Machine Learning
Introduction to CNNs

Week 8

Barbara Metzler

Research Associate in Computer vision and remote sensing
The Alan Turing Institute, London
bmetzler@turing.ac.uk
@abarbarame

Outline

- Part 1
  - Image data and characteristics
  - Convolution
- Part 2
  - Convolutional Neural Networks (CNNs)
  - Transfer learning
- Part 3
  - Applied research example: DeepCluster

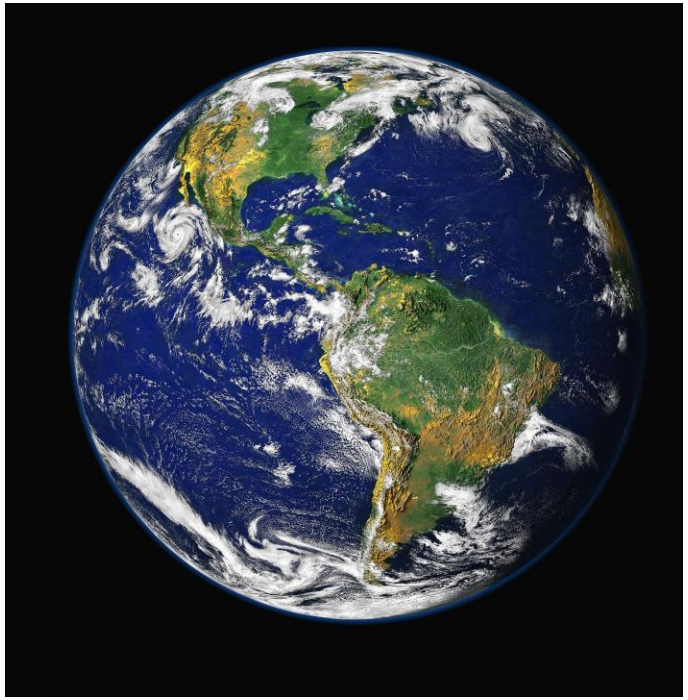
Part 1

Images

Slides adapted from Dr. Wenjia Bai and Dr. Dragana Vuckovic
Department of Computing & Brain Sciences

Digital images

- Digital images did not exist before 1975
- Digital cameras became popular in the late ‘90s / early 2000s
- Phones with cameras revolutionized the availability and sharing of digital images on the internet, from early 2000s
- Satellite images have been collected since the ‘70s
- Google StreetView launched in 2007
- And more!



The data structure of digital images - greyscale

- Grey scale images are two-dimensional matrices or grids
- Each entry in the matrix is a pixel
- Each pixel is a number which corresponds to a shade of grey

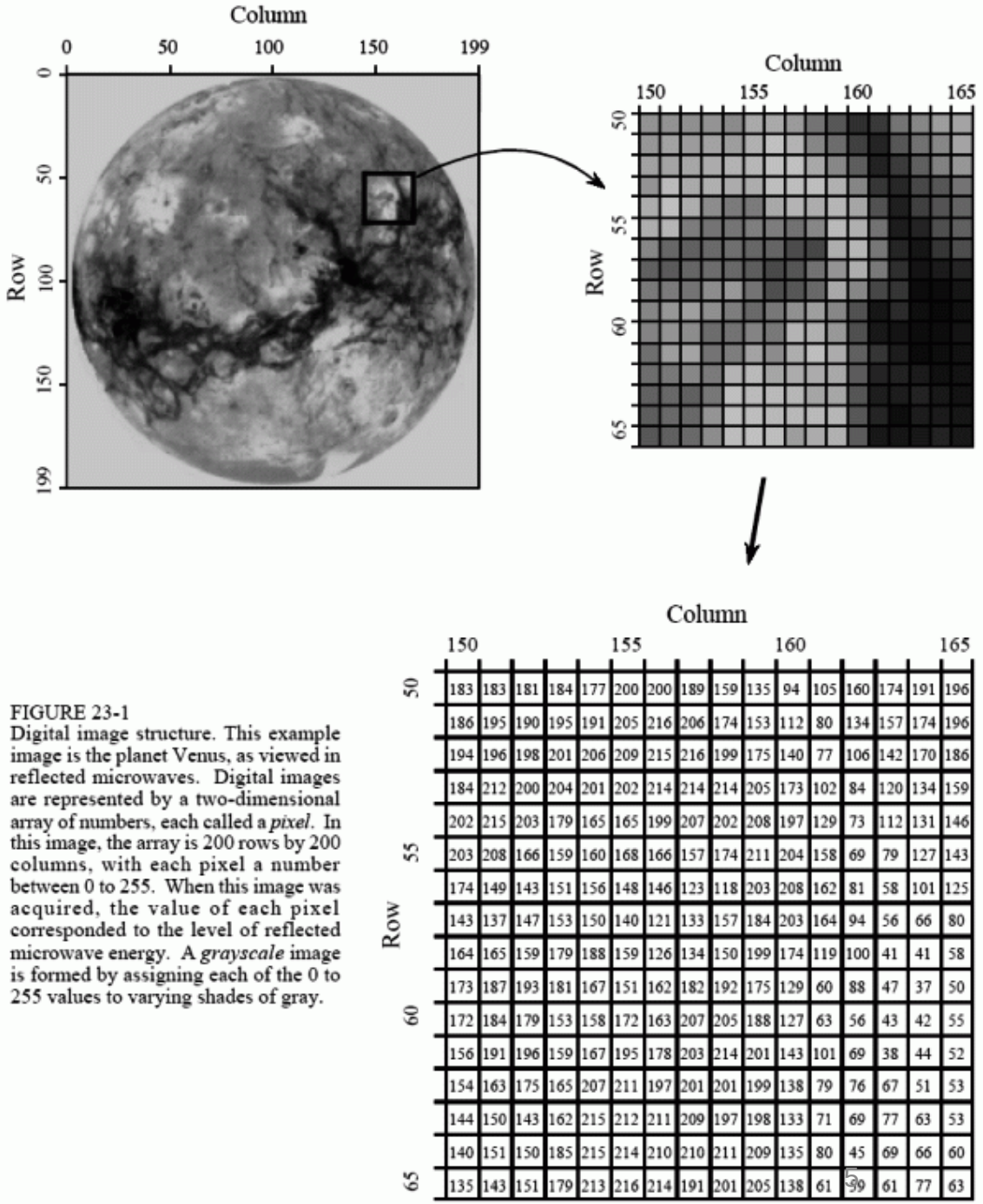
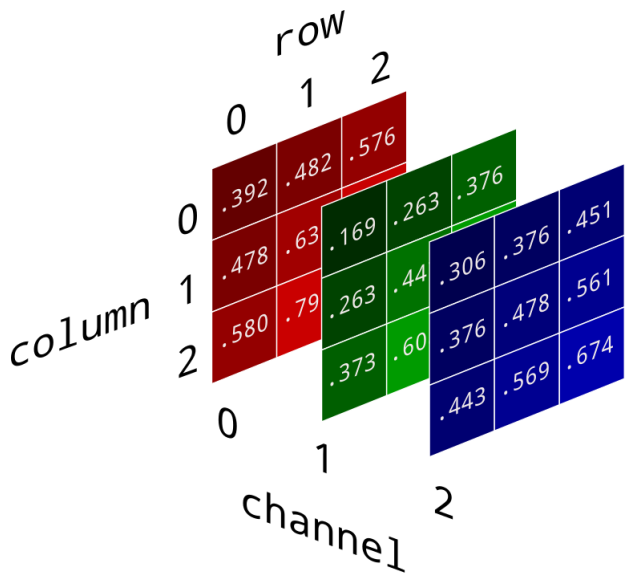


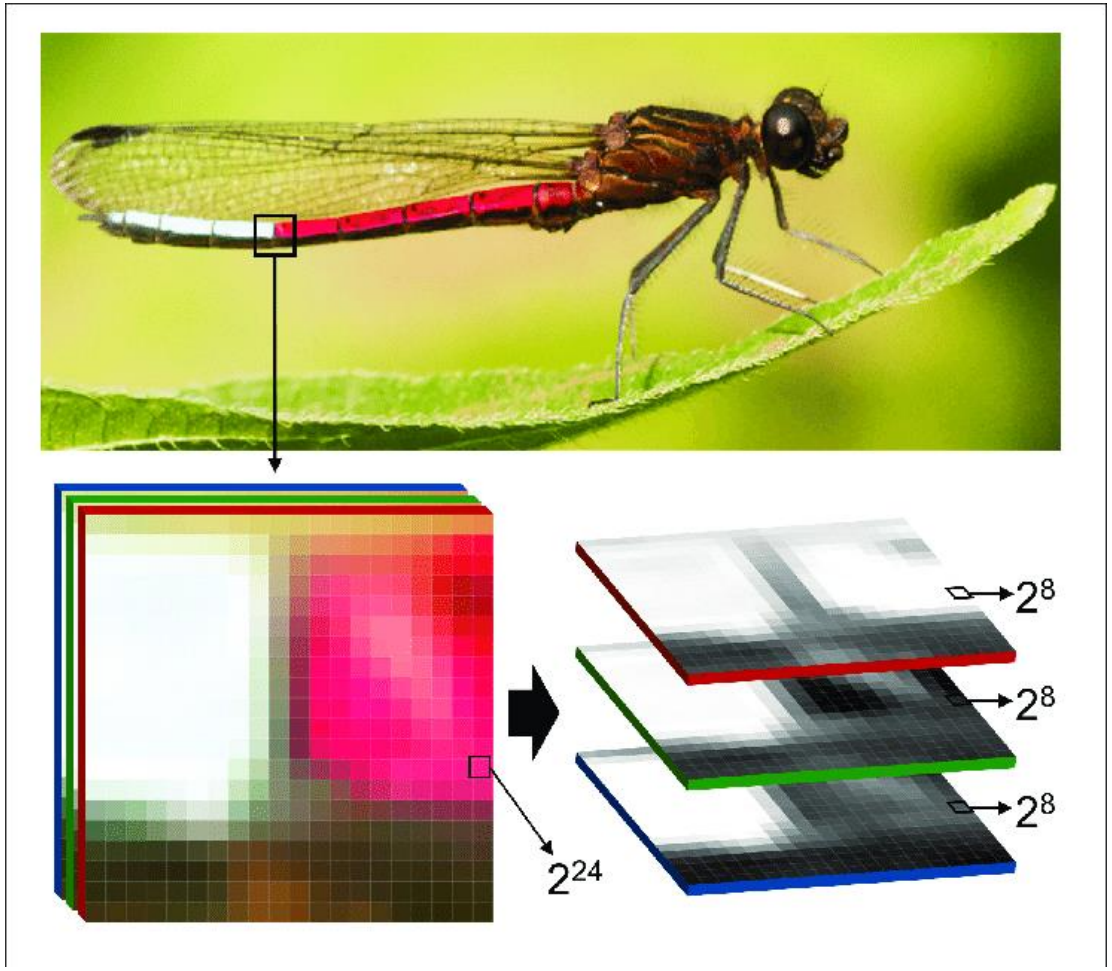
FIGURE 23-1
Digital image structure. This example image is the planet Venus, as viewed in reflected microwaves. Digital images are represented by a two-dimensional array of numbers, each called a *pixel*. In this image, the array is 200 rows by 200 columns, with each pixel a number between 0 to 255. When this image was acquired, the value of each pixel corresponded to the level of reflected microwave energy. A *grayscale* image is formed by assigning each of the 0 to 255 values to varying shades of gray.

The data structure of digital images – colour images

- A color image is represented by 3 matrices, one for each color channel: red, green and blue
- A single byte is normally used for each pixel in each channel allowing for

→ 256 x 256 x 256 = 16.8 million different resulting colors





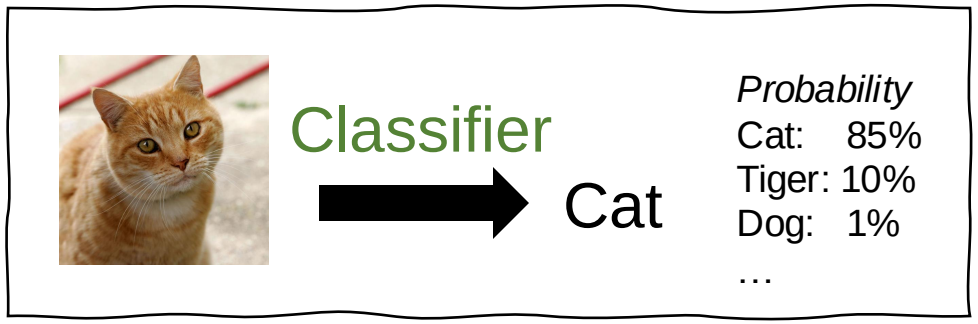
Lürig et al. (2021). Computer Vision, Machine Learning, and the Promise of Phenomics in Ecology and Evolutionary Biology.

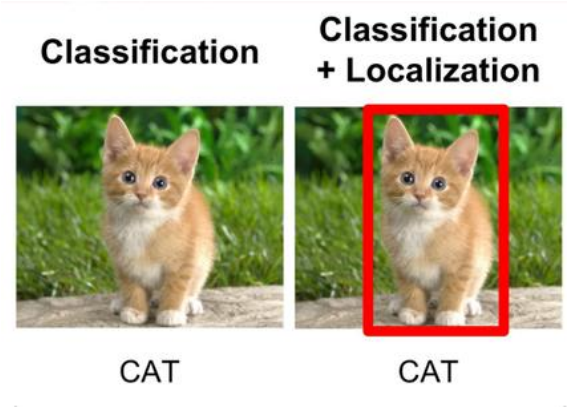
Types of problems

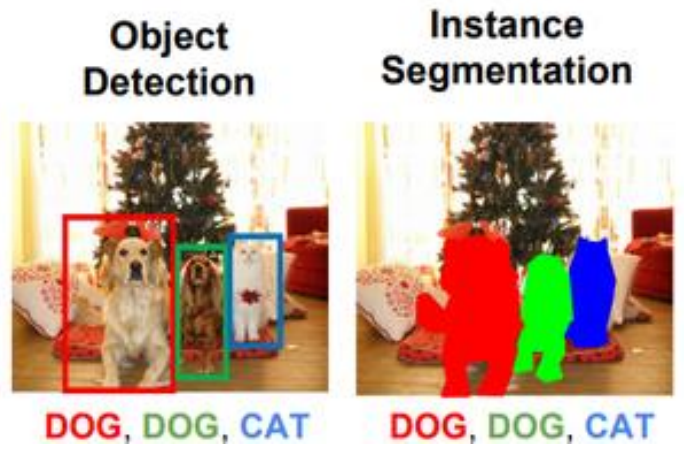
1. Classification: predict a label that defines the image
   - Classification + Localisation: identify a bounding box where the object is in the image

2. Object detection: predict a set of bounding boxes and labels for each object of interest in an image

3. Image segmentation:
   - color the area occupied by each classified object
   - label each area of the image including background







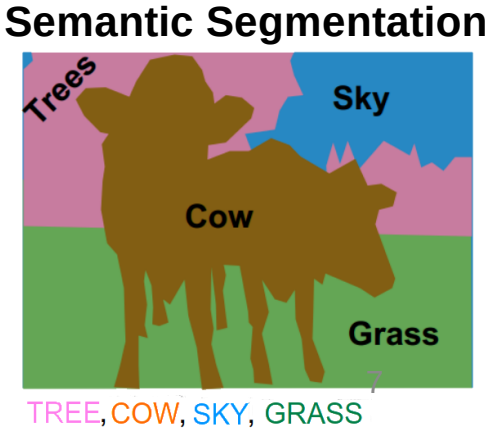


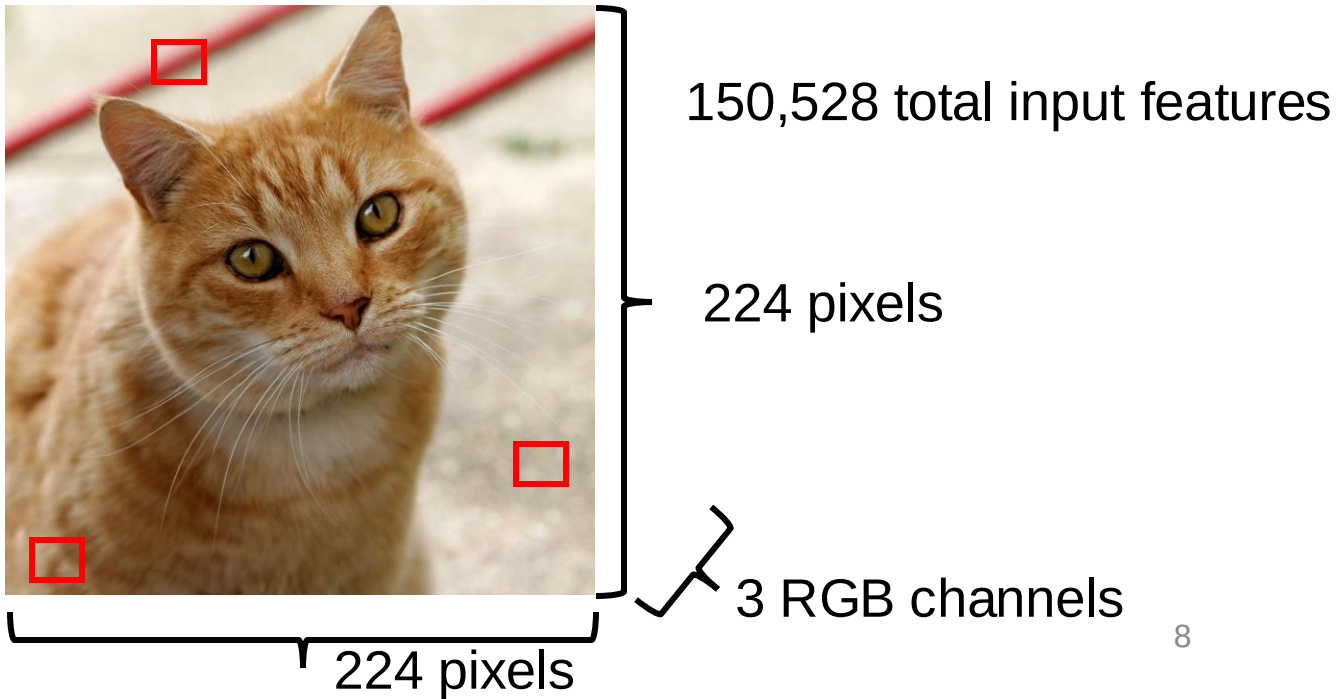
Image processing

If we were to build a fully connected neural network

- suppose you have an image of 224 x 224 pixels x 3 RGB channels = 150,528 input features – same number of parameters for **each neuron**
  - e.g. if we wanted to build the first layer with half the number of neurons (75,000)

  → 150,528 x 75,000 = 11,289,600,000 parameters

- sensitive to image shifts



Prior structural knowledge

Spatial locality:

The set of pixels we will have to take into consideration to find a cat will be near one another in the image

Translation invariance:

The pattern of pixels that characterizes a cat is the same no matter where in the image the cat occurs

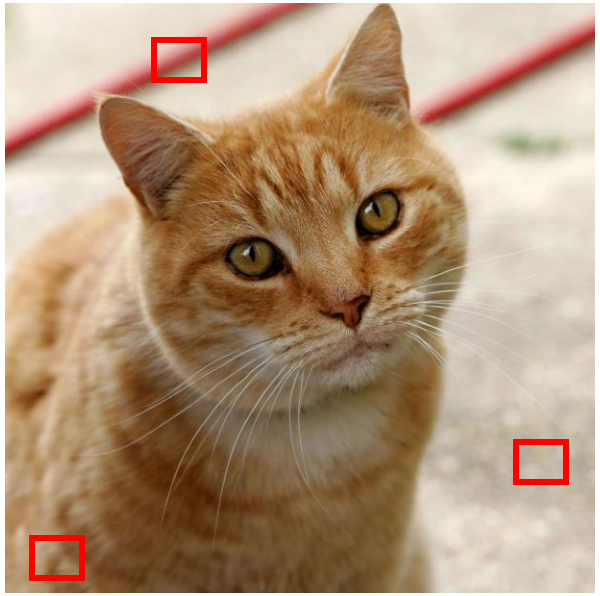
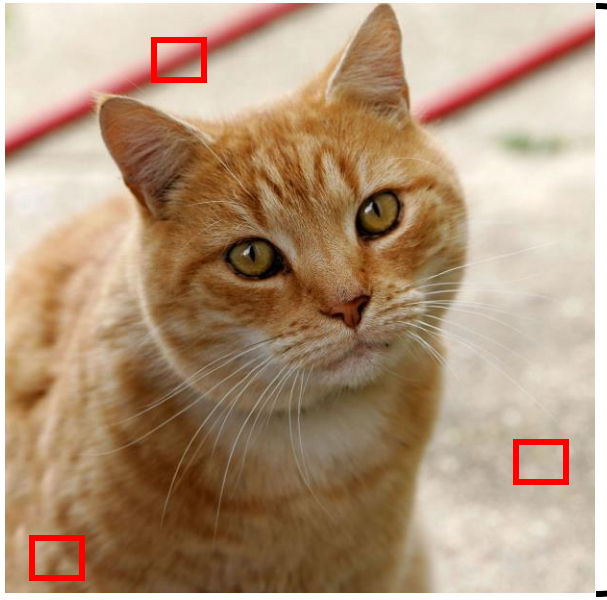


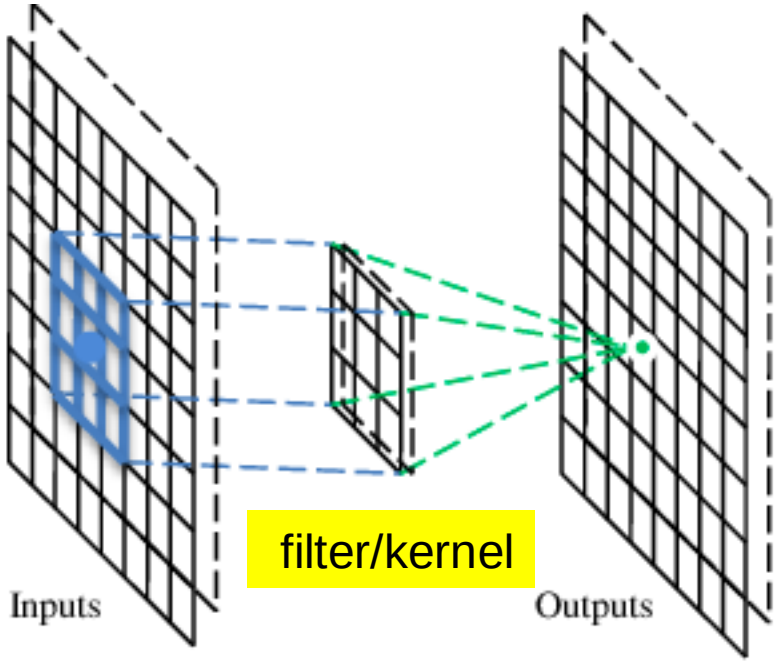
Image processing

- Convolutional Neural Networks overcome this high dimensionality problem by
  - reducing the size of the input
  - taking into account shifts
  - taking into account correlations between pixels



A convolutional layer

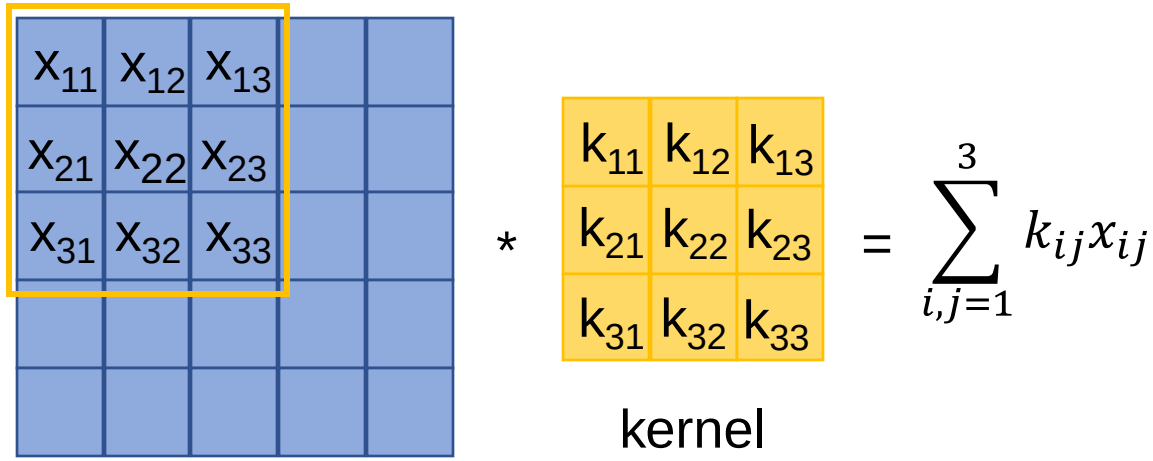
A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that apply a convolutional operation.



