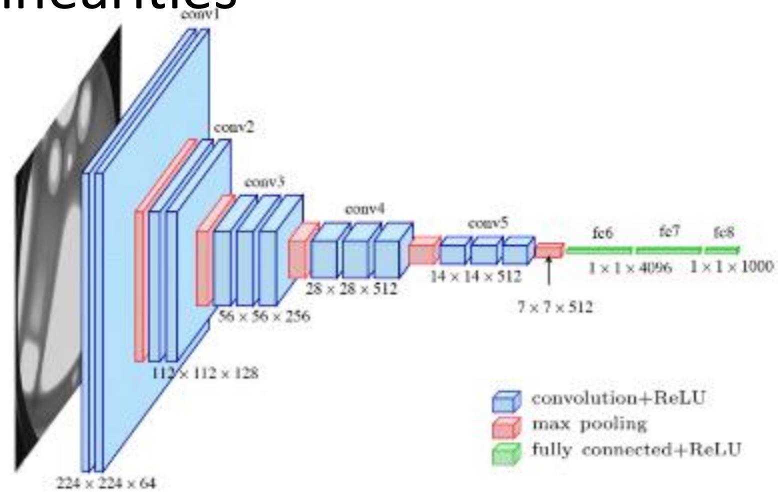


Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks '12

CNN Success Stories

- CNN architectures keep getting refined
- 2014: VGG sets another key benchmark achieving 7.4% error on ImageNet (second best: 14.8% error)
- Key architecture improvements:
  - Reduced kernel size
  - Increased depth

  } same receptive field with more non-linearities

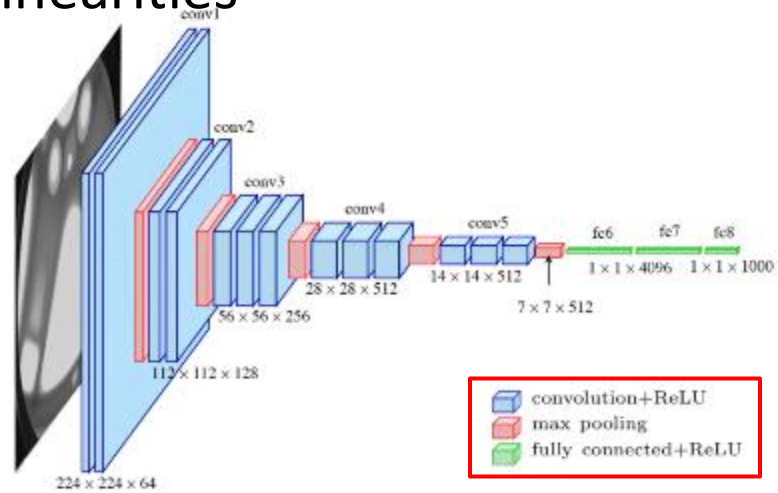


Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." 2014

CNN Success Stories

- CNN architectures keep getting refined
- 2014: VGG sets another key benchmark achieving 7.4% error on ImageNet (second best: 14.8% error)
- Key architecture improvements:
  - Reduced kernel size
  - Increased depth

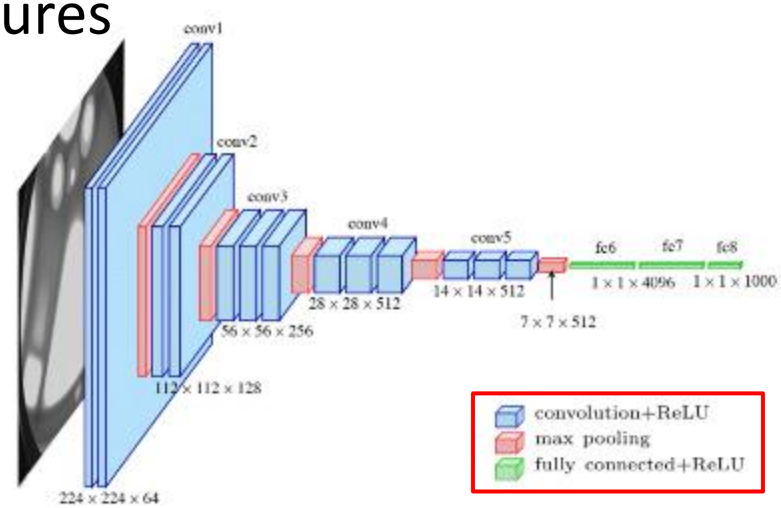
  } same receptive field with more non-linearities



Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." 2014

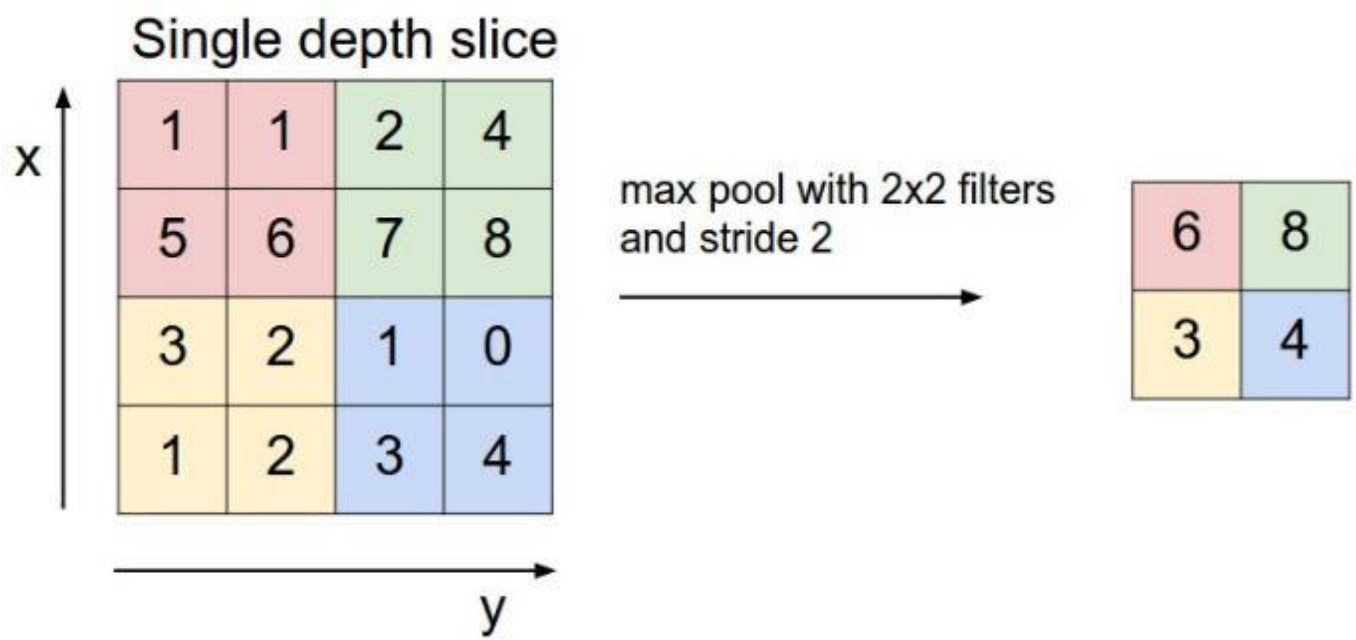
CNN Building Blocks

- We have talked about convolutional layers, fully connected layers and activation functions (ReLU)
- What about max pooling?
  - Dimensionality reduction
  - Introduces translation invariance (could remove)
  - Helps to extract dominant features



CNN Building Blocks

- Max Pooling



cs231n.github.io/convolutional-networks

CNN Success Stories

- Very deep networks need new building blocks to achieve their full potential
- 2015: ResNet achieves 3.57% error on imagenet and is the foundational architecture many subsequent innovations
- Key architecture improvement: residual block
  - Add skip connections -> more stable gradients
  - Intuition: option to rely less on depth

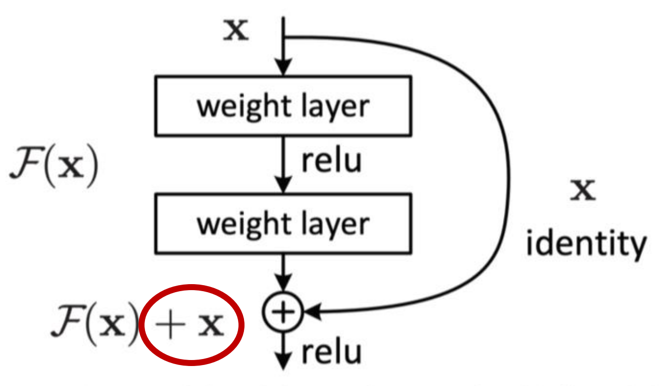
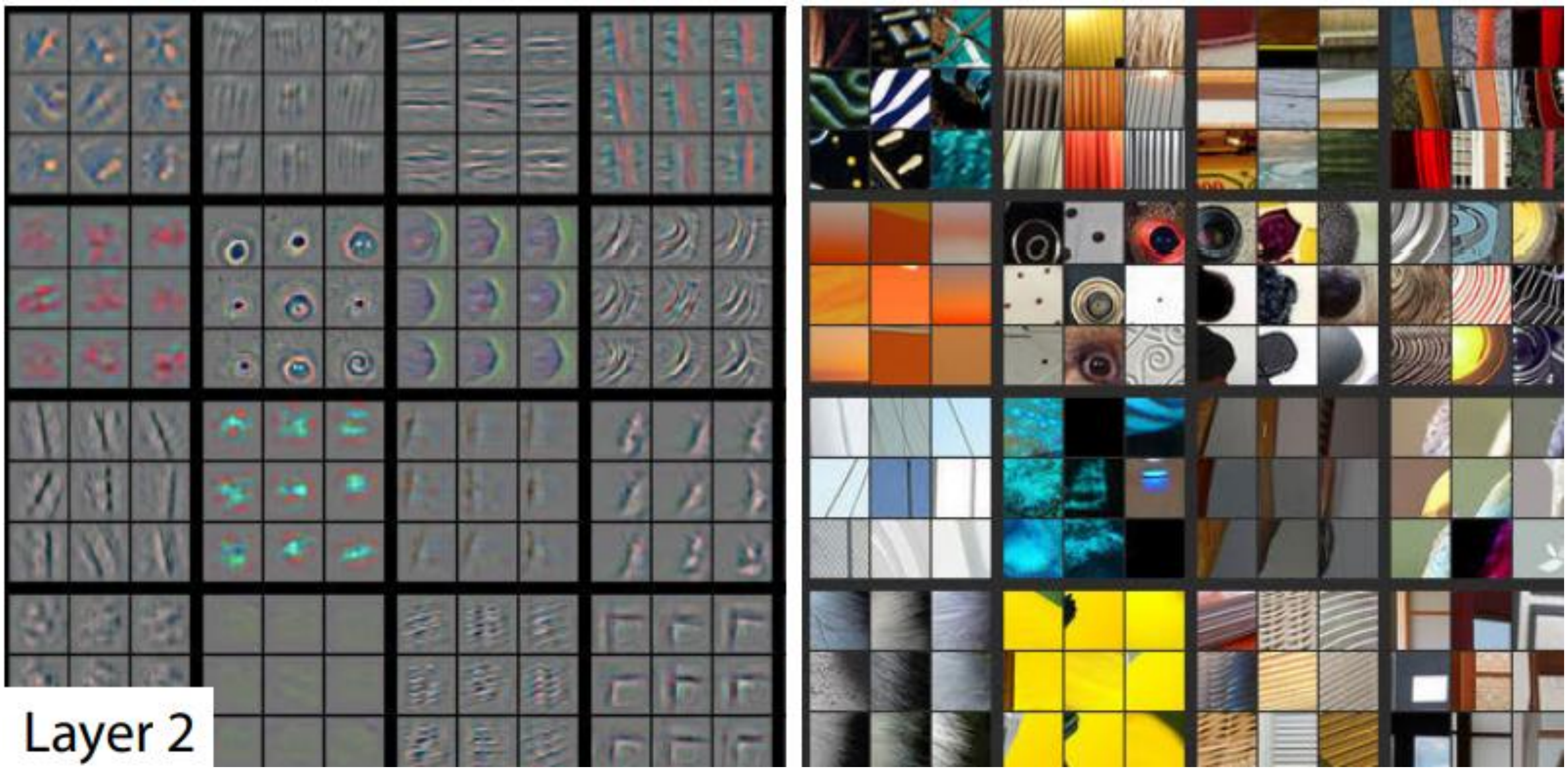


