Advancing Model Robustness: An Ensemble Approach with Extensive EDA for the K-MNIST & dig-MNIST Datasets

Shrihari Hampiholi*
School of Computer Science
KLE Technological University
Hubli, Karnataka
shrihari@hampiholi.com

Anupama Bidargaddi†
School of Computer Science
KLE Technological University
Hubli, Karnataka
anupamapb@gmail.com

Maanasi Shastri‡
School of Computer Science
KLE Technological University
Hubli, Karnataka
shastrimaanasi8@gmail.com

Amrutha Beedikar§
School of Computer Science
KLE Technological University
Hubli, Karnataka
amruthabeedikar@gmail.com

Manav Hegde¶
School of Computer Science
KLE Technological University
Hubli, Karnataka
manavhegde2002@gmail.com

***Abstract*—Handwritten digit recognition in Kannada has particular difficulty due to its unusual script. In this work, we explore the intricacies of the traditional K-MNIST dataset and the dig-MNIST dataset, which is its out-of-domain counterpart. Our main goal is to solve the complex issues preventing advanced, state-of-the-art models from recognising Kannada digits in the dig-MNIST dataset with high accuracy. We do an extensive Exploratory Data Analysis (EDA) on both datasets, examining the morphological characteristics and finer points of differentiation between the numerals. Concentrating on the difficult-to-understand small distinctions between pairs of numerals in the Kannada language is our primary goal. As an out-of-domain (OOD) dataset, the dig-MNIST dataset reveals subtle, unique characteristics and challenges. Our ensemble model achieves an astonishing 91.4% accuracy on the dig-MNIST, which is developed meticulously from two CNN architectures. Interestingly, we deviate from pre-trained models and build these architectures from the ground up. The ensemble benefits from a blend of a traditional CNN model and one inspired by ResNet [1], leveraging diverse learning strategies.**

***Index Terms*—K-MNIST handwritten digit recognition, dig-MNIST handwritten digit recognition, Exploratory Data Analysis, Deep Learning**

I. Introduction

Handwritten digit recognition (HWR) is an essential technology application that is used in many different industries for automation, document digitization, optical character recognition, and accessibility improvement. Although the 1989 introduction of the standard MNIST [2] dataset provided the groundwork, issues with its accuracy still exist, particularly in situations with varying language settings. Kannada-style customised datasets are now essential for improving recognition algorithms.

This paper explores the complexities of handwritten digit recognition, highlighting the Kannada dataset, an essential tool for machine learning studies. Due to minor differences in Kannada [3] numerals, as those between 3 and 7 or 6 and 9, the dataset poses a special difficulty that calls for careful evaluation of these nuances, especially in edge circumstances. Our primary goal is to carefully examine and uncover the mysteries of efficient HWR for the Kannada script by carefully examining these subtleties. This thorough study intends to open the door for reliable and accurate recognition models customised to this particular linguistic environment, going beyond simple numerical correlations.

In addition to the primary goal, this study also involves creating and assessing an ensemble model [4] made up of two different CNN architectural models, and using an average method to make predictions. When tested on the dig-MNIST dataset, this ensemble model, which was trained only on the conventional K-MNIST dataset and made use of sophisticated data augmentation techniques, demonstrated a noteworthy state-of-the-art accuracy of roughly 91.4%. The model's architecture, training procedures, and the results of our experimental investigations will all be covered in later sections.

This work is organized in the following manner, starting with Section II which presents a review of pertinent research in the area. We then transition to a thorough explanation of the datasets that are essential to the research and clarifies their contextual significance in Section III. The next Section, Analysis IV describes in detail the EDA process that was carried out, alongside the building of various architectures for our research. Following which, we present the results of various architectures that were constructed, along with the SOTA ensemble model that was built in Section V. Finally, we conclude by listing the future scope that can be carried off of this research in Section VI.

II. Related Work

This section focuses on handwritten digit recognition research, with a particular focus on Kannada numerals. It critically reviews and discusses previous research in this field. In order to offer insights on various approaches, methodologies, and discoveries pertinent to the subject matter, this section will examine previous work in the topic. By illuminating the shortcomings, opportunities, and weaknesses noted in earlier research, it provides a contextual foundation for the current effort.

A. *Deep Learning*

1) Off-the-shelf CNN architecture: Vinay Uday's research [5] showcases a significant application of deep learning models, particularly employing an off-the-shelf CNN architecture featuring 2 Convolutional layers and 2 Dropout layers for classification tasks the datasets.

The results obtained from training the CNN on the primary K-MNIST dataset of 60,000 training samples demonstrate remarkable performance, achieving a top-1 accuracy of 97.13% on the 10,000 K-MNIST test images. However, the transferability of this performance to the dig-MNIST dataset exhibits a reduced accuracy of 76.2%, indicating certain challenges in model generalization. Notably, specific classes, such as class-2 and class-6, posed considerable difficulties in precision, while classes 0, 3, and 7 suffered from relatively lower recall rates.

2) EffKannadaRes-NeXt: Saini et al. [6] introduce the EffKannadaRes-NeXt model, a novel framework designed for the correct and efficient recognition of Kannada numerals. This model builds upon the foundations of ResNeXt [7], a deep residual network architecture. The unique contribution of EffKannadaRes-NeXt lies in its ability to handle both binary and gray-scale representations of numeral images. The model trained on the K-MNIST training dataset consisting of 60,000 images. They were able to achieve an accuracy of 97.36% on the K-MNIST test dataset of 10,000 images, and an accuracy of 77.31% on the dig-MNIST dataset.

3) CNNPSO: In 2021, Gopal D. Upadhye et al. [8] trained an optimal CNN architecture using Partical Swarm Optimization (PSO) [9] directly on a part of the dig-MNIST dataset, consisting of 8000 images and were able to achieve an accuracy of 91.75% on the 2240 test images of the split dig-MNIST dataset.

4) CRNN: Akshitha [10] presented a CRNN model in this study, which combines recurrent neural network (RNN) [11] and convolutional neural network (CNN) [2]. To obtain high recognition accuracy for the test K-MNIST dataset, the fully connected CNN layer component is replaced by a GRU [12]. Primarily, the CNN makes sure that numerous convolutional and pooling layers can be used to extract the best characteristics from the original image. To add to this, a quick recognition technique called GRU [12] created a sequential link between features in the buried layer. Using the Kannada MNIST handwritten digit dataset, the experiment yielded greater precision, recall, and F1 score ratio in addition to 98.12% training accuracy and 98.95% validation accuracy on the K-MNIST dataset.

