Artificial Neural Networks II

Esra Suel

CASA0006: Data Science for Spatial Systems

Slides adapted from Ender Konukoglu at ETH Zurich & Huanfa Chen at UCL

Outline

Deep learning applications

- Deep learning for images: Convolutional Neural Networks (CNNs)
- CNNs for geospatial research
- Deep learning for text in geospatial research

Deep learning for images

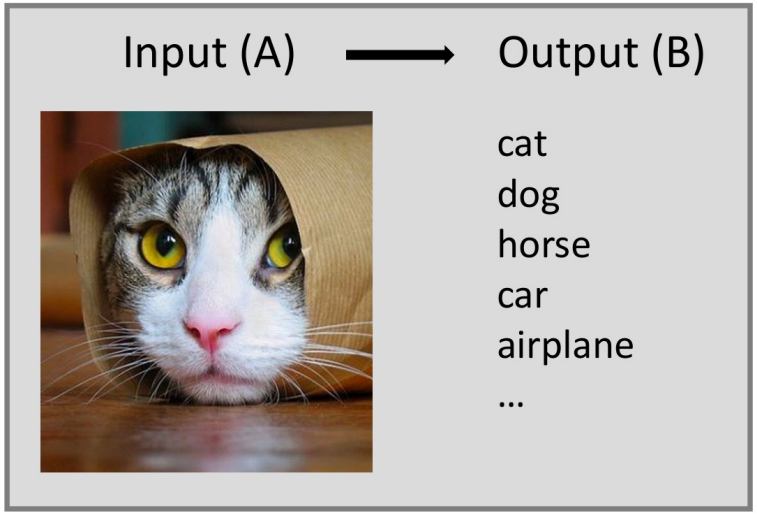
Convolutional Neural Networks (CNNs)

Deep learning for images (computer vision)

Applications

1. Image Classification
2. Image Classification with Localization
3. Object Detection
4. Semantic Segmentation
5. Instance Segmentation

Image Classification



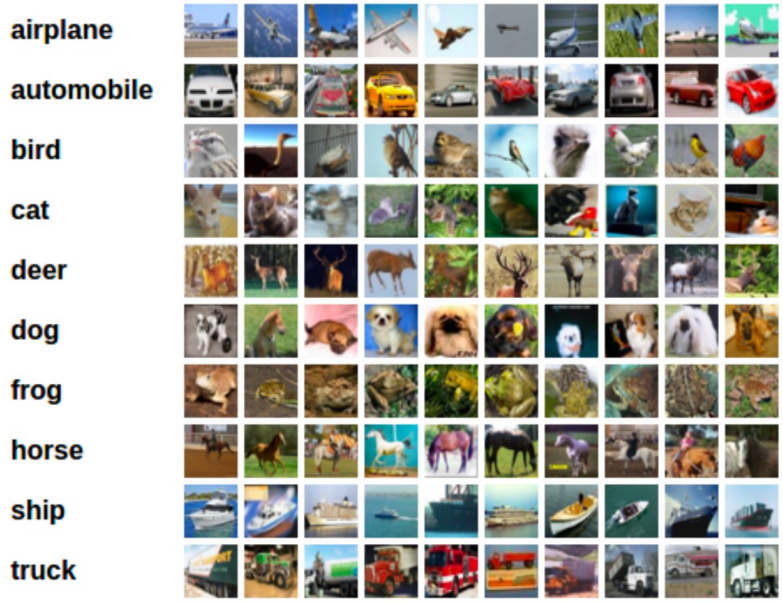
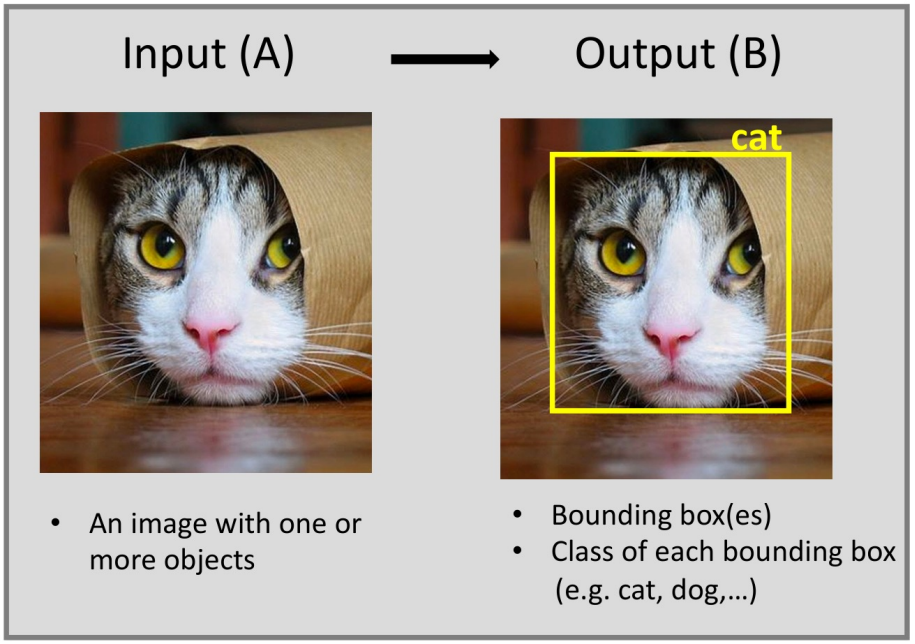


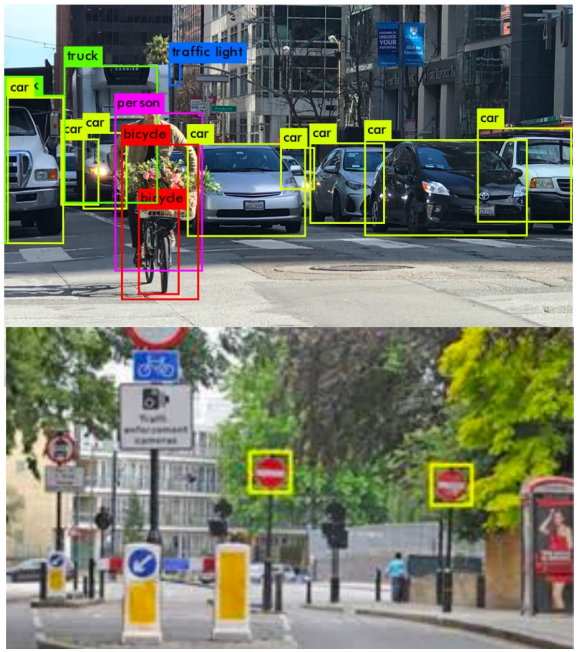
Image Classification with Localization

- Given an image we want to learn the class of the image and the class location in the image. We want a class and a rectangle of where that object is.
- Usually only one object is presented.



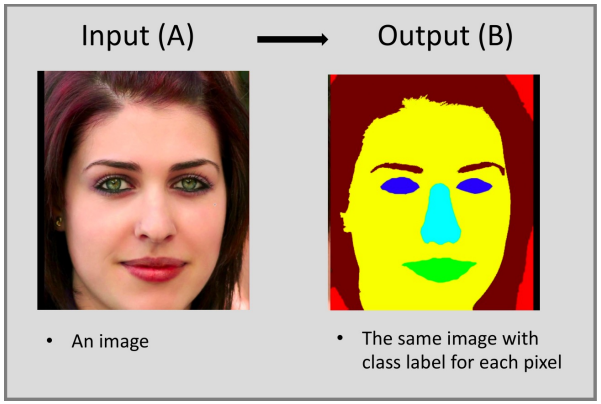
Object Detection

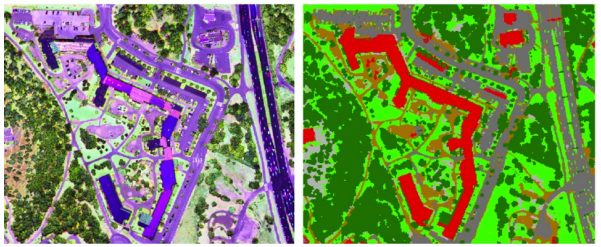
- We want to detect all objects in the image that belong to a specific class and give their boundary box.
- An image can contain more than one object with different classes.



Semantic Segmentation

- We want to label each pixel in the image with a category label.
- Semantic Segmentation don't differentiate instances, only care about pixel labels.
- It detects no objects just pixels – it does not tell you #cars in the image.
- If two objects of the same class are intersected, we won't be able to separate them.





Instance Segmentation

- This looks like the full problem and the most challenging.
- We want to identify and distinguish all objects.

