Stable Long-Term Recurrent Video Super-Resolution

Benjamin Naoto Chiche [1,2], Arnaud Woiselle [1], Joana Frontera-Pons [2,3], Jean-Luc Starck [2]

[1] Safran Electronics & Defense, F-91344 Massy, France

[2] AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Cité, F-91191 Gif-sur-Yvette, France

[3] DR2I, Institut Polytechnique des Sciences Avancées, F-94200 Ivry-sur-Seine, France

benjamin.chiche@safrangroup.com, arnaud.woiselle@safrangroup.com
joana.frontera-pons@cea.fr, https://orcid.org/0000-0003-2177-7794

Abstract

*Recurrent models have gained popularity in deep learning (DL) based video super-resolution (VSR), due to their increased computational efficiency, temporal receptive field and temporal consistency compared to sliding-window based models. However, when inferring on long video sequences presenting low motion (*i.e. *in which some parts of the scene barely move), recurrent models diverge through recurrent processing, generating high frequency artifacts. To the best of our knowledge, no study about VSR pointed out this instability problem, which can be critical for some real-world applications. Video surveillance is a typical example where such artifacts would occur, as both the camera and the scene stay static for a long time.*

In this work, we expose instabilities of existing recurrent VSR networks on long sequences with low motion. We demonstrate it on a new long sequence dataset Quasi-Static Video Set, that we have created. Finally, we introduce a new framework of recurrent VSR networks that is both stable and competitive, based on Lipschitz stability theory. We propose a new recurrent VSR network, coined Middle Recurrent Video Super-Resolution (MRVSR), based on this framework. We empirically show its competitive performance on long sequences with low motion.

1. Introduction

Video super-resolution (VSR) is an inverse problem that extends single-image super-resolution (SISR). While SISR aims to generate a high-resolution (HR) image from its low-resolution (LR) version, in VSR the goal is to reconstruct a sequence of HR images from the sequence of their LR counterparts. The idea behind VSR, which makes it fundamentally different from SISR, is that the fusion of several LR images produces an HR image. Therefore, VSR requires to accumulate information over a number of LR frames as large as possible. Classical VSR methods based on the image sequence formation model, knowledge on motion and iterative algorithms [2,12] could fill this requirement. However, these iterative algorithms are relatively slow and not suitable for real-world applications. Moreover, they perform poorly when the image sequence formation model and the assumptions on motion are too simplified.

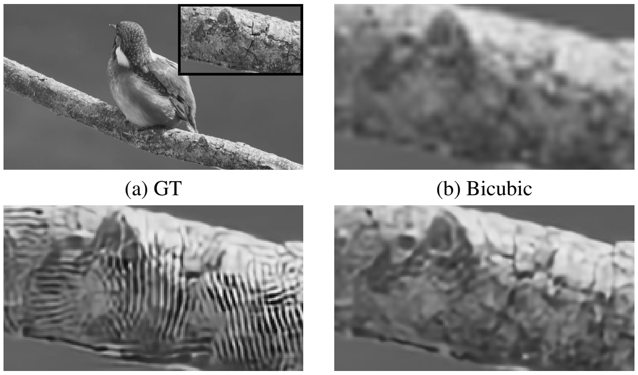


Figure 1. A comparison between a state-of-the-art recurrent VSR network (RLSP) and our proposed network. The former generates high frequency artifacts on long sequences with low motion. The proposed network does not.

VSR has recently benefited from DL methods [3,7–9,20,26,29] that can overcome some of the drawbacks of classical methods. Deep VSR networks can efficiently learn complex spatio-temporal statistics from a training dataset of natural videos, and once trained the reconstruction is faster. There are broadly two classes of deep VSR methods. The first one groups **sliding-window based** models. These models [8,9,13,26,29] take a batch of multiple LR frames as input to fuse them and reconstruct an HR frame. In most cases, this batch contains 5 to 7 LR frames. Therefore, the temporal receptive field—*i.e.* the number of LR frames that are used in order to super-resolve a frame—is limited to 7.

In contrast, methods introduced in [3, 7, 20], that build upon **recurrent** models, enable a larger temporal receptive field. In these networks, to super-resolve a frame at time step t, the hidden states and/or output computed in previous time step $t-1$ are taken as input, in addition to a batch of 1 to 3 LR frames. This recursion allows to propagate information through a large number of frames. As their input batch contains less LR frames and their network structures are mostly simpler, recurrent methods are faster than sliding-window based methods. Moreover, an inference of a recurrent model presents less redundant computations than the one of a sliding-window based model because each frame is processed only once. Finally, sliding-window based methods generate independent output HR frames, which reduces temporal consistency of the produced HR frames, resulting in flickering artifacts. This is not the case for recurrent VSR, in which information about previously super-resolved frame is part of the input at each time step. These considerations make recurrent methods more interesting from a realistic application-oriented point of view.

Because of computational and memory constraints, as well as vanishing and exploding gradients, recurrent VSR models are usually trained on sequences of 7 to 12 images. They are then deployed to super-resolve a sequence of any length. Some applications, such as video-surveillance, would require to super-resolve sequences of arbitrary length. However, recurrent models are not trained on these long sequences. Hence, there is no guarantee that they optimally perform on long sequences. In this study, we show that recurrent VSR networks generate high frequency artifacts when inferring on long video sequences presenting low motion. Such sequences contain parts of the scene that barely move, for instance when the camera is quasi-static. The super-resolution process creates high-frequency information which is accumulated in the long-term recurrence, creating artifacts and causing divergence. Fig. 1 illustrates this phenomenon. To the best of our knowledge, this work is the first study about VSR that raises this instability issue. This unexpected behavior can be critical for some real-world applications, like video surveillance in which both the camera and the scene stay static for a long time.

