Deep TEN: Texture Encoding Network

Hang Zhang Jia Xue Kristin Dana

Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, USA
{zhang.hang,jia.xue}@rutgers.edu, kdana@ece.rutgers.edu

Abstract

We propose a Deep Texture Encoding Network (Deep-TEN) with a novel Encoding Layer integrated on top of convolutional layers, which ports the entire dictionary learning and encoding pipeline into a single model. Current methods build from distinct components, using standard encoders with separate off-the-shelf features such as SIFT descriptors or pre-trained CNN features for material recognition. Our new approach provides an end-to-end learning framework, where the inherent visual vocabularies are learned directly from the loss function. The features, dictionaries, encoding representation and the classifier are all learned simultaneously. The representation is orderless and therefore is particularly useful for material and texture recognition. The Encoding Layer generalizes robust residual encoders such as VLAD and Fisher Vectors, and has the property of discarding domain specific information which makes the learned convolutional features easier to transfer. Additionally, joint training using multiple datasets of varied sizes and class labels is supported resulting in increased recognition performance. The experimental results show superior performance as compared to state-of-the-art methods using gold-standard databases such as MINC-2500, Flickr Material Database, KTH-TIPS-2b, and two recent databases 4D-Light-Field-Material and GTOS. The source code for the complete system are publicly available[1].

1. Introduction

With the rapid growth of deep learning, convolutional neural networks (CNNs) has become the de facto standard in many object recognition algorithms. The goals of material and texture recognition algorithms, while similar to object recognition, have the distinct challenge of capturing an orderless measure encompassing some spatial repetition. For example, distributions or histograms of features provide an orderless encoding for recognition. In classic computer vision approaches for material/texture recognition, hand-engineered features are extracted using interest point detectors such as SIFT [31] or filter bank responses [10, 11, 26, 43]. A dictionary is typically learned offline and then the feature distributions are encoded by Bag-of-Words (BoWs) [9, 17, 23, 39], In the final step, a classifier such as SVM is learned for classification. In recent work, hand-engineered features and filter banks are replaced by pre-trained CNNs and BoWs are replaced by the robust residual encoders such as VLAD [22] and its probabilistic version Fisher Vector (FV) [32]. For example, Cimpoi *et al.* [5] assembles different features (SIFT, CNNs) with different encoders (VLAD, FV) and have achieved state-of-the-art results. These existing approaches have the advantage of accepting arbitrary input image sizes and have

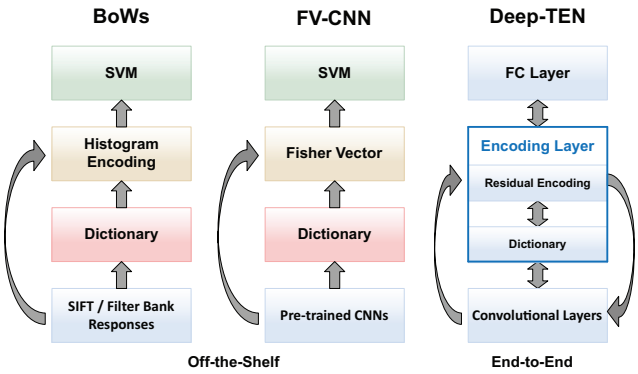

Figure 1: A comparison of classic approaches and the proposed Deep Texture Encoding Network. Traditional methods such as bag-of-words BoW (left) have a structural similarity to more recent FV-CNN methods (center). Each component is optimized in separate steps as illustrated with different colors. In our approach (right) the entire pipeline is learned in an integrated manner, tuning each component for the task at hand (end-to-end texture/material/pattern recognition).

[1] http://ece.rutgers.edu/vision

no issue when transferring features across different domains since the low-level features are generic. However, these methods (both classic and recent work) are comprised of stacking self-contained algorithmic components (feature extraction, dictionary learning, encoding, classifier training) as visualized in Figure 1 (left, center). Consequently, they have the disadvantage that the features and the encoders are fixed once built, so that feature learning (CNNs and dictionary) does not benefit from labeled data. We present a new approach (Figure 1, right) where the entire pipeline is learned in an end-to-end manner.

Deep learning [25] is well known as an end-to-end learning of hierarchical features, so what is the challenge in recognizing textures in an end-to-end way? The convolution layer of CNNs operates in a sliding window manner acting as a local feature extractor. The output featuremaps preserve a relative spatial arrangement of input images. The resulting globally ordered features are then concatenated and fed into the FC (fully connected) layer which acts as a classifier. This framework has achieved great success in image classification, object recognition, scene understanding and many other applications, but is typically not ideal for recognizing textures due to the need for an spatially invariant representation describing the feature distributions instead of concatenation. Therefore, an orderless feature pooling layer is desirable for end-to-end learning. The challenge is to make the loss function differentiable with respect to the inputs and layer parameters. We derive a new back propagation equation series (see Appendix A). In this manner, encoding for an orderless representation can be integrated within the deep learning pipeline.

