CSC412 Project: Exploiting Structure For Classification Of Handwritten Japanese Characters

Robert Balayan

Abstract

Kanji is one of three writing systems used in Japanese language, each character of which represents a word or a concept. All Kanji are constructed using simple characters called radicals. Historically, Kanji classification has been performed directly from image to Kanji. However, there exists a unique mapping from Kanji to radicals. This paper focuses on comparing direct classification of Kanji characters to multi-label classification of radicals using four models of different complexity. Artificial Neural Networks of different scales were tested, ranging for simple linear regression to a Convolutional Neural Network with 7 layers. Direct classification consistently outperformed radical classification, peaking at 94.43% classification accuracy on the test set, versus 37.64% accuracy of its counterpart.

1 Introduction

Kanji is one of three scripts used in Japanese writing, others being *Hiragana* and *Katakana*, which together are called *Kana*. While kana characters are used to represent different sounds, Kanji characters represent complete words or concepts. Kanji characters were brought to Japan from China, but over time they were used less and less. In modern Japanese, only 2,136 *Jōyō* Kanji are used officially. All Kanji are constructed using radicals, smaller and simpler characters that usually have a meaning related to the resulting word. This paper will compare direct classification of Kanji characters to multi-label classification using radicals.

2 Understanding Kanji Structure

As mentioned before, Kanji are constructed using radicals. In addition to that, Kanji can be constructed using other Kanji. On Figure 1 you can see Kanji 時 (time) and its breakdown to radicals. On the first sub-figure 1a is the original character with nothing highlighted. The next sub-figure 1b there are two components highlighted - Kanji 日 (day) and 寺 (temple). These two Kanji have historic connection to the word 'time', as temples used to announce the time. These connections are sometimes very obscure and antiquated, but knowing the meaning of the radicals can often hint on the meaning of the word. 寺 can be broken down further into radicals 土 (earth) and 寸 (measure of distance, but radical is generally used in sense of sticking or gluing), which are highlighted on Figure 1c. Together, all these components can be used to identify the character.

Some radicals can appear in different positions or multiple times within a Kanji, which raises a question of uniqueness of radical to kanji mapping. A basic equivalence test on the whole dataset of Chinese characters showed that there exist cases where multiple characters are constructed using the same radicals (i.e. 𠃌 and 𠄌 are constructed with 口 and 𠂉, but 𠄌 uses one of the radicals multiple times), but limiting the character set to the list of *Jōyō* Kanji resulted in zero matches. Thus, it is safe to use radicals to uniquely identify Kanji characters.

Traditionally there are 214 Kangxi radicals that are used to write Kanji characters. However, the dictionary revealed 21 additional radicals used to write *Jōyō* Kanji, totalling in 235 radicals. It is possible that uniqueness is a result of these extra radicals.

As with most elements of natural languages, radicals follow Zipf's law. Their distribution can be seen on Figure 2. Frequency axis denotes the number of Kanji featuring each radical. As you can see on the graph, about four fifths of the radicals are featured in less than 50 Kanji. The most common radical appears almost 400 times more often than the least common one.

Some radicals are somewhat similar. It is a concern whether the models will be able to differentiate radical '一' (one) from '二' (two) or a line in '口' (mouth, box). There are many other radicals that look

(a) Original Kanji

(b) Breakdown to simpler Kanji 日 and 寺

(c) Breakdown to the simplest components - radicals 日, 土 and 寸

Figure 1: Breakdown of Kanji 時

somewhat similar or contain one another in some way. This will be a challenge that may hinder performance of simpler models that don't use convolutions.

