**Convolutional neural network for image compression**

Some recent works are summarized here:

<https://github.com/flyywh/Image-compression-and-video-coding>

The general flow is as follows:

1. Preprocess
2. Convolutional neural network encodes images to low dimensional representation
3. Quantize or binarize the floating-point representation
4. Further compress the code by entropy coding
5. Convolutional neural network decodes low dimensional representation to original images.
6. Train the network with approximation of rate-distortion cost function
7. Evaluate reconstruction performance based on metric like PSNR, SSIM, or MS-SSIM and plot rate-distortion curve

**Dataset**

Many datasets are used by different works for training including:

ImageNet, MSCOCO, UCID, BSD, and other high-resolution images available online (Flickr)

A standard test set is the KODAK dataset which contains 24 non-compressed images.

**Preprocess**

Scale [0,255] 8-bit image into [0,1]. Some works also perform standardization. Many works only focus on single channel and build network for each channel respectively. Split image into patches or randomly crop images to square with size roughly 32x32, 128x128

**Convolutional neural network encodes images to low dimensional representation**

Use Conv2d(kernel\_size = 3, stride=1, padding=1) to perform regular convolutional mapping. Use Conv2d(kernel\_size = 2, stride=2, padding=0) or other values for kernel\_size, stride and padding to down sample the feature maps to half of its size. Add activation function like ReLU, PReLU after each convolution. Dropout and BatchNorm2d degrade the performance.

**Quantize or binarize the floating-point representation**

The code should be either quantized or binarized to save the space. The gradient of quantization and binarization is set to be identical mapping. They tried other random approximation but identity gradient works the best in practice.

**Further compress the code by entropy coding**

Standard entropy coding used in JPEG2000 and other well-known techniques are necessary to further compress the code. Some works heavily rely on this step to perform well.

**Convolutional neural network decodes low dimensional representation to original images.**

Use Conv2d(kernel\_size = 3, stride=1, padding=1) to perform regular convolutional mapping.

Use ConvTranspose2d(kernel\_size = 2, stride=2, padding=0) or kernel\_size, stride and padding, use subpixel convolution (First use Conv2d to generate more feature maps and the shuffle the feature maps among different channels), and interpolation to up sample the low dimensional representation to original images

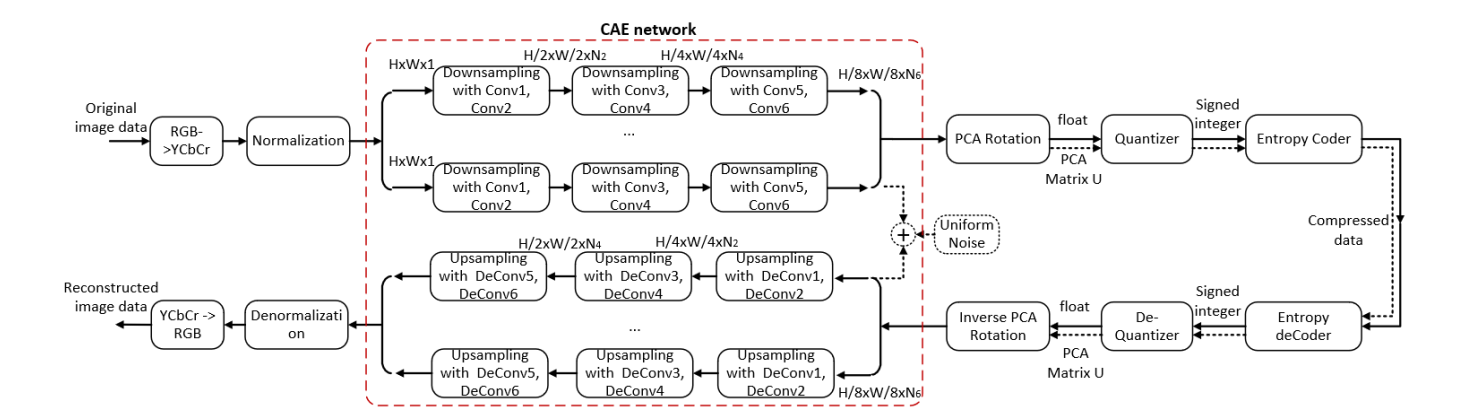
**Train the network with approximation of rate-distortion cost function**

