



**College of Computing and Informatics
University of North Carolina at Charlotte**

A project report on

Handwritten Devanagari Character Recognition

Project Guide

Dr. Minwoo Jake Lee

Team details

Krupa Chand Appikonda (kappikon@uncc.edu)

Maheswara Rao Lekkala (mlekkala@uncc.edu)

Priyanka Jadhav (pjadhav7@uncc.edu)

Rishika Mathur (rmathur5@uncc.edu)

Tagore Pothuneedi (tpothune@uncc.edu)

TABLE OF CONTENT

INTRODUCTION	3
PROBLEM STATEMENT AND CHALLENGES	5
MOTIVATION	6
CONCISE SUMMARY OF OUR APPROACH	6
METHOD	11
EXPERIMENTS	16
TEAM CONTRIBUTION	20
FEEDBACK REFLECTION	20
CONCLUSION	22
REFERENCES	23

INTRODUCTION

Devanagari Script is the most popular script in India which is used in many languages like Sanskrit, Hindi, Marathi, Nepali and Konkani. Most of the Indian languages, including Devanagari originated from the ancient Brahmi script through various transformations. In Devanagari script, there are thirty-six Consonants —Vyanjan and 13 vowels —Swar. When those thirty-six consonants are attached to the vowels, then complex compositions of its constituent symbols are generated. A vowel is followed by a consonant may generate a modified shape, which, depending on the vowels are placed before, after, top and bottom of the consonants are called modifiers or —Matras, because they are used to modify the consonants meaning . There are total twelve modifiers. Devanagari script is a unicas script. There is no concept of upper and lower case letter as Latin script. Words are written in the Devanagari script as they are pronounced so that script is called as phonetic script. Devanagari script is also called as syllabic script because the text is combinations of consonants and vowels that together form syllables. Apart from this, composite & complex characters are formed by combining more than one basic character. Consonants can have a half form when they are joined with other consonants.

Handwriting recognition is the ability of a machine to receive and interpret handwritten input from multiple sources like paper documents, photographs, touch screen devices etc. The handwriting recognition area has been researched extensively till date, whereas recognition of Devanagari script is progressing area of research. Handwriting Recognition is still considered a challenging problem statement. The high variance in handwriting styles across people and poor quality of the handwritten text compared to printed text pose significant hurdles in converting it to machine readable text. Nevertheless it's a crucial problem to solve for multiple industries like healthcare, insurance and banking. Classification of the handwritten Devanagari characters using Convolutional Neural Networks are discussed in this report. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other

classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets can learn these filters/characteristics.

Hindi Alphabet (Devanagari Alphabet)					
अ a	आ ā	इ i	ई ī	उ u	ऊ ū
ओ o	ए e	ऋ r	ऐ ai	औ au	
क ka	ख kha	ग ga	घ gha	ङ ṅa	च ca
छ cha	ज ja	झ jha	ञ ña	ट ṭa	ठ ṭha
ड ḍa	ढ ḍha	ण ṇa	त ta	थ tha	द da
ध dha	न na	फ pha	ब ba	भ bha	म ma
य ya	र ra	ल la	व va	श śa	ष sa
स sa	ह ha	प pa			

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. The traditional CNN classifiers can

learn the important 2D features present in the images and classify them, the classification is performed by using soft-max layer.

PROBLEM STATEMENT AND CHALLENGES

Recognition of handwritten characters is a challenging job for so many reasons :

- **Complex script :**

Devanagari character set has complex shape with various features like strokes, line segments, header line, dots, curves and holes. These features change the entire word and its context.


- **Similar structure :**

In Devanagari script, there are large similarities between the characters.

प	ष	The difference being middle line present
थ	य	The difference being loop present
ड	ड़	The difference being single dot present on side
ढ	द	The difference being loop and down stroke present
इ	ई	The difference being the curve present on the top

- **Writing style variations :**

There are varying styles of writing. Handwriting style of an individual person also varies time to time and is inconsistent. Also, cursive handwriting makes separation and recognition of characters challenging. In the below figure we can see that different characters are written similarly.

		प		य
		ध		घ

- **Gradient Problem:**

In neural networks, the most used Activation functions like Sigmoid, ReLU has gradient learning problems when the inputs are too small or when an activation has "-ve" effect are not considered. The gradient learning problem occurs when the gradients computed during backpropagation are smaller than 1. In our approach we want tackle this problem and provide some solutions for better improvement of learning.

