Convolutional Neural Networks

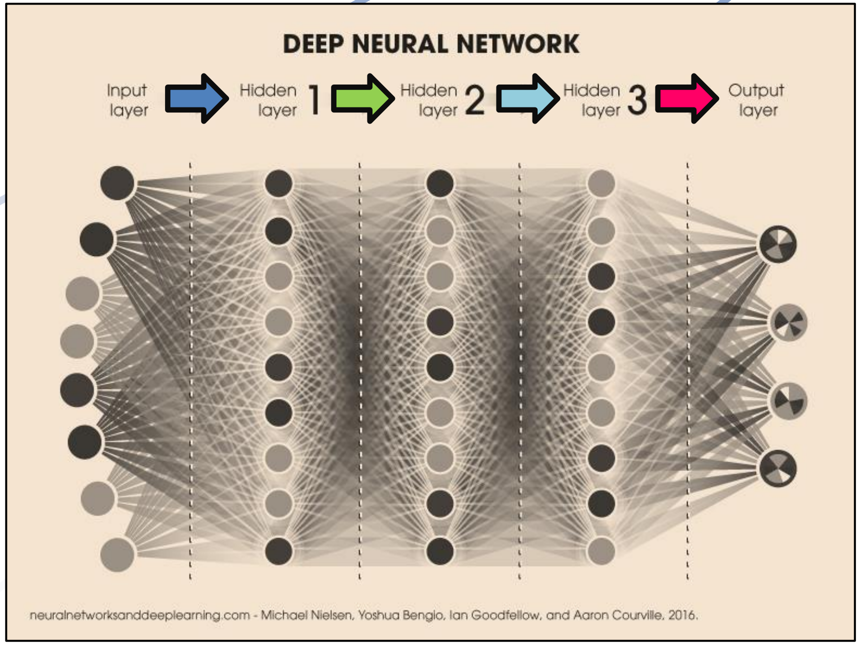
Contributor: Pantelis I. Kaplanoglou
Presenter: Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@aiia.csd.auth.gr
www.multidrone.eu
Presentation version 1.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Deep Neural Networks

Definition

- *Deep Neural Networks (DNNs)* have a count of layers (depth) $L \geq 3$.
  - There are multiple hidden layers with regard to the MLP reference model.



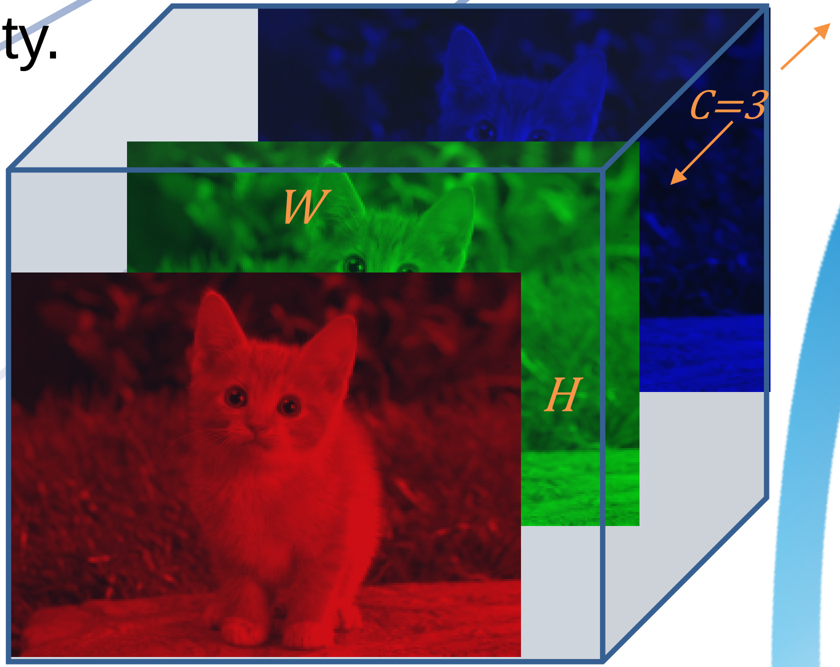
Deep Neural Network with $L = 4$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Deep Neural Networks

Applied for Computer Vision tasks

- There is a vast number of pixel features, e.g., $H \times W \times 3$ color image features for an $H \times W$ RGB image.
    - Problem: Increased computational complexity.
    - Solution: Convolutional Neural Networks (CNNs) that employ sparse connectivity and weight replication.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Long history with recent success

- Foundations: Rummelhart, Hinton, Williams (1985) described a solution to the *T-C problem*.

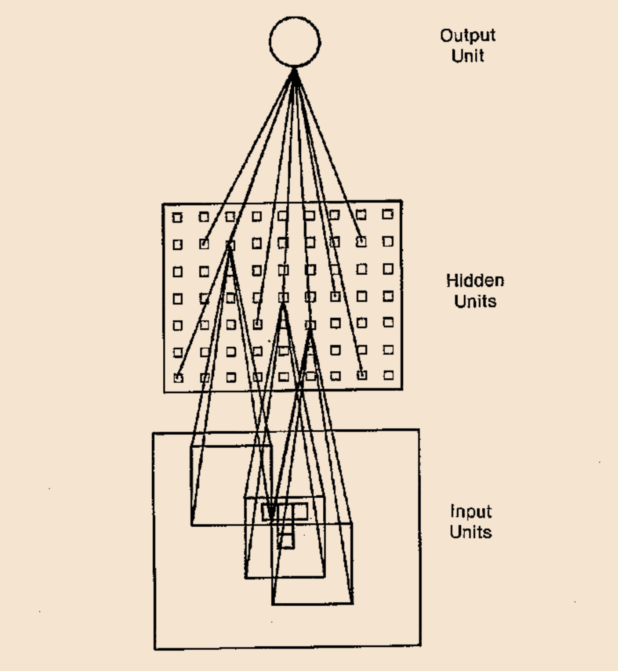


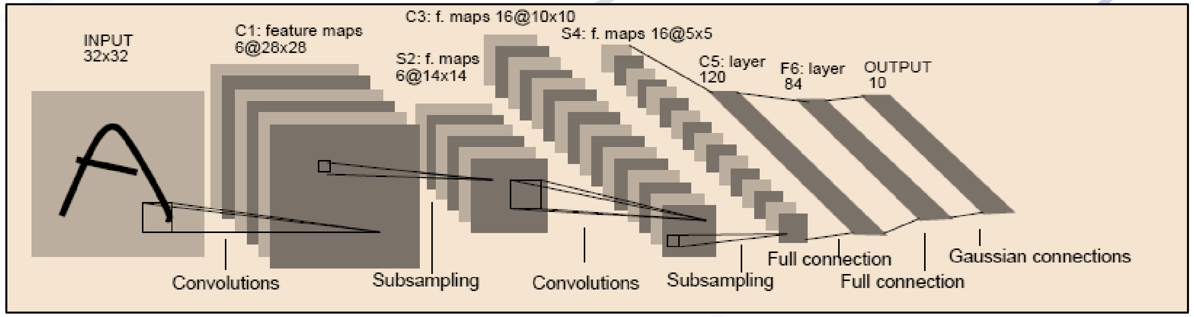
FIGURE 14. The network for solving the T-C problem. See text for explanation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Long history with recent success

- Foundations: Rummelhart, Hinton, Williams (1985) described a solution to the *T-C problem.*
- First CNN: LeCun, Bottou, Bengio, Haffner (1989) proposed convolutional layer for character recognition.



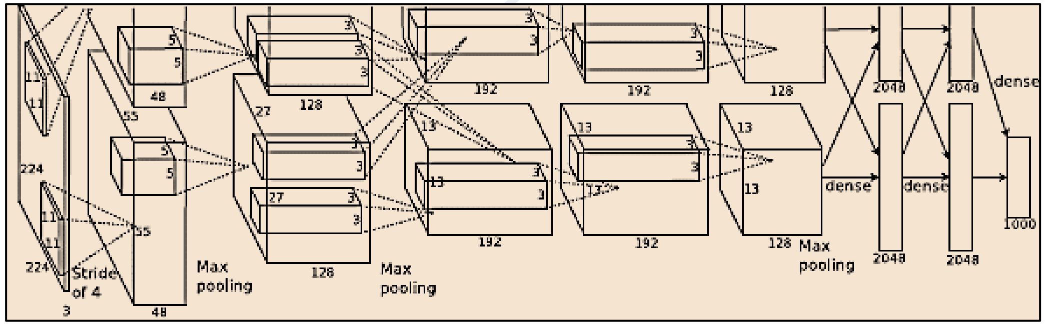
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Long history with recent success

MultiDrone

- Foundations: Rummelhart, Hinton, Williams (1985) described a solution to the *T-C problem*
- First CNN: LeCun, Bottou, Bengio, Haffner (1989) proposed convolutional layer for character recognition
- Success: Krizhevsky, Sutskever, Hinton (2012) significantly outperformed the state-of-the-art for image classification in the *ImageNet* competition.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Research Advances

The ImageNet competition

- The ImageNet Large Scale Image Recognition Challenge (ILSVRC) is an annual competition started in 2010.
- Large scale image datasets are used that have over a million images of natural scenes and objects.
- Since 2012, research in the CNN area produced significant advances, some through the ILSVRC competition.
  - The CNN that won the classification task (CLS) in 2015, surpassed the human classification accuracy on the same test set.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks
Notations

- Standard CNN input is a $H \times W \times C$ multi-channel (color) image represented by 3D tensor $\boldsymbol{X}$.
- The multichannel image is a tensor $\boldsymbol{X}$ or a 3D signal $x(i,j,r): i = 1, \dots, H,\ j = 1, \dots, W, r = 1, \dots, C$.
  - The number of channels C is sometimes called *depth*.
- A multichannel image block of dimensions $h_1 \times h_2$: $h_1 = 2q_1 + 1, h_2 = 2q_2 + 1$ centered at $[i,j]^T$ is noted as the 3D tensor $\boldsymbol{X}_{ij}$.
- The 2D matrix $\boldsymbol{X}_{ij}(r)$ is a slice of $\boldsymbol{X}_{ij}$ that contains the values of channel r (e.g. intensity of red/green/blue) for the $h_1 \times h_2$ image block centered at $[i,j]^T$.

Matrix of feature (channel) r for the image block centered at $[i,j]^T$:

$$\boldsymbol{X}_{ij}(r) = [x(k_1,k_2,r): k_1 = i - q_1, \dots, i + q_1, k_2 = j - q_2, j + q_2]$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Reference CNN Classifier

The AlexNet CNN

Convolutional Module 1
- Convolutional Layer 1
- Max Pooling 1
- Local Response Normalization 1

Convolutional Module 2
- Convolutional Layer 2
- Max Pooling 2
- Local Response Normalization 2

Convolutional Module 3
- Convolutional Layer 3
- Local Response Normalization 3

Convolutional Module 4
- Convolutional Layer 4
- Local Response Normalization 4

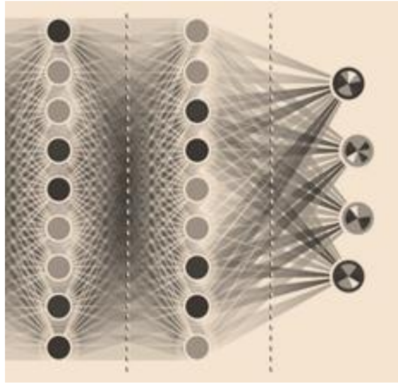
Convolutional Module 5
- Convolutional Layer 5
- Max Pooling 5
- Local Response Normalization 5

Fully Connected Module 1
- FC Layer 6
- Dropout

Fully Connected Module 2
- FC Layer 7
- Dropout

Softmax Output Module
- FC Layer 8
- Softmax Function

- There are several *convolutional layers.*
- There is a classifier that consists of 3 *fully connected* layers.
- Class prediction are given by *softmax* functions.

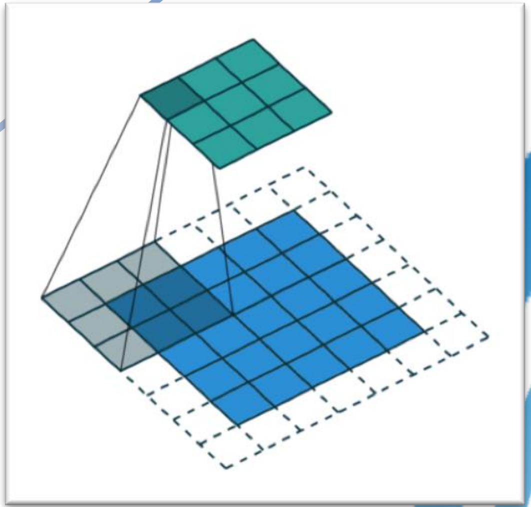


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Characteristics: Sparse Connectivity

- A convolutional neuron implements *sparse connectivity* with a sliding window of dimensions $h_1 \times h_2$ operating locally on the input data.
  The step s of the window motion is called *stride*.
  - Simple notation for *convolutional kernels:* $[h_1 \times h_2/s]$
  - Extended notation for *convolutional kernels:* $[h_1 \times h_2/s|d_{in} \rightarrow d_{out}]$ where d_{in}, d_{out} are the input, output feature depths. For the first layer $d_{in} = C$.



