A Comparative Study of Image Segmentation Techniques

Vijay Varma Kanumuri, Sachin Karumanchi, Abhiroop Chintalapudi, Rohith
Indian Institute of Technology Hyderabad
{ai20btech11012, ai20btech11013, ai20btech11005, ee20btech11031}@iith.ac.in

Abstract

In this paper, we are gonna study the Evolution of Image Segmentation Techniques from its start(using Image Properties like color, contrast, texture etc) to the State of the Art techniques(CNNs, FCNs, Transformer based Models). We are gonna implement the techniques and compare the performance of these techniques on different types of Images.

1. Introduction

Image segmentation is a method of dividing a digital image into subgroups called image segments, reducing the complexity of the image and enabling further processing or analysis of each image segment. Technically, segmentation is the assignment of labels to pixels to identify objects, people, or other important elements in the image.

1.1. Motivation

Image Segmentation is used extensively in real life applications. For image analysis, it is the initial stage. Image Segmentation is a major part of computer vision applications. Its applications vary from Autonomous vehicles to Face Recognition. It is used to locate and identify cancer/tumour cells in the Medical Images. It is also used in Image Based Searches.

2. Literature Review

We have studied three Reference Papers which discusses about various techniques for Image Segmentation and their Implementations and experimental Results.

2.1. Texture-based color image segmentation using local contrast information

In this paper, they discuss about a method called JSEG for Image Segmentation. Further the propose a new method based on JSEG which is better than the original.

2.1.1 JSEG Method

In JSEG, colours are first quantized and then images are spatially segmented. The JSEG segmentation algorithm begins by reducing the colours in the image through quantization based on peer group filtering (PGF) and vector quantization.

The result of colour quantization is a class-map which logically associates the label of a colour class with each pixel belonging to the class. The spatial segmentation that follows as the second step in JSEG is based on the information in the class-map.

The JSEG method then calculates a J value over a tiny window centred at each pixel in turn. A J-image or J-map is created as a result, with the associated pixel values determined by local J values. The likelihood that a pixel is close to an area boundary increases with the local J value.

The J Map can be used to check for boundaries but it does not take into account the strength of the boundary. This is the reason for Over Segmentation by JSEG. The Following approach overcomes this.

2.1.2 Improved Contrast Map

The Base of the this approach lies in the claim that color contrast is weakly correlated with luminance and color levels. So they converted the original image to the CIE LAB* color space.

They assumed the Euclidean Distance between two colors representation approximates the Perceptual Difference between the two colors. They calculated the contrast using a 3*3 window using Euclidean Distance in CIE LAB color space.

But the constructed Contrast Map is usually noisy. To overcome this they proposed a method of two stages.

The first step comprises of quantizing the contrast map which preserves fine details or produces strong noise cancellation.

The second step uses Russo's Filter to eliminate the noise even more. This improved contrast map (ICMap) reduces noise and enhances the boundaries to a large extent.

2.1.3 Improved Contrast JSEG

For an M×N image, Contrasted Map is first calculated according to classic equation, and then improved according to Equation (1) to get ICmap. At this point, the magnitude is normalized as follows:

$$ICMAP(i,j) = (L-1)(1 - min\mu_{SM}(B_1), \mu_{SM}(B_2)) \quad (1)$$

$$w_{IC}(i,j) = \frac{ICMAP(i,j)}{ICMAP_{max}} \quad (2)$$

The J value (also normalized) of the local region is computed according to JSEG method used above in section 2.1.1. Then the proposed measure is formed by weighting the J as follows:

$$J_{IC}(i,j) = w_{IC}(i,j).J(i,j) \quad (3)$$

Using the new JIC measure, we construct a JIC image or map, where pixel values can be used to indicate the interior and boundary of the region.

2.1.4 Experimental Results

We have implemented the Algorithms Peer Group Filtering and Vector Quantization for a sample Image and working on to implement the JSEG Method. We were not able to exactly implement the JSEG and the Improved Contrast JPEG Method since the Algorithm and the Hyper Parameters were not Clear.

