10 (2) 2024 160-174 ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
Register: *Jurnal Ilmiah Teknologi Sistem Informasi*

Contents lists available at www.journal.unipdu.ac.id

Register

Journal Page is available to www.journal.unipdu.ac.id/index.php/register

Research article

Deep Learning-Based Inpainting for Reconstructing Severely Damaged Handwritten Javanese Characters

Fitri Damayanti [a], *Eko Mulyanto Yuniarno* [b*], *Yoyon Kusnendar Suprapto* [c]

[a,b,c] *Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo, Surabaya, 60111, Indonesia*
[b,c] *Department of Computer Engineering, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo, Surabaya, 60111, Indonesia*
email: [a] 07111960010017@student.its.ac.id, [b] ekomulyanto@ee.its.ac.id, [c*] yoyonsuprapto@ee.its.ac.id
* *Correspondence*

ARTICLE INFO

Article history:
Received August 12th, 2024
Revised September 13th, 2024
Accepted October 12th, 2024
Available online December 31th, 2024

Keywords:
Deep Learning
Extensive Damage Areas
Handwritten Javanese Character
Inpainting
Reconstruction

Please cite this article in IEEE style as:
F. Damayanti, E. M. Yuniarno, and Y. K. Suprapto, " Deep Learning-Based Inpainting for Reconstructing Severely Damaged Handwritten Javanese Characters," *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 10, no. 2, pp. 160-174, 2024.

ABSTRACT

Long-term storage at museums can damage ancient Javanese manuscripts; for instance, temperature changes and other factors may cause parts of the script to disappear. The Javanese script exhibits similarities among its letters, making the reconstruction process challenging, particularly when dealing with severe damage to the script's characteristic areas. To address this issue, we conducted a character painting technique that utilizes deep learning architecture, specifically the convolutional autoencoder, partial convolutional neural network, UNet, and ResUNet. The dataset contains 12,000 handwritten Javanese characters. We evaluated the restoration of missing characters using SSIM and PSNR metrics. The ResUNet achieves the best performance compared to other methods, with an SSIM value of 0.9319 and a PSNR value of 18.9507 dB. According to this study, the ResUNet models can reconstruct Javanese manuscripts with strong performance, offering an alternative solution to ensure the preservation and accessibility of these valuable historical documents for future generations.

1. Introduction

The Javanese script, as one of the traditional scripts of Indonesia, plays an essential role in Javanese culture and history. The preservation of this script is crucial not only for understanding historical literature and documents but also for maintaining cultural identity. The Javanese script (initially called *Hanacaraka*, *Carakan*, and *Dentawyanjana*) is one of the traditional Indonesian scripts developed in the Java island. It is a descendant of *Brahmi* and is written in *Abugida*, which consists of 20 basic letters [1]. The basic Javanese or Nglegana script includes 20 characters, as shown in Fig. 1, comprising *Ha, Na, Ca, Ra, Ka, Da, Ta, Sa, Wa, La, Pa, Dha, Ja, Ya, Nya, Ma, Ga, Ba, Tha, Nga*. The Javanese script contains similarities between many of its letters. These similarities often involve basic shapes with minor variations that distinguish them. For example, some letters feature additional curves or lines that set them apart. It can be seen that the characters "*ha*," "*ka*," and "*ta*" are similar and distinguish the middle part. The characters "*na*" and "*da*" are similar and distinguish the middle part. The characters "*ca*" and "*sa*" are similar in distinguishing the upper right part. The characters "*ca*" and "*wa*" are also similar; the difference is the lower middle. The characters "*wa*" and "*pa*" distinguish the upper right part similarly. Similarly, the characters "*tha*" and "*nga*" are also similar; the difference is the lower middle. These comparisons can be observed in Table 1.

There is no publicly available character dataset for the Javanese script, either printed or handwritten. The availability of large amounts of image data is essential for machine and deep learning learning. Therefore, creating a dataset of Javanese scripts, both printed and handwritten, is a significant

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

challenge. Research related to the Javanese script that has been carried out includes recognition/classification of the Javanese script[2],[3],[4],[5], and segmentation of the Javanese script [6]. However, research related to reconstructing the damage to the Javanese script is rarely conducted. A study on the reconstruction of the Javanese script was carried out by Himamunanto et al. in 2018 using the Text Block Identification method. The study used a dataset from an ancient Javanese document entitled "Hamong Tani." The number of datasets used was 452 primary Javanese characters that were damaged. With a small area of damage, several examples of datasets from the research conducted by Himamunanto et al. are shown in Fig. 2. The damage to the Javanese script is still visible to the naked eye. The resulting accuracy was 82.07%. The research was able to carry out restoration when the damaged area was small.

Fig. 1. Basic Javanese script or Nglegana script

Table 1. Similarities of Javanese script

Character	Script	Similarity
	*"Ha"*	In the middle of the script
	*"Ka"*	
	*"Ta"*	
	*"Na"*	In the middle of the script
	*"Da"*	
	*"Ca"*	At the top right
	*"Sa"*	
	*"Ca"*	In the lower center
	*"Wa"*	
	*"Wa"*	At the top right
	*"Pa"*	
	*"Tha"*	In the middle of the bottom
	*"Nga"*	

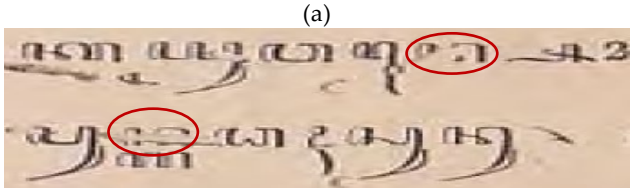
There is no publicly available character dataset for the Javanese script, either printed or handwritten. The availability of large amounts of image data is essential for machine and deep learning learning. Therefore, creating a dataset of Javanese scripts, both printed and handwritten, is a significant challenge. Research related to the Javanese script that has been carried out includes recognition/classification of the Javanese script[2],[3],[4],[5], and segmentation of the Javanese script [6]. However, research related to reconstructing the damage to the Javanese script is rarely conducted. A study on the reconstruction of the Javanese script was carried out by Himamunanto et al. in 2018 using the Text Block Identification method. The study used a dataset from an ancient Javanese document entitled "Hamong Tani." The number of datasets used was 452 primary Javanese characters that were damaged. With a small area of damage, several examples of datasets from the research conducted by Himamunanto et al. are shown in Fig. 2. The damage to the Javanese script is still visible to the naked eye. The resulting accuracy was 82.07%. The research was able to carry out restoration when the damaged area was small.

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

Fig. 2. Examples of Javanese script corruption

(a)



(b)

Fig. 3. Damage to ancient Javanese manuscripts: (a) and (b)

In reality, damage to the Javanese script in ancient manuscripts can be found in most areas of the script, as seen in Fig. 3(a). Additionally, the damage can also occur in areas that are characteristic or distinguishing features of the Javanese script, as shown in Fig. 3(b). A red circle marker indicates the damage to the Javanese script. Handwritten Javanese script has a style that is not standardized between individuals. Therefore, if there is damage to the handwritten Javanese script with a large area or damage to distinctive areas, archaeologists will need assistance reconstructing the script.

Fig. 4. Examples of handwritten Javanese script with extensive damage.

(a) (b) (c) (d) (e)

Fig. 5. Examples of handwritten Javanese script with damage to its distinctive features.

