Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 7344–7351
Marseille, 20-25 June 2022

Handwritten Character Generation using Y-Autoencoder for Character Recognition Model Training

Tomoki Kitagawa[1], **Chee Siang Leow**[1,2], **Hiromitsu Nishizaki**[1]
[1]Integrated Graduate School of Medicine, Engineering, and Agricultural Sciences, University of Yamanashi
4-3-11 Takeda, Kofu, Yamanashi 400-8511, JAPAN
[2]Artibrains LLC, 3-8-6 Joto, Kofu, Yamanashi 400-0861, JAPAN
kitagawatomoki@alps-lab.org, siang@artibrains.com, hnishi@yamanashi.ac.jp

Abstract

It is well-known that the deep learning-based optical character recognition (OCR) system needs a large amount of data to train a high-performance character recognizer. However, it is costly to collect a large amount of realistic handwritten characters. This paper introduces a Y-Autoencoder (Y-AE)-based handwritten character generator to generate multiple Japanese Hiragana characters with a single image to increase the amount of data for training a handwritten character recognizer. The adaptive instance normalization (AdaIN) layer allows the generator to be trained and generate handwritten character images without paired-character image labels. The experiment showed that the Y-AE could generate Japanese character images then used to train the handwritten character recognizer, producing an F1-score improved from 0.8664 to 0.9281. We further analyzed the usefulness of the Y-AE-based generator with shape images and out-of-character (OOC) images, which have different character image styles in model training. The result showed that the generator could generate a handwritten image with a similar style to that of the input character.

Keywords: handwritten characters generation, optical character recognition (OCR), Y-Autoencoder

1. Introduction

The re-emergence of deep learning since the third winter of artificial intelligence has led to significant achievements in various fields. Specifically, it has led to the mainstreaming of deep learning-based systems that use large amounts of data to train a model. Supervised-learning deep learning models have been particularly successful among the various deep learning methods. Various studies have extensively investigated semi-supervised and unsupervised learning; however, most commercialized systems still commonly use supervised learning.

In the fields of text image recognition among the various research fields, machine-printed optical character recognition (OCR) (Li et al., 2019; Min et al., 2020), and connectionist temporal classification (CTC)-based (Graves et al., 2006) end-to-end handwritten text recognition (HTR) (Graves and Schmidhuber, 2008; Ly et al., 2017; Ingle et al., 2019) require a large amount of data to train, which makes the text recognizer more generalized and high-performance. Therefore, data augmentation, such as CutMix (Yun et al., 2019), mixup (Zhang et al., 2018), and Randaugment (Cubuk et al., 2020), and mosaic augmentation (Hao and Zhili, 2020), which is based on image processing, is generally used to learn more general models by increasing the amount of data.

However, adequate data augmentation methods still have to be designed by humans. Therefore, several methods of increasing the amount of data for supervised learning-based deep learning models require a large amount of data, including online-based character generation using recurrent neural networks (RNNs) and offline-based adversarial adaptive generation networks (GANs), which generate images. Among them, DeepWriting (Aksan et al., 2018) is one of the methods that generate online-based online characters, while ScrabbleGAN (Fogel et al., 2020) generates handwriting-like text images from actual text images.

However, most of these methods have been studied mainly for the English (alphabets), which has a small number of character types, while there is little research on handwritten character generation in Chinese and Japanese, which have more character types than English. In English, words composed of alphabetic strings rather than character units are the smallest constituent units of meaning. However, in Japanese and Chinese, a single character has a significant meaning, so it is important to identify and recognize the character units. Hence, there is a high demand for highly accurate character recognizers; however, they require a large number of character images for training.

Especially in languages like Chinese and Japanese, some characters are used frequently, and some are not, which can often lead to data imbalanced. This means that there is little data on rare characters. However, in terms of increasing this rare character data, the related costs of data preparation are too high, as it involves having humans write and label each character. Therefore, a method to increase the number of images per character is essential for recognizing character images in these languages.

