Analysis of Deep Learning Models using Convolution Neural Network Techniques

N.Durai Murugan, SP.Chokkalingam, Samir Brahim Belhaouari

ABSTRACT--- Deep Learning is the one of the souls of Artificial Intelligence and it is rapid growing in the medical data analysis research field, in many conditions Deep learning models are look like the neurons in brain, although both contain enormous number of computation Neurons units also called neurons that are not extremely intelligent in separation but improve optimistically when they interact with each other. The key objective is that many Convolution Neural Network models are available for image analysis which gives different accuracy in different aspects by training the model. A major analysis of Convolution models using Multilayer Perceptron is driven to analyses the image dataset of handwritten digits and to experiment by variations that are occurred in during the various changes that applied to the Convolution techniques like padding, stride and pooling to get best models in terms of the best accuracy and time optimization by minimizing the loss function.

Keywords: Convolution Neural Network, Multilayer Perceptron, Deep learning, Deep Neural Network, Activation Function

I. INTRODUCTION

Deep learning contains Convolution Neural Networks, Multilayer Perceptron that are shown on similar networks existing in the human brain. As data moves through this artificial net which forms multiple hidden layers based on the weights in each layer processes a feature of the data, cleans the outliers, locates the spots of familiar units, and produces the predicted output.

The Input layer contains neurons which receive the input sets and it does not do anything other than passing it to hidden layers. Each number of nodes in input layer should be equal with the attributes or features in dataset. The Hidden Layer is the defined with input layer by applying any activation function with corresponding weights of each input node, there will be number hidden layers based on the type of deep learning model. Hidden layers contain vast number of neurons (Fig.1); the neurons in the hidden layer apply conversions to the inputs before it passing them to next layer as the network is educated the weights get updated to be more predictive, it basically depends on the type of model we choose to build the CNN Model. Finally, the predicted feature is in the Output Layer Neuron Weights refer to the strength or amplitude of a connection between two neurons, linear regression is used to compare weights on inputs like coefficients in a regression equation. Weights are initialized to every input node with small value of 0 to 1 range, often randomly.

Feedforward is supervised neural networks among most popular learning algorithms, also called as Deep Networks and multi-layer Perceptron. Each Neuron is associated with other neuron with some weight in a single hidden layer. The network progressions of input layers would upward be activating neurons in each hidden layer and finally get the output value. This network is called a feedforward network.

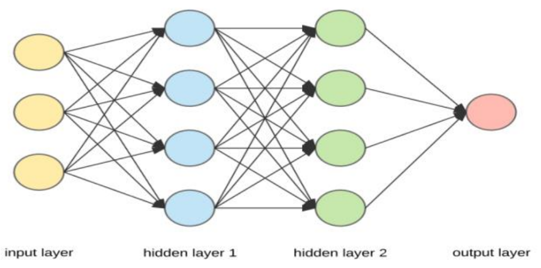


Fig.1 Deep Neural Network

Back Propagation: The predicted value with the DNN network is enforced to compared with expected output, if there is any variation it consider as error and is recalled and allows for bask propagation towards the entire network, one layer at a moment, by adjusting the weights using square measure in step updated with the value contributed with error, a little bit control of clever calculation is called the Back-Propagation algorithmic rule (Fig.2). This method is persistent to all of instances in the training data, in case the network has circular variation in the entire training dataset which would be named associate with each epoch and could be trained for several thousands of epochs.

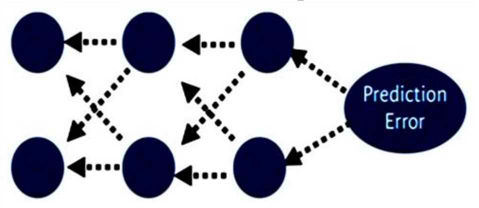


Fig.2: Backpropagation

Manuscript published on 28 February 2019.
* Correspondence Author (s)

