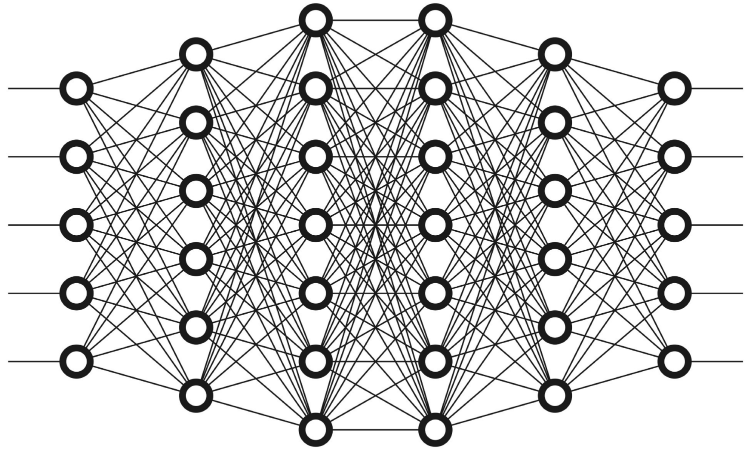
COMP9517
Computer Vision

2024 Term 2 Week 7

Dr Dong Gong



Deep Learning

Overview of Convolutional Neural Networks

Challenges in CV

Consider object detection as an example:

- Variations in viewpoint
- Differences in illumination
- Hidden parts of images
- Background clutter

Linear Classifier for Image Classification

Linear Classifiers

- Image classification with linear classifier

$$\boxed{f(x,W)}_{10\times1} = \underbrace{\boxed{W}}_{10\times3072}\overbrace{\boxed{x}}^{3072\times1} + \boxed{b}\ \ 10\times1$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

→ $f(\mathbf{x},\mathbf{W})$ → **10** numbers giving class scores

↑

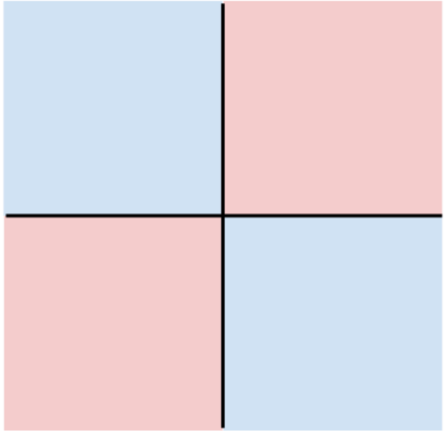
W
parameters
or weights

Linear Classifiers

- Hard cases for a linear classifier.
- Extracting better features (manually) may help but cannot (always) solve the problems.

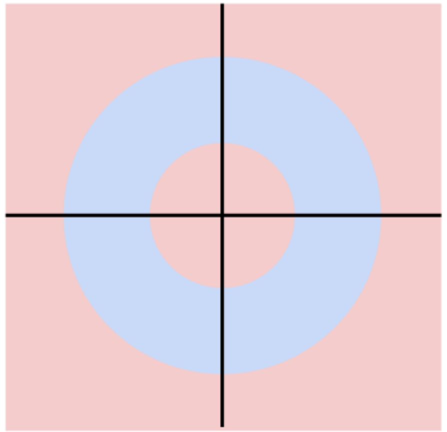
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



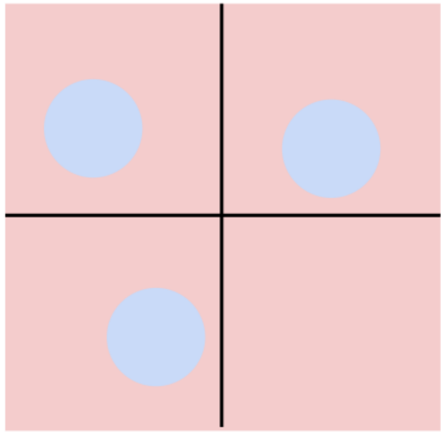
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



From Linear Classifiers to (Non-linear) Neural Networks

Neural Networks

- Starting from the original linear classifier

(**Before**) Linear score function: $f = Wx$

$$x \in \mathbb{R}^D, W \in \mathbb{R}^{C \times D}$$

Neural Networks

- 2 layers

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

(In practice we will usually add a learnable bias at each layer as well)

Neural Networks

- 2 layers
- Also called as fully connected network
- Fully connected (FC) layer

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

“Neural Network” is a very broad term; these are more accurately called “fully-connected networks” or sometimes “multi-layer perceptrons” (MLP)

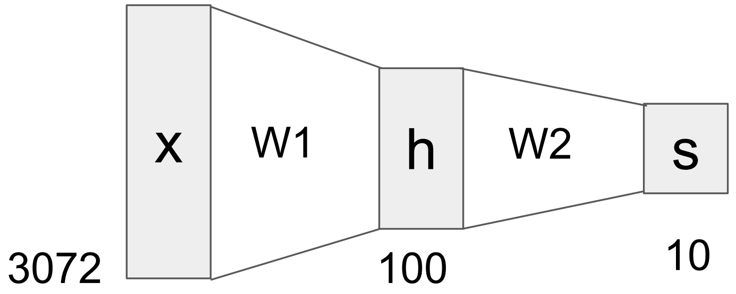
(In practice we will usually add a learnable bias at each layer as well)

Neural Networks

- 2 layers
- Also called as fully connected network
- Fully connected (FC) layer

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$

Neural Networks

- 3 layers

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H_1 \times D}, W_2 \in \mathbb{R}^{H_2 \times H_1}, W_3 \in \mathbb{R}^{C \times H_2}$$

(In practice we will usually add a learnable bias at each layer as well)

Neural Networks

- Activation function
- The function $\max(0, z)$ is called the activation function.

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \boxed{\max(0,} W_1 x)$

- What if without the activation function?

Neural Networks

Activation function

The function $\max(0, z)$ is called the activation function.

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \boxed{\max(0,} W_1 x)$

What if without the activation function?

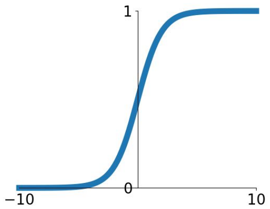
– The model will be linear.

Neural Networks

- Activation functions
  - Non-linear functions

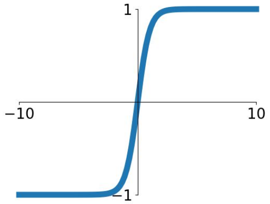
Sigmoid

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



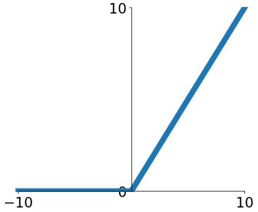
tanh

$\tanh(x)$



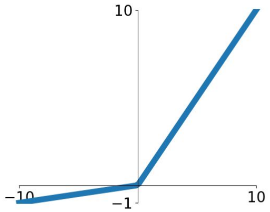
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

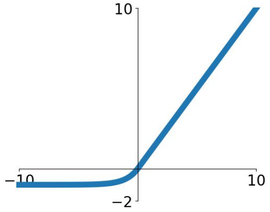


Maxout

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

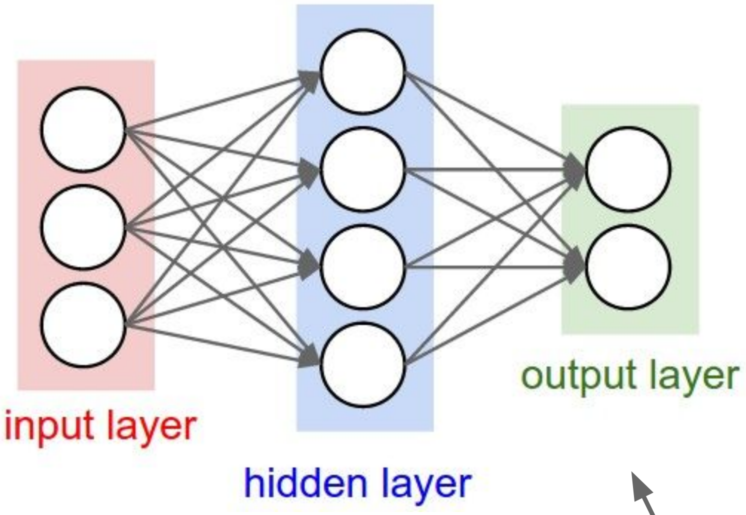
ELU

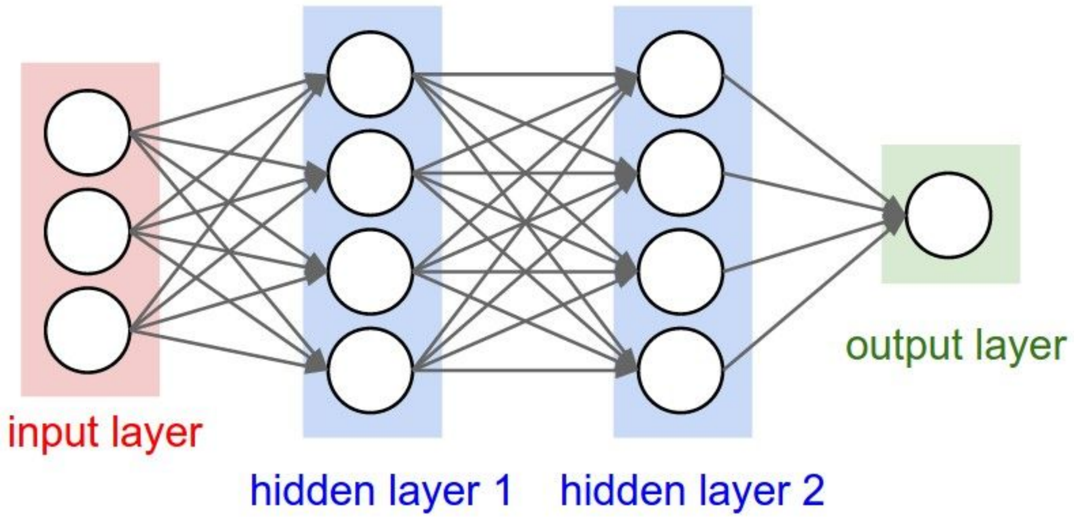
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Networks

Architectures (for MLP)





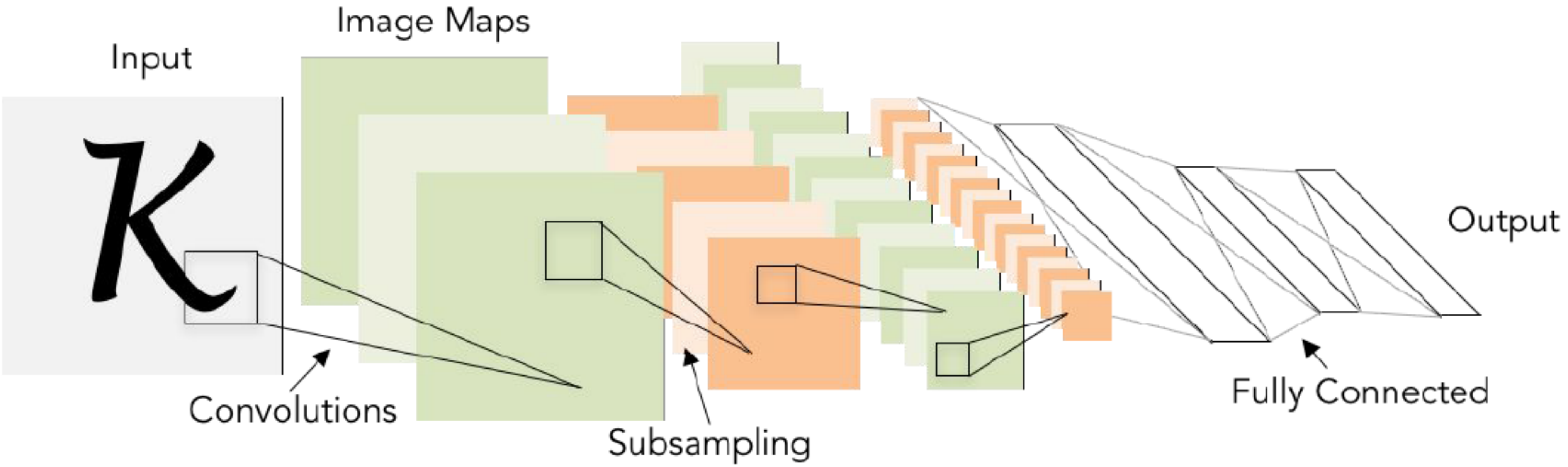
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

"Fully-connected" layers

Neural Networks

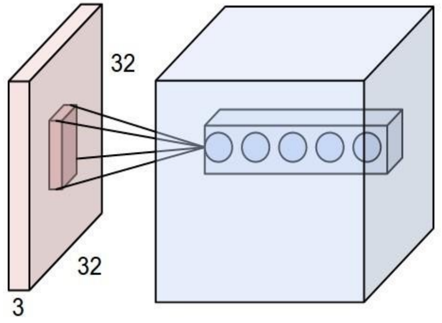
- Architectures (for CNNs)



From Neural Networks to “Deep Learning”

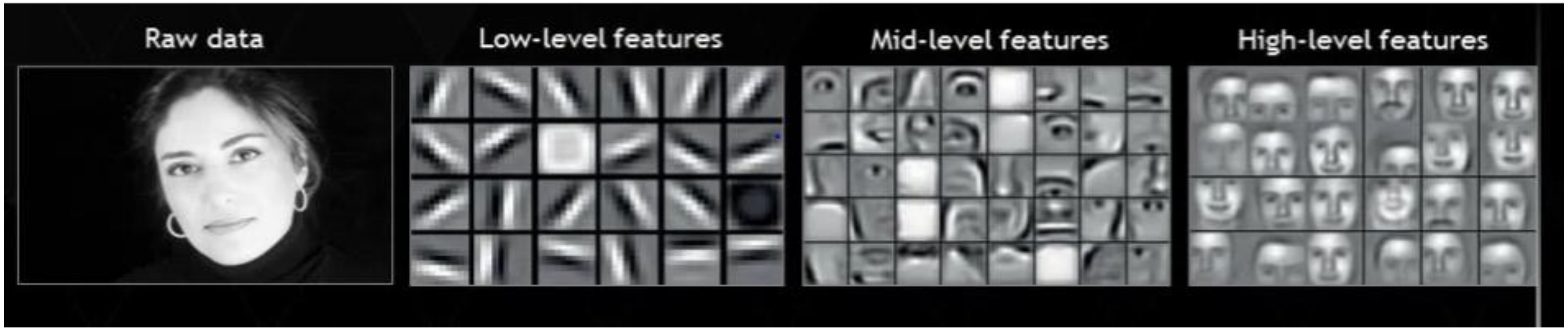
Deep Learning

- Deep learning is a collection of artificial neural network techniques that are widely used at present
- Predominantly, deep learning techniques rely on large amounts of data and deeper learning architectures
- Some well-known paradigms for different types of data and applications:
  - **Convolutional Neural Networks (CNNs) (Week 7)**
  - Recurrent Neural Networks (Week 8)
  - GAN (Week 8)
  - Transformer (Week 8)



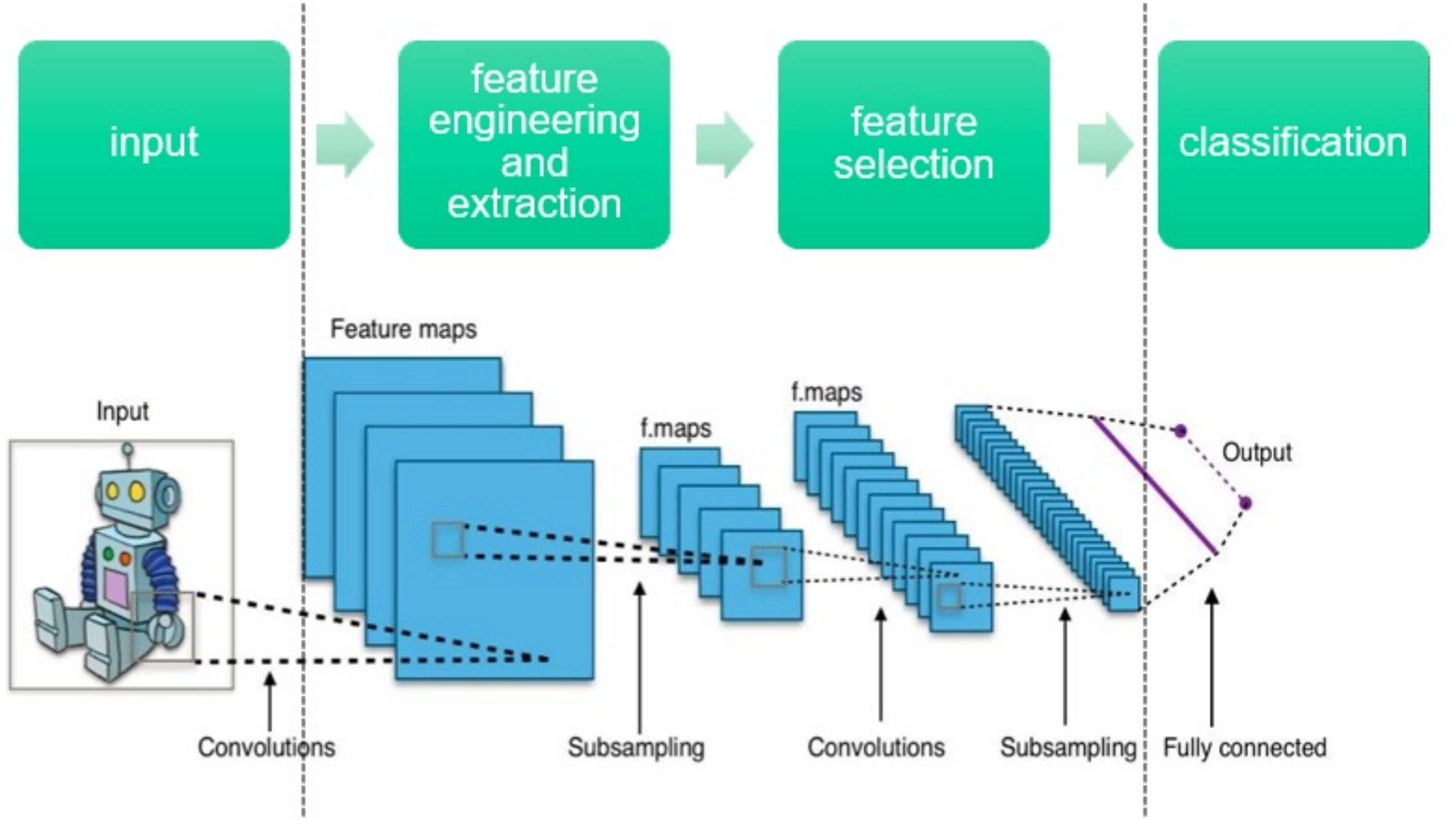
Traditional Approach vs DL

- Convolutional neural networks (CNNs) are a type of DNNs for processing images.
- CNNs can be interpreted as gradually transforming the images into a representation in which the classes are separable by a linear classifier.
- CNNs will try to learn low-level features such as edges and lines in early layers, then parts of objects and then high-level representation of an object in subsequent layers.



http://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/

Traditional Approach vs DL



https://towardsdatascience.com/convolutional-neural-networks-for-all-part-i-cdd282ee7947

From Neural Networks to “Deep Learning”

Core ideas go back many decades

Core ideas go back many decades!

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized
letters of the alphabet

Frank Rosenblatt, ~1957: Perceptron

From Neural Networks to "Deep Learning"

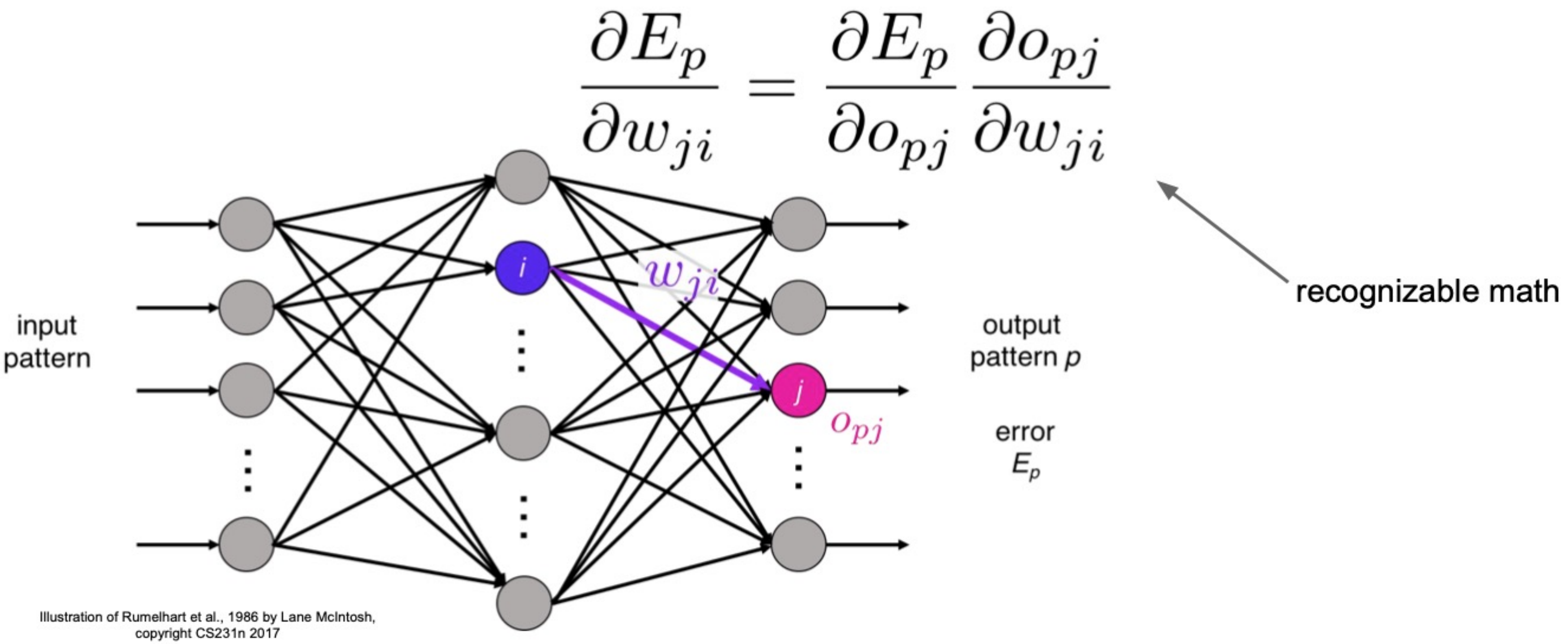


Illustration of Rumelhart et al., 1986 by Lane McIntosh, copyright CS231n 2017

Rumelhart et al., 1986: First time back-propagation became popular

From Neural Networks to "Deep Learning"

First strong results

Acoustic Modeling using Deep Belief Networks
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

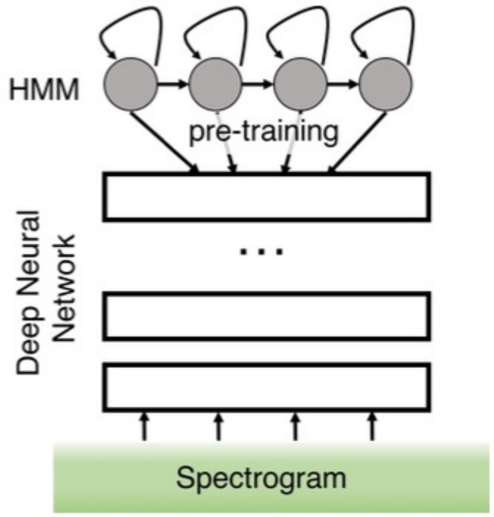
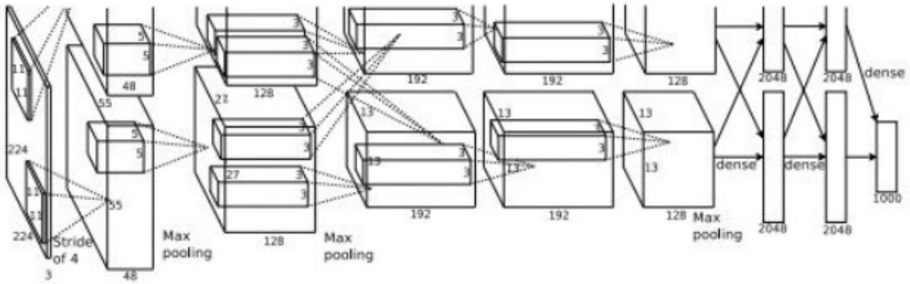
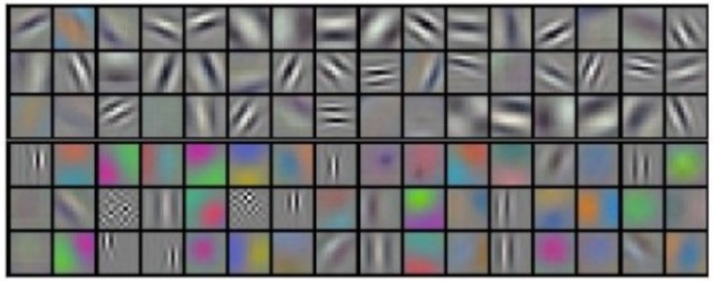


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

Imagenet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012





From Neural Networks to "Deep Learning"

First strong results

Acoustic Modeling using Deep Belief Networks
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

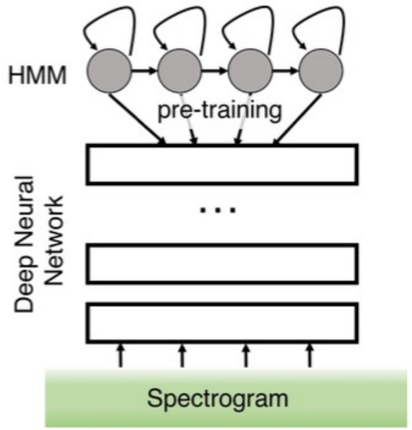
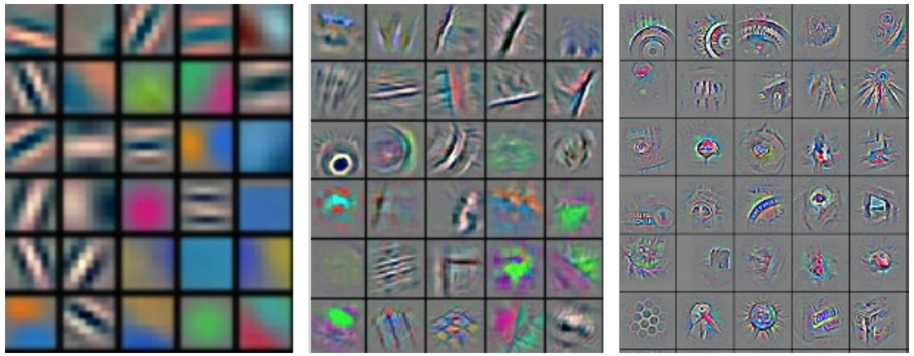
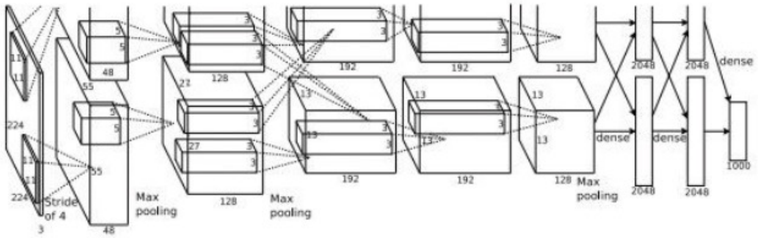


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017



Visual features extracted in different layers in CNN

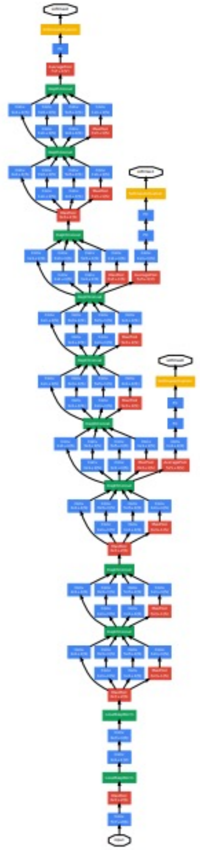
Imagenet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



