ELMNET: FEATURE LEARNING USING EXTREME LEARNING MACHINES

Dongshun Cui, Guang-Bin Huang, L.L. Chamara Kasun, Guanghao Zhang and Wei Han

50 Nanyang Avenue, Nanyang Technological University, Singapore 639798
Emails: {dcui002, egbhuang, chamarakasun, gzhang009, hanwei}@ntu.edu.sg

ABSTRACT

Feature learning is an initial step applied to computer vision tasks and is broadly categorized as: 1) deep feature learning; 2) shallow feature learning. In this paper we focus on shallow feature learning as these algorithms require less computational resources than deep feature learning algorithms. In this paper we propose a shallow feature learning algorithm referred to as Extreme Learning Machine Network (ELMNet). ELMNet is module based neural network consist of feature learning module and a post-processing module. Each feature learning module in ELMNet performs the following operations: 1) patch-based mean removal; 2) ELM auto-encoder (ELM-AE) to learn features. Post-processing module is inserted after the feature learning module and simplifies the features learn by the feature learning modules by hashing and block-wise histogram. Proposed ELMNet outperforms shallow feature learning algorithm PCANet on the MNIST handwritten dataset.

Index Terms— ELMNet, Feature Learning, ELM-AE

1. INTRODUCTION

Computer vision tasks such as object detection [1], object recognition [2], object segmentation [1, 3] and image classification [4] are negatively affected by illumination, occlusion, and deformations. Hence learning features which are robust to illumination, occlusion, and deformations is important to computer vision tasks [5].

Deep feature learning algorithms such as multi-column deep neural network perform better than a human in the traffic sign recognition task. While Pyramid Convolution Neural Network and extreme sparse learning achieve results better than humans in face representation and emotion recognition tasks. However deep feature learning algorithms require a large amount of computational resources and large training time [6, 7]. Ciregan *et al.* [8] proposed a multi-column deep neural network which is a wide and deep artificial neural network architecture, which was claimed could match human performance on tasks like recognition of traffic signs. A Pyramid Convolutional Neural Network was proposed for face representation by adopting a greedy-filter-and-down-sample operation in [9]. Shojaeilangari *et al.* [10] used extreme sparse learning to represent facial images for robust facial emotions recognition and achieved a high accuracy.

In contrast shallow feature learning algorithms require less computational resources and for some tasks achieves performance comparable to deep feature learning algorithms. For example, Coates *et al.* [11] analyzed the feature learning ability of single-layer networks and attained a high performance when the number of hidden nodes and other parameters are pushed to their limits. A method of unsupervised representation learning based on ELM is proposed on in [12], in which, ELM-AE was adopted as the learning unit, and a transferred layer was introduced. Besides, local contrast normalization (LCD) and whitening were employed as preprocessing steps. Chan *et al.* [13] proposed a simple learning architecture named PCA network (PCANet), which is quite intuitive and can be efficiently estimated. PCANet has attained remarkable results. However computing PCA costs a lot of time and resources [14].

Our work is motivated by the recent results in dimension reduction using ELM [14]. In this paper, we investigate an ELM module based neural network for feature learning. Motivated by PCANet architecture, we proposed ELMNet by embedding the ELM auto-encoder (ELM-AE). We also perform binary hashing [15] and block-wise histogram [16] after feature learning. In the following sections, we introduce the related work firstly. Then, we explain the whole feature learning architecture we proposed, particularly in the details of ELMNet. The proposed algorithm is tested with a digit database (MNIST variant [17]), and conclusions are given finally.

2. RELATED WORK

PCANet was introduced as a shallow feature learning algorithm which can be used as a simple baseline for deep feature learning algorithms. We first describe PCANet architecture as the proposed ELMNet is based upon.

PCANet consists of two types of modules: 1) PCA feature learning module; 2) post-processing module. PCANet is created by connecting the PCA feature learning modules and post-processing module as shown in Figure 1.

