7. Appendix to "Computationally-Efficient Neural Image Compression with Shallow Decoders"

7.1. More details on the two-layer synthesis with residual connection

Fig. 10 illustrates the proposed two-layer architecture with a residual connection, where $\mathbf{a} = \texttt{conv\_1}(\mathbf{z})$ denotes the output of the first transposed conv layer. We implement the residual connection (the lower computation path in the figure) with another transposed convolution layer `conv_res`, using the same configuration (stride, kernel size, etc.) as `conv_1`. In our main experiments we use $k_1 = 13, s_1 = 8, k_2 = 5, s_2 = 2$ and $N = 12$.

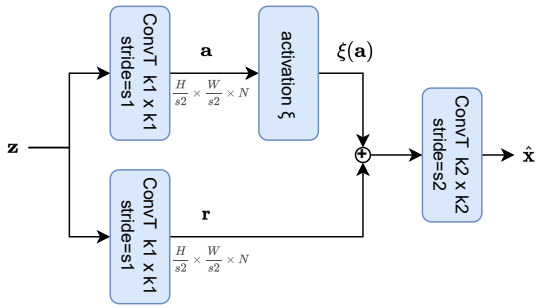


Figure 10. Diagram of the proposed two-layer synthesis transform.

The residual connection $\mathbf{r}$ is inspired by its success in recent NTC architectures [13, 25], and can also be interpreted as a data-dependent and spatially-varying bias term that modulates the nonlinear activation $\xi(\mathbf{a})$. At lower bit-rates, we found employing the residual connection to be more parameter- and compute-efficient than a simple composition of two transposed layers without the residual connection.

In further experiments, we re-trained models with "mixed quantization" [36] instead of additive uniform noise as in the main paper, and found that with comparable decoding complexity, a simple two-layer architecture with $N = 24$ hidden channels (and no residual connection) in fact slightly outperforms the one with residual connection and $N = 12$, while keeping all other hyperparameters the same.

7.2. Theoretical Result

In the following, we derive the decomposition of the R-D cost of neural lossy compression. To lighten notation, we use non-bold letters (x, z instead of $\mathbf{x}, \mathbf{z}$), and adopt the general setting where the latent space $\mathcal{Z}$ is a Polish space (which includes, among many other examples, the Euclidean space $\mathbb{R}^{|\mathcal{Z}|}$ commonly used for continuous latent variables, or the set of lattice points $\mathbb{R}^{|\mathcal{Z}|}$ in nonlinear transform coding), and P_Z is a prior probability measure. It is harmless to focus on the common case where P_Z admits a density $p(z)$ (denoted $p(\mathbf{z})$ in the main paper), s.t., $P_Z(dz) = p(z)dz$; in the discrete case, $p(z)$ is a PMF, and the integral (w.r.t. the counting measure on $\mathcal{Z}$) reduces to a sum. Similarly, $Q_{Z|X}$ is family of probability measures, such that for each value of x it defines a conditional distribution $Q_{Z|X=x}$, which may admit a density $q(z|x)$.

A (learned) lossy compression codec consists of a prior distribution P_Z, a stochastic encoding transform $Q_{Z|X}$, and a deterministic decoding transform $g : \mathcal{Z} \to \hat{\mathcal{X}}$. Suppose relative entropy coding [16, 44, 19] operates with minimal overhead, i.e., a sample from $Q_{Z|X=x}$ can be transmitted under P_Z with a bit-rate close to $KL(Q_{Z|X=x}\|P_Z)$ (which may require us to perform block coding), then given a data realization x, the rate-distortion compression cost is, on average,

$$\mathcal{L}(Q_{Z|X}, g, P_Z, x) := \lambda \mathbb{E}_{z\sim Q_{Z|X=x}}[\rho(x, g(z))] + KL(Q_{Z|X=x}\|P_Z), \tag{7}$$

where $\rho : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty)$ is the distortion function, and $\lambda \geq 0$ is a fixed hyperparameter trading off between rate and distortion, and both ρ and λ are specified by the lossy compression problem in advance.

By Lemma 8.5 of [24], this compression cost admits a similar decomposition to the negative ELBO,

$$\mathcal{L}(Q_{Z|X}, g, P_Z, x) = -\log \Gamma_{g,P_Z}(x) + KL(Q_{Z|X=x}\|P_{Z|X=x}), \tag{8}$$

Our code can be found at https://github.com/mandt-lab/shallow-ntc.

where $P_{Z|X}$ denotes the Markov kernel (transitional distribution) defined by

$$P_{Z|X=x}(dz) := \frac{e^{-\lambda\rho(x,g(z))}P_Z(dz)}{\Gamma_{g,P_Z}(x)}, \tag{9}$$

and the normalizing constant is

$$\Gamma_{g,P_Z}(x) := \int_{\mathcal{Z}} e^{-\lambda\rho(x,g(z))}P_Z(dz). \tag{10}$$

As in variational Bayesian inference, the normalizing constant has the interpretation of a marginal log-likelihood specified by the prior P_Z and model g. We note that the definition of $P_{Z|X}$ depends on g and P_Z, but leave this out to lighten notation. Eq. 8 together with the non-negativity of KL divergence imply that $P_{Z|X}$ is the optimal channel ("inference distribution") for reverse channel coding, under which the compression cost equals:

$$\min_{Q_{Z|X=x}} \lambda\mathbb{E}_{z\sim Q_{Z|X=x}}[\rho(x,g(z))] + KL(Q_{Z|X=x}\|P_Z) = -\log\Gamma_{g,P_Z}(x) \tag{11}$$

