Embodied AI for computational perception and understanding of spatial designs

RO014

Contents

1 Introduction and Our Objective

Understanding the layout and design of apartments and buildings is an important task for interior designers, architects, construction workers and various other professions. Hence, it has been found to be useful to use Deep Machine Learning and Computer Vision for a faster and more effective understanding of these layouts.

In this project, Semantic Segmentation [1], which is a Computer Vision task for segmenting an image into various classes, has been used. It differs from standard object detection and classification tasks as the model tries to find exact boundaries for the given objects in the image and every pixel in a given image belongs to a specific class.

However, the problem with Semantic Segmentation models are that they require a lot of data and a lot of computational resources to perform well. Hence, Transfer Learning [2] has been used to optimize the task. Transfer Learning is a Machine Learning technique where knowledge gained in one task is used for a similar second task. In our case, a pretrained Semantic Segmentation model was used and only the weights in the last few layers of the model were trained, hence reducing gradient computation time and decreasing the size of the dataset required for a quality model.

The project has been split into two tasks. In the first task, hereby referred to as the Interior Task, Semantic Segmentation is performed on HDB apartment interiors. Since there are no available datasets with semantic annotations of HDB interiors, a portion of the ADE20K dataset is used which contains interiors of homes. In the second task, hereby referred to as the Exterior Task, Semantic Segmentation is performed on HDB exterior facades. We use a dataset provided by SUTD's *Artificial-Architecture* Laboratory which contains images of HDB buildings and their annotations.

2 Methodology

2.1 Data Collection

2.1.1 ADE20K Dataset

The ADE20K dataset [3] is made up of over 27,000 images and their annotations from the SUN [4] and Places [5] databases. This dataset is used for the Interior Task of our project, Hence, only images that contained images of interiors of **houses** and **hotels** for training were used.

2.1.2 HDB Facade Dataset

Figure 1: HDB Facade Dataset Sample

The HDB Facade Dataset has been provided by the *Artificial-Architecture* Research Laboratory at the Singapore University of Technology and Design (SUTD). The dataset contains renders of HDB buildings and their annotations. The Grey class represents the walls of the building, the Red class represents the windows of the building and the Green class represents the voids/corridors of the building. However, the annotated images were obtained by first recolouring the model and then rendering the image. This led to the problem of their being different shades of grey, green and red as seen in the annotated image in Figure 1. Hence, data preprocessing techniques were used to correct the images prior to their usage in the Exterior Task.

2.2 Data Preprocessing and Cleaning

2.2.1 Color Classification

The HDB Facade Dataset is taken via Screenshots of Specific Facades, which are susceptible to altered shades instead of definite class-based colours of grey, red and green. Hence, a requirement of the Exterior HDB Semantic Segmentation Task is to modify this data to fit the pretrained model.

To fix the problem, a few steps were taken to alter the annotated images to fit into four specific colours. The initial approach of Colour Clustering with the K-Means was taken but failed due to the 4 colours varying across different images. Hence, an approach of categorising the colours into various groups based on a heuristic was taken. All the images were in an RGBA format. All of the background pixels had an alpha value of 0. Hence, it was easy to obtain all the pixels which belonged to the background class. It is observed that for RGB pixels, grey values have

