Online-trained Upsampler for Deep Low Complexity Video Compression
—Supplementary Material—

Jan P. Klopp
National Taiwan University
kloppjp@gmail.com

Keng-Chi Liu
Taiwan AI Labs
calvin89029@gmail.com

Shao-Yi Chien
National Taiwan University
sychien@ntu.edu.tw

Liang-Gee Chen
National Taiwan University
lgchen@ntu.edu.tw

7. Additional Results

7.1. Convergence Ablation

While fast encoding is not critical in offline encoding (e.g. as performed for video on demand), it can reduce resource usage and enable high latency online encoding (e.g. non-responsive content like sports events). Fig. 10 shows coding gain as a function of the training iterations. The graphs are annotated with the average use of a pixel as training data (at 100%, a pixel is seen once during training on average). As all datasets receive the same optimisation, this only depends on the frame size. For high resolutions, this number drops far below 100% for 250 iterations. Nevertheless, the gains drop only moderately, especially for the JVET A dataset. This observation indicates local generalisation that at least holds for data in the temporal/spatial vicinity of the training data. Hence even with few training cycles, the coding gains are still significant.

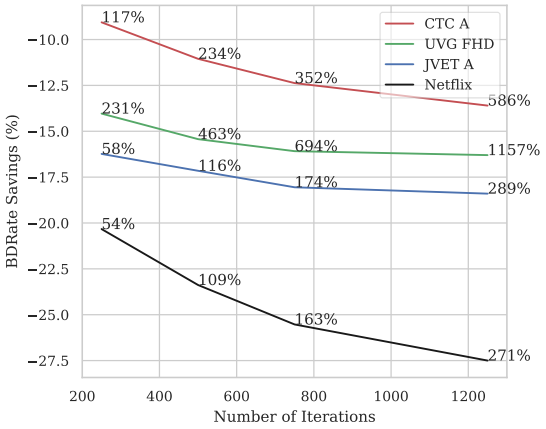

Figure 10. BDRate savings over number of optimisation iterations spent on one group of 32 pictures. Percentages indicate the amount of training pixels over the total number of pixels in a group.

7.2. Generalisation Ablation

Due to space constraints, the generalisation ablation for 128 frames did not fit in the main paper and can be found in Fig. 11. Like the results for 32 frames in the main paper, one can observe that halving the number of frames available for training does not severely impact the coding gain.

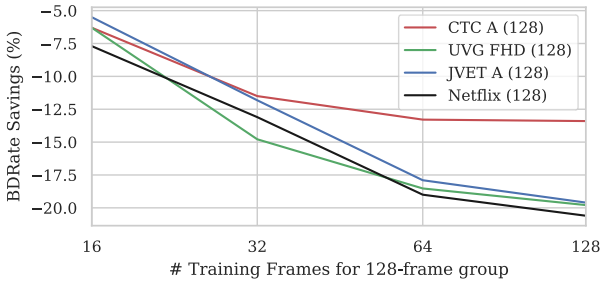

Figure 11. BDRate savings over the number of frames available for training in each group of pictures. Frames are taken from the beginning of the group. The group size is 128 frames.

7.3. Gain Distribution over Rate and Distortion

If our method mainly improved sequences with a high PSNR at a low data rate (i.e., low bandwidth demand), this may show great improvements but not translate into bandwidth reduction in practice. This issue depends on the distribution of sequences within a dataset. Figures 12, 13,

and 14 show the rate saving distribution over rate and PSNR for the remaining three quality steps (the one for QP=22 is in the main paper). Sequences in the bottom right corner are likely to be more challenging to encode efficiently as their PSNR is low while their rates are high. Their rates are almost an order of magnitude above the majority on the left-hand side. Similar to the graphic in the main paper, the gains provided by our algorithm do not follow any obvious distribution, and high bandwidth sequences do also achieve high gains.

