Image compression using neural auto-encoder and quantization

(https://opensource.org/licenses/MIT)

This project is a simple implementation of auto-encoder neural network for image compression.
The auto-encoder neural network is trained on the ImageNet dataset. The trained model is then used to compress and
decompress the images.

Navigation:

Model architecture

Model represents a variational auto-encoder with residual blocks and skip connections.

- Encoder: *ResNet-18 architecture with fully connected layers*
- Decoder: *ResNet-18 architecture with transposed convolution layers*
- Loss: *VGG loss + MSE loss*

Download pretrained models

Models were trained
on 130k Images (512x512) - Universal Image Embeddings (https://www.kaggle.com/datasets/rhtsingh/130k-images-512x512-universal-image-embeddings)
dataset from Kaggle.

Here are the links to download the pretrained models:
B = number of quantization levels

- B=2, resnet18 (https://drive.google.com/drive/folders/1FaeWzeRW3BMqqZwGsHUjhf7PuAOsiY6E?usp=sharing)
- B=8, resnet18 (https://drive.google.com/drive/folders/1fYDc0e43cUR7xsIYatpz8fdJ_6KMJmSs?usp=sharing)

Put downloaded models in models directory.

Quick start

compress_all.sh (scripts/compress_all.sh) compresses all images from assets/images directory and saves them
in assets/compressed directory.

compress_all.sh takes 3 arguments:

- qb - number of quantization levels
- resnet-model - resnet model architecture
- device - torch device to evaluate on

```
# Compress all images from assets/images directory
bash scripts/compress_all.sh 8 resnet18 cpu
```

decompress_all.sh (./scripts/decompress_all.sh) decompresses all images from assets/compressed directory and saves
them in assets/decompressed directory.

decompress_all.sh takes 3 arguments:

- qb - number of quantization levels
- resnet-model - resnet model architecture
- device - torch device to evaluate on

```
# Decompress all images from assets/compressed directory
bash scripts/decompress_all.sh 8 resnet18 cpu
```

Compression

```
# Compress the `baboon` image from assets/images directory
python compress.py \
  --image=assets/images/baboon.png \
  --output=assets/compressed/baboon.bin \
  --models-dir=models \
  --resnet-model=resnet18 \
  --qb=8 \
  --device=cuda
```

Decompression

```
# Decompress the compressed image
python decompress.py \
  --file=assets/compressed/baboon.bin \
  --output=assets/decompressed/baboon.png \
  --qb=8 \
  --resnet-model=resnet18 \
  --models-dir=models \
  --device=cuda
```

