Accelerating and Enhancing Turbulent Flow Analysis Using Deep Learning Techniques

Javad Mortazavian
Mechanical Engineering Department
Cleveland State University
Cleveland, Ohio USA
s.mortazaviannajafabadi@vikes.csuohio.edu

Christopher J Abel
Computer Science Department
Cleveland State University
Cleveland, Ohio USA
c.j.abel@csuohio.edu

***Abstract*—Turbulent flow simulations are critical for understanding complex fluid dynamics in both engineering and environmental applications. However, the high computational cost associated with high-fidelity methods such as Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) poses significant challenges for real-time analysis and design. This project explores the use of deep learning (DL) models to accelerate and enhance turbulent flow analysis by replacing or augmenting traditional simulation methods. Specifically, we develop and evaluate advanced neural architectures—such as Transformers, ConvLSTMs, and improved SRCNN and FlowFormer-based networks—for two primary tasks: (1) temporal forecasting of turbulent flow evolution, and (2) spatial super-resolution of coarse flow fields. Our approach utilizes data from canonical isotropic turbulence databases, with models trained to predict future velocity and pressure fields or reconstruct high-resolution features from downsampled inputs. Additionally, we investigate the generalization capability of our models across different flow regimes and Reynolds numbers. Results of the current study demonstrate that the proposed DL models can reduce simulation times by orders of magnitude while maintaining strong agreement with DNS/LES data. The outcomes of this project have the potential to transform the way turbulence is analyzed and predicted, making high-fidelity insights accessible for real-time applications such as urban airflow forecasting, aerodynamic design, and climate modeling. This research represents a step toward integrating deep learning with computational fluid dynamics to develop hybrid physics-ML frameworks that are both accurate and computationally efficient.**

Keywords—Turbulent Flow; Deep Learning; Spatiotemporal Forecasting; Data-Driven Simulation

I. INTRODUCTION

Given the computational cost of high-fidelity Computational Fluid Dynamics (CFD) approaches—such as Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES)—in capturing instantaneous velocity fluctuations near high-rise buildings, and the limited accuracy of low-fidelity methods like Reynolds-Averaged Navier–Stokes (RANS) simulations, there is a critical need for surrogate models that can effectively balance accuracy and computational efficiency.

Turbulent flows are inherently multi-scale in space and time, exhibiting high-dimensional characteristics with intermittent rotational and translational coherent structures. These complexities present a valuable opportunity for deep neural networks (DNNs) to enhance turbulence modeling and analysis [1] [2].

II. PROBLEM STATEMENT

Direct Numerical Simulation (DNS) of turbulent flows provides highly detailed and accurate representations of fluid dynamics by resolving all relevant spatiotemporal scales. However, the computational cost associated with generating and storing large-scale DNS datasets is extremely high, often limiting their use in real-time prediction and analysis tasks. This computational burden poses a significant challenge for advancing fundamental research and practical applications in turbulence modeling, control, and forecasting.

To address this challenge, our project explores the application of deep learning (DL) models to leverage the wealth of information embedded in large DNS databases. Specifically, we aim to develop efficient and scalable DL-based frameworks for two key tasks: (1) estimating the temporal evolution of turbulence, and (2) performing super-resolution reconstruction of flow fields. These models are intended to serve as cost-effective surrogates that maintain high physical accuracy while dramatically reducing the need for expensive simulations.

By doing so, we seek to accelerate data-driven turbulence analysis, enable real-time forecasting, and facilitate high-resolution flow field recovery, thereby expanding the usability of DNS data in both research and engineering applications.

III. LITERATURE REVIEW

The integration of deep learning into optical flow estimation began with **FlowNet** [3], the first convolutional neural network (CNN)-based model for this task. FlowNet introduced two variants—FlowNetS and FlowNetC—capable of predicting dense motion fields directly from image pairs. While it demonstrated the feasibility of end-to-end optical flow learning, it struggled with fine-scale motion and generalization.

To address these limitations, **FlowNet2** [4] stacked multiple FlowNet modules, incorporating warping and refinement steps. This led to significant accuracy gains but at the expense of

increased computational complexity. In parallel, **SPyNet** [5] proposed a more efficient alternative by employing a coarse-to-fine spatial pyramid approach, greatly reducing memory usage and model size while maintaining reasonable accuracy.

Further advancements came with **PWC-Net** [6] and **LiteFlowNet** [7], which adopted pyramidal processing combined with feature warping and cost volume computation. These models offered an excellent balance between accuracy and efficiency and quickly became benchmarks in optical flow estimation.

A transformative shift occurred with the introduction of **RAFT** [8], a recurrent architecture using all-pairs correlation and iterative refinement via Conv-GRU units. RAFT achieved state-of-the-art performance on standard benchmarks and set a new bar for robustness and accuracy. Its success led to domain-specific adaptations such as **PIV-RAFT** and **RAFT-PIV** [9], which enhanced velocity field estimation for Particle Image Velocimetry (PIV), significantly improving prediction fidelity in fluid flow datasets.