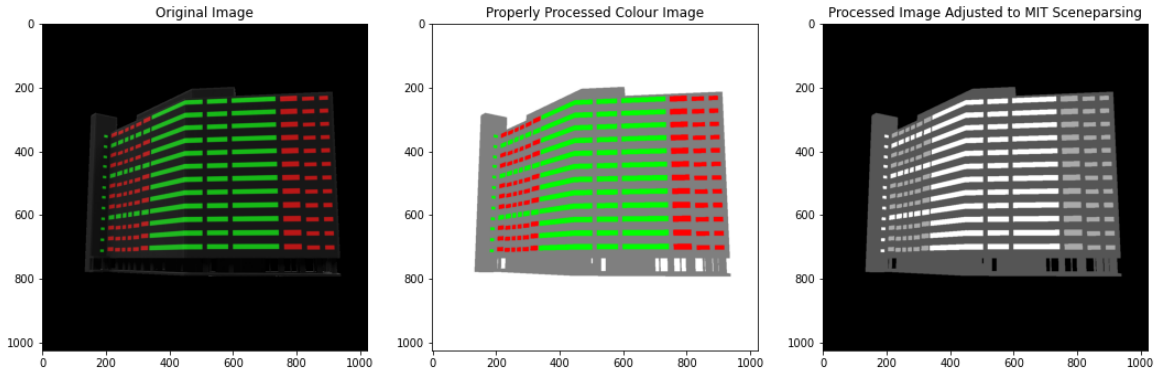

Figure 2: The Facade Images Converted and Processed (The image on the far right is brightened)

a similar R, G and B values. Hence, the heuristic [A.4] was used to determine nearly all the grey pixels. Next, we converted the image to HSV to detect green in the image as nearly all green pixels fall within a hue range of 80 to 140 degrees. Hence, the heuristic [A.5] was used to determine all the green pixels in the image. All the remaining pixels were red. Hence, we were able to reduce the number of colours to exactly 4, in every annotated image as seen in the second image in Figure 2. The solution was not foolproof but provided a high enough quality annotation.

2.2.2 Image Transformation

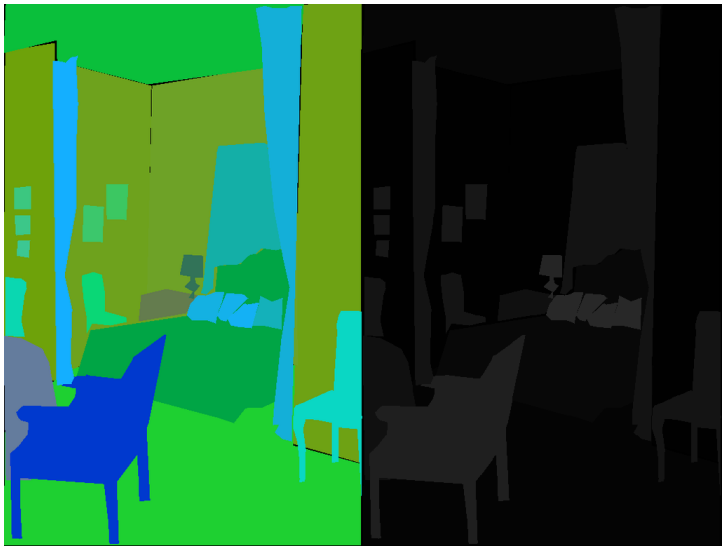

Figure 3: Annotations converted

The images in ADE20K have a very high resolution and there are over 3000 object classes. Since a large number of these classes were unnecessary for the interior task, a lowered number

of these classes were used. The annotated images are converted such that if a given pixel in an image is labelled as an object which has a class number of 56, the colour of that pixel in the transformed annotated image will be (56, 56, 56). A similar approach was taken for the Exterior Task images as seen in the third image of Figure 2. This method is adapted from the MIT Scene Parsing Benchmark [6].

2.3 Training

The model was written in PyTorch [7]. Stochastic Gradient Descent was used with ResNet-101 [8] as the Encoder Network and UperNet [9] as the Decoder Network with Negative Log Likelihood Loss[A.1]. Transfer Learning has been used to optimize the given task. In this case, MIT's Semantic Segmentation model, which acheived a IOU[A.2] of 42.66 on the sample set [10], was used as the baseline model and the last layer was retrained with a different number of classes. This model is trained on the ADE20K Scene Parsing Dataset, which is a slightly different dataset than ADE20K with fewer images and classes. The layers trained had to be limited due to compute power restrictions. The weights of the remaining frozen layer's weights did not change.