The structure of the article is the following. First, we review studies related to VSR and instabilities of recurrent networks. Then, based on Lipschitz stability theory, we propose a new framework of recurrent VSR network that is both stable and competitive on long sequences with low motion. After this, we introduce a new recurrent VSR network MRVSR as an implementation of this framework. Finally, we empirically analyze instabilities of existing recurrent VSR models on long sequences with low motion and show the stability and superior performance of the proposed network. A new long sequence dataset has been created for our experiments. We make it publicly available.

2. Related work

2.1. Recurrent video super-resolution

Authors of [20] were pioneers of recurrent VSR. They introduced FRVSR, in which the previous output frame is warped based on a dense optical flow estimation and fed back as an additional input to a super-resolution network at the next time step. The optical flow is estimated by another network and the two networks are jointly trained end-to-end. Hence, FRVSR operates *frame-recurrence*.

A more recent recurrent VSR architecture called *recurrent latent space propagation* (RLSP) was introduced in [3]. In this approach, the previous output frame and the previously estimated locality based hidden state are used as an extra input at the next time step. Compared to frame-recurrence, RLSP can be interpreted as maximizing the depth and width of the recurrent connection. In contrast to FRVSR, RLSP is based on implicit motion compensation. The overall architecture is computationally efficient, which enables RLSP to be the fastest VSR network at this time.

RSDN [7] is so far the recurrent VSR network that reportedly performs the best for relatively short sequences, according to its performance on Vid4 dataset, composed of 4 videos between 34 to 49 frames [12]. Its architecture presents a recurrent hidden state coupled with a hidden-state adaptation module and structure-detail decomposition. The input LR frames and the hidden state are decomposed into structure and detail components and fed to two interleaved branches to reconstruct the corresponding components of HR frames.

2.2. Instabilities of recurrent neural networks

Recurrent Neural Networks (RNNs) are difficult to train [18]. First of all, they involve backpropagation through time (BPTT), *i.e*. their unrolling through time, that is costly in terms of memory. Secondly, these architectures risk vanishing and exploding gradients issues. Correlated to this, RNNs are prone to divergence when inferring on long sequences. Authors of [15] showed, in the context of multi-layer and LSTM networks, that an RNN is stable if its Lipschitz constant is smaller than 1. To enforce this constraint, they proposed to clip singular values of the matrix associated with the recurrence map to 1. Several works circumvent vanishing and exploding gradients problems by setting all the singular values to 1 [1, 10, 14, 25, 27, 30].

Some studies are related to enforcing the Lipschitz constraint in the context of convolutional neural networks. Authors of [22] proposed to clip singular values of the block matrix of doubly block-circulant matrices associated with the convolutional layer. The work [16] explored *spectral normalization*, that relies on the power iteration to estimate maximal singular value of the reshaped kernel tensor of the convolutional layer. Authors of [6, 24] suggested not using this reshaping and instead proposed to directly use the kernel tensor in the power iteration. Finally, the work [21] proposed Stable Rank Normalization (SRN), an algorithm that seeks to enforce either the Lipschitz constraint or its softer version.

In the context of recurrent video denoising, authors of [23] pointed out instabilities. They first brought out unforeseeable, colorful and black mask-like artifacts in long-term video denoising. Then, inspired by studies on adversarial examples [5], they proposed a diagnosis tool to check stability of a trained recurrent video processing network. Finally, they improved upon the SRN algorithm to propose *Stable Rank Normalization of Layer* (SRNL). While SRN reshapes the kernel tensor of the convolutional layer, SRNL avoids this reshaping, similarly to [6, 24]. They applied this method on convolutional layers of their recurrent video denoising network and demonstrated its effectiveness.

To conclude this section, the following points summarize the limits of existing works regarding long-term recurrent VSR and our contributions:

- existing recurrent VSR networks have been only evaluated on relatively short generic sequences. Their performances have not been measured on long sequences. We demonstrate these networks perform poorly on such sequences when the motion amplitude is low, due to their recurrent structure. We create a novel dataset of long and low motion sequences, because existing datasets only contain sequences that either are too short or present fast scene motion;
- the relationship between instabilities and scene motion in video has not been investigated. We show that when inferring on long sequences presenting low motion, existing recurrent VSR models diverge;
- the Lipschitz constraint has not been applied on existing recurrent VSR networks. Indeed, in order to have a stable recurrent VSR network, we could first take one of these networks and directly apply a Lipschitz constraint to all convolutional layers in the recurrent loop. We show that this strategy fails when super-resolving long sequences with low motion;
- we design a recurrent VSR framework that is stable on long sequences with low motion, while not being globally Lipschitz constrained. We demonstrate the superior performance of a network based on this framework.

3. Method

3.1. Stability of recurrent video processing models

A recurrent video processing model is determined by a *recurrence map* $\phi^L : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ and an *output map* $\psi : \mathbb{R}^n \to \mathbb{R}^c$. The recurrent information $h_t \in \mathbb{R}^n$ and the output image $\hat{y}_t \in \mathbb{R}^c$ are updated at each time step t as follows:

$$\begin{cases} h_t = \phi^L(h_{t-1}, x_t) \\ \hat{y}_t = \psi(h_t) \end{cases} \tag{1}$$

where $x_t \in [0,1]^d$ is an input image provided at time t.

