Kannada-MNIST: A new handwritten digits dataset for the Kannada language

Vinay Uday Prabhu

August 3, 2019

In this paper, we disseminate a new handwritten digits-dataset, termed *Kannada-MNIST*, for the Kannada script, that can potentially serve as a direct drop-in replacement for the original MNIST dataset[1]. In addition to this dataset, we disseminate an additional real world handwritten dataset (with 10*k* images), which we term as the *Dig-MNIST*[2] dataset that can serve as an out-of-domain test dataset. We also duly open source all the code as well as the raw scanned images along with the scanner settings so that researchers who want to try out different signal processing pipelines can perform end-to-end comparisons. We provide high level morphological comparisons with the MNIST dataset and provide baselines accuracies for the dataset disseminated. The initial baselines obtained using an oft-used CNN architecture (96.8% for the *main test-set* and 76.1% for the *Dig-MNIST* test-set) indicate that these datasets do provide a sterner challenge with regards to generalizability than MNIST or the KMNIST datasets. We also hope this dissemination will spur the creation of similar datasets for all the languages that use different symbols for the numeral digits.

1 Introduction

Kannada is the official and administrative language of the state of Karnataka in India with nearly 60 million speakers worldwide [3]. Also, as per articles 344(1) and 351 of the Indian Constitution, Kannada holds the status of being one of the 22 scheduled languages of India [4]. The language is written using the official Kannada script, which is an *abugida* of the Brahmic family and traces its origins to the *Kadamba script* (325-550 AD).

Distinct glyphs are used to represent the numerals 0-9 in the language that appear distinct from the modern Hindu-Arabic numerals in vogue in much of the world today. Unlike some of the other archaic numeral-systems, these numerals are very much used in day-to-day affairs in Karnataka, as in evinced by the prevalence of these glyphs on license-plates of vehicles captured in fig 1.

Fig 2 captures the evolution of the numerals through the ages. Modern Kannada scholars [5] posit that the emergence of these numeral-glyphs can be traced to the *Gudnapur inscriptions* [6], dating back to the 6th century AD when the Kadamba rulers held sway over the region[7]. The Kannada digits for 0-9 are shown in Fig 1 (Unicode: 0CE6 through to 0CEF) [8] .

Fig 4 captures the MNIST-ized renderings of the variations of the

[1] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2:18, 2010

[2] The word *dig* is an ode to the diminutive used to denote Kannada speakers in the main author's alma mater
Evelyn Richter. Student slang at IIT Madras: a linguistic field study. Master's thesis, Technische Universitat Chemnitz, Str. der Nationen 62, 09111 Chemnitz, Germany, 2006

[3] Kannada. https://en.wikipedia.org/wiki/Kannada, 2019. [Online; accessed 16-Mar-2019]

[4] Eighth schedule to the constitution of india. https://en.wikipedia.org/wiki/Eighth_Schedule_to_the_Constitution_of_India, 2019. [Online; accessed 16-Mar-2019]

[5] Special correspondent;. Scholar throws light on the origin, evolution of kannada numerals. The Hindu, Nov 2017

[6] BR Gopal. Gudnapur inscription of kadamba ravivarma. Srikanthika: Dr S. Srikantha Sastri Felicitation Volume, pages 61–72, 1973

[7] G. S. Gai. In01046 no.22: Plate xxii gudnapur inscription of ravivarman, 1996

[8] Unicode charts. https://unicode.org/charts/PDF/U0C80.pdf, 2019. [Online; accessed 16-Mar-2019]

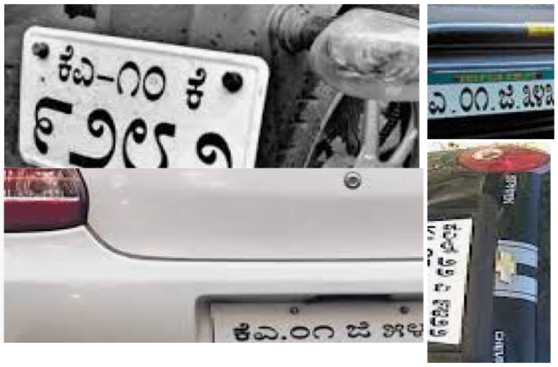


Figure 1: Usage of the Kannada numerals on vehicular license plates

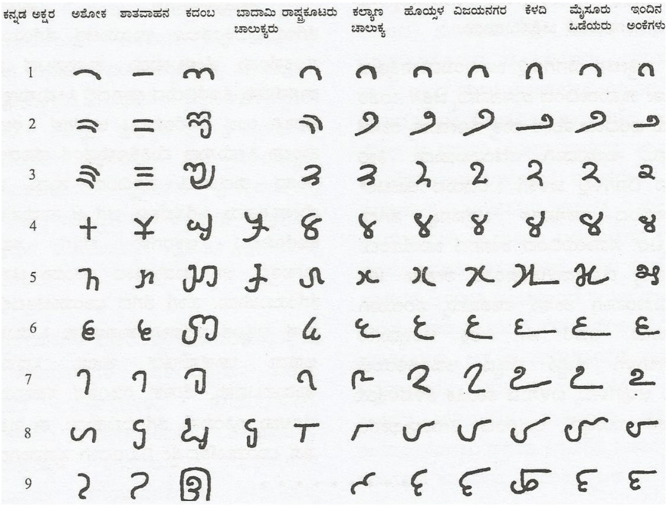
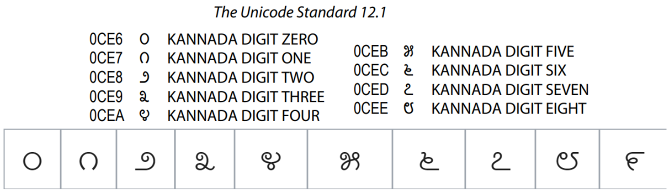

ಕನ್ನಡ ಅಕ್ಷರ
ಅಶೋಕ
ಶಾತವಾಹನ
ಕದಂಬ
ಬಾದಾಮಿ ಚಾಲುಕ್ಯರು
ರಾಷ್ಟ್ರಕೂಟರು
ಕಲ್ಯಾಣ ಚಾಲುಕ್ಯ
ಹೊಯ್ಸಳ
ವಿಜಯನಗರ
ಕೆಳದಿ
ಮೈಸೂರು ಒಡೆಯರು
ಇಂದಿನ ಅಂಕೆಗಳು
1
2
3
4
5
6
7
8
9

Figure 2: Evolution of the Kannada numerals through the ages ([a]M. G. Manjunath and G. K. Devarajaswamy. Kannada lipi vikasa. Technical report, Jagadhguru Sri Madhvacharya Trust, Sri Raghavendra Swami Matta, Mantralaya, 2004)

[a] [

glyphs across the following modern fonts: *Kedage, Malige-i, Malige-n, Malige-b, Kedage-n, Malige-t, Kedage-t, Kedage-i, Lohit-Kannada, Sampige and Hubballi-Regular*.