2.3.1 Interior Task

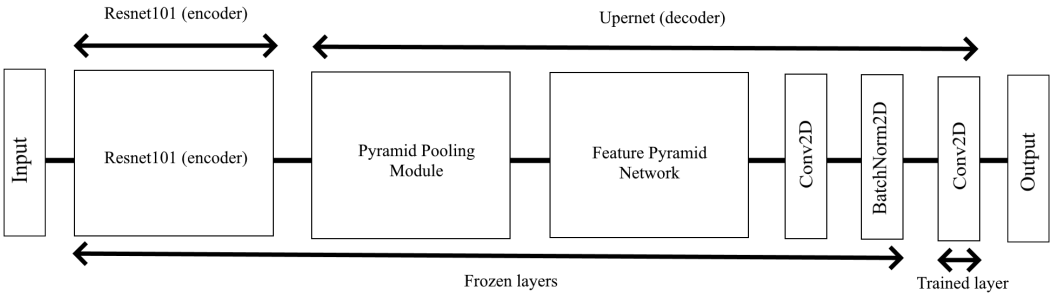

Figure 4: Interior Task Model.

The pretrained model had been trained on a similar dataset with similar data as the Interior Task. The model also had 150 classes, hence only the last Convolutional layer was trained as seen in figure 4. A training accuracy of 87.89% was achieved after 10 epochs with 120 iterations each. An attempt was made to train the model with more layers but the training time per iteration was significantly longer.

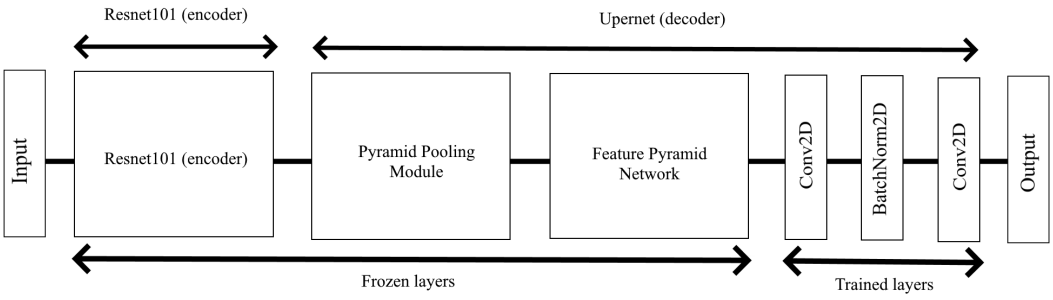

Figure 5: The Exterior Task Model.

2.3.2 Exterior Task

The Exterior Task had very different data than that which was used to train the pretrained model. It had 4 classes representing the background, the building, the windows and the voids. At first, only the last Convolutional layer was trained, and a training accuracy of 93.18% was achieved after 10 epochs with 120 iterations each. On a second attempt, the last two Convolutional layers and the last Batch Normalisation layer were trained as seen in Figure 5. A training accuracy of 98.28% was achieved after 10 epochs with 120 iterations each. An attempt was made to train the model with more layers but the training time per iteration was significantly longer.

3 Results and Discussion

3.1 Validation

Measuring accuracy is difficult for Semantic Segmentation tasks as pixel accuracy values can vary a large amount. To test the model, two metrics were used, Intersection Over Union (IOU) [A.2] and Pixel Accuracy [A.3].

3.1.1 Interior Task

The Interior Task model had a Mean Pixel Accuracy of 78.34% and a Mean IOU score of 46.35%. As seen from Figure 6, the model correctly identifies the boundaries between the different objects in the frame. However, there is a high amount of noise in the predicted segmentation's are slightly incorrect. It can also be seen that the model has a difficult time with partial objects as it fails to correctly identify the tables underneath the lamps and the table in the front of the image.