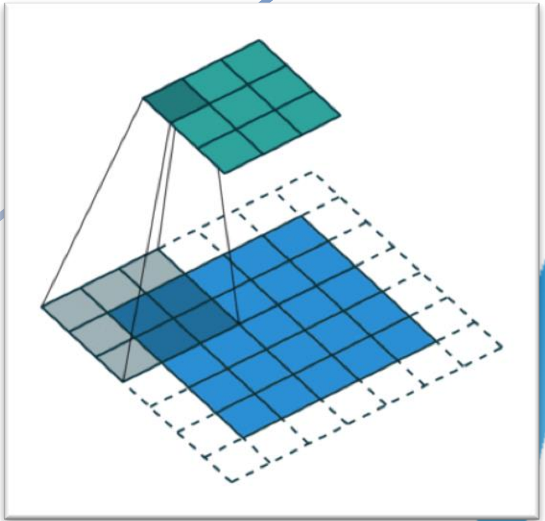
- For 2D image data two different strides s_H, s_V may be used for the horizontal and the vertical sliding window motion.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Characteristics: Local Receptive Fields

- Each neuron has a *local receptive field* that implements a dynamic mapping to a small local region $\mathbf{X}_{ij}$ of the input image $\mathbf{X}$ centered at $[i,j]^T$.



- Padding with zeros at the edges of input $\mathbf{X}$, so that the convolution operator can operate on the entire input image domain.

- Square receptive fields have $h_1 = h_2 = h$. They can overlap when the stride
of the convolution operation is $1 \leq s < h$.

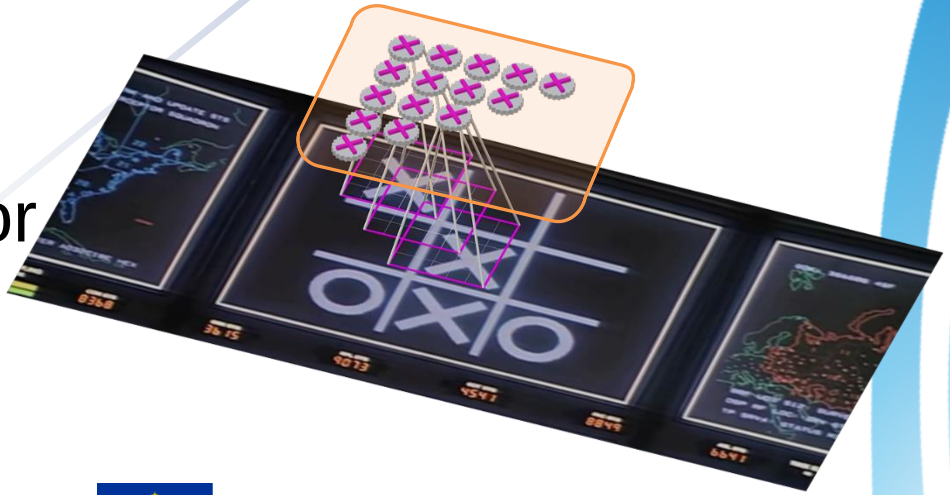
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks
Characteristics: Weight sharing

- A convolutional layer abstracts multiple neurons that cover all regions of input and share the same synaptic weights.

- *Weight sharing* (or weight replication) is also implemented by the sliding window mechanism.

- The *convolutional kernel* is a 4D tensor of all shared weights noted as $\mathbf{W}$.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

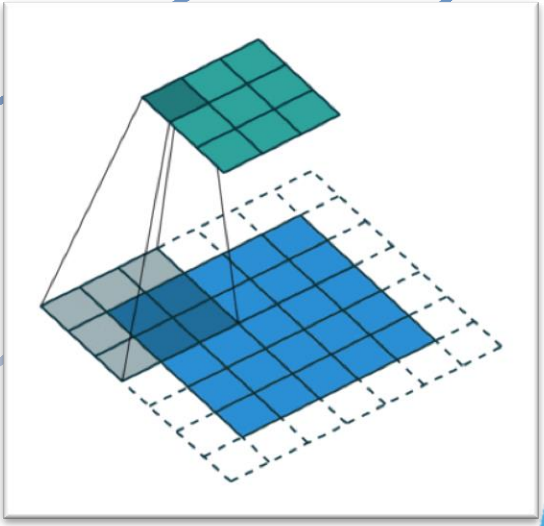
Convolutional Neural Networks

Characteristics: Local Receptive Fields

MultiDrone

- The convolution kernel is given by the 4D tensor:

$$\boldsymbol{W} = [w_{k_1,k_2,r,o}: \quad k_1 = 1,\dots,h_1, k_2 = 1,\dots,h_2, \\ r = 1,\dots,d_{in}, o = 1,\dots,d_{out}]: \boldsymbol{W} \in \mathbb{R}^{h_1 \times h_2 \times d_{in} \times d_{out}}.$$



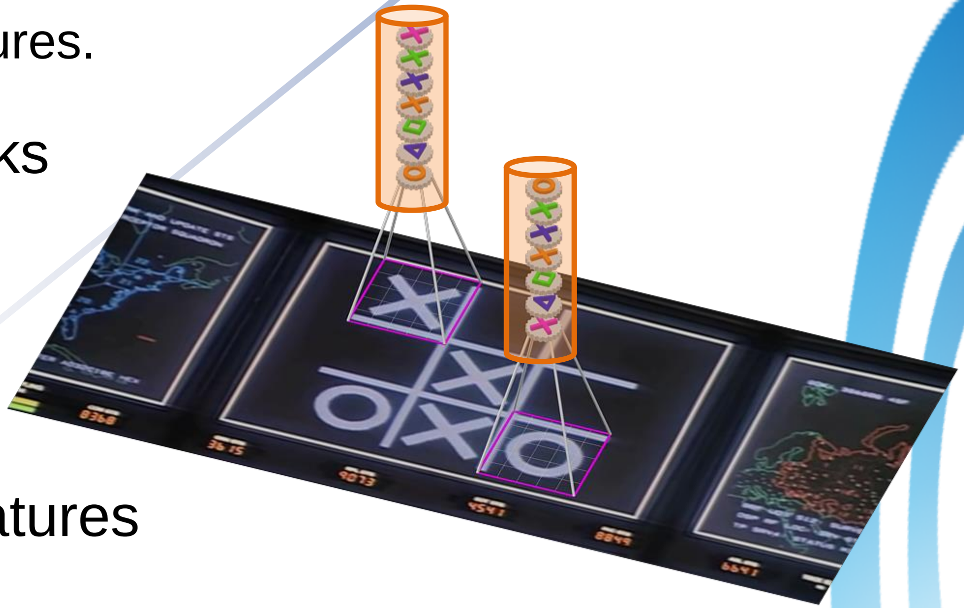
- For specific $r, o,$ the $h_1 \times h_2$ *convolution filters* $\boldsymbol{w}(r,o)$ contain the synaptic weights for the $h_1 \times h_2$ neuron receptive field.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Characteristics: Multiple Local Feature Extractor

- A *convolutional layer* is an abstraction of a column of neurons, that share a common receptive field, operating on the same region of input $\boldsymbol{X}_{ij}$.
  - There are different 3D weight tensors $\boldsymbol{W}(o)$ for each neuron, extracting a different feature o from incoming features of $\boldsymbol{X}_{ij}$.
  - The input region $\boldsymbol{X}_{ij}$ has R,G,B intensities as features.
- In each layer, there are $n \times m$ local image blocks of size $h_1 \times h_2$, depending on stride *s*.
  - If $s = 1$ then $n = H, m = W$.
- A d_{out}-dimensional *feature descriptor* vector: $\boldsymbol{y}_{ij} = [y_{ij}(o): o = 1, \dots, d_{out}]^T$ holds all output features for an input local block $[i, j]^T$.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

Activation map spatial dimensions

- For a centered convolution operation with squared input image dimensions $n_{in} \times n_{in}$, square filter window $h \times h$, stride s and padding n_{pad}, the dimension of the $n_{out} \times n_{out}$ output image (activation map) is:

$$\boxed{n_{out} = \frac{n_{in} - h + 2\, n_{pad}}{s} + 1\ , \qquad h = 2q + 1\,, \qquad n_{in}, n_{out}, q, n_{pad} \in \mathbb{N},}$$

- Specific input sizes are chosen due to this expression. Examples:
  1. AlexNet CNN with an input image resolution of 227×227 pixels and a convolutional filter

     $[11 \times 11/2]$ without zero padding $(\mathrm{n_{pad}} = 0\,)$ has $n_{out} = \frac{227-11+0}{4} + 1 = 55.$
  2. An input image resolution of 223×223 pixels can have $n_{out} = \frac{223-7+2}{2} + 1 = 110,$

     when a $[7 \times 7\ /2]$ convolutional filter is used with zero padding $n_{pad} = 1.$

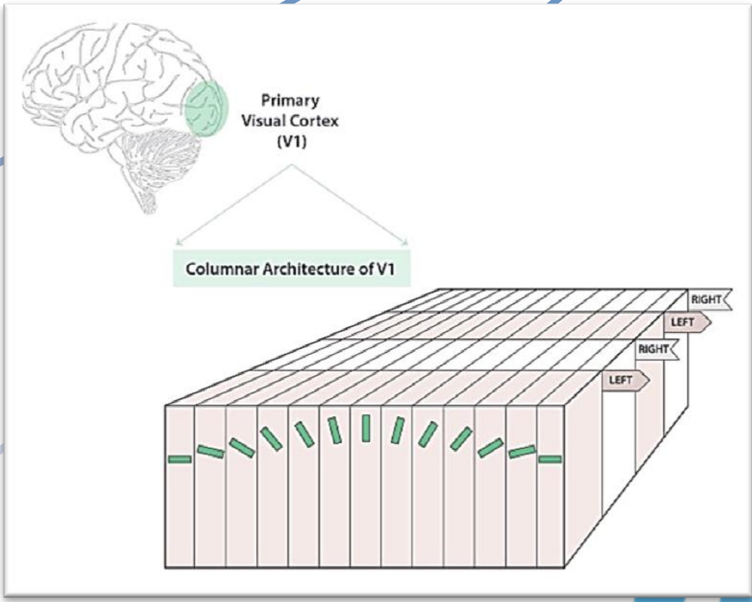
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

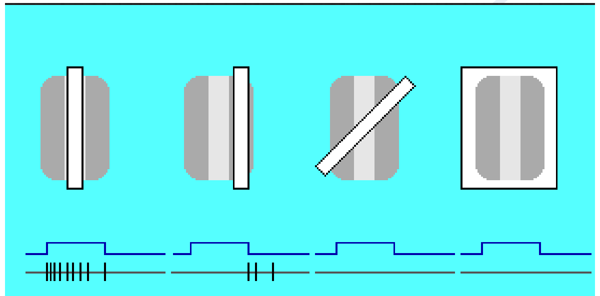
Inspiration from visual neurons

Biological V1 Hypercolumn

MultiDrone

- CNNs were inspired by brain neurons in the mammalian primary visual cortex (V1).
- V1 cells are mapped to the same local region of the retina, forming *hypercolumns*.
- V1 simple cells detect image lines and are sensitive to orientation.





This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

Synaptic summation

- The classic neurons have a weighted sum on vectors (1D) of features values.

Perceptron Neuron Summation

$$u = w_0 + \sum_{i=1}^{N} w_i x_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}.$$

- Convolutional neurons implement the sum u with a *convolution operation* $\boldsymbol{W} * \boldsymbol{X}$ (or element-wise matrix multiplication $\boldsymbol{W} \odot \boldsymbol{X}_{ij}$) on 2D single feature image (matrix) $\boldsymbol{X}$, 2D kernel (coefficient matrix) $\boldsymbol{W}$ plus a scalar bias b:

Convolutional Neuron Summation

$$u(i,j) = b + \sum_{k_1=1}^{h_1} \sum_{k_2=1}^{h_2} w(k_1, k_2) x(i - k_1, j - k_2) = b + (\boldsymbol{W} * \boldsymbol{X})(i,j) = b + \boldsymbol{W} \odot \boldsymbol{X}_{ij}.$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

Synaptic summation centered on input

Convolutional Neuron Summation

$$u(i,j) = b + \sum_{k_1=1}^{h_1} \sum_{k_2=1}^{h_2} w(k_1,k_2)x(i-k_1, j-k_2) = b + (\boldsymbol{X} * \boldsymbol{W})(i,j).$$

- Using an odd convolution window with $h_1 = 2q_1 + 1$ and $h_2 = 2q_2 + 1$, the filter output is mapped to the filter window center at location $[i,j]^T$.

Convolutional Neuron Summation - Centered on Input:

$$u(i,j) = b + \sum_{k_1=-q1}^{q_1} \sum_{k_2=-q2}^{q_2} w(k_1,k_2)x(i-k_1, j-k_2) = b + (\boldsymbol{X} * \boldsymbol{W})(i,j).$$

For a convolutional filter $h_1 \times h_2$ where $h_1 = 2q_1 + 1$ and $h_2 = 2q_2 + 1$.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

Single feature extracted from grayscale images

- For a convolutional layer l with an activation function $f_l(\cdot)$, a single input feature (e.g., grayscale image intensities) and a single feature output:

Output feature extracted from image block centered at $[i,j]^T$

$$y^{(l)}(i,j) = f_l\left(b^{(l)} + \sum_{k_1=1}^{h_1^{(l)}} \sum_{k_2=1}^{h_2^{(l)}} w^{(l)}(k_1,k_2)x^{(l)}(i-k_2, j-k_2)\right).$$

- The *activation map* $\boldsymbol{A}^{(l)}(o), = \boldsymbol{Y}^{(l)}, o = 1$ is a table of the output feature values for all $n \times m$ regions.

