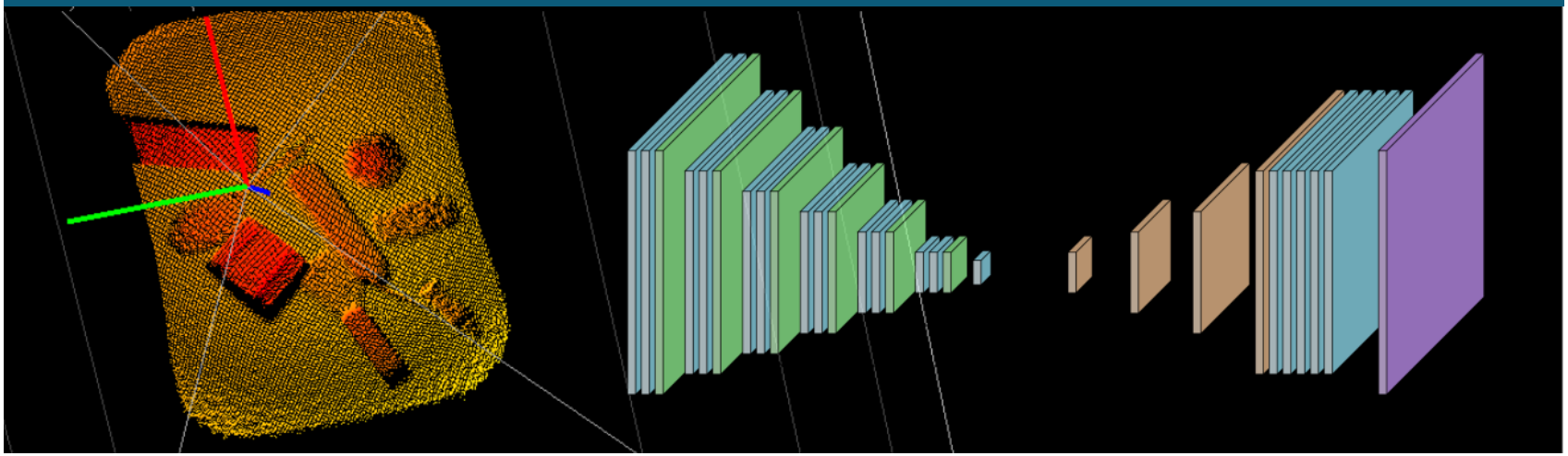
LIÈGE université Sciences Appliquées

LIÈGE université

Semantic Segmentation of Depth Images for Robotic Grasping

INFO8010 — Deep learning



COLINE Joachim

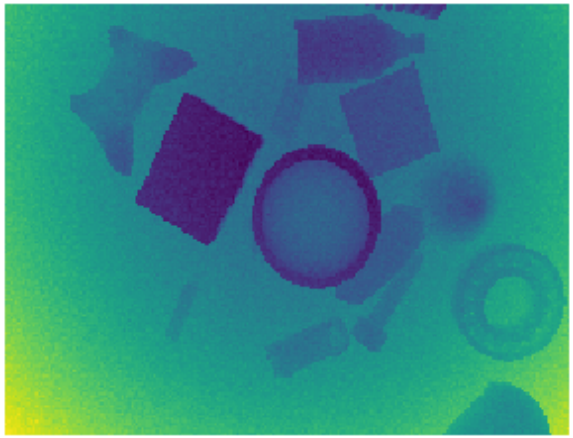
Outline

Outline

Problem addressed

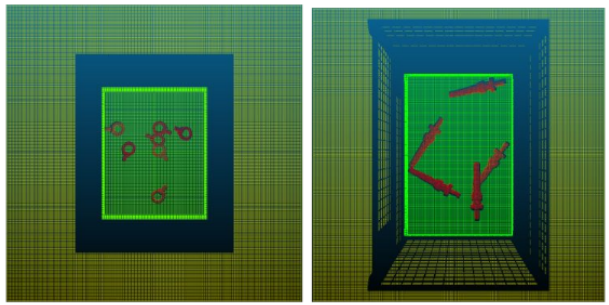
- Autonomous robotic manipulation of unknown objects is a **challenge**
- Applications include production lines, manufacturing processes, sorting tasks, home service
- INTEGRIA researchers are working on AI solutions for grasping tasks using **depth sensing**

→ This work tackles the problem of **binary segmentation** of depth data



Related work

- Similar problems have been addressed in the literature but mostly deal with RGB/RGBD images
- The work published by Khalid et al. in 2019 is very close to ours and showed promising results using a **fully convolutional** neural network



M. U. Khalid, J. Hager, W. Kraus, M. Huber, and M. Toussaint.
Deep workpiece region segmentation for bin picking.
08 2019.

