2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Deep Feature Extraction in the DCT domain

Arthita Ghosh
Center for Automation Research
University of Maryland
College Park, Maryland 20742
Email: arthita@umd.edu

Rama Chellappa
Center for Automation Research
University of Maryland
College Park, Maryland 20742
Email: Rama@umiacs.umd.edu

***Abstract*—We explore the effectiveness of deep features extracted by Convolutional Neural Networks(CNNs) in the Discrete Cosine Transform(DCT) domain for various image classification tasks such as pedestrian and face detection, material identification and object recognition. We perform the DCT operation on the feature maps generated by convolutional layers in CNNs. We compare the performance of the same network on the same datasets, with the same hyper-parameters with or without the DCT step. Our results indicate that a DCT operation incorporated into the network after convolution+thresholding and before pooling can have certain advantages such as convergence over fewer training epochs and sparser weight matrices that are more conducive to pruning and hashing techniques.**

I. Introduction

Convolutional neural networks are recently being used for a wide variety of applications. They have shown remarkably good performance on various computer vision tasks such as image classification, object detection, face recognition and digits/character classification. However, most of these CNN-based models deal with millions of parameters depending on the depth of the network. Deeper neural networks, with a large number of layers and more nodes per layer, while providing good performance for image classification tasks, often require a large number of training epochs to converge and more storage space. Some recently proposed methods aim to optimize the convolutional layers which are responsible for the bulk of computational requirements in training CNNs. For example, a method to modify pretrained CNNs to obtain speed up on character recognition experiments is proposed in [1] by approximating the CNN filter banks using a low-rank basis of filters. A strategy for reduction of training and inference time is proposed in [2] by computing convolutions as point wise products in the Fourier domain and reusing the transformed feature maps.

In another recent work, there has been some focus on image classification tasks using fast learning, shallow convolutional networks[20]. They report state-of-the-art performance(or near state-of-the-art) on MNIST, SVHN and NORB-small datasets, for digit recognition or 3D shape recognition respectively. For applications where online learning or periodic updating of the model weights is required when fresh training data becomes available, shallow models that learn faster and need fewer training epochs are desirable.

Several recent papers also aim at compressing CNNs using various techniques to make them more space and energy efficient. For example, some of these works involve pruning the network by removing redundant connections and learned quantization of the weights so that multiple connections share the same weight. Trained quantization is performed as a separate step in the pipeline proposed by [4], in which clustering of weights and codebook learning are done separately after network training is completed. Although many such strategies have been proposed to compress CNNs, mostly as a separate stage after the training process, an elegant way to do this would be to integrate compression techniques within the training phase of the network so that the resulting weight matrices learned by training the network are automatically more concentrated with fewer higher value weights. Such actions can facilitate the process of pruning weights and compressing the network structure into smaller files that are easier to access.

In [3] the authors demonstrate, that using a two layer convolution network, where the layers were replaced by wavelet filters, yielded results for the Caltech dataset that are comparable to pretrained CNNs of the same depth. They use the Scattering Transform, which is composed of cascaded wavelet convolutions and modulus non-linearities, to replace the learning and pooling steps in the initial layers of the network. Using this technique in the first two layers they capture translation, rotation and scaling variability. While wavelets work in the space-frequency domain, the DCT is another analogous image transform technique that operates in the frequency domain. When it comes to energy compaction techniques, the DCT is a classic and time-tested method which has been in wide use since the introduction of the JPEG standard in 1992 [21]. DCT operation on images help in compaction of information into just a few coefficients in the frequency domain and has been shown to retain essential perceptual information within those few coefficients. In addition, DCT can be computed very efficiently.

In this paper, we study the effect of deep feature extraction using CNNs in the DCT domain by transforming the feature maps generated by the network. We test our model on different types of datasets and demonstrate it's effectiveness in solving different computer vision tasks by contrasting the performance with CNNs without the DCT operation where the feature extraction takes place entirely in the visual domain. While no explicit quantization was performed during the training phase, we observe greater compaction in the trained weight matrices with a higher proportion of near zero weights which is

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

promising for applications that may benefit by weight pruning, hashing or quantization techniques for compression of deep neural networks.

The rest of the paper is organized as follows: Section II covers some related work on compressive CNNs and DCT and Section III describes our proposed modification to CNNs and its implementation. In Section IV, we describe the datasets and compare the results obtained by CNNs and our model. This is followed by a discussion of the results in Section V and conclusions in Section VI.

