ANCIENT LANGUAGE HANDWRITTEN RECOGNITION UNDER DATA DEFICIENT CONDITION

Sujit Amin, Darshan Rao, Krishna Wankhede, Arnav Kumar, Tetsuro Kai
Department of Computer Science
University of Southern California
Los Angeles, CA 90007, USA
`{sujitami, raod, wankhede, kumararn, tkai}@usc.edu`

1 INTRODUCTION

Major headway has been made in the field of handwritten character recognition for widely spoken languages (Cireşan et al., 2010). However, this progress poses challenges for many languages due to the limited availability of sufficiently large and annotated datasets required for model training (Krizhevsky et al., 2012). Various methods have been employed in character recognition, each with its methodology and associated issues. CNNs show 99% accuracy in digit recognition using MNIST dataset, but may struggle with complex character sets or limited training samples due to their high data requirements. Advanced neural architectures, such as bidirectional networks have integrated innovative techniques like style memory and dynamic routing to achieve low error rates on MNIST. Nevertheless, their reliance on extensive datasets for training constrains their effectiveness, particularly in character recognition tasks with minimal samples. One-shot learning models like Siamese networks are limited for character recognition tasks, while Generative Adversarial Networks and Variational Autoencoders are suitable for data generation but struggle with labeling and class perturbation.

The primary goal of this project is to significantly improve the accuracy of handwritten character recognition for ancient indian languages under very less data samples per class. Since these languages have historically faced challenges due to the scarity of labeled data. We aim to develop an innovative approach that leverages Capsule Networks to address this issue and achieve state-of-the-art performance with 190 training samples per character. By doing so, we seek to enable more comprehensive analysis and interpretation of historical texts. This project holds considerable importance as it addresses a longstanding problem in the field of character recognition. Ancient languages, being a crucial part of our cultural heritage, often lack extensive labeled datasets. Consequently, existing methods struggle to provide accurate character recognition, impeding historical research and preservation efforts. In our research, we introduce a novel approach to address the issue of limited dataset sizes using capsule networks (Sabour et al., 2017a). We leverage Capsule Networks capacity to augment data through manipulation of instantiation parameters (Hinton et al., 2011). CapsNets not only learn an image's presence but also its attributes, making them advantageous for character recognition with limited data. Our design modifies the capsule network architecture by substituting the decoder network from Sabour et al. (2017a) with a deconvolutional network and making adjustments to the capsule network. By introducing a controlled level of noise to instantiation parameters, we simulate human like realistic variations, leading to a more effective data generation method compared to traditional affine transformations. Our system closely matches state-of-the-art results with just 190 data points per character.[1]

The main contributions of our study are:

1. Achieving leading results on Sanskrit, Tamil, and EMNIST-digits character datasets, training our system with 190 training samples per class.
2. Developing a method to train capsule networks effectively with as few as 190 samples per class, maintaining test sample consistency, and achieving top-tier performance.

[1]Code can be found at: https://github.com/sujitaminusc/ancient-hand-writtten-recognition-under-less-data-condition-csci-567

