Data Science and Engineering (2022) 7:213–224
https://doi.org/10.1007/s41019-022-00192-6

RESEARCH PAPERS

Toward Enhancing Room Layout Estimation by Feature Pyramid Networks

Aopeng Wang[1,2] · Shiting Wen[1] · Yunjun Gao[3] · Qing Li[4] · Ke Deng[5] · Chaoyi Pang[1]

Received: 2 March 2022 / Revised: 13 July 2022 / Accepted: 1 August 2022 / Published online: 12 August 2022

Abstract
As a fundamental part of indoor scene understanding, the research of indoor room layout estimation has attracted much attention recently. The task is to predict the structure of a room from a single image. In this paper, we illustrate that this task can be well solved even without sophisticated post-processing program, by adopting Feature Pyramid Networks (FPN) to solve this problem with adaptive changes. The proposed model employs two strategies to deliver quality output. First, it can predicts the coarse positions of key points correctly by preserving the order of these key points in the data augmentation stage. Then the coordinate of each corner point is refined by moving each corner point to its nearest image boundary as output. Our method has demonstrated great performance on the benchmark LSUN dataset on both processing efficiency and accuracy. Compared with the state-of-the-art end-to-end method, our method is two times faster at processing speed (32 ms) than its speed (86 ms), with 0.71% lower key point error and 0.2% higher pixel error respectively. Besides, the advanced two-step method is only 0.02% better than our result on key point error. Both the high efficiency and accuracy make our method a good choice for some real-time room layout estimation tasks.

Keywords Layout estimation · Scene understanding · Feature Pyramid Network

1 Introduction

Indoor room layout estimation is a fundamental task for many applications, including indoor navigation, augmented reality, and scene reconstruction. This study focuses on the indoor layout estimation to predict the structure of a room through a single RGB image.

Figure 1 shows the definition of different room layout types and an example in LSUN dataset. From this figure, we can see that a room has three parts: ceiling, wall and floor. The wall can be further divided into the left wall, middle wall and right wall in general, from different perspectives. These parts are labeled with different semantic for semantic segmentation. The junction points of different parts are considered as the key points in connections and each point is labeled with a number, representing their order for a given room type. The example at the bottom of Fig. 1 illustrates the segmentation map and room layout for an input image, which is the expected output of this task.

Intuitively, as long as these key points are correctly detected and connected in a proper order, the correct room layout can be obtained. Therefore, the major task in room layout estimation is to correctly detect the key points in the

✉ Shiting Wen
wensht@nit.zju.edu.cn

Aopeng Wang
wap8cn@gmail.com

Yunjun Gao
gaoyj@zju.edu.cn

Qing Li
qing-prof.li@polyu.edu.hk

Ke Deng
ke.deng@rmit.edu.au

Chaoyi Pang
chaoyi.pang@nit.zju.edu.cn

1 School of Computer and Data Engineering, NingboTech University, Ningbo, China

2 Polytechnic Institute, Zhejiang University, Ningbo, China

3 School of Computer Science and Technology, Zhejiang University, Hangzhou, China

4 Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

5 School of Science, RMIT University, Melbourne, Australia

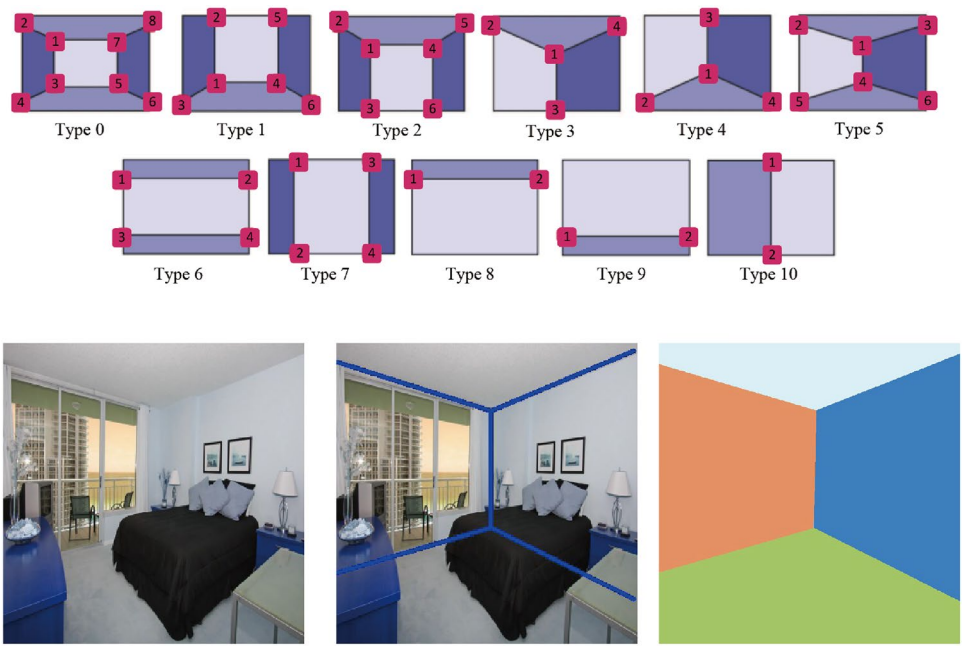

Fig. 1 Definition and an example of room layout in LSUN dataset

right order. However, it is challenging to accomplish this task through a single RGB image, because rooms in a photo can be cluttered and shadowy in different ways.

In this paper, we see the key points as small objects and adopt the Feature Pyramid Networks (FPN) [1, 2], which is one of the most effective approaches to deal with both the small objects detection and segmentation task, to predict the room layout in end-to-end style. With proper modification, we demonstrate that FPN could solve this task well even without the commonly used post-processing step in prior works and we record the modified FPN as M-FPN. Besides, by keeping the order between key points stable during the data augmentation stage, the proposed model can get a better performance. Moreover, this step is a low-cost operation and can be applied in other similar order-related work, but has not been discussed in previous works.

