Image Compression with Autoencoder

Replication of work in ***Deep Convolutional AutoEncoder-based Lossy Image Compression***

Tian Qiu, Jiawen Zhang
ECE, UC San Diego

Z. Cheng, H. Sun, M. Takeuchi and J. Katto, "Deep Convolutional AutoEncoder-based Lossy Image Compression," *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, 2018, pp. 253-257, doi: 10.1109/PCS.2018.8456308.

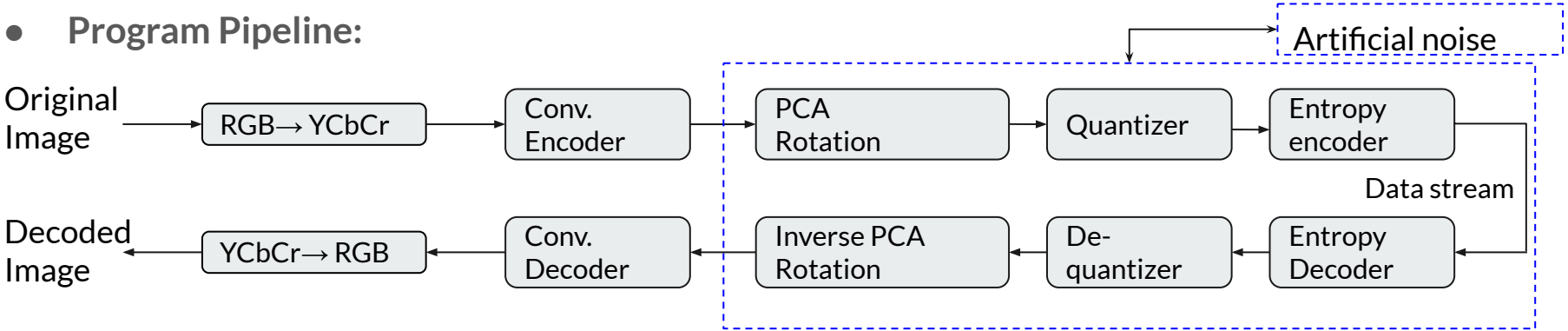
Background and motivation

- Developing image compression standards: time-consuming
- Deep-learning models can capture complex image features
- Researchers are starting to explore **image compression with deep learning**
- Works include deep-learning based tools and end-to-end models
  - Toderici, 2017: Recurrent neural network(RNN)
  - Theis, 2017 : Autoencoder
  - Katto, 2018 : Autoencoder with PCA for higher compression rate (Our reference)
- Remaining challenges: Approximating quantization error, computation complexity, limited generalization ability
- Motivation: Try to replicate a work
  - Learn from doing

Toderici, George, et al. "Full resolution image compression with recurrent neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
Theis, Lucas, et al. "Lossy image compression with compressive autoencoders." *ICLR*. 2017

Overview of reference paper

- **Objective:** Color image compression with convolutional autoencoder(CAE)
- **Main contributions**
  - Replace forward/inverse transform with symmetric CAE.
  - After encoder, use PCA to generate more compact information representation
- **Program Pipeline:**



  - Training stage: Short-circuit the pipeline, add artificial noise between convolutional encoder and decoder to approximate quantization error
  - Application stage: Apply all blocks in series

Overview of reference paper

- **Dataset:**
  - Training/Val set: A subset of ImageNet (5500 images)
    Original images cropped into 128x128 patches
  - Test set: Kodak Lossless Image dataset
    All images in 768x512 size. 24-bit color PNG file.
- **Metrics and main results**
  - Compression performance: PSNR/SSIM
  - Runtime performance

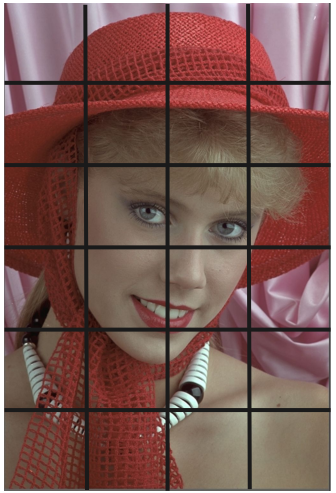
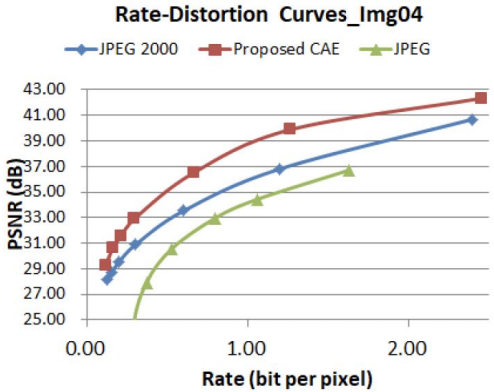
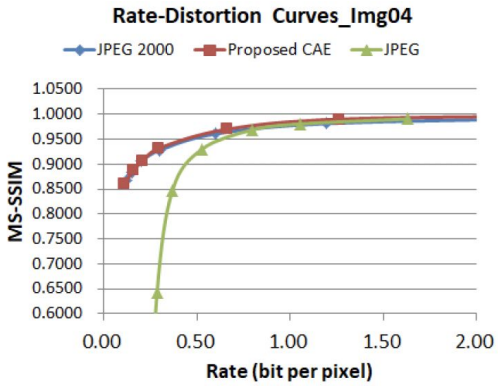


TABLE I: Average running time comparison.

