Simplicity of Kmeans versus Deepness of Deep Learning: A Case of Unsupervised Feature Learning with Limited Data

Murat Dundar*, Qiang Kou†, Baichuan Zhang*, Yicheng He* and Bartek Rajwa‡
*Department of Computer and Information Sciences
Indiana University - Purdue University, Indianapolis, IN 46202
Email: dundar@cs.iupui.edu
†Department of Biohealth Informatics
Indiana University - Purdue University, Indianapolis, IN 46202
‡Bindley Bioscience Center
Purdue University, W. Lafayette, IN 47907

***Abstract*—We study a bio-detection application as a case study to demonstrate that Kmeans–based unsupervised feature learning can be a simple yet effective alternative to deep learning techniques for small data sets with limited intra- as well as inter-class diversity. We investigate the effect on the classifier performance of data augmentation as well as feature extraction with multiple patch sizes and at different image scales. Our data set includes 1833 images from four different classes of bacteria, each bacterial culture captured at three different wavelengths and overall data collected during a three-day period. The limited number and diversity of images present, potential random effects across multiple days, and the multi-mode nature of class distributions pose a challenging setting for representation learning. Using images collected on the first day for training, on the second day for validation, and on the third day for testing Kmeans–based representation learning achieves 97% classification accuracy on the test data. This compares very favorably to 56% accuracy achieved by deep learning and 74% accuracy achieved by handcrafted features. Our results suggest that data augmentation or dropping connections between units offers little help for deep-learning algorithms, whereas significant boost can be achieved by Kmeans–based representation learning by augmenting data and by concatenating features obtained at multiple patch sizes or image scales.**

I. Introduction

Deep-learning (DL) algorithms harness large amount of data to extract features at multiple levels of abstraction by nonlinearly mapping raw data onto higher-level semantic descriptors potentially interpretable by human beings. DL algorithms have recently shown great promise in many benchmark image- and speech-recognition challenges, outperforming state-of-the-art techniques by a large margin and attracting tremendous interest among researchers and practitioners alike.

Although significant effort has been devoted to scale DL methods by addressing important problems in back propagation and network initialization so they are applicable for large data sets, only limited effort has gone into understanding the potential pitfalls facing DL algorithms. The most important limitation of DL that is yet to be fully addressed is possible overfitting and lack of generalizability when the method is used with limited data. Although randomly dropping connections between units and/or augmenting data may offer some help, there is little evidence suggesting that DL algorithms work well with limited data.

Most object-classification tasks employed as benchmarks involve natural images (a domain in which DL has proved very effective), a plethora of which can be found on the web. However, there are other important imaging domains, for instance in life sciences, where the amount of data produced is inherently limited. Unlike natural images, successful classification of these data requires identification of often complex yet subtle image descriptors that may pose a serious challenge even for domain experts.

In this study, we use a real biological data set representing a classification task from the field of label-free biodetection/biosurveillance. The complete set contains only 1833 images. We compare Kmeans–based representation learning against DL algorithms and handcrafted feature-extraction techniques currently employed for these types of data. We study the effect of patch size, image scale, number of descriptors, and data augmentation on classifier performance. The results suggest that when faced with the problem of limited data, Kmeans–based unsupervised feature learning is a powerful alternative to deep-learning algorithms as well as to conventional techniques that rely on handcrafted features.

II. Related Work

The term "unsupervised feature learning" describes the task of utilizing unlabeled data to construct representations of samples as feature vectors. It has wide applicability in computer vision, speech recognition and audio classification, and natural-language processing. Methodologies in this area can be broadly categorized into single-layer and deep architectures.

Owing to speed and scalability the Kmeans–based approach representing a single-layer unsupervised feature-learning framework is a popular choice in image-classification [1], [2], [3], [4] and object-recognition [5] tasks. Like Kmeans, sparse coding [6] has also attracted a great deal of interest as a single-layer unsupervised feature-learning method. Although

sparse coding is computationally more expensive than Kmeans, it can be more effective with higher-dimensional data.

DL algorithms that stack layers of features to build deep architectures have been successfully employed for unsupervised feature learning. The major drawbacks of these methods are their complexity and the computational expense involved in training and tuning. Many of these algorithms, in addition to determining the network architecture, require adjustment of a large number of parameters such as learning rate, momentum, sparsity penalties, and weight decay. Finding the right combination of these parameters, especially when data are limited, is not trivial and could be considered more of an art than science.

