

PROJECT-I REPORT
ON
**Comparative Analysis of Deep Learning Algorithms for
Devanagari Character Recognition**

*SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR
THE AWARD OF THE DEGREE OF*

BACHELOR OF ENGINEERING
(INFORMATION TECHNOLOGY)



Supervisor:

Dr. Veenu Mangat
Associate Professor
IT-Dept

Submitted By:

Abhishek Anand (UE158003)
Anish Narang (UE158017)
Ankit Shankhala (UE158018)
Kunaldeep Jindal (UE158058)

To

Department of Information Technology
University Institute of Engineering and Technology,
Panjab University, Chandigarh
2018-19

DECLARATION

We hereby declare that the project entitled “**Comparative Analysis of Deep Learning Algorithms for Devanagari Character Recognition**” submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering-Information Technology, is our original work and the project has not formed the basis for the award of any other degree/diploma/fellowship or any other similar titles. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Abhishek Anand UE158003

Anish Narang UE158017

Ankit Shankhala UE158018

Kunaldeep Jindal UE158058

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of my project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We are highly indebted to Dr. Veenu Mangat for her guidance and constant supervision as well as for providing necessary information regarding the project; also, for providing extreme support and vision in completing the project.

We would like to express our special gratitude and thanks to Mr. Naman Aggarwal, for giving us such attention, guidance and time throughout the completion of our project.

We would also like to thank other faculty members of Department of Information Technology University Institute of Engineering and Technology, Panjab University, Chandigarh who have taught us various subjects which have proved very helpful and were not less than a backbone behind this project.

Our gratitude and appreciation also go to our classmates for their help and encouragement and to all others who willingly helped us with their abilities.

Thankyou.

INDEX

| S.No. | Content | Page No. |
|-------|---|----------|
| 1 | Introduction | 5 |
| | Aim | 7 |
| | Objective | 7 |
| | Challenges on Character Recognition | 8 |
| | Previous Work done | 10 |
| | Techniques used for Character Recognition | 12 |
| 2 | Minimum Requirements | 14 |
| | Hardware Requirements | 14 |
| | Software Requirements | 14 |
| 3 | Dataset | 15 |
| 4 | Implementation | 17 |
| | RNN | 17 |
| | LSTM | 19 |
| | GRU | 21 |
| | CNN | 22 |
| 5 | Project Snapshots | 24 |
| 6 | Analysis and Results | 28 |
| 7 | Conclusion | 29 |
| 8 | Future Scope | 30 |
| 9 | Bibliography | 31 |

INTRODUCTION

With the development of new technologies, demand of digital content increases day by day. There are a lot of handwritten contents are stored in libraries, but they are not accessible from outside the libraries. So, there is need of content digitalisation for converting handwritten contents into editable and digital form to avail the content online. To generate the digital content, need for the development of an online or offline Handwriting Recognition System with high performance has become essential. Handwriting is a personal skill of every individual. Each person has their own unique style of handwriting; it may be everyday handwriting or their personal signature. Now time is to use the digital signature. The characteristics of handwriting are: slop, specific shape, thickness, average size of letters, pressure to the paper and regular or irregular spacing between letters. These characteristic are used to recognize the handwritten characters. The Handwriting Recognition System (HRS) is used to take handwritten document (texts) as input and produce editable document (digital text) as an output Optical Character Recognition systems are least explored for Devanagari characters. The report presents few deep learning algorithms for recognition of Devanagari characters. In this report we'll introduce a new publicly available dataset, Devnagari Character Dataset, of 92 thousand images of 46 Devanagari characters. Then, we also propose Deep learning architecture to classify the characters in DCD. Introduction of multilayer perceptron network has been a milestone in many classification tasks in computer vision. But, performance of such a network has always been greatly dependent on the selection of good representing features.

Deep Neural Networks on the other hand do not require any feature to be explicitly defined, instead they work on the raw pixel data generating the best features and using it to classify the inputs into different classes. Deep Neural networks consist of multiple nonlinear hidden layers and so the number of connections and trainable parameters are very large. Besides being very hard to train, such networks also require a very large set of examples to prevent overfitting. One class of DNN with comparatively smaller set of parameters and easier to train are Convolutional Neural Network and Recurrent Neural Network. The ability of these Deep Learning algorithm to correctly model the input dataset can be varied by changing the number of hidden layers and the trainable parameters in each layer and they also make correct assumption on the nature of images. Like a standard feed forward network, they can model complex non-linear relationship between input

and output. But these algorithms have very few trainable parameters than a fully connected feed-forward network of same depth. Algorithms introduce the concept of local receptive field, weight replication and temporal subsampling which provide some degree of shift and distortion in- variance. algorithms for image processing generally are formed of many convolution and sub-sampling layers between input and output layer. These layers are followed by fully connected layers thereby generating distinctive representation of the input data.

Beside image recognition, these algorithms have also been used for speech recognition. Although deep neural networks have a small and inexpensive architecture compared to standard feed forward network of same depth, training a deep neural network still requires a lot of computation and a large labeled dataset. Training such a network was not so effective and did not produce any superior result to traditional shallow network, until recently. With the availability of large labeled dataset like IMAGENET, MNIST, SVHN, development of state of the art GPUs and introduction of unsupervised pre-training phase, DNNs have at present proven to surpass traditional feed forward network in a number of classification tasks. In DNNs, initializing the weight randomly and applying gradient descent and backpropagation to update the weights seems to generate poorer solution for a deep network. So, generally, greedy layer wise unsupervised pre training is applied prior to supervised training. The current deep-learning method still has room for further development. So in this project we took on the task to compare the deep neural networks which are new and tried to implement on Hindi Devanagari Dataset which is available on UCI Machine Learning Repository and on Kaggle. We chose this Dataset because Hindi has a very different pattern of handwritten character because of the common upper horizontal line on all the characters called “shirorekha”. Also as the dataset is also new, clean and balanced, we tried to experiment with the data in order to discover the unexplored sections in this field.

Aim

- This project seeks to compare different deep learning algorithms for classifying an isolated handwritten Hindi language character.
- Our aim is to perform comparative analysis w.r.t accuracy and execution time parameters of several state-of-the-art algorithms.


