Visualizing and Understanding Deep Texture Representations

Tsung-Yu Lin
University of Massachusetts, Amherst
tsungyulin@cs.umass.edu

Subhransu Maji
University of Massachusetts, Amherst
smaji@cs.umass.edu

Abstract

A number of recent approaches have used deep convolutional neural networks (CNNs) to build texture representations. Nevertheless, it is still unclear how these models represent texture and invariances to categorical variations. This work conducts a systematic evaluation of recent CNN-based texture descriptors for recognition and attempts to understand the nature of invariances captured by these representations. First we show that the recently proposed bilinear CNN model [25] is an excellent general-purpose texture descriptor and compares favorably to other CNN-based descriptors on various texture and scene recognition benchmarks. The model is translationally invariant and obtains better accuracy on the ImageNet dataset without requiring spatial jittering of data compared to corresponding models trained with spatial jittering. Based on recent work [13, 28] we propose a technique to visualize pre-images, providing a means for understanding categorical properties that are captured by these representations. Finally, we show preliminary results on how a unified parametric model of texture analysis and synthesis can be used for attribute-based image manipulation, e.g. to make an image more swirly, honeycombed, or knitted. The source code and additional visualizations are available at http://vis-www.cs.umass.edu/texture.

1. Introduction

The study of texture has inspired many of the early representations of images. The idea of representing texture using the statistics of image patches have led to the development of "textons" [21, 24], the popular "bag-of-words" models [6] and their variants such as the Fisher vector [30] and VLAD [19]. These fell out of favor when the latest generation of deep Convolutional Neural Networks (CNNs) showed significant improvements in recognition performance over a wide range of visual tasks [2, 14, 20, 33]. Recently however, the interest in texture descriptors have been revived by architectures that combine aspects of texture representations with CNNs. For instance, Cimpoi *et al.* [4] showed that Fisher vectors built on top of CNN activations
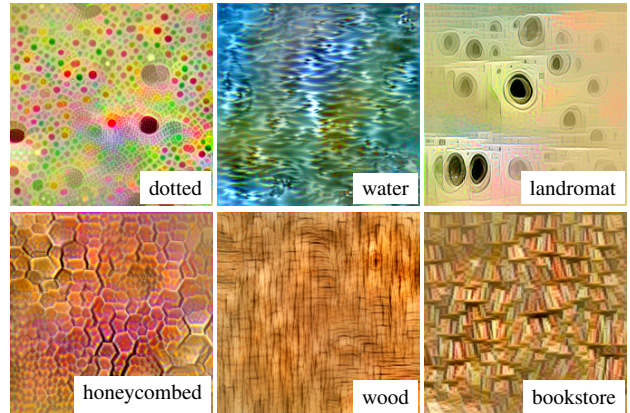

Figure 1. **How is texture represented in deep models?** Visualizing various categories by inverting the bilinear CNN model [25] trained on DTD [3], FMD [34], and MIT Indoor dataset [32] (each column from left to right). These images were obtained by starting from a random image and adjusting it though gradient descent to obtain high log-likelihood for the given category label using a multi-layer bilinear CNN model (See Sect. 2 for details). *Best viewed in color and with zoom.*

lead to better accuracy and improved domain adaptation not only for texture recognition, but also for scene categorization, object classification, and fine-grained recognition.

Despite their success little is known how these models represent *invariances* at the image and category level. Recently, several attempts have been made in order to understand CNNs by visualizing the layers of a trained model [40], studying the invariances by inverting the model [8, 28, 35], and evaluating the performance of CNNs for various recognition tasks. *In this work we attempt to provide a similar understanding for CNN-based texture representations*. Our starting point is the bilinear CNN model of our previous work [25]. The technique builds an orderless image representation by taking the location-wise outer product of image features extracted from CNNs and aggregating them by averaging. The model is closely related to Fisher vectors but has the advantage that gradients of the model can be easily computed allowing fine-tuning and inversion. Moreover, when the two CNNs are identical the bi-

1

linear features capture correlations between filter channels, similar to the early work in parametric texture representation of Portilla and Simoncelli [31].

Our *first contribution* is a systematic study of bilinear CNN features for texture recognition. Using the *Flickr Material Dataset* (FMD) [34], *Describable Texture Dataset* (DTD) [3] and *KTH-T2b* [1] we show that it performs favorably to Fisher vector CNN model [4], which is the current state of the art. Similar results are also reported for scene classification on the *MIT indoor scene* dataset [32]. We also evaluate the role of different layers, effect of scale, and fine-tuning for texture recognition. Our experiments reveal that multi-scale information helps, but also that features from different layers are complementary and can be combined to improve performance in agreement with recent work [16, 26, 29].

Our *second contribution* is to investigate the role of translational invariance of these models due to the orderless pooling of local descriptors. Recently, we showed [25] that bilinear CNNs initialized by pre-training a standard CNN (*e.g.*, VGG-M [2]) on ImageNet, truncating the network at a lower convolutional layer (*e.g.*, *conv5*), adding bilinear pooling modules, followed by domain-specific fine-tuning leads to significant improvements in accuracy for a number of fine-grained recognition tasks. These models capture localized feature interactions in a translationally invariant manner which is useful for making fine-grained distinctions between species of birds or models of cars. However, it remains unclear what the tradeoffs are between explicit translational invariance in these models versus implicit invariance obtained by spatial jittering of data during training. To this end we conduct experiments on the ImageNet LSVRC 2012 dataset [7] by training several models using different amounts of data augmentation. Our experiments reveal that bilinear CNN models can be trained from *scratch*, resulting in better accuracy *without* requiring spatial jittering of data than the corresponding CNN architectures that consist of standard "fully-connected" layers trained with jittering.