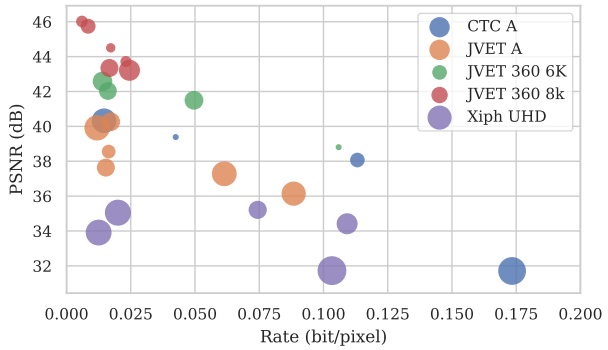

Figure 12. Distribution of rate savings over rate and distortion at test QP 27 (QP 22 for the underlying low resolution codec). Savings are indicated by the radius of each blob.

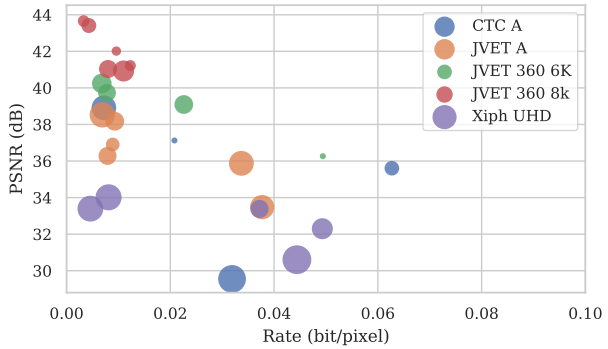

Figure 13. Distribution of rate savings over rate and distortion at test QP 32 (QP 27 for the underlying low resolution codec). Savings are indicated by the radius of each blob.

7.4. Average Rate Shares

In addition to the results in the main paper, Table 8 shows the average rate share of the network's parameters for other datasets. The dataset for which this data was presented in the main paper (UVG FHD) contains more low bandwidth sequences, not only due to the lower resolution but also because the content is simpler (less motion and texture in some sequences). The results presented here show that even under low-quality conditions, our algorithm's overhead remains in the single-digit range, hardly exceeding 3% for offline encoded sequences.

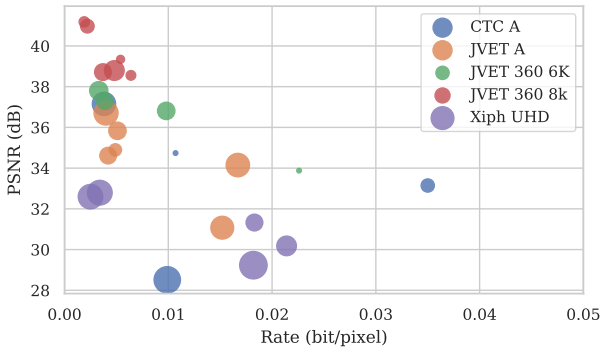

Figure 14. Distribution of rate savings over rate and distortion at test QP 37 (QP 32 for the underlying low resolution codec). Savings are indicated by the radius of each blob.

Table 8. Average rate share for different datasets, given for each QP of the conventional codec that encodes the low resolution signal. The zero-latency data refers to a signalling interval of 16 frames.

Dataset	Offline	Zero Lat.	17	22	27	32
JVET A	✓		0.4%	0.7%	1.2%	2.2%
CTC A	✓		0.4%	0.8%	1.6%	3.1%
JVET 360 6K	✓		0.3%	0.5%	1.3%	2.2%
JVET 360 8K	✓		0.3%	0.5%	1.0%	2.0%
Xiph UHD	✓		0.2%	0.4%	0.9%	1.9%
JVET A		✓	0.5%	1.2%	2.3%	4.4%
CTC A		✓	0.4%	1.2%	3.0%	7.1%

7.5. LPIPS

Learned Perceptual Image Patch Similarity [46] has been proposed as a learned metric for image fidelity. The metric has been tuned according to feedback from human raters viewing an original and a distorted image. Various kinds of noise have been applied to the original image, among them also JPEG compression. Table 9 shows the results of testing LPIPS on the PSNR-trained models, i.e. without tuning to the LPIPS metric, for three datasets. The results show similar or even higher improvements than testing on PSNR, indicating that our PSNR/MS-SSIM results are not overfitting to the particular metric. A simple intuition is that our method uses comparably few parameters to perform upsampling. Hence overfitting to a metric may be less likely than with more complex methods.

