International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August 2019

A Novel Deep Neural Network Model for Image Classification

N.Karthika, B.Janet, Himanshu Shukla

Abstract—In this article, we have trained neural network based on deep learning architectures to classify images on standard Fashion-MNIST and CIFAR-10 dataset. The various CNN- based classification architecture and RNN-based classification architecture are trained as well as tested on those standard datasets. In CNN architecture, we include CNN with 1, 2 and 3 Convolutional Layer and in RNN architecture, we include Long- Short Term Memory (LSTM) with one and two LSTM layer. Our models show remarkable outcome on the standard benchmark dataset. The tested models like CNN1 show greater accuracy on the MNIST fashion dataset and CNN3, LSTM1 and LSTM2 performed better than other models on the CIFAR-10 dataset.

Keywords: Deep Neural Networks, CNN, LSTM, Multi-Class Classifications.

I. INTRODUCTION

Typical machine learning techniques like Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA), and Support Vector Machine (SVM) were used in the analysis of image information [1]. However, when we deal with large-scale image information, these techniques still have constraints. Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), is being implemented to address the problem of absence of ability to manage large-scale and unstructured data [2]. Deep learning techniques have been commonly used to estimate and classify issues such as image captioning, object detection, speech recognition, age recognition, stock price prediction to name a few [3]. In this article, the MNIST fashion images and CIFAR-10 images are classified with Convolutional Neural Networks and Recurrent Neural Networks and their predictive performance and results are analyzed. Rectified Linear units (ReLU) are used as an activation function for hidden layers along with SoftMax as a classification function in the deep learning model.

Revised Manuscript Received on August 30, 2019.

* Correspondence Author

N.Karthika*, Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Trichy, Tamilnadu, India. Email: bharathikarthika@gmail.com

B.Janet, Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Trichy, Tamilnadu, India. Email: janet@nitt.edu

Himanshu Shukla, Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Trichy, Tamilnadu, India. Email: himshukla007@gmail.com

II. METHODOLOGY

Google TensorFlow [1][4] backend is utilized along with various other compute libraries such as matplotlib [7] [5], numpy[14] [6], and Scikit-learn[11] [7].

A. Convolutional Neural Network

A Convolutional Neural Network (CNN or ConvNet) in deep learning is a class of deep neural networks and most frequently is used for visual image analysis [2] [8]. CNNs are built up of neurons with learnable weights and biases. Every neuron takes multiple inputs, collects a weighted sum over them after it is transferred over an activation function and responds with an output [9]. Technically, every input image will proceed through a sequence of convolution layers with filters (Kernels), pooling, fully linked layers (FC) and use SoftMax [10] to categorize an object with probabilistic values as 0 to 1 [11].

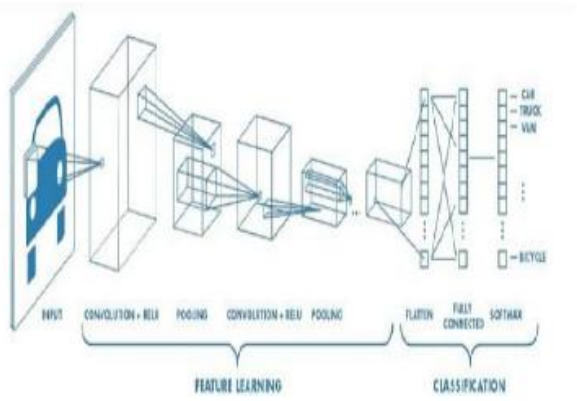

Fig. 1: Convolutional Neural Network

Convolution is the first layer to extract characteristics of the image. Convolution maintains the connection between pixels through the use of tiny squares of input information to learn image characteristics. It is a mathematical operation that requires two inputs such as the matrix of the image and a filter or kernel.

$$S(i,j) = (l * K)(i,j) = \sum_m \sum_n l(m,n) K(i-m, j-n)$$

where K is Kernel(Convolution function), l is input(may be 2D Array)

For 3*3 kernel, the equation becomes

$$S(i,j) = (l * K)(i,j) = \sum_{m=1}^{3} \sum_{n=1}^{3} l(i-m, j-n) K(m,n)$$

We will apply a non-linear activation function to the resulting dot product in the next phase. The result will be x = max(0,x). The operation is

$$A(i,j) = \begin{cases} 0, & if \ (i,j) < 0 \\ \text{S(i,j)}, & if \ \text{(i,j)} \geq 0 \end{cases}$$

It operates with width and height of the data information and accomplish down sampling on them. The volume of the data is decreased as a consequence. This implies that if certain characteristics (such as boundaries) have been recognized in the past convolution operation, a comprehensive information is no longer required for further processing. It is compressed to remove detailed data information. It is essential to add a fully connected layer after completing the sequence of convolutional, nonlinear and pooling layers. The output data from convolution networks is taken from this layer. Attaching a fully linked layer to the last stage of the network outcomes in an N dimensional vector, where N is the number of classes from which the model chooses the required class [4].