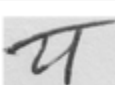


Objective

- The purpose of this project is to compare several state-of-the-art algorithms for Character Recognition on the basis of various parameters for prediction of Hindi language character.
- The objective of the project is to identify the best algorithm which will be able to perform the prediction for Hindi language character.
- It is also desirable that the algorithm must take time factor in consideration: Both training and validation time.
- It is desirable that the loss factors of the best algorithm should be minimum.
- The project should be able to clearly compare the algorithms on the basis of accuracy graphs, loss graphs and time graphs.
- The objective is to take every common parameters of the different algorithm models indistinguishable in order to compare the algorithms in a better way.
- The objective is to create models of several state-of-the-art algorithms and implement them on UCI Devanagari data in order to compare those algorithms for prediction purpose.

Challenges in Devanagari Character Recognition

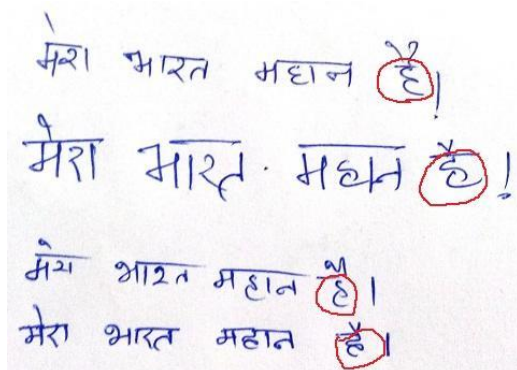
- **Similar structure characters:**

There are many pairs in Devnagari script, that has similar structure differentiating each with structure like dots, horizontal line etc. Some of the examples are illustrated in Fig. below. The problem becomes more intense due to unconstrained cursive nature of writings of individuals. Two such examples are shown in figure below.

| | | | |
|---|---|---|---|
|  | प |  | य |
|  | घ |  | ध |

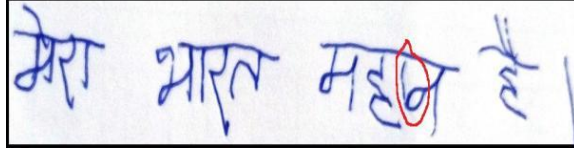
- **Handwriting Style Variations:**

Different person have various handwriting styles. Even same writer have various handwriting style over the time. Many times a writer finds herself /himself unable to recognize her/his own handwriting. Hence, practically it is much difficult to recognize handwriting by machine efficiently.



- **Deformed distortion and overlapping characters:**

The characters should not be touching each other. Though segmentation of touching characters has been one of the toughest jobs in text recognition and this alone has remained one of the toughest area of research.



- **Inputs image should not contain ardhakshers:**

Ardhakshers were excluded from the recognition domain because of the following two main reasons:

1. There is a huge variation in writing styles of writing ardhakshers. Relative positions of ardhakshers, such as 'l' and 'v' with respect to the following character vary drastically. So the identification of these ardhakshers become quite difficult job.
2. In addition to uncertainty in the relative position of ardhakshers, most of these are written by joining them to the alphabet following them and it becomes difficult to segment them out. Since the problem of segmentation of characters is not bend addressed in this project, this assumption was taken.

- **Text lines should be almost straight:**

The sentences input to the OCR should not have large slopes. They should be almost straight. Though the system is robust enough to handle lines with slopes of 10-20 degrees.

Previous Work Done

The origins of character recognition can actually be found back in 1870. This was the year that C.R.Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells. The first true OCR reading machine was installed at Reader's Digest in 1954. This equipment was used to convert typewritten sales reports into punched cards for input to the computer.

In recent years attempts have been made to develop text recognition systems in almost all the languages and scripts of the world. Devanagari script, the attempts were made as early as 1977 in a research report on handwritten Devnagari characters with a limited success. Devanagari script is written by joining the characters, even merging characters to have compound characters and also putting "matras" in various forms, this makes the interpretation or recognition extremely difficult. Arora S., Bhattacharjee D., Nasipuri M., Basu D. K., Kundu M. and L. Malik, proposed a Devnagari Character Recognition system which uses different feature extraction and recognition algorithms. Their proposed system was tested on approximately 1500 handwritten Devnagari character database collected from different people. It was observed that the proposed system achieved 98.16% recognition rates. Singh Raghuraj, Yadav C. S., Verma Prabhat and Vibhash Yadav have presented a scheme to develop complete OCR system for different five fonts and sizes of printed Devanagari characters and the accuracy is found to be quite high. Dongre Vikas J and Vijay H Mankar have proposed a simple histogram based approach to segment Devanagari document with accuracy of 100% in case of line segmentation and 91% in case of word segmentation.

Neha Sahu, Nitin Kali Raman , describes the development and implementation of one such system consisting combination of various stages. Artificial Neural Network technique is used to designed to preprocess, segment and recognize Devanagari characters. The system was designed, implemented, trained and found to exhibit an accuracy of 75.6% on noisy characters. Sonika Dogra, Chandra Prakash, describes a recognition system which can be used for the recognition of offline handwritten Hindi characters. For this proposed system Support Vector Machine is used as classifier and Diagonal feature extraction approach is used to extract features. From the results it is clear that combination of SVM classifier and

diagonal feature extraction approach is better method for the recognition of handwritten characters.

The current deep-learning method still has room for further development. So in this project we took on the task to compare the deep neural networks which are new and tried to implement on Hindi Devanagari Dataset which is available on UCI Machine Learning Repository and on Kaggle. We chose this Dataset because Hindi has a very different pattern of handwritten character due to the common upper horizontal line on all the characters called “shirorekha”. Also as the dataset is also new, clean and balanced, we tried to experiment with the data in order to discover the unexplored sections in this field.

Techniques Used For Character Recognition

1. Template Matching

This is the simplest way of character recognition, based on matching the stored prototypes against the character or word to be recognized. The matching operation determines the degree of similarity between two vectors (group of pixels, shapes, curvature etc.) A gray-level or binary input character is compared to a standard set of stored prototypes. According to a similarity measure (e.g.: Euclidean, Mahalanobis, Jaccard or Yule similarity measures etc). A template matcher can combine multiple information sources, including match strength and k-nearest neighbor measurements from different metrics. The recognition rate of this method is very sensitive to noise and image deformation. For improved classification Deformable Templates and Elastic Matching are used.

2. Statistical Techniques

Statistical decision theory is concerned with statistical decision functions and a set of optimality criteria, which maximizes the probability of the observed pattern given the model of a certain class. Statistical techniques are based on following assumptions:

- a.** Distribution of the feature set is Gaussian or in the worst case uniform,
- b.** There are sufficient statistics available for each class,
- c.** Given collection of images is able to extract a set of features which represents each distinct class of patterns.

