End-to-end optimized image compression with competition of prior distributions

Benoit Brummer
Christophe De Vleeschouwer

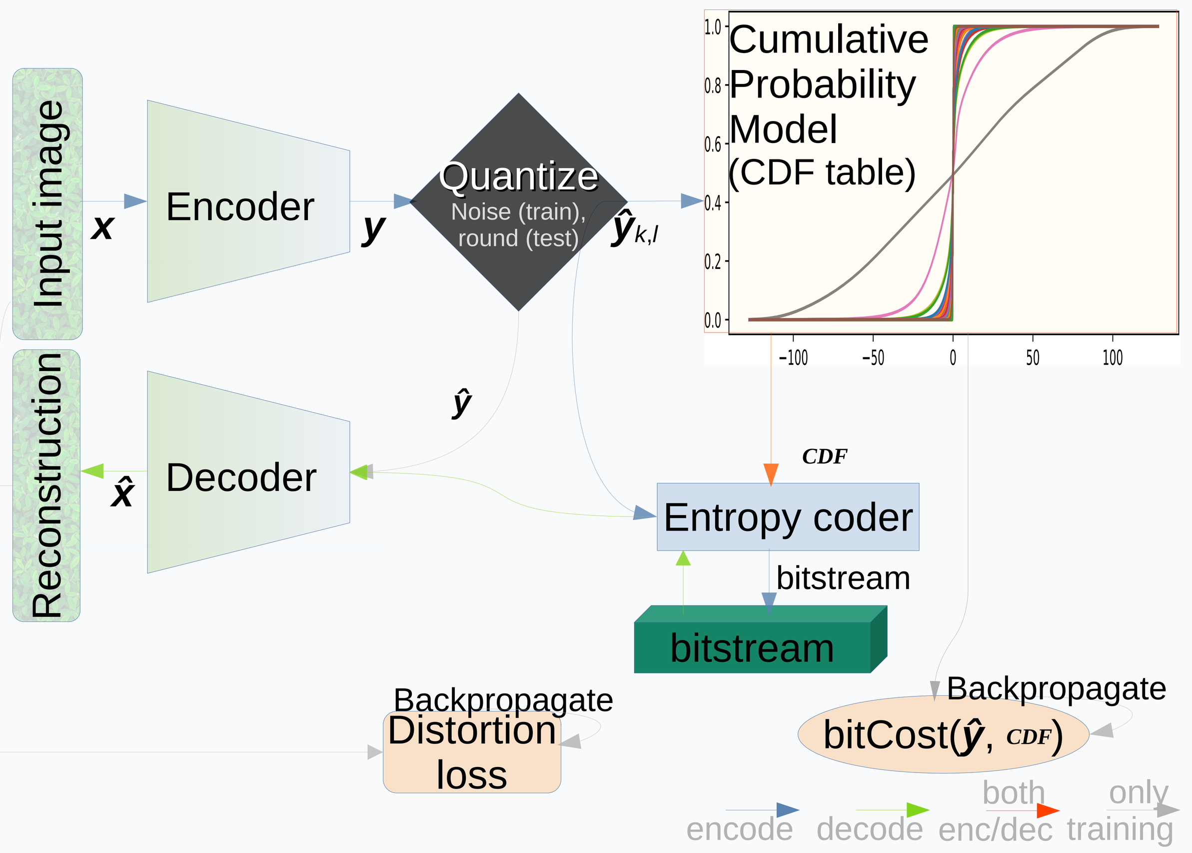
UCLouvain — intoPIX — Wallonie

PRAIRIE/MIAI Artificial Intelligence Summer School P.A.I.S.S. 5>9 JULY 2021 ONLINE

Image compression: background

- Image compression consists of three recurring steps:
  - transformation: reduce image redundancy through more efficient representation
    - Eg: discrete wavelet transform, convolutional autoencoder
  - quantization: use a finite set of discrete symbols
    - Eg: quantization table, round
  - entropy coding: use known statistics (prior info.) to encode common symbols with fewer bits
    - Eg: range coding

Prior work: "End-to-end optimized image compression" scheme by Johannes Ballé et al.