B. Long Short-Term Memory networks

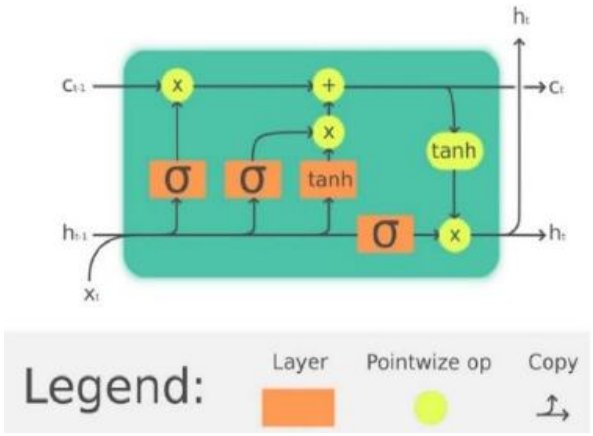

Fig. 2: Long Short-Term Memory networks

Long Short-Term Memory networks (LSTMs) are a primitive type of RNN that can learn long-term dependencies. It was introduced by Hochreiter Schmidhuber (1997). It operates very well on a wide range of issues and are nowadays commonly in use for many applications such as natural language processing and image recognition [13]. LSTMs have a chain like structure as a framework, but there is a distinct structure in the repeating module.

A popular LSTM unit consists of a cell, an input gate, an output gate and a forget gate [13]. The cell recalls values over arbitrary times of moment, and the other three control data flows into and out of the cell.

Cell state: Long term memory is also known as cell state. It is recursive in nature. By this, the information from earlier intervals is saved in the LSTM cell. The state of the cell is changed by forget gate.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Where f_t is forget gate, i_t is output from sigmoid, C_{t-1} is output from previous cell state, $\sim C_t$ output from modulation input gate.

Forget Gate: It is also known as remember vector. The yield of the forget gate informs the cell state which data should be forgotten by multiplying zero to a matrix position. If the forget gate produces one, the data will be held in the state of the cell.

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f)$$

Where f_t is forget gate, W_f is weight, h_{t-1} is output from previous timestamp t-1, x_t is current information, b_f is bias.

Input Gate: The other name of the input gate is save vector. These gates decide the data that should arrive at the cell state. It has input gate and input modulation gate as well. The significant components are activation functions of each gates. The input gate has a sigmoid function which varies in the range of [0,1]. So, when added with earlier cell state, the sigmoid function will add memory and therefore not be able to remove memory. So that the input modulation gate has tanh activation function which ranges from [-1,1] and allows to forget selective memory.

$$i_t = \sigma(W_f . [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C . [h_{t-1}, x_t] + b_C)$$

Where i_t is input gate, W_f, and W_c is weight, h_{t-1} is output from previous timestamp t-1, x_t is current information, b_f is bias, $\sim C_t$ is input modulation gate.

Output Gate: This is also referred to as the focus vector. Of all the feasible matrix values, it determines which will move to the next hidden state.

$$O_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$$

Where O_t is output gate, W_o is weight, h_{t-1} is output from previous timestamp t-1, x_t is current information, b_f is bias.

Hidden State: The hidden state is termed as the working memory.

$$h_t = O_t * \tanh(C_t)$$

Where h_t is hidden gate, O_t is output gate, C_t is cell state.

C. Deep LSTM

Deep LSTM RNNs are constructed by piling up of various layers of LSTM. The depth of profound LSTM RNNs owns an extra significance, the input to the network at a specified time step passes through various layers of LSTM in relation to time propagation and layers of LSTM [9] [15]. Benefit of using Deep LSTM RNNs to ordinary LSTM are they could better utilize parameters by sharing them across various layers. Deep LSTM helps to prevent the network overfit.

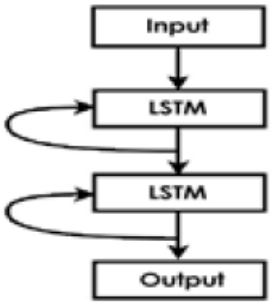

Fig. 3: Deep Long Short-Term Memory networks

Retrieval Number F8832088619/2019©BEIESP
DOI: 10.35940/ijeat.F8832.088619
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

D. Dropout

A fully linked layer occupies most of the parameters and therefore, during training, neurons create co-dependence between each other that curbs the individual strength of each neuron, resulting in over-fitting of training data. At the time of training, some number of layer outputs are deliberately neglected or dropped out [16]. Regularization prevents over-fitting in machine learning. By applying a penalty to the loss function, regularization decreases overfitting [17]. The model is trained by incorporating this penalty to prevent it from learning interdependent set of weights of attributes.

Dropout is a regularization strategy that helps to reduce interdependent neuronal learning. It approximately doubles the amount of iterations needed to converge. Nevertheless, time to train for every epoch is less. With H hidden units, each of which can be dropped, we have $2^{(H)}$ possible models. The whole network is counted in the testing stage and each activation is decreased by a random node fraction, p.