The measurements taken from n-features of each word unit can be thought to represent an n-dimensional vector space and the vector, whose coordinates correspond to the measurements taken, represents the original word unit. The major statistical methods, applied in the OCR field are Nearest Neighbor (NN), Likelihood or Bayes classifier, Clustering Analysis, Hidden Markov Modeling (HMM) , Fuzzy Set Reasoning , Quadratic classifier.

3. Support Vector Machine Classifier

It is primarily a two-class classifier. Width of the margin between the classes is the optimization criterion, i.e., the empty area around the decision boundary defined by the distance to the nearest training patterns. These patterns, called support vectors, finally define the classification function. Their number is minimized by maximizing the margin. The support vectors replace the prototypes with the main difference between SVM and traditional template matching techniques is that they characterize the classes by a decision boundary. Moreover, this decision boundary is not just defined by the minimum distance function, but by a more general possibly nonlinear, combination of these distances.

4. Neural Network

Character classification problem is related to heuristic logic as human beings can recognize characters and documents by their learning and experience. Hence neural networks which are more or less heuristic in nature are extremely suitable for this kind of problem. Various types of neural networks are used for OCR classification. A neural network is a computing architecture that consists of massively parallel interconnection of adaptive 'neural' processors. Because of its parallel nature, it can perform computations at a higher rate compared to the classical techniques. Because of its adaptive nature, it can adapt to changes in the data and learn the characteristics of input signal. Output from one node is fed to another one in the network and the final decision depends on the complex interaction of all nodes.

Several approaches exist for training of neural networks viz. error correction, Boltzman , Hebbian and competitive learning. They cover binary and continuous valued input, as well as supervised and unsupervised learning. Neural network architectures can be classified as, feed-forward and feedback (recurrent) networks. The most common neural networks used in the OCR systems are the multilayer perceptron (MLP) of the feed forward networks and the Kohonen's Self Organizing Map (SOM) of the feedback networks. One of the interesting characteristics of MLP is that in addition to classifying an input pattern, they also provide a confidence in the classification . These confidence values may be used for rejecting a test pattern in case of doubt. He has shown that Feed-forward, perceptron higher order network, Neuro-fuzzy system are better suited for character recognition.

MINIMUM REQUIREMENTS

Hardware Requirements

- **Processors:** Intel Atom® processor or Intel® Core™ i3 processor
- **Disk space:** 1 GB
- **Operating systems:** Windows* 7 or later, macOS, and Linux
- **Python* versions:** 2.7.X, 3.6.X
- **Included development tools:** conda*, conda-env, Jupyter Notebook* (IPython)
- **Compatible tools:** Microsoft Visual Studio*, PyCharm*
- **Included Python packages:** NumPy, SciPy, scikit-learn*, pandas, Matplotlib, Numba*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP*, and others.

Software Requirements

- **PIP and NumPy:** Installed with PIP, Ubuntu*, Python 3.6.2, NumPy 1.13.1, scikit-learn 0.18.2
- **Windows:** Python 3.6.2, PIP and NumPy 1.13.1, scikit-learn 0.18.2
- Intel® Distribution for Python* 2018

DATASET

Kaggle.com proved to be very useful for us in finding the dataset we needed for this project (<https://www.kaggle.com/rishianand/devanagari-character-set>).

We believe this is the largest dataset of Devanagari characters till date. This dataset only for the research purpose. The dataset consists of 92000 rows (sample images), and 1025 columns. Each row contains the pixel data ("pixel0000" to "pixel1023"), in grayscale values (0 to 255). The column "character" represents the Devanagari character name corresponding to each image.

Devanagari Handwritten Character Dataset is created by collecting the variety of handwritten Devanagari characters from different individuals from diverse fields. Handwritten documents are then scanned and cropped manually for individual characters. Each character sample is 32x32 pixels and the actual character is centered within 28x28 pixels. Padding of 0 valued 2 pixels is done on all four sides to make this increment in image size. The images were applied gray-scale conversion. After this the intensity of the images were inverted making the character white on the dark background. To make uniformity in the background for all the images, we suppressed the background to 0 value pixel. Each image is a gray-scale image having background value as 0. Devanagari Handwritten Character Dataset contains total of 92,000 images with 72,000 images in consonant data set and 20,000 images in numeral dataset.



1452



1464



8077



8084



8091



10986



10993



10997



11003



11025



13006



13009



17178



17183



17187

We have worked on the preprocessed csv file of the above dataset imported from Kaggle.

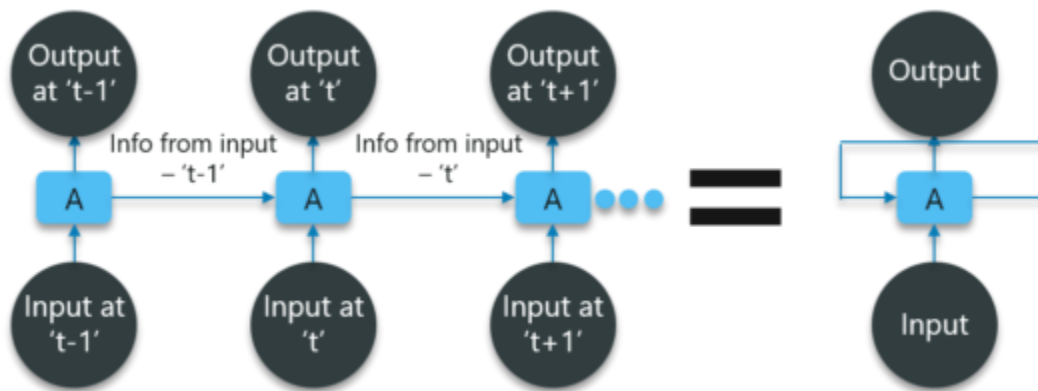
Source: <https://www.kaggle.com/rishianand/devanagari-character-set/home>

| AME | AMF | AMG | AMH | AMI | AMJ | AMK | AML |
|------------|------------|------------|------------|------------|------------|-------------------|-----|
| pixel_1018 | pixel_1019 | pixel_1020 | pixel_1021 | pixel_1022 | pixel_1023 | character | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 character_01_ka | |

IMPLEMENTATION

RNN

Recurrent Networks are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, numerical times series data emanating from sensors, stock markets and government agencies.



