Lab course

Image Analysis I

ST 2023

Hubert Kanyamahanga
(kanyamahanga@ipi.uni-hannover.de)

Lab 3: Deep Learning & Pixelwise classification

Content of the 3rd Lab

Goal:

- Hands-on experience and gain a deep intuitive understanding of DL algorithms
  - By **applying** Convolutional Neural Networks (CNNs)
  - By **investigating** some approaches to **improve** CNNs performance

Tasks:

1. Implement and train CNN classifiers
   ✓ Application on existing benchmark for **image categorization**

2. Implement a Fully Convolutional Neural Network
   ✓ Application on existing benchmark for **image segmentation**

3. Discussion and evaluation (**Visually** and **Quantitively**)

Topics of Lab 3

- Deep learning
  - Architecture
  - Tasks
  - Training
- Frameworks:
  - Overview
- Datasets

Topics of Lab 3

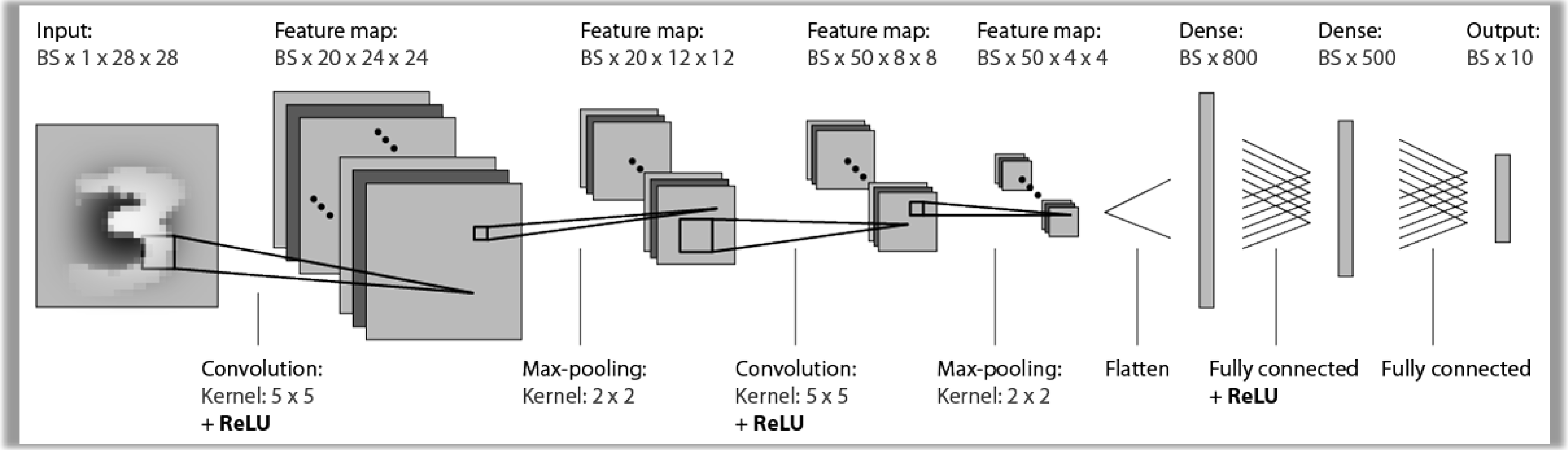
- Deep learning
  - Architecture
  - Tasks
  - Training
- Frameworks:
  - Overview
- Datasets

Deep Learning

- **Neural networks** had gone out of fashion, because:
  - Networks with **few** layers & neurons:
    → not adaptable enough
  - Networks with **many** layers & neurons:
    → numerical problems in the determination of the parameters
- **Neural networks** have come back in the context of "**Deep Learning**"
- Networks with many layers ("**deep**" networks) & many neurons
  - Deep networks come in different flavors; here:
    Convolutional Neural Networks (CNN) → work well for image-based tasks
- Parameters: **Weights** are interpreted as coefficients of linear filter matrices which can be learned

