3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–13

Generative Video Compression as Hierarchical Variational Inference

Ruihan Yang RUIHAN.YANG@UCI.EDU
Yibo Yang YIBO.YANG@UCI.EDU
Joseph Marino JMARINO@CALTECH.EDU
Stephan Mandt MANDT@UCI.EDU

Abstract

Recent work by Marino et al. (2020) showed improved performance in sequential density estimation by combining masked autoregressive flows with hierarchical latent variable models. We draw a connection between such autoregressive generative models and the task of lossy video compression. Specifically, we view recent neural video compression methods (Lu et al., 2019; Yang et al., 2020b; Agustsson et al., 2020) as instances of a generalized stochastic temporal autoregressive transform, and propose avenues for enhancement based on this insight. Comprehensive evaluations on large-scale video data show improved rate-distortion performance over both state-of-the-art neural and conventional video compression methods.

1. Introduction

Recent advances in deep generative modeling have enabled a surge in applications, including learning-based compression. Generative models have already demonstrated empirical improvements in image compression, outperforming classical codecs (Minnen et al., 2018; Yang et al., 2020d), such as BPG (Bellard, 2014). In contrast, the less developed area of neural video compression remains challenging due to complex temporal dependencies operating at multiple scales. Nevertheless, recent neural video codecs have shown promising performance gains (Agustsson et al., 2020), in some cases on par with current hand-designed, classical codecs, e.g., HEVC.

Source compression fundamentally involves decorrelation, i.e., transforming input data into white noise distributions that can be easily modeled and entropy-coded. Thus, improving a model's capability to decorrelate data automatically improves its compression performance. Likewise, we can improve the associated 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 *model* structure. One family of models, autoregressive flows, maps between less structured distributions, e.g., uncorrelated noise, and more structured distributions, e.g., images or video (Dinh et al., 2014, 2016). The inverse mapping can remove dependencies in the data, making it more amenable for compression.

This paper draws on recent insights in hierarchical sequential latent variable models with autoregressive flows (Marino et al., 2020). In particular, we identify connections between this family of models and recently proposed neural video codecs based on motion estimation (Lu et al., 2019; Agustsson et al., 2020). By interpreting this technique as instantiation of a type of autoregressive flow transform, we propose various alternatives and improvements based on insights from generative modeling. Our main contributions are as follows:

1. **A new framework.** We interpret existing video compression methods through the more general framework of generative modeling, variational inference, and autoregressive flows, allowing us to readily investigate extensions and ablations. In particular, we compare fully data-driven approaches with motion-estimation-based neural compression schemes. This framework also provides directions for future work.
2. **A new model.** Our main proposed model is an improved version of Scale-Space Flow (SSF) (Agustsson et al., 2020) model, by augmenting a shift transform by scale-then-shift inspired by performance gains in extending NICE (Dinh et al., 2014) to RealNVP (Dinh et al., 2016). Structured priors can improve the performance further.
3. **A new dataset.** The neural video compression community is lacking large, high-resolution benchmark datasets. While we performed extensive experiments on the publicly available Vimeo-90k dataset (Xue et al., 2019), we also collected and utilized a larger dataset, YouTube-NT[1], which is used for our ablation study. Details can be found in Appendix D.

2. Video Compression through Deep Autoregressive Modeling

We identify commonalities between hierarchical autoregressive flow models (Marino et al., 2020) and state-of-the-art neural video compression architectures (Agustsson et al., 2020), and will use this viewpoint to propose improvements on existing models.

2.1. Background

We first review VAE-based compression schemes (Ballé et al., 2017; Theis et al., 2017) and formulate existing video codecs in this framework; we then review autoregressive flows.

Generative Modeling and Source Compression. Let $\mathbf{x}_{1:T} \in \mathbb{R}^{T \times D}$ be a sequence of video frames. Lossy compression seeks to find a compact description of $\mathbf{x}_{1:T}$ that simultaneously minimizes the description length $\mathcal{R}$ and information loss $\mathcal{D}$. The distortion $\mathcal{D}$ measures how much reconstruction error accrues due to encoding $\mathbf{x}_{1:T}$ into a latent representation $\bar{\mathbf{z}}_{1:T}$ and subsequently decoding it back to $\hat{\mathbf{x}}_{1:T}$, while $\mathcal{R}$ measures the bit rate (file size). In learned compression methods (Ballé et al., 2017; Theis et al., 2017), the above process is parameterized by flexible functions f ("encoder") and g ("decoder") that map between the input video and its latent representation: $\bar{\mathbf{z}}_{1:T} = f(\mathbf{x}_{1:T}), \hat{\mathbf{x}}_{1:T} = g(\bar{\mathbf{z}}_{1:T})$, and minimize a rate-distortion loss with hyperparameter $\beta > 0$:

$$\mathcal{L} = \mathcal{D}(\mathbf{x}_{1:T}, g(\lfloor \bar{\mathbf{z}}_{1:T} \rceil)) + \beta \mathcal{R}(\lfloor \bar{\mathbf{z}}_{1:T} \rceil).$$

We adopt the end-to-end compression approach of Ballé et al. (2017); Lombardo et al. (2019), which approximates the rounding operations ($\lfloor \cdot \rceil$) by uniform noise injection to enable gradient-based optimization during training. With an appropriate choice of probability model $p(\mathbf{z}_{1:T})$, the approximated version of the above R-D (rate-distortion) objective then corresponds to the VAE objective:

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

1. https://anonymous.4open.science/r/7fba4f65-e003-446e-81d5-fdd5fed01335/

In this model, the likelihood $p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$ follows a Gaussian distribution with mean $\hat{\mathbf{x}}_{1:T} = g(\mathbf{z}_{1:T})$ and diagonal covariance $\frac{\beta}{2\log 2}\mathbf{I}$, the approximate posterior q is chosen to be a unit-width uniform (thus zero-differential-entropy) distribution whose mean $\bar{\mathbf{z}}_{1:T}$ is predicted by an amortized inference network f. The prior density $p(\mathbf{z}_{1:T})$ interpolates its discretized version, so that it measures the code length of discretized $\bar{\mathbf{z}}_{1:T}$ after entropy-coding.

