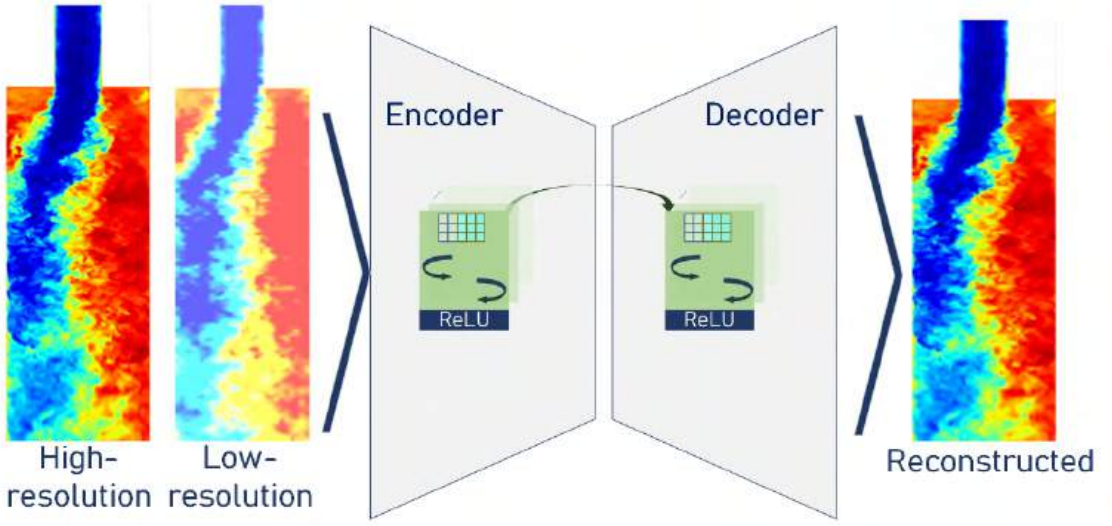
Graphical Abstract

Comparison of super-resolution deep learning models for flow imaging

Filippos Sofos, Dimitris Drikakis, Ioannis William Kokkinakis



Highlights

Comparison of super-resolution deep learning models for flow imaging

Filippos Sofos, Dimitris Drikakis, Ioannis William Kokkinakis

- A turbulent velocity flow field with instabilities
- Comparison of five deep learning models for flow image reconstruction
- Architectures based on convolutional neural networks
- Low-resolution images upscaled to high-resolution accuracy

Comparison of super-resolution deep learning models for flow imaging

Filippos Sofos[a,b], Dimitris Drikakis[b], Ioannis William Kokkinakis[b]

[a] *Condensed Matter Physics Laboratory, Department of Physics, University of Thessaly, Lamia, 35100, Greece*
[b] *Institute for Advanced Modeling and Simulation, University of Nicosia, Nicosia, CY-2417, Cyprus*

Abstract

The primary goal of this study is to introduce deep learning (DL) methods as a cost-effective alternative to the computationally intensive Direct Numerical Simulation (DNS) simulations. We show that one can obtain a parametric field from a low-resolution input and map it to a fine grid output, significantly reducing the computational burden. We assess five super-resolution models for up-scaling low-resolution flow data into fine-grid numerical simulations' output for accuracy and efficiency. The proposed architectures employ convolutional neural networks interconnected in encoder/decoder branches. We investigate these models using turbulent velocity fields inside a suddenly expanded channel characterized by complex features, including turbulence, instabilities, asymmetries, separation, and reattachment. Our results reveal that an encoder/decoder model with residual connections delivers the fastest results, a U-Net-based model with skip connections excels at producing sharper edges in regions prone to blurring, while deeper models incorporating maximum and average pooling layers show superior performance in reconstructing velocity profiles. These findings significantly contribute to our understanding of the

Email address: drikakis.d@unic.ac.cy (Dimitris Drikakis)

potential of deep learning in fluid mechanics. The models presented in this study are trained and validated on standard computer hardware and can be easily adapted to other problems. The findings are promising for discovering and analyzing flow physics, highlighting the potential for DL techniques to improve the accuracy of the available fluid mechanics computational tools.

Keywords: deep learning, super-resolution, flow reconstruction, turbulence

PACS: 47.27.nb, 84.35.+i, 47.80.Jk, 47.11.-j

2000 MSC: 68T45, 76F65, 94A08

---

1. Introduction

Simulating turbulent flows is a computationally expensive task and poses a challenge in many fields of science and engineering. Machine Learning (ML) models combined with high-fidelity flow data automated analysis [1, 2] can provide a predictive capability faster than expensive simulations, potentially leading to physics discovery and engineering optimization with significant impact on science, engineering, and medicine [3, 4]. The success of these models will depend on careful, objective verification and validation that will shed light on the accuracy, efficiency, and uncertainty regarding the models' predictions. Therefore, a comparative assessment between different models for complex flow data is required.

Classical ML models were proven to be insubstantial in raw data processing [5], such as the values of pixels in an image that correspond to quantities of interest. On the other hand, Deep Learning (DL) models have promoted learning through such representations by synthesizing nonlinear functions at various levels of abstraction [6] in a layer-wise fashion. Having started by mimicking biological models that stem from the human brain and its neural connectors [7], the proposed DL networks have shown their superiority in extracting hidden features from raw data [8]. This attribute has

further flourished in DL-based image processing, in tasks covering image retrieval, creation, analysis, and visualization [9].

Two-dimensional fluid flow data is well-suited for convolutional neural network (CNN) processing. Such DL networks employ the convolution operation to transform an image into layers of various dimensions under a fast-performing learning method [10]. Features of a CNN architecture utilize kernels that transform an image into a sequence of layers. In each imaging sequence, the first layers can capture image characteristics, such as object shapes and edges, while deeper CNN layers can recognize more abstract/complex features. Current CNN architectures are adjusted for all related computer vision applications [11, 12, 13, 14].

Super-resolution (SR) is a technique that improves the sparse grid field resolution, representing velocity, pressure, or temperature, among others. Deep learning can help establish nonlinear relationships between image data, even in noisy and sparse inputs.[15, 16] Still, this does not imply that DL can explain causation about data, and this is a topic of future research. Super-resolution implementations compare low- and high-resolution images, providing an enhanced reconstructed counterpart that approaches the fine input [17]. In addition to CNN architectures, SR has been implemented with generative adversarial networks (GANs) [18, 19, 20, 21, 22], physics-informed neural networks (PINNs) [23, 24], long-short-term memory (LSTM) networks [25, 26], graph neural networks (GNNs) [27], or, in combination of these techniques [28, 29, 30]. Their main differences are mainly focused on architecture characteristics, the depth of the network, the loss functions implied, and the learning method [31]. Of equal importance is the ability to extract meaningful representations from data [32].

SR models have given exceptional results during the past decade. Starting from the early approach of the SR convolutional neural network (SRCNN) [33], other pop-

ular architectures include the down-sampled skip-connection/multi-scale (DSC/MS) [34] and its updated model [35]. Moreover, the CNN-III multi-branch architecture [36] has performed remarkably in reconstructing temperature flow fields. The U-Net architecture is the essential component of many successful SR architectures, such as the recent multi-scale temporal path U-Net (MST-UNET) [28], the Deep Learning Flow Image (DELFI) [16], and the multi-level information compensation and U-net (MICU) [37]. A similar technique to SR, image inpainting, which is focused on reconstructing a damaged region inside an image, has been also implemented with DL architectures, such as the recent computer vision-oriented Deep Neural Networks and Attention Mechanism (DNNAM) [38] and the Multi-scale Feature Model and Attention Model (MFMAM) [39] colour black.

There is also increasing interest in more complex architectures that employ generative AI elements, such as GANs. The super-resolution GAN (SRGAN)[20] has been used to increase the resolution of various properties, while Wang et al. [40] have introduced the enhanced SRGAN, ESRGAN. The cycle-consistent GAN (CycleGAN) has shown excellent performance in reconstructing turbulent fields [41]. Recently, Bode and Göbbert [30] proposed the physics-informed ESRGAN (PIESRGAN) network as a subfilter model aiming to substitute DNS simulations.

All of the above are well-performing SR applications, from computer vision to fluid mechanics, with successful implementation. The motivation for this study stems from applying these techniques to internal flows, in general, and ventilation systems, in particular. The research here aims to enhance the resolution of flow data obtained from numerical simulations and experimental measurements using DL models. Various models were considered here with training images from a planar suddenly-expanded (PSE) turbulent channel flow, depicting bulk velocity fields [42]. Figure 1 illustrates the flow asymmetry that develops despite the geometric symmetry of the

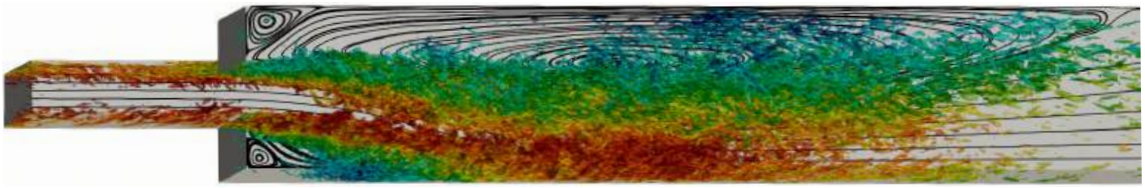


Figure 1: Iso-surfaces of the Q-criterion coloured by the streamwise velocity illustrate the turbulent flow's asymmetric expansion despite the symmetric corner. Background streamlines of the velocity illustrate the recirculation bubbles that form.

sudden expansion corner.

High-resolution flow data is crucial for accurate analysis and simulations in fluid mechanics, but generating such data is computationally intensive. Therefore, efficient methods are needed to upscale low-resolution data while maintaining computational efficiency and low memory demands, even on standard computer hardware. For this paper, the main contributions are as follows:

(1) The primary goal of this study is to introduce DL methods as a cost-effective alternative to the computationally intensive Direct Numerical Simulation (DNS) simulations. By using DL methods, we can obtain a parametric field from a low-resolution input and map it to a fine grid output, significantly reducing the computational burden.

(2) Moreover, as experimental sensor measurements for fluid mechanics correspond to a low-resolution field, this could also be upscaled by employing DL methods. More specifically, the SR technique is investigated for its ability to perform this mapping.

(3) Five well-established, base DL architectures, widely incorporated in the relevant literature, are constructed and compared on 2D simulation data obtained by the implicit Large Eddy Simulations (ILES) of subsonic flows through a symmetric suddenly expanded channel. The DL models employed provide excellent recon-

struction, as demonstrated through metrics such as the peak signal-to-noise ratio ($PSNR$), the root mean squared errors ($RMSE$), and the mean absolute error (MAE). Moreover, the local characteristics of the flow field are extracted, and the accuracy is further assessed using local error estimation.

(4) From the comparison made, it is concluded that an encoder/decoder model with residual connections delivers the fastest results, a U-Net-based model with skip connections excels at producing sharper edges in regions prone to blurring, while deeper models incorporating maximum and average pooling layers show superior performance in reconstructing velocity profiles. These findings significantly contribute to our understanding of the potential of deep learning in fluid mechanics.

(5) The models presented in this study are not only trained and validated on standard computer hardware but can also be easily adapted to a fluid mechanics computational platform. Importantly, no dimensionality reduction techniques or more complex DL network elements, such as GANs or PINNs, are required, so implementing these models is straightforward and practical.

The methods incorporated and the data curation procedure are described in Section 2. Section 3 presents the models and the results of the models' assessment in terms of accuracy and efficiency, while Section 4 summarizes the conclusions drawn from this study, discussing the challenges and providing significant future insights.

2. Scientific Computing Methods

2.1. Computational details

The basic component that applies to image reconstruction tasks is the convolutional layer. A CNN performs DL operations due to its ability to deal with big data

by following a selective image features processing, in contrast to common fully connected neural networks (FCNNs) that can be very computationally intensive. The mathematical expression for discrete convolution, given an image $\mathbf{X} \in \mathbb{R}^{W \times H}$ and a kernel $w \in \mathbb{R}^{M \times N}$ the resulting convoluted image is[43]

$$\mathbf{X_{conv}} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \mathbf{X}(m+i, n+j) w(i,j) \tag{1}$$

where $m = 0, .., M-1,\ n = 0, .., N-1$. The convolution operation affects the image size, and care has to be taken to obtain an output with the same dimensions as the input image. After applying Equation 1, $\mathbf{X_{conv}}$ is of shape $(W - M + 1) \times (H - N + 1)$. The computational method decides how the kernel slides over image pixels, i.e., the stride parameter, or the adding up of zeros around an image to ensure that the convoluted image retains the shape of the input image, i.e., the padding parameter. The models proposed here ensure that the CNN output image is of the same dimension as the input image by setting $stride = 1$ and $padding = 0$.

In addition to convolution layers, up-sampling layers can also be found in SR architectures. Most of the time, up-scaling is implemented with transposed convolution or deconvolution, which is the inverse operation of convolution, acting through a kernel that has gaps or holes, effectively increasing the spatial resolution of the image [44].f a convolution, acting through a kernel that has gaps or holes, effectively increasing the spatial resolution of the image [44]. The image dimensions may have been decreased inside the network without affecting spatial invariance by the pooling layers. These downsampling/pooling operations can be of different types, such as max pooling and average pooling (which has been investigated in this paper), as well as min pooling and sum pooling [45, 46]. MaxPooling slides as a kernel over the image pixels and passes the maximum values to the next step, while AveragePooling

keeps the average values for the next layer. For example, during a MaxPooling operation with a (2×2) kernel, only one maximum value continues to the next layer, and the resulting image is two times smaller in width and height. Another standard implementation inside the network that follows a convolution layer is the activation function, such as the ReLU function. It limits its output to $max(0, x)$ and allows dealing with nonlinear and more complex data.

The available image data set comprises 1221 turbulent velocity field images taken from accurate CFD simulations. From this pool, 970 images are employed for training and 251 for validation. The fine prototype images are considered the ground truth, upon which the models are compared for their obtained accuracy. However, the training is done with images of lower resolutions. The fine RGB images, which are of dimension $(W \times H \times 3) = (2296 \times 325 \times 3)$, are down-scaled in their dimensions by a factor of 4, giving images of dimension $(W \times H \times 3) = (574 \times 81 \times 3)$ (from now on, as LR4), and they are linearly interpolated to their initial dimensions again [47], constituting blurred images of lower resolution. For validation reasons, we have also employed low-resolution images of dimensions $(W \times H \times 3) = (382 \times 54 \times 3)$, referred to as LR6, and $(W \times H \times 3) = (287 \times 40 \times 3)$, referred as LR8.

Moreover, their pixels are normalized in the range of $0 - 1$ to achieve better training and remove possible biases, which is standard practice in neural network operations. Here, we also consider training iterations (epochs) controlled by the early stopping method [48], which forces the algorithm to stop training when the model performance is not improving any more, which is a technique that helps to prevent overfitting due to many training steps on a specific dataset.

2.2. Data curation

This paper uses the high-fidelity results obtained by the implicit Large Eddy Simulations (ILES) of subsonic flows through a symmetric suddenly expanded channel carried out by Karantonis et al. [42]. The case corresponds to a relatively high Reynolds number of $10,000$, based on the inlet bulk velocity and the step height of the channel (h) to investigate the structure of the resulting turbulence. Based on the bulk velocity at the channel inlet, the Mach number is 0.1, which is reduced by one-third after expansion.

Figure 2 shows a schematic of the sudden expansion, which comprises two channels, each of a different height. The flow domain consists of an inlet channel of height h and a downstream channel of height $3h$. The characteristic length of the channel is the step height, h, with a value of 1. The total length of the domain is $84h$, where the inlet channel has a length of $4h$, and the downstream channel has a size of $80h$. These particular geometrical features were chosen to ensure that the flow (a) is fully developed (turbulent) before reaching the expansion step and (b) that the buffer layer at the end of the domain dampens unsteadiness in the flow before exiting the domain. The expansion ratio, important when simulating suddenly-expanded flows, equals 3, having the same value as that of Casarsa and Giannattasio [49]. The aspect ratio ($AR = w/h$) considered is 5.

The block-structured grid code CNS3D solves the Navier-Stokes equations using the finite-volume method (FVM). CNS3D can be used for implicit Large Eddy Simulations (ILES) and DNS. The advective terms are solved using the Godunov-type (upwind) method, whose inter-cell numerical fluxes are calculated by solving the Riemann problem using the reconstructed values of the primitive variables at the cell interfaces. A one-dimensional swept unidirectional stencil is used for the spatial reconstruction. The numerical simulations herein were obtained using an augmented

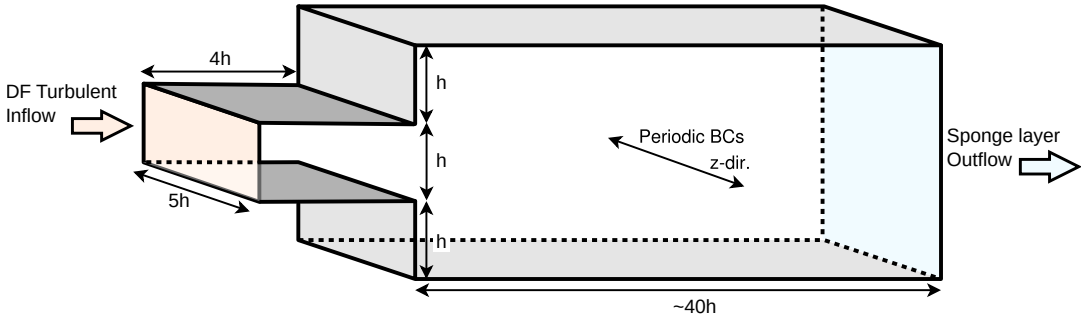


Figure 2: Schematic diagram of the planar sudden expansion (PSE) configuration.

11th-order WENO scheme [50, 51] for interpolation, in conjunction with the approximate HLLC Riemann solver [52]. The viscous terms are solved using a second-order central scheme. The solution is advanced in time using an explicit five-stage (fourth-order accurate) optimal Strong Stability preserving (SSP) Runge-Kutta method [53]. Further details of the numerical aspects of the code can be found in past literature [54, 51] and references therein.

A synthetic turbulent boundary condition based on the digital filter (DF) technique originally proposed by Klein et al. [55] and developed further by Touber and Sandham [56], was employed at the inlet. The DF approach produces a velocity signal in three directions by matching ad hoc first- and second-order statistical moments, length and time scales, and energy spectra. We selected the DF approach for generating artificial inflow data as the filtering operation applied only in 2D (rather than 3D), making the whole process much faster and computationally efficient.