CNN Example: LeNet

- Example: Recognizing hand-written digits [LeCun et al., 1998]

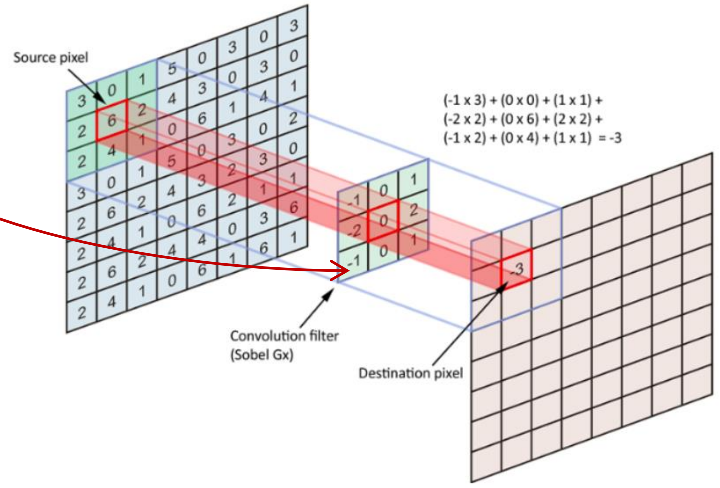


INPUT **Feature Extraction:** Low level: Edges, borders, shapes
Higher Level: Specific objects type like Numbers
Classification **OUTPUT**

- Convolutional Layers with 5×5 convolution kernels
- Pooling Layers → Subsampling by factor 4
- Fully Connected Layers → Two hidden layers

Convolution Layers

- Convolution: **Linear operation**; Matrix multiplication and addition
- Followed by a non-linear function like **ReLU**
- Goal: **Extract features** from input image
- **Sliding window** technique using a *k* x *k* matrix filter or kernel (ex: 3 x 3 filter)
- In practice, CNN learns the **values** of these filters on its own from the data (**training process**)
  - Parameters to specify for a CONV layer:
    - ***Number of filters*** *and* ***filter size*** *(spatial extent)*
    - ***Activation function*** *(ReLU)*
    - ***Stride*** *and* ***padding***



  - The results of a CONV layer is called **feature map or activation map**

Pooling Layers

- Reduce data volume (spatial size) by increasing the scale of the feature maps
- Reduce the computational complexity of the network and extract prominent features
- Combine *k* x *k* pixels by selecting one representative value
  - Take local maximum of the filter responses → **max pooling** (more frequently used)
  - Average → **average pooling** (produces smoother results than max pooling)
- Pooling increases the robustness due to local shifts and to noise