MOTIVATION

- Many ancient Indian texts are written in Devanagari. Because of their volume, most people could not read it. But if we convert them to digital documents (Digitization), they will be available to many people.
- It can be used by online education portal to help teachers understand the handwritten answers by students.
- For students who are learning to write Devanagari, it can be a good tool.
- Majority of Optical Character Recognition (OCR) software are not very accurate for Devanagari because they fail to understand the context of the text.

CONCISE SUMMARY OF OUR APPROACH

- To overcome the challenge of gradient learning process we focused on activation functions which can range differently for various inputs.

- Our approach is to customize the activation functions which can range well when the gradient is calculated. For example, if a convolution which has large value after max pooling, and it doesn't contribute to the right prediction, the activation should not be only zero. But the neuron should also have "-ve" impact on it.
- Our customized activation functions are :
 - Param_tanh
 - ReLu_moid
 - Param_sigmoid

- Data Preparation:

Reading the images from the Devanagari dataset into DataFrame. Then converting the image into a grayscale and finally in binary image.

- Classification Models :

Building and training the Random Forest, SVM, K-nearest neighbors, Naïve Bayes and CNN models. We will calculate accuracy, F1 score, precision and recall for all the models.

- Image Thinning :

It helps to reduce distortions among character shapes while preserving the fundamental skeleton which helps in feature extraction.

- CNN Architecture :

Building the Sequential model. The input layer(CONV2D) consists of the raw pixel values from the 32x32 pixel binary image and has no trainable parameters and has Kernel size (3*3). The input neurons are 32. We have 2 hidden Layers (CONV2D and Dense) in our model each of 64 and 128 neurons respectively. These layers will use the custom activation functions. Adding Regularizing parameter to the model which helps to generalize more on

the input. There will be one output layer (Dense) which contains 46 units, and use softmax as an activation function for classification.

- Study of Activation Functions:

For performance evaluations of our custom activation functions used in CNN, we are using ReLU, tanh, sigmoid as benchmarks to compare.

- Comparing customized activation functions with ReLU, tanh and sigmoid functions.

BACKGROUND / RELATED WORK

Literature Survey

- **SVM for Handwritten Devanagari Numeral Recognition by Shailedra Kumar Shrivastava, Sanjay S.Gharde**

This paper applies the SVM classification algorithm for recognizing the handwritten numerals of Devanagari script. They had built the data using ANESP and around 2000 samples of data from different people varying in ages, having different writing styles. This approach of numeral recognition can be extended to recognize the characters of Devanagari.

Pros :

1. SVM model is computationally efficient.
2. Implemented using multi SVM which is best technique nonlinear classification or regression.
3. Feature Extraction like moment invariants and affine moment invariants are used.

Cons:

1. Use Kernel function like Rbf, polynomial, sigmoidal functions to further increase accuracy.
2. The dataset is very small and only few people are selected as a sample to build the dataset.

▪ **Handwritten Devanagari Character Recognition using CNN by Aarati Mohite, Sushama Shelke**

This paper has implemented the recognition of character using transfer learning for convolution neural network architecture where existing architecture Alexnet is used. The architecture consists of 8 Layers extracting features followed by ReLU and max-pooling layer. Multi Class SVM is used for fitting extracted features and labels. Using this approach has a very good advantage of reducing efforts on feature extract and building entire network scratch.

Pros:

1. Saves time by using Pre-trained architecture to develop CNN rather building from scratch.
2. As the author opted for CNN the feature extraction is done automatically by neural networks itself.

Cons:

1. Recognition accuracy can be further improved by using more preprocessing technique like image cropping, normalization etc.

▪ **Optical Character Recognition on Android OS for Business Cards by Sonia Bhaskar, Nicholas Lavassor, Scott Green**

The project involves working with the optical character recognition engine Tesseract. MATLAB implementation of the algorithm is done to optimize the image for input to the Tesseract OCR engine. Preprocessing steps like Gray scaling, Adaptive thresholding, perspective transformation are executed before the image is fitted using baseline fitting algorithm. A fixed pitch detection approximates the character width which Tesseract uses for correct incremental extraction of character. Techniques like Non-fixed pitch spacing delimiting is used for characters that are uniform width. Once all the possible blobs of characters are found tesseract does word by word recognition.