Let P_X be the data distribution. Taking the expected value of Eq. 7 w.r.t. $x \sim P_X$ gives the population-level compression cost, which can then be rewritten as follows

$$\begin{aligned}
\mathcal{L}(Q_{Z|X}, g, P_Z) &:= \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,g(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_Z)] && (12)\\
&= \mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})] && (13)\\
&= \inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log\Gamma_{\omega,P'_Z}(x)] + \left(\mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] - \inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log\Gamma_{\omega,P'_Z}(x)]\right) && (14)\\
&+ \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})] && (15)\\
&= \inf_{\omega\in\mathcal{G}} F_\omega(\lambda) + \underbrace{\left(\mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] - \inf_{\omega\in\mathcal{G}} F_\omega(\lambda)\right)}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})]}_{\text{inference gap}}, && (16)
\end{aligned}$$

where for any choice of decoder function $\omega \in \mathcal{G}$, we define

$$F_\omega(\lambda) := \inf_{Q_{Z|X}} I(X;Z) + \lambda\mathbb{E}[\rho(X,\omega(Z))] \tag{17}$$

as the R-axis intercept of the line tangent to the ω-dependent rate-distortion function [50] [6] with slope $-\lambda$. The last equation follows from Eq. 11 and the following calculation

$$\begin{aligned}
&\inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log\Gamma_{\omega,P'_Z}(x)] && (18)\\
&= \inf_{P'_Z,\omega}\inf_{Q_{Z|X}} \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] && (19)\\
&= \inf_{\omega}\inf_{Q_{Z|X}}\inf_{P'_Z} \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] && (20)\\
&= \inf_{\omega}\inf_{Q_{Z|X}} \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \inf_{P'_Z}\mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] && (21)\\
&= \inf_{\omega}\inf_{Q_{Z|X}} \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + I(P_X Q_{Z|X}) && (22)\\
&= \inf_{\omega} F_\omega(\lambda) && (23)
\end{aligned}$$

To summarize, we have broken down the R-D cost for a data source P_X into three terms,

$$\mathcal{L}(Q_{Z|X}, \omega, P_Z) = \inf_{\omega\in\mathcal{G}} F_\omega(\lambda) + \underbrace{\left(\mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] - \inf_{\omega\in\mathcal{G}} F_\omega(\lambda)\right)}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})]}_{\text{inference gap}}. \tag{24}$$

[6] The g-dependent distortion function [50] is defined as $R_g(D) := \inf_{Q_{Z|X}:\mathbb{E}[\rho(X,g(Z))]\le D} I(X;Z)$.

- The first term (which we dentoed by the shorthand "$\mathcal{F}(\mathcal{G})$" in the main text) represents the fundamentally irreducible cost of compression determined by the source P_X and the family $\mathcal{G}$ of decoding transforms used. This is the information-theoretically optimal cost of compression within the transform family $\mathcal{G}$. If we let $F(\lambda) := \inf_{Q_{\hat{X}|X}} I(X;\hat{X}) + \lambda\mathbb{E}[\rho(X,\hat{X})]$ be the optimal Lagrangian associated with the R-D function of P_X, then it can be shown that [50] $F(\lambda) \leq F_g(\lambda)$. When the latent space $\mathcal{Z}$ and the transform family $\mathcal{G}$ are sufficiently large, it holds that $F(\lambda) = \inf_g F_g(\lambda)$, i.e., the first term of the R-D cost is determined solely by the rate-distortion function of the source distribution P_X (and distortion function ρ), which is the lossy analogue of the Shannon entropy.

- The second term represents the excess cost of doing compression with a *particular* transform g and prior P_Z compared to the *best possible* transform and prior, while always operating with the optimal channel (Eq. 8) in each case. Note that this term only depends on the modeling choices of g and P_Z, and does not depend on the encoding/inference distribution $Q_{Z|X}$; therefore we call it the *modeling gap*. It is due largely to imperfect model training/optimization, and/or a mismatch between the training data and the target data P_X (which may not be the same).

- The third term represents the overhead of compression caused by a (potentially) sub-optimal encoding/inference distribution $Q_{Z|X}$, given a particular model g and P_Z. This overhead can be eliminated by using the optimal channel $P_{Z|X}$ given in Eq. 8 (which depends on g and P_Z). Therefore we call this term the *inference gap*.

Remarks The decomposition of the lossy compression cost has a natural parallel to that of lossless compression under a latent variable model.

Consider a latent variable model $(p_\theta(z), p_\theta(x|z))$ with parameter vector θ, which defines a model of the marginal data density $p_\theta(x) := \int_{\mathcal{Z}} p_\theta(x|z)p_\theta(z)dz$. The cost of lossless compression under ideal bits-back coding [20, 45] is equal to the negative ELBO, and admits a similar decomposition [53]:

$$\mathbb{E}_{x\sim P_X}[\mathbb{E}_{z\sim q(z|x)}[-\log p_\theta(x|z)] + KL(q(z|x)\|p_\theta(z))] \tag{25}$$

$$=\mathbb{E}_{x\sim P_X}[-\log p_\theta(x)] + \mathbb{E}_{x\sim P_X}[KL(q(z|x)\|p_\theta(z|x))] \tag{26}$$

$$= \underbrace{H[P_X]}_{\text{data entropy}} + \underbrace{KL(P_X\|p_\theta(x))}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(q(z|x)\|p_\theta(z|x))]}_{\text{inference gap}} \tag{27}$$

Again, we have decomposed the compression cost into a first term that represents the intrinsic compressibility of the data, a second term that depends entirely on the choice of the model, and a third overhead term from using a sub-optimal inference distribution $q(z|x)$ and which can be eliminated using the optimal inference distribution $p_\theta(z|x)$ (the Bayesian posterior) given each choice of our model.