II. Related Work

A. Compressive CNNs

Recent research in deep learning indicates that there is significant parameter redundancy in several deep learning models. Deep networks typically require large amounts of training data in addition to significant amount of computational time and resources. These requirements arise primarily owing to the fact that typical deep models learn a large number of parameters(in the millions) from the training data which are then used for predictions on the test data. This makes it quite challenging to implement online learning deep models. However recent studies suggest that some of these issues can be tackled using different approaches to compress deep networks or remove redundant parameters.

A recent work by [6] applies hashing techniques to compress neural networks. The resulting network known as HashedNets employs a low-cost hash function to group weights randomly into buckets so that all connections within a bucket share the same parameter value. This technique reduces storage requirements without deteriorating the generalization performance of the network. Another work [4] employs pruning, trained quantization and Huffman coding to achieve compression. An algorithm to automatically optimize the network architecture by selective removal of parameters at the training time is proposed in [5]. Another work in predicting the parameters in Deep Networks [7] illustrates the redundancy in parameters by using a small set of given weights to predict the remaining (95%) network weights. All these works show that there's a considerable benefit in integrating compressive techniques into deep networks.

B. Discrete Cosine Transform

The Discrete Cosine Transform(DCT) has been applied to various computer vision tasks in the past. It has been particularly successful in image compression related applications and forms the core of the JPEG standard. The DCT operation in visual image data converts 2D spatial information to spectral information which is then quantized and manipulated for compression.

Besides image compression, DCT has also been applied to other problems such as image feature extraction in the DCT domain. For example, SIFT feature extraction in the DCT domain led to fewer SIFT keypoints and higher efficiency [22]. In many Content Based Image Retrieval(CBIR) applications, different types of DCT features have been used [23],[24],[25], [26]. A significant number of images available on the internet are represented in a compressed format using the JPEG standard. Image feature extraction on these compressed images can be made more efficient by performing the extraction directly in the compressed domain. The works mentioned above are motivated by this fact and develop low-level feature extraction techniques using DCT.

In the paper on efficient image retrieval in the DCT domain by hypothesis testing [25], the authors use the KL-divergence based-ranking on empirical distributions of DCT coefficients of query and target images obtained by partial decoding of JPEG images. In another work on image retrieval in the DCT domain[23], the authors extract perceptual information in the compressed domain using the quantized histogram of statistical texture features in DCT blocks and show the robustness of this approach for image retrieval. Such examples of applications of DCT in image feature extraction are numerous and widespread [21].

III. The DCT-CNN Model

In our model, which we shall refer to as the DCT-CNN model, we add the following operation: a Discrete Cosine Transform of the feature maps generated by the Convolutional layer of the CNN. This step, performed after non-linear thresholding of the convolutional features, is followed by the usual stages of max pooling, and subsequent convolutional layers. We evaluated many ways of integrating DCT into the network and observed that the best results were achieved when the DCT operation was performed only once, following the very first convolution and threholding steps.

A. Motivation

The usual effect of DCT when applied to 8x8 blocks of image data is to concentrate most of the larger values in the block to the upper left-hand. However, in our model, we apply the DCT on the feature maps generated by the first convolutional layer in the network. A visualization of the effect of this operation on the feature maps is provided in Figure 1. The features maps generated by earlier convolutional layers in a network usually try to capture low level image features. A DCT operation performed on this map has a similar effect, visually, on the features maps as it would have on a normal image. The high intensity values are concentrated in the upper left hand corner. The features extracted from the image are thresholded and then converted to the DCT domain and thereafter follows the usual process of pooling and convolutions in the same domain. Thus subsequent feature extraction beyond the very first convolution layer, occurs in the frequency domain. It has been established in several previous studies that important perceptual information in the images are well preserved under the DCT operation. This observation serves as a motivation for our work to study the performance of deep feature extraction in the DCT domain for various image classification tasks. Another objective is to observe the effect of the transformation on the network weights and compare and contrast the network weight matrices obtained when feature

