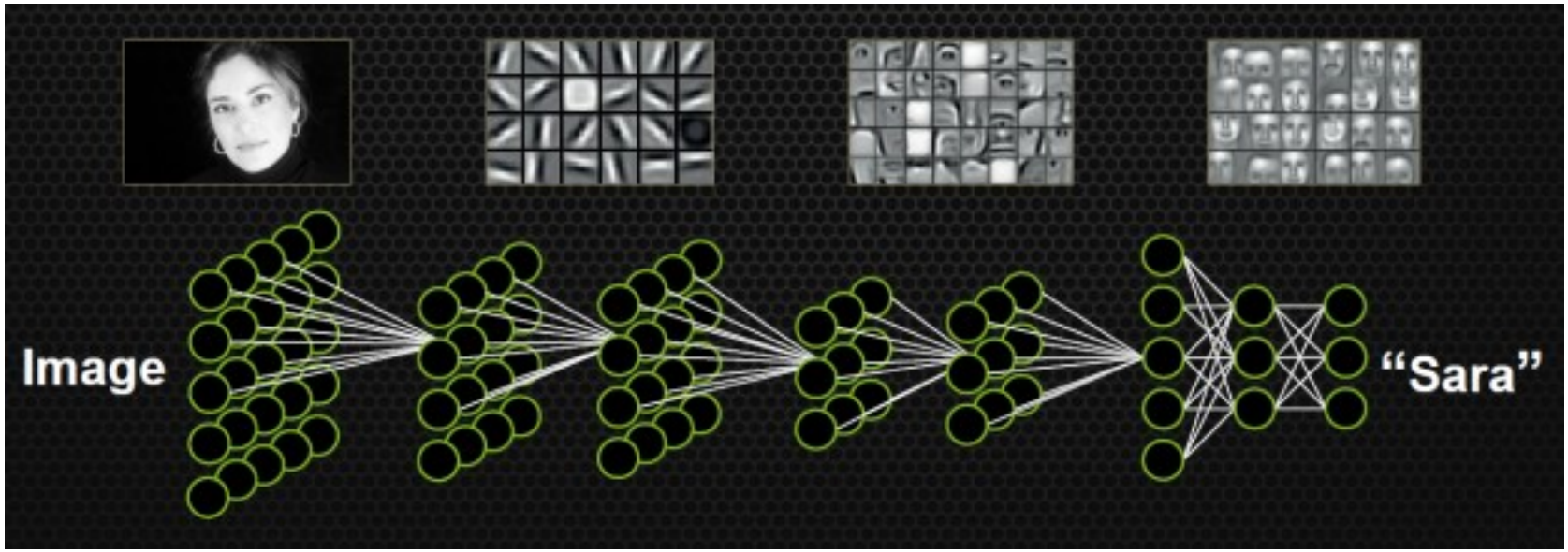
Inspire…Educate…Transform.

Convolutional Neural Networks

Dr. Kishore Reddy Konda

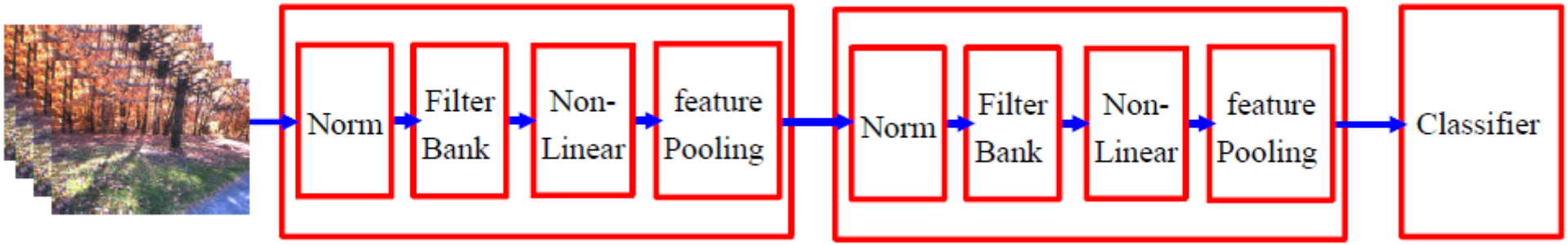
Mentor, International School of Engineering

Why do we need deep architectures?



Hierarchy of features learned on face images in a classification task

More “Non-Linear Expansion → Pooling” blocks stacked together



Normalization: variations on whitening

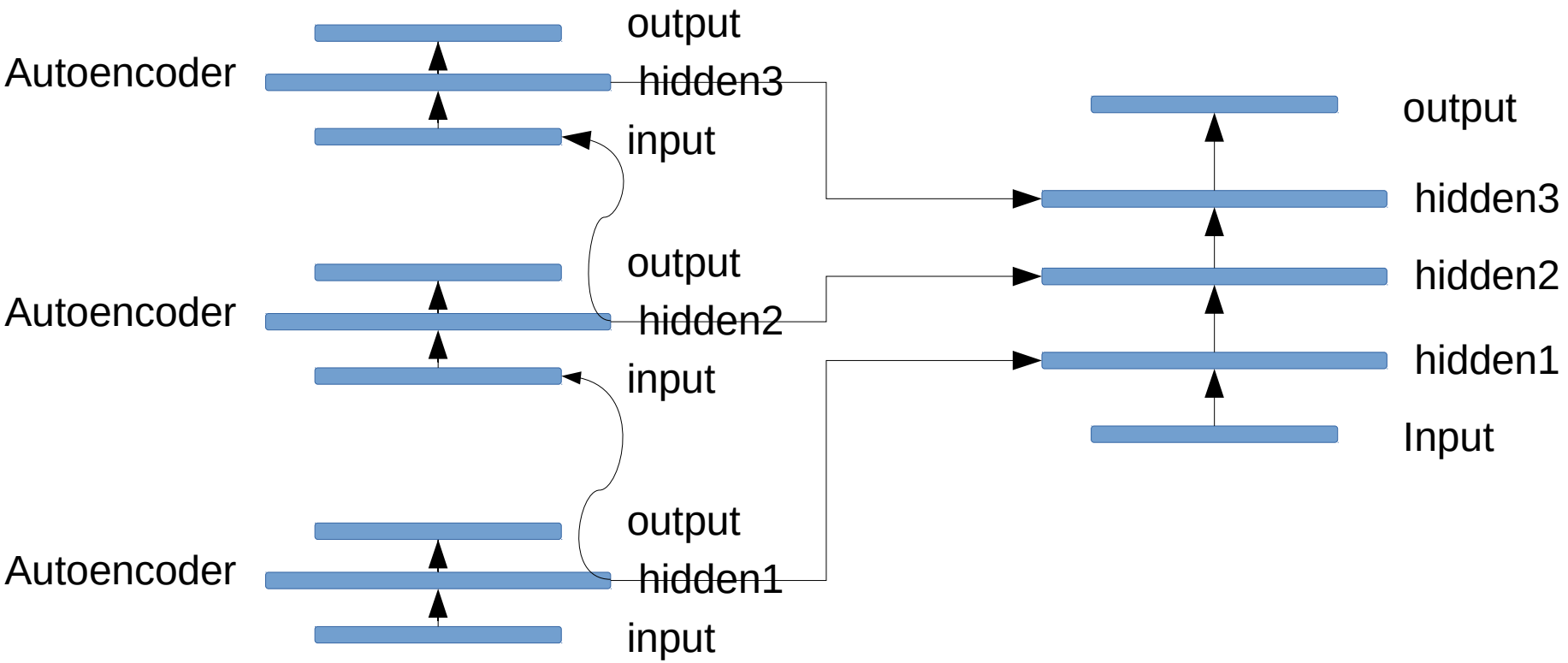
Filter Bank: dimension expansion, projection on overcomplete basis

Non-Linearity: sparsification, saturation, lateral inhibition

Pooling: aggregation over space or feature type

Y. LeCun,M.A. Ranzato. ICML-2013

Stacked unsupervised models as Deep MLP



Each layer weights are initialized using an unsupervised learning model.
Hidden representations of one model are input to the next one in the stack.
Stacked unsupervised models learn hierarchical features.

Auto-encoders

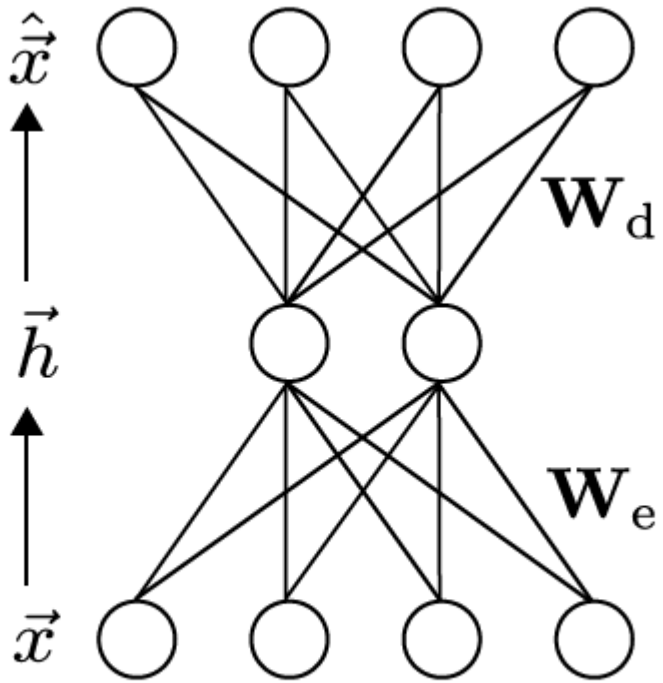
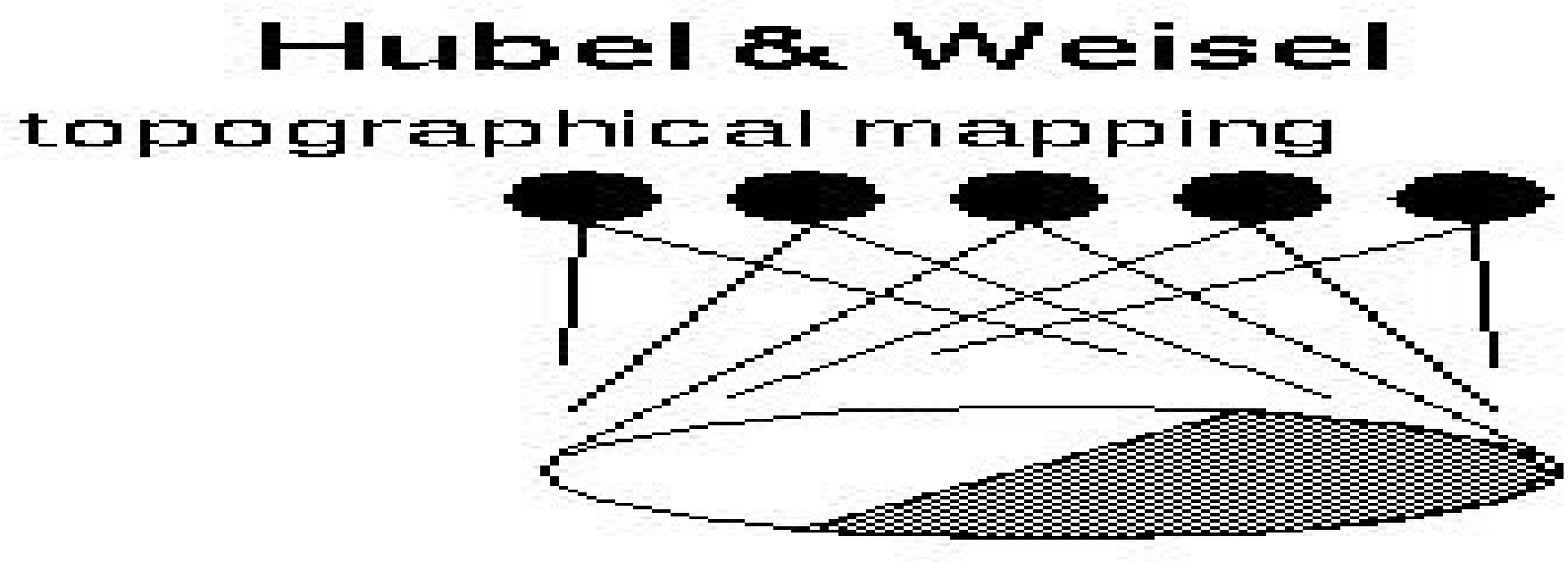


FIGURE 1.3: The standard autoencoder model.

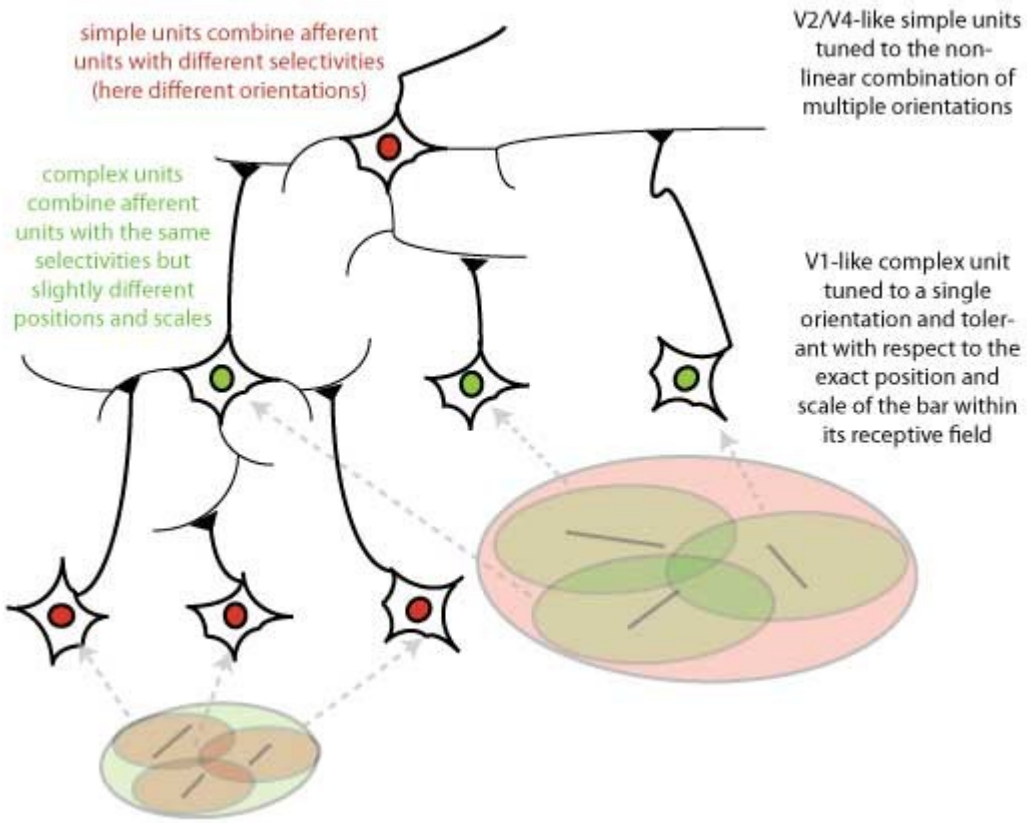
Inspiration from biology

Early 1968 (Hubel, D. H.;Wiesel, T. N. (1968-03-01)) work showed that the animal visual cortex contains complex arrangements of cells, responsible for detecting light in small, overlapping sub-regions of the visual field, called receptive fields.



