Single-Class Instance Segmentation for Vectorization of Line Drawings

Rhythm Vohra[a], Amanda Dash[b] and Alexandra Branzan Albu[c]
University of Victoria, Canada

Keywords: Segmentation, Visual Attention, Image Vectorization.

Abstract: Images can be represented and stored either in raster or in vector formats. Raster images are most ubiquitous and are defined as matrices of pixel intensities/colours, while vector images consist of a finite set of geometric primitives, such as lines, curves, and polygons. Since geometric shapes are expressed via mathematical equations and defined by a limited number of control points, they can be manipulated in a much easier way than by directly working with pixels; hence, the vector format is much preferred to raster for image editing and understanding purposes. The conversion of a raster image into its vector correspondent is a non-trivial process, called image vectorization. This paper presents a vectorization method for line drawings, which is much faster and more accurate than the state-of-the-art. We propose a novel segmentation method that processes the input raster image by labeling each pixel as belonging to a particular stroke instance. Our contributions consist of a segmentation model (called Multi-Focus Attention UNet), as well as a loss function that handles well infrequent labels and yields outputs which capture accurately the human drawing style.

1 INTRODUCTION

Images are rich sources of information which can be stored in either raster or vector formats. Cameras use sensors that capture the real-world scene in a grid of photosensitive elements (with each of these elements corresponding to pixels in the image), and digital screens display images as an array of pixels. As a result, most of the images available to us are stored in raster format. Raster images have applications in various contexts, including digital photography, print, media, web designs, screens, and many others. Since raster images consist of a fixed number of pixels, when zoomed-in or modified, the quality of the raster image gets degraded, as shown in Figure 1.

A high-quality raster image may consist of thousands to millions of pixels; thus raster images are difficult to edit at a pixel level. Additionally, these images require ample storage space. Instead, images can be compressed using various algorithms which reduce the quality of raw raster images. A trade-off exists between a high compression ratio and a high-quality image. Vector images are a more elegant solution to this trade-off, as they use less storage space while maintaining high image quality.

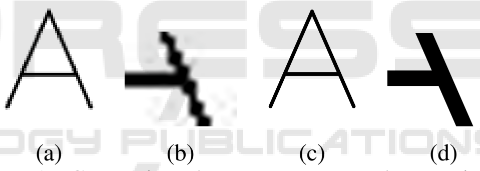


Figure 1: Comparison between vector and raster images when zoomed in; vector images maintain clean lines, while raster images get blurred because of constraints in resolution. (a) Raster image, (b) Zoomed-in raster image, (c) Vector image, (d) Zoomed-in vector image.

Vector images are described by geometric primitives such as curves and their control points. Bezier curves are used most frequently; they are based on Bernstein polynomials (de Casteljau, 1963), constructing smooth and continuous shapes. Vector images have phenomenal flexibility because they can be stretched, rotated, or scaled while not compromising their clarity, just by changing the position of control points. Editing these mathematically described shapes is much easier than editing pixels in a raster image. The edited images maintain their sharpness across a wide range of resolutions, a highly desirable attribute (see Figure 1).

Graphical designers perform many drawings manually, using a pen or stylus. As a result, there is a significant need for converting hand-drawn raster images into vector formats. Commercial vectorization tools such as InkSpace, CorelDRAW, and Ado-

[a] https://orcid.org/0009-0004-0071-0720
[b] https://orcid.org/0000-0001-8654-1593
[c] https://orcid.org/0000-0001-8991-0999

Vohra, R., Dash, A. and Branzan Albu, A.
Single-Class Instance Segmentation for Vectorization of Line Drawings.
DOI: 10.5220/0012465900003660
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 3: VISAPP*, pages 215-226
ISBN: 978-989-758-679-8; ISSN: 2184-4321
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

beTrace use manual tracing of the vector lines from the raster drawings by a skilled user. However, this manual tracing process is labour-intensive and time-consuming.

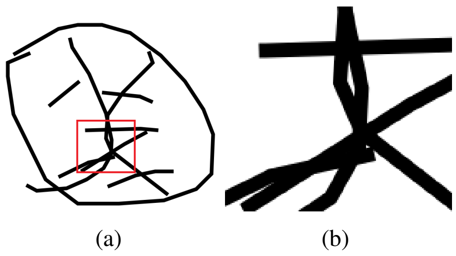


Figure 2: (a) Example of an input image from our dataset with red highlighted region zoomed in. (b) The zoomed-in version of the input image depicts the overlapping regions, emphasizing the importance of considering the global context of the overall image to accurately identify which label belongs to each stroke.

Our approach performs Single-Class Instance Segmentation (SCIS), i.e., classification of each stroke individually, to convert hand-drawn strokes (i.e. instances) from binary (one-class) raster images into vector-based stroke representations. We propose a segmentation method that processes the input raster image by labeling each pixel as belonging to a particular stroke instance. The labelled regions in the output of our segmentation model are separately vectorized using a commercial tool called Potrace (Selinger, 2003). The vectorized forms of all strokes are then combined to form the final vector image (see Figure 3).

Our contributions are two-fold:

1. We propose a novel segmentation model named Multi-Focus Attention UNet (MFAU) that outperforms a state-of-the-art method (Kim et al., 2018), on several datasets using various performance metrics.
2. We propose a novel loss function, the Margin-Regularised Loss Function (MRLF) , that generalizes well on less frequent labels in a highly imbalanced dataset. Our loss function also enables us to generate outputs that are consistent with users' drawing styles and with the perceptual grouping principles of similarity, continuity and closure.

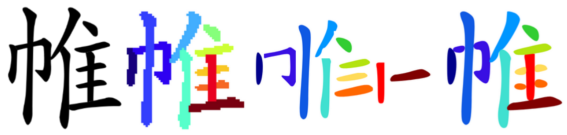


Figure 3: Vectorization by segmenting each stroke instance of a line drawing. *Left to Right:* Grayscale raster image; segmented output of raster image; instances separated and vectorized, relative position is intentionally modified for visualization purposes; final vectorized result.