3 Related work

Because Kanji are almost a proper subset of Chinese characters (some characters were modified differently in modern Chinese and Japanese), research into their classification is relevant to this problem. In 2015 Fujitsu has achieved an accuracy rate of 96.7%, surpassing the human equivalent recognition rate of 96.1% for classification of Chinese characters [2]. A system similar to radical classification was designed in 1982 for on-line Chinese/Kanji character classification by Greanias, E.C. and Yhap, E.F. [3]. Their system, however, used proprietary set of radicals that they claim can be used to generate all characters. There is a similar stroke-based on-line classifier based on Hidden Markov Models developed by M. Nakai, N. Akira, H. Shimodaira and S. Sagayama that was published by IEEE [5]. A paper on classification of Japanese scriptures titled "Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks" was published by Charlie Tsai of Stanford University [6] that compared classification of all three writing systems using convolutional neural nets of different complexity, ranging from a 6 layer CNN with 4 convolutions to a 16-layer CNN. Tsai reported classification accuracy of 99.64% achieved on test set by a model that will be explored later in the paper. Tsai's paper mainly focused on classification of Kana scriptures, which he believes is an under-researched field.

4 Data

Multiple datasets were used in this project:

1. Images of handwritten Kanji characters were obtained from the Electrotechnical Laboratory (ETL) Character Database, which has been collected by National Institute of Advanced Industrial Science and Technology (AIST). Specifically, dataset ETL9G was used, which in total contains 421,200 relevant samples. Only 2106 out of 2136 *Jōyō* Kanji were available because the dataset was collected between '84 and '03, while the missing Kanji were added to the list of *Jōyō* Kanji in 2010. For each Kanji there are 200 127x128 images provided.
   The dataset is stored as 50 files, each file containing 4 images per Kanji.

2. The list of conversion from JIS X 0208 encoding to Unicode characters was obtained from charset.7jp.net - website specialized in collecting tables for different Japanese encodings.

