Wavelet Convolutional Neural Networks for Texture Classification

Shin Fujieda
The University of Tokyo, Digital Frontier Inc.
sfujieda@graphics.ci.i.u-tokyo.ac.jp

Kohei Takayama
Digital Frontier Inc.
ktakayama@dfx.co.jp

Toshiya Hachisuka
The University of Tokyo
hachisuka@ci.i.u-tokyo.ac.jp

Abstract

Texture classification is an important and challenging problem in many image processing applications. While convolutional neural networks (CNNs) achieved significant successes for image classification, texture classification remains a difficult problem since textures usually do not contain enough information regarding the shape of object. In image processing, texture classification has been traditionally studied well with spectral analyses which exploit repeated structures in many textures. Since CNNs process images as-is in the spatial domain whereas spectral analyses process images in the frequency domain, these models have different characteristics in terms of performance. We propose a novel CNN architecture, wavelet CNNs, which integrates a spectral analysis into CNNs. Our insight is that the pooling layer and the convolution layer can be viewed as a limited form of a spectral analysis. Based on this insight, we generalize both layers to perform a spectral analysis with wavelet transform. Wavelet CNNs allow us to utilize spectral information which is lost in conventional CNNs but useful in texture classification. The experiments demonstrate that our model achieves better accuracy in texture classification than existing models. We also show that our model has significantly fewer parameters than CNNs, making our model easier to train with less memory.

1. Introduction

Texture is a key component used for various applications in computer graphics. While its definition varies slightly, texture is typically a surface image of an object and it does not represent the shape of object. For example, a photograph of an entire human face is usually not considered to be a texture, but a close-up of a human skin is. In rendering, artists use textures to add surface details to objects without having to increase geometric complexity. For image processing, texture is used to represent types of surfaces that are independent of shape. Texture can be thought as a basic element that captures the appearance of surfaces of objects.

Accurate classification of textures is also fundamental in many important applications such as inspection and segmentation for image processing and generation of texture database for rendering. At the same time, texture classification is a challenging problem since textures often vary a lot within the same class, due to changes in viewpoints, scales, lighting configurations, etc. In addition, textures usually do not contain enough information regarding the shape of objects which are informative to distinguish different objects in image classification tasks. Due to such difficulties, even the latest approaches based on convolutional neural networks achieved a limited success, when compared to other tasks such as image classification.

We propose a unification of two major classification approaches, convolutional neural networks and spectral analyses, to approach the difficulty of texture classification. Convolutional neural networks (CNNs) process an input texture as-is and collect statistics in the spatial domain. Spectral analysis transforms an input texture into a spectral domain and uses frequency statistics. CNNs are usually good at capturing spatial features, while a spectral analysis is good at capturing scale invariant features. We aim to consider both the spatial and spectral information so that it captures both types of features well under a single model. The key idea is that the pooling layer and the convolution layer in CNNs can be thought as a limited form of a spectral analysis. Based on this idea, we generalize both layers to perform a spectral analysis using multiresolution analysis by wavelet transform. We thus named our model as *wavelet convolutional neural networks* (wavelet CNNs). The overview of wavelet CNNs is shown in Figure 1. We demonstrate that wavelet CNNs achieve better or competitive classification accuracy while having a significantly smaller number of trainable parameters than conventional CNNs. Our model is thus easier to