Recap from the lecture on introduction to neural networks

Hidden layers

$$h_l = \sigma(\mathbf{W}_l h_{l-1} + b_l)$$

$$h_0 = \mathbf{x}, \;\; h_L = f(\mathbf{x}; \theta)$$

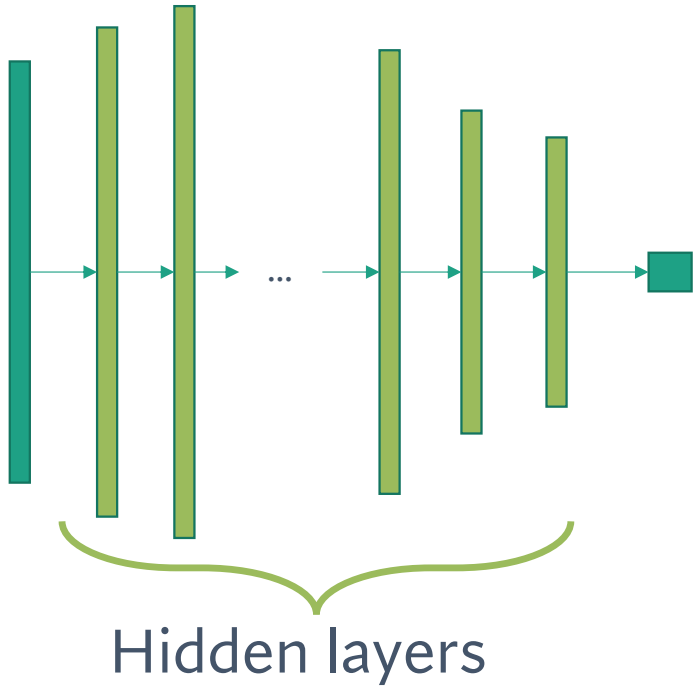
Function view

$$f_l(h_{l-1}; \theta_l) = \sigma(\mathbf{W}_l h_{l-1} + b_l)$$

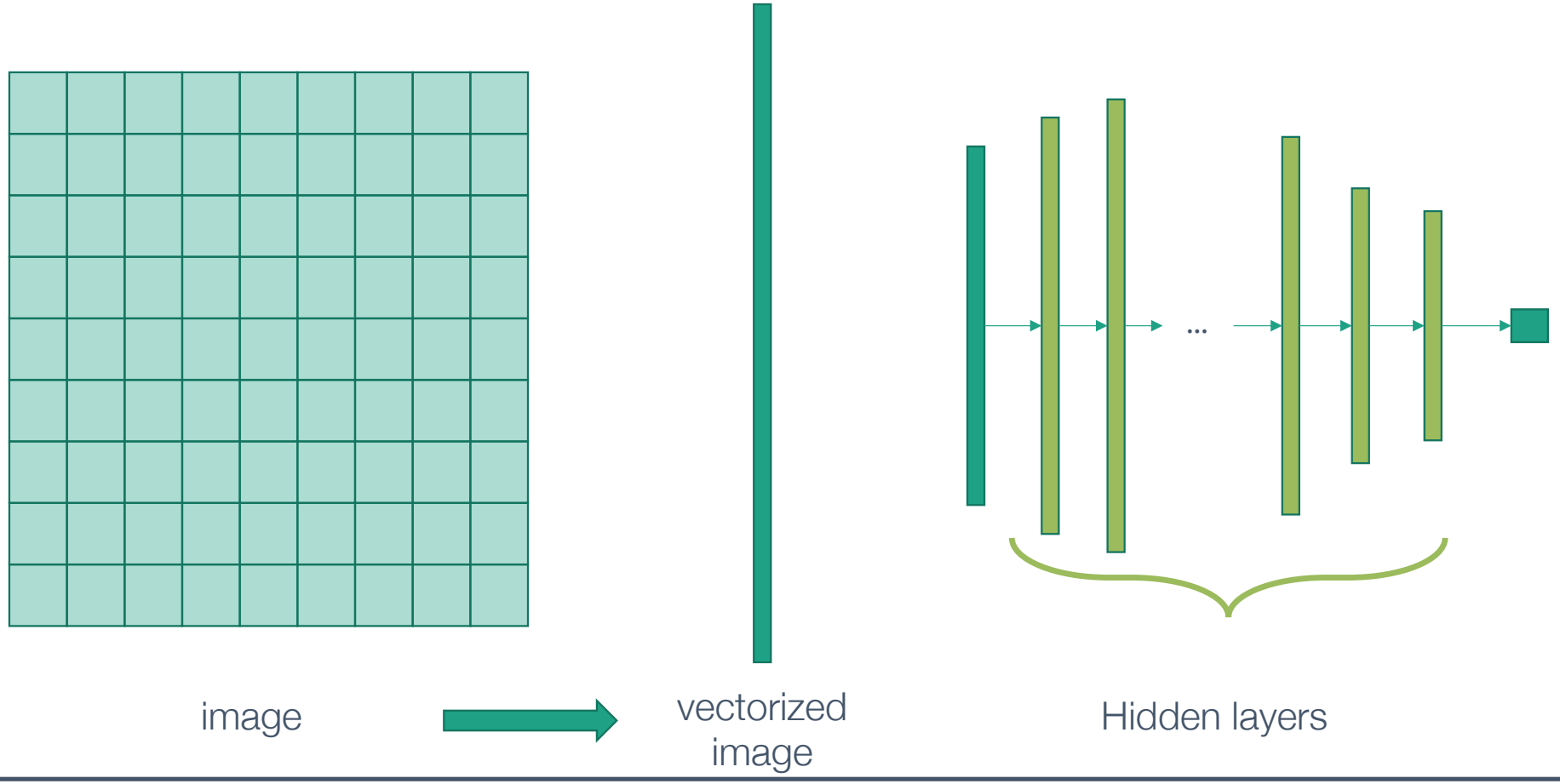
$$f(\mathbf{x}; \theta) = f_L \circ \cdots \circ f_2 \circ f_1(\mathbf{x})$$

$$\theta = \{\theta_L, \ldots, \theta_1\}$$

We always considered x as a vector.
What to do when x is an image?

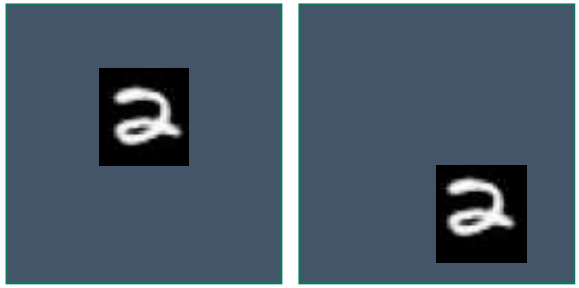


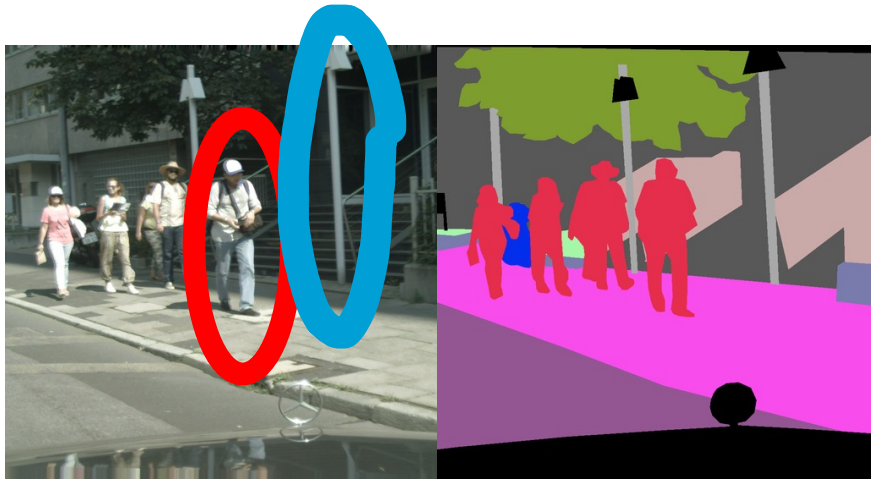
Naïve approach



Challenges with using a fully connected network

- Fully connected links lead to too many parameters.
- Images are composed of a hierarchy of local statistics
- Lack of translation invariance – we would want the network to still recognize the object if its position is changed.





Convolutional neural networks (CNNs)

Fully connected architecture

$$a_{l,k} = \sum_j w_{l,kj} h_{l-1,j} + b_{l,k}$$

$$h_{l,k} = \sigma \left(\sum_j w_{l,kj} h_{l-1,j} + b_{l,k} \right)$$

Convolutional architecture

$$a_{l,k} = \sum_j w_{l,kj} * h_{l-1,j} + b_{l,k}$$

$$h_{l,k} = \sigma \left(\sum_j w_{l,kj} * h_{l-1,j} + b_{l,k} \right)$$

Convolutional layers

Convolution

$$w * x$$

Image: x

544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

Convolution kernel or filter: w

1	0	-1
1	0	-1
1	0	-1

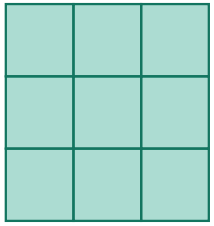
Convolution

$$w * x$$

Image: x

544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

Convolution kernel: w



$$a_{ij} = \sum_p \sum_q x_{(i-p)(j-q)} w_{(p)(q)}$$

Convolution

$$w * x$$

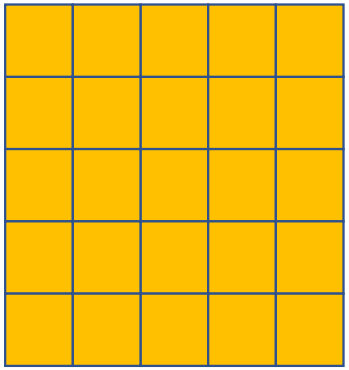
Image: x

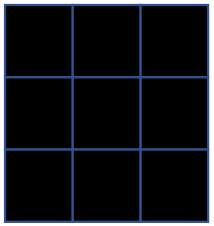
544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