N.Durai Murugan, Research Scholar, Department of CSE, Saveetha Institute of Medical and Technical Sciences Assistant Professor, Department of CSE, Rajalakshmi Engineering College, Chennai, Tamil Nadu, India (durains@gmail.com)
SP.Chokkalingam, Professor, Department of CSE, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, Tamil Nadu, India (cho_mas@yahoo.com)
Samir Brahim Belhaouari, Associate professor, College of Science and Engineering, Hamad Bin Khalifa University, Qatar, College of Science, Alfaisal University (sbelhaouari@hbku.edu.qa)

Convolution in Deep Learning

Convolution is a most important technique in deep learning it is a pointwise mathematical operation like addition, multiplication of two functions to produce a third function, or a derivative, and this is complex operation, while it can be very useful to simplify even more complex equations. Convolutional nets that catapulted deep learning to the front of almost any machine learning task. The convolution in CNN is used for feature extraction from the input image which is basic purpose of convolution, after a short mathematical development of convolution, it will relate and integrate designs between these fields of science and deep learning. While the ANNs suffer from curse of dimensionality when it comes to high resolution images, CNNs do a little pre-processing, that means the network learns the filters before doing the real classification. it uses filters for corresponding fields for identifying the spatial locality action by imposing a pattern connectivity among the neurons with layers adjacent to the local node. In the case of image processing, it is the multiplying process of all element in matrix with its near neighbors, weighted or biased by the kernel (or filter). For example, given a matrix A and kernel c, the discrete convolution operation *conv(A,c)* is defined.

$$A = \begin{pmatrix} a_{00}\ a_{01}\ a_{02}\ a_{03}\ a_{04} \\ a_{10}\ a_{11}\ a_{12}\ a_{13}\ a_{14} \\ a_{20}\ a_{21}\ a_{22}\ a_{23}\ a_{24} \\ a_{30}\ a_{31}\ a_{32}\ a_{33}\ a_{34} \\ a_{40}\ a_{41}\ a_{42}\ a_{43}\ a_{44} \end{pmatrix} \text{ and } c = \begin{pmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{pmatrix}$$

convolution operation *conv(A,c) is,*

conv(A,c)[1,1] = $a_{11}*c_{00} + a_{12}*c_{01} + a_{13}*c_{02} + a_{21}*c_{10} + a_{22}*c_{11} + a_{23}*c_{12} + a_{31}*c_{20} + a_{32}*c_{21} + a_{33}*c_{22}$

While applying convolution for two dimensions of image namely height and width will be applied to the images. First the input image has three matrices of pixels for RGB color channels, each color channels contains a value as integer in the ranges from 0 to 255. Second the convolution kernel or filter matrix as floating-point numbers with the pattern how to intertwine input image with convolution kernel operation after convolution it has the altered image matrix as output of the kernel, which if formed by doing sliding operation with the kernel to the image by doing the dot product calculating is called the 'Convolved Feature' for each color channel, which is also called as Activation Map. There are different Filter Weights are applied while convolution neural networks are automatically estimating the weighs of the filter, some of the filters are shown in Fig.3., detecting vertical edges is shown in Fig.4.

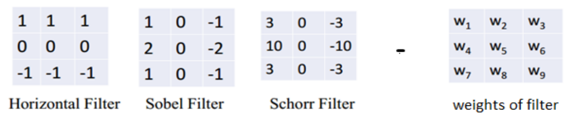


Fig.3: Filters

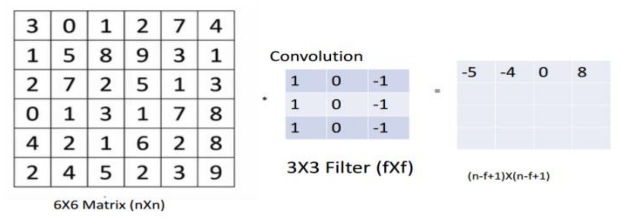


Fig.4: Detecting Vertical Edges

Padding and Stride:

Padding is used to preserve the original dimensions of the input and will add Zeros to outside of the input, number of zero layers depend upon the size of the kernel or filter as shown in Fig.5., next layer will be formed by doing padding p, filter f, input matrix with padding nXn. Without padding means that, the data size will get reduced for the next layer. Whereas with the introduction of adequate padding will keep the size intact.

