A Non-iterative Radial Basis Function Based Quick Convolutional Neural Network

Toshi Sinha
Centre for Intelligent Systems
School of Engineering and Technology
Central Queensland University
Australia
t.sinha@cqu.edu.au

Brijesh Verma
Centre for Intelligent Systems
School of Engineering and Technology
Central Queensland University
Australia
b.verma@cqu.edu.au

Abstract— **In the past few years, Convolutional Neural Networks (CNNs) have achieved surprisingly good results for objects classification in real world images. However, training a CNN from scratch for large datasets is still a nightmare, when it comes to time and resources. The main reason for this problem is long iterative training process used in CNN's fully connected layer which is also called a classification layer. Therefore, in this paper we propose a novel approach to make the convolutional neural network quicker and more efficient for image classification tasks. The proposed approach consists of a convolutional feature extraction layer and a non-iterative radial basis function-based classification layer. The proposed approach has been evaluated on three benchmark datasets such as CIFAR-10, MNIST and Digit. The experimental results have demonstrated that the proposed approach can achieve same or higher accuracy in lesser time than the standard CNN.**

Index Terms—**Convolutional Neural Network, Feature Extraction, Radial Basis Function, Image Classification**

I. INTRODUCTION

Convolutional neural network is a deep neural network that mimics human's visual cortex. In the past few years, CNN based models have been widely used to solve various computer vision problems including complex image recognition, object detection and optical character recognition [1] [2]. Convolutional neural networks have obtained substantial attention in machine learning problems. Due to strong image recognition capabilities, CNNs are used across variety of applications including image classification [3], video recognition, natural language processing [4], disease categorization and identification [5].

CNN was mainly introduced to improve the segmentation and classification in image datasets. It models the representation of animal's visual cortex, when it comes to response to an image, each part of the convolutional layer tries to learn some features of the image. It automatically learns relational mapping between inputs and outputs [6]. The CNN differs to other network topologies by the connections between its layers, neurons in the next layer will only be connected to a small region of previous layer called as receptive field. The convolutional neural networks mainly consist of three types of layers: convolutional layers, pooling layers and fully connected layers.

On broader spectrum, CNN can be easily divided into two parts. The first part, where a set of discriminating features at multiple levels of abstraction are extracted using convolutional and pooling layers [2]. And the second part as classifier, where all the features extracted from first part, are fed into one or multiple fully connected layers and the classification is done.

The input image is resized accordingly to a square image and passed to convolution layer where it is convolved. To maintain non-linearity an activation layer is added to the architecture. Since Rectifier Linear Unit (ReLu) is faster to train, ReLu is used as activation function in each building block [7].

The convolutional layer with ReLu is usually followed by a pooling layer. The pooling layer is used to reduce the dimensions of the data by combining the outputs of neurons clustered at one layer to a single neuron in next layer. One or more fully connected layers are used at the end of the CNN architecture. The extracted features are weighted and combined in the fully connected layer to perform the classification task. In most of the CNN architectures multinomial logistic regression, known as Softmax activation function is used for class prediction and to minimize cross-entropy loss.

Although CNNs are very powerful, the performance of CNN depends upon the network depth (deeper the network better the results) [8] and large training data [9], which needs a long training time and resources. The solution to this problem was to use previously trained networks on a large dataset and only retrain the classification part on new dataset [10] [11] [4] . Still it takes considerably longer training time. A novel approach is proposed in this paper to make the convolutional neural network quicker and more efficient for an image classification task.

The aim of this paper is to introduce a new radial basis function-based layer in CNN which is a non-iterative layer. This makes a new type of CNN which is partially non-iterative making it much faster than the traditional CNN. The rest of the paper is organized as follows. Section II reviews related literature. Section III describes the proposed approach. Section IV presents experimental results and comparative analysis. A conclusion is drawn in section V.

II. RELATED WORK

In past few years, a lot of research has been conducted on reducing the time and resources taken by the CNN to get it trained. Due to robust hardware and software (multiple GPUs) the training time has reduced, and the accuracy has improved [1]. However, the usages of powerful hardware and plenty of time to train the model on large datasets cannot be overlooked. To solve this problem, researchers suggested to use pre-trained networks like "AlexNet"[1], "LeNet"[12], "Resnet"[15], etc., where models are already trained on large datasets. And only retrain the classification part with own/new training dataset [6] [16]. The feature extraction block – i.e. the learned convolution kernels of the previously trained network were retained and the classification block - i.e. the fully connected layers were retrained on different datasets using softmax function [16]. Therefore, it was found that the learned feature extractor from pre-trained network is applicable to multiple situations. Even though the feature extractor was trained on ILSVRC-12 dataset, containing natural images and scenes, it still achieved excellent results with digits from the MNIST dataset. This finding indicates that further datasets can be classified with the same feature extractor. This approach reduced the training time by ten folds and gave comparable accuracy on MNIST, CIFAR-10 and CIFAR-100 datasets. This has helped substantially but still the pretraining of network on large datasets like ILSVRC-12 is expensive.

