Deep Learning-based Semantic Segmentation: Vesuvius Challenge

Zongyuan LANG z5319088
zongyuan.lang@student.unsw.edu.au
Zikai Zhang z5400473
zikai.zhang@student.unsw.edu.au
Yu TIAN z5325731
yu.tian2@student.unsw.edu.au
Houke Zhao z5429120
houke.zhao@student.unsw.edu.au
Ruotian(Jarrett) Jia z5240505
jarrett.jia@student.unsw.edu.au

***Abstract*—In this paper, we will present deep learning models which are using the fully convolutional network, Unet, Unet++, attention Unet, and Deeplabv3 to detect ink stains in volcanic ash. Our deep learning model can well predict the presence of ink detection in the input 3D CT images. The training dataset we choose is the EduceLab-Scrolls open-source dataset. The EduceLab-Scrolls dataset consists of CT images obtained by scanning four fragments using X-rays. EduceLab-Scrolls also provides black-and-white images visually labeled to show whether there is ink in an area of a CT photo. This creates a learnable mapping between the 3D image and the deep learning model. This mapping allows supervised learning for the detection of "unseen" carbon inks in CT images. This task is "impossible" even for human labeling experts. Our model is able to accurately display lines of text on reel segments where the underlying facts are known. The final text predicted by our model is validated by quantitative image metrics for visual recognition.**

***Index Terms*—ink detection, fully convolutional network, U-Net, U-Net++, attention U-Net, Deeplabv3**

Note: This paper is based on the Kaggle artificial intelligence competition topic ink detection(Resurrect an ancient library from the ashes of a volcano). For more information please refer to these websites: scrollprize.org, vesuvius-challenge-ink-detection.

I. Introduction

In 79 AD, Mount Vesuvius erupted. At Herculaneum, 20 meters of hot mud and ash buried a huge villa once owned by Julius Caesar's father-in-law. The villa contained a library with a large collection of papyrus scrolls. Hot gases and ash from Mount Vesuvius heated the papyrus scrolls to combustion temperatures and depleted the oxygen surrounding the scrolls, carbonizing them. But they were also preserved. Over the centuries, almost all ancient documents exposed to the air decayed and disappeared, while the Villa Pulp Library waited intact underground.

The buried villa was discovered by an Italian farmer in 1750 AD. Excavations revealed more than 900 of these scrolls. These scrolls were carbonized, ashen, and very fragile. But the temptation to open these scrolls is great. If we could read these scrolls, we could add a great deal of recognizable ancient writing and scholarship. This is why people call these papyri that have been carbonized "Herculaneum papyri". It is the only set of ancient libraries ever discovered.

Early attempts to open the scrolls, unfortunately, destroyed many of them. An Italian monk, who had labored for decades to open several of them, found that they contained philosophical texts written in Greek. Many of the scrolls that have been forced open have shattered into many small pieces, or parts that once existed have been completely lost. But there are still hundreds of scrolls that are well preserved, but their contents are still completely unknown.

In recent years, as physics technology has evolved, advanced non-destructive imaging techniques have replaced the violent physical destruction of the past. In 2019 Dr. Seales and his team were determined to virtually unwrap the "Herculaneum papyrus", so they began testing a new idea. Some of the detached fragments of the papyrus were readable under infrared light, and these fragments appeared to serve as the data underlying a machine-learning model that could detect otherwise invisible ink stains from X-rays.

To obtain X-rays at the highest possible resolution, the team used a particle gas pedal to scan two complete

scrolls and several fragments. With 16 bits of density data per pixel at a resolution of 4-8 microns, they believe the machine learning model can capture subtle surface patterns on the papyrus that indicate the presence of carbon-based ink.

However, although modern technology has greatly assisted in the recovery of various ancient documents, non-invasive techniques still struggle to reveal the contents of the Herculaneum papyri. First, detecting the ink on the writing substrate was a problem. The ink used by the Herculaneum writers was almost pure carbon. The main component of the papyri material is the organic plant book, and the remaining part after burning is also carbon. As a result, the contrast of the ink is virtually invisible under X-rays, and the traditional virtual unwrapping of Herculaneum's CT data produces an image that shows a blank page with no text.

As difficult as these challenges are, we believe humans can now be met with advances in multiple technologies. With the gradual development of image resolution, deep learning models, and specific algorithms, it is enough to predict the content of Herculaneum papyri. A deep learning model can detect low-contrast inks. This model makes good use of the separated fragments of visible ink detection as the training dataset.

With the accumulation of data, the EduceLab-Scrolls multimodal image dataset [1] was created. EduceLab-Scrolls contains the data that has been used to date in the non-invasive presentation of the hidden text of the Herculaneum Scrolls. The dataset contains high-resolution scroll images from X-ray micro CT. Both complete scrolls and small detached scroll fragments are available. To our knowledge, this is the largest published dataset in the field of ancient texts.

The most important achievement of EduceLab-Scrolls is not only the resolution but also the number of Herculaneum papyri CT images, but also the matching of volumetric X-ray CT images with 2D surface images. With these aligned labels shown in Figure 1 [1], supervised learning can be performed on CT input images. This trained a model capable of detecting ink marks in CT images. To the best of our knowledge, this image alignment framework is unique to this dataset and is the largest dataset released to date for deep learning.

We also trained five deep-learning models to learn CT images to determine the presence or absence of ink detection. These models are fully convolutional networks(FCN), U-Net, Deeplabv3, U-Net++, and attention U-Net. In addition, we compared the accuracy, F0.5, and other model parameters of these models to discuss the performance of these models in ink detection.

FCNs classify images at the pixel level, thereby addressing the problem of image segmentation at the semantic level [2]. Different from the classic CNN that uses fully connected layers to obtain fixed-length feature vectors for classification after convolutional layers, FCN accepts input images of arbitrary size and uses deconvolutional layers to upsample the feature maps of the last convolutional layer. The feature maps are then restored to the same size as the input image, thereby generating per-pixel predictions while preserving the spatial information in the original input image, and finally, the upsampled feature maps are classified pixel-by-pixel.

DeepLabv3 is also a deep-learning architecture for semantic image segmentation [3].DeepLabv3 uses dilation convolution, which allows the model to capture information at multiple scales without increasing computational complexity. By varying the dilation rate, the network can focus on context regions of different sizes. In addition, DeepLabv3 introduces ASPP(Atrous Spatial Pyramid Pooling), which captures multi-scale image information by employing dilation convolutions with different dilation rates in parallel. This pyramid structure ensures that objects at different scales are captured efficiently.

And U-Net is proposed in 2015 to solve the medical image segmentation problem [4]. U-Net has two parts, one is used to extract the abstract features of the image and the other is the feature fusion operation. Compared with the traditional FCN, U-Net realizes the fusion of shallow low-resolution and deep high-resolution information through the feature fusion operation, fully utilizes the contextual information of the image, and uses the symmetric U-shape structure to make the feature fusion more thorough.

In addition, U-Net++ focuses on exploring the question of how big the encoder and decoder need to be [5], as a basis for proposing networks that incorporate U-Net structures of different sizes. The innovation is to incorporate U-Net structures of different sizes into one network.