We were not able to exactly replicate the results as in the Paper. The Algorithm is not giving the Expected Results. We used some sample Images for testing the Algorithm. The Results are as follows:

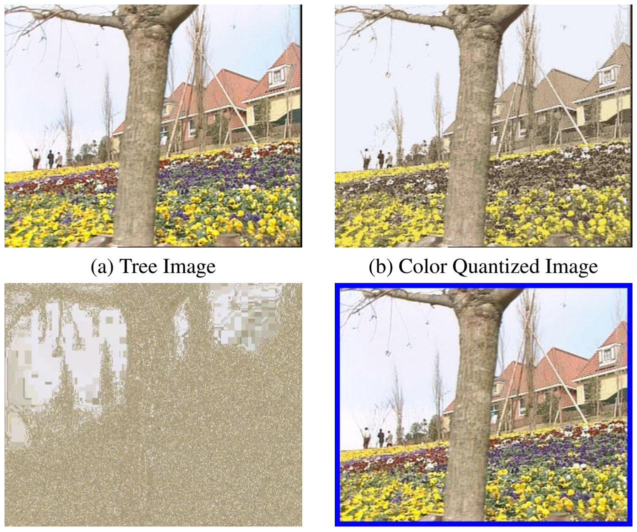

(a) Tree Image (b) Color Quantized Image (c) Color Quantized Image 2 (d) JSEG Segmentation

Figure 1. Segmentation Results

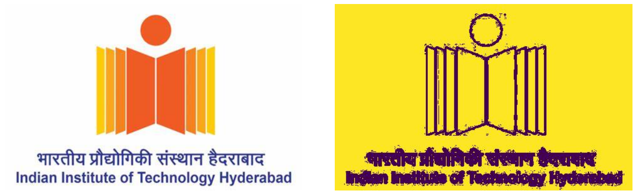

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad
(a) IITH Logo (b) IC JSEG Method

Figure 2. IC JSEG Result

2.2. Image Segmentation using Convolutional Neural Network for Image Annotation

In this paper they used Convolutional Neural Network for Image Segmentation which in turn used for Image Annotation. The Main Objective of the Paper is Automatic Image Annotation (AIA). The reason behind this Automatic Image Annotation (AIA) is to automatically allot labels to objects in the Image. AIA methods can roughly be classified into five categories: (i) Nearest neighbor models, (ii) Generative models, (iii) Discriminative models, (iv) Tag completion- based AIA models and v) Deep learning based models.

Deep learning-based models enable AIA tasks to be solved using deep learning-based feature representation. CNN is used for robust feature generation in image annotation. For image annotation, several models are used, including the CNN-RNN framework used in the RIA model and the Deep Multiple Instance Learning (DMIL) model.

The AIA framework is made up of steps like image segmentation, feature extraction, and classification. Image segmentation is a critical module in AIA. The paper focuses on the Image Segmentation Part. Unstructured regions in images of natural scenes necessitate computationally complex analysis for segmentation. Several proposed segmentation algorithms in the literature make use of image elements such as colors, textures, shape, and object compositions.

The image segmentation aims to correctly segment the image because it is the foundation of the region annotation. Many image segmentation algorithms have been developed and are widely used in image annotation, including JSEG and NCUT. It is, however, ineffective in segmenting regions with similar color distributions. The image's main elements are texture, color, and shape. These characteristics are crucial in the effective segmentation of objects or regions.

Colors are perceived by the human eye and are the primary feature that distinguishes objects. As a result, a color enhanced segmentation algorithm based on CNN is proposed. The proposed algorithm attempts to distinguish objects of similar color by processing each R, G, and B color space separately, aided by edge detection.

