Insights From Generative Modeling for Neural Video Compression

Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt, *Member, IEEE*

***Abstract*—While recent machine learning research has revealed connections between deep generative models such as VAEs and rate-distortion losses used in learned compression, most of this work has focused on images. In a similar spirit, we view recently proposed neural video coding algorithms through the lens of deep autoregressive and latent variable modeling. We present these codecs as instances of a generalized stochastic temporal autoregressive transform, and propose new avenues for further improvements inspired by normalizing flows and structured priors. We propose several architectures that yield state-of-the-art video compression performance on high-resolution video and discuss their tradeoffs and ablations. In particular, we propose (i) improved temporal autoregressive transforms, (ii) improved entropy models with structured and temporal dependencies, and (iii) variable bitrate versions of our algorithms. Since our improvements are compatible with a large class of existing models, we provide further evidence that the generative modeling viewpoint can advance the neural video coding field.**

***Index Terms*—Autoregressive models, generative models, normalizing flow, variational inference, video compression.**

I. Introduction

NEURAL data compression [1] has evolved to become a promising new application and testing domain for generative modeling.[1] Generative models such as hierarchical variational autoencoders have already demonstrated empirical improvements in image compression, outperforming classical codecs [2], [3] such as BPG [4]. For neural video compression, progress has proved harder due to complex temporal dependencies operating at multiple scales. Nevertheless, recent neural video codecs have shown promising performance gains [5], in some cases on par with current hand-designed, classical codecs, e.g., HEVC [6], [7]. Compared to hand-designed codecs, learnable codecs are not limited to a specific data modality and offer a promising approach for streaming specialized content, such as sports or video conferencing. Therefore, improving neural video compression is vital for dealing with the ever-growing amount of video content being created.

The common approach to source compression transforms the data into a white noise distribution that can be more easily modeled and compressed with an entropy model. This way, data compression fundamentally involves decorrelation. Improving a model's capability to decorrelate data helps improves its compression performance. On the other hand, we can improve the entropy model (i.e., the model's prior) to capture any remaining dependencies. Just as compression techniques attempt to *remove* structure, generative models attempt to *generate* structure. For example, autoregressive flows map between less structured distributions, e.g., uncorrelated noise, and more structured distributions, e.g., images or video [8], [9]. The inverse flow can remove dependencies in the data, making it more amenable to compression. Thus, a natural question to ask is how autoregressive flows can best be utilized in compression and if mechanisms in existing compression schemes can be interpreted as normalizing flows.

This paper draws on recent insights in hierarchical sequential latent variable models with autoregressive flows [10]. In particular, we identify connections between this family of models and recent neural video codecs based on motion estimation [5], [11], [12], [13], [14], [15]. By interpreting this technique as an instantiation of a more general autoregressive flow transform, we propose various alternatives and improvements based on insights from generative modeling. Specifically, our contributions are as follows:

First, we interpret existing video compression methods through the framework of generative modeling and variational inference, allowing us to readily investigate extensions and ablations. In particular, we discuss the relationship between sequence modeling and sequence compression. We identify autoregressive transform as a key component in both cases and suggest incorporating it in a sequential latent variable as the basis of our approach.

Second, we improve a popular model for neural video compression, Scale-Space Flow (SSF) [5]. This model uses motion estimation to predict the frame being compressed and further compresses the residual obtained by subtraction. Our proposed model extends the SSF model with a more flexible decoder and prior, and obtains improved rate-distortion performance. Specifically, we incorporate a learnable scaling transform to

Manuscript received 20 July 2021; revised 26 January 2023; accepted 13 March 2023. Date of publication 22 March 2023; date of current version 30 June 2023. The work of Yibo Yang was supported by the HPI Research Center in Machine Learning and Data Science at UC Irvine. The work of Stephan Mandt was supported in part by the National Science Foundation (NSF) under NSF CAREER Award under Grant 2047418, NSF under Grants 2003237 and 2007719, in part by the Department of Energy under Grant DE-SC0022331, and IN PART by gifts from Qualcomm and Disney. Recommended for acceptance by T. Tran *(Corresponding author: Ruihan Yang.)*

Ruihan Yang, Yibo Yang, and Stephan Mandt are with the Department of Computer Science, University of California Irvine, Irvine, CA 92697 USA (e-mail: ruihan.yang@uci.edu; yibo.yang@uci.edu; mandt@uci.edu).

Joseph Marino is with the DeepMind, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jmarino@caltech.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3260684, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3260684

[1]Proceedings of the First ICLR Workshop on Neural Compression: From Information Theory to Applications, neuralcompression.github.io

allow for more expressive and accurate reconstruction. Augmenting a shift transform by scale-then-shift is inspired by the extension of NICE [8] to RealNVP [9]. We also compare motion-estimation-based versus purely CNN-based approaches to predictive coding.

Third, our probabilistic perspective allows us to explore improved entropy models for SSF and its relatives. In particular, we explore structured priors that jointly encode motion and residual information. As the two tend to be spatially correlated, encoding residual information conditioned on motion information results in a better prior. Since the residual dominates the bitrate, our improved entropy model reduces the overall bitrate significantly. We also investigate different versions of temporal priors, where we either condition on latent variables or on frame reconstructions and discuss their tradeoffs in terms of bit savings and optimization challenges.

Finally, also from the perspective of generative modeling, we present variable bitrate versions of our models, i.e., training a single encoder and decoder that works at different points on the rate-distortion curves. This step is considered important for making neural coding schemes practical. Our experiments show that the difference in rate-distortion performance of variable bitrate models and models tuned to individual bit rates depends on the model complexity.

Our paper is structured as follows. We establish our viewpoint and model improvements in Section II, discuss related work in Section III, and present experiments in Section IV. Conclusions are provided in Section V.

II. Video Compression Through Deep Autoregressive Modeling

We first review low-latency video compression, including the classical predictive coding technique. We then draw connections between data compression and data modeling with a Masked Autoregressive Flow (MAF) [16], a generative model based on a temporal autoregressive transform that resembles predictive coding. Finally, inspired by hierarchical autoregressive flow models [10], we combine the strength of autoregressive modeling with the end-to-end optimizable transform coding approach of VAEs [17], resulting in a sequential VAE [18] with an autoregressive encoding/decoding transform. The resulting model captures many existing neural video compression methods [5], [11], [12], [13], [14], [15], and serves as the basis of our proposed improvements to the decoding transform as well as the prior model.

Notation. We use bold letters (e.g., $\mathbf{x}, \mathbf{z}$) to denote random variables and variables with superscripts to indicate deterministically computed quantities (e.g., $\hat{\mathbf{x}}, \bar{\mathbf{z}}$). We use $p(\mathbf{x})$ to denote the probability distribution or density induced by random variable $\mathbf{x}$ and write $P(\mathbf{x})$ to emphasize when it is a probability mass function.

A. Background

As follows, we review lossy video compression, focusing on the low-latency online compression setting. We then review the classical technique of predictive coding, which provides the high-level algorithmic framework of many video compression methods, including ours. As our final building block, we review normalizing flow models, in particular the Masked Autoregressive Flow for sequence modeling.

Video Compression. Given a typical sequence of video frames $\mathbf{x}_{1:T}$, lossy video compression ultimately aims to find a short bitstring description of $\mathbf{x}_{1:T}$, from which a faithful (but not necessarily perfect) reconstruction $\hat{\mathbf{x}}_{1:T}$ can be recovered. Denoting the description length ("rate") by $\mathcal{R}$ and the reconstruction error ("distortion," often the Mean-Squared Error) by $\mathcal{D}$, lossy video compression aims to minimize the rate-distortion (R-D) objective function,

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\text{data}}}[\mathcal{D}(\mathbf{x}_{1:T}, \hat{\mathbf{x}}_{1:T}) + \beta \mathcal{R}(\hat{\mathbf{x}}_{1:T})], \tag{1}$$

where $\beta > 0$ is a hyperparameter weighing the two costs, and the expectation is with respect to the source data distribution p_{data}, which is approximated by averaging over a training set of videos in practice. One simple approach is to compress each frame $\mathbf{x}_t$ separately using an image compression algorithm, which can exploit the spatial redundancy between the pixels within each frame. However, a key feature distinguishing video from image compression is the significant amount of temporal redundancy *between* frames that can be exploited to improve the compression rate.

Online Video Compression by Predictive Coding. In theory, the optimal rate-distortion performance is achieved by compressing exceedingly long blocks of frames together [19]. However, such an approach is rarely implemented in practice because of its prohibitive computation expense and the latency caused by buffering the frames into long blocks. We consider video compression in the sequential/online setting, widely used in both conventional and recent neural codecs [5], [20] and are suitable for real-time applications such as video conferencing and live streaming. In this setting, each video frame $\mathbf{x}_t$ is compressed in temporal order so that at any time $t < T$, the source frames up to time t, i.e., $\mathbf{x}_{1:t}$, are encoded and transmitted, and similarly the reconstructed frames up to time t, $\hat{\mathbf{x}}_{1:t}$, are available to the receiver. Since past frames are often highly indicative of future frames, the basic idea for exploiting this temporal redundancy is to use knowledge of the previous frame reconstructions, $\hat{\mathbf{x}}_{<t}$, to aid the compression of the current frame $\mathbf{x}_t$. Indeed, the earliest video codecs are based on transmitting frame differences $\mathbf{x}_t - \hat{\mathbf{x}}_{t-1}$ [21], analogous to the classic modeling technique in dynamical systems whereby the state-space becomes first-order Markov when redefined in terms of temporal changes [10], [22]. This technique is further refined by *predictive coding* [23], where instead of simply using the previous reconstructed frame $\hat{\mathbf{x}}_{t-1}$, a *motion-compensated prediction* of the current frame, $\bar{\mathbf{x}}_t$, is computed, and the residual $\mathbf{x}_t - \bar{\mathbf{x}}_t$ is transmitted instead.

In more detail, the idea of predictive coding is typically implemented in traditional video codecs as follows: *1. motion estimation*: The encoder estimates the motion vector $\mathbf{m}_t$ between the current frame $\mathbf{x}_t$ and the previous reconstruction $\hat{\mathbf{x}}_{t-1}$; *2. motion compensation*: The encoder simulates the coding of $\mathbf{m}_t$, and uses the reconstruction $\hat{\mathbf{m}}_t$ (as would be received by the decoder in step 3) to transform the previous frame reconstruction $\hat{\mathbf{x}}_{t-1}$ into a prediction of the current frame $\bar{\mathbf{x}}_t$. Conceptually,

this is done by shifting the pixels of $\hat{\mathbf{x}}_{t-1}$ according to the motion vector $\hat{\mathbf{m}}_t$. The compression of $\mathbf{m}_t$ could either be lossy ($\hat{\mathbf{m}}_t \approx \mathbf{m}_t$) or lossless ($\hat{\mathbf{m}}_t = \mathbf{m}_t$). *3. residual compression:* The residual is computed as $\mathbf{r}_t = \mathbf{x}_t - \bar{\mathbf{x}}_t$, and is encoded. The decoder, upon receiving the bit-streams for $(\mathbf{m}_t, \mathbf{r}_t)$ and reconstructing them as $(\hat{\mathbf{m}}_t, \hat{\mathbf{r}}_t)$, computes the prediction $\bar{\mathbf{x}}_t$ as in step 2, and finally the reconstruction of the current frame, $\hat{\mathbf{x}}_t = \bar{\mathbf{x}}_t + \hat{\mathbf{r}}_t$. The current reconstruction $\hat{\mathbf{x}}_t$ then serves as the reference frame in the predictive coding of the next frame $\mathbf{x}_{t+1}$.