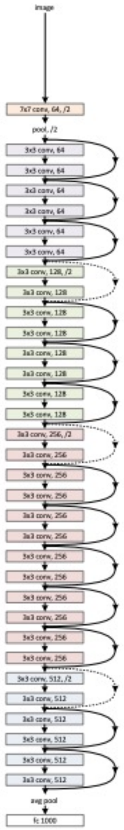
Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

From Neural Networks to “Deep Learning”

Simonyan and Zisserman, 2014



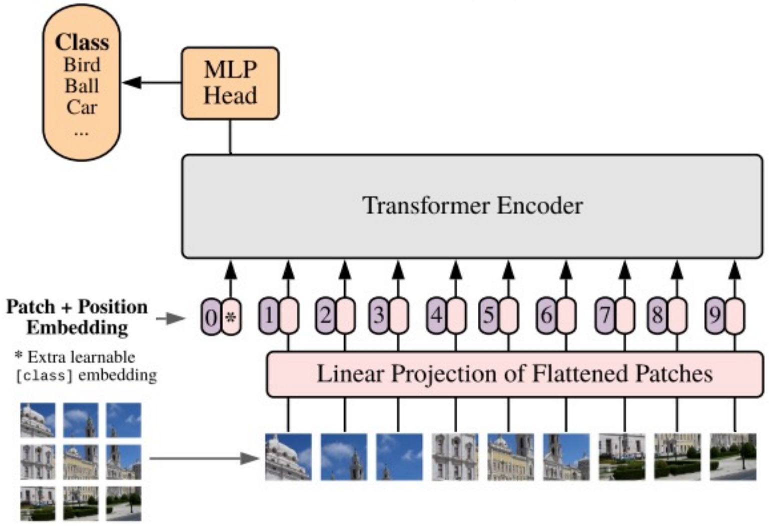
GoogleNet (Szegedy et al., 2015)



ResNet (He et al., 2015)

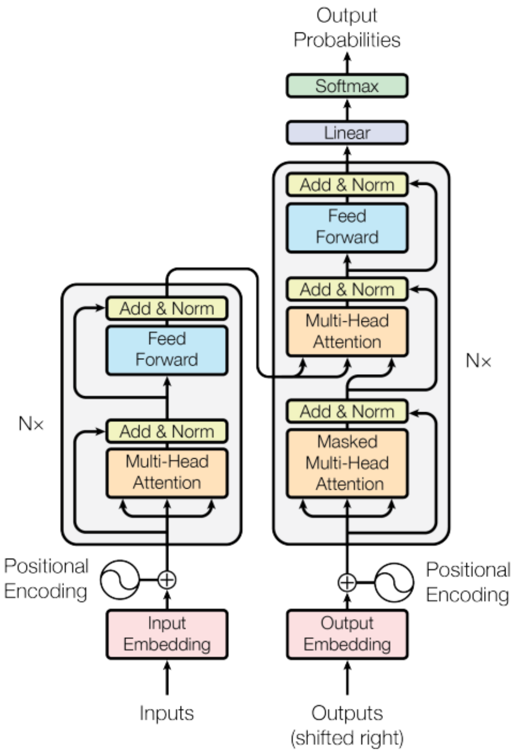
From Neural Networks to “D

Vision Transformer (ViT)



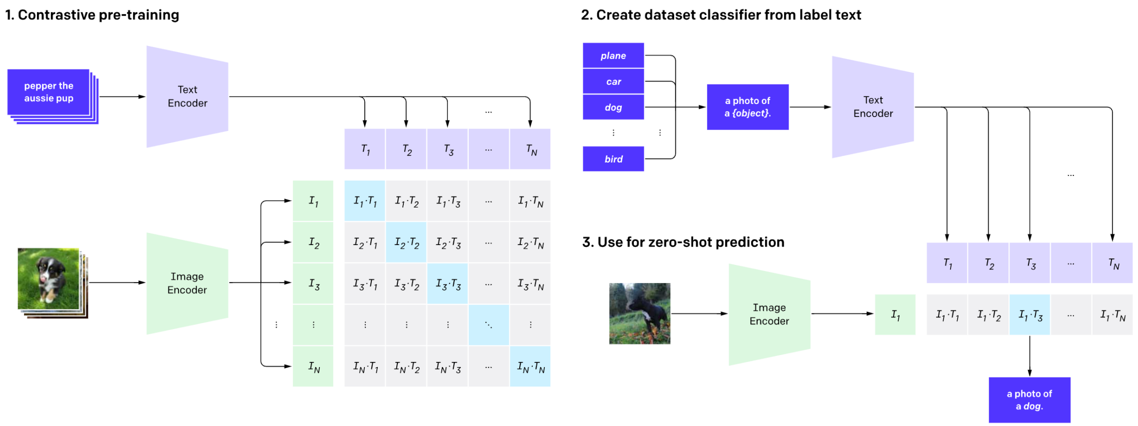
Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Transformer



(Vaswani et al., 2017)

CLIP (Contrastive Language–Image Pre-training)



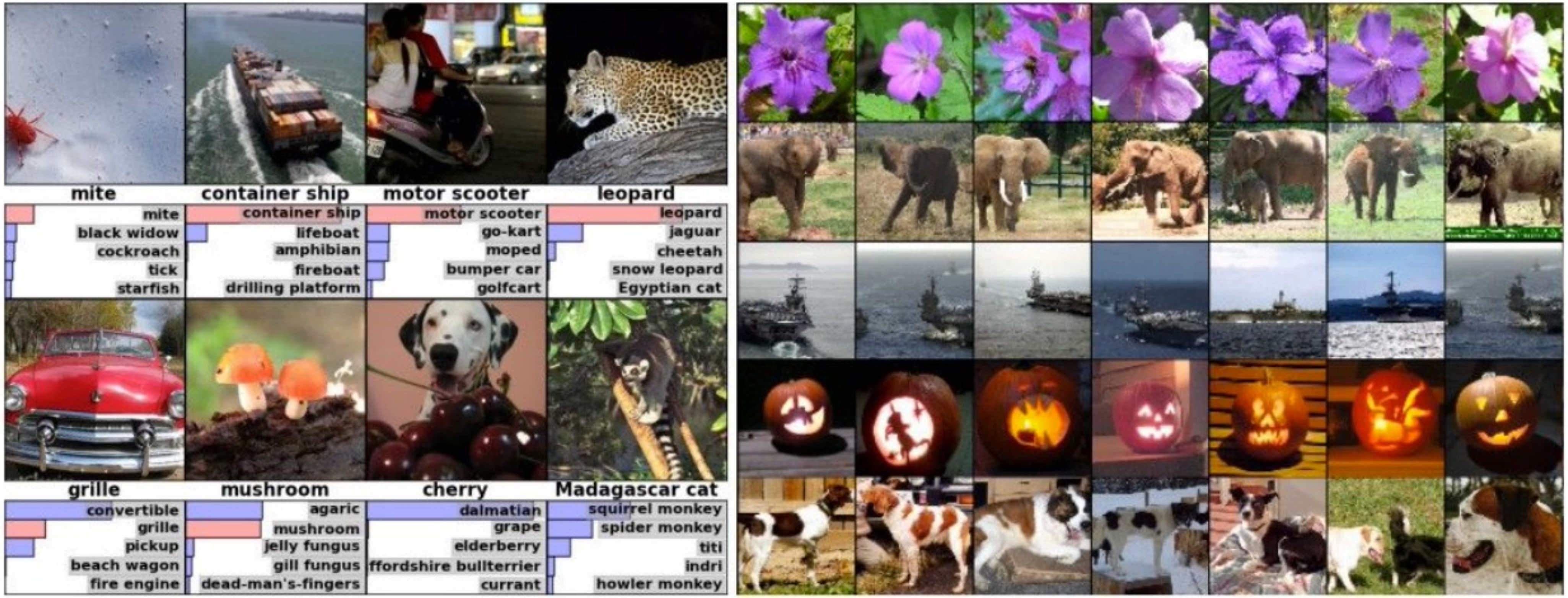
https://openai.com/research/clip

From Neural Networks to “Deep Learning”

- DL is everywhere

Classification

Retrieval

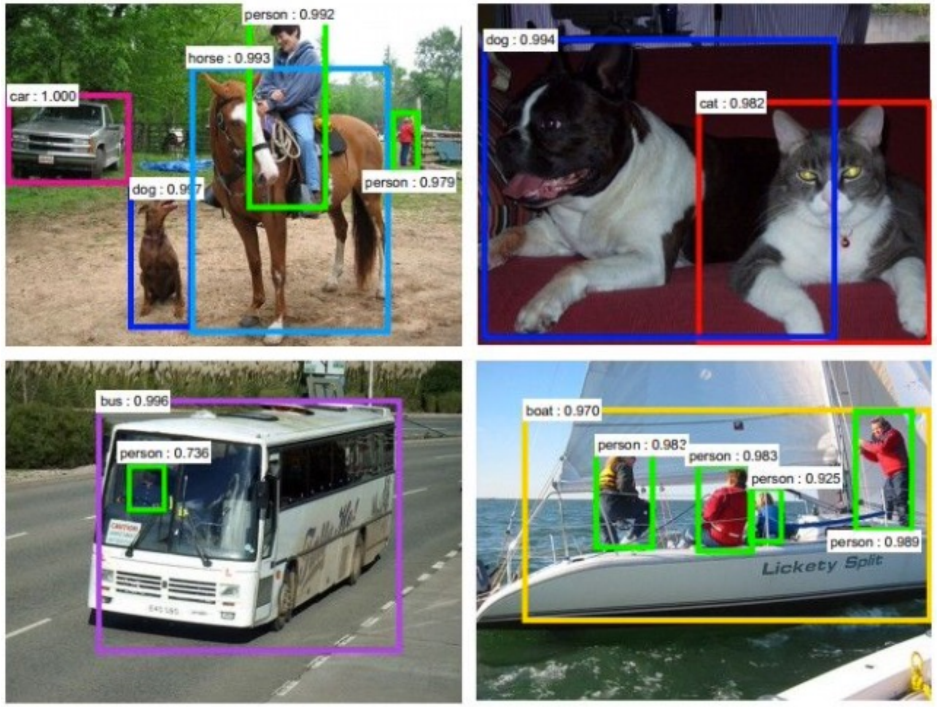


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

From Neural Networks to “Deep Learning”

- DL is everywhere

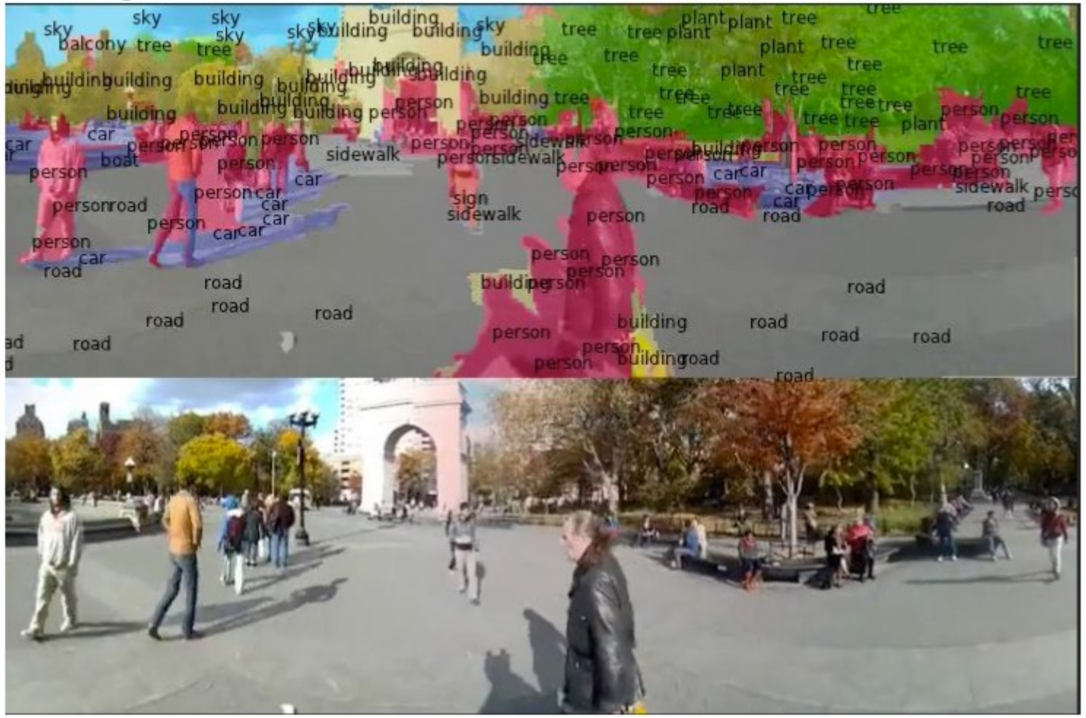
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

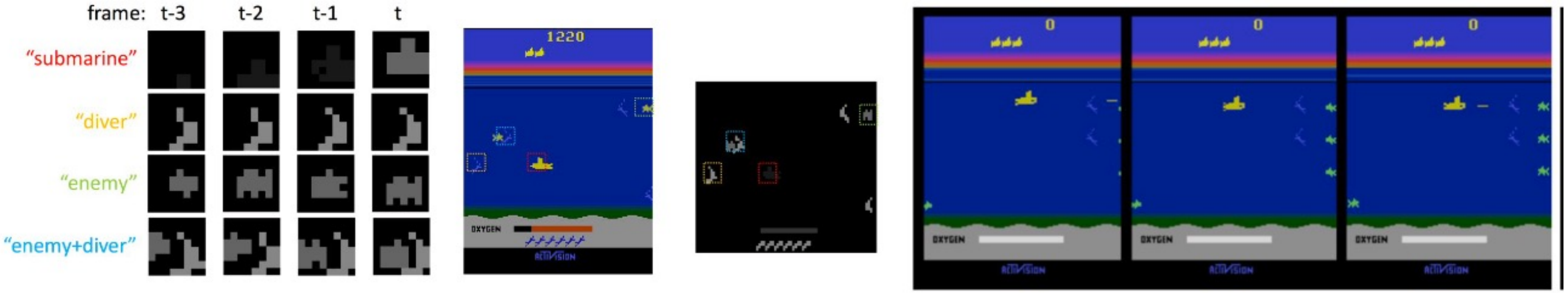
[Farabet et al., 2012]

From Neural Networks to “Deep Learning”

- DL is everywhere

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]

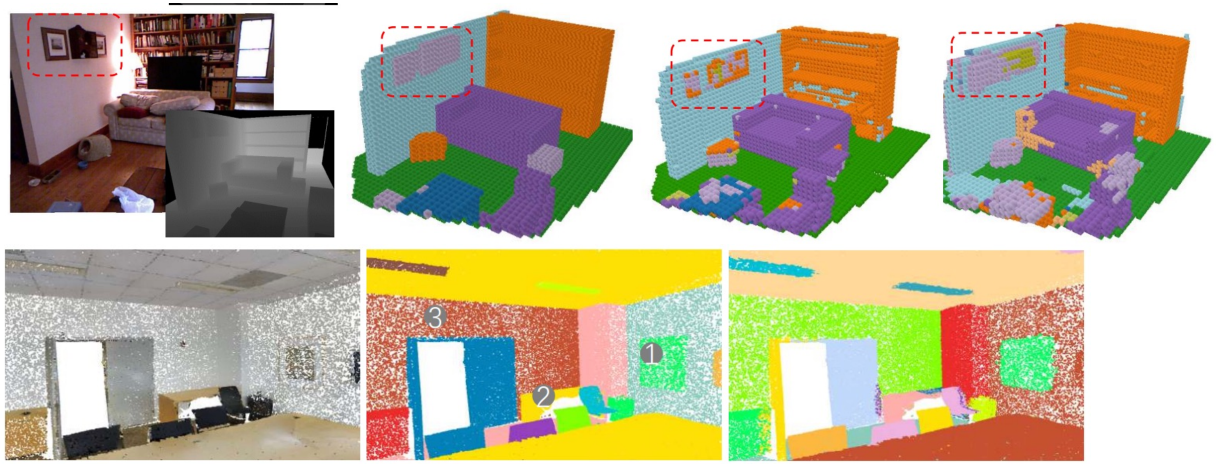


[Guo et al. 2014]

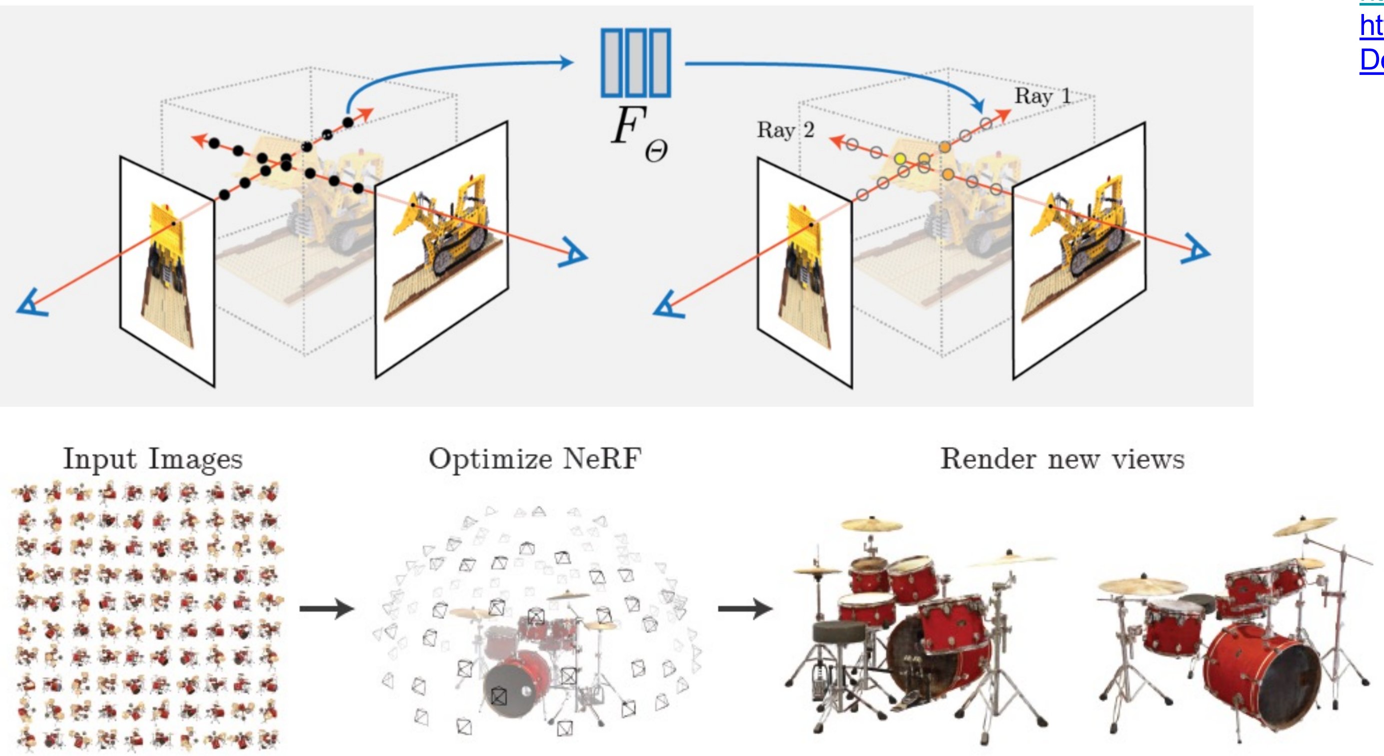
Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

From Neural Networks to '

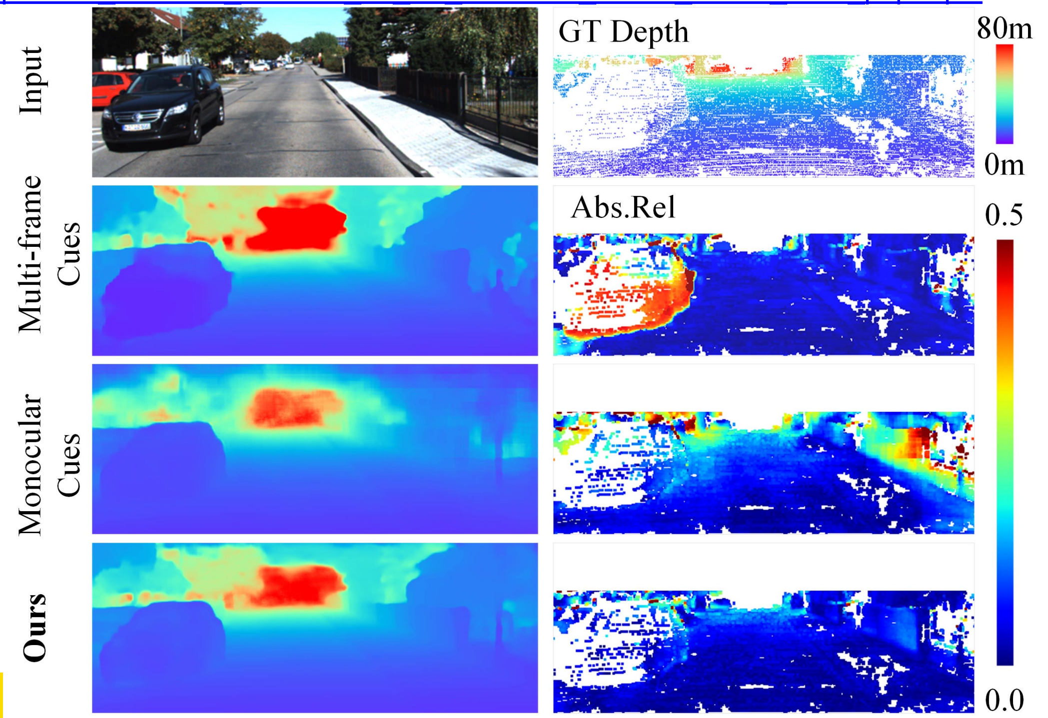
- DL is everywhere



3D vision understanding
https://arxiv.org/pdf/2001.01349.pdf
http://openaccess.thecvf.com/content_CVPR_2019/papers/Li_RGBD_Based_Dimensional_Decomposition_Residual_Network_for_3D_Semantic_Scene_CVPR_2019_paper.pdf



Neural Radiance Fields (NeRF) for 3D vision
https://www.matthewtancik.com/nerf



Deep learning for depth estimation
https://ruili3.github.io/dymultidepth/index.html

From Neural Networks to “Deep Learning”

- DL is everywhere

Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010

Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;

Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016
Gatys et al, “Controlling Perceptual Factors in Neural Style Transfer”, CVPR 2017

https://github.com/donggong1/learn-optimizer-rgdn
https://donggong1.github.io/blur2mflow.html

UNSW SYDNEY

From Neural Networks to “Deep Learning”

- DL is everywhere

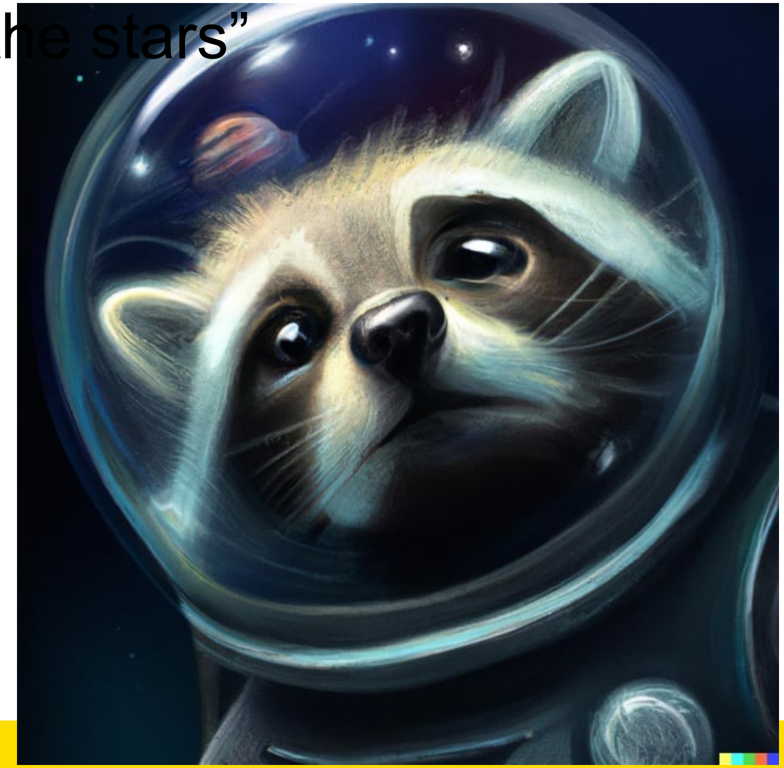
Image Captioning. Vinyals et al, 2015 Karpathy and Fei-Fei, 2015

Vision question answering (VQA)

From Neural Networks to “Deep Learning”

Ramesh et al, “DALL·E: Creating Images from Text”, 2021. https://openai.com/blog/dall-e/

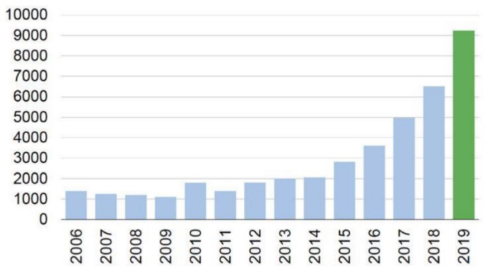
“A raccoon astronaut with the cosmos reflecting on the glass of his helmet dreaming of the stars”



Generated by DALL·E 2

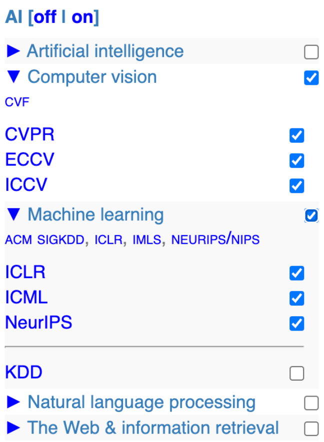
The IEEE/CVF Conference on Computer Vision and Pattern Recognition

CVPR Attendance Trend



>9k submissions, 2,360 accepted papers

ICCV23
PARIS



3 major international CV conferences: CVPR, ICCV, ECCV; and others

Top machine learning conferences with CV research: NeurIPS, ICML, ICLR …

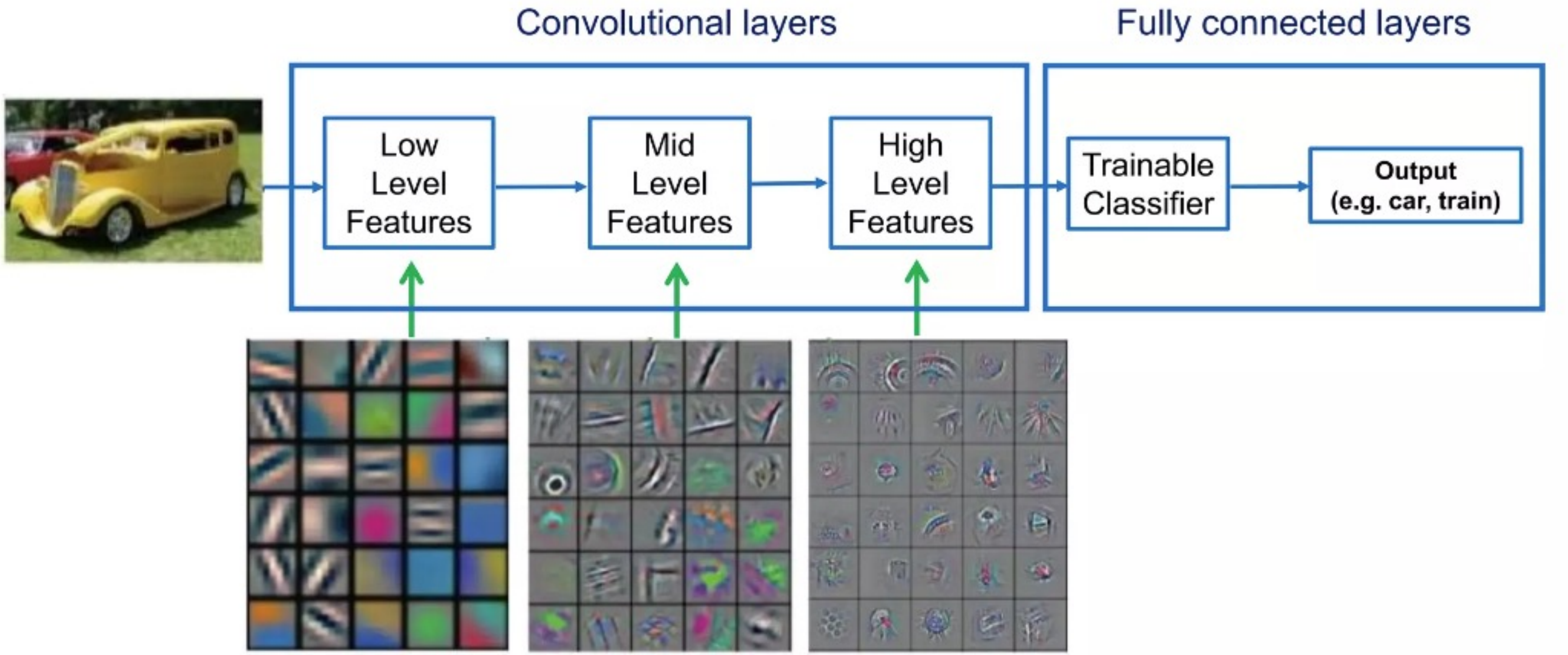
https://csrankings.org/#/index?vision&mlmining&australasia

Convolutional Neural Network (CNN), from MLP to CNN

What are CNNs?

