Video Compression With Rate-Distortion Autoencoders

Amirhossein Habibian, Ties van Rozendaal, Jakub M. Tomczak, Taco S. Cohen
Qualcomm AI Research*, Amsterdam, the Netherlands
{habibian, ties, jtomczak, tacos}@qti.qualcomm.com

Abstract

In this paper we present a deep generative model for lossy video compression. We employ a model that consists of a 3D autoencoder with a discrete latent space and an autoregressive prior used for entropy coding. Both autoencoder and prior are trained jointly to minimize a rate-distortion loss, which is closely related to the ELBO used in variational autoencoders. Despite its simplicity, we find that our method outperforms the state-of-the-art learned video compression networks based on motion compensation or interpolation. We systematically evaluate various design choices, such as the use of frame-based or spatio-temporal autoencoders, and the type of autoregressive prior.

In addition, we present three extensions of the basic method that demonstrate the benefits over classical approaches to compression. First, we introduce semantic compression*, where the model is trained to allocate more bits to objects of interest. Second, we study* adaptive compression*, where the model is adapted to a domain with limited variability,* e.g. *videos taken from an autonomous car, to achieve superior compression on that domain. Finally, we introduce* multimodal compression*, where we demonstrate the effectiveness of our model in joint compression of multiple modalities captured by non-standard imaging sensors, such as quad cameras. We believe that this opens up novel video compression applications, which have not been feasible with classical codecs.*

1. Introduction

In recent years, tremendous progress has been made in generative modelling. Although much of this work has been motivated by potential future applications such as model based reinforcement learning, *data compression* is a very natural application that has received comparatively little attention. Deep learning-based video compression in particular has only recently started to be explored [11, 33, 40]. This is remarkable because improved video compression would have a large economic impact: it is estimated that very soon, 80% of internet traffic will be in the form of video [12].

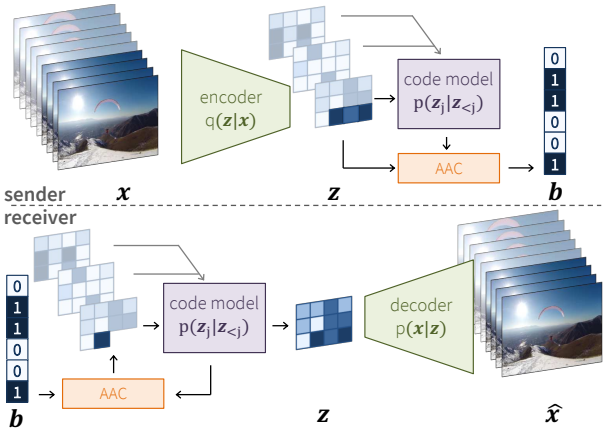


Figure 1: Overview of the proposed compression inference pipeline. The encoder encodes a sequence of frames $\mathbf{x}$ into a sequence of quantized latent variables $\mathbf{z}$. A code model $p(\mathbf{z}_t|\mathbf{z}_{<t})$ is used to transform $\mathbf{z}$ into a bitstream $\mathbf{b}$ using adaptive arithmetic coding (AAC). On the receiver side, the bitstream is used to reconstruct $\mathbf{z}$ which is then lossily decoded into $\hat{\mathbf{x}}$.

In this paper, we present a simple yet effective and theoretically grounded method for video compression that can serve as the basis for future work in this nascent area. Our model consists of off-the-shelf components from the deep generative modelling literature, namely autoencoders (AE) and autoregressive models (ARM). Despite its simplicity, the model outperforms all methods to which a direct comparison is possible, including substantially more complicated approaches.

On the theoretical side, we show that our method, as well as state-of-the-art image compression methods [28] can be interpreted as VAEs [25, 31] with a discrete latent space and a deterministic encoder. The VAE framework is an especially good fit for the problem of *lossy* compression, because it provides a natural mechanism for trading off *rate* and *distortion*, as measured by the two VAE loss terms [3]. However, as we will argue in this paper, it is not beneficial for the purpose of compression to use a stochastic encoder

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

(approximate posterior) as usually done in VAEs, because any noise added to the encodings results in increased bitrate without resulting in an improvement in distortion [18].

On the experimental side, we perform an extensive evaluation of several architectural choices, such as the use of 2D or 3D autencoders, and the type of autoregressive prior. Our best model uses a ResNet [17] autoencoder with 3D convolutions, and a temporally-conditioned gated PixelCNN [37] as prior. We benchmark our method against existing learned video compression methods, and show that it achieves better rate/distortion. We also find that our method outperforms the state-of-the-art traditional codecs when these are used with restricted settings, as it is done in previous work, but more work remains to be done before it can be claimed that these learned video compression methods suppress traditional codecs under optimal settings.

Additionally, we introduce several extensions of our method that highlight the benefits of using learned video codecs. In *semantic compression*, we bridge the gap between semantic video understanding and compression by learning to allocate more bits to objects from categories of interest, *i.e*., people. During training, we weight the rate and distortion losses to ensure a high quality reconstruction for regions of interest extracted by off-the-shelf object detection or segmentation networks, such as Mask R-CNN[16].

