Im-ROCKET: A Proof-of-Concept Framework for Adapting ROCKET Methodology to 2D Image Classification

Titania Ervin
George Mason University
Fairfax, VA 22030
tervin3@gmu.edu

Abstract

In the domain of 2D and 3D (2D + 3 color channels) image identification, Convolutional Neural Networks (CNNs) dominate in any methodology where feature extraction is required. The use of a CNN in any given classification pipeline is so widespread that contemporary research is often focused on the model proceeding the feature extraction step, and the CNN itself is rarely touched or modified aside from the configuration of the convolutional and pooling layers. In the domain of 1D time series identification, a CNN derived methodology known as ROCKET has existed for some time. The ROCKET methodology excels over CNNs in model training time due to its use of randomized convolutional kernels, whose kernel's values require no training or backpropagation from proceeding classifier. In this paper, we propose the proof-of-concept methodology "Im-ROCKET", which uses a self-intersecting curve to transform 2-dimensional single-channel image data into the 1-dimensional time-series domain for use with the MultiROCKET feature extractor, with mild success.

1 Introduction

In gray-scale and RGB 2D image classification, Convolutional Neural Networks (CNNs) have been leveraged as a powerful feature extraction tool. With such an effective method for extracting features, researchers can focus on engineering effective classifiers that act on those features. The domain of Kanji identification is one such area where CNNs dominate as a feature extractor, with proceeding classification techniques that vary greatly in contemporary literature. One underexplored area of Kanji identification, and image identification in general, is the usage and development of other feature extraction techniques or techniques that greatly modify the commonly used CNN architecture.

Naturally, it is easy to understand why the CNN dominates as a feature extractor, allowing contemporary classification models to achieve extremely high classification accuracy, even in scenarios with high class counts and imbalanced datasets. However, exploring other techniques for feature extraction may still be useful, even if it does not disrupt the status quo domination of the CNN. Exploration away from the CNN or towards modified CNN architectures can lead to further understanding of feature extraction techniques, better understanding of the underlying classification problem being explored, and even better explainability of the CNN's feature extraction output. To achieve this, we propose Im-ROCKET as a proof-of-concept methodology that leverages the time-series ROCKET architecture for feature extraction and a linear classifier for usage in a imbalanced 150 class image-classification problem. The Python code, trained models, and results discussed below are available on the Im-ROCKET GitHub page here[1]. It should be noted that the GitHub repository does not include the pickle for the 3832 class model due to being too large to upload, but is available upon request.

[1] https://github.com/WhatsAnIStem/im-rocket

The novelty of the Im-ROCKET methodology comes from the usage self-intersecting space-filling curves to map 2D images onto the 1D time-series domain. Evaluation of Im-ROCKET is split between the evaluation of such a mapping, discussed in Section 3.1.2 and 4.3, as well as the performance of the entire classification pipeline in Section 4.3. Discussion of the results will focus on speculation about the high precision, low recall performance of the proposed Im-ROCKET model for the dataset chosen.

2 Related Work

2.1 Image Classification

There are countless sub-domains of image classification, including medical, computer-vision, and language processing. The sub-domain of Kanji identification was chosen due to the fact that many kanji classification datasets have high class counts and class complexities, as well as its personal interest to the author. In this field, there are countless methodologies and model frameworks being proposed and developed with the ultimate goal of correctly identifying Chinese based Kanji characters. Among the most advanced of these models is the Radical Analysis Network (RAN) proposed by Zhang, Du, and Dai(1) which is capable of identifying multi-label Kanji data from real-world images and is capable of zero-shot identification, trained on the ICPR MTWI-2018 dataset. Gyohten, Ohki, and Takami(2) developed a handwritten Kanji identification model that acts as a highly explainable Neural Network while performing at levels comparable to standard CNN models. Solis et. al.(3) designed a CNN ensemble which performs slightly better than its component parts in identifying handwritten Japanese Kana and Kanji characters. Despite the varying datasets, purpose, performance, and complexity of these models, they all leveraged standard CNN feature extraction techniques to transform their input data into a useful feature space. The ubiquitous nature of CNNs for feature extraction across Kanji Identification (and all image classification domains) makes it an interesting target for iteration and research. Im-ROCKET attempts to explore one way that the typical CNN feature extractor might be changed by drawing inspiration from a modified CNN architecture in the 1-dimensional time-series domain.

