

COMP9517: Computer Vision

Deep Learning

Part 1

Dong Gong

<https://donggong1.github.io/>
School of CSE, UNSW

Some slides are from Fei-Fei Li et al. and Francois Fleuret.

2023 T2 © UNSW CSE

Challenges in CV

Consider object detection as an example:

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



Content: [link](#)

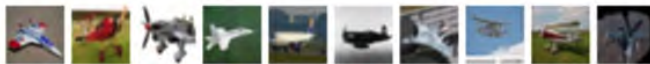


Linear Classifier for Image Classification

Image Classification

Dataset: CIFAR10 [Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.]

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck

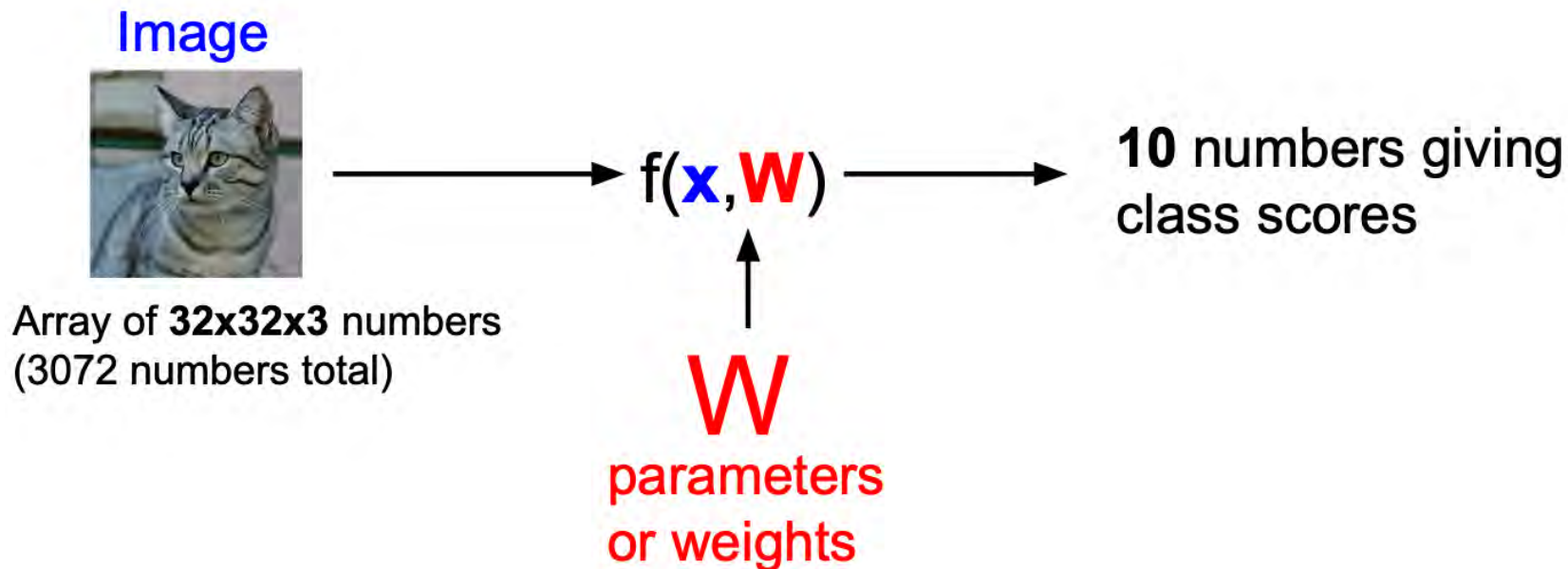


10 classes

50,000 training images

10,000 testing images

Image Classification



In this toy example, classification is performed directly on image pixels.

Image Classification

- Image classification with linear classifier

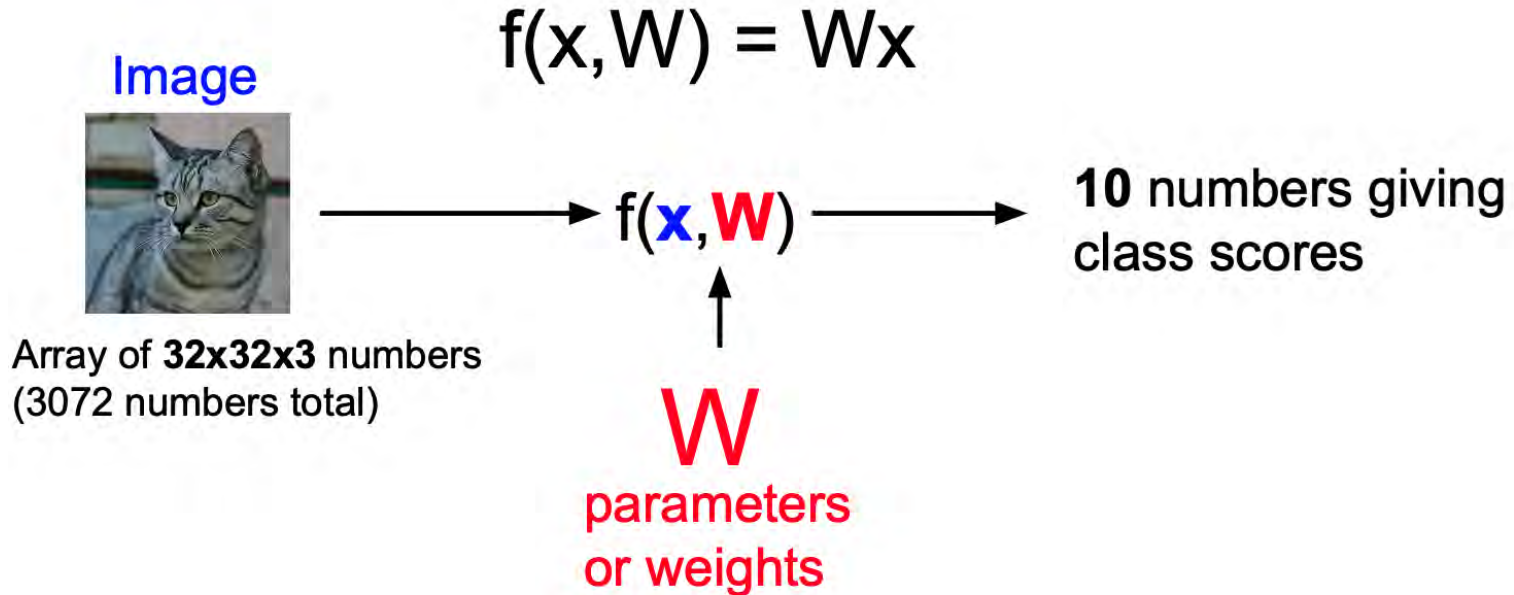


Image Classification

- Image classification with linear classifier

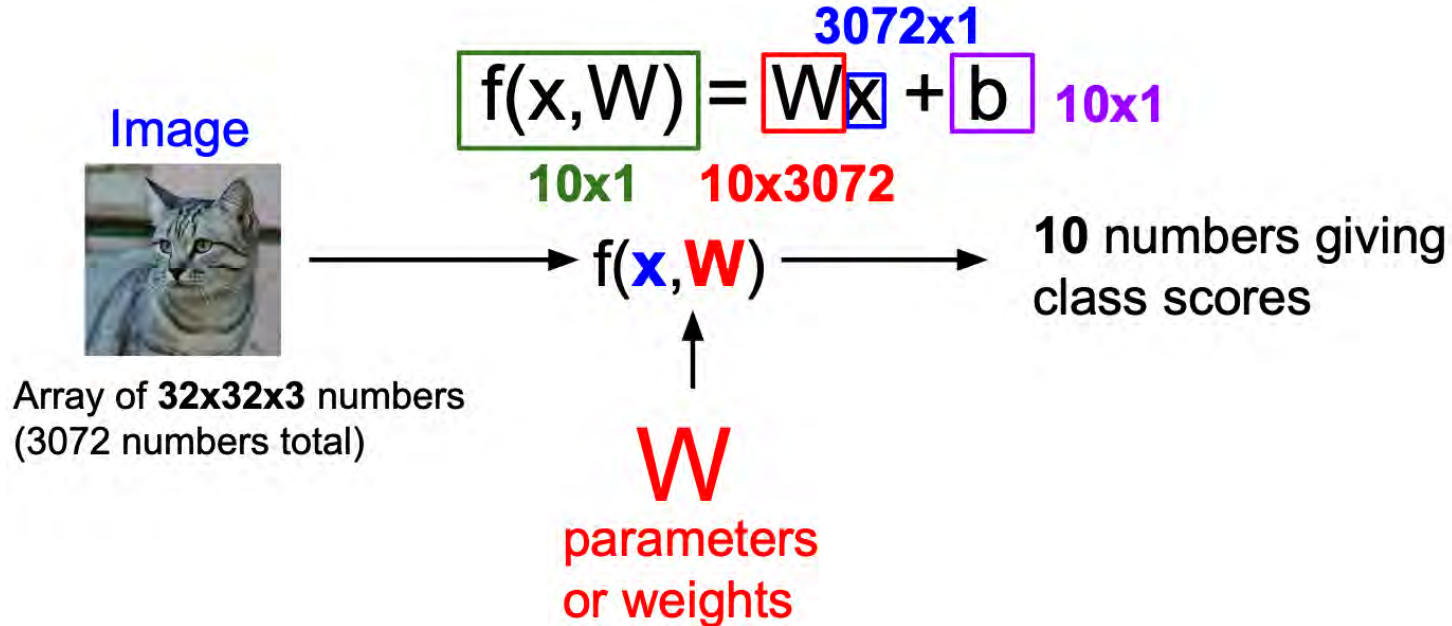


Image Classification

- An image example with 4 pixels and 3 classes.

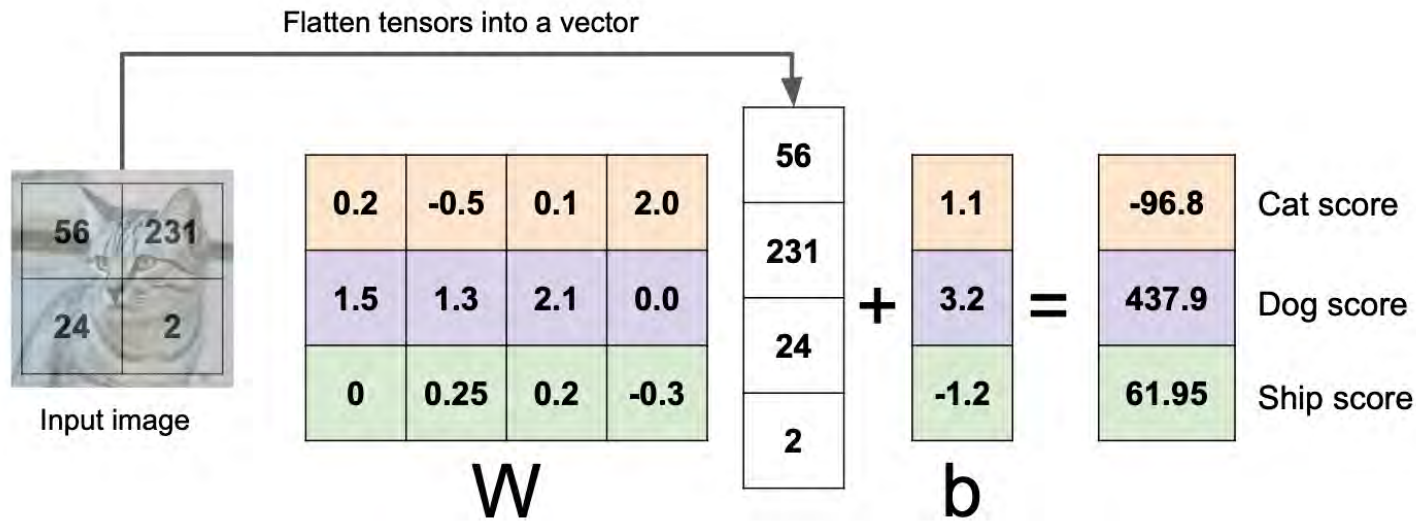
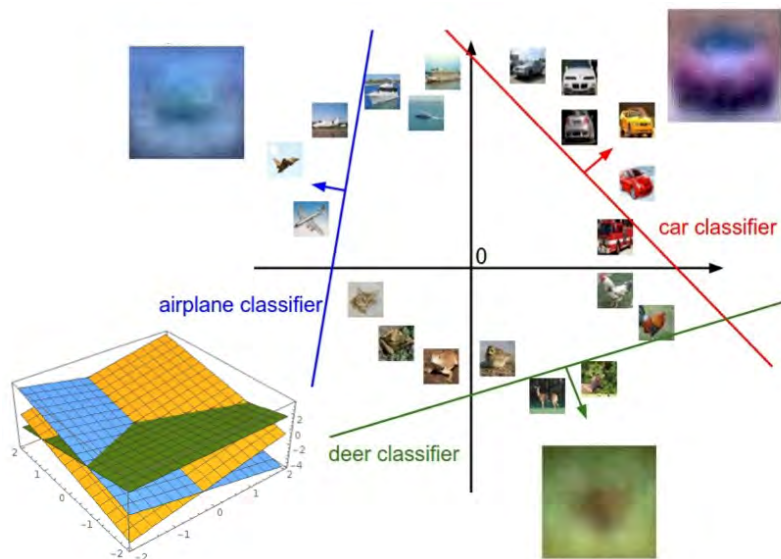


Image Classification

- Interpreting a linear classifier.



$$f(x, W) = Wx + b$$



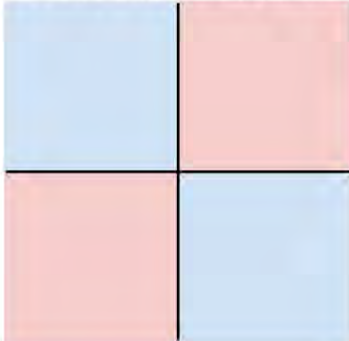
Array of **32x32x3** numbers
(3072 numbers total)

Image Classification

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

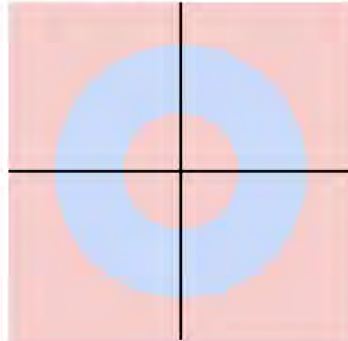
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



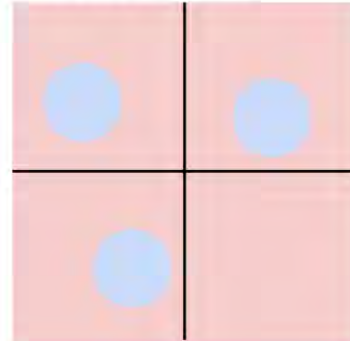
Class 1:
 $1 \leq L2 \text{ norm} \leq 2$

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)

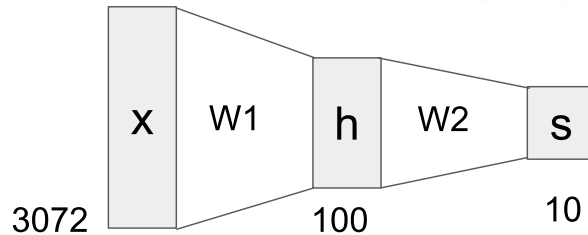
(In practice we will usually add a learnable bias at each layer as well)

Neural Networks

- 2 layers
- Also called as **fully connected network**
- **Fully connected 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 or 3-layer Neural Network $f = W_2 \max(0, W_1 x)$

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

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

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

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

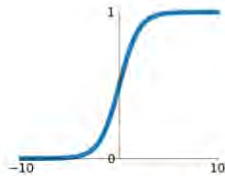
- What if without the activation function?
 - The model will be linear.