Masked Autoregressive Flow (MAF). MAF[2] [16] is a type of normalizing flow [24] that models the distribution of a random sequence $p(\mathbf{x}_{1:T})$ in terms of a simpler distribution $p(\mathbf{y}_{1:T})$ of its underlying noise variables $\mathbf{y}_{1:T}$. The two variables are related by the following invertible autoregressive transform,

$$\mathbf{y}_t = \frac{\mathbf{x}_t - h_\mu(\mathbf{x}_{<t})}{h_\sigma(\mathbf{x}_{<t})} \;\Leftrightarrow\; \mathbf{x}_t = h_\mu(\mathbf{x}_{<t}) + h_\sigma(\mathbf{x}_{<t}) \odot \mathbf{y}_t, \quad (2)$$

for $t = 1, 2, \ldots, T$. Here, $\odot$ denotes element-wise multiplication, $\mathbf{x}_{<t}$ denotes all frames up to time t, and h_μ and h_σ are two deterministic neural network mappings.[3] The base distribution $p(\mathbf{y}_{1:T})$ is typically fixed to be a simple factorized distribution such as an isotropic Gaussian, and is related to the distribution $p(\mathbf{x}_{1:T})$ through the standard change-of-variable formula between probability densities. While the forward MAF transform ($\mathbf{y}_{1:T} \to \mathbf{x}_{1:T}$) converts a sequence of standard normal noise variables into a data sequence, the inverse transform ($\mathbf{x}_{1:T} \to \mathbf{y}_{1:T}$) removes temporal correlations and "normalizes" the data sequence.

Although originally proposed for modeling static data (e.g., still images interpreted as a sequence of pixels), MAF can be applied along the temporal dimension of sequential data and is shown to improve video modeling performance [10]. This motivates us to consider the potential of MAF for sequential data compression.

B. On the Relation Between MAF and Sequence Compression

In this section, we identify the commonality and difference between sequence modeling with MAF and sequence compression with predictive coding. On the one hand, MAF implements a core idea of decorrelation in transform coding, and the autoregressive transform underlying MAF resembles and generalizes that of traditional predictive coding. On the other hand, MAF does not consider the quantization and reconstruction error of lossy compression and is, therefore, not directly suitable for compression. Motivated by these two aspects, we will later on consider the model family of Variational Autoencoders (VAEs) [25] that is more suited for compression (Section II-C) but reintroduce the MAF-style autoregressive transform into the encoding and decoding procedure of the VAE (Section II-D), resulting in a hybrid model that forms the basis of our proposal.

We begin by drawing conceptual connections between MAF and transform coding, the predominant paradigm of lossy compression. In transform coding, the data is first transformed to the domain of transform coefficients via a function $G : \mathbf{x} \to \mathbf{y}$, and the resulting coefficients $\mathbf{y}$ are compressed after scalar quantization. Although this two-stage approach is theoretically suboptimal compared to vector quantization [26], it has considerably lower computational complexity and is the default approach used in modern media compression algorithms [27]. Conventionally, the transform G is chosen to be orthogonal, and the rate-distortion-optimal transform is often characterized by its ability to *decorrelate* the transform coefficients, i.e., $\mathrm{cov}(\mathbf{y}_i, \mathbf{y}_j) = 0$, for $i \neq j$ [27]. The idea of decorrelation is also a guiding principle behind data modeling with MAF or normalizing flow in general. Specifically, training a normalizing flow is, in fact, equivalent to decorrelating and "normalizing" the data distribution into a simple base distribution. Consider a flow model $p(\mathbf{x})$ in the data space induced by passing a noise base distribution $p(\mathbf{y})$ through a (forward) flow transform $F : \mathbf{y} \to \mathbf{x}$. If $p_{\text{data}}(\mathbf{x})$ is the true data-generating distribution, then the following relation holds [16],

$$\mathrm{KL}\left[p_{\text{data}}(\mathbf{x}) \| p(\mathbf{x})\right] = \mathrm{KL}\left[F^{-1}(p_{\text{data}}(\mathbf{x})) \| p(\mathbf{y})\right], \quad (3)$$

where $F^{-1}(p_{\text{data}}(\mathbf{x}))$ denotes the distribution that $p_{\text{data}}(\mathbf{x})$ would follow when passed through the inverse transform F^{-1}. In other words, training a flow model by maximum-likelihood is equivalent to fitting the "normalized" data distribution" $F^{-1}(p_{\text{data}}(\mathbf{x}))$ to the base distribution $p(\mathbf{y})$. The connection to transform coding is then clear: the normalizing transform F^{-1} plays a similar role to the transform G, $F^{-1}(p_{\text{data}}(\mathbf{x}))$ is the empirical distribution of "transform coefficients" $\mathbf{y}$, and $p(\mathbf{y})$ corresponds to the factorized entropy model in transform coding.

Moreover, the autoregressive transform used by MAF is closely related to that of predictive coding and generalizes the latter. We can view predictive coding as implementing an autoregressive transform between the residual sequence and data sequence

$$\mathbf{y}_t = \mathbf{x}_t - h_\mu(\hat{\mathbf{x}}_{t-1}); \quad \texttt{// encode;} \quad (4)$$

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}) + \hat{\mathbf{y}}_t; \quad \texttt{// decode,} \quad (5)$$

for $t = 2, \ldots, T$.[4] In the language of predictive coding from Section II-A, $h_\mu(\hat{\mathbf{x}}_{t-1}) = \bar{\mathbf{x}}_t$ is the prediction of the current frame, and $\mathbf{y}_t$ is the residual $\mathbf{r}_t$. The resulting transform can be seen as a special case of the more general shift-and-scale transform used by MAF in (2), if we fix $h_\sigma \equiv 1$ and limit $\hat{\mathbf{x}}_{<t}$ to only the most recent (reconstructed) frame.

We now discuss the difference between sequence modeling and compression and the reasons why MAF may not be directly suitable for lossy compression. In our discussion above, we have left out the issue of quantization. Unlike in sequence modeling, in lossy compression, the noise variable $\mathbf{y}_t$ must be communicated to the decoder in a lossy manner, e.g., via quantization

[2] As a clarification, even though the original MAF was implemented with the masking approach of MADEs [16] (hence "masked" in the name), we use the term "MAF" to refer more broadly to the normalizing flow defined by the temporal autoregressive transform in (2).

[3] In general, (h_μ, h_σ) can be a different pair of neural networks for each time step t, although for practical sequence modeling, they are often shared across time and only receive a fixed length-k context $x_{(t-k+1):t}$ as input. In the special case $t = 1$, the networks receive no input and reduce to two learnable parameters.

[4] At time $t = 1$, $\mathbf{x}_1$ is compressed and reconstructed as $\hat{\mathbf{x}}_1$ separately without reference to any context frame.

$\hat{\mathbf{y}}_t = Q(\mathbf{y}_t)$ followed by (lossless) entropy coding. Similarly, we no longer have access to the history of ground truth frames $\mathbf{x}_{<t}$, but only their lossy reconstructions $\hat{\mathbf{x}}_{<t}$. Taking this into account, it is possible to construct a transform coding procedure using a learned MAF transform: 1. apply the learned (h_μ, h_σ) networks to compute the noise $\mathbf{y}_t$ via

$$\mathbf{y}_t = \frac{\mathbf{x}_t - h_\mu(\hat{\mathbf{x}}_{t-1})}{h_\sigma(\hat{\mathbf{x}}_{t-1})}, \tag{6}$$

similar to (4); 2. quantize the noise to $\hat{\mathbf{y}}_t = Q(\mathbf{y}_t)$; 3. compute the reconstruction $\hat{\mathbf{x}}_t$ via

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}) + h_\sigma(\hat{\mathbf{x}}_{t-1}) \odot \hat{\mathbf{y}}_t, \tag{7}$$

similar to (5), and finally, 4. iterate the above over time steps $t = 1, 2, \ldots, T$. Unfortunately, this simple transform coding procedure comes with a few practical drawbacks and is generally suboptimal in terms of rate-distortion performance. First, since the flow transform F is only trained to maximize the data likelihood, the resulting reconstruction error $\mathcal{D}(\mathbf{x}_{1:T}, \hat{\mathbf{x}}_{1:T})$ due to quantization is uncontrolled. Empirically, trained flow transforms are often close to singular and suffer numerical stability issues [28], resulting in large reconstruction error $\mathcal{D}(\mathbf{x}, F((F^{-1}(\mathbf{x}))))$ even without the quantization step (despite F being invertible in theory). Second, the invertibility of the transform F places restrictions on the kinds of computation allowed. This often necessitates deeper and more expensive neural network transforms to achieve similar expressivity to unconstrained neural network transforms. State-of-the-art normalizing flow models such as GLOW [29] often require a deep stack of bijective transforms and are computationally much more expensive than comparable VAEs or GANs, making them potentially less suited for real-time video transmission applications.

Lastly, we do note, however, that a connection exists between density modeling and the *lossless* compression of quantized data through the technique of dequantization [30], and the latter has been used extensively in training and evaluating normalizing flows. Moreover, under fine quantization, the negative log density under a normalizing flow model can be recovered as the NELBO of a particular latent variable model, allowing bits-back coding to be applied for lossless compression [31].

C. Latent Variable Models for Learned Sequence Compression

We can overcome the suboptimality of normalizing flows for lossy compression by switching to another class of generative models. In this section, we motivate sequence compression with latent variable models, particularly VAEs, that can be trained to perform nonlinear transform coding [17] and optimize for rate-distortion performance in an end-to-end manner. From this generative modeling perspective, we give a detailed account of the probabilistic structure of the sequential latent variable model for learned online video compression, in particular, the correspondence between the data compression process and inference-generative process.