As the **first contribution** of this paper, we introduce a novel *learnable residual encoding layer* which we refer to as the *Encoding Layer*, that ports the entire dictionary learning and residual encoding pipeline into a single layer for CNN. The Encoding Layer has three main properties. (1) The Encoding Layer generalizes robust residual encoders such as VLAD and Fisher Vector. This representation is orderless and describes the feature distribution, which is suitable for material and texture recognition. (2) The Encoding Layer acts as a pooling layer integrated on top of convolutional layers, accepting arbitrary input sizes and providing output as a fixed-length representation. By allowing arbitrary size images, the Encoding Layer makes the deep learning framework more flexible and our experiments show that recognition performance is often improved with multi-size training. In addition, (3) the Encoding Layer learns an inherent dictionary and the encoding representation which is likely to carry domain-specific information and therefore is suitable for transferring pre-trained features. In this work, we transfer CNNs from object categorization (ImageNet [12]) to material recognition. Since the network is trained end-to-end as a regression, the convolutional features learned together with Encoding Layer on top are easier to transfer (likely to be domain-independent).

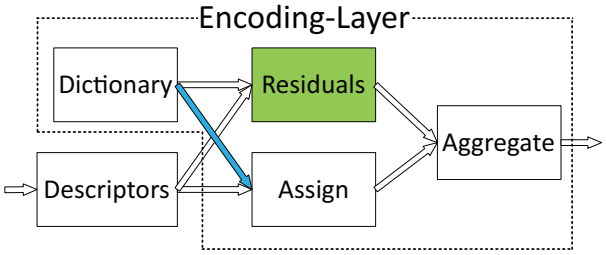

Figure 2: The *Encoding Layer* learns an inherent *Dictionary*. The *Residuals* are calculated by pairwise difference between the input visual descriptors and the codewords of the dictionary. The *Assignment Weights* based on pairwise distance between input descriptors and codewords. Finally, the residual vectors are *aggregated* with the assignment weights.

The **second contribution** of this paper is a new framework for end-to-end material recognition which we refer to as *Texture Encoding Network - Deep TEN*, where the feature extraction, dictionary learning and encoding representation are learned together in a single network as illustrated in Figure 1. Our approach has the benefit of gradient information passing to each component during back propagation, tuning each component for the task at hand. Deep-Ten outperforms existing modular methods and achieves the **state-of-the-art** results on material/texture datasets such as MINC-2500 and KTH-TIPS-2b. Additionally, this Deep Encoding Network performs well in general recognition tasks beyond texture and material as demonstrated with results on MIT-Indoor and Caltech-101 datasets. We also explore how convolutional features learned with Encoding Layer can be transferred through joint training on two different datasets. The experimental result shows that the recognition rate is significantly improved with this joint training.

2. Learnable Residual Encoding Layer

Residual Encoding Model Given a set of N visual descriptors $X = \{x_1, ..x_N\}$ and a learned codebook $C = \{c_1, ...c_K\}$ containing K codewords that are D-dimensional, each descriptor x_i can be assigned with a weight a_{ik} to each codeword c_k and the corresponding residual vector is denoted by $r_{ik} = x_i - c_k$, where $i = 1, ...N$ and $k = 1, ...K$. Given the assignments and the residual vector, the residual encoding model applies an aggregation operation for every single codeword c_k:

$$e_k = \sum_{i=1}^{N} e_{ik} = \sum_{i=1}^{N} a_{ik} r_{ik}. \qquad (1)$$

	Deep Features	Dictionary Learning	Residual Encoding	Any-size	Fine-tuning	End-to-end Classification
BoWs		✓		✓		
Fisher-SVM [41]		✓	✓	✓		
Encoder-CNN (FV [5] VLAD [18]	✓	✓	✓	✓		
CNN	✓				✓	✓
B-CNN [28]	✓				✓	
SPP-Net [19]	✓			✓	✓	✓
Deep TEN (ours)	✓	✓	✓	✓	✓	✓

Table 1: Methods Overview. Compared to existing methods, Deep-Ten has several desirable properties: it integrates deep features with dictionary learning and residual encoding and it allows any-size input, fine-tuning and provides end-to-end classification.

The resulting encoder outputs a fixed length representation $E = \{e_1, ...e_K\}$ (independent of the number of input descriptors N).

Encoding Layer The traditional visual recognition approach can be partitioned into feature extraction, dictionary learning, feature pooling (encoding) and classifer learning as illustrated in Figure 1. In our approach, we port the dictionary learning and residual encoding into a single layer of CNNs, which we refer to as the *Encoding Layer*. The Encoding Layer simultaneously learns the encoding parameters along with with an inherent dictionary in a fully supervised manner. The inherent dictionary is learned from the distribution of the descriptors by passing the gradient through assignment weights. During the training process, the updating of extracted convolutional features can also benefit from the encoding representations.