2.2 Time-Series Classification

In the 1-dimensional time-series classification domain, there has existed a framework for feature extraction based on a modified CNN kernel named ROCKET, originally proposed by Dempster, Petitjean, and Webb (4). Several derivatives of the ROCKET framework have been proposed by the same team of authors, most notably MINIROCKET(5) and MultiROCKET(6). Each of the mentioned ROCKET frameworks have their own advantages and drawbacks, but they all operate on the same principle of using randomized convolutional kernels to facilitate feature extraction. While MultiROCKET was chosen for use in the Im-ROCKET model due to the large amount of features per kernel, ROCKET and MINIROCKET were mentioned as they remain potential areas for future exploration.

2.3 Space-Filling Curves

To facilitate the mapping from 2 dimensions into 1 dimension, we make use of space-filling curves. Such space-filling curves were originally proposed by Giuseppe Peano in 1890 (7), and expanded upon by many other mathematicians at the time, namely David Hilbert with his proposal of the Hilbert curve (8) and Lebesgue, who is attributed with the creation of the Z-order curve, although primary sources corroborating this are difficult to find. Regardless, a formal description of the Z-order curve can be found in a 1966 report by G. M. Morton (9). Such space-filling curves provide one-to-one mappings of a 2-dimensional plane into a 1-dimensional line, and are expanded upon by Im-ROCKET to provide a better mapping for the purpose of using the MultiROCKET kernel.

