4 Deep convolutional neural networks for Floor plan Analysis and Retrieval

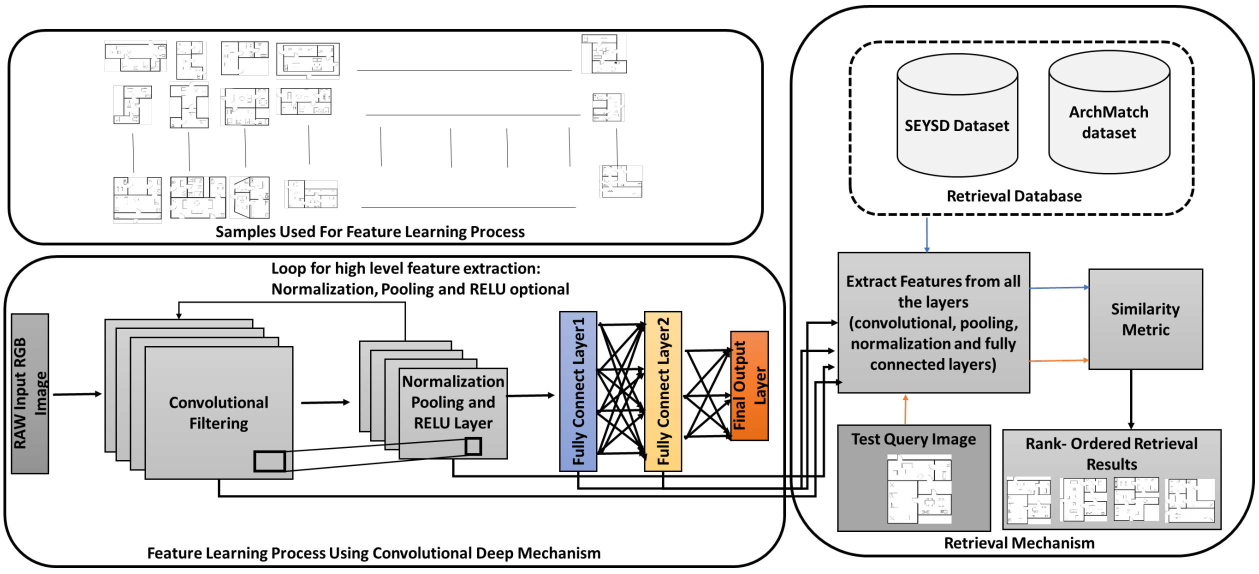
In Chapter 3, a baseline retrieval framework was explored by extracting topological features for understanding the layout in a floor plan. Additionally the decor features were extracted, to understand the room semantics better. During retrieval, these two features are combined and the floor plans are rank ordered based on the feature similarity. For efficient and accurate characterization of a given floor plan, as well as coming up with a feature representation that is invariant to various perturbation, feature engineering is a key component. Broadly the feature engineering approaches are classified into two categories, namely: hand-crafted features and machine learned features (deep features). The features discussed in Chapter 3, are hand-crafted. For hand-crafted features the pipeline for feature extraction is designed such that it is particularly suitable for the underlying application, and sufficient to meet the requirements. These features are robust and give satisfactory retrieval performance. However, the hand-crafted features have the following deficiencies:

1. The hand-crafted features mostly reflect the low-level identities of the objects present in the scene or the image under consideration. As a result these features are not directly suitable for representing the mid-level or even high-level depiction of the objects and require further processing or integration with additional knowledge.

2. During the feature extraction process the image level information is ignored, i.e. the features extracted are independent of the content present in the image.

This limits the matching algorithm to identify the similarities between two given floor plans. Recently, Deep Neural Networks (DNNs) have demonstrated their great power in computer vision and have achieved better performance for many tasks. For example, Convolutional Neural Networks (CNNs) have achieved good performance for ImageNet image classification [Russakovsky *et al.*, 2015]. The CNNs automatically learn the image representation from the images. The image labels are used to make the image representations separable. Such CNN learned features are termed as the machine learned features. These machine learned features capture the distribution information about the underlying data, and such information is especially important to the image representation. Hence, in this Chapter a deep learning framework is proposed for the analysis and retrieval of similar floor plans, with query by example paradigm, where the queries are floor plan images.

The rest of the chapter is organized as follows: Section 4.1, gives a brief description of the proposed technique. Section 4.2 and 4.3 give an insight into the proposed deep architecture for floor plan feature extraction and retrieval. Section 4.4 analyses the qualitative and quantitative results obtained through the extensive experiments that were carried out. Finally, Sec. 4.5 concludes the Chapter.



Samples Used For Feature Learning Process
SEYSD Dataset
ArchMatch dataset
Retrieval Database
Loop for high level feature extraction: Normalization, Pooling and RELU optional
RAW Input RGB Image
Convolutional Filtering
Normalization Pooling and RELU Layer
Fully Connect Layer1
Fully Connect Layer2
Final Output Layer
Extract Features from all the layers (convolutional, pooling, normalization and fully connected layers)
Similarity Metric
Test Query Image
Rank- Ordered Retrieval Results
Feature Learning Process Using Convolutional Deep Mechanism
Retrieval Mechanism

Figure 4.1. : Deep Learning Model based Framework for Floor Plan Retrieval

4.1 BRIEF OVERVIEW

Figure 4.1 depicts the complete framework of the proposed deep learning technique. A brief overview of the proposed framework for similar floor plan retrieval is as follows. The entire framework has been divided into two main phases. They are: (1) deep feature representation; and (2) matching and retrieval. In the deep feature representation task, deep neural network layers like Convolution, Normalization, Pooling and ReLU are used. In this work an additional Normalization layer is used which basically is useful when ReLU neurons are being dealt with. This is due to the fact that ReLU neurons have unbounded output and thus need to be normalized. If there is a normalization around the local neighborhood of the excited neuron, it becomes even more sensitive as compared to its neighbors. At the same time, it dampens the responses that are uniformly large in any given local neighborhood. If all the values are large, then normalizing those values diminishes all of them. Therefore, the normalization layer encourages some kind of inhibition and boosts the neurons with relatively larger activations. The output of all the layers mentioned above is passed through a network of fully connected layers to finally give the feature vector for an input image. The framework first learns the deep feature representation for floor plan retrieval task using a few samples from the ROBIN dataset. The learned deep representation helps to extract deep features from the samples of ROBIN, and stores it in the feature database. During the retrieval stage, the framework extracts the deep features from query image also using the same deep representation, where query image is both from the SESYD and the ROBIN dataset. The extracted deep features are then matched with the deep features in the database. A similarity score for a particular layout is thus calculated for all the query samples. To understand the effectiveness of the individual layers or feature representation of the deep learning network, the similarity is calculated using the features at the outputs of all the layers.