Convolutional Layer Activation Map (2D matrix) from single input feature

$$a_{ij}^{(l)}(1) = f_l\left(b^{(l)} + \boldsymbol{W}^{(l)}(1,1) * \boldsymbol{X}_{ij}^{(l)}(1)\right)$$

$\boldsymbol{W}^{(l)}(1,1)$: *2D filter for the 1st output feature,* $\boldsymbol{X}_{ij}$*(1): grayscale intensities of the local block at* $[i,j]^T$

$$\boldsymbol{A}^{(l)}(1) = \left[a_{ij}^{(l)}: i = 1,..,n^{(l)}, j = 1,..,m^{(l)}\right]$$

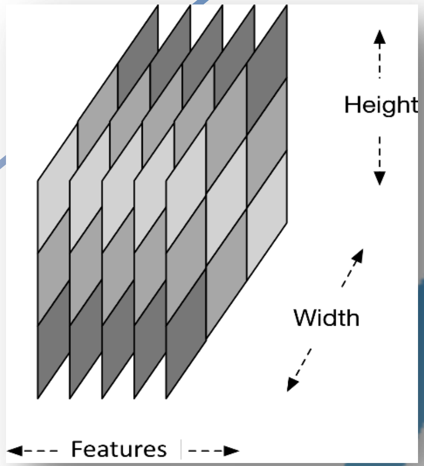
The activation map for the 1st output feature at layer l is the matrix $\boldsymbol{A}^{(l)}(1)$ that contains $a_{ij}^{(l)}$ across all regions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

For RGB images

- For a convolutional layer l with an activation function $f_l(\cdot)$, multiple incoming features d_{in} and one single output feature



Multiple input features to single feature o transformation

$$y^{(l)}(i,j,o) = f_l\left(b^{(l)} + \sum_{r=1}^{d_{in}} \sum_{k_1=-q_1}^{q_1^{(l)}} \sum_{k_2=-q_2}^{q_2^{(l)}} w^{(l)}(k_1,k_2,r,o)\, x^{(l)}(i-k_1, j-k_2, r) \right)$$

Convolutional Layer Activation Volume (3D tensor)

$$a_{ij}^{(l)}(o) = f_l\left(b^{(l)}(o) + \sum_{r=1}^{d_{in}} \boldsymbol{W}^{(l)}(r,o) * \boldsymbol{X}_{ij}^{(l)}(r)\right) \qquad \boldsymbol{A}^{(l)} = \left[a_{ij}^{(l)}(o): i = 1,..,n^{(l)}, j = 1,..,m^{(l)}, o = 1,\dots,d_{out}\right]$$

where $\boldsymbol{A}^{(l)}$ is the activation volume for the convolutional layer l, $\boldsymbol{W}^{(l)}(r,o)$ is a 2D slice of the convolutional kernel $\boldsymbol{W}^{(l)} \in \mathbb{R}^{h_1 \times h_2 \times d_{in} \times d_{out}}$ for input feature r and output feature o, $b^{(l)}(o)$ a scalar bias and

$\boldsymbol{X}_{ij}^{(l)}(r)$ a region of input feature r centered at $[i,j]^T$, e.g. $\boldsymbol{X}^{(1)}(1)$ the R channel of an image $d_{in} = C = 3$.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

As many-to-many feature transformation

- The convolutional layer is a feature transformation operation $\boldsymbol{A}^{(l-1)} \rightarrow \boldsymbol{A}^{(l)}$ $(d_{in} \rightarrow d_{out})$ that extracts multiple output features from multiple input features.
- The convolutional kernel is a 4D tensor of weights. It can be considered a collection of 2D filters for each input-to-output feature correspondence.

Convolutional Layer

$$a^{(l)}(i,j,o) = f_l\left(b^{(l)}(o) + \sum_{r=1}^{d_{in}} \boldsymbol{W}^{(l)}(r,o) * \boldsymbol{A}_{ij}^{(l-1)}(r)\right)$$

$\boldsymbol{A}^{(l)}= [\, a^{(l)}(i,j,o), i = 1, \dots, n^{(l)}, j = 1, \dots, m^{(l)}, o = 1, .., d_{out}^{(l)} \,],\ \boldsymbol{A}^{(l)} \in \mathbb{R}^{n^{(l)} \times m^{(l)} \times d_{out}^{(l)}}$,

$\boldsymbol{A}_{ij}^{(l-1)}(r) \in \mathbb{R}^{n^{(l-1)} \times m^{(l-1)} \times d_{in}^{(l)}}$ where $\boldsymbol{A}_{ij}^{(l-1)}(r) = \boldsymbol{X}^{(l)}(r)$ is a 2D slice of previous layer activation volume for feature r at region $[i,j]^T$ used as input to the layer, $\boldsymbol{W}^{(l)}(r,o)$ is the 2D slice of the convolutional kernel $\boldsymbol{W}^{(l)} \in \mathbb{R}^{h_1^{(l)} \times h_2^{(l)} \times d_{in}^{(l)} \times d_{out}^{(l)}}$ for input feature r and output feature o and $b^{(l)}(o)$ is the scalar bias.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Layer

Basic Activation Functions

- Sigmoid and hyperbolic tangent function are not proper for CNNs, because they lead to the *vanishing gradients* problem.

- Rectifiers are more suitable for activation functions.
  - **ReLU** - Rectified Linear Unit

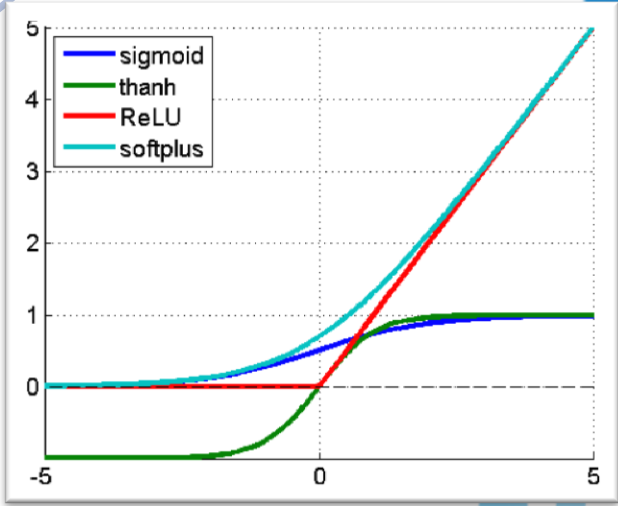
    $y = ReLU(u) = \max\{u, 0\} \quad : \mathbb{R} \rightarrow [0, +\infty)$

  - **ReLU6** - Rectified Linear Unit Bounded by 6

    $y = min\ \{ReLU(u), 6\} = \min\{\max\{u, 0\}, 6\} \quad : \mathbb{R} \rightarrow [0,6]$

  - **Softplus**

    $y = softplus(u) = log(1 + e^u) \quad : \mathbb{R} \rightarrow [0, +\infty)$



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

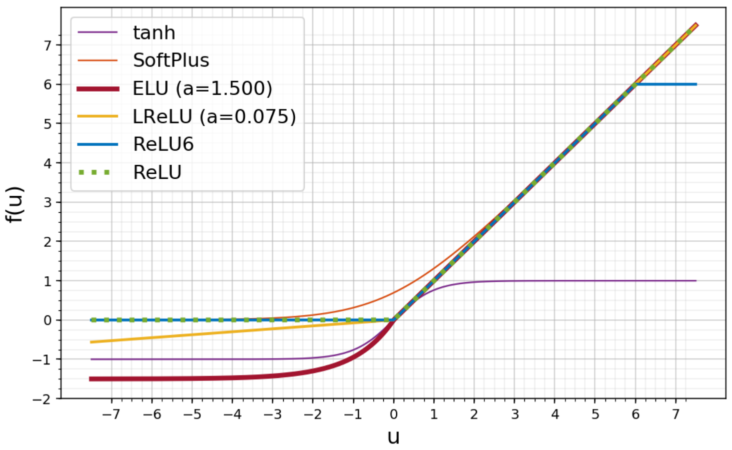
Convolutional Layer

Advanced Activation Functions

- Activation functions with positive range of values have mean activation larger than zero, leading to the *bias shift* problem.
  - Activations of neurons should be capable of cancelling each other at the next layer.
- Advanced activation functions try to mitigate the negative effects of ReLU.
  - **LReLU** - Leaky Rectified Linear Unit (2013)

$$y = LReLU(u) = \begin{cases} ReLU(u) & , u \geq 0 \\ -a \cdot ReLU(-u) & , u < 0 \end{cases} : \mathbb{R} \rightarrow (-\infty, +\infty)$$

  - **PReLU** - Parametric Rectified Linear Unit) (2015)
  - **RReLU** - Randomized Leaky Rectified Linear Unit (2015)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

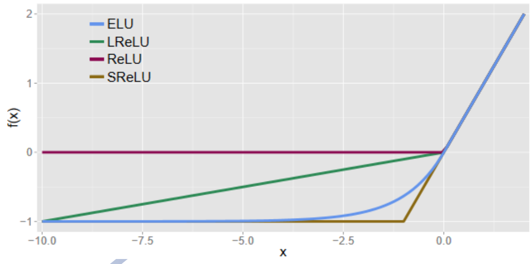
Convolutional Layer

ELU - Exponential Linear Unit (2015)

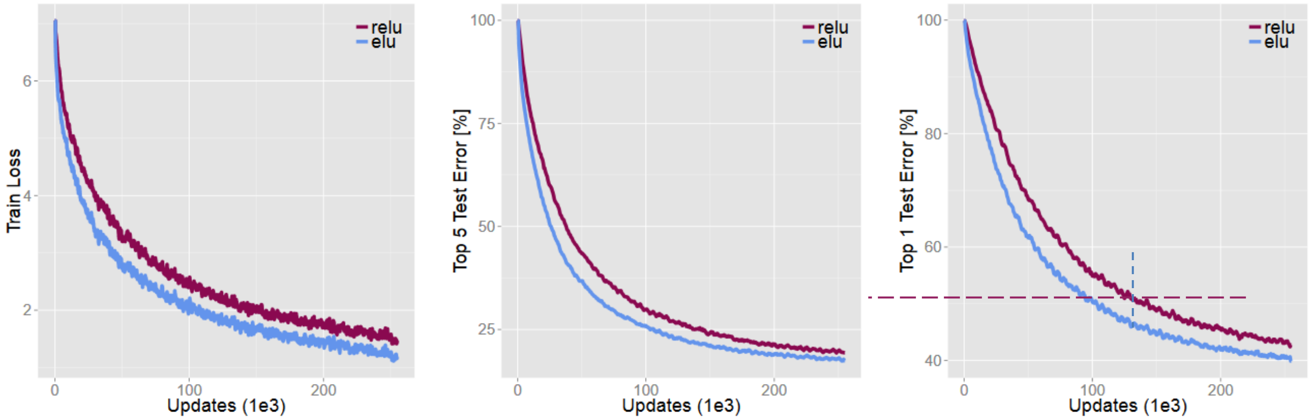
- The ELU activation function aims to mitigate the bias shift problem:

$$y = ELU(u) = \begin{cases} ReLU(u) = \max\{u, 0\} & , u \geq 0 \\ a \cdot (e^{u} - 1) & , u < 0 \end{cases} \quad : \mathbb{R} \rightarrow (-a, +\infty)$$

$$0 < a \leq 1$$



- It achieves better training convergence in shorter amount of time.



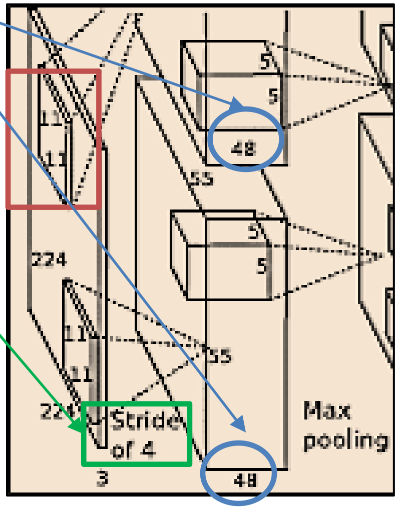
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

As many-to-many feature transformation

- For RGB images there is $d_{in} = 3$ and $d_{out} > 1$ features which are extracted by the first layer with the use of the 4D convolutional kernel.

- The notation for a convolutional layer can be augmented to indicate the input and output features as $[h_1 \times h_2 / s \mid d_{in} \rightarrow d_{out}]$
  - E.g.: AlexNet 1st convolutional layer is $[\mathbf{11 \times 11} \; /\mathbf{4} \mid 3 \rightarrow \mathbf{96}]$



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

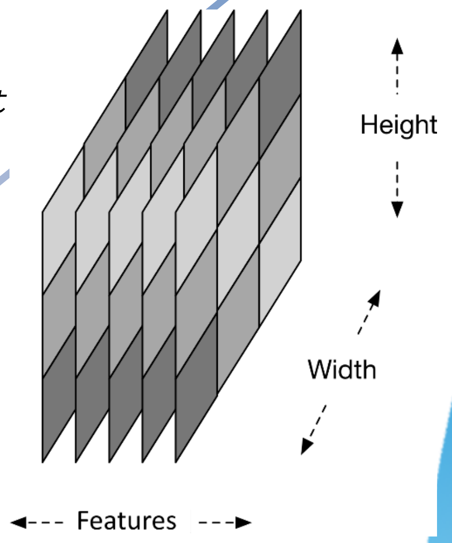
Convolutional Neural Networks

Neural Features Representation

- An input image is an RGB representation of a natural scene.
- An activation volume of a convolutional layer $\boldsymbol{A}^{(l)} \in \mathbb{R}^{n^{(l)} \times m^{(l)} \times d_{out}^{(l)}}$ is a multi-dimensional *neural feature representation* of the input image $\boldsymbol{A}^{(0)} = \boldsymbol{X} \in \mathbb{R}^{H \times W \times 3}$ at layer l.
- Feature depth increases as resolution decreases.
  - First convolutional layer represents *local features.*

    e.g. First convolutional layer in AlexNet outputs 96 features in a 55×55 map.
  - Last convolutional layer represents *global features.*

    e.g. Last convolutional layer in AlexNet output 256 features in 6×6 map.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Neural Features Representation

- Image is downsampled but feature descriptors become more discriminative than original RGB pixel features.