Consider the assigning weights for assigning the descriptors to the codewords. Hard-assignment provides a single non-zero assigning weight for each descriptor x_i, which corresponds to the nearest codeword. The k-th element of the assigning vector is given by $a_{ik} = \mathbb{1}(\|r_{ik}\|^2 = min\{\|r_{i1}\|^2, ...\|r_{iK}\|^2\})$ where $\mathbb{1}$ is the indicator function (outputs 0 or 1). Hard-assignment doesn't consider the codeword ambiguity and also makes the model non-differentiable. Soft-weight assignment addresses this issue by assigning a descriptor to each codeword [42]. The assigning weight is given by

$$a_{ik} = \frac{\exp(-\beta\|r_{ik}\|^2)}{\sum_{j=1}^{K}\exp(-\beta\|r_{ij}\|^2)}, \tag{2}$$

where β is the smoothing factor for the assignment.

Soft-assignment assumes that different clusters have equal scales. Inspired by guassian mixture models (GMM), we further allow the smoothing factor s_k for each cluster center c_k to be learnable:

$$a_{ik} = \frac{\exp(-s_k\|r_{ik}\|^2)}{\sum_{j=1}^{K}\exp(-s_j\|r_{ij}\|^2)}, \tag{3}$$

which provides a finer modeling of the descriptor distributions. The Encoding Layer concatenates the aggregated residual vectors with assigning weights (as in Equation 1). As is typical in prior work [1, 32], the resulting vectors are normalized using the $L2$-norm.

End-to-end Learning The Encoding Layer is a directed acyclic graph as shown in Figure 2, and all the components are differentiable *w.r.t* the input X and the parameters (codewords $C = \{c_1, ...c_K\}$ and smoothing factors $s = \{s_1, ...s_k\}$). Therefore, the Encoding Layer can be trained end-to-end by standard SGD (stochastic gradient descent) with backpropagation. We provide the details and relevant equation derivations in the Appendix A.

2.1. Relation to Other Methods

Relation to Dictionary Learning Dictionary Learning is usually learned from the distribution of the descriptors in an unsupervised manner. K-means [30] learns the dictionary using hard-assignment grouping. Gaussian Mixture Model (GMM) [15] is a probabilistic version of K-means, which allows a finer modeling of the feature distributions. Each cluster is modeled by a Gaussian component with its own mean, variance and mixture weight. The Encoding Layer makes the inherent dictionary differentiable *w.r.t* the loss function and learns the dictionary in a supervised manner. To see the relationship of the Encoding Layer to K-means, consider Figure 2 with omission of the residual vectors (shown in green of Figure 2) and let smoothing factor $\beta \to \infty$. With these modifications, the Encoding Layer acts like K-means. The Encoding Layer can also be regarded as a simplified version of GMM, that allows different scaling (smoothing) of the clusters. A concurrent work also achieves supervised task-driven dictionary learning [40].

Relation to BoWs and Residual Encoders BoWs (bag-of-word) methods typically hard assign each descriptor to the nearest codeword and counts the occurrence of the visual words by aggregating the assignment vectors $\sum_{i=1}^{N} a_i$ [36]. An improved BoW employs a soft-assignment weights [29]. VLAD [22] aggregates the residual vector with the hard-assignment weights. NetVLAD [22] makes two relaxations: (1) soft-assignment to make

the model differentiable and (2) decoupling the assignment from the dictionary which makes the assigning weights depend only on the input instead of the dictionary. Therefore, the codewords are not learned from the distribution of the descriptors. Considering Figure 2, NetVLAD drops the link between visual words with their assignments (the blue arrow in Figure 2). Fisher Vector [32] concatenates both the 1st order and 2nd order aggregated residuals. FV-CNN [5] encodes off-the-shelf CNNs with pre-trained CNN and achieves good result in material recognition. Fisher Kernel SVM [41] iteratively update the SVM by a convex solver and the inner GMM parameters using gradient descent. A key difference from our work is that this Fisher Kernel method uses hand-crafted instead of learning the features. VLAD-CNN [18] and FV-CNN [5] build off-the-shelf residual encoders with pre-trained CNNs and achieve great success in robust visual recognition and understanding areas.

Relation to Pooling In CNNs, a pooling layer (Max or Avg) is typically used on top of the convolutional layers. Letting $K = 1$ and fixing $c = 0$, the Encoding Layer simplifies to Sum pooling ($e = \sum_{i=1}^{N} x_i$ and $\frac{d_\ell}{d_{x_i}} = \frac{d_\ell}{d_e}$). When followed by $L2$-normalization, it has exactly the same behavior as Avg pooling. The convolutional layers extract features in a sliding window, which can accept arbitrary input image sizes. However, the pooling layers usually have fixed receptive field size, which lead to the CNNs only allowing fixed input image size. SPP pooling layer [19] accepts different size by fixing the pooling bin number instead of receptive field sizes. The relative spatial orders of the descriptors are preserved. Bilinear pooling layer [28] removes the globally ordered information by summing the outer-product of the descriptors across different locations. Our Encoding Layer acts as a pooling layer by encoding robust residual representations, which converts arbitrary input size to a fix length representation. Table 1 summarizes the comparison our approach to other methods.

3. Deep Texture Encoding Network

We refer to the deep convolutional neural network with the Encoding Layer as *Deep Texture Encoding Network (Deep-TEN)*. In this section, we discuss the properties of the Deep-TEN, that is the property of integrating Encoding Layer with an end-to-end CNN architecture.

Domain Transfer Fisher Vector (FV) has the property of discarding the influence of frequently appearing features in the dataset [32], which usually contains domain specific information [49]. FV-CNN has shown its domain transfer ability practically in material recognition work [5]. Deep-TEN generalizes the residual encoder and also preserves

	output size	Deep-TEN 50
Conv1	176×176×64	7×7, stride 2
Res1	88×88×256	3 × 3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
Res2	44×44×512	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
Res3	22×22×1024	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
Res4	11×11×2048	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
Projection + Reshape	121×128	conv 1×1, 2048⇒128
		W×H×D⇒N×D
Encoding	32×128	32 codewords
$L2$-norm + FC	n classes	1×1 FC

Table 2: Deep-TEN architectures for adopting 50 layer pre-trained ResNet. The 2nd column shows the featuremap sizes for input image size of 352×352. When multi-size training for input image size 320×320, the featuremap after Res4 is 10×10. We adopt a 1×1 convolutional layer after Res4 to reduce number of channels.

this property. To see this intuitively, consider the following: when a visual descriptor x_i appears frequently in the data, it is likely to be close to one of the visual centers c_k. Therefore, the resulting residual vector corresponding to c_k, $r_{ik} = x_i - c_k$, is small. For the residual vectors of r_{ij} corresponding to c_j where $j \neq k$, the corresponding assigning weight a_{ij} becomes small as shown in Equation 3. The Encoding Layer aggregates the residual vectors with assignment weights and results in small values for frequently appearing visual descriptors. This property is essential for transferring features learned from different domain, and in this work we transfer CNNs pre-trained on the object dataset ImageNet to material recognition tasks.

Traditional approaches do not have domain transfer problems because the features are usually generic and the domain-specific information is carried by the dictionary and encoding representations. The proposed Encoding Layer generalizes the dictionary learning and encoding framework, which carries domain-specific information. Because the entire network is optimized as a regression progress, the resulting convolutional features (with Encoding Layer learned on top) are likely to be domain-independent and therefore easier to transfer.

Multi-size Training CNNs typically require a fixed input image size. In order to feed into the network, images have to be resized or cropped to a fixed size. The convolutional layers act in a sliding window manner, which can allow any input sizes (as discussed in SPP [19]). The FC (fully connected) layer acts as a classifier which take a fix length representation as input. Our Encoding Layer act as a pooling layer on top of the convolutional layers, which converts arbitrary input sizes to a fixed length representation. Our experiments show that the classification results are often improved by iteratively training the Deep Encoding Network with different image sizes. In addition, this multi-size training provides the opportunity for cross dataset training.

Joint Deep Encoding There are many labeled datasets for different visual problems, such as object classification [6, 12, 24], scene understanding [45, 47], object detection [14, 27] and material recognition [2, 48]. An interesting question to ask is: how can different visual tasks benefit each other? Different datasets have different domains, different labeling strategies and sometimes different image sizes (e.g. CIFAR10 [24] and ImageNet [12]). Sharing convolutional features typically achieves great success [19, 34]. The concept of *multi-task learning* [37] was originally proposed in [8], to jointly train cross different datasets. An issue in joint training is that features from different datasets may not benefit from the combined training since the images contain domain-specific information. Furthermore, it is typically not possible to learn deep features from different image sizes. Our Encoding Layer on top of convolution layers accepts arbitrary input image sizes and learns domain independent convolutional features, enabling convenient joint training. We present and evaluate a network that shares convolutional features for two different datasets and has two separate Encoding Layers. We demonstrate joint training with two datasets and show that recognition results are significantly improved.

4. Experimental Results

Datasets The evaluation considers five material and texture datasets. *Materials in Context Database* (MINC) [2] is a large scale material in the wild dataset. In this work, a publicly available subset (MINC-2500, Sec 5.4 of original paper) is evaluated with provided train-test splits, containing 23 material categories and 2,500 images per-category. *Flickr Material Dataset* (FMD) [35], a popular benchmark for material recognition containing 10 material classes, 90 images per-class used for training and 10 for test. *Ground Terrain in Outdoor Scenes Dataset* (GTOS) [46] is a dataset of ground materials in outdoor scene with 40 categories. The evaluation is based on provided train-test splits. *KTH-TIPS-2b* (KTH)- [3], contains 11 texture categories and four samples per-category. Two samples are randomly

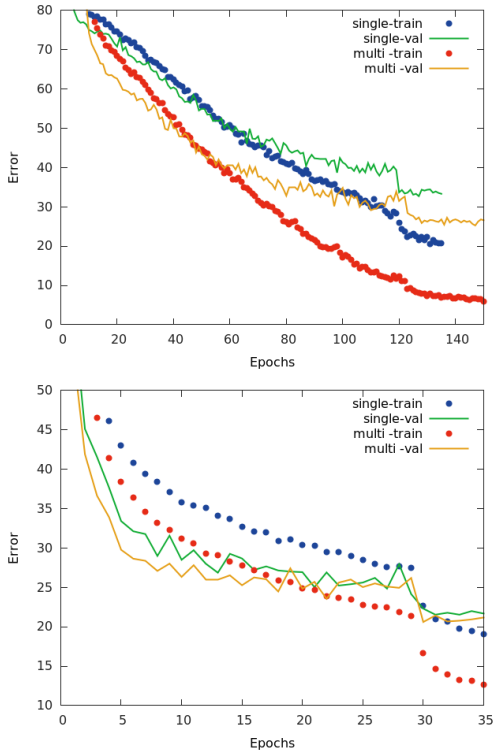

Figure 3: Comparison between single-size training and multi-size training. Iteratively training Deep-TEN with two different intput sizes (352×352 and 320×320) makes the network converging faster and improves the performance. The top figure shows the training curve on MIT-Indoor and the bottom one shows the first 35 epochs on MINC-2500.

picked for training and the others for test. *4D-Light-Field-Material* (4D-Light) [44] is a recent light-field material dataset containing 12 material categories with 100 samples per-category. In this experiment, 70 randomly picked samples per-category are used as training and the others for test and only one angular resolution is used per-sample. For general classification evaluations, two additional datasets are considered. *MIT-Indoor* [33] dataset is an indoor scene categorization dataset with 67 categories, a standard subset of 80 images per-category for training and 20 for test is used in this work. *Caltech 101* [16] is a 102 category (1 for background) object classification dataset; 10% randomly picked samples are used for test and the others for training.

Baselines In order to evaluate different encoding and representations, we benchmark different approaches with single input image sizes without ensembles, since we expect that the performance is likely to improve by assembling features or using multiple scales. We fix the input image size to 352×352 for SIFT, pre-trained CNNs feature extractions

	MINC-2500	FMD	GTOS	KTH	4D-Light	MIT-Indoor	Caltech-101
FV-SIFT	46.0	47.0	65.5	66.3	58.4	51.6	63.4
FV-CNN (VGG-VD)	61.8	75.0	77.1	71.0	70.4	67.8	83.0
Deep-TEN (ours)	**80.6**	**80.2**$_{\pm 0.9}$	**84.3**$_{\pm 1.9}$	**82.0**$_{\pm 3.3}$	**81.7**$_{\pm 1.0}$	**71.3**	**85.3**

Table 3: The table compares the recognition results of Deep-TEN with off-the-shelf encoding approaches, including Fisher Vector encoding of dense SIFT features (FV-SIFT) and pre-trained CNN activations (FV-CNN) on different datasets using single-size training. Top-1 test accuracy mean±std % is reported and the best result for each dataset is marked bold. (The results of Deep-TEN for FMD, GTOS, KTH datasets are based on 5-time statistics, and the results for MINC-2500, MIT-Indoor and Caltech-101 datasets are averaged over 2 runs. The baseline approaches are based on 1-time run.)

	MINC-2500	FMD	GTOS	KTH	4D-Light	MIT-Indoor
FV-CNN (VGG-VD) multi	63.1	74.0	79.2	77.8	76.5	67.0
FV-CNN (ResNet) multi	69.3	78.2	77.1	78.3	77.6	76.1
Deep-TEN (ours)	80.6	**80.2**$_{\pm 0.9}$	84.3$_{\pm 1.9}$	82.0$_{\pm 3.3}$	**81.7**$_{\pm 1.0}$	71.3
Deep-TEN (ours) multi	**81.3**	78.8$_{\pm 0.8}$	**84.5**$_{\pm 2.9}$	**84.5**$_{\pm 3.5}$	81.4$_{\pm 2.6}$	**76.2**

Table 4: Comparison of single-size and multi-size training.

and Deep-TEN. FV-SIFT, a non-CNN approach, is considered due to its similar encoding representations. SIFT features of 128 dimensions are extracted from input images and a GMM of 128 Gaussian components is built, resulting in a 32K Fisher Vector encoding. For FV-CNN encoding, the CNN features of input images are extracted using pre-trained 16-layer VGG-VD model [38]. The feature maps of conv5 (after ReLU) are used, with the dimensionality of 14×14×512. Then a GMM of 32 Gaussian components is built and resulting in a 32K FV-CNN encoding. To improve the results further, we build a stronger baseline using pre-trained 50-layers ResNet [20] features. The feature maps of the last residual unit are used. The extracted features are projected into 512 dimension using PCA, from the large channel numbers of 2048 in ResNet. Then we follow the same encoding approach of standard FV-CNN to build with ResNet features. For comparison with multi-size training Deep-TEN, multi-size FV-CNN (VD) is used, the CNN features are extracted from two different sizes of input image, 352×352 and 320×320 (sizes determined empirically). All the baseline encoding representations are reduced to 4096 dimension using PCA and $L2$-normalized. For classification, linear one-vs-all Support Vector Machines (SVM) are built using the off-the-shelf representations. The learning hyper-parameter is set to $C_{svm} = 1$, since the features are $L2$-normalized. The trained SVM classifiers are recalibrated as in prior work [5, 28], by scaling the weights and biases such that the median prediction score of positive and negative samples are at $+1$ and -1.

Deep-TEN Details We build Deep-TEN with the architecture of an Encoding Layer on top of 50-layer pre-trained ResNet (as shown in Table 2). Due to high-dimensionality of ResNet feature maps on Res4, a 1×1 convolutional layer is used for reducing number of channels (2048⇒128). Then an Encoding Layer with 32 codewords is added on top, followed by $L2$-normalization and FC layer. The weights (codewords C and smoothing factor s) are randomly initialized with uniform distribution $\pm\frac{1}{\sqrt{K}}$. For data augmentation, the input images are resized to 400 along the short-edge with the per-pixel mean subtracted. For in-the-wild image database, the images are randomly cropped to 9% to 100% of the image areas, keeping the aspect ratio between $3/4$ and $4/3$. For the material database with in-lab or controlled conditions (KTH or GTOS), we keep the original image scale. The resulting images are then resized into 352×352 for single-size training (and 320×320 for multi-size training), with 50% chance horizontal flips. Standard color augmentation is used as in [25]. We use SGD with a mini-batch size of 64. For fine-tuning, the learning rate starts from 0.01 and divided by 10 when the error plateaus. We use a weight decay of 0.0001 and a momentum of 0.9. In testing, we adopt standard 10-crops [25].

Multi-size Training Deep-TEN ideally can accept arbitrarily input image sizes (larger than a constant). In order to learn the network without modifying the standard optimization solver, we train the network with a pre-defined size in each epoch and iteratively change the input image size for every epoch as in [19]. A full evaluation of combinatorics of different size pairs have not yet been explored. Empirically, we consider two different sizes 352×352 and 320×320 during the training and only use single image size in testing for simplicity (352×352). The two input sizes result in 11×11

	MINC-2500	FMD	GTOS	KTH	4D-Light
Deep-TEN* (ours)	**81.3**	$80.2_{\pm 0.9}$	**84.5**$_{\pm 2.9}$	**84.5**$_{\pm 3.5}$	**81.7**$_{\pm 1.0}$
State-of-the-Art	$76.0_{\pm 0.2}$ [2]	**82.4**$_{\pm 1.4}$ [5]	N/A	$81.1_{\pm 1.5}$ [4]	$77.0_{\pm 1.1}$ [44]

Table 5: Comparison with state-of-the-art on four material/textures dataset (GTOS is a new dataset, so SoA is not available). Deep-TEN* denotes the best model of Deep Ten and Deep Ten multi.

and 10×10 feature map sizes before feeding into the Encoding Layer. Our goal is to evaluate how multi-size training affects the network optimization and how the multi-scale features affect texture recognition.

4.1. Recognition Results

We evaluate the performance of Deep-TEN, FV-SIFT and FV-CNN on aforementioned golden-standard material and texture datasets, such as MINC-2500, FMD, KTH and two new material datasets: 4D-Light and GTOS. Additionally, two general recognition datasets MIT-Indoor and Caltech-101 are also considered. Table 3 shows overall experimental results using single-size training,

Comparing with Baselines As shown in Table 3, Deep-TEN and FV-CNN always outperform FV-SIFT, which shows that pre-trained CNN features are typically more discriminant than hand-engineered SIFT features. FV-CNN usually achieves reasonably good results on different datasets without fine-tuning pre-trained features. We can observe that the performance of FV-CNN is often improved by employing ResNet features comparing with VGG-VD as shown in Table 4. Deep-TEN outperforms FV-CNN under the same settings, which shows that the Encoding Layer gives the advantage of transferring pre-trained features to material recognition by removing domain-specific information as described in Section 3. The Encoding Layer's property of representing feature distributions is especially good for texture understanding and segmented material recognition. Therefore, Deep-TEN works well on GTOS and KTH datasets. For the small-scale dataset FMD with less training sample variety, Deep-TEN still outperforms the baseline approaches that use an SVM classifier. For MINC-2500, a relatively large-scale dataset, the end-to-end framework of Deep TEN shows its distinct advantage of optimizing CNN features and consequently, the recognition results are significantly improved (61.8%⇒80.6% and 69.3%⇒81.3, compared with off-the-shelf representation of FV-CNN). For the MIT-Indoor dataset, the Encoding Layer works well on scene categorization due to the need for a certain level of orderless and invariance. The best performance of these methods for Caltech-101 is achieved by FV-CNN(VD) multi (85.7% omitted from the table). The CNN models VGG-VD and ResNet are pre-trained on ImageNet, which is also an object classification dataset like Caltech-101. The pre-trained features are discriminant to target datasets. Therefore, Deep-TEN performance is only slightly better than the off-the-shelf representation FV-CNN.

Impact of Multi-size For in-the-wild datasets, such as MINC-2500 and MIT-Indoor, the performance of all the approaches are improved by adopting multi-size as expected. Remarkably, as shown in Table 4, Deep-TEN shows a performance boost of 4.9% using multi-size training and outperforms the best baseline by 7.4% on MIT-Indoor dataset. For some datasets such as FMD and GTOS, the performance decreases slighltly by adopting multi-size training due to lack of variety in the training data. Figure 3 compares the single-size training and multi-size (two-size) training for Deep-TEN on MIT-Indoor and MINC-2500 dataset. The experiments show that multi-size training helps the optimization of the network (converging faster) and the learned multi-scale features are useful for the recognition.

Comparison with State-of-the-Art As shown in Table 5, Deep-TEN outperforms the state-of-the-art on four material/texture recognition datasets: MINC-2500, KTH, GTOS and 4D-Light. Deep-TEN also performs well on two general recognition datasets. Notably, the prior state-of-the-art approaches either (1) relies on assembling features (such as FV-SIFT & CNNs) and/or (2) adopts an additional SVM classifier for classification. Deep-TEN as an end-to-end framework neither concatenates any additional hand-engineered features nor employs SVM for classification. For the small-scale datasets such as FMD and MIT-Indoor (subset), the proposed Deep-TEN gets compatible results with state-of-the-art approaches (FMD within 2%, MIT-indoor within 4%). For the large-scale datasets such as MINC-2500, Deep-TEN outperforms the prior work and baselines by a large margin demonstrating its great advantage of end-to-end learning and the ability of transferring pre-trained CNNs. We expect that the performance of Deep-TEN can scale better than traditional approaches when adding more training data.

4.2. Joint Encoding from Scratch

We test joint training on two small datasets CIFAR-10 [24] and STL-10 [6] as a litmus test of Joint Encoding from scratch. We expect the convolutional features learned

	STL-10	CIFAR-10
Deep-TEN (Individual)	76.29	91.5
Deep-TEN (Joint)	**87.11**	91.8
State-of-the-Art	74.33 [51]	-

Table 6: Joint Encoding on CIFAR-10 and STL-10 datasets. Top-1 test accuracy %. When joint training with CIFAR-10, the recognition result on STL-10 is significantly improved. (Note that traditional network architecture does not allow joint training with different image sizes.)

with Encoding Layer are easier to transfer, and can improve the recognition on both datasets.

CIFAR-10 contains 60,000 tiny images with the size 32×32 belonging to 10 classes (50,000 for training and 10,000 for test), which is a subset of tiny images database. STL-10 is a dataset acquired from ImageNet [12] and originally designed for unsupervised feature learning, which has 5,000 labeled images for training and 8,000 for test with the size of 96×96. For the STL-10 dataset only the labeled images are used for training. Therefore, learning CNN from scratch is not supposed to work well due to the limited training data. We make a very simple network architecture, by simply replacing the 8×8 Avg pooling layer of pre-Activation ResNet-20 [21] with Encoding-Layer (16 codewords). We then build a network with shared convolutional layers and separate encoding layers that is jointly trained on two datasets. Note that the traditional CNN architecture is not applicable due to different image sizes from this two datasets. The training loss is computed as the sum of the two classification losses, and the gradient of the convolutional layers are accumulated together. For data augmentation in the training: 4 pixels are padded on each side for CIFAR-10 and 12 pixels for STL-10, and then randomly crop the padded images or its horizontal flip into original sizes 32×32 for CIFAR-10 and 96×96 for STL-10. For testing, we only evaluate the single view of the original images. The model is trained with a mini batch of 128 for each dataset. We start with a learning rate of 0.1 and divide it by 10 and 100 at 80^{th} and 120^{th} epoch.