The contribution of this paper can be summarized as follows:

- M-FPN is the first approach that uses FPN to solve room layout estimation in end-to-end style. It shortens the processing time and can be used for real-time processing tasks.
- We propose a new technique called "order-preserving" to enhance model's performance. This step makes each point consistent with its activation area and can basically eliminate the wrong matching problem by preserving the order between key points in the process of data augmentation. We are the first to devise and deploy them to indoor layout estimation.
- The proposed model demonstrates great performance on the challenging benchmark LSUN dataset [3]. Our method is comparable to the advanced end-to-end methods on the pixel error metric, while it outperforms the end-to-end methods on the key point error metric. In particular, it achieves a similar level of key point error as that of the state-of-the-art two-step methods. To the best of our knowledge, it is the fastest method of indoor layout estimation and reaches 31fps at processing speed.

The rest of this article is organized as follows. Section 2 reviews the related work of indoor layout estimation. Section 3 describes our proposed method and framework. Section 4 reports the experimental results. Section 5 concludes this article.

2 Related Work

The methods of indoor layout estimation can be divided into two styles according to whether an independent post-processing program is used.

2.1 Two-Step Methods

The two-step methods usually follow the proposal-ranking scheme, in which the whole process, from feature extraction to proposal ranking is processed by a human-designed program [4–8]. As the rise of deep learning, some two-step methods use neural networks for feature extraction with great improvements in accuracy [9–13]. In general, these methods use an additional post-processing step to elevate the results' precision but require a much longer processing

time and more manually participating than that of the end-to-end methods.

2.2 End-to-End Methods

The end-to-end methods [14–16] have faster processing speed but these methods usually lack a well-designed post-processing mechanism resulting in less sufficient performance on accuracy than two-step methods. RoomNet [14] is the first end-to-end method to solve the layout estimation task. It follows the SegNet [17] structure, which is featured with encoder-decoder architecture. RoomNet adds an extra classifier composed of three fully connected layers at the bottleneck of SegNet to predict the room type. Since there are 48 points in total in the 11 types of rooms in the LSUN dataset, this network outputs a feature map of 48 channels with each channel corresponding to a point. To avoid using an extra classifier, Roth, et al. [15] proposed a 'Smart Hypothesis Generation' (SHG) method to divide the training data into three groups according to the number of walls in the input images, where each group is used to train a network respectively. Each network has two parts, the first of which outputs semantic segmentation maps while the second outputs key point heatmaps from the segmentation maps. After three sets of semantic segmentation and key point heatmaps are obtained, a scoring function is used to calculate the consistency between the segmentation maps and the key point heatmaps. The algorithm then chooses the highest score as the final result. This method gets rid of an extra classifier by using 3 groups of networks and a scoring function to select the best matches. Although this method avoids an explicit room type classifier, the disadvantage is that it exploits 3 groups of networks, which makes the training process more complicated and takes more time in inference.

In this article, we use the same output setting of RoomNet but we increase the accuracy of the classifier by using a stronger feature extraction backbone. Our proposed method is also different from SHG [15] by employing a single network which can be more efficient than SHG method. The output size of our method is only 1/4 of the input size, which is very similar to that of model compression. To enhance the accuracy of M-FPN, we also make each point consistent with its activation area. From this, our method achieves the state-of-the-art performances for both the processing speed and accuracy of the tested data sets.

3 M-FPN Method

In this section, we first introduce the proposed end-to-end method M-FPN in Sect. 3.1. We then propose the "order-preserving" strategy in Sect. 3.2 to optimize the proposed network for better outputs in accuracy. Section 3.3 indicates how to refine the positions of corner points through the coarse positions.

3.1 M-FPN Network

Although SegNet has been used in both RoomNet and SHG method, we see another choice in basic network architecture: Feature Pyramid Network. As shown in Fig. 2, M-FPN is a single-network and basically modified from the network architecture of [2] and has the following characteristics:

3.1.1 Characteristics

Compared to the encoder–decoder structure of SegNet, FPN sets most parameters and performs most computations on the encoder (backbone) part to get stronger features, which ensures that the extra classifier will have higher accuracy and will not bottleneck the model's overall performance. Meanwhile, it merges the multi-scale features with just a few operations in the decoder part. The proposed network can be seen as an asymmetric, lightweight encoder-decoder, which is very efficient in computing and very appropriate for saving GPU memory. By adopting this architecture, our proposed model improves the classification accuracy from 81.5% in [14] to 83.5%.

Besides, considering that the key points belong to the small details in the picture, we first upsample the four different scale features in the feature pyramid to the same scale and then integrate them together as shown in Fig. 2. The integrated feature retains both strong semantic information and richer details. The model's performance is then effectively improved. By comparison, RoomNet only uses the pooling index recorded at the subsampling process to produce dense feature maps of a larger size during the upsampling process. This network will miss the information in the former extracted features, a common problem caused by the symmetrical encoder-decoder network.

Our proposed network works as follows:

- It takes a 320*320 image as input and outputs a feature pyramid with the feature size ranging from 1/32 to 1/4 of the original size.
- We add a classifier, which includes an average pooling layer, a dropout layer [18] and a fully connected layer, after the Block4, to predict the room type. The dropout layer can prevent the model from being overfitting. A single fully connected layer can decrease the number of parameters and provides relatively high accuracy.
- As shown in Fig. 2, f1∼f4 in the feature pyramid are upsampled different times to get the p1∼p4 at 1/4 of the original input size, and then we add them in pixel-wise and perform a 3*3 convolution to get the final result.

