Processing of 3D Scans using Machine Learning

Author: Bc. Lukáš Gajdošech
Supervisor: RNDr. Martin Madaras, PhD.

Master's Thesis Website

May 13, 2020

Problem

Input

Organized pointcloud (single-view). The dimensions of the scan are not fixed. For every point, we know the following:

- position
- intensity
- normal

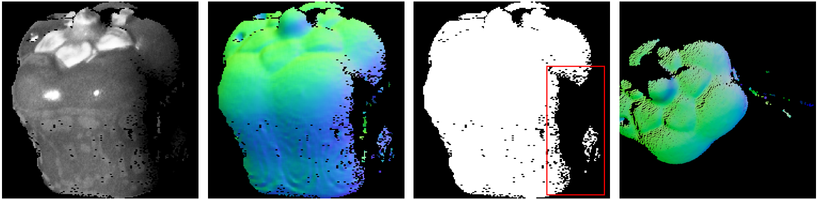
Output

Prediction masks for different tasks, such as:

- **artefacts removal (binary mask)**
- semantic segmentation
- material segmentation
- ...

Goal

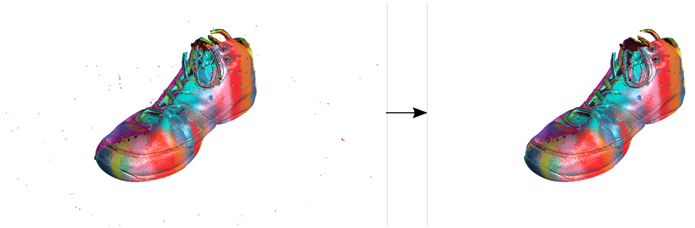
- All these tasks *may be solvable* using ML systems.
- The first task to examine is **artefacts removal**.
- The *goal* is to develop a **modular pipeline** for these tasks, evaluate it and compare to existing non-ML solutions.
- The input scans are obtained using *Photoneo 3D Scanner* (light-structured scanner) and/or *virtual scanner*. Part of the work is to collect and prepare a suitable **dataset**.



(a) intensity map (b) normal map (c) ground truth (d) different view

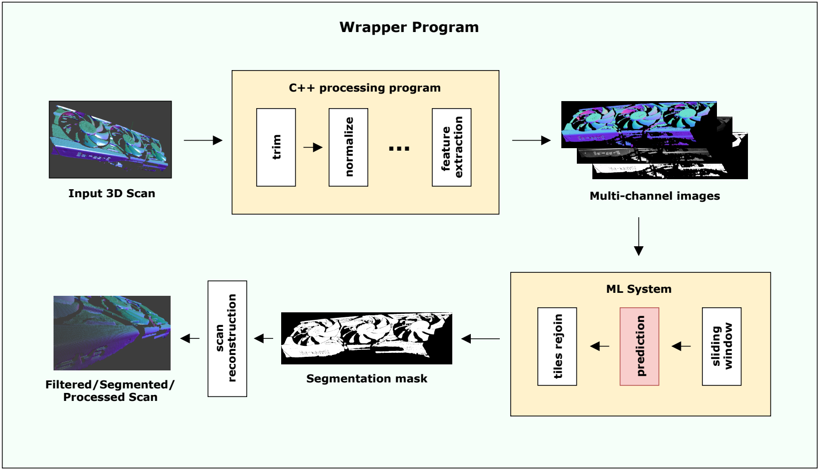
Existing Solution

- The existing (non-ML) solution for the artefacts removal task exploits the redundancy and overlaps between scans.



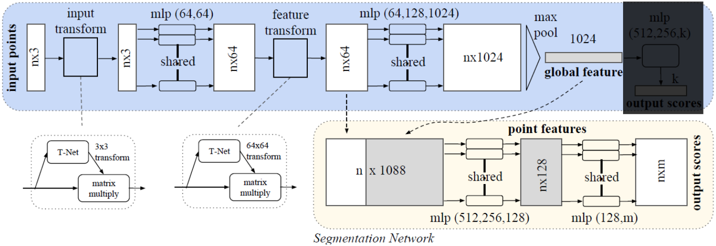
- *Disadvantage*: We need several scans (multi-view). Trained ML systems *should* be able to do this from a single scan.

