

Fragmented-cuneiform-based convolutional neural network for cuneiform character recognition

Agi Prasetiadi, Julian Saputra

Faculty of Informatics, Institut Teknologi Telkom Purwokerto, Purwokerto, Indonesia

Article Info

Article history:

Received Dec 17, 2022

Revised Feb 3, 2023

Accepted Mar 10, 2023

Keywords:

Convolutional neural networks

Cuneiform

Fragmented characters

Kernel

Model architecture

ABSTRACT

Cuneiform has been a widely used writing system in one of the human history phases. Although there are millions of tablets, have been excavated today, only around 100,000 tablets have been read. The difficulty in translating also increased if the tablet has damaged areas resulting in some of its characters become fragmented and hard to read. This paper investigates the possibility of reading fragmented cuneiform characters from Noto Sans Cuneiform font based on convolutional neural network (CNN). The dataset is built on extracted 921 characters from the font. These characters are then intentionally being damaged with specific patterns, resulting set of fragmented characters ready to be trained. The model produced by this training phase then being used to read the unseen fragmented pattern of cuneiform sets. The model also being tested for reading normal characters set. From the simulation, 83.86% accuracy of reading fragmented characters are obtained. Interestingly, 96.42% accuracy is obtained while the model is being tested for reading normal characters.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Agi Prasetiadi

Department of Informatics, Telkom Purwokerto Institute of Technology

Purwokerto, Central Java 53147, Indonesia

Email: agi@ittelkom-pwt.ac.id

1. INTRODUCTION

The damaged area of excavated cuneiform tablets raised problems in interpreting the old text. It decreases the overall contents in regards to understanding the text due to the vanishing sentences. To preserve the information as much as possible, if a fragment of a character remains, the full form of a character can be conjectured based on the remaining fragment. Guessing what the character next is quite challenging, but it is not impossible. Cuneiform is a writing system developed in 3500-3000 BC by the Sumerians in the Mesopotamian city of Uruk, or what is now known as Iraq [1]-[3]. Cuneiform became the writing system used in several languages of the Mesopotamian civilization, such as Sumerian, Akkadian, and Hittite [2], [4], [5]. Although this letter was used as writing system in several Mesopotamian civilizations, the way to read it was different. For example, even though Sumerians and Akkadians used cuneiform, the way to read each letter was different because of the differences in the language of each civilization [6]-[8]. The reader may realize that the readings are trivial, the same character does not guarantee the same reading in the other part of the sentence. Instead, the reading depends on character or the words that may follow [9].

According to the Unicode character encoding standard, cuneiform is divided into three blocks of the number of letters in the Sumero-Akkadian script. The first block covers the most basic cuneiform characters, starting from U+12000 to U+123FF. The second block starts from U+12400 to U+1247F that includes cuneiform numbers and punctuation. Lastly, the third block starts from U+12480 to U+1254F, including addi-

tional cuneiform characters dating from the early dynastic period [8]. This paper only used the first block, until U+12398, as the main training source, because the basic characters are assigned effectively until this area.

It has been recorded that archaeologists have excavated and found around 0.5-2 million cuneiform tablets, of which 30,000-100,000 have existed and were successfully read by researchers. In comparison, 1.9 million cuneiform tablets still need to be read [10], [11]. It will be beneficial if we could make automation cuneiform character recognition. We have the technology to do this; we did not implement it yet to Cuneiform character and set it according to its characteristics.

Convolutional neural network (CNN) is one of the technologies used widely in image classification. It can be used to face recognition with 99.63% accuracy [12]. It also can recognize disease on the plant until 99.53% accuracy [13]. It is also used to implement optical character recognition. It reaches 98.96% accuracy for recognizing Chinese uppercase characters in the application of the internet of things [14]. In the previous research, cuneiform character recognition reaches 91% accuracy for its classification using polygon approximation for 350 cuneiform dataset symbols [15]. This study investigates the possibility of reading fragmented or full form cuneiform characters based on trained CNN models with pure fragmented characters only as its training source.

2. METHOD

2.1. CNN

CNN is a unique type of neural network that differs from other types of neural networks. It is specifically designed for feature extraction from images using a systematic extraction process based on multi-convolution [16]-[18]. Compared to regular neural networks, CNN is more efficient in image recognition since only image features are processed in hidden layers, rather than millions of rows of data as in regular NNs. CNN is renowned for its high level of image classification, with up to 99% accuracy in recognizing plant diseases through leaf imagery [13]. Moreover, it can extract abstract features such as physical effects simulated in videos [19].