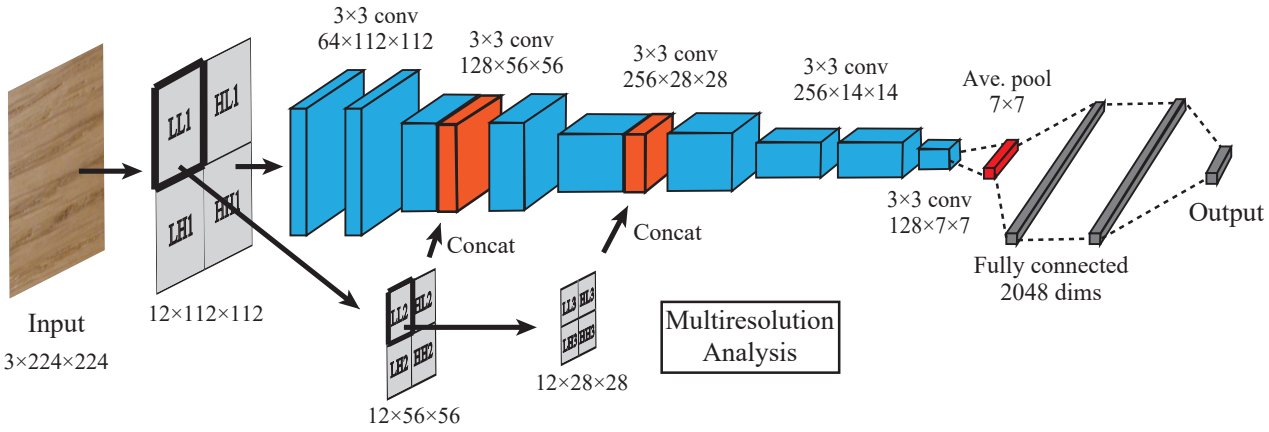

Figure 1. Overview of our model with 3-level decomposition of the input image. Our model processes the input image through convolution layers with 3×3 kernels and 1×1 padding. 3×3 convolutional kernels with the stride of 2 and 1×1 padding are used to reduce the size of feature maps. Additionally, the input image is decomposed through multiresolution analysis and the decomposed images are concatenated. The output of convolution layers is vectorized by an average pooling layer followed by three fully connected layers. The size of the output is equal to the number of classes included in the input dataset.

train and consumes less memory than CNNs. To summarize, our contributions are:

- Combination of CNNs and spectral analysis within one model.
- Generalization of pooling and convolution as a spectral analysis.
- Accurate and efficient texture classification using our model.

Several numerical experiments in the results section validate that our model successfully classified failure cases of existing models.

2. Related Work

Conventional Texture Descriptors There are a variety of approaches for extracting texture descriptors based on domain-specific knowledge. Structural approaches [36, 12] represent texture features based on spatial arrangements of selected pixels. Statistical approaches [34] consider a set of statistics of pixel intensities such as mean and co-occurrence matrices [15, 35] as features. These approaches can yield certain features of textures in a compact manner, but only under the assumptions in each approach.

More general approaches such as Markov random fields [28], fractal dimensions [6], and Wold model [26], described texture images as a probability model or a linear combination of a set of basis functions. The coefficients of these models are texture features in these approaches. While these approaches are quite general, it is still difficult choose the most suitable model for given textures.

Convolutional Neural Networks CNNs essentially replaced conventional descriptors by achieving significant performance [25] in various tasks without relying on much of domain-specific knowledge. For texture classification, however, directly applying CNNs is known to achieve only moderate accuracy [14]. CNNs alone thus are not very suitable for texture classification, despite its successes in other problems.

Cimpoi et al. [8] demonstrated a CNN in combination with Fisher Vectors (FV-CNN) can achieve much better accuracy in texture classification. Their model uses a pre-trained CNN to extract texture features and this CNN part is not trained with existing texture datasets. Inherited from conventional CNNs, this model has a large number of parameters that makes it difficult to train in general. Our model uses a fewer parameters and achieves competitive results to FV-CNN by fusing a CNN and a spectral analysis into one model.

Andrearczyk et al. [2] proposed texture CNN (T-CNN) which is a CNN specialized for texture classification. T-CNN uses a novel energy layer in which each feature map is simply pooled by calculating the average of its activated output. This results in a single value for each feature map, similar to an energy response to a filter bank. This approach does not improve classification accuracy, but to its simple architecture reduces the number of parameters. While our model is inspired by the problem of texture classification, it is not specialized for texture classification. As discussed in Section 5, we confirmed that our model achieves competitive performance to AlexNet [25] for image classification.

Spectral Approaches Spectral approaches transform textures into the frequency domain using a set of spatial filters. The statistics of the spectral information at different scales and orientations define texture features. This approach has been well studied in image processing and achieved practical results [37, 3, 22].

Feature extraction in the frequency domain has two advantages. First, a spatial filter can be easily made selective by enhancing only certain frequencies while suppressing others. This explicit selection of certain frequencies is difficult to control in CNNs. Additionally, the periodical structure of a texture can be represented as a certain spatial frequency in the spectral domain.

Figure 1 shows that the structure of our model is similar to that of CNNs using skip-connections [27, 30]. While deep neural networks including CNNs with skip-connections are known to be universal approximators [18], it is not clear whether CNNs can learn to perform spectral analyses in practice with available datasets. We thus propose to directly integrate spectral approaches into CNNs, particularly based on multiresolution analysis using wavelet transform [37]. Our experiments support this observation that a CNN with more parameters cannot be trained to become equivalent to our model with available datasets in practice. The key difference is that certain layers of wavelet CNNs have no trainable parameters. Instead, those layers perform multiresolution analysis using fixed parameters defined by wavelet transform.

3. Wavelet Convolutional Neural Networks

Overview We propose to formulate convolution and pooling in CNNs as filtering and downsampling. This formulation allows us to connect convolution and pooling with multiresolution analysis. In the following explanations, we use a single-channel 1D data for the sake of brevity. Applications to 2D images with multiple channels are trivially possible as was done by CNNs.

3.1. Convolutional Neural Networks

A convolutional neural network [25] is a variant of the neural network which uses a sparsely connected deep network. In the regular neural network model, every input is connected to every unit in the next layer. In addition to the use of an activation function and a fully connected layer, CNNs introduce convolution/pooling layers that connect only to local neighborhoods (called a local receptive field) around each input. Figure 2 illustrates the configuration we explain in the following.

