CNN Features Off-the-Shelf: Clustering and Finetune-free

Qun Liu
Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana
qliu14@lsu.edu

Supratik Mukhopadhyay
Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana
supratik@csc.lsu.edu

***Abstract*—Deep convolutional features learned from a large-scale labeled dataset, contain substantial representations which could be effectively used in a new domain. Despite the fact that generic features achieved good results in many visual tasks, fine-tuning is required for pretrained deep CNN models to be more effective and achieve state-of-the-art performance. Undoubtedly, the backpropagation algorithm, known as a time and resource consuming process, adopts for that purpose to fine-tune the millions of parameters with a significant amounts of labeled data. This paper is dedicated to propose a pipeline framework integrating Hopfield Associative Memory as memory bank, to eliminate the backpropagation while utilizing off-the-shelf features on new domain. Due to simply finding worthwhile memory patterns, clustering as an unsupervised learning approach used for computing retrieved feature maps (patterns) consists in memory bank. Our proposed method has achieved empirical performance over previously published results on multiple benchmark datasets.**

***Index Terms*—Deep Convolutional Features, CNN Transfer, Hopfield Associative Memory, Non-Backpropagation**

I. Introduction

Recent advances in the Convolutional Neural Networks (CNN) has given rise to powerful techniques for solving a variety of problems in Computer Vision [1]–[3], especially on image classification and segmentation, visual tracking, etc. It is common that the features pooled from the CNN models indicate the advanced and complex descriptors compared to previously handcrafted ones [4]. As along with deep learning arises, transfer learning acquired great success and performance in the aspect of research and industry fields, it possessed a vital role for attaining good feature representations from the pretrained CNN model. Due to finding the target large-scale dataset is very hard and almost infeasible for obtaining robust representations in some domains, the CNN model pretrained on the large-scale dataset (e.g. ImageNet [5]) has achieved competitive effectiveness on other related domains after finetuning, which its images do not have much different in context with that in the pretrained dataset. In this paper, we deployed the pretrained CNN model in our pipeline framework, for feature extraction in classification task.

Supported by a TIRE grant from Louisioana Transportation Research Center

Among all previous literatures [6], [7] demonstrating the ability and prevalent in CNN transfer for classification, our work differs and devotes in two aspects. First, the pretrained CNN need to finetune the parameters on the new dataset to ensure the model has better accuracy. But finetunes the millions of parameters trained on large-scale dataset through back propagation, it suffers time and resource consuming. In this work, we made initial effort, using associative memory, for eliminating the back propagation while keeping good performance upon classification on new large-scale dataset. It once retrained, though tested on other different relevant datasets, it showed good resistant to the variance and also obtained competitive performance even without retraining. Second, due to the small dataset or less data, overfitting issue is easier occur [8]. In some cases one usually adopts linear classifier, e.g. Support Vector Machine (SVM), on the top of network instead, which has achieved state-of-the-art performance as the advances of normal SVM outperforms well on small dataset, but it suffers from the training complexity of the large dataset [9], [10]. Some recent studies made efforts on improving SVM on large-scale dataset, but our novel framework based on memory bank exhibits good traits with respect to mitigating all that aspects simultaneously while obtains competitive accuracy.

In our this work, we integrated Hopfield Associative Memory, a recurrent artificial neural network, as memory bank to store patterns for classification pipeline. Recently some studies [11], [12] demonstrates the potential of neural associative memory in pattern recognition and the robust its to adversarial inputs, and yet it still have not been used in deep learning. We investigated and devoted our initial effort on this area. As shown in Fig. 1, the framework utilized pretrained CNN classification model to extract feature maps from the input images, then compute the centers of feature maps as stored patterns feeding to memory bank (Hopfield Network) to infer the test image class.

To the best of our knowledge, this is the first work integrating memory bank, taking full advances of artificial neural network, for classification in CNN model paradigm.

We summarize our contributions as below:

- Introduced the notion of memory bank in CNN classification model, which its patterns used for recognizing the test images and its class label.
- Selected a recurrent artificial neural network, Hopfield Network, as memory bank, then combined it with the pretrained CNN model. We proposed a novel pipeline framework for classification problem.
- Utilized unsupervised learning approach in memory for computing centers of feature maps, which stored as core patterns. We speculate that multiple centers adopted to represent well the intra-class variance in the memory.