A buffer layer is employed at the outflow to avoid any numerical reflections. The boundary surfaces in the wall-normal (y) direction were assigned standard no-slip, viscous, and adiabatic wall boundary conditions. Periodic conditions were chosen for the boundary surfaces normal to the spanwise (z) direction.

In this study, CNS3D has been used to produce data in the framework of wall-

resolved ILES. The present mesh spacing (Δy) is scaled using the conventional inner variable method $\Delta y^+ = u_\tau \Delta y/\nu_w$, where $u_\tau = \sqrt{\tau_w/\rho_w}$ is the friction velocity, ν_w, τ_w and ρ_w are the near wall kinematic viscosity, wall shear stress, and density, respectively.

The grid resolution at the channel inlet in the streamwise, wall-normal and spanwise directions is $\Delta x^+ \simeq 26$, $y_w^+ = 1$ (corresponding to $y^+ = 0.5$ in the first cell center off the wall) and $\Delta z^+ \simeq 15$, respectively. At the expansion corner, $\Delta x^+ \approx y_w^+$. It then gradually coarsens towards the exit boundary, reaching approximately a value of $\Delta x^+ \simeq 50$ at the expected $L_1 \simeq 14.4h$ (location of upper recirculation layer reattachment). After that, it increases linearly towards the outlet, eventually reaching a value of $\Delta x^+ \simeq 180$. In total, the computational domain was discretized by 34,810,000 cells. The grid resolution and numerical scheme yielded accurate results [54] for wall-bounded turbulent flows at low Mach numbers. The present fine mesh corresponds to wall-resolved ILES following typical resolution recommendations for LES and DNS simulations [57, 58, 59].

In [42], the total simulation time was $300h/U_B$, from which statistics were obtained over the last $100h/U_B$, where U_B is the bulk velocity of the channel inlet and h its height. Over the statistics time, 407 images of the 2D surface contour plot of the streamwise velocity are extracted at a constant time interval. The aforementioned is performed in three equidistant z-normal (xy) planes, specifically $z/h = 1.0$, 2.5, and 4.5, to enhance the image sample size further. Thus, a total of 1,221 images are available for the present study.

Table 1: Details of the models incorporated for SR investigation.

Model	Description
U-SIMPLE	Network layers forming a UNET architecture in its simplest form, without skip connections [61, 62]
ED-RES	Encoder/decoder model with one residual connection adjusted from [63]
U-SKIP	A UNET-type architecture with one skip connection [61, 62]
SR-MAX	SR architecture with max-pooling layers, without the internal fully connected layer [34]
SR-AVE	SR architecture with average pooling layers without the internal fully connected layer [34]

3. Results and discussion

3.1. Super resolution models

Five different models have been constructed in TensorFlow [60] for flow image reconstruction, which scales over architecture complexity and applicability, as shown in Table 1. These are (a) a more straightforward instance of the UNET model (U-SIMPLE), (b) an encoder/decoder scheme with one residual connection (ED-RES), (c) a UNET-type with one skip connection (U-SKIP), and two similar SR architectures with different pooling layers, i.e., (d) with max pooling (SR-MAX), and (e) with average pooling (SR-AVE). All these architectures have been adjusted to the implied image dimensions, while a minimum number of network layers has been selected to keep them simple and fast.

All these models are based on well-established architectures proposed for image reconstruction tasks in the literature, not only for computer vision but also for fluid mechanics applications. They are trained by connecting the LR input to its HR counterpart. The SR model can draw a mapping function between LR and HR images when the training pool is large. A reconstructed image is, thus, produced when features from the LR image are extracted and upsampled to HR features. At

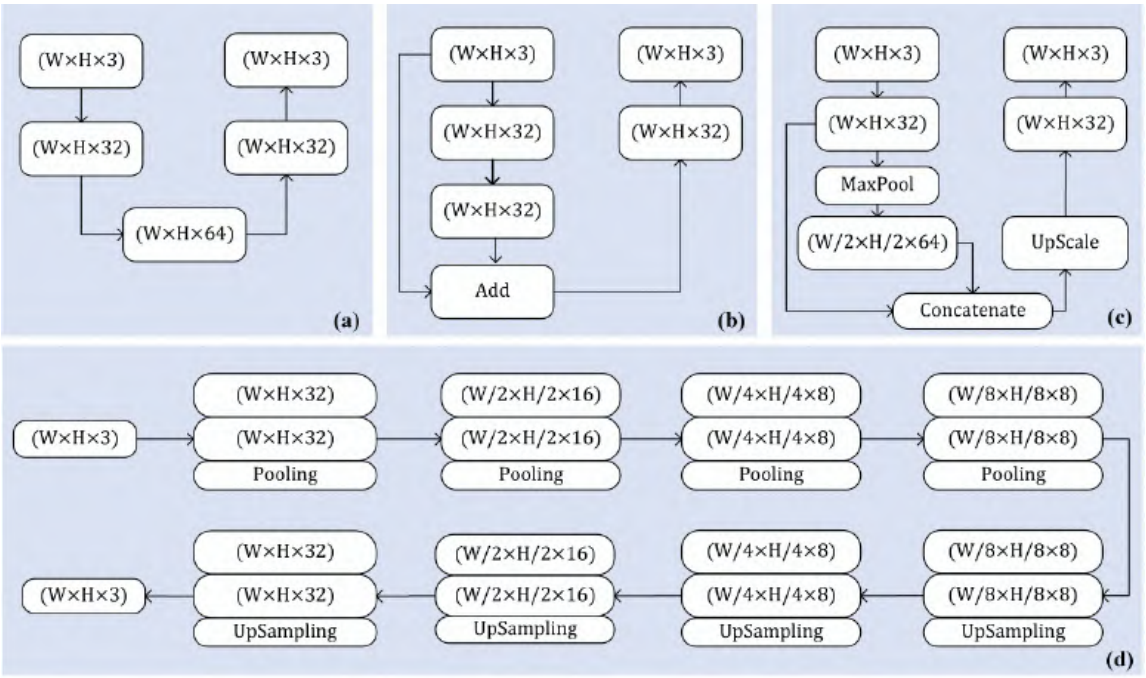


Figure 3: Models implemented for flow image SR, (a) U-SIMPLE, (b) ED-RES, (c) U-SKIP, and (d) SR-MAX (when Pooling=MaxPooling) and SR-AVE (when Pooling=AveragePooling).

this point, the reconstructed image is compared to the ground truth (the HR). The model is assessed by minimising the loss function between the two images.

More specifically, to obtain an HR flow field $\mathbf{X_{HR}} \in \mathbb{R}^m$ from a LR field $\mathbf{X_{LR}} \in \mathbb{R}^n$, where $n < m$, the model has to learn the relation giving $\mathbf{X_{LR}} \rightarrow \mathbf{X_{HR}}$. In reconstruction problems, a field $\mathbf{X_{REC}} \in \mathbb{R}^r$ is proposed, where the desired result is obtained as $r \rightarrow n$. The model is trained by image couples of $(\mathbf{X_{HR}}^i, \mathbf{X_{LR}}^i)$, with i images, and suggests a mapping function $\mathcal{F} : \mathbf{X_{LR}} \rightarrow \mathbf{X_{REC}}$, such that $\mathbf{X_{REC}} = \mathcal{F}(\mathbf{X_{LR}})$.

As for the architectures implied, the U-NET expands in two branches that consist of a U-type model [61]. The input images enter the left branch and pass through a contracting CNN network. The right branch follows with an expansive CNN network

until the reconstructed output is extracted. A similar implementation is our U-SIMPLE model, in Fig. 3(a). The algorithm behind U-SIMPLE is given in pseudo-code in Algorithm 1. In cases where high-resolution reconstruction is the question, an additional step would be the incorporation of sending context information flow from the left to the right branch, such as in the U-SKIP model, in Fig. 3(c) and in Algorithm 3. This model appends image dimension decrease on the left with a max pooling layer and an up-sampling layer on the right. This decreased dimension process leads to information loss, which the skip connection anticipates.

The encoder/decoder model with residual connections (ED-RES), shown in Fig. 3(b), incorporates a connection of the input (where all image information is available) to an internal CNN layer. In the original paper found in the literature, the ResNet architecture is more complex but better suited for demanding computer vision applications [63]. Here, a more straightforward implementation is chosen, which is fast and accurate simultaneously. The pseudo-code is shown in Algorithm 2.

The SR model proposed by Fukami et al. [34] has been adjusted in our case in Fig. 3(d), with two different instances, one with MaxPooling layers (SR-MAX) and one with AveragePooling layers (SR-AVE). This architecture involves multiple CNN operations and four pooling layers in the input branch, significantly decreasing the input dimensions and enhancing the processing speed. Conversely, the output branch includes four upsampling layers that reconstruct the processed images into one output, with dimensions equal to the input. In similar models [35], a multi-layer perceptron intervenes between the input and the output branch. Here, without losing accuracy, we have omitted this intermediate step due to excess memory and computational time demands, which would make it impossible to run on standard hardware. The pseudo-code for these two models is shown in Algorithms 4-5. The only difference is the Pooling layer incorporated in the encoder branch.

Algorithm 1 U-SIMPLE

```
Input: Coarse input image
input ← Input
cnn_1 ← Conv2D(32, (3, 3),'relu',padding='same')(input)
cnn_mid ← Conv2D(64, (3, 3),'relu',padding='same')(cnn_1)
cnn_up1 ← Conv2D(32, (3, 3),'relu',padding='same')(cnn_mid)
output ← Conv2D(3, (3, 3),'relu',padding='same')(cnn_up1)
model ← Model(inputs=[input], outputs=[output])
return model
```

Algorithm 2 ED-RES

```
Input: Coarse input image
input ← Input
cnn_1 ← Conv2D(32, (3, 3),'relu',padding='same')(input)
cnn_2 ← Conv2D(32, (3, 3),'relu',padding='same')(cnn_1)
Res ← Conv2D(32, (1, 1),'relu',padding='same')(input)
Add ← Add()([Res,cnn_2])
cnn_3 ← Conv2DTranspose(32, (3, 3),'relu',padding='same')(Add)
output ← Conv2DTranspose(3, (3, 3),'relu',padding='same')(cnn_3)
model ← Model(inputs=input, outputs=output)
return model
```

3.2. Reconstruction accuracy

The five different models are compared based on various computational and accuracy metrics, such as the total training time, the epochs (iterations) needed for each model to converge, and the peak signal-to-noise ratio ($PSNR$) values obtained when the models are validated with images of various resolutions (Fig. 4). The software implementing the models is executed on a 40-core CPU with 120 GB of memory due to the size of the input image files and the information flow between CNN layers. Still, it can be easily modified to run on standard computer hardware by pipelining the input images during training. This technique is based on transfer learning, where a small set of images is initially used for training, and a consecutive set of images is fed iteratively for training, thus increasing the accuracy in this stepping

Algorithm 3 U-SKIP

```
Input: Coarse input image
input ← Input
cnn_1 ← Conv2D(32, (3, 3),'relu', padding='same')(input)
pool_1 ← MaxPooling2D((2, 2))(cnn_1)
cnn_mid ← Conv2D(64, (3, 3),'relu', padding='same')(pool_1)
cnn_up1 ← Conv2D(32, (3, 3),'relu', padding='same')(cnn_mid)
up1 ← UpSampling2D((2, 2))(cnn_up1)
output ← Conv2D(3, (3, 3),'relu', padding='same')(up1)
model ← Model(inputs=[input], outputs=[output])
return model
```

manner [64]. It is emphasized that the main focus of this investigation is to keep the proposed models as simple and practical as possible so that they can be tested on standard hardware and provide results faster without the need to upscale on a High-Performance Computer (HPC).

Regarding the required time for training, the ED-RES model is faster, followed by the U-SIMPLE, as shown in Fig. 4(a). These are the two simpler architectures with CNN layers that do not alter the image dimensions. Furthermore, they only need 24 and 26 epochs, respectively, to converge (Fig. 4(b)). In contrast, the two SR-type models (SR-MAX and SR-AVE) are more than twice slower than ED-RES and converge in 36 epochs. The U-SKIP model has been even slower to converge in 43 epochs, requiring more than 20 hours to run. This is attributed to the more complex and layered architecture, which performs a broader range of computations inside the network. Nevertheless, it is observed that all models achieve the same $PSNR$ values, as seen in Fig. 4(c), with ED-RES reaching the highest value in this range. Furthermore, high $PSNR$ values are obtained for all models when the input image set used for validation is of the exact resolution as the training image set, i.e., the input images are of LR4 resolution. Lower $PSNR$ values, suggestive of a poorer

Algorithm 4 SR-MAX

```
Input: Coarse input image
input ← Input
Encoder:
x ← Conv2D(32, (3, 3),'relu', padding='same')(input)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← MaxPooling2D((2, 2), padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← MaxPooling2D((2, 2), padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← MaxPooling2D((2, 2), padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← MaxPooling2D((2, 2), padding='same')(x)
Decoder:
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
decoded ← Conv2D(3, (3, 3),'relu', padding='same')(x)
model ← Model(inputs=input, outputs=decoded)
return model
```

reconstruction, are obtained for LR6 validation, and even poorer are obtained for LR8 resolution.

On the other hand, the calculated total root mean squared error $RMSE$, aver-

Algorithm 5 SR-AVE

```
Input: Coarse input image
input ← Input
Encoder:
x ← Conv2D(32, (3, 3),'relu', padding='same')(input)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← AveragePooling2D((2, 2), padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← AveragePooling2D((2, 2), padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← AveragePooling2D((2, 2), padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← AveragePooling2D((2, 2), padding='same')(x)
Decoder:
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← Conv2D(8, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← Conv2D(16, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← Conv2D(32, (3, 3),'relu', padding='same')(x)
x ← UpSampling2D((2, 2))(x)
decoded ← Conv2D(3, (3, 3),'relu', padding='same')(x)
model ← Model(inputs=input, outputs=decoded)
return model
```

aged across the entire image validation set (Fig. 4(d)), is smaller for the SR-MAX and SR-AVE models. Of importance is also the fact that these $RMSE$ values remain low for all the different resolution inputs the models are asked to reconstruct. The $RMSE$ considers all errors equally, without emphasizing specific frequency components, while the $PSNR$ depends on image frequency characteristics. The ideal

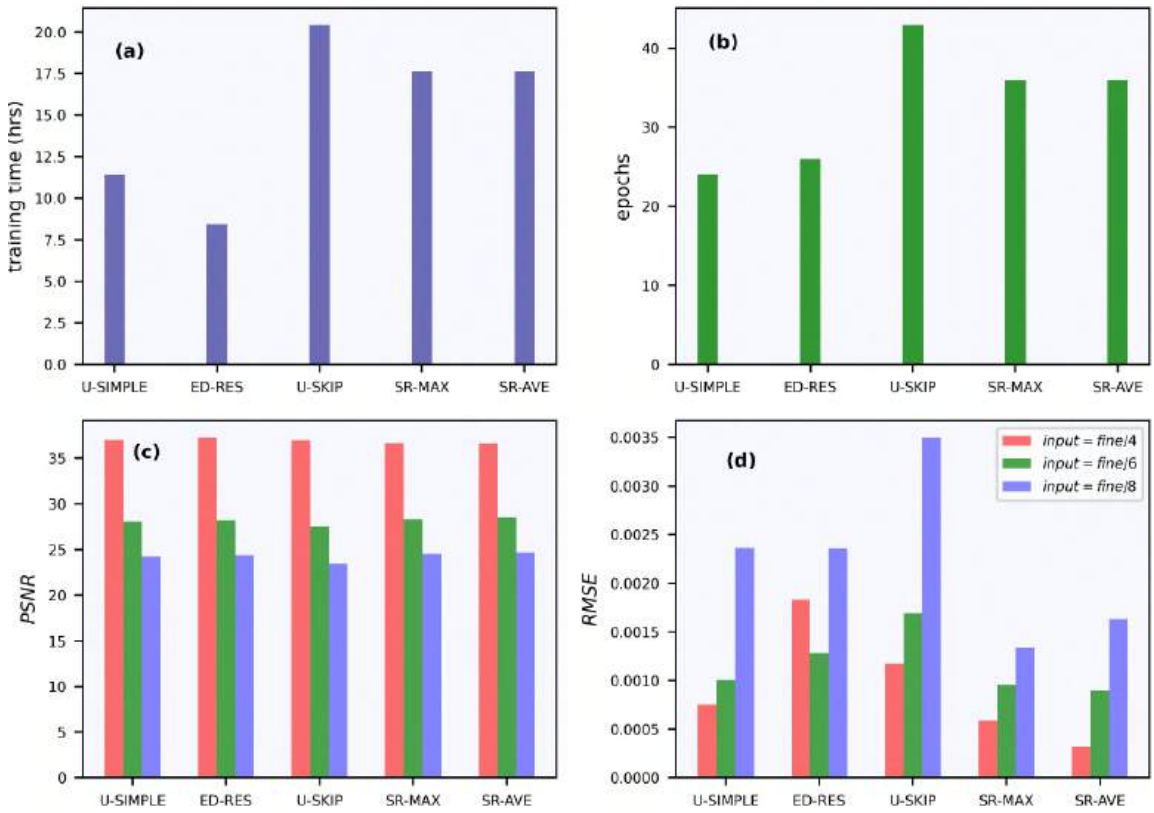


Figure 4: Per model analysis, (a) time needed for training the network, (b) epochs until convergence, after imposing early stopping method, (c) calculated $PSNR$, and (d) $RMSE$, as average values for the whole image validation set, for three different image resolutions.

scenario would be a high $PSNR$ and low $RMSE$ model. However, there are cases where high $PSNR$ does not necessarily connect with low $RMSE$ [65]. Such behavior differences can arise due to the different characteristics and sensitivities of the two metrics. $PSNR$ and $RMSE$ are not interchangeable, and their interpretation depends on the specific characteristics of the compared images.

The resulting loss on training and validation samples is depicted in Fig. 5. As can be seen, there are no significant differences between training and validation loss, which is evidence of no overfitting during this process. Training and validation loss remain similar after a few epochs at the beginning of training. When training

starts, the more complex U-SKIP model has larger loss values, while ED-RES has the smallest. However, it takes only 1-2 epochs for all models to converge.

