Agenda

1

General idea

The origin of CNNs

2

Convolution

How does a CNN work?

3

CNN architecture

Example VGGnet

EURO AUSTRIA

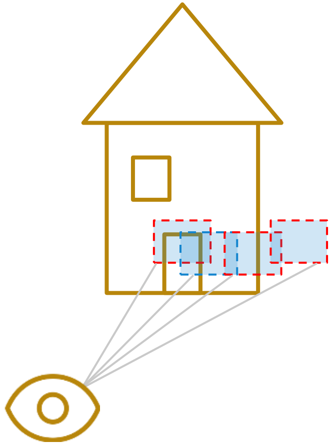
General idea

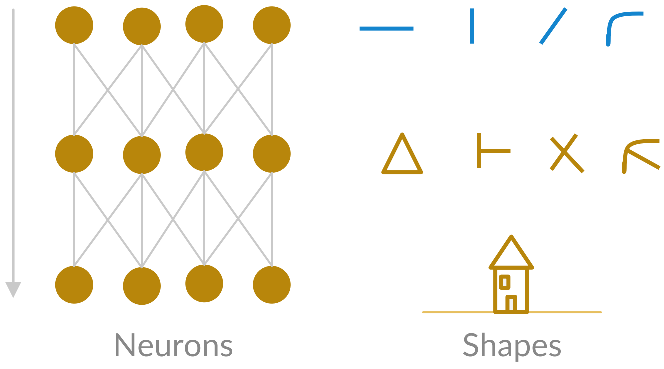
The origin of CNNs

The origin of CNNs

- Used for image recognition since 1980s
- Inspired by the brain's visual cortex
    - Neurons in visual cortex have a small local receptive field
    - Receptive fields of different neurons overlap
    - Together they tile the whole visual field
    - Some neurons only react to specific shapes
    - Some neurons react to more complex shapes from lower levels
- Powerful architecture of lower and higher-level neurons to detect complex patterns

Source: Géron (2019)

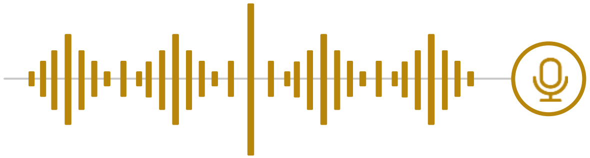




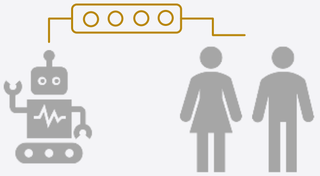
Fields of application

Image detection

Voice recognition



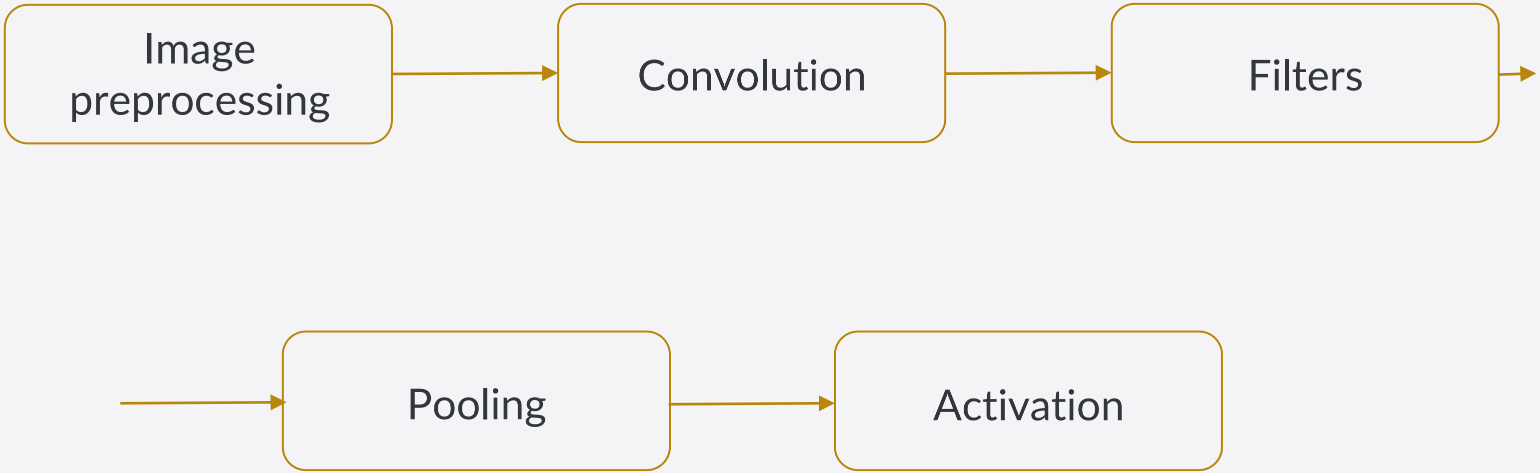
Natural language processing (NLP)