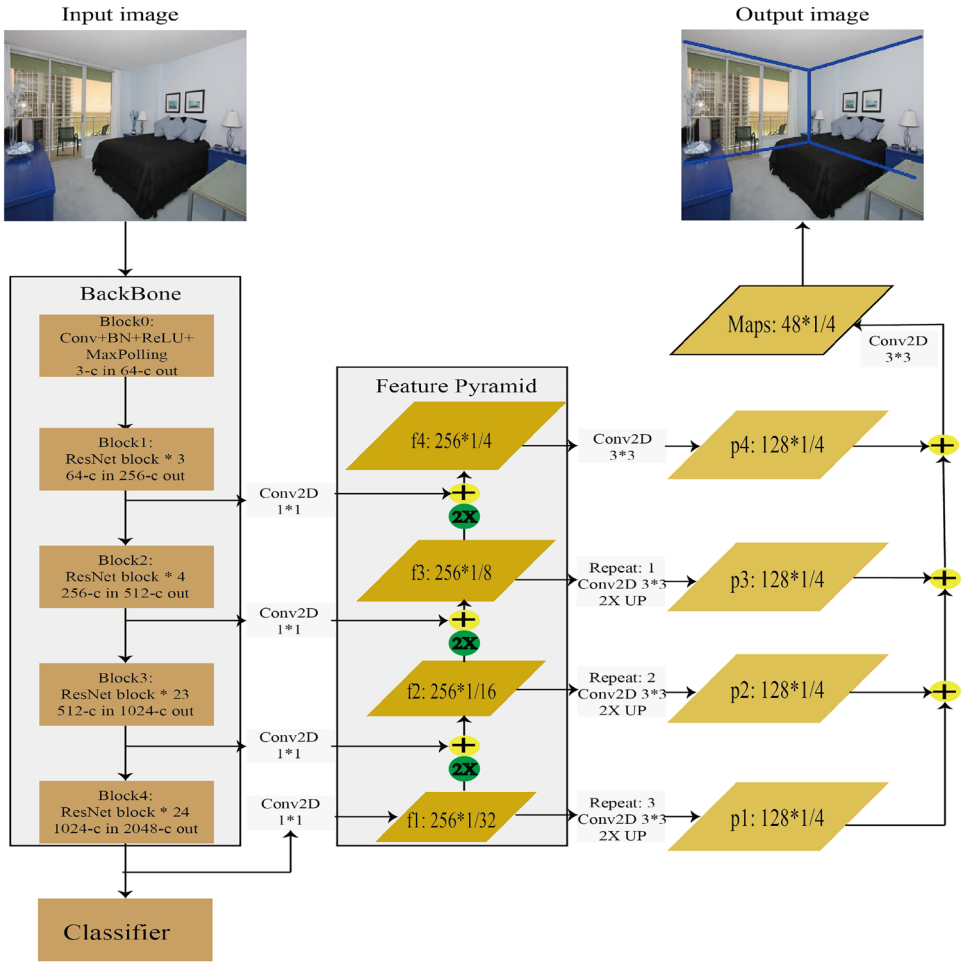

Fig. 2 Pipeline and the proposed network structure

Here, we choose 1/4 input resolution (80*80) as the final output size to save memory and reduce the network parameters and as demonstrated from many experimental tests of previous methods, it is sufficient enough to use 80*80 resolution feature maps for the layout estimation task.

3.1.2 Loss Function

The ground truth of each key point is a 2D Gaussian centred heatmap at 80*80 size with a standard deviation of 10 pixels. In this article, we follow the notation used in [14] as M-FPN shares the same output settings (Table 1). Different from the paper of [14], we use the focal loss [19] as the cost function for the room type prediction to alleviate the data imbalance problem in the LSUN dataset. The Euclidean loss is used as the cost function for key point heatmaps regression. The total loss function can be written as follows:

$$L(I, y, t) = \sum_{k} I_{k,t} \left\| G_k(y) - \varphi_k(I) \right\|^2 - \lambda \sum_{c} \mathbb{I}_{c,t} FL(\psi_c(I)) \tag{1}$$

The regressor φ produces an ordered set of 48-channel heatmaps that are ordered according to the room type order. Figure 1 shows the point order defined in the LSUN dataset. For example, the first eight channels correspond to the eight ordered points in room type 0 and the next six channels correspond to the six ordered points in room type 1 and so on. In Equation (1), $I_{k,t}$ is the Euclidean loss between the predicted heatmaps and the ground truth heatmaps if and only if key point k appears in ground truth room type t. Similarly, $\mathbb{I}_{c,t}$ is the focal loss if and only if room type index c is equal to the ground truth room type t. λ is used to balance the two loss items and is set to 1 in our model. Based on many conducted experimental tests, we set the hyper-parameter of Focal Loss $\alpha = 1, \gamma = 2$ respectively.

At the test stage, we use the predicted room type to select the corresponding set of key point heatmaps in the final output. We then find the maximum value's coordinates for each heatmap and scale the obtained coordinates back to their original resolution to get the final coordinates.

3.2 Preserving Points Order

Horizontal flipping is the commonly used data augmentation technique in this task. However, after this procedure, the order between key points will be messed and no prior work has discussed the result of this. Here, after the augmentation, we preserve the order of flipped points consistent wth the order of original points (Fig. 3). By doing this, each point in the flipped image stays in the relatively identical area to its original counterpart and model trained by these rectified images performs better. The reason can be explained as follows:

- There are 11 different room types defined in the LSUN dataset. As depicted in Fig. 1(top), each room type has a specific points ordering and structure. Since each point belongs to a relatively fixed location in the image, a coarser position of each point can be learned through the network.
- Preserving the order of flipped points being consistent with that of the original points can greatly reduce the training complexity of the network, enables the network to converge faster and achieves much better outputs.
- The order-preserving process reduces the semantic ambiguity on points. For example, the activation area of points at the symmetrical position can overlap a lot if we do not execute the order-preserving step. In this case, the network will learn two symmetrical activation areas in the image making false prediction rate doubled for each point.

Table 1 Notations

Symbol	Meaning
G	Gaussian function
(I, y, t)	Training sample of input image I: y is the ground truth coordinates of k key points; t is the room type
ψ	Room type classifier (output from the classifier in Fig. 2)
φ	Key point heatmap regressor (final 48-channel maps in Fig. 2)

We demonstrate the effectiveness of the order-preserving step in Fig. 3. The general practice of horizontal flipping is to simply flip the input image (including the points in it) as the way shown in the top middle of Fig. 3. This can disrupt the order among the points and enlarge the activation area of each point, resulting that the network model is more likely to produce bad predictions. For example, if we train our model as the conventional practices, our model will learn two symmetrical activation areas for point 1, the upper left and the upper right part of the ground truth image in Fig. 3, which are overlapped with the activation areas for point 7. This greatly increases the possibility of making wrong predictions. As can be seen in Fig. 3, the area of point 7 exists a bright light in the ground truth so that when we predict it using the model trained without the order-preserving step, we get the result (bottom middle image in Fig. 3) where point 7 is near point 1, both at the upper left position of the image, and a similar situation occurs between point 3 and point 5. In contrast, when we use the model trained with the order-preserving step, we get a good result (bottom