Inspiration from biology

Simple and complex cells



http://serre-lab.clps.brown.edu/wp-content/uploads/2012/09/hierarchy.jpg

Inspiration from biology

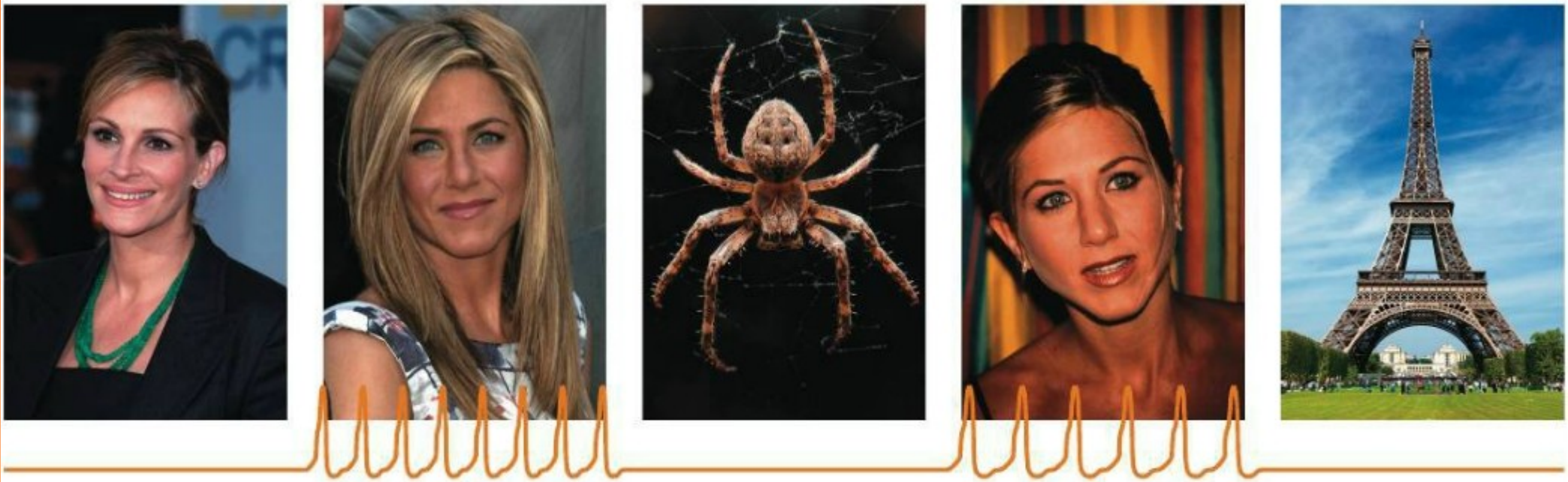
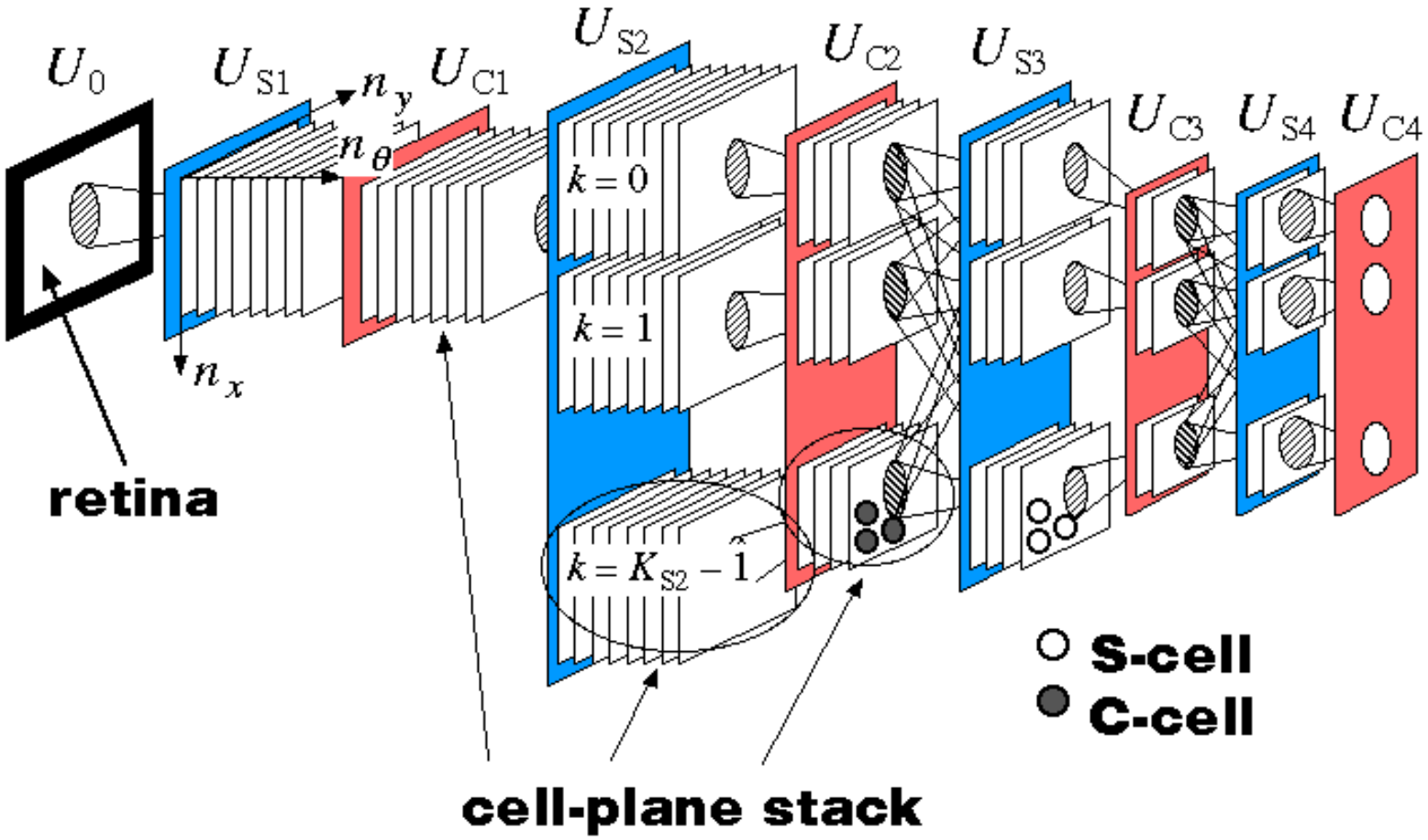


FIGURE 46.22 Single-Neuron Recording Reveals that Some Neurons in the Brain Recognize Specific Concepts. The graphs below each image show how a single neuron fires in response to images of actress Jennifer Aniston but not to other images.

DATA: Quiroga, R. Q., L. Reddy, G. Kreiman, et al. 2005. *Nature* 435: 1102–1107.

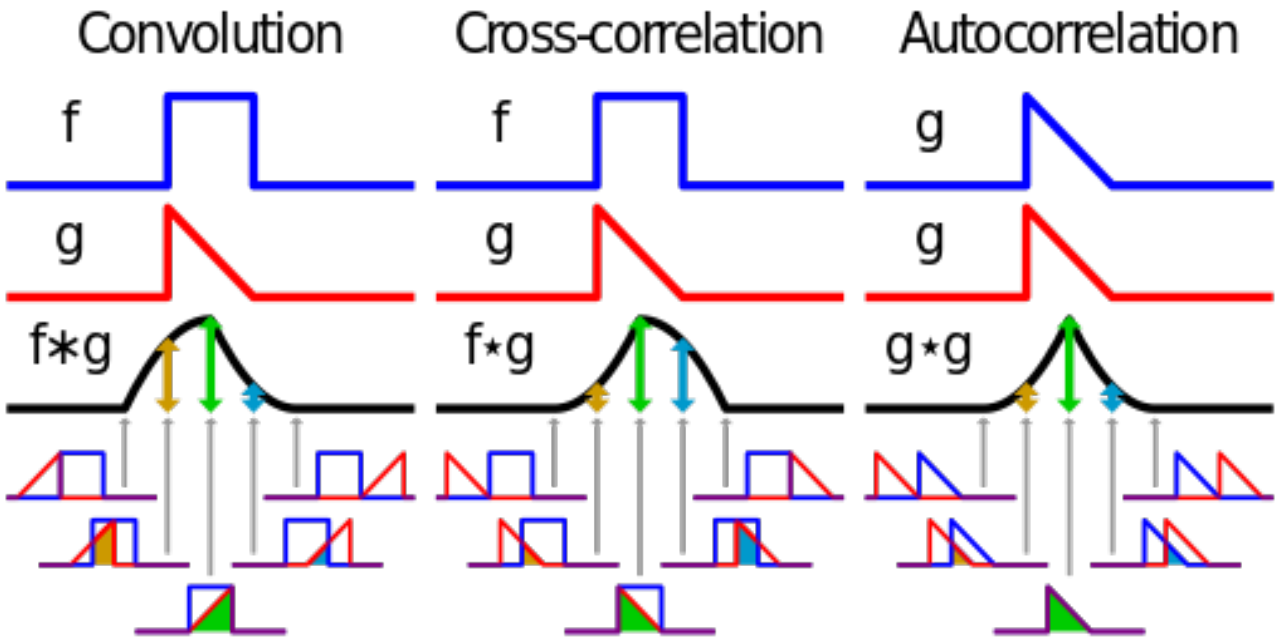
Neo-cognitron: Basis for modern day CNN



http://www.aso.ecei.tohoku.ac.jp/~shun/imgs/RNC.GIF

Convolution operation

$$f(t) * g(t) \stackrel{\text{def}}{=} \underbrace{\int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau}_{(f*g)(t)},$$



https://en.wikipedia.org/wiki/Convolution#/media/File:Comparison_convolution_correlation.svg

Convolution operation

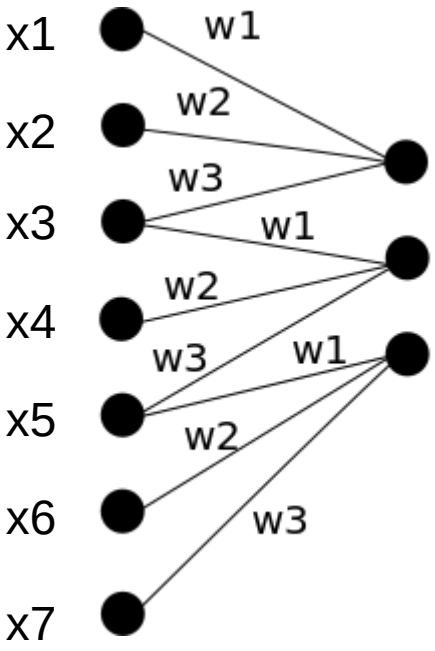
1D convolution example

http://www.fit.vutbr.cz/study/courses/ISS/public/demos/conv/

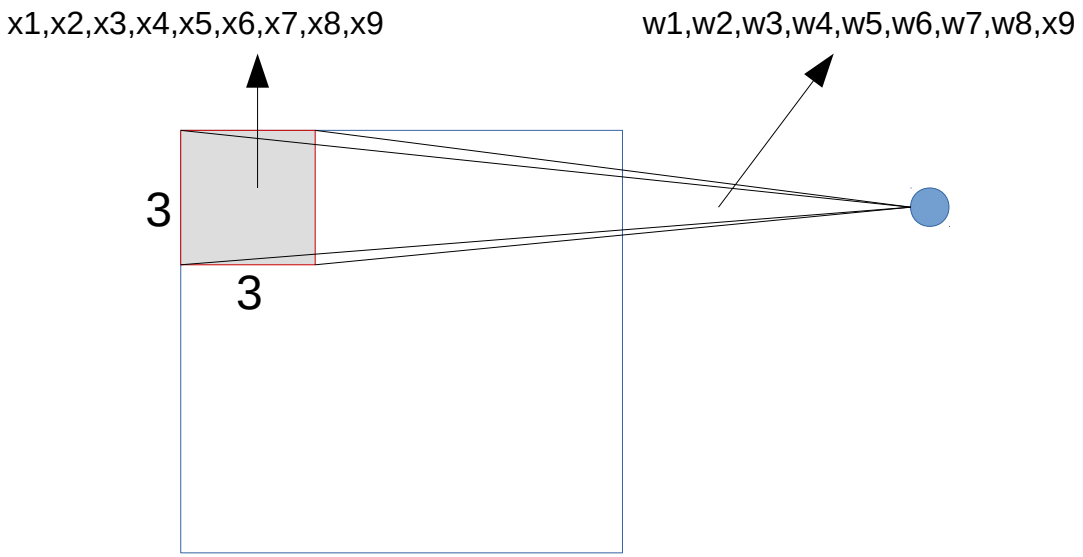
2D convolution example

https://graphics.stanford.edu/courses/cs178/applets/convolution.html

Convolution operation in ANN

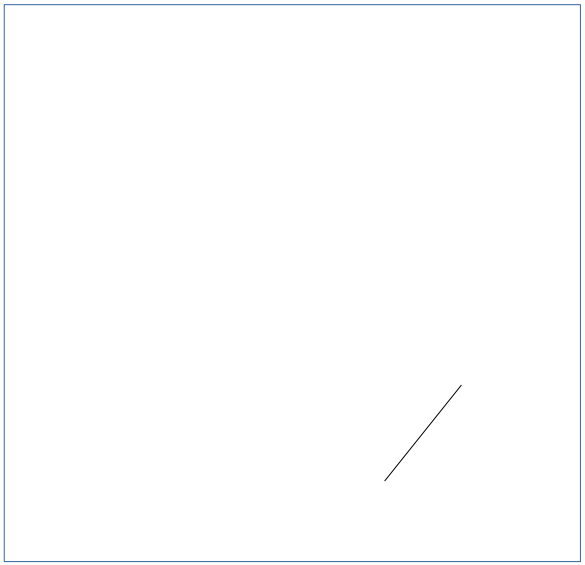


Three neurons applying same filter in three different locations of the input with a little overlap.



Convolution operation in ANN

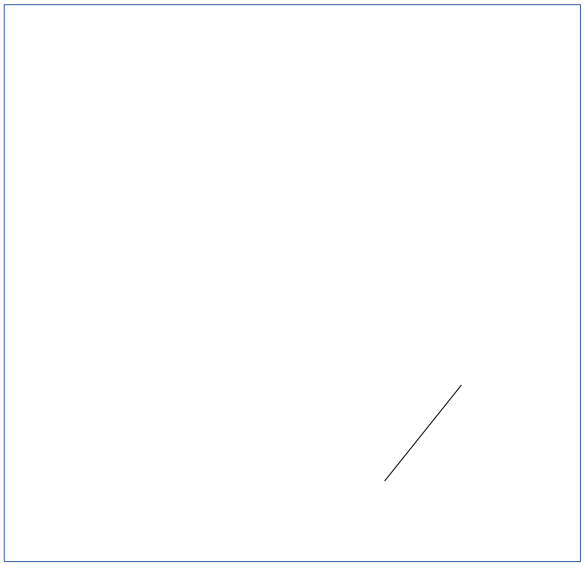
Two different input images

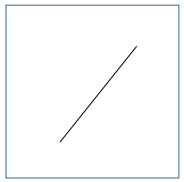


What should be the filters/features in an MLP to detect the lines in the two input images.

Convolution operation in ANN

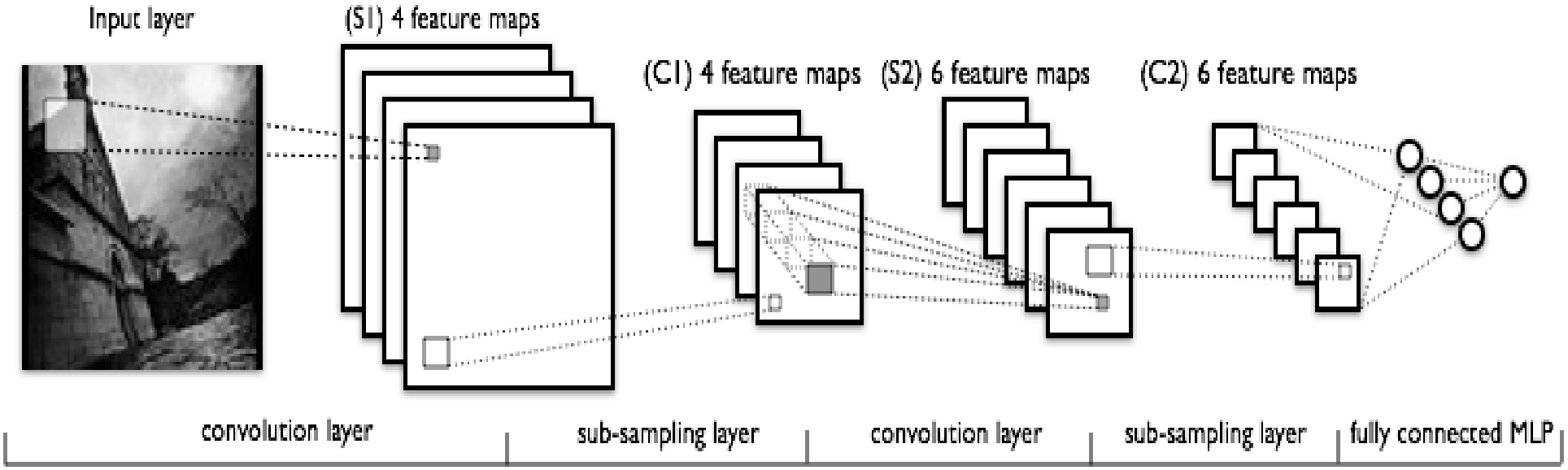
Two different input images



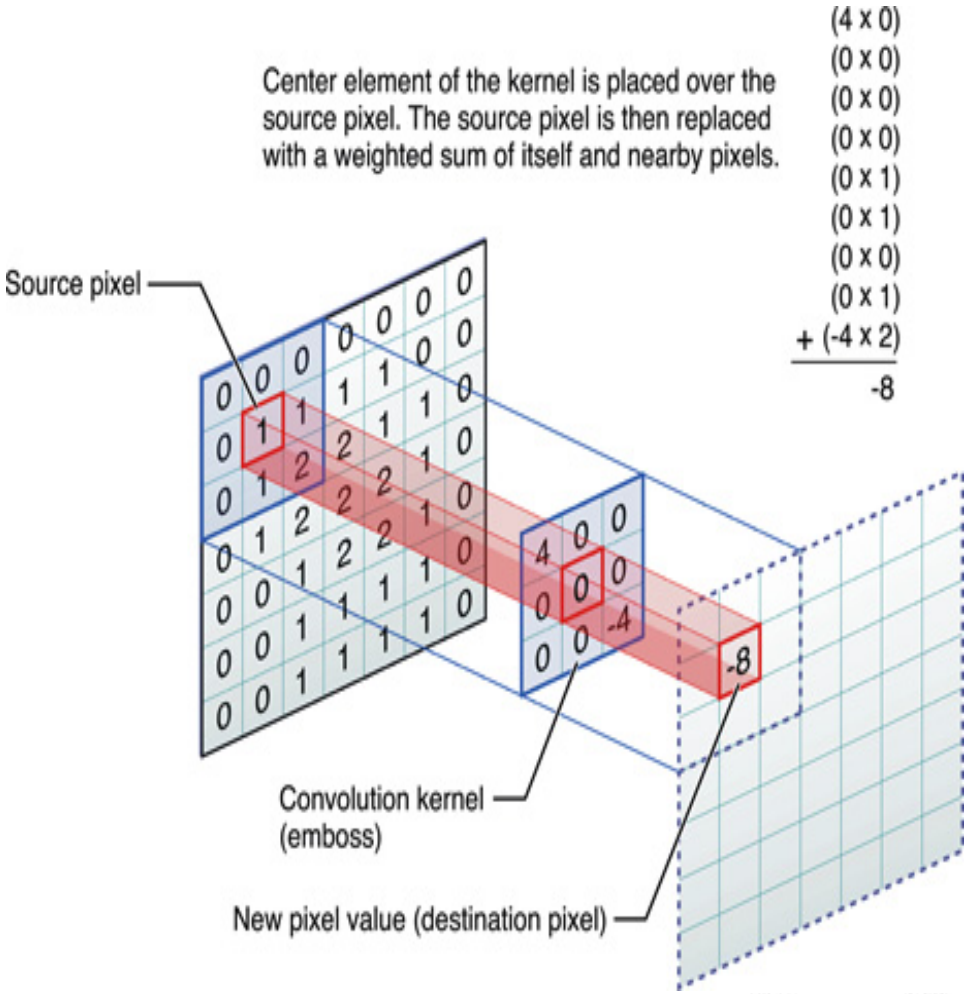


Will convolution with this single filter be able to detect the line in any location of the image?