15	12	2	6
8	3	8	4
5	4	1	0
6	7	3	4

→

15	8
7	4

2 x 2 max pooling

vs

15	12	2	6
8	3	8	4
5	4	1	0
6	7	3	4

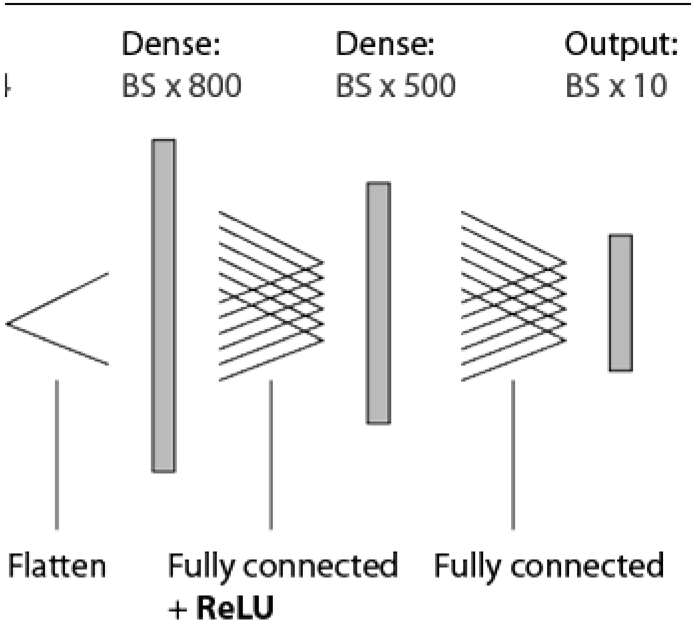
→

9.5	5
5.5	2

2 x 2 avg pooling

Fully Connected Layers

- Traditional **Multilayer Perceptron** that uses a softmax activation function in the output layer
- Every **layer neuron** on the previous layer is connected **to every neuron** on the next layer
- Acts on **a flattened output** of the last layer of Conv block
- If we have an activation volume, we must flatten into a vector first
- **It's okay to flatten here since**
  - We have already passed through all of the CONV layers
  - And applied the filters
- Fully connected layer acts as a **Classifier**
- No convolution operations are involved here!




```
# fully connected layers
f_c_1 = nn.Linear(800, 500)
f_c_2 = nn.Linear(500, 10)
```

Topics of Lab 3

- Deep learning
  - Architecture
  - Tasks
  - Training
- Frameworks:
  - Overview
- Datasets

Task 1: Image Classification - CNNs

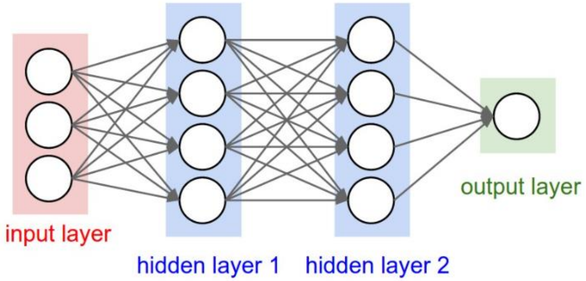
- **RECALL**: **From the 2ND Lab [Implemented a Neural Network: MLP]** to classify images
- Only one class label was predicted for each image

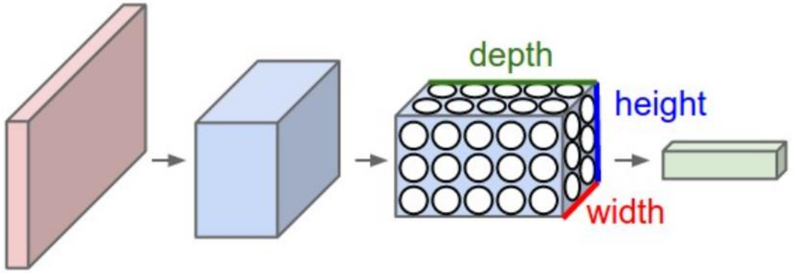
→ human face 0.98

→ frog 0.01

→ bird 0.01

- **For the 3rd Lab**: **Convolutional Neural Networks**
- Replace '**hidden layers**' of NNs with **Convolution blocks** for CNNs