To motivate this approach, consider transform coding with a pair of flexible (but not necessarily invertible) transforms f ("encoder") and g ("decoder") that map between the video $\mathbf{x}_{1:T}$ and its transformed representation $\bar{\mathbf{z}}_{1:T} = f(\mathbf{x}_{1:T})$. Given sufficiently expressive f and g, quantization can be performed by element-wise rounding to the nearest integer, $\lfloor \cdot \rceil$, resulting in the following R-D objective,

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\text{data}}} \left[\mathcal{D}\left(\mathbf{x}_{1:T}, g\left(\lfloor f(\mathbf{x}_{1:T}) \rceil\right)\right) + \beta \mathcal{R}\left(\lfloor f(\mathbf{x}_{1:T}) \rceil\right)\right]. \tag{8}$$

Following [32], we approximate the rounding operations by uniform noise injection to enable gradient-based optimization during training. The resulting, relaxed version of the above R-D objective can be shown to be equal to the expected Negative Evidence Lower BOund (NELBO) of a particular *compressive VAE* model, described below [32], [33]

$$\tilde{\mathcal{L}} = \mathbb{E}_{\mathbf{x}_{1:T} \sim p_{\text{data}}} \left[\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})} \left[-\log p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) - \log p(\mathbf{z}_{1:T}) \right]\right]. \tag{9}$$

In this compressive VAE model, the noisy quantization, $\lfloor \bar{\mathbf{z}}_{1:T} \rceil \approx \bar{\mathbf{z}}_{1:T} + \mathbf{u}_{1:T}, \mathbf{u}_{1:T} \sim \mathcal{U}(-\mathbf{0.5}, \mathbf{0.5})$, is equivalently obtained by sampling from a particular variational posterior distribution $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \mathcal{U}(\bar{\mathbf{z}}_{1:T} - 0.5, \bar{\mathbf{z}}_{1:T} + 0.5)$, i.e., a unit-width uniform distribution whose mean $\bar{\mathbf{z}}_{1:T}$ is predicted by an amortized inference network f. The likelihood $p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$ follows a Gaussian distribution with fixed diagonal covariance $\frac{\beta}{2 \log 2}\mathbf{I}$, and mean $\hat{\mathbf{x}}_{1:T} = g(\mathbf{z}_{1:T})$ computed by the generative network g, so that the negative log-likelihood $-\log p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$ equals the squared error distortion $\frac{1}{\beta}\|\mathbf{x}_{1:T} - \hat{\mathbf{x}}_{1:T}\|^2$ weighted by $\frac{1}{\beta}$. Following [32], the prior density $p(\mathbf{z}_{1:T})$ is parameterized to interpolate its discretized version $P(\mathbf{z}_{1:T})$, so that given integer valued $\mathbf{z}_{1:T}$, the negative log density $-\log p(\mathbf{z}_{1:T})$ measures the code length $-\log P(\mathbf{z}_{1:T})$ assigned by the entropy model $P(\mathbf{z}_{1:T})$. By minimizing the approximate R-D objective of (9) with respect to the parameters of f, g, and $p(\mathbf{z}_{1:T})$, training an end-to-end neural compression model is thus equivalent to learning a VAE by maximum-likelihood and amortized variational inference. Note this is in contrast to the maximum-likelihood estimation of a normalizing flow model, which does not account for the distortion of lossy compression and results in suboptimal rate-distortion performance (as discussed in Section II-B).

Given a compressive VAE, the compression of a data sequence $\mathbf{x}_{1:T}$ via transform coding closely corresponds to an inference-generation pass through the VAE, described in the following steps. *1. encoding:* The encoder passes $\mathbf{x}_{1:T}$ through f to obtain a transformed representation $\bar{\mathbf{z}}_{1:T} = f(\mathbf{x}_{1:T})$, thus computing the mean parameters $\bar{\mathbf{z}}_{1:T}$ of the variational distribution $q(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$ by amortized variational inference; *2. quantization and entropy coding:* A posterior sample $\mathbf{z}_{1:T}$ is drawn, i.e., $\mathbf{z}_{1:T} \sim q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ (or, deterministically computed as $\mathbf{z}_{1:T} = \lfloor \bar{\mathbf{z}}_{1:T} \rceil$ at test time), and a bit-string encoding $\boldsymbol{\xi}$ of $\mathbf{z}_{1:T}$ is transmitted under the entropy model $P(\mathbf{z}_{1:T})$; *3. decoding:* The decoder decodes $\mathbf{z}_{1:T}$ from bit-string $\boldsymbol{\xi}$ using the entropy model $P(\mathbf{z}_{1:T})$, then computes the reconstruction by $\hat{\mathbf{x}}_{1:T} = g(\mathbf{z}_{1:T})$, corresponding to the mean parameters of the Gaussian likelihood model $p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$. Note that step 3 (decoding) is analogous to sampling from the generative model, but without adding diagonal Gaussian noise to $\hat{\mathbf{x}}_{1:T}$ dictated by the Gaussian likelihood $p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$. Indeed, if the bitstring $\boldsymbol{\xi}$ consists of a sequence of purely random bits, then it is well known that decoding $\boldsymbol{\xi}$ under the entropy model $P(\mathbf{z}_{1:T})$

produces a sample $\mathbf{z}_{1:T} \sim P(\mathbf{z}_{1:T})$ [19]. For this reason, in the following discussions, we occasionally blur the distinction between the (random) latent variable $\mathbf{z}_{1:T} \sim P(\mathbf{z}_{1:T})$ and the quantized latent representation $\lfloor \bar{\mathbf{z}}_{1:T} \rceil$ (as would be decoded from a bitstring $\boldsymbol{\xi}$) to simplify notation.

A data compression process with such a learned transform coding algorithm also implicitly defines a VAE model for the data. Specifically, let us consider the compression procedure of an online video compression codec, in which individual frames $\mathbf{x}_t$ are transmitted sequentially. The encoding-decoding process is specified recursively as follows. Given the ground truth current frame $\mathbf{x}_t$ and the previously reconstructed frames $\hat{\mathbf{x}}_{<t}$, the encoder is restricted to be of the form $\bar{\mathbf{z}}_t = f(\mathbf{x}_t, \hat{\mathbf{x}}_{<t})$, and similarly the decoder computes the reconstruction sequentially based on previous reconstructions and the current encoding, $\hat{\mathbf{x}}_t = g\left(\hat{\mathbf{x}}_{<t}, \lfloor \bar{\mathbf{z}}_t \rceil\right)$). Existing codecs usually condition on a *single* reconstructed frame, substituting $\hat{\mathbf{x}}_{<t}$ by $\hat{\mathbf{x}}_{t-1}$ in favor of efficiency. In the language of generative modeling and variational inference, the sequential encoder corresponds to a variational posterior of the form $q(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{<t})$, i.e., filtering, and the sequential decoder corresponds to the likelihood $p(\mathbf{x}_t|\mathbf{z}_{\leq t}) = \mathcal{N}\left(\hat{\mathbf{x}}_t, \frac{\beta}{2\log 2}\mathbf{I}\right)$; in both distributions, the probabilistic conditioning on $\mathbf{z}_{<t}$ is based on the observation that $\hat{\mathbf{x}}_{t-1}$ is a deterministic function of $\mathbf{z}_{<t}$, if we identify $\lfloor \bar{\mathbf{z}}_t \rceil$ with the random variable $\mathbf{z}_t$ and unroll the recurrence $\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{<t}, \mathbf{z}_t)$. As we show, all sequential compression approaches considered in this work follow this paradigm and implicitly define generative models of the data as $p(\mathbf{x}_{1:T}) \propto p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})p(\mathbf{z}_{1:T}) = p(\mathbf{z}_{1:T}) \prod_t p(\mathbf{x}_t|\mathbf{z}_{\leq t})$, where the likelihood model $p(\mathbf{x}_t|\mathbf{z}_{\leq t})$ at each time step is centered on the reconstruction $\hat{\mathbf{x}}_t$. The key difference is in the definition of the decoding transform for computing $\hat{\mathbf{x}}_t$ as a (stochastic) function of $\hat{\mathbf{x}}_{<t}$ and $\mathbf{z}_t$.

D. Hybrid Model With Stochastic Temporal Autoregressive Transform

Having laid out the general approach for end-to-end learned video compression with sequential VAEs, we specify the per-frame likelihood model $p(\mathbf{x}_t|\mathbf{z}_{\leq t})$ by revisiting the autoregressive transform of MAF from Section II-B. The resulting model captures several existing low-latency neural compression methods as specific instances [5], [11], [12], [13], [14], [15] and gives rise to the exploration of new models. Consider computing the reconstruction $\hat{\mathbf{x}}_t$ using the forward temporal autoregressive transform of a MAF as in (7),

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{<t}) + h_\sigma(\hat{\mathbf{x}}_{<t}) \odot \hat{\mathbf{y}}_t. \tag{10}$$

As follows, we augment it with a latent variable $\mathbf{z}_t$. We may interpret the reconstructed $\hat{\mathbf{y}}_t$ as being decoded from the random variable $\mathbf{z}_t$ given some prior $p(\mathbf{z}_t)$, and a decoding transform g_z; additionally, $\mathbf{z}_t$ may enter into the computation of the shift h_μ and scale h_σ transforms. By combining a sequential latent variable model with temporal autoregressive transforms, we therefore arrive at the most general form of the proposed *stochastic temporal autoregressive transform*

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{<t}, \mathbf{z}_t) + h_\sigma(\hat{\mathbf{x}}_{<t}, \mathbf{z}_t) \odot g_z(\mathbf{z}_t). \tag{11}$$

In this work, we only consider the common Mean Squared Error (MSE) distortion for simplicity. Therefore the above decoding transform computes the mean of a diagonal Gaussian frame observation model $p(\mathbf{x}_t|\mathbf{z}_{\leq t})$, with a fixed diagonal co-variance parameterized by the desired Lagrange multiplier β (see explanation at the end of Section II-B). We note, however, other kinds of distortion functions such as MS-SSIM are possible, resulting in a different form of $p(\mathbf{x}_t|\mathbf{z}_{\leq t})$ parameterized by $\hat{\mathbf{x}}_t$ [32].

This stochastic decoder model has several advantages over existing generative models for compression, such as simpler normalizing flows or sequential VAEs. First, the stochastic autoregressive transform $h_\mu(\hat{\mathbf{x}}_{<t}, \mathbf{z}_t)$ involves a latent variable $\mathbf{z}_t$ and is therefore more expressive than a deterministic transform [34], [35]. Second, compared to MAF, which directly models $\mathbf{y}_t$, the additional nonlinear transform $g_z(\mathbf{z}_t)$ enables more expressive residual noise, reducing the burden on the prior by allowing it to model a simpler distribution of $\mathbf{z}_t$. Finally, as visualized in the video compression example in Fig. 2, the scale parameter computed by h_σ effectively acts as a gating mechanism, determining how much variance is explained in terms of the autoregressive transform and the residual noise distribution. This provides an added degree of flexibility, in a similar fashion to how RealNVP improves over NICE [8], [9].