Stride is how long the convolutional kernel or filter jumps when it looks at the next set of data. It reduces the size of the next layer, and lets you decide how much overlap you want between two output values in a layer, stride of 3, while still looking at that 5x5 input. Besides, with more strides means that we are limiting the overlap of two subsequent dot products in the convolution operation. This means that each output value in the activation will be more independent of the neighbouring values.

Stride(s=3) $$\boldsymbol{Floor}\left(\frac{n+2p-f}{s}+1\right)\ \boldsymbol{X}\ \boldsymbol{Floor}\left(\frac{n+2p-f}{s}+1\right)$$

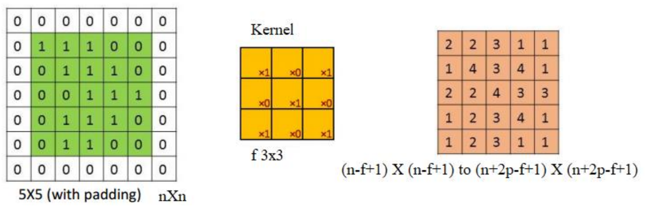


Fig.5: Padding

Pooling

Pooling is one of most important technique in CNN, the input map is divided into a rectangle set of size depends on the convolved feature and outputs the maximum for nonlinear down-sampling. Role of pooling layer is to reduce the spatial size of the image progressively and retains features of the convolved feature but it is required rotational invariants and translational for classification. Backpropagation is used for training of pooling operation, while here two pooling methods max pooling and average pooing were done by converting 4x4 to 2x2 matrix with filter size 2 and stride size 2 as shown in Fig.6.

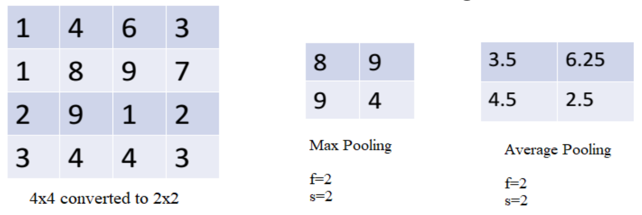


Fig.6: Pooling

II. ACTIVATION FUNCTION

It is most important in CNN where it maps the summation of weights of corresponding input neuron to produce the hidden layer as output. The nodes in every hidden layer is activated by a linear function to produce next hidden layer, based on another linear activation function, since it directs the neuron when it starts to activated output strengths. The activation function is the nonlinear transformation that we do over the input signal. This transformed output is then sent to the next layer of neurons as input.

$$\mathbf{Y} = \sum(\mathbf{weight} * \mathbf{input}) + \mathbf{bias}$$

ReLu

Many Deep learning networks uses the Rectified Linear Activation (ReLUs). In this activation, if input is less than zero then output is zero or it would be raw output. output will be equal to input, if input is greater than zero. It is like a real human body neuron, these are the simplest non-linear activation in use. When positive input is received, the derivative is one, so there isn't the embracing outcome on backpropagated errors.

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{max}(\boldsymbol{x}, \mathbf{0})$$

Softmax

The calculation of probabilities distribution number of different events is called softmax, it classifies in multiple categories, if it required to categorize pictures, this calculates each target class over by probabilities of all target classes like scene, Animals, Humans, Vehicles etc., then in that case it will have more outputs from the softmax function which will give the probabilities of each of these categories. Sum of probabilities will be one and that with the highest probability as shown as below, where the inputs vector as z, the output indexes as j, where, j = 1, 2, ..., k.

$$\boldsymbol{A}(\boldsymbol{z})_{\boldsymbol{j}} = \frac{\boldsymbol{e}^{\boldsymbol{z_j}}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{K}} \boldsymbol{e}^{\boldsymbol{z_k}}}$$

From the above discussions, activation function performs the necessary feature that it ought to be differentiable. We'd like to use this on performing backpropagation for improvement strategy whereas propagating backwards within the network to reason gradients of Error (loss) with relevancy Weights then consequently optimize weights exploitation Gradient descend or the other improvement technique to cutoff the Error.