To enhance robustness under noisy conditions, **CC-FCN** [10] integrated traditional cross-correlation operations within a fully convolutional network, offering improved resilience in challenging flow estimation scenarios common in experimental setups.

Most recently, **FlowFormer** [11] introduced a Transformer-based architecture that leverages global attention to capture long-range dependencies in motion fields. Unlike CNN and RNN predecessors, FlowFormer excels at modeling complex global interactions, making it highly effective in tasks such as turbulence forecasting, velocity field reconstruction, and PIV data analysis. Although it demands greater memory and computational resources, its performance in capturing unsteady, high-dimensional flow structures surpasses previous models, establishing it as the current state-of-the-art.

In summary, the evolution from early CNN-based models to recurrent and Transformer-based architectures reflects a trajectory of increasing expressiveness and accuracy. Among them, FlowFormer stands out as the most capable model to date for spatiotemporal flow estimation in complex turbulent environments.

The authors of [12] have provided a comprehensive survey of deep neural networks for image super-resolution. The survey covers the network architecture, types of upsampling stages, training strategies, and loss metrics.

One of the earliest and most well-known applications of CNNs to image super resolution was the SRCNN [13]. This network structure begins with a standard operation to upsample or interpolate a low-resolution (LR) image to a high-resolution (HR) image, then follows with a three-stage CNN to refine the HR image. Because the CNN only needed to refine the coarse HR image, the SRCNN could be relatively simple and easy to train. In [14], the same authors investigated a derivative of the SRCNN, called the FSRCNN, in which the upsampling stage followed the CNN stages. Since these stages operate on images with reduced $H \times W$, the number of computations involved in training and evaluating the CNN are substantially reduced. This has made the FSRCNN an even more popular architecture for image super-resolution.

Other work, such as the LapSRN detailed in [15], has utilized multiple cascades of convolutional and upsampling stages to generate an HR image from an LR image in multiple steps – gradually building up the fine details of the HR image.

Just as in other deep-learning applications, image super-resolution has benefitted from the concept of residual learning introduced with ResNet [16]. When using residual learning in SR applications, the convolutional layers need only train on the residuals between coarse HR images and the ground-truth HR images. The Deep Recursive Residual Network (DRRN) [17] uses both a global residual connection between an upsampled LR image and the output, and local residual connections within a deep CNN consisting of many convolutional layers.

The development of multi-path or multi-branch networks [18] have inspired similar developments in super-resolution applications. For example, the DSRN described in [19] uses parallel paths of data – one consisting of features extracted from the LR image, and a second from a preliminary upsampled HR image – to construct the final super-resolved image.

For our work with super-resolving images derived from fluid turbulence data, we have worked with networks similar to the SRCNN, and also experiment with global residual learning.

IV. Database Configuration

The turbulent flow datasets that we have used in this work were obtained from the Johns Hopkins Turbulence Database (JHTDB) [20], [21], [22]. The JHTDB has several different datasets, each representing DNS simulations of the Navier-Stokes (NS) equations, subject to different sets of boundary conditions, turbulence forcing functions, and fluid Reynolds numbers.

In this study, we focused on the Forced Isotropic 10243 dataset [23]. In the simulations that generated this dataset, the NS equations are solved

- on a grid consisting of 1024 equally-spaced points along each of the three spatial dimensions (x, y, z)
- with a simulation time step = 200μs; the data is sampled in $\Delta t = 2ms$ and stored in the database

At each (x, y, z, t) point, the database includes simulated

- Pressure (p)
- 3 Pressure gradients $\left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right)$
- 3 Components of Velocity $\vec{u} = (u_x, u_y, u_z)$
- 9 Velocity gradients $\left(\frac{\partial u_x}{\partial x}, \frac{\partial u_x}{\partial y}, \frac{\partial u_x}{\partial z}, \dots\right)$
- Hessians (second derivatives) of pressure and each of the velocity components

Note that the JHTDB stores the simulated results at each time point in arrays of 643 spatial points, compressing them

into a Zarr format for efficient storage and transmission [20]. An example of the velocity data is shown in Fig. 1.

	x	y	z	ux	uy	uz
0	0.0	-1.000000	4.712389	1.769139e-18	0.000000	-5.729756e-19
1	0.0	-0.866667	4.712389	8.825606e-01	-0.019833	5.115691e-02
2	0.0	-0.733333	4.712389	9.747395e-01	0.002217	8.462451e-02
3	0.0	-0.600000	4.712389	1.002418e+00	0.037070	9.942410e-02
4	0.0	-0.466667	4.712389	1.096980e+00	-0.012544	5.457124e-02

Fig. 1. Raw Data from JHTDB

To make use of this data to train and test our CNNs for temporal evolution and super-resolution, we downloaded temporal snapshots of the data using the ***givernylocal*** Python library, for example

```
result = getData(dataset, 'velocity',
   1.0, 'none', 'lag4', 'field', points)
ux = np.array(result[0])[:, 0]
   .reshape(128,128)
```

and then converted the data to images, using the ***seismic*** color map, which maps a specified minimum data value to a blue pixel, and a maximum value to a red pixel.

```
plt.imsave('image.png',ux,cmap='seismic',
      vmin=-3,vmax=3)
```

For our training and testing sets, we downloaded data slices sampled at 128 points across both the x and y axes, with a constant value of $z = \pi$, at increments of $\Delta t = 2ms$.