8. Quantisation

The number of bits reserved for weights (Q_w) and bias (Q_b) decreases with decreasing quality, i.e. increasing

Table 9. Bjontegaard Deltas for Rate and Distortion as measured by PSNR and LPIPS over x265 with tune `psnr`. LPIPS measures were obtained by optimising for PSNR (i.e. MSE). Negative rate savings indicate that our method requires fewer bits of code to deliver the same quality. All quality measures are taken in dB, LPIPS was converted to log scale (decibel) as follows: $\text{LPIPS}_{\text{dB}} = -10.0 \log(\text{LPIPS})$

	PSNR		LPIPS	
	ΔRate	ΔPSNR	ΔRate	ΔLPIPS
CTC A	-13.6%	+0.2954	-11.5%	+0.0295
JVET A	-18.4%	+0.4813	-25.6%	+0.4059
UVG FHD	-16.3%	+0.5156	-34.8%	+0.6175

quantisation parameter q as listed in Tab. 10.

Table 10. Quantisation bitdepth for weights and bias depending on quantisation parameter q.

QP q	Q_w	Q_b
$q \leq 20$	12	10
$20 \leq q \leq 25$	11	9
$25 \leq q \leq 30$	10	8
$30 \leq q$	9	8

9. Pre-Training

Figure 15 displays the concept of pre-training exploited in our work in a simplified manner.

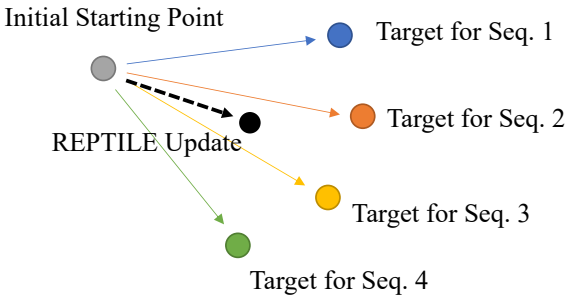

Figure 15. Simplified schematic of the behaviour of NN parameters in the pre-training process. By averaging the parameter updates (coloured arrows) for several different sequences a better initial starting point is reached (black dashed arrow). This new starting point is then used for a different sequence (i.e. not sequences 1 to 4) to obtain an objective measurement of the impact of REPTILE.

10. Algorithms

This section lists the algorithms we used to make reproduction easier and allow for a more detailed understanding. Table 11 lists the relative resolutions (with respect to the original sequence) for most variables used throughout the paper and the algorithm listings. Algorithm 1 contains the main steps for the encoding and the decoding procedure on a high level. Algorithm 2 describes in detail all steps required to compute the inference through the proposed network architecture. Algorithm 3 contains the training and quantisation procedures. The pre-training is listed in Algorithm 4. Lastly, Algorithm 5 describes the position encoding process.

Table 11. Resolutions relative to the resolution of the original sequence.

Variable	Symbol	Relative Resolution
Original sequence	x	1×1
Bilinear downsampled seq.	$x_{1/2}$	$1/2 \times 1/2$
Reconstructed downs. seq.	$\hat{x}_{1/2}$	$1/2 \times 1/2$
Internal features	z_{int}	$1/4 \times 1/4$
Encoded int. feat.	z_{Enc}	$1/2 \times 1/2$
Absolute positions	p_{Abs}	$1/2 \times 1/2$
Encoded positions	p_{Enc}	$1/2 \times 1/2$
Attention weightings	z_{Att}	$1/2 \times 1/2$
Reconstructed sequence	$\hat{x}$	1×1

11. Network Architectures

In this section, we list the network architecture for f_{Feat}, f_{Pos}, and f_{Denoise} in Tables 12, 13, and 14, respectively. Also, Tables 15 and 16 describe the network architecture used in the comparision with [16].

Algorithm 1: Encoding & Decoding Procedure.

