

DEEP KERNEL BASED CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE RECOGNITION

G.Haritha¹, B.Ramyabharathi², K.Sangeetha³

^{1,2,3}Students Dept. of computer science and Engineering, Arasu Engineering college, Tamilnadu, India.

Abstract- In the field of image processing, image recognition can be termed as the ability of software to identify objects, places, people, writing and action in images. Software for image recognition requires deep machine learning. Performance is best on convolutional neural net processor as the specific task requires massive amount power for its compute-intensive nature images are often trained on millions of prelabelled pictures with guided computer learning. Recognizing image in which deep kernel convolutional neural networks are applied with a very efficient CPU implementation. System architecture is carefully designed and image datasets are learned and pre-processed in a supervised way. Deep kernel architecture are based on the results on the benchmarks for image classification (ImageNet datasets). The original ImageNet dataset is a popular large-scale benchmark for training Deep Neural Networks. DeepNets are trained efficiently using a relative straight forward algorithm called Backpropagation. For faster training - use a non-saturating neurons using ReLUs. The ReLUs non linearity is applied to the output of every convolutional and fully connected layer for the better performance. On the very competitive MNIST handwriting benchmark, the neural network has 15,000 images for training, consists of 4 convolutional layers. That is followed by the activation function, dropout that proved to be very effective with the CPU implementation. Images trained will save the network to a file after each iterations and verification of images is done using Backpropagation.

KEY WORDS: Image Processing, convolutional neural network, Backpropagation, Deep learning and image recognition.

1. INTRODUCTION

1.1 Introduction to Image Processing

The key objective is to develop image recognition techniques that are efficient and less complex. Image Processing is used in the checking for presence, object detection and localization, measurement, identification and verification. It is pre-processed to enhance the image according to the specific task as noise reduction, brightness and contrast enhancement. The measuredly devices used must be calibrated to the specified things.

Camera calibration includes Geometric calibration and color calibration. Geometric calibration corrects the lens distortion and determines the relationship between pixels

and real-world units like millimeter. Color calibration ensures an accurate reproduction of colors.

1.2 Machine learning

Machine learning at its most basic is the practice of using algorithm to parse data, learn from it and then make a determination or prediction .So rather than hand coding software routines with a specific set of instructions to accomplish a specific task, the machine is trained using large amount of data and algorithms that provide the ability to learn how to perform task. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting.

Until recently, datasets of labeled images were relatively small—on the order of tens of thousands of images Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label preserving transformations. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized, but it has only recently become possible to collect labeled datasets with millions of images.

1.3 Deep learning

Deep learning has enabled many practical applications of Machine Learning and by extension the overall field of AI. Deep learning breaks down tasks in ways that makes all kinds of machine assists seem possible. Transfer learning is the task of using the knowledge provided by a pretrained network to learn new patterns in new data.

Fine-tuning a pretrained network with transfer learning is typically much faster and easier than training from scratch. Using pretrained deep networks enables you to quickly learn new tasks without defining and training a new network, having millions of images, or long training times. Transfer learning is suitable when you have small amounts of training data (for example, less than 1000 images).

1.3.1 ReLU

The advantage of transfer learning is that the pretrained network has already learned a rich set of features that can be applied to a wide range of other similar tasks. The non-linearity activation functions (ReLU), performed better and decreased training time relative to the activation function.

ReLU has been a default activation function for deep networks.

Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label preserving transformations. To combat the problem of overfitting, recently developed regularization technique “dropout” that proved to be effective in training data. The proposed style of having successive convolution and pooling layers, followed by fully-connected layer has been the state-of-the-art networks today.

1.4 Convolutional layer

The convolutional layer is parameterized by the size and the number of maps, kernel sizes, skipping factors and the connection table. Pooling layer skips nearby pixels prior to convolution and leads to faster convergence, superior invariant features and improve generalization.

