Texture classification using Deep Multitask Learning

Shantanu Ghosh
Computer and Information Science and Engineering
University of Florida
Gainesville, USA
shantanughosh@ufl.edu

Dapeng Oliver Wu
Department of Electrical & Computer Engineering
University of Florida
Gainesville, USA
dpwu@ufl.edu

***Abstract*—Texture bestows important characteristics of many types of images in computer vision and classifying textures is one of the most challenging problems in pattern recognition which draws the attention of computer vision researchers over many decades. Recently with the popularity of deep learning algorithms, particularly Convolution Neural Network (CNN), researchers can extract features that helped them to improve the performance of tasks like object detection and recognition significantly over previous handcrafted features. In texture classification, the CNN layers can be used as filter banks for feature extraction whose complexity will increase with the depth of the network. In this study, we introduce a novel multitask texture classifier(MTL-TCNN) where we used multitask learning instead of pretraining sharing feature representation between two common tasks; one task being identifying the objects from Imagenet dataset using Alexnet and second, being classifying the textures using TCNN3. For evaluation, we used two standard benchmark datasets (KTH-Tips and DTD) for texture classification. Our experiments demonstrated enhanced performance classifying textures over TCNN3.**

***Index Terms*—texture classification, convolution neural network, TCNN, multitask learning**

I. Introduction

Texture, which is one of the powerful visual cues, like color provides important information of images and texture classification is essential in many computer vision and pattern recognition problems such as material classification, object detection, and scene recognition. Texture regions or images obey some statistical properties exhibiting repeated structures known as textons with some degree of variability in their appearance and relative positioning [1]–[3]. Designing feature descriptors that can attune large intra-class variation and low inter-class variation in a much effective way has been the center of attention across various literature. Before the deep learning era, the most widely used approach to classify textures was based on the mid-level encoding of handcrafted local texture descriptors, for example, Gabor [4], LBP(local binary pattern) [5], GLCM(gray-level co-occurrence matrix) [6] and HOG(histogram of oriented gradients) [7] features of the defects are extracted, then the defects are classified by machine learning classifiers, such as SVM(support vector machines) [8], Random Forests [9], AdaBoost [7], and so forth. One of the major drawbacks of these traditional methods is the requirement of experts with the experience of image preprocessing, feature extraction, feature reduction, and classifier selection.

With the recent development in deep learning, the convolution neural networks(CNNs) [10] achieved start of the art accuracy in problems in computer vision such as object recognition and classification as features extracted by CNNs are more generally more discriminative than other methods which make CNNs suitable for the problem of texture classification. In general, the convolution layers which is present deeper in a network are responsible for extracting more complicated features that are used by the fully connected layers to obtain information about the overall shape of the image. In the texture classification, this overall shape analysis is inadequate. However, researchers in texture analysis have paid great attention to the dense orderless features which are extracted by the initial layers of a classic CNN.

In this study, we introduce a multitask CNN classifier(MTL-CNN) that we built on top of TCNN3 [11] architecture and Alexnet architecture [10] - both of which are combined using multitask learning [12]. In most of the networks, the popular approach to train is to pretrain the network with some large dataset like Imagenet first to extract edge like features which detecting the shape of an object of an image and then train it with the necessary texture dataset like kth-tips [13] or DTD [14]. One of the limitations of this approach is the time taken to train the CNN. However, as both tasks are similar, we can reduce the training time by amalgamating these two tasks using a multitask classifier where the primary task is the texture classification using texture specific dataset and the auxiliary task is to detect the shape of the object from large dataset like Imagenet. Our experiments have shown our model using this approach trained from scratch, generalizes well by reducing the over-fitting problem.

We used two simple networks Alexnet and TCNN3 in our multitask classifier instead of using complicated network architectures like Resnet [15] and Fisher Vector CNN (FV-CNN) [16] as we want to study the effect of multitask learning in texture classification. The rest of the paper is organized as follows: in Section II, the related work is articulated. In Section III, the architecture of the multitask classifier MTL-TCNN is introduced and discussed in details. In Section IV, we describe our experimental protocol and discuss our

results. Finally we conclude the paper in Section VI. The code is available under the MIT license on Github at: https://github.com/Shantanu48114860/

II. Related work

The initial work of texture classification by applying a basic CNN architecture with only four layers inorder to classify Brodatz dataset has been discussed in [17]. After the recent popularity of deep neural network, CNN architecture discussed in [18] is used to solve the problem of forest species classification much similar to texture classification. Compared to [17], the CNN architecture used in [18] is more complex but the classification resut is more impressive compared to the former.

Cimpoi [16] first illustrated how important it is to to extract texture descriptions densely from a CNN with FV-CNN which helped them to obtain impressive results in texture recognition. This method is well suited to region recognition as it computes the convolution layer output once and then pools regions with FV separately. One drawback of the FV-CNN is that is the CNN of FV-CNN does not learn from the texture dataset.

A much simpler yet effective network is discussed in TCNN [11] where we the help of an energy layer, the authors were trying to classify the textures. Based on the popular architecture of AlexNet [10], the authors of TCNN used the convolution layers to extract edge-like features just as normal filter bank. Then they used energy layer simply performed the average pooling operation from the feature descriptor of the last convolution layer which helped them to ascertain the big picture from the specific feature descriptor. The architecture is simple yet effective in classifying textures more accurately.

III. Model Description

The basic TCNN3 [11] architecture is motivated from Alexnet [10] where TCNN3 extracts features using convolution and max pooling operations. For texture classification, they introduced a new energy layer where each feature map of the last convolution layer is simply pooled by calculating the average of its rectified linear activation output [11].In this section, we describe how we adopted TCNN3 architecture and amalgamate with Alexnet to create a multitask network for better feature extraction which will help us to achieve better classification result in Kth-tips dataset.

A. Multitask Learning

Multitask Learning (MTL) [12] is a learning paradigm in machine learning which aims to utilizes shared features contained in multiple related tasks which helps to reduce the problem of overfitting in deep artificial neural networks and generalizes the model in much better way. MTL is useful in those problems where we intend to find prediction in multiple tasks related to each other, for example finance or economics forecasting, bioinformatics etc. In most of the scenarios, we often are interested about the performance of one task. To make this happen, we often use another appropriate task as auxiliary task in order to use the advantages of MTL. The primary goal of using an auxiliary task which boosting the performance of the primary task in a Multitask network is to reduce the overfitting and helps the network to generalize well.

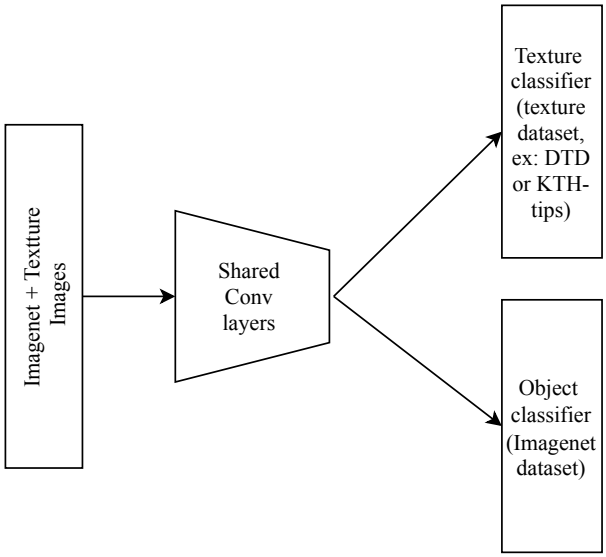

Fig. 1. Block diagram of MTL-TCNN

B. Combining Multitask Learning with Alex Net and TCNN

Given the texture classification problem, this idea of MTL is useful as in many cases to extract complicated features of texture within an image, the researchers use the deep and complicated network which suffers from the problem of overfitting. Also, if we simply use a simple network like TCNN3, it will unable to extract the complicated features from an image and on the other hand, the use of networks like Alexnet in the problem of texture classification will not be helpful as Alexnet intends to detect and recognize objects. However, the initial layers of Alexnet can be thought of as filter banks which helps us to extract edge-like features that are useful in classifying textures. The initial layers of TCNN3 are the same as Alexnet which also extracts the edge-like features whereas the TCNN3 architecture uses average pooling within a feature map to extract the big picture from the feature map and helps in extracting the most important features which are more relevant to texture classification. Motivated by the commonality of the initial feature extraction technique used by the two simple networks - Alexnet and TCNN3 corresponding to the tasks of object detection and texture classification respectively, we created a multitask network MTL-TCNN, where the initial common layers of Alexnet and TCNN3 are encapsulated in the shared convolution layer of MTL-TCNN and layers for texture classification and object detection are shown in two separate tasks, shown in figure 1. Here object detection acts as an auxiliary task that serves as a regularizer which helps the network to generalize well.

IV. EXPERIMENTS

A. Network details

MTL-TCNN is divided into three parts - 1) Shared convolution network, consisting of the shared convolution layers between Alexnet and TCNN3, 2) Texture classifier network consisting of the energy layer and the fully connected layers of TCNN3 specific to Texture classification task and 3) Object classifier network consisting of the remaining convolution layers and fully connected layers of the Alexnet responsible for object detection task which is the auxiliary task. The details of each network is explained in tables I, II, III respectively.

We used Pytorch framework to develop the network. We utilized Adam optimiser [19] with a learning rate and weight decay of 0.0001 and 0.0005 and a batch size of 32 for both object detection and texture classification tasks respectively. We cropped all the images to 227 x 227 to train MTL-TCNN. The training strategy employed to train MTL-TCNN is discussed in Algorithm 1. We did not use pre-training strategy using all the images of a large image dataset like *ImageNet* 2012 as by the most of the researchers, we used handpicked images from ImageNet, relevant to classifying textures and train them with a given texture dataset in end to end manner.

TABLE I
SHARED CONVOLUTION NETWORK ARCHITECTURE

Size / operation	Filter	Depth	Stride	Padding
Conv1 + Relu	11 * 11	96	4	
Max Pooling	3 * 3		2	
Conv2 + Relu	5 * 5	256	1	
Max Pooling	3 * 3		2	
Conv3 + Relu	3 * 3	384	1	1

TABLE II
TEXTURE CLASSIFIER ARCHITECTURE

Size / operation	Filter	Depth	Stride	Padding
Average Pool	13 * 13		1	
FC6 + Relu				
FC7 + Relu				

TABLE III
OBJECT CLASSIFIER ARCHITECTURE

	DTD	Depth	Stride	Padding
Conv4 + Relu	3 * 3	384	1	1
Conv5 + Relu	3 * 3	256	1	1
Max Pooling	3 * 3		2	
FC6 + Relu				
FC7 + Relu				

B. Datasets

We used a total of 3 datasets - 1) We handpicked images from *ImageNet* 2012 dataset that are more relevant to texture classification, and 2) DTD and Kth-tips datasets for Texture classification. The ImageNet 2012 dataset [10] contains 1000 classes. The train- ing set contains 1,281,167 images and the validation set (used for testing) 50,000 images (50 images/class) of size 256×256. As we are using MTL, we did not pre-train the MTL-TCNN using this huge dataset, we selected 56 classes of texture images from ImageNet as per Appendix A and Appendix C mentioned in [11] and created the dataset for the object detection task. The idea behind this strategy is to have images with common features as the images in the texture datasets. So, when we employ MTL to train the network, the network extracts the shared features which helps it to regularize to prevent overfitting. In this way, we created a subset of ImageNet consisting of 38951 images across 56 classes and cropped it to 227 x 227.

For kth-tips dataset, the official link https://www.nada.kth.se/cvap/databases/kth-tips/kth-tips2.pdf is dismissed, so we downloaded the entire kth-tips dataset the data mentioned in [20]. This dataset contains 3194 images of 11 classes including kth-tips and kth-tips2. We divided the dataset into train and test set randomly with train set size as 80% of the entire dataset. Then we resized each image of the dataset to 227 x 227. We continued this for 10 iterations and report the average of all the classification results.

DTD [14] is a texture dataset containing 47 classes of 120 images *"in the wild"* each. As this dataset contains images of multiple sizes, we cropped all the images to 227 x 227 same way like we did for the other datasets. This dataset is divided into 10 annotated splits with 40 training, 40 validation and 40 test images for each class. The final classification result is averaged over this 10 splits.

Algorithm 1 Training MTL-TCNN

1: Resize all the images of texture dataset to 3 x 227 x 227,
2: Initialize model parameters Θ randomly.
3: Handpick images from ImageNet dataset which are related to textures and create dataset for object detection task(auxilary task).
4: Create batches from ImageNet and texture datasets
5: **for** $epochs = 1, 2, \ldots$ **do**
6: Merge all the batches generated in step 4
7: Randomly shuffle all the batches
8: **for** $batches = 1, 2, \ldots$ **do**
9: Compute the classification loss based on the batch of the task: $L(\Theta)$ as follows,
$L(\Theta)$ = Loss for object classification
$L(\Theta)$ = Loss for texture classification
10: Compute Gradient: $\nabla(\Theta)$
11: Update Model: $\Theta = \Theta - \eta\nabla(\Theta)$
12: **end for**
13: **end for**

V. RESULTS

The results of our experiments is mentioned in the Table IV. We compared our results with the end to end result of TCNN3 architecture. While for the TCNN3 architecture with end to end training, 27.8 % ($\pm$ 1.2) classification accuracy

was achieved, we achieved a classification accuracy of 33.8 % ($\pm$ 0.014). For the kth-tips-2b dataset, TCNN3 architecture achieved an accuracy of 48.7% ($\pm$1.3), MTL-TCNN achieved 86.2 % ($\pm$ 0.02) but we trained MTL-TCNN with entire Kth-tips dataset due to the break down of the official website of the Kth-tips-2b dataset.

TABLE IV
CLASSIFICATION RESULTS (ACCURACY %)

Architecture	DTD	Kth-tips
TCNN3	27.8 $\pm$ 1.2	-
MTL-TCNN	33.8 $\pm$ 0.014	86.2 $\pm$ 0.02

VI. CONCLUSION

In this article we developed a new architecture MTL-TCNN motivated by multitask learning and TCNN. However, we can see that the accuracy of DTD dataset is comparatively low compared to the state of the art models for DTD dataset, we believe that our model for texture classification task is very simple and we also have not used the pre-training. While our approach is novel in solving the texture classification problem as we are the first ones to use the power of multitask learning in texture classification problem ,in future research, the accuracy of classifying textures can also be improved by using FV-CNN instead of TCNN for texture classifier and complicated convolution architectures like ResNet [15]. Also we can use noise to generate new texture images from existing ones in the dataset to augment the texture dataset to help the model to extract more complicated features which will further help to improve the classification result.

REFERENCES

[1] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[2] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International journal of computer vision*, vol. 43, no. 1, pp. 29–44, 2001.

[3] D. Forsyth and J. Ponce, "Computer vision: A modern approach. always learning," 2012.

[4] R. Medina, F. Gayubo, L. M. González-Rodrigo, D. Olmedo, J. Gómez-García-Bermejo, E. Zalama, and J. R. Perán, "Automated visual classification of frequent defects in flat steel coils," *The International Journal of Advanced Manufacturing Technology*, vol. 57, no. 9-12, pp. 1087–1097, 2011.

[5] M. Chu and R. Gong, "Invariant feature extraction method based on smoothed local binary pattern for strip steel surface defect," *ISIJ International*, vol. 55, no. 9, pp. 1956–1962, 2015.

[6] J. L. Raheja, S. Kumar, and A. Chaudhary, "Fabric defect detection based on glcm and gabor filter: A comparison," *Optik*, vol. 124, no. 23, pp. 6469–6474, 2013.

[7] D. Shumin, L. Zhoufeng, and L. Chunlei, "Adaboost learning for fabric defect detection based on hog and svm," in *2011 International conference on multimedia technology*, pp. 2903–2906, IEEE, 2011.

[8] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858–864, 2013.

[9] B.-K. Kwon, J.-S. Won, and D.-J. Kang, "Fast defect detection for various types of surfaces using random forest with vov features," *International Journal of Precision Engineering and Manufacturing*, vol. 16, no. 5, pp. 965–970, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[11] V. Andrearczyk and P. F. Whelan, "Using filter banks in convolutional neural networks for texture classification," *Pattern Recognition Letters*, vol. 84, pp. 63–69, 2016.

[12] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[13] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh, "On the significance of real-world conditions for material classification," in *European conference on computer vision*, pp. 253–266, Springer, 2004.

[14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[16] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3828–3836, 2015.

[17] F. H. C. Tivive and A. Bouzerdoum, "Texture classification using convolutional neural networks," in *TENCON 2006-2006 IEEE Region 10 Conference*, pp. 1–4, IEEE, 2006.

[18] L. G. Hafemann, L. S. Oliveira, and P. Cavalin, "Forest species recognition using deep convolutional neural networks," in *2014 22nd International Conference on Pattern Recognition*, pp. 1103–1107, IEEE, 2014.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Y. Huang, C. Qiu, X. Wang, S. Wang, and K. Yuan, "A compact convolutional neural network for surface defect inspection," *Sensors*, vol. 20, no. 7, p. 1974, 2020.