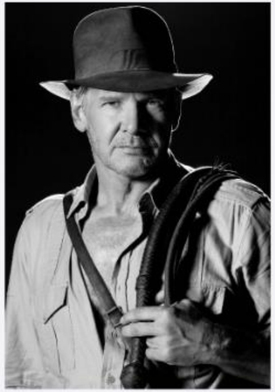
Convolution

How a CNN works

How does a CNN work?



What does a computer see?



input image
3600 x 2400

resized image
36 x 24

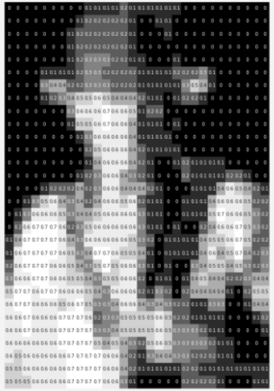
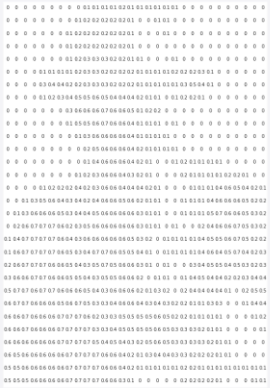


Image as matrix
of numbers [0, 1]

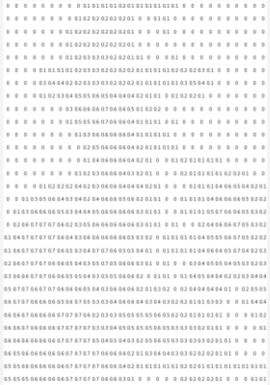


Matrix of numbers
[0, 1]
(864 values)

Source:
https://www.youtube.com/watch?v=iaSUYvmCek
l

Task in computer vision

input image



matrix of numbers

Classification

Harrison Ford
[0.8]

Sean Connery
[0.1]

Roger Moore
[0.05]

Tom Cruise
[0.05]

prediction

Why not simply use a deep network with fully connected layers?

Indie resized to 100 x 100 ...

The picture has 10.000 pixels

With a 1.000 neuron input layer ...

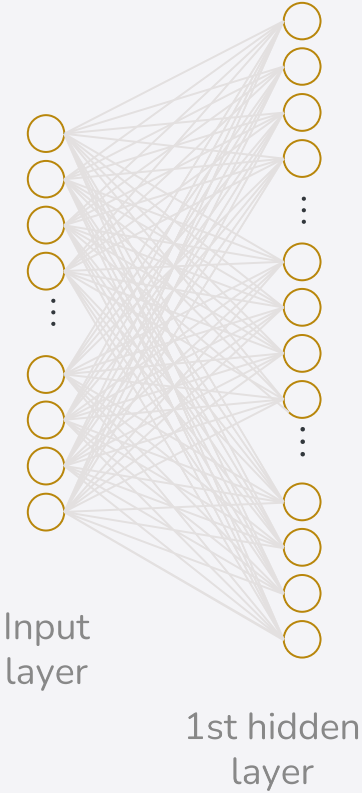
and a fully connected 1st hidden layer, ...

this first operation amounts to a total of 10 million connections (weights, parameters).