Convolution operation

Assume a single-color channel

- The convolution uses a sliding kernel of a certain size, 3x3 in the example below
- The kernel is a 2D set of weights or a matrix of weights
- Convolution is the dot product between the kernel and a portion of the image



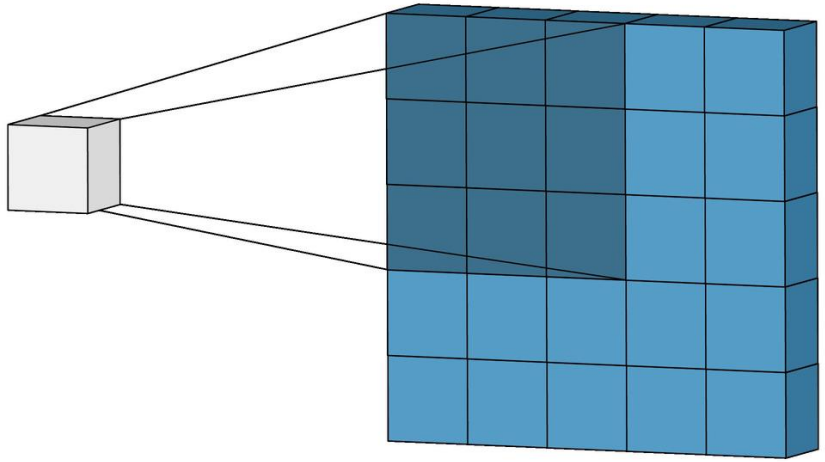


image credit: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

These are the network parameters to be learned!

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product →

3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

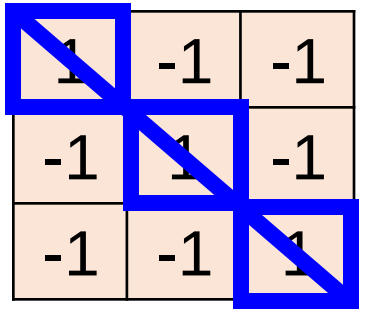
If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

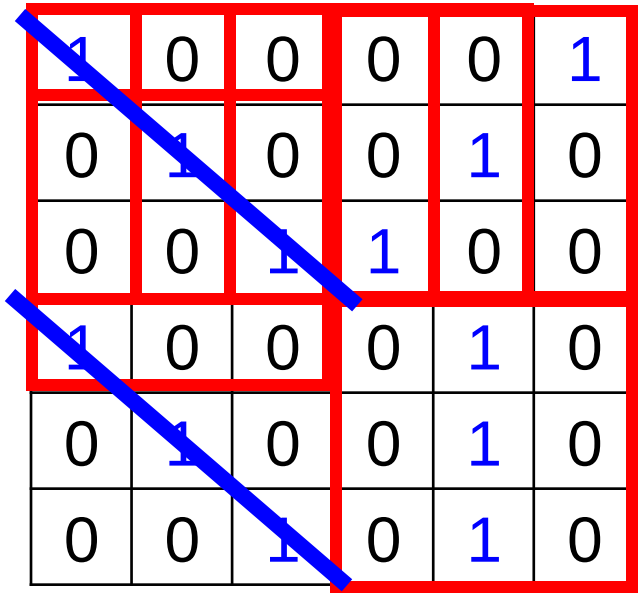
3 -3

Convolution

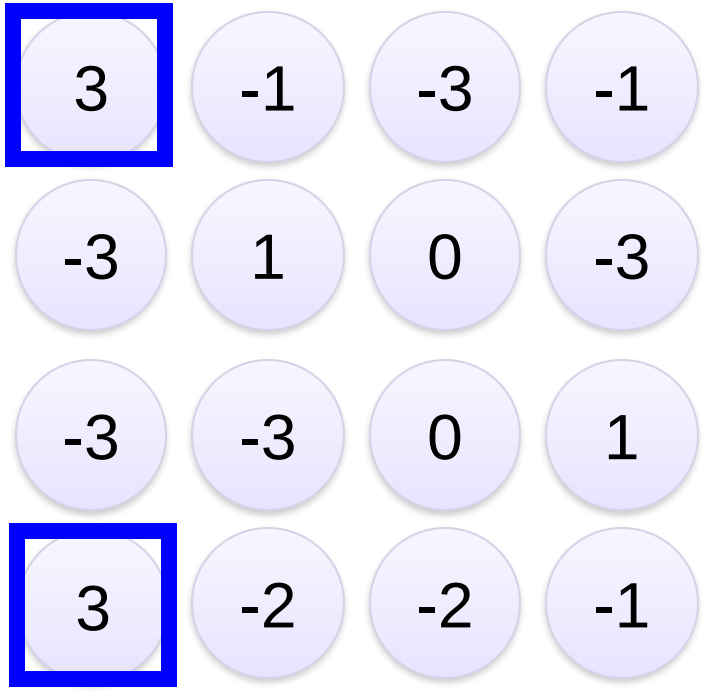


Filter 1

stride=1



6 x 6 image



Convolution

-1	1	-1
-1	1	-1
-1	1	-1

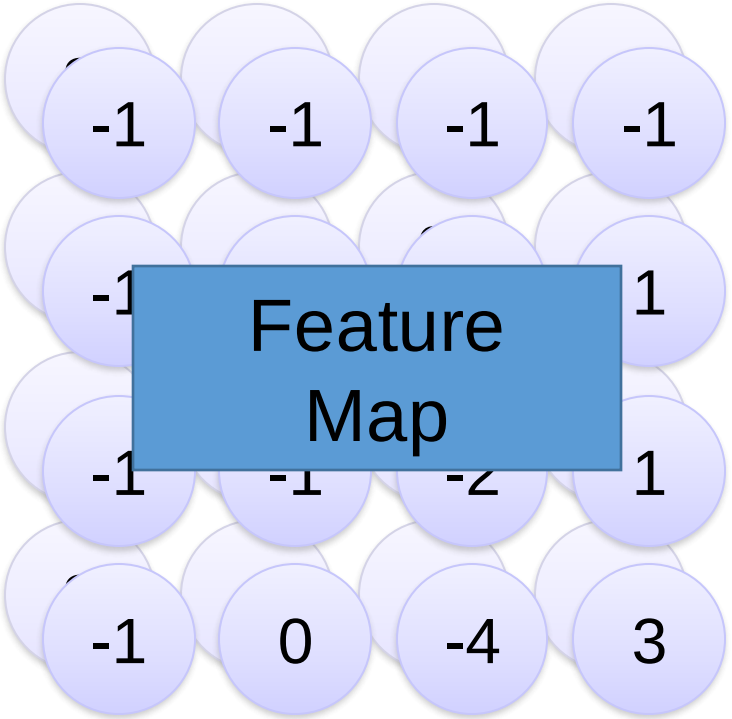
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

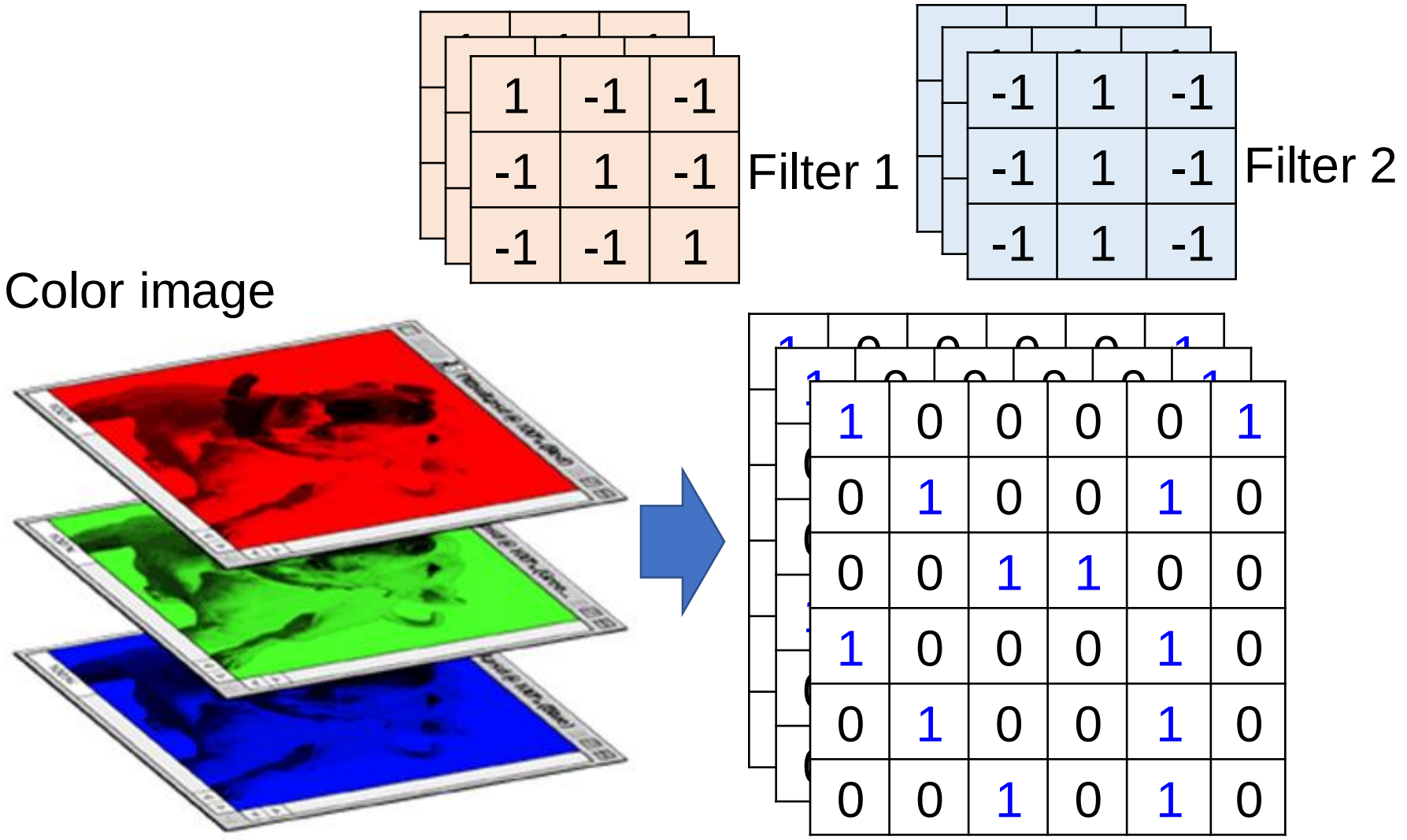
6 x 6 image

Repeat this for each filter



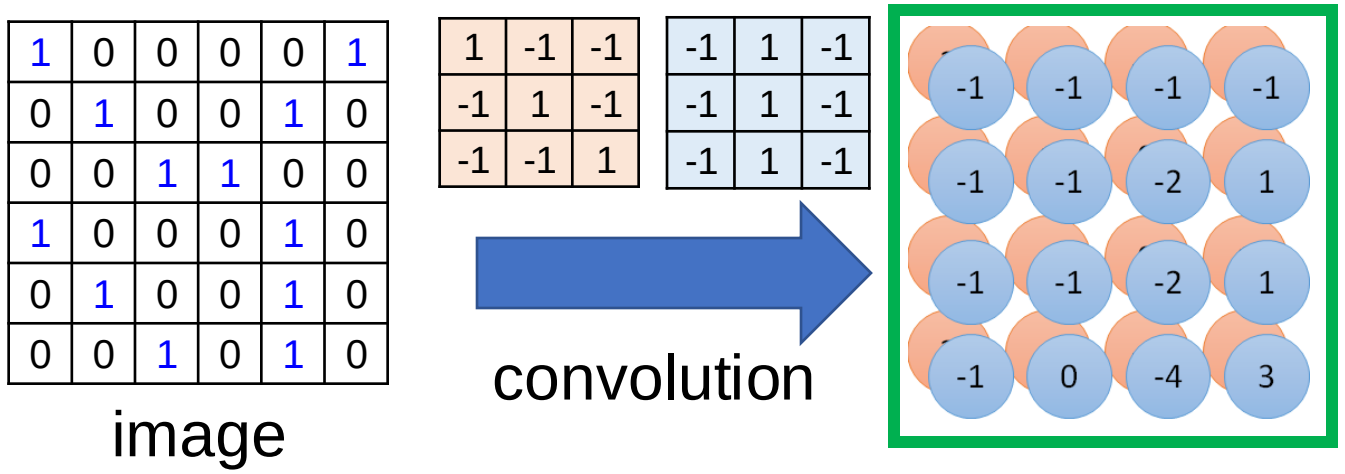
Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB

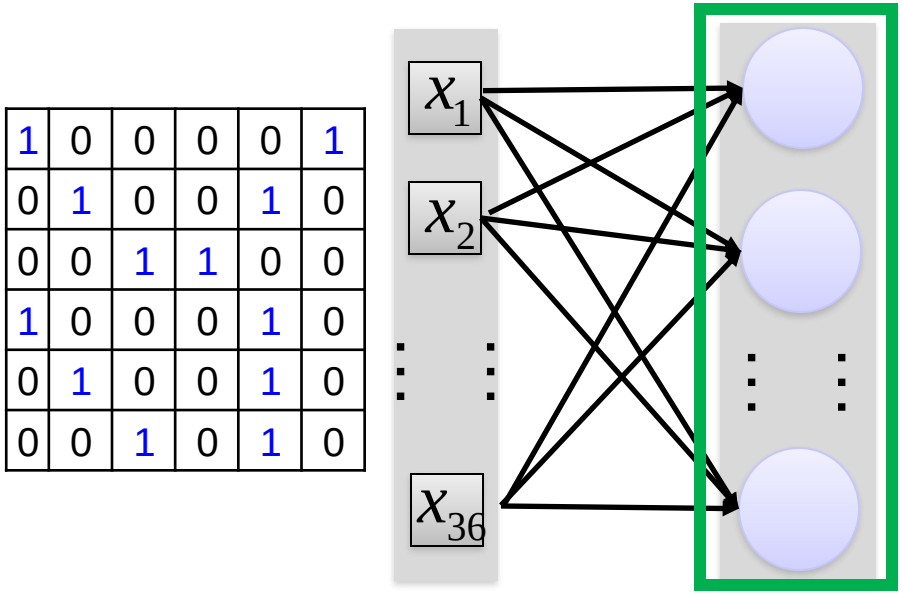


Convolution vs fully-connected

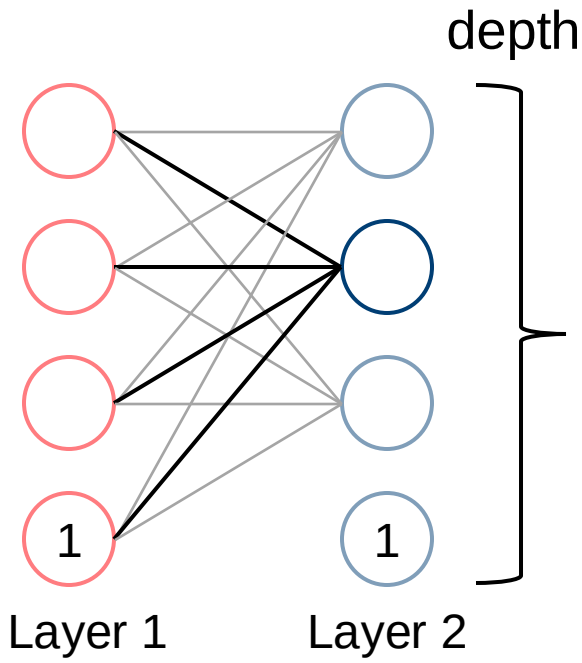
Convolutional layer



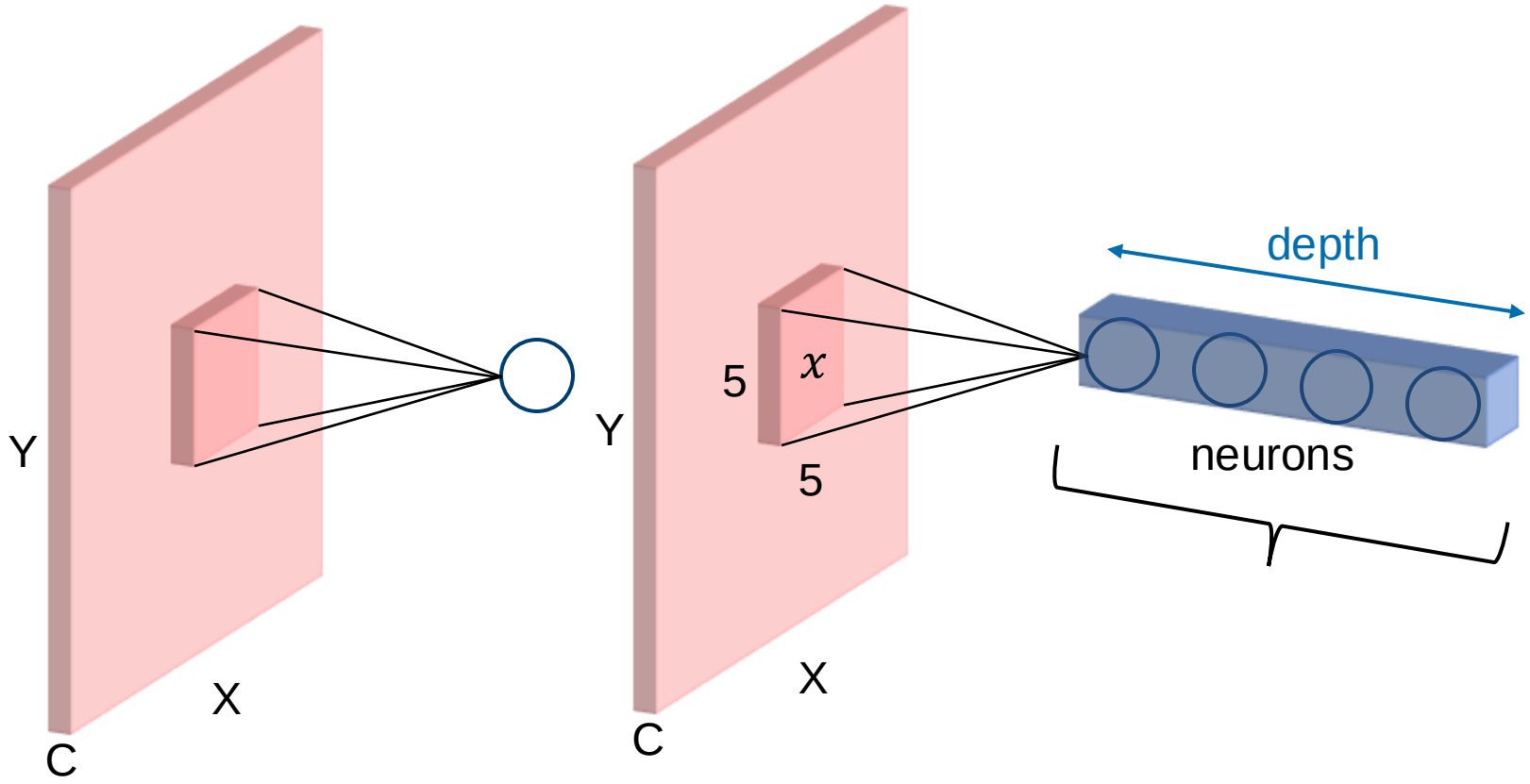
Fully-connected layer



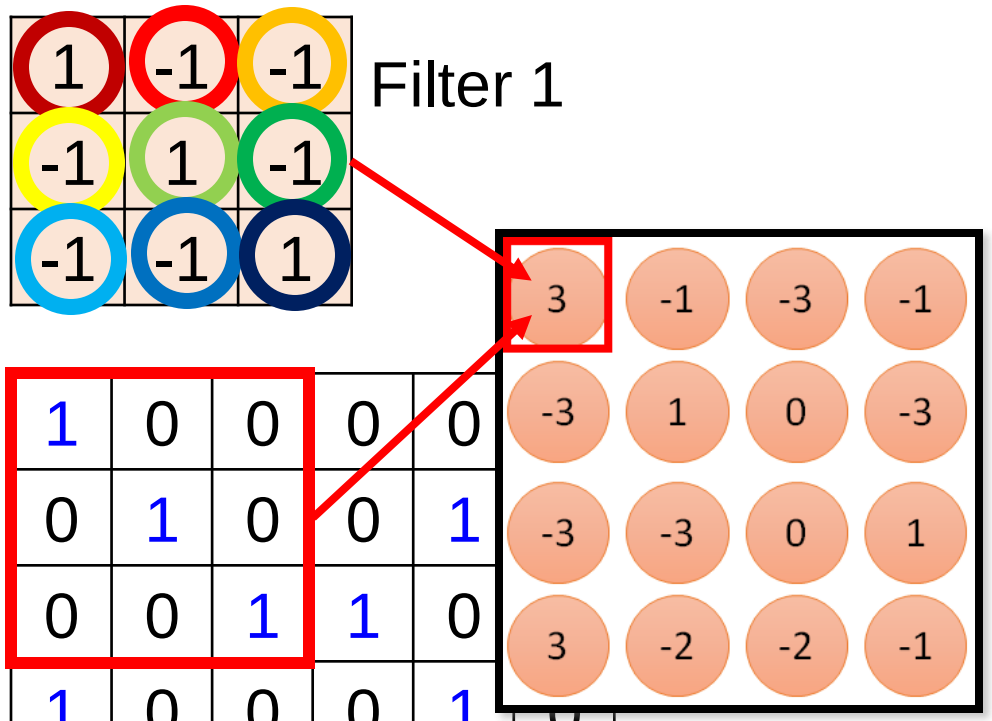
Convolutional neural networks



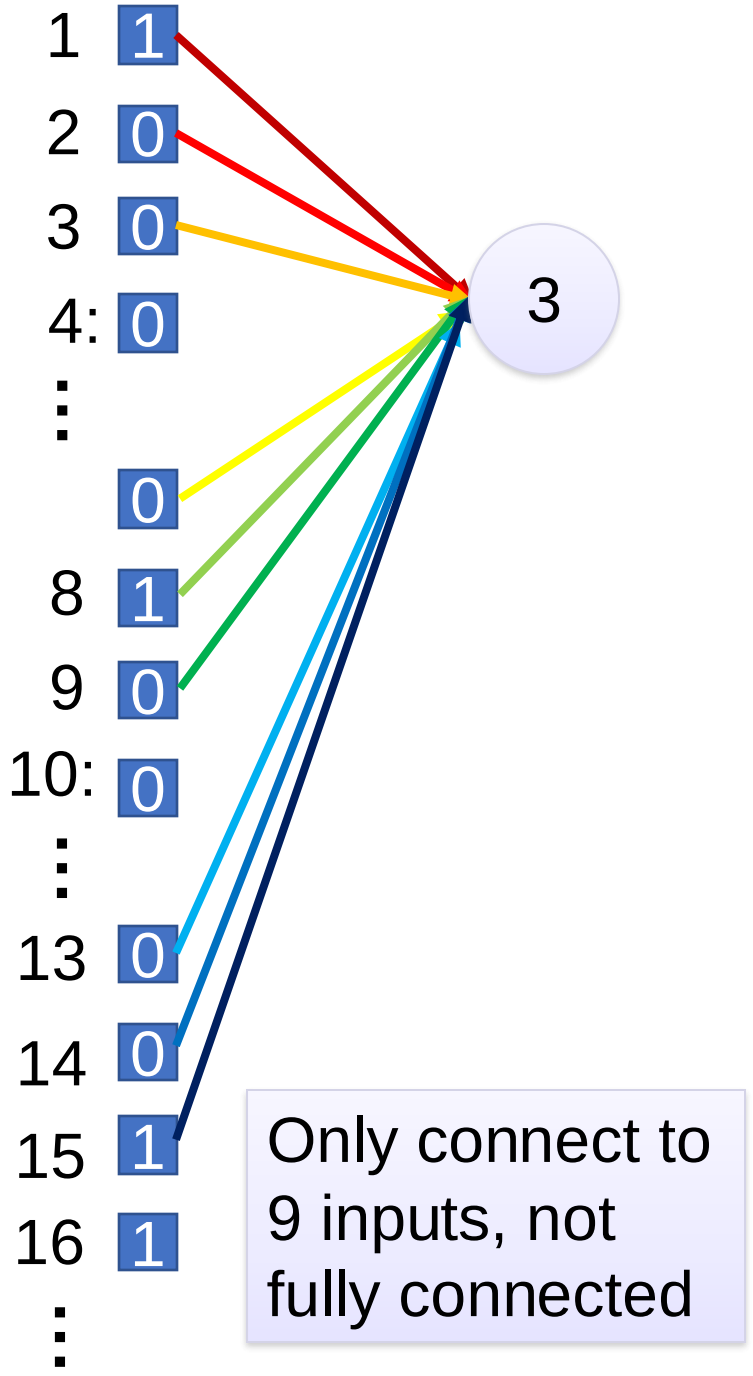
A fully connected layer in MLP.
A neuron in Layer 2 depends on all the neurons in the previous layer.



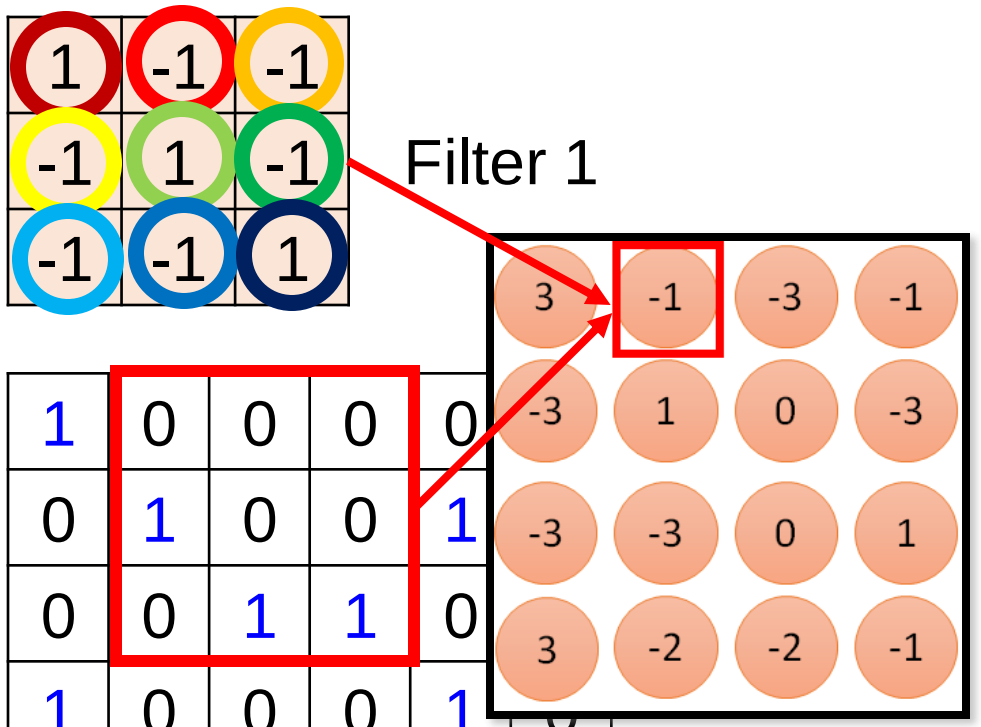
A convolutional layer in CNN.
A neuron in Layer 2 only depends on a small local region in the previous layer.



fewer parameters!

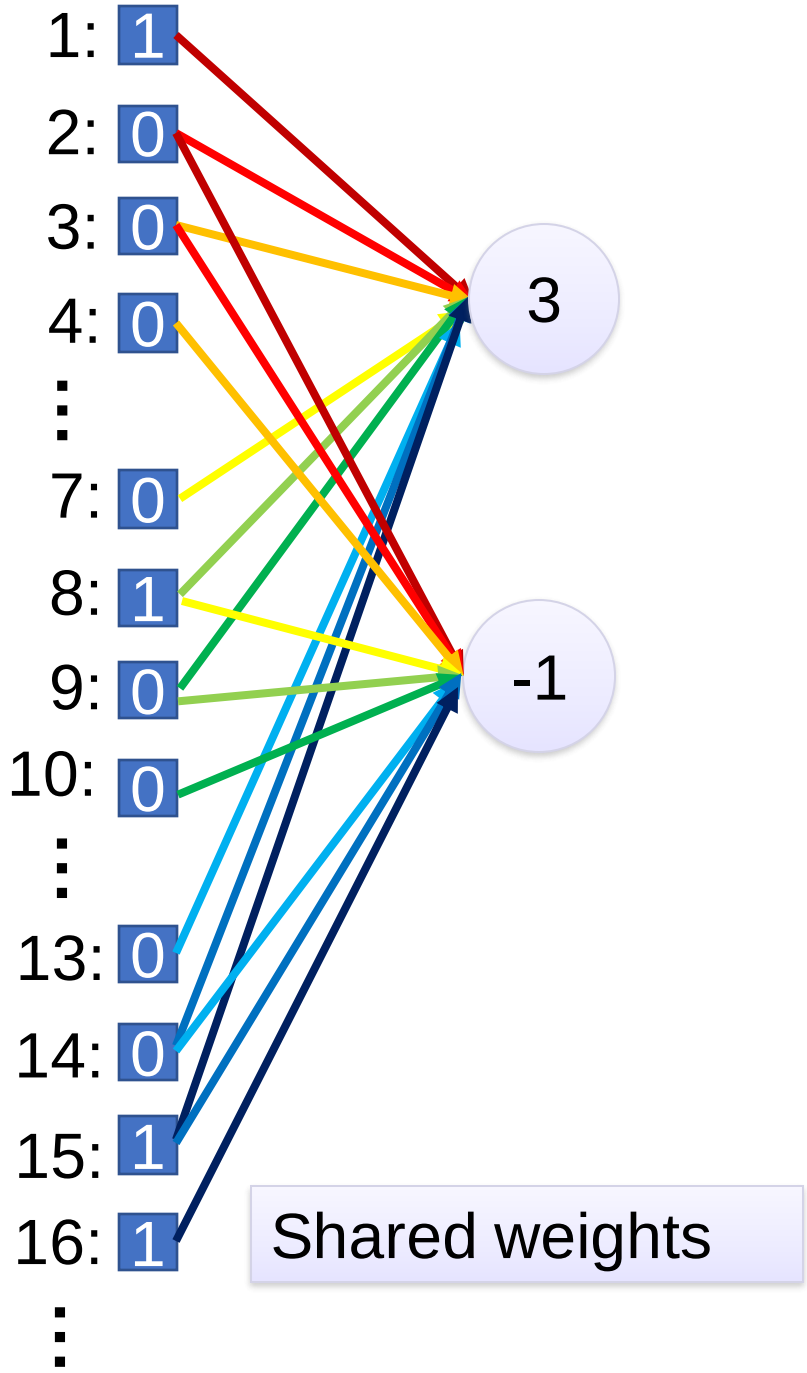


Only connect to 9 inputs, not fully connected



Fewer parameters

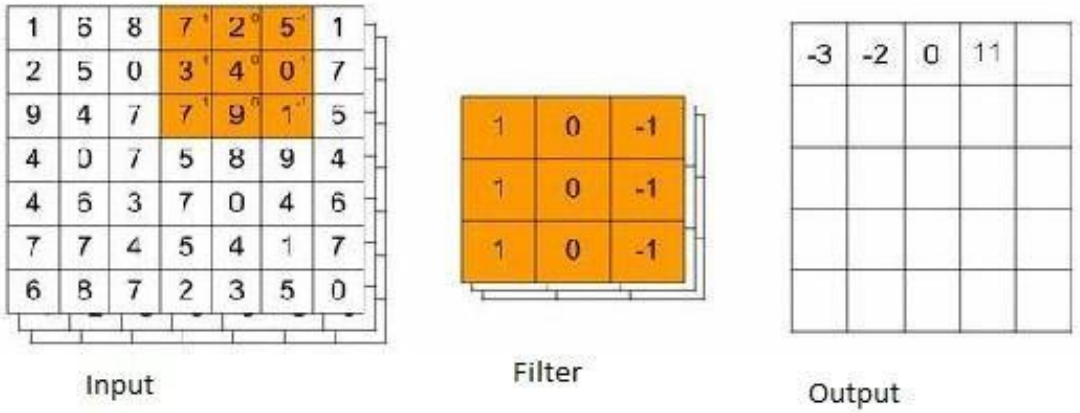
Even fewer parameters

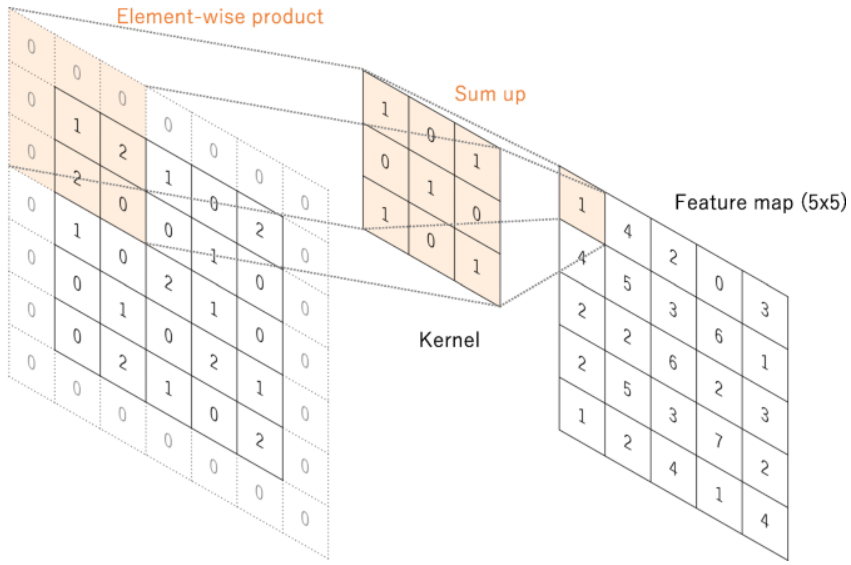


Shared weights

Convolution operation

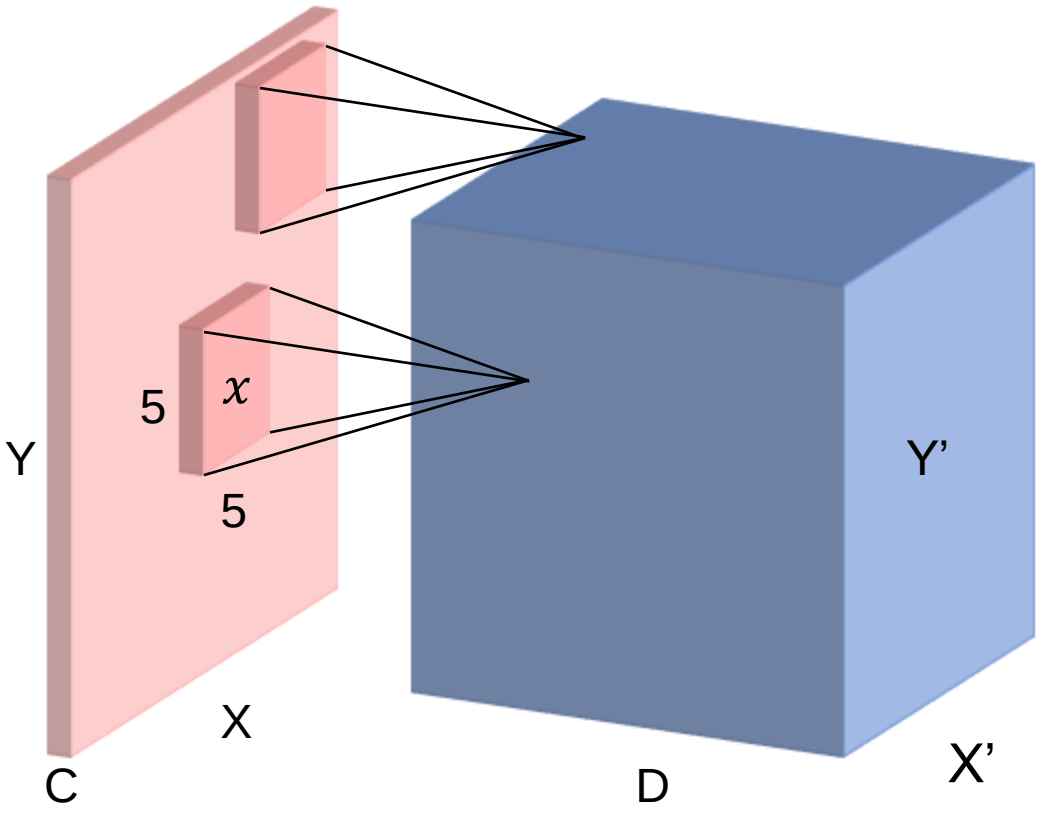
- The size of the convolution output depends on the slide size, which can be 1 step or more
- Sometimes an outside frame of 0s is added to ensure equal coverage (**zero padding**)





padding

Convolutional layer



- We can move the small window across the input image and get an output cube of size X' x Y' x D.
- If is use zero padding and a stride (s) of 1, X=X' and Y=Y'
- After convolution a bias term is also added:

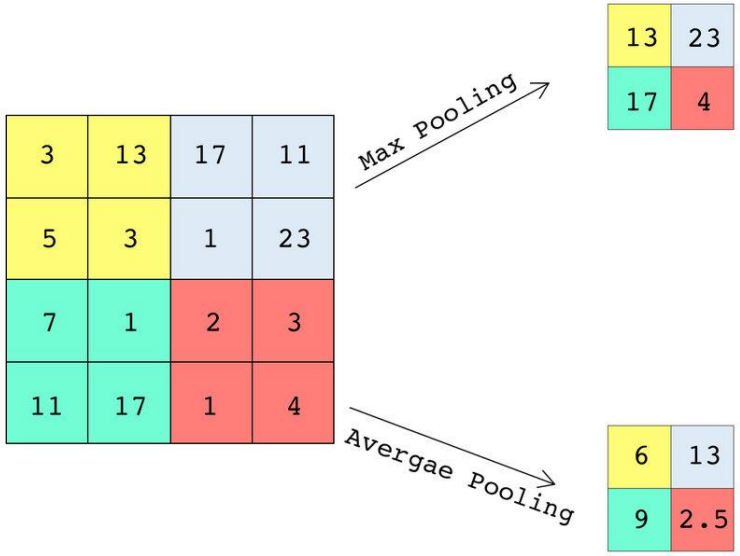
$$convolution\ output = \sum_{ijk} W_{dijk} x_{ijk} + b_d$$

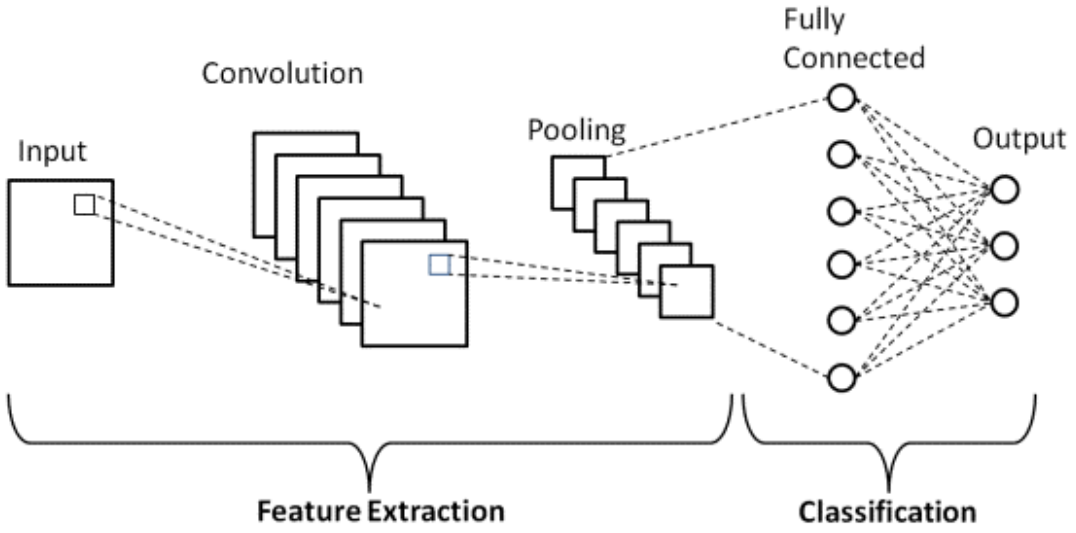
Output depth
Kernel width
Kernel height
Input depth

If more than one channel exist, each one is processed separately by three different kernels then the output is summed
X x Y x 3 → zero padding and s=1 → [X x Y x 1] x D (the depth)

Activation, pooling and dense layers

- Convolution layers are still linear operations
- An activation function is needed to introduce non-linearity (as previously)
- The convolution layer considers the correlation between pixels and reduces the number of parameters, but does not in general reduce the dimension of the output
- Pooling layers are introduced to reduce dimensions
- When dimension has been sufficiently reduced, standard densely connected layers are introduced





Network depth = number of layers

Training

- CNN are trained as traditional neural networks
- Backpropagation is used to find the optimal parameters
- Exploding or vanishing gradients might create issues

To control for exploding gradients:

- Clip by value
  - For each element of the gradient g, clip by a minimal value and a maximal value,

$$g_i = \begin{cases} v_{min}\text{ , if } g_i < v_{min} \\ v_{max}\text{ , if } g_i > v_{max} \\ g_i\text{ , otherwise} \end{cases}$$

- Clip by norm
  - Clip by the L2-norm of the gradient g,

$$g = \begin{cases} \frac{g}{||g||} v\text{ , if } ||g|| > v \\ g\text{, otherwise} \end{cases}$$

Activation functions

To avoid vanishing gradients:

- ReLU
- Leaky ReLU

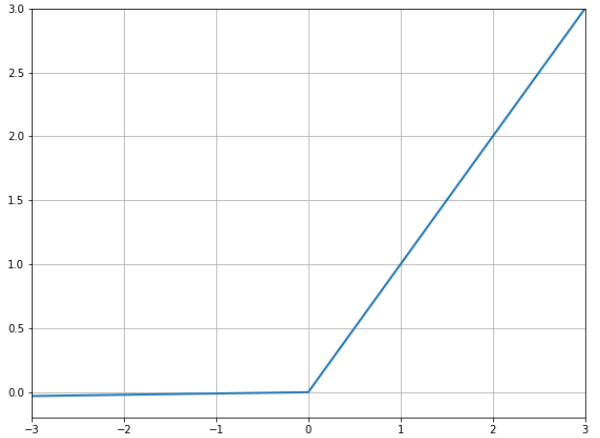
$$f(z) = \begin{cases} (0.01)z, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- Parameteric ReLU (PReLU): similar to leaky ReLU but learning the parameter a in training.