Supervised learning: Convolutional Neural Networks- CNN



Convolution layer

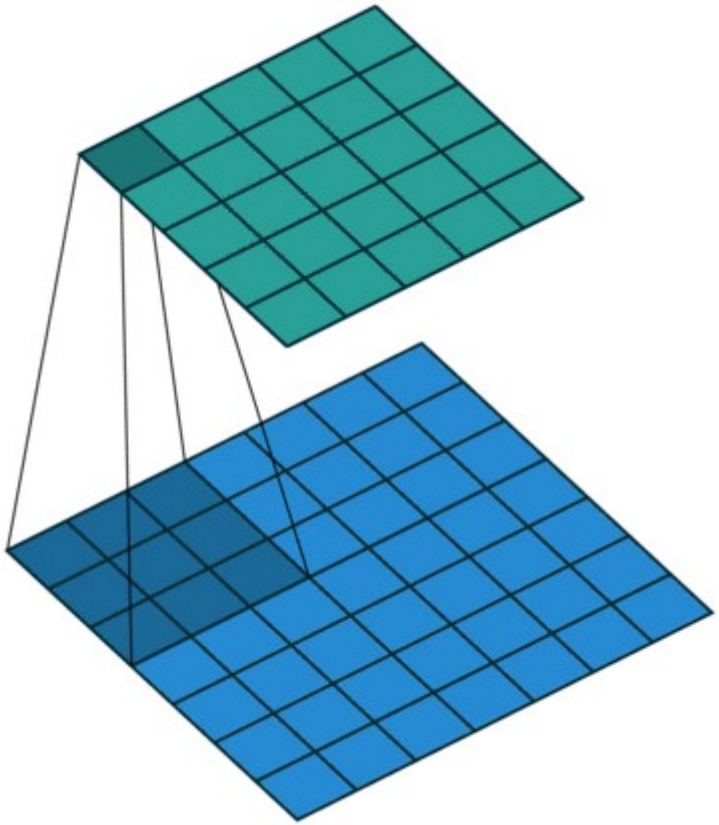


Convolution layer:

- Local connectivity
- Spatial arrangement
- Parameter sharing

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

Convolution operation on an image



Feature Map: Every cell in feature map is the result of applying (dot product) a kernel/filter/weight-matrix on a specific region of input.

An Input image represented as matrix of pixel values

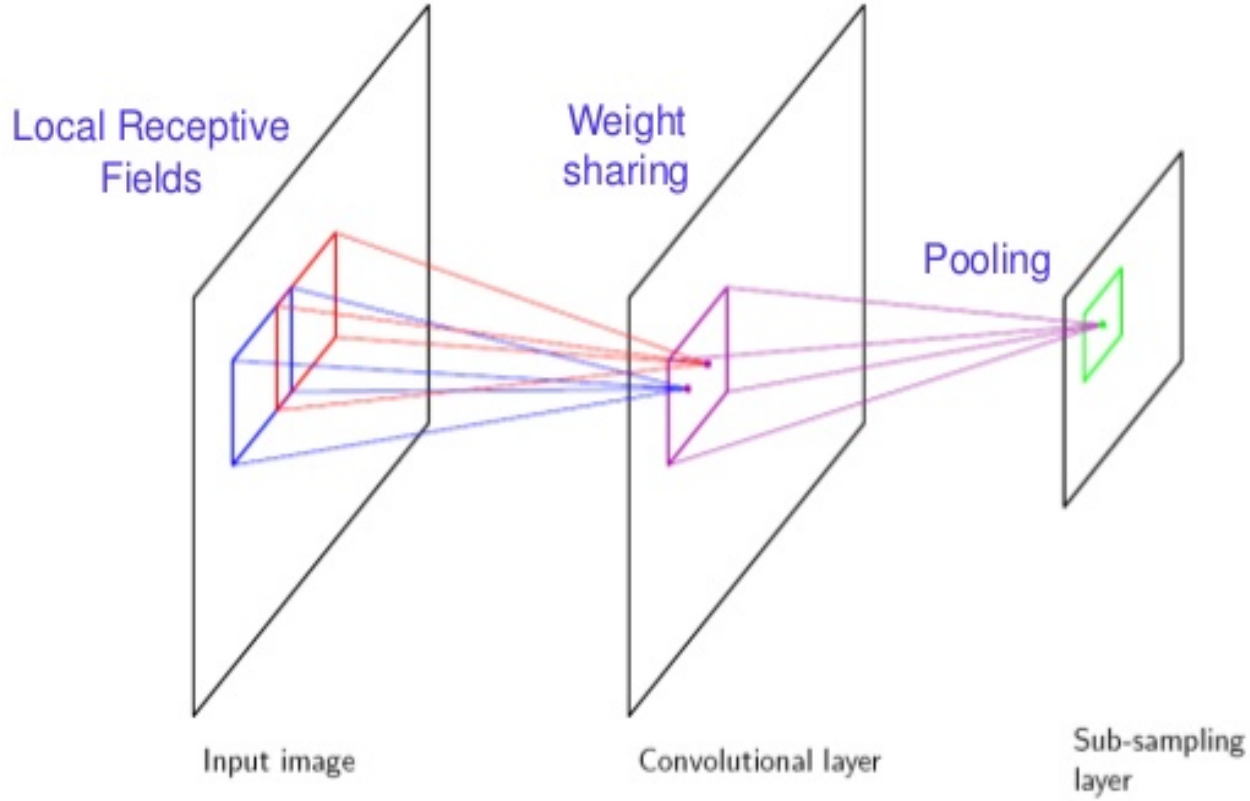
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

Blue is the input. Green is the resultant feature map. Shaded blue is kernel.

Convolution layer: local connectivity

(LeCun et al., 1989)



Convolution layer: spatial arrangement

Retaining spatial location information of a detected pattern

Kernel

	1	2	3	4	5
1	0	0	-1	1	0
2	0	0	-1	1	0
3	0	0	-1	1	0
4	0	0	-1	1	0
5	0	0	-1	1	0

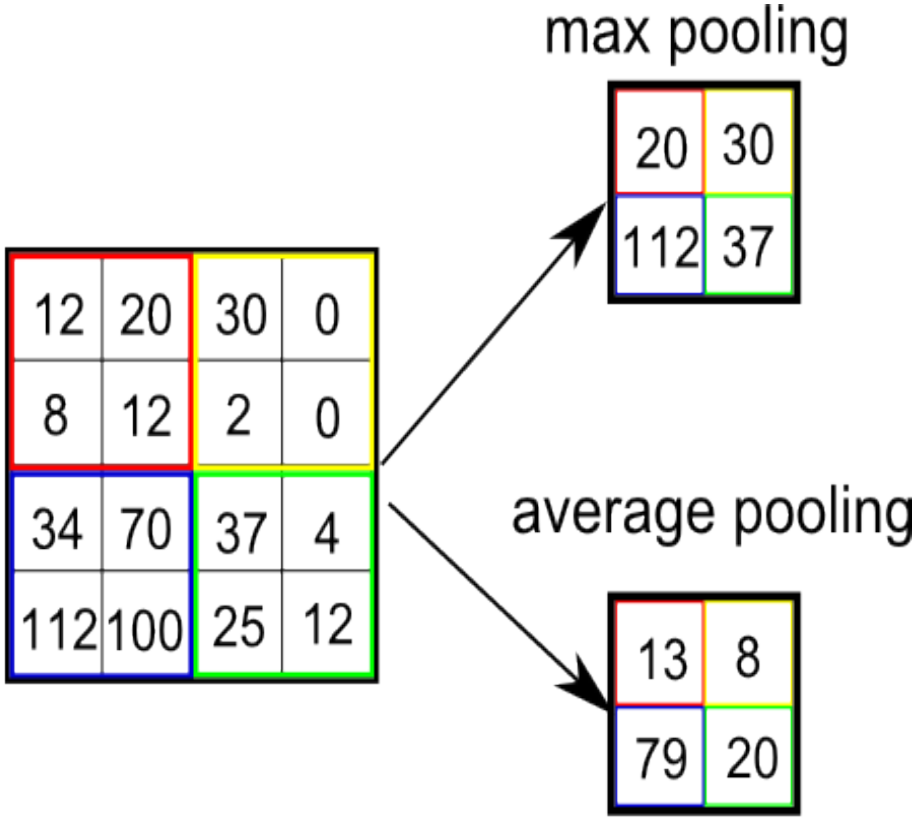
Convolution layer: parameter sharing

Using same kernel on multiple spatial locations

Kernel

	1	2	3	4	5
1	0	0	-1	1	0
2	0	0	-1	1	0
3	0	0	-1	1	0
4	0	0	-1	1	0
5	0	0	-1	1	0

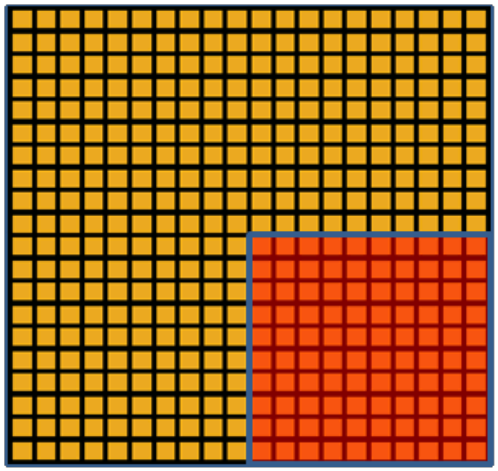
Pooling operation



Max-pooling layer:

- Local translational invariance
- dimensionality reduction

Pooling operation: local translational invariance



1	7
5	9

Convolved feature

Pooled feature

Pooling results in invariance to local translations in the input feature-map/image

No change in pooling layer output for small local shifts in input

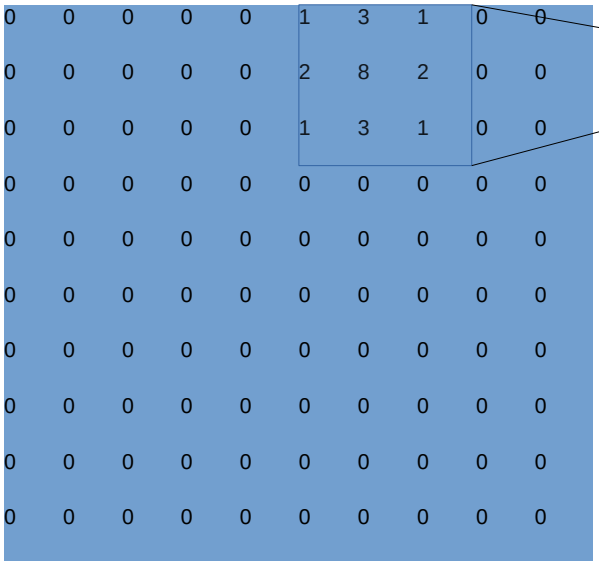
http://ufldl.stanford.edu/wiki/index.php/Pooling