Attention U-Net enhances its performance by introducing attention modules [6]. These modules help the model to focus on the critical parts of the image and ignore the less important regions. By directing the model to focus on important features, the accuracy and consistency of segmentation can be improved. In Attention U-Net, gated attention modules are used to selectively weight features between the encoder and decoder. This mechanism uses a soft attention method that learns the weights of specific regions and combines global and local information to better capture key features. Compared to traditional U-Net, Attention U-Net's incorporation of the attention mechanism allows better handling of complex patterns and relationships in images, resulting

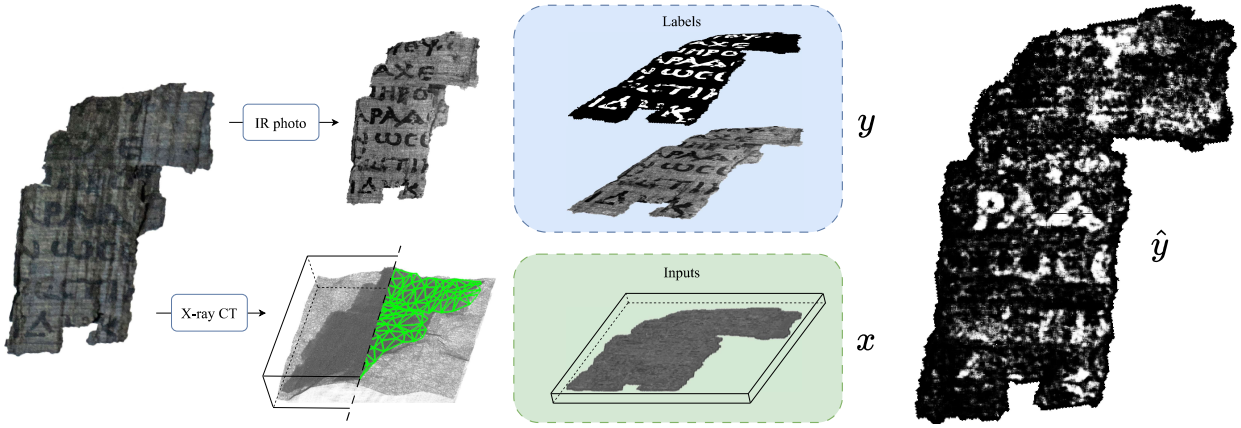

Fig. 1: EduceLab-Scrolls dataset geometry demo clip [1].

in improved segmentation quality and accuracy.

The experimental results show that a deep learning model trained on EduceLab-Scrolls is able to recover the hidden text. This method operates on image pixels, identifying them as ink or non-ink. Our model performs calculations on the pixels of an image to identify it as ink or not. But a trained model doesn't understand alphabets, optical character recognition (OCR), paleography, or handwriting analysis. The text characters that appear are sheerly the result of drawing localized ink-detection "dots" on the resulting image.

Since the scrolls were first unearthed, people have been looking for a solution to this research challenge. This is despite the fact that there are other non-invasive methods for discovering texts from the Herculaneum papyri. However, we believe that deep learning will be the most suitable method capable of generating large amounts of text suitable for literary research. The final predicted results of our model can be seen by visually comparing the valid set label to see that our deep learning model is able to meet the performance criterion of accurately recognizing ink blots.

II. Exploratory Data Analysis

A. Data description

The challenge was to recover where the ink was located by performing 3D X-ray scans of detached fragments of ancient papyrus scrolls. It is an important subproblem in the overall task of the Vesuvius Challenge.

B. Overview directories files

[train/test]/[fragment]/surfacevolume/[image].tif slices from the 3d x-ray surface volume. Each file contains a greyscale slice in the z-direction. Each fragment contains 65 slices. Combined this image stack gives us the 'width * height * 65' number of voxels per fragment.

The ***train/*** directory has three 3d x-ray surface volumes. The files in these three 3d x-ray surface volumes are all the same.

The ***test/*** directory has three 3d x-ray surface volumes. Which together are roughly the same as the ***train/*** directory.

train	test
1	a
2	b
3	

TABLE I: train and test directory structure table.

C. Data visualization

Each of train volumes has next files:

- mask.png — a binary mask of which pixels contain data.
- -ir.png — the infrared photo on which the binary mask is based.
- inklabels.png — a binary mask of the ink vs no-ink labels.
- inklabels.csv — a run-length-encoded version of the labels, generated using this script.

First, the files of the 1st volumes of the train directory are displayed

Shown below is the ir.png in Fig.2

It's an infrared photo since the ink is better visible in infrared light. Based on observation, no more information can be gained from this picture, it is just a display.

Shown below is the ***mask.png*** and ***inklabels.png*** images in Fig.3

Shown in this file is a run-length encoded version of the inklabels.png image in tableII.

Fig. 2: ir.png

Fig. 3: mask and inklabels

The composition of the other two volumes of the ***train/*** directories is the same as that of the first volume of the ***train/*** directories. The following is the display of each file in Fig.4 and Fig. 5

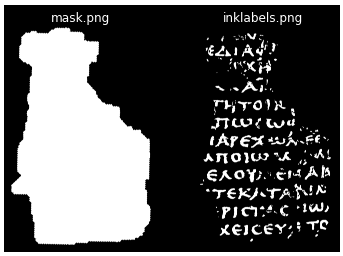

Fig. 4: train/2 image

Fig. 5: train/3 image

D. Labels of train directory

The table shows the ink distribution for the three training instances. To compute the distribution, first take their mask, and then compute the distribution, since this way the assumptions will be made for training and inference. In the visualization, 3 labels are shown in each table in Fig.6

	Id	Predicted
0	1	606211 19 612538 26 618867 39 625196 44 631525...

TABLE II: visualization csv file table.

- No ink — is inside the mask, but is not ink.
- Ink— is inside the mask and ink.
- Empty space— is outside the mask.

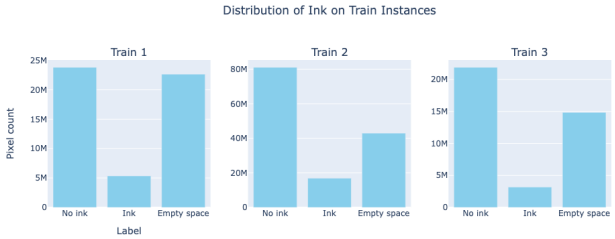

Fig. 6: Labels of train

E. Overview volumes of the test/ directory

Unlike the train/ directory, In the test/ directory, there are only two sets of files, each containing a ***mask.png*** image and grayscale slices on the z-axis, and these slices are also 65 in total. When loading the ***mask.png*** image of the two groups, it is the Fig.7

Fig. 7: test masks

After stitching them together, you will find that they are the same as ***mask.png*** in the 1st volume of the ***train/***directory, and they have the same height plot in Fig.8 and the sum of their Hight is the same as the Hight of the volume of the train 1st, table.III

name	size
Mask for train 1st images size:	(6330, 8181)
Mask for test a images size:	(6330, 2727)
Mask for test b images size:	(6330, 5454)

TABLE III: mask images in test table.

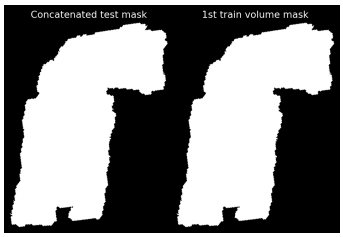

Fig. 8: concatenated test

F. Training set and Test set

According to the structure of the provided data set, for the convenience of training, after discussion, we decided to combine the three image stacks first, then select a specific area in the new image stack, and divide the pixels in the bitmask into rectangular areas Two areas, inside and outside. These two regions will be used as training and validation sets, respectively.

Fig.9 shows combine three image stacks into a new image stack Fig.10 shows the selected test set region.