PCA feature learning module consists of two states: 1) pre-processing; 2) PCA filter convolution. In the pre-

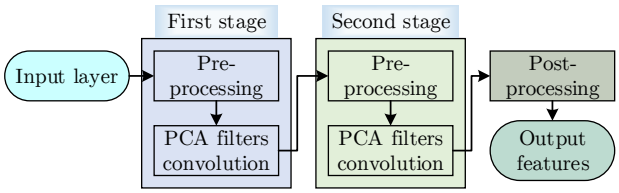

Fig. 1. A schematic of two-stage PCANet.

processing state, sliding window with no stride is used to extract patches and remove the mean as:

$$\bar{\boldsymbol{x}}_k = \boldsymbol{x}_k - \frac{\sum_{i=1}^{n} \boldsymbol{x}_{k_i}}{n}\mathbf{1}, \tag{1}$$

where $\boldsymbol{x}_k$ denotes the k-th patch which contains n pixel values, and $\mathbf{1}$ is an all one vector whose size is the same with $\boldsymbol{x}_k$. All patches are combined together to form a matrix $\boldsymbol{X}$. The objective function of PCANet is to minimize the reconstruction error with a set of filters, which is expressed as

$$\underset{\boldsymbol{V}}{\operatorname{argmin}} \|\boldsymbol{X} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{X}\|_{\mathrm{F}}^2, \tag{2}$$

where $\boldsymbol{V}$ is an orthogonal matrix, and $\|\cdot\|_{\mathrm{F}}^2$ is the Frobenius norm. The solution of the objective function is known as the principal eigenvectors of $\boldsymbol{X}$'s covariance matrix. First l principal vectors (called PCA filters) are chosen as a convolution core to extract the features of the original images.

The second stage is similar to the first stage, and the main difference is that the total number of patches to be processed is L (the number of filters in the first stage) times of the first stage. With the increase of stages, the quantity of patches to be processed becomes larger and larger.

Extreme learning machines (ELM) was proposed along with the interdisciplinary development of effective machine learning theories and techniques in 2006 [18]. ELM has been attracting the increasing attentions from more and more researchers due to the amazing abilities of classification, regression, feature learning, sparse coding, and compression [19]. It has successfully been applied in more and more real industrial applications and usually achieves comparable or better results than many conventional learning algorithms at much fast learning speed [20, 21]. The basic ELM is expressed as:

$$\boldsymbol{\beta} = \boldsymbol{Y}(g(\boldsymbol{W}\boldsymbol{X} + \boldsymbol{B}))^{\dagger}. \tag{3}$$

In Equ 3, $\boldsymbol{X}$ is a $D \times N$ matrix that denotes the input data, where D is the number of features, and N is the number of samples. $\boldsymbol{W}$ is a $N_h \times D$ weight matrix between the input layer and the hidden layer, and the elements in W is randomly initialized. $\boldsymbol{B}$ is a $N_h \times N$ bias matrix of the hidden neurons, where the first column is randomly initialized, and other columns are the duplicates of the first column. $\boldsymbol{Y}$ is the target data, and it is replaced by X for the auto-encoder, that is, $\boldsymbol{Y} = \boldsymbol{X}$. $\boldsymbol{\beta}$ is a $D \times N_h$ weight matrix between output layer and hidden layer. $g(\cdot)$ is an activation function, which can be a Sigmoid function (default), a Sine function, a Radbas function, etc. The $(\cdot)^{\dagger}$ is a Moore-Penrose pseudoinverse operator.

Feature learning with ELM based on auto-encoder (ELM-AE) for big data is introduced in [22]. Recently, dimension reduction with extreme learning machine is analyzed in [14]. In this paper, ELM-AE is adopted as the core function of learning the features of input patches.

Most closely to our work is the hierarchical extreme learning machine (H-ELM) proposed in [12], which also uses ELM-AE to extract local receptive features and the training patches are randomly selected. In our proposed method, we drop out no patches and adopt a full connection network, which simplifies the network and need fewer parameters. Besides, the network architecture of our method is far simpler since we have no whitening pre-processing and the connections between the first layer and last layer.

3. ELMNET FOR FEATURE REPRESENTATION

Motivated by PCANet and ELM-AE, we propose a shallow feature learning algorithm named ELMNet, whose framework is shown in Fig. 2. Post-processing module consists of two steps: 1) binary hashing; 2) block-wise histogram. ELMNet will be introduced in details, which is followed by the binary hashing and block-wise histogram.

Assume the database has N images I_i ($i \in [1, N]$, and the resolution of each image is $W \times H$) and the corresponding labels are Y_i. The target of learning a more efficient representation $R(\cdot)$of I_i is to minimize the loss function $\sum_{i=1}^{N} \|C(R(I_i) - Y_i)\|$, where $C(\cdot)$ denotes a classifier. In our work, support vector machine (SVM, [23]) is adopted.