Pooling operation: local translational invariance

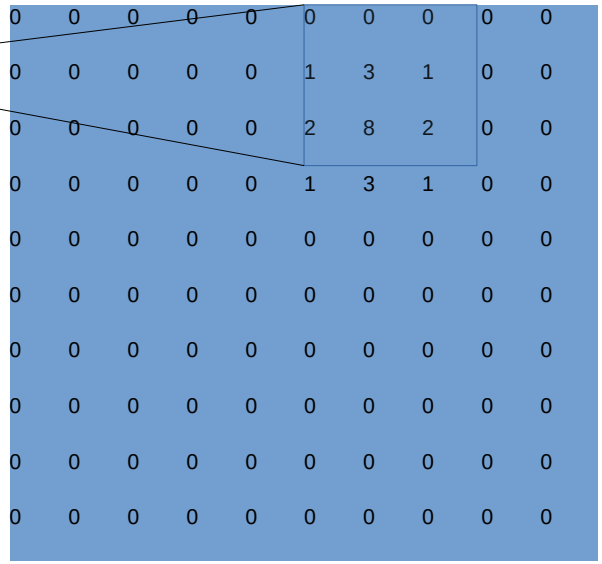
Input Image 1

Filter/kernel

Input Image 2

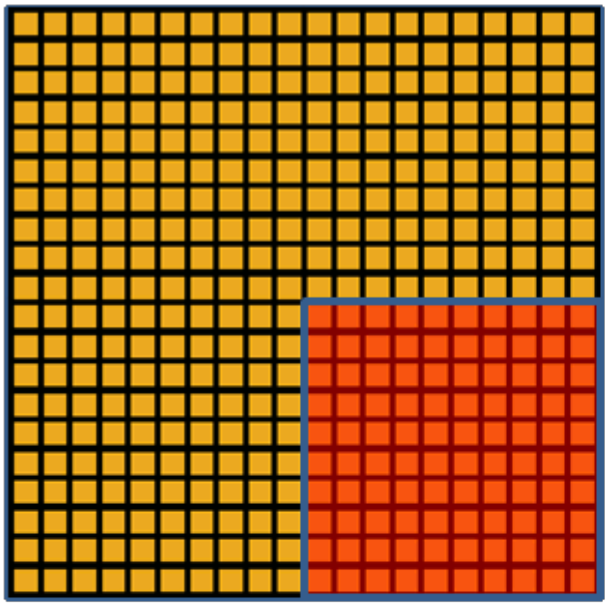


8



Feature map

Pooling operation: dimensionality reduction

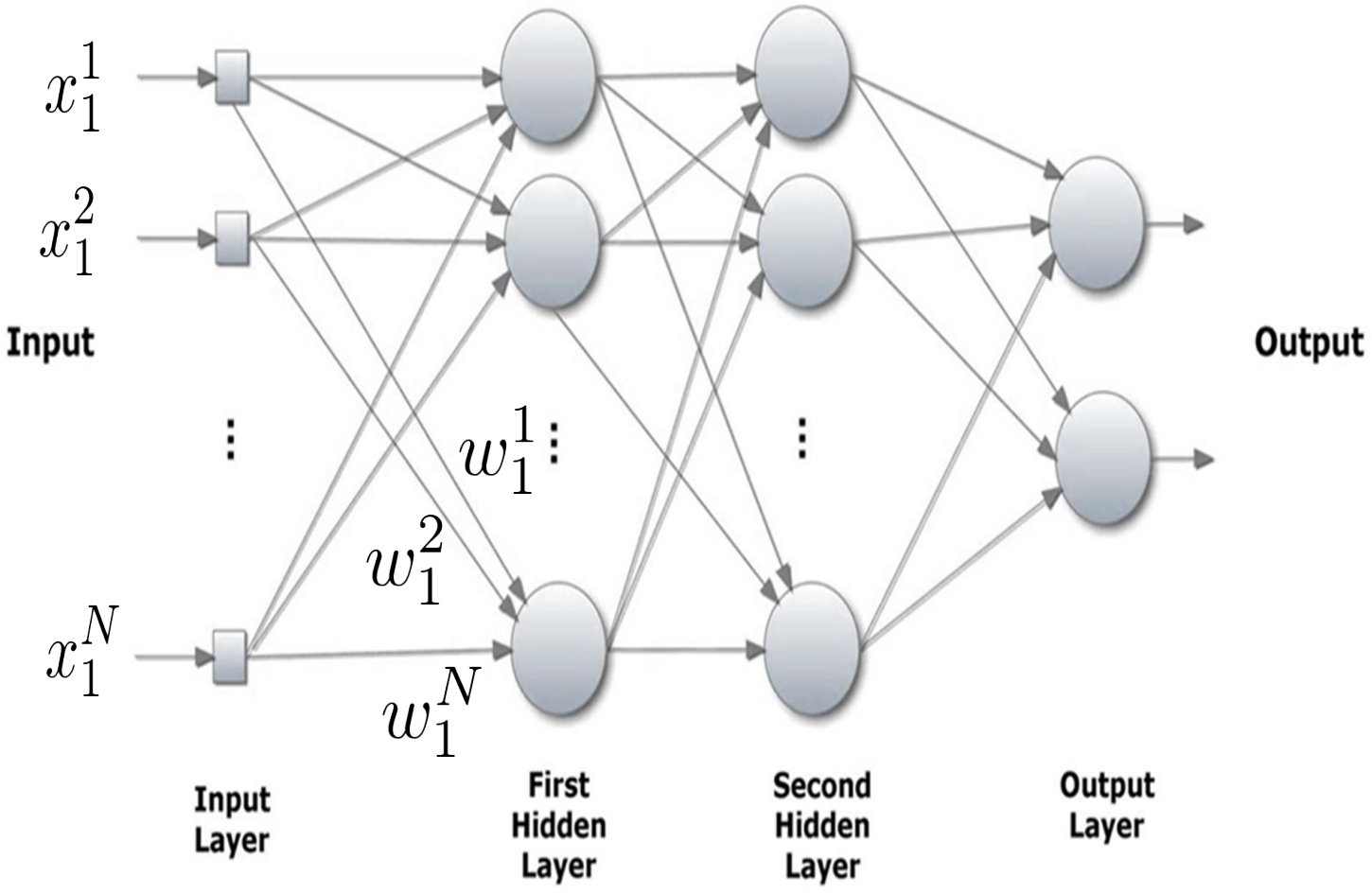


1	7
5	9

Convolved feature

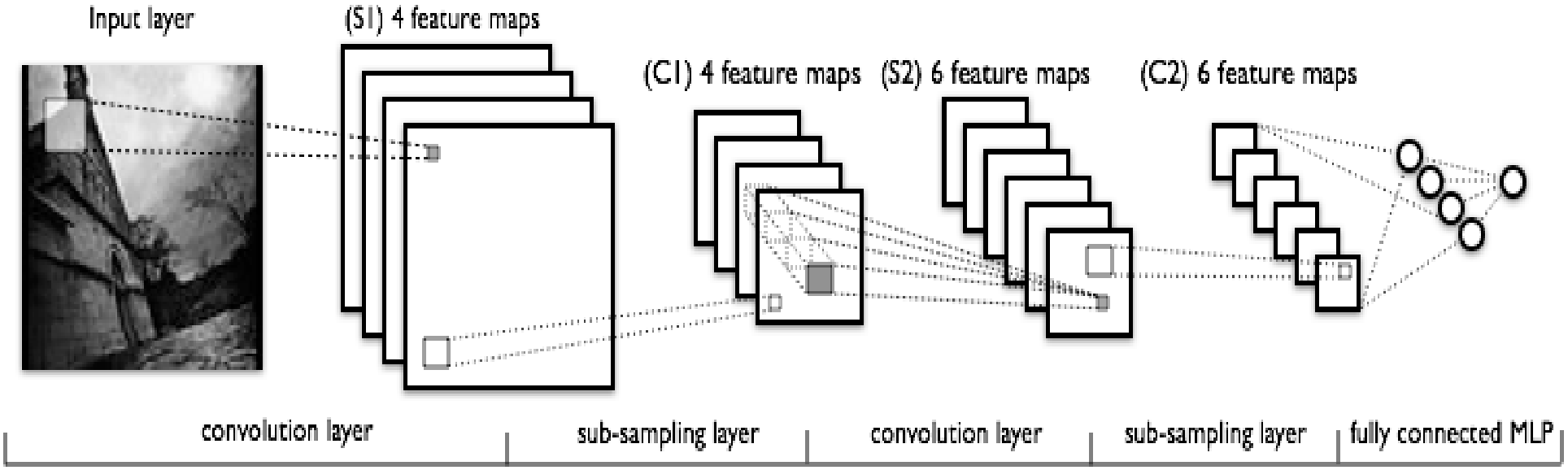
Pooled feature

MLP



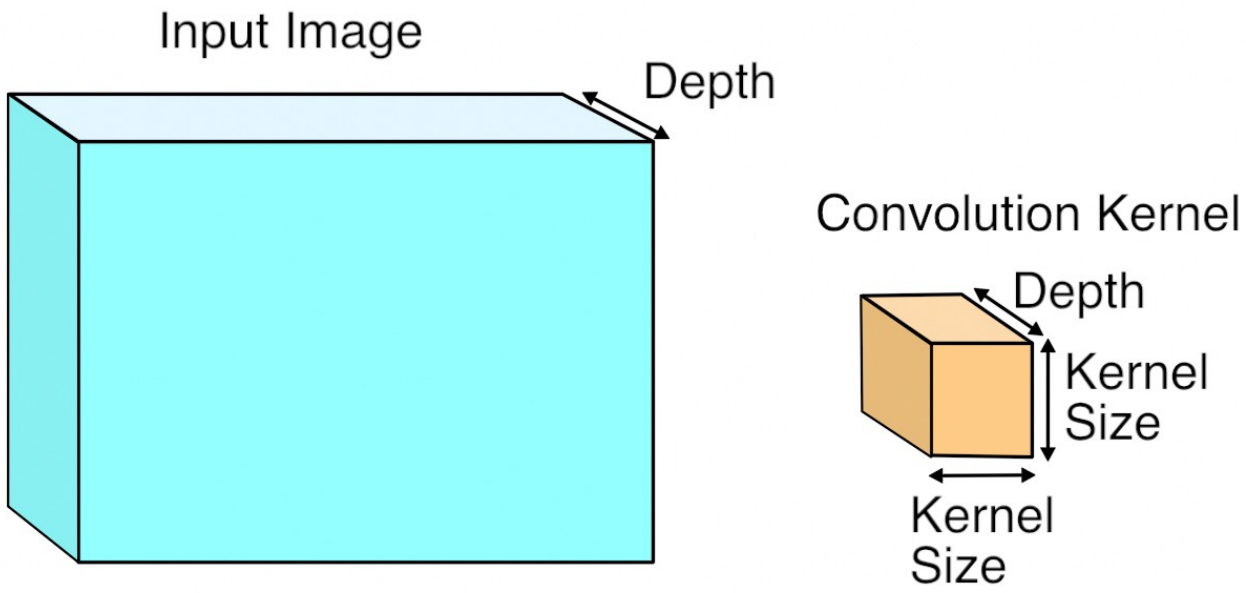
A multilayer perceptron (Feed forward network)

Supervised learning: Convolutional Neural Networks- CNN

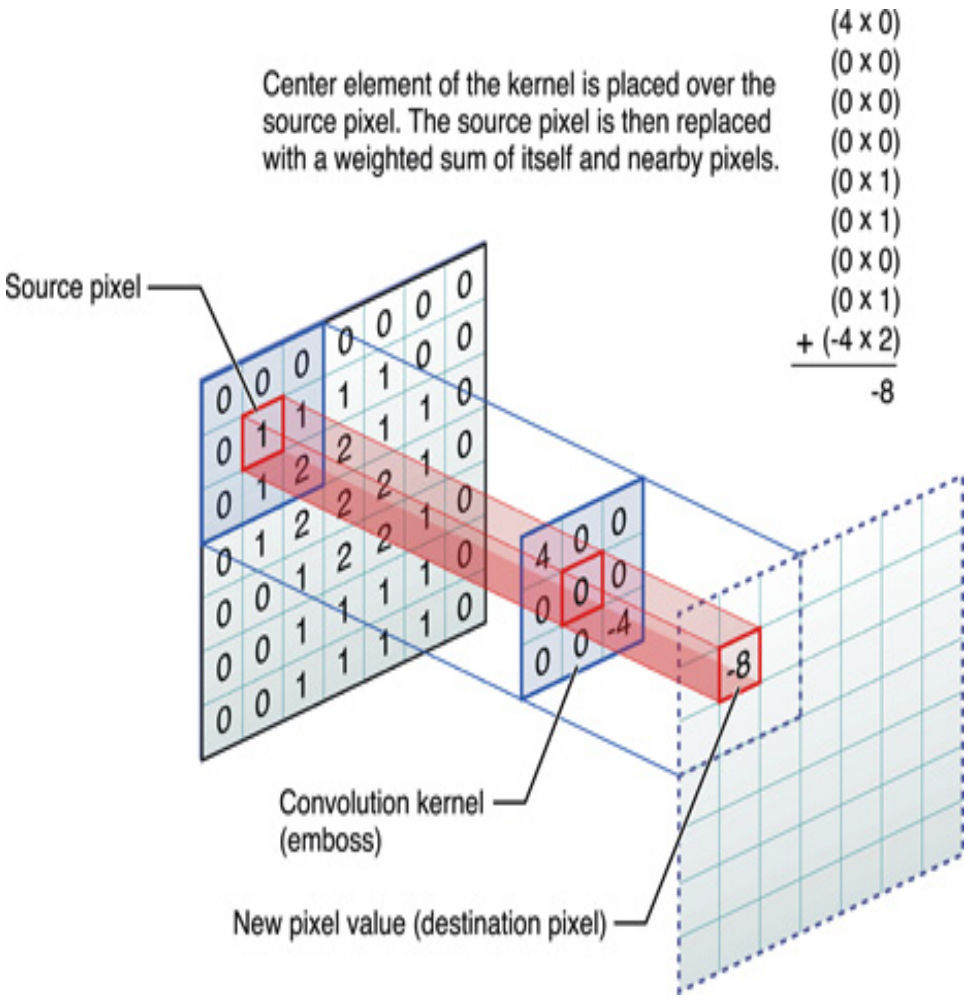


Feature maps/multi-channel image as input to convolution layer

What happens if the input is not a single channel image but a RGB image or set of feature maps (output of the previous layer?)



Hyper parameters in a convolutional layer



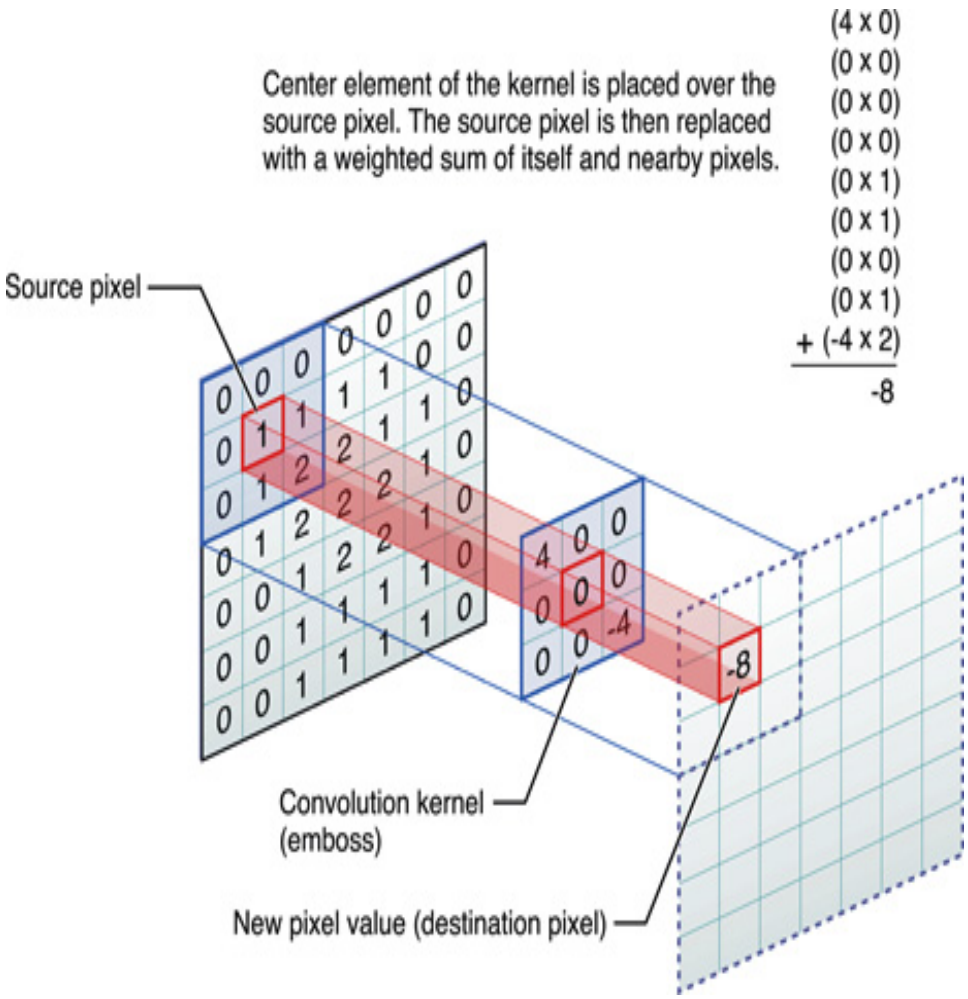
Convolutional stride

Number of kernels

Kernel size

Padding

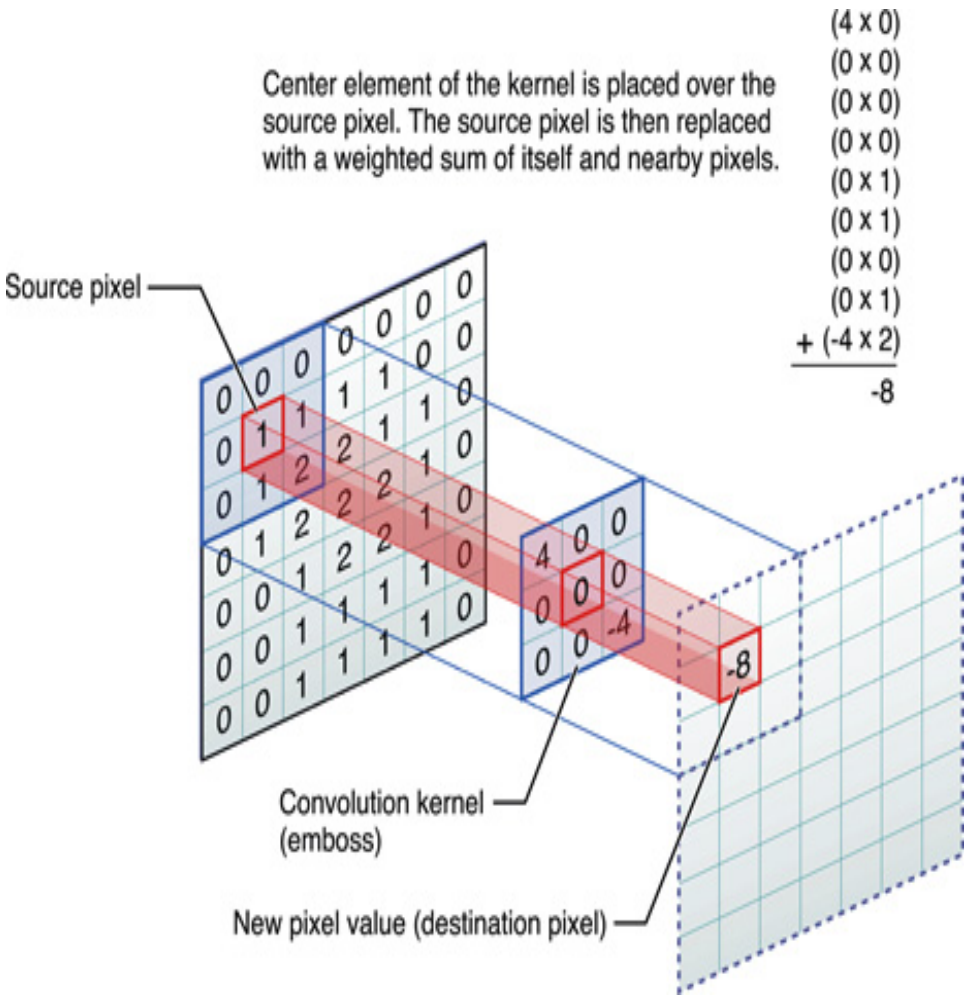
Hyper parameters in a convolutional layer