- Essentially neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- Convolutional layer acts as a feature extractor that extracts feature of the inputs as edges, corners or endpoints.

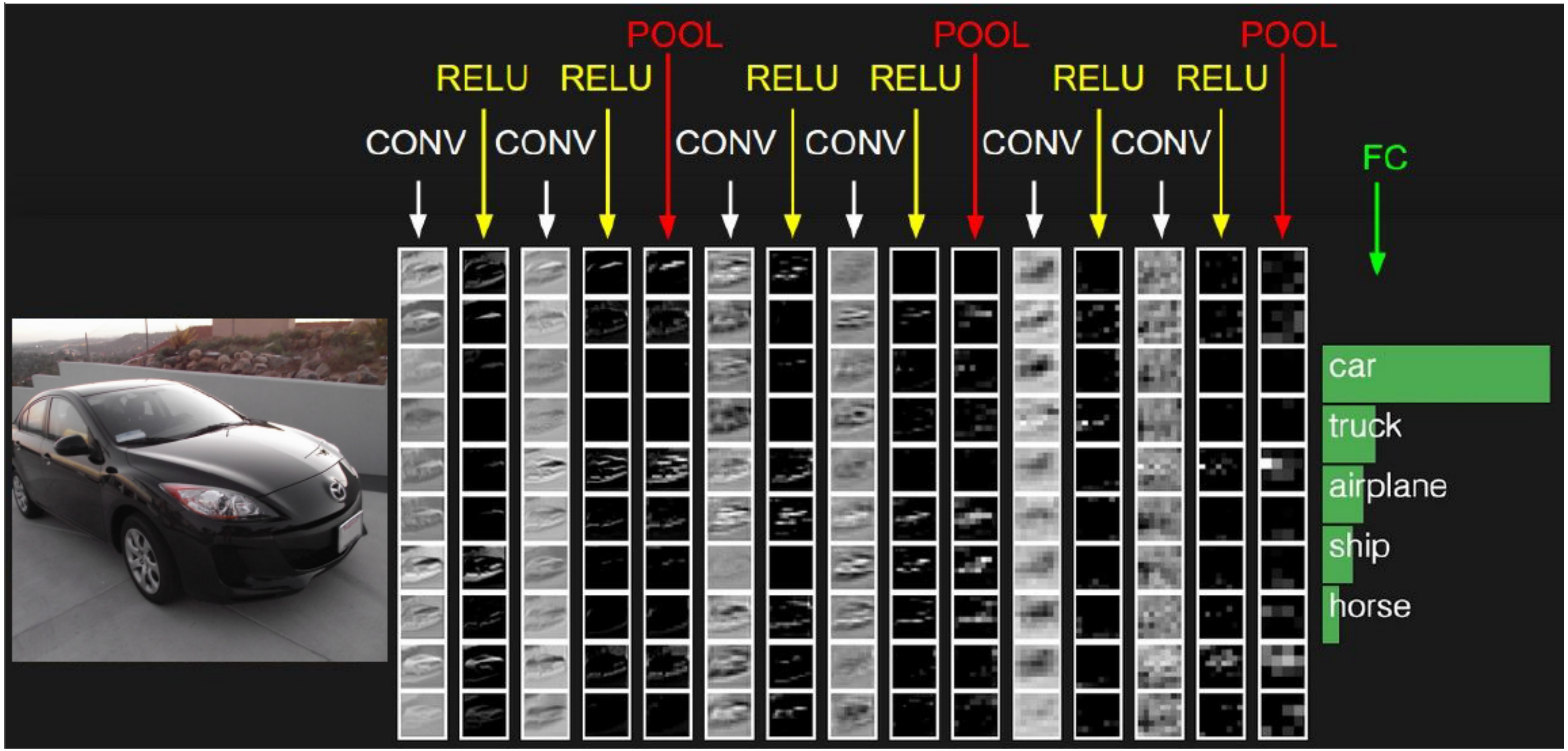
CNN- What do they learn?



https://www.slideshare.net/NirthikaRajendran/cnn-126271677

[From recent Yann LeCun slides]

CNN-Components

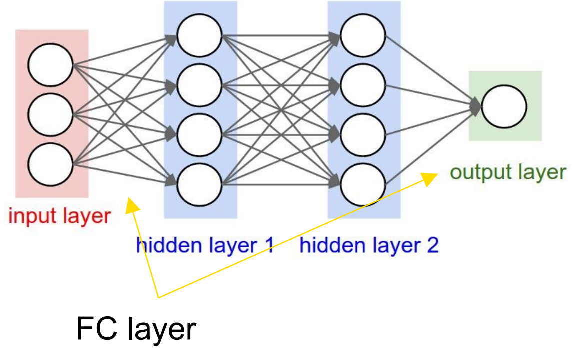


CNNs

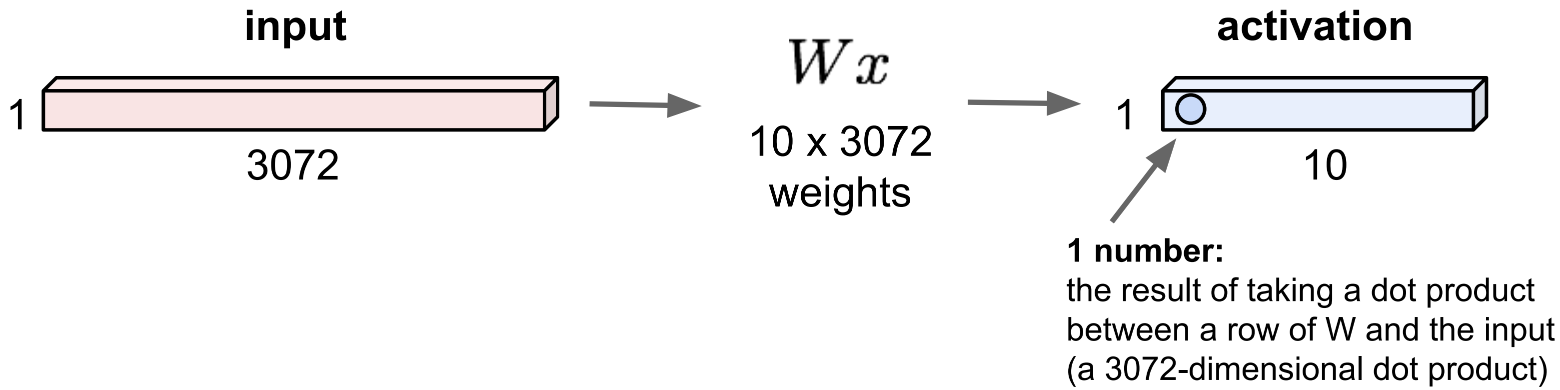
- CNNs are made up of neurons with learnable weights, as other to regular Neural Networks
- CNN architecture assumes that inputs are images
  - Using specific assumptions for images
  - So that we have local features
  - Which allows us to
    - encode certain properties in the architecture that makes the forward pass more efficient and
    - significantly reduces the number of parameters needed for the network

Convolutional Neural Networks (CNNs)

- Recap: fully connected (FC) layer
- A linear model, not CNNs.
- A component of CNNs



32x32x3 image -> stretch to 3072 x 1



Convolution operator parameters

- Filter size
- Padding
- Stride
- Dilation
- Activation function

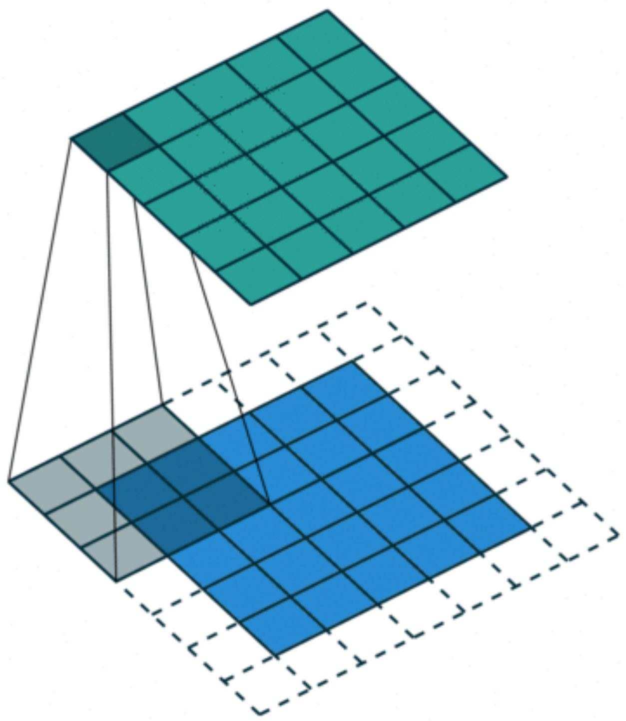
Filter size

- Filter size can be 5 by 5, 3 by 3, and so on
- Larger filter sizes should be avoided in many cases (not always!)
  - As learning algorithm needs to learn filter values (weights)
- Odd sized filters are used more often than even sized filters (not always!)
  - Nice geometric property of all input pixels being around output pixel

Padding

- After applying 3 by 3 filter to 4 by 4 image, we get a 2 by 2 image – Size of the image has gone down
- If we want to keep image size the same, we can use padding
  - We pad input in every direction with 0's before applying filter
  - If padding is 1 by 1, then we add 1 zero in evey direction
  - If padding is 2 by 2, then we add 2 zeros in every direction, and so on

3 by 3 filter with padding of 1

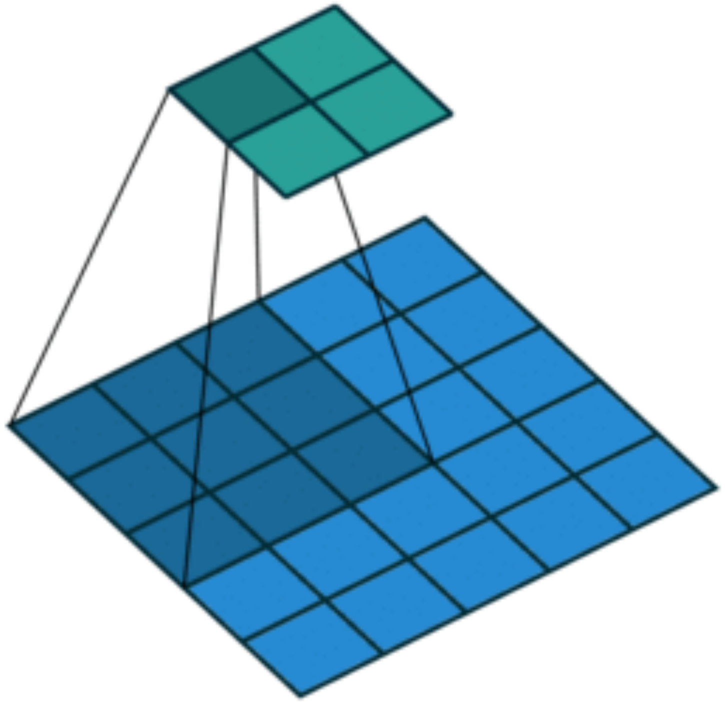


https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/slides-plain.html

Stride

- How many pixels we move filter to the right/down is stride
- Stride 1: move filter one pixel to the right/down
- Stride 2: move filter two pixels to the right/down

3 by 3 filter with stride of 2

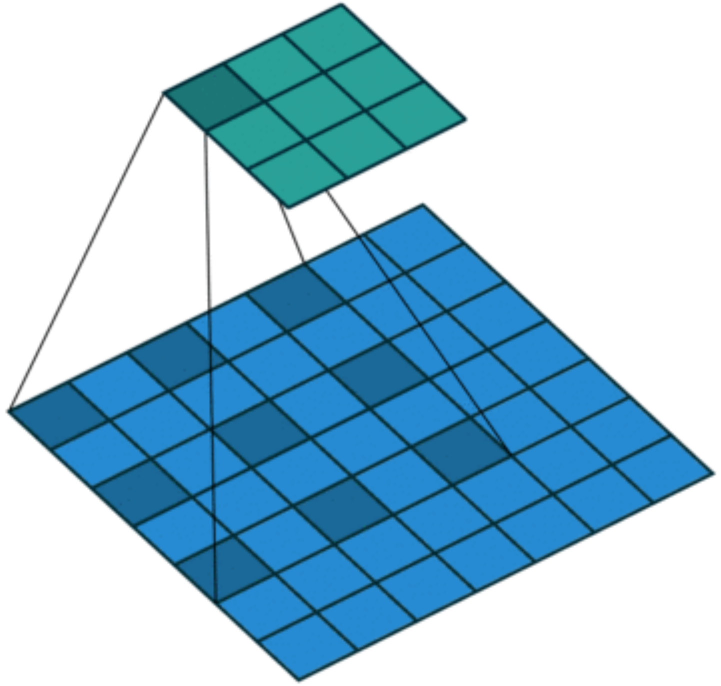


https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/slides-plain.html

Dilation

- When we apply 3 by 3 filter, output affected by pixels in 3 by 3 subset of image
- Dilation: To have a larger receptive field (portion of image affecting filter's output)
- If dilation set to 2, instead of contiguous 3 by 3 subset of image, every other pixel of a 5 by 5 subset of image affects output

3 by 3 filter with dilation of 2



https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/slides-plain.html

Activation function

- After filter applied to whole image, apply activation function to output to introduce non-linearity
- Preferred activation function in CNN is ReLU
- ReLU leaves outputs with positive values as is, replaces negative values with 0

0	1	-4
0	2	0
-1	4	3

Filter output

-->

0	1	0
0	2	0
0	4	3

Filter output after ReLU

Single channel 2D convolution

0	2	1	0	1
0	1	2	1	0
1	0	2	0	0
1	0	0	1	1
0	1	1	2	2

Input Vector

0	1	-4
0	2	0
-1	4	3

Filter

5		

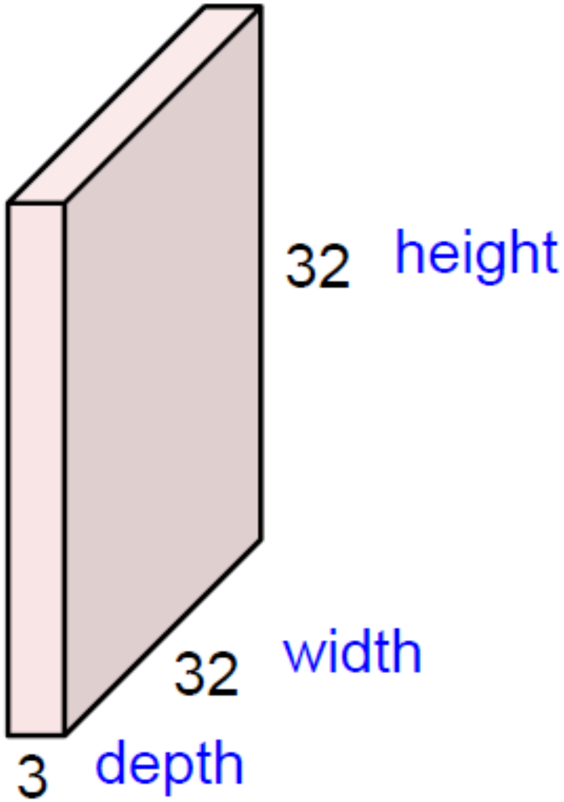
Output

$$\begin{aligned}= &0 \times 0 + 2 \times 1 + 1 \times -4 + \\ &0 \times 0 + 1 \times 2 + 2 \times 0 + \\ &1 \times -1 + 0 \times 4 + 2 \times 3 = 5\end{aligned}$$

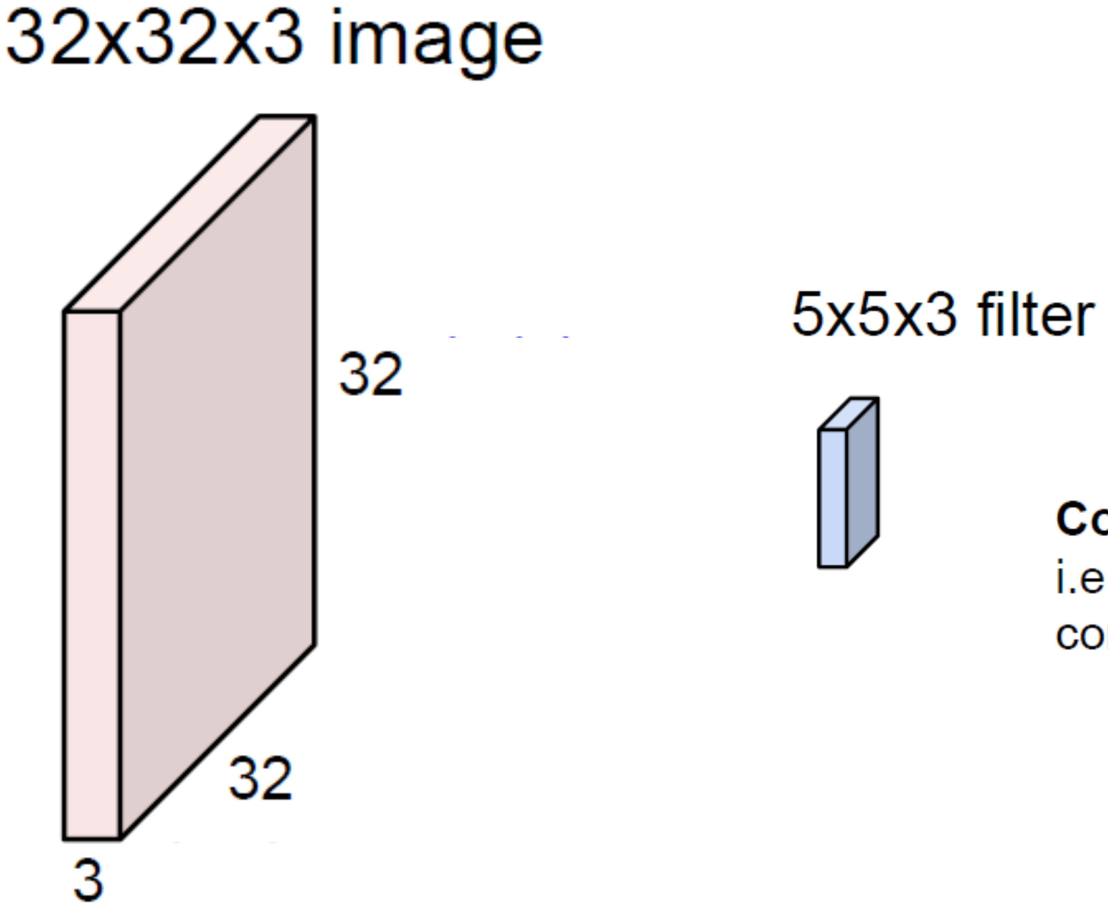
https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/slides-plain.html