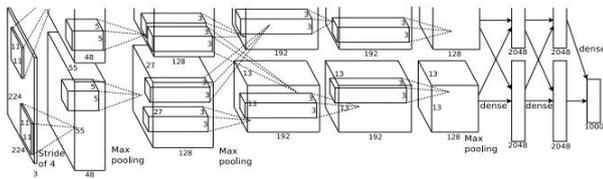


Fig. 1 Architecture of CNN with dimensions

The layer enables position invariance over larger local regions and down samples the input image. CNNs are hierarchical neural networks whose convolutional layers alternate with subsampling layers, reminiscent of simple and complex cells in the primary visual cortex.

1.5 Convolutional Neural Networks

Convolutional Neural Networks (CNN) constitutes one such class of models. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, Stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feed forward neural networks with similarly sized layers, CNNs have much fewer connections and parameters and so they are easier to train. Image net consists of variable-resolution images, while our system requires a constant input dimensionality.

Therefore we down-sampled the images to a fixed resolution of 256*256. Backpropagation worked well for a variety of tasks, but in the 1980s it did not live up to the very high expectations of its advocates. In particular, it proved to be very difficult to learn networks with many layers and these were precisely the networks that should have given the most impressive results.

Many researchers concluded, incorrectly, that learning a deep neural network from random initial weights was just

too difficult. Twenty years later, we know what went wrong: for deep neural networks to shine, they needed far more labeled data and hugely more computation. On the very competitive MNIST handwriting benchmark, the neural network has 15,000 images for training, consists of 4 convolutional layers. That is followed by the activation function, dropout that proved to be very effective with the CPU implementation.

Combining the predictions of many different models is a very successful way to reduce test errors, but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently introduced technique, called “dropout”, consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in back propagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.

It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At test time, all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks is used. Without dropout, the network exhibits substantial over fitting. Dropout roughly doubles the number of iterations required to converge.

For the MNIST dataset the networks are trained on deformed images, continually generated in on-line fashion. (Translation, rotation, scaling, horizontal shearing) and elastic deformations are combined. We use a variable learning rate that shrinks by a multiplicative constant after each epoch. Fully connected convolutional layers lead to an exploding number of network connections and weights, making training of big and deep CNNs for hundreds of epochs impractical even on GPUs. Partial connectivity alleviates this problem and is also biologically more plausible.

We reduce the number of connections between convolutional layers in a random way. Additional layers result in better networks, the best one achieving a test error of 0.35% for best validation and a best test error of 0.27%. The best previous CNN result on MNIST is 0.40%, 0.35% error rate was recently also obtained by a big, deep MLP with many more free parameters. Deeper nets require more computation time to complete an epoch, but we observe that they also need fewer epochs to achieve good test errors.

2. LITERATURE SURVEY

“IMAGENET CLASSIFICATION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS” by Alex krizhevsky, Ilyasuts kever, and Geoffrey. The paper Contributes the error rates on the Fall 2009 version of ImageNet with 10,184 categories and 8.9 million images were reported. On this dataset, the convention in the literature of using half of the images for training and half for testing was followed. Since there is no established test set, split necessarily differs from the splits used by previous authors, but this does not affect the results appreciably. Here, top-1 and top-5 error rates on this dataset are 67.4% and 40.9%, attained by the net described above but with an additional, sixth convolutional layer over the last pooling layer. The best published results on this dataset are 78.1% and 60.9%.

“HIGH PERFORMANCE NEURAL NETWORKS FOR VISUAL OBJECT CLASSIFICATION” by Dan C. Cirean, Velimeier, Jonathan masus, Luca M. Gambardella, and Jurgen. The paper Contributes the high-performance GPU-based CNN variants trained by on-line gradient descent, with sparse random connectivity, computationally more efficient and biologically more plausible than fully connected CNNs. Principal advantages include state-of-the-art generalization capabilities, great exibility and speed. All structural CNN parameters such as input image size, number of hidden layers, number of maps per layer, kernel sizes, skipping factors and connection tables are adaptable to any particular application. The networks to benchmark datasets are applied for digit recognition (MNIST), 3D object recognition (NORB), and natural images (CIFAR10). On MNIST the best network achieved a recognition test error rate of 0.35%, on NORB 2.53% and on CIFAR10 19.51. Currently the particular CNN types discussed in this paper seem to be the best adaptive image recognizers, provided there is a labeled dataset of sufficient size. No unsupervised pretraining is required. Good results require big and deep but sparsely connected CNNs, computationally prohibitive on CPUs, but feasible on current GPUs, where the implementation is 10 to 60 times faster than a compiler-optimized CPU version.