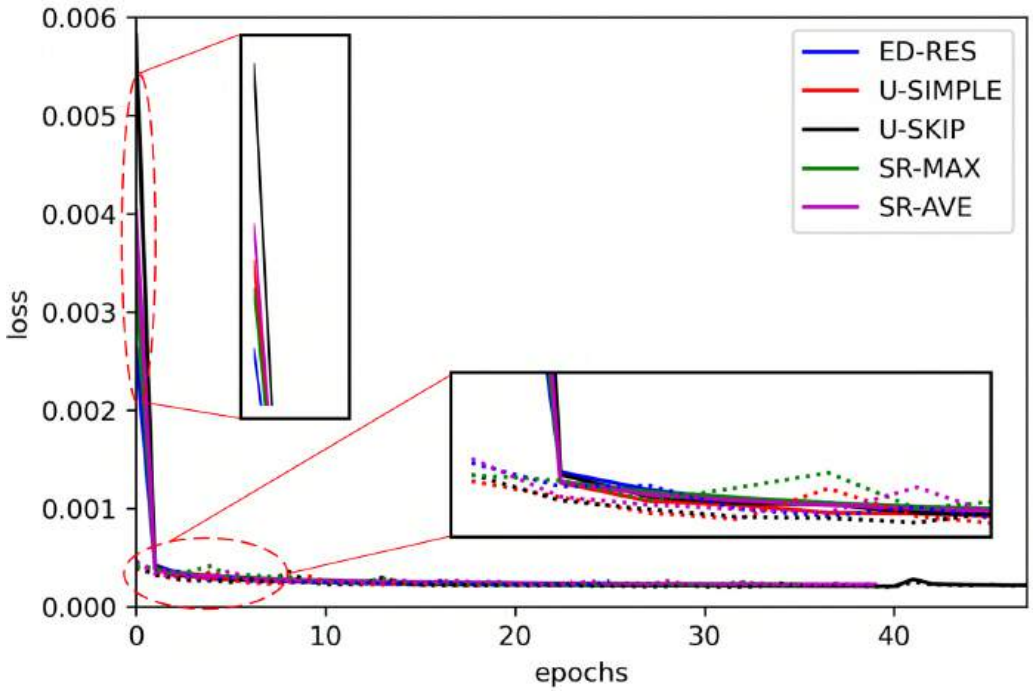


Figure 5: Loss diagrams for both training and validation images. All diagrams converge to a minimum value in less than 46 epochs. Straight lines are the training, and dotted lines are the validation losses. The insets focus on diagram details.

3.3. Local field investigation

An example of an image used for validation and the reconstructed result from the five different reconstruction models are presented in Fig. 6. For presentation reasons, we mark five cross-section areas in the input image (Fig. 6(a)) and assess the accuracy result of each model. The LR4, blurred image and ground truth images are shown in Fig. 6(b-c), respectively, for comparison. Smaller regions inside the cross sections (with dotted lines) are further spotted to spot the reconstruction accuracy. In general view, it is observed that all five models, in Figs. 6(d-h), have given satisfying image

reconstruction, removing the blurred pixels and capturing spatial details. Taking a closer look at the dotted circles in each cross-section, it seems that the U-SKIP model has removed blurring by achieving, in parallel, sharper edges on the specific image elements, resembling the HR image. This result is more pronounced near "blue" edges, such as in the circles of cross-sections 1, 3 and 4, which correspond to regions of lower frequency.

The models' accuracy is further evaluated on more blurred inputs. In Fig. 7, an LR6-type input enters the reconstruction process in all models. All models are found to enhance the LR6 input, though with lower resolution compared to the LR4 case shown in Fig. 6. More specifically, inside the dotted circle of cross-section 1, where a red point is located (see the HR Fig. 6(c)), no model has managed to recover it in the reconstructed output. Deblurring the fluid edges inside the dotted circles of cross-sections 2,3, 4, and 5 is satisfactorily achieved by all models.

All the proposed models can reconstruct turbulence flow image characteristics, even when the image input is four or six times smaller than the ground truth image. Moreover, in these encoder/decoder schemes, passing context information from the encoder to the decoder through skip connections results in better network training and, finally, better reconstruction accuracy.

3.4. Velocity profiles

One of the advantages of DL in image processing is the ability to process pixel intensity values and connect the images to a physical variable. Toward this direction, the pixel values (which correspond to the velocity) are averaged in each R, G, and B layer and each cross-section (see Fig. 6(a)). At the same time, metrics such as the coefficient of determination, R^2, MAE and RMSE are calculated. Velocity profiles are timed over the entire validation set for (i) the HR (y_{HR}), (ii) the low-resolution

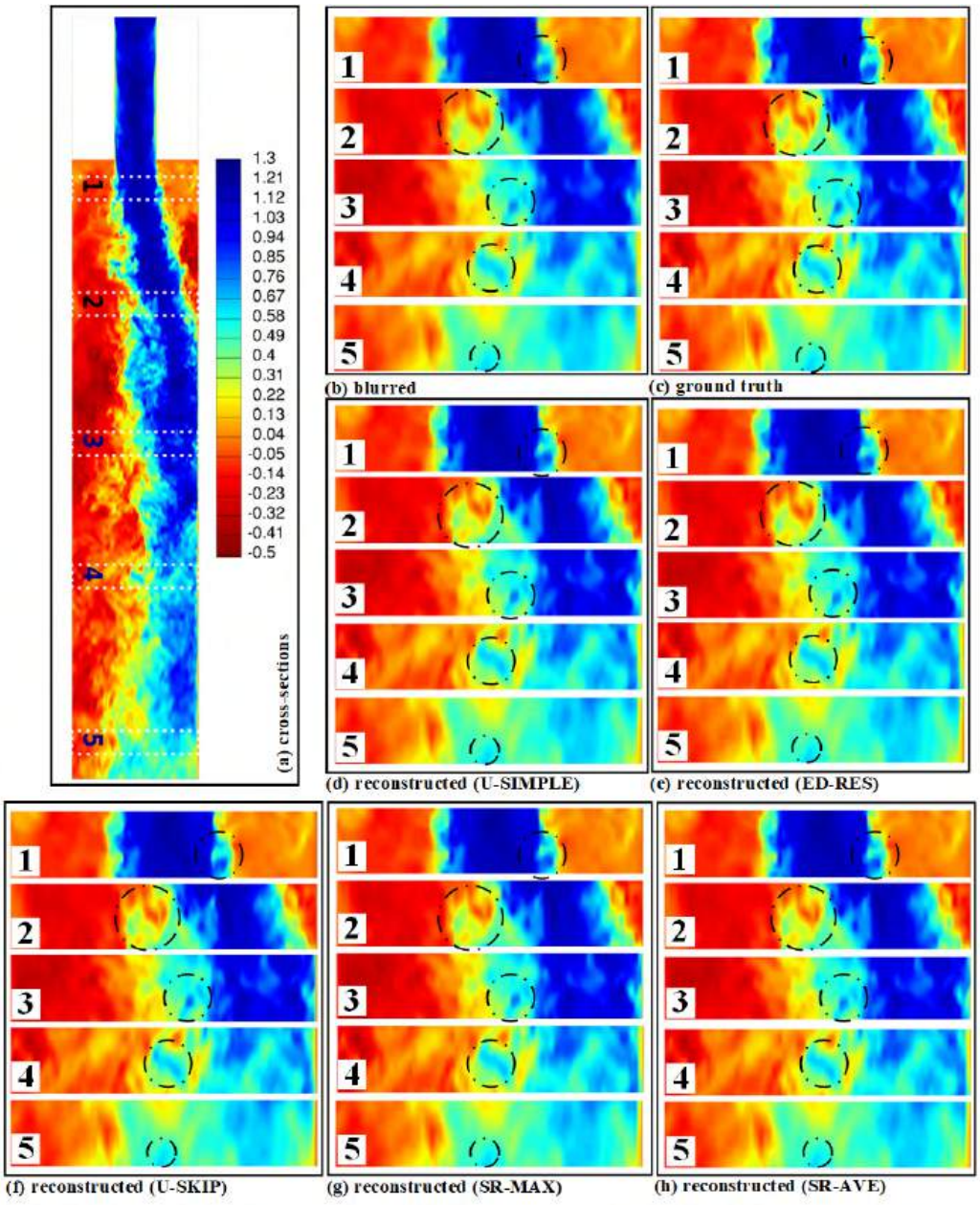


Figure 6: Image reconstruction outcome for an LR4 input, for all models investigated. (a) One sample (HR) input image; five cross sections (1-5) are defined to discuss reconstruction accuracy, (b) LR4 image regions that enter each model for reconstruction, (c) respective ground truth (HR) image, and the reconstructed regions after applying the: (d) U-SIMPLE model, (e) ED-REs model, (f) U-SKIP model, (g) SR-MAX, and (h) SR-AVE models. Dotted circles denote regions where differences between the blurred, fine, and reconstructed counterparts can be better understood.

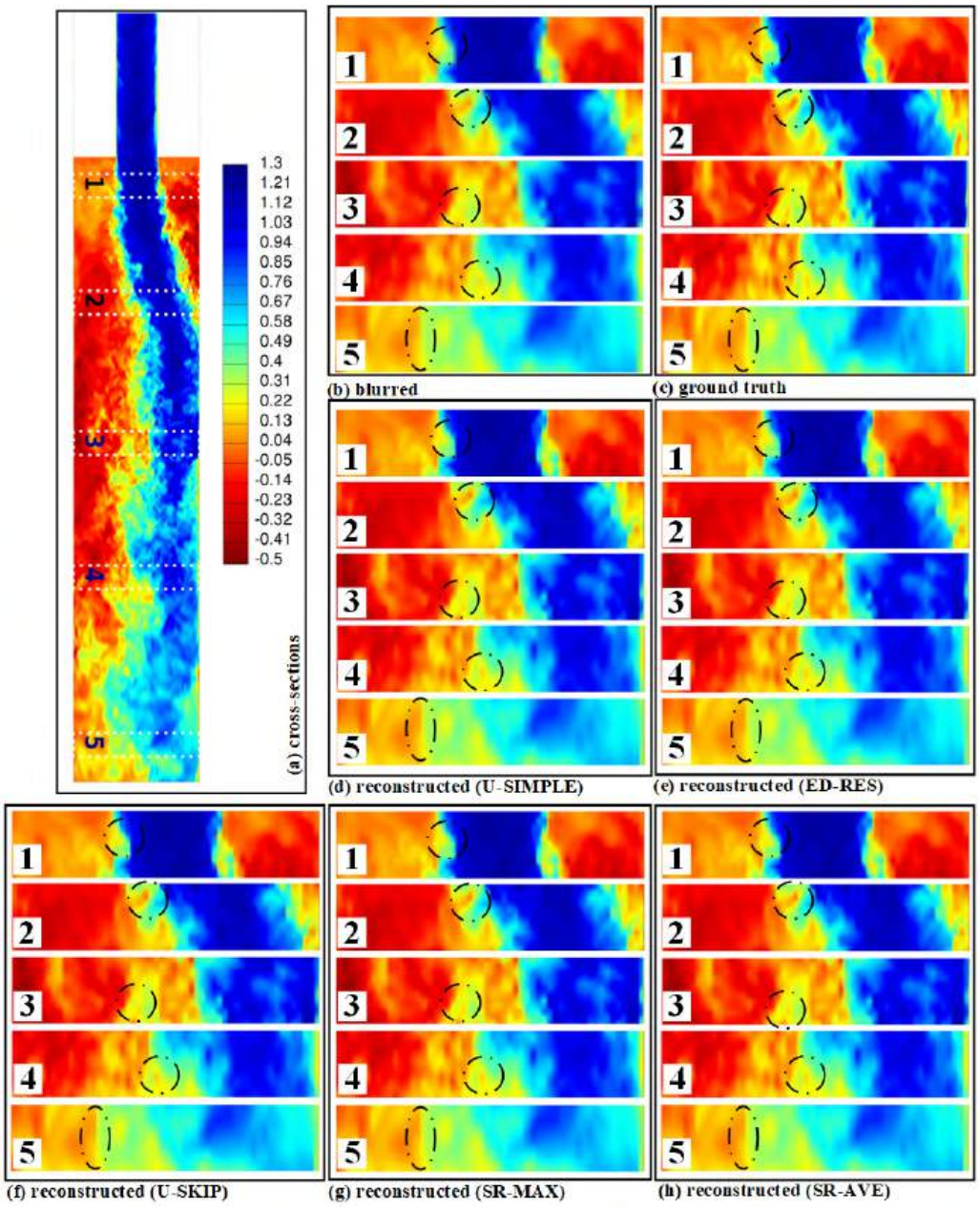


Figure 7: Image reconstruction outcome for an LR6 input, for all models investigated. (a) One sample (HR) input image; five cross sections (1-5) are defined to argue on reconstruction accuracy, (b) LR6 image regions that enter each model for reconstruction, (c) respective ground truth (HR) image, and the reconstructed regions after applying the: (d) U-SIMPLE model, (e) ED-REs model, (f) U-SKIP model, (g) SR-MAX, and (h) SR-AVE models. Dotted circles denote regions where differences between the blurred, fine, and reconstructed counterparts can be better understood.

counterpart (y_{LR}), and (iii) their respective reconstructed profiles (y_{rec}), separately. Next, the velocity profile from region 1 is selected, and the results for every model and every image resolution considered are shown.

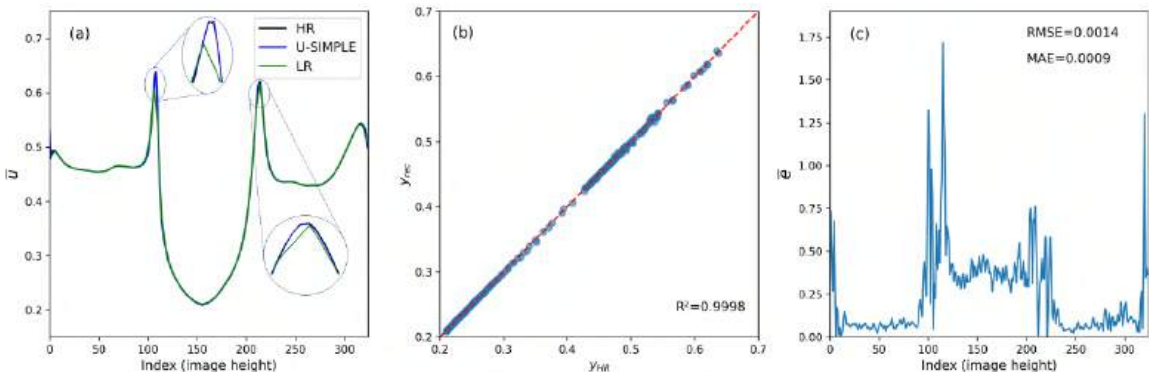


Figure 8: U-SIMPLE: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

In Figure 8(a), the reconstructed profile refers to the U-SIMPLE model and LR4 resolution, and it seems that there is a good reconstruction result, as the average velocity profile is practically identical for the y_{HR} and the y_{rec} images. It should be noted, though, that image degradation in the LR4 case is minimal, so the blurred image profile y_{LR} also has a good similarity to the respective HR profile. The identity plot in Fig. 8(b) has given $R^2 \to 1$, validating the excellent reconstruction output. The relative error between y_{HR} and y_{rec}, in Figure 8(c), is small but significant at two regions inside the channel. At first, the areas where the fluid is attached to the solid boundary have a higher reconstruction error, which can be attributed to statistical and numerical discontinuities since velocity equals zero at the boundary. On the other hand, the two high-error in-channel points are those where velocity values present abrupt changes. The fluid jet (i.e., the blue jet in Region 1, see Fig. 6(a)) that enters the contracted channel with high velocity meets a low-velocity region

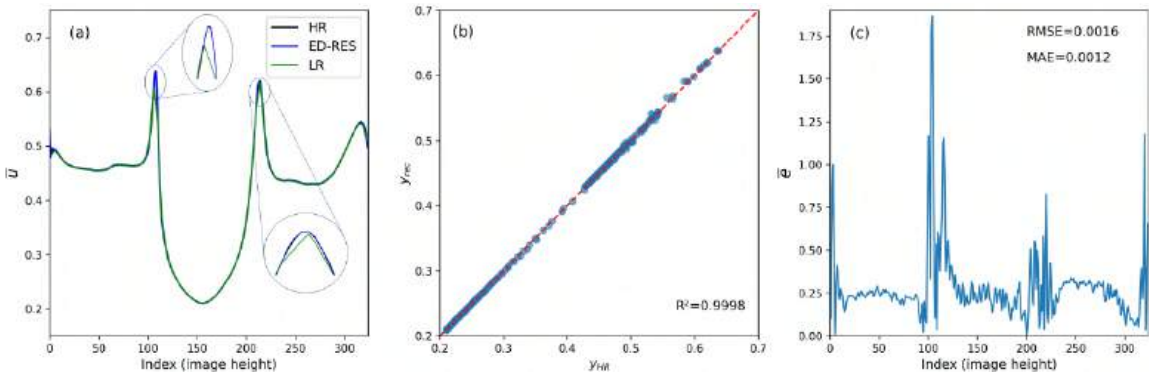


Figure 9: ED-RES: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}/|y_{HR}$, with MAE and RMSE shown.

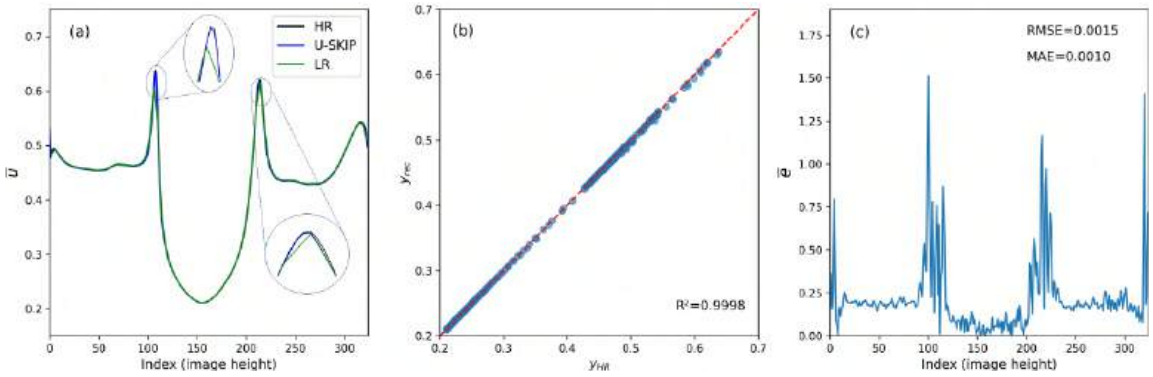


Figure 10: U-SKIP: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

(i.e., the orange part in Region 1). In all the remaining points in the velocity profile, there is a perfect match between y_{HR} and y_{rec}, with MAE and RMSE remaining close to zero.