Fig. 4 shows several examples of handwritten Javanese script with large areas of damage. It can be seen that almost all characters are covered with masking, which marks the damaged areas. Fig. 5 shows several examples of handwritten Javanese script where the damage lies in characteristic areas of the script. Fig. 5(a) shows that if reconstruction is implemented, the result could be the characters "*ca*", "*wa*," or "*dha*." Fig. 5(b) shows that the reconstruction could result in the characters "*tha*" or "*nga*" Fig. 5(c) shows that the reconstruction could result in the characters "*na*" or "*da*". Fig. 5(d) shows that the reconstruction could result in the characters of "*wa*" or "*pa*." Moreover, Fig. 5(e) shows that the reconstruction could result in the character "*ca*" or "*sa*." Therefore, if Fig. 4 and Fig. 5 are reconstructed, it is likely to produce characters that are not from the original Javanese script.

Based on the problems described earlier, damage to the Javanese script with a large area or damage to characteristic areas of the script will result in incorrect reconstruction. Therefore, a method or approach is needed to overcome these problems so that the reconstruction results align with the original Javanese script (Ground Truth).

With the advancement of digital technology, especially in computer vision and machine learning, new techniques have been developed to address the problem of visual damage to documents. One of these techniques is image inpainting, which fills in missing or damaged parts of an image. Due to ongoing advancements in computer technology and image processing, image inpainting has emerged

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

as a significant area of study in the fields of computer vision and computer graphics[7]. Extensive research [8],[9] has been conducted on the issue of high-quality image inpainting, specifically focusing on generative semantic understanding. As a result, numerous traditional image inpainting algorithms based on deep learning have been developed.

This study uses deep learning-based character inpainting techniques to overcome the challenges posed by damage to large areas and the characteristics of handwritten Javanese script, as shown in Figures 4 and 5. The methods selected for this study include Convolutional Autoencoder, Partial Convolutional Neural Network (P.C.N.N.), UNet, and ResUNet, each chosen for its specific contribution to the field of image inpainting.

Lian et al. used a convolutional auto-encoder (C.A.E.) to restore images in the corrupted CIFAR-10 and CIFAR-100 datasets by learning abstract representations from unlabeled data to improve image classification. The advantage of this method is its ability to learn end-to-end and its highly efficient capture of relevant image features while maintaining spatial structure. Symmetrical skip connections in the C.A.E. help preserve image details during reconstruction.

Guilin Liu et al. used the P.C.N.N. method to repair images with irregular holes in the ImageNet, Places2, and CelebA-HQ datasets. This technique fills in the missing parts of the image with a more natural result without the need for additional post-processing. The advantage of this method is that convolution is performed only on valid pixels, thus reducing visual artifacts such as color differences or blurriness. In addition, this method can handle irregular holes and provide more realistic results than other approaches.

Ozturk et al. used a modified U-Net method to accelerate content-based image retrieval (CBIR) on the C.O.R.E.L. and I.R.M.A. datasets by generating a more concise binary hash code of the image without the need for label or class information[10]. The inpainting process removes 30% of the random pixels from the input image then uses U-Net to reconstruct the image. Features are obtained from the middle layer to generate the hash code. The advantage of this method is its ability to quickly generate robust features, even on unlabeled datasets, and provide competitive performance in image search.

The UNet Residual Attention method, employed by Hosen et al. is used to remove face masks and restore facial structure with fine details[11]. This method utilizes residual blocks to address the vanishing gradient problem and attention units to focus on essential areas of the image, thereby improving the model's efficiency and speed. The dataset used is CelebA, which consists of 200,000 images of mask-free faces. Each image is processed at a size of 256x256 and simulated as a masked face. The advantages of this method include not requiring an intermediate stage such as segmentation, faster predictions, and producing realistic, consistent results in recovering faces from masked images.

This research employs these methods to contribute to identifying the best deep-learning model for reconstructing handwritten Javanese scripts using inpainting techniques. The results are expected to aid in restoring the damaged handwritten Javanese script to a perfect and correct form of the original script.

2. Materials and Methods

2.1. Character Inpainting

Character inpainting primarily relies on neural network models and is constrained by character datasets, which are predominantly studied by Chinese researchers. The two objectives of this project are to improve the accuracy of Chinese character identification and to safeguard traditional cultural resources[12],[13],[14]. Yu Weng et al. (2019) reconstructed Chinese characters using the Conditional Generative Adversarial Network (CGAN) method[15]. In 2021, In-su Jo et al. employed the Variational Autoencoder with Classification (VAE-C) model to correct imperfect Chinese characters[13]. In the same year, Ge Song et al. [12], attempted to reconstruct damaged Chinese characters using a method based on attention and adversarial classification loss. Benpeng Su et al. (2022) achieved authentic reconstruction of ancient Yi letters by building a muti-stage restoration network combining shape restoration and texture restoration [16]. In 2022, Gongbo Sun et al. explored the reconstruction of Chinese letters using the RubbingGAN method. Rubbing GAN consists of one generator, denoted by G, and two discriminators, denoted as D1 and D2. The generator employs the U-Net architecture. Disdiscriminator1 uses the PatchGAN architecture, and Disdiscriminator2 utilizes the Auto Encoder Decoder architecture [17].

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

Research on other UNet-based character inpainting was conducted by [18],[19]. In 2023, several studies related to character inpainting were carried out by Chen Xing et al., who used the Context Encoder method [20]. In the same year, Long Zhao et al. conducted research using the UNet and Self Attention methods [19]. Long Zhao et al. also performed character inpainting research using the Branch Convolutional Channel Attention Module method [21].

2.2. Materials

This study utilized a dataset of handwritten Javanese scripts obtained from https://www.kaggle.com/datasets/hannanhunafa/javanese-script-aksara-jawa-augmented. The Javanese script is handwritten in several positions, namely upright, tilted left, and tilted to the right. The sizes vary, ranging from small to large. Examples of handwritten Javanese script datasets are shown in Fig. 6. The script used in this study is the Nglegana script (basic script), which consists of 20 letters. The dataset for this study is 12000 samples, divided into 8400 training data, 2400 validation data, and 1200 test data.