Figure 2. Residual learning: a building block.

Understanding CNNs

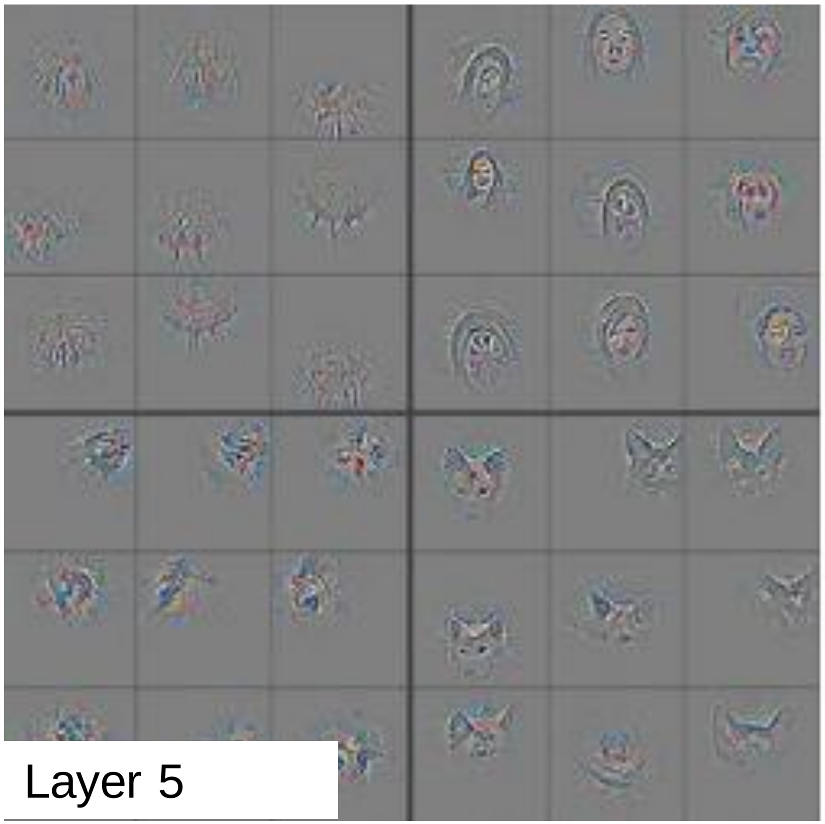


Activation

Input image patch

Zeiler et al, 2014

Understanding CNNs



Activation