III. RESULTS & DICUSSIONS

The cost function is that a neural network given for training input and therefore the expected output. It might rely on attributes like weights and biases. The cost function is single-valued and not a vector as a result of it rates however well the neural network accomplished as a full. An optimized Gradient Descent algorithm is used for weights square measure updated incrementally when every period.
Sum of squared errors:

$$\boldsymbol{J(w)} = \frac{\mathbf{1}}{\mathbf{2}}\sum_{i}((\boldsymbol{target})^{\boldsymbol{i}} - (\boldsymbol{output})^{\boldsymbol{i}})^{\mathbf{2}}$$

by just a step to opposite direction of the cost gradient, it will calculate magnitude as well as direction of the weight update.

$$\Delta \boldsymbol{w_j} = -\boldsymbol{\eta}\frac{\boldsymbol{\delta J}}{\boldsymbol{\delta w_j}}$$

Where, η is the learning rate, the updated weight of vector w of all weight coefficient of w,

$$\Delta \boldsymbol{w_j} = \frac{\mathbf{1}}{\mathbf{2}}\sum_{i}((\boldsymbol{target})^{\boldsymbol{i}} - (\boldsymbol{output})^{\boldsymbol{i}})\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})}$$

Calculate the gradient descent until the derivative reaches the minimum error, by sharpness of gradient slope.

Analysis and Result

Handwritten digits images dataset is been used, and is set for basic Preparation in which the bath size is of 128 and the number of classes is 10 to identifying the numbers zero to nine in the dataset, the image dimensions used here are 28x28. The dataset is consisting of about sixty thousand images in which it has been shuffled and categorized into two samples as train samples and test samples (10% from dataset), once the dataset is loaded reshaping is done for both the train data and test data based on the channels in the image data format with respect to image rows and image column of respective train shape and test shape and then the reshaped image in taken as the input shape and float32 would be the type of train input and test input and both train and test data are divided by the 255 pixels with one epochs, the input train shape and input test shape ware convert class vectors to binary class matrices.

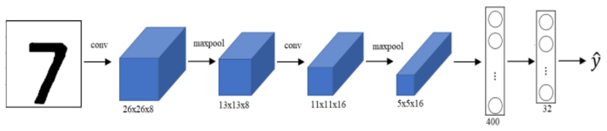


Fig.7: Model Diagram

The model is build based on the input shape, here two-dimensional convolution neural network of height 26 and width 26 is been used and implemented in a sequential order for multiple layers which has been convoluted with kernel size of 3x3 and used ReLU activation for first hidden layer then the output filters size is 8 and the total parameter obtained is 80; after first convolution pooling is done here, max pooling with size 2x2 is applied, next layer is been used with kernel size 3x3 and activation ReLU with 16 number of output filters in the convolution and applied a max pooing of size 2x2 with the dropout of 0.25. the model shown in Fig.7.

Layer	Output	Param
Convolution 2D	26, 26, 8	80
Max Pooling	13, 13, 8	0
Convolution 2D	11, 11, 16	1168
Max Pooling	5, 5, 16	0

Dropout	5, 5, 16	0
Flatten	400	0
Dense	32	12832
Dropout	32	0
Dense	10	330

Fig.8.Summary of model

Literally going to flatten the pooled feature map into a column it is needed to insert this data into an artificial neural network later on, a linear activation can be applied but, generally a non-linear activation is followed with dense of size 32 is applied to the previous layer to get the Fully connected layer with which the activation applied is ReLU with dropout of about 0.5 for getting the next final output layer is been implemented with dense of number of classes as 10 (zero to nine) with activation function softmax, the summary of the model is shown in Fig.8.

The total trainable paraments for the model is about 14,410 whereas for non-trainable parameter is nil. The trained model compiled for optimizer, metrics and losses by applying the categorical cross entropy for losses and Adadelta optimizer where it fit for the batch size and the verbose as 1 by validated with the test data. The Test loss is about 13.11%, Test accuracy about 96.13%, total time of about 21 sec. and the prediction score for the number of classes (10) defined in softmax is 9.96 and also the threshold score is denoted which is calculated by setting the value prediction>0.5 * 1, the predicted image of size 28x28 is shown in Fig.9.