The Unicode Standard 12.1

0CE6	೦	KANNADA DIGIT ZERO
0CE7	೧	KANNADA DIGIT ONE
0CE8	೨	KANNADA DIGIT TWO
0CE9	೩	KANNADA DIGIT THREE
0CEA	೪	KANNADA DIGIT FOUR
0CEB	೫	KANNADA DIGIT FIVE
0CEC	೬	KANNADA DIGIT SIX
0CED	೭	KANNADA DIGIT SEVEN
0CEE	೮	KANNADA DIGIT EIGHT

೦	೧	೨	೩	೪	೫	೬	೭	೮	೯

Figure 3: The character code tables for Kannada-MNIST from the Unicode Standard, Version 12.1

1.1 The curious case of glyphs for 3,7 and 6

In this section, we focus on some idiosyncrasies with regards to the shapes of the glyphs used to represent numerals in Kannada. Three interesting observations emerge from Fig3 and Fig4.

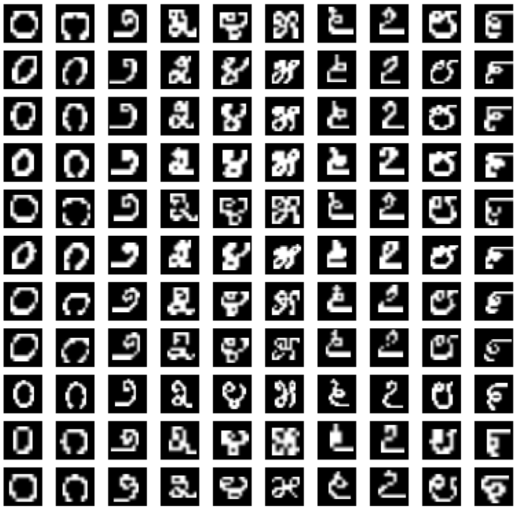


Figure 4: MNIST-ized renderings of the 0-9 Kannada numerals in 11 modern fonts

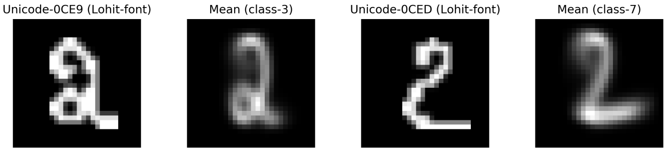


Figure 5: The similarity between the glyphs for 3 and 7 in Kannada

The first observation is that the glyph for 0 is the same as in the Hindu-Arabic system. Secondly, the shapes of the digits for 3 and 7 in Kannada look rather similar to the glyph for 2 in the modern Hindu-Arabic numeral system (Fig 5).

These will be leveraged during our dataset curation procedure as a sanity check for scanned and segmented digits using a pre-trained MNIST-digits classifier.

The third observation is related to the peculiar intra-class variation of the representation (refer to fig 4 and fig 2) for 6 across different fonts and different eras. The modern day deformations are represented in Fig 6, where we observe the deviation from the puritanical *textbook* representation of the symbol (as seen in the unicode-derived image on the extreme left) and the more colloquial usage which looks like a mirror image of 3 in the Hindu-Arabic system. As will be seen in the upcoming section, many of the volunteers who helped curate the dataset used one or both of these glyphs, resulting in high intra-class variation.

1.2 Related work

There have been some nascent attempts made towards Kannada handwritten digit classification, albeit at a smaller scale. In [9], the authors used the chain code histogram idea to achieve 98% accuracy on a dataset of 2300 digit-images. In [10], the authors used a nearest neighbor classifier to achieve 91% accuracy of 250 test numerals. Support Vector Machines (SVMs) were used to achieve 98% accuracy on a small dataset of 5000 40×40 numeral-images in [11]. The largest dataset currently used in academic literature that contains Kannada characters is the `Chars74k dataset` [12] that contains 657 characters of the Kannada script collected using a tablet PC, albeit with a mere 25 samples per-number. In [13], the authors harnessed standard augmentation techniques to create an augmented dataset of 18000 digit images harnessing the `Chars74k dataset` and trained Convolutional Neural Networks (CNNs) and Deep Belief Networks (DBNs) to obtain 98% test accuracy. Earlier this year, we proposed a Seed-Augment-Train/Transfer (SAT) framework that contains a synthetic seed image dataset generation procedure for languages with different numeral systems using freely available open font file datasets (Lohit to be more specific). This seed dataset of images was then augmented to create a purely synthetic training dataset, using which we trained a deep neural network and tested on held-out real world small-sized handwritten digits dataset spanning five Indic scripts, Kannada, Tamil, Gujarati, Malayalam, and Devanagari, containing 1280 digits each.

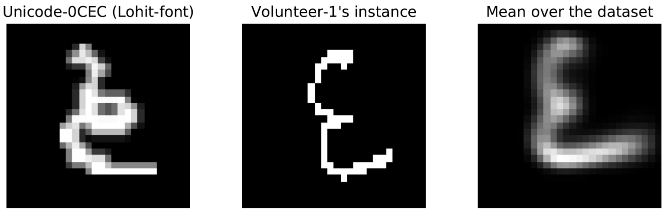


Figure 6: Deformations in the numeral glyphs for 6

Through this paper, we hope to address this paucity of an MNIST-sized dataset for the Kannada language.

1.3 Main contributions of the paper

The main contributions are:

1. Contributing a real world handwritten *Kannada-MNIST* dataset that was collected in Bangalore, India, that can potentially serve as a direct drop-in replacement for the original MNIST dataset[14].

[9] Nabin Sharma, U Pal, and Fumitaka Kimura. Recognition of handwritten kannada numerals. In 9th International Conference on Information Technology (ICIT'06), pages 133–136. IEEE, 2006

[10] GG Rajput and Mallikarjun Hangarge. Recognition of isolated handwritten kannada numerals based on image fusion method. In International Conference on Pattern Recognition and Machine Intelligence, pages 153–160. Springer, 2007

[11] GG Rajput, Rajeswari Horakeri, and Sidramappa Chandrakant. Printed and handwritten mixed kannada numerals recognition using svm. International Journal on Computer Science and Engineering, 2(05):1622–1626, 2010

[12] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal, February 2009

[13] Anirudh Ganesh, Ashwin R Jadhav, and KA Cibi Pragadeesh. Deep learning approach for recognition of handwritten kannada numerals. In International Conference on Soft Computing and Pattern Recognition, pages 294–303. Springer, 2016

[14] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2:18, 2010

2. Contributing an additional 10k real world handwritten *Dig-MNIST* dataset that was collected in Redwood City, CA, that can serve as an out-of-domain test dataset.

3. Open sourcing all the code required to generate such datasets for other languages.

4. Open-sourcing the raw scanned images along with the scanner settings so that researchers who want to try out different signal processing pipelines can perform end-to-end comparisons.

5. Performing high level morphological comparisons with the MNIST dataset and providing baselines accuracies for the dataset disseminated.

6. Open sourcing the code-templates for generating synthetic seed images for various modern Kannada fonts.

The rest of the paper is organized as follows: Section-2 covers the dataset preparation process, Section-3 details the comparisons vis-a-vis the standard MNIST dataset. In Section-4, we present the classification baseline results obtained using an off-the-shelf CNN, and Section-5 concludes the paper.