Neural networks including CNNs, use iterative learning to train the models. During training, in each iteration the model processes the training dataset batch, calculates the cost function and using backpropagation calculates the gradient associated with each parameter weighting [17]. In iterative learning after each iteration the algorithm's parameters get updated. Gradient descent tries to minimize the cost function of a model, so that the trained model can give the most accurate results [12]. This type of iterative learning takes place in the classification block i.e. fully connected layer of the network. The number of parameters in the fully connected layer are huge and hence it takes a lot of time and resources to train such models using iterative learning. In deeper architectures where there are more fully connected layers, the problem becomes even more complex due to a greater number of parameters [8] [15]. Non-iterative learning [18] [19] [25] was an alternative to solve this problem.

Further it was well understood that the true power of neural network lies in their capability of extracting powerful, invariant and complex features [18] from different forms of data (images and numeric data). Better accuracy was achieved by combining these features derived from neural networks with alternate classification models like Support Vector Machines (SVM) [25] or K-Nearest Neighbor [18] [19]. In many cases, the features extracted using the CNN significantly improved the capabilities of SVMs and KNNs compared to running these algorithms on the raw features, and in some cases also surpass the performance of the neural network alone. Given a good feature vector, Gaussian-kernel SVMs are good at producing decision surfaces i.e. try to find the maximum margin between data points of different classes. Unlike the hinge loss of a standard SVM (L1-SVM), the loss for the L2-SVM is differentiable and penalizes errors much heavily showed significant gain on popular deep learning datasets MNIST, CIFAR-10 and ICML 2013 Representation Learning Workshop's face expression recognition challenge [20] [21].

In majority of cases, for transfer learning, features are extracted from fully connected layer of CNN [22] [23] [24]. These features suffer from lack of description of local patterns [24], which is critical when occlusion or truncation exists. On other hand if intermediate features are used, results of He et al. [22] on the Caltech-101 dataset [23] suggest that the Conv5 feature is better if Spatial Pyramid Pooling (SPP) is used and is inferior to FC6 features if no pooling step is there. Whereas, when translation and occlusion are intense, it is recommended to extract features from 'conv5 + avg/max pooling' layer than from FC6 and FC7 layers. Using average/max pooling on activation maps of intermediate CNN features improve recognition performance over raw features. Further combining features from different CNN layers improves the object recognition accuracy. Also, it is beneficial to use large size images instead of resized ones as a feed for CNN.

Few researchers have combined CNN with Random Vector Functional Link (RVFL) and proposed Convolutional Random Vector Functional link network (CRVFL) [27] to solve the visual tracking problem. In this architecture convolutional filters are initialized randomly and kept same, only the parameters of fully connected layers are learned. Hence the system does not need global fine tuning and not sensitive to hyper-parameters however the manual selection of parameter in RVFL deteriorate the performance in real world applications.

A hybrid deep architecture for traffic sign recognition has been presented which uses CNN for feature extraction and Extreme Learning Machine (ELM) as classifier [28]. Similar concept has been applied to handwritten digit recognition [29] [30], lane detection task [31] and 3D feature learning [32]. These hybrid architectures have low model complexity and better performance.

III. Proposed approach

This section describes our proposed approach which is called non-iterative radial basis function based Quick Convolutional Neural Network (Q-CNN). In Section III-A, the architecture of proposed Q-CNN is described in detail, and the network output is derived. In Section III-B and Section III-C the algorithms used for training and testing of the proposed model is presented.

A. Proposed Architecture

The proposed architecture of the quick convolutional neural network is illustrated in Fig. 1.