Therefore, in this paper, we propose a method for generating various handwritten characters using a Y-Autoencoder (Y-AE) model in order to build a dataset for training a handwritten character recognizer auto-

matically. Although there are many papers on the generation of handwritten character images and handwritten style conversion, few studies on languages have as many character types as Japanese. In some cases, pairs of characters are required. Our proposed conditional handwritten character generator based on Y-AE is unique in that the training data of the handwritten character recognizer can be used as the training data for the generator. Besides, inspired by (Karras et al., 2019) and (Patacchiola et al., 2019), we use a Y-AE training strategy and use an embedding layer to learn the representation of each character and inject the features using adaptive instance normalization (AdaIN) computation into the upsampling features. Our proposed architecture can train the generative model without paired character images but also learn the writing style of a handwritten image based on encoded features to generate the specific character image.

Since we would like to achieve a variety of character handwriting for training a character recognizer, our research approach is different in property from the existing research on style transformation models. Moreover, we can realize more diverse characters by applying the characters generated by the Y-AE model to existing style transformation models. However, in this study, we show that the generated characters have a positive effect on the training of the character recognizer.

In this paper, we attempt to generate one-to-many images, which can be used to train and improve the performance of the handwritten character recognizer. The experiment results showed that the proposed Y-AE handwritten character generator could be used to increase the amount of training data for training a better character recognizer. Furthermore, the experiment showed that the handwritten character recognizer achieved an 0.0617 point improvement in F1-score compared to the recognizer trained without any generated images.

The contributions of this paper are summarized as the following:

1. We show that injecting the character embedding features into the AdaIN allows the Y-AE model to control the content compared to a simple vector.
2. We test whether the handwritten images generated by the Y-AE-based generator can be used to train the handwritten character recognizer with improved recognition accuracy.
3. We show that the Y-AE model can generate a specific handwritten character image by using different out-of-character (OOC) images with different character styles.

2. Related Works

Before the deep learning era, the autoencoder (Rumelhart et al., 1986) had been introduced, which

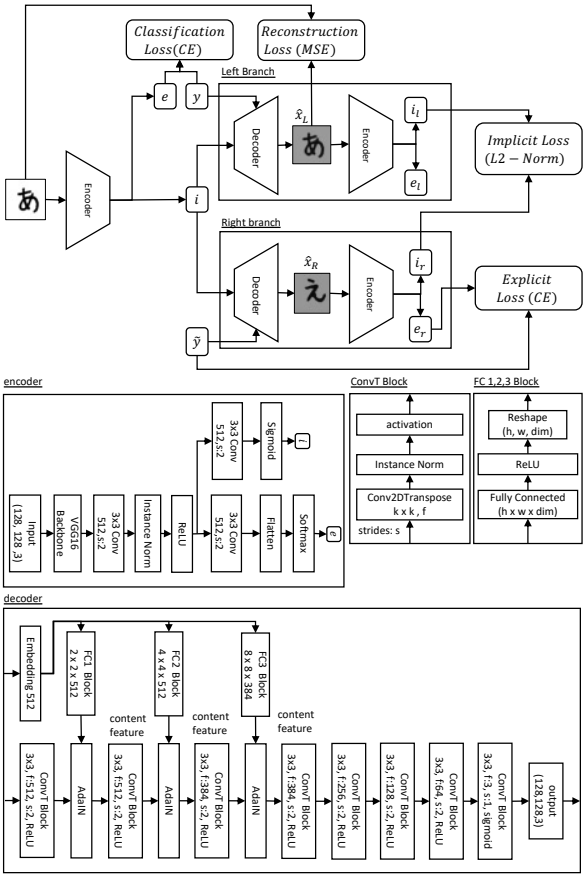

Figure 1: Overall architecture of the proposed image generator model

showed that latent presentation is useful for pre-training the neural network. Since deep learning became the focus of attention, studies on autoencoders have become popular again. This led (Kingma and Welling, 2014) let to propose a variational autoencoder (VAE), showing that the autoencoder can be used to learn the latent variables to be centered isotropic multivariate Gaussians, which can be useful for data generation.

Moreover, (Goodfellow et al., 2014) proposed adversarial generative networks (GAN), from which numerous types of GANs have now been proposed. For example, (Zhu et al., 2017) proposed CycleGAN to transfer a style to another with an unpaired image. (Bo Chang et al., 2018) introduced a handwritten Chinese characters generation using CycleGAN, which transferred the style of a machine-printed font image to a handwritten-characters image using the paired and unpaired character images. Although their approach could generate a very realistic pencil style image, it could only convert a single font image to only a single type of handwritten character image, and it was also limited to generating single patterns due to its using the same font style. With the idea of the GAN and VAE, (Kong and Xu, 2017) proposed cCGAN and cCVAE, which could be used to generate handwritten Chinese characters. However, they claimed that the quality of the generated image was not stable. Additionally, they

faced numerical issues in computing the Kullback – Leibler (KL) divergence loss.