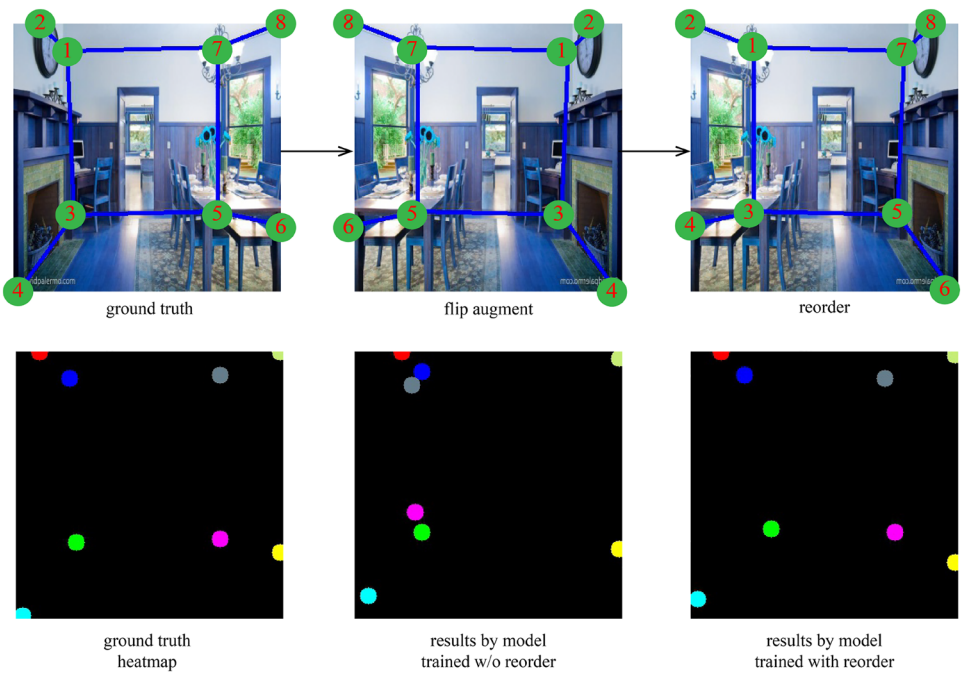

Fig. 3 Top row: process from ground truth to reordered result. Bottom row: impact of the reordering process on the prediction result through a control experiment

right image in Fig. 3). The reason is that during the training process we make the network learn a relatively individual activation area for each point so that the model can give the accurate predictions.

Overall, the order-preserving step is an essential step for improving the model's final accuracy and has not been used in the previous work.

3.3 Corner Refinement

To further improve the accuracy of outputs, we constrain the predicted corner point to its nearest image boundary. It should be noted that a corner point (i.e., points 2, 4, 6, 8 in room type 0) is a point that is supposed to stay at the image boundary(i.e., top, bottom, left, right). As shown in Fig. 8b, by doing this the segmentation map gets a lot of improvements. This phenomenon is related to the process of producing the segmentation map. To make things easier to understand, we use Fig. 4 to illustrate this process in a simplified way.

In Fig. 4, we use a 5*5 table to denote an image. Each blank space means a pixel. A and B are the corner points. When corner points are correctly predicted (like Fig. 4a), the algorithm will connect them by a line segment. After that, we can get two parts (connected components) from this image, and each part will be given a number respectively, representing different semantic label. However, this algorithm's result is sensitive to the position of corner points. Like Fig. 4b shows that A and B are predicted at the wrong position with only 1-pixel distance to its right position, but the image can not be correctly divided, producing a huge pixel error.

Due to this reason, we conduct the step called corner refinement. It will restrict each predicted corner point to its nearest border of the picture so that the algorithm can calculate the number of connected components and divide the image correctly. The impact of the corner refinement step will be discussed in Sect. 4.4 in details.

4 Experiments

4.1 Dataset and Cleaning

The current mainstream dataset includes the Hedau [4] dataset and LSUN (Large-scale Scene Understanding Challenge room layout) [20] dataset. The Hedau dataset, released in 2009, was sampled from the internet and LabelMe [3]. It contains 209 training pictures, 53 validation pictures and 105 test pictures. This dataset is relatively small and usually only used to compare pixel error. However, we could not find public resources on the internet. Therefore, we test our model on the LSUN dataset in this paper.

The LSUN dataset, sampled from the SUN [21] dataset, includes 4000 training pictures, 394 validation pictures and 1000 test pictures. Since the challenge has been closed, we cannot get the ground truth of the test set, so we use the validation set to evaluate our model, like many other methods [11, 14–16]. In all the experimental tests, we scale the model's outputs to the original resolution and evaluate the result by the LSUN toolkit as many other methods did.

As shown in Fig. 5, there are many wrong samples in the LSUN dataset. We need to clean the dataset before the formal experiments. In the room type 1 sample, the point order is reversed between both point tuple (2, 3) and point tuple (5, 6). Similarly, in the type 3 sample and type 9 sample, the order of point tuple (3, 4) and point tuple (1, 2) is reversed respectively. In addition, the problem of reversed order appears not only within the point tuple but also between different point tuples. For example, the point tuple (1, 2) and point tuple (3, 4) in the type 6 sample will sometimes be reversed too.

Apart from the point order problem, some room types can be mislabelled or the positions of some points can be incorrect. Since the number of such "dirty" samples is small, we simply discard them before the training process.

4.2 Implementation Details

Our experimental environment is a single 2080Ti GPU, CPU i9-9900K, and the operating system is ubuntu16.04. Training