Our *third contribution* is a technique to "invert" these models to construct invariant inputs and visualize pre-images for categories. Fig. 1 shows the inverse images for various categories – *materials* such as wood and water, *describable attributes* such as honeycombed and dotted, and *scene categories* such as laundromat and bookstore. These images reveal what categorical properties are learned by these texture models. Recently, Gatys *et al*. [13] showed that bilinear features (they call it the Gram matrix representation) extracted from various layers of the "verydeep VGG network" [36] can be inverted for texture synthesis. The synthesized results are visually appealing, demonstrating that the convolutional layers of a CNN capture textural properties significantly better than the first and second order statistics of wavelet coefficients of Portilla and Simoncelli. However, the approach remains impractical since it requires

hundreds of CNN evaluations and is orders of magnitude slower than non-parametric patch-based methods such as image quilting [9]. We show that the two approaches are complementary and one can significantly speed up the convergence of gradient-based inversion by initializing the inverse image using image quilting. The global adjustment of the image through gradient descent also removes many artifacts that quilting introduces.

Finally, we show a novel application of our approach for *image manipulation* with *texture attributes*. A unified parametric model of texture representation and recognition allows us to adjust an image with high-level attributes – to make an image more *swirly* or *honeycombed*, or generate hybrid images that are a combination of multiple texture attributes, *e.g.*, *chequered* and *interlaced*.

1.1. Related work

Texture recognition is a widely studied area. Current state-of-the-art results on texture and material recognition are obtained by hybrid approaches that build orderless representations on top of CNN activations. Cimpoi *et al*. [4] use Fisher vector pooling for material and scene recognition, Gong *et al*. [15] use VLAD pooling for scene recognition, *etc*. Our previous work [25] proposed a general orderless pooling architecture called the *bilinear CNN* that outperforms Fisher vector on many fine-grained datasets. These descriptors are inspired by early work on texture representations [6, 24, 30, 19] that were built on top of wavelet coefficients, linear filter bank responses, or SIFT features [27].

Texture synthesis has received attention from both the vision and graphics communities due to its numerous applications. Heeger and Bergen [17] synthesized texture images by histogram matching. Portilla and Simoncelli were one of the early proponents of parametric approaches. The idea is to represent texture as the first and second order statistics of various filter bank responses (*e.g.*, wavelets, steerable pyramids, *etc*.). However, these approaches were outperformed by simpler non-parametric approaches. For instance, Efros and Lueng [10] proposed a pixel-by-pixel synthesis approach based on sampling similar patches – the method was simple and effective for a wide range of textures. Later, Efros and Freeman proposed a quilting-based approach that was significantly faster [9]. A number of other non-parametric approaches have been proposed for this problem [23, 39]. Recently, Gatys *et al*. showed that replacing the linear filterbanks by CNN filterbanks results in better reconstructions. Notably, the Gram matrix representation used in their approach is identical to the bilinear CNN features of Lin *et al*., suggesting that these features might be good for texture recognition as well. However for synthesis, the parametric approaches remain impractical as they are orders of magnitude slower than non-parametric approaches.

Understanding CNNs through visualizations has also been widely studied given their remarkable performance. Zieler and Fergus [40] visualize CNNs using the top activations of filters and show per-pixel heatmaps by tracking the position of the highest responses. Simonyan and Zisserman [35] visualize parts of the image that cause the highest change in class labels computed by back-propagation. Mahendran and Vedaldi [28] extend this approach by introducing natural image priors which result in inverse images that have fewer artifacts. Dosovitskiy and Brox [8] propose a "deconvolutional network" to invert a CNN in a feed-forward manner. However, the approach tends to produce blurry images since the inverse is not uniquely defined.

Our approach is also related to prior work on editing images based on example images. Ideas from patch-based texture synthesis have been extended in a number of ways to modify the style of the image based on an example [18], adjusting texture synthesis based on the content of another image [5, 9], *etc*. Recently, in a separate work, Gatys *et al*. [12] proposed a "neural style" approach that combines ideas from inverting CNNs with their work on texture synthesis. They generate images that match the style and content of two different images producing compelling results. Although the approach is not practical compared to existing patch-based methods for editing styles, it provides a basis for a rich parametric model of texture. We describe an novel approach to manipulate images with high-level attributes and show several examples of editing images with texture attributes. There is relatively little prior work on manipulating the content of an image using semantic attributes.