Our approach is inspired by [10], who analyzed a restricted version of the model in (11), aiming to hybridize autoregressive flows and sequential latent variable models for video modeling. In contrast to the stochastic transform in (11), the hybrid model in [10] is based on applying a *deterministic* temporal autoregressive transform (as in MAF) to a sequence of residual noise variables $\mathbf{y}_{1:T}$ modeled by a sequential VAE, $p(\mathbf{y}_{1:T}) \propto p(\mathbf{y}_{1:T}|\mathbf{z}_{1:T})p(\mathbf{z}_{1:T})$. The data distribution $p(\mathbf{x}_{1:T})$ under the resulting model (after applying the change-of-variable formula to the density of $p(\mathbf{y}_{1:T})$) does not admit a simple conditional likelihood distribution like $\mathbf{x}_t|\hat{\mathbf{x}}_t \sim \mathcal{N}\left(\hat{\mathbf{x}}_t, \frac{\beta}{2\log 2}\mathbf{I}\right)$, and maximum-likelihood training of $p(\mathbf{x}_{1:T})$ is not directly aligned with the R-D objective of video compression. We note that, however, the learned MAF transform of such a model may be used by a transform coding algorithm in a manner similar to our discussion in Section II-B, but the resulting algorithm is likely to be suboptimal in R-D performance and faces similar issues with decoding.

E. Example Models

Next, we will show that the general framework expressed by (11) captures a variety of state-of-the-art neural video compression schemes and gives rise to extensions and new models.

Temporal Autoregressive Transform (TAT). The first special case among the class of models that are captured by (11) is the autoregressive neural video compression model by Yang et al. [15], which we refer to as temporal autoregressive transform (TAT). Shown in Fig. 1(a), the decoder g implements a deterministic scale-shift autoregressive transform of decoded noise $\hat{\mathbf{y}}_t$,

$$\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{t-1}, \mathbf{z}_t) = h_\mu(\hat{\mathbf{x}}_{t-1}) + h_\sigma(\hat{\mathbf{x}}_{t-1}) \odot \hat{\mathbf{y}}_t, \quad \hat{\mathbf{y}}_t = g_z(\mathbf{z}_t) \tag{12}$$

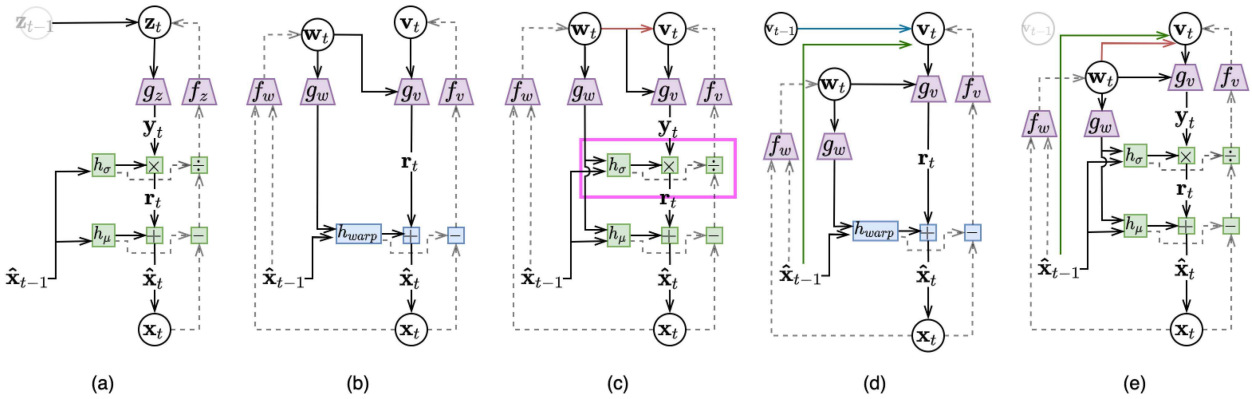


Fig. 1. *Model diagrams* for the generative and inference procedures of the current frame $\mathbf{x}_t$, for various neural video compression methods. Random variables are shown in circles; all other quantities are deterministically computed; solid and dashed arrows describe computational dependency during generation (decoding) and inference (encoding), respectively. Purple nodes correspond to neural encoders (CNNs) and decoders (DCNNs), and green nodes implement temporal autoregressive transform. **(a)** TAT; **(b)** SSF; **(c)** STAT or STAT-SSF; the magenta box highlights the additional proposed scale transform absent in SSF, and the red arrow from $\mathbf{w}_t$ to $\mathbf{v}_t$; highlights the proposed (optional) structured prior. **(d)** SSF-TP/SSF-TP+ and **(e)** STAT-SSF-SP-TP+ illustrate the temporal prior extension based on our proposal; the blue arrow shows the temporal dependency on the previous residual latent $\mathbf{v}_{t-1}$, and the green arrow highlights the improved dependency on the previous reconstructed frame $\hat{\mathbf{x}}_{t-1}$.

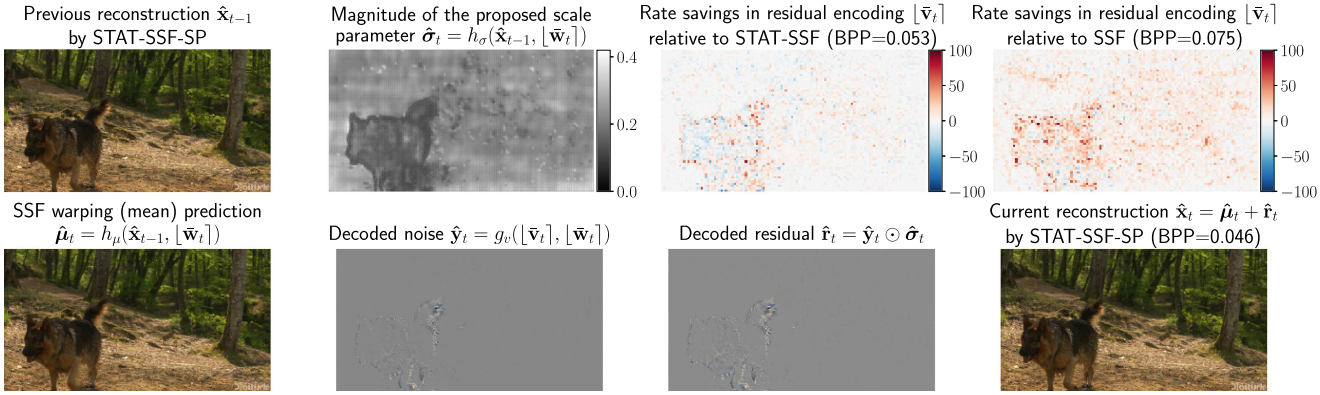


Fig. 2. *Visualizing the encoding/decoding computation of the STAT-SSF-SP model* on one frame of UVG video "Shake-NDry". See Fig. 1(c) for the model's computation diagram. In this example, the warping prediction $\hat{\boldsymbol{\mu}}_t$ (bottom, first) incurs a large error around the dog's moving contour but models the mostly static background well, with the residual latents $\lfloor \bar{\mathbf{v}}_t \rceil$ taking up an order of magnitude higher bit-rate than $\lfloor \bar{\mathbf{w}}_t \rceil$. The proposed scale parameter $\hat{\boldsymbol{\sigma}}_t$ (top, second) gives the model extra flexibility when combining the noise $\hat{\mathbf{y}}_t$ (bottom, second) with the warping prediction $\hat{\boldsymbol{\mu}}_t$ to form the reconstruction $\hat{\mathbf{x}}_t = \hat{\boldsymbol{\mu}}_t + \hat{\boldsymbol{\sigma}}_t \odot \hat{\mathbf{y}}_t$ (bottom, fourth). The scale $\hat{\boldsymbol{\sigma}}_t$ downweights contribution from the noise $\hat{\mathbf{y}}_t$ in the foreground where it is very costly, and reduces the residual bit-rate $\mathcal{R}(\lfloor \bar{\mathbf{v}}_t \rceil)$ (and thus the overall bit-rate) compared to STAT-SSF and SSF, as illustrated in the third and fourth figures in the top row. The (BPP, PSNR) performance for STAT-SSF-SP, STAT-SSF, and SSF [5] are (0.046, 36.97), (0.053, 36.94), and (0.075, 36.97), respectively. Thus, STAT-SSF and SSF here have comparable reconstruction quality to STAT-SSF-SP but worse bit-rate.

The encoder f inverts the transform to decorrelate the input frame $\mathbf{x}_t$ into $\bar{\mathbf{y}}_t$ and encodes the result as $\bar{\mathbf{z}}_t = f(\mathbf{x}_t, \hat{\mathbf{x}}_{t-1}) = f_z(\bar{\mathbf{y}}_t)$, where $\bar{\mathbf{y}}_t = \frac{\mathbf{x}_t - h_\mu(\hat{\mathbf{x}}_{t-1})}{h_\sigma(\hat{\mathbf{x}}_{t-1})}$. The shift h_μ and scale h_σ transforms are parameterized by neural networks, f_z is a convolutional neural network (CNN), and g_z is a deconvolutional neural network (DCNN) that approximately inverts f_z.

The TAT decoder is a simple version of the more general stochastic autoregressive transform in (11), where h_μ and h_σ lack latent variables. Indeed, focusing on the generative process of $\hat{\mathbf{x}}$, TAT implements the model proposed by [10], transforming $\mathbf{y}$ into $\hat{\mathbf{x}}$ by a MAF. However, the generative process underlying lossy compression (see Section II-C) adds additional white noise to $\hat{\mathbf{x}}$, with $\mathbf{x} := \hat{\mathbf{x}} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \frac{\beta}{2 \log 2}\mathbf{I})$. Thus, the generative process from $\mathbf{y}$ to $\mathbf{x}$ is no longer invertible nor corresponds to an autoregressive flow. Nonetheless, TAT was shown to better capture the low-level dynamics of video frames than the autoencoder (f_z, g_z) alone, and the inverse transform decorrelates raw video frames to simplify the input to the encoder f_z [15].

DVC [11], Scale-Space Flow (SSF, [5]), Among Others [12], [13], [14]. The second class of models captured by (11) belong to the neural video compression framework based on predictive coding; both models make use of two sets of latent variables $\mathbf{z}_{1:T} = \{\mathbf{w}_{1:T}, \mathbf{v}_{1:T}\}$ to capture different aspects of information being compressed, where $\mathbf{w}$ captures estimated motion information used in the warping prediction, and $\mathbf{v}$ helps capture residual error not predicted by warping the previous reconstruction frame.