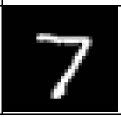
Parameter	**Value**
Total	14,410
Trainable	14,410
Non-trainable	nil
Loss	74.33%
Accuracy	75.01%
Optimize	21 sec
Test loss	13.11%
Test accuracy	96.13%
Prediction Score	9.9689817e-01
Threshold Score	[0 0 0 0 0 0 **1** 0 0 0]
Predicted Digit	7
Predicted Image	

Fig.9.Model Evaluation

DEEP CONVOLUTION LAYER VISUALIZATION

The model Building is done for Deep Convolution Layer Visualization by doing the reshaping of the predicted image from the model generated with the placeholder of both dimensions of about 784x10 of type float and reshape size of about 28x28 and is applied with slim convolution of size 5x5 by pooling at 2x2 and connected by slim flatten with the softmax activation function then the model is trained with batch size of 50 under iteration of each steps the raining accuracy is shown in Fig.10., and the test accuracy is about 90. The activation values and the plotting in each hidden layer are denoted and the filtered images at each layer are shown in Fig.11.

At Step	Training Accuracy
100	0.42
200	0.7
300	0.82
400	0.88
500	0.82
600	1
700	0.92
800	0.86
900,	0.78
1000,	0.88

Fig.10. Training Accuracy

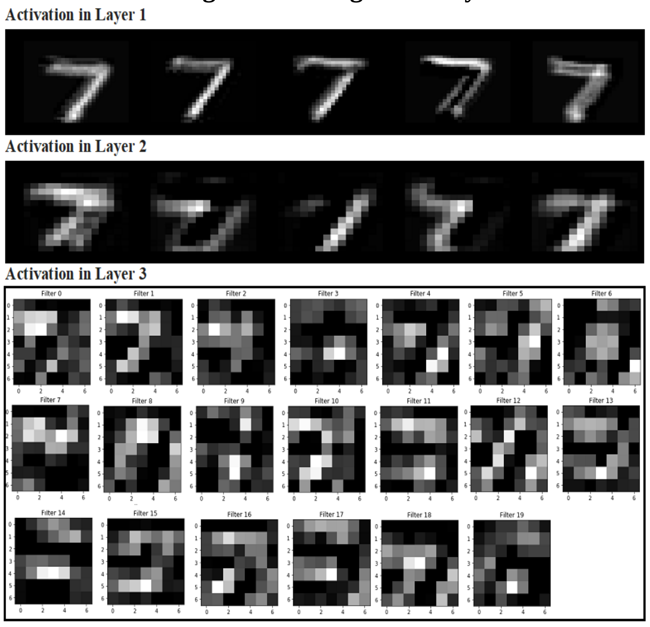


Fig.11. Filter image in Activation Layer

MODEL ANALYSIS BASED ON KERNEL SIZE

Influence of convolution size in Convolutional Neural Networks by modifying the model is based on the input shape, here two-dimensional convolution neural network of height 26 and width 26 is been used and implemented in a sequential order for multiple layers which has been convoluted with kernel size of 3x3 and the activation function ReLU for first hidden layer and output kernel of size 8 and the total parameter obtained is 80, next layer is been used with kernel size 3x3 and activation ReLU with 16 number of output filters in the convolution with the parameter 1168 and the flatten and dense 1 & 2 are applied. The accuracy while using kernel size 3x3 convolution is 96% and the Time Taken to run the model is 185.29 seconds.

When the same model is convoluted with kernel size 7x7 and the activation function ReLU for first hidden layer and output kernel of size 8 and the total parameter obtained is 400, next layer is been used with kernel size 7x7 and activation ReLU with 16 number of output filters in the convolution with the parameter 6288 and the flatten and dense 1 & 2 are applied. The accuracy while using kernel size 3x3 convolution is 97% and the Time Taken to run the model is: 77.30 seconds shown in Fig 12. In this analysis it is infer that while using 7x7 kernel size the test accuracy in increased and also the time taken to run the model is minimized to 50% than while using 3x3 kernel size.