At the same time, neural style transfer was introduced by (Gatys et al., 2016), who used excited the use of a neural network to render images from one domain style to another and introduced the style loss and content loss. Their results concluded that the style and content in the convolutional neural network (CNN) were separable. (Ulyanov et al., 2016) proposed the TextureNetwork for improving the complex textures style transfer, and (Ulyanov et al., 2017) found that the TextureNetwork could be improved by changing the batch normalization (BN) (Ioffe and Szegedy, 2015) to instance normalization (IN) by computing the mean and variance across the spatial dimensions independently for each channel and each sample. This led (Huang and Belongie, 2017) to introduce a novel normalization, adaptive instance normalization, which is a normalization method that aligns the mean and variance of the content features with specific style features that try to tackle the problem of computation speed cost and the restrictions to a pre-defined set of styles. (Karras et al., 2019) proposed a style-based generator architecture for generative adversarial networks, which significantly improved the style transfer performance and re-designed the control over the image synthesis process.

As mentioned above, there are many studies on increasing image variations by using VAE or by style transformation. Our approach differs from the existing studies in that we propose a method to generate images for data augmentation based on the Y-AE model, using the training set for character recognition as is.

3. Y-AE-Based Handwritten Character Image Generator

3.1. Model architecture

The overall architecture of our Hiragana character image generator model is shown in Fig. 1. The proposed model is based on the Y-AE (Patacchiola et al., 2019) architecture, which consists of an encoder and a conditional decoder. The encoder uses a VGG16 (Simonyan and Zisserman, 2015) backbone feature extractor to encode RGB images with a $(128, 128, 3)$ shape and output the style representation i and a character label e of an input image. The decoder takes both the style representation i and a character label e as input to decode an output handwritten character image. The character label e is converted to a 512-dimensional embedding vector by an embedding layer. The embedding vector is input to three different fully connected (FC) layers to extract the content feature shown in Equation (1):

$$content\_feature(s) = FC(Emb(s)) \tag{1}$$

where FC is a fully connected layer, Emb is the embedding layer, and s is the character label $\tilde{y}$ or label y. The output dimension of the FC layer should be matched to the shape of the intermediate upsampling features, and the output vector of the FC is reshaped to the exact shape of the upsampling features.

To inject the content features from the FC layers, we use the AdaIN shown in Equation (4) by injecting the content information, as shown in Equation (2) and (3),

$$x = content\_feature(s) \tag{2}$$

$$y = ConvT(i) \tag{3}$$

$$AdaIN(x, y) = \sigma(y)(\frac{x - \mu(x)}{\sigma(x)}) + \mu(y) \tag{4}$$

where $ConvT$ is a transposed convolutional layer, and i is the style representation encoded from the input image. The kernel size used in the CNN and upsampling is 3×3, strides 2×2 and apply IN with ReLU activation except the i output of encoder and decoder $\hat{x}$ apply the sigmoid activation function, and the encoder output label $\tilde{y}$ uses the softmax function.

3.2. Loss functions

The losses used in this paper were adapted from the original Y-AE (Patacchiola et al., 2019), in which they consist of four separate components. Firstly, it calculates the classification loss, followed by the cross entropy (CE) between the output e with label y using Equation (5) and the reconstruction loss, the mean squared error (MSE), with Equation (6),

$$\mathcal{L}_{cls} = CE(e, y) \tag{5}$$

$$\mathcal{L}_{reconst} = |\hat{x}_L - x|^2 \tag{6}$$

where e is the output of the encoder, y is the label of the character image, $\hat{x}_L$ is the left branch decoded image, and x is the input image. Secondly, we calculate the implicit loss, the L2-norm with the following Equation (7):

$$\mathcal{L}_{im} = \|i_r - i_l + \epsilon\|_2 \tag{7}$$

where the i_r and i_l are the left branch and right branch style outputs, and ϵ is $1.0e - 15$. Thirdly, the explicit loss, the CE as shown in following Equation (8):

$$\mathcal{L}_{ex} = CE(e_r, \tilde{y}) \tag{8}$$

where the e_r is the output e of the right branch, and $\tilde{y}$ is the random character label. Finally, the total loss is shown in Equation (9), which is used to backpropagate the gradient to the Y-AE model.