Convolution Layers: Given an input vector with n components $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{R}^n$, a convolution layer outputs a vector of the same number of components

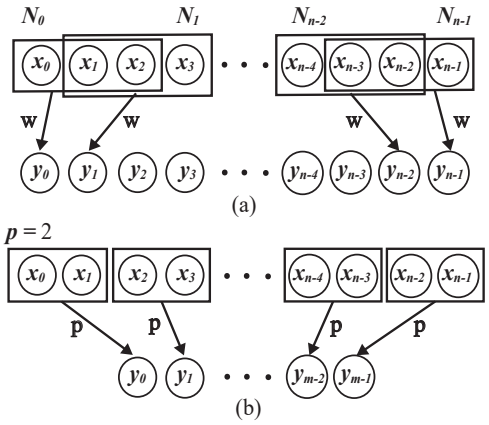

Figure 2. Concepts of convolution and average pooling layers. (a) Convolution layers compute a weighted sum of neighbors in each local receptive field. (b) Pooling layers take an average and perform downsampling.

$\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{R}^n$:

$$y_i = \sum_{j \in N_i} w_j x_j, \tag{1}$$

where N_i is a set of indices in the local receptive field at x_i and w_j is a weight. Following the notational convention in CNNs, we consider that w_j includes the bias by having a constant input of 1. The equation thus says that each output y_i is a weighted sum of neighbors $\sum_{j \in N_i} w_j x_j$ plus constant.

Each layer defines the weights w_j as constants over i. By sharing parameters, CNNs reduce the number of parameters and achieve translation invariance in the image space. The definition of y_i in Equation 1 is equivalent to convolution of x_i via a filtering kernel w_j, thus this layer is called a convolution layer. We can thus rewrite y_i in Equation 1 using the convolution operator $*$ as

$$\mathbf{y} = \mathbf{x} * \mathbf{w}, \tag{2}$$

where $\mathbf{w} = (w_0, w_1, \ldots, w_{m-1}) \in \mathbb{R}^m$. Convolution layers in CNNs typically use different sets of weights for the same input and output the results as a concatenated vector. This common practice just applies Equation 1 repeatedly for each set of weights.

Pooling Layers: Pooling layers are typically used immediately after convolution layers to simplify the information. While max pooling is used in many applications of CNNs, Gatys et al. [10] showed that average pooling is more suitable for extracting texture features. We thus focus on average pooling, which in fact allows us to see the connection with multiresolution analysis. Given an input $\mathbf{x} \in \mathbb{R}^n$, average

pooling outputs a vector of a fewer components $\mathbf{y} \in \mathbb{R}^m$ as

$$y_j = \frac{1}{p} \sum_{k=0}^{p-1} x_{pj+k}, \tag{3}$$

where p defines the support of pooling and $m = \frac{n}{p}$. For example, $p = 2$ means that we reduce the number of outputs to a half of the inputs by taking pair-wise averages. Using the standard downsampling operator $\downarrow$, we can rewrite Equation 3 as

$$\mathbf{y} = (\mathbf{x} * \mathbf{p}) \downarrow p, \tag{4}$$

where $\mathbf{p} = (1/p, \ldots, 1/p) \in \mathbb{R}^p$ represents the averaging filter. Average pooling mathematically involves convolution via $\mathbf{p}$ followed by downsampling with the stride of p.

3.2. Generalized Convolution and Pooling

Equation 2 and Equation 4 can be combined into a generalized form of convolution and downsampling as

$$\mathbf{y} = (\mathbf{x} * \mathbf{k}) \downarrow p. \tag{5}$$

The generalized weight $\mathbf{k}$ is defined as

- $\mathbf{k} = \mathbf{w}$ with $p = 1$ (convolution in Equation 2)
- $\mathbf{k} = \mathbf{p}$ with $p > 1$ (pooling in Equation 4)
- $\mathbf{k} = \mathbf{w} * \mathbf{p}$ with $p > 1$ (convolution followed by pooling).

Our insight is that Equation 5 is equivalent to a part of a single level application of multiresolution analysis. Suppose that we use a pair of low-pass $\mathbf{k}_l$ and high-pass $\mathbf{k}_h$ filters to decompose data into the low-frequency part $\mathbf{x}_{\text{low}}$ and the high frequency part $\mathbf{x}_{\text{high}}$ with $p = 2$:

$$\begin{aligned} \mathbf{x}_{\text{low}} &= (\mathbf{x} * \mathbf{k}_l) \downarrow 2 \\ \mathbf{x}_{\text{high}} &= (\mathbf{x} * \mathbf{k}_h) \downarrow 2. \end{aligned} \tag{6}$$

Our key insight is that multiresolution analysis [9] further decomposes the low frequency part $\mathbf{x}_{\text{low}}$ into its low frequency part and high frequency part by repeatedly applying the same form of Equation 6. By defining $\mathbf{x}_{\text{low},0} = \mathbf{x}$, multiresolution analysis can be written as

$$\begin{aligned} \mathbf{x}_{\text{low},l+1} &= (\mathbf{x}_{\text{low},l} * \mathbf{k}_l) \downarrow 2 \\ \mathbf{x}_{\text{high},l+1} &= (\mathbf{x}_{\text{low},l} * \mathbf{k}_h) \downarrow 2. \end{aligned} \tag{7}$$

The number of applications l is called a level in multiresolution analysis. Multiresolution analysis is thus equal to repeated applications of generalized convolution and pooling layers on low-frequency parts with a specific pair of convolution kernels.

Figure 3 illustrates how CNNs and our wavelet CNNs differ under this formulation. Conventional CNNs can be seen as using only the low frequency parts and discard all the high frequency parts. Our model instead uses both the low frequency parts *and* the high frequency parts within CNNs, so that we do not lose any information of the input $\mathbf{x}$ by the definition of multiresolution analysis. While this idea might look simple after the fact, our model is powerful enough to outperform the existing more complex models as we will show in the results.

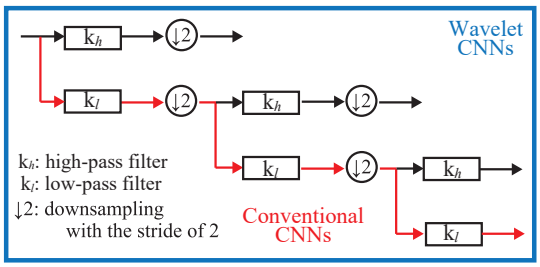

Figure 3. Relationship between conventional CNNs and wavelet CNNs in terms of multiresolution analysis. Conventional CNNs apply convolution and pooling repeatedly to the input, which is essentially equivalent to use only the low frequency components of multiresolution analysis. Our wavelet CNNs instead consider all the components including high frequency components.

Note that we cannot use an arbitrary pair of filters ($\mathbf{k}_l$ and $\mathbf{k}_h$) to perform multiresolution analysis. To avoid any loss of frequency information of the input $\mathbf{x}$, a pair should satisfy the quadrature mirror filter relationship. For wavelet transform, $\mathbf{k}_h$ is known as the wavelet function and $\mathbf{k}_l$ is known as the scaling function. We used Haar wavelets [13] for our experiments, but our model is not restricted to Haar. This constraint also suggests why it is difficult to train conventional CNNs to perform the same computation as wavelet CNNs do: weights in CNNs are ignorant of this important constraint to satisfy and just try to learn it from datasets.

Rippel et al. [29] proposed a related approach of replacing convolution and pooling by discrete Fourier transform and truncation of the coefficients. This approach, called *spectral pooling*, is equivalent to using only the low frequency part in our model, thus it is not essentially different from conventional CNNs. Our model is also different from merely applying multiresolution analysis on input data and using CNNs afterward, since multiresolution analysis is built inside the network and the input first go through CNN layers both before and after multiresolution analysis.

3.3. Implementation

Network Structure Figure 1 illustrates our network structure. We designed our network structure after a VGG-19 network [33] since it has been successfully used for extracting features of textures for different purposes [10, 1]. We use 3×3 convolutional kernels exclusively and 1×1 padding to ensure the output is the same size as the input.

Instead of using the pooling layers to reduce the size of the feature maps, we exploit convolution layers with the increased stride. If 1×1 padding is added to the layer with a stride of two, the output becomes half the size of the input layer. This approach can be used to replace max pooling without loss in accuracy [20]. In addition, since both the VGG-like architecture and image decomposition in multiresolution analysis have the same characteristic that the size of images is reduced to a half successively, we combine each level of decomposed images with feature maps of the specific layer that are the same size as those images.

For texture classification, inserting an energy layer directly before fully connected layers improves the performance of a network while keeping the number of parameters small [2]. We used this approach and a complete network of wavelet CNNs we tested consists of nine convolution layers, the same number of wavelet layers as decomposition levels of multiresolution analysis and an energy layer followed by three fully connected layers. We implemented this network using Caffe [21]. The codes for our model will be available on our website.

Learning Wavelet CNNs exploit an energy layer with the same size as the input of the layer, so the size of input images is required to be the fixed size. We thus train our proposed model exclusively with images of the size 224×224. These images are achieved by first scaling the training images to 256×256 pixels and then conducting random crops to 224×224 pixels and flipping. This random variation helps the model to prevent overfitting. For further robustness, we use batch normalization [19] throughout our network before activation layers during training. For the optimizer, we exploit the Adam optimizer [23] instead of SGD. We use the Rectified Linear Unit (ReLU) [11] as the activation function in all the experiments.

4. Results

Datasets For our experiments, we used two publicly available texture datasets: *kth-tips2-b* [16] and *DTD* [7]. The *kth-tips2-b* dataset contains 11 classes of 432 texture images. Each class consists of four samples and each sample has 108 images. Each sample is used for training once while the remaining three samples are used for testing. The results for kth-tips2-b are shown as the mean and the standard deviation over the four splits. The *DTD* dataset contains 47 classes of 120 images "in the wild" which means that images are collected in uncontrolled conditions. This dataset includes 10 available annotated splits with 40 training images, 40 validation images, and 40 testing images for each class. The results for DTD are averaged over the 10 splits. We processed the images in each dataset by global contrast normalization.We calculated the accuracy as percentage of images that are correctly labeled which is a common metric in texture classification.

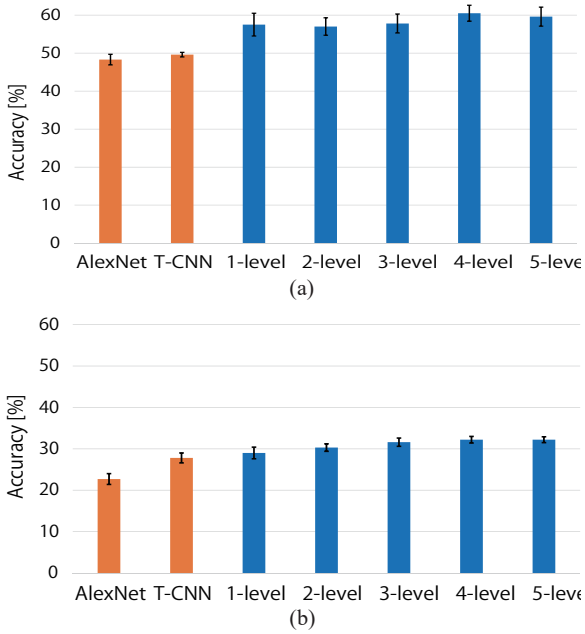

Figure 4. Classification results of (a) kth-tips2-b and (b) DTD for networks trained from scratch. We compared our models (blue) with AlexNet and T-CNN.

Training from scratch We compared our model with AlexNet and T-CNN using texture datasets to train each model from scratch. For initialization of the parameters, we used a robust method that specifically accounts for ReLU [17]. While the structure of our models is designed after VGG networks, we found that VGG networks tend to perform poorly due to over-fitting with a large number of trainable parameters, if trained from scratch. We thus used AlexNet as an example of conventional CNNs instead of VGG networks for this experiment. Figure 4 and Table 1 show the results of training our model with different levels of multiresolution analysis. For both datasets, our models perform better than AlexNet and T-CNN.

Comparing between different levels within wavelet CNNs, we found that the network with 4-level decomposition performed the best. In our experiments, the model with 5-level decomposition achieved almost the same accuracy as 4-level, but with more trainable parameters. Similar to the number of layers in CNNs, the decomposition level of wavelet CNNs is another hyper-parameter which can be tuned for different problems.

Training with fine-tuning Figure 5 and Table 2 show the classification rates using the networks pre-trained with the

	AlexNet	T-CNN	1-level	2-level	3-level	4-level	5-level
kth-tips2-b	$48.3_{\pm 1.4}$	$49.6_{\pm 0.6}$	$57.5_{\pm 3.0}$	$57.0_{\pm 2.3}$	$57.8_{\pm 2.5}$	$\mathbf{60.5_{\pm 2.1}}$	$59.6_{\pm 2.5}$
DTD	$22.7_{\pm 1.3}$	$27.8_{\pm 1.2}$	$29.0_{\pm 1.4}$	$30.3_{\pm 0.9}$	$31.6_{\pm 1.0}$	$\mathbf{32.2_{\pm 0.8}}$	$\mathbf{32.2_{\pm 0.7}}$

Table 1. Classification results for networks trained from scratch indicated as accuracy (%).

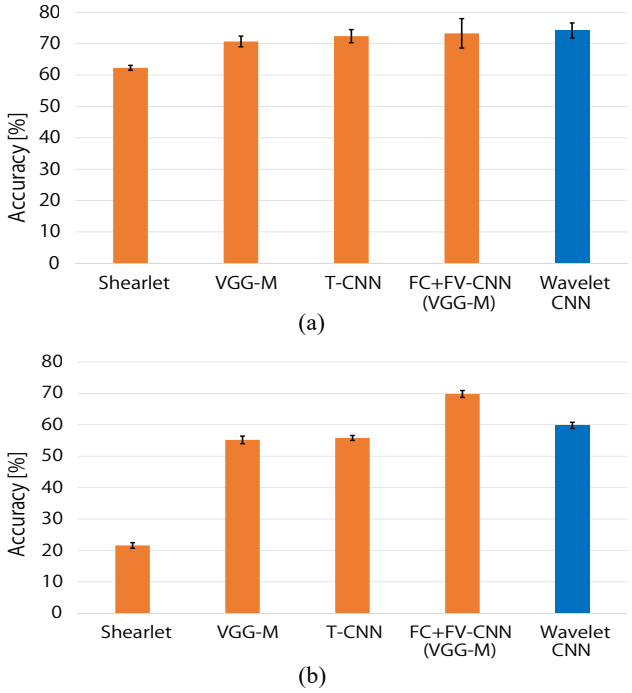

Figure 5. Classification results of (a) kth-tips2-b and (b) DTD for networks pre-trained with ImageNet 2012 dataset. We compared our model (wavelet CNN with 4-level decomposition) with shearlet transform, VGG-M, T-CNN, and FC+FV-CNN.

	Shearlet	VGG-M	T-CNN	FC+ FV-CNN (VGG-M)	Wavelet CNN
kth-tips2-b	$62.3_{\pm 0.8}$	$70.7_{\pm 1.7}$	$72.4_{\pm 2.1}$	$73.9_{\pm 4.9}$	$\mathbf{74.2_{\pm 1.2}}$
DTD	$21.6_{\pm 0.9}$	$55.2_{\pm 1.2}$	$55.8_{\pm 0.8}$	$\mathbf{69.8_{\pm 1.1}}$	$59.8_{\pm 0.9}$

Table 2. Classification results for networks pre-trained with ImageNet indicated as accuracy (%).

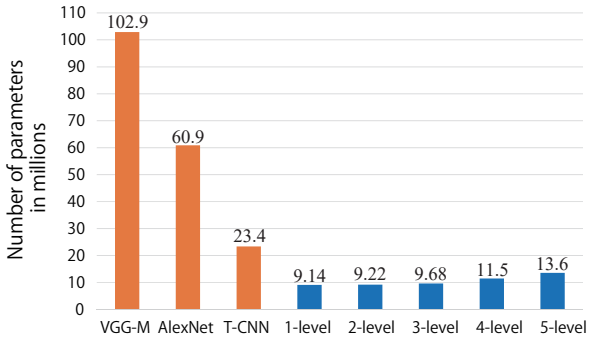

Figure 6. The number of trainable parameters in millions. Our model, even with 5-level of multiresolution analysis, has a fewer parameters than any other competing models we tested.

ImageNet 2012 dataset [31]. Since the model with 4-level decomposition achieved the best accuracy in the previous experiment, we used this network in this experiment as well. We compared our model with a spectral approach using shearlet transform [24], a VGG network [5], T-CNN [2], and FV-CNN [8] with a fully connected layer (FC).

Our model achieved the best performance for the *kth-tips2-b* dataset, while it is outperformed only by FV-CNN for the *DTD* dataset. We analyzed the results and found that some classes of the *DTD* dataset contain non-texture images that clearly show the shape of an object. Since FV-CNN has a significantly larger number of trainable parameters than our models, we speculate that FV-CNN is just trained to account of for non-texture images as special cases. We put FC-CNN as it is the state-of-the-art, a comparison with FV-CNN only on accuracy is not necessarily very fair because of this sheer difference in model complexity.

Number of parameters To assess the complexity of each model, we compared the number of trainable parameters such as weights and biases for classification to 1000 classes (Figure 6). Conventional CNNs such as VGG-M (which is used also in FV-CNN) and AlexNet have a large number of parameters while their depth is a little shallower than our proposed model. Even compared to T-CNN, which aims at reducing the model complexity, the number of parameters in our model with 4-level decomposition is less than the half. We also remind that our model achieved higher accuracy than T-CNN does.

This result confirms that our model achieves better results with a significantly reduced number of parameters than existing models. The memory consumption of each Caffemodel is: 392 MB (VGG-M), 232 MB (AlexNet), 89.1 MB (T-CNN), and 43.8 MB (Ours). Our network is thus suitable to run on a system with a limited amount of memory. The small number of parameters also generally suppress over-fitting of the model for small datasets.

Visual comparisons of classified images Figure 7 shows some extracted images for several classes in our experiments. The images in top two rows of Figure 7 are from *kth-tips2-b* dataset, while the images in the bottom three rows of Figure 7 are from *DTD* dataset. A red square indicates that the texture is inappropriately classified to the class. We can visually see that a spectral approach is insensitive to the scale variation

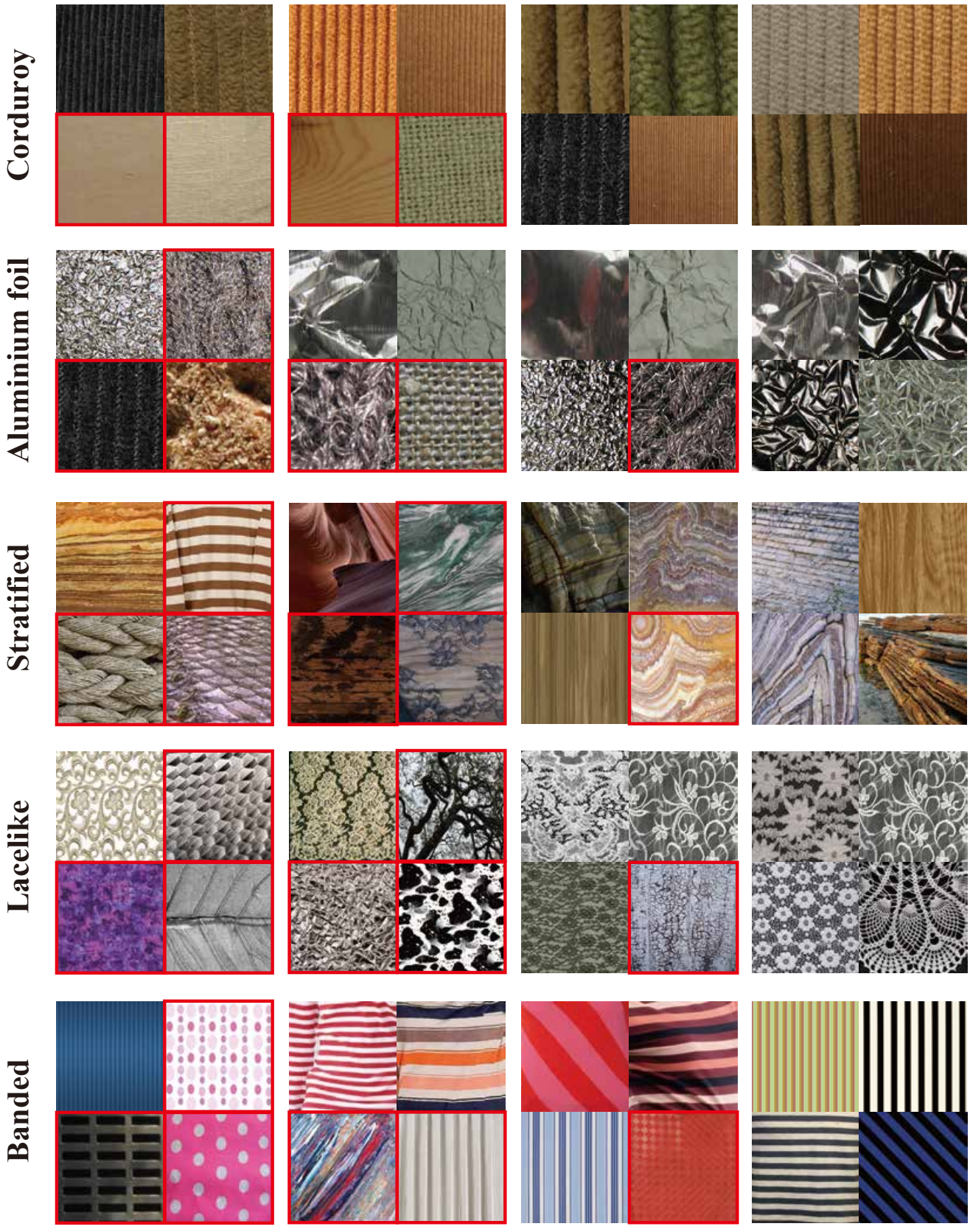

Figure 7. Some results classified by (a) shearlet transform, (b) VGG-M, (c) our model and (d) references. The images on top two rows are extracted from *kth-tips2-b* and the rests are extracted from *DTD*. The images in red squares are wrongly classified images.

and extract detailed features, whereas a spatial approach is insensitive to distortion. For example, in *Aluminium foil*, the image of wrinkled aluminium foil is properly classified with a shearlet transform, but not with VGG-M. In *Banded*, VGG-M classifies distorted lines into the correct class. Since our model is the combination of both approaches, it can assign texture images to the correct label in every variation above.

5. Discussion

Application to image classification Since we do not assume anything regarding the input, our model is not necessarily restricted to texture classification. To confirm the generality, we trained a wavelet CNN with four-level decomposition and AlexNet with the ImageNet 2012 dataset from scratch to perform image classification. Our model obtained the accuracy of 59.8% whereas AlexNet resulted in 57.1%. We should remind that the number of parameters of our model is about five times smaller than that of AlexNet (Figure 6). Our model is thus suitable also for image classification with smaller memory footprint. Other applications such as image recognition and object detection with our model should be similarly possible.

L_p pooling An interesting generalization of max and average pooling is L_p pooling [4, 32]. The idea of L_p pooling is that max pooling can be thought as computing L_∞ norm, while average pooling can be considered as computing L_1 norm. In this case, Equation 4 cannot be written as linear convolution anymore due to non-linear transformation in norm calculation. Our overall formulation, however, is not necessarily limited to multiresolution analysis either and we can simply replace downsampling part by corresponding norm computation to support L_p pooling. This modification however will not retain all the frequency information of the input as it is no longer multiresolution analysis. We focused on average pooling as it has a clear connection to multiresolution analysis.