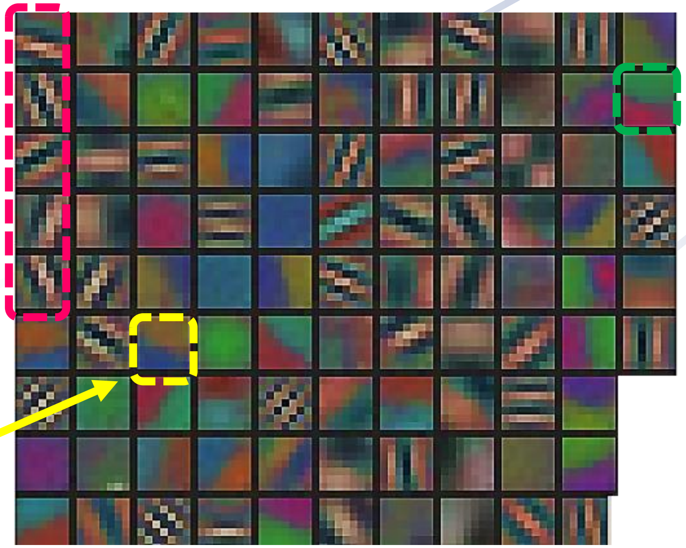
- The features are optimized through gradient descent, in contrast to handcrafted feature extractors like SIFT, HOG, SURF.
  - Performance of learned CNN features has shown superiority in experiments, with a formal explanation.
  - Robustness of learned features is an open research issue.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks
Visualization

- Visualizing the $d_{out} = 96$ features learned from RGB pixels in the 1st layer of the Zeiler & Fergus CNN (ZFNet) shows characteristics of biological vision.
  - ZFNet is an improvement of AlexNet with 1st layer: $[7 \times 7\ /2|\ 3 \rightarrow 96]$
  - Feature visualization indicated poorly trained convolutional kernels in AlexNet.

orientation selectivity found in V1 simple cells



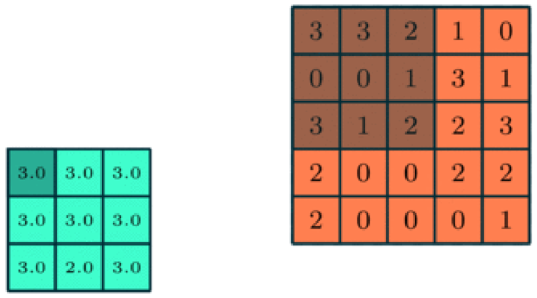
green/red color opponency observed in retinal neurons and human visual perception

blue/yellow color opponency observed in retinal neurons and human visual perception

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks
Pooling Layers

- Pooling layers are added inside a CNN architecture primarily for downsampling, aiming to reduce the computational cost. Secondarily helps on translation invariance.
  - The pooling window is moved over an activation map $\boldsymbol{A}_{ij}^{(l)}(o)$ along i, j with stride s.
  - Typical pool window sizes 2×2, 3×3.
  - Downsampling usually with $s = 2$.
  - Pools could overlap, e.g., $[3 \times 3 \ / \ 2]$



- Ad-hoc decision to use pooling or not.
- No formal justification for the effect of overlapping on pooling regions.

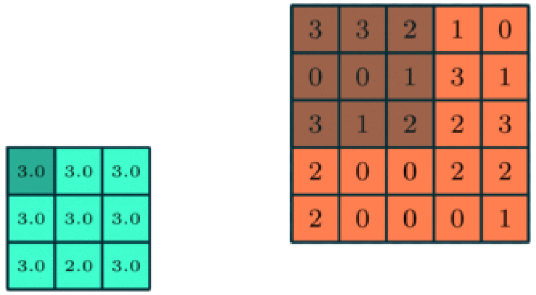
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Max Pooling Layer

- Max pooling keeps the strongest activation in a $n \times m$ region of an activation map.
  - Edges between high and low activations could be lost.
  - Downsampling is preferred to be done in max pooling layers and not in convolutional layers.

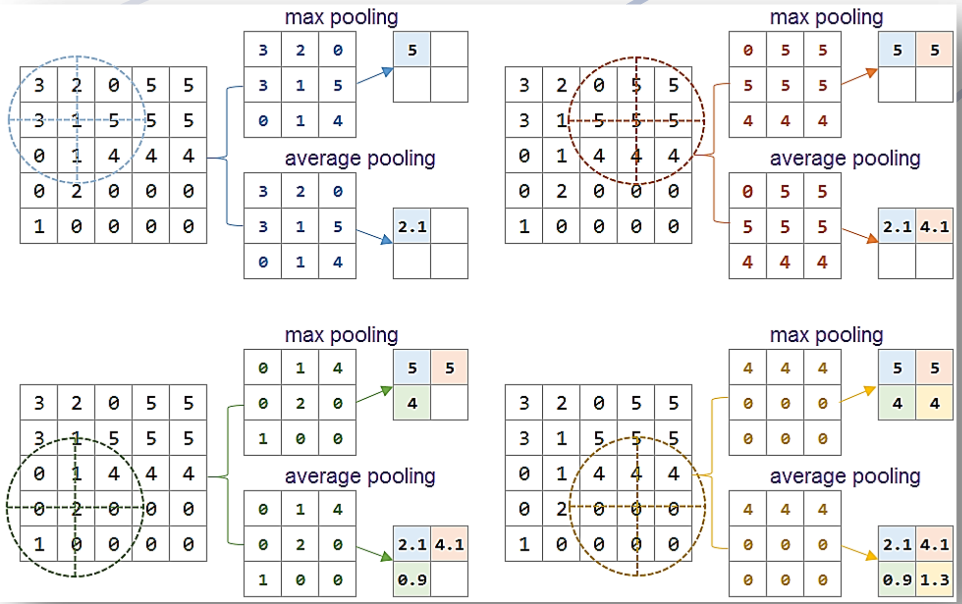
- No formal justification for the benefits of keeping the strongest activation.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks
Max Pooling vs Average Pooling

- Local max pooling is preferred over local *average pooling*.
  - Average pooling is a smoothing operation, i.e. a low-pass filter.
  - Average pooling represents better the regions that have small total activation.



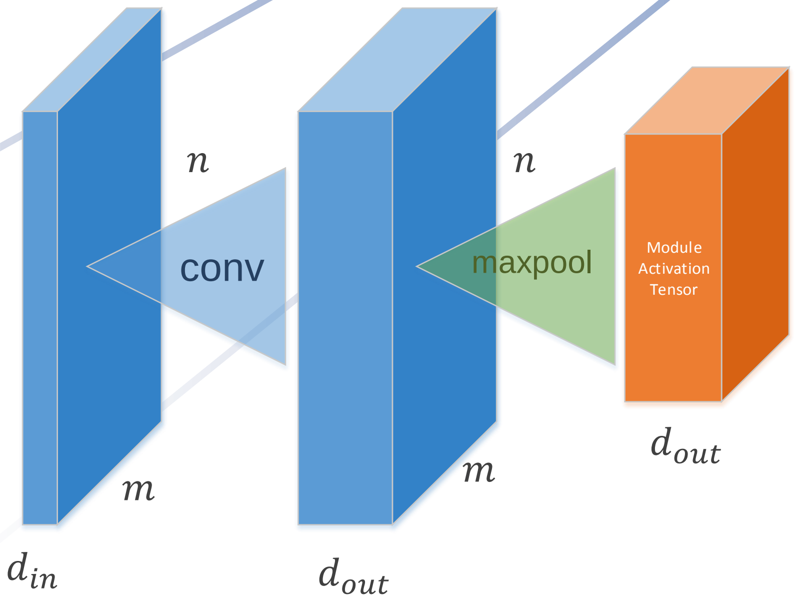
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Simple Convolutional Module

- The simplest convolutional module has one convolutional layer with stride $s = 1$ and one max pooling layer with stride $s_{pool} = 2$ for downsampling.

  - Input has $n \times m$ spatial regions and d_{in} features.

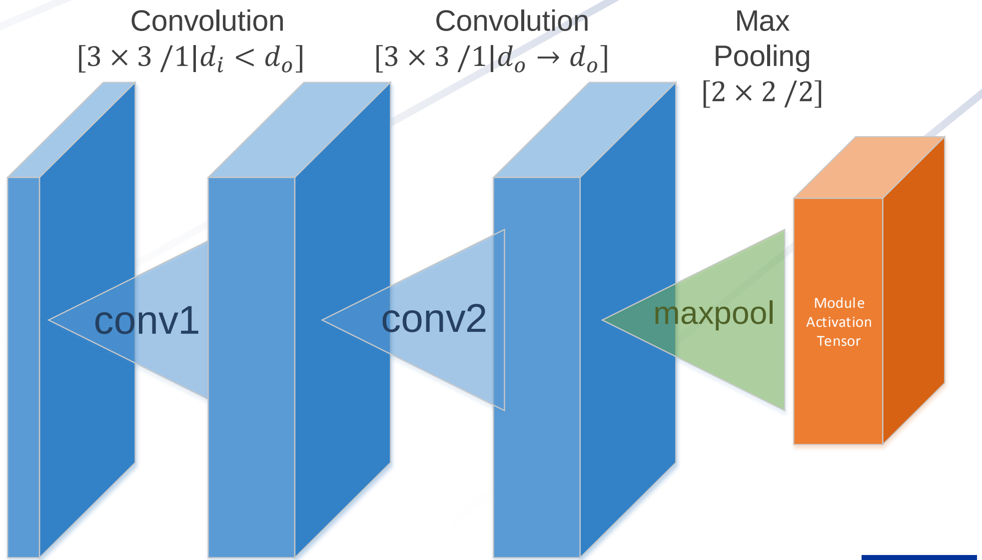


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

VGG Convolutional Module

- A stack of two 3×3 convolutional layers has an effective spatial size of 5×5.
- Uses consecutive $[3 \times 3/1]$ convolutional layers with the same d_{out} followed by 2×2 non-overlapping max pooling for downsampling.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Classification and Regression

- **Classification:** If we have a class label set $\boldsymbol{C}=\{c_1,\ldots,c_L\}$, train a NN model to assign a class label vector $\hat{\boldsymbol{y}} \in [0,1]^L$ to an object $\boldsymbol{x}$: $\hat{\boldsymbol{y}} = f_{NN}(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the CNN trainable parameter vector.
  - Essentially, we assign (predict) probabilities $P(\hat{\boldsymbol{y}} \mid \boldsymbol{x})$ that an object $\boldsymbol{x}$ belongs to each of the L classes.
  - **Training:** Given $N_{training}$ ground truth pairs $\{\boldsymbol{x}_\mathrm{i}, \boldsymbol{y}_\mathrm{i}\}$, $i = 1, \ldots, N_{training}$, estimate $\boldsymbol{\theta}$ by minimizing an error function $\min_{\boldsymbol{\theta}} J(\boldsymbol{y}\text{-}\,\hat{\boldsymbol{y}})$.
  - **Testing:** Given N_{test} ground truth validation pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}, i = 1, \ldots, N_{test}$ calculate (*predict*) $\hat{\boldsymbol{y}}_I$, $i = 1, \ldots, N_{test}$ and calculate *a performance metric*.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Classification and Regression

- **Classification:**
  - Two class (*L*=2) and multiple class (*L*>2) classification.
  - **Example:** *Face detection* (two classes), *face recognition* (many classes).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Classification and Regression

- **Regression:** If we have a function $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, train a NN model to predict **real-valued quantities** (vector $\boldsymbol{y}$ entries), $\hat{\boldsymbol{y}} = f_{NN}(\boldsymbol{x}, \boldsymbol{\theta})$, so that an error function $\min_{\boldsymbol{\theta}} J(\boldsymbol{y} - \hat{\boldsymbol{y}})$ is minimized.
  - **Training:** Given $N_{training}$ ground truth pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}, i = 1, \ldots, N_{training}$, estimate $\boldsymbol{\theta}$ by minimizing an error function $\min_{\boldsymbol{\theta}} J(\boldsymbol{y} - \hat{\boldsymbol{y}})$.
  - **Testing:** Given N_{test} ground truth validation pairs $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}, i = 1, \ldots, N_{test}$ calculate (*predict*) $\hat{\boldsymbol{y}}_i, i = 1, \ldots, N_{test}$ and calculate an error function $J(\boldsymbol{y} - \hat{\boldsymbol{y}})$, e.g. MSE.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Classification and Regression

- **Regression:**
  - **Example:** In object detection, regress object ROI parameters (width *W*, height *H*, offsets *X*, *Y*).
  - **Function approximation:** it is essentially regression, when the function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is known.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Gradient descent on a large number of images

- CNNs are trained with the same gradient descent methods as multilayer perceptron.
  - Convolution is a differentiable operation.
  - Mini-batch Stochastic Gradient Descent methods are used for image batches.
- Optimization methods:
  - *Learning rate decay* are scheduled changes to the learning rate at the various training epochs. For non-adaptive mini-batch SGD methods, e.g. *Momentum*.
  - *ADAM* is an optimization method with an adaptive learning rate.
- Large scale datasets are needed to adequately train a CNN.
  - CNNs are prone to over-fitting.
  - Training images count in the magnitude of 10 or 100 thousands.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

The ILSVRC2012 dataset

- State-of-the-art CNNs are trained on this dataset
  - The de-facto benchmark to evaluate the state-of-the-art in image classification.
- Ground Truth Set
  - Training Set: 1.28 million images that are labeled with 1000 classes
  - Validation Set: 50.000 images that are labeled with 1000 classes. Used to check the training quality at the end of a training epoch.
- Unknown Test Set:
  - 100.000 images that are used for testing a trained network.
  - Labels remain unknown until the end of the competition.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Object detection: Definitions