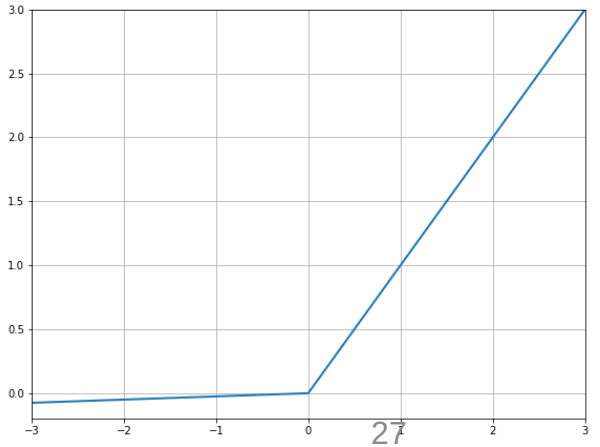
$$f(z) = \begin{cases} az, & z < 0 \\ z, & z \geq 0 \end{cases}$$

- They allow a small gradient when z is negative and may perform even better than ReLU

Leaky ReLU

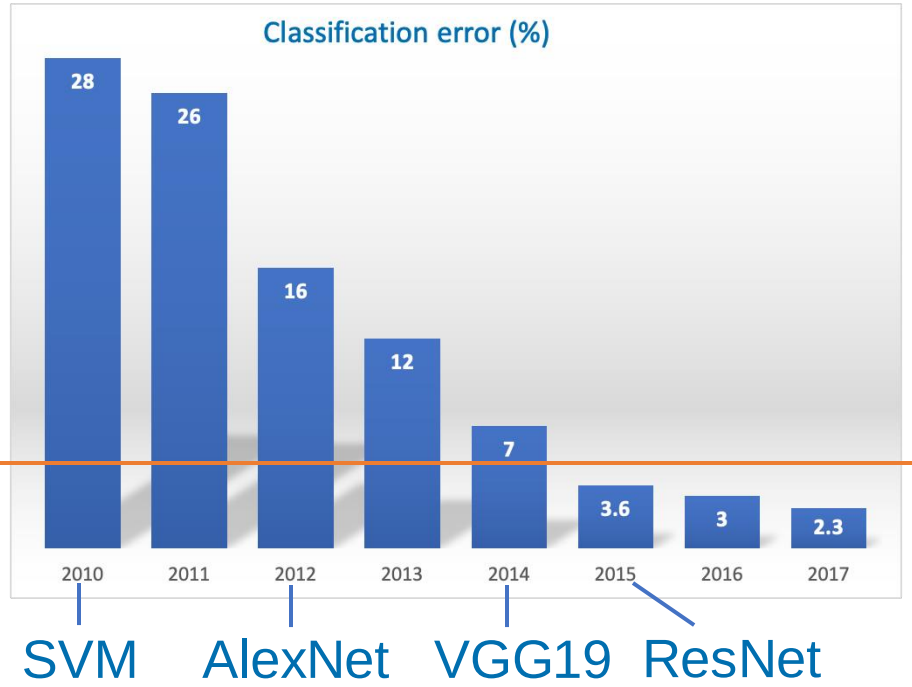


PReLU



ImageNet challenge

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
  - Train an algorithm that can classify 1.2 million images into 1,000 classes.
- The project was started and is led by Prof. Li, Professor of Computer Science at Stanford



human error rate ~5%

http://www.image-net.org

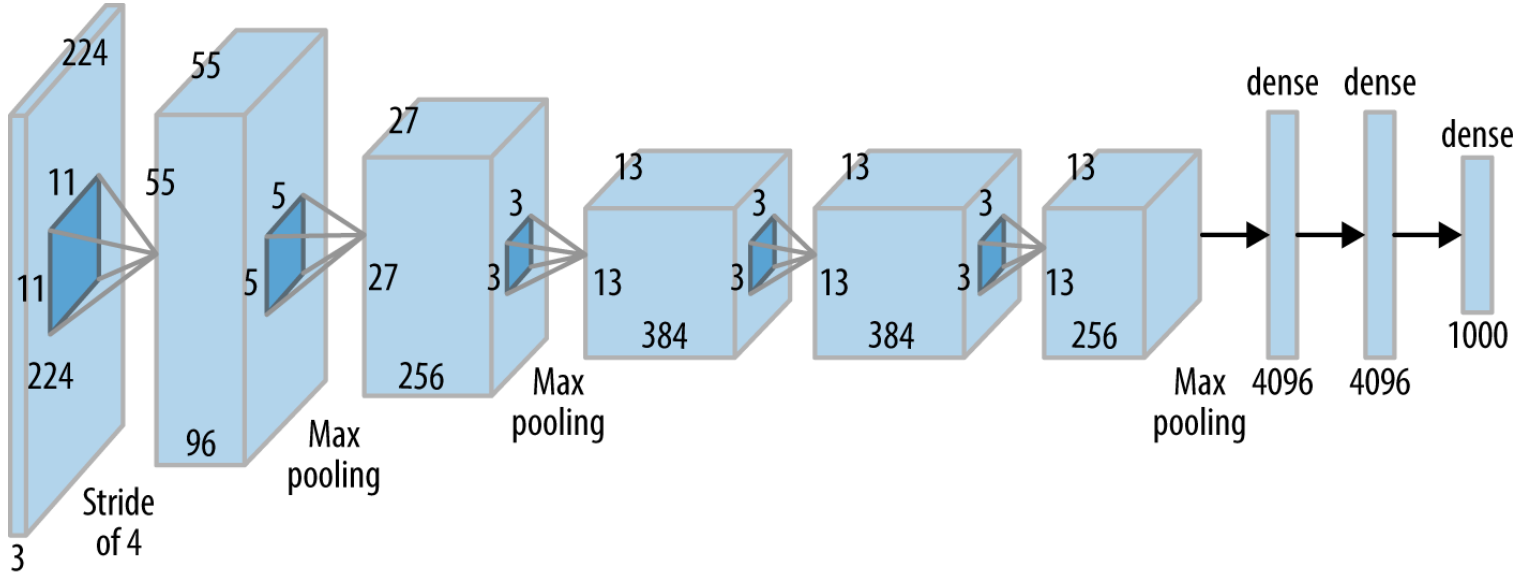
Prof. Fei-Fei Li

Part 2

CNN architectures

Slides adapted from Dr. Wenjia Bai and Dr. Dragana Vuckovic
Department of Computing & Brain Sciences

AlexNet



Conv 1, Norm 1, Pool 1

- Architecture
  - Input
    - 224 x 224 x 3
  - Layer 1 (Conv 1)
    - 11 x 11 convolution, stride of 4, 96 filters
    - Output feature map: 55 x 55 x 96
  - Layer 2 (Norm 1)
    - Local response normalisation and using ReLU as activation function
  - Layer 3 (Pool 1)
    - 3 x 3 max pooling window, stride of 2
    - Output feature map: 27 x 27 x 96

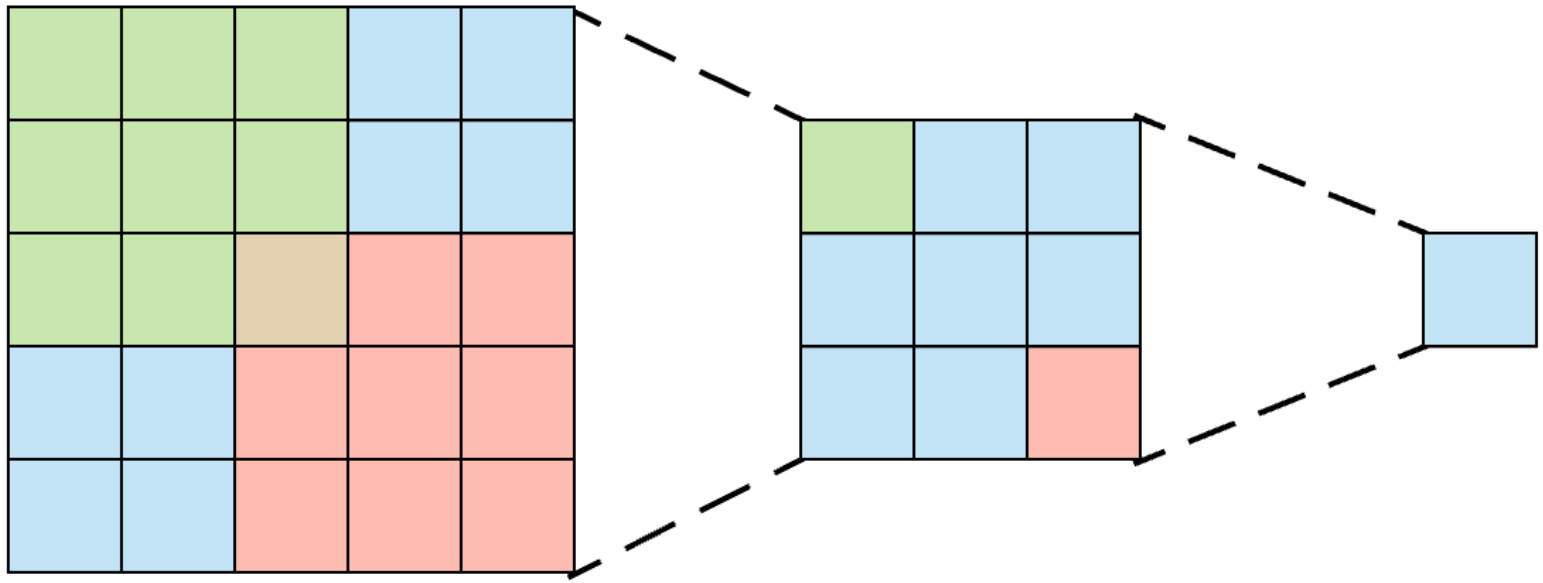
Alex Krizhevsky et al. ImageNet classification with deep convolutional neural networks. NIPS 2012.

VGG

- Visual Geometry Group (VGG) at Oxford
- Let us go deeper.
- But how? How do we design the network architecture?
- Previous networks used different kernel sizes.
  - 11x11, 7x7 or 5x5 convolutional kernels.
- VGG only uses 3x3 convolutional kernels.
  - Rationale: large kernel can be formed by convolving small kernels.
  - Small kernels need less parameters → the network can be deeper

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015.

Convolution of two 3x3 kernels is equivalent to a 5x5 kernel

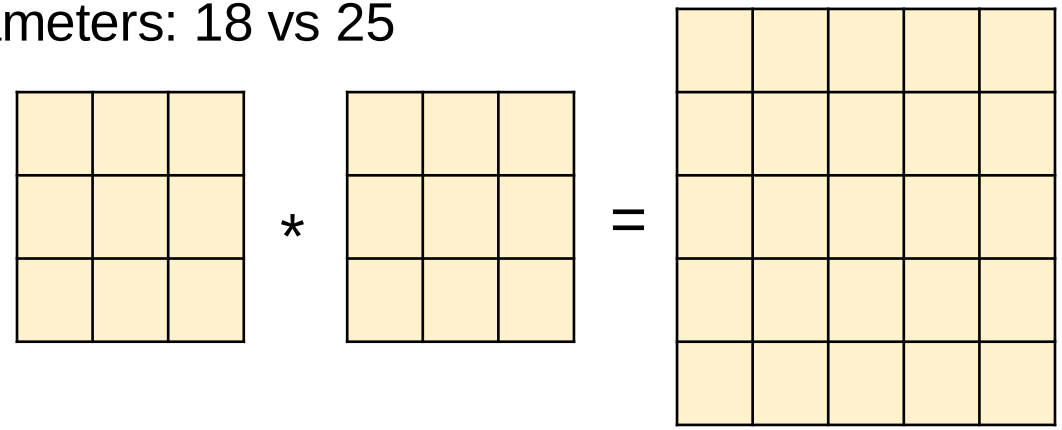


5x5 receptive field

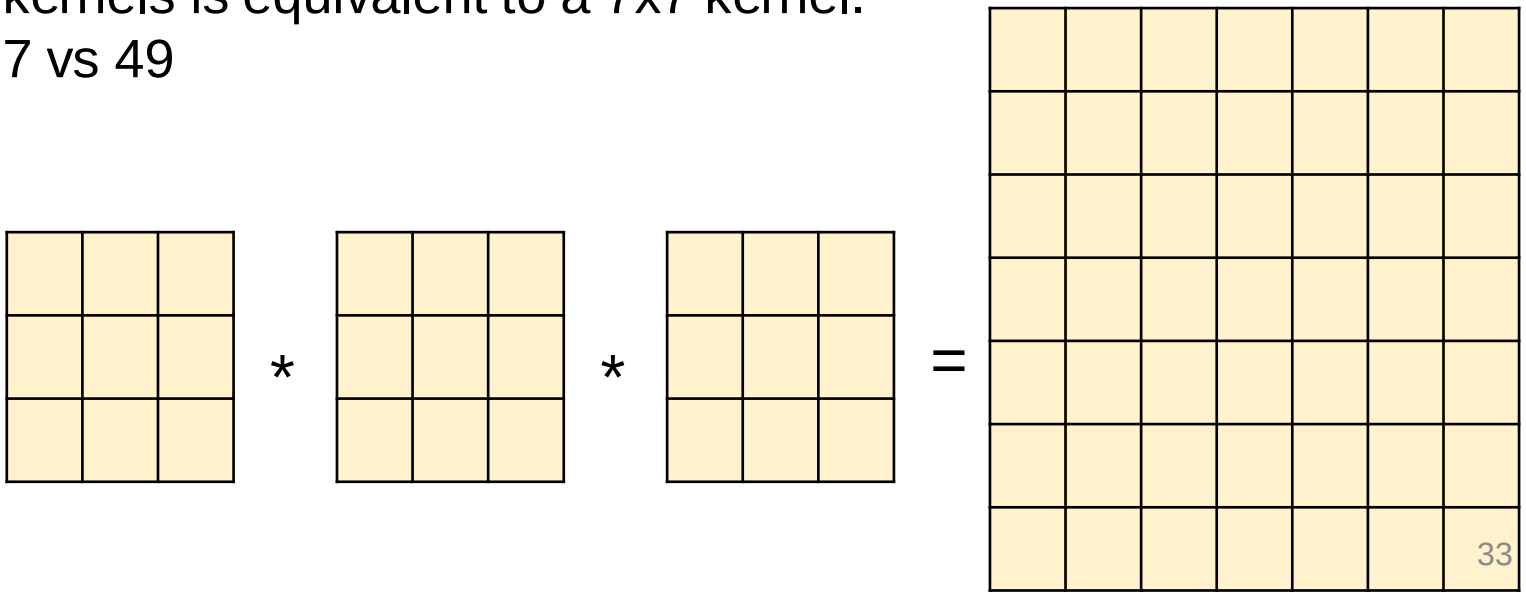
3x3 receptive field

VGG

- Convolution of two 3x3 kernels is equivalent to a 5x5 kernel.
- Number of parameters: 18 vs 25



- Convolution of three 3x3 kernels is equivalent to a 7x7 kernel.
- Number of parameters: 27 vs 49



VGG-16

[224x224x3] Input
[224x224x64] 3x3 conv1, 64
[224x224x64] 3x3 conv1, 64
[112x112x64] Pool
[112x112x128] 3x3 conv2, 128
[112x112x128] 3x3 conv2, 128
[56x56x128] Pool
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[28x28x256] Pool
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[14x14x512] Pool
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[7x7x512] Pool
[4096] FC, 4096
[4096] FC, 4096
[1000] FC, 1000

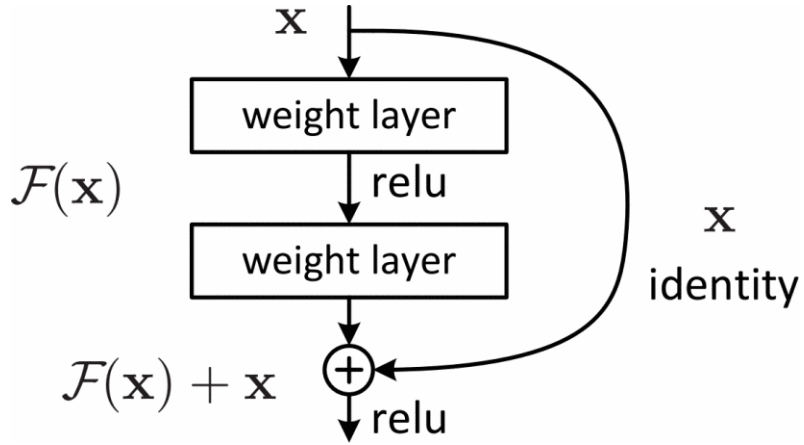
VGG-19

[224x224x3] Input
[224x224x64] 3x3 conv1, 64
[224x224x64] 3x3 conv1, 64
[112x112x64] Pool
[112x112x128] 3x3 conv2, 128
[112x112x128] 3x3 conv2, 128
[56x56x128] Pool
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[56x56x256] 3x3 conv3, 256
[28x28x256] Pool
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[28x28x512] 3x3 conv4, 512
[14x14x512] Pool
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[14x14x512] 3x3 conv5, 512
[7x7x512] Pool
[4096] FC, 4096
[4096] FC, 4096
[1000] FC, 1000

ResNet

- Deeper architectures do not seem to improve any further
- One possible cause is a problem with the backpropagation through many layers (vanishing gradient)
- ResNet (developed by Microsoft Research) introduces "residual units"

Residual block



- Residual unit

$$z^{(l)} = x^{(l)} + F\left(x^{(l)}; W^{(l)}\right)$$
$$x^{(l+1)} = f(z^{(l)})$$

  - $x^{(l)}$: input to the l-th residual unit
  - $x^{(l+1)}$: output of the l-th residual unit

The identity mapping ensures that the gradient will not vanish during backpropagation even for deep networks.

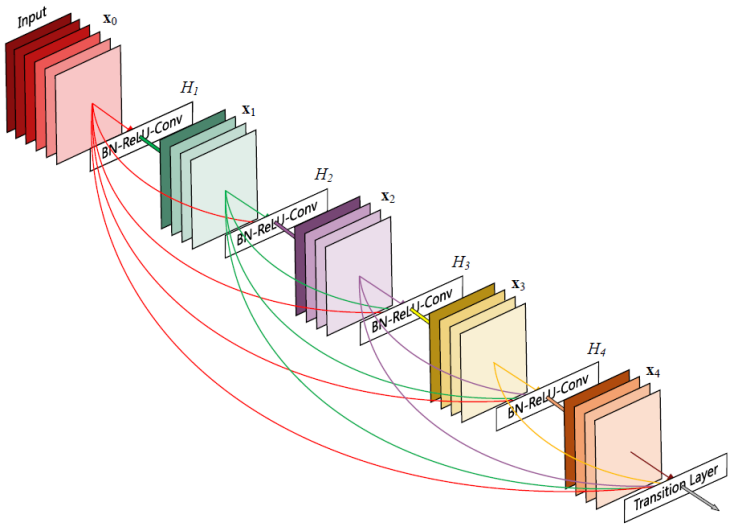
- Differentiate of the loss L with respect to x using chain rule,

$$\frac{\partial L}{\partial x^{(l)}} = \frac{\partial L}{\partial x^{(l+1)}} \cdot \frac{\partial x^{(l+1)}}{\partial x^{(l)}} = \frac{\partial L}{\partial x^{(l+1)}} \cdot f'(z^{(l)}) \cdot (1 + \frac{\partial F\left(x^{(l)}; W^{(l)}\right)}{\partial x^{(l+1)}})$$

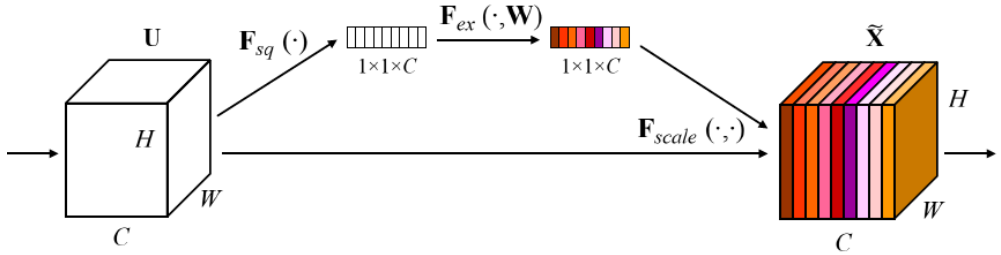
K. He, et al., "Deep Residual Learning for Image Recognition," doi: 10.1109/CVPR.2016.90.