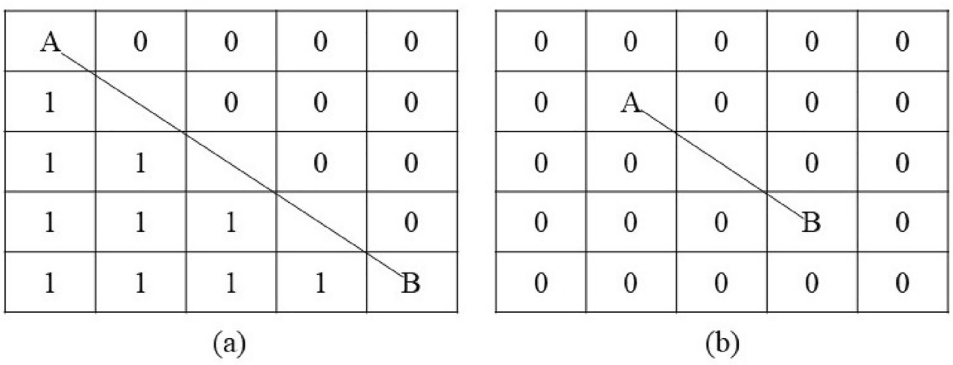

Fig. 4 The impact of corner point's position on the sematic label

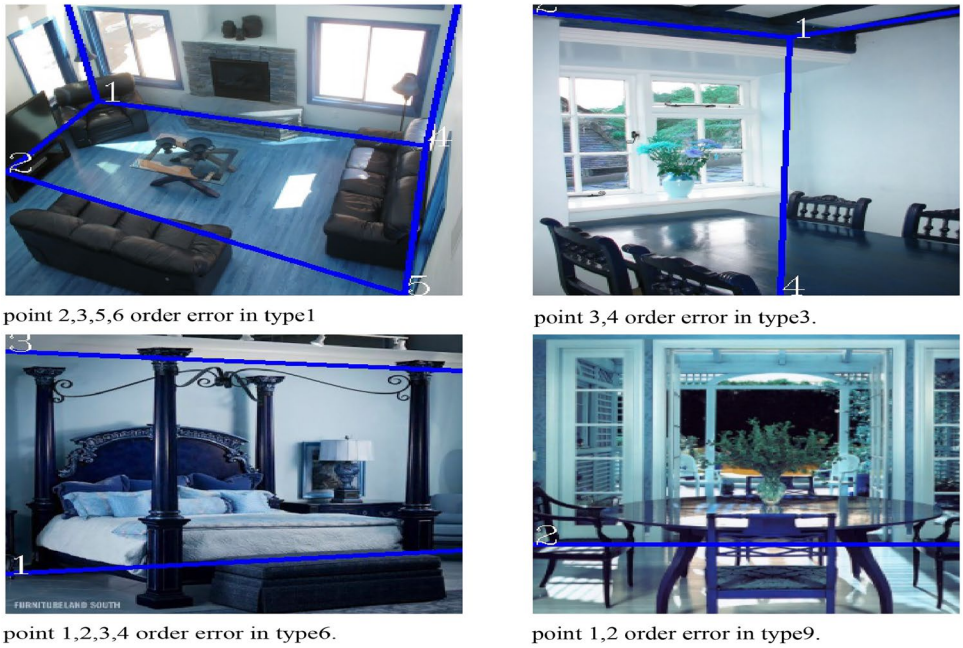

Fig. 5 Wrongly labelled samples in ground truth. The order of the points is incorrect, which appears in most bad samples

our model from scratch takes about 20 h on our machine. All input RGB images are resized to 320*320 and the outputs are the 48-channel key point heatmaps of resolution 80*80 and the labeled room types. The Adam optimizer [22] is employed to train the network for 150 epochs with batch size 8, weight decay 0.0005 and learning rate 0.00001. In addition, we use the ReduceLROnPlateau function to monitor the loss value in order to dynamically adjust the learning rate. The attenuation coefficient is set to 0.5, the patience value is 4, and the minimum learning rate is set to 1e−8. The backbone ResNeXt-101 [23] is set as the default with a group number of 32, group width of 8, and we initialize the backbone with weights pre-trained on ImageNet. The classifier consists of an average pooling layer, a dropout [18] layer and a fully connected layer, with a dropout rate of 0.5. In terms of data augmentation, apart from the horizontal flipping, we also use the ColorJitter function to produce small variations in image brightness, saturation, contrast and hue. We weaken the gradients of the background pixels in the key point heatmaps by multiplying the l2 loss at the background with a factor of 0.2 as did in [14] as the output of the network tends to converge to zero due to the imbalance between the foreground and background pixels. Our model is implemented in the open source deep learning framework PyTorch [24], version 1.1.0.

4.3 Testing Results

The metrics for the room layout estimation are pixel and key point errors (PE and KE) as defined in the LSUN toolkit. PE calculates the pixel-wise accuracy of the segmentation maps between the ground truth and the predicted segmentation. KE calculates the Euclidean distance between the predicted key points and ground truth points, normalized by the image diagonal length. Both PE and KE are averaged over all tested images.

In Table 2, we show the comparison of inference time and quantitative results on the LSUN dataset. Compared to

Table 2 Comparison of quantitative results and inference time

Method	Year	Post process	PE (%) / KE (%)	Time (ms)
Hedau, et al. [4]	2009	Yes	24.23 / 15.48	–
Mallya, et al. [9]	2015	Yes	16.71 / 11.02	–
Dasgupta, et al. [10]	2016	Yes	10.63 / 8.20	–
Ren, et al. [11]	2016	Yes	9.31 / 7.95	–
Zhao, et al. [25]	2017	Yes	5.29 / 3.84	–
RoomNet [14]	2017	**No**	9.86 / 6.30	166
LayoutNet [16]	2018	**No**	11.96 / 7.63	39
Edge Semantic [12]	2019	Yes	6.94 / 5.16	–
Double Refinement [13]	2019	Yes	**6.72 / 5.11**	–
Smart Hypothesis [15]	2020	**No**	**7.79** / 5.84	86
Our method	2020	**No**	**7.99 / 5.13**	32

Bold values indicate better results than other methods on LUN Dataset

PE pixel point errors, *KE* key point errors

other end-to-end methods, our method clearly outperforms all other methods on both error metrics except the 'Smart Hypothesis', which advantages our method slightly 0.2% in PE. In contrast, our method outperforms it by 0.71% in KE. In fact, our method has reached the level of state-of-the-art two-step method of Double Refinement [13] in KE.

It should be noted that [25] is a transfer learning method which requires a large amount of training data far beyond the LSUN dataset for pre-training and then fine-tunes the model on the LSUN dataset. Therefore, it is incomparable with other methods in our designed environment.

In terms of inference time among the end-to-end algorithms, our model takes only 32 ms, the fastest processing speed among the methods in a single NVIDIA Titan X GPU. It is about 31 fps and suitable for real-time application. The time costs of two-step methods are not listed in Table 2 as these methods usually require from several seconds to several minutes for predictions, which are too high to compare with our method.