The rest of this paper is organized as follows. Section 2 presents a brief literature review on vectorization of line drawings. Section 3 presents the proposed segmentation model and loss function. Section 4 discusses experimental results and compares the proposed model with the state of the art. Section 5 draws conclusions and outlines future work directions.

2 RELATED WORK

This section discusses first vectorization methods that focus on detecting geometric primitives with traditional computer vision and image processing techniques; next, we focus on recent deep-learning based approaches.

Early works (Kultanen, 1990; Jimenez and Navalon, 1982; Roosli and Monagan, 1996; Tombre et al., 2000; Hilaire and Tombre, 2006) combined a variety of image processing operators and techniques such as thinning, thresholding, layering, contour finding, edge detection, curve fitting, feature point extraction, Hough transform, and polygonal approximation.

(Dori, 1997) proposed an Orthogonal Zig-Zag (OZZ) algorithm for vectorizing engineering drawings that focuses on line recognition with improved performance and with significantly lower computational complexity with respect to the Hough Transform. This algorithm detects lines in binary images by using the same principle as the propagation of a ray of light through optical fiber. The one-pixel wide ray traverses the black pixels, treating them as a conducting path. Whenever the ray encounters a white pixel, its direction changes by 90°. The ray therefore collects information about the presence of lines, their width, their start and end points, and enables skipping junctions.

(Bartolo et al., 2007) proposed a technique to vectorize scribbled drawings for computer-aided design (CAD) interpretation which is compatible with the natural drawing habits of the user. They used Gabor filters to simplify the scribbles, and then extracted vector lines with a Kalman filter.

(Favreau et al., 2016) proposed the first algorithm that balances fidelity to the raster input along with simplicity of the output to generate an accurate and compact number of curves via a global optimization approach. Their algorithm's robustness is shown on a variety of drawings and human-made sketches.

Some vectorization techniques have matured enough to be integrated in commercial tools. Potrace Inkscape (Selinger, 2003), Adobe Illustrator Image Tracing (Wood, 2012), and CorelDraw (Bouton, 2014), focus on maintaining high visual accuracy and quality during the vectorization process.

(Najgebauer and Scherer, 2019) introduced a method that performs fast vectorization of line drawings based on multi-scale second derivative detector and inertia-based line tracing to improve accuracy at junctions.

(Zhang et al., 2022) proposed a deep-learning based semantic segmentation method for line vectorization of engineering drawings. They used a combination of feature extraction, a non-local segmentation algorithm based on dilated convolution and self-attention module, and local rasterization. They claimed robustness to typical line drawing problems such as blurring and line breakage.

(Inoue and Yamasaki, 2019) introduced an instance segmentation technique that is data-driven and utilized deep networks to segment strokes based on the global image context. They used a model based on MaskRCNN (He et al., 2017) to perform instance segmentation of hand drawings in a more efficient way. They proposed two modifications to the MaskRCNN architecture: upsampling the masks generated from the mask branch in order to detect finer details, and a post-processing for the correction of mismatched pixels.

(Kim et al., 2018) elegantly formulated the problem of vectorization as computing an instance segmentation of the most likely set of paths that could have created the input raster image. They trained a pair of networks to help solve ambiguities in regions where multiple path intersect and hence overlap. Both networks take into account the global context of the image, hence yielding a semantic vectorization.

We consider (Kim et al., 2018) as a state-of-the-art approach in vectorization viewed as an instance segmentation problem. Our approach proposes a different way of solving ambiguities generated by multi-path overlap, based on single-class instance segmentation and perceptual grouping principles (similarity, continuity, and closure). Our method is significantly faster than (Kim et al., 2018) also compares favourably in terms of accuracy. The following section describes our proposed approach.

3 PROPOSED APPROACH

Our proposed architecture, Multi-Focus Attention UNet (MFAU) (see Figure 4), is inspired from Attention UNet (Oktay et al., 2022) and performs single class instance segmentation (SCIS). Our changes to Attention UNet consist in the Multi-Focus Attention Gate (MFAG) (Sec. 3.1), and the Highway Skip-Connections (Sec. 3.2). Our loss function is described in (Sec. 3.3).

3.1 Multi-Focus Attention Gate (MFAG)

Figure 5 provides the block diagram of *MFAG* (the MFAG block represents the modifications we made to the AG (Oktay et al., 2022). One may note that, until the sigmoid function, both *MFAG* and *AG* are performing the exact same task. In the final step of *AG*, the attention coefficients $\alpha \in [0,1]$ are calculated to highlight each spatial region of the input feature vector from the encoder (by multiplying α by the input feature vector x). Figure 2 shows the zoomed in version of a sample input image containing overlapping regions; one may note how difficult it is to attribute labels to strokes. The global image context is necessary for a correct labelling. Our modifications enable the network to encode this contextual information via the gating feature vector, which is collected at a coarser scale of the decoding layer. This is achieved by utilizing attention coefficients, which involve multiplying their complement (1 - α) with the gating feature vector **H**. Thus, *MFAG* controls the amounts of higher-level contextual information (from the gating signal) and lower-level information (from the input feature vector) that need to be combined to further enhance relevant semantic details of the input image and discard the irrelevant ones.

