Text Mining

Summer term 2025

Sandipan Sikdar

Leibniz Universität Hannover

Word embeddings

“Complex Data”

labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren.

“Representation”

1.5
0.2
24
16
2.9

Machine learning algorithm

Prediction, recommendation, ….

Word2vec, GLove

CNNs, RNNs, …

Text Classification

- Supervised learning task
- Given a text, map it to a class
- Example: Sentiment classification

The staff were extremely helpful and friendly. Positioning is great, close to bus/train station and within walking distance to all main sights. Had view over square and room was very clean and comfortable.

Was expensive but perhaps this is simply Venice.

Text Classification

- Train data: $\{(x^i, y^i)\}_{i=1}^{N}$

$y^i \in \{0,1\}$

x^i

😃 The staff were extremely helpful and friendly. Positioning is great, close to bus/train station and within walking distance to all main sights. Had view over square and room was very clean and comfortable.

- Loss: Binary cross-entropy loss

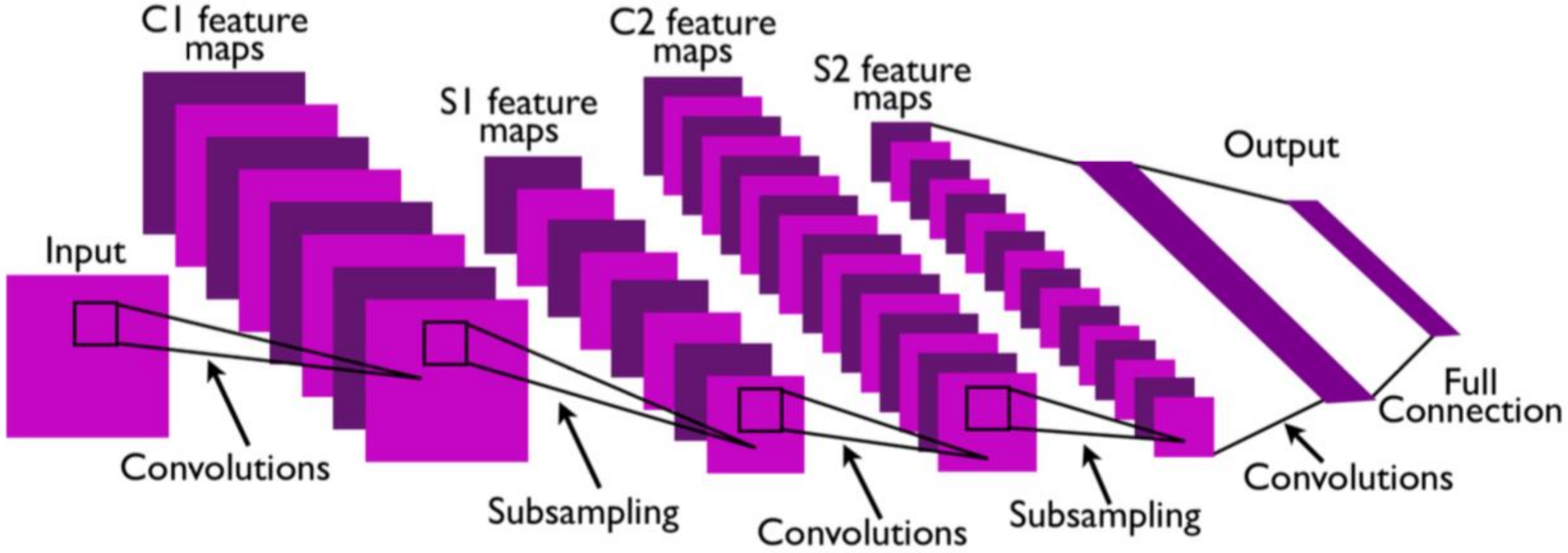
$$J(\theta) = -\sum_{i=1}^{N} y^i \log\left(f_\theta\left(x^i\right)\right) + \left(1 - y^i\right)\log(1 - f_\theta(x^i))$$

Prediction model with parameters θ

Convolutional neural networks

- … are a type of partially connected “ feedforward networks” (details later)
- … originated from computer vision
- … are often used for image classification
- … outperform classical image recognition by a large margin
- … model features over areas of growing size in images
    - Early layers model local neighbourhoods detect edges ,
    - Later layers combine the features to detect larger objects

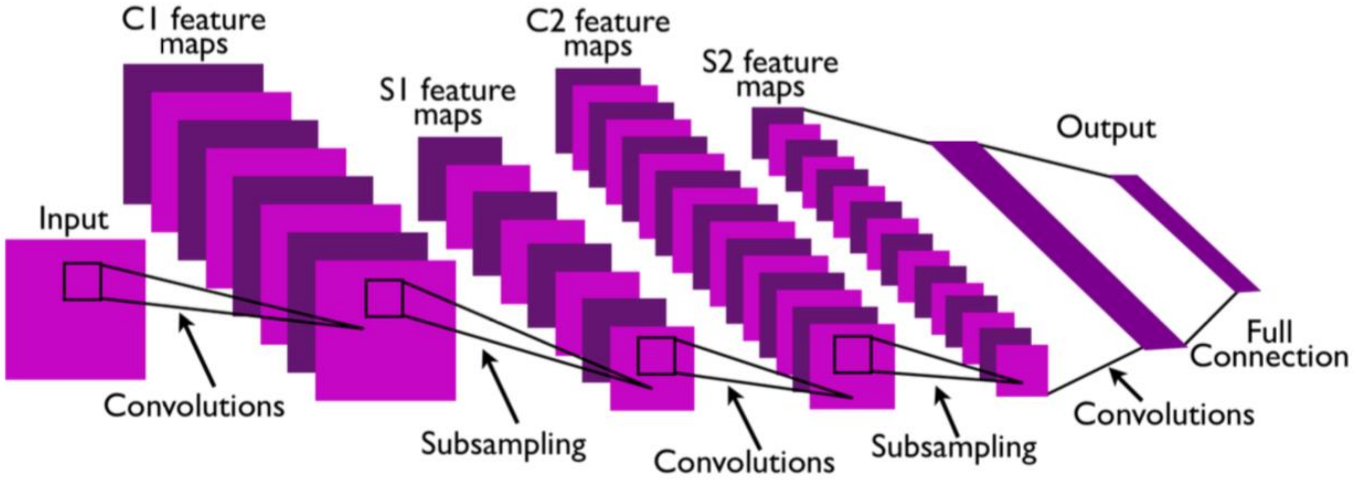
Convolutional neural networks



https://wiki.tum.de/display/lfdv/Convolutional+Neural+Networks

CNNs in Computer Vision

- Typical structure of CNN
    1. Convolutional layer
    2. Pooling layer
    3. Repeat 1 and 2 multiple times
    4. Fully connected layer
    5. Softmax



CNN: A bit of history

A bit of history:

Hubel & Wiesel,

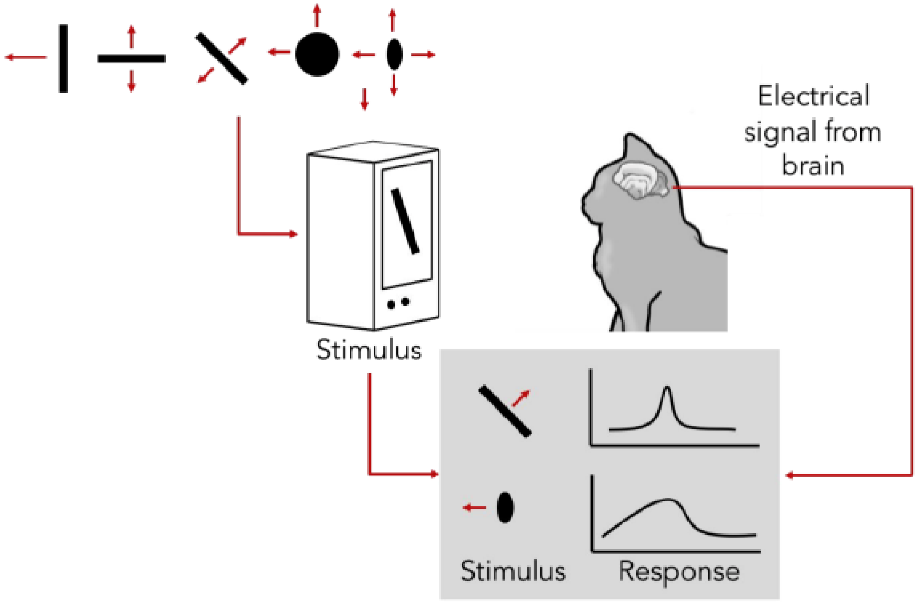
1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...



CNN: A bit of history

Hierarchical organization

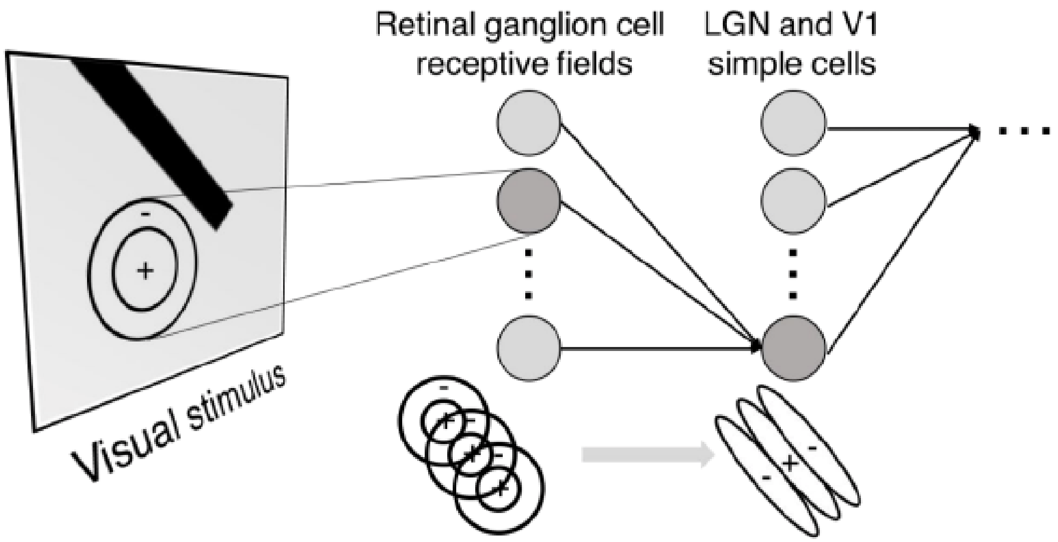
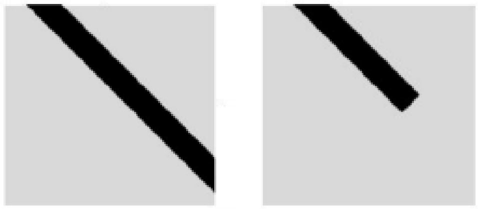