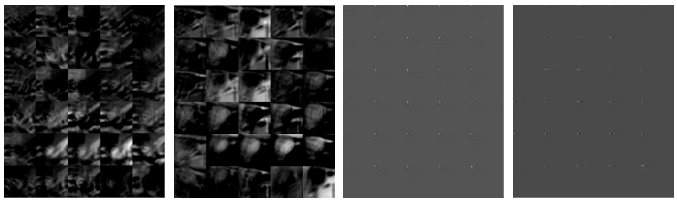

Fig. 1: Feature maps before and after DCT. Figures 1(a) and 1(b) show several feature maps generated by the first convolution layer in the network during the training phase. Figures 1(c) and 1(d) show feature maps under DCT. All the feature maps were generated simultaneously in the same iteration, by the same layer in the same network. The white specs in the top left corner of each map correspond to low frequency DCT coefficients.

extraction takes place under the transformation versus without it. The expectation is that the DCT operation within the network will result in sparser weight matrices trained over data which is more conducive to compression techniques such as hashing and quantization for efficient compression and storage of CNNs.

B. Framework

The DCT-CNN structure: Figure 2 shows the incorporation of the DCT operation in a CNN. The usual architecture of a CNN consists of an input layer followed by convolutional, thresholding and pooling layers which are repeated multiple times followed by a classifier (e.g. LogSoftmax) on top. This network structure is augmented, as shown in Figure 2, with a DCT operation following the first convolutional layer that extracts several feature maps from the input RGB image and a thresholding operation. Thus the subsequent feature extraction steps are carried out in the DCT domain.

REDFT10 transform - type II DCT: The 2D DCT operation used on the feature maps is REDFT10. An REDFT10 transform (type-II DCT, sometimes called the DCT) is defined by:

$$Y_k = 2\sum_{j=0}^{n-1} X_j cos[\pi(j+1/2)k/n] \quad (1)$$

For two-dimensional DCT, which is a separable transform, the operation is first performed on the columns of the matrix followed by the rows. Each element of the transformed list is the inner (dot) product of the input list and a basis vector. The constant factors are chosen so that the basis vectors are orthogonal and normalized. The DCT can be written as the product of a vector (the input list) and the n x n orthogonal matrix whose rows are the basis vectors.

Implementation: We augment the network by introducing this step between the first convolutional and thresholding layer and the subsequent pooling stage. We take the output of the convolutional layer which is a three or four dimensional matrix depending on the batch size [BatchSize x NumFeatureMaps x

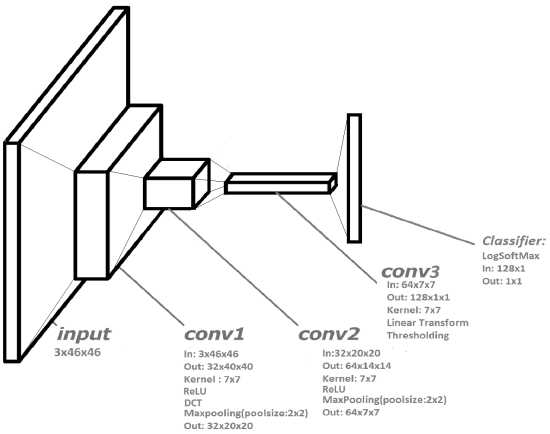

Fig. 2: A shallow CNN structure with DCT

Width x Height] followed by non-linear thresholding operation and perform DCT on this output cube layer by layer. Thereafter the transformed maps are passed onto the pooling layer for sub-sampling. Since we use small sized images(e.g. 32x32) we performed DCT on the entire image instead of any sliding windows. We studied the effect of this augmentation on fairly shallow networks. The networks had 3 convolutional layers. Maxpooling was seen to yield best results. Different types of activation functions(e.g. ReLU,Tanh) and the dropout technique were tested. Data augmentation was used in some cases to augment the small training sets. The experiments were done using CPUs. Other variations in the network structure, e.g. number of feature maps were also studied.

IV. Experiments

A. Datasets

In order to establish the differences in the performance of CNNs versus CNNs with a DCT operation, we considered the following computer vision tasks: pedestrian detection, face detection, material categorization and object detection. We used the INRIA person dataset, the ETH Zurich Pedestrian datasets, the FDDB dataset, the Purdue ELab Face dataset, the Flickr Materials Dataset and Cifar-10 dataset.

The **INRIA** Person Dataset is a large set of 1805 cropped images(64x128) of upright people in various orientations standing against a wide variety of backgrounds. It was first introduced in [8].

The **ETH Zurich** pedestrian dataset was used in [9]. It was subsequently used for experiments in Multi-Person Tracking [10] and Moving Obstacle Detection [11]. This dataset offers several sequences of different types of scenes, with frames of size 640 x 480. We used the BAHNHOF and SUNNY DAY sequences for testing and training respectively. We used the updated annotations provided in [12] which contain bounding boxes for pedestrians down to a size of 48 pixels whereas in

the original work bounding boxes of size 60 pixels or higher only were considered.

Face Detection Data Set and Benchmark(FDDB) is a dataset of face regions that was constructed to study unconstrained face detection problems [13]. The Purdue e-Lab Face-dataset is another dataset [14] on which we tested our model.

The **Flickr Material Dataset** appeared in a recent work [16] and consists of high resolution images of ten different classes of materials, namely, fabric, foliage, glass, leather, metal, paper, plastic, stone, water, and wood with 100 images per category and has been used to study human material categorization.

Finally, the **CIFAR-10** [19] dataset is an object recognition database of ten different object categories, namely, airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each class has 5000 32x32 color images for training and 1000 for testing.

B. Person Detection

A recent result on the pedestrian detection published in [15] uses a multistage, shallow deep network of two hidden layers and reports results on the Caltech, TUD Brussels and ETH Zurich datasets(trained on the INRIA dataset). For the experiments on the INRIA person dataset, we also deployed a fairly shallow CNN with the following specification:

- Input : 3x46x46
- SpatialConvolution : 3→32, 7x7
- ReLU
- DCT
- MaxPooling 2x2 , 2x2
- SpatialConvolution : 32 →64, 7x7
- ReLU
- Max Pooling 2x2, 2x2
- SpatialConvolution : 64 →128, 7x7

This was followed by reshaping and classification using a two layer MLP (linear transform and thresholding) with a LogSoftMax function applied at the end.

We trained and tested these models on the INRIA dataset using the Torch7 demo code. The models were trained on 5000 images and tested on 1000 images. Here, we have highlighted the difference between adding a DCT operation on the feature maps generated by the first convolutional layer vs. training without it (Figure 3). It is evident from the plots that the model converges within fewer training epochs when DCT is performed on the convolution feature maps (Figure 3 : left). The maximum test accuracy for model without DCT (Figure 3: right) is **99.90%** whereas with DCT the highest accuracy is **99.49%**. There is a small drop of accuracy (0.4%) but convergence was much faster.

Experiments were carried out, with the same networks on the ETH Zurich dataset. The network was trained on the SUNNY DAY sequence which has 354 frames and 1900 each of pedestrian and background images(cropped). The BAHN-HOF sequence which has 999 frames and 8467 bounding box annotations was used for testing. The maximum test accuracy achieved by the DCT-CNN network is around **96.3%**. After

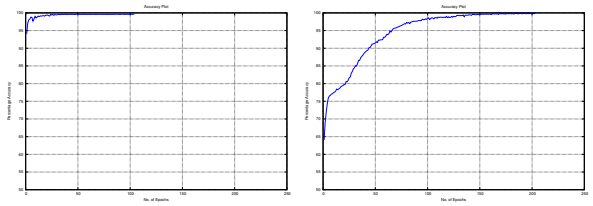

Fig. 3: Test Accuracy Plots for Pedestrian Detection,left: DCTCNN, right: CNN, on the INRIA dataset

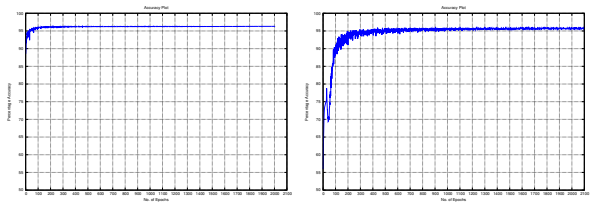

Fig. 4: Test Accuracy Plots for Pedestrian Detection, left: DCTCNN, right: CNN, on the ETH dataset

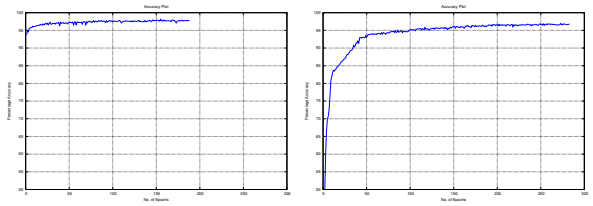

Fig. 5: Test Accuracy Plots for Face Detection , left: DCTCNN, right: CNN, on the FDDB dataset

three training epochs the model reached 90.4% accuracy and in about 100 training epochs the network reached around 96% and almost converged, with only minute changes in accuracy beyond that. The same network without the DCT layer with exactly same values of hyper-parameters was trained and tested on the same dataset again. In this case, the maximum test accuracy achieved is **95.8%**. It crossed the 90% mark at 100 epochs, and convergence to around 95% accuracy was observed in about 300 training epochs.

C. Face Detection

The network described above was used on the FDDB dataset. The training data size was 5864 and the test dataset had 1822 samples. Negative samples were generated by random crops from the background. The highest test accuracy for the model without DCT is **96.81%** whereas with DCT this value is **98.10%** Thus in this case we observed a slight gain in test accuracy besides faster convergence. The plots in Figure 5 (left : Model with DCT right: Model without DCT) show the convergence with number of epochs.

Another set of experiments was carried out on the Purdue ELab face dataset. For this we again used the available demo code for Torch7 (train-face-detector). The training set size was 33,013 whereas the test set had 8254 samples. On the

PurDue ELab Data, the DCT-CNN network achieved 95% test accuracy.

D. Material Identification

The Flickr Material Dataset(FMD) is particularly challenging. Images of objects belonging to one particular class vary widely. For example, as described in their website, "The images of the satin ribbon, the crocheted nylon cap , the stuffed snail toy, and the flannel bedding look very different from each other". Results reported on this dataset upto IJCV'13 show that the highest accuracy at 60.6% was obtained with SVMs and using SIFT, color and micro-jet features. For our experiments, we used the masks to obtain a number of small 46x46 sized croppings from the object surface that we used for training. Moreover, we left out two images per class for generating text sample croppings in the same manner. We worked with a training data size of 19600 (1960 samples per class) and 400 test samples (40 per class). We tried different CNN structures, different thresholding techniques and the dropout technique. The CNN structure that gave the best performance is provided below. Table I summarizes the values of maximum test accuracy for a few different values of hyper-parameters on networks with the following no. of feature maps :

- $32- > 64- > 128- > 128$ :: Network 1
- $64- > 128- > 256- > 256$ :: Network 2

Point wise non-linearity introduced by means of Rectified Liner Units (ReLu) provided better results than TanH. A network with higher no. of feature maps($64- > 128- > 256- > 256$) yielded 36.98% accuracy without the DCT operation. Dropout on the same network slightly diminished the performance(35.15%). The same network, with the DCT operation gave the highest accuracy of 38.3% and along with dropout, yielded 37.24%. So overall for the various network sizes and configurations we tested, feature extraction in the DCT domain proved to be helpful.

The overall performance of shallow CNNs on this dataset could not beat the SVM performance reported in [17]. In [17], the authors used many handcrafted features, split the data randomly into 50 training and 50 test images per class and contrasted the performance of various features and classifiers. The performance for one specific split was reported. In our experiments, we took 48 images per class for training and tested on two. This was done to maximize the amount of training data for the CNNs. It was observed that adding a DCT layer certainly seemed to help improve the top accuracy. Moreover convergence was faster. In general, the CNN training does well when the training data sets are larger. However, results indicate that feature extraction in the DCT domain can somewhat help the network performance even when there aren't as many training samples to capture all the variety in the dataset.

E. Object Recognition

On the Cifar-10 dataset the maximum accuracy achieved by the network in the DCT domain is 61.4% whereas the network in the visual domain yielded 68.5% accuracy. So in this case we observe a degradation in performance. However, what is worth noting in these experiments is that the CNN in the DCT domain converged a lot quicker. It started off with a higher accuracy (32.66%) and within 232 epochs it reached 61.1% accuracy. Whereas in case of the CNN in visual domain, the network started off with 8.94% accuracy and reached 61% accuracy in 734 epochs. It reached the maximum performance : 68.5% at 1712 epochs. Thus, although the maximum performance is slightly lower, the DCT-CNN is a lot faster in reaching around 60% accuracy.

TABLE I: Performance on FMD dataset

Network	Learning Rate	Dropout	DCT	Performance
1	1e-3	No	Yes	**40.65%**
1	1e-5	No	Yes	39.6%
1	1e-4	No	Yes	38.2%
1	1e-3	No	No	36.1%
1	1e-4	No	No	35.2%
2	1e-3	No	Yes	38.3%
2	1e-3	Yes	Yes	37.24%
2	1e-3	No	No	36.98%
2	1e-3	Yes	No	35.15%

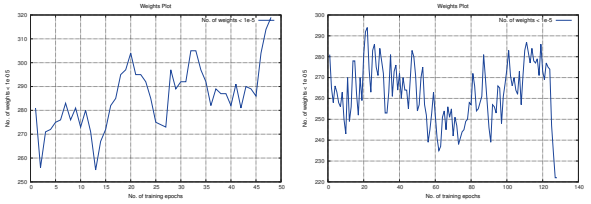

Fig. 6: Plots showing variation in count of weights $abs(w) < 1e-5$ for left: DCTCNN, right: CNN on the FDDB dataset

F. Effect on Trained Weight Matrices

The DCT operation within the CNN also led us to observe certain effects on the trained weight matrices. In Table II, we report the number of weight parameters with absolute values less than $1e^{-5}$ learned in the convolution modules in models trained using a CNN and a DCT-CNN on various datasets. The number of training epochs after which this number is reported is different for CNNs and DCT-CNNs. This is because the latter reaches *maximum accuracy* more quickly. Also included is a plot [Figure 6 (left: DCT-CNN, right: CNN)] showing how the total number of weights($abs(w) < 1e^{-5}$) vary on FDDB dataset. DCT-CNNs achieve both greater test accuracy and more near-zero weights in a shorter period of time. In case of the trained networks on INRIA and FDDB datasets, when we round all the weights ($abs(w) < 1e^{-3}$) to zero, no significant change in performance is observed for both the DCT-CNN and the CNN models.

V. Discussion

This work was structured in a way so that it brings out the effects of applying DCT on the feature maps of a CNN. The accuracy plots of models with and without the DCT step

TABLE II: Weight Matrix Elements

Dataset	DCT	No. of epochs	No. of weights: $abs(w) < 1e-5$	Test Accuracy
INRIA	Yes	56	303	99.49%
INRIA	No	151	257	99.49%
FDDB	Yes	49	319	96.9%
FDDB	No	254	256	96.8%

have been contrasted with each other. In almost all the cases, with different variations of the network and various values of hyperparameters such as learning rate, momentum, weights decay rate etc. and on all these different types of datasets, we observed a higher test accuracy for the DCT-CNNs in the very initial stages of network training and found faster convergence of the network. During the training phase we retain the low energy terms to maintain uniform size of the feature maps. The DCT operation performed at later stages on the network, (after the application of the non-linear operations like pooling or thresholding) did not yield good results. We observed an advantage(in terms of faster convergence) when DCT was applied right after the first convolution step.

The complexity of applying a DCT-II on an image of size NxN is $O(N^2 \log(N))$. While it is a fact that more work is being done per epoch to perform DCT, by parallelizing the operation faster implementations can be achieved so that multiple feature maps can be transformed simultaneously. In applications where the model weights need frequent updating and retraining, such gains can add up to saving a lot of time over a period. For example, even without parallelizing, for the INRIA dataset, per sample training time on a 1.6 GHz Intel Xeon Processor was 7ms for DCT-CNN and 4ms for the CNN. However, in about 15 epochs, the DCT-CNN reached 99% training accuracy which is faster than the CNN which reached 99% in 129 epochs. therefore the DCT-CNN is five times faster in this case, even though it was not parallelized. This ratio can be further improved by performing DCT in parallel over multiple feature maps. The time improvement will be even more dramatic for networks where the number of output maps from the very first convolutional layer is large. In some of the cases the DCT operation was observed to improve the test performance slightly. Moreover the weight matrices of the trained models contain fewer of the larger weights and hence it is easier to prune some of the inactive connections. All these experiments were CPU based and carried out on relatively shallow CNNs. Such models with lesser computational complexity can be more readily adapted for online learning purposes.

VI. Conclusion

This paper presents the results of an investigation into the effects of deep feature extraction in the DCT domain using CNNs. It compares them with the results of the same networks in the visual domain. The observations indicate that the network convergence is faster in the training phase when feature extraction takes place in the DCT domain and yet performance is comparable, or in some cases, better. Thus this could be an effective technique for training online models that require frequent training and updating of weights. Moreover the weight matrices of the final trained model have a higher proportion of elements close to zero when the DCT operation is integrated into the network. This can be useful for hashing and quantization aimed at compressing convolutional neural networks for storage purposes.