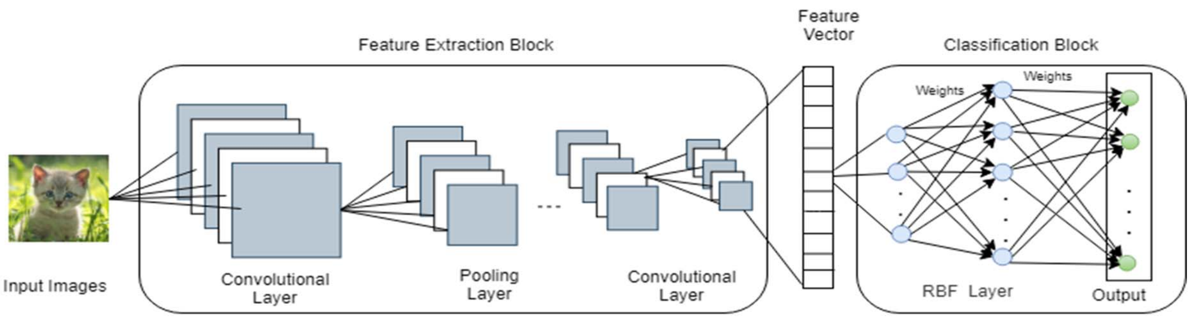

Fig. 1. Architecture of Proposed Q-CNN

The proposed radial basis function-based layer replaces the fully connected layer of classification block. The approach takes images as an input to the regular feature extraction block and produces feature vector with automatically extracted feature values. This feature vector is passed to non-iterative Radial Basis Function (RBF) layer for classification. The proposed architecture consists of input, convolutional layer, pooling layer, radial basis function layer and the output.

In Q-CNN the input image is resized to a square image and passed to convolution layer where it is convolved with multiple learned kernels using shared weights. Based on number of filters, the feature maps are created. To maintain non-linearity an activation layer, ReLu is added to the architecture. After a convolution layer, a pooling layer is used for down-sampling. Images are passed through convolutional and pooling layers to learn all the discriminative features at different level of abstraction. Further these learnt features are passed to a non-iterative radial basis function layer for training to learn the classification. This makes the proposed architecture comparatively faster than the traditional or standard CNN. The output from output layer of traditional CNN is estimated probabilities of input image and each probability is calculated using activation function. The input to activation function is linear combination of outputs from previous hidden layer with trainable weights plus bias. So, the outputs from previous layers can be considered as features and can be easily passed to RBF layer.

In this approach the features are extracted from the last pooling layer of the network in the form of feature vector and are passed to a non-iterative RBF layer, for further training and classification of input images. The convolutional and pooling layers are used as a feature extractor and they convert the input image into a feature vector represented as $\boldsymbol{F}_{[\boldsymbol{x_n} * \boldsymbol{z_0}]}$, where $\boldsymbol{x_n}$ is row of matrix $\boldsymbol{F}$ that represents the number of training records and $\mathbf{z_0}$ is column of matrix **F** and represents the number of features.

Feature vector $\mathbf{F}_{[\mathbf{x_n} * \mathbf{z_0}]}$ is passed to the input of RBF layer. This input can be modeled as a vector of real numbers and can be represented as

$$F_{[x_n * z_0]} \in \mathbb{R}^n \tag{1}$$

All the inputs are connected to hidden layer of the network. The hidden layer $\boldsymbol{\phi(F)}$ has a non-linear RBF activation function, which is a Gaussian function [17] and can be presented as follows:

$$\phi(F) = \rho(\|F - c_i\|) \tag{2}$$

$$\rho(\|x - c_i\|) = \exp[-\beta \| x - c_i \|^2] \tag{3}$$

Where $\boldsymbol{c_i}$ is the centre vector for neuron $\boldsymbol{i}$ and $\boldsymbol{F}$ is the input feature vector.

The Gaussian basis functions are local to the centre vector in the sense that changing parameters of one neuron has only a small effect for input values that are far away from the centre of that neuron

$$\lim_{\|x\| \to \infty} \beta(\| x - c_i \|) = 0 \tag{4}$$

The output of the hidden layer is a linear output. This can be stated as, the output of the network is a scalar function of the input vector [33], $\boldsymbol{y}$**:** $\mathbb{R}^n \rightarrow \mathbb{R}$, and is given by

$$y(F) = \sum_{i=1}^{N} w_i \, \rho(\|F - c_i\|) \tag{5}$$

Where $\boldsymbol{N}$ is the number of neurons in the hidden layer, $\boldsymbol{c_i}$ is the centre vector for neuron $\boldsymbol{i}$ and $\boldsymbol{w_i}$ is the weight of neuron $\boldsymbol{i}$ in the linear output neuron. Using equation (2) and equation (5), output of the network can be written as

$$y(F) = \sum_{i=1}^{N} w_i \, \phi(F) \tag{6}$$

The weights for RBF layer are calculated using non-iterative methods.