Like most classical approaches to video compression by predictive coding, the reconstruction transform in the above models has the form of a prediction shifted by residual error (decoded noise), and lacks the scaling factor h_σ compared to

the autoregressive transform in (11)

$$\hat{\mathbf{x}}_t = h_{warp}(\hat{\mathbf{x}}_{t-1}, g_w(\mathbf{w}_t)) + g_v(\mathbf{v}_t, \mathbf{w}_t). \tag{13}$$

Above, g_w and g_v are DCNNs, $\mathbf{o}_t := g_w(\mathbf{w}_t)$ has the interpretation of an estimated optical flow (motion) field, h_{warp} denotes warping [36], the h_μ of (11) is defined by the composition $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) := h_{warp}(\hat{\mathbf{x}}_{t-1}, g_w(\mathbf{w}_t))$, and the residual $\mathbf{r}_t := g_v(\mathbf{v}_t, \mathbf{w}_t) = \hat{\mathbf{x}}_t - h_{warp}(\hat{\mathbf{x}}_{t-1}, \mathbf{o}_t)$ represents the prediction error unaccounted for by warping. DVC [11] only makes use of $\mathbf{v}_t$ in the residual decoder g_v, and performs simple 2D warping by bi-linear interpretation; Lin et al. [13] make use of multiple reference frames $\hat{\mathbf{x}}_{(t-3):t}$ for estimating the optimal flow (motion) field; SSF [5] augments the optical flow (motion) field $\mathbf{o}_t$ with an additional scale field, and applies scale-space-warping to the progressively blurred versions of $\hat{\mathbf{x}}_{t-1}$ to allow for uncertainty in the warping prediction. The encoding procedure in the above models computes the variational mean parameters as $\bar{\mathbf{w}}_t = f_w(\mathbf{x}_t, \hat{\mathbf{x}}_{t-1}), \bar{\mathbf{v}}_t = f_v(\mathbf{x}_t - h_{warp}(\hat{\mathbf{x}}_{t-1}, g_w(\mathbf{w}_t)))$, corresponding to a structured posterior $q(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{<t}) = q(\mathbf{w}_t|\mathbf{x}_t, \mathbf{z}_{<t})q(\mathbf{v}_t|\mathbf{w}_t, \mathbf{x}_t, \mathbf{z}_{<t})$. We illustrate the above generative and inference procedures in Fig. 1(b).

F. Proposed Models (Base Versions)

Finally, we consider a version of stochastic temporal autoregressive transform (11) in the context of predictive video coding,

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) + h_\sigma(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) \odot g_v(\mathbf{v}_t, \mathbf{w}_t). \tag{14}$$

As in DVC and SSF, the latent variable $\mathbf{z}_t$ consists of two components $\mathbf{v}_t, \mathbf{w}_t$, and the shift and scale parameters are computed using only the previous reconstruction $\hat{\mathbf{x}}_{t-1}$. See Fig. 1(c) for a diagram of the resulting generative model. We study two main variants, categorized by how they implement h_μ and h_σ:

- *STAT* uses DCNNs for h_μ and h_σ as in Yang et al. [15], but both networks receive the latent variable $\mathbf{w}_t$ as additional input, which helps guide the transform. In theory, the universal approximation property of neural networks should allow us to learn whichever flexible functions (h_μ, h_σ) achieve the best compression performance. However, in practice, we find the following variant based on warping to be more performant and parameter-efficient.
- *STAT-SSF* is a more domain-knowledge-driven variant of the above that still uses scale-space warping [5] in the shift transform, i.e., $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) = h_{warp}(\hat{\mathbf{x}}_{t-1}, g_w(\mathbf{w}_t))$. This can also be seen as an extended version of the SSF model, whose shift transform h_μ is preceded by a new learned scale transform h_σ. We motivated the scaling transform in Section II-D, and provide a visualization of its effect in Fig. 2.

Structured Prior (SP). Besides improving the autoregressive transform (affecting the likelihood model for $\mathbf{x}_t$), one variant of our approach also improves the topmost generative hierarchy in the form of a more expressive latent prior $p(\mathbf{z}_{1:T})$, affecting the entropy model for compression. We observe that motion information encoded in $\mathbf{w}_t$ can often be informative of the residual error encoded in $\mathbf{v}_t$. In other words, large residual errors $\mathbf{v}_t$ incurred by the mean prediction $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$ (e.g., the result of warping the previous frame $h_\mu(\hat{\mathbf{x}}_{t-1})$) are often spatially collocated with (unpredictable) motion as encoded by $\mathbf{w}_t$.

The original SSF model's prior factorizes as $p(\mathbf{w}_t, \mathbf{v}_t) = p(\mathbf{w}_t)p(\mathbf{v}_t)$ and does not capture such correlation. We, therefore, propose a structured prior by introducing conditional dependence between $\mathbf{w}_t$ and $\mathbf{v}_t$, so that $p(\mathbf{w}_t, \mathbf{v}_t) = p(\mathbf{w}_t)p(\mathbf{v}_t|\mathbf{w}_t)$. At a high level, this can be implemented by introducing a new neural network that maps $\mathbf{w}_t$ to parameters of a parametric distribution of $p(\mathbf{v}_t|\mathbf{w}_t)$ (e.g., mean and variance of a diagonal Gaussian distribution). This results in variants of the above models, *STAT-SP* and *STAT-SSF-SP*, where the structured prior is applied on top of the proposed **STAT** and *STAT-SSF* models, respectively.

G. Temporal Prior (TP) Extensions

In previous sections and other similar works in variational compression [5], [11], [37], the prior model $p(\mathbf{z}_{1:T})$ typically assumes that the latent variables $\mathbf{z}_{1:T}$ are temporally factorized: $p(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t)$. However, such assumptions may be unrealistic for real world data, such as modeling natural video [10], [38], [39], where temporal dependencies may persist even after removing low-level motion information. To this end, we model these dependencies by introducing a temporal prior: $p(\mathbf{z}_{1:T}) = p(\mathbf{z}_1)\prod_{t=2}^{T} p(\mathbf{z}_t|\mathbf{z}_{<t})$ to more accurately predict the density of the latent variable. Given the framework described in the previous section, we implement the prior of the video compression model with the following two types of temporal conditioning. Meanwhile, the motion latent $\mathbf{w}_t$ is not discussed here, as it usually has a much lower bitrate than $\mathbf{v}_t$.

Temporal Residual Conditioning $p(\mathbf{v}_t|\mathbf{v}_{t-1})$ (TP). Most video compression methods are based on the Markov assumption, e.g., an optical flow's vector field is only conditioned on the most recent frame. Empirically, we find that the information content of the compressed flow field is much smaller than that of the residual, $-\log p(\mathbf{v}_t) \gg -\log p(\mathbf{w}_t)$. Therefore, we only place a temporal prior on $\mathbf{v}_t$ and keep $p(\mathbf{w}_t)$ temporally factorized for simplicity. Temporal conditioning for $\mathbf{v}$ is implemented by using an additional neural network that takes $\mathbf{v}_{t-1}$ as input to a conditional Gaussian prior for $\mathbf{v}_t$. However, this conditioning scheme may require special handling of the initial frames because the individual image compression model for the I-frame (the first frame) does not have a "residual." This indicates that such temporal conditioning can only start from the 2nd frame, as $p(\mathbf{v}_t|\mathbf{v}_{t-1})$ is only available for $t \geq 3$. Instead, we use a separate factorized prior $p(\mathbf{v}_2)$ for the 2nd frame.

Conditioning on the Previous Frame $p(\mathbf{v}_t|\hat{\mathbf{x}}_{t-1})$ (TP+). Learning a temporal prior by only conditioning on the latent variable $\mathbf{v}$ is challenging in practice. First, $\mathbf{v}$ is a noisy quantity during training and only carries information of the previous frame's *residual*, lacking explicit information of the previous frame. Furthermore, the latent representation can change throughout training because of the noise injection, potentially complicating the learning process. Instead, we explore an alternative scheme, *TP+*, in which $\mathbf{v}_t$ is conditioned on the previous reconstruction, $\hat{\mathbf{x}}_{t-1}$, which maintains a less noisy, more

informative, and more consistent feature representation throughout training and simplifies the learning procedure. In this scenario, the model also no longer requires the extra prior for $p(\mathbf{v}_2)$, as in *TP*; moreover, the resulting prior $p(\mathbf{v}_t|\hat{\mathbf{x}}_{t-1}) = p(\mathbf{v}_t|\mathbf{w}_{<t}, \mathbf{v}_{<t})$ (since $\hat{\mathbf{x}}_{t-1}$ is a deterministic function of $\mathbf{z}_{<t}$) offers a strictly more expressive probabilistic model than *TP*, $p(\mathbf{v}_t|\mathbf{v}_{t-1})$.

H. Variable Bitrate Extensions

Classical compression methods such as HEVC or AVC typically use *one* compression model to compress a video for 52 different quality levels. For example, in the FFMPEG library, this can be achieved by varying the value of the Constant Rate Factor (CRF). However, current end-to-end trained neural codecs largely focus on optimal rate-distortion performance at a fixed bitrate. As discussed, various neural video compression models [5], [11], [12], [37] minimize the NELBO objective, $\mathcal{L} = \mathcal{D} + \beta\mathcal{R}$, where β controls the rate-distortion tradeoff, and a separate compression model needs to be optimized for each setting of β. The overhead of training and deploying multiple models (e.g, 52 sets of neural network parameters for 52 quality levels) potentially makes neural video compression impractical. Therefore, we consider a variable-rate compression setting, where we train a single model with multiple β values and adapt it to different quality factors at deployment time.

The analogy between the compression models and deep latent variable models (see Section II-C) inspires us to draw on conditional variational autoencoders [40] that model conditional distributions instead of unconditional ones. By conditioning the networks on the quality control factor B and generating random Q-samples during training, we can learn a single set of encoder, decoder, and prior model that operates near-optimally at different bitrates. Specifically, we draw on a proposal by [41] for images and generalize it to the class of neural video codecs studied here.

In more detail, we define a quality control factor B, represented by a one-hot vector, as the condition of the compression model where each B has a unique corresponding β. Then we can derive the new conditional optimization objective from (9)

$$\tilde{\mathcal{L}_{\text{VBR}}} = \mathbb{E}_{\mathbf{x}_{1:T}\sim p_{\text{data}}}\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},B)} [-\log p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}, B) - \beta \log p(\mathbf{z}_{1:T}|B)] \tag{15}$$

$$\equiv \mathcal{D}(B) + \beta\mathcal{R}(B). \tag{16}$$

During training, a (B, β) pair is selected randomly for each training sample. Following [41], we also replace all convolutions with conditional convolution to incorporate the B variable. We found that one can technically use simpler methods (like directly reshaping and concatenating B to the network hidden features) to achieve the same result.

III. Related Work

We divide related work into three categories: neural image compression, neural video compression, and sequential generative models.