The recurrent model is *Lipschitz stable* if ϕ^L is *contractive* in h *i.e.* if ϕ^L is L-Lipschitz in h with $L < 1$ (the superscript in ϕ^L highlights this Lipschitz continuity). L is the Lipschitz constant of ϕ^L. This stability ensures that the full recurrent system is globally stable when running the network an arbitrary number of times, avoiding any divergence. Assume that ϕ^L is composed of K convolutional layers interspaced with ReLU non-linearities. Each convolutional layer can be encoded by a weight matrix, obtained from the layer's kernel tensor as a block matrix of doubly block-circulant matrices. Because Lipschitz constant of the ReLU activation is 1, L is upper-bounded by the product of the spectral norms of the weight matrices of the convolutional layers:

Proposition 1. *For a recurrent model ϕ^L constituted of K convolutional layers with weight matrices $W_1, ..., W_K \in \mathbb{R}^{n\times n}$ interspaced with ReLU non-linearities, the Lipschitz constant L of ϕ^L verifies:*

$$L \leq \prod_{k=1}^{K} ||W_k|| \tag{2}$$

where $||.||$ is the spectral norm.

Given this inequality, the Lipschitz stability can be ensured under the hard Lipschitz constraint:

Constraint 1. ***Hard Lipschitz constraint (HL)***
$\forall k \in [\![1, K]\!]$, *we impose* $||W_k|| \leq 1$.

However, the upper bound in Eq. (2) mostly overestimates L. For example, if ϕ^L is constituted of 2 convolutional layers with weight matrices W_1 and W_2, the only case where $L = ||W_1|| \cdot ||W_2||$ is when the first right singular vector of $||W_1||$ and the first left singular vector of W_2 are aligned. Hence, the constraint is overly restrictive. One can thus decide to relax the latter, leading to the soft Lipschitz constraint:

Constraint 2. ***Soft Lipschitz constraint (SL)***
$\forall k \in [\![1, K]\!]$, *we set* $||W_k|| = \alpha > 1$ *and minimize srank*(W_k) *based on training data, where srank is the Stable rank.*

Stable rank is an approximation of the rank operator that is stable under small perturbations of the matrix. This soft constraint does not theoretically guarantee the Lipschitz stability, so it is important to empirically verify the non divergence.

To enforce these constraints in the context of convolutional neural networks, *Stable Rank Normalization of Layer* (SRNL) can be applied to a convolutional layer during the training stage. This sets the spectral norm of the matrix of this layer to a desired value α and minimizes the stable rank of the matrix during training, controlled by β. α and β are among hyperparameters of the algorithm. When $\beta = 1$, it is equivalent to performing spectral normalization on the matrix. After training, a normalization step is required just before test time, so the algorithm does not introduce any overhead in runtime and model size at inference time.

3.2. Unconstrained Stable Recurrent VSR framework

In approaches such as RLSP, FRVSR and RSDN, every convolutional layer of super-resolving networks is recurrent within feedback loops. This seeks to increase the depth and width of the recurrent connection by giving the hidden state and the previous output to the input of super-resolving networks. Therefore, these layers both incorporate past information and contribute to the deconvolution task. Adopting the notations from Eq. (1), in these networks ψ is reduced to the identity mapping (followed by pixel shuffling or transposed convolutions). In order to have a stable recurrent VSR network, a naive approach would be to directly apply SRNL to one of these VSR networks. However, this approach presents some difficulties.

First, we applied SRNL to RLSP with $(\alpha, \beta) = (2.0, 0.1)$ and empirically verified that SL was not capable of removing the artifacts on long sequences (Fig. 4d). Second, we did the same experiment with $(\alpha, \beta) = (1.0, 1.0)$ to enforce HL and this resulted in a stable network but with poor VSR performance (detailed in Sec. 5.2). This is because the resulting architecture has been constrained to be globally 1-Lipschitz, and a successful super-resolving function—that operates both upsampling and deconvolution—cannot be 1-Lipschitz; since some frequencies need to be boosted as the Wiener filter does in the optimal linear case. This is not the case for a denoising function, that can be 1-Lipschitz while correctly performing.

Considering these points, we define a new framework of recurrent VSR network that is stable and performs competitively on long sequences:

Definition 1. *An* ***Unconstrained Stable Recurrent VSR*** *network is defined by an input network* $\xi : [0,1]^{d\times(2T+1)} \to \mathbb{R}^d$*, a contractive recurrent network* $\phi^L : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ *and an output network* $\psi : \mathbb{R}^n \to \mathbb{R}^c$*. The features* z_t*, the hidden state* h_t *and the output image* $\hat{y}_t$ *are updated at each time step* t *as follows:*

$$\begin{cases} z_t = \xi(X_t) \\ h_t = \phi^L(h_{t-1}, z_t) \\ \hat{y}_t = \psi(h_t) \end{cases} \tag{3}$$

where $X_t = \{x_t\}_{t-T\leq t\leq t+T} \in [0,1]^{d\times(2T+1)}$ *is an input batch of LR images provided to the network at* t *and* $2T+1$ *denotes the size of the batch.*

Let ϕ^L *be constituted of* K *convolutional layers with weight matrices* $W_1, ..., W_K \in \mathbb{R}^{n\times n}$ *interspaced with ReLU activations.* ϕ^L *is contractive in* h *based on the hard Lipschitz constraint:* $\forall k \in [\![1, K]\!]$, $||W_k|| \leq 1$.