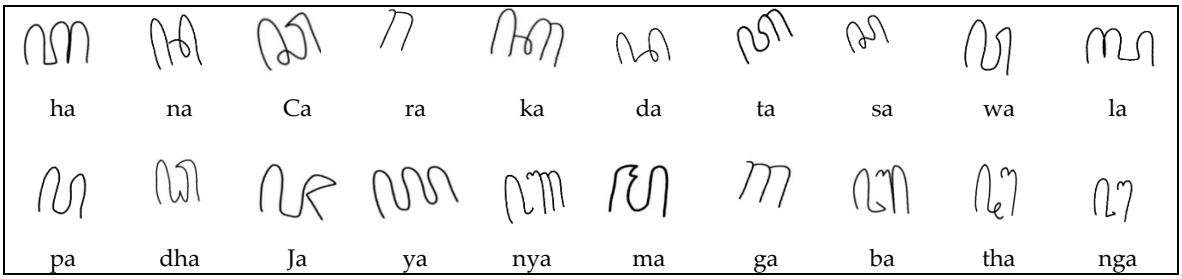


Fig. 6. Examples of handwritten Javanese script datasets

2.3. Methods

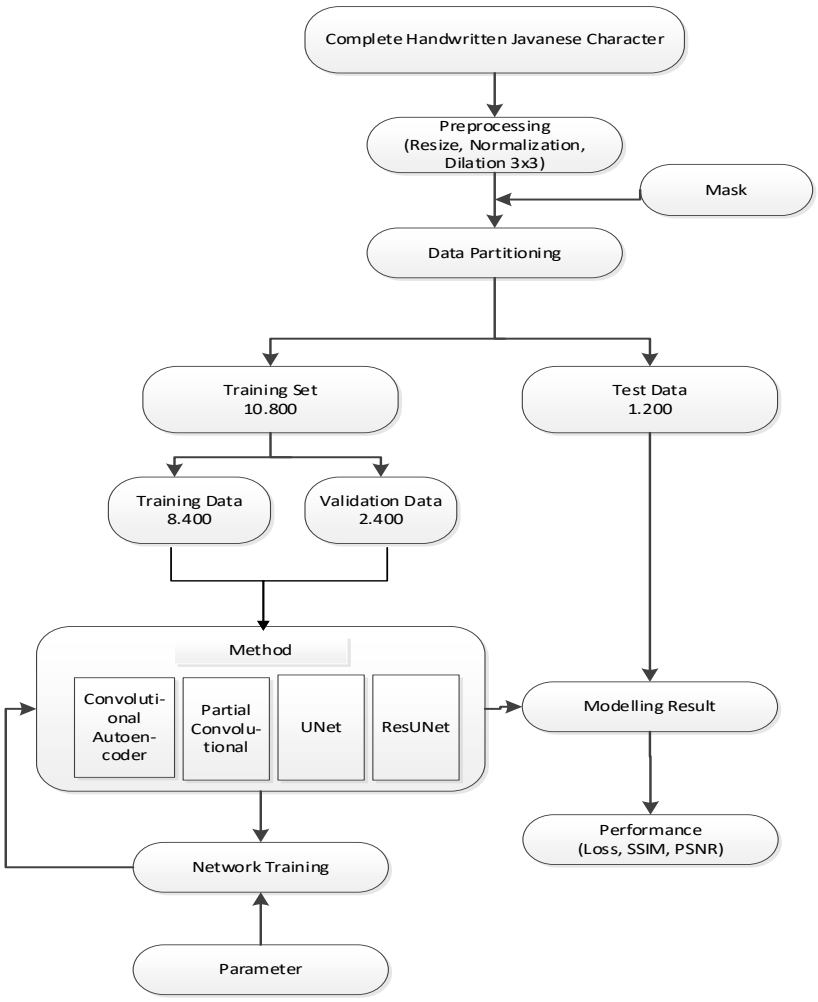


Fig. 7. Research Diagram

http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

The complete process for all the methods used in this study is illustrated in the block diagram shown in Fig. 7. The handwritten Javanese script is in good condition and undergoes preprocessing steps such as resizing, normalization, and 3x3 dilation. 3x3 kernel dilation is commonly used in image processing for several reasons : (a) Computing Efficiency: The small size of the kernel allows dilation operations to be performed quickly, reducing the computational load without sacrificing too much detail. This is crucial when processing large volumes of images or high-resolution images. (b) Proper Resolution: The 3x3 kernel is small enough to retain the important characteristic details of the Javanese script, which is often complex and intricate, yet large enough to make a significant difference in the visibility and clarity of the characters when analyzed by OCR algorithms.

The preprocessing results are combined with masks to partition the data into training, validation, and testing sets. The training process utilized four methods (Convolutional Autoencoder, Partial Convolutional Neural Network, UNet, and ResUNet). The parameters were adjusted and tested multiple times until the best model of each method was obtained. Subsequently, a trial was carried out using testing data, and performance was obtained based on loss values, SSIM, and PSNR.

2.4. Convolutional Autoencoder

The first convolutional inpainting approach pertains to the structure of CNNs with a symmetrical encoder-decoder architecture [22]. Although the structure is relatively simple, this method uses convolutional layers and max-pooling operations to exploit the feature extraction properties in the encoder's initial half. The convolutional autoencoder consists of two main parts, namely, the encoder and the decoder. The encoder is responsible for converting the input image into a lower-dimensional representation in the latent space. This part includes several convolution and max pooling layers. The decoder part is responsible for converting the latent representation back into an image with the same dimensions and structure as the original input. This part consists of several layers of upsampling and convolutional layers. Fig. 8. illustrates the Convolutional Autoencoder architecture used in this study.

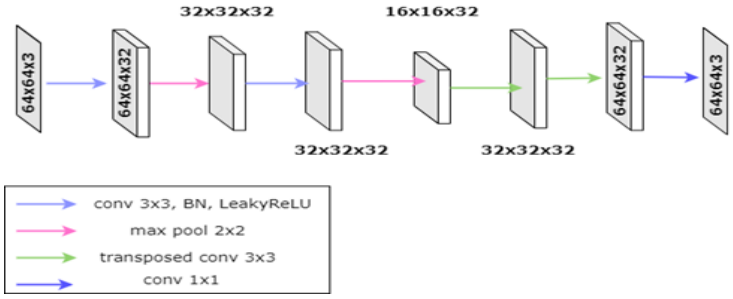


Fig. 8. Autoencoder Convolutional Architecture

2.5. UNet

The UNet architecture was first used in biomedical image segmentation. UNet extends the architecture of the fully convolutional neural network used for pixel-by-pixel classification by introducing matching contractile and expansive pairs of operations that mirror each other at the farthest end of the network. Similar to the convolutional autoencoder introduced in the previous method, U-Net is shaped like the letter "U" and consists of two main parts: the encoder (downsampling path) and the decoder (upsampling path). What distinguishes it from the convolutional autoencoder methodis that UNet also has a connecting part between the encoder and the decoder called a bridge. In addition, UNet also has a section called concatenation, which combines the features of the encoder layer with those of the corresponding decoder layer. This concatenation helps maintain high-resolution information in the decoder. Fig. 9 illustrates the UNet architecture used in this study. Current denoising and segmentation research employs UNet due to its simplicity and robustness. There are several studies on the topic of inpainting using the UNet method, as conducted by [10],[18],[19].

2.6. Partial Convolutional Neural Network

The third algorithm applied in this study is an alternative modification of the UNet architecture, which replaces the conventional convolutional layer with a partial convolutional layer, similar to the approach carried out [23]. This method uses a partial convolution approach to fill in the missing or damaged parts of the image, where convolution is performed only on valid pixels (non-masked areas). If there are invalid pixels (masks), then they are excluded from the convolution process. Only valid portions are

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

used to calculate the convolution output. The mask is updated at each convolution step to reflect the parts that have been filled in or remain blank. Fig. 10. shows the architectuopic of inpainting using the Partial Convolutional Neural Network method, which was conducted by [24].