Reconstruction metrics on the LR4 dataset and the ED-RES model are shown in Fig. 9. There is no significant difference between the U-SIMPLE and the ED-RES model. However, as seen in Fig. 9(c), both RMSE and MAE increase compared to

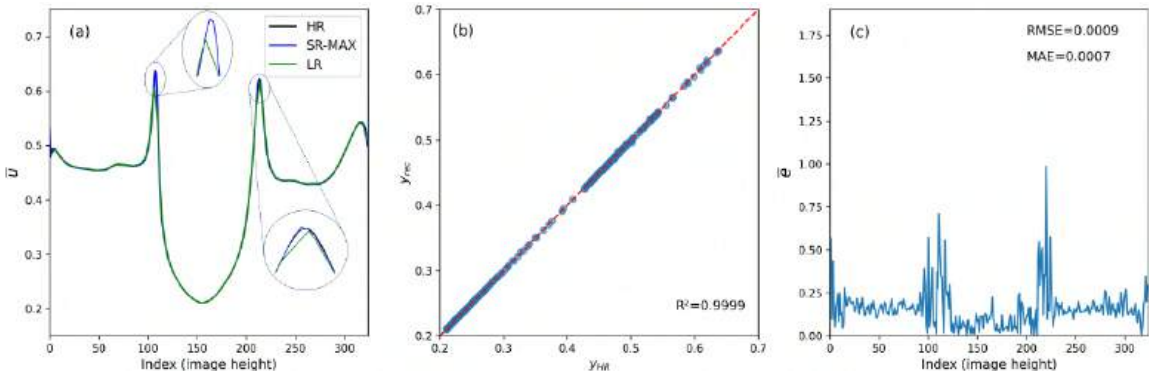


Figure 11: SR-MAX: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

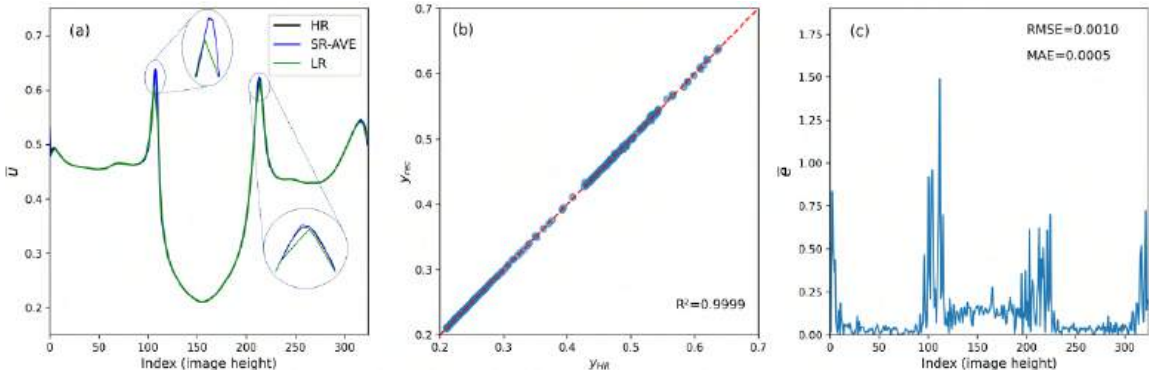


Figure 12: SR-AVE: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

Fig. 8(c). At the same time, the relative error $\overline{e}$ is greater than the U-SIMPLE model in regions where the velocity does not undergo abrupt changes. Similar behaviour is observed by the U-SKIP model as far as the profile and R^2 are concerned in Figs. 10(a-b), respectively. Nonetheless, the relative error in Fig. 10(c) is close to zero in the middle of the channel, where $u = u_{max}$. However, it should be noted that the SR-MAX and SR-AVE models, in Figs. 11 and 12, respectively, have given the

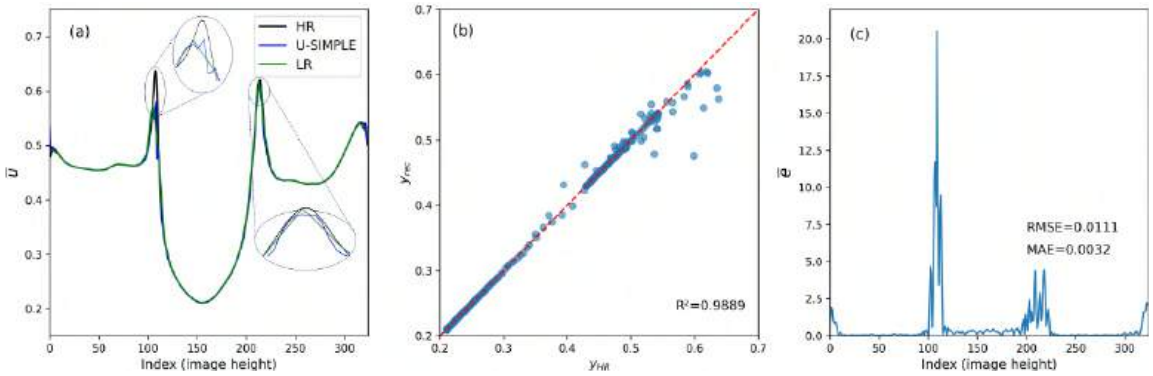


Figure 13: U-SIMPLE: Metrics and average velocity profiles reconstruction for the LR6 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

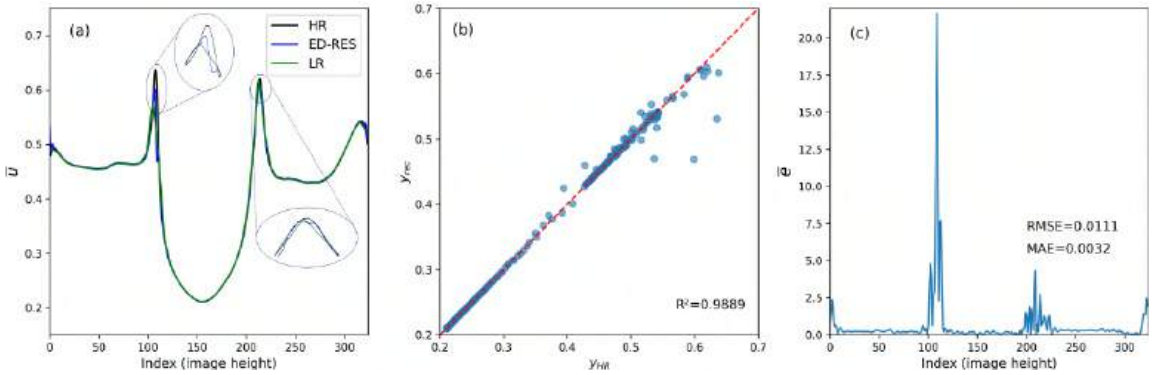


Figure 14: ED-RES: Metrics and average velocity profiles reconstruction for the LR6 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

smallest errors, and as a result the best reconstruction for the velocity values.

Reconstruction metrics degrade when the input image set has a lower resolution (LR6). Figures 13-17 present the reconstructed profiles and the respective error metrics for all of the models investigated. Maximum errors are observed in the channel interior in the regions where velocity values have abrupt changes. Here, the error reaches close to 20%, which is significantly higher than the LR cases, where

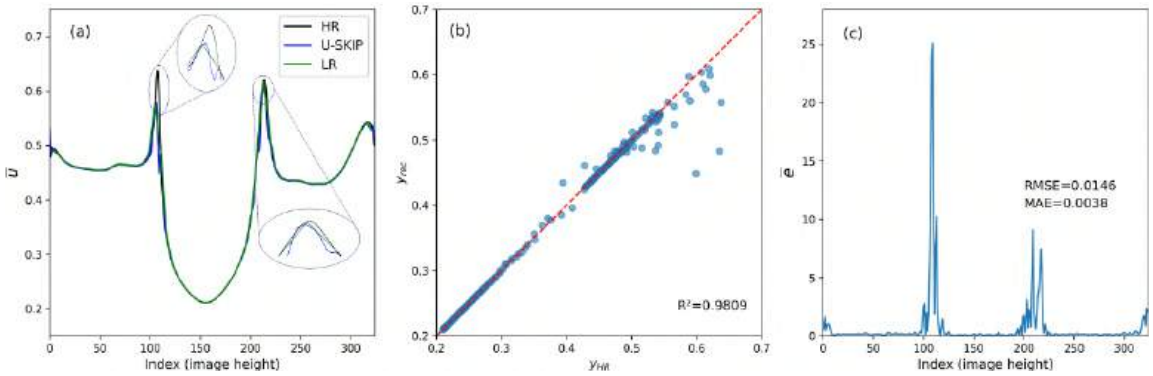


Figure 15: U-SKIP: Metrics and average velocity profiles reconstruction for the LR6 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

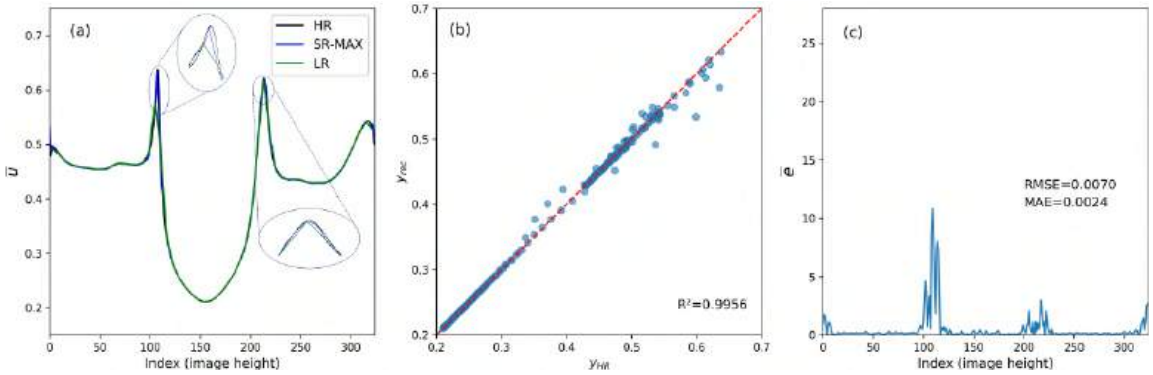


Figure 16: SR-MAX: Metrics and average velocity profiles reconstruction for the LR6 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

the maximum error has been kept less than 2% (see Figures 8-12). The coefficient of determination, R^2, has also decreased, but as an average value, it is still satisfactory since $R^2 > 0.98$. Furthermore, in the LR6 reconstruction task, the SR-MAX and SR-AVE architectures performed better than U-SIMPLE and ED-RES, while U-SKIP gave higher MAE and RMSE values.

We have also extracted velocity profiles for regions 2-5 (see Figures 6 and 7) for all

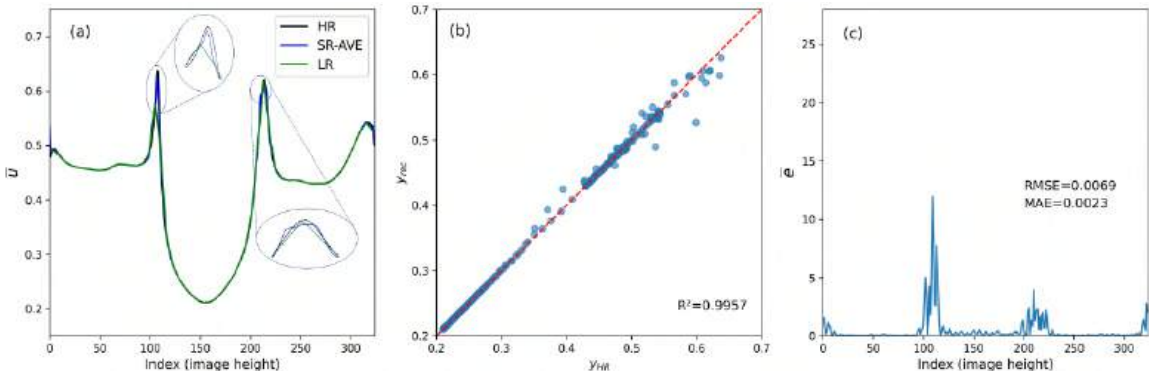


Figure 17: SR-AVE: Metrics and average velocity profiles reconstruction for the LR6 validation dataset. (a) Velocity profile, with insets in the regions of largest error value, (b) identity plot (ground truth vs reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = |y_{HR} - y_{rec}|/y_{HR}$, with MAE and RMSE shown.

models and all input image resolutions (LR4-LR6). These profiles present smoother velocity profiles without abrupt changes. They can be found in the Appendix, in Figures A1-A40.

4. Conclusions

It is a fact that high-resolution simulations are computationally expensive and time-consuming. However, integrating SR techniques in flow setup and analysis will generate high-quality data from lower-resolution inputs. This not only significantly reduces the computational load but also makes efficient use of available hardware and memory. Here, we have compared five DL models to perform the SR method to process several low-resolution turbulent flow field images. All models employ convolutional neural networks that are trained with high/low-resolution image pairs and are capable of providing a reconstructed image from a coarse field. Apart from the inherent advantage of giving a fine-resolution image by processing a low-resolution one, these convolutional models can dive deep into data characteristics hidden inside image pixels. The image pixels carry information that can be further processed and

analysed.

The constructed models include variants of the U-Net architecture, with (U-SKIP) and without (U-SIMPLE) skip connections, the encoder/decoder with residual connections (ED-RES), and SR architectures with max and average pooling layers (SR-MAX and SR-AVE, respectively). The main idea is to keep these models simple to run on standard hardware and pose an accurate alternative to classical computational fluid dynamics simulations. All proposed models achieve enhanced $PSNR$, with the encoder/decoder model with residual connections being the fastest implementation during training. Conversely, the U-Net-based model with skip connections is about three times slower, but its performance is better since it can produce sharper edges in regions prone to blurring. As we become more complex architectures and increase the depth of the convolutional network in SR-MAX and SR-AVE, superior performance in reconstructing velocity profiles is obtained. However, here, we need to make sure that no over-fitting occurs.

Regarding the $RMSE$, the SR-MAX and SR-AVE have achieved the best score, an order of magnitude lower than ED-RES. We also highlight the error increase when the input image is highly blurred, for example, when the high-resolution image is scaled down to 6 (LR6) or 8 (LR8) times and used as coarse input. For example, the U-SIMPLE $RMSE$ has increased by approximately 30% when the input resolution changes from LR4 to LR6 and 130% from LR6 to LR8. Nevertheless, in qualitative terms, the U-SKIP has given sharper edges in image regions where blurring was high.

Furthermore, time-averaged velocity profiles are extracted for the reconstructed fields and compared to the ground-truth profiles. It is found that time-averaged data have smaller errors compared to instantaneous 2D fields. Of interest is the fact that reconstruction is more accurate when the respective profile is smoother, without abrupt changes in values. In regions where velocity values present discontinuities,

the relative error of the ground truth vs. the reconstructed field has taken off from nearly 0 to 1.75 for the U-type and ED-RES models, while on the other hand, it remained less than 1.00 for the SR-MAX model.

The results indicate that simpler SR architectures, similar to the ones investigated in this paper, can be easily fitted to fluid mechanics computations that cover various types of fluid flows, from simple laminar to complex turbulent flows. This versatility makes them a powerful tool in the field. Coarse fields representing dynamic properties (such as velocity, pressure, temperature, and vorticity) can be mapped to a detailed, fine field without requiring time-consuming fine-grid simulations. This approach can also provide a fine parametric field that enhances the measurements from an experimental sensor setup, saving effort and device cost.

Super-resolution-aided fluid dynamics research offers significant detail, accuracy, and resource efficiency benefits. Its application spans various cross-disciplinary fields, including aerospace, automotive, environmental engineering, and biomedical applications. However, it's crucial to acknowledge the limitations of such methods. One primary concern is the potential for over-fitting, where the model performs well on training data but needs to generalise to unseen data. This is closely connected to the quality and quantity of the training data since insufficient or biased training datasets can lead to inaccurate or unreliable results.

Furthermore, it's crucial to exercise care during the training of an SR model, with proper memory management and exploitation of transfer learning techniques. There's also the risk of introducing artefacts or noise into the up-scaled data, which may distort the physical accuracy of the simulations. Finally, the black-box nature of many deep learning models makes them hard to pose as interpretable and trustworthy alternatives. This is important for understanding and justifying the model's decision in critical engineering applications. However, simpler models can enhance

interpretability and trustworthiness by offering more transparency in their operations and easier identification of how input features influence the output, making them more reliable for practical use.

Acknowledgements

This paper is supported by the European Union's Horizon Europe Research and Innovation Actions programme under grant agreement No 101069937, project name: HS4U (HEALTHY SHIP 4U). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure, and Environment Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. This work was also supported by computational time granted by the National Infrastructures for Research and Technology S.A. (GRNET S.A.) in the National HPC facility - ARIS - under project pr013008–AI–PHYSICS.

Data availability

The data supporting this study's findings are available on request.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] M. Yu, D. Sun, Q. Zhou, P. Liu, X. Yuan, Coherent structures and turbulent model refinement in oblique shock/hypersonic turbulent boundary layer interactions, Physics of Fluids 35 (2023) 086125. URL: https://doi.org/10.1063/5.0163259. doi:10.1063/5.0163259.

[2] D. Drikakis, F. Sofos, Can artificial intelligence accelerate fluid mechanics research?, Fluids 8 (2023). URL: https://www.mdpi.com/2311-5521/8/7/212. doi:10.3390/fluids8070212.

[3] R. Pugliese, S. Regondi, R. Marini, Machine learning-based approach: global trends, research directions, and regulatory standpoints, Data Science and Management 4 (2021) 19–29. doi:https://doi.org/10.1016/j.dsm.2021.12.002.

[4] M. Frank, D. Drikakis, V. Charissis, Machine-learning methods for computational science and engineering, Computation 8 (2020) 15. URL: https://www.mdpi.com/2079-3197/8/1/15. doi:10.3390/computation8010015.

[5] J. Valente, J. Antonio, C. Mora, S. Jardim, Developments in image processing using deep learning and reinforcement learning, Journal of Imaging 9 (2023). doi:10.3390/jimaging9100207.

[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444. doi:10.1038/nature14539.

[7] Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, J. Morimoto, Deep learning, reinforcement learning, and world models, Neural Networks 152 (2022) 267–275. doi:10.1016/j.neunet.2022.03.037.

