Contextually Guided Convolutional Neural Networks for Learning Most Transferable Representations

1st Olcay Kursun
Department of Computer Science
Auburn University at Montgomery
Montgomery, Alabama
okursun@aum.edu

2nd Semih Dinc
EagleView Technologies
Bellevue, Washington
semih.dinc@eagleview.com

3rd Oleg V. Favorov
Joint Department of Biomedical Engineering
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina
favorov@email.unc.edu

***Abstract*—Implementing local contextual guidance principles in a single-layer CNN architecture, we propose an efficient algorithm for developing broad-purpose representations (i.e., representations transferable to new tasks without additional training) in shallow CNNs trained on limited-size datasets. A contextually guided CNN (CG-CNN) is trained on groups of neighboring image patches picked at random image locations in the dataset. Such neighboring patches are likely to have a common context and therefore are treated for the purposes of training as belonging to the same class. Across multiple iterations of such training on different context-sharing groups of image patches, CNN features that are optimized in one iteration are then transferred to the next iteration for further optimization, etc. In this process, CNN features acquire higher pluripotency, or inferential utility for any arbitrary classification task. In our applications to natural images and hyperspectral images, we find that CG-CNN can learn transferable features similar to those learned by the first layers of the well-known deep networks and produce favorable classification accuracies.**

***Index Terms*—Deep Learning, Contextual Guidance, Unsupervised Learning, Transfer Learning, Feature Extraction, Pluripotency**

I. Introduction

Although supervised deep CNNs are good at extracting pluripotent inferentially powerful transferable features, they require big labeled datasets with detailed external training supervision. Also, the backpropagation of the error all the way down to early layers can be problematic as the error signal weakens (a phenomenon known as the gradient vanishing [2]). To avoid these difficulties, in this paper we describe a self-supervised approach for learning pluripotent transferable features in a single CNN layer without reliance on feedback from higher layers and without a need for big labeled datasets. We demonstrate the use of this approach on two examples of a single CNN layer trained first on natural RGB images and then on hyperspectral images. Of course, there is a limit to sophistication of features that can be developed on raw input patterns by a single CNN layer. However, more complex and descriptive pluripotent features can be built by stacking multiple CNN layers, each layer developed in its turn by using our proposed approach on the outputs of the preceding layer(s).

Similar to deep CNNs, cortical areas making up the sensory cortex are organized in a modular and hierarchical architecture [11, 14]. Column-shaped modules (referred to as columns) making up a cortical area work in parallel performing information processing that resembles a convolutional block (convolution, rectification, and pooling) of a deep CNN. Each column of a higher-level cortical area builds its more complex features using as input the features of a local set of columns in the lower-level cortical area. Thus, as we go into higher areas these features become increasingly more global and nonlinear, and thus more descriptive [4, 7, 6, 10, 11].

Unlike deep CNNs, cortical areas do not rely on error backpropagation for learning what features should be extracted by their neurons. Instead, cortical areas rely on some local guiding information in optimizing their feature selection. While local (from the spatial and temporal context), such guiding information nevertheless promotes feature selection that enables insightful perception and successful behavior.

II. Contextually Guided Convolutional Neural Network (CG-CNN)

A. Basic Design

In this paper we apply the cortical context-guided strategy of developing pluripotent features in individual cortical areas to individual CNN layers. To explain our approach [13, 12], suppose we want to develop pluripotent features in a particular CNN layer (performing convolution + ReLU + pooling) on a given dataset of images. We set up a three-layer training system (Fig. 1-A) as:

1) The Input layer, which might correspond to a 2-dimensional field of raw pixels (i.e., a 3D tensor with two axes specifying row and column and one axis for the color channels) or the 3D tensor that was outputed by the preceding CNN layer with already developed features;
2) The CNN layer ("Feature Generator"), whose features we aim to develop;

3) The Classifier layer, a set of linear units fully connected with the output units of the CNN layer, each unit (with softmax activation) representing one of the training classes in the input patterns.

As in standard CNNs, during this network's training the classification errors will be backpropagated and used to adjust connection weights in the Classifier layer and the CNN layer.

