# A Network Architecture

**Base Network:** We use an MLP with 10 layers, width of 512 and ReLU non-linearity as our base network $f_\theta$.
**Hypernetwork:** All hypernetworks $h^l$ used to modulate a layer $l$ of the base network have 3 layers with a hidden dimension of 512 and tanh as non-linearity. Unless specified, we only modulate the first hidden layer of the base network.
**Latents:** Each latent $Z_t$ corresponding to a frame has a dimension of 512 and is initialized to be standard Gaussian before training. We set our learning rate as 5e-4 and used the standard Adam optimizer without any weight decay.

# B Compression

## B.1 Fourier Features

We use the multiresolution hash grid for positional encoding in all our models. In table 5 we show results for full coordinate resolution using fourier features for positional encoding. Due to lack of a hash grid, the resulting models train upto 30% faster, but at the cost of inferior reconstruction.

## B.2 Quantization

Instead of quantizing all components equally, we notice that retaining the latents and the base network at full precision provides better reconstruction at negligible additional storage.

## B.3 Effect of latent dimension

To study the effect of latent dimension on compression, we train models by varying it and encode the "bosphore" video from UVG dataset. The results are presented in Figure 11. We notice that there is positive gains till dimension 512 and diminishing returns thereafter. Hence we choose that as our default latent size in all our experiments.

# C Video Retrieval

We perform two retrieval tasks on the COIN dataset[53] - *class-level*, and *segment-level*. In both settings, we use the standard val set as the database. For *class-level*, we use the distinct video-level task names in COIN as our query set. For *segment-level*, we use the set of distinct clip-level captions in COIN as our query set. We get the CLIP ViT-B/32 text embeddings of each of these captions, and these become our query vectors. For database vectors, we use the per-frame learned latents for each video in the database. For comparison with CLIP, we replace these database vectors with the CLIP ViT-B/32 image embeddings for each frame. For *class-level* retrieval, we consider a result frame a positive match if it belongs to

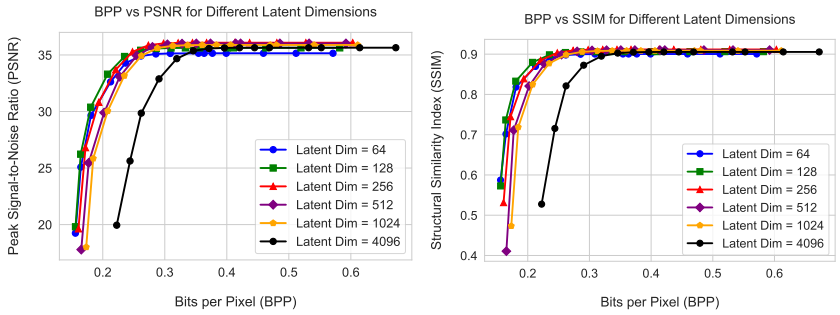


Fig. 11: Effect of varying latent dimension across different bitrates.


Fig. 12: Nearest Neighbours for segment-level matching of sample queries from COIN validation set. The green boxes denote the true positives and the red ones are false pos- itives. We show the inner product similarity between the image and the corresponding query inside the green boxes at the bottom of each image

a video with the same class label as the queried caption. On the other hand, for *segment-level* retrieval, we consider a result frame a positive match only if it belongs to a segment with the same caption as the query. Further, this search is done over all videos. We use FAISS[20] as our retrieval implementation and use *cosine similarity* as the distance metric.

We perform whole-level video retrieval as described in the main paper. For text, we use CLIP to compute a feature for the paragraph caption. For the video, we compute a per-frame feature for CLIP, or use the learnt latents from Latent-INR. For a single video feature, we then average these per-frame features. We normalize all features, and perform retrieval by finding the closest embeddings using dot product similarity. Both text-to-video and video-to-text are performed in the same manner, the only difference being which features are used as query and key.