Limitations We designed wavelet CNNs to put each high frequency part between layers of the CNN. Since our network has four layers to reduce the size of feature maps, the maximum decomposition level is restricted to five. This design is likely to be less ideal since we cannot tweak the decomposition level independently from the depth (thereby the number of trainable parameters) of the network. A different network design might make this separation of hyper-parameters possible.

While wavelet CNNs achieved the best accuracy for training from scratch, its performance with fine-tuning with the ImageNet 2012 dataset is only comparable to FV-CNN, albeit with a significantly smaller number of parameters. We speculated that it is partially because a more complex model of FV-CNN, but another possibility is that pre-training with the ImageNet 2012 dataset is simply not appropriate for texture classification. An exact reasoning of failure cases, however, is generally difficult for any neural network models, and our model is not an exception. We however note that we could not find a publicly available texture dataset at the same scale as the ImageNet 2012 dataset.

We should also point that, while our model improves accuracy a lot when we used textures as only training datasets, the accuracy is still around 60% for *kth-tips2-b* and 32% for *DTD*. This level of accuracy might not be enough yet for certain applications and there is still room for improvement when compared to image classification.

6. Conclusion

We presented a novel CNN architecture which incorporates a spectral analysis into CNNs. We showed how to reformulate convolution and pooling layers in CNNs into a generalized form of filtering and downsampling. This reformulation shows how conventional CNNs perform a limited version of multiresolution analysis, which then allows us to integrate multiresolution analysis into CNNs as a single model called wavelet CNNs. We demonstrated that our model achieves better accuracy for texture classification with smaller number of trainable parameters than existing models. In particular, our model outperformed all the existing models with significantly more trainable parameters when we trained each model from scratch. A wavelet CNN is a general learning model and applications to other problems are interesting future works.

References

[1] M. Aittala, T. Aila, and J. Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Trans. Graph.*, 35(4):65:1–65:13, July 2016. 4

[2] V. Andrearczyk and P. F. Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters*, 84:63 – 69, 2016. 2, 5, 6

[3] S. Arivazhagan, T. G. Subash Kumar, and L. Ganesan. Texture classification using curvelet transform. *International Journal of Wavelets, Multiresolution and Information Processing*, 05(03):451–464, 2007. 3

[4] Y. Boureau, J. Ponce, and Y. Lecun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118. Omnipress, 2010. 8

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014. 6

[6] B. B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):72–77, Jan 1995. 2

[7] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[8] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2, 6

[9] J. L. Crowley. A representation for visual information. Technical report, 1981. 4

[10] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, pages 262–270. Curran Associates, Inc., 2015. 3, 4

[11] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323, 2011. 5

[12] R. K. Goyal, W. L. Goh, D. P. Mital, and K. L. Chan. Scale and rotation invariant texture analysis based on structural property. In *Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on*, volume 2, pages 1290–1294, Nov 1995. 2

[13] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910. 4

[14] O. L. S. Hafemann, L. G. and P. R. Cavalin. Forest species recognition using deep convolutional neural networks. In *22nd International Conference on Pattern Recognition (ICPR)*, pages 1103–1107, 2014. 2

[15] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973. 2

[16] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh. *On the Significance of Real-World Conditions for Material Classification*, volume 4, pages 253–266. 2004. 5

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Dec 2015. 5

[18] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 3

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 5

[20] T. B. J. T. Springenberg, A. Dosovitskiy and M. Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations Workshop Track*, 2015. 5

[21] 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 *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 5

[22] M. Kanchana and P. Varalakshmi. Texture classification using discrete shearlet transform. *International Journal of Scientific Research*, 5, 2013. 3

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[24] K. G. Krishnan, P. T. Vanathi, and R. Abinaya. Performance analysis of texture classification techniques using shearlet transform. In *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1408–1412, March 2016. 6

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114. 2012. 2, 3

[26] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):722–733, Jul 1996. 2

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 3

[28] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, Aug 1996. 2

[29] O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2449–2457, 2015. 4

[30] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351, pages 234–241, 2015. 3

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[32] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 3288–3291, Nov 2012. 8

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4

[34] G. N. Srinivasan and G. Shobha. Statistical texture analysis. *Proceedings of World Academy of Science, Engneering and Technology*, 36, 2008. 2

[35] J. Y. Tou, Y. H. Tay, and P. Y. Lau. One-dimensional grey-level co-occurrence matrices for texture classification. In *International Symposium on Information Technology*, volume 3, pages 1–6, Aug 2008. 2

[36] M. Tüceryan and A. K. Jain. Texture segmentation using voronoi polygons. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(2):211–216, Feb 1990. 2

[37] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, Nov 1995. 3