Learning Discriminative Features via Label Consistent Neural Network

Zhuolin Jiang[†*], Yaming Wang[‡*], Larry Davis[‡], Walter Andrews[§], Viktor Rozgic[†]
[†]Raytheon BBN Technologies, Cambridge, MA, 02138
[‡]University of Maryland, College Park, MD, 20742
[§]Sierra Nevada Corporation, San Antonio, TX, 78229
{zjiang,vrozgic}@bbn.com, {wym,lsd}@umiacs.umd.edu, walter.andrews@sncorp.com

Abstract

Deep Convolutional Neural Networks (CNN) enforce supervised information only at the output layer, and hidden layers are trained by back propagating the prediction error from the output layer without explicit supervision. We propose a supervised feature learning approach, Label Consistent Neural Network, which enforces direct supervision in late hidden layers in a novel way. We associate each neuron in a hidden layer with a particular class label and encourage it to be activated for input signals from the same class. More specifically, we introduce a label consistency regularization called "discriminative representation error" loss for late hidden layers and combine it with classification error loss to build our overall objective function. This label consistency constraint alleviates the common problem of gradient vanishing and tends to faster convergence; it also makes the features derived from late hidden layers discriminative enough for classification even using a simple k-NN classifier. Experimental results demonstrate that our approach achieves state-of-the-art performances on several public datasets for action and object category recognition.

1. Introduction

Convolutional neural networks (CNN) [20] have exhibited impressive performances in many computer vision tasks such as image classification [17], object detection [5] and image retrieval [27]. When large amounts of training data are available, CNN can automatically learn hierarchical feature representations, which are more discriminative than previous hand-crafted ones [17].

Encouraged by their impressive performance in static image analysis tasks, several CNN-based approaches have been developed for action recognition in videos [12, 15, 25, 28, 35, 44]. Although promising results have been reported, the advantages of CNN approaches over traditional ones [34] are not as overwhelming for videos as in static images. Compared to static images, videos have larger variations in appearance as well as high complexity introduced by temporal evolution, which makes learning features for recognition from videos more challenging. On the other hand, unlike large-scale and diverse static image data [2], annotated data for action recognition tasks is usually insufficient, since annotating massive videos is prohibitively expensive. With only limited annotated data, learning discriminative features via deep neural network can lead to severe overfitting and slow convergence. To tackle these issues, previous works have introduced effective practical techniques such as ReLU [24] and Drop-out [10] to improve the performance of neural networks, but have not considered directly improving the discriminative capability of neurons. The features from a CNN are learned by back-propagating prediction error from the output layer [19], and hidden layers receive no direct guidance on class information. Worse, in very deep networks, the early hidden layers often suffer from vanishing gradients, which leads to slow optimization convergence and the network converging to a poor local minimum. Therefore, the quality of the learned features of the hidden layers might be potentially diminished [43, 6].

To tackle these problems, we propose a new supervised deep neural network, *Label Consistent Neural Network*, to learn discriminative features for recognition. Our approach provides explicit supervision, *i.e.* label information, to late hidden layers, by incorporating a label consistency constraint called "discriminative representation error" loss, which is combined with the classification loss to form the overall objective function. The benefits are two-fold: (1) with explicit supervision to hidden layers, the problem of vanishing gradients can be alleviated and faster convergence is observed; (2) more discriminative late hidden layer features lead to increased discriminative power of classifiers at the output layer; interestingly, the learned discriminative features alone can achieve good classification performance even with a simple k-NN classifier. In practice, our new formulation can be easily incorporated into any neural network trained using backpropagation. Our approach is evaluated on publicly available action and object recognition datasets. Although we only present experimental results for action and object recognition, the method can be applied to other tasks such as image retrieval, compression, restorations *etc*., since it generates class-specific compact representations.

1.1. Main Contributions

The main contributions of LCNN are three-fold.

- By adding explicit supervision to late hidden layers via

*Indicates equal contributions.

a "discriminative representation error", LCNN learns more discriminative features resulting in better classifier training at the output layer. The representations generated by late hidden layers are discriminative enough to achieve good performance using a simple k-NN classifier.

- The label consistency constraint alleviates the problem of vanishing gradients and leads to faster convergence during training, especially when limited training data is available.
- We achieve state-of-the-art performance on several action and object category recognition tasks, and the compact class-specific representations generated by LCNN can be directly used in other applications.

2. Related Work

CNNs have achieved performance improvements over traditional hand-crafted features in image recognition [17], detection [5] and retrieval [27] *etc*. This is due to the availability of large-scale image datasets [2] and recent technical improvements such as ReLU [24], drop-out [10], 1×1 convolution [23, 32], batch normalization [11] and data augmentation based on random flipping, RGB jittering, contrast normalization [17, 23], which helps speed up convergence while avoiding overfitting.

AlexNet [17] initiated the dramatic performance improvements of CNN in static image recognition and current state-of-the-art performance has been obtained by deeper and more sophisticated network architectures such as VGGNet [29] and GoogLeNet [32]. Very recently, researchers have applied CNNs to action and event recognition in videos. While initial approaches use image-trained CNN models to extract frame-level features and aggregate them into video-level descriptors [25, 44, 38], more recent work trains CNNs using video data and focuses on effectively incorporating the temporal dimension and learning good spatial-temporal features automatically [12, 15, 28, 36, 41, 35]. Two-stream CNNs [28] are perhaps the most successful architecture for action recognition currently. They consist of a spatial net trained with video frames and a temporal net trained with optical flow fields. With the two streams capturing spatial and temporal information separately, the late fusion of the two produces competitive action recognition results. [36] and [41] have obtained further performance gain by exploring deeper two-stream network architectures and refining technical details; [35] achieved state-of-the-art in action recognition by integrating two-stream CNNs, improved trajectories and Fisher Vector encoding.