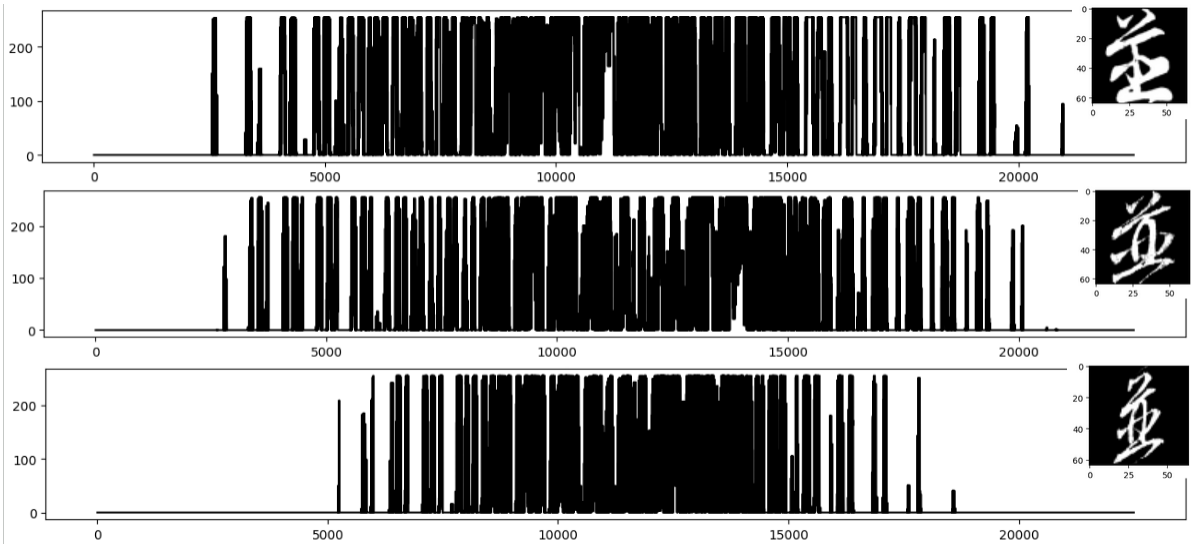

Figure 1: Mapping kernel data visualization

3 Methodology

3.1 2D Mapping Curve

As mentioned previously, space-filling curves have existed since the late 1800s. For Im-ROCKET, space-filling curves play an integral role in acting as the mapping function from the image domain to the time-series domain. However, there are many such algorithms that generate different space-filling curves (7)(8)(9), which then act as different mapping functions, considerably altering the resulting time-series and feature space output. Furthermore, the mapping from 2 dimensions to the 1 dimensional domain destroys a significant amount of the locality encoded by neighboring pixel values in an image. This destructive effect has a significant impact on the ability of ROCKET kernels to capture features that may exist in the original image. In an attempt to recover some of this destroyed locality, we choose to relax the one-to-one quality of the space-filling curves. Exactly how Im-ROCKET generates these curves are described in Section 3.1.1, with a novel curve evaluation function proposed in Section 3.1.2

3.1.1 Curve Generation

Algorithm 1 Mapping Function Generator

function GENERATE_MAPPING_FUNCTION(image_size, kernel_size, kernel_stride)
    mapping_function ← []
    $pos_x, pos_y \leftarrow 0$
    map_kernel ← $generate\_z\_curve$(kernel_size)
    while pos_y >= image_size.Y **do**
        for coordinate ∈ map_kernel **do**
            $walk_x$, $walk_y$ ← coordinate.$get\_coordinates$()
            mapping_function.$append$($(walk_x + pos_x, walk_y + pos_y)$)
        end forpos_x += kernel_stride
        if pos_x >= image_size.X **then**
            pos_x = 0
            pos_y += kernel_stride
        end if
    end while
    return mapping_function
end function

To prevent some amount of locality information encoded in the original image from being lost, we allow the mapping curves used by Im-ROCKET to be self-intersecting. Unfortunately, the practice of mapping 2-dimensions to 1-dimension using such self-intersecting curves is not well explored in academic literature, and the exploration of other potential methods to facilitate such a mapping is warranted for future research. As a starting point for generating these self-intersecting curves, we propose the simple algorithm in Algorithm 1, which is used to generate the curves for Im-ROCKET. To generate the Z-curves and Hilbert curves used by Im-ROCKET, pip packages zCurve by René Schubotz and numpy-hilbert-curve by Ryan Adams are used. A summary of the algorithm is as follows: First, determine a desired mapping $kernel\_size$ and mapping kernel stride. Second, generate a self-avoiding space-filling curve of $kernel\_size$ and overlap that kernel with a corner of the image that needs to be transformed.

There are a few qualities to the resulting time-series curve that appear when allowing the mapping curve to self-intersect. The most concerning quality is that allowing self-intersection allows the resultant time-series to drastically increase in length as we increase the size of the mapping kernel, assuming the stride stays the same size. Second, there is now a much larger variance in the kinds of paths that the mapping kernel can walk to generate the mapping function. While the Im-ROCKET methodology simply walks left-to-right and top-to-bottom, the exact path taken could be any desired path, even a new curve altogether. To explore the best possible mapping kernel size and mapping kernel stride, we propose a new adjacency preservation metric in Section 3.1.2 and conduct a grid search on a selection of possible parameters in Section 4.3.

3.1.2 Curve Evaluation

Algorithm 2 Adjacency Preservation Calculation

function GET_ADJACENCY_PRESERVED(mapping_function, len$_r$, len$_c$)
    $adjacency\_dict \leftarrow \{\}$
    for $point_c \in$ mapping_function **do**
        if NOT $point_c \in adjacency\_dict.keys()$ **then**
            $adjacency\_dict[point_c] = [\,]$
        end if
        point$_{xc}$, point$_{yc}$ $\leftarrow point_c.get\_image\_coordinates()$
        for $point_n \in$ {points within radius len$_r$/2 of $point_c$ in mapping_function} **do**
            point$_{xn}$, point$_{yn}$ $\leftarrow point_n.get\_image\_coordinates()$
            if point$_{xc}$, point$_{yc}$ within len$_c$/2 of point$_{xn}$, point$_{yn}$ **then**
                $adjacency\_dict[point_c].append(point_n)$
            end if
        end for
    end for
    $total \leftarrow 0$
    for $key \in adjacency\_dict.keys()$ **do**
        $total += len(adjacency\_dict[key])$
    end for
    $total /= len(adjacency\_dict.keys())$*len$_c$* len$_c$
    return total
end function