The paper is organized as follows. In Section II, we will present the related work briefly. The pipeline framework architecture and Hopfield Network will be discussed in Section III, the classification algorithm is provided. We will demonstrate the experimental results in Section IV and the conclusions presented in Section V.

II. Related Work

Deep learning attracted a lot of world-wide efforts spanned from academic research to real-world industry applications, as its incredible success since been achieved on large-scale data by a superior margin substantially [13]. Varieties of deep learning models then proposed and the accuracy been improved time after time. Its powerful in recognition have already surpassed human-level performance [14], [15]. Deep features learned from the pretrained model have showed many significances and possessed state-of-the-art performance in vision related tasks. No doubt deep features reusing becomes a trend as it is, boosted the performance properties in many cutting-edge domains, many research works [16], [17] appeared in this hot area, for image classification, segmentation, object detection, etc.

Though deep learning as a supervised approach attained competitive accuracy and performance with amounts of labeled data, but in the real world, obtaining more labeled data is very expensive and impractical at some extents. Unsupervised learning is getting rapidly attention in machine learning era since it learned from unlabeled data [18]. With small labeled data, one can combine the labeled and unlabeled data in a semi-supervised learning approach [19]. In the typical setting of computer vision, attentions has gained from world wide researchers to utilize that address the image classification related problems. K-means as a popular clustering approach in unsupervised learning, its widely use can be found in many works, in deep learning [20]–[22].

Hopfield Associative Memory [23], as a recurrent neural network, has adopted in our framework for its concise mechanics simulating human brain activities based on energy function as well as its previous successful performance in pattern recognition and its robust to adversarial inputs [11], [12]. However, other types of auto-associative memory could also be considered in our framework utilized as memory bank.

The CapsNet architecture has been proposed recently which designed in an innovative way [24]. It based on the conception of capsules. The base layers of its network convert image pixels intensities to activities as input for the primary capsules, which are the lowest level of multi-dimensional entities in the length of the vectors. The DigitCaps layer indicates presence of an instance of each class, for calculating the classication loss. In our pipeline framework, core patterns adopted instead, which are the instances of class, and memory patterns represented in the vectors retrieved from our pipeline framework. In computing classication loss, both used the euclidean distance for minimizing. Utilizing the whole vector for an explicit representation of the pose of the feature also can be found in the early work [25]. It demonstrated its competitive to deal with variations using the whole vector of instantiation parameters from simple separated capsules. More recently, in vision recognition a probabilistic generative model proposed [26] and it introduced a hierarchical model named Recursive Cortical Network(RCN), it handles the recognition, segmentation, and reasoning in an unified way and it modeled objects as a combination of contours and surfaces. But differently in our pipeline framework we simply used the uniformed way to learn the features taking the fully advantages of the pretrained CNN model.

III. Proposed Method

In this section, we present the overview of our proposed method, then next we demonstrate the Memory Bank used in our pipeline framework. The core patterns selection and the Hopfield network described in Section III-C and Section III-D.

A. Overview

The pipeline framework is designated to integrate the associative memory with CNN model to eliminate the back-propagation process, while achieves and retains state-of-the-art performance. The overall architecture illustrated in Fig.1. As shown from the figure, we utilize the pretrained CNN model for feature embeddings extraction, here the ResNet-50 [27], is pretrained on ImageNet [5], adopted for the framework basement as its competitive classification performance. The features our framework extracted from pool5 (of ResNet-50) before the dense layers, utilized as the representations of the input images. During the training phase, the framework calculate the class-specific features set, which is a set of all images features, then the core patterns subsequently computed, we address the details of this procedure in Section III-C. The set of core patterns stored in memory bank, which is the memory of Hofield Network. During testing stage, we extracted features of each test image from pool5, and then based on Hopfield network to compute patterns from input features, we find its associated core pattern in memory bank and return its label for the test image, more details demonstrated in III-E.