Outline

Narrowing the scope

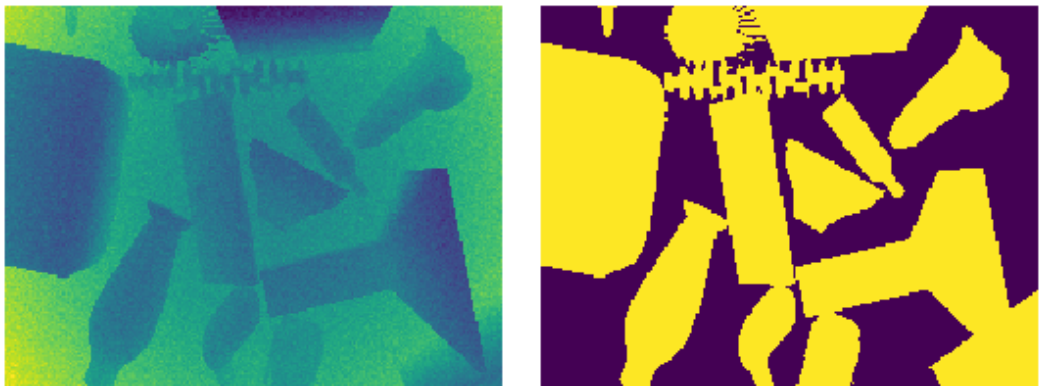
- Vertical grasp $\rightarrow$ sensor oriented downward
- Flat background
- Graspable objects (size $\sim$ 10cm)
- TOF sensor of resolution 224$\times$171

FIGURE – Pico flexx by PMD Technologies

Synthetic dataset generation

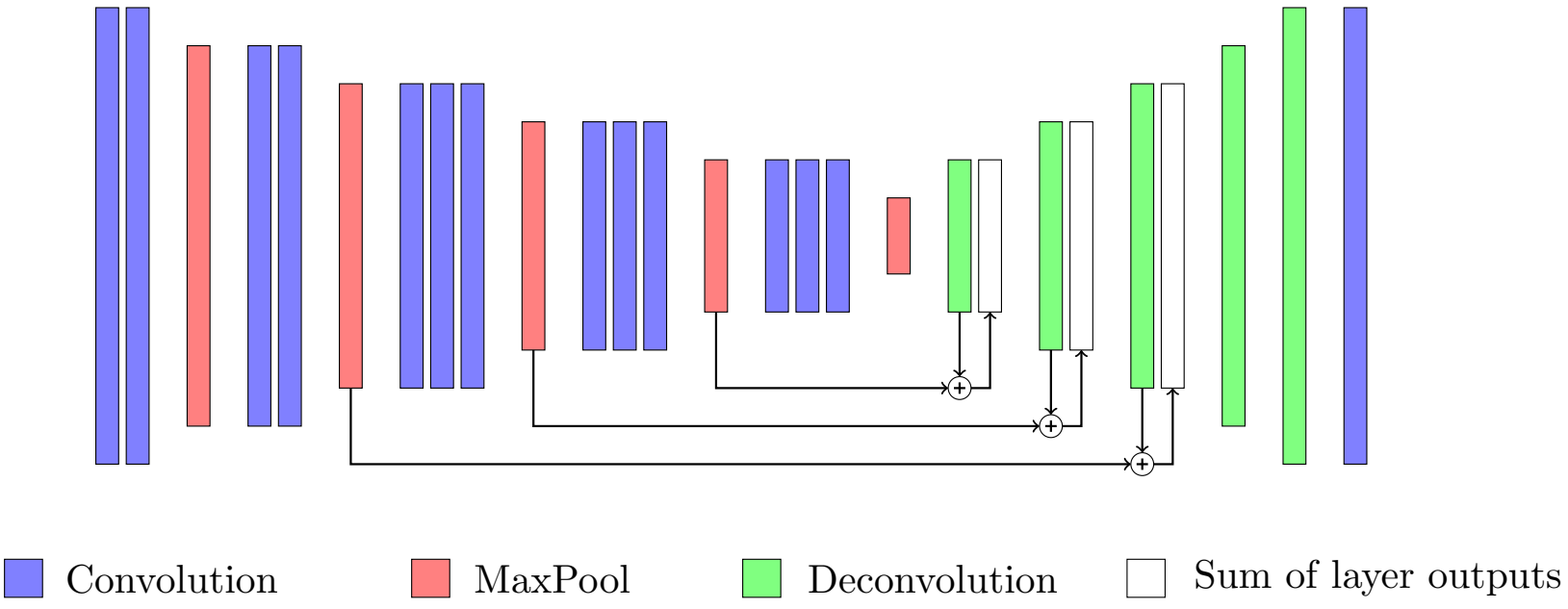
- Simulation of scenes using **Blender**
  - 10,809 models picked from existing datasets
  - Automated drop-to-ground scenarios using Blender Rigid Body Physics
  - Randomness in object position, orientation, size and sensor orientation