↗ Learned compression scheme with a single cumulative probability model (ie End-to-end optimized image compression by Johannes Ballé et al.)

simple and effective:

- Encode image into a latent code (the autoencoder's bottleneck)
  - More channels, smaller spatial dimensions
- Quantize: round to nearest integer
  - or add random noise for differentiable training
- Model the **cumulative distribution function** (CDF) of each channel
  - CDF(x) = probability that a variable is < x
  - Independent channels; each has its CDF
  - Used to calculate bitcost and for entropy coding
- Decode the image back to RGB

Training:

minimize Loss = distortion × λ + bitrate

Encoder and probability model adapt to one other

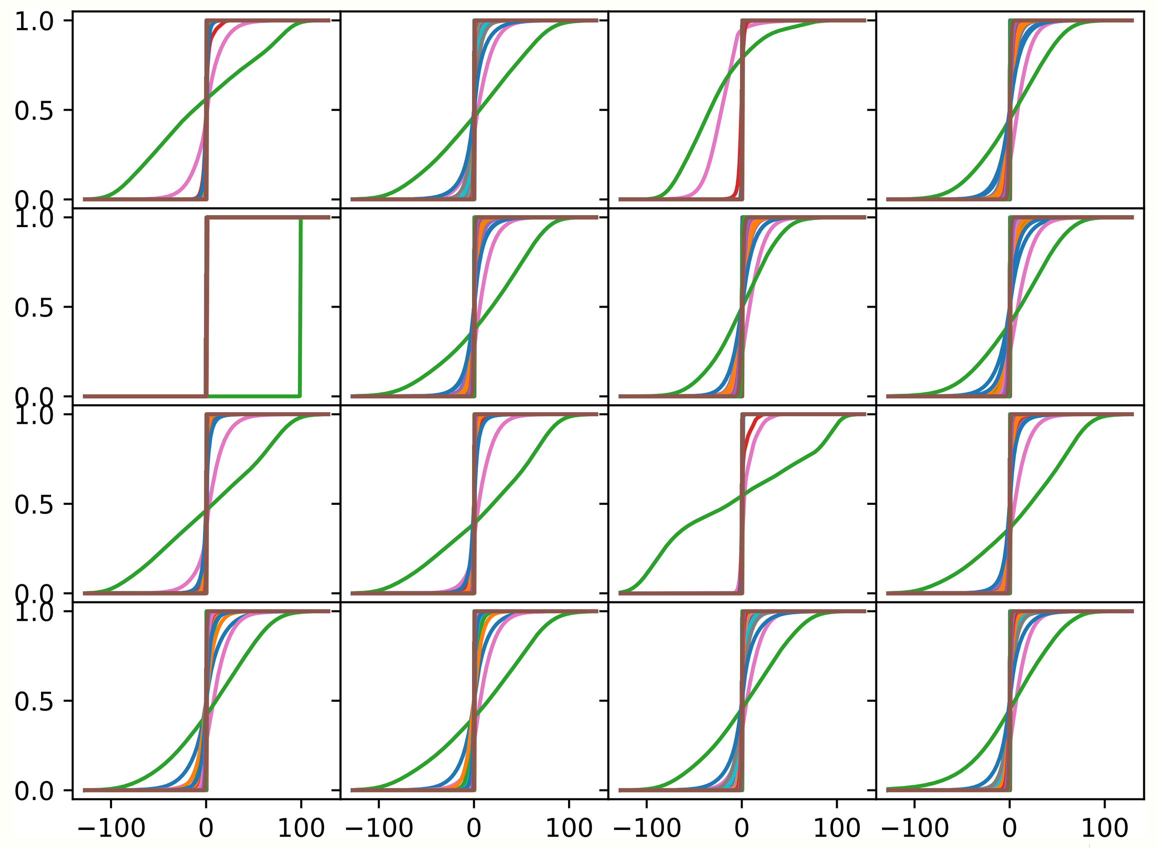
Cons:

- JPEG2000-like performance, unique autoencoder and probability model limit one other's expressiveness
- A common solution is to analyze the latent code with a "hyperprior" sub-network (Balle2018), and parametrize the CDF s.t. it is different for each symbol to be encoded.
- But constant switches and memory accesses increase the complexity and runtime.

