Using 2D Segmentation Deep Neural Network to Detect, Classify, and Remove Handwritten Annotations

MATTHEW P. BURRUSS
A final assignment for CS 5891: Deep Learning
Vanderbilt University April 2020

Abstract

Automatic detection and classification of annotations is useful in a variety of domains including augmented reality (AR) and virtual reality (VR). A growing field in AR is sketch-based authoring, where sketches on a 2D medium, like paper, are converted into a 3D virtual object. In sketch-based authoring applications, annotations can be used to customize and interact with the virtual content. However, these annotations must be removed in order to maintain the integrity of the derived virtual 3D object. The contributions of this report are two-fold. Firstly, heavy data augmentation is used to create a segmentation data set using the EMNIST handwritten character classification data set as a starting point. The augmented data set is used to train U-Net, a 2D segmentation network. Secondly, this paper presents a case study where the segmentation network is deployed in a sketch-based AR application to allow the use of annotations to alter the properties of the virtual content, for example changing its color. A Google Colab notebook was used to perform data augmentation and train the segmentation network https://colab.research.google.com/drive/1NC0uj2SfhoQalnspKcLX2T_h6h18V3Cv.

I. INTRODUCTION

Sketching ideas on paper is one of the simplest ways we are able to communicate. Sketching is a creative way for people to explore ideas and innovations. However, an obvious limitation of traditional sketches is the restriction to a 2D medium. Augmented reality (AR) uses computer-graphics to enable a user to interact with 3D virtual content in a mixed reality environment and provides a medium in which users can create 3D content. For example, AR systems have been used to transform real 3D objects into 3D virtual objects which can then be manipulated in the AR environment [1]. However, when it comes to content creation, sketching still remains one the easiest tools available to use. Previous AR systems have used sketch-based authoring to convert 2D objects into 3D mechanical objects [2] and to educate children on basic geometry [3]. However, in order to extend sketch-based authoring to more creative and complex domains, the 3D content must also be customizable and easily manipulated.

One simple way to manipulate sketch-based virtual content is through annotations; however, this presents new challenges in that the annotations must not degrade the fidelity of the created virtual object. One approach to this problem uses a limited set of annotations and requires that the annotations use a separate color than the 2D sketch [2]. However, such an approach is not user-friendly, which is required for widespread adoption of sketch-based applications in industries like fashion, interior design, and manufacturing.

This report presents a new approach to detect, classify, and remove annotations using a 2D segmentation model. After segmentation, the label of the segmented annotation can be mapped to specific actions to manipulate the virtual object. Furthermore, because each pixel is labeled, the annotation can be ignored by the algorithm extracting the virtual object from the sketch in order to avoid degrading the integrity of the virtual object. The segmentation model is trained on a heavily augmented data set which is based on the EMNIST data set [4]. Finally, this report describes a case study in which the trained segmentation model is deployed in a sketch-based AR application to create a green-sided virtual pyramid from annotated 2D sketches.

II. METHODOLOGY

A. Problem

In order to address the problem of detecting, classifying, and removing annotations from an RGB image, two problems must be addressed. The first problem is the lack of an existing data set. To the best of the author's knowledge, no data set exists which contains labeled segmentations of handwritten characters. To address this problem, we use heavy data augmentation of the EMNIST data set. The EMNIST data set includes handwritten lowercase and uppercase letters and is traditionally used in classification tasks. However, the augmentation procedure is able to use EMNIST to create a data set suitable for segmentation. The second problem

that must be addressed is training a deep neural network (DNN) segmentation model to perform segmentation of the handwritten characters. In this paper, the popular U-Net architecture is used which consists of a contracting path that down samples the original image and an expanding path which ultimately predicts the label of each individual pixel [5].

B. Data Augmentation

The EMNIST data set contains images of handwritten characters and letters and is an extension of the popular MNIST data set [6]. Each input consists of a centered, 28x28 gray-scaled handwritten digit (0-9), lowercase letter (a-z), or uppercase letter (A-Z).

Traditionally, the EMNIST letter data set is used for classification where the output of the k'th instance is $y^k \in \{0, ..., 26\}$; however, for this segmentation task, the uppercase and lowercase letters are used as input into the data augmentation procedure whose output is paired with a target segmentation map where the label of each pixel $I^k(i,j)$ in the input image I^k is $y^k_{i,j} \in \{0, ..., 27\}$. In this scheme, the labels $\{1, ..., 27\}$ relate to the set $\{A, ..., Z\}$ and $\{a, ...z\}$ and $y_{i,j} = 0$ indicates that pixel $I(i,j)$ is not a letter.