B. Proposed Algorithm

Algorithm for training the full architecture of Q-CNN is shown in Fig. 2 and the algorithm for testing with different datasets is shown is Fig. 3.

Algorithm: Q-CNN Training

Input: Training data **X**= $\{x_1\, y_1 \ldots x_n\, y_n\}$ and **T** = {t₁, …tₙ} where n is number of training samples
Output: Trained Q-CNN

```
1. Pass training data (X, T) to Convolutional and
   Pooling Layers
2. Extract features F using convolutional layer and
   pooling layer
       $F[i,j] = f_{pool}(f_{conv}(X))$        (7)
   Where F is the matrix with feature values, with i as
   row representing number of training samples and j as
   column representing the number of features.
   $f_{conv}$ is the convolution function which takes
   training data X, convolves it and creates the feature
   maps and $f_{pool}$ is the pooling function which reduces
   the dimension of feature maps.
3. Pass the feature matrix F and target matrix T to RBF
   layer
4. Generate centres ($c_i$) from training samples
5. Calculate weights of RBF layer as follows:
   Using equations (3) and (5) the RBF layer can be
   stated as follows
       $T = \sum_{i=1}^{N} w_i * exp(-\beta\, ||F - c_i||^2)$        (8)
   This implies:
       $\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \begin{bmatrix} e^{(-\beta||F1-c1||^2)} & \ldots & e^{(-\beta||F1-cn||^2)} \\ e^{(-\beta||F2-c1||^2)} & \ldots & e^{(-\beta||F2-cn||^2)} \\ \vdots & & \vdots \\ e^{(-\beta||Fn-c1||^2)} & \ldots & e^{(-\beta||Fn-cn||^2)} \end{bmatrix}$
   Using above equations, the weights of the RBF layer
   are calculated as follows:
       $w_i = \phi(F)^{-1} * T$        (9)
   where i = 1 to N (number of neurons in hidden layer).
6. Train the RBF network using weights $w_i$ extracted in
   Step 5.
```

Fig. 2. Q-CNN Training Algorithm

Algorithm: Q-CNN Testing

Input: Testing data **X**= $\{x_1\, y_1 \ldots x_n\, y_n\}$, **T** = {t₁, …, tₙ} where n is number of test samples and Trained Q-CNN
Output: Testing accuracy and Time consumed

```
1. Pass testing data (X, T) to Convolutional and Pooling
   Layers.
2. Extract the test features using equation (7).
3. Pass the test features extracted to trained RBF layer.
4. Tabulate the results using a confusion matrix.
5. Convert confusion matrix into percentage form.
6. Display the mean accuracy and time consumed in
   each run on test data.
```

Fig. 3. Q-CNN Testing Algorithm

IV. Experiments

A. Benchmark datasets used for experiments

Three benchmark datasets are used in this study: CIFAR-10 [34], Digit [35] and MNIST [36]. The CIFAR-10 dataset consisted of 60,000 images with 10 different classes. The 10 objects used in this benchmark dataset are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each class represented 5000 training images and 1000 test images. All the images are colored and 32x32 pixels. The Digit dataset consisted of 10,000 images with 10 different classes. The 10 classes used in this dataset are digits from 0 to 9. It has 7,000 training samples and 3,000 test samples. All images are 8x8 pixels with 256 grey levels. The MNIST dataset consisted of 70,000 images with 10 different classes that are from 0 to 9. It has 60,000 training samples and 10,000 test samples. All images are 28x28 pixels with 256 grey levels.

B. Standard CNN used for comparison

The following standard CNN is used for fetching features and comparing the accuracy. The structure of various CNN layers is shown below.

```
imageInputLayer([h w c])
convolution2dLayer(5,20)
reluLayer
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(10)
softmaxLayer
classificationLayer
```

The features are fetched from pooling layer, in this case "maxPooling2dLayer". Further features are passed to RBF layer for training and classification.

C. Results and Analysis

The proposed approach was implemented, and many experiments were conducted. Three phases are followed in this study: extraction of features by different feature layers, training of features by radial basis function layer, and finally evaluating the proposed approach using testing data. The proposed approach was evaluated with three benchmark datasets MNIST, CIFAR- 10 and Digit datasets. The results are shown in Table I, Table II and Table III. The presented results are average of ten runs.