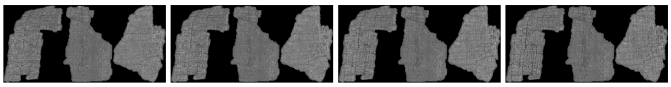

Fig. 9: new image stack slices

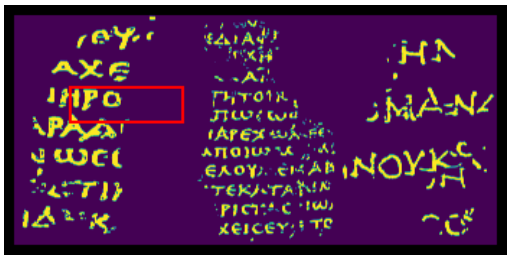

Fig. 10: test set

G. Data augmentation

Although the experimental data file is large, the amount of data is small, so we believe that data augmentation is necessary to increase the size of the training data set. By performing data augmentation, we create new samples that are similar to, but slightly altered from the original samples. This helps improve the generalization and robustness of the model during training and prevents overfitting.

Based on the above purpose, in order to increase the diversity of the data, and reduce the use of memory, we further improve the training data after creating a new image heap. The training data set will capture within the random patches of the masked area and outside of the validation area, these patches use a square box of 128*128 random boxes to build a training set, and the final shape of the training set becomes (32, 16, 128, 128). The verification set is changed to (32, 1, 128, 128) in the same way. The verification set is also generated by random patches, but it is in the red box. Visualized training and validation sets are shown in Fig.11 and Fig.12

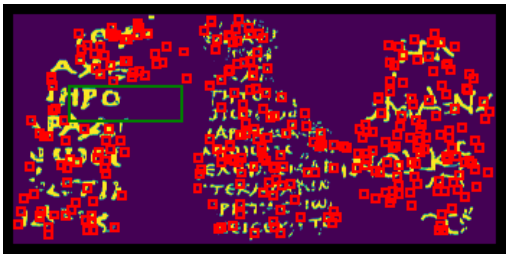

Fig. 11: random patches of train

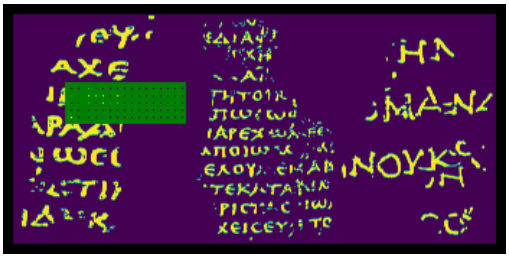

Fig. 12: random patches of test

We have also done other data enhancement work, such as Normalization, Flip. A partial plot is shown in Fig.13

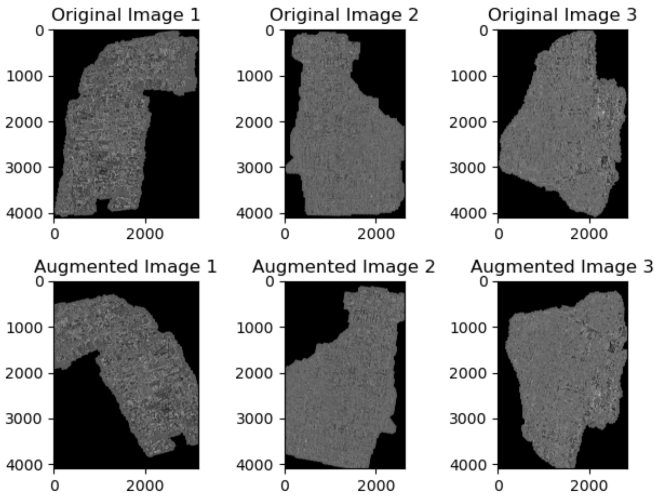

Fig. 13: Image Augmentation

III. Methodology

We will introduce 5 state-of-the-art models to handle this challenge.

A. FCN

Fully Convolutional Network (FCN) is a method for image segmentation proposed by Long et al. [2]. It consists of three main components:

- CNN Encoder: The CNN encoder initially maps the original image into a highly abstract feature space with reduced height and width. This process can be described by the following equation:

$$f_i = \sigma(W_i * f_{i-1} + b_i)$$

  where $\boldsymbol{f}_i$ is the feature map of the i-th layer, $\boldsymbol{\sigma}$ is the activation function, and W_i and b_i are the weights and bias of the i-th layer respectively.
- Up-sampling Part: After the transformation, an up-sampling layer using an interpolation method maps these features back to the original dimensions of height and width. The interpolation operation can be represented as:

$$f_i' = \text{interpolate}(f_i, \text{method})$$

  where f_i' is the up-sampled feature map and method denotes the interpolation method used.
- 1x1 Convolution Layer: Subsequently, a 1x1 convolution is executed for pixel prediction. This step can be represented as:

$$p_i = W_i'' * f_i' + b_i''$$

  where $\boldsymbol{p_i}$ is the pixel-level prediction and W_i' and b_i'' are the weights and bias of the 1x1 convolution respectively.

A visual overview of this architecture can be seen in Figure 14 [2].

In a more detailed perspective, each block within the architecture generally consists of several CNN layers, accompanied by Batch Normalization, ReLU activation, skip connections, and down-sampling techniques. Traditionally, MaxPooling is employed as the down-sampling layer, but in our approach, we adopt the architecture of ResNet [7], using a 1x1 convolution with a stride of 2 as the down-sampling layer.

The final results are achieved by aggregating features from various layers, each with different up-sampling rates, as illustrated in Fig 14 [2]. These prediction maps, corresponding to different up-sample rates, are then either added or concatenated. This process is followed by a 1x1 convolution and a sigmoid activation function. In the pursuit of more efficient computation, we have chosen to add these elements together, rather than concatenating them.

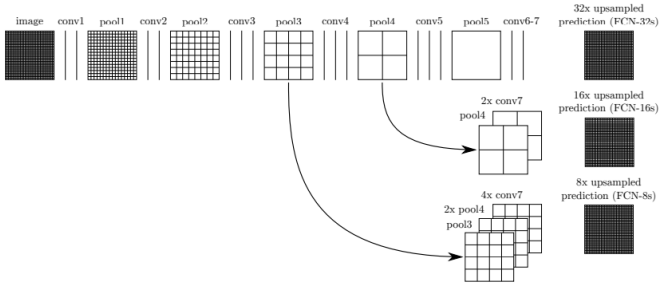

Fig. 14: FCN Architecture [2]

B. U-Net

For this ink detection challenge, the desired output is a binary image where each pixel is assigned a label to indicate whether it is an ink blot or not, which is referred to as a semantic segmentation task. Hence, our group intends to implement ink detection based on the U-Net structure proposed by Olaf et al [4]. In addition, we modified parts of the model to make it more suitable for our task. The original network architecture is illustrated in Figure 15 [4]

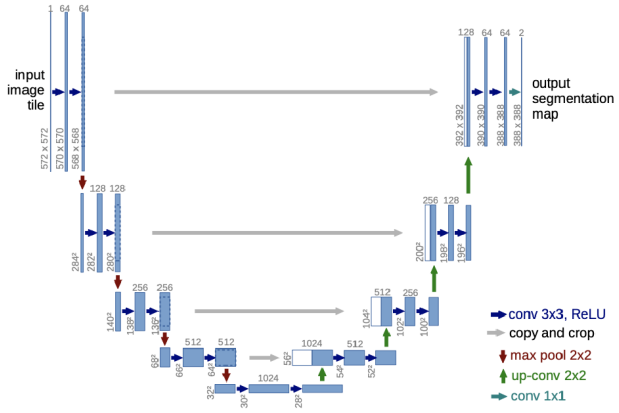

Fig. 15: U-Net Architecture [4]

In summary, the original architecture contains two symmetrical structures. The encoder, located on the left half of the network, consists of a series of stacked convolutional and maximal pooling layers to capture high-level features in the image and gradually reduce the spatial resolution of the image. The decoder, located in the symmetric position, progressively recovers the achieved feature mapping into the desired output size image through a series of transposed convolution and upsampling operations.

