Neural Networks II

Lecture 12

Manuel Balsera

IEOR, Columbia University

April 26, 2019

Outline

Introduction

Today, we describe **Convolutional Neural Networks** an special case of the dense neural networks we discussed in last lesson.

- Convolutional Neural networks are **less general** that dense neural networks.
- They incorporate strong **prior assumptions** about the locality and translation invariance of features.
- Compared to a dense network of the same size, a CNN has a much smaller number of **trainable parameters**.
- When these assumptions are appropriate they generalize better than dense networks.
- CNNs are the **state of the art** for image classification and many other problems.
- We will see examples from Image Classification, Speech Recognition and Text Classification.
- think of this class as an *appetizer* for a deep learning course.

References

- Chapter 9 of Goodfellow, et al, Deep Learning [1].
- Stanford's CS231N module on neural networks for image classification.
- For CNN use in text classification see [2].
- For CNN use in speech recognition see [3].

The Learning Problem

Orientation

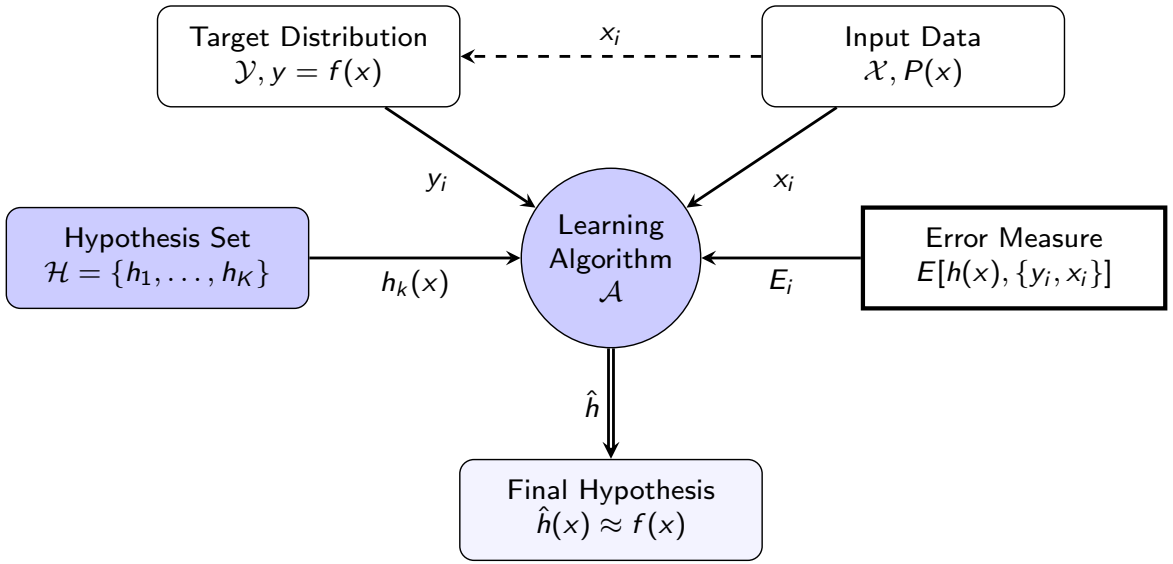
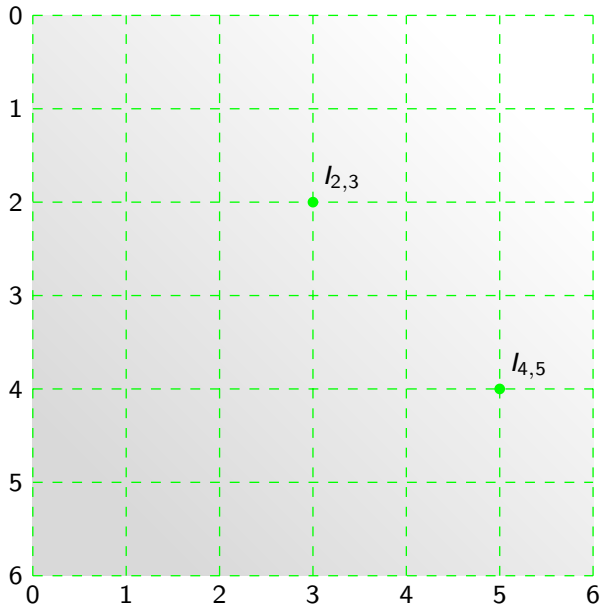


Figure 1: Training Process

- Convolutional Neural Networks **restrict** the hypothesis space of dense neural networks.
- Specialized software and hardware exist to accelerate forward and backward propagation.
- Training is still achieved by SGD and related methods.

How to represent a 2D Gray Scale Image

Review



- Gray Scale image $R \times C{=}7 \times 7$ (stored **by row**).
- $x_{d=17} = x_{2C+3} = I_{2,3}$, $x_{d'=33} = x_{4C+5} = I_{4,5}$
- vector representation x_d treats each pixel independently.
- No concept of distance between pixels $d \neq d'$.
- Color Image as an extra color index (the **color plane**) s , $x_{y3C+3x+s} = I_{y,x,s}$

Feature Enhancement for a Logistic Image Classifier

Review

We would like to define features as

- a function of a small **neighborhood** of a pixel position d.
- to capture position independence the same function should be applied to all pixels.
- to increase the hypothesis space, we also need a non-linearity.

The solution is define a **Matrix Convolution**

Matrix convolution

$$(K * I)_{y,x} = \sum_{s_y} \sum_{s_x} K_{s_y,s_x} I_{y-s_y,x-s_x} \tag{1}$$

where the **Kernel Matrix** K has the property that $K_{s_u,s_x} = 0$ if $|s_y| > \Delta$ or $|s_x| > \Delta$.

- Δ is called the **size** of the kernel.
- define features

$$F_{y,x} = \sigma((K * I)_{y,x}) \tag{2}$$

- $\sigma(z)$ is some non-linear function.
- F has dimension $\approx D$, the number of pixels.

Example: Edge Detection

Review

An edge in an image is a region where $I_{x,y}$ changes abruptly (large gradient).
We can estimate the image gradient using the Sobel Kernel

Sobel Kernel

$$S_v = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad S_h = \begin{bmatrix} +1 & 2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

Symmetric approximation to $\frac{\partial}{\partial x} I(y,x)$ and $\frac{\partial}{y} I(y,x)$.

- Gradient in direction θ can be approximated by

$$G^{\theta}_{y,x} = \cos(\theta)(S_v * I)_{y,x} + \sin(\theta)(S_h * I)_{y,x} \tag{4}$$

  To introduce a non-linearity we can **threshold** at zero

$$F^{\theta}_{y,x} = \max(0, G^{\theta}_{y,x}) \tag{5}$$

- To reduce size of feature matrix can then **poll** (aggregate) the F matrix over a small local region by averaging or taking the max over region.

$$P^{\theta}_{y,x} = \max_{y+\pm\Delta, x+\pm\Delta} (F^{\theta}_{y,x}) \quad \text{or} \quad \sum_{y+\pm\Delta, x+\pm\Delta} (F^{\theta}_{y,x}) \tag{6}$$

Edge Detection Kernel

Review

There are many possible choices of Kernel just for the **gradient** computation.

Prewitt

$$S_{\text{Prewitt}} = \begin{bmatrix} +1 & 1 & 1 \\ +0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{7}$$

Sobel

$$S_{\text{Sobel}} = \begin{bmatrix} +1 & 2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{8}$$

Scharr

$$S_{\text{Scharr}} = \begin{bmatrix} +3 & 10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \tag{9}$$

- Kernel must be chosen by **cross-validation** or simply selected *a priori*.
- See `skimage.filters` for kernels specialized for many other tasks.

Example: Sobel Gradient Convolution

Review

Original Image

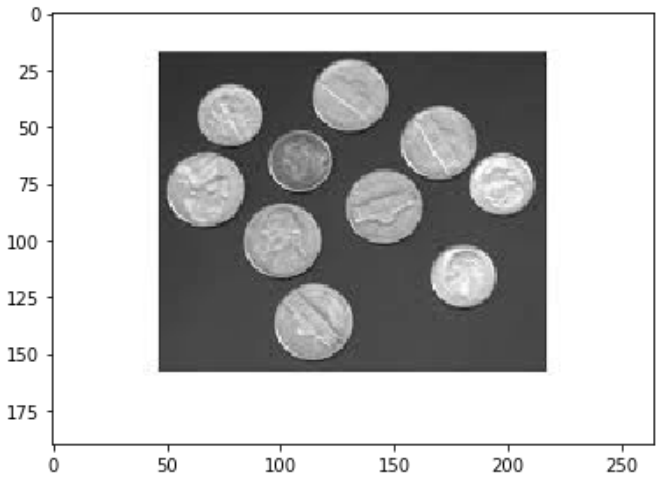
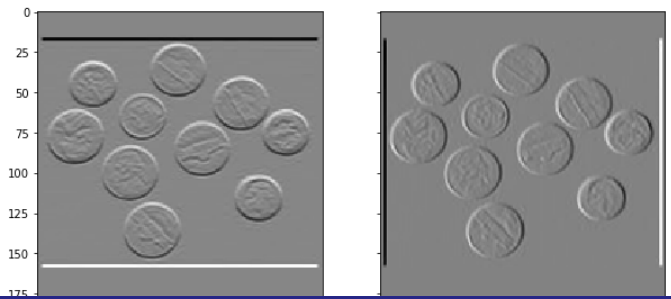


