# Neural Video Compression with Spatio-Temporal Cross-Covariance Transformer (Supplementary Materials)

**ACM Reference Format:**
. 2023. Neural Video Compression with Spatio-Temporal Cross-Covariance Transformer (Supplementary Materials). In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3581783.3611960

This document is supplementary material for the paper "Neural Video Compression with Spatio-Temporal Cross-Covariance Transformers." We provide additional architectural details (Section A), more experimental settings (Section B.1), additonal quantitative results (Section B.2), supportive ablation studies (Section B.3) and individual quantitative results (Section B.4).

## A ARCHITECTURAL DETAILS

### A.1 Details of ST-XCT

This section details the architectural decisions of our Spatio-Temporal Cross-Covariance Transformer (ST-XCT), which was presented in Section 3.1 of the main manuscript. As discussed there (and detailed in Fig.1 from the paper), ST-XCT is composed of two main components: *Spatio-temporal Feature Generator (STFG)* and *3D Feed-Forward Gate (3FFG)*. In what follows, we denote an input joint feature as $\mathbf{F}_j \in \mathcal{R}^{H\times W\times 2\times C}$.

As in conventional transformer strategies, the *Spatio-Temporal Feature Generator (STFG)* component is designed to generate Queries (Q), Keys (K), and Values (V). For that, we first apply two 3D convolutional layers with $1\times1\times1$ kernels and then $3\times3\times3$ kernels. For both layers, we use $C$ as the number of output channels. Once we concatenate all attention features from each sub-head to $\mathbf{F}_p$, we apply a 3D convolutional operation with $1\times1\times1$ kernel and the same number of output channels as $C$.

In our *3D Feed-Forward Gate (3FFG)* component, we use two branches to separately generate distinct features. In each branch, we first apply $1\times1\times1$ 3D convolutional layer with the number of output channels as $FFG\_factor \cdot C$. We then apply a $3\times3\times3$ 3D convolutional layer with the number of output channels as $C$. Eventually, we adopt a $1\times1\times1$ 3D convolutional layer with output channel number as $C$, after applying element-wise multiplication for these two distinct features for better feature transformation.


*MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.*

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3611960

### A.2 Details of our NVC with ST-XCT

*ST-XCT instantiation.* The overall architecture of our ST-XCT is the same in the different components in our NVC framework (please refer to Section 3.2 of the main paper for more details), but instantiated with different hyper-parameters. The hyper-parameters we configure are the number of ST-XCT blocks, the number of heads, and $FFG\_factor$ in different components. The configuration details for each ST-XCT module in Fig. 3 of the main manuscript are provided in Table S1. As discussed in what follows, we have different variants of XC-MS-TFE and MS-TFD to operate with large and small scale features.

| | #Blocks | #Heads | FFG_factor |
|---|---|---|---|
| MS-TFE-Large-Scale | 4 | 2 | 1 |
| MS-TFE-Small-Scale | 6 | 2 | 1 |
| MS-TFD-Large-Scale | 4 | 2 | 1 |
| MS-TFD-Small-Scale | 6 | 2 | 1 |
| THEM | 16 | 6 | 3 |

**Table S1: Instantiation details of ST-XCT modules in the different components of our NVC framework. More details can be found in Sections 3.2.2, 3.2.3, and 3.2.4.**

*Residual Blocks.* As shown in Fig. 3, we used several residual blocks in both our MS-TFE and MS-TFD components. Specifically, we apply the residual blocks shown in Fig. S1 in our MS-TFE and MS-TFD modules.

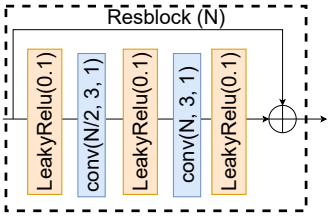


**Figure S1: Architecture of Residual Blocks. LeakyReLU(s) represents LeakyReLU activation function with slope value s, while Conv(C, K, S) means the 2D convolutional operation with output channel as $C$, $K\times K$ kernel and stride as $S$,**

*A.2.1 MS-TFE.* The hyper-parameters of those 2D convolutional operations and the residual blocks of MS-TFE are:

$Concat \rightarrow Conv(64,3,1) \rightarrow ResBlock(128) \rightarrow$
*ST-XCT-Large-Scale* $\rightarrow Conv(64,3,1) \rightarrow ResBlock(128) \rightarrow$