Convolution is a fundamental technique used in image processing for feature extraction, which involves using a small matrix to apply operations such as blurring, line detection, and image sharpening [20]. In CNN, an image undergoes multiple convolution processes, resulting in the extraction of specific features that are then classified using a regular NN. The architecture of CNN is illustrated in Figure 1.

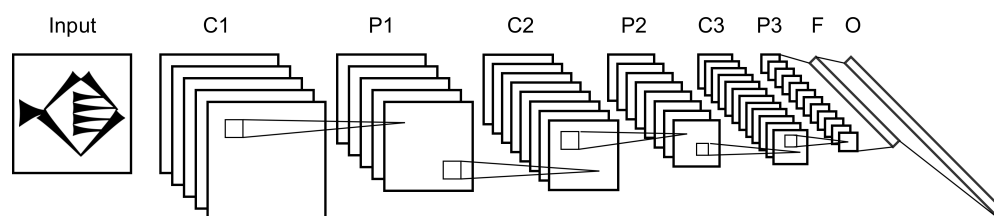


Figure 1. CNN architecture

2.2. Architecture

There are various types of hidden layers, such as the convolution layer (C), the sub-sampling layer (S), the flatten layer (F), and the dense layer (D). This paper exploits mostly the first four types of hidden layer. The first type of hidden layer is responsible for convolution, the second is responsible for local averages or local maximum for sub-sampling and resolution reduction, while the third and the fourth acts as a traditional multi-layered perceptron classifier.

No standard requires these layers to be arranged to form a particular architecture. The architecture arrangement could be C8-M2-C16-M2-F, or C8-C8-M2-C8-C8-M2-F, or C8-C8-C8-M2-C16-C16-M2, or many possible sequences. The composition and parameters of each layer will be examined in this study, ranging from 3 to 4 convolution layers stacked similar with LeNet architecture [21], together with pretrained architecture such as visual geometry group 16 (VGG16) [22] and ResNet-RS50 [23]. Since we are trying to read Cuneiform characters, thus we need at least 921 nodes in this output layer.




2.3. Cuneiform character complexity

In order to prepare a reasonable number of nodes during the classification process, the overall Cuneiform character complexity needs to be analyzed. Cuneiform characters have a much higher number of strokes compared to the Latin alphabet. If the scaled image feed into the CNN pipelines is too small, the aliasing effect will be higher. Due to its black and white nature of the image, if $\mathbf{P}_{m \times n}$ is the matrix of certain Cuneiform character, then in order to capture full complexity of a character, the edges should be evaluated horizontally and vertically altogether by subtracting its own shifted matrix. Thus, the maximum edges of a character, for $\{i, j \in \mathbb{N} \mid 1 \leq i < m, 1 \leq j \leq n\}$, can be defined as follows.

$$m = \left\lceil \max \left(\max_{i \in m} \left(\sum_{j=1}^{n-1} |p_{i,j} - p_{i,j+1}| \right), \max_{j \in n} \left(\sum_{i=1}^{m-1} |p_{i,j} - p_{i+1,j}| \right) \right) \right\rceil \quad (1)$$

As a result, the following Table 1 contains the most complex characters. Based on this table, it can be seen that the most complex characters based on the number of strokes, is called as "Cuneiform Sign an Plus Naga Squared" with unicode U+12031. This character has maximum 46 lines alternately within one characters.

Table 1. List of detected most complicated characters

Rank	Unicode	Character
1st	U+12031	
2nd	U+1228E	
3rd	U+1216C	

This most complex Cuneiform character can be divided into 23 stroke parts, or into 11.5 part since these strokes are mostly repeatable more than twice. If all directions from horizontal or vertical part are taken into account, total we need at least 133 parts to cover this characters. As the result, the number of nodes needed during the classification process before reaching 921 conclusion nodes, are at least 133 nodes within fully connected layers. In this paper, the number is recommended to be increased to 160 nodes for diversification.

3. THE MODELS

There are 3 kind of schemes conducted in this experiment. The first scheme, model A, aims to distinguish whether a character is a fragmented character or an unfragmented character. Thus, two classes are needed in order to train model A, which are 'fragmented characters' and 'unfragmented characters'. This function similar to automatic faulty diagnosis [24], [25]. Let's take a look at Figure 2. The first third fragment models consist of fragmented triangular characters. The second, third fragment models consist of half-block fragmented characters, and the last third fragment models consist of quarter block fragmented characters. These are the fragmented character models used in this paper. The dataset can be found at [26].