More architectures

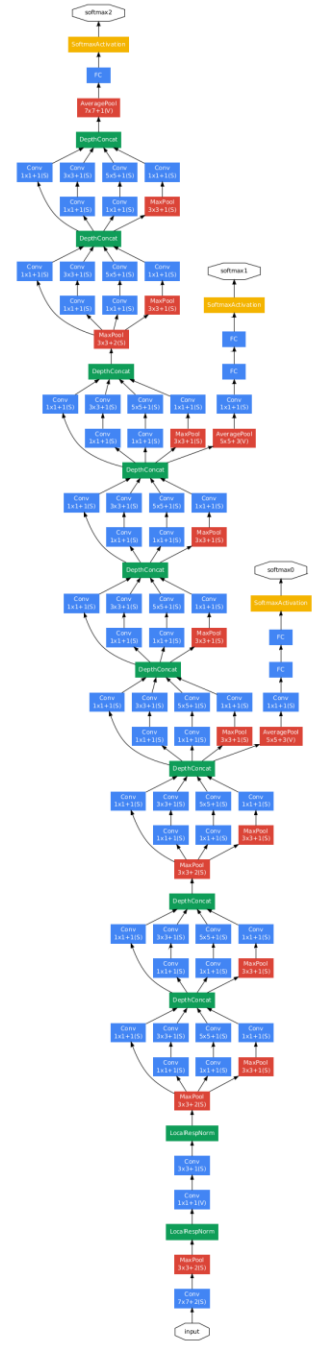
- GoogLeNet (C. Szegedy, CVPR 2015)
- Wide ResNet (S. Zagoruyko, BMVC 2016)
- DenseNet (G. Huang, CVPR 2017)
- SENet (J. Hu, CVPR 2018)



DenseNet

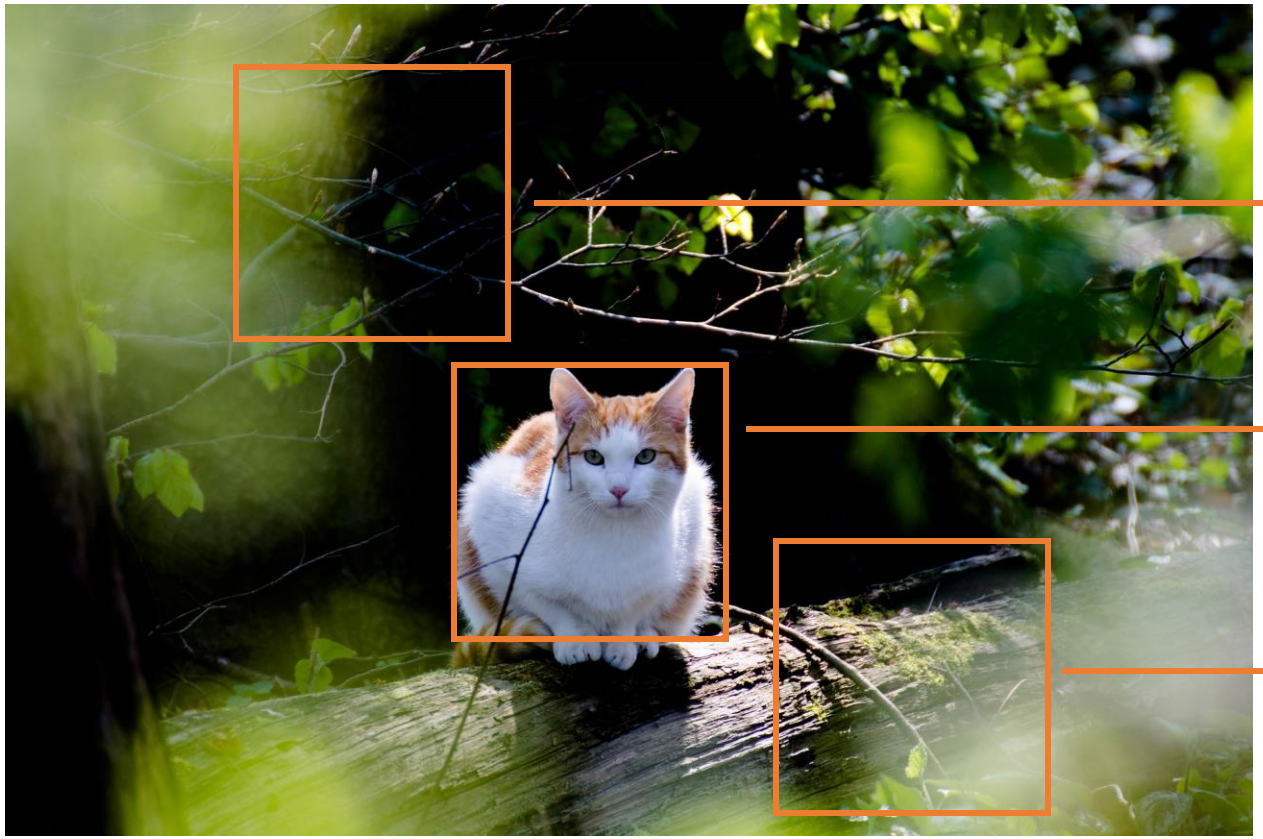


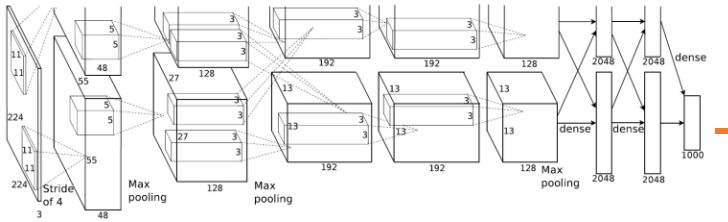
SENet



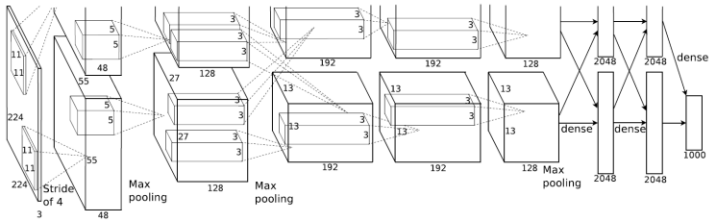
GoogLeNet

Object detection

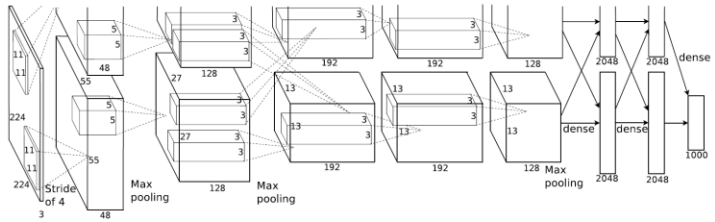




Background



Cat

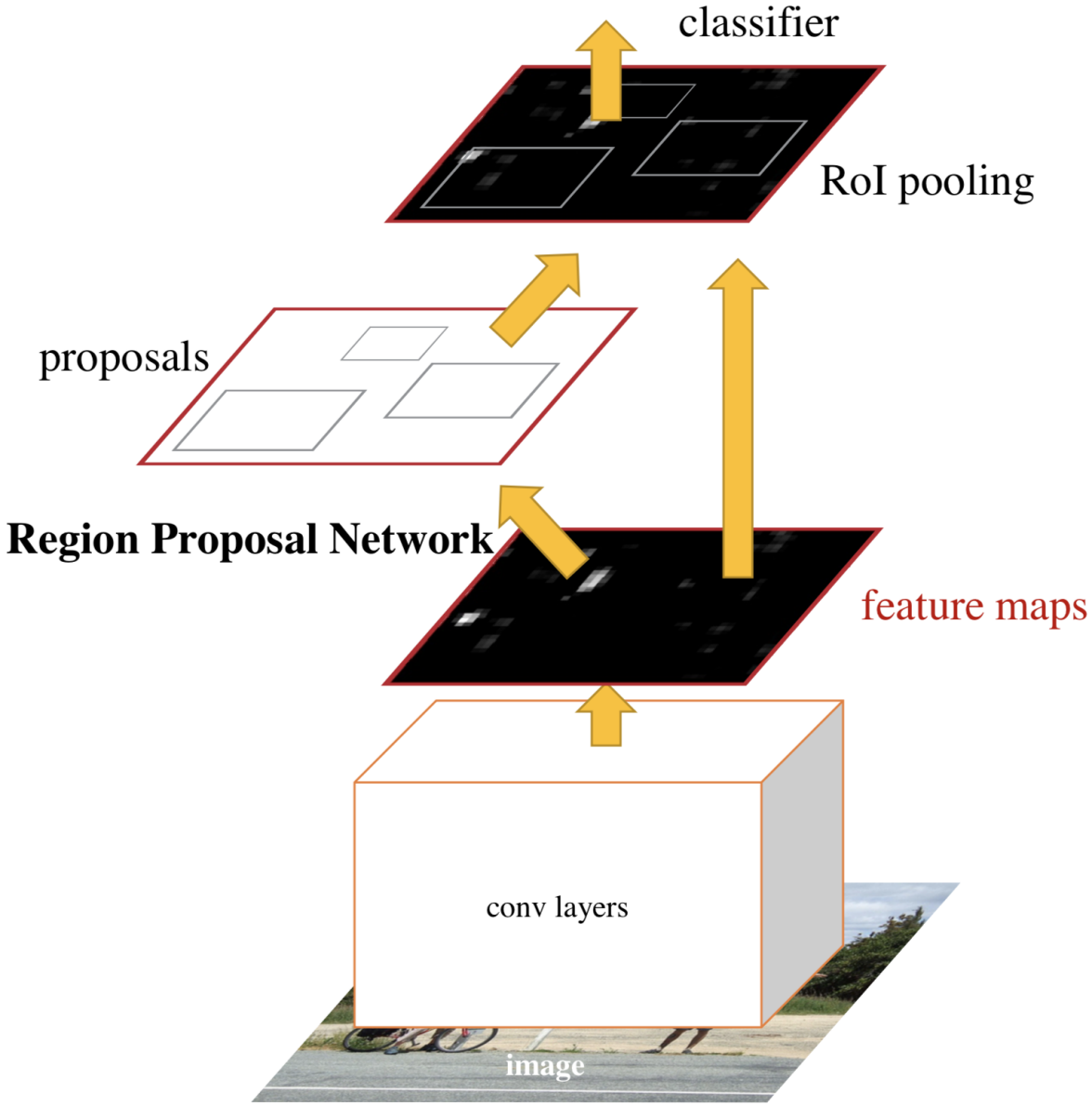


Background

Two-stage object detection

- Stage 1: a region proposal network (RPN)
- Stage 2: a detection network
  - We will focus on “Faster R-CNN”, which uses a CNN for both stages.

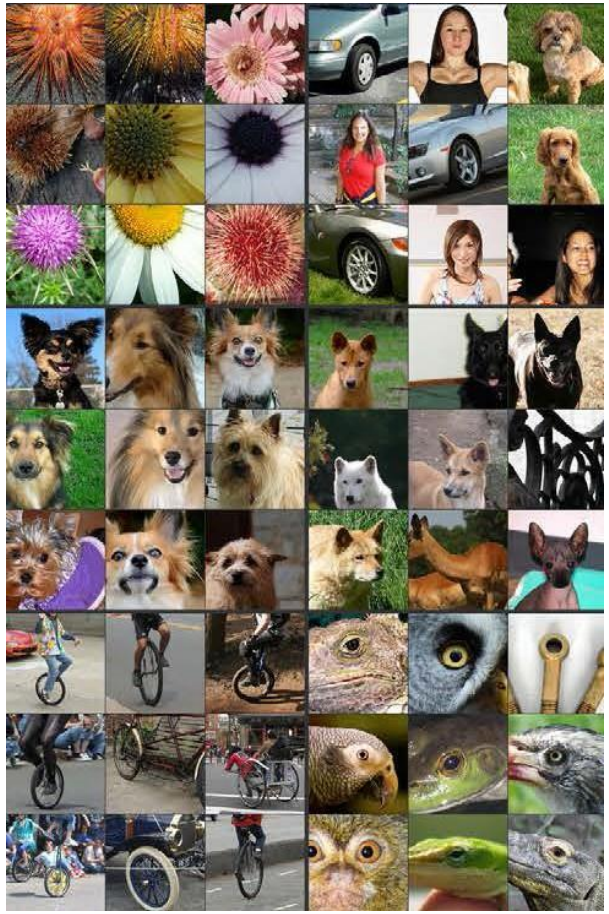
- The RPN relies on **feature maps**, which are the outputs of the convolutional layers and capture areas of the image that are similar to its filters



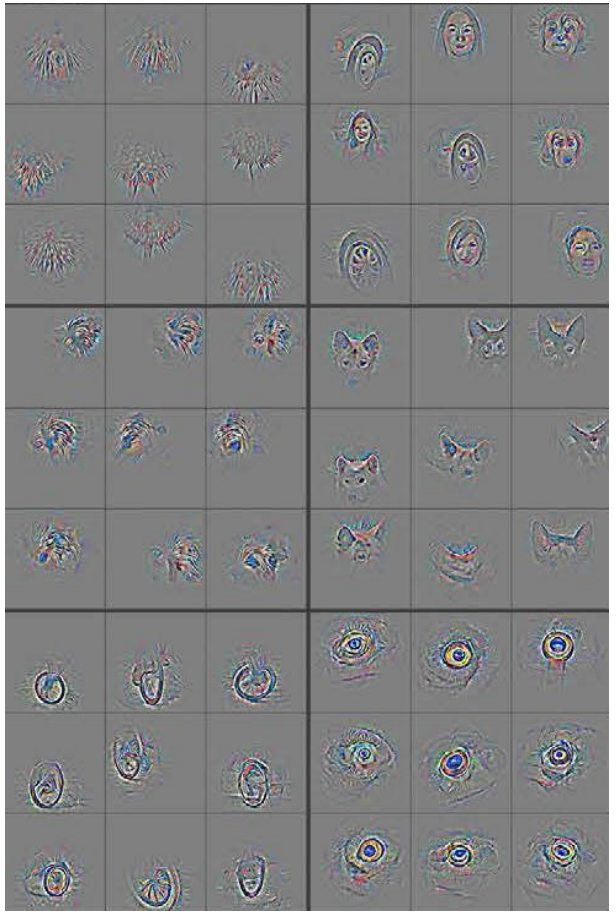
S. Ren et al. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS 2015.

Convolutional feature map

- Each pixel of the feature map is a high-dimensional feature vector, which describes the content of a small region in the input image.
- This feature vector may tell us whether this is an interesting region or not.



Input images

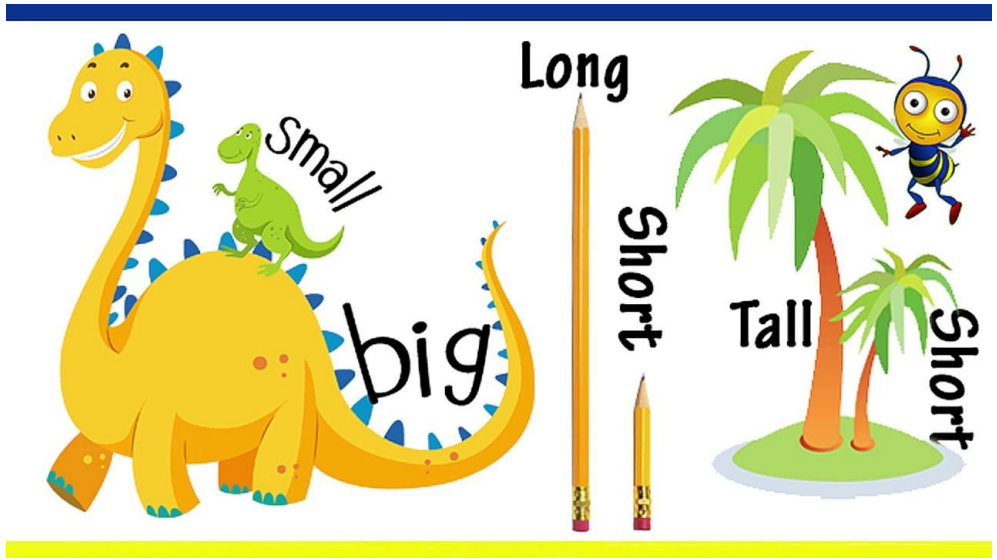


Visualization of features maps

M.D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. ECCV 2014.

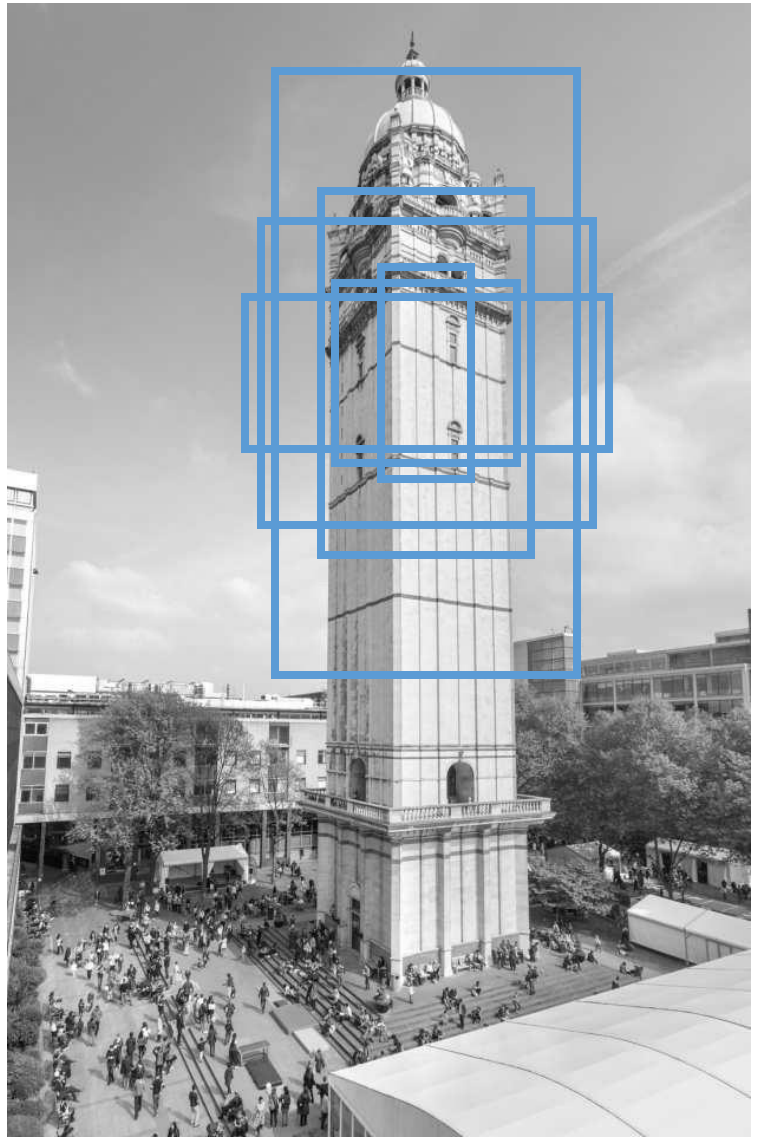
Region proposal network

- To proposing regions, we go across the convolutional feature map using a 3x3 sliding window.
- At each location, we perform a binary classification.
  - 0: not interesting
  - 1: interesting
- However, how does this fixed size window handle objects of different sizes and different aspect ratios?



Region proposal network

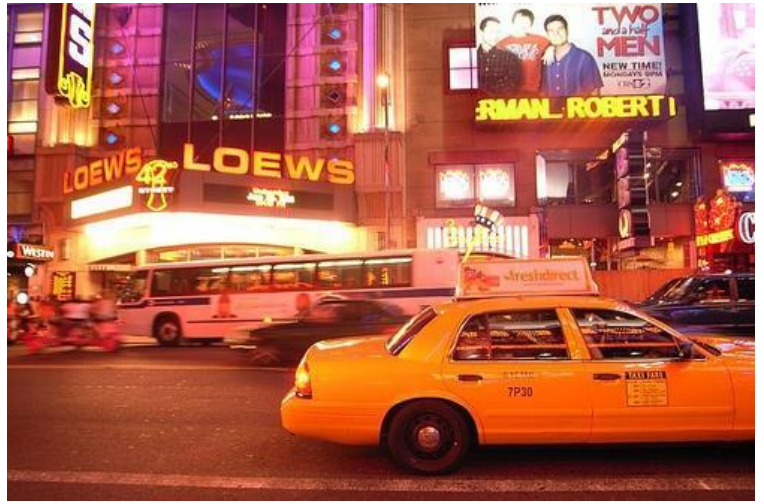
- At each sliding window, the network makes k predictions (k bounding boxes), for objects of different scales and aspect ratios.
  - 3 scales: 128^2, 256^2, 512^2
  - 3 aspect ratios: 1:1, 1:2, 2:1
  - In total, k = 3 x 3 = 9
  - These bounding boxes are called “anchors”.
- Classification is performed for each bounding box
- In the next stage, we refine both the classification and localisation.



At each location, the network aims to classify whether each anchor (e.g. 128x128 anchor, 128x256 anchor, 128x64 anchor, 256x256 anchor …) is an object or not.

RPN vs Detection network

- RPN moves the 3 x 3 sliding window (green), classifies it to be something interesting (binary classification) and proposes a region (blue).
- The detection network takes the features inside the blue region, pools it into a 7 x 7 window, classifies it to be a car (multi-class classification) and refines the bounding box (purple).



Input image

Feature map

Image credit: K. He.

Part 2b

Representation and Transfer Learning

Chapter 15
Deep Learning
I. Goodfellow, Y. Bengio, A. Courville

Transfer Learning

It takes a long time and expertise to build and train a deep network!
Can we re-use / up-cycle it?

Transfer Learning refers to storing knowledge (aka a trained deep learning network) from one problem in order to apply it to a different but related problem.

There are several pre-trained networks that are available to load and use:

- If the task is the same, you can apply the trained model to your data
- If the task is similar but slightly different, you can re-train only small parts of the model – usually the last one or few layers.

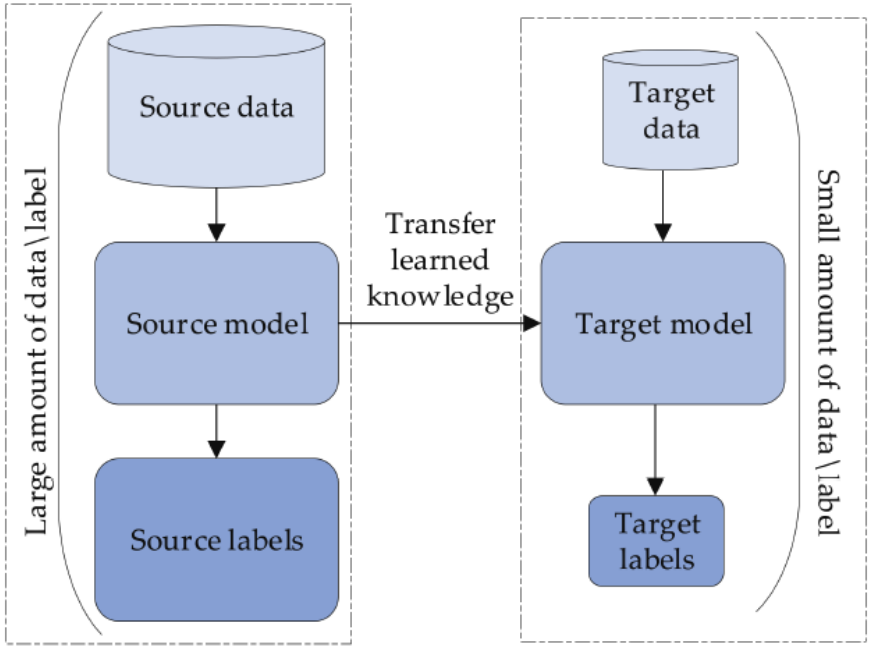


Figure 6. The concept of the transfer learning.

image credit: 10.3390/app10134523

Trained neural networks - advantages

- Time and computational cost savings as a large network is not trained from scratch
- Only the original training dataset needs to be very big, while the ”application” dataset can be much smaller

Medical datasets

Medical imaging datasets are notoriously small sized, or they contain unlabeled data. Labelling data requires a lot of time and is expensive (trained experts are needed).