Neural Image Compression. Considerable progress has been made by applying neural networks to image compression; here we focus on the lossy (instead of lossless) compression setting more relevant to us. Early works [42], [43] leveraged LSTMs to model spatial correlations of the pixels within an image. [33], [44] were among the first to train an autoencoder-stlye model with rate-distortion loss for end-to-end lossy image compression, and proposed uniform noise injection [44] or the straight-through estimator [45] to approximately differentiate through quantization. The end-to-end approach based on uniform noise injection was given a principled *probabilistic* interpretation as implementing a deep latent variable model, a variational autoencoder (VAE) [25]. In subsequent work, [46] encoded images with a two-level VAE architecture involving a scale hyper-prior, which can be further improved by autoregressive structures [2], [47] or by optimization at encoding time [3]. [48] and [49] demonstrated competitive image compression performance without a pre-defined quantization grid. Recently, [50] attempted to apply normalizing flow to lossy image compression; noting the incompatibility between density modeling with flow and the R-D objective of lossy compression (which we discuss in more detail in Section II-B), they essentially trained an invertible autoencoder with a R-D loss instead, resulting in superior rate-distortion performance in the high bitrate regime but otherwise inferior performance compared to the VAE approach [32], [46] without the invertibility requirement.

Neural Video Compression. Compared to image compression, video compression is a significantly more challenging problem, as statistical redundancies exist not only within each video frame (exploited by intra-frame compression) but also along the temporal dimension. Early works by [51], [52] and [53] performed video compression by predicting future frames using a recurrent neural network, whereas [54] and [55] used convolutional architectures within a traditional block-based motion estimation approach. These early approaches did not outperform the traditional H.264 codec and barely surpassed the MPEG-2 codec. [11] adopted a hybrid architecture that combined a pre-trained Flownet [56] and residual compression, which leads to an elaborate training scheme. [37] and [12] combined 3D convolutions for dimensionality reduction with expressive autoregressive priors for better entropy modeling at the expense of parallelism and runtime efficiency. Our method extends a low-latency model proposed by [5], which allows for end-to-end training, efficient online encoding and decoding, and parallelism.

Sequential Deep Generative Models. We drew inspiration from a body of work on sequential generative modeling. Early deep learning architectures for dynamics forecasting involved RNNs [18]. [38] and [57] used VAE-based stochastic models in conjunction with LSTMs to model dynamics. [39] introduced both local and global latent variables for learning disentangled representations in videos. Other video generation models used generative adversarial networks (GANs) [58], [59] or autoregressive models and normalizing flows [8], [9], [16], [29], [60], [61]. Recently, [10] proposed to combine latent variable models with autoregressive flows for modeling dynamics at different levels of abstraction, which inspired our models and viewpoints.

Algorithm 1: Gaussian Pyramid for Scale Space Volume.

Result: *ssv*: Scale-space 3D volume
Input: *image* input image; σ_0 base scale; M scale depth;
ssv = [image];
kernel = Gaussian_Kernel(σ_0);
for *i=0 to M-1* **do**
  image = Conv(image, kernel);
  if *i == 0* **then**
    ssv.append(image);
  else
    scaled = image;
    for *j=0 to i-1* **do**
      scaled = UpSample2x(scaled);
    end
    ssv.append(scaled);
  end
  image = DownSample2x(image);
end
return Concat(ssv)

IV. Experiments

In this section, we demonstrate the effectiveness of our models compared with baseline neural video compression solutions [5], [11] as well as a state-of-the-art classic video codec when evaluated on two publicly available benchmark datasets. The backbone modules and training scheme for our models largely follow [5].

Regarding the implementation of scale-space volume [5], we adopted a Gaussian pyramid approach with successive convolution and downsampling. Compared with naively applying separate Gaussian kernels with exponentially increasing kernel width (referred to as *Gaussian-blur* approach), the Gaussian pyramid-based implementation uses a fixed small convolution kernel, yields similar scale-space volume, and is much more efficient in terms of computation. The pseudo-code is presented in Algorithm 1

A. Training and Evaluation

Training. We train on the Vimeo-90k [62] dataset as in previous works [11], [12], [63]. We follow largely the same procedure as SSF [5], and give more details in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2023.3260684. Table I summarizes our proposed models and the baseline methods whose published results we compare with.

Evaluation. We report compression performance on two widely used datasets: *UVG*[5] [64] and *MCL_JCV* [65]. Both consist of raw videos in YUV420 format. UVG contains seven 1080p videos at 120fps with smooth and mild motions or stable camera movements. MCL_JCV contains thirty 1080p videos at 30 fps, which are generally more diverse, with a higher degree of motion and scene cuts.

We report the bitrate (bits-per-pixel, or BPP for short) and the reconstruction quality, measured in PSNR on 8-bit RGB space (same as in training), averaged across all frames. PSNR is a more challenging metric than MS-SSIM [66] for learned codecs [5], [11], [37], [63], [67], [68]. Since most existing neural compression methods assume video input in 8-bit RGB444 format (24 bits per pixel), we follow this convention by converting test videos from YUV420 to RGB444 in our evaluations for meaningful comparison. It is worth noting that HEVC is mainly designed for YUV420 with sub-sampled chroma components, which means that it can exploit the fact that chroma components in test videos are subsampled and thus gains an unfair advantage in the comparison against neural codecs. Nevertheless, we report the performance of HEVC as a reference.

B. Base Results

First, we examine the performance of our proposed approach without introducing temporal conditioning in the prior. Fig. 3(a) compares our proposed models (*STAT-SSF* and *STAT-SSF-SP*) with classical codec HEVC and baseline neural codecs on the UVG test dataset. As can be seen from the rate-distortion results, our *STAT-SSF-SP* model is uniformly better than previously known neural codecs represented by SSF [5] and DVC [11]. *STAT-SSF-SP* even outperforms HEVC(YUV) (using YUV420 video input and evaluating on RGB444, the setting is available in the Supplementary Material, available online) at bitrates ≥ 0.07 BPP. As expected, our proposed *STAT* model improves over *TAT* [15], with the latter lacking stochasticity in the autoregressive transform compared to our proposed STAT and its variants.

Fig. 3(a) shows that the performance ranking on MCL_JCV is similar to that on UVG, despite MCL_JCV having more diverse and challenging (e.g., animated) content. We provide qualitative results in Figs. 2 and 4, offering insight into the behavior of the proposed scaling transform and structured prior, as well as visual qualities of different methods.

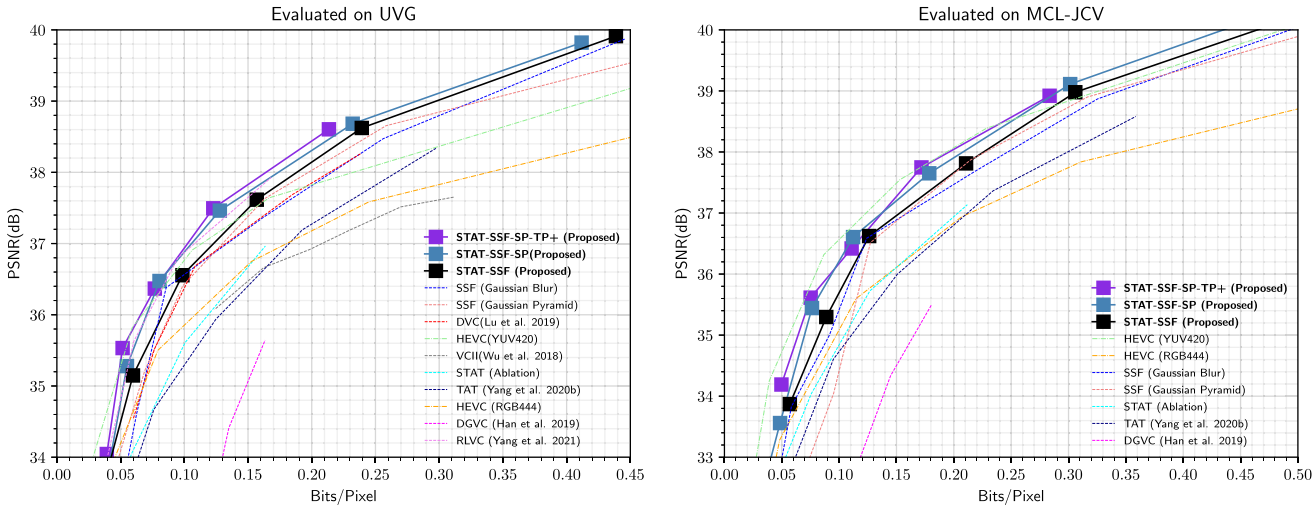
Additionally, we observe that different implementations of scale space volume (Gaussian pyramid or Gaussian blur) also slightly influence the rate-distortion performance. While the Gaussian blur version seems to have more advantage at the high bitrate regime, the Gaussian pyramid performs better at mid-range bitrate, as shown in Fig. 3. The Gaussian pyramid usually generates a blurrier high-scale image compared to Gaussian blur when the base scales are the same.

Finally, we ablate on our proposed STAT-SSF-SP model. This allows us to study the performance contribution of each of our proposed components, stochastic temporal autoregressive transform (STAT) and structured prior (SP), in isolation. Compared to the baseline SSF [5] model, STAT-SSF introduces an additional elementwise scaling transform, while SSF-SP introduces the structured prior (see discussions in Section II-F). As shown in Fig. 5(a), *SSF-SP* provides a comparable or greater improvement than *STAT-SSF*, relative to *SSF*. The improvements from the two components are complementary, and combining them into *STAT-SSF-SP* yields even further improvement as seen in Fig. 3(a).

[5]UVG dataset is provided by http://ultravideo.fi/ and under CC-by-NC 3.0 license.

TABLE I
OVERVIEW OF VARIOUS COMPRESSION METHODS CONSIDERED AND THE CONTEXTS IN WHICH THEY APPEAR

Model Name	Category	Remark
STAT-SSF	Proposed	Proposed autoregressive transform with scale space flow
STAT-SSF-SP	Proposed	Same as above (STAT-SSF) but with structured prior
STAT-SSF-SP-TP+	Proposed	Structured prior and improved temporal prior, $p(\mathbf{v}_t\|\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$
SSF	Baseline	Agustsson et al. [5]
DVC	Baseline	Lu et al. [11]
VCII	Baseline	Wu et al. [51] (trained on the Kinectics dataset)
DGVC	Baseline	Han et al. [53] modified for low-latency compression setup
TAT	Baseline	Yang et al. [15]
HEVC	Baseline	FFMPEG-HEVC with RGB 4:4:4 color space input
HEVC(YUV)	Baseline	FFMPEG-HEVC with YUV 4:2:0 color space input
STAT	Ablation	STAT-SSF without optical flow
SSF-SP	Ablation	SSF with structured prior
SSF-TP	Ablation	SSF with temporal prior conditioned on $\mathbf{v}_{t-1}$, $p(\mathbf{v}_t\|\mathbf{v}_{t-1})$
SSF-TP+	Ablation	SSF with temporal prior conditioned on $\hat{\mathbf{x}}_{t-1}$, $p(\mathbf{v}_t\|\hat{\mathbf{x}}_{t-1})$



(a) (b)

Fig. 3. *Rate-Distortion Performance* of various models and ablations. Results are evaluated on **(a)** UVG and **(b)** MCL_JCV datasets. All the learning-based models (except VCII [51]) are trained on Vimeo-90 k. STAT-SSF-SP-TP+ (proposed) achieves the best performance.