“CONVOLUTION DEEP BELIEF NETWORKS FOR SCALABLE LEARNING OF HIERARCHICAL REPRESENTATION” by Honglak Lee, Roger Grosse, Rajesh Ranganath, Andrew Ng. The convolutional deep belief network, a scalable generative model for learning hierarchical representations from unlabeled images is presented, and showed that model performs well in a variety of visual recognition tasks. Thus the approach holds promise as a scalable algorithm for learning hierarchical representations from high-dimensional, complex data.

“MULTI-COLUMN DEEP NEURAL NETWORKS FOR IMAGE CLASSIFICATION” by Dan Cires, Ueli Meier and Jurgen Schmidhuber. This is the first time human-competitive results are reported on widely used computer vision benchmarks. On many other image classification

datasets our MCDNN improves the state-of-the-art by 30-80% .The recognition rates on MNIST, NIST SD 19, Chinese characters, traffic signs, CIFAR10 and NORB were drastically improved. Our method is fully supervised and does not use any additional unlabeled data source. Single DNN already are sufficient to obtain new state-of-the-art results; combining them into MCDNNs yields further dramatic performance boosts.

3. PROPOSED WORK

The proposing system uses deep kernel method in deep CNN. The mixture of base kernel with a deep architecture has best expressive power and scalability. The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map. A non-linearity layer in a convolutional neural network consists of an activation function that takes the feature map generated by the convolutional layer and creates the activation map as its output. A rectification layer in a convolutional neural network performs element-wise absolute value operation on the input volume (activation volume).

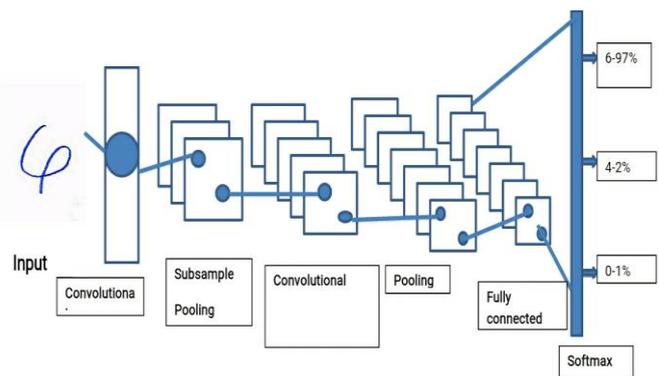


Fig.2 SYSTEM ARCHITECTURE

The architecture of the Convolutional neural network layer is given as For testing representation, the performance of the model on the MNIST handwritten digit classification task is a widely-used benchmark .To learn new patterns in new data, transfer learning is the task of using the knowledge provided by a pretrained network, which is much faster and easier than training from scratch. In CNN the core building block is a convolutional layer, which consists of a set of learnable filters (or kernels), with a small receptive field, but extend through the full depth of the input volume. To compute the dot product between the entries of the filter and the input, each filter is convolved across the width and height of the input volume, during the forward pass and producing a 2-dimensional activation map of that filter. To create various new applications of deep learning neural networks, the deep layers of processing, convolutions, pooling, and a fully connected classification layer are used opened. In addition to image processing, the CNN has been applied to video recognition and various tasks successfully within natural

language processing. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. For forming the full output volume of the convolution layer, the activation maps are stacked for all filters along the depth dimension.

Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map. On every depth slice of the input, the pooling layer operates independently and resizes it spatially. The filters of size 2×2 applied in the pooling layer with a stride of 2 down samples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations is the most common form. In this case, every max operation is over 4 numbers. The dimension of the depth remains unchanged. Finally, in neural network the high level reasoning is done via fully connected layers, after several convolutional layer and max pooling layers. The Neurons in the fully connected layer have connections to all activations in the previous layer and their activations can be computed with a matrix multiplication followed by a bias offset. Convolutional neural networks use three basic ideas: local receptive fields, shared weights, and pooling.

3.1 Local receptive fields

The inputs were showed as a vertical line of neurons in fully connected layers. In a convolutional net, instead of the input square of neurons, those values correspond to the same dimensional pixel intensities. The input pixels to a layer of hidden neurons are connected. But not every input pixel to every hidden neuron is connected. Instead, only connections in small, localized regions of the input image are made. To be more precise, each neuron in the first hidden layer will be connected to a small region of the input neurons. That region in the input image is called the local receptive field. For each local receptive field, there is a different hidden neuron in the first hidden layer. Then the local receptive field over by one pixel to the right is slid (i.e., by one neuron), to connect to a second hidden neuron. And so on, building up the first hidden layer. We have an input image, and local receptive fields, and then there will be neurons in the hidden layer. In fact, sometimes a different stride length is used.

3.2 Shared weights and biases

Each hidden neuron has a bias and weights connected to its local receptive field. The *same* weights and bias is used for each of the hidden neurons. This means that all the neurons in the first hidden layer detect exactly the same feature. Features detected by a hidden neuron as the kind of input pattern that will cause the neuron to activate, it might be an edge in the image, for instance, or maybe some other type of shape, just at different locations in the input image. That ability is also likely to be useful at other places in the image. And so it is useful to apply the same feature detector

everywhere in the image. To put it in slightly more abstract terms, convolutional networks are well adapted to the translation invariance of images: move a picture of a cat (say) a little ways, and it's still an image of a cat. In fact, for the MNIST digit classification problem, the images are centered and size-normalized. So MNIST has less translation invariance than images found. Still, features like edges and corners are likely to be useful across much of the input space. For this reason, sometimes the map from the input layer to the hidden layer a *feature map* is called and the weights defining the feature map the *shared weights* is called. And we call the bias defining the feature map in this way the *shared bias*. The shared weights and bias are often said to define a *kernel* or *filter*.

A big advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network. For each feature map we need shared weights, plus a single shared bias. So each feature map requires parameters based on the shared weights and bias mentioned. By comparison, suppose we had a fully connected first layer, with input neurons, and a relatively modest hidden neurons. In other words, the fully-connected layer would have more times as many parameters as the convolutional layer. Thus the direct comparison between the number of parameters cannot be done, since the two models are different in essential ways. But, intuitively, it seems likely that the use of translation invariance by the convolutional layer will reduce the number of parameters it needs to get the same performance as the fully-connected model. That, in turn, will result in faster training for the convolutional model, and, ultimately, will help us build deep networks using convolutional layers.

Incidentally, the name *convolutional* comes from the fact that the operation is sometimes known as a *convolution*. A little more precisely, sometimes the equation is written as $a_1 = \sigma(b + w * a_0)$, where a_1 denotes the set of output activations from one feature map, a_0 is the set of input activations, and $*$ is called a convolution operation. Any deep use of the mathematics of convolutions is not used, so the information about the connection need not be worried.

3.3 Pooling layers

In addition to the convolutional layers just described, convolutional neural networks also contain *pooling layers*. Pooling layers are usually used immediately after convolutional layers. What the pooling layers do is simplify the information in the output from the convolutional layer.