Convolutional stride

Rate at which the kernel is shifted during convolution operation

Hyper parameters in a convolutional layer



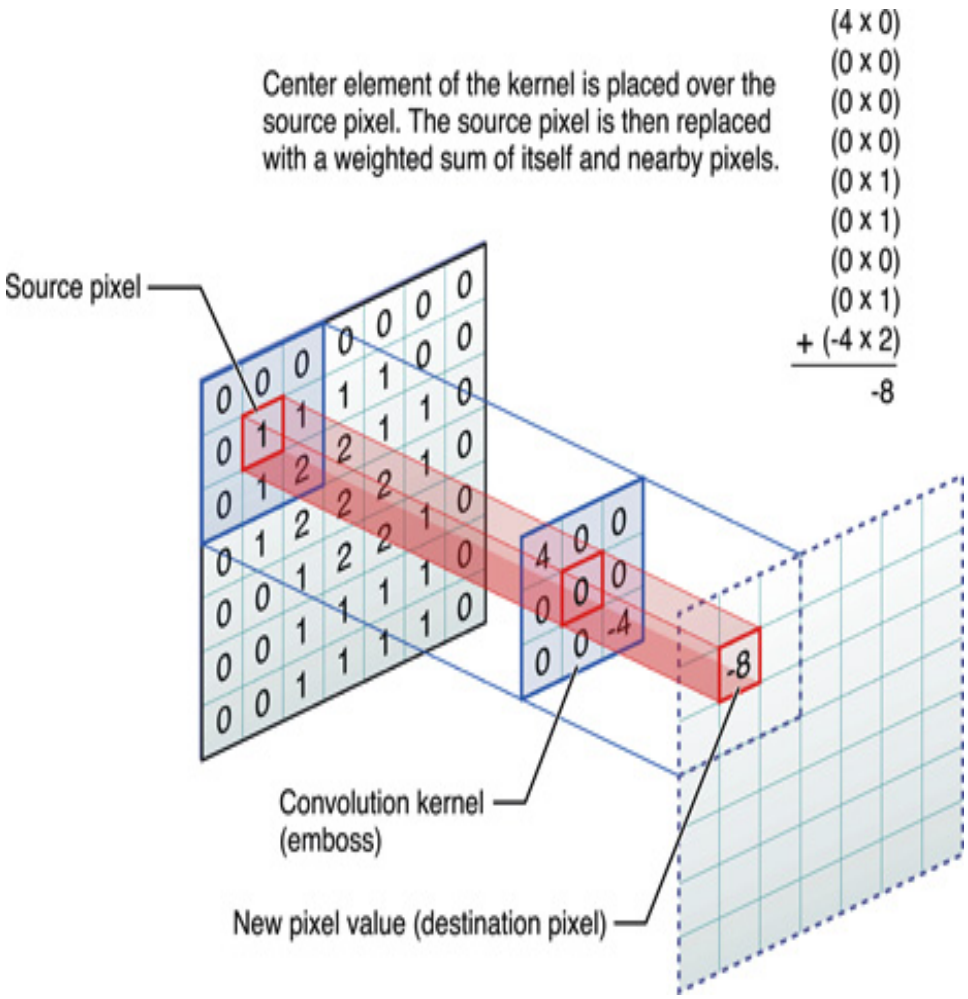
Number of kernels

Total number of kernels used in a convolutional layer

Usually the number increases as we go deeper into the network

Number of output feature maps is equal to number of kernels

Hyper parameters in a convolutional layer



Kernel size

kernel width X kernel height x Number of input feature maps

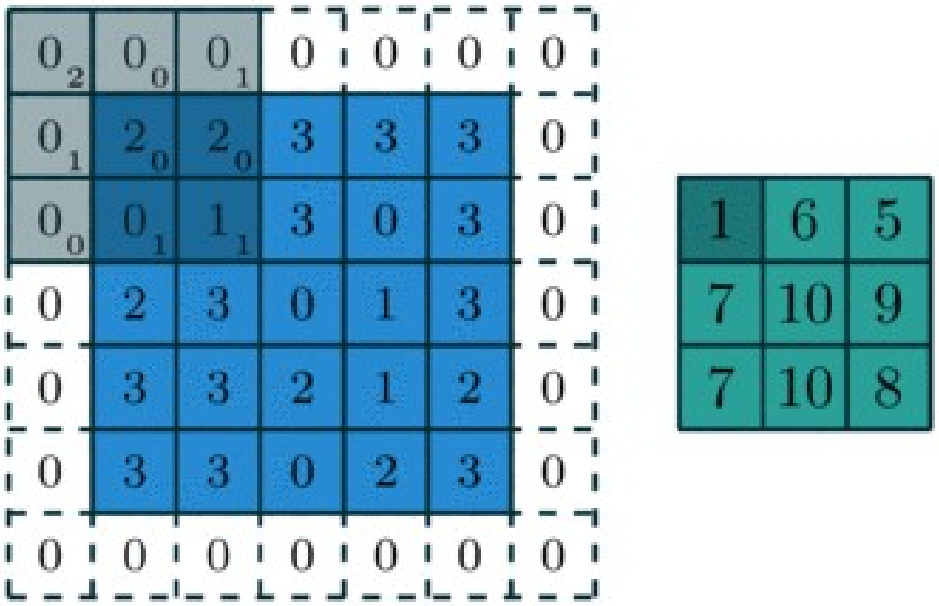
Example:

Input RGB image (3 channels/feature-maps)

Kernel wight = kernel height= 16

Kernel size: 16x16 x 3

Hyper parameters in a convolutional layer



Padding

Additional values added at borders of an input to the convolutional layer

Zero padding is most often used

Convolution layer: output Feature map size calculation

The input width size : W

The receptive field size / Kernel width or height of the Conv Layer: F

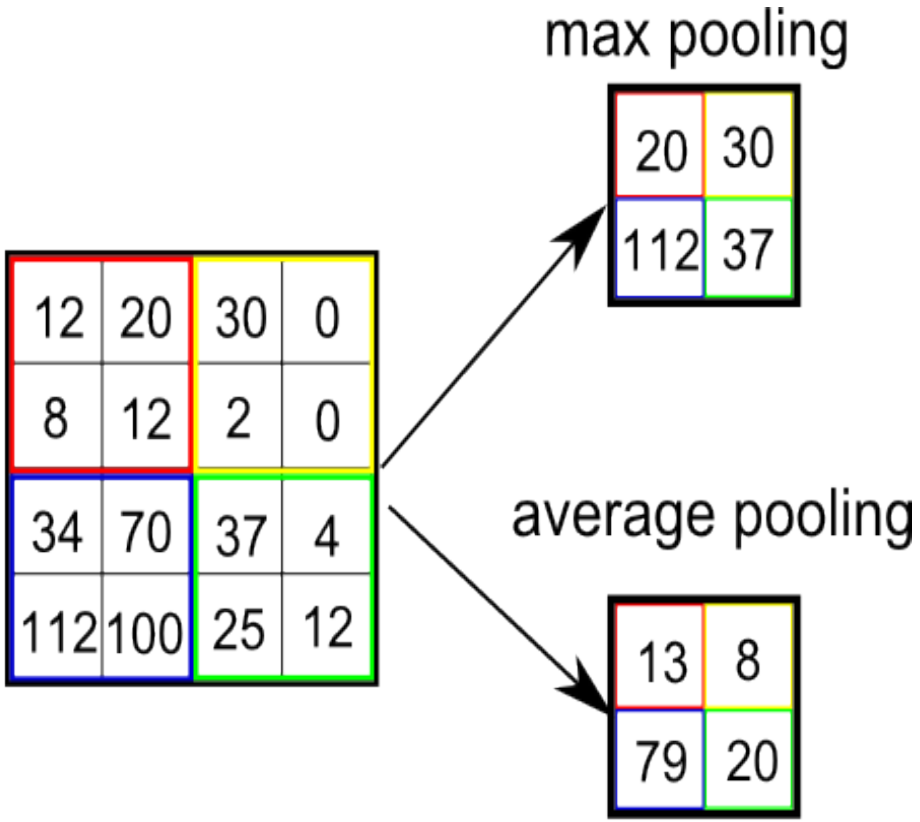
The stride: S

The zero padding used: P

Resulting features map width: **(W−F+2P)/S+1.**

What happens if resulting size is a floating point value?

Pooling layer hyper parameters:



Pooling kernel size

Pooling Stride

Pooling layer: output Feature map size calculation

Accepts a volume of size W1×H1×D1

Requires two hyper parameters:

Spatial extent / Pooling kernel size F

Stride S

Produces a volume of size W2×H2×D2

where:

W2=(W1−F)/S+1

H2=(H1−F)/S+1

D2=D1

What happens if resulting size is a floating point value?

What about training?

Luckily it is,

Stochastic Gradient descent + back propagation

Learning rate, momentum, epochs, batch-size etc. are still the hyper-parameters.

Dropout, Batch normalization are used for regularization.

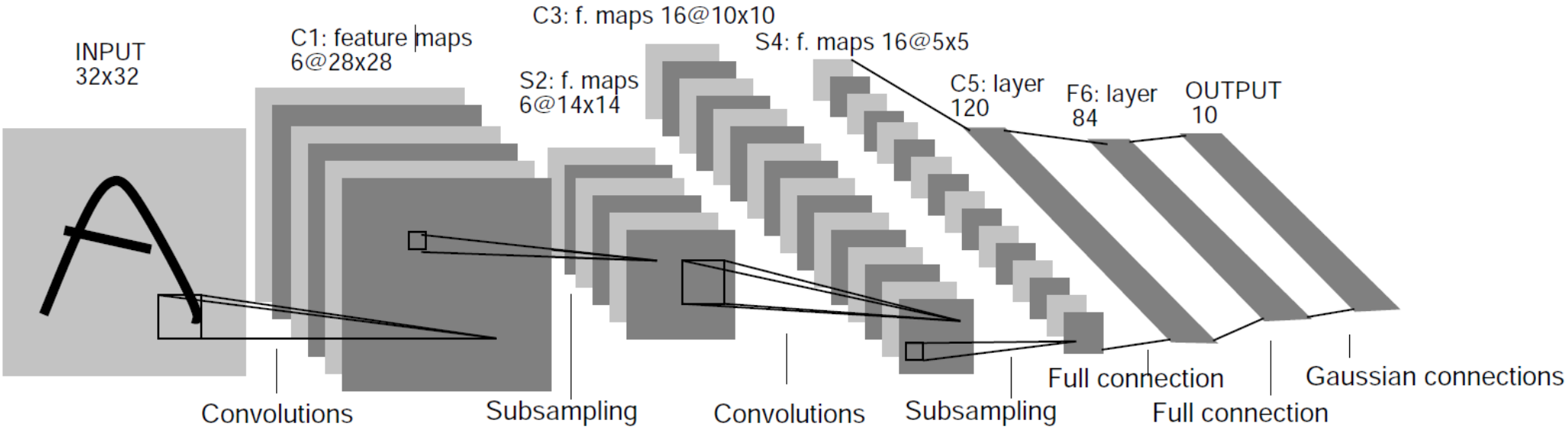
Data augmentation.

We know that in most cases more the data better the model we learn.

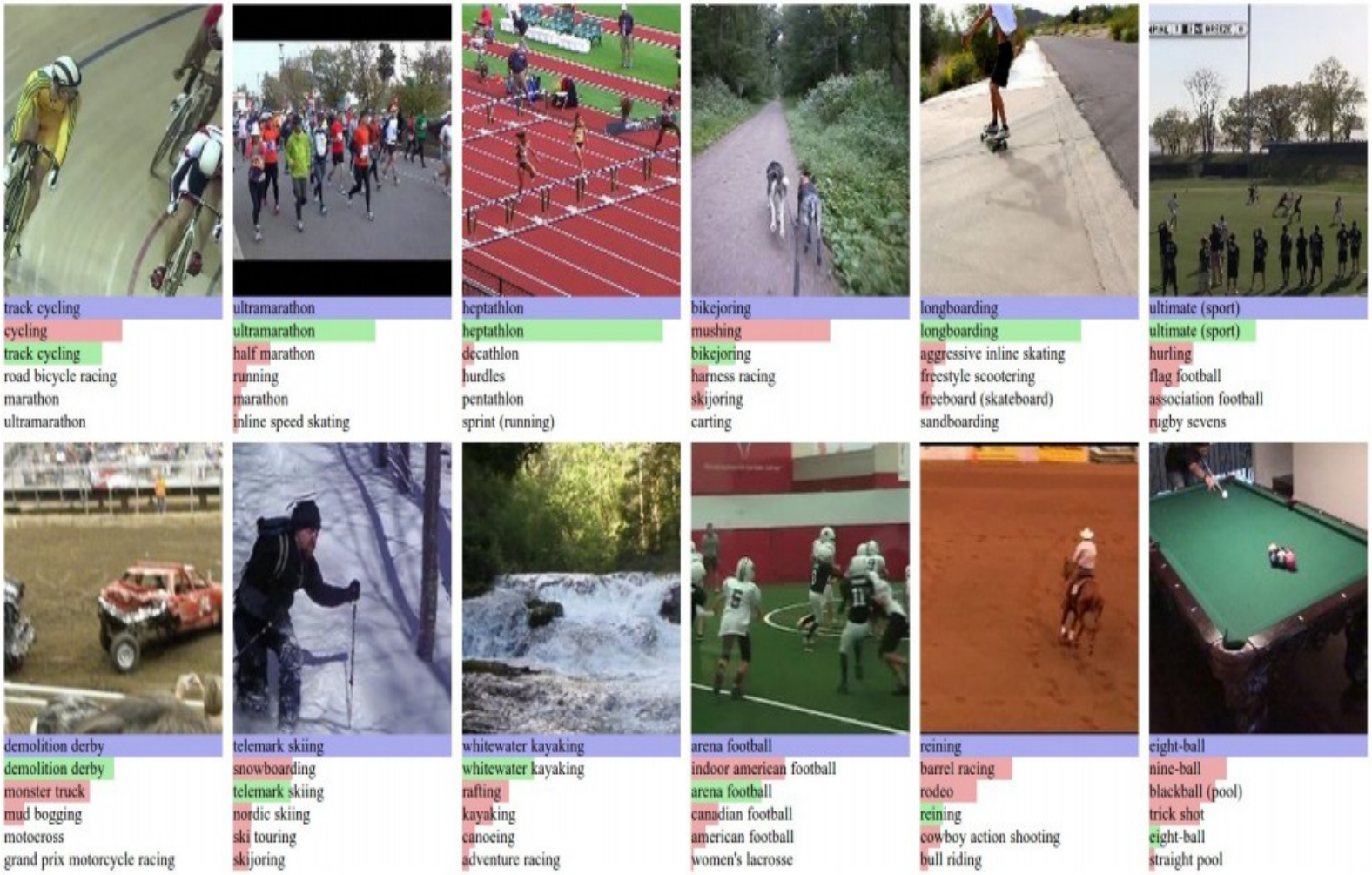
What can done to increase data based on the given dataset, especially for images.

Vertical flipping, small rotations and shifts, color shifts etc. can be used for image data augmentation.