V. Methodology

A. Turbulence Temporal Evolution

We investigate the temporal evolution of turbulence by leveraging data-driven models trained on high-resolution Direct Numerical Simulation (DNS) datasets. The objective is to predict the future states of turbulent flow fields over short time horizons using learned spatiotemporal patterns. Specifically, the model is trained to forecast flow dynamics from $t = 0$ to $t = 8$ seconds and evaluated on its ability to generalize from $t = 8$ to $t = 10$ seconds. This temporal split allows us to test the model's capability in extrapolating beyond the training window.

To further understand the robustness and generalizability of the models, we examine multiple configurations of the training dataset, including variations in:

- **Database size**: Testing how different volumes of training data affect predictive performance.
- **Spatial planes**: Evaluating model performance across different cross-sectional planes—namely *x*–*y*, *x*–*z*, and *y*–*z*—to assess directional consistency.
- **Hyperparameters**: Investigating the effects of training duration (number of epochs) and optimization dynamics (learning rates) on convergence and accuracy.

Our approach employs a modular deep learning pipeline inspired by the original FlowFormer model [11] composed of two key components: (1) a feature extraction module and (2) a spatiotemporal prediction module. The proposed FluidFlowFormer (F^3) model architecture has been shown in the Fig. 1. The original structure of the FlowFormer model has been majorly revised to be adopted to the task of temporal evolution of turbulence prediction as follows: the patchification section has been reorganized as follows (……). The spatiotemporal prediction module is revised completely and a ConvLSTM [24] module is replaced with the original GRU because ConvLSTM preserves spatial structure through convolutional operations, making it ideal for predicting pixel-wise turbulent flow fields, unlike GRU which flattens spatial data. It integrates seamlessly with CNN-based encoders and is more parameter-efficient for high-dimensional inputs. Empirical studies show ConvLSTM outperforms GRU in spatiotemporal vision tasks like video and fluid flow forecasting.

1. **Feature Extraction Module**: This component captures spatial structures and relevant physical features in turbulent fields. Different architectures are explored, including convolutional neural networks (CNNs) and attention-based encoders, to extract compact yet informative representations of input frames.
2. **Spatiotemporal Prediction Module**: This component is responsible for learning temporal dependencies across sequential frames. We evaluate the efficacy of various recurrent architectures such as Convolutional Long Short-Term Memory (ConvLSTM) networks and Gated Recurrent Units (GRUs). These architectures are specifically chosen for their ability to preserve spatiotemporal coherence in fluid flow evolution.

The **encoder** consists of a hierarchical stack of residual convolutional blocks that extract multi-scale spatial features from input turbulent flow frames. It includes three stages: the first maps 3-channel RGB inputs to 64 channels using a 7×7 kernel with stride 2; the second increases to 128 channels with a 5×5 kernel and further downsamples; the third outputs a 256-dimensional embedding using a 3×3 kernel. Each block incorporates batch normalization and ReLU activations. The output is then passed through 1×1 convolutions to generate key and value feature maps for attention. Skip connections from all three stages are preserved for later use in the decoder to support high-resolution reconstruction.

The **decoder** integrates attention, ConvLSTM recurrence, and transformer processing to forecast the next frame. It starts with a spatial attention block that enhances the input query using encoder-derived key and value maps. The resulting features, concatenated with the query, pass through a ConvLSTM cell to capture temporal dynamics. This output is refined by a spatial transformer block with multi-head self-attention. A convolutional head predicts the flow increment. The decoder then upsamples the hidden state in three stages, each merging with encoder skip features and refining via convolutional layers. The final output is an RGB image representing the predicted future flow field.

B. *Loss Function Formulation for Turbulence Forecasting*

The model is trained using supervised learning with ground-truth DNS frames, and its performance is evaluated using a combination of three loss functions, each capturing a different aspect of prediction quality.

1) End-Point Error (EPE)

The End-Point Error (EPE) quantifies pixel-level motion accuracy and is widely used in optical flow estimation. It computes the average Euclidean distance between the predicted and ground-truth flow vectors:

$$EPE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(u_{p,i} - u_{gt,i})^2 + (v_{p,i} - v_{gt,i})^2} \quad (1)$$

where $(u_{p,i},\ v_{p,i})$ and $(u_{gt,i},\ v_{gt,i})$ are the predicted and ground-truth optical flow components at pixel i, and N is the total number of pixels.