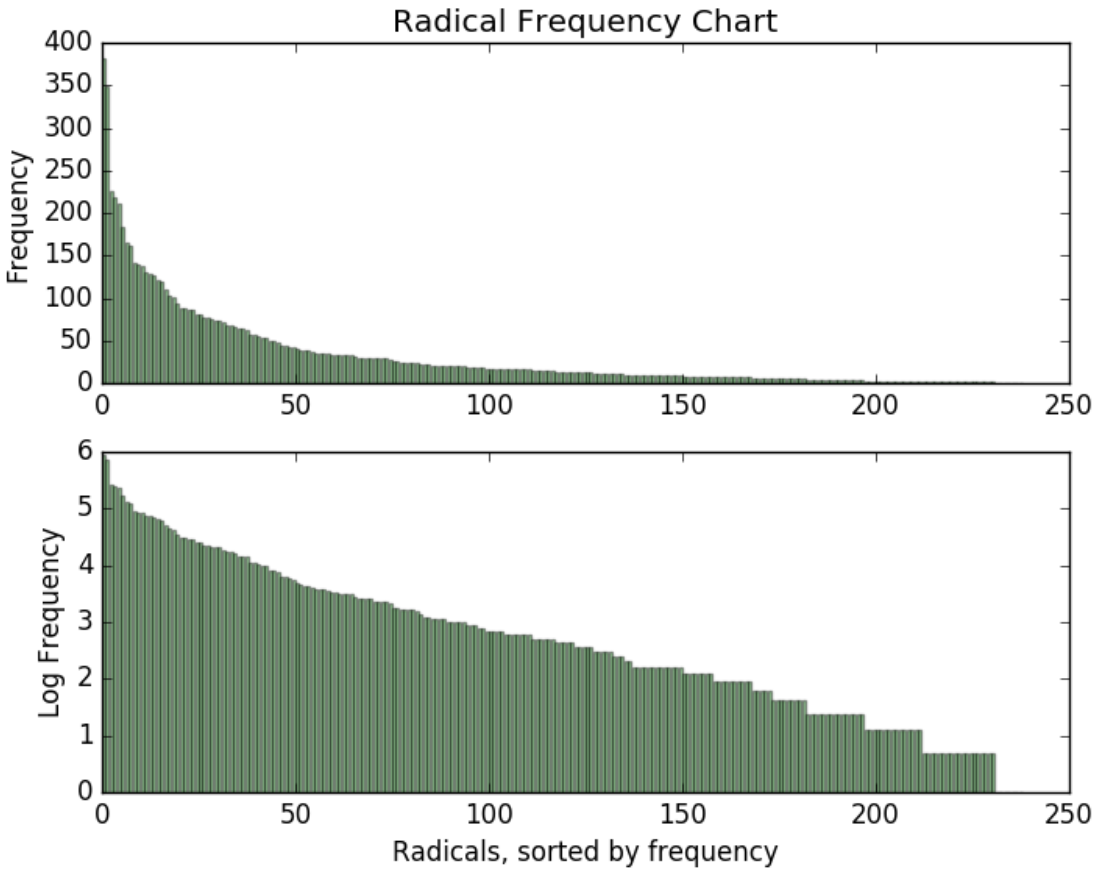


Figure 2: Radical frequency diagram

3. Mappings from Kanji to radicals and vice versa were obtained from RADKFILE and KRADFILE databases.

4. The list of *Jōyō* Kanji was obtained from the Agency for Cultural Affairs, maintained by Japan's Ministry of Education, Culture, Sports, Science and Technology.

Below on Figure 3 are sample images from ETL9G dataset. As you can see, not all samples are perfectly sliced (fig 3a) and there might be noise on some samples (fig 3b). One's handwriting can really affect the look of the Kanji. An example of that can be shown with Kanji 鋭 on Figure 4, on sample 4b the box radical 口 is distorted so much that it looks like a triangle. The radical below it, 儿 (legs), is joint in a way that makes it look like another radical, for example 入. Some other examples of this Kanji can be seen on figures 4c and 4d. Again, there are some overlapping characters on these samples.

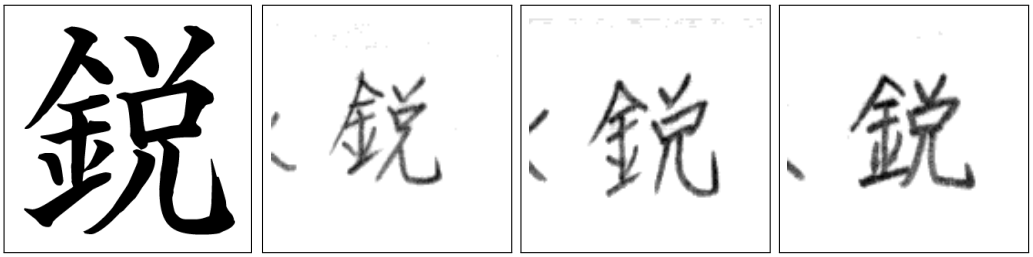
5 Methods

5.1 Pre-processing

First, redundant data was removed from all dictionaries, like non-*Jōyō* Kanji and radicals used only in those Kanji. This resulted in reducing the number of radicals from 252 originally in the dictionaries to 238.

(a) Overlapping sample (b) Sample with noise in the background (c) Sample with noise on the writing

Figure 3: Various deformed samples from ETL9G dataset



(a) Kanji 鋭 (b) Messy handwriting (c) Example handwriting (d) Example handwriting

Figure 4: Different handwriting examples of Kanji 鋭

Furthermore, some Kanji referenced themselves as their radicals in addition to their actual radicals, but were never used as radicals in any other Kanji. These 3 Kanji (鼓, 香 and 鼻) were removed from the list of radicals to simplify classification. Using the resulting Kanji-to-radical mapping, all Kanji were encoded using categorical format, where each character is represented by a vector with 1's in the class(es) it belongs to and 0's in the rest. Then the dataset was split into training, validation and testing sets. For direct image-to-Kanji classification 50 or 100 images per Kanji were used to train the models (105300 or 210600 total samples), then 10 images per Kanji were used in each validation and testing sets (21060 samples in each set). For radical classification much smaller sets were needed, because there are less classes and each image belonged to multiple classes. Please see Section 5.3 below for specifics. Testing and validation sets for this model contained 4 images per kanji.