2.2.1 Color Image Segmentation using CNN

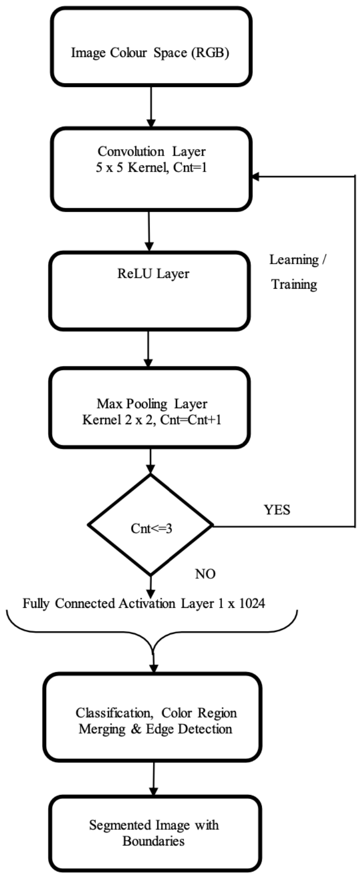


Image Colour Space (RGB)
Convolution Layer 5 x 5 Kernel, Cnt=1
ReLU Layer
Learning / Training
Max Pooling Layer Kernel 2 x 2, Cnt=Cnt+1
Cnt<=3
YES
NO
Fully Connected Activation Layer 1 x 1024
Classification, Color Region Merging & Edge Detection
Segmented Image with Boundaries

Figure 3. CNN Architecture

The Paper discusses the Architecture of the CNN for Image Segmentation as in 3. They used a Convolutional Layer with Kernel Size 5*5*3 for the 3 color channels. Convolutional layer initially starts with the smoothing operations similar to the average filter. The kernel's starting weights are one by twenty-five, and they are adjusted throughout the training process to predict objects from image pixels. It then uses a Relu Activation Layer (to tackle vanishing gradient problem).

After the Activation Layer, it then uses Max Pooling Layer with Kernel of Size 2*2*3. It improves the pixels' score, which aids in classification even more. The benefit of this pooling is that it maintains the objective of predicting similar pixels or regions while giving similar pixels in the training procedure a similar weight. The Input Images are resized to 284*284*3 and then passed to CNN. The CNN is trained using Backpropagation of the Softmax Loss of each class error. Batch Gradient Descent is used to perform the Backpropagation. Also the Learning Rate is varied from 0.05 to 0.01 as training reaches the end.

R, G, and B color spaces are separately processed by the algorithm using CNN. As a result, each item is tagged or identified in every color space. When R, G, and B color space results are finally combined, the resulting segmentation is quite better.

They used the Corel-10K Images Dataset for Image Segmentation. The Dataset contains 10000 Images and 100 Classes. They trained the above CNN on the Corel Dataset and compared the results with some of the CNN based Image Segmentation approaches.

2.2.2 Experimental Results

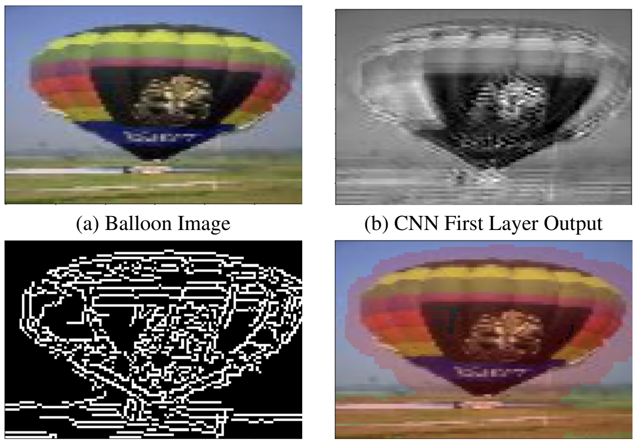
Dataset

We have used the Corel10K Dataset for training the Classifier Model. This can be found here. It contains 10,800 Images with 80 Classes. We used only a subset of it with 2720 Images and 21 Classes(Objects).

Model Architecture

We tried many architectures to implement the Classification Model. But finally, we have used Resizing Layer, DepthwiseConv2D, MaxPool2D three times repetitively followed by a Flatten and a Dense Layer. We trained it in Colab using Tensorflow GPU for 20 Epochs.