Let the inputs to our proposed *MFAG* architecture (as shown in Figure 5a) at layer l from the decoding layer be represented as g^l, and the input from the encoding layer be represented as x^l. The gating feature vector g^l is used to ascertain the regions of focus and incorporate higher-level contextual information. To perform sequential linear transformations and extract the local and global context of the image, a convolution followed by batch normalization is applied to the gating and local feature vectors at each layer. The resultant transformed feature vectors are represented as $D(g^l, W_g^l)$ and $X(x^l, W_x^l)$, and parameterized by W_g^l and W_x^l for gating and local features, respectively. For simplicity, the transformed feature vectors $D(g^l, W_g^l)$ and $X(x^l, W_x^l)$ will be used in the equations as $\mathbf{D}^l$ and $\mathbf{X}^l$. These transformed feature vectors are then summed, given as:

$$AG_{sum}{}^l = \mathbf{D}^l + \mathbf{X}^l \tag{1}$$

Following this summation, a non-linear activation function σ_1 is applied to the resultant output of Eqn. 1. (Oktay et al., 2022) used the Rectified Liner Unit (ReLU) as a non-linear activation, thus we also use

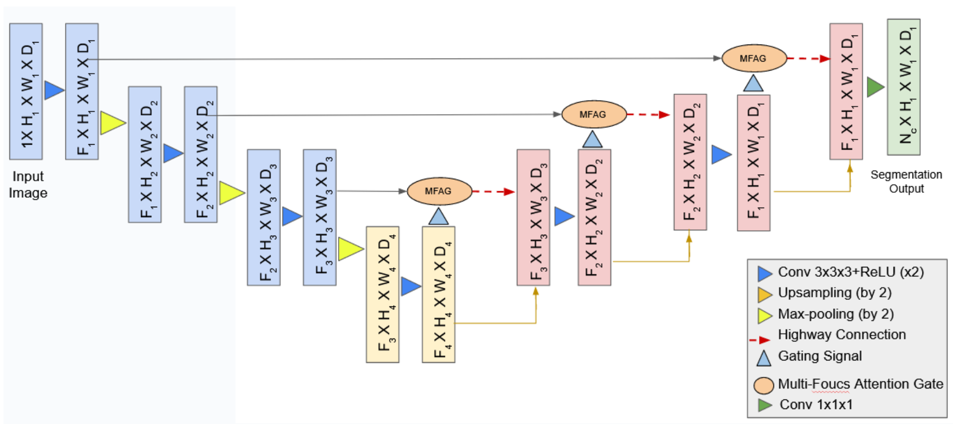
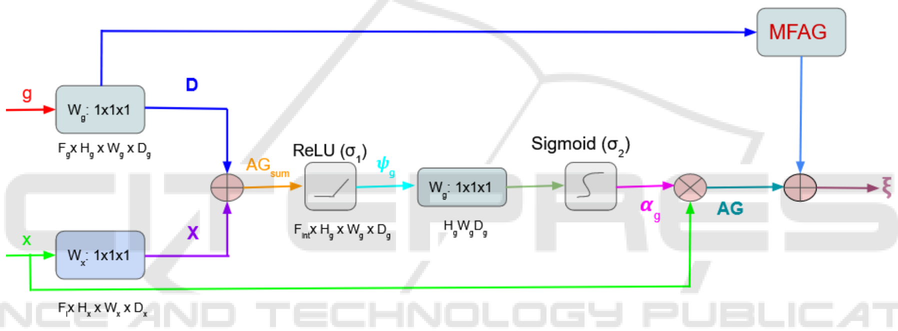
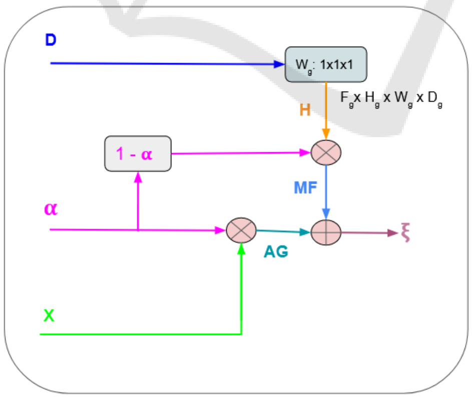


Figure 4: Block diagram of our proposed Multi-Focus Attention UNet (MFAU) model. The input image is downsampled in the encoder path (blue) and upsampled in the decoder path (red). The bottleneck (yellow) is a bridge between the encoder and decoder. Our model is similar to Attention UNet (Oktay et al., 2022); the differences lie in the features from the encoder and decoder being passed through the MFAG (orange ellipses) and further through the *highway skip-connections* (red dashed arrows). The final layer is shown in green. The $FxHxWxD$ represents the size of the feature maps where F is the number of channels, and $HxWxD$ are the height, width, and depth of the image. N_c in the final layer represents the number of instance labels.



(a)



(b)

Figure 5: a) Block Diagram of the Multi-Focus Attention Gate. b) MFAG block. The attention coefficient α, is computed using the input feature (x) from the encoder and the gating signal (g) from the decoder. AG is the output of the attention gate. The W_g : $1x1x1$ are $1x1x1$ convolutions.

it for consistency. The following equation shows the non-linear activation being applied:

$$\psi^l = \sigma_1(AG_{sum}{}^l) = max(0, AG_{sum}{}^l) \qquad (2)$$

where ψ^l represents the intermediate coefficients. A linear transformation is applied to ψ using 1x1x1 convolutions (not using spatial dimensions), utilized to downsample the input feature maps to the gating feature vector dimensions. A non-linear activation function, i.e., sigmoid (represented as σ_2), is applied to the linearly transformed results. We use a sigmoid activation to yield sparser activations instead of a softmax function that normalizes the attention coefficients, thus resulting in better training convergence in Attention Gates (Oktay et al., 2022).

$$\alpha^l = \sigma_2(\psi^l) = \frac{1}{1+e^{-\psi^l}} \tag{3}$$

where α represents the attention coefficients $\in [0,1]$, which highlight feature responses from the salient regions and suppress the ones with lesser semantic values. In AG, a grid resampler is applied to the attention coefficients that are implemented using trilinear interpolation to make the dimensions of the feature maps similar to the dimensions of the input feature vector to multiply them element-wise. In our case, the dimensions of the attention coefficient feature maps are the same as the input feature vector x. Therefore, for the purpose of simplicity, we omit this step. The element-wise product of attention coefficients with the input feature vector from the encoding stage (i.e., used to scale each spatial location of the input feature vector based on the attention scores $\alpha \in [0,1]$), is given as:

$$AG^l = \alpha^l.x^l \tag{4}$$

To perform the multi-focus attention mechanism (see Figure 5b), the transformed gating feature vector, represented as $\mathbf{D}^l$, is sequentially transformed again. This transformation serves the purpose of further enhancing the gating features at a coarser scale through the utilization of 1x1x1 convolution operations, denoted as $H(g^l, W_h^l)$ and parameterized by W_h^l for multi-focus feature vectors ($H(g^l, W_h^l)$ is represented as $\mathbf{H}^l$ for simplicity). This transformed gating feature vector is element-wise multiplied with the complement of attention coefficients to further improve the focusing ability of the network, select the most relevant features, and discard the less important ones.