References

[1] M. Jaderberg, A. Vedaldi and A. Zisserman, *Speeding up Convolutional Neural Networks with Low Rank Expansions.* In BMVC, 2014.

[2] M. Mathieu, M. Henaff and Y. Lecunn, *Fast Training of Convolutional Networks through FFTs.* *arXiv* preprint *arXiv*:1312.5851,2014.

[3] E. Oyallon, L. Sifre, S.E. Mallat *Generic Deep Networks with Wavelet Scattering* *arXiv*preprint *arXiv*:1312.5940, 2014.

[4] S. Han, H. Mao, and W.J. Dally *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding* In ICLR 2016

[5] M. Kim, L. Rigazo *Deep Clustered Convolutional Kernels* In NIPS 2015

[6] W. Chen, J. T. Wilson, S. Tyree, K.Q. Weinberger, and Y. Chen *Compressing Neural Networks with the Hashing Trick* In ICML 2015.

[7] M. Denil, B. Shakibi, L. Dinh, MarcAurelio Ranzato, and Nando de Freitas *Predicting Parameters in Deep Learning* In NIPS 2013.

[8] N. Dalal and B. Triggs *Histograms of Oriented Gradients for Human Detection* In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2005.

[9] A. Ess, B. Leibe, and L. van Gool *Depth and Appearance for Mobile Scene Analysis* In Proceedings of ICCV 2007.

[10] A. Ess, B. Leibe, K. Schindler and L. van Gool *A Mobile Vision System for Robust Multi-Person Tracking* In Proceedings of CVPR 2008.

[11] A. Ess, B. Leibe, K. Schindler and L. van Gool *Moving Obstacle Detection in Highly Dynamic Scenes* In Proceedings of ICRA 2009.

[12] Christian Alexander Wojek *Monocular Visual Scene Understanding from Mobile Platforms* TU Darmstadt, Ph.D. Thesis (2010)

[13] V. Jain and E. Learned-Miller *FDDB: A Benchmark for Face Detection in Unconstrained Settings* In UM-CS-2010-009 University of Massachusetts, Amherst, 2010

[14] C. Farabet, E. Culurciello *Face Detector e-Lab* In torch7-demos, *https://github.com/e-lab/torch7-demos/tree/master/train-face-detector-elab*

[15] Xi. Zeng W. Ouyang and X. Wang *Multi-Stage Contextual Deep Learning for Pedestrian Detection* In ICCV, 2013.

[16] L. Sharan, R. Rosenholtz, and E. H. Adelson *Material perception: What can you see in a brief glance?* In Journal of Vision, vol. 14, no. 9, article 12, 2014

[17] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson, *Recognizing materials using perceptually inspired features*, In International Journal of Computer Vision, vol. 108, no. 3, pp. 348-371, 2013

[18] C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz, *Exploring features in a Bayesian framework for material recognition*, In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 239-246, 2010

[19] A. Krizhevsky *Learning Multiple Layers of Features from Tiny Images*, 2009.

[20] M.D. McDonnell and T. Vladusich *Enhanced Image Classification With a Fast-Learning Shallow Convolutional Network* In IJCNN, 2015.

[21] Gregory K. Wallace *The JPEG still picture compression standard* In IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, FEBRUARY 1992.

[22] W. Zhen, X. Zhe, R. Zhang and S. Li Shao-Mei *SIFT Feature Extraction Algorithm for Image in DCT Domain* In ISCCCA, 2013.

[23] Fazal-e-Malik , B. B. Baharudin and K. Ullah *Efficient Image Retrieval Based on Quantized Histogram Texture Features in DCT Domain* In IEEE Frontiers of Information Technology , 2011.

[24] Wenyin Zhang, Zhenli Nie, Zhenbing Zeng *Image Retrieval Based on Salient Points from DCT Domain* In LNCS Volume 3789, pp 386-395

[25] D. He, Z. Gu, N. Cercone *Efficient image retrieval in DCT domain by hypothesis testing* In ICIP 2009

[26] G. Feng, J. Jiang *JPEG Image Retrieval Based on Features from DCT Domain* InLNCS Volume 2383, pp 120-128