Object Detection

- Object detection consists of:
- Object classification:
  - Find the object class out of *C* classes.
- Object localization:
  - Find object ROI (bounding box) *A* parameters $[H_A, W_A, X_A, Y_A]$ through (CNN) regression.
- Ground truth used in training: Object class labels C_i and Bounding boxes (ROIs) B_i, $i=1,\ldots,N_{training}$.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Performance metrics

Classification: Top-5 Error

- Given the ground truth object class label C_i and top 5 predicted class labels $c_{i1}, \dots, c_{i5}$ the prediction is correct when there is $c_{ij} = C_i$. The error of a single prediction is:

$$e_{CLS}(c_{ij}, C_i) = \begin{cases} 1, & c_{ij} \neq C_i, \quad j = 1, \dots, 5 \\ 0, & otherwise. \end{cases}$$

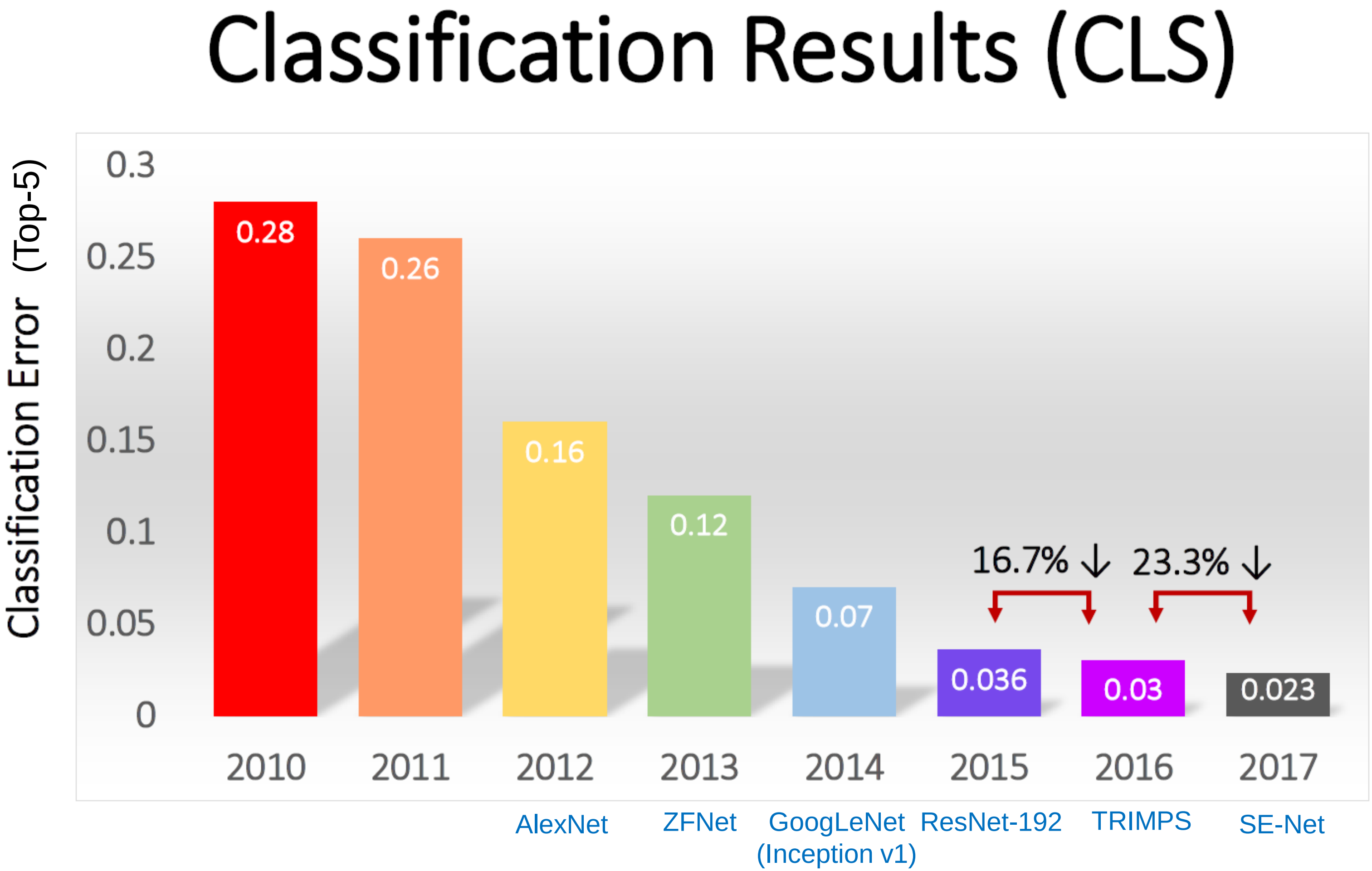
- The top-5 error is the fraction of N_{test} test images on which the prediction is wrong:

$$top5error_{CLS} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \min_j \{e_{CLS}(c_{ij}, C_i)\}, j = 1, \dots, 5.$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Advances

Image Classification



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Performance metrics

Single Object Localization: Top-5 Localization Error

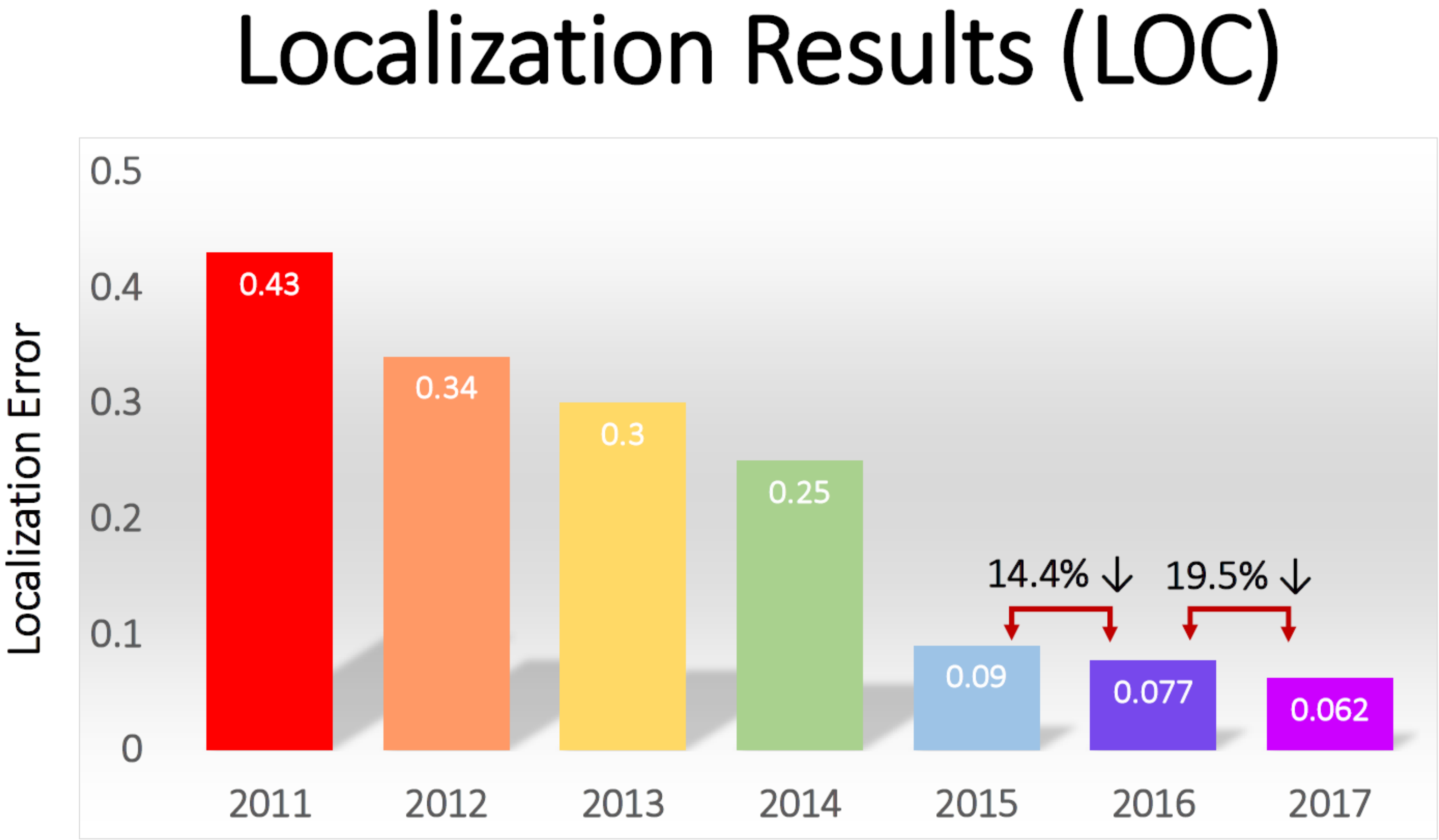
- Let us have a pair of ground truth a) label C_i and b) bounding box B_{ik}, a set of classification/localization predictions $\{(c_{ij}, A_{ij})\}_{j=1}^{5}$ of class labels c_{ij} with corresponding bounding boxes A_{ij}.
- Localization error $e_{LOC}(A_{ij}, B_{ik}) = \begin{cases} 1, & J(A_{ij}, B_{ik}) \leq 0.5 \\ 0, & J(A_{ij}, B_{ik}) > 0.5 \end{cases}$, where *intersection over union* or *Jaccard similarity coefficient* $J(\cdot)$ is defined for sets $\hat{Y}, Y$ as $J(\hat{Y}, Y) = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cup Y|}$.

$$top5error_{LOC} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \min_j \{e_{LOC}(A_{ij}, B_{ik})\}, j = 1, \dots, 5.$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Advances

Object Localization



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Performance metrics

Object Detection

- Object detection consists of:
- Object classification
  - Performance measured by e.g., top5error
- Object localization
  - find object ROI (bounding box) parameters [*H*,*W*, *X*,*Y*] through (CNN) regression.
  - Performance measured by the *Jaccard similarity coefficient* or *Intersection-over-Union (IoU)* ratio.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Performance metrics

Object Detection: Mean Average Precision (mAP)

- Object detection predicts pairs of bounding boxes A_{ij} and confidence score s_{ij} for each detection j on image $i = 1, \dots, Ntest$. If A_{ij} is matched to the ground truth box B_{ik} according to $J(A_{ij}, B_{ik}) > T(B_{ik})$ then $z_{ij} = 1$. The matching threshold depends on dimensions $H \times W$ of the ground truth box $T(B) = min\left(0.5, \frac{HW}{(W+1)(H+1)}\right)$.

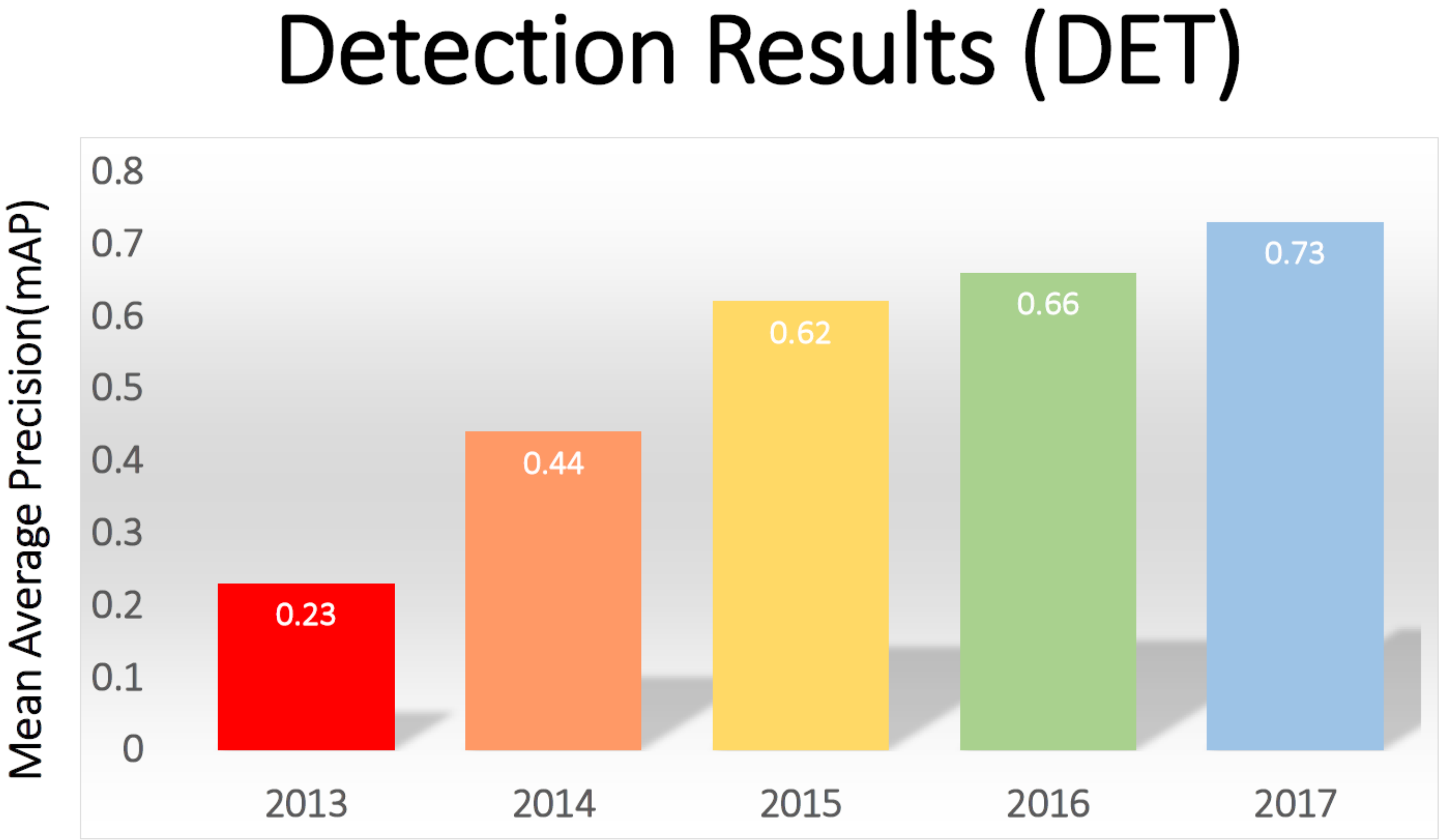
- For a confidence threshold t: $Recall(t) = \frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{C}$ and *Precision*$(t) =$ $\frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{\sum_{ij} 1[s_{ij} \geq t]}$,
  C *total number of* ground truth objects instances in the dataset.

