6 Additional Experimental Details

6.1 Dataset Summary

Our experiments use the DAVIS and Set8 datasets datasets (Pont-Tuset et al., 2017; Tassano et al., 2020). The DAVIS train-val dataset consists of 90 sequences and the number of frames in a sequence varies from 40 - 104 with resolution 480×854. The DAVIS testing dataset consists of 30 sequences. Set8 consists of four sequences from a GoPro camera and four sequences from the DAVIS test set. For Set8, the number of frames in a sequence varies from 35 - 85 with resolution 540×960.

6.2 Upgrading Space-Only Attention

Is Losing Spatial Resolution Worth Space-Time Features? In practice, space-time attention might replace space-only attention. Is the loss of spatial information worth the temporal information? Theoretically assessing the impact of losing spatial information on model quality is difficult, so we instead seek experimental evidence. Using the COLA-Net architecture for video denoising, we find that space-time attention is significantly better than global or non-local space-only attention.

Experimental Details. We finetune the COLA-Net denoising network provided by the original authors for 30 epochs of 400 randomly selected sample sequences from the DAVIS training dataset (Pont-Tuset et al., 2017). Each sequence consists of 5 frames with resolution 128×128. We train on two Titan RTX GPUs with a batch size of 8 using PyTorch Lightning (Falcon et al., 2019). The optimizer is Adam with a learning rate of $1 \cdot 10^{-4}$ and the learning rate scheduler is stepped (e.g. StepLR) with a step size of 5 and decay rate $\gamma = 0.1$ (Kingma & Ba, 2014). As a control, we finetune the original model and observe no meaningful change in denoising quality.

Space-Time Features are Better than Space-Only Features for Denoising. Table 3 tests the network on DAVIS validation set where the column indicates the attention module and each row indicates the Gaussian noise intensity (σ) (Pont-Tuset et al., 2017; Mou et al., 2021). COLA-Net originally uses cross-scale attention, which is a global, strided search. The results in Table 3 show space-time non-local attention with optical flow outperforms the original network by over 1 dB PSNR, and it is the best of all the attention modules. Figure 11 shows qualitative examples. The space-time search and the space-time attention module use a spatial window of size 15×15, a patch size of 7, and a query stride of 4. The space-time search uses a temporal window of size 2. We aggregate the non-local patches as a weighted sum of patches. Since denoising is inherently a task local to a region of pixels, the usefulness of space-time features may be expected. The utility of space-time features remains unclear for tasks depending on semantic information, which is often spread globally across the image.

Space-Time Attention is Faster Than Global Attention. Table 3's runtime and memory consumption is measured using a 5 frame video of resolution 230×230. This is the largest input video which fits on our 24 GB NVIDIA RTX 3090 GPU. The global attention module is over 3.5 times slower than our space-time search and consumes almost 20 times more memory. The space-only and space-time searches use our Shifted Non-Local Search module with a temporal window size of 1 and 5, respectively.

Table 3: **Upgrading Search-Only Search.** [PSNR↑/SSIM↑/ST-RRED↓] This table reports denoising results on the DAVIS dataset using various attention modules within COLA-Net. The original attention module uses a global, strided search named cross-scale attention. We report results using the noise intensities (σ^2) to match the original paper.

Dataset	σ	Original (Global)	Non-Local	Space-Time
DAVIS	15	34.26/0.915/1.3	34.14/0.912/1.4	**35.78/0.936/0.8**
	30	30.93/0.848/5.2	30.80/0.843/5.3	**32.41/0.878/2.6**
	50	28.67/0.784/13.6	28.51/0.776/13.5	**29.73/0.810/7.2**
Runtime (seconds)		1.703	0.242	0.439
Memory (GB)		14.449	0.730	0.736