2. Methodology and overview

We describe our framework for parametric texture recognition, synthesis, inversion, and attribute-based manipulation using CNNs. For an image $\mathcal{I}$ one can compute the activations of the CNN at a given layer r_i to obtain a set of features $F_{r_i} = \{f_j\}$ indexed by their location j. The bilinear feature $B_{r_i}(\mathcal{I})$ of F_{r_i} is obtained by computing the outer product of each feature f_j with itself and aggregating them across locations by averaging, *i.e.*,

$$B_{r_i}(\mathcal{I}) = \frac{1}{N} \sum_{j=1}^{N} f_j f_j^T. \quad (1)$$

The bilinear feature (or the Gram matrix representation) is an *orderless representation* of the input image and hence is suitable for modeling texture. Let $r_i, i = 1, \ldots, n$, be the index of the ith layer with the bilinear feature representation B_{r_i}. Gatys *et al*. [13] propose a method for texture synthesis from an input image $\mathcal{I}$ by obtaining an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ that matches the bilinear features at various layers by solving the following optimization:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \alpha_i L_1 \left(B_{r_i}, \hat{B}_{r_i} \right) + \gamma \Gamma(\mathbf{x}). \quad (2)$$

Here, $\hat{B}_{r_i} = B_{r_i}(\mathcal{I})$, α_i is the weight of the ith layer, $\Gamma(\mathbf{x})$ is a natural image prior such as the total variation norm (TV norm), and γ is the weight on the prior. Note that we have dropped the implicit dependence of B_{r_i} on $\mathbf{x}$ for brevity. Using the squared loss-function $L_1(x, y) = \sum (x_i - y_i)^2$ and starting from a random image where each pixel initialized with a *i.i.d* zero mean Gaussian, a local optimum is reached through gradient descent. The authors employ L-BFGS, but any other optimization method can be used (*e.g*., Mahendran and Vedaldi [28] use stochastic gradient descent with momentum).

Prior work on minimizing the reconstruction error with respect to the "un-pooled" features F_{r_i} has shown that the *content* of the image in terms of the color and spatial structure is well-preserved even in the higher convolutional layers. Recently, Gatys *et al*. in a separate work [12] synthesize images that match the style and content of two different images $\mathcal{I}$ and $\mathcal{I}'$ respectively by minimizing a weighted sum of the texture and content reconstruction errors:

$$\min_{\mathbf{x}} \lambda L_1 \left(F_s, \hat{F}_s \right) + \sum_{i=1}^{n} \alpha_i L_1 \left(B_{r_i}, \hat{B}_{r_i} \right) + \gamma \Gamma(\mathbf{x}). \quad (3)$$

Here $\hat{F}_s = F_s(\mathcal{I}')$ are features from a layer s from which the target content features are computed for an image $\mathcal{I}'$.

The bilinear features can also be used for predicting attributes by first normalizing the features (signed square-root and ℓ_2) and training a linear classifier in a supervised manner [25]. Let $l_i : i = 1, \ldots, m$ be the index of the ith layer from which we obtain attribute prediction probabilities C_{l_i}. The prediction layers may be different from those used for texture synthesis. Given a target attribute $\hat{C}$ we can obtain an image that matches the target label and is similar to the texture of a given image $\mathcal{I}$ by solving the following optimization:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \alpha_i L_1 \left(B_{r_i}, \hat{B}_{r_i} \right) + \beta \sum_{i=1}^{m} L_2 \left(C_{l_i}, \hat{C} \right) + \gamma \Gamma(\mathbf{x}). \quad (4)$$

Here, L_2 is a loss function such as the *negative log-likelihood* of the label $\hat{C}$ and β is a tradeoff parameter. If multiple targets $\hat{C}_j$ are available then the losses can be blended with weights β_j resulting in the following optimization:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \alpha_i L_1 \left(B_{r_i}, \hat{B}_{r_i} \right) + \beta_j \sum_{i=1, j}^{m} L_2 \left(C_{l_i}, \hat{C}_j \right) + \gamma \Gamma(\mathbf{x}). \quad (5)$$

Implementation details. We use the 16-layer VGG network [36] trained on ImageNet for all our experiments. For the image prior $\Gamma(\mathbf{x})$ we use the TV_β norm with $\beta = 2$:

$$\Gamma(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \right)^{\frac{\beta}{2}} . \quad (6)$$

The exponent $\beta = 2$ was empirically found to lead to better reconstructions in [28] as it leads to fewer "spike" artifacts than $\beta = 1$. In all our experiments, given an input image we resize it to 224×224 pixels before computing the target bilinear features and solve for $\mathbf{x} \in \mathbb{R}^{224 \times 224 \times 3}$. This is primarily for speed since the size of the bilinear features are independent of the size of the image. Hence, an output of any size can be obtained by minimizing Eqn. 5. We use L-BFGS for optimization and compute the gradients of the objective with respect to $\mathbf{x}$ using back-propagation. One detail we found to be critical for good reconstructions is that we ℓ_1 normalize the gradients with respect to each of the L_1 loss terms to balance the losses during optimization. Mahendran and Vedaldi [28] suggest normalizing each L_1 loss term by the ℓ_2 norm of the target feature $\hat{B}_{r_i}$. Without some from of normalization the losses from different layers are of vastly different scales leading to numerical stability issues during optimization.

Using this framework we: (i) study the effectiveness of bilinear features B_{r_i} extracted from various layers of a network for texture and scene recognition (Sect. 3), (ii) investigate the nature of invariances of these features by evaluating the effect of training with different amounts of data augmentation (Sect. 4), (iii) provide insights into the learned models by inverting them (Sect. 5), and (iv) show results for modifying the content of an image with texture attributes (Sect. 6). We conclude in Sect. 7.

3. Texture recognition

In this section we evaluate the bilinear CNN (B-CNN) representation for texture recognition and scene recognition.

Datasets and evaluation. We experiment on three texture datasets – the *Describable Texture Dataset* (DTD) [3], *Flickr Material Dataset* (FMD) [34], and *KTH-TISP2-b* (KTH-T2b) [1]. DTD consists of 5640 images labeled with 47 describable texture attributes. FMD consists of 10 material categories, each of which contains 100 images. Unlike DTD and FMD where images are collected from the Internet, KTH-T2b contains 4752 images of 11 materials captured under controlled scale, pose, and illumination. The KTH-T2b dataset splits the images into four samples for each category. We follow the standard protocol by training on one sample and test on the remaining three. On DTD and FMD, we randomly divide the dataset into 10 splits and report the mean accuracy across splits. Besides these, we also evaluate our models on *MIT indoor scene* dataset [32].

Indoor scenes are weakly structured and orderless texture representations have been shown to be effective here. The dataset consists of 67 indoor categories and a defined training and test split.

Descriptor details and training protocol. Our features are based on the "verydeep VGG network" [36] consisting of 16 convolutional layers pre-trained on the ImageNet dataset. The FV-CNN builds a Fisher Vector representation by extracting CNN filterbank responses from a particular layer of the CNN using 64 Gaussian mixture components, identical to setup of Cimpoi *et al.* [4]. The B-CNN features are similarly built by taking the location-wise outer product of the filterbank responses and average pooling across all locations (identical to B-CNN [D,D] in Lin *et al.* [25]). Both these features are passed through signed square-root and ℓ_2 normalization which has been shown to improve performance. During training we learn one-vs-all SVMs (trained with SVM hyperparameter $C = 1$) and weights scaled such that the median positive and negative class scores in the training data is $+1$ and -1 respectively. At test time we assign the class with the highest score. Our code in implemented using MatConvNet [38] and VLFEAT [37] libraries.

3.1. Experiments

The following are the main conclusions of the experiments:

1. B-CNN compares favorably to FV-CNN. Tab. 1 shows results using the B-CNN and FV-CNN on various datasets. Across all scales of the input image the performance using B-CNN and FV-CNN is virtually identical. The FV-CNN multi-scale results reported here are comparable ($\pm 1\%$) to the results reported in Cimpoi *et al.* [4] for all datasets except KTH-T2b (-4%). These differences in results are likely due to the choice of the CNN [1] and the range of scales. These results show that the bilinear pooling is *as good as* the Fisher vector pooling for texture recognition. One drawback is that the FV-CNN features with 64 GMM components has half as many dimensions ($64 \times 2 \times 256$) as the bilinear features (256×256). However, it is known that these features are highly redundant and their dimensionality can be reduced by an order of magnitude without loss in performance [11, 25].

2. Multi-scale analysis improves performance. Tab. 1 shows the results by combining features from multiple scales $2^s, s \in \{1.5:-0.5:-3\}$ relative to the 224×224 image. We discard scales for which the image is smaller than the size of the receptive fields of the filters, or larger than 1024^2 pixels for efficiency. Multiple scales consistently lead to an improvement in accuracy.

---

[1] they use the *conv5_4* layer of the 19-layer VGG network.

Dataset	FV-CNN			B-CNN		
	$s=1$	$s=2$	ms	$s=1$	$s=2$	ms
DTD	67.8 ±0.9	70.6 ±0.9	73.6 ±1.0	69.6 ±0.7	71.5 ±0.8	72.9 ±0.8
FMD	75.1 ±2.3	79.0 ±1.4	80.8 ±1.7	77.8 ±1.9	80.7 ±1.5	81.6 ±1.7
KTH-T2b	74.8 ±2.6	75.9 ±2.4	77.9 ±2.0	75.1 ±2.8	76.4 ±3.5	77.9 ±3.1
MIT indoor	70.1	78.2	78.5	72.8	77.6	79.0

Table 1. **Comparison of B-CNN and FV-CNN.** We report mean per-class accuracy on DTD, FMD, KTH-T2b and MIT indoor datasets using FV-CNN and B-CNN representations constructed on top of *relu5_3* layer outputs of the 16-layer VGG network [36]. Results are reported using input images at different scales: $s=1$, $s=2$ and ms are with images resized to 224×224, 448×448 and pooled across multiple scales respectively.

Dataset	*relu2_2*	*relu3_3*	*relu4_3*	*relu5_3*
DTD	42.9	59.0	68.8	69.9
FMD	49.6	62.2	73.4	80.2
KTH-T2b	59.9	71.3	78.8	79.0
MIT indoor	32.2	54.5	71.1	72.8

Table 2. **Layer by layer performance.** The classification accuracy using B-CNN features based on the outputs of different layers on several datasets using input at $s=1$, i.e. 224×224 pixels. The numbers are reported on the first split of all datasets.

3. Higher layers perform better. Tab. 2 shows the performance using various layers of the CNN. The accuracy improves using the higher layers in agreement with [4].

4. Multi-layer features improve performance. By combining features from all the layers we observe a small but significant improvement in accuracy on DTD $69.9\% \rightarrow 70.7\%$ and on MIT indoor from $72.8\% \rightarrow 74.9\%$. This suggests that the features from multiple layers capture complementary information and can be combined to improve performance. This is in agreement with the "hypercolumn" approach of Hariharan *et al*. [16].

5. Fine-tuning leads to a small improvement. On the MIT indoor dataset fine-tuning the network using the B-CNN architecture leads to a small improvement $72.8\% \rightarrow 73.8\%$ using *relu5_3* and $s = 1$. Fine-tuning on texture datasets led to insignificant improvements which might be attributed to their small size. On larger and specialized datasets, such as fine-grained recognition, the effect of fine-tuning can be significant [25].

4. The role of translational invariance

Earlier experiments on B-CNN and FV-CNN were reported using pre-trained networks. Here we experiment with training a B-CNN model *from scratch* on the ImageNet

LSRVC 2012 dataset. We experimenting with the effect of spatial jittering of training data on the classification performance. For these experiments we use the VGG-M [2] architecture which performs better than AlexNet [22] with a moderate decrease in classification speed. For the B-CNN model we replace the last two fully-connected layers with a linear classifier and softmax layer on the outputs of the square-root and ℓ_2 normalized bilinear features of the *relu5* layer outputs. The evaluation speed for B-CNN is 80% of that of the standard CNN, hence the overall training times for both architectures are comparable.

We train various models with different amounts of spatial jittering – "f1" for flip, "f5" for flip + 5 translations and "f25" for flip + 25 translations. In each case the training is done using stochastic sampling where one of the jittered copies is randomly selected for each example. The network parameters are randomly initialized and trained using stochastic gradient descent with momentum for a number of epochs. We start with a high learning rate and reduce it by a factor of 10 when the validation error stops decreasing. We stop training when the validation error stops decreasing.

Fig. 2 shows the "top1" validation errors and compares the B-CNN network to the standard VGG-M model. The validation error is reported on a single center cropped image. Note that we train all networks with neither PCA color jittering nor batch normalization and our baseline results are within 2% of the top1 errors reported in [2]. The VGG-M model achieves 46.4% top1 error with flip augmentation during training. The performance improves significantly to 39.6% with f25 augmentation. As fully connected layers in a standard CNN network encode spatial information, the model loses performance without spatial jittering. For B-CNN network, the model achieves 38.7% top1 error with f1 augmentation, outperforming VGG-M with f25 augmentation. With more augmentations, B-CNN model improves top1 error by 1.6% (38.7% → 37.1%). Going from f5 to f25, B-CNN model improves marginally by < 1%. The results show that B-CNN feature is discriminative and robust to translation. With a small amount of data jittering, B-CNN network achieves fairly good performance, suggesting that explicit translation invariance might be preferable to the implicit invariance obtained by data jittering.

5. Understanding texture representations

In this section we aim to understand B-CNN texture representation by synthesizing *invariant images*, i.e. images that are nearly identical to a given image according to the bilinear features, and *inverse images* for a given category.

Visualizing invariant images for objects. We use *relu1_1, relu2_1, relu3_1, relu4_1, relu5_1* layers for texture representation. Fig. 3 shows several invariant images to the image on the top left, *i.e.* these images are virtually identical as far as the bilinear features for these layers are con-
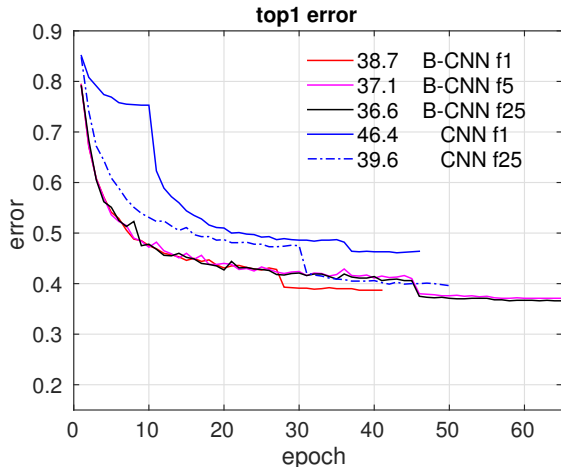
Figure 2. **Effect of spatial jittering on ImageNet LRVC 2012 classification.** The top1 validation error on a single center crop on ImageNet dataset using the VGG-M network and the corresponding B-CNN model. The networks are trained with different levels of data jittering: "f1", "f5", and "f25" indicating flip, flip + 5 translations, and flip + 25 translations respectively.

Figure 3. **Invariant inputs.** These six images are virtually identical when compared using the bilinear features of layers *relu1_1, relu2_1, relu3_1, relu4_1, relu5_1* of the VGG network [36].

cerned. Translational invariance manifests as shuffling of patches but important local structure is preserved within the images. These images were obtained using $\gamma = 1e - 6$ and $\alpha_i = 1 \; \forall i$ in Eqn. 5. We found that as long as some higher and lower layers are used together the synthesized textures look reasonable, similar to the observations of Gatys *et al.*

Role of initialization on texture synthesis. Although the same approach can be used for texture synthesis, it is not practical since it requires several hundreds of CNN evaluations, which takes several minutes on a high-end GPU. In comparison, non-parametric patch-based approaches such as *image quilting* [9] are orders of magnitude faster. Quilting introduces artifacts when adjacent patches do not align with each other. The original paper proposed an approach
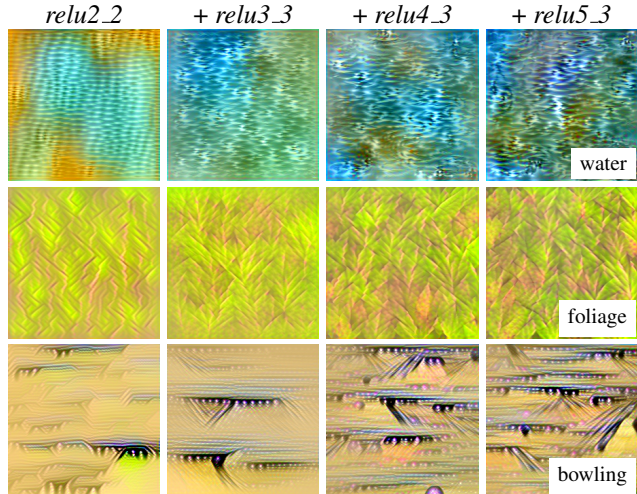

Figure 4. **Effect of layers on inversion.** Pre-images obtained by inverting class labels using different layers. The leftmost column shows inverses using predictions of *relu2_2* only. In the following columns we add layers *relu3_3*, *relu4_3*, and *relu5_3* one by one.

where a one-dimensional cut is found that minimizes artifacts. However, this can fail since local adjustments cannot remove large structural errors in the synthesis. We instead investigate the use of quilting to initialize the gradient-based synthesis approach. Fig. 5 shows the objective through iterations of L-BFGS starting from a random and quilting-based initialization. Quilting starts at a lower objective and reaches the final objective of the random initialization significantly faster. Moreover, the global adjustments of the image through gradient descent remove many artifacts that quilting introduces (digitally zoom in to the *onion image* to see this). Fig. 6 show the results using image quilting as initialization for style transfer [12]. Here two images are given as input, one for content measured as the *conv4_2* layer output, and one for style measured as the bilinear features. Similar to texture synthesis, the quilting-based initialization starts from lower objective value and the optimization converges faster. These experiments suggest that patch-based and parametric approaches for texture synthesis are complementary and can be combined effectively.

Visualizing texture categories. We learn linear classifiers to predict categories using bilinear features from *relu2_2, relu3_3, relu4_3, relu5_3* layers of the CNN on various datasets and visualize images that produce high prediction scores for each class. Fig. 1 shows some example inverse images for various categories for the DTD, FMD and MIT indoor datasets. These images were obtained by setting $\beta = 100$, $\gamma = 1e - 6$, and $\hat{C}$ to various class labels in Eqn. 5. These images reveal how the model represents texture and scene categories. For instance, the *dotted* category of DTD contains images of various colors and dot sizes and the inverse image is composed of multi-scale multi-colored dots. The inverse images of *water* and *wood* from FMD are
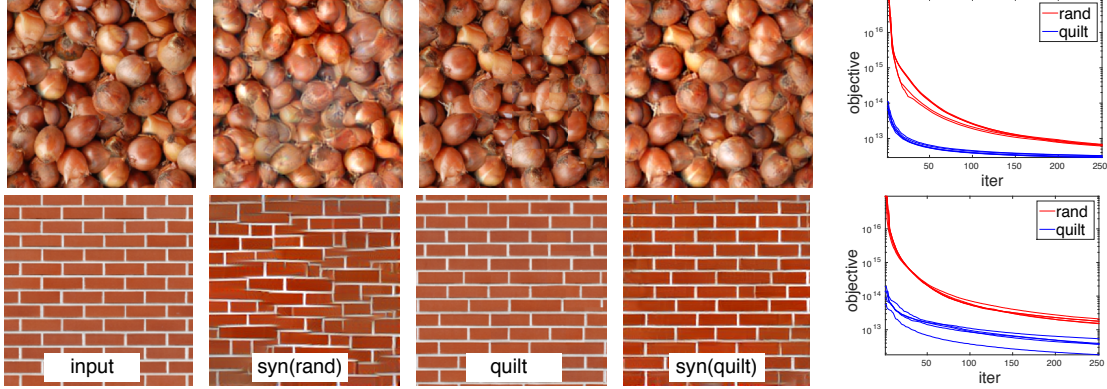
Figure 5. **Effect of initialization on texture synthesis.** Given an input image, the solution reached by the L-BFGS after 250 iterations starting from a random image: *syn(rand)*, and image quilting: *syn(quilt)*. The results using image quilting [9] are shown as *quilt*. On the right is the objective function for the optimization for 5 random initializations. Quilting-based initialization starts at a lower objective value and matches the final objective of the random initialization in far fewer iterations. Moreover, many artifacts of quilting are removed in the final solution (*e.g.*, the top row). *Best viewed with digital zoom.* Images are obtained from http://www.textures.com.
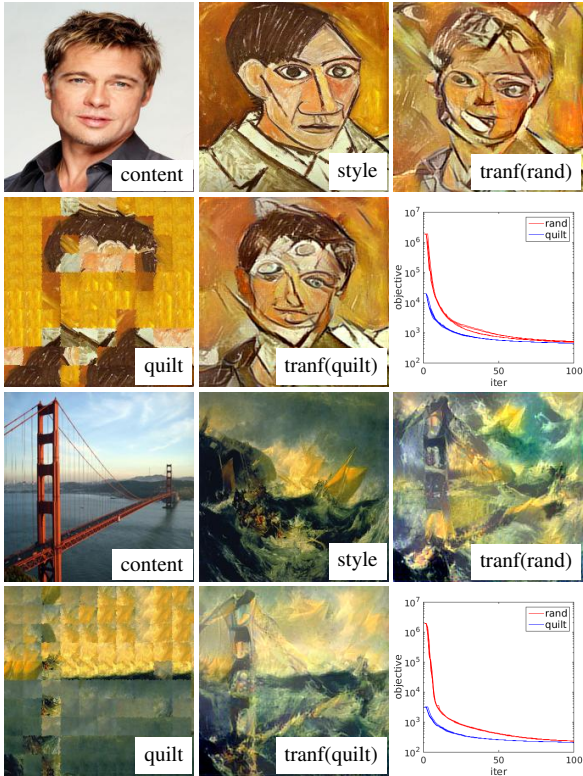

Figure 6. **Effect of initialization on style transfer.** Given a content and a style image, the results of style transfer using L-BFGS after 100 iterations starting from a random image: *tranf(rand)*, and image quilting: *tranf(quilt)*. The results using image quilting [9] are shown as *quilt*. On the bottom right is the objective function for the optimization for 5 random initializations.

highly representative of these categories. Note that these images cannot be obtained by simply averaging instances within a category which is likely to produce a blurry image. The orderless nature of the texture descriptor is essential to produce such sharp images. The inverse scene images from the MIT indoor dataset reveal key properties that the model learns – a *bookstore* is visualized as racks of books while a *laundromat* has laundry machines at various scales and locations. In Fig. 4 we visualize reconstructions by incrementally adding layers in the texture representation. Lower layers preserve color and small-scale structure and combining all the layers leads to better reconstructions. Even though the *relu5_3* layer provides the best recognition accuracy, simply using that layer did not produce good inverse images (not shown). Notably, color information is discarded in the upper layers. Fig. 7 shows visualizations of some other categories across datasets.

6. Manipulating images with texture attributes

Our framework can be used to edit images with texture attributes. For instance, we can make a texture or the content of an image more honeycombed or swirly. Fig. 8 shows some examples where we have modified images with various attributes. The top two rows of images were obtained by setting $\alpha_i = 1 \,\forall i$, $\beta = 1000$ and $\gamma = 1e-6$ and varying $\tilde{C}$ to represent the target class. The bottom row is obtained by setting $\alpha_i = 0 \,\forall i$, and using the *relu4_2* layer for content reconstruction with weight $\lambda = 5e - 8$.

The difference between the two is that in the content reconstruction the overall structure of the image is preserved. The approach is similar to the neural style approach [12], but instead of providing a style image we adjust the image with attributes. This leads to interesting results. For instance, when the face image is adjusted with the interlaced attribute (Fig. 8 bottom row) the result matches the scale and orientation of the underlying image. No single image in the DTD dataset has all these variations but the categorical representation does. The approach can be used to modify an image with other high-level attributes such as artistic styles by learning style classifiers.
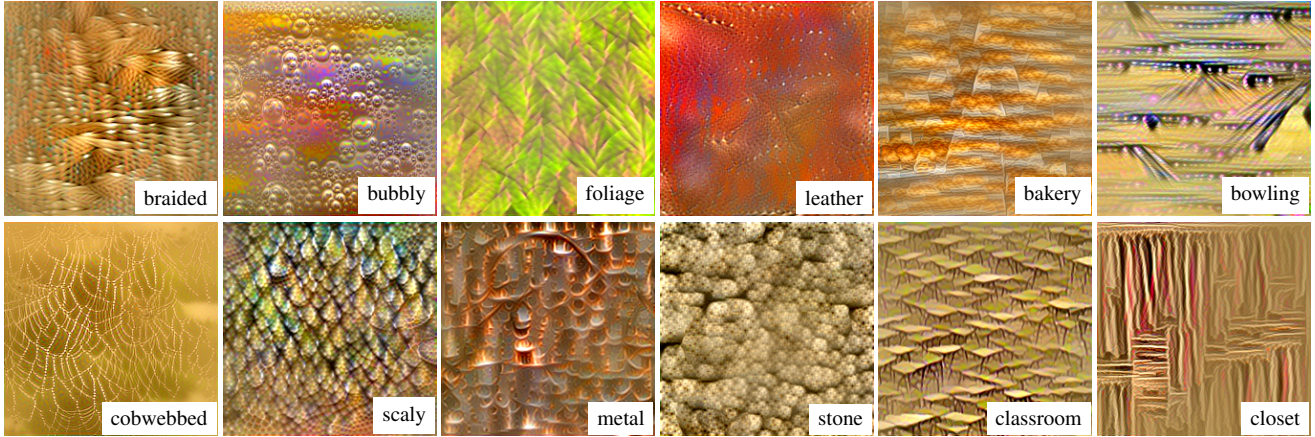
Figure 7. **Examples of texture inverses (Fig. 1 cont.)** Visualizing various categories by inverting the bilinear CNN model [25] trained on DTD [3], FMD [34], and MIT Indoor dataset [32] (two columns each from left to right). *Best viewed in color and with zoom.*
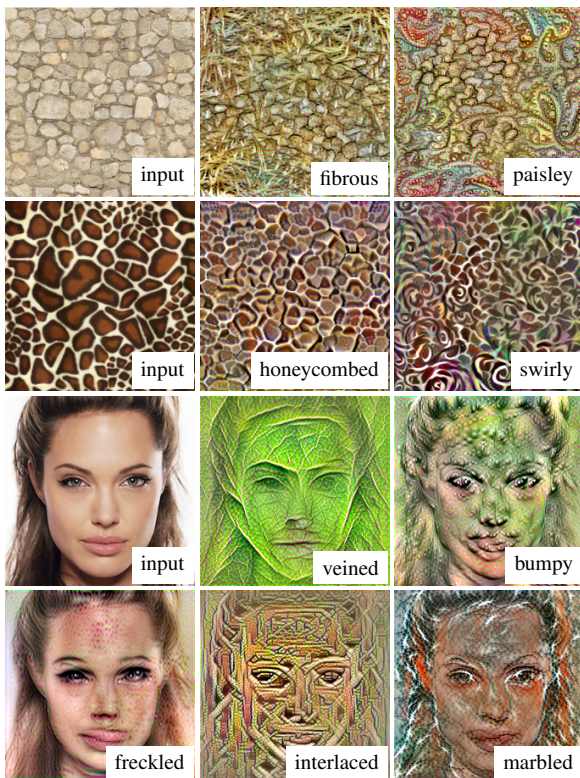

