Finding Hidden Patterns in High Resolution Wind Flow Model Simulations

Yuqiao Zhao[1], Yulin Wei[2] Ming Chi Wong[3], Anna Fortenberry[4], and Spencer Smith[5]

[1] City University of Hong Kong
zhyuqiao@gmail.com,
[2] The Chinese University of Hong Kong
yulinwei96@gmail.com
[3] City University of Hong Kong
wilson89006114@gmail.com
[4] University of North Texas
AnnaFortenberry@my.unt.edu
[5] University of North Texas
SpencerSmith4@my.unt.edu

Abstract. The optimized designs of wind farms are dependent on accurate microscale wind flow dynamic data. Large Eddy Simulations (LES) are used to simulate atmospheric properties at a microscale level when driven by boundary conditions from ERA5. High resolution simulations can be computationally expensive and require large amounts of storage, so a method to upscale a low resolution output of LES data is sought after. Two promising approaches have been presented in a hierarchical autoencoder [5] and a GAN model [6]. This paper explores three additional approaches to this problem: an integrated 2D and 3D CNN, an interpolation model, and a ResUnet model. The integrated CNN demonstrates the most promising results but is computationally expensive. Interpolation falls second to CNN in accuracy but produces fast, cheap results. The U-Net shows promise but is too computationally expensive to train with limited resources.

Keywords: wind flow dynamics, LES, ERA5, PCA, t-SNE, 3DCNN, interpolation, ResUnet

1 Introduction

Accurate microscale wind flow dynamic data is essential to the design of wind farms. Wind flow dynamics can be highly sensitive to terrain irregularities, and wind conditions may drastically change from one location to another over small distances [1]. Computational Fluid Dynamics (CFD) offers an approach for accurately assessing atmospheric flow properties with a mathematical model known as Large Eddy Simulation (LES). It is able to simulate turbulence at a reasonable cost. Boundary conditions derived from ERA5 data, a global weather model with a resolution around thirty kilometers, drives the LES model. LES is able downscale the wind field to a resolution around ten to one hundred meters, outputting detailed data on microscale wind flow dynamics in an area. LES datasets are obtained by running the simulation over the course of a year in increments of ten minutes. As a consequence, high resolution simulations require a storage size of several hundred gigabytes. To resolve this issue, a method to first generate a low dimensional LES dataset and then accurately upscale it to a high dimension is desired. This paper tests three methods for solving the upscaling problem: an integrated 2D and 3D CNN, an interpolation model for super resolution, and a ResUnet.

This problem is solved in three parts: exploratory data analysis and visualization, dimensionality reduction of the grid, and upscaling from a low resolution to high resolution grid. The rest of this paper is as follows. A background section presents related works and preliminary information for understanding our solutions. The designs of our three upscaling methods are discussed in section three. A methodology section details the processes each of the three parts. Results and discussion follow this, and a conclusion highlights the key contributions and future directions.

2 Background

2.1 Related Works

Two significant contributions related to upscaling low resolution data to a higher resolution were presented with this data challenge. The first uses a Generative Adversarial Network (GAN) to super resolve wind velocity and solar irradiance from global output models [6]. Resolution enhancements of 50x and 25x were achieved for wind and solar data, respectively. It was able to learn and preserve features from training datasets, as well as reduce storage space and computational requirements. However, this model does not used for data on a microscale. The second contribution presents an hierarchical autoencoder that can extract nonlinear autoencoder modes of flow fields while preserving the contribution order of latent vectors [5]. It maps high dimensional systems into low dimensional spaces. However, it is stated that a few issues remain with this model and advanced designs are necessary.

This paper contributes to the literature by testing three new approaches, including an integrated 2D and 3D CNN, an interpolation method, and a ResUnet. Two promising approaches to solving problems of this nature exist; we present additional ones for ongoing research efforts. The results signify whether these approaches are considerable options for similar applications in the future.

2.2 Datasets and Dimensionality

The problem statement for this paper was presented as a Data Challenge by the Oak Ridge National Lab [1]. Five datasets were provided from the U.S. Department of Energy, project GELESSMC under Contract DE-AC05-00OR22725.

Three of the datasets are compiled from the entirety of the year of 2020, including an ERA5 dataset, a high resolution LES dataset, and a low resolution LES dataset. The remaining two are from the month of January in 2020 only, making up a high resolution LES sample and low resolution LES sample. The ERA5 model provides hourly estimates of atmospheric variables at a resolution of about thirty kilometers. The data is extracted at a single point, (-7.737°E, 39.7°N), for this study; thus, the output is 1D + time. To make the LES datasets a downloadable size, the data was reduced to hourly timestamps instead of ten-minute timestamps. Furthermore, this data is reduced from a dimensionality of 3D + time to 2D + time by setting a constant height of 100m. This results in 256 x 256 x 1 x 8760 nodes and time steps. Figure 1 demonstrates this dimensionality constraint by portraying a terrain-following slice.

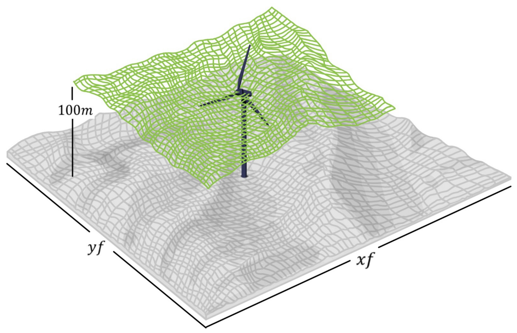

Fig. 1: Visualization of data: terrain-following slice

The high resolution LES dataset is 80m x 80m x 1H frequency and the low is 160m x 160m x 1H frequency. Table 1 lists the variables present in these datasets.