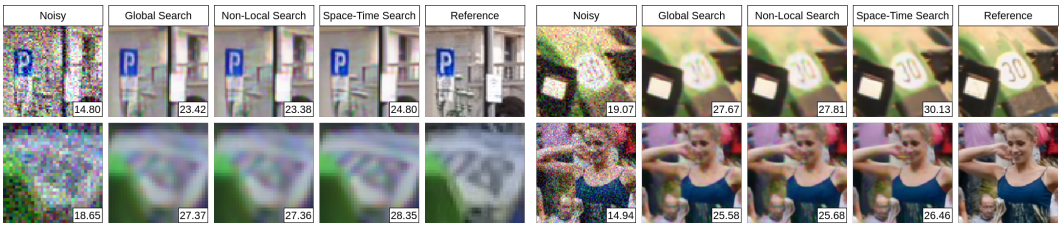

Figure 11: **Qualitatively Comparing Denoised Outputs.** [PSNR↑] Replacing COLA-Net's global, cross-scale attention with both the space-only and space-time search impacts restoration quality. The space-time search can restore details not available from a single frame.

6.3 Upgrading Space-Time Attention: Additional Details

Upgrading Guided Deformable Attention. A Guided Deformable Attention (GDA) module requires offsets, denoted $\boldsymbol{F}_{\text{out}}$, with shape $H \times W \times 9 \times 2$ where $\boldsymbol{F}_{\text{out}}[h_i, w_i, l]$ the (x, y) shift from (h_i, w_i) between frames at time $t-1$ and t. These offsets are output from a network whose input includes a single optical-flow-like offset, denoted $\boldsymbol{F}_{\text{in}}$ with shape $H \times W \times 2$. Written another way, $\boldsymbol{F}_{\text{out}} = \text{Auxiliary Network}(\boldsymbol{X}_{\text{in}}, \boldsymbol{F}_{\text{in}})$. We replace their auxiliary network with our Shifted Non-Local Search: $\_, \boldsymbol{F}_{\text{out},L} = \text{Shifted-NLS}(\boldsymbol{X}_{\text{in}}, \boldsymbol{F}_{\text{in}}, L)$ with $L = 9$ to match RVRT.

Training Details. The upgraded RVRT networks are trained for 90,000 iterations with a batch size of 8 with 10 frames each at resolution 256×256. Each batch is corrupted with Gaussian noise using a random noise parameter, $\sigma \sim \text{Uniform}[0, 50]$. Weight updates use the Adam optimizer with an initial learning rate of 4×10^{-4} and decrease with the Cosine Annealing learning rate scheduler (Kingma & Ba, 2014; Loshchilov & Hutter, 2016). RVRT's internal SpyNet model is initialized with pre-trained weights which are fixed for the first 30,000 iterations and learns with a 75% reduced learning rate. (Ranjan & Black, 2017b; Niklaus, 2018). Notably, we execute only 90,000 iterations instead of 600,000 due to limited computing resources.

Ablation Study. Our Shifted Non-Local Search uses hyperparameters not learned during network training. To better understand their impact on video denoising quality, we train networks for various configurations. The search stride, S_K, denotes the spacing between two points in the grid search. When $S_K = 0.5$ or $S_K = 1$, then each grid point is spaced $1/2$ or 1 pixel apart, respectively. Table 4 shows the results for a various number of search parameters. The best settings used a search stride of 0.5 and a spatial window of 9, outperforming the same window size with a search stride of 1. This suggests subpixel corrections to the predicted offsets may be more beneficial than a larger search radius. A search radius that is too large or too small also decreases the network quality. We hypothesis a large search radius will overfit to noise and a small radius is insufficient to properly correct errors.

Table 4: **Ablation Experiments for Shifted Non-Local Search.**[PSNR↑/SSIM↑/ST-RRED↓] A small spatial window ($W_s = 3$) is unable to correct errors and a large spatial window ($W_s = 15$) overfits to noise. A fractional search stride allows for subpixel correction.