Masked Autoregressive Flow (MAF). As a final component in neural sequence modeling, we discuss MAF (Papamakarios et al., 2017), which models the joint distribution of a sequence $p(\mathbf{x}_{1:T})$ in terms of a simpler distribution of its underlying noise variables $\mathbf{y}_{1:T}$ through the following autoregressive transform and its inverse:

$$\mathbf{x}_t = h_\mu(\mathbf{x}_{<t}) + h_\sigma(\mathbf{x}_{<t}) \odot \mathbf{y}_t; \Leftrightarrow \mathbf{y}_t = \tfrac{\mathbf{x}_t - h_\mu(\mathbf{x}_{<t})}{h_\sigma(\mathbf{x}_{<t})}. \tag{2}$$

The noise variable $\mathbf{y}_t$ usually comes from a standard normal distribution. While the forward MAF transforms a sequence of standard normal noises into a data sequence, the inverse flow "whitens" the data sequence and removes temporal correlations. Due to its invertible nature, MAF allows for exact likelihood computations, but as we will explain in Section 2.3, we will not exploit this aspect in compression but rather draw on its expressiveness in modeling conditional likelihoods.

2.2. A General Framework for Generative Video Coding

We now describe a general framework that captures several existing low-latency (see Appendix B) neural compression methods as specific instances and gives rise to the exploration of new models. To this end, we combine latent variable models with autoregressive flows into a joint framework. We consider a sequential decoding procedure of the following form:

$$\hat{\mathbf{x}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) + h_\sigma(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t) \odot g_v(\mathbf{v}_t, \mathbf{w}_t). \tag{3}$$

Eq. 3 resembles the definition of the MAF in Eq. 2, but augments this transform with two sets of latent variables $\mathbf{w}_t, \mathbf{v}_t \sim p(\mathbf{w}_t, \mathbf{v}_t)$. Above, h_μ and h_σ are functions that transform the previous reconstructed data frame $\hat{\mathbf{x}}_{t-1}$ along with $\mathbf{w}_t$ into a shift and scale parameter, respectively. The function $g_v(\mathbf{v}_t, \mathbf{w}_t)$ converts these latent variables into a noise variable that encodes residuals with respect to the mean next-frame prediction $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$. Our approach is inspired by Marino et al. (2020) who analyzed a restricted version of the model in Eq. 3, aiming to hybridize autoregressive flows and sequential latent variable models for video prediction. In contrast to Eq. 3, their model involved deterministic transforms as well as residual noise that came from a sequential VAE.

2.3. Example Models and Extensions

Next, we show that the general framework expressed by Eq. 3 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 Eq. 3 is the autoregressive neural video compression model by Yang et al. (2020b), which we denote as temporal autoregressive transform (TAT). Shown

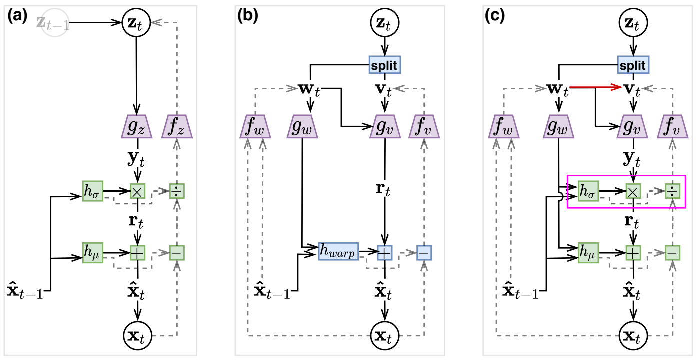


Figure 1: **Model Diagrams**. Graphical models underlying the generative and inference procedures for current frame $\mathbf{x}_t$ of various neural video compression methods. Random variables are shown in circles, all other quantities are deterministically computed; solid and dashed arrows describe computational dependencies during generation (decoding) and inference (encoding), respectively. Purple nodes correspond to neural encoders (CNNs) and decoders (DNNs), and green nodes implement temporal autoregressive transform. (a) TAT; (b) SSF; (c) STAT or STAT-SSF; optional structured prior shown by red arrow from $\mathbf{w}_t$ to $\mathbf{v}_t$. Hyper latent variables in (b) and (c) are left out for clarity.

in Figure 1(a), the decoder g implements a deterministic scale-shift autoregressive transform of decoded noise $\mathbf{y}_t$,

$$\hat{\mathbf{x}}_t = g(\mathbf{z}_t, \hat{\mathbf{x}}_{t-1}) = h_\mu(\hat{\mathbf{x}}_{t-1}) + h_\sigma(\hat{\mathbf{x}}_{t-1}) \odot \mathbf{y}_t, \quad \mathbf{y}_t = g_z(\mathbf{z}_t). \tag{4}$$

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 (DNN) that approximately inverts f_z.

DVC (Lu et al., 2019) and Scale-Space Flow (SSF, Agustsson et al. (2020)). The second class of models captured by Eq. 3 belong to the conventional video compression framework based on predictive coding (Cutler, 1952; Wiegand et al., 2003; Sullivan et al., 2012); 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 motion estimation used in warping prediction, and $\mathbf{v}$ helps capture residual error not predicted by warping.