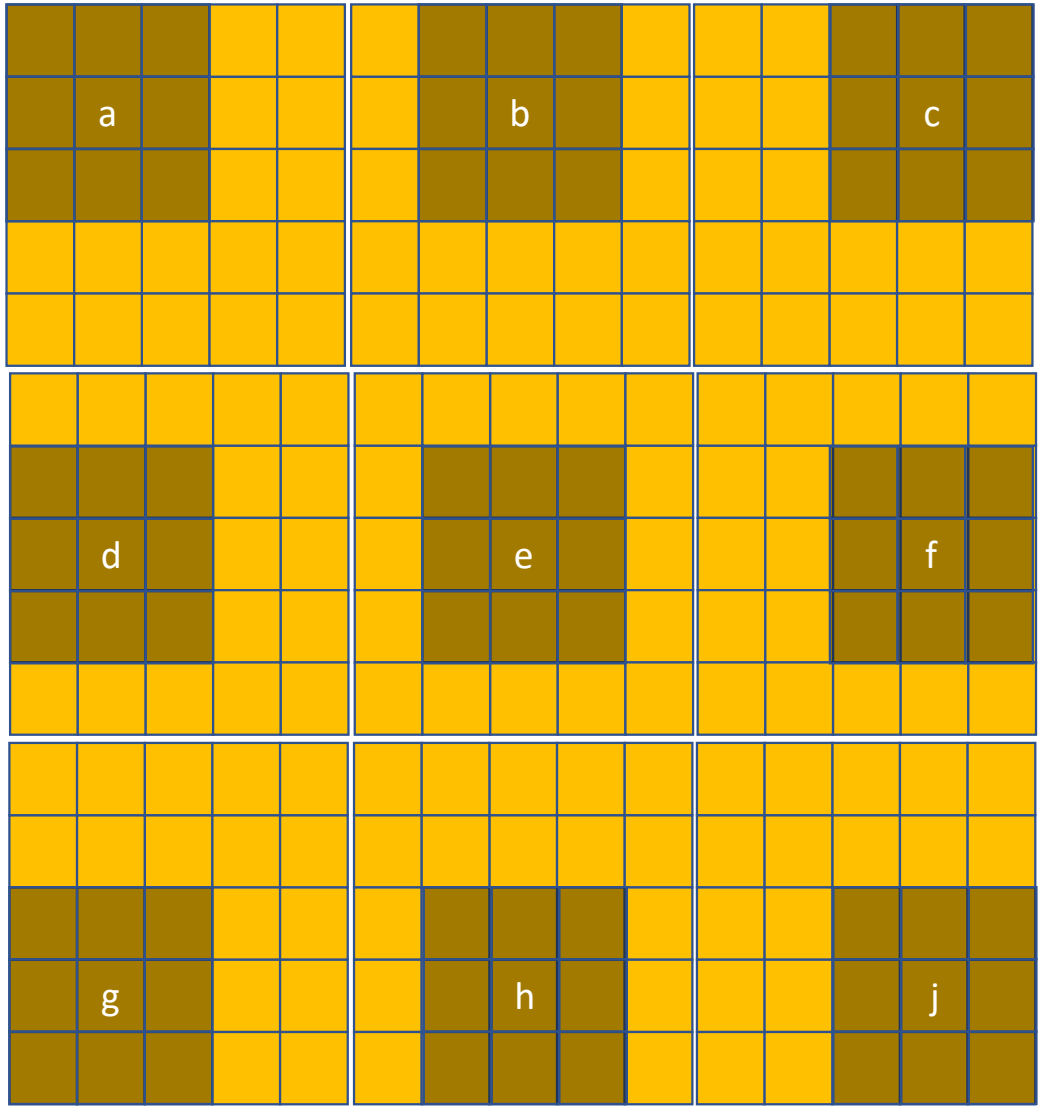
Convolution kernel: w

1	0	-1
1	0	-1
1	0	-1

$$a_{ij} = \sum_p \sum_q x_{(i-p)(j-q)} w_{(p)(q)}$$

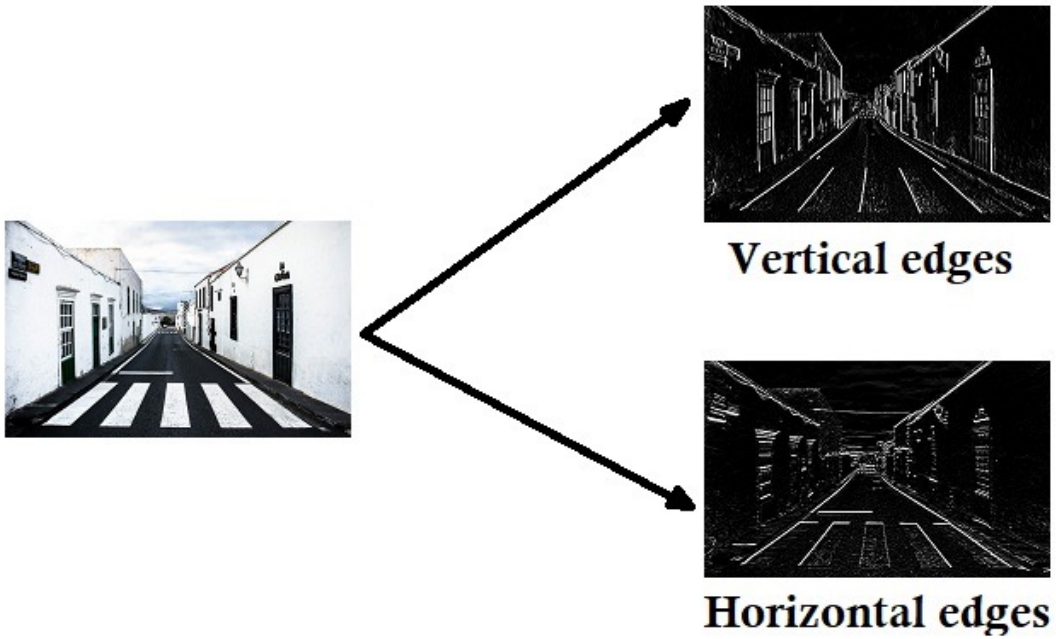






a	b	c
d	e	f
g	h	j

Convolution operation and filter (kernel)



Convolution operation and filter (kernel)

vertical edge filter

1	0	-1
1	0	-1
1	0	-1

Vertical edges

Horizontal edges

Convolution operation and filter (kernel)

vertical edge filter

1	0	-1
1	0	-1
1	0	-1

Vertical edges

Horizontal edges

1	1	1
0	0	0
-1	-1	-1

horizontal edge filter

Convolution operation and filter (kernel)

vertical edge filter

1	0	-1
1	0	-1
1	0	-1

Vertical edges

Horizontal edges

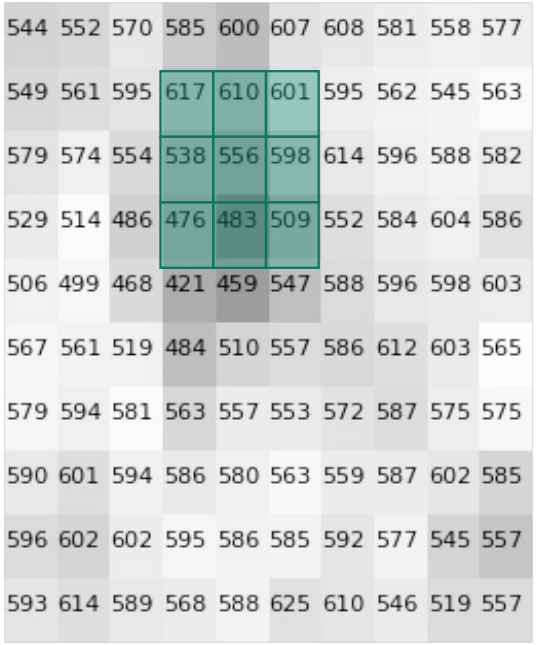
1	1	1
0	0	0
-1	-1	-1

horizontal edge filter

Padding

Padding

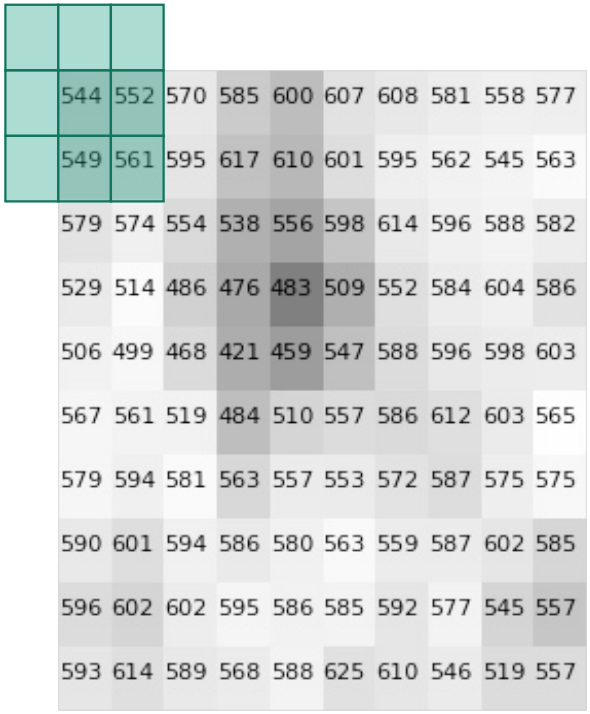
The output channel size is determined by the kernel size and the type of convolution



- When the kernel is placed in the image – no problem

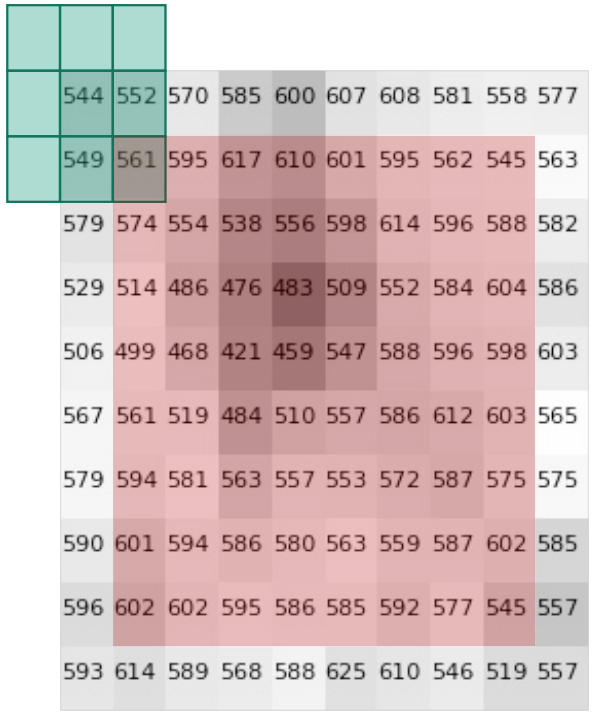
Padding

Output/activation size is linked with kernel size and the type of convolution



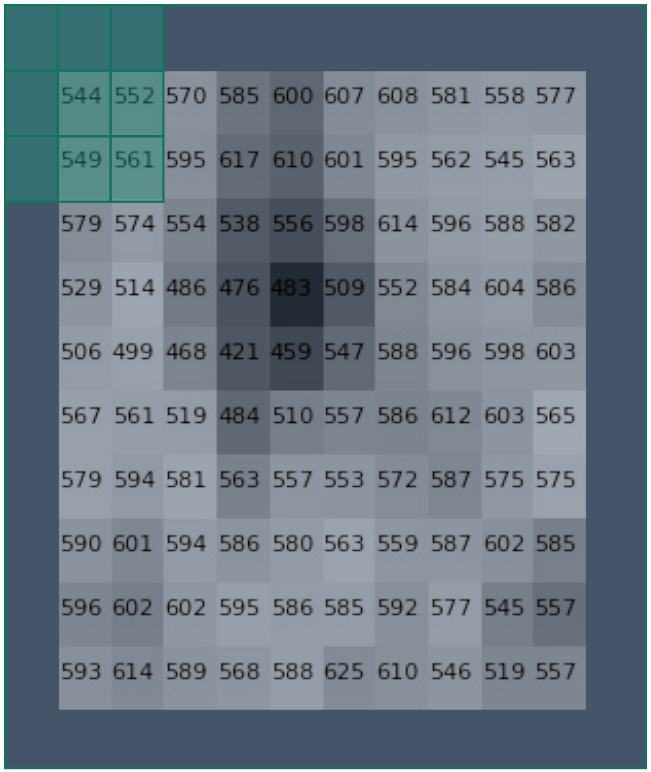
- When the kernel is placed in the image – no problem
- When it is placed on the boundary – it is not well defined
- Out-of-boundary values are not defined
- Two options:
  1. Valid convolution: only evaluate convolution when all the elements are defined
  2. Padding (Same): pad the boundaries so that result of the convolution will have the same size