3. Analyzed performance with different loss functions to establish the best loss function combination.

2 Related Work

Sabour et al. (2017a) introduces a multi-layer system for recognition, it is relevant for improving character recognition through advanced neural networks. Hinton et al. (2011) proposes neural networks learning parameters for handling variations, it aligns with your approach of data augmentation for character recognition. Lecun et al. (1998) highlights the effectiveness of CNNs in character recognition, this provides insights for improving accuracy. Ciresan et al. (2012) presents biologically inspired neural network architectures for image recognition, this offered innovative perspectives for character recognition. (Labusch et al., 2008) proposes a feature extraction method for digit recognition utilizing scarce coding and local maximum operations, achieving state-of-the-art results on MNIST, it is relevant for enhancing character recognition in our model. Ranzato et al. (2006) presents an unsupervised method for learning sparse, over-complete features with applications in stroke detection and image patches, which offered insights for improving feature extraction in character recognition. Jarrett et al. (2009) explores the influence of non-linearities in feature extraction, investigates supervised and unsupervised filter learning, and suggests the advantage of two-stage feature extraction, providing valuable considerations for your character recognition approach. Cohen et al. (2017) introduces the Extended MNIST (EMNIST) dataset, an expansion of MNIST involving letters and digits, which is the dataset for benchmarking and evaluation the character recognition models. Dufourq & Bassett (2017) introduces the EDEN algorithm for evolving deep neural network architectures efficiently, showcasing its potential for improving classification tasks, including sentiment analysis, this is applicable in character recognition that we built. Wiyatno & Orchard (2018) presents a network with style memory for classification and reconstruction tasks, exploring bidirectional information flow and its potential in mitigating ambiguous or adversarial input, which is used in our character recognition model. Goodfellow et al. (2014) proposes an adversarial framework for generative models, where a generative model is trained to capture data distribution while a discriminative model distinguishes training data from generated data, offering a novel approach to data generation, this was used as one of our experiment before working on current architecture. Doersch (2021) discusses Variational Autoencoders (VAEs) as a popular approach for unsupervised learning, emphasizing their effectiveness in generating various types of data, including handwritten digits, which is relevant for character recognition like ours. Zeiler et al. (2010) presents a learning framework for extracting mid-level image representations based on convolutional decomposition under sparsity constraints, offering an unsupervised approach for robust image analysis, we used this it enhance our character recognition features. Polesel et al. (2000b) introduces an adaptive unsharp masking method for contrast enhancement in images, which we used to improving the visibility of characters in character recognition tasks, particularly for enhancing high detail areas.

3 Scope and structure of the implementation

3.1 Scope

The implementation of this project is limited to 3 datasets Sanskrit, EMNIST-Digits and Tamil. Out of which Sanskrit and Tamil is ancient Indian languages. Our Tamil (Sudalairajkumar, 2020) dataset [2] and Sanskrit (Pant et al., 2012) dataset [3] are taken from kaggle, along with the EMNIST-Digits (Cohen et al., 2017) dataset [4]. Additionally, this project is scoped to building near state-of-the-art accuracy model on multiple datasets including the ancient languages. We included the EMNIST-Digits datasets in addition to the Sanskrit and Tamil datasets to test the robustness of our model. We have used Python and Tensorflow for implementing this project.

[2]Tamil dataset: https://www.kaggle.com/datasets/sudalairajkumar/tamil-nlp

[3]Sanskrit dataset: https://www.kaggle.com/datasets/ashokpant/devanagari-character-dataset/data

[4]EMNIST dataset: https://www.nist.gov/itl/products-and-services/emnist-dataset

3.2 High Level Structure Of the Code

Firstly, there are 2 components in the code first is for generating the instantiation parameters (for classifying the data into several classes) and second is decoder network (for generating images). These 2 are interconnected. For developing the capsule network and decoder network we have taken the reference from Sabour et al. (2017b) research. We have modified this work from the original paper since the original paper focused on implemented a transforming auto-encoder which converted the classes to 1D array which was reducing the accuracy of our model. Architecture in Sabour et al. (2017b) research served a different purpose. Instead we added a de-convolutional layer which suites the purpose of generating the human like images for training the character recognition model. Here the classes represent the every character present in the language. Initially we train the capsule network and decoder network model using 190 samples per class. This generates the instantiation parameters.

Then we train generate image using our perturbation algorithm (controlled noise), from here we noticed that using just 190 samples the images that are generated by decoder network are blur and this can't be used for used for training our model. Therefore, we apply unsharp masking (Polesel et al., 2000a) to enhance the clarity of the images. Once we generate images this is used for training the both capsule network and decoder network. This results in near state-of-the-art accuracy model.

Note: 190 samples per class is the lower bound we ran several experiments with this architecture. We first start with around 2000 samples per class. Then we kept on lowering the samples until we hit the lower bound which is 190 samples. Anything lower than this number gives significantly low accuracy.