5) Chars74k: Using the Chars74k dataset, the researchers in [13] were able to generate a large set of 18,000 digit images by utilising standard augmentation techniques. It is important to note that the Chars74k dataset is not unique to the Kannada script, even if it is useful for character recognition applications. Convolutional Neural Networks (CNNs) and Deep Belief Networks (DBNs) [14] achieved an amazing 98% accuracy on the test set by using the augmented dataset as their training set. This work highlights the flexibility of using external datasets to improve deep learning models' performance in digit recognition tasks, even if they are not naturally Kannada-focused.

B. *Transfer Learning*

The focus of Qisheng Hu's [15] work is on using ten Convolutional Neural Network (CNN) models that have already been trained using the large ImageNet dataset. In order to handle the complex problems of handwritten digit recognition, these models were developed and refined by incorporating advanced data augmentation techniques. Among these ten models, the pre-trained InceptionV3 model proved to be the most effective in this study. When evaluated using the K-MNIST dataset, its versatility and sturdy architecture allowed it to attain an exceptional accuracy of 98.44%. Additionally, the InceptionV3 model demonstrated an impressive 90.27% accuracy when tested on the dig-MNIST dataset, proving to be the best model until now.

C. *Machine Learning*

1) Nearest Neighbour Classifier: Using a closest neighbour classifier, Rajput and Hangarge [16] used a novel picture fusion technique in their early investigation of Kannada numeral identification. Their experiment, which was published in 2007, recorded an impressive 91% accuracy on 250 separate handwritten Kannada numbers. Notably, this study was conducted before more current developments in the availability of Kannada datasets, highlighting the method's historical significance. Significantly, their emphasis on an alternative dataset emphasises how crucial it is to take into account a variety of training data outside of the conventional K-MNIST, providing insights into early approaches for Kannada numeral identification.

2) SVM: Support vector machines (SVMs) were used by Rajput, Horakeri, and Chandrakant [17] to obtain an astounding 98% accuracy on a special dataset of 5,000 (40×40) Kannada numeral pictures. Their study, which was published

in 2010, shows how versatile SVMs are in Kannada number recognition beyond the established benchmarks, deviating from the traditional K-MNIST standard dataset.

III. Methodology

This section first introduces both the datasets; K-MNIST & dig-MNIST, after which it offers a perceptive investigation of the development of various CNN architectures. A brief explanation of every architecture is provided. The performance & analysis of the networks are examined in detail in the following part.

A. Datasets' description

The Kannada MNIST (K-MNIST) dataset analyzed in this study uses 70,000 Kannada numeral images in total—60,000 for training and 10,000 for testing. With the help of 65 native Kannada speakers in Bangalore, India, this dataset—which initially had 83,200 photos—was meticulously selected.

Functioning as a drop-in substitute for the conventional MNIST, the 83,200 images were reduced in size to accommodate the total of 70,000 images to comply with the traditional MNIST. Of them, 60,000 images were allocated for model training.

In an attempt to emulate the MNIST [2] dataset's performance, the selection criterion was to maintain over 99% accuracy. Deliberate random selection was used to choose the training and test datasets as a step in the curation process. To further enhance the representational variety of the dataset, users were grouped based on their difficulty scores, which has been explained in detail by Vinay Uday Prabhu [5] in the dataset curation process.

The dig-MNIST dataset was built as a 'out-of-domain' dataset consisting of 10,000 photographs that were painstakingly collected and preprocessed by non-native Kannada speakers who live in Redwood City, California [5]. Curating this dataset was primarily done to support the creation of a model that demonstrates strong generalization across the K-MNIST and dig-MNIST datasets. In particular, the dig-MNIST dataset presents a unique issue because it contains a significant amount of new, unexpected, and perhaps some irrelevant information that has never been seen before, or more aptly put, the model has *never* trained on before.
The samples in Fig. 2 strongly demonstrate the above put proposition. Additionally, Fig. 1 show-cases the Kannada numerals as per the Noto Sans format.

The samples in Fig. 1 showcase the Kannada numerals as per the Noto Sans font, emphasizing its distinct style and representation. In contrast, Fig.2 vividly illustrates the proposition of unexpected, unique, and subtle differences between the two datasets, K-MNIST & dig-MNIST respectively.

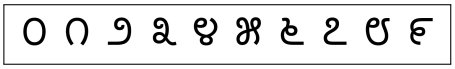
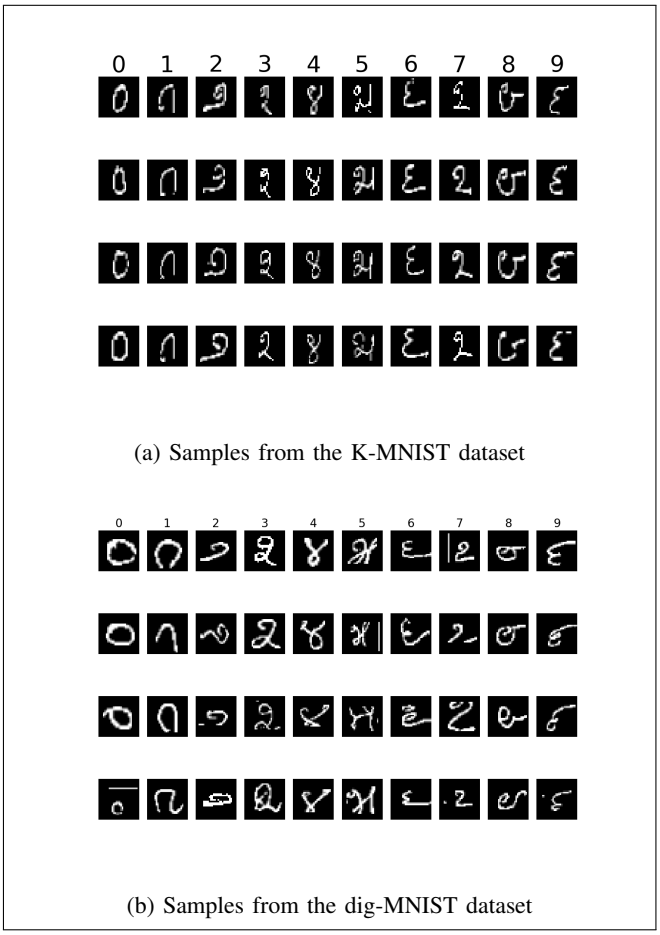


Fig. 1: Kannada Numerals as per the Noto Sans Format



(a) Samples from the K-MNIST dataset

(b) Samples from the dig-MNIST dataset

Fig. 2: Comparison of datasets

B. Development of various architectures

We initiated our exploration by training a relatively simple Sequential model consisting of three Convolutional layers [18], that were followed by Max-Pooling layers to reduce dimensionality. ReLU 1 was utilized as an activation function in all of the hidden layers, introducing non-linearity. This architecture also integrated Dropout and Batch Normalization layers for regularization. Finally, two dense layers were employed at the end to facilitate classification, where soft-max 2 activation function was utilized.

