Deep Learning Approaches to Semantic Segmentation of Diagram

By

Duc Tri Dang

A thesis submitted to the

Department of Mathematics and Computer Science

Mount Allison University in partial fulfillment of the requirements for

the Bachelor of Science degree with Honours in Computer Science

April 7, 2025

“Doubt kills more dreams than failure ever will.”

Suzy Kassem

Acknowledgement

Personally, this honor thesis is a miracle. There were moments when I felt desperate, and I could not finish it. But finally, I was able to write these lines, expressing my deepest gratitude to the people I have accompanied along the way.

First and foremost, I would like to thank my supervisor, Dr. Michael Cormier. Thank you for choosing me to be your honor student, and thank you for believing in me. You are the one who always encouraged me when I felt there was no light at the end of the road. I have learned a lot from your expertise, experience, and spectacular sense of humor, and I believe these lessons will be memorable for me in life. Thank you for being an awesome supervisor to me!

I would like to thank my second reader, and my best programming team coach, Dr. Keliher. I have improved a lot since I was a naïve first-year computer science student, to first place in the Atlantic region in the ICPC, thanks to you. Thank you for being the best programming team coach I have ever had.

I would like to thank my fellow co-researchers from the University of Victoria, Dr. Miguel Nacenta and Yichun Zhao. Thank you for proposing a meaningful project to low-vision and blind people and thank you for such a wonderful journey we have been together.

I also want to thank Laura Paul, the previous research student who studied this topic. Your amazing work has been my guideline and led me along with this thesis. You have been the invisible support I had on this journey, and I hope we can meet again someday.

To Cameron Swift, one of my greatest teammates at Mount Allison University. We have been through so many memories, from the nights fixing bugs in Object-Oriented Design, to the nights video calling for Software Design. You have been such an ideal model to me, you are absolutely the ultimate "All-rounder". Believe in yourself, and I wish you the best of luck for your future.

I also want to say thank you to my study mates. To Lenale Le, Alice Tran, Chau Vo, Nicholas Nagi, Sabrina Sandy, and Sawyer Stanley, thank you for staying alongside me and motivating me to overcome adversities. I also wish you the best for your future.

To my mental supporters, from the faculties and students of the Department of Mathematics and Computer Science to random people who just saw me once but wished me luck with my thesis, thank you!

Last but not least, I would love to express my deepest gratitude to my Mom and Dad, who I did not spend most of the time with while doing this thesis. I know you were worried about me, worried whether I have enough meals in a day, worried if I feel lonely or overwhelmed with this thesis. I know I am the kind of person who prefers actions over saying, but I really want to say: "Thank you for being my Mom and Dad! I am proud to be your son."

And to the readers, thank you for reading this paper, and I hope this paper inspires you, like the way I was inspired by a plethora of papers I have read.

Again, thank you for everything.

Cảm ơn vì tất cả.

Abstract

This thesis is a part of the TADA project, short for “Touch-and-Audio-based Diagram Access for Blind and Low-Vision People”. The TADA project is a tablet-based interactive system designed to make diagram exploration accessible to blind and low-vision (BLV) people through musical tones and speech. In this paper, we are conducting the experiment of producing the semantic segmentation from diagrams using deep learning methodologies, as a part of the TADA project pipeline. The main concepts of deep learning had been used is Convolutional Neural Network (CNN), with applying linear model network and encoder-decoder network. Furthermore, we also experimented with superpixel segmentation, how did it perform with CNN and semantic segmentation. Overall, the linear model network without using superpixel performed the best, and we also figured that the idea of superpixel was not suitable for the purpose of creating semantic segmentation for diagrams.

Table of Contents

1. Introduction 5
2. Related Works 7
  2.1. CNNs and SegNet Model 7
    2.1.1. Convolutional Neural Networks 7
    2.1.2. SegNet Model 8
  2.2. Superpixel Approaches 10
    2.2.1. The first introduced and how it was used 10
    2.2.2. Superpixel Benchmark 14
    2.2.3. Fully Convolutional Neural Networks application 16
  2.3. Gestalt Theory 19
3. Architecture of approaches 23
  3.1. Input Data Generation 24
  3.2. A Convolution Neural Networks approach 29
  3.3. Fully Convolution Neural Networks with Superpixel Segmentation 30
  3.4. Fully Convolution Neural Networks with Superpixel Segmentation with revision 32
4. Experiment and Result 34
  4.1. Overall Performance 34
  4.2. Trial run of the models 36
  4.3. Performance of model V1 37
  4.4. Performance of model V2 38
  4.5. Performance of model V3 39
  4.6. Training and Validating Performance Confusion Matrices 40
  4.7. Testing performance confusion matrix 41
5. Discussion 42
6. Conclusion and future work 46
7. Reference 47

1. Introduction

Image Segmentation was first introduced in the 1960s, with the appearance of an operator for detecting the edges between different parts of an image named Roberts operator (Khosrowpour, 2009). After that, it has been developed and improved with a plethora of methods until today. The basic definition of image segmentation is the action of identifying different components inside a picture. As a matter of fact, a picture is a group of pixels that indicate multiple components. For instance, a landscape picture includes components of trees, mountains, a lake, etc. In computer vision, a component of a picture is defined as a group of similar pixels that represent specific objects and can be disparate from one another by the difference in features, namely color, texture, and direction. In other words, the process of image segmentation is the creation of partitions so the computer can examine each part and recognize them particularly.

Image segmentation has various applications, and its main application is detecting and recognizing objects. In normal life, it is not rare to observe an application of the segmentations, from face and fingerprint recognition to object recognition in airport security and detection in traffic cameras. It is also used in medicine to analyze images for diagnosis of anatomy structure (Kamalakannan et al., 2010) surgery planning, locating tumors and other pathologies (Wu et al., 2014), radiotherapy (Georgescu et al., 2022), etc.

There are two main types of image segmentation, respectively semantic segmentation, and instance segmentation, and this paper will focus on semantic segmentation. Semantic segmentation creates the classification of different components in the image. It differs from instance segmentation where it does not need to disparate different parts of the same component. For example, given a picture with 5 people, the mission of semantic segmentation is figuring out which part is people, and which part is landscape, while instance segmentation needs to know how many people are inside the picture, where is the first person, who is the second, etc.

As mentioned above on applications of image segmentation, this thesis is a contribution to a specific application of Image Segmentation to the community. “Touch-and-Audio-based Diagram Access for Blind and Low-Vision People”, or as known as TADA, is a tablet-based interactive system designed to make diagram exploration accessible to blind and low-vision (BLV) people through musical tones and speech (Zhao et al., 2024) Nowadays, it is challenging for BLV people to access diagrams, while alternative texts cannot allow people to conduct tasks on a diagram such as identifying a node or its path. Furthermore, there are very few interaction techniques built for BLV people, and specialized hardware is very inconvenient for transportation. Based on those difficulties, TADA offers an accessible and interactive solution for diagram discovery for BLV people. For the TADA pipeline, the idea is that the semantic segmentation is used to identify major categories in the image; that data is fed to instance segmentation to identify individual nodes, edges, and so on. In order to understand the graph, however, we will still need to associate edges, nodes, and labels with each other. That will give us an interpretation of the diagram that can be used with the TADA interface. Therefore, this thesis focuses on semantic segmentation for the TADA project in parsing information from diagrams.

The process of semantic segmentation is a classification problem, in which the algorithm must assign objects to their corresponding classes. There are various machine learning methodologies which may be suitable for this task, including Naïve Bayes, Random Forest, and Neural Networks. Neural networks are inspired by how the brain processes information. These models are biologically inspired rather than exact replicas of how the brain works (Islam et al., 2019). During the research, we will concentrate on a specific type of Neural Network, Convolution Neural Networks, because they have proven to be the most suitable for this project. This paper will discuss the concepts of CNNs, their performance, and observations within the models.

1. Introduction
2. Related works: In this section, we will describe the foundations ideas of CNNs, the architecture of SegNet model, superpixel segmentation methodologies, and take a step back in the learning patterns according to Gestalt Theory.
3. Architecture of models: In this section, we will introduce our three models in completing the task of semantic segmentation.
4. Performance: This section will illustrate the experiment results of the models.
5. Discussion: In this section, we will discuss the result received in section 4.

6. Future Work and Conclusion: This section will conclude the paper, and point out what need to be done for the future.

2. Related Works

2.1. CNNs and SegNet Model

2.1.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been known for its the strong abilities in image classification and recognition (He et al., 2015). CNNs are trained through their convolutional layers to recognize various patterns in the input images. Convolutional layers are followed by a fully connected neural network, which is used to translate those features obtained from the previous layers to the given output phases (Karimpouli & Tahmasebi, 2019).

Some common layers used in modern CNNs are described below (Karimpouli & Tahmasebi, 2019).

- **Input layer:** Images are considered as input data, which are introduced to CNN in this layer.
- **Convolutional layer:** In this layer, input images or feature maps from the last layer are convolved with some small size filters (or kernels) to generate new feature maps. These convolutions are being performed with a shift of “n” pixels, which are called stride (Krizhevsky et al., 2012).
- **Batch Normalisation:** Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. This layer is used to speed up training of the convolutional neural network and reduce the sensitivity to network initialization.
- **ReLU (Rectified Linear Units) Layer:** The purpose of this layer is to introduce nonlinearity to a system that basically has just been computing by linear operations (multiplications and summations) during the convolutional layers. The ReLU layer applies the function $f(x) = \max(0, x)$ to all the values in the input volume. The logic behind ReLU is that this layer changes all the negative values to zero.

- **Max-pooling layer:** This layer is a form of down-sampling, where it is used to summarize data by choosing the local maximum in a sliding window moving across the feature maps with a stride of the same length.
- **Fully connected layer:** It is used to translate feature maps or patterns obtained in previous layers to a classification.
- **Soft-max layer:** The soft-max or normalized exponential function is another activation function, which produces a categorical probability distribution such that the total sum of the outputs is equal to one. This layer is located in the final layer (Karimpouli & Tahmasebi, 2019).

2.1.2. SegNet Model

In modern machine learning, encoder-decoder architectures has the ability to show solid performance on sequence-to-sequence tasks such as machine translation (Bahdanau et al., 2016; Luong et al., 2015), language modeling (Raffel et al., 2020), speech-to-text (Chiu et al., 2018), etc. (Aitken et al., 2021). Encoder-decoder architectures consist of two major components: an encoder that takes the input, and a decoder that acts as a conditional language model, taking in the encoded input and the leftwards context of the target sequence and predicting the subsequent token in the target sequence (Aitken et al., 2021). However, the input of the encoder-decoder architecture does not limit at a sequence of words like the classical version, but the concept itself can also be used in image processing. The SegNet architecture had successfully performed semantic segmentation using the idea of encoder-decoder architecture (Badrinarayanan et al., 2017).

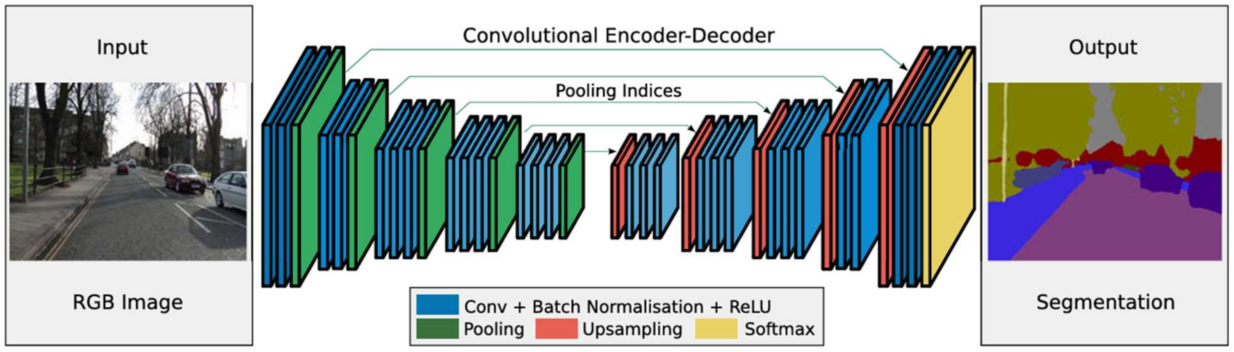


Figure 2.11. An illustration of the SegNet architecture (Badrinarayanan et al., 2017)

Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps. After that, it continues with a *batchNormalization*, then following with rectified-linear non-linearity *ReLU* $\max(x, 0)$ layer. Next, a *maxPooling* layer with a 2×2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. Max pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large input image context (spatial window) for each pixel in the feature map (Badrinarayanan et al., 2017). These layers will be repeated for several time as encoder blocks, before transferring the pooling features into the decoder.

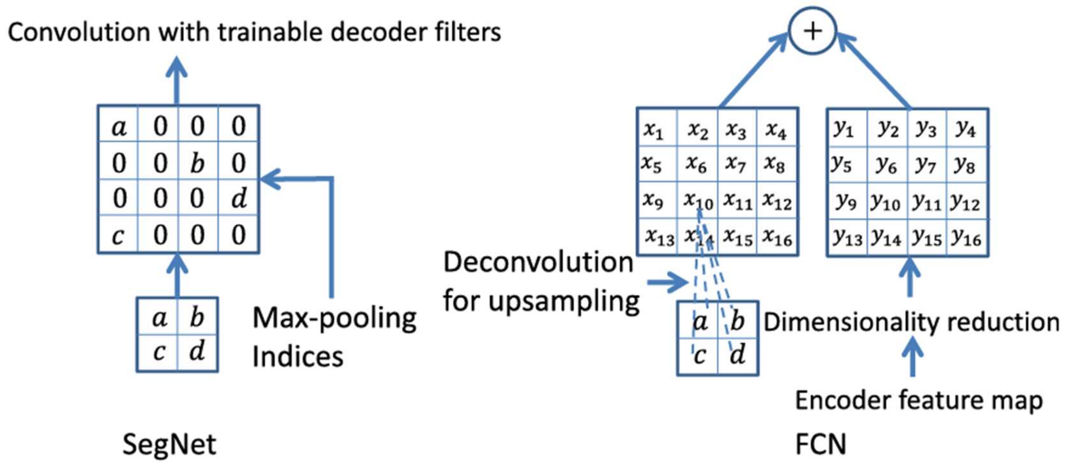

Convolution with trainable decoder filters
a 0 0 0
0 0 b 0
0 0 0 d
c 0 0 0
a b
c d
Max-pooling Indices
SegNet
+
x_1 x_2 x_3 x_4
x_5 x_6 x_7 x_8
x_9 x_{10} x_{11} x_{12}
x_{13} x_{14} x_{15} x_{16}
y_1 y_2 y_3 y_4
y_5 y_6 y_7 y_8
y_9 y_{10} y_{11} y_{12}
y_{13} y_{14} y_{15} y_{16}
Deconvolution for upsampling
a b
c d
Dimensionality reduction
Encoder feature map
FCN

Figure 2.12. An illustration of SegNet and FCN [2] decoders. a; b; c; d correspond to values in a feature map (Badrinarayanan et al., 2017).