Then we extracted the output from the First Convolution Layer and then passed it to the Canny Edge Detection Method. But the Edges are very high in number for segmentation. An example Image is:



(a) Balloon Image (b) CNN First Layer Output

(c) Canny Edges (d) Segmented Image

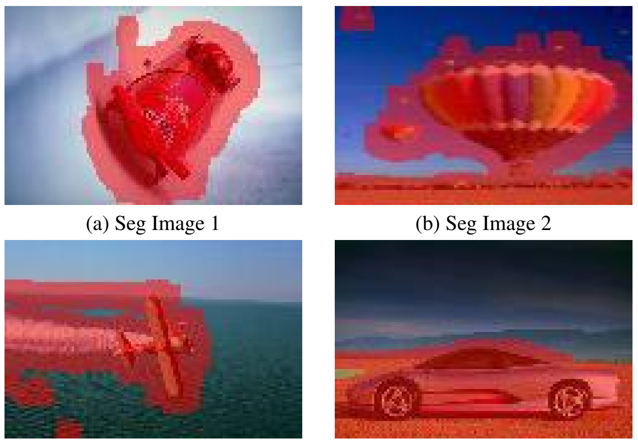
Figure 4. Segmentation Images

So we used the Dilation Method First on the Canny Edge Map. This filled small gaps present in the edge map.

Then we applied Erosion Method on the above output and this eliminated small blobs. The above process refined the Segmentation and helped for effective segmentation.

Results

The Results obtained for Some of the Test Images are:



(a) Seg Image 1 (b) Seg Image 2 (c) Seg Image 3 (d) Seg Image 4

Figure 5. Segmentation Images

Metrics

The Dataset used is a Classification Dataset which does not contain any Annotations useful for Segmentation. So we could not provide any Metrics for the Segmented Images. But the Segmented Image is not so bad and can be further refined to give good Segmentation Results.

2.3. SEMANTIC SEGMENTATION OF HIGH-RESOLUTION REMOTE SENSING IMAGES USING AN IMPROVED TRANSFORMER

In the era of Deep Learning, we use CNNs for Image Segmentation. CNNs are widely used for Semantic Segmentation due to their powerful ability in extracting local features from Images. But to have better understanding of scenes for semantic segmentation, global information is of vital importance especially for High Spacial Resolution Images which have more complex geometric structure and finer texture features resulting in extreme difficulties for accurate semantic segmentation.

A Transformer with Attention mechanism was proposed to capture global information from the Images which the CNN doesn't capture. Previous approaches used a Swin Transformer using shifted windows to combine local information in a single window just like convolution in CNN. A model was proposed using Swin Transformer and Uper Head in UperNet (Swin Transformer with UPer head (STUP)).

2.3.1 Architecture of Swin Transformer (Backbone)

The Swin Transformer operates by processing an image in a hierarchical manner, where the patches are grouped into smaller and larger "windows" and processed using a transformer network. Here's a high-level overview of how the Swin Transformer operates:

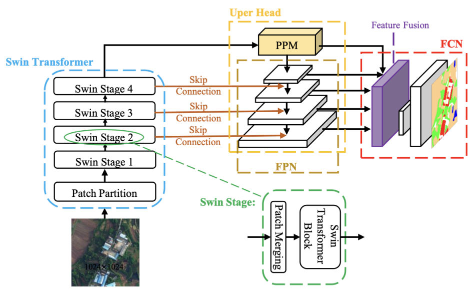


Figure 6. Model Architecture

1. **Patch Embedding:** The image is first divided into non-overlapping patches of a fixed size, and each patch is flattened into a vector. These patch vectors are then projected to a lower-dimensional space using a learnable linear projection to obtain patch embeddings.

2. **Shifted Windows:** Instead of processing all the patches in a grid-like pattern, the Swin Transformer processes them in shifted windows. In each layer, the patches are grouped into non-overlapping windows of a fixed size, and these windows are shifted in a certain pattern. Each patch only interacts with a subset of the patches in the same layer, reducing the number of connections and computations required.

3. **Multi-Scale Processing:** The Swin Transformer uses a hierarchical design where the patches are first processed within small windows in the lower layers, and then in progressively larger windows in the higher layers. This approach allows the network to capture both local and global features of the image.

4. **Transformer Encoder:** The Swin Transformer uses a transformer encoder to process the patch embeddings within each window. The transformer encoder consists of a stack of transformer layers, where each layer has a self-attention mechanism to capture global dependencies and a feed-forward network to capture local features. The Swin Transformer uses a variant of the transformer encoder called the Swin Transformer Block, which incorporates layer normalization, residual connections, and a window-based feed-forward network.