Codec	Time (s)
JPEG	0.39
JPEG2000	0.59
Balle's work[5] with CPU	7.39
Propose CAE with CPU	**2.29**
Propose CAE with GPU	**0.67**

Test dataset: True Color Kodak Images





Project Overview

- **Objective:** Replicate work in Katto paper
- **Result:** Partially finished the model proposed in the paper
- **Main work:**
    - Implemented and trained convolutional autoencoder and decoder
    - Implemented PCA analysis to reduce information redundancy
    - Implemented integer quantizer and dequantizer
    - Analysis of model performance on test dataset
- **Missing part:** Entropy encoding with Adaptive BAC

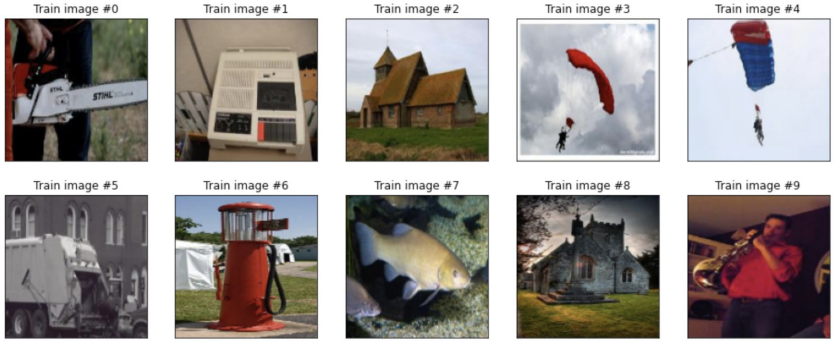
Implementation details: Setups and dataset

- **Programming environment:**
  UCSD datahub server with GPU. Python 3 and TensorFlow 2.1 environment
- **Main packages:**
  Image processing: opencv-python, Pillow
  Data processing: numpy, scikit-learn, scipy
- **Train/Val dataset:**
  Imagenette: A small subset of ImageNet, consisting of ten easily distinguishable classes.
  Total images = 10 classes x 300 pics/class = 3000
  Use 80-20 split → 2400 training images, 600 test images
- **Test dataset:**
  Same as reference(Kodak dataset). 24 images of various topic with rich content.

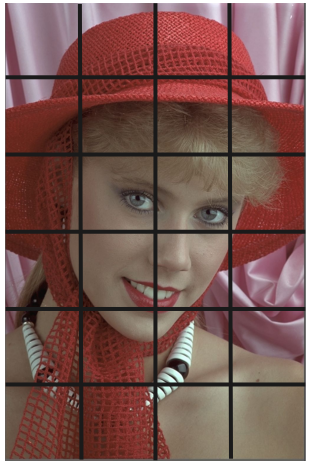
Imagenette: A small subset of ImageNet developed by fast.ai: https://github.com/fastai/imagenette

Implementation details: Image pre-processing

- **Image resizing**
  For train/val sets, resize all images to 128x128.



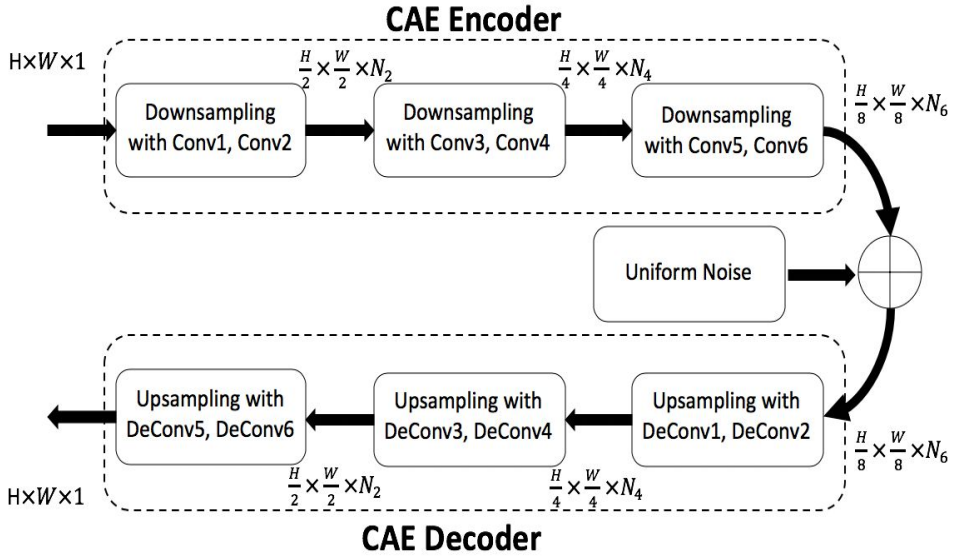
Test image partition:

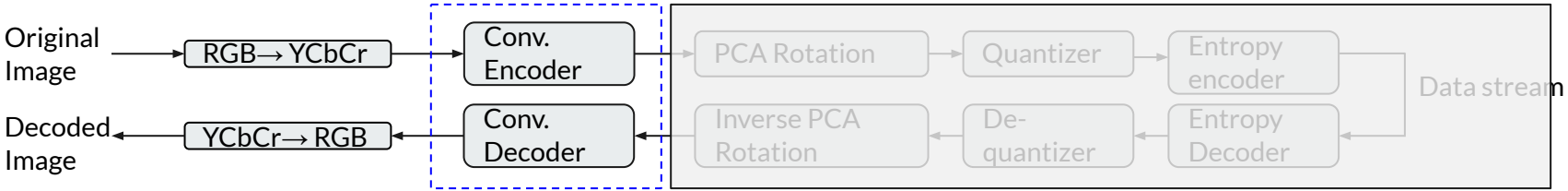


- **Image partition**
  For test set, all images can be partitioned to 128x128 blocks.
  Encode each block individually
- **Color space conversion**
  RGB→ YCbCr with opencv

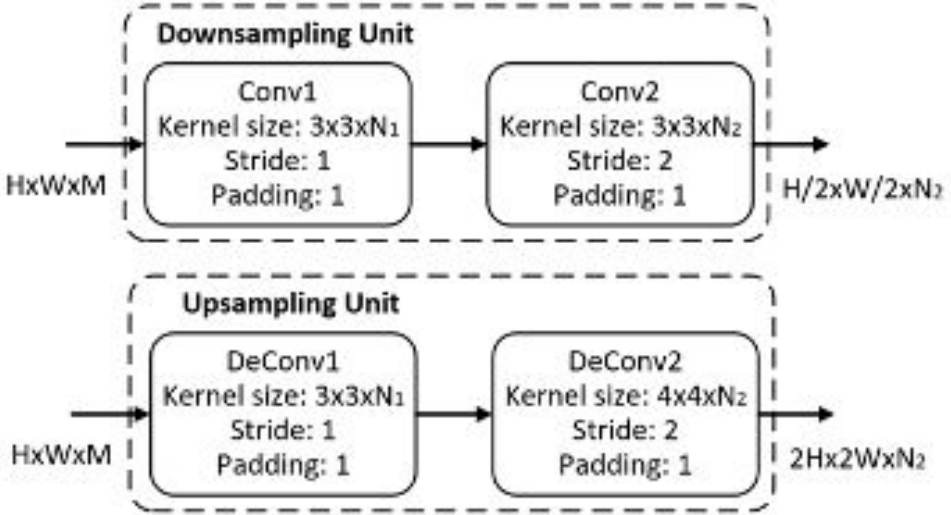
Implementation details: CAE Architecture

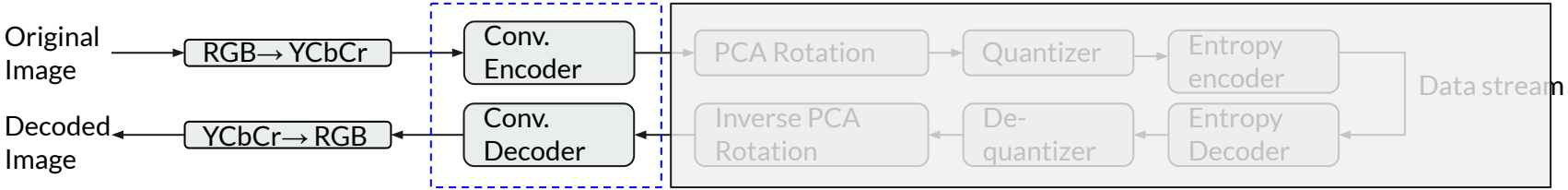
- Symmetric CAE network using convolution and deconvolution filters.
- Input size: 128x128x1.
- Number of filters: 32, 32, 64, 64, 64, 32.
- Add uniform noise ranging from -2^-10 to 2^-10 to replace quantization.