3 Upscaling Methods

3.1 Integrated 2D and 3D CNN

Our first approach was to use an integration of a 2D and 3D CNN, created by a student in an REU program at UTK in 2019. Two models were tested, titled *5-Parallel* and *Delta*. The *5-parallel* model consists of five 2D convolution layers in parallel, as seen in Figure 2. It is concatenated and passed to a 3D convolution stack. A 2D component is responsible for feature extraction, and a 3D component is responsible for super resolution [11]. The output of the 3D stack is then passed to the *conv2d* layer and the *limitoutputlayer*. The latter uses a modified version of sigmoid, shown in equation one. The model is trained with five recurrents for both the 2D and 3D parts. Other hyperparameters include the Adam optimizer set to a 0.00001 learning rate and twenty epochs.

Table 1: Variables of ERA5 and LES

Variables(ERA5)	Variables interpretations
't2m'	2 meter above ground level temperature in K
'u100'	100 meter above ground level U wind component in m/s
'v100'	100 meter above ground level V wind component in m/s
'i10fg'	10 meter above ground level instantaneous wind gust
Variables(LES)	**Variables interpretations**
'temp'	1H average of temperature in Kelvin
'vel'	1H average of horizontal wind speed in m/s
'u'	1H average of U component of wind speed (along 'xf') in m/s
'v'	1H average of V component of wind speed (along 'yf') in m/s
'std'	1H average of standard deviation of horizontal wind speed in m/s
'absolute_height'	Height above sea level in meter, only depends on (xf, yf) not time

$$S(x) = (MAX - MIN) * sigmoid(x/100) + MIN \tag{1}$$

Delta has a similar structure to *5-Parallel*, shown in Figure 3. Instead of processing the video in parallel in the 2D convolution part, Delta lets each frame pass through different numbers of 2D convolution. It is then concatenated to the 3D convolution. This model uses symmetrical frames. Blocks 1 and 2 are responsible for extracting features from a video, and block 3 enlarges the image size [11].

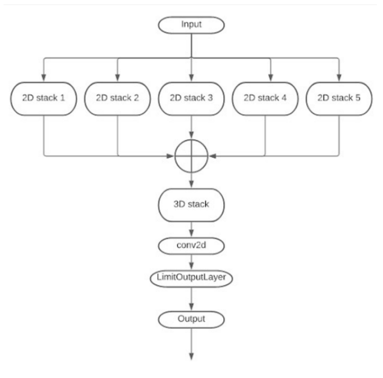

Fig. 2: 5-Parallel Architecture

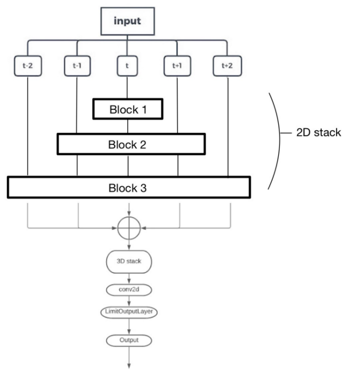

Fig. 3: Delta Architecture

3.2 Interpolation for Super Resolution

We tested interpolation as an approach with the following tutorial [13]. It uses OpenCV in Python. A function called cv2.resize() takes a parameter called interpolation, which can be assigned the following flags: INTER_NEAREST, INTER_LINEAR, and INTER_CUBIC. This allowed us to test nearest-neightbor interpolation, bilinear interpolation, and bicubic interpolation, respectively.

3.3 ResUnet

The deep residual U-Net architecture (ResUnet) [12] is similar to the architecture of U-Net [2], but includes residual units with skip connections. These connections can allow for better model loss minimization and decrease the number of parameters needed.

4 Methodology

4.1 Exploratory Data Analysis and Visualization

Data Pre-processing The objective of part one is to determine whether systematic bias or correlation exists between the ERA5 and high resolution LES datasets and whether these results are dependent on the simulated day or grid position. To allow for comparison, both the ERA5 and LES datasets required preprocessing. The ERA5 dataset contains a time series at one position, while the LES dataset contains a time series on a grid. To force the datasets to have the same dimension, the LES dataset was flattened along *xf*, the longitude, and *yf*, the latitude, as seen in Figure 1. Any missing values were replaced with the mean value of the dataset. Next, zero-mean normalization was applied. For each value in a column, the mean of the column was subtracted from it. Finally, PCA was applied to centralize the information of the full grid and create a new dimension, in which location and time were combined. For ERA5, the dimension of position was first squeezed, since it is for one point locationally. Then, the data was standardized by subtracting the mean of the dataset.

Bias and Correlation Calculation The high resolution LES data set has six variables, while the ERA5 dataset has four, as seen in Table 1. Thus, only the corresponding variables of the datasets were compared for bias and correlation. These include temperature, *t2m* and *temp*, horizontal wind speed, *i10fg* and *vel*, U component of wind speed, *u100* and *u*, and V component of wind speed *v100* and *v*. Equation 2 was applied to each of these sets or corresponding variables for determining bias. Bias was calculated by simply dividing the sum of differences between corresponding datapoints in the ERA5 and LES by the number of datapoints in the ERA dataset. This same process was repeated to find correlation, using the Pearson correlation coefficient [8]. It is useful for detecting the degree of linear correlation between two sets of data.

Detection of Time or Location Dependencies To detect whether the bias and correlation calculations were dependent on the simulated day or grid position, two tests were completed. First, the original high resolution LES dataset was split into four segments based on time. The preprocessing steps and the bias and correlation steps were performed again. Second, the LES dataset was split into four segments along the midpoints of the longitude and latitude. The same preprocessing steps and calculations were performed. Both of these tests produced four outputs, the first set dependent on a segment of time in the ERA5 dataset and the second set dependent on a LES grid location. The outputs within a set were compared with each other for similarity.