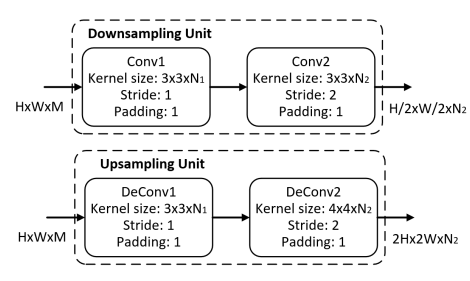
Most works propose use L1 Penalty to sparsify the codes in order to adjust the compression rate since redundant zeros can be further decompressed with entropy coding. For different compression rate, we do need to train different networks. There do exists some works try to train one network with variable compression rate.

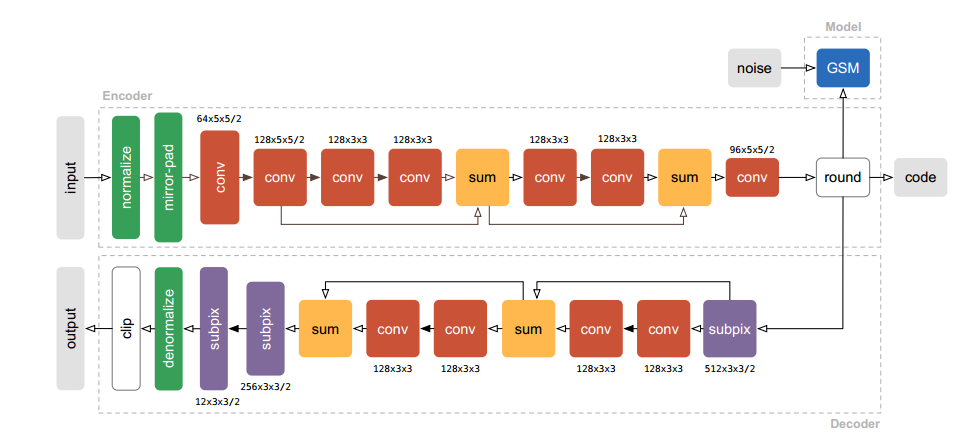
**Evaluate reconstruction performance based on metric like PSNR, SSIM, or MS-SSIM and plot rate-distortion curve**

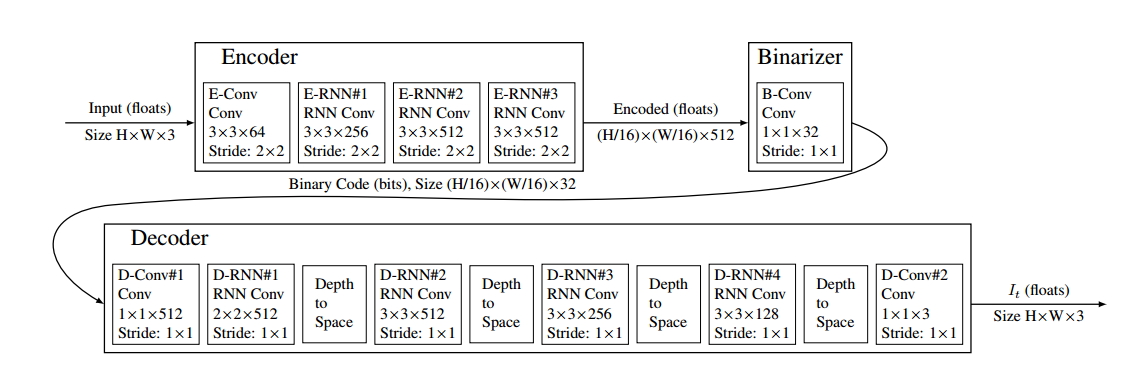
PSNR(Peak SNR), SSIM(Structural Similarity Index) or MS-SSIM(Multi-scale Structural Similarity Index) are widely used metric to evaluate the performance of reconstruction. The Area under the curve of rate-distortion curve is used to visually compare different works.