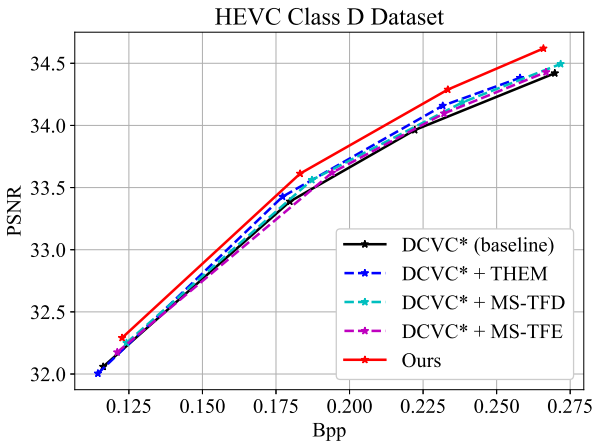


**Figure S2: Ablation study adding the different transformer-based components from our NVC framework.**

*ST-XCT-Small-Scale* $\rightarrow$ *Conv*$(64, 3, 2)$ $\rightarrow$ *Conv*$(96, 3, 2)$,

where *Conv*$(C, K, S)$ denotes a 2D convolutional operation with $C$ output channels, a $K \times K$ kernel and stride $S$. *ResBlock*$(N)$ represents the residual block shown in Fig S1 with $N$ input channel. More details can be found in Section 3.2.2 of the main manuscript.

*A.2.2 THEM..* In our work, we directly extend the hybrid entropy model as in [4]. Hence, we use an identical network architecture as in [4] for the Temporal Prior Encoder, Hyperprior Encoder, Hyperprior Decoder, and Hybrid Entropy Model in our Fig. 3 (b) of the main manuscript. More details can be found in our Section 3.2.3 and [4].

*A.2.3 MS-TFD.* The hyper-parameters of the 2D-convolutional operation (*i.e.*, subconv in Fig. 3(a)) andthe residual blocks of MS-TFD are listed as follows:

*Conv*$(100, 3, 1)$ $\rightarrow$ *Shufflex2* $\rightarrow$
*Conv*$(256, 3, 1)$ $\rightarrow$ *Shufflex2* $\rightarrow$ *ST-XCT-Small-Scale* $\rightarrow$
*ResBlock*$(128)$ $\rightarrow$ *Conv*$(256, 3, 1)$ $\rightarrow$ *Shufflex2* $\rightarrow$
*ST-XCT-Large-Scale* $\rightarrow$ *ResBlock*$(128)$ $\rightarrow$ *Conv*$(128, 3, 1)$ $\rightarrow$
*Shufflex2* $\rightarrow$ *Conv*$(96, 3, 2)$ $\rightarrow$ *Concat* $\rightarrow$ *U-Net,*

where we simply adopt the standard shuffle operation [2] to shuffle the features from the shape of $H \times W \times C$ to $2H \times 2W \times C/4$. We use an identical U-Net architecture as in [4]. More details can be found in Section 3.2.4.

# B EXPERIMENTS

## B.1 Settings of Traditional Codecs

We compare the performance with the reference software of H.265 and H.266, which are HM-16.21 [1] and VTM-13.2 [3], respectively. The low delay configuration with the highest compression ratio is used for both. All videos are compressed and reconstructed in YUV 420 format and RGB frames are extracted to compute distortion metrics. The detailed settings of HM and VTM are:

- **HM**

```
TAppEncoder -c encoder_lowdelay_main_rext.cfg [args]
```

- **VTM**

```
EncoderApp -c encoder_lowdelay_vtm.cfg [args]
```

where both codecs use the following common command line arguments ([args]):

```
--InputFile={input_filename}
--BitstreamFile={bitstream_filename}
--ReconFile={reconstructed_filename}
--DecodingRefreshType=2
--InputBitDepth=8
--OutputBitDepth=8
--OutputBitDepthC=8
--InputChromaFormat=420
--FrameRate={frame_rate}
--FramesToBeEncoded=96
--SourceWidth={width}
--SourceHeight={height}
--IntraPeriod=32
--QP={quantization_parameter}
--Level=6.2
```

## B.2 Additional Ablation Studies

*Impact of the individual transformer components.* To verify the effectiveness of our ST-XCT module, we conducted three ablation studies and demonstrated the results in Fig. 5 of the main manuscript, where we removed the ST-XCT modules at different components in our pipeline. Fig. S2 shows further ablation study results in which we add the ST-XCT modules at various stages into the baseline method DCVC* [4] to replace its original concatenation operation.

The results indicate a noticeable enhancement in performance when MS-TFE, THEM and MS-TFD were added, respectively reducing 0.44% , 1.59% and 1.78% bitrates from our baseline DCVC*. This further indicates the superior ability of ST-XCT for extracting and fusing multi-scale spatio-temporal features by using the cross-covariance attention mechanism to exploit the spatio-temporal correlations. Our findings also indicate that we could achieve maximum effectiveness when all these three Transformer-based components are employed. Such observations again support our claim that it is essential to consider all components when designing an end-to-end Transformer-based video compression framework. It is also the main difference between our proposed framework and previous Transformer-based methods (*e.g.*, VCT [5]), which only focuses on one particular component (*e.g.*, entropy model).