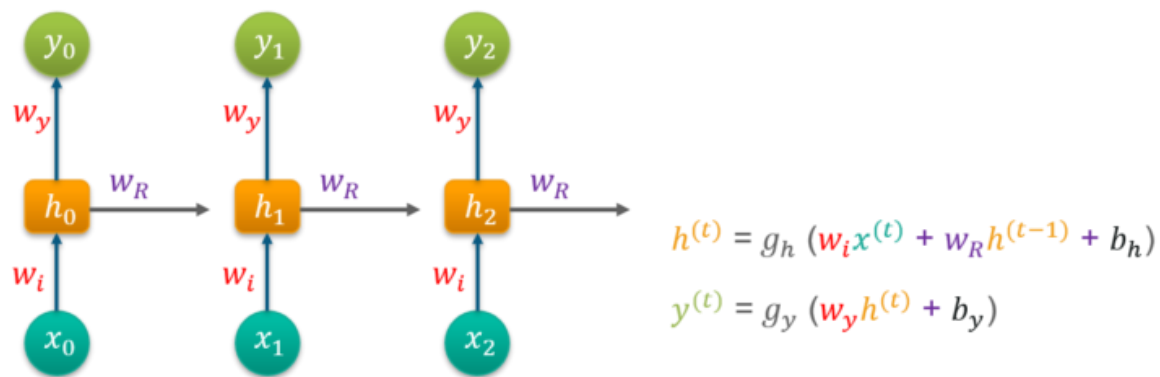
<https://www.edureka.co/blog/recurrent-neural-networks/>

In the above diagram, we have certain inputs at 't-1' which is fed into the network. These inputs will lead to corresponding outputs at time 't-1' as well.

At the next timestamp, information from the previous input 't-1' is provided along with the input at 't' to eventually provide the output at 't' as well.

This process repeats, to ensure that the latest inputs are aware and can use the information from the previous timestamp is obtained.

Working of RNN:



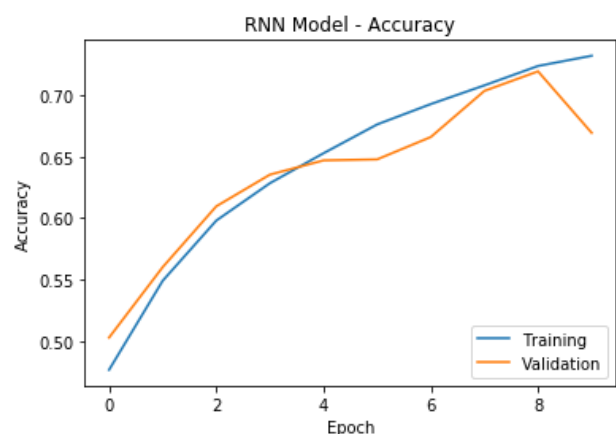
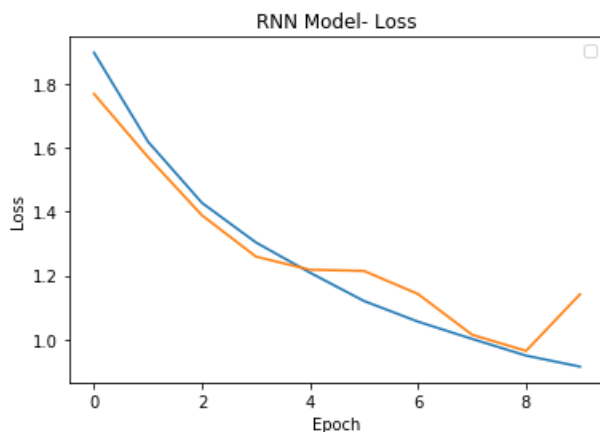
Source: <https://www.edureka.co/blog/recurrent-neural-networks/>

Consider ‘w’ to be the weight matrix and ‘b’ being the bias:

At time $t=0$, input is ‘ x_0 ’ and the task is to figure out what is ‘ h_0 ’. Substituting $t=0$ in the equation and obtaining the function $h(t)$ value. Next, the value of ‘ y_0 ’ is found out using the previously calculated values when applied to the new formula.

This process is repeated through all of the timestamps in the model to train a model.

The graphs below shows the Loss and Accuracy of RNN Model respectively on training and testing dataset.

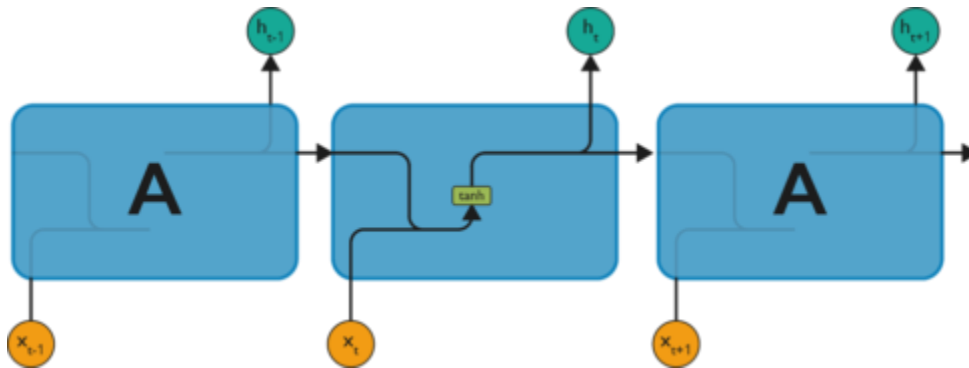


LSTM

Long Short-Term Memory networks are usually just called “LSTMs”.

They are a special kind of Recurrent Neural Networks which are capable of learning long-term dependencies.

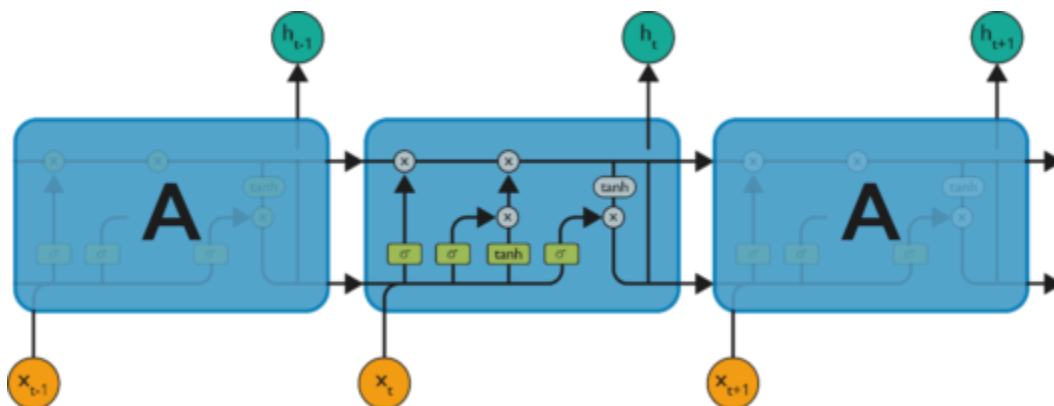
LSTM have chain-like neural network layer. In a standard recurrent neural network, the repeating module consists of one single function as shown in the below figure:



Source: <https://www.edureka.co/blog/recurrent-neural-networks/>

As shown above, there is a tanh function present in the layer. This function is a squashing function. It is a function which is basically used in the range of -1 to +1 and to manipulate the values based on the inputs.