Layer (type)	**Convolution by (3 x 3)**		**Convolution by (7 x 7)**	
	Output Shape	**Parameter**	**Output Shape**	**Parameter**
Convolution 2D	26, 26, 8	80	22, 22, 8	400
Convolution 2D	24, 24, 16	1168	16, 16, 16	6288
Flatten	9216	0	4096	0
Dense	32	294944	32	131104
Dense	10	330	10	330
Total parameter	296,522		138,122	
Loss	11.37%		6.55%	
Accuracy	96.66%		97.76%	
Time Taken	185.29 seconds		77.29	

Fig 12. Model Analysis based on kernel Size

Model Analysis based on Strides, Padding and Pooling

When model 7x7 convolution ReLU activation function by moving strides to 2 steps of with the number of output filters in the convolution is 8 where the total parameter obtained is 400, while next layer is been used with kernel size 7x7 and activation ReLU by moving stride to 2 steps with 16 number of output filters in the convolution with the parameter 6288 and the flatten and dense 1 & 2 are applied. The accuracy by applying stride 2 is 94% and the Time Taken to run the model is: 11.77 seconds, while applying padding to same and reduce the stride to 1 step, the accuracy by applying stride 1 and padding to same is 97% and the Time Taken to run the model is: 200.20 seconds. When pooling 3x3 is applied for the kernel size 3x3 convolution the accuracy is 95% and the time taken to run the model is 17.59 seconds, the model summary is shown in Fig.13

Layer	**Stride 2**		**Padding (same)**		**Pooling (3,3)**	
	O/p Shape	**Param**	**O/p Shape**	**Param**	**O/p Shape**	**Param**
Convolution 2D	11, 11, 8	400	28, 28,8	400	26, 26, 8	80
Max Pooling	-	-	-	-	8, 8, 8	0
Convolution 2D	3, 3, 16	6288	28,28,16	6288	6, 6, 16	1168
Max Pooling	-	-	-	-	3, 3, 16	0
Flatten	144	0	12544	0	144	0
Dense	32	4640	32	401440	32	4640
Dense	10	330	10	330	10	330
Total parameter	11,658		408,458		6,218	
Loss	20.30%		7.82%		12.92%	
Accuracy	94.06%		97.57%		95.98%	
Time Taken	11.768 seconds		200.20 seconds		17.59 seconds	

Fig 13. Model Analysis based on stride and padding

IV. CONCLUSION

The deep learning model using CNN and MLP with various Activation function applied for the input and hidden layer where the analysis is based on the convolution with the kernel size and the increase in stride, padding and the pooling size it infers that when stride size is increased the accuracy is maximum and time taken to run the model is very less whereas when the same model is applied with padding the accuracy is maximum but the time taken is also maximum but while pooling is applied the time is less and the accuracy is somehow more than when compared to stride. Although Deep learning models contain enormous number of computations, it is inferring that when the kernel size is 7x7 with the same padding and allow 2 stride steps the accuracy level increase, and depends up on the application and the user requirement that are not extremely intelligent in separation by Appling the different models to improving the accuracy and time optimization.

ACKNOWLEDGEMENT

This research was supported by King Abdulaziz City of Science and Technology (KACST), so we would like to express our gratitude and appreciation to King Abdulaziz City of Science and Technology (KACST) for providing the research grant, **Project Number 367-34**