In this Chapter, the goal is to come up with new effective deep learning based feature representation and to investigate the effectiveness of learned features on floor plan retrieval tasks. In particular, two open issues are addressed:

- How to learn a new efficient deep learning feature representation for floor plan retrieval task?

- How the individual deep feature layers affect the performance of the retrieval system?

The proposed architecture has two main components, (1) Feature Representation and (2) Matching and Retrieval, which are discussed in detail in the subsequent subsections:

4.2 FEATURE REPRESENTATION

Traditionally, feature extractors are manually engineered and optimised through a laborious trial-and-error cycle involving re-learning the classifiers. In this work, the representation is learned using a CNN instead, by jointly optimising the performance of the features, as well as, of the classifiers. Existing literature also validates that deep features are a better way to represent an image. However, it is also shown that improper training of deep layers may lead to over-fitting or under-fitting of a model which can be a disadvantage.

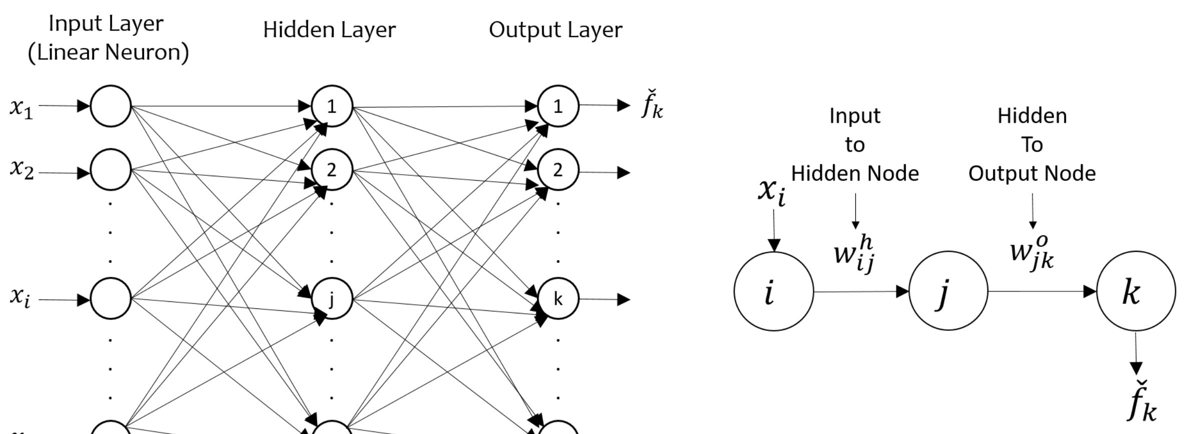
Convolutional Neural Networks (CNNs) have shown promising results for various computer vision tasks like classification and detection. Convolutional neural networks are similar to feed-forward neural networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and generally follows it with a non-linearity. The entire network expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other end. There is a loss function on the final fully-connected layer. The difference that convolutional neural networks have from feed-forward networks is that, these architectures make the explicit assumption that the inputs are images. This assumption then leads to a vast reduction in the number of parameters in the network. Convolutional neural networks have a variety of layers serving different purposes. Using the approach of CNNs, many classification related tasks have been done in the past. It was observed that researchers have not extensively explored the CNNs for floor plan retrieval tasks. The work proposed in this Chapter uses a combination of convolution, pooling, normalization, ReLU, and fully connected layers along with dropout regularization technique (see Fig. 4.1). Before going into the detailed description of the proposed architecture, in the next sub-section, for the sake of completeness, a brief description of multilayer feed forward neural network and its relation to CNNs is presented.

4.2.1 Multilayer feed-forward neural network Architecture

Multilayer feed-forward neural networks (MLFFNN) receive an input (a single vector), and transform it through a series of hidden layers. Figure 4.2 depicts an architecture of a typical MLFFNN. Each hidden layer has a number of neurons. Each neuron is connected to all neurons in the previous layer (Fig. 4.2(a)). Neurons in a single layer do not share any connections. The final fully-connected layer called the output layer represents the class scores. Let, the input feature vector $\bar{x}$ be of d dimension and there are k output classes (in a k-class classification problem). According to the Universal Approximation theorem, if $\Phi(.)$ is a non-constant, bounded, monotonically increasing continuous function, then there exists an integer J such that:

$$\tilde{f}(\bar{x}) = \sum_{j=1}^{J} \alpha_j \Phi_j \left(\sum_{i=1}^{d} w_{ij} x_i + w_{jo} \right) \tag{4.1}$$

is an approximate realization of the desired function $f(\bar{x})$. Here, J is the number of neurons in the hidden layer, and w_{ij}s are the synaptic weights between the pre-synaptic neuron i, and post-synaptic neuron j. The theorem guarantees that we will be able to design an artificial neural network with J



(a) Illustration of a typical multilayer feed-forward neural network

(b) Illustration of a connection between pre- and post-synaptic neurons, interconnected with synaptic weights

Figure 4.2. : Architecture of feed-forward neural network, with the input layer, a hidden layer and the output layer

hidden nodes and synaptic connections, which will be able to represent the input feature vector to the best possible extent. In Fig. 4.2(b), the synaptic weights between input to hidden and hidden to output layer are explicitly shown. To determine the class label for a given test sample, "winner takes all" decision logic is followed.

In case of MLFFNN, the input vector dimension plays an important role in the scalability of the entire solution. An increase in the value of d leads to increase in the number of synopsis, and hence the synaptic weights. As a result, there are huge number of parameters to be learned. For example, a given image of size, say, $32 \times 32 \times 3$, requires 3072 weights for each neuron. For a bigger image that is 200×200, each neuron in the first layer would require 120000 parameters. This network with a huge number of parameters, would not only take long to train, but may also be prone to overfitting.