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, without scaling factor h_σ compared to the autoregressive transform in Eq. 3

$$\hat{\mathbf{x}}_t = h_{warp}(\hat{\mathbf{x}}_{t-1}, g_w(\mathbf{w}_t)) + g_v(\mathbf{v}_t, \mathbf{w}_t) \tag{5}$$

where g_w and g_v are DNNs, $\mathbf{o}_t := g_w(\mathbf{w}_t)$ has the interpretation of an estimated optical flow (motion) field, h_{warp} is the computer vision technique of warping, 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.

Proposed: models based on Stochastic Temporal Autoregressive Transform. Finally, we consider the most general models as described by the stochastic autoregressive transform in Eq. 3, shown in Figure 1(c). We study two main variants, categorized by how they implement h_μ and h_σ:

1. **STAT** uses DNNs for h_μ and h_σ as in (Yang et al., 2020b), but complements it with the latent variable $\mathbf{w}_t$ that characterizes the transform. In principle, more flexible transform functions h_μ and h_σ should lead to better rate-distortion results; however, we find the following variant more performant and parameter-efficient in practice:

2. **STAT-SSF**: a less data-driven variant of the above that still uses scale-space warping (Agustsson et al., 2020) 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 scale transform h_σ.

Besides increasing the flexibility of the reconstruction transform (hence the likelihood model for $\mathbf{x}_t$), we also consider improving the topmost generative hierarchy in the form of a more expressive latent prior $p(\mathbf{z}_{1:T})$, corresponding to improving the entropy model for compression. Specifically, we note that the prior in the SSF model assumes the factorization $p(\mathbf{w}_t, \mathbf{v}_t) = p(\mathbf{w}_t)p(\mathbf{v}_t)$, which can be restrictive. We propose a structured prior by introducing conditional dependence between $\mathbf{w}_t$ and $\mathbf{v}_t$, so that $p(\mathbf{w}_t, \mathbf{v}_t) = p(\mathbf{w}_t)p(\mathbf{v}_t|\mathbf{w}_t)$. 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.

3. Experiments

In this section, we present performance comparisons of our model with neural and classical codecs across multiple evaluation datasets. Ablations study about our new dataset, different architecture components and efficiency optimization is elucidated in Appendix D and H.

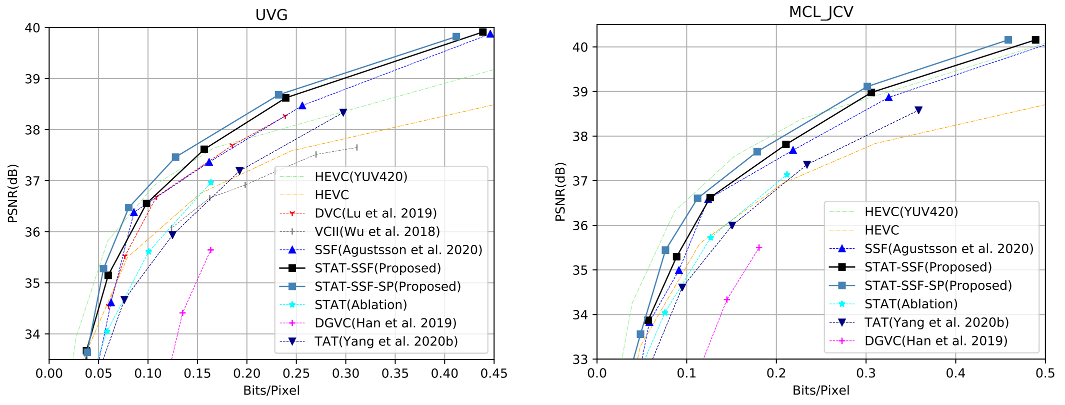
3.1. Training Datasets

Vimeo-90k (Xue et al., 2019) consists of 90,000 clips of 7 frames at 448x256 resolution collected from `vimeo.com`. As this dataset has been used in previous works (Lu et al., 2019; Yang et al., 2020a; Liu et al., 2019), it provides a benchmark for comparing models.

3.2. Evaluation Scheme

We evaluate compression performance on **UVG** (Mercat et al., 2020) and **MCL_JCV** (Wang et al., 2016) datasets, which both consist of YUV420 format RAW videos. UVG is widely used for testing the HEVC codec and contains seven 1080p videos at 120fps with smooth and mild motions or stable camera moving. MCL_JCV contains thirty 1080p videos at 30fps, which are generally more diverse, with a higher degree of motion and camera movement.

We compute the bit rate (bits-per-pixel, BPP) and the reconstruction quality as measured in PSNR, averaged across all frames. We note that PSNR is a more challenging metric than MS-SSIM (Wang et al., 2003) for learned codecs (Lu et al., 2019; Agustsson et al., 2020; Habibian et al., 2019; Yang et al., 2020a,c).



(a) (b)

Figure 2: **Rate-Distortion Performance** of various models and ablations. Results are evaluated on **(a)** UVG and **(b)** MCL_JCV datasets. All the neural-based models (except VCII (Wu et al., 2018)) are trained on Vimeo-90k (Xue et al., 2019). STAT-SSF (proposed) achieves the best performance.

3.3. Baseline Analysis

To provide a benchmark comparison with baseline models, listed in Table 1, we primarily focus on the Vimeo-90k (Xue et al., 2019) training dataset. In Figure 2(a), we compare our proposed models (STAT-SSF, STAT-SSF-SP) with previous neural codecs and the classical codec, HEVC, on the UVG evaluation dataset. Our models provide superior performance at bitrates ≥ 0.12 bits/pixel, outperforming state-of-the-art models SSF (Agustsson et al., 2020), as well as DVC (Lu et al., 2019). Finally, we note that, as expected, our proposed STAT model improves over TAT (Yang et al., 2020b) by adding stochasticity to the autoregressive transform.