2) Perceptual Loss

The Perceptual Loss measures differences in high-level structural and textural features using a pretrained deep network (e.g., VGG19). It is defined as:

$$L_{perceptual} = \|\phi(\hat{y}) - \phi(y)\|_2^2 \quad (2)$$

where $\hat{y}$ and y are the predicted and ground-truth images, and $\phi(\cdot)$ denotes the extracted feature representations. This loss emphasizes perceptual quality and structural similarity beyond pixel-wise matching.

3) Gradient Loss

The Gradient Loss promotes spatial smoothness and edge consistency by comparing gradients along spatial directions:

$$L_{grad} = \|\nabla_x(\hat{y}) - \nabla_x(y)\|_1 + \|\nabla_y(\hat{y}) - \nabla_y(y)\|_1 \quad (3)$$

where ∇_x and ∇_y denote gradients in the x and y directions, respectively.

4) Hybrid Loss Function

To capture both numerical precision and perceptual fidelity, we define a hybrid loss function as a weighted sum of these components:

$$loss = \alpha \times EPE + \beta \times L_{perceptual} + \gamma \times L_{grad} \quad (4)$$

where α, β, and γ are tunable weights. This formulation enables the model to balance accurate motion prediction with visually coherent and structurally smooth flow field generation.

This systematic exploration provides insights into the optimal configuration for accurate and computationally efficient turbulence forecasting.

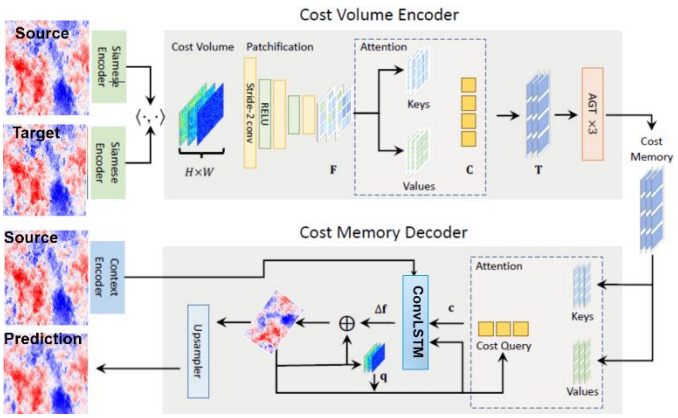


Fig. 2. The proposed FluidFlowFormer (F^3) model for forecasting the temporal evolution of fluid flow (Adopted and modified from [11]).

C. *Super-Resolution Reconstruction*

The super-resolution task was performed with generalizations of the SRCNN network introduced in [13]. **Error! Reference source not found.** shows the structure and p arameters of the network.

The generalized SRCNN network consists of 4 key stages:

- An **upsampling** stage, which generates a raw 128 x 128 image (high-resolution, or HR) from a low-resolution (LR) 32 x 32 image. Two different methods were tested:
  - Upsample with bicubic interpolation.
  - Transpose Convolution, with a kernel = 4 and stride = 4.
- **Stage 0**: Convolution + ReLU, with 4 channels, filter size $f0 = 3$, stride = 1, and padding = 1. This stage is only included when ConvTranspose2d is used as the upsampling stage.
- **Stage 1**: Convolution + Batch Normalization + ReLU, $\boldsymbol{n_1}$ channels, and filter size=$\boldsymbol{f_1}$ (variable).
- **Stage 2**: Convolution + Batch Normalization + ReLU, $\boldsymbol{n_2}$ channels, and filter size=$\boldsymbol{f_2}$ (variable).
- **Stage 3**: Convolution, 4 channels, filter size $\boldsymbol{f_3}$.

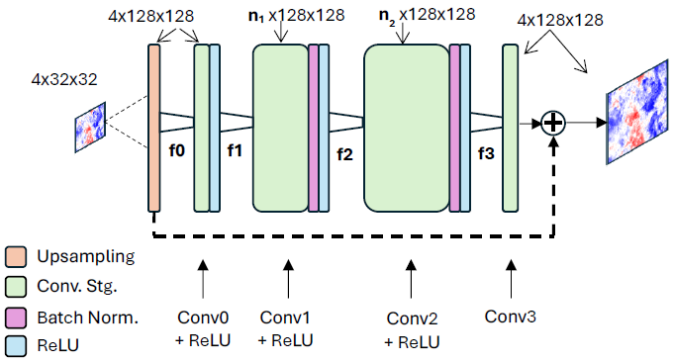


Fig. 3. Generalized SRCNN Network

We also considered a residual-learning option – adding the output tensor of the initial upsampling stage to the tensor output from convolution stage 3 [25]. In this configuration, the forward convolutional path encodes the residue between the upsampled HR image and the ground-truth full-scale image. Our approach is essentially identical to the approach described in [25], [17], and [26].