In our model, we apply utilized the vanilla ReLU as shown below:

$$\text{ReLU}(x) = \max(0, x) \tag{1}$$

For the classification layers, the Softmax activation function was employed. For a vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, the Softmax function is defined as:

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad \text{for} \quad i = 1, 2, \ldots, n \tag{2}$$

The training process for this network involved the utilization of the Adam optimizer 4 for optimal convergence & categor-

ical cross-entropy loss function 3 to facilitate the multi-class classification.

$$\text{Categorical Cross-Entropy}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \quad (3)$$

where $\mathbf{y}$ is the true distribution and $\hat{\mathbf{y}}$ is the predicted distribution.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned} \quad (4)$$

where t is the time step, β_1 and β_2 are decay rates, g_t is the gradient, η is the learning rate, ϵ is a small constant, and θ_t are the parameters.

As our secondary goal was to attain the best possible performance out of the networks, we realized how important it was to investigate various architectural arrangements. We developed 10 more networks, each of which possesses a distinct function that enables us to explore how various design decisions affect the performance of the model. By carefully examining different techniques, we can learn about their advantages and disadvantages, which helps us develop a more sophisticated knowledge of the issue at hand & allow us to understand the datasets, and the subtle differences in them.

We conducted extensive experiments on a number of cutting-edge Convolutional Neural Network (CNN) designs and evaluated them critically. Table I provides a brief summary of the salient features of these architectures.
The stemming sections will expound upon the particulars of each architecture, examining its distinct attributes, design decisions, and performance results.

TABLE I: Models designed for this research

Model	Architecture
Model_1	6 Conv, 3 Max Pool, 2 Dropout & Batch Norm.
Model_2	11 Conv, 2 Residual, 4 Dropout & Batch Norm.
Model_3	6 Conv, 3 Max Pool, 4 Dropout & 4 Batch Norm with different hyperparameters.
Model_4	Same as Model_3, but differed in hyperparameters.
Model_5	7 Conv, 3 Max Pool with Dropouts and Batch Norm layers.
Model_6	Identical to Model_5, differed in hyperparameters.
Model_7	Identical to Model_3 with different hyperparameters.
Model_8	6 Conv, 1 Max Pool with different hyperparameters.
Model_9	Same as Model_2, with different hyperparameters.
Model_{10}	Same as Model_2, but trained with a few dig-MNIST images to prove a hypothesis.
Model_{11}	Same as Model_2 but with different hyperparameters.

IV. Analysis

In this section, we explore the subtleties of misclassifications that we saw when we evaluated all of the eleven architectures, (Model_1 Model_{11}) on the K-MNIST and dig-MNIST datasets. Our goal is to obtain important insights into the networks' strengths and limits by looking closely at the particular cases where they failed. Each dataset's unique patterns of misclassifications are used to segment the study and highlight areas that could be improved upon and refined in upcoming model iterations. Specifically, this detailed analysis enables us to thoughtfully take into account the unique qualities of the images found in the dig-MNIST dataset, offering insightful information that will help future model iterations be improved by gaining a deeper comprehension of these unique qualities.

Our initial model, Model_1 was trained for 20 epochs with the Adam Optimizer 4 & with a learning rate of 0.001.

A. *Model_1's misclassification analysis*

1) ***On the K-MNIST test misclassifications****:* Model_1 demonstrated a commendable accuracy rate of approximately 96.42% on the K-MNIST test set of 10,000 images, but closer examination unveiled instances of intriguing misclassifications. A detailed analysis on the predictions for the test K-MNIST dataset revealed the following results:

- The digit '0' was misclassified as '1' for 103 images, which was astonishing because '0' and '1' are totally distinguishable as can be seen in Fig. 4 by the roundness of the digit '0', while '1' isn't round at all at the bottom.
- The digit '1' was misclassifed as '0' for 56 images. This again was very interesting to see because of the above made argument between the strokes in '0' and '1'.
- The digit '5' was misclassified as '4' for 37 images, which again was astionishing because they both clearly differ in the strokes as can be seen in Fig. 5.
- The digit '6' was misclassified as '9' for 21 images. This wasn't as surprising to us, because both the digits are exactly similar as can be viewed in Fig. 6, except the fact that one of them has a straight line stroke above, while the other has it below.

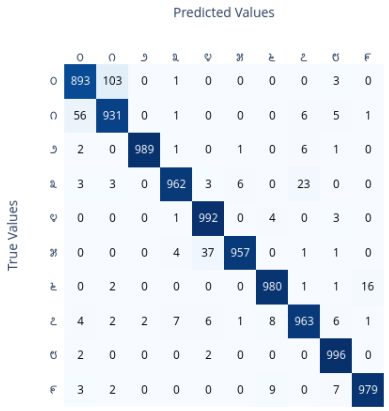
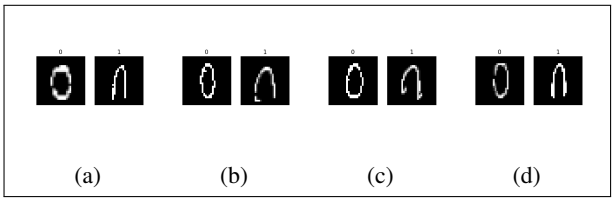
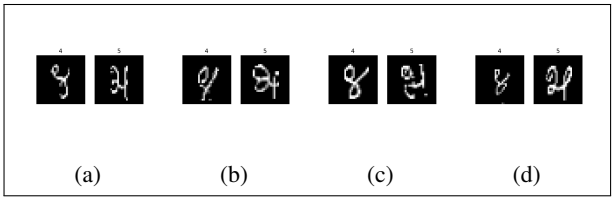


