HIERARCHICAL AUTOREGRESSIVE MODELING FOR NEURAL VIDEO COMPRESSION

Anonymous authors
Paper under double-blind review

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

With recent advances, deep generative modeling has seen a surge in applications, including neural-based compression. Generative models have already demonstrated empirical improvements in image compression, outperforming classical codecs, such as BPG (Bellard, 2014). In contrast, the less developed area of neural-based 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. Compared with hand-designed codecs, learnable codecs can be tuned to specific data modalities, providing a promising approach for streaming specialized content, such as sporting events or video chatting. Therefore, improving neural-based video compression is a necessity for dealing with the ever-growing amount of video content being created.

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. 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 flows.

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 a recently proposed neural video codec based on scale-space warping (Agustsson et al., 2020). By interpreting this technique as a specific instantiation of a flow, we propose various alternatives and improvements, drawing on insights from generative modeling and autoregressive flows.

In more detail, 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. Following the classical predictive coding approach to video compression, this model uses motion estimation to predict the frame being compressed, and further compresses the residual obtained by subtraction. We show that incorporating a learnable scaling transform improves performance. Augmenting a shift transform by scale-then-shift is 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 will be made available through executable scripts. As previous works (Agustsson et al., 2020) did not provide their training datasets, YouTube-NT will be a useful resource for making and comparing further progress in this field.

2 Related Work

We divide relevant 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. Early works proposed by Toderici et al. (2017) and Johnston et al. (2018) leveraged LSTMs to model spatial correlations of the pixels within an image. Theis et al. (2017) first proposed an autoencoder architecture for image compression and used the straight-through estimator (Bengio et al., 2013) for learning a discrete latent representation. The connection to *probabilistic* generative models was drawn by Ballé et al. (2017), who firstly applied variational autoencoders (VAEs) (Kingma & Welling, 2013) to image compression. In subsequent work, Ballé et al. (2018) encoded images with a two-level VAE architecture involving a scale hyper-prior, which can be further improved by autoregressive structures (Minnen et al., 2018; Minnen & Singh, 2020) or by optimization at encoding time (Yang et al., 2020d). Yang et al. (2020e) and Flamich et al. (2019) demonstrated competitive image compression performance without a pre-defined quantization grid.

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 Wu et al. (2018); Djelouah et al. (2019) and Han et al. (2019) performed video compression by predicting future frames using a recurrent neural network, whereas Chen et al. (2019) and Chen et al. (2017) 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. Lu et al. (2019) adopted a hybrid architecture that combined a pre-trained Flownet (Dosovitskiy et al., 2015) and residual compression, which leads to an elaborate training scheme. Habibian et al. (2019) and Liu et al. (2019) 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 Agustsson et al. (2020), 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 (Chung et al., 2015). Denton & Fergus (2018) and Babaeizadeh et al. (2017) used VAE-based stochastic models in conjunction with LSTMs to model dynamics. Li & Mandt (2018) introduced both local and global latent variables for learning disentangled representations in videos. Other video generation models used generative adversarial networks (GANs) (Vondrick et al., 2016; Lee et al., 2018) or autoregressive models and normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2014; 2016; Kingma & Dhariwal, 2018; Kingma et al., 2016; Papamakarios et al., 2017). Recently, Marino et al. (2020) proposed to combine latent variable models with autoregressive flows for modeling dynamics at different levels of abstraction, which inspired our models and viewpoints.

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

3 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.

3.1 BACKGROUND

We first review VAE-based compression schemes (Ballé et al., 2017; Theis et al., 2017) and formulate existing low-latency video codecs in this framework. We also review flow-based models.

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 the shortest description $\mathcal{R}$ of $\mathbf{x}_{1:T}$ without exceeding a certain level of 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 $\bar{\mathbf{z}}_{1:T} = f(\mathbf{x}_{1:T})$. Using a parameter β, both terms can be weighted against each other:

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

We adopt the end-to-end compression approach of Ballé et al. (2017); Han et al. (2019), which approximates the rounding operations ($\lfloor \bullet \rceil$) 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 objective then corresponds to the VAE objective:

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

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.

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 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 hierarchy of the corresponding generative process of video $\mathbf{x}$.

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:

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

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 3.3, we will not exploit this aspect in compression but rather draw on its expressiveness in modeling conditional likelihoods.