Now, let us consider the structure of an LSTM network:



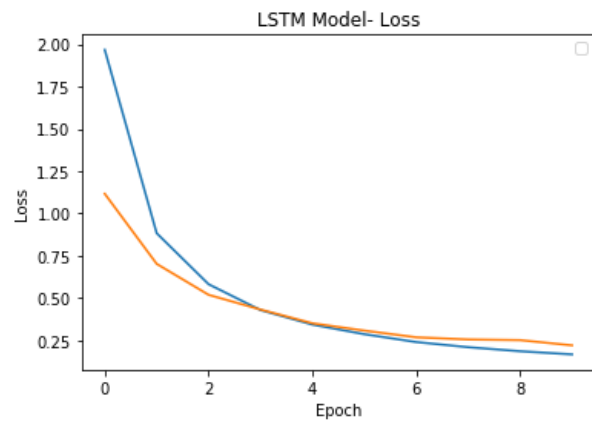
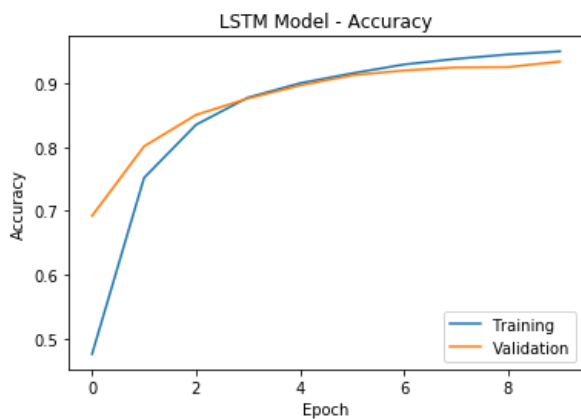
Source: <https://www.edureka.co/blog/recurrent-neural-networks/>

As denoted from the image, each of the functions in the layers has their own structures when it comes to LSTM networks. The cell state is the horizontal line in

the figure and it acts like a conveyor belt carrying certain data linearly across the data channel.

Steps :

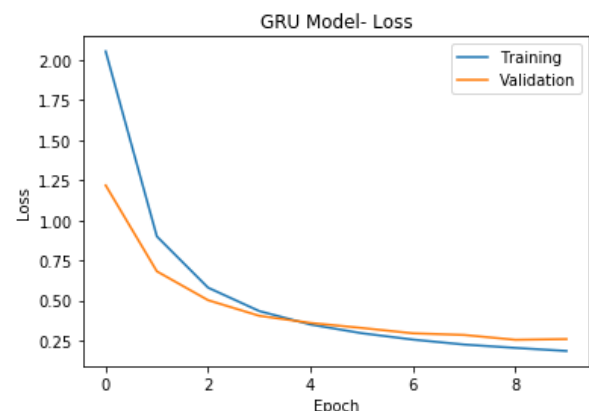
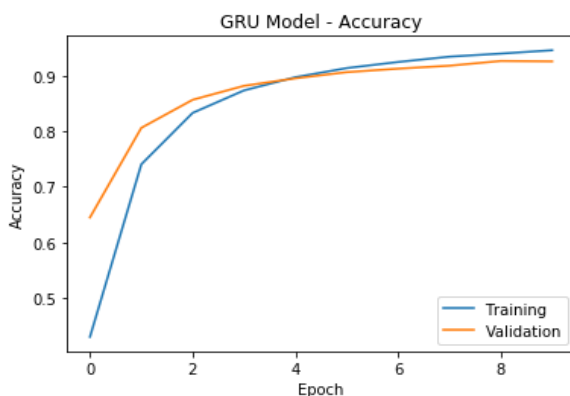
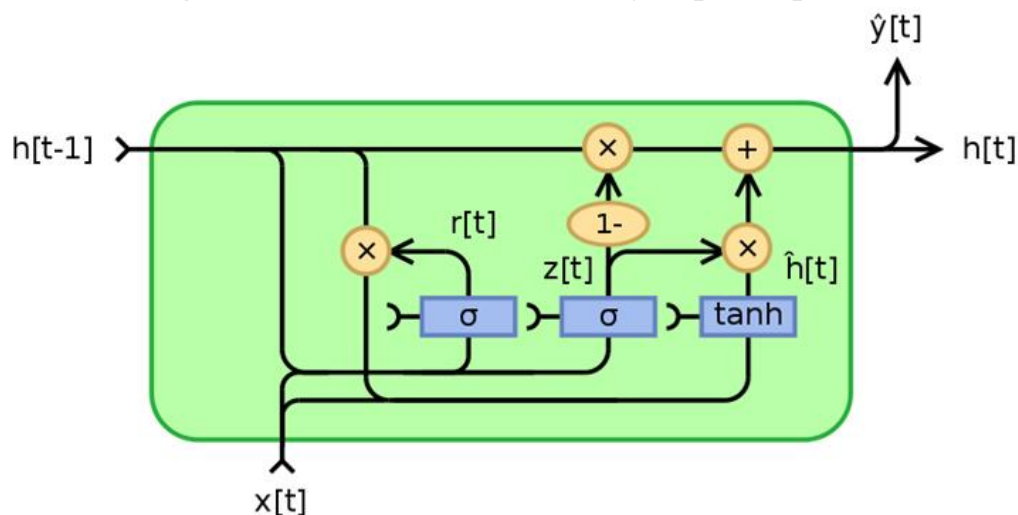
1. In the first step, we found out what was needed to be dropped.
2. The second step consisted of what new inputs are added to the network.
3. The third step was to combine the previously obtained inputs to generate the new cell states.
4. Lastly, we arrived at the output as per requirement.