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



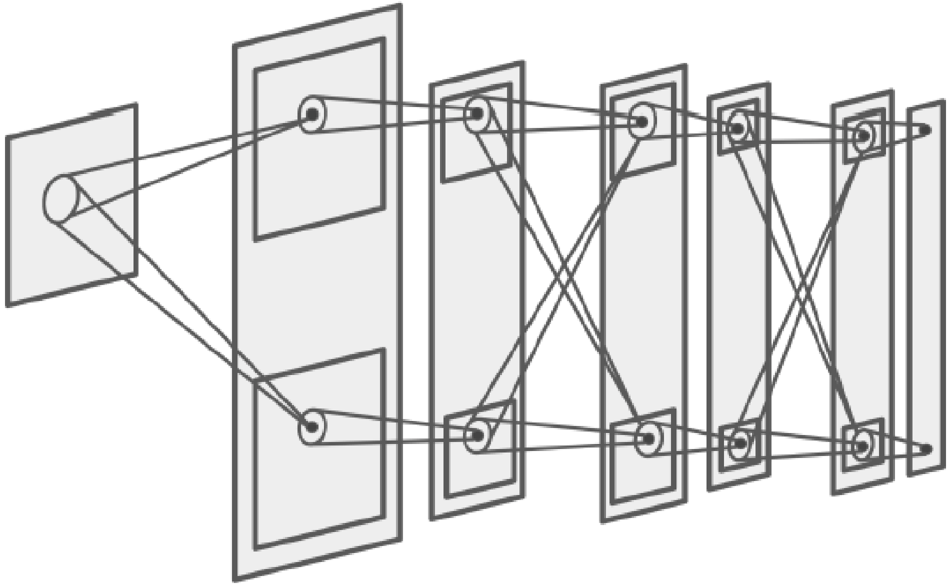
No response

Response (end point)

CNN: A bit of history

A bit of history:

Neocognitron
[Fukushima 1980]



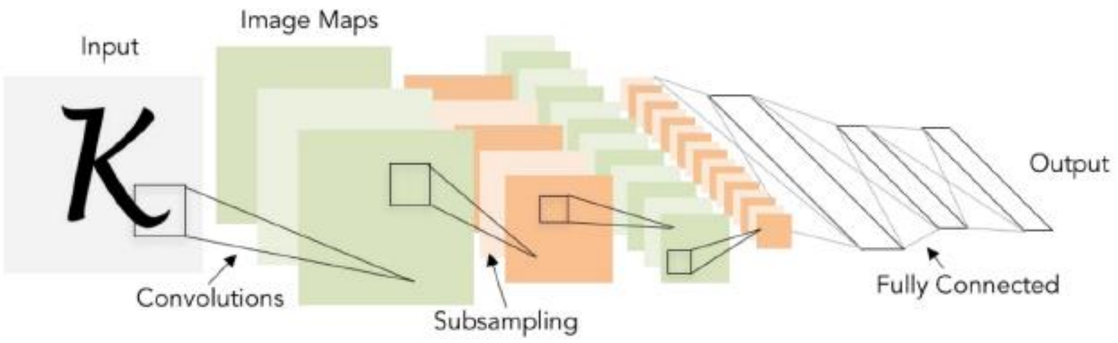
"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

CNN: A bit of history

"LeNet 1", the first convolutional network that could recognize handwritten digits with good speed and accuracy (1989)

CNN: A bit of history

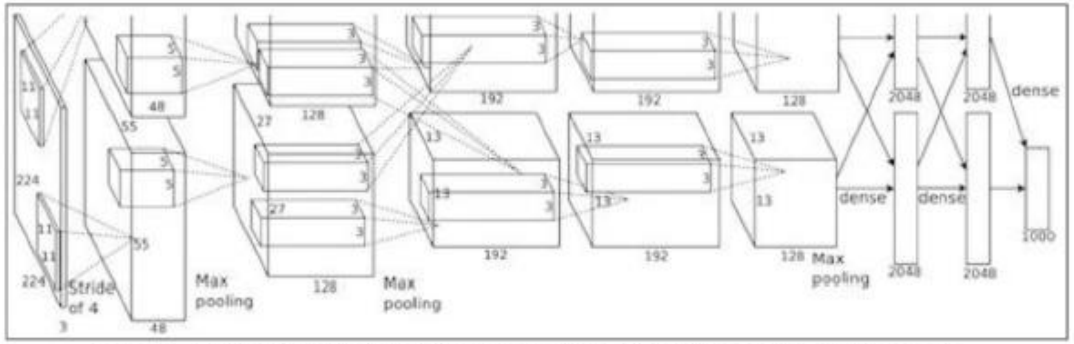
Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

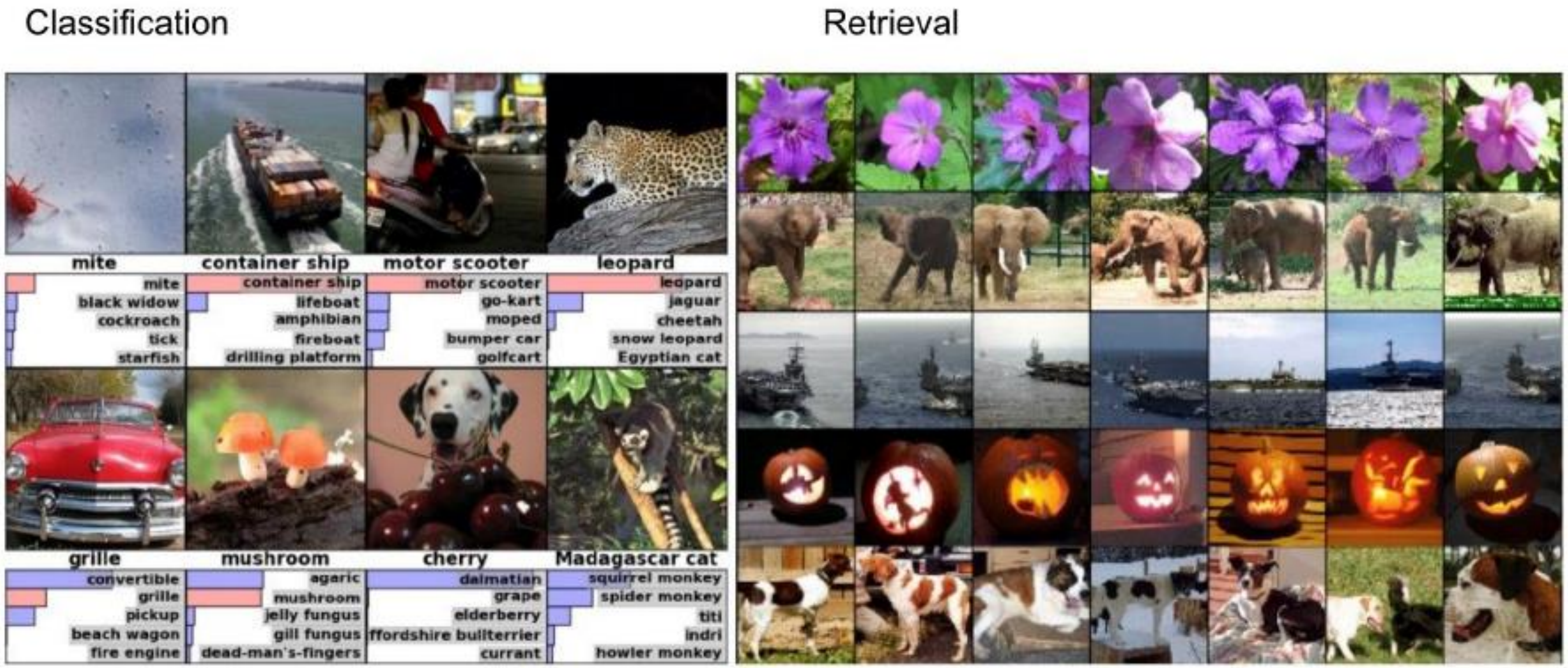
CNN: A bit of history

ImageNet Classification with Deep Convolutional Neural Networks
[Krizhevsky, Sutskever, Hinton, 2012]



"AlexNet"

CNNs are everywhere



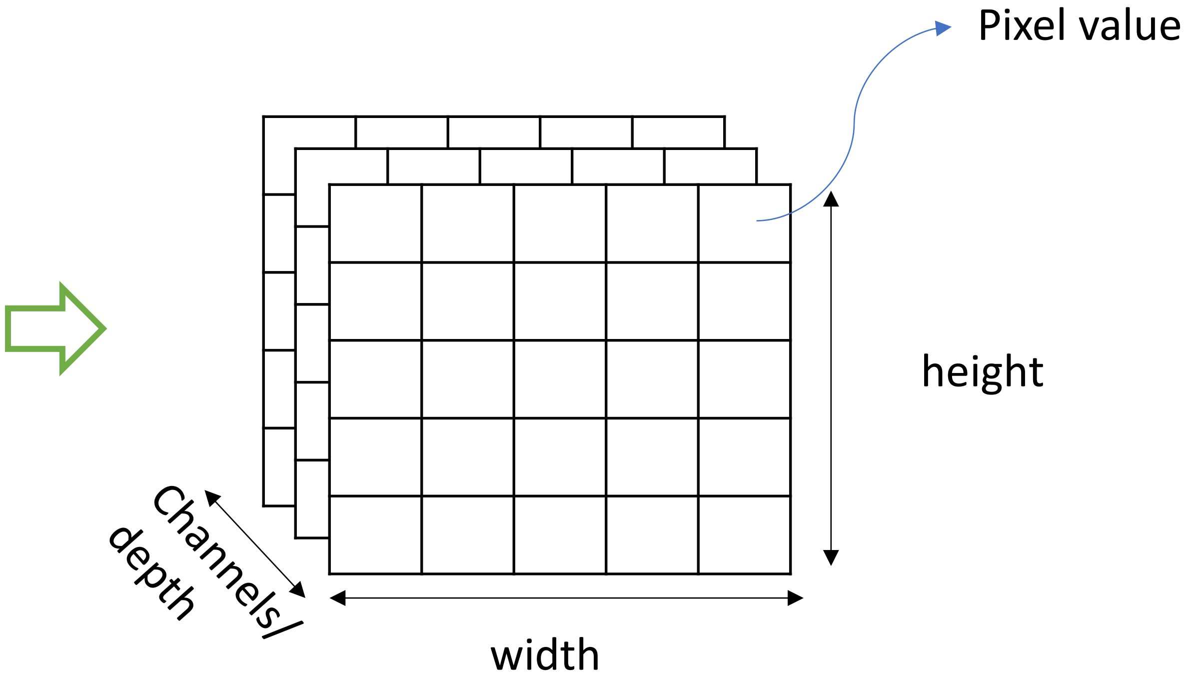
Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012.

