# Character Recognition for Handwritten Japanese Katakana using Deep Learning Techniques

**Lena Godosar**
Institute for Language and Information
Heinrich Heine University Düsseldorf
`Lena.Godosar@hhu.de`

## Abstract

Automated identification of characters from images is a well known field in natural language processing. This project deals with handwritten character recognition using Deep Learning methods. The task is performed on Japanese katakana from the ETL-1 dataset. Using the presented model (a deep neural network with convolutional layers), a final accuracy of ∼90% is achieved. The results will be evaluated and discussed by looking at the confusion matrix.

## 1 Introduction

Character recognition is a common task in natural language processing. It has wide applications in areas like license plate recognition, document scanning, and many more. The project described in this paper deals with handwritten characters. This is particularly challenging due to the differences in writing style among people and overall variations based on human imperfections. With the surge in populartiy of Machine Learning in recent years, many advances in the field have been made. Especially Deep Learning models turn out to be very powerful classifiers that can achieve a high level of performance.

In this paper I present a Deep Learning-based approach to classifying handwritten Japanese katakana using a convolutional neural network. The network was implented using PyTorch (Paszke et al., 2019).

The paper at hand starts with a description of the dataset that was used, and the preprocessing stage. Then, the model is introduced and the training process is explained. Finally, the results, including the achieved accuracy, are presented, discussed, and evaluated.

| L | Char | Trans | L | Char | Trans |
|---|---|---|---|---|---|
| 0 | ア | a | 24 | ノ | no |
| 1 | イ | i | 25 | ハ | ha |
| 2 | ウ | u | 26 | ヒ | hi |
| 3 | エ | e | 27 | フ | fu |
| 4 | オ | o | 28 | ヘ | he |
| 5 | カ | ka | 29 | ホ | ho |
| 6 | キ | ki | 30 | マ | ma |
| 7 | ク | ku | 31 | ミ | mi |
| 8 | ケ | ke | 32 | ム | mu |
| 9 | コ | ko | 33 | メ | me |
| 10 | サ | sa | 34 | モ | mo |
| 11 | シ | shi | 35 | ヤ | ya |
| 12 | ス | su | 36 | ユ | yu |
| 13 | セ | se | 37 | ヨ | yo |
| 14 | ソ | so | 38 | ラ | ra |
| 15 | タ | ta | 39 | リ | ri |
| 16 | チ | chi | 40 | ル | ru |
| 17 | ツ | tsu | 41 | レ | re |
| 18 | テ | te | 42 | ロ | ro |
| 19 | ト | to | 43 | ワ | wa |
| 20 | ナ | na | 44 | ヰ | wi |
| 21 | ニ | ni | 45 | ヱ | we |
| 22 | ヌ | nu | 46 | ヲ | wo |
| 23 | ネ | ne | 47 | ン | n |

Table 1: List of all katakana with their respective labels and transcription.

## 2 Data

### 2.1 Dataset

Modern Japanese is written in a mixture of three scripts: *kanji*, a morphographic script mostly derived from Chinese, and *kana*, which are further divided into the two syllabaries *hiragana* and *katakana* (Smith, 1996). The latter is commonly used for emphasis and transliteration of foreign names and words, like マリオ (”Mario”) or ホテル (’hoteru’, ”hotel”). It is also used to write onomatopoeic words, e.g. コンコン ('konkon', "knock-knock").

A full list of all katakana and their roman transcriptions can be seen in table 1. Note that in modern Japanese, two of the katakana (ヰ 'wi' and ヱ 'we') are pronounced as vowels, making them obsolete. They are included in this project nonetheless.

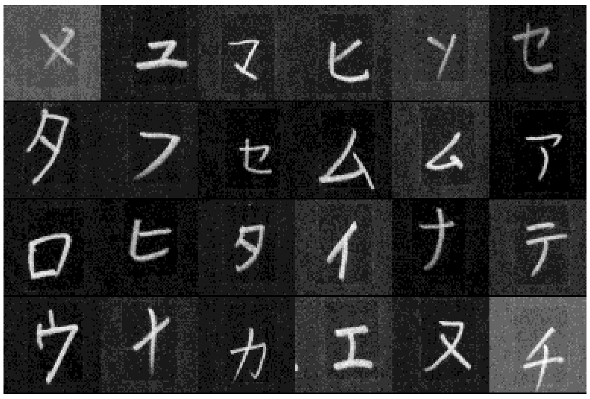


Figure 1: Some examples of what the data looks like.