Applications of CNNs: Visual image recognition

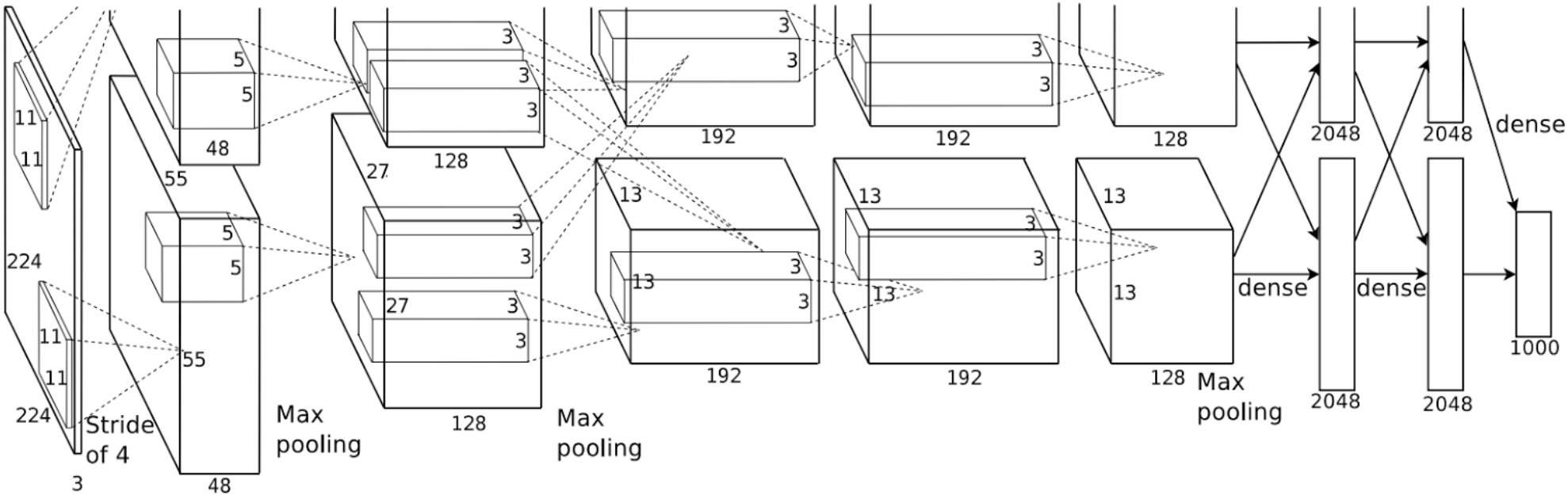


Applications of CNNs: Visual image recognition



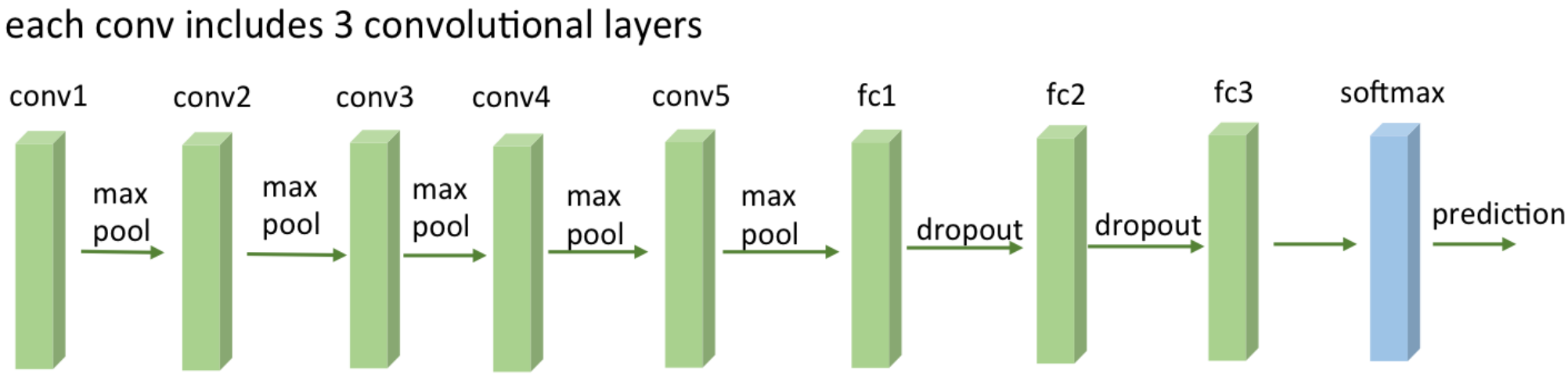
ImageNet challenge: 1000 classes, 1 million training images. [Krizhevsky.2012]

Applications of CNNs: Visual image recognition



ImageNet challenge: 1000 classes, 1 million training images.
[Krizhevsky.2012]

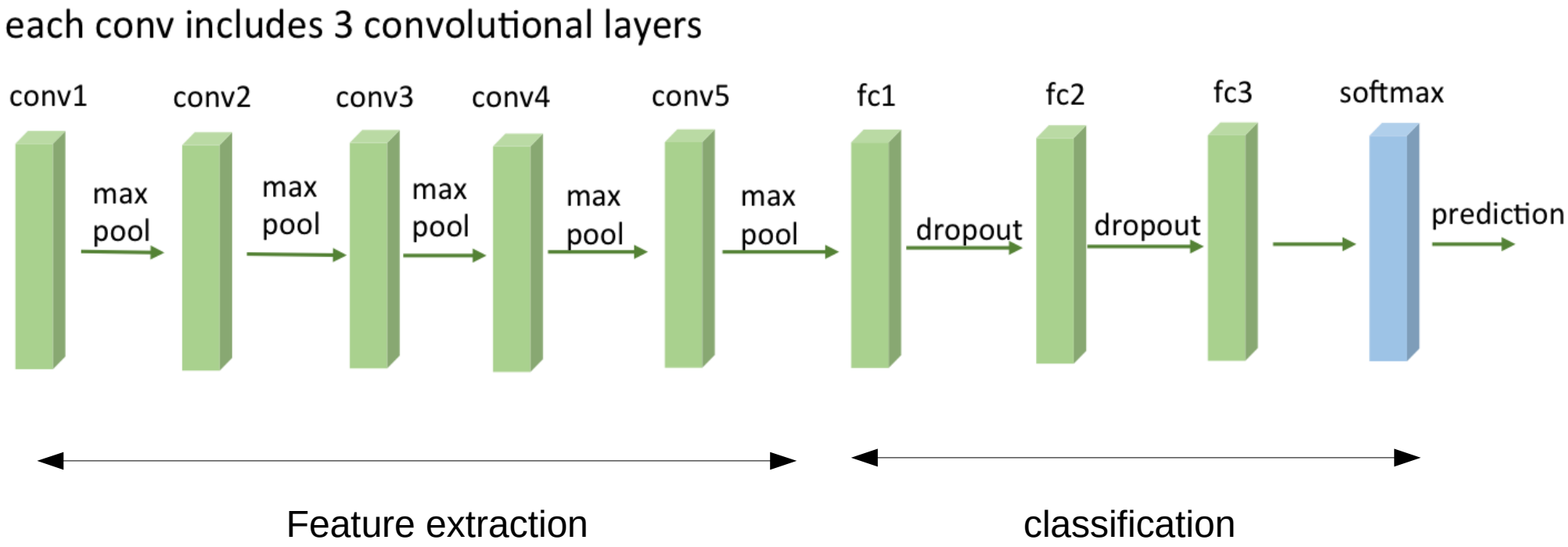
Applications of CNNs: Visual image recognition



ImageNet-VGG-16-layer network

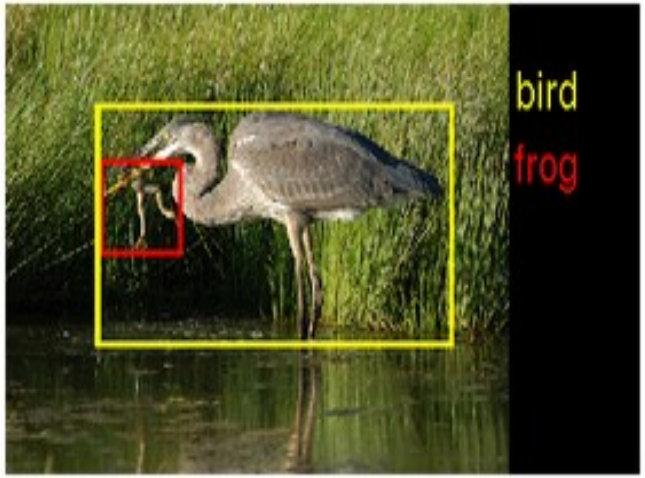
https://arxiv.org/pdf/1409.1556

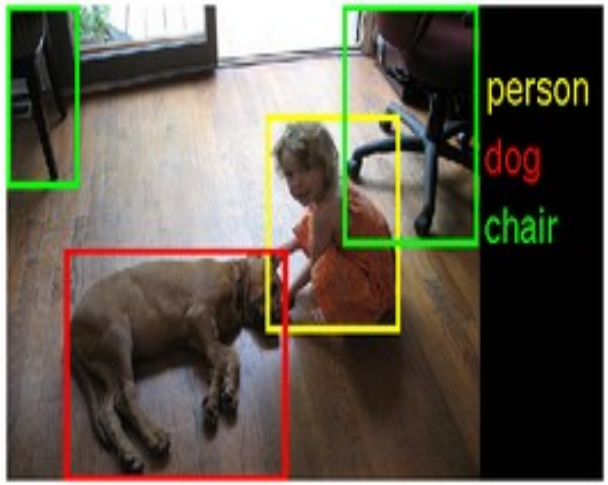
Applications of CNNs: knowledge transfer



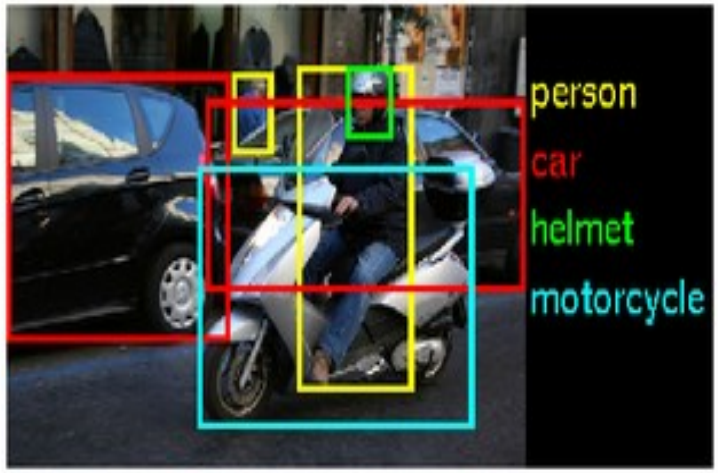
1. Train CNN on large dataset like ImageNet
2. Re-initialize only the classifier part
3. Train the classifier part or the whole network on new smaller dataset

Applications of CNNs: Image Segmentation





FasterRCNN



Applications of CNNs: Image annotation

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.	 A dog is jumping to catch a frisbee.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.	 A refrigerator filled with lots of food and drinks.
 A herd of elephants walking across a dry grass field.	 A close up of a cat laying on a couch.	 A red motorcorcycle parked on the side of the road.	 A yellow school bus parked in a parking lot.

--http://googleresearch.blogspot.in

Applications of CNNs: Video classification

Large-scale Video Classification with Convolutional Neural Networks,
--Andrej Karpathy
(Google/Stanford)

Applications of CNNs: Natural language processing (NLP)

- semantic parsing
- search query retrieval
- sentence modeling
- classification
- prediction

Applications of CNNs: Text classification

Words and sentences are of varying length, is this a problem for using them as input to a machine learning algorithm?

Images have pixel intensities which can act as direct inputs to a neural network.

While text needs to be encoded into a vector form.

Popular methods for generating word embeddings:

Word2Vec - Mikolov, Tomas; et al. "Efficient Estimation of Word Representations in Vector Space

GloVe - http://nlp.stanford.edu/projects/glove/

Applications of CNNs: Text classification

Word2Vec and GloVe

- Unsupervised learning neural network based algorithm for obtaining vector representations for words.
- Trained on large corpus of text data.
- representations showcase interesting linear substructures of the word vector space.
- Generates a fixed length vector embedding for each word.

If the length of a given sentence is s, then the dimensionality of the sentence matrix is s×d (where d is the word2vec dimensions).

The parameter d can be in range of 100 to 1000, typically. This is decided when training the unsupervised models (*Word2Vec or GloVe*).

Word2Vec: Skip-Gram Model

- A single hidden layer neural network is trained to perform a fake task.
- After training the fake task is dumped and only hidden weights are used.

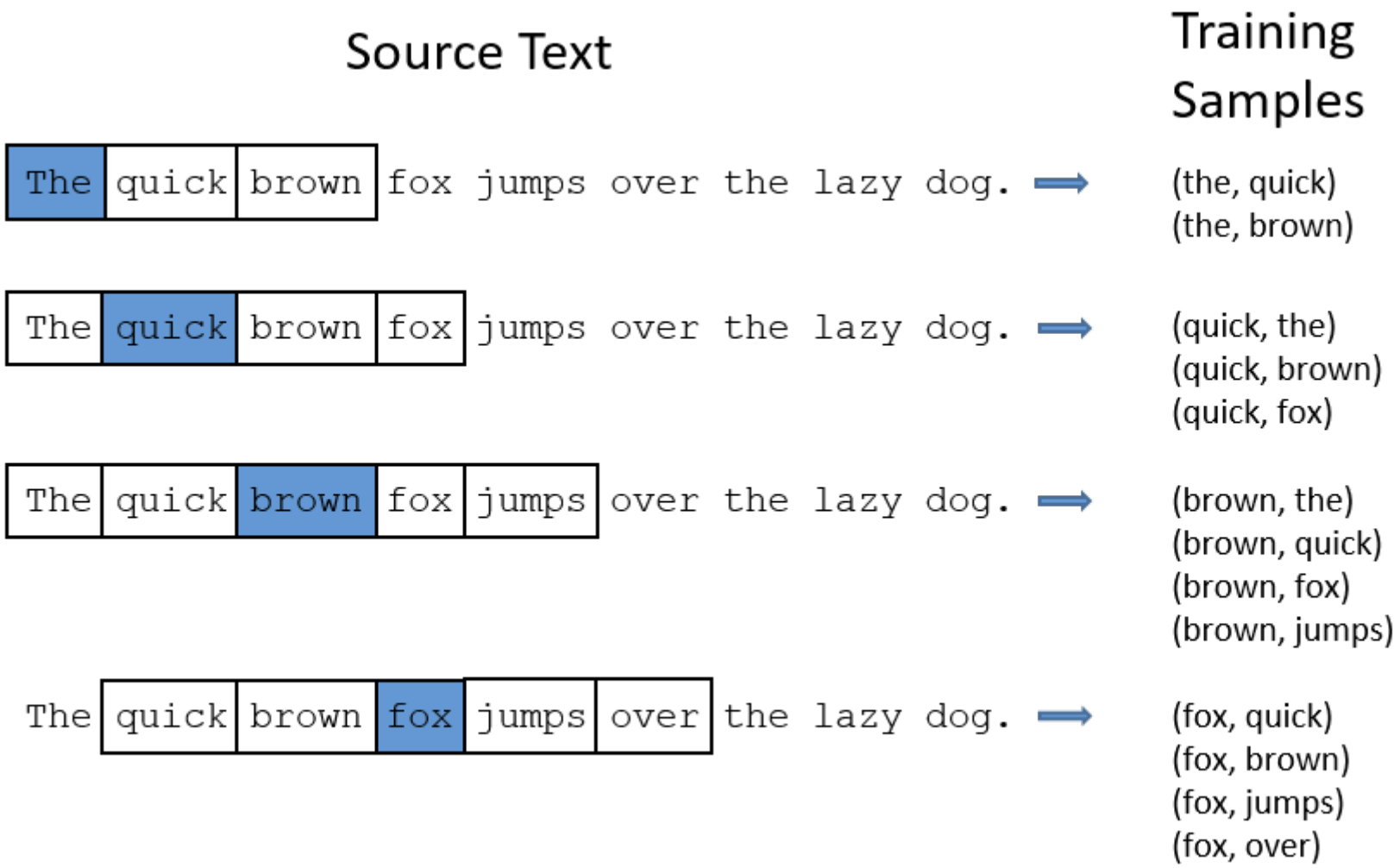
Fake Task: Given a sentence,

"The quick brown fox jumps over the lazy dog."