B. Memory Bank

We proposed the notion of memory bank for storing the patterns, which used in the architecture. It is simply as a collection of all core patterns. Hopfield network applied on memory bank, for the purpose of retrieving patterns during test

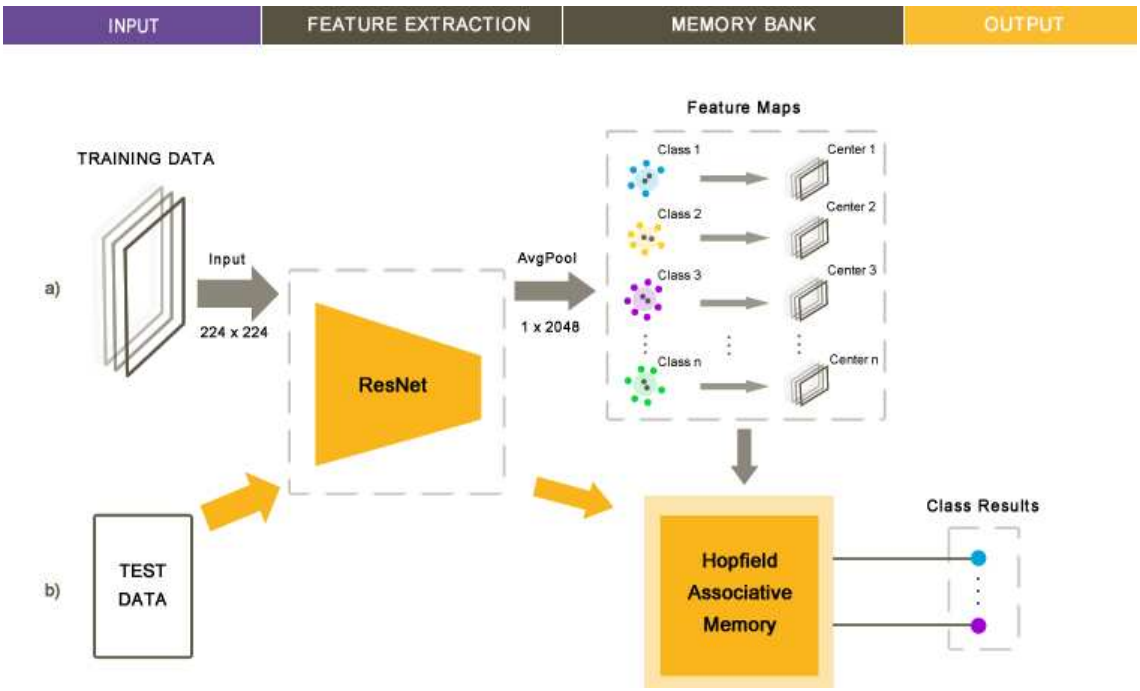

Fig. 1. Overview of Pipeline Framework Architecture

stage and storing patterns during training stage. Given training images of n classes, m images for class n_i which denotes the i-th class, have p_i $=\{p_i^1, p_i^2,...,p_i^m\}$ memory patterns, each memory pattern computed from the features f pooled from pool5 in the pretrained CNN model. Core patterns then calculated among sets of memory patterns, details presented in the next subsection.

C. Core Patterns Selection

In the memory bank, we have set of patterns p_i, which denotes the set of patterns for i-th class, the core patterns selection process simply utilize K-means, an unsupervised clustering approach, for calculating the cluster centers $c_i =\{c_i^1,c_i^2,...,c_i^k \mid k\in 1,2,...,\mathrm{m}\}$, as core patterns for each set p_i, by computing Euclidean distance. Then the core pattern sets c spanned overall classes obtained through above and further utilized for Hopfield network. However, after training stage, all core patterns have achieved, these core patterns in the memory bank served as the candidates for Hopfield network to retrieve. When test image feed to the network, its corresponding core pattern restored through memory bank and then its associated class subsequently be determined.

D. Hopfield Network

The Hopfield associative memory is a single layer of fully connected recurrent neural network, shown in Fig.2. It stimulates the storing and retrieving process in our human brain. The neurons in Hopfield network can be updated in asynchronous or synchronous. For asynchronous, a neuron gets update in random or fixed order once its weighted input sum calculated, while in synchronous, all neurons get updated at same time. Given a network with N neurons, the weighted input sum of a neuron known as local field can be described as follows,

$$\xi_i = \sum_{j=1}^{N} w_{ij} x_j \tag{1}$$

where $i, j \in 1, 2, ..., N$, the synaptic weight w_{ij} is for the connection weight of i-th and j-th input. x_j is the state of j-th input. The state of the entire network can be represented by a vector $v = [x_1, x_2, ..., x_N]$. For each input pattern φ has N dimensions which represented by N neurons in Hopfield network.