Fig. 3: Confusion Matrix for the test K-MNIST by Model_1



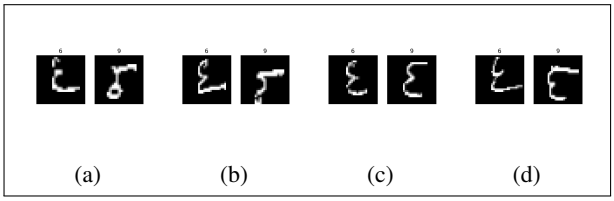
(a) (b) (c) (d)

Fig. 4: One vs Zero



(a) (b) (c) (d)

Fig. 5: Four vs Five



(a) (b) (c) (d)

Fig. 6: Six vs Nine

*2) **On the dig-MNIST Misclassifications**:* Similarly, upon testing Model_1 on the dig-MNIST dataset, despite achieving an accuracy of 80.19%, intriguing misclassifications were observed.

- Digit '0' was often misclassified as
  - '9' for 117 images
  - '8' for 76 images
  - '3' for 44 images
- Digit '1' was misclassified as
  - '0' for 368 images
  - '9' for 69 images
- and many more such instances that were very interesting to note can be seen in the Fig. 7.

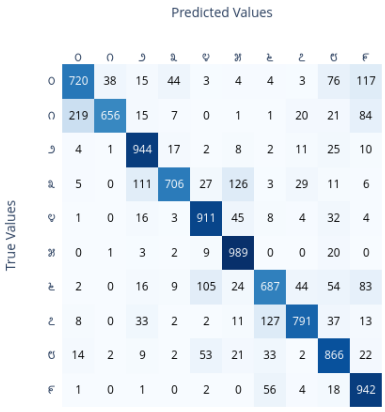


Fig. 7: Confusion Matrix for the dig-MNIST by Model_1

Realising that Model_1 has performed less well than ideal on the dig-MNIST dataset, we have opted not to do a thorough study of misclassifications from the standpoint of this specific model. We have made this decision based on our overall research strategy, which includes building cutting-edge networks iteratively. Building ever-better models naturally addresses and improves upon the shortcomings of earlier models, a methodology we passionately believe in.

Our methodology consists of utilising performance gaps to find areas that could be improved, which in turn leads to the creation of further models. As such, we move our attention from studying misclassifications in Model_1 to the more general setting of creating better models.

This tactical decision guarantees that our research efforts are focused on models that are improvements over the models that came before them, creating a thorough and forward-thinking framework for dataset comprehension and model improvement.

It is important to note that after building these upgraded architectures, we will go back and reexamine the misclassification study, looking at it through the lens of the most recent models to learn more about the subtle intricacies present in the datasets and to guide future improvements, as we firmly believe that if the most advanced models are unable to correctly classify some photographs, then those particular test samples should be investigated further.

B. Model_2 to Model_{11} misclassification analysis

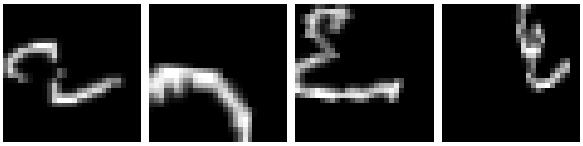
We built 10 models from scratch, among which 6 of them were distinct in their architectures, while the remaining 4 adopted one of the above 6 architectures, but they differed in the hyperparameters used while training them. The K-MNIST training dataset was split into training of 85% and validation of 15% of the total 60,000 samples. All the below mentioned models can be downloaded from the github page[1]. All the models below, except the Model_{11} were CNN architectures and trained with the below hyper-parameters:

- Adam Optimizer 4 with β_1 of 0.9 and β_2 of 0.999
- Learning Rate of 0.001
- Batch size of 64
- Kernel sizes of (3x3) and (5x5)
- Number of filters ranging from 32 to 64 for each Convolutional Layer
- ReLU activation function 1 throughout the network *except* at the last layer, where we employed soft-max 2
- Patience Parameter of 3

The networks were also trained with Augmented images, a few of which are shown in Fig. 8 from the training set of 60,000 images. Below were the configurations used for the Augmentation process for all the networks except Model_{11}:

- Zoom range of 0.05
- Rotation range of 10 degrees
- Width and height shifts of 0.2 times the image size of [28x28] pixels

[1] www.github.com/ShriHari33/EDA-on-dig-MNIST/tree/master



(a) Rotated digit '7' Anti-clockwise (b) Vertical shift & Zoom in on digit '0' (c) Horizontal shift of digit '6' (d) Clockwise rotation of digit '6'

Fig. 8: Augmented Images

Model_{11} was the last model that was trained a bit differently than the above. It differed in the below hyper-parameters:

- First 200 epochs with Adam Optimizer 4 with β_1 of 0.9 and β_2 of 0.999
- Next 200 epochs with Nadam Optimizer 5 with β_1 of 0.9 and β_2 of 0.999
- Epoch count of 200 inclusive of both Optimizers

The goal of moving the neural network towards the optimal point after the first 200 epochs with the Adam Optimizer 4 led to the choice to use two different optimizers for training. We took a calculated risk and got better results by adding the Nadam Optimizer 5 later on, which is renowned for promoting faster convergence when the starting point is close to the minima. These improved outcomes are examined in depth in the outcomes section that follows.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t}, \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \end{aligned} \tag{5}$$

where t is the time step, β_1 and β_2 are decay rates, g_t is the gradient, η is the learning rate, ϵ is a small constant, and θ_t are the parameters.

The below changes were made to the Data Augmentation in training the Model_{11}:

- Rotation range of 15 degrees
- Zoom range of 0.3 times the size of the image (28x28) pixels
- Vertical shift of images by 0.3 times the size of the image (28x28) pixels
- Horizontal shift of image by 0.2 times the size of the image (28x28) pixels

It was evident that all of the models show-cased in the Table I performed poorly on similar kind of images in the dig-MNIST dataset. This led us analyze those specific cases, and we are going to especially look those images from Model_2 & Model_{11}'s point of view, as they were able to provide the best accuracy on the dig-MNIST.