In an effort to better understand the potential shortcomings of the 2d-to-1d mapping function, as well as to compare different parameter configurations of the mapping function, we propose a new algorithm to calculate a lower bound on how similar a hypothetical ROCKET kernel of length len_r is to a 2-dimensional convolutional kernel of size $len_c * len_c$, in terms of which pixel values are operated on at the same time. Specifically, for some pixel value in an image's time-series curve $pixel_c$, we count how many other pixels within radius $len_r/2$ are within radius $len_c/2$ of $pixel_c$ in the original 2-dimensional image. A pseudo-code of our implementation of this function is shown in Algorithm 2. The result returned by this function is hereby referred to as the "Adjacency Preservation Metric".

3.2 Feature Extraction and Classifier

With a suitable mapping function generated, Im-ROCKET walks the path described by the function and creates a 1-dimensional time-series curve for use in the MultiROCKET kernel. Three such mappings from 2-dimensional images to 1-dimensional time-series curves are shown in Figure 1, along with their corresponding images. Im-ROCKET then uses the transformed time-series dataset with the sktime Python library to fit and apply 1,250 MultiROCKET kernels, resulting in a feature space of 9,408 features per instance. This feature space is then used to train and test on a Scikit-Learn stochastic gradient descent (SGD) classifier with the "perceptron" loss function. This specific loss function was used as an attempt to emulate the multi-layer perceptron (MLP) classifier that normally proceeds the convolutional layer in a CNN. It is worth mentioning that an attempt was made to train a Scikit-Learn MLP to classify the extracted features with several different activation functions and hidden layer sizes, but all failed. It is unknown if this failure was due to an issue with implementation of the MLP or that a MLP may be poorly suited to operate on these kinds of features. There may also be a focus on hyper-parameter tuning the MultiROCKET kernels and SGD classifier in future work, but was not done in this work.

4 Experiments

4.1 Data

For developing and testing Im-ROCKET, the KKanji — alternatively known as Kuzushiji-Kanji — dataset was identified. The dataset consists of 3832 classes over 140,426 instances. The dataset is also highly imbalanced, with the most common class containing 1766 instances, and the least common class containing only a single instance. Each instance in the dataset is a single 64x64 gray-scale image depicting a hand-drawn Japanese Kanji. An imbalanced dataset with more than two potential classes is desired to gauge the general performance of the initial framework for both low-shot and multiclass identification potential.

After identification of the dataset, (3) was identified for the purpose of comparing the performance of Im-ROCKET to several already existing CNNs. In an effort to mimic the data used by (3) as closely as possible, the 150 most occurring classes were separated from the 3832 total classes and used for training and evaluation. A second Im-ROCKET model was trained using the entire dataset, whose results are also included in Section 4.3. For future clarity, any mention of Im-ROCKET parameters that do not distinguish between the 150-class case and 3832-class case should be assumed as applied to both.

4.2 Experimental Setup

After being shuffled, the dataset was split into a training and testing set at ratios of 7:3 (following []) and 4:1 for the 150 class case and 3832 class case respectively. It should be clarified that there is no validation set because there was no parameter tuning past the mapping kernel grid search, which is independent of the actual content in the images. The mapping and MultiROCKET kernel was applied once for the entire 3832 class case, and the relevant class data was separated for use in the 150 class case. The mapping, MultiROCKET feature extraction, and classifier training was all done on a single Amazon EC2 R8G-2xLarge instance, which features 8 virutal CPU cores and 64 gigabytes of memory. The total feature extraction time of the MultiROCKET kernel was taken for the entire 140,426 image dataset, and from this the extraction time of the 150 class case was also estimated. The performance of the 150 and 3832 class cases were measured with macro and weighted precision, recall, and f-1 scores. The accuracy calculated from these three metrics were compared to the accuracy of the CNN ensemble from (3) to gauge relative performance of the Im-ROCKET model.