Valid convolution



- If the kernel is centered, then convolution can only be evaluated within the red area
- You loose a pixel at each end of the picture
- Output channel size is smaller than your input image

Same padding



Alternatively, you can pad the image on the boundaries so that channels will have the same size across layers.

Commonly you would use centered kernels and padding around the image as shown on the left

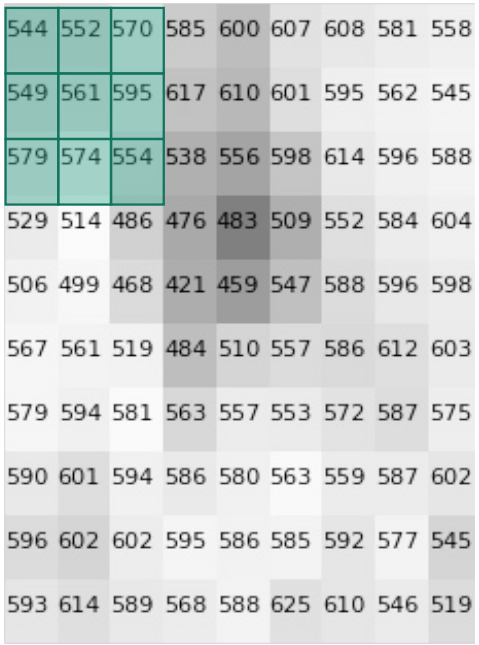
The value you pad is a parameter, 0 is used often but you can use symmetric padding for certain applications.

How much padding you want to have will also depend on your filter size

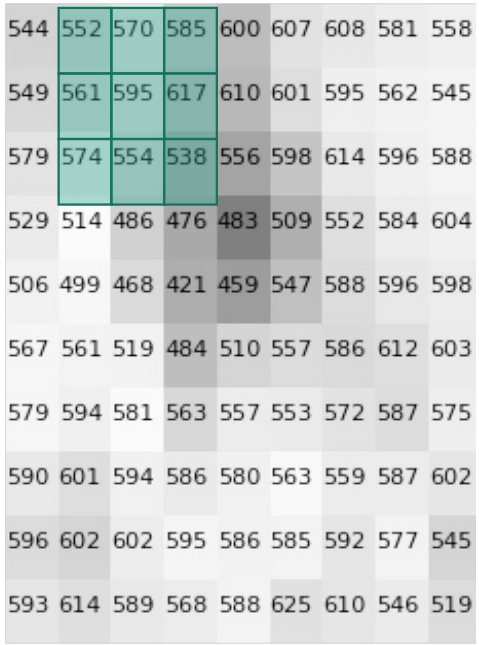
Stride

Strided convolution

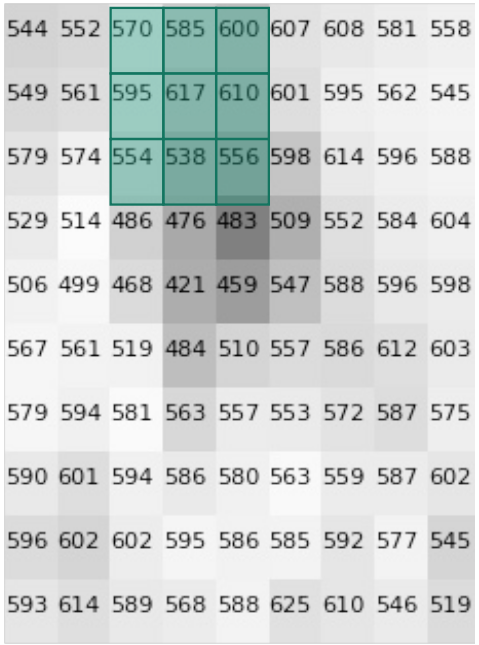
Stride controls how much we shift the filter at each step when sliding it over the image. A larger stride results a smaller output image.



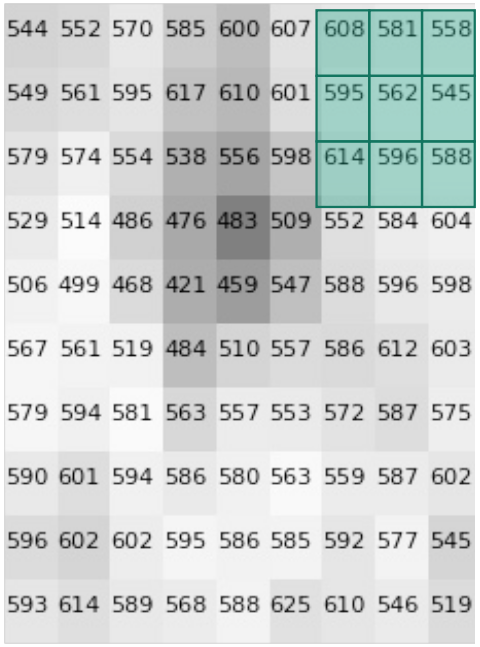
s=1



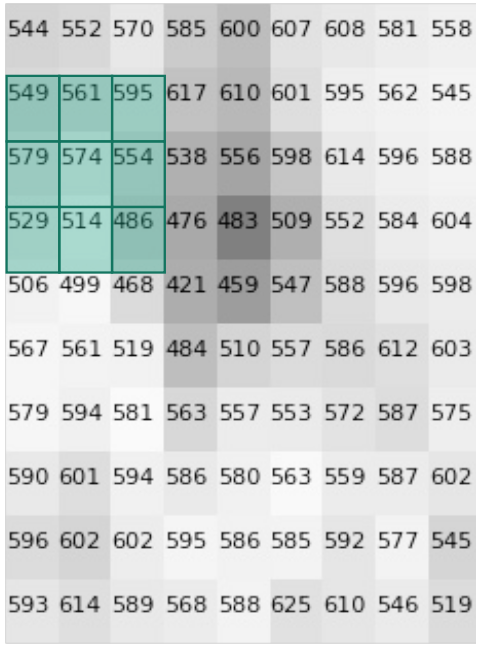
s=2



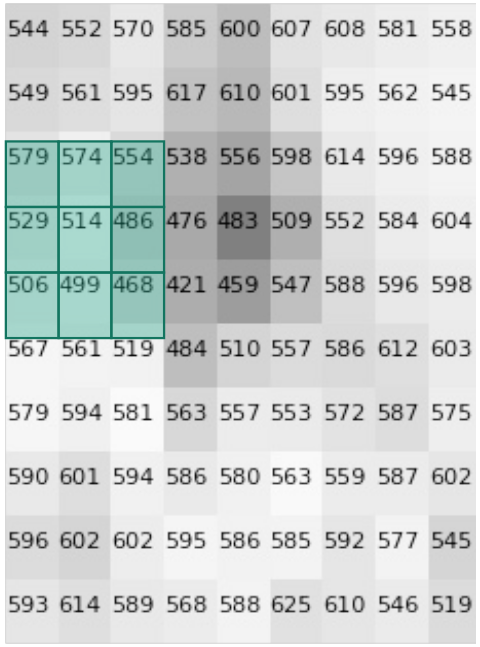
Strided convolution



s=1

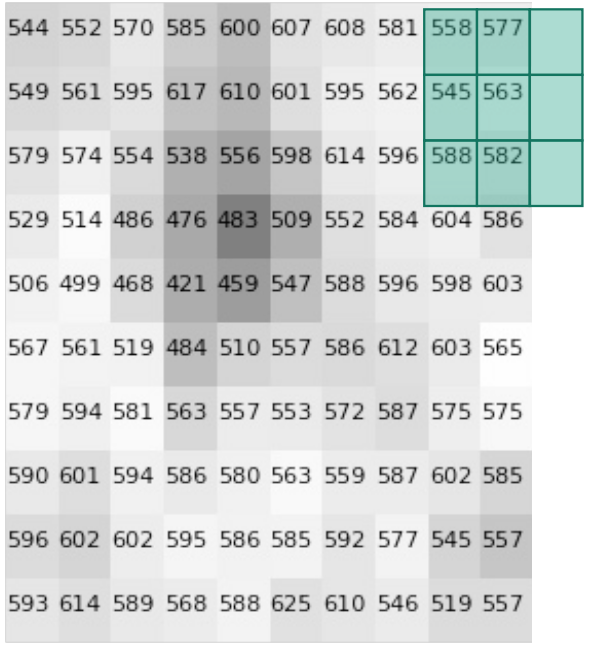


s=2



Strided convolution

- Larger strides will mean output channel size will be smaller.
- Note that if the filter/kernel does not lie entirely lie within the image + padding region – you don't do the computation and not output is generated.