$$\mathcal{L}_{total} = \mathcal{L}_{reconst} + \mathcal{L}_{cls} + \mathcal{L}_{im} + \mathcal{L}_{ex} \tag{9}$$

4. Handwritten Character Recognizer

It is known that the evaluation of a deep learning-based generative model is very difficult, and the evaluation metrics are active in the research field (Lucic et al., 2018; Guan and Loew, 2019). In this paper, we focused on evaluating the generated images on

Table 1: Datasets and statistics for training of the generator and the recognizer models

(a) Dataset for the Y-AE generator. The training image set for Y-AE and the seed image set used for image generation are the same.

	Dataset	Num. of images
Training	ETL7	32,200
Image generation	ETL7	32,200

(b) Datasets for training and evaluation of the recognition model

	Dataset	Num. of images
Training	ETL7	32,200
Validation	ETL8G	8,510
Testing	ETL9G	9,200

whether they can be used to improve the accuracy of the handwritten character recognition. To evaluate the generated character images, we use the simple CNN-based character recognizer shown in Fig. 2 to evaluate the improvements in recognition accuracy. The CNN-based model takes a gray-scaled image with a $(128, 128, 1)$ shape. An input image is normalized in the range of 0 to 1 to make a prediction.

5. Experiments

5.1. ETL Character Database

In this paper, we aim to improve the accuracy of the handwritten character recognition by using the Y-AE model to generate different writing styles of the same character and using the generated images to train a handwritten character recognition. To experiment with the generation capabilities of the proposed Y-AE model, we experimented with the 46 Seion characters image types to evaluate the generated character image. We used the ETL character database (Electrotechnical Laboratory, et. al., 2011), which consists of nine subsets containing millions of Japanese handwritten characters. Table 1, shows the subsets used for training. ETL7 was used for training the Y-AE and generating the handwritten character images. The character images were converted to RGB due the VGG16 backbone and denoised by using the OpenCV's fastNlMeansDenoisingColored[1] method. After denoising the image, the images were randomly processed by one of the following transformations.

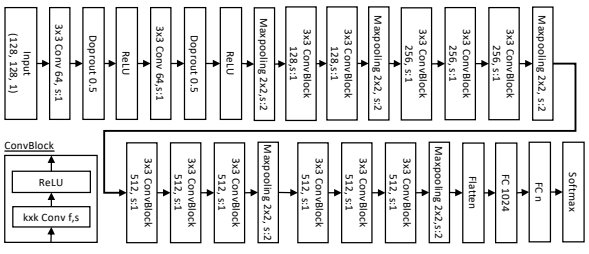

Figure 2: The architecture of a CNN-based simple character recognizer model

Table 2: Character recognition performances (F1-score)

Training dataset	F1-score
ETL7 Seion only	0.8664
Generated image only	0.9192
ETL7 Seion + generated images	0.9281

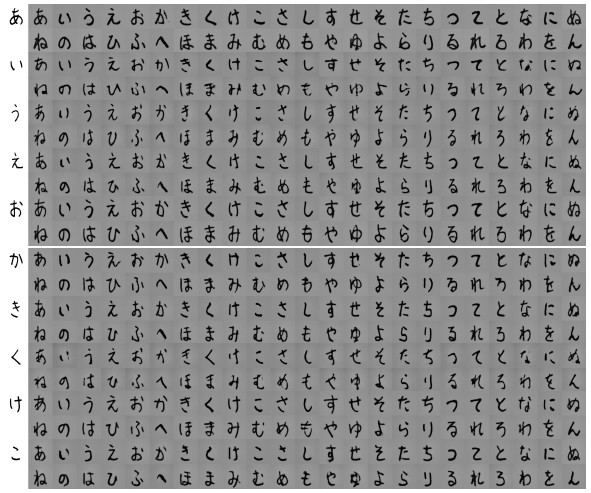

Figure 3: Y-AE Generated Japanese Hiragana Images

1. Normalized to range between 0 and 1.
2. Binarization by thresholding the image with the average of pixels, and normalizing to range between 0 and 1.
3. Multiplying the binarized image via thresholding, denoising the image to smooth the text contours, and normalizing to range between 0 and 1.