Figure 2: Gray Scale Image

Convolutions with Sobel's horizontal and vertical kernels



Example: Directional Gradient and Thresholding

Review

We can then compute directional derivatives

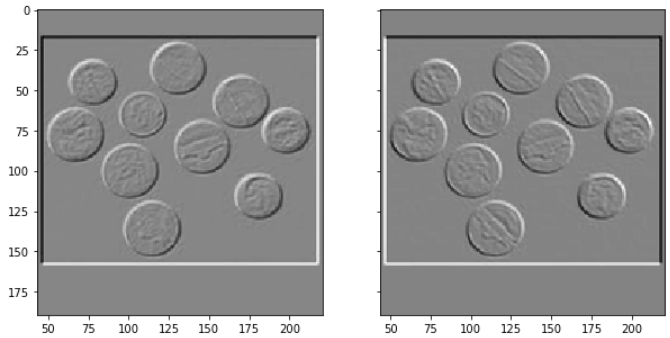


Figure 4: 45 deg and -45 deg directional derivatives

and, finally, apply a threshold

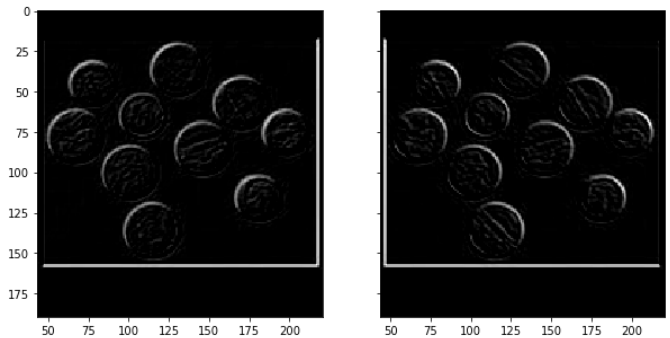
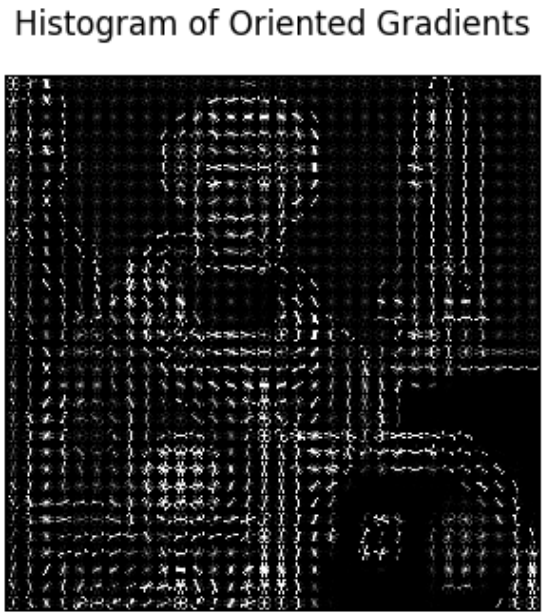


Figure 5: Thresholded directional derivatives

Histogram of Oriented Gradients

Review

- A more sophisticated, state of the art, hand crafted set of features is the Histogram of Oriented Gradients (HOG)[4].
- They are a variation on the directional gradient features we have described with careful specification of non-linearity and polling.
- Implemented in `skimage`
- See `skimage.features` module for many other pre-defined feature definitions.
- What feature to use is **problem** specific.



Higher Order Features

- Some times we want **higher order features** as combinations of local features, i.e:
  Divergence of Gradients (Laplacian) How fast gradients are growing around a point.
  Curl Rate of rotation of gradient around a point.
- We can design explicit higher order features features as **convolutions** of the first order features.
- See illustrations from [5] on breast tissue classification.

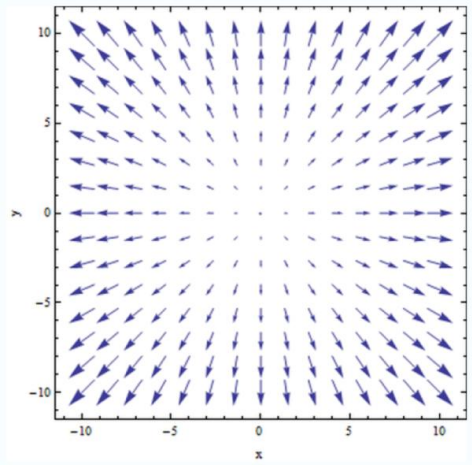


Figure 7: Gradient Divergence ($\nabla \cdot \nabla$) as a local property of vector field.

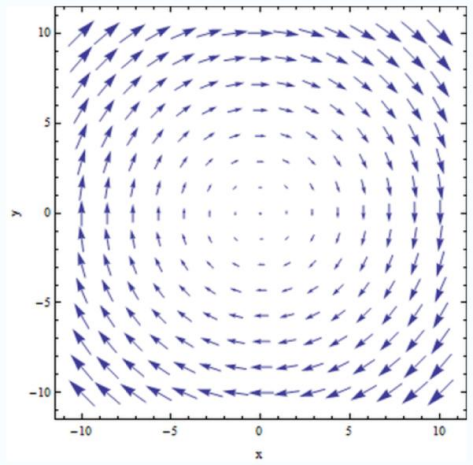


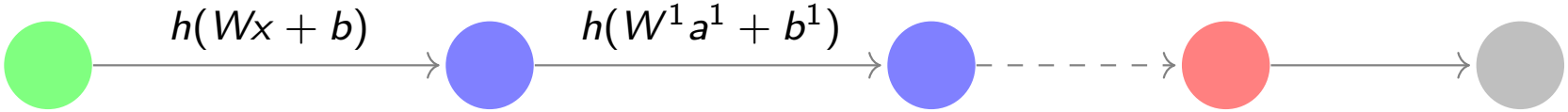
Figure 8: Curl of Gradient ($\nabla \times \nabla$) as a local property of vector field.

Feed Forward Function Valuation

Review

Focusing in just one path on the network, information flows **forward** from x to L

$$x \qquad a^1 \qquad a^2 \qquad \hat{y}(\eta(x)) \qquad L[y, \hat{y}(x)]$$



- We have defined a **direct, acyclic graph** that defines the calculation of $\eta(x)$.
- The lost L is a extra node at the end of the graph compares η to y.
- Each layer has associated a set of parameters W and b.

Feedforward Neural Network Diagram

Review

Graphically a feedforward neural network with two hidden layers can be represented as

x a^1 a^2 $\hat{y}(\eta(x))$ $L(y, \hat{y}(x))$

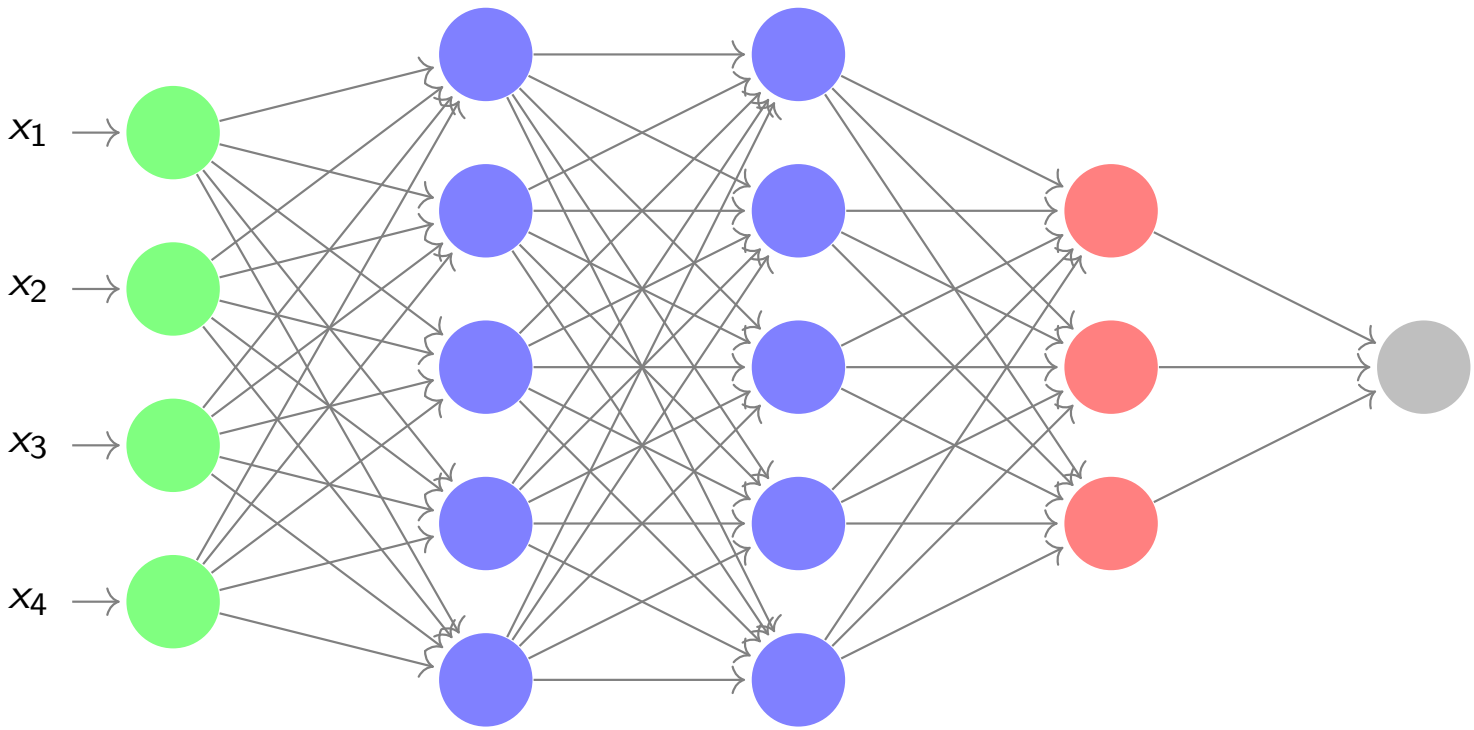


Figure 9: Dense Neural Network Diagram

Convolutional Neural Network

We can reduce significantly the number of learnable parameters if the loadings of each node are

Localized so each node only talks to its *neighbors*.

