An End-to-End Compression Framework Based on Convolutional Neural Networks

Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, Debin Zhao, Member, IEEE
Traditional image compression algorithms

• JPEG attempt to distribute available bits

• Deblocking
New solution

- The first CNN, named ComCNN, learns an optimal compact representation from an input image, which is then encoded using an image codec (e.g., JPEG, JPEG2000 or BPG).

- The second CNN, named RecCNN, is used to reconstruct the decoded image with high quality in the decoding end.
Architecture of compression framework

comCNN:

- combination of conv and ReLU
- 1\textsuperscript{st} layer: 64 filters of size 3 x 3 x c with ReLU
- 2\textsuperscript{nd} layer: 64 filters of size 3 x 3 x 64 with ReLU, stride = 2.
- 3\textsuperscript{rd} layer: c filters of size 3 x 3 x 64
Architecture of compression framework

• RecCNN:

- 20 layers
- **Layer 1**: 64 filters of $3 \times 3 \times c$ (Convolution + ReLU)
- **Layer 2 to 19**: 64 filters of size $3 \times 3 \times 64$ (Convolution + batch normalization + ReLU)
- **Layer 20**: $c$ filters of size $3 \times 3 \times 64$
- The compressed image is unsampled to the original image size using bicubic interpolation
Learning Algorithm

- Optimization goal is to find –

\[
\left( \hat{\theta}_1, \hat{\theta}_2 \right) = \underset{\theta_1, \theta_2}{\arg\min} \left\| \text{Re} \left( \theta_2, \text{Co} \left( C_r \left( \theta_1, x \right) \right) \right) - x \right\|^2
\]

**Algorithm 1** The Proposed Compression Framework for Training Sub-Networks

1: **Input:** The original image \( x \)
2: **Initialization:** Random initial \( \hat{\theta}_0 \) and \( \hat{\theta}_0 \)
3: for \( t = 1 \rightarrow T \) do
4: Update \( \hat{x}^t_m \) by computing Eq.(2)
5: for \( x_m = \hat{x}^t_m \) do
6: Update \( \hat{\theta}_2 \) by training RecCNN to compute Eq.(5)
7: end for
8: for \( \hat{\theta}_2 = \hat{\theta}_2 \) do
9: Update \( \hat{\theta}_1 \) by training ComCNN to compute Eq.(13)
10: end for
11: end for
12: **Return:** \( \hat{\theta}_1 \), \( \hat{\theta}_2 \) and \( \hat{x}^t_m \)

\[
\hat{\theta}_1 = \underset{\theta_1}{\arg\min} \left\| \text{Re} \left( \hat{\theta}_2, \text{Co} \left( C_r \left( \theta_1, x \right) \right) \right) - x \right\|^2, \quad (2)
\]

\[
\hat{\theta}_2 = \underset{\theta_2}{\arg\min} \left\| \text{Re} \left( \theta_2, \hat{x}_m \right) - x \right\|^2. \quad (5)
\]

\[
\hat{\theta}_1 = \underset{\theta_1}{\arg\min} \left\| \text{Re} \left( \hat{\theta}_2, C_r \left( \theta_1, x \right) \right) - x \right\|^2. \quad (13)
\]
Loss functions (MSE)

• For comCNN:

\[ L_1(\theta_1) = \frac{1}{2N} \sum_{k=1}^{N} \left\| Re\left( \hat{\theta}_2, Cr\left( \theta_1, x_k \right) \right) - x_k \right\|^2 \]

• For recCNN:

\[ L_2(\theta_2) = \frac{1}{2N} \sum_{k=1}^{N} \left\| res\left( Co\left( \hat{x}_{m_k} \right), \theta_2 \right) - (Co(\hat{x}_{m_k}) - x_k) \right\|^2 \]
Reading

• https://en.wikipedia.org/wiki/Discrete_cosine_transform