Figure 2. Training pattern

First, 921 different characters with Unicode code from U+12000 to U+12398 are produced for 'unfragmented characters' class and 'fragmented characters' class. The 'unfragmented characters' class is being prepared without further operation. However, within 'fragmented characters' class, each character is being applied crop function among one of these 12 fragmented character models randomly. Thus, we have 921 images for each class. Next stage, each class member is separated into two more groups, 'training' and 'testing' set, with ratio 70:30, resulting in 645 characters of training class and 276 characters of each testing class.

The second scheme, model B, aims to guess Cuneiform characters from its fragmented character. This training dataset consists of 921 different classes representing 921 cuneiform characters. Inside the training dataset, each character class is divided into 12 different fragment models. No unfragmented characters are being trained in this model. Then inside the testing dataset, each character class is divided into 15 distinct fragmented models as can be seen at Figures 3(a) to 3(d), each of them differs their style with their former training model, resulting 921×15 characters are being prepared for testing the model.

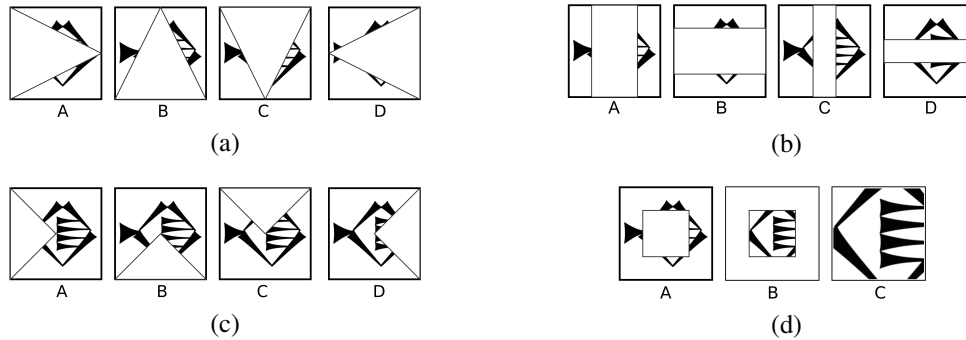


Figure 3. 2nd scheme, B model tests (a) B1: half triangle, (b) B2: rectangle, (c) B3: quarter triangle, and (d) B4: middle

The third scheme with the same model B aims to read Cuneiform characters in its full shape. This third scheme validation is easier than the previous scheme. Full unfragmented cuneiform characters are needed to test model B. Table 2 shows the summary of each scheme.

Table 2. Scheme 1, 2, and 3 for training the models

Scheme	Model	Classes	Detail
1	A	2	differentiate whether it is fragmented or not
2	B	921	guess the fragmented reading
3			guess the full shape reading

3.1. Data augmentation

The characters prepared on the dataset are extracted from Noto Sans Cuneiform, a font type developed by Google. However, the 12 different fragment models used for each character are limited compared to real Cuneiform characters excavated from sites. As a comparison, the Chinese ancient handwritten characters database (CASIA-AHCDB) includes over 2.2 million annotated handwritten characters based on 10,350 character classes with nearly 213 variation samples per character [27]. Offline Japanese handwriting databases have also been developed with a higher degree of variation and 2.005 variation samples per character class [28]. A higher number of samples typically leads to better recognition of each class variation [29].

Thus, 12 variations for each character are not good enough, representing the real case scenario. However, there is exist a technique to enrich the dataset variation called augmentation. [30] intentionally derived various datasets based on their medical image dataset which, by nature, is hard to obtain. In this research, further augmentation is performed to obtain various styles and models for each character.

Since curving in clay has fewer of freedom variations than handwriting in the paper, each character class is augmented into 100 different styles. There are 5 effects applied on each character augmented in this preprocessing phase; distortion, rotation, zoom, crop, and resize. Each effect has different spectrum of variation. The example of this augmentation result can be seen at Figure 4.



Figure 4. Augmentation variation

3.2. Architecture

The first scheme, called model A, has 3 types with little variation on its fully-connected layer (FCL): A32, A64, and A128. Table 3 shows the architecture. After flattening, nodes connect to the FCL based on its model type: A32, A64, and A128 have 32, 64, and 128 nodes in its FCL, respectively. All nodes are combined into one output node to determine whether the input image is a fragmented cuneiform character. The second and third schemes use the same model B. Model B is created by selecting the best-performing models from the configurations in Table 4. These models are trained with the fragmented characters dataset's training and validation data. The treatment after training is divided into 2 schemes: Scheme 2 and Scheme 3.