- Is it possible to use transfer learning in a healthcare/medical context ?
  - i.e. are networks trained on dogs and cats able to distinguish between benign and malignant?

Representation learning

The theory behind transfer learning is the following:

- the final layer of the network is the classifier
- all the other layers are learning a **representation of the data**

- This is true for both supervised and unsupervised learning

- **Example:**
  - A network trained to distinguish between images of cats and dogs might be re-used to distinguish between ants and wasps
  - Why? the hidden layers are learning about the general characteristics of the images e.g. contours, colors, lighting effects, etc and only the final layers are involved in separating the data for classification
  - We can even use the trained network to learn a representation and later classify with a more traditional method that works better on smaller datasets

Representation learning

- Sometimes, when up-cycling networks, it is more useful to re-train the first layers and keep the last ones fixed
    - **Example:** a speech recognition systems will have very different inputs depending on the person speaking (or the language spoken), while the final output will be valid sentences in any case
- **Domain adaptation/transfer:** Network can also be up-cycled to perform the same task in different settings
    - **Example:** a sentiment analysis developed for comments on books can be used for comments on electronics
    - Note: a sentiment analysis is a natural language processing (NLP) model that can distinguish, in this case, bad and good reviews

Example

In this paper, the authors use a pre-trained CNN (AlexNet) to obtain a new representation for a dataset of Otitis Media (OM) images.

Dataset: OM images (sample size 1,195 including true and false i.e. healthy cases)

Method

- AlexNet is trained on 15 million images over 22,000 categories of everyday objects.
- It has 8 layers (5 convolutional and 3 fully-connected)

- OM images are fed to AlexNet and output from layers 5, 6 and 7 is extracted
- SVM is then used for classification on each layer and results are compared to human classification

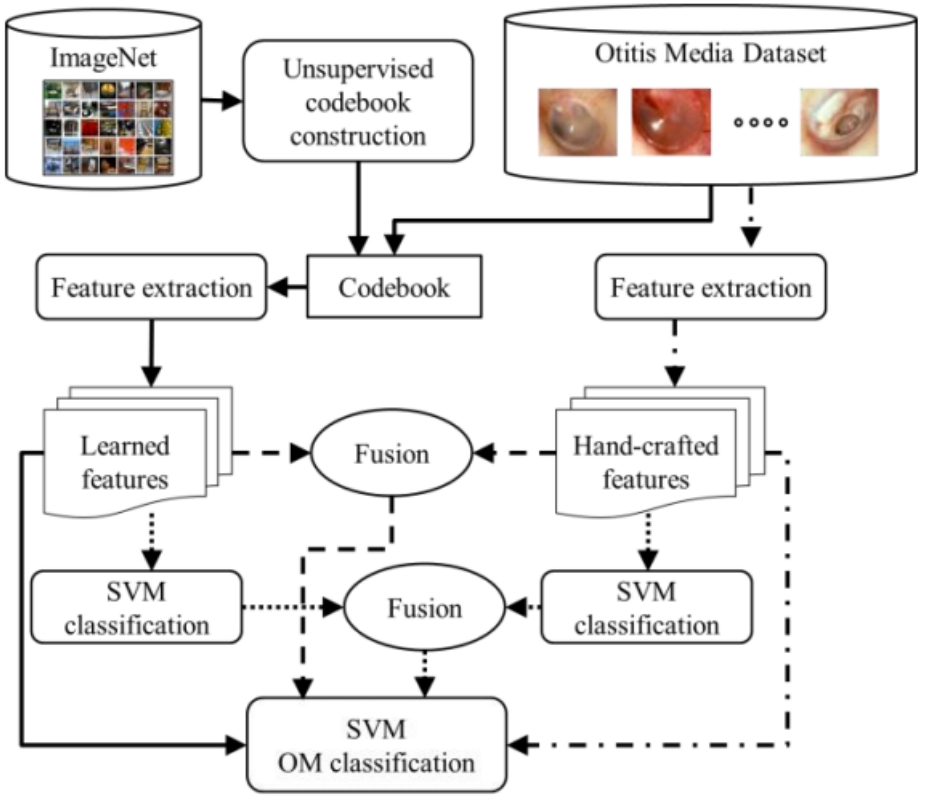


Figure 2. Four classification flows

Results

TABLE I. OM CLASSIFICATION EXPERIMENTAL RESULTS

Method	Measures			
	Accuracy	*Sensitivity*	*Specificity*	*F_1-score*
Heuristic w/ seg	80.11%	83.33%	75.66%	0.822
Heuristic w/o seg	76.19%	79.38%	71.74%	0.79
Transfer w/ seg (pool5)	87.86	89.72%	86.26%	0.89
Transfer w/o seg (pool5)	88.37%	89.16%	87.08%	0.894
Transfer w/ seg (fc6)	87.58%	89.33%	85.04%	0.887
Transfer w/o seg (fc6)	**88.5%**	89.63%	86.9%	0.895
Transfer w/ seg (fc7)	85.6%	87.5%	82.7%	0.869
Transfer w/o seg (fc7)	86.9%	88.5%	84.9%	0.879
OM-trained codebook	71.8%	95.06%	41.66%	0.818
Feature fusion	89.22%	90.08%	87.81%	0.90
Classifier fusion	**89.87%**	89.54%	90.2%	0.898

human

transfer learning

CNN on the small dataset

combination of human and CNN

Conclusions and summary

- The data structure of images is not suitable for densely connected neural networks
- Convolution layers can detect patterns in the data with fewer parameters while accounting for correlations between the pixels
- Deeper networks outperform shallow ones at image classification
- Deep learning networks learn data representations that can be useful in new problems
- Transfer learning refers to using pre-trained networks on new dataset
- There is potential for transfer learning to be useful in medical applications, but a lot of research and development is still needed

Part 3

Applied research – Dr. Barbara Metzler

Measuring and analysing urban development in Sub-Saharan Africa using satellite imagery and machine learning

www.equitablehealthycities.org

Summary

- CNNs for analysing imagery of environments
  - Measuring relevant features of the social and physical environment with images and CNNs

- Example from my research
  - Characterising urban environments with unsupervised ML and satellite imagery

Street View in London

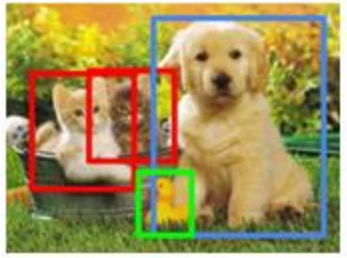
Cape Town from air

Deep learning for computer vision tasks

Classification

CAT

Object detection



CAT, CAT,
DUCK, DOG

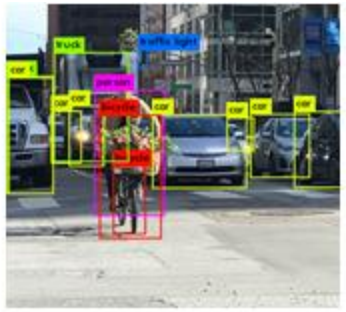
Instance segmentation

CAT, CAT,
DUCK, DOG

Semantic segmentation
Pixel classification

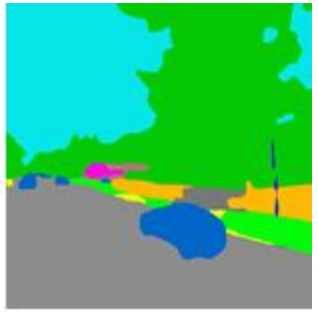
GRASS, CAT,
TREE, SKY

Deprived / poor



Street images
object locations & counts

Building footprint
extraction

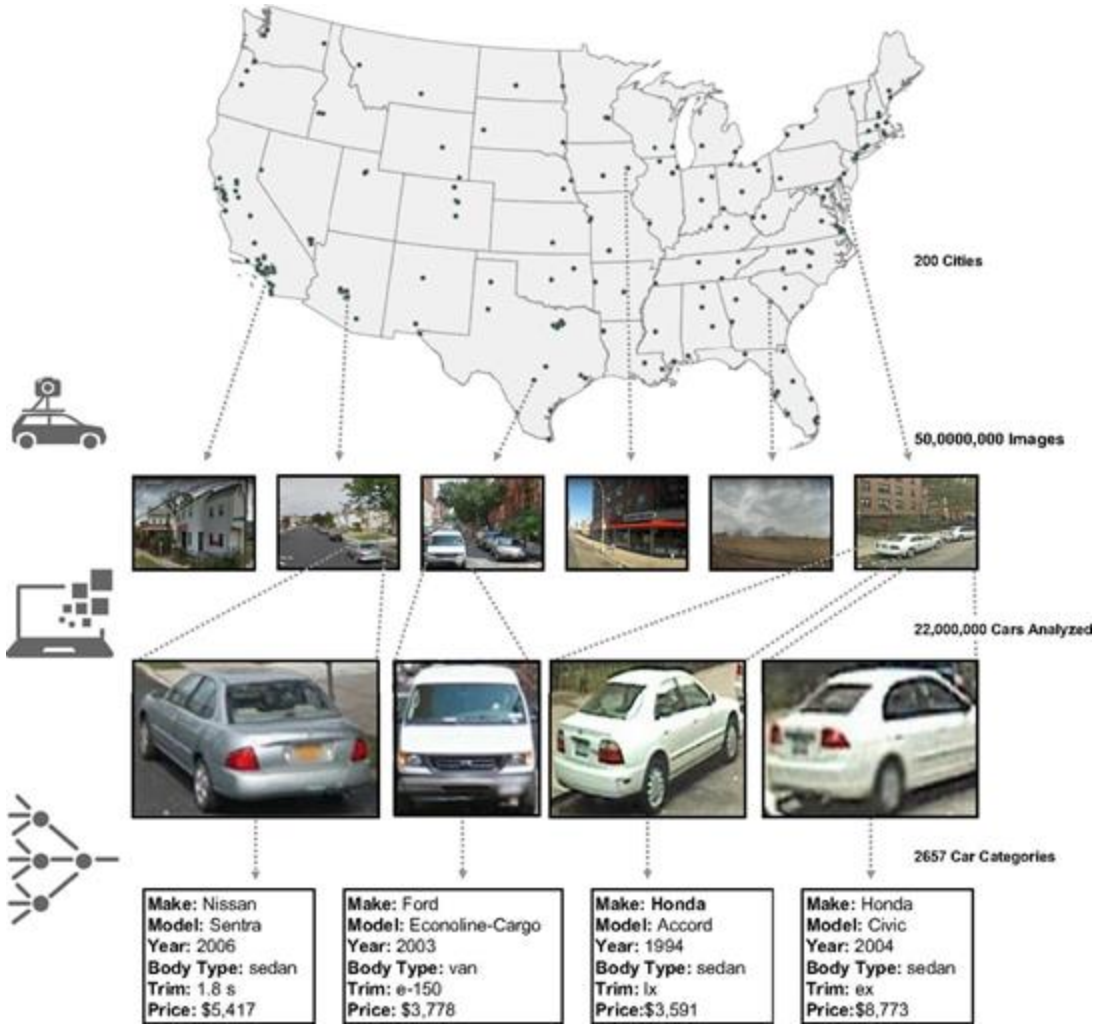


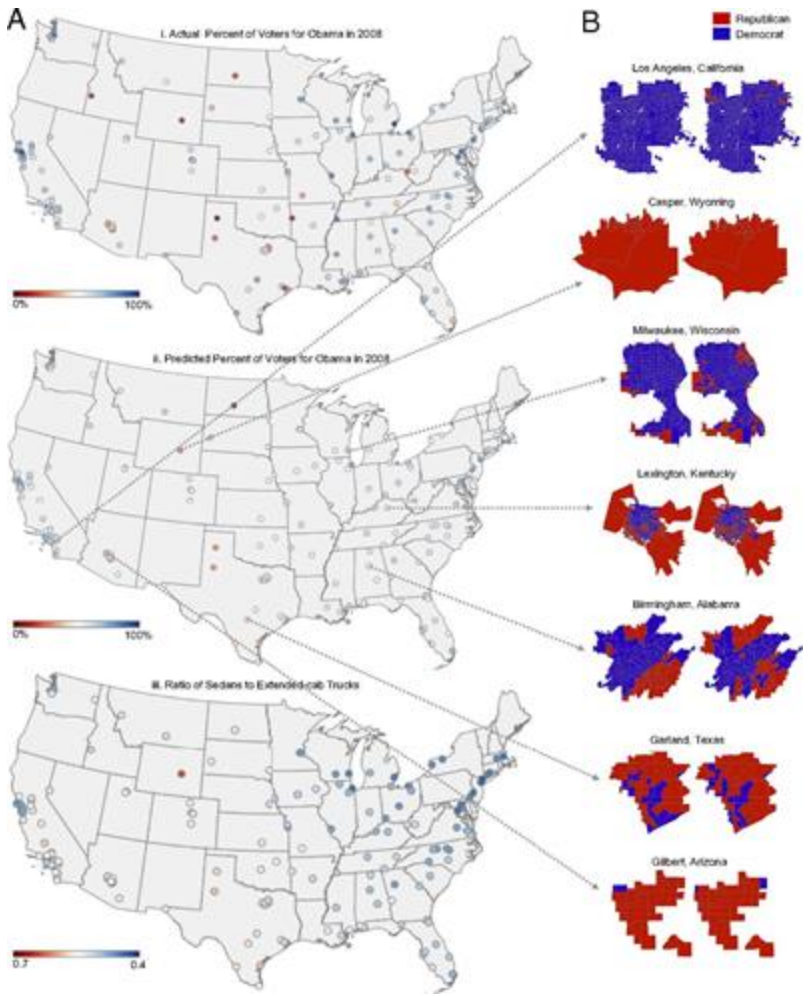
Street images
pixel classification

Object detection

Nathvani & Clark et al. (2021). Spatiotemporal characterization of urban activity and environment with imagery and deep learning.

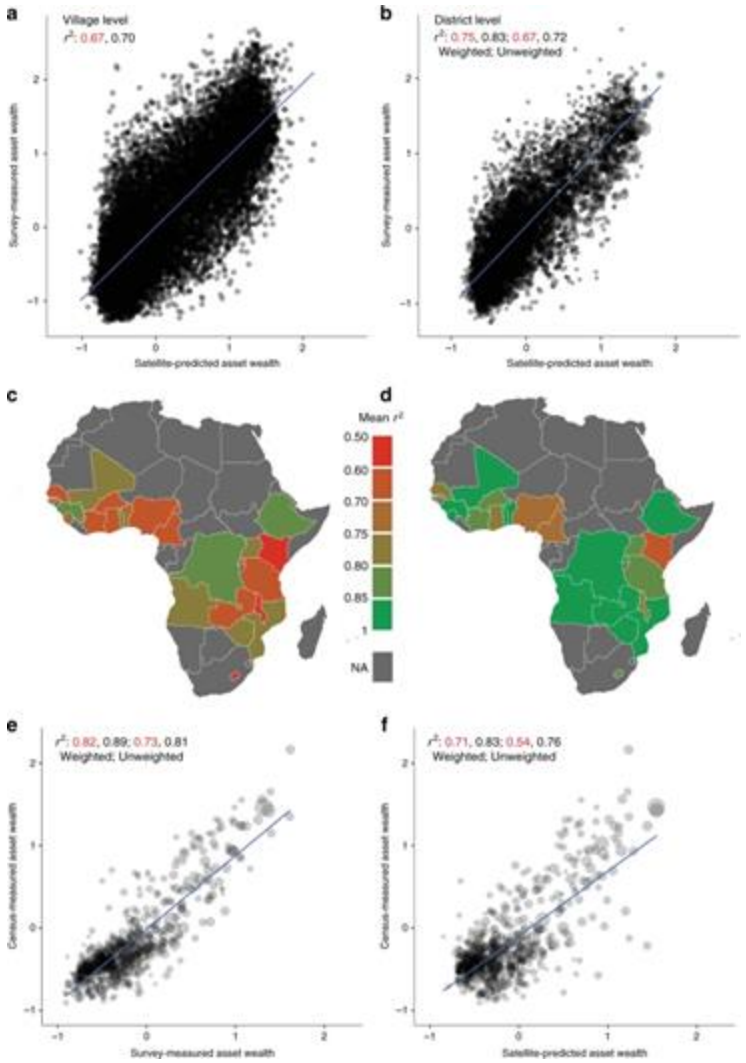
Using object detection to estimate demographic makeup of neighbourhoods across the US

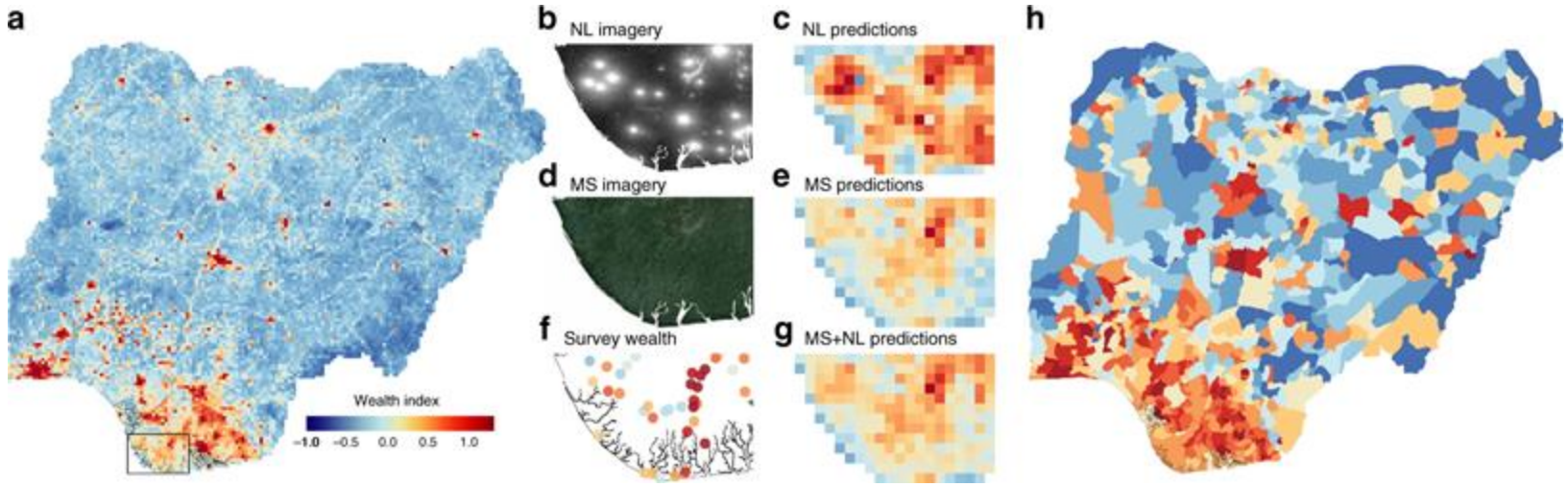




Gebru et al, 2017

Measuring economic well-being in SSA with satellite imagery





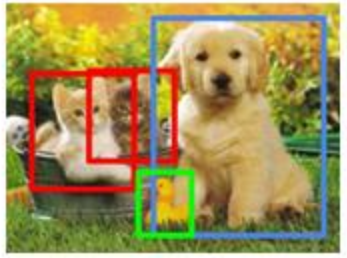
Yeh et al, 2020

Deep learning for computer vision tasks

Classification

CAT

Object detection



CAT, CAT,
DUCK, DOG

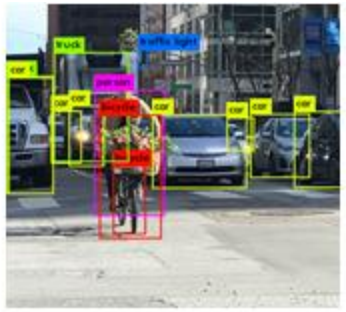
Instance segmentation

CAT, CAT,
DUCK, DOG

Semantic segmentation
Pixel classification

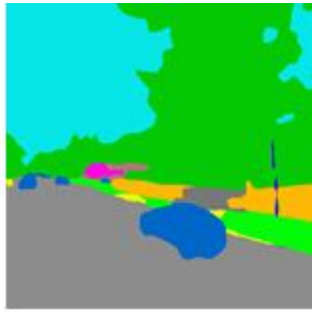
GRASS, CAT,
TREE, SKY

Deprived / poor



Street images
object locations & counts

Building footprint
extraction



Street images
pixel classification

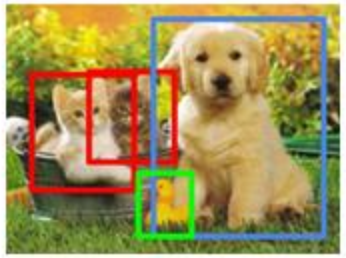
Deep learning for computer vision tasks

LABELS

Classification

CAT

Object detection



CAT, CAT,
DUCK, DOG

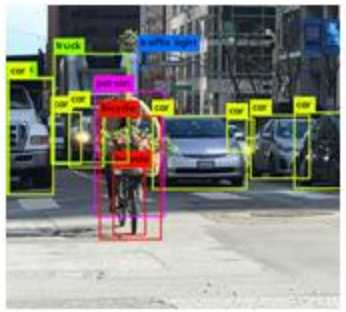
Instance segmentation

CAT, CAT,
DUCK, DOG

Semantic segmentation
Pixel classification

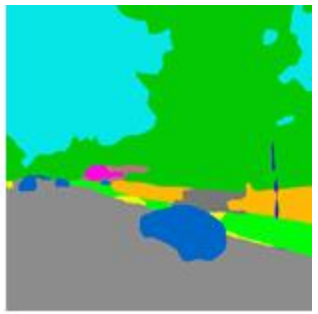
GRASS, CAT,
TREE, SKY

Deprived / poor



Street images
object locations & counts

Building footprint
extraction



Street images
pixel classification

Computer vision meets remote sensing

Can we capture variations in urban form in a completely unsupervised way?