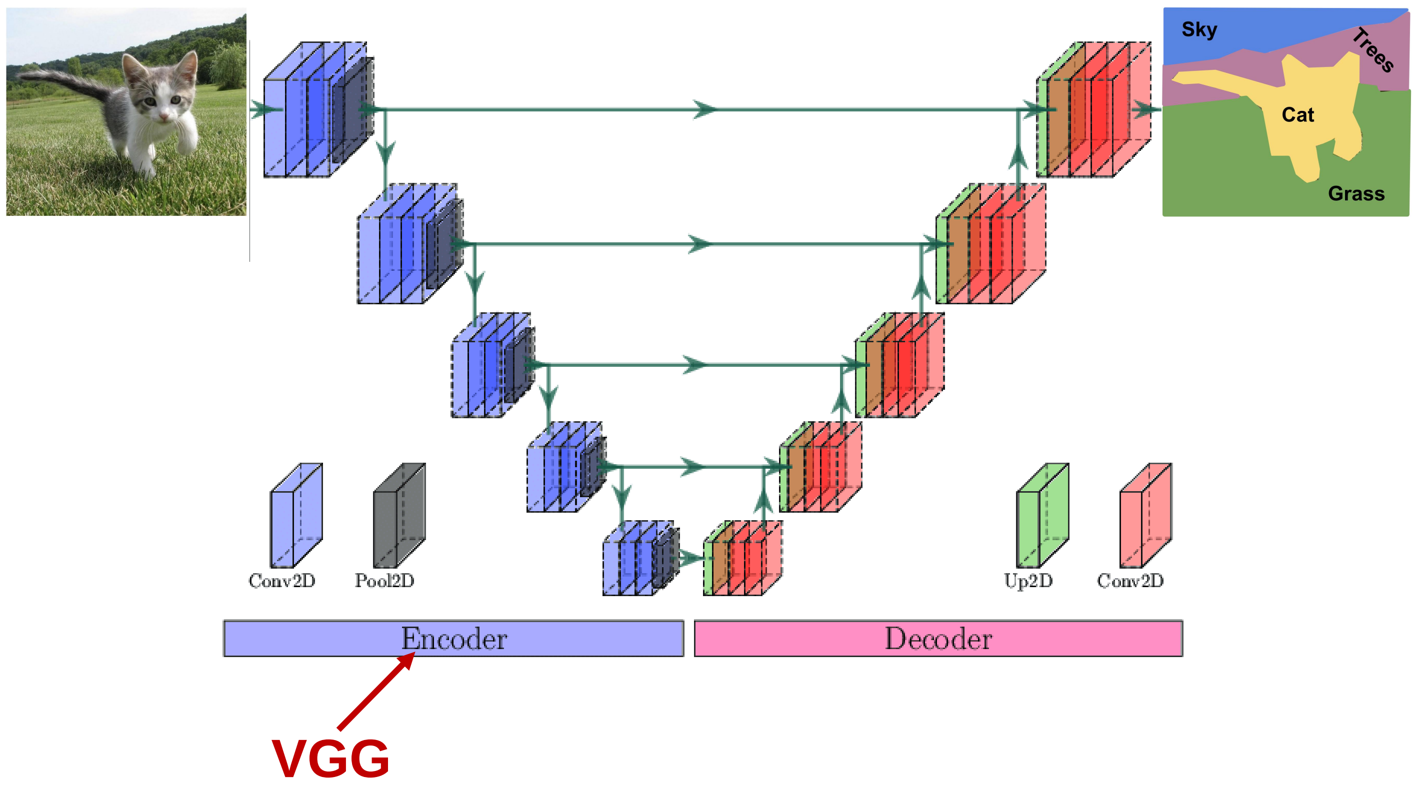
Input image patch

Zeiler et al, 2014

ETH

Beyond Classification

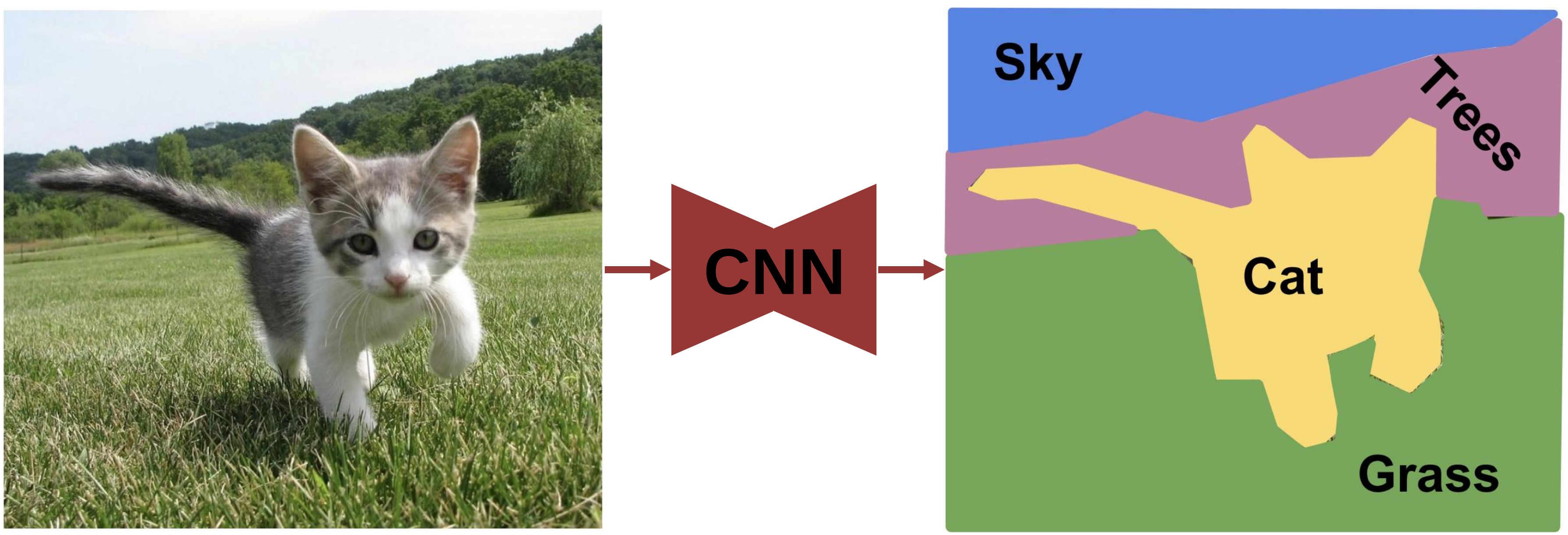
- Classification networks are powerful backbones for other tasks