CNN: Convolutional Layer

32x32x3 image -> preserve spatial structure

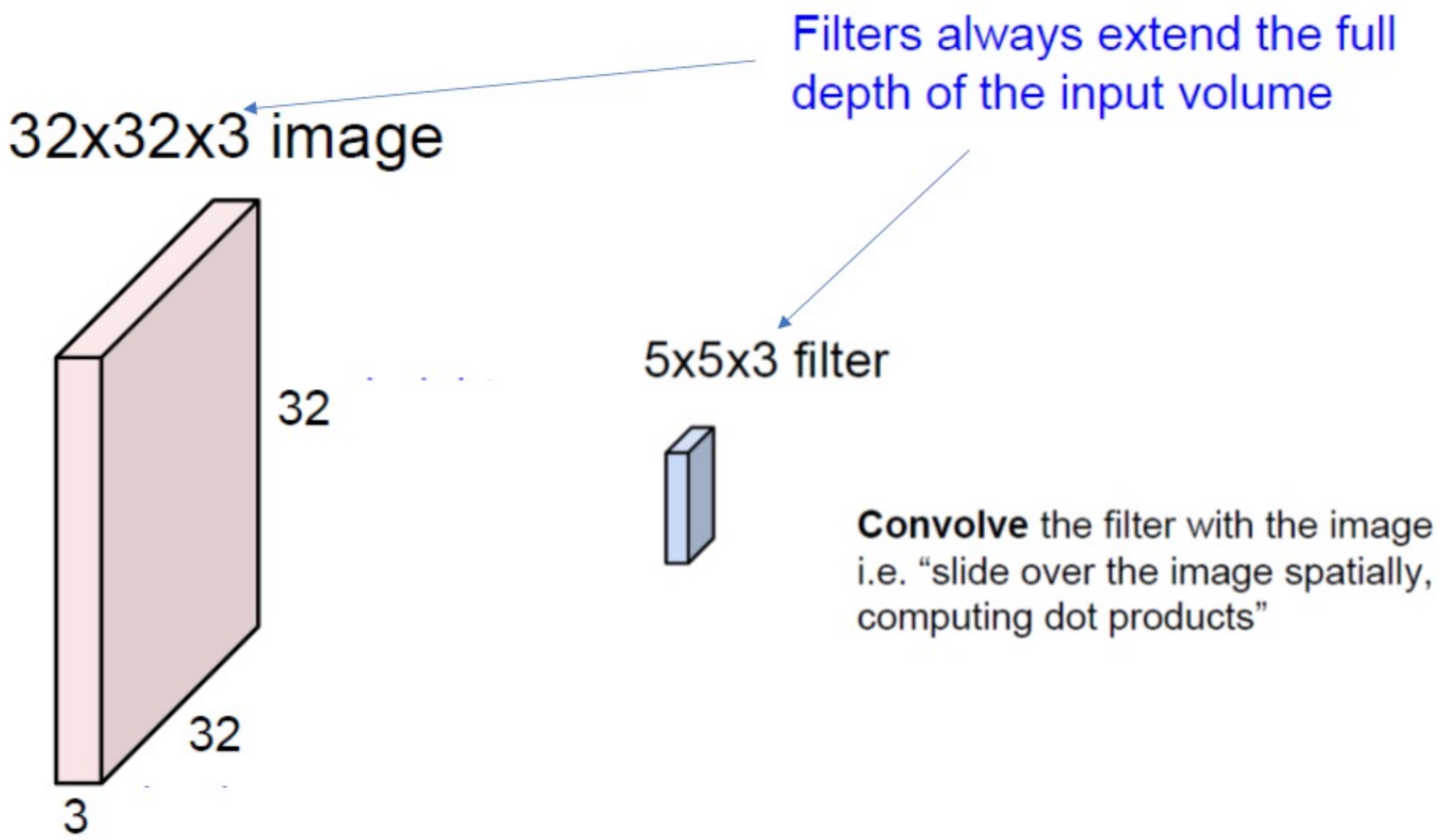


CNN: Convolutional Layer

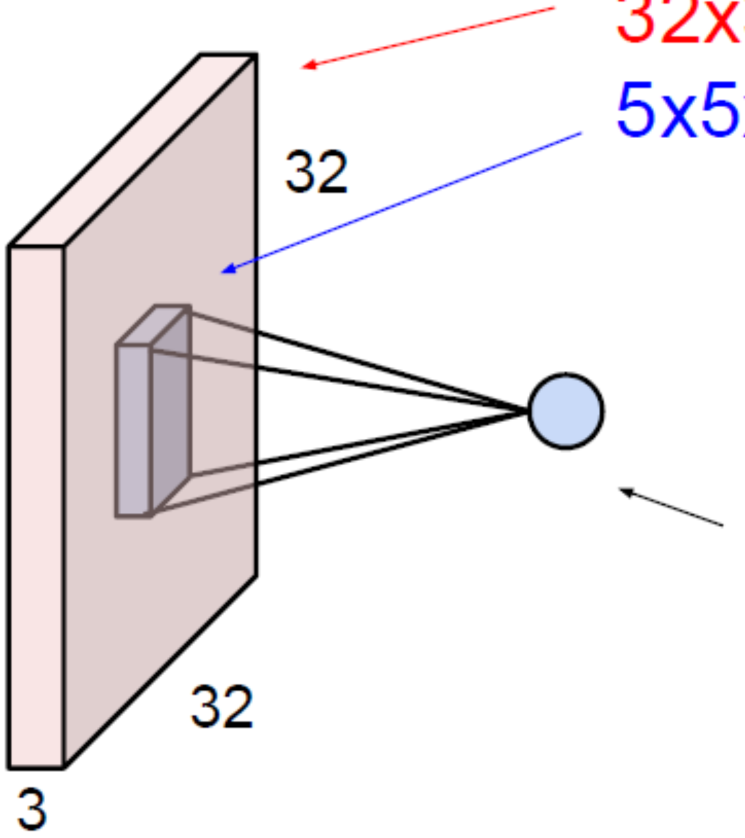


Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

CNN: Convolutional Layer

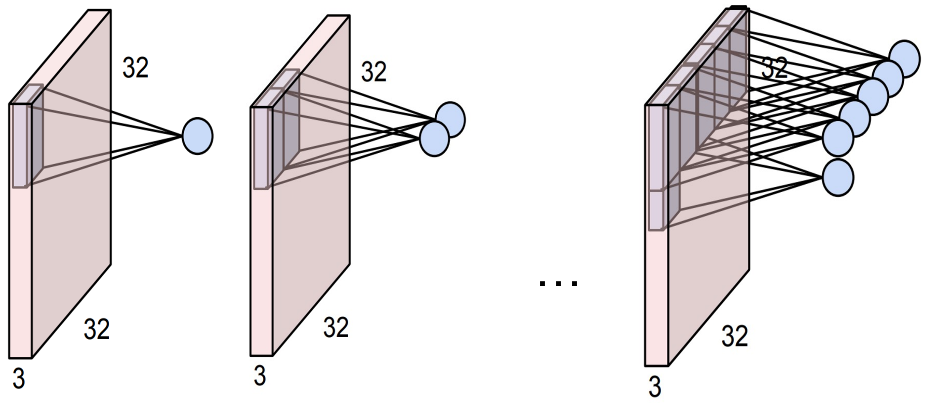


CNN: Convolutional Layer



32x32x3 image

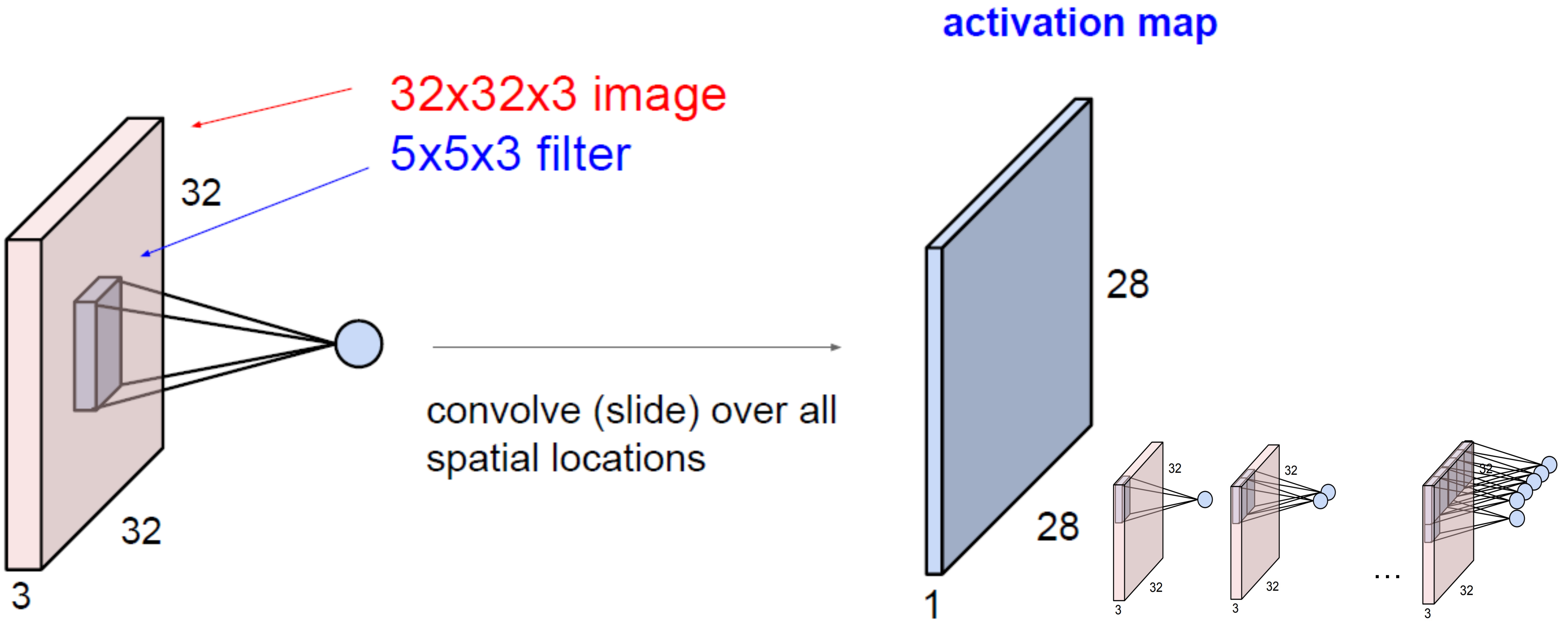
5x5x3 filter w



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

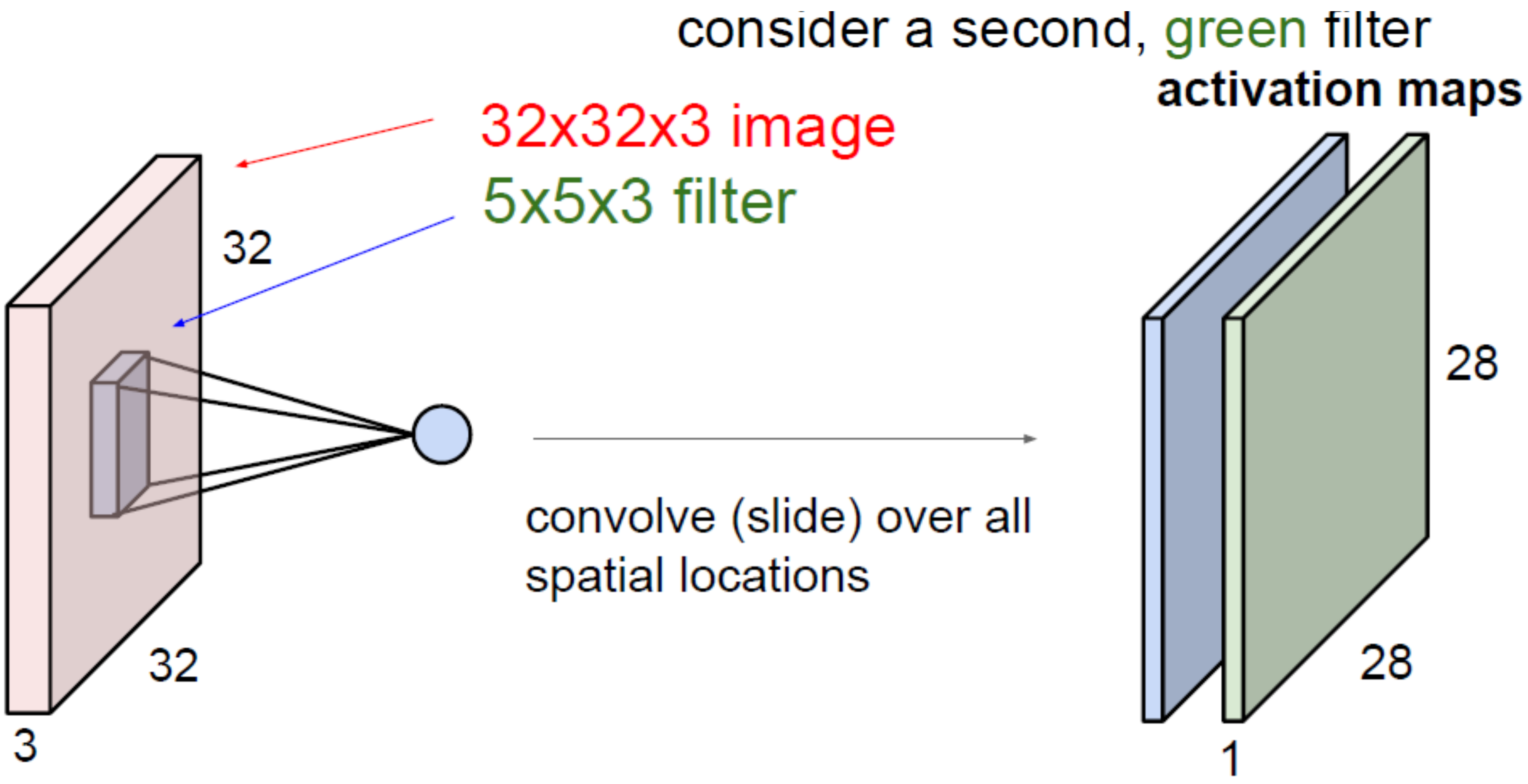
CNN: Convolutional Layer



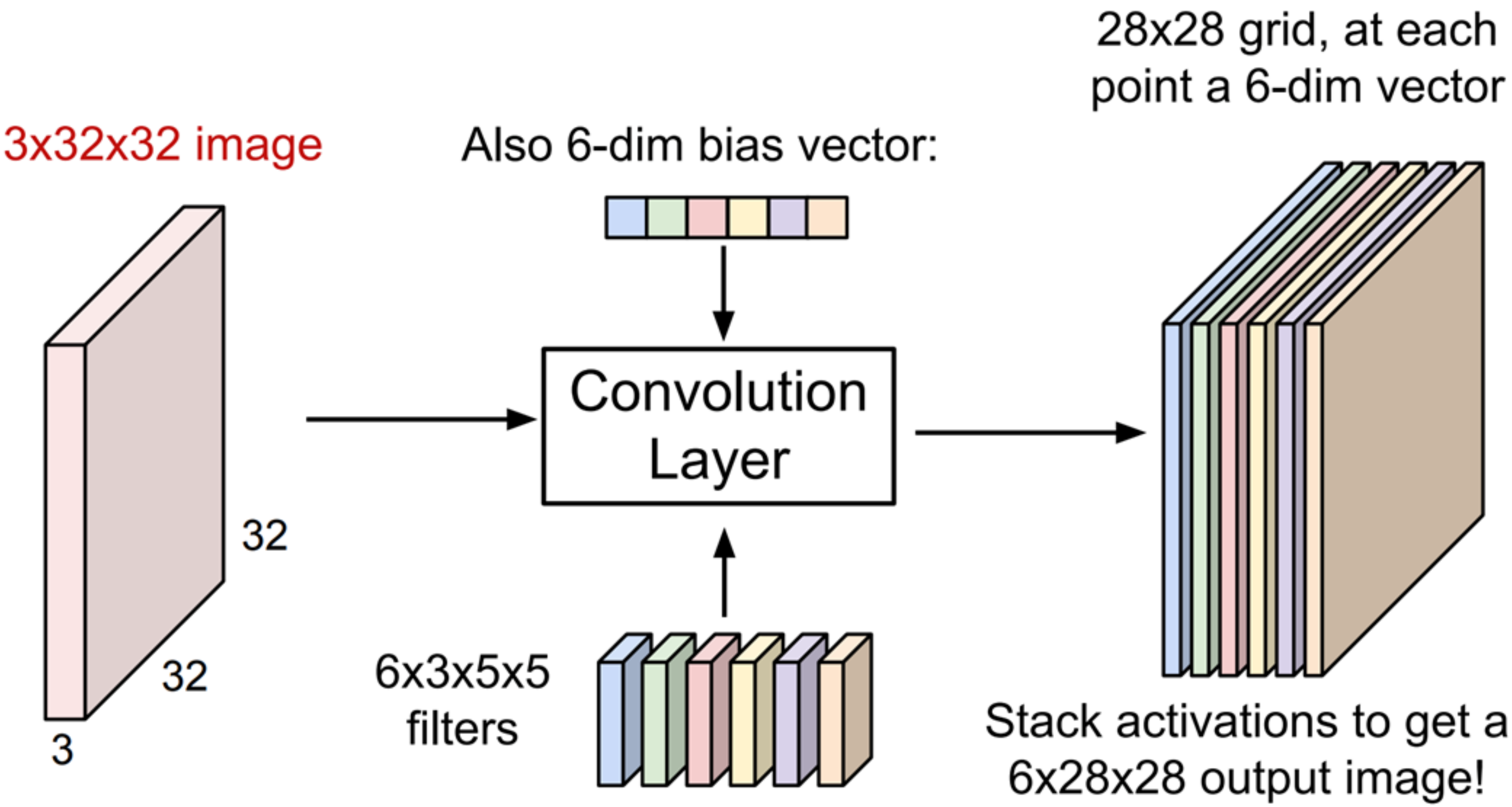
CNN: Convolutional Layer

- The output of the Conv layer can be interpreted as holding neurons arranged in a 3D volume.
- The Conv layer's parameters consist of a set of **learnable filters**. Every filter is small spatially (along width and height), but extends through the full depth of the input volume.
- During the forward pass, each filter is slid (convolved) across the width and height of the input volume, producing a 2-dimensional activation map of that filter.
- Network will learn filters (via backpropagation) that activate (through the activation function) when they see some specific type of feature at some spatial position in the input.

CNN: Convolutional Layer



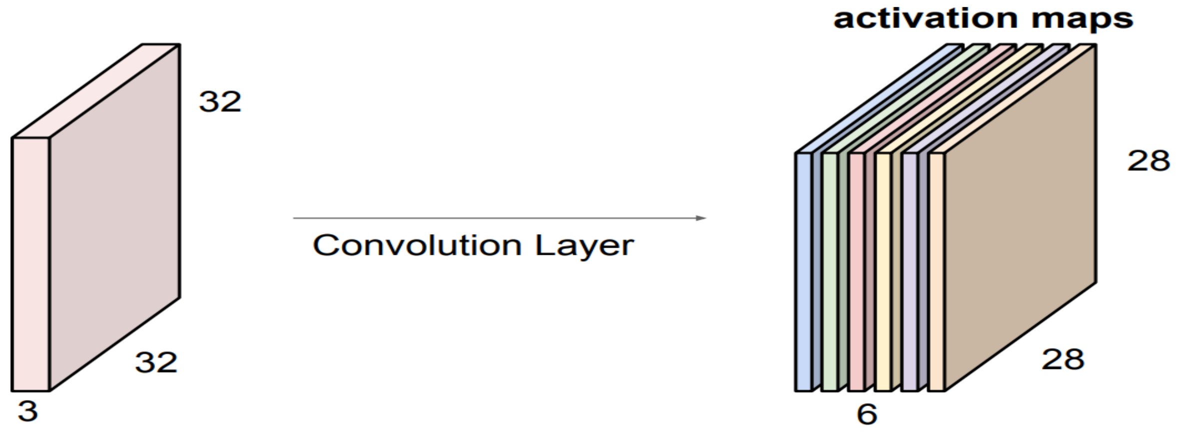
CNN: Convolutional Layer



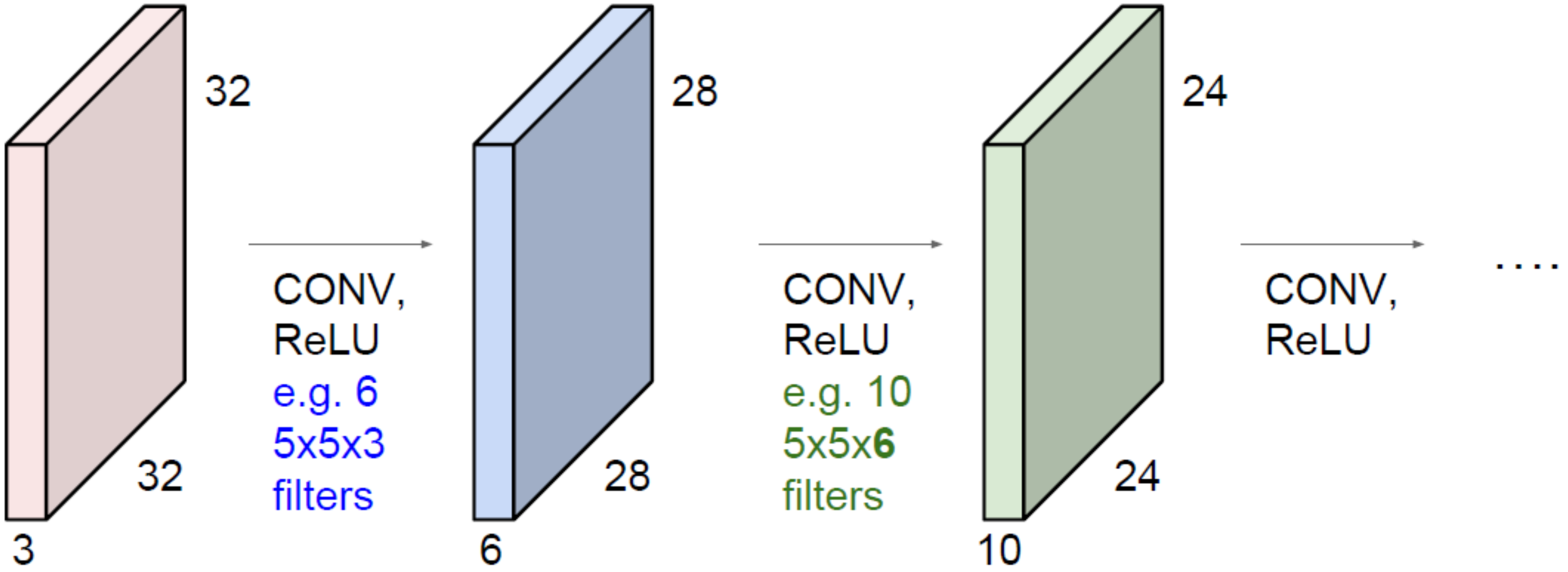
CNN: Convolutional Layer

- Stacking these activation maps for all filters along the depth dimension forms the full output volume
- Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at only a small region in the input and ***shares parameters*** with neurons in the same activation map (since these numbers all result from applying the same filter)

With 6 filters, we get
6 activation maps



Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

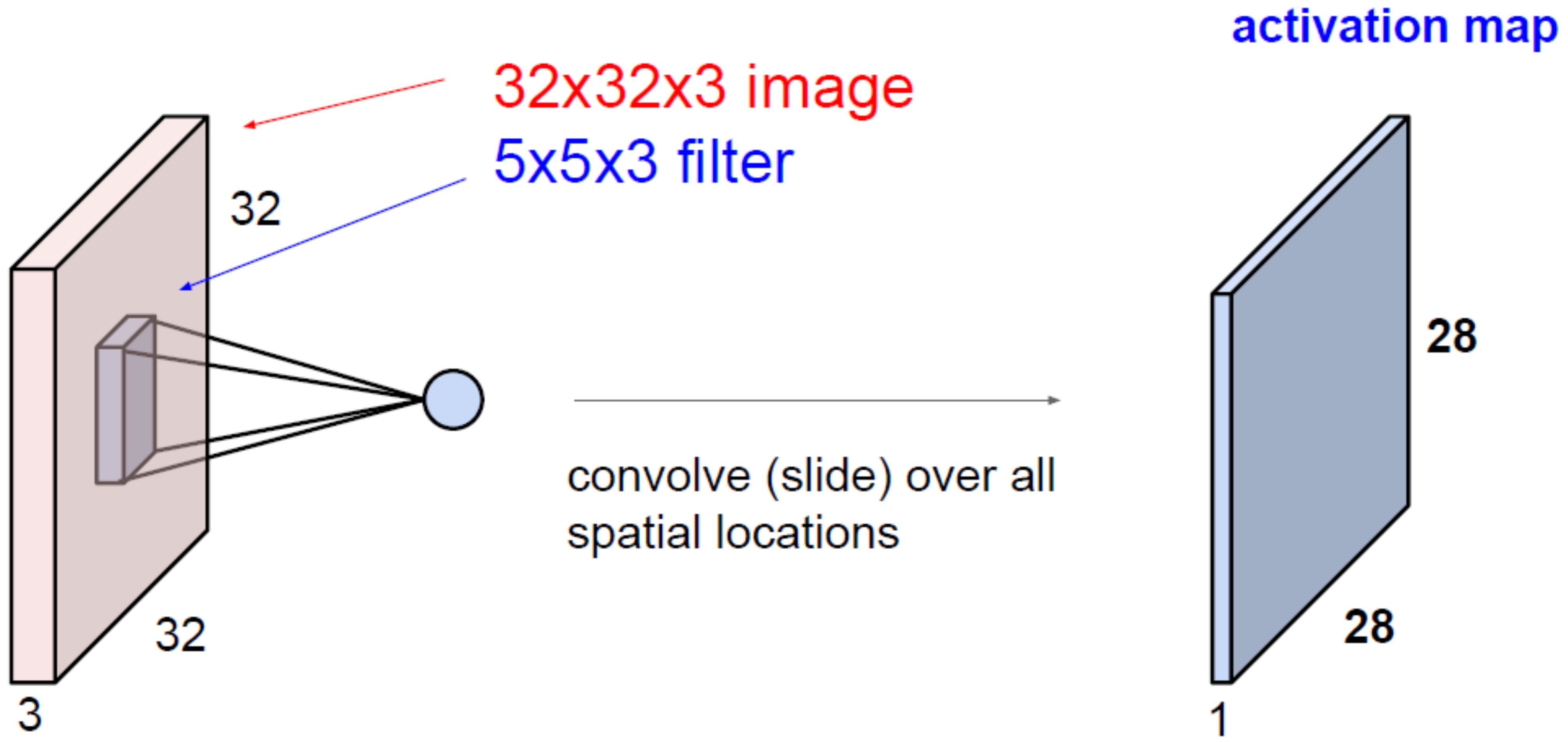


CNN: Convolutional Layer

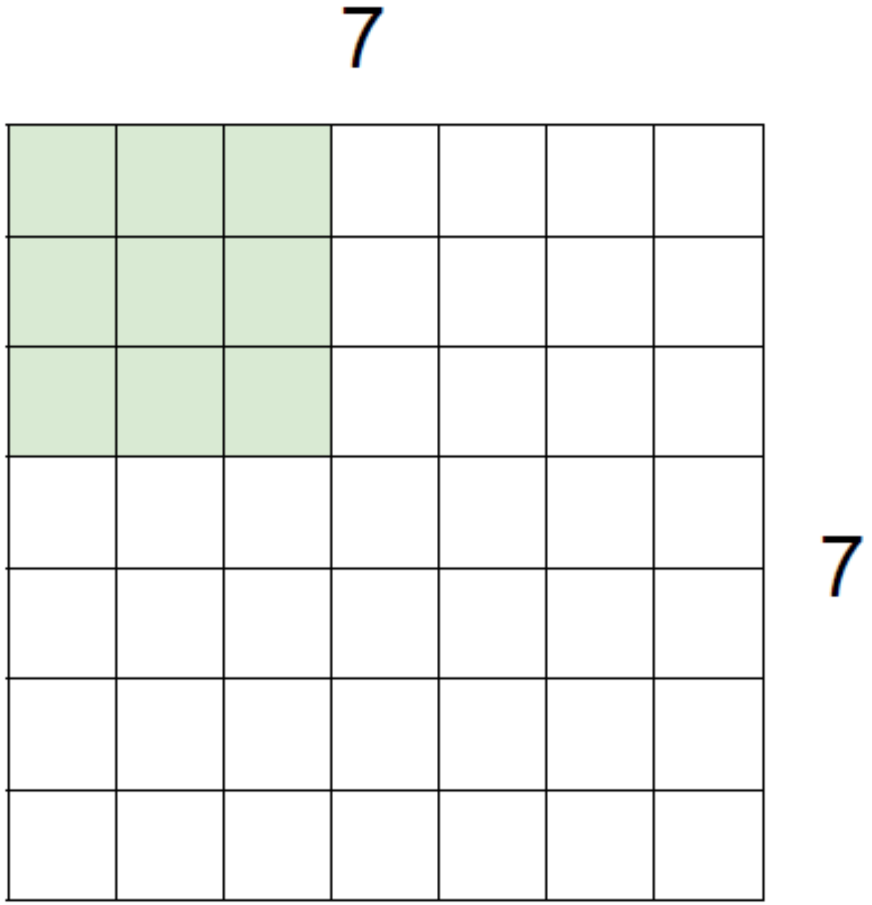
Local Connectivity

- As we have realized by now, it is impractical to use fully connected networks when dealing with high dimensional images/data
- Hence the concept of local connectivity: each neuron only connects to a local region of the input volume.
- The spatial extent of this connectivity is a concept called ***receptive field*** of the neuron.
- The extent of the connectivity along the depth axis is always equal to the depth of the input volume.
- The connections are local in space (along width and height), but always full along the entire depth of the input volume.

A closer look at spatial dimensions

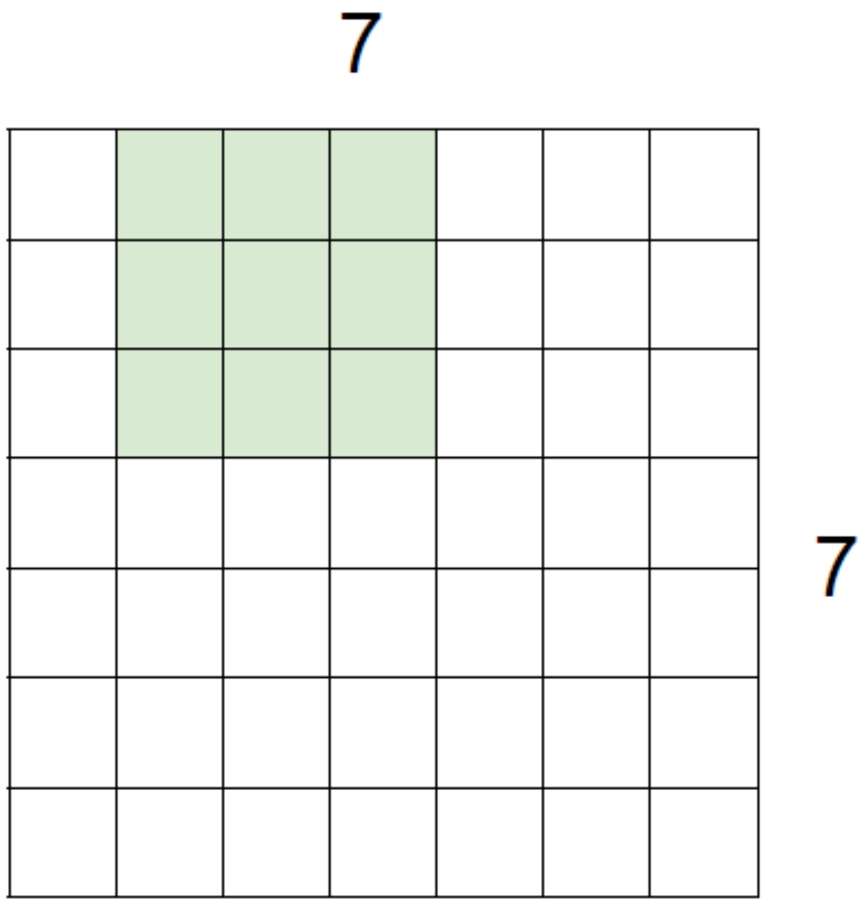


A closer look at spatial dimensions



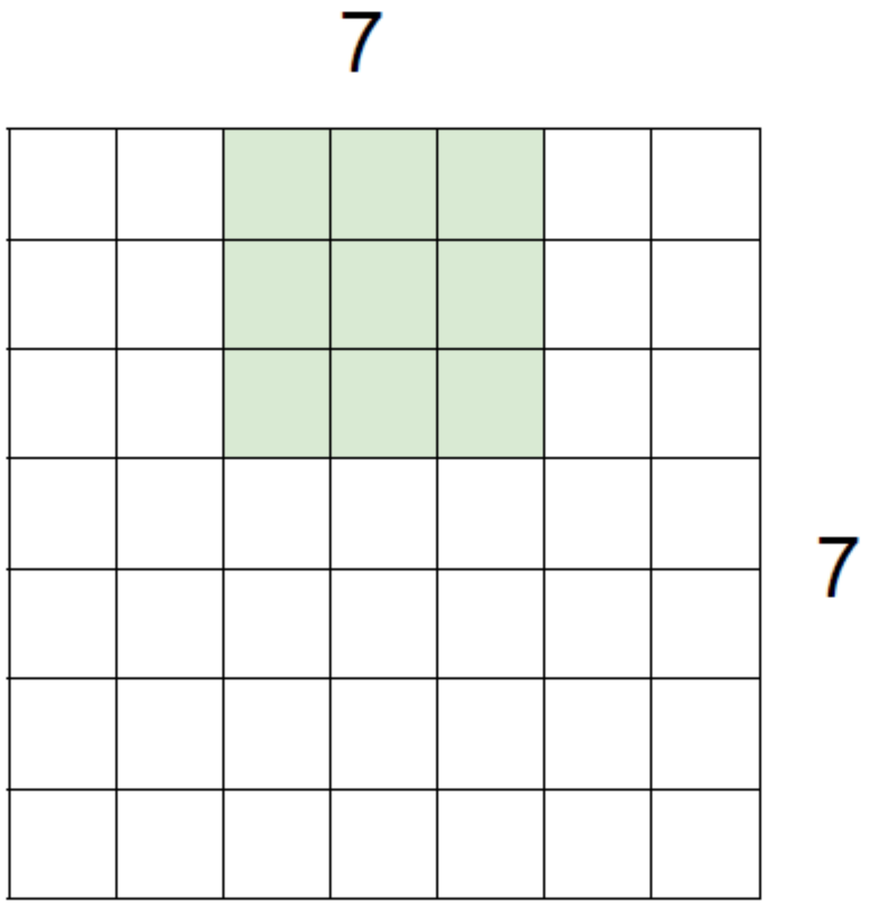
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions



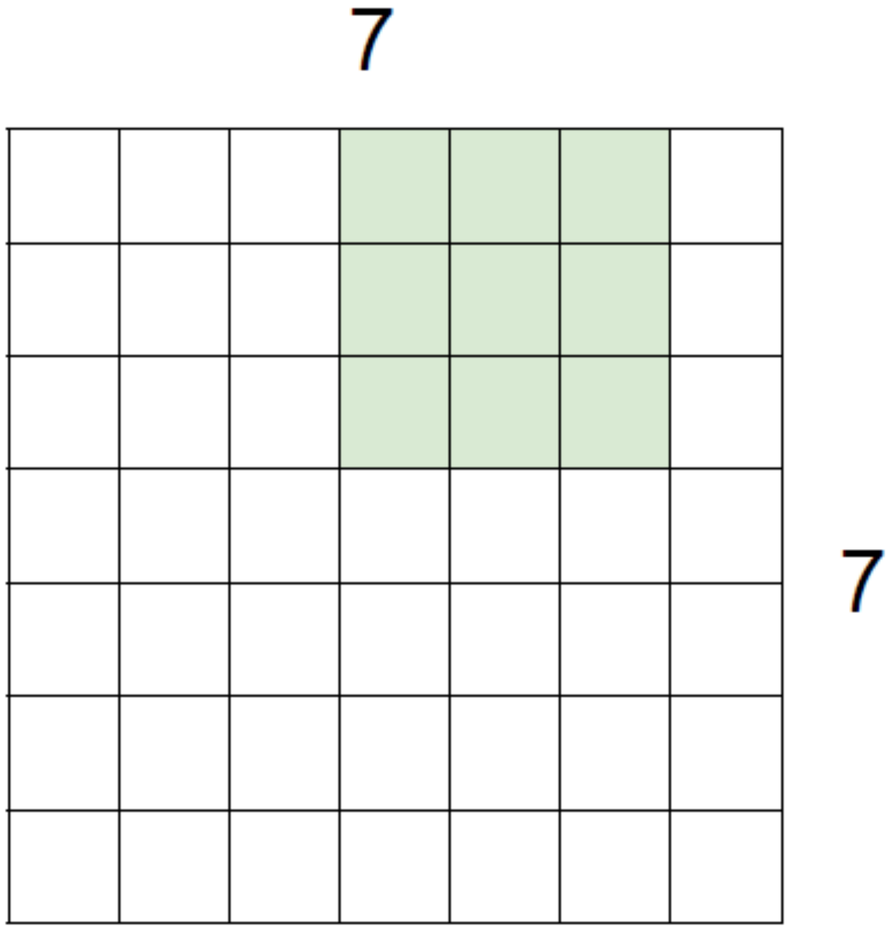
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions



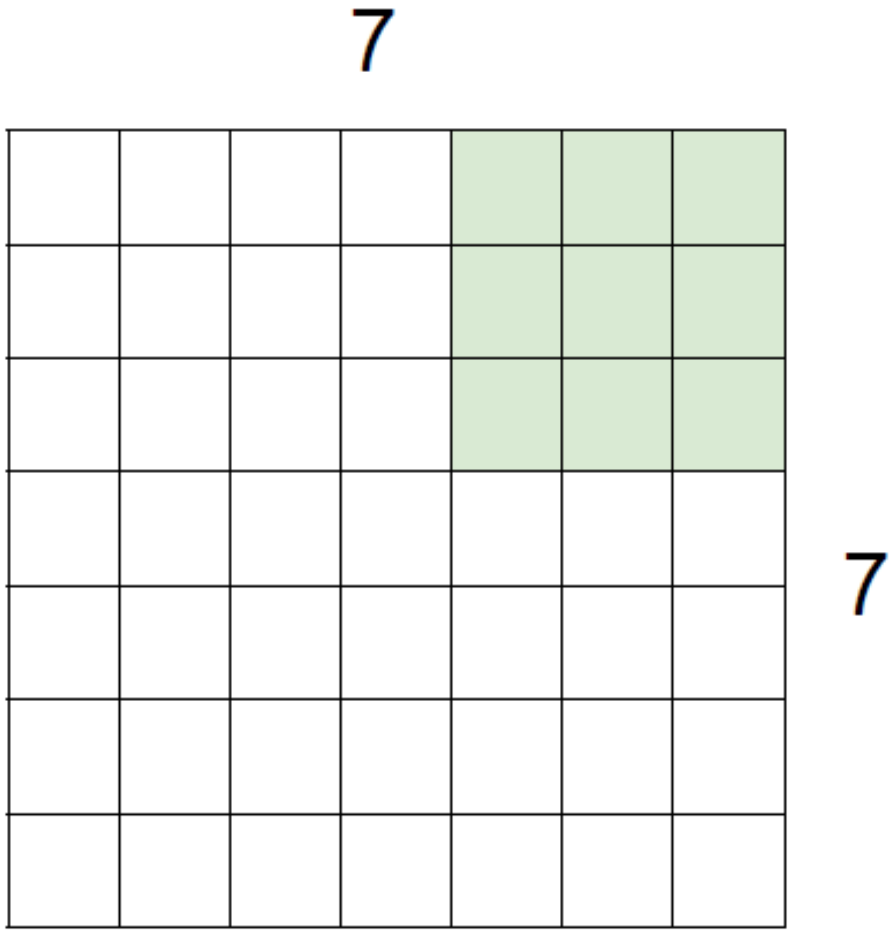
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

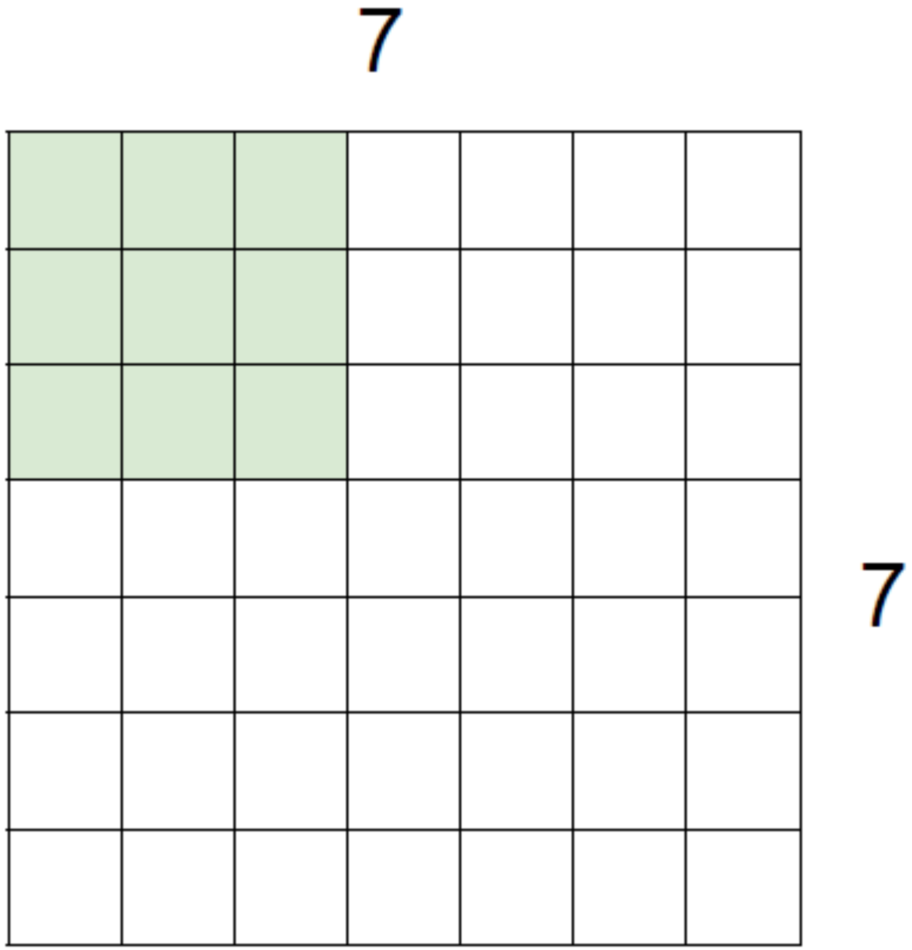
A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

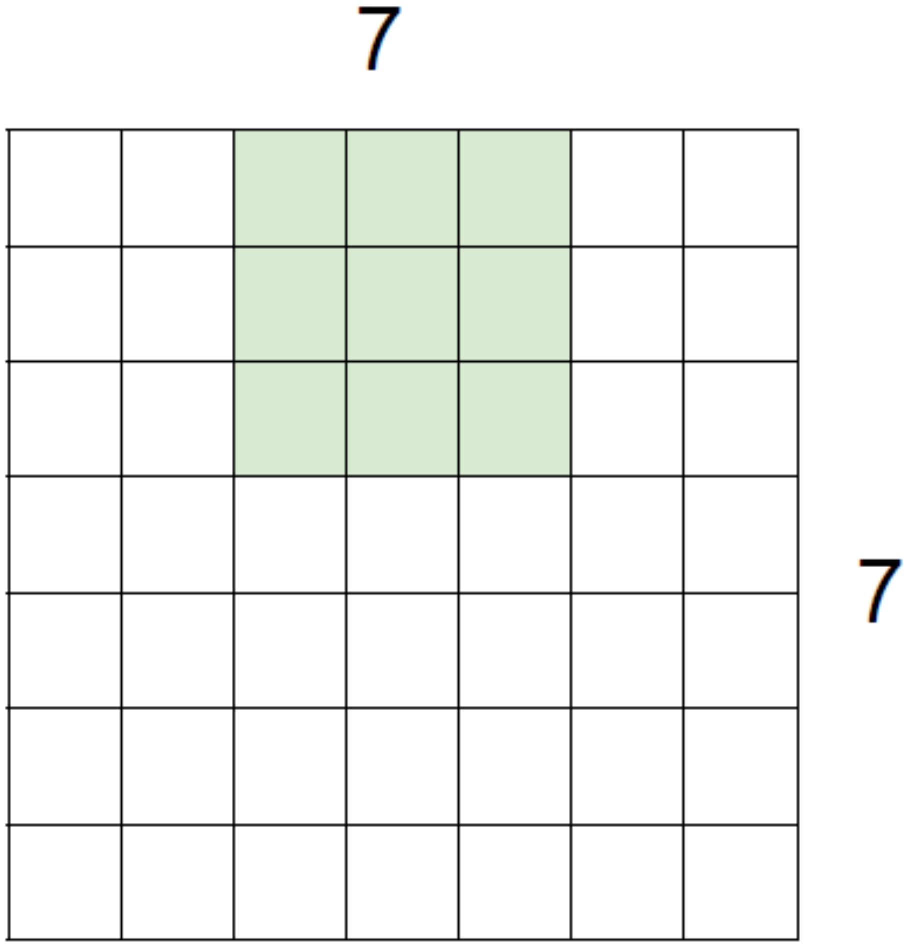
=> 5x5 output

A closer look at spatial dimensions



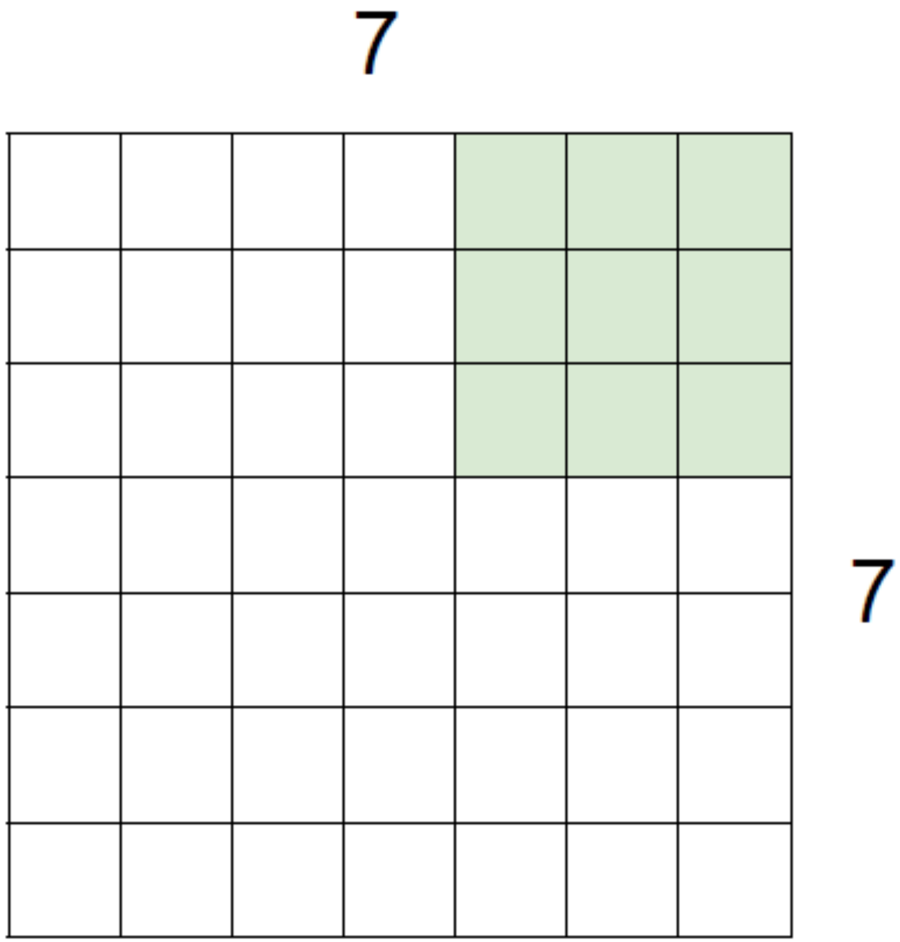
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

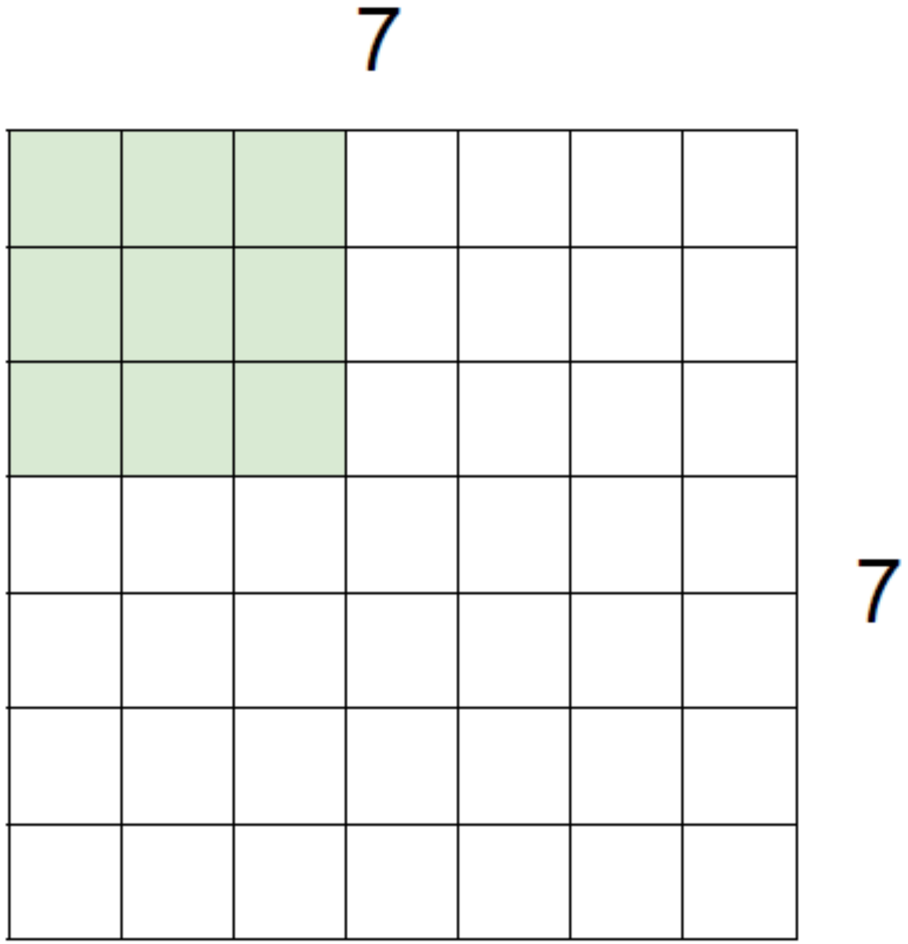
A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

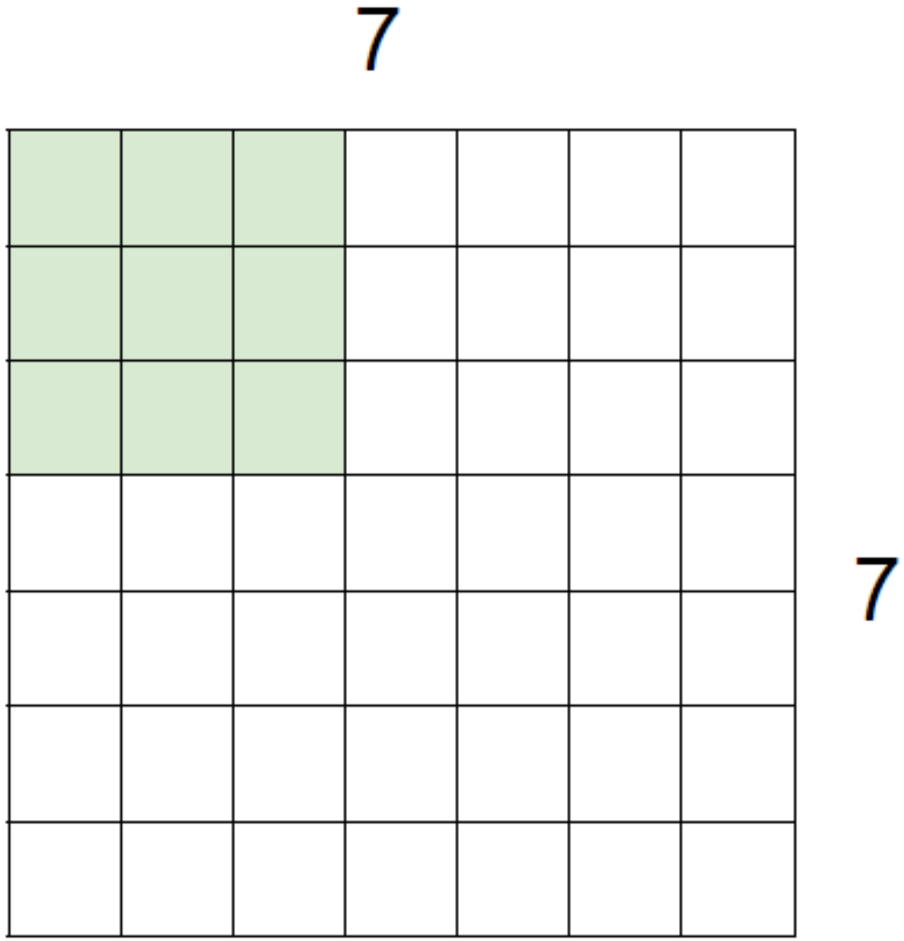
=> 3x3 output!

A closer look at spatial dimensions



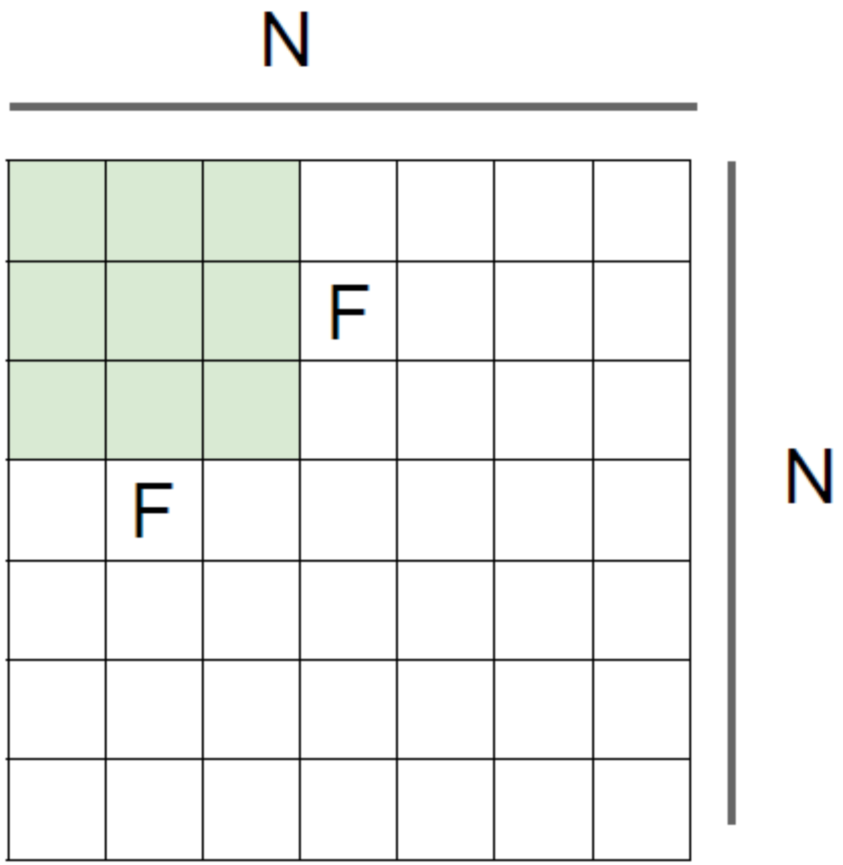
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

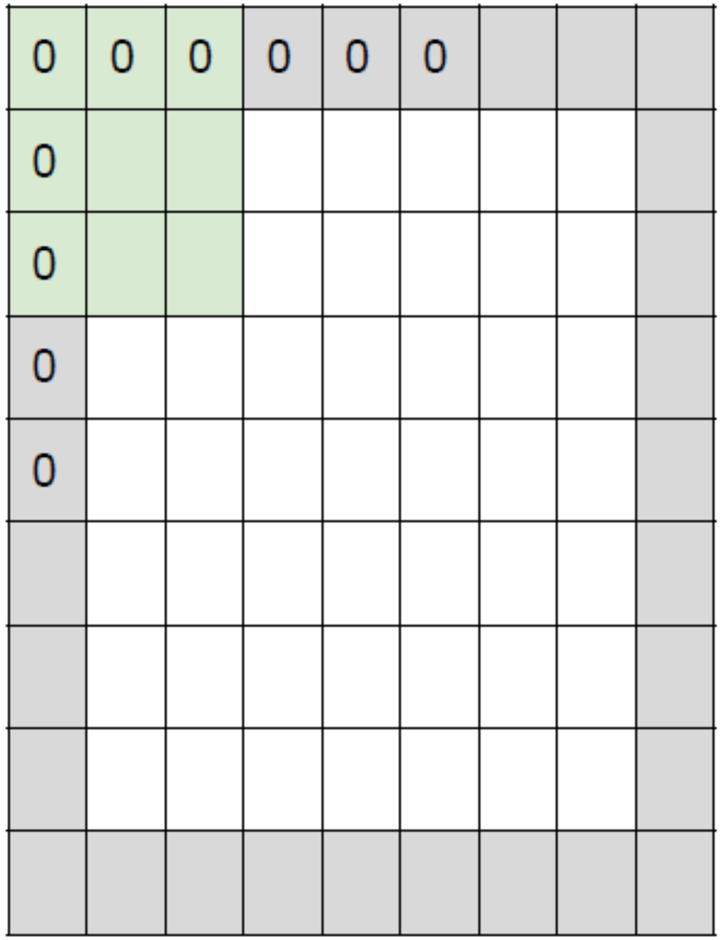
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



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 :\

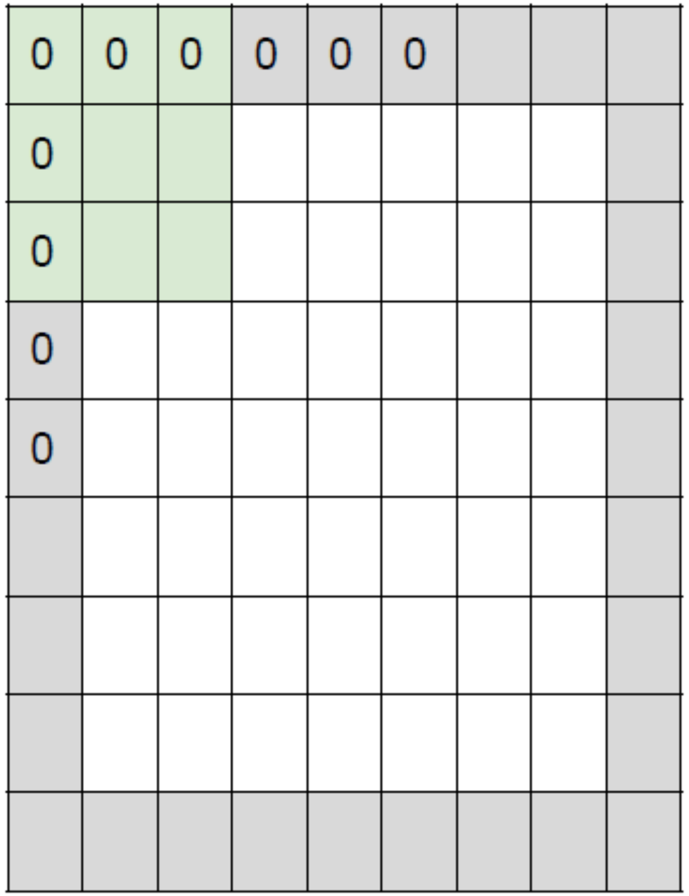
In practice: Common to zero pad the border



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

(recall:)
(N - F) / stride + 1

In practice: Common to zero pad the border

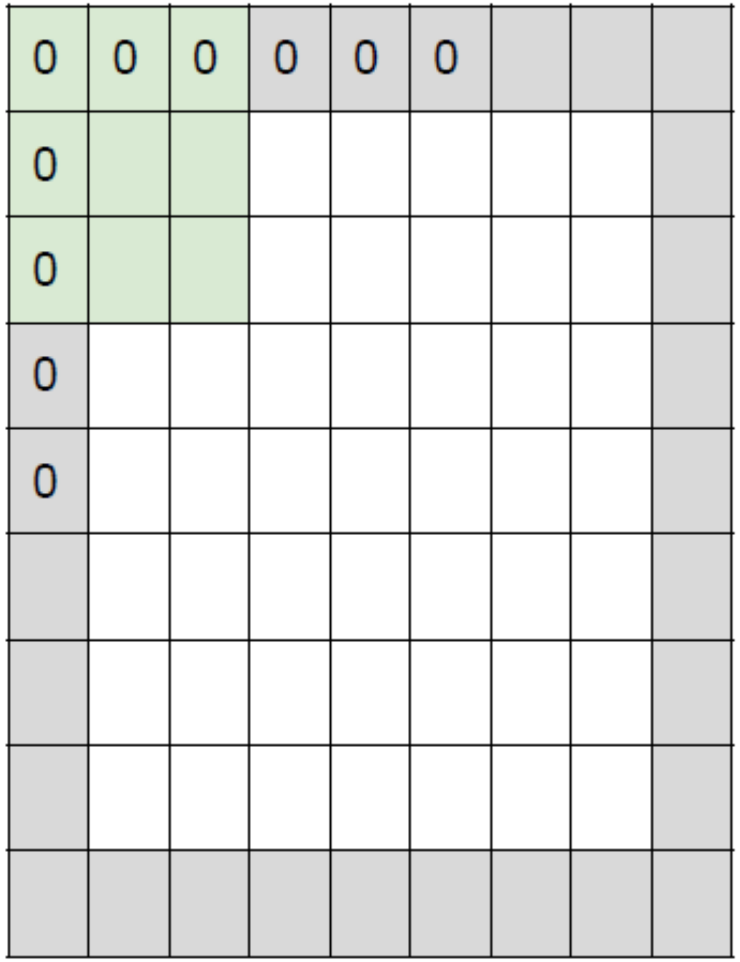


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

7x7 output!

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?