This process entailed a thorough examination of these occurrences in order to uncover commonalities or distinguishing qualities among them. We attempted to discover the fundamental patterns that contribute to misclassifications across several top-performing models using this technique.

The visualization and examination of misclassified samples enabled the identification of commonalities and distinguishing traits, shedding light on the mechanisms causing misclassifications. This analytical phase greatly improved our comprehension of the dataset's intricacies, providing useful insights into its complexities and assisting us in reaching educated judgments.

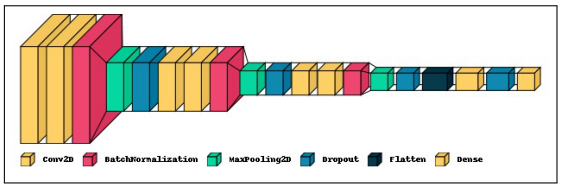


Fig. 9: Architectural view of Model_5

1) ***0 being predicted as 9 on dig-MNIST****:* On average, there are about 90 images predicted by the networks in Table I on which there is a consistent report of the image actually labelled as a '0', but the networks predict them as a '9'. This was not expected to be the case, as the digits '0' and '9' are very different in their strokes. But upon further examination on those wrongly predicted images by the model, we could immediately see where the problem was.

Several images, of which a few shown in Fig. 10 from the dig-MNIST were misclassified as '9' rather than '0' by the aforementioned networks. The probabilities with which the networks on average, confidently predict as a '0' or a '9' are annotated in the top section of the images.

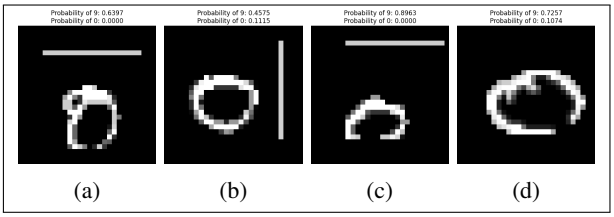


Fig. 10: Digit 0 predicted as a 9

We could easily see from this that the networks are strongly biased towards predicting *any horizontal line* at the top of the (28x28) grid of pixels as '9'. This is because the sole numeral having a horizontal line at the top in the Kannada Numeral System from 0-9 is '9'. Even to the human eye, we feel that the horizontal line acts as a instinctual cue, after having used Kannada extensively up-until our life. Throughout the training process, these networks learned to predict these specific images as '9', potentially due to the discovery of what we refer to as *shortcuts*. Despite the digit '0' context within these images primarily residing at the bottom, the model seems to have developed a bias.

For reference, we have included the probabilities with which it was forecasted as '0' and '9' in Fig. 10

*2) **1 being predicted as 0 on the dig-MNIST**:* There are on average about 50 in the dig-MNIST that are predicted as a '0' by the networks, where-as the true label is a '1'. When we looked into those images, it was quite evident as to *why* there was even a confusion to begin with, as '0' and '1' are very different, and easy to identify to a human-eye.

The images in Fig. 11 illustrate such wrong predictions by these networks. The probabilities with which the networks on average, confidently predict as a '0' or a '1' are annotated in the top section of the images.

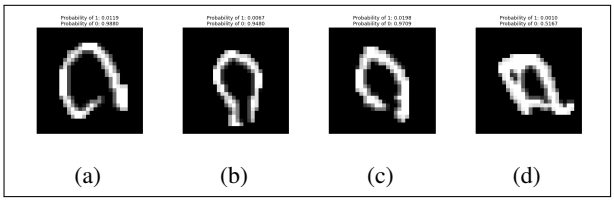


Fig. 11: Digit 1 predicted as a 0

These images on first sight, do seem to be of 1 except perhaps Fig. 11b and Fig. 11d. But when we account for the fact that the nature of these strokes are very different than what was provided to the networks during training as shown below in 12, we can cut a little slack to our networks and be sure that there isn't such a broad aspect of training images present in the dataset. The training images are distinguished very well, with the digit '1' having adequate space between the bottom two parallel lines, which makes the network see it as a '1'.

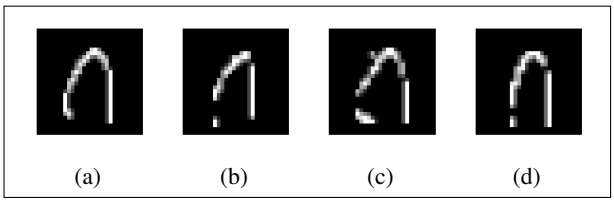


Fig. 12: Digit 1 provided during training

*3) **0 being predicted as a 1**:* There are on average 50 images in the test, and 35 in the dig-MNIST dataset that are predicted as a '0', rather than a '1'. We strongly believe that the vertical shift performed during the data-augmentation phase of training may have caused the networks a confusion in predicting either of the above digits. When we apply a vertical down-shift to an image of a '0', it causes the bottom stroke to completely vanish in some of the images, especially those having the curve stroke very near at the bottom.

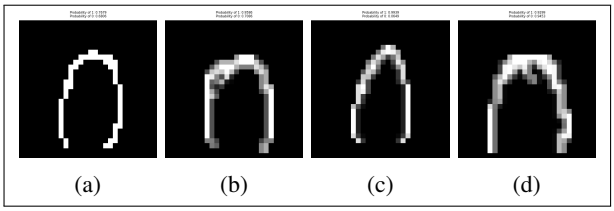


Fig. 13: Wrong Predictions on test dataset

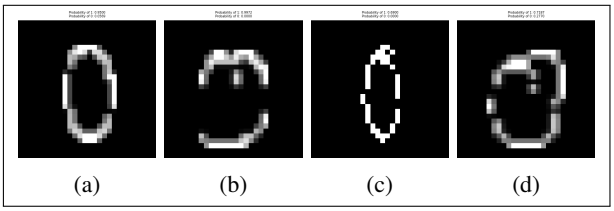


Fig. 14: Wrong Predictions on dig dataset

- The images in Fig. 13 posed a big question to us. They definitely do look like a '1', but they are labelled as '0' in the test dataset. We deduced that perhaps the bottom stroke were cut, which makes the resulting image resemble as a '1' due to the absence of the bottom stroke, due to which the prediction is made as '1'. We think that if there are genuinely such situations, then we definitely need to provide the network to learn such mappings during training too.
- The images in Fig. 14 definitely do look like a '0', so we were confused as to *why* the networks predicted them as a '1'. We think that the spacing between the parallel lines is what makes the network assume that it is a '1'. Images (a) and (c) have a relatively small distance between the lines, which in training, the zero's are much wider, and don't usually have such small gaps. Images (b) and (d) although demonstrate that the distance is enough, present themselves with noises in parts that are not provided during training. Combined with the fact that, the top stroke and bottom stroke are separated with a *whitespace*, makes it believe that it is a '1', because '1' is the only image that has that distinct nature in the Kannada Numeral system.