Figure 6 shows the qualitative results of our model on the LSUN validation set. It is not hard to see that in a daily indoor environment, our network can effectively estimate the layout for different types of rooms. It can be seen from samples (a), (b) and (e) that our network has strong robustness to occlusion for any input room types. In sample (a), the key points are covered by the bed. In samples (b) and (e), the key points are obscured by chairs. Both sample (c) and sample (d) have only two key points that can be identified easily from a human perspective, but the hard thing is how to select the specific tiny key point area from the many potential areas in the picture. From the results, we can see that our model performs well.

Here we want to specifically mention sample (e). This sample belongs to room type 1, but two points are missed in its ground truth label, which can be distinguished from the semantic segmentation map, and the line at the top slightly exceeds the boundary of the picture and beyond the field of view. Even though, in this situation, our model can still accurately predict the layout of the room, demonstrating that our model is very capable of learning the patterns from the dataset.

Figure 7 shows some representative failure cases. In case (a), there is a misleading oblique line in the house structure, which does not fit most cube-structured houses and makes the network mistakenly regard this oblique line as the final prediction result. Especially when there are multiple such structural lines in the house, it will be difficult for the model to make correct judgments. We call the representative situation in case (b) the network using "imagination". Since some areas in the picture, such as corners, have a higher

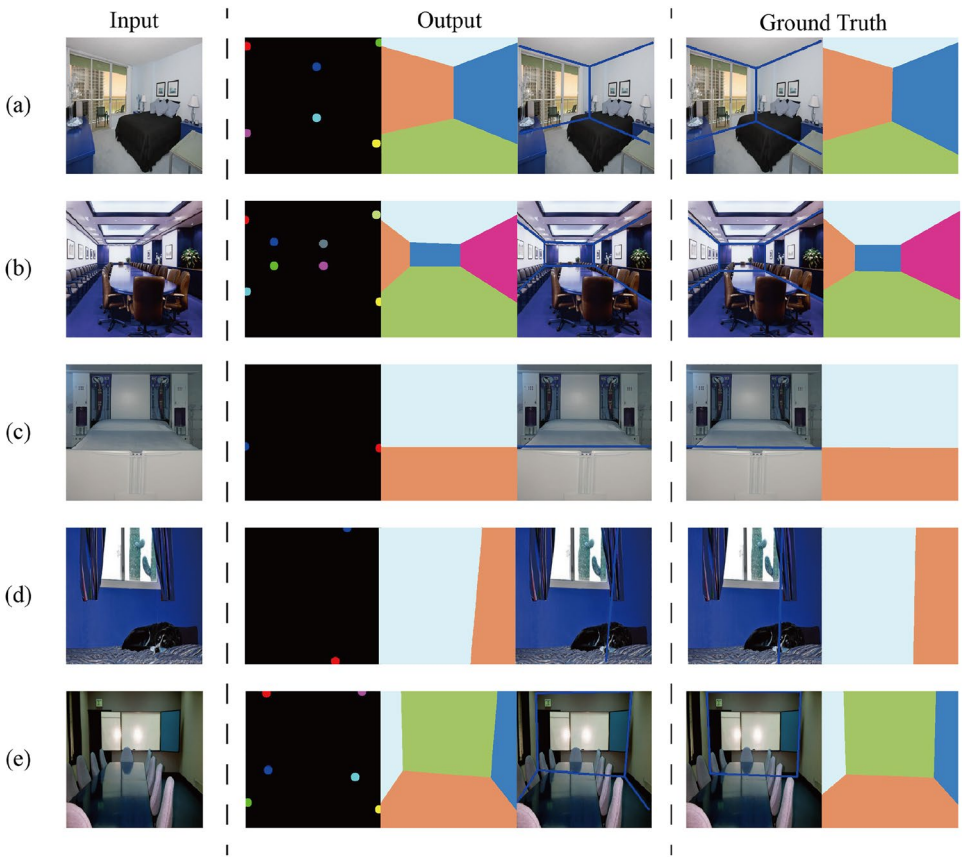

Fig. 6 Some qualitative results. The first column is the input image, the middle 3 columns are the key point heatmaps, semantic segmentation map, and room layout results in sequence produced by the network, and the last two columns are ground truth

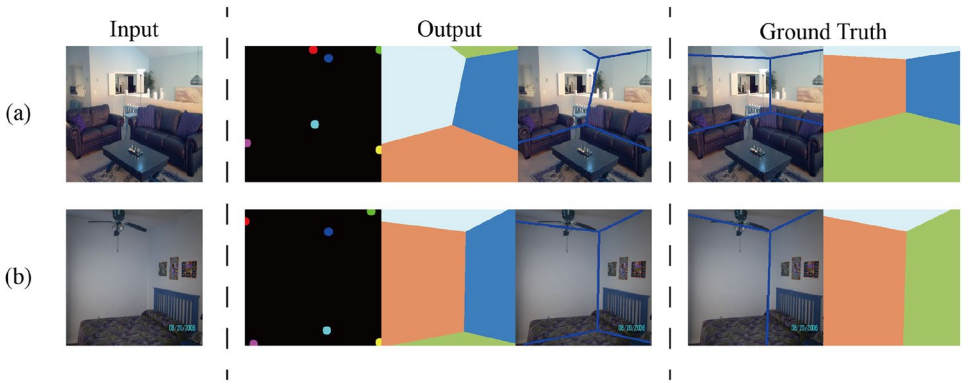

Fig. 7 Some failure cases. In case (a), the oblique line structure on the upper left of the house is mistaken by the network as the room layout. In case (b), the network uses its "imagination" to predict two key points from the occlusion, which turns case (b) from type 3 to type 5

Table 3 Ablation experiment of network configuration

Set	Backbone	fc	FC	Multi-Branch	KE (%)	PE (%)
(a)	Res-50	✓			11.44	19.89
(b)	Res-50		✓		12.49	20.74
(c)	Res-101	✓			11.07	19.33
(d)	Res-101	✓		✓	10.91	19.86

Table 4 Ablation experiment of data-clean, reorder, corner refine

Set	Data clean	Reorder	Corner refine	KE (%)	PE (%)
(a)	✓	✓	✓	5.13	7.99
(b)	✓		✓	6.72	10.98
(c)	✓	✓		5.27	47.79
(d)			✓	7.23	12.39

probability of being key points, when these areas are blocked and the classifier predicts a wrong room type, the network intends to use "imagination" to predict possible key points in these areas matching the room type that accounts for a higher proportion in the dataset.