Result: Network parameters $\boldsymbol{\Theta}$, reconstruction $\hat{\boldsymbol{X}}$, Conventional code $\boldsymbol{Y_{1/2}}$
Input: Conventional codec C, quantisation parameter q, hyperparameters γ, image sequence x, length of a frame group N_{Frame}

— Encode —
$\boldsymbol{\Theta} \leftarrow \emptyset$ # CNN parameters
$\boldsymbol{Y_{1/2}} \leftarrow \emptyset$ # Conventional code
if θ_{init} *not yet available* **then**
  $\theta_{\text{Init}} \leftarrow \text{PreTrain}(\dots)$ # Algorithm 4
end
foreach *frame group* i **do**
  # i: indices to all frames in the frame group
  $x_{1/2} \leftarrow \text{BilinRescale}(x[i], 0.5)$ # Resize
  $y_{1/2} \leftarrow C_{\text{Enc}}(x_{1/2}, q)$ # Conventional encoding
  $\hat{x}_{1/2}, z_{\text{int}} \leftarrow C_{\text{Dec}}(y_{1/2})$ # Rec. & Internal Signal
  $\theta_q \leftarrow \text{Training}(x[i], \hat{x}_{1/2}, z_{\text{int}}, \gamma, \theta_{\text{Init}})$ # Algorithm 3
  $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \theta_q$
  $\boldsymbol{Y_{1/2}} \leftarrow \boldsymbol{Y_{1/2}} \cup y_{1/2}$
end
— Decode —
$\hat{\boldsymbol{X}} \leftarrow \emptyset$ # Reconstruction
foreach $\left(\theta_q, y_{1/2}\right) \in \left(\boldsymbol{\Theta}, \boldsymbol{Y_{1/2}}\right)$ **do**
  $\hat{x}_{1/2}, z_{\text{int}} \leftarrow C_{\text{Dec}}(y_{1/2})$ # Rec. & Internal Signal
  $\hat{x} \leftarrow \text{Inference}(\hat{x}_{1/2}, z_{\text{int}}; \theta_q)$ # Algorithm 2
  $\hat{\boldsymbol{X}} \leftarrow \hat{\boldsymbol{X}} \cup \hat{x}$
end

Algorithm 2: Network Inference.

Result: Output $\hat{x}$ of resolution $H \times W$
Input: Network parameters θ, low resolution reconstruction $\hat{x}_{1/2}$, internal signals z_{int}

Prepare Input: Upsample U,V to $H/2 \times W/2$
$\hat{x}_{\text{in,Y}} \leftarrow \hat{x}_{1/2,Y}$
$\hat{x}_{\text{in,U}} \leftarrow \text{BilinRescale}\left(\hat{x}_{1/2,U}, 2.0\right)$
$\hat{x}_{\text{in,V}} \leftarrow \text{BilinRescale}\left(\hat{x}_{1/2,V}, 2.0\right)$
Prepare encoded positions (Algorithm 5)
$(\boldsymbol{\Omega}_t, \boldsymbol{\Phi}_t) \leftarrow \theta_{\text{Temp}}$ # Temporal frequencies/phases
$(\boldsymbol{\Omega}_s, \boldsymbol{\Phi}_s) \leftarrow \theta_{\text{Spat}}$ # Spatial frequencies/phases
$p_{\text{Enc}} \leftarrow \text{PosEnc}\left(H/2, W/2, \boldsymbol{\Omega}_t, \boldsymbol{\Phi}_t, \boldsymbol{\Omega}_s, \boldsymbol{\Phi}_s\right)$
Process internal features
$z_{\text{Enc}} \leftarrow f_{\text{Feat}}\left(z_{\text{int}}; \theta_{\text{Feat}}\right)$
Prepare input for attention network
$z_{\text{Att}} \leftarrow f_{\text{Pos}}\left(\text{concat}\left(z_{\text{Enc}}, p_{\text{Enc}}; \text{dim=1}\right); \theta_{\text{Pos}}\right)$
Compute the Denoiser's output
Resolution here is $H/2$, $W/2$
$o_{\text{Last}} \leftarrow \emptyset$ # Initialise last layer's output
foreach *Layer* l **do**
  # Provide input (layers 1, 5, 7)
  if l *requires input* **then**
    $o_{\text{Last}} \leftarrow \text{concat}\left(\hat{x}_{\text{in}}, o_{\text{Last}}; \text{dim} = 1\right)$
  end
  # Attention weighting (layer 6's input)
  if l *requires attention* **then**
    $o_{\text{Last}} \leftarrow o_{\text{Last}} \odot \text{sigmoid}\left(z_{\text{Att}}\right)$
  end
  $o_{\text{Last}} \leftarrow f_{\text{Denoise,Layer}=l}\left(o_{\text{Last}}; \theta_{\text{Denoise,Layer}=l}\right)$
end
Split output (channels 1-4 $\rightarrow Y$, 5 $\rightarrow U$, 6 $\rightarrow V$)
$o_Y, o_U, o_V \leftarrow \text{split}\left(o_{\text{Last}}; [1,2,3,4], [5], [6]\right)$
Unshuffle Y from the channel to spatial dimension
$o_Y \leftarrow \text{depth2space}(o_Y; [2,2])$
Resolution of o_Y is now: $H \times W$
Compute the final output
$\hat{x}_Y \leftarrow \text{BilinRescale}\left(\hat{x}_{1/2,Y}, 2.0\right) + o_Y$
$\hat{x}_U \leftarrow \text{BilinRescale}\left(\hat{x}_{1/2,U}, 2.0\right) + o_U$
$\hat{x}_V \leftarrow \text{BilinRescale}\left(\hat{x}_{1/2,V}, 2.0\right) + o_V$