2 Dataset Creation

In order to avoid the kind of uncertainties, folklore and trivia surrounding MNIST (as evinced in [15]), we have decided to detail and open source all aspects of the data collection process. Further, we have also decided to open-source the raw scan images to facilitate end-to-end experimentation with disparate signal processing pipelines. In this section, we will cover the details of creating the following two datasets:

1. The *main* Kannada-MNIST dataset that consists of a training set of 60000 28 × 28 gray-scale sample images and a test set of 10000 sample images uniformly distributed across the 10 classes. This dataset is based off of the efforts of 65 volunteers from Bangalore, India, who are native speakers and users of the Kannada language and the script. This was curated to serve as a direct one-to-one drop-in replacement for the original MNIST dataset (akin to Fashion-MNIST [16] and K-MNIST [17] datasets).

2. The `Dig-MNIST` dataset that consists of 10240 28 × 28 gray-scale images that was curated with the purpose of providing a more challenging test dataset that was curated in Redwood City, CA, with the help of volunteers many of whom were encountering

[15] Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. Technical report, arxiv-1905.10498, may 2019

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017

[17] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018

the Kannada script for the first time and had fair difficulty in replicating the shape of the glyphs. This test dataset, we hope will facilitate domain adaptation experiments.

2.1 Main dataset

Fig 7 presents the work-flow followed to curate the *main* Kannada-MNIST dataset. The whole process was split into four phases: Data-gathering, pre-processing and slicing, Sanity-check, Train-test split. In the following subsections, we will cover each of these in detail. Data-gathering 65 volunteers were recruited in Bangalore, India, who were native speakers of the language as well as day-to-day users of the numeral script. Each volunteer filled out an A3 sheet containing a 32×40 grid. This yielded filled-out A3 sheets containing 128 instances of each number which we posit is large enough to capture most of the natural intra-volunteer variations of the glyph shapes. All of the sheets thus collected were scanned at 600 dots-per-inch resolution using the `Konica Accurio-Press-C6085` scanner that yielded 65 4963×3509 png images. Pre-processing and slicing In this sub-phase, each of the 4963×3509 sized scanned 32×40 grid png images were passed through two pre-processing stages[18] as detailed in [19]. This approach was originally used as an extraction framework to eke out the digits for a *Sudoku*-solver. The first pre-processing stage entails:

1. Applying a Gaussian-blur filter of kernel-size 9×9
2. Performing adaptive thresholding using 11 nearest neighbour pixels
3. Applying a `bitwise-NOT` operator to perform colour inversion to ensure that the target gridlines have non-zero pixel values

The second pre-processing phase entailed two operations. The first was to estimate the corners of the largest polygon that was then harnessed to crop and warp the 32×40 grid-image. The intermediate images after these phases are as shown in Fig 7.

The cropped-and-warped image thus obtained was then segmented into 1280 slices to yield the constituent individual digit images which were then *MNIST-ized*[20]. For this, we followed the procedure in [21] that entails pixel-thresholding, row-column padding and finally inflicting a `Best-shift` transformation to drag the current center-of-mass of the digit image to the center of the target *MNIST-ized* 28×28 image. At the end of this phase, we had a $128 \times 10 \times 28 \times 28$ image tensor per scanned image. The class-label associated with each image was obtained using the row index of the image in

[18] https://bit.ly/32WTxeT

[19] Sudoku solver 2. https://gist.github.com/mineshpatel1/209038c64c19d5e78e0a878320797631#file-sudoku_cv-py, 2017. [Online; accessed 16-July-2019]

[20] https://medium.com/@o.kroeger/tensorflow-mnist-and-your-own-handwritten-digits-4

[21] Tensorflow, mnist and your own handwritten digits. https://medium.com/@o.kroeger/tensorflow-mnist-and-your-own-handwritten-digits-4 2016. [Online; accessed 16-July-2019]

the 32×40 grid. Sanity-check One natural question that emerged during our new dataset curation was how to ensure the MNIST-compatibility of the same? In order to assuage these concerns, we decided to literally use a CNN pre-trained on the original MNIST digits and perform inference on the newly created digit images targeting classes in Kannada that looked *similar* to the MNIST digits.This formed an integral part of a series of *sanity checks* we performed that are as shown in Fig 7 and listed below:

1. Firstly, we perform class-wise checks by looking at the histogram of counts of the labels as well as eye-balling out the class-wise mean images of the 128×10 digits array and visually verifying that they indeed *look* like their archetypal glyphs.

2. Secondly, using the observations made in Section 1.1, we perform 3 sets of classifications on the *MNIST-ized* digit images. We use a high accuracy (99.4% test-set accuracy) CNN[22] pre-trained on MNIST digits to classify the images belonging to class `zero`(as the glyph for zero is the same), `three` and `seven` (as the glyphs look very similar to 2 in MNIST). This produces a triple of accuracies which are used to ascertain the *quality* and *MNIST-likeness* of the images produced by the parsing procedure we've deployed. With regard to Fig 7, 95% of the 128 zero-images were classified by the MNIST-CNN as zero, 96% of the 128 `three`-class images were classified by the MNIST-CNN as class-2 and 83% of the 128 `seven`-class images were classified by the MNIST-CNN as class-2.

We have duly shared the implementation of the step-wise procedure described above as a `colab` notebook accessed here: https://github.com/vinayprabhu/Kannada_MNIST/blob/master/colab_notebooks/0)Scan_parse_example_main.ipynb Train-test split: Worst cohort selection Using the curation procedure described above, we were able to collect $65 \times 1280 = 83200$ images. In order to ensure that our dataset would serve as drop-in replacement to the MNIST dataset, we had to select 60000 images for the final training set and 10000 images for the final test dataset. Upon random selection, we were able to hit similar levels of accuracy (>99%) that is achieved for the MNIST dataset. This could very well be attributed to the fact that random sampling based train-test data segmentation essentially allows the CNN to cover the span of the possible *glyph-modes* used by a user to represent a numeral whereas the *real* generalization challenge lies in being able to model the possible deviations that the glyphs shape might take across *unseen* users. Hence, we first sorted the users according to their difficulty scores. The difficulty score for a user was computed by taking the mean of the elements of the proxy-score-vector, which represents the probabilities that the user's 0, 3 and 7

[22] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

representations in Kannada would be classified as 0, 2 and 2 by the MNIST-CNN classifier as explained in Section 2.1. We then picked the top/worst 8 users into a test cohort and sampled 10000 digits to form the final test dataset. We then picked the next 47 users and sampled 60000 digits to form the final train dataset. This implies that any test-accuracy achieved by a machine learning classifier model will actually map to the ability of the classifier to predict the digit-classes from images emanating from users *hitherto unseen* during the training phase. The triple of 0-3-7 class accuracy-vectors of the train and test datasets thus formed were $[0.943, 0.962, 0.9575]$ and $[0.825, 0.87, 0.743]$ respectively.