The diversity of the MCL_JCV dataset provides a more challenging evaluation benchmark. However, our STAT-SSF model still maintains a competitive edge over other baseline methods. Our structured prior model, STAT-SSF-SP, provides further performance improvements over STAT-SSF. This suggests that our structured prior may further help capture intensive motion and camera moving information.

4. Discussion

We provide a unifying perspective on sequential video compression and temporal autoregressive flows (Marino et al., 2020), and elucidate the relationship between the two in terms of their underlying generative hierarchy. From this perspective, we consider several video compression methods, particularly a state-of-the-art method Scale-Space-Flow (Agustsson et al., 2020), as instantiations of a more general stochastic temporal autoregressive transform. Further, we provide a new high-resolution video dataset. Together, we hope that this new perspective and dataset will drive further progress in the nascent yet highly impactful field of learned video compression.

References

Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations*, 2017.

Fabrice Bellard. Bpg image format, 2014. URL https://bellard.org/bpg/bpg_spec.txt.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

Cassius C Cutler. Differential quantization of communication signals, July 29 1952. US Patent 2,605,361.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *IEEE International Conference on Computer Vision*, pages 7033–7042, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Haojie Liu, Lichao Huang, Ming Lu, Tong Chen, Zhan Ma, et al. Learned video compression via joint spatial-temporal correlation exploration. *arXiv preprint arXiv:1912.06348*, 2019.

Salvator Lombardo, Jun Han, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pages 9287–9298, 2019.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

Joseph Marino, Lei Chen, Jiawei He, and Stephan Mandt. Improving sequential latent variable models with autoregressive flows. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–16, 2020.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *ACM Multimedia Systems Conference*, pages 297–302, 2020.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Oren Rippel, Sanjay Nair, Carissa Lew, S. Branson, Alexander G. Anderson, and Lubomir D. Bourdev. Learned video compression. *IEEE International Conference on Computer Vision*, pages 3453–3462, 2019.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *International Conference on Learning Representations*, 2017.

Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *IEEE International Conference on Image Processing*, pages 1509–1513. IEEE, 2016.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402. Ieee, 2003.

Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *European Conference on Computer Vision*, pages 416–431, 2018.

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8): 1106–1125, 2019.

Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020a.

Ruihan Yang, Yibo Yang, Joseph Marino, Yang Yang, and Stephan Mandt. Deep generative video compression with temporal autoregressive transforms. *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2020b.

Yang Yang, Guillaume Sautière, J Jon Ryu, and Taco S Cohen. Feedback recurrent autoencoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3347–3351. IEEE, 2020c.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *arXiv preprint arXiv:2006.04240*, 2020d.

Appendix A. Qualitative Result

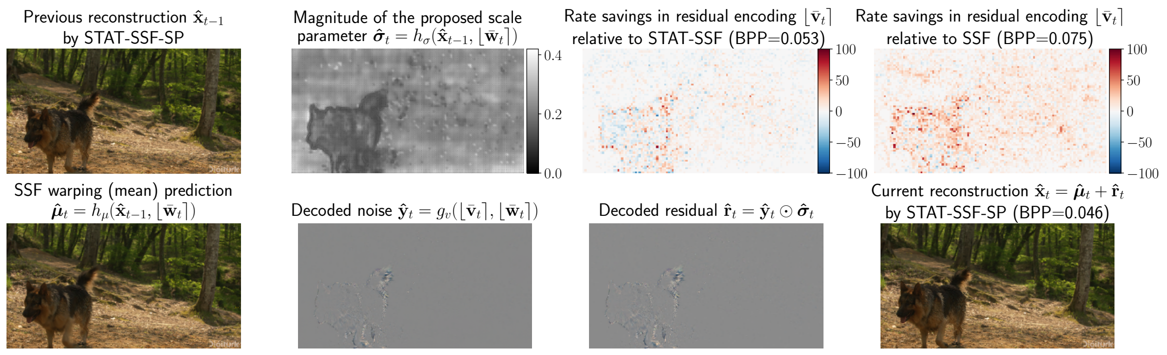
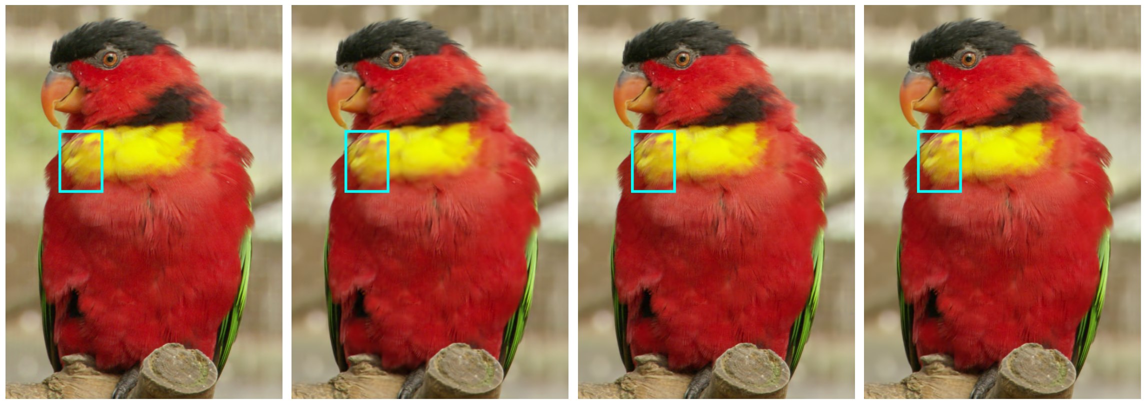