7.3. Implementation and reproducibility details

Our models are implemented in tensorflow using the `tensorflow-compression`[7] library. We implemented the Mean-scale Hyperprior model based on the open source code [8] and the model configuration from Johnston et al. [28]. We borrowed the ELIC [25] transforms from the VCT repo [9].

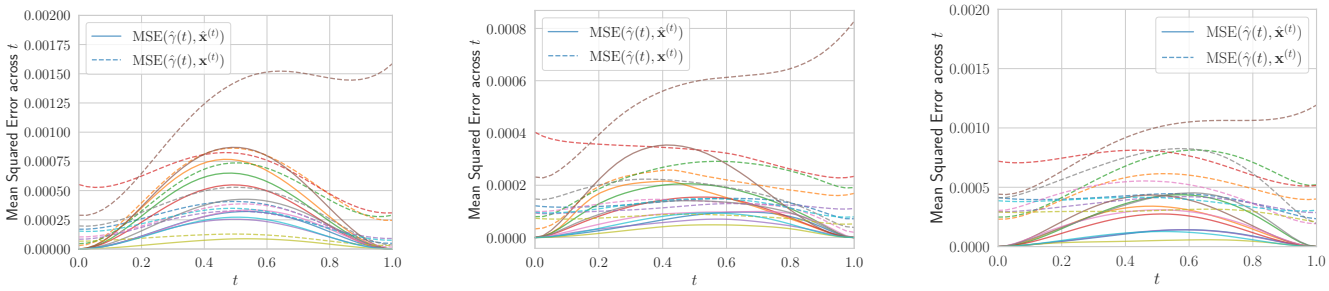
Our experiments were run on Titan RTX GPUs. All the models were trained with the Adam optimizer following standard procedure (e.g., in [35]) for a maximum of 2 million steps. We use an initial learning rate of $1e-4$, then decay it to $1e-5$ towards the end of training. For each model architecture, we trained separate models for $\lambda \in \{0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$. For SGA [49], we use similar hyperparameters as in [49], using Adam optimizer, learning rate $5e-3$, and a temperature schedule of $\tau(t) = 0.5\exp\{-0.0005(\max(0, t-200))\}$ for 3000 gradient steps.

7.4. Additional Results

7.4.1 Results on the reconstruction manifold.

We train three popular NTC architectures [5, 35, 36] for MSE distortion with $\lambda = 0.08$ and observe similar results when traversing the manifold of reconstructed images. We use random pairs of image crops from COCO [30] to define the start and end points of latent traversal (see Sec. 3.1); we use the same random seed across the three different architectures. All the images are scaled to $[-0.5, 0.5]$.

MSEs between trajectories In Fig. 11 shows the distance between the resulting trajectory of decoded curve $\hat{\gamma}(t)$ and two kinds of straight paths, interpolating between reconstructions $\hat{\mathbf{x}}^{(t)} := (1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$ or ground truth images $\hat{\mathbf{x}}^{(t)} := (1-t)\mathbf{x}^{(0)} + t\mathbf{x}^{(1)}$. See detailed discussions in Sec. 3.1.



(a) Factorized Prior [5]. (b) Mean-scale Hyperprior [35]. (c) Channel-wise Autoregressive [36].

Figure 11. The distance from the trajectory of decoded curve $\hat{\gamma}(t)$ to the straight path between end-point reconstructions $\hat{\mathbf{x}}^{(t)} := (1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$, and to the straight path between ground truth images $\hat{\mathbf{x}}^{(t)} := (1-t)\mathbf{x}^{(0)} + t\mathbf{x}^{(1)}$.

Quantifying the curvature of decoded curves of reconstructions. Additionally, we quantify how much the curves of reconstructed images deviate from straight paths by computing the curve lengths. Recall given two tensors of latent coefficients $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ (obtained by passing two images $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ through the analysis transform), we let $\gamma : [0,1] \to \mathcal{Z}$ be the straight line in the latent space defined by their convex combination, i.e., $\gamma(t) := (1-t)\mathbf{z}^{(0)} + t\mathbf{z}^{(1)}$. The curve of reconstructions is then defined by $\hat{\gamma}(t) := g(\gamma(t))$, with end-points $\hat{\mathbf{x}}^{(0)} := g(\mathbf{z}^{(0)})$ and $\hat{\mathbf{x}}^{(1)} := g(\mathbf{z}^{(1)})$.