To create the segmentation data set, an empty 128x128 matrix X^k was converted into a realistic, paper-like material by adding random blotches of various intensities and sizes and smoothening the background with noise. X^k was then normalized by dividing by 255. The original 28x28 EMNIST input image was also normalized and then randomly upscaled and downsampled using interpolation. Each pixel value was flipped ($I^k(i,j) = 1 - I^k(i,j)$) to make the character black on a white background, to mimic the majority of sketches.

The ground truth segmentation map was created by first initializing an empty 28x28 matrix T^k and setting $T^k(i,j) = y^k_i$ when $I^k(i,j) < 0.2$. Next, a random position in X^k was selected and I^k was inserted into X^k at this position wherever $I^(i,j) < 0.2$. Furthermore, T^k was inserted into the same position in an empty 128x128 matrix Y^k. Then, Y^k was one-hot encoded along the channels producing a matrix of size 128x128x27. The pair (X^k, Y^k) represents the k'th input-output pair used to train the segmentation model. This procedure was used to create 4000 training, 1000 validation, and 500 testing instances.

Before loading an input-output pair, further data augmentation was used to increase the model's invariance. First, for each input, a rectangle, circle, or a line were randomly drawn on the image with various scales and locations. Then, the brightness of the image was randomly increased or decreased to make the model invariant to pixel intensities. Next, the input was randomly thinned using openCV's *erode* [7] function to change the thickness of the annotation. Finally, the annotation was randomly translated to increase positional invariance. When necessary, the target matrix was modified to accommodate for the augmentations.

C. Model Description

This report uses U-Net, a popular model for performing 2D segmentation [5]. Figure 1 shows a graphical representation of the model. The model has 23 convolutional layers and downsamples the input using a series of 3x3 convolutions followed by ReLU activation and 2x2 max pooling. The upsampling region is accomplished using 2x2 up-convolutions. The final 1x1 convolution maps the final 64-channel feature map into the number of classes (k=27).

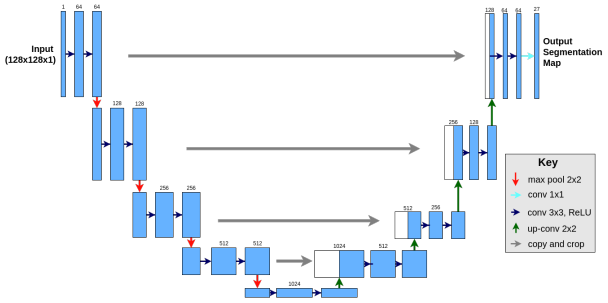


Fig. 1: The U-Net architecture where where each blue box is a multi-channel feature map and the white box is a copied feature map. The number of channels is denoted on the top of the box. The output of the model is a 27-channel segmentation map.

D. Training Procedure

The U-Net architecture is defined using the Pytorch deep learning library API and trained on a Google Colab GPU run-time. The model is trained for 190 epochs using a batch size of 32 and a learning rate of 0.001, using the Adam optimizer with default parameters [8] and categorical cross entropy loss which is defined below in Equation 1.

$$J(\hat{y}_i, y_i) = -\sum_{i}^{C} y_i log(\hat{y}_i) \tag{1}$$

In Eq. 1, C is the number of channels, $\hat{y}_i$ is the predicted segmentation map and y_i is the ground truth segmentation map. The validation and training loss curve are shown in Figure 2.

E. Evaluation on Augmented Test Data

A common approach to evaluating a segmentation model is to use the Sørensen-Dice coefficient or simply dice coefficient [9] which can be used to measure

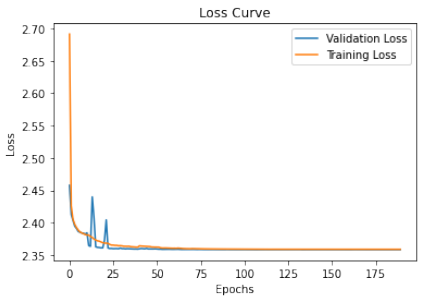


Fig. 2: Loss curves (epochs=190).

the similarity between $\hat{y}_i$ and y_i. The Dice coefficient (DSC) is defined in Eq. 2 below

$$DSC = \frac{2|\hat{y}_i \cap y_i|}{|\hat{y}_i| + |y_i|} \tag{2}$$

where a dice score of 1 indicates exact similarity between the segmentation maps and a score of 0 indicates complete dissimilarity. The dice score can also be interpreted as measuring the ratio of the true positive (TP) to false positive (FP) and false negative (FN) as shown in Eq. 3.