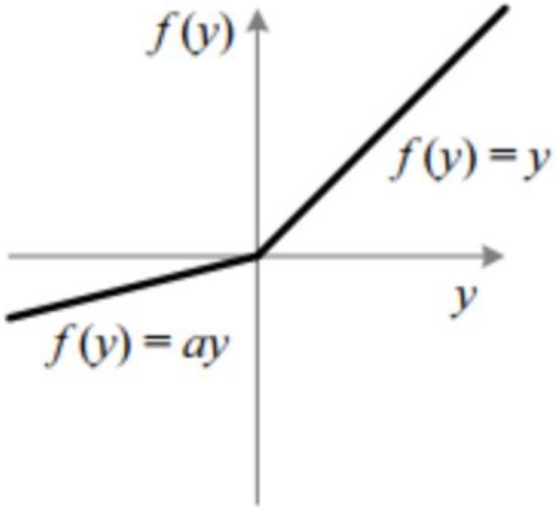
Implementation details: CAE Architecture

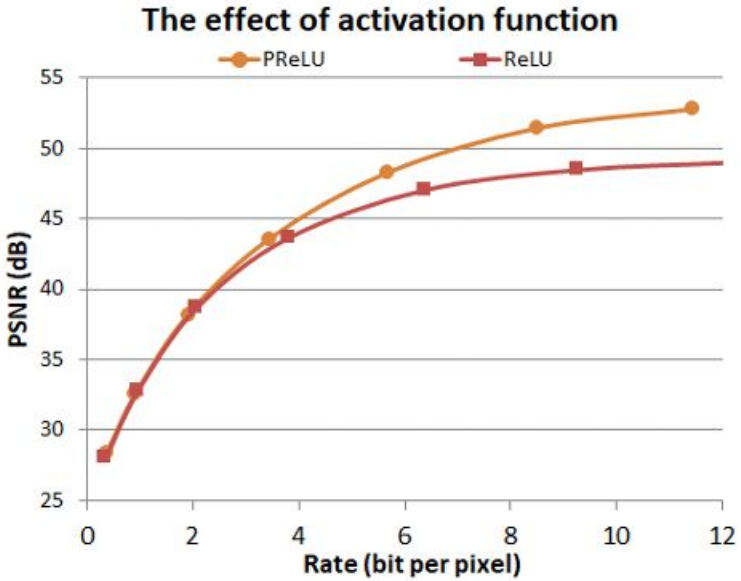


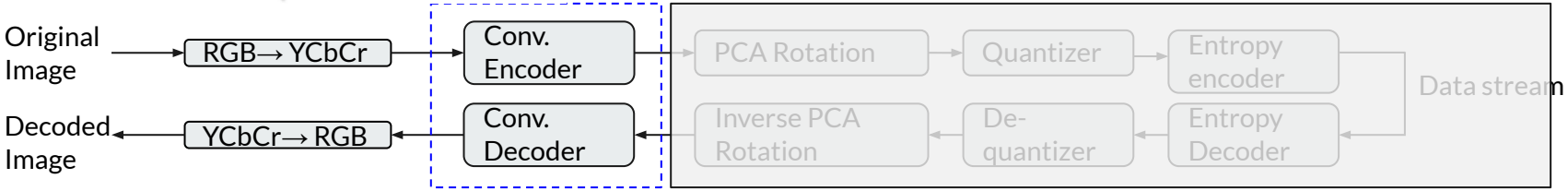


Implementation details: CAE Architecture

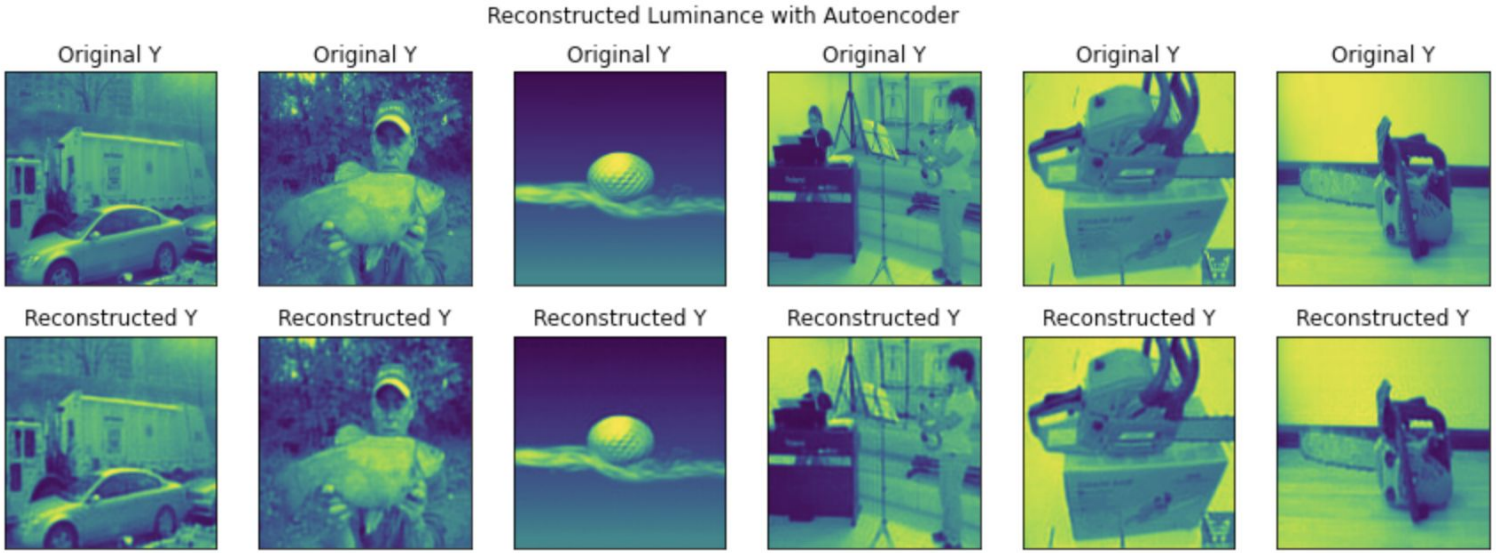
- Activation function: PReLu