5.2 Classification

With each image being 127x128, direct mapping from image to Kanji becomes a very hard problem to train, because the mapping itself is dealing with very large spaces (16256 dimensions to 2106). Classifying Kanji by guessing the radicals is a problem dealing with a much smaller space (16256 dimensions to 235), but a mistake labeling a single radical will result in complete miss-labeling of the input image. Direct classification is the equivalence of classifying a whole English word at once instead of recognizing letters separately: while classifying the whole word is a less ambiguous problem, the search space is much greater than simply classifying the letters.

Four models will be discussed and compared to a baseline set by direct logistic regression:

1. A shallow Neural Network with a single hidden layer of 235 layers.

2. A Neural Network with three fully-connected (FC) hidden layers with 5000 hidden units in each.
3. A shallow Convolutional Neural Network with four convolutional layers followed by a FC layer with 256 units.
4. A deeper Convolutional Neural Network with four convolutional layers followed by two FC layers with 4096 hidden units.

The last model is based on the M7-1 model used by Charlie Tsai in his paper: M7-1 was the architecture with best performance on the validation set of Kanji (99.55%) and second best performance on the test set (99.55% again, best result was 99.64%).
All convolutional layers feature small 3x3 kernels, number of which differs in each model. Some convolutional layers are followed by max pooling layers with 2x2 kernels. Specific layer breakdown can be seen on table 1. REctified Linear Unit (RELU) activation function, defined as $max(0, X)$, was used after each layer. A version of Model 2 with L2 regularization was also tested, with 0.1 weight coefficient.

<table>
<tr><th>Model 1</th><th>Model 2</th><th>Model 3</th><th>Model 4</th></tr>
<tr><td colspan="4">Input (127x128 Greyscale Image)</td></tr>
<tr><td rowspan="8"></td><td rowspan="8"></td><td>conv3 x 32
conv3 x 32</td><td>conv3 x 64</td></tr>
<tr><td colspan="2">maxpool</td></tr>
<tr><td>conv3 x 64
conv3 x 64</td><td>conv3 x 128</td></tr>
<tr><td colspan="2">maxpool</td></tr>
<tr><td rowspan="2"></td><td>conv3 x 512
conv3 x 512</td></tr>
<tr><td>maxpool</td></tr>
<tr><td>FC-235</td><td>FC-5000
FC-5000
FC-5000</td><td>FC-256</td><td>FC-4096
FC-4096</td></tr>
<tr><td colspan="4">FC-Classes</td></tr>
</table>

Table 1: Neural Net Configurations

5.3 Sampling

Because of Zipfian distribution, using the dataset for training on radicals as-is would overfit on some radicals while still being very untrained on others, since there are Kanji that are radicals of themselves and don't appear anywhere else. To ensure that the models are well trained on the more common radicals, but still know how the less common ones look like, a minimum threshold on the number of the radicals was set. All uncommon radicals are used in conjunction with the most common radicals, so setting a minimum of 200 images per radical ensures that there are enough samples for uncommon ones, but still allows for a possibility of overfitting on the more common ones (there are 11 radicals that appear over 2000 times in a dataset of 32,000 samples). So, the Zipfian distribution still holds, but the difference is much smaller: before scaling most common radical appeared 400 times more often than the least common, with scaling applied it appears only 20 times more often. The resulting distribution can be seen on the Figure 5 below. Setting the

threshold lower keeps the proportion at the same level. Because some radicals are very common, there are many Kanji characters that weren't used to train the model. In fact, only the characters that had unique radicals were included 200 times, other characters had far less samples, many not being needed at all. This allows for a possibility of a radical appearing in a position unseen during training.