In this study we adopt the Kmeans–based representation-learning framework discussed in [1] and characterize the applicability of this approach to biological images by investigating the effect of patch size, number of descriptors, image scale, and data augmentation.

III. Bacteria Colony Classification as a Case Study

The majority of tools for pathogen detection and classification are based on physiological properties or genetic markers of microorganisms. However, concerns about biosecurity have led to an enormous interest among the scientific community and government agencies in devising truly reagentless biosensors that would operate utilizing the intrinsic properties of samples without the need for sensing and reporting biochemistry. The phenotypic biophysical sensors are closest to realizing this goal. Whether they are based on spectroscopy or measurement of elastic light scattering, the phenotypic methods rely on a library of signatures/fingerprints generated for different bacterial classes to subsequently detect and classify future samples of unknown nature.

Our study uses laser-light forward-scatter patterns formed by bacterial colonies and represented as gray-level images. The data from four different genera of bacteria (*E. coli, Listeria, Salmonella*, and *Staphylococcus*) are collected by the BARDOT (BActeria Rapid Detection using Optical scattering Technology) system [7]. This instrument employs lasers that illuminate the centers of individual bacteria colonies at three different wavelengths to create forward-scatter signatures dependent on the colony structure and subsequently analyzed for classification. For each colony three scatter patterns, one for each wavelength, were captured. To ensure that the classification system is robust to daily variations in experimental settings (amount of light present, growth rate of colonies, agar media used, etc.) the images were collected during a 3-day period. A total of 1833 bitmaps of size 1280 by 1024 were acquired (600/day and 450/genus) for analysis. Apart from containing a relatively low number of images with limited diversity, this data set is challenging to analyze by automated techniques for three other reasons. First, owing to the three different wavelengths at which each sample is interrogated, the classes exhibit multi-modal distributions. Second, random effects due to daily variations in experimental settings can be significant. Third, unlike standard object-detection problems involving natural images, the descriptive features across different bacteria classes can be subtle and not easy to define. Sample images from four bacteria classes captured at three different wavelengths are shown in Figure 1.

IV. Methods

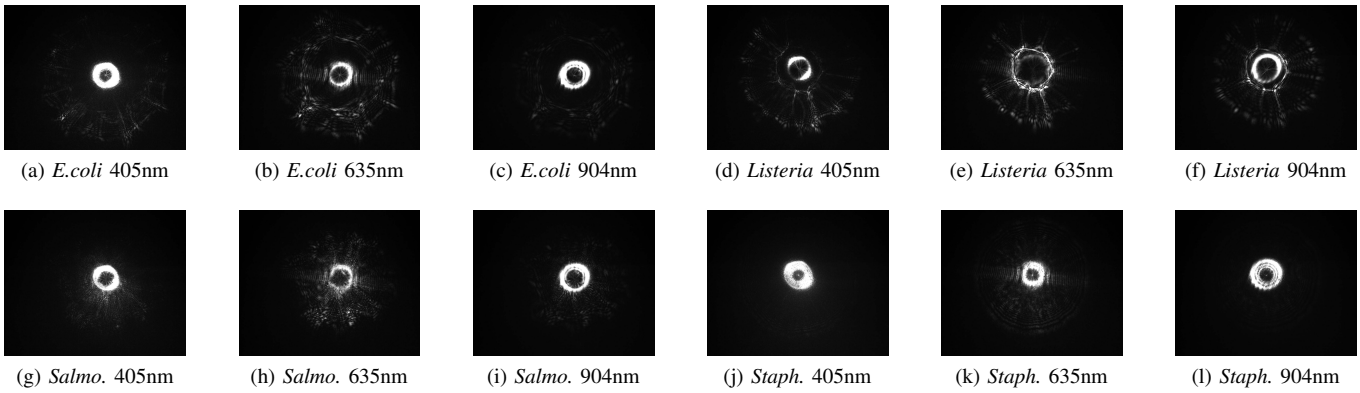
A. Unsupervised Feature Learning by Kmeans

In this section we discuss the three main stages involved in Kmeans–based representation learning. Before these stages are executed, the image scale (s), the number of descriptors (k) and the patch size (w) must be designated. The first stage involves running the Kmeans algorithm with the image patches obtained from all available images to learn image descriptors. The centers of the k clusters obtained are treated as the k image descriptors. For a data set with n images with each image having r rows and c columns, and for a patch size of w, there will be $n(r-w+1)(c-w+1)$ many patches for running the Kmeans algorithm. For larger data sets this number can grow rapidly, so subsampling may be required. Normalization of image patches plays a key role in the overall results. Although the work in [1] recommends whitening for natural images, we found that standardization, i.e., pre-processing each patch for zero mean and unit variance, is a more effective approach for these bacteria data sets.