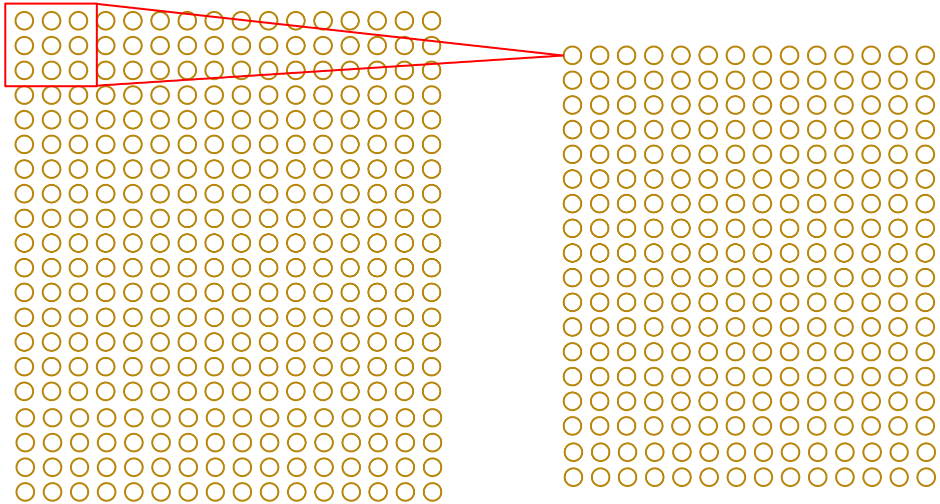
And that is just the 1st layer!

For large images, a deep neural network breaks down.

AND we do not capture the spatial information of the pixels



The convolution layer – using spatial structure



Input image
1 neuron for 1 pixel

Hidden layer
1 neuron for all pixels

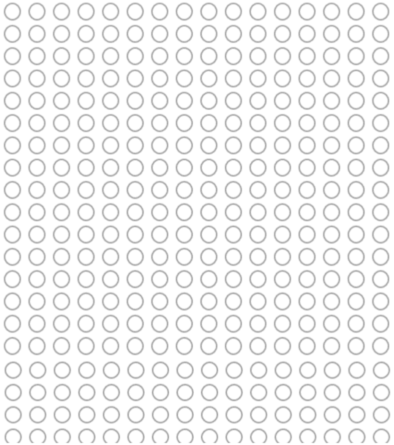
Idea: connect smaller sections of the input image to respective neuron in the hidden layer.

Receptive field (FILTER) marks a specific area in the input image.

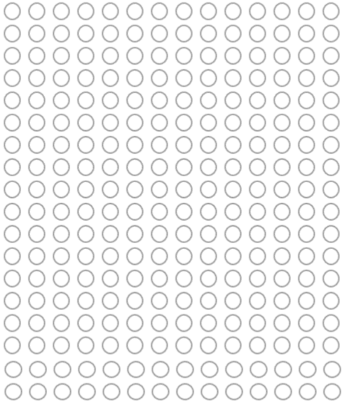
Use a sliding window to define the connections.

The GOAL: How to *weight* the FILTER to detect particular features in the image?

The convolution layer – using spatial structure



Input image
1 neuron for 1 pixel



Hidden layer
1 neuron for all pixels

Apply a set of weights – a filter – to extract local features

Use multiple filters to extract different features

Spatially share parameters of each filter

EURO AUSTRIA

Element-wise multiply and the outputs

part of input image

filter

(1x1) + (3x2) + (5x2) + (2x1) = 19

Application of a filter

Convolutional layer: Connection between neurons and only those pixels within their *receptive field*.