For all convolution stages, padding is varied along with the filter size to maintain a per-stage output tensor $H \times W = 128 \times 128$.

In our experiments with the Generalized SRCNN, several architectural parameters and hyperparamteers were varied:

- Comparison of the two upsampling stages.
- The use of batch normalization stages after convolution stages 1 and 2.
- Variations in batch size.
- With vs. without the residual-learning option [25]

D. *Multi-Branch SRCNN Network*

In addition to variations on the standard SRCNN network and the residual-learning SRCNN, we have considered a multi-path variant [27] [28], as shown in Fig. 4.

The underlying physics of turbulent fluid flow, as quantified by the Navier-Stokes equations, indicate that at any point in space, the fluid pressure and velocity – along with the spatial gradients of each – are related.

Our experiments with the network shown in Fig. 4 attempt to take advantage of this relationship to combine LR spatial samples of two physical variables to improve the accuracy of super-resolved HR image of one of the variables:

- The dual-path combination of $4 \times 32 \times 32$ images of u_x and p, captured at the same point in time and space, to produce an HR $4 \times 128 \times 128$ image of u_x.

One other variant seems to be valuable, but was not evaluated in this work:

- The triple-path combination of $4 \times 32 \times 32$ images of p, $\frac{\partial p}{\partial x}$, and $\frac{\partial p}{\partial y}$ to produce an HR image of p.

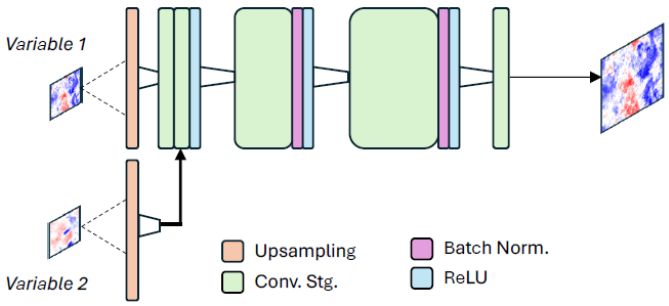


Fig. 4. Multi-Branch Variant of SRCNN

VI. Results and Discussion

A. *Turbulence Temporal Evolution*

The model is designed to predict the future state of a turbulent flow field based on a sequence of past frames. It takes as input a temporal sequence of 10 RGB images representing turbulent flow, sampled at regular intervals. Each input sequence has the shape $[B, T, C, H, W]$, where B is the batch size, $\mathrm{T} = 10$ is the number of input frames, $\mathrm{C} = 3$ is the number of color channels, and $\mathrm{H}, \mathrm{W} = 256$, denote the spatial resolution. The model is trained to forecast the next frame after a temporal gap of 0.2 seconds, which corresponds to 100 simulation frames ahead in the dataset. The output is a single RGB image with shape $[\mathrm{B}, 3, \mathrm{H}, \mathrm{W}]$, representing the predicted future turbulent field.

To capture temporal dependencies across the input sequence, the model employs a ConvLSTM module that operates directly on spatial feature maps. Unlike traditional LSTMs, ConvLSTM preserves spatial structure while modeling temporal evolution, making it well-suited for turbulent flow data. During inference, the hidden and cell states of the ConvLSTM are propagated through each input frame in the sequence, enabling the model to accumulate temporal context and refine its prediction of the future frame.

To preserve fine-grained spatial details, the model incorporates skip connections from the encoder at multiple scales into the decoder. These features are fused during the upsampling process, allowing the decoder to recover high-resolution details lost during downsampling. The decoder performs multi-stage upsampling in three steps, progressively increasing the resolution through bilinear interpolation, followed by convolutional layers for feature refinement. This strategy ensures that the final predicted frame is both spatially accurate and visually coherent.

A hybrid loss function is employed to guide the training process by balancing perceptual accuracy, structural similarity, and edge sharpness. It combines Perceptual $+ L1$ loss (using pretrained $VGG19$ features), $SSIM$ loss to preserve image structure, and gradient loss to enhance edge details. The loss components are weighted by coefficients $\alpha = 0.6$, $\beta = 0.2$, and $\gamma = 0.2$, respectively. These weights were studied and optimized to ensure stable training and high-fidelity predictions of turbulent flow frames.

The model is trained using the Adam optimizer with a learning rate of 5×10^{-4} and a batch size of 6 over 30 epochs. All input and output images are resized to a resolution of 256×256 to ensure consistent spatial dimensions. The dataset is organized into training and testing sets, where each input sample consists of a sequence of 10 consecutive frames, and the target is a future frame 100 time steps ahead (representing a 0.2-second gap). This design enables the model to learn both spatial and temporal dynamics effectively within a supervised learning framework.