3.2 A General Framework for Generative Video Coding

We now describe a general framework that captures several existing low-latency 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 $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$.

This stochastic decoder model has several advantages over existing generative models for compression, such as simpler flows or sequential VAEs. First, the stochastic autoregressive transform $h_\mu(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$ involves a latent variable $\mathbf{w}_t$ and is therefore more expressive than a deterministic transform (Schmidt & Hofmann, 2018; Schmidt et al., 2019). Second, compared to MAF, the additional nonlinear transform $g_v(\mathbf{v}_t, \mathbf{w}_t)$ enables more expressive residual noise, reducing the burden on the entropy model. Finally, as visualized in Figure 2, the scale parameter $h_\sigma(\hat{\mathbf{x}}_{t-1}, \mathbf{w}_t)$ 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 (Dinh et al., 2014; 2016).

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.

3.3 Example Models and Extensions

Next, we will 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 in Figure 1(a), the decoder g implements a deterministic scale-shift autoregressive transform on 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.

The TAT decoder is a simple version of the more general stochastic autoregressive transform in Eq 3, where h_μ and h_σ lack latent variables. Indeed, interpreting the probabilistic generative process of $\hat{\mathbf{x}}$, TAT implements the model proposed by Marino et al. (2020), as the transform from $\mathbf{y}$ to $\hat{\mathbf{x}}$ is a MAF. However, the generative process corresponding to compression (reviewed in Section 3.1) 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 an autoregressive flow. Regardless, 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 (Yang et al., 2020b).

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 estimated motion information used in warping prediction, and $\mathbf{v}$ helps capture residual error not predicted by warping.

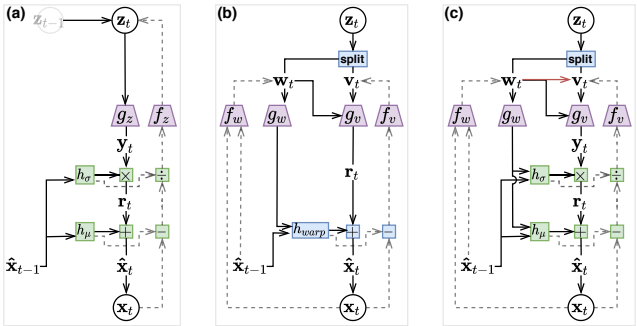


Figure 1: **Model Diagrams**. Graphical models underlying the generative and inference procedures for current frame $\mathbf{x}_t$ of various neural video compression methods. Random variables are shown in circles, all other quantities are deterministically computed; solid and dashed arrows describe computational dependency 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.

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 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. Lu et al. (2019) only makes use of $\mathbf{v}_t$ in the residual decoder g_v, and performs simple 2D warping by bi-linear interpretation; SSF (Agustsson et al., 2020) 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 compute 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 Figure 1(b).

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 w_t that characterizes the transform. In principle, more flexible transforms 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.

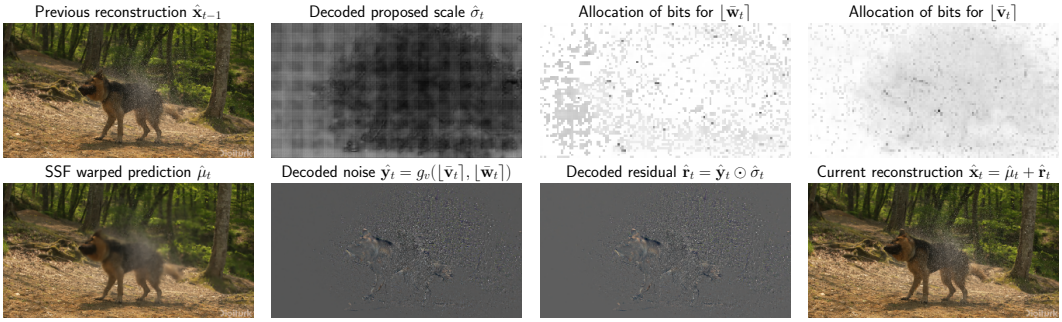


Figure 2: **Qualitative visualization** of the proposed stochastic autoregressive transform on "ShakeNDry" from the UVG dataset. The proposed scale $\hat{\boldsymbol{\sigma}}_t = h_\sigma(\hat{\mathbf{x}}_{t-1}, \lfloor \bar{\mathbf{w}}_t \rceil)$ can be seen to as a gating mechanism in the reconstruction transform g, switching off the contribution from the noise $\hat{\mathbf{y}}_t$ where it is costly and instead relying on the warping prediction $\hat{\boldsymbol{\mu}}_t = h_\mu(\hat{\mathbf{x}}_{t-1}, \lfloor \bar{\mathbf{w}}_t \rceil)$ decoded from $\mathbf{w}_t$.