**Some example models I have tried**

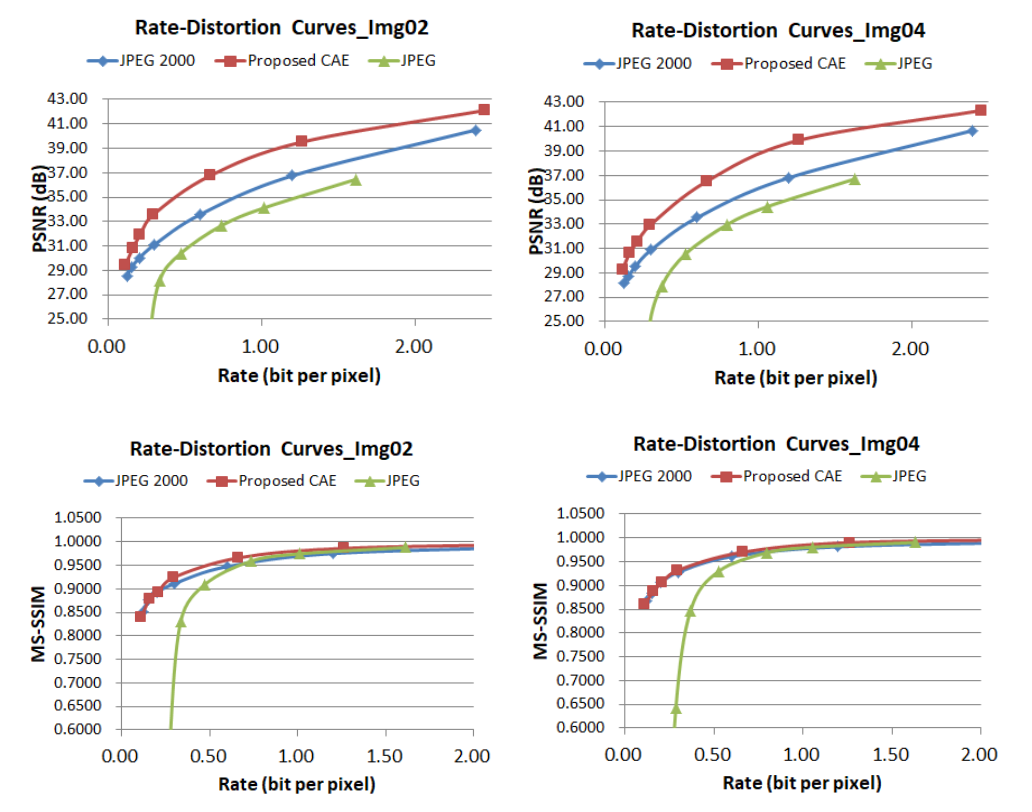


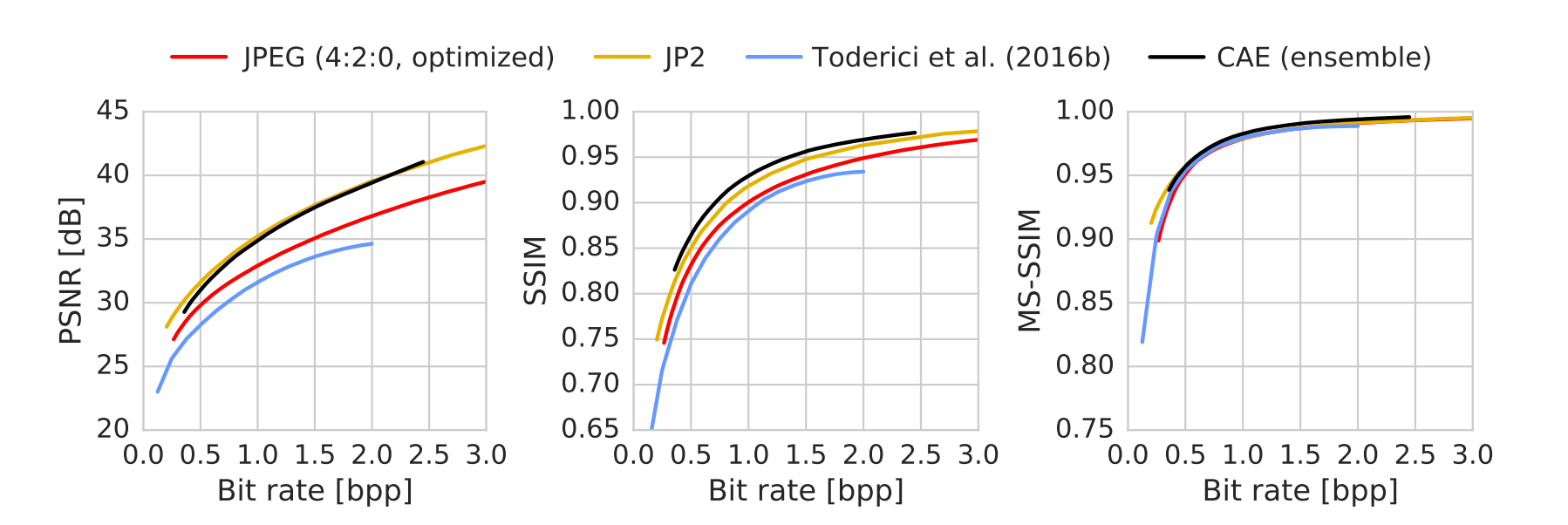