4.4 Ablation Experiment

We conduct ablation experiments to figure out a better network configuration in Table 3, and to illustrate the effectiveness of the various processing steps we have proposed in Table 4.

Table 3 shows 4 neural networks with different configuration. (a): a ResNet-50 as backbone, a fully connected(fc) layer as classifier, three deconvolution layers to output key point heatmaps; (b): the same with (a), except that the classifier is replaced by a 1*1 Fully Convolutional(FC) layer; (c): the same with (a), except that the backbone is replaced by ResNet-101; (d): Add another identical head-to-output segmentation maps on the basis of (c).

From the comparison of (a) and (b), we can see that the network has better performance on both KE and PE when a fully connected layer is used as classifier. One possible reason for this result is that a fully connected layer has more parameters than a fully convolution layer so that it has bigger feature space to catch the differences between different types of room layout.

Using a stronger backbone as (c) does improve the model's performance. This is because a stronger backbone can extract high-level features with rich semantic information, and semantic information plays a vital role in room layout estimation task. Based on this fact, we use ResNeXt-101 as backbone of M-FPN, which also elevates the prediction accuracy.

We add another parallel head to predict the segmentation maps in (d) on the basis of (c) trying to test wether, the multi-branch network has a better performance. The result is neither better nor worse. The model has lower KE (from 11.07 to 10.91%) and higher PE (from 19.33 to 19.86%). However, considering that a multi-branch network needs more inference time and parameters, it is not a good choice to use a multi-branch network, especially when you can not make the parallel heads work together in a united loss function.

In Table 4, the importance of order-preserving step can be seen from the comparison of (a) and (b): Without this step, KE increases from 5.13 to 6.72%, and PE increases from 7.99 to 10.98%. As indicated in Fig. 3, the activation area can overlap a lot between the symmetrical key points if the order-preserving step is not performed. Therefore, when one of the symmetrical key points is severely obscured or in other similar cases, the network will predict the original symmetrical key points in a different area with a high probability.

From (b) and (d), KE increases from 6.72 to 7.23%, and PE increases from 10.98 to 12.39% when the data clean is not performed. The reason is that an incorrectly labelled order of points in the dataset will cause the model to learn the wrong activation area for each point and affect the results' accuracy.

By comparing (a) and (c), it can be found that KE has changed 0.14% but PE has increased from 7.99 to 47.79%. This is related to the process of calculating the semantic segmentation map from the key point maps in the LSUN toolkit as explained in Sect. 3.3. Ideally, the connectivity of 4 pixels restricts the predicted point coordinates to be strictly at the boundary of the image during the process to find connected components. But in fact, as shown in Fig. 8 case (a), although the predicted corner point is already very close to the boundary of the picture, there can still have an error of about tens of pixels when the corner point coordinates are scaled to the original resolution. Therefore, the situation in Fig. 8b can occur. After the points are connected, the algorithm can miscalculate the number of connected components. Thus, it is a very necessary step to constrain the corner points and move them to the picture boundaries. The time cost of this step is almost negligible.

To illustrate the effectiveness of the proposed optimization steps, we run these steps on a re-implementation of RoomNet(basic), and the result is shown in Table 5. By comparing (b) and (d), it is obvious that order-preserving step works well on the RoomNet. We get both lower KE (from 7.39 to 6.42%) and PE (from 12.53 to 9.61%). From (c) and (d), the PE is reduced from 46.2 to 9.61%, which shows that the corner refinement step works well on RoomNet too. After running these optimization steps, the RoomNet(re-imp.) gets better performance than the original RoomNet(basic) and is comparable to the best result of RoomNet in Table 2, which can be seen in (a) and (d). From the experiment results, we would like to say that the proposed steps can be helpful for other similar methods too.

Table 5 Ablation experiment of optimization steps on RoomNet

Set	Network	Reorder	Corner refine	KE (%)	PE (%)
(a)	RoomNet (basic)			6.95	10.46
(b)	RoomNet (re-imp.)		✓	7.39	12.53
(c)	RoomNet (re-imp.)	✓		6.42	46.20
(d)	RoomNet (re-imp.)	✓	✓	6.42	9.61

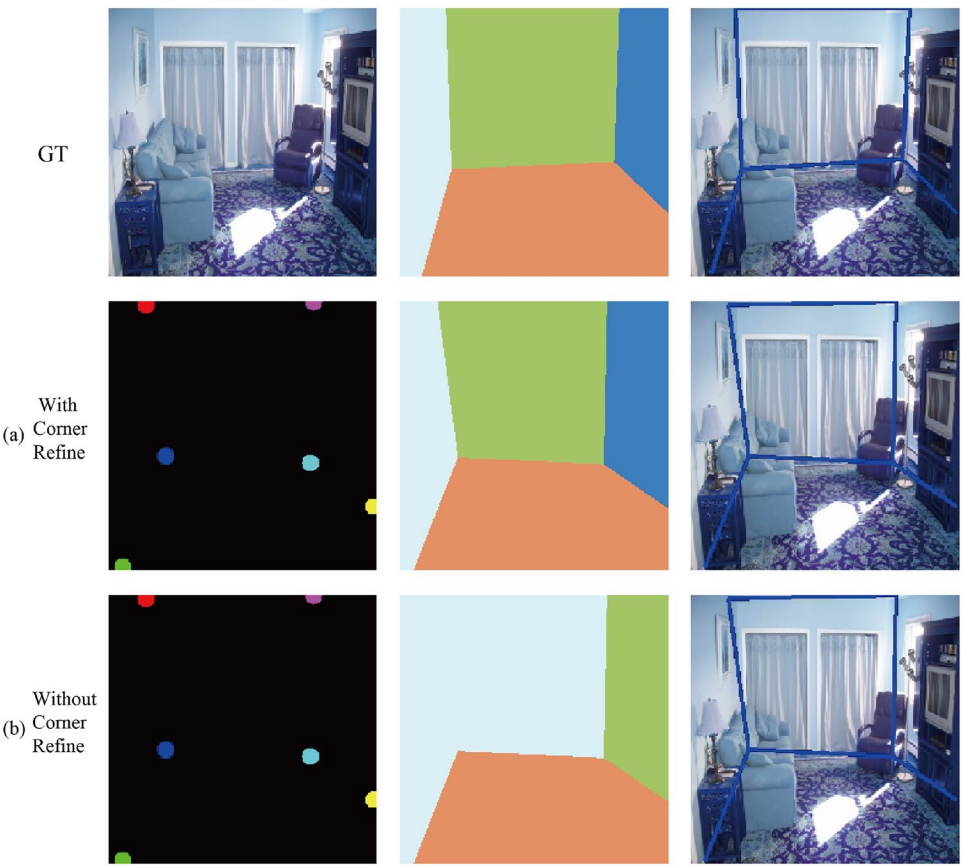

Fig. 8 The first row is the input image and ground truth. Case (**a**) and case (**b**) respectively show the prediction result with and without the corner refine step