The decoder in the network up samples its input feature map(s) using the memorized max pooling indices from the corresponding encoder feature map(s). This step produces sparse feature map. This Seg-Net decoding technique is illustrated in Figure 2.12. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A *batchNormalization* step is then applied to each of these maps. The high dimensional feature representation at the output of the final decoder is fed to a train soft-max classifier. This *softmax* classifies each pixel independently. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel (Badrinarayanan et al., 2017).

2.2. Superpixel Approaches

2.2.1. The first introduced and how it was used

Another interesting approach can be conducted that is researching the learning performance of the Convolution Neural Networks on not only pixel, but also “superpixel”. Basically, superpixel is a term which refers to a group of pixels that share common features. The idea of superpixel was introduced in a paper published in 2003 by Ren and Malik named “Learning a classification model for segmentation”. They claimed that this is a result of oversegmentation, when the image was grouping in a very large number of partitions. The motivations of this preliminary grouping are:

- Pixels are not natural entities; they are merely a consequence of the discrete representation of images (Ren & Malik, 2003).
- The number of pixels is high even at moderate resolutions; this makes optimization on the level of pixels intractable (Ren & Malik, 2003).

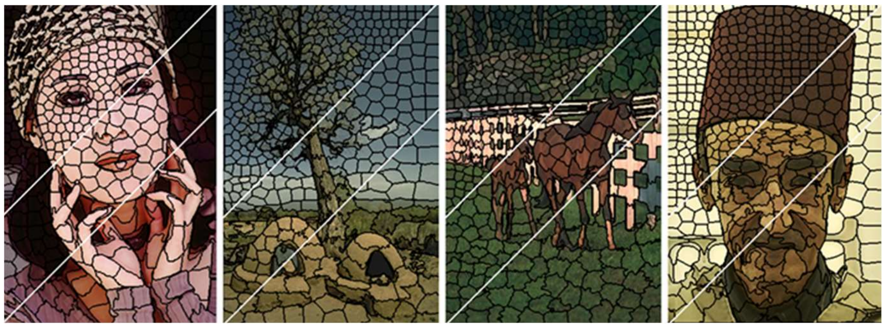


Figure 2.4.1. Example of superpixels (Zhengqin Li & Jiansheng Chen, 2015)

Ren and Malik described ideal superpixels as local, coherent, and preserve most of the structure necessary for segmentation at the point of interest. The main reason why Ren and Malik introduced superpixel is that superpixel would be used as a preprocessing step before image classification training step. They observed that the superpixels are roughly homogeneous in shape and size, and this will simplify the computation in later stages.

Before having a look at some common and potential options for superpixel, it is crucial to see how Ren and Malik created the superpixels and used them in image classification model for the first time. First and foremost, the authors created a superpixel map of 200 superpixels by a Normalized Cut algorithm. To be more specific, a Normalized Cut is a segmentation algorithm that treats the image as a graph where each pixel is a node. The edges between each node represent their likelihood of two pixels being in the same class. After that, the algorithm will eliminate the small values of edges to create the segments in the image, and in this case, Ren and Malik created the “superpixel” map (Ren & Malik, 2003). Therefore, instead of dealing with every pixel in the image, they would create the segments from these superpixels. After that, they chose the features for every segment based on Gestalt principles: proximity, similarity, good continuation, closure, symmetry, and parallelism. The similarity rule had two concepts: inter-region and intra-region. This will be discussed more in section 2.2.3. The features had been chosen were:

- Texture Similarity:

The texture difference of two regions is then determined as the distance X^2 between two histograms. Consequently, X^2 distance would be converted to a log likelihood ratio: let Ω_{same} indicated a set of all superpixels pairs (q_1, q_2) that they appear in the same segment of a human segmentation, and let Ω_{diff} indicated the set of all superpixels pairs such that they appear in different segments in a human segmentation. The next step was computing X^2 distance d_T for all pairs of superpixels in Ω_{same}, then denote the distribution of these X^2 distances as P_{same}. After that, compute P_{diff} , the distribution of X^2 distances on the set Ω_{diff} similarly. Let $d_T(q, S)$ be the X^2 distance between the texture histogram of a superpixel q and S is defined as:

$$T(q,S) = \log \frac{P_{same}(d_T(q,S))}{P_{diff}(d_T(q,S))}$$

Therefore, the basic texture similarity measure $T(q, S)$ to define texture two texture features for a segment S, then intra-region texture similarity sums over all the superpixels in the region:

$$T_{int}(S) = \sum_{q \in S} T(q,S)$$

After that, the inter-region similarity sums over all the superpixels on ∂S, the boundary of S:

$$T_{ext}(s) = \sum_{q \in \partial S} T(q, S'(q))$$

- Brightness Similarity:

The intra-region brightness similarity and inter-region brightness similarity are defined similarly. The brightness descriptor for each region is a histogram of brightness values. The distance of histogram of brightness values X^2 can be computed and converted to a log likelihood ratio by using empirical data.

- Contour Energy:

The contour energy can be computed by determining orientation energy (Malik et al., 2001). The stronger orientation energy between two pixels $p1$ and $p2$, the truer the hypothesis that $p1$ and $p2$ belong to the same group. After that, the orientation energy was converted to a soft "contourness", p_{con} by a non-linear transform (Ren & Malik, 2002). The inter-region contour energy $E_{ext}$is the total of $p_{con}(x)$ over all the pixels on the boundary of S, and the intra-region contour energy E_{int} is the total of $p_{con}(x)$ over all the pixels on the superpixel boundaries inside S.

- Curvilinear continuity:

For each adjacent pair of superpixels q and q' on the boundary of a segmentation S, there is a change of tangent at the junction with an angle α as figure 2.4.2 below.

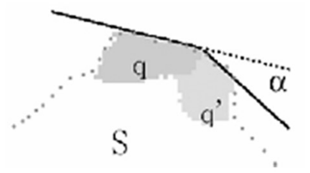


Figure 2.4.2. Curvilinear continuity of is measured by the tangent changes at superpixel junctions along the boundary of S.

The larger the angle α, the less smooth the boundary of S. Let $J(S)$ be the set of all superpixel junctions on the boundary of S, the curvilinear continuity $C(S)$ is de fined as:

$$C(S) = \sum_{(q,q')\in J(S)} \log P_{tangent}(\alpha(q,q'))$$

where $P\_tangent(\alpha)$ describe the distribution of tangent, and this can be achieved from the boundaries.

- Normalizing the features:

After predicting the features, the final step was normalizing them so they can be directly compared to each other on the same scale. To normalize a feature function $F(S)$, the normalization $N\big(F(S)\big)$ would be:

$$N\big(F(S)\big) = \frac{F(S) - m\mu}{\sqrt{m}\sigma}$$

where μ is the mean and σ^2 is the variance for all pairs of a superpixel in its segment, and the maximum likelihood estimates of μ and σ are used.

Before training, Ren and Malik claimed that the cue of boundary contours was the most important grouping cue. Besides, the texture and brightness cues are approximately equally informative. Intra-region and inter-region were a powerful combination that contributed significantly to the result. Curvilinear had shown themselves as a vital cue as well, when they could be able to detect bad segmentations. After that, a classifier $G(S)$ was trained to compute the segmentation quality of segment S.

$$G(S) = \sum_j c_j F_j(S) - \theta$$

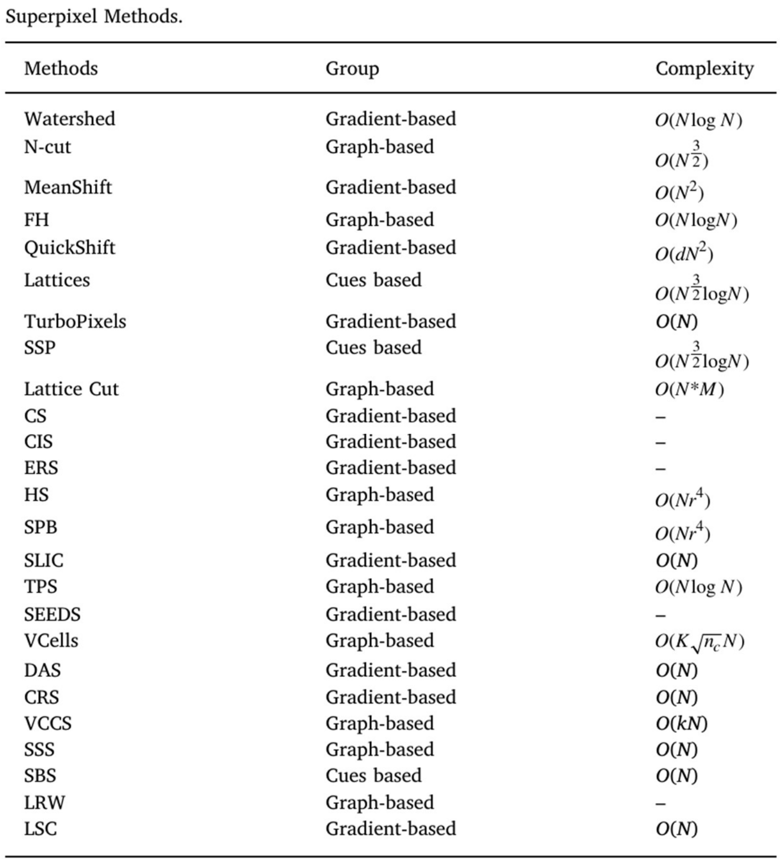
Let $f(S)$ is the total of all classifier function $G(S)$:

$$f(S) = \sum_{Segment \in S} G(Segment) = \sum_{Segment \in S} \left(\sum_j c_j F_j(Segment) - \theta \right)$$

The main idea for the algorithm is finding a segmentation to optimize f in the space of all segmentations. For every step, they created a new segment S', the algorithm would pick random a move to manipulate the superpixels or segments: shift a superpixel from a segment to another, merge two adjacent segments, or split a segment into two superpixels. After creating a new segment S', it would be accepting if $f(S') > f(S)$. However, when $f(S') \leq f(S)$, they accepted with probability $\exp\left(\frac{f(S')-f(S)}{T}\right)$ where T is the temperature which got decreased linearly over time. Overall, that was how Ren and Malik made use of superpixels and this concept could be used to deal with TADA diagrams.

2.2.2. Superpixel Benchmark

Superpixels had been an effective method to simplify the complexity of various image processing tasks. There are multiple algorithms for superpixel creation, separated into two categories: graph-based method and clustering-based method. In 2017, a paper was published name "Superpixel segmentation: A benchmark" by Wang et al, which comparing algorithms and evaluate their effectiveness. This is not the first benchmark on superpixel, because in 2012, Neubert and Chemnitz also published one benchmark with fewer algorithms (Neubert & Chemnitz, 2012). It is worth to note that in the figure Wang et al provided (Figure 2.4.3), the meaning of N is different based on the definition of the corresponding algorithm indicated in Wang et al. paper.



Superpixel Methods.

Methods	Group	Complexity
Watershed	Gradient-based	$O(N\log N)$
N-cut	Graph-based	$O(N^{\frac{3}{2}})$
MeanShift	Gradient-based	$O(N^2)$
FH	Graph-based	$O(N\log N)$
QuickShift	Gradient-based	$O(dN^2)$
Lattices	Cues based	$O(N^{\frac{3}{2}}\log N)$
TurboPixels	Gradient-based	$O(N)$
SSP	Cues based	$O(N^{\frac{3}{2}}\log N)$
Lattice Cut	Graph-based	$O(N*M)$
CS	Gradient-based	–
CIS	Gradient-based	–
ERS	Gradient-based	–
HS	Graph-based	$O(Nr^4)$
SPB	Graph-based	$O(Nr^4)$
SLIC	Gradient-based	$O(N)$
TPS	Graph-based	$O(N\log N)$
SEEDS	Gradient-based	–
VCells	Graph-based	$O(K\sqrt{n_c}N)$
DAS	Gradient-based	$O(N)$
CRS	Gradient-based	$O(N)$
VCCS	Graph-based	$O(kN)$
SSS	Graph-based	$O(N)$
SBS	Cues based	$O(N)$
LRW	Graph-based	–
LSC	Gradient-based	$O(N)$

Figure 2.4.3. Superpixel method with their group and complexity, and note the N is different for every algorithm (Wang et al., 2017).

Start from the time complexity, Wang et al listed all algorithms and from the time complexity, there are some algorithms seems to have more potential to have better performance than the others. However, this is not trivial because of the different meaning of N in each situation, and this is just theoretical worst-case scenario complexity. In the 2012 paper, the criteria were the quality of the segmentation compared to human ground truth segmentations and the robustness to affine image transformations (Neubert & Protzel, 2018). Wang et al. figured that it was more categorized to divide the metrics into three main criteria to rate the algorithm with the total of 13 metrics in total.

- The first criterion is segmentation quality metrics, focuses on measuring algorithms the segmentation accuracy. The algorithms will be checked on two sub-criteria: one is how the segmentation results adhere to the boundary (by a Precision-Recall metric), and second is the variation of pixels in the segmentations by Variation of Information (VI), Probabilistic Rand Index (PRI) and Segmentation Covering (SC) metrics (Wang et al., 2017)
- The second criterion is superpixel quality metrics, where superpixels are usually used as a pre-processing step in image processing, so they should preserve image information as much as possible. Coherence, compactness and regularity are the most important properties for good superpixels (Hanbury, 2008). Therefore, undersegmentation error, achievable segmentation accuracy, compactness and explained variation, regularity index are the most popular metrics for superpixel quality testing (Wang et al., 2017).
- The third criterion is runtime.

After the evaluation, the authors concluded that graph based superpixel algorithm had a better performance than clustering algorithms on segmentation accuracy. Moreover, there are some algorithms which add compact constraints to objectness function such as VCells and CIS, perform slightly differently in producing compactness, coherent, and regular superpixels. In terms of runtime, it experienced a phenomenon where accurate algorithms namely LRW, SEEDS, Lattice Cut had slower run time than the others. Most of the running time is dependent on the number of superpixels, the largest can reach up to thousands of superpixels, and graph-based algorithms substantially increase the run time with large number of superpixels.

Overall, this is a good reference to have an overview of superpixel segmentation, and the criteria of evaluation whether a superpixel generation is suitable for the research. The following section will introduce an alternative way of generating the superpixel, which used the idea of the model with encoder-decoder design mentioned above in 2.2.2.

2.2.3. Fully Convolutional Neural Networks application

Different to the original idea from Ren and Malik, a paper named "Superpixel Segmentation with Fully Convolutional Network" by Yang et al was published to demonstrate the use of fully convolutional network in superpixel generation.