Input and output channel dimensions

$n \; x \; n$ image $\quad\quad$ $f \; x \; f$ filter

padding p $\quad\quad$ stride s

Output channel dimensions:

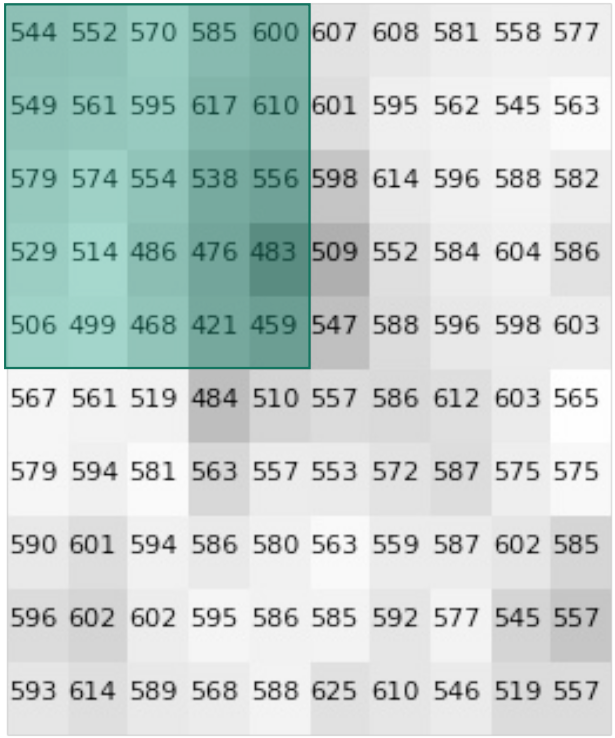
$$\left\lfloor \frac{n + 2p \; - f}{s} + 1 \right\rfloor \quad \text{X} \quad \left\lfloor \frac{n + 2p \; - f}{s} + 1 \right\rfloor$$

Multiple filters/kernels in each layer

At each layer, we don’t just use a single filter. Instead, we typically use multiple filters to capture different features of the image, such as edges, textures, or patterns. Each filter detects a unique feature, and the resulting feature maps form the output of the layer i.e., channel outputs.

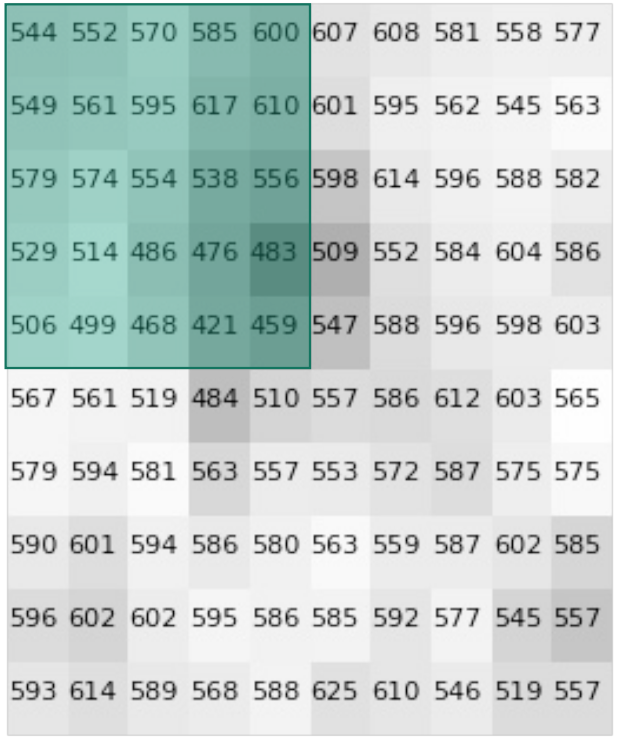
Pooling layers

Pooling



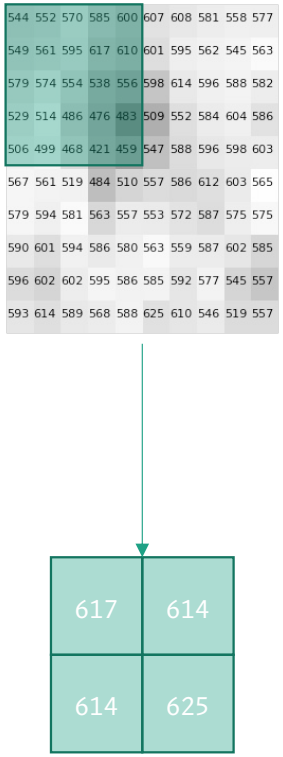
- Pool information in a neighborhood
- Represents the region with one number >> summarizes information
- Applied to each channel separately
- **Max-pooling** – maximum of the activation values
- **Min-pooling** – minimum of the activation values
- Both are non-linear operations, like median filtering
- **Averaging pooling** – linear operator
- Max-pooling is the most commonly used version

Max pooling



- Represents the entire region with the neuron that achieves the highest activation
- 617 in the highlighted area can be in any of the neurons, the pooled value will not change
- Often applied with strides equal to the size of the kernel

Dimensionality reduction



- Leads to a substantial dimensionality reduction
- Even when the pooling kernel is of size 2x2, it can halve the image!
- As the size of the pooling kernel increase, the reduction increases as well
- Non-linear dimensionality reduction
- Only the most prominent activation is transmitted to the next layer
- More advanced pooling mechanisms exist CapsuleNets [Sabour, Frosst and Hinton 2017]

Putting it all together

Putting it all together

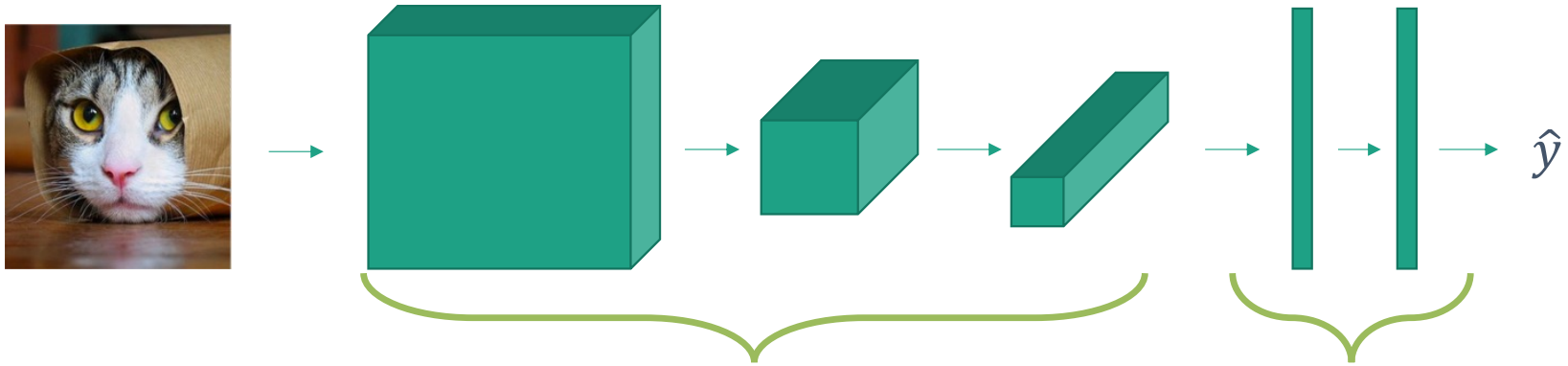
- We've covered the key operations and concepts in CNNs — convolution with strides and padding, and pooling — how do these operations come together in the overall architecture of a CNN?
- The architecture of a CNN is built by stacking these layers, where each layer extracts increasingly abstract features, starting from low-level patterns and progressing to higher-level features, like objects and scenes. Note that we typically use multiple filters/kernels at each layer.
- Let's look at an example:

Recognition network with pooling

Putting it together

Training set: $(x^{(1)}, y^{(1)})$ .... $(x^{(m)}, y^{(m)})$

Parameters: w, b



Cost function: $J = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}, y^{(i)})$

Use optimizer (e.g., gradient descent) to optimize parameters to minimize J

Essential blocks lead to powerful algorithms

Convolutional layers and pooling are the essential blocks

They have been used to create complicated networks

LeNet-5 in 1998

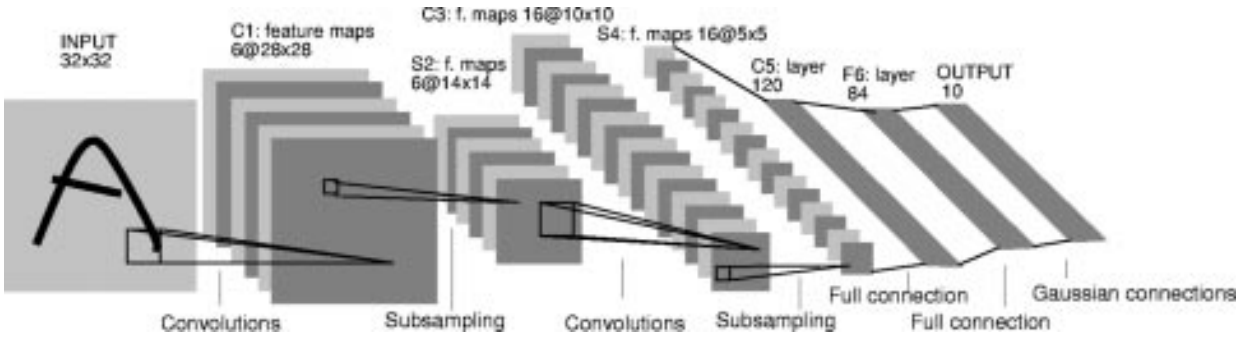


Fig. 2. Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.

