Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Computer, Communication and Information Sciences

Extraction of geometry information from floor plan images with deep learning

Leo Kärkkäinen

Master's Thesis
2021

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Leo Kärkkäinen

Title of thesis Extraction of geometry information from floor plan images with deep learning

Programme Computer, Communication and Information Sciences

Major Computer Science

Thesis supervisor Prof. Alexander Ilin

Thesis advisor(s) Prof. Alexander Ilin

Collaborative partner Aalto University, Termotohtori Oy

Date 07.07.2021 **Number of pages** 47 **Language** English

Abstract

Transformation of raster and scanned floorplan images into room geometry was investigated using CNN and U-Net methods that were also evaluated against the traditional image processing methods employed in prior work found in the literature review.

A novel CNN design with only 9 million parameters is proposed and trained to 90 % prediction accuracy on label maps of exterior and interior space, walls, windows and doors, where the existing state-of-the-art implementation also reviewed is 93 % accurate on the same task but uses 17 million parameters, limiting its use to rather small size or low resolution images. Both neural network models were found to be fairly flexible in processing different styles of floorplans with fairly minor mistakes, that is a huge benefit over the traditional methods evaluated. My network uses multiplication for combining its large area and local high resolution pathways in a way that requires both to be activated for an output to be produced. Pre-trained PyTorch networks are publicly available and they can be easily pretrained for detection of further symbols.

A novel algorithm is presented for converting those label maps into polygonal exterior wall geometry where the coordinates of any doors and windows are also marked. The new algorithm can handle angled and curved walls where prior work on the subject has been restricted to 90 degree corners. The algorithm needs a label map with wall detection performed by either of the neural networks evaluated in this work.

Building's heat loss calculation, as needed in energy certification and engineering applications, is presented as a practical use case for the extracted geometry. The resulting calculation is created promptly, avoiding manual engineering in measuring and entering each element from the original drawing.

The geometry information may also be used for creation of CAD or 3D models that have notable commercial use e.g. in real estate sales and in engineering work.

Keywords Floorplan images, Convolutional neural networks, U-Net, CubiCasa5k

Aalto-yliopisto
Aalto-universitetet
Aalto University

Tekijä Leo Kärkkäinen

Työn nimi Geometriatiedon tunnistaminen rakennuspiirustuksista syväoppimisella

Koulutusohjelma Computer, Communication and Information Sciences

Pääaine Tietotekniikka

Vastuuopettaja/valvoja Prof. Alexander Ilin

Työn ohjaaja(t) Prof. Alexander Ilin

Yhteistyötaho Aalto-yliopisto, Termotohtori Oy

Päivämäärä 07.07.2021 **Sivumäärä** 47 **Kieli** englanti

Tiivistelmä

Rasteroitujen ja skannattujen rakennuspiirustusten vektorointia tutkittiin CNN ja U-Net -neuroverkkoihin perustuvilla menetelmillä, joita verrattiin myös kirjallisuuskatsauksessa löydettyihin perinteistä kuvankäsittelyä käyttäviin malleihin.

Työssä esittämäni uusi CNN-malli sisältää vain 9 miljoonaa parametria ja se pääsee 90 % tarkkuuteen ulko- ja sisätilojen, seinien, ikkunoiden ja ovien segmentoinnissa rakennuspiirustuskuvista. Paras aiempi toteutus pääsee 93 % tarkkuuteen, mutta vaatii 17 miljoonaa parametria ja rajoittuu raskautensa vuoksi vain varsin pienikokoisten taikka matalaresoluutioisten kuvien käsittelyyn. Molemmat neuroverkkomallit kykenevät huomattavan joustavaan eri tyyppisten kuvien analyysiin suhteellisen vähäisin virhein, mikä on selkeä ero suhteessa perinteisiin menetelmiin. Oma verkkoni käyttää kertolaskua erillisten laajan skaalan ja paikallisen korkearesoluutioisen käsittelypolun yhdistämiseen, siten että molempien tulee olla aktivoituja, jotta verkko tuottaisi tunnistuksen tietyssä kohdassa. Molemmat verkot ovat saatavilla esikoulutettuina PyTorchille, siten että niitä voi helposti lisäkouluttaa uusien symbolien löytämiseksi.

Esitän uuden algoritmin geometrian muodostamiseen em. rakennetyyppikartasta siten, että ulkoseinät kuvataan polygonina, jolle ovet ja ikkunat on merkitty. Uusi algoritmi käsittelee myös muussa kuin suorassa kulmassa olevat seinät sekä kaarevat seinät, joita ei ole voinut aiemmilla menetelmillä käsitellä. Algoritmi edellyttää seinien tunnistusta, johon voidaan käyttää joko tässä työssä kehitettyä neuroverkkoa tai aiempaa toteutusta.

Rakennuksen tehontarvelaskenta, jota tarvitaan energiasertifikaatteihin ja suunnittelukäyttöön, esitetään käytännöllisenä sovelluksena piirustuksista puretulle geometriatiedolle. Tuloksena saatava laskelma syntyy nopeasti ja sillä välitetään suunnittelijan työ, jossa jouduttaisiin alkuperäisestä piirustuksesta mittaamaan ja merkitsemään kyseiset elementit.

Geometriatietoa voidaan hyödyntää myös CAD-kuvien tai 3d-mallien tuottamiseen, joille on huomattavaa kaupallista kysyntää mm. asuntokaupassa ja teknisissä suunnittelutöissä.

Avainsanat Rakennuspiirustukset, konvoluutioverkot, U-Net, CubiCasa5k

Contents

Acknowledgements

Professor Alexander Ilin's guidance has been valuable in the neural network research that this work is based on. Termotohtori Oy has provided a corpus of HVAC blueprints for evaluation. This research has been funded by Aalto University and Termotohtori Oy.

I wish to thank all parties involved.

Otaniemi, 7 July 2021
Leo Kärkkäinen

Abbreviations and Acronyms

AI	Artificial Intelligence
AR	Augmented Reality
CAD	Computer-Assisted Design
CNN	Convolutional Neural Network
DPI	Dots Per Inch, the number of pixels per 25.4 mm
FC network/layer	Fully Connected, not Convolutional!
FCN	Fully Convolutional Network
HVAC	Heating, Venting, and Air-Conditioning
OCR	Optical Character Recognition, text recognition
RAM	Random-Access Memory, computer memory
ReLU	Rectified Linear Unit, activation function
SVG	Scalable Vector Graphics, vector image format
U value	Heat loss metric, W K^(-1) m^(-2)

1 Introduction

Architectural and engineering blueprints describe buildings' dimensions and technical details that are important not only in construction but also throughout the lifetime of a building. The need for blueprints was recognised early on, and these paper documents are often archived also for older buildings. Nowadays scanned or rasterized images are stored in electronic archives hosted by municipalities.

The availability of drawings alone is not enough unless meaningful information may be extracted from them. There are a wide range of applications from real estate sales to engineering where either automatic searches need to be performed on a corpus of data, or specific dimensions and specifications need to be extracted on a particular building.

Automatic extraction of such information is the topic of this work, and for this purpose I review the history of prior research and current neural network models. Different methods of raster to geometry conversion are attempted, with a building's heating requirement calculation being shown as a practical application. The novel work includes a CNN network and a custom algorithm for polygonal geometry extraction, where the algorithm is shown to produce state of the art results when combined with a heavyweight pre-trained neural network.

For a large part of urbanization and city building in the 20th Century construction blue-prints were hand-drawn on paper. Although in the past decades such design has moved prominently into CAD drawings, often no machine-readable representation is available because all too often only the paper printouts are properly archived, while the digital files might remain only with the companies that were involved in the architectural and building design.

The prominent software packages used in Finland are Autocad, usually along with MagiCAD or other add-on package, and Cadmatic CADS. This allows for great variation even for drawing a simple rectangular room, as depending on packages and de-signer workflow a rectangular room might be drawn either as a single 3d volume, by four distinct wall objects, or even manually with thin lines representing inner and outer wall boundaries. The blueprint material examined in this research showed plenty of mistakes such as disjoint lines and symbols drawn only partially. The diversity of different packages and workflows also leads to about as many different drawing styles as there are individual engineering agencies or employees doing the drawing.

While it is to be expected that improved CAD tools make it harder to create inconsistent drawings, and that archival requirements can guarantee future access to blueprints of buildings constructed now, the lack of machine-readable blueprints of old buildings remains a serious issue.

Specifically in HVAC design every building undergoes a basic adjustment roughly every 20 years to accommodate for changes in building technology

and renovation. Renewed heat loss calculations are still all too often performed manually using a paper blueprint and scale ruler to measure room and window dimensions, as well as other items relevant to the designer. This involves plenty of manual labour and data entry that should be avoided.

The purpose of this work is to build a system that could automate much of the work, converting paper blueprints or CAD 2d image rasters into machine-readable elements and coordinates. Automatic heat loss calculations are shown as a proof of concept, although conversion into CAD images would be one of the more interesting applications.

Due to the large amount of variation in drawing styles, both computer-made and hand-drawn, I believe the problem cannot practically be tackled by traditional computer vision methods, and that by necessity neural networks or other very flexible heuristic methods are required as well.

2 Methods

2.1 Literature review

Blueprint information extraction has been addressed by a number of studies over the past decades.

Filipski & Flanderena (1992) describe GTX 5000, a commercial product originally brought to market in 1987, that could convert electrical logic schematics from scanned images into CAD format with less manual labour than would otherwise be required. Notably the system employed neural networks for character and symbol detection, making it a very early case of commercial application of convolutional image detection.

Early work in conversion of floor plans looked for time savings in converting crude hand-drawn sketches into CAD format. Aoki, et al. (1996) assumes drawings using thin and thick lines to mark interior and external walls. Line segments and closed regions are extracted from a pre-processed bitmap, and then combined in element identification phase that employs pattern matching. Lladós, et al. (1997) uses Hough transform (Dura & Hart, 1971) to pattern-match walls and other elements of interest which become nodes of a relation graph. This approach is proposed specifically for hand-drawn sketch floor plans, which must use only predetermined markings such as a hatching-pattern within walls, which also must be constrained to 90 degree angles.

Another approach to the problem is constructing a 3d point cloud by photography or LIDAR imaging of the building's interior spaces. Okorn, et al (2010) propose an optimised method of converting such point clouds into floor plans. They reduce the 3d points into a 2d density map and use Hough transform to detect line segments, which are further post processed to reduce noise and overlap.

Ahmed, et al. (2012) analyses a blueprint dataset using wall thickness to segment interior and external walls. Holes such as windows are then filled by connecting segments and a building boundary extracted and broken into disjoint rooms, where they find 89 % accuracy against the ground truth. A novel contribution of the group is to use OCR to extract room names from the drawing and using those to break down the found rooms into semantic rooms that correspond with the labels.

Ruwanthika, et al. (2017) attempts to build full 3d models of houses for use in AR applications. The part of the paper involving blueprints uses Harris corner detector to identify walls and corners from a blueprint. Their blueprints used hollow outer walls which would get detected as two separate walls, where the extra lines needed to be removed by the authors' custom algorithm.

Over the articles that could be found in this review, nearly all research seems to using traditional image processing algorithms and highly simplified blueprints or scans. In deed it would appear that the problem of tackling designs in real buildings seems very difficult and due to the long AI winters, with neural networks becoming really useful in image processing only in recent years, most have favoured the more traditional approaches.

After 2017 there has been a clear resurgence in neural network approaches.

Dodge, et. al (2017) uses Fully Convolutional Networks (FCN) to parse architectural floor plans in a manner similar to my aim. The paper states as is objective to handle simplified plans used in real-estate websites, stating that parsing geometric and semantic information from them couldn't be done via traditional means. Different sizes of FCNs are evaluated against a dataset also published alongside the paper, for finding wall pixels. The authors also evaluate traditional and statistical approaches, employing Google Vision API to remove text elements that might confuse such analysis. A comparison of methods is shown in figure 1. The authors show automatic 3d model creation as a possible application that should be of interest in real estate and interior designs.

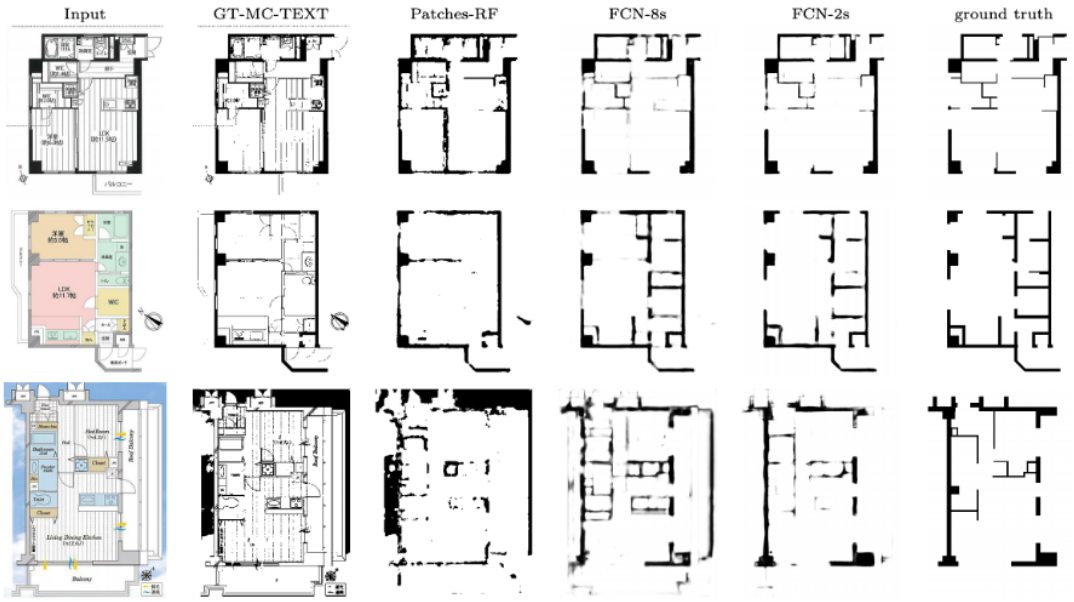


Figure 1: Evaluation of various methods for wall segmentation of floor plans, comparing FCNs with a combination method involving thresholding, morphological closing and text removal (GT-MC-TEXT), and with a method involving statistical algorithms (Patches-RF). Dodge, et. al (2017).

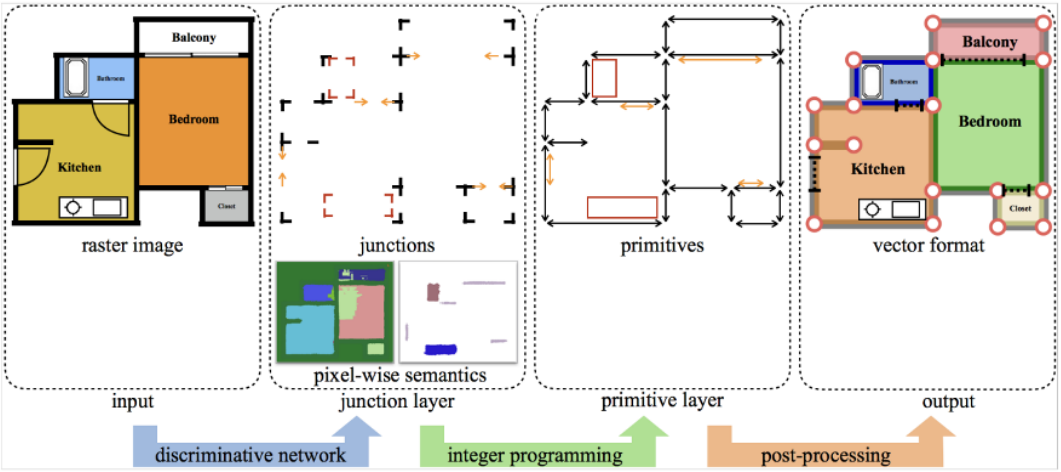


Figure 2: Liu, et. al. (2017).

Liu, et al. (2017) takes another approach, not using a fully convolutional network but instead a ResNet-based CNN to both detect wall junctions and other points of interest, icons, walls and areas by function (e.g. living room against bedroom). These heatmaps are post-processed to obtain a vectorized representation, although the post-processing is limited to Manhattan geometry (no angled walls), with a process shown in figure 2. The geometry representation is handled as a linear programming problem, where each room is required to have a closed loop around the junction points forming its walls.

Kalervo, et al. (2019) continues the work of Liu, et. al. by retraining the model with larger labelled archive CubiCasa5k, which they also publish. All code has been rewritten in Python, with a post-processing algorithm also limited to Manhattan geometry (although it is not documented whether there are algorithmic changes).

The dataset contains 5000 colourful and monochrome raster scans which have been manually recreated and labelled in SVG format. A tool is included for converting these label SVGs into image tensors which are then used for training and evaluating the model.

The model is a ResNet-152 first modified for human pose estimation (Bulat & Tzimiropoulos 2016), then further modified for floor plan analysis. It employs a multi-task CNN for predicting room and icon heatmaps that may be further cleaned and post-processed into polygons. The post processing algorithm is limited to strictly horizontal and vertical edges, and as such cannot support angled walls despise them being well predicted in the heatmaps. The network also finds labels such as OH (Finnish abbreviation for a living room) for room functional segmentation as seen in **Figure 3**.

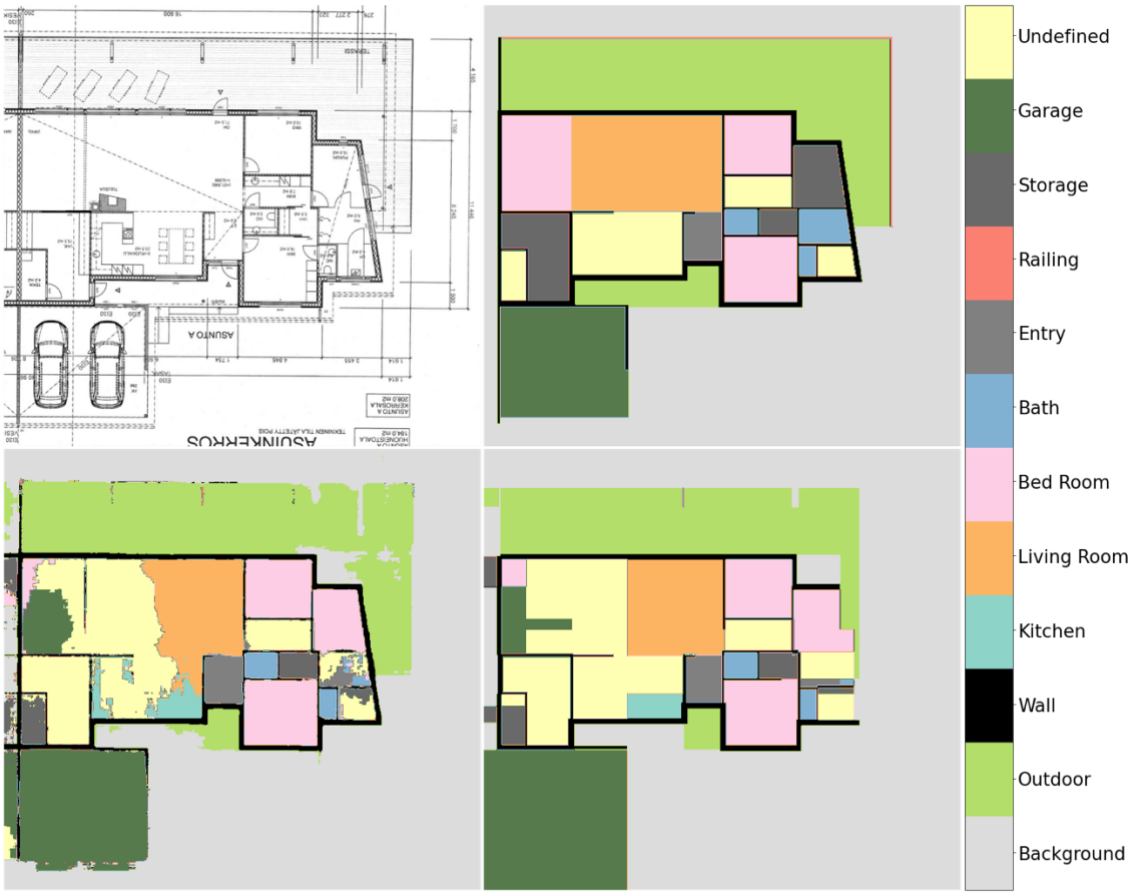


Figure 3: Kalervo, et al. (2019) CubiCasa5k. Raster input (top left), ground truth (top right), room prediction (bottom left) and post-processed output (bottom right). Only the raster input is used for predictions.

In this work I tackle the floor plan raster to vector problem using various approaches all built on neural networks in PyTorch. The emphasis is on a more complicated setting of HVAC blueprints that contain a high amount of detailed markings and noise that would make analysing them by the traditional approaches quite impossible.

In this work I evaluate the CubiCasa5k CNN and dataset in more detail for the application in HVAC design, where heat losses through exterior walls need to be calculated.

2.2 Problem definition

In the corpus of blueprints there are nearly as many different drawing styles as there are buildings. Walls are drawn usually in solid black or hollow (figure 4) or filled with other patterns, or in combinations of different layers denoting insulation, concrete and/or façade. Other semantic elements exhibit similar variation, meaning that the model will have to be able to adapt to styles and combinations of styles it has not been trained with. Drawing

style is generally consistent within a building but markings and mistakes may cause minor deviation.

Many of the images used are HVAC drawings rather than architectural blueprints, meaning that radiators and pipelines along with their types and measurements are marked on the drawings. Some of the drawings also contain furniture and other non-structural elements that need to be ignored. Since many of the images are scans of old paper blueprints, there are minor geometric deviations and noise not present in pixel-accurate CAD exports of new designs.

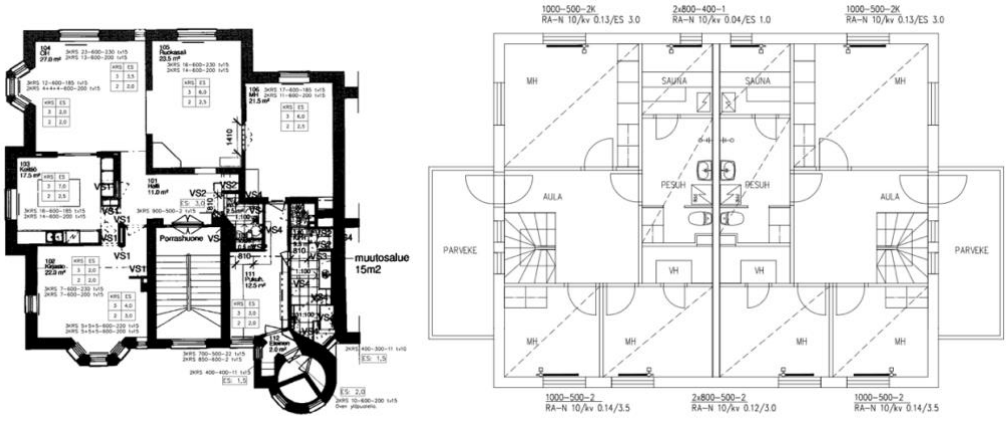


Figure 4: Floor plans in the most prominent drawing styles, with solid walls and with hollow walls.

Given a small piece of a floor plan showing only two parallel lines it can be concluded that there is a wall but it may be impossible to tell whether it is an interior or an exterior wall. Some drawings use different fill patterns for the two, and it may be possible to discern by wall thickness. Even if a wall is known to be exterior, it is usually impossible to tell which side of the wall faces outside, or if the whole wall is outdoors (e.g. balcony walls).

Drawings are assumed to be in scale, which is either known, or can be found by measuring the number of pixels that correspond to a dimension marking on the drawing. Images may be scaled in pre-processing to correspond to the same pixel-to-meter scale used in neural network training, to ease element recognition.

It is assumed that all drawing elements of interest may be identified locally but that a larger context covering several rooms is necessary to determine even which areas are indoor spaces, and to build a representation of the building's outer shell and of the rooms inside, such that all spaces may be represented as closed polylines consisting of walls, windows, doors and openings. Buildings that enclose exterior areas within them can be represented alike with an inner polygon for the outdoor area but since such buildings are rare, this part is ignored. Similarly, multiple disconnected

building parts may be processed as separate buildings, each with its own external polygon.

In the corpus of floor plan images used most buildings have various angled and curved walls and entire wings even if most corners are perpendicular. This is noted as an additional obstacle in geometry analysis and representation.

A practical problem of estimating heating requirement (heat loss) of a building is explored. For this, areas of exterior surfaces, most notably those of walls, windows and roof need to be extracted. Roof area is usually equal to floor area, which can be measured from floor plans. The horizontal dimensions of walls and windows can be found on the blueprint, while their heights need to be otherwise known, along with the U values of the constructs used.

Blueprints contain textual data such as labelling of room use as well as technical data that can be easily extracted by standard OCR packages that have also evolved with the recent AI boom. Some background work inspected here makes use of such technology but this part is purposefully left out of my research, which concentrates on the semantic and geometry extraction of the buildings' shape.

2.3 Introduction to classification, heatmaps and segmentation

Neural networks can be used in image analysis for object detection in a variety of ways. We examine different methods, which are best illustrated by a simplified example.

1. An image classifier can tell whether there is a cat or not.
2. A convolutional heatmap tells which pixels look like fur.
3. Semantic segmentation tells which pixels make up the cat.

Methods 1 and 3 look at the whole source image, but while the classifier only returns a single value, segmentation produces such values for each pixel. Method 2 falls in between the two, as it essentially does classification on each pixel using a region of the source image surrounding that pixel, and it turns out that the heatmap can be used very similarly to those of semantic segmentation networks such as FCN and U-Net.

2.4 Convolutional networks

Convolutional networks are the basis for all this, and although they were invented and used as early as 1980's, only in recent years have they seen rapid development with the coincidence of faster hardware, better training algorithms and the invention of new deep neural network structures.

Convolution is place-invariant filter that is highly useful in neural network processing due to its low number of coefficients and calculation, compared to fully connected networks.

The 2d convolution runs over the spatial dimensions of an input image, employing a kernel of coefficients to transform a small neighbourhood of pixels into output values. In image processing the kernel is typically a 3d matrix which is initialized with random values and then trained using Adam Optimizer or other gradient descent methods. The kernel is not dependent on source image size but the output image dimensions are.

In neural networks each linear transformation is followed by an elementwise activation function (typically ReLU) which provides the necessary non-linearity for operation of multi-layer networks, as otherwise they would become equivalent of just a single linear transformation. This is such a common practice that the activation is often not even shown in diagrams.

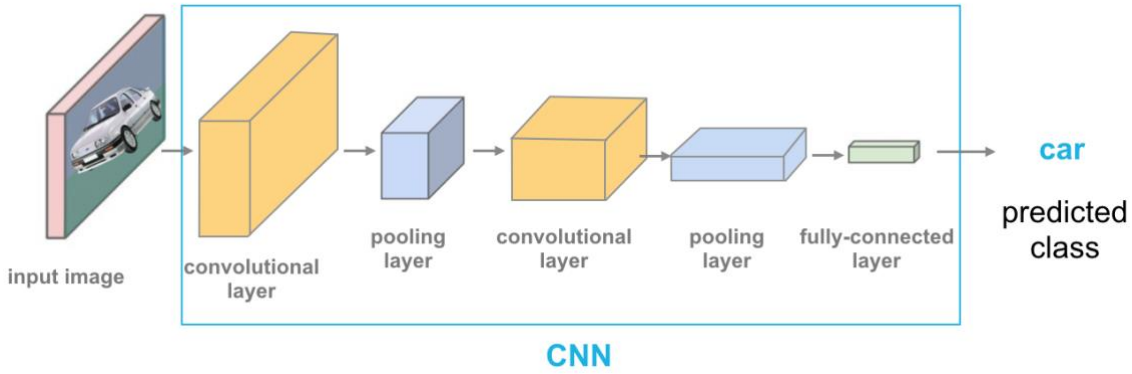


Figure 5: Camocho (2018). The principle of a CNN image classifier.

A CNN image classifier is a deep neural network structure where a series of layers is used to reduce an image into one or more scalar values denoting classes.

The simple classifier in figure 5 uses a series of convolutional and pooling layers which are used to further reduce the input size by only using the strongest values, discarding others. In this process the source image is transformed from its three RGB channels to a large number of intermediate channels in the convolutional and fully connected layers while reduced in spatial dimensions in pooling. The final fully connected layer reduces all those channels to just a single scalar output denoting the prediction of whether the image is of a car or not.

The fully connected layer combines all source image pixels together and produces a single output value for each class the network is trained on (often also one for unknown class). Unlike convolution and pooling, this part

depends on input image size, thus such CNN classifiers can only be used with the specific image size they were trained on.

Often the source images are pre-processed by cropping and scaling, avoiding the size requirement and allowing for object localization and detection applications.

A CNN classifier could tell whether the input image is a floor plan and that is not of much use in this work, but it turns out that CNNs form the basis of all neuronal methods that may be used to identify locations of rooms and other elements of interest.

2.5 Heatmap creation

Heatmap creation may be best understood via image classification. Where image classifiers reduce an image into a single value denoting whether the image contains something of interest or not, segmentation produces such value for each pixel.

It is functionally equivalent to replacing the fully connected layers of a CNN classifier with convolutional layers on images reduced to 1x1 pixel size in prior layers. In PyTorch this avoids the manual reshaping of dimensions and also allows heatmap generation when the source image is in fact larger such that the output is no longer just a single pixel.

A CNN that doesn't reduce the resolution all the way and that doesn't include the final fully connected layers may be used to produce a heatmap where each position give predictions of classes of specific objects appearing at that location in the original image. For this I build a custom design which is easy to train, and evaluate its properties in finding walls, windows and doors.

Further analysis is done using the much larger pretrained network employed in Kalervo, et. al (2019), named CubiCasa5k. This model processes far more symbols and interestingly can also identify room spaces in addition to walls and other symbols directly marked on drawings.

Semantic segmentation by U-Net is an entirely different approach in that it processes the entire floor plan globally, making it able to make the distinction between interior and exterior space even when a local region of a drawing doesn't include that information.

Label maps where one integer value on each pixel gives the most likely classification of a pixel may be obtained from heatmaps e.g. by choosing on each pixel the heatmap layer with the highest value and storing the layer number on a corresponding labelmap (argmax over heatmap layers). Another approach is to choose layers where a certain threshold (such as 0.5) is exceeded, in an order of precedence, which may be better when the heatmaps are heterogenous and not directly comparable, or if some elements are deemed more crucial to detect even in the presence of other stronger signals.

2.6 Semantic segmentation

2.6.1 Fully Convolutional Network (FCN)

Where a simple convolutional network can be used to classify each pixel based on its neighbourhood without affecting image resolution, the range used for input will be limited in spatial dimensions. Such a network can pick up local patterns much alike the blind men who inspected an elephant by feeling and touching parts of the creature. Each mistook it for different things. A more global approach is useful, and that is accomplished by downscaling as in image classification, essentially first coming to a conclusion of whether there is an elephant, and then working back to details about its trunk, feet, body and tail.

For instance, a classification network trained to find cat pictures should produce a number closer to 1 than 0 when used on an image of a kitten, but a segmentation network would show the silhouette of a cat in bright colour, as shown in figure 6.

Figure 6: FCN segmentation overlaid in green over the original photo of a kitten.

Fully convolutional network (FCN) for semantic segmentation was proposed in Long, et. al (2014), using a series of convolutional layers and downscaling to reduce the image's spatial resolution, while increasing the number of channels. The resulting very low resolution image in effect performs classification on each of its pixels using a large portion or all of the original image for input.

Noh, et al. (2015) augmented the FCN by adding deconvolution layers to upscale back to original resolution, which may be seen as a form of AI super

scaling. The resulting network has an hourglass shape shown in figure 7, with a low spatial resolution chokepoint in the middle.

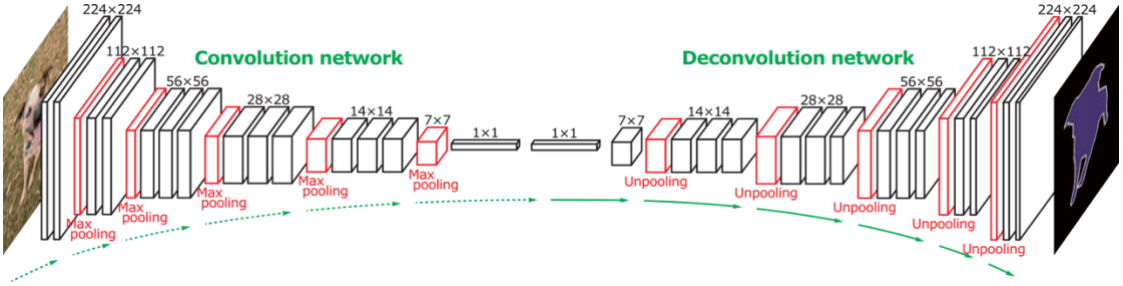


Figure 7: Noh, et al. (2015). FCN with deconvolution to obtain a high resolution semantic map.

2.6.2 U-Net

The low resolution intermediate stage in FCNs loses much of the original detail so that the resulting area is a blob that does not precisely follow the original pixels. The U-Net is an improved architecture of FCNs, proposed in Ronneberger, et. al (2015), addressing this very limitation by adding bypass connections that skip the lowest resolution choke-point.

Ronneberger's research problem involved microscopic images of living cells, where inner and outer regions of cells needed to be found. Just as with building floor plans, it is often impossible to determine inner and outer regions of a cell by looking at the local neighbourhood of pixels, where the cellular wall may angle outwards alike a recess of a building.

The paper suggests that the new design needs fewer training images, while offering improved output precision and requiring less computational resources for evaluating images. While the bypass connections certainly help in training and output resolution, I remain sceptical of any improvements on the computational resources on training given that the amount of convolution needed changes very little and is easily overcome by any tweaks in network structure and the bypass stages in any case increase memory requirements over the original feed-forward FCN.

Just like the original FCN, U-Net uses downscaled layers to the aid long-reaching or global decision making of which regions are inside and which are outside the membranes or walls delimiting them.

The original FCN's hourglass shape is in U-Net twisted into an U shape, with multiple levels providing bypass connections to corresponding deconvolution layers as illustrated in figure 8.

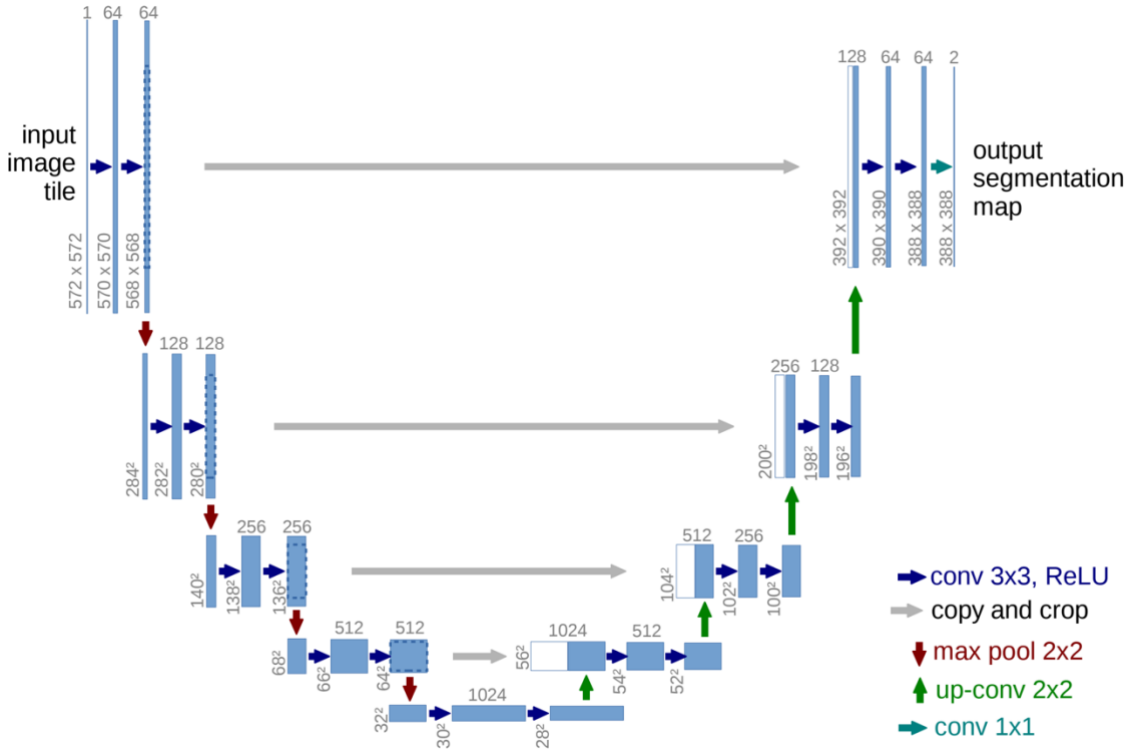


Figure 8: U-net schematic (Ronneberger et al. 2015), showing a five level design with two output channels. Notice how the dimensions shrink in each convolution so that the final output only covers a portion of the input image. The network gets its name from the U-shape of layers as seen in this diagram.

Each level halves the resolution and doubles the number of channels such that at five levels deep any image is reduced to 1/32th of its original dimensions. A modified version of the network, as suggested to me by prof. Ilin, uses a fully connected layer at the bottom to remove any remaining spatial distancing for a truly global approach.

First the input image is downscaled through all the levels, each using convolutional block of three convolutions followed by MaxPool2d reduction, with each level's image stored separately for bypass. The deepest level acts as a decision making mechanism for choosing what is deemed inside of a structure. The latter half expands that decision to the full spatial detail of the original, employing also bypass images from the corresponding levels. The image from lower level is bilinearly upscaled and its channels concatenated to that of the level's stored image, then fed through another convolutional block, all the way up, until the original resolution is reached.

The convolutional blocks do not scale the image, but without internal padding the outermost areas get cropped in each convolution within each block, resulting a smaller output labelmap than what the input image was, even though the resolution halving and doubling operations do even out.

If padding were added to each convolution, this could be avoided, keeping the original image size, but then the model is no longer oblivious of the coordinates. It in effect sees the image border by all the zeroes supplied there as a padding. It could thus learn that the centre of the image contains indoor area, as in most training samples. By not using any padding inside the network it is forced to use the input pixels rather than their coordinates. In order to match the image and labelmap sizes, enough padding needs to be added on the input image (using the same colour as the blank space in the drawing) such that the labelmap becomes at least the size of the area of interest. In general it is impossible to match the output exactly with input dimensions, prompting for the labelmap to be cropped accordingly so that it matches the original image.

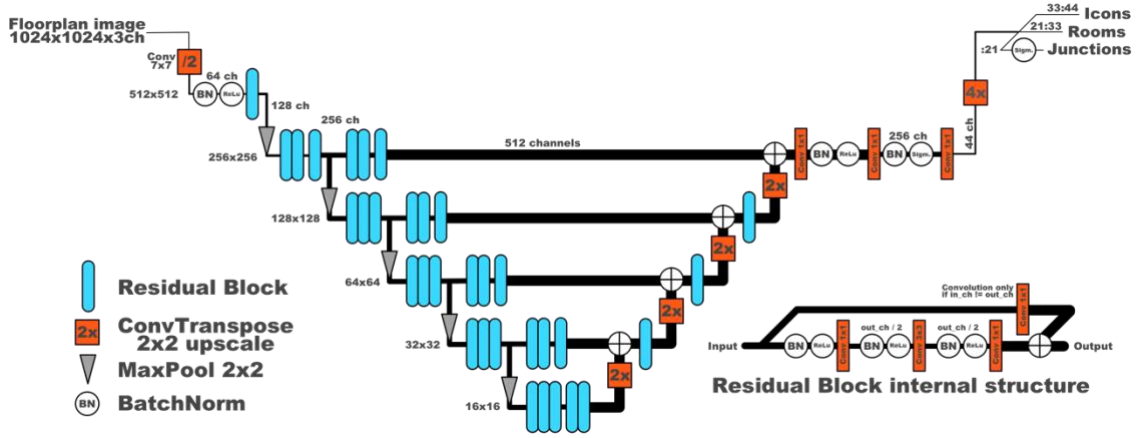
Another possible approach would be to use no padding of the image either but that would lead to a labelmap covering only a fairly small central area, making it infeasible for this project.

I found the method of padding to have a visually apparent effect during training. With padding done only on the drawing, the model first marks interior areas only nearby walls, and gradually expands to fill the entire building. Otherwise even completely blank images would show a halo marking the location where most training images had interior space.

The model was trained using algorithmically generated blueprint images with two-pixel lines for external walls and corresponding label images for interior space. Several network configurations were attempted up to six layers, which turned out both hard to train and very memory-hungry, while five layers seemed quite optimal with these 128x128 pixel images.

2.6.3 CubiCasa5k

Figure 9: The CubiCasa5k network is a variation of U-Net, augmented with ResNet blocks. The image is scaled down 4x prior to most expensive processing, and at the end scaled back to original resolution. All blocks and symbols represent processing steps (not data tensors), and the lines showing data flows increase in thickness with the channel count.



This network is an evolution based on the work of multiple research groups at Google and elsewhere, up to Aalto University. The residual stage of ResNet image classifier was combined with U-Net architecture for human pose estimation, producing pixel-level heatmaps. (Bulat & Tzimiropoulos 2016). Then the final deconvolution layer was replaced by three parallel deconvolution layers trained on floor plans (Liu, et. al 2017) for detection of junction points, rooms and icons. This pre-trained network was further modified by Kalervo, et. al (2019) replacing the final 1x1 convolution layer with their own to allow for different labelling used in their CubiCasa5k dataset, and applied further training to accommodate these changes, as well as to support for more diversity in drawing styles and the Finnish symbols which are prominent in their dataset.

The resulting network (figure 9) has 17 million trained parameters in 125 convolution layers and 102 batch normalisation layers, making it a heavy design.

The input image is downscaled to quarter resolution prior to any heavier processing and the output maps are upscaled back as the final stage. Further savings over the standard U-Net are found by restricting the number of channels to 128 or 256 in most convolutions, and by not reducing the image down to a single pixel. Padding is added at each layer such that the input image does not need to be padded and the output resolution always matches the input. Slight bilinear scaling is used after each upscale to make dimensions match when input image dimensions not divisible by 64 otherwise cause rounding errors.

Limiting channels and levels, and not requiring padding make a compromise between a simple convolutional network and the standard U-Net, allowing medium-sized drawings to be processed without running out of RAM and computational resources. The drawback is that the room segmentation has a limited range of sight, causing misclassification in very large rooms. Due to internal padding, the network uses knowledge of drawing's boundaries, which is apparent in Figure 3 with the next apartment on the left being classified as Unknown because it is situated between a wall and the drawing's border.

The network output is a heatmap with 44 channels representing different features.

- Junction points (wall corners and other junctions) are produced as the first 21 features, representing corners in I, L, T and X shapes, in all possible 90-degree rotations, as well as the endpoints of windows and other elements.
- Rooms (walls and area functions) are classified as a group of 12 features which denote different room functions, outdoor areas and the space taken by walls.
- Icons (incl. windows and doors) form another group of 11 features.

These groups have incompatible output levels and simply picking the maximum value among all channels of interest didn't produce useful results. Instead, each group has to be processed separately and a most likely type chosen within the group, and only the relevant data should be picked from each group. In particular, I took the wall regions and overlaid windows and doors on top of that for a labelmap similar to those that my own network produced.

The original work uses the 90 degree corner junctions for constructing geometry in its post-processing stage. Corners of other angles had no points marked in junction layers, so I opted to ignore the junction data and rather employ the rooms prediction for geometry.

The area function classifications done by this network work primarily using knowledge of the nearby markings (including Finnish labels such as "OH", olohuone, for living rooms). Downscaling the image down to 1/64th of original resolution for the lowest level, where five 3x3 convolutions are performed, allows a using a rather large neighbourhood of each output pixel for determining the room function. This works for medium-sized rooms but cannot accommodate larger halls where there are no labels or walls nearby.

The network is somewhat unreliable in classifying individual room functions such as kitchen and living room. This ambiguity in function is not in all cases a problem with the model itself because with open kitchen designs there is no clear real world boundary between the two and there are other ambiguities within the drawings as well.

For the application of heat loss calculation the exact function of a room is irrelevant, so the simpler classification into internal and external spaces

suffices. When relevant room classes are clumped into sets representing interior and exterior spaces, satisfactory results can be obtained. The room classes are assigned as follows:

Interior:	Kitchen, Living Room, Bed Room, Bath, Entry, Storage, Undefined
Exterior:	Background, Outdoor
Ambiguous:	Wall, Railing, Garage (can be interior or exterior)

While ideally we could determine interior and exterior space within walls and railings too, the lack of such data is circumvented in the geometry extraction discussed in Section 2.10. Garages should be manually assigned either as interior or exterior depending on whether they are warm, i.e. enclosed by exterior walls.

2.7 Design of the CNNFloor model

Looking at a small local area, a few centimetres across on a 50:1 scale printout of floor plan, allows identifying elements directly at that location. For instance, a wall, a window and a staircase may all locally consist of parallel lines and white space, but in different patterns such that they can be quite reliably be told apart by local inspection only.

However, if there is only a straight part of a wall visible within the area of sight, there is no way to tell if either side of the wall is exterior or internal to the building, a feat crucial for heat loss calculation. You need to look at the big picture.

Following this observation, I designed a convolutional network with two pathways, one for the local information and another for the big picture. The latter pathway sequentially downscales the input tensors, to obtain further reach. This, of course, is a common practice in CNN networks, where staying in the original resolution and using very large convolution kernels rarely does any good.

The high resolution pathway can learn to be very good at identifying different patterns that may be used to draw walls, whether that be two thin lines, a solid fill, a hatched pattern or little stones drawn to denote a concrete wall. Also, it can tell in its output precisely where the wall boundaries are located. Surely, it is possible to make a fairly good wall detector network by this pathway only, as long as enough 3x3, 5x5 or 7x7 convolution layers are added in series (odd pixel sizes to keep them centred on the input and moderately small to allow proper generalization in training).

The low resolution path makes use of some initial processing (the pre sequence) and keeps doing further processing as the data is reduced in spatial resolution down to 1/32th of original. A 7x7 convolution at this resolution in essence processes 224x224 pixels in the original resolution, and with all the

other convolutions included, a visual range of several hundred pixels is obtained. Presumably this pathway looks at where the building's boundaries go, although it may also construct a higher level vision of walls to help eliminating graphics that aren't true walls even if they locally seem so.

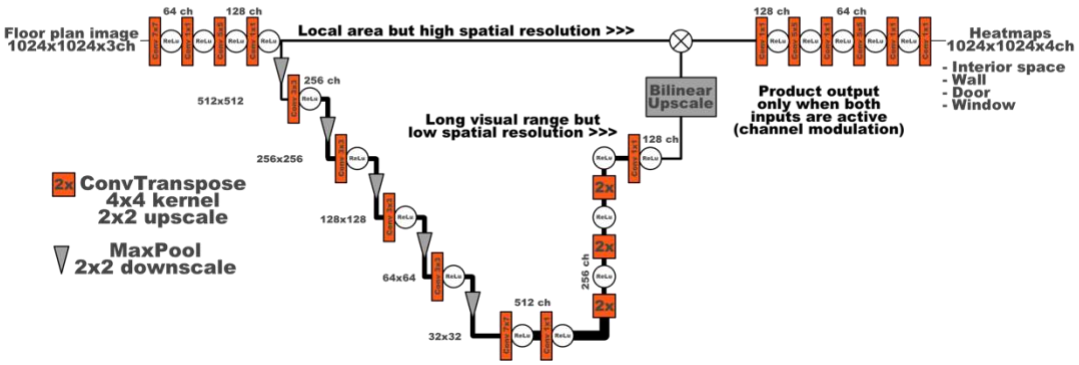
So, how do we combine the two pathways? The standard solutions are either concatenation or just simple addition. Concatenation temporarily doubles the channel count, which when done in full resolution is expensive, and probably because of this most networks prefer addition. Adding two unrelated inputs together may seem odd, but the network should during training find matching use for the corresponding channels so that the sum of two makes sense.

Autograd software and in particular the PyTorch framework nowadays used for most neural network implementations keep track of all operations done with data tensors in the "forward" direction such that the framework can go backward and calculate the gradients used for optimizing the network's parameters. Thus we are in no way restricted to the operations that others might be using.

The output should say "wall" only if the high resolution network sees the correct markings and if the low resolution pathway confirms that there actually seems to be a contiguous wall segment spanning some distance. Similarly, we expect interior space only where there is no wall at the immediate location. If both pathways end with ReLU activation (Sigmoid would work as well), the output of an "active" neuron is always positive, while negative values turn the activated output to plain zero. This suggests that multiplication is a very good combining operator. First of all, it is well handled by autograd. Secondly, since neither input can go negative, unexpected changes in direction with sign changes are avoided. Third, this accomplishes the goal of activating the output only when both inputs agree.

After such combination, a post sequence of final processing follows such that the data may be mixed and filtered properly, forming clean and solid output.

Figure 10: The CNNFloor network employs two pathways where one retains the original image resolution while the other that has a far longer visual range mainly due to the 7x7 convolution at the very bottom.



It turns out that the design shown in figure 10 works surprisingly well in practice, although one specific problem seems to occur. Doorways are usually marked by two parallel lines in the opening, and a circular arc or sometimes a triangle on the side where the door opens. All good and well, one pathway sees the arc and the other sees the lines between which it is supposed to output that it is a doorway. The only problem is that sometimes the two lines are missing, and then the network really struggles. Presumably the high resolution pathway says that there definitely cannot be a door here, so nothing is marked.

Since the CubiCasa5k dataset used in training already comes in a low resolution, keeping the processing mostly at full resolution seemed a good idea. If a floor plan comes in a much higher resolution, the network can be augmented by adding pre- and post-scaling to a lower resolution either by additional convolution layers on both ends or by simple image scaling algorithms. Without scaling the computational resources needed might exceed what is available, and more importantly elements on the drawing would otherwise appear too large and go undetected by the model. HVAC CAD exports in particular come in very high resolution and contain very thin lines where extra downscaling convolutional layers will probably do much better than image scaling that fades the lines away.

Training the model was performed on the CubiCasa5k training dataset (N=4200) in shuffled order and with random 90 degree rotations applied. The labels used in the dataset were processed to match with the four heatmaps chosen for this model (see section 2.6.3 for details).

The network has 9 million trained parameters, which makes it a fairly small convolutional network by today's standards.

2.8 Memory use considerations

Architectural floor prints often come in dimensions in the order of 10000 pixels across, and that became a practical problem because FCNs and U-Nets require a great amount of RAM quickly scaling up with the number of pixels in the image. Most blueprints contain very thin markings and fine detail that could not be scaled down prior to U-Net application without losing relevant information, and thus the model was hard to evaluate and even more so to train within the limited memory available on GPUs.

Large buildings could be broken into separate sections processed individually but that would further reduce the reliability of the segmentation, especially in cases where there are large halls, as in commercial buildings with open office spaces.

The image resolution in U-Net trials had to be kept to a small size at most a few hundred pixels across, to accommodate hardware requirements. The Titan X GPU employed had 12 GB of RAM and a model any larger would not run at all.

At the time of writing very expensive gaming GPUs have up to 24 gigabytes of video RAM, while professional Quadro RTX 8000 computing cards costing 7 200 euros up the memory size to 48 gigabytes, and even that would allow only a moderate increase in image dimensions or in model complexity.

The simple convolutional networks on the other hand do not require more parameters for larger input, and thus scale better with input image resolution, so that the 10000+ pixel HVAC plans can be processed without cropping or scaling.

The CubiCasa5k model falls in between, allowing dimensions up to around thousand pixels at most, when running on a GPU. I was able to obtain an Nvidia RTX 3090 card while finishing this work, and even with the doubled memory the largest images would need to be processed in cropped pieces.

Further, all models examined in this work can be evaluated on CPU within a reasonable time (in the order of seconds to a minute per image), making 64 gigabytes or more RAM practically available. Training remains infeasibly slow on CPU alone, so pre-trained networks are preferred except for the simplest designs.

The PyTorch toolkit used also couldn't release all resources as soon as they were no longer needed, and in many cases it is hard to write the code in a way that allows instant garbage collection of tensors that are no longer needed. Using in-place operations when possible, manually deleting tensor variables and frequently triggering garbage collection runs seems to help in keeping the memory use in check. Running in a Jupyter Notebook and leaving any tensor variables within the global context causes guaranteed memory leaks.

2.9 Heat loss of a building

A polygonal model of outer walls could be used to estimate the heating requirement of a building based on its outer wall, window and floor areas, at a given outdoor temperature.

The heat loss

$$\phi = \sum_s \Delta T_s \, U_s \, A_s \qquad (1)$$

is a sum of heat losses through all surfaces s, depending on the surface area A_s, the structural heat penetration U_(s) and the temperature difference. ΔT is the difference between indoor and outdoor temperatures. Using indoor temperature of 22 °C and outdoor temperature of -26 °C for all surfaces we get ΔT=48 K. Different exterior temperatures may occur in particular for floors that are against the ground rather than ventilated with exterior air but for simplicity this does not need to be concerned here. The chosen outdoor temperature of -26 °C gives the nominal heating requirement in Helsinki where extended periods colder than that seldom occur.

The floor area appears directly as an area on a drawing, and may be calculated when the scale of the image is known such that pixels can be converted into meters. The scale may be found either from measuring a known dimension, or it may be calculated if the scanning DPI and the drawing scale such as 1:50 are known.

Wall length and window and door widths can be similarly measured but their heights are not drawn onto floor plans and often the dimensions are given in text. Still, in many cases the only way to find these dimensions is to physically visit the site and use the measuring tape. It helps to know that most buildings have a floor height of 3 metres (2.8 m indoor wall height), and that's also the value I use in this work.

Window and door areas are simply products of their width and height but the wall area

$$A_{wall} = hl - \sum_w A_w - \sum_d A_d \qquad (2)$$

where h is floor height and l is the exterior wall length, must be calculated taking care to account for the window areas A_w and the door areas A_d which are not wall area.

There's also a minor difference between inner and outer dimensions which would affect heat loss calculations. Using the midline within walls is a good compromise that avoids doing special calculations to account for corner and edge losses.

Table 1: Heat losses across exterior surfaces of a building (Finland. Ministry of the Environment. Rakennusmääräyskokoelma, 2017).

Surface type	U value, W/(m²K)
Floor, against exterior air	0.09
Ceiling, against exterior air	0.09
Exterior wall	0.17
Window	1.00
Door	1.00

The U values are given for specific construction materials and constructs depending on their heat insulating properties. Modern buildings have lower values mainly due to stricter energy requirements in construction code such that the year of construction gives a fairly good estimation of the level of insulation used. Table 1 shows the U-values required for new buildings that are used in this work.

Ventilation and air conditioning incur their own losses that are not included in the calculation. Provided that the inlet air is preheated to near room temperature, as is the case with new buildings, this puts no burden on primary heating systems such as radiators or floor heating, and thus can be readily ignored in the adjustment of those systems. The energy use of ventilation is of course relevant for energy certificates and cost analysis but the various complexities in outlet air heat recovery and other ventilation concerns are beyond the scope of this work.

The floor, wall and window areas need to be obtained from floor plans, assuming a given room and window heights, allowing the respective areas to be calculated using widths measured from the drawing.

Occasionally window and door dimensions, including their heights, are marked on drawings in millimetres or decimetres. When available, such information could be read by OCR and used instead of the geometrical measurement, or to validate the correctness of processing.

2.10 Finding the geometry

2.10.1 Tracing the exterior wall

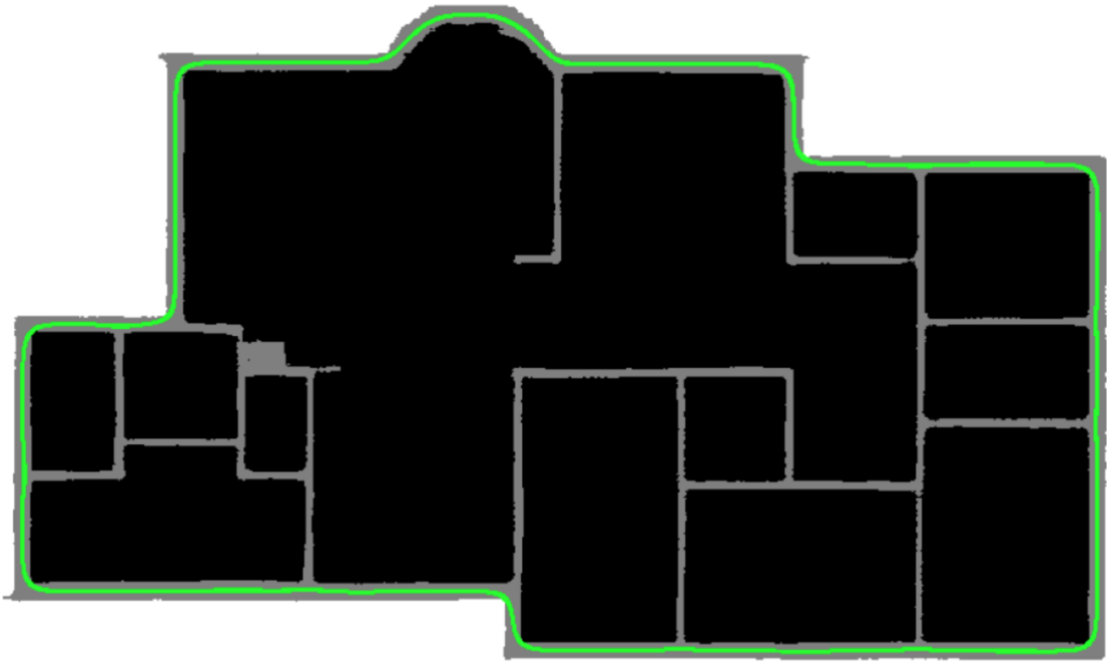


Figure 11: A three-level labelmap showing indoor areas (black), outdoor areas (white) and walls (grey). A contour bitmap (green) following the exterior wall midlines is found by processing the labelmap.

The objective is to create polygonal geometry with point coordinates, using a labelmap like the one seen in figure 11 as produced by the CubiCasa5k model, and finding the contour of indoor space is the first step into geometry conversion.

Tracking the boundary between external and wall isn't a good approach as often walls such as those between or around balconies extend to the exterior. Further, it is preferable to have the curve follow the midline of the walls, not the edge of the wall. Instead, we map the three levels to values -1, 0 and 1 (indoor, wall and outdoor, respectively) and apply gaussian blur to smoothen the map to reduce noise and to ease further processing. A gaussian with σ=10 was found to be ideal such that the smoothing penetrates the walls but doesn't round the corners too much. The value may need to be adjusted depending on the scale that the drawings were scanned in, or if the exterior walls are particularly thick.

While the midline is then found on the contour where the blurred map is near zero, further filtering is added by requiring for the magnitude of the gradient to also be sufficiently large, so that only walls on the building's

boundary qualify. The resulting contour bitmap in shown in green in figure 11.

The initial coordinates are randomly picked off any of the contour points, after which the contour may be traced taking small steps in current direction which is moderately adjusted at each step to avoid falling off the contour. Points are added along the way. The roundness added by smoothing eases following the curve around corners such that there are no accidental 180 degree turnarounds.

2.10.2 Corner fuzzing and clean-up

Many extra points are created in addition to the true corners and the geometry is at this stage distorted. An iterative algorithm is used for post processing, consisting of two alternating stages:

1. Normal-distributed noise is applied to each corner point for slight displacement, and then the walls connecting to the point are checked and scored based on how well they hit the wall regions on the original labelmap. A score-weighted average of 400 such random displacements is taken as the new position.
2. Any points that form roughly 180 degree bends are removed. A cross product between the connecting wall segments having a value less than 400 was found to be a good heuristic.

Twenty iterations, gradually scaling down the noise scale over the rounds (σ from 15 to 5), seems to be able to produce very well aligned geometry with all excess points removed. Despite the impressive amount of calculation, the algorithm executes in a fraction of a second using CPU only.

These points form the exterior wall polygon, with a point at each true corner or equally distributed points along curved sections. Due to this fuzzing the polygon aligns very well with the wall midlines. No assumptions of wall orientation are used but the resulting polygon shows nearly vertical and horizontal edges where needed.

2.10.3 Floor area calculation

The area of a polygon, that may be concave but not self-intersecting, can be directly calculated from its corner coordinates using equation (Bourke 1988)

$$A = \frac{1}{2}|x_1y_2 - y_1x_2 + x_2y_3 - y_2x_3 + \ldots + x_ny_1 - y_nx_1| \quad (3)$$

In buildings that don't contain holes (inner areas of outdoor space), the polygon area is equal to floor area, and can be used for both floor and ceiling heat loss calculation as per equation (1). Areas of any such holes would need to be subtracted from the area calculated for the outer polygon but due to such architecture being rare in Finnish buildings, this was not investigated further.

2.10.4 Windows and doors

Openings are found by following the polygon in roughly one pixel steps, checking the neural network output for detection of such openings. The sampling employs a 3x3 median filter for noise reduction, to avoid any single-pixel breaks or misdetection that would otherwise occur.

Since windows and doors are generally not curved, simple line segments are added along the wall polygon between coordinates where these features are detected. The door and window widths can be found by calculating the line lengths.

2.11 Trials on randomly generated floor plans

As a part of the early trials of this work, the heat loss calculation was evaluated using a simple feed forward type convolutional network that was trained on generated images which eased its training. Images from such testing are shown here for illustration in figure 1. This section documents the methods used in these trials, although eventually superseded by other means as described in prior sections.

The random plans use filled exterior wall markings added with faux lines and text for noise, as well as random rotation and scaling to simulate imprecisions of paper scans. The advantage of random generation is that it given an infinite supply of training images with the ability to also create the corresponding ground truth label maps according to specific needs. In this instance, one specific need was to have exterior wall corners marked on the training labels, so that they would be found by the neural network when in use.

The floor plan shown in has its walls, windows and corners detected nearly perfectly, with only some darker shades denoting less confidence on the

upper-right corner. Corners (green) appear yellow in the prediction because they are also walls (red). The output falls on the pixel grid where each wall is exactly one unit thick.

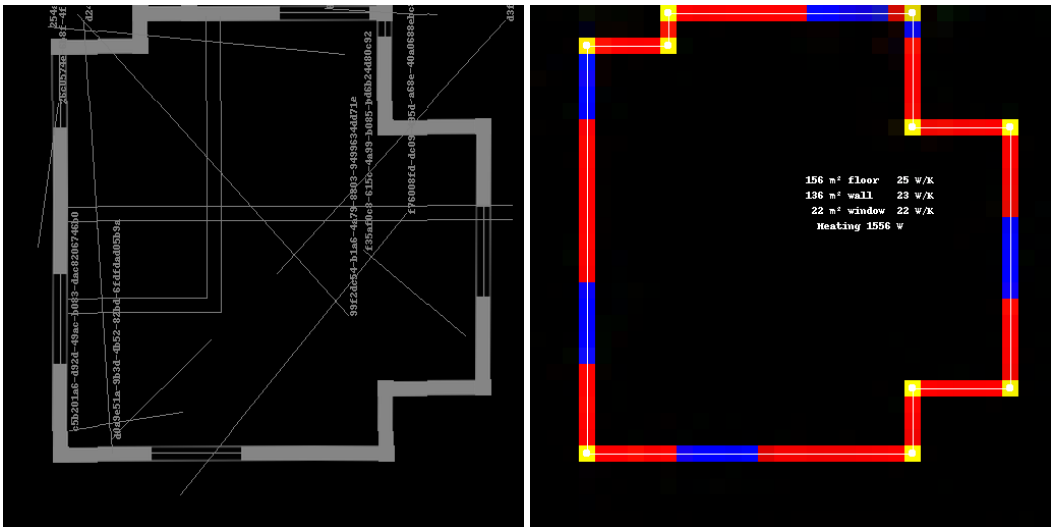


Figure 12: A randomly generated faux floor plan on left. The right side shows a heatmap prediction for walls, windows and corners in colour, and a geometry conversion as a polygon connecting the white dots, along with a heat loss calculation based on the geometry.

The generated plans had hollow interior walls, which the model was trained not to detect as walls, leaving only the outer shell of the building marked in red and blue.

All likely corner points' coordinates are extracted from this detection, and connected as a single closed polygon in an order determined by the traveling salesman algorithm, which is weighted with a distance matrix that favours crossing walls and windows on the heatmap.

Geometry construction begins by finding the most likely corner points by a neural network which produces a heatmap showing corner points as well as walls.

The strongest detected corners are traversed using the traveling salesman algorithm, in all possible combinations, picking up the one that covered the greatest amount of walls and windows with the least amount of total circumference, so that a closed polygon covering all exterior walls could be obtained.

Simply minimizing the polygon circumference is not adequate because the shortest path often doesn't follow the walls. The scoring function used in traveling salesman needs to be modified so that it prefers routes that overlap with wall and window detection. To aid this, a matrix containing the travel cost

$$c_{\boldsymbol{ab}} = -\frac{1}{1 + \|\boldsymbol{b}-\boldsymbol{a}\|}\int_{\mathbf{a}}^{\mathbf{b}} \max\left(p_{wall}(\boldsymbol{s}),\ p_{window}(\boldsymbol{s})\right) d\boldsymbol{s} \qquad (4)$$

between each pair of corner points a and b. The cost is based on the ratio of walls hit over its total distance according to this equation which was found by experimentation to yield good results, although there is no clear theoretical justification to the form of the scaling factor in front of the integral. The added constant in the denominator gives adequate weight for minimizing total distance and not only the ratio, so that the shortest path gets chosen over candidates that otherwise hit the wall predictions equally well. The function p gives the prediction of a wall or a window under the path between a pair of corners, at position s. The integral is evaluated numerically in small steps, picking predictions along the line from a to b. A polygon connecting all corners with the lowest total cost gets chosen.

A heat loss calculation is shown as a proof of concept using specific U values and temperatures where the required wattage is calculated using the detected wall and window lengths as well as the calculated polygonal area for floor and ceiling losses. For the purpose of the example, one pixel in labelmap corresponds to 0.5 meters. The wall height is 3 meters and the window height is 1.8 meters.

2.12 Trials on interior space detection with standard U-Net

The early attempts taken in this work also include a standard U-Net which was trained to find the interior space using the external walls. Since it quickly became apparent that the network was limited to small resolution images due to its heavy RAM usage, it was postulated that the low resolution wall output of the convolutional model described in section 2.11 above could be used in this further processing.

Initial training was done using a simple random generator that produced 1-pixel outlines and filled interior label maps similar to building shapes. Although this approach was ultimately abandoned, examining it during training gives insight into how the other models with feats similar to U-Net behave.

The model was able to learn this task almost perfectly as seen in a screenshot of the training tool (figure 13) where the evaluation error decreases to almost zero except for occasional spikes denoting mistakes, and the prediction matches the ground truth in shape.

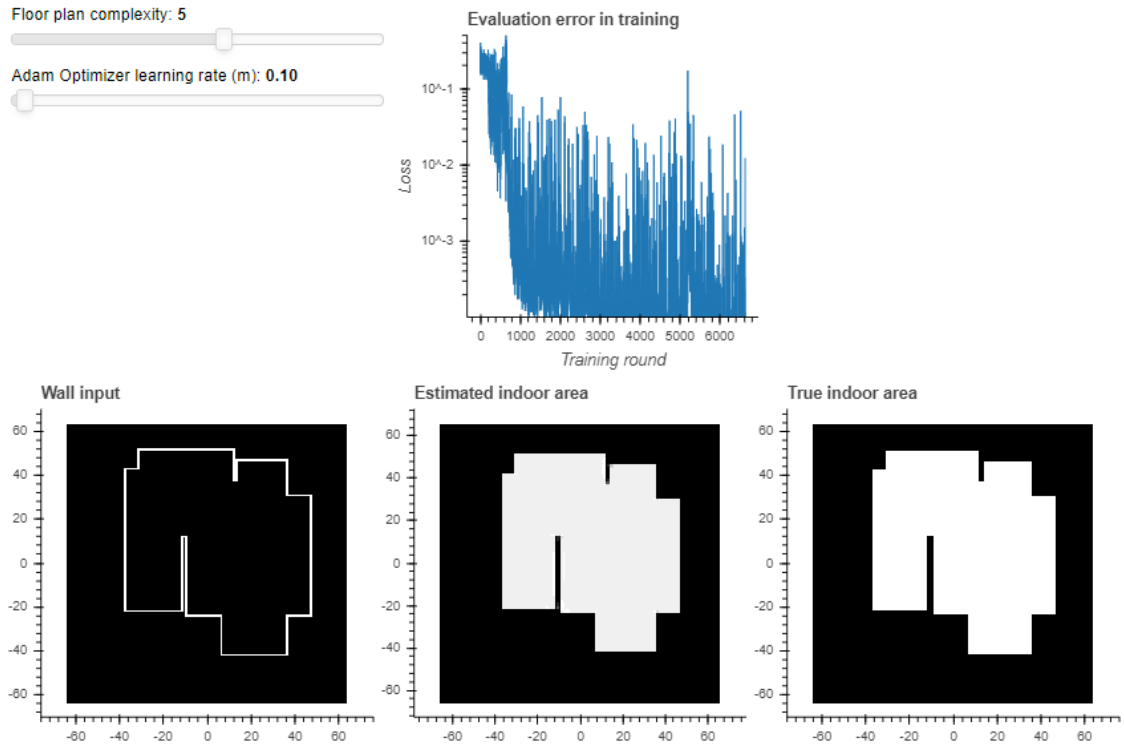


Figure 13: Training U-Net after 6600 rounds using randomly generated floor plans (bottom left) and their corresponding labelling of indoor space (bottom right). The predicted heatmap is shown in the middle and it appears at a very slightly dimmer shade of white due to the uncertainty still present.

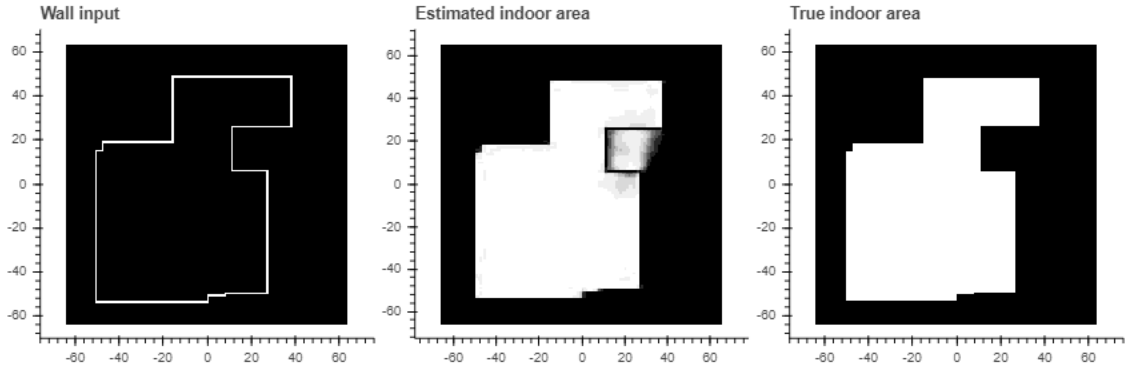


Figure 14: A screenshot taken midway in training still shows misdetection of a recess as an interior space, showing the difficulty of determining which side is inside by local inspection only.

In early training rounds the network was clearly first employing the bypass connections and thus only inspecting the very local neighbourhood, first only a few pixels around the walls showing anything but grey. Figure 14 shows a stage where the deeper levels have already come into play but the network

still struggles with concave sections and other oddities like holes inside the building that locally look the opposite of what they are considering the whole building.

The model displayed excellent results in testing with convex buildings and a moderate amount of uncertainty with various concave constructs. With increased training even the complex shapes could be detected almost perfectly. In contrast to a simple flood fill algorithm, the neural network was resilient to noise and could also correctly handle recesses, holes and separated sections of buildings that would be otherwise difficult to process automatically.

The other models evaluated in this work can accomplish good segmentation results directly on higher resolution drawings that also contain inner walls, employing similar principles but not taking a fully global view of the building. For ones wishing to pursue this design, I suggest, rather than doubling the number of channels at each deeper level, to cap the channels to about 256, as is done in CubiCasa5k, which also employs a 4x downscale prior to its U-Net stage, and a corresponding 4x upscale on the network output, making it able to handle larger images.

3 Experimental results

3.1 CNNFloor

Figure 15: The evaluation errors (losses) for the training and validation images during training. Training is cut off at round 80 000 to avoid overfit.

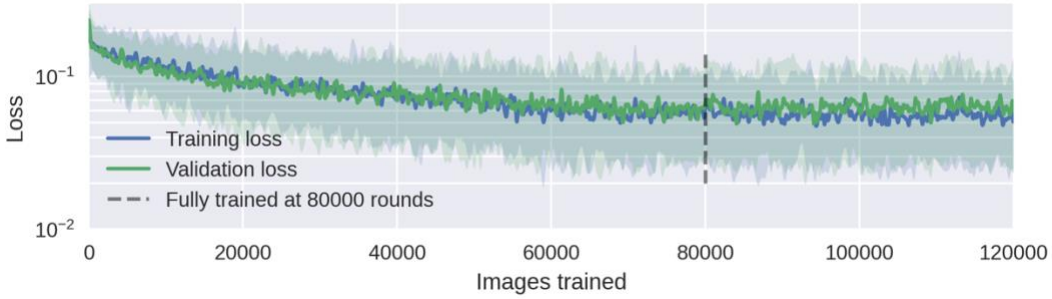


Figure 16: Floorplan images of the CubiCasa5k validation set superimposed with markings found by the model: walls (red), windows (blue), doors (green) and interior space (dark shade).

The model was trained on the training set (N=4200) of the CubiCasa5k dataset using Adam optimizer. A training session lasted roughly 12 hours on an Nvidia RTX 3090. Training and validation losses were tracked and the network was saved at regular checkpoints so that backtracking to an optimal cut off point was possible. (Figure 15)

Finding the interior floor area seemed easier than expected during the early training, where a rough outline was found before even the walls were

properly identified. Initially the model took any black pixels as walls but eventually learned to pick up the relevant patterns and to process hollow walls properly, while ignoring any other markings such as text and furniture.

The plans in figure 16, none of which were used in training, show typical performance. There is mostly correct interior area detection, with only minor errors in particularly convoluted locations where walls may also go undetected (e.g. bottom left building's corner). Windows easily get confused with walls especially on low quality original drawings where they are hard to tell apart by human eye, and there are sometimes problems with doors, which turned out to be the hardest element to detect.

A slight misalignment by a few degrees was apparently so common in the training dataset that the model had only minor issues with the pair of angled buildings on the left but one common difficulty appears with 45 degree angled sections that often go undetected due to their scarcity in the dataset.

Evaluation results over the CubiCasa5k test set (N=400) are summarized in table 2, where this model is 90 % accurate in predicting the correct class.

In the good cases the model output can be used for further processing in fully automated geometry extraction but often there are errors that would require at least some manual intervention.

Table 2: CNNFloor model accuracy measured over the test dataset

True label	Predicted Label, counts in megapixels					Prediction
	Exterior	Interior	Wall	Door	Window	Accuracy
Exterior	112.21	8.03	2.06	0.14	0.28	91 %
Interior	7.63	99.00	2.69	0.41	0.21	90 %
Wall	1.15	1.09	14.63	0.11	0.24	85 %
Door	0.09	0.29	0.12	1.17	0.06	68 %
Window	0.27	0.11	0.49	0.02	2.88	76 %
					Overall:	**90 %**

Table 3: CubiCasa5k model accuracy measured over the test dataset

True label	Predicted Label, counts in megapixels					Prediction
	Exterior	Interior	Wall	Door	Window	Accuracy
Exterior	116.35	3.93	2.14	0.07	0.22	95 %
Interior	5.07	103.34	1.40	0.08	0.05	94 %
Wall	1.01	1.37	14.74	0.03	0.08	86 %
Door	0.13	0.25	0.29	1.04	0.03	60 %
Window	0.25	0.22	0.50	0.02	2.77	74 %
					Overall:	**93 %**

3.2 CubiCasa5k

The pretrained model published with the CubiCasa5k dataset was used to find walls and indoor areas on its rooms labelmap and door and window on its icons labelmap, which were then overlaid into a single labelmap in a manner similar to the other model evaluated, producing the labelmap seen in figure 17 which shows a very high degree of detection accuracy.

Figure 17: The CubiCasa5k pre-trained model is used to detect walls (red), windows (blue), doors (green) and interior space (dark shade) on an architectural floorplan (testing set of the CubiCasa5k dataset).

Despite the scarcity of angled walls in the dataset the model handles 45 degree and other angled sections fairly well, and in general produces very few gross errors for the labels used here.

For small and medium sized rooms the model usually produces correct mapping of indoor and outdoor areas, when the room labelmap is interpreted such that background, outdoor and railing are taken as exterior areas. The garage label turned out to be problematic and would need to be manually set as exterior for buildings with an outdoor garage. Separate room functions are noisy but not of concern with this simplified mapping.

Evaluation results over the CubiCasa5k test set (N=400) are summarized in table 3 where this model performs on most statistics slightly better than the CNNFloor model, with an overall accuracy of 93 %.

Although errors remain, the result is remarkably good and on most floor plans suitable for further processing.

3.3 Geometry extraction for heat loss estimation

The geometry processing employs only the information present within the labels visualised in figure 17 above. The exterior boundary is first identified and traced into a polygon, which is further cleaned by the custom fuzzing algorithm after which windows and doors are separated as their own line segments as described in section 2.11.

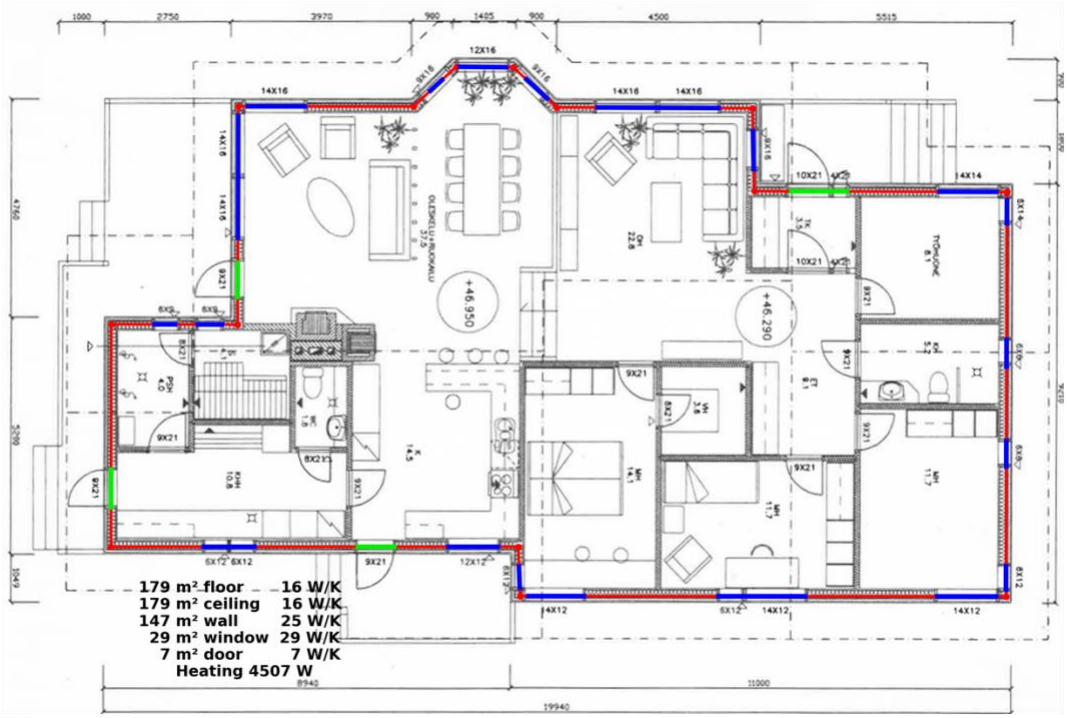


Figure 18: Polygonal representation (point coordinates) for the exterior walls (red) is first traced and then sampled on the label map (figure 17) so that line segments may be separated for doors (green) and windows (blue). The heat loss calculation uses polygon area, circumference as well as door and window widths, showing the maximum heating requirement (at -26 °C outdoor temperature).

The wall geometry extracted (figure 18) does not make any assumptions about orientation, and thus is able to find the angled sections (top middle) correctly, and it also doesn't depend on the network to detect corner locations. This is in contrast to the rectangular representation used in earlier work where that section and parts of the exterior wall are missing (figure 19).

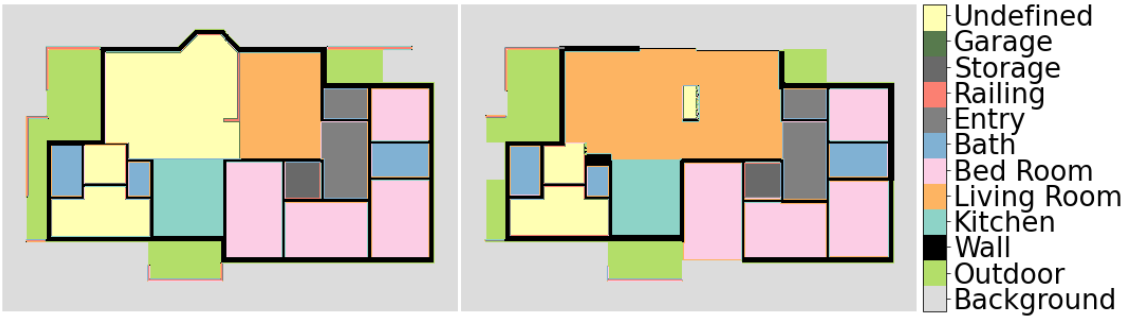


Figure 19: Kalervo, et al. (2019). Rendered from CubiCasa5k sample notebook. The ground truth room labelling (left) compared to the geometric presentation (right) built via original post-processing code.

The dimensions found closely approximate the actual dimensions, as can be verified by the window and door markings. E.g. for doors the written dimensions on this drawing give 8.6 m² total area, only slightly larger than that detected via geometry. Windows on the angled part are only partially detected on the labelmap but no errors affect the heat loss calculation in any relevant magnitude. For the sake of simplicity, all windows were assumed to be 1.4 m high, as the textual markings were not being processed.

4 Analysis

The CubiCasa5k model was the most reliable in wall detection and while the room type detection was very noisy, the exterior vs. interior discrimination was fairly accurate and most windows were detected correctly. Although apparently not particularly trained for, it could handle angled walls and windows but circular walls went largely undetected.

The model produces a rich set of 44 output features which notably can recognize room functions. Other features such as corner markers seem less useful as they are restricted to 90 degree corners. The CubiCasa5k dataset, alike, lacks labelling for such corners even though this information could be extracted from the SVG representation with fairly simple logic. It needs to be noted, though, that most buildings are mostly or entirely rectangular, so the choice seems justified.

The CNNFloor model despite using only half the number of parameters and no pretrained precursor networks did surprisingly well. It can certainly be used in real applications and due to its lesser hardware requirements and full resolution internal processing it might be better suitable of the two for high resolution blueprints, in contrast to the somewhat messy drawings in the CubiCasa5k dataset. The biggest shortcomings were with angled walls, where both models had issues but these were far more apparent with my model. Presumably both models could use additional training for these cases, applying random rotations rather than 90 degree rotations in training.

Most text, single lines, furniture and other markings were ignored correctly by both models, even when the symbol in question had not been trained for, and both were also notably resilient against the dense markings present in HVAC drawings partially obscuring the elements of interest.

Finding a contiguous loop of exterior walls posed its own challenges because the outline of a building could not be determined from the drawing symbols. Both models evaluated somehow learned to leave out balconies and to generally produce logical interior segmentations.

CubiCasa5k showed better reliability in detecting all walls and in this regard the model proved very useful, as in many cases it was able to produce a nearly error-free labelling, paving way for geometry extraction. In many but not all images a solid continuous contour line was found that could be traced into geometry with no a priori data on corner points needed.

Wall thicknesses would often vary within the same wall for various architectural details in ways that could not be easily translated into a geometry following the middle line within the wall. The geometry fuzzing algorithm I implemented can in most cases create the correct geometry with a single edge representing a wall along the middle of its contiguous straight segment if there is one, and in general following the midline of walls whenever possible. Only rarely an incorrect configuration of corner points are found so that the polygons does not properly follow the walls because of

a missing point on some true corner. Because the problem is rare, I did not implement point insertion when such a problem is detected.

Angled and curved walls, when detected by the model, can be converted into polygon geometry, which to my knowledge has not been possible in prior work.

5 Conclusion

Commercially viable solutions to fully automated floor plan analysis and geometric conversion exist, at least for the purposes of real estate sales where the strictest accuracy might not be required. If augmented by manually correcting specific corner cases (pun intended), the current solutions could drastically reduce the amount of work required in recreating CAD drawings.

Engineering work such as heat loss calculations required in energy certificates and hydronic balancing still require information not present in architectural or HVAC drawings, so even an ideal solution could never be fully automatic but the reduction of work by the current level of automation is already substantial and thus it should have a real market impact if further refined into toolsets used in this field.

The solution presented in Liu, et. al (2017) and further refined in Kalervo, et. al (2019) can produce nearly error-free conversion with remarkable precision of measurements either with the groups' own post processing or with my geometry conversion algorithm. The CubiCasa5k model is also finding use in entirely unrelated fields, such as video games (Hauglig, 2020).

The original research group has also worked on related projects that might interest the reader. For instance, conversion of depth video into floor plans and 3d geometry. (Liu, et. al. 2018).

Building half a dozen distinct CNN models over the course of this work and experimenting with U-Net and CubiCasa5k has been an educational experience which already led to a couple of side projects, one a commercial machine vision project and another an automatic captcha solver which I wrote in a few hours with the insights gained working on this thesis.

More than anything, this work has shown me the vast ability of convolutional networks to find the requested features from within complex input data far beyond than what one could imagine using only hard-coded traditional computer vision models. However, this magic only happens with a sufficiently large, diverse and presentative training dataset, or otherwise features specific to the training samples are found rather than those intended and perhaps obvious to a human observer.

At this point in time it seems safe to say that there won't be another AI winter but the latest revolution that began circa 2015 is only getting stronger. Neural networks are already widely used in various detection tasks which are not going away. Still, my personal interests lie in making AIs that can produce better code, e.g. learning from Github commits what are the preferable changes and applying them to other code. Time will tell whether something like that will fly, and whether a code-improving AI is the path to singularity.

References

Ahmed, S., Liwicki, M., Weber, M. and Dengel, A., 2012, March. Automatic room detection and room labeling from architectural floor plans. In *2012 10th IAPR International Workshop on Document Analysis Systems* (pp. 339-343). IEEE.

Aoki, Y., Shio, A., Arai, H. and Odaka, K., 1996, August. A prototype system for interpreting hand-sketched floor plans. In *Proceedings of 13th International Conference on Pattern Recognition* (Vol. 3, pp. 747-751). IEEE.

Bourke, P., 1988. Calculating the area and centroid of a polygon. *Swinburne Univ. of Technology*, *7*.

Bulat, A. and Tzimiropoulos, G., 2016, October. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision* (pp. 717-732). Springer, Cham.

Camocho, C., 2018. Convolutional Neural Networks. Available at: https://cezannec.github.io/Convolutional_Neural_Networks/ (Accessed: 3 May 2021).

Dodge, S., Xu, J. and Stenger, B., 2017, May. Parsing floor plan images. In *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)* (pp. 358-361). IEEE.

Duda, R.O. and Hart, P.E., 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), pp.11-15.

Hauglig, J.O., 2020. RuneScape Mazemap. Available at: https://github.com/rune-mazemap/CubiCasa5k/ (Accessed: 22 May 2021)

Filipski, A.J. and Flandrena, R., 1992. Automated conversion of engineering drawings to CAD form. *Proceedings of the IEEE*, *80*(7), pp.1195-1209.

Finland. Ministry of the Environment, Rakennusmääräyskokoelma, 2017. *Ympäristöministeriön asetus uuden rakennuksen energiatehokkuudesta* (1010/2017), 24 §. Available at: https://www.finlex.fi/fi/laki/alkup/2017/20171010 (Accessed: 31 May 2021).

Kalervo, A., Ylioinas, J., Häikiö, M., Karhu, A. and Kannala, J., 2019, June. Cubicasa5k: A dataset and an improved multi-task model for floor plan image analysis. In *Scandinavian Conference on Image Analysis*, pp. 28-40. Springer, Cham.

Liu, C., Wu, J., Kohli, P. and Furukawa, Y., 2017. Raster-to-vector: Revisiting floor plan transformation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2195-2203).

Liu, C., Wu, J. and Furukawa, Y., 2018. Floornet: A unified framework for floorplan reconstruction from 3d scans. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 201-217). Available at: http://art-programmer.github.io/floornet.html (Accessed: 06 July 2021)

Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431-3440.

Noh, H., Hong, S. and Han, B., 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1520-1528).

Okorn, B., Xiong, X., Akinci, B. and Huber, D., 2010, May. Toward automated modeling of floor plans. In *Proceedings of the symposium on 3D data processing, visualization and transmission* (Vol. 2).

Ronneberger O., Fischer P. and Brox T., 2015, November. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham.

Ruwanthika, R.G.N., Amarasekera, P.A.D.B.M., Chandrasiri, R.U.I.B., Rangana, D.M.A.I., Nugaliyadde, A. and Mallawarachchi, Y., 2017, January. Dynamic 3D model construction using architectural house plans. In *2017 6th National Conference on Technology and Management (NCTM)* (pp. 181-184). IEEE.