The reason most of the model candidates configured in C64(5)-S2-C96(5)-S2-C128(5)-S architecture is that to emulate the established Japanese characters recognition architecture based on CNN from [31], which is configured at model 3 as original architecture. It is because that the Japanese characters have similar characteristic with Cuneiform characters. Model 5 and 6 are the transfer learned models based on VGG16 and ResNetRS50 architecture. Full base architecture can be seen at the following Table 5.

Table 3. Model A architecture

Layer	w × h × n	Filter
Input	32 × 32 × 1	-
C	30 × 30 × 32	3 × 3
S	15 × 15 × 32	2 × 2

Table 4. The architecture candidates for model B

Model	Architecture
1	C64(5)-S2-C96(5)-S2-C128(5)-S5-F-D160
2	C64(5)-S2-C96(5)-S2-C128(5)-S2-F-D160
3	C64(5)-S2-C96(5)-S2-C128(5)-S3-F
4	C64(5)-S2-C96(5)-S2-C128(5)-S3-C160(3)-F
5	VGG16-F
6	ResNetRS50-F

Table 5. Base architecture for model 1, 2, 3, 4

Layer	w × h × n	Filter
Input	64 × 64 × 1	-
C ₁	60 × 60 × 64	5 × 5
S ₁	30 × 30 × 64	2 × 2
C ₂	26 × 26 × 96	5 × 5
S ₂	13 × 13 × 96	2 × 2
C ₃	9 × 9 × 128	5 × 5

4. SIMULATION RESULT

4.1. Scheme 1

Figure 5(a) is the simulation result for the first scheme. It can be seen that model A32 has the highest accuracy around 98.73% differentiating fragmented characters and not during its testing phase. Contrarily, model A64 has the lowest accuracy 98.17% on average among three models.

4.2. Scheme 2

The first thing to do before diving into scheme 2 is to select the best model from 6 candidates as mentioned before in Table 4. The models are trained with fragmented characters, and then their accuracy, loss, and accuracy bias are observed. In this paper, the accuracy bias is defined as the distance between training accuracy and validation accuracy. If the bias is very close to 0, then it can be said that the model experience little overfitting. Based on Figures 5(b) and 5(c), it can be seen that model 4 has the best performance. The validation accuracy in reading fragmented characters reaches 83.86%, while its training accuracy reaches 96.67%. Model 4 is the further modification based on original Japanese Characters Recognition at model 3, where another convolutional layer with 160 filters and 3×3 pixel kernel is added. Thus, model 4 is selected for model B.

After this, 4 different kinds of model tests are conducted to see model B ability to read different fragmented Cuneiform characters. Each model test has different blocker shapes that will be applied to all 921 Cuneiform characters resulting set of fragmented characters. Table 6 shows the detail of each model test.

The block positions for model B are presented in Figure 3, while the effect of these positions on reading accuracy is demonstrated in Figure 6. In the case of B1, there are 4 variants of triangle block positioning as can be seen at Figure 3(a). These 4 patterns block half of overall characters' shape. It can be seen from Figure 6(a) that all blockers affects model B's reading accuracy fluctuatively on all epochs. On average, model B has the highest accuracy around 6.73% when reading fragmented Cuneiform characters with B pattern, which the characters are blocked from the lower side. As contrast, model B has the lowest accuracy around 2.5% when reading fragmented characters with C pattern. This result gives a glimpse idea that somehow the combination of north west and north east part of the characters are significant part where information lies.

Let's see B2 case. There are also 4 variants of rectangular block positioning as can be seen at Figure 3(b). Two patterns block half of overall characters' shape and the other two patterns block quarter of overall characters' shape. Figure 6(b) shows that on average, model B reads fragmented characters with highest accuracy around 42.45% on quarter rectangle horizontal pattern. However, half reactangle vertical pattern on average gives the lowest reading accuracy on 1.21%, the lowest even among B1 scheme, where it can only read 12 characters correctly out from 921 characters.

For the case of B3, it also has 4 variants of triangle blocks just like B1 case. The difference is that the blockers on B3 case block only quarter of the overall characters' shape as can be seen at Figure 3(c). The simulation as can be seen from Figure 6(c) starts showing promising results. All patterns reach convergence at their 5th epoch. This time, the highest reading accuracy is obtained by A pattern at 94.47% accuracy. It seems like that in their nearly full shape characters, A pattern which occupies northern, eastern, and southern part of the characters, exposes the most needed information to identify a character's identity.