The second stage involves obtaining image representations for each scatter pattern in the data set by feature encoding. For each patch in the image, the Euclidean distance from the patch to each of the learned centroids is computed. Based on these distances the patch is either hard-assigned to the closest centroid or soft-assigned to multiple closer centroids. Although soft assignment can be done in multiple ways as discussed in [4] we used the triangle encoding technique proposed in [1] for its effectiveness and speed. Once assignment vectors for all patches in an image are obtained, an image-level representation vector for an image is obtained, either by mean- or sum-pooling over all assignment vectors. Throughout our experiments we used sum pooling. During feature encoding the patches with all pixels having the same value, i.e., patches with standard deviation equal to zero, are ignored.

The third stage involves classification in which the feature vectors obtained in the second stage are used. We split our data set into three sets according to the day on which the data set was collected. All images collected on day one were used for training (606 images), on day two for validation (612 images), and on day three for final testing (615 images). For classification we used the L2-regularized L2-loss linear support vector machine available in the Liblinear package [8]. Each feature in the feature matrix is preprocessed to have zero mean and unit variance. The main steps involved in K means–based representation learning used in this study are outlined below.

1) Set image scale s, patch size w, and number of descriptors k.
2) For each image in the data set extract and normalize patches, and reshape them into w^2 by 1 vectors. Pool together all patch vectors, subsample if needed, and run the Kmeans algorithm to cluster the data into k clusters.
3) For each patch in each image use triangle encoding to obtain $k-$dimensional assignment vectors and sum-pool assignment vectors belonging to the same image to obtain feature vectors representing images.
4) Train a classifier using the image subset collected on day one, tune classifier parameters to optimize accuracy on the image subset collected on day two,



(a) *E.coli* 405nm (b) *E.coli* 635nm (c) *E.coli* 904nm (d) *Listeria* 405nm (e) *Listeria* 635nm (f) *Listeria* 904nm

(g) *Salmo.* 405nm (h) *Salmo.* 635nm (i) *Salmo.* 904nm (j) *Staph.* 405nm (k) *Staph.* 635nm (l) *Staph.* 904nm

Fig. 1. Sample images from four bacteria classes captured at three different wavelengths.

and perform a one-shot testing on the image subset collected on day three; report classification accuracies on day-two and -three image subsets as validation and test accuracies, respectively.

B. Unsupervised Feature Learning by Deep Architectures

In this section we will briefly review three deep-learning algorithms used in our experiments for comparison: convolutional neural nets, deep neural nets with dropout, and stacked denoising autoencoders.

1) Convolutional neural nets: Convolutional neural networks (CNNs) are ideally suited for recognizing visual patterns directly from pixel data [9]. The network architecture is built by repeated stacking of doublet layers of a convolutional layer followed by a subsampling layer which are connected to a multi-layer perceptron in the end. Each convolutional layer of CNN exploits local spatial correlations among pixels, but as more of these layers are stacked together increasingly global and highly nonlinear image properties can be captured. CNNs use the concept of feature maps, which are replicated groups of units introduced to detect image descriptors regardless of their spatial position. CNNs also use the idea of max-pooling to implicitly down-sample images. The size of the receptive field, the number of feature maps and units in each layer, the size of the region over which max-pooling is done, the specific data-preprocessing technique used, the learning rate, and the learning rate decay are among the tunable parameters required for successful implementation of a CNN. Owing to the large number of parameters, exhaustive tuning of a CNN may become impractical even for limited data sets. The general practice is to start with a configuration that has proved effective in a similar problem and adapt parameter values to the current problem by taking into consideration the type and size of images involved.