GRU

The idea behind a GRU layer is quite similar to that of a LSTM layer. A GRU has two gates, a reset gate r , and an update gate z . Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. If we set the reset to all 1's and update gate to all 0's we again arrive at our plain RNN model. The basic idea of using a gating mechanism to learn long-term dependencies is the same as in a LSTM, but there are a few key differences:

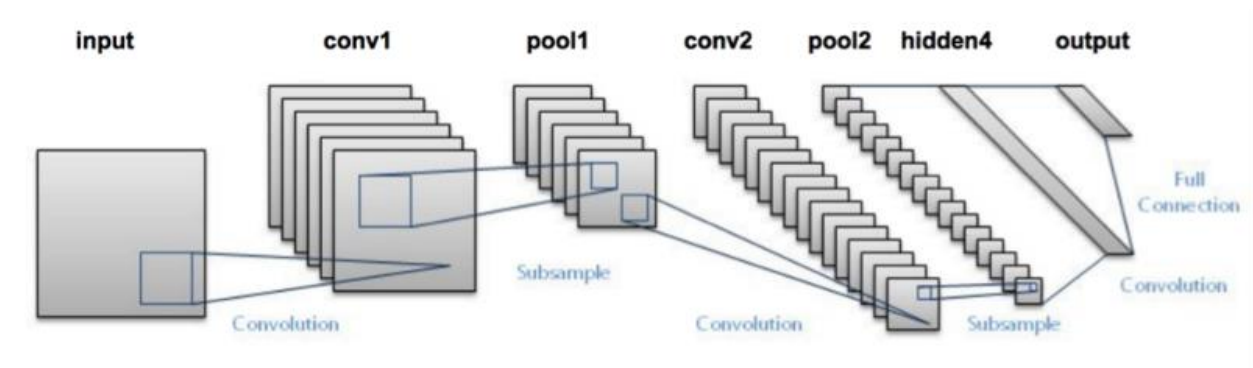
- A GRU has two gates, an LSTM has three gates.
- GRUs don't possess an internal memory (c_t) that is different from the exposed hidden state. They don't have the output gate that is present in LSTMs.
- The input and forget gates are coupled by an update gate z and the reset gate r is applied directly to the previous hidden state. Thus, the responsibility of the reset gate in a LSTM is really split up into both r and z .



CNN

Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weight sand biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on Convolutional Neural Networks.



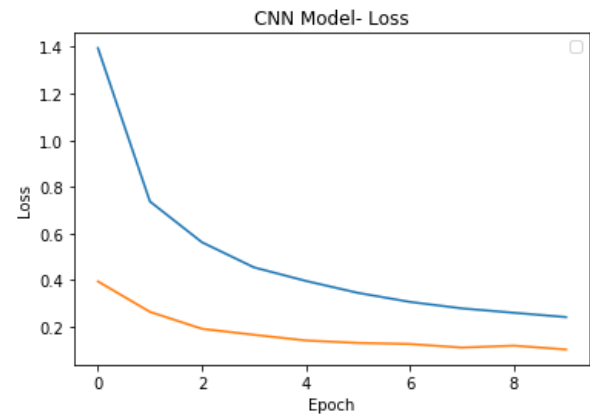
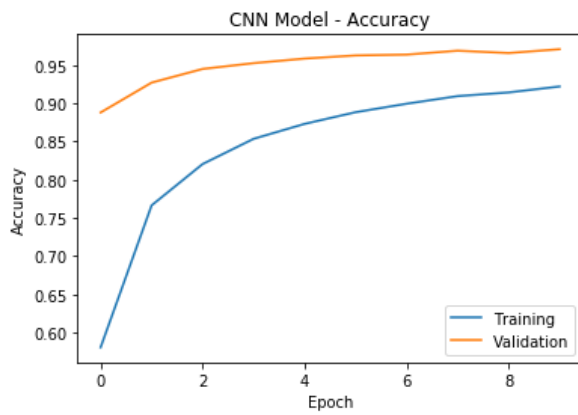
Source: www.analyticsvidya.com/CNN

Steps for CNN :

- We pass an input image to the first convolutional layer. The convoluted output is obtained as an activation map. The filters applied in the convolution layer extract relevant features from the input image to pass further.
- Each filter shall give a different feature to aid the correct class prediction. In case we need to retain the size of the image, we use same padding(zero padding), otherwise valid padding is used since it helps to reduce the number of features.
- Pooling layers are then added to further reduce the number of parameters
- Several convolution and pooling layers are added before the prediction is made. Convolutional layer help in extracting features. As we go deeper in the network more specific features are extracted as compared to a shallow network where the features extracted are more generic.
- The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.
- The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined in the fully

connected output layer to compute the mean square loss. The gradient of error is then calculated.

- The error is then back propagated to update the filter(weights) and bias values.
- One training cycle is completed in a single forward and backward pass.



PROJECT SNAPSHOTS

Layers of RNN Model

| Layer (type) | Output Shape | Param # |
|---------------------------|----------------|---------|
| ===== | ===== | ===== |
| simple_rnn_1 (SimpleRNN) | (None, 32, 50) | 4150 |
| simple_rnn_2 (SimpleRNN) | (None, 32, 16) | 1072 |
| simple_rnn_3 (SimpleRNN) | (None, 32) | 1568 |
| dense_1 (Dense) | (None, 46) | 1518 |
| activation_1 (Activation) | (None, 46) | 0 |
| ===== | ===== | ===== |
| Total params: 8,308 | | |
| Trainable params: 8,308 | | |
| Non-trainable params: 0 | | |
| ===== | | |
| None | | |

Layers of LSTM Model

| Layer (type) | Output Shape | Param # |
|---------------------------|----------------|---------|
| ===== | ===== | ===== |
| lstm_13 (LSTM) | (None, 32, 50) | 16600 |
| lstm_14 (LSTM) | (None, 32, 16) | 4288 |
| lstm_15 (LSTM) | (None, 32) | 6272 |
| dense_4 (Dense) | (None, 46) | 1518 |
| activation_4 (Activation) | (None, 46) | 0 |
| ===== | ===== | ===== |
| Total params: 28,678 | | |
| Trainable params: 28,678 | | |
| Non-trainable params: 0 | | |
| ===== | | |
| None | | |