- *Mean Average Precision* (mAP) is calculated for M levels of recall achieved by varying the confidence threshold t: $mAP_{DET} = \frac{1}{M} \sum_{i=1}^{M}$ *Precision*(t_i).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Advances

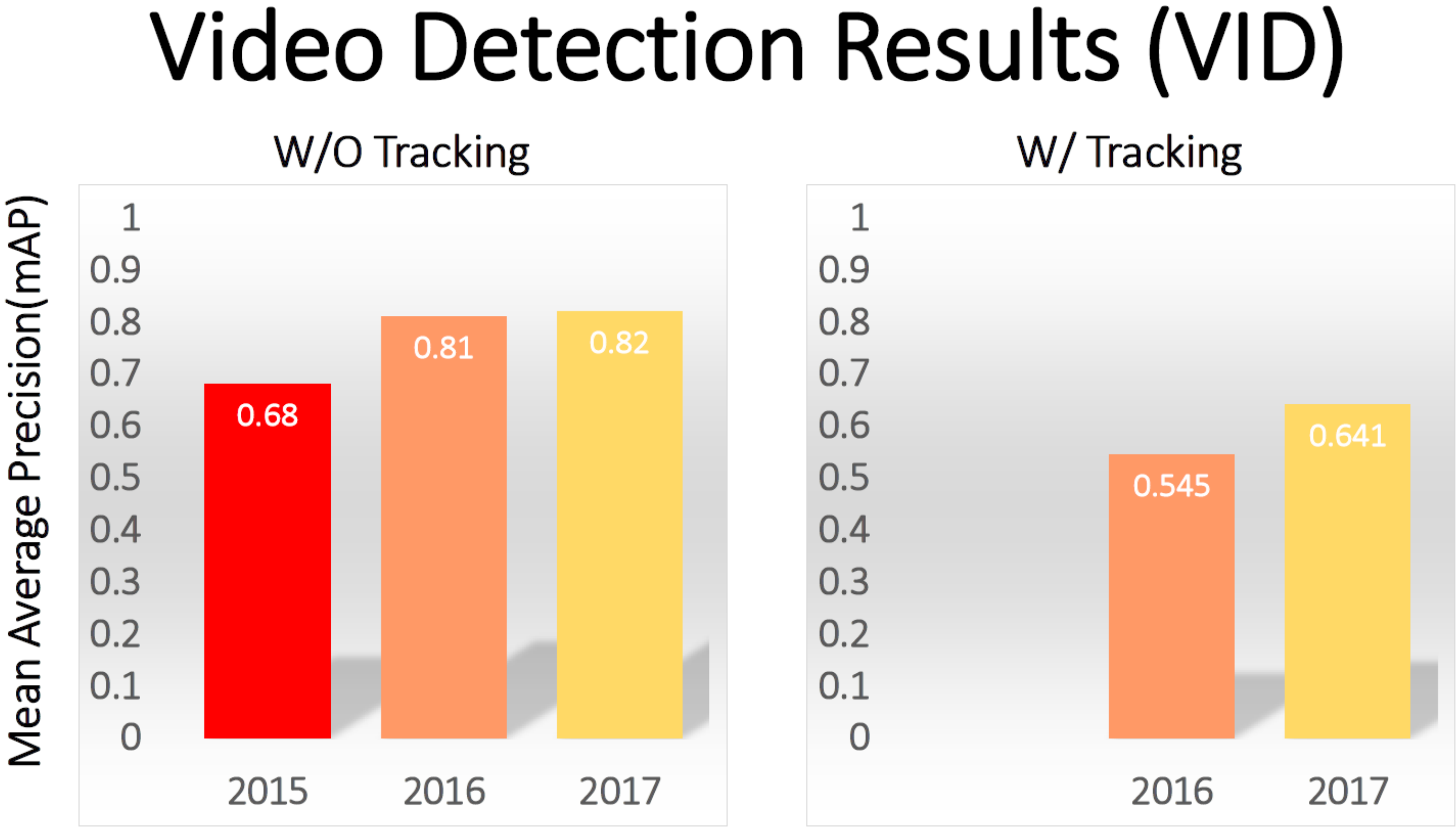
Object Detection



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Advances

Object Detection from Video



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs???

Data Preprocessing

- Images are preprocessed before the training process.
  - Mean centering by subtraction of mean R,G,B values from n images with $W \times H = m$ pixels: $\boldsymbol{X} = \boldsymbol{X}_R, \boldsymbol{X}_G, \boldsymbol{X}_B$: $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n} \times \boldsymbol{m}}$

$$\boldsymbol{X}_{MEAN\_CENTERED} = \boldsymbol{X} - \overline{\boldsymbol{X}}$$

  - Apply *a whitening* filter $\boldsymbol{W}$ using the eigendecomposition $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^T$, where $\boldsymbol{D}$ has the eigenvalues on the diagonal:

$$\boldsymbol{X}_{WHITENED} = \boldsymbol{W}\,\boldsymbol{X}_{MEAN\_CENTERED}$$

  - PCA Whitening: $\boldsymbol{W} = \boldsymbol{W}_{PCA} = \boldsymbol{D}^{-1/2}\boldsymbol{P}^T$
  - ZCA Whitening: $\boldsymbol{W} = \boldsymbol{W}_{\boldsymbol{ZCA}} = \boldsymbol{P}\boldsymbol{D}^{-1/2}\boldsymbol{P}^T$. ZCA is preferred for natural images.
- Preprocessing helps illumination invariance and convergence on certain visual features distribution, e.g. natural images of ILSVRC2012.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Weight initialization

- Completely random initialization of weights has experimentally proven ineffective to train deep CNNs.
- Values are chosen from a uniform distribution and restricted in an interval that relates to the layer input/output feature depths

$$\boxed{\left[-\sqrt{\frac{3\ \cdot a\ \cdot \beta}{d_{in} + (a-1)\cdot d_{out}}}, \sqrt{\frac{3\ \cdot a\ \cdot \beta}{d_{in} + (a-1)\cdot d_{out}}}\right]}$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Xavier initialization (2010)

$$\left[-\sqrt{\frac{3 \cdot a \cdot \beta}{d_{in} + (a-1) \cdot d_{out}}}, \sqrt{\frac{3 \cdot a \cdot \beta}{d_{in} + (a-1) \cdot d_{out}}}\right]$$

- Two intervals for the respective initialization methods:
  - Glorot initialization uses $\beta = 1, a = 2$, $\sqrt{\frac{6}{d_{in} + d_{out}}}$.
  - Xavier initialization is simplified as $\frac{2}{d_{in} + d_{out}}$.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Data Augmentation

- It is used to avoid overfitting.
- The training image set is augmented during training with label-preserving transformation of the samples:
  - Image translations and random image crops.
  - Photometric distortions, i.e. altering the intensities of RGB channels.
  - Scaling and rotation, e.g. at $\leq 90^o$
  - Vertical reflections, e.g. mirror.
  - Addition of Salt and Pepper noise.
- Data augmentation can be done with minimal computation cost inside the training process.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Normalization Layers

- Level the intermediate activations to a desired range.

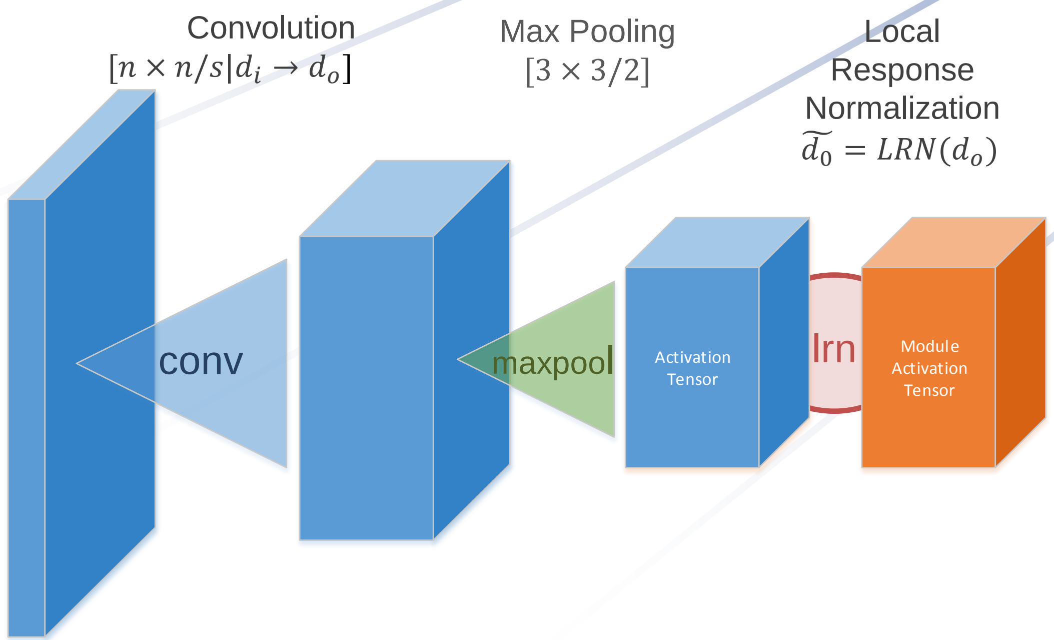
- Prevent the *exploding gradients problem,* during gradient descent training of CNNs, when using activation functions that have no upper bound.
  - **LCN** - Local Contrast Normalization is both a subtractive and divisive normalization (2009).
  - **LRN** - Local Response Normalization resembles LCN (2012).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Modules

AlexNet / ZFNet Convolutional Module

- It contains one convolutional layer, followed by a max pooling layer, followed by a local response normalization layer.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

In-layer normalization

- When using mini-batches, the probability distribution of the input for a CNN layer constantly changes during training. This causes a shift in the probability distribution of learned weights, and thus, of the CNN activations.
- The root cause is the change in the training data distribution between training mini-batches. This has been studied as the *internal covariate shift problem.*
- In-layer batch normalization fixes the distribution of the input samples.
  - BN - Batch normalization (2015)
  - BRN - Batch renormalization (2017)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Batch Normalization

- Batch normalization introduces extra parameters γ and β that scale and shift the normalized values for the mini-batches of images. These are learned together with network weights via gradient descent.

Batch Normalization

$$BN_{\gamma,\beta,\varepsilon}(x_i) = \gamma \frac{(x_i - \bar{x})}{\sqrt{s^2 + \varepsilon}} + \beta:\ i = 1,..,\mathrm{C}$$

N training samples and C minibatch size

$k = \frac{C}{N},\ \ k \in \mathbb{N}$ count of mini-batches

Mean and Sample Standard Deviation

$$\bar{x} = \frac{1}{C}\sum_{n=1}^{C} x_i$$

$$s^2 = \frac{1}{C}\sum_{n=1}^{C} (x_i - \bar{x})^2$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

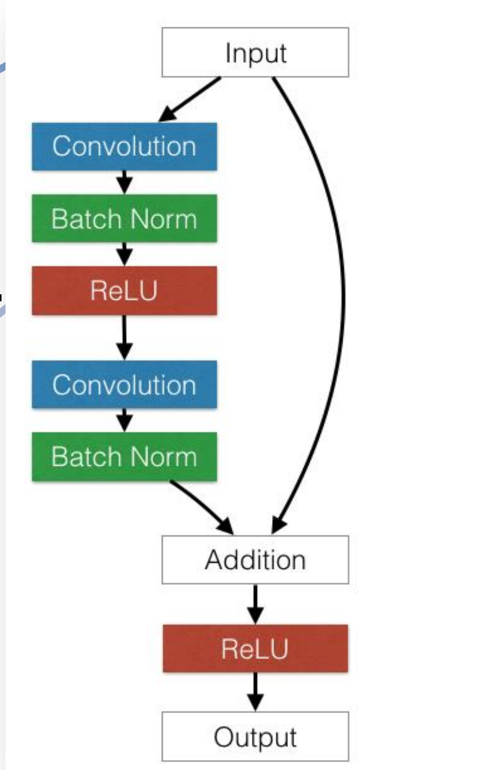
Advanced CNN modules

Residual Convolutional Module

- The basic module of *ResNets.*
- A *shortcut connection* bypasses layers. BN is used before the activation function. These implement *identity mapping*.

$$\mathbf{A}^{(l+2)} = f_{l+2}(\mathbf{A}^{(l)} + BN_{l+2}\big(\mathbf{b}^{(l+2)} + \mathbf{W}^{(l+2)} * f_{l+1}(BN_{l+1}(\mathbf{b}^{(l+1)} + \mathbf{W}^{(l+1)} * \mathbf{A}^{(l)}))\big)$$

- *Residual learning* makes possible to train extremely deep CNNs up to 192 layers.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Regularization

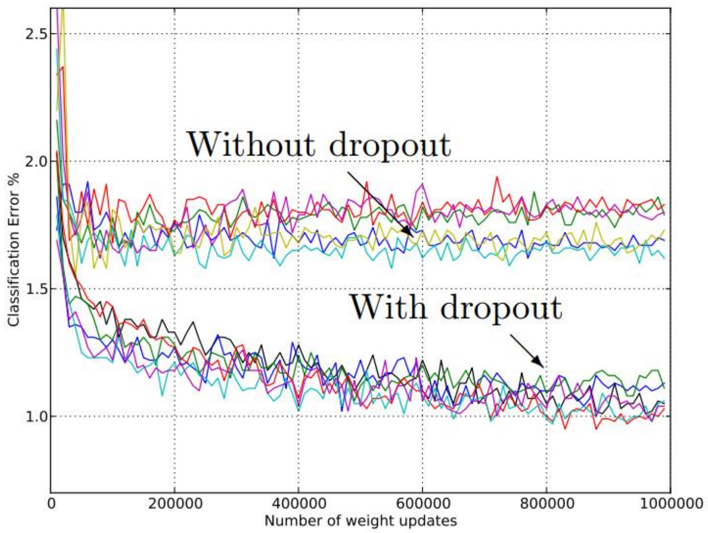
- Regularizers aim to reduce over-fitting during training of deep CNNs on image data:
  - *Weight decay* is a well-known approach that adds norms of the networks weights, like the L_2 (Euclidean norm), as terms of the loss function.
  - *Dropout* randomly excludes a set of neurons from a certain training epoch with a constant keep out probability p_{keep}.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Training CNNs

Dropout

- Activations of dropped out neurons are set to zero.
  - They do not participate in the loss, thus excluded from back-propagation.
  - Dropout was initially used in AlexNet after each fully connected layer.
  - During testing a trained model, all neurons are used with their already learned weights.
- Induces dynamic sparsity during training.
- Prevents complex co-adaptations of the synaptic weights, that may lead to correlated activations of neurons.