The dataset used is the ETL-1 dataset, which is part of the ETL character database (Electrotechnical Laboratory, 1973-1984). It was collected and supplied under cooperation with various japanese institutes, universities and other research organisations. It contains, among other characters, a set of 51 (48 unique) katakana from 1411 writers, resulting in 71961 entries overall. Each character is represented by an image with a resolution of 63x64 pixels. Some examples of the raw data are shown in figure 1.

Reading the data is done in multiple steps. The ETL database provides an unpack script, which reads the raw files and produces a series of images. Those are further processed and split into individual entries.

### 2.2 Preprocessing

First of all it is to note that three of the 51 categories are actually duplicates of other characters that are already contained in the data. That is because the syllables 'yi', 'ye', and 'wu' are not used in modern Japanese and thus do not have corresponding katakana. In the dataset, the missing spots are filled with duplicates of イ, エ, and ウ ('i', 'e', and 'u'). They are removed for that reason, resulting in 48 unique katakana remaining.

Because the original images have a pixel dimension of 63x64, an empty row is added to get a square shape of 64x64, which is more convenient to handle. Since this is the same for every single entry in the dataset, no information is added or subtracted, so it should not impact the network at all.

Furthermore, two individual characters were completely missing (one instance of each ナ 'na' and リ 'ri'). To keep the balance between classes, and for the ease of data handling, they are replaced by inserting a copy of another character from the same class. This is expected to only have a negligible impact on the performance.

These steps result in final data specifications of 48 katakana with exactly 1411 entries each (overall 67728 samples), consisting of 64x64 pixels.

Labels are assigned to every katakana, as shown in table 1.

The data is futher preprocessed by dividing the values by their maximum, ensuring that they are in [0,1], and splitting the dataset into three distinct subsets: a train-, dev-, and test set with 900, 100 and 411 entires per class respectively. The dev set provides a validation during the training process and the test set is what the final model is evaluated on. This is done in such a way that each category is equally prevalent in each of the subsets. E.g. there are exactly 900 instances of any given katakana in the train set. This guarantees an even distribution of every class in the data, when it is presented to the network.

## 3 Model description and training

The model is a deep convolutional neural network. Its exact architecture is as follows: It has 3 convolutional layers with a kernel size of 3 and zero padding that keeps the original tensor shape. They result in 32, 64, and 128 feature layers. Each is followed by max pooling with kernel sizes of 2, 4, and 4 respectively. After that, the input is flattened, resulting in 512 nodes. An additional dense layer with 128 nodes is added. Both of the last two (flat) layers have a dropout of 0.1 applied to them. Finally, there is an output layer with 48 nodes, each representing one katakana. After every layer, a ReLU activation function is applied. The output layer uses a softmax activation, which is commonly used for classification problems. It normalises the outputs in such a way, that they can be interpreted as probabilities of a given input belonging to the respective class. Overall, this results in 164528 trainable model parameters.

The training is performed using the Adam optimiser, with a learning rate of $5 \cdot 10^{-4}$. The network

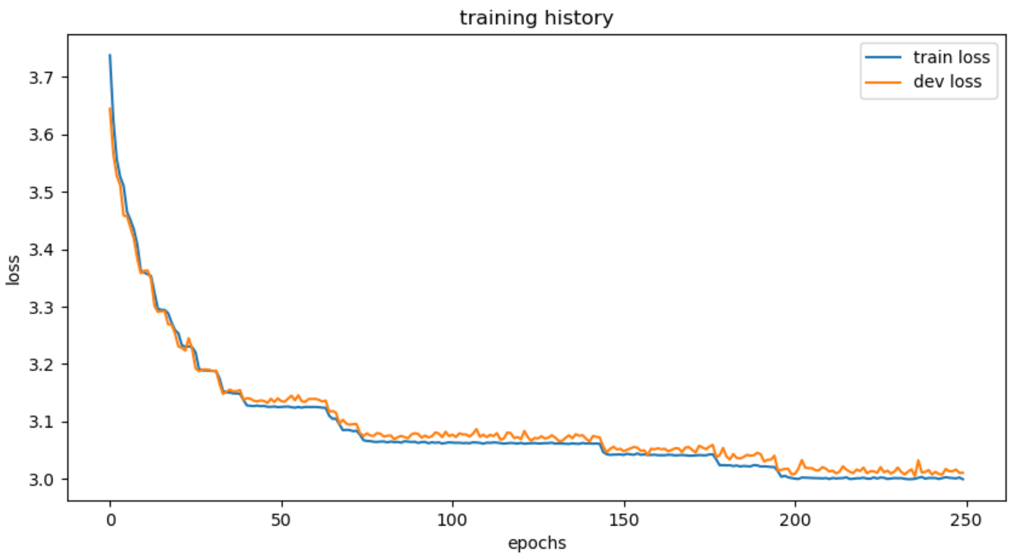


Figure 2: The evolution of loss over epoch.

is trained for 250 epochs with a batch size of 32, while using cross-entropy loss.