3.3 Components Of the Code

3.3.1 Dataset Description

Sanskrit dataset comprises three categories - Numerals, Vowels, and Consonants - with samples collected from 40 individuals across various fields, featuring 288 samples per class for 10 numeral classes, 221 samples per class for 12 vowel classes, and 205 samples per class for 36 consonant classes, all meticulously cropped to focus on character boundaries. The Tamil dataset consists of 156 characters, including 12 vowels, 18 consonants, 216 compound characters formed by combining vowels and consonants, and 10 unique digits for numbers. Finally, The EMNIST digits constants of 10 digits.

3.3.2 Capsule Network and Convolutional neural network

Convolutional neural network is an important component of this project. It is excellent for when the data involves images. Capsule Network has 3 convolutional neural network with kernel size of 3. Activation function of RELU. Next layer is primary capsule layer of 8 convolutional units and 32 channels. Output of this layer is the instantiation parameters.

3.3.3 Decoder network

Output of the capsule network (instantiation parameters) is used as an input to the decoder layer. First, we do the masking. We have used sequential decoder. First layer is a dense deconvolutional layer with activation function of RELU. Then we do batch normalization. Next 5 layers are de-convolutional layer. First 4 has activation function of RELU and last one has activation function of Sigmoid. Since we noticed Sigmoid was able to increase the accuracy of our model.

3.3.4 Perturbation Algorithm in Decoder Network

We are generating images by using our perturbation algorithm. Here we calculated the the variance of each class (every character in the dataset). Then we take the average of the variance. This become the higher bound for adding noise to our original image. Without this upper bound the several images generate will be distorted.

Note: here also we ran several experiments on this architecture and found out that by calculating the variance and taking the average of the variance gives us upper bound for adding noise necessary for creating images and avoiding distorted images.

3.3.5 Unsharp masking

Unsharp masking is to sharpen the decoder network generated images. We do un-sharp masking using radius of 1 and threshold of 1 as mentioned in the Polesel et al. (2000a).

4 Methodology

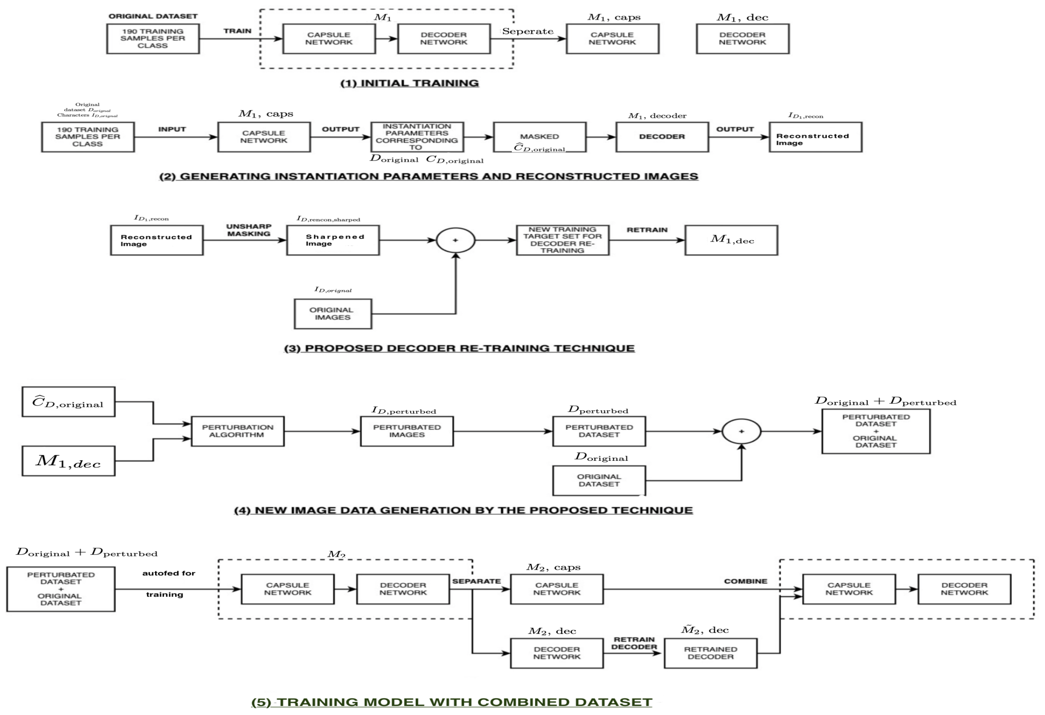


Figure 1: Overall Architecture

Figure 1 illustrates our overall architecture.