4.3 Experimental Results

This section shows the performance of the mapping kernel grid-search in Table 1 with the chosen parameters for the mapping kernel bold-faced and underlined. A majority of the written discussion will be focused on the results of the proposed Im-ROCKET model on both the 150 and 3832 class dataset, whose precision, recall, f-1, and accuracy is shown in Table 2.

Table 1: Grid-search of mapping kernel size and stride

Kernel Size	Stride	Adjacency Preserved	Resultant Time Series Length
2	1	0.102	15876
3	1	0.157	34596
3	2	0.137	8649
4	1	0.270	59536
4	2	0.173	15376
4	3	0.147	7056
5	1	0.401	90000
5	***2***	***0.261***	***22500***
5	3	0.192	10000
5	4	0.138	5625
6	1	0.423	125316
6	2	0.239	32400
6	3	0.212	14400
6	4	0.170	8100
7	1	0.441	164836
7	2	0.299	41209
7	3	0.233	19600
7	4	0.155	11025

Table 2: Im-ROCKET results on trimmed 150 class dataset and entire 3832 class dataset

	150 Class			3831 Class		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Macro Average	0.55	0.28	0.28	0.45	0.03	0.03
Weighted Average	0.66	0.37	0.38	0.75	0.22	0.23
Accuracy	0.37			0.22		
Training Time (hr:m)	0:08			5:43		

Aside from the metrics calculated from the grid-search in Table 1, the feature extraction time of the mapping kernel and MultiROCKET kernel was measured for the entire dataset, taking a total of 8 hours and 10 minutes over 140,426 samples. This averages to about 0.2 seconds per sample, or just under 5 samples per second. Adjusting this time for the 70,599 samples in the 150 class case comes to around 14,780 seconds, or 4 hours and 6 minutes. After running feature extraction on the entire dataset, the new feature space was saved to five separate .csv files, each taking up about 7 gigabytes of memory.

Aside from the metrics shown in Table 2, the breakdown of each metric for each class is also available in the mentioned GitHub[2] repository, named "150_class_stats_out.txt" and "single_class_stats_out.txt" for the 150 class case and 3832 class case respectively. Some of the trends visible for certain classes in the "150_class_stats_out.txt" is discussed further in Section 5. The most apparent difference between the 150 and 3832 class case is the drastic difference in the classifiers training time. Unfortunately, it is unknown whether the difference in training time between the two models was due to the large difference in classes between the two or the 150 class model converging before the 1000 epoch threshold, but it is likely a combination of the two. Futher exploration to determine this for certain is warranted.

5 Discussion

5.1 Performance

Focusing on the results of the 150 class case, it seems that the Im-ROCKET model is able to precisely classify certain classes at the cost of low recall. Investigating further, it seems that when the model was not able to effectively learn a certain class, it chooses one or two classes to fall back on in its prediction. The predictions of the model on the test set actually support this claim, as the two classes

[2]https://github.com/WhatsAnIStem/im-rocket

"煮" and "長" are predicted 4655 and 2908 respective times out of the 21180 samples in the testing set. These two classes take up a total of 35% of all predictions, despite the fact that the two classes appear only 74 and 72 times, respectively.

The observation above does not mean that the Im-ROCKET model was unable to learn any classes at all. In fact, the best performing class was the "一" class, with an f-score of 0.99 over 548 occurrences in the ground truth. There are a handful of other classes with f-scores above 0.80 of varying handwritten complexity. Further, it seems that there are quite a few classes of the 150 total with an f-1 score of 0.00, which the model was unable to correctly distinguish, but confident enough to incorrectly guess for some test instances. Of the 150 classes, there are also 41 classes that were not predicted by the model at all, and several of the remaining 109 classes which were predicted less than 10 times. These trends in the model's performance suggest that there is a large amount of room for improvement in future iterations of Im-ROCKET.