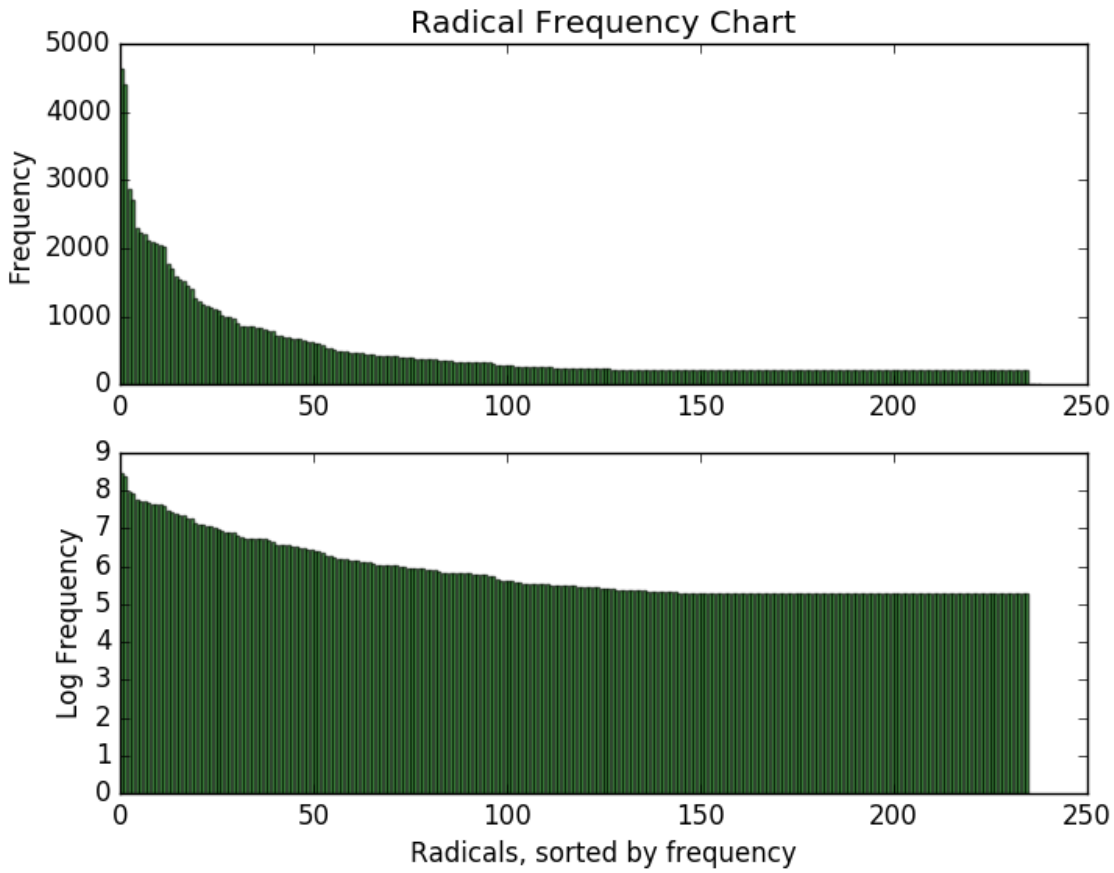


Figure 5: Radical frequency after setting minimal threshold to 200 images per radical

5.4 Training

For direct classification softmax was used as the final activation layer with categorical cross entropy as the loss function, defined in equation 1 below:

$$softmax(X) = \frac{e^X}{\sum_{x \in X} e^x}$$

$$Cat(X, Y) = -\sum Y * log(softmax(X)) \quad (1)$$

For radical classification sigmoid activation layer was used as the input for binary cross entropy, defined in equation 2 below. Binary cross entropy was chosen over mean absolute error and mean squared error after

training linear regression and Model 2 for 40 epochs with each function.

$$sigmoid(X) = \frac{1}{1 + e^{-X}}$$

$$Bin(X, Y) = Y * -log(sigmoid(X)) + (1 - Y) * -log(1 - sigmoid(X)) \quad (2)$$

All models were trained using ADAM optimizer with default parameters described in the paper [4].

