The Sibling Neural Estimator: Improving Iterative Image Decoding with Gradient Communication

Ankur Mali*, Alexander G. Ororbia†, C. Lee Giles*

* The Pennsylvania State University, University Park, PA, 16802, USA

†Rochester Institute of Technology, Rochester, NY, 14623, USA

Abstract

For lossy image compression, we develop a neural-based system which learns a nonlinear estimator for decoding from quantized representations. The system links two recurrent networks that "help" each other reconstruct the same target image patches using complementary portions of the spatial context, communicating with each other via gradient signals. This dual agent system builds upon prior work that proposed an iterative refinement algorithm for recurrent neural network (RNN) based decoding. Our approach works with any neural or non-neural encoder. Our system progressively reduces image patch reconstruction error over a fixed number of steps. Experiments with variations of RNN memory cells show that our system consistently creates lower distortion images of higher perceptual quality compared to other approaches. Specifically, on the Kodak Lossless True Color Image Suite, we observe gains of 1.64 decibel (dB) over JPEG, a 1.46 dB over JPEG 2000, a 1.34 dB over the GOOG neural baseline, 0.36 over E2E (a modern competitive neural compression model), and 0.37 over a single iterative neural decoder.

Introduction

Image compression is a fundamental problem that has been at the core of signal and image processing research and applications for decades. Traditional and widely-used approaches such as JPEG and JPEG-2000 rely on hand-crafted codecs and incorporate fixed transformation matrices along with quantization and entropy encoders. However, one cannot expect a fixed methodology to obtain optimal solutions for all images with variable content and type. Recently, deep neural networks (DNNs) have had success on challenging problems such as speech processing, computer vision, and natural language processing. Current efforts have shown impressive results with end-to-end image compression systems using DNNs. [1, 2, 3]. While powerful, these systems require carefully designing and training effective encoding/decoding functions as well as quantizers. Since quantizers are discrete in nature, careful design of good, smooth approximations of the quantizer [4] is necessary in order to facilitate gradient-based optimization. Unfortunately, end-to-end DNN approaches often rely on large amounts of data and expensive hyper-parameter search, resulting in an expensive training process (in terms of computational time and memory). In contrast, recent work has shown that efficient lossy compression systems can be designed by simply combining well-established image encoders with recurrent networks that, in effect, learn to iteratively estimate a nonlinear decoder, a process known as iterative refinement [5].

In this paper, we build on top of the framework of iterative refinement and propose what we call the sibling neural estimator system. Our approach generalizes the original algorithm to effectively exploit a pair of complementary recurrent neural networks (RNNs) that together work to extract useful information from contextual image patches of different neighboring regions. Importantly, when predicting any given target image patch, these sibling RNNs learn to communicate using their gradient signals through an imbalanced communication channel. This encourages one RNN to play the role of source estimator (which extracts rich context information from immediately nearby context patches) while the other plays the role of co-estimator (extracting distant context information that might help the source RNN when reconstructing the target patch while learning to correct its own internal state using the source estimator as a guide). Our contributions are as follows:

- Propose a sibling neural estimator approach for learned iterative decoding, combining two complementary RNNs via a gradient-based communication channel.
- Develop a noise-based state regularization scheme and custom optimization process to ensure robust training.
- Investigate the effect of skipping states.

1 Related Work

Traditional lossy image compression techniques, e.g., JPEG, JPEG2000 (JP2), combine fixed transformation entropy-based encodings with optimized bit allocation in order to achieve better compression at low bit rates [6]. As a result, these approaches are often computationally and memory efficient. Recently, deep neural networks have become the de facto successful approach in many fields. The earliest breakthroughs in image compression using RNNs outperformed traditional techniques at lower bit rates in terms of visual quality [7, 3, 8]. These proposed end-to-end differentiable architectures were based on convolutional and deconvolutional LSTMs. Other research efforts have focused on designing artificial neural networks specialized for predicting images sequentially in two dimensions [9]. Variational autoencoders [10] have also been adapted as an alternative learning framework for compression. Other work has achieved state-of-the-art results using approaches based on spatial-temporal energy compaction [11] or energy compaction [12], aiming to extract better latent representations of images. Other proposed methods [13, 2, 14, 15] have focused on using convolution networks or generative adversarial networks (GANs). Recent work on using filter-bank based convolution networks has also shown improved performance when evaluating on inter-sub band dependencies [16]. Another research direction proposed an algorithm called iterative refinement [5] which combined a custom RNN component with traditional encoders. This approach yielded better rate-distortion tradeoff at low bit-rates.