5.2 Future Work

Future work on the Im-ROCKET methodology should focus mainly on improvement and exploration of the 2d-to-1d mapping kernel. Specifically, modifying the path taken by the mapping kernel across the wider image and exploring the exact relationship between the adjacency preservation metric and the model's wider performance is a good place to start. Another potential avenue for iterating the Im-ROCKET architecture is to compare the performance of faster ROCKET kernels, namely the base ROCKET and MINIROCKET on the performance of the final classification model. The faster ROCKET/MINIROCKET kernels provide less features per randomized kernel, but their increase in speed can allow for more kernels to be used, potentially providing more or better distinguishing features for each data instance. Ideally, future iterations of Im-ROCKET would be able to leverage the extremely fast transformation times of ROCKET kernels while achieving comparable performance to a standard CNN.

5.3 Limitations

A fact that is insignificant to the overall performance of the model, but may be worth mentioning, is that the KKanji dataset was not the first identified dataset for use training the Im-ROCKET model. Initially, the ICPR MTWI-2018 challenge 2 dataset was identified and used for development of the Im-ROCKET methodology. However, efforts to use this dataset were abandoned after the realization was made that the multi-label class data of the dataset would not be well suited toward the SGD classifier used at the tail end of the Im-ROCKET model. The time cost of having to pivot datasets, recompute, and rework certain methods of the Im-ROCKET model took time away from efforts where further iteration of the mapping kernel could have focused.

6 Conclusion

When considering the relative low-complexity of the Im-ROCKET pipeline with the results of the proposed model, we believe that the current state of the Im-ROCKET methodology serves as a decent proof-of-concept methodology for feature extraction and classification. While a majority of time consumed by the model is due to the feature extraction of the MultiROCKET kernel, and shows that further time improvement could be made. Combining potential performance improvements with drastic increases in time promised by the ROCKET kernels, the eventual development of Im-ROCKET into a competitor against traditional CNN feature extraction may be possible.

All work on this project was done by the sole author, Titania Ervin. Several pip packages mentioned in previous sections were used to accelerate the project's development where pre-existing methods and algorithms were needed.

References

[1] Zhang, J., Du, J., & Dai, L. (2020). Radical analysis network for learning hierarchies of Chinese characters. *Pattern Recognition*, 103, 107305-. https://doi.org/10.1016/j.patcog.2020.107305

[2] Gyohten, K., Ohki, H., Takami, T., Lu, H., Cho, S.-B., Yagi, Y., Blumenstein, M., Kamiya, T., & Liu, C.-L., (2023). Character Structure Analysis by Adding and Pruning Neural Networks in Handwritten Kanji Recognition. In *Pattern Recognition* (Vol. 14407, pp. 129–142). Springer. https://doi.org/10.1007/978-3-031-47637-2_10

[3] Solis, A. I., Zarkovacki, J., Ly, J., & Atyabi, A. (2023). Recognition of Handwritten Japanese Characters Using Ensemble of Convolutional Neural Networks. *arXiv.Org*. https://arxiv.org/abs/2306.03954

[4] Dempster, A., Petitjean, F., & Webb, G. I. (2020). ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery, 34*(5), 1454–1495. https://doi.org/10.1007/s10618-020-00701-z

[5] Dempster, A., Schmidt, D. F., & Webb, G. I. (2021). MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 248–257. https://doi.org/10.1145/3447548.3467231

[6] Tan, C. W., Dempster, A., Bergmeir, C., & Webb, G. I. (2022). MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery, 36*(5), 1623–1646. https://doi.org/10.1007/s10618-022-00844-1

[7] Peano, G. (1890). Sur une courbe, qui remplit toute une aire plane. *Mathematische annalen, 36*(1), 157–160. https://doi.org/10.1007/BF01199438

[8] Hilbert, D. (1935). Über die stetige Abbildung einer Linie auf ein Flächenstück. In *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes* (pp. 1–2). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-38452-7_1

[9] Morton, G. M. (1966). A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. IBM Co. Ltd. https://dominoweb.draco.res.ibm.com/reports/Morton1966.pdf