We also demonstrate *adaptive compression*, where the model is trained on a specific domain, either before or after deployment, to fine-tune it to the distribution of videos it is actually used for. We show that adaptive compression of footage from autonomous cars can result in large improvement in terms of rate and distortion. With classical codecs, finetuning for a given domain is often not feasible.

Finally, we show that our method is very effective in joint compression of multiple modalities, which exist in videos from depth, stereo, or multi view cameras. By utilizing the siginifcant redundancy, which exist in multimodal videos, our model outperforms HEVC/H.265 and AVC/H.264 by a factor of 4.

The main contributions of this paper are: *i)* We present a simple yet effective and theoretically grounded method for video compression that can serve as the basis for future work. *ii)* We clarify theoretically the relation between rate-distortion autoencoders and VAEs. *iii)* We introduce semantic compression to bridge the gap between semantic video understanding and compression. *iv)* We introduce adaptive compression to adapt a compression model to the domain of interest. *v)* We introduce multimodal compression to jointly compress multiple modalities, which exist in a video using a deep video compression network.

The rest of the paper is organized as follows. In the next section, we discuss related work on learned image and video compression. Then, in section 3, we discuss the theoretical framework of learned compression using rate-distortion autoencoders, as well as the relation to variational autoencoders. In section 4 we discuss our methodology in detail, including data preprocessing and autoencoder and prior architecture. We present experimental results in section 5, comparing our method to classical and learned video codecs, evaluating semantic compression, adaptive compression, and multimodal compression. Section 6 concludes the paper.

2. Related Work

Learned Image Compression Deep neural networks are the state-of-the-art in image compression outperforming all traditional compression algorithms such as BPG and JPEG2000. They often embed an input image into a low dimensional representation using fully convolutional [28] or recurrent networks [4, 22, 36]. The image representation is quantized by soft scalar quantization [2], stochastic binarization [36], or by adding uniform noise [5] to approximate the non-differentiable quantization operation. The discrete image representation can be further compressed by minimizing the entropy during [10, 28] or after training [5, 6, 26]. The models are typically trained to minimize the mean squared error between original and decompressed images or by using more perceptual metrics such as MS-SSIM [32] or adversarial loss [34].

The closest to us is the rate-distortion autoencoder proposed in [28] for image compression. We extend this work to video compression by: *i)* proposing a gated conditional autoregressive prior using 2D convolutions [37] with, optionally, a recurrent neural net for better entropy estimation over time, *ii)* encoding/decoding multiple frames by using 3D convolutions, *iii*) simplifying the model and training by removing the spatial importance map [26] and disjoint entropy estimation, without any loss on compression performance.

Learned Video Compression Video compression shares many similarities with image compression, but the large size of video data, and the very high degree of redundancy create new challenges [15, 30, 33, 40]. One of the first deep learning-based approaches proposes to model video autoregressively with a RNN-conditioned PixelCNN [23]. While being powerful and flexible, this model scales rather poorly to larger videos, and can only be used for lossless compression. Hence, we employ this method for lossless compression of latent codes, which are much smaller than the video itself. An extension of this method was proposed in [11] where blocks of pixels are modeled in an autoregressive fashion and the latent space is binarized like in [36]. The applicability of this approach is rather limited since it is still not very scalable, and introduces artifacts in the boundary between blocks, especially for low bit rates.

The method described in [40] compresses videos by first encoding key frames, and then interpolating them in a hi-

erarchical manner. The results are on par with AVC/H.264 when inter-frame compression is limited to only few (up to 12) frames. However, this method requires additional components to handle a context of the predicted frame. In our approach, we aim at learning these interactions through 3D convolutions instead. In [15] a stochastic variational compression method for video was presented. The model contains a separate latent variable for each frame, and for the inter-frame dependencies, and uses the prior proposed in [6]. By contrast, we use a simpler model with a single latent space, and use a deterministic instead of stochastic encoder.

Very recently the video compression problem was attacked by considering flow compression and residual compression [27, 33]. The additional components for flow and residual modeling allow to improve distortion in general, however, for low bit rates the proposed method is still outperformed by HEVC/H.265 on benchmark datasets. Nevertheless, we believe that these ideas are promising and may be able to further improve the result presented in this paper.

3. Rate-Distortion Autoencoders & VAEs

Our general approach to lossy compression is to learn a latent variable model in which the latent variables capture the important information that is to be transmitted, and from which the original input can be approximately reconstructed. We begin by defining a joint model of data $\mathbf{x}$ and *discrete* latent variables $\mathbf{z}$,

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) \tag{1}$$

In the next section we will discuss the specific form of $p_\theta(\mathbf{z})$ (the prior / code model) and $p_\theta(\mathbf{x}|\mathbf{z})$ (the likelihood / decoder), both of which will be defined in terms of deep networks, but for now we will consider them as general parameterized distributions.

Since the likelihood $\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}$ is intractable, one optimizes the variational bound [8, 38],

$$-\log p(\mathbf{x}) \leq E_q[-\log p(\mathbf{x}|\mathbf{z})] + \mathrm{KL}[q(\mathbf{z}|\mathbf{x})|p(\mathbf{z})], \tag{2}$$

where $q(\mathbf{z}|\mathbf{x})$ is a newly introduced approximate posterior. In the VAE [25, 31], one uses neural networks to parameterize both $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$, which can thus be thought of as the encoder and decoder part of an autoencoder.