On the other hand, the convolutional neural network has its origin deeply rooted at the seminal work by Hubel and Wiesel (Nobel prize awarded in the 1980's). Hubel and Wiesel's "simple" and "complex" cell's computational properties were later well-described by linear models and rectifiers. The key components incorporated in convolutional neural networks, which are similar to the various components of the visual cortex are:

- CNNs are spatially organized, so that nearby cells act on nearby parts of the input image (similar to the Retinotopic map)
- CNNs use spatially localized linear filters, which are followed by thresholding (resembling the simple cells)
- CNNs use pooling units to incorporate invariance to shifts of the position of the feature (mimics the complex cells)

Some recent work has shown that artificial neurons in convolutional neural networks have similar hierarchical structure to the visual pathway. CNN (refer Fig. 4.3) takes advantage of the fact

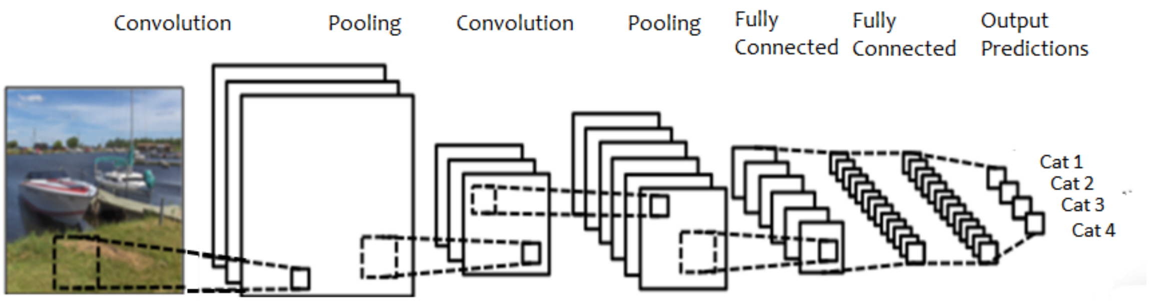


Figure 4.3. : Architecture of a Convolutional Neural Network Model for classification of an image [LeCun *et al.*, 2015]

that the input are images and they constrain the architecture. In particular, the layers of a CNN have neurons arranged in 3 dimensions: width, height and depth, where, depth refers to the third dimension of an activation volume.

4.2.2 Proposed CNN architecture

The Convolutional Neural Network (CNN) proposed in this Chapter for the classification of floor plans has an architecture comprising of an Input layer, a Convolution layer, a ReLU layer, a Pooling layer, a Normalization layer and a Fully Connected layer. Details of each layer are as follows:

- The Input layer holds the raw pixel values of the floor plan image.
- Convolutional layer computes the output of neurons that are connected to local regions in the input, each neuron computing a dot product (weighted sum) between its weights and a small region it is connected to in the input image. Figure 4.4 depicts an example of such a weighted sum between an input image and a filter. The convolutional layer defines a collection of filters (or activation maps), each with the same dimension as the input. Let, the input be of dimensionality (w,h,d), and the filter dimensionality be (w_f,h_f,d), typically $w_f < w$. The depths being equal means that the output of this convolution operation is 2D. The output slices of the convolution operations with each filter are composed together to form a $(w+w_f-1,h+h_f-1,n_f)$ tensor, where n_f is the number of filters. A convolutional layer

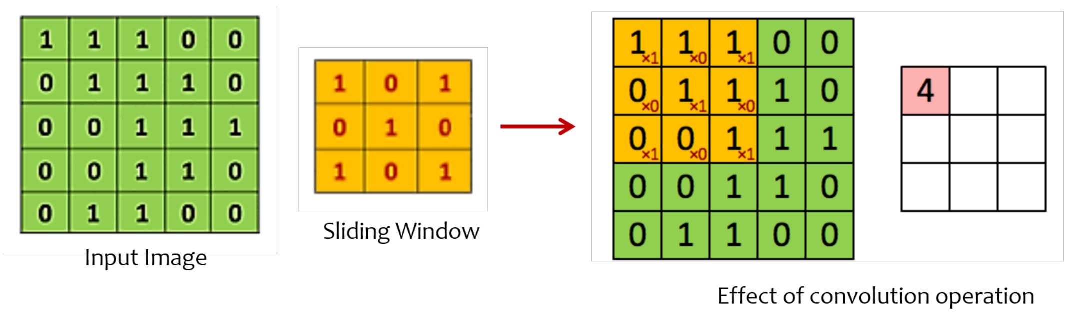
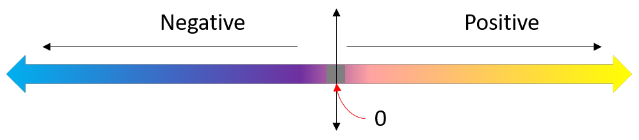
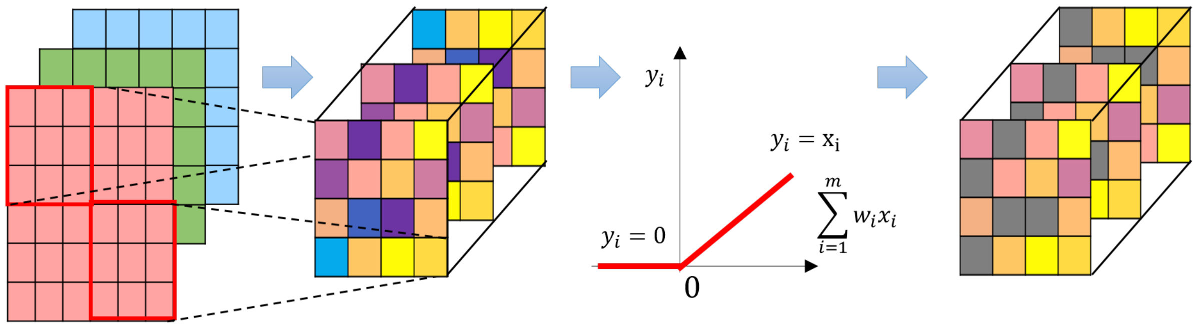


Figure 4.4. : Example of how convolution operation is performed on an image matrix.



(a) Color-bar indicating range of activation values (positive and negative). The zero is indicated by "red" arrow.