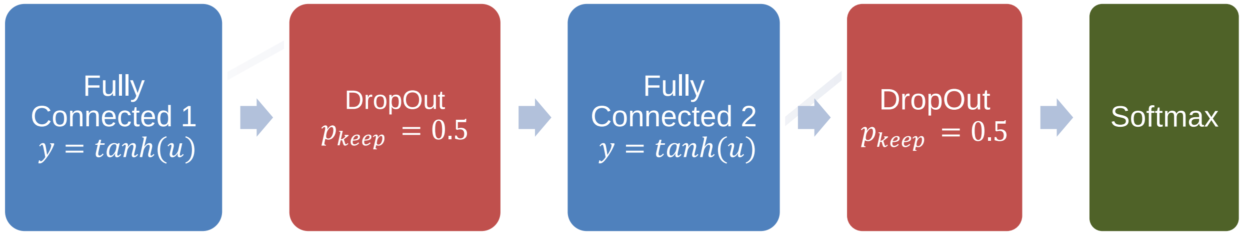
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Classifiers

AlexNet / ZFNet Fully Connected Classifier

- Dropout of 50% of neurons is used during classifier training.

- There are 2 fully connected layers with $d_{out} = 4096$ neurons and a Softmax output layer with N neurons for N classes.

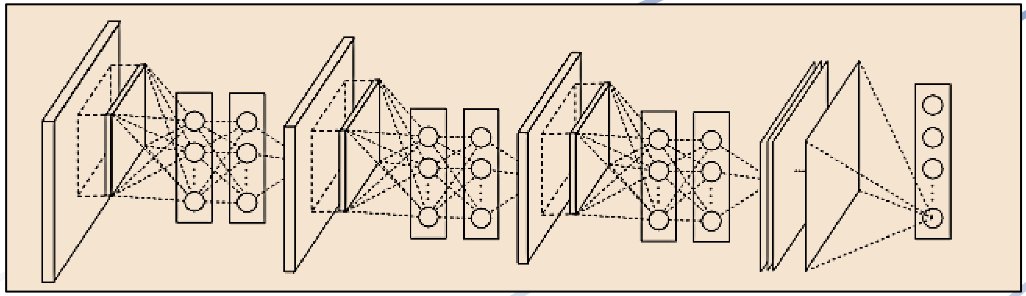


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN methods

Network-In-Network (NiN)

- A micro neural network is placed between two neural network layers.



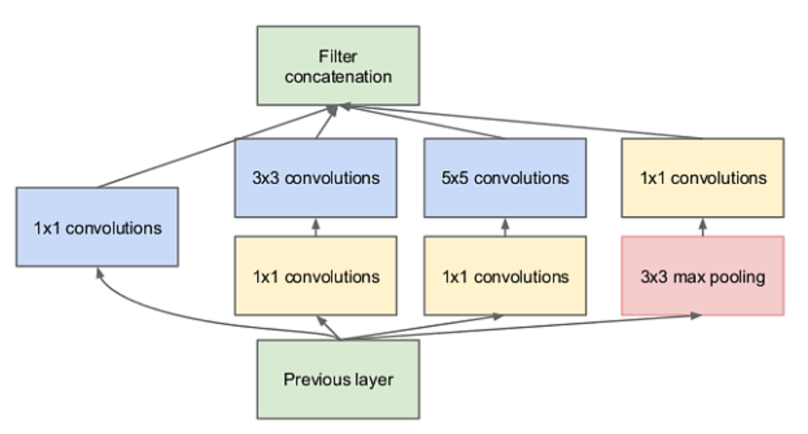
- This idea is extensively used by the state-of-the-art CNNs as a 1×1 convolutional layer $(h_1 = 1, h_2 = 1)$.
- Output features are weighted averaged (plus activation functions) of the input features.
- Operating at a single region of input, increases the representation power its feature descriptor.
  - Feature dimensionality can increase or decrease.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Advanced CNN modules

Inception Convolutional Module

- Basic idea is to have multiple window sizes for convolutions that operate over the same input region.
  - Building block of GoogleLeNet (Inception v1) CNN.
  - Latest upgrade to the model is Inception v4.



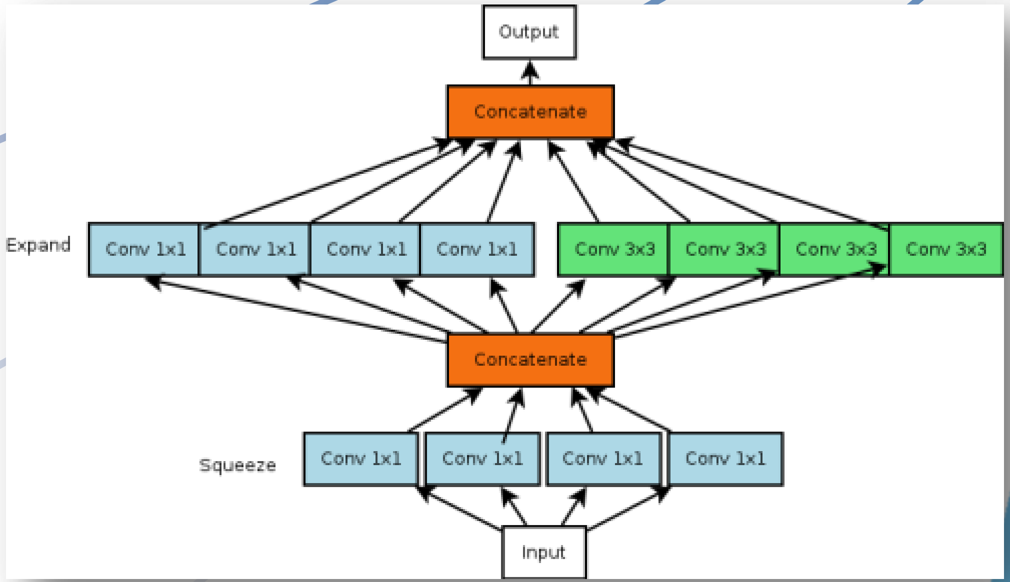
- The activation volumes are concatenated in the module output along the feature dimension.
  - $d_{out} = d_{1\times1} + d_{3\times3} + d_{5\times5} + d_{maxpool}$
- The 1×1 convolutional layers are used before the computationally intensive 3×3 and 5×5 convolution operations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Lightweight CNN Modules

Fire Module

- Dimensionality of features is reduced by parallel 1×1 layers.
  - "Squeezed" activations are concatenated.
- Dimensionality of features is increased by multiple and 3×3 and 1×1 layers.
  - "Expanded" activations are concatenated.



- Building block of SqueezeNet that achieves AlexNet classification accuracy with $50\times$ fewer parameters. This research area is *model compression.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN methods

Global Average Pooling

- In the NiN paper, a *Global Average Pooling* layer is proposed, which averages the values of each activation map.

Global Average Pooling Operation

$$y^{(l)}(o) = \frac{1}{n_{out}^{(l-1)} \times m_{out}^{(l-1)}} \sum_{i=1}^{n_{out}^{(l-1)}} \sum_{j=1}^{m_{out}^{(l-1)}} y_{ij}^{(l-1)}(o)$$

on $n_{out} \times m_{out}$ activation map for feature o of previous layer $l-1$

- It is used in a CNN classifier before the softmax layer.
- It links feature activations with class predictions.
  - Feature activation maps become confidence maps for the predicted classes.
  - Global Average Pooling can replace computationally expensive fully connected layers.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Classifiers

Inception Output Classifier

- It contains a global average pooling, 40% dropout and a fully connected (FC) layer of $d_{out} = 1024$ neurons with a linear activation function before the Softmax layer.
  - It replaces the memory demanding classifier that uses two FC layers of $d_{out} = 4096$ each.

Global Average Pooling → DropOut $p_{keep} = 0.6$ → Fully Connected Linear $y = f(u)$ → Softmax

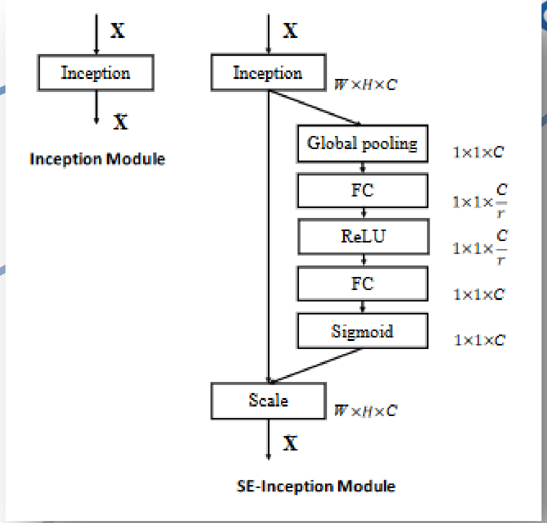
- There are two auxiliary classifiers in the Inception v1 (GoogleLeNet) CNN for two additional gradient flows.

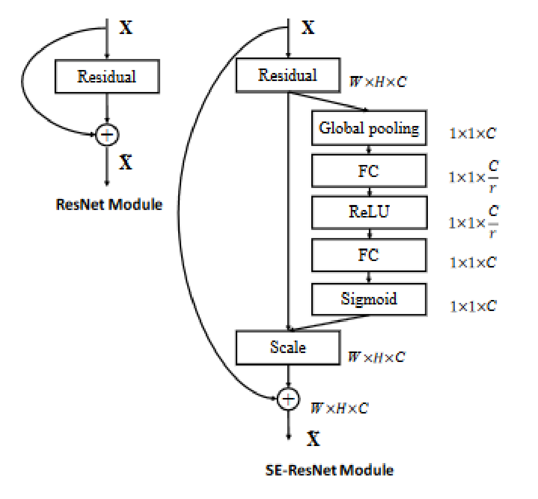
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Current state-of-the-art CNN

Squeeze and Excitation Module

- Global spatial information is “squeezed” into one feature descriptor with global average pooling.
- A bottleneck mini-network implements a gating mechanism.
  - First fully connected (FC) layer reduces dimensionality before the ReLU non-linearity.
  - Second FC layer increases dimensionality before the Sigmoid gating function. Output is $s^{(l)} \epsilon [0,1]$
- Convolutional activations are scaled $\boldsymbol{Z}^{(l)} = \boldsymbol{A}^{(l)} s^{(l)}$.





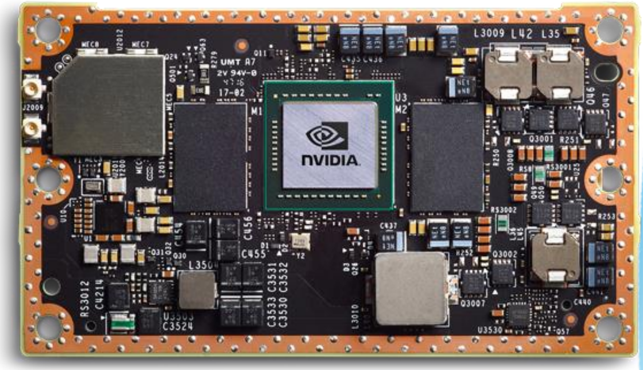
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Convolutional Neural Networks

Computational Cost

- Current research aims to reduce the number of CNN model parameters, thus reducing the complexity.
  - Less parameters means less memory and faster recall of patterns.
  - Prediction accuracy of models should be competitive or the same.

- Lightweight CNNs need to fit in the memory of mobile accelerators, like the *NVIDIA Jetson TX2* that is used in MultiDrone.
  - Less computations means less energy consumption.
  - Better models may lead to increased flight time.