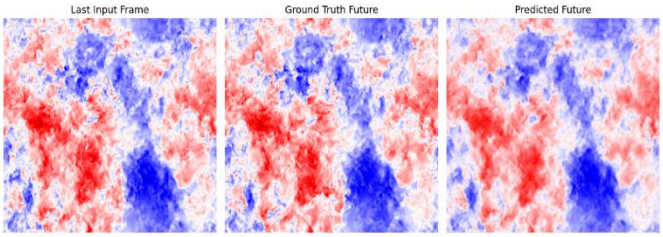


Fig. 5. Predicted frame 0.2 seconds ahead, corresponding to 100 timesteps into the future

The performance of the proposed model is visually illustrated in Fig.4, which compares the last input frame, the ground truth future frame, and the model's predicted future frame. As shown, the predicted frame closely resembles the ground truth in terms of both large-scale structures and finer turbulent features. The model successfully captures key flow patterns, including the spatial distribution and intensity of red and blue regions, which likely represent vorticity or velocity fluctuations. Despite some minor smoothing effects, the predicted field maintains a high level of **structural** and **visual** similarity, demonstrating the model's ability to accurately forecast the temporal evolution of complex turbulent flow.

B. Super-Resolution Training and Testing Results

For the super-resolution experimental results described in this section, three sets of images (p, u_x, and u_y) were downloaded from the JHTDB using the procedure described previously. Each image consists of a slice at a single time point across the $x-y$ plane, with 128 points along each spatial dimension. A 128×128 image was captured every $\Delta t = 2ms$ of Navier-Stokes simulation time from $t = 0$ to $t = 5s$. Thus, 2501 images for each of the three primary fluid-flow variables were generated from JHTDB queries.

Each of these three datasets was split into a training set of 2251 images, and 250 testing images, with a test image taken every $\Delta t = 20ms$ in order to avoid having the test images concentrated at the end of the fluid-flow simulation time window.

With one exception, all of the networks were trained over 200 epochs, with an image batch size of 8. Data augmentation – random horizontal and vertical flips, and rotations of 90°, 180°, and 270° -- was applied to the final variant, without any noticeable improvement in trained loss.

The parametric details of the variants of the SRCNN network that we tested are summarized in **Error! Reference s ource not found.**. Five variants were considered:

- One variant (1b) used a ConvTranspose2d layer, followed by a convolutional layer with filter size $f_0 = 3$. The use of these two additional trainable layers was found not to have made any improvement in trained MSE loss performance, so the remaining 4 variants used `Upsample` with bicubic interpolation.
- Four of the five variants used the standard filter sizes $(f_1, f_2, f_3) = (9, 1, 5)$ that were used in the original SRCNN network [13]. The fifth (1d), increased the filter size of the second convolutional stage to 5; however, this did not make any noticeable improvement in MSE loss (this differs from the results in [13]).
- All but the first variant (1a) included batch normalization stages after the first and second convolutional stages.

TABLE I. ARCHITECTURE OF SRCNN VARIANTS

Variant	Upsamp Stage	Filter Sizes (f_1-f_2-f_3)	Stage Chan. (n_1-n_2)	Batch Norm?	Resid. Learn?
1a	Upsample bicubic	9-1-5	64-32	No	No
1b	Conv Transpose 2d, $f_0 = 3$			Yes	
1c	Upsample bicubic				
1d		9-5-5	64-32		
1e		9-1-5	64-32		Yes

The best MSE loss performance was for variant 1e, which included the residual learning path. Fig. 6 compares the loss vs. epoch for each of the variants 1a – 1e in TABLE I.

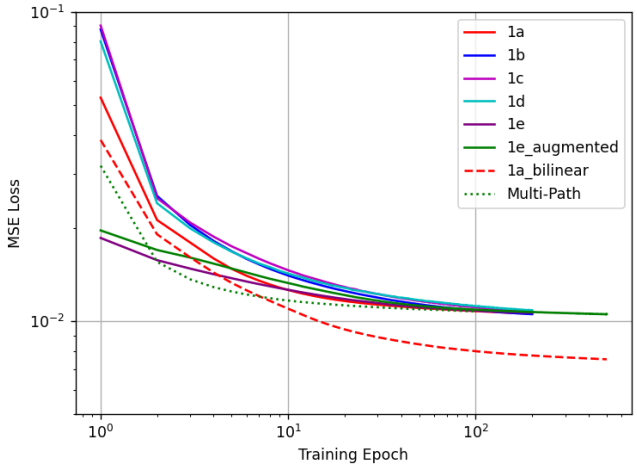
Performance comparisons for each of the SRCNN variants on the testing set can be made using the PSNR (Peak Signal-to-Noise Ratio) metric, as defined in [12]:

$$PSNR = 10 \log_{10}\left(\frac{L^2}{\frac{1}{N}\sum(I - I_{gt})^2}\right) \quad (5)$$

where L is the maximum pixel value, and $\frac{1}{N}\sum(I - I_{gt})^2$ is the pixel-level MSE. In our case, pixel values were in the range $[0, 1]$, thus $L = 1$. As a result, PSNR is essentially the logarithm of the inverse of the MSE.