Neural Networks

- Activation functions
 - Non-linear functions

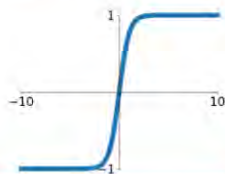
Sigmoid

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



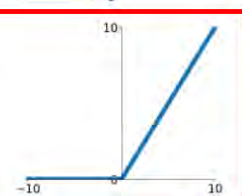
tanh

$$\tanh(x)$$



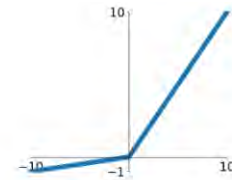
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

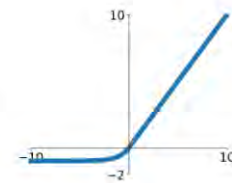


Maxout

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

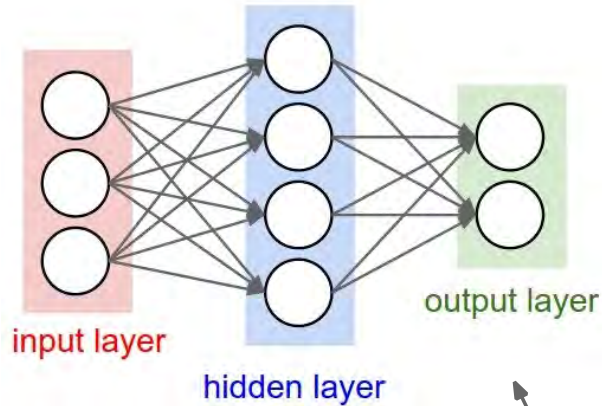
ELU

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

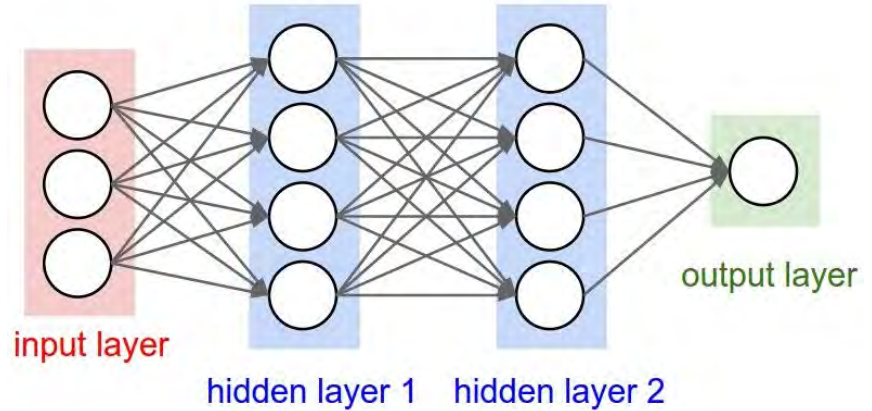


Neural Networks

- Architectures (for MLP)



“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

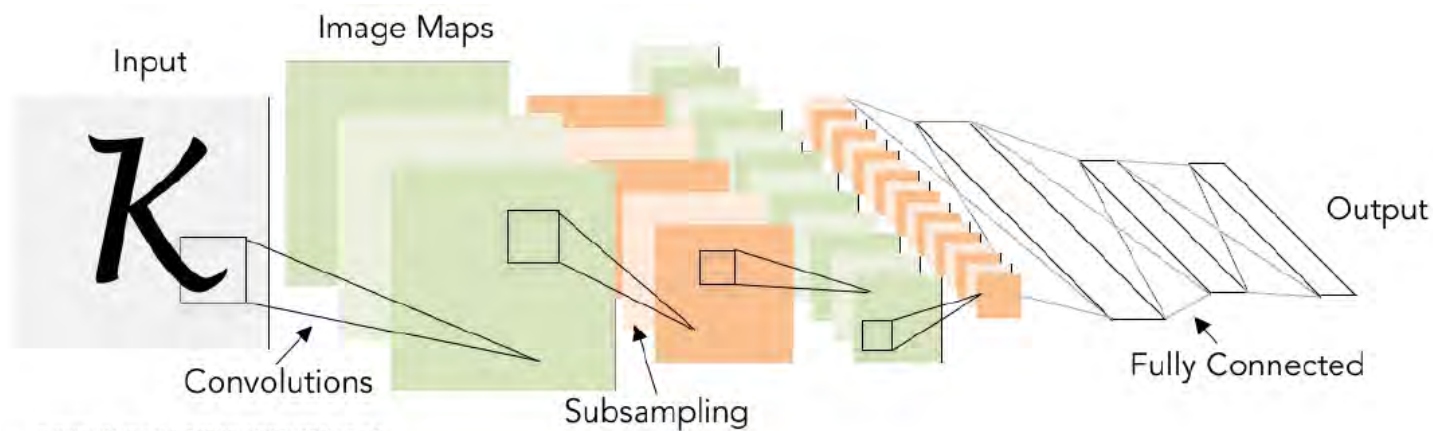


“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

“Fully-connected” layers

Neural Networks

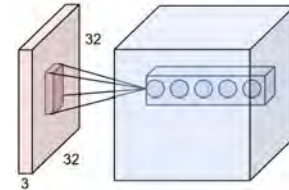
- Architectures (for CNNs)



From Neural Networks to “Deep Learning”

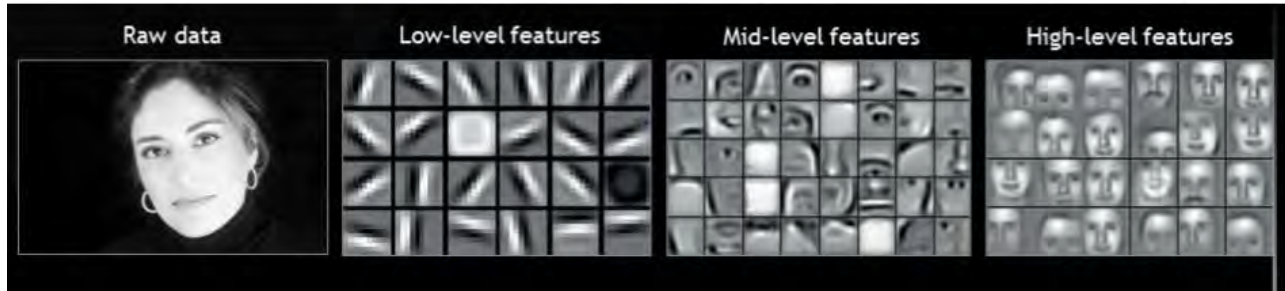
Deep Learning

- 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)**
 - Recurrent Neural Networks
 - Auto-encoders
 - Restricted Boltzmann Machines
 - Transformer

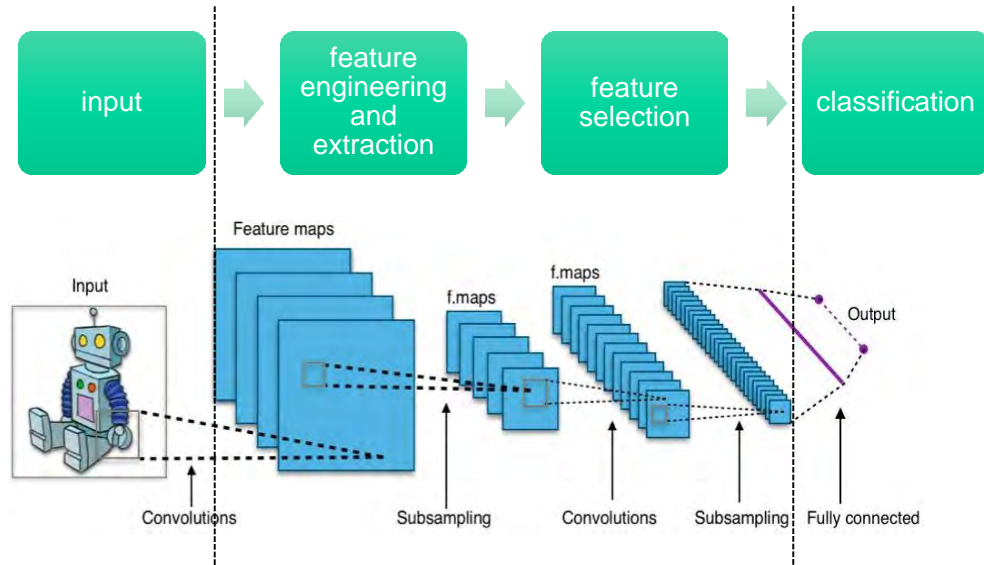


Traditional Approach vs DL

- 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.



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

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.

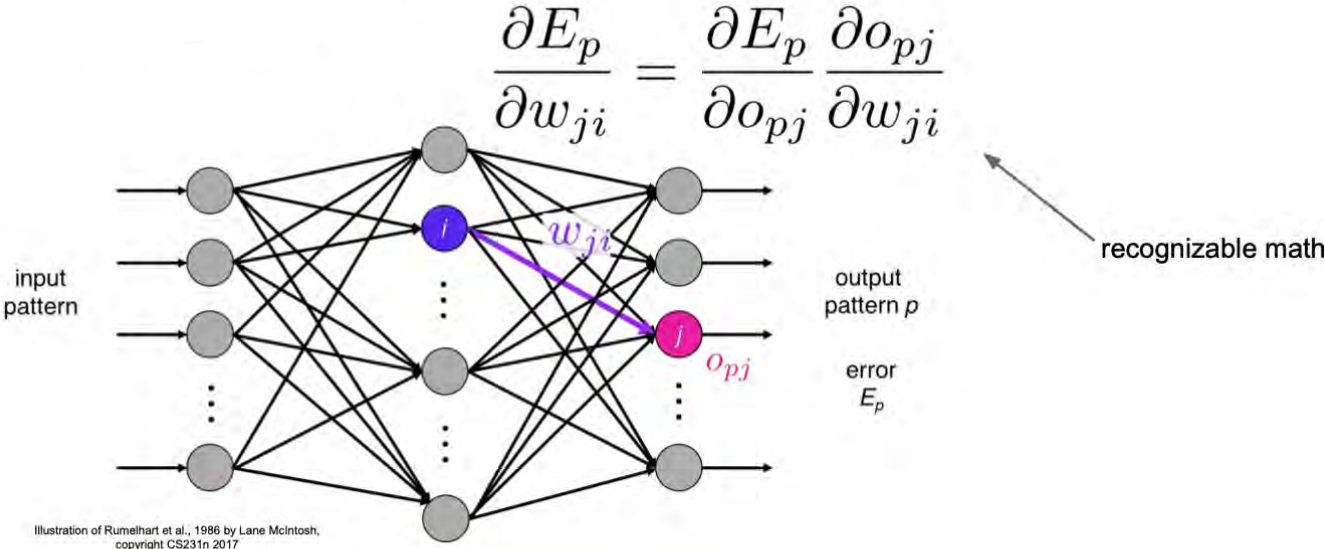
recognized
letters of the alphabet

Frank Rosenblatt, ~1957: Perceptron



This image by Rocky Acosta is licensed under [CC-BY 3.0](https://creativecommons.org/licenses/by/3.0/)

From Neural Networks to “Deep Learning”



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

Imagenet classification with deep convolutional neural networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

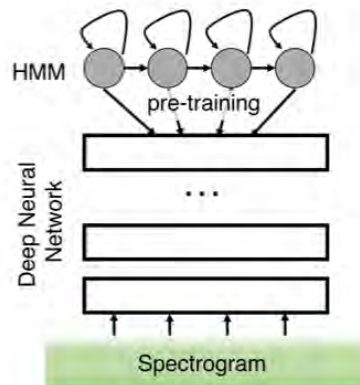
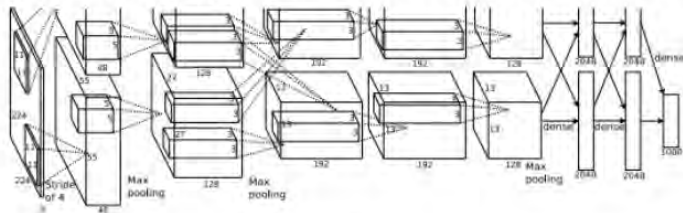


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

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

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

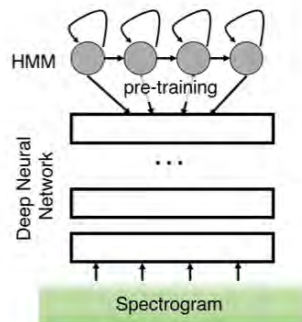
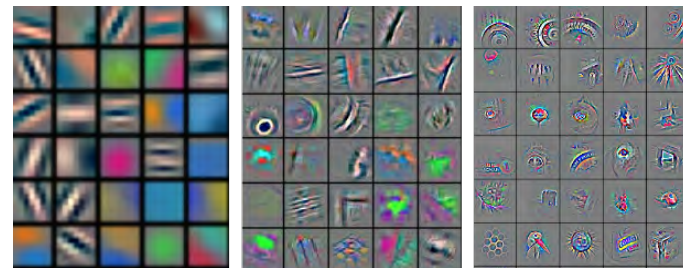


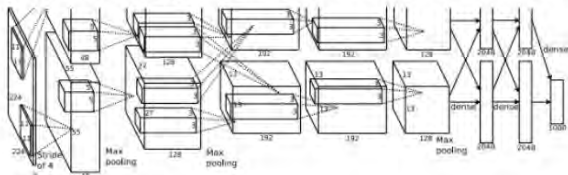
Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017



Visual features extracted in different layers in CNN

Imagenet classification with deep convolutional neural networks

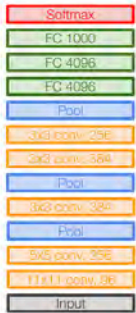
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



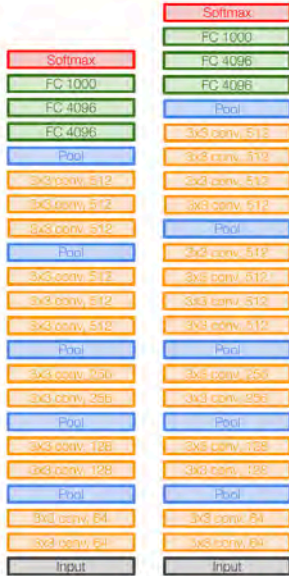
Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.