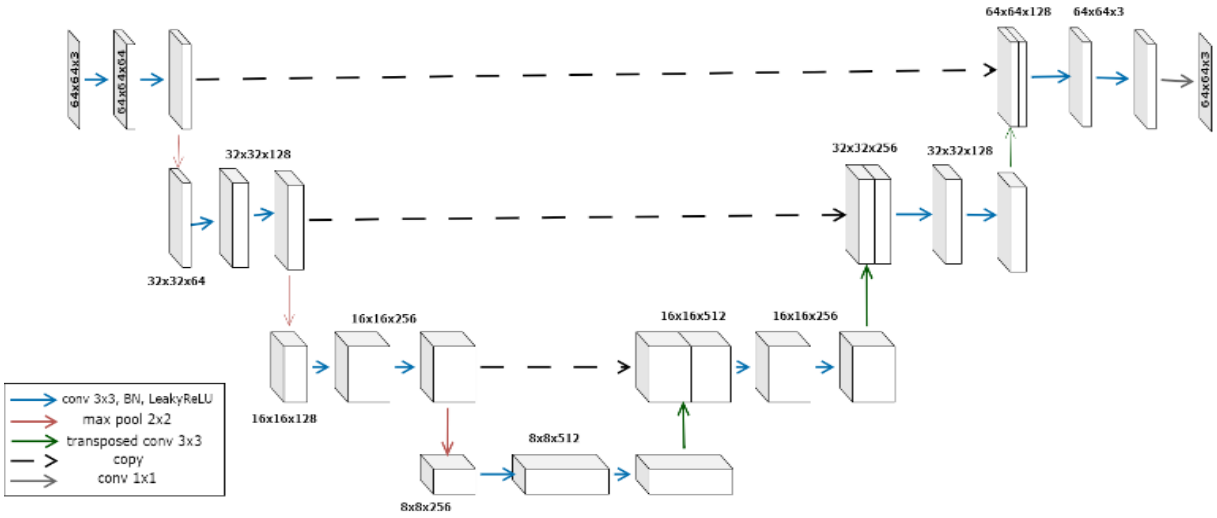


Fig. 9. UNet architecture

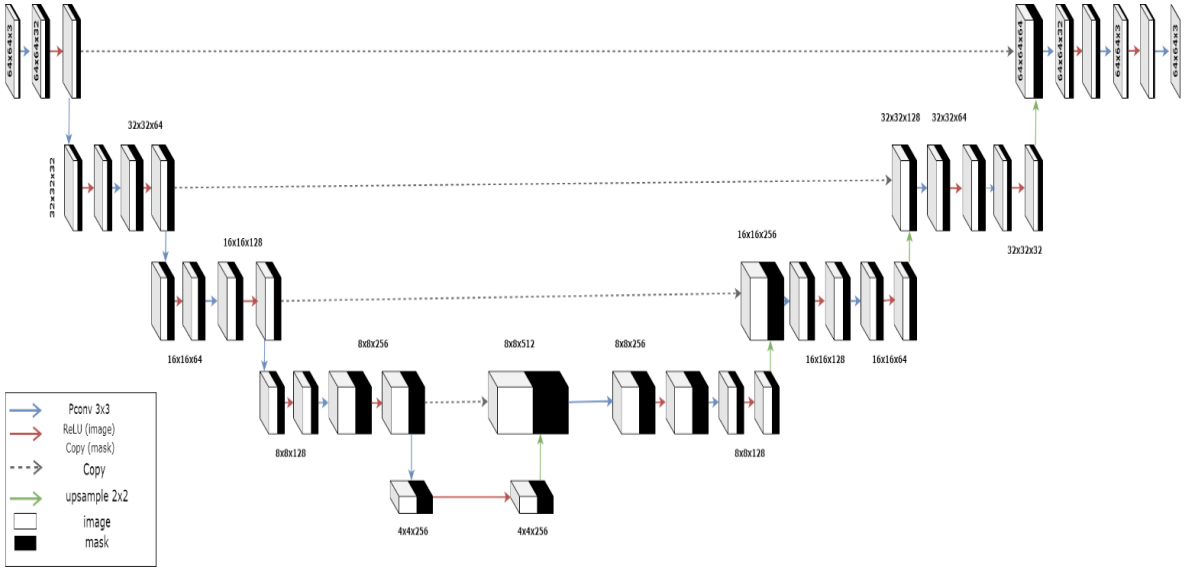


Fig. 10. The Partial Convolutional Neural Network.

2.7. ResUNet

The ResUNet architecture is a development of the UNet architecture that incorporates the principles of residual learning. ResUNet is designed to improve feature extraction and training efficiency in deep neural networks. The main component of ResUNet is the Residual Block, which distinguishes ResUNet from UNet. These blocks help address the problem of vanishing gradients by providing a direct shortcut connection from input to output. This connection enables the network to learn residual functions instead of direct mappings, facilitating and improving training efficiency. Fig. 11. illustrates the residual block, which consists of : (a) Input: $\boldsymbol{x}$, (b) Convolutional layer: consists of two (more) sequential batch normalization (BN) operations and activation functions, (c) Shortcut Connection: a bypass path that directly connects the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$ of the last convolution layer, (d) Element-wise Addition: the result of the shortcut connection added element-wise. This addition was made before the last activation function.

Mathematically, it can be expressed as Equation 1 [25], where $\boldsymbol{x}$ is the input, $\boldsymbol{y}$ is the output, and $\boldsymbol{F}(\boldsymbol{x})$ is is a non-linear function consisting of two layers of convolution, batch normalization, and activation function.

$$y = F(x) + x \tag{1}$$

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

Fig. 12. and Table 2 show the ResUNet architecture used in this study. The architecture consists of nine levels. Level one is the input, which is in the form of masked images of handwritten Javanese script. Levels two through four are encoder blocks, each consisting of Convolutional2D, Residual Block (ResBlock), and Maxpooling processes. Level five is a bridge that performs the Residual Block process. Levels six to eight are decoder blocks, where each level consists of the Convolutional2D Transpose, Concatenate, and Residual Block processes. Finally, at level nine, the Convolutional2D process is executed with the "sigmoid" activation function. At this level, the output obtained is the result of inpainting, which restores the handwritten Javanese script.

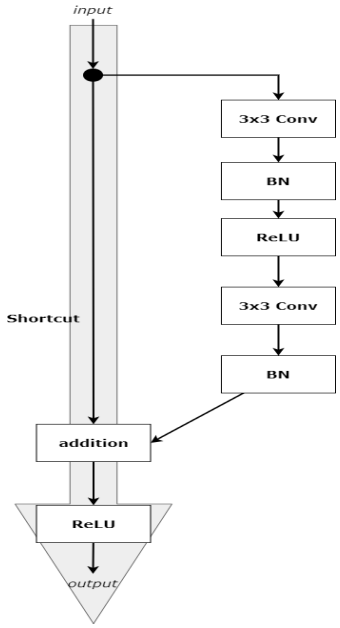


Fig. 11. Residual Block Architecture

Several segmentation studies using ResUNet, due to its advantages, have been conducted extensively, such as those by [25],[26],[27],[28]. There are several studies on the topic of inpainting using ResUNet-based methods such as those conducted by [11],[29].