The Hopfield network memorize the core patterns z denote as $\varphi_1, \varphi_2, ..., \varphi_z$ during training stage, and retrieve the stored patterns in test stage. Hebbian learning used for memorizing the patterns in Hopfield network, specifically, determine the synaptic weights w_{ij}, which is given by

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_z \varphi_{z,i} \varphi_{z,j} & i \neq j \\ 0 & i = j \end{cases} \tag{2}$$

note that the weight connection of i-th and j-th neuron is symmetric, which $w_{ij} = w_{ji}$.

For retrieving purpose, suppose we have the test pattern φ_{test}, then x_i is the state of the i-th element in test pattern, denotes as $\varphi_{test,i}$ and $i = 1, 2, ..., N$. Then all the elements in state vector v update over the network asynchronously as described by the following,

$$x_i(t+1) = sign(\sum_{j=1}^{N} w_{ij} x_j(t)) \tag{3}$$

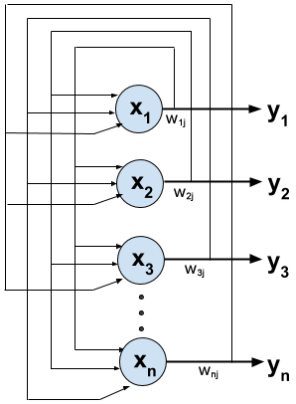

Fig. 2. Hopfield network structure.

where $x_j(t)$ is the state of the j-th neuron at time t. The update changes in the direction of reducing the network energy, once the energy of the network minimized, all the states become stable and retain unchanged, the final stable network states denotes as v_s obtained. Following (1) the energy E_i for neuron i can be described as belows,

$$E_i = -\frac{1}{2}\xi_i x_i \tag{4}$$

Then the energy for the entire network [28] can be computed by the following,

$$E\left(v\right) = \sum_{i=1}^{N} E_i = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j \tag{5}$$

Even the test pattern is incomplete or broken, Hopfield network can still retrieve the corresponding stored pattern from the memory due to its intrinsic of error-correcting and noise-resilience property. The network is guaranteed to converge to stable states, once the energy of the network has minimized and reached the energy minimum or attractor basin, the associated final states in the network then represented as memory patterns in response to the test pattern.

E. Problem Formulation

We utilized Hopfield network as associative memory to store and retrieve patterns and its details explained in the previous section. In classification problem, when obtained the retrieved pattern, the associated class with it then attained and computed. The problem therefore transforms to the class match problem, and the corresponding algorithm have been implemented.

The patterns memorized in the Hopfield network as described can be simply treated as the weight matrix [29] and the (2) can be transformed by using matrix product, which is given by

$$W = \frac{1}{N}\sum_{z} \varphi_z^{\mathsf{T}}\varphi_z - MI \tag{6}$$

where φ_z^{T} is the transpose of the vector φ_z then matrix product calculated and W is the weight matrix. In the network there is no self connections so that the synaptic weight $w_{ii} = 0$ and I in (6) denotes the identity matrix.

So that we can obtain the retrieved pattern by matching the test pattern with the stored patterns through computing and comparing similarities among their weight matrix, Euclidean distance can be calculated for the similarity as follows,

$$Diff(T,S) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(T_{ij} - S_{ij})^2} \tag{7}$$

where T and S are weight matrix for test pattern and stored pattern, n is the dimensions of the weight matrix. Then compute the retrieved pattern can be formulated as belows,

$$argmin_i\ Diff(T, S_i) \tag{8}$$

where S_i is the i-th stored pattern, then the one which has the smallest difference with the test pattern is represented as retrieved pattern. The class label of the test pattern is subsequently identified.

The algorithm implemented for classification is presented in Alg. 1. For each test pattern t in test data, the algorithm computes the difference between t and stored patterns S_i in each class following formula (7), and find the stored pattern R_i which has the minimum difference in its class. Lastly all the stored patterns R_k over k classes obtained and following formula (8), the smallest difference pattern acquired with its associated class label l, and eventually returned l as the result of classification.