III. DATASET

A. Fashion-MNIST Dataset

Fashion-MNIST is a dataset of Zalando's substance pictures composed of 60,000 examples in training set and 10,000 examples in test set. Each substance is a grayscale image of 28x28 that is connected with any one of the 10-classes. Fashion-MNIST is a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. The fashion MNIST dataset is downloadable from GitHub. The sample image is shown in fig.1 where each row comprises one fashion item.

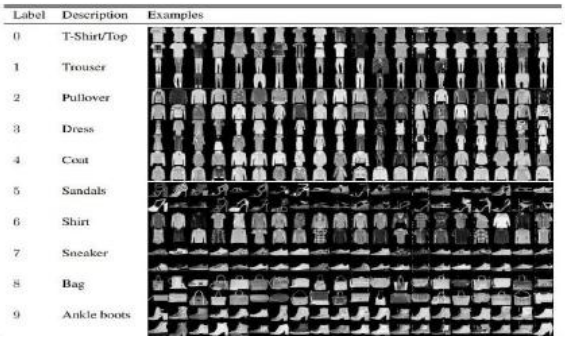

Fig. 4: Fashion-MNIST Dataset

There are some specific reasons why we select this fashion-MNIST dataset. The first is some researchers use relatively new methods, so we have a large number of modern results to compare with and it has a good level of complexity and some classifiers can hardly achieve a perfect score on it, so there is still scope for optimization i.e. The Fashion-MNIST dataset guarantees to be more varied so that algorithms in machine learning (ML) need to learn more sophisticated characteristics to be able to reliably distinguish the individual classes.

B. CIFAR-10 Dataset

The CIFAR-10 is a labeled subgroup of 80 million tiny images dataset. The dataset of CIFAR-10 contains 60000 32x32 color images in 10 groups, with 6000 images for every class. There are 50,000 images of training and 10,000 images of testing. The classes are totally excluded from each other. Automobiles and trucks do not overlap. "Automobile" involves such stuff as sedans, SUVs, etc. Only large trucks are included in" Truck."

Fig. 5: CIFAR-10 Dataset

IV. EXPERIMENTAL RESULTS AND ANALYSIS

We have attempted five network models here in this work on the two conventional datasets outlined above. The first model of the network has a single layer of 128 filters. The first convolutional network's kernel size is 3 * 3. In this layer, a ReLU activation is implemented. Following this, 2 * 2 kernel size MaxPooling is used. The dropout of 20% is then used to regulate overfitting [18]. Finally, we have SoftMax layer to compute the loss function.

The second model of the network has two convolutional layers of respectively 32 and 64 filters. For both layers, the kernel size of 3 * 3 is used. An ReLU activation is used in the same way as the first network model for each layer. We use max pooling of 2 * 2 kernel size after each layer. 20% of dropout is implemented after the first layer. 25% of dropout is used after the second layer to prevent overfitting. SoftMax is employed in the final layer to calculate the loss function.

Architecture	Train Accuracy	Validate Accuracy	Test Accuracy
CNN with 1 layer	94.23	90.32	92.02
CNN with 2 Layer	92.55	90.81	89.42
CNN with 3 layer	84.01	87.26	86.33
LSTM with 1 layer	87.87	86.37	85.58
LSTM with 2 layer	93.13	90.62	92.01

TABLE I: Fashion MNIST Data-Accuracy

The third network model has three convolution layers of respectively 32,64, and 128 filters. In the first layer, the kernel size of 5 * 5 is used. 3 * 3 kernel size is used in the second and third layers. Activation of ReLU is implemented in all three layers. Kernel size 3 * 3 max pooling is performed after the first layer and 2 * 2 kernel size max pooling is enforced after the second and third layer. The 25% dropout is implemented after the first layer and the second layer, while the 20% dropout is activated after the third layer to lower the overfitting. Then lastly SoftMax is fully connected to estimate the loss function as in past network models.

Architecture	Train loss	Validate loss	Test loss
CNN with 1 layer	38.45	34.45	34.82
CNN with 2 Layer	19.37	26.64	29.71
CNN with 3 layer	44.19	35.16	37.38
LSTM with 1 layer	33.07	38.05	31.47
LSTM with 2 layer	18.56	26.54	31.22

TABLE II: Fashion MNIST Data-Loss

The fourth and fifth network models are based on recurrent neural network. Usually the conventional RNNs are sometimes volatile, especially when propagating gradients through long windows that may trigger gradient vanishing or explosion. Therefore, the model is developed using the LSTM technique. The fourth model uses a sequential model that is a linear layer stack. The model is tested on 100 batch size. The first layer is a 128-unit LSTM layer and returns sequences. This is performed to guarantee that sequences are received by the next LSTM layer and not just randomly dispersed data. Finally, as a fully connected layer, we have the last layer with a Soft-Max activation and neurons equivalent to the number of distinctive classes.