**Some results their papers claim**





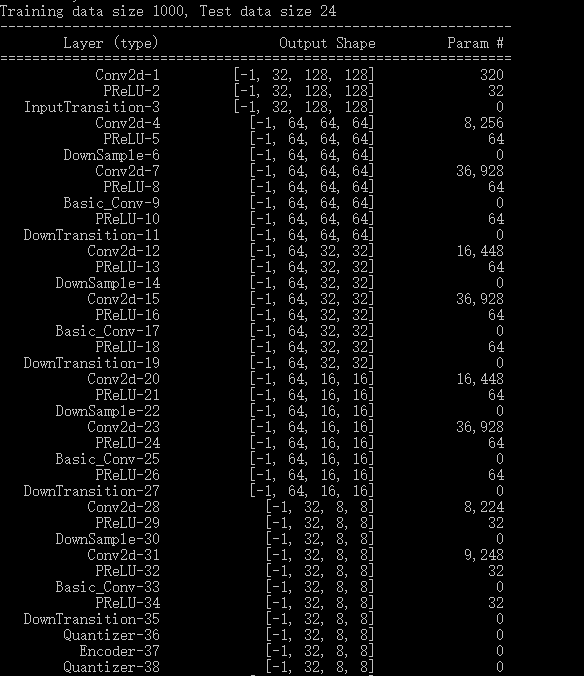
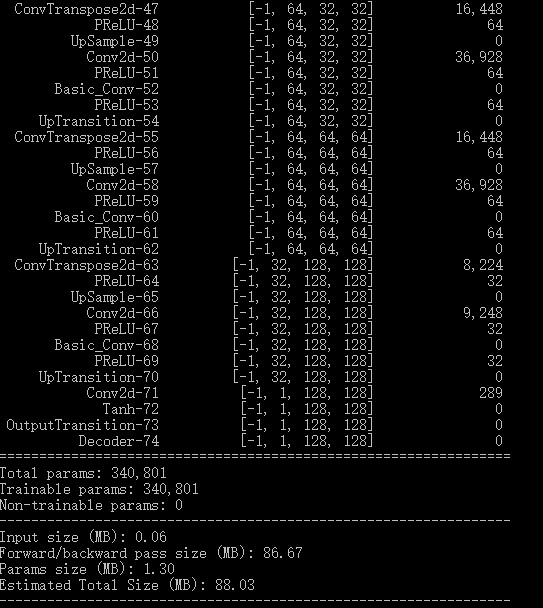
**Experiment Experience**