5 Conclusion

In this paper, we have proposed M-FPN as an end-to-end model for indoor layout estimation. With a processing speed up to 31 fps, M-FPN has reached the level of the state-of-the-art two-step method on the KE metric with the fastest speed. Compared with the most advanced end-to-end method, there is only a slight 0.2% gap in the PE metric. This is due to a series of operations that we perform on the dataset, in addition to our network's superior capability of integrating multi-scale features. Among them, keeping the consistency of each key point's activation area and applying the corner refinement step plays a vital role. The performance on the LSUN dataset and the ablation experiments have demonstrated the superiority of our model. In the future, we intend to use a single network to predict key point maps and semantic segmentation maps at the same time, so as to further achieve better results.

Acknowledgements The authors would like to thank the data providers of [20] for the testing data sets. This work was partially supported by the Ningbo Science and Technology Special Projects (No. 2021Z019 and 2022Z095), and the Hebei "One Hundred Plan" Project (No. E2012100006), and National Talent Program (No. G20200218015).

Authors' contributions All the authors listed in this manuscript have made substantial contributions to this article and signed appropriately.

Declarations

Competing interests There is no conflict of interest in this manuscript.

Ethics approval Not applicable.

Consent to participate All authors agreed to participate

Consent for publication Not applicable.

References

1. Lin T, Dollár P, Girshick RB, He K, Hariharan B, Belongie SJ (2017) Feature pyramid networks for object detection. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, pp 936–944
2. Kirillov A, Girshick RB, He K, Dollár P (2019) Panoptic feature pyramid networks. In: IEEE conference on computer vision and pattern recognition, CVPR 2019, pp 6399–6408
3. Russell BC, Torralba A, Murphy KP, Freeman WT (2008) Labelme: a database and web-based tool for image annotation. Int J Comput Vis 77(1–3):157–173
4. Hedau V, Hoiem D, Forsyth DA (2009) Recovering the spatial layout of cluttered rooms. In: IEEE 12th international conference on computer vision, ICCV 2009, pp 1849–1856
5. Lee DC, Gupta A, Hebert M, Kanade T (2010) Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In: Proceedings of the 23rd international conference on neural information processing systems-volume 1, pp 1288–1296
6. Hedau V, Hoiem D, Forsyth DA (2012) Recovering free space of indoor scenes from a single image. In: 2012 IEEE conference on computer vision and pattern recognition, pp 2807–2814
7. Zhang J, Kan C, Schwing AG, Urtasun R (2013) Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In: IEEE international conference on computer vision, ICCV 2013, pp 1273–1280
8. Ramalingam S, Pillai JK, Jain A, Taguchi Y (2013) Manhattan junction catalogue for spatial reasoning of indoor scenes. In: 2013 IEEE conference on computer vision and pattern recognition, pp 3065–3072
9. Mallya A, Lazebnik S (2015) Learning informative edge maps for indoor scene layout prediction. In: 2015 IEEE international conference on computer vision, ICCV 2015, pp 936–944
10. Dasgupta S, Fang K, Chen K, Savarese S (2016) Delay: robust spatial layout estimation for cluttered indoor scenes. In: 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, pp 616–624
11. Ren Y, Li S, Chen C, Kuo CJ (2016) A coarse-to-fine indoor layout estimation (CFILE) method. In: Lai S, Lepetit V, Nishino K, Sato Y (eds) Computer vision—ACCV 2016—13th Asian conference on computer vision, revised selected papers, Part V. Lecture notes in computer science, vol 10115, pp 36–51
12. Zhang W, Zhang W, Gu J (2019) Edge-semantic learning strategy for layout estimation in indoor environment. CoRR arXiv:1901.00621
13. Kruzhilov I, Romanov M, Babichev D, Konushin A (2019) Double refinement network for room layout estimation. In: Palaiahnakote S, di Baja GS, Wang L, Yan WQ (eds) Pattern recognition—5th Asian Conference, ACPR 2019, Revised selected papers, Part I. Lecture notes in computer science, vol 12046, pp 557–568
14. Lee C, Badrinarayanan V, Malisiewicz T, Rabinovich A (2017) Roomnet: end-to-end room layout estimation. In: IEEE international conference on computer vision, ICCV 2017, pp 4875–4884
15. Hirzer M, Roth PM, Lepetit V (2020) Smart hypothesis generation for efficient and robust room layout estimation. In: IEEE winter conference on applications of computer vision, WACV 2020, pp 2901–2909
16. Zou C, Colburn A, Shan Q, Hoiem D (2018) Layoutnet: reconstructing the 3d room layout from a single RGB image. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, pp 2051–2059
17. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep convolutional encoder–decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
18. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

19. Lin T, Goyal P, Girshick RB, He K, Dollár P (2017) Focal loss for dense object detection. In: IEEE international conference on computer vision, ICCV 2017, pp 2999–3007
20. Zhang Y, Yu F, Song S, Xu P, Seff A, Xiao J (2015) Large-scale scene understanding challenge: room layout estimation. In: CVPR Workshop
21. Xiao J, Hays J, Ehinger KA, Oliva A, Torralba A (2010) SUN database: large-scale scene recognition from abbey to zoo. In: The twenty-third IEEE conference on computer vision and pattern recognition, CVPR 2010, pp 3485–3492
22. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Bengio Y, LeCun Y (eds) Proceedings of 3rd international conference on learning representations, ICLR 2015
23. Xie S, Girshick RB, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, pp 5987–5995
24. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32: annual conference on neural information processing systems 2019, NeurIPS 2019, pp 8024–8035
25. Zhao H, Lu M, Yao A, Guo Y, Chen Y, Zhang L (2017) Physics inspired optimization on semantic transfer features: an alternative method for room layout estimation. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, pp 870–878