$$MF^l = (1-\alpha^l).\mathbf{H}^l \tag{5}$$

The final output of the MFAG is given by concatenating Eqn. 4 with Eqn. 5:

$$\xi^l = AG^l + MF^l \tag{6}$$

$$\xi^l = \alpha^l.x^l + (1-\alpha^l).\mathbf{H}^l \tag{7}$$

where ξ^l represents the selection feature vectors at layer l. The attention coefficient $\alpha \in [0,1]$ acts as a weighting mechanism that controls the amount of focus to be given to the input feature map from the encoding stage or to the gating feature map from a coarser decoding layer. If α is close to 0, the output of the *MFAG* will focus more on the spatial information, whereas if α is close to 1, the output will focus more on the contextual information from the input image.

Based on the Eqn. 6, a special case of saturation of the attention coefficients $\alpha \in [0,1]$ is given below:

$$\xi^l = \begin{cases} x, & \text{if } \alpha = 0 \\ \mathbf{H}, & \text{if } \alpha = 1 \end{cases} \tag{8}$$

For simplicity, the symbol for layer l has been omitted. If α is 0, the input feature vectors from the encoding stage are highlighted, whereas if α is 1, the gating feature vectors from the decoder layer collected at a coarser scale undergo a linear transformation that helps to recognize complex patterns of the feature map and highlight these patterns.

3.2 Highway Skip-Connections

There is plenty of evidence related to how the depth of a neural network improves its performance; however, optimizing a deeper network is a challenging task. Highway networks (Srivastava et al., 2015) were explicitly designed to overcome these challenges and train very deep neural networks. They introduce adaptive gating units, a component that learns to control the flow of information in the network. As a result, there can be paths through which the information can traverse directly several layers without being altered. These adaptive gating units (known as highway connections) are used at each layer in the highway networks and were inspired from the gating mechanism used in Long-Short Term Memory (LSTM) networks (Graves and Graves, 2012). The main goal of these gating functions is to learn to select whether the information needs to be passed or transformed through the network. This way, the network is able to dynamically learn which features are relevant to a particular problem and adapt accordingly. To the best of our knowledge, we are the first group to use the concept of highway networks for a small number of layers and integrate it with a UNet-based model as skip-connections.

The output of the MFAG, represented as ξ^l, acts as an input to the highway connection at each decoding layer l. Two sets of operations are applied on the input feature vector ξ^l: a *transformation gate* and a *transformation layer* (the difference between these two operations is the use of different non-linear activation functions). These operations are defined as $G(\xi^l, W_G^l)$ and $T(\xi^l, W_T^l)$, parameterized by W_G^l and W_T^l at every layer l for the transformation gate and transformed input, respectively. The *transformation gate* consists of a 1x1x1 convolution operation and a sigmoid function. This sigmoid function produces a value in the

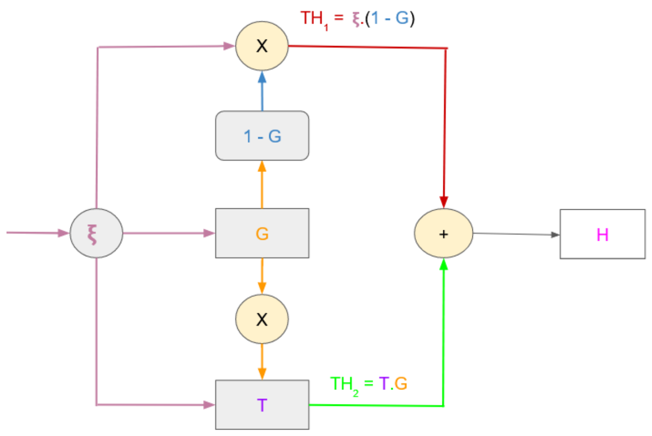


Figure 6: Block diagram of Highway Connections. ξ represents the input from the encoding stage, while H represents the output of the Highway connections, G and T are the transformation gate and transformation layer, respectively. The red and green path shows the two paths from where the information flows.

range 0 and 1, depicting how much of the transformed input should be passed through the networks. If the value is close to 1, the input flows through the layer, whereas if the value is close to 0, most of the inputs get transformed. The *transformation layer* consists of a 1x1x1 convolution operation followed by a ReLU function to apply a non-linear transformation on the input image. The transformation function enables the network to capture complex features from the input image. The two non-linear activation functions are used in a certain way to mitigate the vanishing gradient problem, as even if the transformation gate collapses due to saturation towards 0, the input is still passed through the network. If the saturation of the transformation gate approaches 1, the input gets fully transformed (this special case is shown in Eqn. 13).

Information is passed through two neural paths (the paths are shown in red and green in Figure 6). In the first path (red), the input feature vector ξ^l is element-wise multiplied with the complement of the *transformation gate* feature maps, represented as $1-G(\xi^l, W_G^l)$ (henceforth written as (1 - G)^l for simplicity). This element-wise product for the first information path is given as:

$$TH_1^l = (1-G)^l.\xi^l \tag{9}$$

In the second information path (green), the feature vectors from both the *transformation gate* $G(\xi^l, W_G^l)$ (represented by G) and *transformed input* $T(\xi^l, W_T^l)$ (represented by T) are element-wise multiplied by each other at each layer l.

$$TH_2{}^l = G^l.T^l \tag{10}$$

Highway connections are formed by combining information paths constructed in Eqns. 9 and 10. Therefore, the overall highway connections are represented as:

$$H^l = TH_1{}^l + TH_2{}^l \tag{11}$$

$$H^l = G^l.T^l + (1-G)^l.\xi^l \tag{12}$$

where H symbolizes highway connections output at each decoding layer l. The dimensions of H, G, T and ξ will always be the same.