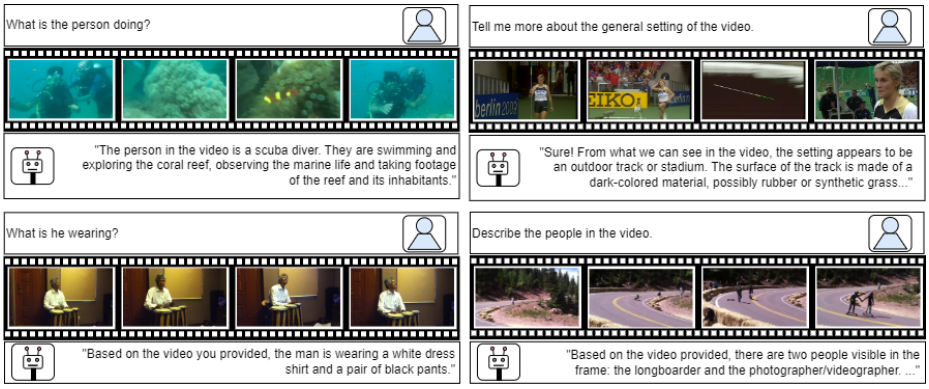


Fig. 13: Additional results for Latent-INR interface with Video-LLM.

Fig.12, shows the retrieval results on the COIN data in the *segment-level* setting. It can be seen that a majority of failure cases could be attributed to visual similarity across different tasks when seen at an individual frame level.

## D Video Chat

We interface our latents with learned features from Video-Llama [58] to enable interactive chat with the compressed videos. In [58], the $N$ video frames are passed through a ViT based visual encoder to extract features of size $k \times d$ per frame. These are then passed through a Query Former [24] to obtain a unified video representation of size $k_v \times d_v$. This tensor is then passed to a trainable MLP layer before aligning with an LLM of our choice (LLama-2 [54] in our models).
We align our latents $Z$ with these per-video features of size $k_v \times d_v$ using a linear projection layer which is trained end to end. The loss function is slightly modified to incorporate a cosine similarity loss between the terms.

$$L = L_{\text{MSE}} + \lambda \cdot L_{cos}(F_t, F_t^{\text{V-LLM}}) \tag{7}$$

where $F_t$ is the predicted feature and $F_t^{\text{V-LLM}}$ is the corresponding Video-LLama extracted features. We show additional results of the interactive chat in Figure 13.

## E Video-wise results

We plot the results for each video from UVG dataset [31] in Figures 14, 15, 16, 17, 18, 19, and 20. We show three versions of our model based on the dimension of the *low-rank* modulating matrix. The *Ours-s, Ours-m*, and *Ours-l* correspond respectively to $size = 50, 100, 200$ The *Ours-m* model achieves reasonable performance when compared to other methods, and at the same time can do the

Table 5: Fourier Feature Models

| Method | PSNR | BPP |
|---|---|---|
| Ours- Fourier - $S$ | 31.99 | 0.31 |
| Ours- Fourier - $M$ | 33.69 | 0.62 |
| Ours- Fourier - $L$ | 33.19 | 0.84 |

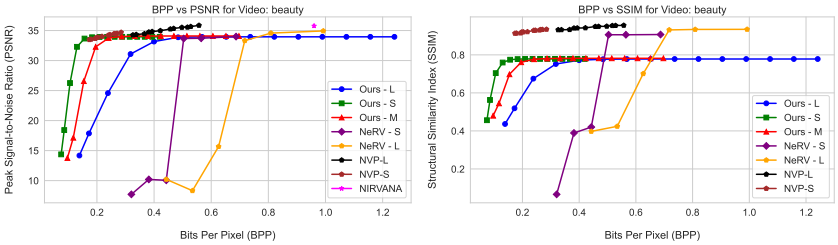


Fig. 14: BPP vs. PSNR, SSIM for beauty.

downstream tasks of interpolation and retrieval which none of the compared methods can.

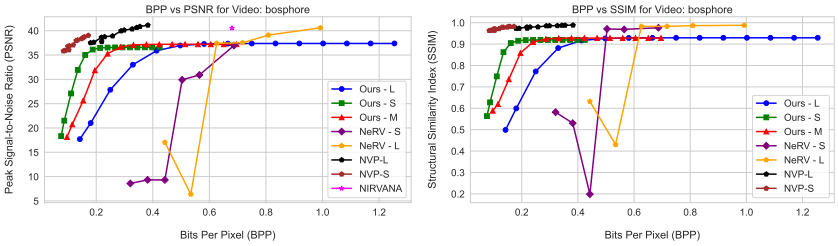


Fig. 15: BPP vs. PSNR, SSIM for bosphore.