Learning rate differed depending on the model, ranging from 10^{-4} to 10^{-6}. For example, Model 2 showed no signs of training with learning rate of 10^{-4}, but simpler models worked perfectly fine. Simpler models converged to the same value using even learning rate of 10^{-3}, while more complex models required at least 10^{-4}.

Various batch sizes were tested in different orders of magnitude, from 16 to 15,000. Batches of size 500 have proven to be the most effective, resulting in best performance over 5 epochs.

All models were trained for at least 60 epochs, with simpler models being trained for as many as 400 epochs. An exception to this was Model 4 for direct classification, which was trained for only 24 epochs due to time constraints.

All models were implemented using Keras wrapper for Theano [1]. Training was done on GPU using p2.xlarge instances of Amazon Web Services EC2 servers, except for Model 4, which was trained on CPU using c4.8xlarge instances due to its size. The compiled model was unable to fit on Nvidia Tesla K80 GPU, so training had to be done on CPU.

5.5 Testing

5.5.1 Validation and testing datasets

While there is a lot of data available in the ETL9G dataset (200 images per character), very little of it was used for validation and testing. For direct classification, validation and testing sets each contained 10 images per Kanji. This amounts to 10% of the training set each, with the rest of the dataset unused. This kept the sizes manageable while still being enough to gauge the performance. For radical classification less data was used. Because training was done on radicals, but performance is measured on Kanji, no sub-sampling was applied to the testing and validation sets. Instead, each set contained 8 samples of each Kanji, totaling at 16848 samples each. This meant that both training and validation sets each were almost 50% of the training set in terms of size. On a per-radical basis, however, the relation is very different and varies a lot.

5.5.2 Accuracy

For direct classification prediction was considered correct of the maximum value of the softmax layer matched that of the input's one-hot encoding. Then, accuracy was the number of correct predictions over all samples.

For radical classification a similar metric was used. Because by default binary cross entropy is interested in classifying each label separately, its accuracy is not representative of the models actual ability to classify Kanji characters: it could predict that there are no radicals on any image and have accuracy in high 90's. Because of that, accuracy was measured to be the number of samples with each label classified correctly over the total number of samples tested. Specifically, mean absolute error on rounded predictions was used, which is equal to 0 if every label is classified correctly. This metric has also proven to be representative of when a model actually starts learning: some models showed 0% accuracy for as many as 120 epochs, but when this accuracy started going up, it continued until the model plateaued.

6 Results and discussion

Linear regression didn't perform well for neither direct classification nor classification through radicals even after 400 epochs. The former topped with just over 3% accuracy on the training set and a percent lower on validation and testing. The latter was unable to classify even a single percent of the data, maxing out at 0.024%. This is a very low baseline for other models to beat.

The rest of the models followed a similar pattern: overfitting everywhere. Other than Models 3 and 4, all models reached reasonably high performance on the training set and a fraction of that on the other sets. Model 2 for direct classification, for example, reached 100% accuracy on the training set, but on the other sets accuracy was little over 20%. This model model was also tested with L2 regularization, which showed mixed results. Direct classification was improved by a lot. Even though the model plateaued after 100 epochs, accuracy rose to 72.15% on validation set and 65.36% on the training set. Radical classification fell to 0%, unable to beat the baseline even after 650 epochs. The model was unable to train with different learning rates (10^{-4}, 10^{-5} and 10^{-6} were tested) and smaller batches either. It is possible that such high regularization limited the model's expressiveness, much needed for a target of that complexity.

Model 1 was created to see if it is possible to learn radicals by direct classification, but the model showed poor results and the weights looked like white noise with nightmarish lines imitating Kanji.