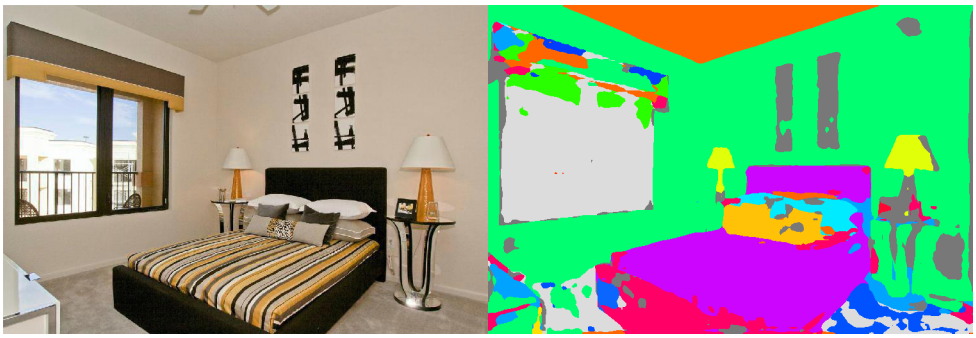

Figure 6: Sample image, Interior Task

3.1.2 Exterior Task

The Exterior Task model has a Mean Pixel Accuracy of 95.20% and a Mean IOU score of 59.79%. As seen from Figure 7, the model correctly identifies the building data, but it has a difficult time discerning between windows and void data. The high accuracy compared to the Interior Task is likely due to the lower class count of 4 and the larger number of layers trained.

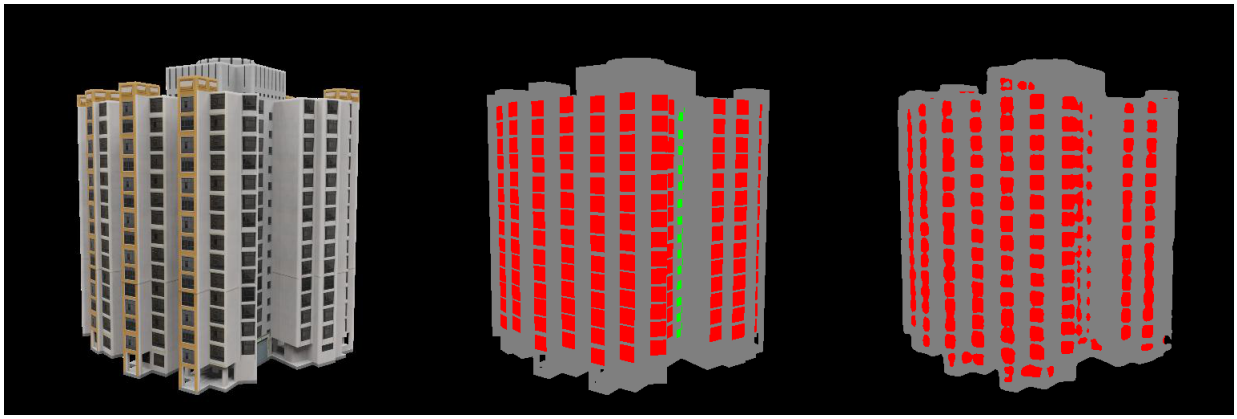

Figure 7: Sample Image, Exterior Task. From Left to Right: Original Image, Ground Truth and Predicted Image.

3.2 Conclusion and Recommendations

In conclusion, 2 models were successfully created for the Interior and Exterior Tasks. A better understanding of Semantic Segmentation and Transfer Learning was gained. However, more work can be done to improve the accuracy of both tasks. An important takeaway is that a large number of layers do not have to be trained to achieve a quality model. The model in the Exterior Task has difficulty distinguishing between void and windows and the model in the Interior Task is relatively poorer at the segmentation task in general.

4 Future Work

4.1 Material Type Detection

Previous research papers have used the UperNet model for not only segmentation of a given image based on the types of objects in the image but also the material type of the given objects such as wood or cotton. We would like to do a similar thing for the interior task so that there is a better understanding of the elements of the image. Understanding the materials of the objects in the image could also help us when it comes to issues such as reflectivity and glare, mainly improving IOU scores for objects such as mirrors or windows.

4.2 HDB Buildings with Backgrounds

A large problem is that the HDB buildings in the images do not contain scenery or backgrounds. This means that if one were to take a picture of an HDB building and run the model on it, it would return an incorrect result. Hence, in the future, a dataset is needed of HDB buildings and the backgrounds, so that a better model can be trained.