Ghosh, Tarun Kanti, et al. "Multi-class probabilistic atlas-based whole heart segmentation method in cardiac CT and MRI." 2021

Beyond Classification

- Semantic segmentation
  - Instead of classifying an image, we can classify each pixel



$$\mathcal{L} = -\sum_i \sum_c y_{ic} \log(p_{ic})$$

Semantic Segmentation

- Semantic segmentation SOTA: Segment Anything
  - Trained on 1B+ MASKS

segment-anything.com

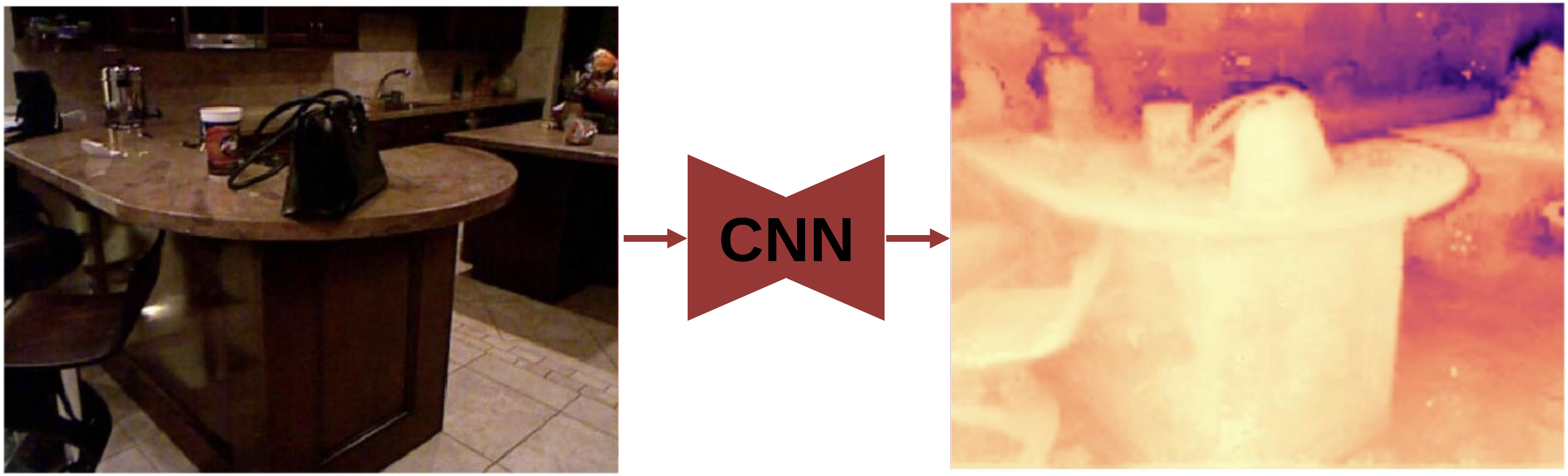
Semantic Segmentation

- Semantic segmentation SOTA: Segment Anything
  - Can easily transfer labels to never before seen data

segment-anything.com

Depth Estimation

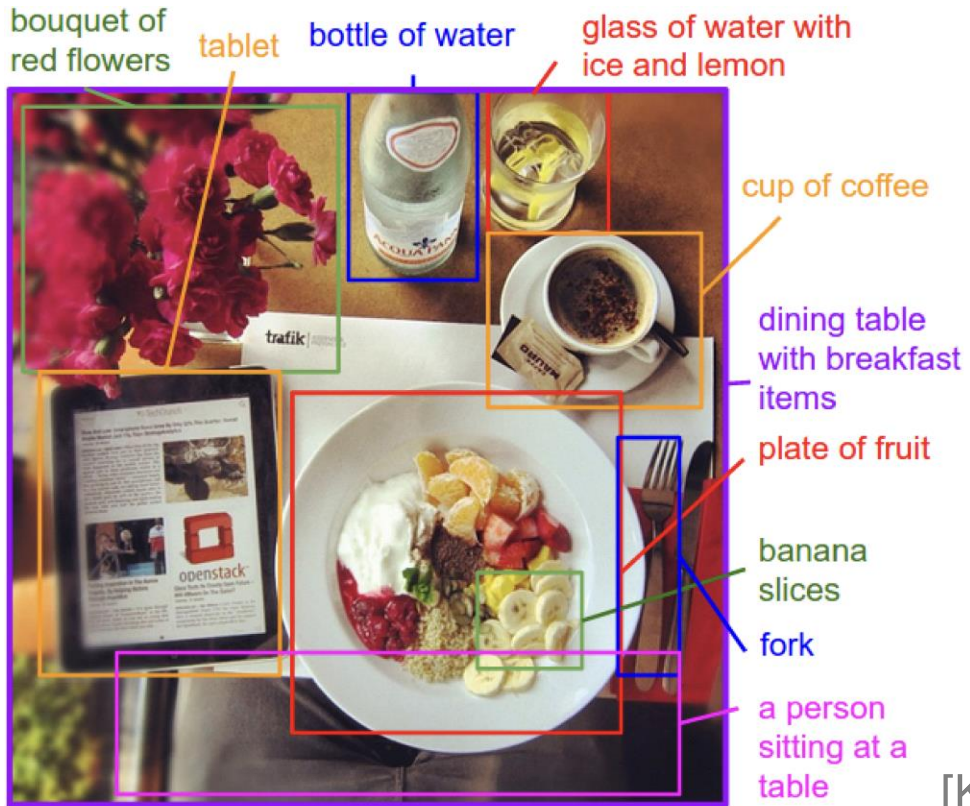
- What about regression?
- Depth estimation: regressing the depth of every pixel



