To be presented at *IEEE Multimedia Signal Processing Workshop (MMSP)*, Vancouver, BC, Aug. 2018

DFTS: Deep Feature Transmission Simulator

Harshavardhan Unnibhavi
Department of Electronics Engineering
Indian Institute of Technology (ISM), Dhanbad, India
harshanavkis@gmail.com

Hyomin Choi, Saeed Ranjbar Alvar, and Ivan V. Bajić
School of Engineering Science
Simon Fraser University, Burnaby, BC, Canada
{chyomin, saeedr, ibajic}@sfu.ca

***Abstract*—*Collaborative intelligence* is a deployment paradigm for deep AI models where some of the layers run on the mobile terminal or network edge, while others run in the cloud. In this scenario, features computed in the model need to be transferred between the edge and the cloud over an imperfect channel. Here we present a simulator to help study the effects of imperfect packet-based transmission of deep features. Our simulator is implemented in Keras and allows users to study the effects of both lossy packet transmission and quantization on the accuracy.**

***Index Terms*—Collaborative intelligence, deep feature transmission, quantization, packet loss**

I. Prototype Description

A recent study [1] has shown that the power usage and latency of inference by deep AI models on mobile devices can be minimized if the model is split into two parts: one that runs on the mobile and the other that runs in the cloud. Such deployment strategy has been termed *collaborative intelligence*. Running deep models in this way requires compression and transmission of deep feature values between the mobile and the cloud. Initial studies on deep feature compression have been reported in [2]. Here we present a simulator intended for studying deep feature transmission over unreliable channels (Fig. 1) and the effect of packet loss on the model's accuracy.

The simulator runs with Keras models. The user selects at which layer the model is split and specifies whether uniform n-bit quantization is applied to deep features prior to transmission. Packetization of feature values is done row-by-row, then channel-by-channel across the feature tensor. The user can specify how many rows of features are placed in a packet.

Two channel models are provided: independent loss channel, where the user can specify the loss probability P_B, and a two-state Markov channel (a.k.a. bursty, or Gilbert channel), where the user can specify the loss probability P_B and the average burst length L_B. At the receiver side, missing feature values can either be set to zero (i.e., no concealment), or they can be linearly interpolated from the nearest available neighboring packets in the same feature channel. A summary of simulator options is given in Table I.

The user sets the above parameters and specifies the path to the test data and ground truth. The simulator then runs a Monte Carlo simulation over channel realizations for all test data and provides the average and standard deviation of the accuracy metric. The number of runs through all test data can be specified by the user. The simulator will be made publicly available on GitHub[1] in August 2018.

This work was supported in part by the Mitacs Globalink Fellowship and the NSERC Grant RGPIN-2016-04590

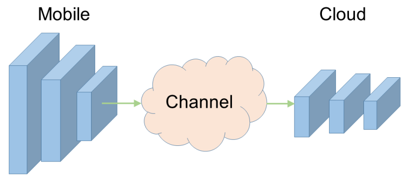

Fig. 1. Deep feature transmission

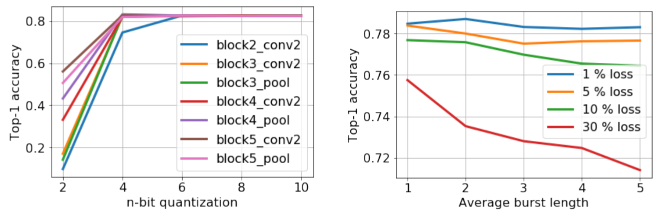

Fig. 2. Sample simulation results on VGG16

Fig. 2 shows sample simulation results on VGG16. The left plot shows Top-1 classification accuracy for one class ("German shepherd") on 88 images from ImageNet, when features produced by various layers are quantized. As seen, the deeper we go, the less sensitive the features are to quantization. The right plot shows the Top-1 accuracy of the same model on 882 images of 10 classes for transmission over a Gilbert channel with various P_B and L_B, and with 8 rows of features from `block3_pool` per packet. Increase in either P_B or L_B negatively impacts the accuracy, although the reduction is not very large over the range of parameters tested.

References

[1] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2017, pp. 615–629.
[2] H. Choi and I. V. Bajić, "Deep feature compression for collaborative object detection," in *Proc. IEEE ICIP'18*, to appear.

[1]https://github.com/SFU-Multimedia-Lab/DFTS

TABLE I
A SUMMARY OF SIMULATOR FEATURES

Component	Options
Model	Any Keras model
Split	Any layer
Quantization	None or uniform n-bit
Packetization	# feature rows per packet
Channel	Independent or Gilbert
Concealment	None or linear