Stable: all the layers in the inner recurrent loop of such a network are contractive, which guarantees its stability over time.

Unconstrained: such a network is not globally constrained in terms of Lipschitz continuity, due to its non contractive input and output networks which can keep their full expressiveness.

Most of the deconvolution task is done by ξ and ψ. ϕ^L incorporates past information. When ξ and ψ are simultaneously identity mappings, the *unconstrained* property is lost, as the network becomes globally 1-Lipschitz. This is the case encountered when imposing HL on all convolutional layers of networks such as RLSP, FRVSR and RSDN.

3.3. Middle Recurrent Video Super-Resolution

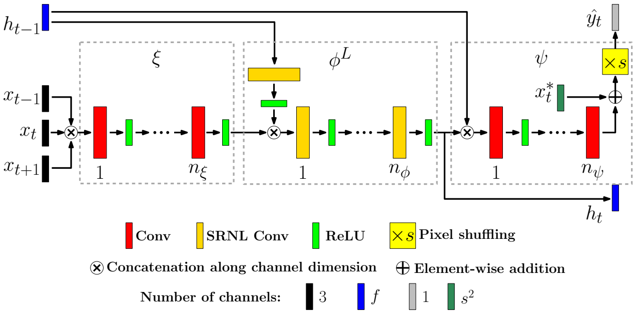


Figure 2. MRVSR architecture. SRNL Conv denotes convolutional layer under HL enforced by SRNL. Each convolutional layer uses 3×3 kernel with stride 1 and outputs f feature maps ($f = 128$ in our study), except the last one which outputs $s^2 = 16$ feature maps, where s is the scaling factor. The network outputs the brightness channel Y of YCbCr color space. Cb and Cr channels are upsampled independently with bicubic interpolation. Input LR frames $\{x_i\}_{t-1\leq i\leq t+1}$ are in RGB colorspace. Besides, x_t is converted from RGB to Y and replicated $s^2 = 16$ times in the channel dimension, which gives x_t^* for the residual connection. Pixel shuffling rearranges elements in a tensor of shape $(C \times s^2, H, W)$ to a tensor of shape $(C, H \times s, W \times s)$.

As an implementation of the proposed framework, we design a new network coined **Middle Recurrent Video Super-Resolution** (MRVSR). Its architecture is illustrated in Fig. 2. The first part of the network, ξ, has a feed-forward architecture with n_ξ convolutional layers and interspaced ReLU activations. The second part ϕ^L is composed of n_ϕ+1 convolutional layers under HL and interspaced ReLU activations. The third part ψ has a feed-forward architecture with n_ψ convolutional layers interlaced with ReLU activations and followed by a pixel shuffling layer. This part takes as input the current hidden state h_t and the hidden state from the previous time step. This mecanism, called *feature-shifting*, is helpful to promote temporal consistency between two successively output frames.

Incorporating past information via the recurrent connection is a simpler task than deconvolution. This can be illustrated revisiting the traditional, non DL based Shift-and-Add agorithm [2]. In the latter, historical information is captured via averaging or median aggregating past frames after projection on a HR grid and motion compensation. Averaging or median aggregating are rather simple mathematical operations. Therefore, n_ϕ can be smaller than $n_\xi + n_\psi$. In practice, one can fix $n_\xi + n_\phi + n_\psi$ to satisfy some constraint on computational cost, set a small value for n_ϕ and then select n_ξ and n_ψ. In our setting, we have found that under the condition $n_\xi + n_\phi + n_\psi = 7$ (that enables both fast computations and good performance), the value $n_\phi = 1$ lead to the best performance among other values of n_ϕ on our validation set (described in Sec. 4.2).

4. Experiments

4.1. Networks

For comparison, we implement the following state-of-the-art recurrent VSR networks in Pytorch [19]: FRVSR 10-128 [20], RSDN 9-128 [7] and RLSP 7-128 [3]. The numbers after each network respectively indicate the number of repeated building blocks and the number of filters in each convolutional layer. These hyperparameters enable reasonably fast training and testing and satisfactory performance on short sequences. In the following, we omit these numbers for simplicity. For RSDN, our implementation is based on the official codes released by its authors.[1] Additionally, we implement modified RLSP where all its layers have been normalized by SRNL with hyperparameter sets $(\alpha, \beta) = (2.0, 0.1)$ and $(\alpha, \beta) = (1.0, 1.0)$ to enforce the soft and hard Lipschitz constraints respectively. We call these networks RLSP-SL and RLSP-HL.

We compare these networks against the proposed MRVSR. We select (n_ξ, n_ϕ, n_ψ) so that $n_\xi + n_\phi + n_\psi = 7$ for the reason stated in Sec. 3.3. This number equals the number of convolutional layers in RLSP (excluding the layer that processes the hidden state), which yields fair comparison. Among MRVSR with different sets (n_ξ, n_ϕ, n_ψ), the network with $(n_\xi, n_\phi, n_\psi) = (3, 1, 3)$ was the best performing model on our validation set. Therefore, in Sec. 5 we only report performances recorded by MRVSR with this hyperparameter set. We use SRNL with $(\alpha, \beta) = (1.0, 1.0)$ to impose the HL.