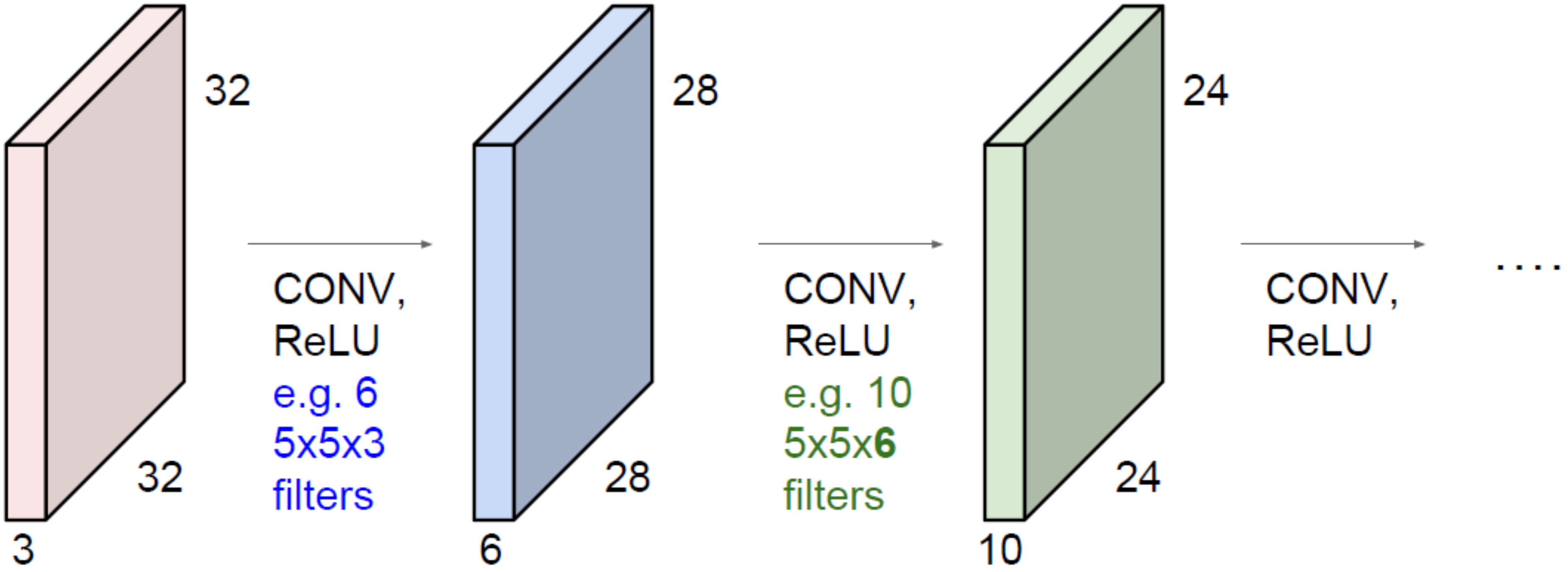
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)

e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

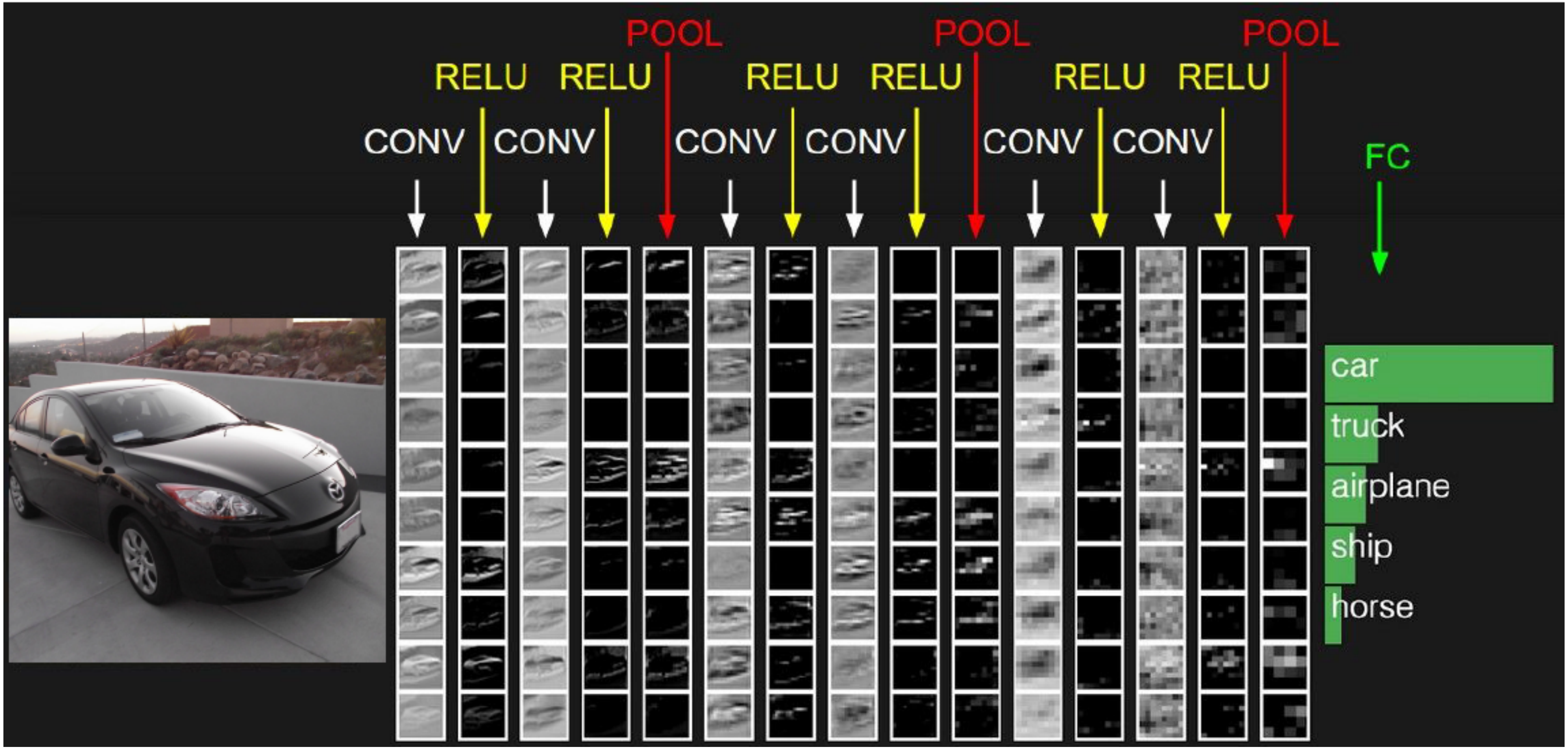
Other padding operations: replication padding, reflection padding …

Remember back to…

E.g. 32x32 input convolved repeatedly with **5x5** filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

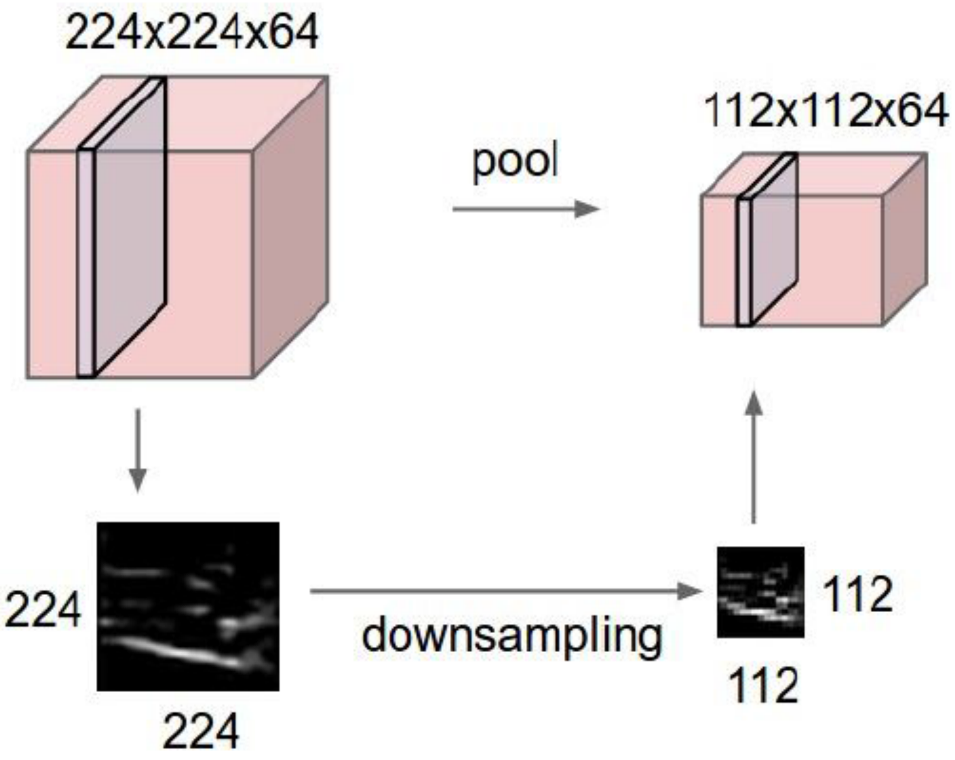


two more layers to go: POOL/FC



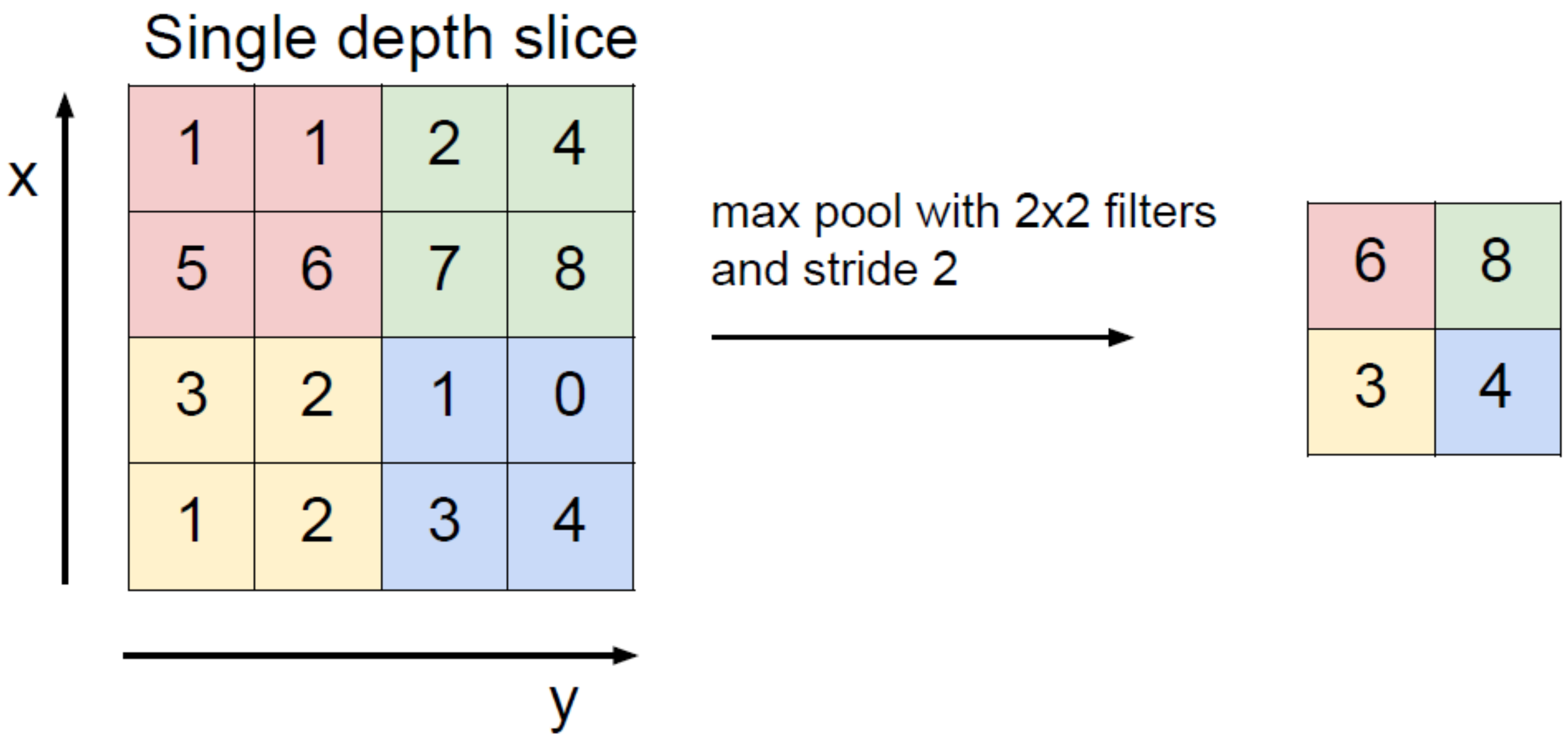
Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



From Fei-Fei Li & Andrej Karpathy & Justin Johnson lecture slides

Max Pooling



From Fei-Fei Li & Andrej Karpathy & Justin Johnson lecture slides

Pooling Layer: summary

Let's assume input is W_1 x H_1 x C
Conv layer needs 2 hyperparameters:

- The spatial extent **F**
- The stride **S**

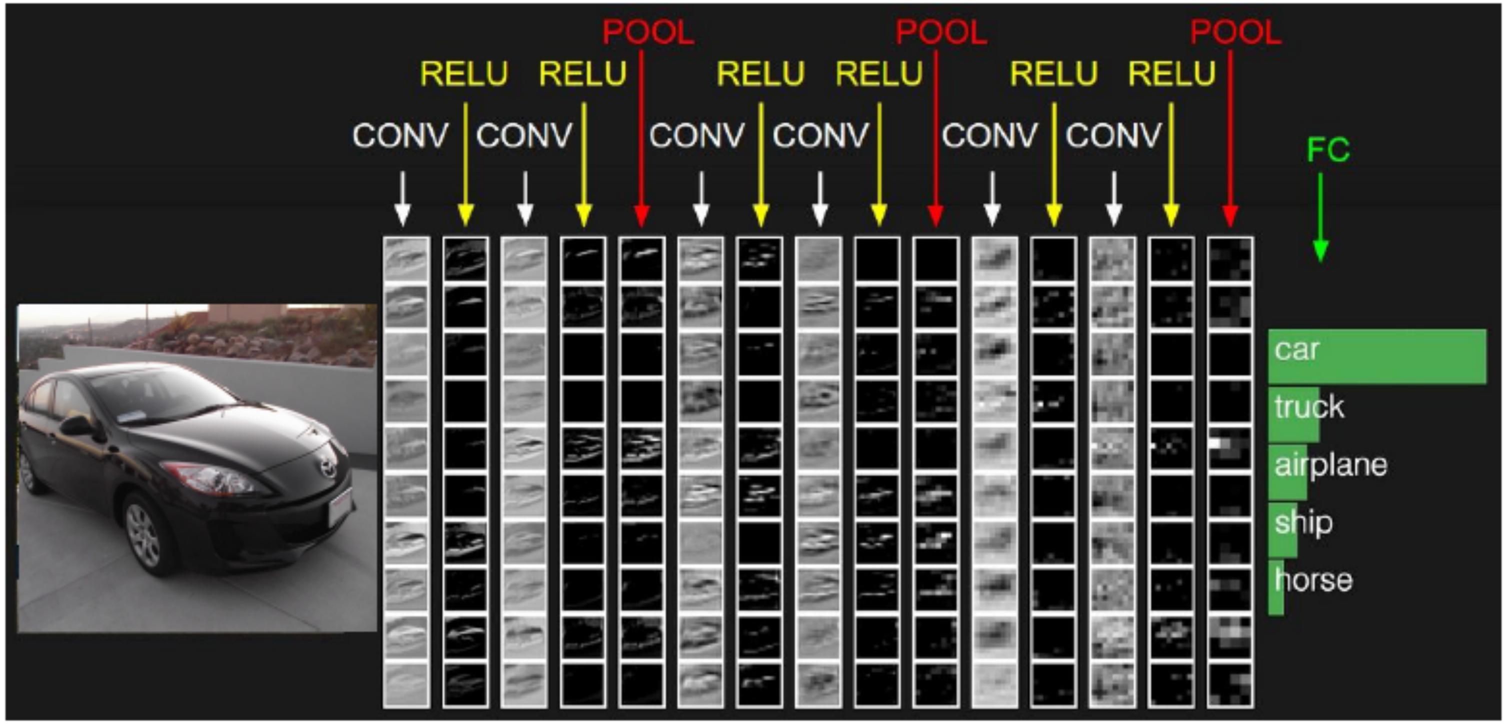
This will produce an output of W_2 x H_2 x C where:

- $W_2 = (W_1 - F)/S + 1$
- $H_2 = (H_1 - F)/S + 1$

Number of parameters: 0

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

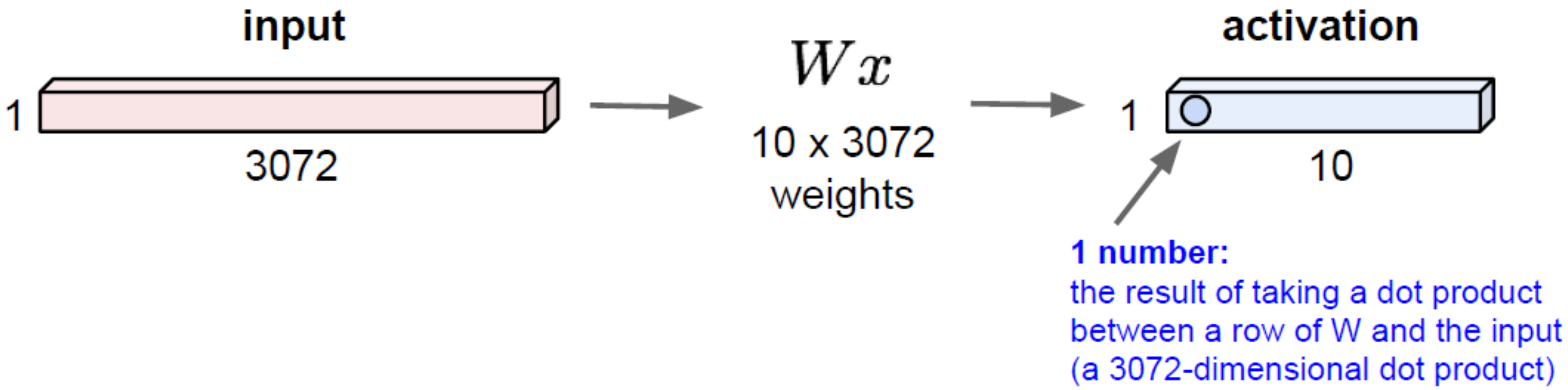


From Fei-Fei Li & Andrej Karpathy & Justin Johnson lecture slides

Fully Connected Layer (FC layer)

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



Summary of CNNs

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Historically architectures looked like **[(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX,** where N is usually up to ~5, M is large, 0 <= K <= 2.
  - but recent advances such as ResNet/GoogLeNet have challenged this paradigm

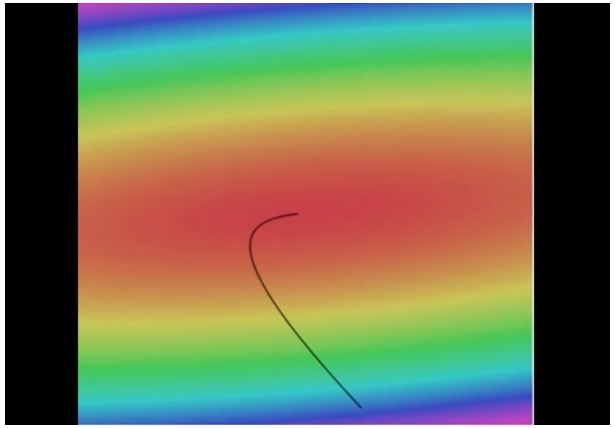
Training CNNs/Deep Neural Networks

CNN: Training

- A loss function is used to compute the model's prediction accuracy from the outputs
  - Most commonly used: categorical cross-entropy loss function

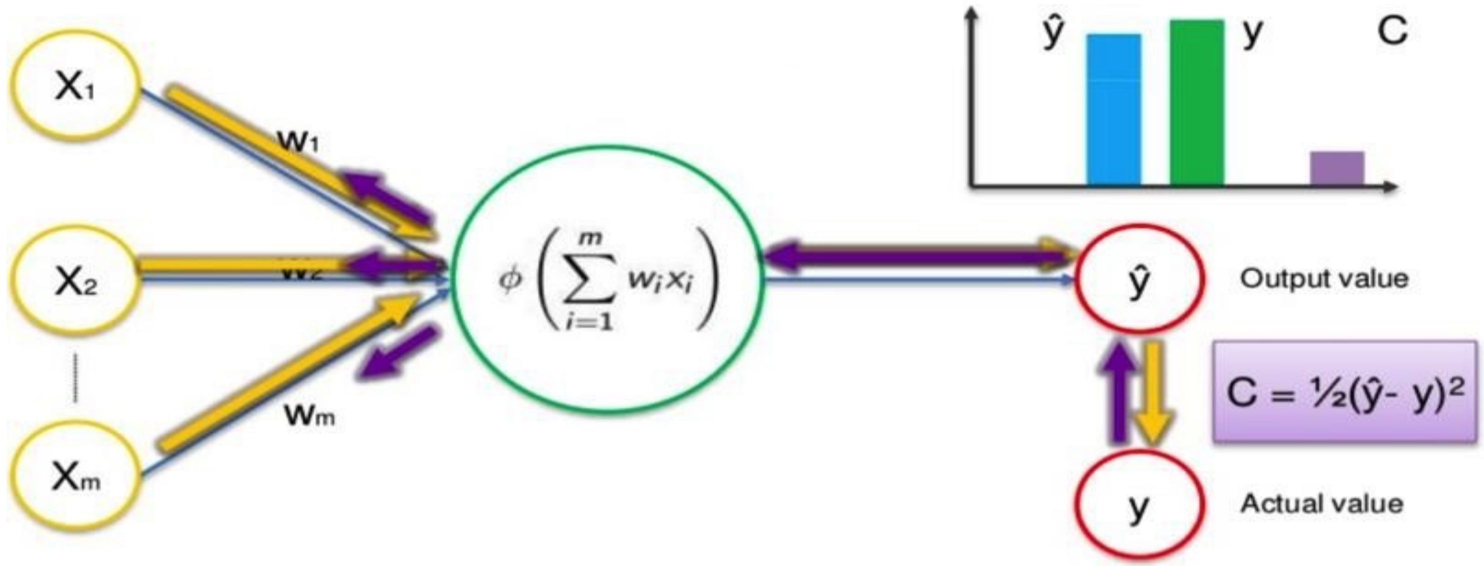
$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

- The training objective is to minimise this loss
- The loss guides the backpropagation process to train the CNN model
- Gradient descent based methods, such as Stochastic gradient descent and the Adam optimizer, are commonly used algorithms for optimisation

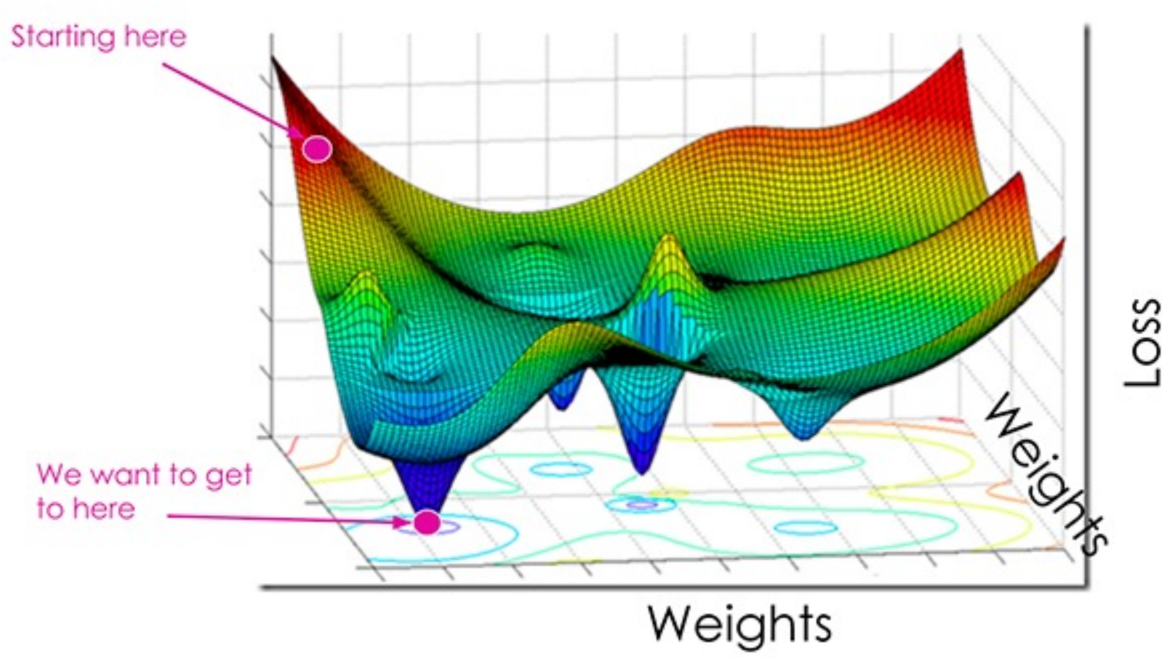


CNN: Training

- Backpropagation in general:
  - Calculating gradients for gradient descent
  - Directly deriving and calculating gradient is difficult, due to the complexity of DNNs

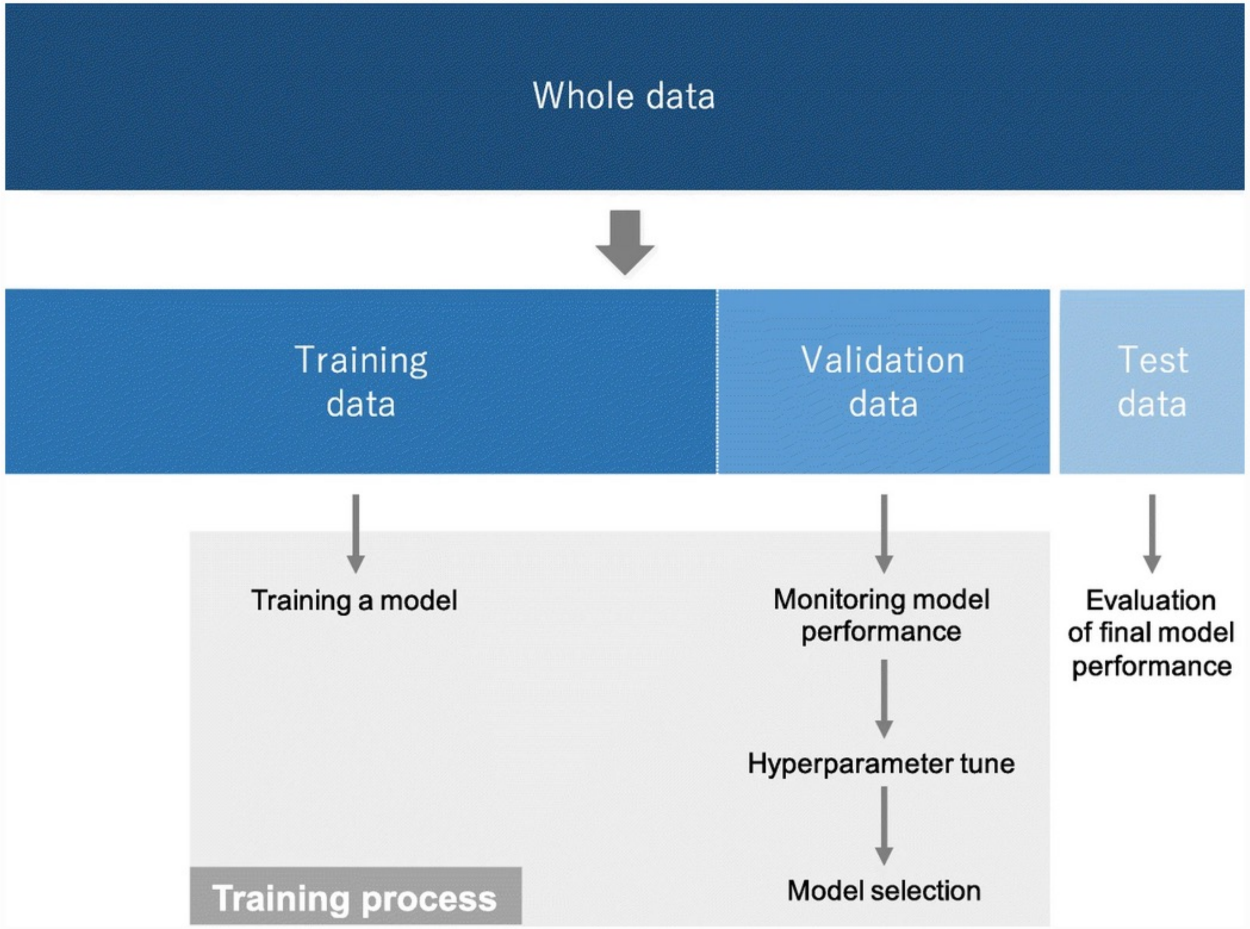


Why training Deep Neural Networks is hard?