Algorithm 1: Classification Algorithm

Input : Test pattern t;
Stored patterns $\{S_i\}_{i=1}^{z}$ in each class C_k.
Output: Class label set l.
initialize $l \leftarrow \emptyset$;
for *k=1 to n* **do**
  for *i=1 to z* **do**
    $R_i \leftarrow Union(i, Diff(t, S_{ki}))$;
    // Compute difference within a class
  end
  $R_k \leftarrow argmin_i\ R_i$;
  // Compute minimum within a class
end
$l \cup \{k|argmin_k\ R_k\}$;
// set of final class labels
if $len(l) > 1$ **then**
  return $random\_choice(l)$;
else
  return l;
end

IV. Experiment

To better illustrate the performance of our pipeline framework for classification, we conducted empirical experiments

presented in this section, mainly focus on two popular object classification datasets: Caltech101 [30] and Caltech256 [31]. First demonstrated the details of these datasets and then we compared the results in two network structures with the state-of-the-arts. We utilized ResNet-50 and VGG-16 as the pretrained CNN model in our pipeline framework.

A. Datasets

Caltech101 consists of 9144 images of 101 object categories and 1 background category. The variety of classes include faces, animals, camera, etc. The images in the dataset have varies in the degree of shape and scale. For each categories it has about 40 to 800 images and most categories have about 50 images.

Caltech256 containing 30607 images for 256 object categories and 1 clutter category. It has minimum of 80 images for each category, compared to Caltech101, Caltech256 is more complex and challenging due to it has more varieties in the size, background, etc.

B. Implementation details

In our pipeline framework applied on these datasets in the experiments, we utilized ResNet-50 [27] and VGG-16 [32] as pretrained CNN models for features extraction, both models are pretrained on ImageNet [5]. The experiments mainly concentrate on the ResNet-50 model due to its state-of-the-art performance obtained for classification and its competitive feature representations, but the results based on VGG-16 model also provided. Both models all have five convolutional blocks and the features pooled from the pool5, which has 1x1x2048 dimensions and 1x1x256 dimensions in ResNet-50 and VGG-16, respectively. The input image size for ResNet-50 and VGG-16 are same at the size of 224x224. For unsupervised learning on clustering to attain the core patterns, K-means simply applied on the patterns which are pooled features from the pretrained CNN model (see section III-C), and the algorithm implemented for classification provided in section III-E. The Caltech101 and Caltech256 datasets which the pretrained CNN models applied on, are divided into 30% for testing and 70% for training in the experiments. The evaluation metric we adopted for the performance evaluation of our pipeline framework is the average of the per-class accuracies obtained on the datasets.

C. Evaluation

We evaluate the performance of our pipeline framework on various number of core patterns, recall that the core patterns are the center patterns computed by K-means. The results shown in Fig.3 and Fig.4 we reported based on the Caltech101 and Caltech256 datasets from ResNet-50 and VGG-16 models. It is clear that from Fig.3 the pipeline framework exhibited good performance and the classification accuracy improved with the increase in the number of core patterns for each class, but it reached relatively stable or a slight decrease in the further, and it also showed similar on Caltech256 dataset but has more relatively stable after increased at the first. This validates that multiple core patterns utilized in pipeline framework is beneficial for performance improvement instead of one, since unsupervised clustering needed to increase the number of core patterns to learn varieties of common features to obtain competitive performance, but the performance would not have improved if exceeds its varieties, since there is no information gain yield.

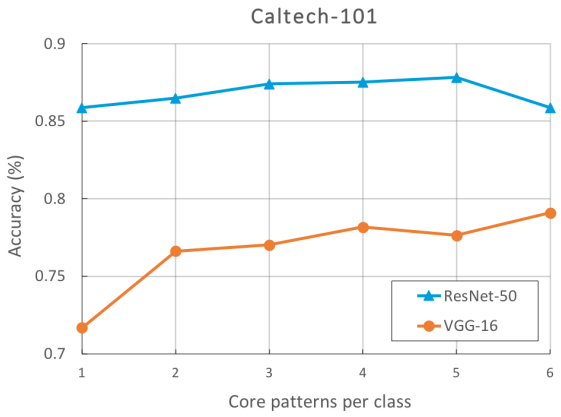

Fig. 3. Impact of number of core patterns per class on Caltech101 dataset.

D. Comparison with state-of-the-art methods

In this section we compared our classification results with the state-of-the-art methods on Caltech101 and Caltech256 datasets, we showed the comparisons in Table I. The classifiers for SVM and Softmax in the table are added on the top of the seven layers fixed model which pretrained on ImageNet dataset [33] and then retrained on the new corresponding training datasets.