4.2 Dimensionality Reduction of the Grid

Analysis of LES Dataset Variables To accomplish part one of the solution, dimensionality reduction was performed on the LES high resolution dataset to allow for comparison to the ERA5 dataset. Part two examines this process in further detail. The goal of dimensionality reduction is to compress the site behavior into a lower dimensional space without losing wind flow model properties. To accomplish this, the variables in the high resolution LES dataset were tested for high correlation. A high correlation between variables can insinuate redundant information. First, the Pearson correlation coefficient was applied to each combination of two variables. After this, the variance [9] of each variable was tested to get the degree of dispersion.

Standard Dimension Reduction Approach The first method we applied to the high resolution dataset was PCA [3], a linear dimensionality reduction method. After standardizing the data, computing the covariance matrix, and performing singular value decomposition, it was determined that the first principal component captured the most information of the data. The data was projected along this component. To compute the percentage of variance, or information, accounted for, the first eigenvalue was divided by the sum of all the eigenvalues.

Deep Learning Dimension Reduction Approach T-SNE [4], a nonlinear reduction method, is contrasted to PCA. Two hyperparameters of this technique include perplexity and learning rate. These are important because they affect the shape of latent space. Perplexity was adjusted in small increments until the low dimensional mapping fit the high dimensional mapping closely. The learning rate corresponds to the local structure. We discovered that the larger it was, the better the local structure was preserved.

Comparison of PCA and t-SNE PCA and t-SNE were compared in three aspects. First, we considered whether each method was able to regenerate the full LES grid from the reduced state. Second, we considered the amount of information retained by the output latent spaces. Finally, we examined whether the latent spaces reflected the features of the original datasets. For both methods, we reduced the dimension in two directions. The first reduced the LES data with separate variables from dimension 744*192*192 to 744. The second reduced LES data from 36864*6 to 36864*2, where 36864 is the combined position and time and six is the number of variables.

Interpretability of the Latent Space To interpret the features of the output latent spaces, we first added labels to the original high resolution LES dataset. The selected scale for this process was wind power density [10] because it is important data to the wind industry. Additionally, it combines the *temp* and *vel* variables. Since air density is not a variable of the LES dataset, we calculated it from the temperature variable using this resource [14]. There are eight wind power density classifications. The criteria are based on the classification method established by Onea et al. [15]. We added an eighth label for when the wind power density was over 1148.75, labeled C8.

For further comprehension of the latent spaces, we tested different timesteps dropped different sets of variables. The tested timesteps include one hour, two hours, four hours, and twenty-four hours. For the sets of dropped variables, we first dropped *absolute height*. This left five variables combined. Next we dropped the u and v wind components, which left four variables combined. The final test dropped *absolute height*, u, and v. Comparisons between the different latent spaces allowed us to find features. As a final test, $\sqrt{u^2 + v^2}$ was used instead of *vel* to calculate wind power density, and we applied updated the labels. The variable *vel* was dropped and the generated latent space was compared the output of the test where we dropped u and v.

4.3 Upscaling from a Low-Resolution to High-Resolution Grid

3DCNN After constructing the model, a few optimizations were performed. Gradient computation was turned off during validation, some parts of the data were successfully run in parallel on GPU, automatic mixed precision was used, and different learning rates were tested. The code is available at https://github.com/CheukHinHoJerry/3DCNN-SUPER-2021-pytorch.

Interpolation We had more success in our approach with the interpolation method. The *resize* function allowed us to test each of the three flags by simply setting *interpolation* to each one.

ResUnet The ResUnet proved to be problematic, as the CNN did. When we attempted to train the model with the full dataset, the model failed to converge.

5 Results and Discussion

5.1 Exploratory Data Analysis and Visualization

The bias and correlation between ERA5 and high/low resolution LES data are shown as Table 2. It can be seen that LES is unbiased due to the small bias. Table 2 also illustrated that LES has high linear correlation with ERA5 since the correlations are all near to 1.

Table 2: bias and correlation of ERA5 and LES data

Measurement	Temperature	Horizontal speed	U component	V component
		ERA5 and high-resolution LES		
bias	-2.13 × 10^{-5}	-1.66 × 10^{-7}	3.48 × 10^{-8}	5.29 × 10^{-8}
correlation	0.8833	0.8065	-0.9677	0.9561
		ERA5 and low-resolution LES		
bias	-2.13 × 10^{-5}	-1.66 × 10^{-7}	3.48 × 10^{-8}	5.29 × 10^{-8}
correlation	0.8949	0.8037	-0.9702	0.9594

Table 3 presents the results of the bias and correlation computations where the LES dataset is segmented by time. It illustrates that bias and correlation depends on time since they are various for different time periods. However, the results based on the different segmented positions are similar to each other, as Table 4 displays, which means that bias and correlation do not depend on the positions in the grid.

5.2 Dimensionality Reduction of the Grid

Table 5 illustrates the correlation between variables of LES. It shows that "temp" is correlated with U (0.4760) and V component (0.3534) of wind speed, and "vel" has high correlation with "std" (0.7910). As shown in Table 6, the variance of "absolute_height" is 6129.4165, which can be dropped later.

The Fig 4 and 5 below show the 1 Dimension latent space reduced by PCA and t-SNE methods respectively. With similar shapes to ERA5, PCA preserves most information of the original data. However, only part of the information is retained by t-SNE, and the rest part is reversed.