4.1 Capsule Network (used for Character Recognition)

The baseline character recognition model is meticulously crafted with a hierarchical architecture, seamlessly integrating the strengths of both convolutional and capsule layers. The initial layers, serving as adept feature extractors, are composed as follows: Layer 1 with 64 filters of 3x3 kernels and a stride of 1, Layer 2 with 128 filters of 3x3 kernels and a stride of 1, and Layer 3 with 256 filters of 3x3 kernels and a stride of 2. These convolutional layers play a pivotal role in capturing intricate hierarchical features present in the input images. Following this, the fourth layer is a primary capsule layer, featuring 32 channels of 8-dimensional capsules. Each primary capsule encapsulates 8 convolutional units armed with a 9x9 kernel and a stride of 2, dedicated to capturing spatial hierarchies within the input features. Moving forward, the fifth layer embodies the character capsule layer, functioning as a fully connected capsule layer. This layer comprises 16-dimensional capsules per class, resulting in M capsules for M classes in the dataset. The implementation of dynamic routing between the primary and character capsule layers, as proposed by Sabour et al. (2017b), employs 3 routing iterations to establish connections. The ultimate output manifests as a J × M × 16 dimensional tensor C, housing instantiation parameters for each training sample. This intricate network structure showcases the model's prowess in hierarchically discerning and recognizing characters, promising enhanced performance in character recognition tasks.

4.2 Decoder Network Architecture

The decoder network plays a pivotal role in the baseline model, tasked with reconstructing images from the instantiation parameters generated by the capsule network. Comprising a structured architecture, the decoder consists of a singular fully connected layer followed by five deconvolutional layers. The unique design of the decoder aligns with the hierarchical nature of the overall model, allowing it to efficiently translate the information encapsulated in the instantiation parameters into a meaningful visual representation. The input to the decoder is a masked instantiation parameter matrix specific to the true class, ensuring targeted reconstruction. This targeted approach facilitates the model's ability to reconstruct character images accurately, contributing to the overall efficacy of the character recognition system.

4.3 Unsharp Masking for Image Enhancement

To further refine the reconstructed images, the baseline model employs an iterative process known as unsharp masking (Polesel et al., 2000a). This enhancement technique is applied individually to each reconstructed image, employing specific parameters: a radius of 1, a threshold of 1, and an unsharp strength increased by a factor of 10. The primary objective of unsharp masking is to address blurriness observed in scenarios with limited training samples. By iteratively enhancing the clarity of reconstructed images, the model mitigates potential distortions introduced during the reconstruction process. This meticulous approach ensures that the decoder network not only reconstructs character images accurately but also actively addresses and rectifies any loss of fine details or blurriness, ultimately contributing to the model's robustness in low-sample situations.

4.4 Reconstruction of Images Using Capsule Networks

The comprehensive workflow unfolds in twelve meticulously orchestrated steps aimed at refining the character recognition model and enhancing its capacity for reconstruction. In the initial phase, Model (M_1) undergoes training on the full training sets. Following this, a reduced dataset (D_{original}) with 190 training samples per class is employed to extract instantiation parameters ($C_{D,\text{original}}$) using the trained capsule network ($M_{1\text{caps}}$). The parameters are then masked to retain only those corresponding to the true class ($\hat{C}_{D,\text{original}}$). Subsequently, the decoder network ($M_{1\text{dec}}$) is engaged to reconstruct images ($I_{D,\text{recon}}$) from the masked instantiation parameters, with an evaluation of reconstruction quality focusing on issues such as blurriness and fine detail preservation.