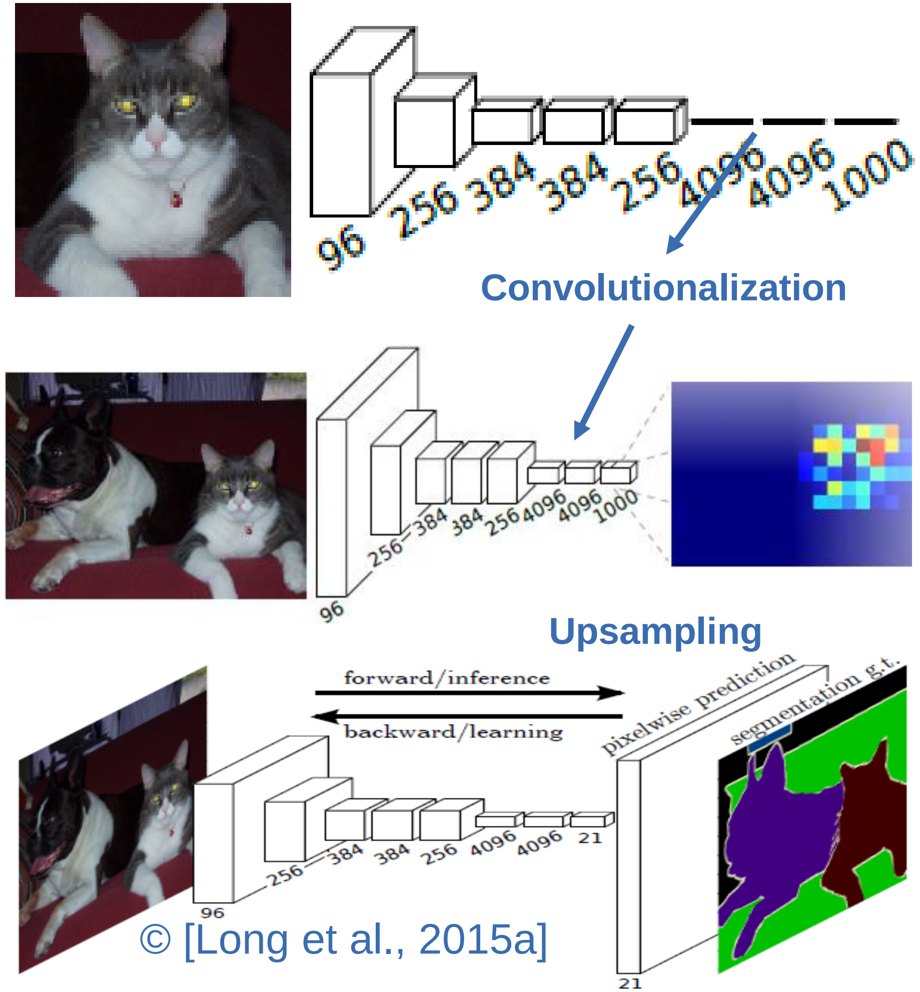
Task 2: Pixelwise Classification - FCNs

- **RECALL**: For standard CNN, the input consisted of an entire image of a given size
- Pixel-wise classification of images of arbitrary size:
  - Sliding window approach :
    - ➢Shift the input domain over the image
    - ➢Predict class of the central pixel at each position → slow
- Architecture:
  - o Fully Convolutional Networks
    - → E.g. Decoder - Encoder Networks

Fully Convolutional Network

- Better solution: **Fully convolutional networks (FCN)** with **down-sampling** and **upsampling** inside the network [Long et al., 2015a]
- Standard CNN: Predict class for one image patch
- Fully convolutional networks:
- FCNs replace the final dense in CNNs layers with Conv
  - Perform each convolution over the **entire image**
  - Result: Down-sampled score for each class
  - Needs to be upsampled to get initial size image