A common strategy for superpixel segmentation is partitioning an original image using a grid where each cell is an initial superpixel (Tu et al., 2018). After that, the eventual superpixel segmentation is obtained by finding a mapping which assigns each pixel p to one of the superpixel, which was called "seed" in this paper. Considering a pixel p stays at the coordinate (i, j), it is written as $p = (u, v)$, and a seed has its centre at (i, j), it can be written as $s = (i, j)$. Mathematically, if the (u, v)th pixel belongs to the (i, j)th superpixel, it can be written as $g_s(p) = g_{i,j}(u, v) = 1$, and 0 otherwise. Nonetheless, it was computationally costly to compute $g_{i,j}(u, v)$ for all pixel-superpixel pairs (Yang et al., 2020). To replace that expensive method, they put a constraint the search to the set of surrounding grid cells N_p for a given pixel p. Specifically, for each pixel p, they only consider the 9 grid cells for assignment (Figure 4.4.5). Therefore, this mapping can be written as a tensor $G \in \mathbb{Z}^{H \times W \times N_p}$ where H, W are the height and width of the original image.

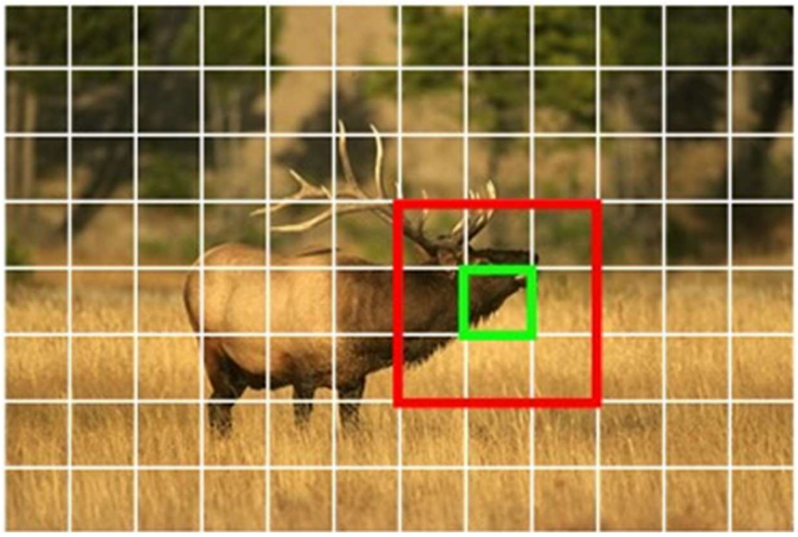


Figure 4.4.5. Illustration of N_p. For each pixel p in the green box, the authors just consider assigning 9 grid cells in the red box (Yang et al., 2020).

To compute G, they directly learned the mapping using a deep neural network and replace hard assignment G with a soft association map $Q \in \mathbb{R}^{H \times W \times |N_p|}$. Therefore, the entry $q_s(p)$ represents the probability that a pixel p is assigned to each seed $s \in N_p$ such that $\sum_{s \in N_p} q_s(p) = 1$. Lastly, the superpixel were obtained by assigning each pixel to the grid cell with the highest probability: $s* = argmax_s q_s(p)$.

The deep neural network model they used following the design of standard encoder-decoder design to create the final superpixel map. The visualisation of the model can be found in Figure 4.4.6. The encoder takes a color image as input and produces high-level feature maps via a convolutional network. The decoder then gradually up samples the feature maps via deconvolutional layers to make final prediction, considering also the features from corresponding encoder layers. They used *LeakyReLU* for all layers except for the prediction layer, where *SoftMax* is applied (Yang et al., 2020).

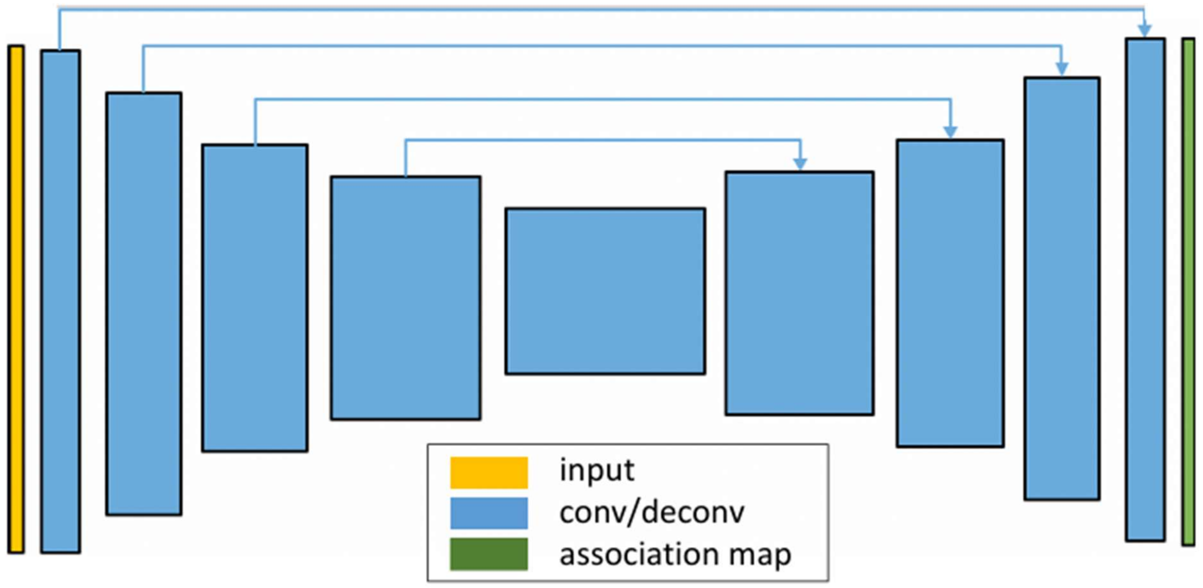


Figure 4.4.6. The encoder-decoder design architecture for the model (Yang et al., 2020)

Given that $f(p)$ is the pixel features of its corresponding superpixel, and p is the position of a pixel given by its coordinate x, y. For instance, $f(p)$ can demonstrate a 3-dimensional CIELAB color vector, and/or a N-dimensional one-hot encoding vector of semantic labels, where N is the number of classes, and many others (Yang et al., 2020). The loss function of the model can be calculated as:

$$L(Q) = \sum_{\mathbf{p}} \text{dist}(\mathbf{f}(\mathbf{p}), \mathbf{f}'(\mathbf{p})) + \frac{m}{S}\|\mathbf{p} - \mathbf{p}'\|_2$$

where S is the superpixel sampling interval, and m is a weight balancing the two terms.

Overall, superpixel is a potential concept for handling the semantic segmentation. A potential idea will start with choosing superpixel generation algorithm, then conduct the superpixel map, then find the features of each superpixel then classify them. For generating superpixel, a fully conventional neural network can be used to learn the features of the image and cluster them into superpixel, it can be algorithm from Yang et al, then we can assign the classification algorithm on the superpixel and see how well the performance of semantic segmentations using the concept of superpixel segmentation.

2.3. Gestalt Theory

In order to be able to segment images, it is necessary to have a step back and figure out the features of similarity between two pixels, to be able to determine the different learning patterns from a human perspective in identifying groups and objects. First and foremost, it is worth looking at the famous Gestalt theory and its relation to image segmentation. When Wertheimer published his paper in 1912, and it was considered the first appearance of Gestalt theory (Wagemans et al., 2012). Gestalt theory started with the assumption of active grouping laws in visual perception (Wertheimer, 1923). The list of elementary grouping laws given by Wertheimer and Gaetano Kanizsa in Grammatica del Vedere is:

- Vicinity (*vicinanza*): The word "vicinanza" translates into English is "proximity", and this rule is also known as the proximity rule, one of the most important rules to apply into Semantic Segmentation. The vicinity law applies when distance between objects is small enough with respect to the rest. In other words, objects can be considered in the same group when they are closed to each other and otherwise (Desolneux et al., 2008).

Figure 2.1. According to vicinity/proximity law, there are only two groups of black dots (Desolneux et al., 2008). People usually not consider these as 20+ groups of black dots, because some of them are fairly near each other, and only one huge distance in the middle.

- Similarity (*somiglianza*): The similarity law is one of the most important rules that will be used in this project. It disparate groups based on the difference between groups and similarities within the groups. There are two main concepts in similarity rule:
    - Intra-region similarity: the elements in a region are similar. This includes similar brightness, similar texture, and low contour energy inside the region.

- Inter-region dissimilarity: the elements in different regions are dissimilar. This in turn includes dissimilar brightness, dissimilar texture, and high contour energy on region boundaries (Ren & Malik, 2003).

Figure 2.2. The similarity rules make people's mind think that there are two groups in each picture *a)* and *b)*. For *a)*, there are two groups because of the different in shapes, while in *b)*, the different in texture disparate *b)* into two groups (Desolneux et al., 2008).

- Continuity of direction (*continuita di direzione*): Continuity of direction can be applied to an array of objects. This is significantly useful for the project, especially in detecting a path of dots that create a path in the diagram.

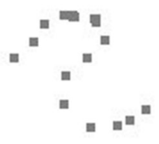


Figure 2.3. The dots create a curve due to the continuity of direction (Desolneux et al., 2008).

- Amodal completion (*completamento amodale*): The amodal completion law applies when a curve stops on another curve, thus creating a "T-junction". In such a case our perception tends to interpret the interrupted curve as the boundary of some object undergoing occlusion. The leg of the T is then extrapolated and connected to another leg in front whenever possible. In other words, people's mind is still able to recognize a group, whether it is covered by another group or object.

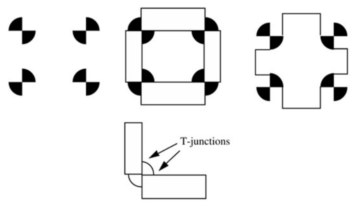


Figure 2.4: T-junctions entail an amodal completion and a completely different image interpretation. It is possible to observe the black circles had been covered by the white objects (Desolneux et al., 2008).

- Closure (*chiusura*): The closure rule reduce the complexity by reducing the number of elements needed to organize and communicate information (Desolneux et al., 2008)

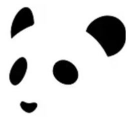


Figure 2.5. Instead of displaying a full border of a panda picture, just by five shapes, people still can be able to recognize that this is a panda (Desolneux et al., 2008).

- Constant width (*larghezza constante*): Constant width law applies to group the two parallel curves, perceived as the bound aries of a constant width object. This law is constantly in action since it is involved in the perception of writing and drawing.

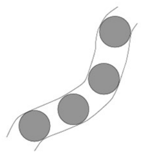


Figure 2.6. Two parallel curves: the width constancy law applies (Desolneux et al., 2008)

- Tendency to convexity (*tendenza alla convessita*): The convexity law, as the closure law, intervenes in people's decision on the figure-background dilemma. Any convex curve (even if not closed) suggests itself as the boundary of a convex body.

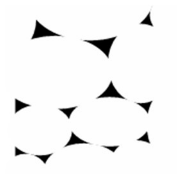


Figure 2.7: There are two ways of interpreting this figure, which are white ovals on black background, or black triangles on white background (Desolneux et al., 2008).

- Symmetry (*simmetria*): symmetry law applies to group any set of objects which is symmetric with respect to some straight line(axial symmetry) or a point(point symmetry). This is a useful principle, but it is hard to apply.

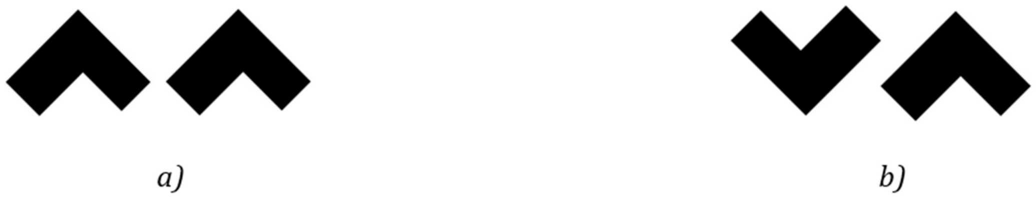


Figure 2.8: In *a)*, the image is axial symmetric through an invisible line, while in *b)*, it is symmetric through an invisible point.

- Common motion (*movimento solidale*): This relates to a lot of other concepts of later Gestalt theory, such as parallelism, linear and co-linear. Basically, this concept refers that the motion of objects also disparate themselves into groups, but this is not very relevant in this project.

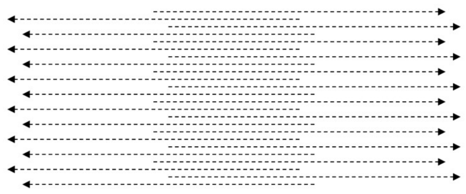


Figure 2.9. There are two groups in this image, which are two group of arrows, not various small lines and triangles (Desolneux et al., 2008).

- Color constancy: The color constancy law is a crucial and useful rule for the project that states that connected regions where luminance or color does not vary strongly are unified.

Figure 2.10. According to the color constancy law, this can be thousands of black dots, or a single back shape (Desolneux et al., 2008).

- Past experience (*esperienza passata*): The past experience rule basically refers to shapes that people have seen, and by looking people can recognize that quickly. In terms of artificial intelligent (AI), with enough data learning, AI can also recognize shapes and image.

Gestalt theory is an interesting and useful concept, that it indicates the human perspective of determining groups of objects. Furthermore, this is a potential idea to deal with more complicated situation, where the CNN models can also learn these patterns of disparate objects and perform better with several methods, such as when combining with the Normalized Cut, SLIC algorithms (mentioned in Section 2.2.2.), etc.

3. Architecture of approaches

The goal of each approach is performing semantic segmentation, which is a classification problem in which the algorithm must assign each pixel of the image to its corresponding class of image features. Based on the potential demonstrated during the research period, we have decided to carry out the project primarily using materials from deep learning concepts, divided into two major parts:

- Firstly, we want to see the performance of a simple convolutional neural network model on the raw data, and also to get used to the process of implementing deep learning model using Python libraries such as tensorflow, numpy, etc.

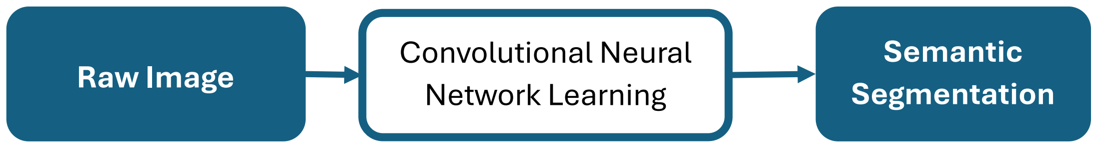


Figure 3.1. Plan pipeline for the first part of the project.

- Secondly, we want to evaluate the performance of superpixel segmentation then assign corresponding classes to the superpixel using a convolutional neural network.