It is also worth comparing our LCNN with limited prior work which aims to improve the discriminativeness of learned features. [1] performs greedy layer-wise supervised pre-training as initialization and fine-tunes the parameters of all layers together. Our work introduces the supervision

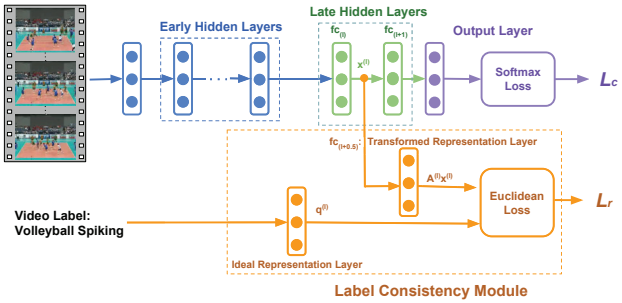


Figure 1. An example of the LCNN structure. The label consistency module is added to the l^{th} hidden layer, which is a fully-connected layer fc_l. Its representation $\mathbf{x}^l$ is transformed to be $\mathbf{A}^{(l)}\mathbf{x}^l$, which is the output of the transformed representation layer $\text{fc}_{l+0.5}$. Note that the applicability of the proposed label consistency module is not limited to fully-connected layers.

to intermediate layers as part of the objective function during training and can be optimized by backpropagation in an integrated way, rather than layer-wise greedy pretraining and then fine-tuning. [40] replaces the output softmax layer with an error-correcting coding layer to produce error correcting codes as network output. Their network is still trained by back-propagating the error at the output and no direct supervision is added to hidden layers. Deeply Supervised Net (DSN) [21] introduces an SVM classifier for *each* hidden layer, and the final objective function is the linear combination of the prediction losses at all hidden layers and output layer. Using all-layer supervision, balancing between multiple losses might be challenging and the network is non-trivial to tune, since only the classifier at the output layer will be used at test time and the effects of the classifiers at hidden layers are difficult to evaluate. Similarly, [31] also adds identification and verification supervisory signals to each hidden layer to extract face representations. In our work, instead of adding a prediction loss to each hidden layer, we introduce a novel representation loss to guide the format of the learned features at late hidden layers only, since early layers of CNNs tend to capture low-level edges, corners and mid-level parts and they should be shared across categories, while the late hidden layers are more class-specific [43].

3. Feature Learning via Supervised Deep Neural Network

Let $(\mathbf{x}, y)$ denote a training sample $\mathbf{x}$ and its label y. For a CNN with n layers, let $\mathbf{x}^{(i)}$ denote the output of the i^{th} layer and L_c its objective function. $\mathbf{x}^{(0)} = \mathbf{x}$ is the input data and $\mathbf{x}^{(n)}$ is the output of the network. Therefore, the network architecture can be concisely expressed as

$$\mathbf{x}^{(i)} = F(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)}), \quad i = 1, 2, ..., n \tag{1}$$

$$L_c = L_c(\mathbf{x}, y, \mathbf{W}) = C(\mathbf{x}^{(n)}, y), \tag{2}$$

where $\mathbf{W}^{(i)}$ represents the network parameters of the i^{th} layer, $\mathbf{W}^{(i)}\mathbf{x}^{(i-1)}$ is the linear operation (*e.g.* convolution in convolutional layer, or linear transformation in fully-connected layer), and $\mathbf{W} = \{\mathbf{W}^{(i)}\}_{i=1,2,...,n}$; $F(\cdot)$ is a non-linear activation function (*e.g.* ReLU); $C(\cdot)$ is a prediction error such as softmax loss. The network is trained with back-propagation, and the gradients are computed as:

$$\frac{\partial L_c}{\partial \mathbf{x}^{(i)}} = \begin{cases} \frac{\partial C(\mathbf{x}^{(n)}, y)}{\partial \mathbf{x}^{(n)}}, & i = n \\ \frac{\partial L_c}{\partial \mathbf{x}^{(i+1)}} \frac{\partial F(\mathbf{W}^{(i+1)}\mathbf{x}^{(i)})}{\partial \mathbf{x}^{(i)}}, & i \neq n \end{cases} \tag{3}$$

$$\frac{\partial L_c}{\partial \mathbf{W}^{(i)}} = \frac{\partial L_c}{\partial \mathbf{x}^{(i)}} \frac{\partial F(\mathbf{W}^{(i)}\mathbf{x}^{(i-1)})}{\partial \mathbf{W}^{(i)}}, \tag{4}$$

where $i = 1, 2, 3, ..., n$.

4. Label Consistent Neural Network (LCNN)

4.1. Motivation

The sparse representation for classification assumes that a testing sample can be well represented by training samples from the same class [37]. Similarly, dictionary learning for recognition maintains label information for dictionary items during training in order to generate discriminative or class-specific sparse codes [14, 39]. In a neural network, the representation of a certain layer is generated by the neuron activations in that layer. If the class distribution for each neuron is highly peaked in one class, it enforces a label consistency constraint on each neuron. This leads to a discriminative representation over learned class-specific neurons.