Based on Eqn. 12, the special case of saturation of the transformation gate can be given by the following two conditions:

$$H = \begin{cases} \xi, & \text{if } G = 0 \\ T, & \text{if } G = 1 \end{cases} \tag{13}$$

For simplicity, the symbol for layer l has been omitted. Therefore, depending on the value of the transformation gate, highway connections control the flow of information by selecting whether to emphasize the transformed input or the original input.

3.3 Compounded Loss Function

We have a specific use case that is inherently imbalanced in a heavy-tailed distribution manner. The output of the segmentation model consists of a static number of channels, corresponding to the maximum number of possible stroke instances (see Section 4.1). As a result, we observe a dense distribution until the average number of strokes followed by a decreasing trend.

The output of the proposed instance segmentation model is a static number of channels, which makes this problem analogous to a multi-label classification problem. However, unlike a multi-label classification problem where each channel represents a specific class prototype, in our use case, we do not have any fixed class prototype, i.e., any instance of the stroke can appear in any channel. Moreover, different artists have different drawing styles, as a result, the same stroke can represent different instances. Figure 7 illustrates different drawing styles for a cat; most of the time a cat is drawn with two ears, a face, and each whisker separately labeled, however, there are cases where the two ears are labeled as a single entity, or the whole face is labeled as one. The skewed distribution in our use case means that infrequent logit channels may suffer from floating point truncation. This issue is mitigated by our margin scaling of the logits. Margin scaling prevents dead neurons which allows for the full power of the attention gates to be used to model the pattern variability. Therefore, we propose a loss function named Margin-Regularized Loss that addresses this variation and identifies the individual instances accurately.

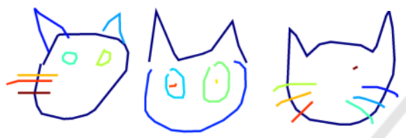


Figure 7: Example of variations of strokes in a cat dataset. *Left*: The common way the cats are labeled. *Middle*: The two ears are labeled as one and the face as another. *Right*: The whole face is labeled as one.

Apart from assigning correct labels to each instance of the stroke, we also need to consider the spatial information of the strokes to incorporate the precise positioning, overlapping, and relationship between these strokes. The spatial imbalances favor instances with larger spatial strokes in the dataset to the detriment of smaller ones. To address spatial imbalance issues, Dice Loss is used to assess how well the ground truth and predicted output overlap with each other. It gives equal weights to pixel-wise true positives and false negatives, encouraging the model to capture the spatial region correctly.

The total loss leverages the strength of both loss components and is expressed as:

$$L = \beta * DL(y_T, y_p) + (1-\beta) * S((y_T, y_p)) \tag{14}$$

where β is a hyperparameter, $DL(y_T, y_p)$ represents the Dice Loss (Sorensen, 1948; Sudre et al., 2017) and $S((y_T, y_p))$ represents the Margin-Regularised loss (see Sec. 3.3.1). In our case, the value of β is 0.5 to equally leverage the benefits of both the loss functions.

3.3.1 Margin-Regularized Loss Function

Inspired by Label-Distribution-Aware Margin (LDAM) (Cao et al., 2019), we developed a loss function that calculates the loss in three steps: (1) calculating the margins, (2) margin scaling, and (3) adding a regularization term.

Let $\mathbf{y_T}$ and $\mathbf{y_p}$ represent the ground truth labels and predicted outputs. The predicted output of the model for N classes can be written as $\mathbf{z_N} = [z_1, z_2, z_3, ..., z_N]^T$. The i^{th} output of the model f can be defined as $z_i = f(y_T)_i$. The margins are calculated in the following way (similar to a sigmoid function):

$$M(y_T, y_p) = \frac{1}{1 + se^{-y_T \cdot y_p}} \tag{15a}$$

$$0 \leq M(x,y) \leq 1 \tag{15b}$$

where s is a scaling factor. The value of s, in our case, is 200 and has been experimentally determined.

Observation 1. (Asymptotic Behaviour) As $xy \to \infty, M(x,y) \to 1$ when $s > 0$ and $M(x,y) \to 0$ when $s < 0$
Proof. If $\lim_{xy\to\infty} M(x,y)$, then $e^{-xy} \to 0$, and, if $s > 0$, $se^{-xy} \to 0$. As a result, $(1+se^{-xy}) \to 1$ and $M(x,y) \to 1$. Conversely, if $s < 0$, se^{-xy} still approaches 0. However, $(1+se^{-xy}) \to 0$ and $M(x,y) \to 0$.

Observation 2. Due to the randomness of the labels in the training data, the margin $M(x,y)$ is bounded between 0 and 1 for all the real values of x and y.
Proof. As we know, the value of e^{-xy} lies between 0 and 1, i.e., it will always be positive for all the real values of x and y. Therefore, the denominator $1+se^{-xy}$ will always be positive and is bounded between 0 and s+1. When the denominator is inversed, the bounded range is back to the interval (0, 1). As a result, $M(x,y)$ is always bounded by 0 and 1.