[8] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, J. N. Kutz, Shallow neural networks for fluid flow reconstruction with limited sensors, Proceedings of the Royal Society A 476 (2020) 20200097. doi:10.1098/rspa.2020.0097.

[9] M. I. Razzak, S. Naz, A. Zaib, Deep Learning for Medical Image Processing: Overview, Challenges and the Future, Springer International Publishing, Cham, 2018, pp. 323–350. doi:10.1007/978-3-319-65981-7-12.

[10] S. Suganyadevi, V. Seethalakshmi, K. Balasamy, A review on deep learning in medical image analysis, International Journal of Multimedia Information Retrieval 11 (2022) 19–38. URL: https://doi.org/10.1007/s13735-021-00218-1. doi:10.1007/s13735-021-00218-1.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 25, Curran Associates, Inc., 2012.

[13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[15] F. Sofos, D. Drikakis, I. W. Kokkinakis, S. M. Spottswood, Convolutional neural

networks for compressible turbulent flow reconstruction, Physics of Fluids 35 (2023) 116120. doi:10.1063/5.0177654.

[16] F. Sofos, D. Drikakis, I. W. Kokkinakis, S. M. Spottswood, A deep learning super-resolution model for turbulent image upscaling and its application to shock wave–boundary layer interaction, Physics of Fluids 36 (2024) 025117. doi:10.1063/5.0190272.

[17] K. Fukami, K. Fukagata, K. Taira, Super-resolution analysis via machine learning: a survey for fluid flows, Theoretical and Computational Fluid Dynamics 37 (2023) 421–444. URL: https://doi.org/10.1007/s00162-023-00663-0. doi:10.1007/s00162-023-00663-0.

[18] L. Yu, M. Z. Yousif, M. Zhang, S. Hoyas, R. Vinuesa, H.-C. Lim, Three-dimensional ESRGAN for super-resolution reconstruction of turbulent flows with tricubic interpolation-based transfer learning, Physics of Fluids 34 (2022) 125126. doi:10.1063/5.0129203.

[19] Z. Deng, C. He, Y. Liu, K. C. Kim, Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework, Physics of Fluids 31 (2019) 125111. doi:10.1063/1.5127031.

[20] A. Guemes, S. Discetti, A. Ianiro, B. Sirmacek, H. Azizpour, R. Vinuesa, From coarse wall measurements to turbulent velocity fields through deep learning, Physics of Fluids 33 (2021) 075121. doi:10.1063/5.0058346.

[21] H. Kim, J. Kim, S. Won, C. Lee, Unsupervised deep learning for super-resolution reconstruction of turbulence, Journal of Fluid Mechanics 910 (2021) A29. doi:10.1017/jfm.2020.1028.

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi, Photo-realistic single image super-resolution using a generative adversarial network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 105–114. doi:10.1109/CVPR.2017.19.

[23] C. Cheng, G.-T. Zhang, Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems, Water 13 (2021). URL: https://www.mdpi.com/2073-4441/13/4/423. doi:10.3390/w13040423.

[24] D. Shu, Z. Li, A. Barati Farimani, A physics-informed diffusion model for high-fidelity flow field reconstruction, Journal of Computational Physics 478 (2023) 111972. doi:https://doi.org/10.1016/j.jcp.2023.111972.

[25] Z. Deng, Y. Chen, Y. Liu, K. C. Kim, Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework, Physics of Fluids 31 (2019). doi:10.1063/1.5111558, 075108.

[26] M. Z. Yousif, M. Zhang, L. Yu, R. Vinuesa, H. Lim, A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers, Journal of Fluid Mechanics 957 (2023) A6. doi:10.1017/jfm.2022.1088.

[27] A. Kiener, S. Langer, P. Bekemeyer, Data-driven correction of coarse grid cfd simulations, Computers & Fluids 264 (2023) 105971. doi:https://doi.org/10.1016/j.compfluid.2023.105971.

[28] K. Bao, X. Zhang, W. Peng, W. Yao, Deep learning method for super-resolution reconstruction of the spatio-temporal flow field, Advances in Aerody-

namics 5 (2023) 19. URL: https://doi.org/10.1186/s42774-023-00148-y. doi:10.1186/s42774-023-00148-y.

[29] T. Muther, A. K. Dahaghi, F. I. Syed, V. Van Pham, Physical laws meet machine intelligence: current developments and future directions, Artificial Intelligence Review 56 (2023) 6947–7013. doi:10.1007/s10462-022-10329-8, iD: Muther2023.

[30] M. Bode, J. H. Göbbert, Acceleration of complex high-performance computing ensemble simulations with super-resolution-based subfilter models, Computers & Fluids 271 (2024) 106150. doi:https://doi.org/10.1016/j.compfluid.2023.106150.

[31] Z. Wang, J. Chen, S. C. H. Hoi, Deep learning for image super-resolution: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021) 3365–3387. doi:10.1109/TPAMI.2020.2982166.

[32] A. Khan, A. Sohail, U. Zahoora, A. S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, Artificial Intelligence Review 53 (2020) 5455–5516. URL: https://doi.org/10.1007/s10462-020-09825-6. doi:10.1007/s10462-020-09825-6.

[33] C. Dong, C. C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham, 2014, pp. 184–199.

[34] K. Fukami, K. Fukagata, K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, Journal of Fluid Mechanics 870 (2019) 106–120. doi:10.1017/jfm.2019.238.

[35] K. Fukami, K. Hasegawa, T. Nakamura, M. Morimoto, K. Fukagata, Model Order Reduction with Neural Networks: Application to Laminar and Turbulent Flows, SN Computer Science 2 (2021) 467. doi:10.1007/s42979-021-00867-3.

[36] C. Kong, J.-T. Chang, Y.-F. Li, R.-Y. Chen, Deep learning methods for super-resolution reconstruction of temperature fields in a supersonic combustor, AIP Advances 10 (2020) 115021. doi:10.1063/5.0030040.

[37] Y. Chen, R. Xia, K. Yang, K. Zou, Micu: Image super-resolution via multi-level information compensation and u-net, Expert Systems with Applications 245 (2024) 123111. doi:https://doi.org/10.1016/j.eswa.2023.123111.

[38] Y. Chen, R. Xia, K. Yang, K. Zou, Dnnam: Image inpainting algorithm via deep neural networks and attention mechanism, Applied Soft Computing 154 (2024) 111392. doi:https://doi.org/10.1016/j.asoc.2024.111392.

[39] Y. Chen, R. Xia, K. Yang, K. Zou, Mfmam: Image inpainting via multi-scale feature module with attention module, Computer Vision and Image Understanding 238 (2024) 103883. doi:https://doi.org/10.1016/j.cviu.2023.103883.

[40] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C. C. Loy, Esrgan: Enhanced super-resolution generative adversarial networks, in: L. Leal-Taixé, S. Roth (Eds.), Computer Vision – ECCV 2018 Workshops, Springer International Publishing, Cham, 2019, pp. 63–79.

[41] H. Kim, J. Kim, S. Won, C. Lee, Unsupervised deep learning for super-resolution reconstruction of turbulence, Journal of Fluid Mechanics 910 (2021) A29. doi:10.1017/jfm.2020.1028.

[42] K. Karantonis, I. W. Kokkinakis, B. Thornber, D. Drikakis, Compressibility in suddenly expanded subsonic flows, Physics of Fluids 33 (2021) 105106. doi:10.1063/5.0065257.

[43] H. Habibi Aghdam, E. Jahani Heravi, Convolutional Neural Networks, Springer International Publishing, Cham, 2017, pp. 85–130. doi:10.1007/978-3-319-57550-6-3.

[44] L. Xu, J. S. Ren, C. Liu, J. Jia, Deep convolutional neural network for image deconvolution, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 27, Curran Associates, Inc., 2014.

[45] K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. L. Billinge, E. Holm, S. P. Ong, C. Wolverton, Recent advances and applications of deep learning methods in materials science, npj Computational Materials 8 (2022) 1–26. doi:10.1038/s41524-022-00734-6, number: 1 Publisher: Nature Publishing Group.

[46] R. F. Cerqueira, E. E. Paladino, Development of a deep learning-based image processing technique for bubble pattern recognition and shape reconstruction in dense bubbly flows, Chemical Engineering Science 230 (2021) 116163. doi:https://doi.org/10.1016/j.ces.2020.116163.

[47] J. Kim, J. K. Lee, K. M. Lee, Accurate image super-resolution using very deep convolutional networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1646–1654. doi:10.1109/CVPR.2016.182.

[48] L. Prechelt, Automatic early stopping using cross validation: quantifying the cri-

teria, Neural Networks 11 (1998) 761–767. doi:https://doi.org/10.1016/S0893-6080(98)00010-0.

[49] L. Casarsa, P. Giannattasio, Three-dimensional features of the turbulent flow through a planar sudden expansion, Physics of Fluids 20 (2008) 015103. doi:10.1063/1.2832780.

[50] D. S. Balsara, C.-W. Shu, Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy, Journal of Computational Physics 160 (2000) 405 – 452. doi:10.1006/jcph.2000.6443.

[51] I. W. Kokkinakis, D. Drikakis, K. Ritos, S. M. Spottswood, Direct numerical simulation of supersonic flow and acoustics over a compression ramp, Physics of Fluids 32 (2020) 066107. doi:10.1063/5.0010548.

[52] E. F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1994) 25 – 34. doi:10.1007/BF01414629".

[53] R. Spiteri, S. Ruuth, A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM Journal on Numerical Analysis 40 (2002) 469–491. doi:10.1137/S0036142901389025.

[54] I. Kokkinakis, D. Drikakis, Implicit large eddy simulation of weakly-compressible turbulent channel flow, Computer Methods in Applied Mechanics and Engineering 287 (2015) 229 – 261. doi:10.1016/j.cma.2015.01.016.

[55] M. Klein, A. Sadiki, J. Janicka, A digital filter based generation of inflow data for spatially developing direct numerical or large eddy simulations, Journal of Computational Physics 186 (2003) 652 – 665. doi:10.1016/S0021-9991(03)00090-1.

[56] E. Touber, N. D. Sandham, Large-eddy simulation of low-frequency unsteadiness in a turbulent shock-induced separation bubble, Theoretical and Computational Fluid Dynamics 23 (2009) 79 – 107. doi:10.1007/s00162-009-0103-z.

[57] N. J. Georgiadis, D. P. Rizzetta, C. Fureby, Large-eddy simulation: current capabilities, recommended practices, and future research., AIAA Journal 48 (2010) 1172–1784. doi:10.2514/1.J050232.

[58] H. Choi, P. Moin, Grid-point requirements for large eddy simulation: Chapman's estimates revisited, Physics of Fluids 24 (2012) 011702. doi:10.1063/1.3676783.

[59] J. Poggie, N. J. Bisek, R. Gosse, Resolution effects in compressible, turbulent boundary layer simulations, Computers & Fluids 120 (2015) 57–69. doi:10.1016/j.compfluid.2015.07.015.

[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: A system for large-scale machine learning, arXiv preprint arXiv:1603.04467 (2016).

[61] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: N. Navab, J. Hornegger, W. M. Wells, A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 234–241.

[62] T. Falk, D. Mai, R. Bensch, O. Cicek, A. Abdulkadir, Y. Marrakchi, A. Bohm, J. Deubner, Z. Jackel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, O. Ronneberger, U-Net: deep

learning for cell counting, detection, and morphometry, Nature Methods 16 (2019) 67–70. URL: https://doi.org/10.1038/s41592-018-0261-2. doi:10.1038/s41592-018-0261-2.

[63] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[64] J. A. L. Marques, F. N. B. Gois, J. P. do Vale Madeiro, T. Li, S. J. Fong, Chapter 4 - artificial neural network-based approaches for computer-aided disease diagnosis and treatment, in: A. K. Bhoi, V. H. C. de Albuquerque, P. N. Srinivasu, G. Marques (Eds.), Cognitive and Soft Computing Techniques for the Analysis of Healthcare Data, Intelligent Data-Centric Systems, Academic Press, 2022, pp. 79–99. doi:https://doi.org/10.1016/B978-0-323-85751-2.00008-6.

[65] M. Gupta, H. Taneja, L. Chand, Performance enhancement and analysis of filters in ultrasound image denoising, Procedia Computer Science 132 (2018) 643–652. doi:https://doi.org/10.1016/j.procs.2018.05.063, international Conference on Computational Intelligence and Data Science.

Appendix A. Velocity profiles and metrics

Velocity profiles and error metrics are given in the following Figures, from cross-sections 2-5, for LR4 and LR6 image sets. The velocity profile is shown in every subfigure (a), an identity plot (ground truth vs reconstructed profile), with R^2 shown, in (b) and the relative error $\overline{e} = \frac{|y_{HR} - y_{rec}|}{y_{HR}}$, with MAE and RMSE shown, in (c).

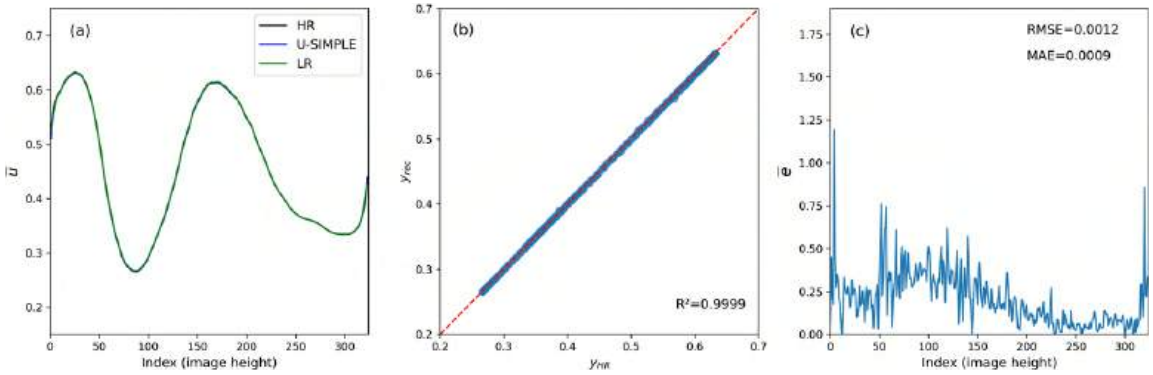


Figure A.18: U-SIMPLE - Cross-section 2: LR4 validation dataset.

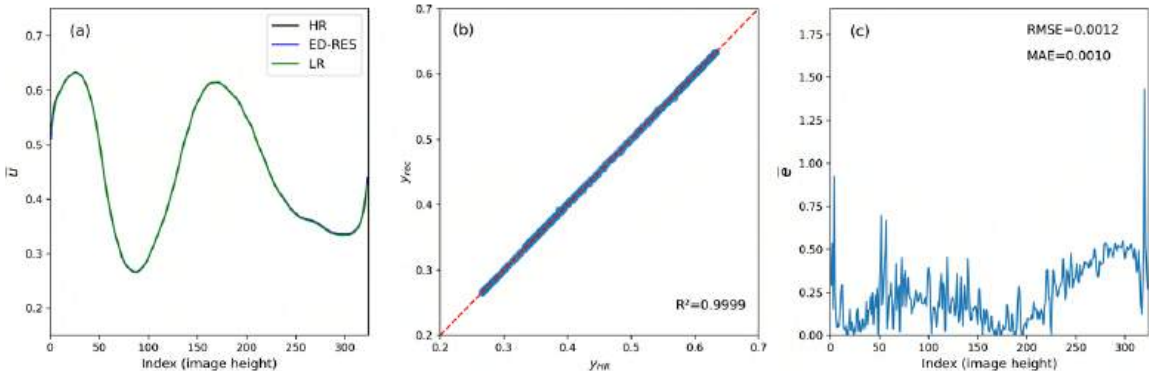


Figure A.19: ED-RES - Cross-section 2: LR4 validation dataset.

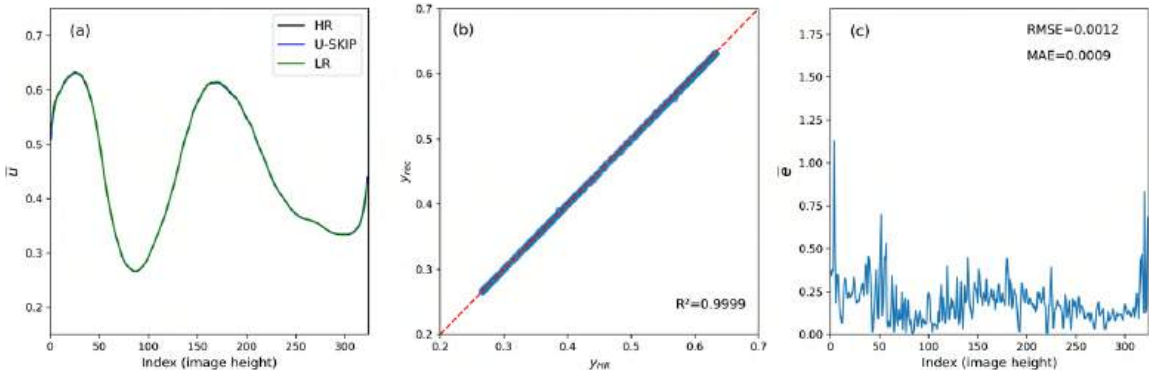


Figure A.20: U-SKIP - Cross-section 2: LR4 validation dataset.

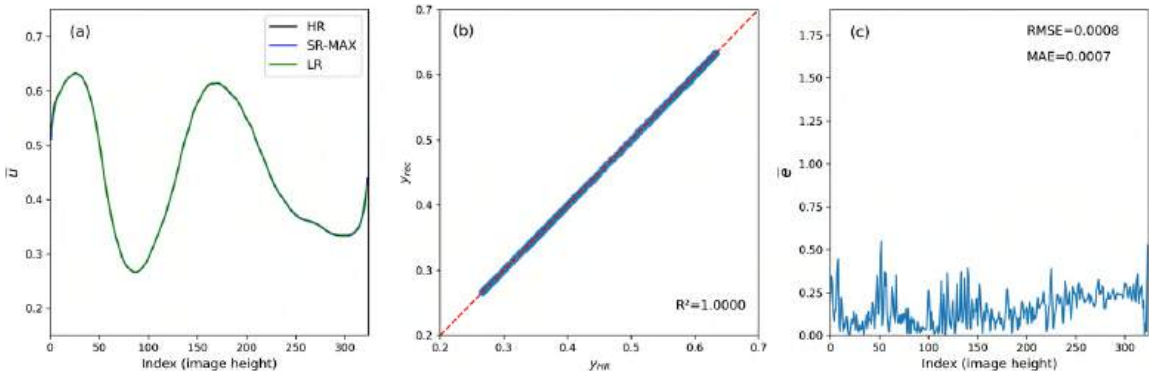


Figure A.21: SR-MAX - Cross-section 2: LR4 validation dataset.