The Fig 5 and 6 below compare the 2 Dimension latent space reduced by both methods. It can be seen that there is only one cluster for PCA and a pattern formed by different clusters for t-SNE. Therefore, t-SNE extracts the features of the original dataset better.

Fig 8-10 are different latent spaces generated by t-SNE technique. The first 3 figures are formed from the same one-hour data and latter three are formed by 24-hour data.

Table 3: bias and correlation of ERA5 and time-segmented low-resolution-LES

	Time period 1	Time period 2	Time period 3	Time period 4	Entire time series
			Temperature		
bias	5.4771	-1.5317	-7.8915	3.9460	-2.13 $\times 10^{-5}$
correlation	0.7780	0.9011	-0.8267	0.8935	0.8949
			Horizontal speed		
bias	0.2436	-0.0565	6.9149 $\times 10^{-5}$	-0.1872	-2.13 $\times 10^{-5}$
correlation	0.8647	0.7599	0.6475	0.8647	0.8037
			U component		
bias	0.9747	-0.6195	-0.6428	0.2876	3.48 $\times 10^{-8}$
correlation	0.9781	-0.9672	-0.9421	0.9789	-0.9702
			V component		
bias	-0.1252	-0.1767	0.7601	-0.4582	5.29 $\times 10^{-8}$
correlation	0.9672	0.9684	0.9407	0.9641	0.9594

Table 4: bias and correlation of ERA5 and grid-divided low-resolution-LES

	Area 1	Area 2	Area 3	Area 4	Full grid
			Temperature		
bias	-2.13 $\times 10^{-5}$	-2.13 $\times 10^{-5}$	-2.13 $\times 10^{-5}$	-2.13 $\times 10^{-5}$	-2.13 $\times 10^{-5}$
correlation	0.8924	0.8987	0.8911	0.8961	0.8949
			Horizontal speed		
bias	-1.66 $\times 10^{-7}$	-1.66 $\times 10^{-7}$	-1.66 $\times 10^{-7}$	-1.66 $\times 10^{-7}$	-1.66 $\times 10^{-7}$
correlation	0.8095	0.7778	0.7929	0.6564	0.8037
			U component		
bias	3.48 $\times 10^{-8}$	3.48 $\times 10^{-8}$	3.48 $\times 10^{-8}$	3.48 $\times 10^{-8}$	3.48 $\times 10^{-8}$
correlation	0.9641	-0.9653	0.9503	-0.9505	-0.9702
			V component		
bias	5.29 $\times 10^{-8}$	5.29 $\times 10^{-8}$	5.29 $\times 10^{-8}$	5.29 $\times 10^{-8}$	5.29 $\times 10^{-8}$
correlation	0.9367	0.9395	0.9449	0.9325	0.9594

Table 5: Correlation of Variables between Dataset Variables

	u	v	vel	std	temp	absolute_height
u	1.0000	0.3254	-0.2319	-0.1787	0.4760	-0.0255
v	0.3254	1.0000	-0.2056	-0.1747	0.3534	-0.0108
vel	-0.2319	-0.2056	1.0000	0.7910	-0.1807	0.2064
std	-0.1787	-0.1747	0.7910	1.0000	-0.1255	0.0482
temp	0.4760	0.3534	-0.1807	-0.1255	1.0000	-0.2464
absolute_height	-0.0255	-0.0108	0.2064	0.0482	-0.2464	1.0000

Table 6: variance of LES variables

	u	v	vel	std	temp	absolute_height
variance	8.7451	12.3115	6.9760	0.2423	4.4894	6129.4165