Table 2. Parts of the ResUNet architecture

Process		Unit Level	Input	Output	Stride
Input		Level 1	(64,64,3)	(64,64,3)	-
Encoder Block	Conv2D Residual Block	Level 2	(64,64,3)	(64,64,64)	(1,1)
	Maxpooling		(64,64,64)	(32,32,64)	(2,2)
	Conv2D Residual Block	Level 3	(32,32,64)	(32,32,128)	(1,1)
	Maxpooling		(32,32,128)	(16,16,128)	(2,2)
	Conv2D Residual Block	Level 4	(16,16,128)	(16,16,256)	(1,1)
	Maxpooling		(16,16,256)	(8,8,256)	(2,2)
Residual Block		Level 5	(8,8,256)	(8,8,512)	(1,1)
Decoder Block	Conv2DTranspose Concatenate Residual Block	Level 6	(8,8,512)	(16,16,256)	(2,2)
	Conv2DTranspose Concatenate Residual Block	Level 7	(16,16,256)	(32,32,128)	(2,2)
	Conv2DTranspose Concatenate Residual Block	Level 8	(32,32,128)	(64,64,64)	(2,2)
Output		Level 9	(64,64,64)	(64,64,3)	(1,1)

http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

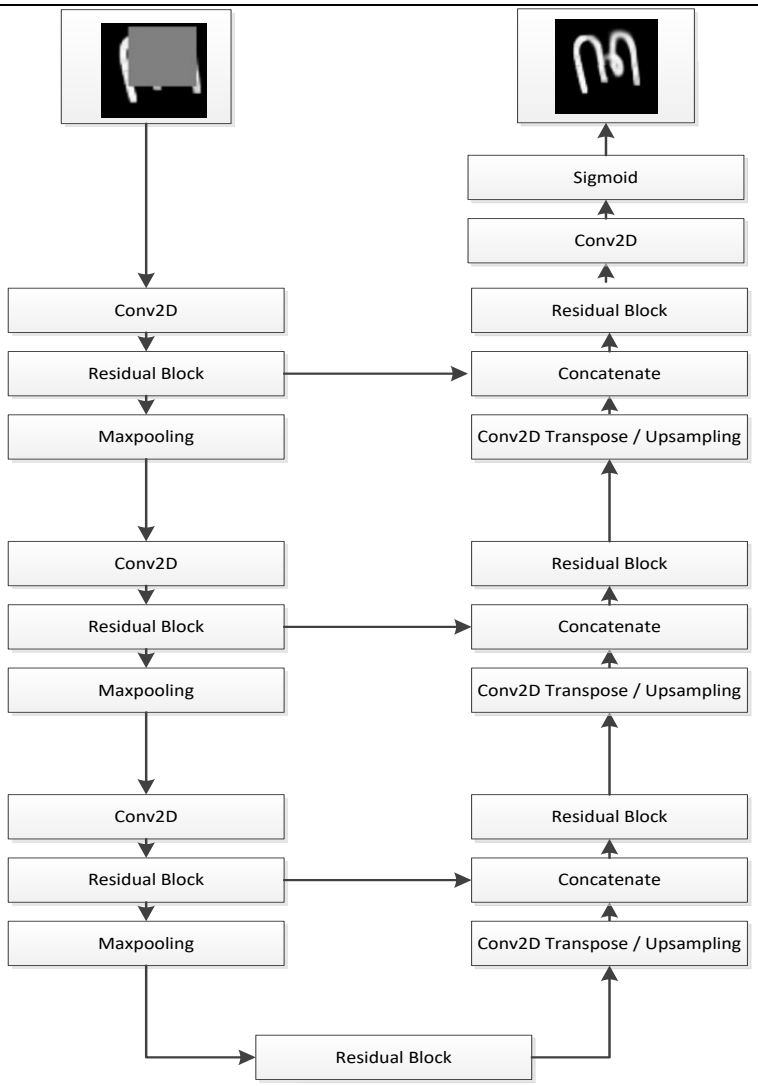


Fig.12. ResUNet architecture

3. Results and Discussion

This study compares four methods, namely the Convolutional Autoencoder method, Partial Convolutional Neural Network, UNet, and ResUNet. In this study, random masking is used, as shown in Fig. 13. The use of random masking is adapted to real conditions, where damage to the Javanese script can occur in different positions. The image size used is 64 x 64, and the masking size is half of the image size, which is 32 x 32. In Figure 13, it can be seen that the masking position may cover the entire script area, most of the script area, a small part of the script area, or even the characteristic parts of the script that distinguish it from other characters.

Fig. 13. Multiple masking positions

The hyperparameters used in this study are shown in Table 3. This hyperparamater were obtained through several trials, with the best results being selected. The number of epochs was set to 100, and the Adam optimizer was used. All experiments were conducted using Python scripts executed

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

on Google Colab Professional. The training was performed on a Windows system using an NVIDIA Intel Core i5 laptop.

Table 3. Hyperparameters used

Method	Batch Size	Learning Rate
Convolutional Autoencoder	32	0,001
Partial Convolutional	32	0,001
UNet	32	0,001
ResUNet	32	0,001

3.1. Comparative Experiments

3.1.1. Quantitative Experiments

There is currently no unique assessment metric for character inpainting, and this study utilizes image inpainting metrics to evaluate the model. A quantitative comparison of SSIM and PSNR was conducted. The structural difference between the reconstructed and original handwritten Javanese script images (Ground Truth) was used to assess reconstruction quality. While the human brain can quickly distinguish between two images, computers struggle with this task. Image similarity can be measured using methods beyond distance calculation. SSIM (Structural Similarity Index Measure) and PSNR (Peak Signal to Noise Ratio) are now widely used evaluation indicators.

The Structural Similarity Index Measure (SSIM) is used to calculate the similarity between two images. The SSIM value ranges from -1 to 1, with a value of 1 indicating perfect resemblance and -1 indicating total dissimilarity. The higher the SSIM value, the greater the similarity between the two images. SSIM can be calculated using Equation 2, where μ_A is the average pixel value of the image A, μ_B is the average pixel value of the image B. σ_{AB} is the covariance of the image A to the image B. σ_A is a variant of the image A and σ_B is a variant of the image B. $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$, where L is the dynamic range of the image $(2^{bit} - 1)$. Default values $k_1 = 0.01$ and $k_2 = 0.03$[19].

$$SSIM\ (A, B) = \frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)} \quad (2)$$

PSNR is an objective measurement method for image quality, and its results are nearly identical to those of human visual detection methods. PSNR is used to assess the degree of uniformity between an image and its reference image. PSNR is expressed in decibel logarithmic units (dB). The higher the PSNR value, the greater the similarity between the images. PSNR can be determined using Equation 4, where C_max^2 represents the maximum pixel value in the image. To calculate the PSNR, the MSE (Mean Square Error) value is required. MSE is the square of the error value obtained from all observed pixels. Equation 3 illustrates the MSE formula, where x and y represent the image's pixel coordinates. M and N are the dimensions of the image. S is the restored image, whereas C is the original image (ground truth)[30].

$$MSE = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} (S_{xy} - C_{xy})^2}{MN} \quad (3)$$

$$PSNR = 10 \log_{10} \left(\frac{C_{max}^2}{MSE} \right) \quad (4)$$

Table 4 displays the values of the loss, SSIM and PSNR functions for the four methods used in this study. These values are based on the testing data experiment. The values printed in bold represent the best results. The loss function value produced by the ResUNet method is the lowest compared to the other three methods (Convolutional Autoencoder and Partial Convolutional, and UNet), which is: 0.0167. The SSIM value achieved by the ResUNet method outperforms the other three methods, amounting to: 0.9319. Similarly, the PSNR value of the ResUNet method is the highest, namely: 18.9507. Therefore, it can be concluded that the ResUNet method is the most suitable approach for complatig the inpainting task in handwritten Javanese script.