The workflow seamlessly progresses with an image enhancement step utilizing unsharp masking, where specific parameters are applied to improve clarity. The sharpened images ($I_{D,\text{recon,sharpened}}$) are then combined with the initial reconstructed images, forming a new target set for retraining the decoder network ($M_{1\text{dec}}$). This retraining, executed over a set number of epochs, aims to bolster the decoder's proficiency in generating sharper images.

Pivoting towards perturbation, controlled random noise is introduced to non-zero instantiation parameters in $\hat{C}_{D,\text{original}}$. Careful consideration is given to perturb one instantiation parameter at a time to avoid distortions. The subsequent steps involve calculating and constraining noise values, ensuring that the perturbations remain within acceptable bounds. The perturbed instantiation parameter is then passed through the retrained decoder, generating new reconstructed image ($I_{D,\text{perturbed}}$). This iterative process culminates in combining the original and perturbed samples, forming a new target set for the subsequent model iteration ($M_{2\text{dec}}$).

The final steps of the workflow involve training the ultimate model (M_1) using the combined dataset ($D_{\text{original}} + D_{\text{perturbed}}$). The retrained decoder undergoes further refinement, and the culmination of this intricate process results in a robust character recognition model poised for advanced classification tasks. This comprehensive novel approach leverages iterative refinement, controlled perturbation, and advanced training techniques to push the boundaries of reconstruction quality and model performance in character recognition.

4.5 Perturbation Algorithm

For each instantiation parameter representing a specific feature of a character ($C_{m,k,j}$), we introduce controlled random noise. This noise is added to each parameter individually, creating a perturbed instantiation parameter ($C_{m,k,j}^{perturbed}$). This step is crucial for generating variations in the original instantiation parameters. To select instantiation parameter we calculate the variance ($\sigma_{m,k}$) for each instantiation parameter ($C_{m,k,j}$) within each class (m). Variance reflects how much the values of an instantiation parameter vary across different samples within the same class. Parameters are then sorted based on their variance in descending order, giving priority to those with more significant variations. Then to calculate and constrain noise values we do for each instantiation parameter k within each class m, the first step is to identify the maximum noise that can be added, denoted as $(\tau_{m,k})$. This is achieved by considering the perturbed values within the class. Once these maximum noise values are identified for each class, the next step involves averaging these values across all classes to obtain τ_k, which represents the overall constraint on the magnitude of noise that can be added. Finally, it is crucial to ensure that the added noise for any instantiation parameter (k) does not exceed the calculated τ_k. This is important to prevent abrupt variations in the parameters, maintaining consistency and stability in the system.

5 Results and Discussion

For each dateset in Table 1, we train our model on 190 training samples per class.

Table 1: Dataset Summary

Dataset Name	**Number of Classes**	**Samples taken per Class**	**Total Training Samples**
Sanskrit	58	190	11,020
Tamil	156	190	29,640
EMNIST-Digits	10	190	1,900

5.1 Handwritten Character Recognition

In this section, we present a comparison of our results with the current state-of-the-art accuracy's (Table 2). We included the results that we obtained with limited training samples (190 samples per class). Despite the limited number of training samples, our results are comparable to state-of-the-art model accuracy.

In the case of Sanskrit that model is able to perform close to state-of-the-art accuracy model with accuracy of 95.27%. In the case of EMNIST-digits dataset the model also perform very close to state-of-the-art accuracy. We introduced EMNIST-digits dataset to show robustness of our model. Finally, in Tamil model we are able to achieve 95.86% accuracy.

Table 2: Model Performance Results

Dataset Name	**State-of-the-Art Model Accuracy (full-training set)**	**Our Model (190 Samples/Class)**
Sanskrit	98.47% Acharya et al. (2015)	95.27%
EMNIST-Digits	99.79% Dufourq & Bassett (2017)	99.64%
Tamil	98.04% Ulaganathan et al. (2020)	95.86%

6 Conclusion and future work

This project has primarily focused on the challenge of handwritten character recognition for ancient languages, which typically suffer from limited data availability. By leveraging a novel approach tailored for these data-scarce conditions, we have successfully demonstrated the capability of our model on languages such as Sanskrit and Tamil. Our work highlights the potential of using limited datasets to effectively train models for accurate character recognition in ancient languages, a significant step forward in the domain of digital preservation and analysis of historical texts.