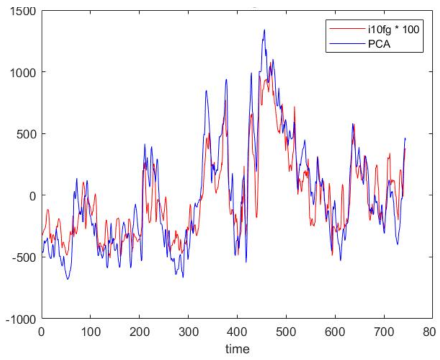

Fig. 4: "vel" reduced by PCA and "i10fg" magnify 100 times

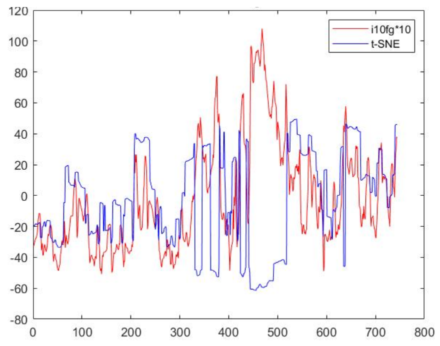

Fig. 5: "vel" reduced by t-SNE and "i10fg" magnify 10 times

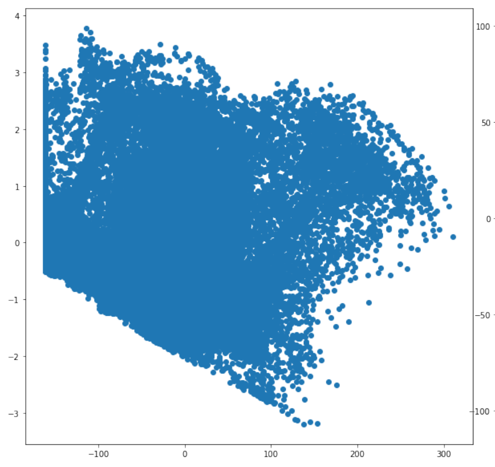
Fig. 6: reduced 2D plot by PCA method

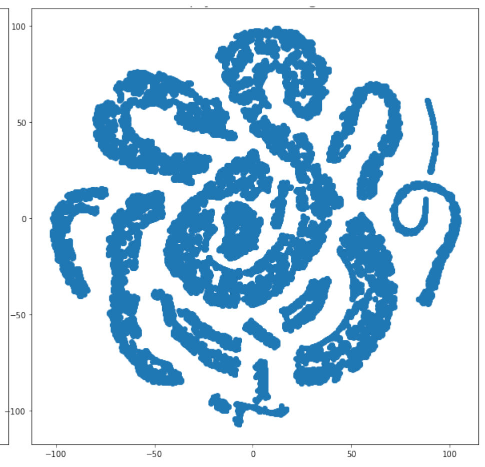
Fig. 7: reduced 2D plot by t-SNE method

The 1-hour data shows a good separation of different classes of data on each cluster, and the categories are arranged according to the level of classes. The left part of Fig 8 shows the latent space generated by the high-resolution LES with all 6 variables (”vel”, ”temp”, ”std”, ”u”, ”v”, and ”absolute_height”). The right part of Fig 8 is generated by LES with 4 variables, which are not including ”u” and ”v”. Comparing two figures in Fig 8, the latent space without ”u” and ”v” has smoother boundaries that distinguish different classes. However, dropping ”u” and ”v” has little effect on the shape of the latent space. Conversely, there is only one large cluster formed after dropping the variable ”absolute_height”, as shown in the left part of Fig 9. It indicates that the shape with separated clusters in the left of Fig 8 is due to the diversity of ”absolute_height”. Hence, removing ”absolute_height” is helpful to observe the magnitude of wind power density, which gradually decreases from left to right in the left part of Figure 9.

Different from the latent spaces formed by one-hour data, the ones gotten from 24-hour data is shaped like a sphere. Nevertheless, comparing the latent spaces generated by different dates can also illustrate which date has more wind power density. Since the main color of left part of Fig 10 is blue, which represents ”C1”, and it is pink (”C7”) in the right of Fig 10, it can be concluded that the date 01-22 has much stronger wind power than 01-14.

There is one more observation from Fig 11 that the well segmented bands of different classes would turn into rings with the increasing of input time periods.

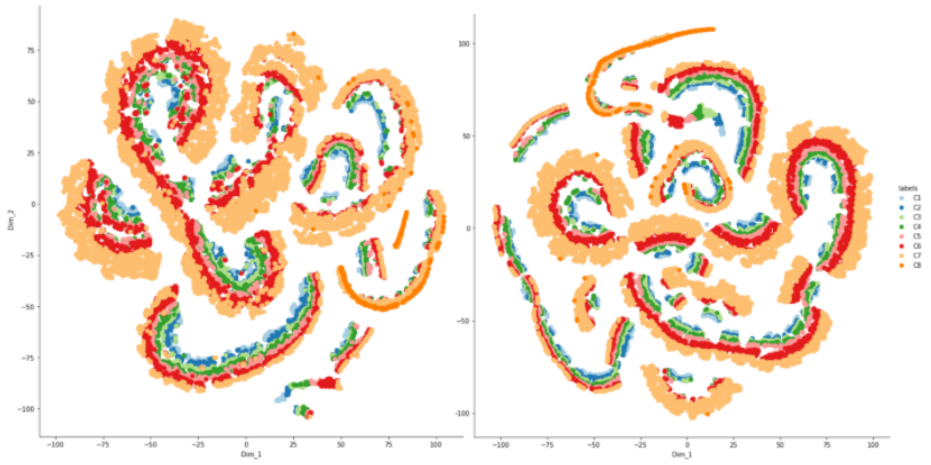

Fig. 8: 1-hour LES with 6 variables(left) and 4 variables(right)

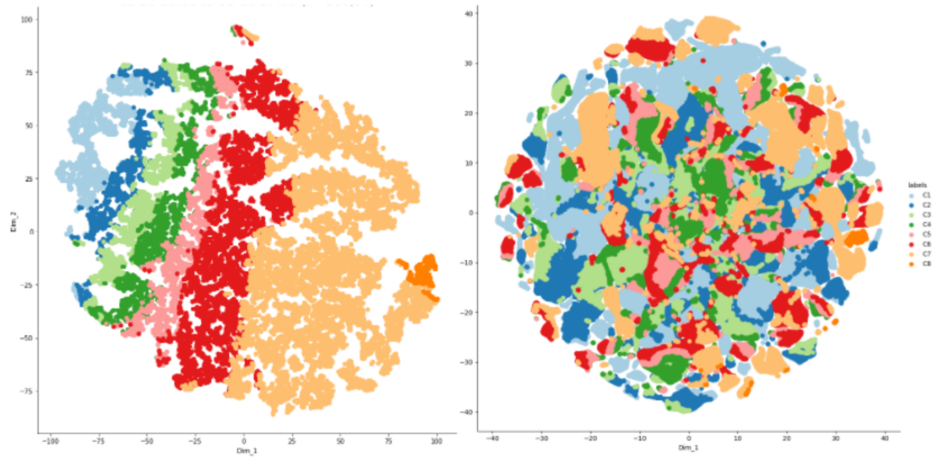

Fig. 9: 5 variables ('u','v','std','vel', and 'temp')- 1 hour (left) and 24 hours (right)