Pros:

1. Simplified approach to run a character recognition model on a mobile device.
2. Less complex.

Cons:

1. Real world photos may introduce various environmental factors.
2. Uneven illumination ,off-angle rotation and perspective can hinder the performance of the model.

▪ **Offline Handwritten Devanagari Vowels Recognition using KNN Classifier by Rakesh Rathi, Ravi Krishan Pandey, Mahesh Jangid, Vikas Chaturvedi**

This paper has used a K-Nearest Neighbors classifier for recognition of handwritten Devanagari characters. Before the data is feed to the classifier and after preprocessing steps the feature extraction is done through recursive subdivision technique. The dataset

contains 9191 samples which is divided into 1:3 for testing and training purposes. Paper was only limited to Devanagari Vowels. This approach can be use and extended for all the characters of Handwritten Devanagari characters.

Pros:

1. Level of accuracy can be tested at different threshold of granularity levels and can be stopped when maximum accuracy is achieved.

Cons:

1. The accuracy of KNN Classifier can be severely degraded when there is noise in the data or if there are any irrelevant features.
2. The paper was only limited to small dataset.

METHOD

Data Gathering:

The Devanagari Character dataset is available on the UCI Machine Learning Repository. There are 46 classes with 2000 examples each. All the images are in grayscale with resolution 32 by 32 with .png format.

Data preprocessing:

Input images are structured into 46 classes/groups (folders)– 36 consonants and 10 digits.

Reading Image Data into a DataFrame. Creating a DataFrame (df) which will act as input Dataset for ML models. preprocessing for each character image by converting input image into a grayscale image and to binary image. We normalized pixel values which will reduce the scaling range. Flatten the image and add the preprocessed image into the list.

A temporary DataFrame created with a target column as a character name from the preprocessed image list. Merge all temporary DataFrames into the main DataFrames which are used for

training and testing the ML model. Prepare predictors matrix (X) and target vector(y). Partition the dataset into train and test.

Below is a sample image of letter “ka” before and after preprocessing and feeding it into the network.

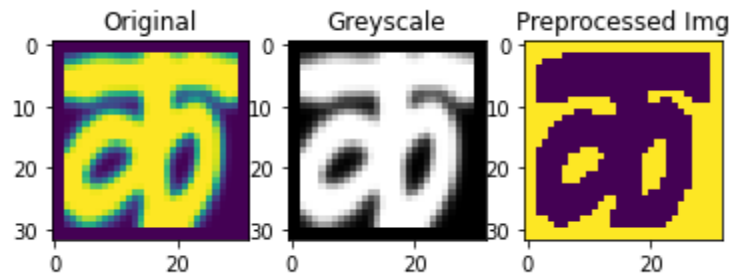


Fig. Image of a character Ka

Building a Convolutional Neural Network

We used TensorFlow Keras to build a CNN model as shown in the figure. A sequential model was used to build models. The input layer (CONV2D) consists of the raw pixel values from the 32x32 pixel binary image and has no trainable parameters and has Kernel size (3*3). The input neurons are 32. We have 2 hidden Layers (CONV2D and Dense) in our model, each of 64 and 128 neurons, respectively. All hidden layers will use the custom activation functions. Followed by a pooling layer which helps the classifier more immune shift and distortion. (2x2) region from previous convolution layer is pooled into aggregates and produces a feature map as output and input to next convolutional layer. We added a Regularizing parameter to the model which helps to generalize more on the input. The fully connected output layer (Dense) which contains 46 units, and we use SoftMax as an activation function for classification.

For performance evaluations of our custom activation functions used in CNN, we are using ReLU, tanh, sigmoid and Swish as benchmarks to compare.