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{3}$$

On the test data (n=1000), the mean dice score was **0.996**, the standard deviation was **0.004**, and the median was **0.997**.

F. Evaluation on Handwritten Examples

The trained U-Net model was then tested on a few non-EMNIST derived examples. To do this, a camera captured an image of several handwritten letters. The images were then gray-scaled, resized to 128x128 using interpolation, and normalized by dividing by 255. Figure 3 shows three examples of non-augmented test examples and the resulting segmentation maps. The output segmentation map is refined to only show the second most common prediction to ignore the predictions where $y(\hat{i}, j) = 0$. Without a ground truth, it is difficult to measure the accuracy of the results; however, "eyeing" the segmentation results seem to show that the model is able to accurately segment non-augmented results, providing evidence that the heavy data augmentation was effective.

III. Case Study: AR Sketch-Based Authoring of a Green-Sided Pyramid

In this section, the trained U-Net model is evaluated in an AR sketch-based authoring application where it is tasked with detecting, classifying, and removing annotations from a hand-drawn sketch. Specifically, a pyramid with green sides will be constructed in this case study.

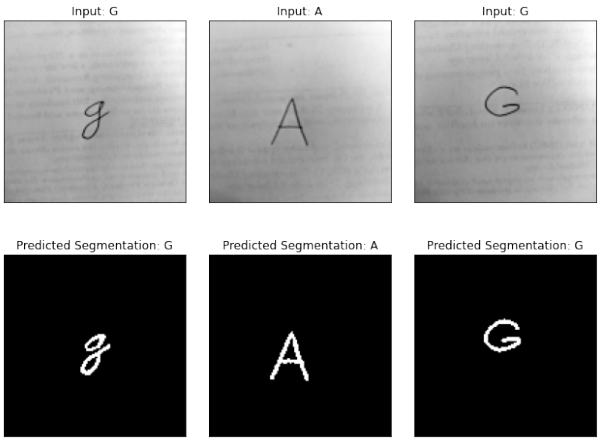
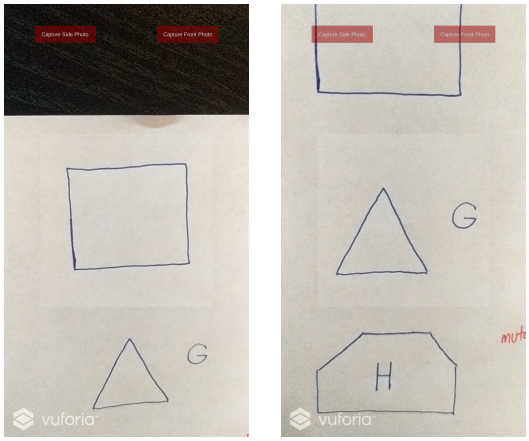


Fig. 3: Top Row: Three examples of gray-scaled and resized (128x128) inputs. Bottom row: The resulting segmentation showing the second most common prediction to ignore prediction of zero.

The AR sketch-based application consists of two components: a Python server and the client AR application deployed on an Android device. The AR application provides an interface for the user to capture images of sketches of the side and the front face of the virtual object that is to be created. Figure 4 shows an example of the client AR interface. The AR application sends these images to the server for processing.



(a) Front face sketch **(b)** Side face sketch

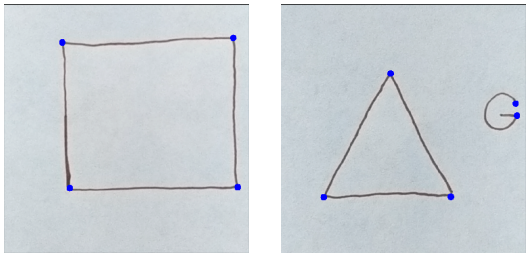
Fig. 4: A screenshot of the AR application. The two red buttons control which faces (front or side) are being captured and then sends an image of the contents in the lightly shaded box to the server for processing.

The server first performs segmentation using the trained U-Net model to look for annotations with defined actions, such as the annotation 'G' which maps to coloring that specific face green. Upon detecting and classifying a defined annotation, the server will detect the corners of the sketch while simultaneously ignoring

any corners that overlap the predicted segmentation map, thus ensuring the integrity of the sketch is maintained. Once the processing step is complete, the server sends the necessary information to the AR application to construct the virtual content. The user can then interact with the object in the mixed reality setting.