$$\mathcal{L} = \sum_i (y_i - \hat{y}_i)^2$$

Object Detection

- Instead of segmenting per pixel classes we can estimate bounding boxes of objects

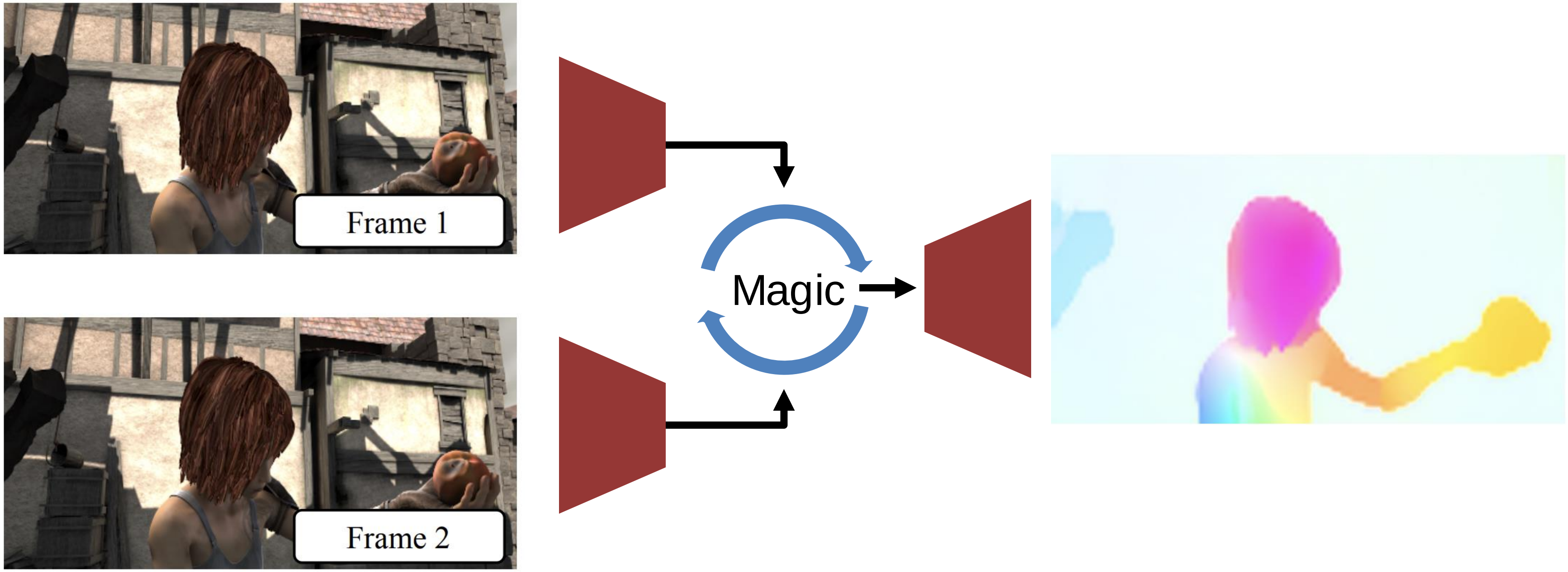


[Karpathy, Li ‘14]