Algorithm 3: Network Training.

Result: Quantised network parameters θ_q

Input: Initial network parameters θ_{Init}, target x, low resolution reconstruction $\hat{x}_{1/2}$, internal signals z_{int}, hyper parameters γ, loss function $\mathcal{L}$

$\theta_t \leftarrow \theta_{\text{Init}}$
for γ_{iter} *iterations* **do**
  $x_b, \hat{x}_{1/2,b}, z_{\text{int},b} \leftarrow$ SamplePatches $\left(x, \hat{x}_{1/2}, z_{\text{int}}; \gamma_{\text{patchSize}}, \gamma_{\text{batchSize}}\right)$
  $\hat{x}_b \leftarrow$ Inference $\left(\hat{x}_{1/2,b}, z_{\text{int},b}; \theta_t\right)$ # Algorithm 2
  $\theta_t \leftarrow$ Adam $\left(\frac{\partial \mathcal{L}}{\partial \theta}\left(x_b, \hat{x}_b\right); \gamma_{\text{lr}}\right)$
end
Quantise θ_t
$\theta^q \leftarrow \emptyset$
Iterate over all layers in all networks
foreach *layer* l **do**
  # Remove batch normalisation (before conv)
  # Only filter index f and channel index c shown
  $w'_{f,c} \leftarrow \frac{w_{f,c}}{\sigma_c}$
  $b'_f \leftarrow b_f - \sum_c \frac{w_{f,c}\mu_c}{\sigma_c}$
  # Q_w bits for weights, Q_b for bias (see Tab. 10)
  $Q_w, Q_b \leftarrow \gamma_{\text{Quant}}$
  # Value ranges
  $r_w \leftarrow 2^{Q_w - 1} - 1$
  $r_b \leftarrow 2^{Q_b - 1} - 1$
  # Weight quantisation per channel:
  $\alpha_c \leftarrow \text{round}\left(\frac{r_w}{\max_k |w_{k,c}|}\right)$
  $w^q_{f,c} \leftarrow \lfloor 0.5 + \frac{w'_{f,c}}{\alpha_c} \rfloor$
  # Bias quantisation:
  $\beta \leftarrow \text{round}\left(\frac{r_b}{\max_k |b'_k|}\right)$
  $b^q \leftarrow \lfloor 0.5 + \frac{b'}{\beta} \rfloor$
  $\theta_q \leftarrow \theta_q \cup \{\beta, b^q\}$
  foreach *channel* c **do**
    $\theta_q \leftarrow \theta_q \cup \left\{\alpha_c, w^q_{f,c}\right\}$
  end
end
No quantisation necessary for position encoding parameters
$\theta_{\text{Temp}} \leftarrow (\mathbf{\Omega}_t, \mathbf{\Phi}_t)$
$\theta_{\text{Spat}} \leftarrow (\mathbf{\Omega}_s, \mathbf{\Phi}_s)$

Algorithm 4: Pre-training.

Result: Initial network parameters θ_{Init} for quantisation parameter (quality) q

Input: N_{Seq} Sequences X, not including the sequence to be tested. Hyperparameter γ, quantisation parameter q, N_{meta} meta iterations.