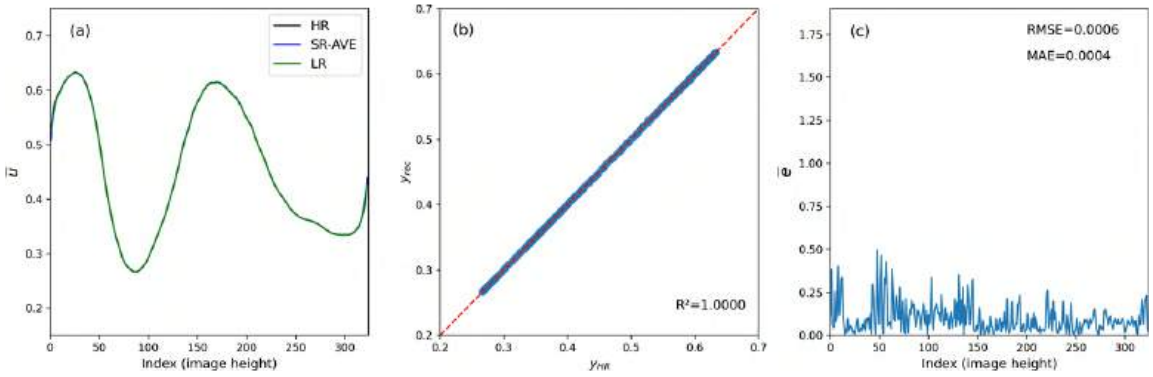


Figure A.22: SR-AVE - Cross-section 2: LR4 validation dataset.

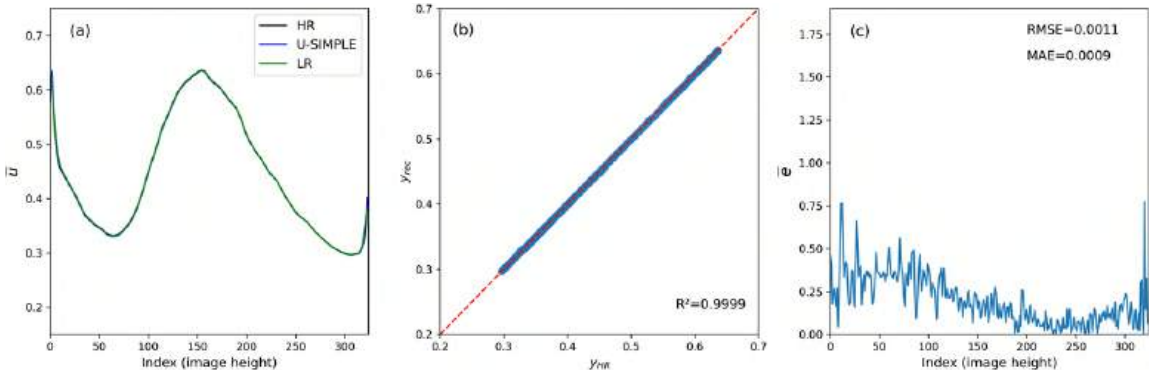


Figure A.23: U-SIMPLE - Cross-section 3: LR4 validation dataset.

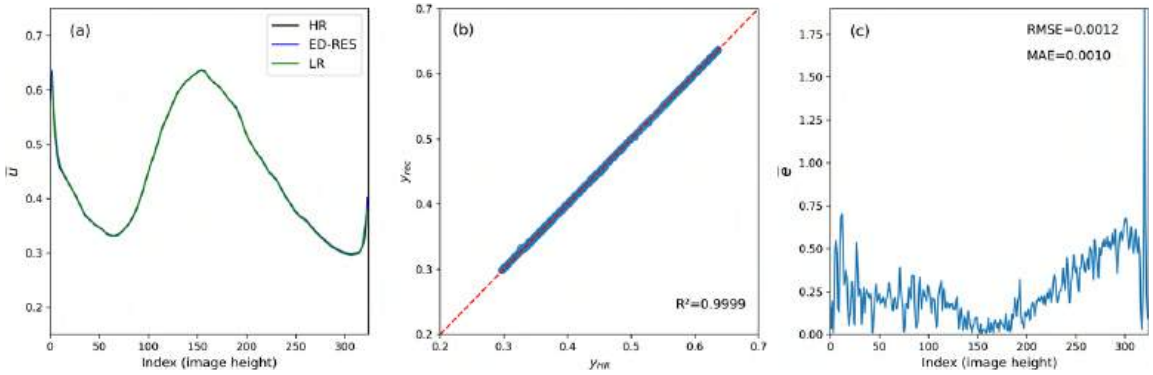


Figure A.24: ED-RES - Cross-section 3: LR4 validation dataset.

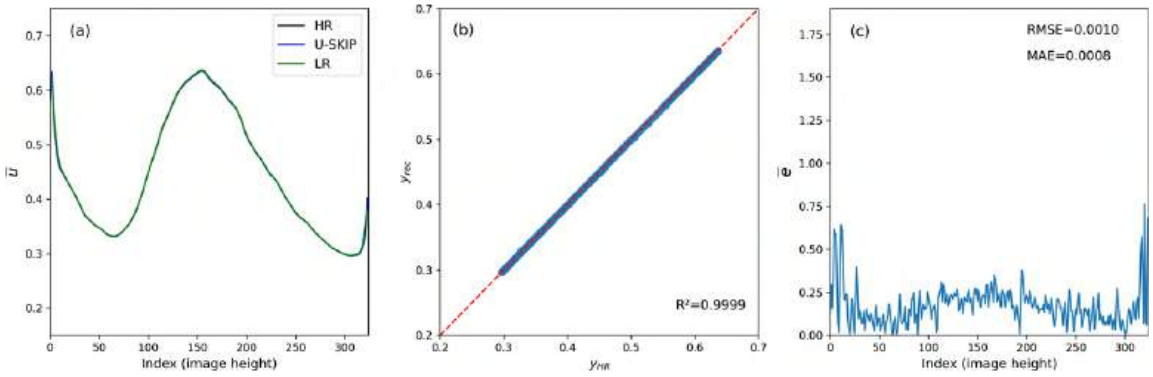


Figure A.25: U-SKIP - Cross-section 3: LR4 validation dataset.

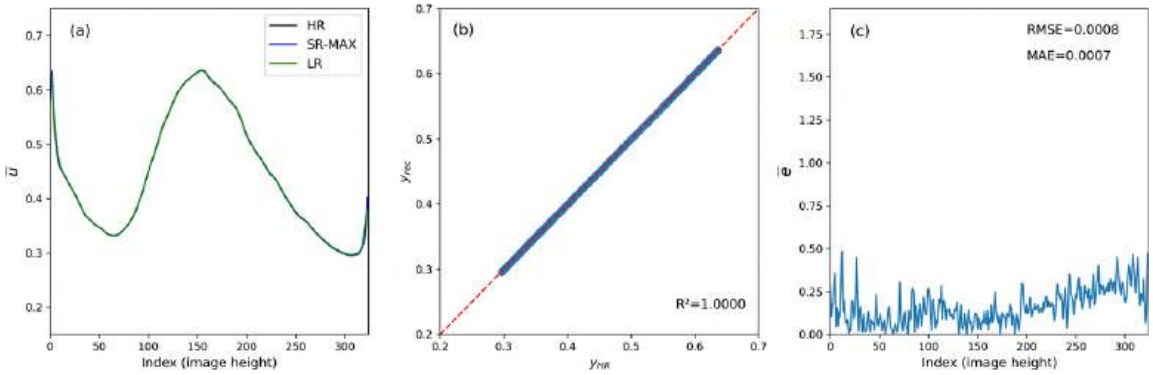


Figure A.26: SR-MAX - Cross-section 3: LR4 validation dataset.

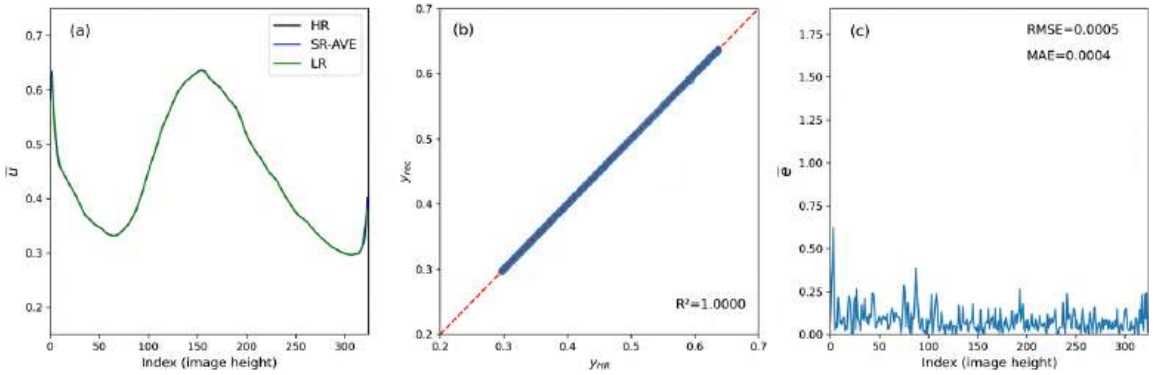


Figure A.27: SR-AVE - Cross-section 3: LR4 validation dataset.

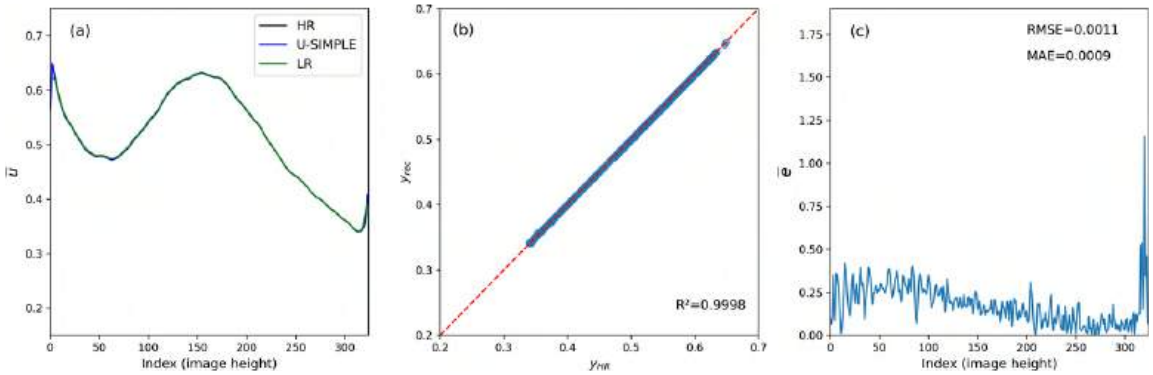


Figure A.28: U-SIMPLE - Cross-section 4: LR4 validation dataset.

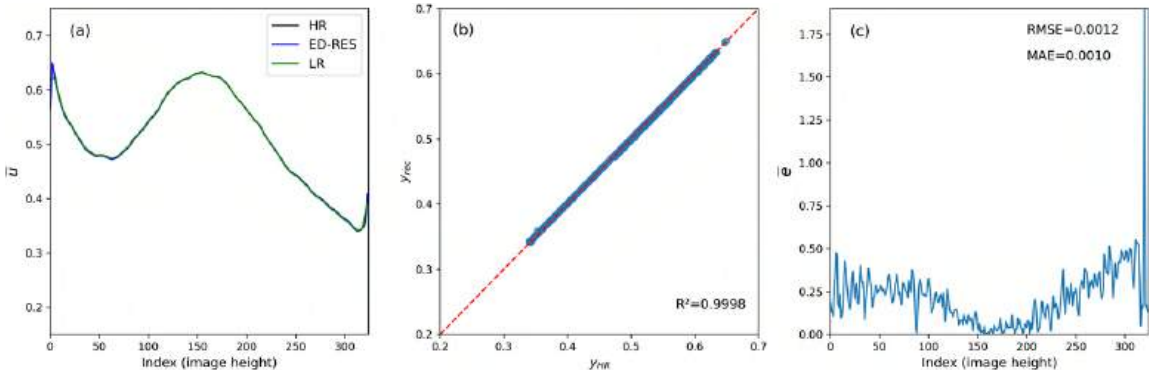


Figure A.29: ED-RES - Cross-section 4: LR4 validation dataset.

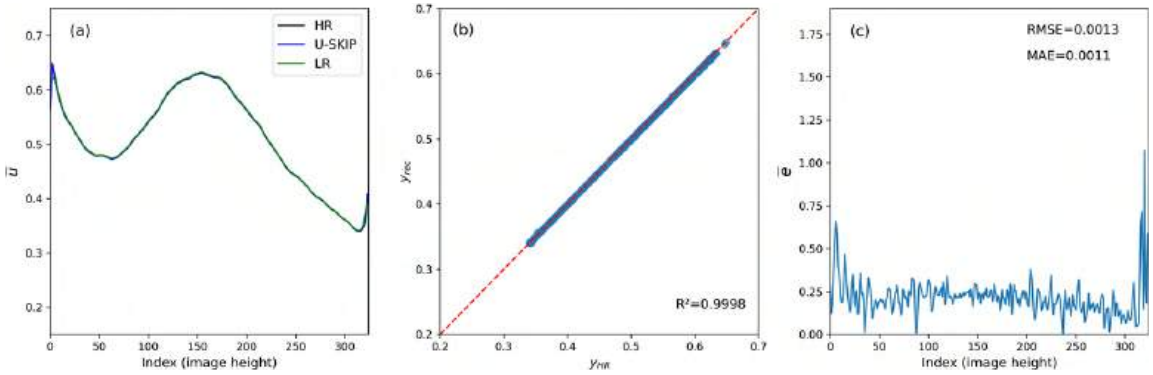


Figure A.30: U-SKIP - Cross-section 4: LR4 validation dataset.

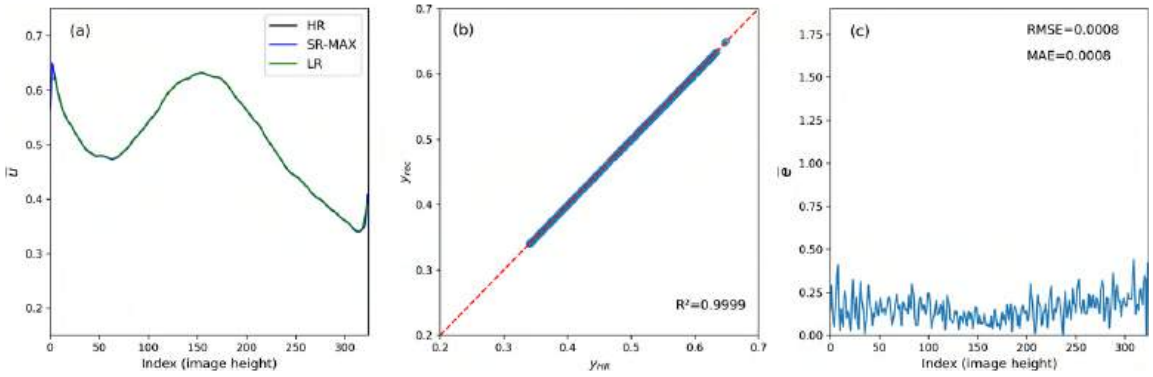


Figure A.31: SR-MAX - Cross-section 4: LR4 validation dataset.

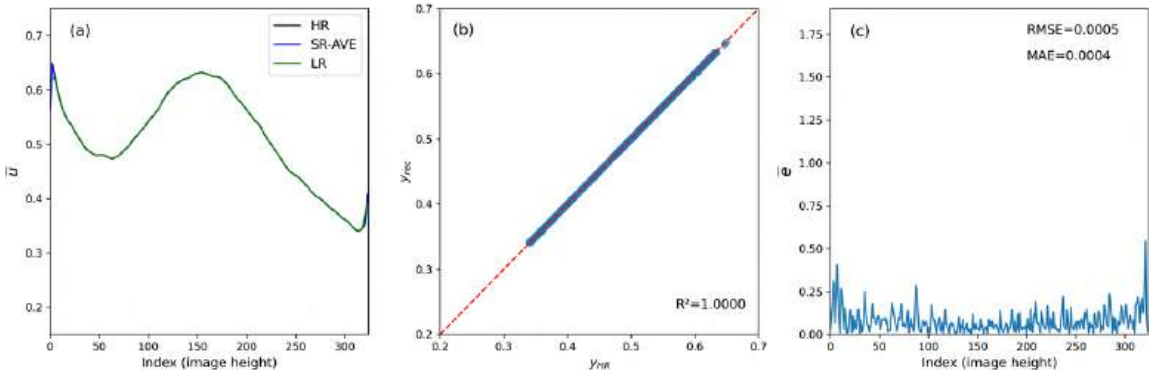


Figure A.32: SR-AVE - Cross-section 4: LR4 validation dataset.

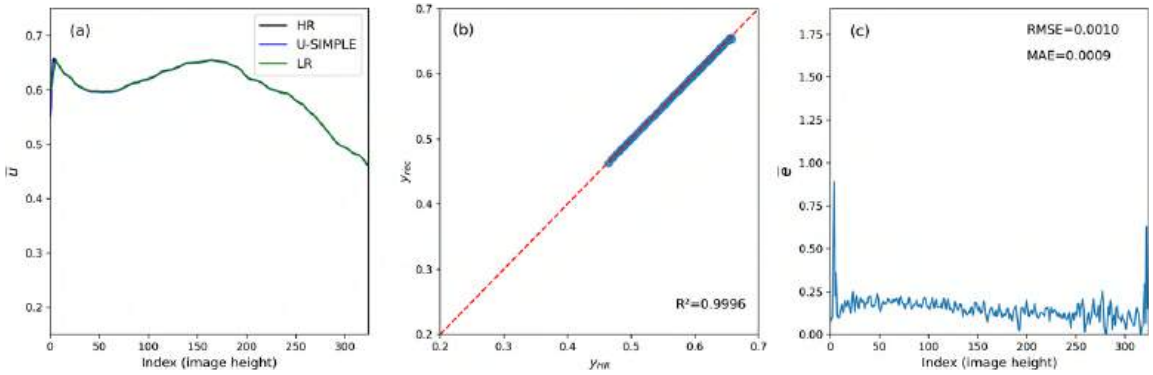


Figure A.33: U-SIMPLE - Cross-section 5: LR4 validation dataset.