The experimental results show that the recognition result of STL-10 dataset is significantly improved by joint training the Deep TEN with CIFAR-10 dataset. Our approach achieves the recognition rate of 87.11%, which outperforms previous the state of the art 74.33% [51] and 72.8% [13] by a large margin.

5. Conclusion

In summary, we developed an Encoding Layer and built the network Deep-TEN and demonstrated the effectiveness on various material and texture recognition datasets. we developed an Encoding Layer which bridges the gap between classic computer vision approaches and the CNN architecture. This layer has two main advantages: (1) the resulting deep learning framework is more flexible by allowing arbitrary input image size, and (2) the learned convolutional features are easier to transfer since the Encoding Layer is likely to carry domain-specific information. The Encoding Layer shows superior performance in transferring pre-trained CNN features. Deep-TEN outperforms traditional off-the-shelf methods and achieves state-of-the-art results on MINC-2500, KTH and two recent material datasets: GTOS and 4D-Lightfield.

The Encoding Layer is efficient using GPU computations and our Torch [7] implementation of 50-layer Deep-Ten (as shown in Table 2) takes the input images of size 352×352, runs at 55 frame/sec for training and 290 frame/sec for inference on 4 Titan X Maxwell GPUs.

Acknowledgment

We thank Wenhan Zhang from Physics department, Rutgers University for discussions of mathematic models. This work was supported by National Science Foundation award IIS-1421134. A GPU used for this research was donated by the NVIDIA Corporation.

A. Encoding Layer Implementations

We provide the explicit expression for the gradients of the loss ℓ with respect to (*w.r.t*) the layer input and the parameters for implementing Encoding Layer in [50].

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *arXiv preprint arXiv:1511.07247*, 2015. 3

[2] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3479–3487, 2015. 5, 7

[3] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1597–1604. IEEE, 2005. 5

[4] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, 118(1):65–94, 2016. 7

[5] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3836, 2015. 1, 3, 4, 6, 7

[6] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001(48109):2, 2010. 5, 7

[7] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 8

[8] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 5

[9] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004. 1

[10] O. G. Cula and K. J. Dana. Compact representation of bidirectional texture functions. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:1041–1067, December 2001. 1

[11] O. G. Cula and K. J. Dana. Recognition methods for 3d textured surfaces. *Proceedings of SPIE Conference on Human Vision and Electronic Imaging VI*, 4299:209–220, January 2001. 1

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2, 5, 8

[13] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014. 8

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 5

[15] B. S. Everitt. *Finite mixture distributions*. Wiley Online Library, 1981. 3

[16] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. 5

[17] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE, 2005. 1

[18] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer, 2014. 3, 4

[19] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014. 3, 4, 5, 6

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 6

[21] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 8

[22] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010. 1, 3

[23] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998. 1

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *University of Toronto, Technical Report*, 2009. 5, 7

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2, 6

[26] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001. 1

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 5

[28] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015. 3, 4, 6

[29] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *2011 International Conference on Computer Vision*, pages 2486–2493. IEEE, 2011. 3

[30] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 3

[31] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1

[32] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010. 1, 3, 4

[33] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE, 2009. 5

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 5

[35] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson. Recognizing materials using perceptually inspired features. *International journal of computer vision*, 103(3):348–371, 2013. 5

[36] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*, pages 1–15. Springer, 2006. 3

[37] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances*

in Neural Information Processing Systems, pages 568–576, 2014. 5

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

[39] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 370–377. IEEE, 2005. 1

[40] X. Sun, N. M. Nasrabadi, and T. D. Tran. Supervised multilayer sparse coding networks for image classification. *arXiv preprint arXiv:1701.08349*, 2017. 3

[41] V. Sydorov, M. Sakurada, and C. H. Lampert. Deep fisher kernels-end to end learning of the fisher kernel gmm parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1402–1409, 2014. 3, 4

[42] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV 2008, PART III. LNCS*, pages 696–709. Springer, 2008. 3

[43] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *European Conference on Computer Vision*, pages 255–271. Springer, 2002. 1

[44] T.-C. Wang, J.-Y. Zhu, E. Hiroaki, M. Chandraker, A. Efros, and R. Ramamoorthi. A 4D light-field dataset and CNN architectures for material recognition. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 5, 7

[45] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010. 5

[46] J. Xue, H. Zhang, K. Dana, and K. Nishino. Differential angular imaging for material recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[47] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5

[48] H. Zhang, K. Dana, and K. Nishino. Reflectance hashing for material recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3071–3080, 2015. 5

[49] H. Zhang, K. Dana, and K. Nishino. Friction from reflectance: Deep reflectance codes for predicting physical surface properties from one-shot in-field reflectance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 4

[50] H. Zhang, J. Xue, and K. Dana. Deep ten: Texture encoding network. *arXiv preprint arXiv:1612.02844*, 2016. 8

[51] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 8