Computer vision meets remote sensing

Potential of imagery

- Multidimensional factors of urban development can be measured from images of the urban environment
- Satellite imagery has become more widely available

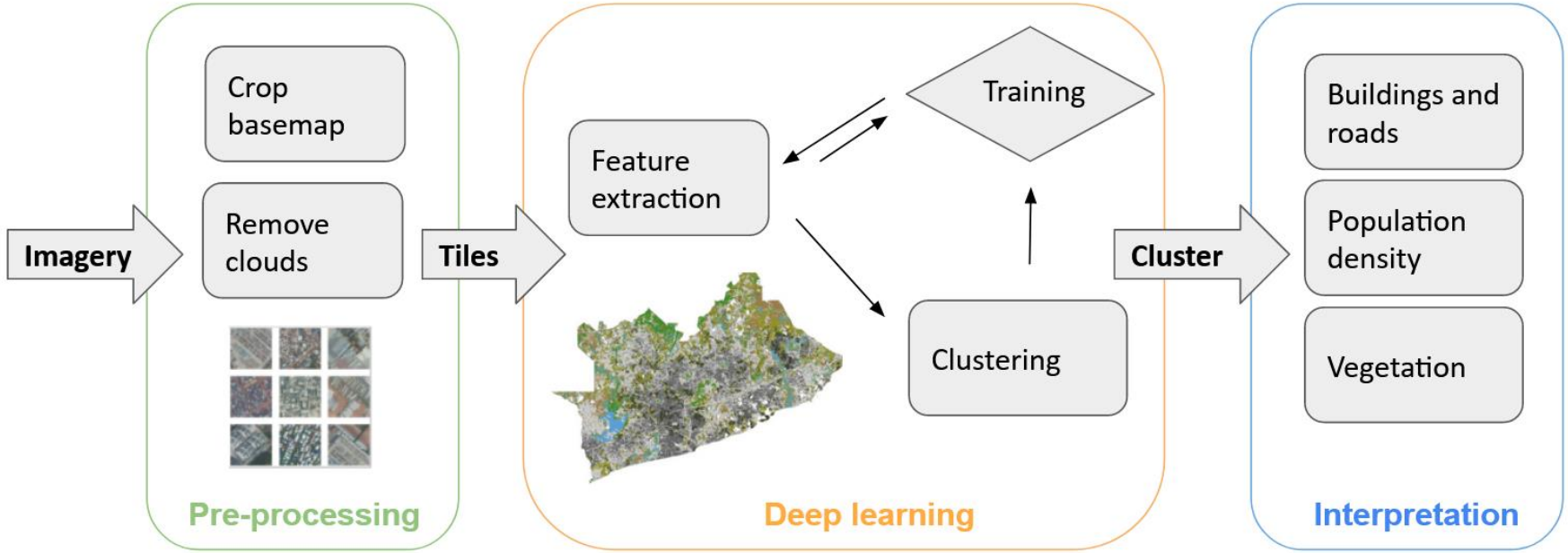
Unsupervised deep learning approach for clustering satellite imagery

Science of The Total Environment
Volume 893, 1 October 2023, 164794

Phenotyping urban built and natural environments with high-resolution satellite images and unsupervised deep learning

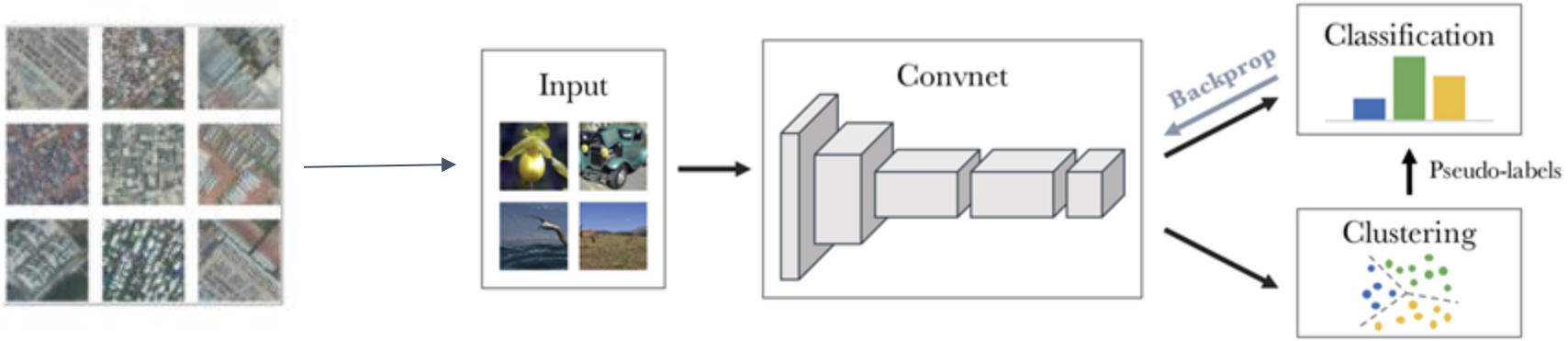
A. Barbara Metzler [a b], Ricky Nathvani [a b], Viktoriia Sharmanska [c d], Wenjia Bai [d e], Emily Muller [a b], Simon Moulds [f], Charles Agyei-Asabere [g], Dina Adjei-Boadi [h],

- **Very high resolution** satellite image of Accra, 2019
- 0.3m/pixel, RGB bands



- **Sensitivity analysis**: cluster development over varying number of clusters and scale of tiles

Method: DeepCluster



n x 256 x 256 x 3
(RGB)

Feature extraction + clustering

Caron et al, 2018

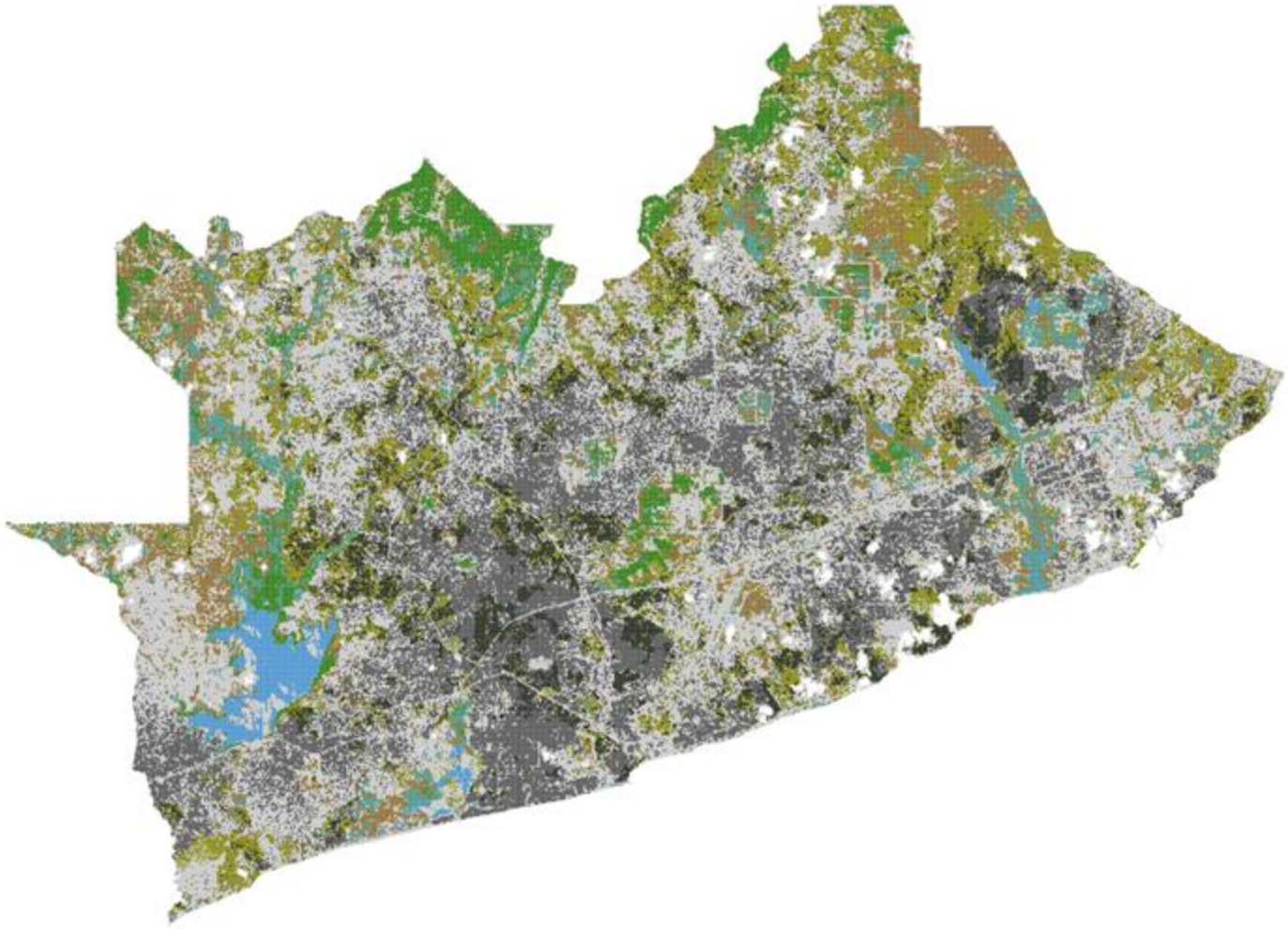
Combined approach: feature extraction and clustering

i) Image tiles are fed into a CNN (pretrained on ImageNet)

ii) Deep features are clustered with k-means

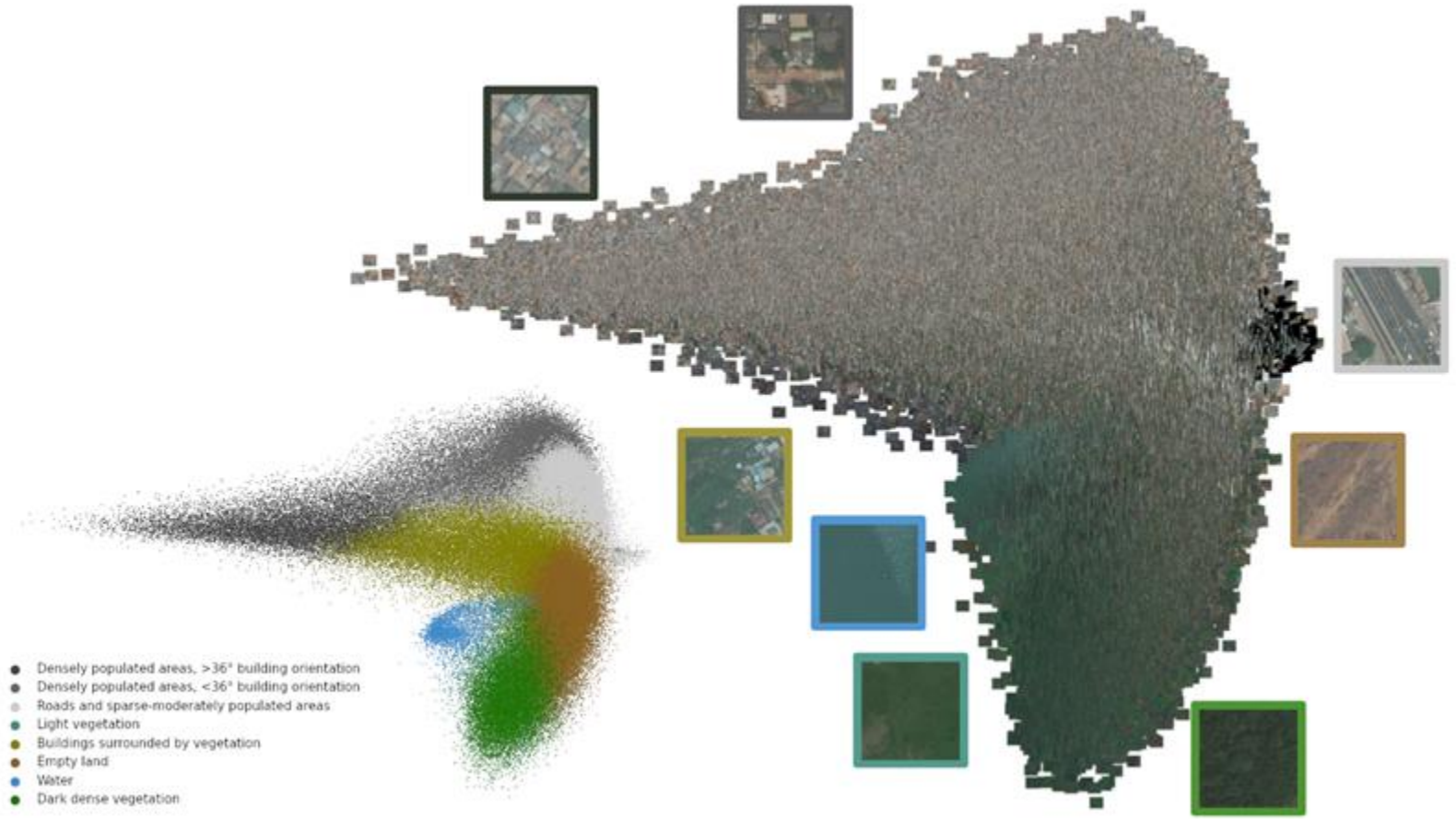
iii) Cluster assignments are used as pseudo-labels to learn the parameters of the CNN

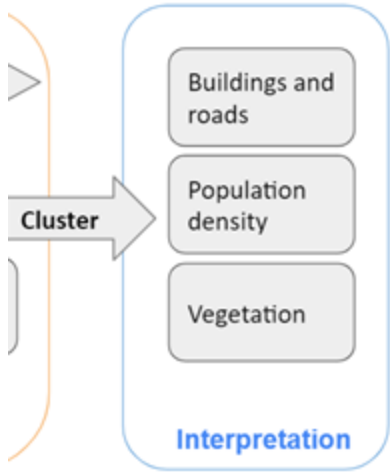
Results (K=8)



Visualising the feature space

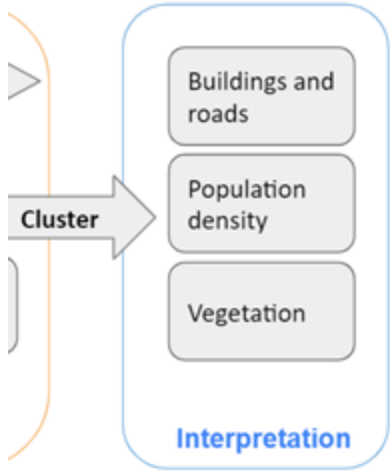
Image features plotted on the first and second PC





How do we evaluate these clusters?

External evaluation

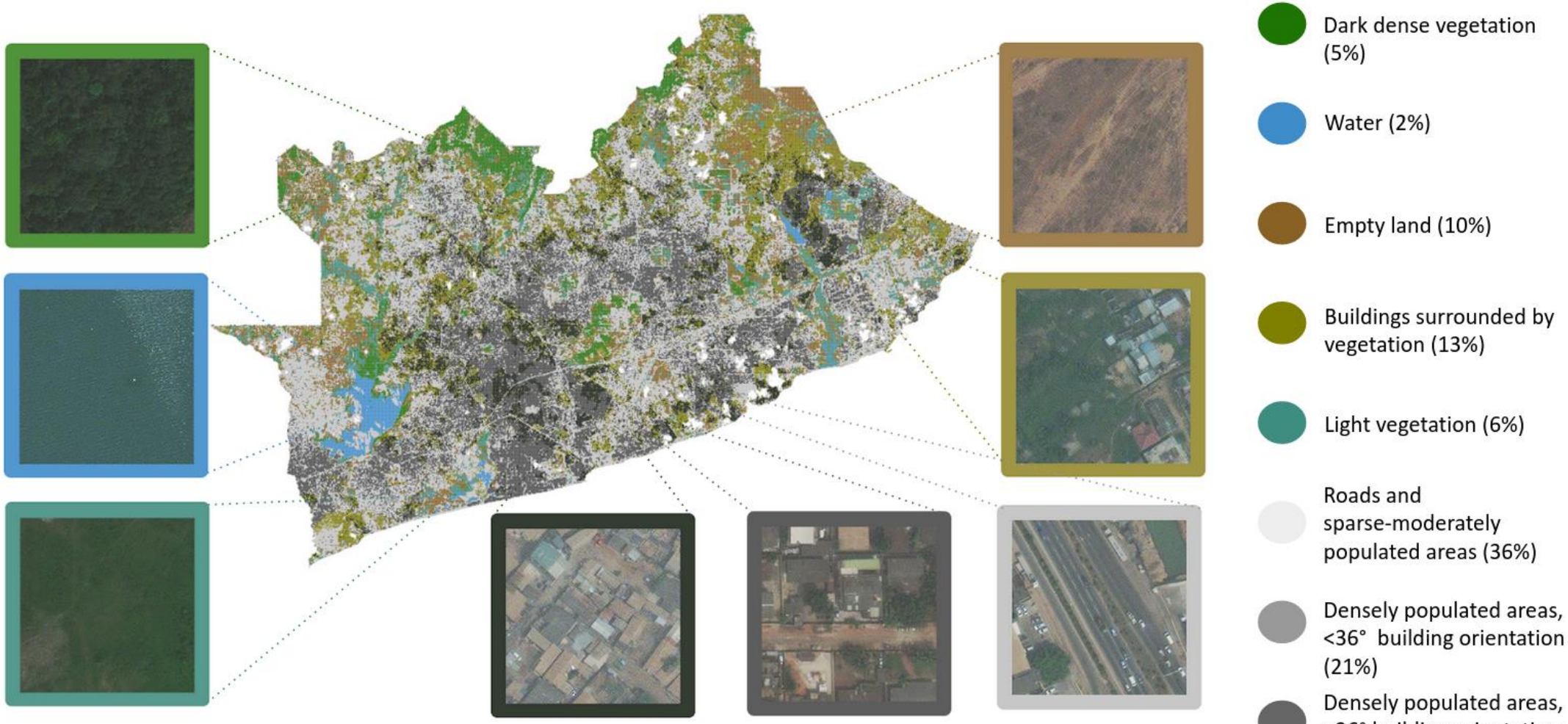


Use external datasets to interpret the clusters

Measures of urban form

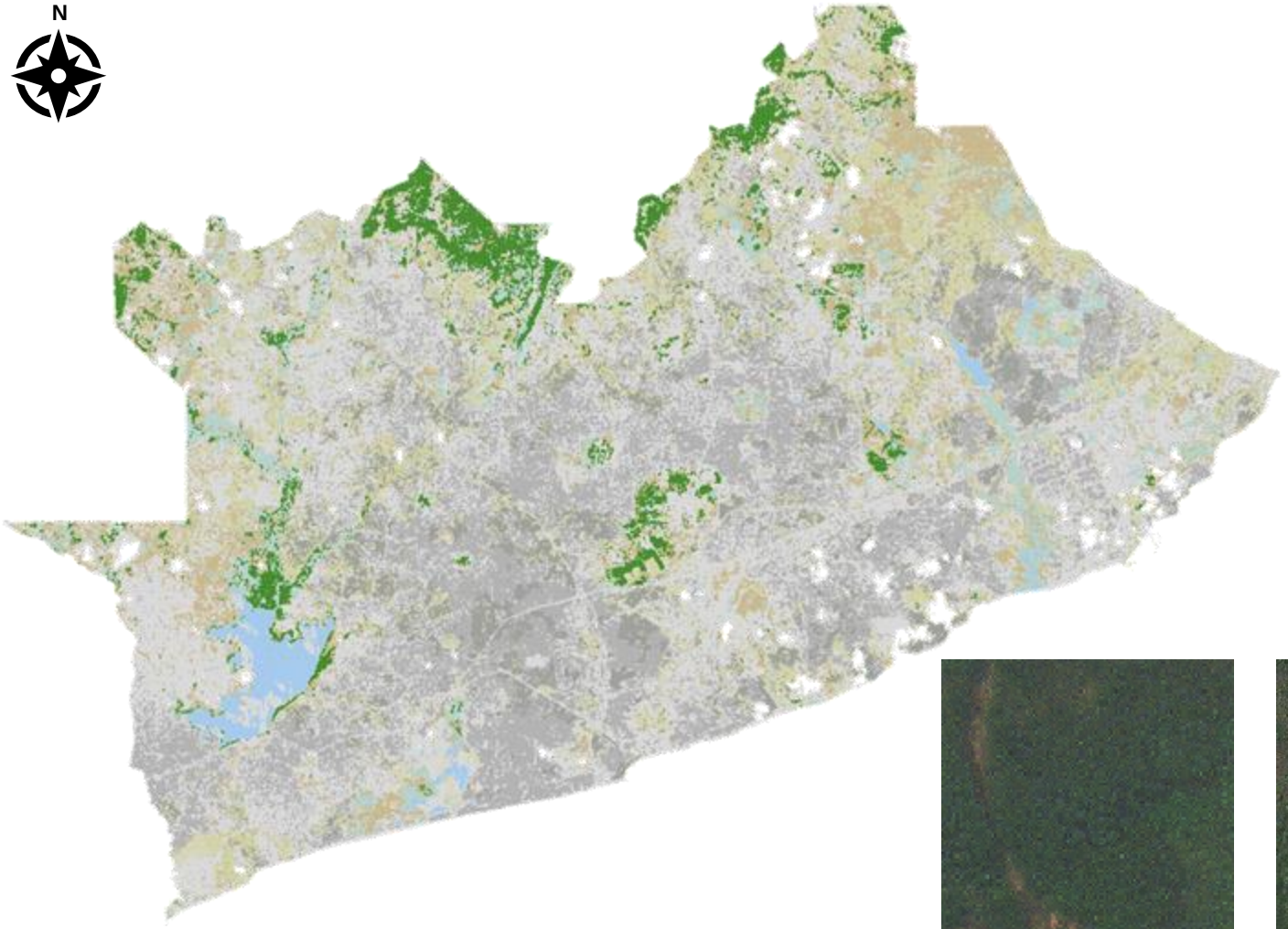
- Building information: building count, building area (m^2), average building size (m^2), building orientation
- Road information: distance to closest road (m), distance to closest major road (m), length of road segment (m)
- NDVI and population density