In order to measure the benefit from constrained recurrence map, we also implement MRVSR without its recurrence and feature-shifting, which coincides with RLSP without its recurrence. This can be seen as an extension of SISR that takes 3 consecutive LR frames as an input at each time step. Its architecture is feed-forward with 7 convolutional layers with interlaced ReLU activations. We call this network RFS3 for **R**esidual **F**usion **S**huffle network with **3** input frames. This network will serve as baseline against recurrent models. In addition, we also implement RFS with an input batch of 7 LR frames, that we call RFS7. This serves as a representative sliding-window based model to compare against MRVSR, because most of sliding-window based VSR models take a batch of 5 to 7 LR frames.

4.2. Datasets

We prepare the training dataset in a similar way as in [3]. From the 37 high resolution Vimeo videos that were used in this study, after downsampling them by a factor of 2 we extract 40,000 random cropped sequences of size $I \times 256 \times 256 \times 3$, where $I \geq 12$. The delimiting keyframes are excluded from the sequence. At training time, we sample random sub-sequences of these crops with length 12. By excluding the first and the last frames, we obtain ground truth (GT) sequences with length 10. The first and last frames of the sampled sequences are used to produce x_{-1} at the beginning and x_{10} at the end. Data augmentation (random flip/transposition) is also employed.

We also prepare a validation set of 4 sequences. They come from videos with no constraints on motions of objects and count between 30 and 50 frames each.

We introduce a new test set of long sequences in which the camera is quasi-static and foreground objects move. This dataset will be complementary to the existing datasets (Vid4 [12], REDS [17] and Vimeo-90K [28]) which contain only videos that either are short, or present fast scene motion. To generate this new dataset, we download videos from `vimeo.com` and `youtube.com` and extract 4 sequences with quasi-static scene and moving objects inside. The first two of them are respectively Full HD and HD Ready and the two others are 4K. The HD and 4K sequences are downsampled respectively by a factor of 2 and 4. These 4 sequences respectively have the following lengths in number of frames: 379, 379, 379 and 172. They constitute the test dataset we call **Quasi-Static Video Set**. We limited the lengths of the sequences to 379 to ensure dataset homogeneity, but the video containing the first sequence contains a much larger number of frames. Therefore, we have also prepared a longer version of the first sequence called *Sequence 1-XL*. The latter contains 8782 frames. All of these sequences are available on `https://github.com/bjmch/MRVSR`.

The train and validation sets contain standard, relatively

[1] `https://github.com/junpan19/RSDN`

short sequences with no constraints on motion, whereas the test set contains long sequences with low motion. It aims at testing the capability of networks trained on short sequences to work on real-life long sequences that may have low-motion periods. We remind the reader that training recurrent networks on such long sequences is not realistic for reasons explained in Sec. 1, so the generalization gap between short and long sequences cannot be addressed with training data.

We additionally compare the reconstruction performances on the standard Vid4 dataset.

From each of the training, validation and test sequences in HR space, the corresponding LR sequence is generated by applying gaussian blur with σ and sampling every $s = 4$ pixel in both spatial dimensions. We set $\sigma = 1.5$, except when testing RSDN. In the case of this network, we use the pre-trained weights available on its official github repository. We thus adapted the codes of the corresponding degradations that are available on this repository to generate the LR sequence and the value of $\sigma = 1.6$ was used.

4.3. Training procedure and evaluation

All of the networks we prepare are trained from scratch after the Xavier initialization [4], except RSDN. The loss function is pixel-wise mean-squared-error between pixels in the brightness channel Y of YCbCr color space of GT frames and the network's output. The networks are trained with Adam optimizer [11] and a batch size of 4. The learning rate starts at 10^{-4} and is divided by 10 after the 200th and 400th epochs. RFS3, RFS7 and MRVSR are trained for 600 epochs. Other models except RSDN are trained between 400 and 600 epochs until convergence, based on train and validation losses.

We numerically evaluate the networks based on frame PSNR and SSIM. Qualitative evaluation that checks the presence of artifacts is of equal importance. We also assess the temporal consistency by examining temporal profiles from output sequences.

Moreover, the diagnosis tool from [23] can be used in order to visualize Spatio-Temporal Receptive Field (STRF) of a recurrent network. This tool, that is inspired by studies on adversarial examples [5], works as follows: given a trained recurrent video processing network, it looks for an input sequence $X = (x_{-\tau}, ..., x_{\tau})$ that is optimized to maximize the response at the center pixel in the output sequence $Y = (y_{-\tau+1}, ..., y_{\tau-1})$. To do so, the L1 norm of the center pixel $|p|$ in y_0 is maximized. This optimization only affects pixels in X that have an effect on p. Therefore, the optimized sequence X can be interpreted as a visualization of the STRF for the pixel p. τ is typically set to 40, values of pixels in X are randomly initialized between 0 and 1 and images in X have dimensions $64 \times 64 \times 3$. In our experiment, the optimization is solved using gradient descent and Adam optimizer for 1500 iterations. The learning rate starts at 1 and is divided by 10 after 750 and 1250 iterations.

5. Results

5.1. Performance of existing recurrent networks

Fig. 3 shows the evolution of the PSNR per frame for some of the networks, averaged over the first three sequences of Quasi-Static Video Set. The curve of RFS3 is taken as a baseline and subtracted to the other ones, and the resulting curves are displayed. We see that until a relatively small number of processed frames, existing recurrent networks (RLSP, RSDN and FRVSR) perform optimally and remain better than the baseline model. But at a certain point their performance drop and they become worse than the baseline model, indicating that the recursion integrates harmful information at each new frame. This can be seen as divergence.