In detail, a pooling layer takes each feature map; the nomenclature is being used loosely here. In particular, "feature map" is used to mean not the function computed by the convolutional layer, but rather the activation of the hidden neurons output from the layer. This kind of mild abuse of nomenclature is pretty common in the research

literature. Output from the convolutional layer and prepares a condensed feature map. For instance, each unit in the pooling layer may summarize a region of (say) $2 \times 2 \times 2$ neurons in the previous layer. As a concrete example, one common procedure for pooling is known as *max-pooling*. Max-pooling isn't the only technique used for pooling. Validation data to compare several different approaches to pooling is used to optimize result, and the best approach is chosen.

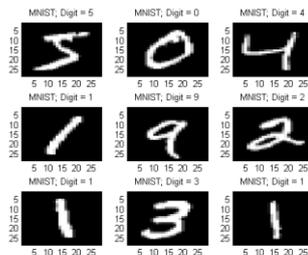


Fig.3 Sample MNIST image dataset

This image is a simple neural network model with a single hidden layer and creates a simple multi-layer perceptron model. The training dataset is structured as a 3-dimensional array of instance, image width and image height. The pixel values are gray scale between 0 and 255. It is almost always a good idea to perform some scaling of input values when using neural network models. The pixel values can be quickly normalized to the range 0 and 1 by dividing each value by the maximum of 255. Finally, the output variable is an integer from 0 to 9. This is a multi-class classification problem.

4. SYSTEM IMPLEMENTATION

Using pretrained deep networks enables you to quickly learn new tasks without defining and training a new network, having millions of images, or long training times. Transfer learning is suitable when you have small amounts of training data (for example, less than 1000 images). The advantage of transfer learning is that the pretrained network has already learned a rich set of features that can be applied to a wide range of other similar tasks.

The image is trained on 15,000 images and the dataset is trained automatically on a specified network configuration. The dataset reached 94% in several minutes and 99.2% in couple of hours. The last parameter is removed and trained a bit longer to achieve 99.2%. Once it has gone below threshold value, the training is stopped automatically. To check the network on MNIST dataset, a small GUI is opened with a user defined function and shows the network filters on some input image. The network consists of 4 layers. The 4 layers are processed with 15 iterations.

Each iteration takes 1000 images, its mean weight, training error, testing error, success rate, and its total time Training done will save the network to a file after each iteration. The

properties of the network involve number of layers, number of outputs, version, sources, weights, input size. Ultimately, the maximum samples to be tested are reached and it is displayed with the total time for each iteration with the success filters on some input image. The network consists of 4 layers. The 4 layers are processed with 15 iterations. The maximum samples to be tested are reached and it is displayed with the total time for each iteration with the success rate of 97.010000%.

4.1 Convolutional layer

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptron, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks. A convolutional neural network is also known as a ConvNet. The following is the algorithm followed for convolutional layer

- Step 1 :Take training data where each element consist of a feature vector
- Step2: Generate stronger feature vectors that are more invariant image distortion and position.
- Step 3 :Generate feature maps
- Step 4 :we get better feature by alternating convolutional

4.2 Non-Linearity layer

The purpose of the activation function is to introduce non-linearity into the network in turn, this allows you to model a response variable that varies non-linearly with its explanatory variables. On-linear means that the output cannot be reproduced from a linear combination of the inputs another way to think of it: without a non-linear activation function in the network, a NN, no matter how many layers it had, would behave just like a single-layer perceptron, because summing these layers would give you just another linear function. The following is the algorithm followed for non-linearity layer.

- Step 1: Maps the data non-linearly in to some feature space
- Step 2: The co-ordinates of the space are given by the activations of the last hidden units
- Step 3: The network performs logistic regression
- Step 4: The function will be non-linear, as the consequence of the non-linear mapping from input to hidden unit activations.

4.3 Pooling layer

The most common methods for reduction are max pooling and average pooling. Max-pooling layers follow the convolutional layers for down sampling. Hence reducing the number of connections to the following layers (usually a fully connected layer). They do not perform any learning themselves, but reduce the number of parameters to be learned in the following layers. They also help reduce overfitting. Max pooling has demonstrated faster convergence and better performance in comparison to the average pooling. The following is the algorithm followed for Pooling layer.