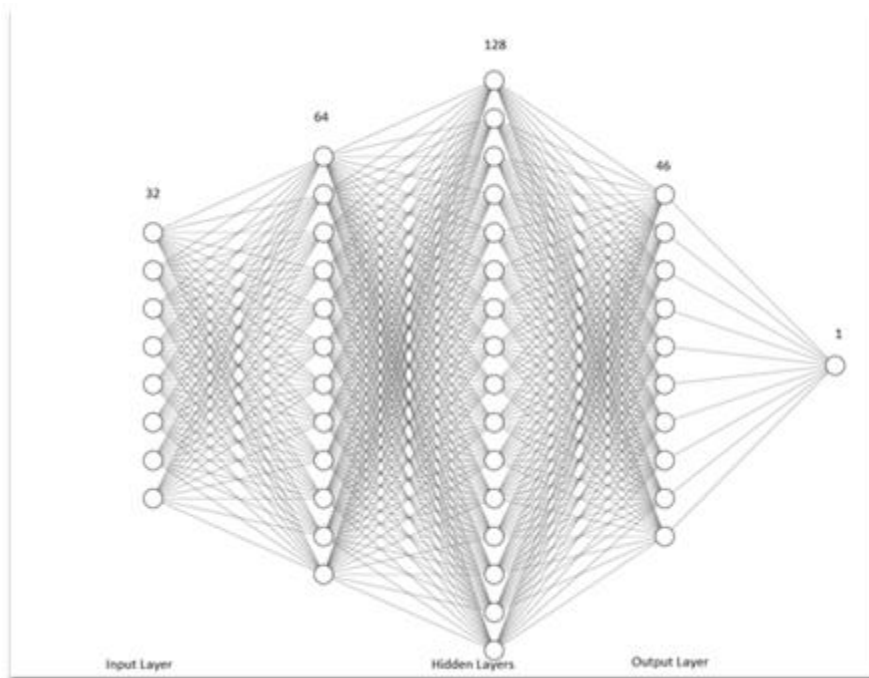


Fig. CNN Model for Classification

Customized Activations Functions (Novel)

To overcome the challenges faced by conventional activation functions Our project has mainly focus on gradient learning. And to improve learning, the activations play a major role in making the networks more efficient and reactive to changes. But as already mentioned the activations like Sigmoid, Tanh, ReLU has issues in learning process say it be slow learning, saturated neurons or ability to learn. For this we have designed customized activation functions which can scale better and improve the gradient learning problems. Our approach is by taking the activations like sigmoid, tanh and ReLU and tried scaling them by doing some arithmetic operations or see if we can somehow extract the best features of each function and combine them into one.

So, after doing some plotting, we came up with the following activation functions.

Param_tanh:

As the hyperbolic tangent has range -1 to 1 . It scales well into putting values in different signs. To reduce the impact of gradient learning, we have used the properties of Hyperbolic tangent which have a good scaling range. we considered using \tanh and customize the activation by doing some arithmetic operations like added with a parameter which can scale further to better learn the gradients. We want to test the \tanh activation by adding the input parameter to tangent function as

$$F(x) = x + \tanh(x)$$

This Function uses \tanh and scales it linearly and ranges is expanded into negative as well, which should reduce the gradient learning problem further.

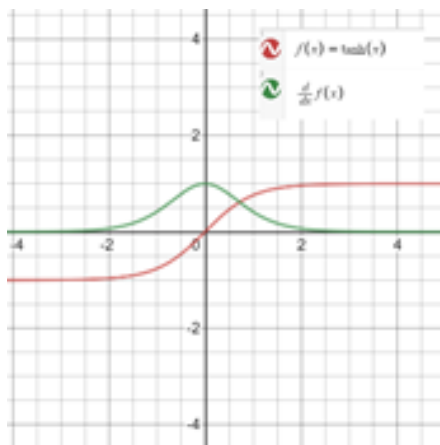


Fig. tanh

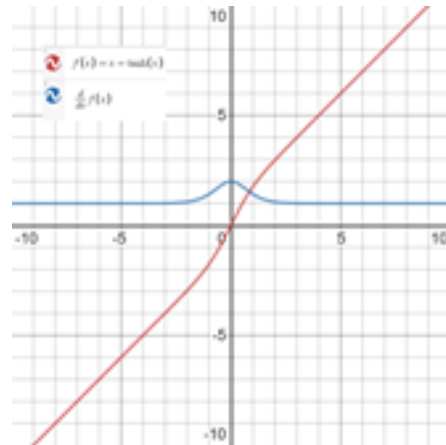


Fig. Param_tanh

ReLU_moid:

In ReLU activation function, the value is simply threshold to zero if input is less than zero, which could make network sparser and is faster in convergence but sometimes during training

the neurons can die due to large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate again on any input data. The problems of gradient learning effects on ReLU can be reduced if we take advantage of ReLU and sigmoid into one function. So, we built a function as below.

$$F(x) = \begin{cases} x & x > 0 \\ \frac{1}{1+e^{-x}} & \text{otherwise} \end{cases}$$

This way we can benefit from ReLU's and sigmoid advantages like with respect to the negative segment of the function, while the positive segment is smooth in terms of gradients.