Credit: Adrian Rosebrock, PyImageSearch, https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/

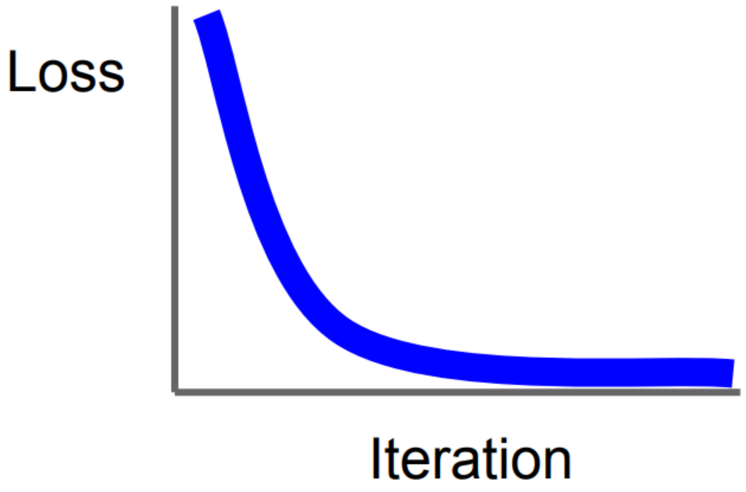
Training Methodology

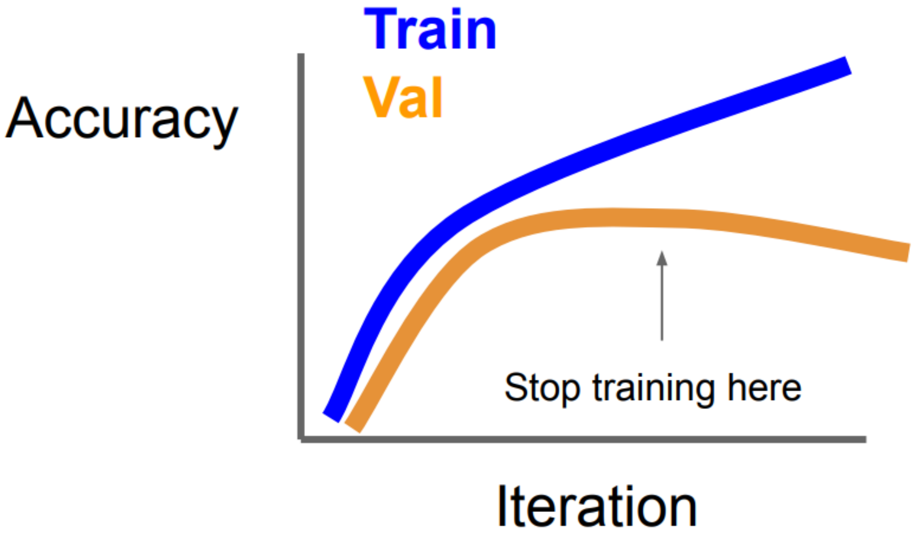


Source: Yamashita R. et al. (2018) Convolutional neural networks: an overview and applications in radiology

Training vs. Testing Error

- Proper optimizer and training strategy can minimize the loss.
- Small training error is not always corresponding to a small testing/validation error.



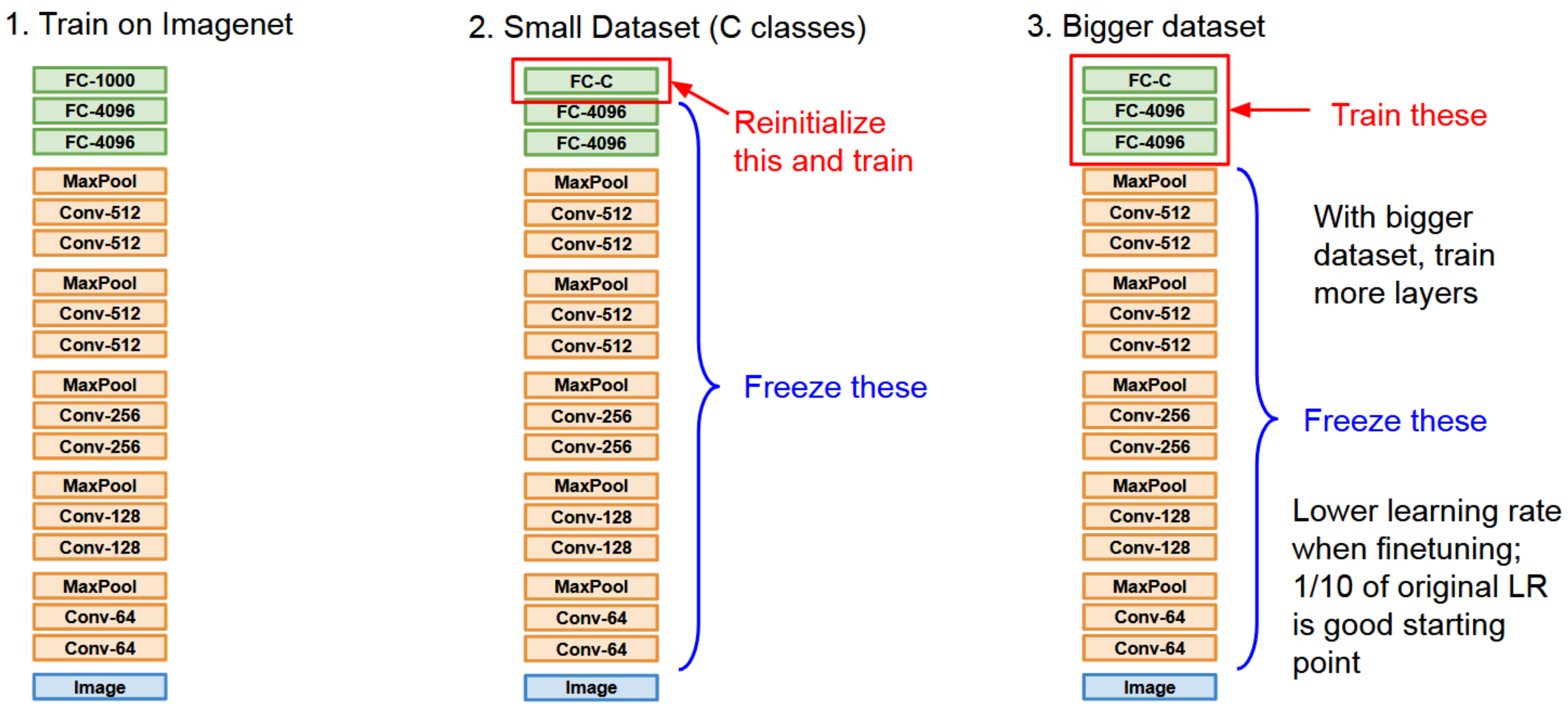


Transfer Learning

- Transfer learning aims to leverage the learned knowledge from a resource-rich domain/task to help learning a task with not sufficient training data.
  - Sometimes referred as **domain adaptation**
- The resource-rich domain is known as the **source** and the low-resource task is known as the **target**.
- Transfer learning works the best if the model features learned from the source task are **general** (i.e., domain-independent)

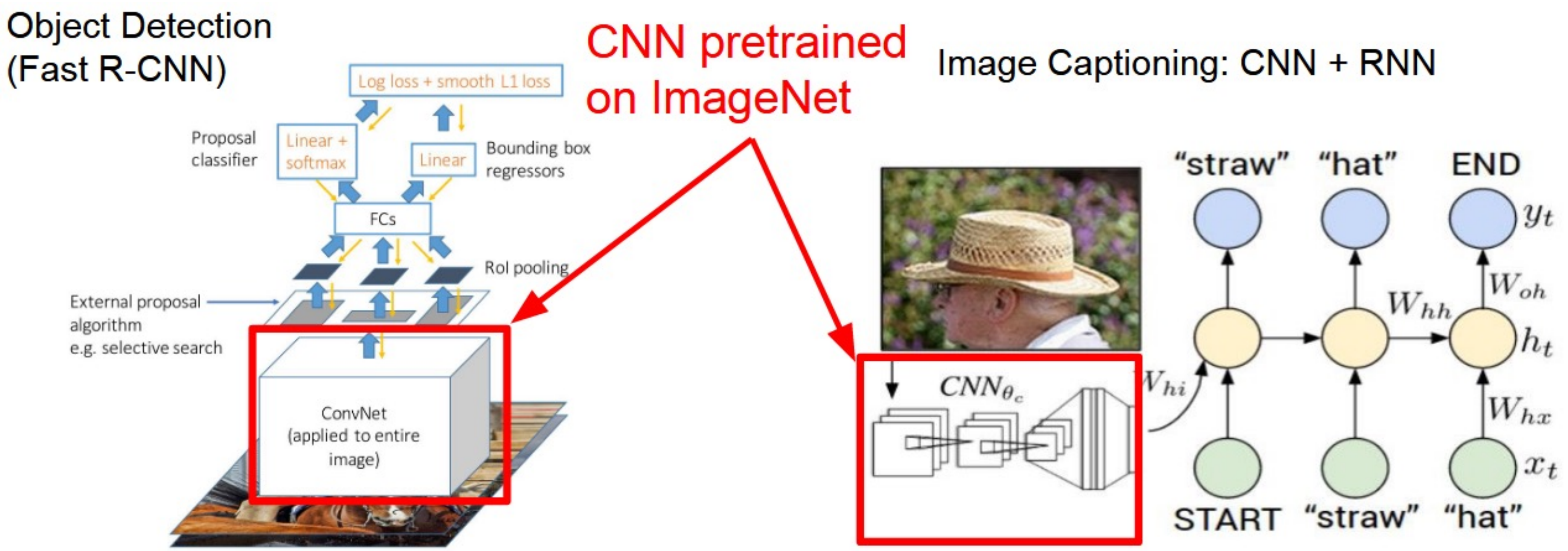
Credit: Mahammadreza Ebrahimi An Introduction to Deep Transfer Learning

Transfer Learning with CNNs



Slide Credit: Stanford CS231n Course

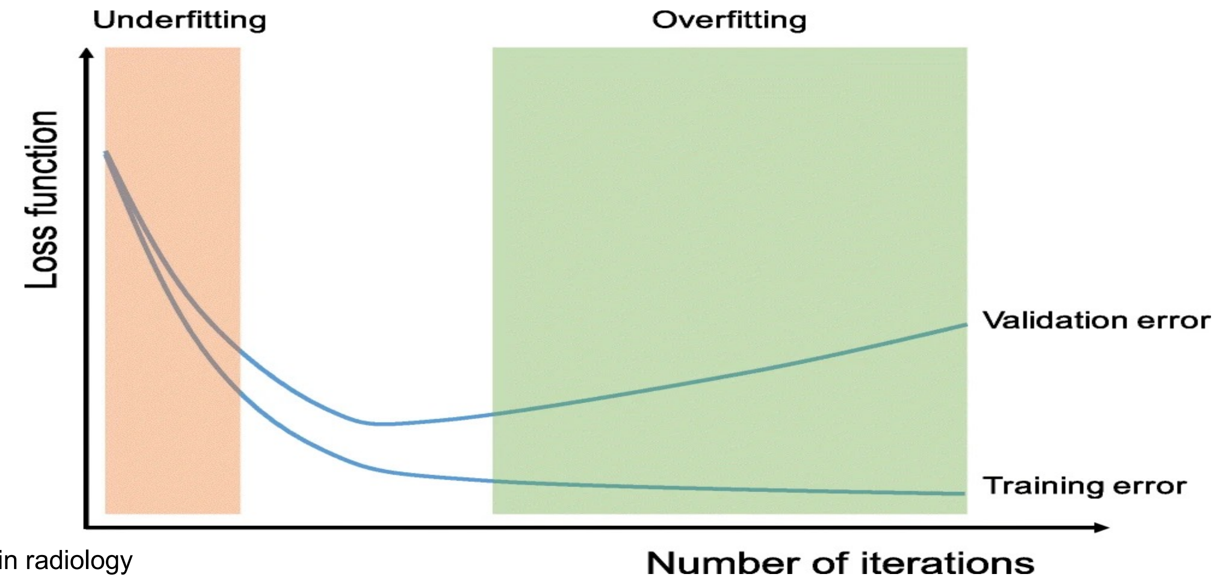
Transfer Learning is common in all applications



Slide Credit: Stanford CS231n Course

Overfitting and Underfitting

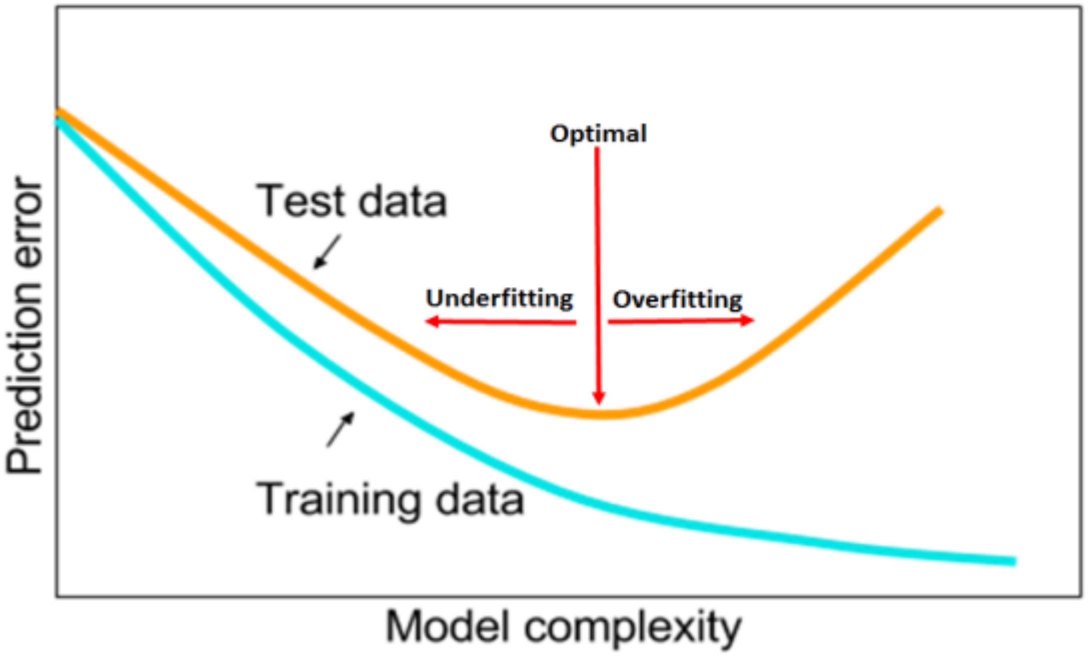
- Monitor the loss on training and validation sets during the training iteration.
- If the model performs poorly on both training and validation sets: Underfitting
- If the model performs well on the training set compared to the validation set: Overfitting



Source: Yamashita R. et al. (2018) Convolutional neural networks: an overview and applications in radiology

Common methods to mitigate overfitting

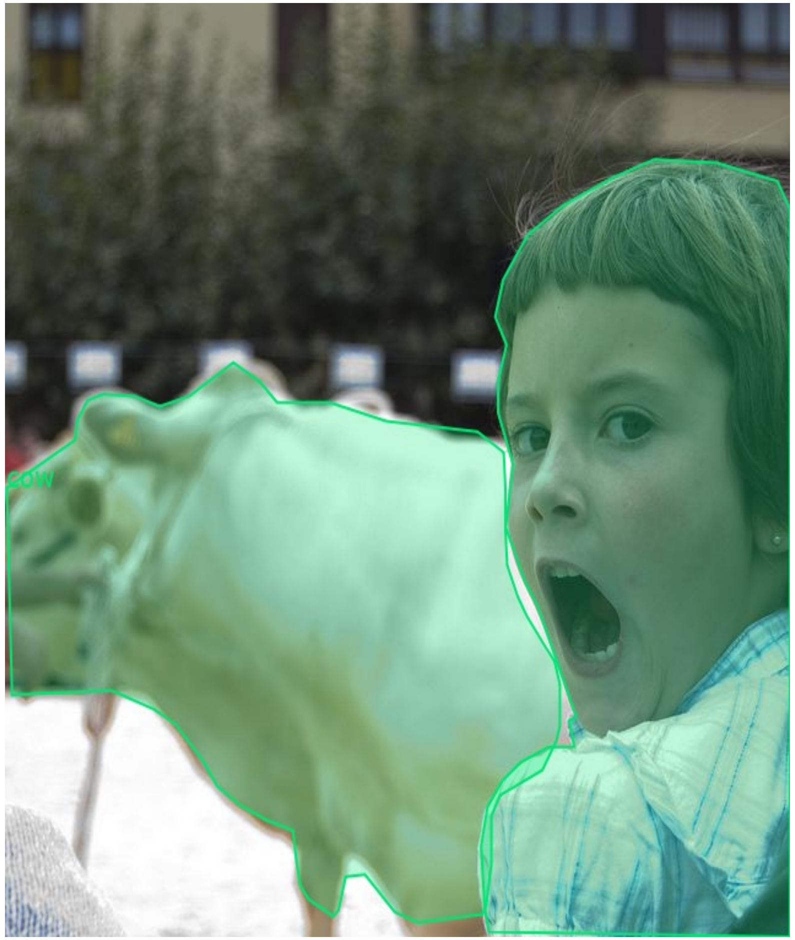
- ➢ More training data
- ➢ Early Stopping
- ➢ Data Augmentation
- ➢ Regularization (weight decay, dropout)
- ➢ Batch normalization



Source: Yamashita R. et al. (2018) Convolutional neural networks: an overview and applications in radiology
Image Credit: Hyper-parameters tuning practices: learning rate, batch size, momentum, and weight decay. Medium

More training data

- Costly
- Time consuming
- Need experts for specialized domains



Source: Fast Annotation Net: A framework for active learning in 2018. https://medium.com/diffgram/fast-annotation-net-a-framework-for-active-learning-in-2018-1c75d6b4af92
Image Datasets — ImageNet, PASCAL, TinyImage, ESP and LabelMe — what do they offer ? Medium Blog

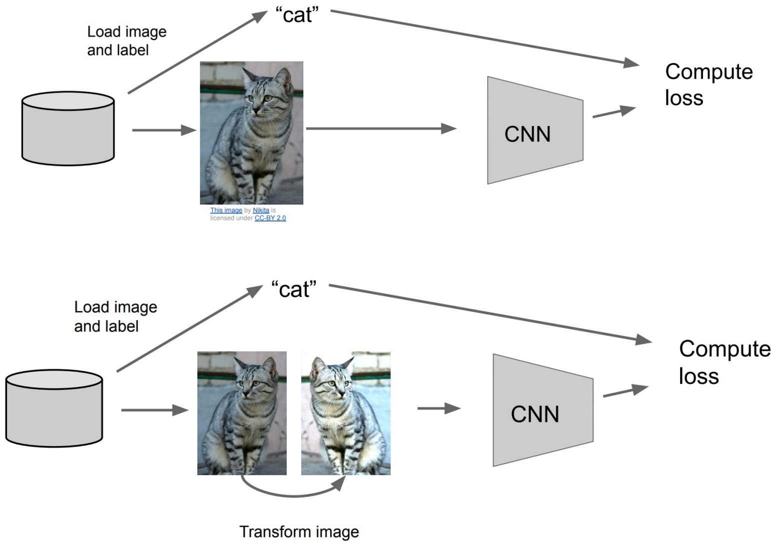
Early Stopping

- Training too little mean model will underfit on the training and testing sets
- Training too much mean model will overfit the training dataset and hence poor performance on test set
- Early Stopping:
  - To stop training at the point when performance on a validation set starts to degrade.
  - Idea is to stop training when generalization error increases
- How to use Early Stopping
  - Monitoring model performance: Using metric to evaluate to monitor performance of the model during training
  - Trigger to stop training:
    - No change in metric over a given number of epochs
    - A decrease in performance observed over a number of epochs
- Some delay or "patience" is good for early stopping

Source: Machine Learning Mastery: A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks URL: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

Data Augmentation

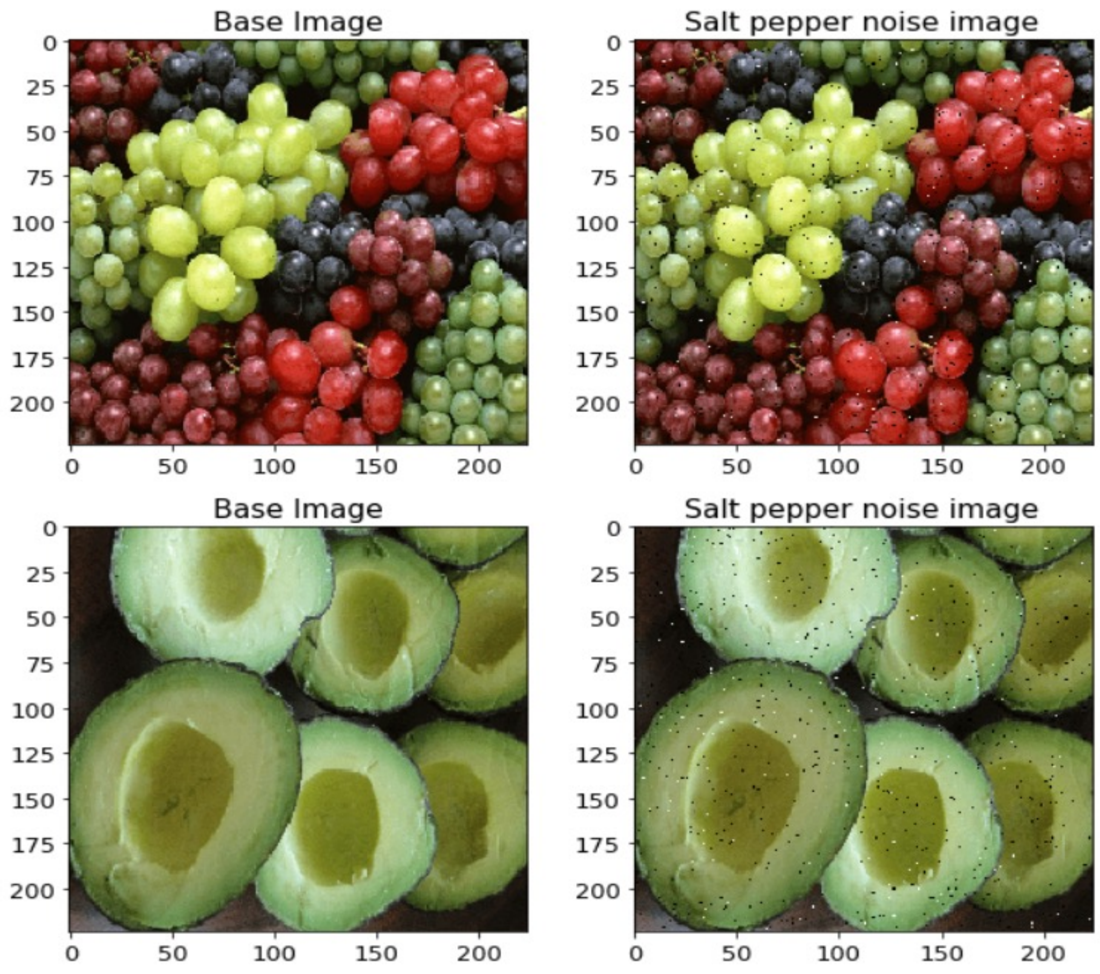
- ➢Data augmentation generate different versions of a real dataset artificially to increase its size
- ➢Improving the robustness of the networks
- ➢We use data augmentation to handle data scarcity and insufficient data diversity
- ➢Data augmentation helps to increase performance of deep neural networks
- ➢Common augmentation techniques:
  - ➢ Adding noise
  - ➢ Cropping
  - ➢ Flipping
  - ➢ Rotation
  - ➢ Scaling
  - ➢ Translation
  - ➢ Brightness
  - ➢ Contrast
  - ➢ Saturation
  - ➢ Generative Adversarial Networks (GANs)