From Neural Networks to “Deep Learning”



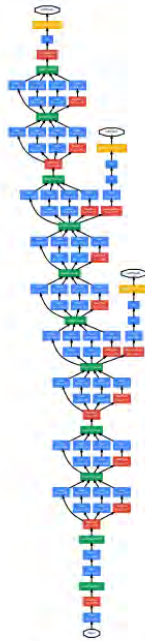
AlexNet



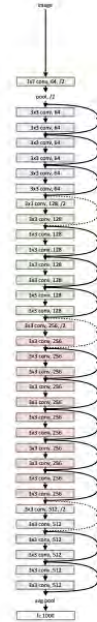
VGG16

VGG19

Simonyan and Zisserman, 2014



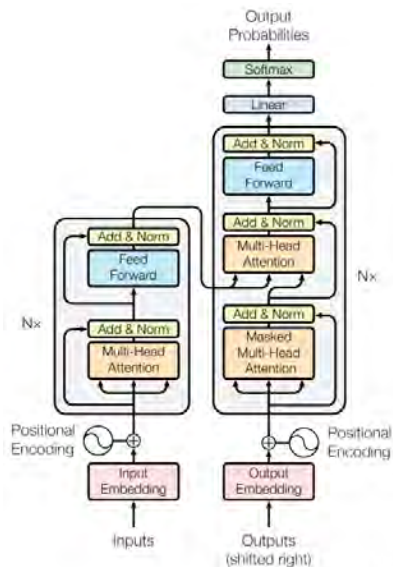
GoogleNet (Szegedy et al., 2015)



ResNet (He et al., 2015)

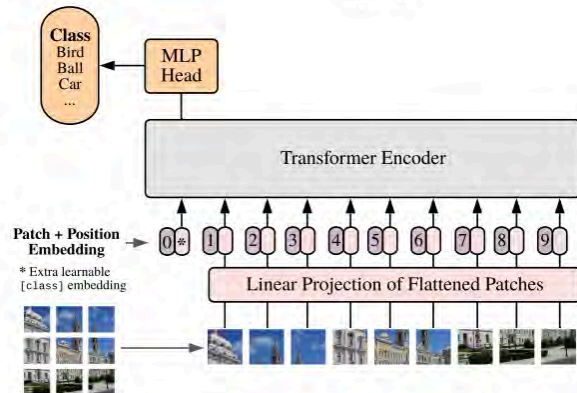
From Neural Networks to “Deep L

Transformer



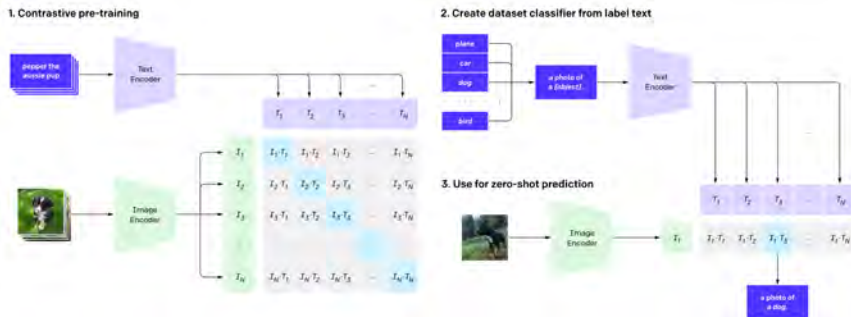
(Vaswani et al., 2017)

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).

CLIP (Contrastive Language–Image Pre-training)



<https://openai.com/research/clip>

From Neural Networks to “Deep Learning”

- DL is everywhere

Classification



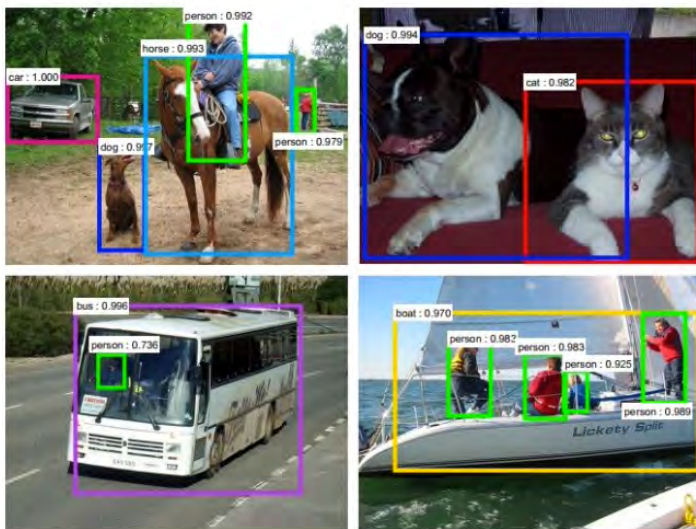
Retrieval



From Neural Networks to “Deep Learning”

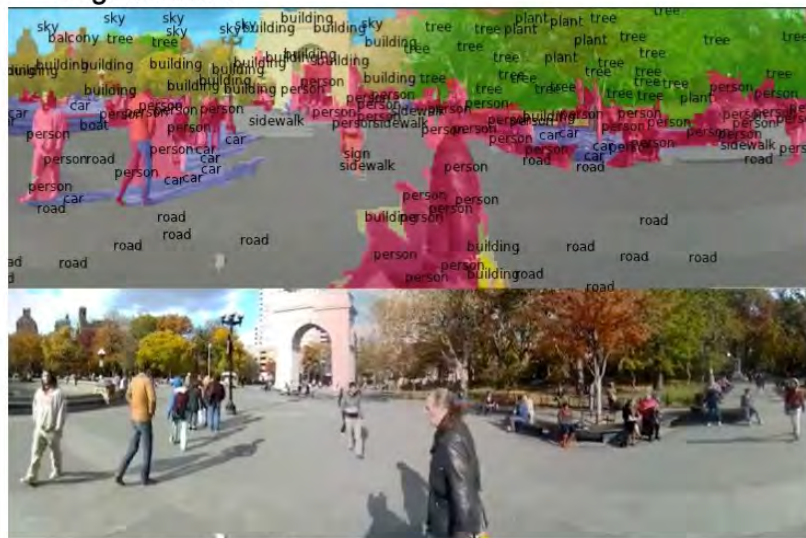
- DL is everywhere

Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, 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]

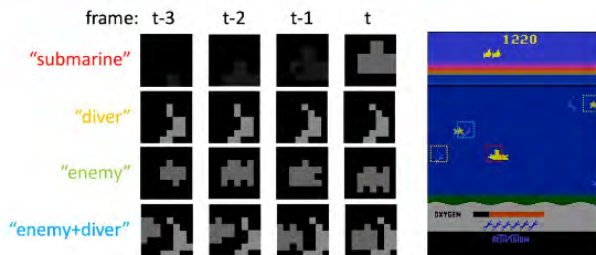
From Neural Networks to “Deep Learning”

- DL is everywhere

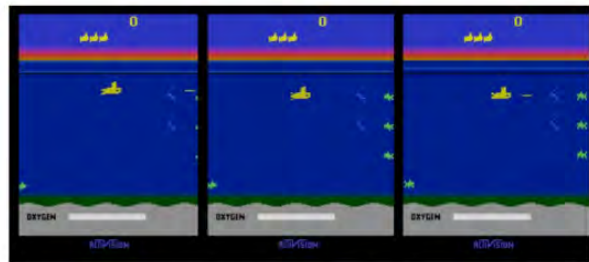


Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]



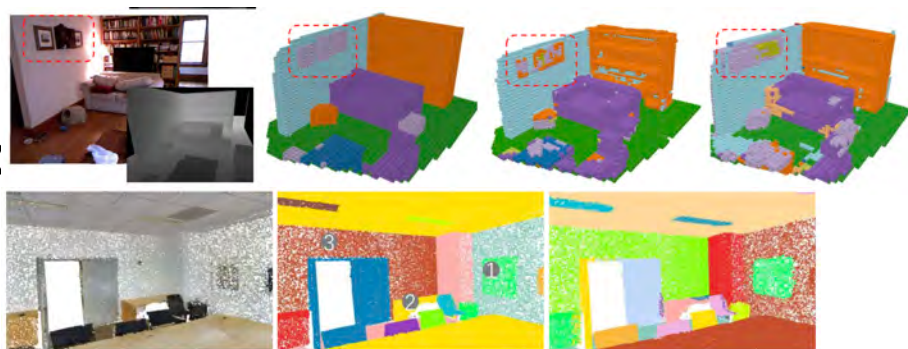
[Guo et al. 2014]



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

From Neural Networks to “Deep

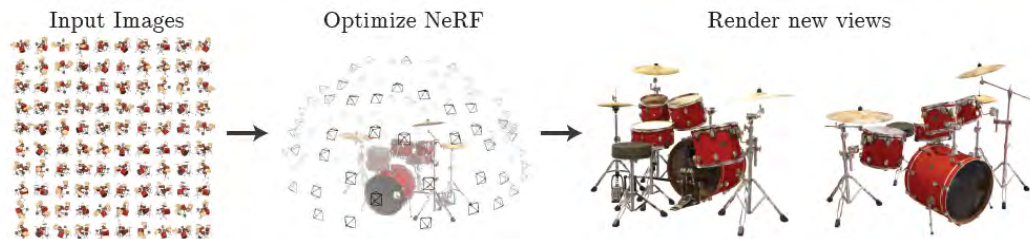
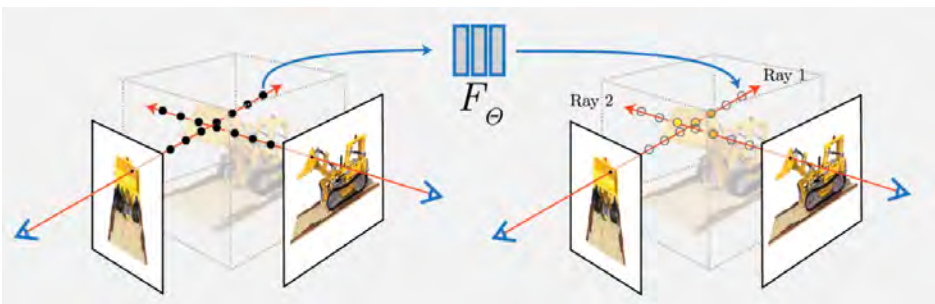
- DL is everywhere



3D vision understanding

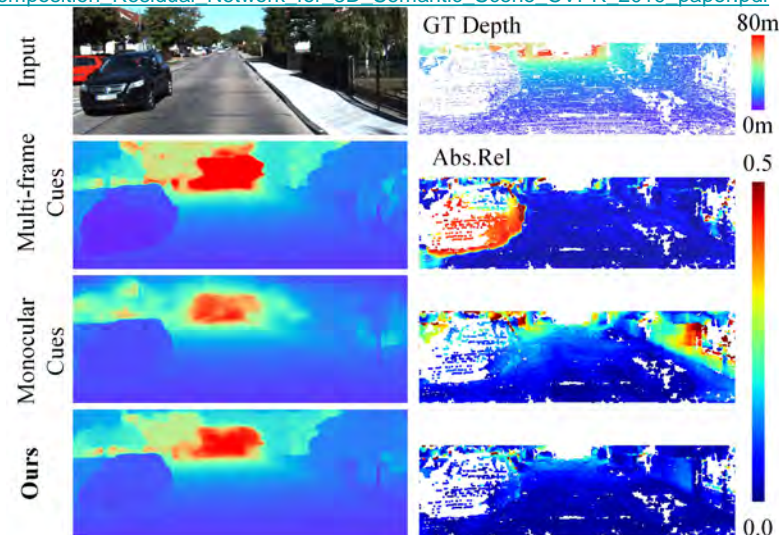
<https://arxiv.org/pdf/2001.01349.pdf>

http://openaccess.thecvf.com/content_CVPR_2019/papers/Li_RGBD_Based_Dimensionality_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



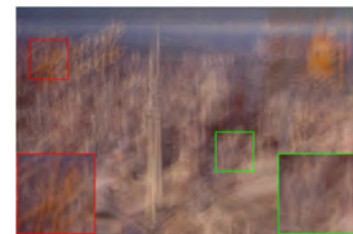
self-driving cars

Photo by Lane Michotsh. Copyright CS231n 2017.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokah 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



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



Whale recognition, Kaggle Challenge



Mnih and Hinton, 2010

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

From Neural Networks to “Deep Learning”

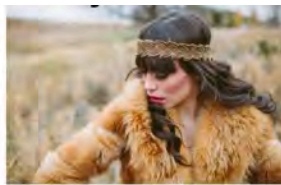
- DL is everywhere



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

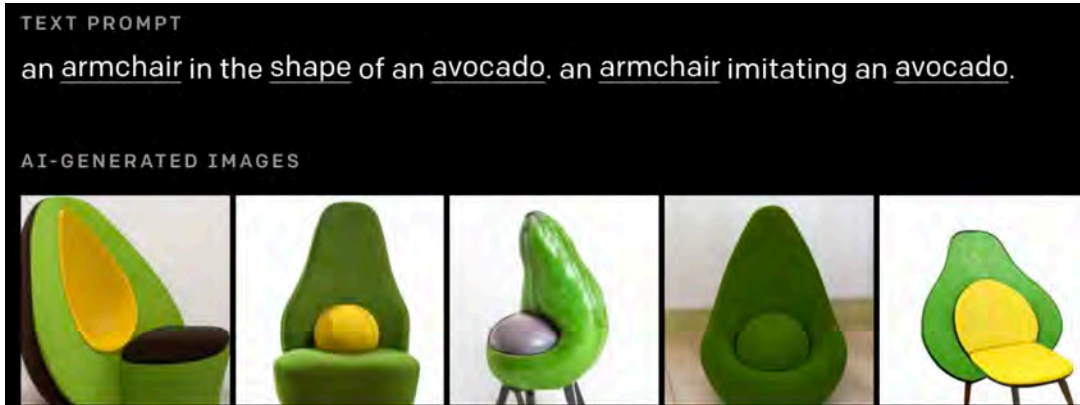


Image Captioning. Vinyals et al, 2015 Karpathy and Fei-Fei, 2015

Vision question answering (VQA)

From Neural Networks to “Deep Learning”

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



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

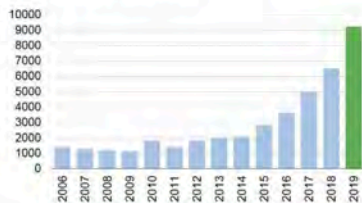


Generated by DALL·E 2

The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023



CVPR Attendance Trend



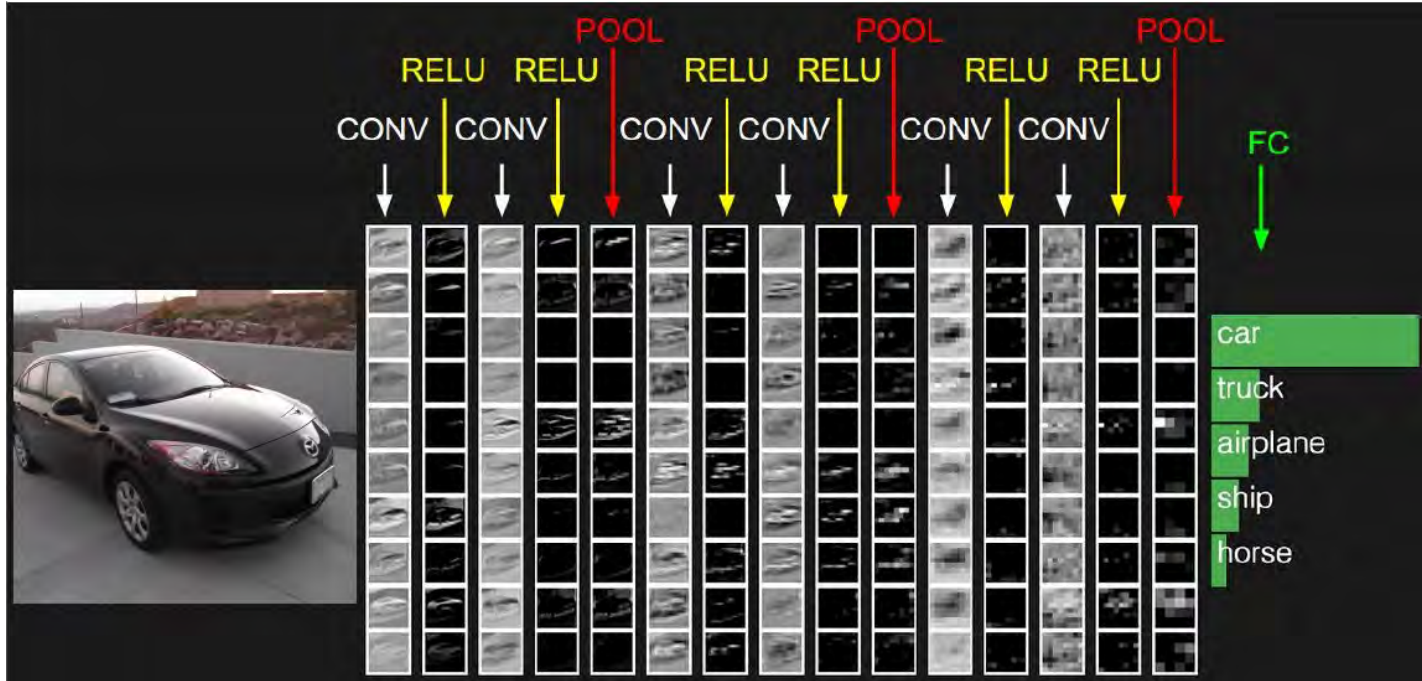
>9k submissions, 2,360 accepted papers



3 major international CV conferences: CVPR, ICCV, ECCV; and others
Top machine learning conferences with CV research: NeurIPS, ICML, ICLR ...
Robotic vision: RSS, ICRA, IROS.