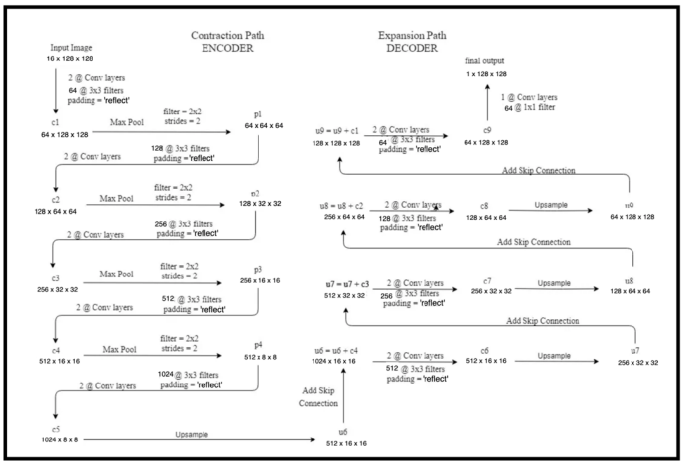

Fig. 16: Modified U-Net Architecture

In order to achieve the requirements of the task there are two points that could be improved. Firstly, since Olaf et al. [4] did not require the same size for the input image and the output image, the original structure does not have padding during the convolution operation. However, this task requires the input and output images to be of the same size, so padding is performed during the convolution operation. Besides, to ensure the preservation of feature information and to avoid introducing any interfering information, mirror padding is chosen. Secondly, batch normalization is introduced to avoid "internal covariate shift" [8] [9]. After the above modification, the double convolutional layers become 2@(Convolution + BN + ReLU). The improved structure is shown in Fig. 16.

C. Deeplabv3

Deeplabv3 is another SOTA deep learning network in segmentation field, compared to Unet. It is the latest vision of the original DeepLab framework, which is well-known for its ability of encoding multi-scale contextual information. [3]

There are three main characters of Deeplabv3, which are Dilated Convolutions, Multi-Path Aggregation and ASPP (Atrous Spatial Pyramid Pooling).

$$y[i] = \sum_{k} x[i + r.k]w[k] \tag{1}$$

In 1, y is the output feature map, w is the weights of convolution filter, x is the input feature map and r is atrous rate. The output feature map is the aggregation of the i_{th} pixel and its nearby pixels according to its dilated rate.

- Dilated Convolution is also called Atrous convolution, which is very different from the traditional convolutional operation. It has gaps or holes in the kernel, so that a small kernel can extract more features. Instead of making use of the nearby pixels, the dilated Convolution can collect pixels in a certain range according to the dilation rate.
- Multi-Path Aggregation is similar to the architecture of Unet. Each upsampling stage has an output branch to be connected with the same shape of the downsampling one, which means that the decoder will get more feature details from encoder and will not lose too much information during the feature extraction process.
- Depthwise separable convolution is a combination of a normal convolution with a pointwise convolution (1×1 convolution). Therefore, it can implement a spatial convolution to each channel.
- The main idea of ASPP is using dilated convolution with different dilated rates to extract features and gather them together. It is achieved by Spatial pyramid pooling. So, it can collect both local and global information.

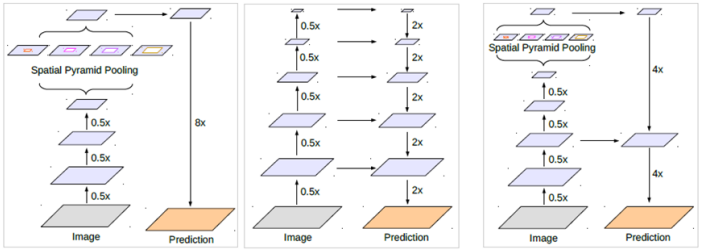

Fig. 17: Deeplabv3 Architecture [3]

D. Unet++

UNet Plus Plus is a variant of the original Unet architecture, and it makes a huge improvement in the following partsi.e., the decoder and skip connections. In the decoder, Unet++ implements a more complex feature extractor and uses a series of nested, dense skip pathways to collect details in the feature map. Also, the redesigned skip pathways deepen the connections between encoder and decoder, aggregate and concatenate features from multiple layers. Another advantage is the function of deep supervision, which can meet the demands of both outputting the final result or a single segmentation branch. [5]

$$x^{i,j} = \begin{cases} \eta(x^{i-1,j}) & j = 0 \\ \eta([x^{i,k}]_{k=0}^{j=1}, \quad \mu(x^{i+1,j-1})]) & j > 0 \end{cases} \tag{2}$$

In equation 18 $\eta(.)$, $\mu(.)$ and [] stand for convolution, up-sampling and concatenation layers respectively. $x^{i,j}$ is the output of i^{th} up-sampling layer and j^{th} dense convolution layer in the skip pathway. In the condition of j=0, the network only receives input from the previous layer. When j¿1, the network will receive j+1 inputs from both previous layers in the skip pathways and the last input, which is the output from up-sampling operation.

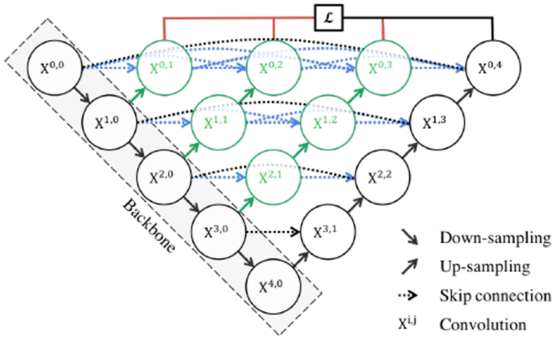

Fig. 18: Unet++ Architecture [5]

- Blacks are the original encoder and decoder parts of U-Net with some single skip connections.
- Greens refer to dense convolution blocks, which perform better than traditional convolutions.
- Blues are skip pathways, which build more connections between the dense convolution blocks and encoder and decoder to avoid feature loss.
- Reds indicate deep supervision, providing feature maps of different shapes from the network.
- As can be seen, all Red, green, and blue components are the improvements of Unet++.

E. Attention U-Net

Recalling the standard or our improved U-Net architecture, we could observe that it is not perfect in regenerating spatial information during the up-sampling process. This is due to the utilization of a skip connection approach that combines the spatial information from the down-sampled and up-sampled paths, as illustrated in Fig. 16 where u6, u7, u8, and u9 are generated. Due to the limited feature representation capability of the initial layer, this behavior leads to the incorporation of numerous redundant low-level feature extractions. Therefore, the attention mechanism is introduced to optimize the improved U-Net [10]. Ozan Oktay et al. proposed the integration of a soft attention gate into U-Net, which helps to reduce computational resources used on irrelevant activation and enhances the network's generalization capabilities [6]. Our team would reproduce the neural network proposed by Ozan Oktay et al. and validate our hypothesis with the ink detection task.