References: **Balle2017**:End-to-end optimized image compression with competition of prior distributions by Johannes Ballé et al.
Balle2018: Variational image compression with a scale hyperprior by Johannes Ballé et al.

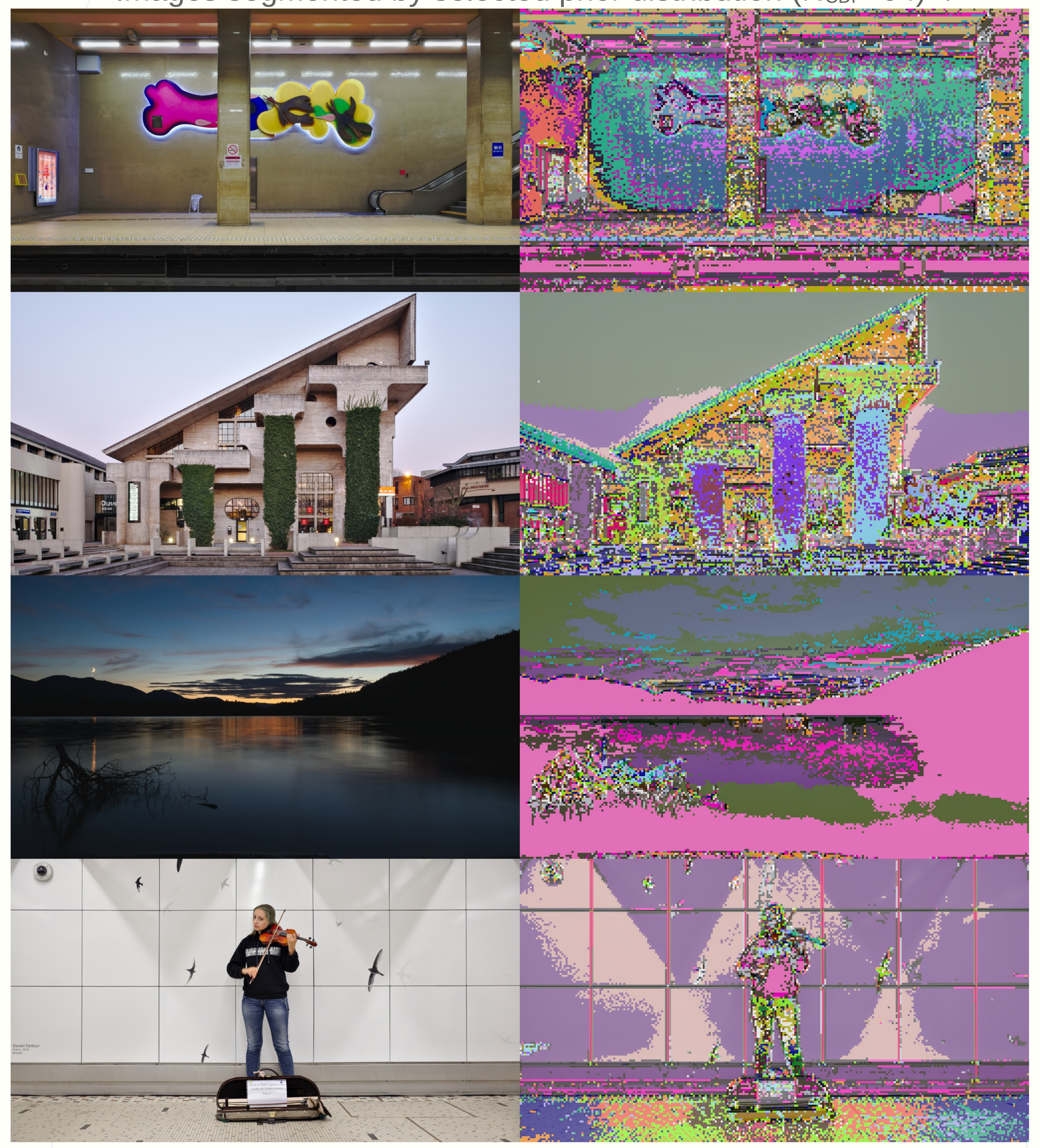
Image compression with competition of prior distributions; our contribution