Convolutions

- To understand CNNs, we take a look at their basic building blocks Convolutions
- Convolution = classic tool in image processing even without machine learning
- Intuition: Slide a function over an image , create a modified image
- Can model operations like edge detection , blurring , sharpening

Convolutions

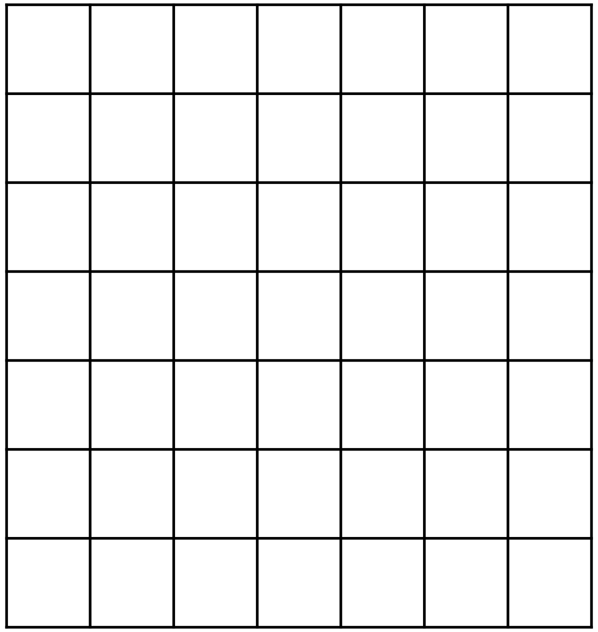
- Representing image



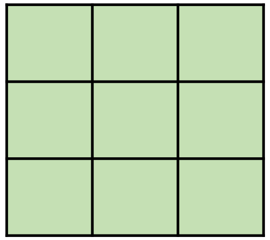
Convolutions

We consider a single channel image

Image

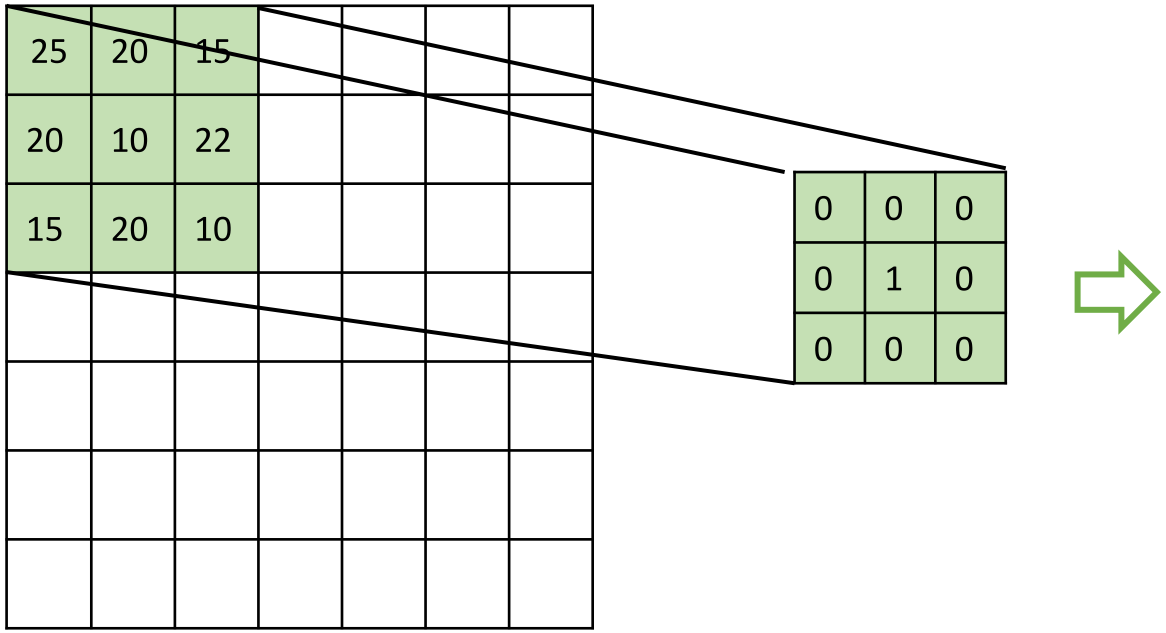


Filter



- Input image (matrix)
- Kernel/filter (matrix)
- Convolve the filter over the image i.e., slide over the image spatially computing dot products

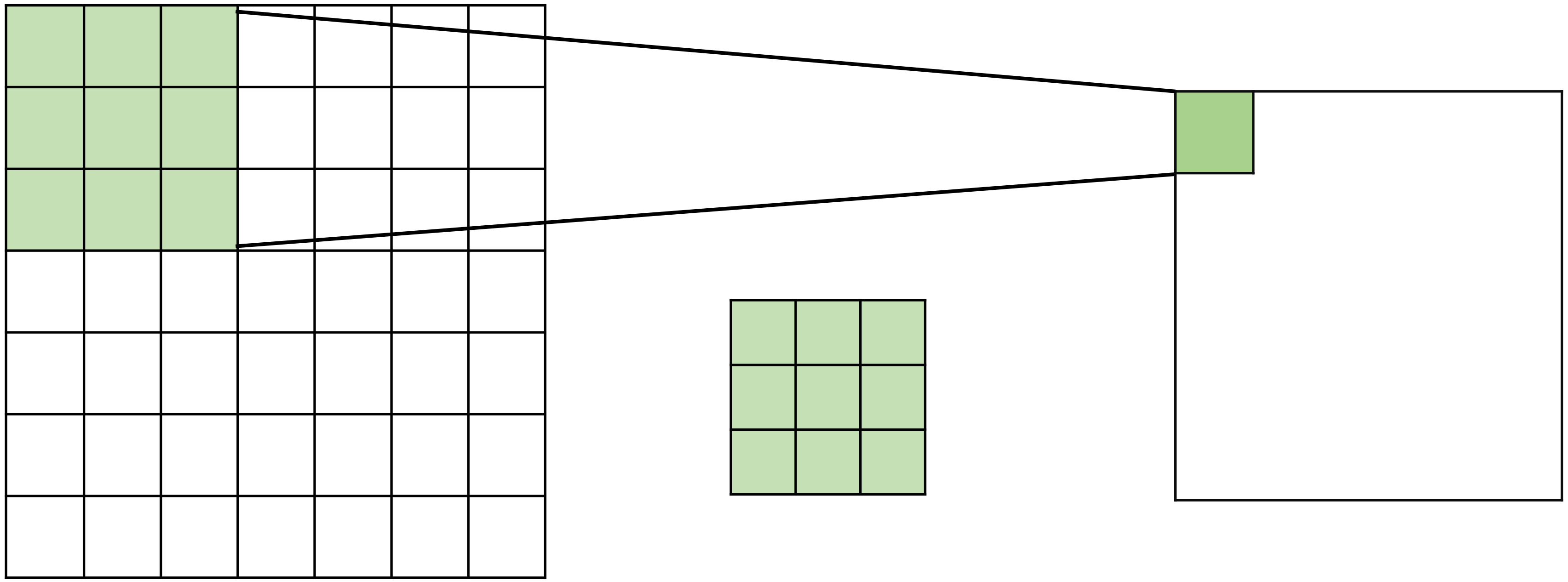
Convolutions



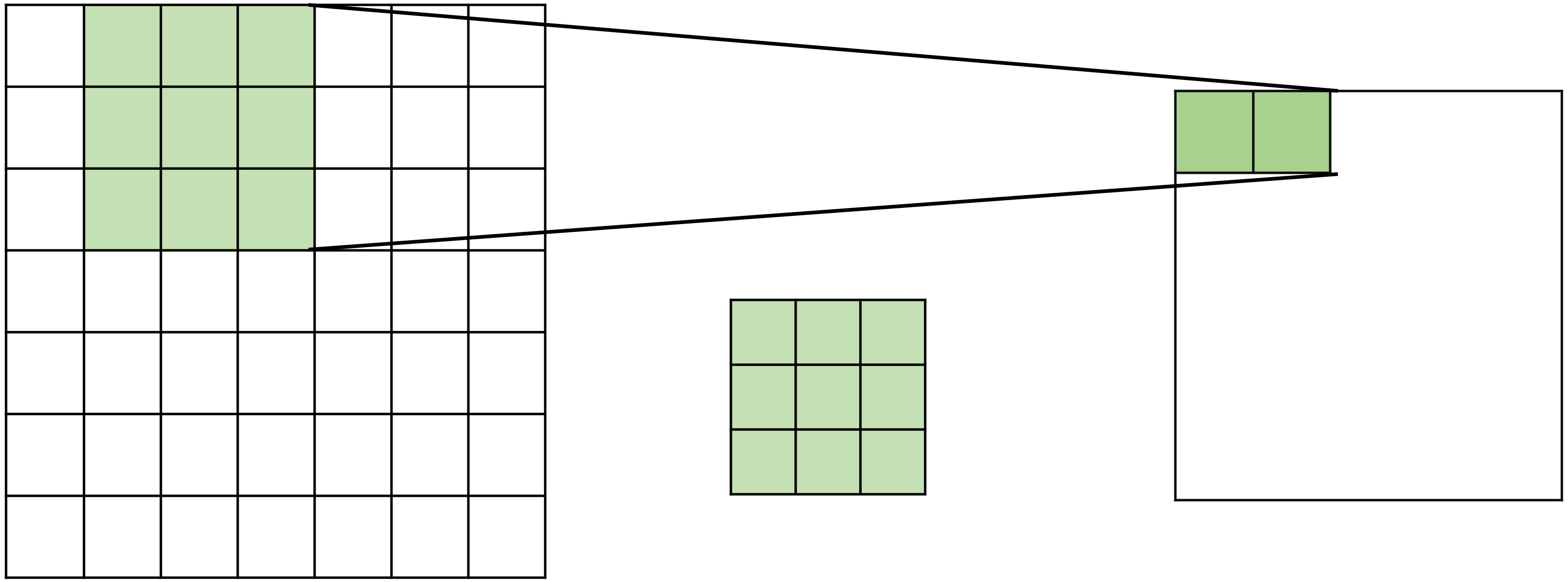
$$out = w^T x + b$$

$25 * 0 + 20 * 0 + 15 * 0 + 20 * 0 + 10 * 1 + 22 * 0 + 15 * 0 + 20 * 0 + 10 * 0 + 0 = 10$
(assuming $b = 0$)