Following Chen et al. [12], the curve length of $\hat{\gamma}$ is given by

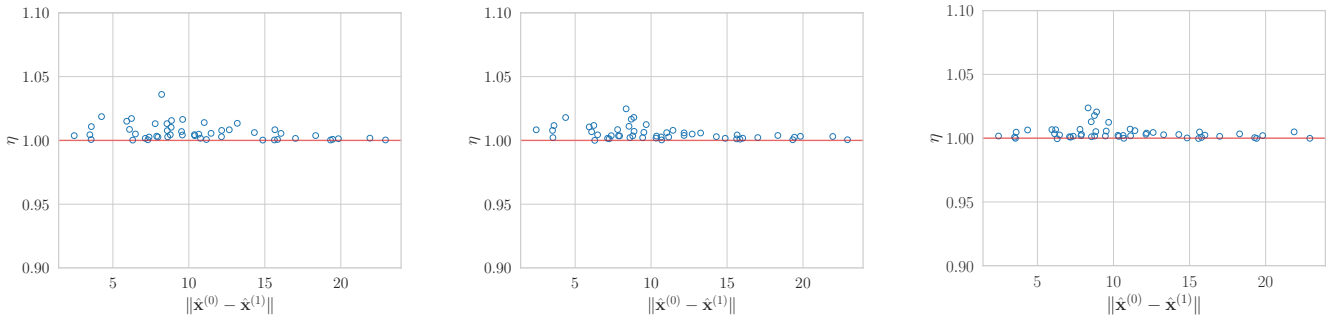
$$L(\hat{\gamma}) := \int_0^1 \|\frac{\partial g(\gamma(t))}{\partial t}\|\mathrm{d}t = \int_0^1 \|\frac{\partial g(\gamma(t))}{\partial \gamma(t)}\frac{\partial \gamma(t)}{\partial t}\|\mathrm{d}t = \int_0^1 \|\mathbf{J}_t\frac{\partial \gamma(t)}{\partial t}\|\mathrm{d}t \tag{28}$$

where $\mathbf{J}_t \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{Z}|}$ is the Jacobian of the synthesis transform evaluated at $\mathbf{z}^{(t)} = \gamma(t)$. As in [12], we compute this integral approximately with a Riemann sum.

[7] https://github.com/tensorflow/compression

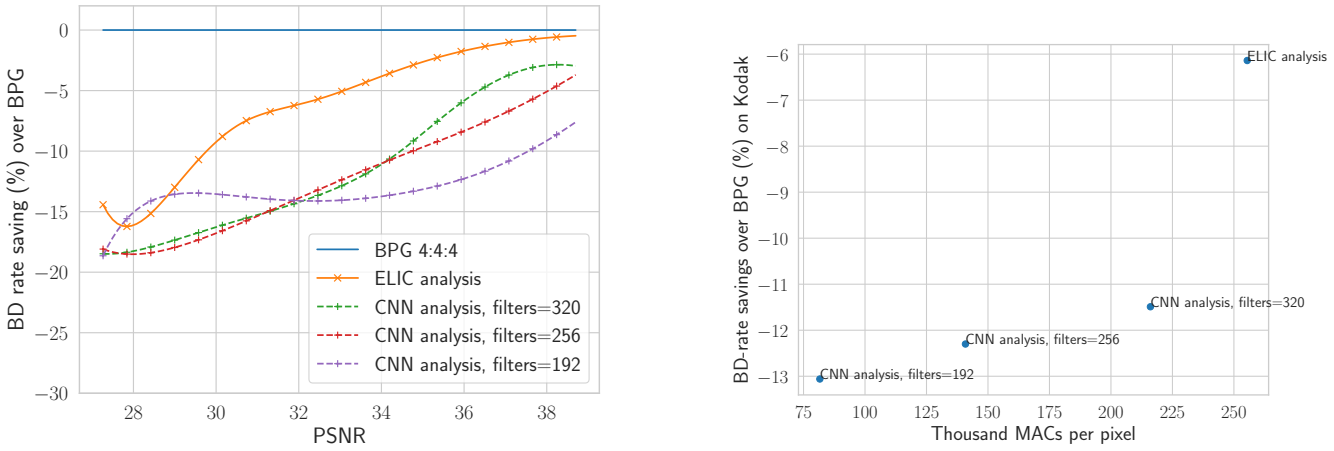
[8] https://github.com/tensorflow/compression/blob/master/models/bmshj2018.py

[9] https://github.com/google-research/google-research/blob/master/vct/src/elic.py



(a) Factorized Prior [5]. (b) Mean-scale Hyperprior [35]. (c) Channel-wise Autoregressive [36].

Figure 12. The curve-length ratio η v.s. the straight-path-length for randomly chosen image pairs, in three different nonlinear transform coding architectures [5, 35, 36]. In all cases, the curve lengths are close to the lengths of straight paths.



(a) BD-rate savings over BPG on Kodak, for various choices of analysis transforms.

(b) Aggregate BD-rate savings on Kodak, v.s. analysis transform complexity, measured in KMACs per pixel.

Figure 13. Ablation results on the choice of analysis transform in the proposed two-layer synthesis architecture. Using the CNN analysis from Mean-scale Hyperprior [35] gave relatively worse R-D performance than the analysis transform from ELIC [25].

The shortest path (in Euclidean geometry) between the two end-points of $\hat{\gamma}$ is given simply by the linear interpolation $(1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$, with a distance of $\|\hat{\mathbf{x}}^{(1)} - \hat{\mathbf{x}}^{(0)}\|$. Therefore, we define the curve-to-shortest-path length ratio

$$\eta := \frac{L(\hat{\gamma})}{\|\hat{\mathbf{x}}^{(1)} - \hat{\mathbf{x}}^{(0)}\|} \tag{29}$$

as a measure of how much the curve $\hat{\gamma}$ deviates from a straight path, with $\eta = 1$ indicating a completely straight line.

We compute the curve-to-shortest-path-length ratio η on 50 randomly chosen image pairs in three NTC architectures [5, 35, 36]. We use random 16×16 image crops (the results are similar for larger images) from COCO [30]. We compute the curve length integral in Eq. 28 via a Riemann sum, $\frac{1}{T}\sum_{t_i} \|\mathbf{J}_{t_i}(\mathbf{z}^{(1)} - \mathbf{z}^{(0)})\|$, with $T = 100$. Fig. 12 plots the resulting η values against the straight-path lengths. In all cases, the curve lengths are close to the straight-path lengths (η concentrated near 1), and this property appears to hold globally across randomly chosen image pairs.