Table I Results using Q-CNN on MNIST dataset

Methods	**Training Time [Secs.]**	**Test Time [Secs.]**	**Accuracy Training Set**	**Accuracy Test Set**
Standard CNN	628.5	1.55	99.53%	90.01%
Proposed Q-CNN	320.23	12.2	99.94%	99.03%

Results presented in Table I for MNIST dataset show that the training and test accuracies obtained by proposed approach were always higher than the training and test accuracies obtained by standard CNN. Also, the time taken for training and testing was much lower than the standard CNN. For Q-CNN, the training time was only around 321 seconds which includes fetching of features after few epochs and training the RBF layer using fetched features. The testing time is 12.2 second, which is slightly higher than standard CNN, but the test accuracy was 99.03%, which is relatively better than standard CNN.

Table II Results using Q-CNN on CIFAR-10 dataset

Methods	**Training Time [Secs.]**	**Test Time [Secs.]**	**Accuracy Training Set**	**Accuracy Test Set**
Standard CNN	23761.3	61.88	94.35%	89.18%
Proposed Q-CNN	16973.00	105.34	99.91%	87.67%

Results presented in Table II for CIFAR-10 dataset show that test accuracy obtained by proposed approach Q-CNN was slightly lower (1.5%) compare to the test accuracy obtained by standard CNN. However, when it comes to performance time the proposed Q-CNN was much faster than the standard CNN. Training time for standard CNN was 6.6 hours (23,761 seconds) whereas for Q-CNN it was only 4.7 hrs (16,973 seconds). The training accuracy for standard CNN is 94.35% after 40 iterations, whereas training accuracy for proposed Q-CNN is 99.91%. And the test accuracy for standard CNN is 89.18% which is slightly higher than the test accuracy achieved by the proposed Q-CNN which is 87.67%.

Table III Results using Q-CNN on Digit dataset

Methods	**Training Time [Secs.]**	**Test Time [Secs.]**	**Accuracy Training Set**	**Accuracy Test Set**
Standard CNN	108.51	0.621	99.97%	90.00%
Proposed Q-CNN	50.12	3.13	99.96%	99.80%

The results presented in Table III for Digit dataset (a small dataset compare to MNIST and CIFAR-10) shows that the training accuracy obtained by Q-CNN is close to standard CNN whereas the testing accuracy of proposed Q-CNN is slightly higher than the standard CNN. Training time taken for standard CNN is 108.51 seconds, whereas training time taken for the proposed Q-CNN is almost half (50.12 seconds). Test accuracy obtained by Q-CNN is 99.80%, which shows just one image was not classified correctly. The results were also compared with results from recently published papers on feature extraction and classification using different approaches.

Table IV Comparison of classification accuracies listed in published papers

	Accuracy on Test Set		
Methods	**MNIST**	**CIFAR-10**	**Digit**
Standard CNN	99.01%	89.18%	90.0%
Proposed Q-CNN	99.03%	87.67%	99.8%
ConvNet+Softmax [19]	99.01%	86.0%	NA
CNN+ SVM [20]	99.04%	NA	NA

By analyzing the comparative results shown in Table-I - Table-IV above, it was found that the test accuracy obtained by our proposed approach is higher than the other approaches for Digit dataset. The test accuracy for CIFAR-10 dataset is 1.67% higher than the ConvNet+Softmax [19] but was slightly lower than CNN. Whereas for MNIST dataset, the test accuracy obtained by the proposed approach was found competitive with recently published paper on ConvNet+Softmax [19] and CNN+SVM [20]. The training time for the proposed approach is nearly half of the training time taken by the CNN. The training time for other approaches was not available in published papers.

V. Conclusion

A novel non-iterative radial basis function based quick convolutional neural network was presented and investigated in this paper. The evaluation of the proposed Q-CNN was conducted on three benchmark datasets which showed that Q-CNN can perform much better than the traditional CNN. The test accuracy obtained by Q-CNN was higher than the standard CNN in most of the cases. The training time for the proposed Q-CNN were much lower than the time taken by the standard CNN for all datasets. The main reason for better performance was because RBF layer as it was much faster than iterative layer, however it must be noted that it used large number of hidden neurons to achieve high accuracy. In our future work, we will analyze the number of hidden neurons and various least square based methods in RBF layer.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, 2012.

[2] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," in *Nature,* vol. 521, no. 7553, pp. 436-44, May 2015.

[3] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[4] W. Shi, Y. Gong, X. Tao, and N. Zheng, "Training DCNN by combining max-margin, max-correlation objectives, and correntropy loss for multilabel image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2896–2908, Jul. 2017.