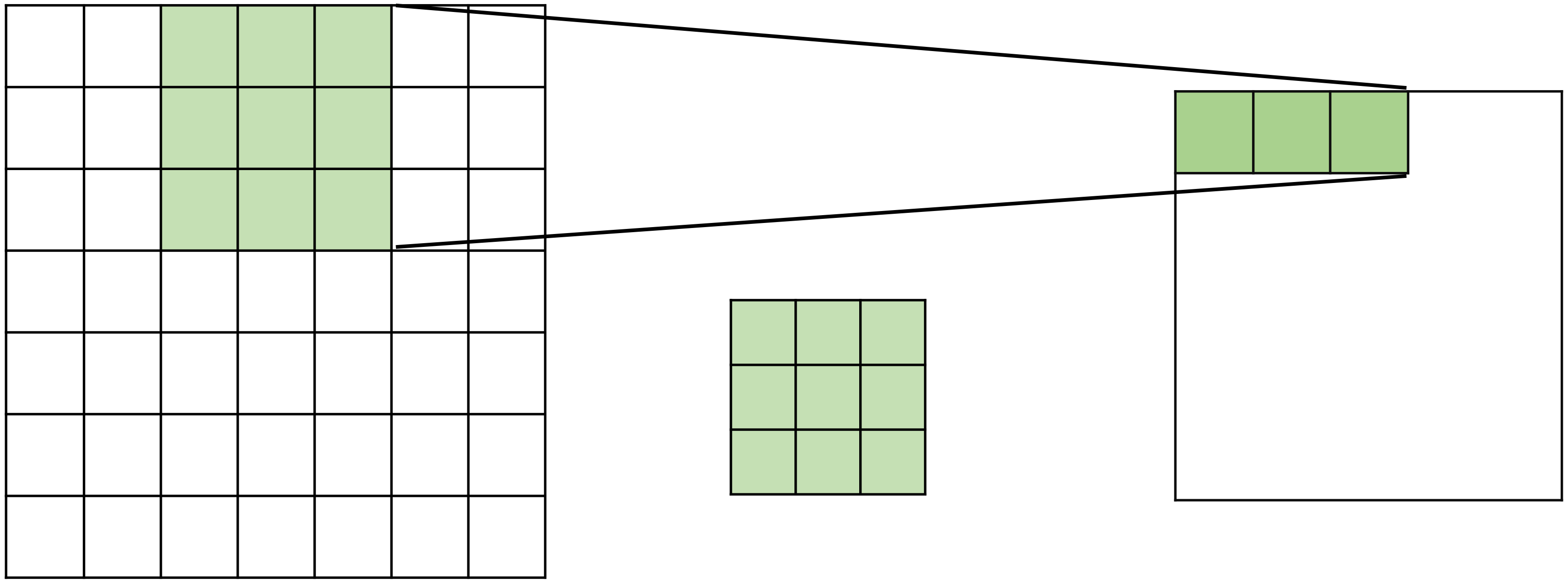
Convolutions



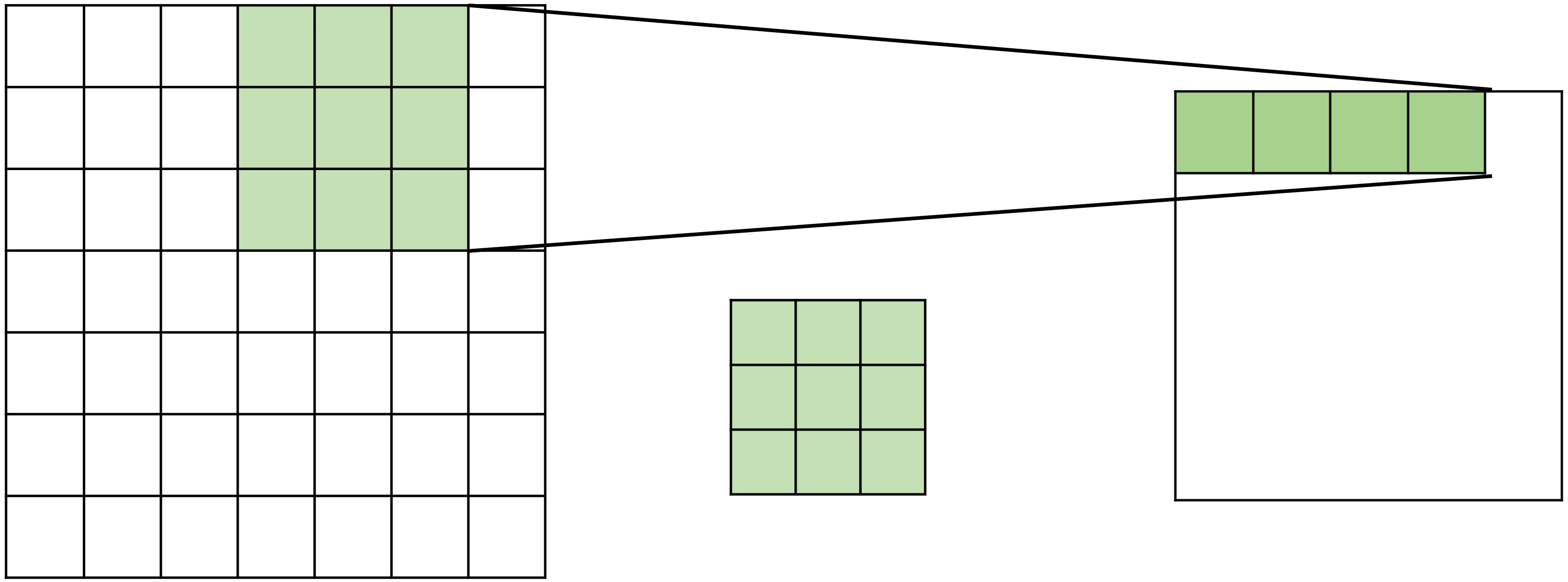
Convolutions



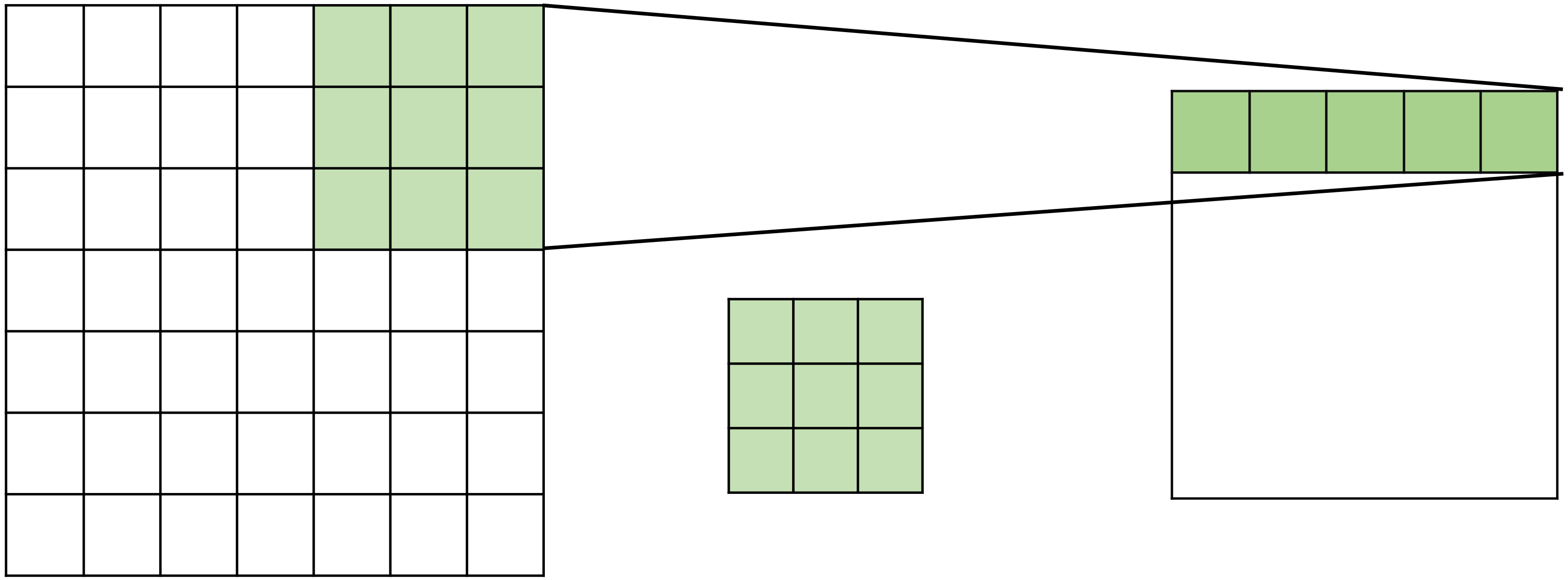
Convolutions



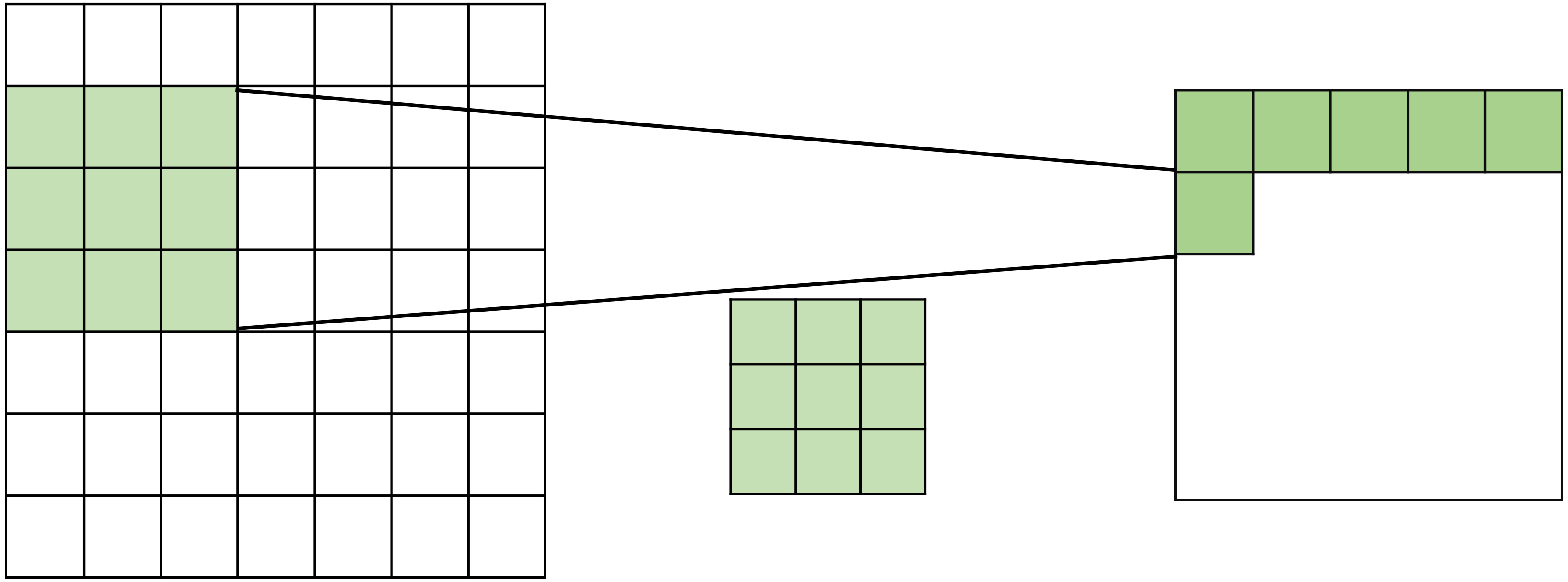
Convolutions



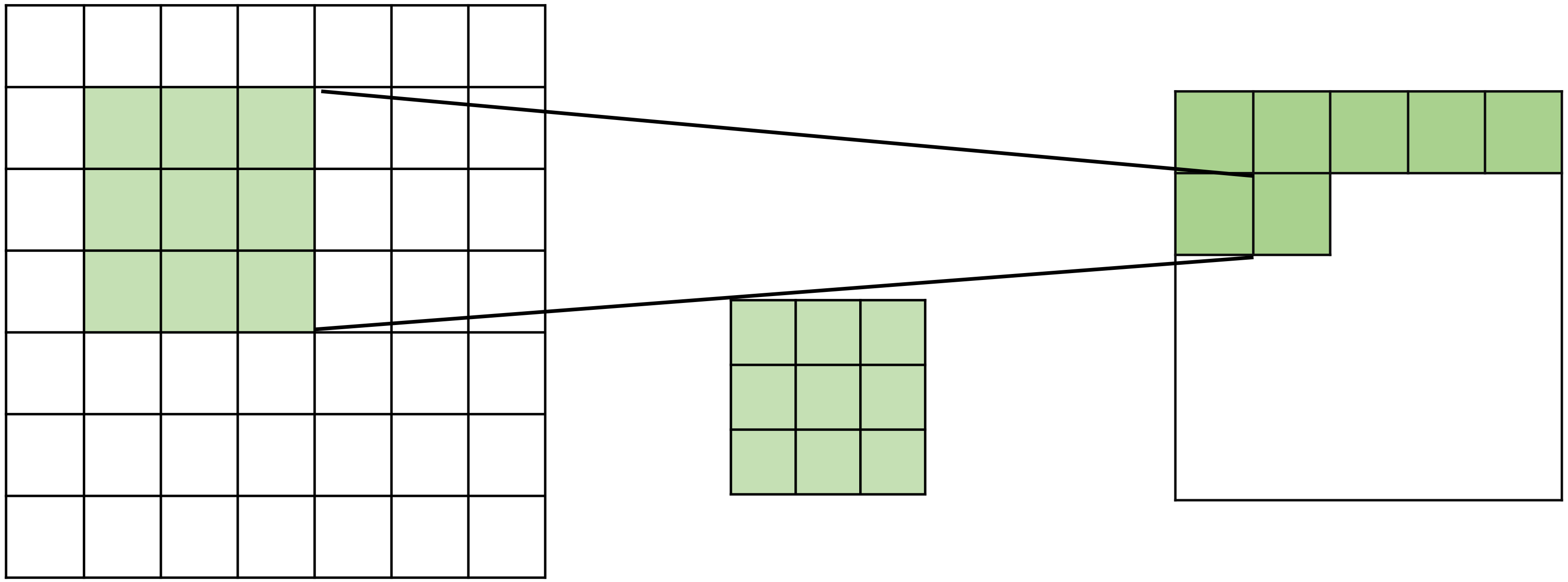