C. Temporal Prior Experiments

We also conduct experiments to illustrate the rate-distortion performance of *SSF* [5] and our proposed models using different temporal priors. Refer to Table I for naming conventions.

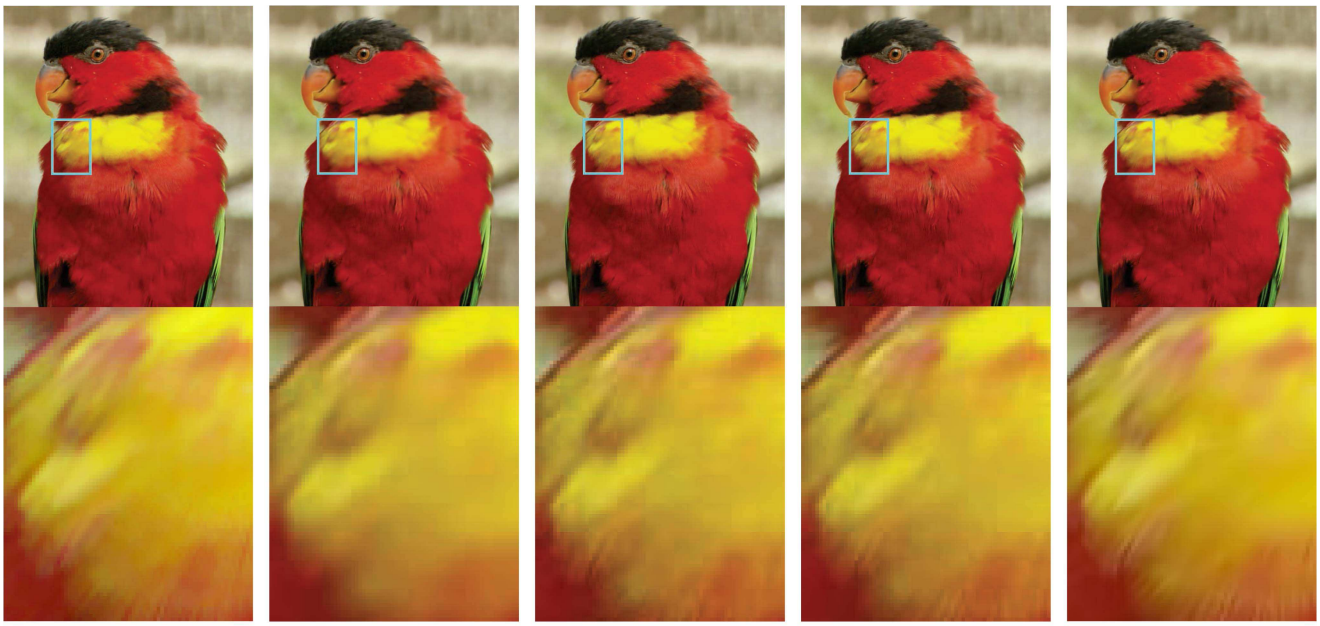
Fig. 6 shows the RD curve of each ablation model. We see that performing temporal conditioning through the prior does indeed improve performance, both for *STAT* and *SSF*. We see that temporally conditioning on the previous reconstruction (*SSF-TP+*) outperforms temporally conditioning on the previous latent (*SSF-TP*) in almost all bitrate regimes across both datasets. Finally, *SSF-TP+* even performs comparably with *STAT-SSF-SP*, and the hybrid model *STAT-SSF-SP-TP+* offers slight improvement further.

However, we also observe that compared with the *SSF-TP+* or *SSF* model, *SSF-TP* shows fluctuating rate-distortion performance during evaluation if we train the same model with different random initializations. We hypothesize that this may be caused by either numerical instability or video length inconsistency between training and evaluation data. To the best of our knowledge, most proposed sequential generative or compression models with a temporal prior are trained with more than three consecutive frames [10], [12], [38], [39], [63], [69]. In contrast, our model only uses three consecutive frames during training for better efficiency.

Fortunately, we can still avoid the issue by using "β-annealing". We first train a model at a high bitrate and then finetune the model to lower bitrates using the increasingly larger β value. In our experiment, the initial model is trained with $\beta = 1.5625 \times 10^{-4}$, and the following models are finetuned with $\beta = \{3.125 \times 10^{-4}, 6.25 \times 10^{-4}, 1.25 \times 10^{-3}, 2.5 \times 10^{-3}, 5 \times 10^{-3}\}$, respectively.

D. Variable-Bitrate Experiments

In contrast to previous experiments that train separate models, one for each β value, in this section, we show results of the proposed conditional autoencoder based variable-bitrate scheme. The conditioning factor B is defined as a one-hot vector with length 7 and seven β values are used to match



(a) HEVC; BPP $= 0.087$, PSNR $= 38.10$

(b) SSF; BPP $= 0.082$, PSNR $= 37.44$

(c) STAT-SSF (ours); **BPP = 0.077**, PSNR $= 38.11$

(d) STAT-SSF-SP (ours); **BPP = 0.055**, PSNR $= 38.10$

(e) STAT-SSF-SP-TP+ (ours); **BPP = 0.054**, PSNR $= 38.09$

Fig. 4. *Qualitative comparisons* of various methods on a frame from MCL-JCV video 30. Figures in the bottom row focus on the same image patch on top. Here, models with the proposed scale transform (*STAT-SSF* and *STAT-SSF-SP*) outperform the ones without, yielding visually more detailed reconstructions at lower rates. The structured prior (*STAT-SSF-SP*) and temporal prior (*STAT-SSF-SP-TP+*) reduce the bitrate further.

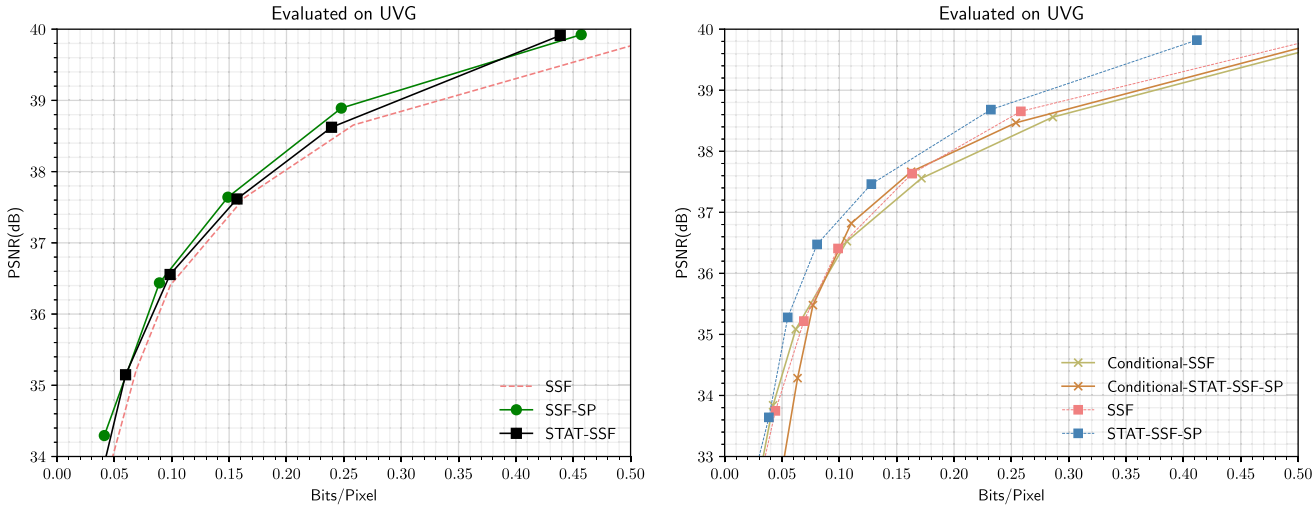


Fig. 5. (a) Ablation study of STAT-SSF-SP, examining the effect of two proposed components, STAT (stochastic temporal autoregressive transform) and SP (structured prior), with R-D results evaluated on the UVG dataset. Compared to STAT-SSF-SP, SSF-SP lacks the learned elementwise scaling transform in STAT (Section II-D), STAT-SSF lacks the structured prior, while SSF [5] lacks both components. See discussion in Section IV-B. (b) Comparison of the Rate-Distortion performance between variable-bitrate models and non-variable-bitrate models.

each one-hot vector: $\{10^{-2}, 5\times10^{-3}, 2.5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 2.5\times10^{-4}, 10^{-4}\}$. For each training *datum*, the (B,β) pair is selected randomly. We also observe that this sampling scheme results in better rate-distortion performance than sampling (B,β) per *batch*.

In Fig. 5(b), we compare the performance of the variable-bitrate models and models with separate β values. We observe that *Conditional SSF* performs slightly worse than *SSF* at high bitrate. While *Conditional STAT-SSF-SP* consistently has worse performance than *STAT-SSF-SP*, it performs better than

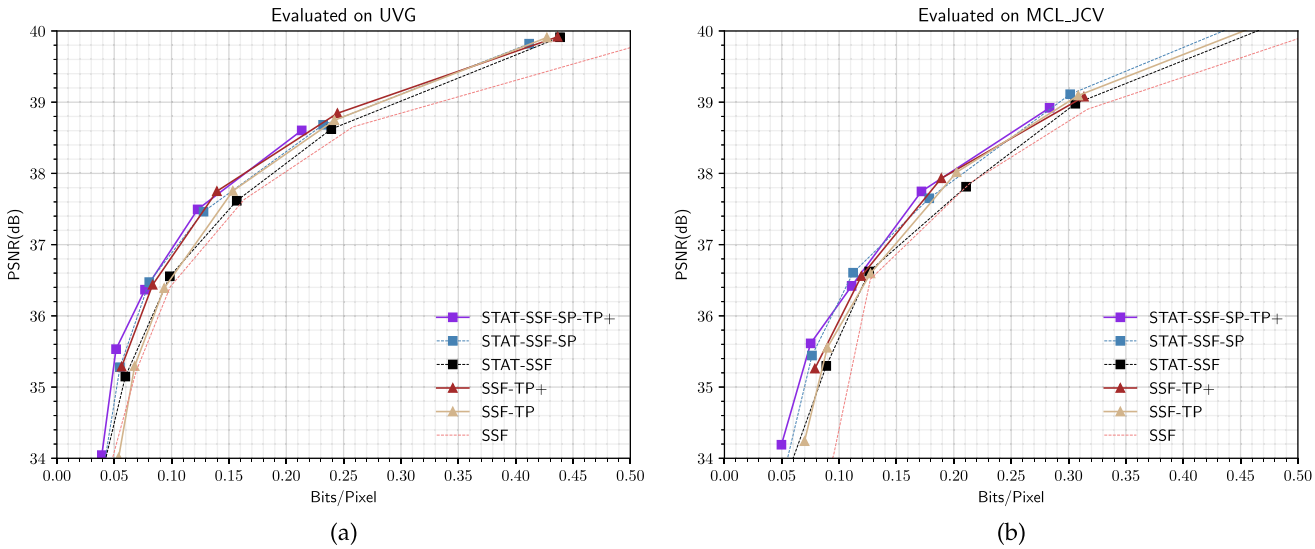


Fig. 6. *Rate-Distortion Performance* of various models and ablations (see Table I). Results are evaluated on **(a)** UVG and **(b)** MCL_JCV datasets. The best results are obtained by models with deterministic temporal conditioning (TP+), while the improvement from conditioning on previous latents (TP) is less.