ETH

Optical Flow

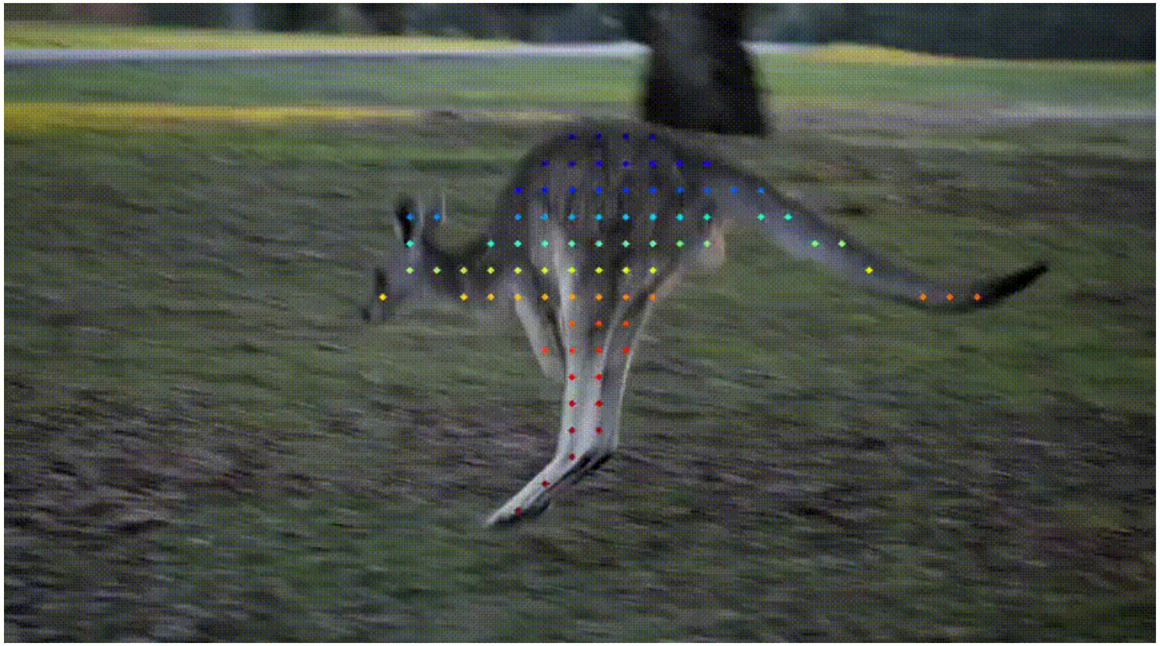
- We can also go beyond individual objects
- Using optical flow networks we can track objects across frames



$$\mathcal{L} = \sum_i ((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2)$$

Tracking

- SOTA methods use different representations for tracking
- OmniMotion represents a video in a 3D canonical volume to track objects



https://omnimotion.github.io

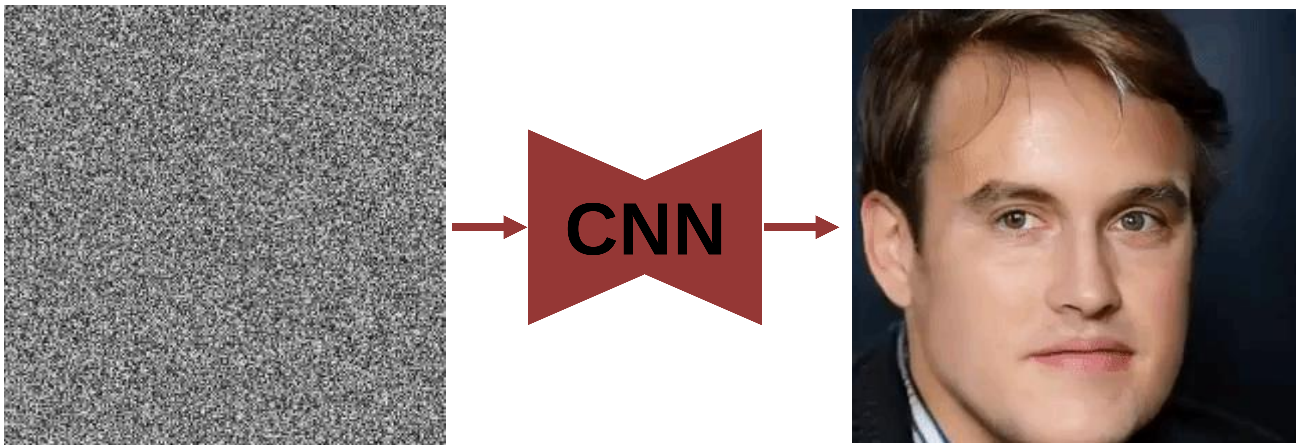
Tracking

- SOTA methods use different representations for tracking
- OmniMotion represents a video in a 3D canonical volume to track objects
- This way it can even track through occlusions