We have shared the implementation of this procedure through a colab notebook shared here: https://github.com/vinayprabhu/Kannada_MNIST/blob/master/colab_notebooks/1b)_Main_dataset_tensor_generation_worst_cohort.ipynb

The class-wise mean images of the train set (top-row) , the test set (second-row) and the difference between the train and test classwise-means is shown in Fig 8

2.2 10k Dig-MNIST dataset

As stated above, we also disseminate an additional more challenging 10k Dig-MNIST dataset in this paper that was collected using volunteers in Redwood City, many of whom were, in fact, seeing the numeral glyphs for the first time and trying their best to reproduce the shapes. This sampling-bias, combined with the fact that we used a completely different writing sheet dimension and scanner settings, resulted in a dataset that would turn out to be far more challenging than the *easy* test dataset curated in the above sub-section. The rest of this sub-section details the specifics of the procedure and the companion colab notebook can be obtained at: https://github.com/vinayprabhu/Kannada_MNIST/blob/master/colab_notebooks/2)_Kannada_MNIST_10k_RWC.ipynb.

2.3 Dataset curation

8 volunteers aged 20 to 40 were recruited to generate a 32 × 40 grid of Kannada numerals (akin to 2.1), all written with a black ink Z-Grip Series | Zebra Pen on a commercial Mead Cambridge Quad Writing Pad, 8-1/2" x 11", Quad Ruled, White, 80 Sheets/Pad book. We then scan the sheet(s) using a Dell - S3845cdn scanner (See Fig 9)with the following settings:

- Output color: Grayscale
- Original type: Text

- Lighten/Darken: Darken+3
- Size: Auto-detect

The reduced size of the sheets used for writing the digits (US-letter vis-a-vis A3) resulted in smaller scan (.tif) images that were all approximately 1600 × 2000. The `Darken+3` scanner option resulted in the grid lines being visible enough that it allowed us to use the same signal processing pipeline detailed in Section 2.1 built on the sudoku-digit extraction idea. Fig 10 captures the user-wise class-wise mean-images for the dataset thus curated.

3 Comparisons with the MNIST dataset

In the section, we provide some qualitative and quantitative comparisons between the MNIST and the Kannada-MNIST datasets[23]

3.1 Morphological comparisons

In Fig 11, we provide a comparison of the mean pixel-wise intensities between the MNIST and the Kannada-MNIST datasets. As seen, the Kannada-MNIST dataset is much less *peaky* with a maximal mean pixel-intensity of ~ 0.3 as compared to the MNIST dataset, that has a few pixel indices with mean pixel-intensities of ~ 0.6. In Fig 12, we used the *Morpho-MNIST* framework [24] to generate the statistics of morphological traits such as length, thickness, slant, width and height of the handwritten digits for the two datasets. As seen, the bi-modality of length as well as width is less pronounced for the Kannada digits. The slant-to-width joint-scatter-plots were visibly different between the two datasets as well.

3.2 Dimensionality reduction comparisons

To begin with, we used the Uniform Manifold Approximation and Projection (UMAP) [25] technique to visualize a two-dimensional projection of the two datasets. As seen in Fig 14, the two sub-plots paint a very different picture of the lower dimensional representations for the two datasets. We also performed dimensionality reduction analysis using PCA to understand the variation of explained variance across the PCA components. As seen in Fig 13,the top-50 PCA components *explain* 83% of the total variance for the MNIST dataset and only 63% for Kannada-MNIST.

[23] In all the results shared in the section, we used the MNIST-10k-Test dataset and the Kannada-MNIST-Test dataset for the experiments.

[24] Daniel C. Castro, Jeremy Tan, Bernhard Kainz, Ender Konukoglu, and Ben Glocker. Morpho-MNIST: Quantitative assessment and diagnostics for representation learning. 2018

[25] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018

class	precision	recall	f1-score	support
0	0.9702	0.9130	0.9408	1000
1	0.9239	0.9830	0.9525	1000
2	0.9960	0.9970	0.9965	1000
3	0.9627	0.9800	0.9713	1000
4	0.9538	0.9920	0.9725	1000
5	0.9828	0.9700	0.9763	1000
6	0.9466	0.9740	0.9601	1000
7	0.9967	0.9010	0.9464	1000
8	0.9822	0.9930	0.9876	1000
9	0.9771	0.9820	0.9796	1000
accuracy			0.9685	10000
macro_avg	0.9692	0.9685	0.9684	10000
weighted_avg	0.9692	0.9685	0.9684	10000

Table 1: Classification report for the Kannada MNIST dataset

4 Classification results

In this section, we present the classification results obtained by training an off-the-shelf CNN[26] (See Fig 15) using Adadelta optimizer with `learning-rate=1.0` and$\rho = 0.95$. For the *main* dataset, with $60,000 - 10,000$ train-test split, we achieved 97.13% top-1 accuracy. The classification report is as shown in Table 1 and the epoch-wise accuracy and loss plots are as shown in Fig 17. In Fig 16, we have the confusion matrix. This pre-trained CNN achieved 76.2% top-1 accuracy on the `dig`-10k dataset. In terms of precision, class-2 (63.9%) and class-6 (55.1%) were the most challenging. In terms of recall, classes 0,3 and 7 were all at the sub-61% level. This showcases the fragile nature of the CNN's ability to truly generalize across author-cohorts and provides for an interesting challenge to the machine learning community at large. Fig 18 and Table 2 provide the confusion matrix and the per-class classifcation report for the `dig` dataset.

Lastly, for the single-author 1280 digits dataset used in [27], we achieved 83.67% top-1 accuracy. Again, as seen in Table 3 the CNN did struggle to achieve good precision class-6 (60%) and good recall for classes -0 and 7 ($60 - 63\%$). Figure 19 provides the confusion matrix for the same. Given the smaller size of the test dataset, we did dig in to visualize the images of the digits that the CNN classified for classes 0 (Fig 20), 7(Fig 21) and 8(Fig 22). The title of each of the plots represents the predicted class by the CNN. As seen, for a human eye, most of the images do look like the glyphs. But, upon closer inspection, we observe the presence of *rogue* non-glyph pixels (akin to naturally occurring adversarial perturbations) and discontinuities

[26] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

Class	precision	recall	f1-score	support
0	0.8360	0.6074	0.7036	1024
1	0.9013	0.7578	0.8233	1024
2	0.6385	0.9434	0.7615	1024
3	0.8787	0.6016	0.7142	1024
4	0.9191	0.7432	0.8218	1024
5	0.7441	0.9541	0.8361	1024
6	0.5511	0.7949	0.6509	1024
7	0.8918	0.5713	0.6964	1024
8	0.7732	0.7725	0.7728	1024
9	0.7936	0.8711	0.8305	1024
accuracy			0.7617	10240
macro_avg	0.7927	0.7617	0.7611	10240
weighted_avg	0.7927	0.7617	0.7611	10240

Table 2: Classification report for the `dig` dataset

Class	Precision	Recall	f1-score	Support
0	0.99	0.61	0.75	128
1	0.88	0.95	0.91	128
2	0.75	1.00	0.86	128
3	0.99	0.79	0.88	128
4	0.98	0.87	0.92	128
5	0.83	0.98	0.90	128
6	0.60	0.89	0.72	128
7	0.95	0.63	0.76	128
8	0.78	0.71	0.74	128
9	0.88	0.93	0.90	128
Accuracy			0.84	1280
macro_avg	0.86	0.84	0.84	1280
weighted_avg	0.86	0.84	0.84	1280

Table 3: Classification report for the 1280 digits dataset used in [a]]Vinay Uday Prabhu, Sanghyun Han, Dian Ang Yap, Mihail Douhaniaris, Preethi Seshadri, and John Whaley. Fonts-2-handwriting: A seed-augment-train framework for universal digit classification. arXiv preprint arXiv:1905.08633, 2019

[a] [

in the strokes in many of the erroneously classified images, which we posit might well explain the misclassifications.