The fifth model is also similar to fourth model i.e. it is also a stack of linear layers. The first layer is an LSTM layer with 300 memory units in place of the past model's 128 units and returns sequences. There are again 300 units in the second layer. After second LSTM layer, a dropout layer is implemented to prevent model over-fitting. Finally, as a fully connected layer, we have the last layer with SoftMax activation. The summary of various network models examined is shown in tables III, IV and the architecture of Models shown in figures from 16 to 19.

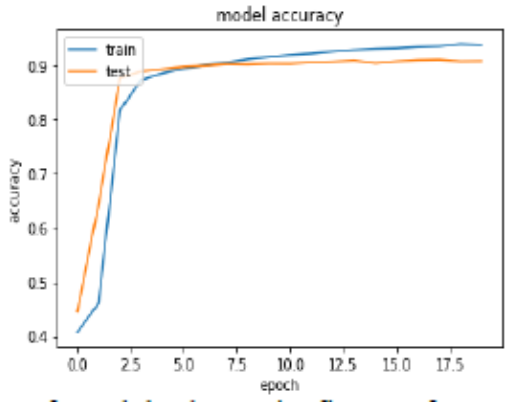

Fig. 6: Fashion MNIST Dataset-CNN with one layer

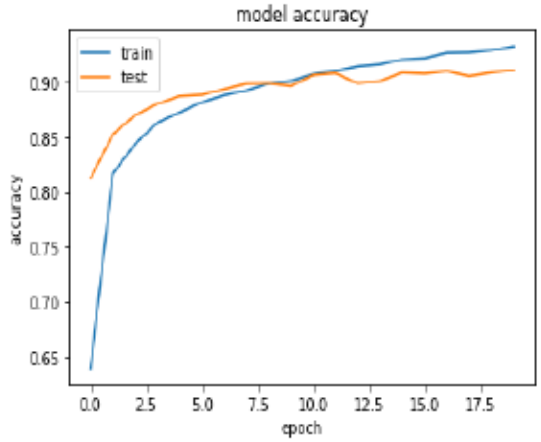

Fig. 7: Fashion MNIST Dataset-CNN with two layers

Summary of Model analysed		
Cnn1 Model		
Layer	**Output Shape**	**Parameter**
Conv2D-1 (Conv2D)	(None,26,26,128)	1280
MaxPool (MaxPooling2D)	(None,13,13,128)	0
Dropout (Dropout)	(None,13,13,128)	0
flatten (Flatten)	(None,21632)	0
Dense (Dense)	(None,32)	692256
Output (Dense)	(None,10)	330
CNN2 Model		
Layer	**Output Shape**	**Parameter**
Conv2D-1 (Conv2D)	(None,26,26,32)	320
MaxPool-1 (MaxPooling2D)	(None,13,13,32)	0
Dropout-1 (Dropout)	(None,13,13,32)	0
Conv2D-2 (Conv2D)	(None,11,11,64)	18496
MaxPool-2 (MaxPooling2D)	(None,5,5,64)	0
Dropout-2 (Dropout)	(None,5,5,64)	0
flatten (Flatten)	(None,1600)	0
Output (Dense)	(None,10)	16010
CNN3 Model		
Layer	**Output Shape**	**Parameter**
Conv2D-1 (Conv2D)	(None,26,26,32)	320
MaxPool-1 (MaxPooling2D)	(None,13,13,32)	0
Dropout -1(Dropout)	(None,13,13,32)	0
Conv2D-2 (Conv2D)	(None,11,11,64)	18496
MaxPool-2 (MaxPooling2D)	(None,5,5,64)	0
Dropout-2 (Dropout)	(None,5,5,64)	0
Conv2D-3 (Conv2D)	(None,3,3,128)	73856
MaxPool-3 (MaxPooling2)	(None,1,1,128)	0
Dropout -3(Dropout)	(None,1,1,128)	0
flatten (Flatten)	(None,128)	0
Dense (Dense)	(None,128)	16512
Output (Dense)	(None,10)	1290

TABLE III: Summary of Convolutional network models analyzed.