[Lecun, Bottou, Bengio and Haffner; Gradient-based learning applied to document recognition; 1998]

Then silence for a long time

Why silence

Models had too many parameters

They overfit for small datasets

We did not have very large datasets

Even for large sets, we did not have enough computation power to train the models until…

General purpose Graphical Processing Units (GPUs)
Allowed parallel processing

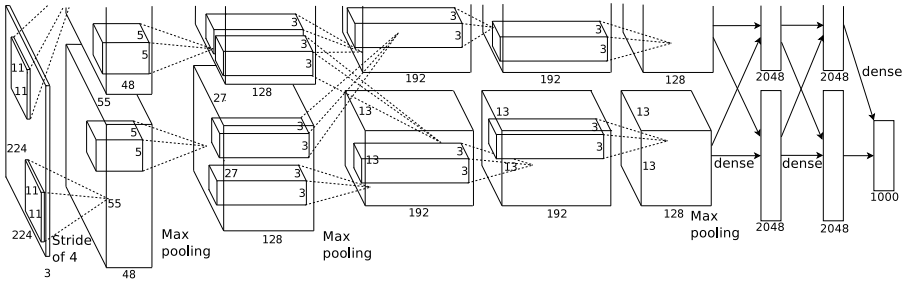
Then in 2012 - AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

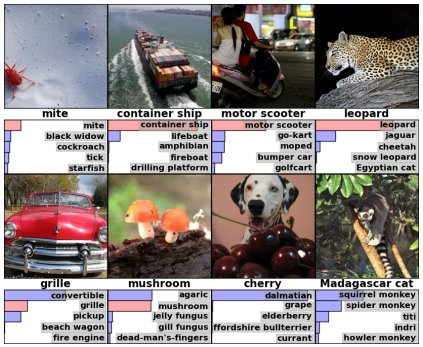
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

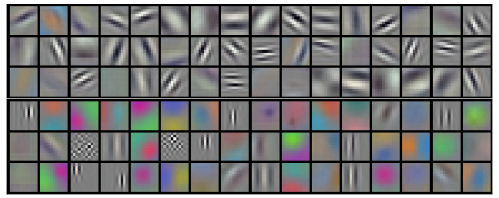


Network

Krizhevsky et al. almost halved the error rate in the ImageNet challenge

A simple CNN





First layer filters 11x11

A word about the ImageNet challenge

ImageNet: A Large-Scale Hierarchical Image Database

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei
Dept. of Computer Science, Princeton University, USA
{jiadeng, wdong, rsocher, jial, li, feifeili}@cs.princeton.edu

[CVPR 2009]

International Journal of Computer Vision
December 2015, Volume 115, Issue 3, pp 211–252 | Cite as

ImageNet Large Scale Visual Recognition Challenge

Authors Authors and affiliations

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei

Large scale object recognition challenge started in 2010

1.2 million training images

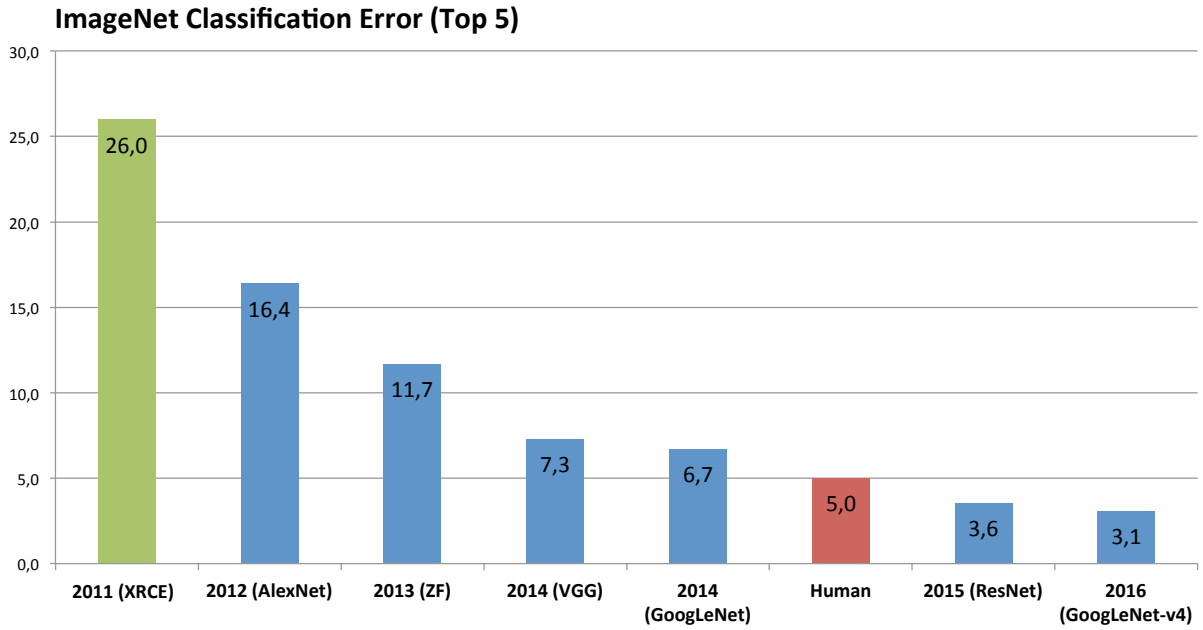
1000 object categories

200k validation and test images

2017 – 3 challenges:

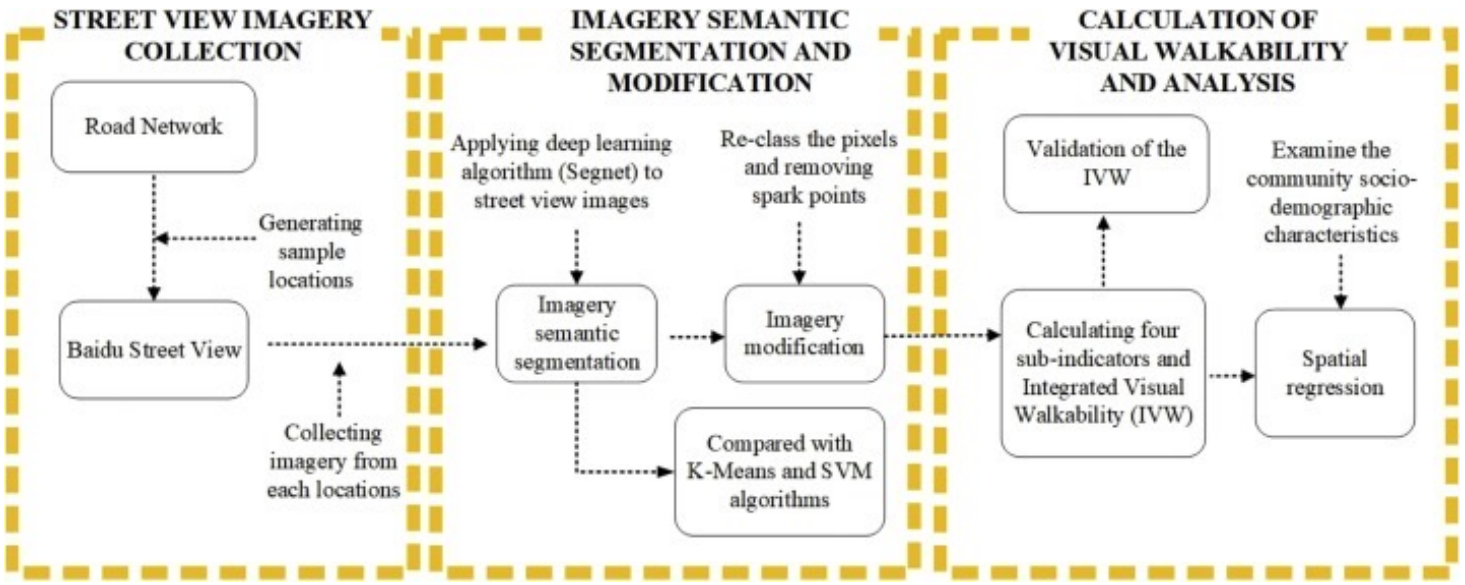
- Object localization
- Object detection
- Object detection from video

Historical evolution of the ImageNet Challenge



CNNs for Geospatial Research

Example 1: Visual walkability



Zhou et al., 2019

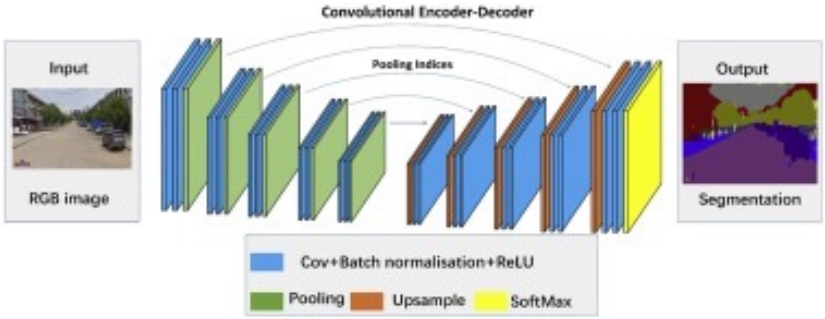
Visual walkability

- These indicators are derived from street images and used to build the IVW index.

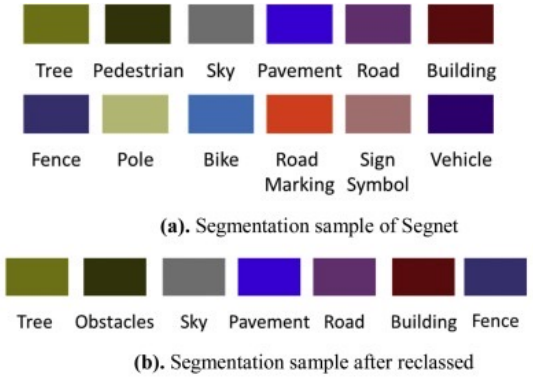
Indicators	Definition	Formula	Explanation
Psychological Greenery	Extent to which the visibility of street vegetation can influence pedestrian psychological feelings	$G_i = \frac{\left(\sum_1^6 Tn\right)}{6*Sum}$	T_n is the number of tree pixels; Sum is the total pixel number;
Visual Crowdedness	Extent to which the visibility of obstacles can influence pedestrian experiences	$C_i = \frac{\left(\sum_1^6 Cn\right)}{6*Sum}$	C_n is the number of obstacles pixels; Sum is the total pixel number;
Outdoor Enclosure	How the room-like outdoor space is (the ratio of vertical objects to horizontal features)	$S_i = \frac{\sum_1^6 Bn + \sum_1^6 Tn}{\sum_1^6 Pn + \sum_1^6 Rn + \sum_1^6 Fn}$	B_n is the number of building pixels; T_n is the number of tree pixels; P_n refers to the number of pavement pixels; R_n refers to the number of road pixels; F_n refers to the number of fence pixels
Visual Pavement	Psychological impacts of the proportion of road and sideway on pedestrian experience	$D_i = \frac{\sum_1^6 Pn + \sum_1^6 Fn}{\sum_1^6 Rn}$	R_n refers to the number of road pixels; P_n refers to the number of pavement pixels; F_n refers to the number of fence pixels;
Integrated Visual Walkability (IVW)	Integrated Visual Walkability index	IVW=(G-level + C-level + S-level + D-level)*5	

Zhou et al., 2019

Visual walkability



(a). Structure of SegNet



(a). Segmentation sample of Segnet

(b). Segmentation sample after reclassed

(c). Street view image and corresponding labelled training image

Zhou et al., 2019

Example 2: measuring urban inequality

Measurements of inequalities: based on census costly and infrequent

High spatial and temporal resolution – challenging even for data-rich cities and countries

Opportunities to leverage emerging large-scale datasets: street-level images, satellite data, etc.