- Simulation of TOF sensor captures using **BlenSor**
  - With Pico flexx specifications
  - With default noise parameters $\rightarrow$ to be modified in future work



Neural network

- FCN-8s architecture
- Pretrained VGG16 encoder
- Deconvolutional decoder with
  - 3,929,025 trainable parameters
  - ReLu and batch normalization after each deconvolution
- Additional sigmoid activation function for inference



Training

- Preprocessing
  - Standardization
  - 3-channel tensor
  - 5× downscalable by factor 2
    → zero padding
- Loss function
  - BCE with logits

$$\mathcal{L}(x, y) = -\left[y \log \sigma(x) + (1 - y) \log \left(1 - \sigma(x)\right)\right] \tag{1}$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

- Optimizer
  - Adam algorithm
- Initial weights
  - Kaiming initialization
- Data splitting
  - 70% training, 20% validation, 10% test

Training (2)

Initialize model and Adam optimizer parameters
Set number of epochs and batch size
for each epoch do
  for each training batch do
    Perform a forward pass with the input data
    Compute $\mathcal{L}$
    Compute gradient of $\mathcal{L}$ w.r.t. all trainable parameters
    Update all trainable parameters using Adam optimizer
  end for
  Compute and store the mean loss $\mathcal{L}^\star$ over validation set
  if $\mathcal{L}^\star <$ all previous $\mathcal{L}^\star$ **then**
    Update stored model weights
  end if
end for
Assign stored model weights to final model
Assess model on test set

FIGURE – Training algorithm overview

Evaluation metrics

$$\text{Mean pixel accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

$$\text{Intersection over union} = \frac{\text{intersection of predicted and true object masks}}{\text{union of predicted and true object masks}}$$

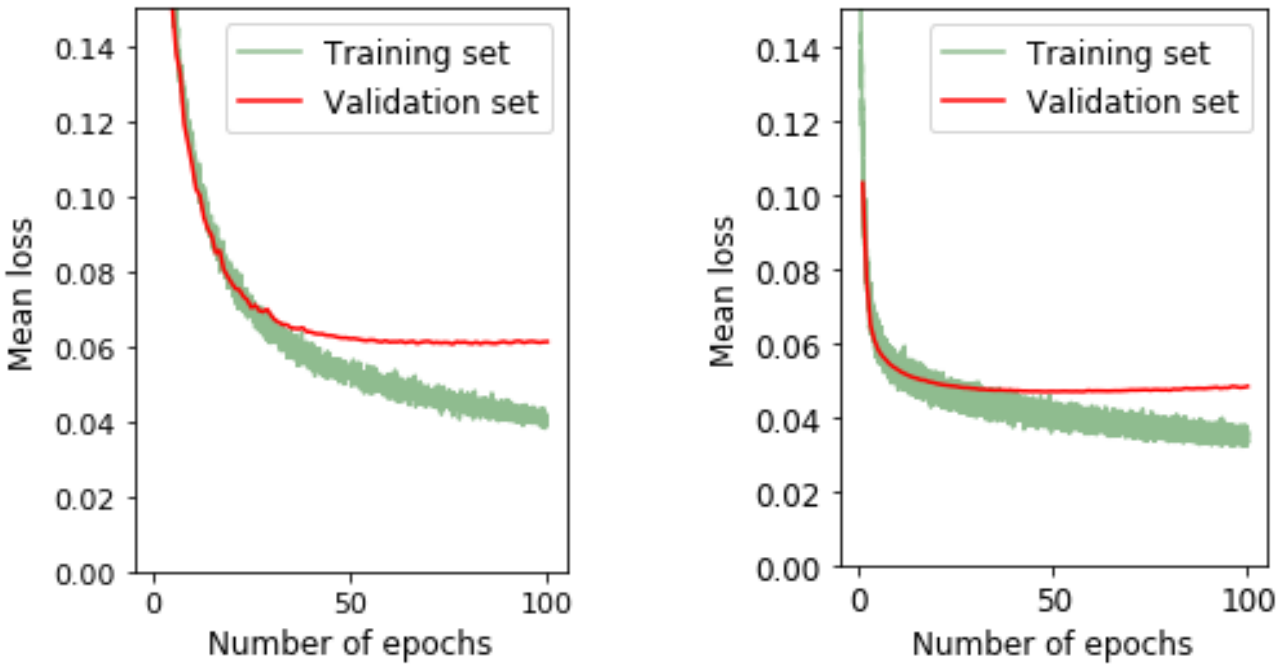
$$\text{Precision} = \frac{\text{correct object predictions}}{\text{all object predictions}}$$