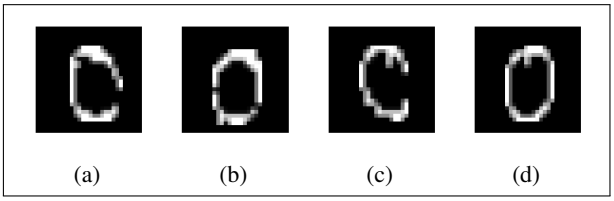


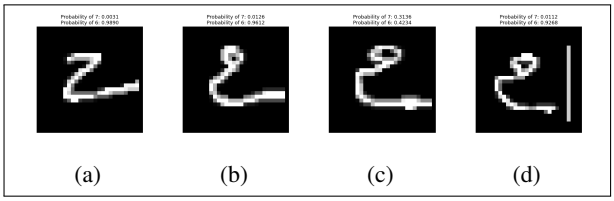
Fig. 15: Digit 0 provided during training

*4) **7 being predicted as a 6**:* An interesting observation about the misclassification of some images labelled as '7' and projected as '6' by the models is made when the dig-MNIST dataset is analysed effectively. In particular, the models misidentify the number '7' as '6' on average in about 130 unique cases. An interesting trend emerges from further analysis of these misclassifications: the training photos of the number seven usually have a characteristic loop at the top of the numeral, which is not present in the matching '7' images from the dig-MNIST dataset.

This difference in the top loop's presence appears to be a major factor in the models' inaccurate predictions. The absence of this loop in the dig-MNIST photos forces the models to depend on previously learnt patterns, which eventu-

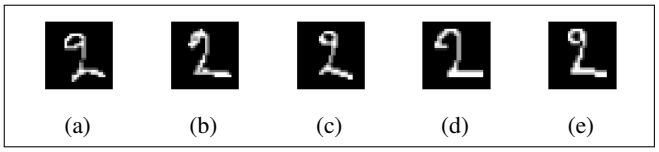
ally makes them hallucinate and predicts these occurrences as '6'. These photos are misclassified in part because of their sensitivity to minute pixel changes in the surrounding regions. The significance of varied and representative training datasets is highlighted by this nuanced understanding, which also highlights the models' limits when faced with differences in stroke patterns.

There's a fascinating twist to this: if we were to fictitiously alter these precise '7' photos by eliminating a few vertical pixels at one location of (3x3) grid, the result is highly perceivable as a '6'. This is *why* the networks predict these images as a '6', and not just any other numeral. A visual picture is shown in Fig.19a, where the red-region indicates the pixels to be removed.



(a) (b) (c) (d)

Fig. 16: Digit 7 predicted as a 6

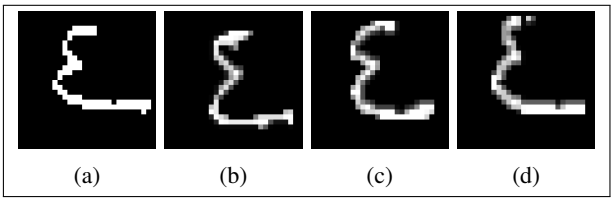


(a) (b) (c) (d) (e)

Fig. 17: Digit 7 provided during training

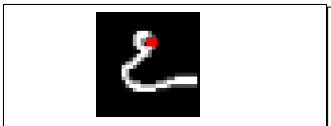
We also notice another interesting bit of information: the curve connecting the top of the image to the bottom of it has distinct characteristics in the training and dig-MNIST datasets i.e., the training images as shown below in Fig. 17 have a very steep curve that connects the top to the bottom, where-as the dig-MNIST images as shown in Fig. 16 have an abundantly lesser steepness for that same curve, which we think is what confuses the model. This same steepness is very evident in the images of '6' in the training, which combined with the above mentioned characteristics make the networks' predict it as a 6! A visual depiction is provided in Fig. 19b, where adding pixels to the red-region makes it just like a seven, almost as if completing the loopy pattern for the digit '7'!

This discovery emphasises how sensitive the models are to small changes and how much they depend on observable patterns, even in situations when minute differences are crucial to classification judgements. As our goal is to have a generalizable model, we need to ensure that the networks train extensively on this minute (3x3) pixel grid as mentioned above, of (28x28) pixels.



(a) (b) (c) (d)

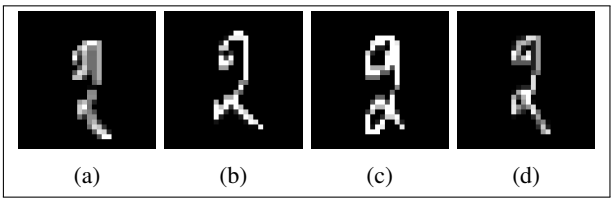
Fig. 18: Digit 6 provided during training



(a) Removing pixels from the "red" area for the Fig. 17b

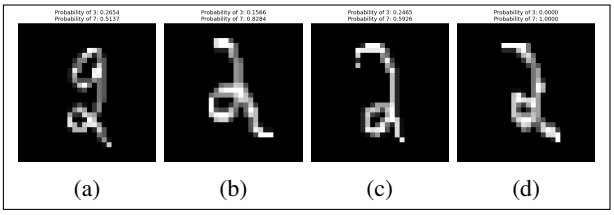
(b) Adding pixels in the "red" area for the Figure. 18a

5) ***3 vs 7:*** A handful of images labelled as a '3', around 70 on average are predicted as a '7' by the networks. Although '3' and '7' share quite similar characteristics, they really do differ in the additional loopy pattern and a different curvy pattern at the bottom that is present in a '3', and not in '7'. This is evident in training data too, apart from a few anomalies, which can be seen here 20 of training images of '3'.