Previous reconstruction $\hat{\mathbf{x}}_{t-1}$ by STAT-SSF-SP
Magnitude of the proposed scale parameter $\hat{\boldsymbol{\sigma}}_t = h_\sigma(\hat{\mathbf{x}}_{t-1}, \lfloor\bar{\mathbf{w}}_t\rceil)$
0.4
0.2
0.0
Rate savings in residual encoding $\lfloor\bar{\mathbf{v}}_t\rceil$ relative to STAT-SSF (BPP=0.053)
100
50
0
−50
−100
Rate savings in residual encoding $\lfloor\bar{\mathbf{v}}_t\rceil$ relative to SSF (BPP=0.075)
100
50
0
−50
−100
SSF warping (mean) prediction $\hat{\boldsymbol{\mu}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}, \lfloor\bar{\mathbf{w}}_t\rceil)$
Decoded noise $\hat{\mathbf{y}}_t = g_v(\lfloor\bar{\mathbf{v}}_t\rceil, \lfloor\bar{\mathbf{w}}_t\rceil)$
Decoded residual $\hat{\mathbf{r}}_t = \hat{\mathbf{y}}_t \odot \hat{\boldsymbol{\sigma}}_t$
Current reconstruction $\hat{\mathbf{x}}_t = \hat{\boldsymbol{\mu}}_t + \hat{\mathbf{r}}_t$ by STAT-SSF-SP (BPP=0.046)

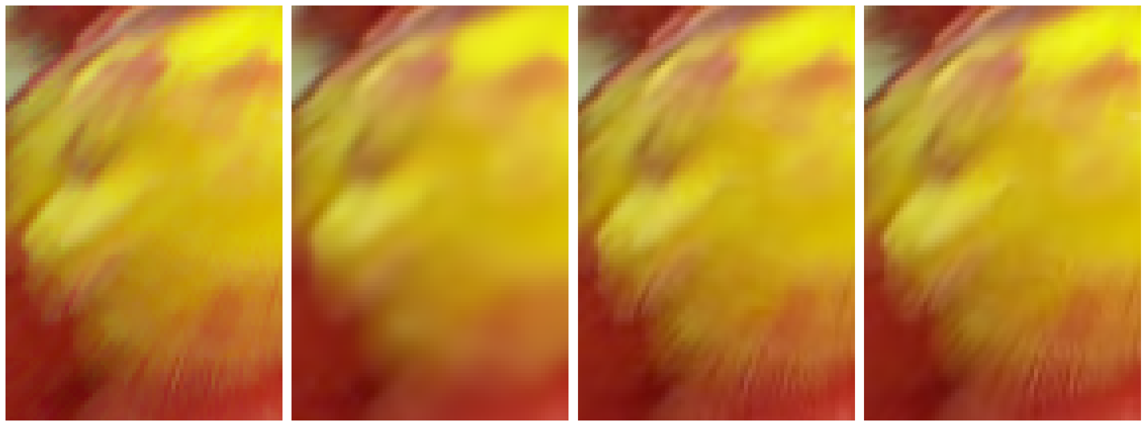
Figure 3: **Visualizing the proposed STAT-SSF-SP model** on one frame of UVG video "ShakeNDry". Two methods in comparison, STAT-SSF (proposed) and SSF (Agustsson et al., 2020), have comparable reconstruction quality to STAT-SSF-SP but higher bit-rate; the (BPP, PSNR) for STAT-SSF-SP, STAT-SSF, and SSF are (0.046, 36.97), (0.053, 36.94), and (0.075, 36.97), respectively. In this example, the warping prediction $\hat{\boldsymbol{\mu}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}, \lfloor\bar{\mathbf{w}}_t\rceil)$ incurs 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 $\lfloor\bar{\mathbf{w}}_t\rceil$ in the three methods. The proposed scale parameter $\hat{\boldsymbol{\sigma}}_t$ gives the model extra flexibility when combining the noise $\hat{\mathbf{y}}_t$ (decoded from $(\lfloor\bar{\mathbf{v}}_t\rceil, \lfloor\bar{\mathbf{w}}_t\rceil)$) with the warping prediction $\hat{\boldsymbol{\mu}}_t$ (decoded from $\lfloor\bar{\mathbf{w}}_t\rceil$ only) to form the reconstruction $\hat{\mathbf{x}}_t = \hat{\boldsymbol{\mu}}_t + \hat{\boldsymbol{\sigma}}_t \odot \hat{\mathbf{y}}_t$: 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 (with similar reconstruction quality), as illustrated in the third and fourth figures in the top row.

Appendix B. Low-Latency Sequential Compression

We specialize Eq. 1 to make it suitable for low-latency video compression, widely used in both conventional and recent neural codecs (Rippel et al., 2019; Agustsson et al., 2020). To this end, we encode and decode individual frames $\mathbf{x}_t$ in sequence. Given the ground truth current frame $\mathbf{x}_t$ and the previously reconstructed frames $\hat{\mathbf{x}}_{<t}$, the encoder is restricted to be



(*a*)



(*b*)