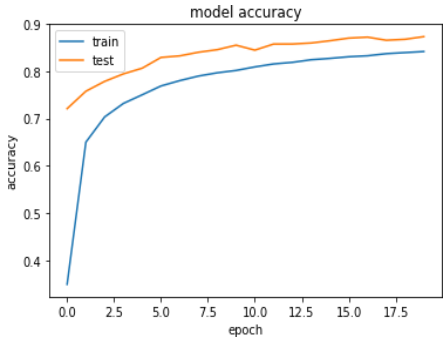

Fig. 8: Fashion MNIST Dataset-CNN with three layers

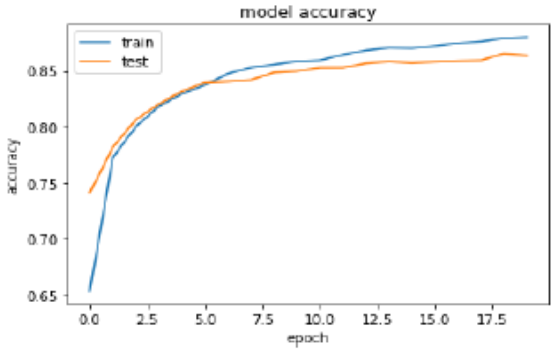

Fig. 9: Fashion MNIST Dataset-RNN with one layer

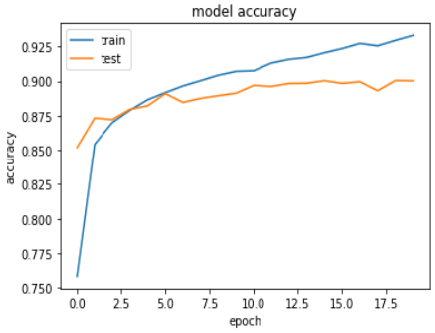

Fig. 10: Fashion MNIST Dataset-RNN with two layers

Summary Of Models Analyzed					
Long Short Term Memory					
LSTM1 Model			LSTM2 Model		
Layer	Output Shape	Parameter	Layer	Output Shape	Parameter
lstm-1 (LSTM)	(None,128)	80384	lstm-1 (LSTM)	(None,32,300)	399600
dense-1 (Dense)	(None,10)	1290	lstm-2 (LSTM)	(None,300)	721200
			dropout-1 (Dropout)	(None,300)	0
			dense-1 (Dense)	(None,10)	3010

TABLE IV: Summary of LSTM Models Analyzed

The respective train accuracy, test accuracy and validation accuracy of the Fashion MNIST and CIFAR-10 are shown in the tableI and tableV as well as the accuracy plot of the respective datasets shown from figure6 to figure15. The associated test loss, train loss and validation loss shown in tableII for Fashion MNIST data and for CIFAR-10 is shown in tableVI.

A. CIFAR-10 Dataset

CIFAR-10 Accuracy			
Architecture	TrainAccuracy	ValidateAccuracy	TestAccuracy
CNN with 1 layer	97.9	69.3	69.6
CNN with 2 Layer	96.5	77.0	76.9
CNN with 3 layer	93.2	81.11	80.2
LSTM with 1 layer	86.5	85.4	83.2
LSTM with 2 layer	89.2	92.1	90.2

TABLE V: CIFAR-10 - Accuracy of various networks

CIFAR 10 Dataset Loss			
Architecture	Train loss	validate loss	Test loss
CNN with 1 layer	0.6	15.2	15.1
CNN with 2 Layer	0.9	9.8	9.6
CNN with 3 layer	1.9	6.9	7.1
LSTM with 1 layer	0.3	0.2	1.6
LSTM with 2 layer	0.2	0.1	0.2

TABLE VI: CIFAR-10 - Loss of various networks

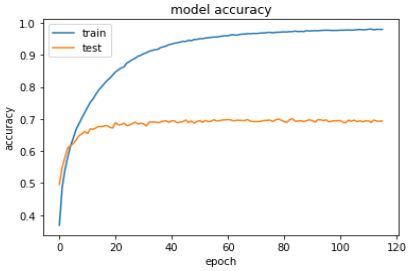

Fig. 11: CIFAR-10 Dataset-CNN with one layer

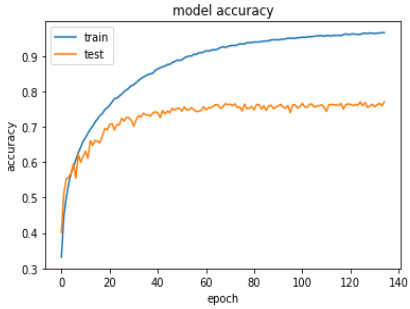

Fig. 12: CIFAR-10 Dataset-CNN with two layers

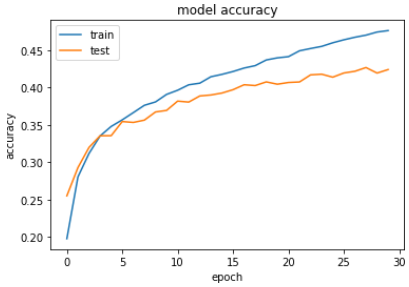

Fig. 13: CIFAR-10 Dataset-CNN with three layers

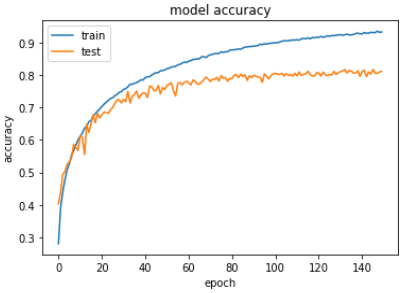

Fig. 14: CIFAR-10 Dataset-RNN with single layer

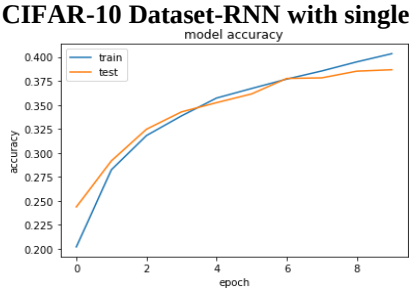

Fig. 15: CIFAR-10 Dataset-RNN with two layers

S.no	Method	Cifar-10	
		Neural Models	Non-Neural Models
1.	CNN3(This Work)	**80.2**	
2.	LSTM1(This Work)	**83.2**	
3.	LSTM2(This Work)	**90.2**	
4.	S-CNN (64-128-256_512) [19]	72.68	
5.	S-CNN (92-256-512_1024) [19]	75.17	
6.	Evolutionary deep learning framework [20]	74.5	
7.	Kmeans-triangle(4000 features) [21]		79.6
8.	Sparse autoencoder [21]		73.4