The VAE is commonly interpreted as a regularized autoencoder, where the first term of the loss measures the reconstruction error and the KL term acts as a regularizer [25]. But the variational bound also has an interesting interpretation in terms of compression / minimum description length [10, 14, 18, 19, 20]. Under this interpretation, the first term of the rhs of Eq. 2 measures the expected number of bits required to encode $\mathbf{x}$ given that we know a sample $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$. More specifically, one can derive a code for $\mathbf{x}$ from the decoder distribution $p(\mathbf{x}|\mathbf{z})$, which assigns roughly $-\log p(\mathbf{x}|\mathbf{z})$ bits to $\mathbf{x}$ [13]. Averaged over q, one obtains the first term of the VAE loss (Eq. 2).

We note that in lossy compression, we do not actually encode $\mathbf{x}$ using $p(\mathbf{x}|\mathbf{z})$, which would allow lossless reconstruction. Instead, we only send $\mathbf{z}$ and hence refer to the first loss term as the distortion.

The second term of the bound (the KL) is related to the cost of coding the latents $\mathbf{z}$ coming from the encoder $q(\mathbf{z}|\mathbf{x})$ using an optimal code derived from the prior $p(\mathbf{z})$. Such a code will use about $-\log p(\mathbf{z})$ bits to encode $\mathbf{z}$. Averaging over the encoder $q(\mathbf{z}|\mathbf{x})$, we find that the average coding cost is equal to the cross-entropy between q and p:

$$E_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{z})] = \mathrm{CE}[q(\mathbf{z}|\mathbf{x})|p(\mathbf{z})]. \tag{3}$$

The cross-entropy is related to the KL via the relation $\mathrm{KL}[q|p] = \mathrm{CE}[q|p] - \mathrm{H}[q]$, where $\mathrm{H}[q]$ is the entropy of the encoder q. So the KL measures the coding cost, except that there is a discount worth $\mathrm{H}[q]$ bits: randomness coming from the encoder is free. It turns out that there is indeed a scheme, known as bits-back coding, that makes it possible to transmit $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ and get $\mathrm{H}[q]$ bits back, but this scheme is difficult to implement in practice, and can only be used in lossless compression [18].

Since we cannot use bits-back coding for lossy compression, the cross-entropy provides a more suitable loss than the KL. Moreover, when using discrete latents, the entropy $\mathrm{H}[q]$ is always non-negative, so we can add it to the rhs of Eq. 2 and obtain a valid bound. We thus obtain the rate-distortion loss

$$\mathcal{L}(\mathbf{x}) = E_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}|\mathbf{z}) - \beta \log p(\mathbf{z})], \tag{4}$$

where β is a rate-distortion tradeoff parameter.

Since the cross-entropy loss does *not* include a discount for the encoder entropy, there is a pressure to make the encoder more deterministic. Indeed, for a fixed $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$, the optimal solution for $q(\mathbf{z}|\mathbf{x})$ is a deterministic ("one hot") distribution that puts all its mass on the state $\mathbf{z}$ that minimizes $-\log p(\mathbf{x}|\mathbf{z}) - \beta \log p(\mathbf{z})$.

For this reason, we only consider deterministic encoders in this work. When using deterministic encoders, the rate-distortion loss (Eq. 4) is equivalent to the variational bound (Eq. 2), because (assuming discrete $\mathbf{z}$), we have $\mathrm{H}[q] = 0$ and hence $\mathrm{KL}[q|p] = \mathrm{CE}[q|p]$.

Finally, we note that limiting ourselves to deterministic encoders does not lower the best achievable likelihood, assuming a sufficiently flexible class of prior and likelihood. Indeed, given *any* fixed deterministic encoder q, we can still achieve the maximum likelihood by setting $p(\mathbf{z}) = \sum_{\mathbf{x}} p(\mathbf{x}) q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{x}) q(\mathbf{z}|\mathbf{x})$, where $p(\mathbf{x})$ is the true data distribution.

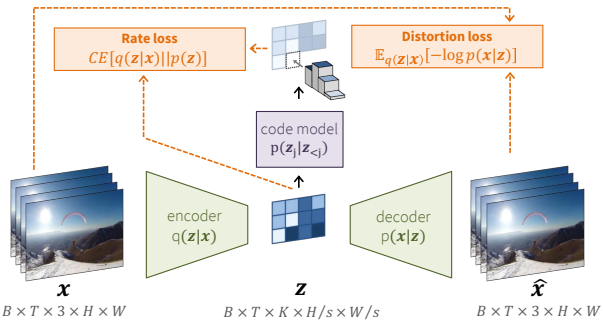
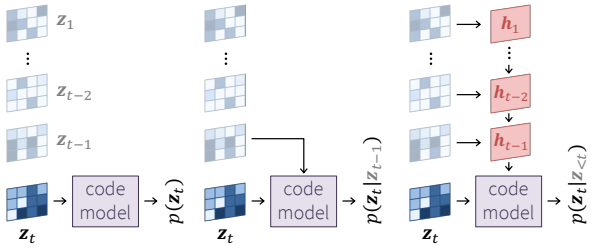


Figure 2: Training Rate-Distortion autoencoders. The rate loss is a measure for the expected coding cost, under the autoregressive code model, while the distortion loss expresses the reconstruction error.