Lastly, B4 case consists of 3 variants of fragmented characters. Those fragments are varied based on their middle rectangular shape as can be seen at Figure 3(d). Based on Figure 6(d), model B reads exceptionally well C pattern reaching 61.24% accuracy and model B also reads well B pattern reaching 51.36% accuracy. Interestingly model B can't guess the characters properly on A pattern even 75% of its characters' shape is revealed from four directions, where it is only able to reach 6.08% accuracy.

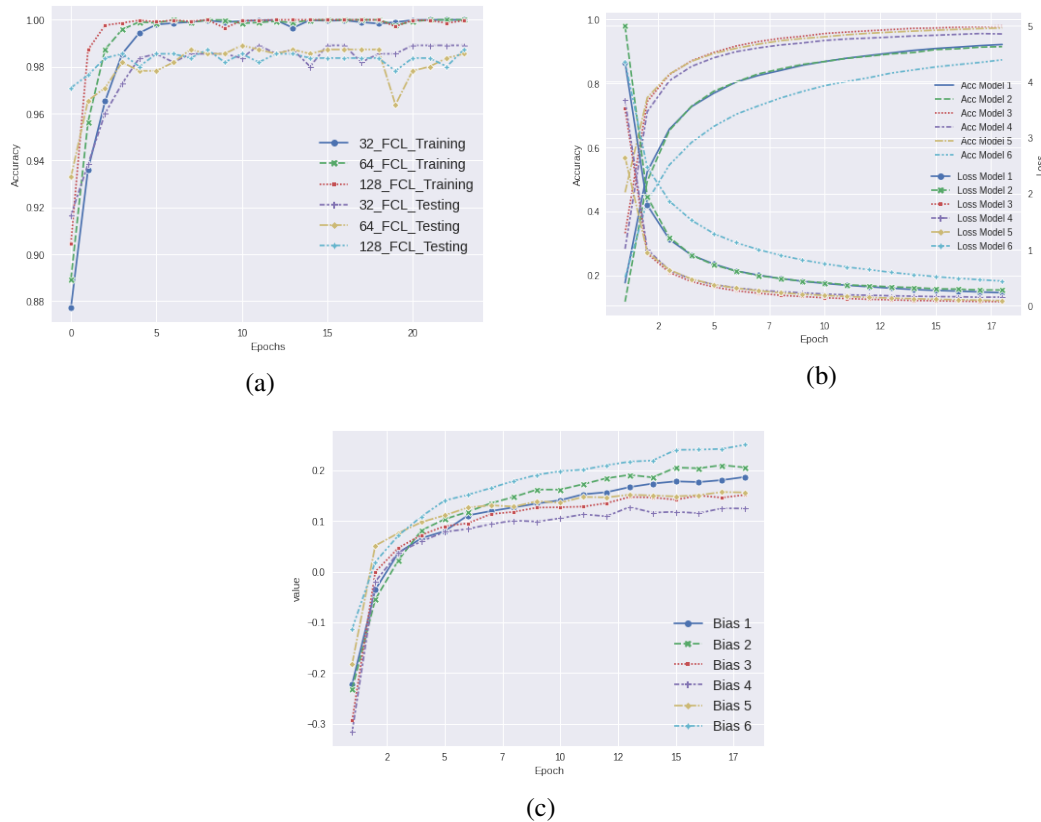


Figure 5. The overall performances for (a) 1st scheme, classifying fragmented and full shape characters, (b) 2nd scheme performances' measurement, and (c) 2nd scheme, performance bias

Table 6. Patterns for model test on reading fragmented characters

Model Test	Blocker Type
B1	Half Triangles
B2	Half and Quarter Rectangles
B3	Quarter Triangles
B4	Middle Rectangles