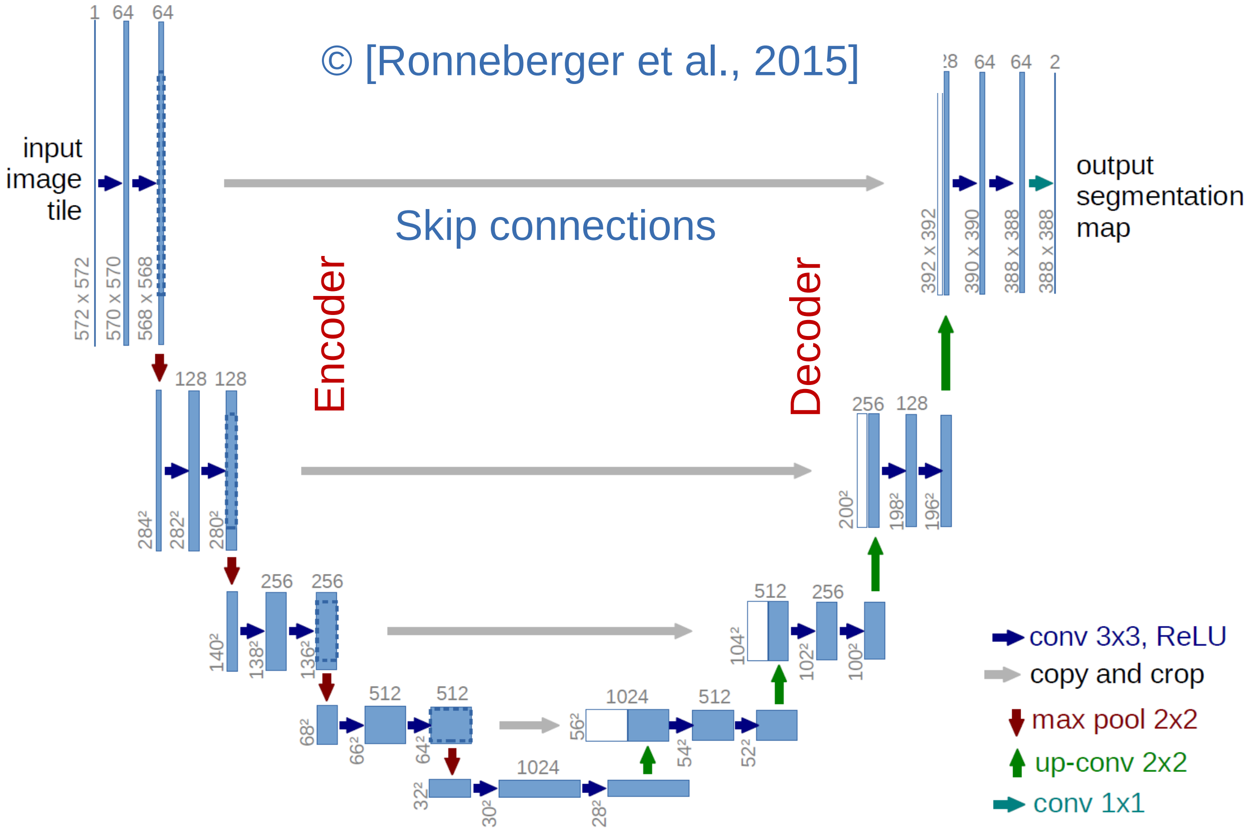
    → **Bilinear interpolation** or **Transposed convolution**



© [Long et al., 2015a]

Encoder-Decoder Architecture: UNet

- U-Net [Ronneberger et al.,2015]:
  - Encoder-decoder network for pixel-wise classification
  - Skip connections to preserve object boundaries
  - Standard configuration in many remote sensing applications



Topics of Lab 3

- Deep learning
  - Architecture
  - Tasks
  - Training
  - Summary
- Frameworks:
  - Overview
- Datasets

CNN Training: Loss Functions

- CNNs are trained like any other neural networks
- Stochastic minibatch gradient descent with or without momentum
- Adam (faster convergence and more reliable)
- Most frequently used loss: Softmax (cross-entropy) loss: use output y_{nk} of last layer (aka. logits) as argument of the softmax function:

$$E(\mathbf{w}) = \sum_n -\log\left(\frac{e^{y_{nr}(\mathbf{w},\mathbf{x}_n)}}{\sum_k e^{y_{nk}(\mathbf{w},\mathbf{x}_n)}}\right) \rightarrow \min$$

y_{nr} : output for the class label C_n of the training sample

Summary

- (From the 2nd Lab), Networks with **many** layers & neurons

  → numerical problems in the determination of the parameters → vanishing gradients

- **Standard Convolutional Neural networks (e.g LeNet)**:
  - Stack of convolution, non-linearity, pooling + one or more fully connected layer(s)
    → **Input:** image of fixed-size and the **output** is one class label for each image

- **Fully Convolutional Neural networks (FCN)**
  - **Similar** architecture to CNNs **without Fully connected layers** at the end of the network
  - **Fully Connected Network → 1x1 Convolutional + Upsampling**
    → **Input:** image arbitrary size and the **output** is the class of the central pixel position
  - Example **: Encoder-Decoder Networks (Unet)**