Table 4. Values of loss, SSIM, and PSNR(dB) functions in the test data

Method	Loss Function	SSIM	PSNR(dB)
Convolutional Autoencoder	0,1870	0,7562	13,2133
Partial Convolutional	0,0255	0,8753	16,5963
UNet	0,0188	0,9223	18,1474
ResUNet	**0,0167**	**0,9319**	**18,9507**

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

3.1.2. Qualitative Experiments

The success of the method is measured not only quantitatively but also qualitatively. Qualitative measurements are demonstrated using SSIM and epoch comparison graphs. The four methods trained in this study produce SSIM graphs that are convergent and do not exhibit overfitting. These graphs are shown in Fig. 14. Fig. 14(a) is a comparison chart of SSIM and epochs generated using the Convolutional Autoencoder method. Fig. 14(b) corresponds to the Partial Convolutional Neural Network method, Fig. 14(c) to the UNet method, and Fig. (d) to the ResUNet method. Additionally, qualitative measurements are illustrated through comparative images of inpainting data testing using the four methods, which are shown in Fig. 15. and Fig. 16. Fig. 15. indicates that if the masking position covers most of the characters, some of the methods employed in this study produces incorrect inpainting results. In addition to the masking position covering most of the characters, the inpainting error can also be caused by the masking position covering the area that is a feature or differentiator from the other characters, as shown in Fig. 16.

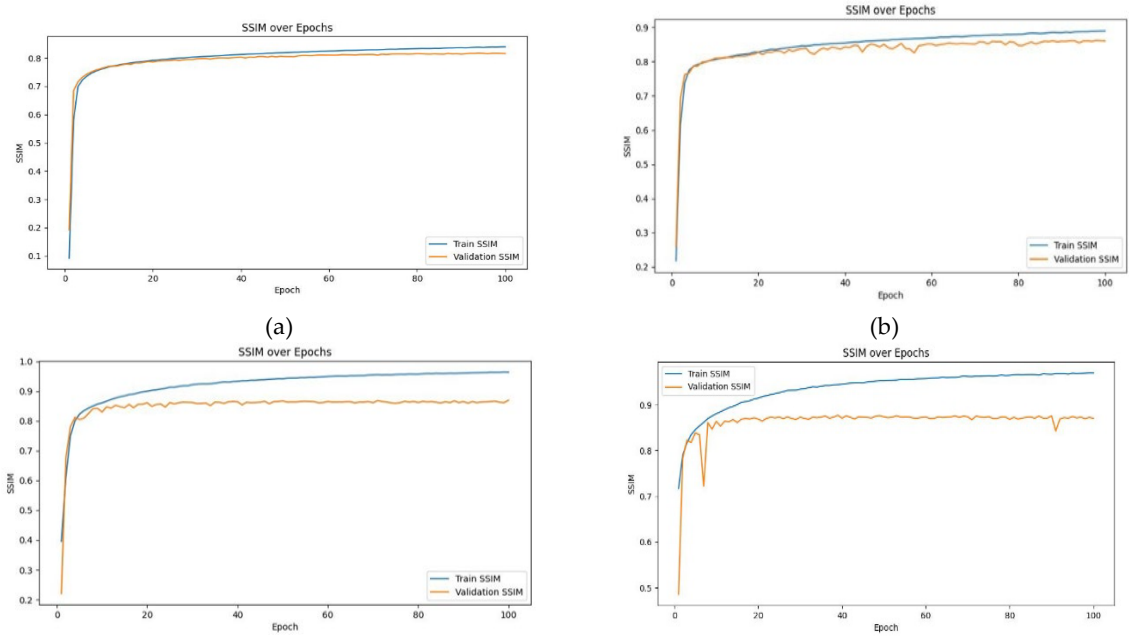


Fig. 14. SSIM graph

In Fig. 15., the Convolutional Autoencoder method is able to recognize the character "ra," although the inpainting result is not perfect. The trial using the Partial Convolutional Neural Network method produced results that matched the ground truth, even though they were not perfect (such as characters: "ra", "ta", and "ha"). The results that are in accordance with the ground truth and perfect are the characters: "nga" and "wa". Other script trials resulted in incorrect scripts (not in accordance with ground truth). The inpainting results using the UNet method produced characters that matched the ground truth, though some script forms were not perfect. Except for the inscription "ca", the inpainting is wrong. The trial with the ResUNet method produced characters that matched the ground truth, although the inpainting results for the "ca" script were not perfect.

Fig. 16. shows the results of inpainting under damage conditions in areas that are characteristic of the script. It can be observed that using the ResUNet method, the reconstructed script matches the ground truth. In contrast, the inpainting results using the UNet, Partial Convolutional Neural Networks, and Convolutional Autoencoder methods are less accurate compared to the ground truth. There is also the result of his inpainting in the form of incorrect handwritten Javanese script prediction.

Based on Table 4, Fig. 15. and Fig. 16., the ResUNet method demonstrates superiority over the other methods used in this study. The ResUNet method achieves better SSIM (Structural Similarity Index) and PSNR (Peak Signal-to-Noise Ratio) values because ResUNet uses the residual learning concept introduced in ResNet (Residual Networks). This enables the network to learn the difference between input and output more efficiently. Residual connections help in: (a) Addressing Gradient Degradation: Making it easier to train very deep networks by reducing the problem of vanishing

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

gradients, (b) Improved Convergence: Speeding up the training process and helps the network achieve better convergence, (c) Direct Information: Providing a direct path to relevant information, thereby minimizing the loss of critical information during the convolution process.

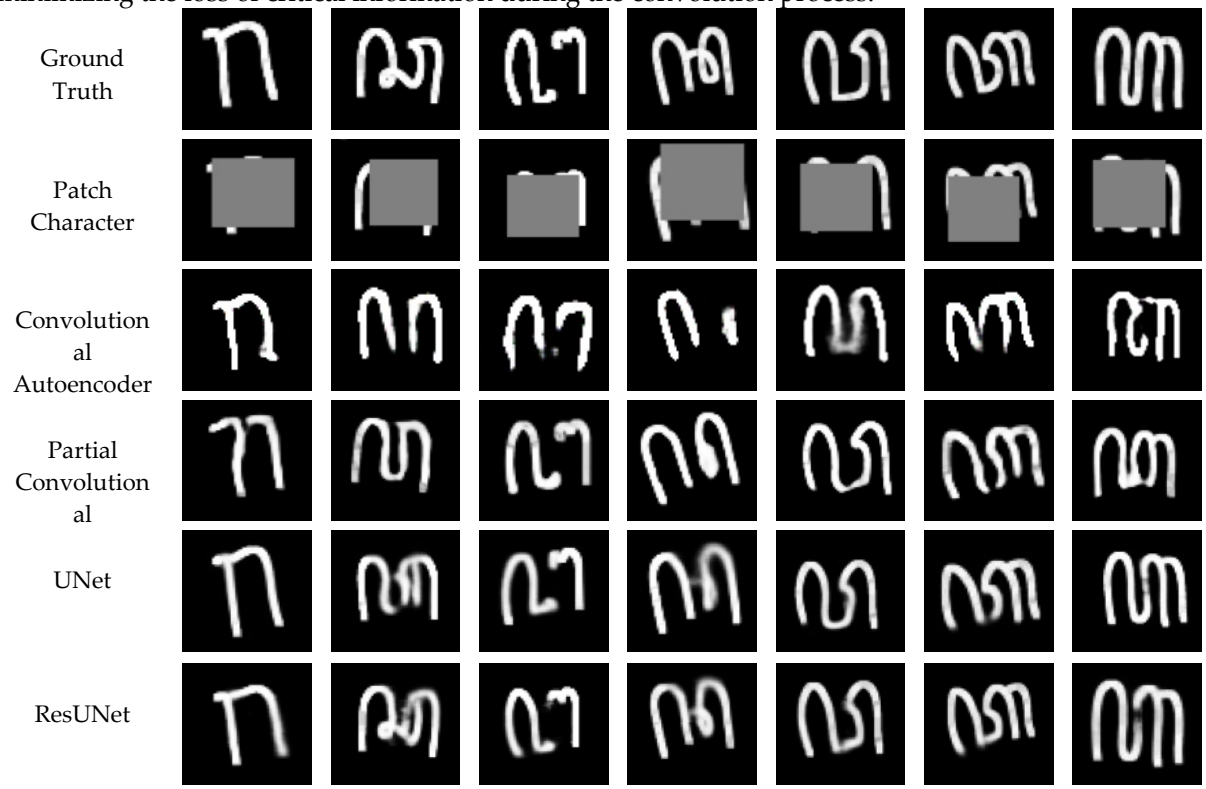


Fig. 15. Inpainting results with the mask covering most of the characters.