It has been observed that early hidden layers of a CNN tend to capture low-level features shared across categories such as edges and corners, while late hidden layers are more class-specific [43]. To improve the discriminativeness of features, LCNN adds explicit supervision to late hidden layers; more specifically, we associate each neuron to a certain class label and ideally the neuron will only activate when a sample of the corresponding class is presented. The label consistency constraint on neurons in LCNN will be imposed by introducing a "discriminative representation error" loss on late hidden layers, which will form part of the objective function during training.

4.2. Formulation

The overall objective function of LCNN is a combination of the discriminative representation error at late hidden layers and the classification error at the output layer:

$$L = L_c + \alpha L_r \tag{5}$$

where L_c in Equation (2) is the classification error at the output layer, L_r is the discriminative representation error in Equation (6) and will be discussed in detail below, and α is a hyper parameter balancing the two terms.

Suppose we want to add supervision to the l^{th} layer. Let $(\mathbf{x}, y)$ denote a training sample and $\mathbf{x}^{(l)} \in \mathbb{R}^{N_l}$ be the corresponding representation produced by the l^{th} layer, which is defined by the activations of N_l neurons in that layer. Then the discriminative representation error is defined to be the difference between the transformed representation $\mathbf{A}^{(l)}\mathbf{x}^{(l)}$ and the ideal discriminative representation $\mathbf{q}^{(l)}$:

$$L_r = L_r(\mathbf{x}^{(l)}, y, \mathbf{A}^{(l)}) = \|\mathbf{q}^{(l)} - \mathbf{A}^{(l)}\mathbf{x}^{(l)}\|_2^2, \tag{6}$$

where $\mathbf{A}^{(l)} \in \mathbb{R}^{N_l \times N_l}$ is a linear transformation matrix, and the binary vector $\mathbf{q}^{(l)} = [q_1^{(l)}, \ldots, q_j^{(l)}, \ldots, q_{N_l}^{(l)}]^{\mathrm{T}} \in \{0, 1\}^{N_l}$ denotes the ideal discriminative representation which indicates the ideal activations of neurons (j denotes the index of neuron, *i.e.* the index of feature dimension). Each neuron is associated with a certain class label and, ideally, only activates to samples from that class. Therefore, when a sample is from Class c, $q_j^{(l)} = 1$ if and only if the j^{th} neuron is assigned to Class c, and neurons associated to other classes should not be activated so that the corresponding entry in $\mathbf{q}^{(l)}$ is zero. Notice that $\mathbf{A}^{(l)}$ is the only parameter needed to be learned, while $\mathbf{q}^{(l)}$ is pre-defined based on label information from training data.

Suppose we have a batch of six training samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_6\}$ and the class labels $\mathbf{y} = [y_1, y_2, \ldots, y_6] = [1, 1, 2, 2, 3, 3]$. Further assume that the l^{th} layer has 7 neurons $\{d_1, d_2, \ldots, d_7\}$ with $\{d_1, d_2\}$ associated with Class 1, $\{d_3, d_4, d_5\}$ Class 2, and $\{d_6, d_7\}$ Class 3. Then the ideal discriminative representations for these six samples are given by:

$$\mathbf{Q}^{(l)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \tag{7}$$

where each column is an ideal discriminative representation corresponding to a training sample. The ideal representations ensured that the input signals from the same class have similar representations while those from different classes have dissimilar representations.

The discriminative representation error (6) forces the learned representation to approximate the ideal discriminative representation, so that the resulting neurons have the label consistency property [14], *i.e.* the class distributions of each neuron [1] from layer l are extremely peaked in one class. In addition, with more discriminative representations, the classifier, especially linear classifiers, at the output layer can achieve better performance. This is because the discriminative property of $\mathbf{x}^{(l)}$ is very important for the performance of a linear classifier.

[1] Similar to computing the class distributions for dictionary items in [26], the class distributions of each neurons from the l^{th} layer can be derived by measuring their activations $\mathbf{x}^{(l)}$ over input signals corresponding to different classes.