While eventually (after its training) this CNN layer might be used as a part of a deep CNN to discriminate some particular application-specific classes of input patterns, during the training period the class labels will have to be assigned to the training input patterns internally; i.e., without any outside supervision. Adopting the cortical contextual guidance strategy, we can create a training class by picking at random a set of neighboring window patches in one of database images (Fig. 1-B). Being close together or even overlapping, such patches will have a high chance of showing the same object and those that do will share something in common (i.e., the same context). Other randomly chosen locations in the dataset images – giving us other training classes – will likely come from other objects and at those locations the neighboring window patches will have some other contexts to share. We can thus create a training dataset $\mathbf{X} = \{x^t \mid 1 \le t \le CN\}$ of $C \times N$ class-labeled input patterns by treating C sets of N neighboring window patches – each set drawn from a different randomly picked image location – as belonging to C training classes, uniquely labeled from 1 to C. These inputs are small $a \times a \times b$ tensors, $a \times a$ patches (feature-maps) with b features. We will refer to each such class of neighboring image patches as a "contextual group."

Upon a presentation of a particular input pattern x^t from the training dataset $\mathbf{X}$, the response of the CNN layer is computed as:

$$y_j^t = MaxPool([W_j * x^t]^+) \tag{1}$$

where y_j^t is the response of output unit j in the CNN layer with d units (i.e., y_j is CNN's feature j, where $1 \le j \le d$), W_j is the input connection weights of that unit (each unit has $w \times w \times b$ input connections), symbol * denotes convolution operation, and $[\cdot]^+ = \max\{\cdot, 0\}$ denotes the ReLU operation. Next, the response of the Classifier layer is computed by the softmax operation as:

$$z_l^t = \frac{\exp\left(V_l \cdot y^t\right)}{\sum_{c=1}^{C} \exp\left(V_c \cdot y^t\right)} \tag{2}$$

where z_l^t is the response of output unit l in the Classifier layer (expressing the probability of this input pattern x^t belonging to class l), $y^t = [y_j^t]_{j=1}^d$ is the d-dimensional feature vector computed as the output of the CNN layer, and V_l is the vector of connection weights of that unit from all the d units of the CNN layer.

During training, connection weights W and V are adjusted by error backpropagation so as to maximize the log-likelihood (whose negative is the loss function):

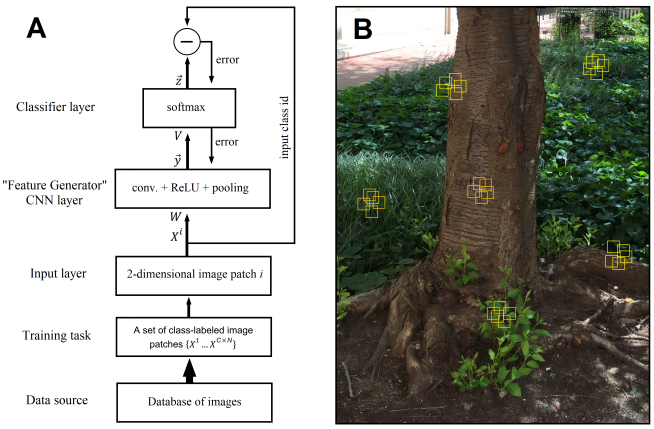

Fig. 1. Contextually Guided Convolutional Neural Network (CG-CNN) design. **(A)** CG-CNN architecture. **(B)** Class-defining contextual groups of image patches. Each image patch – used as input in CG-CNN training – is shown as a small square box superimposed on one of the database images. Neighboring patches constitute a contextual group and during network training are treated as belonging to the same class. During network training, locations of contextual groups are picked at random. Six such groups, or classes, are shown on this photo with five patches in each ($C = 6$ and $N = 5$).

$$\mathcal{L}\left(V, W \middle| \mathbf{X}\right) = \sum_{t=1}^{CN} \sum_{c=1}^{C} r_c^t \log z_c^t \tag{3}$$

where $r_c^t \in 0, 1$ indicates whether input pattern x^t belongs to class c.