Further details of the corner detection algorithm and generation of 3D objects from sketches of the front and side faces is beyond the scope of this report; however, details can be found in the Github repository which also includes a demo video of the AR application [1].

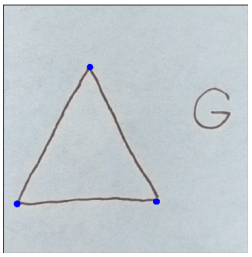
To create a green-sided pyramid, the user first captures a photo of a sketch of a square for the front face and a sketch of a triangle for the side faces with the annotation 'G' to indicate the sides should be colored green. Figure 5a shows the results of extracting the corners from the sketch of the front face, and Figure 5d shows the corners extracted from the sketch of the side faces after using the segmentation map shown in 5c to ignore the annotation.



(a) Front face detected corners

(b) Side face corners w/out annotation removal

(c) Side face annotation segmentation



(d) Side face detected corners w/ annotation removal

Fig. 5: (a) The detected corners of our sketch of the front-face are shown in blue. (b) The annotation interferes with the integrity of the corner detection algorithm when not ignored, however, after segmentation (c) the corner detection algorithm can ignore the annotation and accurately detect the corners of the triangle (d).

Figure 5b shows how the integrity of the corner detection algorithm is compromised if the segmentation map is not predicted. However, the annotation can be ignored using the results of the segmentation map in order to maintain the integrity of the sketch. Figure 6 shows the the final constructed green-sided pyramid projected into the mixed reality environment.

[1] https://github.com/burrussmp/Augmented-Reality-Sketch-Authoring

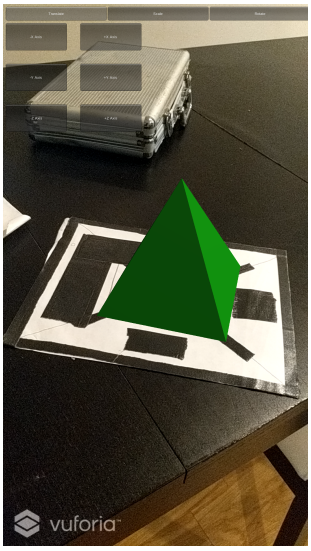


Fig. 6: The green-sided pyramid constructed by the AR sketch-based application.

IV. Related Work

This section explores work related to segmentation, data augmentation, and sketch-based authoring. One of the earliest segmentation models used a sliding window approach to predict the label of each window by considering a local region or patch [10]. The drawback of this technique was that it is slow to run because it must process each patch individually and requires setting the size of sliding window which introduces a tradeoff between useful context and label specificity. The fully convolutional network (FCN) was proposed as a solution to the sliding window. FCN does not contain any dense layers and uses only convolutional blocks, including 1x1 convolutions, to extract features [11]. A benefit of this architecture is that inputs of various sizes can be used without modification to the architecture and it is able to process the entire image without using patches. The U-Net architecture used in this report was originally designed as an improvement of the FCN architecture [11] and specifically designed for use in segmentation of medical data where the exact label of pixels is often needed and training data sets are often small. U-Net is able to outperform FCN due the upsampling layers and the residual connections connecting the contracting and expanding portions of the model. As a result, U-Net requires fewer training instances than FCN and typically has a higher mean dice score in segmentation tasks.

The value of data augmentation in learning invariance has been shown in the scope of unsupervised learning tasks [12]. Heavy data augmentation was used in this paper in order to increase the invariance of U-Net to pixel intensity, annotation size, annotation thickness, etc. In a medical image segmentation task, data augmentation

has also been shown to reduce the number of required training inputs through effective augmentation [13]. Reducing the number of training instances is also useful in efficiently using GPU space, which is often quickly used up in segmentation tasks due to the large number of learnable parameters and size of the input space.

There have been many systems proposed for authoring 3D content in AR/VR environments, including BuildAR which provides a graphical user interface to simplify the creation process [14] and ARpm which uses 3D Studio Max to create polgonal models [15]. However, this report focuses soley on sketch-based authoring techniques due to its ease-of-use. For example, one AR sketch-based application focused on using the application to educate children on geometry [3]. Their technique of performing sketch-based authoring was most similar to the one used in the case-study; however, this approach did not describe any method for supporting annotations to manipulate the 3D content. Other sketch-based authoring interfaces have supported annotations [2]; however their approach required that the annotations be created with red ink so that the annotations could be ignored by the 3D object extraction algorithm. The Sketch Understanding Group at MIT designed a system called "Magic Paper" to analyze user's strokes to provide context in addition to geometric constraints to determine the shape being drawn with fidelity [16]. This annotation segmentation network mainly complements systems such as "Magic Paper" in that the segmentation model can be used to allow whatever system is processing the sketch to ignore the annotation, as segmentation maps can be intermittently predicted.