Shared All nodes in a given layer share the same loadings.

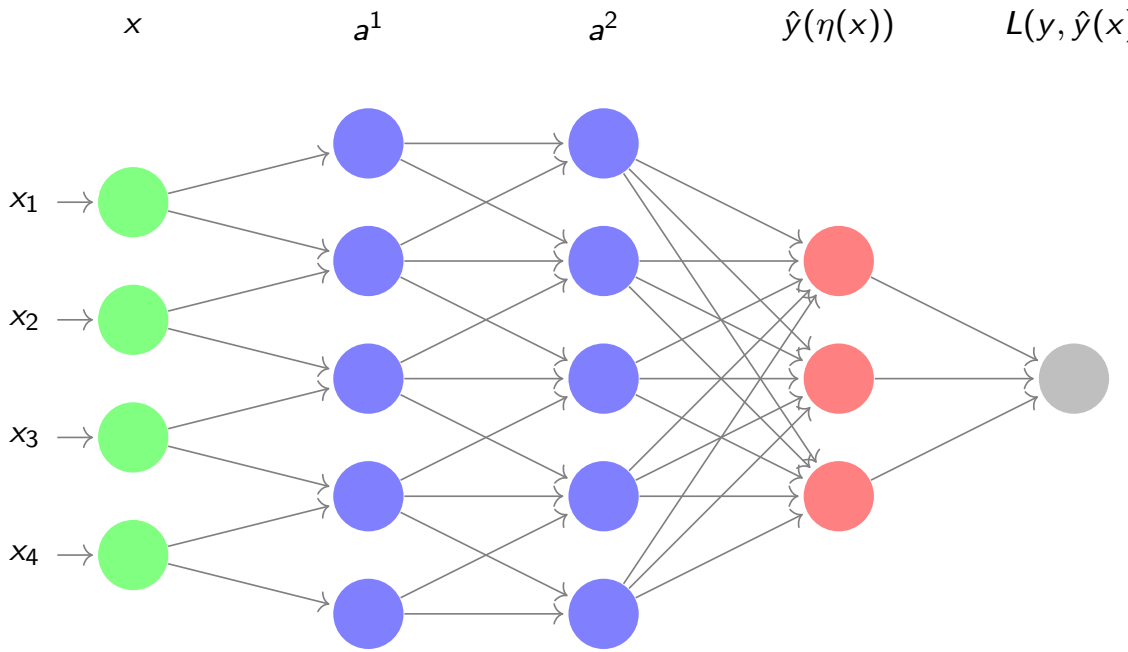


Figure 10: Dense Neural Network Diagram

Convolutional Layer II

A convolution layer is defined by

input $I_{r,c,i}$, a $H \times W \times C$ image *volume* where

- H is the height (in pixels) of the image.
- W is the width (in pixels) of the image.
- C is the number of input *channels* or **features**.

stride a pair of positive numbers s_W, s_H.

output $O_{r,c,o}$, a $H' \times W' \times F$ image *volume* where[1]

1. $H' = \frac{H}{s_H}$ is the height (in pixels) of the image.
2. $W' = \frac{W}{s_W}$ is the width (in pixels) of the image.
3. F is the number of output *channels* or **features**.

kernel $K_{r,c,i,o}$, a $K_H \times K_W \times C \times F$ tensor

bias b_o, a F vector

[1] We assume *SAME* padding for simplicity, otherwise formula is

$$H' = \frac{H - K_H + 2P_H}{s_H} + 1 \tag{10}$$

where P_H is the amount of padding.

Convolutional Layer III

The **image convolution** is defined by

$$(K * I)_{r,c,o} = \sum_{r',c',i} I_{rs_H+r',cs_W+c',i} K_{r',k',i,o} \tag{11}$$

a **convolutional layer** is then defined by the equation

$$a^{\alpha} = h(K * a^{\alpha-1} + b) \tag{12}$$

which is a special case of the dense neural layer where the dense multiplication Wa is replaced by the kernel convolution $K * a$.

x a^1 a^2 $\hat{y}(\eta(x))$ $L[y, \hat{y}(x)]$

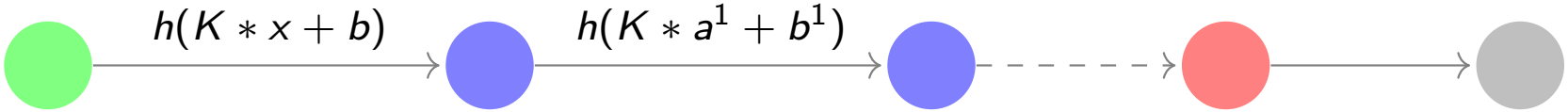


Figure 11: Two convolutional layers in a neural network.

Convolutional Layer IV

The convolutional layer, thus, connects two rank 3 tensors as follows

$$H \times W \times C \to \frac{H}{s_H} \times \frac{W}{s_W} \times F \tag{13}$$

which can be depicted (for C=1) graphically as

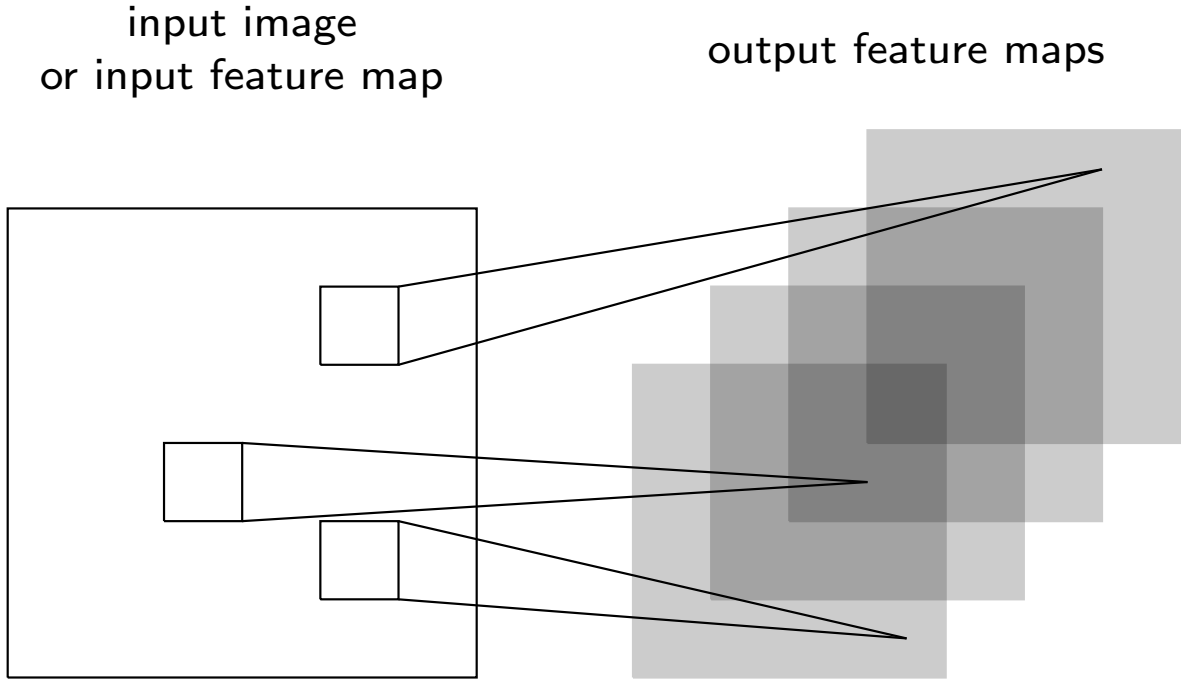


Figure 12: Illustration of a single convolutional layer. If layer l is a convolutional layer, the input image (if $l = 1$) or a feature map of the previous layer is convolved by different filters to yield the output feature maps of layer l.

Complexity of a Convolutional Layer

The order of magnitude of the memory, computation, and learnable parameters is given by

Learnable Parameters $K_H \times K_W \times C \times F + F$

Computation $(K_H \times K_W \times C) \times \left(\frac{H}{s_H} \times \frac{W}{s_W} \times F\right)$

Memory $\frac{H}{s_H} \times \frac{W}{s_W} \times F$

- by keeping the kernel size $(K_H \times K_W)$ small we can significanly reduce the number of learnable parameters compared to an equivalent dense layer

$$(H \times W \times C) \times \left(\frac{H}{s_H} \times \frac{W}{s_W} \times F\right), \tag{14}$$

  where computional and memory requirements would be of the same order.
- Convolutional layers can involve quite a bit of computation while learning only a few parameters.
- **GPU**s and other specialized hardware can greatly accelerate the training of convolutional layers.
- Use of memory for intermediate results can be a limiting factor

Learning with Convolutional Layers

- We can think of a computation layer as creating a set of nonlinear features $\Phi(x;\theta)$, as in the case of a dense layer.
- As opposed to a dense layer, the number of parameters θ is small.
- Convolutional Layers make two key assumption about the features learned

  Local features involve local properties of image (like gradients, or curvature).