2) Deep neural nets with dropout: Deep neural nets with dropout (DNNs) [10] is a technique motivated by model averaging, a concept that has proved very effective with limited data. Model diversity is critical for model averaging to be effective, and this can be achieved either by training multiple neural nets with different architectures or by using different data sets for training. The former is prohibitively expensive and the latter requires partitioning already limited data into multiple subsets. Dropout is a technique that gets around these limitations by dropping units in a network with a certain probability, so every time a training sample is presented to the network a slightly different architecture is used. In addition to standard parameters involved in the training of neural nets, in a dropout network the probabilities of dropping units in each layer must be tuned as well.

3) Stacked denoising autoencoders: An autoencoder is a neural network that is trained to learn an approximation of the identity function so that the output matches the input as closely as possible. In a sparse autoencoder hidden layers are forced to have a smaller number of units than the input layer in order to obtain a compressed representation of the input. A stacked denoising autoencoder (SDA) [11] is an extension of the sparse autoencoder that applies stochastic noise to the input so that a more robust representation of the input can be achieved. In other words, an SDA reconstructs the input from a corrupted version of it by exploiting statistical dependencies between input variables. The SDA is trained with all available data during the pretraining phase. During the classifier training phase the network is updated, with the decoder replaced by a softmax classifier.

C. Handcrafted Feature Extraction

The elastic-light-scatter colony classifiers implemented in BARDOT currently use two groups of features: pseudo-Zernike moments (PZM) [12] and Haralick texture features [13]. To compute the Zernike moments of a given image, the center of the image is taken as the origin and pixel coordinates are mapped to the range of the unit circle. Rotational invariance is obtained by using the magnitudes of the pseudo-Zernike moments as features. Haralick texture descriptors are derived from the gray-level co-occurrence matrix (GLCM). The GLCMs are used to extract twelve low- and high-frequency texture properties. We used the mean and the range of these twelve texture descriptors as features.

V. Experiments

In this section we present results of our experiments with Kmeans–based representation learning, deep-learning techniques, and handcrafted feature extraction. In all experiments classifiers were trained using images collected on day one (training set), classifier parameters were tuned to optimize performance on images collected on day two (validation set), and final models were tested on images collected on day three (testing set). We report classification accuracies for both validation and test sets.

TABLE I. CLASSIFICATION ACCURACIES ACHIEVED FOR DIFFERENT IMAGE DOWNSCALE RATES AND PATCH SIZES. ALL NUMBERS ARE IN PERCENT. FOR EACH PAIR OF s AND w THE TOP ROW INCLUDES VALIDATION ACCURACIES AND THE BOTTOM ROW INCLUDES TEST ACCURACIES.

	No data augmentation				Training set augmented			
	k				k			
	100	250	500	1000	100	250	500	1000
$s = 10\%$, $w = 6$	89.71	94.28	94.28	93.95	80.23	85.78	90.20	94.12
	48.46	52.52	54.80	43.74	68.29	82.11	87.15	85.37
$s = 10\%$, $w = 10$	93.46	93.46	95.10	95.26	88.40	91.83	95.42	95.92
	53.98	53.01	55.12	57.07	81.14	93.98	94.47	94.63
$s = 5\%$, $w = 6$	92.16	92.65	93.46	93.46	86.27	93.63	95.26	96.41
	51.71	59.51	57.72	55.45	84.39	89.27	94.80	95.45
$s = 5\%$, $w = 10$	79.74	80.88	76.63	83.33	85.46	95.75	95.59	96.57
	60.49	66.18	70.89	73.50	80.16	85.2	92.52	92.68

A. Experiments with K means–based Representation Learning

For Kmeans–based representation learning, we used multiple choices for patch size, $w = \{6, 10\}$, image downscale rate, $s = \{5\%, 10\%\}$, and number of descriptors $k = \{100, 250, 500, 1000\}$. We investigated individual as well as joint effects of these parameters on the overall classifier performance with and without data augmentation. During descriptor learning the Kmeans algorithm was run on a subset of the data obtained by subsampling 100 patches per image. However, during feature encoding all patches in an image were used to obtain representation vectors.

1) Kmeans without data augmentation: With this set of experiments we wanted to see the effects of number of descriptors, patch size, and image downscale rate on the overall classifier performance when no data augmentation was performed. All results are included in the first part of Table I. When we downscaled each image to 10% of its original size we obtained classification accuracies of over 90% on the validation set but only ∼50% on the test set, suggesting severe overfitting of the validation set irrespective of the patch size (w) and the number of descriptors (k) used.