σ	$W_s = 9, S_K = 0.5$	$W_s = 9, S_K = 1$	$W_s = 15, S_K = 1$	$W_s = 3, S_K = 1$
10	**38.90/0.967/0.004**	38.77/0.967/0.004	38.68/0.966/0.004	38.61/0.965/0.004
20	**35.58/0.936/0.012**	35.40/0.934/0.013	35.27/0.933/0.013	35.23/0.932/0.013
30	**33.68/0.907/0.024**	33.44/0.904/0.025	33.29/0.901/0.026	33.27/0.900/0.026
40	**32.35/0.881/0.040**	32.07/0.875/0.042	31.89/0.872/0.043	31.87/0.871/0.044
50	**31.30/0.880/0.027**	30.97/0.847/0.061	30.78/0.843/0.063	30.76/0.841/0.064

6.4 Space-Time Attention Network (STAN) for Video Denoising

Architecture. The macro-level architecture is designed to match the multi-scale architecture of UNet (Ronneberger et al., 2015). Each non-local block contains layer normalization, our space-time attention module, channel attention, and lastly Residual Swin Transformer Blocks (Hu et al., 2018; Liang et al., 2021). The block structure is inspired by the RVRT network (Liang et al., 2022b). The space-time attention layer is described in Section 3.1, and we use the aggregation scheme described Section 3.2. Notably, there are no positional embeddings. Our networks are set to $N_1 = 1$, $N_2 = 2$, and $N_3 = 4$ with 1, 4, and 12 heads. STAN uses TV-L1 for optical flow (Zach et al., 2007). Only the

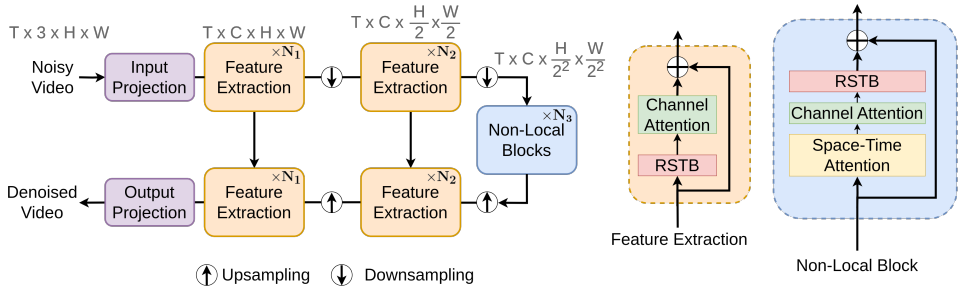

Figure 12: **Space-Time Attention Network (STAN) Architecture.** The STAN macro-level architecture design is inspired from UNet (Ronneberger et al., 2015). Each block's design is inspired by the RVRT network (Liang et al., 2022b).

first non-local block executes the space-time search while subsequent blocks use the previous layer's top-L offsets.

Experimental Details. The RVRT and STAN networks are trained for 240,000 iterations with a batch size of 8 using clip lengths of 16 with resolution 256×256. Each batch is corrupted with Gaussian noise using a random noise parameter, $\sigma \sim \text{Uniform}[0, 50]$. Weight updates use the Adam optimizer with an initial learning rate of 4×10^{-4} and decrease with the Cosine Annealing learning rate scheduler (Kingma & Ba, 2014; Loshchilov & Hutter, 2016). The internal SpyNet model is initialized with pre-trained weights which are fixed for the first 30,000 iterations and uses a learning rate reduced by 75%. (Ranjan & Black, 2017b; Niklaus, 2018). Notably, we executed only 240,000 iterations instead of 600,000 due to limited computing resources. Because our lab operates on a time-limited SLURM computer restricted to 4 GPUs, training time changes from 21 to 90 days, depending on the server's scheduler. Perhaps more than doubling the training time would close the gap of over 1 dB difference.

Reproducibility Description. While the authors of RVRT do not provide training code, a procedure is described in their paper. When following their training procedure, the network's quality is significantly worse than their original report. An open issue on RVRT's GitHub also reports a reproducibility issue. Since training a single RVRT network takes approximately 10 days, which costs about $5,875 on AWS[2], we do not have the resources for further investigation. We also note our SLURM-based training environment may also be a source of the issue. We use 4 GPUs for 4-hour increments. While is explicitly supported by open source code, bugs are still found (example issue 1 and issue 2).