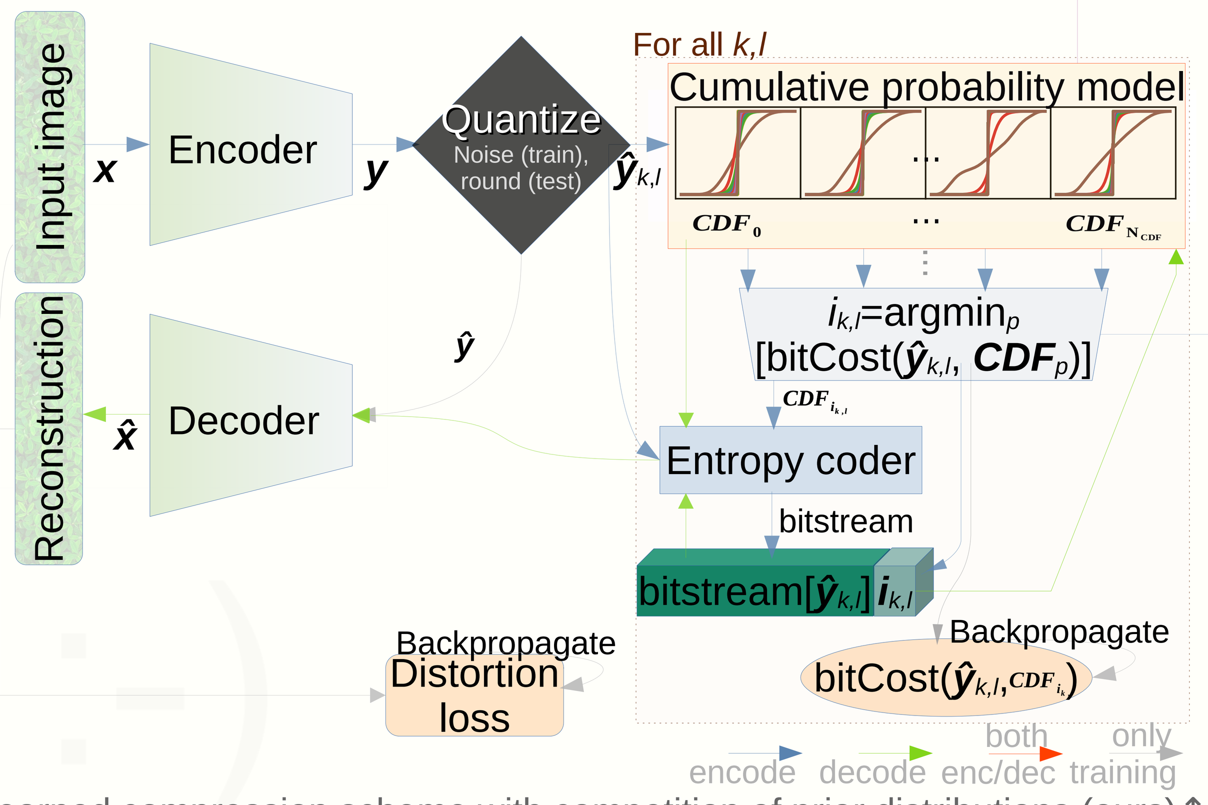
- Many ***CDF tables*** are learned
  - eg : N_{CDF}=64
  - Where each CDF table covers all (typ. 256) channels
- The best ***CDF table*** for each location is optimized on that location (training), or its index is stored with the bitstream (encoding)