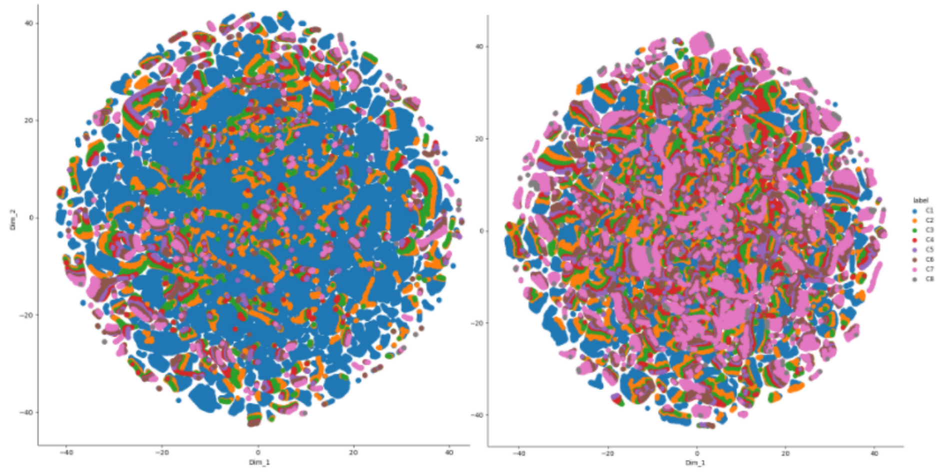

Fig. 10: left: 01-14 right: 01-22

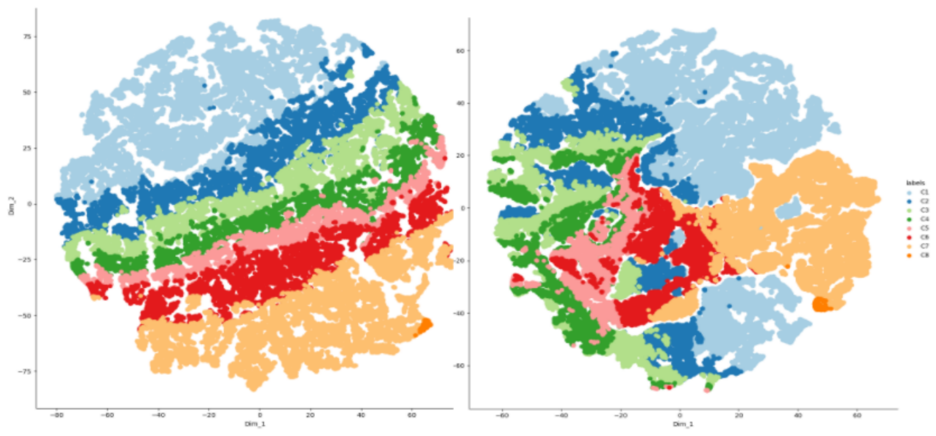

Fig. 11: 5 variables ('u','v','std','vel', and 'temp')- 2 hours (left) and 4 hours (right)

5.3 Upscaling from a Low-Resolution to High-Resolution Grid

3DCNN With the dataset used during the 2019 REU, the following results were achieved. A low resolution LES output was upscaled to a higher resolution. This is shown in Figure 11.

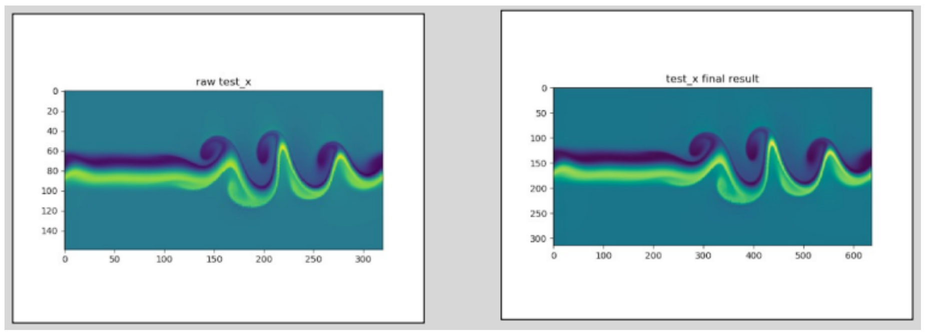

Fig. 12: Upscaling of LES data by 3DCNN

Interpolation The PSNR of nearest-neighbor, bilinear, and bicubic interpolations are 23.48, 24.12 and 24.13 respectively. The SSIM of them are 0.64, 0.67 and 0.68. Figure 12 compares the upscaled low-resolution LES with 'vel' variable by cubic interpoltion and the original high-resolution LES.

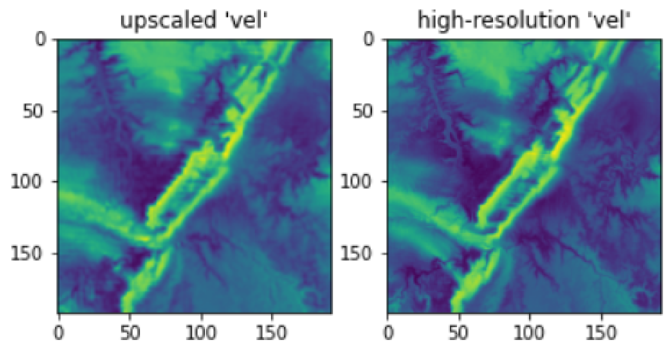

Fig. 13: left: upscaled by cubic interpolation right: original high-resolution LES

ResUnet When performing inference on the sample dataset with the trained model, we achieve an average MSE loss of 25.938, which is not ideal. We were not able to train the model with the full dataset, but the model failed to converge.