2 Nonlinear Estimation using Iterative Decoding and Sibling Models

The Iterative Refinement Procedure: This method reconstructs images from a compressed representation, treating the problem as a multi-step reconstruction process. First, an RNN nonlinear estimator is defined, with parameters $\Theta = \{\Theta_s, \Theta_t, \Theta_d\}$, which will take in N neighboring patches as input. The estimator is defined by three key components:

- $\mathbf{e} = e(\mathbf{q}^1, \cdots, \mathbf{q}^N; \Theta)$, a transformation function of a set of encoded quantized neighboring patches that correspond to a target patch (that will be reconstructed).
- $\mathbf{s}_k = s(\mathbf{e}, \mathbf{s}_{k-1}; \Theta)$, a state function that combines a (vector) summary of past states with the summary of input spatial context (provided by the transformation function).
- $\widetilde{\mathbf{p}}_k^j = d(\mathbf{s}_k; \Theta)$, a function to predict a target given state $\mathbf{s}_k$ over K steps (an "episode").[1]

[1]We exclusively use the LSTM for our proposed SNE-RNN system.

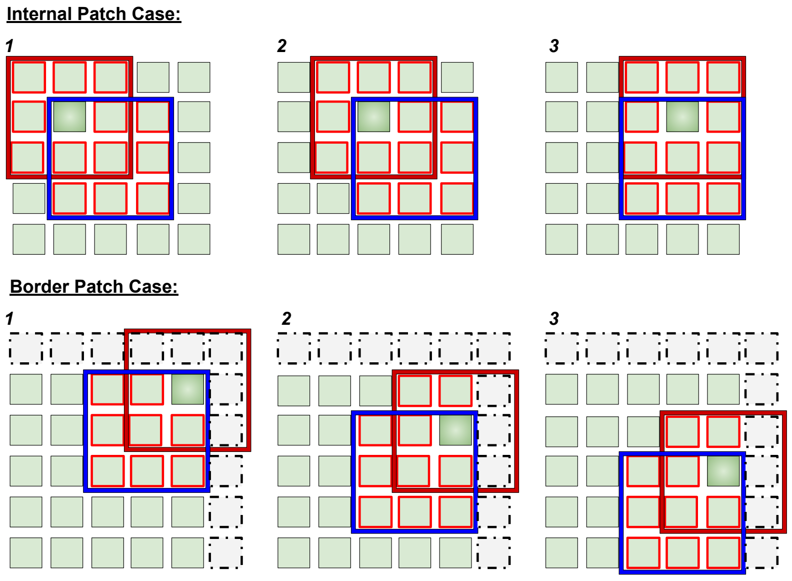

Figure 1: Scan pathway taken by the sibling RNNs. The boxes with dot-dashed boundaries are "ghost" patches, or zero-padded image patches. The red box depicts the spatial context selected as input to the central RNN while the blue depicts the spatial context provided as input to the auxiliary RNN. The dark green patch represents the target patch both sibling RNNs aim to reconstruct (which is not fed in as input to either). We have provided pseudocode for scan pathway in appendix.

Note that $\mathbf{e} \in \mathbb{R}^Z$, $\mathbf{s}_k \in \mathbb{R}^Z$, and $\mathbf{p}_k \in \mathbb{R}^P$ where Z is the number of neurons in an RNN's hidden layer and P is dimensionality of the input pixel space. For further details describing iterative refinement, we refer the reader to [5].