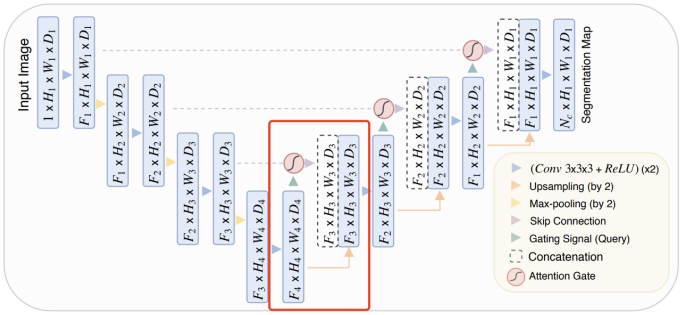

Fig. 19: Attention U-Net Architecture [6]

The architecture of attention U-Net is presented in Figure 19. The red box is where the attention gate is implemented. The structure of the attention gate is shown in Fig.20 [6].

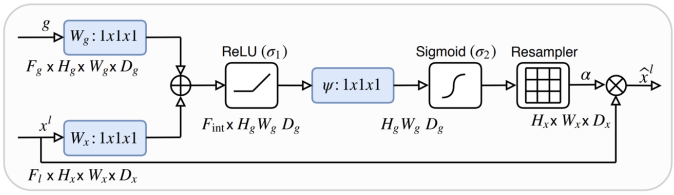

Fig. 20: Attention Gate Architecture [6]

We would refer to the input of the skip connection as x and the input from the previous block as g. Then the whole module can be represented by the following equations:

$$x_{attention} = Attention(g, x) \tag{3}$$

$$x_{upsampled} = UpSample(g, scale = 2) \tag{4}$$

$$output = ConvBlock(concatenate(x_{attention}, g_{upsampled}) \tag{5}$$

To explain further, $Attention()$ is the attention gate, $UpSample()$ is a simple up-sampling module that uses nearest neighbor interpolation, and $ConvBlock()$ is just a sequence of $2@(Convolution + BN + ReLU)$ blocks.

The whole process is referred to Fig.20:

- x and g are both fed into a 1x1 convolution that changes them to the same number of channels without changing the size.
- After the upsampling operation (with the same size), they are accumulated and passed through the ReLU.
- Through another 1x1 convolution and a sigmoid, an importance score from 0 to 1 is obtained and assigned to each part of the feature map.

- This attention graph is then multiplied by the input of the skip connection to produce the final output of this attention block.

According to Ozan Oktay et al., the mentioned approach is universal and modular without applying external object localization models and could easily cope with semantic segmentation of fine-grained objects, which is very suitable for this competition [6]. Further explanations will be presented in the experimental section.

F. Vision Transformers

We also tried to use VIT (Vision Transformer) to explore the possibility of this new model on 3D images; The structure of vision transformers is Fig.21 [11]

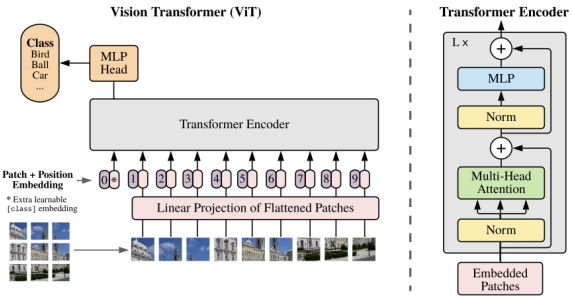

Fig. 21: Vision Transformer

In summary, the VIT model consists of three modules:

- Embedding layer [11]
  For the traditional Transformer module, the input it needs is a token (vector) sequence, that is, a two-dimensional matrix [num token, token dim] In computer vision training, generally, the data format of the picture is [H, W, C], which is a three-dimensional matrix. So this obviously does not conform to the Transformer calculation method. Therefore, in the Embedding layer, the data needs to be transformed first. As shown in the figure above, first divide a picture into a bunch of patches according to the given size. Then map each Patch to a one-dimensional vector through a linear map [11]. Taking ViT-B/16 as an example [11], each Patch data shape is [16,16,3] and a vector with a length of 768 is obtained through mapping[16, 16, 3] turn to [768] [11]
- Transformer Encoder
  The Transformer Encoder actually repeats L times to stack the Encoder Block, which mainly consists of the following parts: Layer Norm, this Normalization method is mainly proposed for the NLP field. In the vision field, this is the Norm processing of each token. [11] Multi-Head Attention, the Dropout layer was used in the original paper code. [11] MLP Block consists of full connection + GELU activation function + Dropout. [11]
- MLP Head (final layer structure for classification)
  The output shape and the input shape after passing the Transformer Encoder above remain unchanged. The original paper of MLP Head mentioned that it is composed of Linear + tanh activation function + Linear when training mageNet21K.

By analyzing the principle and model structure of Vision Transformer. Moreover, judging from the results given by the experiment in the original paper, although SOTA has been achieved, more data sets are required than CNN [11]. After designing the Dataset, this project actually only trained a 3D picture with a height of 65. Considering that ViT is generally difficult to train, it requires a large amount of video memory and a long training time [12]. If the increase in performance comes at the cost of a huge sacrifice in speed, we think the model is much less meaningful, which is not the focus of this project. For ViT, reducing input tokens can be used for reasoning acceleration. Based on this experiment, its huge size, such as 1st volume of the *train (6330, 8181), is obviously not suitable for processing with VIT. Based on this, we only analyzed the VIT model and did not conduct experiments.

IV. Experiment

A. Optimizer

1) Stochastic Grident Descent: SGD is a method using a random batch to calculate the gradient decent.

$$g_t = \frac{\sigma loss}{\sigma W_t} \tag{6}$$

$$W_{t+1} = W_t - lr * \frac{\sigma loss}{\sigma W_t} \tag{7}$$

2) RMSprop: RMSprop is an improvement of SGD adding a momentum

$$V_t = \beta * V_{t-1} + (1-\beta) * g_t^2 \tag{8}$$

$$W_{t+1} = W_t - lr * g_t / (\sqrt{\beta * V_{t-1} + (1-\beta) * g_t^2}) \tag{9}$$

3) Adam: Adam can be considered as a combination of RMSprop and Momentum.

$$m_t = \frac{m_t}{1-\beta 1^t} \tag{10}$$

$$V_t = \frac{V_t}{1-\beta 2^t} \tag{11}$$

$$W_{t+1} = W_t - lr * \frac{m_t}{1-\beta 1^t} / \sqrt{\frac{V_t}{1-\beta 2^t}} \tag{12}$$

We have tried these three optimizers using Unet++ for 40 epochs.

learning rate = 1e-4

SDG: momentum=0.9, eight_decay=0.0005

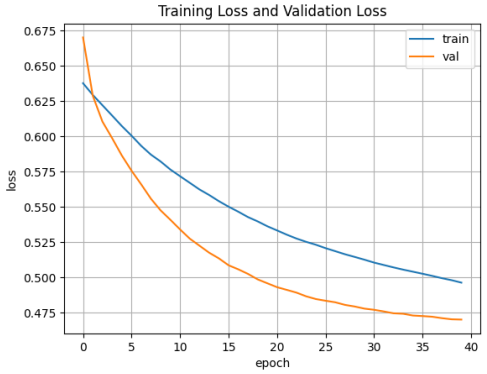

Fig. 22: SGD loss

RMSpropalpha=0.99, eps=1e-8, weight_decay=0, momentum=0

Fig. 23: RMSprop loss

Adam betas=(0.9, 0.999), eps=1e-08

We can see from figure 22,23,24 that Adam performns the best because the training loss is declining smoothly and reach the smallest value in the end. The loss curve of RMSprop is similar to Adam, but it takes more time to achieve the optimal point and SGD is the worst. So we choose Adam as our main optimizer method.