The results of average PSNR for the entire testing set for each trained variant in TABLE I. is shown in TABLE II.

Fig. 6. Loss vs. Training Epoch for SRCNN Variants



Note that TABLE II. shows that **we observed ~1.5dB improvement with network variant 1a** when the LR image input to the network was generated from the ground-truth image by using Resize with bilinear interpolation, rather than nearest-neighbor interpolation (the loss vs. training epoch is shown in **Error! Reference source not found.** as a red dashed l ine with the label '1a_bilinear'). Unfortunately, this improvement only holds true when the bilinear interpolation is also applied to generate the LR test image. Since this operation would not be possible if we were applying the network to an image obtained from the JHTDB with true 32x32 spatial sampling, we have not considered this improvement further.

As is clearly evident, none of the variations, including data augmentation, made any substantial difference in performance of the trained network.

TABLE II. PSNR FOR UX TESTING SET FOR EACH TRAINED NETWORK

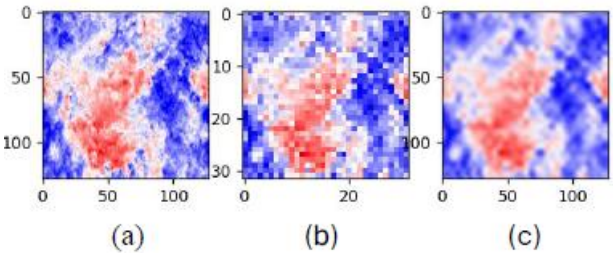
Variant	Description	PSNR (dB)
SRCNN Variants		
1a	SRCNN w/ 9-1-5	19.76
1b	1a, with ConvTranspose upsamp.	19.79
1c	1a, with batch norm.	19.73
1d	SRCNN w/ 9-5-5	19.71
1e	SRCNN w/ Residual Learning Path	19.73
1e - augmented	1e w/ Data augmentation	19.78
1a	1a with **LR image derived from GT image by resize w/ bilinear interp.**	**21.23**
Multi-Path Variant		
Multi-Path	Fig. 3 with primary = ux and secondary = p	19.68

A comparison of a ground-truth 128x128 image, a scaled 32x32 equivalent image, and the scaled image super-resolved to 128 x 128 by the SRCNN with residual learning (variant 1e) is shown in Fig. 7.

The PSNR results of a trained version of the dual-path network shown in **Error! Reference source not found.** are a lso shown in **Error! Reference source not found.**, with the label "Multi-Path." The convolutional stages had filter sizes of $f_1 = 9$, $f_2 = 1$, and $f_3 = 5$, while the number of channels in stages 1 and 2 were doubled (i.e. $n_1 = 128$ and $n_2 = 64$) to account for the fact that two 4-channel images have been concatenated at the input.

Unfortunately, just as for the simple SRCNN variants, the addition of pressure as a secondary variable did not improve the PSNR of the super-resolved ux image.

Fig. 7. Image Comparison - Ground-Truth 128x128 (a), Scaled 32x32 (b), Super-Resolved (c)



VII. SUMMARY AND CONCLUSIONS

This work presents a deep learning-based framework for *temporal evolution forecasting***:** predicting the future evolution of turbulent flow fields using a hybrid architecture that integrates convolutional encoders, ConvLSTM for temporal modeling, attention mechanisms, and transformer-based refinement. The model is trained with a hybrid loss function that combines perceptual, SSIM, and gradient terms, enabling it to preserve both global structures and fine details in the predicted flow. Qualitative results show strong agreement between predicted and ground truth frames, confirming the model's effectiveness in capturing complex spatiotemporal dynamics.

For future work, this model can be extended to handle longer temporal forecasts and more diverse flow regimes. Incorporating physics-based constraints or coupling with reduced-order models may further improve generalization. Additionally, exploring transformer-based temporal modules or self-supervised pretraining could enhance the model's ability to learn from limited labeled data, making it suitable for broader applications in data-driven turbulence modeling and real-time flow prediction.

We also investigated the application of variants of the well-known SRCNN network [13] for *super-resolution reconstruction* of images derived from turbulent fluid flow: increasing the resolution of these images from an LR 32x32 grid to an HR 128x128 grid.

- We considered the original SRCNN, with filter sizes $f_1 - f_2 - f_3 = 9 - 1 - 5$, as well as $9 - 5 - 5$.
- For the pre-upsampling stage, we compared the use of standard bicubic interpolation with a learnable ConvTranspose2d layer.
- We investigated a variant of SRCNN that used a global residual learning path to add the pre-upsized image to the output of the final convolutional layer.
- We examined the effect of batch normalization and image augmentation on the final results.