(b) Input Image (c) Feature Maps generated after convolution (d) Rectifier $y = \max(0, x)$ (e) Feature Maps after Rectification

Figure 4.5. : Example of how a ReLU Operation works.

finally learns a set of n_f filters, $Filt = \{filt_1, filt_2....filt_{n_f}\}$ with input floor plan image I to produce a set of 2D feature maps:

$$z_k = filt_k \circledast I \tag{4.2}$$

where $\circledast$ is the convolution operator. Sharing of weights is done over the entire image which in turn results in the reduction of the number of parameters. Also, if there are m inputs and n outputs in a hidden layer, a MLFFNN would require $O(mn)$ operations to compute the output. Whereas, in CNN each output is connected to only $k = w_f h_f d$ inputs, and thus requires $O(kn)$ operations to compute the output, thereby reducing the computation time.

- In the network proposed in this Chapter, ReLU (Rectified Linear Unit) layer is used to induce non-linearity. The ReLU clips negative values to zero while keeping positve values unchanged. Figure 4.5 depicts how ReLU works. The popularity of ReLU is due to the fact that it highly enhances the acceleration of the convergence of stochastic gradient descent as opposed to the sigmoid/tanh functions due to its linear non-saturating form. It also avoids and rectifies vanishing gradient problem [Krizhevsky *et al.*, 2012b]. This is due to the fact that as seen in Eq. 4.3 the gradient maintains a constant value. An added advantage is that it is computationally less expensive. ReLU has the following mathematical form:

$$y = max(0, x) \tag{4.3}$$

It applies an element-wise activation function, in the form of $max(0, x)$ with thresholding at zero which leaves the size of the volume unchanged.

- In the proposed architecture, a pooling layer is used for reducing the size of the feature maps and to provide invariance between floor plan images with very little differences. Max pooling is usually preferred as it prevents blurring of the activations and gradients throughout the network because the gradient is placed in a single location during back propagation. It also avoids cancellation of negative elements (refer Fig. 4.6) [Cadène *et al.*, 2016; Yin *et al.*, 2016]. Pooling layer performs a down-sampling operation along the spatial dimensions.

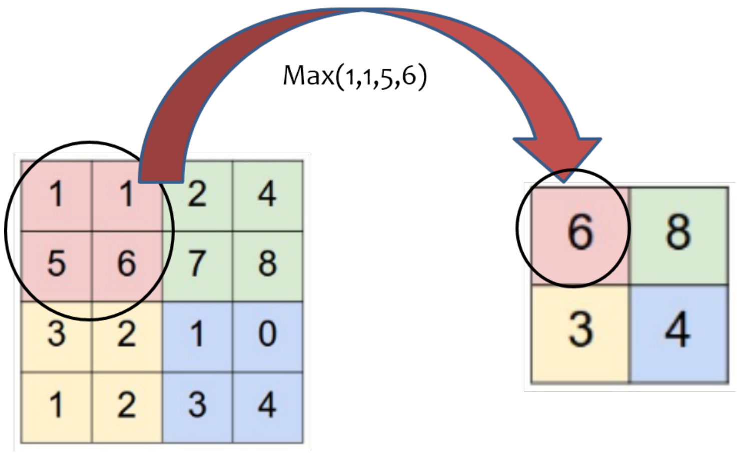


Figure 4.6. : Example of how a Max-Pooling Operation behaves on an image matrix.

- The next layer in the proposed architecture is the Batch normalization layer, which is widely popular as it converges faster [Ioffe and Szegedy, 2015]. The Batch normalization layer leads to an addition of a normalization step which helps in shifting the inputs to zero-mean and unit variance [Cadène *et al.*, 2016]. It allows activation functions to not get stuck in the saturation mode (gradient equal to 0).
- The fully-connected layer computes the class scores. Classification into categories is done by this layer. Neurons in a fully connected layer have connections to all activations in the previous layer.

In this way, the proposed Convolutional Neural Network transforms the original floor plan image, layer by layer from the original pixel values to the final class scores. In particular, the Convolutional/Fully Connected layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/Pooling layers implement a fixed function. The parameters in the Convolutional/Fully Connected layers are trained with gradient descent so that the class scores that the Convolutional Neural Network computes are consistent with the labels in the training set for each image.

This proposed architecture takes inspiration from the methodology described in the work AlexNet [Krizhevsky *et al.*, 2012a], to obtain an effective feature representation from floor plan images (refer Fig. 4.7).

The basic highlights of the AlexNet framework [Krizhevsky *et al.*, 2012a] are:

- ReLU is used instead of Tanh to add non-linearity. This layer accelerates the speed by 6 times at the same accuracy for classification tasks.
- Dropout is used instead of regularisation to deal with overfitting.
- Pooling is overlapped to reduce the size of network. It reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively.
  The AlexNet framework contains 5 convolutional layers and 3 fully connected layers. ReLU is applied after every convolutional and fully connected layer. Dropout is applied before the first and the second fully connected layer. Through Dropout, at each training stage, individual nodes are dropped out of the net to ensure that there is no overfitting in the network.

However, the model proposed in this Chapter is different from AlexNet in terms of two

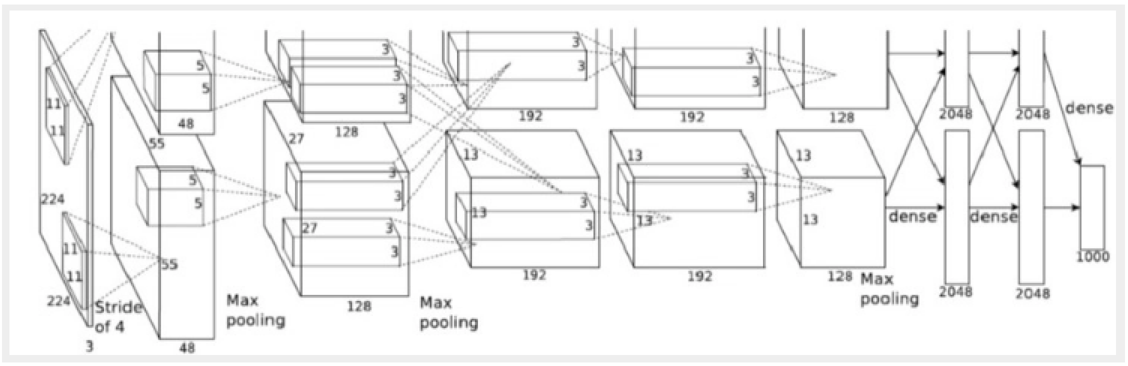


Figure 4.7. : Sequence of layers in AlexNet Framework [Krizhevsky *et al.*, 2012a]

properties:

- The network is not trained with data-augmentation.
- The order of pooling and normalization layers is switched (pooling is done before normalization) [Jia *et al.*, 2014].