(a) (b) (c) (d)

Fig. 20: Digit 3 provided during training



(a) (b) (c) (d)

Fig. 21: Digit 3 predicted as a 7

The network classifies the images depicted in Fig. 21 as '7'. While the presence of a loop at the top in Fig. 21a might explain the model's inclination to predict it as a '7', it's noteworthy that the other images in Fig. 21b, Fig. 21c and Fig. 21d lack this loop, and still receive a '7' prediction. This behavior can be attributed to the model's lack of emphasis on the distinctive bottom loop characteristic of '3' and absent in '7'. Referring Fig. 17 and Fig. 20, it becomes evident that the model's focus during training may not sufficiently capture the nuances of the bottom loop, leading to misclassifications in unseen instances.

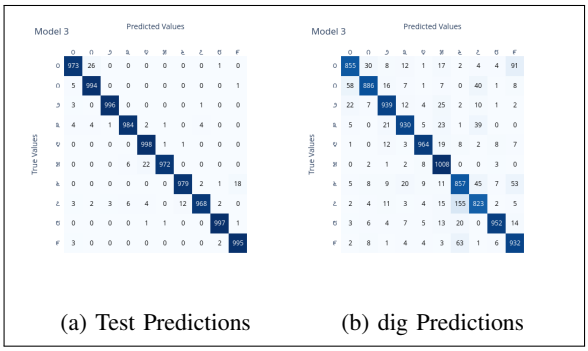
The examination of the dig-MNIST dataset in comparison to the K-MNIST dataset utilised for network training highlights its unique features. The previous analysis and visual representations provided insight into some of the notable "noise" found in dig-MNIST—valid problems that mimic real-world circumstances, particularly in Handwritten Recognition (HWR), including handling dirt buildup on vehicle number plates.

It is important to recognise that the analyses and visualisations offered provide insights into a limited portion of the dig-MNIST dataset's test cases. There are many more examples in the collection, all of which add to its distinct features. Although a thorough investigation of particular cases has been conducted, a more thorough comprehension is shown through the analysis of confusion matrices. These matrices go beyond the particular situations that were previously mentioned and offer insights into a variety of patterns and difficulties inside dig-MNIST.

V. Results

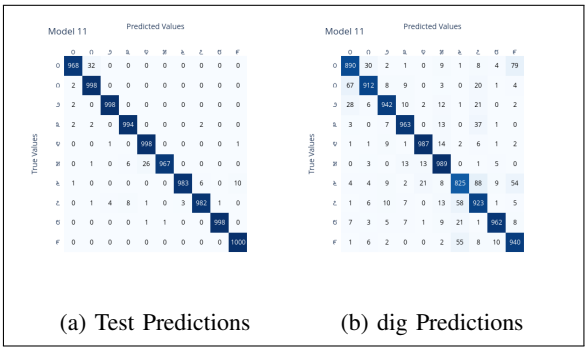
In this section, we deal with the performance metrics of the various networks show-cased in Table I. The accuracy metrics presented in Table II highlight the superior performance of Model_{11} on the dig-MNIST dataset. Due to the use of dual optimizers, Adam 4 and Nadam 5 during training, this noteworthy accomplishment may be understood. The model performed exceptionally well overall since the first 200 epochs of training with the Adam Optimizer gave the subsequent Nadam training phase a strong basis. Fig. 24 provides a visual view on the accuracies of all the Models on the test K-MNIST & the dig-MNIST datasets. Fig. 25 showcases the fact that there wasn't an overfit during training of Model_{11}.

A thorough understanding of Model_3's and Model_{11}'s performance is provided by their confusion matrices in Fig. 22 & Fig. 23, which shows a common propensity to incorrectly identify particular image classes. This pattern of persistent misclassification highlights the unique difficulties presented by the dig-MNIST dataset. It also highlights the need for sophisticated techniques to enhance the model's capacity for generalisation, especially in cases when the main training data comes from the K-MNIST dataset. Resolving these issues is essential to improving prediction accuracy using the dig-MNIST dataset.



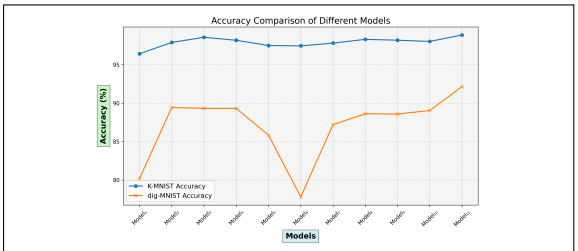
(a) Test Predictions (b) dig Predictions

Fig. 22: Confusion Matrices for Model_3's predictions

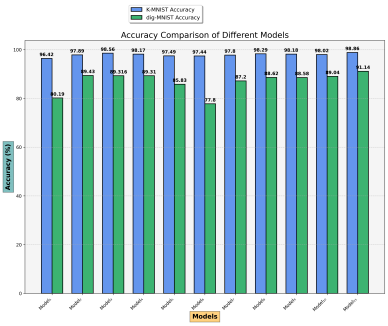


(a) Test Predictions (b) dig Predictions

Fig. 23: Confusion Matrices for Model_{11}'s predictions



(a) Accuracy Trend for the various trained Models



(b) Comparison on the K-MNIST & dig-MNIST accuracies on the trained Models

Fig. 24: Accuracy graphs for all the trained models

In order to attain our secondary objective of improving accuracy, we employed an ensembling technique, which involves predicting separately on the dig-MNIST first for the Model_3 and Model_{11}, and then executing a weighted arithmetic mean to allow for improved outcomes. The reason for this was that Model_3 performed better in predicting an actual '6', whereas Model_{11} projected it as a 7, which is twice as large as what Model_3 anticipated. However, because Model_{11} performed better overall, we chose to give its predictions a higher weightage. As a result, we multiplied Model_3's predictions by 0.8 and Model_{11}'s predictions by 1.2, resulting in a SOTA accuracy of 91.425%.