B. Iterative Training Algorithm

CG-CNN training is performed over multiple iterations, with each iteration using a different small sample of contextual groups as training classes. That is, in each iteration a new small (e.g., $C = 100$) number of contextual groups is drawn from the database and the system is trained to discriminate them. Once this training is finished, a new small number of contextual groups is drawn and training continues in the next iteration on these new classes without resetting the already developed CNN connection weights.

For such iterative training of the CG-CNN system, we use an expectation-maximization (EM) algorithm [1]. The EM iterations alternate between performing an expectation (**E**) step and a maximization (**M**) step. At each EM iteration, we create a new training dataset $\mathbf{X} = \{x^t \mid 1 \le t \le CN\}$ of $C \times N$ self-class-labeled input patterns and randomly partition it into two subsets; one subset $\mathbf{X_E}$ to be used in the E-step, the other subset $\mathbf{X_M}$ to be used in the M-step. Next, we perform the E-step, which involves keeping W connection weights from the previous EM iteration (W_{old}), while training V connections of the Classifier layer on the newly created $\mathbf{X_E}$ subset so as to maximize its log-likelihood $\mathcal{L}$ (Eq. 3):

$$\text{E-step: } V_{new} = \underset{V}{\operatorname{argmax}}\, \mathcal{L}\left(V, W_{old} \middle| \mathbf{X_E}\right) \tag{4}$$

Next, we perform the M-step, which involves holding the newly optimized V connection weights fixed, while updating

W connections of the CNN layer on the $\mathbf{X_M}$ subset so as to maximize log-likelihood $\mathcal{L}$ one more time:

$$\text{M-step: } W_{new} = \underset{W}{\operatorname{argmax}}\, \mathcal{L}\big(V_{new}, W \big| \mathbf{X_M}\big) \tag{5}$$

By continuing to update the CNN layer weights W, while the contextual groups to be discriminated by the Classifier keep changing with every EM iteration, CG-CNN spreads the potentially high number of contextual groups (classes) needed for learning image-domain contextual regularities into multiple iterations [9]. The proposed EM algorithm for training CG-CNN achieves an efficient approach to learning the regularities that define contextual classes, which otherwise would theoretically require a C value in orders of tens of thousands [5].

No particular CNN architecture is required for applying the CG-CNN training procedures. CG-CNN accepts a small $a \times a \times b$ tensor as input. In CG-CNN's application to image-pixels directly, b simply denotes the number of color bands, and a denotes the width of the image patches that form the contextual groups. The kernel size of the convolutions and the stride are denoted by w and s, respectively. CG-CNN's Feature Generator (the CNN layer) learns to extract d features that most contextually and pluripotently represent any given $a \times a$ image patch. Note that at this level CG-CNN is not trying to solve an actual classification problem and is only learning a powerful local representation; only a pyramidal combination of these powerful local features can be used to describe an image big enough to capture real-world object class.

III. Experimental Results

A. Demonstration on Natural Images

To demonstrate the feasibility of CG-CNN developing pluripotent features using a limited number of images without any class-labels, we used images from the Caltech-101 dataset [8]. We used images from a single (face) class to emphasize that the proposed algorithm does not use any external supervision for tuning to its discriminatory features. Thus, our dataset had 435 color images, with sizes varying around 400×600 pixels.

Fig. 2. CG-CNN features after 1, 5, 20, 50, and 100 EM iterations.

We used a moderate number of contextual groups ($C = 100$) for the CG-CNN training. For selecting image patches for each contextual group, we used $g = 25$ pixels for the extent/slide of the seed window for spatial contextual guidance. We also used color jitter and color-to-gray conversion to enrich contextual groups. With each EM iteration, the network's features become progressively more defined and more resembling visual cortical features (gratings, Gabor-like features, and color blobs) as well as features extracted in the early layers of deep learning architectures AlexNet, GoogLeNet, and ResNet (see Fig. 2).