Figure 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; structured prior (STAT-SSF-SP) reduces the bit-rate further. Left to right: **HEVC** BPP=0.087 PSNR=38.099dB; **SSF** BPP=0.092, PSNR=37.44; **STAT-SSF(ours)** BPP=0.0768 PSNR=38.108; **STAT-SSF-SP(ours)** BPP=0.0551 PSNR=38.0975

of the form $\bar{\mathbf{z}}_t = f(\mathbf{x}_t, \hat{\mathbf{x}}_{<t})$, and similarly the decoder computes reconstruction sequentially based on previous reconstructions and the current encoding, $\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{<t}, \lfloor \bar{\mathbf{z}}_t \rceil))$. Existing codecs usually condition on a *single* reconstructed frame, substituting $\hat{\mathbf{x}}_{<t}$ by $\hat{\mathbf{x}}_{t-1}$ in favour of efficiency. In the language of 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 $p(\mathbf{x}_t|\mathbf{z}_{\leq t}) = \mathcal{N}(\hat{\mathbf{x}}_t, \frac{\beta}{2\log 2}\mathbf{I})$; 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 $\mathbf{z}_{<t}$. As we show, all sequential compression approaches considered in this work follow this paradigm, and the form of the reconstruction transform $\hat{\mathbf{x}}$ determines the lowest hier-

archy of the corresponding generative process of video $\mathbf{x}$. We note that this scheme does not require future frames $\hat{\mathbf{x}}_{>t}$ to help compress frame $\hat{\mathbf{x}}_t$ and this is so-called *low-latency*, which is widely used in some real-time applications such as live streaming.

Appendix C. Comparison Table

Table 1: **Overview of models and codecs**.

Model Name	**Category**	**Vimeo-90k**	**Youtube-NT**	**Remark**
STAT-SSF	Proposed	✓	✓	Proposed autoregressive transform with efficient scale-space flow model
STAT-SSF-SP	Proposed	✓	✗	Proposed autoregressive transform with efficient scale-space flow model and structured prior
SSF	Baseline	✓	✓	Agustsson et al. 2020 CVPR
DVC	Baseline	✓	✗	Lu et al. 2019 CVPR
VCII	Baseline	✗	✗	Wu et al. 2018 ECCV (Trained on Kinectics dataset (Carreira and Zisserman, 2017))
DGVC	Baseline	✓	✗	Han et al. 2019 NeurIPS without using future frames
TAT	Baseline	✓	✗	Yang et al. 2020b ICML Workshop
HEVC(RGB)	Baseline	N/A	N/A	State-of-the-art conventional codec with RGB color format
STAT	Ablation	✓	✓	Replace scale space flow in STAT-SSF with neural network
SSF-SP	Ablation	✗	✓	Scale space flow model with structured prior

Appendix D. New Dataset

YouTube-NT. This is our new dataset. We collected 8,000 nature videos and movie/video-game trailers from `youtube.com` and processed them into 300k high-resolution (720p) clips, which we refer to as YouTube-NT. In contrast to existing datasets (Carreira and Zisserman, 2017; Xue et al., 2019), we provide YouTube-NT in the form of customizable scripts to enable future compression research. Table 2 compares the current version of YouTube-NT with Vimeo-90k (Xue et al., 2019) and with Google's proprietary training dataset (Agustsson et al., 2020). In Figure 5(b), we display the evaluation performance of the SSF model architecture after training on each dataset.

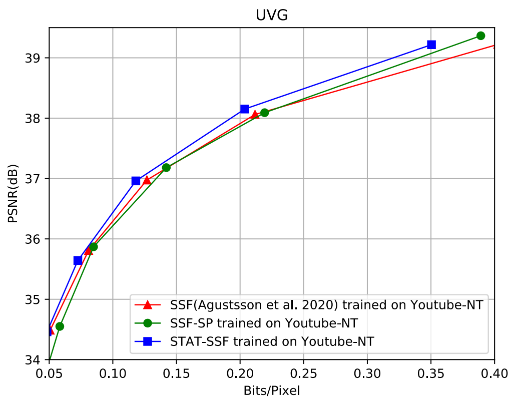
Table 2: **Overview of Training Datasets**.

Dataset name	**Clip length**	**Resolution**	**# of clips**	**# of videos**	**Public**	**Configurable**
Vimeo-90k	7 frames	448x256	90,000	5,000	✓	✓✗
YouTube-NT (**ours**)	6-10 frames	1280x720	300,000	8,000	✓	✓
Agustsson 2020 et al.	60 frames	1280x720	700,000	700,000	✗	✗

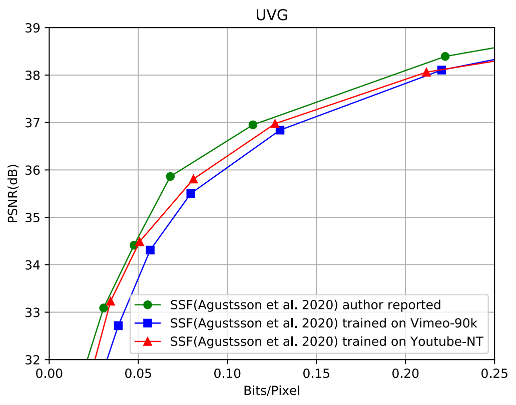
Appendix E. Ablation

Using the baseline SSF (Agustsson et al., 2020) model and YouTube-NT training dataset, we demonstrate the improvements of our proposed components, stochastic temporal autoregressive transform (STAT) and structured prior (SP), evaluated on UVG. As shown in Figure 5(a), STAT improves performance to a greater degree than SP, consistent with the results in Section 3.3 Figure 2(a).

To quantify the effect of the training dataset on performance, we compare performance on UVG for the SSF model architecture after training on Vimeo-90k (Xue et al., 2019) and YouTube-NT. We also compare with the reported results from Agustsson et al. (2020),



(a)



(b)

Figure 5: **Ablations & Comparisons**. **(a)** An ablation study on our proposed components. **(b)** Performance of SSF (Agustsson et al., 2020) trained on different datasets. Both sets of results are evaluated on UVG.

trained on a larger proprietary dataset. This is shown in Figure 5(*b*), where we see that training on YouTube-NT improves evaluation performance over Vimeo-90k, in some cases bridging the gap with the performance from the larger proprietary training dataset of Agustsson et al. (2020). At higher bitrate, the model optimized with Vimeo-90k(Xue et al., 2019) tends to have a similar performance with YouTube-NT. This is likely because YouTube-NT currently only covers 8000 videos, limiting the diversity of the short clips.