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

image

x

1	0	1
0	1	0
1	0	1

filter

=

4	3	4
2	4	3
2	3	4

feature map

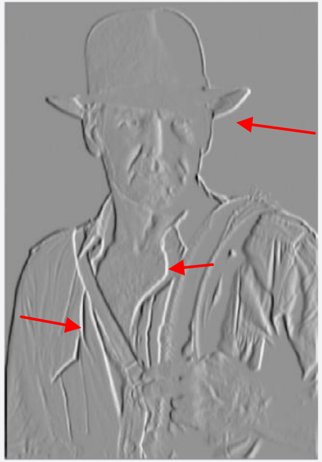
Applying different filters

Horizontal edge detection

-1	-2	-1
0	0	0
1	2	1

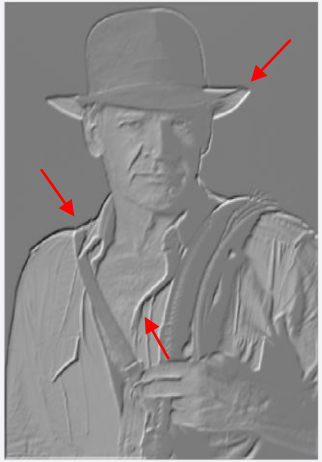
vertical edge detection

-1	0	1
-2	0	2
-1	0	1



mixed edge detection

0	-2	0
-2	1	2
0	2	0



Padding

Adding additional space to preserve the same height and width of previous layer.

zero padding

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

image

x

1	0	1
0	1	0
1	0	1

filter

=

2	2	3	1	1
1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
0	2	2	1	1

feature map

Also, different kinds of paddings possible (i.e., One padding).

Highlight pixels at the edges of image.

Using a larger stride

The shift from one receptive field to the next one is called stride.

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

image

x

1	0	1
0	1	0
1	0	1

filter

=

4	4
2	4

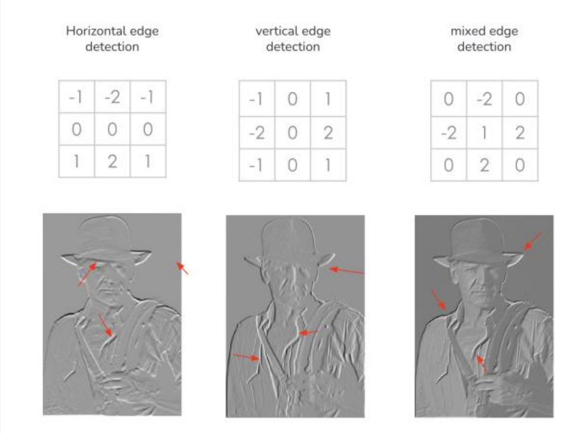
feature map

Connect larger input to smaller layer.

Reduction of the model's computational complexity.

Stacking multiple feature maps

Applying different filters results in different feature maps



-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

0	-2	0
-2	1	2
0	2	0

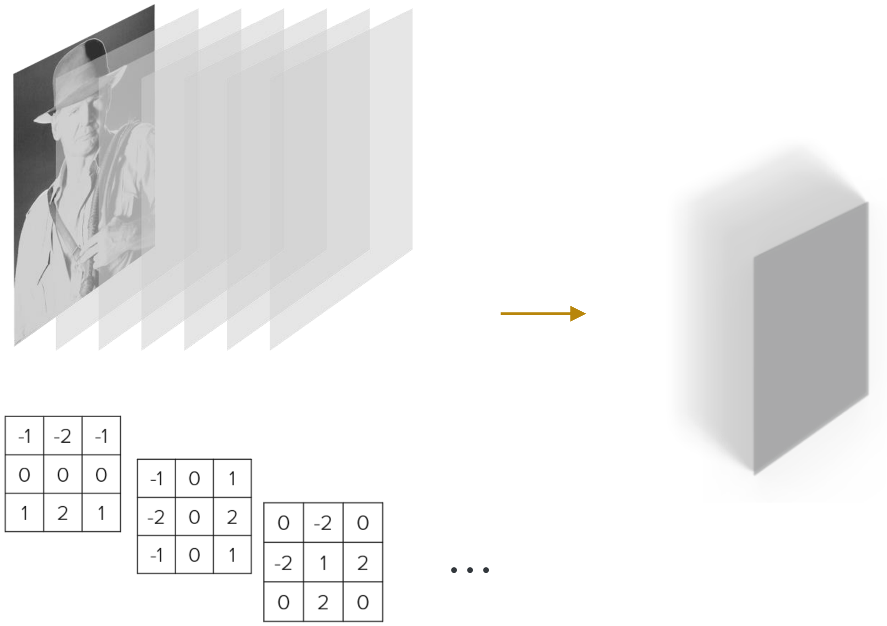
. . .