- We also investigated a minor variant in which the LR image was derived from the ground-truth HR image by Resize with bilinear interpolation. For reasons that we have not identified, this produced approximately 1.5 dB higher PSNR on the testing set.

The results of the super-resolution experiments performed on images from the isotropic turbulence database fell far short of expectations.

- The PSNR using the base (9-1-5) SRCNN network was 8dB below what was reported by the authors of the original paper [13] for a 4x image upscaling.
- The authors in [13] reported a 0.5dB improvement in PSNR when increasing the filter size of the second convolutional stage from 1 to 5; unfortunately, in our work this improvement did not materialize.
- Surprisingly, the addition of a global residual learning path (variant 1e in **Error! Reference source not found.**) showed no statistically significant improvement in PSNR for our dataset.

One possible explanation for the discrepancy between our results and those of other authors is that the images in our training set are not sufficiently differentiated to adequately train the CNNs. Future work could consider beginning with a network trained on a large dataset consisting of a richer variety of color images and then refining the training state with a limited exposure to a dataset consisting of turbulence images.

VIII. References

[1] J. N. Kutz, "Deep learning in fluid dynamics," *J. Fluid Mech.*, vol. 814, pp. 1-4, March 2017.

[2] J. Ling, A. Kurzawski and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *J. Fluid Mech.*, vol. 807, pp. 155-166, 2016.

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision (ICCV)*, 2015.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] D. Sun, X. Yang, M. Y. Liu and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] T. W. Hui, X. Tang and C. C. Loy, "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Glasgow, 2020.

[9] C. Lagemann, K. Lagemann, S. Mukherjee and W. Schroder, "Deep recurrent optical flow learning for particle image velocimetry data," *Nature Machine Intelligence*, vol. 3, no. 7, pp. 641-651, July 2021.

[10] Q. Gao, Q. Gao, H. Lin, H. Tu, H. Zhu, R. Wei, G. Zhang and X. Shao, "A robust single-pixel particle image velocimetry based on fully convolutional networks with cross-correlation embedded," *Physics of Fluids*, vol. 33, no. 12, 2021.

[11] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai and H. Li, "FlowFormer: A transformer architecture for optical flow," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 668-685, 2022.

[12] Z. Wang, J. Chen and S. C. Hoi, "Deep Learning for Image Super-Resolution: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3365-3387, 1 Oct 2021.

[13] C. Dong, C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295-307, 1 Feb 2016.

[14] C. Dong, C. C. Loy and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," 2016. [Online]. Available: https://arxiv.org/abs/1608.00367.

[15] W. S. Lai, J. B. Huang, N. Ahuja and M. H. Yang, "Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, 2017.

[16] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[17] Y. Tai, J. Yang and X. Liu, "Image Super-Resolution via Deep Recursive Residual Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, 2017.

[18] X. Xu, W. Li, Q. Ran, Q. Du, L. Gao and B. Zhang, "Multisource Remote Sensing Data Classification Based on Convolutional Neural Network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 937-949, 2017.

[19] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock and T. S. Huang, "Image Super-Resolution via Dual-State Recurrent Networks," in *2018 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, Salt Lake City, 2018.

[20] Johns Hopkins University, "Johns Hopkins Turbulence Database (JHTDB)," [Online]. Available: http://turbulence.idies.jhu.edu.

[21] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," *Journal of Turbulence*, vol. 9, pp. 1 - 29, 2008.

[22] E. Perlman, R. Burns, Y. Li and C. Menevau, "Data Exploration of Turbulence Simulations using a Database Cluster," in *Proceedings of the 2007 ACM / IEEE Conference on Supercomputing (SC '07)*, 2007.

[23] M. Wan, S. Chen, G. Eyink, C. Meneveau, E. Perlman, R. Burns, Y. Li, A. Szalay, P. Johnson and S. Hamilton, "Forced Isotropic Turbulence Data Set (Extended)," [Online]. Available: https://turbulence.idies.jhu.edu/docs/isotropic/README-isotropic.pdf.

[24] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 29th International Conference on Neural Information Processing Systems*, 2015.

[25] J. Kim, K. Lee and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[26] Z. Hui, X. Wang and X. Gao, "Fast and Accurate Single Image Super-Resolution via Information Distillation Network," in *2018 IEEE / CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[27] S. Dargahi, A. Aghagolzadeh and M. Ezoji, "Single Image Super Resolution Using Multi-path Convolutional Neural Network," in *2021 5th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, 2021.

[28] C. H. Liu, T. H. Hsieh, K. Y. Huang and P. Y. Chen, "TMSR: Tiny Multi-path CNNs for Super Resolution," in *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering (ECICE)*, Yunlin, 2023.