Figure 2 shows the development of train- and dev loss for each epoch of training. It can be seen that the loss is continually decreasing. Also, the dev loss is close to the train loss, which implies that there is no overtraining. It is also apparent that the network gets stuck temporarily in multiple local minima, indicated by the plateaus in the loss history.

## 4 Results and evaluation

A final accuracy of 89.96% is achieved on the train set with the model described above (section 3).

During development, various modifications were tested, and the most successful model is the one discussed before. Those experiments and their outcomes included the following. Models with less convolutional layers seem to be unable to capture the whole complexity of the input, and performed worse. Networks with more convolutional layers similarly did not reach the same accuracy. Consequently, 3 layers were chosen as optimal. Adopting a higher learning rate lead to strong oscillations around the minimum. The current setting seems to strike a good balance between escaping local minima and not fluctuating too strongly. Adding the extra dense layer after flattening improved the performance tremendously.

We can gather more information about the model's performance from the confusion matrix (figure 3). It shows the frequency of any predicted label for each true label, allowing us to see which characters were identified as which katakana. A high score on the diagonal corresponds to many correctly classified characters, while any off-diagonal entries mean that some characters were misidentified.

We observe that most predicted lables are correct, which is compatible with the accuracy of ∼90% presented earlier. Only four categories are dominantly classified wrong. We can understand why by looking at those cases in more detail. The most striking example is class 18 (テ 'te'), which gets classified as 16 (チ 'chi') with a very high probability. The same is true for 22 (ヌ 'nu') and 12 (ス 'su'). The katakana with the lables 38 (ラ 'ra') and 4 (オ 'o') are also not correctly identified, but with larger uncertainties. The former is predicted as 46 (ヲ 'wo') for over 50% of the cases. The latter's top two predictions are almost equally the classes 29 (ホ 'ho') and 20 (ナ 'na'). Figure 4 shows examples from all misclassified categories. We can see that the katakana which are consistently confused by the network actually look very similar. They share a lot of the same features, with only minor differences. Taking the human imperfections and inconsistencies into consideration, it is no suprise that the model has difficulties distinguishing them.