7.4.2 Ablation results.

The analysis transform. Here, we examine how different choices of the analysis transform affect the performance of our method based on the two-layer synthesis transform and ELIC's analysis transform.

We adopt the simpler CNN analysis transform from Mean-Scale Hyperprior [35], which consists of 4 layers of convolutions with $F = 192$ filters each, except for the last layer which outputs $C = 320$ channels for the latent tensor. Fig. 13 shows the resulting performance with varying F, in both BD-rate savings as well as computational complexity. We see that the CNN

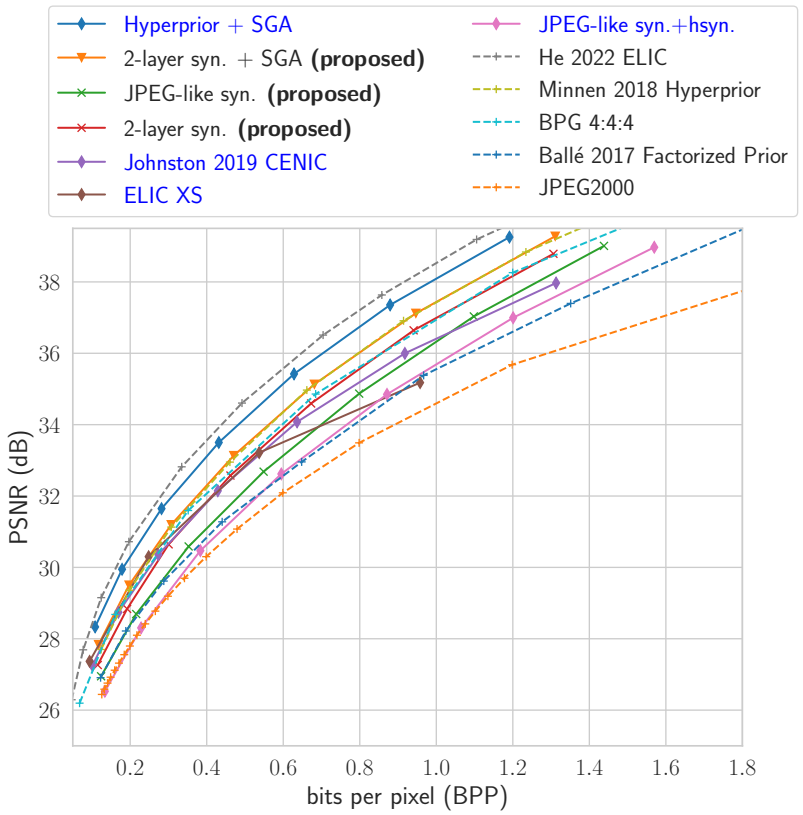


Figure 14. Miscellaneous additional results, with blue legend labels to be distinguished from results in the main paper. We (1) apply SGA also to the Hyperprior baseline; (2) simply scale down existing NTC architectures to roughly match the FLOPs of our two-layer synthesis; (3) explore a JPEG-like hyper synthesis transform to improve its computational efficiency.

analysis gave worse performance than ELIC analysis, and the gap can be closed to some extent by increasing F, but with diminishing returns and increasingly high encoding complexity.

Additional investigations. We present results giving additional insight into our method and how it compares to alternatives, evaluated on Kodak. The additional results are highlighted with blue legend titles in Figure 14. First, we consider also applying SGA to the Hyperprior baseline; as shown by the solid blue line in Figure 14, this also results in a sizable boost in R-D, even larger than what we observe for our more shallow decoders. We hypothesize that this may be caused by a relatively larger inference gap in the Hyperprior architecture than ours with shallow decoders.

Next, we show that simply scaling down existing neural compression models tends to result in worse performance than our approach. We consider two existing architectures: the mean-scale Hyperprior [35] and ELIC [25], and slim down their synthesis transforms to match (to our best ability) the FLOPs of our two-layer shallow synthesis. For Hyperprior, we adopt a pruned synthesis transform given by CENIC [28] (specifically, we use their architecture # 178, which uses about 7.3 KMACs/pixel, or about 1.4 times of our two-layer synthesis; we keep the hyper synthesis intact). For ELIC, we simply reduce the number of conv channels in the synthesis to be 32, so that it uses about 16.5 KMACs/pixel (we also keep its hyper synthesis intact). We train the resulting architectures from scratch; as shown by the purple ("Johnston 2019 CENIC") and brown curves ("ELIC XS"), this results in progressively worse R-D performance in the higher-rate regime compared to our two-layer synthesis (red curve).

Finally, we conduct a preliminary exploration of a JPEG-like architecture for the hyper synthesis transform. We implement this with a single transposed-conv layer with stride 4 and (6, 6) kernels. We applied it on top of our linear JPEG-like synthesis, and observe a 10% worse BD-rate (green curve $\rightarrow$ pink curve in Figure 14) but nearly a 10-fold reduction in the hyper synthesis FLOPs (15.18 $\rightarrow$ 1.8 KMACs/pixel).

7.4.3 Additional R-D results

Below we include additional R-D results on the 100 test images from Tecnick [2] and 41 images from the professional validation set of CLIC 2018 [10]. We additionally evaluate on the perceptual distortion LPIPS [54]. Overall, we observe that our proposed two-layer synthesis with iterative encoding matches the Hyperprior performance when evaluated on PSNR, but under-performs by $8\% \sim 12\%$ (in BD-rate) when evaluated on perceptual metrics such as MS-SSIM or LPIPS. This is consistent with results on Kodak in Sec. 4.2.