4 EXPERIMENTS

In this section, we present performance comparisons of our model with neural and classical codecs across multiple training and evaluation datasets (see Table 2), as well as our new dataset, YouTube-NT. We demonstrate that our proposed model improvements yield state-of-the-art performance at high bitrates, and our new dataset results in improvements over publicly-available datasets. Our model architecture and training scheme largely follows that of Agustsson et al. (2020), incorporating the components described in the previous section. We refer to the Appendix A.4 for further details.

Differing from Agustsson et al. (2020), we implement a more efficient version of the scale-space module. Agustsson et al. (2020) leverages a method with multiple Gaussian kernels and increasing kernel sizes, which may lead to an efficiency problem. Our implementation utilizes a Gaussian pyramid and interpolation to avoid the issue. Pseudocode is available at Appendix A.3.

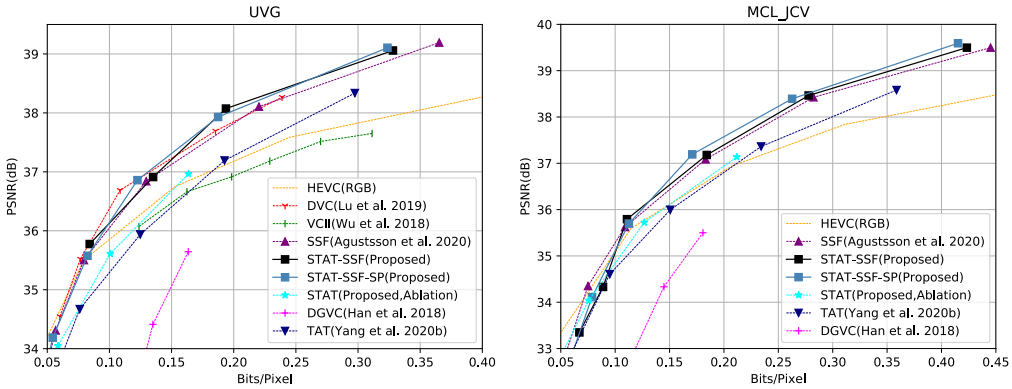
4.1 TRAINING DATASETS

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. While other publicly-available video datasets exist, e.g., Kinetics (Carreira & Zisserman, 2017), such datasets are lower-resolution and primarily intended for human action recognition. Accordingly, previous works that have trained using Kinetics generally report sub-par PSNR-bitrate performance (Wu et al., 2018; Habibian et al., 2019; Golinski et al., 2020). Agustsson et al. (2020) uses a significantly larger and higher-resolution dataset collected from `youtube.com`, however, this is not publicly available.

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 & Zisserman, 2017; Xue et al., 2019), we provide YouTube-NT in the form of customizable scripts to enable future compression research. Table 1 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 4b, we display the evaluation performance of the SSF model architecture after training on each dataset.

Table 1: **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	✗	✗



(a) (b)

Figure 3: **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.

4.2 Training and Evaluation Scheme

Training. All models are trained with three consecutive frames and batchsize 8, which are randomly selected from each clip, then randomly cropped to 256x256. We trained on MSE loss, following similar procedure to Agustsson et al. (2020) (Refer to Appendix A.2 for details).

Evaluation. 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 a more unstable camera.