V. Discussion

Incorporating annotations into a sketch-based AR application is an intuitive way to increase customization because annotations are easy to add to a sketch and can map to different actions depending on the application domain. Still, annotations must be processed and removed before an algorithm extracts the 3D object from the sketches. This report showed how heavy data augmentation was able to create a data set suitable for training a model to perform 2D segmentation of hand-written letters, allowing the AR system to use the segmentation results to remove the annotation. In addition to creating a suitable data set, data augmentation was used to increase the model's invariance. For example, changing the scale of the annotation and its thickness introduced invariance to the size and stroke width of the annotation; randomly altering the brightness increased invariance to pixel intensity; and translating the annotation increased invariance to location.

One limitation of the data augmentation was that the model was never exposed to multiple annotations in a single input-output pair. In reality, the AR application would need to support multiple annotations to provide a fully customizable experience. However, the case study provided evidence that the trained model can accurately segment non-augmented test inputs and there is no obvious reason why the data augmentation procedure could not be extended to support multiple annotations. Future work should continue advancing the data augmentation procedure, for example changing the color of the paper-like background and ink of the annotation.

The annotation detection, classification, and removal was successful due to the accuracy of the 2D U-Net architecture. The residual connections connecting the down sampling and up sampling portions of the architecture provide context during training that increases the integrity of the segmentation. Overall, this shows the promise of using DNNs in this application. Future work should look at multi-task architectures to perform not only segmentation, but also corner detection and even 2D object classification. A multi-task architecture can use the classification results as context for a network to more accurately determine the location of corners. It is even possible that whole 3D objects could be extracted from a single sketch, allowing for a more user-friendly sketch-based authoring experience. Finally, a DNN that can not only extract the basic shape of the 3D object but also estimate the surface normals could be used to create highly advanced virtual content. This is the most promising application of DNNs in sketch-based authoring tasks and should be explored in future work.

VI. Conclusion

Sketch-based authoring will become more useful as AR tools become more refined and widespread because sketch-based authoring is a natural method for rapid prototyping and customization of 3D objects. One limitation of current sketch-based applications is the inability to flexibly detect, classify, and remove annotations from the sketch. This report showed that heavy data augmentation can be used to train a DNN to accurately perform segmentation of handwritten letters. A U-Net segmentation model was used in an AR sketch-based authoring application to remove annotations from a sketch so that the integrity of the virtual object was maintained. Future work should further advance the data augmentation procedure and investigate extending DNNS to other tasks in the sketch-based authoring process.

REFERENCES

[1] A. van den Hengel, R. Hill, B. Ward, and A. Dick, "In situ image-based modeling," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 107–110, IEEE, 2009.

[2] O. Bergig, N. Hagbi, J. El-Sana, and M. Billinghurst, "In-place 3d sketching for authoring and augmenting mechanical systems," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 87–94, IEEE, 2009.

[3] S. M. Banu, "Augmented reality system based on sketches for geometry education," in *2012 International Conference on E-Learning and E-Technologies in Education (ICEEE)*, pp. 166–170, IEEE, 2012.

[4] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters. corr abs/1702.05373 (2017)," *arXiv preprint arxiv:1702.05373*, 2017.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[6] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[7] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] T. Sørensen, T. Sørensen, T. Sørensen, T. SORENSEN, T. Sorensen, T. Sorensen, and T. Biering-Sørensen, "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," 1948.

[10] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, pp. 2843–2851, 2012.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[12] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in neural information processing systems*, pp. 766–774, 2014.

[13] A. Zhao, G. Balakrishnan, F. Durand, J. V. Guttag, and A. V. Dalca, "Data augmentation using learned transformations for one-shot medical image segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[14] "Buildar."

[15] P. Fiala and N. Adamo-Villani, "Arpm: an augmented reality interface for polygonal modeling," in *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, pp. 196–197, IEEE, 2005.

[16] R. Davis, "Magic paper: Sketch-understanding research," *Computer*, vol. 40, no. 9, pp. 34–41, 2007.