$$\text{Recall} = \frac{\text{correct object predictions}}{\text{all object ground truths}}$$

- Output is thresholded for prediction, e.g. 0.5

Outline

Loss curves



(a) 7,000 training samples

(b) 70,000 training samples

FIGURE – Evolution of the mean value of $\mathcal{L}$ measured on the training and validation sets during learning.

Quantitative results

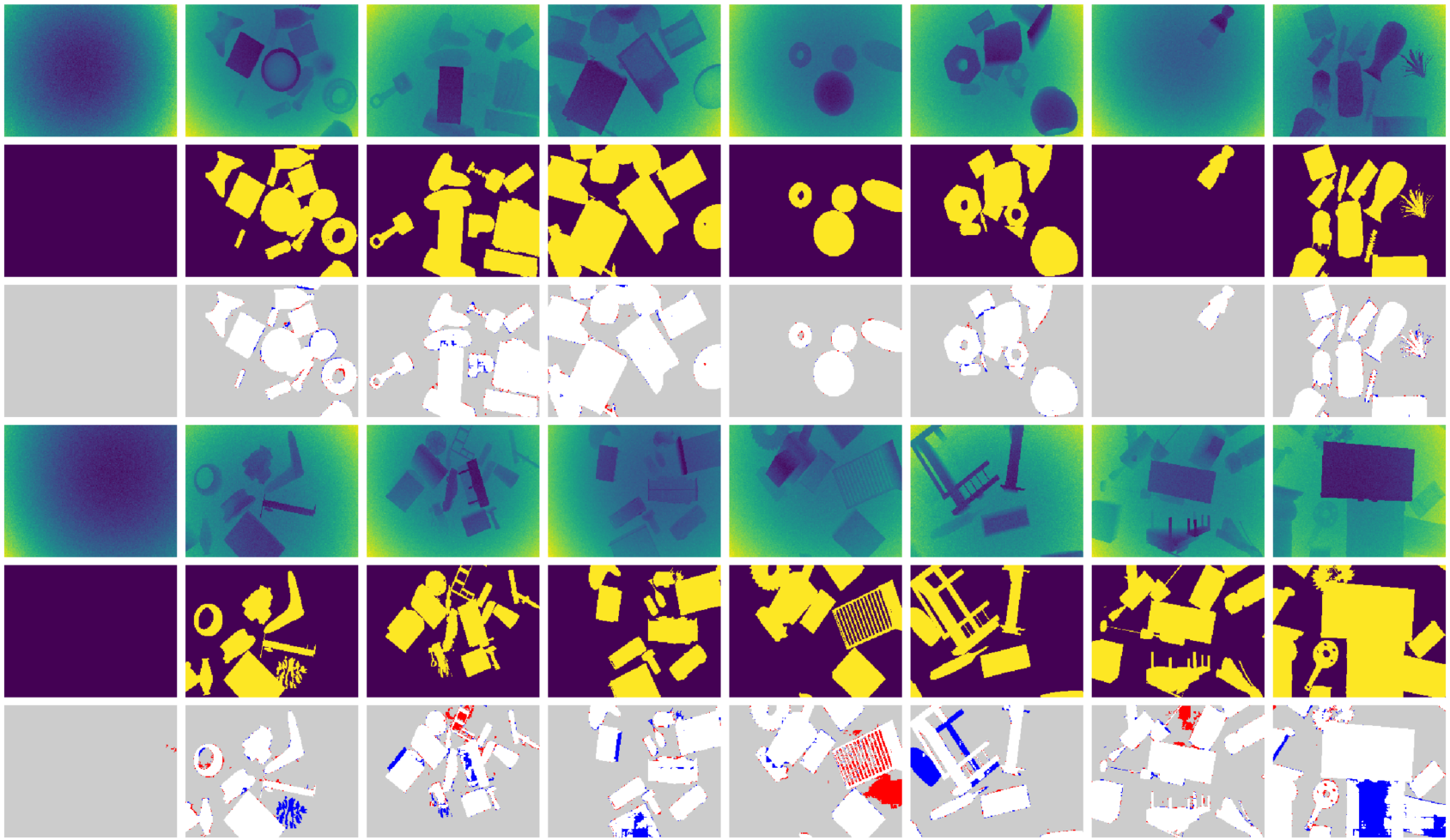
Training dataset size	Pixel acc.	IoU	Precision	Recall
7,000 samples	97.87	87.59	92.82	94.27
70,000 samples	**98.31**	**90.84**	**95.30**	**95.20**