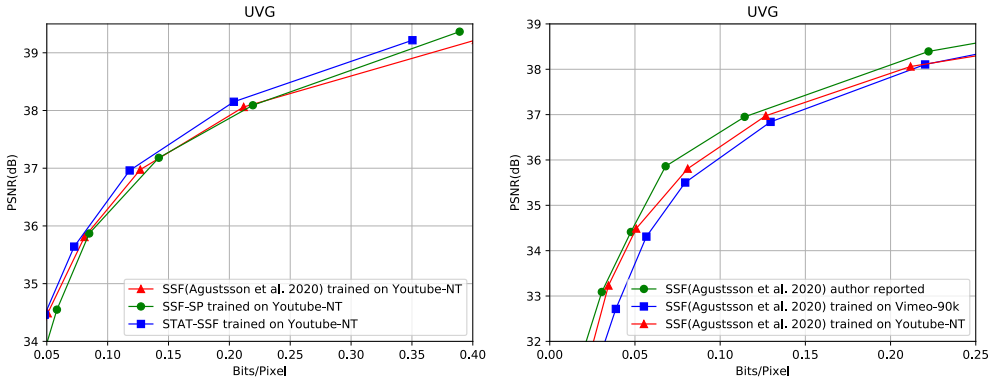
We compute the bit rate (bits-per-pixel, BPP) and the reconstruction quality as measured in PSNR, averaged across all frames. We note that PSNR is a more challenging metric than MS-SSIM (Wang et al., 2003) for learned codecs (Lu et al., 2019; Agustsson et al., 2020; Habibian et al., 2019; Yang et al., 2020a;c). We also note that current neural-based compression methods take video input in 8-bit RGB format (24bits/pixel), whereas `FFmpeg` also supports YUV420 format (12bits/pixel), allowing classical codecs to effectively halve the input bitrate on the test data. For fair comparison, we report rate-distortion performance for HEVC in RGB mode (Appendix A.1).

Table 2: **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 conditional 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 & 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 ESSF-AT with black box neural network
STAT-SP	Ablation	✗	✓	Efficient scale space flow model with conditional prior

4.3 Baseline Analysis

To provide a benchmark comparison with baseline models, listed in Table 2, we primarily focus on the Vimeo-90k (Xue et al., 2019) training dataset. In Figure 3a, 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), which leverages a more complicated model and a multi-stage training procedure. The classical



(a) (b)

Figure 4: **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.

codec, HEVC (RGB), barely outperforms the model proposed by Wu et al. (2018). 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, outperforming all other baselines at ≥ 0.12 bits/pixel. This suggests that our structured prior may further help capture intensive motion and camera moving information.

4.4 Ablation Analysis

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 4a, STAT improves performance to a greater degree than SP, consistent with the results in Section 4.3 Figure 3a.

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), trained on a larger proprietary dataset. This is shown in Figure 4b, 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.

5 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, which allows us to naturally extend the Scale-Space-Flow model and obtain improved rate-distortion performance on two public benchmark datasets. Further, we are providing scripts to generate a new high-resolution video dataset, YouTube-NT, which is substantially larger than current publicly-available datasets. 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8503–8512, 2020.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

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

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *In ICLR*, 2018.

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

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

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

T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma. Deepcoder: A deep neural network based video compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2017.

Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu. Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):566–576, 2019.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

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

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018.

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.

A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers. Neural inter-frame compression for video coding. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6420–6428, 2019.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. Compression without quantization. In *OpenReview*, 2019.

Adam Golinski, Reza Pourreza, Yang Yang, Guillaume Sautiere, and Taco S Cohen. Feedback recurrent autoencoder for video compression. *arXiv preprint arXiv:2004.04342*, 2020.

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

Jun Han, Salvator Lombardo, Christopher Schroers, and Stephan Mandt. Deep generative video compression. 2019.

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

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

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.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 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*, pp. 1–16, 2020.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 297–302, 2020.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. *arXiv preprint arXiv:2007.08739*, 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*, pp. 10771–10780, 2018.

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

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

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

Florian Schmidt and Thomas Hofmann. Deep state space models for unconditional word generation. In *Advances in Neural Information Processing Systems*, pp. 6158–6168, 2018.

Florian Schmidt, Stephan Mandt, and Thomas Hofmann. Autoregressive text generation beyond feedback loops. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3391–3397, 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.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pp. 613–621, 2016.

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 *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1509–1513. IEEE, 2016.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pp. 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 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 (IJCV)*, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Ruihan Yang, Yibo Yang, Joseph Marino, Yang Yang, and Stephan Mandt. Deep generative video compression with temporal autoregressive transforms. *ICML 2020 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 *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3347–3351. IEEE, 2020c.

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

Yibo Yang, Robert Bamler, and Stephan Mandt. Variational bayesian quantization. In *International Conference on Machine Learning*, 2020e.

A APPENDIX

A.1 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.

A.2 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 & 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.

A.3 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.