5 Conclusion and Future work

In this paper, we described in detail the creation of a new handwritten digits dataset for the Kannada language, which we term as `Kannada-MNIST` dataset. We have duly open sourced all aspects of the dataset creation including the raw scan images, the specific brand of paper used[28], the exact scanner model used, the signal processing script used to slice and extract the individual digits and the CNN models used to obtain the baseline accuracies. We were able to attain

[27] Vinay Uday Prabhu, Sanghyun Han, Dian Ang Yap, Mihail Douhaniaris, Preethi Seshadri, and John Whaley. Fonts-2-handwriting: A seed-augment-train framework for universal digit classification. arXiv preprint arXiv:1905.08633, 2019

$\sim$ 97% top-1 accuracy when we trained and tested on what we term as the *main* dataset with 60000 28 × 28 gray-scale training images and 10000 test images. This is meant to be in a drop-in replacement for the standard MNIST dataset. We also achieved a top-1 accuracy of $\sim$ 77% when we trained on the 60000 *main* dataset and tested on 10240 28 × 28 gray-scale test images from what we term as the `Dig-MNIST` dataset. The images in the *Dig*-MNSIT dataset are noisier with smudges and grid borders sneaking in during the grid-image segmentation phase.

We propose the following open challenges to the machine learning community at large.

1. Achieve MNIST-level accuracy by training on the Kannada-MNIST dataset and testing on the Dig-MNIST dataset without resorting to image pre-processing.

2. To characterize the nature of catastrophic forgetting when a CNN pre-trained on MNIST is retrained with Kannada-MNIST. This is particularly interesting given the observation that the typographical glyphs for 3 and 7 in Kannada-MNIST hold uncanny resemblance with the glyph for 2 in MNIST.

3. Get a model trained on purely synthetic data generated[29] using the fonts (as in [30]) and augmenting using frameworks such as [31] and [32] to achieve high accuracy of the Kannada-MNIST and Dig-MNIST datasets.

4. Replicate the procedure described in the paper across different languages/scripts, especially the Indic scripts.

5. With regards to the *dig*-MNIST dataset, we saw that some of the volunteers had transgressed the borders of the grid and hence some of the images either have only a partial slice of the glyph/stroke or have an appearance where it can be argued that they could potentially belong to either of two different classes. With regards to these images, it would be worthwhile to see if we can design a classifier that will allocate proportionate softmax masses to the *candidate* classes.

6. The main reason behind us sharing the raw scan images was to foster research into auto-segmentation algorithms that will parse the individual digit images from the grid, which might in turn lead to higher quality of images in the upgraded versions of the dataset.

[28] We have decided to also donate the hard copies of the handwritten digit-grids (See Fig 23) to an academic research lab. If interested, the researcher(s) are requested to contact the main author with a proposal

[29] https://github.com/vinayprabhu/Kannada_MNIST/blob/master/colab_notebooks/5)_Synthetic_seed_image_generation.ipynb

[30] Vinay Uday Prabhu, Sanghyun Han, Dian Ang Yap, Mihail Douhaniaris, Preethi Seshadri, and John Whaley. Fonts-2-handwriting: A seed-augment-train framework for universal digit classification. arXiv preprint arXiv:1905.08633, 2019

[31] Daniel C. Castro, Jeremy Tan, Bernhard Kainz, Ender Konukoglu, and Ben Glocker. Morpho-MNIST: Quantitative assessment and diagnostics for representation learning. 2018

Acknowledgement

To begin with, we'd like to acknowledge the contribution of all the volunteers who contributed towards this dataset. Specifically, we'd like to thank Kaushik BK, who was instrumental in helping manage the cohort of 65 volunteers who contributed to the main dataset in Bangalore, India. We'd also like to acknowledge the contributors of the Dig-MNIST dataset in Redwood City, including John Whaley, Nick Richardson, Joseph Gardi and Preethi Sheshadri. Last but not the least, we'd like to acknowledge the helpful advice shared by the authors of the K-MNIST paper, Tarin Clanuwat, Alex Lamb(Mila) and David Ha (Google Brain).

References

[1] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2:18, 2010.

[2] Evelyn Richter. Student slang at IIT Madras: a linguistic field study. Master's thesis, Technische Universitat Chemnitz, Str. der Nationen 62, 09111 Chemnitz, Germany, 2006.

[3] Kannada. https://en.wikipedia.org/wiki/Kannada, 2019. [Online; accessed 16-Mar-2019].

[4] Eighth schedule to the constitution of india. https://en.wikipedia.org/wiki/Eighth_Schedule_to_the_Constitution_of_India, 2019. [Online; accessed 16-Mar-2019].

[5] Special correspondent;. Scholar throws light on the origin, evolution of kannada numerals. The Hindu, Nov 2017.

[6] BR Gopal. Gudnapur inscription of kadamba ravivarma. Srikanthika: Dr S. Srikantha Sastri Felicitation Volume, pages 61–72, 1973.

[7] G. S. Gai. In01046 no.22: Plate xxii gudnapur inscription of ravivarman, 1996.

[8] M. G. Manjunath and G. K. Devarajaswamy. Kannada lipi vikasa. Technical report, Jagadhguru Sri Madhvacharya Trust, Sri Raghavendra Swami Matta, Mantralaya, 2004.

[9] Unicode charts. https://unicode.org/charts/PDF/U0C80.pdf, 2019. [Online; accessed 16-Mar-2019].

[10] Nabin Sharma, U Pal, and Fumitaka Kimura. Recognition of handwritten kannada numerals. In 9th International Conference on Information Technology (ICIT'06), pages 133–136. IEEE, 2006.

[11] GG Rajput and Mallikarjun Hangarge. Recognition of isolated handwritten kannada numerals based on image fusion method. In International Conference on Pattern Recognition and Machine Intelligence, pages 153–160. Springer, 2007.

[12] GG Rajput, Rajeswari Horakeri, and Sidramappa Chandrakant. Printed and handwritten mixed kannada numerals recognition using svm. International Journal on Computer Science and Engineering, 2(05):1622–1626, 2010.

[13] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal, February 2009.