$\epsilon \leftarrow 0.1$ # Meta learning rate
$\theta_{\text{Init}} \leftarrow$ rand() # Random initialisation
Random parameters don't work well for position encoding
$\theta_{\text{Temp}} \leftarrow \left([0.5, 0.25], [0, 0]\right)$ # $(\mathbf{\Omega}_t, \mathbf{\Phi}_t)$
$\theta_{\text{Spat}} \leftarrow \left([0.5, 0.25, 0.125], [0, 0, 0]\right)$ # $(\mathbf{\Omega}_s, \mathbf{\Phi}_s)$
for N_{meta} *iterations* **do**
  $\theta \leftarrow 0$ # Accumulate updates
  for $x \in X$ **do**
    $i \leftarrow$ Random frame group
    # En-/Decode frame group i
    $x_{1/2} \leftarrow \text{BilinRescale}(x[i], 0.5)$ # Resize
    $y_{1/2} \leftarrow C_{\text{Enc}}(x_{1/2}, q)$
    $\hat{x}_{1/2}, z_{\text{int}} \leftarrow C_{\text{Dec}}(y_{1/2})$
    # Train for 50 iterations, start with θ_{Init}
    $\theta \leftarrow \theta + \text{Training}(x[i], \hat{x}_{1/2}, z_{\text{int}}, \gamma, \theta_{\text{Init}})$
  end
  $\theta_{\text{Init}} \leftarrow (1 - \epsilon)\,\theta_{\text{Init}} + \frac{\epsilon}{N_{\text{Seq}}}\theta$
end

Algorithm 5: Position Encoding.

Result: Encoded positions p_{Enc}

Input: Sequence size: T frames of height H' and width W', encoding parameters $\mathbf{\Omega}_t, \mathbf{\Phi}_t$ (temporal) and $\mathbf{\Omega}_s, \mathbf{\Phi}_s$ (spatial)

Note that this happens for the down sampled
sequence, hence $H' = H/2$ and $W' = W/2$
$p_{\text{Abs}} \leftarrow \text{zeros}\left(\text{size} = [T, 3, H', W']\right)$
Generate normalised absolute positions with three
channels for temporal and two spatial dimensions
$p_{\text{Abs}}\left[t, :, h, w\right] \leftarrow \left[\frac{t}{T}, \frac{h}{H'}, \frac{w}{W'}\right]$
$p_{\text{Enc}} \leftarrow \emptyset$
Temporal encoding
foreach $(\omega_t, \phi_t) \in (\mathbf{\Omega}_t, \mathbf{\Phi}_t)$ **do**
    $p_{\sin} \leftarrow \sin\left(2\pi\omega_t p_{\text{Abs}}\left[:, 0\right] + \phi_t\right)$
    $p_{\cos} \leftarrow \cos\left(2\pi\omega_t p_{\text{Abs}}\left[:, 0\right] + \phi_t\right)$
    $p_{\text{Enc}} \leftarrow \text{concat}\left(\left[p_{\text{Enc}}, p_{\sin}, p_{\cos}\right], \text{dim} = 1\right)$
end
Spatial encoding
foreach $(\omega_s, \phi_s) \in (\mathbf{\Omega}_s, \mathbf{\Phi}_s)$ **do**
    # Vertical
    $p_{\sin} \leftarrow \sin\left(2\pi\omega_s p_{\text{Abs}}\left[:, 1\right] + \phi_s\right)$
    $p_{\cos} \leftarrow \cos\left(2\pi\omega_s p_{\text{Abs}}\left[:, 1\right] + \phi_s\right)$
    $p_{\text{Enc}} \leftarrow \text{concat}\left(\left[p_{\text{Enc}}, p_{\sin}, p_{\cos}\right], \text{dim} = 1\right)$
    # Horizontal
    $p_{\sin} \leftarrow \sin\left(2\pi\omega_s p_{\text{Abs}}\left[:, 2\right] + \phi_s\right)$
    $p_{\cos} \leftarrow \cos\left(2\pi\omega_s p_{\text{Abs}}\left[:, 2\right] + \phi_s\right)$
    $p_{\text{Enc}} \leftarrow \text{concat}\left(\left[p_{\text{Enc}}, p_{\sin}, p_{\cos}\right], \text{dim} = 1\right)$
end

Table 12. **Feature Network.** Input are the internal features z_{int}. Resolution refers to the relative resolution with respect to the original sequence. The computational complexities are given as operations per pixel of the original size. F/W stands for inference at test time, F/W (Trn) for the inference at training time (where BatchNorm is still present). BN is the BatchNorm update where μ and σ need to be computed. B/W refers to the backward computation, Grad. to the gradient computation. #W and #B are the numbers of weights and biases, respectively (BN is fused with Conv). Bias is disabled for all layers in this network. Note that the first two layers do not need to backpropagate because the input does not require a gradient.