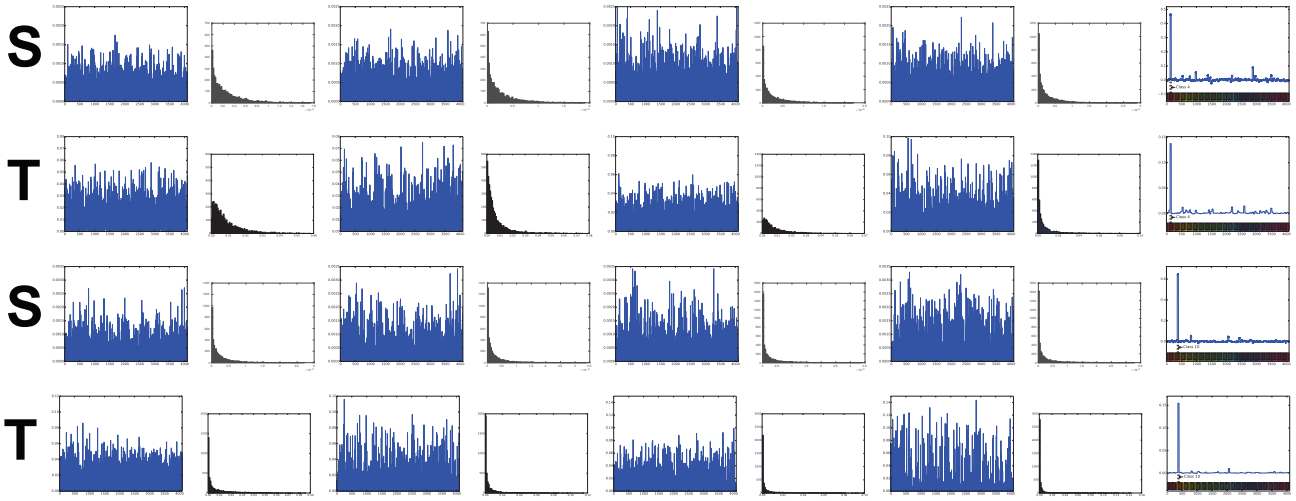


Figure 2. Examples of learned representations from layers fc_6, fc_7 and $fc_{7.5}$ using LCNN and the baseline (VGGNet-16). Each curve indicates an average of representations for different testing videos from the same class in the UCF101 dataset. The first two rows correspond to class 4 (Baby Crawling, 35 videos) while the third and fourth rows correspond to class 10 (Bench Press, 48 videos). The curves in every two rows correspond to the spatial net (denoted as 'S') and temporal net (denoted as 'T') in our two-stream framework for action recognition. (a) fc_6 representations using VGGNet-16; (b) Histograms (with 100 bins) for representations from (a); (c) fc_6 representations using LCNN; (d) Histograms for representations from (c); (e) fc_7 representations using VGGNet-16; (f) Histograms for representations from (e); (g) fc_7 representations using LCNN; (h) Histograms for representations from (g); (i) $fc_{7.5}$ representations (i.e. transformed fc_7 representations) using LCNN. The entropy values for representations from (a)(c)(e)(g) are computed as: (11.32, 11.42, 11.02, 10.75), (11.2, 11.14, 10.81, 10.34), (11.08, 11.35, 10.67, 10.17), (11.02, 10.72, 10.55, 9.37). LCNN can generate lower-entropy representations for each class compared to VGGNet-16. Each color from the color bars in (i) represents one class for a subset of neurons. The black dashed lines indicate that the curves are highly peaked in one class. The figure is best viewed in color and 600% zoom in.

An example of the LCNN architecture is shown in Figure 1. The linear transformation is implemented as a fully-connected layer. We refer it as 'Transformed Representation Layer'. We create a new 'Ideal Representation Layer' which transforms a class label into the corresponding binary vector $\mathbf{q}^{(l)}$; then we feed the outputs of these two layers into the Euclidean loss layer.

In our experiments, we allocate the neurons in the late hidden layer to each class as follows: assuming N_l neurons in that layer and m classes, we first allocate $\lfloor N_l/m \rfloor$ neurons to each class and then allocate the remaining $(N_l - m\lfloor N_l/m \rfloor)$ neurons to the top $(N_l - m\lfloor N_l/m \rfloor)$ classes with high intra-class appearance variation. Therefore each neuron in the late hidden layer is associated with a category label, but an input signal of a category certainly can (and does) use all neurons (learned features), as the representations in Figure 2(i) illustrate, *i.e.* sharing features between categories is not prohibited.

4.3. Network Training

LCNN is trained via stochastic gradient descent. We need to compute the gradients of L in Equation (5) w.r.t. all the network parameters $\{\mathbf{W}, \mathbf{A}^{(l)}\}$. Compared with standard CNN, the difference lies in two gradient terms, *i.e.* $\frac{\partial L}{\partial \mathbf{x}^{(l)}}$ and $\frac{\partial L}{\partial \mathbf{A}^{(l)}}$, since $\mathbf{x}^{(l)}$ and $\mathbf{A}^{(l)}$ are the only parameters which are related to the newly added discriminative error $L_r(\mathbf{x}^{(l)}, y, \mathbf{A}^{(l)})$ and the other parameters act independently from it.

It follows from Equations (5) and (6) that

$$\frac{\partial L}{\partial \mathbf{x}^{(i)}} = \begin{cases} \frac{\partial L_c}{\partial \mathbf{x}^{(i)}}, & i \neq l \\ \frac{\partial L_c}{\partial \mathbf{x}^{(l)}} + 2\alpha(\mathbf{A}^{(l)}\mathbf{x}^{(l)} - \mathbf{q}^{(l)})^{\mathrm{T}}\mathbf{A}^{(l)}, & i = l \end{cases} \tag{8}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(i)}} = \frac{\partial L_c}{\partial \mathbf{W}^{(i)}}, \quad \forall i \in \{1, 2, ..., n\} \tag{9}$$

$$\frac{\partial L}{\partial \mathbf{A}^{(l)}} = 2\alpha(\mathbf{A}^{(l)}\mathbf{x}^{(l)} - \mathbf{q}^{(l)})\mathbf{x}^{(l)\mathrm{T}}, \tag{10}$$

where $\frac{\partial L_c}{\partial \mathbf{x}^{(i)}}$ and $\frac{\partial L_c}{\partial \mathbf{W}^{(i)}}$ are computed by Equations (3) and (4), respectively.

5. Experiments

We evaluate our approach on two action recognition datasets: UCF101 [30] and THUMOS15 [8], and three object category datasets: Cifar-10 [16], ImageNet [2] and Caltech101 [22]. Our implementation of LCNN is based on the CAFFE toolbox [13].

