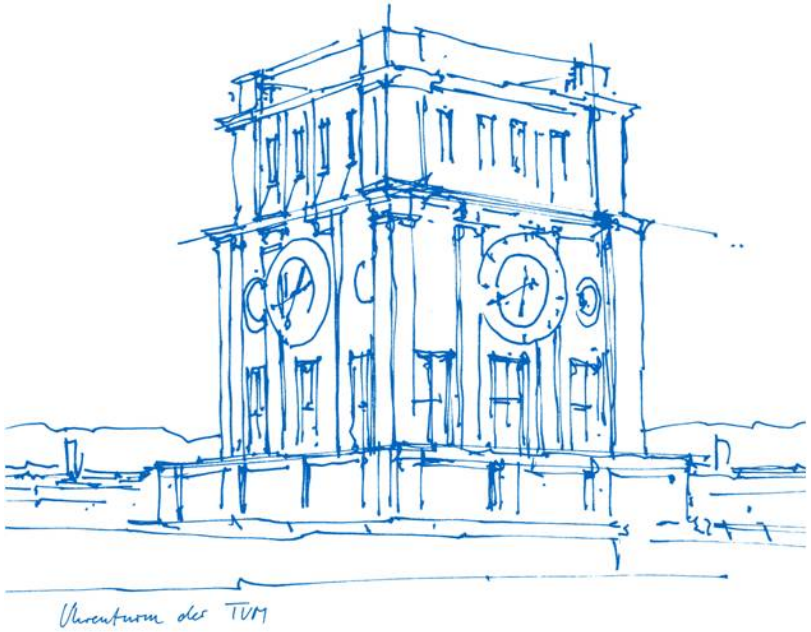
Sparkmap

Marsil Zakour, Sebastian Schlegel, Vladimir Yugay
Technical University of Munich
Department of Mathematics
Data Innovation Lab
Garching, 18. February 2020



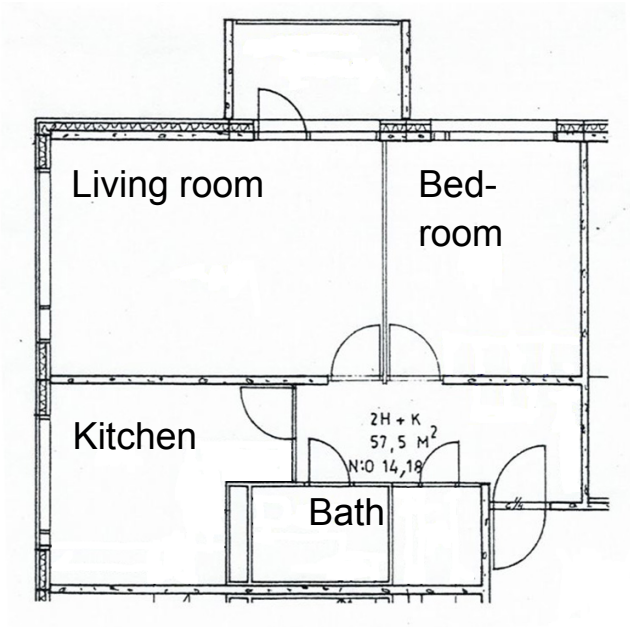
Agenda

- Introduction & Objectives
- Related work
- Dataset
- "Feature Vector" Approach
- Solution
  - Segmentation Network
  - Room Proposal Extraction
  - Algorithm for placing icons
- Results
- Conclusion & limitations

Introduction & Objectives

For a lot of simple architectural tasks a person is needed

One reason: Used data in the form of rasterized floorplan images

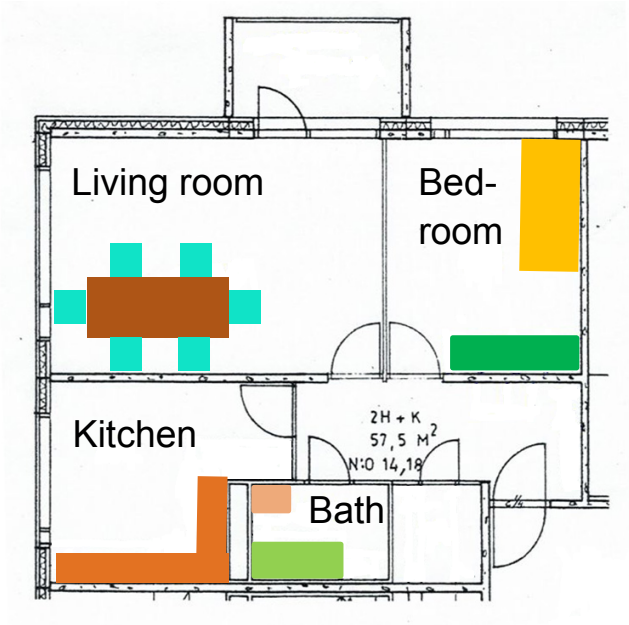


Introduction & Objectives

For a lot of simple architectural tasks a person is needed

One reason: Used data in the form of rasterized floorplan images

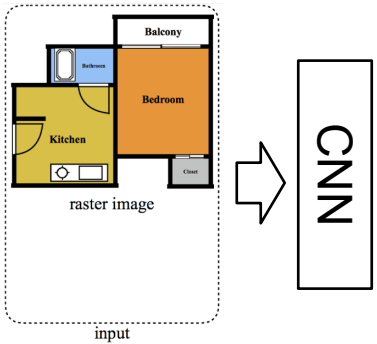
Goal of this Project: To place icons (furniture, facilities) on an "empty" floorplan



Related work – Liu et al. 2017

A multi-step process using a convolutional neural network (CNN) for automatically parsing rasterized floorplan images

Approach:

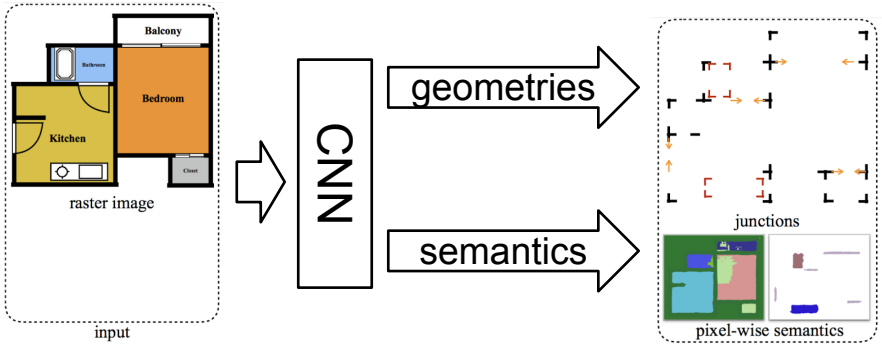


Related work – Liu et al. 2017

A multi-step process using a convolutional neural network (CNN) for automatically parsing rasterized floorplan images

Approach:

- Extracting geometric and semantic information independently

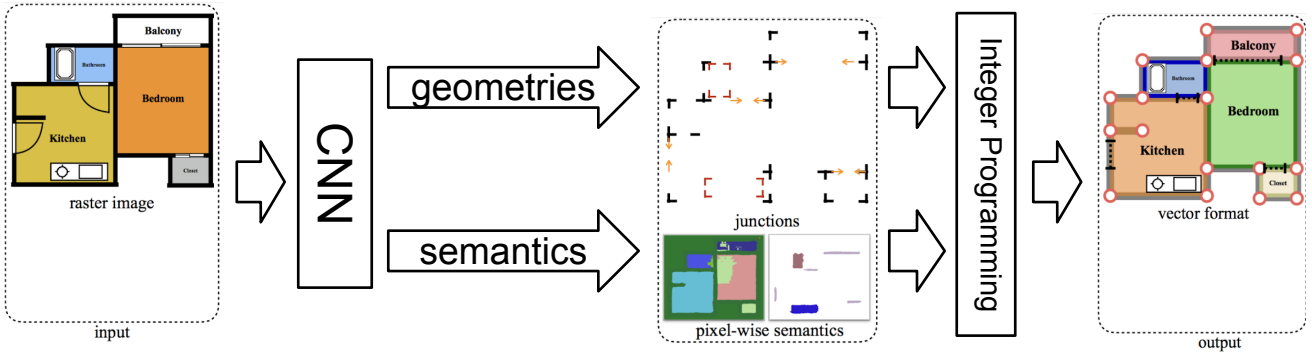


Related work – Liu et al. 2017

A multi-step process using a convolutional neural network (CNN) for automatically parsing rasterized floorplan images

Approach:

- Extracting geometric and semantic information independently
- Merging both rule-based using Integer Programming



Related work – Liu et al. 2017

Geometric information is represented through "junction points"

- 4 different junctions for "openings" (windows, doors):

Related work – Liu et al. 2017

Geometric information is represented through "junction points"

- 4 different junctions for "openings" (windows, doors):

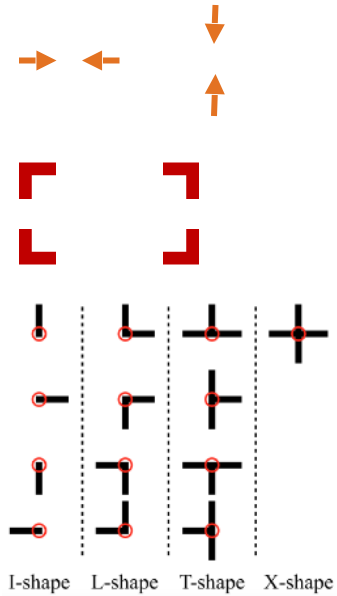
- 4 different junctions for icons:

Related work – Liu et al. 2017

Geometric information is represented through "junction points"

- 4 different junctions for "openings" (windows, doors):
- 4 different junctions for icons:
- 13 different junctions for walls:

In total: **21** different junction types

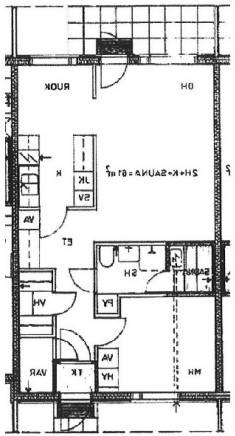


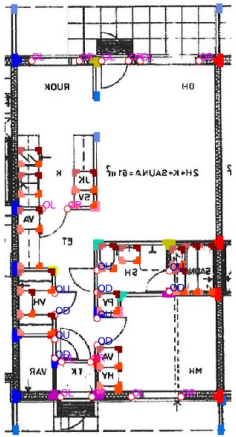
Related work – Liu et al. 2017

Network (modified ResNet 152) output:

Geometric information:

- 21 heatmaps – one regressed for every junction type





Related work – Liu et al. 2017

Network (modified ResNet 152) output:

Geometric information:

- 21 heatmaps – one regressed for every junction type

Semantic information:

- One per-pixel classification for room types

Related work – Liu et al. 2017

Network (modified ResNet 152) output:

Geometric information:
- 21 heatmaps – one regressed for every junction type

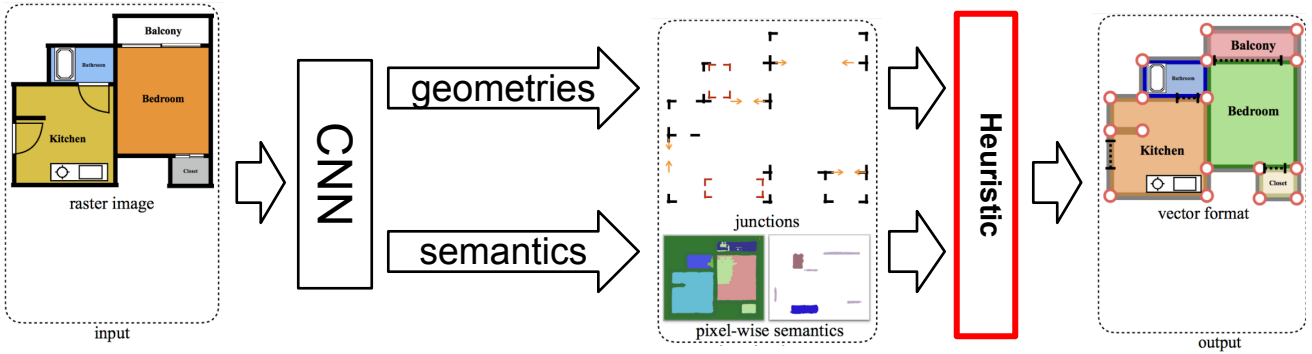
Semantic information:
- One per-pixel classification for room types
- One per-pixel classification for icons, windows & doors

Related work – Kalervo et al. 2019 ("CubiCasa")

Modifying the approach of Liu et al.

Differences:

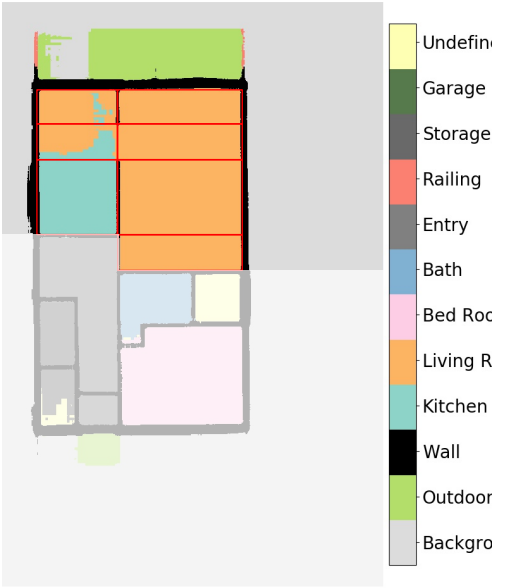
- Applying automatic weighting to the multi-task CNN of Liu et al.
- Instead of Integer Programming a heuristic is used for merging



Related work – Kalervo et al. 2019 ("CubiCasa")

Merging geometric and semantic information

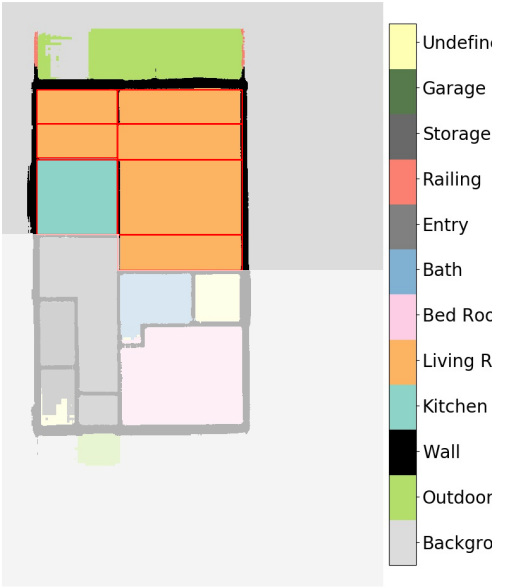
- Split floorplan in a grid of rectangular cells using triplets of Junction points



Related work – Kalervo et al. 2019 ("CubiCasa")

Merging geometric and semantic information

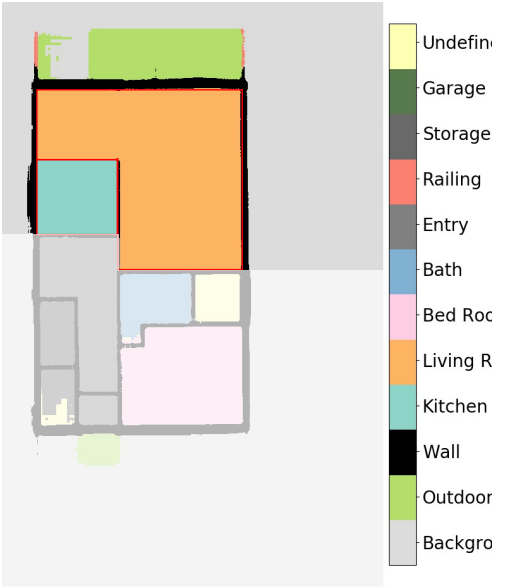
- Split floorplan in a grid of rectangular cells using triplets of Junction points
- Label cells applying pixel-wise maximum voting



Related work – Kalervo et al. 2019 ("CubiCasa")

Merging geometric and semantic information

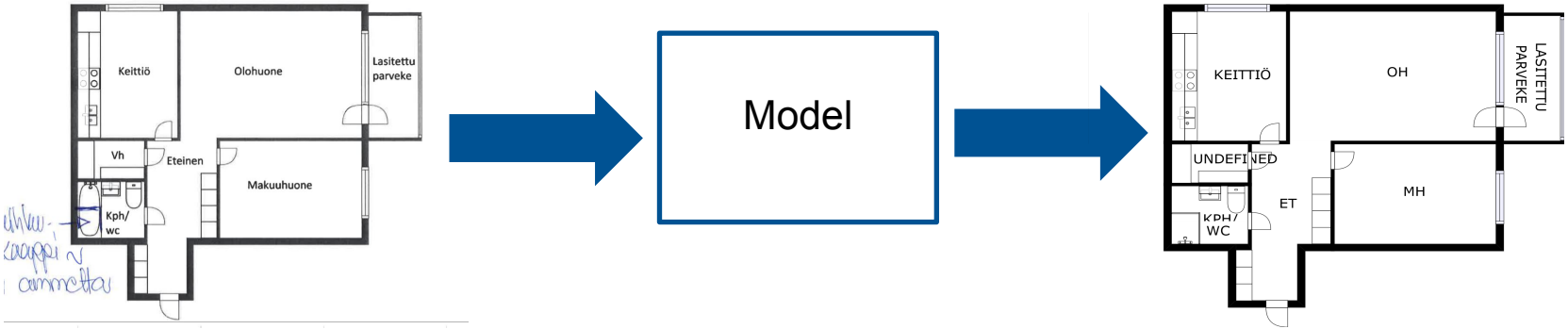
- Split floorplan in a grid of rectangular cells using triplets of Junction points
- Label cells applying pixel-wise maximum voting
- Merge cells with the same label if no separating wall is in between



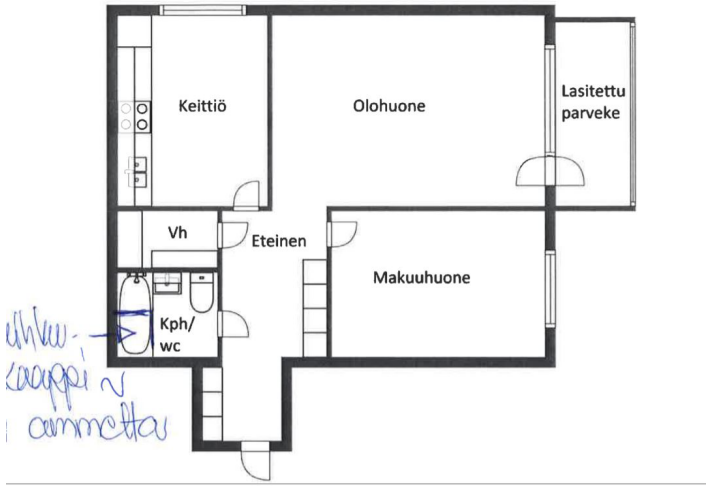
Agenda

- Introduction & Objectives
- Related work
- Dataset
- "Feature Vector" Approach
- Solution
  - Segmentation Network
  - Room Proposal Extraction
  - Algorithm for placing icons
- Results
- Conclusion & limitations

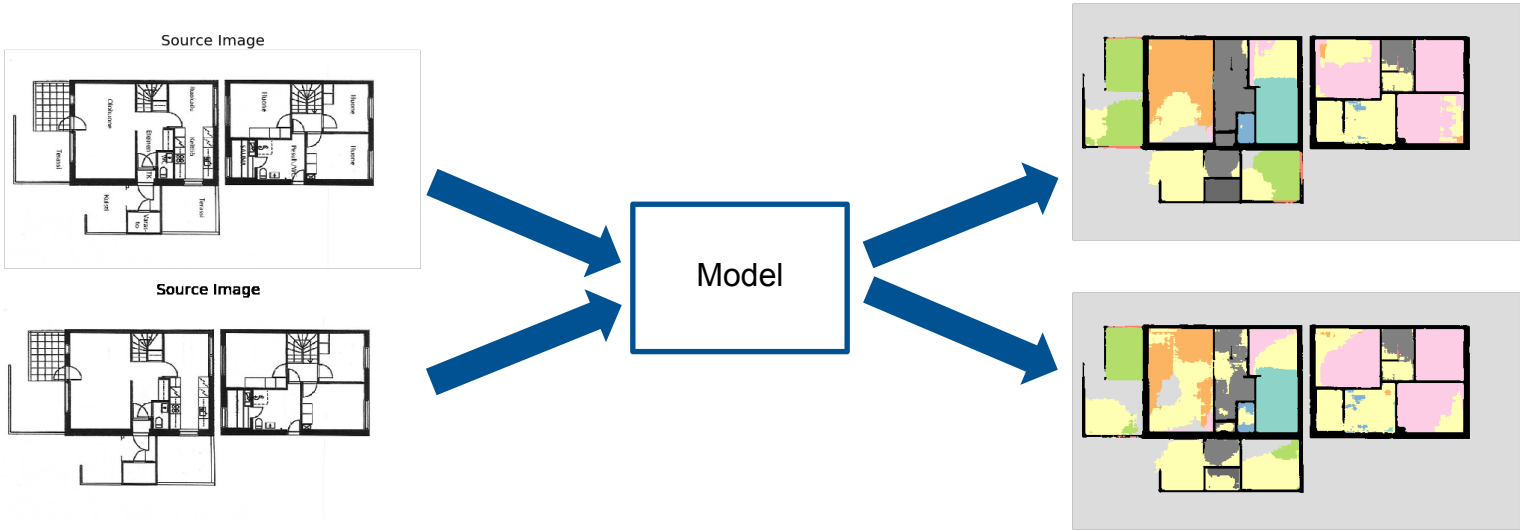
CubiCasa Dataset



CubiCasa input



CubiCasa model generalization

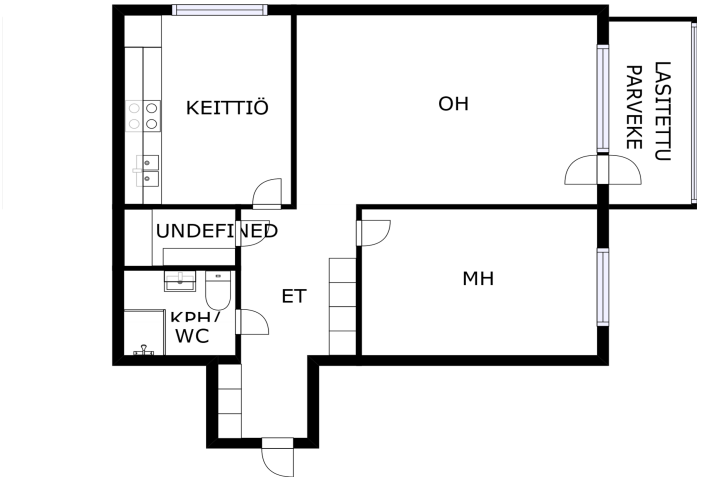


CubiCasa model evaluation

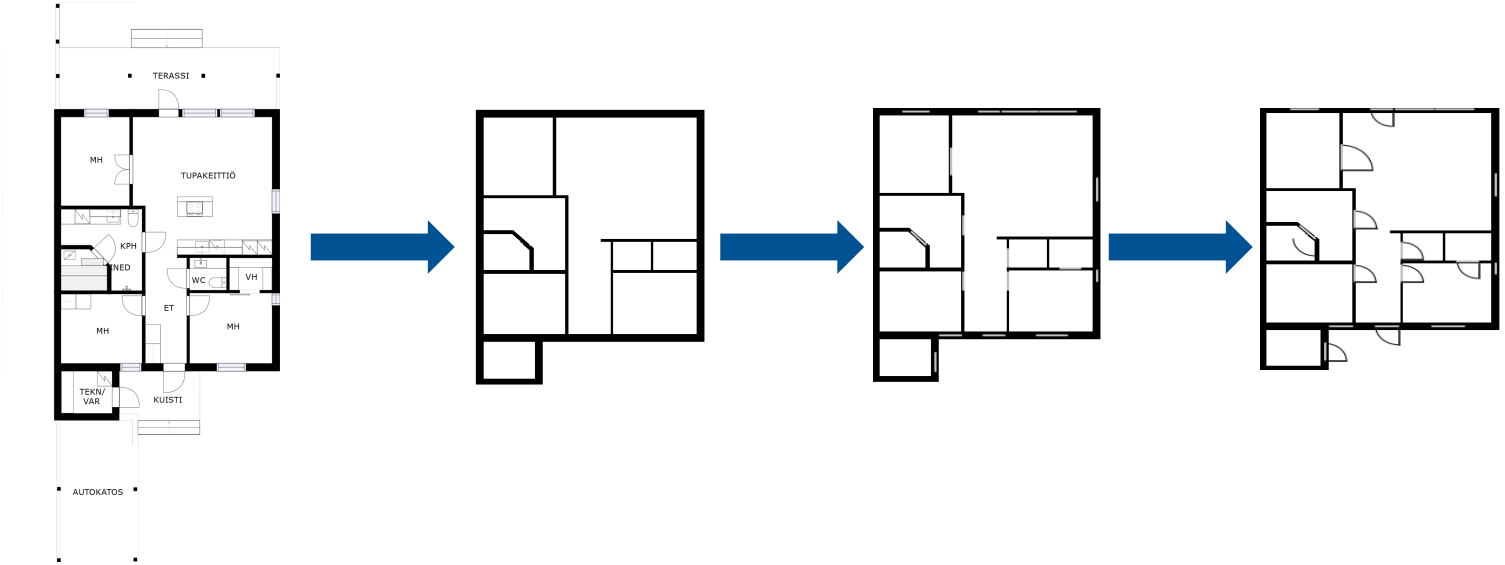
```
2019-11-08 15:16:43,358 - eval - INFO - IoU & Acc
2019-11-08 15:16:43,358 - eval - INFO - Background & 88.9 & 95.6 \\ \hline
2019-11-08 15:16:43,358 - eval - INFO - Outdoor & 56.2 & 74.0 \\ \hline
2019-11-08 15:16:43,358 - eval - INFO - Wall & 65.5 & 72.9 \\ \hline
2019-11-08 15:16:43,358 - eval - INFO - Kitchen & 63.9 & 85.5 \\ \hline
2019-11-08 15:16:43,358 - eval - INFO - Living Room & 73.1 & 84.8 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Bedroom & 76.2 & 93.5 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Bath & 56.8 & 63.7 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Hallway & 57.6 & 75.0 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Railing & 13.3 & 13.9 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Storage & 53.4 & 57.2 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Garage & 0.0 & 0.0 \\ \hline
2019-11-08 15:16:43,359 - eval - INFO - Other rooms & 49.7 & 64.4 \\ \hline
2019-11-08 15:16:43,359
```

```
228 2019-11-08 15:05:33,043 - eval - INFO - IoU & Acc
229 2019-11-08 15:05:33,043 - eval - INFO - Background & 80.3 & 95.8 \\ \hline
230 2019-11-08 15:05:33,043 - eval - INFO - Outdoor & 50.3 & 66.0 \\ \hline
231 2019-11-08 15:05:33,043 - eval - INFO - Wall & 64.0 & 72.0 \\ \hline
232 2019-11-08 15:05:33,043 - eval - INFO - Kitchen & 50.6 & 68.0 \\ \hline
233 2019-11-08 15:05:33,044 - eval - INFO - Living Room & 24.8 & 28.4 \\ \hline
234 2019-11-08 15:05:33,044 - eval - INFO - Bedroom & 37.0 & 45.0 \\ \hline
235 2019-11-08 15:05:33,044 - eval - INFO - Bath & 8.4 & 9.0 \\ \hline
236 2019-11-08 15:05:33,044 - eval - INFO - Hallway & 21.2 & 23.8 \\ \hline
237 2019-11-08 15:05:33,044 - eval - INFO - Railing & 12.1 & 12.9 \\ \hline
238 2019-11-08 15:05:33,044 - eval - INFO - Storage & 5.3 & 5.3 \\ \hline
239 2019-11-08 15:05:33,044 - eval - INFO - Garage & 0.0 & 0.0 \\ \hline
240 2019-11-08 15:05:33,044 - eval - INFO - Other rooms & 33.3 & 64.8 \\ \hline
241 2019-11-08 15:05:33,044
```

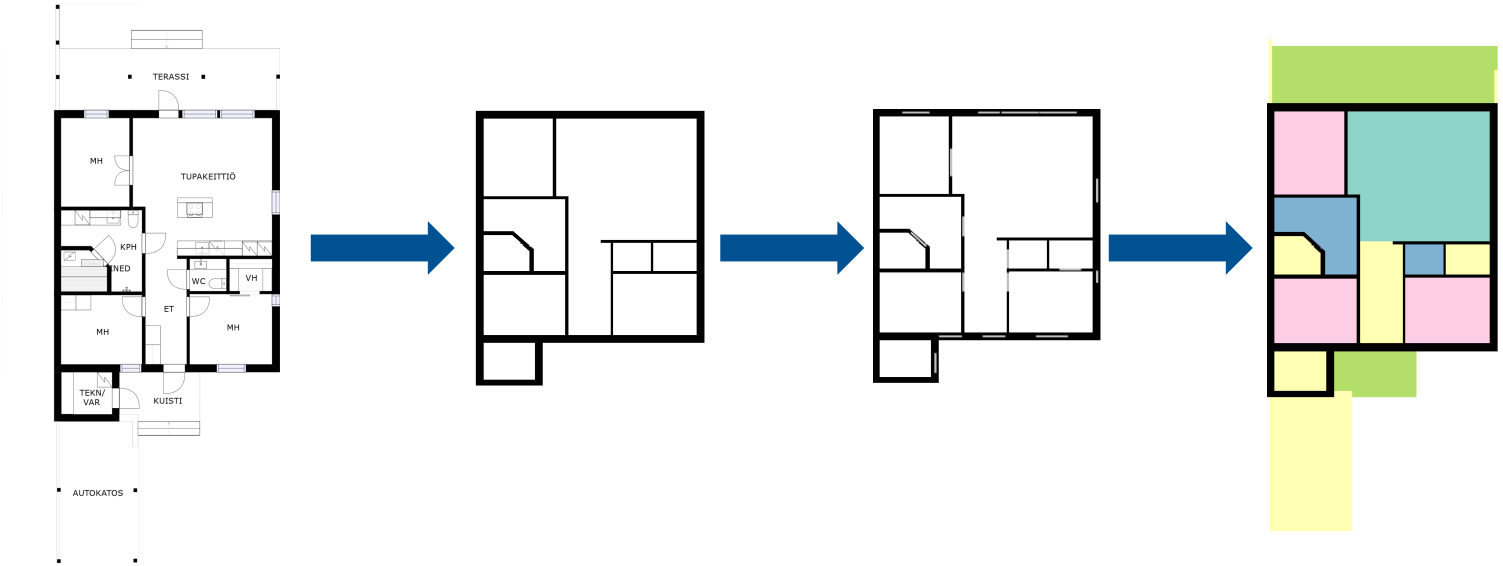
CubiCasa label



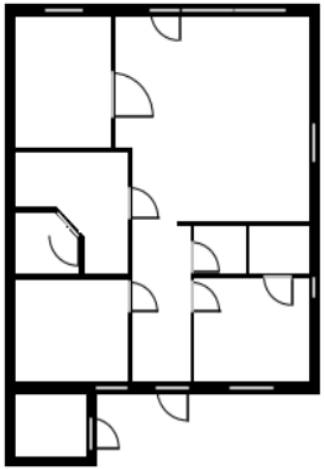
Input generation

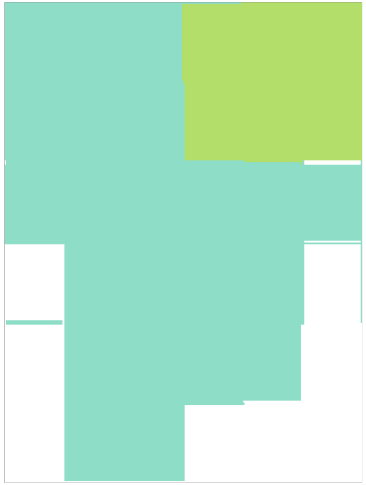


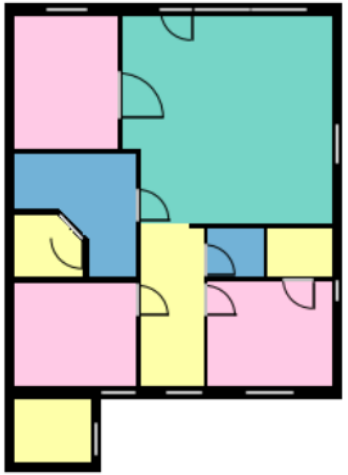
Label generation



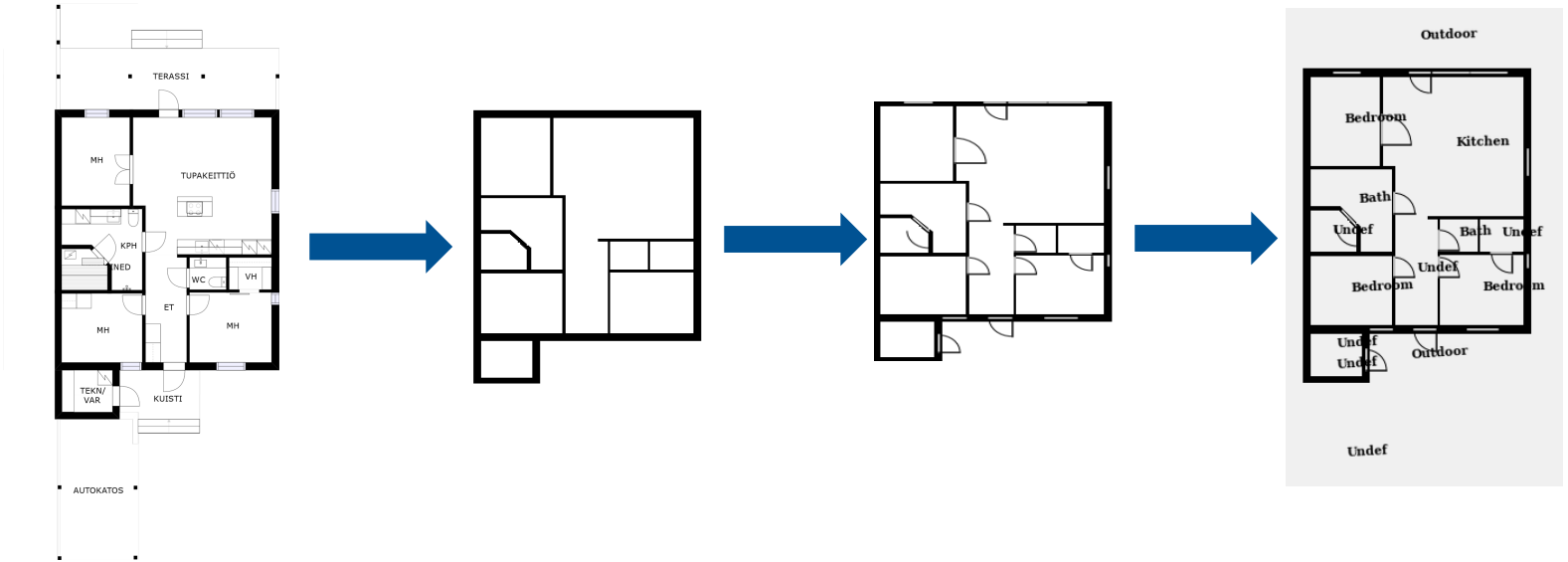
Input generation







Input generation



Dataset randomization

- Each image has a random texture
- Each text annotation has a random font size, weight and family
- Each text annotation is rotated from 0 to 11 degrees

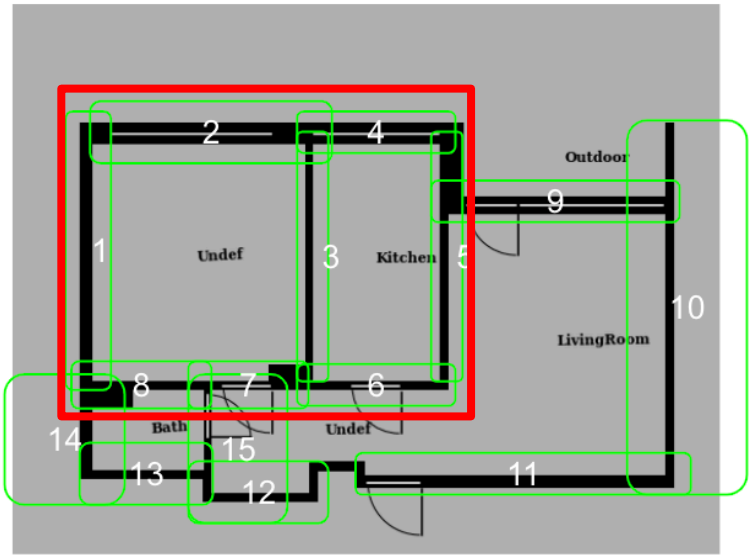
Dataset statistics

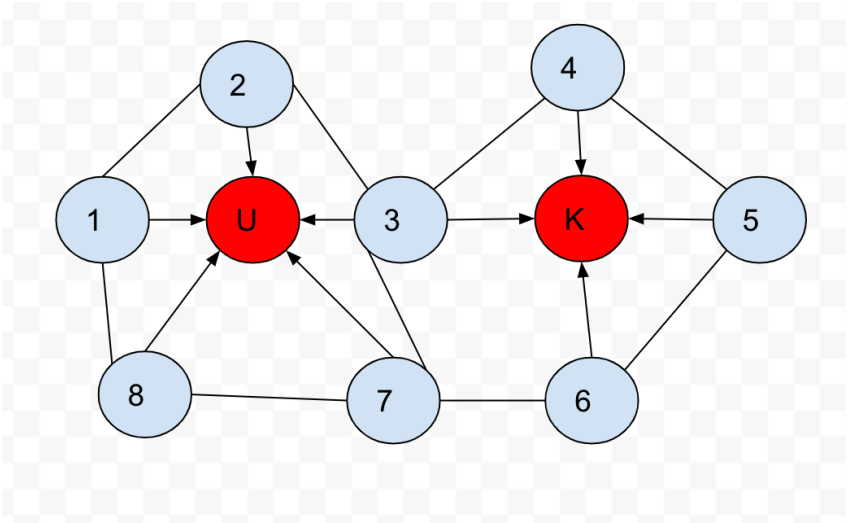
- Number of room types were reduced from 65 to 8
- Only rooms with kitchen and bathroom were kept
- In total 1881 images split in 1281/300/300 as train/validation/test

Feature Vector Approach – Reusing Trained Model

- Geometrical features (i.e. walls) are invariant to text removal
- Detect Rooms, and Find their types using walls.

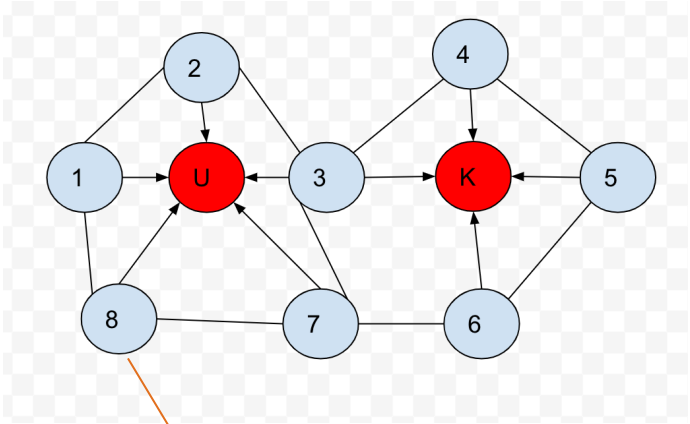
Feature Vector Approach – Building a Graph





Feature Vector Approach – Features

- What Set of nodes makes a room.
- Wall Features:
  - Geometrical
  - Relational
- Room Features:
  - Aggregation/summary of walls features.
  - Or better use Graph Convolutions

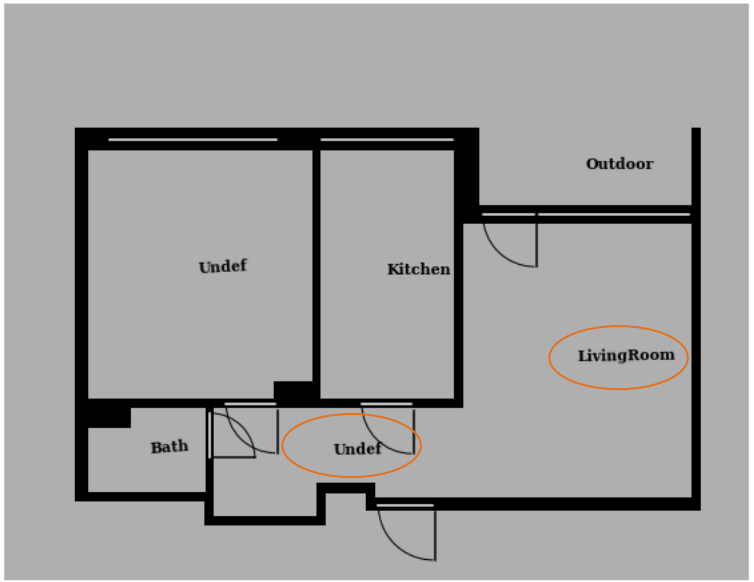


Wall features
height
width
rooms
walls

Feature Vector Approach - Stop

Why we Stopped:

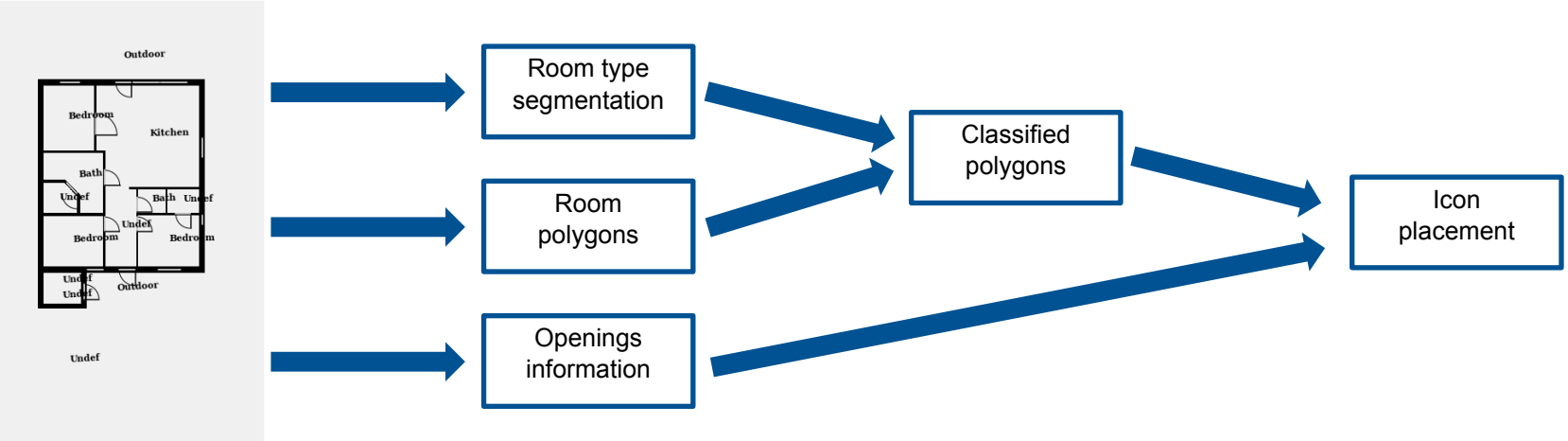
- Rooms are not only cycles
- More support for segmentation than for GCNs



Agenda

- Introduction & Objectives
- Related work
- Dataset
- "Feature Vector" Approach
- Solution
  - Segmentation Network
  - Room Proposal Extraction
  - Algorithm for placing icons
- Results
- Conclusion & limitations

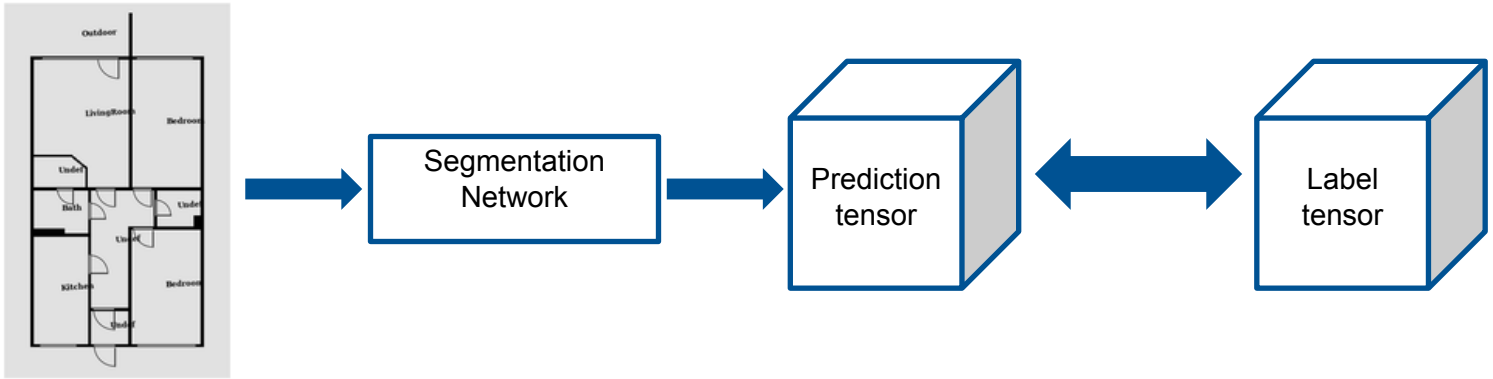
Solution


Outdoor
Bedroom
Kitchen
Bath
Undef
Bath
Undef
Undef
Bedroom
Bedroom
Undef
Undef
Outdoor
Undef
Room type segmentation
Room polygons
Openings information
Classified polygons
Icon placement

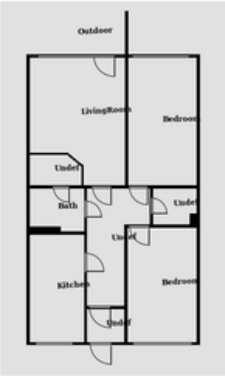
Segmentation network details

- U-Net architecture
- ResNet as a backbone
- 25 millions of parameters
- Dice loss

Segmentation network

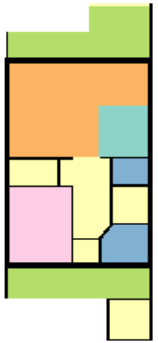


Room type segmentation



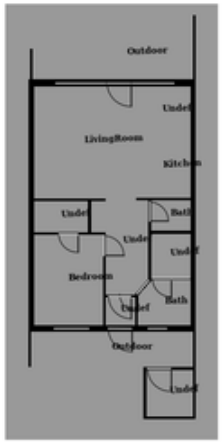
Input image

Predicted image



Label image

Openings segmentation

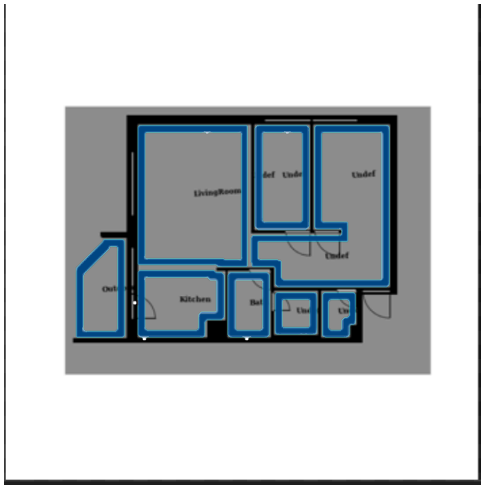


Input image

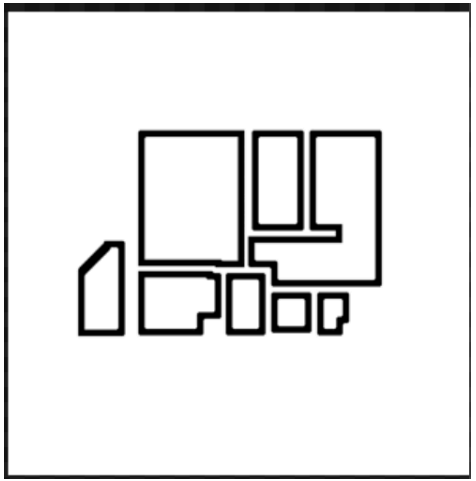
Predicted image

Label image

Room Proposals – New Formulation



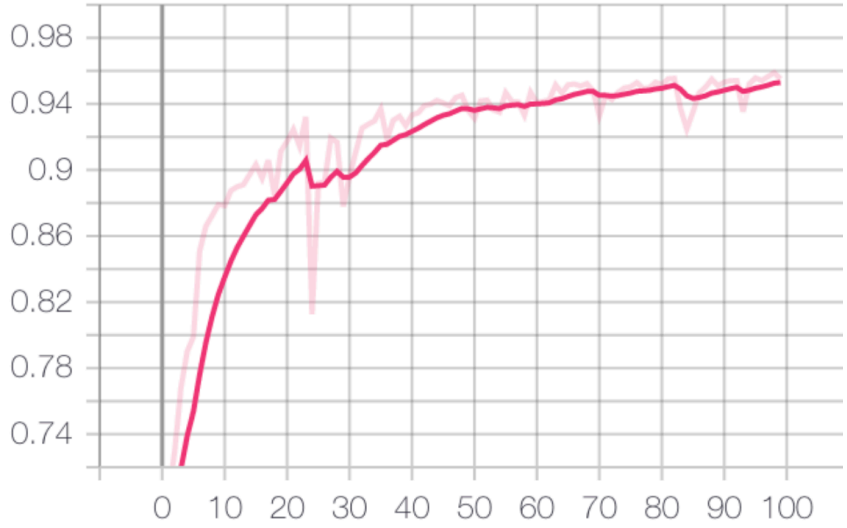
Input



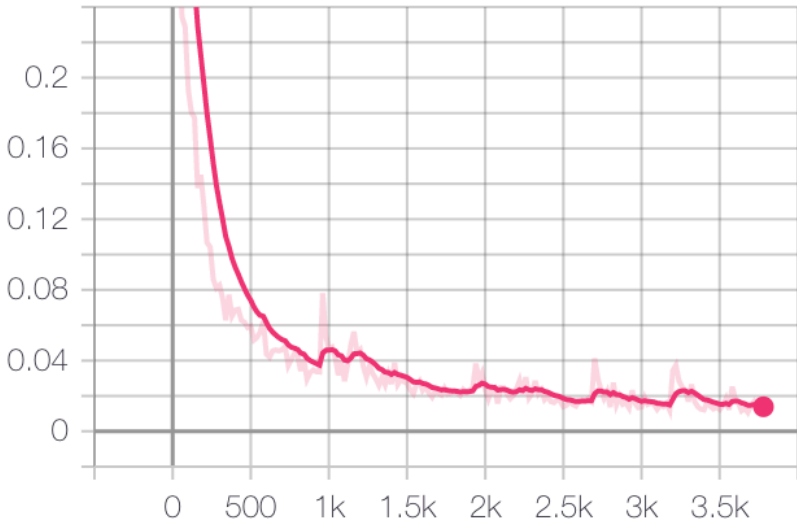
Proposals

Room Proposals - Training

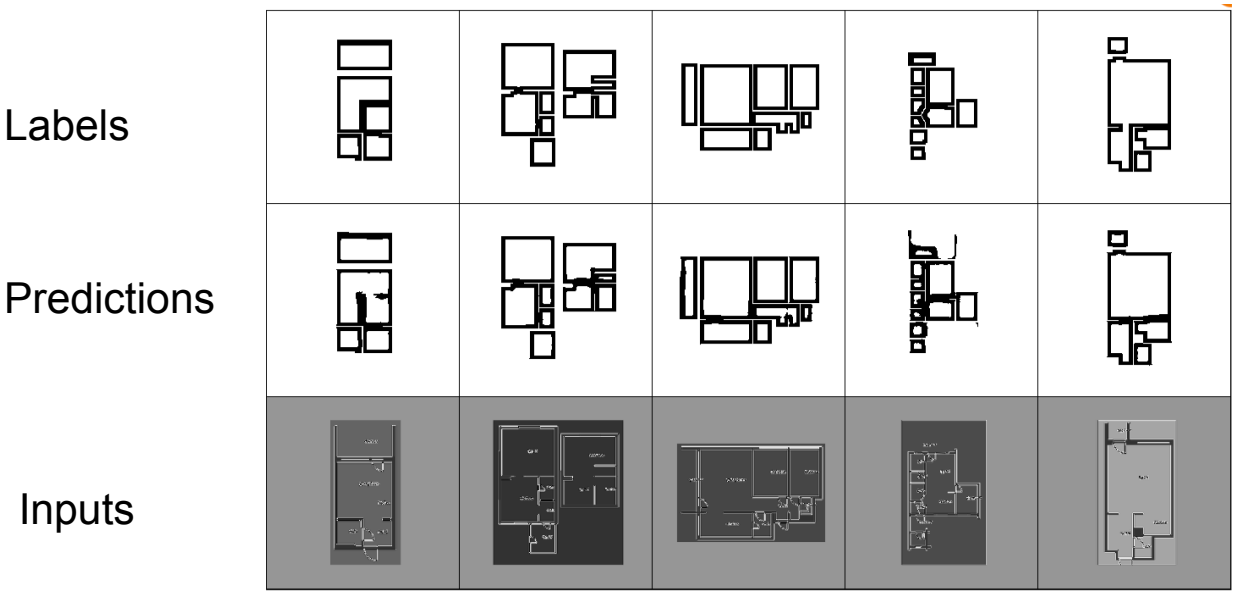
F1-score
tag: Val/F1-score



Loss
tag: Train/Loss



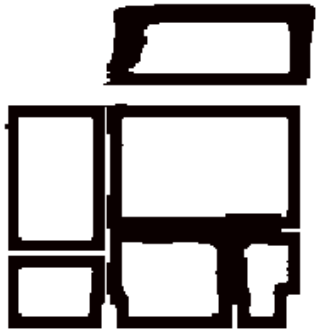
Room Proposals – Qualitative Results



Room Proposal Qualitative Results

Predictions Fusion - Input

input



Room Proposal

Room Type

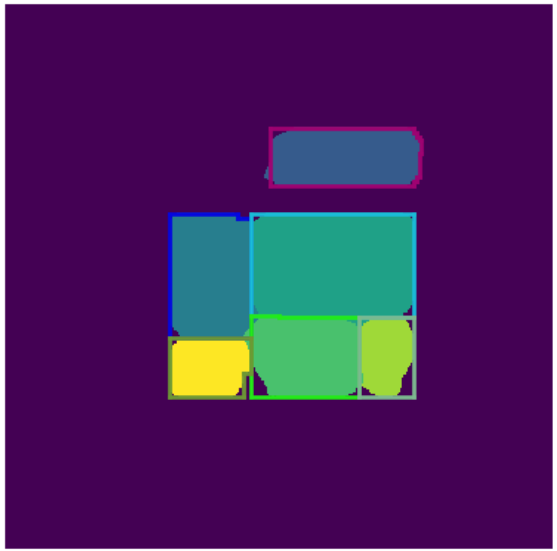
Predictions Fusion - Voting and Inpaint

Align

Vote

Inpaint

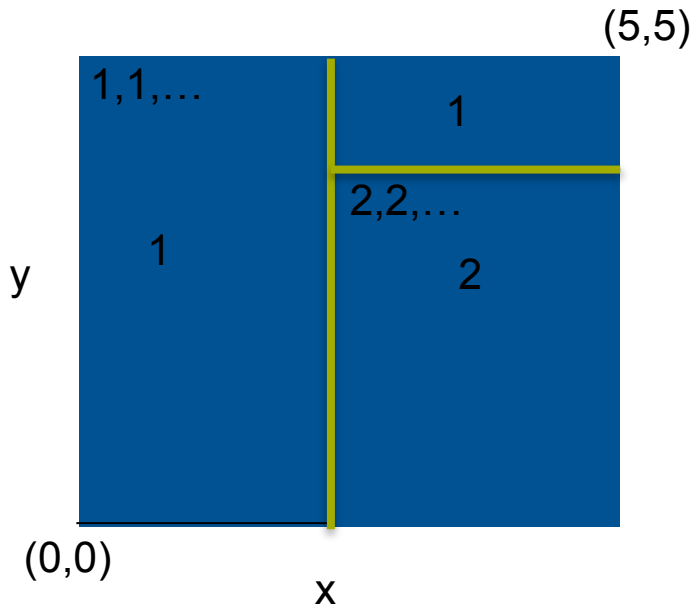
Predictions Fusion - Extract Polygons

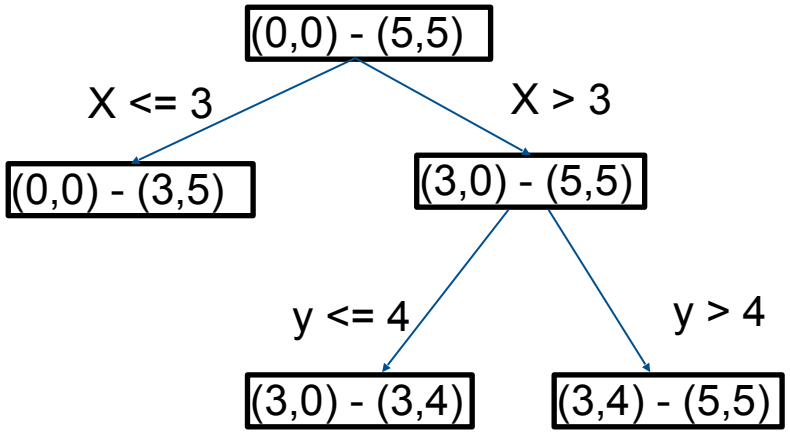


Extract Polygons

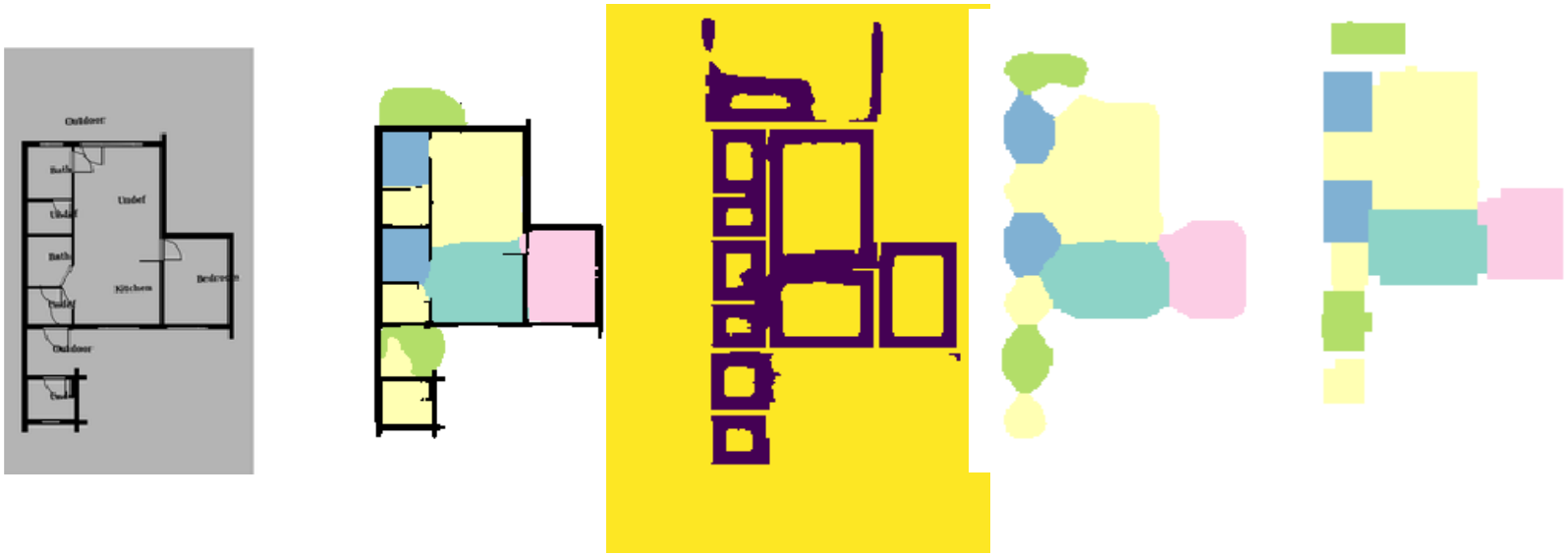
Polygons types

Predictions Fusion - Extract Polygon cont.





Predictions Fusion - Example

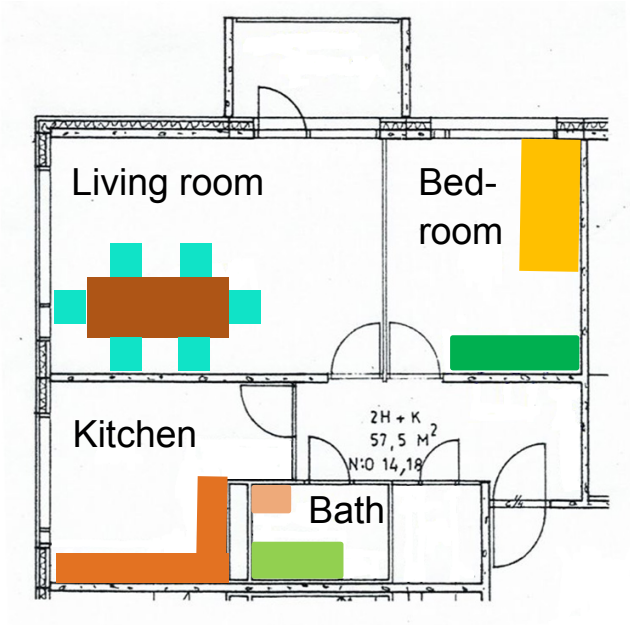


Algorithm for placing icons

The task of placing icons is

rule-based:

- "don´t place an icon in front of a door"
- "place a bath tube in a bath"
- …

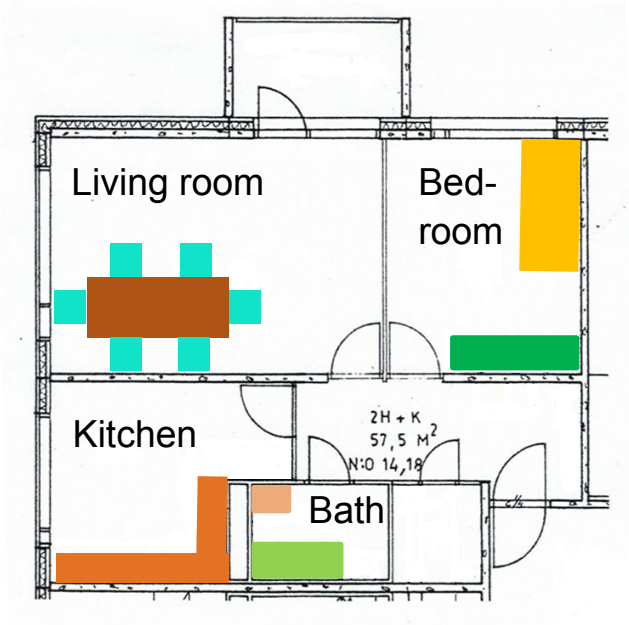


Algorithm for placing icons

The task of placing icons is

rule-based:
- "don´t place an icon in front of a door"
- "place a bath tube in a bath"
- …

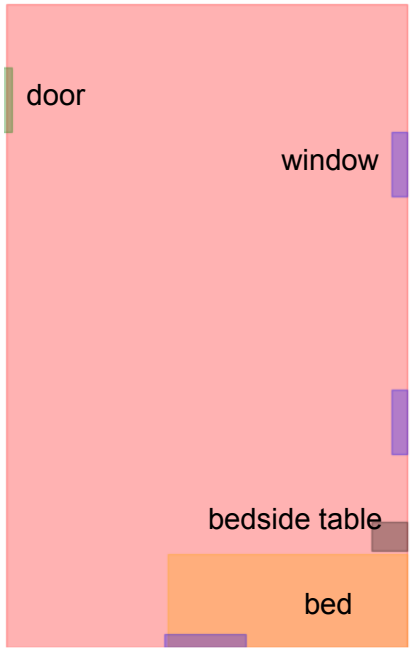
"creative" (inconclusive):
- There is not one optimal solution
- Multiple constellations are possible
- "Best choice" can depend on "taste" of a person



Algorithm for placing icons

Information used for placing icons:

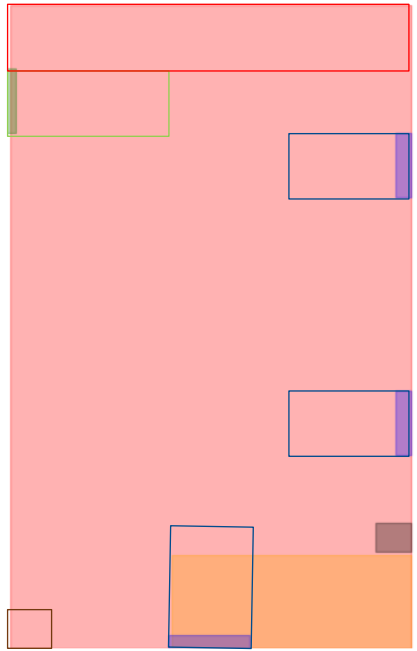
- Geometry of a room
- Type of a room
- Location of windows and doors



Algorithm for placing icons

Information used for placing icons:

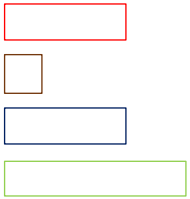
- Distance from wall
- If pixel in corner
- Dist. From window
- Dist. from door



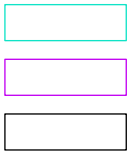
Algorithm for placing icons

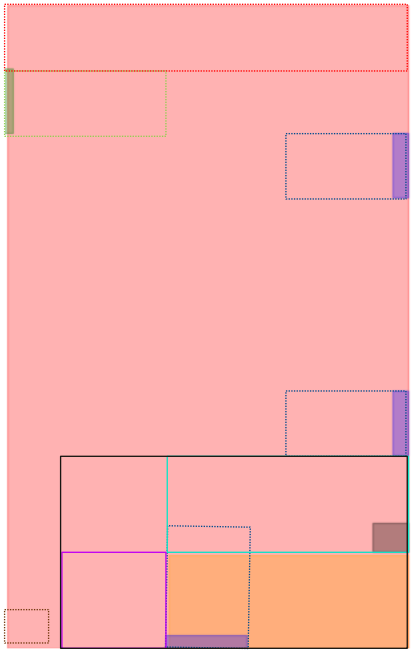
Information used for placing icons:

- Distance from wall
- If pixel in corner
- Dist. from window
- Dist. from door



- Dist. from icon – long side
- Dist. from icon – short side
- eucl. Dist. from icon edge

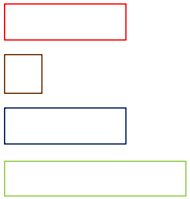




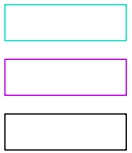
Algorithm for placing icons

Information used for placing icons:

- Distance from wall
- If pixel in corner
- Dist. from window
- Dist. from door

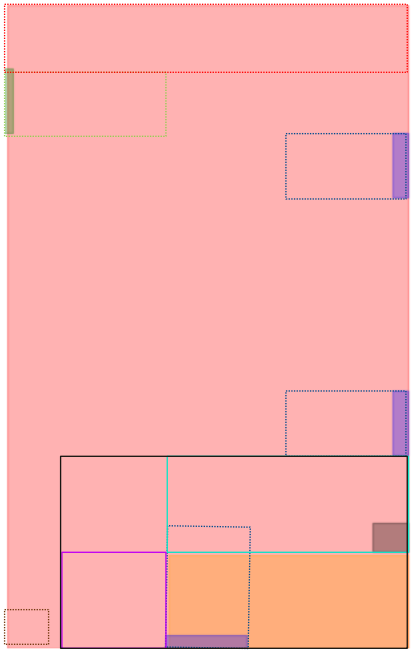


- Dist. from icon – long side
- Dist. from icon – short side
- eucl. Dist. from icon edge



One distance vector for every pixel

$$\begin{pmatrix} d_{wall} \\ d_{corner} \\ d_{window} \\ d_{door} \\ d_{icon1\text{-}s} \\ d_{icon1\text{-}l} \\ d_{icon1\text{-}eu} \\ d_{icon2\text{-}s} \\ d_{icon2\text{-}l} \\ d_{icon2\text{-}eu} \\ \vdots \end{pmatrix}$$



Algorithm for placing icons

Rules for placing icons:

Implemented through a weighting vector

- Weights can be positive (minimize dist.)
- Or negative (maximize dist.)

$$\begin{Bmatrix} W_{wall} \\ W_{corner} \\ W_{window} \\ W_{door} \\ W_{icon1\text{-}s} \\ W_{icon1\text{-}l} \\ W_{icon1\text{-}eu} \\ W_{icon2\text{-}s} \\ W_{icon2\text{-}l} \\ W_{icon2\text{-}eu} \\ \vdots \end{Bmatrix}$$

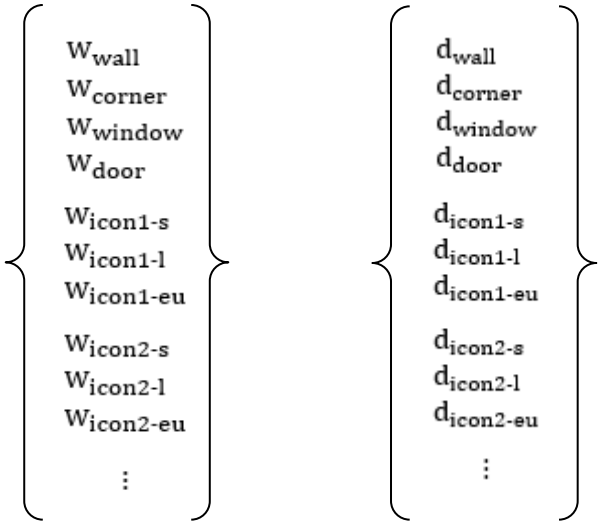
weighting vector

Algorithm for placing icons

Rules for placing icons:

Implemented through a weighting vector

- Weights can be positive (minimize dist.)
- Or negative (maximize dist.)



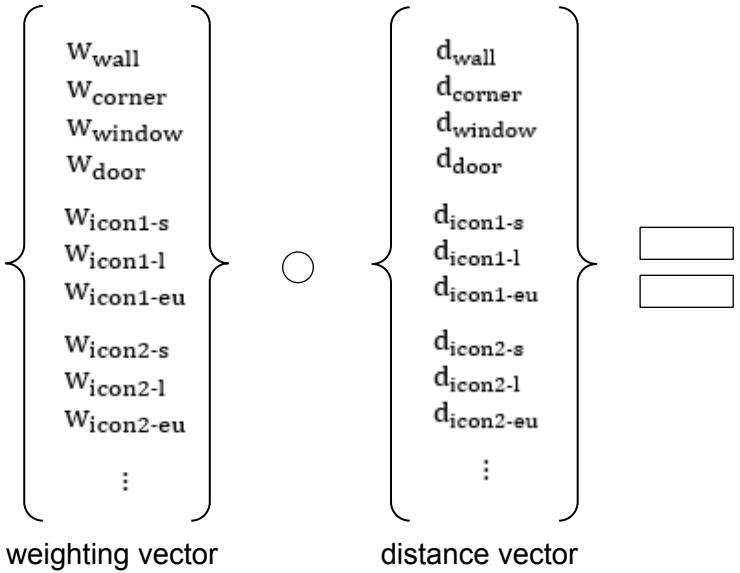
weighting vector distance vector

Algorithm for placing icons

Rules for placing icons:

Implemented through a weighting vector

- Weights can be positive (minimize dist.)
- Or negative (maximize dist.)



Value for each pixel

Algorithm for placing icons

Rules for placing icons:

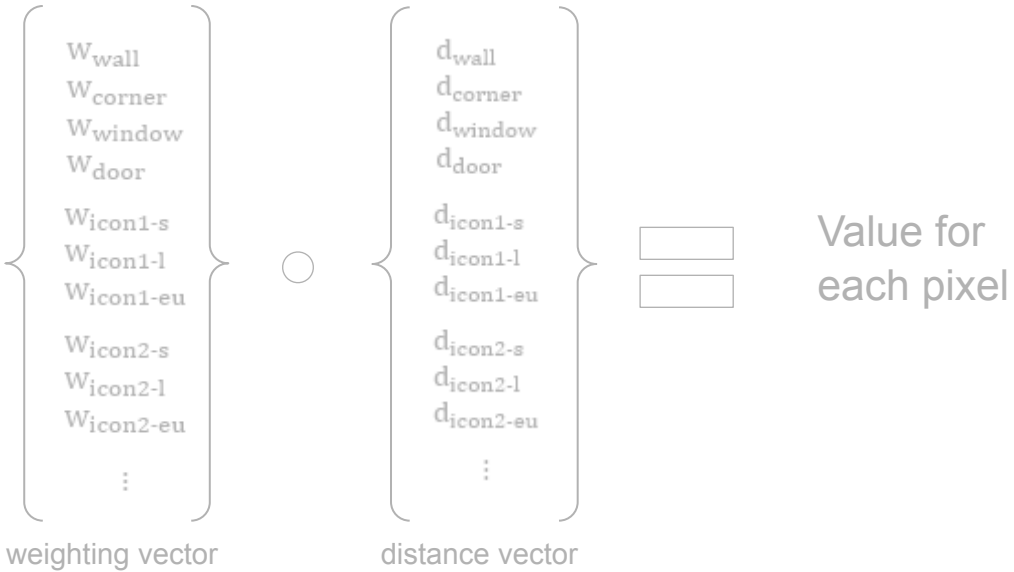
Implemented through a weighting vector

- Weights can be positive (minimize dist.)
- Or negative (maximize dist.)

- Additional conditions e.g.:

- Placing "dummies" between icons allows for more complex constellations

$$\text{If } d_{\text{wall}} == 20\text{: value} += 5$$

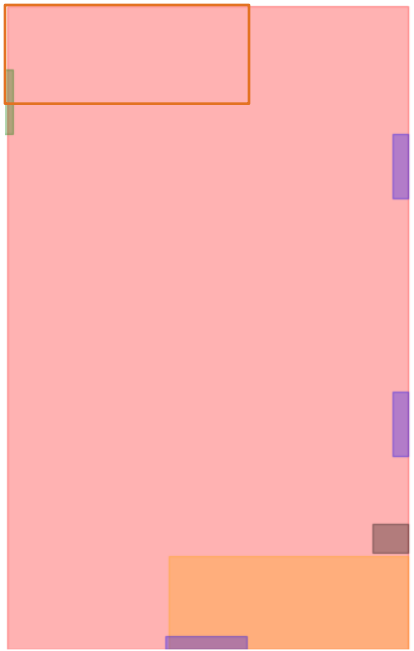


Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot

- An area value is calculated for every spot

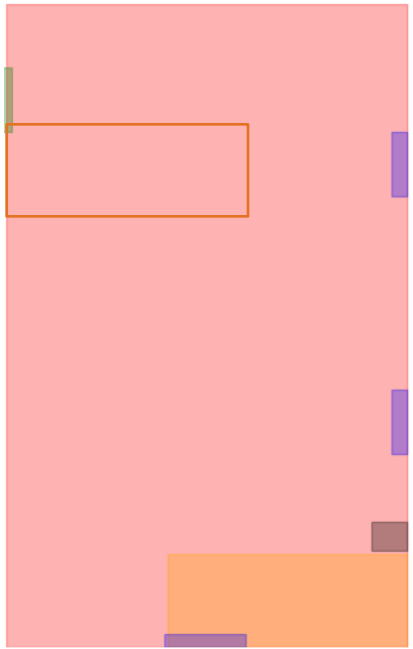


Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot

- An area value is calculated for every spot

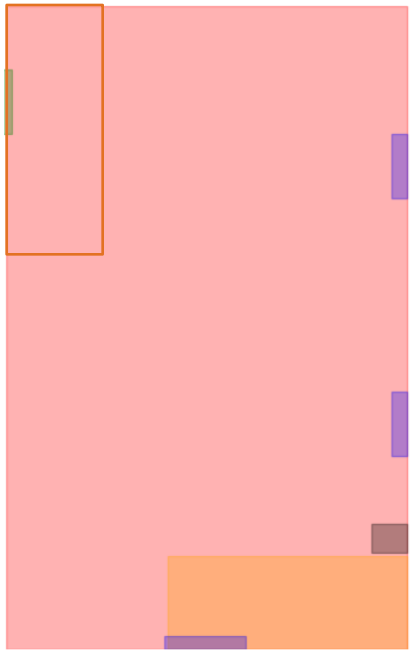


Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot

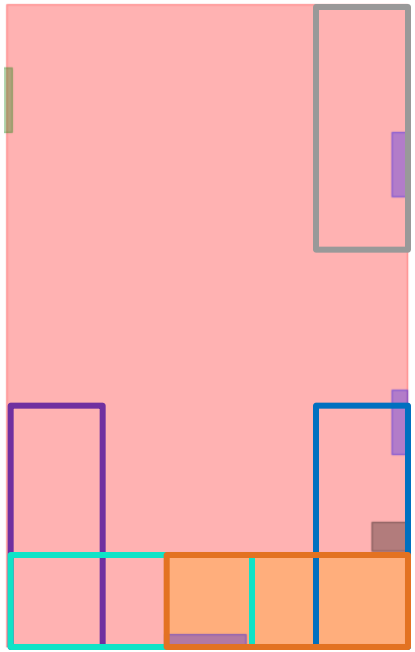
- An area value is calculated for every spot
- Repeated for icon being rotated by 90 degrees



Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot
- An area value is calculated for every spot
- Repeated for icon being rotated by 90 degrees

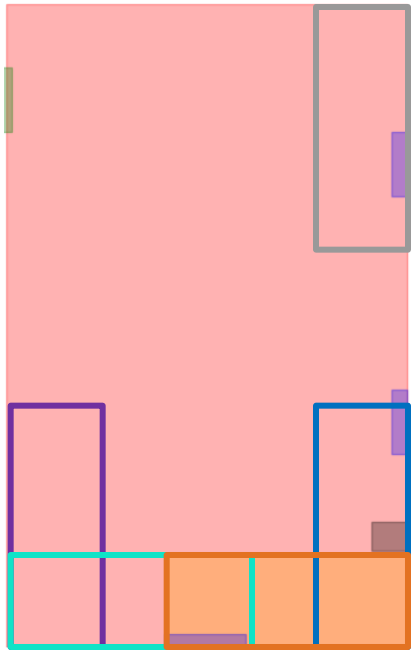


Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot
- An area value is calculated for every spot
- Repeated for icon being rotated by 90 degrees

Random factor for higher variability
- Small random value for every area to randomly pick one of the optimal spots

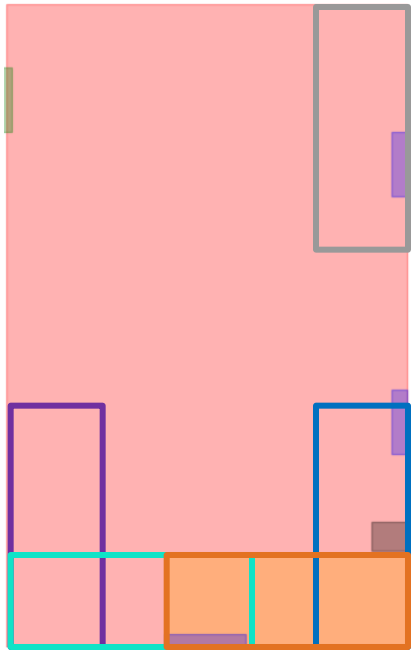


Algorithm for placing icons

Searching for best icon spots:

Grid search over every possible icon spot
- An area value is calculated for every spot
- Repeated for icon being rotated by 90 degrees

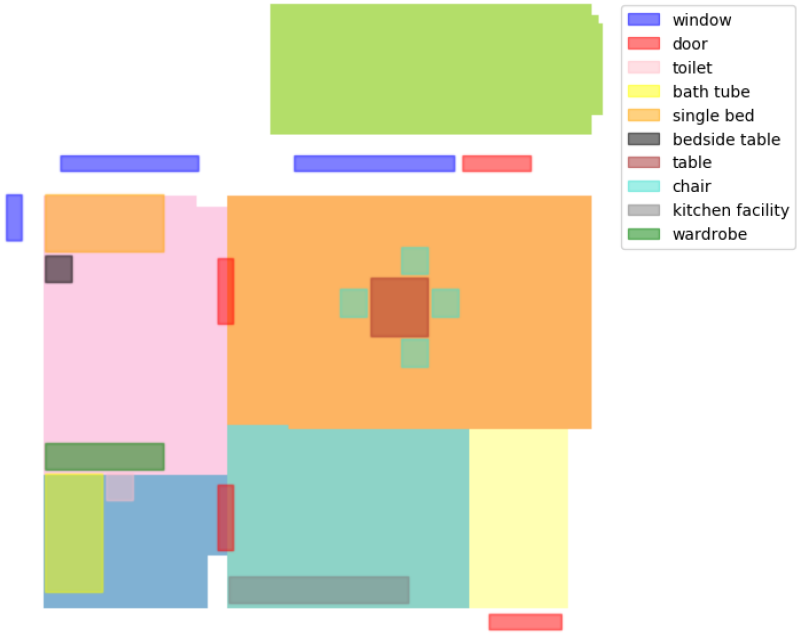
Random factor for higher variability
- Small random value for every area to randomly pick one of the optimal spots
- Order in which icons are placed is randomly, taking rules into account (e.g. place chair after table)



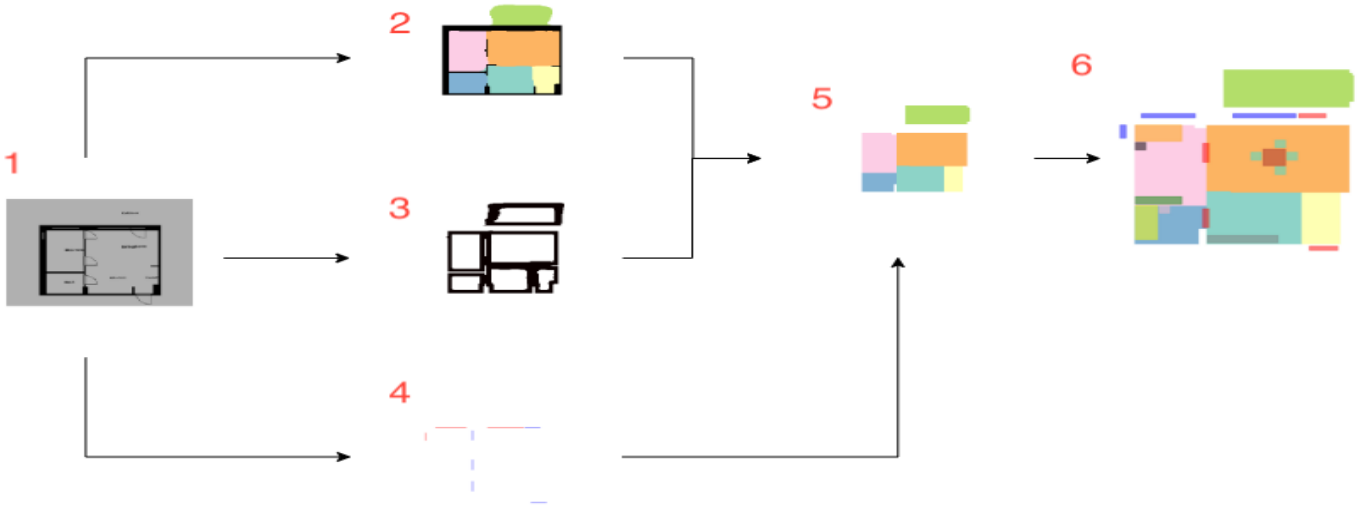
Algorithm for placing icons

"Learning of rules":

- Rules were first set due to common sense
- Rules were fine-tuned according to feedback
- Outlook: learn user preferences in a loop of human-machine-interactions



Results



Conclusion & Limitations

Initial goals were met

Limitations:

- Room segmentation can´t tolerate text rotated by 90 degrees
- Room proposal heavily depends on good proposals segmentation
- Icon placing faces problems when dealing with complex room shapes

Backup

Related work – early approaches

Computer-based processing of floorplan images is well researched in pattern regocnition

Early approaches mainly rely on low-level image processing heuristics:

- Separating textual data from graphical
- Detecting and grouping lines
- Overcoming gaps through polygonal approximation or edge-linking

To increase performance, recent works applied deep convolutional neural networks (CNNs)

Related work – Liu et al. 2017

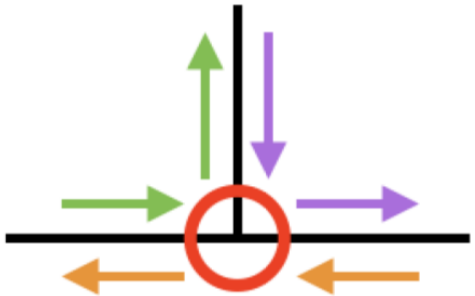
Merging geometric and semantic information applying predefined constraints

5 different constraints

- e.g. loop constraint
  - High level: exterior boundary of a room must form a closed loop
  - Enforced locally: For each pair of walls the room type must be the same

Constraints are applied using Integer programming

Result: performance – especially for geometries – increases significantly



Related work – Liu et al.

Performance

Method	Junction		Opening		Icon		Room	
	acc	recall	acc	recall	acc	recall	acc	recall
[13]	70.7	**95.1**	67.9	91.4	22.3	77.4	80.9	78.5
[13] + IP	94.7	91.7	91.9	90.2	84.0	74.6	84.5	88.4

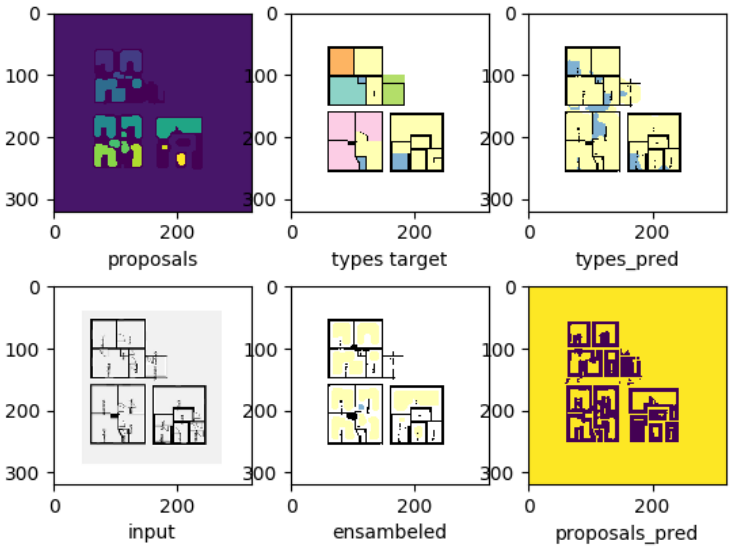
Related work – Kalervo et al.

Performance

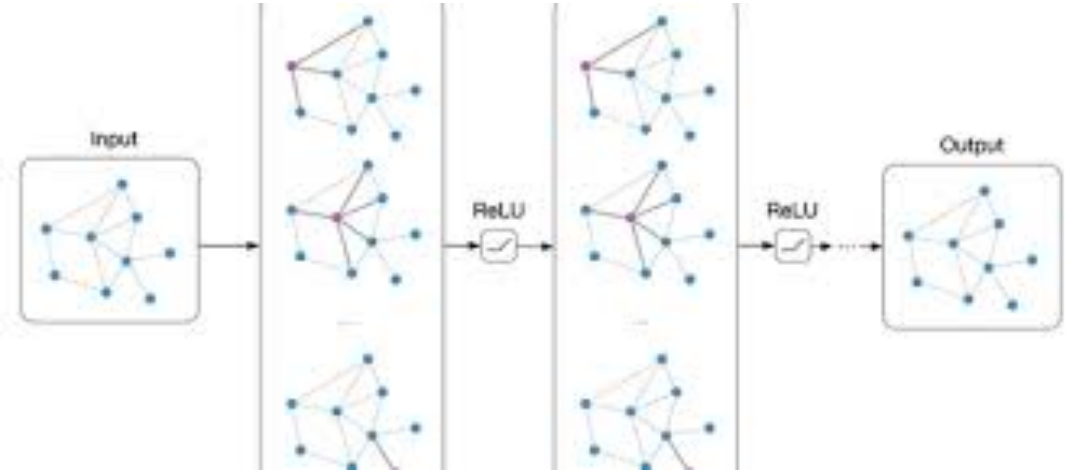
Method	Junction		Opening		Icon		Room	
	acc	recall	acc	recall	acc	recall	acc	recall
Ours	82.4	92.0	82.3	93.3	34.6	**88.3**	90.0	87.6
Ours (TTA)	90.2	91.9	89.6	**93.9**	46.1	88.0	91.5	88.0
Ours + IP	94.1	89.6	93.2	92.6	92.9	87.7	91.7	**90.8**
Ours (TTA) + IP	**95.0**	89.7	**94.5**	92.9	**93.6**	87.3	**92.2**	90.2

	Overall Acc		Mean Acc		Mean IoU	
	val	test	val	test	val	test
Rooms	84.5	82.7	72.3	69.8	61.0	57.5
Roomp	79.0	77.3	64.2	61.6	52.4	49.3
Icons	97.8	97.6	62.8	61.5	56.5	55.7
Iconsp	97.0	96.7	94.8	45.3	43.7	41.6

Rotated Input



Graph Convolutional Networks (GCNs)



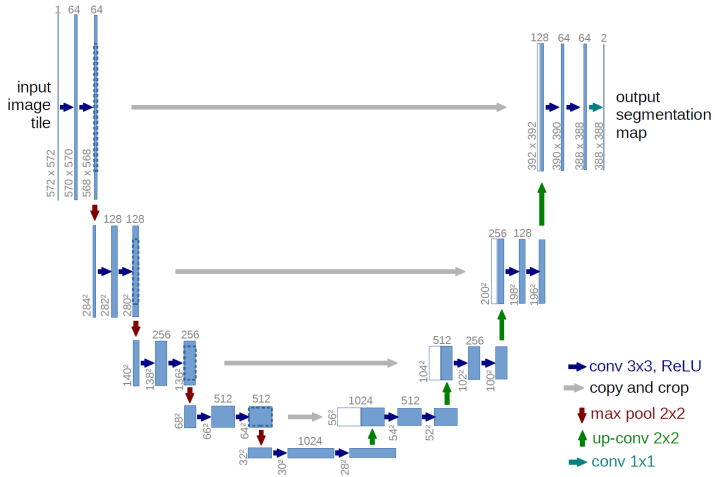
(image from https://datawarrior.wordpress.com/2018/08/08/graph-convolutional-neural-network-part-i/)

F1-loss

$$F_1 = \left(\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} \right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

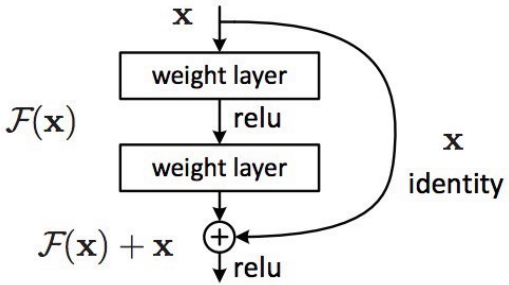
Voting

U-Net



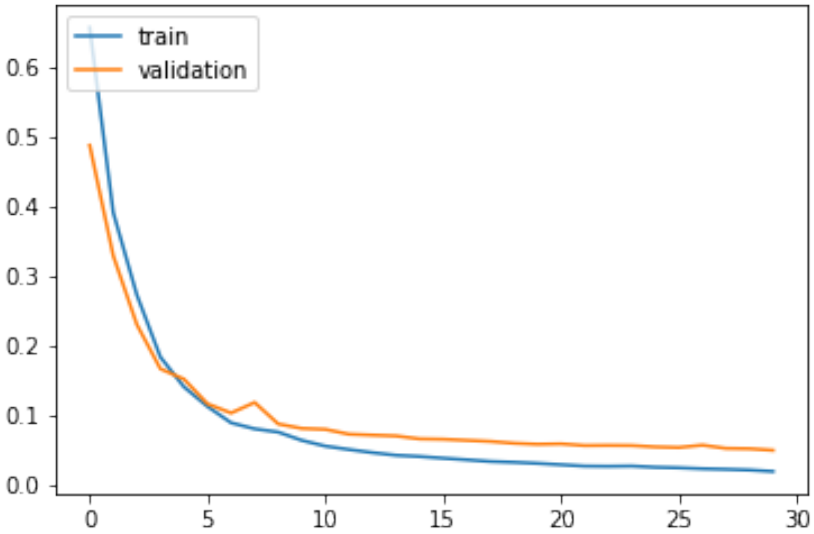
Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

ResNet



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Dice Loss



Dice loss