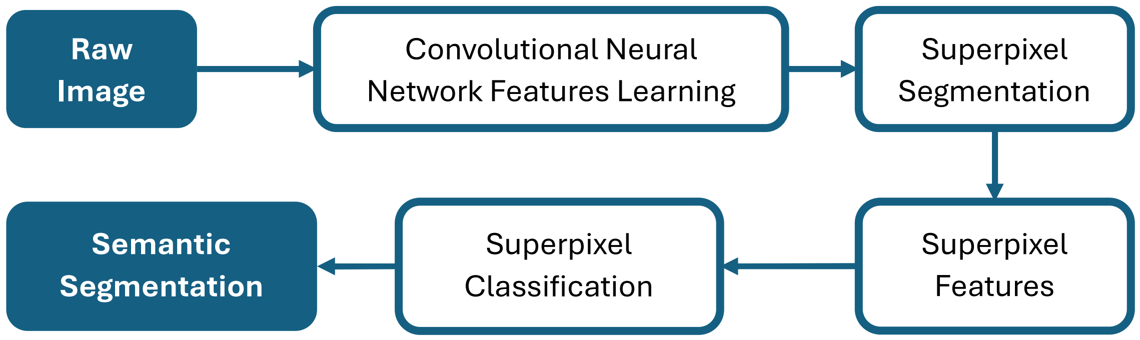


Figure 3.2. Plan pipeline for the second part of the project.

3.1. Input Data Generation

The data set will include a large number of pictures of graphs with random nodes, edges, and texts. The data generation part of thesis was created thanks to my supervisor Dr. Cormier, Yichun Zhao, Laura Paul, and another fellow honour student Cameron Swift.

- In terms of nodes, they can be in a random shapes in a set of *['box', 'polygon', 'ellipse', 'oval', 'circle', 'egg', 'triangle', 'diamond', 'trapezium', 'parallelogram', 'pentagon', 'hexagon', 'septagon', 'octagon', 'doublecircle', 'doubleoctagon', 'tripleoctagon', 'invtriangle', 'invtrapezium', 'rect', 'rectangle', 'square', 'star', 'note', 'tab', 'folder', 'box3d', 'component'].* Furthermore, it will also be assigned to a random colour.
- About generating edges, they can be in any length, width and colour. The edges will be randomly assigned to be the connection between a pair of nodes, or itself (self-loop).

- There are numerous fonts for text, such as *"Times-Roman", "Courier", "Helvetica", "Palatino-Roman", "Bookman-Demi", "AvantGarde-Book", "NewCenturySchlbk-Roman"*. Texts are divided into two main types:
    - Text coordinates with nodes will have a larger size and a random colour.
    - Text coordinates with edges will have a smaller size and a random colour.

In this research, we decided that the colours will not correspond to the classes of the input data. Therefore, we can have a situation where the model cannot be based on the colour of a pixel to figure out its class, so it must use the structure of the image for learning progress.

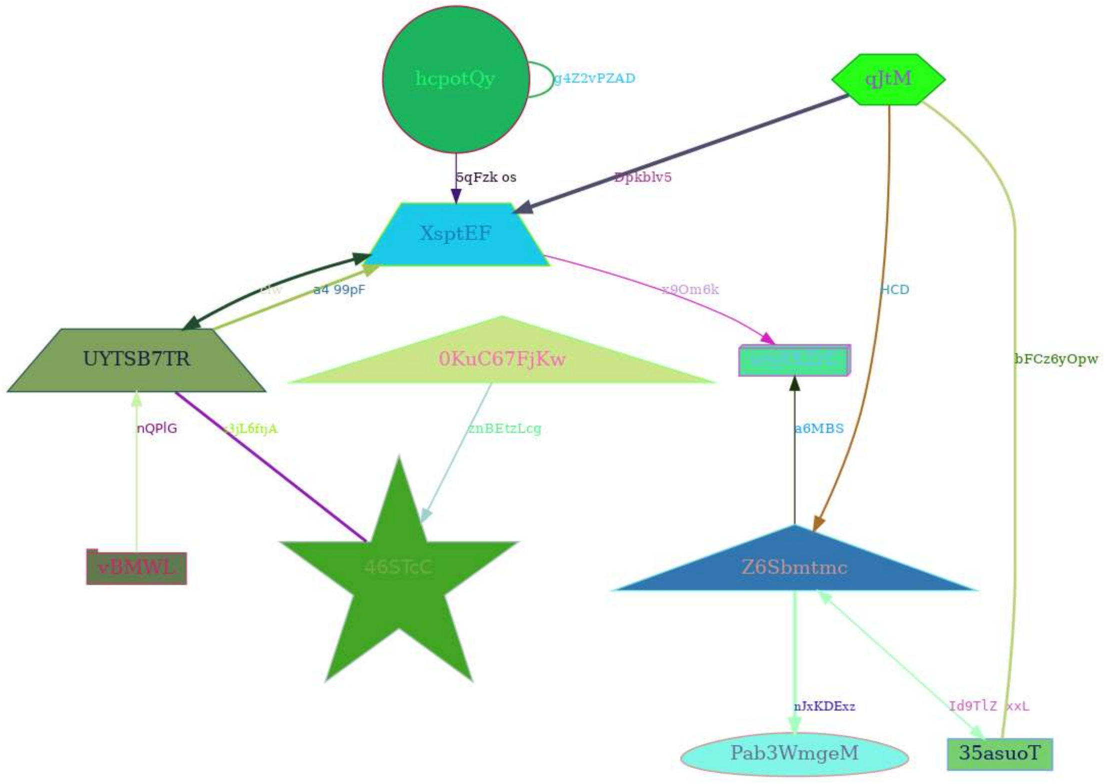


Figure 3.3. An example of input data as image

[145,123,76]	[22,2,185]	...	[48,95,236]	[149,122,110]	[237,188,158]
[139,227,206]	[235,135,98]	...	[111,75,41]	[150,32,28]	[34,255,13]
[98,194,180]	[26,137,65]	...	[193,77,16]	[146,137,47]	[6,174,42]
[23,247,220]	[149,211,17]	...	[100,128,158]	[213,37,41]	[184,159,207]

⋮	⋮		⋮	⋮	⋮
[66,230,61]	[215,44,151]	...	[43,55,138]	[75,102,147]	[88,153,26]

Figure 3.4. An example of input data as array A, where each pixel $A_{x,y}$ includes 3 information of RGB values as $[red, green, blue]$.

The label of each input graphs will demonstrate 5 main classes that each pixel belongs to, which are also the components of the input data:

1. Background: represented as class 0
2. Text: represented as class 1.
3. Edge: represented as class 2.
4. Node interior: represented as class 3.
5. Border of a node: represented as class 4.

Note that this is the assumption were made that every diagram specified towards the TADA project will follow this class set, and we can spend time to work on more complicated classes as future work. Furthermore, all TADA project's diagrams will be imported/exported in the environment of computer graphics, and they will not be imported/exported as paper work or scanning documents.

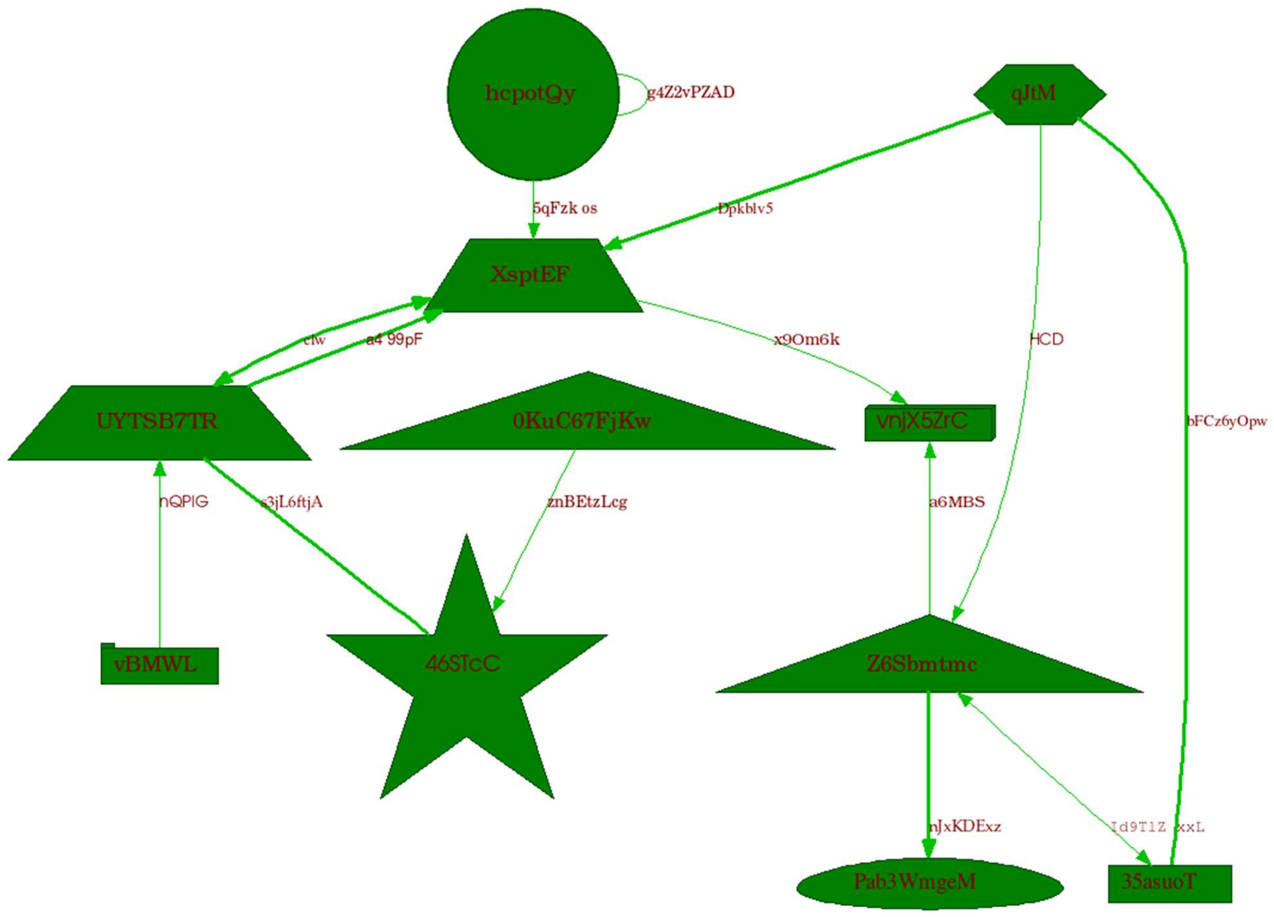


Figure 3.5. An example of the label as image

$$\begin{array}{cccccc}
[1,0,0,0,0] & [1,0,0,0,0] & \ldots & [0,0,0,1,0] & [0,0,0,0,1] & [0,0,0,1,0] \\
[1,0,0,0,0] & [1,0,0,0,0] & \ldots & [0,0,0,1,0] & [0,0,0,0,1] & [0,0,0,1,0] \\
[1,0,0,0,0] & [1,0,0,0,0] & \ldots & [0,0,0,1,0] & [0,0,0,0,1] & [0,0,0,1,0] \\
[1,0,0,0,0] & [0,0,0,1,0] & \ldots & [0,1,0,0,0] & [0,1,0,0,0] & [0,1,0,0,0] \\
\vdots & \vdots & & \vdots & \vdots & \vdots \\
[1,0,0,0,0] & [0,0,1,0,0] & \ldots & [1,0,0,0,0] & [0,1,0,0,0] & [0,1,0,0,0]
\end{array}$$

Figure 3.6. An example of input data as array A, where each pixel $A_{x,y}$ includes 5 probabilities value of which class is it $[\ p_{background},\ p_{text},\ p_{edge},\ p_{interior},\ p_{border}]$

The data will be divided into smaller patches for the models to learn and validate. The standard size of a patch in our experiments is 128×128. A patch will include 2 parts, a graph input and the label input:

- The graph input patch pixel has 3 colour values as Red, Green, Blue. Therefore, the shape of one single patch of graph input is (128,128,3).
- The label input patch pixel has 5 probabilities value $[p_{background}, p_{text}, p_{edge}, p_{interior}, p_{border}]$ as the probability of being classified as each component in the graph input.

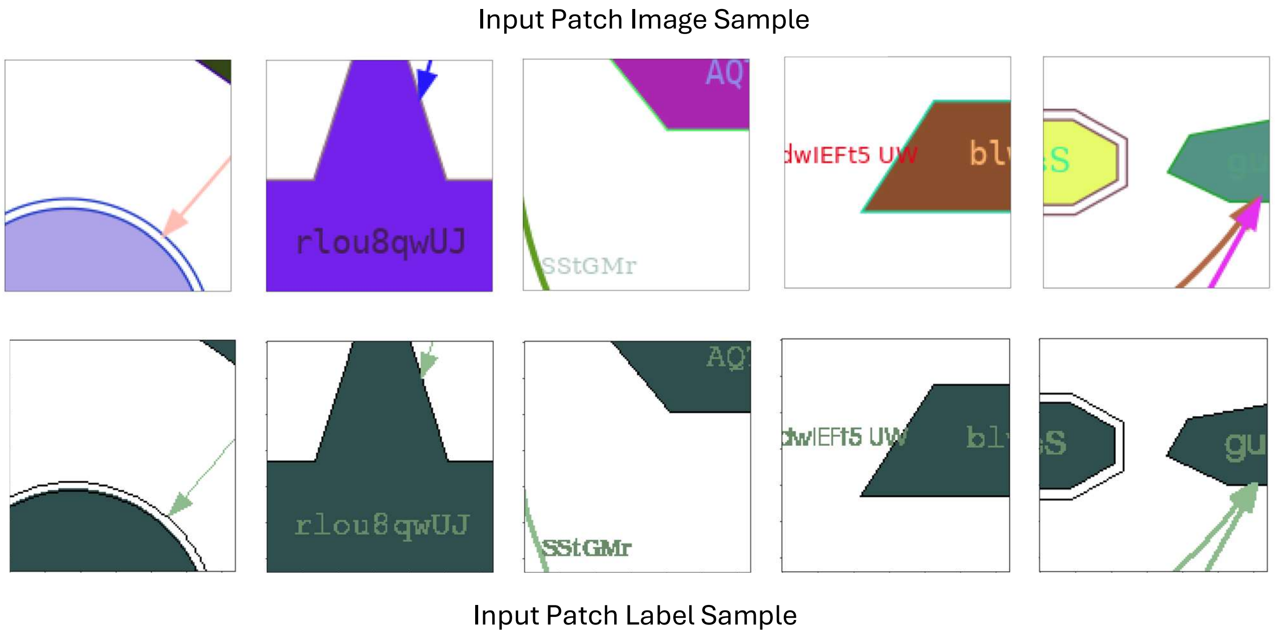


Figure 3.7. A sample set of input patches

There are 3 types of data: training, validation and testing data:

- Training: There are 2500 images divided into 10000 smaller patches with size 128×128 along side with their labels.
- Validation and Testing: There are 1000 images divided into 3000 smaller patches of image with size 128×128 along side with their labels for each of validation and testing set.

3.2. A Convolution Neural Networks approach

Being inspired by the model architecture of Karimpouli and Tahmasebi, which also based on the basic *SegNet* model introduced in section 2.2. by Badrinarayanan, we conducted our first model V1 for the earlier state of the experiments. The original model of Karimpouli and Tahmasebi is illustrated in Figure 3.8.