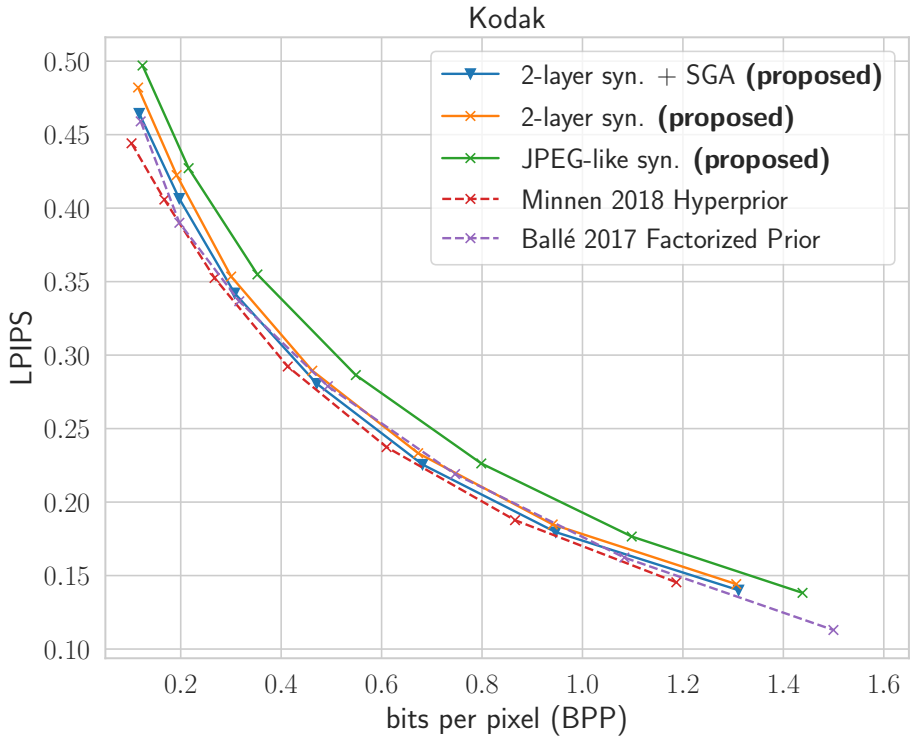


Figure 15. Aggregate LPIPS v.s. BPP performance on Kodak.

[10] http://clic.compression.cc/2018/challenge/

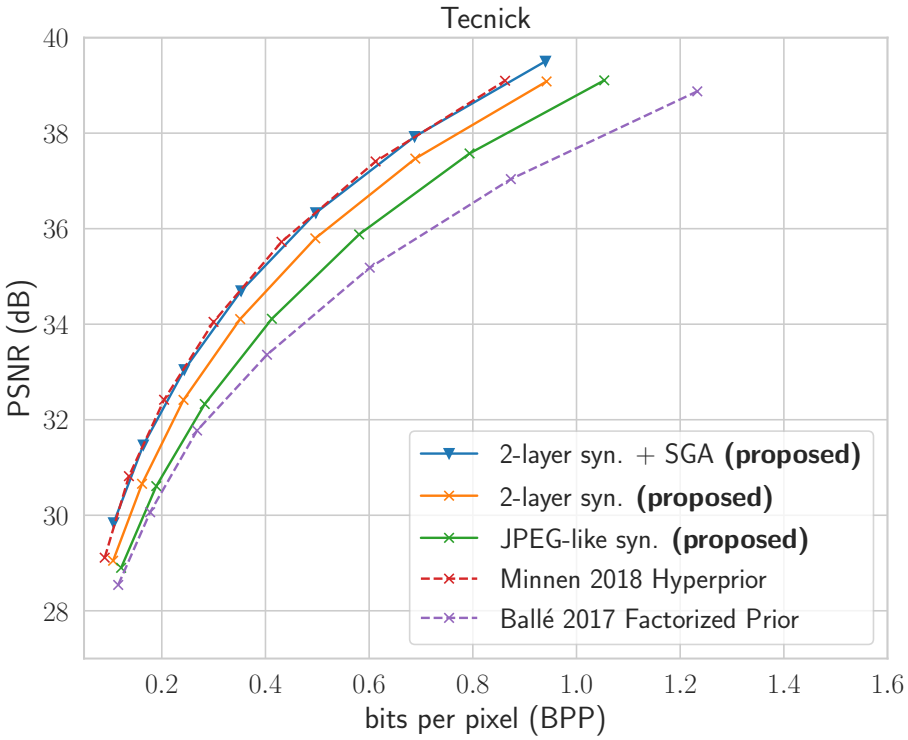


Figure 16. Aggregate PSNR v.s. BPP performance on Tecnick.