Convolutional Neural Network (CNN), from MLP to CNN

An overview of a CNN



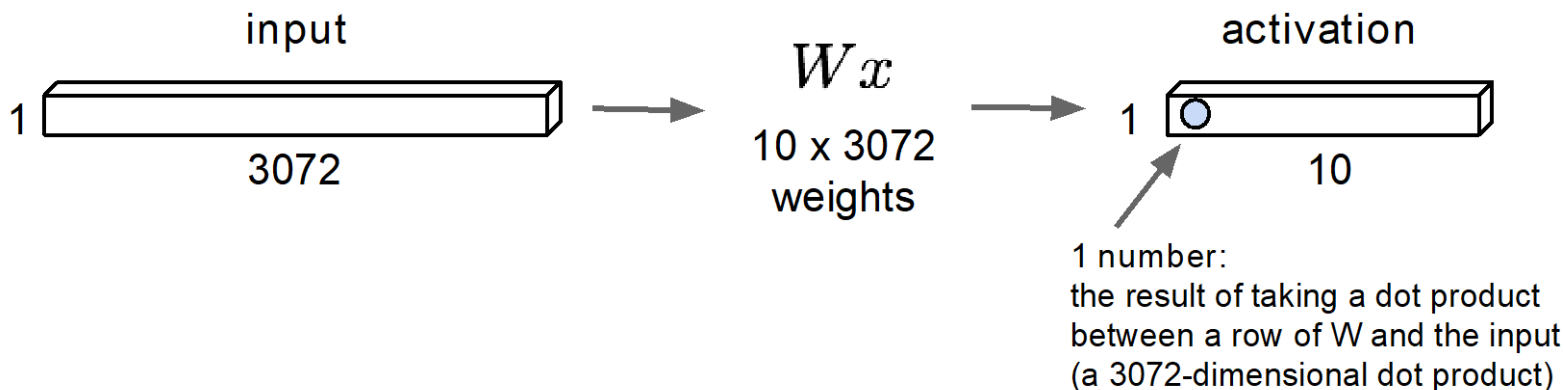
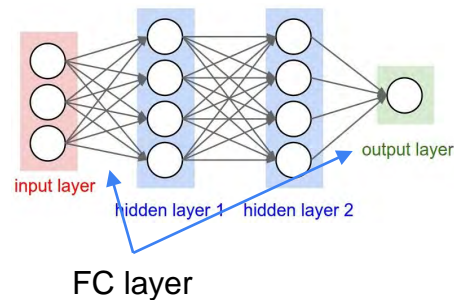
CNNs

- 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)

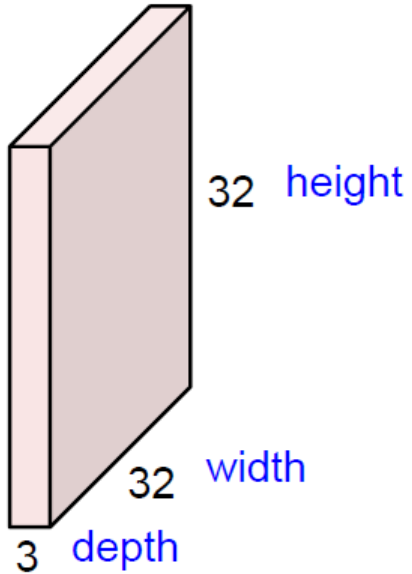
- Recap: fully connected (FC) layer
- A linear model, **not CNNs**.
- A component of CNNs

32x32x3 image -> stretch to 3072 x 1



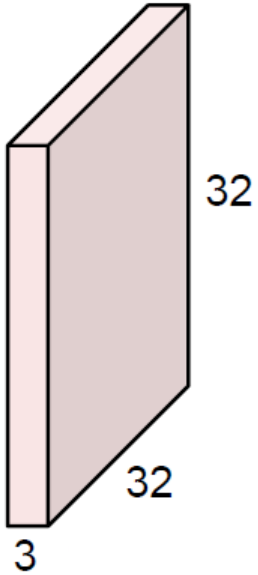
Convolution Layer (2D)

32x32x3 image -> preserve spatial structure



Convolution Layer

32x32x3 image



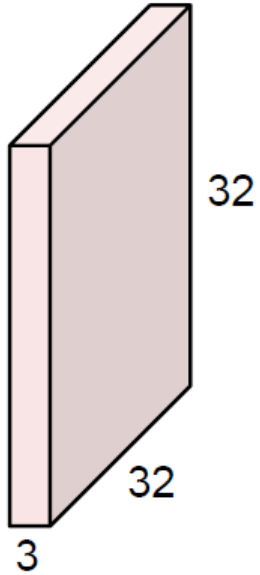
5x5x3 filter



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

Convolution Layer

32x32x3 image



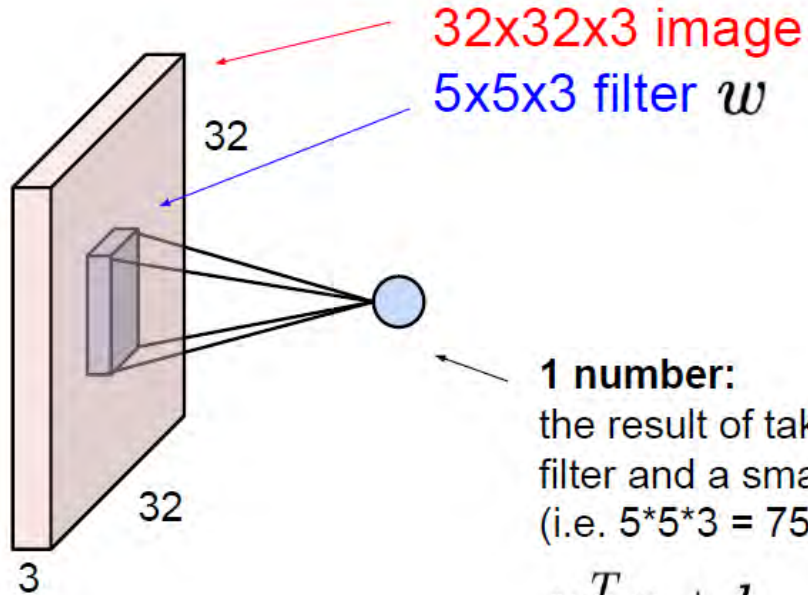
Filters always extend the full depth of the input volume

5x5x3 filter



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

Convolution Layer

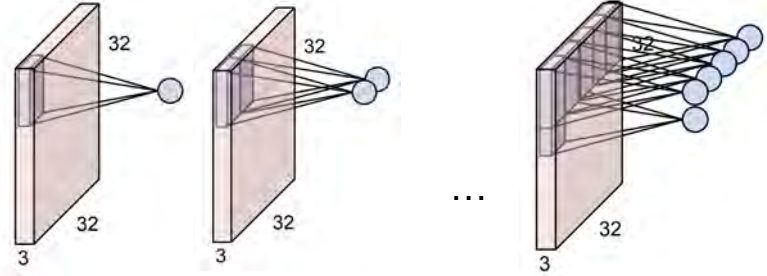


32x32x3 image
5x5x3 filter w

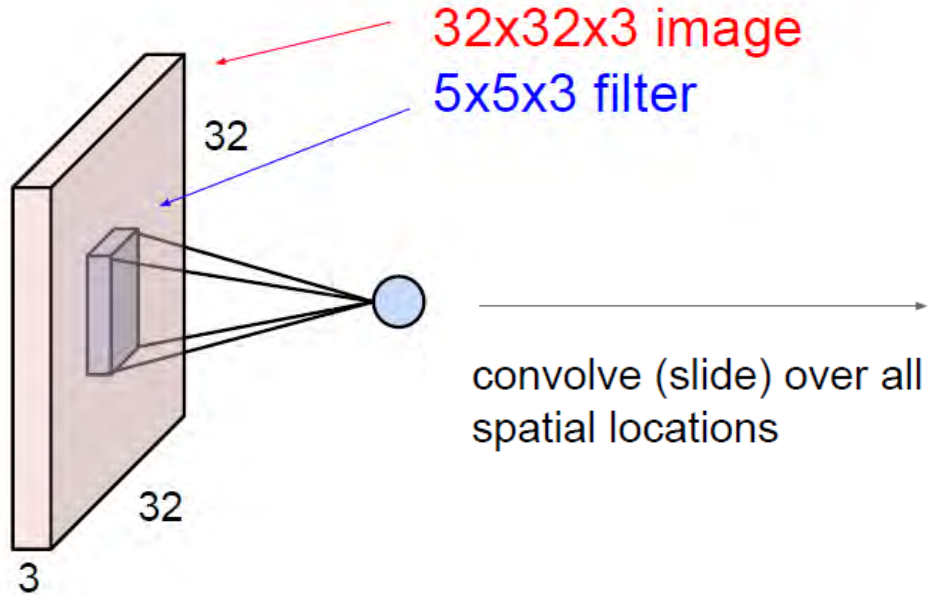
1 number:

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)

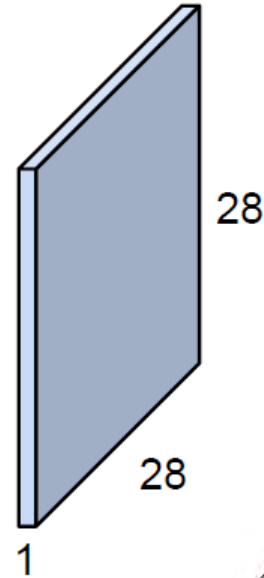
$$w^T x + b$$



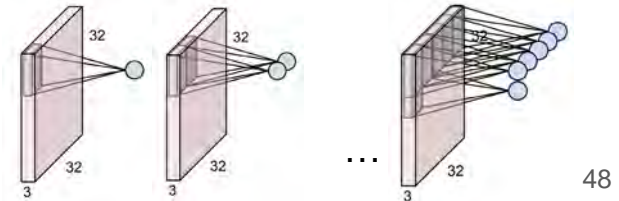
Convolution Layer



activation map



Noticed the difference on size?
Why?

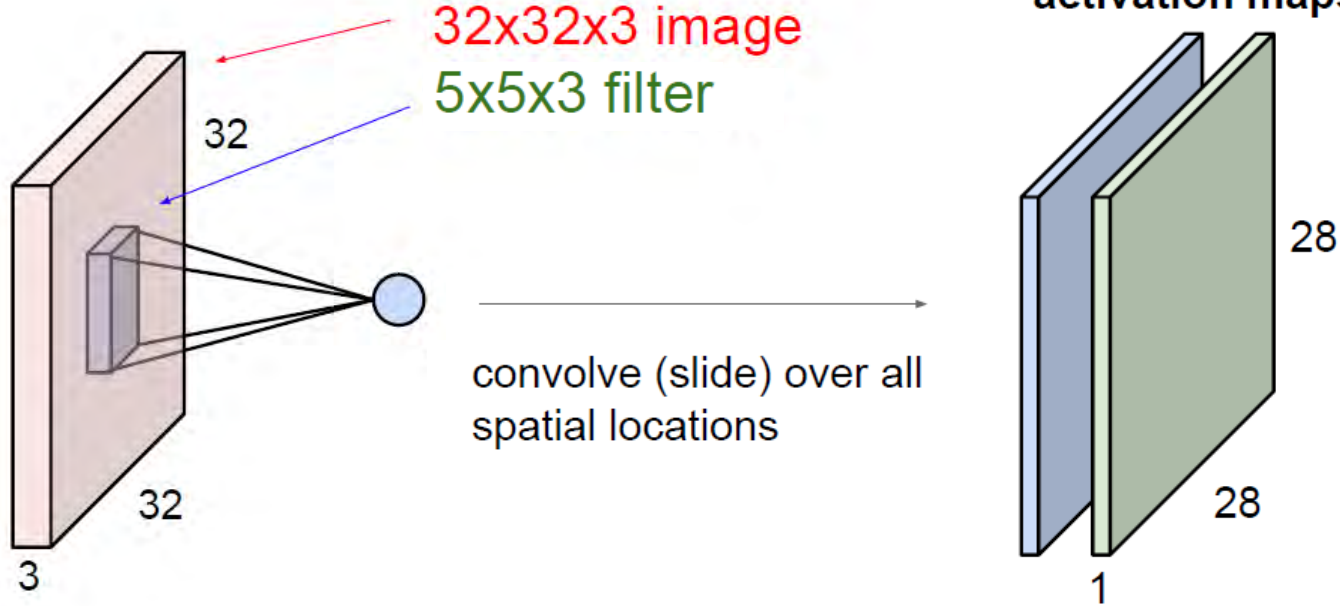


CNN: Convolutional Layer

- 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.