When we repeated this experiment with images scaled down to 5% of their original sizes, results were mostly similar to those of the previous experiment. However, with a patch size of $w = 10$ slightly better accuracies were obtained on the test data compared to $w = 6$ and test accuracies trended higher as k increased.

We also tried combining different pairs of patch sizes and image scales by concatenating corresponding feature vectors, but no significant improvement in validation and test accuracies were obtained. Overall, without data augmentation we achieved a moderately high classification accuracy on the validation set that does not hold on the test set. Although there are specific settings for which the classifier can generalize slightly better, these results suggest that with only 606 images available for training overfitting is almost unavoidable and using a different number of descriptors, patch sizes, or image scales and their combinations does not offer much help.

2) Kmeans with data augmentation: We repeated the previous set of experiments, this time augmenting the original data tenfold with each image flipped up-down, left-right, and rotated by 45-degree steps. We split the experiments in this section into two according to whether or not test and validation data sets were also augmented along with the training data set in order to see the effect of voting on the overall classifier performance. Max voting is used to assign labels to each image in the validation and test sets. The results of experiments when only the training set was augmented are included in the second part of Table I. Based on these results, the following observations can be made. Although validation accuracies did not improve much compared to the settings without data augmentation, test accuracies improved significantly, reaching over 95% accuracy for the setting $k = 1000$, $s = 5\%$, and $w = 6$. For most settings, test and validation accuracies were comparable, suggesting significantly improved generalizability of the classifier. For all pairs of (s, w) both validation and test accuracies trended higher as k increased. Next, we augmented validation and test sets in addition to the training set in order to see the effect of voting on the overall classifier performance. Results of these experiments are shown in Table II along with the results of experiments when only the training set was augmented. Although there was no obvious trend suggesting that voting helps improve classifier generalizability for all settings, it did improve classifier accuracy up to four percentage points for ($s = 5\%$, $w = 10$). For this setting, validation and test accuracies reached 98.69% and 96.75%, respectively, for $k = 1000$.

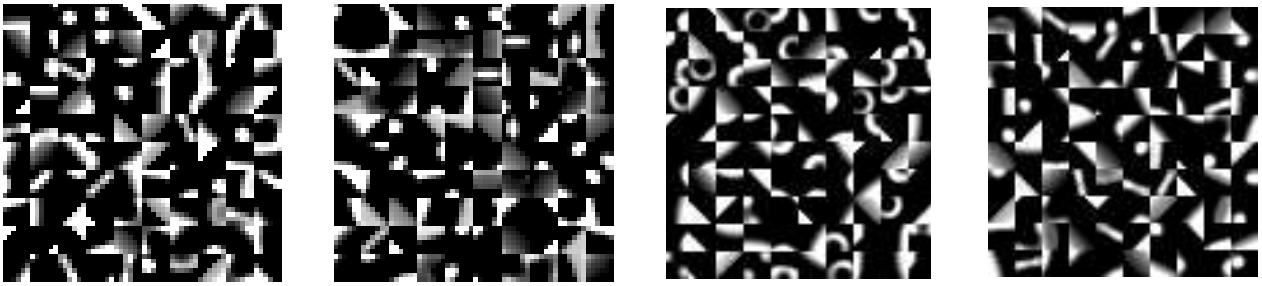
3) K-means by concatenating feature vectors obtained at different patch sizes and image scales: Finally, we try concatenating feature vectors for different (s, w) pairs to see if combining feature vectors improve classifier accuracy. For this experiment we use feature vectors obtained with all data sets augmented. Results of these experiments are included in Table III. These results suggest that significant benefit can be gained by feature concatenation. When we concatenate feature vectors corresponding to patch sizes $w = 6$ and $w = 10$ obtained with images downscaled to 5% test accuracy reach 97.89% for $k = 1000$. Perhaps what is most interesting about feature concatenation is the last two rows of Table III that show impressive validation and test accuracies of 99.02% and 96.42%, respectively, when feature vectors corresponding to $k = 100$ are concatenated for all possible pairs of s and w. Using as few as 400 descriptors extracted for multiple pairs of image scale and patch size we can replicate classification accuracy obtained with 1000 descriptors extracted for a single pair of image scale and patch size. This result show that diverse set of descriptors obtained by combining multiple image scales and patch sizes can be more effective for classification. Image descriptors obtained for $k = 100$ for all four possible pairs of (w,s) are shown in Fig. 2.

