Learned architectures (models)

ljpeg.py

This file contains a learned JPEG variant based on a differentiable JPEG substitute in tensorflow. The quantization tables used in the JPEG pipeline are learned through backpropagation.

ljpeg_attention.py

This file contains the code of out proposed solution: Additionally to the code in ljpeg.py, an attention network is used to employ spatially varying smoothing to the image.

ljpeg_PIL.py

This is a version of our pipeline that is used for evaluation of the standard JPEG performance. Hence, we use the standard IJG quantization tables instead of learned ones.

google_baseline.py

This file contains the implementation that follows Talebi et al., 2020 closely, however it only uses MSE and an estimated BPP in the training loss.

Parameters

The following parameters can be specified for training:

- --checkpoint_dir: Directory where to store (and load) checkpoints. Training can be resumed from previous checkpoints.
- --train_glob: Glob pattern that refers to the list of training images.
- --batchsize: Batch size of training crops
- --patchsize: size of square patches used for training that are obtained by random cropping
- --lambda: Rate-distortion tradeoff parameter
- --last_step: Total number of training steps
- --lr: learning rate used for the Adam optimizer
- --lr_decay: Factor for reducing learning rate after 10000 steps, 1 if no decay intended

Only ljpeg and ljpeg_attention:

- --lpips_weight: Weight on LPIPS term in distortion loss
- --total_weight_regularizer: weight on L1 loss of quantization tables

Run Training

Training can be launched by:

python3 ljpeg_attention.py train --train_glob <my_train_set/*.png\>

Utils

image_utils.py

Contains image operations such as the DCT and IDCT that are shared among different models.

arg_parser.py

Contains the argument parser for the models.

conv_blocks.py

Contains convolutional blocks for editing the image, namely the feedforward ResConvLayer used by Talebi et al. and our VGGAttention network.

Batch Job Training

batch_job_attention.sh

Example script that trains an ljpeg_attention architecture for several rate-distortion tradeoffs and stores the checkpoints in a folder structure that is compatible with the evaluation script below. It takes one argument to set the target directory of the training files, for example:

/bin/bash batch_job_attention.sh <my_experiments_folder/run_1\>

Inside the batch script, the hyperparameters can be set and are then applied to each training session.

Evaluation

evaluate.py

This evaluation script evaluates multiple experiments at once. In particular the expected folder structure is:

<my_experiments_folder\>

- <run_1\>
  - lambda_<01\>
  - train
  - lambda_<005\>
  - train
  - …
- <run_2\>
  - …

where the train folders contain the checkpoints and the values in brackets \<> may change.
It is important to understand that each folder "lambda_<…\>" will correspond to one dot on the final plots and each run will be drawn in the same color. The script ignores any run folder that starts with an underscore _. The results for the JPEG Pillow pipeline will be putt into the "_output_PIL" folder. The results from the experiments will go into the "_output" folder. The script creates pickle files that allows quick re-evaluation. These files store the metrics computed by the evaluation. For recomputing the metrics, the pickle files need to be deleted or moved to an archive folder.

Parameters

The evaluation script takes the following parameters:

- --batch_dir: directory where all runs are stored, for example <my_experiments_folder\> above
- --google_dir: optional parameter to compare against the google_baseline, path should point to directory that contains the checkpoints
- --jpeg_type: Choose the learned architecture type to be evaluated, possible values : [ljpeg, ljpeg_attention]
- --test_image_glob: test images, for example "KODAK/*.png"

A full command may look like this:

python3 evaluate.py --batch_dir attention_experiment --jpeg_type ljpeg_attention --test_image_glob "KODAK/*.png" --google_dir google_baseline_lr_1e-5/lambda_005/train

Requirements

The required packages to run the models are:

- tensorflow 1.15
- tensorflow-compression
- lpips-tensorflow (from github source)
- Pillow
- numpy

Provided trained models

In the folder example_experiments we provide 3 trained experiments for different LPIPS weight parameters:

- attention_learned_q_w10_scale1e-5
- attention_learned_q_w10_scale1e-5_lpips500
- attention_learned_q_w10_scale1e-5_lpips5000

that are also shown in the thesis.
Additionally, we provide our best run of the google_baseline architecture:

- google_baseline_lr_1e-5

  as it is also shown in the thesis. The evaluation results can be found in the _output folder for all models.