TABLE – Evaluation scores of FCN-8s (with VGG16 encoder) models on 10,000 new samples

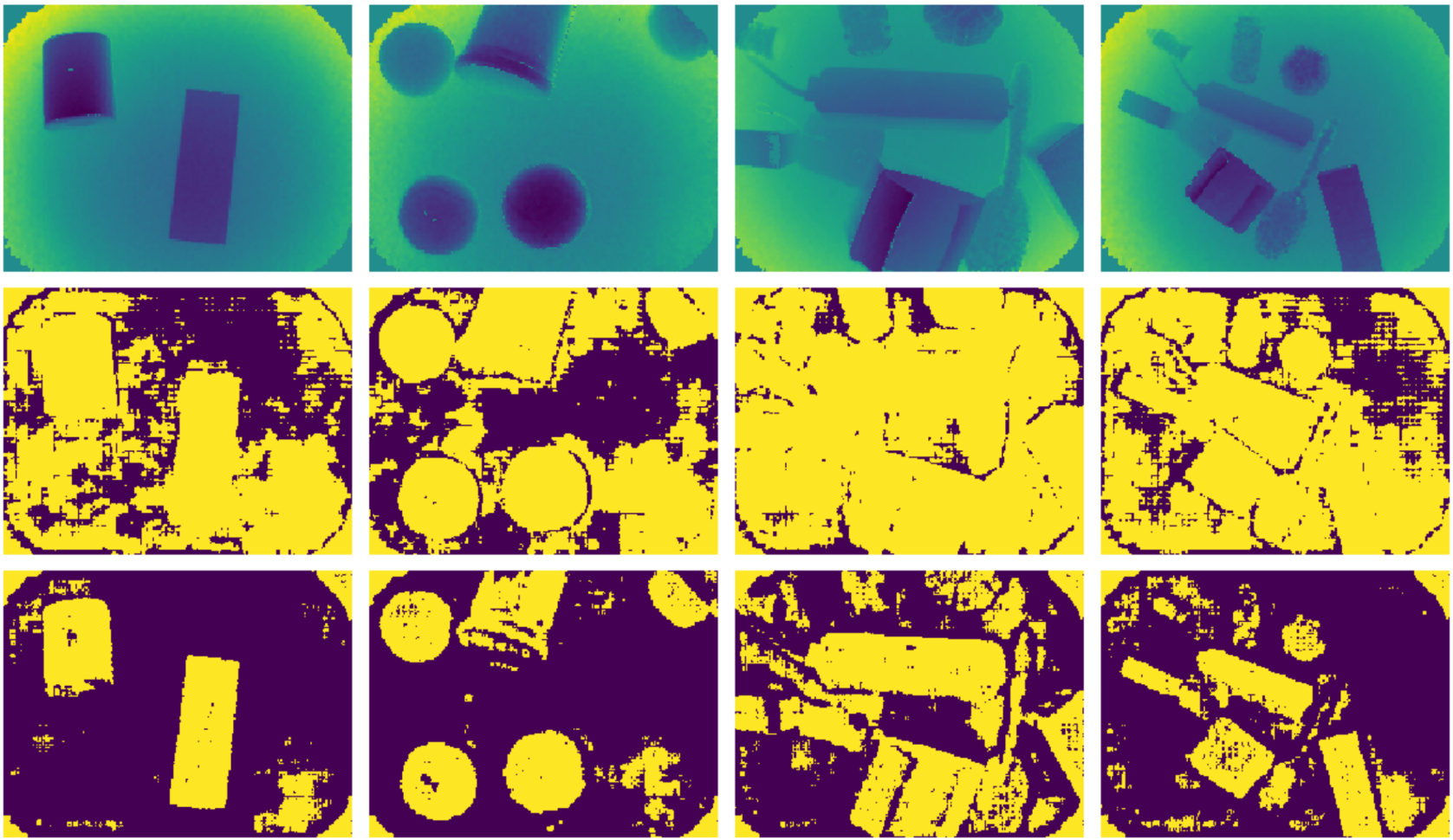
Model	Pixel acc.	IoU	Precision	Recall	Forward pass
VGG11	98.35	**91.80**	**96.07**	95.40	**1.31ms**
VGG16	98.31	90.84	95.30	95.20	1.66ms
VGG19	**98.36**	91.56	95.80	**95.43**	1.86ms