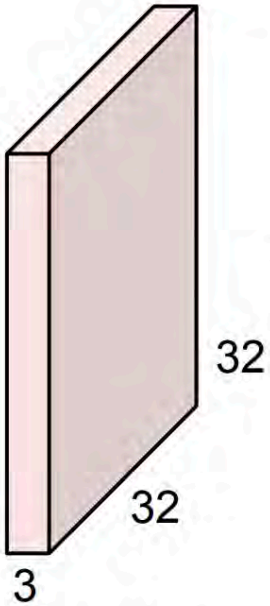
Convolution Layer

consider a second, **green** filter
activation maps

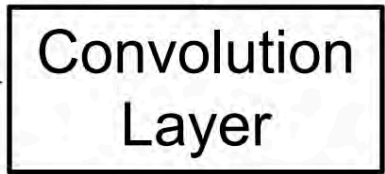


Convolution Layer

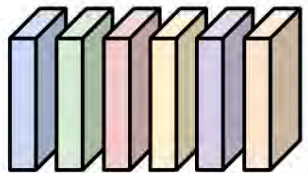
3x32x32 image



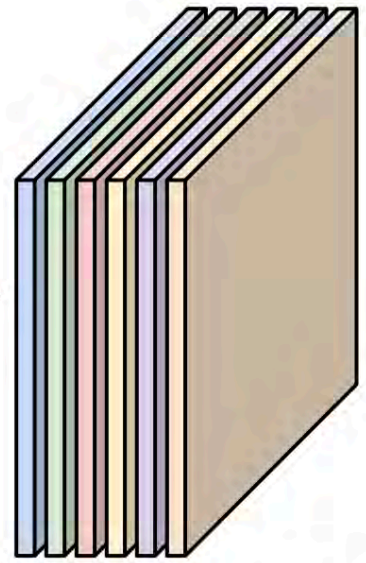
Also 6-dim bias vector:



6x3x5x5 filters



28x28 grid, at each point a 6-dim vector

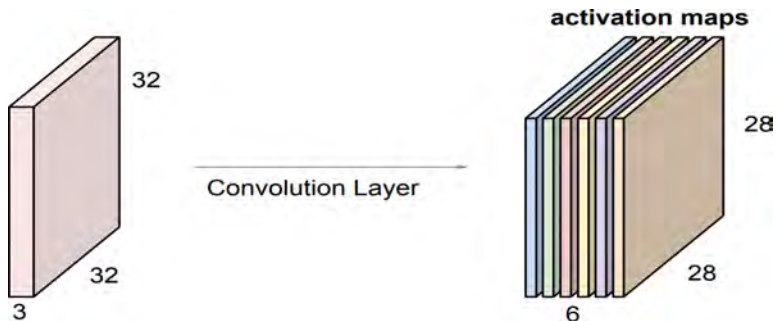


Stack activations to get a 6x28x28 output image!

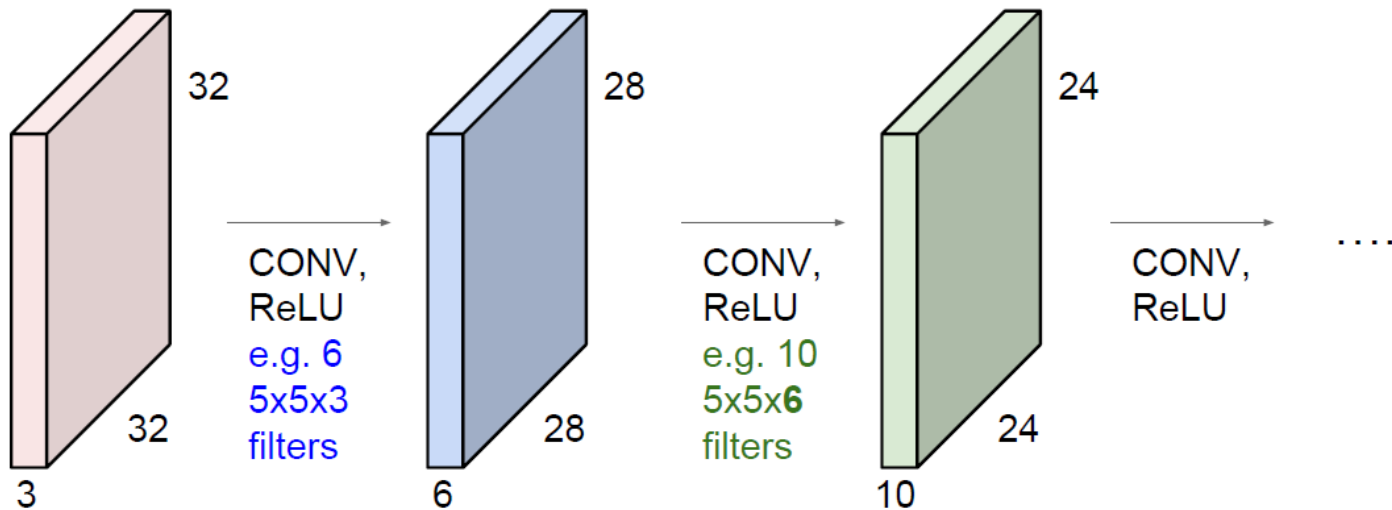
CNN: Convolutional Layer

- 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)

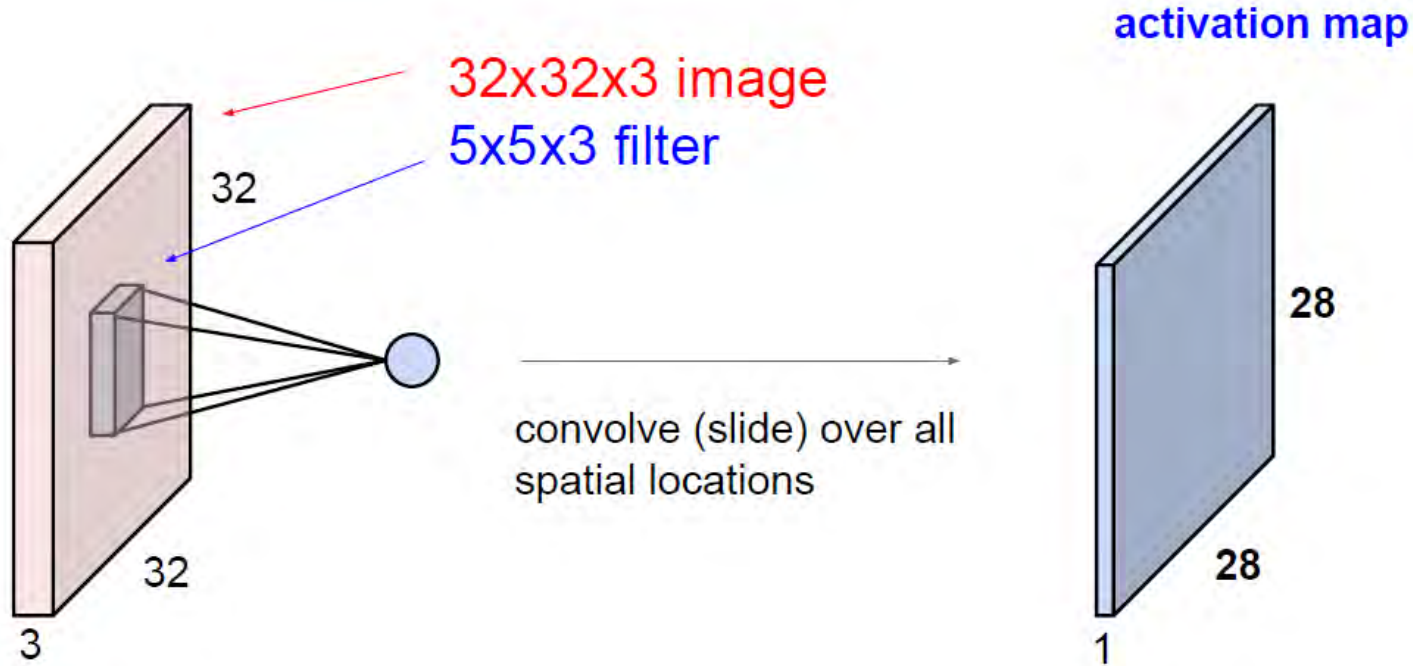
With 6 filters, we get
6 activation maps



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



A closer look at spatial dimensions:

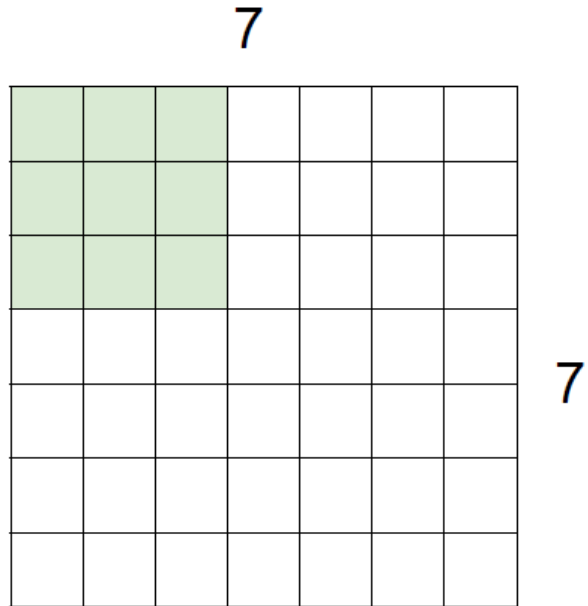


CNN: Convolutional Layer

Local Connectivity

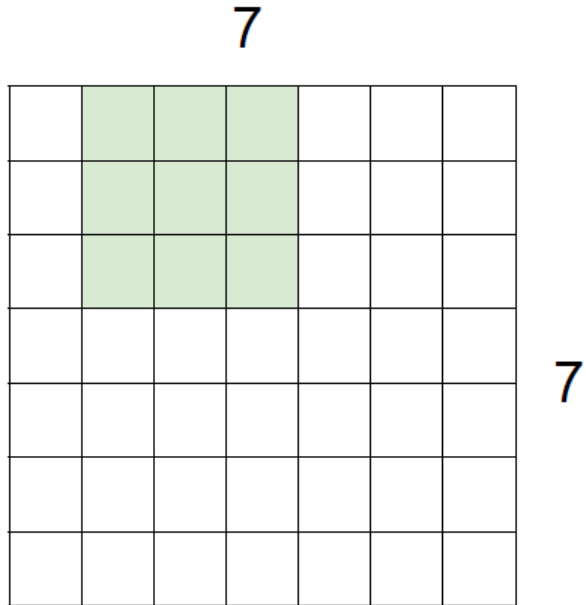
- As we have realized by now, it is impractical to use fully connected networks when dealing with high dimensional images/data
- Hence the concept of local connectivity: each neuron only connects to a local region of the input volume.
- The spatial extent of this connectivity is a concept called ***receptive 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:



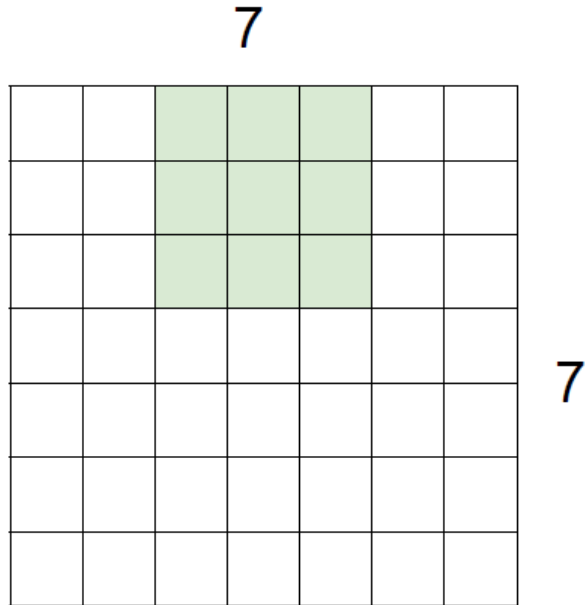
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



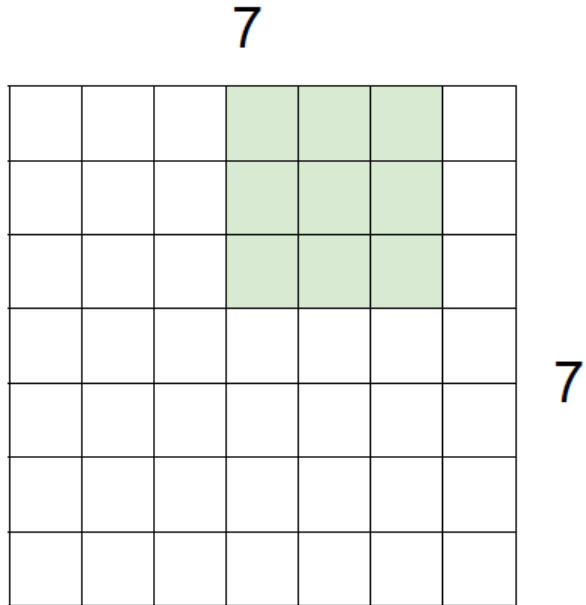
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



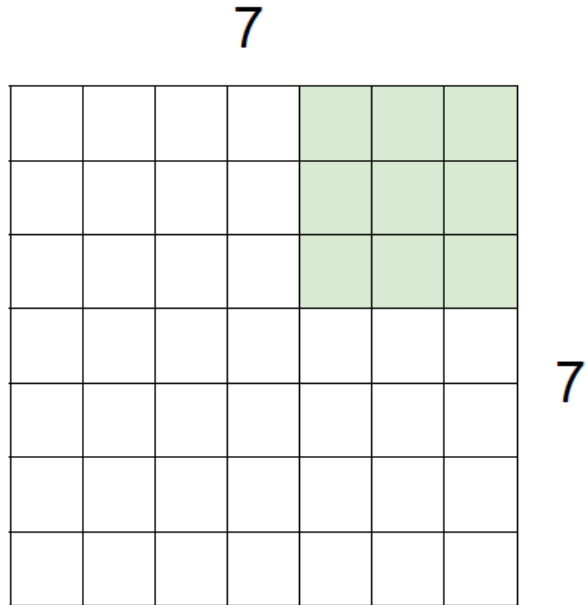
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

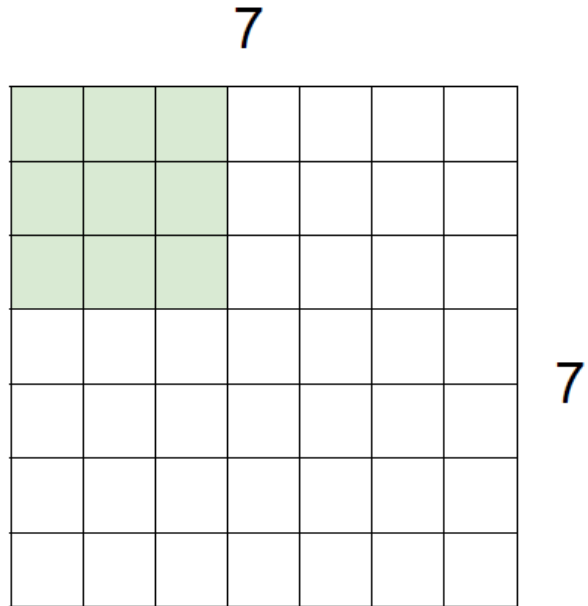
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

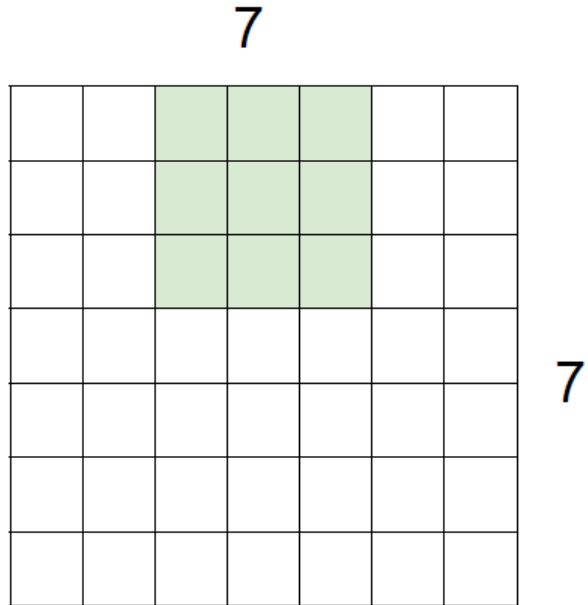
=> 5x5 output

A closer look at spatial dimensions:



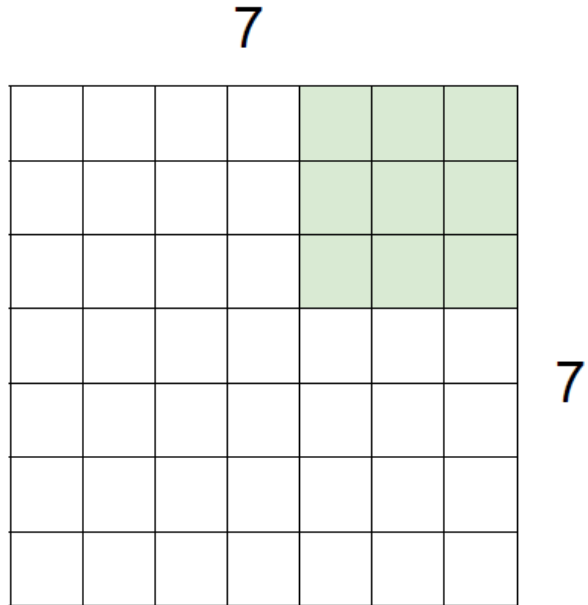
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



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

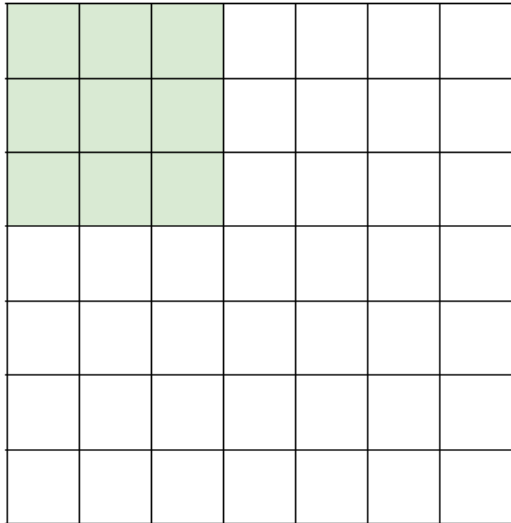
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:

7

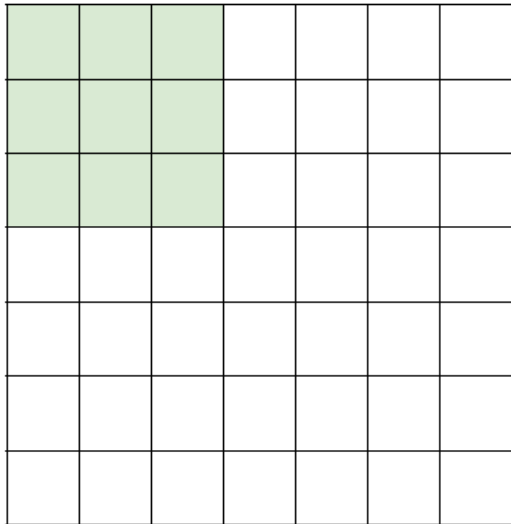


7

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

A closer look at spatial dimensions:

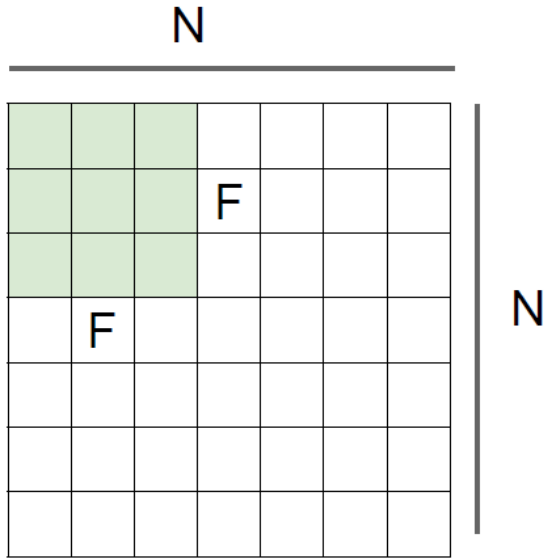
7



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

7

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



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7, F = 3$:

stride 1 $\Rightarrow (7 - 3) / 1 + 1 = 5$

stride 2 $\Rightarrow (7 - 3) / 2 + 1 = 3$

stride 3 $\Rightarrow (7 - 3) / 3 + 1 = 2.33 \text{ :}\backslash$

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

(recall:)

$$(N - F) / \text{stride} + 1$$

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

(recall:) N will not be 7 after padding.

$$(N - F) / \text{stride} + 1$$

7x7 output!

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

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 $F \times F$, and zero-padding with $(F-1)/2$. (will preserve size spatially)

e.g. $F = 3 \Rightarrow$ zero pad with 1

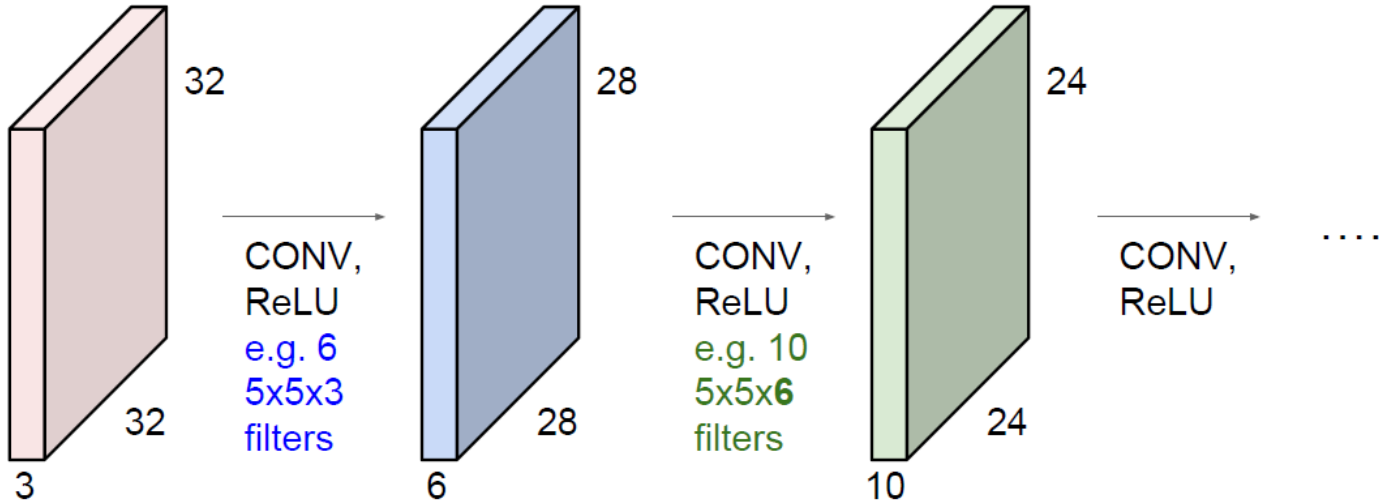
$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

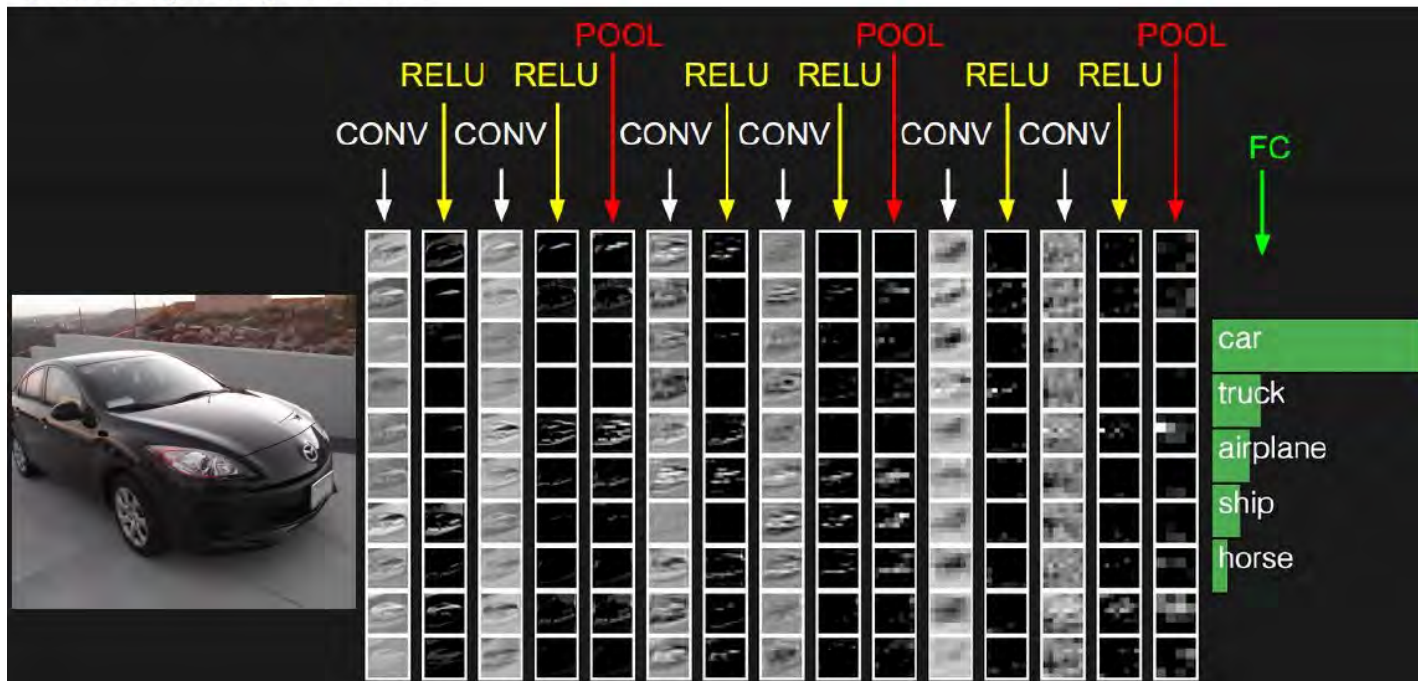
Other padding operations: replication padding, reflection padding ...

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.

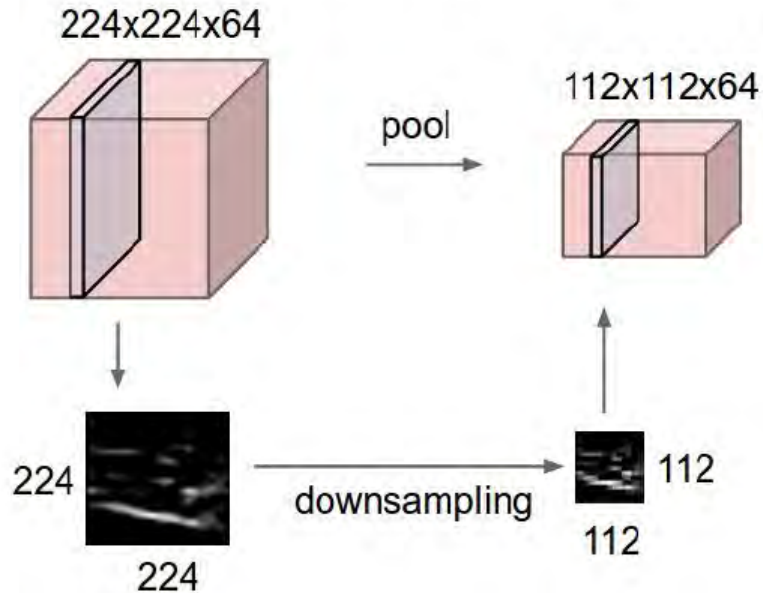


two more layers to go: POOL/FC



Pooling layer

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



Max Pooling

Single depth slice

x ↑

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

→ y

max pool with 2x2 filters
and stride 2



6	8
3	4

Other pooling operations: average/median pooling ...

Pooling layer: summary

Let's assume input is $W_1 \times H_1 \times C$

Conv layer needs 2 hyperparameters:

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

This will produce an output of $W_2 \times H_2 \times C$ where:

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

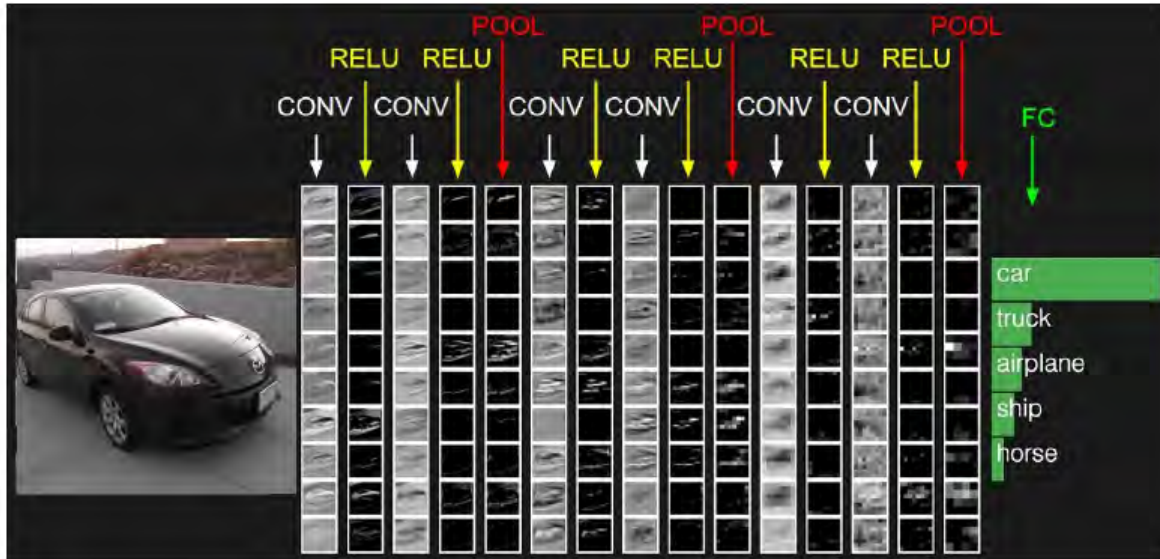
Number of parameters: 0

CNN: Pooling Layer

- The function of pooling layer
 - to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and
 - hence to also control overfitting
- The Pooling Layer operates
 - independently on every depth slice of the input and resizes it spatially, typically using the MAX operation (ie: max pooling)
 - The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2, which downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations

Fully Connected Layer (FC layer)

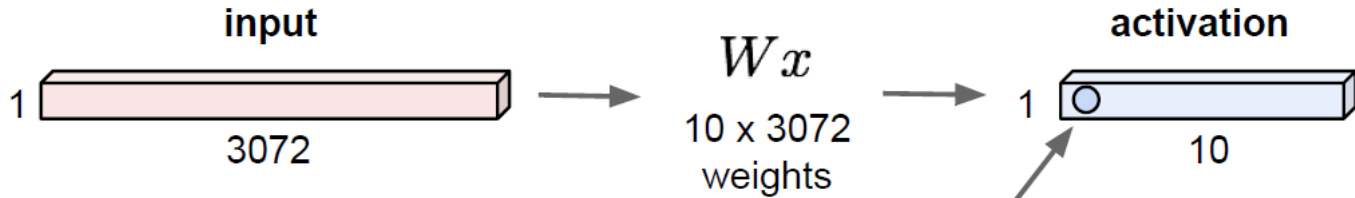
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron
looks at the full
input volume



1 number:
the result of taking a dot product
between a row of W and the input
(a 3072-dimensional dot product)

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, $0 \leq K \leq 2$.
 - but recent advances such as ResNet/GoogLeNet have challenged this paradigm