Functional Forms: The transformation function $\mathbf{e} = e(\mathbf{q}^1, \cdots, \mathbf{q}^N; \Theta)$ and reconstruction function $\widetilde{\mathbf{p}}^j_k = d(\mathbf{s}_k; \Theta)$ can be parameterized by multilayer perceptrons (MLPs) as follows:

$$\mathbf{e} = \phi_e(W_1\mathbf{q}_1 + \cdots + W_N\mathbf{q}_N), \text{ and, } \widetilde{\mathbf{p}}_k = \phi_d(U\mathbf{s}_k + \mathbf{c})$$

where $\phi_e(v) = v$ and $\phi_d(v) = v$ (or identity functions). For the state function $\mathbf{s}_k = s(\mathbf{e}, \mathbf{s}_{k-1}; \Theta)$, one may select from a wide variety of recurrent structures, ranging from the classical Elman RNN to gated structures such as the Long Short Term Memory (LSTM), the gated recurrent unit (GRU), or the Δ-RNN. In preliminary experiments, we used a wide variety of possible memory cell types and determined that LSTM cells yielded best performance overall. Thus, we exclusively use the LSTM for our proposed SNE-RNN system.

2.1 Sibling Neural Decoding System

Here we propose a new generalization of iterative refinement we call the sibling neural estimator (SNE). The intuition is that the original form of iterative refinement only directly fed into an RNN estimator a limited amount of spatial context and heavily relied on the model's internal state to carry information across reconstruction episodes. The primary reasons a stateful estimator should be used for image decoding are that recurrent weights can be unrolled over a K-step reconstruction process and the unfolded network can be viewed as a deep highly nonlinear MLP with weights tied across each step. Preliminary experiments showed that a simple extension of the original estimator could force the transformation function $\mathbf{e} = e(\mathbf{q}^1, \cdots, \mathbf{q}^N; \Theta)$ to take in additional, more distant context patches and incur only a small increase in complexity. However, this leads to a degradation in PSNR by as much as 0.1 decibels (dB). Instead, we aim to extend the stateful neurons themselves by coupling together two RNN estimators (the "siblings"), yielding a more powerful, implicit way of extracting useful information in distant patches. Figure 1 depicts two typical pathways followed by a pair of sibling neural estimators (the internal patch case and the border patch case, since, according to the original iterative refinement procedure, the estimator predicts border patches last in a circular scan fashion). [2] Note that our system is different from bi-directional RNNs [17] since the two RNN estimators in our system are not processing a single chain-like sequence forwards and backwards in time and, instead, are jointly learning to error correct their patch prediction signals as a sort of multi-agent team. Formally, at each decoding step, the pair of sibling RNN estimators update their respective hidden states, $\mathbf{z}^e_{t-1} \in \mathbb{R}^Z$ (the main estimator) and $\mathbf{z}^y_{t-1} \in \mathbb{R}^Z$ (the co-estimator). The state update equations for the SNE RNNs, at step t, are defined as follows:

$$\mathbf{z}^e_t = \phi(U\mathbf{q}_t + V\mathbf{z}^e_{t-1}) \tag{1}$$

$$\mathbf{z}^y_t = \phi(U\mathbf{q}_t + V\mathbf{z}^y_{t+1}) \tag{2}$$

where ϕ is non-linear activation function, such as the logistic sigmoid $\phi(v) = 1/(1+exp(-v))$ or the hyperbolic tangent $\phi(v) = (exp(2v) - 1)/(exp(2v) + 1)$. U and V are learnable synaptic weight matrices. [3]. Both $\mathbf{z}^e_t$ and $\mathbf{z}^y_t$ contain useful information for predicting the current patch, or $\mathbf{p}^j$ and the key is to create a communication channel between them. Since we desire a test-time decoding process that is as fast as the original version of iterative refinement, we will have the sibling RNNs exchange gradients through their states. This means that the sibling RNNs will only be linked together at training time, and, at test-time decoding, since no gradients are calculated, we will throw away the co-estimator and only use the source estimator. In other words, the co-estimator is only used to improve the reconstruction ability of the source estimator. Furthermore, during training, we want the communication between the two RNNs to be imbalanced – $\mathbf{z}^e_t$ should be treated as the richer source of information while $\mathbf{z}^y_t$ is treated as only a helpful, auxiliary information source meant to enrich $\mathbf{z}^e_t$. Having specified the qualities we want for the communication between the sibling estimators, we can now define how gradients are passed:

$$H_{Comm} = ||(W_{err}\mathbf{z^y_t}) - \mathbf{z^e_t}||. \tag{3}$$

Note that the above is a weighted euclidean distance function that measures how far off the co-estimator $\mathbf{z}^y_t$ is from the source estimator $\mathbf{z}^e_t$. Since the communication H_{Comm} is meant to be imbalanced, we introduce the additional parameter matrix W_{err}, i.e., the error correction matrix which functions as a linear correction factor for the co-estimator RNN. In addition, W_{err} serves to "weaken" the co-estimator state $\mathbf{z}^y_t$ preventing the degenerate case of both RNNs collapsing to identical state values (which leads to unstable training given that both estimators take in as input different context patches). Crucially, we observe that the above distance function works in two directions. During training, the gradients calculated for the source estimator will traverse through this channel into the co-estimator and vice versa.

[2]Please see the appendix of this paper, available at https://ankurmali.github.io/, for full details of the scanning algorithmic process.

[3]Note that the above equations assume an Elman-style recurrence structure for the sake of illustration, but one could instead use LSTM, GRU, or Δ-RNN units instead.

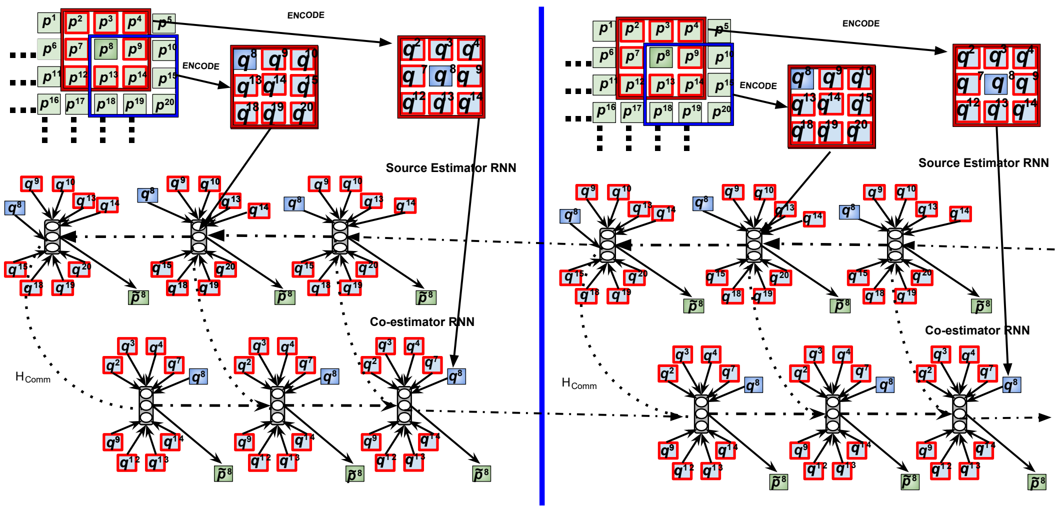

Figure 2: Iterative refinement (2 episodes, $K = 3$) for the sibling RNN estimator system (top-level view). Dot-dashed arrows indicate the latent state across an episode. H_{Comm} is the communication link between the source estimator and co-estimator RNNs. The dashed arrow indicates that the reconstruction memory is carried across each step within an episode. The source estimator RNN takes in as input the set of patches in the red boxed zone while the co-estimator RNN takes in as input the set of patches in the blue boxed zone.

To improve generalization ability, we design a noise-based regularization scheme where an adaptive variance parameter is used to generate the amount of noise based on the epoch index (such an adaptive scheme has proven to be more effective than simple gradient noise or L2/L1 decay when regularizing more complex, higher-order RNNs [18]). To start, we pick an initial value for the variance at the first epoch and lower it as a function of epoch. While this scheme has been applied directly to the synaptic weights in prior work, interestingly enough, we found that directly regularizing the state of (only) the source estimator RNN proved to be far more effective. Given this, we can rewrite H_{Comm} as follows:

$$H_{Reg-Comm} = ||(I\mathbf{z^y_t}) - (\mathbf{z^e_t} + \mathcal{N}(\mu, \sigma^2))|| \tag{4}$$

where I is the identity matrix and μ (mean) and σ^2 (variance) are user-set meta-parameters to control the strength of the state regularization noise. Our regularization schedule proceeds as follows: for every 8th epoch, we apply use the regularized channel $H_{Reg-Comm}$ instead of H_{Comm} to link the sibling RNNs (and set the number of iterative refinement steps to $K = 3$, while for the other epochs, we use H_{Comm} instead with number of refinement steps set to $K = 2$).After 120 epochs, we stop using the adaptive noise state regularization scheme and only use H_{Comm} as the communication channel.