[14] Anirudh Ganesh, Ashwin R Jadhav, and KA Cibi Pragadeesh. Deep learning approach for recognition of handwritten kannada numerals. In International Conference on Soft Computing and Pattern Recognition, pages 294–303. Springer, 2016.

[15] Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. Technical report, arxiv-1905.10498, may 2019.

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

[17] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018.

[18] Sudoku solver 2. https://gist.github.com/mineshpatel1/209038c64c19d5e78e0a878320797631#file-sudoku_cv-py, 2017. [Online; accessed 16-July-2019].

[19] Tensorflow, mnist and your own handwritten digits. https://medium.com/@o.kroeger/tensorflow-mnist-and-your-own-handwritten-digits-4d1cd32bbab4, 2016. [Online; accessed 16-July-2019].

[20] Daniel C. Castro, Jeremy Tan, Bernhard Kainz, Ender Konukoglu, and Ben Glocker. Morpho-MNIST: Quantitative assessment and diagnostics for representation learning. 2018.

[21] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.

[22] Vinay Uday Prabhu, Sanghyun Han, Dian Ang Yap, Mihail Douhaniaris, Preethi Seshadri, and John Whaley. Fonts-2-handwriting: A seed-augment-train framework for universal digit classification. arXiv preprint arXiv:1905.08633, 2019.

[23] Marcus D Bloice, Peter M Roth, and Andreas Holzinger. Biomedical image augmentation using augmentor. Bioinformatics, 2019.

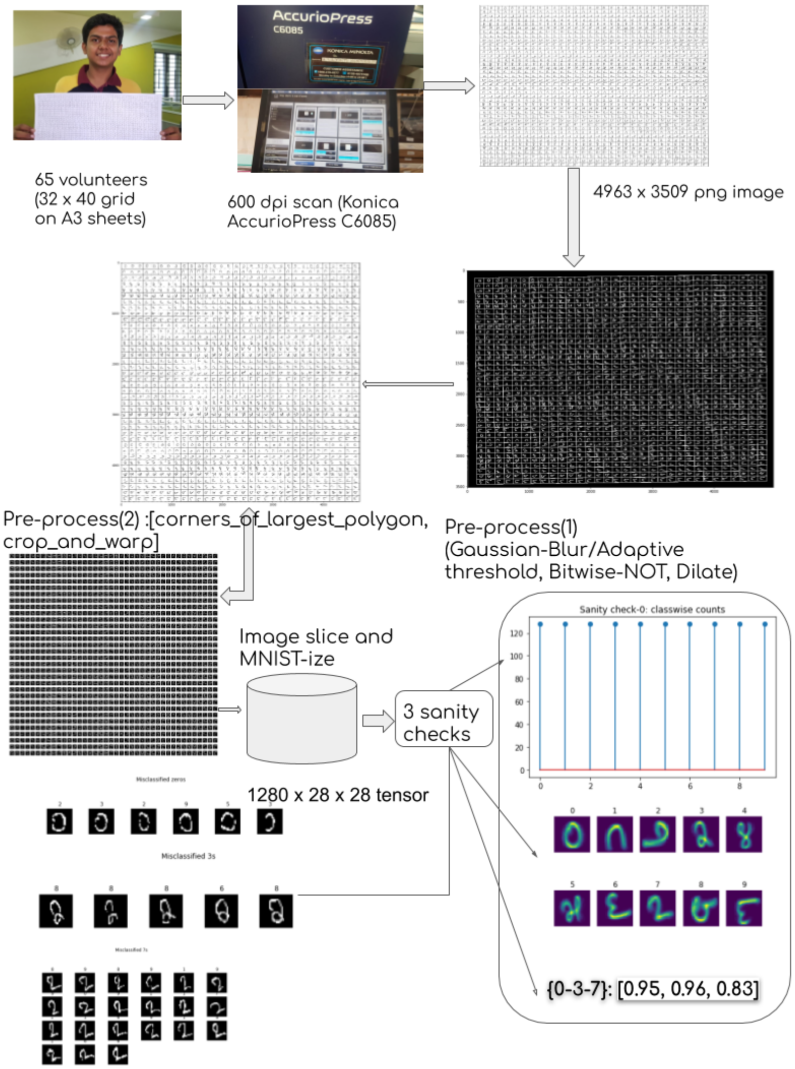


Figure 7: The main dataset creation workflow

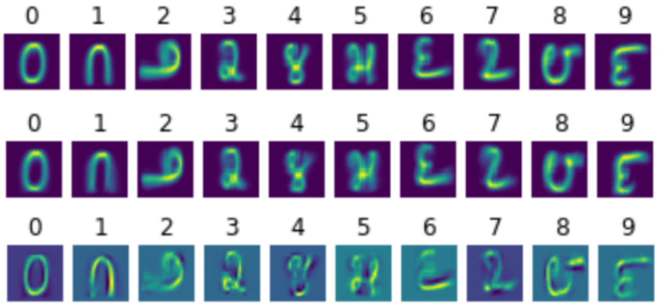


Figure 8: Class-wise mean images of the train set, the test set and the difference between the means of the train and test sets