TABLE – Effect of changing the encoder architecture on overall performance of models measured on 10,000 new samples

Qualitative results



Performance on real data



Outline

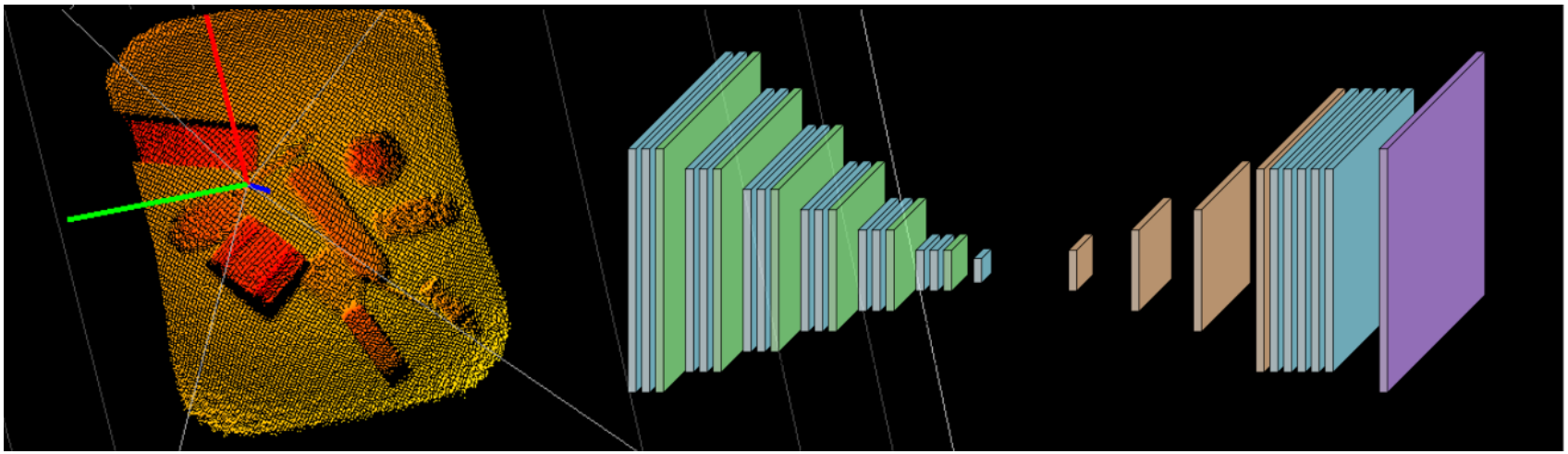
Discussion

- Poor prediction with small/flat/complex objects
- Possible improvements include
  - Control object shape
  - Review environment assumptions
  - Change the encoder
  - Optimize initialization and Adam hyperparameters
  - **Study real sensor noise**