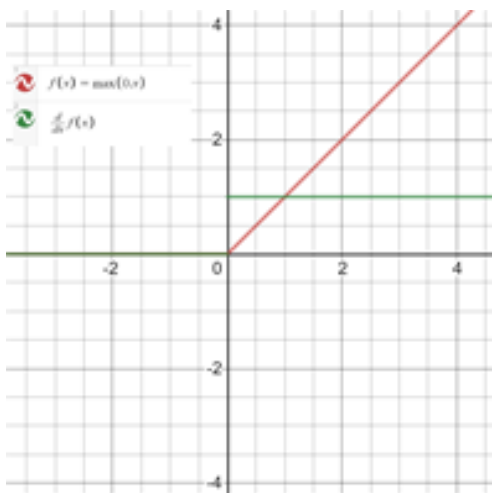


Fig ReLU

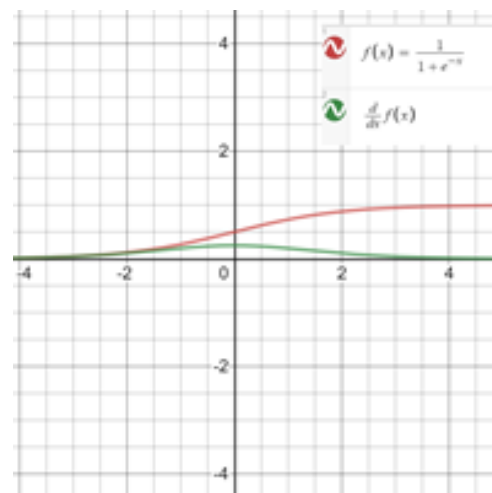


Fig Sigmoid

Param_sigmoid: The ReLU_moid has really capture the best features of both most used activation functions ReLU and sigmoid. Sigmoid is applied mostly if we must predict the probability as an output. The function is differentiable, and we can find the slope at any two given points. But we are focused to see If the sigmoid's gradient can be scaled further if we do a multiplication on the sigmoid with the input parameter.

$$F(x) = \frac{x}{1 + e^{-x}}$$

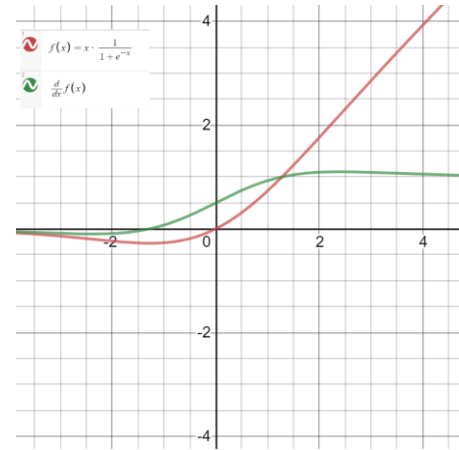


Fig.Param_sigmoid

EXPERIMENTS

Implementation code Git Hub Repository Link : <https://github.com/pjadhav3/ml-labs>

File name: Handwritten Character Recognition.ipynb

We tested the dataset with same model CNN architecture but by varying on different customized activation functions and for comparison and analysis we also test on standard activations like hyperbolic tangent, sigmoid and ReLU. We are training our model with batch size of 128 and epochs of 12. The training and test data are split into 80-20 for training and validation testing, respectively. The main advantage of the Neural network are they easily do feature extraction. We build our model using keras in build modelling libraries. For building our customized activations we used keras generic_utils and custom_objects for defining the activation in our implementations. The advantage of using keras to define or implement our own activation is that the auto differentiation feature which will automatically differentiate and backpropagate for learning the weights. Once we define the activations we update the library with the function names. For the model optimizers we used Adam , metric of accuracy and loss as categorical_crossentropy. The results of all the customized activations comparing to standard are presented as follows.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 32)	320
conv2d_3 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 128)	1605760
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 37)	4773