5 Reflection

6 References

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV].

[2] Stevo Bozinovski. "Reminder of the First Paper on Transfer Learning in Neural Networks, 1976". In: *Informatica* 44.3 (Sept. 2020). DOI: 10.31449/inf.v44i3.2828. URL: https://doi.org/10.31449/inf.v44i3.2828.

[3] MIT. *ADE20K dataset*. URL: https://groups.csail.mit.edu/vision/datasets/ADE20K/.

[4] Jianxiong Xiao et al. "SUN database: Large-scale scene recognition from abbey to zoo". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3485–3492. DOI: 10.1109/CVPR.2010.5539970.

[5] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[6] Bolei Zhou et al. "Scene Parsing through ADE20K Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[7] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[8] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[9] Tete Xiao et al. *Unified Perceptual Parsing for Scene Understanding*. 2018. arXiv: 1807.10221 [cs.CV].

[10] Shervin Minaee et al. *Image Segmentation Using Deep Learning: A Survey*. 2020. arXiv: 2001.05566 [cs.CV].

[11] Irem Ulku and Erdem Akagunduz. *A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images*. 2022. arXiv: 1912.10230 [cs.CV].

[12] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2016. arXiv: 1511.00561 [cs.CV].

[13] *Changing Colorspaces*. URL: https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html.

A Appendix

A.1 Negative Log Likelihood Loss

$$L(y) = -log(y)$$

Where,

- y is the output of the network
- $L(y)$ is the Loss with respect to y

A.2 Intersection Over Union (IOU)

The Intersection Over Union (IOU), or the Jaccard Index, is a statistical metric often used for comparing the similarity and difference in sample data sets [11]. In the given task of Semantic Segmentation, the IOU is a measure of the overlap between the predicted pixel-wise classification and the ground truth. The IOU values for every class were averaged to get the final IOU score.

$$\begin{aligned} IOU &= \frac{\sum_{j=1}^{k} n_{jj}}{\sum_{j=1}^{k}(n_{ij} + n_{ji} + n_{jj})} \\ &= J(A, B) = \frac{|A \cap B|}{|A \cup B|} \end{aligned}$$

Where,

- i and j are different classes in the dataset.
- n_{jj} is the number of pixels which are both labelled and classified as j. In other words, they are the *True Positives* for class j.
- n_{ij} is the number of pixels which are labelled as class i, but classified as class j. In other words, they are *False Positives* (false alarms) for class j.
- n_{ji}, the total number of pixels labelled as class j, but classified as class i are the *False Negatives* (misses) for class j.
- A and B are the ground truth and the predicted pixel-wise segmentation maps respectively [10].

A.3 Pixel Accuracy

Pixel Accuracy, or Global Accuracy [12], finds the absolute ratio of predicted pixels properly classified to match ground truth against the total number of pixels in a given image [10]. For a given $N + 1$ classes, the Pixel Accuracy is defined as follows: [11]

$$PA = \frac{\sum_{j=1}^{N} n_{jj}}{\sum_{j=1}^{N} t_j}$$

Where,

- n_{jj}, as previously defined in A.2, is the number of *True Positives* for a given class j.
- t_j denotes the total number of pixels labelled as class j.

A.4 Check Grey Function

This is a tried and tested function used to detect whether a given pixel in Red, Green and Blue (RGB) format is definitively grey in color. It is defined as followed:

$$|R - G| + |R - B| + |G - B| < 20$$

Where,

- R is the red value of a pixel (ranging from 0 to 255)
- G is the green value of a pixel (ranging from 0 to 255)
- B is the blue value of a pixel (ranging from 0 to 255)

A.5 Check Green Function

This is another function used to detect whether a given pixel in Hue, Saturation and Value (HSV) format is green in color. The Hue value is conventionally between 0 and 359, but the OpenCV Library uses a range of 0 to 179 [13].

$$40 < H < 70$$

Where H is the Hue value of a pixel (ranging from 0 to 179)