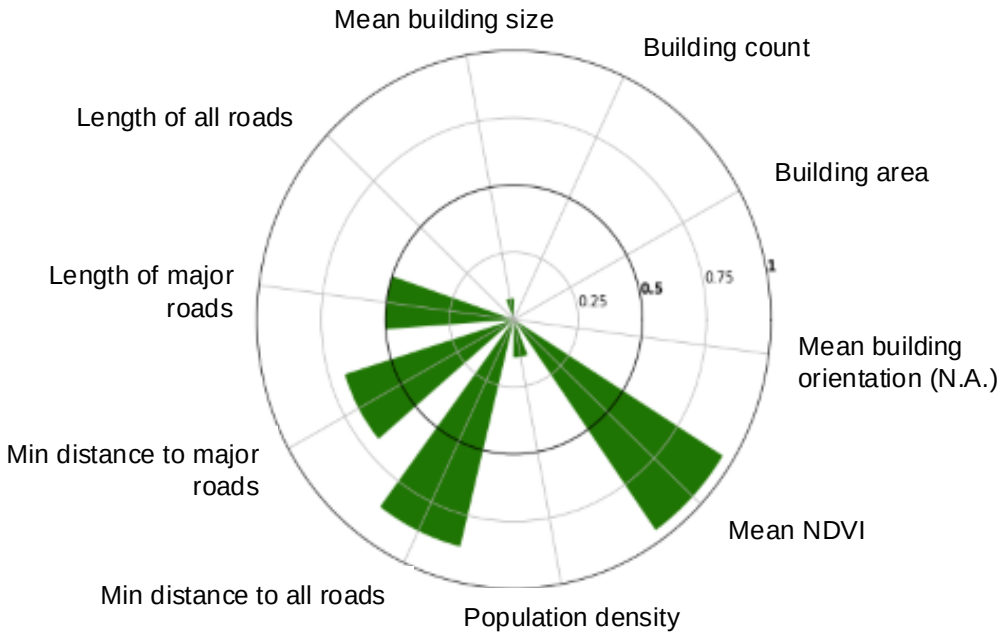
Eight clusters of Accra, Ghana



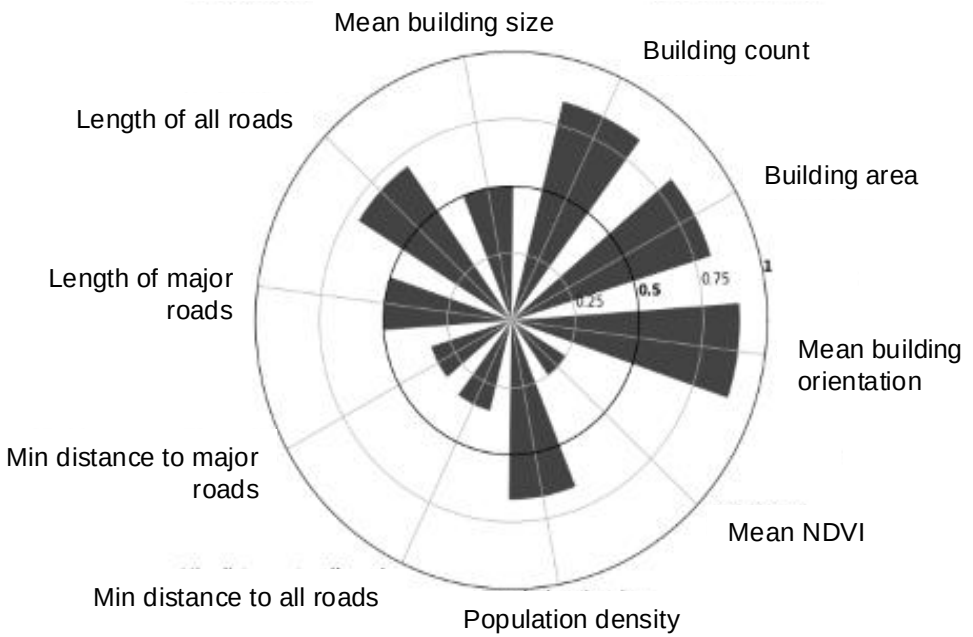
- Dark dense vegetation (5%)
- Water (2%)
- Empty land (10%)
- Buildings surrounded by vegetation (13%)
- Light vegetation (6%)
- Roads and sparse-moderately populated areas (36%)
- Densely populated areas, <36° building orientation (21%)
- Densely populated areas, >36° building orientation (7%)

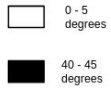
Dark dense vegetation





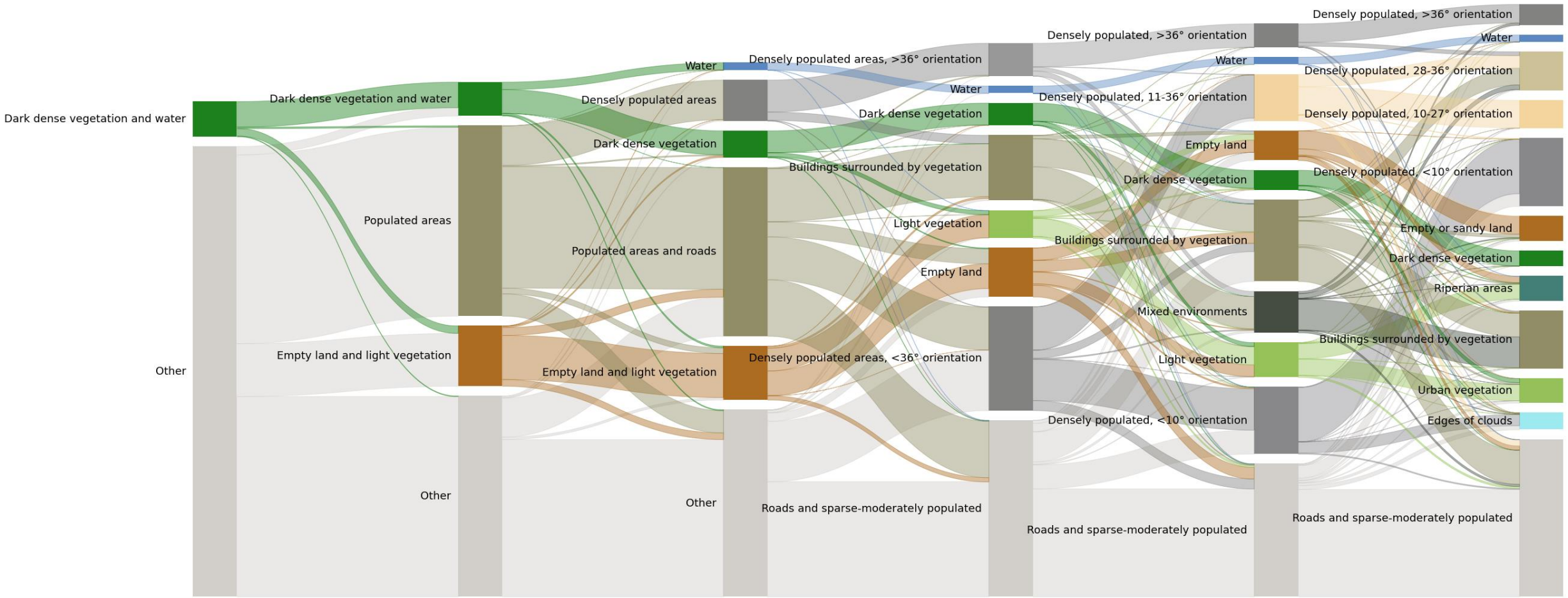
Densely populated areas with building orientation that deviates from north/south



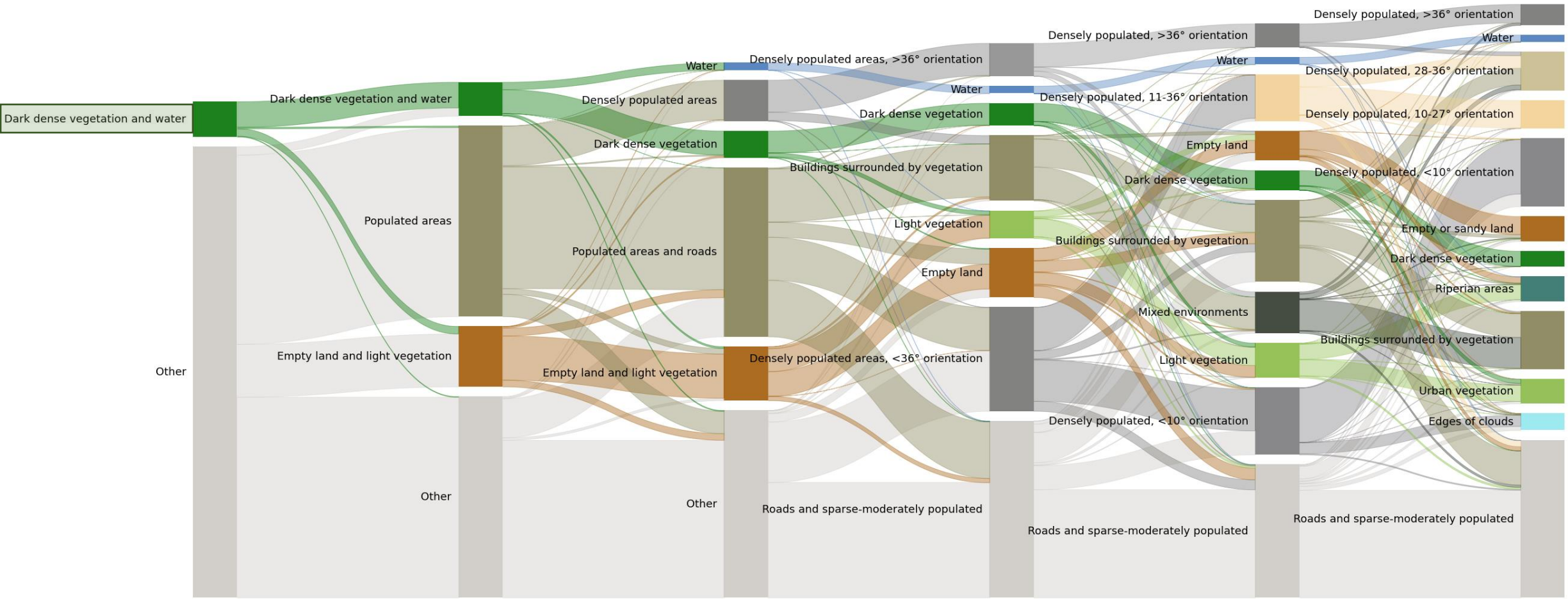


Ashaiman, GAMA

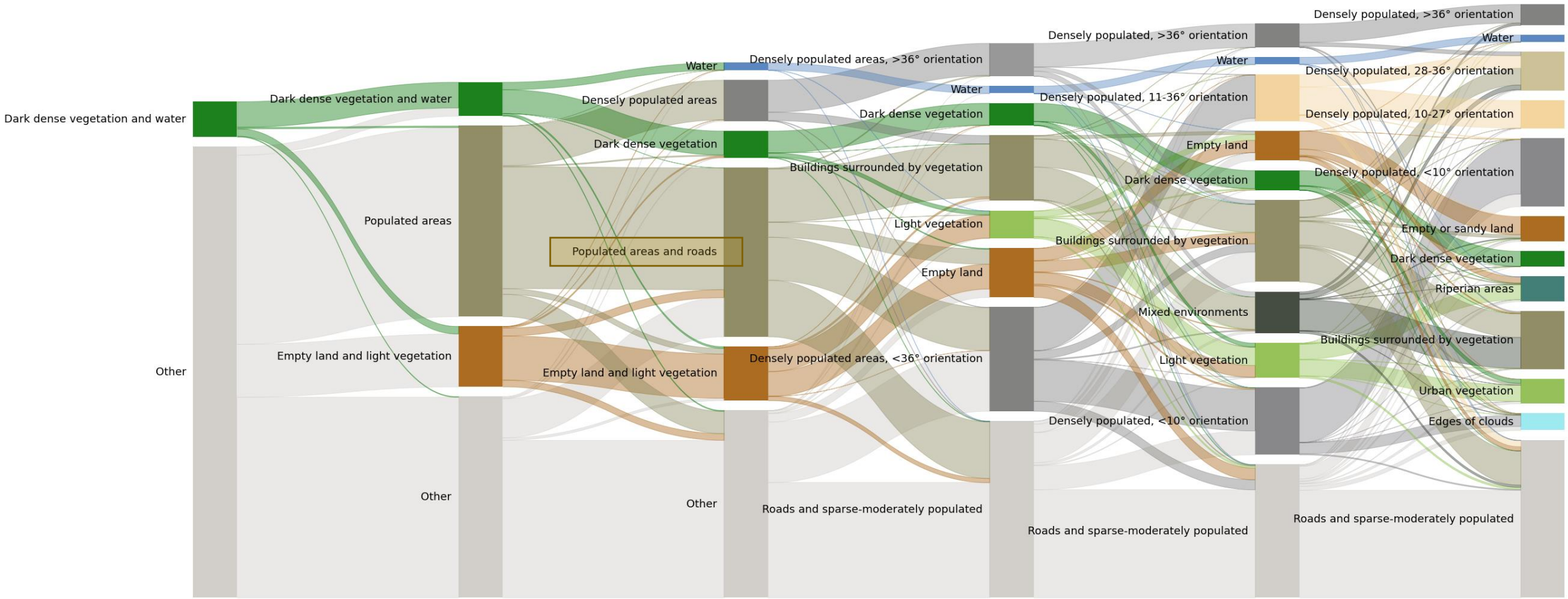
Cluster development



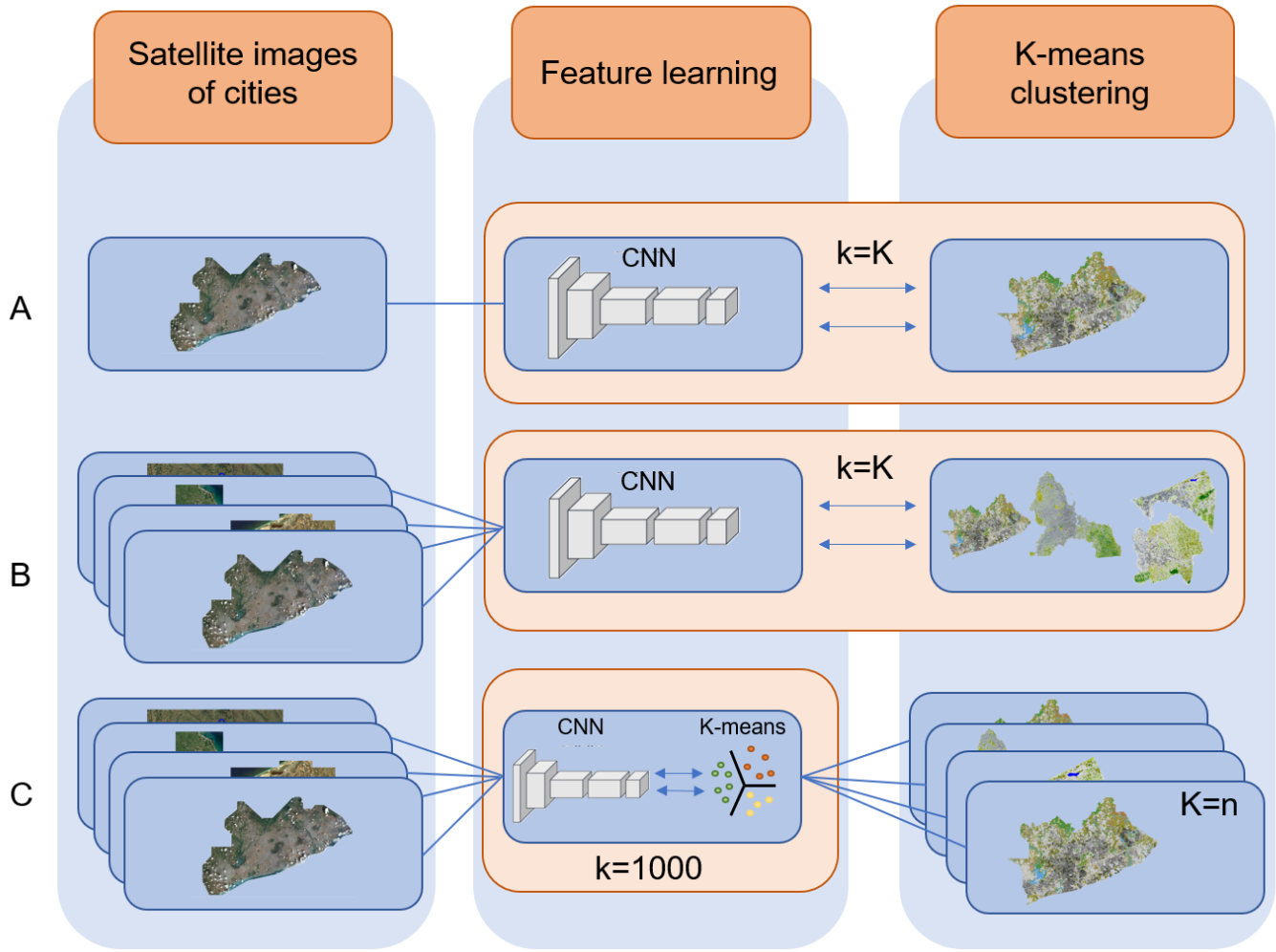
Cluster development



Cluster development

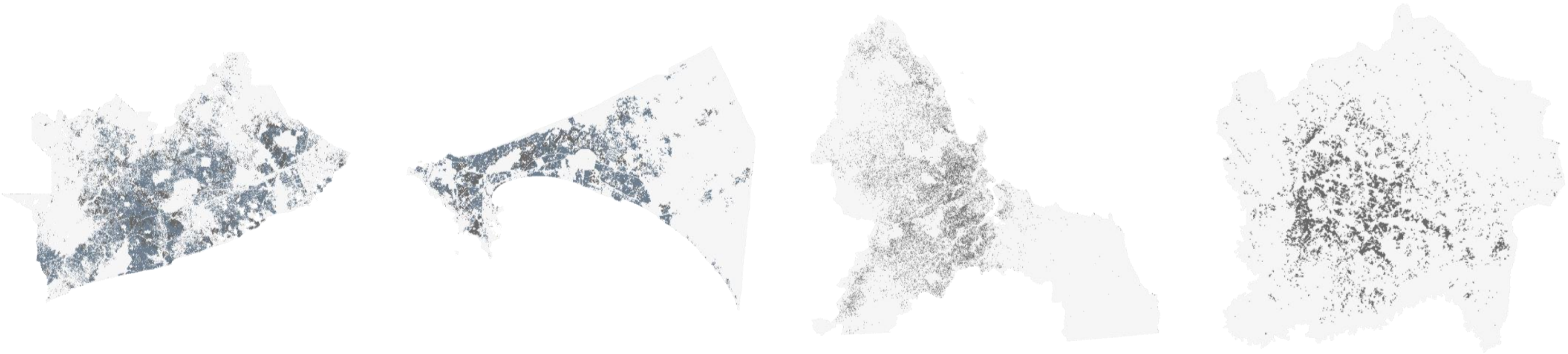


Moving from one city to multiple cities

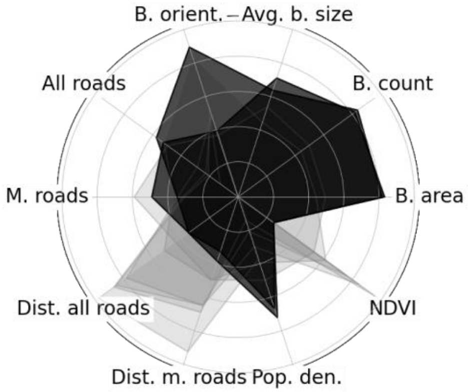


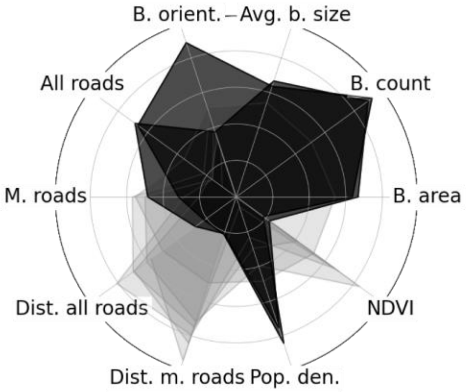
Potential for tracking urban change

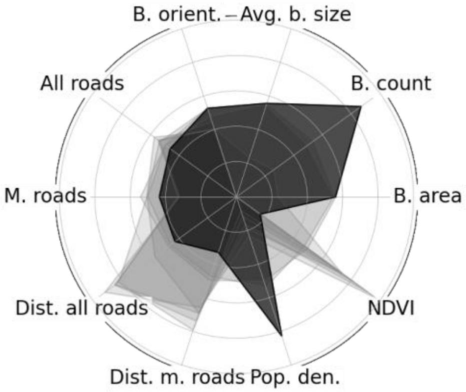
- Comparative study across different cities in East and West Africa

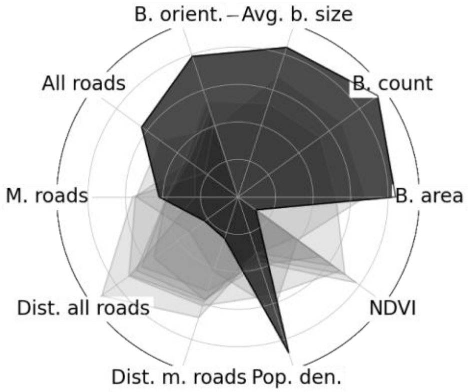


Accra (Ghana) Dakar (Senegal) Dar es Salaam (Tanzania) Kigali (Rwanda)









Any questions?

(antje.metzler18@imperial.ac.uk)

bmetzler@turing.ac.uk

Appendix: What is building orientation?

Ashaiman, GAMA