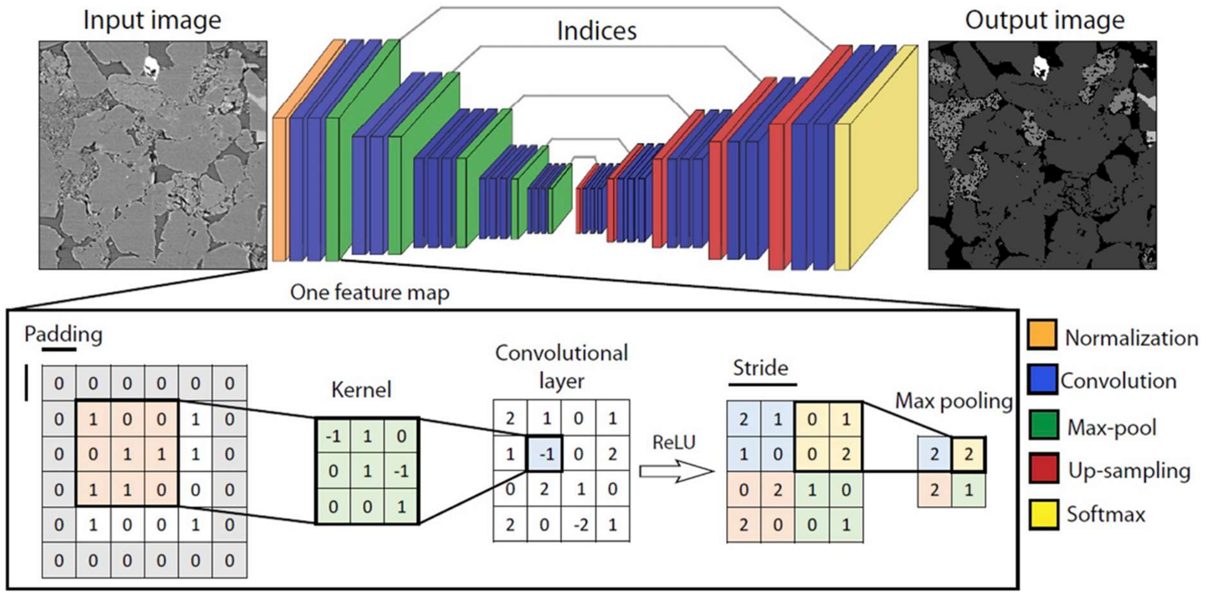


Figure 3.8. The general architecture of Karimpouli and Tahmasebi's SegNet (Karimpouli & Tahmasebi, 2019).

There are two adjustments that we made in model V1 compared to the *SegNet* architecture to fit the model to our research and avoid overfitting. We observed that there are some signs of overfitting in the model when handling with the input data. Therefore, we have added in some Dropout Layers after *MaxPooling* and *UpSampling* Layers. The purpose of these Dropout Layers is temporarily removing a set of nodes (hidden and visible) randomly in the neural network, along with all its incoming and outgoing connections (Srivastava et al., 2014). We also significantly simplify the models by eliminating a small number of Convolution Layers, from 20 layers in the original down to 12 layers. Furthermore, we want to try an adjustment when we did not pass the pooling indices from the encoder to the decoder. Therefore, this model is no longer following the encoder-decoder network, and it is a linear model. The final architecture of model V1 is demonstrated in Figure 3.9.

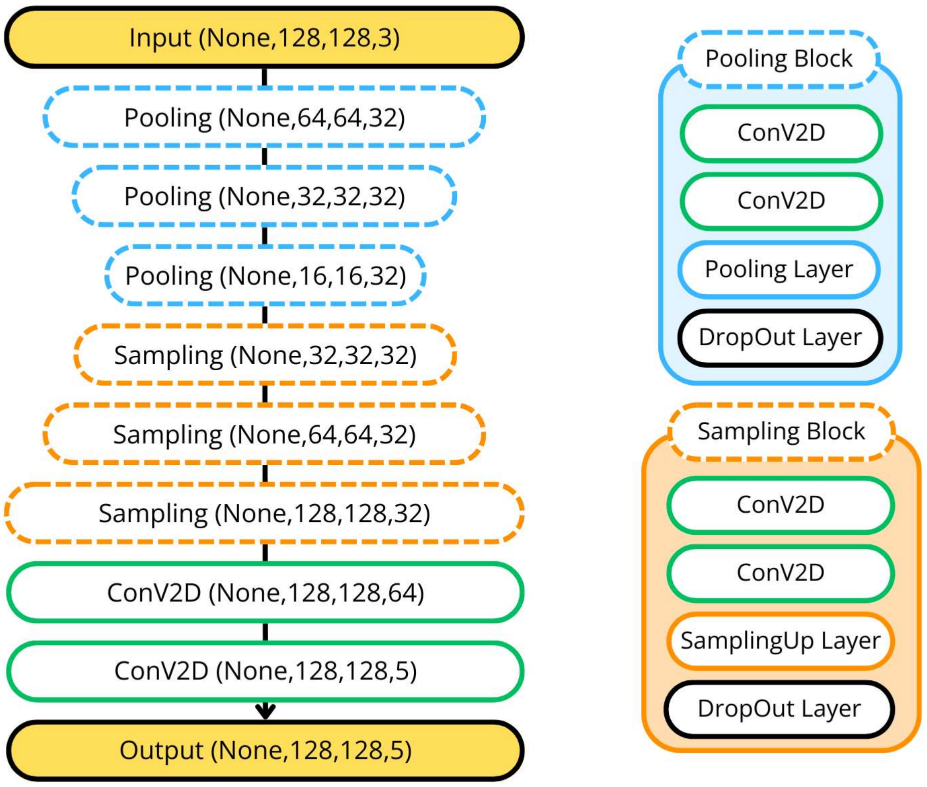


Figure 3.9. The visualisation of model V1, note that the notation of each layer block is represented as (Layer Block Name, Output Size).

3.3. Fully Convolution Neural Networks with Superpixel Segmentation

In order to experiment with superpixel generation with semantic segmentation, we based our model V2 on the Fully Convolutional Neural Network model published by Yang, then added in our custom layers to classify each superpixel, then assign those classes to every pixel for final semantic segmentation result (Yang et al., 2020).

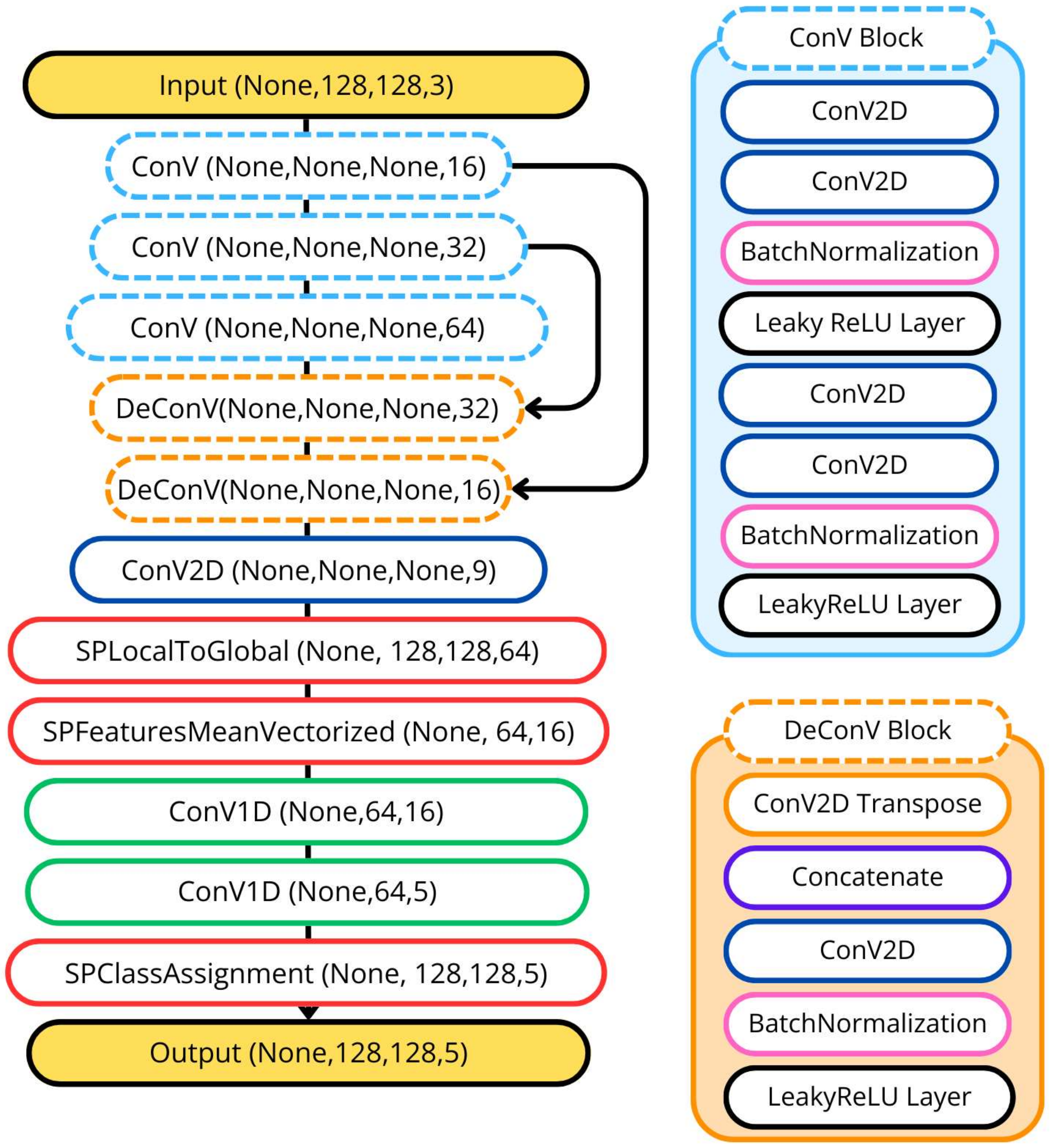


Figure 3.10. The visualisation of model V2, note that the notation of each layer block is represented as (Layer Block Name, Output Size). The superpixel segmentation is the blocks of ConV and DeConV block.

The model passes the raw input into the superpixel segmentation, and the model returns its prediction. There are 8 neighbor pixels of every pixel p, we will call them np_i where $0 \leq i \leq 8$. The result of the superpixel segmentation will be a probability set $\{P_{np_i}, 0 \leq i \leq 8\}$, where

P_{np_i} is the probability of pixel p is associated with pixel np_i in the same superpixel (Yang et al., 2020). The encoder-decoder layers that Yang et al. used had some similarity with the concepts Badrinarayanan introduced, where an encoder block will contain some *ConV2D* Layers, followed by a *BatchNormalization* Layer and a *LeakyReLu* Layer. After that, decoder block (in this case, Yang et al. called it *DeConV*) contains 1 layer of *ConV2D* Transpose, then it concatenated with the result sent along from the encoder, then followed by a *Conv2D*, a *BatchNormalization* Layer and a *LeakyReLu* Layer (Badrinarayanan et al., 2017). Finally, there is a *ConV2D* Layer to convert the learning result into a form of an array of $[P_i]$ where i is the index of the local pixel for every pixel p.

The Layer *SPLocaltoGlobal* performed a soft assignment to every pixel according to its superpixel probability received from the learning layers, then created a superpixel map. After calculating the map, the Layer *SPFeaturesMeanVectorized* will return a series of features values of all superpixels, which determined by the average features of all pixels among the superpixels. Using the mean values of the features of the superpixel, the model can conduct a simple classification task for each superpixel, into the 5 classes. The method of assigning the classification is using 2 *ConV1D* Layers so it can learn the features quickly without causing overfitting to the whole model. Last but not least, after having all superpixel classified, the model maps the class of the superpixel to all pixels, and that results in the final output of the model.

3.4. Fully Convolution Neural Networks with Superpixel Segmentation with revision

After the experiment, model V2 had experienced a slight underperformance in comparison to model V1. Therefore, from realising model V1 had the advantage of fast and accurate learning method, we adapted the architecture of model V1 into the learning part of model V2, and model V3 was born (Figure 3.11.). Basically, model V3 is model V1 with an extension on the final layer, instead of exporting into 5 classes, it will output to 9 classes representing the probability P of every pixel p_i can be associated in the same superpixel to one of the 9 nearby cells. After that, the result will be transferred to superpixel semantic segmentation similar to model V2, then produce the final semantic segmentation prediction. Note that for

Model V3, we also experimented with a smaller size of superpixel and a larger number of superpixels.

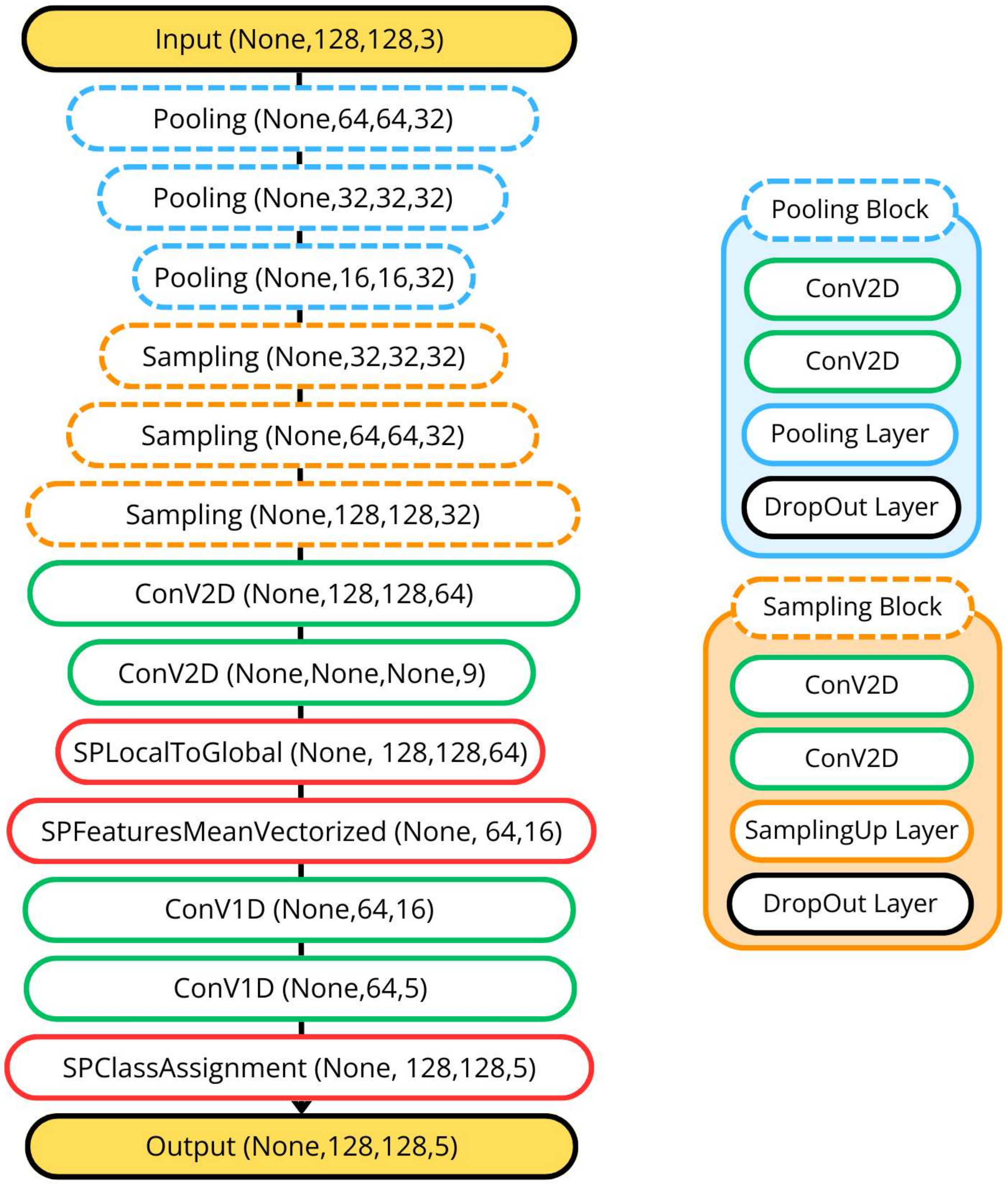


Figure 3.11. The visualisation of model V2, note that the notation of each layer block is represented as (Layer Block Name, Output Size).