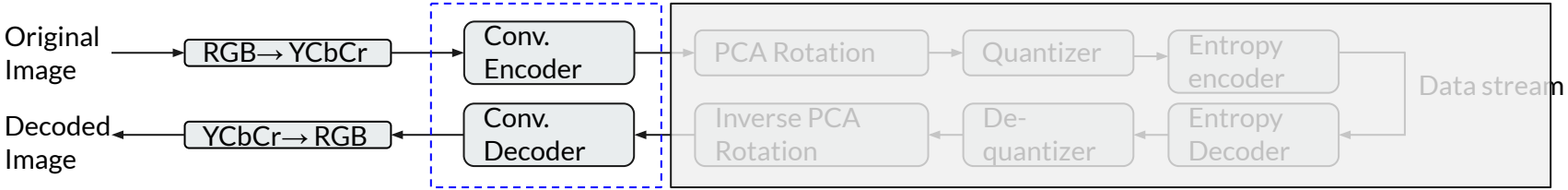




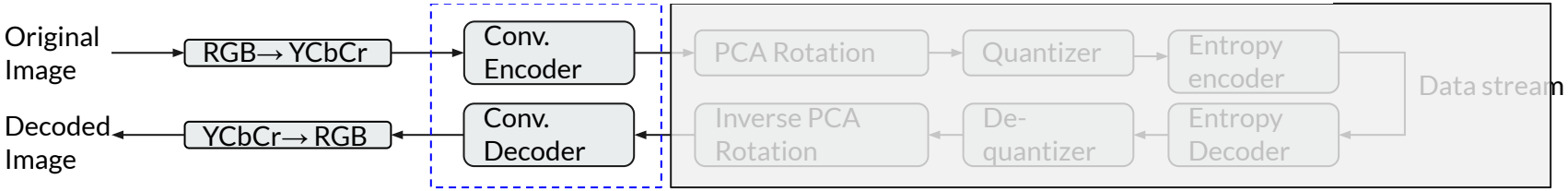


Implementation details: CAE Training





Implementation details: CAE Training



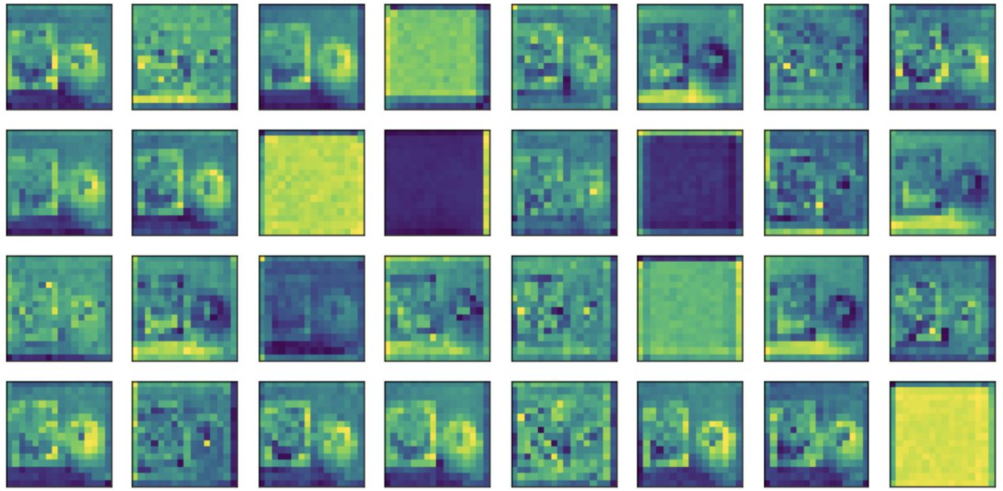
Implementation details: PCA rotation

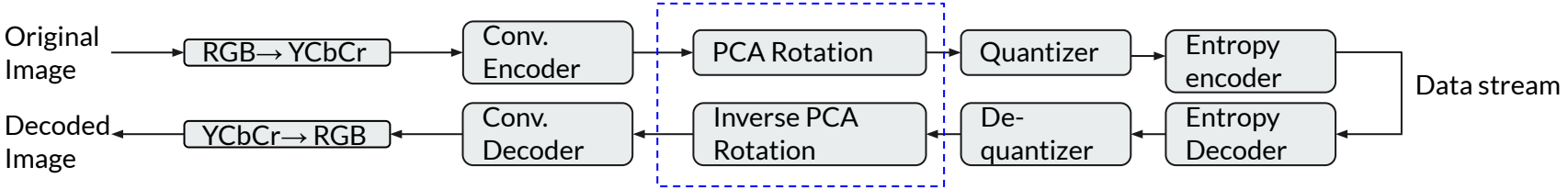
- Input:

  16x16x32 Encoder output

- Output:

  16x16x32 data with a more energy-compact representation

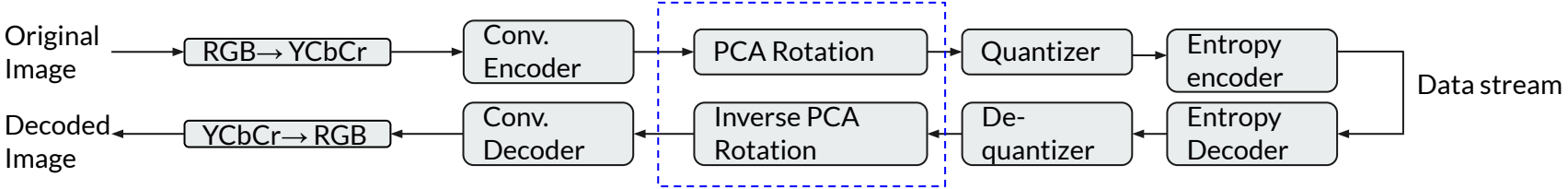




Implementation details: PCA rotation

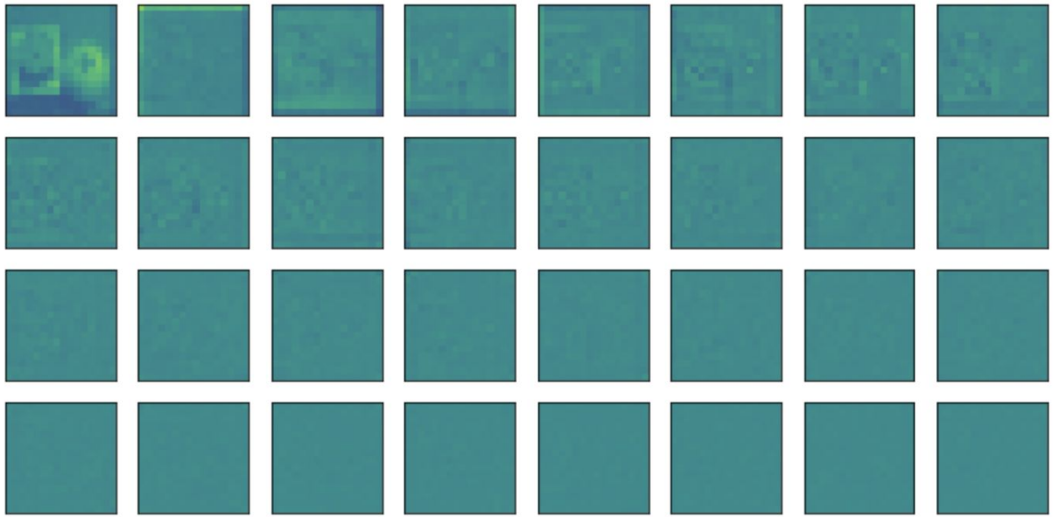
Algorithm

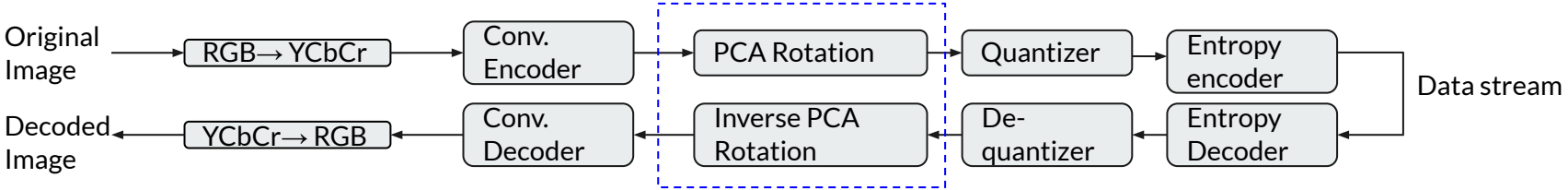
1. Reshape the input data from 16x16x32 to 256x32.
2. Compute the covariance matrix of the reshaped data.
3. Compute the 32 eigenvectors of the covariance matrix.
4. Stack the 32 eigenvectors in columns to form matrix *U*.
5. Do matrix multiplication
6. Reshape the data back to 16x16x32



Implementation details: PCA rotation

- Feature maps are sorted in descending order.
- The first feature map has the most information.