  Translational symmetry The same feature can be present anywhere in the image with the same meaning.
- Convolutional layers represent features similar to the HOG (Histogram of Gradients) we discused before.
- The kernel matrix representing the feature, however, is **learned** from the data, rather than imposed **a priory** and chosen by cross-validation.
- Review how Tiwari [6] and Tan [5] SVM image recognition pipelines selected features.
- In practice, if you have little data, features are **designed** by hand, and selected by cross-valiation. With enough data is better to learn features using a convolutional layer.

Pooling Layer

Another layer commonly found in convolutional networks is the **pooling layer**. It is defined by

input $I_{r,c,i}$, a $H \times W \times C$ image *volume* where

stride a pair of positive numbers s_W, s_H.

pool size a pair of positive numbers p_W, p_H.

output $O_{r,c,o}$, a $H' \times W' \times C$ image *volume*

1. $H' = \frac{H}{s_H}$ is the height (in pixels) of output.
2. $W' = \frac{W}{s_W}$ is the width (in pixels) of output.

The output is computed as

$$O_{r,c,o} = g(\{I_{s_H r+\Delta r, s_W c+\Delta c, o}\}) \tag{15}$$

where $\Delta r = 1, \cdots p_H$ and $\Delta c = 1, \cdots p_W$ and g is some **aggregating function**, usually like `sum` or, more frequently `max`.

Pooling Layer II

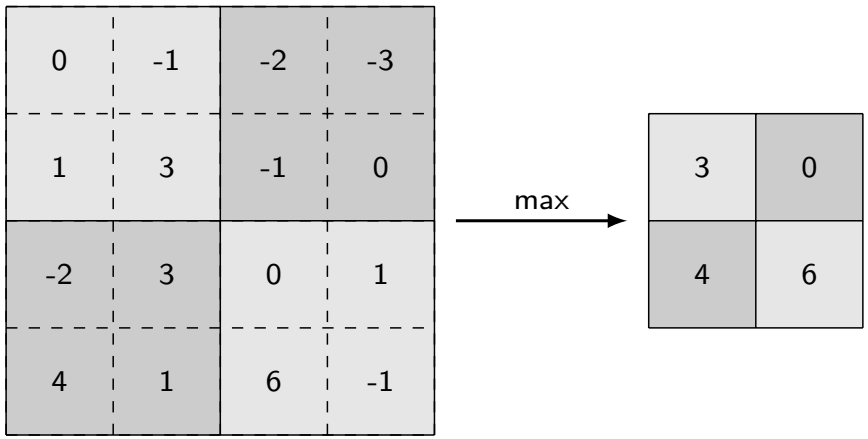


Figure 13: Max Pooling operation with (2,2) pool size and (2,2) stride.

- pooling operations provide for *local* translation invariance within a pool block.
- Pooling operations reduce the feature volume by a factor $s_H \times s_W$.
- Pooling layers have **no trainable parameters**.

Pooling III

The pooling operation reduces the feature volume

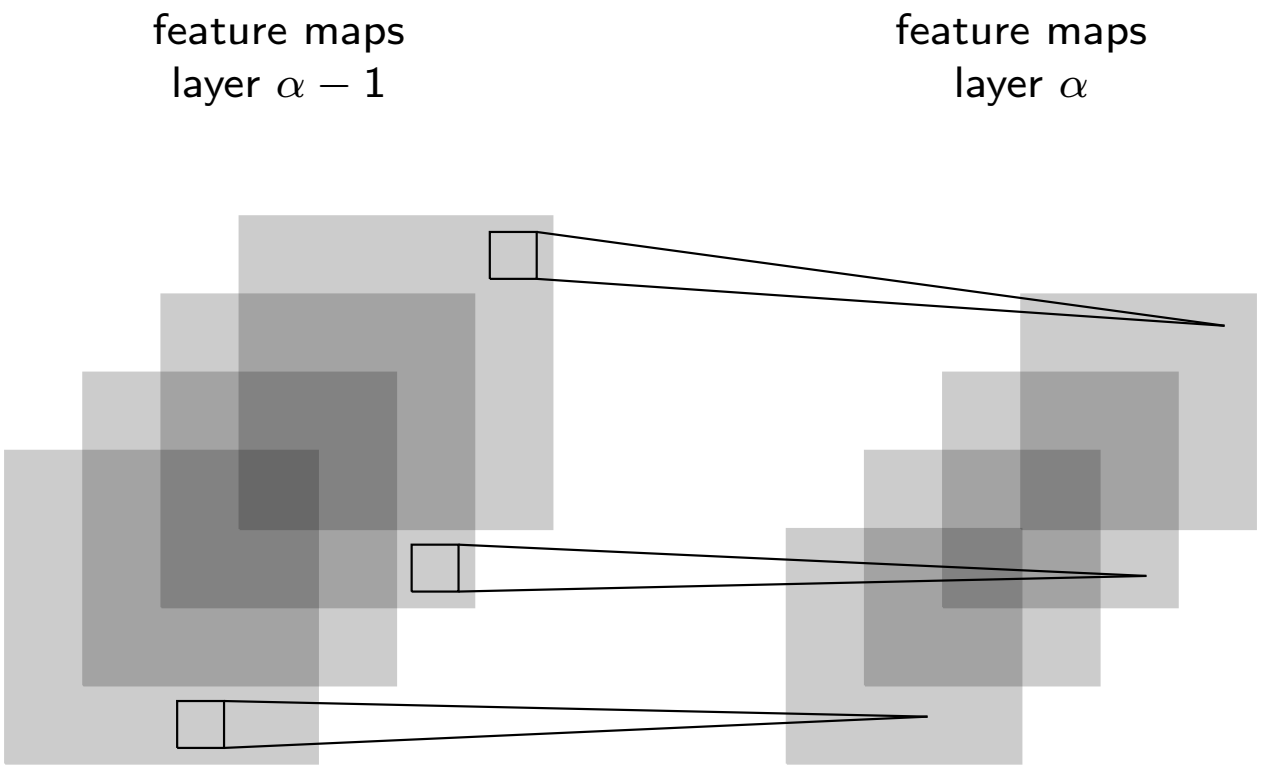


Figure 14: Illustration of the impact of a pooling layer on the feature volume.

Convolutional Network Architecture

A Typical convolutional network architecture will have

- Alternating convolutional and pooling layers
- optional dropout layers (implementing node dropout)
- A few dense layers (usually one or two)
- Finaly one linear layer to define η with activation $\hat{y}(\eta)$.
- the log likelihood loss.

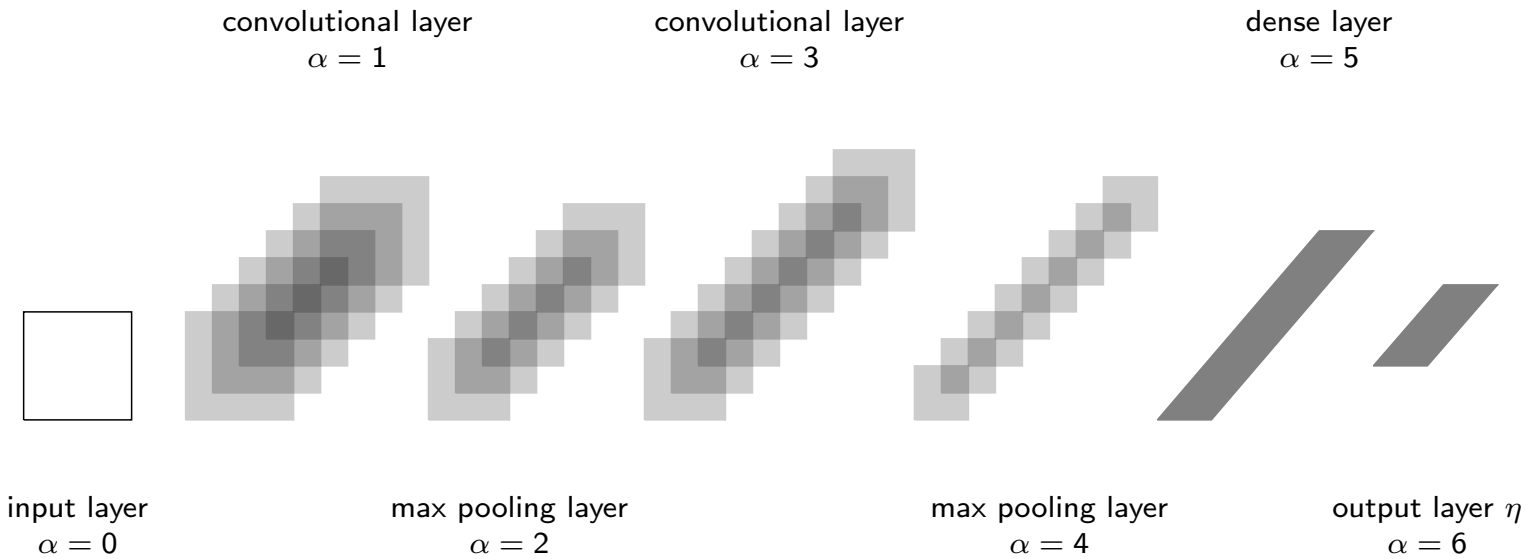


Figure 15: The architecture of the original convolutional neural network, as introduced LeCun et al 1989