Total params: 1,629,349

Trainable params: 1,629,349

Non-trainable params: 0

Training and test Accuracy of customized activations

Test loss: 3.5852901935577393
Test accuracy: 0.02777777798473835

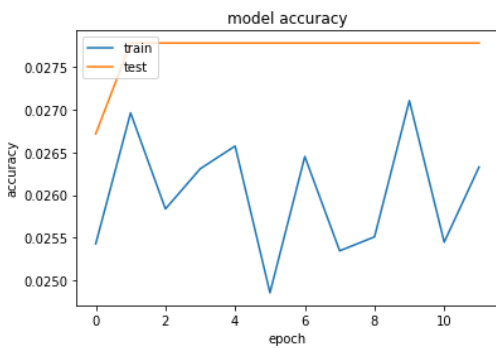


Fig. ReLU_moid

Test loss: 0.15537482500076294
Test accuracy: 0.9544934630393982

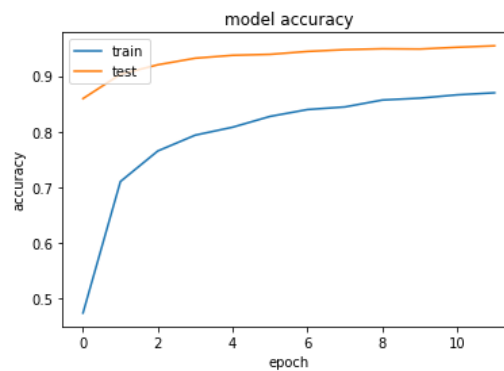


Fig. ReLU

Test loss: 0.3084907829761505
Test accuracy: 0.9146241545677185

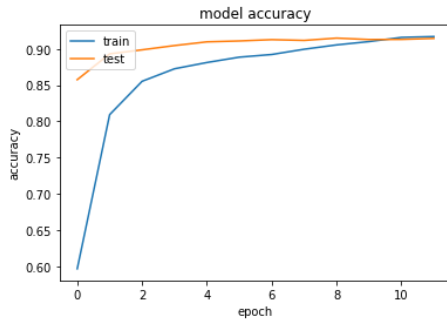


Fig. Param_tanh

Test loss: 0.22805872559547424
Test accuracy: 0.9301470518112183

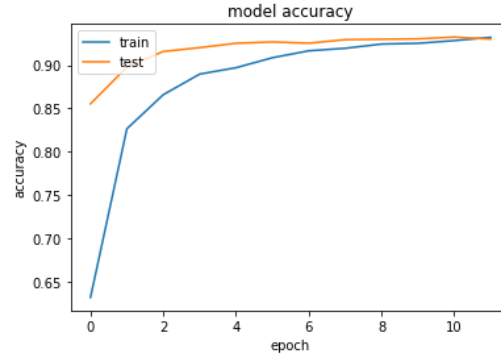


Fig. Tanh

Test loss: 0.13292258977890015
Test accuracy: 0.9607843160629272

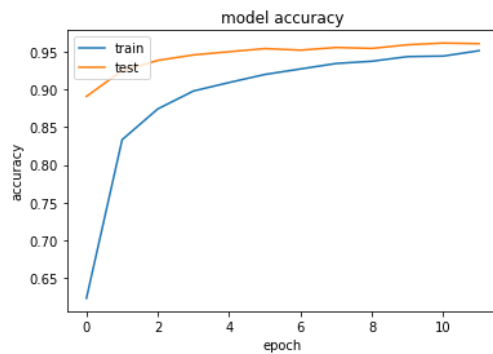


Fig. Param_sigmoid

Test loss: 3.585348606109619
Test accuracy: 0.02777777798473835

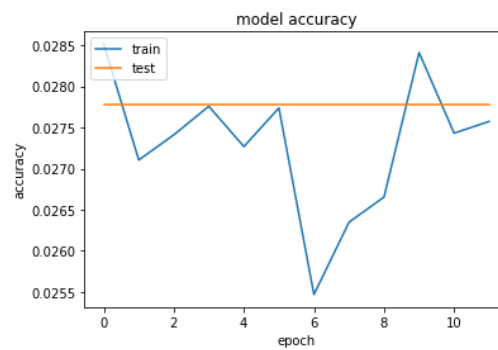
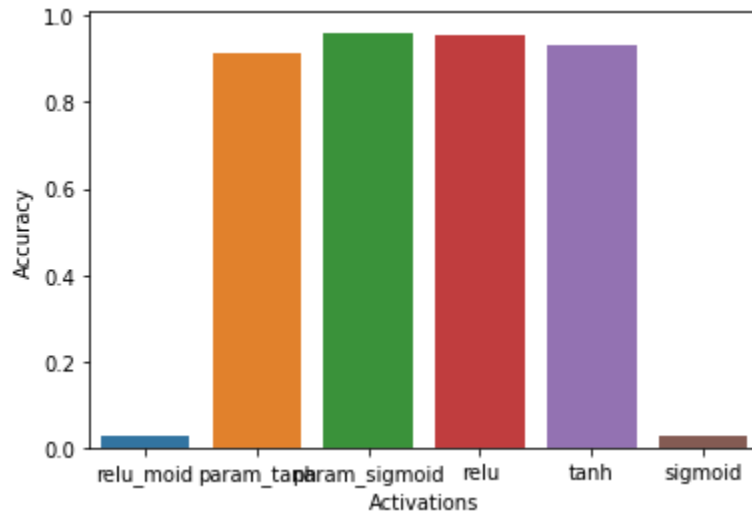