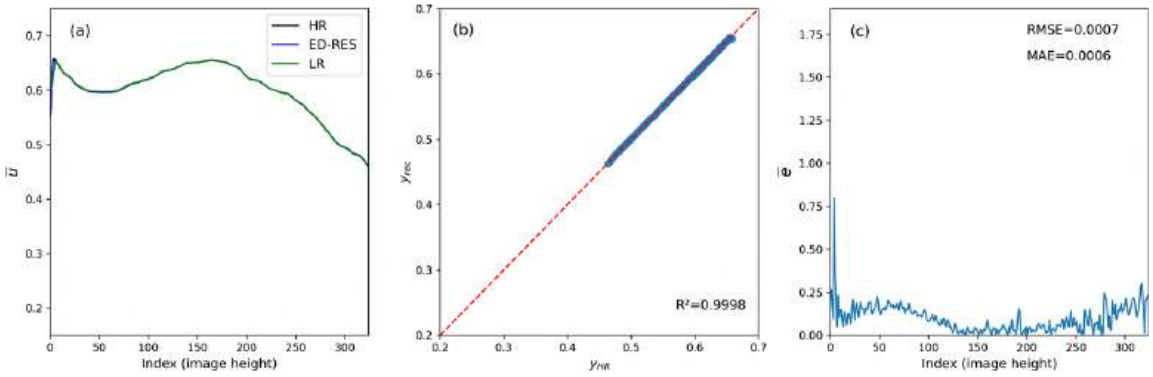


Figure A.34: ED-RES - Cross-section 5: LR4 validation dataset.

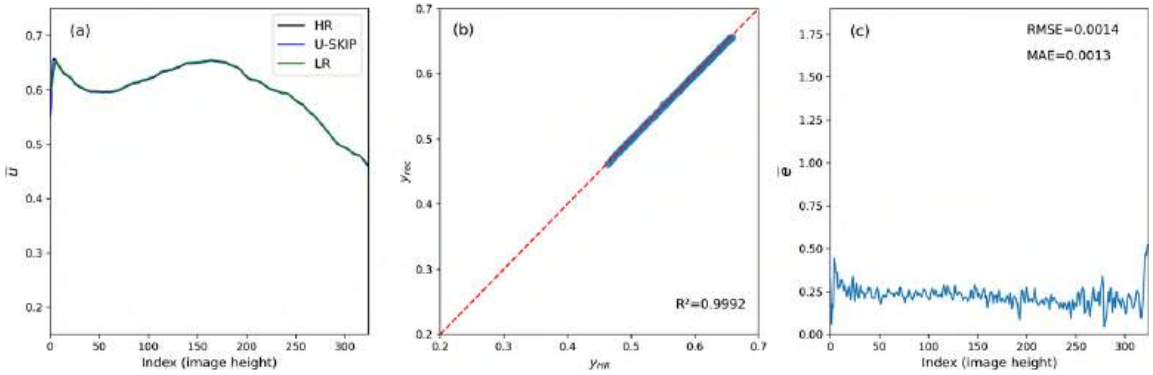


Figure A.35: U-SKIP - Cross-section 5: LR4 validation dataset.

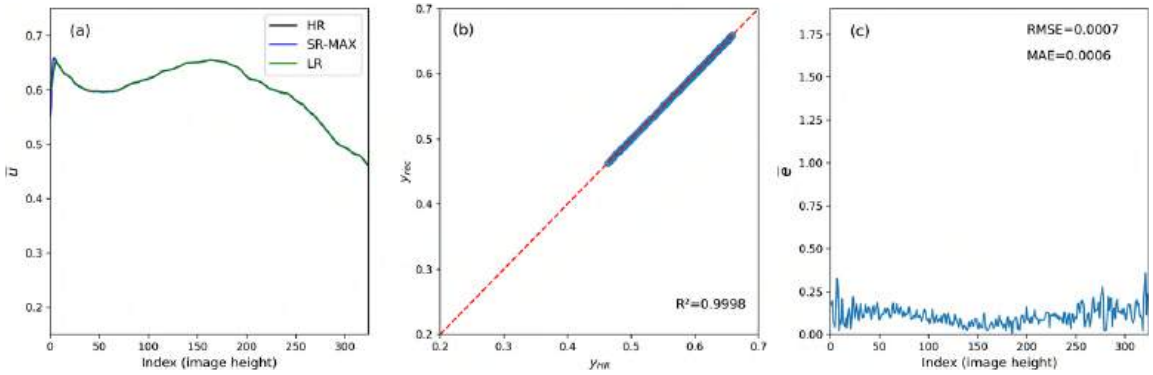


Figure A.36: SR-MAX - Cross-section 5: LR4 validation dataset.

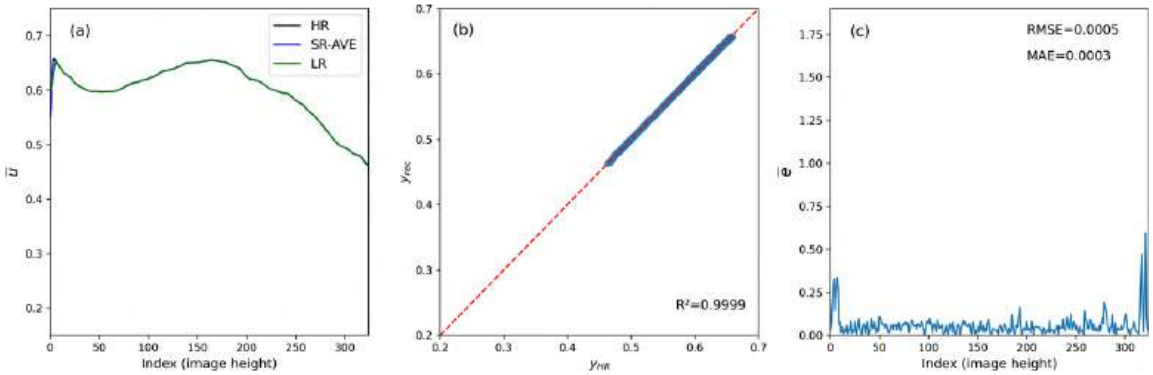


Figure A.37: SR-AVE - Cross-section 5: LR4 validation dataset.

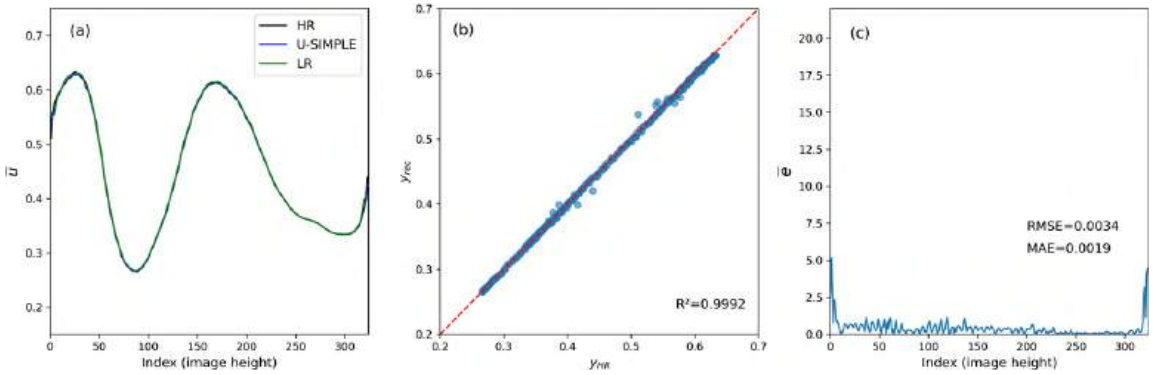


Figure A.38: U-SIMPLE - Cross-section 2: LR6 validation dataset.

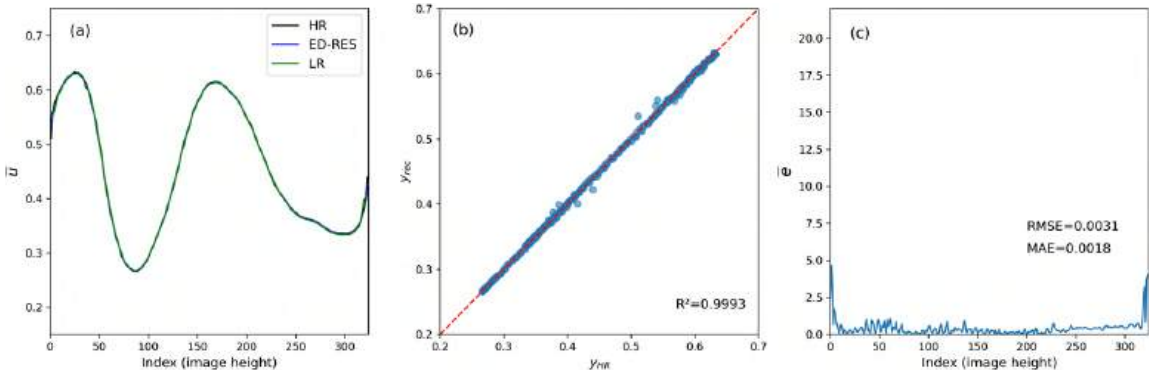


Figure A.39: ED-RES - Cross-section 2: LR6 validation dataset.

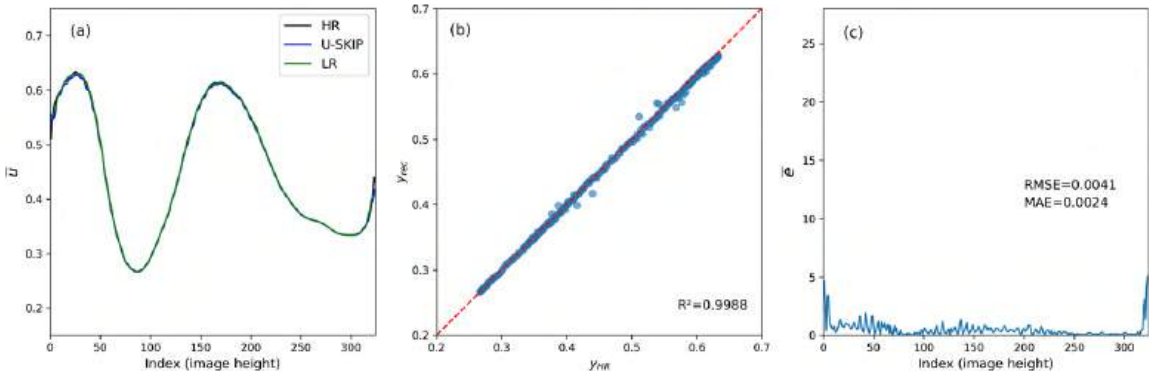


Figure A.40: U-SKIP - Cross-section 2: LR6 validation dataset.

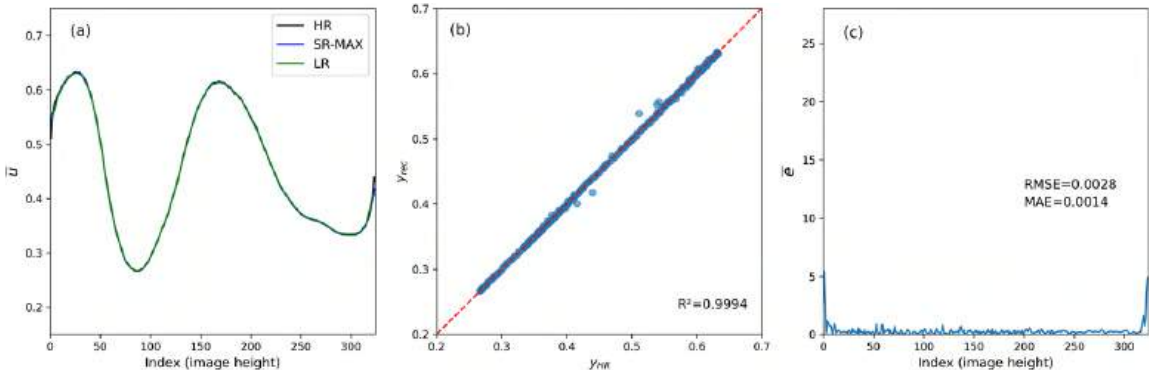


Figure A.41: SR-MAX - Cross-section 2: LR6 validation dataset.

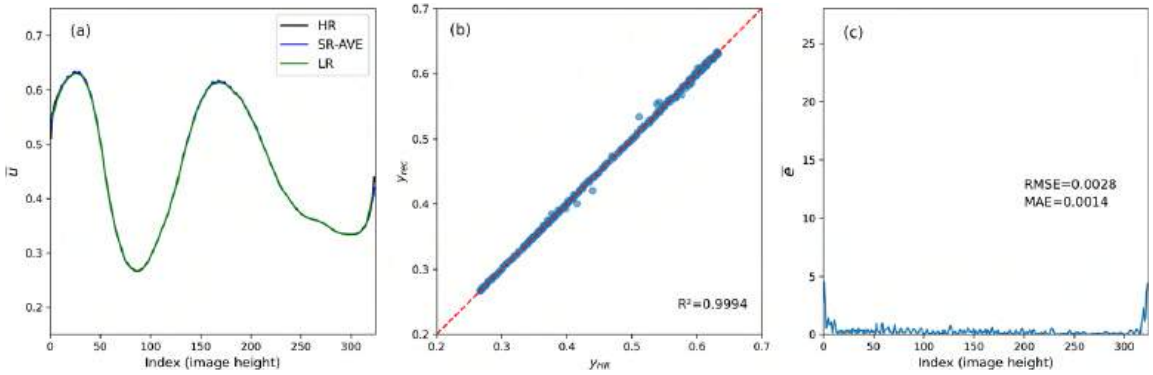


Figure A.42: SR-AVE - Cross-section 2: LR6 validation dataset.

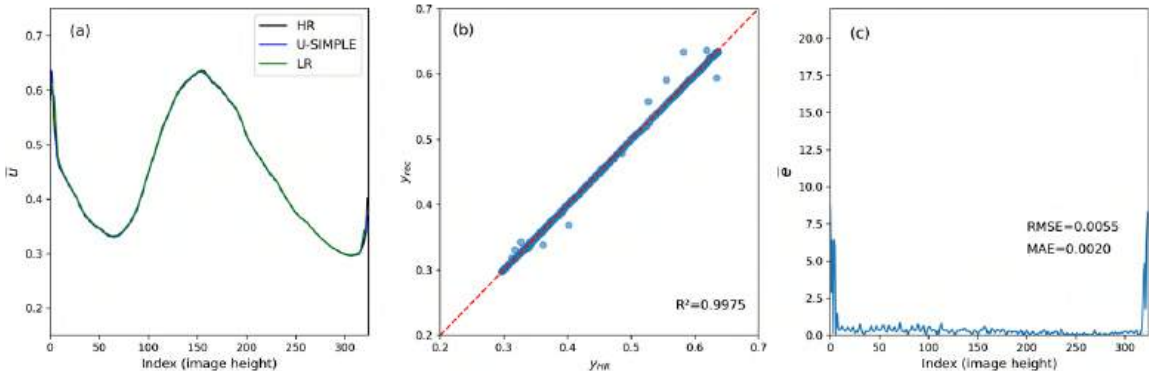


Figure A.43: U-SIMPLE - Cross-section 3: LR6 validation dataset.

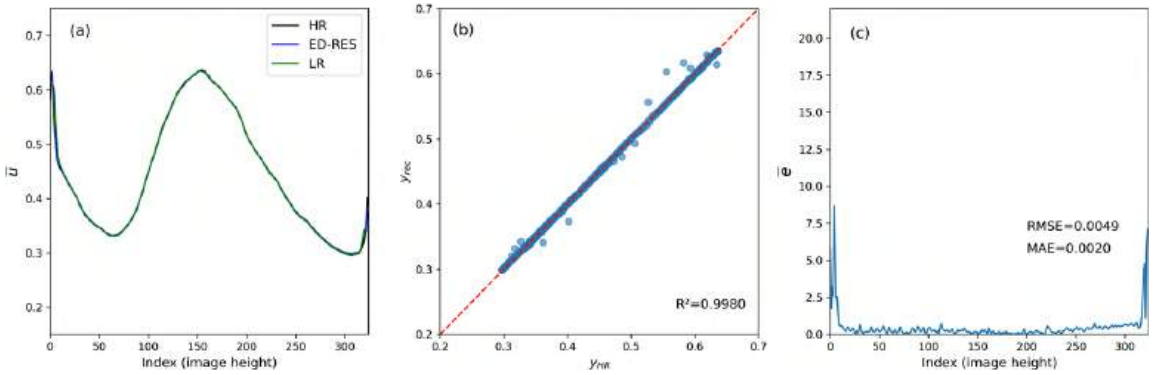


Figure A.44: ED-RES - Cross-section 3: LR6 validation dataset.

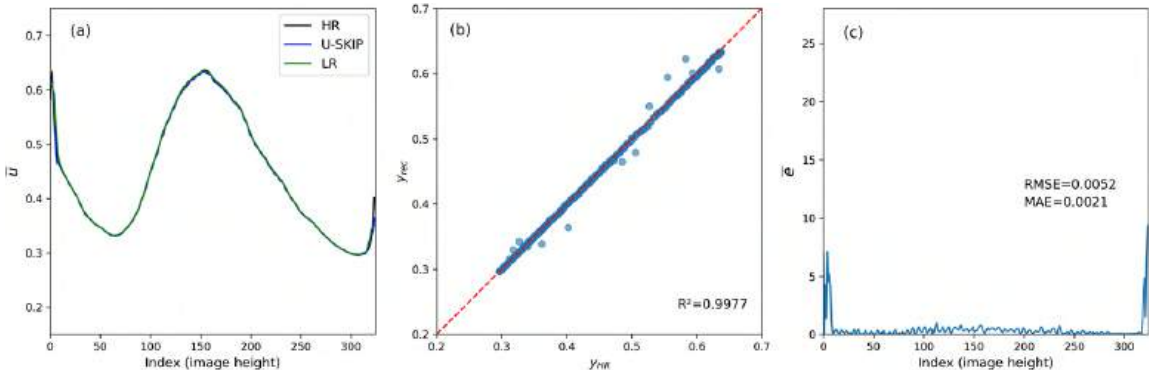


Figure A.45: U-SKIP - Cross-section 3: LR6 validation dataset.