REFERENCES

1. Sze, Vivienne; Chen, Yu-Hsin; Yang, Tien-Ju; Emer, Joel (2017). "Efficient Processing of Deep Neural Networks: A Tutorial and Survey"
2. Mellars, Paul (February 1, 2005). "The Impossible Coincidence: A Single-Species Model for the Origins of Modern Human Behavior in Europe"
3. Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117.
4. Marcus, Gary (November 25, 2012). "Is "Deep Learning" a Revolution in Artificial Intelligence?". The New Yorker. Retrieved 2017-06-14.
5. Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images." (2014).
6. Deng, L.; Yu, D. (2014). "Deep Learning: Methods and Applications" (PDF). Foundations and Trends in Signal Processing. 7 (3–4): 1–199.
7. Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, “Greedy Layer-Wise Training of Deep Networks”, in J. Platt et al. Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007
8. Marc’Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun “Efficient Learning of Sparse Representations with an Energy-Based Model”, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007
9. Li Deng, Xiaodong He, Jianfeng Gao, Deep stacking networks for information retrieval , Acoustics, speech and signal processing, 2013 IEEE International conference
10. Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik “Multi-Scale Orderless Pooling of Deep Convolutional Activation Features”University of North Carolina at Chapel Hill
11. Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun (2015), “Deep Residual Learning for Image Recognition” Microsoft Research
12. Manuel Fern´andez-Delgado, Eva Cernadas, Sen´en Barro (2014) “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?”, Journal of Machine Learning Research
13. A. Graves, Supervised sequence labelling with recurrent neural networks, vol. 385, Springer, 2012
14. J. Islam and Y. Zhang, Visual Sentiment Analysis for Social Images Using Transfer Learning Approach, 2016 IEEE Int. Conf. Big Data Cloud Comput. (BDCloud), Soc. Comput. Netw. (SocialCom), Sustain. Comput. Commun., pp. 124130, 2016.
15. A. Severyn and A. Moschitti, Twitter Sentiment Analysis with Deep Convolutional Neural Networks, Proc. 38th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR 15, pp. 959962, 2015.
16. L. Yanmei and C. Yuda, Research on Chinese Micro-Blog Sentiment Analysis Based on Deep Learning, 2015 8th Int. Symp. Comput. Intell. Des., pp. 358361, 2015.
17. Q. You, J. Luo, H. Jin, and J. Yang, Joint Visual-Textual Sentiment Analysis with Deep Neural Networks, Acm Mm, pp. 10711074, 2015.
18. X. Ouyang, P. Zhou, C. H. Li, and L. Liu, Sentiment Analysis Using Convolutional Neural Network, Comput. Inf. Technol. Ubiquitous Comput. Commun. Dependable, Auton. Secur. Comput. Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM), 2015 IEEE Int. Conf., pp. 23592364, 2015.
19. C. Li, B. Xu, G. Wu, S. He, G. Tian, and H. Hao, Recursive deep learning for sentiment analysis over social data, Proc. - 2014 IEEE/WIC/ACM Int. Jt. Conf. Web Intell. Intell. Agent Technol. - Work. WI IAT 2014, vol. 2, pp. 13881429, 2014.
20. R. Socher, A. Perelygin, and J. Wu, Recursive deep models for semantic compositionality over a sentiment treebank, Proc., pp. 16311642, 2013.
21. W. Li and H. Chen, identifying top sellers in underground economy using deep learning-based sentiment analysis, Proc. - 2014 IEEE Jt. Intell. Secur. Informatics Conf. JISIC 2014, pp. 6467, 2014.
22. C. Baecchi, T. Uricchio, M. Bertini, and A. Del Bimbo, A multimodal feature learning approach for sentiment analysis of social network multimedia, Multimed. Tools Appl., vol. 75, no. 5, pp. 25072525, 2016.
23. H. Yanagimoto, M. Shimada, and A. Yoshimura, Document similarity estimation for sentiment analysis using neural network, 2013 IEEE/ACIS 12th Int. Conf. Comput. Inf. Sci., pp. 105110, 2013.
24. R. Silhavy, R. Senkerik, Z. K. Oplatkova, P. Silhavy, and Z. Prokopova, Artificial intelligence perspectives in intelligent systems: Proceedings of the 5th computer science on-line conference 2016 (CSOC2016), vol 1, Adv. Intell. Syst. Comput., vol. 464, pp. 249261, 2016.
25. A. Hassan, M. R. Amin, A. Kalam, A. Azad, and N. Mohammed, Bangla Text (BRBT) using Deep Recurrent models.
26. T. Chen, R. Xu, Y. He, Y. Xia, and X. Wang, Using a Sequence Model for Sentiment Analysis, no. August, pp. 3444, 2016.