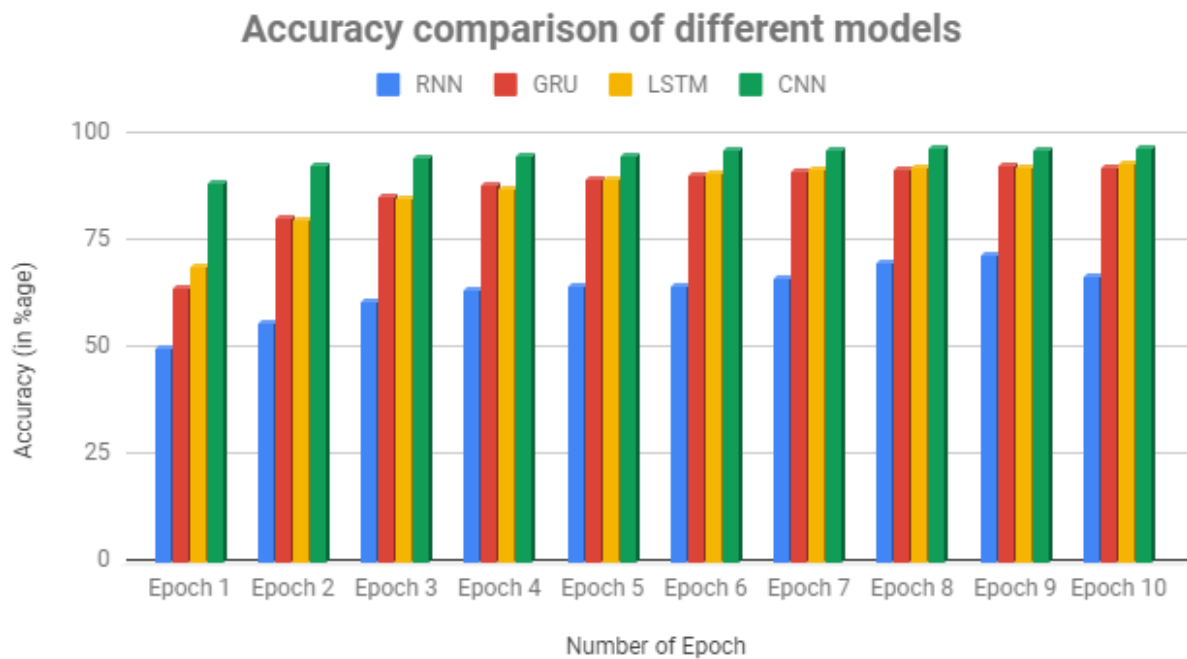
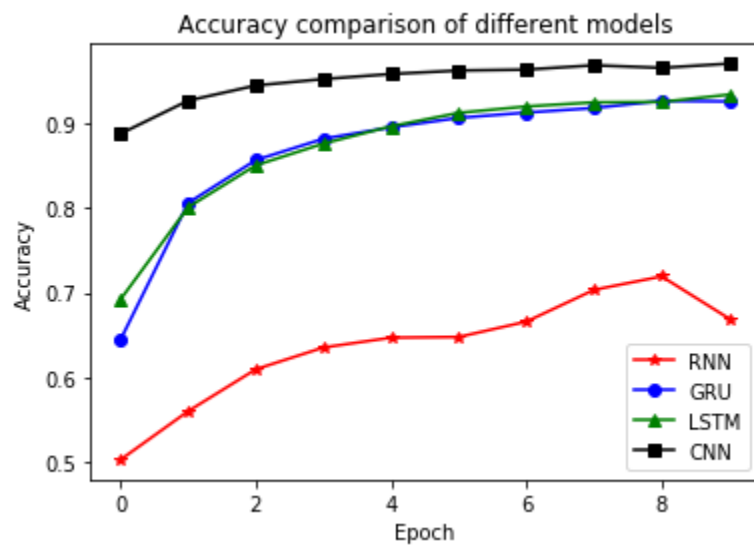
Layers of GRU Model

| Layer (type) | Output Shape | Param # |
|---------------------------|----------------|---------|
| gru_1 (GRU) | (None, 32, 50) | 12450 |
| gru_2 (GRU) | (None, 32, 16) | 3216 |
| gru_3 (GRU) | (None, 32) | 4704 |
| dense_6 (Dense) | (None, 46) | 1518 |
| activation_6 (Activation) | (None, 46) | 0 |
| Total params: 21,888 | | |
| Trainable params: 21,888 | | |
| Non-trainable params: 0 | | |
| None | | |