4. Experiment and Result

4.1. Overall Performance

The evaluation of a model will be based on the criteria of accuracy, loss, and the quality of the prediction image over several epochs. Epoch is defined as the number times that the learning algorithm will work through the entire training dataset. Accuracy refers to the proportion of correct predictions and the total predictions in both training and validation data. The range of the accuracy value can be varied from 0% (all wrong) to 100% (perfectly predicted). In this project, we are aiming for the accuracy of the models to be around 80%. Secondly, loss demonstrates the difference between the ground truth and the predicted values. This loss value is expected to decrease over learning time. Furthermore, the quality can be evaluated through the comparison between the predicted images and the ground truth image through a confusion matrix. This confusion matrix represents the occurrence of predicting a pixel as class A and its actual value is class B. A true positive happens when class A = class B. The quality of prediction is referring to the occurrences of true positive over all occurrences of the model. Last but not least, we want to test how the models perform with a completely new dataset that it has never seen before. This process is call testing, and will demonstrate its performance through the confusion matrix.

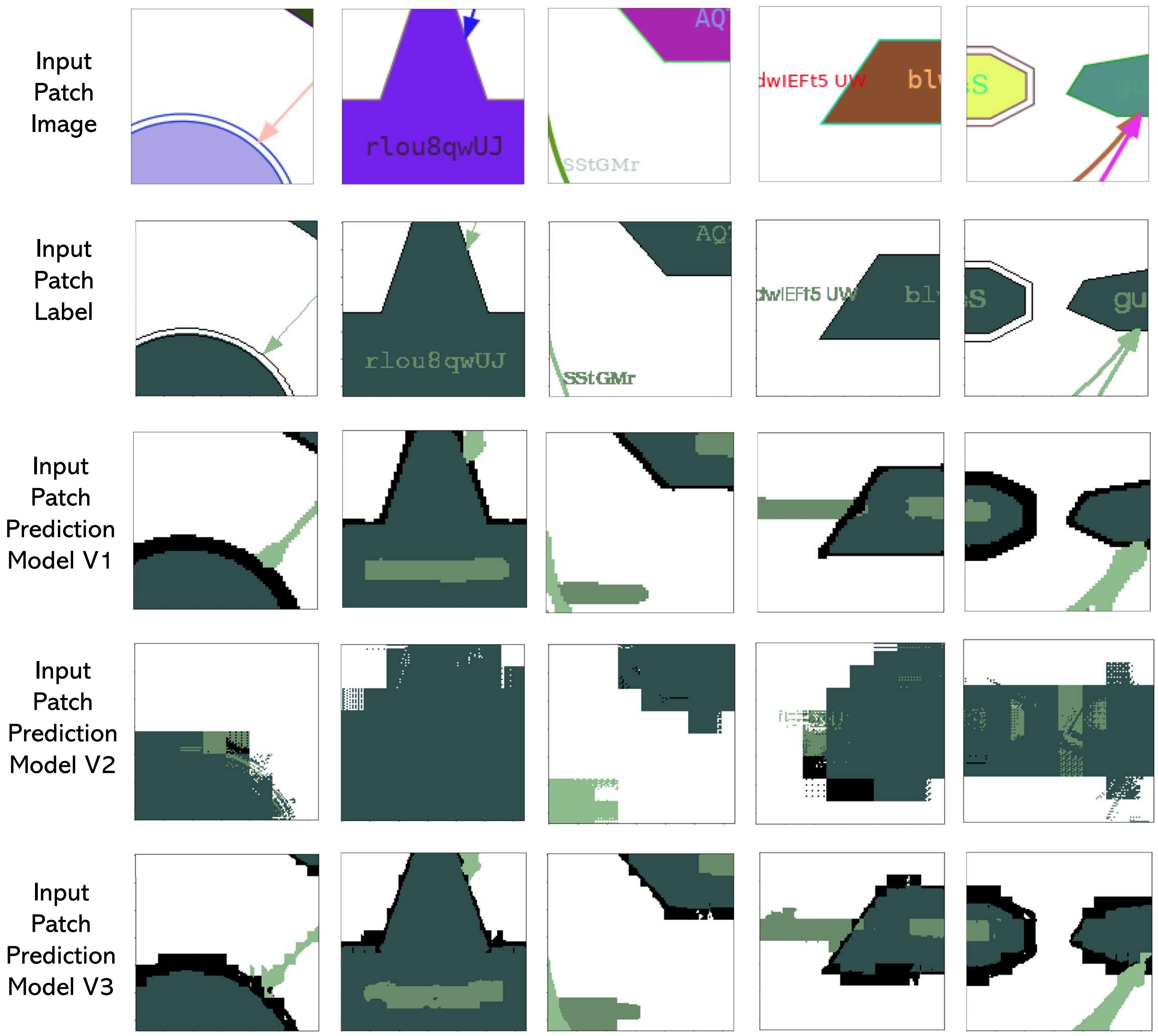

Input Patch Image
Input Patch Label
Input Patch Prediction Model V1
Input Patch Prediction Model V2
Input Patch Prediction Model V3
rlou8qwUJ
AQ
SStGMr
dwIEFt5 UW
bl
S
gu

Figure 4.1. Visualization

4.2. Trial run of the models

Here are the results of our run for each model. Note that the experiment time would be different for each machine, and in this case, the experiments were made in a machine with dual RTX 2080 Ti GPUs.

Run ID	Type	Accuracy	Loss	Total Time (min)
1	Train	0.888	0.2057	45.45
1	Validation	0.8838	0.1936	45.45
2	Train	0.8886	0.2025	46.25
2	Validation	0.8713	0.1893	46.25
3	Train	0.8869	0.2076	45.99
3	Validation	0.8647	0.1926	45.99
4	Train	0.8895	0.2014	45.87
4	Validation	0.8756	0.1934	45.87
5	Train	0.8904	0.2028	45.11
5	Validation	0.8677	0.1929	45.11
6	Train	0.8881	0.2029	45.05
6	Validation	0.8733	0.189	45.05
7	Train	0.8892	0.2034	45.22
7	Validation	0.8769	0.1855	45.22
AVERAGE		**0.881**	**0.1973**	**45.563**

Table 4.1. A sample of Model V1 runs with 256 epochs and learning rate = 2×10^{-5}.

Run ID	Type	Accuracy	Loss	Total Time (hour)
1	Train	0.3583	2.1449	1.406
1	Validation	0.3816	2.0719	1.406
2	Train	0.5031	2.1374	1.403
2	Validation	0.535	2.0413	1.403
3	Train	0.7605	2.1388	1.403
3	Validation	0.773	2.0352	1.403
4	Train	0.6897	2.1861	1.415
4	Validation	0.7117	2.1075	1.415
5	Train	0.65	2.0528	1.397
5	Validation	0.6809	1.9553	1.397
6	Train	0.6402	2.1921	1.395
6	Validation	0.6687	2.1092	1.395
7	Train	0.5639	2.1099	1.406
7	Validation	0.5929	2.0216	1.406
AVERAGE		**0.60782**	**2.0931**	**1.404 ≈ 84.24min**

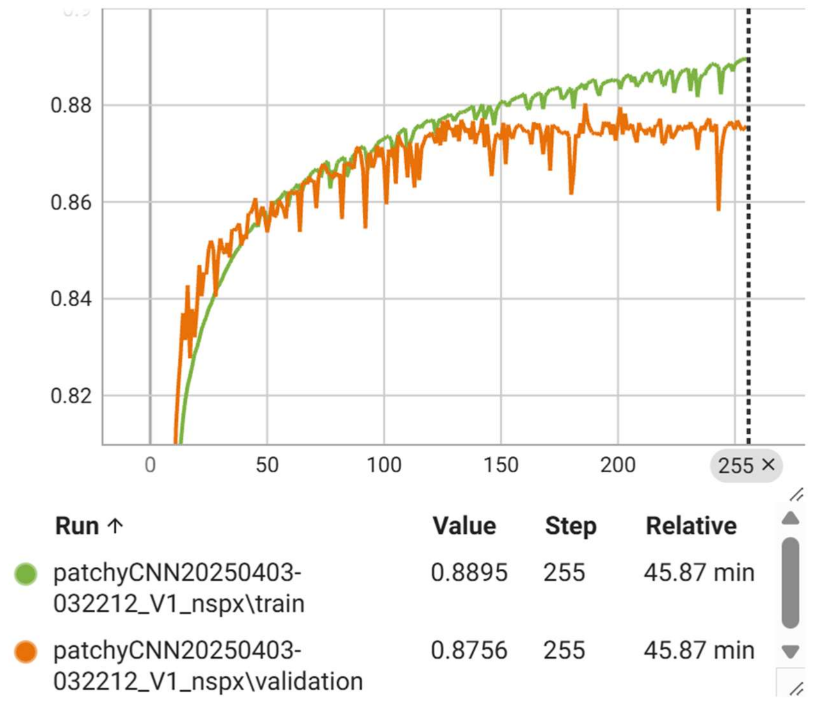
Table 4.2. A sample of Model V2 runs with 256 epochs and learning rate = 7×10^{-8}.

Run ID	Type	Epoch	Accuracy	Loss	Total Time (hour)
1	Train	256	0.8222	0.3437	2.814
1	Validation	256	0.8329	0.3229	2.814
2	Train	256	0.8198	0.3511	2.817
2	Validation	256	0.8367	0.3252	2.817
3	Train	256	0.8224	0.3439	2.812
3	Validation	256	0.8368	0.3213	2.812
4	Train	140	0.8077	0.4064	1.53
4	Validation	140	0.8221	0.3875	1.53
5	Train	176	0.8133	0.387	2.044
5	Validation	176	0.8328	0.3652	2.044
6	Train	128	0.7982	0.4213	1.398
6	Validation	128	0.8176	0.3918	1.398
7	Train	128	0.8114	0.4005	1.4
7	Validation	128	0.8274	0.3731	1.4
AVERAGE			**0.82152**	**0.3672**	

Table 4.3. A sample of Model V3 runs with learning rate = 1×10^{-5}.

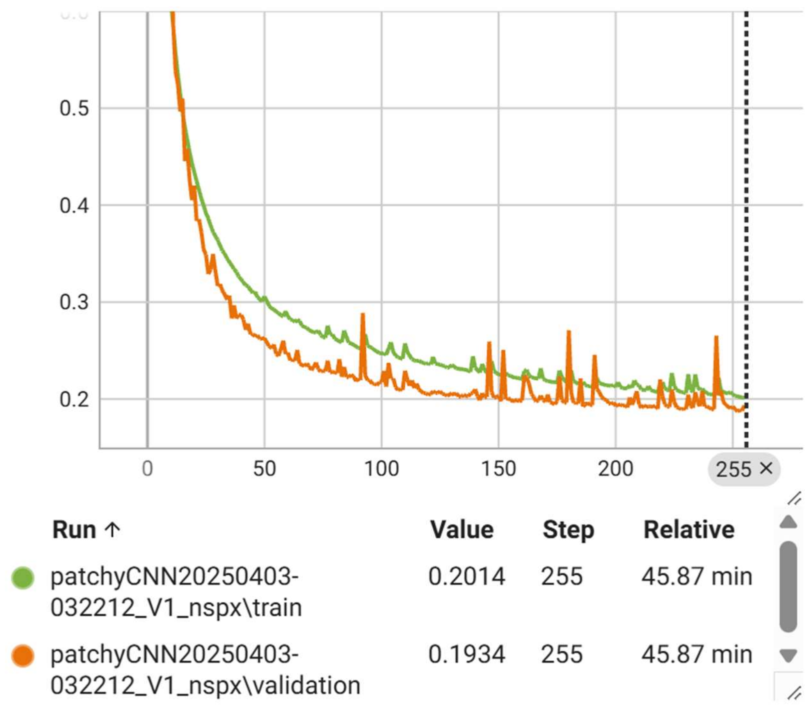
4.3. Performance of model V1

The graphs below illustrate the performance in a particular sample of model V1.



Run ↑	Value	Step	Relative
● patchyCNN20250403-032212_V1_nspx\train	0.8895	255	45.87 min
● patchyCNN20250403-032212_V1_nspx\validation	0.8756	255	45.87 min

Figure 4.2. The accuracy of model V1 in training and validation per epoch (Run ID = 4).

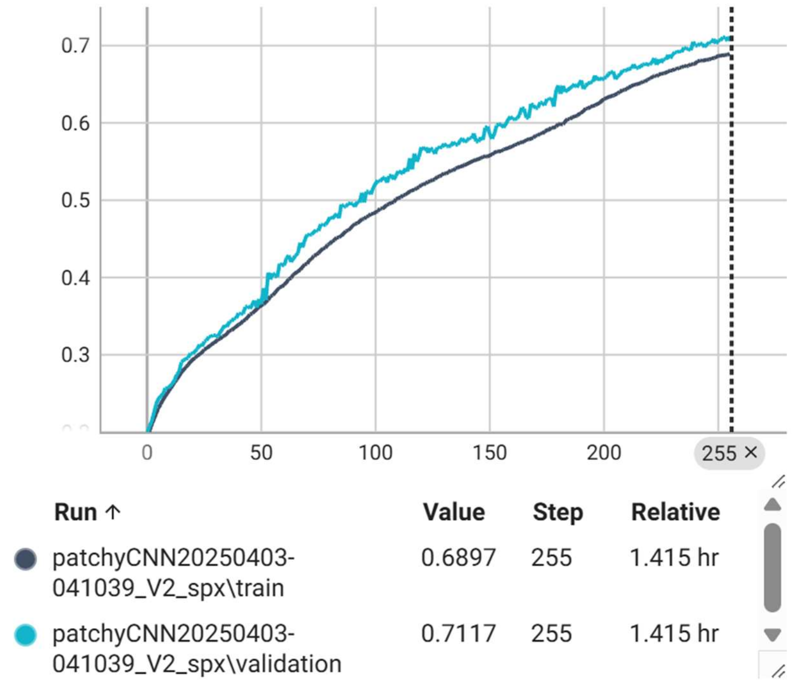


Run ↑	Value	Step	Relative
● patchyCNN20250403-032212_V1_nspx\train	0.2014	255	45.87 min
● patchyCNN20250403-032212_V1_nspx\validation	0.1934	255	45.87 min

Figure 4.3. The loss of model V1 in training and validation per epoch (Run ID = 4).