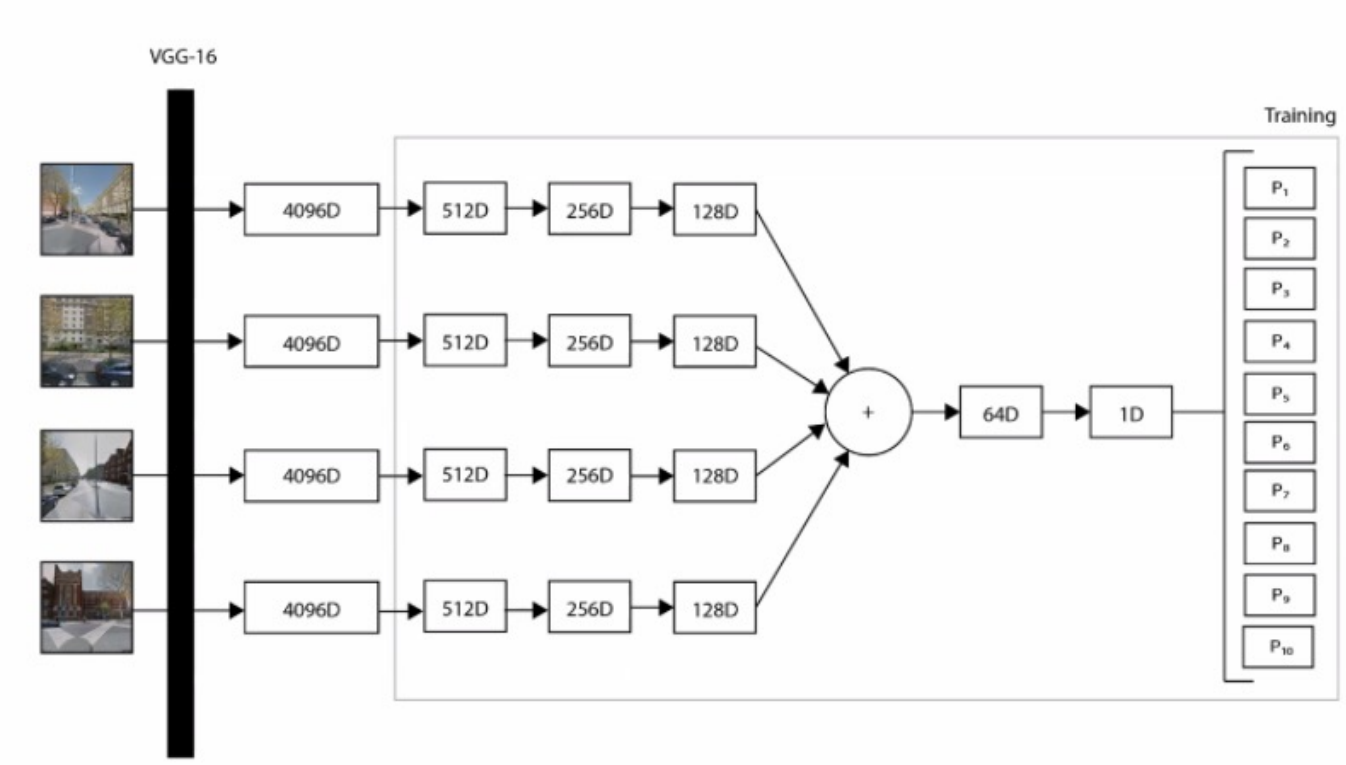
Example 2: measuring urban inequality

Case study: using street images to predict the decile of index of multiple deprivation (IMD)

~525,000 images from ~180,000 postcodes in London

Suel et al., 2019

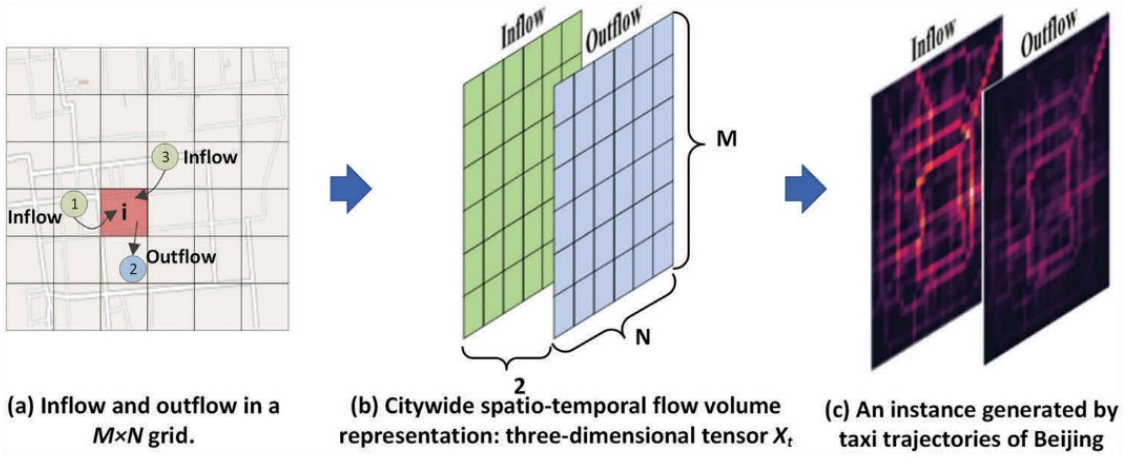
Example 2: measuring urban inequality



Suel et al., 2019

Example 3: spatio-temporal prediction of traffic flows in urban cells

The idea is to transform traffic flow into image-shape data and then apply CNN to predict flows.



(a) Inflow and outflow in a $M \times N$ grid.

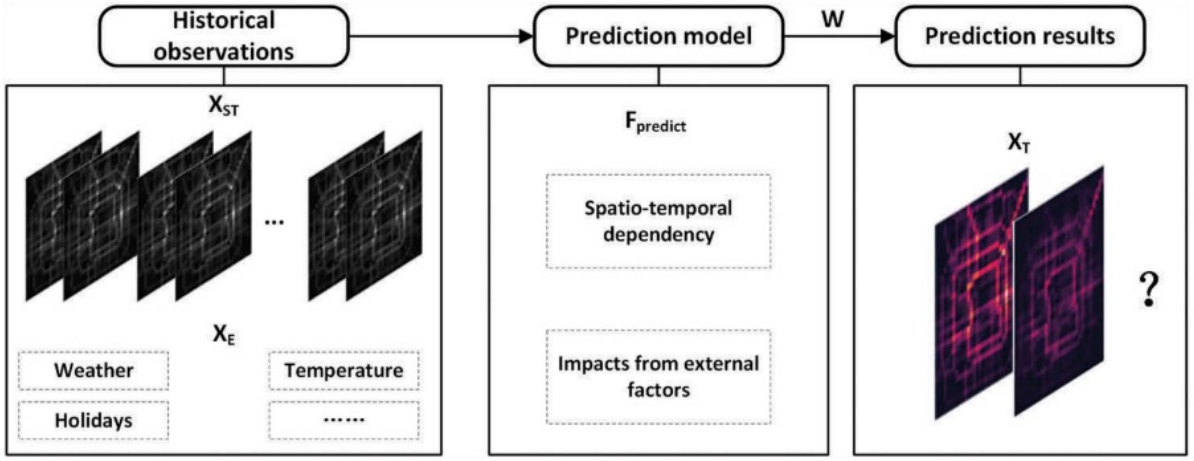
(b) Citywide spatio-temporal flow volume representation: three-dimensional tensor X_t

(c) An instance generated by taxi trajectories of Beijing

Ren et al., 2020

Example 3: spatio-temporal prediction of traffic flows in urban cells

The idea is to transform traffic flow into image-shape data and then apply CNN to predict flows.



Ren et al., 2020

Summary – deep learning for images

DL provides powerful CNN-based tools to extract patterns and knowledge from images for varied purposes.

Most models are open-source and easy to reuse. Most models have similar structure (incl. Conv, Pooling, FC layers).

For most applications, you don’t need to design a CNN from scratch. It is advisable to test classical models and adjust the models if needed.

Potential Problems with CNN

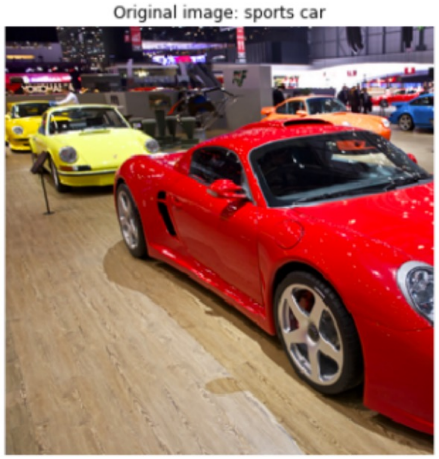
Garbage in, garbage out

If labels are not reliable, the model trained would be problematic.