(a) Unconditional (b) Frame-conditioned (c) GRU-conditioned

Figure 3: Proposals for temporal conditioning of prior.

4. Methodology

In the previous section, we have outlined the general compression framework using rate-distortion autoencoders.Here we will describe the specific models we use for encoder, code model, and decoder, as well as the data format, preprocessing, and loss functions.

4.1. Preprocessing

Our model processes chunks of video $\mathbf{x}$ of shape $T \times C \times H \times W$, where $T = 8$ denotes the number of frames, C denotes the number of channels (typically $C = 3$ for RGB), and H, W are the height and width of a crop, which we fix to 160 pixels in all of our experiments. The RGB values are not scaled, *i.e.*, they always lie in $\{0, 1, \ldots, 255\}$.

4.2. Autoencoder

The encoder takes as input a chunk of video $\mathbf{x}$ and produces a discrete latent code $\mathbf{z}$. If the input has shape $T \times C \times H \times W$, the latent code will have shape $T \times K \times H/s \times W/s$, where $K = 32$ is the number of channels in the latent space, and $s = 8$ is the total spatial stride of the encoder (so the latent space has spatial size $H/s = W/s = 160/8 = 20$). We do not use stride in the time dimension.

The encoder and decoder are based on the architecture presented by [28], which in turn is based on the architecture presented in [35]. The encoder and decoder are both fully convolutional models with residual connections [17], batchnorm [21], and ReLU nonlinearities. In the first two convolution layers of the encoder, this model uses filter size 5 and stride 2. The remaining layers are 5 residual blocks with two convolution layers per block, filter size 3, 128 channels, batchnorm, and ReLU nonlinearities. The final layer is a convolution with filter size 3, stride 2, and 32 output channels. The decoder is the reverse of this, and uses transposed convolutions instead of convolutions. More details on the architecture can be found in the supplementary material.

We will evaluate two versions of this autoencoder: one with 2D convolutions applied to each frame separately, and one with 3D spatio-temporal convolutions. To apply the 2D model to a video sequence, we simply fold the time axis into the batch axis before running the 2D AE.

The encoder network first outputs continuous latent variables $\tilde{\mathbf{z}}$, which are then quantized. The quantizer discretizes the coordinates of $\tilde{\mathbf{z}}$ using a learned codebook consisting of L centers, $\mathcal{C} = \{c_1, \ldots, c_L\}$, where $c_l \in \mathbb{R}$. In the forward pass, we compute $z_j = \arg\min_i |\tilde{z}_j - c_i|$ (where $j = (t, c, h, w)$ is a four dimensional multi-index). As a probability distribution, this corresponds to a one-hot $q(z_j|\mathbf{x})$ that puts all mass on the computed value z_j. Because the argmin is not differentiable, we use the gradient of a softmax in the backward pass, as in [7, 28]. We found this approach to be stable and effective during training.

On the decoder side, we replace $z_j \in \{1, \ldots, L\}$ by the corresponding codebook value c_{z_j}, to obtain an approximation of the original continuous representation $\tilde{\mathbf{z}}$. The resulting vector is then processed by the decoder to produce a reconstruction $\hat{\mathbf{x}}$. In a standard VAE, one might use $\hat{\mathbf{x}}$ as the mean of a Gaussian likelihood $p(\mathbf{x}|\mathbf{z})$, which corresponds to an L2 loss: $-\log p(\mathbf{x}|\mathbf{z}) \propto \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + const$. Instead, we use the MS-SSIM loss (discussed in Sec. 4.4), which corresponds to the unnormalized likelihood of the Boltzmann distribution, $\ln p(\mathbf{x}|\mathbf{z}) = \text{ms-ssim}(\mathbf{x}, \hat{\mathbf{x}}) - \ln C$, where $\ln C$ is the log-partition function treated as a constant, because it better reflects human subjective judgments of similarity.

4.3. Autoregressive Prior

Instead of naively storing / transmitting the latent variables $\mathbf{z}$ using $D \log_2 L$ bits (for a D-dimensional latent space with L states per variable), we encode the latents using the prior $p(\mathbf{z})$ in combination with adaptive arithmetic coding. For $p(\mathbf{z})$, we use a gated PixelCNN [37] over individual latent frames, optionally conditioned on past latent frames as in video pixel networks [23]. In Figure 3, we illustrate the three priors considered in this paper.

In the simplest case, we model each frame independently, i.e. $p(\mathbf{z}) = \prod_t p(z_t)$, where a latent frame z_t is modelled autoregressively as $p(z_t) = \prod_i p(z_{t,i}|z_{t,<i})$ by the PixelCNN. Here $i = (c, h, w)$ denotes a 3D multi-index over channels and spatial axes, and $z_{t,<i}$ denotes the ele-

(a) AVC/H.264 (0.037 BPP) (b) HEVC/H.265 (0.036 BPP) (c) Our model (0.037 BPP)

Figure 4: Compression results for the state-of-the-art traditional codecs, AVC/H.264 and HEVC/H.265, and our proposed model. On a similar bitrate, our model approachs these codecs while generatinng less artifacts.

ments that come before i in the autoregressive ordering.