## B.3 Per-Sequence Quantitative Comparison

Lastly, we provide a few selected rate-distortion curves for individual sequences from each dataset (see Fig.S3 and S4). Specifically, we provide two cases for each dataset: the best (left column) and the worst cases (right column). The best and worst cases are defined with regard to how our proposed NVC framework surpasses our baseline method *i.e.*, DCVC* [4]). It is important to highlight that on average, for all datasets, our NVC framework is able to outperform the baseline and VTM methods, which is supported by the quantitative comparison (please refer to the details in Fig. 4 and Table. 1 of the main manuscript).

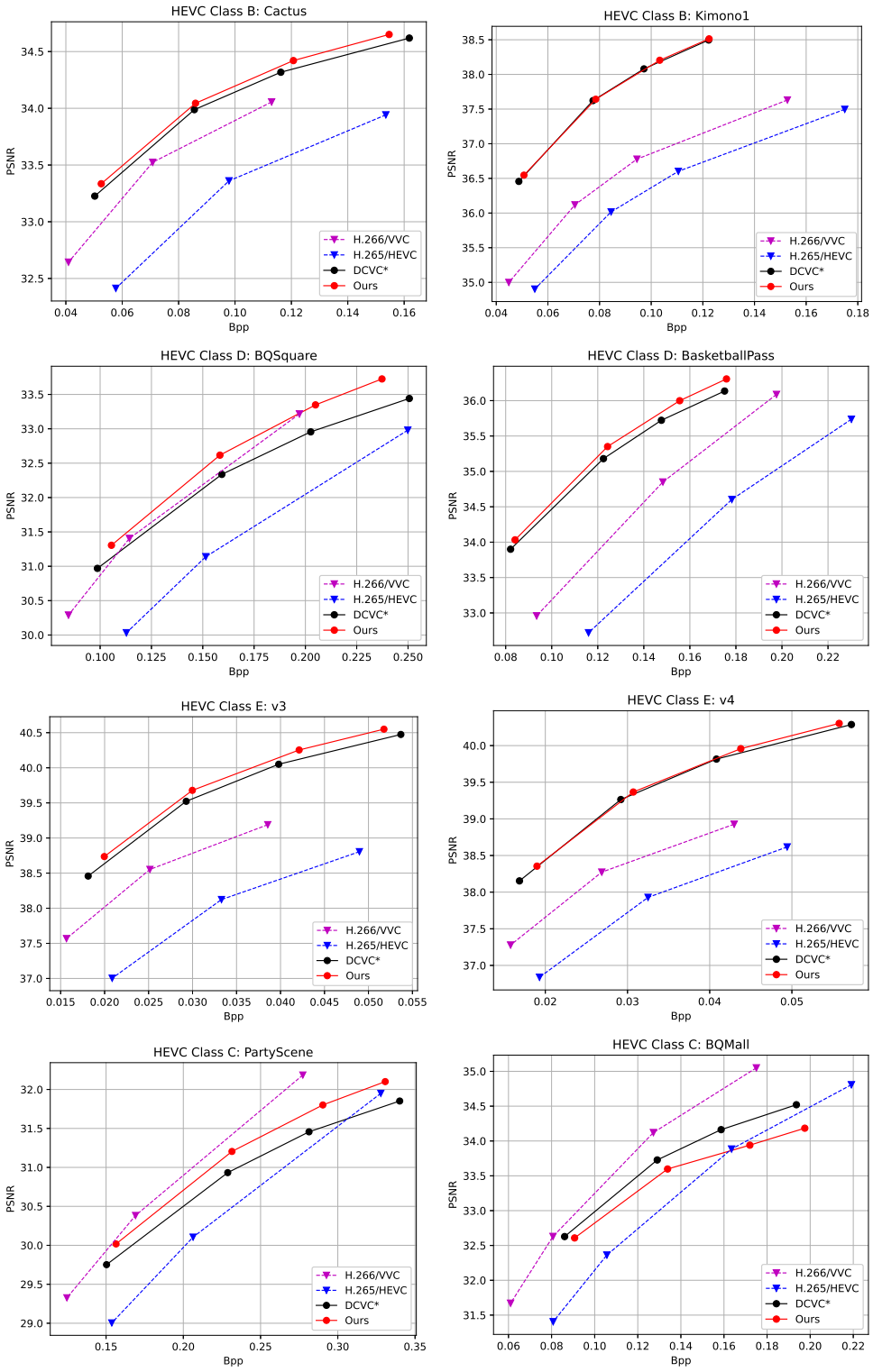


**Figure S3: Selected per-sequence rate-distortion (*i.e.*, bitrate vs PSNR) performance comparison in HEVC datasets, showing successful (left) and worst case (right) on the dataset.**