Training from scratch

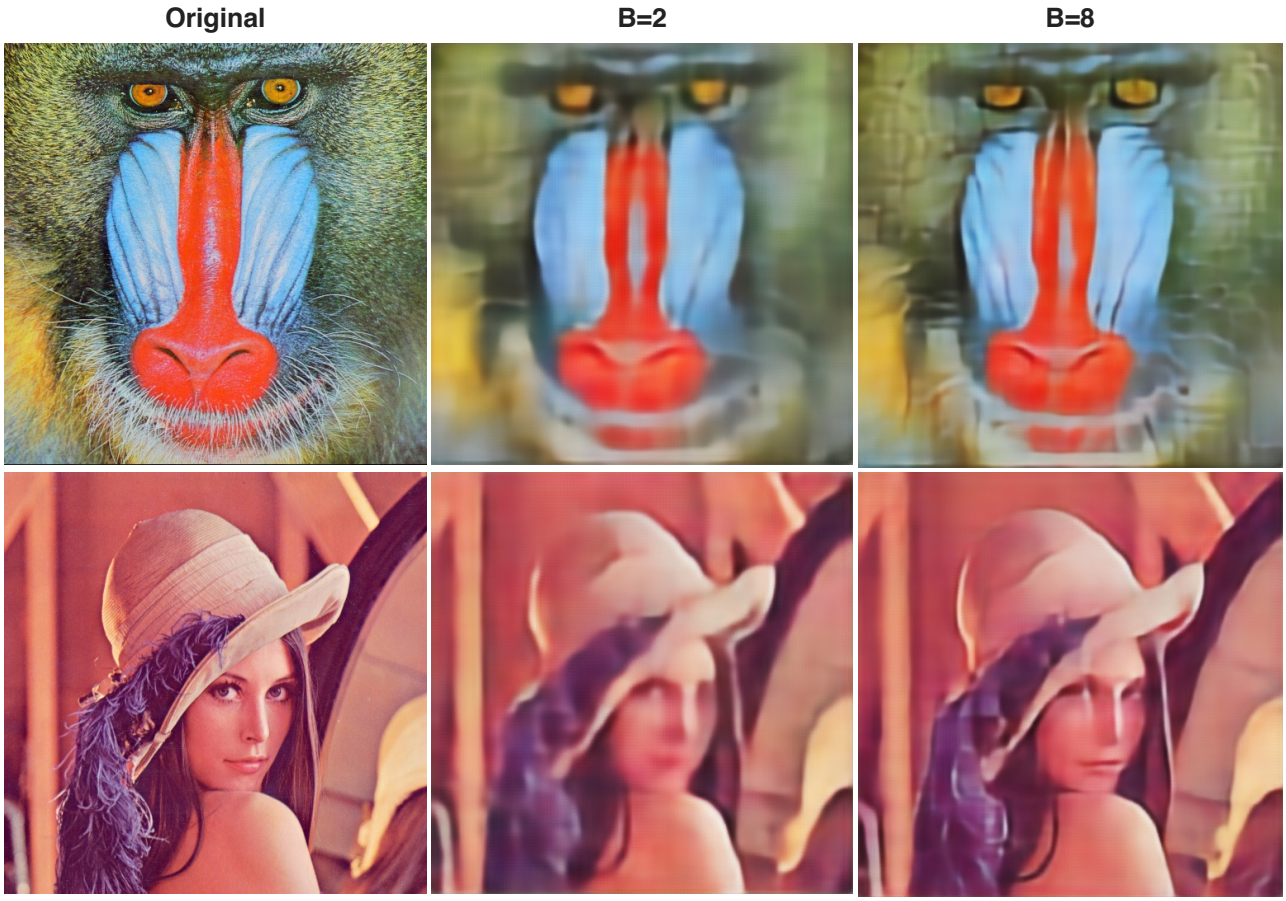
```
python train.py \
  --root [path to images] \
  --test-root [path to test images] \
  --resnet-model [resnet model architecture] \
  --qb [number of quantization levels] \
  --epochs [number of epochs] \
  --batch-size [batch size] \
  --lr [learning rate] \
  --device [torch device to train on] \
  --save-results-every [save results every n epochs] \
  --save-models-dir [path to save models] \
  --use-checkpoint [use checkpoint to resume training]
```