Predict the probabilities of different words from vocabulary occurring in a fixed window size around the chosen word.

http://mccormickml.com

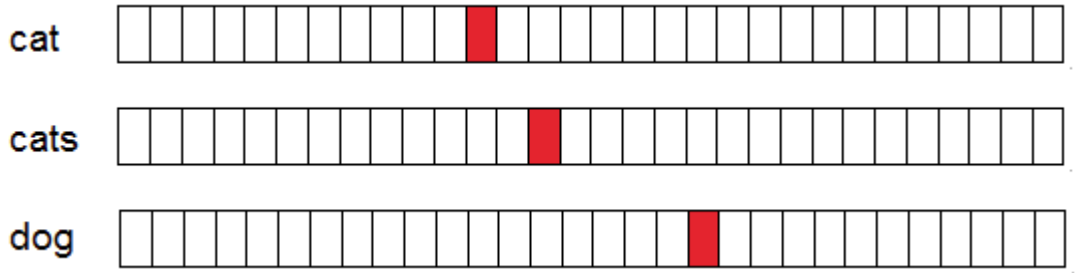
Word2Vec: Skip-Gram Model



http://mccormickml.com

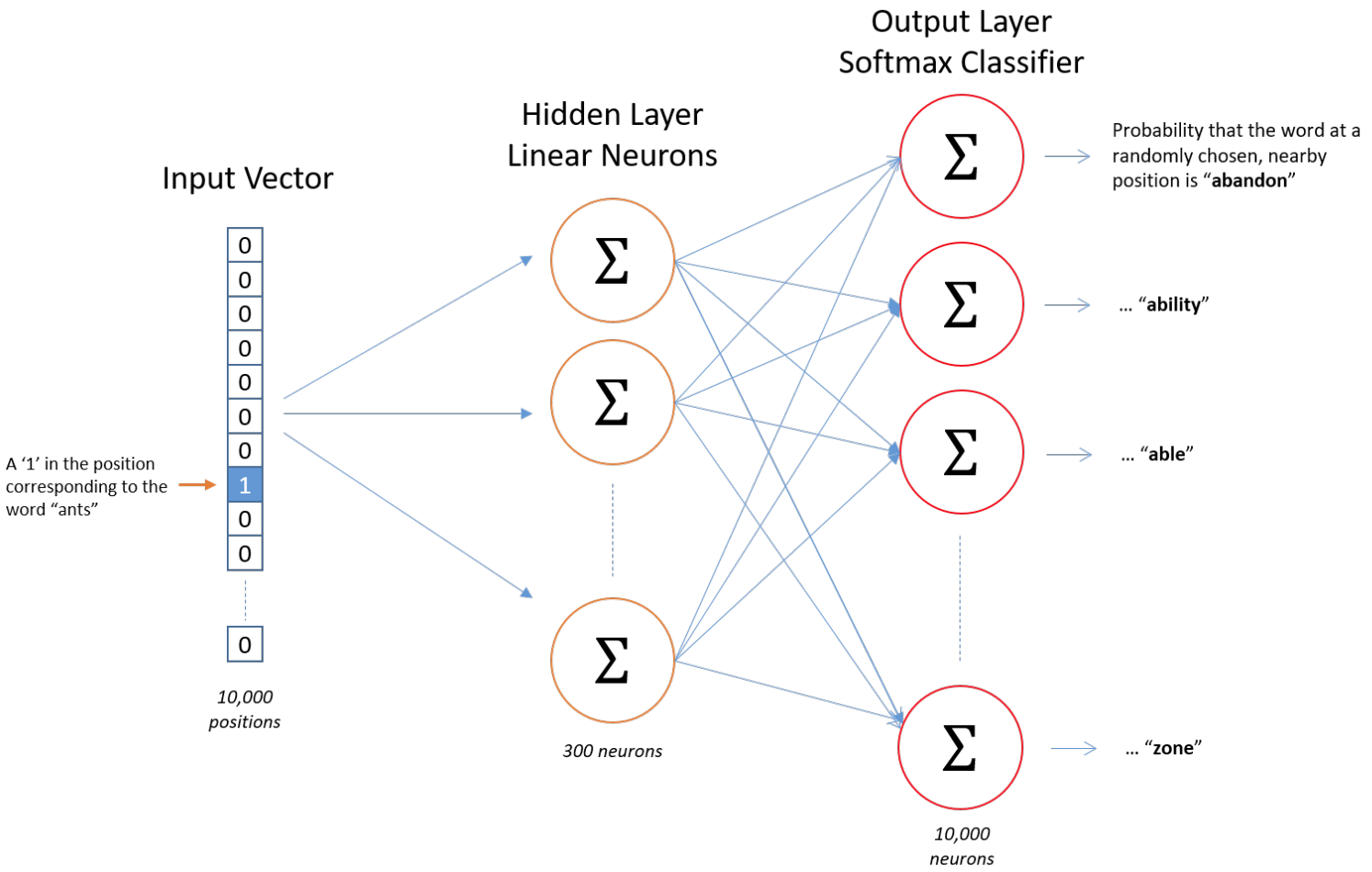
One hot encoding of a word:

- We need a numerical representation for each word to train our skip-gram model.
- If you have a vocabulary of 10000 words treat each word as a state of categorical variable and dummify it.



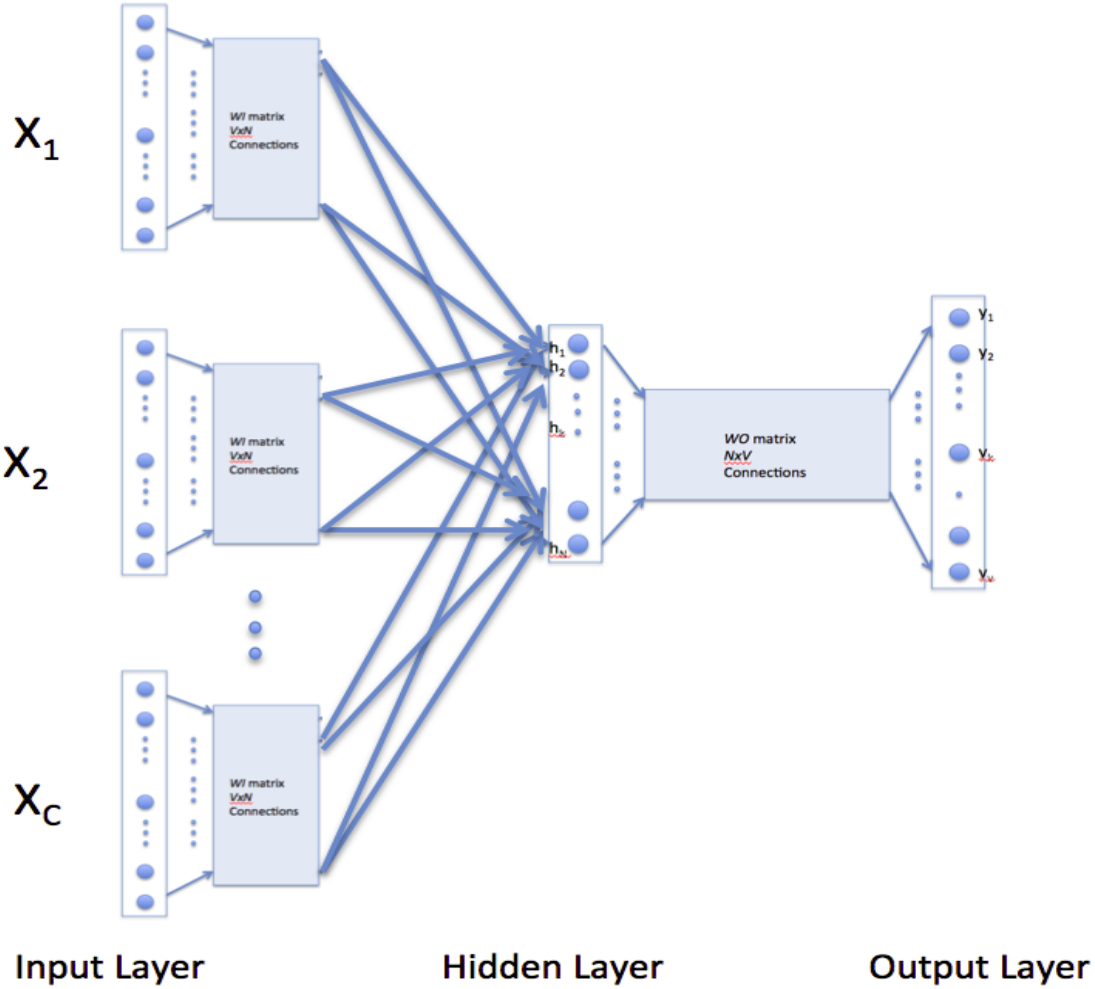
http://mccormickml.com

Word2Vec: Skip-Gram Model



http://mccormickml.com

Word2Vec: CBOW



Training sample:

Given Sentence:

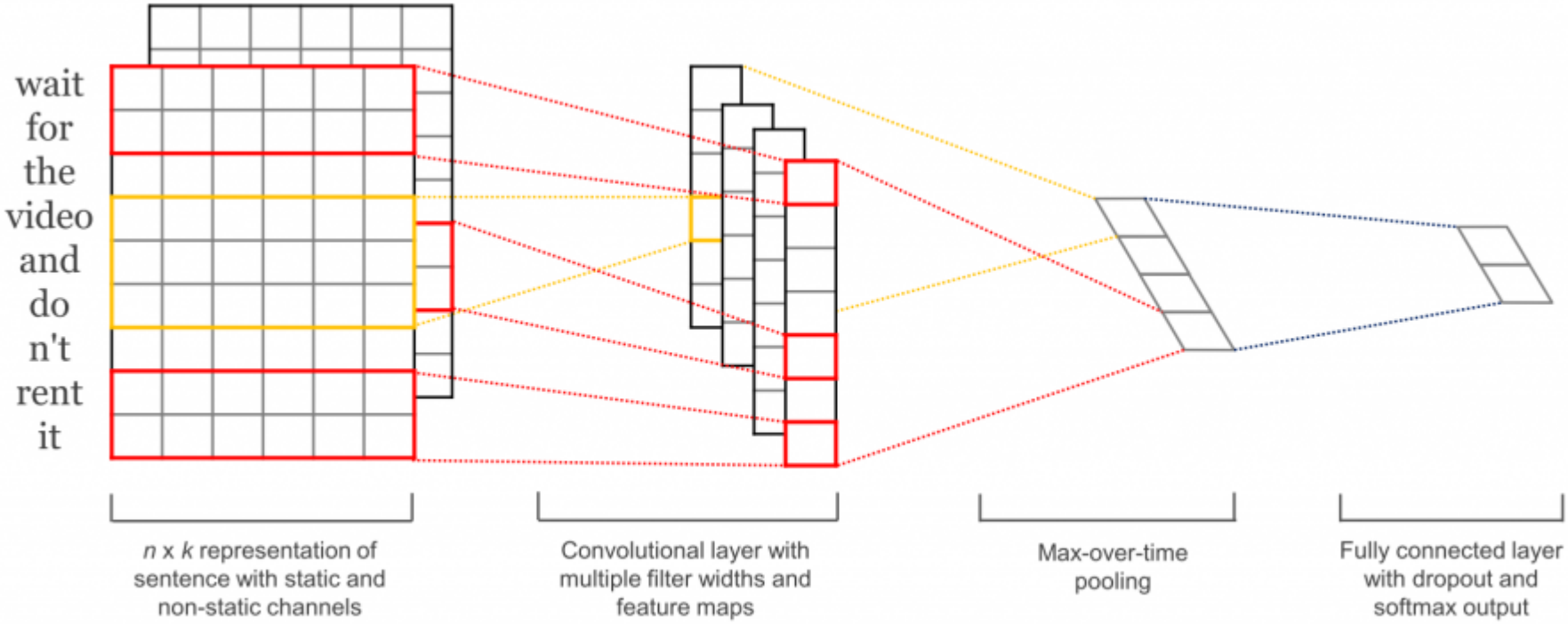
"The quick brown fox jumps over the lazy dog"

(quick,brown,jumps:fox)

(jumps,the,dog: lazy)

http://mccormickml.com

Applications of CNNs: Text classification



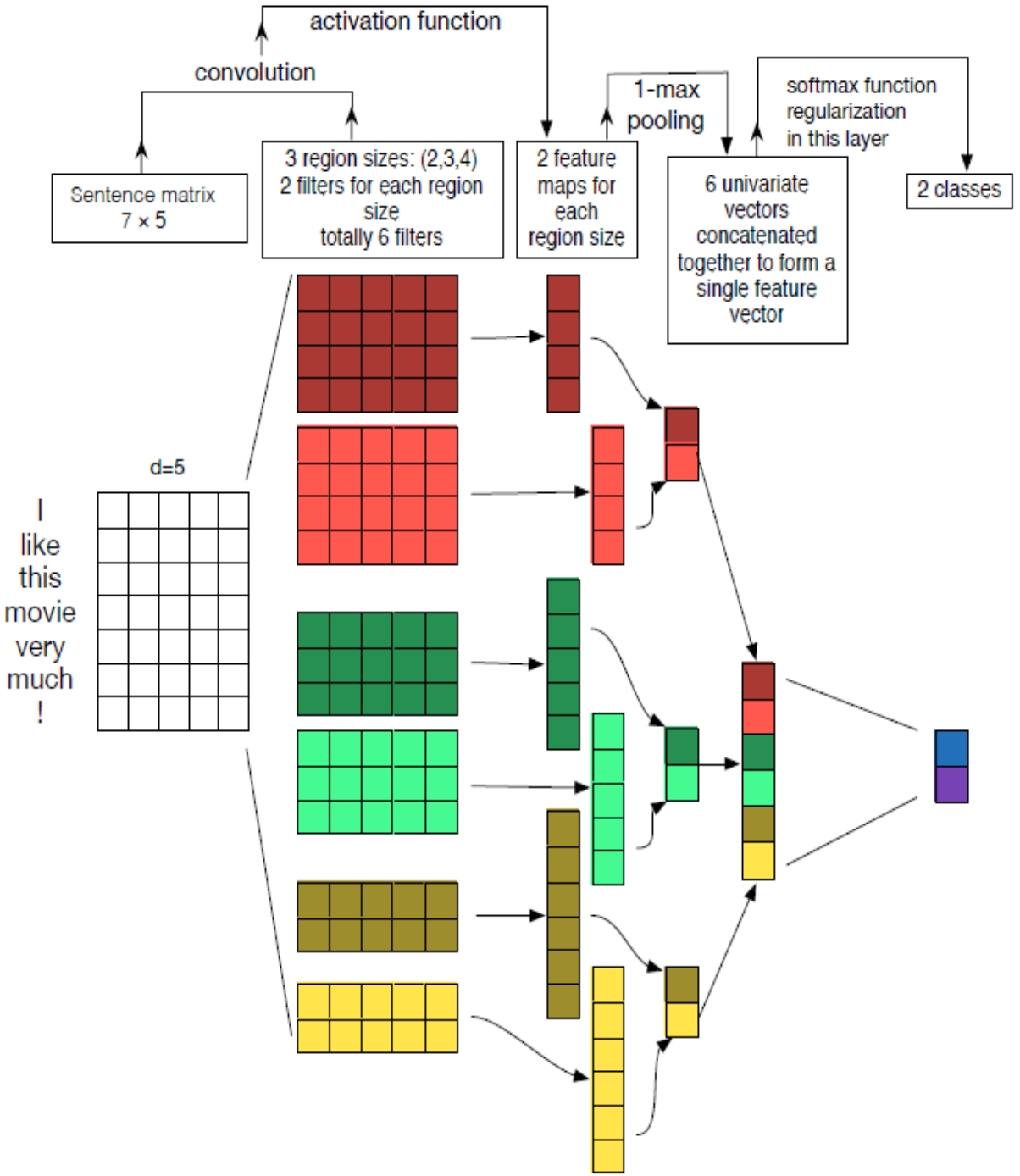

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Illustration of a CNN architecture for sentence classification.

Three filter region sizes: 2, 3 and 4, each of which has 2 filters.

Filters perform convolutions on the sentence matrix and generate (variable-length) feature maps;

1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded.

Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer.

The final softmax layer then receives this feature vector as input and uses it to classify the sentence;

Here binary classification is assumed and hence depict two possible output states.

Applications of CNNs: Text classification

Word embeddings are generated from pre-trained unsupervised models like Word2Vec.

In most cases the parameters of Word2Vec model don't change while training the CNN model for classification. This case is termed as ***Static-CNN***

Given the models like Word2Vec are also neural network based can be we fine tune the pre-trained models while training CNN on a specific task?

Yes, we can such a case where a Word2Vec (or others) is also fine-tuned while training the CNN on a task is termed,
Non-static CNN

Dataset	Non-static word2vec-CNN	Non-static GloVe-CNN	Non-static GloVe+word2vec CNN
MR	81.24 (80.69, 81.56)	81.03 (80.68,81.48)	81.02 (80.75,81.32)
SST-1	47.08 (46.42,48.01)	45.65 (45.09,45.94)	45.98 (45.49,46.65)
SST-2	85.49 (85.03, 85.90)	85.22 (85.04,85.48)	85.45 (85.03,85.82)
Subj	93.20 (92.97, 93.45)	93.64 (93.51,93.77)	93.66 (93.39,93.87)
TREC	91.54 (91.15, 91.92)	90.38 (90.19,90.59)	91.37 (91.13,91.62)
CR	83.92 (82.95, 84.56)	84.33 (84.00,84.67)	84.65 (84.21,84.96)
MPQA	89.32 (88.84, 89.73)	89.57 (89.31,89.78)	89.55 (89.22,89.88)
Opi	64.93 (64.23,65.58)	65.68 (65.29,66.19)	65.65 (65.15,65.98)
Irony	67.07 (65.60,69.00)	67.20 (66.45,67.96)	67.11 (66.66,68.50)

Table 3: Performance using non-static word2vec-CNN, non-static GloVe-CNN, and non-static GloVe+word2vec CNN, respectively. Each cell reports the mean (min, max) of summary performance measures calculated over multiple runs of 10-fold cross-validation. We will use this format for all tables involving replications

Practical guidelines:

- The kernel/filter width is always same as input width (word embedding dimensions)
- Convolutional layer can have multiple kernels with different height/range: Each kernel can span over different number of words in the input
- Most text classification experiments involve one convolution layer followed by a max-pooling layer and then a fully connected softmax layer.
- Have different kernel sizes is computationally very expensive. Choosing a fixed size is faster but some times result in less performance.
- Selection of height/range of kernels mostly depends on the dataset.

Multiple region size	Accuracy (%)
(7)	81.65 (81.45,81.85)
(3,4,5)	81.24 (80.69, 81.56)
(4,5,6)	81.28 (81.07,81.56)
(5,6,7)	81.57 (81.31,81.80)
(7,8,9)	81.69 (81.27,81.93)
(10,11,12)	81.52 (81.27,81.87)
(11,12,13)	81.53 (81.35,81.76)
(3,4,5,6)	81.43 (81.10,81.61)
(6,7,8,9)	81.62 (81.38,81.72)
(7,7,7)	81.63 (81.33,82.08)
(7,7,7,7)	**81.73 (81.33,81.94)**

Table 5: Effect of filter region size with several region sizes on the MR dataset.