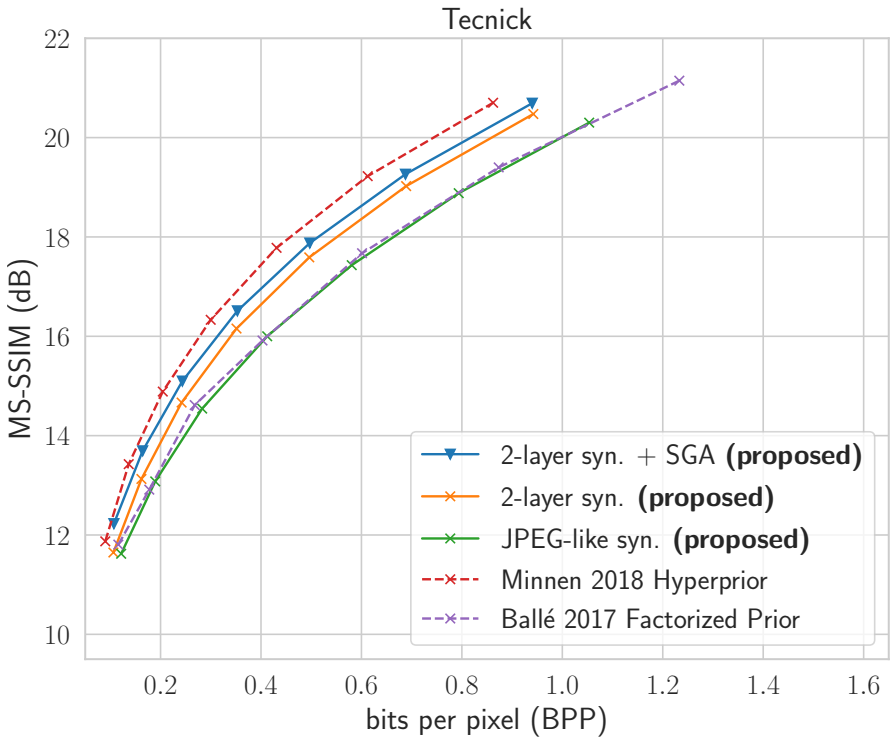


Figure 17. Aggregate MS-SSIM v.s. BPP performance on Tecnick.

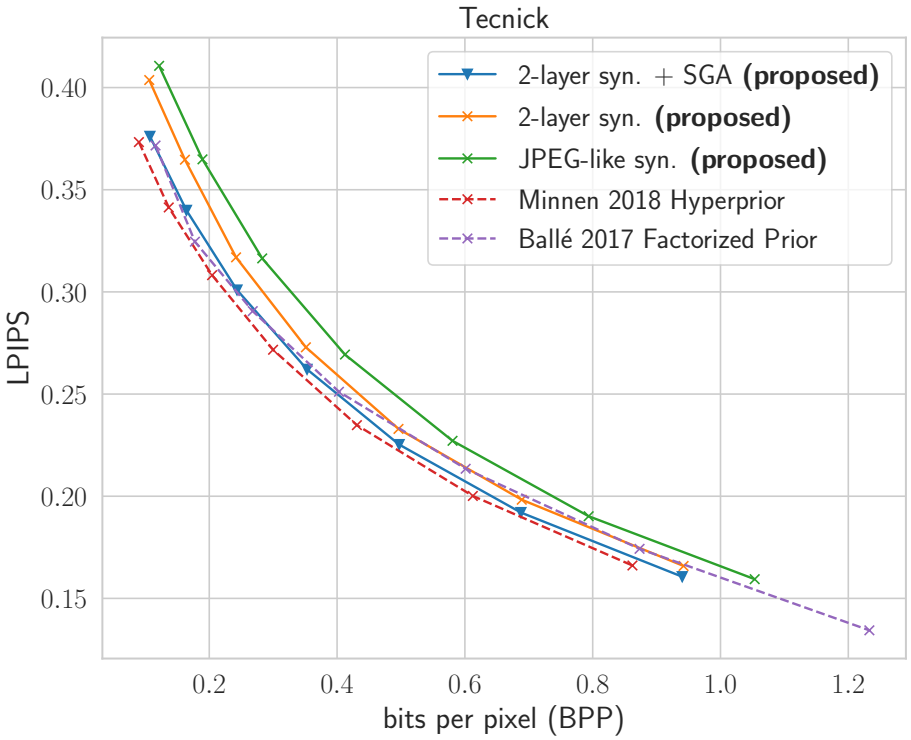


Figure 18. Aggregate LPIPS v.s. BPP performance on Tecnick.