To optimize our model parameters, we define our loss function to be the mean square error (MSE) to measure the mismatch between each estimator's reconstructed patch and the target patch $\mathbf{p}^j$. Formally, the reconstruction loss (over a K-step iterative refinement episode) is defined (over a mini-batch of B patches) as follows:

$$\mathcal{D}_{MSE}(\mathbf{p}^j, \widehat{\mathbf{p}}^j) = \frac{1}{(2BK)} \sum_{k=1}^{K} \sum_{b=1}^{B} \sum_{i} (\widetilde{\mathbf{p}}^{j,b}_k[i] - \mathbf{p}^{j,b}[i])^2. \tag{5}$$

We then apply this reconstruction loss to both the source and co-estimator RNNs. The final objective function then consists of the two reconstruction loss terms as well as the communication channel term (in this case H_{Comm}). Note that for certain epochs, as described in our noise scheme, we temporarily swap in $H_{Reg-Comm}$. The loss is defined as:

$$L = \mathcal{D}_{MSE}(\mathbf{p}^j, \widehat{\mathbf{p}}^{j,y}) + \mathcal{D}_{MSE}(\mathbf{p}^j, \widehat{\mathbf{p}}^{j,e}) + \alpha H_{Comm} \tag{6}$$

where α is an externally set coefficient to control the strength of the communication channel. The prediction of the target patch $\mathbf{p}^j$ is labelled differently for each sibling estimator, i.e., $\mathbf{p}^{j,y}$ is the co-estimator's prediction while $\mathbf{p}^{j,e}$ is the source estimator's prediction (note that we do not include the $\mathcal{D}_{MAE}$ of the original algorithm in [5] as we found this actually worsened performance for a the sibling estimator system). Backpropagation through time (BPTT) is used to calculate gradients. As a result of extensive preliminary experimentation, we found that a stable optimization process for updating the sibling estimator weights was as follows: for 120 epochs we update weights using the Adam update rule and then switch (at the moment we shut off our state regularizer) to stochastic gradient descent to conduct a gentle fine-tuning phase. Again, we reiterate that at test time decoding, we simply run the iterative refinement procedure [5] using only the source estimator RNN, giving us the same test-time decoding computational complexity as the original algorithm. After training, we discard the co-estimator RNN and only decode image patches using the source estimator RNN.

2.2 Learning to Skip States

Finally, we experimented with a variation of our RNN estimators that is based on the SkipRNN model [19]. SkipRNN, in short, entails incorporating a mechanism into the RNN's computation that essentially allows it to "skip" states as it processes a sequence. This is done to shorten the length of the underlying computational graph when using BPTT. Since our iterative decoding system employs two RNNs during training, using this will reduce computational cost without compromising overall system generalization ability. We experimented with three ways of utilizing SkipRNN: 1) SNE-RNN-SkipE, in which the system must only learn to skip the states of the source estimator RNN, 2) SNE-RNN-SkipY, in which the system learns to only skip the states of the co-estimator RNN, and 3) SNE-RNN-SkipB, in which the system learns to skip states for both sibling RNN estimators. In our implementation of the SkipRNN, we augment a network with a (scalar) binary state update gate, $u_t \in \{0, 1\}$, where 0 means we copy (or "skip") the prior time step and 1 represents the choice to update the current state. SNE-RNN-SkipB's update equation is defined as:

$$u_t^y = f_{bin}(\hat{u_t^y}), \quad u_t^e = f_{bin}(\hat{u_t^e}) \tag{7}$$

$$\mathbf{z_t^y} = u_t^y \phi(U\mathbf{x_t} + V\mathbf{z_{t-1}^y}) + (1 - u_t^y)\mathbf{z_{t-1}^y} \tag{8}$$