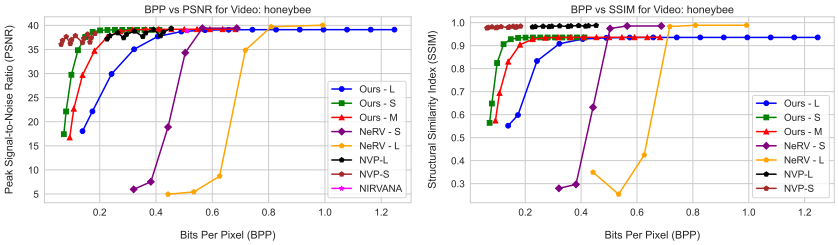


Fig. 16: BPP vs. PSNR, SSIM for honeybee.

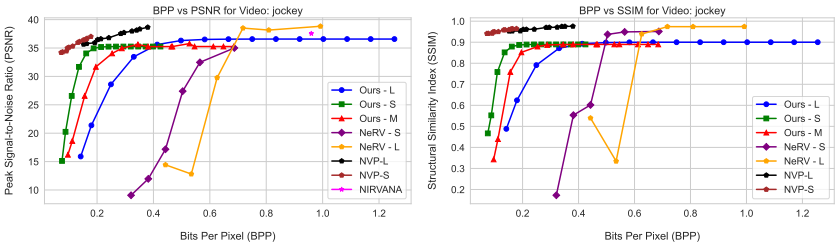


Fig. 17: BPP vs. PSNR, SSIM for jockey.

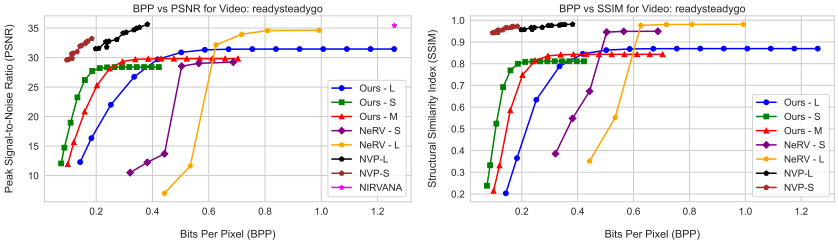


Fig. 18: BPP vs. PSNR, SSIM for readysteadygo.

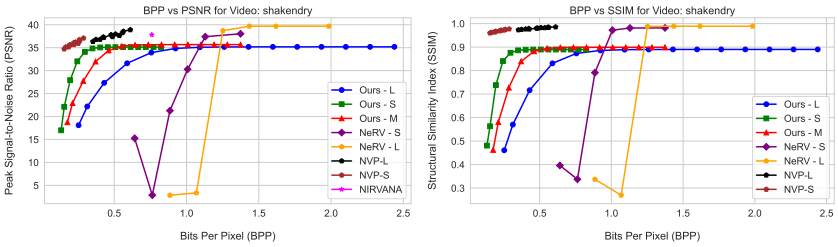


Fig. 19: BPP vs. PSNR, SSIM for shakendry.

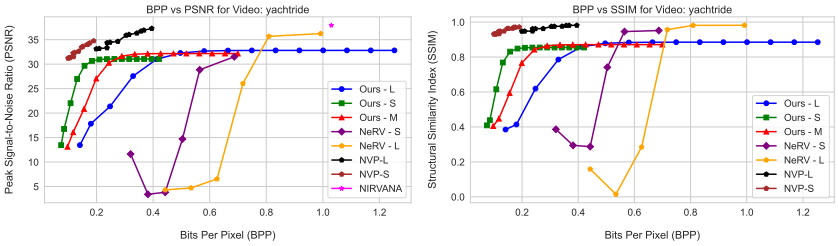


Fig. 20: BPP vs. PSNR, SSIM for yachtride.