References

1. Julian, A., Davoust, S., Charuvaka, A.: Challenge 3: Finding Hidden Patterns in High Resolution Wind Flow Model Simulations. Oak Ridge National Laboratory (2022). https://smc-datachallenge.ornl.gov/ch3_windflow/
2. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. LNCS, vol 9351. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
3. Holland, S. M.: PRINCIPAL COMPONENTS ANALYSIS (PCA). Department of Geology, University of Georgia, Athens, GA 30602-2501.(2019). http://strata.uga.edu/8370/handouts/pcaTutorial.pdf
4. Maaten, L. V., Hinton, G.: Visualizing Data using t-SNE. In: Jornal of Machine Learning Research, vol 9. (2008). https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbcl
5. Fukami, K., Nakamura, T., Fukagata, K.: Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. Physics of Fluids, 32(9), 095110. (2020). https://arxiv.org/abs/2006.06977v2
6. Stengel, K., Glaws, A., Hettinger, D., King, R. N.: Adversarial super-resolution of climatological wind and solar data. In: Proceedings of the National Academy of Sciences, 117(29), 16805-16815. (2020). https://www.pnas.org/doi/full/10.1073/pnas.1918964117
7. . Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. (2015). https://arxiv.org/pdf/1411.4038.pdf
8. How to Calculate a Pearson Correlation Coefficient by Hand Statology (2020). https://www.statology.org/correlation-coefficient-by-hand/
9. Glen, S.: Variance: Simple Definition, Step by Step Examples StatisticsHowTo: Elementary Statistics for the rest of us! urlhttps://www.statisticshowto.com/probability-and-statistics/variance/
10. Kasper, D.: Wind Energy and Power Calculations. Dutton e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University. https://www.e-education.psu.edu/emsc297/node/649
11. Ho, C., Li, J.: Video Super Resolution with Integrated 2D and 3D Convolution Neural Network. (2021). https://www.jics.utk.edu/recsem-reu/recsem21.
12. Zhang, Z., Qingjie, L., Wang, Y.: Road Extraction by Deep Residual U-Net (2017). IEEE Geoscience and Remote Sensing Letters. https://arxiv.org/pdf/1711.10684.pdf
13. OpenCV resize image using cv2.resize(). TutorialKart. (2020). https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/
14. Hall, N.: Earth Atmosphere Model. NASA. (2021). https://www.grc.nasa.gov/www/k-12/airplane/atmosmet.html
15. Onea, F., Ruiz, A., Rusu, E.: An Evaluation of the Wind Energy Resources along the Spanish Continental Nearshore. Energies. (2020).

6 Appendix

Listing 1.1: Code for Question 1 - low resolution

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import xarray as xr
import netCDF4 as nc
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan,strategy='mean')
from sklearn.decomposition import PCA
pca = PCA(n_components=1)

#load datasets
ds_era5 = xr.load_dataset('/data/perdigao_era5_2020.nc')
ds_high_res = xr.load_dataset('/data/perdigao_high_res_1H_2020.nc')
ds_low_res = xr.load_dataset('/data/perdigao_low_res_1H_2020.nc')

#pre-process ds_low_res from 3D to 1D
#vel
vel=np.array(ds_low_res['vel'])
n=vel[0,:,:].reshape(-1).shape[0]
m=ds_low_res.time.shape[0]
Bve=np.empty((m,n))

#"vel" of all positions 96*96 for each time node
for i in range(m):
    Bve[i,:]=vel[i,:,:].ravel()

#replace the missing data with mean
imp.fit(Bve)
Bve =imp.transform(Bve)

#to get zero mean matrix (standardize in column)
##vel
Bve -= Bve.mean(axis=0)
reduced_Bve = pca.fit_transform(Bve)
reduced_Bve = np.squeeze(reduced_Bve)

#pre-processing ERA5_temp
##i10fg
i10fg_A = np.array(ds_era5['i10fg'])
i10fg_A = np.squeeze(i10fg_A)
i10fg_A -= np.mean(i10fg_A)

# bias between era5 and ds_low_res
diff_BAve = reduced_Bve - i10fg_A #wind speed
```

```
biasVe=sum(diff_BAve)/m
# to compare the correlation between era5 and ds_high_res
corr_Bveh_Ai10 = np.corrcoef(reduced_Bve_h,i10fg_A) #wind speed
print(corr_Bveh_Ai10)

#divided by time
def divide_timeseries (LES,LES_var,ERA5_var,divided_number):
    var = np.array(LES[LES_var])
    era = np.squeeze(np.array(ds_era5[ERA5_var]))
    era -= np.mean(era)
    n=var[0,:,:].reshape(-1).shape[0]
    m=LES.time.shape[0]
    mm=int(m/divided_number)
    B=[None]*divided_number
    reduced_B = [None]*divided_number
    corr_B = [None]*divided_number
    bias_B = [None]*divided_number

    for i in range(divided_number):
        B[i]=np.empty((mm,n))
        for j in range(mm):
            B[i][j,:]=var[j+mm*i,:,:].ravel()
        imp.fit(B[i])
        B[i] =imp.transform(B[i])
        B[i] -= B[i].mean(axis=0)
        reduced_B[i]= np.squeeze(pca.fit_transform(B[i]))
        corr_B[i] = np.corrcoef(reduced_B[i],era[mm*i:mm*(i+1)])
        bias_B[i] = sum(reduced_B[i]-era[mm*i:mm*(i+1)])/mm

    print(corr_B)
    print(bias_B)

divide_timeseries(ds_low_res,'temp','t2m',4)
divide_timeseries(ds_low_res,'vel','i10fg',4)
divide_timeseries(ds_low_res,'u','u100',4)
divide_timeseries(ds_low_res,'v','v100',4)