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

```
Result: ssv: Scale-space 3D tensor
Input: input input image; σ_0 base scale; M scale depth;
ssv = [input];
kernel = Create_Gaussian_Kernel(σ_0);
for i=0 to M-1 do
    input = GaussianBlur(input, kernel);
    if i == 0 then
        ssv.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(ssv)
```

A.4 ARCHITECTURE

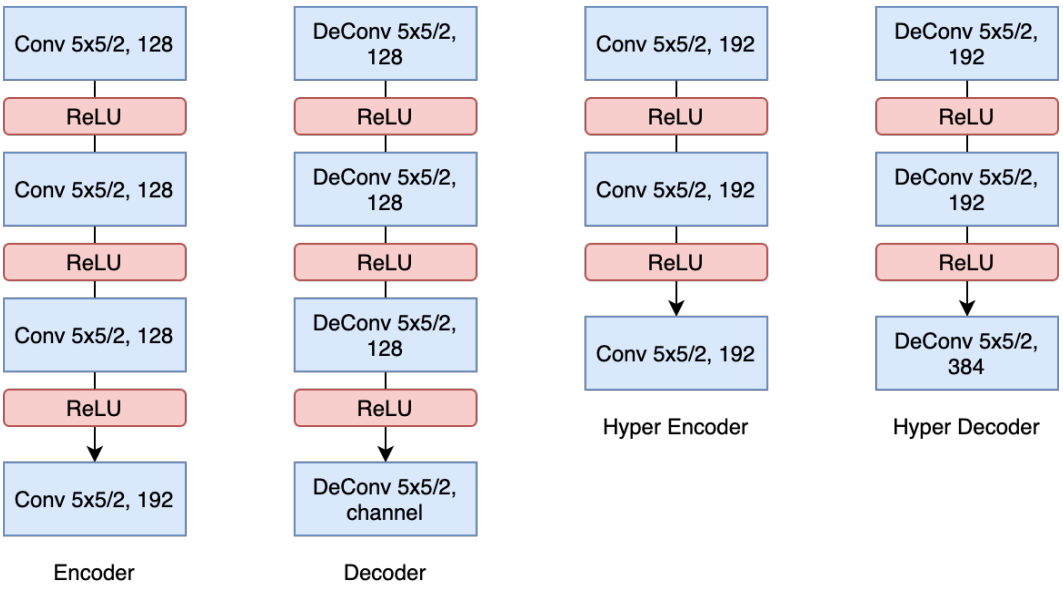


Figure 5: Backbone module architectures

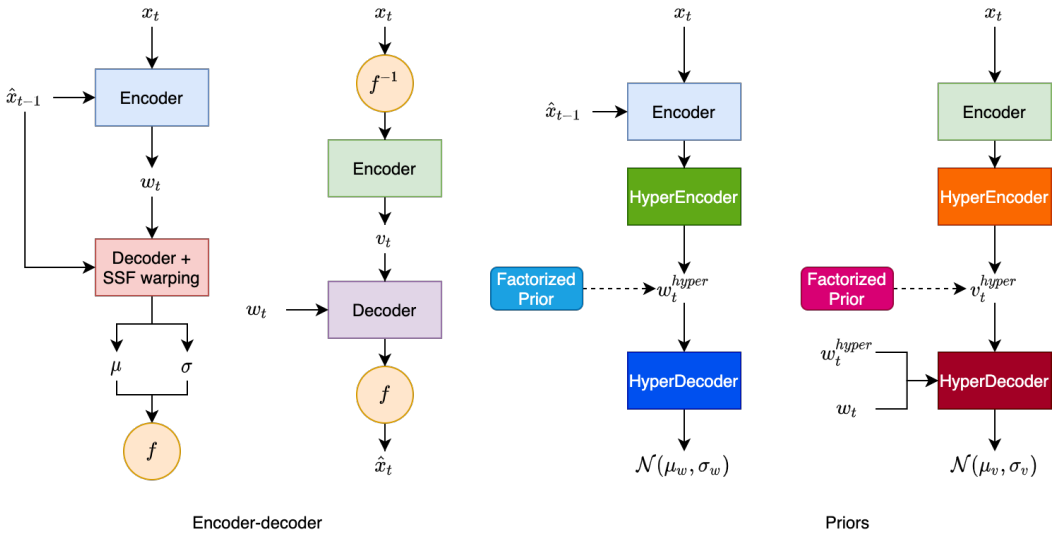


Figure 6: Model Flowchart