The values of the parameters used in the framework such as, number of layers, type of layers, channels, filter sizes, convolution strides, pooling size, pooling stride and the padding size are listed out in Tab. 4.1. The parameters left blank in the table are the ones which do not have any role to play in that particular layer.

As an example taking the first layer, which is a combination of a convolution layer, a maxpool layer and a normalization layer. The size of the output image after passing through a convolution layer can be computed as:

Let, O_c = Size of the output image, I_c = Size of input image, K_c = Size of channels used in the convolution layer, S_c = Stride of the convolution operation and P_c = Padding Size. The size of the output image O_c is calculated using the equation below:

$$O_c = \frac{I_c - K_c + 2P_c}{S_c} + 1 \tag{4.4}$$

Now as, our input size I_c is 512×512. The first convolutional layer has 96 kernels of size 11×11. The stride is 4 and the padding is 0, therefore, the output image size is:

$$O_c = \frac{512 - 11 + 2 \times 0}{4} + 1 = 126 \tag{4.5}$$

The output image size is : $126 \times 126 \times 96$, corresponding to the number of channels. This image is then sent to the maxpool layer.

Let O_m be the size of the output image, I_m be the size of the input image to maxpool layer, S_m be the stride of the pooling operation and P_m be the pool size. The size of the output image is

Layers	1	2	3	4	5	6
Type	conv1+ maxpool1+ norm1	conv2+ maxpool2+ norm2	conv3	conv4	conv5+ maxpool3	full (fc6,fc7)
Channels	96	256	384	384	256	4096
Filter Size	11×11	5×5	3×3	3×3	3×3	-
Convolution Stride	4×4	1×1	1×1	1×1	1×1	-
Pooling Size	3×3	3×3	-	-	3×3	-
Pooling Stride	2×2	2×2	-	-	2×2	-
Padding Size	-	1×1	1×1	1×1	1×1	-

Table 4.1. : Network Parameters for the proposed framework

calculated using the following equation:

$$O_m = \frac{I_m - P_m}{S_m} + 1 \tag{4.6}$$

As per this equation, the size of the output image after passing through the first maxpool layer is:

$$O_m = \frac{126 - 3}{2} + 1 = 62 \tag{4.7}$$

So, the output image is of size $62 \times 62 \times 96$. After passing through the normalization layer, the size of the output remains unchanged, i.e. $62 \times 62 \times 96$. In a similar manner according to the parameters, sizes of the output across each of the layers are calculated.

For learning a representation of CNN (deep) layers, a few samples from the proposed floor plan dataset are used. Unlike the networks mentioned in the literature, the activations of all the

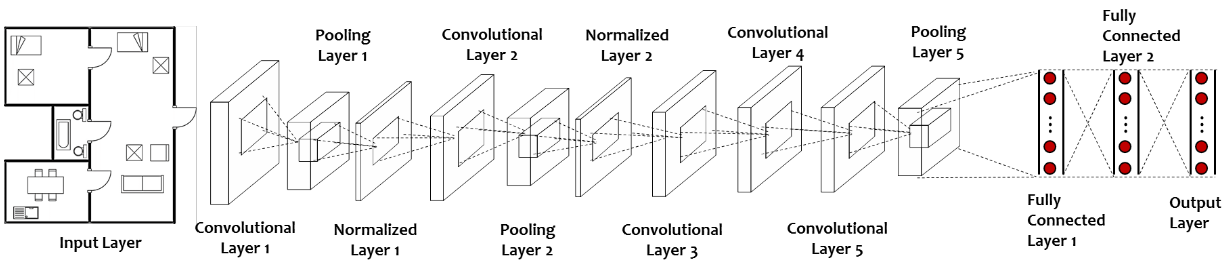


Figure 4.8. : Architecture of the Convolutional Neural Network (CNN) model used for feature extraction from the floor plan images.

layers (convolution, pooling, norm, and fully connected) for the task of floor plan retrieval are taken to investigate the effectiveness of each individual layer (see Fig. 4.8). The major motivation behind using lower convolution neural network layers is to understand the effectiveness of low level features as compared to higher-level feature representation for floor plan retrieval tasks.

To obtain the feature representation from query as well as the database samples, the images are directly fed into the input layer of the learned CNN model, and then the activation values from all the layers are taken (see Fig. 4.8). For conv, norm, and pool layers, the activation output is vectorized. Since computation from the feed-forward network based on the matrix multiplication is only needed once, the whole scheme is highly time efficient. In the next stage, i.e. matching and retrieval, query samples are matched with database samples to effectively retrieve the rank-ordered samples from the database.

4.3 MATCHING AND RETRIEVAL

Floor plan layout matching using the extracted features is essential for retrieving similar layouts to the query layout. In the rest of this section, the procedure to calculate matching score M for a pair (query and database) of layouts is described, corresponding to learned deep features.

Let N be the total number of images in the form of (x_ρ, y_ρ), $\rho = 1,2...,N$ where, x_ρ is the observed variable and y_ρ is the corresponding class label in a retrieval database. Let $F_H(x_\rho) : \{F_{H1}(x_\rho), F_{H2}(x_\rho), .., F_{HL}(x_\rho)\}$ be the set of hidden layer features extracted for x_ρ sample, where $F_{Hp}(x_\rho)$ represents p^{th} hidden layer feature extracted from x_ρ sample in database and L represents the total number of hidden layers in deep feature hierarchy. For experimentation purpose, features are extracted from 12 hidden layers of CNN model (see Fig. 4.8). For each sample in the database, features from each hidden layer $(F_{Hp}(x_\rho))$ are extracted and stored in the feature database as discussed. Let $F_H(q) : \{F_{H1}(q), F_{H2}(q), .., F_{HL}(q)\}$ be the the set of hidden layer features extracted from a query image (q). In this Chapter, analysis of the proposed framework is done using features features extracted from all the hidden layers:

- **Similarity based on each learned hidden layer feature representation**: In this approach, extensive experimental analysis was performed by comparing the performance of individual hidden layer for proposed floor plan retrieval task. The Matching Score (M) between query image (q) and an image sample from the database (x_ρ) using p^{th} hidden layer feature is calculated (and used for ranking), as:

$$M = \sum_{i=1}^{|F_{Hp}(x_\rho)|} |F^i{}_{Hp}(x_\rho) - F^i{}_{Hp}(q)|_2 \tag{4.8}$$