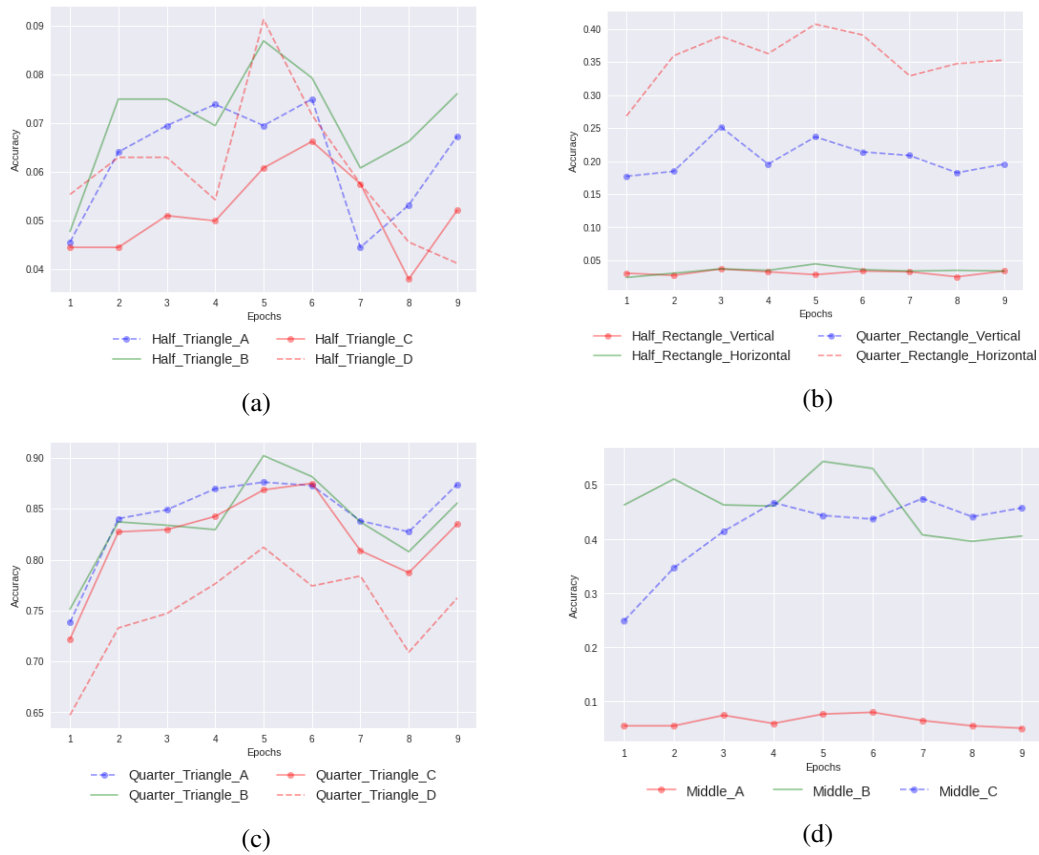


Figure 6. 2nd scheme, B simulation results (a) B1 simulation result, half triangle patterns, (b) B2 simulation result, rectangle patterns, (c) B3 simulation test, quarter triangle patterns, and (d) B4 simulation result, middle patterns

4.3. Scheme 3

Lastly, model B is being tested to read Cuneiform character in its full shape. Based on Figure 7, it can be seen that even the training process on its fragmented shape is quite a struggle, the testing phase shows exceptionally accurate result for reading full shape cuneiform characters, reaching 96.42%. Thus, model B, which is built upon fragments of characters, is sufficient to read full shape cuneiform characters. The same model can be used to read all shapes of Cuneiform characters, whether it is fragmented or not.

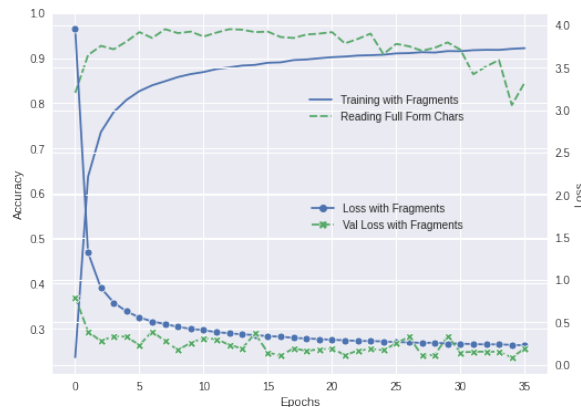


Figure 7. 3rd scheme, full shape cuneiform reading accuracy

5. CONCLUSION

It is actually reasonable to built Cuneiform characters recognition based on fragmented characters dataset. It has advantages for being able to read fragmented and full form cuneiform characters at the same time, thus saving time to train the model. The simple model based on 1 convolution layer with 32 filters is able to differentiate between fragmented and full form cuneiform characters with accuracy reaching 98.73%. It is also can be noted that 4 layers convolution layers reaches 83.86% validation accuracy in reading fragmented Cuneiform characters. This model also able to read full form Cuneiform characters with 96.42% reading accuracy. Model with too much convoluted layers tend to experience overfitting. We can also concur that the most significant part of Cuneiform characters lies on its half vertical and center area. It nearly impossible to read when the character loss 50% of its middle area. Finally, this experiment only conducted on emulated fragmented characters based on Google Noto Sans Cuneiform. Further real fragmented characters dataset from artifacts are needed to build full Cuneiform characters recognition with fuzzy style reading.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to LPPM Institut Teknologi Telkom Purwokerto for their support and generous funding, which has enabled the publication of this paper.