[5] H. Shin et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and

transfer learning," in *IEEE Trans. on Med. Imag.*, vol. 35, pp. 1285-1298, 2016.

[6] N. Tajbakhsh et al., "Convolutional neural networks for medical image analysis: full training or fine tuning?," in *IEEE Trans. on Med. Imag.*, vol. 35, pp. 1299-1312, 2016.

[7] W. Sun, T. Tseng, J. Zhang and W. Qian, "Enhancing deep convolutional neural network scheme for breast cancer diagnosis with unlabeled data," *in Comput. Med. Imag. and Graphics*, vol. 57, pp. 4-9, 2017.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," arXiv:1409.4842, 2014.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[10] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. on Art. Intell. and Statistics*, pp. 315-323, 2011.

[11] S. S. Haykin, *Neural Networks and Learning Machines*, vol. 3. Upper Saddle River, NJ, USA: Pearson, 2009.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conf. on Comput. Vis. and Pattern Recognit.*, pp. 1-9, 2015.

[14] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conf. on Comput. Vis. and Pattern Recognit.*, pp. 770-778, 2016.

[16] L. Hertel, E. Barth, T. Kaster and T. Martinetz. (2015). "Deep Convolutional Neural Networks as Generic Feature Extractors," in *IEEE Int. J. Conf. on Neural Netw.,* [Online]. Available: https://arxiv.org/abs/1710.02286

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. Jackel, "Backpropagation applied to handwritten zip code recognition," in *Neural Comp.,* vol.1, pp. 541-551, 1989.

[18] S. Notley and M. Ismail. (2018). "Examining the use of neural networks for feature extraction: A comparative analysis using deep learning, support vector machines, and k-nearest neighbor classifiers." [Online]. Available: https://arxiv.org/abs/1805.02294

[19] J. F. Huang and Y. LeCun, "Large-scale learning with svm and convolutional for generic object categorization," *in IEEE Conf. Comput. Vis. and Pattern Recognit.*, pp. 284–291, 2006.

[20] Y. Tang. (2013). "Deep learning using linear support vector machines." [Online]. Available: https://arxiv.org/abs/1306.0239

[21] A. F. Agarap. (2017). "An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification." [Online]. Available: https://arxiv.org/abs/1712.03541

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[23] L. Fei-Fei, R. Fergus and P. Perona, "One-shot learning of object categories," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006.

[24] A. Sharif Razavian, H. Azizpour, J. Sullivan and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *CVPR Workshops*, pp. 806-813, 2014.

[25] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," in *Pattern Recognit.*, vol. 45, no. 4, pp. 1318-1325, April 2012.

[26] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Apr. 2009.

[27] L. Zhang and P.N. Suganthan , "Visual tracking with convolutional random vector functional link network," in *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3243-3253, 2017.

[28] Y. Zeng , X. Xu , Y. Fang and K. Zhao , "Traffic sign recognition using extreme learning classifier with deep convolutional features," in *Proceedings of the 2015 International Conference on Intelligence Science and Big Data Engineering,* vol. 9242, pp. 272–280, 2015.

[29] S. Pang and X. Yang , "Deep convolutional extreme learning machine and its application in handwritten digit classification," in *Comput. Intell. Neurosci.* pp. 1–10, 2016 .

[30] L. Guo and S. Ding , "A hybrid deep learning CNN-ELM model and its application in handwritten numeral recognition," in *J. Comput. Inf. Syst*. vol. 11, pp. 2673–2680, 2015 .

[31] J. Kim , J. Kim , G.-J. Jang and M. Lee , "Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection," in *Neural Netw.* vol.87, pp. 109-121, 2017.

[32] Z. Xie , K. Xu , W. Shan , L. Liu , Y. Xiong and H. Huang , "Projective feature learning for 3D shapes with multi-view depth images," in: *Computer Graphics Forum,* Vol. 34, Wiley Online Library, pp. 1–11, 2015 .

[33] J. Ghosh and A. Nag "An overview of radial basis function networks," Springer, in *Radial basis function networks* 2, pp. 1-36, 2001.

[34] CIFAR-10 dataset, https://www.cs.toronto.edu/~kriz /cifar.html

[35] Digit dataset from Matlab Toolbox

[36] Y. LeCun, C. Cortes, and C.J.C. Burges. "MNIST handwritten digit database," AT&TLabs [Online]. Available: http://yann. lecun. com/exdb/mnist 2 (2010).