- **Similarity based on combination of hidden layer feature representation**: On the other hand, in this approach, the focus is on identification of the best performing hidden layer representation for each retrieval sample. This was achieved by combining the matching score outputs of each hidden layers using a *min* operation. The Matching score (M) between a query image (q) and a database image (x_ρ) using all the hidden layer features is calculated (and

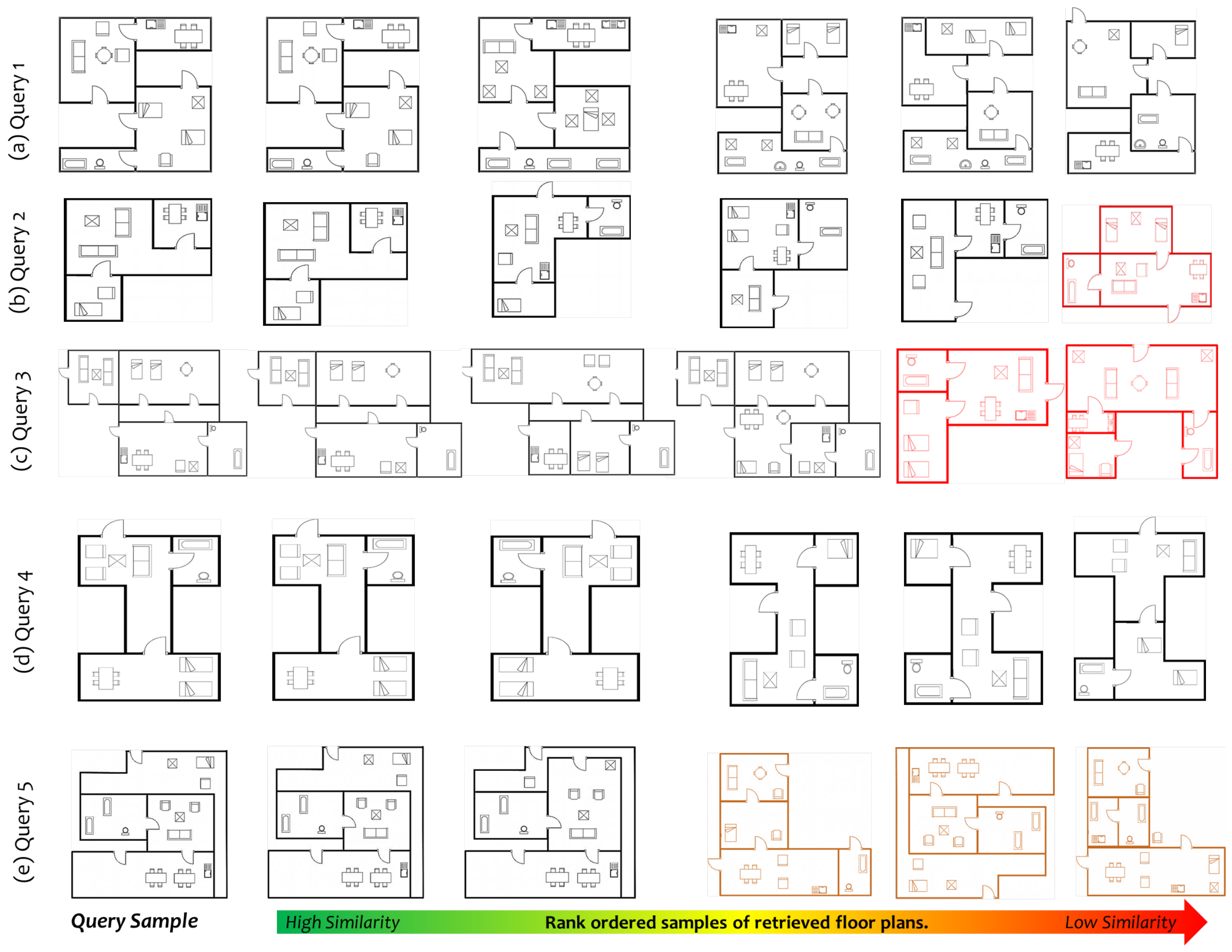


Figure 4.9. : Rank ordered retrieval result of the proposed framework for five different query floor plans from the ROBIN dataset. Here, top five rank ordered floor plans are shown for each query.

used for ranking), as:

$$M = argmin_p \sum_{i=1}^{|F_{Hp}(x_p)|} |F^i{}_{Hp}(x_p) - F^i{}_{Hp}(q)|_2, \quad p \in (1, ..., L) \tag{4.9}$$

The next section describes the qualitative and quantitative assessment of the proposed framework.

4.4 EXPERIMENTS AND RESULTS

The experiments were performed on the SESYD [Delalandre *et al.*, 2010b] and the ROBIN dataset to show the effectiveness of the framework. As already illustrated the SESYD dataset [Delalandre *et al.*, 2010b], contains 10 different types of layouts, each with 100 images varying in their arrangement of furniture inside the rooms. The image sizes in the database vary from

6775×2858 to 2056×1837. All floor plans are binarized to ensure use of only structural information while analysis. On the other hand, ROBIN dataset contains 510 floor plans divided into 3 broad categories based on the global layout shape.

The ROBIN dataset is split into two sets: (a) training - comprises of 30% samples and used for deep feature representation; and (b) testing - comprises of rest 70% samples and used as test (query) samples. As a retrieval database, the entire ROBIN dataset was used. Each floor plan image has been resized to a size of 512×512 for both matching and feature representation task. For deep model configuration, the architecture described in Fig. 4.8 is used, with same parameters as discussed in [Krizhevsky *et al.*, 2012a]. The same model pretrained on the ROBIN dataset is used for feature representation of the SESYD dataset.

4.4.1 Qualitative Results

The proposed framework is able to successfully retrieve a rank order set of floor plan images using the proposed deep learning framework. The top 5 rank ordered retrieval results for five different queries on ROBIN, are shown in Fig. 4.9. The arrow underneath the diagram depicts the decreasing order of similarity as one goes from left to right in Fig. 4.9 with green colour signifying high similarity and red signifying low similarity. The samples which are erroneously retrieved, i.e. not from the same class as that of the query class, are highlighted with "red" color. For all the results shown in Fig. 4.9, features extracted from the second normalization layer of the CNN stack were used. The rationale behind this choice was that the second normalizaton layer yields the best performance. Details analysis of such a selection is discussed in details in the next sub-section.