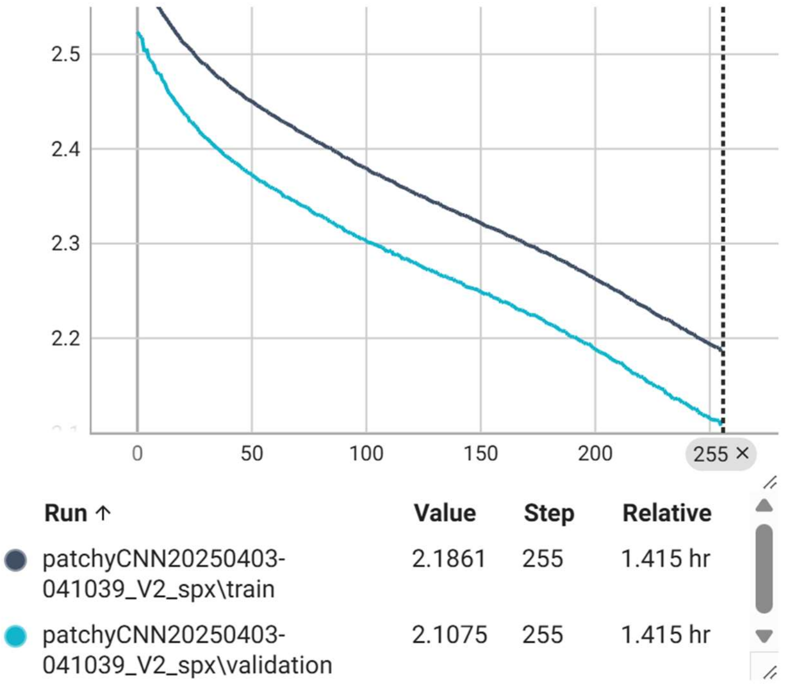
4.4. Performance of model V2

The graphs below illustrate the performance of model V2.



Run ↑	Value	Step	Relative
● patchyCNN20250403-041039_V2_spx\train	0.6897	255	1.415 hr
● patchyCNN20250403-041039_V2_spx\validation	0.7117	255	1.415 hr

Figure 4.6. The accuracy of model V2 in training and validation per epoch (Run ID = 4).

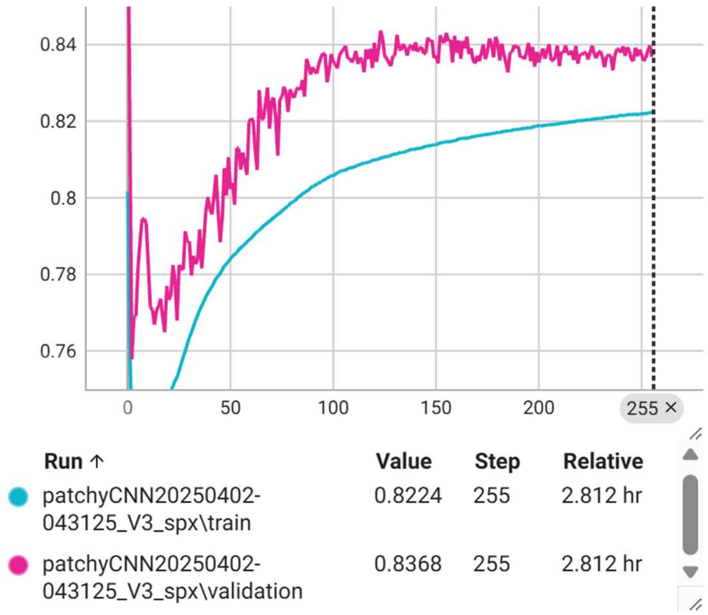


Run ↑	Value	Step	Relative
● patchyCNN20250403-041039_V2_spx\train	2.1861	255	1.415 hr
● patchyCNN20250403-041039_V2_spx\validation	2.1075	255	1.415 hr

Figure 4.7. The loss of model V2 in training and validation per epoch (Run ID = 4).

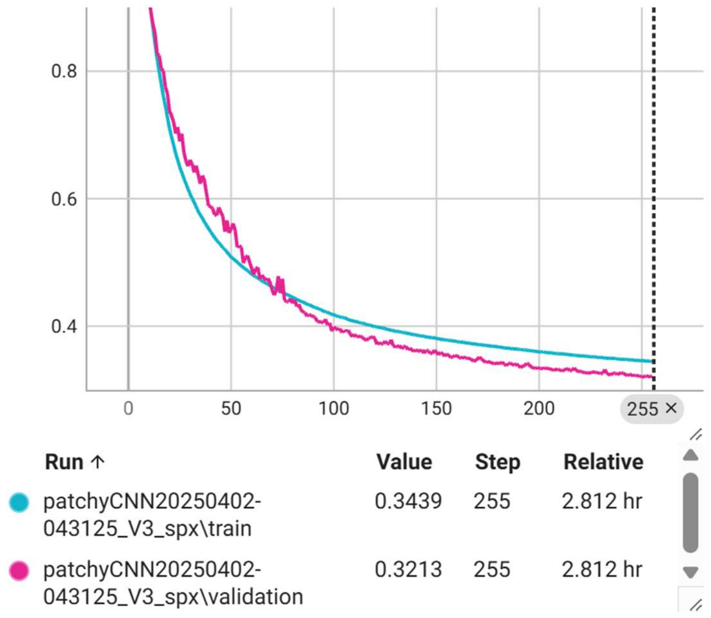
4.5. Performance of model V3

The graphs below illustrate the performance of model V3.



Run ↑	Value	Step	Relative
● patchyCNN20250402-043125_V3_spx\train	0.8224	255	2.812 hr
● patchyCNN20250402-043125_V3_spx\validation	0.8368	255	2.812 hr

Figure 4.10. The accuracy of model V3 in training and validation per epoch (Run ID = 3).



Run ↑	Value	Step	Relative
● patchyCNN20250402-043125_V3_spx\train	0.3439	255	2.812 hr
● patchyCNN20250402-043125_V3_spx\validation	0.3213	255	2.812 hr

Figure 4.11. The loss of model V2 in training and validation per epoch (Run ID = 3).

4.6. Training and Validating Performance Confusion Matrices

The confusion matrices of the models are put in the order of Model V1 will be on the first row, then Model V2 and Model V3 for the training and validating process. The illustrations describe the result of the predictions in three model runs V1,V2, and V3, with run ID = 4,3,3 respectively. The horizontal axis showed the predicted class, while the vertical showed the actual class. A true positive is determined by the count of a prediction where the predicted class is similar to the actual class. Therefore, the accuracy is calculated by the number of True Positive occurrences divided by the total number of occurrences.

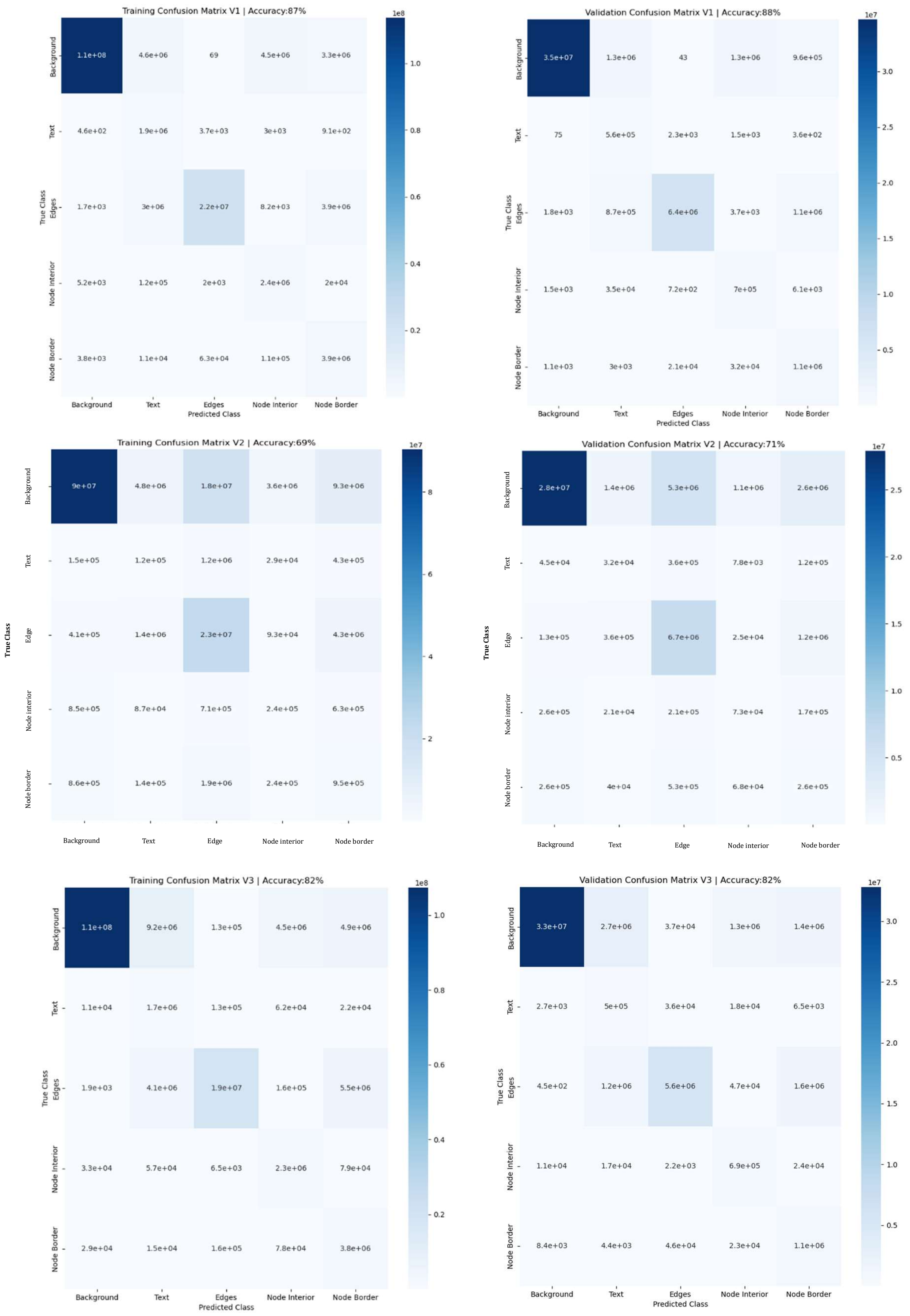


Figure 4.12. The confusion matrices of model V1, V2, V3, respectively.

4.7. Testing performance confusion matrix

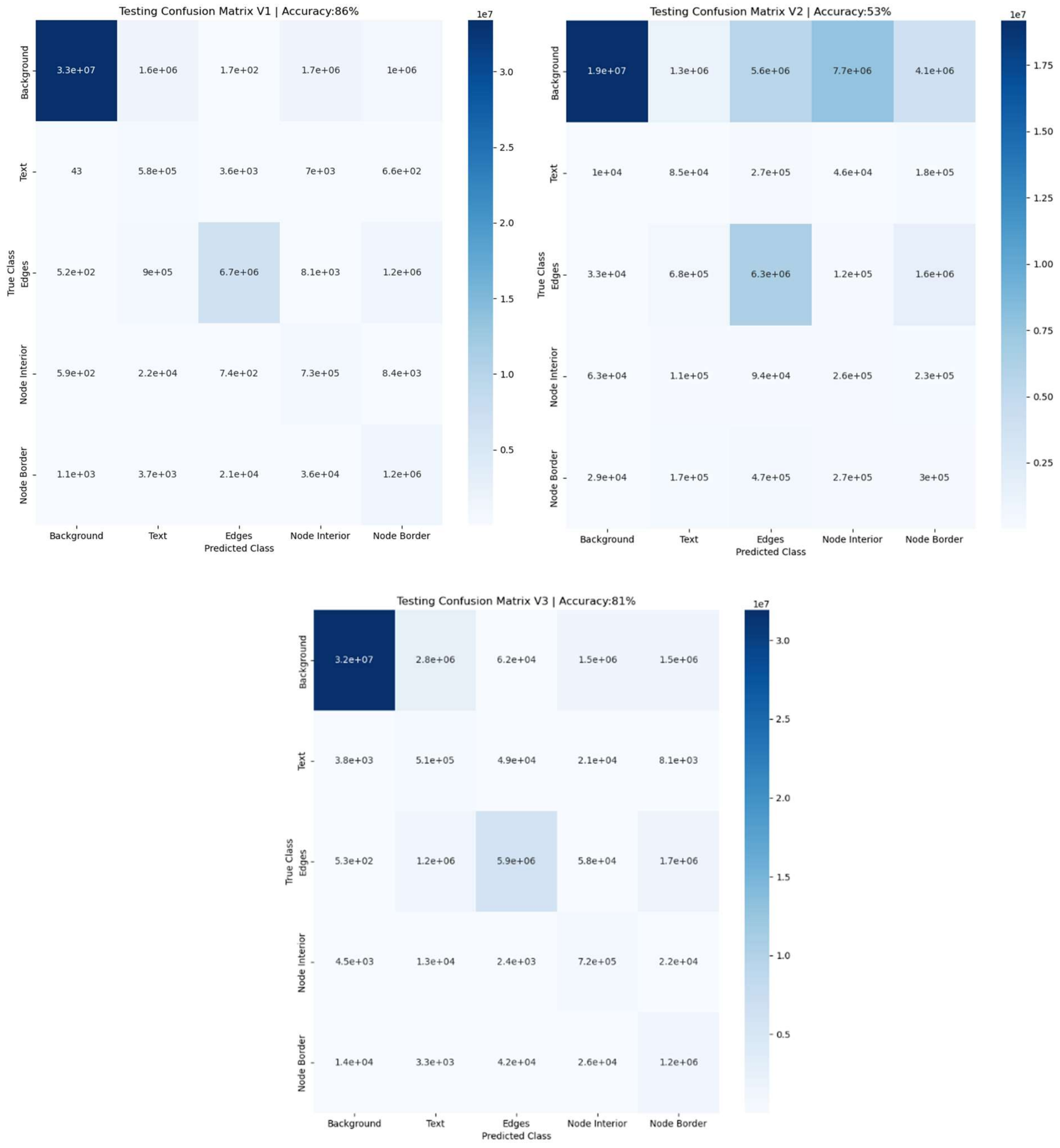


Figure 4.12. The confusion matrices of model V1, V2, V3 for testing dataset, respectively.