TABLE VII: CIFAR-10 - Comparison of Accuracy with other existing approaches

The table.VII shows the accuracy attained by different models on the CIFAR-10 dataset.

In the model S-CNN (64-128-256 512) [19], the author used three convolution layers with 64, 128 and 256 filter units together with stride 1 and padding 2 and the author achieved data accuracy of 72.68% while in the model S-CNN (92- 256-512 1024) [19], the author used longer models than the prior model. They used 92,256,512 and 1024 filters together with stride 2 and zero padding and attained accuracy of 75.17% which is bigger than our tested model. The author [20] attempted evolutionary deep neural networks, i.e. the generated chromosomes are passed to a network of 10,100 filters of convolutional nodes and max-pooling and embedding of size 100,300. Finally, fully connected dense layer with the Adam optimizer to achieve 74.5% accuracy. The table. VIII shows the summary of models used in [19], and [20].

The KMeans with triangle activation with 4000 features achieved 79.6% accuracy [21] and Sparse auto encoder un-supervised learning tried to obtain a feature vector and trained with back propagation to avoid squared reconstruction error. Thus achieved 73.4% accuracy with varying features. It shows that the network model's specific selection is more essential than the selection of supervised or unsupervised algorithm. Out of tested models CNN3, LSTM1 and LSTM2 (which are described in detail above) performed better than other models on the standard CIFAR-10 dataset.

S.no	Methods	Neural Models	Non -Neural Model
1.	CNN1(This work)	**92.02**	
2.	Evolutionary deep learning framework [20]	90.60	
3.	CNN using SVM activation [22]	90.72	
4.	CNN using Softmax activation [22]	91.86	
5.	CNN2 [23]	91.17	
6.	LSTM without network pruning [24]	88.26	
7.	LSTM with pruning [25]	89.94	
8.	SVC C = 10 with kernel : rbf [26]		89.70

TABLE IX: Fashion MNIST- Comparison of Accuracy with other existing approaches

The table. IX shows the accuracy obtained on Fashion MNIST data by various other network model along with our model. Here the author [23] used 2 layers of convolution with 32 filters followed by max pooling and the kernel size is 3 * 3 and acquired 91.17% accuracy. The author [20] attempted. evolutionary deep neural networks, i.e. the generated chromosomes are passed to a network of 10,100 filters of convolutional nodes and max-pooling and embedding of size 100,300. Finally, fully connected dense layer with the Adam optimizer to achieve 90.60% accuracy. The CNN-SoftMax [22] model utilizes the SoftMax activation and obtained 91.86% accuracy in the fully connected dense layer, while the CNN-SVM model [22] used the support vector machine as the classification and though this way the author accomplished 90.72 % accuracy. The table. IX shows the other existing convolutional models on fashion MNST data and the table.X shows the existing LSTM model summary on the fashion MNIST data. In [24], There are 4 hidden layers in model LSTMs while 64 hidden neurons are present in each layer and 28-time steps, 28 input neurons, and 10 output neurons in the LSTMs model. The model without the pruning of the network is chosen because my pruning algorithm does not reduce the computational cost and the accuracy reached 88.26% whereas the author [25] tried two-layer LSTM model with 35 memory blocks. He used Network Pruning with a threshold of 0.03 on it. His predictive accuracy on the test set after retraining is 89.94 %. The model (CNN1) which we tried is showing better results than other previously model and shows the accuracy of 92.02%.

In experiments we found that:

- CNN with 1 convolution layer and LSTM with 2 layer gives more than 90 percentage of test accuracy on Fashion MNIST Dataset and CIFAR-10 Dataset respectively.
- CNN3, LSTM1 and LSTM2 performed better than other models on the standard CIFAR-10 dataset.