TABLE II: Summary of Models with Description

Model	Epochs	Additional Hyperparameters description	K-Acc (%)	D-Acc (%)	Summary
M_1	20	Adam, LR=0.001	97.88	89.43	Base model. Good D-Acc.
M_2	100	-	97.89	89.43	Specialized with Residual layers.
M_3	100	Employed filter size of (7x7) to facilitate better learning	98.56	89.32	Improved accuracy on test dataset.
M_4	50	-	98.17	89.31	Similar accuracy with longer training.
M_5	50	-	97.49	85.83	Lower accuracy with larger kernel.
M_6	30	-	97.44	77.80	Lower accuracy due to fewer epochs.
M_7	50	-	97.80	87.20	Base model with slight D-Acc decrease.
M_8	100	Increased filter count at each layer to 128	98.29	88.62	Improved accuracy with more filters.
M_9	200	-	98.18	88.58	Longer training with slightly lower D-Acc.
M_{10}	100	-	98.02	89.04	Good accuracy on tampered images.
M_{11}	200	Nadam, LR=0.001, Epochs=200	98.86	91.14	Best accuracy achieved.

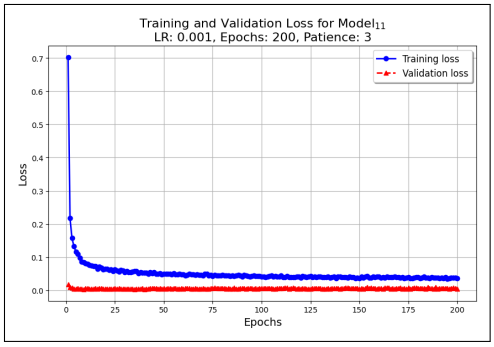


Fig. 25: Training & Validation Loss Curves for Model_{11}

VI. Conclusion

In this research, we have concluded that the dig-MNIST dataset is vastly different from the training K-MNIST dataset, which makes it an incredibly hard task to build a network that is capable to generalize on both the datasets. We could see from the Analysis Section IV the subtle changes that the image possesses which makes the networks go haywire and predict something else entirely. Combined with the fact that the grid is of (28x28) pixels, it just bumps up the complexity even more. For example, just a few pixels off in Fig. 19a, the image can easily be misinterpreted as a '6' than a '7'. Similarly, in Fig. 19b, adding just a few pixels changes the interpretation to a '7' than a '6'. Many have been listed above, but the point is, these attributes cause a lot of trouble because the networks are not given any such noisy, out-of-domain images.

Our research has revealed a notable difference between the training K-MNIST dataset and the dig-MNIST dataset, which poses a serious obstacle to the creation of a neural network that can generalise effectively across both datasets. The subtle differences in the images that confuse neural networks and lead to incorrect predictions are clarified in the Analysis Section IV. The fact that there are so many of these options indicates how complex the task is. Only a handful of them that we found interesting were listed in this paper, but there exists several more that can be inferred from the Confusion Matrices in Fig. 22 & Fig. 23. The (28x28) pixel grid makes things more difficult since even small variations, as seen in Fig. 19a and Fig. 19b, can cause misunderstandings and create it more difficult to tell apart '6' from '7', and vice-versa.

The result of our research also ended up with an ensemble model of weighted averages that was trained just on K-MNIST images, but it was able to achieve an accuracy of 91.45% on the dig-MNIST dataset. We stress that the insights obtained from our exploratory data analysis (EDA) offer useful optimisations for dealing with this dataset, even though this represents a minor increase of only 1.1% over the previous state-of-the-art model by Qisheng Hu [15].

This dataset has mostly been examined by people—including our research team—using CNN models. But we also support a change in the direction of taking our EDA results into account when building models, particularly in the transformative domain of transformer architectures. We suggest that using these ideas may result in better outcomes. Whether utilising a transformer design or incorporating our EDA results into any architecture that is created, we think that paying close attention to these subtleties can help improve the model's overall prediction performance by making it more sensitive to minute features.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep Residual Learning for Image Recognition*, *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778. DOI: https://doi.org/10.1109/CVPR.2016.90

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998. DOI: https://doi.org/10.1109/5.726791

[3] Wikipedia, *Kannada*, https://en.wikipedia.org/wiki/Kannada.

[4] Yoav Freund and Robert E. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, *Journal of Computer and System Sciences*, 1997.

[5] Vinay Uday Prabhu, *Kannada-MNIST: A new handwritten digits dataset for the Kannada language*, arXiv preprint arXiv:1908.01242, 2019.

[6] A. Saini, S. Daniel, S. Saini, and A. Mittal, *KannadaRes-NeXt: A Deep Residual Network for Kannada Numeral Recognition*, in *Proceedings of the International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI)*, 2021. DOI: https://doi.org/10.1007/978-981-15-9492-2_4

[7] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated Residual Transformations for Deep Neural Networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. DOI: https://doi.org/10.1109/CVPR.2017.634

[8] G. Upadhye, U. V. Kulkarni, and D. Mane, *Improved Model Configuration Strategies for Kannada Handwritten Numeral Recognition*, *Image Analysis & Stereology*, vol. 40, pp. 181-191, 2021. DOI: https://doi.org/10.5566/ias.2586

[9] J. Kennedy and R. Eberhart, *Particle swarm optimization*, Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4. DOI: 10.1109/ICNN.1995.488968

[10] Akshitha L S, *Kannada MNIST Digit Recognition Using CNN and RNN Model*, *International Advanced Research Journal in Science, Engineering and Technology*, 2022, vol. XX, pp. YY. DOI: YourDOIHere

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, *Nature*, vol. 323, no. 6088, pp. 533-536, 1986. DOI: https://doi.org/10.1038/323533a0

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. DOI: https://doi.org/10.3115/v1/D14-1179

[13] Anirudh Ganesh, Ashwin Jadhav, K. Pragadeesh, *Deep Learning Approach for Recognition of Handwritten Kannada Numerals*, in *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence (PReMI)*, 2018, pp. 294-303. DOI: https://doi.org/10.1007/978-3-319-60618-7_29

[14] Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006. DOI: https://doi.org/10.1162/neco.2006.18.7.1527

[15] Q. Hu, *Evaluation of Deep Learning Models for Kannada Handwritten Digit Recognition*, in *2020 International Conference on Computing and Data Science (CDS)*, Stanford, CA, USA, 2020, pp. 125-130. DOI: https://doi.org/10.1109/CDS49703.2020.00031

[16] 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*, Springer, 2007, pp. 153–160.

[17] GG Rajput, Rajeswari Horakeri, and Sidramappa Chandrakant, *Printed and handwritten mixed Kannada numerals recognition using SVM*, *International Journal on Computer Science and Engineering*, vol. 2, no. 05, pp. 1622–1626, 2010. DOI: DOI_here

[18] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.