Step 1 :N: Batch Size

Depth: Input volume depth

H:Image Height

W:Image Width

H: Pooled image height

W:Pooled image width

Step 2 :Parameters Stride(s) Pooling windows sizes(H-P,W-P)

Step 3 :Output*W*Depth*N

Depth does not change, only spatial information

$H=1+(H-H_P)/S$

$W=1+(W-W_P)/S$

Output Depth=Depth

5. Conclusion

Convolutional neural networks are an architecturally different way of processing dimensioned and ordered data. Instead of assuming that the location of the data in the input is irrelevant convolutional and max pooling layers enforce weight sharing transnationally. This models the way the human visual cortex works, and has been shown to work incredibly well for object recognition and a number of other tasks. We can learn convolutional networks through traditional stochastic gradient descent. This convolutional architecture is quite different to the architectures used. But the overall picture is similar: a network made of many simple units, whose behaviors are determined by their weights and biases. And the overall goal is still the same: to use training data to train the network's weights and biases so that the network does a good job classifying input digits.

6. Future Enhancement

The proposed recognition system is implemented on handwritten digits taken from MNIST database. Handwritten digit recognition system can be extended to a recognition system that can also able to recognize handwritten character and handwritten symbols, images of animals such as cats, dogs.

REFERENCES

1. Bell, R., Koren, Y. Lessons from the netflix prize challenge. ACM SIGKDD Explor. Newsl. 9, 2 (2007), 75–79.
2. Berg, A., Deng, J., Fei-Fei, L. Largescale visual recognition challenge 2010. www.image-net.org/challenges.2010.
3. S. Behnke. Hierarchical Neural Networks for Image Interpretation, volume 2766 of Lecture Notes in Computer Science. Springer, 2003.
4. K. Chellapilla, S. Puri, and P. Simard. High performance convolutional neural networks for document processing. In International Workshop on Frontiers in Handwriting Recognition, 2006.
5. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code Recognition. Neural Computation, 1, 541–551.
6. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., et al. Handwritten digit recognition with a back-propagation Network. In Advances in Neural Information Processing Systems (1990).
7. P. R. Cavalin, A. de Souza Britto Jr., F. Bortolozzi, R. Sabourin, and L. E. S. de Oliveira. An implicit segmentation-based method for recognition of handwritten Strings of characters. In SAC, pages 836–840, 2006. 4
8. Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, and S. Cho, editors, Neural Networks: The Statistical Mechanics Perspective, pages 261–276. World Scientific, 1995. 3
9. Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. International Conference on Machine Learning (pp. 759–766).
10. Lee, H., Grosse, R., Ranganath, R., Ng, A. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th Annual International Conference on Machine Learning+ (2009). ACM, 609616.

11. K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161{180, 2003.
12. R. Uetz and S. Behnke. Large-scale object recognition with CUDA-accelerated hierarchical neural networks. In *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, 2009.
13. Ranzato, M., Huang, F.-J., Boureau, Y.-L., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*
14. P. J. Grother. NIST special database 19 - Handprinted forms and characters database. Technical report, National Institute of Standards and Technology (NIST), 1995. 1, 4
15. Turaga, S., Murray, J., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Dank, W., Seung, H. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.* 22, 2 (2010), 511–538.

BIOGRAPHIES



Haritha.G received B.E degree in Computer Science and Engineering from Arasu college of Engineering and pursuing M.E degree in the stream of computer science and engineering at Arasu engineering college Kumbakonam.



Ramyabharathi.B received B.E degree in Computer science and Engineering from University college of Engineering, BIT Campus, Tiruchirappalli and pursuing M.E degree in the stream of computer science and engineering at Arasu engineering college Kumbakonam.



Sangeetha.K received B.E degree in Computer Science and Engineering from Arasu college of Engineering and pursuing M.E degree in the stream of computer science and engineering at Arasu engineering college Kumbakonam