The first rank ordered result retrieved from the layout database is the query image itself. This is due to maximum matching score between the retrieved layout and the query layout. The subsequent results differ in the matching scores and are thus, ranked lower. In case of Fig. 4.9 (a), all the top 5 samples belong to the query class. Global layouts of the all the floor plans are identical and so is the number of rooms in the floor plans. The individual plans differ in terms of the number of furniture present in the rooms and their position. Readers are requested to note the difference in the shape and position of the individual rooms also for the rank ordered samples. There are 4 rooms in each floor plans, however the framework is able to rank order the samples correctly by ordering the most similar ones to the query first, followed by the rest.

Figure 4.9 (b-c) depict some of the failure cases. In Fig. 4.9 (b), the 4^{th} ranked result is erroneously retrieved (highlighted by "red") by the framework. However, the the 5^{th} sample is correctly retrieved. Justification of the rank 4 erroneously retrieved result can be understood through qualitative analysis. For example, room 1 in the query layout is adjacent to room 2, 3 and 4, that is identical to the adjacencies shared by room 1 in rank 2 retrieved result. Moreover, the size and type of furniture in the room 1 is same in case of the query and the rank 4 sample. Hence, similarity score between the query floor plan and the retrieved rank 4 layout is high, as compared to that with the rank 5 result. However, the similarity score of the rank 1-3 samples are marginally high as compared to rank 4 samples and hence they are ranked higher. In contrast, Fig. 4.9 (c) shows an instance where two out of top five retrieved results are incorrect. The rank 4 image has only three rooms, while the the rank 5 floor plan has four rooms in it. In this case, the framework is unable to capture the detailed features of the floor plans correctly. Even though, the number of rooms in a floor plan is different than the query, the framework ranks it higher.

Layer	conv1	pool1	norm1	conv2	pool2	norm2	conv3	conv4	conv5	pool5	fc6	fc7
MAP	0.503	0.527	0.554	0.537	0.557	0.561	0.539	0.537	0.541	0.536	0.472	0.465

Table 4.2. : Performance Comparison (MAP) of CNN layers used for floor plan retrieval task (on ROBIN dataset).

4.4.2 Quantitative Results

To quantify the performance of the framework the Precision (P) and Recall (R) metric is used. The precision values are averaged over all the queries for the particular recall values. Given a query floor plan image, retrieved layouts should belong to the same sub-category of layouts as the query, keeping in mind the preference set by the property seeker during querying. Remember that the global shape of the layout is the criteria for a given floor plan to belong to a certain class. The PR plot is shown for only the proposed *ROBIN* Dataset, as for SESYD dataset, the proposed and some of the existing state-of-the-art techniques [Sharma *et al.*, 2016a,b; Dalal and Triggs, 2005] yielded a flat PR curve (Precision value 1 for all Recall values). The reason behind such a flat curve for SESYD dataset is the simplicity of the dataset.

To understand the effectiveness of the proposed CNN layers, the Mean Average Precision (MAP) values were calculated for floor plan retrieval task using each layer separately (see Fig. 4.10 and Tab. 4.2). From Fig. 4.10 and Tab. 4.2, it can be concluded that: (1) it is not always true that the final layer i.e. the fully connected layer will always work best; and (2) normalization and pooling layers are more powerful as compared to convolutional layer for floor plan retrieval task.

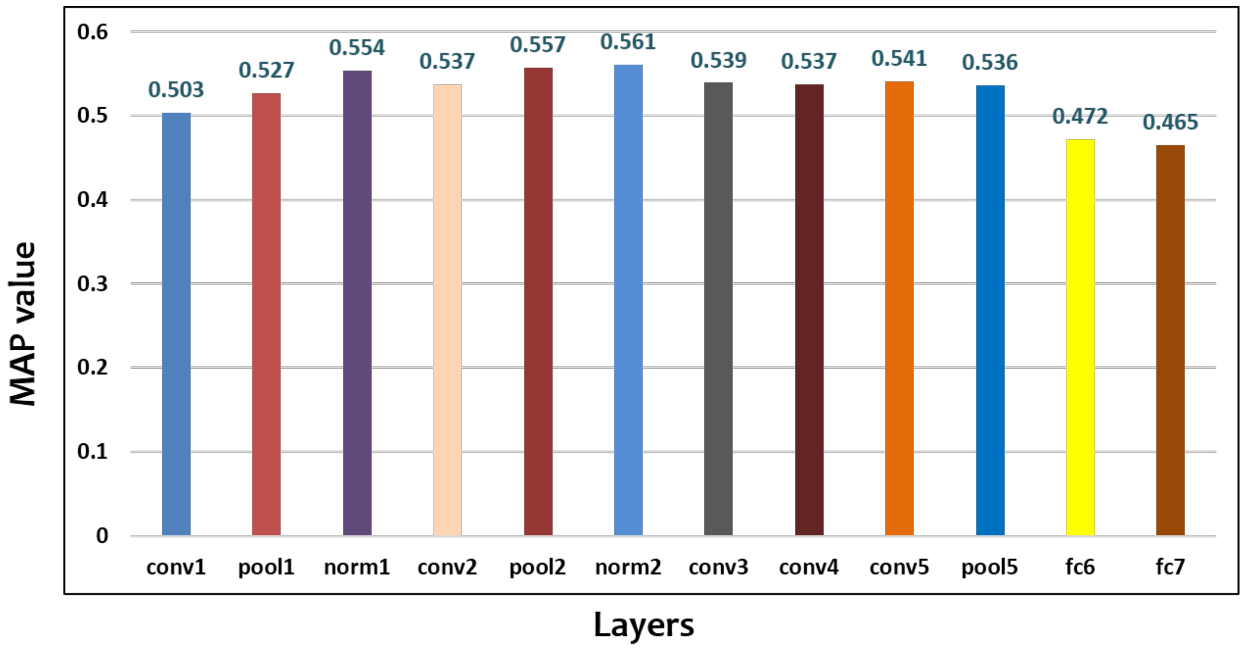


Figure 4.10. : Mean Average Precision values, comparing result of the framework on ROBIN dataset, using features from all the hidden layers of the CNN.