Training CNNs/Deep Neural Networks

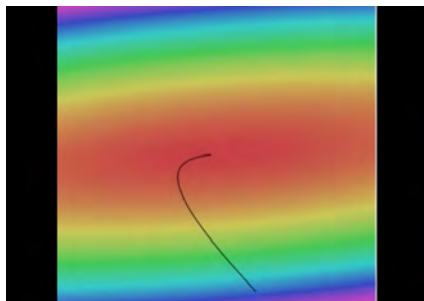
Some slides at this part are from Dr Sonit Singh sonit.singh@unsw.edu.au

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

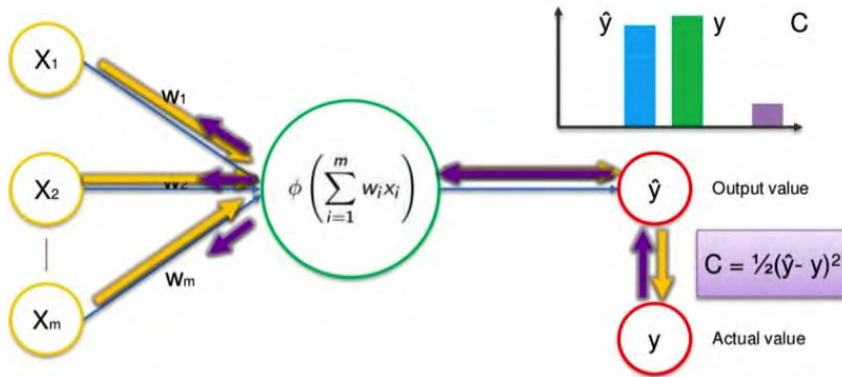
$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = - \sum_i y_i \log \hat{y}_i$$

- 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

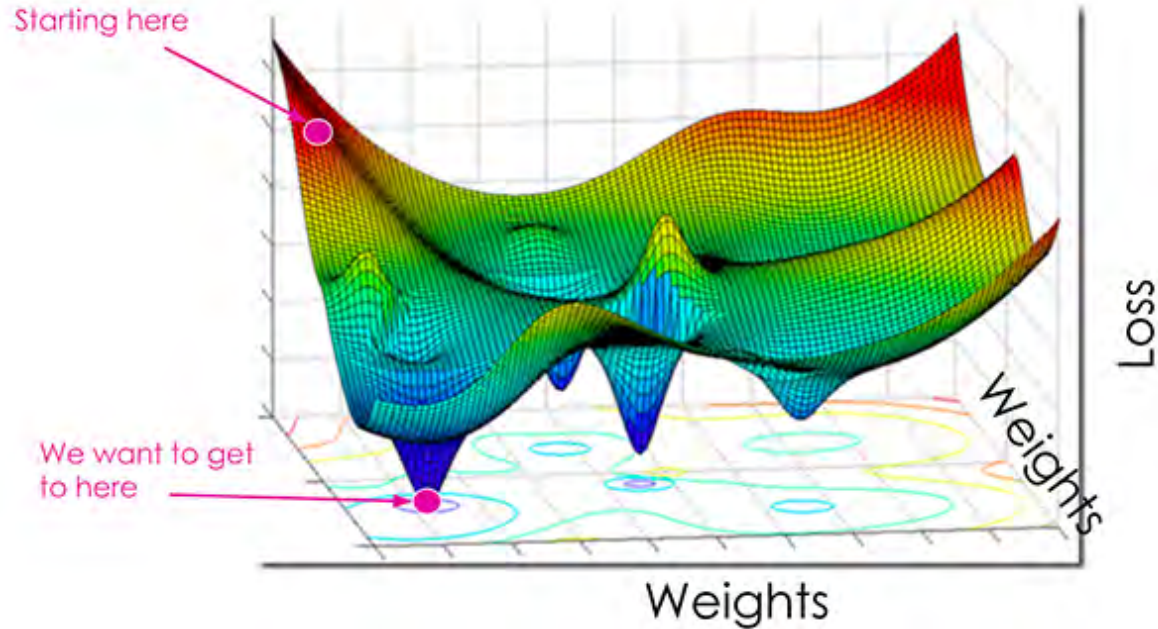
- Backpropagation in general:



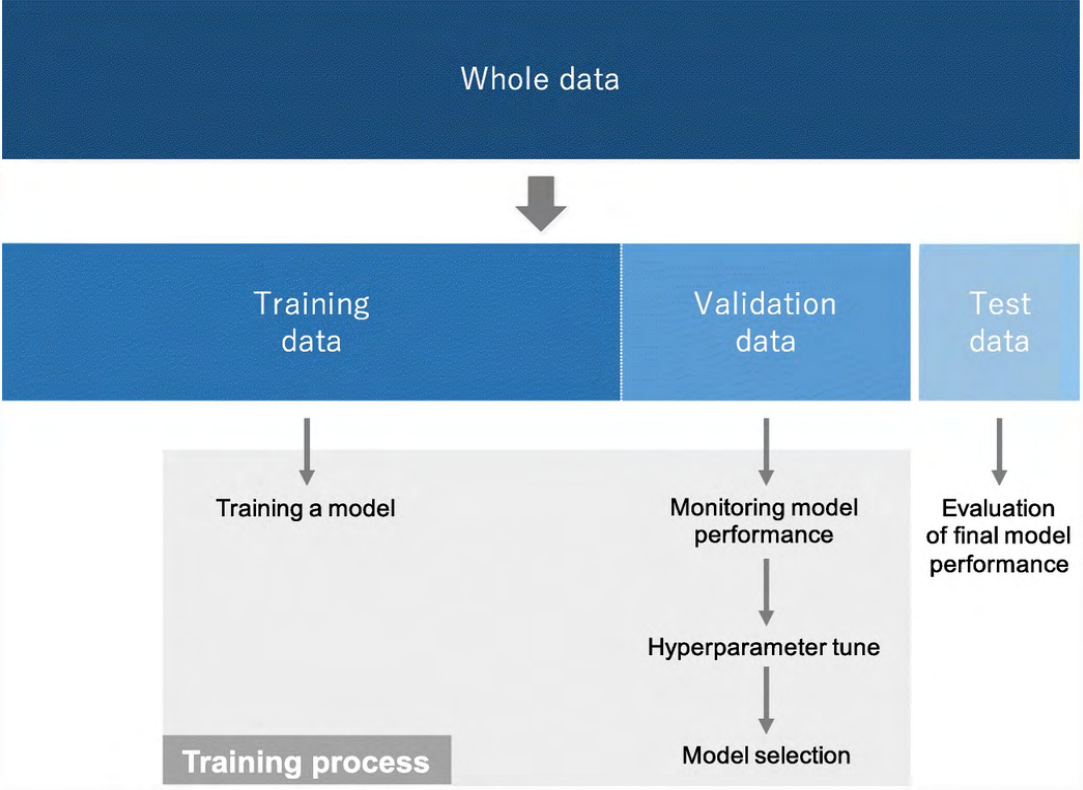
<https://www.superdatascience.com/blogs/artificial-neural-networks-backpropagation>

- 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?

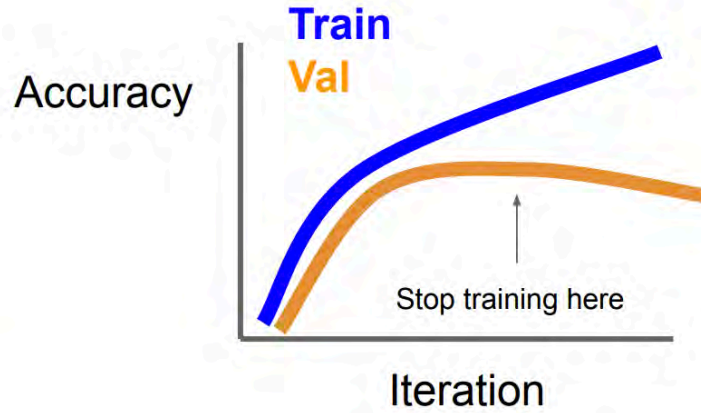
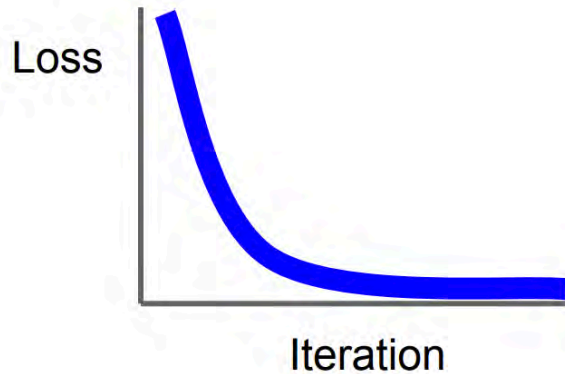


Training Methodology



Training vs. Testing Error

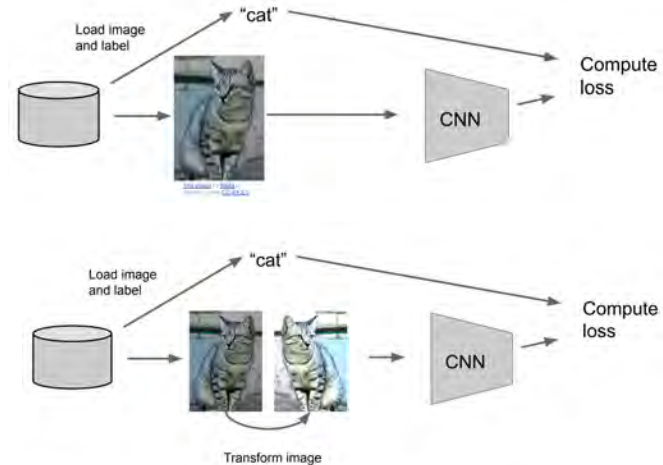
- Proper optimizer and training strategy can minimize the loss.
- Small training error is not always corresponding to a small testing/validation error.



Data Augmentation

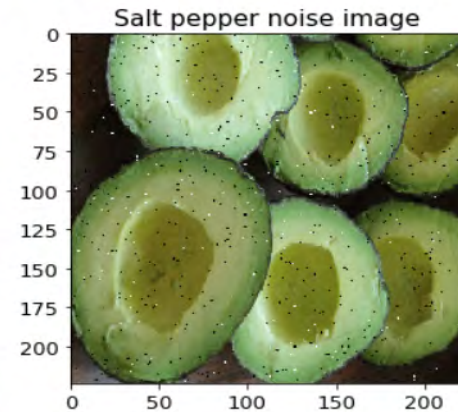
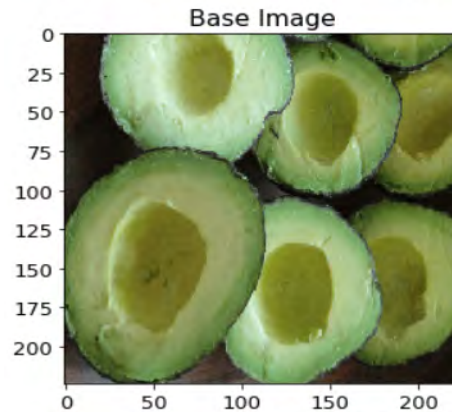
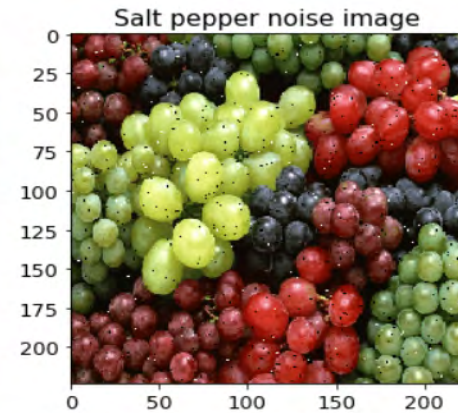
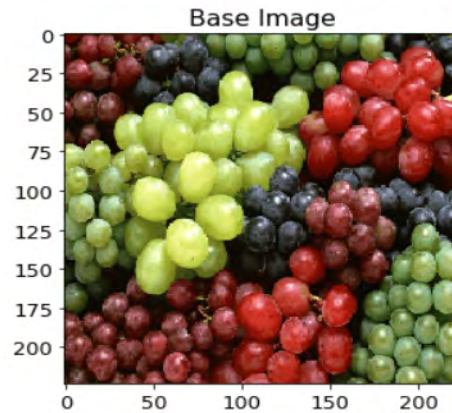
- Data augmentation generate different versions of a real dataset artificially to increase its size
- Improving the robustness of the networks
- We use data augmentation to handle data scarcity and insufficient data diversity
- Data augmentation helps to increase performance of deep neural networks

- Common augmentation techniques:
 - Adding noise
 - Cropping
 - Flipping
 - Rotation
 - Scaling
 - Translation
 - Brightness
 - Contrast
 - Saturation
 - Generative Adversarial Networks (GANs)



Data Augmentation

➤ Adding noise



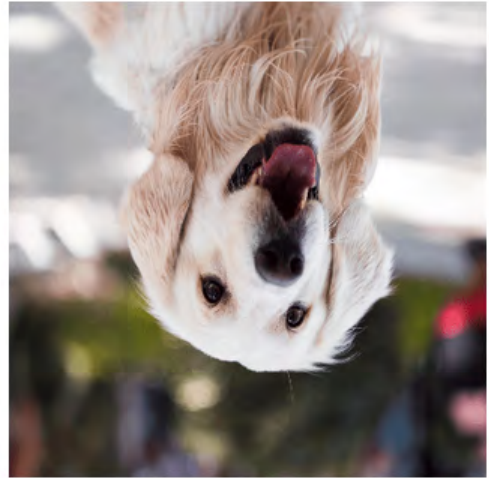
Data Augmentation

➤ Cropping



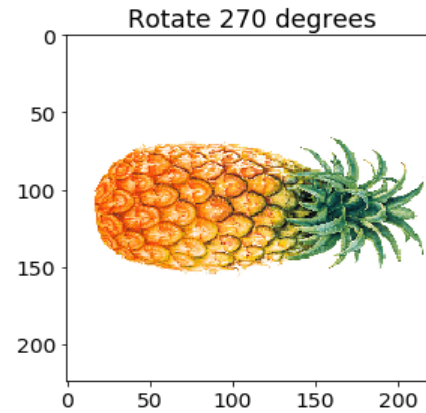
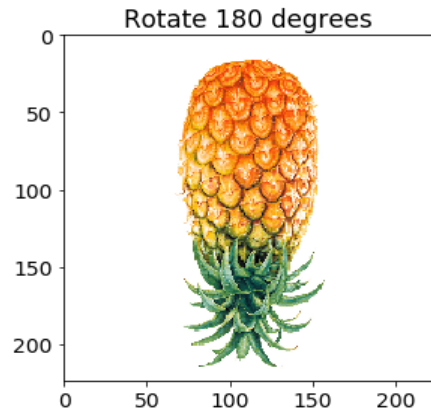
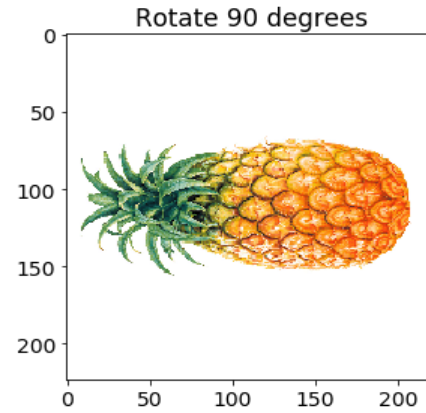
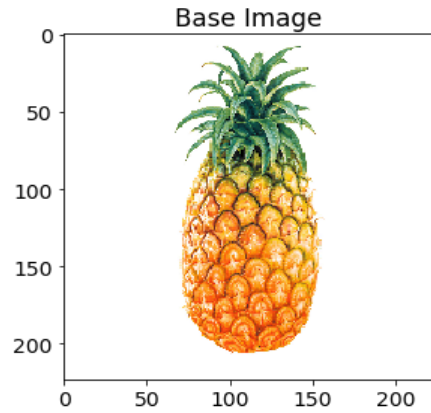
Data Augmentation