Topics of Lab 3

- Deep learning
  - Architecture
  - Tasks
  - Training
- Frameworks:
  - Overview
- Datasets

Frameworks for Deep Learning

- Overview of DL-Frameworks:
  - Tensorflow (Google)
  - **PyTorch** (Facebook)
  - Caffe (UC Berkeley)
  - MXNet (Apache)
  - …

[Comparison on Wikipedia]

Frameworks for Deep Learning

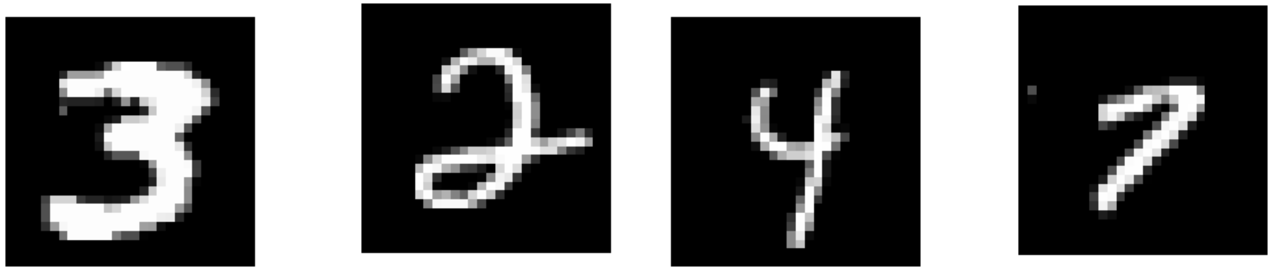
- Reasons for using DL-Frameworks:
  - Efficient and automated gradient computation
  - GPU support: e.g. device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
  - Use existing implementations of
    - Activation functions (Sigmoid, Tanh, ReLU, …)
    - Operations (Convolution, pooling, …)
    - High level models for layers / networks

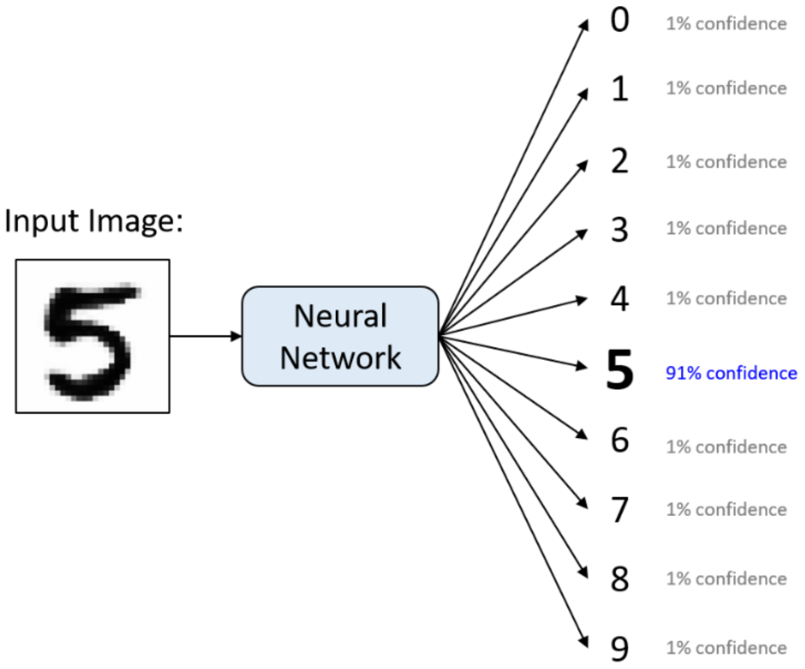
Topics of Lab 3

- Deep learning
  - Architecture
  - Tasks
  - Training
  - Summary
- Frameworks:
  - Overview
- Datasets

