The goal of lossless [image compression](https://www.sciencedirect.com/topics/computer-science/image-compression) is to represent an image signal with the smallest possible number of bits without loss of any information, thereby speeding up transmission and minimizing storage requirements. The number of bits representing the signal is typically expressed as an [average bit rate](https://www.sciencedirect.com/topics/engineering/average-bit-rate).\

Each of the file formats are appropriate for different types of images. Choosing the proper format results in higher quality images and smaller file sizes. Choosing the wrong format means your images won’t be as high-quality as they could be and that their file sizes will likely be larger than necessary.

For simple graphics like logos or line drawings, GIF formats often work best. Because of GIF’s limited color palette, graphics with gradients or subtle color shifts often end up posterized. While this can be overcome to some extent by using dithering, it’s often better to use a different file format.

The benefit of [lossy compression](https://www.sciencedirect.com/topics/engineering/lossy-compression" \o "Learn more about Lossy Compression from ScienceDirect's AI-generated Topic Pages) is that it achieve the best possible fidelity given an available communication or storage bit rate capacity or to minimize the number of bits representing the image signal subject to some allowable loss of information. In this way, a much greater reduction in bit rate can be attained as compared to [lossless compression](https://www.sciencedirect.com/topics/engineering/lossless-compression), which is necessary for enabling many [real-time applications](https://www.sciencedirect.com/topics/computer-science/real-time-application) involving the handling and transmission of audiovisual information. The function of compression is often referred to as coding, for short. [Transform encoding](http://en.wikipedia.org/wiki/Transform_coding) is the type of encoding used for JPEG images. In images, transform coding averages out the color in small blocks of the image using a [discrete cosine transform](http://en.wikipedia.org/wiki/Discrete_cosine_transform) (DCT) to create an image that has far fewer colors than the original.

[Chroma subsampling](http://en.wikipedia.org/wiki/Chroma_subsampling) is another type of lossy compression that takes into account that the human eye perceives changes in brightness more sharply than changes of color, and takes advantage of it by dropping or averaging some chroma (color) information while maintaining luma (brightness) information. It’s commonly used in video encoding schemes and in JPEG images

Arithmetic Coding A strategy that almost achieves the lower bound H(˜p) (for long enough symbol streams) is arithmetic coding It encodes the entire stream into a single number a 0 ∈ [0, 1), by subdividing [0, 1) in each step (encoding one symbol) as follows: Let a, b be the bounds of the current step (initialized to a = 0 and b = 1 for the initial interval [0, 1)). We divide the interval [a, b) into |X | sections where the length of the j-th section is p˜(j)/(b − a). Then we pick the interval corresponding to the current symbol, i.e., we update a, b to be the boundaries of this interval. We proceed recursively until no symbols are left. Finally, we transmit a 0 , which is a rounded to the smallest number of bits s.t. a 0 ≥ a. Receiving a 0 together with the knowledge of the number of encoded symbols and p˜ uniquely specifies the stream and allows the receiver to decode.

To encode/decode images with L3C (and other methods outputting a probability distribution), a pass with an entropy coder is needed. We implemented a relatively simple pipeline to encode and decode images with L3C. We did not optimize our code for speed, yet still obtain practical runtimes. We also note that to use other likelihood-based methods for lossless compression, similar steps are required. While our encoding time is in the same order as for classical approaches, our decoder is slower than that of the other approaches. This can be attributed to more optimized code and offloading complexity to the encoder – while in our approach, decoding essentially mirrors encoding. However, combining encoding and decoding time we are either faster (FLIF) or have better bitrate (PNG, WebP, JPEG2000).

We proposed and evaluated a fully parallel hierarchical probabilistic model with auxiliary feature representations. Our L3C model outperforms PNG, JPEG2000 and WebP on all datasets. Furthermore, it significantly outperforms the RGB Shared and RGB baselines which rely on predefined heuristic feature representations, showing that learning the representations is crucial. Additionally, we observed that using PixelCNN-based methods for losslessly compressing full resolution images takes two to five orders of magnitude longer than L3C. To further improve L3C, future work could investigate weak forms of autoregression across pixels and/or dynamic adaptation of the model network to the current image. Moreover, it would be interesting to explore domain-specific applications, e.g., for medical image data.