Appendix F. Command for HEVC codec

To avoid `FFmpeg` package taking the advantage of the input file color format (YUV420), we first need to dump the `video.yuv` file to a sequence of lossless `png` files:

```
ffmpeg -i video.yuv -vsync 0 video/%d.png
```

Then we use the default low-latency setting in `ffmpeg` to compress the dumped `png` sequences:

```
ffmpeg -y -i video/%d.png -c:v libx265 -preset medium \
    -x265-params bframes=0 -crf {crf} video.mkv
```

where `crf` is the parameter for quality control.

Appendix G. Training schedule

Training time is about four days on an NVIDIA Titan RTX. Similar to Agustsson et al. (2020), we use the Adam optimizer (Kingma and Ba, 2014), training the models for 1,050,000 steps. The initial learning rate of 1e-4 is decayed to 1e-5 after 900,000 steps, and we increase the crop size to 384x384 for the last 50,000 steps. All models are optimized using MSE loss.

Appendix H. Efficient scale-space-flow

Agustsson et al. (2020) leverages a simple implementation of scale-space flow by convolving the previous reconstructed frame $\hat{x}_{t-1}$ with a sequence of Gaussian kernel $\sigma^2 = \{0, \sigma_0^2, (2\sigma_0)^2, (4\sigma_0)^2, (8\sigma_0)^2, (16\sigma_0)^2\}$. However, this may lead to a large kernel size when σ is increasing and significantly reduce the efficiency. For example, Gaussian kernel with $\sigma^2 = 256$ usually requires kernel size 97x97 to avoid artifact (usually $kernel\_size = (6*\sigma+1)^2$). To alleviate the problem, we leverage an efficient version of Gaussian scale-space by using Gaussian pyramid with upsampling. In our implementation, we use $\sigma^2 = \{0, \sigma_0^2, \sigma_0^2+(2\sigma_0)^2, \sigma_0^2+(2\sigma_0)^2+(4\sigma_0)^2, \sigma_0^2+(2\sigma_0)^2+(4\sigma_0)^2+(8\sigma_0)^2, \sigma_0^2+(2\sigma_0)^2+(4\sigma_0)^2+(8\sigma_0)^2+(16\sigma_0)^2\}$, because by using Gaussian pyramid, we can always use Gaussian kernel with $\sigma = \sigma_0$ to consecutively blur and downsample the image to build a *pyramid*. At the final step, we only need to upsample all the downsampled images to the original size to approximate a scale-space 3D tensor.

```
Result: sst: Scale-space 3D tensor
Input: input input image; $\sigma_0$ base scale; $M$ scale depth sst = [input] kernel = Create_Gaussian_Kernel($\sigma_0$) for i=0 to M-1 do
    input = GaussianBlur(input, kernel) if i == 0 then
        sst.append(input)
    else
        tmp = input for j=0 to i-1 do
            tmp = UpSample2x(tmp); {step upsampling for smooth interpolation}
        end
        ssv.append(tmp)
    end
    input = DownSample2x(input)
end
return Concat(sst)
```

Algorithm 1: An efficient algorithm to build a scale-space 3D tensor

Appendix I. Architecture

Figure 6 illustrates the low-level encoder, decoder and hyper-en/decoder modules used in our proposed STAT-SSF and STAT-SSF-SP models, as well as in the baseline TAT and SSF models, based on Agustsson et al. (2020). Figure 7 shows the encoder-decoder flowchart for $\mathbf{w}_t$ and $\mathbf{v}_t$ separately, as well as their corresponding entropy models (priors), in the STAT-SSF-SP model.