REFERENCES

- [1] M. Van de Mierop, *Cuneiform texts and the writing of history*, London, UK: Routledge, 2005, doi: 10.4324/9780203978375.
- [2] J. J. Glassner, and D. M. Herron, *The invention of cuneiform: writing in Sumer*, Baltimore, Maryland, USA: JHU Press, 2003.
- [3] C. B. F. Walker, *Cuneiform*, Berkeley, CA, USA: University of California Press, 1987, Vol. 3 doi: 10.2307/603965.
- [4] M. Weeden, The Influence of Sumerian on Hittite Literature, in *A Companion to Ancient Near Eastern Languages*, Hoboken, NJ, USA: Wiley, 2020, pp. 505-520, doi: 10.1002/9781119193814.ch27.
- [5] S. Izre'el, and R. Drory (Eds.), *Language and culture in the Near East*, Leiden, Netherlands: Brill, 1995.
- [6] P. Michalowski, Sumerian, in *A Companion to Ancient Near Eastern Languages*, R. Hasselbach-Andee Ed., Hoboken, NJ, USA: Wiley, 2020, pp. 83-105, doi: 10.1002/9781119193814.ch5.
- [7] C. J. Crisostomo, Language, writing, and ideologies in contact: Sumerian and Akkadian in the early second millennium BCE, In *Semitic Languages in Contact*, 1995, pp. 158-180, doi: 10.1163/9789004300156_010.
- [8] T. Jauhainen, H. Jauhainen, T. Alstola, and K. Lindén, Language and dialect identification of cuneiform texts, *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, 2019, doi: 10.18653/v1/w19-1409.
- [9] T. Homburg and C. Chiarcos, Word Segmentation for Akkadian Cuneiform, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016.
- [10] M. López-Nores, J. L. Montero-Fenollós, M. Rodríguez-Sampayo, J. J. Pazos-Arias, S. González-Soutelo, and S. Reboreda-Morillo, CuneiForce: Involving the Crowd in the Annotation of Unread Mesopotamian Cuneiform Tablets Through a Gamified Design, In *Conference on e-Business, e-Services and e-Society*, Cham, Switzerland: Springer, 2020, pp. 158-163, doi: 10.1007/978-3-030-39634-3_14.
- [11] S. E. Anderson, and M. Levoy, Unwrapping and visualizing cuneiform tablets, *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 82-88, 2002, doi: 10.1109/MCG.2002.1046632.
- [12] F. Schroff, D. Kalenichenko, and J. Philbin, FaceNet: A Unified Embedding for Face Recognition and Clustering, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, doi: 10.1109/CVPR.2015.7298682.
- [13] K. P. Ferentinos, Deep learning models for plant disease detection and diagnosis, *Computers and Electronics in Agriculture*, vol. 145, pp. 311-318, 2018, doi: 10.1016/j.compag.2018.01.009.
- [14] Y. Yin, W. Zhang, S. Hong, J. Yang, J. Xiong and G. Gui, Deep Learning-Aided OCR Techniques for Chinese Uppercase Characters in the Application of Internet ChinUp, in *IEEE Access*, vol. 7, pp. 47043-47049, 2019, doi: 10.1109/ACCESS.2019.2909401.
- [15] A. M. S. Rahma, A. A. Saeid and M. J. A. Hussien, Recognize assyrian cuneiform characters by virtual dataset, *2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)*, 2017, pp. 1-7, doi: 10.1109/ICTA.2017.8336049.
- [16] M. Jogin, M. S. Madhulika, G. D. Divya, R. K. Meghana, and S. Apoorva, Feature extraction using convolution neural networks (CNN) and deep learning, In: *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*, 2018, pp. 2319-2323, doi: 10.1109/RTEICT42901.2018.9012507.
- [17] A. A. Barbhuiya, R. K. Karsh, and R. Jain, CNN based feature extraction and classification for sign language, *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 3051-3069, 2021, doi: 10.1007/s11042-020-09829-y.
- [18] O. D. Annesa, Condro Kartiko, and Agi Prasetiadi, Identification of Reptile Species Using Convolutional Neural Networks (CNN), *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)* vol. 4, no. 5, 2020, pp. 899-906, doi: 10.29207/resti.v4i5.2282.
- [19] T. S. Siadari, M. Han, and H. Yoon, 4D effect classification by encoding CNN features, *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 1812-1816, doi: 10.1109/ICIP.2017.8296594.
- [20] L. Alzubaidi et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, *Journal of Big Data*, vol. 8, no. 1 pp. 53, 2021, doi: 10.1186/s40537-021-00444-8.
- [21] T. Septianto, E. Setyati, and J. Santoso, CNN LeNet Model for Year Digit Recognition on Relic Inscriptions of Majapahit Kingdom, *Jurnal Teknologi dan Sistem Komputer*, vol. 6, no. 3, pp. 106-109, Jul, 2018, doi: 10.14710/jtsiskom.6.3.2018.106-109.
- [22] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, *arXiv e-prints*, 2014, doi: 10.48550/arXiv.1409.1556.