5. **Multi-Scale Integration:** After processing the patch embeddings within each window, the Swin Transformer integrates the information from different scales by combining the output of the different layers. This is done using a learnable fusion mechanism that combines the features from the different scales.

6. **Classification Head:** Finally, the Swin Transformer uses a classification head to predict the label of the image. The classification head consists of a global average pooling layer to aggregate the features from all the patches, followed by a fully-connected layer with a softmax activation function to predict the label.

Overall, the Swin Transformer operates by processing an image in a hierarchical manner using shifted windows and a transformer network, allowing it to capture both local and global features of the image. This approach has been shown to be highly effective in achieving state-of-the-art results on various image recognition tasks.

2.3.2 Architecture of Swin Transformer Block

The Swin Transformer Block is a variant of the transformer encoder used in the Swin Transformer architecture, and it operates by processing the patch embeddings within a shifted window using a self-attention mechanism and a window-based feed-forward network. Here's a high-level overview of how the Swin Transformer Block operates:

1. **Input Embedding:** The Swin Transformer Block takes as input a set of patch embeddings corresponding to the patches within a shifted window.

2. **Layer Normalization:** The patch embeddings are first normalized using layer normalization, which helps to improve the stability and convergence of the network.

3. **Self-Attention:** The Swin Transformer Block uses a self-attention mechanism to capture global dependencies between the patch embeddings within the window. The self-attention mechanism computes a weighted sum of the patch embeddings, where the weights are determined by the similarity between the patch embeddings.

4. **Residual Connection:** The output of the self-attention mechanism is added to the input patch embeddings, using a residual connection. This helps to ensure that the network can learn to capture both local and global features of the image.

5. **Window-based Feed-Forward Network:** The Swin Transformer Block also uses a window-based feed-forward network to capture local features of the patch embeddings. The feed-forward network applies a set of learnable linear transformations to the patch embeddings within the window, followed by a non-linear activation function.

6. **Residual Connection and Layer Normalization:** The output of the feed-forward network is added to the output of the self-attention mechanism, using another residual connection. The resulting vector is then normalized using layer normalization.

7. **Output:** The output of the Swin Transformer Block is a set of patch embeddings corresponding to the patches within the shifted window, but with added information from the self-attention mechanism and the feed-forward network.

Overall, the Swin Transformer Block operates by using a self-attention mechanism and a window-based feed-forward network to process the patch embeddings within a shifted window, and by using residual connections and layer normalization to improve the stability and convergence of the network. This approach has been shown to be highly effective in achieving state-of-the-art results on various image recognition tasks.

2.3.3 UPER Net (Decoder Head)

The main idea behind UperNet is to combine low-level and high-level features to produce accurate and detailed segmentation results.

1. **Backbone Network:** The backbone network is used to extract features from the input image. Here we uses Swin Transformer as the backbone network.

2. **Feature Pyramid Network (FPN):** FPN is used to extract features at multiple scales. UperNet uses FPN to generate a feature pyramid with high-level and low-level features.

3. **Global Convolutional Network (GCN):** GCN is used to capture the global context of the image. It applies a convolution operation with a large kernel size to the feature maps.

4. **RefineNet:** RefineNet is used to refine the segmentation results. It consists of multiple layers of convolutions and skip connections to improve the accuracy of the segmentation.

5. **Final Segmentation:** The final segmentation is produced by combining the outputs of the different stages of the network. The output is a pixel-wise classification of the image, where each pixel is classified into one of the predefined classes.

2.3.4 FCN (Auxiliary Head)

1. FCN is added as a secondary output layer to the main network, and its purpose is to provide additional training signal to improve the performance of the network on specific tasks such as object detection or instance segmentation.

2. The FCN auxiliary head is typically a small convolutional network that takes the features from an intermediate layer of the main network as input and produces a pixel-wise classification of the feature maps. The output of the auxiliary head is then combined with the output of the main network to compute the final loss and update the weights of the network during training.

3. Using an FCN auxiliary head can improve the accuracy and convergence speed of the main network, especially when the main task involves pixel-level predictions.

2.3.5 Experimental Results

Dataset

We have used the Stanford Background Dataset which contains 715 Images with 8 Classes namely Sky, Tree, Road, Grass, Water, Buildings, Mountains, Foreign Objects. We trained the Swin Transformer with this Dataset.