B. Experiments with deep-learning techniques

For experiments with deep-learning techniques we used the Theano [14] implementations of CNNs, DNNs with dropout, and SDA. As with the Kmeans experiment, we ran these models with two different image scales ($s = 5\%$, $s = 10\%$). Each

TABLE II. CLASSIFICATION ACCURACIES ACHIEVED FOR DIFFERENT IMAGE DOWNSCALE RATES AND PATCH SIZES WITH ALL DATA SETS AUGMENTED.

	Training set augmented				All sets augmented			
	k				k			
	100	250	500	1000	100	250	500	1000
$s = 10\%$, $w = 6$	80.23	85.78	90.20	94.12	81.70	86.27	91.34	93.14
	68.29	82.11	87.15	85.37	71.71	83.90	84.88	86.67
$s = 10\%$, $w = 10$	88.40	91.83	95.42	95.92	92.97	95.10	96.57	96.90
	81.14	93.98	94.47	94.63	83.74	85.69	93.17	90.89
$s = 5\%$, $w = 6$	86.27	93.63	95.26	96.41	90.03	94.77	96.57	98.20
	84.39	89.27	94.80	95.45	89.11	89.59	94.96	93.82
$s = 5\%$, $w = 10$	85.46	95.75	95.59	96.57	91.34	97.55	97.88	98.69
	80.16	85.2	92.52	92.68	84.07	87.64	94.31	96.75



(a) $s = 5\%$, $w = 6$ (b) $s = 10\%$,$w = 6$ (c) $s = 5\%$,$w = 10$ (d) $s = 10\%$,$w = 10$

Fig. 2. One hundred descriptors obtained by K means–based representation learning with different settings after data augmentation. (a) $s = 5\%$, $w = 6$ (b) $s = 10\%$, $w = 6$ (c) $s = 5\%$, $w = 10$ (d) $s = 10\%$, $w = 10$

TABLE III. CLASSIFICATION ACCURACIES ACHIEVED BY CONCATENATING FEATURE VECTORS OBTAINED FOR DIFFERENT PAIRS OF IMAGE DOWNSCALE RATE AND PATCH SIZE.

Concatenations				k			
				200	500	1000	2000
$s = 5\%$	$s = 5\%$			96.57	98.04	97.88	98.53
$w = 6$	$w = 10$			94.96	93.98	97.24	97.89
$s = 10\%$	$s = 10\%$			91.83	92.16	95.92	93.79
$w = 6$	$w = 10$			86.34	87.80	92.20	88.13
$s = 10\%$	$s = 5\%$			94.93	95.75	96.57	97.88
$w = 6$	$w = 10$			89.27	92.36	93.82	95.61
$s = 10\%$	$s = 5\%$			97.06	97.55	98.53	98.53
$w = 10$	$w = 6$			91.38	92.85	93.66	96.91
				k			
				400	1000	2000	4000
$s = 5\%$	$s = 5\%$	$s = 10\%$	$s = 10\%$	99.02	99.18	97.88	97.71
$w = 6$	$w = 10$	$w = 6$	$w = 10$	96.42	94.47	96.91	97.07

image was reshaped into a feature vector and preprocessed to have zero mean and unit variance for CNN and DNN but scaled to between zero and one for SDA. Model parameters are coarsely tuned to optimize accuracy on the validation set. For CNNs we used a receptive field size of 8 by 8, feature maps of 20 and 50 for the first and second convolutional layers, respectively, a batch size of 10, and a learning rate of 0.01. For DNN with dropout we used a dropout probability of 0.2 for the input layer and 0.5 for the hidden layers, 1200 units in each hidden layer, a batch size of 100, a learning rate of 1, and a decay rate of 0.998. For SDA we used 2000, 1000, and 500 units for the first, second, and third hidden layer, respectively. We used a corruption level of 0.1 for all three layers, a batch size of 5, and learning rates of 0.01 and 0.1 for pretraining and fine tuning, respectively. The results of experiments with deep-learning techniques are shown in Table IV. It is interesting to note that all three DL techniques generated similar results, with SDA producing the highest test accuracy (56.43%) and CNN the highest validation accuracy (73.93%), both far below the validation (99.18%) and test accuracies (97.89%) achieved by Kmeans–based representation learning.