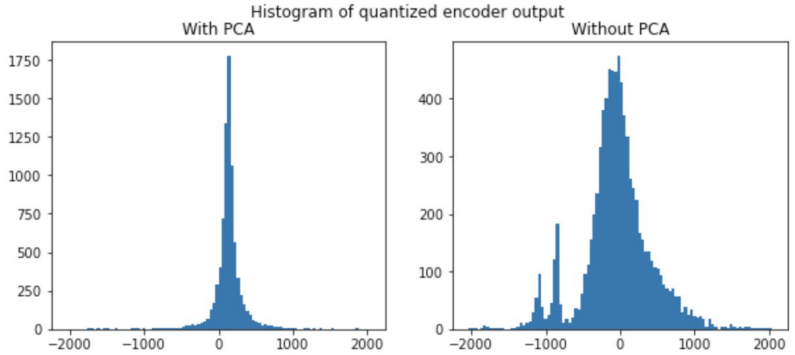


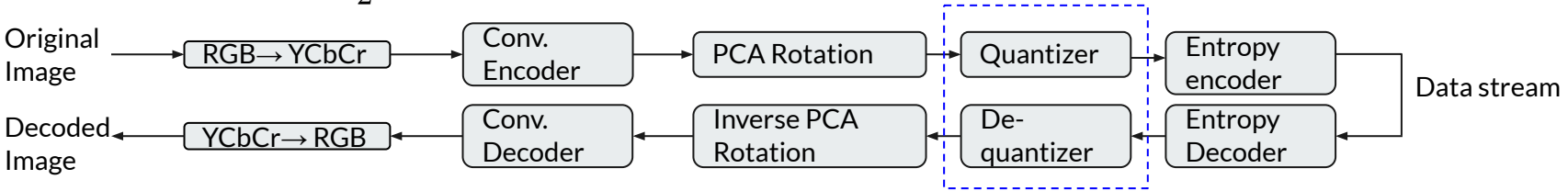


Implementation details: Integer quantizer

- **Input:** Encoder output after PCA rotation.
  Size: [n_samplesx16x16x32] per channel (3 channels)
- **Output:** Quantized integer representation
  of input data. Same size
- **Parameter:** Number of precision = B
  B controls the granularity of uniform quantizer.
  Input data are re-normed and rounded to range [-2^(B-1), 2^(B-1)+1].
  Example: When B= 12, quantization results ranges from -2048 to 2047
- **Formula:** $y' = [y \cdot 2^{B-1}]$

$$\tilde{y} = \frac{y'}{2^{B-1}}$$

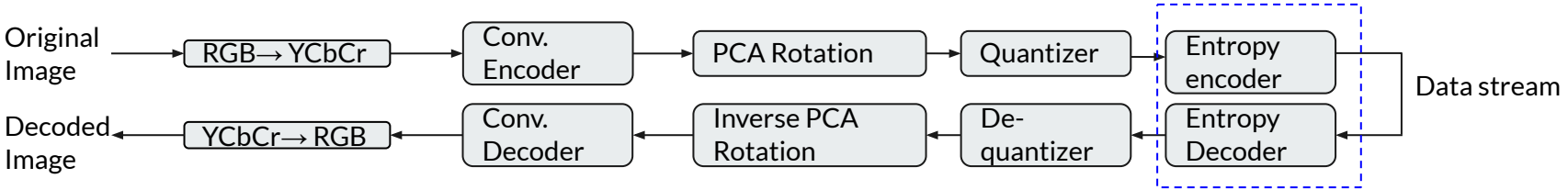




Implementation details: Entropy encoding

- This part is unfinished.
- **Approach in reference paper:** Compress quantized integer with Adaptive Binary Arithmetic Coder(CABAC)
- **Our approach:**
  Do not actually compress. Instead, estimate required bitrate with source entropy.
  H = discrete entropy of quantized integer sequence for one channel (size: 16x16x32)
  Rough estimate of bit per pixel(bpp):

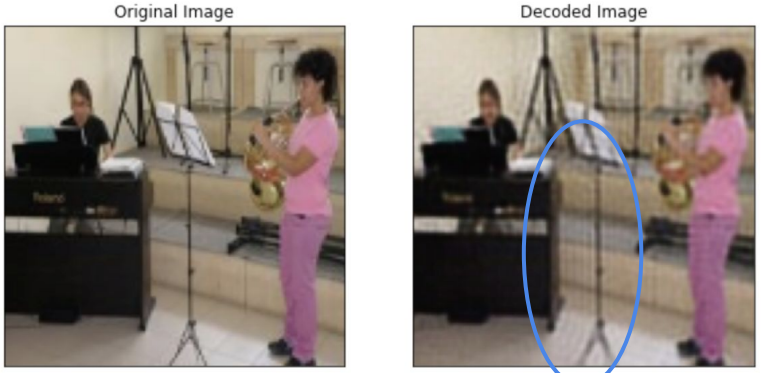
$$\text{bpp} \geq \frac{\text{estimated total encoded bits}}{\text{number of pixels}} = \frac{16 \times 16 \times 32 \times (H_Y + H_{Cr} + H_{Cb})}{128 \times 128}$$