As we have introduced, the proposed ELMNet concludes the ELM feature learning modules and a post-processing module. We first describe the ELM feature learning module. To remove the influence of illumination, we implement a patch operation of subtracting the mean. Assume the width and the height of the patch is w_p and h_p respectively, then we convert each sliding $w_p \times h_p$ window of the image I_i into a column. All the columns construct a matrix X_i by subtracting their means one by one. The width of X_i is $(W - w_p + 1) \times (H - h_p + 1)$ and the height is $w_p \times h_p$.

An auto-encoder based on ELM is implemented to learn the features from all the X_i, which are denoted as X. We have introduced the details of ELM in Section 2. Three differences between ELM-AE and the basic ELM are listed as below:

- The input weight $\boldsymbol{W}$ and the bias $\boldsymbol{B}$ are both orthogonal matrixes. That is, we have $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$ and $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}$ with regard to ELM-AE.

- The output weight $\boldsymbol{\beta}$ is an orthogonal matrix. That is, $\boldsymbol{\beta}^T\boldsymbol{\beta} = \boldsymbol{I}$.

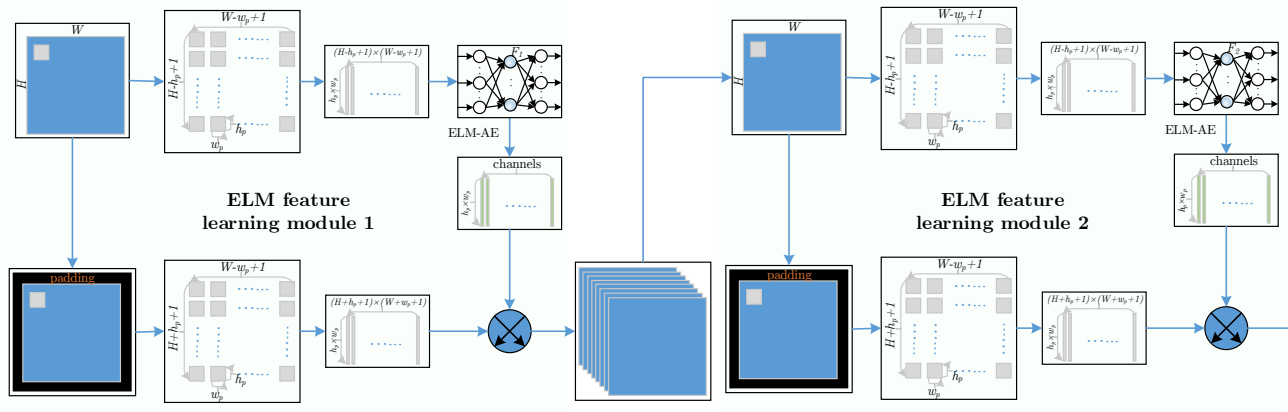

Fig. 2. The framework of ELMNet

- The output of the ELM-AE network equals to the input.

Here, the number of hidden nodes N_{F_k} corresponds to the the number of convolution filters F_k that we use in the k-th stage of the network. The filters are named as "channels" that are presented in Fig. 2. N_{F_k} can be learned by optimization, and empirically it can be assigning a number less than 10 due to the limitation of the resources.

With the filters we get from ELM-AE, we can compute the outputs of the patches (overall denoted as X) from the raw samples. The outputs are used to represent the patterns of the raw samples. To keep the information of the edges, we pad each sample with zeros in its surroundings. In our experiments, we pad $w-1$ columns and $h-1$ rows of zeros before the first element and after the last element along the horizontal and vertical directions respectively. After that, the same process of mean-removal patch operation is performed. Finally, we can achieve all the patches X of one sample, whose output O in the k-th stage of the network is expressed as

$$O = F_k * X, \tag{4}$$

where F_k is the filter banks we have achieved from the ELM-AE, and X indicates the input samples that have been processed with zeros-padding and mean-removal operations. The output O have N_{F_k} layers for each X in the k-th stage.

The following stages of the network are similar to the first stage, and the total amount of stages is represented as S. In the proposed algorithm, We use two ELM feature learning modules in ELMNet.

Post-processing module of ELMNet uses binary hashing and block wise histogram by reason of transferring the output of the ELMNet into a more efficient form. From the explanation of ELM, we can achieve N_F output images for each sample, and all the output images are reshaped into the same size with the input images. Here, we have $N_F = \prod_{k=1}^{S} N_{F_k}$.