Convolutions



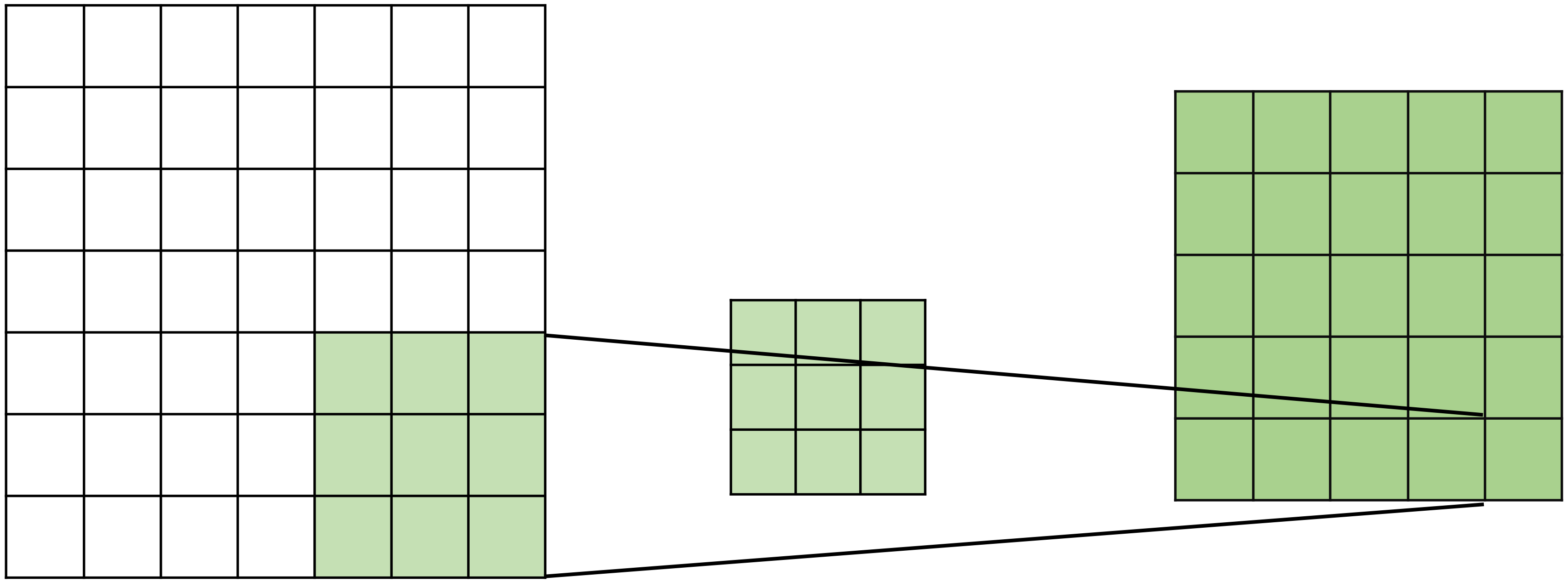
Convolutions



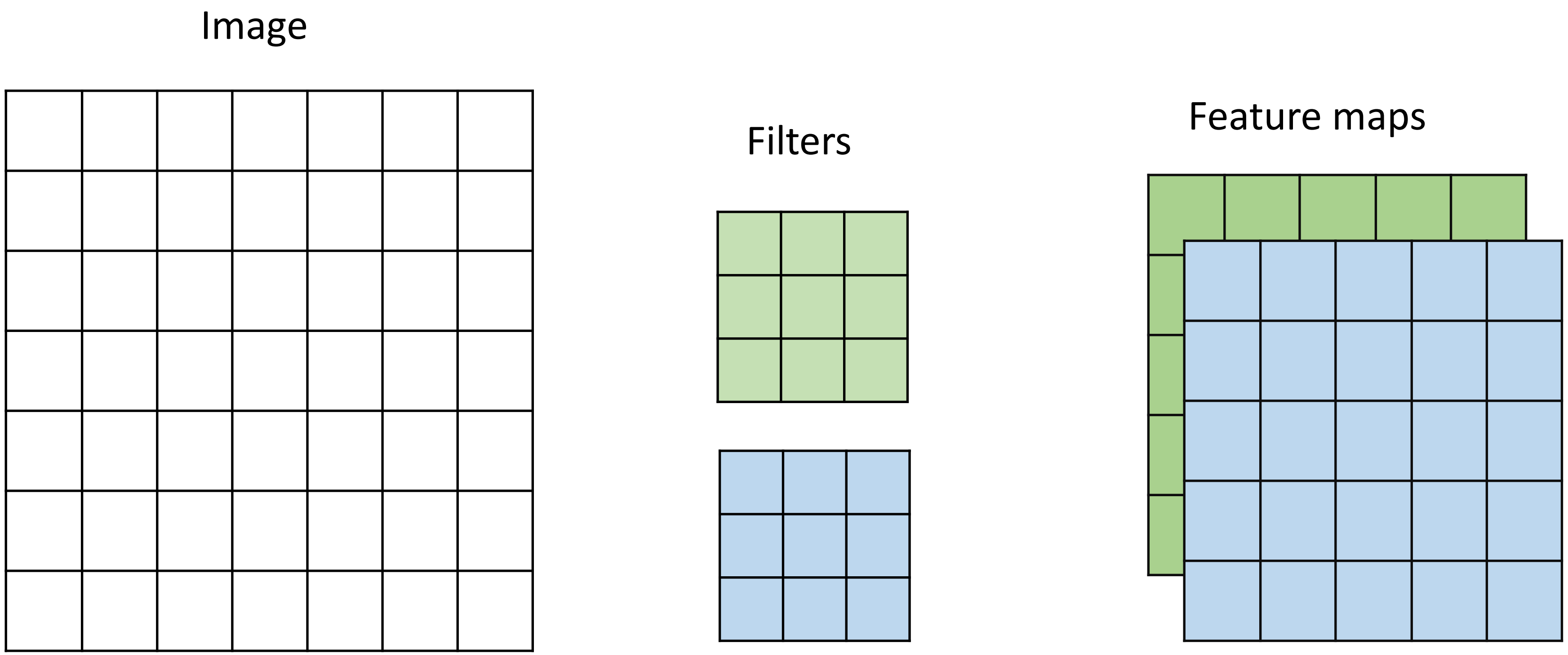
Convolutions



Convolutions

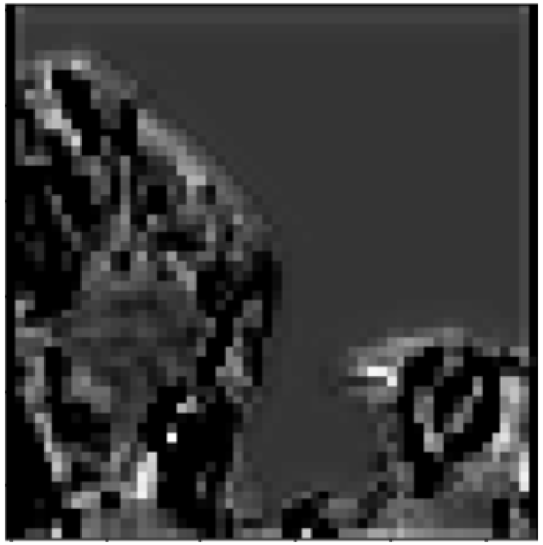


Convolutions

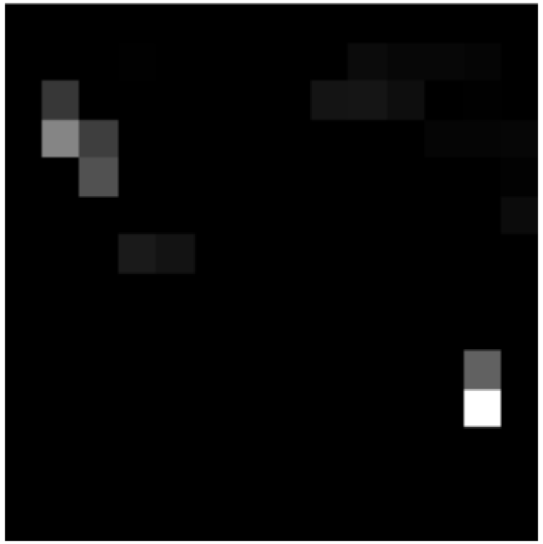


What are these feature maps actually?