Geometrical Shapes Gradients

Review

With the Sobel kernel's

$$S_1 = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad S_2 = \begin{bmatrix} +1 & 2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{16}$$

we define **directional gradients**

$$F^{\theta}_{x,y} = \max\left(0, \left(\cos(\theta)S_1 + \sin(\theta)S_2\right) * I\right)_{x,y} = \max(0, K^{\theta} * I)_{x,y} \tag{17}$$

$$P^{\theta} = \sum_{x,y} F^{\theta}_{x,y} \tag{18}$$

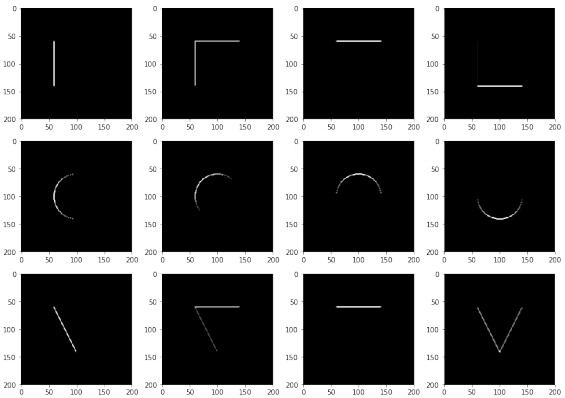


Figure 16: Directional Gradients $F^{\theta}_{x,y}$ for Simple Geometrical Shapes

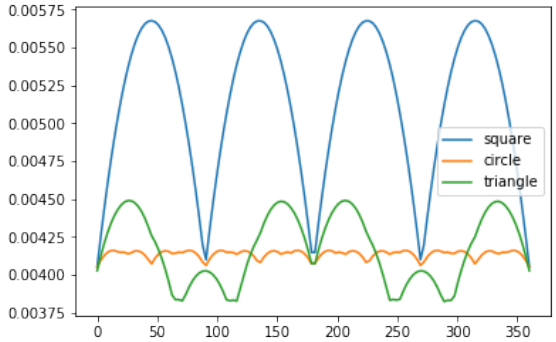


Figure 17: Polled Mean Gradients P^{θ} as a function of θ

A Convolutional Layer

In a convolutional layer we replace the directional gradients

$$F^{\theta}_{x,y} = \max(0, K^{\theta} * I)_{x,y} \tag{19}$$

where K^{θ} is the 3×3 matrix

$$K^{\theta} = \cos(\theta)S_1 + \sin(\theta)S_2 = \begin{bmatrix} \cos(\theta) + sin(\theta) & 2\sin(\theta) & -\cos(\theta) + \sin(\theta) \\ 2\cos(\theta) & 0 & -2\cos(\theta) \\ \cos(\theta) - \sin(\theta) & -2\sin(\theta) & -cos(\theta) - \sin(\theta) \end{bmatrix} \tag{20}$$

With a convolutional layer

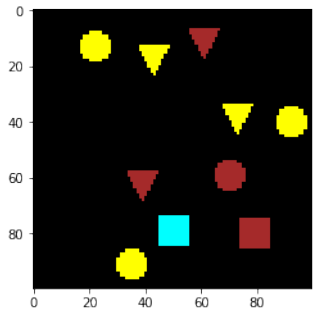
$$F^{c}_{x,y} = \max(W^{c} * I) + b^{c}, 0)_{x,y} \tag{21}$$

- where W^c is has $C \times 3 \times 3$ **learnable** parameter, and b^c is has c parameters.
- We are letting the data tells us what **local properties** to extract from the image.
- Polling layer is unmodified.
- Backpropagation allow us to train the convolutional layers

A Convolutional Network Example

We have the problem of **counting simple geometrical** shapes

- One convolutional (3×3) layer with 32 channels.
- Average (sum) polling layer gives positional independence.
- We use 9 output nodes (3 shapes $\times$ 3 colors) with a Poisson loss



Layer	Filter Size	Features	Parameters	Nodes
Input		3	0	$100^2 \times 3 = 30,000$
Conv	3×3	32	$\approx 3 \times 3 \times 3 \times 32 = 864$	$100^2 \times 32 = 320,000$
Pool	100×100	32	0	32
Dense		64	$\approx 32 \times 64 = 2064$	64
Output		9	$\approx 64 \times 9 = 576$	9

Figure 18: Comparison of the performance of different methods in the MNIST data set raw features

Trained on $10,000$ images achieves **perfect** results in 1,000 epochs.

A Convolutional Network Example II

We now try to count shapes with randomly rotated axis

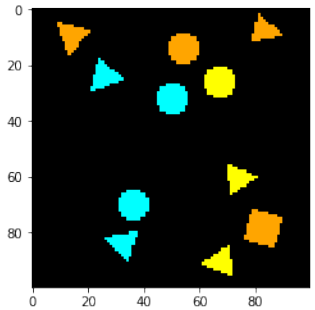


Figure 19: Rotated Shapes

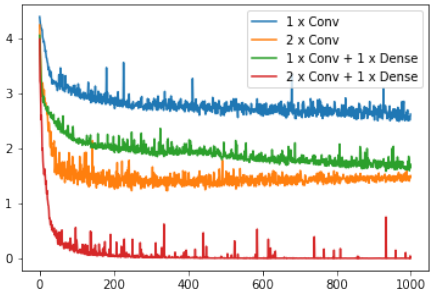


Figure 20: Training plot

- Gradients in fixed directions are no longer predictive features.
- 1 convolutional layer plus one dense layer has high error rate (misses count by 2 shapes in average).
- We need **relationships** between gradients.
- **2 layer** convolutional layers + 1 dense layer counts shapes perfectly.
- 12,000 trainable parameters.

ImageNet

Review from First Lecture

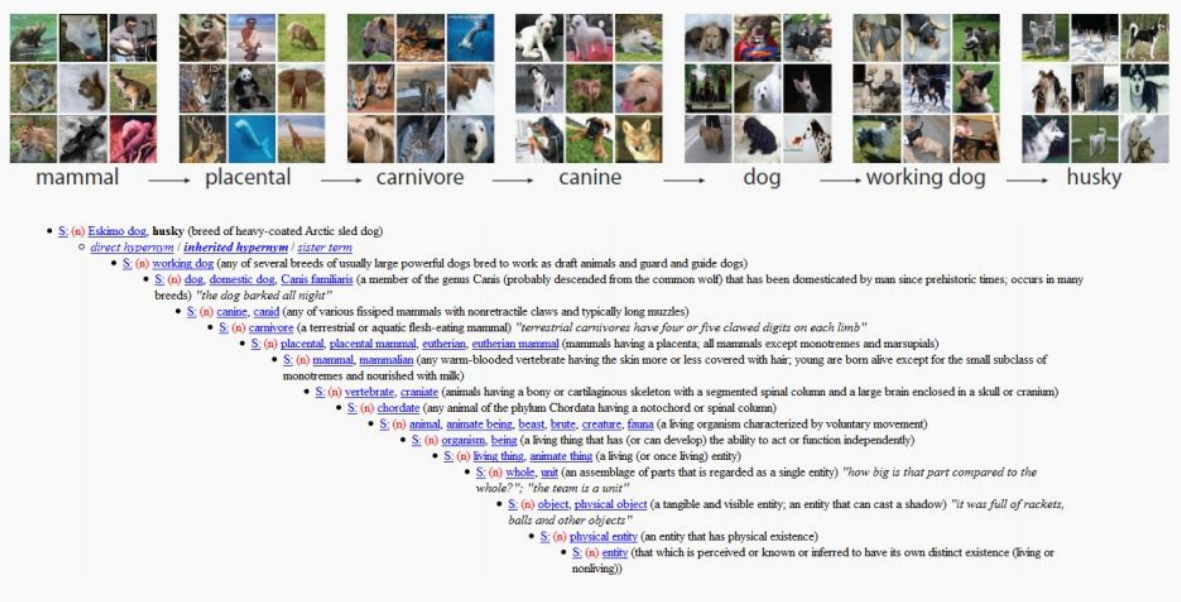


Figure 21: Source: David Yanovsky—Quartz

- Large data sets have pushed forward the field of image recognition.
- Fei-Fei Li's team's Image Net [7] has been instrumental.
- **Big Data**: $\approx$ 1000 classes, tens of millions of images.
- See David Yanovsky Blog post for background history.

Deep Learning Revolution

Review from First Lecture

- Performance on Imagenet Classification Problem has increased $10\times$ in seven years.
- Many groups can **consistently** get under 5% error rate.
- Human Level performance is now **rutine**.
- AlexNet (2012): Convolutional Neural Network with 8 Layers.
- Microsoft RestNet (2016) has ≈ 150 layers (depends how you count)
- This is **Deep Learning**.