Convolutional layers with multiple feature maps

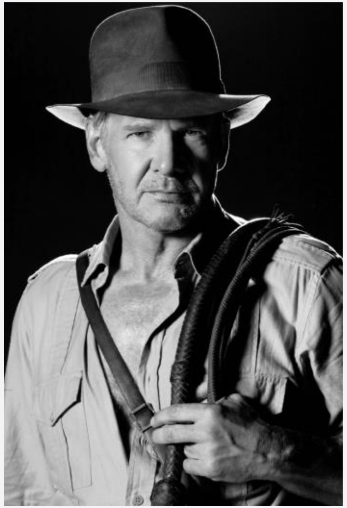
Representation in 3D

Stacking multiple feature maps

- Number and size of filters in each convolution layer are set by design
- Filters initialized at random and then learned (fwd pass, backward prop)
- Convolutional layer learns most useful filters automatically during training for its task
- Layers after this will learn to combine them to more complex patterns



Can you spot the difference?



Full sized image

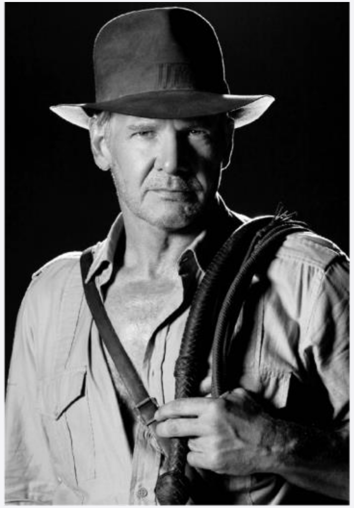


Image subsampled by pooling

Pooling layers

Subsample (i.e., shrink) the input image to reduce the computational load (memory usage, number of parameters)

1	1	2	0	4
0	1	7	1	0
0	8	1	1	1
9	0	1	3	5
0	4	1	0	0

image

Stride = 2
Filter = 3x3

Max pooling

8	7
9	5

feature map

Set filter size, stride and padding as for convolution layer.
No weights attached.

The layer aggregates input with an aggregation function (i.e., max or mean).

Activation by non-linearity

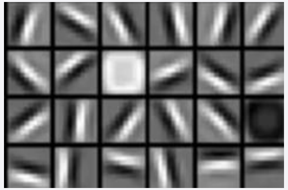
- Apply after every convolution operation
- rectified linear unit (ReLU)
- $f(x)=MAX\ (0,x)$
- pixel-by-pixel operation that replaces all negative values by zero

vertical edges activated

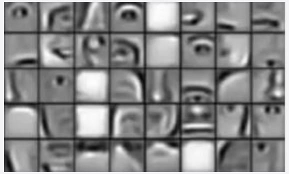
CNN Architecture

Example VGGNET

From basic to detailed features

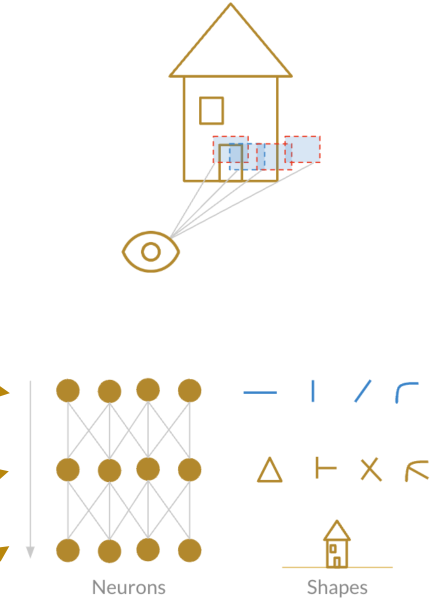


Low level features
(1st Conv Layers)



Mid level features
(...)