B. Demonstration on Texture Image Classification

We used the Brodatz dataset [3] of 13 texture images. To compare with AlexNet-Pool1 (which has 11×11 pixel features, stride $s = 4$, and therefore pooled window size of 19×19 pixels), we trained classifiers to discriminate textures in 19×19 patches. To compare with GoogLeNet and ResNet (which have 7×7 pixel features, stride $s = 2$, and therefore pooled window size of 11×11 pixels), we trained other classifiers to discriminate textures in 11×11 patches. For either of these two window sizes, we subdivided each 512×512 texture image into 256 32×32 subregions and picked 128 training image patches at random positions within 128 of these subregions, and other 128 test image patches at random positions within the remaining 128 subregions. The accuracies of the classifiers are listed in Table I. Classifiers directly applied to pixels performed much worse, indicating non-trivial nature of this classification task. These results demonstrate the superiority of using the transfer learning approach, with transferred features taken from CG-CNN or Pool-1 of pretrained deep networks.

TABLE I
Texture classification accuracies.

Method	11×11 **field**	19×19 **field**
CG-CNN	63.3 ± 0.7	74.3 ± 0.9
AlexNet		72.2 ± 0.7
GoogLeNet	62.2 ± 1.1	
ResNet-101	61.6 ± 0.9	
ResNet-18	61.5 ± 0.9	
RBF-SVM	53.6 ± 0.9	62.9 ± 0.9
Naive Bayes	39.3 ± 1.0	49.4 ± 0.8
Random Forest	34.7 ± 1.3	35.0 ± 1.5
MLP	33.6 ± 0.7	30.7 ± 0.6
K-NN	28.6 ± 0.9	29.2 ± 0.8
Linear-SVM	23.3 ± 1.2	28.0 ± 1.9
LR	22.8 ± 1.5	25.1 ± 0.6

C. Demonstration on Hyperspectral Image Classification

Unlike color image processing that uses a large image window with a few color channels (grayscale or RGB), Hyperspectral Image (HSI) analysis typically aims at classification of a single pixel characterized by a high number of spectral channels (bands). Typically, HSI datasets are small, and application of supervised deep learning to such small datasets can result in overlearning, not yielding pluripotent task-transferrable HSI-domain features. To improve generalization, the supervised

classification can benefit from unsupervised feature extraction of a small number of more complex/informative features than the raw data in the spectral channels. CG-CNN algorithm is applied on the Indian Pines and Salinas datasets with $a = 3$ pixels, $b = 220$ channels (corresponding to the HSI wavelengths), $d = 30$ features, $w = 1$ pixel (i.e., each convolution uses only the bands of a single pixel), $C = 20$ contextual groups, and $g = 2$ pixels for the extent of the spatial contextual guidance. In training of CG-CNN, the class labels of the HSI pixels were not used; instead, local groups of pixels (controlled by the g parameter) were treated as training classes. CG-CNN learns to represent its input HSI image patch, which is a hypercube of size $3 \times 3 \times 220$, in such a way that the image patch and its neighboring windows/positions (obtained by shifting it $g = \pm 2$ pixels in each direction) can be maximally discriminable from other contextual groups centered elsewhere. Note that only a total of $(2 \times 2 + 1)^2 = 25$ image patches are created for each contextual group. (We can also enrich contextual groups by adding band-specific noise or frequency shift, but leave this for future work.) Then, we used the extracted features as inputs to various classifiers. CG-CNN features were fed to these classifiers for the supervised classification task with 16 target classes (various vegetation, buildings, etc.). Pixel classification accuracies were computed using 10-fold cross validation. For comparison, we evaluated the use of all of the original raw variables (i.e., 220 bands) as inputs to the classifiers and we also compared with first 30 principal components of the Principal Component Analysis (PCA) and best 30 features selected by Random Forest. As shown in Tables II and III, the best performance was achieved by CG-CNN, combining CG-CNN features with Random Forest or K-NN classifiers.

TABLE II
HSI CLASSIFICATION RESULTS ON THE INDIAN PINES DATASET.