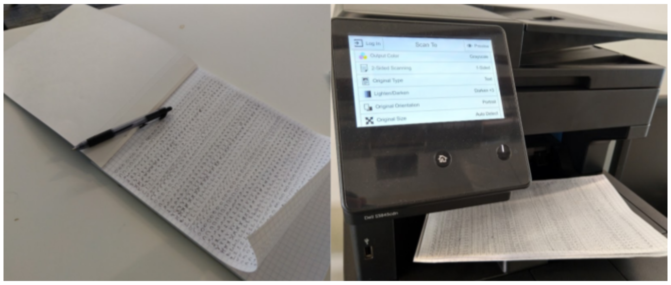


Figure 9: Preparing the *dig*-dataset in Redwood City

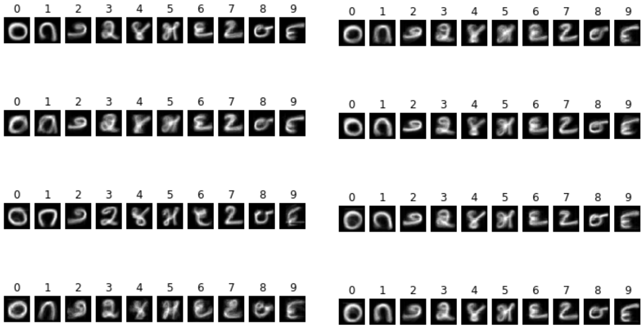


Figure 10: User-wise class-wise mean images for the Dig-10k dataset

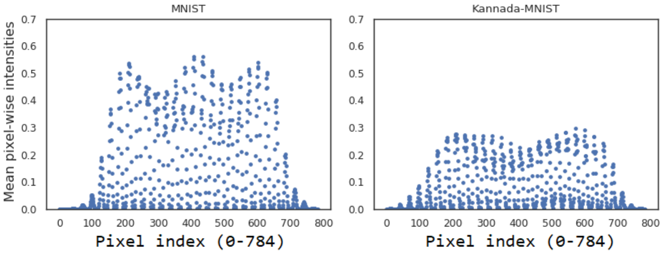
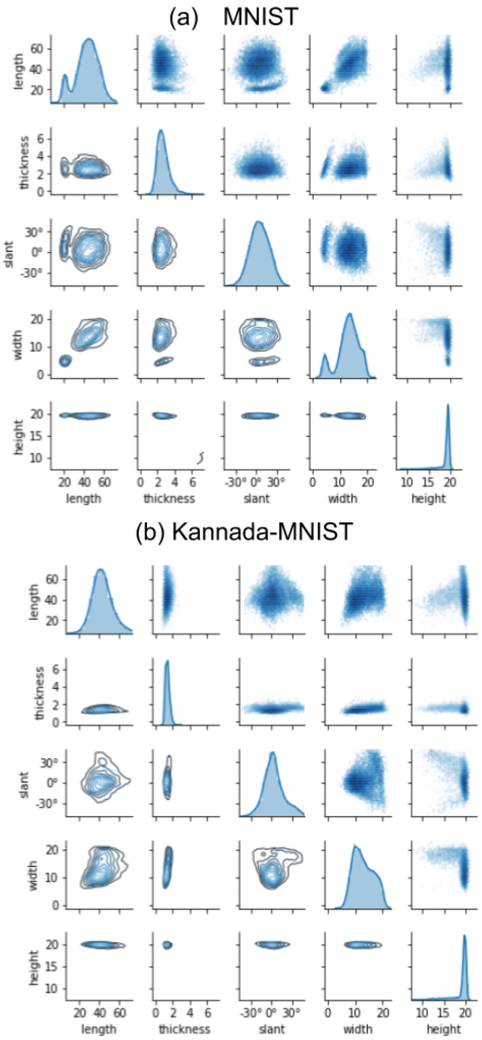


Figure 11: Mean pixel-wise intensities comparisons between MNIST and the Kannada-MNIST datasets



(a) MNIST

(b) Kannada-MNIST

Figure 12: Morphological comparisons between MNIST and the Fashion-MNIST

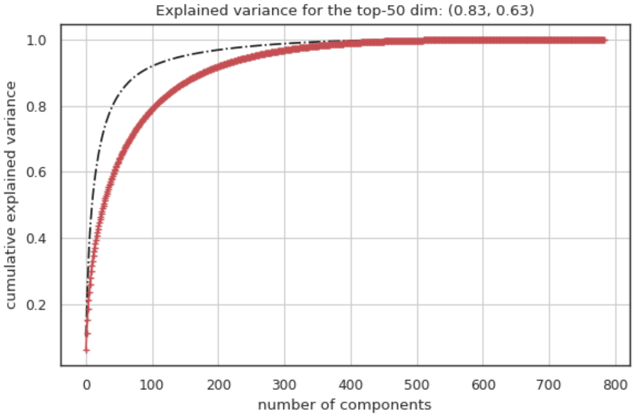


Figure 13: PCA analysis for the two datasets

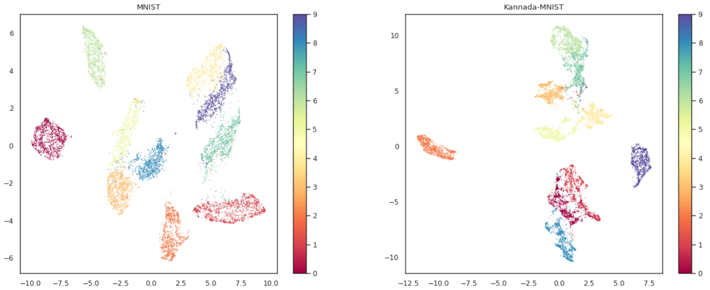
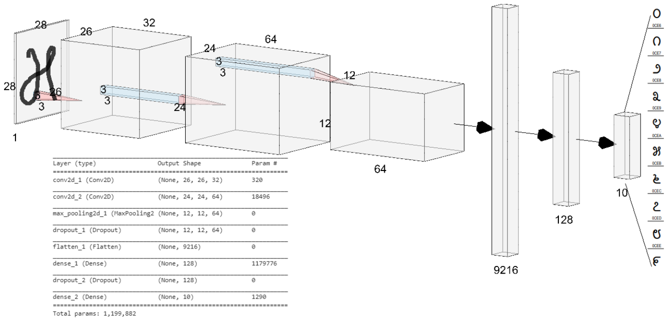


Figure 14: Two-dimensional Uniform Manifold Approximation and Projection (UMAP) plots for the two datasets