For future work, we plan to extend the application of our technique to other languages with limited datasets and RGB images. This includes exploring its effectiveness on a wider range of ancient and regionally localized indian languages, which often suffer from a scarcity of data. Additionally, we aim to investigate the adaptability of our method to various writing styles and fonts. This could involve analyzing the technique's performance on different script types such as cursive, block, and decorative styles, further enhancing its applicability and robustness. Probably by more carefully adding noise to our images and using a better unsharp masking algorithm, we could achieve a better accuracy.

References

Shailesh Acharya, Ashok Kumar Pant, and Prashnna Kumar Gyawali. Deep learning based large scale handwritten devanagari character recognition. In *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–6, Kathmandu, Nepal, 2015. IEEE. doi: 10.1109/SKIMA.2015.7400041.

Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012. URL `http://arxiv.org/abs/1202.2745`.

Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, December 2010. ISSN 1530-888X. doi: 10.1162/neco_a_00052. URL `http://dx.doi.org/10.1162/NECO_a_00052`.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

Carl Doersch. Tutorial on variational autoencoders, 2021.

Eduan Dufourq and Bruce A. Bassett. Eden: Evolutionary deep networks for efficient machine learning. In *PRASA-RobMech*, pp. 110–115, Bloemfontein, South Africa, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2011*, pp. 44–51, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21735-7.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009. doi: 10.1109/ICCV.2009.5459469.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Simple method for high-performance digit recognition based on sparse coding. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 19:1985–9, 12 2008. doi: 10.1109/TNN.2008.2005830.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Ashok Kumar Pant, Sanjeeb Prasad Panday, and Shashidhar Ram Joshi. Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks. In *2012 Third Asian Himalayas International Conference on Internet*, pp. 1–5. IEEE, 2012. URL `https://www.kaggle.com/datasets/ashokpant/devanagari-character-dataset/data`.

A. Polesel, G. Ramponi, and V. J. Mathews. Image enhancement via adaptive unsharp masking. *IEEE Transactions on Image Processing*, 9(3):505–510, 2000a.

A. Polesel, G. Ramponi, and V.J. Mathews. Image enhancement via adaptive unsharp masking. *IEEE Transactions on Image Processing*, 9(3):505–510, 2000b. doi: 10.1109/83.826787.

Marc' aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann Cun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL `https://proceedings.neurips.cc/paper_files/paper/2006/file/87f4d79e36d68c3031ccf6c55e9bbd39-Paper.pdf`.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/2cad8fa47bbef282badbb8de5374b894-Paper.pdf`.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3856–3866, Long Beach, CA, USA, 2017b.

Rajkumar Sudalairajkumar. Tamil nlp. [Data set]. Kaggle, 2020. URL `https://www.kaggle.com/datasets/sudalairajkumar/tamil-nlp`.

Nagul Ulaganathan, Rohith J, Sri Aravind S, Abhinav A S, Vijayakumar V, and Ramanathan L. Isolated handwritten tamil character recognition using convolutional neural networks. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 383–390, 2020. doi: 10.1109/ICISS49785.2020.9315945.

Ryan Wiyatno and Jeff Orchard. Style memory: Making a classifier network generative. *CoRR*, abs/1803.11189, 2018.

Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535, 2010. doi: 10.1109/CVPR.2010.5539957.

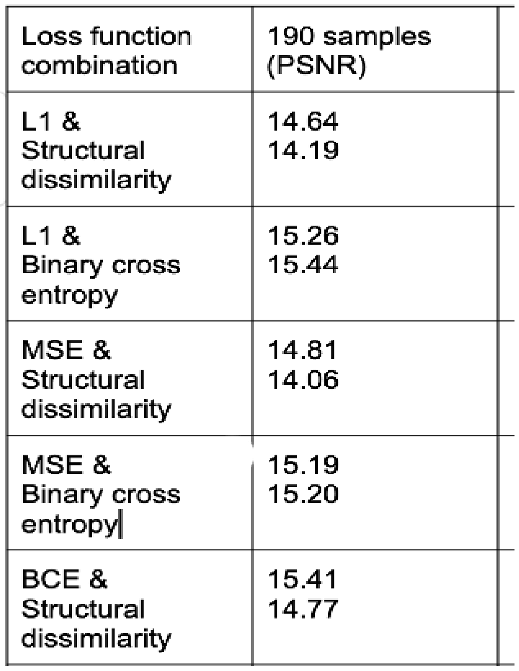
A Appendix

We due page limit we have added few contents here.