16 optimized CDF tables (1 box / CDF table; 1 line / channel) ↑

Images segmented by selected prior distribution (N_{CDF}=64) ↓

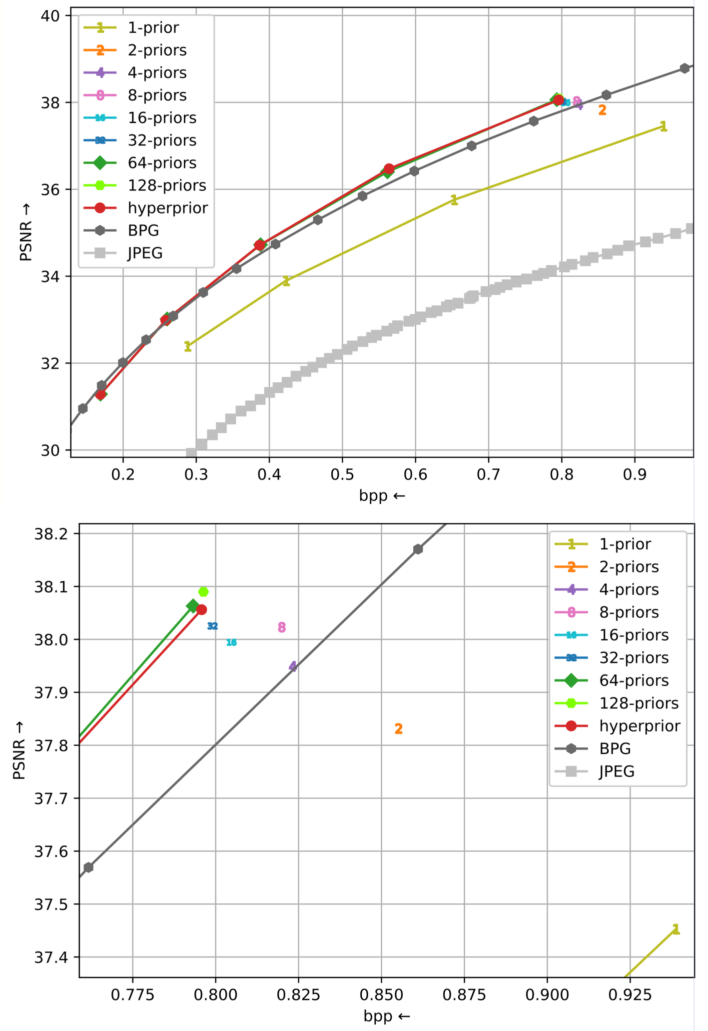




Learned compression scheme with competition of prior distributions (ours) ↑

Results

- Rate-distortion with 64-priors is better than Balle2017 (1-prior), similar to Balle2018 (hyperprior w/ per-symbol CDF)
- **Decoding** complexity is nearly the same as Balle2017
  - CDF generation consumes **0.14x as much CPU time** with 64-priors as with hyperprior
    - Because decoder works with **static CDF tables** which are **defined channel-wide**



Rate-distortion with different number of priors (PSNR on Commons Test Photographs) ↑

Image compressed with different schemes (visual comparison) ↓

Conclusion

- Multiple competing priors improve rate-distortion (RD) performance compared to the use of a single cumulative distribution function (CDF) per channel
- Using 64 static priors reduces complexity and achieves similar rate-distortion performance compared to the use of a hyperprior

:-)

PyTorch implementation: https://github.com/trougnouf/Manypriors