Layer	Type	Channels	Kernel	Filters	Groups	Resolution	F/W	F/W (Trn)	BN	B/W	Grad.	#W	#B
1a	BN	6	1	6	6	$1/4 \times 1/4$		0.375	0.75	0	0		
1b	Conv	6	1	12	1	$1/4 \times 1/4$	4.5	4.5		0	4.5	72	0
2a	BN	12	1	12	12	$1/4 \times 1/4$		0.75	1.5	0.75	0		
2b	Conv	12	3×3	12	12	$1/4 \times 1/4$	6.75	6.75		6.75	6.75	108	0
3a	BN	12	1	12	12	$1/4 \times 1/4$		0.75	1.5	0.75	0		
3b	Conv	12	1	24	1	$1/4 \times 1/4$	18	18		18	18	288	0
4	Unshuffle	24	None	6	1	$1/2 \times 1/2$		0		0	0		
Σ							29.25	31.125	3.75	26.25	29.25	468	0

Table 13. **Position Network.** Input are the concatenated encoded internal features z_{Enc} and the encoded positions p_{Enc}. For the meaning of the columns, please refer to Table 12. Sigmoid is assumed to take 7 floating point operations to compute.

Layer	Type	Channels	Kernel	Filters	Groups	Resolution	F/W	F/W (Trn)	BN	B/W	Grad.	#W	#B
1a	BN	22	1	22	22	$1/2 \times 1/2$		5.5	11	5.5	0		
1b	Conv	22	1	12	1	$1/2 \times 1/2$	66	66		66	66	264	12
2a	BN	12	1	12	12	$1/2 \times 1/2$		3	6	3	0		
2b	Conv	12	3×3	12	12	$1/2 \times 1/2$	27	27		27	27	108	12
3a	BN	12	1	12	12	$1/2 \times 1/2$		3	6	3	0		
3b	Conv	12	1	14	1	$1/2 \times 1/2$	42	42		42	42	168	14
4a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
4b	Conv	14	3×3	14	12	$1/2 \times 1/2$	36.75	36.75		36.75	36.75	147	14
5	Sigmoid					$1/2 \times 1/2$	1.75	1.75		0.25	0		
Σ							173.5	188.5	30	187	171.75	687	52

Table 14. **Denoiser Network.** Input are the low resolution reconstruction $\hat{x}_{1/2}$ (concatenated before layers 5a and 7a). Layer 5c is the attention weighting layer, which has the same complexity as BatchNorm. For the meaning of the columns, please refer to Table 12.

Layer	Type	Channels	Kernel	Filters	Groups	Resolution	F/W	F/W (Trn)	BN	B/W	Grad.	#W	#B
1a	BN	3	1	3	3	$1/2 \times 1/2$		0.75	1.5	0	0		
1b	Conv	3	1	14	1	$1/2 \times 1/2$	10.5	10.5		0	10.5	42	14
2a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
2b	Conv	14	3×3	14	14	$1/2 \times 1/2$	31.5	31.5		31.5	31.5	126	14
3a	BN	14	1	14	14	$1/2 \times 1/2$	0	3.5	7	3.5	0		
3b	Conv	14	1	14	1	$1/2 \times 1/2$	49	49		49	49	196	14
4a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
4b	Conv	14	3×3	14	14	$1/2 \times 1/2$	31.5	31.5		31.5	31.5	126	14
5a	BN	17	1	17	17	$1/2 \times 1/2$		4.25	8.5	0	0		
5b	Conv	17	1	14	1	$1/2 \times 1/2$	59.5	59.5		59.5	59.5	238	14
5c	Hadamard	14	1	14	14	$1/2 \times 1/2$	3.5	3.5		0	0		
6a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
6b	Conv	14	3×3	14	14	$1/2 \times 1/2$	31.5	31.5		31.5	31.5	126	14
7a	BN	17	1	17	17	$1/2 \times 1/2$		4.25	8.5	4.25	0		
7b	Conv	17	1	14	1	$1/2 \times 1/2$	59.5	59.5		59.5	59.5	238	14
8a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
8b	Conv	14	3×3	14	14	$1/2 \times 1/2$	31.5	31.5		31.5	31.5	126	14
9a	BN	14	1	14	14	$1/2 \times 1/2$		3.5	7	3.5	0		
9b	Conv	14	1	6	1	$1/2 \times 1/2$	21	21		21	21	84	0
Σ							329	359.25	60.5	340.25	325.5	1302	112