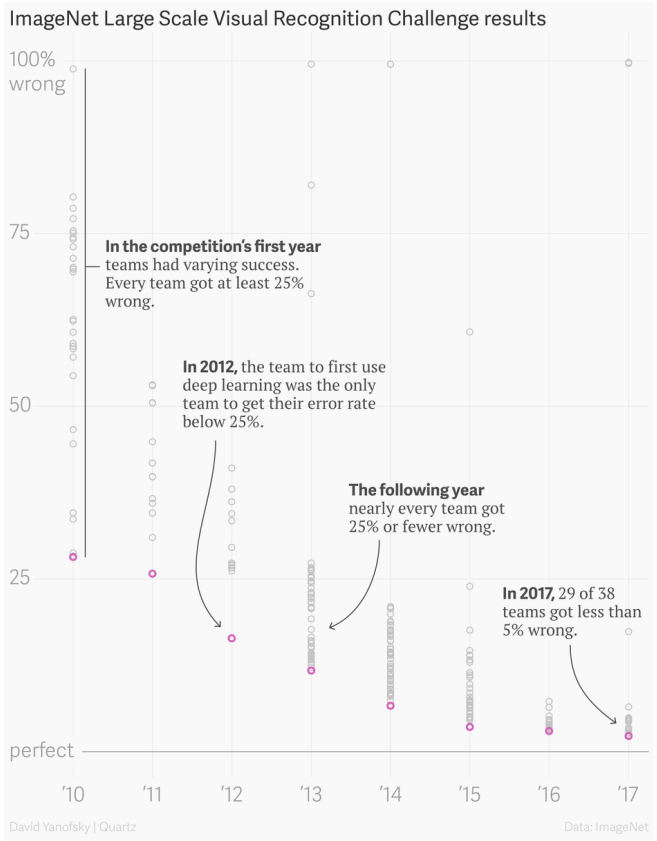


Figure 22: Source: David Yanovsky—Quartz

AlexNet 2012

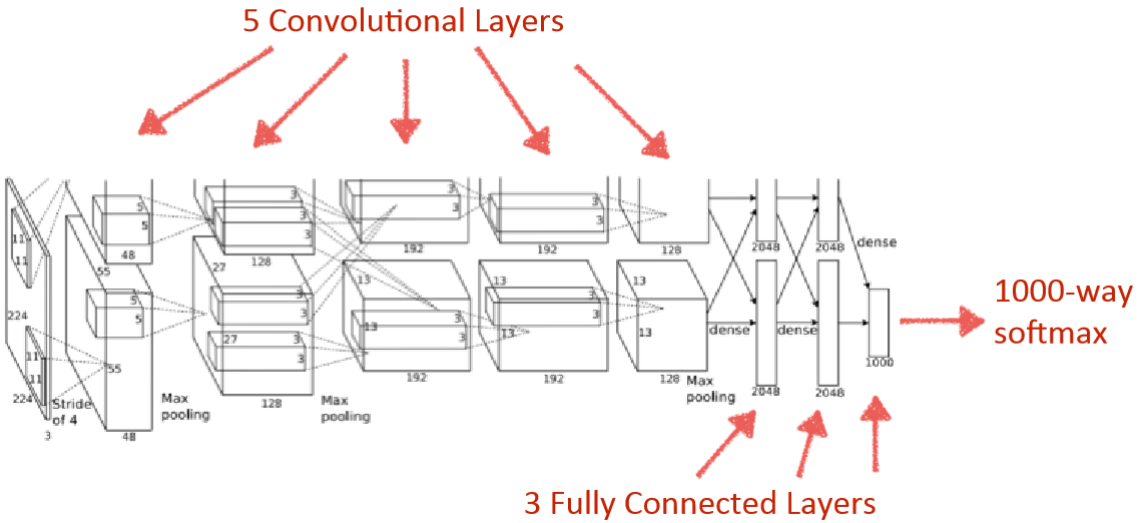


Figure 23: AlexNet 2012 Network architecture.

- Winner of 2012 ImageNet Competition by a wide margin.
- Traditional architecture, convolutions, pooling and dense layers.
- Deeper (8 layers).
- Innovations: Relu activation instead of sigmoid, use of dropout.
- Training used **data augmentation**.
- Accelerated training using GPUs.

Modern architectures have been growing deeper and more complex

Network	Year	Layers
AlexNet	2012	8
VGG	2014	19
GoogleNet	2014	22
ResNet	2015	152

Figure 24: Evolution in the number of layers of convolutional neural networks

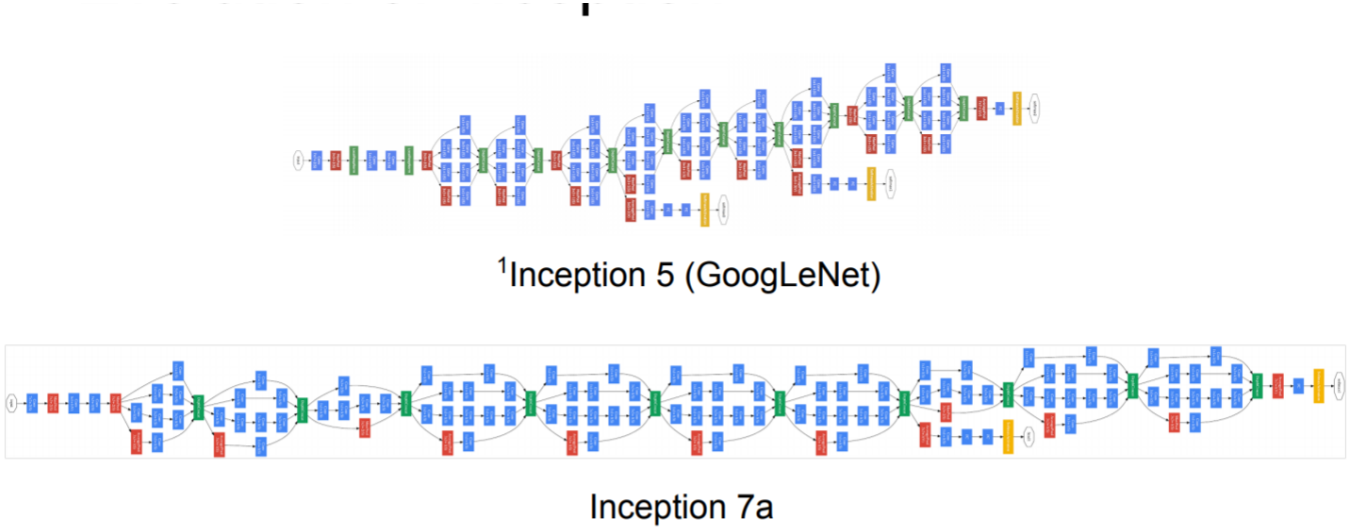


Figure 25: Inception (Google's Image Classifier) v5 and v7: The size of complexity of layers has been growing.

Current architectures are constantly refining the

- kinds of layers
- How to connect the layers

This is still a very active area of research.

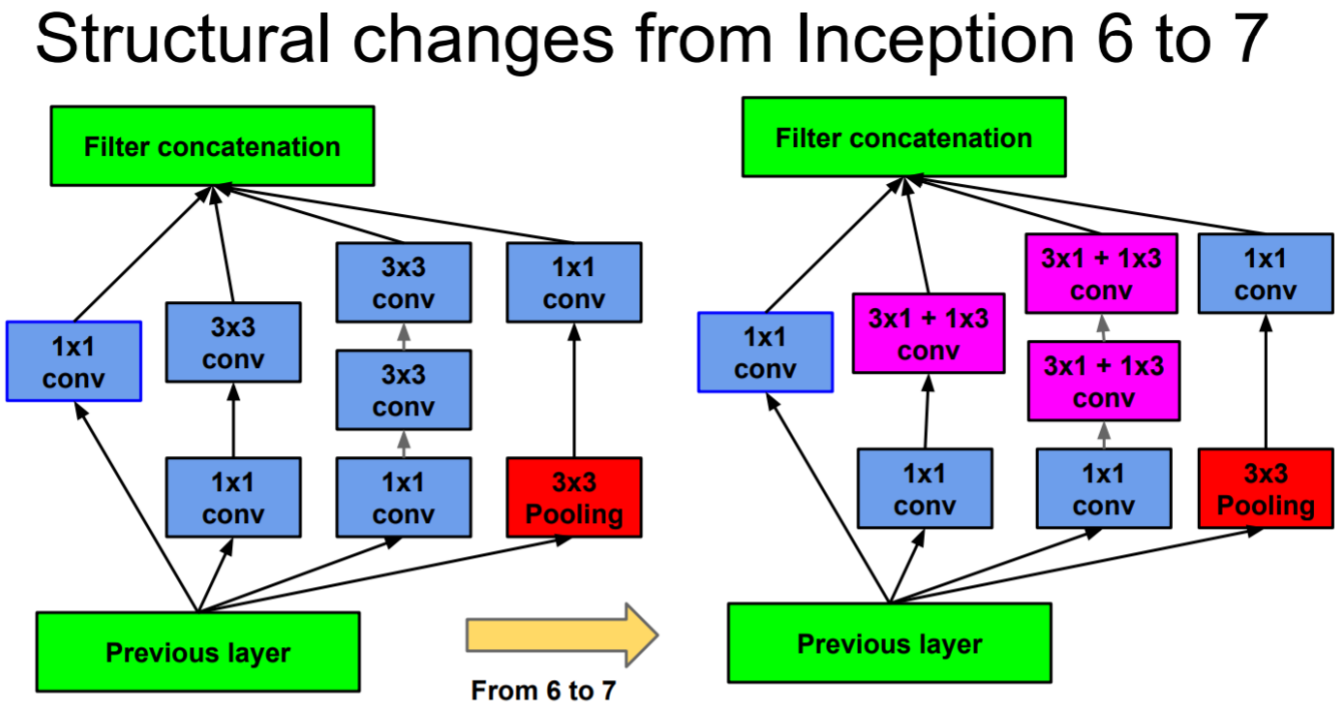


Figure 26: Inception (Google's Image Classifier) v6 and v7: Example on the evolution of one of Inception's modules form v6 to v7

MNIST Digit Classification

On **raw pixels** the following convolutional network ($\approx$ 55k learnable parameters) has a **99.2% accuracy** on MNIST digits and **92.6%** on MNIST fashion.