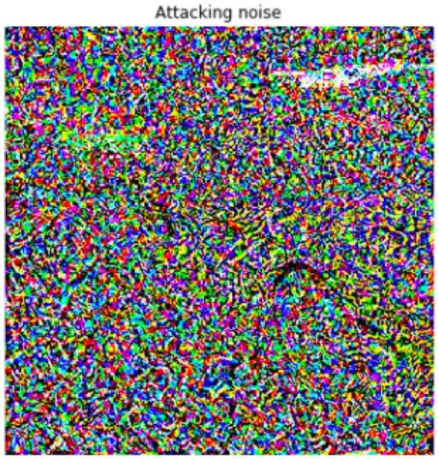
“A flock of birds flying in the air”

“A group of flowers in a field”

Adversarial Attacks



“A car”



“A toaster”

- You can fool a neural network classifier by adding noise to data.
- The noise might not be even visible to a human eye.

Adversarial Attacks

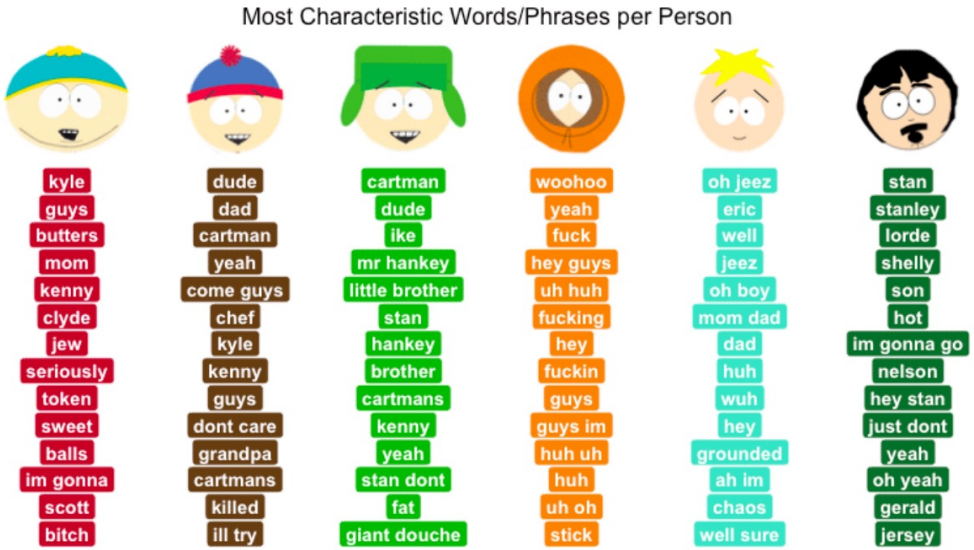
- Stop sign is no longer recognised by a neural network after a few stickers are placed on it.

Deep learning for text data

Keyword Detection

- Keyword extraction identifies the most important tokens or n-grams within a piece of text

- Input: a piece of text
- Output: list of most important tokens (words), optionally grouped by topics



Sentiment Analysis

- Sentiment Analysis assigns a sentiment score to each word in a piece of text

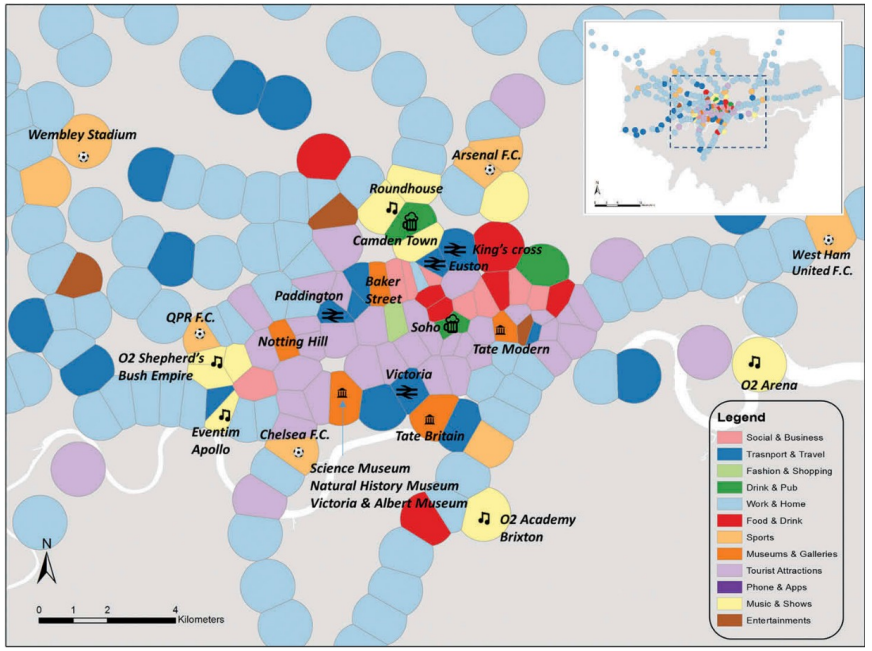
I love data science, and our teachers are awesome	+4 (Strongly positive)
Beer is disgusting, why do people even like it?	-1 (Weakly negative)
It's so great that my train is late every single day	+1 (Weakly positive)

- Input: a piece of text
- Output: a piece of text with a sentiment score assigned to each word

Topic Extraction

- Topic modelling extracts topics from a collection of documents.
- Input: a collection of documents
- Output: a list of topics with words that belong to them

Example:
Dominant topics on Tweets around the stations



Lai, J., Cheng, T., and Lansley, G. (2017) Improved targeted outdoor advertising based on geotagged social media data. Annals of GIS, ,1–14. https://doi.org/10.1080/19475683.2017.1382571

Next-Word Prediction and Large Language Models (LLMs)

- LLMs (like GPT, BERT) are trained on massive datasets to predict the next word or missing words in a sentence.

- "*Paris is the capital of* ___" → The model predicts "France", understanding geography and context.

- Geospatial applications – extract location information from text, urban forecasting for land-use change, next activity predictions …

Summary

- Unstructured data are common in the big data era, including images, text, etc.
- Deep learning provides powerful tools for utilising unstructured datasets. Most tools are open-source and ready to use.
- For applications, you don't (always!) need to fully understand the mathematics of these DL models.
- We cover the foundation and classical models of CNN here. More details can be found in the Coursera module "Convolutional Neural Networks".

https://www.coursera.org/learn/convolutional-neural-networks

Thanks!

Additional slides

Why convolutions

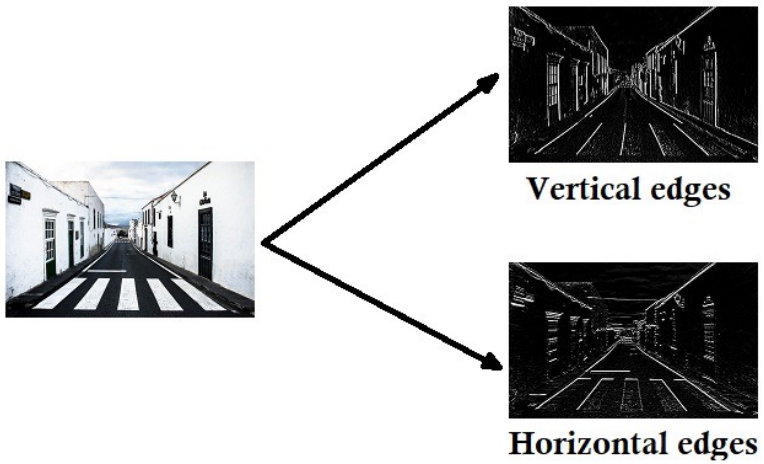
Images can be large. CNN is fast and accurate to work with images.

Lower number of parameters

Why convolutions

Parameter sharing

A filter is a feature detector. A filter that is useful in one part of the image is probably useful in another part of the image.

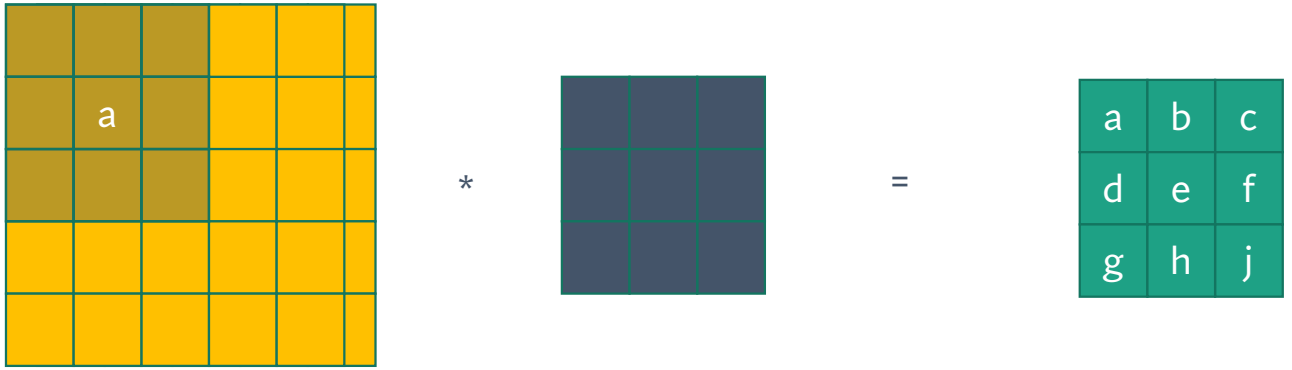


E.g. a vertical edge filter is useful in many parts of this image

Why convolutions

Sparsity of connections

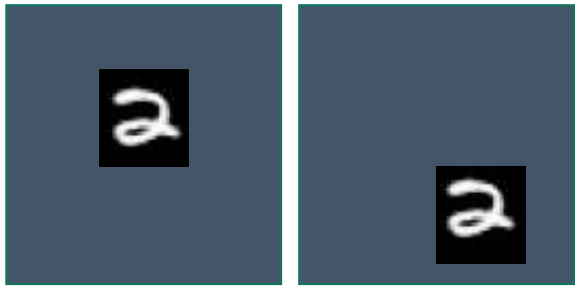
In each layer, each output value depends only on a small number of inputs.



Why convolutions

Translation invariance

If a pattern/object is translated, it is still identifiable by CNN.



CNN would detect the digit ‘two’ object regardless of its location.