High level features
(Last Conv Layers)



Remember the visual cortex?

VGGNET

- Invented by Simonyan and Zisserman from Visual Geometry Group (VGG) at University of Oxford in 2014 [1]
- Large Scale Visual Recognition
- Fixed filter size of 3×3 and the stride of 1
- Different versions (VGG16, VGG19, etc.)
- Why? Reduce the # of parameters in the CONV layers and improve on training time

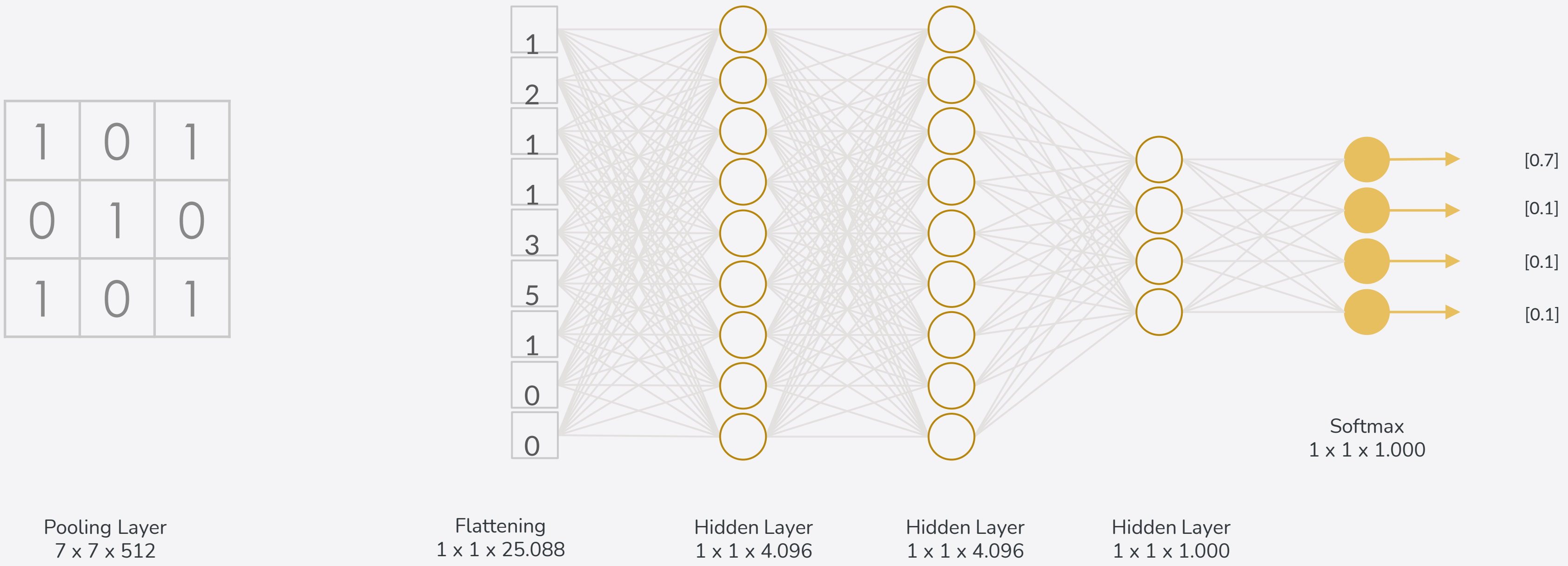
[1] K. S. a. A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in International Conference on Learning Representations (ICLR), San Diego, 2015.

VGGNET 16

Each CNN layer learns filters of increasing complexity.

- Basic feature detection (edges, corners, etc.)
- Parts of objects (for faces i.e., eyes, noses, etc.)
- Higher representations (recognize full objects, in different shapes and positions)

A closer look on final prediction



STAY IN TOUCH

EuroCC Austria

@eurocc_austria

eurocc-austria.at

EURO
AUSTRIA

THANK YOU

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia