Flat internal organization from the footprint

Artem Tokarevskikh

ČVUT - FIT

tokarart@fit.cvut.cz

December 31, 2021

1 Introduction

Artificial intelligence becomes a common tool among various fields such as science, marketing, business, and even art. However, there are not a lot of cases when AI was used in architecture. This field requires a lot of skills in completely different disciplines as well as human expertise. We decided to try to apply modern methods of generative machine learning to the field of design of living spaces. Our goal is to create a model, which gets the basic information about the flat, such as its shape and entrance position and suggests its internal organization in terms of living space position and its size.

2 Data

2.1 Generation

There is no public dataset with flat plans, so we had to figure out how to create our own. The first idea was to scrape house plans from the internet and then extract the plans of single flats. The problem with this approach is that such plans have a different format and look, so it is not usable. We found a plugin for Rhino software for floor plan generation [1]. My colleague implemented an algorithm that generated floor plans for flats of different sizes (1kk, 1+1, 2kk, 2+1, etc.). He generated around 2k of such samples.

2.2 Preprocessing

Generated data was not ready for training, since the aspect ratio of the flats was not correct (1kk and 5kk had the same size) and there was a lot of artifacts and unnecessary parts on the images. Moreover, only plans with marked rooms were generated (targets), we still lacked an input set.

Preprocessing pipeline is the following

1. Find the contour of the floor plan on the image.
2. Resize the flat plan according to the predefined ratio (see Table **??**) and get a target.

n_rooms	1	2	3	4	5
ratio	0.1	0.25	0.5	0.75	0.9

Table 1: Ratio of the plan area to the image area.

3. Find the entrance position (intersection of a forecourt and background) and mark it outside the plan.
4. Fill the flat contour with black colour and get an input image.

I used the cv2 library for preprocessing purposes. Data preprocessing was performed in the notebook `data_prep.ipynb`. Finally, we have a dataset with inputs and targets for 2k flat plans. If we add a simple augmentation of rotation to 90°, 180° and 270° we got a dataset of a volume of 8k samples. Samples from the dataset are displayed in the following Figures.

3 Methods

For the experiments I have used the following tools

- PyTorch + PyTorch Lightning as deep learning framework.
- Google Collab Pro.
- Weights & Biases for experiment logging.

3.1 Autoencoder

I decided to use the convolutional autoencoder architecture for the initial modelling and use it as a baseline model. The architecture is the following (see Figure **??**). The idea is that the encoder part would extract the main features of the input image (such as its area, entrance position, etc.) and the decoder part would suggest the inner flat organization from this lower dimension representation. As a loss function, I choose the pixel-wise MSE loss between output and target. LeakyRelu was used as the activation function to prevent the dead Relu problem

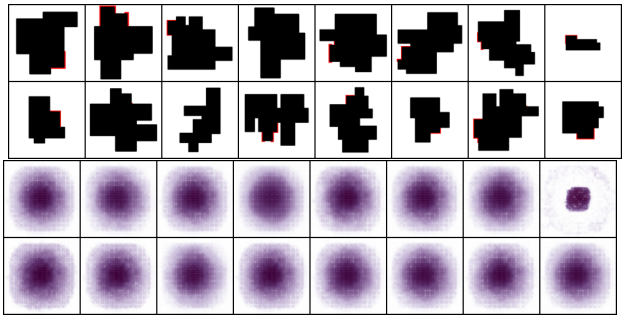

Figure 1: Produced output by VAE.

occurrence. After every convolution, I performed batch normalization to prevent internal covariate shift [2].

I also experimented with VAE. Around 50 epochs the KLD loss was disabled in order the model could learn to reconstruct without the need of generation in accordance with the distribution. Then the KLD loss was turned on with increasing KLD weight on every epoch start. This way we force the network to consider the distribution increasingly with every epoch. However, I did not manage to train this network. After including the KLD term to the loss function, the generated images became just a noise (See Figure **??**) and the network could not converge.

3.2 pix2pix

To evaluate my model performance, I decided to fine-tune the pretrained model and check the results. In the domain of image generation, generative adversal networks are the method of choice. I found the model called *pix2pix* [3]. In the repository [4] the implementation in PyTorch is provided, as well as checkpoints of networks trained on the different data. One of the pretrained networks was trained on labels-to-facade dataset (see Figure 2)which is very similar to the problem we tried to solve. The aim of this experiment is to compare the quality and morphological features of the samples generated by AE and pix2pix GAN.

4 Results

4.1 Autoencoder

Train, resp. test set outputs are shown in the Figure **??** resp. Figure **??**. Generated samples from test set are very noisy and cannot be used for real inner design planning. However, the model learned to place the corridor near the marked entrance (greyish spot). The logic of the suggested rooms and their positions to be discussed with my colleague from FA, CTU and the results will be evaluated from this point of view.

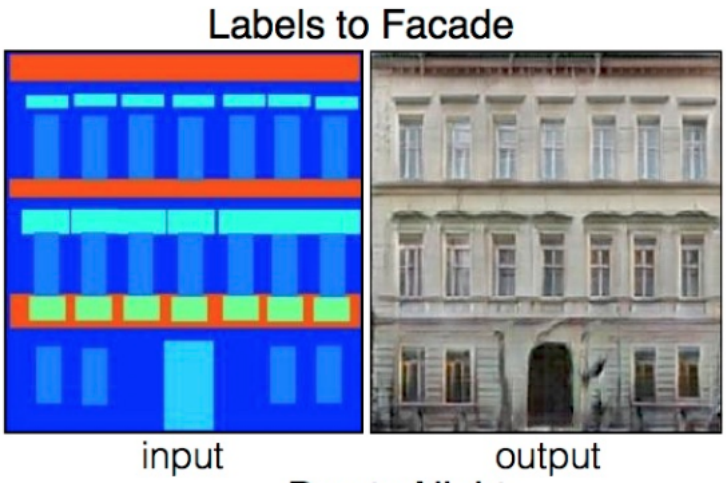

Figure 2: Example from labels2facade dataset.

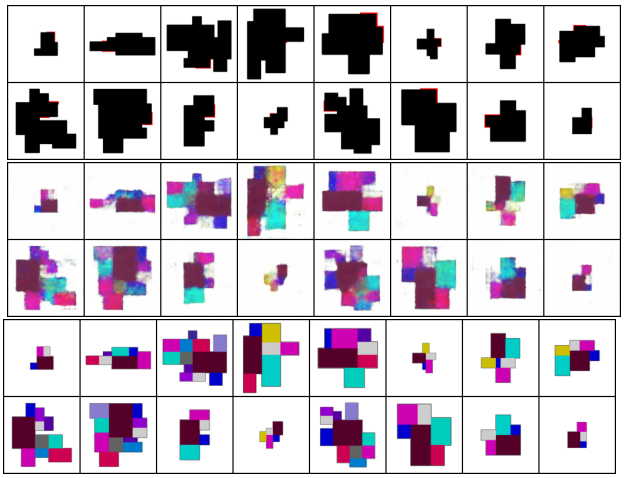

Figure 3: Input, produced output by AE and target from training set.

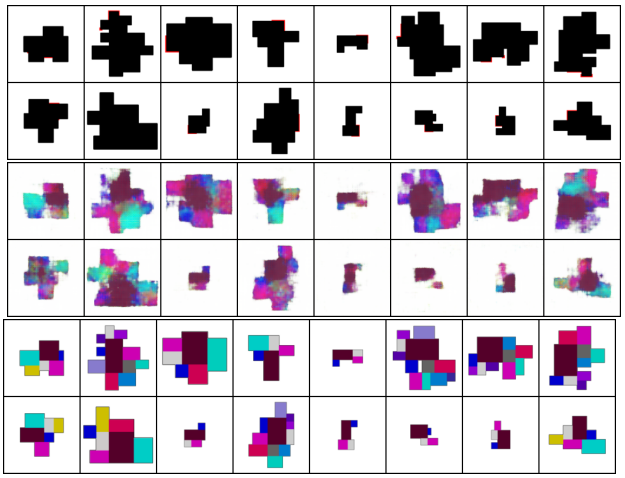

Figure 4: Input, produced output by AE and target from training set.

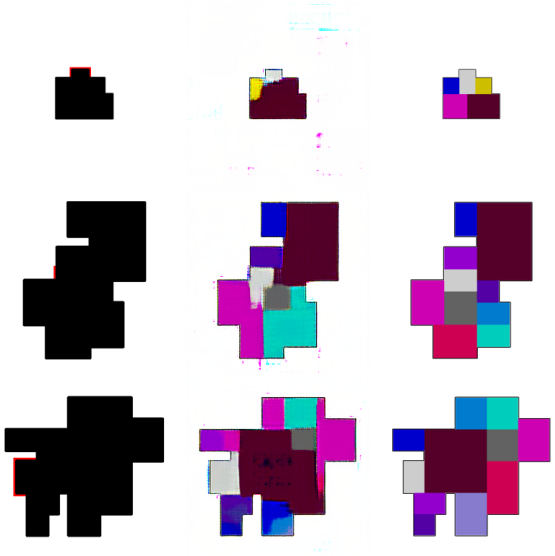

Figure 5: Some result produced by pix2pix model. On the left image is the input, in the middle is the generated output and target on the right

The model in this form cannot be used in production pipelines, however, for the initial modelling it shows that even with such a simple model like AE you can achieve fairly decent results. More time spent on research and the more powerful computational resources would improve the model performance.

4.2 pix2pix

In previously mentioned repository with PyTorch implementation of the pix2pix model, there is a prepared script for fine-tuning. Some changes in the dataset need to be performed to use their implementation. After 200 epochs, the following results were achieved (see Figure 5)

Outputs produced by pix2pix model are cleaner, have less noise. Moreover, the borders between rooms are more significant than in the case of AE. Generated floor plans usually have fewer rooms than the target, moreover, the network prefers some type of rooms over others.

5 Conclusion

In this work, we illustrated that the deep learning methods might be used in architecture, particularly in the floor planning stage. The results leave a lot to be desired. The logic of the suggested rooms and their positions to be discussed with my colleague from FA, CTU and the results will be evaluated from this point of view.

In the future, I would experiment more with (V)AE architecture. First improvement point is to set up a working VAE model. Some resources mentioned that the skip-connections might improve the results. Regarding GAN models, I think they are more promising. However, taking into account the size of the model and the fact that we used already pretrained model on the similar data, having better results is expected. I didn't experiment with own GAN implementation due to time limits and the fact that it is harder to train such network. Hopefully this project will be continued and we will achieve results that might impact the way houses are designed.

References

[1] Martin Bielik. Magnetizing floor plan generator. online.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[4] Jun-Yan Zhu. Image-to-image translation with conditional adversarial networks. pytorch implementation. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.