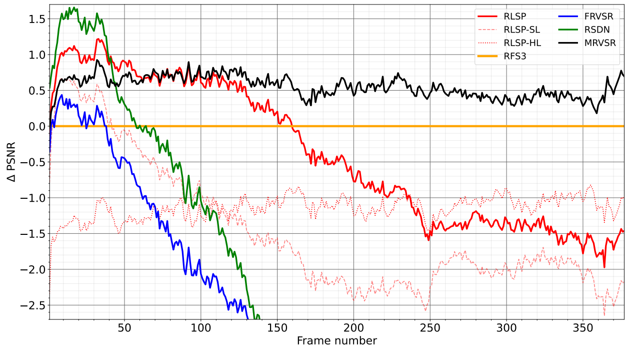


Figure 3. Evolution of PSNR on Y channel per frame averaged over the first three sequences of the Quasi-Static Video Set. We substract the curve of the RFS3 baseline and the graph shows these differences.

Tab. 1 summarizes the performances of the networks on the Quasi-Static Video Set. It summarizes the performances of the methods at the beginning of the sequences, through the entire sequences, and at the end of the sequences. The table conforms with the curves shown on Fig. 3. Based on reported performances, at the beginning of the sequences RLSP and RSDN perform better than the baseline RFS3. However, at the end of the sequences these networks and FRVSR have diverged and perform worse than RFS3. The differences in performance on the last 50 reconstructed frames between RFS3 and respectively RLSP, FRVSR and RSDN are -1.50, -4.39 and -4.09 in PSNR and -0.0029, -0.0790 and -0.0362 in SSIM. They represent in average -3.33dB in PSNR and -0.0394 in SSIM. This performance drop is due to the generation and accumulation of high frequency artifacts. These artifacts appear on objects that barely move. Example artifacts are shown on Figs. 4a to 4c which show a frame near the end of the first sequence

Model	First 50	All	Last 50
Bicubic	30.08 / 0.8362	30.05 / 0.8356	30.11 / 0.8387
RFS3	32.20 / 0.8911	32.04 / 0.8886	32.07 / 0.8911
RFS7	32.38 / 0.8945	32.23 / 0.8921	32.26 / 0.8943
FRVSR	32.15 / 0.8947	29.16 / 0.8442	27.68 / 0.8121
RSDN	33.46 / 0.9181	29.82 / 0.8788	27.98 / 0.8549
RLSP	33.08 / 0.9099	31.67 / 0.8964	30.57 / 0.8882
RLSP-SL	32.45 / 0.8991	30.62 / 0.8708	29.98 / 0.8627
RLSP-HL	30.98 / 0.8618	30.91 / 0.8608	30.95 / 0.8630
MRVSR	32.80 / 0.9030	32.62 / 0.9007	32.62 / 0.9026

Table 1. Mean PSNR / SSIM on Y channel of Quasi-Static Video Set. The metrics are measured excluding the first 3 and last 3 GT frames. 'First 50' means the metrics are computed at the beginning of the sequences *i.e.* on the first 50 reconstructed frames. 'All' means the metrics are computed through the entire sequences *i.e.* on all reconstructed frames. 'Last 50' means the metrics are computed at the end of the sequences *i.e.* on the last 50 reconstructed frames. Red: the best result. Blue: the second best result.

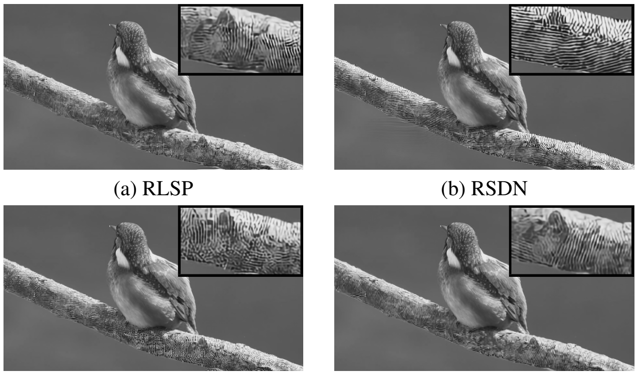

(a) RLSP
(b) RSDN
(c) FRVSR
(d) RLSP-SL

Figure 4. A frame near the end of the first sequence of Quasi-Static Video Set (the 376th frame) reconstructed from state-of-the art recurrent networks, and RLSP-SL. The Y channel is visualized. The networks generate high frequency artifacts on the branch, which is a quasi-static object.

of Quasi-Static Video Set (the 376th frame) reconstructed by each network.

Behavior analysis: These existing recurrent networks are trained to optimize their performance on a very low number of frames (at most 10). In this setting, it is beneficial to the network to produce rapidly a huge amount of details in the output sequence. These high frequency details grow in strength with time, but they are not fed back into the network more than 10 times, so the optimization process is not trained to manage their increase after this period. When inferring on long sequences, these details keep accumulating long after the short-term network's training regime, which produces visible artifacts that diverge over time. In the presence of strong motion, even with short-term training, the network learns to forget the past information, which is inconsistent with the new one. The newly created high frequency content is forgotten at the same time, preventing divergence on scenes with enough motion. In the first sequence of the Quasi-Static Video Set, the bird moves regularly, which is why artifacts do not have time to appear on the bird itself, as can be seen on Fig. 4.