Dataset: Task 1 - Classification

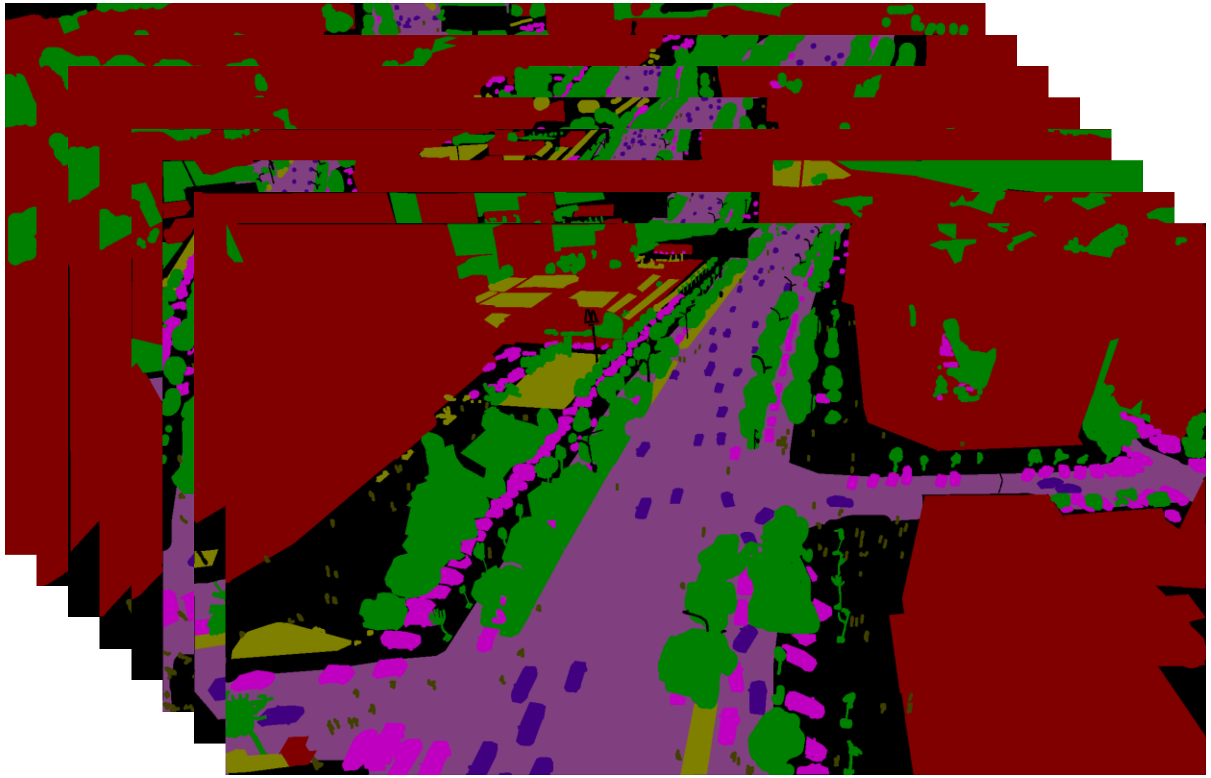
- MNIST: Large database of handwritten digits
  - The “Hello World” of deep learning
  - Classification of images of hand written digits 0-9 → OCR
  - Each image has 28x28 pixel (single channel)
  - 60000 images for training, 10000 for testing





Dataset: Task 2 - Segmentation

- Training samples: 140, Validation samples: 70, Test samples: 60



- 7 Classes: Building, Road, Tree, Low Veg, Human, Moving Car, Static Car
- **UAVid**: UAV video seq. dataset for Sementic Segmentation task focusing on Urban Scenes

Submission of Results

- Submission deadline: **24 July 2023 before 11:00 am**

- Assignment → Jupyter Notebook (only digital)
  - Run the jupyter notebook before submission
  - Includes code and discussions
  - Answer concisely but completely
  - Use meaningful variable names
  - **Consider acceptance rules** Consider IAI_23_Lab_Introduction.pdf
  - Consider IAI_23_Lab_Technical_Details.pdf
  - Read my comments on previous labs (Lab 1 and 2)
  - Contact me for ANY questions!