A better prior is obtained by including temporal dependencies (Figure 3b). In this model, the prior is factorized as $p(\mathbf{z}) = \prod_t p(z_t|z_{t-1})$, where $p(z_t|z_{t-1}) = \prod_i p(z_{t,i}|z_{t,<i}, z_{t-1})$. Thus, the prediction for pixel $i = (c, h, w)$ in latent frame t is based on previous pixels in the same frame, as well as the whole previous frame z_{t-1}. The dependence on $z_{t,<i}$ is mediated by the masked convolutions of the PixelCNN architecture, whereas the dependence on the previous frame z_{t-1} is mediated by additional conditioning connections added to each layer, as in the original Conditional PixelCNN [37].

Conditioning on the previous frame may be limiting if long-range temporal dependencies are necessary. Hence, we also consider a model where a recurrent neural network (Gated Recurrent Units, GRU) summarizes all relevant information from past frames. The prior factorizes as $p(\mathbf{z}) = \prod_t p(z_t|z_{<t})$ with $p(z_t|z_{<t}) = \prod_i p(z_{t,i}|h_{t-1}, z_{t,<i})$, where h_{t-1} is the hidden state of a GRU that has processed latent frames $z_{<t}$. As in the frame-conditional prior, in the GRU-conditional prior, the dependency on $z_{t,<i}$ is mediated by the causal convolutions of the PixelCNN, and the dependency on h_t is mediated by conditioning connections in each layer of the PixelCNN.

4.4. Loss functions, encoding, and decoding

To measure distortion, we use the Multi-Scale Structural Similarity (MS-SSIM) loss [39]. This loss gives a better indication of the subjective similarity of $\hat{\mathbf{x}}$ and $\mathbf{x}$ than a simple L2 loss, and has been popular in (learned) image compression. To measure rate, we simply use the log-likelihood $-\log p(\mathbf{z})$ where $\mathbf{z}$ is produced by the encoder deterministically. The losses are visualized in Figure 2.

To encode a chunk of video $\mathbf{x}$, we map it through the encoder to obtain latents $\mathbf{z}$. Then, we go through the latent variables one by one, and make a prediction for the next latent variable using the autoregressive prior $p(z_j|z_{<j})$. We then use an arithmetic coding algorithm to obtain a bitstream $b_j = \mathrm{ENC}(z_j, p(z_j|z_{<j}))$ for the j-th variable. The expected length of b_j is $-\log p(z_j|z_{<j}))$.

To decode, we take the bitstream b_j and combine it with the prediction $p(z_j|z_{<j})$ to obtain $z_j = \mathrm{DEC}(b_j, p(z_j|z_{<j})$. Once we have decoded all latents, we pass them through the decoder of the AE to obtain $\hat{\mathbf{x}}$.

5. Experiments

5.1. Dataset

Kinetics [9] We use videos with a width and height greater than $720px$, which results in $98,944$ videos as our training set. We only use the first 16 frames for training. The resulting dataset has about $1.6m$ frames, which is sufficient for training our model, though larger models and datasets will likely result in better rate/distortion (at the cost of increased computational cost during training and testing).

Ultra Video Group [1] UVG contains 7 videos with $3,900$ frames in full HD resolution (1920×1080). We use this dataset to compare with state-of-the-art.

Standard Definition Videos SDV contains 20 videos with $\sim 40K$ frames of resolution 352×288. We use this dataset for ablation studies.

Human Activity contains 1257 real-world videos of people in various everyday scenes, and is mostly used for human pose estimation and tracking in video. Following the standard partitions of the data, we use 1087 and 170 videos as train and test set for semantic compression experiments.

Dynamics is an internal dataset containing ego-view video from a car driving on different highways at different times of day. The full dataset consists of 5 clips taken at different dates, times, and locations. We use 4 clips of 20 minutes each (120k frames) as train set, and use the fifth clip of 14 minutes (25k frames) as test sequence.

Berkeley MHAD [29] contains videos of human actions, recorded by four multi-view cameras. We use this dataset

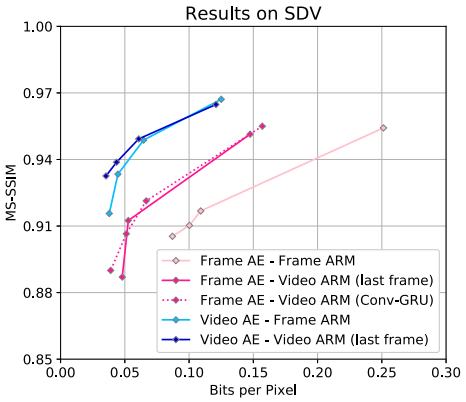


Figure 5: Ablation experiments. The both autoencoder and prior exploit temporal dependencies, in pixel and latent space respectively, to improve video compression.

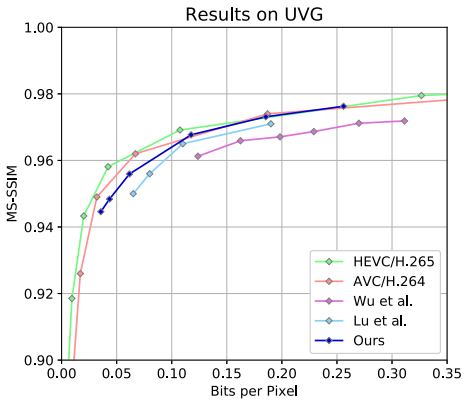


Figure 6: Comparison to the state-of-the-art traditional and learned codecs. Our proposal outperforms the learned counterparts and approaches AVC/H.264 and HEVC/H.265 evaluated in their default setting.

for multi-modal compression experiments. We use all four video streams from the first quad-camera, each of which records the same scene from a slightly shifted vantage point. The MHAD dataset contains 11 actions each performed by 12 participants, with 5 repetitions per participant. We use the first 4 repetitions for training, and the last one for testing.

Kinetics, Dynamics and Human Activity are only available in compressed form, and hence contain compression artifacts. In order to remove these artifacts, we downscale videos from these datasets so that the smallest side has length 256, before taking crops. For uncompressed datasets (UVG, SDV, and MHAD), we do not perform downscaling.

5.2. Training

We train all of our models with batchsize 32, using the Adam optimizer [24] with learning rate 10^{-4} (decaying with $\gamma = 0.1$ every 40 epochs) for a total of $N = 100$ epochs. Only for the Kinetics dataset, which is much larger, we use 10 epochs and learning rate decay every 4 epochs.

We use MS-SSIM (multi-scale structural similarity) as a distortion loss, and the cross-entropy as a rate loss. In order to obtain rate-distortion curves, we train separate models for beta values $\beta \in \{0.1, 0.3, 0.5, 0.7\}$ (unless stated otherwise), and report their rate/distortion score.

5.3. Ablation studies

We evaluate several AE and prior design choices as discussed in Section 4. Specifically, we compare the use of 2D and 3D convolutions in the autoencoder, Frame AE and Video AE respectively, as well as three kinds of priors: a 2D frame-based ARM that does not exploit temporal dependencies (Frame ARM), an ARM conditioned on the previous frame (Video ARM-last frame), and one conditioned on the output of a Conv-GRU (Video ARM-Conv-GRU). We train each model on Kinetics and evaluate on SDV.

The results are presented in Figure 5. The results show that conditioning the ARM on the previous frame yields a substantial boost over frame-based encoding, particularly when using a frame AE. Introducing a Conv-GRU only marginally improves results compared to conditioning on the last frame only.

We also note that using the 3D autoencoder is substantially better than using a 2D autoencoder, even when a video prior is not being used. This suggests that the 3D AE is able to produce latents that are temporally decorrelated to a large extent, so that they can be modelled fairly effectively by a frame AE. The difference between 2D and 3D AEs is substantially bigger than the difference between 2D and 3D priors, so in applications where a few frames of latency is not an issue, the 3D AE is to be preferred, and can reduce the burden on the prior.

For the rest of the experiments, we will use the best performing model: the Video AE + Video ARM (last frame).

5.4. Comparison to state of the art

We benchmark our method against the state-of-the-art traditional and learned compression methods on UVG standard test sequences. We compare against classical codecs AVC/H.264 and HEVC/H.265, as well as the recent learned compression methods presented by [27] and [40]. For the classical codecs, we use the default FFmpeg settings, without imposing any restriction, and only vary the CRF setting to obtain rate/distortion curves. For the other learned compression methods, we use the results as reported in the respective papers. For our method, we use 6 different β values, namely, $0.03, 0.05, 0.1, 0.3, 0.5, 0.7$.

Figure 6 shows that our method consistently outperforms other learned compression methods, and is approaching the performance of classical codecs, particularly in the $0.10 - 0.25$ bpp range. We note that in some previous works, learned compression was shown to outperform classical codecs, when the latter are evaluated under restricted settings by limiting the inter-frame compression to only few

frames, *i.e*. by setting *GOP* flag to 12. The results under restricted setting are reported in supplementary materials.

5.5. Semantic Compression

The perceived quality of a compressed video depends more on how well salient objects are reconstructed, and less on how well non-salient objects are reconstructed. For instance, in video conferencing, it is more important to preserve details of faces than background regions. It follows that better subjective quality can be achieved by allocating more bits to salient / foreground objects than to non-salient / background objects.

Developing such a task-tuned video codec requires a semantic understanding of videos. This is difficult to do with classical codecs as it would require distinguishing foreground and background objects. For learned compression methods, the asymmetry is easily incorporated by using different weights for the rate/distortion losses for foreground (FG) and background (BG) objects, assuming that groundtruth FG/BG annotations are available during training.

In this experiment, we study the semantic compression of the *person* category. The groundtruth person regions are extracted using a Mask R-CNN [16] trained on COCO images. We use bounding boxes around the objects, but the approach is applicable to segmentation masks without any modification required. The detected person regions are converted to a binary mask and used for training.