Eqn. 15a is used to scale margins and apply enforced margins in a cross-entropy loss function. The scaling of the margins is performed as follows:

$$\Delta_y = (1 - M(y_T, y_p)).y_T + M(y_T, y_p).y_T.y_p \tag{16}$$

where Δ_y represents the margin scaling term. The cross-entropy loss function with enforced margins is given below:

$$CE(y_T, y_p) = -\sum_{j=1}^{N} y_{T_i}.\log\Delta_y \tag{17}$$

In order to avoid overfitting and improve feature selection by encouraging the sparser set of feature weights, the L1 regularization term (i.e., the summation of the absolute values of the model's coefficient) is added to the above loss function. In our initial study, we tested our loss function for both L1 and L2 regularization terms and found that our training model converges better with L1 regularization. As a result, we use the L1 regularization:

$$S(y_T, y_p) = CE(y_T, y_p) + \lambda \sum_{i}^{N} |y_{PI}| \qquad (18)$$

where λ is a hyperparameter that controls the amount of regularization.

4 EXPERIMENTAL EVALUATION

The performance of our proposed model is assessed and compared against the state of the art (Kim et al., 2018), as well as against the Attention UNet (Oktay et al., 2022), a model which we have built upon. Section 4.1 provides information about the datasets, and the construction of labeled ground truth. Section 4.2 discusses implementation details and also explains how we constructed our ground truth and how we vectorized the output of our proposed approach. Sections 4.3 and 4.4 analyze the performance of our method in both quantitative and qualitative ways.

4.1 Datasets

We evaluate our model on six datasets constructed by (Kim et al., 2018). They belong to three semantically distinct categories: characters (Chinese and Kanji), synthetic random lines (Lines), and sketches (Baseball, Cat, and Multi-Class). Each dataset consists of images in SVG format which are divided into training and testing sets; the distribution is described in Table 1. To prevent overfitting and improve the model's ability to generalize on unseen data, we created a validation set by further splitting the training set into a 70:30 ratio. Additionally, we are dealing with a specific use case that is inherently imbalanced in a heavy-tailed distribution manner. When evaluating the overall performance of each dataset, it is important to consider the distribution of the number of strokes within the dataset. Table 2 gives details about the number of vector paths used in each dataset.

The datasets from (Kim et al., 2018) are in SVG format. To use these data for training purposes, we need to convert them back to raster: the CairoSVG[1] library is used to convert SVG to PNG, outputting

[1](https://pypi.org/project/CairoSVG/)

Table 1: The number of images used for training, validation, and testing purposes.

	Chinese	Kanji	Line	Baseball	Cat	Multi
Train	5989	7216	31499	85285	77616	56000
Val	2567	3093	13500	36551	33264	24000
Test	951	1145	4999	13536	12318	8888

grayscale input raster images that are used for training. To generate the ground truth, each path extracted from the vector image is individually processed, creating a binary representation where all pixels within a path receive the same label. These labeled masks are then combined to form the final ground truth image. When assigning labels to each pixel along the path, pixels belonging to overlapping regions will ideally be assigned multiple labels. We opt for a simpler labelling approach and assign each pixel in the overlap the label of the last path passing through it. This approach ensures a coherent labeling strategy while enabling us to still detect strokes with some minor discontinuities.

4.2 Implementation

Our PyTorch model was trained on a GPU cluster with four NVIDIA P100 GPU with 12 GB of memory. The inference was performed separately on a single desktop NVIDIA GeForce GTX 1660 Ti GPU with 6 GB of memory so that an accurate runtime measurement could be obtained. For the characters and random line datasets, we trained our model for 500 epochs with an image size of 64 × 64 and batch size of 8. For the sketch datasets, we used a batch size of 32 and an image size of 128 × 128 and trained for 200 epochs.

We trained our model using the Adam optimizer (Kingma and Ba, 2017) and an initial learning rate of $1e^{-4}$. The maximum number of labels used for training our model is fixed at 128. Although this number is significantly larger than the highest number of vector paths among all the six datasets, it was chosen to maintain consistency with (Kim et al., 2018). Also, opting for this value did not have any negative impact on our model's performance. The scaling factor used in our loss function is 200, and the weight used for the compounded loss function is 0.5. These values were determined experimentally during the hyperparameter optimization of the datasets.

4.3 Quantitative Analysis

To evaluate our results, we measure per-stroke intersection-over-union (IoU) between our final output and its corresponding ground truth. Our results are compiled and compared to reference methods in Table 3. Overall, our method performed well for

Table 2: The minimum, maximum, mode, mean and standard deviation of the number of vector paths (i.e., stroke instances) used in training/validation and testing phase for each dataset.

Dataset	Minimum		Maximum		Mode		Mean		Std	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Chinese	1	1	33	28	11	10	11.75	11.73	4.39	4.32
Kanji	1	1	30	29	12	12	12.58	12.39	4.79	4.76
Random	4	4	4	4	4	4	4	4	0	0
Baseball	1	1	84	55	3	3	4.98	4.98	3.34	3.34
Cat	1	1	40	31	9	9	9.87	9.85	3.88	3.90
Multi-Class	1	1	115	48	3	3	7.03	7.0	4.50	4.45

Table 3: Per-pixel intersection-over-union (IoU) performed on the test set. Best results are shown in bold font. The IoU average excludes the random lines dataset, which was included as a counter-example.

	Chinese	Kanji	Baseball	Cat	Multi-Class	Avg.	Random
(Kim et al., 2018)	**0.958**	0.917	0.827	0.811	0.753	0.853	**0.872**
Attention UNet	0.884	0.918	0.853	0.433	0.742	0.766	0.276
Ours	0.937	**0.946**	**0.869**	**0.831**	**0.786**	**0.874**	0.271

Table 4: Average execution time taken by a single image at inference (in seconds).

	Chinese	Kanji	Random	Baseball	Cat	Multi-Class
(Kim et al., 2018)	54.9	26.589	8.46	853	233	409
Attention UNet	0.0106	0.0154	0.0146	0.0404	0.0365	0.0389
Ours	**0.0089**	**0.0095**	**0.0231**	**0.0179**	**0.0223**	**0.0109**

Table 5: Computational Complexity, measured in terms of number of parameters and GFLOPS. The down arrow indicates that it is better to have low values.