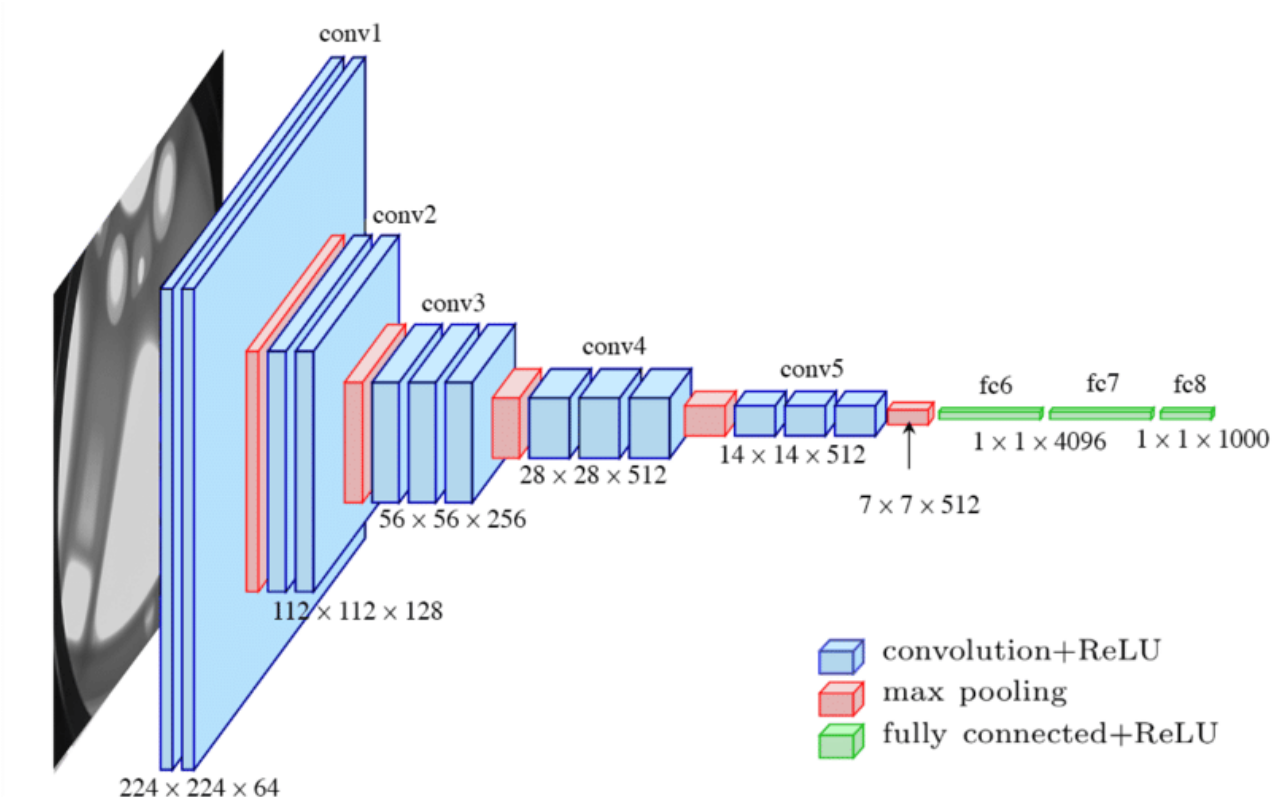
Conclusion

In this project we worked on

- Binary semantic segmentation
- Automatic generation and labelling of realistic depth images
- Implementation and training of a FCN
- Model assessments on simulated (**>98%** accuracy) and real data
- Ideas for possible improvements