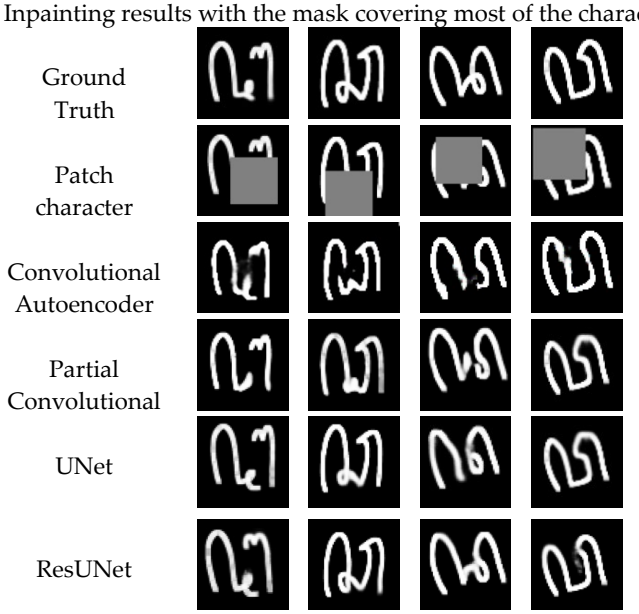


Fig. 16. Inpainting results with the mask covering distinctive areas of the script.

ResUNet retains more detailed information from the original input through residual connections. This is particularly important for Javanese script inpainting, which demands high precision in reconstructing intricate handwriting patterns. The combination of the encoder-decoder architecture and residual connections ensures that the fine details and global structure of the image are preserved, helping to produce a more realistic image that closely resembles the original.

ResUNet is more effective in handling different variations of data, such as Javanese handwriting, which features diverse styles and thicknesses. Its ability to adapt to these variations enhances the quality of the inpainting. In addition to residual connections, ResUNet also employs skip connections, which is similar to UNet. Skip connections help in combining features from the initial layers with the final layers, preserving the spatial information essential for image reconstruction.

 http://doi.org/10.26594/register.v10i2.4929

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

UNet uses skip connections but lacks residual connections. Although effective, UNet is less efficient at maintaining detailed information in highly complex images, such as Javanese script. The Partial Convolutional Neural Network (PCNN) is effective for filling in missing parts through partial convolution. However, without residual connections, PCNN is not as efficient as ResUNet at preserving fine details and coping with large data variations. Convolutional Autoencoders tend to lose detailed information due to their compressed and decompressed nature. Without skip connections and residual connections, the autoencoder produces less detailed and realistic results compared to ResUNet.

Based on the comparison, it can be concluded that the ResUNet method outperformed other existing approaches, as it was able to restore characters even in cases of extensive damage. In contrast, previous methods, such as those used in earlier research with the "Hamong Tani" dataset, were only effective when the damage was minimal, achieving lower accuracy and exhibiting limited restoration capabilities. This highlights the robustness and effectiveness of the ResUNet method in reconstructing Javanese characters with significant damage.

4. Conclusions

This investigation aimed to assess the efficacy of various deep-learning methods in reconstructing damaged handwritten Javanese scripts. Specifically, the study focused on the Convolutional Autoencoder, Partial Convolutional Neural Network, UNet, and ResUNet techniques to identify the most effective method for handling extensive damage and characteristic features of the script.

The most critical finding from this study is that the ResUNet method outperforms the other techniques in qualitative and quantitative measures. ResUNet excels due to its ability to retain detailed information, particularly when handling complex or extensive damage. The residual blocks in ResUNet enable better forward information transfer, ensuring that critical features of the Javanese script remain intact during reconstruction. In contrast, the Convolutional Autoencoder struggles with complex tasks, exhibiting the highest loss and lowest SSIM and PSNR values. The Partial Convolutional Neural Network improves upon the autoencoder by focusing on missing regions but fails to capture fine details. UNet performs well by using skip connections to retain information, yet it lacks the added advantage of residual learning. Without residual blocks, UNet is slightly less effective than ResUNet for detailed reconstruction. ResUNet's superior SSIM and PSNR scores highlight its effectiveness in preserving fine details and overall image quality, making it the most reliable method for reconstructing Javanese script. Even minor imperfections can lead to misinterpretations, emphasizing the importance ResUNet's precision for accurate restoration.

Considering the success of the ResUNet method in this context, future research should explore the scalability of this approach to other types of cultural heritage conservation, particularly in restoring ancient scripts or artworks with similar challenges. Further advancements could include integrating more robust data augmentation techniques or exploring transfer learning to enhance the model's effectiveness across varied datasets. Additionally, deploying this model in practical restoration projects within museums could offer valuable insights into its real-world applicability and help further refine its capabilities.

Author Contributions

F. Damayanti: Conceptualization, formal analysis, funding acquisition, investigation, methodology, project administration, resources, software, supervision, and writing - review and editing. E. M. Yuniarno: Data curation, investigation, resources, software, validation, visualization, and writing – original draft. Y. K. Suprapto: Investigations, visualization, and writing – review and editing.

Declaration of Competing Interest

We declare that we have no conflict of interest.

References

[1] D. Iskandar, S. Hidayat, U. Jamaludin, and S. Mukti Leksono, "Javanese script digitalization and its utilization as learning media: an etnopedagogical approach," *Int. J. Math. Sci. Educ.*, vol. 1, no. 1, pp. 21–30, 2023, doi: 10.59965/ijmsed.v1i1.24.

[2] A. Susanto, I. U. W. Mulyono, C. A. Sari, E. H. Rachmawanto, D. R. I. M. Setiadi, and M. K. Sarker, "Improved Javanese script recognition using custom model of convolution neural

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

network," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 6, pp. 6629–6636, 2023, doi: 10.11591/ijece.v13i6.pp6629-6636.

[3] A. Susanto, I. U. W. Mulyono, C. A. Sari, E. H. Rachmawanto, D. R. I. M. Setiadi, and M. K. Sarker, "Handwritten Javanese script recognition method based 12-layers deep convolutional neural network and data augmentation," *IAES Int. J. Artif. Intell.*, vol. 12, no. 3, pp. 1448–1458, 2023, doi: 10.11591/ijai.v12.i3.pp1448-1458.

[4] A. Setiawan, A. S. Prabowo, and E. Y. Puspaningrum, "Handwriting Character Recognition Javanese Letters Based On Artificial Neural Network Text Mining and Text Information Retrieval View project Artificial intelligence View project Handwriting Character Recognition Javanese Letters Based on Artificial Neura," *Netw. Secur. Inf. Syst.*, vol. 1, no. 1, pp. 39–42, 2019, [Online]. Available: https://www.researchgate.net/publication/343228550.