Model	Number of Parameters ↓	GFLOPs ↓
(Kim et al., 2018)	1.34 M	0.00266
Attention UNet	34.9 M	4.19
Ours	32.8 M	3.79

most of the datasets in terms of accuracy. However, for Chinese characters, (Kim et al., 2018) performed slightly better (+0.012) than our model. Our method requires a larger amount of data for increased performance; the Chinese dataset contains the least number of training images. However, we outperformed the other methods on the other datasets (excluding the random line dataset). This points towards our model's ability to generalize effectively when provided with a large dataset.

The outlier in performance is the random line dataset; our method outperformed (Kim et al., 2018) by an average of 0.0208 when excluding the random lines dataset. The IoU for the random line dataset for both Attention UNet and our method is less than 0.30, despite this dataset consisting of only four labels. This poor performance can be attributed to the totally random nature of the strokes, containing no spatial relationship between them. We hypothesize that as our model captures spatial relationships well, in the absence of these relationships, we are essentially training on "noise". Therefore, this serves as a counter-example, demonstrating our model's ability to leverage the spatial relationships between the instances, as it is able to perform well on all other datasets with more semantically structured contents.

4.3.1 Computation Time

We have compared the average computation time taken per image at the inference of our proposed model with the (Kim et al., 2018) method and Attention UNet. As shown in Table 4, our approach takes less than 50 milliseconds and is 1711500% and 64% faster than the (Kim et al., 2018) method and the Attention UNet, respectively.

4.3.2 Computation Complexity

The computation complexity, in terms of GFLOPs, and the number of parameters of our model and comparable approaches is shown in Table 5. Despite being less complex and having fewer parameters, (Kim et al., 2018) requires a significantly higher execution time. However, the calculation of the number of model parameters and GFLOPs does not include the iterative optimization of the Markov Random Field (MRF); as a result, the parameters and the floating

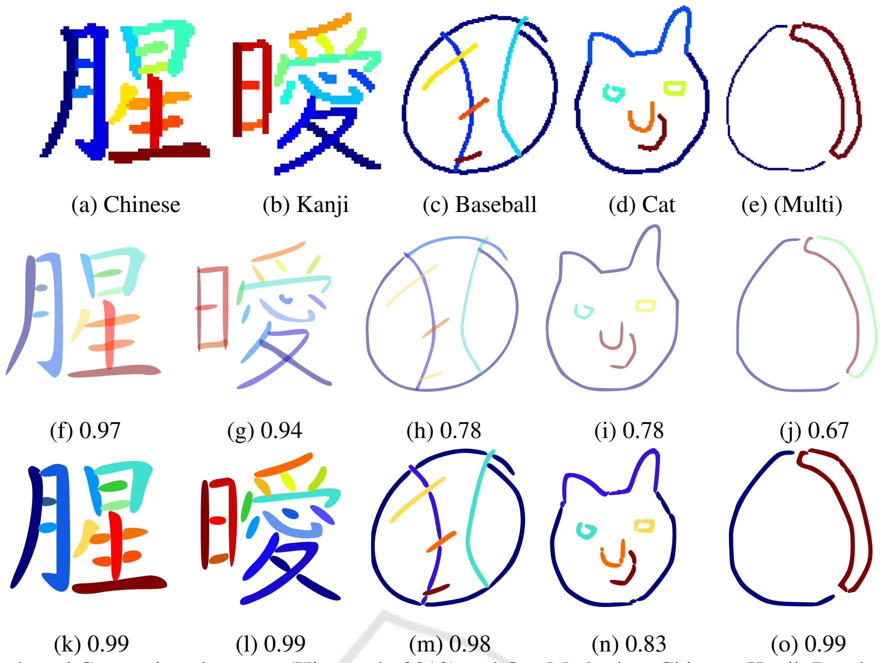


Figure 8: Stroke-based Comparison between (Kim et al., 2018) and Our Method on Chinese, Kanji, Baseball, Cat and Multi-Class dataset for successful cases, where *First row*: Ground truth, *Middle row*: Kim *et al.* IoU results *Last row*: Our IoU results.

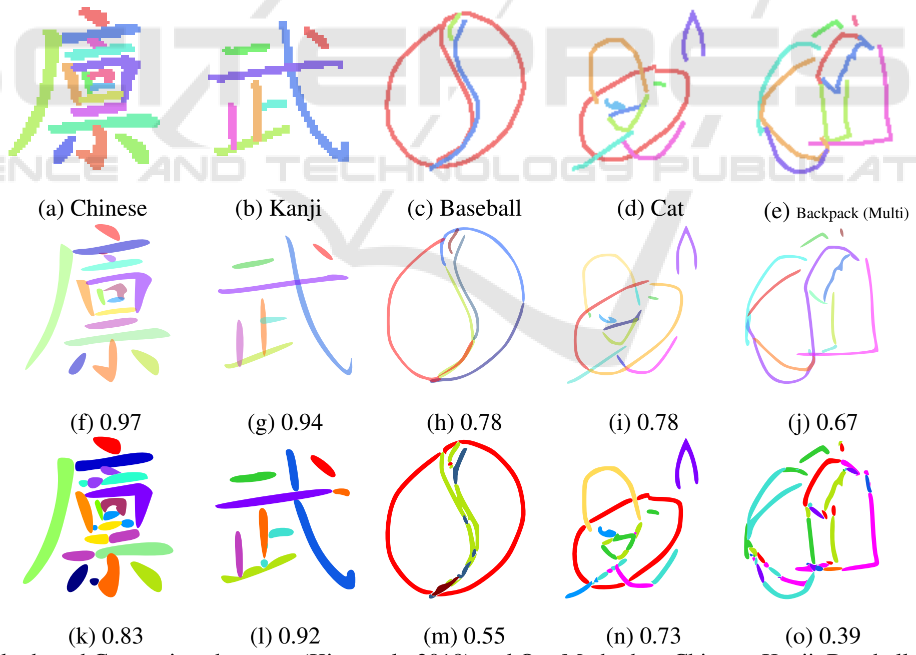


Figure 9: Stroke-based Comparison between (Kim et al., 2018) and Our Method on Chinese, Kanji, Baseball, Cat and Multi-Class dataset for successful cases, where *First row*: Ground truth, *Middle row*: Kim *et al.* IoU results. *Last row*: Our IoU results.

point operations calculated are significantly less than ours. Hence, comparing our model with (Kim et al., 2018) based on the computation complexity of the model may not be relevant.