To show the effectiveness of the proposed framework, the technique proposed in this chapter

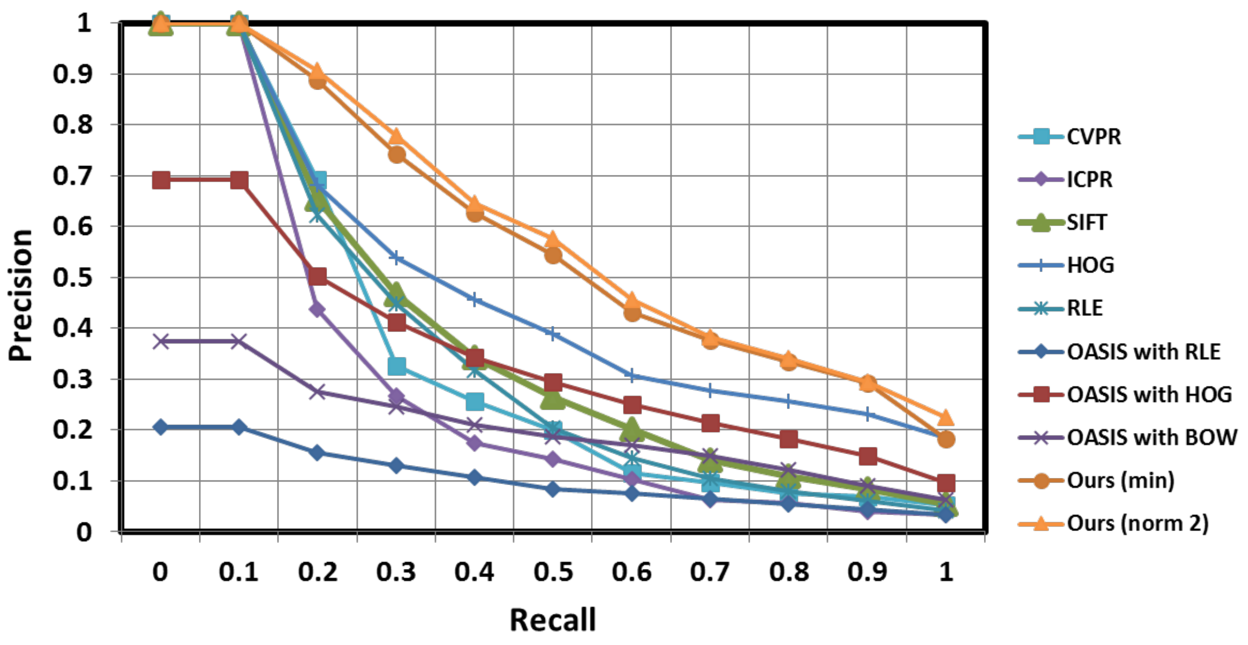


Figure 4.11. : Precision and Recall (PR) plot, comparing state-of-art methods with the proposed framework on ROBIN dataset.

is also compared with several state-of-the-art (SOA) techniques. Figure 4.11 depicts the PR plot comparing result of the framework on ROBIN dataset, with the other SOA techniques. It can be observed that the framework with features extracted from the Normalized layer 2 (see Fig. 4.10) is giving the best retrieval performance. It can also be noted from Fig. 4.11 that the proposed min based approach (which selects best layer for each retrieval sample, refer Eq. 4.9) is also outperforming all the state-of-the-art techniques. There is a significant improvement in the performance while using deep features as compared to other features or techniques.

As shown in Fig. 4.9 (a), given a query sample the rank 1 result is the query floor plan itself. This leads to the highest precision value of 1 during the initial recall. With further retrieval the average precision value decreases due to some incorrectly retrieved results belonging to other categories of layouts as compared to the query layout. It is to be noted that area under the PR curve is the highest for proposed deep learning based retrieval framework, as compared to other techniques applied for floor plan retrieval task as can be noted through the orange marker on the PR plot (Fig. 4.11).

The SOA techniques with which the proposed approach is compared with are standard image retrieval techniques that use hand-crafted features such as HOG [Dalal and Triggs, 2005], SIFT [Lazebnik *et al.*, 2006a], Run-Length Histogram and Online Algorithm for Scalable Image Similarity (OASIS) [Chechik *et al.*, 2009]) but are not able to capture well the complete content of the floor plan images. Also, reason behind the under performance of OASIS [Chechik *et al.*, 2009] is that, it focuses on the difference of similarity values between relevant and irrelevant image pairs, while ignoring the similarity value between highly similar images. Hence, when the same features (HOG, SIFT, RLH) are used under a canonical CBIR paradigm, better results are obtained in comparison to using them in conjuction with OASIS (see Tab. 4.3).

Techniques	**MAP**
CVPR [Sharma *et al.*, 2016b]	0.29
ICPR [Sharma *et al.*, 2016a]	0.23
HOG [Dalal and Triggs, 2005] based CBIR	0.40
SIFT [Lazebnik *et al.*, 2006a] based CBIR	0.33
RLH based CBIR	0.30
OASIS [Chechik *et al.*, 2009] with HOG	0.31
OASIS [Chechik *et al.*, 2009] with BOW	0.19
OASIS [Chechik *et al.*, 2009] with RLH	0.01
Ours (min)	0.54
Ours (norm2)	**0.56**

Table 4.3. : Performance Comparison in terms of MAP on *ROBIN* dataset.

4.5 SUMMARY

In this Chapter a deep learning framework is proposed to extract semantic features from floor plans and those features are used for matching and retrieval of similar floor plan images. Extensive experimentation was conducted and performance was compared with eight different state-of-the-art methods to demonstrate the edge that the deep learning framework has, as compared to the hand-crafted feature based retrieval. Although, deep learning is convenient and bypasses the need of curating features manually, the issue lies in the fact that the layers in the deep learning framework extract features implicitly. This implicit feature extraction at each layer does not give the user, the ability to give priorities to certain aspects or features while retrieval. For example, if a user wants to give higher preference to the feature "size of the rooms" while retrieval, then in the deep learning framework there is no such provision of giving additional weights to a particular feature. This, makes the application of the framework a little less intuitive. Hence, in Chapter 5, high level semantic analysis of floor plans and their subsequent retrieval is proposed. Additionally, a provision to adjust the weights of each feature while retrieval is given in the proposed technique. Such a provision can prove to be more practical to users while looking for similar floor plans sufficing their specific preferences.