Fig. sigmoid

Performance Evaluation

Activation	Accuracy
Param_sigmoid	96.07%
Param_tanh	91.46%
ReLU_moid	2.77%

Activation	Accuracy
ReLU	95.44%
tanh	93.01%
sigmoid	2.77%



Analysis:

Based on the results the customized activations param_sigmoid, param_tanh got very good accuracy rate with 96.07% and 91.46%. Relu_moid activation had failed on classification though the assumption of combining best of ReLU and sigmoid has failed. To our surprise the param_sigmoid has actually had highest accuracy of all the activation and even relu is lower by 0.6%. The standard activation also had failed to classify similar to relu_moid and both have interestingly same accuracy of 2.7% ,the least of all the activations. From training and test plots of param_sigmoid and param_tanh the training and test are good starting with training accuracy of 0 and test with nearly 90% and almost converges in 10th -11th epoch. But the relu_moid has performed better on test data and low in training still have very low accuracy in both cases. Similarly, the standard sigmoid also had same problem and performed worse than relu_moid on test data.

TEAM CONTRIBUTION

Tasks	Priyanka	Rishika	Tagore	Mahesh	Krupa Chand
Data Preparation	✓			✓	
Applying conventional classification model to get an idea about scope of improvement	✓		✓		
Research on the best model to use	✓	✓	✓	✓	✓
Build a CNN model		✓	✓		✓
Mid Project Report	✓	✓	✓	✓	
Study of activation functions	✓	✓	✓	✓	✓
Applying Sigmoid activation function on CNN				✓	✓
Applying Tanh activation function on CNN			✓		
Applying ReLU activation function on CNN		✓			
Designing and implementing custom activation function – 1 (param_tanh)	✓			✓	
Designing and implementing custom activation function – 2 (param_sigmoid)		✓	✓		
Designing and implementing custom activation function – 3 (ReLU_moid)					✓
Fine Tune Model for better results		✓	✓		
Performance evaluation of built CNN model	✓		✓	✓	

FEEDBACK REFLECTION

Comment 1: What challenge is your project's main goal?

The main goal of our project is to predict the characters of Handwritten Devanagari using different approaches like Convolutional Neural Network For neural network and SVM ML models, we are using different custom activation functions to overcome Gradient Learning Problem.

Comment 2: Why is this con?

(Referring “Use Kernel function like Rbf, polynomial, sigmoidal functions to further increase accuracy.” in paper SVM for Handwritten Devanagari Numeral Recognition) . We mean to say,

the authors of ‘SVM for Handwritten Devanagari Numeral Recognition’ could have increased accuracy by using Kernel functions like Rbf, polynomial, sigmoidal functions.

Comment 3: How do you relate the surveyed approaches to your approach?

Our approach to achieve the goal is as follows:

Preprocess the images – convert grayscale images to black and white images and normalize it.

Extract features of character.

Build CNN and SVM classification model.

Train the classifier model with extracted features.

To extract features of characters and train the model, we plan to use surveyed approaches such as CNN and SVM.

Comment 4: Typical pipeline. What is new? What should you highlight?

We are using conventional pipelines and we are focusing more on Feature Extraction and Character Classification stages. We are trying to deal with the Gradient Learning Problem using customized activation functions which is part of Feature extraction and character classification stages.

Comment 5: I was not able to see any “interesting” difference in your project. Please focus your effort on the difference.