Results

Size comparison

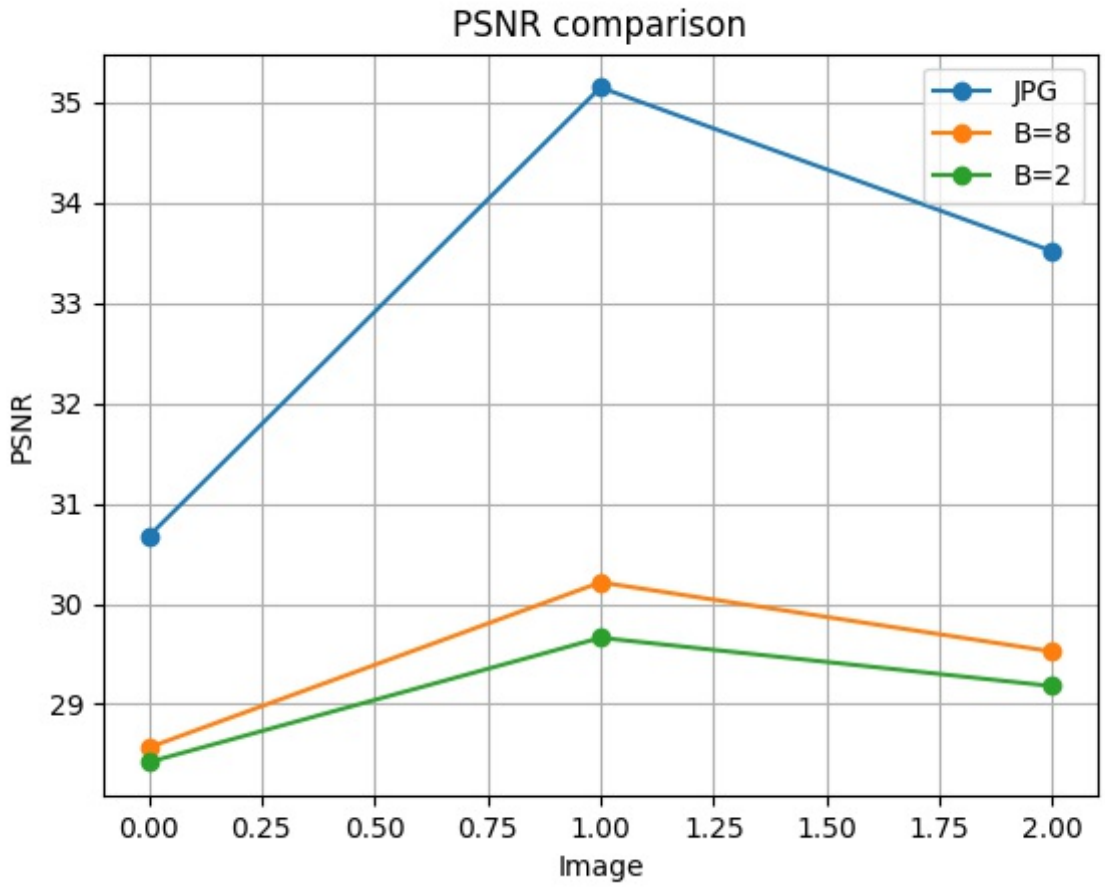
Image	Original (.png)	Jpeg	B=2	B=8
baboon (assets/images/baboon.png)	624 KB	100 KB	20 KB	76 KB
lena (assets/images/lena.png)	504 KB	62 KB	20 KB	76 KB
peppers (assets/images/peppers.png)	528 KB	56 KB	20 KB	76 KB