Fig. 24: Adam loss

B. Binary Cross Entropy Loss

In this task, we use BCE Loss as the main loss function because it is a segmentation challenge, we only need to identify the mask to be 0 or 1. [13]

$$L_N = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log(\sigma(x_n)) + (1-y_n).\log(1-\sigma(x_n)] \tag{13}$$

In 13, N refers to the total number of train data.

C. Evaluation metrics

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

TABLE IV: Confusion Matrix

1) Confusion Matrix:

2) Recall:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negetive} \tag{14}$$

Recall is a metric to calculate the accuracy of the model prediction according to the ground truth label.

3) Precision:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{15}$$

Precision is a metric to calculate the accuracy of the model prediction to be true .

4) $F_{0.5}Score$*:*

$$F - Score = (1+\alpha^2)\frac{P*R}{\alpha^2 * P + R} \tag{16}$$

When α = 1, i.e., F1-Score, which means Precision has the same importance as Recall.

When $\alpha = 0.5$, i.e., F0.5-Score, which means Precision is more important than Recall.

When $\alpha = 2$, i.e., F2-Score, which means Precision is less important than Recall.

In this challenge, we use F0.5-Score as one of our final evaluation metrics.

D. FCN

In the FCN model, 30 epochs were trained, the batch size was set to 16, the learning rate was set to 1e-3, and the training was continued until the model fully converged. Figure 25 shows the training results for both the training and validation sets. Among all the epochs, the 22nd epoch performed the best, when the validation loss reached 0.9001 and the validation accuracy was 0.7777. All the details and metrics of each phase are precisely documented in our jupyter-notebook submission and are presented in TableV.

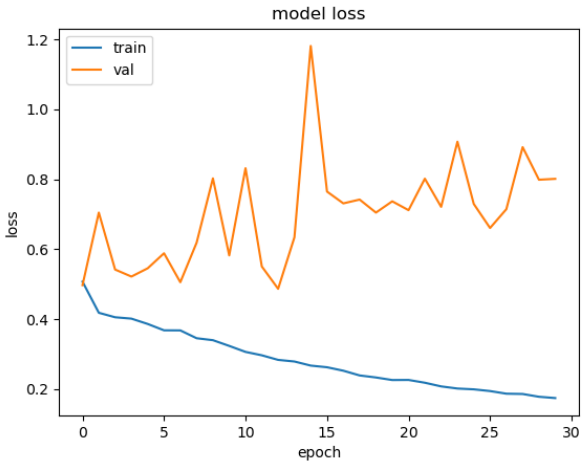

Fig. 25: FCN Training loss vs Val loss

After the completion of the training and validation phases, we further applied the model to the test set to predict the ink distribution for different samples. Figures 26 and 27 depicts the corresponding results for test set A and test set B in detail, visualizing the effectiveness and robustness of the model. However, due to the specificity of the test set, it lacks real labels. To address this challenge, we use the masks of training set 1 as inputs to visualize the results of model learning through Figure 28.

E. U-Net

The U-Net model was trained for 30 epochs with a batch size of 32 and a learning rate of 1e-4. With the loss function set as BCE (13) and the optimizer selected

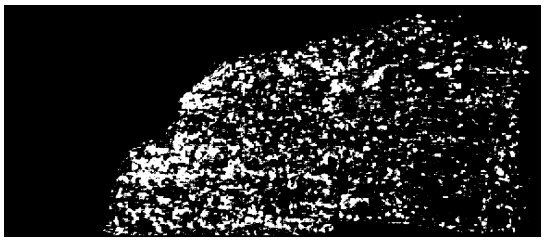

Fig. 26: FCN Prediction on Test A

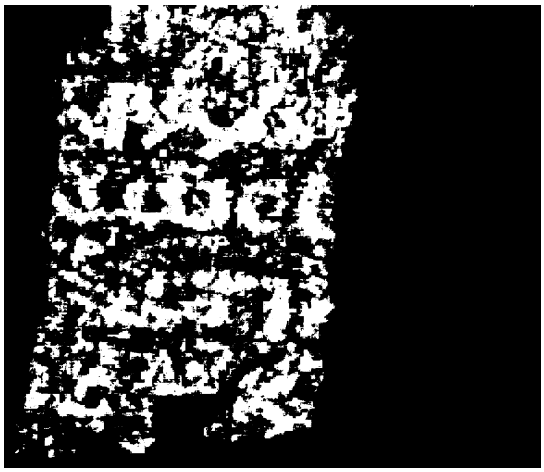

Fig. 27: FCN Prediction on Test B

as Adam, The results are produced on the training and validation sets as shown in Fig. 29. The model is saved with the best-performing under both the training and validation sets, which occurs at the 27th epoch where validation loss is 0.409 and validation accuracy is 0.833. Specific experimental results can be viewed from the submitted jupyter-notebook.

Then, We conducted predictions on the distribution of ink in the test set, and the corresponding results for test set A and test set B can be observed in Fig. 30 and Fig. 31, respectively.

As there are no truth labels available for the test set, the mask of the training set 1 is utilized as input to visualize the model's learning outcomes, shown in Fig. 32. Eventually, the metrics are measured in the validation set which could be viewed in Table V.

F. Attention U-Net

The training process for Attention U-Net is the same as that mentioned above for U-Net IV-E. The best-performing model emerges in epoch 28, with the validation loss and accuracy of 0.417 and 0.837, respectively. The loss of the training and validation sets can be viewed in Fig. 33.

Then, We conducted predictions on the distribution of ink in the test set, and the corresponding results for test

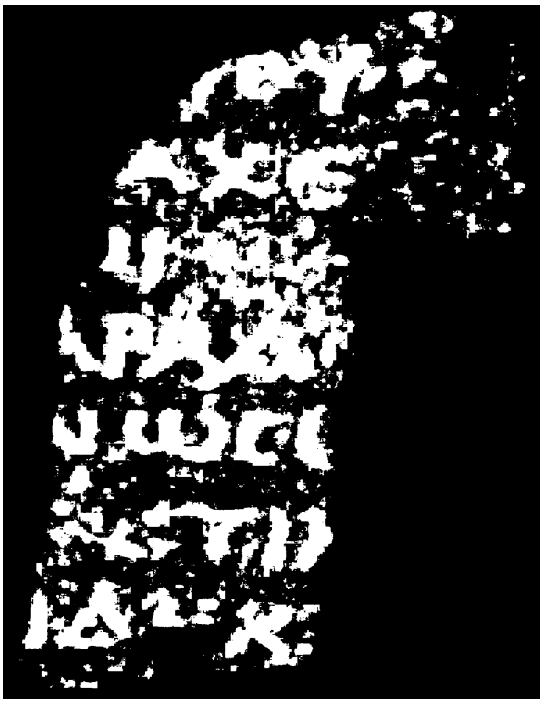

Fig. 28: FCN Prediction on Train 1

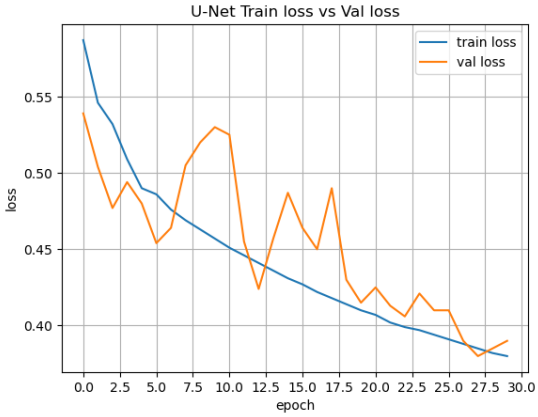

Fig. 29: U-Net Training loss vs Val loss

set A and test set B can be observed in Fig. 34 and Fig. 35, respectively.