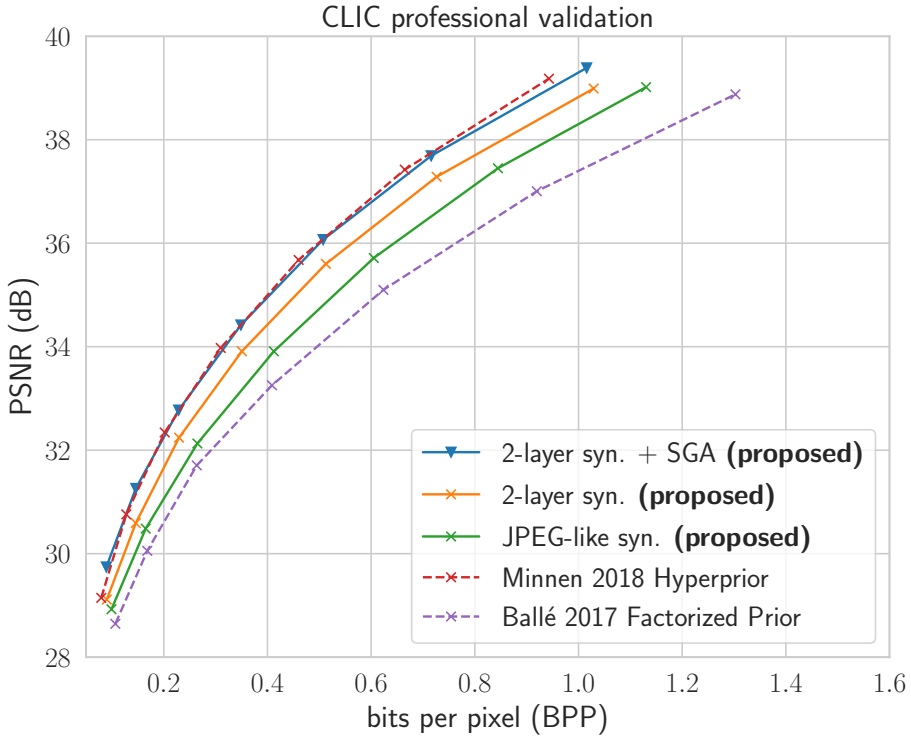


Figure 19. Aggregate PSNR v.s. BPP performance on CLIC professional validation set.

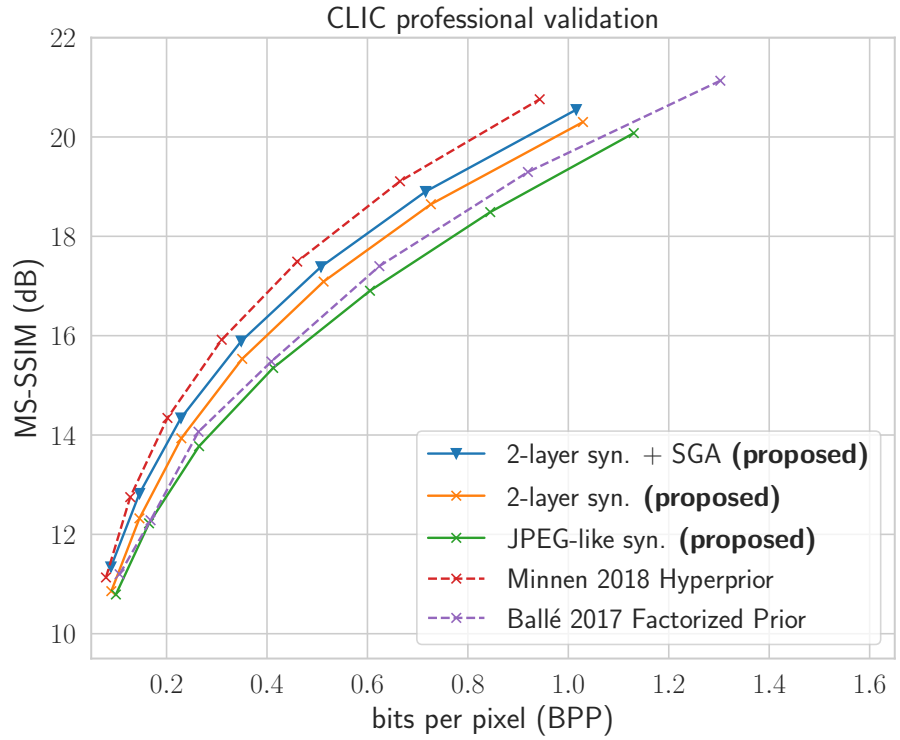


Figure 20. Aggregate MS-SSIM v.s. BPP performance on CLIC professional validation set.

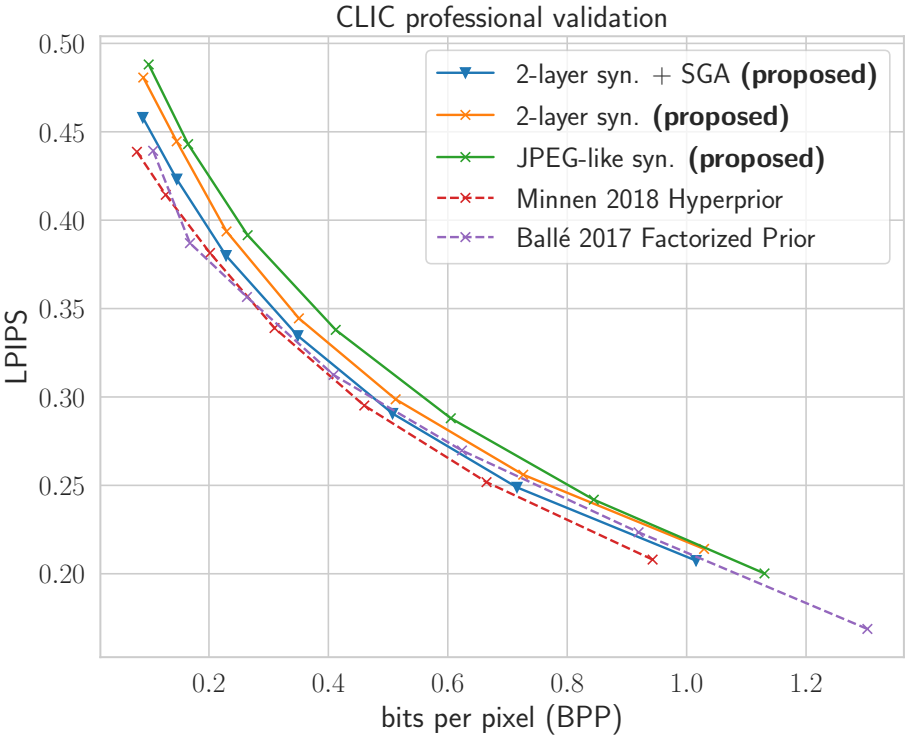


Figure 21. Aggregate LPIPS v.s. BPP performance on CLIC professional validation set.