We are trying to make a difference by customizing the activation functions used in CNN to overcome the Gradient Learning Problem instead of using the most used activation functions such as ReLU, Sigmoid and tanh.

Comment 6: For me you don’t have any clear direction thus lack of detail in your plan.

We have mentioned details about our plan in Approach section of the report.

Comment 7: Everyone does everything also shows a lack of preparation.

Weekly sessions were held to discuss and exchange thoughts to come up with a better plan collectively. But we agree with your point and agreed to delegate the duties to each other based on our availability and interests. The updated work assignment is given on the table above.

Comment 8: Table should highlight the highest performance to compare well.

We highlighted the highest performance (accuracy, precision and recall) in the performance evaluation table in the report.

Comment 9: Performance evaluation plots will be better with bar graphs rather than line plot as the connection does not have any meaning.

We used bar graphs to show the performance evaluation of our customized activation functions in the report.

CONCLUSION

We proposed a neural network model for the Devanagari Character Recognition. We trained a CNN classification model using customized activation functions viz. param_sigmoid, relu_moid and param_tanh on the DHCD dataset to achieve the accuracy of 96.07%.

As a part of data preprocessing, we converted the character images into grayscale images and then into binary images so that the edges of letters become finer and it will help in feature extraction. Then we implemented a simple convolution neural network having three convolution layers in the feature extraction section followed by a SoftMax layer for final classification. The model was able to recognize and classify the characters.

We tried to improve the model by using different combinations of parameters in convolutions. To attain more accuracy and avoid overfitting, we added a dropout layer. Then we thought of using customized activation functions instead, to see if this approach will help us improve accuracy even more.

For the performance evaluation of our custom activation functions, we used ReLU, tanh, and sigmoid as benchmarks. Our experiment with the activation functions param_sigmoid performed better than ReLU, Tanh and Sigmoid and achieved accuracy of 96.07%

This project helped us understand the importance of pre-processing for better results. We learned about Convolutional Neural Networks. Also, it helped us understand the working of different activation functions like ReLU, sigmoid and tanh and we can create our own activation functions according to specific needs.

REFERENCES

- [1] Mohite, A., & Shelke, S. (2018). Handwritten Devanagari Character Recognition using Convolutional Neural Network. Retrieved 2020, from <https://ieeexplore.ieee.org/document/9057991/authors#authors>
- [2] Bhaskar, S., Lavassar, N., & Green, S. (2010). Implementing Optical Character Recognition on the Android Operating System for Business Cards. Retrieved 2020, from <https://www.semanticscholar.org/paper/Implementing-Optical-Character-Recognition-on-the-Bhaskar-Lavassar/1bf3c55d17214452970497a0915ee945464d5742>
- [3] S. Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Kathmandu, 2015, pp. 1- 6. doi: 10.1109/SKIMA.2015.7400041

- [4] Joshi, R., Goel, P., & Joshi, R. (2020, January 19). Deep Learning for Hindi Text Classification: A Comparison. Retrieved 2020, from <https://arxiv.org/abs/2001.10340>
- [5] R. M. K. Sinha, "A journey from Indian scripts processing to Indian language processing," IEEE Ann. Hist. Comput., vol. 31, no. 1, pp. 8–31, Jan./Mar. 2009.
- [6] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," Neural Networks, vol. 61, pp. 85-117, Jan. 2015. [Online]. Available <https://arxiv.org/abs/1404.7828>. Deep Learning Based Real Time Devanagari Character Recognition 62
- [7] A. Ray, A. Chandawala and S. Chaudhury, "Character Recognition Using Conditional Random Field Based Recognition Engine," 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, 2013, pp. 18-22. doi: 10.1109/ICDAR.2013.13
- [8] V. Ganapathy and C. C. H. Lean, "Optical Character Recognition Program for Images of Printed Text using a Neural Network," 2006 IEEE International Conference on Industrial Technology, Mumbai, 2006, pp. 1171-1176. doi: 10.1109/ICIT.2006.372591
- [9] V. Bansal and R. M. K. Sinha, "Segmentation of touching and fused Devanagari characters," Pattern Recognit., vol. 35, pp. 875–893, 2002.
- [10] H. Zhao, Y. Hu and J. Zhang, "Character Recognition via a Compact Convolutional Neural Network," 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Sydney, NSW, 2017, pp. 1-6. doi: 10.1109/DICTA.2017.8227414