Multiple region size	Accuracy (%)
(3)	91.21 (90.88,91.52)
(5)	91.20 (90.96,91.43)
(2,3,4)	91.48 (90.96,91.70)
(3,4,5)	91.56 (91.24,91.81)
(4,5,6)	91.48 (91.17,91.68)
(7,8,9)	90.79 (90.57,91.26)
(14,15,16)	90.23 (89.81,90.51)
(2,3,4,5)	**91.57 (91.25,91.94)**
(3,3,3)	91.42 (91.11,91.65)
(3,3,3,3)	91.32 (90.53,91.55)

The best performing strategy is to simply use many feature maps (here, 400) all with region size equal to 7, i.e., the single best region size. However, we note that in some cases (e.g., for the TREC dataset), using multiple different, but near optimal, region sizes performs best.

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. N: Dataset size. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word vectors. *Test*: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).

http://arxiv.org/pdf/1408.5882v2.pdf

Baseline results from other models

Dataset	bowSVM	wvSVM	bowwvSVM
MR	78.24	78.53	79.67
SST-1	37.92	44.34	43.15
SST-2	80.54	81.97	83.30
Subj	89.13	90.94	91.74
TREC	87.95	83.61	87.33
CR	80.21	80.79	81.31
MPQA	85.38	89.27	89.70
Opi	61.81	62.46	62.25
Irony	65.74	65.58	66.74

Table 1: Accuracy (AUC for Irony) achieved
by SVM with different feature sets.

bowSVM: uni- and bi-gram features.

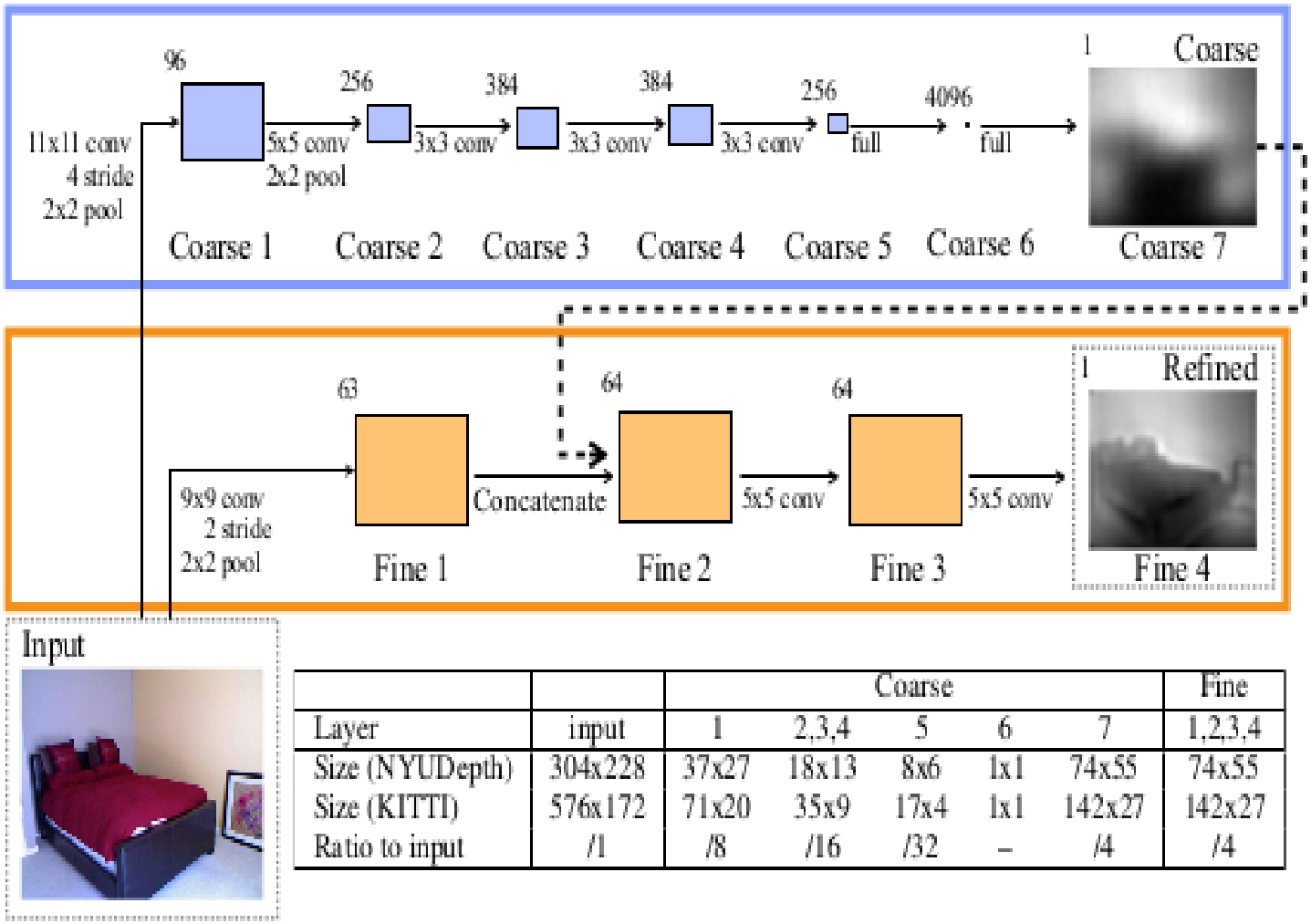
wvSVM: a naïve word2vec-based representation, i.e., the average (300-dimensional) word vector for each sentence.

bowwvSVM: concatenates bow vectors with the
average word2vec representations.

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree with Conditional Random Fields (Nakagawa et al., 2010). **CRF-PR**: Conditional Random Fields with Posterior Regularization (Yang and Cardie, 2014). **SVM_S**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al. (2011)

Applications of CNNs: Depth estimation



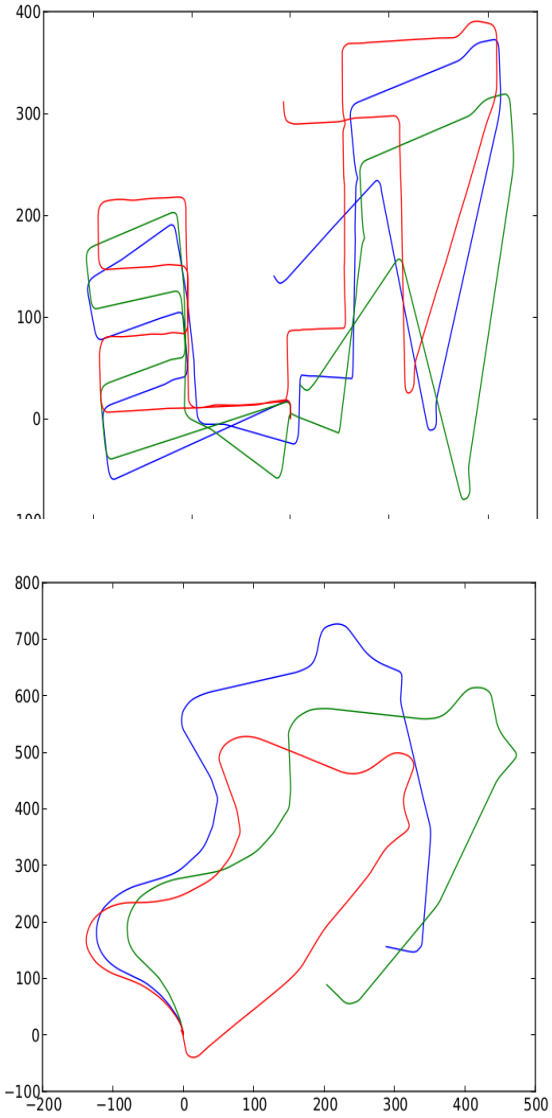
		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, David Eigen

Applications of CNNs: Art

A Neural Algorithm of Artistic Style, Leon A. Gatys

Applications of CNNs: Visual odometry



Konda, Kishore, and Roland Memisevic. "*Learning visual odometry with a convolutional network*"

Applications of CNNs: Atari games

Human-level control through deep reinforcement learning, Google DeepMind

Applications of CNN: Autonomous driving

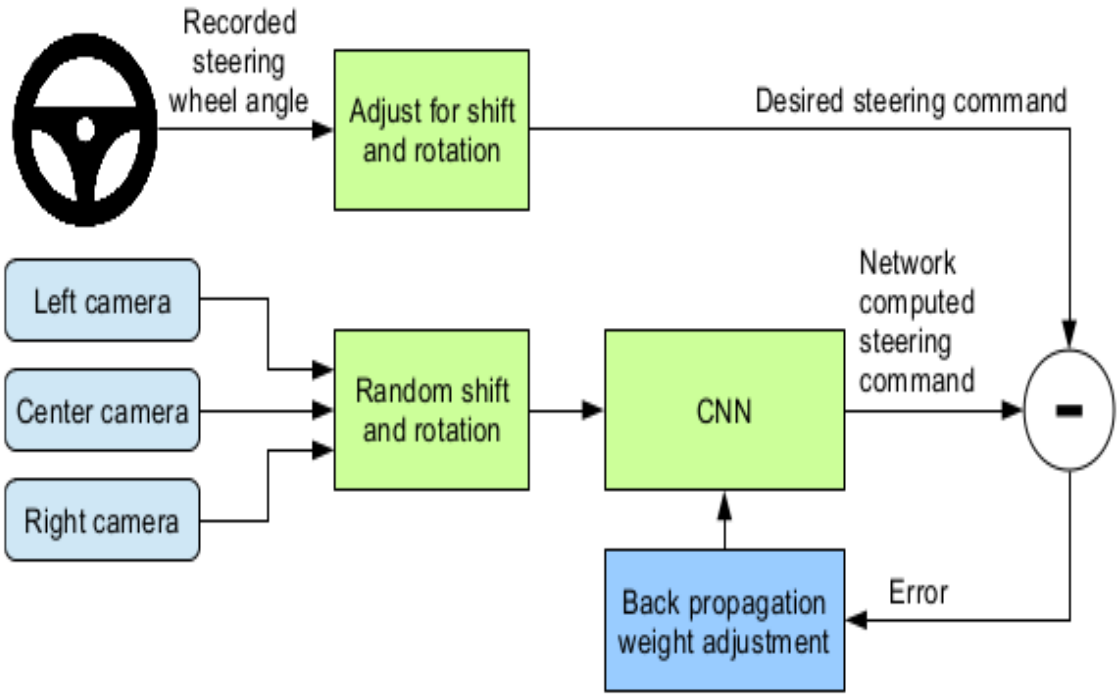


Figure 2: Training the neural network.

Once trained, the network can generate steering from the video images of a single center camera. This configuration is shown in Figure 3.

[NVIDIA Corporation]

Some standard CNN architectures:

- Xception
- VGG Net
- ResNet
- Inception Networks

VGG Net

Network from Oxford's renowned Visual Geometry Group (VGG)

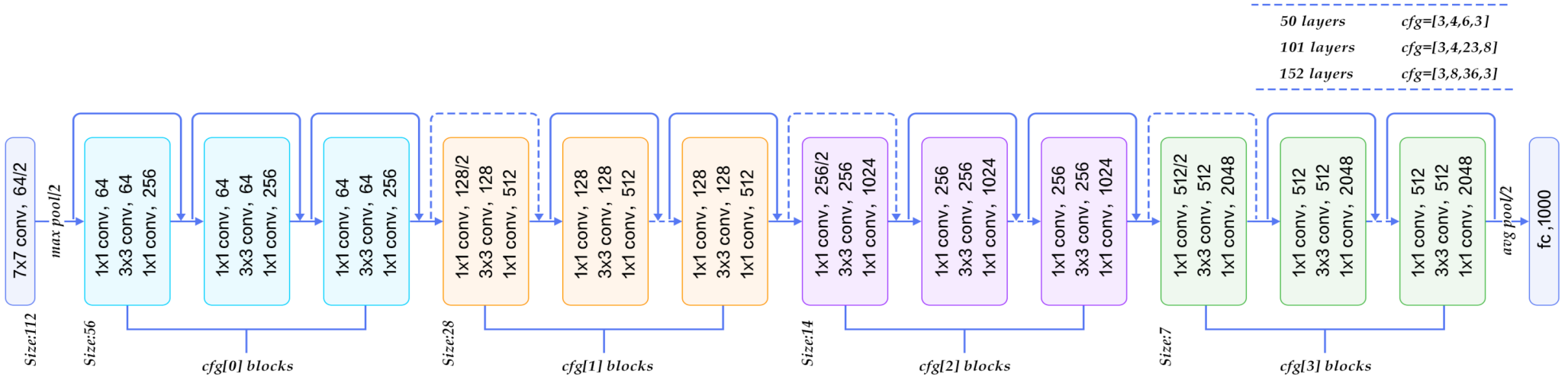
Example: VGG

19 layers
3x3 convolution
pad 1
stride 1

Slideshare.net

Residual Nets

Network from Microsoft



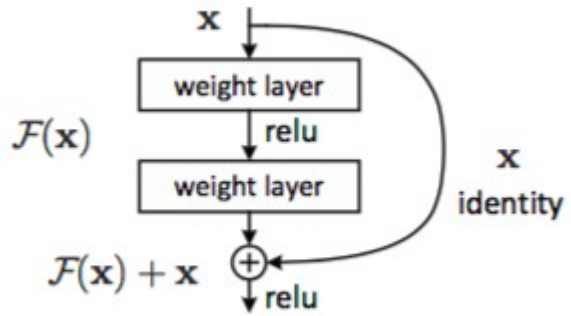
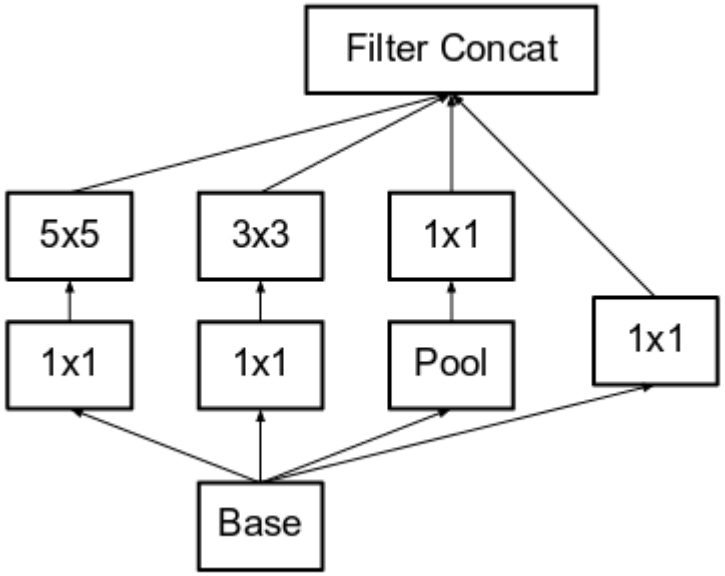


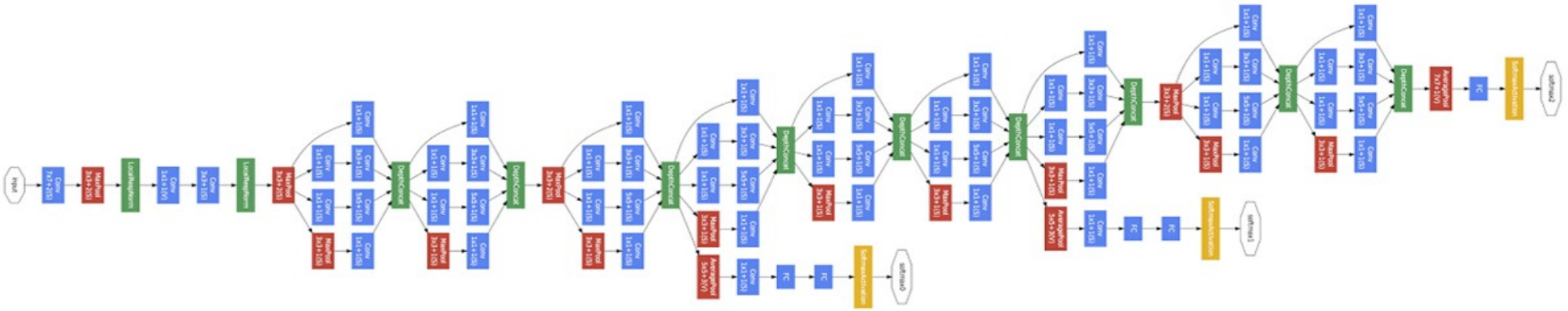
Figure 2. Residual learning: a building block.

Slideshare.net, Quora.com

Inception (GoogLeNet)

Google obviously :)





THANK YOU!

Carry your bag to Super Market!!!

Save Planet!!!