Input Image

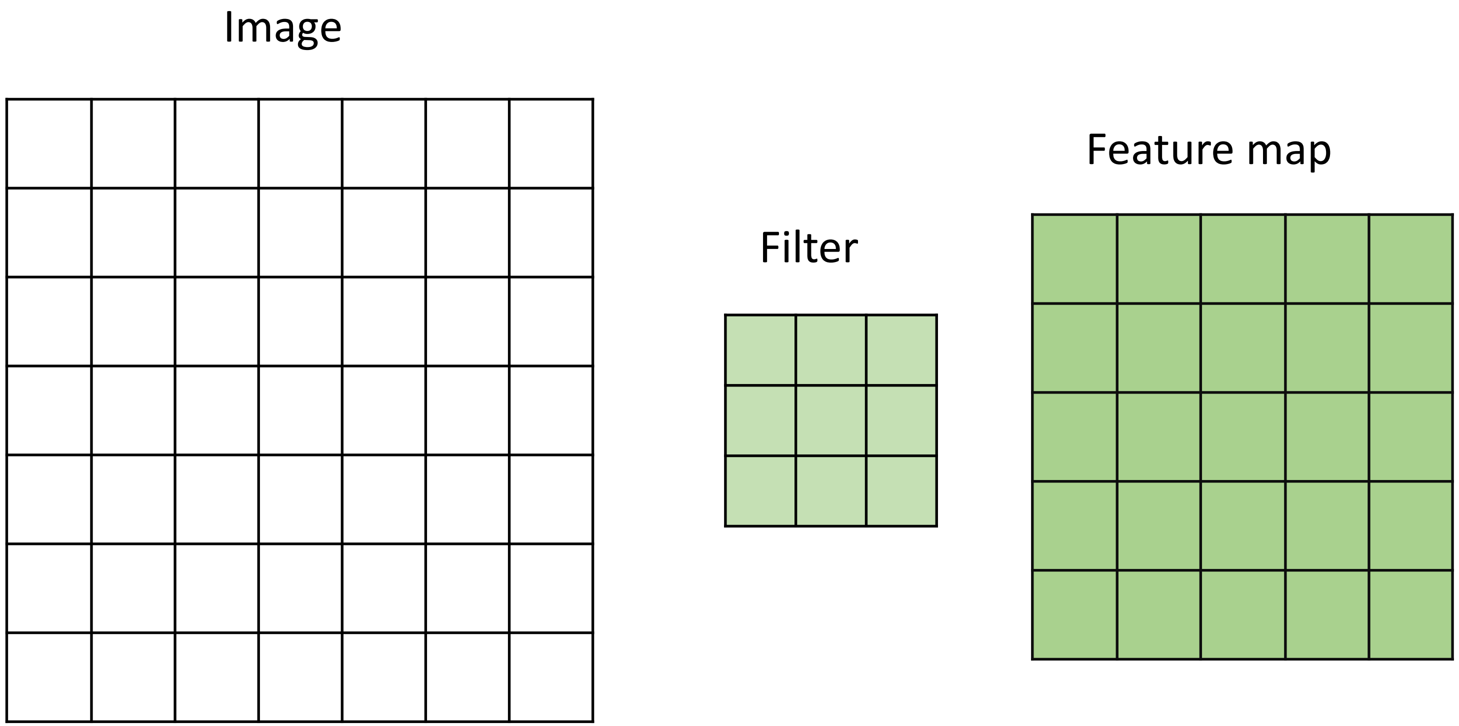


Feature map obtained after first convolution operation



Feature map obtained at upper layers

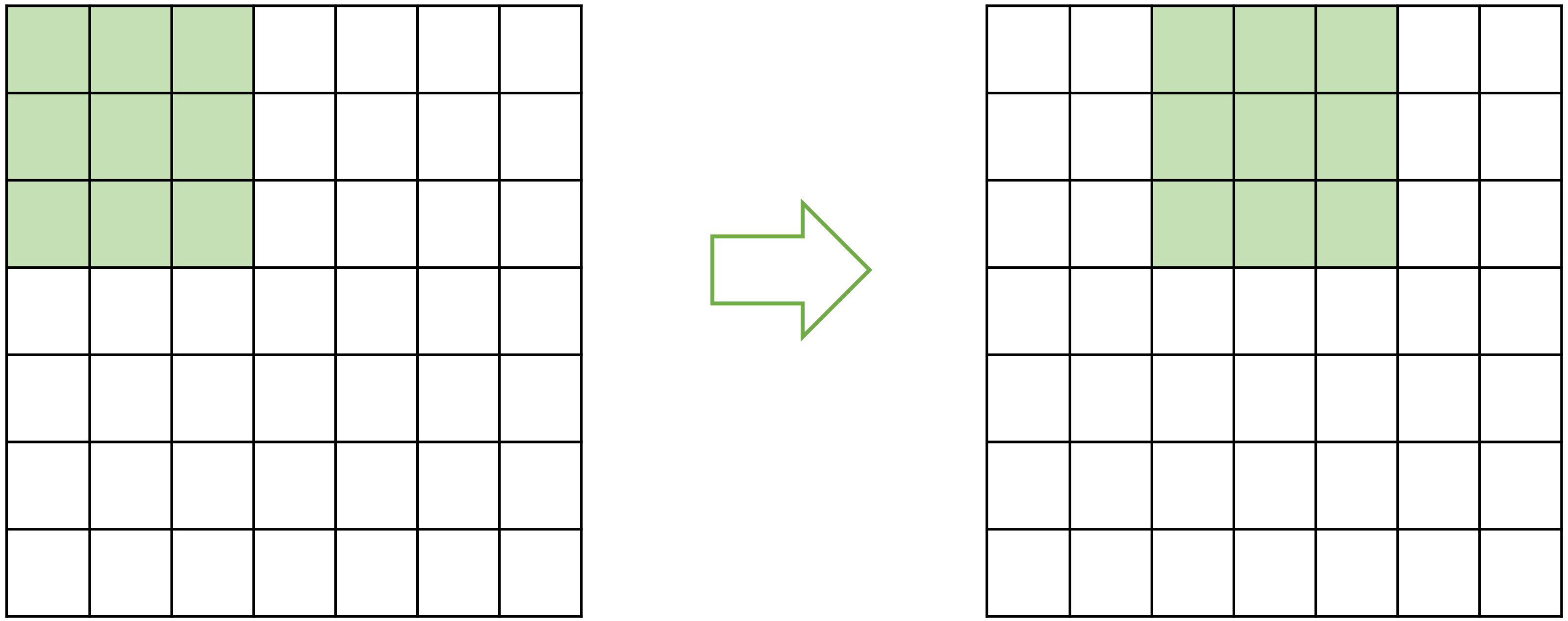
Convolutions



- Image dimension: 7x7
- Filter dimension: 3x3
- Stride: 1 (the number of pixels/units) the filter moves towards right or bottom
- Feature map: 5x5

Convolutions

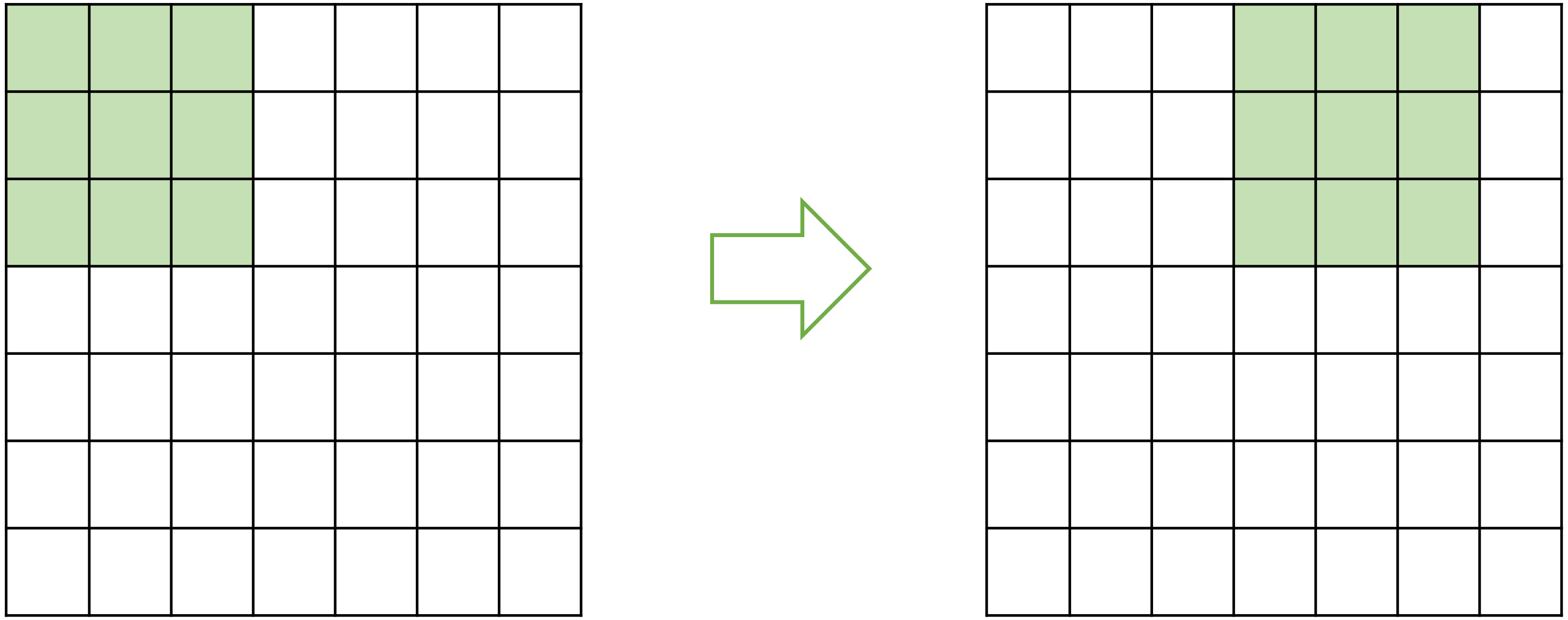
- What is the dimension of the feature map if the stride is 2?



- Feaure map: 3x3

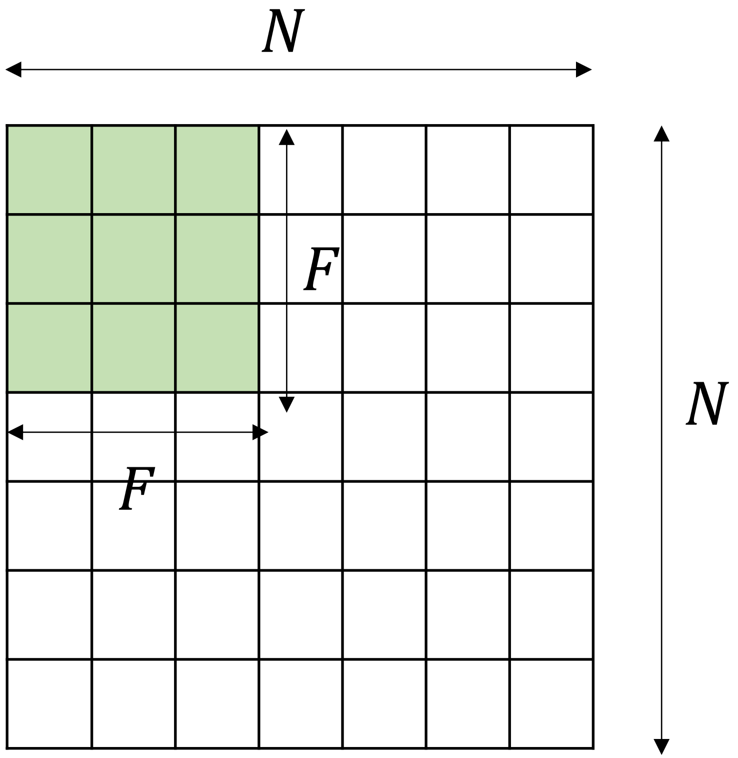
Convolutions

- What happens when the stride is 3?