Experiment results: Visual

- **Validation images:**
  Blurry in general
  Obvious ringing/'staircase' artifact

- **Test images:**
  Less obvious artifact
  Random spots



Precision = 12, PSNR = 29.1, SSIM = 0.91

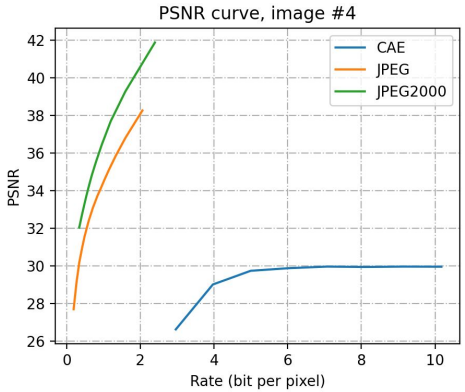
Precision = 12, PSNR = 29.6, SSIM = 0.87

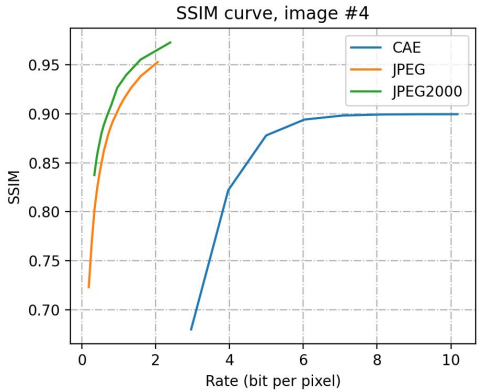
Experiment results: Quality Metrics

- **Rate-distortion curve:**
  PSNR limited under 30
  SSIM limited under 0.9
  Bpp is high

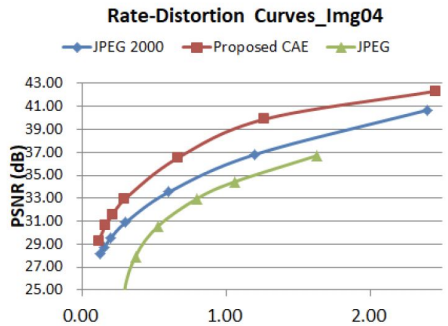
- **Possible reasons**
  - CAE needs better generalization ability to work on large test set.
  - PCA and quantization can be more compact

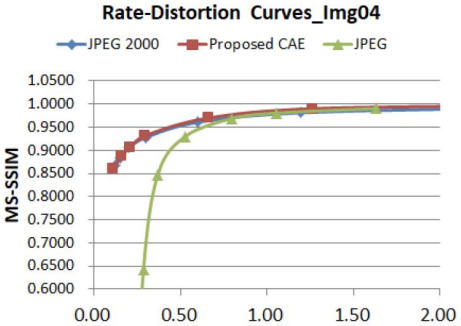
Ours:





Reference:





JPEG implemented with Python Pillow package: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#jpeg
JPEG2000 implemented with Python Glymur package on top of OpenJPEG Library: https://glymur.readthedocs.io/en/latest/

Limitations, future work, conclusion

- Limitations:
  Not fully replicating the paper

- Future work:
  Include entropy encoding/decoding
  Use larger dataset
  Explore more compact structure of autoencoder

- Conclusion:
  Implemented a image compression system with convolutional autoencoder.
  Demonstrated the potential of CAE to compress image.
  Demonstrated the ability of PCA to generate compact information representation.

Notes on project code

- The whole project is in a jupyter notebook
- We used an isolated conda environment for the project
  - A Markdown file ***create_285_env.md*** is included in the submission for reference.
  - A list of required packages is in ***requirements.txt***
  - Use the commands in the Markdown file to recreate the project environment
- We will quickly go through the code after the presentation

Thank you!

- **References:**

Z. Cheng, H. Sun, M. Takeuchi and J. Katto, "Deep Convolutional AutoEncoder-based Lossy Image Compression," *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, 2018, pp. 253-257, doi: 10.1109/PCS.2018.8456308.

Toderici, George, et al. "Full resolution image compression with recurrent neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

Theis, Lucas, et al. "Lossy image compression with compressive autoencoders." *ICLR*. 2017

Uniform noise layer: https://github.com/RitwikGupta/Keras-UniformNoise

Test dataset: True Color Kodak Images

Imagenette: A small subset of ImageNet developed by fast.ai: https://github.com/fastai/imagenette

Python Pillow package: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#jpeg

Python Glymur package: https://glymur.readthedocs.io/en/latest/