For the handwritten character recognition, we used ETL7 for the training data, ETL8G for the validation data, and ETL9G for the testing data. The character images are converted to grayscale and thresholded.

5.2. Hyper-parameters

5.2.1. Y-AE training parameters

The encoder weights are initialized using Xavier's method (Glorot and Bengio, 2010), and the decoder weights are initialized with a random normal initializer with a mean of 0 and standard deviation of 0.01 and then bias initialized with constant -5. To prevent overfitting, we use the $L2$ regularizer with $10e-4$. The CNN layer is computed with padding to maintain the size of the latent features. An Adam optimizer with an initial learning rate of $10e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10e-7$ was used to train the Y-AE. The batch size was 8, and the number of epochs was 200.

[1] https://docs.opencv.org

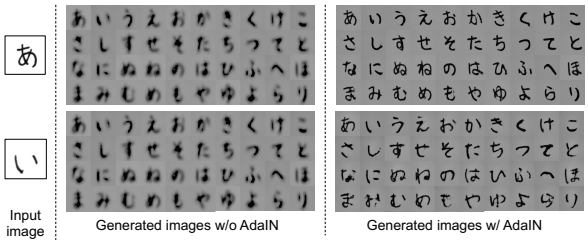

Figure 4: Compare generated images

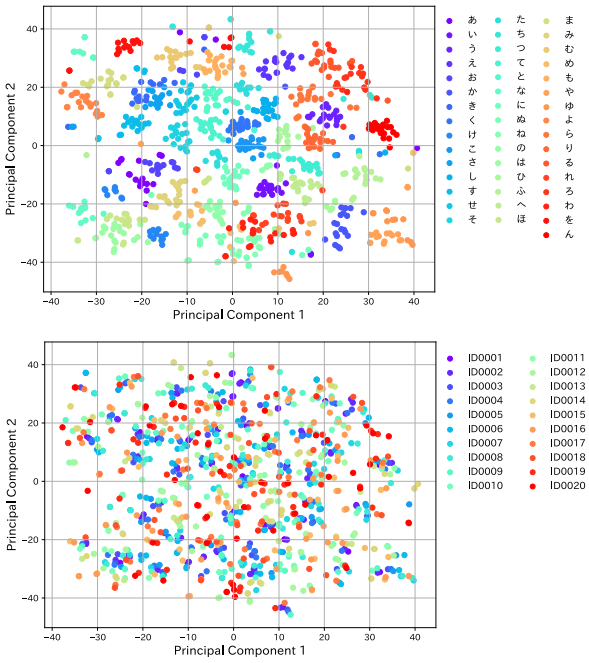

Figure 5: t-Distributed Stochastic Neighbor Embedding(t-SNE) visualization. The top shows the visualization of the characters images and the bottom shows the distributions for each writer.

5.2.2. CNN-based recognizer training parameters

The CNN-based recognizer was trained with an Adam optimizer with an initial learning rate of $10e-3$. We trained the CNN-based recognizer at a fixed 40k iterations due to the different amounts of data. To evaluate the CNN-based recognizer performance, we trained it from scratch with and without the Y-AE generated image.

5.3. Results

The Y-AE generated Japanese Seion Hiragana character images are shown in Fig. 3. In Fig. 3, we visualized generated 46 sorts of Seion Hiragana images from the 20 character images of あ, い, う, え, お, か, き, く, け, and こ. We generated 46 sorts of Hiragana images from each input image. From Fig. 3, we show that the generated image is readable with respect to the Hiragana character. Nevertheless, the generated Hiragana character image differs from the same generated character images. We used all the 700 images from each of the 46 types of Hiragana characters each to generate 46 Seion Hiragana, leading to $700 \times 46 \times 46$,