There are $N_{F_{S-1}}$ outputs Ψ for one raw sample in the stage $S-1$, and each $\psi_\sharp \in \Psi$ ($\sharp \in [1, N_{F_{S-1}}]$) is processed by F_S filters in the stage S and generates F_S corresponding outputs Θ. Each $\theta_\hbar \in \Theta$ ($\hbar \in [1, F_S]$) is converted to a binary data matrix $\theta'_\hbar$ with a zero threshold. A hashing processing is performed on all the θ', and the operation is expressed as

$$T_\sharp = \sum_{\hbar=1}^{F_S} 2^{F_S-\hbar}\theta'_\hbar. \tag{5}$$

The block-wise histograms are computed for all the blocks we achieved with the sliding window technique. Assume the size of the block is represented as $w_b \times h_b$ and the overlap rate of the blocks is denoted as ϵ, then the stride size of the sliding windows is $\langle(1-\epsilon)w_b \times (1-\epsilon)h_b\rangle$ ($\langle\cdot\rangle$ is a rounding operator). We go through the entire $T_\sharp$ with this stride and compute the block-wise histogram for each block and combine them together to form the final features.

4. EXPERIMENTS

We evaluate our proposed algorithm on a variant of the commonly used object classification database: MNIST (Mixed National Institute of Standards and Technology) database. For the original MNIST database, there are 60000 training samples and 10000 testing samples. While for the MNIST variant we used here, there are 10000 training samples, 2000 samples to valid the parameters, and 50000 samples to test. The total number of each digit are shown in Fig. 3, from which we know that these ten digits have a uniform distribution for both the training and test samples.

PCANet is one of the state-of-the-art methods on this database, and the architecture of our proposed method ELMNet is similar to PCANet, so we mainly compare our method with PCANet in our experiments to manifest the effectiveness of ELMNet. We will introduce the parameters setting and results of key steps in the following paragraphs.

The width W and height H of the samples are both equal to 28 in our experiments. To have a better comparison with

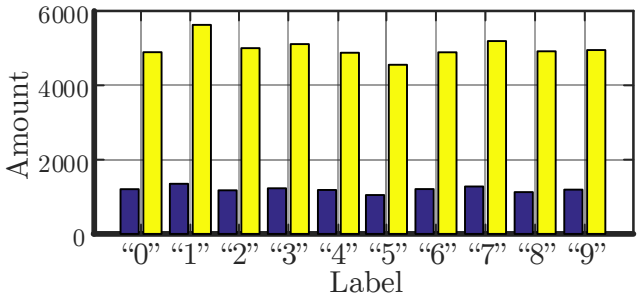

Fig. 3. Amounts of the 10 digits in the MNIST variant. The blue and yellow bars represent the training and test samples respectively.

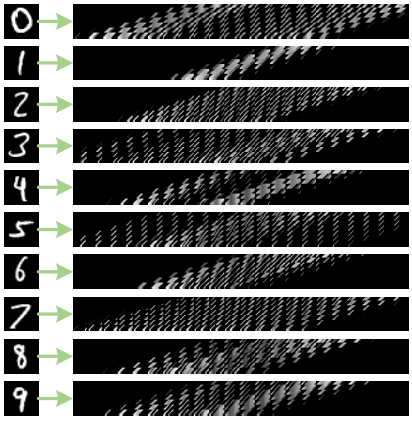

Fig. 4. Results of the mean-removal patch operation.

PCANet, we configure our proposed network with the same parameters to the optimized PCANet (details to vary the parameters can be found in [13]). We also use two-stage ($S = 2$) network and the number of filters N_{F_1} and N_{F_2} are set to eight ((same amounts of filters used in PCANet). The width w_p and height h_p of the patches are both assigned as seven. For the blocks to compute the block-wise histogram, the width w_b and height h_b are also both set to seven. The overlap rate ϵ is increased monotonically from 0 to 0.9 by a step of 0.1.

We have randomly selected ten samples for the 0 9 digits from the MNIST variant database. Each 28×28 image after the 7×7 patch mean-removal process output a 16×484 image, where $16 = (28/7)^2$ and $484 = (28 - 7 + 1)^2$. Their results are shown in Fig. 4.