➤ Flipping



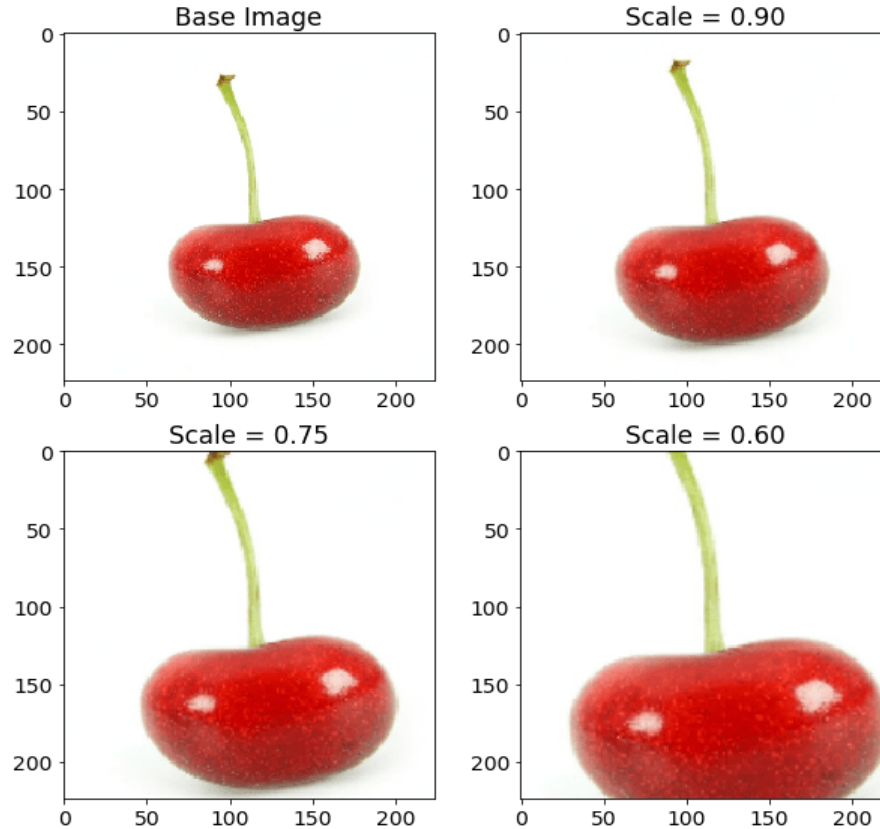
Data Augmentation

➤ Rotation



Data Augmentation

➤ Scaling



Data Augmentation

➤ Translation



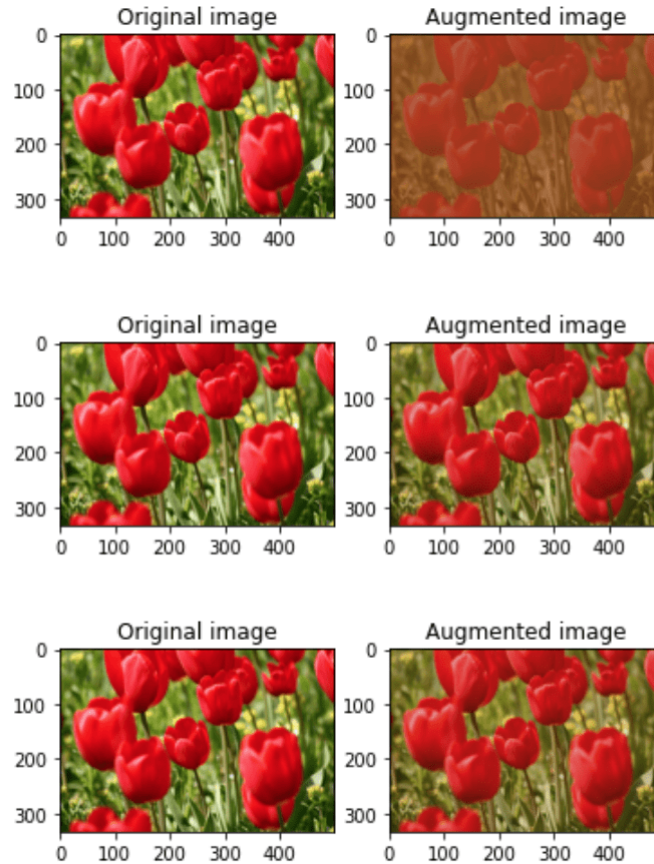
Data Augmentation

➤ Brightness



Data Augmentation

➤ Contrast



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

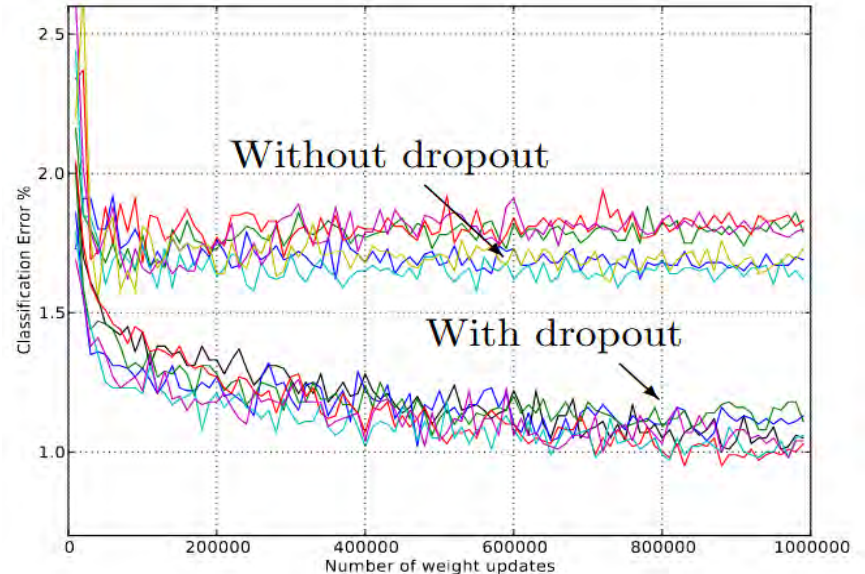
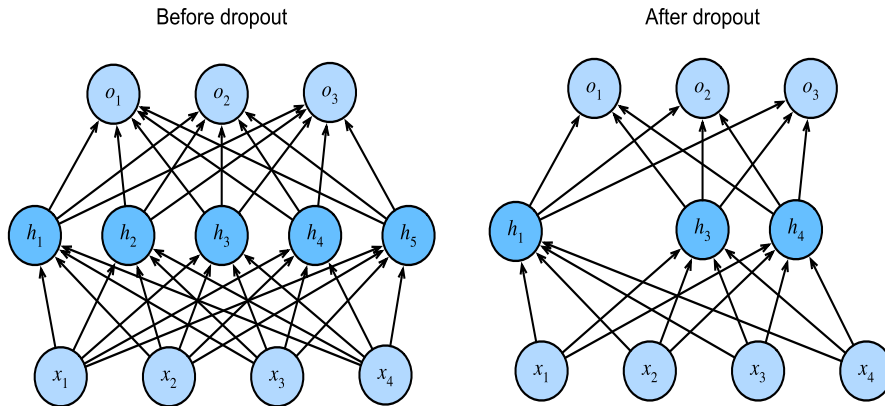
$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_{\theta}(x_i))^2 + \lambda \sum_{i=1}^n |\theta_i|$$

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

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

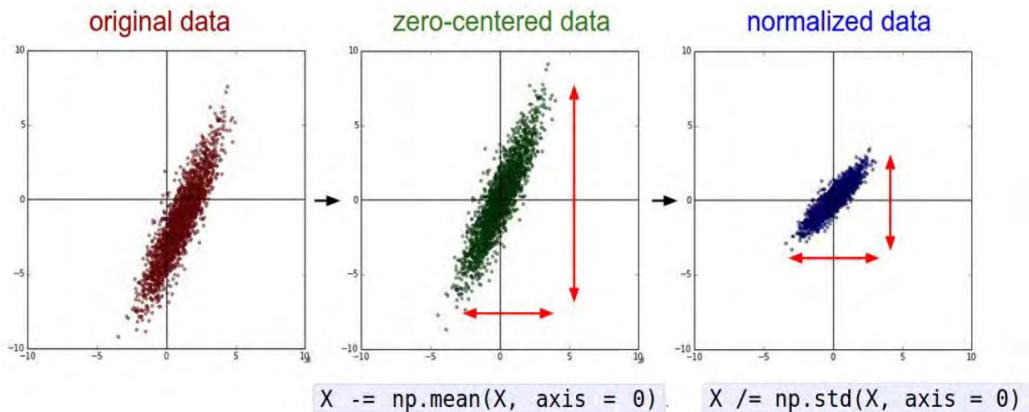
Regularization: Dropout

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



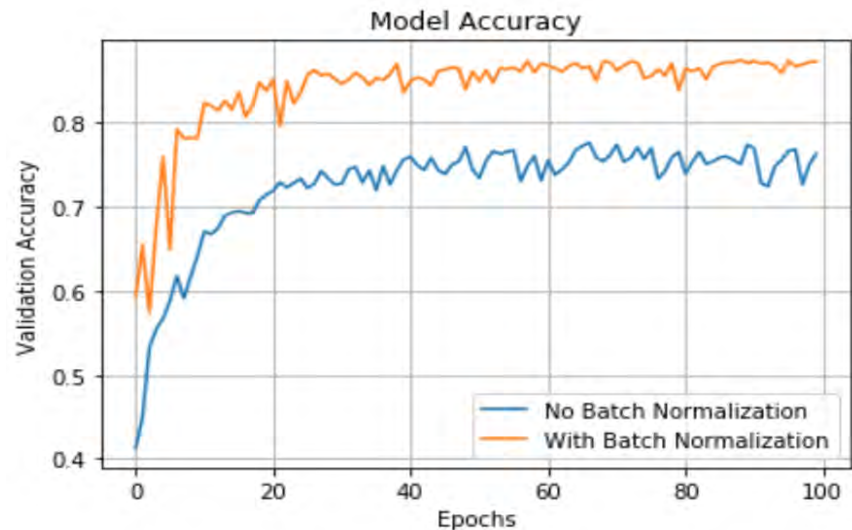
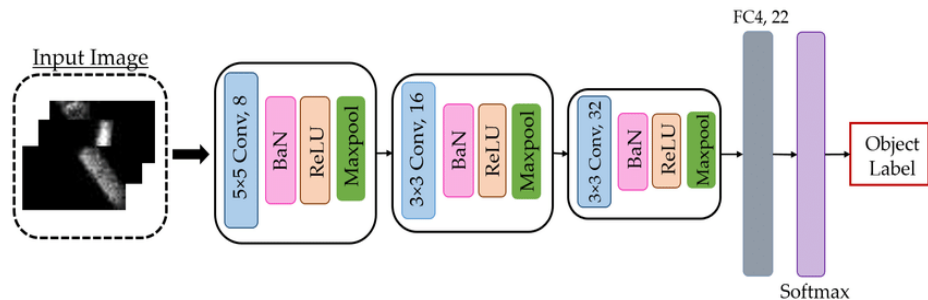
Data Preprocessing

- 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



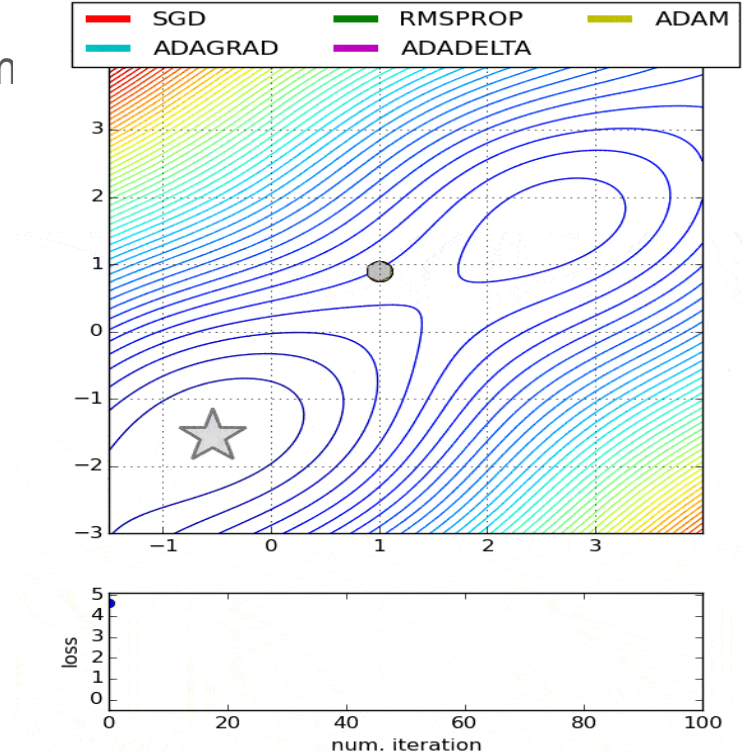
Batch Normalization

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



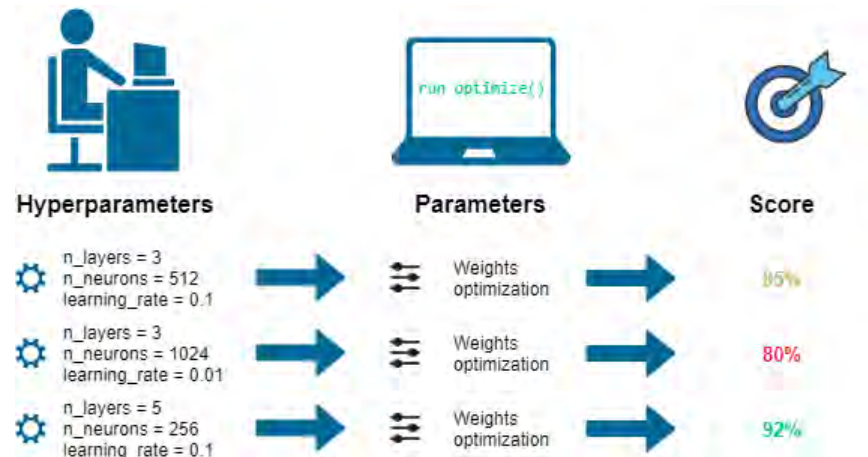
Choice of Optimizers

- 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



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
 - ...



Deep Learning Frameworks/Packages

Caffe

 Caffe2

 Chainer

 Microsoft
Cognitive
Toolkit


MATLAB



 PaddlePaddle

 PyTorch


TensorFlow

 torch

 Wolfram
Language™



 Keras

GPT-4

User What is unusual about this image?



Source: [Bamorama](#)

GPT-4 The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.

User Can you explain this meme?

Sometimes I just look at pictures of the earth from space and I marvel at how beautiful it all is.



GPT-4 This meme is a joke that combines two unrelated things: pictures of the earth from space and chicken nuggets. The text of the meme suggests that the image below is a beautiful picture of the earth from space. However, the image is actually of chicken nuggets arranged to vaguely resemble a map of the world. The humor in this meme comes from the unexpected juxtaposition of the text and the image. The text sets up an expectation of a majestic image of the earth, but the image is actually something mundane and silly.

Segment Anything Model (SAM)