Modular Pipeline



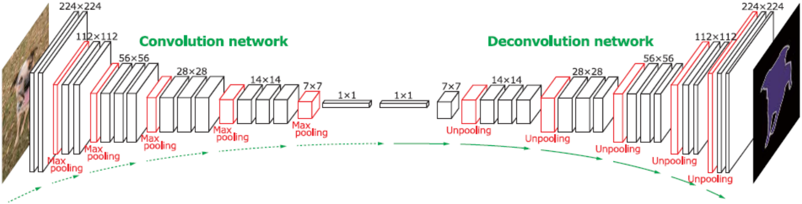
Multiple possibilities for the ML prediction algorithm (ex. CNN architectures), easily changeable for different tasks and models.

PointNet: DL for 3D Classification and Segmentation



- NN trained directly on *unorganized* 3D point cloud data.
- Classifies using $1024 + 64 = 1088$ (global + point) features.
- Focuses on *permutation* and *transformation* invariance, plus the ability to capture *local interactions*.
- Formulates the problem as an approximation of general symmetric $f(x_1, \ldots, x_n)$. Set of these $[f_1, \ldots, f_k]$ can be interpreted as a global signature of the set, which can be concatenated with local information $h(x_i)$ modelled by MLPs.

Deconvolution Network for Semantic Segmentation

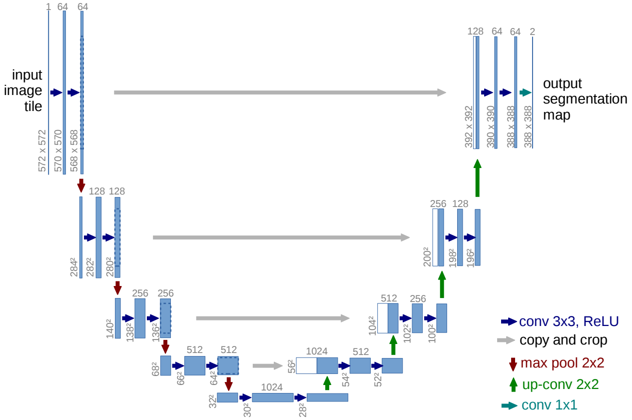


- Simple convolutional **encoder-decoder** structure.
- Comparison to FCN with upsampling bilinear filter.
- Interesting idea to first generate object proposals and then run the network on each generated sub-image G_i. Final segmentation map is obtained by aggregation:

$$P(x, y, c) = \max_i G_i(x, y, c), \forall c \in Classes$$

This should eliminate object scale variations.

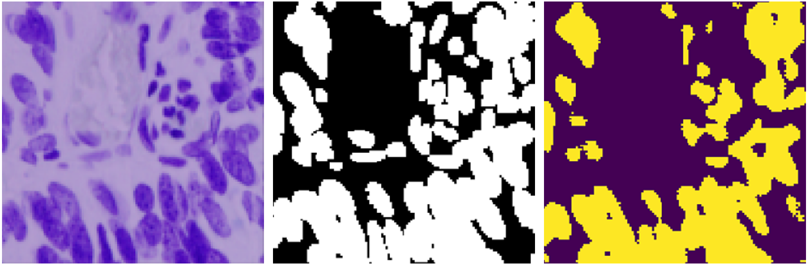
U-Net: CNN for Biomedical Image Segmentations



- Concat the feature maps from encoder to decoder!
- Sparse medical data, various *augmentation techniques*.

Other Sources

- Kaggle.com Competitions:
  - TGS Salt Identification Challenge
  - Nuclei Segmentation - simple example notebook
- CESCG 2020 Academy - Brain Tumor Segmentation (Jupyter Notebook).
- Overall, network for segmentation of images is a *hot topic*.



(a) input (b) ground truth (c) prediction

Figure: Vanilla Unet trained for 2018 Data Science Bow competition.

Other works I

Charles, R., Su, H., Kaichun, M., and Guibas, L.
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.
In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (07. 2017), pp. 77–85.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y.
Learning Hierarchical Features for Scene Labeling.
IEEE Trans. Pattern Anal. Mach. Intell. 35, 8 (2013), 1915–1929.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H.
Brain tumor segmentation with deep neural networks.
Medical Image Analysis 35 (Jan 2017), 18–31.

Other works II

IGLOVIKOV, V., AND SHVETS, A.
Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation, 2018.

LONG, J., SHELHAMER, E., AND DARRELL, T.
Fully convolutional networks for semantic segmentation, 2014.

NOH, H., HONG, S., AND HAN, B.
Learning Deconvolution Network for Semantic Segmentation, 2015.

RONNEBERGER, O., FISCHER, P., AND BROX, T.
U-Net: Convolutional Networks for Biomedical Image Segmentation.
vol. 9351, pp. 234–241.

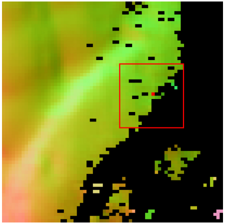
(Example) Approaches

- After acquiring small dataset, first ideas were tried.
- Data is pre-processed by a standalone *C++* program (according to proposed *pipeline*) and the ML part is implemented in Python, using the *Jupyter Notebook* environment and libraries such as *Tensorflow* and *Sklearn*.

Approach 1 - Idea

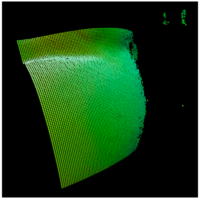
- Take $n \times n$ (n is hyperparameter) neighborhood of a point.
- Make prediction for the center point c based on neighbors $o^{(i)}; 0 \leq i \leq n \times n; \;\; o^{(i)} \neq c$
- Create a feature vector for each $o^{(i)}$: $|c_{depth} - o^{(i)}_{depth}|, |c_{intensity} - o^{(i)}_{intensity}|, dist(c_{normal}, o^{(i)}_{normal})$
- Unroll the matrix of feature vectors into a single vector with $(n \times n - 1) \times 3$ elements.

The resulting vector can be simply fed into a classifier, such as Multi Layer Perceptron or SVM, RFC, etc.

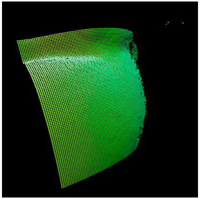


Approach 1 - Problems

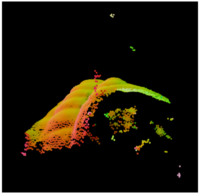
- Small $n \rightarrow$ lack of context, big $n \rightarrow$ lack of details.
- Separate prediction for each point $\rightarrow$ *extremely* slow.



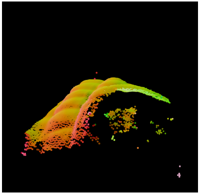
(a) bigger artefact cluster



(b) homogeneous neighborhood



(c) isolated geometry

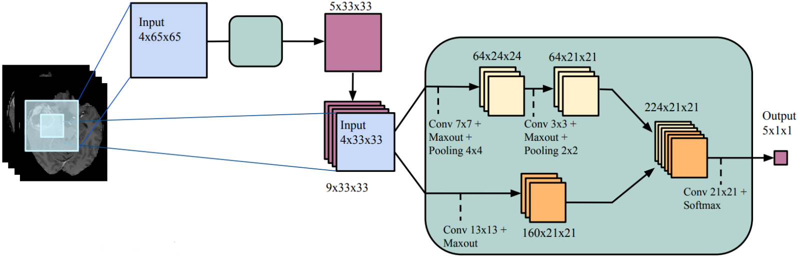


(d) predicted as artefact

Approach 1 - Upgrades

The problem with context vs details *may be solvable* using a 2 branch network, as in (Havaei et al., 2017). In this approach, we have both a smaller and bigger window in a single network.

However, the low performance still remains, so we will probably NOT develop this approach any further.



(a) Cascaded architecture, using input concatenation (InputCascadeCNN).

Approach 2 - UNet

- The dimensions of the scans can be arbitrary, for simplicity, we tile it into windows of constant size.
- During the training, we can also *augment the data* with rotating and flipping the tiles, providing *robustness* and *transformation invariance*.
- Using *Binary Cross-Entropy* loss function and F_1 *Score* as a metric, we trained vanilla UNet on small dataset of scans.

$$BCE = -\frac{1}{M}\sum_{i=1}^{M} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

(ground truth) $y_i = 1 \Leftrightarrow$ true geometry, $y_i = 0 \Leftrightarrow$ artefact

(prediction) $p(y_i) \in \langle 0, 1\rangle \Leftrightarrow$ returned by ML system

Approach 2 - Results

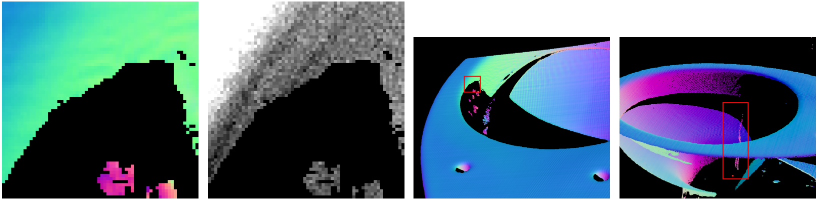


Figure: Normal map, intensity map and position of the tile in test image.

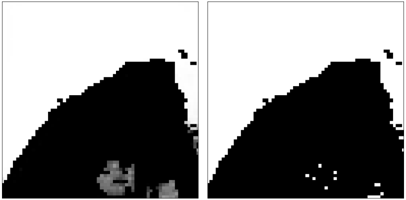


Figure: Raw and thresholded prediction from the UNet.

Evaluation

- Data are *skewed*, $\sim 98\%$ of the points are true geometry.
- *Accuracy* is not useful in this case, better to use metrics based on the *confusion matrix*.
- In the *Approach 2*, we used *BCE* loss and $F_1 Score$ metric, however, that may still not be the best formulation of our *optimization* problem.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Imbalanced Dataset

		Prediction	
		True	Artefact
Actual	True	18508 (TP)	53 (FN)
	Artefact	112 (FP)	166 (TN)

$$Accuracy(CM) = \frac{TP + TN}{TP + TN + FP + FN} \sim \mathbf{0.991}$$

$$PPV(CM)\ (Precision) = \frac{TP}{TP + FP} \sim 0.994$$

$$TPR(CM)\ (Recall) = \frac{TP}{TP + FN} \sim 0.997$$

$$TNR(CM)\ (Selectivity) = \frac{TN}{TN + FP} \sim 0.60$$

$$\textbf{Balanced Accuracy}(CM) = \frac{TPR + TNR}{2} \sim \mathbf{0.80}$$

Confusion Matrix

- *Type II* errors (FN) are critical, they remove *true geometry*.
- On the other hand, *type I* errors can be improved by re-running the filtration several times.
- In other words, we want to minimize *false omission rate*:

$$FOR(CM) = \frac{FN}{FN + TN} \sim 0.24$$

- This is the same as minimizing the conditional probability $p(actual = true \mid prediction = artefact)$ or maximizing the *negative predictive value*:

$$NPV(CM) = \frac{TN}{TN + FN} = 1 - FOR(CM) \sim 0.76$$

- *Conclusion:* improve **architecture** and use better **loss func**.