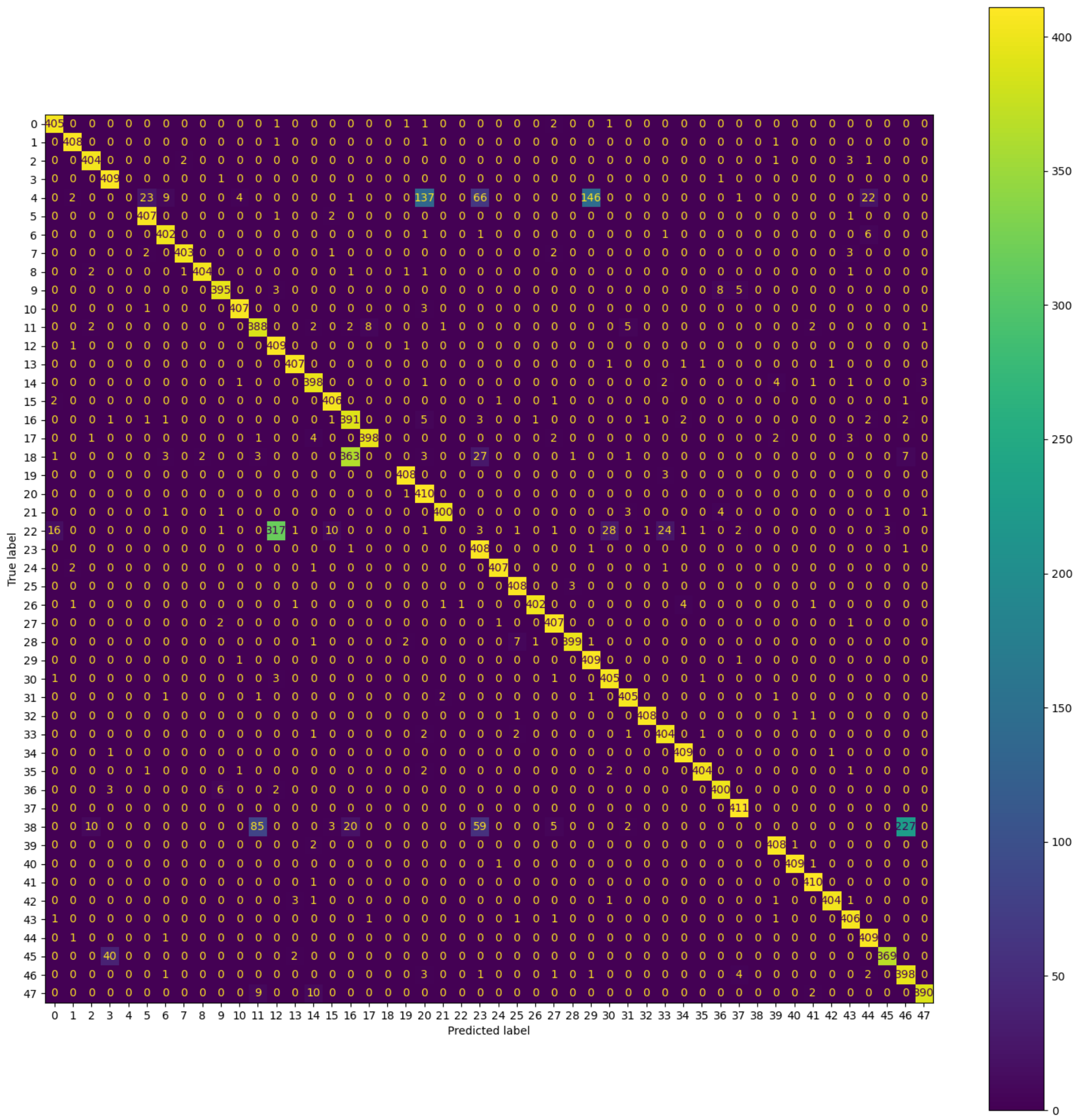
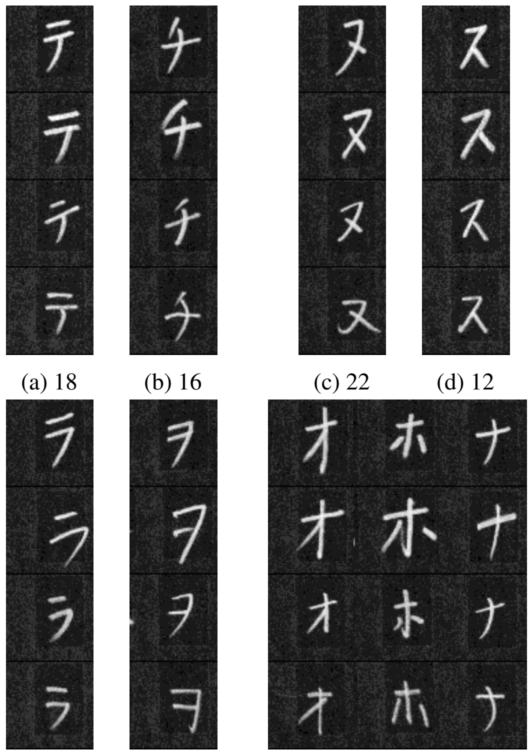


Figure 3: The confusion matrix. Elements on the diagonal correspond to correctly identified characters.


(a) 18 (b) 16 (c) 22 (d) 12

(e) 38 (f) 46 (g) 4 (h) 29 (i) 20

Figure 4: Examples of katakana that are often misidentified. (a) is mostly classified as (b), (c) as (d), (e) frequently as (f), and (g) is equally classified as (h) and (i). The captions represent the labels, as shown in table 1.

## 5 Conclusion

Using Deep Learning techniques I was able to train a classifier to distinguish handwritten katakana. The training and evaluation worked as intended and the network achieved a performance of ∼90%. Further investigations have revealed the limits of the model, namely the distinction of optically similar characters. By utilising more advanced techniques and systematic experiments, one could probably increase the accuracy even further. Considering the scope of this project and technical limitations however, the results are very satisfactory.

## References

Electrotechnical Laboratory. 1973-1984. ETL Character Database. http://etlcdb.db.aist.go.jp/. Accessed: April 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Janet S. Smith. 1996. Japanese writing. In Peter T. Daniels and William Bright, editors, *The world's writing systems*, pages 209–217. Oxford University Press, New York.