Table 15. Network structure of [16], Y channel. Y channel is shuffled $2 \times 2 \rightarrow 4$ channels, hence all computations are performed at lower resolution. Output is unshuffled into 2×2 again. U/V channels are appended to the input.

Layer	Type	Channels	Kernel	Filters	Groups	Resolution	F/W	F/W (Trn)	BN	B/W	Grad.	#W	#B
1a	BN	6	1	6	6	$1/2 \times 1/2$		1.5	3		0		
1b	Conv	6	1	20	1	$1/2 \times 1/2$	30	30			30	120	20
2a	BN	20	1	20	20	$1/2 \times 1/2$		5	10	5	0		
2b	Conv	20	3×3	20	20	$1/2 \times 1/2$	45	45		45	45	180	20
3a	BN	20	1	20	20	$1/2 \times 1/2$		5	10	5	0		
3b	Conv	20	1	20	1	$1/2 \times 1/2$	100	100		100	100	400	20
4a	BN	20	1	20	20	$1/2 \times 1/2$		5	10	5	0		
4b	Conv	20	3×3	20	20	$1/2 \times 1/2$	45	45		45	45	180	20
5a	BN	20	1	20	20	$1/2 \times 1/2$		5	10	5	0		
5b	Conv	20	1	20	1	$1/2 \times 1/2$	100	100		100	100	400	20
6a	BN	20	1	20	20	$1/2 \times 1/2$		5	10	5	0		
6c	Conv	20	3×3	20	20	$1/2 \times 1/2$	45	45		45	45	180	20
7a	BN	20	1	20	20	$1/2 \times 1/2$		3.5	7	3.5	0		
7b	Conv	20	1	4	1	$1/2 \times 1/2$	20	20		20	20	80	
Σ							385	415	60	383.5	385	1540	120

Table 16. Network structure of [16], U/V channels. U/V channels are shuffled $2 \times 2 \rightarrow 4$ channels each, hence all computations are performed at lower resolution. Output is unshuffled into 2×2 again.

Layer	Type	Channels	Kernel	Filters	Groups	Resolution	F/W	F/W (Trn)	BN	B/W	Grad.	#W	#B
1a	BN	8	1	8	8	$1/4 \times 1/4$		0.5	1	0	0		
1b	Conv	8	1	20	1	$1/4 \times 1/4$	10	10		10	10	160	20
2a	BN	20	1	20	20	$1/4 \times 1/4$		1.25	2.5	1.25	0		
2b	Conv	20	3×3	20	20	$1/4 \times 1/4$	11.25	11.25		11.25	11.25	180	20
3a	BN	20	1	20	20	$1/4 \times 1/4$		1.25	2.5	1.25	0		
3b	Conv	20	1	20	1	$1/4 \times 1/4$	25	25		25	25	400	20
4a	BN	20	1	20	20	$1/4 \times 1/4$		1.25	2.5	1.25	0		
4b	Conv	20	3×3	20	20	$1/4 \times 1/4$	11.25	11.25		11.25	11.25	180	20
5a	BN	20	1	20	20	$1/4 \times 1/4$		1.25	2.5	1.25	0		
5b	Conv	20	1	20	1	$1/4 \times 1/4$	25	25		25	25	400	20
6a	BN	20	1	20	20	$1/4 \times 1/4$		1.25	2.5	1.25	0		
6c	Conv	20	3×3	20	20	$1/4 \times 1/4$	11.25	11.25		11.25	11.25	180	20
7a	BN	20	1	20	20	$1/4 \times 1/4$		3.5	7	3.5	0		
7b	Conv	20	1	8	1	$1/4 \times 1/4$	10	10		10	10	160	
Σ							103.75	114	20.5	103.5	103.75	1660	120