TABLE I
CLASSIFICATION ACCURACY (%)

Methods	Caltech101	Caltech256
SHDL [34]	81.5	-
Image Codes [35]	71.4	35.7
FL+EN [36]	83.2	-
Zeiler-Fergus [33]	86.5	74.2
Shaban [37]	75.1	-
SVM [33]	85.5	71.7
Softmax [33]	85.4	72.6
Ours(VGG-16)	79.1	67.8
Ours(ResNet-50)	**87.8**	**78.0**

The classification accuracies our model achieved are **87.8%** on Caltech101 and **78.0%** on Caltech256, which outperform the state-of-the-art methods. In our pipeline framework, even it adopts only one core pattern per class, the accuracies our model reported still be competitive, which are **85.9%** on Caltech101 and **75.7%** on Caltech256.

V. CONCLUSION

This paper proposed the pipeline framework that used pretrained CNN model for feature extraction and Hopfield

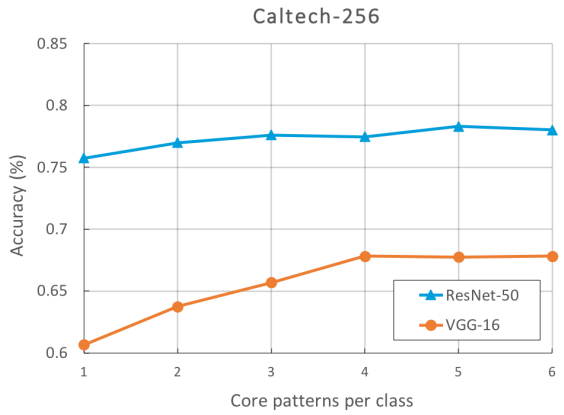

Fig. 4. Impact of number of core patterns per class on Caltech256 dataset.

network as associative memory for memorizing and retrieving patterns while K-means based unsupervised learning is used for core patterns selection. It is also shown in this paper that the associative memory based classifier achieved the state-of-the-art performance in classification, and outperforms other methods simply by the advances in the Hopfield network.

REFERENCES

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[2] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[3] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1134–1142.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[7] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] J. Cervantes, X. Li, W. Yu, and K. Li, "Support vector machine classification for large data sets via minimum enclosing ball clustering," *Neurocomputing*, vol. 71, no. 4, pp. 611–619, 2008.

[10] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn–svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[11] D. Krotov and J. J. Hopfield, "Dense associative memory for pattern recognition," in *Advances in Neural Information Processing Systems*, 2016, pp. 1172–1180.

[12] ——, "Dense associative memory is robust to adversarial inputs," *arXiv preprint arXiv:1701.00939*, 2017.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[18] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[19] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[20] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[21] S. Ke, Y. Zhao, B. Li, Z. Wu, and X. Liu, "Fast image clustering based on convolutional neural network and binary k-means," in *Eighth International Conference on Digital Image Processing (ICDIP 2016)*, vol. 10033. International Society for Optics and Photonics, 2016, p. 100332E.

[22] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic clustering and convolutional neural network for short text categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, 2015, pp. 352–357.

[23] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[24] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3859–3869.

[25] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 44–51.

[26] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang *et al.*, "A generative vision model that trains with high data efficiency and breaks text-based captchas," *Science*, vol. 358, no. 6368, p. eaag2612, 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] F. Sabahi, M. O. Ahmad, and M. Swamy, "Hopfield network-based image retrieval using re-ranking and voting," in *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*. IEEE, 2017, pp. 1–4.

[29] V. M. Ladwani, Y. Vaishnavi, R. Shreyas, B. V. Kumar, N. Harisha, S. Yogesh, P. Shivaganga, and V. Ramasubramanian, "Hopfield net framework for audio search," in *Communications (NCC), 2017 Twenty-third National Conference on*. IEEE, 2017, pp. 1–6.

[30] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[31] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[33] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[34] A. Singh and N. Kingsbury, "Scatternet hybrid deep learning (shdl) network for object classification," *arXiv preprint arXiv:1708.09212*, 2017.

[35] D. Kuang, A. Gittens, and R. Hamid, "Hardware compliant approximate image codes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 924–932.

[36] F. Zhu, Z. Jiang, and L. Shao, "Submodular object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2457–2464.

[37] A. Shaban, H. R. Rabiee, M. Farajtabar, and M. Ghazvininejad, "From local similarity to global coding: An application to image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2794–2801.