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
```

Figure 15: The CNN architecture used in the paper

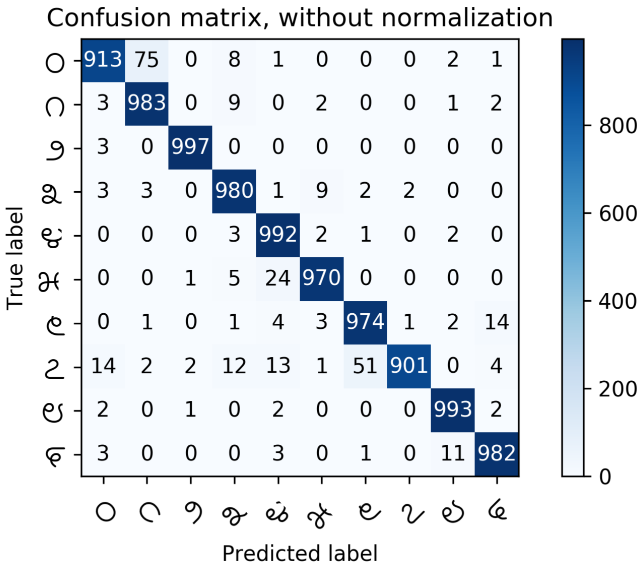


Figure 16: Confusion matrix with regards to the Kannada-MNIST datasets

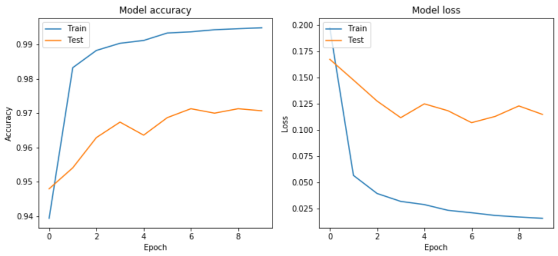


Figure 17: Train and test accuracies for the CNN trained and tested on the *main* dataset

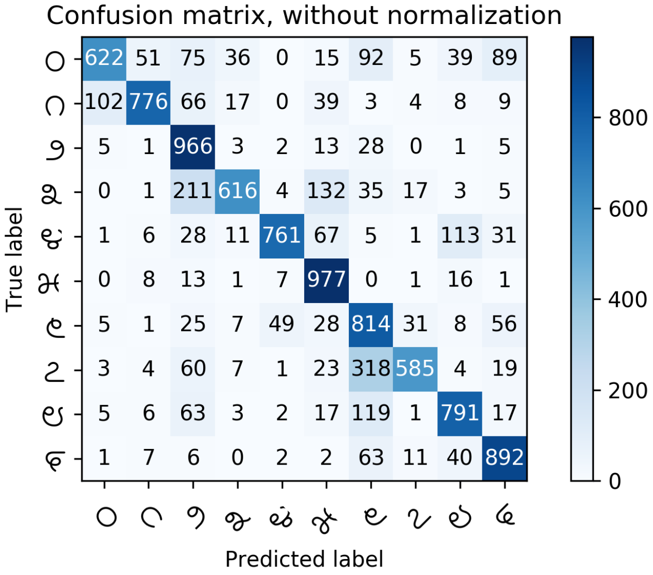


Figure 18: Confusion matrix for the `dig-10k` dataset

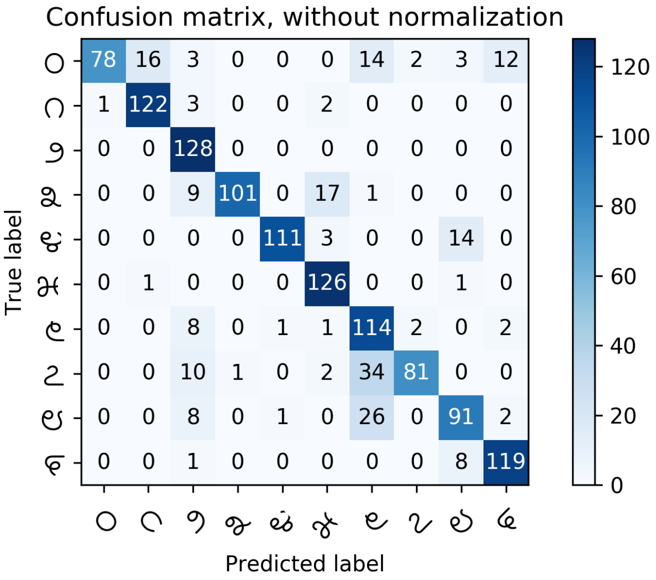


Figure 19: Un-normalized confusion matrix for the 1280 digits dataset using in [a]]Vinay Uday Prabhu, Sanghyun Han, Dian Ang Yap, Mihail Douhaniaris, Preethi Seshadri, and John Whaley. Fonts-2-handwriting: A seed-augment-train framework for universal digit classification. arXiv preprint arXiv:1905.08633, 2019

[a] [

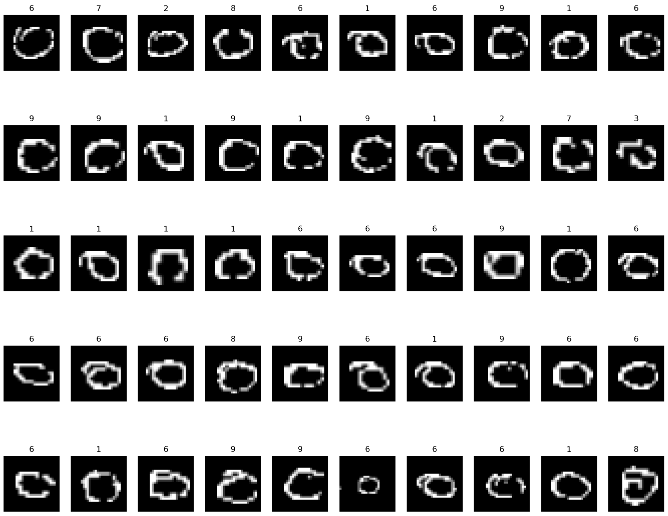


Figure 20: Images belonging to class-0 in the 1280-digits dataset that were misclassified by the CNN trained on the main dataset

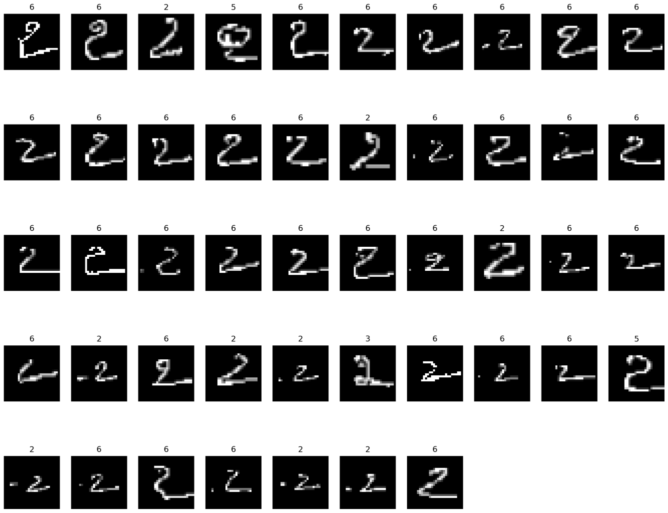


Figure 21: Images belonging to class-7 in the 1280-digits dataset that were misclassified by the CNN trained on the main dataset

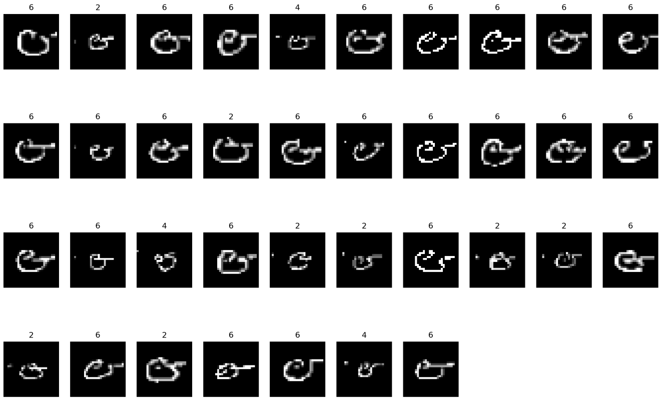


Figure 22: Images belonging to class-8 in the 1280-digits dataset that were misclassified by the CNN trained on the main dataset

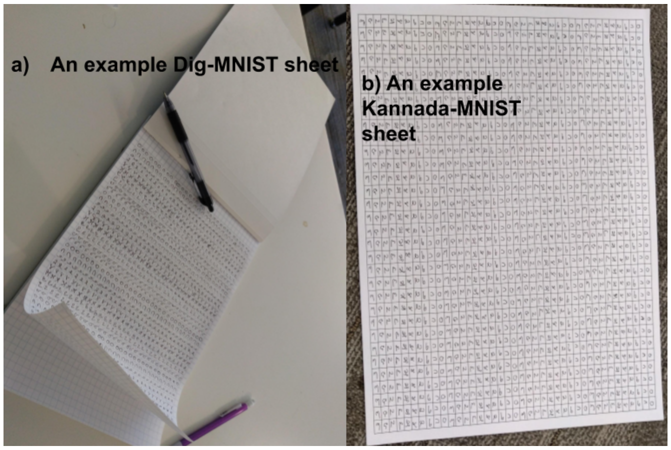

a) An example Dig-MNIST sheet
b) An example Kannada-MNIST sheet

Figure 23: Photos of hard copies of the handwritten sheets for the two datasets