	Accuracy (%)			
	Orig.(220)	PCA(30)	RF(30)	CG-CNN(30)
ID3	67.8 ± 1.1	68.2 ± 1.3	67.0 ± 1.9	76.5 ± 1.6
LDA	79.4 ± 1.0	61.0 ± 1.4	62.1 ± 1.7	72.3 ± 1.5
Linear-SVM	85.2 ± 1.7	75.1 ± 1.4	76.0 ± 1.3	79.5 ± 0.8
Cubic-SVM	91.9 ± 0.8	86.0 ± 1.0	87.0 ± 1.0	94.7 ± 0.7
RBF-SVM	87.1 ± 0.6	81.5 ± 0.8	80.7 ± 1.8	90.5 ± 1.0
K-NN	76.0 ± 1.1	74.2 ± 1.2	80.0 ± 0.9	95.4 ± 0.5
RF	86.5 ± 0.9	82.8 ± 1.1	83.4 ± 1.3	**96.2 ± 0.6**

IV. CONCLUSIONS

The proposed Contextually Guided Convolutional Neural Network (CG-CNN) method uses a shallow CNN network with a small number of output units trained to recognize contextually related input patterns. Once the network is trained on one task, involving one set of different contexts, the convolutional features it develops are transferred to a new task, involving a new set of different contexts, and training continues. In a course of repeatedly transferred training on a sequence of such tasks, convolutional features progressively develop greater pluripotency (the degree of usefulness when transferred to new tasks) as demonstrated on texture and hyperspectral image datasets.

TABLE III
HSI CLASSIFICATION RESULTS ON THE SALINAS DATASET.

	Accuracy (%)			
	Orig.(220)	PCA(30)	RF(30)	CG-CNN(30)
ID3	88.3 ± 0.6	90.2 ± 0.3	85.9 ± 0.5	87.3 ± 0.5
LDA	91.7 ± 0.3	90.6 ± 0.4	89.6 ± 0.5	88.7 ± 0.6
Linear-SVM	92.9 ± 0.4	93.2 ± 0.3	92.2 ± 0.4	92.8 ± 0.4
Cubic-SVM	96.1 ± 1.5	96.2 ± 1.0	94.8 ± 1.0	96.8 ± 0.2
RBF-SVM	95.2 ± 0.3	96.5 ± 0.3	94.1 ± 0.3	95.7 ± 0.2
K-NN	91.9 ± 0.5	92.6 ± 0.4	93.0 ± 0.3	**97.8 ± 0.3**
RF	95.3 ± 0.3	96.1 ± 0.2	95.2 ± 0.3	**97.8 ± 0.3**

ACKNOWLEDGMENT

This work was supported, in part, by the National Science Foundation under Grant No. 2003740, by the Office of Naval Research, by the Arkansas INBRE program, and by the DART grant from NSF EPSCoR RII Track-1.

REFERENCES

[1] Ethem Alpaydin. *Introduction to machine learning, third edition*. The MIT Press, Cambridge, 2014.

[2] M Arjovsky and L Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Neural Information Processing Systems (NIPS) 2016 Workshop on Adversarial Training. In review for ICLR*, volume 2016, 2017.

[3] Phil Brodatz. *Textures: A photographic album for artists and designers*. Dover Pubns, 1966.

[4] Andy Clark and Chris Thornton. Trading spaces: Computation, representation, and the limits of uninformed learning. *Behavioral and Brain Sciences*, 20(1):57–66, 1997.

[5] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014.

[6] Oleg V Favorov and Olcay Kursun. Neocortical layer 4 as a pluripotent function linearizer. *Journal of Neurophysiology*, 105(3):1342–1360, 2011.

[7] Oleg V Favorov and Dan Ryder. Sinbad: A neocortical mechanism for discovering environmental variables and regularities hidden in sensory input. *Biological Cybernetics*, 90(3):191–202, 2004.

[8] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[10] Kalanit Grill-Spector and Rafael Malach. The human visual cortex. *Annual Review of Neuroscience*, 27:649–677, 2004.

[11] Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in Neural Circuits*, 11:81, 2017.

[12] Olcay Kursun, Semih Dinc, and Oleg V. Favorov. Contextually guided convolutional neural networks for learning most transferable representations, 2021.

[13] Olcay Kursun and Oleg V. Favorov. Suitability of features of deep convolutional neural networks for modeling somatosensory information processing. In *Pattern Recognition and Tracking XXX*, volume 10995, pages 94 – 105. International Society for Optics and Photonics, SPIE, 2019.

[14] Adam H Marblestone, Greg Wayne, and Konrad P Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10:94, 2016.