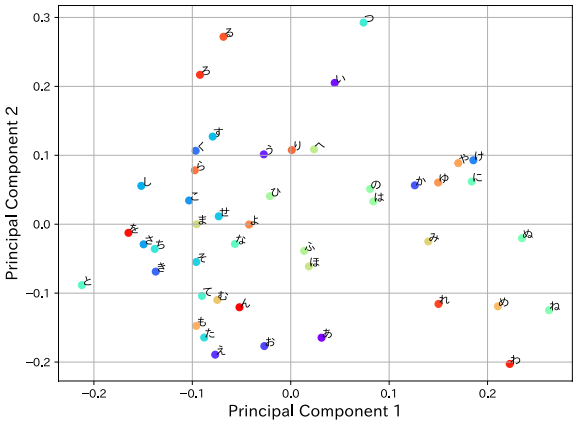

Figure 6: The PCA visualization of embedding features in the decoder of the Y-AE

i.e., a total of 1,481,200 images. To evaluate whether the synthetic data was usable to train a recognizer, we trained a simple CNN-based recognizer with the ETL7 of 700×46 Seion Hiragana, totaling $32,200$ training images, and the ETL9G of 200×46 Seion Hiragana, totaling at $9,200$ images, as baseline test data to compare the recognizer that used the generated images training the CNN-based recognizer.

The generated 1,481,200 images used to train the recognizer were preprocessed with the Otsu binarization threshold method (Otsu, 1979). The test images were used to evaluate both models. The baseline model was trained with only the 32,200 images from ETL7, and the proposed model was trained with both the original images of ETL7 and the generated images based on ETL7. In Table 2, the CNN-based recognizer trained with the images generated by the Y-AE improved the F1-score 0.0617 points from 0.8664 to 0.9281. This showed that the CNN-based recognizer could classify Seion Hiragana character images more precisely.

6. Analysis and Discussion

6.1. Effectiveness of AdaIN

First, we investigate the need to apply AdaIN to the Y-AE model. To analyze the need for AdaIN, we compared the generated images of the Y-AE model with and without AdaIN. The generated images are shown in Fig. 4. Fig. 4 shows that the Y-AE model with AdaIN changed the generated image relative to the input image, whereas the YAE model without AdaIN did not change relative to the input image. In other words, it is clear that the introduction of AdaIN generates an image that takes into account the style information of the input image.

6.2. Accuracy for each character class

Table 2 showed the averages of the F1 scores for all the character classes. By checking the F1 scores for each class, F1 scores did not improve for all the classes (results for each class are omitted). When trained with only the generated images, the F1 scores decreased for

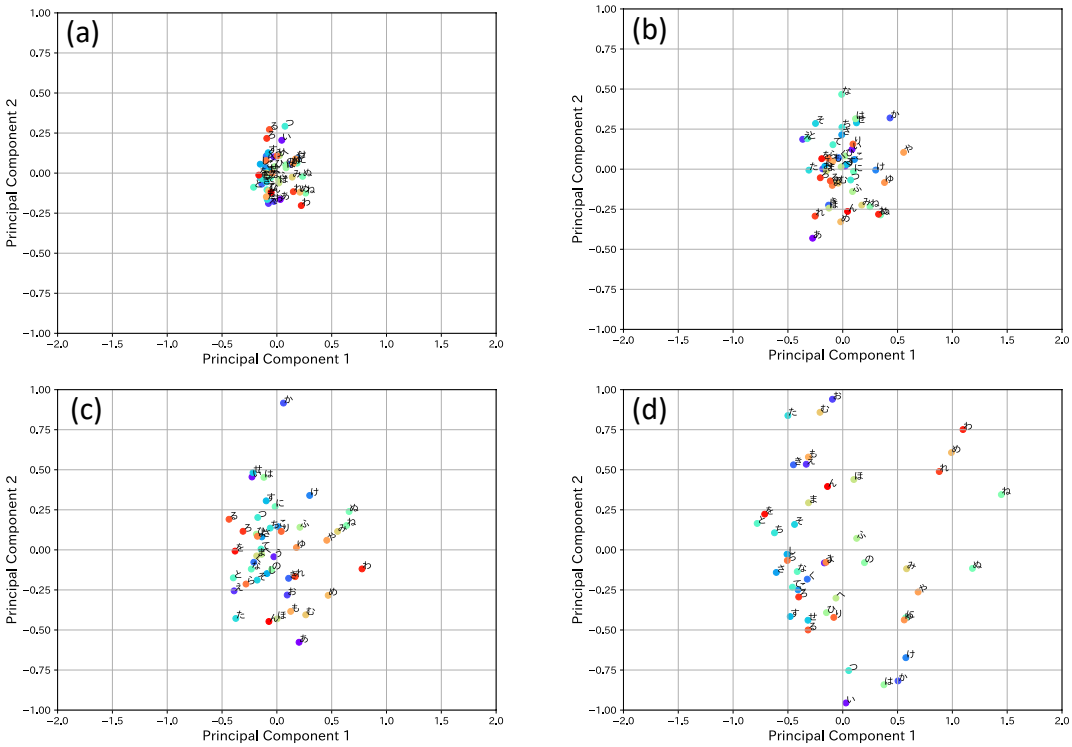

Figure 7: PCA visualization. (a) Embedding features from Fig. 6. (b), (c), (d) are the latent features of the FC1, FC2, and FC3 layers on the same scales, respectively