- Feaure map: Does not fit

Convolutions



Output size = $\frac{N-F}{Stride} + 1$

E.g.,

$N = 7, F = 3, stride = 1$, Output = 5

$N = 7, F = 3, stride = 2$, Output = 3

$N = 7, F = 3, stride = 3$, Output = 2.33

Convolutions

Solution: Zero pad the border

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

Output size = $\frac{N-F}{Stride} + 1$

E.g.,

Input: 7x7

Filter: 3x3

Stride: 1

Pad with 1 pixel border

Output? 7x7

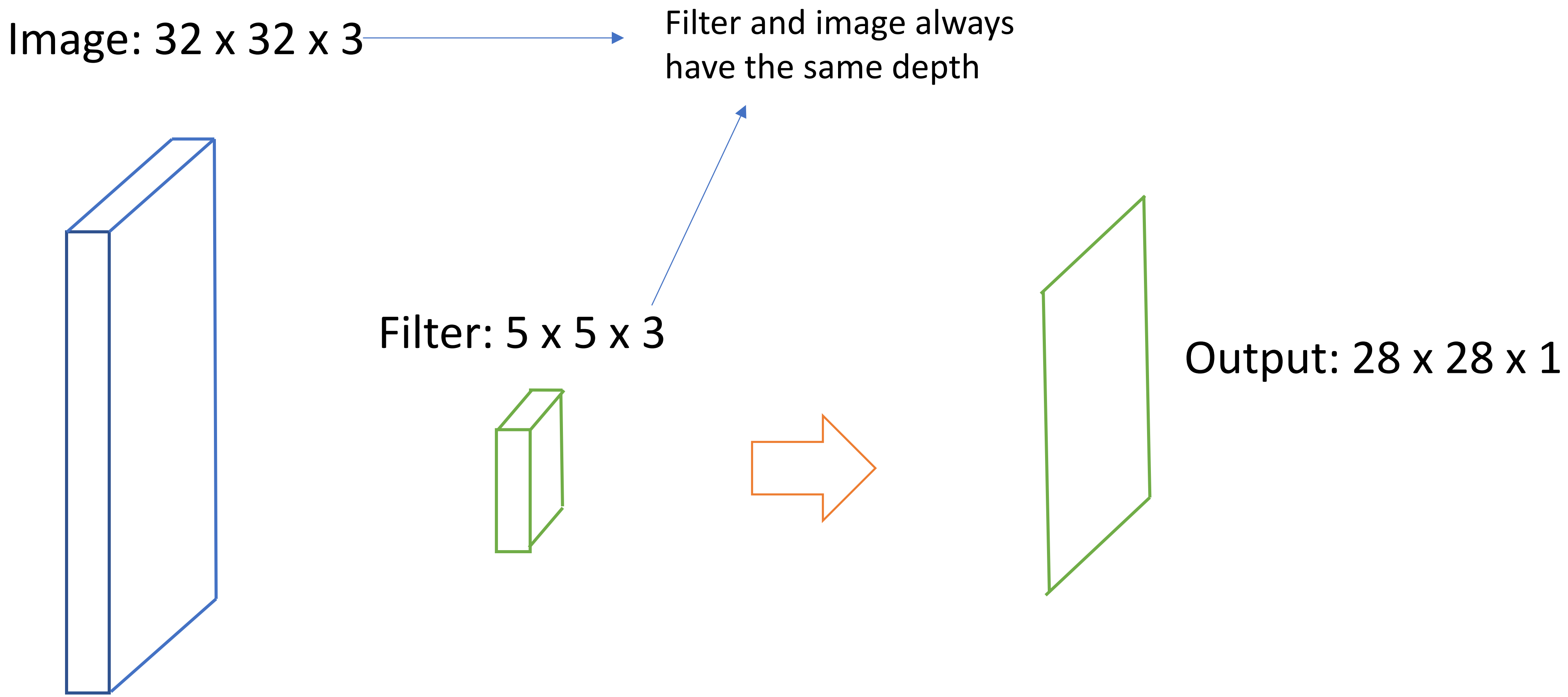
Convolutions

Solution: Zero pad the border

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

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 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

3D convolutions



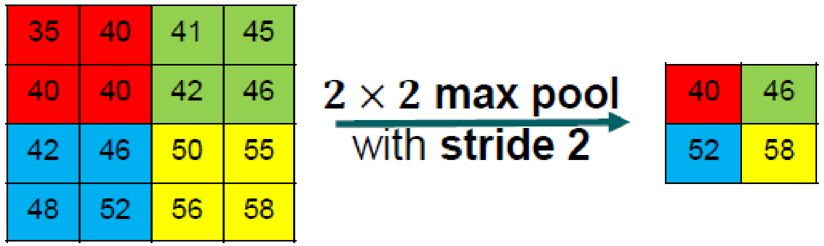
Convolutions

- Input: $W_1 \times H_1 \times D_1$
- Number of filters: K
- Filter dimensions: $F \times F$
- Stride: S
- Padding: P
- Output
  - $W_2 = \frac{W_1 + 2 * P - F}{S} + 1$
  - $H_2 = \frac{H_1 + 2 * P - F}{S} + 1$
  - $D_2 = K$

Convolution operations are followed by non-linear activations such as Sigmoid or ReLU

Pooling

- Second component of CNNs: **Pooling Layers**
- Reduce the size of the input image in a predefined manner
- No learned weights ! Purely deterministic operation
- Common types:
  - Max pooling
    - Extract the maximum of $n \times m$ (pool size) entries in the input
    - Move k (stride) entries ahead in the input



Average pooling: Compute the average of inputs instead of max

Pooling layers

- Why use pooling layers?
- Reduce the number of parameters in following layers
- Not all filters activated on all pixels
- Only use pixels in a close neighbourhood that provide the strongest signal

CNNs vs. Fully connected layers

- Massively fewer parameters than fully connected networks
- Example:
  - 300×300 pixel input , map to hidden layer of same size
  - Fully connected : (300·300)·(300·300) = 8,100,000,000parameters!
  - Convolutional with 100 filters of size 3×3: 100·3·3 = 900parameters
- Less RAM needed , less prone to overfitting

CNNs for Text

CNNs in NLP

- So far: Only described use in Computer Vision…
- CNNs became popular in NLP, too:
  - Kalchbrenner et al., 2014: A Convolutional Neural Network for Modelling Sentences
  - Kim, 2014: Convolutional Neural Networks for Sentence Classification
  - Nguyen and Grishman, 2015: Relation Extraction: Perspective from Convolutional Neural Networks
  - …

CNNs for Sentence Classification

- Kim, 2014 (EMNLP)
- Very simple CNN model…
- … with very good results!
- Beat previous state-of-the-art in several tasks
  - Sentiment Analysis
  - Subjectivity Detection*
  - Question Answering

*didn‘t beat state-of-the-art, but came very close

CNNs for Sentence Classification

- Input representation: Concatenated word embeddings

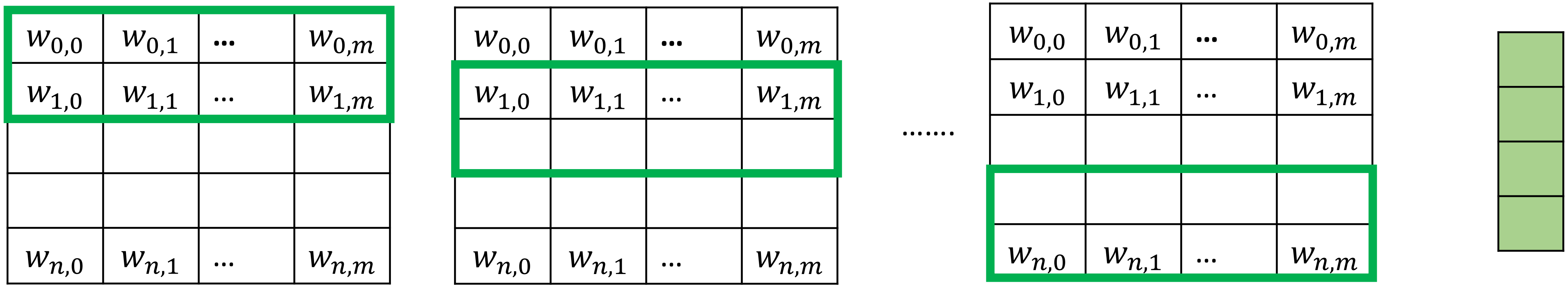
Embedding of a word

$w_{0,0}$	$w_{0,1}$	…	$w_{0,m}$
$w_{1,0}$	$w_{1,1}$	…	$w_{1,m}$
$w_{n,0}$	$w_{n,1}$	…	$w_{n,m}$

Sentence

CNNs for Sentence Classification

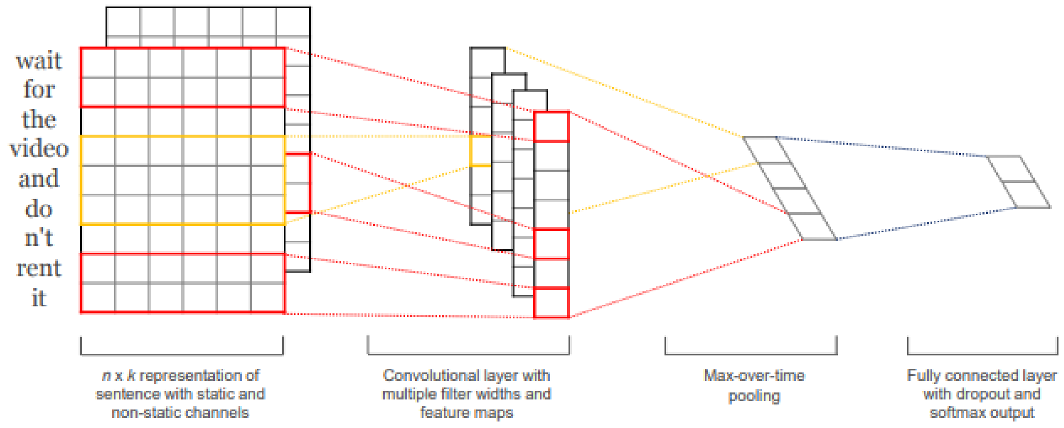
- Convolution operation



- Convolutions over full embeddings!
- Filter size $n \times m$ (n = variable, m = length of the embeddings)

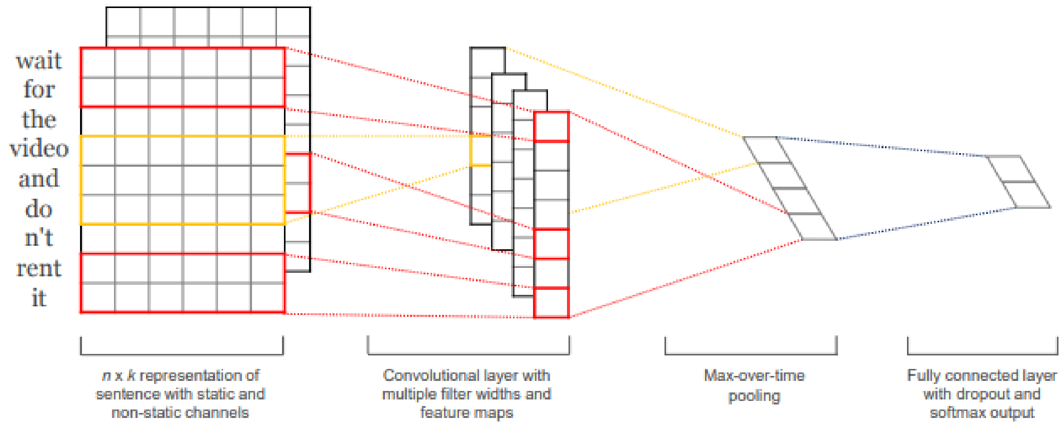
CNNs for Sentence Classification

- Network architecture
  - No stacked convolutional layers!
  - One layer of convolutions
  - Different filter sizes:
    $n \in \{3,4,5\}$
  - Each with 100 filters
  - Stride 1 -> Do not skip any words/n-grams