To verify the effectiveness of our approach, we train LCNN in two ways: (1) We use the discriminative representation error loss L_r only; (2) We use the combination of

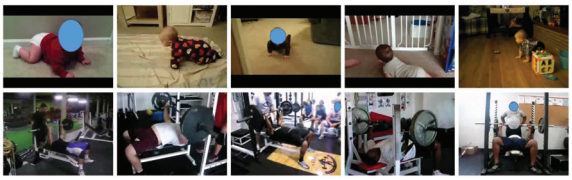


Figure 3. Class 4 (BabyCrawling) and class 10 (BenchPress) samples from the UCF101 action dataset.

Network Architecture	Spatial	Temporal	Both
ClarifaiNet [28]	72.7	81	87
VGGNet-19 [41]	75.7	78.3	86.7
VGGNet-16 [36]	79.8	85.7	90.9
VGGNet-16* [36]	-	85.2	-
baseline	77.48	83.71	-
LCNN-1	80.1	85.59	89.87
LCNN-2 (argmax)	80.7	85.57	91.12
LCNN-2 (k-NN)	81.3	85.77	89.84

Table 1. Classification performance with different two-stream CNN approaches on the UCF101 dataset. The results of [28, 36, 41] are copied from their original papers. The VGGNet-16* result is obtained by testing the model shared by [36]. The 'baseline' are the results of running the two-stream CNN implementation provided by [36], where the VGGNet-16 architecture is used for each stream. LCNN and baseline are trained with the same parameter setting and initial model. The only difference between LCNN-2 and the baseline is that we add explicit supervision to fc_7 layer for LCNN-2. For LCNN-1, we remove the softmax layer from the baseline network but add explicit supervision to fc_7 layer.

L_r and the softmax classification error loss L_c as in Equation (5). We refer to the networks trained in these ways as 'LCNN-1' and 'LCNN-2', respectively. The baseline is to use the softmax classification error loss L_c only during network training. We refer to it as 'baseline' in the following. Note that the baseline and LCNN are trained with the same parameter setting and initial model in all our experiments.

For action and object recognition, we introduce two classification approaches: (1) **argmax**: we follow the standard CNN practice of taking the class label corresponding to the maximum prediction score; (2) k**-NN**: We use the transformed representation $\mathbf{A}^{(l)}\mathbf{x}^{(l)}$ to represent an image, video frame or optical flow field and then do simple k-NN classification. LCNN-1 always uses 'k-NN' for classification while LCNN-2 can use either 'argmax' or 'k-NN' to do classification.

5.1. Action Recognition

5.1.1 UCF101 Dataset

The UCF101 dataset [30] consists of $13,320$ video clips from 101 action classes, and every class has more than 100 clips. Some video examples from class 4 and class 10 are given in Figure 3. In terms of evaluation, we use the standard split-1 train/test setting. Split-1 contains around $10,000$ clips for training and the rest for testing.

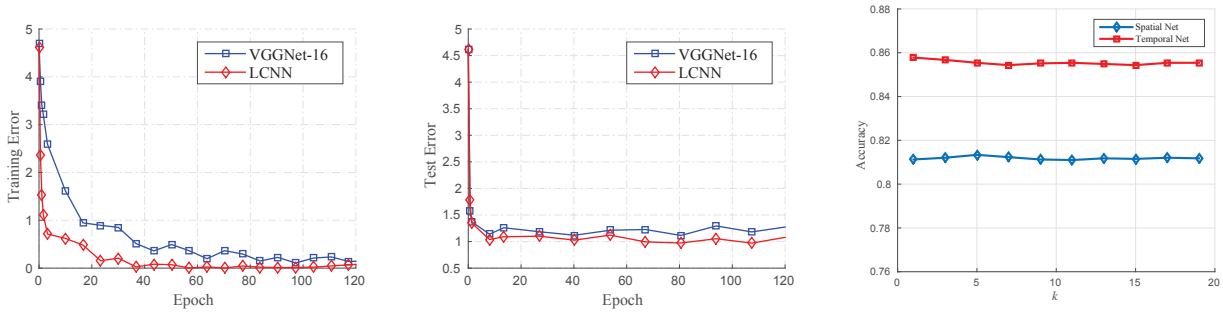
Method	Acc. (%)	Method	Acc. (%)
Karpathy [15]	65.4	Wang [34]	85.9
Donahue [3]	82.9	Lan [18]	89.1
Ng [25]	88.6	Zha [44]	89.6
LCNN-2 (argmax)	91.12		

Table 2. Recognition performance comparisons with other state-of-the-art approaches on the UCF101 dataset. The results of [15, 34, 3, 18, 25, 44] are copied from their original papers.

We choose the popular two-stream CNN as in [28, 36, 41] as our basic network architecture for action recognition. It consists of a spatial net taking video frames as input and a temporal net taking 10-frame stacking of optical flow fields. Late fusion is conducted on the outputs of the two streams and generates the final prediction score. During testing, we sample 25 frames (images or optical flow fields) from a video as in [28] for spatial and temporal nets. The class scores for a testing video is obtained by averaging the scores across sampled frames. In our experiments, we fuse spatial and temporal net prediction scores using a simple weighted average rule, where the weight is set to 2 for temporal net and 1 for spatial net.

We use VGGNet-16 architecture [29] as in [36] for two streams where the explicit supervision is added in the late hidden layer fc_7, which is the second fully-connected layer. More specifically, we feed the output of layer fc_7 to a fully-connected layer (denoted as $\text{fc}_{7.5}$) to produce the transformed representation, and compare it to the ideal discriminative representation $\mathbf{q}^{(\text{fc7})}$. The implementation of this explicit supervision is shown in Figure 5(a). Since UCF101 has 101 classes and the fc_7 layer of VGGNet has output dimension 4096, the output of $\text{fc}_{7.5}$ has the same size 4096, and around 40 neurons are associated to each class. For both streams, we set $\alpha = 0.05$ in (5) to balance two terms.

Benefits of Adding Explicit Supervision to Late Hidden Layers. We aim to demonstrate the benefits of adding explicit supervision to late hidden layers. We first obtain the baseline result by running the standard two-stream CNN implementation provided by [36], which uses softmax classification loss only to train the spatial and temporal nets. Then we remove the softmax layers from this two-stream CNN but add explicit supervision to the fc_7 hidden layers. We call this network as 'LCNN-1'. Next we maintain the softmax layers in the standard two-stream CNN but add explicit supervision to the fc_7 layers. We call this network as 'LCNN-2'. Please note that we do use the same parameter setting and initial model in these three types of neural networks. The results are summarized in Table 1. It can be seen from the results of LCNN-1 that even without the help of the classifier, our label consistency constraint alone is very effective for learning discriminative features and achieves



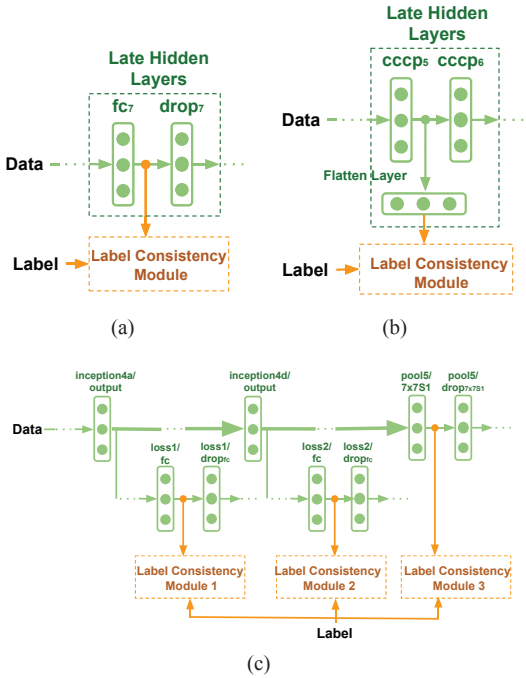
(a) (b) (c)

Figure 4. Training and testing errors of spatial net trained by LCNN-2 and the baseline (VGGNet-16) on the UCF101 dataset. (a) Training error comparison; (b) Testing error comparison; (c) Effects of parameter selection of k-NN neighborhood size k on the classification accuracy performances on the UCF101 dataset. The spatial and temporal nets trained by LCNN-2 are not sensitive to the selection of k.

better classification performance than the baseline. We can also see that adding explicit supervision to late hidden layers not only improves the classification results at the output layer (LCNN-2 (argmax)), but also generates discriminative representations which achieve better results even with a simple k-NN classifier (LCNN-2 (k-NN)). In addition, we compare LCNN with other approaches in Table 2.

Discriminability of Learned Representations. We visualize the representations of *test videos* generated by late hidden layers $\text{fc}_{7.5}$, fc_7 and fc_6 in Figure 2. It can be seen that the entries of layer $\text{fc}_{7.5}$ representations in Figure 2(i) are very peaked at the corresponding class, which forms a very good approximation to the ideal discriminative representation. Please note that a video of a testing class certainly can (and does) use neurons from other classes as shown in Figure 2(i). It indicates that sharing features between classes is not prohibited. Further notice that such discriminative capability is achieved during testing, which indicates that LCNN generalizes well without severe overfitting. For fc_7 and fc_6 representations in Figures 2(c) and 2(g), their entropy has decreased, which means that the discriminativeness of previous layers benefits from the backpropagation of the discriminative representation error introduced by LCNN. In Figure 4(c), we plot the performance curves for a range of k (recall k is the number of nearest neighbors for a k-NN classifier) using LCNN-2. Our approach is insensitive to the selection of k, likely due to the increase of inter-class distances in generated class-specific representations.

Smaller Training and Testing Errors. We investigate the convergence and testing error of LCNN during network training. We plot the testing error and training error w.r.t. number of epochs from spatial net in Figure 4. It can be seen that LCNN has smaller training error than the baseline (VGGNet-16), which can converge more quickly and alleviate gradient vanishing due to the explicit supervision to late hidden layers. In addition, LCNN has smaller testing error compared with the baseline, which means that LCNN has better generalization capability.



(a) (b) (c)

Figure 5. Examples of direct (explicit) supervision in the late hidden layers including (a) fc_7 layer in the CNN architectures including VGGNet [29] and AlexNet [17]; (b) CCCP5 layer in the Network-in-Network [23];(c) loss_1/fc, loss_2/fc and $\text{Pool}_5/7 \times 7\text{S}_1$ in the GoogLeNet [32]. The symbol of three dots denotes other layers in the network.

5.1.2 THUMOS15 Dataset

Next we evaluate our approach on the more challenging THUMOS15 challenge action dataset. It includes all 13,320 video clips from UCF101 dataset for training, and 2,104 temporarily *untrimmed* videos from the 101 classes for validation. We employ the standard Mean Average Precision (mAP) for THUMOS15 recognition task to evaluate LCNN.

We use two-stream CNN based on VGGNet-16 discussed in Section 5.1.1, where explicit supervision is added in the fc_7 layers. We train it using all UCF101 data. We

Network Architecture	Spatial	Temporal	Both
VGGNet-16 [36]	54.5	42.6	-
ClarifaiNet [28]	42.3	47	-
GoogLeNet [32]	53.7	39.9	-
baseline	55.8	41.8	-
LCNN-1	56.9	45.1	59.8
LCNN-2 (argmax)	57.3	44.9	61.7
LCNN-2 (k-NN)	58.6	45.9	62.6

Table 3. Mean Average Precision performance on the THUMOS15 validation set. The results of [36, 28, 32] are copied from [36]. The 'baseline' are the results of running the two-stream CNN implementation provided by [36]. LCNN and baseline are trained with the same parameter setting and initial model. Our result 62.6% mAP is also better than 54.7% using method in [18], which is reported in [8].

used the evaluation tool provided by the dataset provider to evaluate mAP performance, which requires the probabilities for each category for a testing video. For our two classification schemes, *i.e.* argmax and k-NN, we use different approaches to generate the probability prediction for a testing video. For argmax, we can directly use the output layer. For the k-NN scheme, given the representation from $\text{fc}_{7.5}$ layer, we compute a sample's distances to classes only presented in its k nearest neighbors, and convert them to similarity weights using a Gaussian kernel and set other classes to have very low similarity; finally we calculate the probability by doing L1 normalization on the similarity vector.

We obtained the baseline by running the two-stream CNN implementation provided by [36]. We compare our LCNN results with the baseline and other state-of-the-art approaches [36, 28, 32] on the THUMOS15 dataset. The results are summarized in Table 3. LCNN-1 is better than the baseline and LCNN-2 can further improve the mAP performances. Our results in the spatial stream outperform the results in [36], [28] and [32], while our results in the temporal stream are comparable to [28]. Based on this experiment, we can see that LCNN is highly effective and generalizes well to more complex testing data.

5.2. Object Recognition

5.2.1 CIFAR-10 Dataset

The CIFAR-10 dataset contains $60,000$ color images from 10 classes, which are split into $50,000$ training images and 10,000 testing images. We compare LCNN-2 with several recently proposed techniques, especially the Deeply Supervised Net (DSN) [21], which adds explicit supervision to all hidden layers. For our underlying architecture, we also choose Network in Network (NIN) [23] as in [21]. We follow the same data augmentation techniques in [23] by zero padding on each side, then do corner cropping and random flipping during training.

For LCNN-2, we add the explicit supervision to the 5^{th} cascaded cross channel parametric pooling layer (cccp_5) [23], which is a late 1×1 convolutional layer. We first flatten the output of this convolutional layer into a one dimensional vector, and then feed it into a fully-connected layer (denoted as $\text{fc}_{5.5}$) to obtain the transformed representation. This implementation is shown in Figure 5(b). We set the hyper-parameter $\alpha = 0.0375$ during training. For classification, we adopt the argmax classification scheme.

Method (Without Data Augment.)	Test Error (%)
Stochastic Pooling [42]	15.13
Maxout Networks [7]	11.68
DSN [21]	9.78
baseline	10.41
LCNN-2 (argmax)	9.75
Method (With Data Augment.)	**Test Error (%)**
Maxout Networks [7]	9.38
DropConnect [33]	9.32
DSN [21]	8.22
baseline	8.81
LCNN-2 (argmax)	8.14

Table 4. Test error rates on the CIFAR-10 dataset. The results of [42, 7, 33, 21] are copied from [23]. The 'baseline' is the result of Network in Network (NIN) [23]. Following [21], LCNN-2 is also trained on top of the NIN implementation provided by [23]. The only difference between the baseline and LCNN-2 is that we add the explicit supervision to the cccp_5 layer for LCNN-2.

The baseline result is from NIN [23]. LCNN-2 is constructed on top of the NIN implementation provided by [23] with the same parameter setting and initial model. We compare our result with the baseline and other approaches including DSN [21]. The results are summarized in Table 4. Regardless of the data augmentation, LCNN-2 consistently outperforms all previous methods, including the baseline NIN [23] and DSN [21]. The results are impressive, since DSN adds an SVM loss to every hidden layer during training, while LCNN-2 only adds a discriminative representation error loss to one late hidden layer. It suggests that adding direct supervision to the more category-specific late hidden layers might be more effective than to the early hidden layers which tend to be shared across categories.

5.2.2 ImageNet Dataset

We aim to demonstrate that LCNN can be combined with state-of-the-art CNN architecture GoogLeNet [32], which is a very deep CNN with 22 layers and achieved the best performance on ILSVRC 2014. The ILSVRC classification challenge contains about 1.2 million training images and $50,000$ images for validation from 1,000 categories.

To tackle such a very deep network architecture, we construct LCNN on top of the GoogLeNet implementation in CAFFE toolbox by adding explicit supervision to multiple late hidden layers instead of a single one. Specifically, as shown in Figure 5(c), the discriminative representation error losses are added to three layers: loss_1/fc, loss_2/fc and

Network Architecture	Top-1 (%)	Top-5 (%)
GoogLeNet [32]	-	89.93
AlexNet [17]	58.9	-
Clarifai [43]	62.4	-
baseline	62.64	85.54
LCNN-2 (argmax)	68.68	89.03

Table 5. Recognition Performances using different approaches on the ImageNet 2012 Validation set. The result of [32] is copied from original paper while the results of [17, 43] are copied from [40]. The 'baseline' is the result of running the GoogLeNet implementation in CAFFE toolbox. The only difference between the baseline and LCNN-2 is that we add explicit supervision to three layers (loss_1/fc, loss_2/fc and $\text{Pool}_5/7 \times 7\text{S}_1$) for LCNN-2.