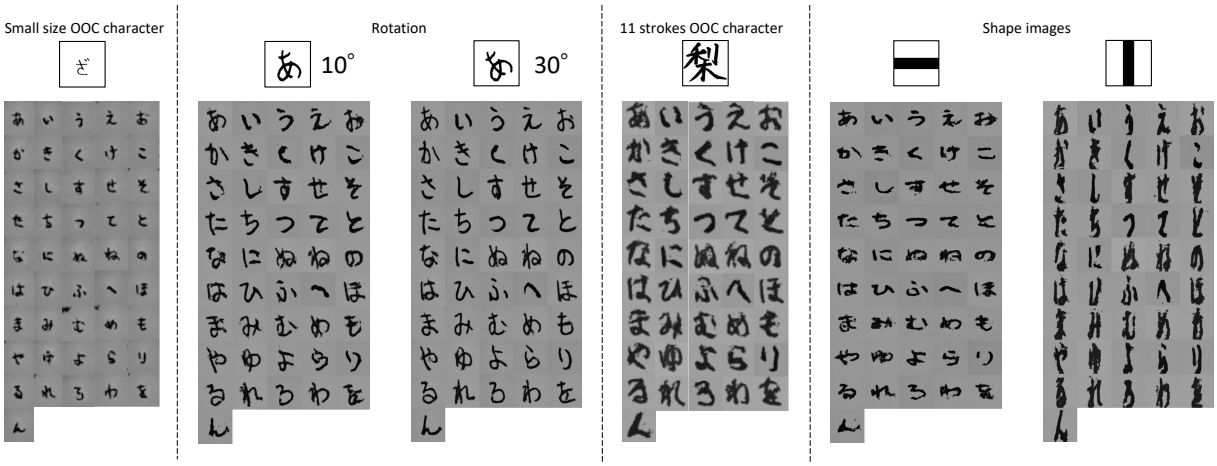

Figure 8: Generated images in resized, different rotation, OOC character, shape images

5 of the 46 classes, and when trained with the generated images and ETL, the F1 scores decreased for 8 of the 46 classes. These results indicate that not all classes of images are well generated. Further improvement of the generation model and selection of generated images are needed in this matter.

6.3. Analysis of Latent Features

We further analyze the proposed Y-AE model by analyzing the encoder and decoder latent variables. To analyze the encoder latent variables, we randomly picked 920 images, including 46 sorts of Hiragana characters written by 20 writers, from ETL7, to extract the encoder latent variables (shown in Fig. 5). To analyze the 920 neighbors of the latent variables, we visualized with t-SNE method. The parameters of the t-SNE method are shown as follows.

- Perplexity: 30
- Iterations: 1,000
- Number of components in embedded space: 2

The top of Fig. 5 shows that the encoder can learn the character image style representation, which is similar to the same character images. While it can differentiate the characters, the bottom of Fig. 5 shows that the

encoder latent variables are independent of the writer's handwriting style.

For the decoder, we visualized the embedding feature extract from the embedding layer and the latent features from the FC layers using principal component analysis (PCA). From Fig. 6, we can see that the embedding features output from the embedding layer in the decoder are capable of learning the differences between the characters. For example, the characters め, ぬ, ね, わ or さ, ち, and き have a very closed distance between each character, which shows the embedding layer can learn to differentiate between the character's embedding vector. We further analyzed the FC1, FC2, and FC3 layers latent variables injected with AdaIN by ranging the PCA components on the same scale. The same-scaled PCA visualization principal components 1 and 2 are shown in Fig. 7. From Fig. 7, we can see the distribution of the latent features scattered in the higher layers. This allows the Y-AE model to slowly distinguish between the characters in the latter layers, which then allows the decoder to generate the targeted character image.