Layers	S-CNN (64-128-256-512) [19]	S-CNN (92-256-512-1024) [19]	Evolutionary deep learning framework [20]
Convolution Layer1	64 filters Kernel Size:(5*5)	92 filters Kernel Size:(5*5)	1D and 2D convolution:[10, 100] 1,6 size
Max Pooling	Kernel Size:(3*3)	Kernel Size:(3*3)	1D and 2D max pooling: [1, 6]
Convolution Layer2	128 filters Kernel Size:(5*5)	256 filters Kernel Size:(5*5)	-
Max Pooling	Kernel Size:(3*3)	Kernel Size:(3*3)	-
Convolution Layer3	256 filters Kernel Size:(5*5)	512 filters Kernel Size:(5*5)	-
Fully Connected Layer	512 filters	1024 filters	[10, 100]
Embedded Layer	-	-	[100, 300]

TABLE VIII: CIFAR-10 - Other existing approaches Summary

Layers	CNN with Softmax [23]	CNN with SVM [23]	Cnn2 [22]	Evolutionary deep learning framework [20]
Convolution Layer1	CONV5: 5 5 size, 32 filters, 1 stride	CONV5: 5 5 size, 32 filters, 1 stride	Conv1:32 filters(3*3)	1D and 2D convolution:[10, 100] 1,6 size
Max Pooling	2 2 size	2 2 size	2 2 size	1D and 2D max pooling: [1, 6]
Convolution Layer2	5 5 size, 64 filters	5 5 size, 64 filters	Conv2:32 filters(3*3)	-
Max Pooling	2 2 size,	2 2 size,	2 2 size	-
Fully Connected Layer	1024 (SoftMax activation)	1024(SVM)	Dense layer	[10, 100]
Embedding Layer	-	-	-	[100, 300]

TABLE X: Fashion MNIST- other existing CNN approaches summary

layers	LSTM Without Network Pruning [24]	LSTM With Pruning [25]
Layer-1	LSTM-1, 64 filters	LSTM-1, 35 filters
Layer-2	LSTM-2, 64 filters	LSTM-2, 35 filters

TABLE XI: Fashion MNIST- other existing LSTM approaches summary

V. CONCLUSION

We have trained neural network based on deep learning architectures to classify images on conventional datasets such as MNIST fashion and CIFAR-10. Several network models have been tested to enhance the model's predictive accuracy. Our models yield convincing outcomes on the conventional benchmark dataset. In comparison with another model, the tested models such as CNN1 shows better accuracy on the fashion MNIST dataset and CNN3, LSTM1 and LSTM2 performed better than other models on the CIFAR-10 dataset. We demonstrate that results are consistent with earlier techniques suggested, while our findings use a simpler architecture and no data augmentation or use of external data. The model still needs further improvement, the accuracy of the models is indeed improved compared to other approaches. As stated, by tuning hyper-parameters and hybridizing CNN and RNN, the models still give a lot of hope for improvement.