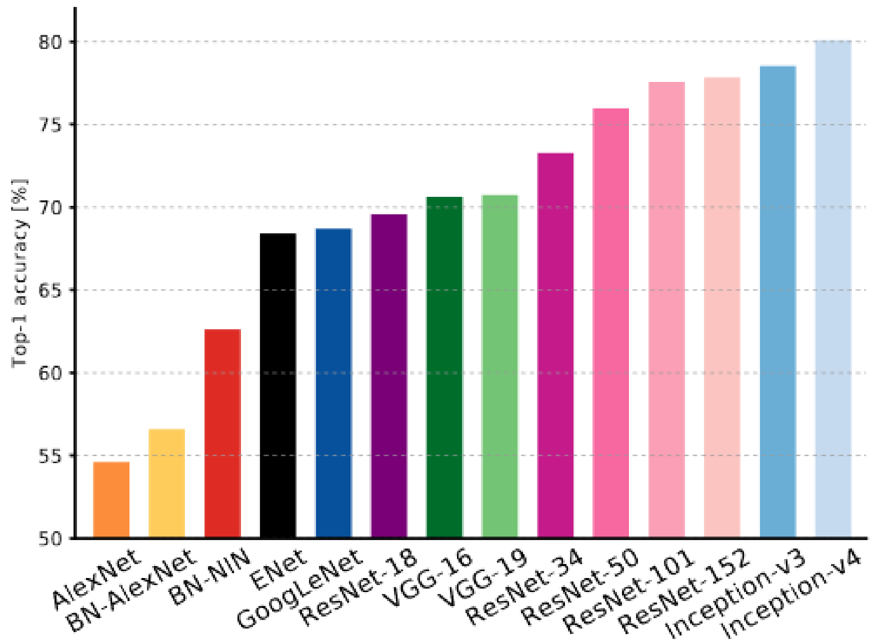


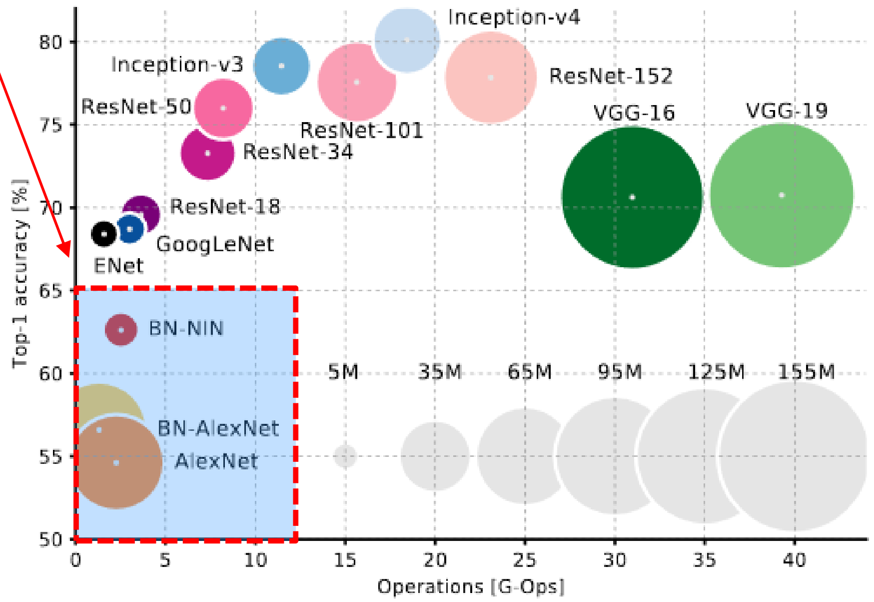
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Computational Cost

Early CNN models

Low performing models



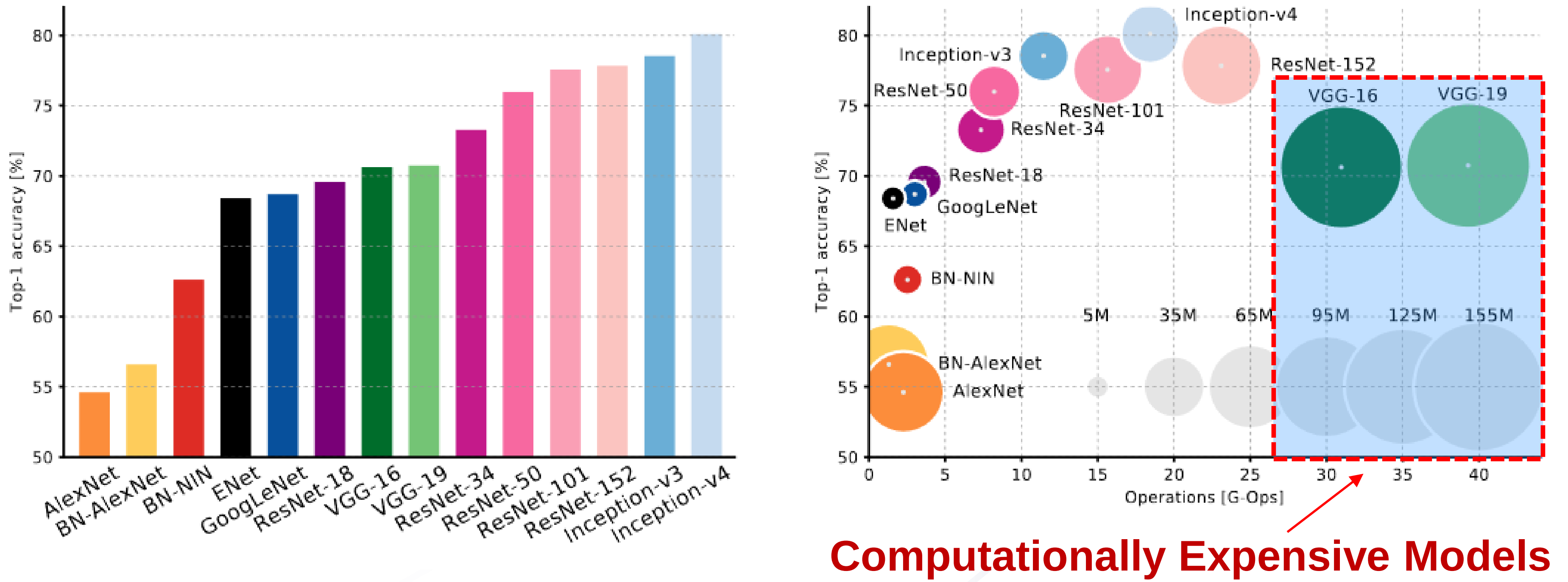


A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” *arXiv:1605.07678 [cs]*, May 2016.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Computational Cost

Heavy memory demands



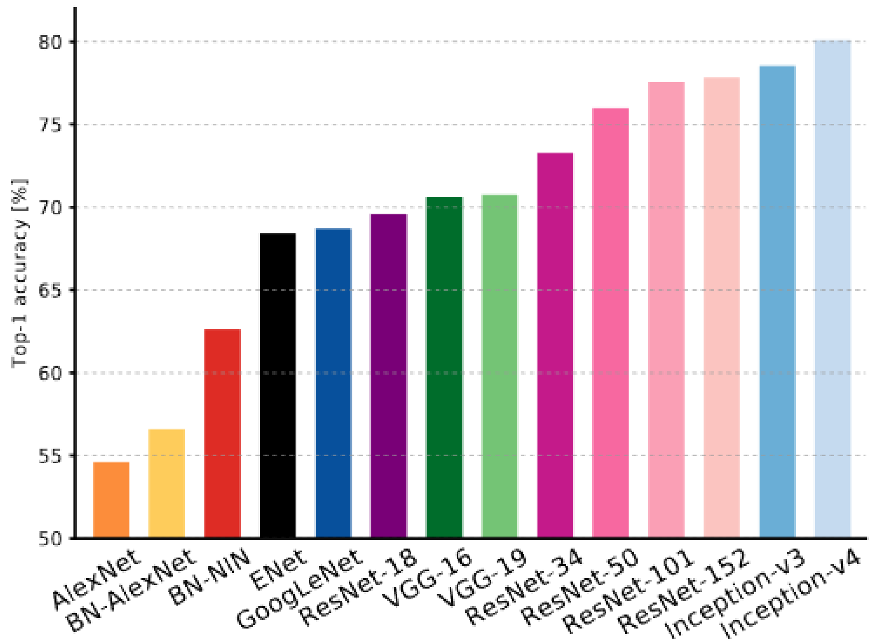
A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *arXiv:1605.07678 [cs]*, May 2016.

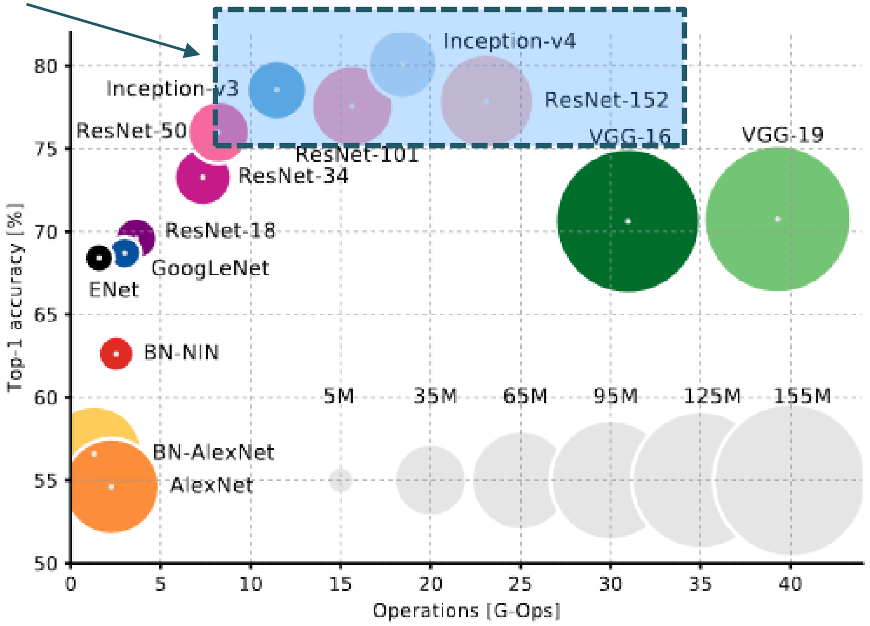
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Computational Cost

State-of-the-art

Best performance but increased complexity





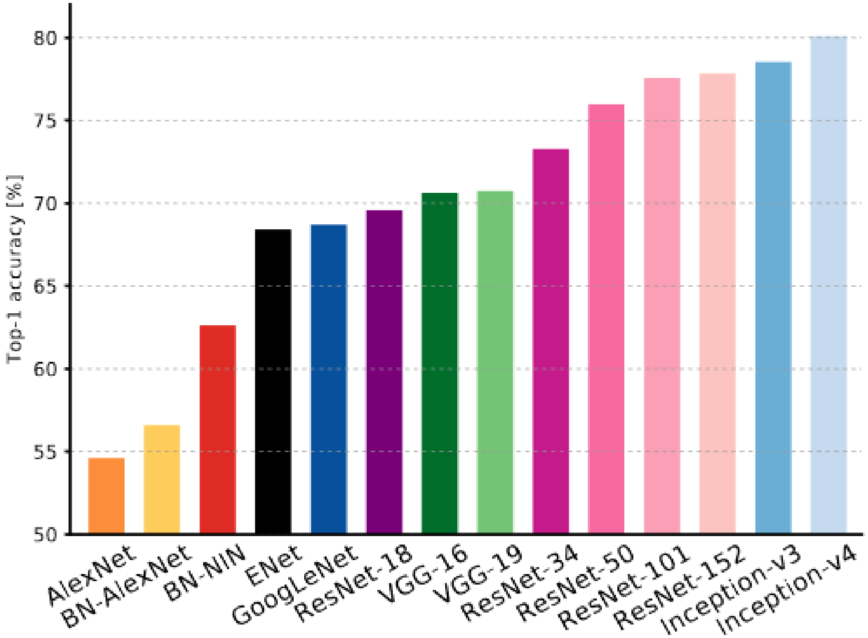
A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” *arXiv:1605.07678 [cs]*, May 2016.

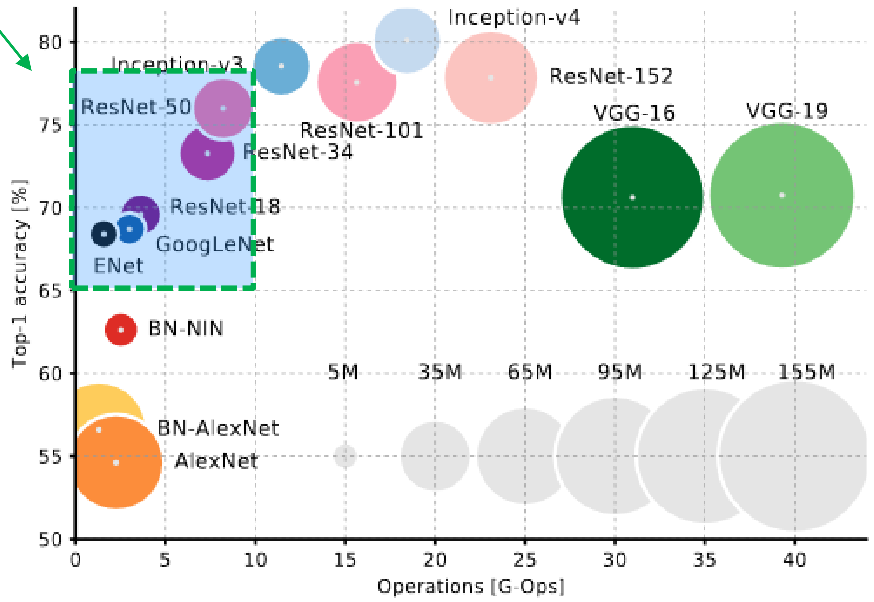
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

CNN Computational Cost

Candidates for real-world applications

"Goldilocks" zone of CNN models





A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *arXiv:1605.07678 [cs]*, May 2016.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)

Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@aiia.csd.auth.gr
www.multidrone.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE)