International Conference on Image Analysis and Recognition, vol. I, pp. 66-70, Jun. 2020.

Detecting Macroblocking in Images Caused by Transmission Error

Ganesh Rajasekar and Zhou Wang

Department of Electrical & Computer Engineering, University of Waterloo
200 University Ave W, Waterloo, ON, N2L 3G1, Canada
Emails: g3rajase@uwaterloo.ca, zhou.wang@uwaterloo.ca

Abstract. Macroblocking is a type of widely observed video artifact where severe block-shaped artifacts appear in video frames. Macroblocking may be produced by heavy lossy compression but is visually most annoying when transmission error such as packet loss occurs during network video transmission. Since receivers do not have access to the pristine-quality original videos, macroblocking detection needs to be performed using no-reference (NR) approaches. This paper presents our recent research progress on detecting macroblocking caused by packet loss. We build the first of its kind macroblocking database that contains approximately 150,000 video frames with labels. Using the database, We make initial attempts of using transfer learning based deep learning techniques to tackle this challenging problem with and without using the Apache Spark big data processing framework. Our results show that it is beneficiary to use Spark. We believe that the current work will help the future development of macroblocking detection methods.

Keywords: Macroblocking · Packet loss · Transmission artifacts · No-reference Image quality assessment · Deep learning · Apache Spark

1 Introduction

Macroblocking is a video artifact in which objects or areas appear to be made up of blocks rather than proper details in the original content. The blocks may appear throughout the image, or just in certain regions. Macroblocking may occur due to heavy video compression, especially in video frames of fast motion or abrupt scene changes, but the most annoying types of macroblocking in practical visual communication systems are often caused by transmission errors such as packet loss. The latter is the main focus of the current research.

Early work in detecting macroblocking caused by packet loss [1] used subjective test to identify circumstances of packet loss, and constructed a classifier that uses objective factors to predict macroblocking. A strong correlation is observed in the variability of mean opinion scores against packet loss levels [2]. The impact of packet loss on QoE in video streaming services is reviewed in [3]. In [4][5], various pixel level statistical features are extracted to detect macroblocking. However, packet loss based assessment is not applicable in the scenarios

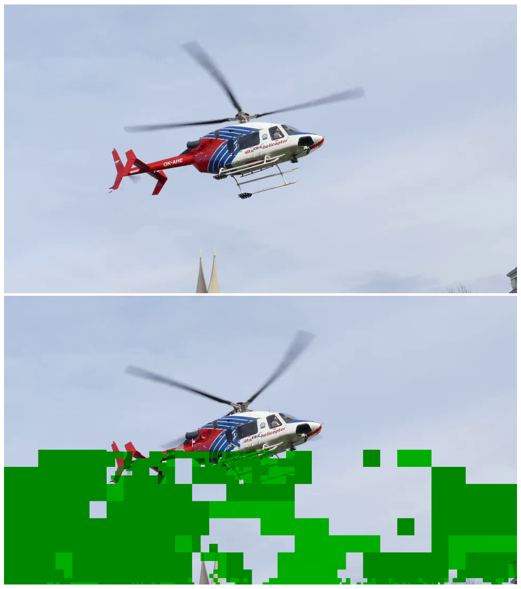

Fig. 1: Sample images without (top) and with (bottom) macroblocking

that the packet information is not available, for example, when the video frames are decoded. In addition, they are not capable of localizing the macroblocking artifacts in the exact video frames and spatial locations. In [6], a neural image assessment model based on transfer learning is proposed that is claimed to produce quality scores well correlated with distributions of subjective scores in a limited test. Nevertheless, the performance of existing methods has not been fully validated largely due to the lack of large-scale high-quality databases that cover a wide range of image content and packet loss levels. Furthermore, the potentials of transfer learning [7] has not been thoroughly investigated.

2 Macroblocking Database Construction

We create the largest of its kind database of video frames of macroblocking caused by packet loss. The database construction is divided into three steps: video clip/frame extraction, macroblocking simulation, and image labeling.

A set of original videos (around 25) of 10-second in length, 1080p resolution (1920×1080), and 24 or 30 frames per second (fps), are collected and all individual frames are extracted. Each video is encoded using 3 types of video codecs of MPEG2, H.264 and H.265, respectively. Macroblocking is simulated by randomly dropping packets from different frames of the video using OpenCV library at 7 drop rate percentages of 1, 5, 10, 20, 50 and 100%, respectively. The 100% drop rate was used to augment the macroblocked class to reduce class imbalance. Therefore, a total of 25*3*7 = 525 video clips are generated with around 150,000 frames. Figure 1 shows sample images with and without macroblocking.