Valid Conclusions Despite the Non-Reproducible Related Work. While we cannot reproduce the results from RVRT, we believe this does not change conclusions regarding our Shifted Non-Local Search module. Since the training procedure is copied directly from the RVRT paper and the procedure is identical for both RVRT and STAN, our conclusions remain scientifically supported.

Ablation Experiment. We include an ablation study to assess the impact of space-time attention's temporal window size in Table 5. Each column's configuration is trained from scratch for $90,000$ epochs, following the training procedure previously described. The testing dataset is Set8.

Table 5: **The Impact of the Temporal Windows.** [PSNR↑/SSIM↑/ST-RRED↓] The temporal window size changes the denoising quality of the STAN architecture. The temporal window size is the number of frames searched for each query point from each frame. When $W_t = 1$, no adjacent frames are searched.

σ	$W_t = 5$	$W_t = 3$	$W_t = 1$
10	**36.71/0.957/0.003**	36.59/0.956/0.003	36.42/0.953/0.003
20	**33.94/0.927/0.008**	33.77/0.925/0.008	33.51/0.920/0.009
30	**32.29/0.900/0.014**	32.10/0.896/0.015	31.82/0.890/0.016
40	**31.11/0.874/0.023**	30.92/0.870/0.024	30.74/0.862/0.026
50	**30.19/0.850/0.033**	30.00/0.845/0.035	29.72/0.838/0.038

[2]An 8-GPU Instance is $24.48 per hour https://aws.amazon.com/ec2/instance-types/p3/

7 PYTORCH EXAMPLE OF SPACE-TIME ATTENTION

```
# -- imports --
import torch as th
conv2d = th.nn.functional.conv2d
conv3d = th.nn.functional.conv3d
from einops import rearrange
import stnls

# -=-=-=-=-=-=-=-=-=-=-=-=-
#         Init
# -=-=-=-=-=-=-=-=-=-=-=-=-

# -- search & aggregate info --
ws = 5 # spatial window size
wt = 2 # temporal window size; searching total frames W_t = 2*wt+1
ps,K,HD = 3,10,2 # patch size, num of neighbors (aka "L"), num of heads
stride0,stride1 = 1,0.5 # query & key stride

# -- input video --
B,T,F,H,W  = 1,5,16,128,128 # batch size, frames, features, height, width
device = "cuda"
V_in = th.randn((B,T,F,H,W),device=device)
vshape = V_in.shape

# -- optical flows --
fflow = th.randn((B,T,2,H,W),device=device)
bflow = th.randn((B,T,2,H,W),device=device)

# -=-=-=-=-=-=-=-=-=-=-=-=-
#        Attention
# -=-=-=-=-=-=-=-=-=-=-=-=-

# -- transform --
proj_weights = th.randn((F,F,1,1),device=device)
q_vid = conv2d(V_in.view(-1,*vshape[2:]),proj_weights).view(vshape)
k_vid = conv2d(V_in.view(-1,*vshape[2:]),proj_weights).view(vshape)
v_vid = conv2d(V_in.view(-1,*vshape[2:]),proj_weights).view(vshape)

# -- search --
search = stnls.search.NonLocalSearch(ws,wt,ps,K,nheads=HD,
                                     stride0=stride0,stride1=stride1)
dists,inds = search(q_vid,k_vid,fflow,bflow)
# print(inds.shape) # B,HD,T,nH,nW,K,3; nH=(H-1)//stride0+1

# -- normalize --
weights = th.nn.functional.softmax(-10*dists,-1)

# -- aggregate --
gather_nl = stnls.agg.NonLocalGather(ps,stride0)
stacked = gather_nl(v_vid,weights,inds)
# stacked.shape = (B,HD,K,T,F',H,W) where F' = F/HD
V_out = rearrange(stacked,'b hd k t f h w -> (b t) (hd f) k h w')
proj_weights = th.randn((F,F,K,1,1),device=device)
V_out = conv3d(V_out,proj_weights,stride=(K,1,1))
V_out = rearrange(V_out,'(b t) f 1 h w -> b t f h w',b=B)
# print("V_out.shape: ",V_out.shape) # B,T,F,H,W
```