https://omnimotion.github.io

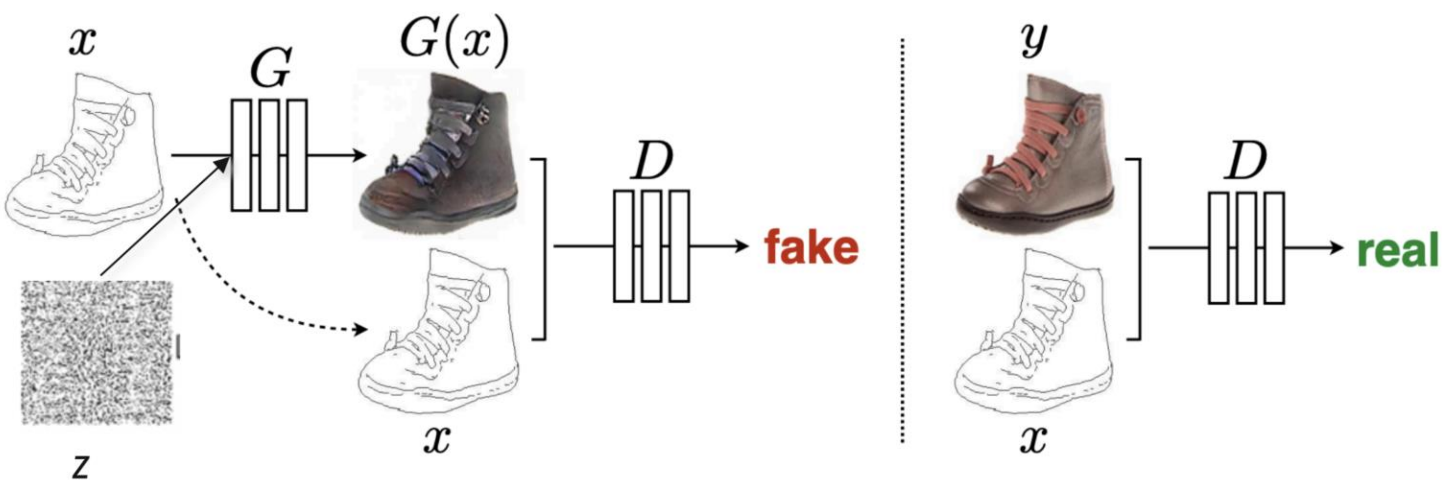
Generative Adversarial Networks

- Previous models were discriminative
- We can also generate data
- Objective functions can get very creative!



Generative Adversarial Networks

- Previous models were discriminative
- We can also generate data
- Objective functions can get very creative!



$$\min_G \max_D V(D,G) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x,z)))]$$

https://arxiv.org/pdf/1611.07004.pdf

ETH

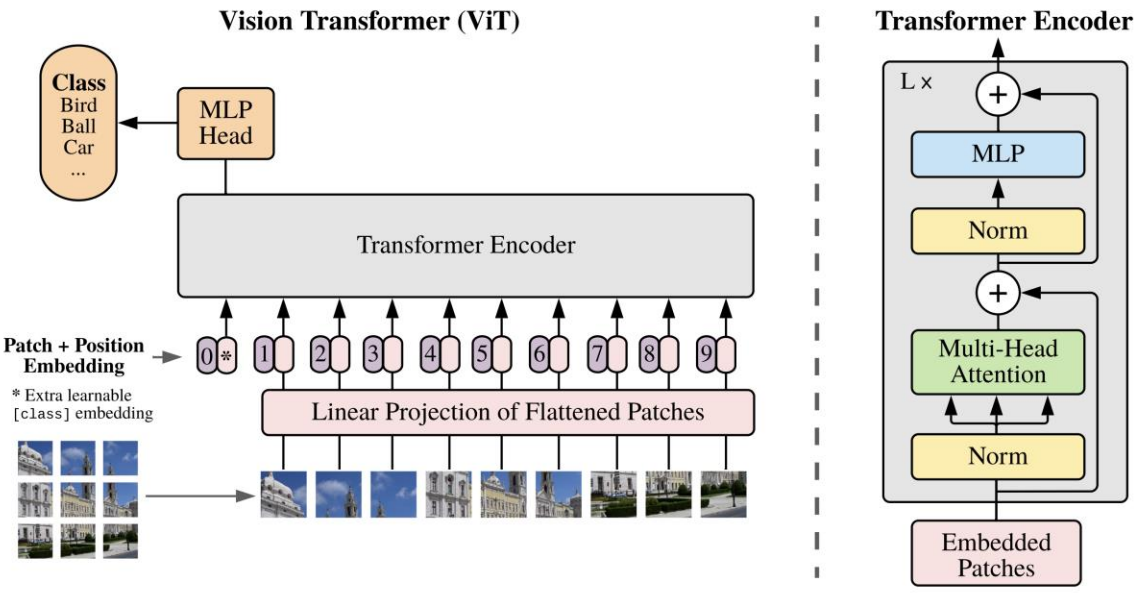
Stable Diffusion

- We can even fuse CNNs with other modalities

https://medium.com/@amritangshu.mukherjee/making-text-to-image-models-smarter-with-controlnet-5f67979ea9a

Where are we now?

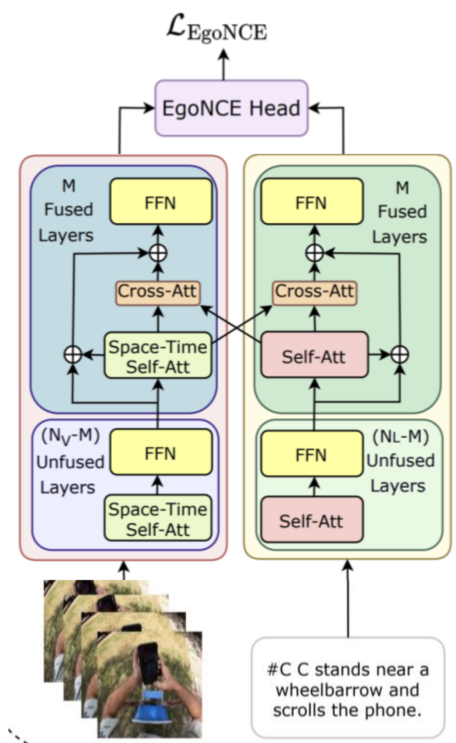
- Vision transformers have a global receptive field



Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale."

Transformers

- Flexible architectures makes fusing modalities easy
- For example, we can use text input to help classify a video



Pramanick, Shraman, et al. "Egovlpv2: Egocentric video-language pre-training with fusion in the backbone." 2023

Summary

- Deep learning minimizes an objective function with data samples (x, y):

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}\big(\boldsymbol{y}, f(\boldsymbol{x}, \theta)\big)$$

- Non-linearity are important for deep networks
- Gradient descent to optimize objective
- Convolutions exploit translation invariance to sparsify model
- Used in many computer vision tasks