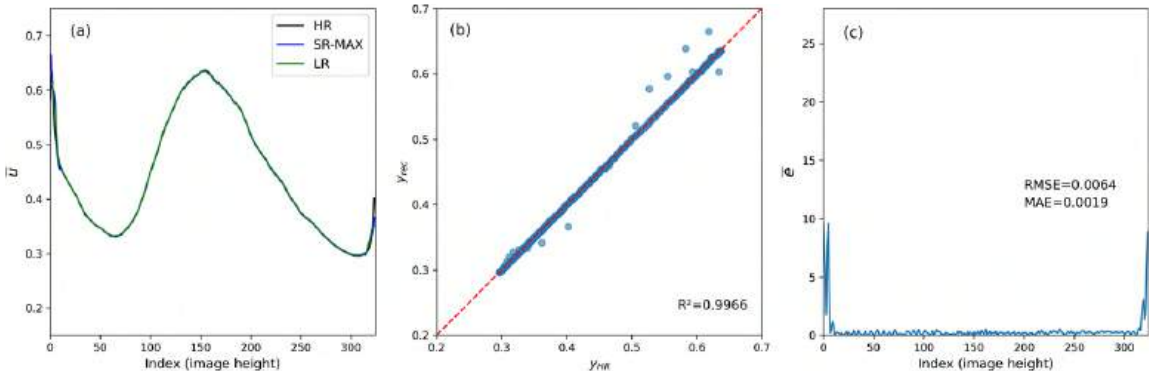


Figure A.46: SR-MAX - Cross-section 3: LR6 validation dataset.

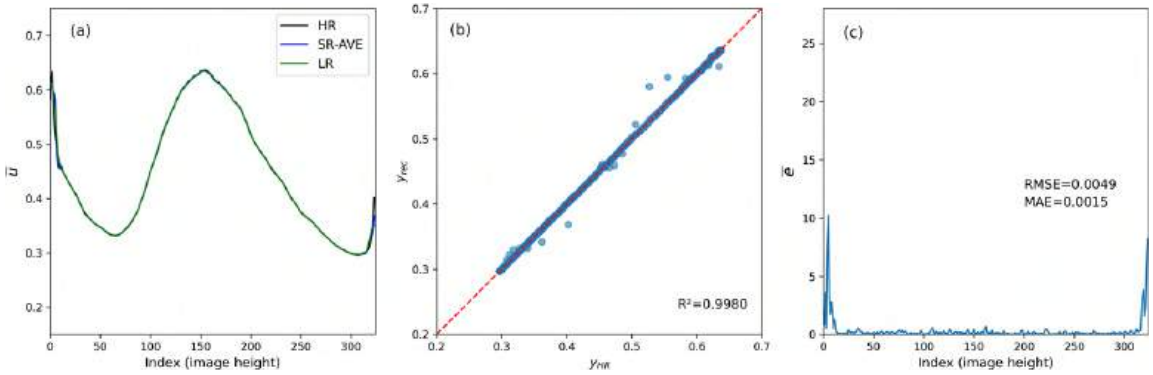


Figure A.47: SR-AVE - Cross-section 3: LR6 validation dataset.

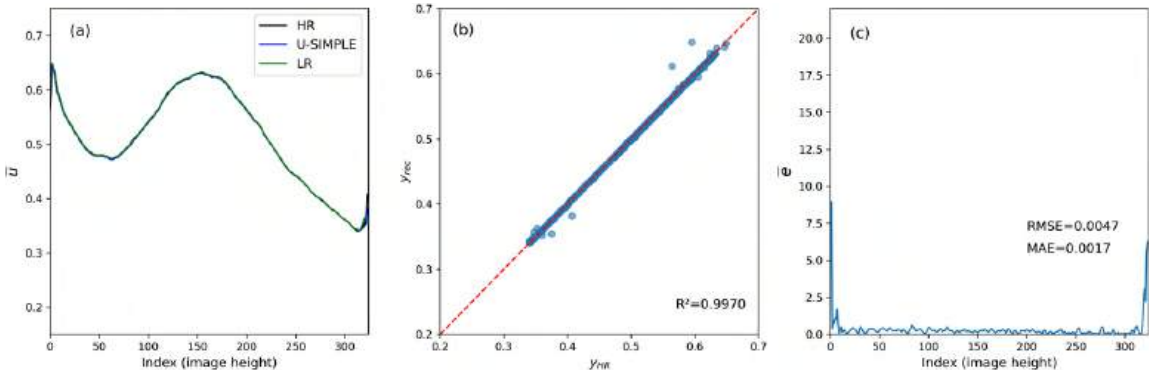


Figure A.48: U-SIMPLE - Cross-section 4: LR4 validation dataset.

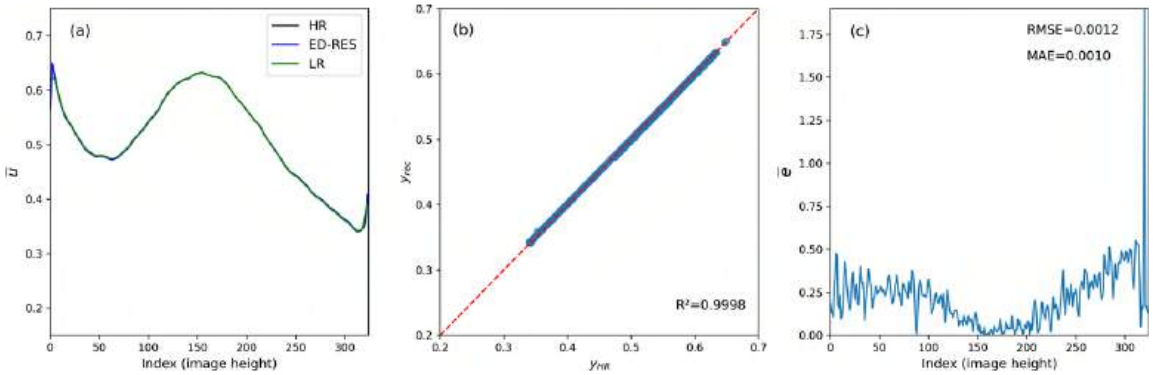


Figure A.49: ED-RES - Cross-section 4: LR4 validation dataset.

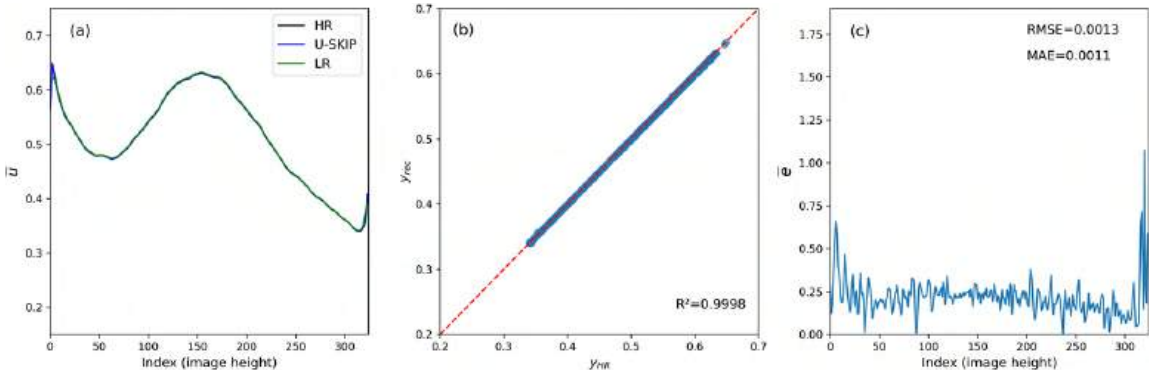


Figure A.50: U-SKIP - Cross-section 4: LR4 validation dataset.

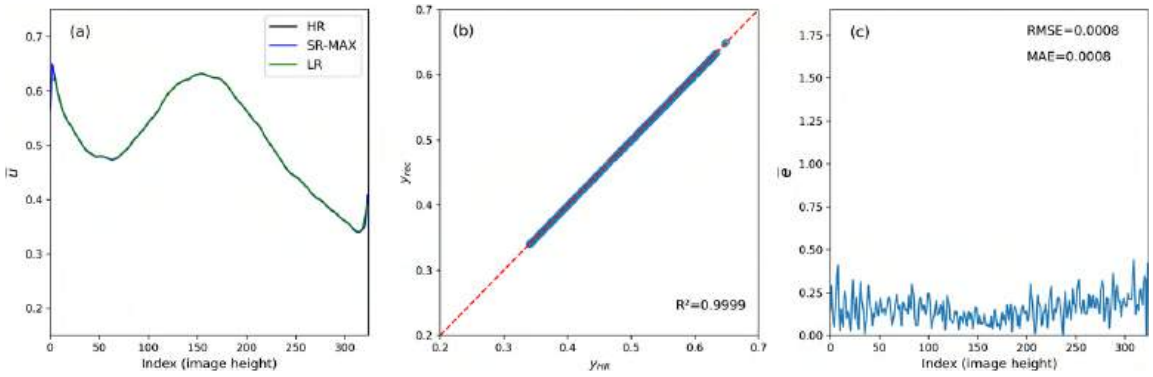


Figure A.51: SR-MAX - Cross-section 4: Metrics and average velocity profiles reconstruction for the LR4 validation dataset. (a-b) Velocity profile, (b) identity plot (ground truth vs. reconstructed profile), with R^2 shown, (c) relative error $\overline{e} = \frac{|y_{HR} - y_{rec}|}{y_{HR}}$, with MAE and RMSE shown.

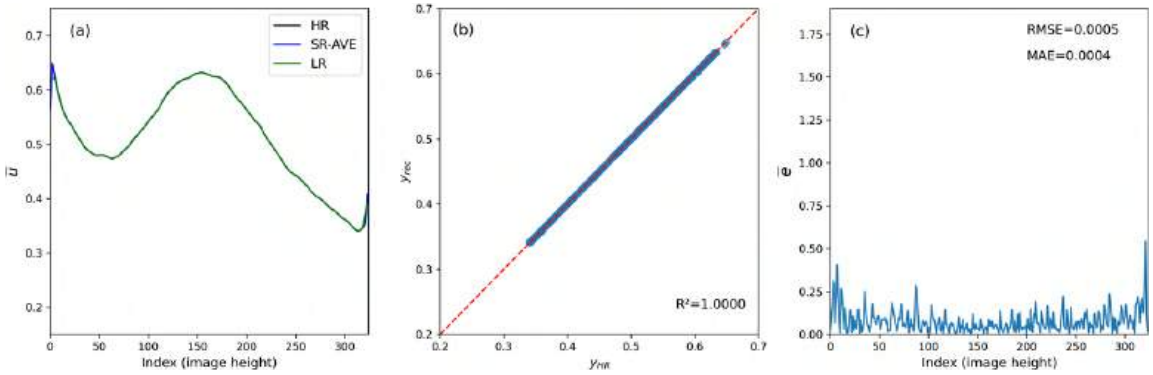


Figure A.52: SR-AVE - Cross-section 4: LR4 validation dataset.

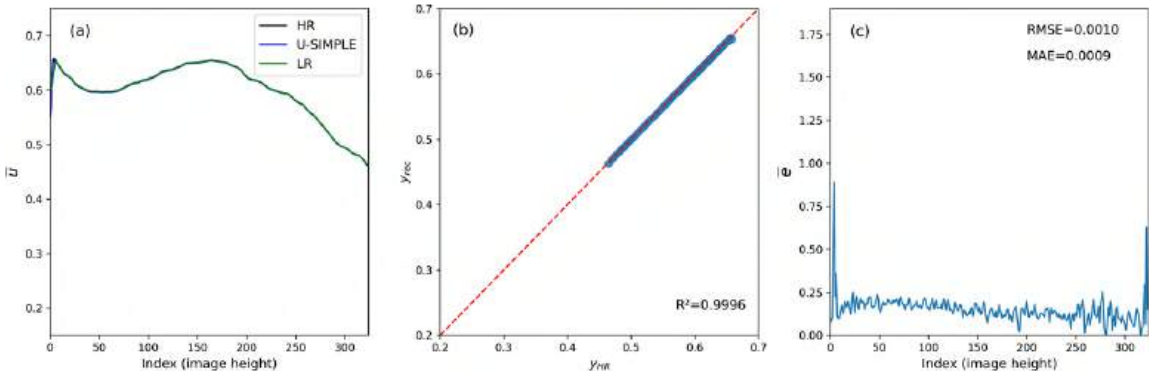


Figure A.53: U-SIMPLE - Cross-section 5: LR4 validation dataset.

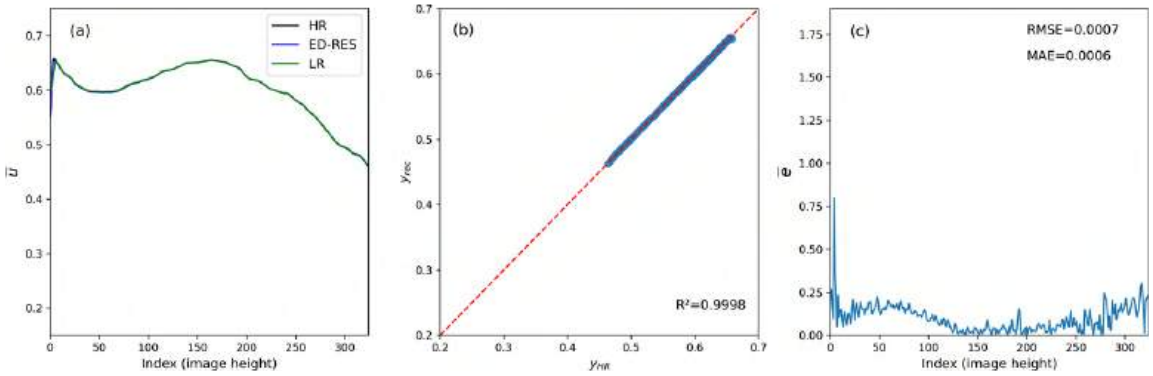


Figure A.54: ED-RES - Cross-section 5: LR4 validation dataset.

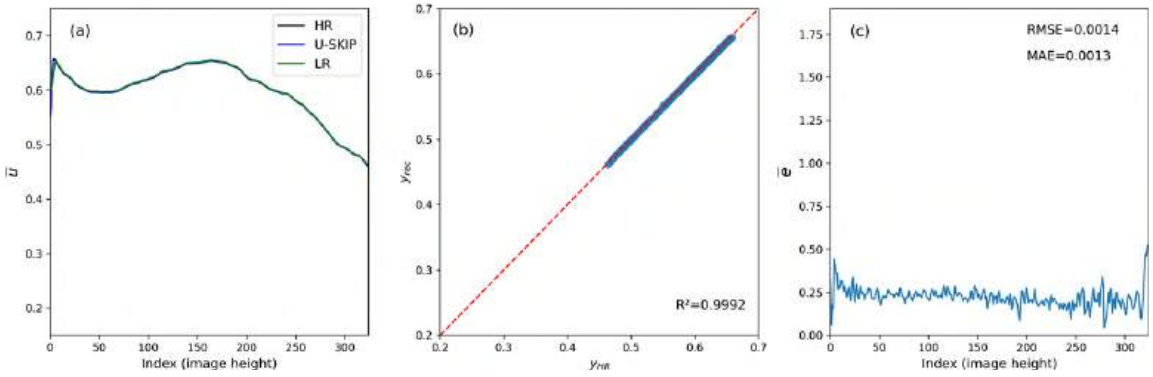


Figure A.55: U-SKIP - Cross-section 5: LR4 validation dataset.

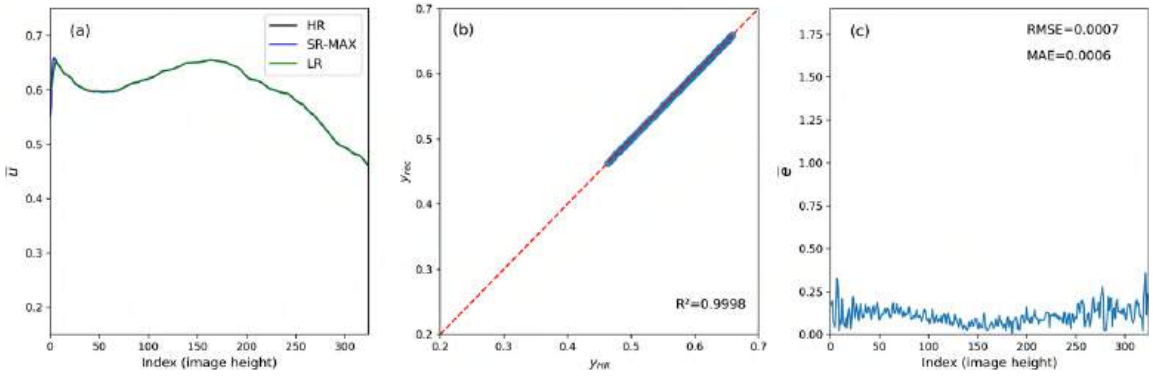


Figure A.56: SR-MAX - Cross-section 5: LR4 validation dataset.

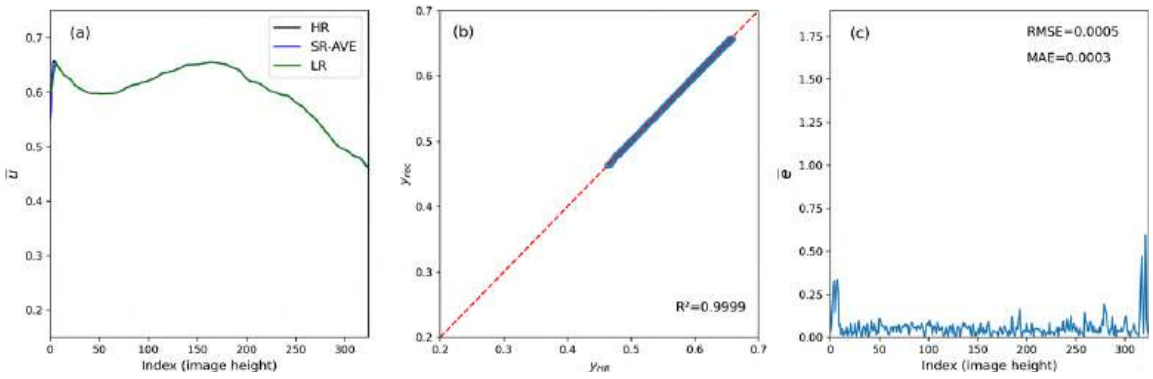


Figure A.57: SR-AVE - Cross-section 5: LR4 validation dataset.