Figure 8. **Manipulating images with attributes.** Given an image we synthesize a new image that matches its texture (*top two rows*) or its content (*bottom two rows*) according to a given attribute (*shown in the image*).

We can also blend texture categories using weights β_j of the targets $\hat{C}_j$. Fig. 9 shows some examples. On the left is the first category, on the right is the second category, and in the middle is where a transition occurs (selected manually).

7. Conclusion

We present a systematic study of recent CNN-based texture representations by investigating their effectiveness on
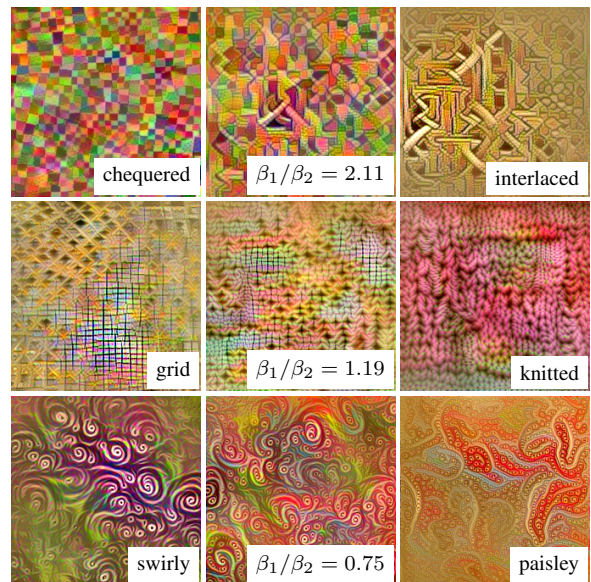

Figure 9. **Hybrid textures** obtained by blending the texture on the left and right according to weights β_1 and β_2.

recognition tasks and studying their invariances by inverting them. The main conclusion is that translational invariance is a useful property not only for texture and scene recognition, but also for general object classification on the ImageNet dataset. The resulting models provide a rich parametric approach for texture synthesis and manipulation of content of images using texture attributes. The key challenge is that the approach is computationally expensive, and we present an initialization scheme based on image quilting that significantly speeds up the convergence and also removes many structural artifacts that quilting introduces. The complementary qualities of patch-based and gradient-based methods may be useful for other applications.

References

[1] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005. 2, 4

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 1, 2, 5

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 1, 2, 4, 8

[4] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *IJCV*, pages 1–30, 2016. 1, 2, 4, 5

[5] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9), 2004. 3

[6] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV Workshop on Stat. Learn. in Comp. Vision*, 2004. 1, 2

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

[8] A. Dosovitskiy and T.Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016. 1, 3

[9] A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001. 2, 3, 6, 7

[10] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *CVPR*, 1999. 2

[11] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. *arXiv preprint arXiv:1511.06062*, 2015. 4

[12] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 3, 6, 7

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In *NIPS*, 2015. 1, 2, 3

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[15] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 2

[16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2, 5

[17] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, 1995. 2

[18] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001. 3

[19] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 1, 2

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia (ACM MM)*, 2014. 1

[21] B. Julesz and J. R. Bergen. Textons, the fundamental elements in preattentive vision and perception of textures. *Bell System Technical Journal*, 62(6, Pt 3):1619–1645, Jul-Aug 1983. 1

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 5

[23] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, 2003. 2

[24] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001. 1, 2

[25] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-grained Visual Recognition. In *ICCV*, 2015. 1, 2, 3, 4, 5, 8

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[27] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 2

[28] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 1, 3, 4

[29] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015. 2

[30] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 1, 2

[31] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1):49–70, 2000. 2

[32] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 1, 2, 4, 8

[33] A. S. Razavin, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop*, 2014. 1

[34] L. Sharan, R. Rosenholtz, and E. H. Adelson. Material perceprion: What can you see in a brief glance? *Journal of Vision*, 9:784(8), 2009. 1, 2, 4, 8

[35] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR workshop*, 2014. 1, 3

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 4, 5, 6

[37] A. Vedaldi and B. Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In *ACM Multimedia (ACM MM)*, 2010. 4

[38] A. Vedaldi and K. Lenc. MatConvNet: convolutional neural networks for matlab. In *ACM Multimedia (ACM MM)*, 2015. 4

[39] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, 2000. 2

[40] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1, 3