$$\mathbf{z_t^e} = u_t^e \phi(U\mathbf{x_t} + V\mathbf{z_{t-1}^e}) + (1 - u_t^e)\mathbf{z_{t-1}^e} \tag{9}$$

$$\triangle\hat{u_t^y} = \sigma(W_p^y \mathbf{z_t^y} + \mathbf{b}_p^y), \quad \triangle\hat{u_t^e} = \sigma(W_p^e \mathbf{z_t^e} + \mathbf{b}_p^e) \tag{10}$$

$$\hat{u_{t+1}^y} = u_t^y \triangle\hat{u_t^y} + (1 - u_t^y)(\hat{u_t^y} + min(\triangle\hat{u_t^y}, 1 - \hat{u_t^y})) \tag{11}$$

$$\hat{u_{t+1}^e} = u_t^e \triangle\hat{u_t^e} + (1 - u_t^e)(\hat{u_t^e} + min(\triangle\hat{u_t^e}, 1 - \hat{u_t^e})) \tag{12}$$

where W_p^y and W_p^e are skipping weights for co-estimator RNNs and source estimator RNNs, respectively. $\mathbf{b}_p^y \in \mathbb{R}^Z$ and $\mathbf{b}_p^e \in \mathbb{R}^Z$ represent bias vectors and $\sigma(v) = 1/1 + exp(-v)$. f_{bin}

is the binarizer (0 or 1). All operations, except f_{bin}, are differentiable. The straight-through estimator is used for the f_{bin} function in order to approximate gradients during BPTT.

3 Experiments

We implement the SNE variations previously discussed and compare test performance to JPEG and JPEG 2000 (JP2) baselines as well as an MLP stateless decoder and the original iterative refinement procedure of [5]. In addition, we compare to the competitive neural architecture[3] (GOOG) as well as the state-of-the-art variational compressor E2E [1]. All models were trained on the *Places365* dataset, using the objective function defined in Equation 6. The only pre-processing entailed normalizing pixel values to the range $[0, 1]$. For evaluation, we converted model output back to the range $[0 - 255]$. Compression results are presented for all models on the six image compression benchmarks used in [5], i.e., Kodak, CB 8-Bit, CB 16-Bit, CB 16-Bit-Linear, Tecnick, and Wikipedia. For a complete description of the compression benchmarks, we refer the reader to prior work [5].

3.1 Experimental Setup

All of our SNE-RNN models contained one layer of $Z = 512$ hidden units. We randomly initialized weights from a uniform distribution,$\sim U(-0.05, 0.05)$. Parameters were updated during training using the optimization switching heuristic we described earlier. Gradients were estimated over mini-batches of 512 samples and hard-clipped to a magnitude of 15 (to prevent exploding gradients). We started training with Adam with an initial learning rate of 2e-4 and used polynomial decay until the 120th epoch. After the 120th epoch, we switch to stochastic gradient descent (SGD) and continue our training up to 300 total epochs. This approach, as we described earlier, exploits the faster convergence properties of Adam with the fine-tuning capabilities of SGD (note that this approach also performs better than the stochastic annealing learning rate used in prior work on neural-based decoder estimation). We also use the two-step shuffling approach [5] to further ensure robust training.

3.2 Experimental Results

Compression results for the 6 benchmark datasets mentioned above are shown in Table 2. Each model was evaluated using three popular metrics [20]: PSNR, structural similarity (SSIM), and multi-scale structural similarity (MS-SSIM [21], or MS^3IM as shown in our tables). Table 2 shows that our proposed SNE approach yields the best results. Our approach consistently yields better performance when compared to classical models (JPEG, JP2), competing end-to-end neural compression models (GOOG, E2E), and the original iterative refinement procedure (LSTM-JPEG , LSTM-JP2). We emphasize that our results are measured on six different independent benchmarks, which contain unseen and out of sample images to test the true generalization performance. Table 1 shows how PSNR chagnes as the number of reconstruction steps K allowed per episode is varied. To create this table, we randomly sample 4 images from our Kodak test suite and report the PSNR measurement for our best SNE model as a function $K = \{1, 2, 3, 7, 9, 11\}$. We observe that for our SNE model, $K = 2$ or $K = 3$ is sufficient to achieve better performance (offering some experimental evidence to back our choice of only a few steps of iterative refinement). This is much improved over the results of earlier work [5]. Note that our results are either the best or comparable to GOOG and E2E, while, notably, our system is much simpler in terms of

Table 1: PSNR of the SNE-RNN-JP2 on the Kodak dataset (bitrate 0.3701 bpp) as a function of K.