For training set 1, the ink stain predictions can be viewed in Fig. 36. The metrics obtained from the validation set can be viewed in table V.

G. Unet++

Here we define σ representing our basic activate function (Relu).

We trained Unet++ for 20 epochs with learning rate 1e-4 and Batch size 64 to its convergence.

For Adam optimizer, we choose the best parameter combinations - betas=(0.95, 0.9999), eps=1e-09.

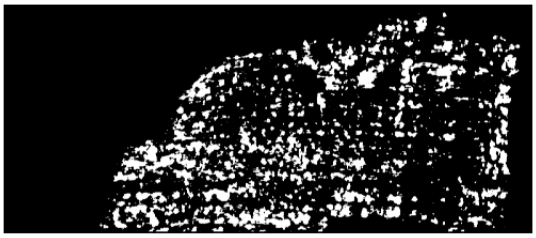

Fig. 30: U-Net Prediction on Test A

Fig. 31: U-Net Prediction on Test B

It uses ContinusParalleConv as its base downsampling operation, which contains a sequence of a BatchNormalization + σ, a 2D convolution + σ and a 2D convolution. After the input enters the encoder, it first generates feature maps in 64 channels and gradually doubles the channel size to 1024 in the end to get the hidden feature state. The decoder upsamples the features in the same way, using the ConvTranspose2d function provided by Pytorch to double the feature map and add the feature of the same shape from the encoder to avoid feature loss. There are totally five stages implemented in our model.

In figure 37, it can seen that the training loss is falling during the training process, while the validation loss is fluctuated all the time.

In figure 40, it shows the final mask predicted by Unet++ for train dataset 1.

H. Deeplabv3

We trained Deeplabv3 for 40 epochs with learning rate 1e-4 and Batch size 64 to its convergence.

We use this function through torchvision.models.segmentation function. It uses resnet101 as its basic CNN operation. For the backbone, it has four layers, containing 3, 4, 23, and 3 bottlenecks. Each bottlenecks consists of a normal convolution,

Fig. 32: U-Net Prediction on Train 1

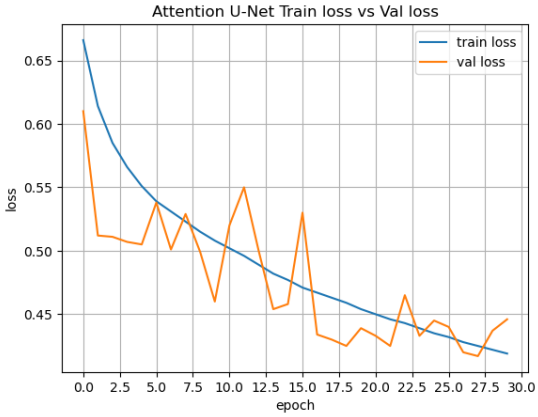

Fig. 33: Attention U-Net Training loss vs Val loss

a BatchNormalization, a diated convolution and a BatchNormalization with σ. For the classifier, it is a ASPP module, which contains 3 ASPPConv and 1 ASPPPooling. ASPPConv is a combination of a dilated convolution, a BatchNorm2d and σ; ASPPPooling includes an AdaptiveAvgPool, a normal convolution, a BatchNorm2d and σ to produce the output.

In figure 41, we can see that although the validation loss is fluctuated, it still declines in the end along with the training loss.

In figure 44, it shows the final mask predicted by Deeplab3 for train dataset 1.

Fig. 34: Attention U-Net prediction on Test A

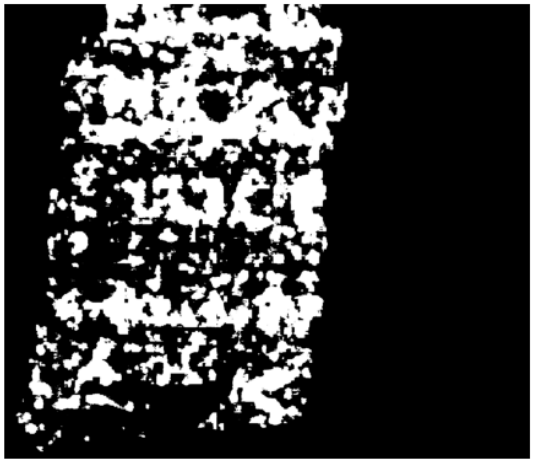

Fig. 35: Attention U-Net prediction on Test B

V. Discussion

The purpose of this section is to discuss the evaluation metrics we used. In addition, the differences between the various models in the context of the architecture and the experimental evaluation results are analyzed and discussed.

Firstly, we would like to explain why these evaluation indicators are chosen. The selection of pertinent indicators is contingent upon the inherent essence of the problem under consideration, the attributes inherent in the dataset, and the precise aims and objectives of the given task. For this task, $F_{0.5}Score$ (16) recall (14), and precision (15) are selected because those potential ink stains are more concerned to be correctly classified, considering that the carbonized paper is often mistaken for ink stains. It is emphasized that $F_{0.5}$ is chosen instead of F_1 because $F_{0.5}$ score weight precision is higher than the recall, which improves the ability to form coherent characters from the detected ink regions. Recall (14): it is expressed as the ratio of true positive predictions to the sum of true positive predictions and false negative predictions. Precision (15): it indicates the accuracy of the model in correctly identifying positive cases among the predicted positive instances. More comparisons of model architectures are discussed below.

TABLE V: Mean training time and metrics on different methods

model	number of parameter	training time	recall	precision	$F_{0.5}Score$
FCN	2,905,921	403 s	0.3741	0.3943	0.3901
U-Net	36,970,689	530 s	0.7711	0.6004	0.6282
Unet++	47,198,212	1188.9s	0.7536	0.6271	0.6043
Deeplabv3	58,817,153	900.6s	0.6873	0.5942	0.5632
Attention U-Net	34,886,061	612 s	0.7890	0.6257	0.6352

Fig. 36: Attention U-Net prediction on Train 1

Fig. 37: Unet++ Loss

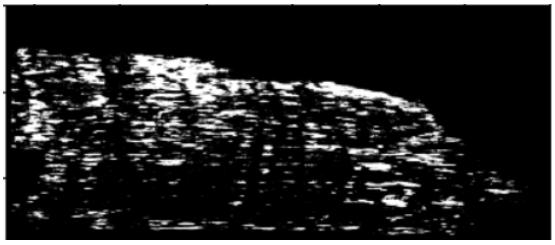

Fig. 38: Unet++ Prediction on Test A

Fig. 39: Unet++ Prediction on Test B

The utilization of FCN in semantic segmentation has been revolutionary and pioneering since its inception, providing a strong groundwork for a plethora of subsequently optimized networks [2]. FCN is a fully convolutional network that operates from end-to-end, performing pixel-to-pixel segmentation by replacing the conventional fully connected layer with a fully convolutional layer. It extracts image features through downsampling and later reconstructs, and integrates the feature information with upsampling to accomplish the segmentation task. U-Net is based on the original architecture of FCN with the addition of skip-connection in the encoder and decoder. It could provide richer feature information, which makes the segmentation result more accurate, especially in small targets or tasks that require finer segmentation results [4]. This is also well supported by the comparison of the experimental results in Fig. And