5. Discussion

In this section, we will discuss the observations, and the outcome derived after the experiment. As could be seen from the training graphs, all models perform as the expectation of a working model, where all accuracies have a tendency of increasing and the loss of each model witnessed the significant decrease over 256 epochs. Regarding to the accuracy of the models, it had experienced a positive trend where all the models illustrated the average above 60%.

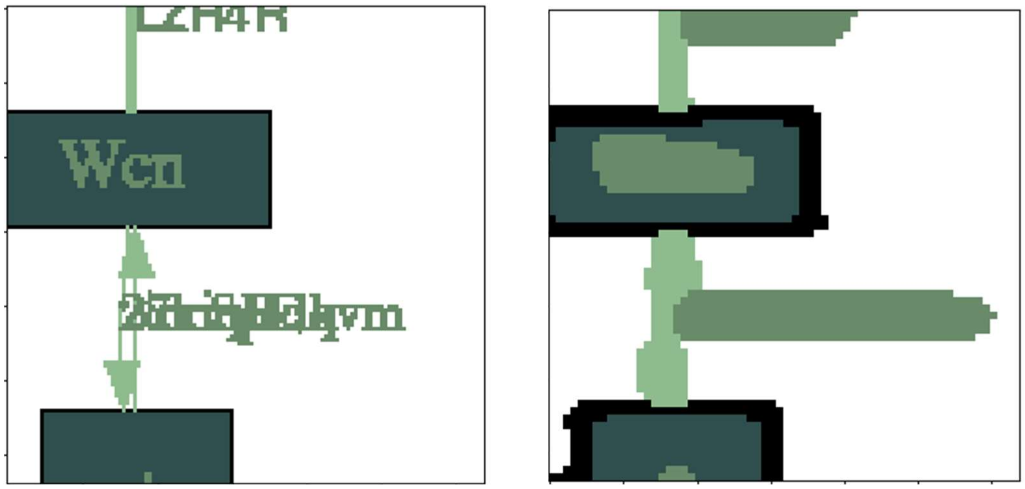


Figure 5.1. Another sample of Ground Truth versus Model V1's prediction

Consider model V1, to our surprise, it showed the best performance among the three. From all of our runs since the model was completed, the accuracy trend of model V1 always meet our expectation and got 88% on average. The loss of the model also experienced the considerable drop and hit the average of 4.0. From the visualization, we can see that the model had very precise predictions, and it can identify the patterns and area of each class. However, the areas of predictions are a little bit too wide, and we can also see this phenomenon through the confusion matrix, where it classified the background as other classes quite frequently. This may not be much of a problem for our application, though; including a bit of background is probably better than cutting off some of the content. Furthermore, by looking at the line of accuracy and loss per epoch, it is possible to see that the model had a little spiky trend where it sometimes immediately decreased its accuracy and increased the loss. However, it had shown its high performance due to the ability of maintaining a high accuracy, low loss, quick running time among all three models.

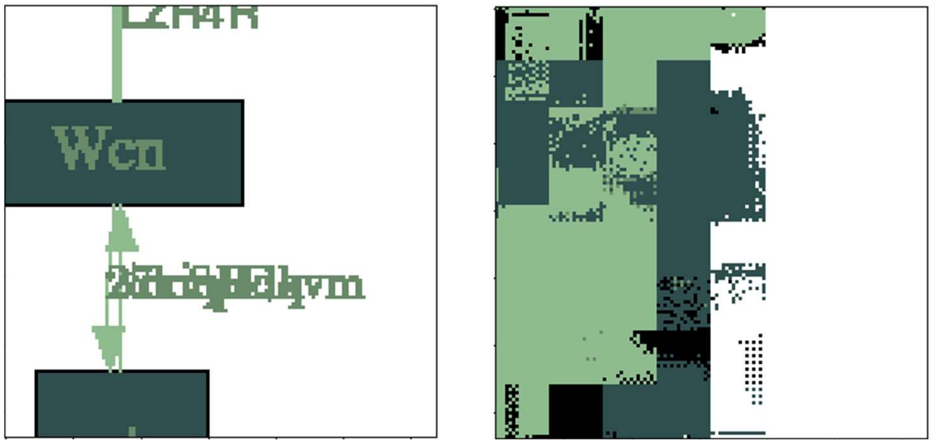


Figure 5.2. Another sample of Ground Truth versus Model V2's prediction

By looking at the visualization of model V2, we can see that this model did not return a very positive result. The classification failed, and there is no pattern seems strongly interpret through the visualization. From our runs, the accuracy rate slowly increased though epochs up to 60% and the loss also gradually dropped as well, but it performed not as well as Model 1. From the confusion matrix in Figure 4.8. and Figure 4.9., the model had shown it is confusing with the classes because of the high proportion of prediction for False Positive and False Negative.

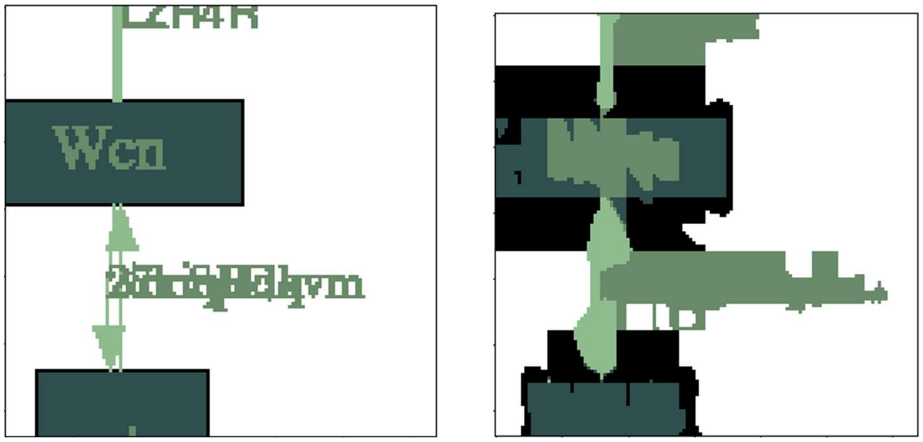


Figure 5.3. Another sample of Ground Truth versus Model V3's prediction

As mentioned in Section 3, Model V3 is an upgrade of Model V2, and it had demonstrated its improvement in comparison to its previous version. Its accuracy increased up to 80-84% on average, the loss trend was substantially declined. As shown in the visualization, the model did learn the patterns of the classes and successfully identified the area of them. Despite the upward trend in accuracy, it still experienced a little proportion of blockiness in the segmentation due to the impact of superpixels. Furthermore, the area of classes in Model V3 also illustrated it was slightly wider than the ground truth, and the run-time of Model V3 would be the slowest one. However, model V3 is a successful enhance version of model V2, in which it demonstrated the improvement in predicting the semantic segmentation.

Overall, we have seen that Model V1 and Model V3 had demonstrated their potential in providing the final result of semantic segmentation. Furthermore, when dealing with the new testing data, model V1 and V3 also have a great result, 86% and 81%, respectively. There are two main takeaways that we concluded after the experiment:

- The models predict a wider area for the class. In other words, for the background pixels which stay closely to another class, instead of correctly predicting the background class, the models predict it as the other class.
- The fact that Model V1 outweighs both Model V2 and Model V3 in terms of accuracy, loss, and run-time showed that Superpixel segmentation might not be a suitable approach to this project.

There are some considerations we proposed for the underperformance of superpixel in this project. Originally, the idea of using superpixel is based on the considerations of using another segmentation algorithm, such as the Normalized Cut. Implementing a Normalized Cut graph-based algorithm will be very complicated for each pixel, but it will be possible to dealing with superpixels. Therefore, we doubt that superpixels might can be used better with another approach, instead of feeding them to a CNN model. In addition, our idea of superpixels generation can be improved, and we can see that by changing the superpixel segmentation layers in Model V2 into Model V3, the accuracy had been improved. As a result, we believe that by using a more suitable superpixel generation might lead to a better result. Lastly, may be from the beginning, superpixels are used to recognise the similarity of nearby pixels of a very massive input image. In the TADA project, there are some factors that might not be fit with superpixels, such as a very thin edge, the border of the nodes, where superpixels can not interpret and learn fully its pattern.

6. Conclusion and future work

In conclusion, this project had covered the related works to semantic segmentations model, the architecture of built models and their potential when handling with real data inputs. As mentioned, the most appropriate approach up to this point is using a linear CNN model, which had illustrated the best performance in accuracy, loss, and time. Superpixel is still an interesting topic to thinking about, especially it had shown an increasing trend in the performance. However, this project still provides a good result for semantic segmentation, which can be used to continue the pipeline of the TADA project. The next step of the project is instance segmentation, and by predicting the class of the pixel correctly as the semantic segmentation, this can provide good inputs to pass into the pipeline.

Furthermore, there are a lot of improvements can be made in the future. First and foremost, the custom layers were made in a naïve way and were not optimized in time and memory complexity, and this is a considerable point to enhance. Secondly, we can test our current models with larger data input and higher number of learning epochs. With larger data input, we believe that it can avoid the problem of overfitting and gain higher accuracy thanks to the longer run. Moreover, new models can be created based on the observations and the takeaways stated above. By having more models, we can see if superpixels segmentation is a good factor to include if it is used appropriately, or it is not belonging to this project. Last but not least, we want to see how the models perform with different data sets, where the data can be augmented, added noises, etc.

7. Reference

Aitken, K., Ramasesh, V. V., Cao, Y., & Maheswaranathan, N. (2021). Understanding how encoder-decoder architectures attend. *Proceedings of the 35th International Conference on Neural Information Processing Systems*.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate* (No. arXiv:1409.0473). arXiv. https://doi.org/10.48550/arXiv.1409.0473

Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., Jaitly, N., Li, B., Chorowski, J., & Bacchiani, M. (2018). State-of-the-Art Speech Recognition with Sequence-to-Sequence Models. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4774–4778. https://doi.org/10.1109/ICASSP.2018.8462105

Desolneux, A., Moisan, L., & Morel, J.-M. (2008). *From Gestalt theory to image analysis: A probabilistic approach*. Springer.

Georgescu, M.-I., Ionescu, R. T., & Miron, A.-I. (2022). *Diversity-Promoting Ensemble for Medical Image Segmentation* (No. arXiv:2210.12388). arXiv. https://doi.org/10.48550/arXiv.2210.12388

Hanbury, A. (2008). How Do Superpixels Affect Image Segmentation? In J. Ruiz-Shulcloper & W. G. Kropatsch (Eds.), *Progress in Pattern Recognition, Image Analysis and*

Applications (Vol. 5197, pp. 178–186). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-85920-8_22

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

Islam, M., Chen, G., & Jin, S. (2019). *An Overview of Neural Network*. *5*, 05. https://doi.org/10.11648/j.ajnna.20190501.12

Kamalakannan, S., Gururajan, A., Sari-Sarraf, H., Long, R., & Antani, S. (2010). Double-Edge Detection of Radiographic Lumbar Vertebrae Images Using Pressurized Open DGVF Snakes. *IEEE Transactions on Biomedical Engineering*, *57*(6), 1325–1334. https://doi.org/10.1109/TBME.2010.2040082

Karimpouli, S., & Tahmasebi, P. (2019). Segmentation of digital rock images using deep convolutional autoencoder networks. *Computers & Geosciences*, *126*, 142–150. https://doi.org/10.1016/j.cageo.2019.02.003

Khosrowpour, M. (2009). *Encyclopedia of information science and technology* (2nd ed). Information science reference.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Luong, M.-T., Pham, H., & Manning, C. D. (2015). *Effective Approaches to Attention-based Neural Machine Translation* (No. arXiv:1508.04025). arXiv. https://doi.org/10.48550/arXiv.1508.04025

Malik, J., Belongie, S., Leung, T., & Shi, J. (2001). Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, *43*(1), 7–27. https://doi.org/10.1023/A:1011174803800

Neubert, P., & Chemnitz, P. P. (2012). *Superpixel Benchmark and Comparison*. https://api.semanticscholar.org/CorpusID:4190710

Neubert, P., & Protzel, P. (2018). *Superpixel Benchmark and Comparison*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Ren & Malik. (2003). Learning a classification model for segmentation. *Proceedings Ninth IEEE International Conference on Computer Vision*, 10–17 vol.1. https://doi.org/10.1109/ICCV.2003.1238308

Ren, X., & Malik, J. (2002). A Probabilistic Multi-scale Model for Contour Completion Based on Image Statistics. In A. Heyden, G. Sparr, M. Nielsen, & P. Johansen (Eds.), *Computer Vision—ECCV 2002* (Vol. 2350, pp. 312–327). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-47969-4_21

Srivastava, N., Hinton, G., 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.

Tu, W.-C., Liu, M.-Y., Jampani, V., Sun, D., Chien, S.-Y., Yang, M.-H., & Kautz, J. (2018). Learning Superpixels with Segmentation-Aware Affinity Loss. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 568–576. https://doi.org/10.1109/CVPR.2018.00066

Wagemans, J., Elder, J. H., Kubovy, M., Palmer, S. E., Peterson, M. A., Singh, M., & von der Heydt, R. (2012). A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization. *Psychological Bulletin*, *138*(6), 1172–1217. https://doi.org/10.1037/a0029333

Wang, M., Liu, X., Gao, Y., Ma, X., & Soomro, N. Q. (2017). Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, *56*, 28–39. https://doi.org/10.1016/j.image.2017.04.007

Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung*, *4*(1), 301–350. https://doi.org/10.1007/BF00410640

Wu, W., Chen, A. Y. C., Zhao, L., & Corso, J. J. (2014). Brain tumor detection and segmentation in a CRF (conditional random fields) framework with pixel-pairwise affinity and superpixel-level features. *International Journal of Computer Assisted Radiology and Surgery*, *9*(2), 241–253. https://doi.org/10.1007/s11548-013-0922-7

Yang, F., Sun, Q., Jin, H., & Zhou, Z. (2020). Superpixel Segmentation With Fully Convolutional Networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13961–13970. https://doi.org/10.1109/CVPR42600.2020.01398

Zhao, Y., Nacenta, M. A., Sukhai, M. A., & Somanath, S. (2024). TADA: Making Node-link Diagrams Accessible to Blind and Low-Vision People. *Proceedings of the CHI*

Conference on Human Factors in Computing Systems, 1–20. https://doi.org/10.1145/3613904.3642222

Zhengqin Li & Jiansheng Chen. (2015). Superpixel segmentation using Linear Spectral Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1356–1363. https://doi.org/10.1109/CVPR.2015.7298741