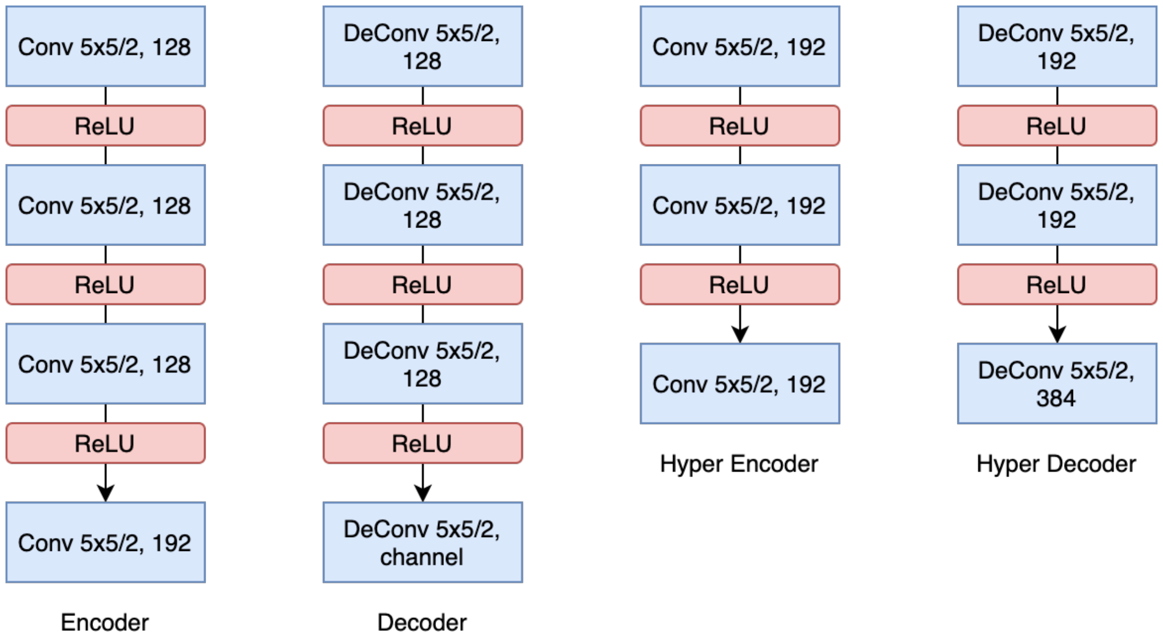
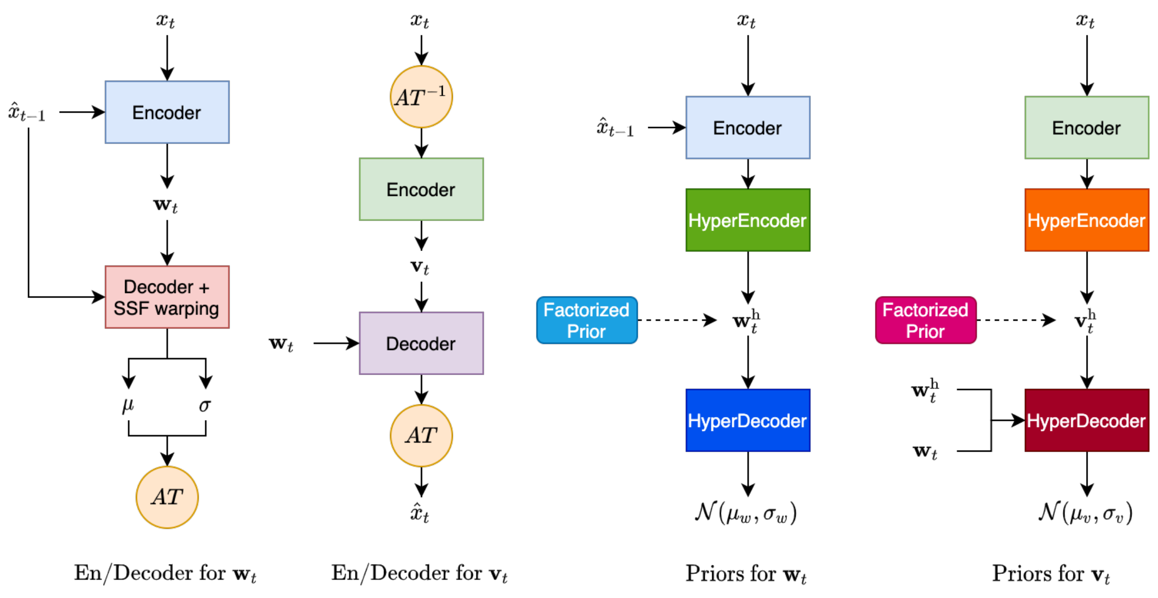


Figure 6: Backbone module architectures, where “5x5/2, 128” means 5x5 convolution kernel with stride 2; the number of filters is 128.



x_t
$\hat{x}_{t-1}$
Encoder
$\mathbf{w}_t$
Decoder + SSF warping
μ
σ
AT
AT^{-1}
Encoder
$\mathbf{v}_t$
Decoder
$\mathbf{w}_t$
AT
$\hat{x}_t$
HyperEncoder
Factorized Prior
$\mathbf{w}_t^{\mathrm{h}}$
HyperDecoder
$\mathcal{N}(\mu_w, \sigma_w)$
$\mathbf{v}_t^{\mathrm{h}}$
$\mathcal{N}(\mu_v, \sigma_v)$
En/Decoder for $\mathbf{w}_t$
En/Decoder for $\mathbf{v}_t$
Priors for $\mathbf{w}_t$
Priors for $\mathbf{v}_t$

Figure 7: Computational flowchart for the proposed STAT-SSF-SP model. The left two subfigures show the decoder and encoder flowcharts for $\mathbf{w}_t$ and $\mathbf{v}_t$, respectively, with "AT" denoting autoregressive transform. The right two subfigures show the prior distributions that are used for entropy coding $\mathbf{w}_t$ and $\mathbf{v}_t$, respectively, with $p(\mathbf{w}_t, \mathbf{w}_t^{\mathrm{h}}) = p(\mathbf{w}_t^{\mathrm{h}})p(\mathbf{w}_t|\mathbf{w}_t^{\mathrm{h}})$, and $p(\mathbf{v}_t, \mathbf{v}_t^{\mathrm{h}}|\mathbf{w}_t, \mathbf{w}_t^{\mathrm{h}}) = p(\mathbf{v}_t^{\mathrm{h}})p(\mathbf{v}_t|\mathbf{v}_t^{\mathrm{h}}, \mathbf{w}_t, \mathbf{w}_t^{\mathrm{h}})$, with $\mathbf{w}_t^{\mathrm{h}}$ and $\mathbf{v}_t^{\mathrm{h}}$ denoting hyper latents (see (Agustsson et al., 2020) for a description of hyper-priors); note that the priors in the SSF and STAT-SSF models (without the proposed structured prior) correspond to the special case where the HyperDecoder for $\mathbf{v}_t$ does not receive $\mathbf{w}_t^{\mathrm{h}}$ and $\mathbf{w}_t$ as inputs, so that the entropy model for $\mathbf{v}_t$ is independent of $\mathbf{w}_t$: $p(\mathbf{v}_t, \mathbf{v}_t^{\mathrm{h}}) = p(\mathbf{v}_t^{\mathrm{h}})p(\mathbf{v}_t|\mathbf{v}_t^{\mathrm{h}})$.