	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 4$	$K = 6$
PSNR	28.9992	29.2221	29.3007	29.3006	28.9974	28.8814

Table 2: Out-of-sample results for the Kodak (bpp 0.37), the 8-bit Compression Benchmark (CB, bpp, 0.341), the 16-bit and 16-bit-Linear Compression Benchmark (CB) datasets (bpp 0.35 for both), the Tecnick (bpp 0.475), and Wikipedia (bpp 0.352) datasets.

	Kodak			**CB 8-Bit**		
Model	**PSNR**	**SSIM**	MS^3IM	**PSNR**	**SSIM**	MS^3IM
JPEG	27.6540	0.7733	0.9291	27.5481	0.8330	0.9383
JPEG 2000	27.8370	0.8396	0.9440	27.7965	0.8362	0.9471
GOOG-JPEG (**Neural**)	27.9613	0.8017	0.9557	27.8458	0.8396	0.9562
MLP-JPEG	27.8325	0.8399	0.9444	27.8089	0.8371	0.9475
Δ-*RNN*-JPEG	28.5093	0.8411	0.9487	28.0461	0.8403	0.9535
GRU-JPEG	28.5081	0.8400	0.9474	28.0446	0.8379	0.9533
LSTM-JPEG	28.5247	0.8409	0.9486	28.0461	0.8371	0.9532
LSTM-JP2 (**Hybrid**)	28.9321	0.8425	0.9596	28.0896	0.8389	0.9562
E2E (**Neural**)	28.9420	0.8502	0.9600	28.0999	0.8396	0.9562
SNE-RNN-SkipE-JP2 (**Ours**)	28.9991	0.8500	0.9602	28.2108	0.8399	0.9588
SNE-RNN-SkipY-JP2 (**Ours**)	28.2210	0.8488	0.9599	28.1001	0.8377	0.9577
SNE-RNN-SkipB-JP2 (**Ours**)	28.8999	0.8400	0.9599	28.0888	0.8388	0.9566
SNE-RNN-JP2 (**Ours**)	**29.3008**	**0.8508**	**0.9622**	**28.2199**	**0.8401**	**0.9600**
	CB 16-Bit			**CB 16-Bit-Linear**		
JPEG	27.5368	0.8331	0.9383	31.7522	0.8355	0.9455
JPEG 2000	27.7885	0.8391	0.9437	32.0270	0.8357	0.9471
GOOG (**Neural**)	27.8830	0.8391	0.9468	32.1275	0.8369	0.9533
MLP-JPEG	27.7762	0.8390	0.9438	32.0269	0.8356	0.9454
Δ-*RNN*-JPEG	28.0093	0.8399	0.9471	32.4038	0.8403	0.9535
GRU-JPEG	28.0081	0.8392	0.9469	32.4038	0.8379	0.9533
LSTM-JPEG	28.0247	0.8310	0.9471	32.4032	0.8371	0.9532
LSTM-JP2 (**Hybrid**)	28.1307	0.8425	0.9496	32.4998	0.8382	0.9541
E2E (**Neural**)	28.2440	0.8426	0.9498	32.5010	0.8387	0.9540
SNE-RNN-SkipE-JP2 (**Ours**)	28.2441	0.8426	0.9499	32.5000	0.8381	0.9541
SNE-RNN-SkipY-JP2 (**Ours**)	28.1007	0.8422	0.9492	32.4992	0.8382	0.9542
SNE-RNN-SkipB-JP2 (**Ours**)	28.1006	0.8412	0.9491	32.4981	0.8381	0.9538
SNE-RNN-JP2 (**Ours**)	**29.4471**	**0.8430**	**0.9510**	**32.6019**	**0.8399**	**0.9559**
	Tecnick			**Wikipedia**		
JPEG	30.7377	0.8682	0.9521	28.7724	0.8290	0.9435
JPEG 2000	31.2319	0.8747	0.9569	29.1545	0.8382	0.9495
GOOG (**Neural**)	31.5030	0.8814	0.9608	29.2209	0.8406	0.9520
MLP-JPEG	31.2287	0.8746	0.9571	29.1547	0.8383	0.9497
Δ-*RNN*-JPEG	31.5411	0.8821	0.9609	29.2772	0.8403	0.9519
LSTM-JPEG	31.5616	0.8820	0.9609	29.2771	0.8403	0.9519
LSTM-JP2 (**Hybrid**)	31.6962	0.8834	0.9619	29.3228	0.8411	0.9526
E2E (**Neural**)	31.7000	0.8836	0.9620	29.3227	0.8412	0.9526
SNE-RNN-SkipE-JP2 (**Ours**)	31.6999	0.8835	0.9621	29.3226	0.8413	0.9525
SNE-RNN-SkipY-JP2 (**Ours**)	31.6950	0.8830	0.9600	29.3221	0.8402	0.9524
SNE-RNN-SkipB-JP2 (**Ours**)	31.6944	0.8822	0.9589	29.3001	0.8345	0.9501
SNE-RNN-JP2 (**Ours**)	**32.7124**	**0.8841**	**0.9622**	**29.9334**	**0.8413**	**0.9528**