- [23] I. Bello *et al.*, Revisiting ResNets: Improved Training and Scaling Strategies, *arXiv a-prints*, 2021, doi: 10.48550/arXiv.2103.07579.
- [24] S. M. Rachmawati, M. A. Paramartha Putra, T. Jun, D. -S. Kim, and J. -M. Lee, Fine-Tuned CNN with Data Augmentation for 3D Printer Fault Detection, *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 902-905, doi: 10.1109/ICTC55196.2022.9952484.
- [25] T. Huang, Q. Zhang, X. Tang, S. Zhao, and X. Lu, A novel fault diagnosis method based on CNN and LSTM and its application in fault diagnosis for complex systems, *Artificial Intelligence Review*, vol. 55, no. 2, pp. 12891315, 2022, doi: 10.1007/s10462-021-09993-z.
- [26] A. Prasetiadi and J. Saputra, pyCuneiform," 2020, [Online]. Available: <https://github.com/agipras/pyCuneiform>.
- [27] Y. Xu, F. Yin, D. Wang, X. Zhang, Z. Zhang, and C. Liu, CASIA-AHCDB: A Large-Scale Chinese Ancient Handwritten Characters Database, *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, 2019, pp. 793-798, doi: 10.1109/ICDAR.2019.00132.
- [28] Y. Seki, Online and Offline Data Collection of Japanese Handwriting, *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, Sydney, Australia, 2019, pp. 13-18, doi: 10.1109/ICDARW.2019.70135.
- [29] O. Russakovsky *et al.*, ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
- [30] Z. Tang, K. Chen, M. Pan, M. Wang, and Z. Song, An Augmentation Strategy for Medical Image Processing Based on Statistical Shape Model and 3D Thin Plate Spline for Deep Learning, in *IEEE Access*, vol. 7, pp. 133111-133121, 2019, doi: 10.1109/ACCESS.2019.2941154.
- [31] S. Shunsuke, C. Kuiting, and B. Takaaki, Tatamikomi Nyuuraru Nettowaaku (CNN) o Mochiita Tegaki Nihongo Moji Ninshiki Shisutemu no Shisaku, *Denki Kankei Gakkai Kyuushuu Shibu Rengou Taikai Kouen Ronbunshuu Heisei 27 Nendo Denki Jouhou Kankei Gakkai Kyuushuu Shibu Rengou Taikai Iinnkai*, 2015, doi: 10.11527/jceek.2015.0_348.

BIOGRAPHIES OF AUTHORS



Agi Prasetiadi     holds a Bachelor of Electronics degree in General Electrical Engineering, VLSI and Computer Vision at Bandung Institute of Technology. He holds a Master of Engineering (M.Eng.) in Realtime Systems, Bio-Inspired Technology at Kumoh National Institute of Technology. He currently teaches at the Purwokerto Telkom Institute of Technology as a lecturer in Informatics. His research areas include computer vision, machine learning, and deep learning. He can be contacted at email: agi@ittelkom-pwt.ac.id.



Julian Saputra     comes from Sampit, currently studying for a Bachelor's Degree in Informatics Engineering at the Telkom Institute of Technology, Purwokerto. Active activities on campus that he has participated in include Lecturer Assistant and practicum assistant for Image and Video Data Analysis course. He joined the campus Institutional organization, namely the Student Council and joined as an Independent Study at the Artificial Intelligence Center Indonesia. His research areas include data science engineer, machine learning, deep learning, natural language processing and education. He can be contacted at email: 19102008@ittelkom-pwt.ac.id.