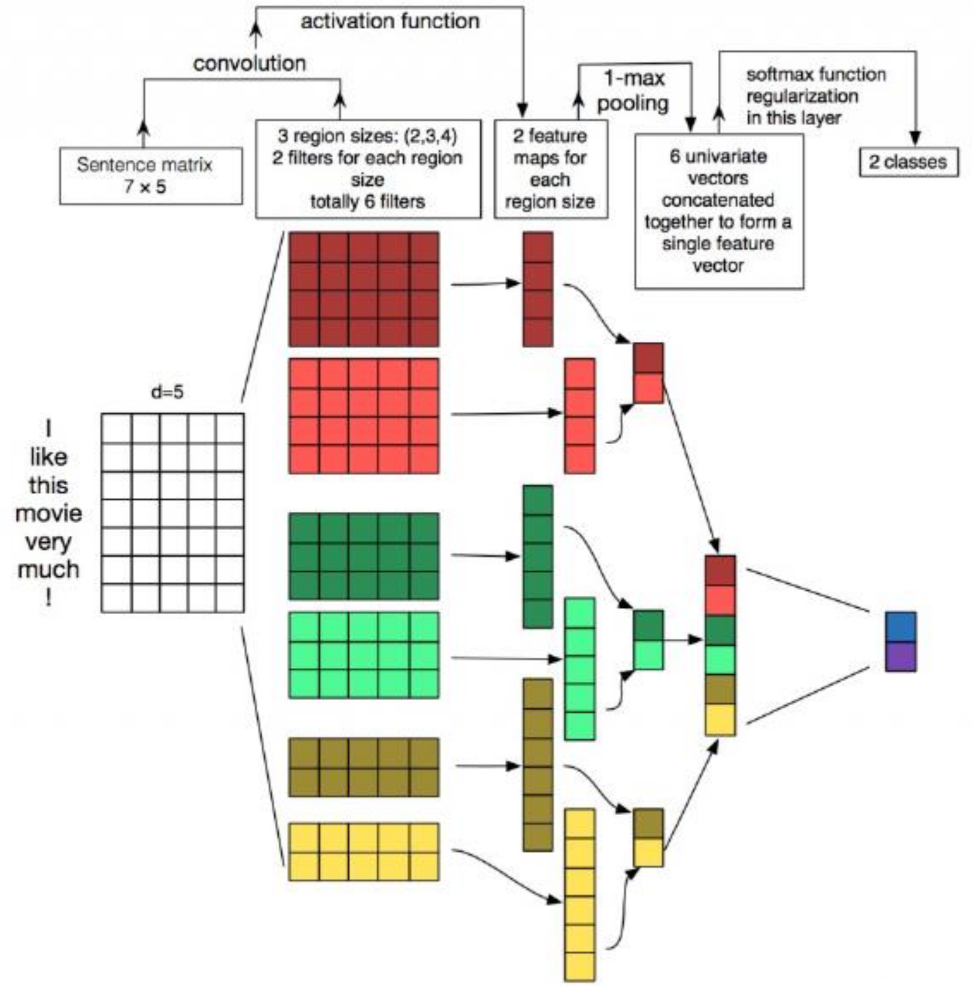


CNNs for Sentence Classification

- Network architecture
  - Max-over-time pooling
  - Max Pooling over the full filter output -> Select the highest output for each filter
  - Fully connected layer
  - Dropout
  - Softmax



CNNs for Sentence Classification



https://dennybritz.com/posts/wildml/understanding-convolutional-neural-networks-for-nlp/

CNNs for Sentence Classification: Embeddings

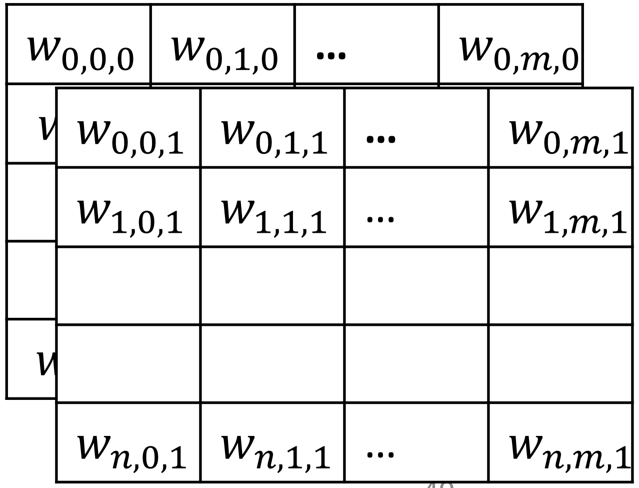
- Word embeddings used to encode the input
- Multiple variants:
  - Randomly initialize embeddings, train with model (cnn-rand)
  - Initialize word embeddings with Word2Vec, keep fixed (cnn-static)
  - Initialize word embeddings with Word2Vec, train with model (cnn-nonstatic)

- Finding: Using pre-trained embeddings and further optimizing them for the task at hand often works best!

CNNs for Sentence Classification: Embeddings

- Trick: Multi-channel word embeddings (cnn-multichannel)
  - Represent input as 3-dimensional matrix
  - 3rd dimension: different, independent word embedding vectors
  - Convolutions computed slightly differently, general ideas still apply

- During training:
  - keep one dimension fixed
  - Train the other
- Effect: Keep information from original embeddings, but also include „extra“ for the task/dataset



CNNs for Sentence Classification: Regularization

- Multiple regularization methods used:
  - Dropout
    - Apply Dropout after the fully connected layer
  - L2-maxnorm constraint
    - Set a hard limit s on l2-norm for weights w of the fully connected layer
    - If, after the update step, $||w||_2 > s$, rescale w to $||w||_2 = s$
    - Prevents single weights from getting too large

CNNs for Sentence Classification: Evaluation

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**	–	–

CNNs for Sentence Classification: Evaluation

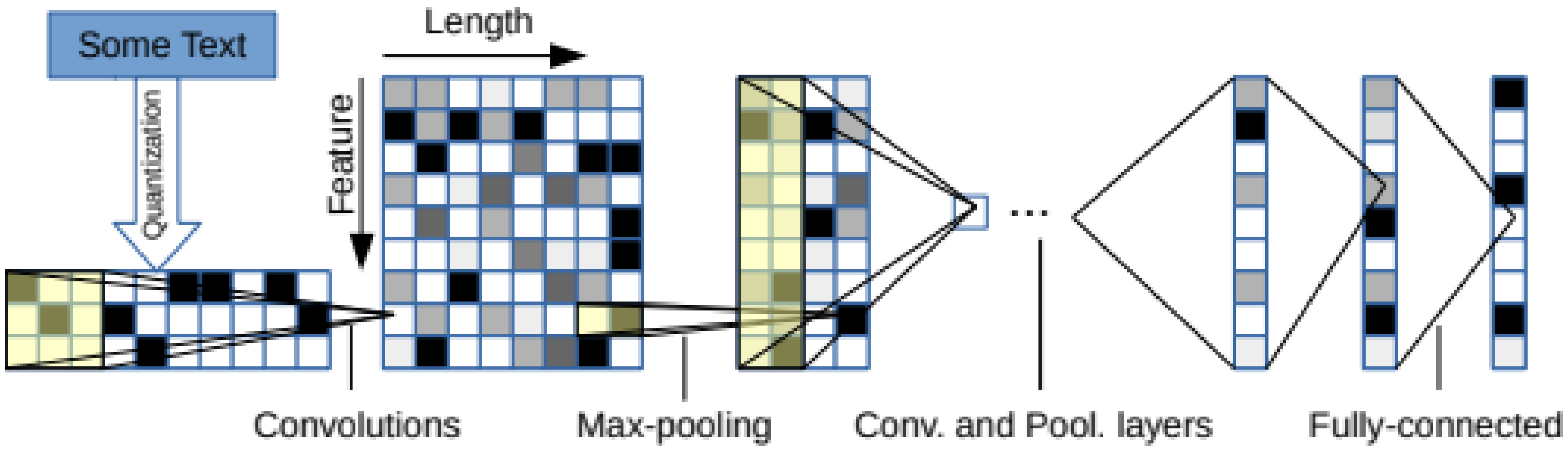
- General findings:
  - Model performs very well!
  - Using pre-trained word embeddings is better than randomly initializing
  - No clear advantage for using cnn-static, cnn-nonstatic or cnn-multichannel

Character level CNNs

- Idea: Learn from scratch, ignore notion of words (!)
- Instead of having a sequence of words for which we train embeddings: Train on sequence of characters with embeddings
- Alphabet used (70 different characters):

```
abcdefghijklmnopqrstuvwxyz0123456789,;.!?:’’’/\|_@#$%^&*˜‘+-=<>()[]{}
```

Character level CNNs



Training

- The algorithm used is stochastic gradient descent (SGD) with a minibatch of size 128, using momentum 0.9 and initial step size 0.01 which is halved every 3 epochs for 10 times.

- Problem: Text length
- Solution: Text a fixed length of 1014 characters (!). If necessary, padding with spaces

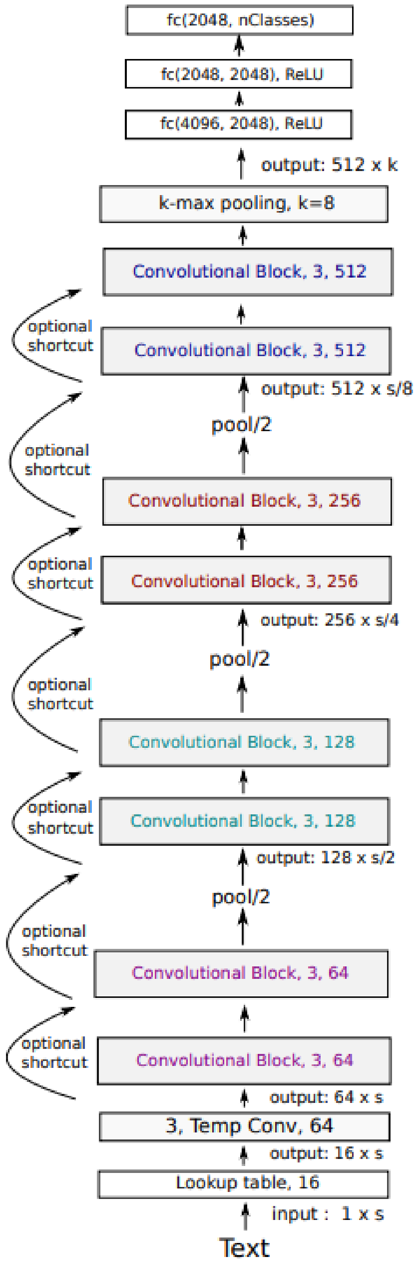
Evaluation

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Small Large

Even deeper CNNs

- Same ideas, deeper architecture
- Upto 49 layers of blocks
- Achieves better performance
- High computational cost



Reference and further reading

- https://cs231n.github.io/convolutional-networks/
- https://www.youtube.com/watch?v=bNb2fEVKeEo