Conditional SSF at bitrate ≤ 0.08 bpp. This result demonstrates that the performance gap between variable and non-variable-bitrate models could be amplified with a more complicated model, while the gap is barely noticeable for simpler models. The investigation of this phenomenon will be left to future research.

V. Discussion

Generative modeling and compression share similar aims, respectively modeling and removing redundant structure in data. Accordingly, the nascent field of learned, neural compression holds the potential to drastically improve performance over conventional codecs. Perhaps nowhere is this more impactful than in the setting of high-resolution video, where capturing temporal redundancy can yield significant reductions in coding costs. Toward this end, we have drawn inspiration from a recently developed technique within deep generative models, temporal autoregressive flows [10], which perform temporal predictive coding via affine transforms. By interpreting recent state-of-the-art neural compression methods via autoregressive flows, we can generalize them to using more complex transforms, with corresponding improvements in performance. We have investigated such improvements on video compression benchmarks, along with extensions to the higher-level latent variable model: additional hierarchical priors, temporal priors, and variable bitrate compression. Of particular importance, our results show that domain knowledge (in our case, next frame prediction via the computer vision technique of warping) remains valuable in video compression and can be more cost-effective than a neural-network-only approach. They also shed light on the potential pitfalls in estimating higher-level temporal dynamics in learned compression models.

The art of transform coding involves the effective design of both the transform and entropy model, and many possibilities remain in the domain of video compression. Unlike for images, where the convolutional neural network has been established as the default transform for capturing spatial redundancy [17], video introduces additional temporal redundancy, and it remains to be seen what transform is most effective while staying within (often stringent) computation budgets. The design space is vast, and the approach of predictive coding combined with computer vision techniques has so far achieved the widest commercial success. Our work shows one way of extending this approach inspired by normalizing flows, and opens up the possibility of compression with more general and potentially non-affine autoregressive transforms [70]. Finally, recent work [71] has shown state-of-the-art video compression performance with a rich entropy model and without predictive coding in the pixel space. This approach essentially offloads the task of designing good transforms to that of modeling high-dimensional discrete distributions, which now can be tackled thanks to the rise of large-scale transformer models [72] and abundance of data, but remains computationally prohibitive. An interesting future research direction could therefore be to explore the complementary strengths between this and our approach for sequence decorrelation, which may lead to further advances in both rate-distortion performance and computational efficiency.

References

[1] Y. Yang, S. Mandt, and L. Theis, "An introduction to neural data compression," 2022, *arXiv:2202.06533*.
[2] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10771–10780.
[3] Y. Yang, R. Bamler, and S. Mandt, "Improving inference for neural image compression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, Art. no. 49.
[4] F. Bellard, "BPG image format," 2014. [Online]. Available: https://bellard.org/bpg/bpg_spec.txt
[5] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8503–8512.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[7] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[8] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," 2014, *arXiv:1410.8516*.

[9] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," 2016, *arXiv:1605.08803*.

[10] J. Marino, L. Chen, J. He, and S. Mandt, "Improving sequential latent variable models with autoregressive flows," in *Proc. Symp. Adv. Approx. Bayesian Inference*, 2020, pp. 1–16.

[11] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11006–11015.

[12] H. Liu, H. Shen, L. Huang, M. Lu, T. Chen, and Z. Ma, "Learned video compression via joint spatial-temporal correlation exploration," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11580–11587.

[13] J. Lin, D. Liu, H. Li, and F. Wu, "M-LVC: Multiple frames prediction for learned video compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3546–3554.

[14] G. Lu et al., "Content adaptive and error propagation aware deep video compression," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 456–472.

[15] R. Yang, Y. Yang, J. Marino, Y. Yang, and S. Mandt, "Deep generative video compression with temporal autoregressive transforms," in *Proc. ICML Workshop Invertible Neural Netw. Normalizing Flows, Explicit Likelihood Models*, 2020.

[16] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2338–2347.

[17] J. Ballé et al., "Nonlinear transform coding," 2020, *arXiv:2007.03034*.

[18] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2980–2988.

[19] T. M. Cover, *Elements of Information Theory*, Hoboken, NJ, USA: Wiley, 1999.

[20] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. D. Bourdev, "Learned video compression," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3453–3462.

[21] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[22] R. E. Kalman et al., "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[23] C. C. Cutler, "Differential quantization of communication signals," Jul. 29, 1952, US Patent 2,605,361.

[24] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *J. Mach. Learn. Res.*, vol. 22, 2021, Art. no. 57.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[26] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, vol. 159, Berlin, Germany: Springer Science & Business Media, 2012.

[27] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 9–21, Sep. 2001.

[28] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen, "Understanding and mitigating exploding inverses in invertible neural networks," in *Proc. 24th Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1792–1800.

[29] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10215–10224.

[30] B. Uria, I. Murray, and H. Larochelle, "RNADE: The real-valued neural autoregressive density-estimator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2175–2183.

[31] J. Ho, E. Lohn, and P. Abbeel, "Compression with flows via local bits-back coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3874–3883.

[32] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[33] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *Proc. Int. Conf. Learn. Representations*, 2017.

[34] F. Schmidt and T. Hofmann, "Deep state space models for unconditional word generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6158–6168.

[35] F. Schmidt, S. Mandt, and T. Hofmann, "Autoregressive text generation beyond feedback loops," in *Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 3391–3397.

[36] C. A. Glasbey and K. V. Mardia, "A review of image-warping methods," *J. Appl. Statist.*, vol. 25, no. 2, pp. 155–171, 1998.

[37] A. Habibian, T. V. Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7033–7042.

[38] E. Denton and R. Fergus, "Stochastic video generation with a learned prior," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 1174–1183.

[39] Y. Li and S. Mandt, "Disentangled sequential autoencoder," in *Proc. 35th Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5670–5679.

[40] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3483–3491.

[41] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3146–3154.

[42] G. Toderici et al., "Full resolution image compression with recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5435–5443.

[43] N. Johnston et al., "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4385–4393.

[44] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Proc. Picture Coding Symp.*, 2016, pp. 1–5.

[45] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[46] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Representations*, 2018.

[47] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 3339–3343.

[48] Y. Yang, R. Bamler, and S. Mandt, "Variational Bayesian quantization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10670–10680.

[49] G. Flamich, M. Havasi, and J. M. Hernández-Lobato, "Compression without quantization," in *Proc. Int. Conf. Learn. Representations*, 2019.

[50] L. Helminger, A. Djelouah, M. Gross, and C. Schroers, "Lossy image compression with normalizing flows," 2020, *arXiv:2008.10486*.

[51] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 416–431.

[52] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6420–6428.

[53] J. Han, S. Lombardo, C. Schroers, and S. Mandt, "Deep generative video compression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, Art. no. 833.

[54] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 566–576, Feb. 2020.

[55] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma, "DeepCoder: A deep neural network based video compression," in *Proc. IEEE Vis. Commun. Image Process.*, 2017, pp. 1–4.

[56] A. Dosovitskiy et al., "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.

[57] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic variational video prediction," in *Proc. Int. Conf. Learn. Representations*, 2018.

[58] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 613–621.

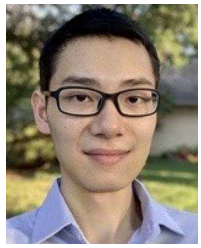
[59] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," 2018, *arXiv:1804.01523*.

[60] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1530–1538.

[61] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4743–4751.

[62] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.
[63] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with hierarchical quality and recurrent enhancement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6627–6636.
[64] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," in *Proc. 11th ACM Multimedia Syst. Conf.*, 2020, pp. 297–302.
[65] H. Wang et al., "MCL-JCV: A JND-based h. 264/AVC video quality assessment dataset," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 1509–1513.
[66] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. IEEE 37th Asilomar Conf. Signals Syst. Comput.*, 2003, pp. 1398–1402.
[67] Y. Yang, G. Sautière, J. J. Ryu, and T. S. Cohen, "Feedback recurrent autoencoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 3347–3351.
[68] N. Johnston, E. Eban, A. Gordon, and J. Ballé, "Computationally efficient neural image compression," 2019, *arXiv:1912.08771*.
[69] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with recurrent auto-encoder and recurrent probability model," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 388–401, Feb. 2021.
[70] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural autoregressive flows," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 2078–2087.
[71] F. Mentzer et al., "VCT: A video compression transformer," 2022, *arXiv:2206.07307*.
[72] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

Ruihan Yang received the BS degree in computer science from NYU Shanghai, in 2018. He is currently working toward the PhD degree with the University of California, Irvine. From 2018 to 2019, his work focused on generative modeling of music, and had some experience on computational material science. He currently works on source compression with generative models.



Yibo Yang is currently working toward the PhD degree with the University of California, Irvine. His research interests include probability theory, information theory, and their applications in machine learning. His recent work develops deep generative modeling approaches for data compression, and he has co-organized tutorials and workshops on the topic.

Joseph Marino received the PhD degree in computation and neural systems from the California Institute of Technology (Caltech). His work focuses on probabilistic models and inference algorithms as they relate to deep generative modeling, reinforcement learning, and theoretical neuroscience. He is currently employed as a research scientist with DeepMind.

Stephan Mandt (Member, IEEE) received the PhD degree in theoretical physics from the University of Cologne, where he received the German National Merit Scholarship. He is an associate professor of computer science and statistics with the University of California, Irvine. From 2016 until 2018, he was a senior researcher and head of the Statistical Machine Learning Group, Disney Research in Pittsburgh and Los Angeles. He held previous postdoctoral positions with Columbia University and Princeton University. He is furthermore a recipient of the NSF CAREER Award, the UCI ICS Mid-Career Excellence in Research Award, the German Research Foundation's Mercator Fellowship, a Kavli fellow of the U.S. National Academy of Sciences, a member of the ELLIS Society, and a former visiting researcher with Google Brain. He is an action editor of the *Journal of Machine Learning Research* and of *Transactions on Machine Learning Research*. His research is currently supported by NSF, DARPA, IARPA, DOE, Disney, Intel, and Qualcomm.