め character can be similar to ぬ, ね, わ, when it goes through the FC layers, Fig. 7 (b), (c), (d) show the latent features slowly distinguish the differences between the similar words.

6.4. Analysis of different input images

To test the synthetic capabilities of the Y-AE model, we tested the different images from training, as shown in Figs. 8. Figs. 8 are resized to smaller OOC characters, rotations, OOC character images and shape images. In Figs. 8, the proposed Y-AE can track and differentiate the sizes or rotations to generate with the respective properties of the image. Additionally, we can see that the Y-AE is even able to generate even the input from the OOC. Furthermore, we tested with a shape image, as which shown in the first and second images from the right of Figs. 8. Surprisingly, it showed that the Y-AE is able to generate character images with the shapes in lines only. From these results, we can see the effectiveness and robustness of the embedding features being injected with the AdaIN. The generator is able to generate the same character but in a different writing style. Hence, the generator can be used to generate Japanese Hiragana character images, which can then be used to improve the recognizer as shown in Section 5.3.

7. Conclusion and Future Work

We proposed a Y-Autoencoder-based handwritten character image generator to generate more images for training data, which can then be used to train and improve the recognizer. The embedding layer and the AdaIN computation allowed us to train a robust handwritten character generator. The generator does not need paired images for training and generates different character images with different input images. From the results in Section 5, we showed that the proposed Y-AE could generate an increased number of handwritten character images, which can then be used to train and improve the recognizer.

Currently, the generator is being restricted that needs at least one image to generate images. In the future, we will try to control the latent variables without a single image to generate more character image styles. Furthermore, we will aim to generate more types of character image types, such as Kanji (Chinese character), with the hope that the Y-AE model can generate several character images at once.

8. Bibliographical References

Aksan, E., Pece, F., and Hilliges, O. (2018). Deepwriting: Making digital ink editable via deep generative modeling. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2018)*.

Bo Chang, Q. Z., Pan, S., and Meng, L. (2018). Generating handwritten chinese characters using cyclegan. In *Proceedings IEEE 2018 Winter Conference on Applications of Computer Vision*.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3008–3017.

Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., and Litman, R. (2020). Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4323–4332.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13 international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, pages 249–256.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, volume 2, pages 2672–2680.

Graves, A. and Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, et al., editors, *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 545–552.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376.

Guan, S. and Loew, M. (2019). Evaluation of generative adversarial network performance based on direct analysis of generated images. In *Proceedings of 2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–5.

Hao, W. and Zhili, S. (2020). Improved mosaic: Algorithms for more complex images. *Journal of Physics: Conference Series*, 1684:012094, nov.

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519.

Ingle, R., Fujii, Y., Deselaers, T., Baccash, J., and Popat, A. (2019). A scalable handwritten text recognition system. In *Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 17–24.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of 2nd International Conference on Learning Representations (ICLR)*, pages 1–14.

Kong, W. and Xu, B. (2017). Handwritten chinese character generation via conditional neural generative models. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*.

Li, H., Wang, P., Shen, C., and Zhang, G. (2019). Show, attend and read: A simple and strong baseline for irregular text recognition. *the AAAI Conference on Artificial Intelligence*, 33:8610–8617, 07.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are gans created equal? a large-scale study. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, volume 31, pages 698–707.

Ly, N.-T., Nguyen, C.-T., Nguyen, K.-C., and Nakagawa, M. (2017). Deep convolutional recurrent network for segmentation-free offline handwritten japanese text recognition. In *Proceedings of the 2017 International Conference on Document Analysis and Recognition (ICDAR)*, pages 5–9.

Min, F., Zhu, S., and Wang, Y. (2020). Offline handwritten chinese character recognition based on improved googlenet. In *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, pages 42–46.

Otsu, N. (1979). A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66.

Patacchiola, M., Fox-Roberts, P., and Rosten, E. (2019). Y-autoencoders: disentangling latent representations via sequential-encoding. *CoRR*, abs/1907.10949.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., (1986). *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press, Cambridge, MA, USA.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. (2016). Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, page 1349–1357.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4113.

Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, pages 1–13.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

9. Language Resource References

Electrotechnical Laboratory, et. al. (2011). *The ETL Character Database*. distributed via The National Institute of Advanced Industrial Science and Technology. `http://etlcdb.db.aist.go.jp`