Only Convolutional Neural Networks for direct classification had real success. These models were trained on only 50 images per class. Even with such small training set, the small CNN showed 95.57% classification accuracy on the validation set and 94.43% on the testing set, which is the closest any of the models has gotten to the human classification rate. With more training it's possible that it would surpass the human recognition rate, the model hasn't converged after 100 epochs and was still showing learning when it had to be terminated. Interestingly, its accuracy on the training set was still very low, barely reaching over 40%. When the model was trained again (training data from the previous session was lost), it was unable to reach the same numbers as quickly as before, but the accuracy was still growing steadily after 100 epochs. This newly-trained model achieved 83.44% on validation set and 78.31% on the testing set. As before, accuracy on the training set is significantly lower, topping at 35.38%. Interestingly, the model's performance on validation set grows very fast during the first 50 epochs and slows down during the remaining 50 epochs. On the training set the accuracy grows in a more linear fashion. You can see the whole training history on Figure 6 below.

The large CNN was showing a lot of promise, but its training had to be aborted due to time constraint. With each epoch taking nearly 2 hours, the net was trained for only 24 epochs and had already achieved an incredible 88.19% accuracy on validation set. Again, the model showed a lower accuracy (64.69%) on the training set. Due to implementation semantics, it was impossible to run the testing set through the model and get the accuracy. Still, the model showed consistent increase in accuracy.

The models' radical counterparts, did not perform well at all. Again, this is textbook overfitting: high performance on the training data, low performance on the validation data.

Overall, all models should have been using some sort of regularization and normalization, but overfitting started showing when it was too late to change everything. This was the pitfall for most models, resulting in a gigantic difference between accuracy on the training set and the testing and validation sets. Convolutional networks trumped over regular neural networks and every direct model outperformed its radical counterpart.

Please refer to Table 2 for accuracy of each model on each set.