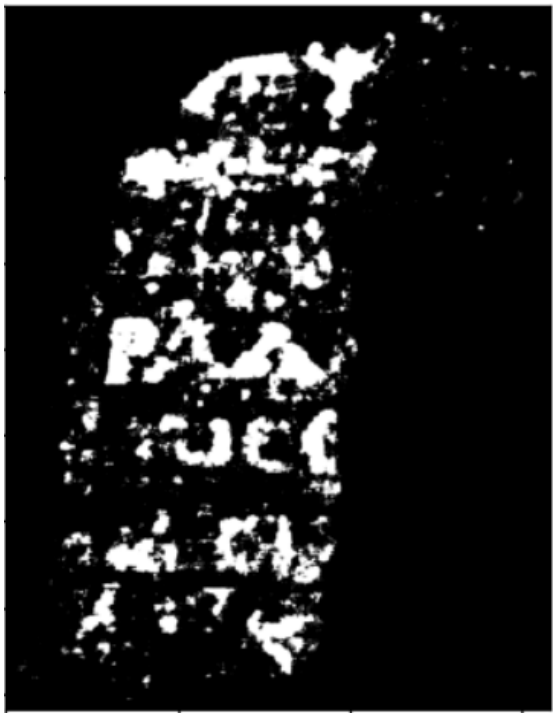

Fig. 40: Unet++ Prediction on Train 1

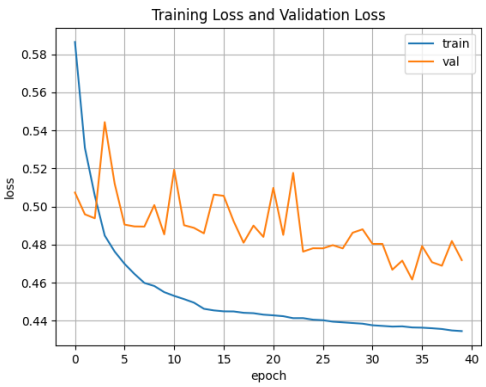

Fig. 41: Deeplabv3 Loss

Fig. 32, and the various evaluation indicators.

For U-Net and attention U-Net, as mentioned in part III-E, the introduction of attention gates improves the skip-connection structure of U-Net so that low-level information is filtered out [14]. This is also supported by the experimental results, view Fig. 32and Fig. 36 and the metrics table V. Despite the marginal improvement, the results confirmed the efficacy of the attention gate in the model.

As a variant of Unet, Unet++ adds more skip pathways to capture features and do more convolution steps to acquire larger precision. In figure V, it is 2% higher than UNet and seems to be the best one among the five models. Comparing figure 32 with figure 40, Unet++

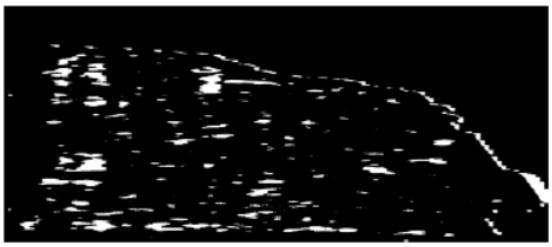

Fig. 42: Deeplabv3 Prediction on Test A

Fig. 43: Deeplabv3 Prediction on Test B

seems to be able to capture more details both locally and globally because the ink edge is more clear. But compared to figure 36, the edge of the ink label is not clear enough, so we guess it can be improved by adding an Attention Unit into every Bottleneck in order to enhance the definition.

As the latest version of Deeplab, Deeplabv3 does not perform well in this challenge. We think the main problem is that this model does not have enough concatenating layers in contrast of Unet. Therefore, it loses too much feature details of the ink label and the multi-view operations may not work to handle the handwriting task. So more experiments may be launched to add connections during the upsampling and downsamping phases.

In the end, we conclude that the attention unit, layer connections and dilated convolution are useful methods in this challenge, while ASPP may not work to some extend.

VI. Conclusion

We demonstrate the performance of FCN, U-Net, Deeplabv3, U-Net++, and attention U-Net deep learning models on the EduceLab-Scrolls dataset. The trained model is able to detect ink stains in Herculaneum papyri. We use a variety of metrics to evaluate the deep learning

Fig. 44: Deeplabv3 Prediction on Train 1

model. The data comparison confirms the good accuracy of our model. We believe that this deep learning is the optimal way to read the complete Herculaneum scroll using current imaging techniques and computational methods.

In addition, through experiments we can also learn from FCN to Attention U-net from as the complexity of the model increases the model predicts better. The introduction of the Attention Gate improves the hopping structure of U-Net, thus filtering out the low-level information. This allows the model to focus more on important information. Thus improving the accuracy.

Of course, there is still a big gap between the deep learning models we trained and the ultimate goal of reading the Herculaneum scrolls in their entirety. But we believe that over time more deep learning frameworks suitable for computer vision emerge. Recovering the complete text of the Herculaneum papyrus will be realized.

Therefore, among these five models, we consider AttenUnet as the best model for this task due to its tremendous attention system. However, it can still be improved in the following ways:

- more skip connections: It seems possible to add dense convolution layers into AttenUnet, which is same as adding Attention Unit to the backbone of Unet++ because the Attention Unit makes a big difference in the experiemnt.
- dilated convolution: In [15], dilated convolution can extract features better than the traditional ones. So it is likely to replace some convolution layers both in the encoder and decoder to improve the performance.
- ASPP: Spatial pyrimid is worth referencing in this challenge although the Deeplabv3 does not perform well. We can add some ASPPConv combined with traditional and dilated convolution layers to form a mixed feature extractor. Also, ASPPPooling can be used during the downsampling phase .

Apart from the model selection, more helpful data augmentation and training methods may enhance the final scores.

REFERENCES

[1] S. Parsons, C. S. Parker, C. Chapman, M. Hayashida, and W. B. Seales, "Educelab-scrolls: Verifiable recovery of text from herculaneum papyri using x-ray ct," in *arXiv.org*. arXiv.org, 2023, p. 2304.02084.

[2] 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*, 2015, pp. 3431–3440.

[3] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[5] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*. Springer, 2018, pp. 3–11.

[6] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz *et al.*, "Attention u-net: Learning where to look for the pancreas," *arXiv preprint arXiv:1804.03999*, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[9] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[10] E. I. Knudsen, "Fundamental components of attention," *Annu. Rev. Neurosci.*, vol. 30, pp. 57–78, 2007.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[12] Z. Pan, P. Chen, H. He, J. Liu, J. Cai, and B. Zhuang, "Mesa: A memory-saving training framework for transformers," *arXiv preprint arXiv:2111.11124*, 2021.
[13] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," *Int. J. Adv. Trends Comput. Sci. Eng*, vol. 9, no. 10, 2020.
[14] H. Pashler, J. C. Johnston, and E. Ruthruff, "Attention and performance," *Annual review of psychology*, vol. 52, no. 1, pp. 629–651, 2001.
[15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.