Model Architecture

The Model we have used is Swin Transformer with UPer Head which is already trained on ADE20K Dataset. We trained the Model for 500 Epochs.

Swin Transformer:

1. Total No of Transformer Blocks = 4
2. Embedded Dimensions in Transformer Blocks = 128
3. Patch Size = 4
4. Window Size = 7
5. Attention Heads in Transformer Blocks = [4,8,16,32]

UPer Net:

1. In Channels at Each Layer = [128,256,512,1024]

FCN:

1. In Channels = 512

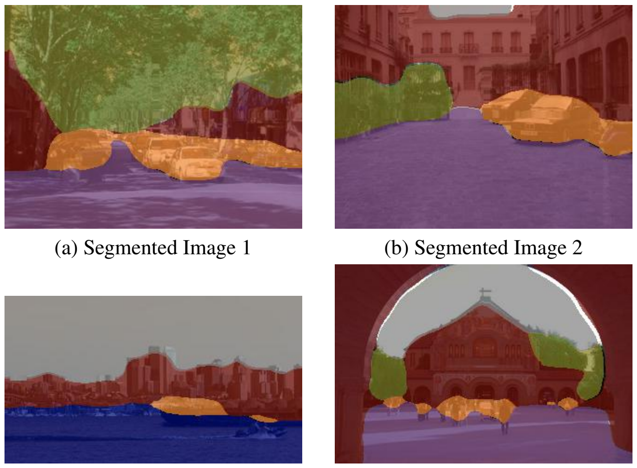
Note

We have followed this tutorial for training the Transformer. We have done several changes including changing the complete architecture and model used and also rectified a Bug (Displaying Image in the Wrong Color Space) in the Tutorial.

Results

Metrics

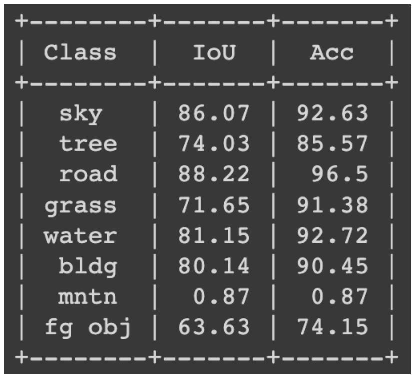
1. aAcc(Average Accuracy): 87.8600
2. mIoU(Mean Intersection of Union): 68.2200
3. mAcc(Mean Pixel Accuracy): 78.0300



(a) Segmented Image 1 (b) Segmented Image 2

(c) Segmented Image 3 (d) Segmented Image 4

Figure 7. Segmentation Images



```
+--------+-------+-------+
| Class  |  IoU  |  Acc  |
+--------+-------+-------+
|  sky   | 86.07 | 92.63 |
|  tree  | 74.03 | 85.57 |
|  road  | 88.22 |  96.5 |
| grass  | 71.65 | 91.38 |
| water  | 81.15 | 92.72 |
|  bldg  | 80.14 | 90.45 |
|  mntn  |  0.87 |  0.87 |
| fg obj | 63.63 | 74.15 |
+--------+-------+-------+
```

Figure 8. Model Metrics

2.4. Contributions

1. Vijay Varma – Worked on implementing the Code of 2nd Paper(CNN)
2. Sachin – Worked on implementing the Code of 3rd Paper(SWIN)
3. Abhiroop – Worked on implementing the Code of 1st Paper(JSEG)
4. Rohith – Worked on collecting datasets, helper functions in the code and also preparing the Report and Presentation.

References

[1] Texture-based color image segmentation using local contrast information by Yuchou Chang, James K. Archibald, Lee Dah-Jye (2007).

[2] Image Segmentation using Convolutional Neural Network for Image Annotation by S. B. Nemade, S. P. Sonavane (2020).

[3] Semantic Segmentation OF High-Resolution Remote Sensing Images using an Improved Transformer by Yuheng Liu, Shaohui Mei, Shun Zhang, Ye Wang, Mingyi He, Qian Du (2022).

[4] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows by Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo (2021).

[5] ChatGPT