5.2. Constraining existing recurrent networks

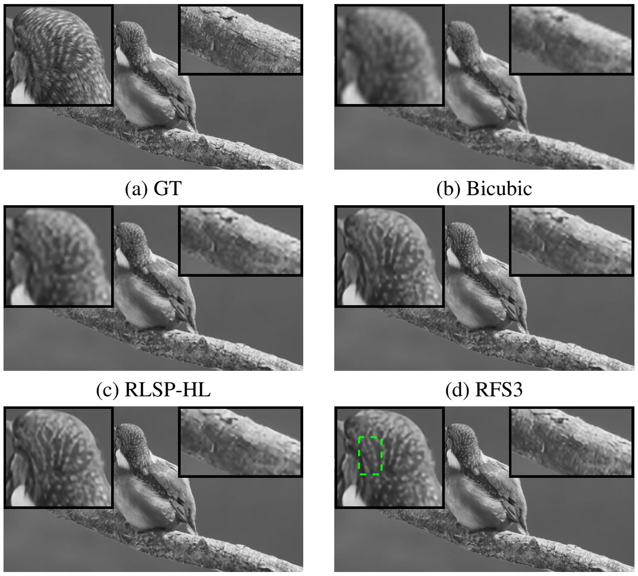

(a) GT
(b) Bicubic
(c) RLSP-HL
(d) RFS3
(e) RFS7
(f) MRVSR

Figure 5. The 376th frame of the first sequence of Quasi-Static Video Set, reconstructed from methods that are stable by design (non recurrent or under HL). MRVSR presents the best quality.

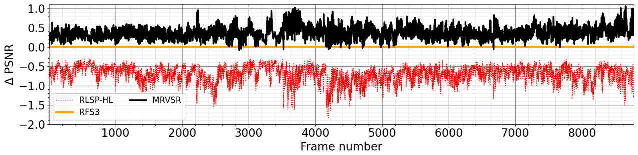


Figure 6. Evolution of PSNR on Y channel per frame on *Sequence 1-XL*. We substract the curve of the RFS3 baseline and the graph shows these differences.

SL: RLSP-SL faces the same issues as existing recurrent networks. After being better than the baseline RFS3 at the beginning of the sequences, it diverges (Fig. 3). It generates high frequency artifacts (Fig. 4d) and its performance at the end of the sequences is poor, as shown in Tab. 1 (-2.09dB in mean PSNR and -0.0284 in mean SSIM compared to RFS3 on the last 50 reconstructions). This proves that SL is not enough to prevent the divergence.

HL: RLSP-HL also obtains an overall poor performance (-1.13dB in average PSNR and -0.0278 in average SSIM compared to RFS3 based on all reconstructed frames, according to Tab. 1). Its reconstruction performance is stable on a long sequence (Figs. 3 and 6), but the reconstructed image is blurred (Fig. 5c). This is because RLSP-HL is globally constrained to be 1-Lipschitz. Thus, as stated in Sec. 3.2, it is poorly suited to the deconvolution task.

5.3. Performance of the proposed network

Model	RFS3	FRVSR	RSDN	RLSP	MRVSR
PSNR	26.43	26.69	27.92	27.46	26.90
# Param. (M)	0.77	5.05	6.18	1.08	1.21
Runtime (ms)	9	55	56	11	12

Table 2. Mean PSNR on Y channel of Vid4, model size and runtime. PSNR values for FRVSR, RLSP and RSDN are taken from their papers. Runtime is measured on an LR size of 180×320, an Intel I9-10940X CPU and one NVIDIA TITAN RTX GPU.

At the beginning of the quasi static sequences (Fig. 3 and Tab. 1) MRVSR cannot match RLSP and RSDN, but performs better than the baseline RFS3 and FRVSR. This performance is compatible with the results on Vid4 (Tab. 2), where MRVSR is 0.56dB behind the unconstrained similar network RLSP. This is due to the Lipschitz constraint on MRVSR, built to ensure its long-term stability at the price of a lower short-term performance.

When considering long-term performance on sequences with low motion, MRVSR gives the best results. Figs. 3, 5f and 6 show that MRVSR does not diverge and does not generate any artifact. According to Tab. 1, MRVSR achieves the best mean performance on the test set, based on all reconstructed frames as well as focusing on the last 50 reconstructed frames. Because MRVSR and RFS3 take the same number of input frames—namely three—the differences of $+0.58$ dB in average PSNR and +0.0121 in average SSIM computed on all reconstructed frames represent the benefit brought by the contractive recurrence map of MRVSR. Moreover, considering that RFS7 takes an input batch of 7 frames, the fact that MRVSR outperforms RFS7 ($+0.39$dB in average PSNR and $+0.0086$ in average SSIM) shows that the temporal receptive field enabled by its contractive recurrence accounts for more than 7 frames. This is confirmed in Fig. 7, where the temporal receptive field of MRVSR spans around 28 frames, which is much larger than the usual length (*i.e.* 7) of temporal receptive field of sliding-window based models. Moreover, temporal profiles produced by MRVSR are less noisy and sharper than the ones produced by RFS3 and RFS7. This shows the contractive recurrence map of MRVSR additionally enables increased temporal consistency. Visually speaking, sequences generated by MRVSR present less flickering artifacts than sequences produced by RFS7 and RFS3. Fig. 8 displays examples of temporal profiles for the first sequence of Quasi-Static Video Set. Finally, MRVSR presents the best long-term reconstruction in terms of visual quality. Some examples can be observed in Fig. 5.