Our proposed model demonstrates superiority over Attention UNet in terms of both model parameters and model complexity. Attention UNet has more than 2.1 million additional parameters compared to our proposed approach. Additionally, in terms of GFLOPs, our model showcases a reduction of 0.4 compared to the Attention UNet. This highlights our model's efficiency in terms of both parameter count and computational workload.

4.4 Qualitative Analysis

We visually compare our outputs with those of (Kim et al., 2018) using examples from all datasets except the random lines. Although both methods are able to segment correct instances of strokes in most cases, both have distinct shortcomings.

Figure 8 presents examples where our method performed qualitatively better than (Kim et al., 2018). As shown in Figure 8k and Figure 8l, despite the presence of a missing pixel in the overlapping regions of the ground truth, we are still able to maintain continuity in the strokes. A user can draw a cat in various styles, such as using a single stroke for the whole face, two separate strokes for the ears and face, or individual strokes for the ears and face. In Figure 8n, our model not only segments the instances but also infers the user's drawing style. This stands in contrast to (Kim et al., 2018), as shown in Figure 8i, which erroneously predicts the face and ears as one stroke (i.e. the most common drawing style).

Figure 9 shows some shortcomings of our method when compared to (Kim et al., 2018). In the character dataset (shown in Figure 9k and Figure 9l), it can be observed that multiple instances are assigned to a single stroke instance, resulting in a phenomenon known as "vector soup". However, if human intervention is allowed as a postprocessing step, the number of required edits would be minimal; thus our output could be successfully utilized for further editing. In Figure 9m and Figure 9n, some of the instances were mislabeled, however, our model accurately identifies circles in both categories, unlike (Kim et al., 2018). Overall, it can be concluded that our model is able to follow three main perceptual grouping principles (similarity, continuity and closure).

5 CONCLUSIONS

We propose a novel segmentation method that processes the input raster image by labeling each pixel as belonging to a particular stroke instance. Our novel architecture, named Multi-Focus Attention UNet builds upon Attention UNet by introducing multi-focus attention gates and highway skip connections, two architectural elements which play a key role in capturing global and local image context and in high computational efficiency. Our loss function includes a margin-regularised component which allows us to handle successfully a heavy-tailed label distribution, as well as infer correctly the user's drawing style. Our approach is significantly faster, exceeding state of the art by seven orders of magnitudes. Future work involves extending our focus to complex line drawing art that would significantly benefit from image vectorization when shown on large displays.

REFERENCES

Bartolo, A., Camilleri, K. P., Fabri, S. G., Borg, J. C., and Farrugia, P. J. (2007). Scribbles to vectors: preparation of scribble drawings for cad interpretation. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 123–130.

Bouton, G. D. (2014). *CorelDRAW X7*. Mcgraw-hill Education-Europe.

Cao, K., Wei, C., Gaidon, A., Arechiga, N., and Ma, T. (2019). Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32.

de Casteljau, P. (1963). "courbes et surfaces a poles," technical report. *Citroen, Paris*.

Dori, D. (1997). Orthogonal zig-zag: an algorithm for vectorizing engineering drawings compared with hough transform. *Advances in Engineering Software*, 28(1):11–24.

Favreau, J.-D., Lafarge, F., and Bousseau, A. (2016). Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)*, 35(4):1–10.

Graves, A. and Graves, A. (2012). Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE ICCV*.

Hilaire, X. and Tombre, K. (2006). Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):890–904.

Inoue, N. and Yamasaki, T. (2019). Fast instance segmentation for line drawing vectorization. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 262–265. IEEE.

Jimenez, J. and Navalon, J. L. (1982). Some experiments in image vectorization. *IBM Journal of research and Development*, 26(6):724–734.

Kim, B., Wang, O., Öztireli, A. C., and Gross, M. (2018). Semantic segmentation for line drawing vectorization using neural networks. In *Computer Graphics Forum*, volume 37, pages 329–338. Wiley Online Library.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Kultanen, P. (1990). Randomized hough transform (rht) in engineering drawing vectorization system. *Proc. MVA*, pages 173–176.

Najgebauer, P. and Scherer, R. (2019). Inertia-based fast vectorization of line drawings. In *Computer Graphics Forum*, volume 38, pages 203–213. Wiley Online Library.

Oktay, O., Schlemper, J., Le Folgoc, L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. (2022). Attention u-net: Learning where to look for the pancreas. In *Medical Imaging with Deep Learning*.

Roosli, M. and Monagan, G. (1996). Adding geometric constraints to the vectorization of line drawings. In *Graphics Recognition Methods and Applications: First International Workshop*, pages 49–56. Springer.

Selinger, P. (2003). Potrace: a polygon-based tracing algorithm.

Sorensen, T. A. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. *Advances in neural information processing systems*, 28.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., and Jorge Cardoso, M. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop*, pages 240–248. Springer.

Tombre, K., Ah-Soon, C., Dosch, P., Masini, G., and Tabbone, S. (2000). Stable and robust vectorization: How to make the right choices. In *Graphics Recognition Recent Advances: Third International Workshop*, pages 3–18. Springer.

Wood, B. (2012). *Adobe illustrator CS6 classroom in a book*. Adobe Press.

Zhang, Y., Zhu, M., Zhang, Q., Shen, T., and Zhang, B. (2022). An approach of line vectorization of engineering drawings based on semantic segmentation. In *2022 IEEE 17th Conference on ICIEA*, pages 1447–1453. IEEE.