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

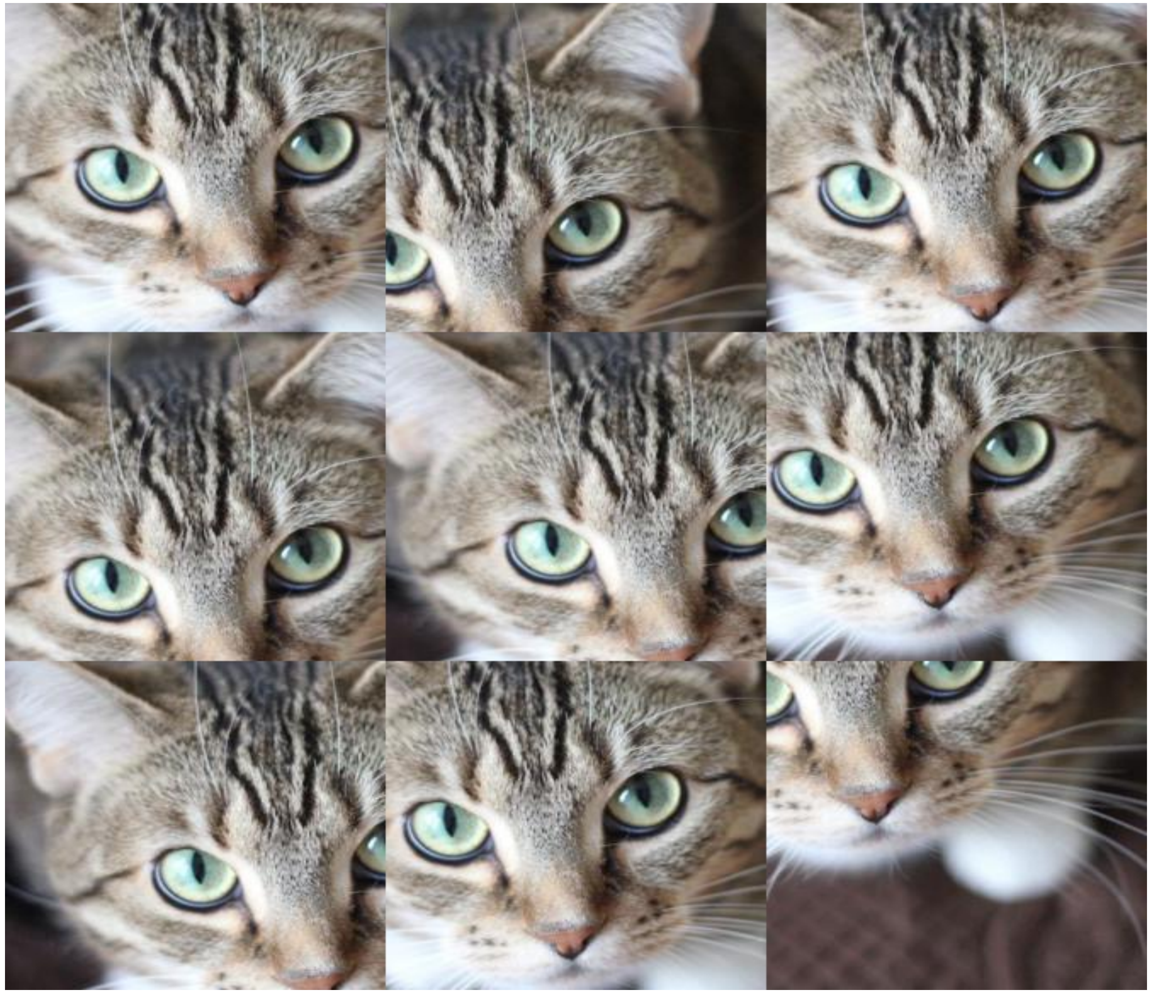
- Adding noise



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

- Cropping



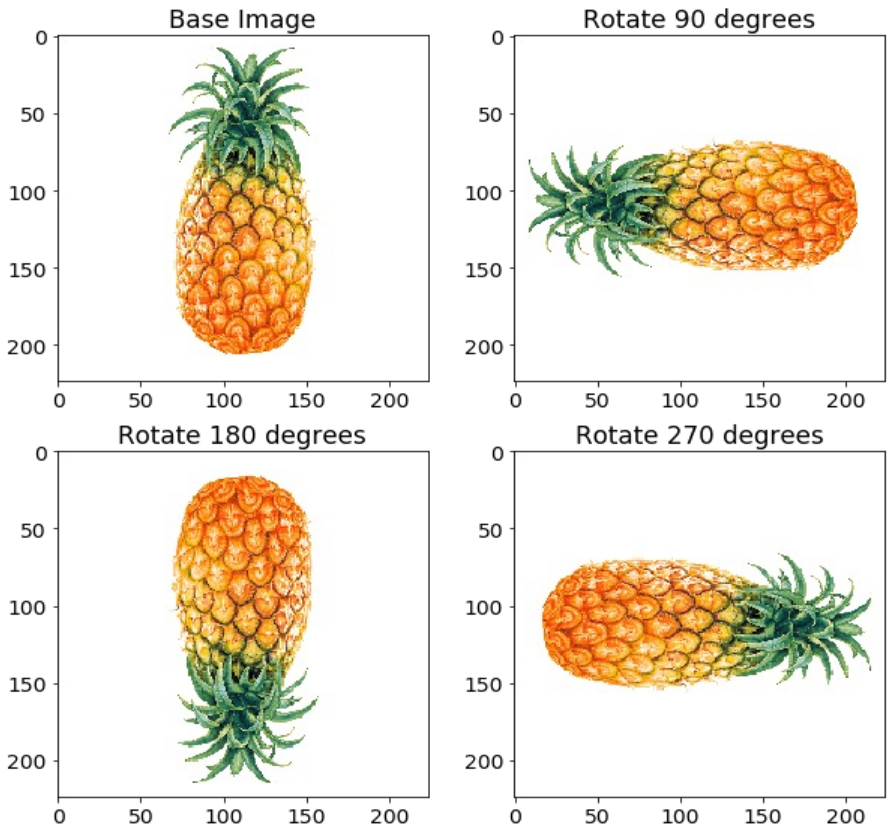
Data Augmentation

- Flipping

Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

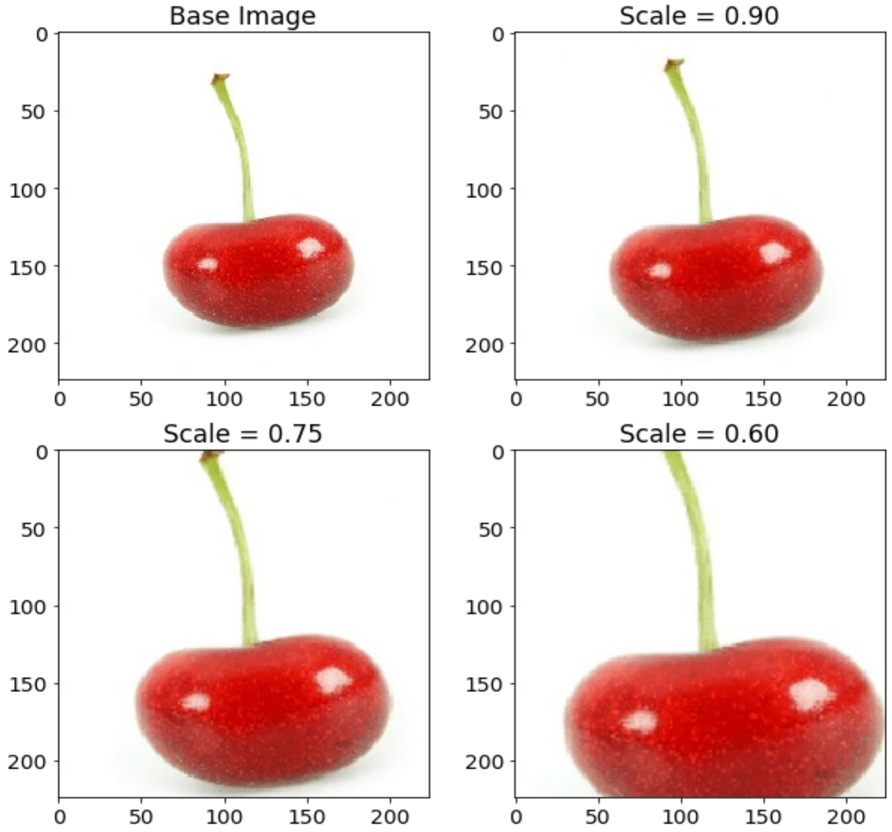
Rotation



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

- Scaling



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

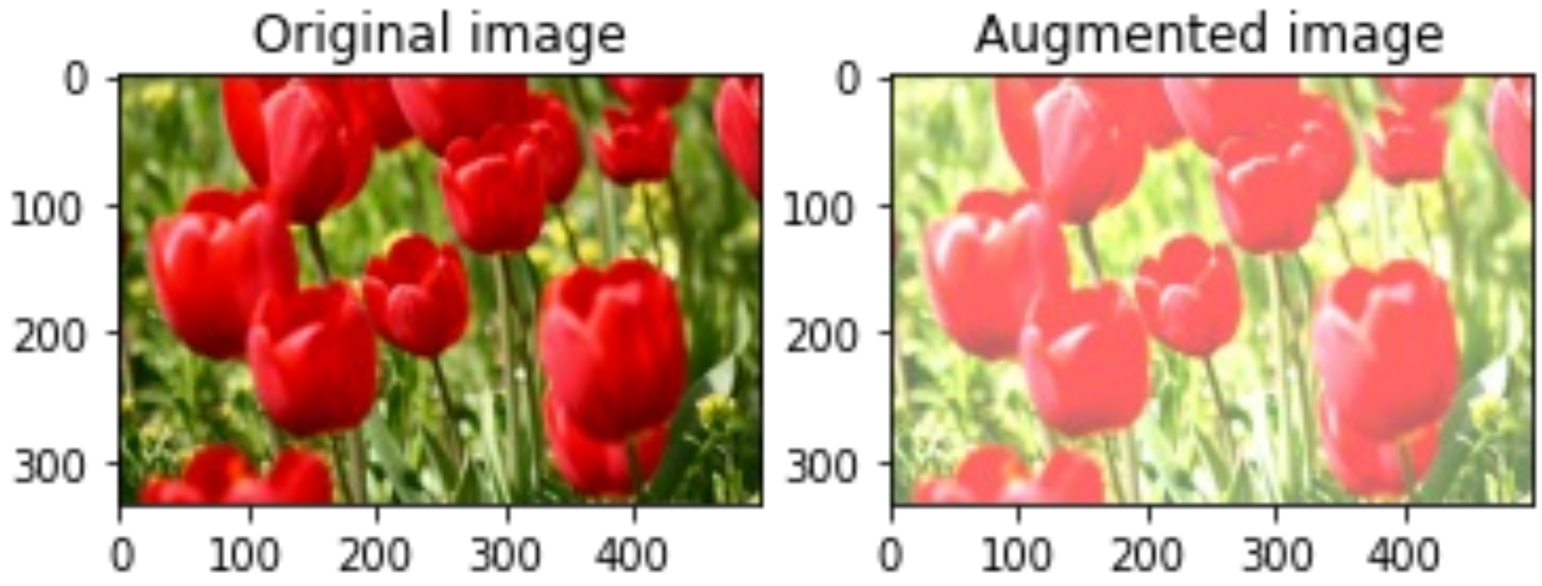
Data Augmentation

Translation

Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/
https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Data Augmentation

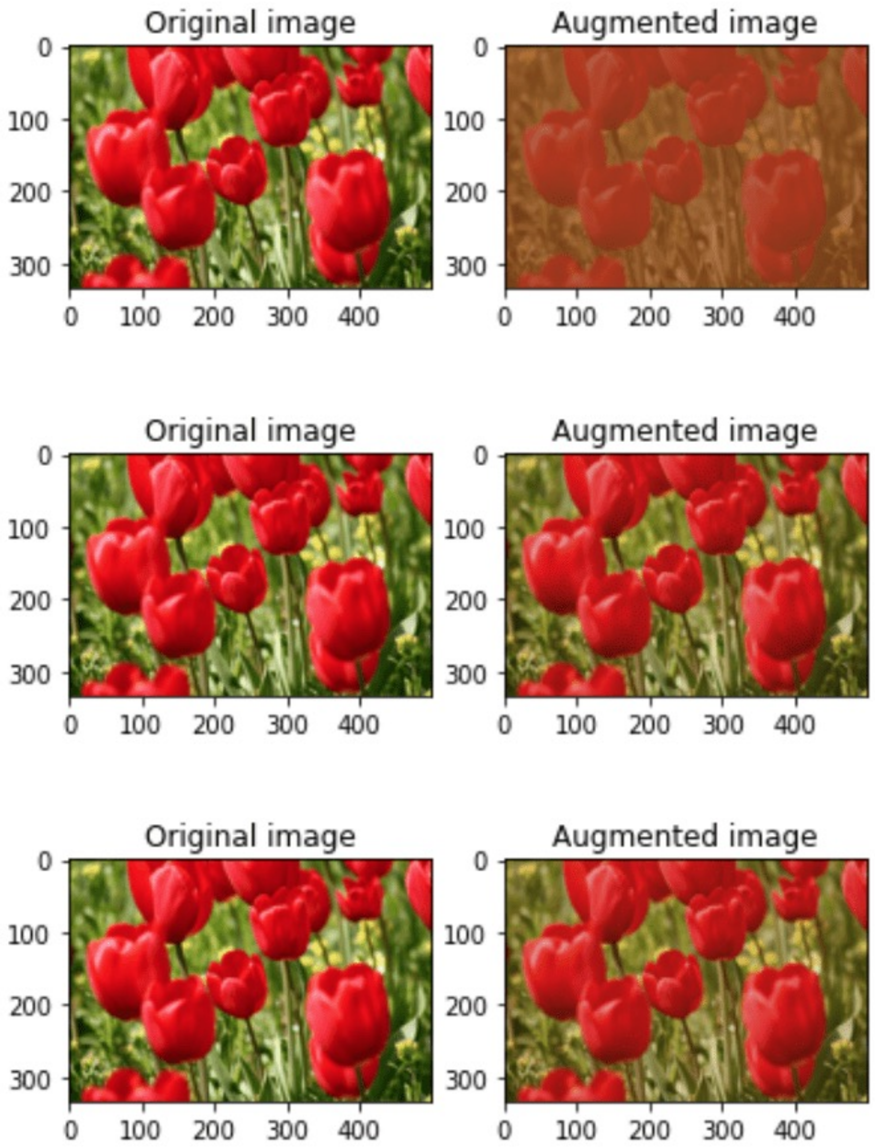
- Brightness



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

- Contrast



Source: 13 Data Augmentation Techniques. https://research.aimultiple.com/data-augmentation-techniques/

Data Augmentation

- ➢Generative Adversarial Networks (GANs) for data augmentation

StyleGAN2 (baseline)

StyleGAN2 + DiffAugment (ours)

Source: Zhao et al., Differential Augmentation for Data-Efficient GAN Training, NeurIPS, 2020

Regularization: Weight Decay

➢It adds a penalty term to the loss function on the training set to reduce the complexity of the learned model

➢Popular choice for weight decay:

➢ L1: The L1 penalty aims to minimize the absolute value of the weights

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n} |\theta_i|}$$

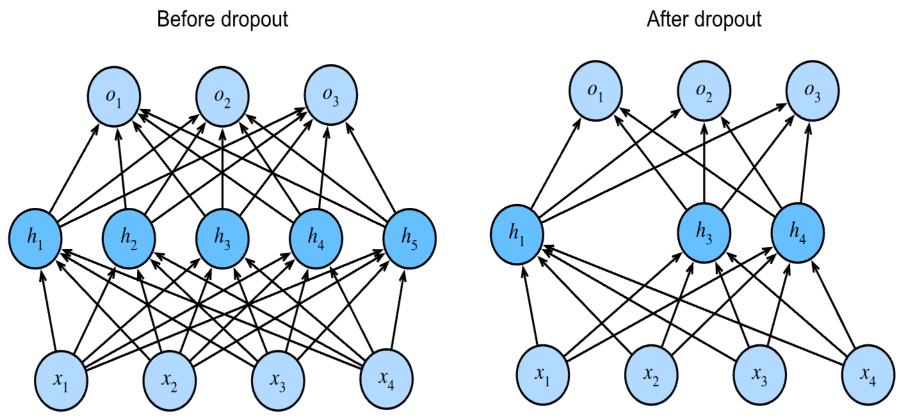
➢ L2: The L2 penalty aims to minimize the squared magnitude of the weights

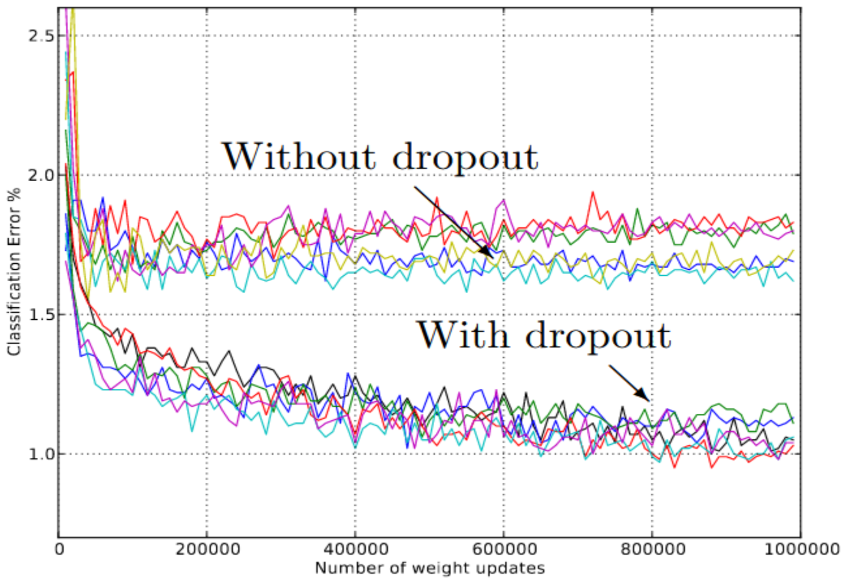
$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n} \theta_i^2}$$

Credit: 5 Techniques to Prevent Overfitting in Neural Networks. https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html

Regularization: Dropout

- L1 and L2 reduce overfitting by modifying the cost function
- Dropout regularizes the network by randomly dropping neurons from the neural network during training



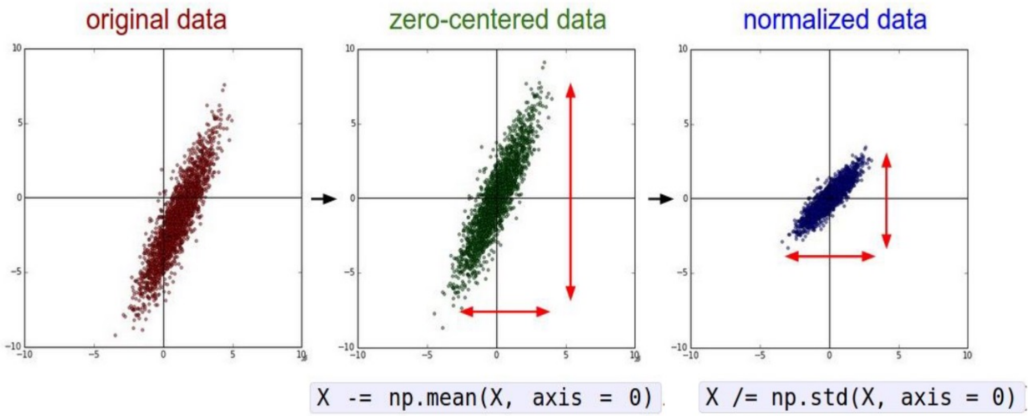


Credit: Srivastava et al., Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014
https://colab.research.google.com/github/d2l-ai/d2l-en-colab/blob/master/chapter_multilayer-perceptrons/dropout.ipynb

Data Preprocessing

- The pixel values in images must be scaled prior to given as input to deep neural networks for training or evaluation

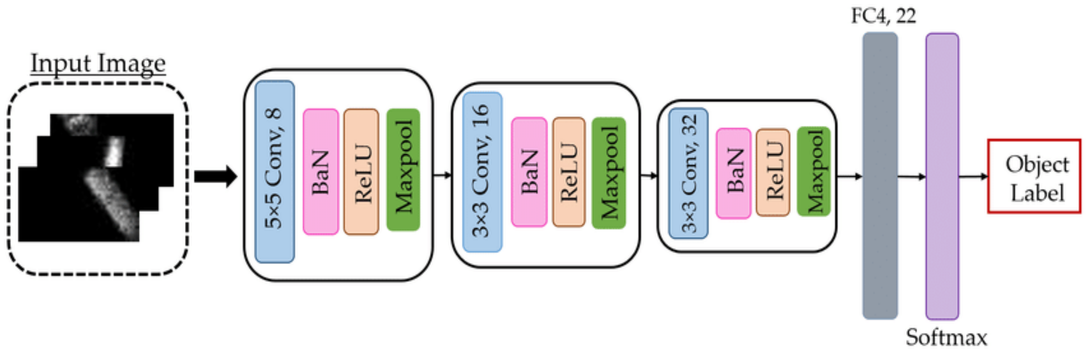
- Three main types of pixel scaling:
- **Pixel Normalization**: scale pixel values to the range 0-1
- **Pixel Centering**: scale pixel values to have a zero mean
- **Pixel Standardization**: scale pixel values to have a zero mean and unit variance

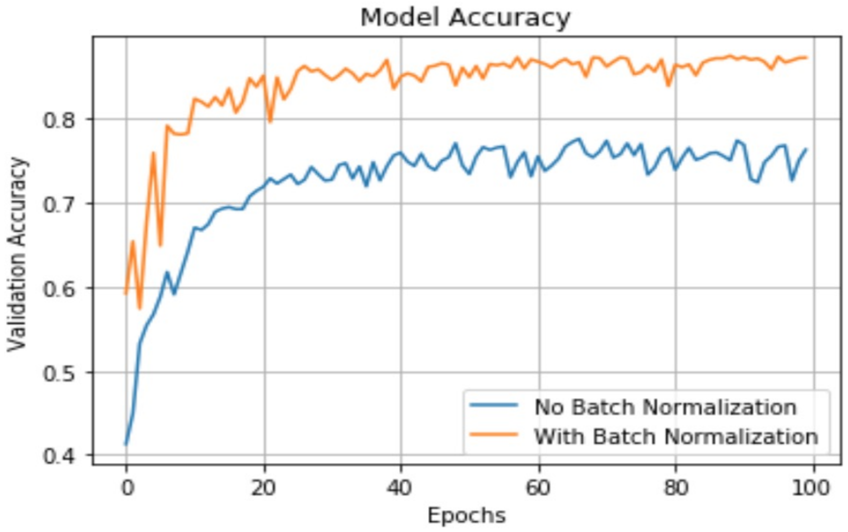


Credit: Stanford CS231n course slides.
Machine Learning Mastery: How to Normalize, Center, and Standardize Image Pixels in Keras

Batch Normalization

- ➢Enables stable training
- ➢Reduces the **internal covariate shift** (ICS)
- ➢ Accelerates the training process
- ➢ Reduces the dependence of gradients on the scale of the parameters





Source: LearnOpenCV: Batch Normalization in Deep Networks. https://learnopencv.com/batch-normalization-in-deep-networks/

Choice of Optimizers

➢Choosing right optimizer helps to update the model parameters and reducing the loss in much less effort

➢ Most DL frameworks supports various optimizers:

➢ Stochastic Gradient Descent (SGD)

➢ Momentum

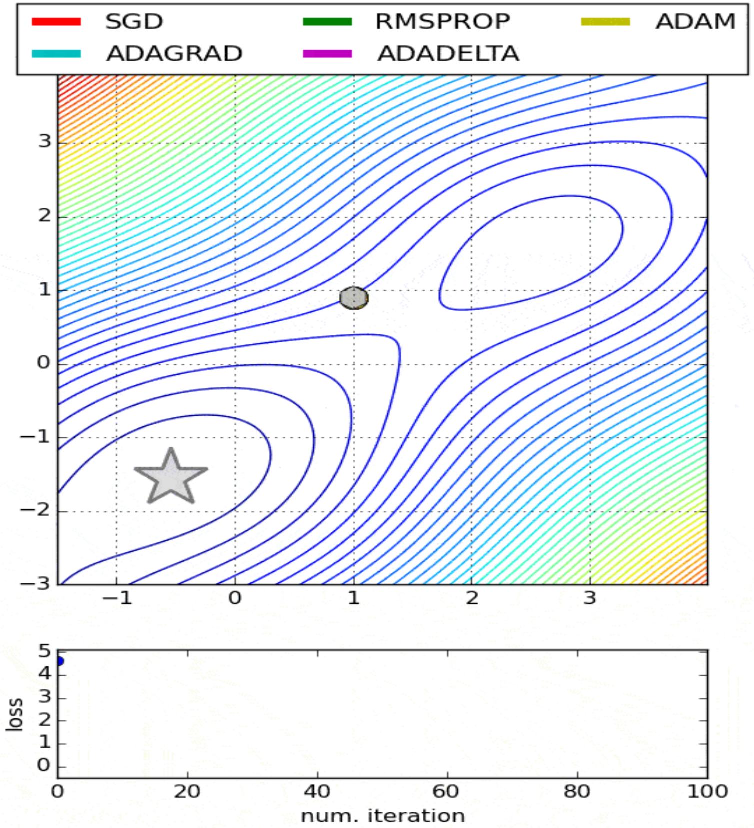
➢ Nesterov Accelerated Gradient

➢ AdaGrad

➢ AdaDelta

➢ Adam

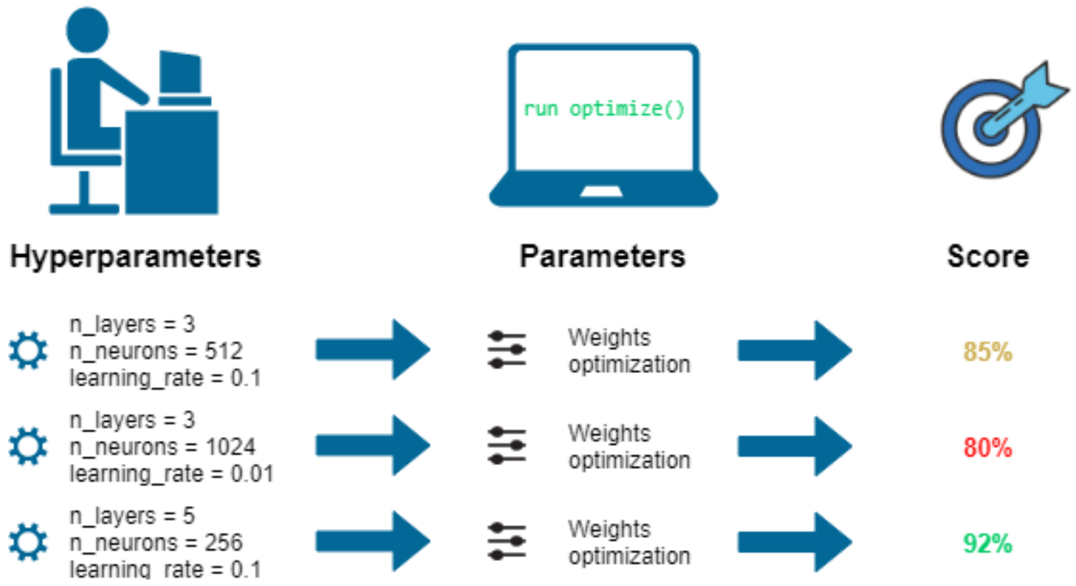
➢ RMSProp



Source: Towards Data Science. Various Optimization Algorithms For Training Neural Network https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Tuning Hyperparameters

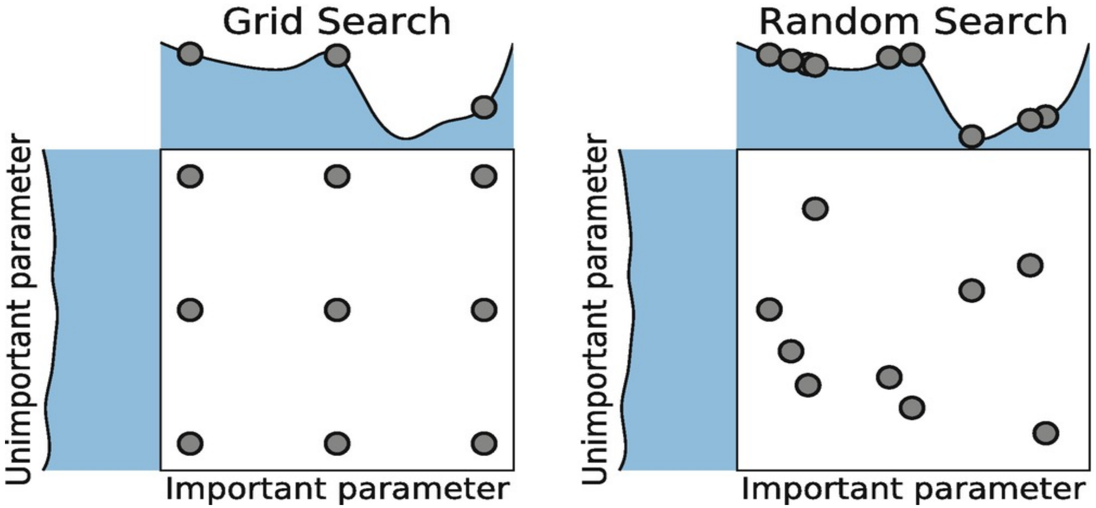
- ➢Hyperparameters are all parameters which can be arbitrarily set by the user before starting training
- ➢Hyperparameters are like knobs or dials of the network (model)
- ➢An optimization problem: We aim to find the right combinations of their values which can help us to find either the minimum (e.g., loss) or the maximum (e.g., accuracy) of a function
- ➢Many hyperparameters to tune:
  - ➢ Learning rate
  - ➢ No. of epochs
  - ➢ Dropout rate
  - ➢ Batch size
  - ➢ No. of hidden layers and units
  - ➢ Activation function
  - ➢ Weight initialization
  - ➢ …



Source: KDnuggets: Practical Hyperparameter Optimization. https://www.kdnuggets.com/2020/02/practical-hyperparameter-optimization.html

Tuning Hyperparameters strategies

- Random Guess
  - Simply use values from similar work
- Rely on your experience
  - Training DNNs is part art, part science
  - With experience sense of what works and what doesn’t
  - Still chances of being incorrect (suboptimal performance)



- Grid Search
  - Set up a grid of hyperparameters and train/test model on each of the possible combinations
- Automated hyperparameter tuning
  - Use of Bayesian optimization and Evolutionary Algorithms
  - Hyperopt: Distributed Asynchronous Hyperparameter Optimization

Deep Learning Frameworks/Packages

Caffe
Caffe2
Chainer
Microsoft Cognitive Toolkit
MATLAB
mxnet
PaddlePaddle
PyTorch
TensorFlow
torch
Wolfram Language
Keras

Key takeaways

- CNN Basics
  - Specialized for image-related tasks.
  - Utilize convolutional and pooling layers.
- Continuous improvement in CNN architectures and heuristics (tips and tricks)
  - always check literature to find state-of-the-art methods
- Convolutional Filters
  - Detect features using filters/kernels.
- Training
  - Backpropagation, optimization, loss functions.

Key takeaways

- Training methodology
  - Split data into training (70 %), validation (10 %), and testing (20 %)
  - Take care of data leakage (e.g., multiple samples of same patients should be in same set)
  - Check distribution of classes, work on balanced datasets (ideally)
  - Tune hyperparameters on validation set. Save best model and do inference on test set (once)
  - Don’t use off-the-shelf model blindly. Do ablation studies to know its working
- Data augmentation techniques are not standardized
  - Get input from experts to know what data augmentations make sense in the domain
  - For e.g., in chest X-rays we don’t want vertical flipping
- Results
  - Use multiple metrics rather a single metric to report results (often they are complementary)
  - Show both qualitative and quantitative results (e.g., image segmentation)

Acknowledgements

- Slides from
  - http://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/
  - https://towardsdatascience.com/convolutional-neural-networks-for-all-part-i-cdd282ee7947
  - https://www.slideshare.net/NirthikaRajendran/cnn-126271677
  - https://research.aimultiple.com/data-augmentation-techniques/
  - https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/slides-plain.html
- Some material drawn from referenced and associated online sources…
  - Credit: Srivastava et al., Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014 https://colab.research.google.com/github/d2l-ai/d2l-en-colab/blob/master/chapter_multilayer-perceptrons/dropout.ipynb
  - Credit: Stanford CS231n course slides. Machine Learning Mastery: How to Normalize, Center, and Standardize Image Pixels in Keras
  - Source: LearnOpenCV: Batch Normalization in Deep Networks. https://learnopencv.com/batch-normalization-in-deep-networks/
  - Source: Towards Data Science. Various Optimization Algorithms For Training Neural Network https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Example exam question

What is the purpose of convolutional layers in CNNs?

A. Reducing the size of the feature maps.

B. Calculating the dot product of the input and kernels.

C. Applying a nonlinear activation function.

D. Making the network learn faster.