TABLE IV. CLASSIFICATION ACCURACIES ACHIEVED BY DEEP-LEARNING TECHNIQUES FOR DIFFERENT IMAGE DOWNSCALE RATES WITH AND WITHOUT DATA AUGMENTATION.

	No data augmentation			Data augmentation		
	CNN	DNN	SDA	CNN	DNN	SDA
$s = 5\%$	70.82	66.50	64.92	72.95	68.95	69.02
	52.14	45.69	30.75	51.80	49.27	56.43
$s = 10\%$	73.93	65.85	65.58	71.80	72.36	68.70
	40.50	32.52	40.98	49.50	45.37	51.87

C. Experiments with handcrafted features

For each image we used a total of 78 handcrafted features, which were extracted using a combination of Zernike moments and Haralick texture descriptors. These are standard features computed for each bacterial scatter pattern collected by the BARDOT system and have been demonstrated to be quite effective in previously reported classification [7]. When we trained a linear SVM using the training set and tuned the regularization parameter to optimize the accuracy on the validation set in the same way as for other experiments, we achieved accuracies of 78.43% and 64.39% on the validation and test sets, respectively. With a greedy feature subset selection these results improved to 90.85% for validation and to 74.15%

for the test set. Although these results are better than those obtained by deep-learning techniques, they are not comparable with the near-perfect accuracies obtained by K means–based representation learning. We believe that the high sensitivity of Zernike moments and Haralick texture descriptors to random effects makes these features less ideal for classification with multi-day data sets.

VI. Conclusions

The biodetection application considered in this study is interesting on many fronts: the limited number of images available, data collection spread over three days, multi-modal class distributions, and the limited image diversity within as well as between classes. We believe that this is a unique setting for testing unsupervised feature-learning techniques. The fact that deep-learning techniques did not perform well on this application is not a great surprise, as these techniques rely on complex networks that are highly prone to overfitting when data are limited. Nonetheless, the almost negligible improvement recorded in classifier accuracy when data were augmented tenfold presents an interesting case suggesting that deep learning is not a remedy for every image-classification task. We believe that the most interesting result of this study is the robustness shown by Kmeans–based representation learning on this challenging task. Although Kmeans–based representation learning uses a single layer of descriptors during feature encoding, by using multiple image scales and patch sizes one can extract descriptors at different levels of abstraction, mimicking the deep-learning behavior of multi-layer networks with improved stability. Results of our experiments also suggest that Kmeans–based representation learning can better tolerate random effects compared to handcrafted features, which seem to be more sensitive to potential variations in experimental conditions. As a follow-up study we would like to test deep-learning techniques on a similar problem containing many more classes and images to see if the performance of deep-learning techniques improves as we increase the data size or if the nature of the classification problem puts an upper bound on the performance of these techniques, at which point Kmeans–based representation learning can be selected as a better alternative.

The bacteria data set used in this study is available on the web [1].

Acknowledgment

This research was sponsored by the National Science Foundation (NSF) under Grant Number IIS-1252648 (CAREER) and by USDA-ARS project number 8072-42000-072-00D in conjunction with the Center for Food Safety Engineering at Purdue University. The content is solely the responsibility of the authors and does not represent the official views of NSF or USDA.

References

[1] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[2] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer, 2012.

[3] Alon Vinnikov and Shai Shalev-Shwartz. K-means recovers ica filters when independent components are sparse. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 712–720, 2014.

[4] Dong Wang and Xiaoyang Tan. C-svddnet: An effective single-layer network for unsupervised feature learning. *arXiv preprint arXiv:1412.7259*, 2014.

[5] Manuel Blum, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller. On the applicability of unsupervised feature learning for object recognition in rgb-d data. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[6] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[7] Bulent Bayraktar, Padmapriya P Banada, E Daniel Hirleman, Arun K Bhunia, J Paul Robinson, and Bartek Rajwa. Feature extraction from light-scatter patterns of listeria colonies for identification and classification. *Journal of Biomedical Optics*, 11:034006, 2006.

[8] 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*, 9:1871–1874, 2008.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[11] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. pages 1096–1103, 2008.

[12] A. Khotanzad and Y.H. Hong. Invariant image recognition by zernike moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5):489–497, 1990.

[13] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 3(6):610–621, 1973.

[14] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[1] Purdue University Research Repository. http://dx.doi.org/10.4231/R7N58J9Z