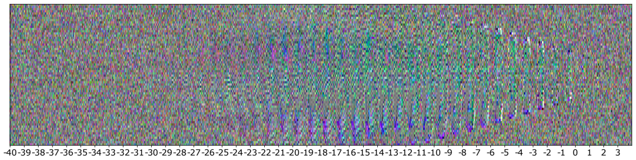
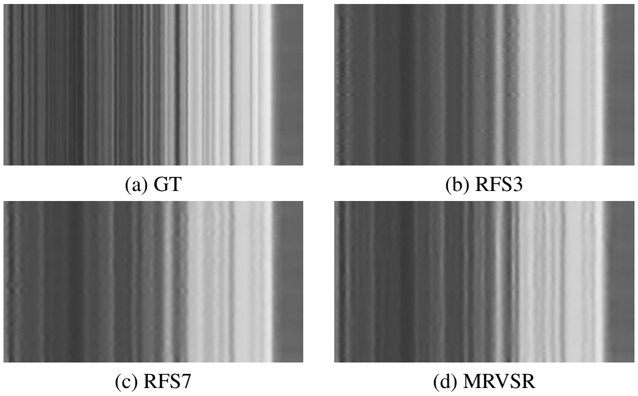


Figure 7. Spatio-temporal receptive fields of MRVSR (vizualization of juxtaposed images in the input sequence $X = (x_{-\tau}, ..., x_{\tau})$ optimized to maximize the L1 norm of the center pixel in the output image y_0). The horizontal axis accounts for the time index t of x_t. The figure is stretched in vertical direction.


(a) GT

(b) RFS3

(c) RFS7

(d) MRVSR

Figure 8. Temporal profiles from the Y channel of the first sequence of Quasi-Static Video Set. We take the 256th horizontal row of all images and stack them vertically.

As one could expect, MRVSR has practically the same computational complexity compared to RLSP (similar runtime and slight overhead in number of parameters, according to Tab. 2). As we stated in Sec. 2.1, RLSP is known to be the fastest VSR network so far. Therefore, MRVSR presents state-of-the-art runtime and compact model size.

6. Conclusion

In this work, we have pointed out the divergence problem of recurrent VSR when facing long sequences with low motion. Existing recurrent VSR networks generate high-frequency artifacts on such sequences. To solve this issue, we defined a new framework of recurrent VSR model, based on Lipschitz stability theory. As an implementation of this framework, we proposed a new recurrent VSR network coined MRVSR. We experimentally verified its stability and state-of-the-art performance on long sequences with low motion. As part of our experiments, we introduced a new test dataset of such sequences, namely Quasi-Static Video Set.

References

[1] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016. 2

[2] Sina Farsiu, M Dirk Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE transactions on image processing*, 13(10):1327–1344, 2004. 1, 5

[3] Dario Fuoli, Shuhang Gu, and Radu Timofte. Efficient video super-resolution through recurrent latent space propagation. In *ICCV Workshops*, 2019. 1, 2, 5

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 6

[5] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 3, 6

[6] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021. 2, 3

[7] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. Video super-resolution with recurrent structure-detail network. In *European Conference on Computer Vision*, pages 645–660. Springer, 2020. 1, 2, 5

[8] Takashi Isobe, Songjiang Li, Xu Jia, Shanxin Yuan, Gregory Slabaugh, Chunjing Xu, Ya-Li Li, Shengjin Wang, and Qi Tian. Video super-resolution with temporal group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8008–8017, 2020. 1

[9] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3224–3232, 2018. 1

[10] Cijo Jose, Moustapha Cissé, and Francois Fleuret. Kronecker recurrent units. In *International Conference on Machine Learning*, pages 2380–2389. PMLR, 2018. 2

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 6

[12] C. Liu and D. Sun. A bayesian approach to adaptive video super resolution. In *CVPR 2011*, pages 209–216, 2011. 1, 2, 5

[13] Xiaohong Liu, Lingshi Kong, Yang Zhou, Jiying Zhao, and Jun Chen. End-to-end trainable video super-resolution based on a new mechanism for implicit motion estimation and compensation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2416–2425, 2020. 1

[14] Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pages 2401–2409. PMLR, 2017. 2

[15] John Miller and Moritz Hardt. Stable recurrent models. In *International Conference on Learning Representations*, 2019. 2

[16] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 2

[17] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, June 2019. 5

[18] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013. 2

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[20] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6626–6634, 2018. 1, 2, 5

[21] Amartya Sanyal, Philip H. Torr, and Puneet K. Dokania. Stable rank normalization for improved generalization in neural networks and gans. In *International Conference on Learning Representations*, 2020. 3

[22] Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. In *International Conference on Learning Representations*, 2019. 2

[23] Thomas Tanay, Aivar Sootla, Matteo Maggioni, Puneet K Dokania, Philip Torr, Ales Leonardis, and Gregory Slabaugh. Diagnosing and preventing instabilities in recurrent video processing. *arXiv preprint arXiv:2010.05099*, 2020. 3, 6

[24] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 3

[25] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pages 3570–3578. PMLR, 2017. 2

[26] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1

[27] Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon,

and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 2

[28] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 5

[29] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3106–3115, 2019. 1

[30] Jiong Zhang, Qi Lei, and Inderjit Dhillon. Stabilizing gradients for deep neural networks via efficient svd parameterization. In *International Conference on Machine Learning*, pages 5806–5814. PMLR, 2018. 2