model complexity, i.e., number of parameters, and requires significantly less training time.

4 Conclusions

We proposed a novel neural-based system for lossy image compression called the sibling neural estimator (SNE) model. Variants of our proposed system within the *iterative refinement* framework consistently improves over not only the original compression results but also consistently outperforms a majority of baselines in terms of several metrics. In addition, our system offers comparable visual quality even at low bit rates and yields a better rate-distortion trade-off using only a small number of iterative refinement steps. It is important to note that our proposed decoder estimator is quite general and would work with any encoder. Future work could entail integrating a neural encoder[3, 1] with our SNE decoder.

References

[1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[2] Oren Rippel and Lubomir D. Bourdev, "Real-time adaptive image compression," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 2922–2930.

[3] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, "Full resolution image compression with recurrent neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5435–5443.

[4] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2017, NIPS'17, p. 1141–1151, Curran Associates Inc.

[5] A. G. Ororbia, A. Mali, J. Wu, S. O'Connell, W. Dreese, D. Miller, and C. L. Giles, "Learned neural iterative decoding for lossy image compression systems," in *2019 Data Compression Conference (DCC)*, March 2019, pp. 3–12.

[6] S. Takamura and M. Takagi, "Lossless image compression with lossy image using adaptive prediction and arithmetic coding," in *Proceedings of IEEE Data Compression Conference (DCC'94)*, March 1994, pp. 166–174.

[7] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, "Variable rate image compression with recurrent neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[8] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4385–4393.

[9] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 1747–1756.

[10] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra, "Towards conceptual compression," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., pp. 3549–3557. Curran Associates, Inc., 2016.

[11] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, “Learning image and video compression through spatial-temporal energy compaction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10071–10080.

[12] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Energy compaction-based image compression using convolutional autoencoder,” *IEEE Transactions on Multimedia*, pp. 1–1, 2019.

[13] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár, “Lossy image compression with compressive autoencoders,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[14] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, “Variational image compression with a scale hyperprior,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[15] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool, “Soft-to-hard vector quantization for end-to-end learning compressible representations,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1141–1151.

[16] S. Li, Z. Zheng, W. Dai, and H. Xiong, “Lossy image compression with filter bank based convolutional networks,” in *2019 Data Compression Conference (DCC)*, March 2019, pp. 23–32.

[17] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[18] Ankur Mali, Alexander Ororbia, and C. Lee Giles, “The neural state pushdown automata,” *CoRR*, vol. abs/1909.05233, 2019.

[19] Víctor Campos, Brendan Jou, Xavier Gir’o i Nieto, Jordi Torres, and Shih-Fu Chang, “Skip RNN: learning to skip state updates in recurrent neural networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[20] Kede Ma, Qingbo Wu, Zhou Wang, Zhengfang Duanmu, Hongwei Yong, Hongliang Li, and Lei Zhang, “Group MAD competition? A new methodology to compare objective image quality models,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 1664–1673.

[21] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.