Many networks require a small learning rate to train, typically requires more than 100 epochs. Some networks actually use a lot number of data (more than 1 million) to train. I only attempt their model on at most 100k usually 10k data and very few epochs to see whether they perform well or not. I can hardly achieve their results. The reason could be they actually train 200 epochs with carefully tuned learning rate. Some works may also report the experiment easier than they actually conduct. For example, they might use more data or more sophisticated learning rate tuning but they never report in their paper. In fact, this image compression sub branch in Deep Learning is rather minor and not much attention has been paid to this topic. In fact, I only find one work that has source code published and that network seems the most convincing. However, their model also needs heavy training and tuning. Due to computation complexity and all the other issues mentioned above, it is hard to conduct the whole experiment that achieve good result.

**Preliminary Results**

I can achieve following result with my own computer (one GTX 1080Ti):

1. For single channel (R channel in RGB)
2. 1000 images with around 10k 128x128 patches ((1,128,128) compressed to (32,8,8), 8 bpp to 1 bpp (bit per pixel) (Channel, Height, Width)
3. Model Summary:

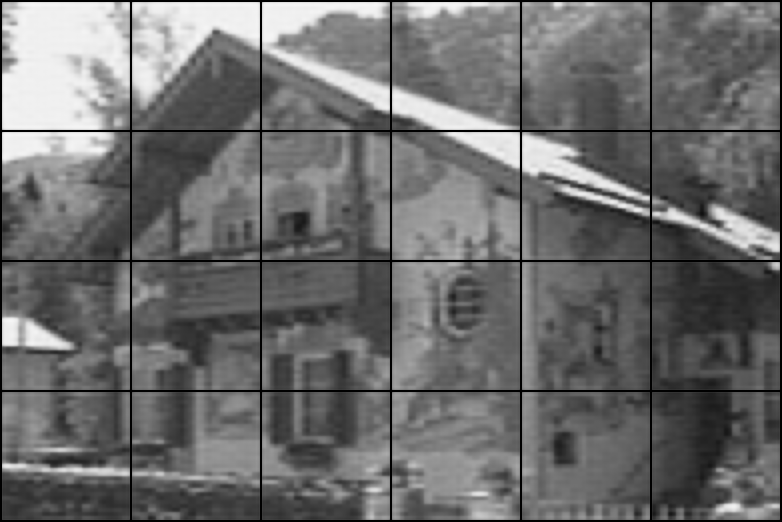


1. Train 50 Epochs (each epoch takes around 70 secs), and Test PSNR: 24dB.
2. Use similar setup but compress to (32,16,16) the test PSNR can achieve 25dB
3. Example reconstructed images:

Original



To (32,8,8)



To (32,16,16)



**Next Step**

*AI for Compression Milestone*

Since now we are able to train a basic codec with pure convolutional neural network, we can move forward from compressing images to detect shot boundary changes in videos. We can use Convolutional LSTM to train a change detection network with the parameters of the convolution cell initialized from the trained model which I have built above. Similar to its usage in NLP, convolutional LSTM is suitable for video data since the spatial structure of images are exploited. There also exists a Google Paper in compressing images that provides source code and trained models that we can use immediately. Their work binarizes the code and it is able to adjust the compression rate with one trained network. We are also looking forward to use this result as a cornerstone of short boundary detection in videos.

*Compression for AI Milestone*

We are also interested in finding a mapping between different types of codecs. Is it possible to train a neural network to find the mapping between lossy compression codes(jpeg) and lossless compression codes(png)?

We will try to use feed-forward fully connected neural network to train a mapping between various kinds of codecs. Then we will try to find the mapping between traditional codes(jpeg) and neural-network-based codes.