The MS-SSIM loss is a sum over scales of the SSIM loss. The SSIM loss computes an intermediate quantity called the similarity map, which is usually aggregated over the whole image. Instead, we aggregate these maps separately for foreground and background, where the FG and BG mask at a given scale is obtained from the high-resolution mask by average pooling. We then sum the FG and BG components over each scale, and multiply the resulting FG and BG losses by separate weights α and $1-\alpha$, respectively. We set the α to 0.95 in our experiments.

The rate loss is a sum of $-\log p(z_i|z_{<i})$, so we can multiply each term with a foreground/background weight. Each latent covers an 8×8 region of pixels, thus, we need to aggregate the pixel-wise labels to obtain a label for each latent. We do this by average pooling the FG/BG mask over 8×8 regions to obtain a weight per latent position which we multiply with the rate loss at that position.

The results are shown in Figure 7a. We observe that in the non-semantic model, BG is reconstructed more accurately than FG at a fixed average bitrate. The same behavior is observed for classical codecs as reported in supplementary materials. The worse reconstruction of FG is not surprising because person regions usually contain more details compared to the more homogeneous background regions. However, when using semantic loss weighting, the relation is reversed. Semantic loss weighting leads to an improvement in MS-SSIM score for FG at the expense of MS-SSIM score for BG. It demonstrates the effectiveness of learned video compression in incorporating semantic understanding of video content into compression. We believe that it opens up novel video compression applications which have not been feasible with classical codecs.

5.6. Adaptive Compression

Classical codecs are optimized for good performance across a wide range of videos. However, in some applications, the codec is used on a distribution of lower entropy videos, *i.e.* scenes with predictable types of activities. For example, a security camera placed at a fixed location and viewpoint will produce a very predictable video. In this experiment we show that learned compression models can utilize the lower entropy videos by simply being finetuned on them, which is difficult to do with classical codecs.

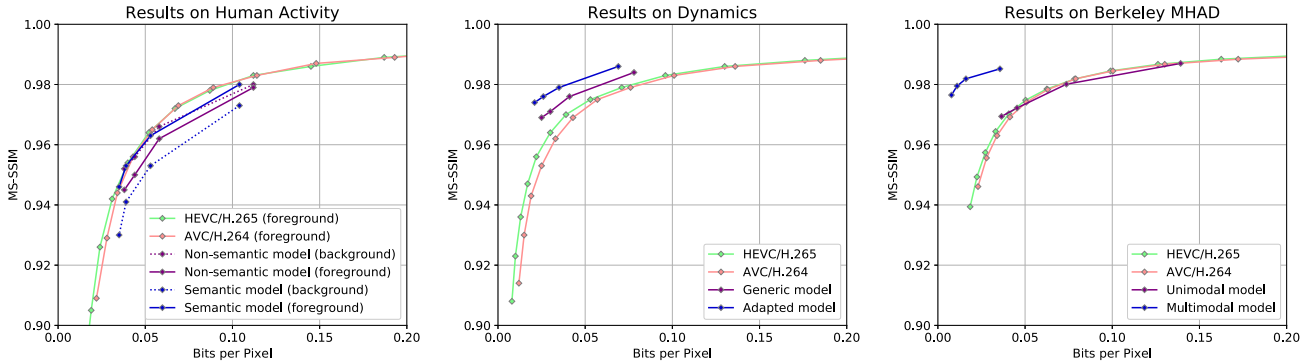
In this experiment, we show that by finetuning a learned compression model on the Dynamics dataset, substantial improvements in compression can be achieved. Figure 7b compares the classical codecs with our *generic model* as well as the *adapted model*. The generic model is trained on a generic training set from Kinetics. The adapted model takes a pretrained generic model and finetunes it on videos of a similar domain. The results show that our generic method outperforms the classical codecs on this dataset, and the adapted method shows even better performance.

This experiment indicates a great practical potential of learned compression models. Finetuning a compression model allows to maintain high reconstruction quality with substantially lower compression rate, while the model could be transferred from a generic compression model.

5.7. Multimodal Compression

Classical codecs are designed for typical videos captured by monocluar color cameras. When other modalities are included, such as depth, stereo, audio, or spectral imaging sensors, classical codecs are often not applicable or not able to exploit dependencies which exist between various modalities. Developing a codec for every new modality is possible, but very expensive considering the amount of engineering work involved in designing classical codecs. Using our learned compression method, however, adding new modalities is as easy as retraining the model on a new dataset with minimal modifications required.

In this experiment, we adapt our learned compression method to compress videos of human actions recorded by quad (four view) cameras from MHAD dataset. We compare four methods: *AVC/H.264* and *HEVC/H.265*, as well as a learned *unimodal model* and a learned *multimodal model*. The unimodal model is trained on the individual video streams, and the multimodal model is trained on the channel-wise concatenation of the four streams. The net-



(a) Semantic Compression (b) Adaptive Compression (c) Multimodal Compression

Figure 7: Three extensions of our model that demonstrate the benefits of learned over classical approaches to compression.

Figure 8: Multimodal compression results for HEVC/H.265 (top) and our proposal (bottom). By utilizing the redundancies between different views of a quad camera (columns), our model achieves a significantly better reconstruction while using $5\times$ less bits (0.007 vs 0.035 BPP).

work architecture for the unimodal model and the multimodal model is the same as the one described in Section 4, the only difference being that the multimodal model has more input channels (4×3 vs 3).