Layers of CNN Model

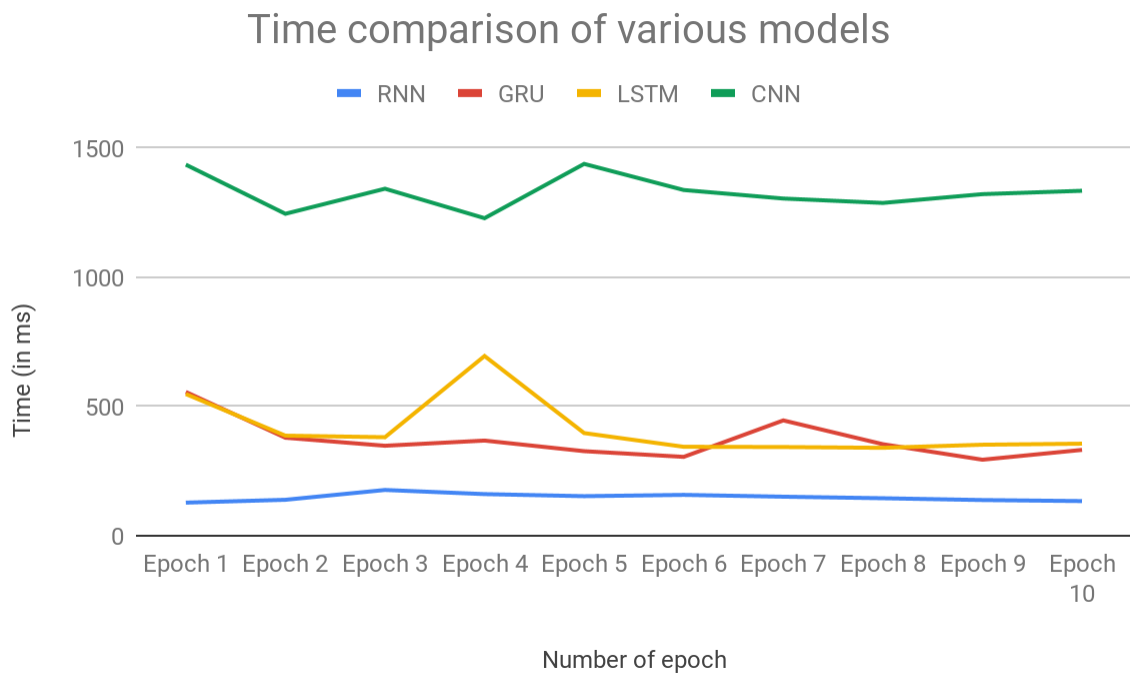
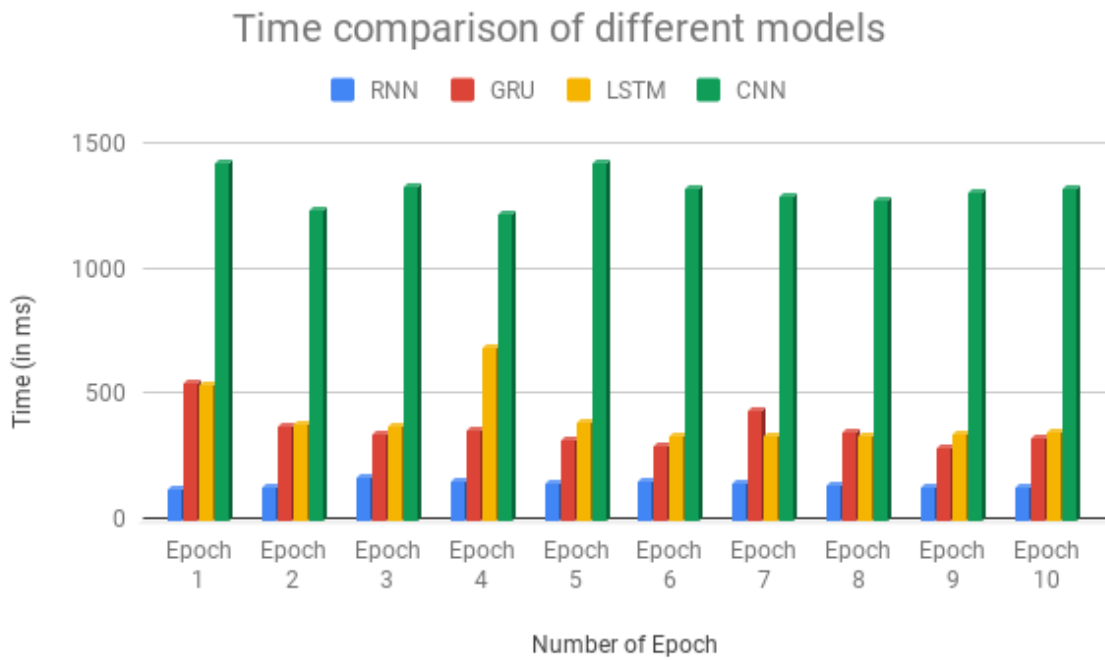
| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d_17 (Conv2D) | (None, 30, 30, 32) | 320 |
| conv2d_18 (Conv2D) | (None, 28, 28, 64) | 18496 |
| max_pooling2d_8 (MaxPooling2D) | (None, 14, 14, 64) | 0 |
| dropout_15 (Dropout) | (None, 14, 14, 64) | 0 |
| flatten_8 (Flatten) | (None, 12544) | 0 |
| dense_26 (Dense) | (None, 128) | 1605760 |
| dropout_16 (Dropout) | (None, 128) | 0 |
| dense_27 (Dense) | (None, 64) | 8256 |
| dense_28 (Dense) | (None, 46) | 2990 |
| Total params: 1,635,822 | | |
| Trainable params: 1,635,822 | | |
| Non-trainable params: 0 | | |
| None | | |

Accuracy Graphs:



Comparison of Accuracy between various Deep Learning Algorithms

Time Graphs:



ANALYSIS & RESULTS

| Algorithms | Parameters | Accuracy | Mean Time for Training per epoch (ms) |
|------------|---|----------|---------------------------------------|
| CNN | optimizer='adam', loss='categorical_crossentropy', metrics='accuracy', activation='softmax' | 97.05 | 20.6 |
| RNN | optimizer='adam', loss='categorical_crossentropy', metrics='accuracy', activation='softmax' | 66.94 | 2 |
| LSTM | optimizer='adam', loss='categorical_crossentropy', metrics='accuracy', activation='softmax' | 93.4 | 6.4 |
| GRU | optimizer='adam', loss='categorical_crossentropy', metrics='accuracy', activation='softmax' | 92.59 | 5.8 |

On implementing the four state-of-the-art algorithms using same parameters on the Devanagari dataset, we deduced the following results:

- In terms of accuracy CNN is still the best among the four algorithms with the accuracy of 97.05%.
- Though LSTM and GRU were very much close to the CNN with accuracies 93.4% and 92.59% respectively.
- RNN showed the worst results with the accuracy of just 66.94%.
- Though in terms of training time per epoch RNN was much faster than other three algorithms with average training of 2ms only.
- LSTM and GRU again showed close results with 6.4ms and 5.8ms per epoch.
- CNN was the slowest with average time 20.6ms.

CONCLUSION

- The above analysis and results conclude that LSTMs and GRUs are good alternatives to CNN in handwritten recognition because of the faster training time.
- In case there is no time factor involved, CNN is still the best for handwritten algorithm.
- Future OCRs can safely use LSTMs and GRUs for the recognition with better and improved architectures which might even give better accuracy results than CNN.

FUTURE SCOPE

- The results of the project can be used to design a better OCR for handwritten recognition not only for Hindi Characters but also for the languages which have similar challenges.
- In future, LSTMs or GRUs can even have better accuracies than CNN with better and improved architecture along with the advantage of faster training time.
- The behaviour of the graphs can even yield some state-of-the-art results which haven't been explored yet.
- This project can be combined with algorithms that segment the character images in a given word image, which can in turn be combined with algorithms that segment the character images in a given image of a whole handwritten page.

BIBLIOGRAPHY

1. S. Acharya, A.K. Pant and P.K. Gyawali Deep Learning Based Large Scale Handwritten Devanagari Character Recognition ,In Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 121-126, 2015.
2. M.s. Neha Sahu, Mr.Nitin Kali Raman, “An Efficient Handwritten Devnagari Character Recognition System Using Neural Network”, 978-I-4673-5090-7/13/\$31.00 ©2013 IEEE,2013.
3. B.Indira, M.Shalini 1, M.V. Ramana Murthy, Mahaboob Sharief Shaik, “Classification and Recognition of Printed Hindi Characters Using Artificial Neural Networks”, I.J. Image, Graphics and Signal Processing, 2012
4. Divya Sharma “Recognition of Handwritten Devanagari Script using Soft Computing”, Thesis submitted to Thapar University, Patiala, June 2009.
5. Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, L.Malik, M. Kundu and D.K. Basu, “Performance Comparison of SVM and ANN for Handwritten Devanagari Character Recognition”, International Journal of Computer Science Issues, pp 18-26, Vol. 7, Issue 3, No. 6, may 2010.
6. <https://www.kaggle.com/ashokpant/devanagari-character-dataset>
7. <https://www.edureka.co/blog/convolutional-neural-network/>
8. <https://www.edureka.co/blog/recurrent-neural-networks/>