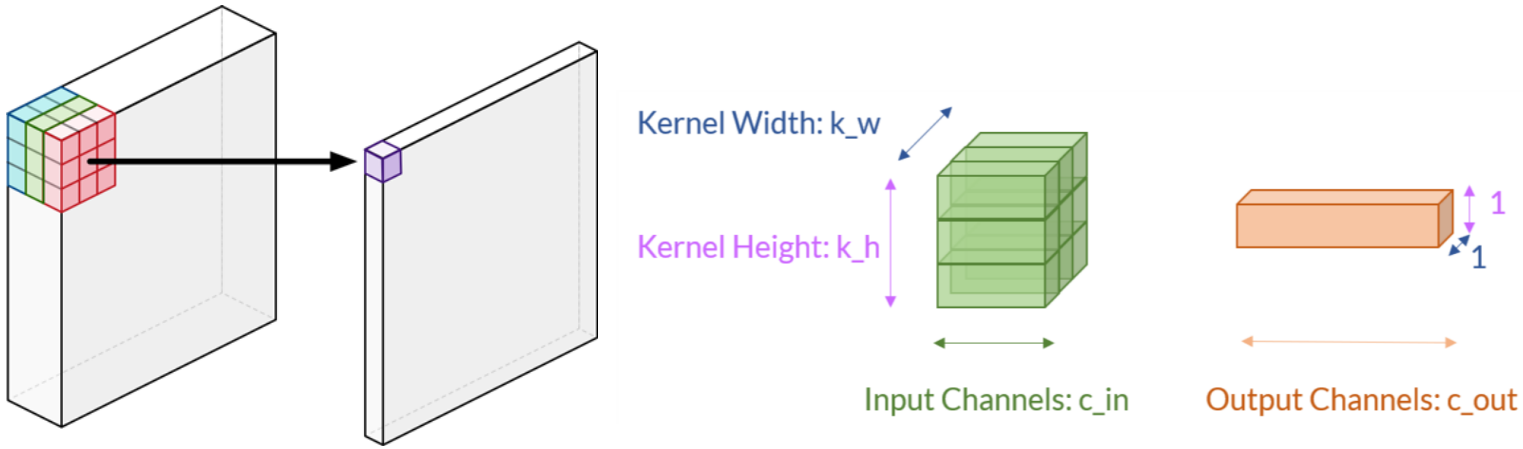


About VGG16



- Convolutions : 3×3 receptive fields, stride 1, padding 1
- MaxPool : 3×3 windows, stride 2

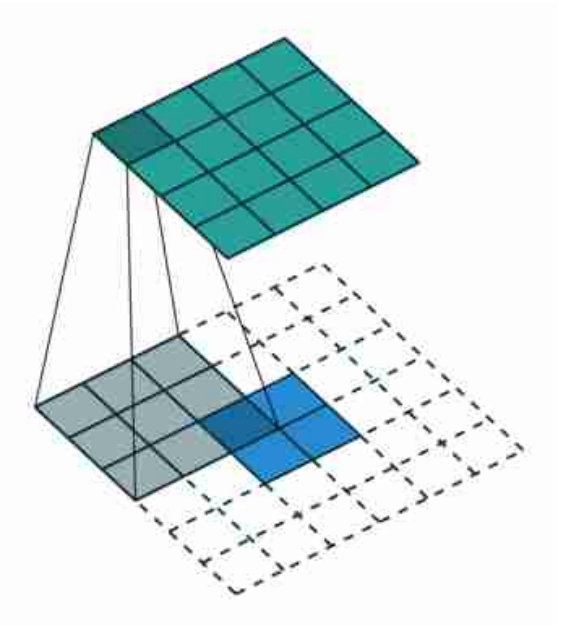
About convolution layers



- Number of learnable parameters : $(k_w \times k_h \times c_{in} \times c_{out}) + c_{out}$

About deconvolution layers

- Learnable upsampling
- *Transposed* convolution
  - A convolution operation $\mathbf{x} \circledast \mathbf{u}$ between tensor $\mathbf{x}$ and kernel $\mathbf{u}$ can be expressed as the reshaped result $\mathbf{h}$ of a matrix operation $\mathbf{U}v(\mathbf{x}) = v(\mathbf{h})$
  - A deconvolution operation writes $\mathbf{U}^T v(\mathbf{x}) = v(\mathbf{h})$ with appropriate reshaping



- Number of learnable parameters : $(k_w \times k_h \times c_{out} \times c_{in}) + c_{in}$

About batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Ioffe and Szegedy, Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015

About Adam optimization

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

- Kingma and Ba, Adam : A Method for Stochastic Optimization, 2014

About Kaiming initialization

- Initial weights are drawn in a normal distribution with mean 0 and standard deviation σ given by

$$\sigma = \sqrt{\frac{2}{n_{in}}} \tag{3}$$

  where n_{in} is the layer input dimension

- Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification, 2015