Appendix A. Architecture specifications

Here we further describe all architectural details we could not fit in the paper, with the goal of rendering our model fully reproducible by the reader.

A.1. State Propagator

This component is composed of 3 different submodels: \(E(\cdot), D(\cdot)\) and \(G(\cdot)\).

For both \(E(\cdot)\) and \(D(\cdot)\) we use a multiscale version \(^2\) of the Dual Path module, by combining the multiscale rendition of DenseNet \(^3\) with the Dual Path idea \(^1\) (see Figure 1). Specifically, we maintain 3 scales, at spatial map ratios \(4, 8, 16\) relative to the input frame dimensions. We set the path size to \(128\), the growth parameter to \(32\), and the number of module repetitions to \(5\). In each module, as in the Dual Path approach, a residual is computed and added to the path, and the state is grown by the growth factor as for DenseNet. These operations are done with \(3 \times 3\) convolutions followed by ReLU nonlinearities (see Figure 2). The layer widths are completely determined by the Dual Path residue and growth parameters. This process is repeated individually for each scale, with the input being a concatenation of the outputs of all scales in the previous module iteration, resampled to the current scale.

Hence, the state \(S_t\) is composed of \(3\) tensors — one for each scale mentioned above. The function \(G(\cdot)\) simply upsamples the lower scales to a common map size ratio of \(4\), concatenates the outputs from all scales, and again upsamples by a factor of \(4\) to attain the final output in pixel space.

The entire model (including the encoder and the decoder) has a total of 29 million trainable parameters.

A.2. Spatial rate controller

This component is composed of the individual branch encoders \(E_1(\cdot), \ldots, E_R(\cdot)\), and branch decoders \(D_1(\cdot), \ldots, D_R(\cdot)\).

The encoders take in the output of the encoder, which is in the form of \(3\) tensors at different scales (described in the previous subsection). The different encoders may have different output map sizes. Each encoder \(E_r(\cdot)\) is constructed in the following way: it first maps all tensors to the map size assigned by the code tensor \(c_r\) associated with branch \(r\), by performing respective upsampling or downsampling operations. The outputs are then concatenated, and mapped through a final \(3 \times 3\) convolution.

Each decoder \(D_r(\cdot)\) performs the inverse operations of encoder \(E_r(\cdot)\). Namely, it first maps the code tensor through a \(3 \times 3\) convolution, and then splits it into \(3\) different tensors, which are resampled to the appropriate scales.

A.3. Coding procedure

For the conditioning context \(C\) within the adaptive entropy coding procedure, we use the bit to the left, bit to the top, the bit to the top left, as well as the bit at the same location at the previous bitplane transmitted.
Appendix B. Detailed description of test sets

B.1. CDVL SD

The Consumer Digital Video Library can be found at http://www.cdvl.org/. To retrieve the SD videos, we searched for VGA resolution at original and excellent quality levels. There were a few instances of near-duplicate videos: in those cases we only retrieved the first. All videos are listed below.

- Bennet-Watt_BeeClose_VGA60fps
- Bennet-Watt_BeeZoom_VGA60fps
- Bennet-Watt_CattleDogs_VGA60fps
- Bennet-Watt_DecantWine_VGA60fps
- Bennet-Watt_FlockSunset_VGA60fps
- ntia_bpit1-vga_original
- ntia_bpit2-vga_original
- ntia_bpit3-vga_original
- ntia_bpit4-vga_original
- ntia_bpit5-vga_original
- ntia_cardark-vga_original
- ntia_cargas-vga_original
- ntia_catjoke-vga_original
- ntia_chhart1-vga_original
- ntia_diner-vga_original
- ntia_drmfeet-vga_original
- ntia_drmside-vga_original
- ntia_fish1-vga_original
- ntia_fish5-vga_original
- NTIA_FlamencoDancers_VGA60fps
- NTIA_FlamencoShoes_VGA60fps
- ntia_flower1-vga_original
- ntia_flower2-vga_original
- ntia_flower3-vga_original
- ntia_flower4-vga_original
- ntia_flower5-vga_original
- ntia_overview1-vga_original
- ntia_rfdev1-vga_original
- ntia_schart1-vga_original
- ntia_spectrum1-vga_original
- ntia_storel-vga_original
- ntia_street1-vga_original
- NTIA_TheFootDrummer_VGA60fps
- NTIA_TheFootPan_VGA60fps
- NTIA_TheFootPiano_VGA60fps
- NTIA_WaveRocks_VGA60fps
- ntia_wboard1-vga_original

B.2. Xiph HD

The Xiph test videos can be found at https://media.xiph.org/video/derf/ We used all videos with 1080p resolution.

- aspen_1080p.y4m
- blue_sky_1080p25.y4m
- controlled_burn_1080p.y4m
- crowd_run_1080p50.y4m
- dinner_1080p30.y4m
- ducks_take_off_1080p50.y4m
- in_to_tree_1080p50.y4m
- life_1080p30.y4m
- old_town_cross_1080p50.y4m
- park_joy_1080p50.y4m
- pedestrian_area_1080p25.y4m
- red_kayak_1080p.y4m
- riverbed_1080p25.y4m
- rush_field_cuts_1080p.y4m
- rush_hour_1080p25.y4m
- snow_mnt_1080p.y4m
- speed_bag_1080p.y4m
- station2_1080p25.y4m
- sunflower_1080p25.y4m
- touchdown_pass_1080p.y4m
- tractor_1080p25.y4m
- west_wind_easy_1080p.y4m

Appendix C. Directions for future improvement

Overall, we see several ways in which this work can be improved and extended:

Generalization to bi-directional prediction. The model presented in this work only addresses the low-latency mode: it only implements the notion of P-frames. It lacks the ability to encode frame into the future, and use these for bi-directional prediction using B-frames – an ability which has gotten modern video codecs a great boost in compression performance.

Architectural improvements. There are many architectural choices we have made that we believe could be improved further. Some of these include better modeling for the encoder/decoder backbones; rethinking of how to best represent the state, as well as propagate it from frame to frame; and exploring the structure of the state-to-frame module beyond simple generation of flow and residual.

Performance optimization. The model presented in this work has not been optimized for speed, and thus is still prohibitively slow for real-life deployment in computationally-constrained environments. We are confident that it can be sped up dramatically via architectural changes, lower data type precision, and so on.

Better performance on trivial videos. We observe our model to significantly outperform the standards for videos that are spatiotemporally complex; however, interestingly, it underperforms for very simple and static videos. Resolving this will lead to another boost in performance.

References