Layer	Filter Size	Strides	Features	Dropout	Nodes	Parameters
Input			1	0%	$28 \times 28 = 758$	0
Conv	5×5	(1,1)	10	0%	7,088	260
Pool	2×2	(2,2)	10	0%	1,960	0
Conv	5×5	(1,1)	20	0%	3,920	5,020
Pool	2×2	(2,2)		50%	980	0
Dense	50		1	0%	50	49,050
Output	10		1	0%	10	510

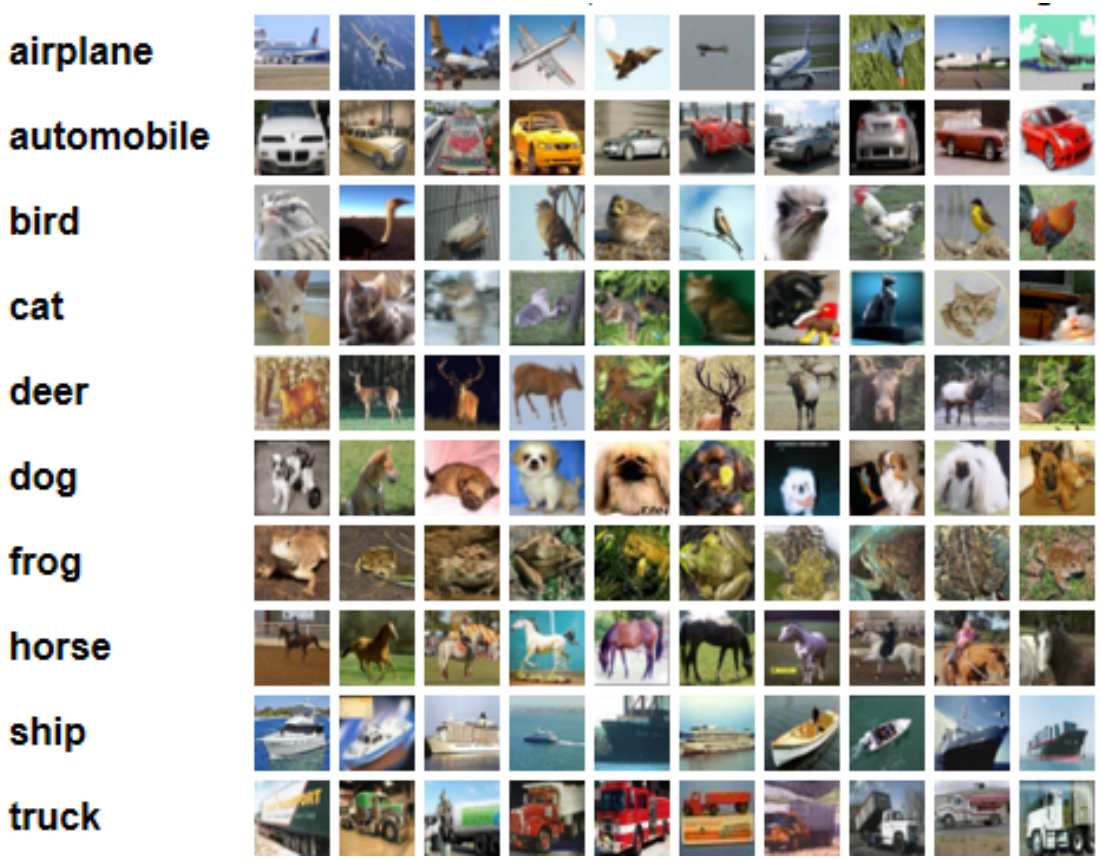
Method	Digits	Fashion
Logistic Regression	92%	82.6%
RBF-SVM	98.4%	89.8%
2 layer DNN	98.3%	90.4%
CNN	**99.2**	**92.6%** [2]

Figure 27: Accuracy of classifiers in the raw pixels of the MNIST digit and fashion data sets

[2] A larger network with 4 convolutional layers and 450k trainable parameters achieves 93.5%

CIFAR10

MNIST is a fairly easy Machine Learning problem. As we have seen, many different methods can achieve fairly good performance.
CIFAR10 is a more challenging data set consisting of 50,000 32×32 color images for a feature space of $3,072$ inputs, nearly 4 times as large as MNIST.



CIFAR-10 II

A small convolutional network ($\approx$ 590k parameters)

Layer	Filter Size	Features	Pooling	Dropout	Dimensions
Conv1	(5,5)	32	(2,2)	30%	$16 \times 16 \times 32$
Conv2	(5,5)	64	(2,2)	30%	$8 \times 8 \times 64$
Conv3	(5,5)	128	(2,2)	25%	$4 \times 4 \times 128$
Dense		64		25%	128

Figure 28: Small CIFAR10 Convolutional Network with 77% accuracy

can achieve **77% accuracy** in a few hours of training.
For comparison we have the following baselines:

Method	Accuracy
Logistic Regression	25%
3-layer DNN	46%
Small CNN	$\approx$ 75%

Figure 29: Small CIFAR10 Convolutional Network with 75% accuracy

CIFAR-10 III

A larger network ($\approx$ 1.3 million parameters)

Layer	Filter Size	Features	Pooling	Dropout	Dimensions
Conv1	(3,3)	32		25%	$32 \times 32 \times 32$
Conv2	(3,3)	32	(2,2)	25%	$16 \times 16 \times 32$
Conv3	(3,3)	32		25%	$32 \times 32 \times 32$
Conv4	(3,3)	64	(2,2)	25%	$8 \times 8 \times 64$
Conv5	(3,3)	64		25%	$8 \times 8 \times 64$
Conv6	(3,3)	1024	(2,2)	25%	$4 \times 4 \times 128$
Pool		1024	(4,4)		$1 \times 1 \times 1024$
Dense1		512		40%	512
Dense2		256		40%	256

Figure 30: Larger CIFAR10 Convolutional Network with 85.2% accuracy

- achieves an 85.2% accuracy.
- an **ensemble** of 7 separately trained networks achieves **88%** accuracy.
- Training takes 12 hours with an NVDIA GPU.
- Training on CPU takes roughly **50 times longer**.

Transfer Learning

- Steva's paper on skin cancer image classification [8] makes use of **transfer learning**.
- Uses a **pre-trained** convolutional network: Google's Inception v3 trained on generic images from ImageNet.
- Weights for all the **convolutional** layers are fixed.
- Dense layers are **re-trained** on the task at hand (skin cancer training data).

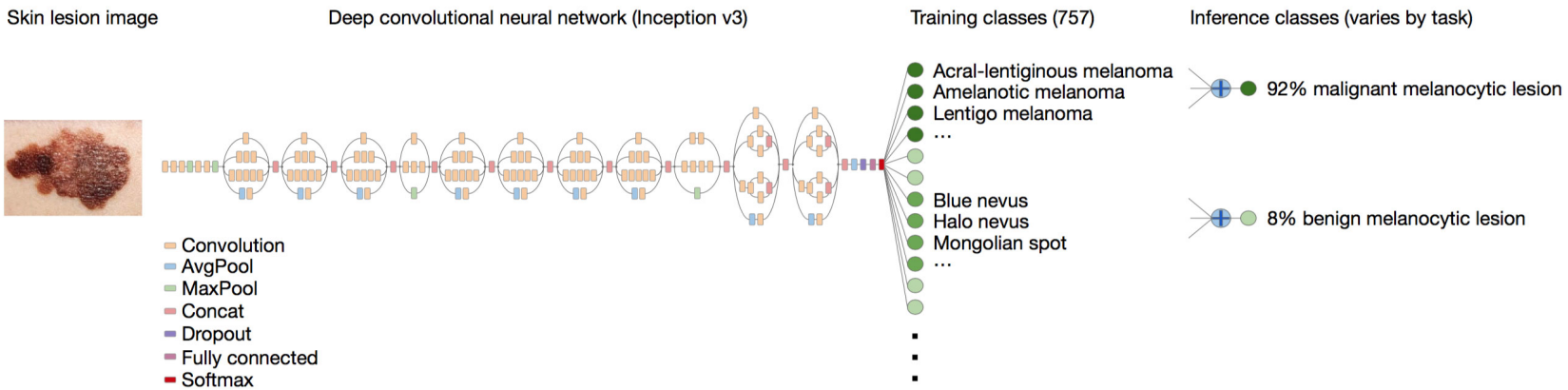


Figure 31: Neural Network Architecture of Steva et al

By now, we have covered all Machine learning concepts employed in [8]. You should read the paper again.

Urban Sound Classification

With a MFCC input of 20 frequencies on 173 0.1s frames, a small network of

Layer	Filter Size	Features	Pooling	Dropout	Dimensions
Conv1	(2,4)	16	(2,4)	20%	$10 \times 43 \times 8$
Conv2	(2,4)	32	(2,4)	30%	$5 \times 10 \times 32$
Conv3	(2,4)	64	(2,4)	30%	$2 \times 2 \times 64$
Dense1		128		20%	128
Dense		128		40%	128

Figure 32: Architecture of Urban Sound's Convolutional Network trained on MFCC features that achieves 90.3% accuracy. Note that frequency and time dimensions are treated asymmetrically.

can achieve a 90.3% accuracy.