$\text{Pool}_5/7\times 7\text{S}_1$ with the same weights used for the three softmax loss layers in [32]. We evaluate our approach in terms of top-1 and top-5 accuracy rate. we adopt the argmax classification scheme.

The baseline is the result of running GoogLeNet implementation in CAFFE. Our LCNN-2 and GoogLeNet are trained on the ImageNet dataset from scratch with the same parameter setting. The results are listed in Table 5. LCNN-2 outperform the baseline in both evaluation metrics with the same parameter setting. Please note that we did not get the same result reported in GoogLeNet [32] by simply running the implementation in CAFFE. Our goal here is to show that as the network becomes deeper, learning good discriminative features for hidden layers might become more difficult solely depending on the prediction error loss. Therefore, adding explicit supervision to late hidden layers under this scenario becomes particularly useful.

5.2.3 Caltech101 Dataset

Caltech101 contains $9,146$ images from 101 object categories and a background category. In this experiment, we test the performance of LCNN with a limited amount of training data, and compare it with several state-of-the-art approaches, including label consistent K-SVD [14].

For fair comparison with previous work, we follow the standard classification settings. During training time, 30 images are randomly chosen from each category to form the training set, and at most 50 images per category are tested. We use the ImageNet trained model from AlexNet in [17] and VGGNet-16 in [29], and fine-tune them on the Caltech101 dataset. We built our LCNN on top of AlexNet and VGGNet-16 respectively in this experiment. The explicit supervision is added to the second fully-connected layer (fc_7). We set the hyperparameter $\alpha = 0.0375$.

The baseline is the result of fine-tuning AlexNet on Caltech101. Then we finetune our LCNN with the same parameter setting and initial model. Similarly, we obtained the baseline* result and LCNN results based on VGGNet-16. The results are summarized in Table 6. With only a limited amount of data available, our approach makes better use of the training data and achieves higher accuracy. LCNN outperforms both the baseline results and other deep learning approaches, representing state-of-the-art on this task.

Method	Accuracy(%)
LC-KSVD [14]	73.6
Zeiler [43]	86.5
Dosovitskiy [4]	85.5
Zhou [45]	87.2
He [9]	91.44
baseline	87.1
LCNN-1 (k-NN)	88.51
LCNN-2 (argmax)	90.11
LCNN-2 (k-NN)	89.45
baseline*	92.5
LCNN-2* (argmax)	93.7
LCNN-2* (k-NN)	93.6

Table 6. Comparisons of LCNN with other approaches on the Caltech101 dataset. The results of [14, 43, 4, 45, 9] are copied from their original papers. The 'baseline' and 'baseline*' are the results by fine-tuning AlexNet model [17] and VGGNet-16 model [29] on Caltech101 dataset, respectively. LCNN-1, LCNN-2 and 'baseline' are trained with the same parameter setting. LCNN-2 and 'baseline*' are trained with the same parameter setting as well.

6. Conclusion

We introduced the Label Consistent Neural Network, a supervised feature learning algorithm, by adding explicit supervision to late hidden layers. By introducing a discriminative representation error and combining it with the traditional prediction error in neural networks, we achieve better classification performance at the output layer, and more discriminative representations at the hidden layers. Experimental results show that our approach operates at the state-of-the-art on several publicly available action and object recognition dataset. It leads to faster convergence speed and works well when only limited video or image data is presented. Our approach can be seamlessly combined with various network architectures. Future work includes applying the discriminative learned category-specific representations to other computer vision tasks besides action and object recognition.

Acknowledgement

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20071. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2006.

[2] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[4] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Bro. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014.

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[7] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[8] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://www.thumos.info/`, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv: 1207.0580*, 2012.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *ICML*, 2010.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678, 2014.

[14] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *CVPR*, 2011.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical Report, 2009.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] Z. Lan, M. Lin, X. L. A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015.

[19] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[21] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[22] F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.

[23] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014.

[24] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[25] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[26] Q. Qiu, Z. Jiang, and R. Chellappa. Sparse dictionary-based representation and recognition of action attributes. In *ICCV*, 2011.

[27] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Rich feature hierarchies for accurate object detection and semantic segmentation. In *ICLR*, 2015.

[28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

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

[30] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012.

[31] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *CVPR*, 2015.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[33] L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[34] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[35] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[36] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. *arXiv: 1507.02159*, 2015.

[37] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *TPAMI*, 31(2):210–227, 2009.

[38] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. *CVPR*, 2015.

[39] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. In *ICCV*, 2011.

[40] S. Yang, P. Luo, C. C. Loy, K. W. Shum, and X. Tang. Deep representation learning with target coding. In *AAAI*, 2015.

[41] H. Ye, Z. Wu, R. Zhao, X. Wang, Y. Jiang, and X. Xue. Evaluating two-stream CNN for video classification. In *ICMR*, 2015.

[42] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

[43] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[44] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. In *BMVC*, 2015.

[45] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.