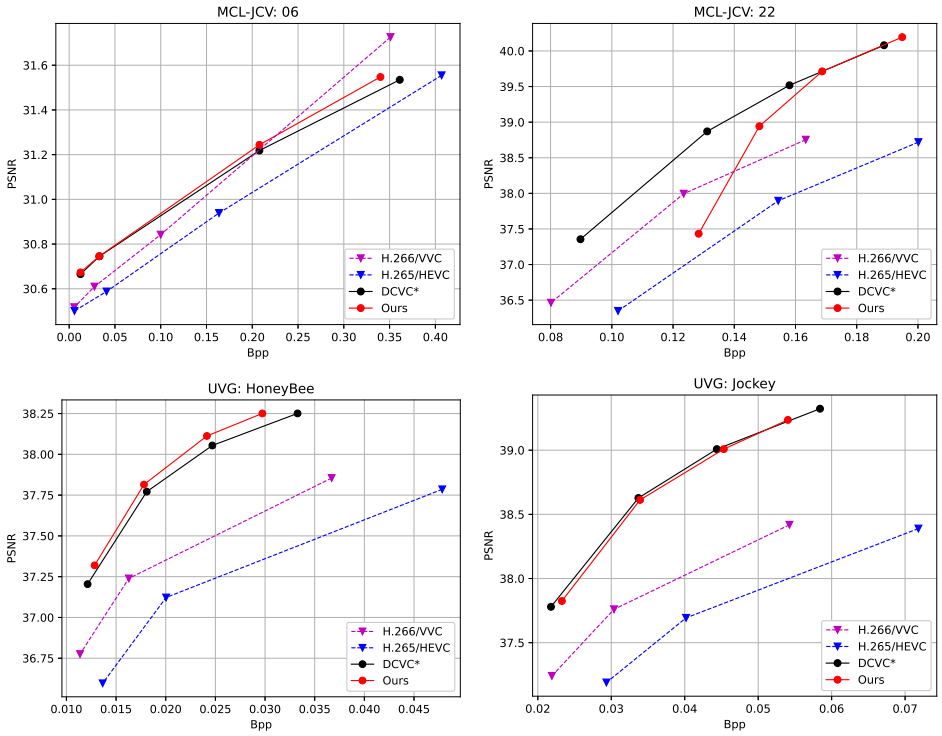


**Figure S4: Selected per-sequence rate-distortion (*i.e.*, bitrate vs PSNR) performance comparison of MCL-JCV and UVG datasets, showing successful (left) and worst case (right) on the datasets.**

From Fig. S3 and S4, it can be observed that for HEVC Class B, HEVC Class D, and HEVC Class E and UVG, even in the worst cases, our proposed NVC can still improve over (or at least not perform worst than) the baseline. As expected, in the worst case such improvements are less prominent than the average. Moreover, although we improve over the baseline and VTM in most cases, it is also interesting to see that neural methods can still be outperformed by H.266/VVC in specific cases (*e.g.*, some sequences on HEVC Class C). We plan to study in more detail such specific cases to better understand why neural methods fail on them and how we can improve our method at the same time that we can provide new insights for the neural video compression community.

## B.4 More Visualization

Here we provided more visualization results as shown in Fig. S5.

## REFERENCES

[1] [n. d.]. Hevc test model (hm). https://hevc.hhi.fraunhofer.de/HM-doc/. Accessed: 2023-03-06.
[2] [n. d.]. Torch Pixel Shuffle. https://https://pytorch.org/docs/stable/generated/torch.nn.PixelShuffle.html/. Accessed: 2023-03-06.
[3] [n. d.]. VVC Reference Model (VTM). https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/. Accessed: 2023-03-06.
[4] Jiahao Li, Bin Li, and Yan Lu. 2022. Hybrid spatial-temporal entropy modelling for neural video compression. In *Proceedings of the 30th ACM International Conference on Multimedia*. 1503–1511.
[5] Fabian Mentzer, George D Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. 2022. VCT: A Video Compression Transformer. *Advances in Neural Information Processing Systems* 35 (2022), 13091–13103.

Received 11 May 2023; revised 7 July 2023; accepted 25 July 2023


**Figure S5: Qualitative comparison between our NVC method, DCVC*, and VTM-13.2. The demonstrated images are labeled as PSNR@bpp. Best viewed on screen.**