We have analyzed the impact of the overlap rate of the blocks in the last step on the error rate of ELMNet and PCANet on the MNIST variant database. The error rate is defined as $\frac{\sum_{i=1}^{N}[P_i \neq T_i]}{N}$, where $[\star]$ is defined to be 1 if $\star$ is true and 0 if $\star$ is false. When we increase the overlap rate ϵ, the error rates of ELMNet and PCANet are declined overall. The detail results of PCANet and ELMNet with regards to different overlap rates are shown in table 1. From the comparison, we know that our proposed method has got a 0.980% error rate, and achieved lower error rates for the nine out of ten overlap rates and it performs 4.18% (the average of the improvement rates) better than PCANet.

Table 1. Results of PCANet and ELMNet on the MNIST variant dataset.

Overlap Rate (ϵ)	Error Rate (%)		Improvement (%)
	PCANet	ELMNet	
0	1.150	**1.082**	+5.91
0.1	1.118	**1.052**	+5.90
0.2	1.118	**1.022**	+8.59
0.3	**1.066**	1.094	-2.63
0.4	1.072	**1.010**	+5.78
0.5	1.072	**1.034**	+3.54
0.6	1.020	**0.990**	+2.94
0.7	1.050	**1.040**	+0.95
0.8	1.058	**1.022**	+3.40
0.9	1.058	**0.980**	+7.37

According to the results from [12], the error rates are 1.05% and 1% for hierarchical ELM network with whitening and without whitening respectively, which are also higher than our proposed algorithm.

5. CONCLUSION

In this paper, we proposed a novel feature representation method named ELMNet using ELM-AE. ELMNet is a multi-stage feature learning network, which contains patch mean-removal process and ELM-AE in each stage. The former removes the influence of illumination, while the later learns the representation of the input patches, filters the raw samples, and outputs the practical features. Two post-processing steps (binary hashing and block-wise histogram) are needed to simplify the final output features.

The experiments on the MNIST variant database prove the effectiveness of our proposed method since ELMNet achieves impressive results by learning a better representation of the input samples. Different overlap rates have been analyzed, and the error rate of ELMNet is lower than that of PCANet and hierarchical ELM.

6. ACKNOWLEDGEMENT

The authors thank Delta Group for the support of the research.

7. REFERENCES

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra 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.

[2] Manuel Blum, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller, "A learned feature descriptor for object recognition in rgb-d data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1298–1303.

[3] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[4] Yongzhen Huang, Zifeng Wu, Liang Wang, and Tieniu Tan, "Feature coding in image classification: A comprehensive study," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 3, pp. 493–506, 2014.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

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

[7] Michael Blot, Matthieu Cord, and Nicolas Thome, "Max-min convolutional neural networks for image classification," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, pp. 3678–3682.

[8] Dan Ciregan, Ueli Meier, and Jürgen 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.

[9] Haoqiang Fan, Zhimin Cao, Yuning Jiang, Qi Yin, and Chinchilla Doudou, "Learning deep face representation," *arXiv preprint arXiv:1403.2802*, 2014.

[10] Seyedehsamaneh Shojaeilangari, Wei-Yun Yau, Karthik Nandakumar, Jun Li, and Eam Khwang Teoh, "Robust representation and recognition of facial emotions using extreme sparse learning," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2140–2152, 2015.

[11] Adam Coates, Honglak Lee, and Andrew Y Ng, "An analysis of single-layer networks in unsupervised feature learning," *Journal of Machine Learning Research*, vol. 1001, no. 48109, pp. 2, 2011.

[12] Wentao Zhu, Jun Miao, Laiyun Qing, and Guang-Bin Huang, "Hierarchical extreme learning machine for unsupervised representation learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[13] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma, "Pcanet: A simple deep learning baseline for image classification?," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[14] Liyanaarachchi Lekamalage Chamara Kasun, Yan Yang, Guang-Bin Huang, and Zhengyou Zhang, "Dimension reduction with extreme learning machine," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3906–3918, 2016.

[15] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou, "Deep hashing for compact binary codes learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2475–2483.

[16] Yanjun Qi, Alexander Hauptmann, and Ting Liu, "Supervised classification for video shot segmentation," in *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*. IEEE, 2003, vol. 2, pp. II–689.

[17] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[18] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, 2006.

[19] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.

[20] Guang-Bin Huang and Lei Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2007.

[21] Guang-Bin Huang and Lei Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16, pp. 3460–3468, 2008.

[22] Guang-Bin Huang Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou and Chi Man Vong, "Representation learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[23] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.