[5] A. Susanto, I. U. Wahyu Mulyono, C. Atika Sari, E. H. Rachmawanto, and D. R. I. Moses Setiadi, "An Improved Handwritten Javanese Script Recognition using Adaptive Threshold and Multi-Feature Extraction," *2022 Int. Semin. Appl. Technol. Inf. Commun. Technol. 4.0 Smart Ecosyst. A New W. Doing Digit. Business, iSemantic 2022*, no. September, pp. 66–70, 2022, doi: 10.1109/iSemantic55962.2022.9920462.

[6] F. Damayanti, Y. K. Suprapto, and E. M. Yuniarno, "Segmentation of Javanese Character in Ancient Manuscript using Connected Component Labeling," *CENIM 2020 - Proceeding Int. Conf. Comput. Eng. Network, Intell. Multimed. 2020*, no. Cenim, pp. 412–417, 2020, doi: 10.1109/CENIM51130.2020.9297954.

[7] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, "Image inpainting based on deep learning: A review," *Displays*, vol. 69, no. March, p. 102028, 2021, doi: 10.1016/j.displa.2021.102028.

[8] H. Liu, B. Jiang, Y. Song, W. Huang, and C. Yang, "Rethinking Image Inpainting via a Mutual Encoder-Decoder with Feature Equalizations," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12347 LNCS, pp. 725–741, 2020, doi: 10.1007/978-3-030-58536-5_43.

[9] M. C. Sagong, Y. G. Shin, S. W. Kim, S. Park, and S. J. Ko, "PEPSI : Fast image inpainting with parallel decoding network," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 11352–11360, 2019, doi: 10.1109/CVPR.2019.01162.

[10] S. Ozturk, "Image Inpainting based Compact Hash Code Learning using Modified U-Net," *4th Int. Symp. Multidiscip. Stud. Innov. Technol. ISMSIT 2020 - Proc.*, no. 120, pp. 1–5, 2020, doi: 10.1109/ISMSIT50672.2020.9255239.

[11] M. I. Hosen and M. B. Islam, "Masked Face Inpainting Through Residual Attention UNet," *Proc. - 2022 Innov. Intell. Syst. Appl. Conf. ASYU 2022*, 2022, doi: 10.1109/ASYU56188.2022.9925541.

[12] G. Song, J. Li, and Z. Wang, "Occluded offline handwritten Chinese character inpainting via generative adversarial network and self-attention mechanism," *Neurocomputing*, vol. 415, pp. 146–156, 2020, doi: 10.1016/j.neucom.2020.07.046.

[13] I. S. Jo, D. Bin Choi, and Y. B. Park, "Chinese character image completion using a generative latent variable model," *Appl. Sci.*, vol. 11, no. 2, pp. 1–19, 2021, doi: 10.3390/app11020624.

[14] H. Li *et al.*, "Generative character inpainting guided by structural information," *Vis. Comput.*, vol. 37, no. 9–11, pp. 2895–2906, 2021, doi: 10.1007/s00371-021-02218-y.

[15] Y. Weng, H. Zhou, and J. Wang, "Image Inpainting Technique Based on Smart Terminal: A Case Study in CPS Ancient Image Data," *IEEE Access*, vol. 7, pp. 69837–69847, 2019, doi: 10.1109/ACCESS.2019.2919326.

[16] B. Su, X. Liu, W. Gao, Y. Yang, and S. Chen, "A restoration method using dual generate adversarial networks for Chinese ancient characters," *Vis. Informatics*, vol. 6, no. 1, pp. 26–34, 2022, doi: 10.1016/j.visinf.2022.02.001.

[17] R. Liu *et al.*, "SCCGAN: Style and Characters Inpainting Based on CGAN," *Mob. Networks Appl.*, vol. 26, no. 1, pp. 3–12, 2021, doi: 10.1007/s11036-020-01717-x.

[18] J. Wang, G. Pan, D. Sun, and J. Zhang, "Chinese Character Inpainting with Contextual Semantic Constraints," *MM 2021 - Proc. 29th ACM Int. Conf. Multimed.*, pp. 1829–1837, 2021, doi: 10.1145/3474085.3475333.

[19] C. Xing and Z. Ren, "Binary Inscription Character Inpainting Based on Improved Context

ISSN 2502-3357 (online) | ISSN 2503-0477 (print)
10 (2) 2024 160-174

Encoders," *IEEE Access*, vol. 11, no. June, pp. 55834–55843, 2023, doi: 10.1109/ACCESS.2023.3282442.

[20] L. Zhao and Y. Lou, "An Auto-Encoding Network for Binary Inscription Character Inpainting based on U-Net and Self- Attention An Auto-Encoding Network for Binary Inscription Character Inpainting based on U-Net," pp. 0–21, 2023.

[21] L. Zhao and Y. Lou, "An Auto-Encoder of Inscription Character Inpainting based on Branch Convolutional Channel Attention Module An Auto-Encoder of Inscription Character Inpainting based on Branch Convolutional Channel Attention Module," pp. 0–20, 2023.

[22] T. Chavez, E. J. Roberts, P. H. Zwart, and A. Hexemer, "A comparison of deep-learning-based inpainting techniques for experimental X-ray scattering," *J. Appl. Crystallogr.*, vol. 55, no. 5, pp. 1277–1288, 2022, doi: 10.1107/S1600576722007105.

[23] A. Kornilov, I. Safonov, and I. Yakimchuk, "Inpainting of Ring Artifacts on Microtomographic Images by 3D CNN," *Conf. Open Innov. Assoc. Fruct*, vol. 2020-April, pp. 200–206, 2020, doi: 10.23919/FRUCT48808.2020.9087422.

[24] H. Patel, A. Kulkarni, S. Sahni, and U. Vyas, "Image Inpainting using Partial Convolution," pp. 1–9, 2021, [Online]. Available: http://arxiv.org/abs/2108.08791.

[25] M. W. Sabir *et al.*, "Segmentation of Liver Tumor in CT Scan Using ResU-Net," *Appl. Sci.*, vol. 12, no. 17, pp. 1–15, 2022, doi: 10.3390/app12178650.

[26] F. I. Diakogiannis, F. Waldner, P. Caccetta, and C. Wu, "ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data," *ISPRS J. Photogramm. Remote Sens.*, vol. 162, no. March 2019, pp. 94–114, 2020, doi: 10.1016/j.isprsjprs.2020.01.013.

[27] E. Mique and A. Malicdem, "Deep Residual U-Net Based Lung Image Segmentation for Lung Disease Detection," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 803, no. 1, 2020, doi: 10.1088/1757-899X/803/1/012004.

[28] V. Pollatos, L. Kouvaras, and E. Charou, "Land cover semantic segmentation using ResUNet," *CEUR Workshop Proc.*, vol. 2844, pp. 55–65, 2020.

[29] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image Inpainting: A Review," *Neural Process. Lett.*, vol. 51, no. 2, pp. 2007–2028, 2020, doi: 10.1007/s11063-019-10163-0.

[30] S. He and L. Schomaker, "DeepOtsu: Document enhancement and binarization using iterative deep learning," *Pattern Recognit.*, vol. 91, pp. 379–390, 2019, doi: 10.1016/j.patcog.2019.01.025.