Images

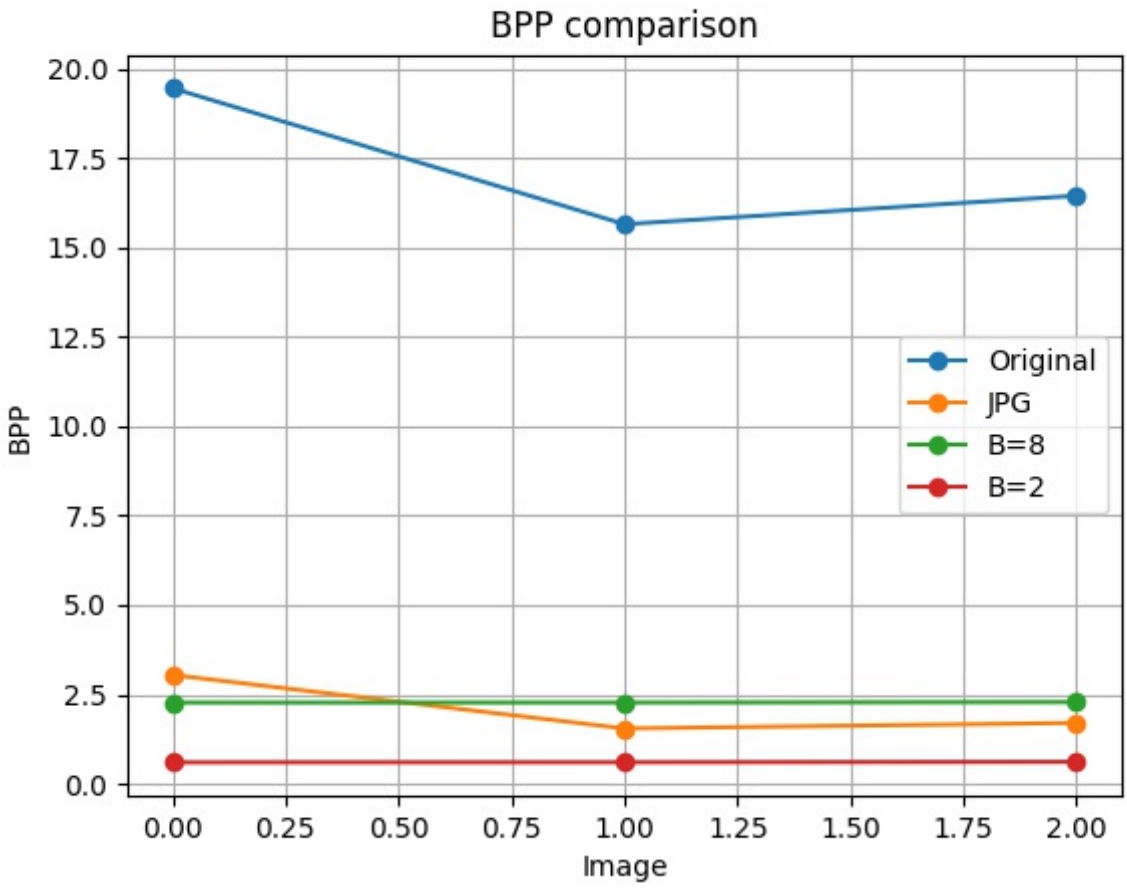


Graphs

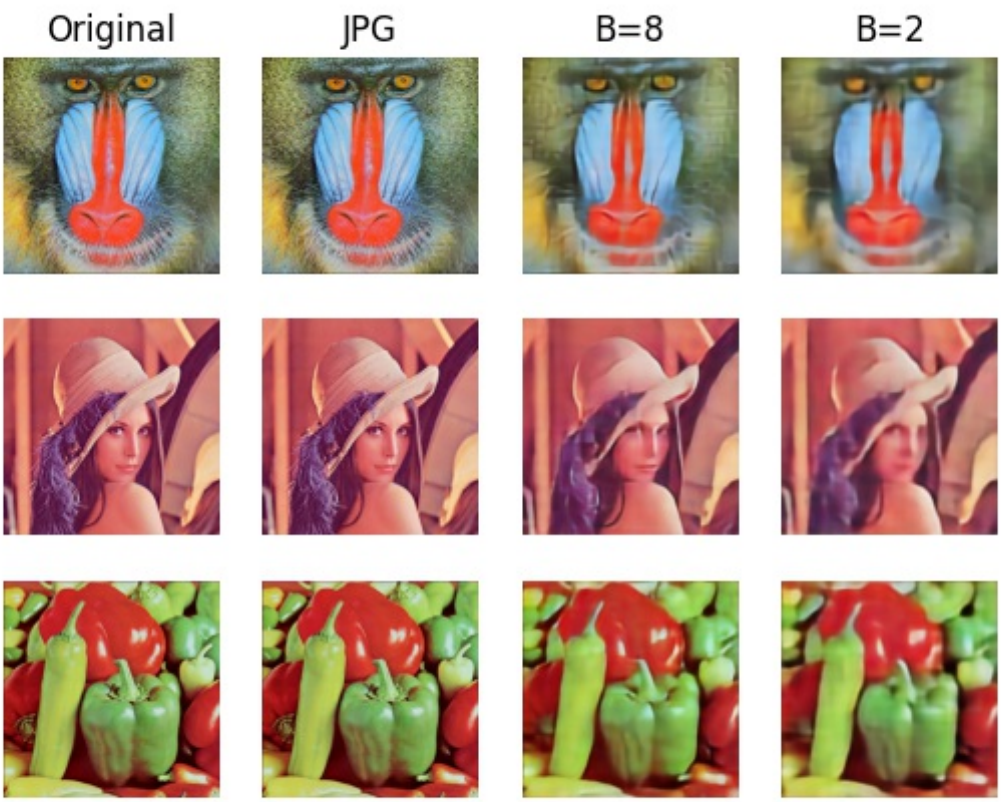
PSNR: ***Peak signal-to-noise ratio***



BPP: ***Bits per pixel***



Quality comparison:



Notebooks

- Kaggle training notebook (notebooks/kaggle-cuda-training.ipynb)
- Analysis notebook (notebooks/analysis.ipynb)