Given the number of frames, manually labeling all of them is infeasible. Thus we use a semi-automatic method to label the images. We first compute the PSNR values of all images and pre-determine two threshold values on the scale of PSNR. Any frame that has a PSNR value larger than the higher PSNR threshold is considered a frame free of macroblocking. Any frame that has a PSNR value smaller than the lower PSNR threshold is considered a frame of macroblocking. Any frame that has a PSNR value larger than the lower threshold and smaller than the higher threshold is visually inspected and assigned a label (macroblocking or not) by a human subject.

3 Macroblocking Detection

The built database allows us to explore machine learning approaches for macroblocking detection, for which we present the results of our initial attempts here. Motivated by the success of transfer learning based approaches designed for other image classification problems [8], we opt to use the framework and pipelines shown in Fig. 2.

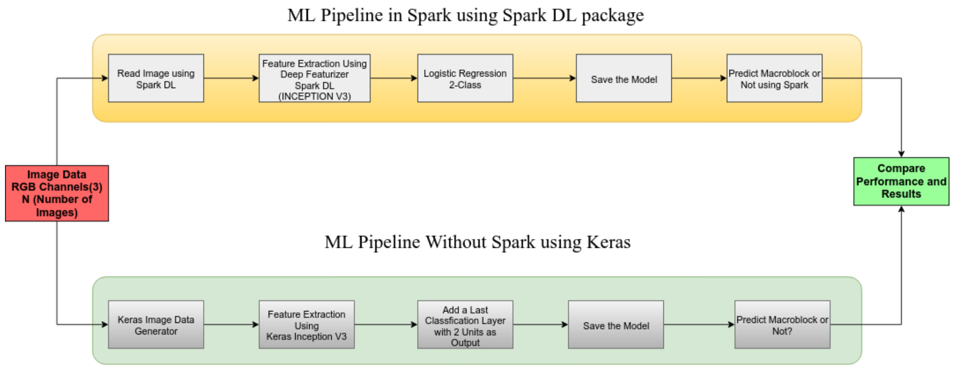
ML Pipeline in Spark using Spark DL package
Read Image using Spark DL
Feature Extraction Using Deep Featurizer Spark DL (INCEPTION V3)
Logistic Regression 2-Class
Save the Model
Predict Macroblock or Not using Spark
Image Data RGB Channels(3) N (Number of Images)
Compare Performance and Results
ML Pipeline Without Spark using Keras
Keras Image Data Generator
Feature Extraction Using Keras Inception V3
Add a Last Classfication Layer with 2 Units as Output
Save the Model
Predict Macroblock or Not?

Fig. 2: Methodology comparison of Spark and non-Spark way to classify images

For data loading, in terms of pipeline 2, in the non-Spark way we use keras image data generator that reads images in batches from file directory and converts them into tensors on the go. In pipeline 1, the spark way we use the efficient readImages library from SparkDL package. This reads a directory of images into a spark dataframe. The dataframe encodes the images as an Im-ageSchema which contains all the pixel level data and other metadata of the image. This efficient read is a parallel read and on the go, stores all the image data in a spark dataframe. For feature extraction, we use a state-of-the-art pre-trained machine learning model (Inception V3) and remove the last classification layer. We feed the Inception V3 feature vector to our own classification model trained to classify the images. In the non-Spark way we directly use the Keras

Method	**Train(70%)**	**Validation(15%)**	**Test(15%)**	**Speed**
Transfer Learning (Keras)	87.6%	82.1%	78.7%	2.10 sec
Transfer Learning (Spark)	88.3%	83.4%	80.1%	0.38 sec

Table 1: Classification accuracy and speed (second/test image) comparison.

Inception V3 model and extract the features sequentially. In the spark pipeline we use SparkDL DeepImageFeaturizer library which performs the same task but extracts the Image features in parallel from the Image loader step. This way it extracts features much faster. In model training we use logistic regression on top of the Inception V3 features. We create the Spark ML pipeline using the "pyspark.ml.Pipeline" package and add DeepImageFeaturizer and LogisticRegression as the stages and run for 20 epochs with regularization parameter of 0.05. In the non-Spark pipeline we use a fully-connected last layer with output units as 2. For the pipeline we use a keras pipeline with the keras image data generator, Inceptionv3 feature extraction and the fully connected last layer as stages and we finally fit the model on the training data. The model once trained is saved and used to predict on the testing data. In spark pipeline we do this parallelly using the transform method from sparkdl and in the keras pipeline we have to do this sequentially.

We use accuracy (percentage of correct classification) to evaluate the prediction results on the test set, and compute the training time for the 20 epochs to compare the run time of both the models. Accuracy here refers to the ratio between correct classification and total number of samples. We use a 10x10 fold validation approach. We ensure each fold represents the original data distribution as close as possible. The results are tabulated in Table 1. The same model was tested on two different frameworks while varying the size of the data (number of images) and the running time was compared and plotted in Figure 4. It can be observed that the non-Spark method does not scale well with the data size whereas using Spark achieves the same accuracy at roughly $\frac{1}{10^{th}}$ of the time.

4 Conclusion and Future Work

In this paper we present our recent research progress on automatic detection of macroblocking caused by transmission errors such as packet loss. We construct the largest database of its kind that is composed of around 150,000 images with or without macroblocking artifacts. Using the database, we investigated two transfer learning approaches with and without using the Apache Spark big data processing framework. The results clearly show that the model performs well on the database with close to 80% accuracy with minimal fine tuning, and it is beneficiary to use Spark. The major bottleneck in terms of training with large number images is that a sequential disk read training is much slower when compared to a distributed training on spark cluster. We believe that the built database and the attempted methods will help the future development of macroblocking detection methods. In the future, the built database may be used

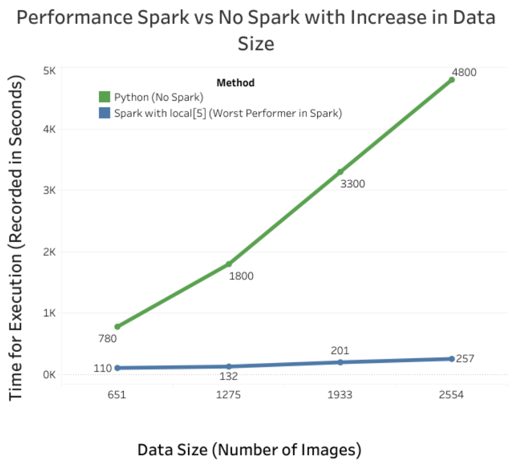

Fig. 3: Running time for Spark Vs No-Spark methods as functions of data size

for other machine learning based approaches. Pixel based approaches may also be incorporated as an additional feature layer on top of the transfer learning approach to further improve the accuracy of macroblocking detection.

References

1. A. R. Reibman, S. Kanumuri, V. Vaishampayan and P. C. Cosman, "Visibility of individual packet losses in MPEG-2 video," 2004 International Conference on Image Processing, 2004, Singapore, 2004, pp. 171-174 Vol. 1.
2. J. Nightingale, Q. Wang, C. Grecos and S. Goma, "Subjective evaluation of the effects of packet loss on HEVC encoded video streams," 2013 IEEE Third International Conference on Consumer Electronics Berlin (ICCE-Berlin), Berlin, 2013, pp. 358-359.
3. J. Greengrass, J. Evans and A. C. Begen, "Not All Packets Are Equal, Part 2: The Impact of Network Packet Loss on Video Quality," in IEEE Internet Computing, vol. 13, no. 2, pp. 74-82, March-April 2009.
4. I. Glavota, M. Vranješ, M. Herceg and R. Grbić, "Pixel-based statistical analysis of packet loss artifact features," 2016 Zooming Innovation in Consumer Electronics International Conference (ZINC), Novi Sad, 2016, pp. 16-19.
5. M. Vranjes, M. Herceg, D. Vranjes and D. Vajak, "Video Transmission Artifacts Detection Using No-Reference Approach," 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, 2018, pp. 72-77.
6. H. Talebi and P. Milanfar, "NIMA: Neural Image Assessment," in IEEE Transactions on Image Processing, vol. 27, no. 8, pp. 3998-4011, Aug. 2018.
7. J. Brownlee, "Transfer Learning in Keras with Computer Vision Models" Sept. 2019.
8. "Zied Sellami", "Making Image Classification Simple With Spark Deep Learning", https://medium.com/linagora-engineering/making-image-classification-simple-with-spark-deep-learning-f654a8b876b8