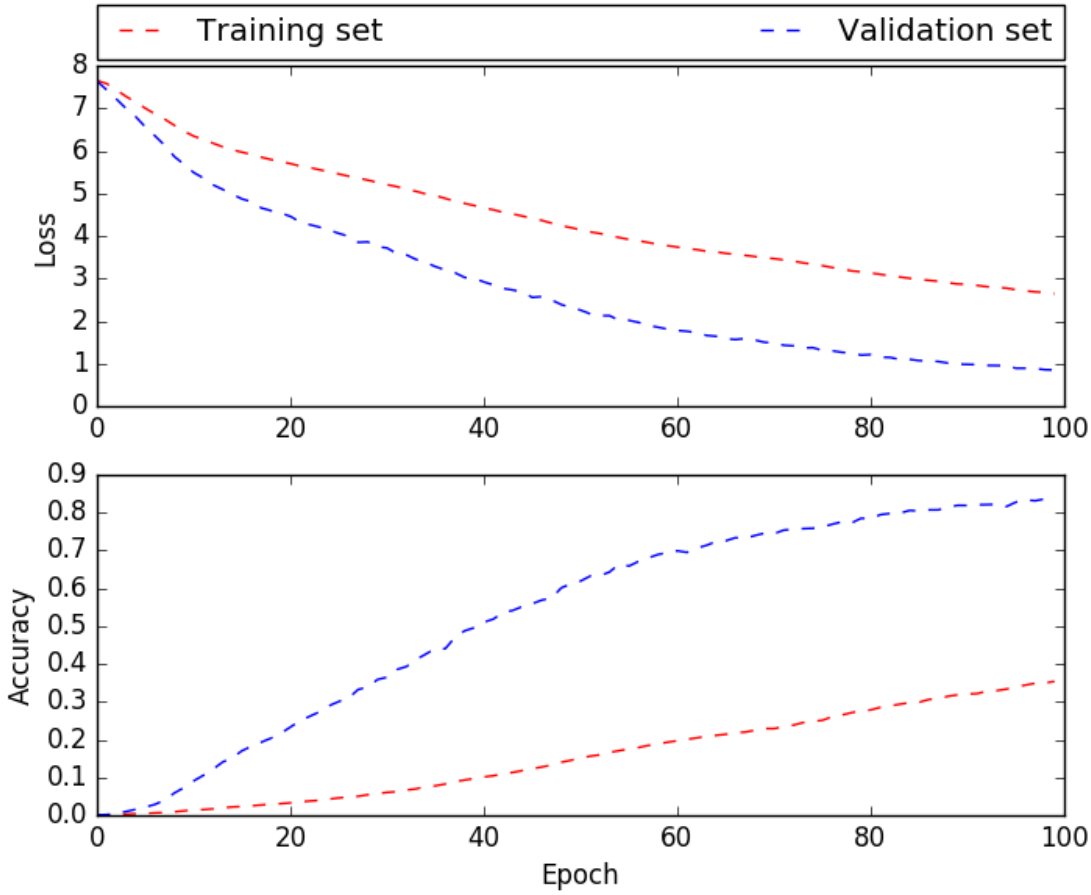


Figure 6: Training history of Model 3 with direct targets

7 Conclusion

In this paper we have explored a viability of classification of Japanese Kanji characters via radicals, their building blocks. While the unique mapping exists and theoretically classifying radicals is a more sensible approach, none of the models tested were able to classify radicals accurately enough to achieve level close to that of their direct classification counterparts, let alone human classification rate. Most models tested showed signs of overfitting the training data. Attempts to reduce its effects showed mixed results by boosting performance of direct model and completely nullifying performance of radical classification.

Earlier an analogy was made about classifying Kanji as classifying complete English words. This paper showed that despite this being a less elegant solution to the problem, 2106 unambiguous characters is still doable even with a simple Convolutional Neural Network. Less complex models were unable to show anything interesting and more complex models take a lot of time to train even with a smaller training dataset. One advantage radical classification has over direct classification is complexity of the mapping. With almost 10 times less classes, each training step took fraction of time used for direct classification. The best example of this is Model 2. Both direct and radical classifications started training at the same time on same configurations, both trained on GPU with the same parameters. By the time the direct model completed its 150th epoch, radical classification was already over 600 epochs in. Unfortunately, that particular model did not show good results, but this is still a good example of how much harder it is to train

	Kanji			Radicals		
	Training	Validation	Test	Training	Validation	Test
Baseline	3.37%	2.35%	2.56%	0.31%	0.02%	0.02%
Model 1	89.5%	19.43%	18.74%	83.24%	0.3%	0.3%
Model 2	100%	21.13%	25.66%	80.58%	0.39%	0.33%
Model 2 + L2	84.69%	72.15%	65.36%	0%	0%	0%
Model 3	78.94%	95.56%	94.43%	48.77%	2.69%	3.27%
Model 4	64.69%	88.19%	N/A	90.38%	33.71%	37.64%

Table 2: Classification accuracy of all models

a complex function with 10 times more classes.

There is still room for more research on this subject. One direction for improvement is restricting the number of radicals to the official set of 214 Kangxi radicals or a custom set of characters like Greanias, E.C. and Yhap, E.F. have done for their model. There is also room for improvement with model architecture. Model 3 has shown that even small models can achieve high classification rate. Building models more tailored for this problem can improve performance. Using binarized dataset can simplify learning and classification. It was too late to add regularization to these models and properly test them, but there is definitely room for improvement in that direction as well.

References

[1] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[2] Ltd. / Fujitsu Laboratories Ltd. Fujitsu R&D Center Co. Fujitsu achieves 96.7characters using ai that mimics the human brain. *Fujitsu Laboratories Ltd. Press Releases*, 2015.

[3] E.C. Greanias and E.F. Yhap. Chinese/kanji on-line recognition system, December 21 1982. US Patent 4,365,235.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama. Substroke approach to hmm-based on-line kanji handwriting recognition. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 491–495, 2001.

[6] Charlie Tsai. Recognizing handwritten japanese characters using deep convolutional neural networks. *N/A, CS231n: Convolutional Neural Networks for Visual Recognition*, 2016. Paper was a submission for a project for CS231n: Convolutional Neural Networks for Visual Recognition at Stanford University.