A.0.1 Experiment 1: Loss Functions

We used experimented with several loss functions listed below.

$$PSNR = 10\log_{10}\left(\frac{MAX_i^2}{MSE}\right)$$



Loss function combination	190 samples (PSNR)
L1 & Structural dissimilarity	14.64 14.19
L1 & Binary cross entropy	15.26 15.44
MSE & Structural dissimilarity	14.81 14.06
MSE & Binary cross entropy	15.19 15.20
BCE & Structural dissimilarity	15.41 14.77

Figure 2: Comparsion of various loss function that we used for image reconstruction

1. **Mean Squared Error (MSE):**
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x(p) - y(p))^2$$
2. **L1 Norm:**
$$L1 = \frac{1}{N}\sum_{i=1}^{N}|x(p) - y(p)|$$
3. **Structural Similarity Index (SSIM):**
$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$
Structural Dissimilarity (DSSIM) is defined by $DSSIM = \frac{1}{N}\sum_{i=1}^{N}(1 - SSIM(p))$.
4. **Binary Cross-Entropy (BCE):**
$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y(p)\log(x(p)) + (1 - y(p))\log(1 - x(p))]$$

The above figure show the compare of various loss functions that we used for image reconstruction. We have used peak noise to signal ratio (PSNR) as the unit. This is essential for noting the reconstruction accuracy. What we found after the experiments is that Binary Cross Entropy (BCE) and Structural Dissimilarity works well on our images for reconstruction.

A.1 Experiment 2: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs)

Here, we have examined existing methods for determining if Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) can augment original handwritten characters while keeping their human readability. These methods are often considered for the purpose of data augmentation because of their ability to generate new images.

In particular, we explore the use of conditional GANs, which are generating images from random noise. While conditional GANs are effective to create new images, they often fail in applying specific perturbations, leading to less realistic variations. This limitation is clear in the figure below, where the images generated by GANs and VAEs are compared with the original.

The image on the far left is the original, depicting the number 2 from MNIST dataset. The images on the right, generated by GANs and VAEs, show a notable lack of sharpness, and in some cases, their legibility is significantly compromised.

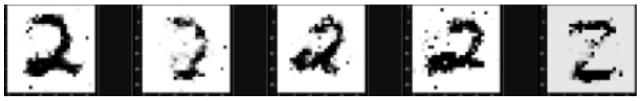


Figure 3: Comparison of original and GAN/VAE-generated images.

A.1.1 Experiment 3: Number of samples per class

we experimented with number of samples per class to find out the least number of samples required to get close to state-of-the-art accuracy. That is how we found the least number of samples required. From the experiments we found 190 samples per class is the lower bound. Below which accuracy of the model significantly reduces.

A.1.2 Experiment 4: Number of layers of convolutional and de-convolutional

We also experimented with number of layers of convolutional and de-convolutional in capsule and decoder network. This experiment helped us to finalize the number of layers required in the current architecture that we proposed.

A.1.3 Sources used for developing this project

Special thanks for these sources and many others that we referred for completing this project. Additionally, thanks to Professor Dani Yogatama and Teaching assistants who helped us when we were stuck while implementing this project.

`https://www.tensorflow.org/tutorials`

`https://www.geeksforgeeks.org/`

`https://www.kaggle.com/`

`https://www.deeplearning.ai/`

`https://www.fast.ai/`

`https://teachablemachine.withgoogle.com/`

`https://machinelearningmastery.com/`

https://papers.nips.cc/paper_files/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html