Features	Feature Dimension	Logistic	DNN	CNN
Mel Spectrum	173×20	41%	79.4%	86.9%
MFCC	173×20	48%	86%	90.3%
Hand crafted	193	68%	94.4%	

Figure 33: Comparison of different methods and features on the Urban Sounds Problem.

Google Speech Commands

With a MFCC input of 20 frequencies on 44 0.1s frames, a convolutional network with architecture

Layer	Filter Size	Features	Pooling	Dropout	Dimensions
Conv1	(1,2)	32	(1,2)	0%	$20 \times 22 \times 16$
Conv2	(2,2)	64	(2,2)	0%	$10 \times 11 \times 32$
Conv3	(1,2)	64	(1,2)	0%	$10 \times 5 \times 64$
Conv4	(1,2)	128	(1,2)	0%	$5 \times 2 \times 128$
Conv5	(1,2)	128	(2,2)	50%	$2 \times 2 \times 128$
Dense1		256		0%	256
Dense2		256		50%	256

Figure 34: Architecture of Urban Sound's Convolutional Network trained on MFCC features that achieves 91.1% accuracy. Note that frequency and time dimensions are treated asymmetrically.

achieves a 91.1% accuracy.

Features	Feature Dimension	Logistic	DNN	CNN
MFCC	44×20	35%	85.8%	**91.8%**

The best network is **very large**!

Word Embedding

Review

A bag of word representation of a documents loses information about **word order** and ignores information word meanings. And alternative representation is a

Word Embedding

Given a vocabulary of size V, and a low dimensional vector space D (in the order of a few hundred dimensions).
A word embedding is a $V \times D$ mapping $E_{w,d}$ that assigns to each work w a D dimensional vector $E_{w,d}$

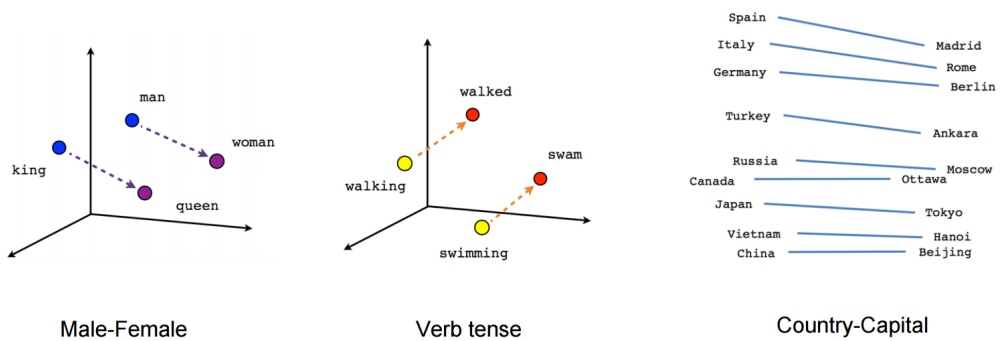


Figure 35: Low dimensional word embedding

Ideally, a word embedding maps sematically related words close to each other in the embedding space.

Neural Networks with Word Embeddings

Review

If we define the logit function as

$$\eta(x) = \sum_t x_{t,d} = \sum_t E_{w_t,d} = \sum_w n_w E_{w,d} \tag{22}$$

where $D = K$, and n_w is the **count** of word w.
This model is **equivalent** to logistic regression on word count features.
But we can also, add a couple of hidden layers between the Embedding and the final ouput

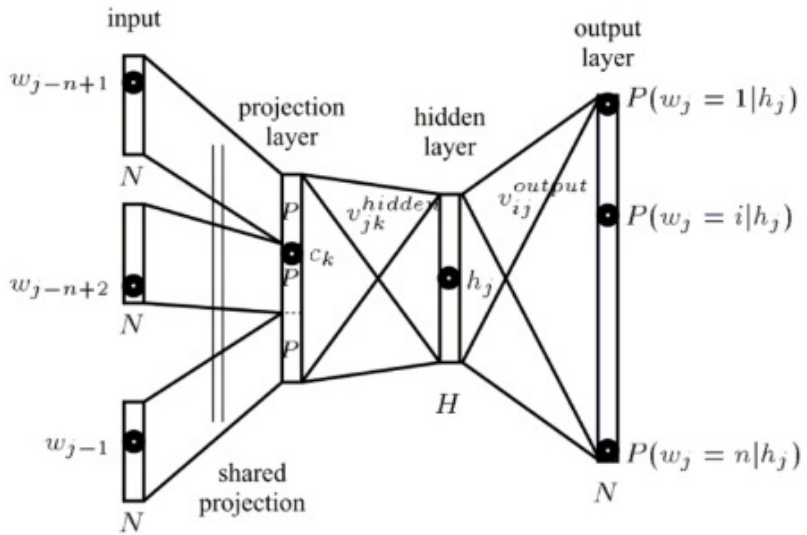


Figure 36: Dense Neural Network Language Model architecture. Source: Bengio et al 2006

Convolutional Networks for Text

In a document with length W represented as a D dimensional embedding $X_{w,d}$ we can represent the interaction between K close by words as the **1 dimensional convolution**

$$c_w = \sum_{i=1}^{K} \sum_{d} w_{i,d} X_{w+i,d} \tag{23}$$

and we can pick up the **most important** cluster of K words by a polling operation

$$\hat{c} = \max_{w} c_w \tag{24}$$

We, do this over many different filters $w^o_{i,d}$ and a few different values of K:

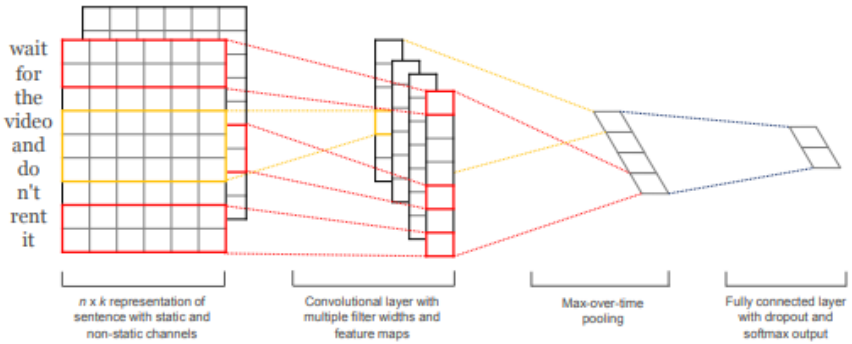


Figure 37: Architecture for text CNN. Source: [2]

CNN Amazon Fine Food Reviews

We define a Text CNN with an embedding dimension $E = 300$. The following convolutional filters

Words	Features
3	256
4	256
5	256

Figure 38: Word combinations of Text CNN achieving a 93.2% accuracy

plus a single, 512 node, dense layer. It achieves a 93.2% accuracy.
A summary of our results so far

Method	Accuracy
All positive	78%
Logistic Regression	88.4%
2-layer DNN	91.8 %
CNN	**93.9%**

Figure 39: Comparison of the performance of different methods in the Amazon Fine Reviews data set

Conclusions

Today we discussed Convolutional Neural Networks

- The incorporate strong **prior** assumptions about locality and translational invariance
- Ideally suited for Image classification problems.
- Can be adapted with small modifications to sound or NLP problems.
- CNN are **practical** with (and mostly **require**) very large data sets.
- The number of possible architectures (hyper parameters) is very large. Network architecture is an intense are area of research.
- CNNs are well adapted to acceleration by specialized hardware: **GPU**, **TPU**.
- CNN like architectures are **state of the art** for many machine learning problems.
- Next week we will discuss **Recurrent Neural Networks**: architectures for learning sequenced data.

Bibliography I


Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep Learning.
MIT Press, 2016.
http://www.deeplearningbook.org.


Yoon Kim.
Convolutional neural networks for sentence classification, 2014.
http://www.aclweb.org/anthology/D14-1181.


Tara N. Sainath and Carolina Parada.
Convolutional neural networks for small-footprint keyword spotting.
In *INTERSPEECH*, 2015.
https://www.isca-speech.org/archive/interspeech_2015/papers/i15_1478.pdf.


Navneet Dalal and Bill Triggs.
Histograms of oriented gradients for human detection.
In *Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.
https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf.

Bibliography II


Maxine Tan, Jiantao Pu, and Bin Zheng.
Optimization of breast mass classification using sequential forward floating selection (sffs) and a support vector machine (svm) model.
International Journal of Computer Assisted Radiology and Surgery, 2014.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176547.


P. Tiwari, P. Prasanna, L. Wolansky, M. Pinho, M. Cohen, A.P. Nayate, A. Gupta, G. Singh, K. Hattanpaa, A. Sloan, L. Rogers, and A. Madabhushi.
Computer-extracted texture features to distinguish cerebral radionecrosis from recurrent brain tumors on multiparametric mri: A feasibility study.
American Journal of Neuroradiology, 2016.
https://doi.org/10.3174/ajnr.A4931.


J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.
ImageNet: A Large-Scale Hierarchical Image Database.
In *CVPR09*, 2009.

Bibliography III

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun.
Dermatologist-level classification of skin cancer with deep neural networks.
Nature, 542:115–118, 2017.
https://cs.stanford.edu/people/esteva/nature/#!