Interestingly, our approach retains more details than the classical codec (e.g., see the face of a person in Figure 8) while obtaining 5 times smaller BPP. The quantitative results, shown in Figure 7c, show that the multimodal compression model substantially outperforms all three baselines by utilizing the great amount of redundancy which exist between multiple data modalities. This shows that without further tuning of the architecture or training procedure, our method can be applied to compress spatio-temporal signals from non-standard imaging sensors.

6. Conclusion

We have presented a video compression method based on variational autoencoders with a deterministic encoder. Our theoretical analysis shows that in lossy compression, where bits-back coding cannot be used, deterministic encoders are preferred. Concretely, our model consists of an autoencoder and an autoregressive prior. We found that 3D spatio-temporal autoencoders are very effective, and greatly reduce the need for temporal conditioning in the prior. Our best model outperforms recent learned video compression methods without incorporating video-specific techniques like flow estimation or interpolation, and performs on par with the latest non-learned codec H.265 / HEVC.

In addition, we have explicitly demonstrated the potential advantages of learned over non-learned compression, beyond mere compression performance. In semantic compression, the rate and distortion losses are weighted by the semantics of the video content, giving priority to important regions, resulting in better visual quality at lower bitrates in those regions. In adaptive compression, a pretrained video compressor is finetuned on a specific dataset. With minimal engineering effort, this yields a highly effective method for compressing domain specific videos. Finally, in our multimodal compression experiments, we have demonstrated a dramatic improvement in compression performance, obtained simply by training the same model on a multi-modal dataset consisting of quad-cam footage.

References

[1] Ultra video group test sequences. http://ultravideo.cs.tut.fi/. Accessed: 2019-03-18.

[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations. In *NIPS*, pages 1141–1151. Curran Associates, Inc., 2017.

[3] Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a Broken ELBO. *arXiv:1711.00464*, Nov. 2017.

[4] Mohammad Haris Baig, Vladlen Koltun, and Lorenzo Torresani. Learning to Inpaint for Image Compression. In *NIPS*, pages 1246–1255, 2017.

[5] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end Optimized Image Compression. 2016.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational Image Compression with a Scale Hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[7] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. 2013.

[8] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st ed. 20 edition, Oct. 2006.

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017.

[10] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational Lossy Autoencoder. *arXiv:1611.02731*, 2016.

[11] Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu. Learning for Video Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, Apr. 2019.

[12] Cisco. The Zettabyte Era: Trends and Analysis. Technical report, 2017.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 2006.

[14] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep AutoRegressive Networks. *arXiv:1310.8499*, Oct. 2013.

[15] Jun Han, Salvator Lombardo, Christopher Schroers, and Stephan Mandt. Deep Probabilistic Video Compression. *arXiv:1810.02845*, 2018.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv:1703.06870*, Mar. 2017.

[17] K He, X Zhang, S Ren, and J Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[18] Geoffrey E Hinton and Drew van Camp. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. In *ACM Conf. on Computational Learning Theory*, 1993.

[19] Geoffrey E Hinton and Richard S Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *NIPS*, pages 3–10, 1994.

[20] A. Honkela and H. Valpola. Variational Learning and Bits-Back Coding: An Information-Theoretic View to Bayesian Learning. *IEEE Transactions on Neural Networks*, 15(4):800–810, July 2004.

[21] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, Feb. 2015.

[22] 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 *CVPR*, 2017.

[23] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video Pixel Networks. In *ICML*, pages 1771–1779, 2017.

[24] D Kingma and J Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[25] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114*, Dec. 2013.

[26] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *CVPR*, pages 3214–3223, 2018.

[27] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An End-to-end Deep Video Compression Framework. *arXiv:1812.00101*, Nov. 2018.

[28] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional Probability Models for Deep Image Compression. In *CVPR*, pages 4394–4402, Jan. 2018.

[29] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, Rene Vidal, and Ruzena Bajcsy. Berkeley MHAD: A comprehensive Multimodal Human Action Database. In *IEEE Workshop on Applications of Computer Vision*, pages 53–60, Clearwater Beach, FL, USA, Jan. 2013.

[30] Jorge Pessoa, Helena Aidos, Pedro Tomás, and Mário AT Figueiredo. End-to-End Learning of Video Compression using Spatio-Temporal Autoencoders. 2018.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 2014.

[32] Oren Rippel and Lubomir Bourdev. Real-Time Adaptive Image Compression. In *ICML*, pages 2922–2930, 2017.

[33] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. Learned Video Compression. *arXiv:1811.06981 [cs, eess, stat]*, Nov. 2018.

[34] Shibani Santurkar, David Budden, and Nir Shavit. Generative compression. In *Picture Coding Symposium*, pages 258–262, 2018.

[35] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders. *arXiv preprint arXiv:1703.00395*, Mar. 2017.

[36] 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 *CVPR*, pages 5306–5314, 2017.

[37] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional Image Generation with PixelCNN Decoders. In *NIPS*, pages 4790–4798. Curran Associates, Inc., 2016.

[38] Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2007.

[39] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004.

[40] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video Compression through Image Interpolation. In *ECCV*, pages 416–431, 2018.