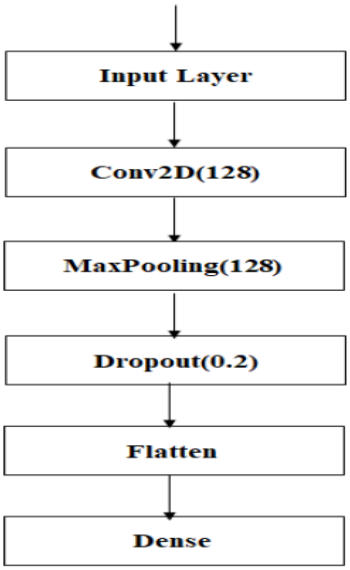

Fig. 16: CNN Layer1 Architecture

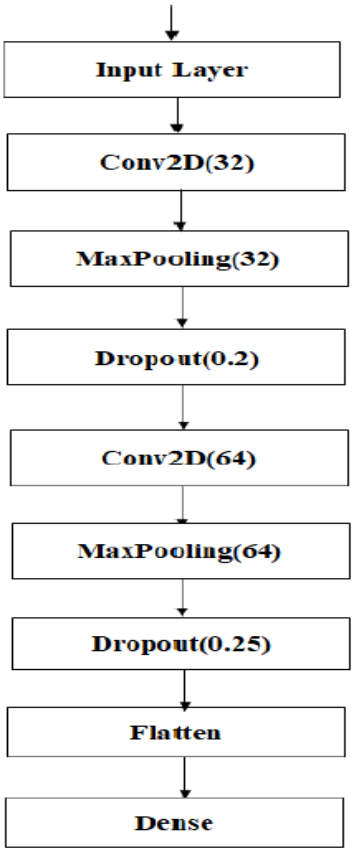

Fig. 17: CNN Layer2 Architecture

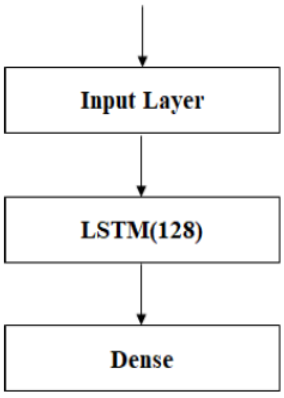

Fig. 18: LSTM1 Architecture

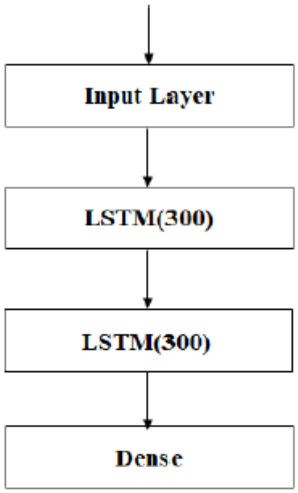

Fig. 19: LSTM2 Architecture

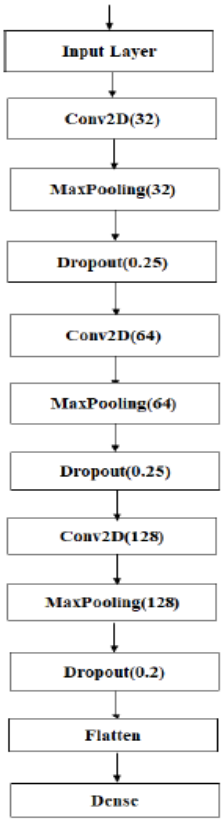

Fig. 20: CNN-3 Layer Architecture

REFERENCES

1. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface:Closing the gap to human-level performance in face verification," in Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1701–1708. [Online]. Available: https://doi.org/10.1109/CVPR.2014.220
2. A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in Business Informatics (CBI), 2017 IEEE 19th Conference on. IEEE, 7 2017, pp. 7–12.
3. A. Iliukovich-Strakovskaia, A. Dral, and E. Dral, "Using pre-trained models for fine-grained image classification in fashion field," in Proceedings of the First International Workshop on Fashion and KDD, KDD, 2016, pp. 31–40.
4. M. Abadi, A. Agarwal, et.al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
5. J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in science & engineering, vol. 9, no. 3, p. 90, 2007.
6. S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," Computing in Science & Engineering, vol. 13, no. 2, p. 22, 2011.
7. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in python," Journal of machine learning research, vol. 12, no. Oct, pp. 2825–2830, 2011.
8. https://en.wikipedia.org/wiki/Convolutional-neural- network:.
9. https://medium.com/@ksusorokina/image-classification- with convolutional- neural-networks-496815db12a8.
10. A. F. Agarap, "Deep learning using rectified linear units (relu)," arXiv preprint arXiv:1803.08375, 2018.
11. https://medium.com/@RaghavPrabhu/understanding-of-convolutiona lneural-network-cnn-deep-learning 99760835f148.
12. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.
13. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.
14. F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continuous prediction with lstm," Technical Report IDSIA-01-99, Tech. Rep., 2000.
15. S. Pal, S. Ghosh, and A. Nag, "Sentiment analysis in the light of lstm recurrent neural networks," International Journal of Synthetic Emotions (IJSE), vol. 9, no. 1, pp. 33–39, 2018.
16. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.
17. https://medium.com/@amarbudhiraja/https-medium-comamarbudhira ja- learning-less-to-learn-better-dropout-in-deep-machinelearning-74334da4bfc5.
18. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," CoRR, vol. abs/1207.0580, 2012,cite arxiv:1207.0580. [Online]. Available: http://arxiv.org/abs/1207.0580
19. A. Ghaderi and V. Athitsos, "Selective unsupervised feature learning with convolutional neural network (s-cnn)," in 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, 2016, pp. 2486–2490.
20. E. Dufourq and B. A. Bassett, "Eden: Evolutionary deep networks for efficient machine learning," in 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech). IEEE, 2017, pp. 110–115.
21. A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 215–223.

22. A. F. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," arXiv preprint arXiv:1712.03541, 2017.
23. S. Bhatnagar, D. Ghosal, and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," in 2017 Fourth International Conference on Image Information Processing (ICIIP). IEEE, 2017, pp. 1–6.
24. S. Shen, "Image classification of fashion-mnist dataset using long shortterm memory networks," in Proceedings of ABCs 2018 1st ANU Bioinspired Computing conference, no. 38, July 2018, pp. 1–7.
25. K. Zhang, "Lstm: An image classification model based on fashion-mnist dataset," in Proceedings of ABCs 2018 1st ANU Bio-inspired Computing conference, Aust. Nat. Uni, no. 92, July 2018, pp. 1–7.
26. H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.

AUTHORS PROFILE

N.Karthika did her undergraduate in B.E Major in Computer Science and Engineering from Anna University, Chennai. She completed postgraduate in M.E (Computer Science and Engineering) with distinction from Anna University, Trichy. Currently, she is a full-time research student at National Institute of Technology, Trichy. Her research interests are multimedia information Processing, information retrieval, deep learning.

B. Janet is an Assistant Professor in the Department of Computer Applications, NIT, Trichy for over 10 years. She has received honours that include University Rank, NET for Lectureship by UGC and deployed a honeypot sensor as part of National Cyber Coordination Centre Project for Cyber Threat Intelligence Generation. She has published 10 papers in International Journals and made 20 International Conference presentations. She has set up the first of its kind Information Processing and Security Laboratory with Industry involvement. She is a champion of open source technology. Her areas of specialization are Information Processing and Security, Internet of Things, Application Development and Deep Learning.

.

Himanshu shukla received his bachelor of technology from SRM University, NCR campus, india in 2016 and he is currently pursuing MTech in Data Analytics from NIT Trichy, Tamilnadu. His interested areas are machine learning and deep learning.