#ds_low_res (divide into 4 parts)
## wind speed
vel=np.array(ds_low_res['vel'])
Bve1=np.empty((m,nn))
Bve2=np.empty((m,nn))
Bve3=np.empty((m,nn))
Bve4=np.empty((m,nn))
for i in range(m):
        Bve1[i,:]=vel[i,:x2f,:x2f].ravel()
        Bve2[i,:]=vel[i,:x2f,x2f:].ravel()
        Bve3[i,:]=vel[i,x2f:,:x2f].ravel()
        Bve4[i,:]=vel[i,x2f:,x2f:].ravel()
```

```
#replace the missing data with mean
imp.fit(Bve1)
Bve1 =imp.transform(Bve1)
imp.fit(Bve2)
Bve2 =imp.transform(Bve2)
imp.fit(Bve3)
Bve3 =imp.transform(Bve3)
imp.fit(Bve4)
Bve4 =imp.transform(Bve4)

#to get zero mean matrix (standardize in column)
Bve1 -= Bve1.mean(axis=0)
reduced_Bve1 = pca.fit_transform(Bve1)
reduced_Bve1 = np.squeeze(reduced_Bve1)
Bve2 -= Bve2.mean(axis=0)
reduced_Bve2 = pca.fit_transform(Bve2)
reduced_Bve2 = np.squeeze(reduced_Bve2)
Bve3 -= Bve3.mean(axis=0)
reduced_Bve3 = pca.fit_transform(Bve3)
reduced_Bve3 = np.squeeze(reduced_Bve3)
Bve4 -= Bve4.mean(axis=0)
reduced_Bve4 = pca.fit_transform(Bve4)
reduced_Bve4 = np.squeeze(reduced_Bve4)

#correaltion
corr_Bve1_Ai10 = np.corrcoef(reduced_Bve1,i10fg_A)
corr_Bve2_Ai10 = np.corrcoef(reduced_Bve2,i10fg_A)
corr_Bve3_Ai10 = np.corrcoef(reduced_Bve3,i10fg_A)
corr_Bve4_Ai10 = np.corrcoef(reduced_Bve4,i10fg_A)
print(corr_Bve1_Ai10)
print(corr_Bve2_Ai10)
print(corr_Bve3_Ai10)
print(corr_Bve4_Ai10)

# bias
diff_BAve1 = reduced_Bve1 - i10fg_A
biasVe1=sum(diff_BAve1)/m
diff_BAve2 = reduced_Bve2 - i10fg_A
biasVe2=sum(diff_BAve2)/m
diff_BAve3 = reduced_Bve3 - i10fg_A
biasVe3=sum(diff_BAve3)/m
diff_BAve4 = reduced_Bve4 - i10fg_A
biasVe4=sum(diff_BAve4)/m
print(biasVe1,biasVe2,biasVe3,biasVe4)
```

Listing 1.2: Code for Question 2

```
import xarray as xr
import netCDF4 as nc
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import scipy.stats as stats

from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan,strategy='mean')
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler

#Import scikitlearn for machine learning functionalities
import sklearn
from sklearn.manifold import TSNE
from sklearn.datasets import load_digits # For the UCI ML handwritten digits dataset

# Import matplotlib for plotting graphs ans seaborn for attractive graphics.
import matplotlib
import matplotlib.patheffects as pe
%matplotlib inline
from sklearn.decomposition import PCA

sp_high_res = xr.open_dataset('/data_samples/perdigao_high_res_1H_2020_01.nc')
s_high_res=sp_high_res.to_dataframe()
s_high_res.fillna(s_high_res.mean(),inplace=True)
s_high_res=s_high_res.drop(['height'],axis=1)

temp = s_high_res['temp']
pressure = 101.29 * (temp/288.08)**5.256
density = pressure/(0.2869*temp)
vel = s_high_res['vel']
power = 0.5 *density *(vel**3) #W/m^2

#PCA (2D)
X_pca = PCA(n_components=2).fit_transform(high_res)
plt.figure(figsize=(10, 10))
plt.title('PCA',fontsize=20)
plt.scatter(X_pca[:, 0], X_pca[:, 1], label="PCA")
plt.show()

def plot_tsne_2d_2p(ply,lr):
  high_744_2d_2p = TSNE(perplexity=ply,learning_rate=lr).fit_transform(s_high_res)
  plt.figure(figsize=(10, 10))
  plt.scatter(high_744_2d_2p[:,0],high_744_2d_2p[:,1])
  plt.title(f't-SNE_2D(ply={ply},learning_rate={lr})',fontsize=20)
  return plt.show()
plot_tsne_2d_2p(50,500)

#set labels
```

```
label=[None]*27426816
for i in range(27426816):
  if 0<=power[i]<114.87:
    label[i]="C1"
  elif 114.87<=power[i]<172.31:
    label[i]="C2"
  elif 172.31<=power[i]<229.75:
    label[i]="C3"
  elif 229.75<=power[i]<287.19:
    label[i]="C4"
  elif 287.19<=power[i]<344.62:
    label[i]="C5"
  elif 344.62<=power[i]<459.5:
    label[i]="C6"
  elif 459.5<=power[i]<1148.75:
    label[i]="C7"
  else:
    label[i]="C8"
s_high_res['label_0']=label

#time based
def select_tsne_2d (variable_number,i,j,ply,lr,label_number):
  if variable_number==6:
    dataset=s_high_res
  elif variable_number==5:
    dataset=s_high_res.drop(['absolute_height'],axis=1)
  elif variable_number==4:
    dataset=s_high_res.drop(['u','v'],axis=1)
  elif variable_number==3:
    dataset=s_high_res.drop(['u','v','absolute_height'],axis=1)
  elif variable_number==2:
    dataset=s_high_res.drop(['u','v','absolute_height','std'],axis=1)

  data = dataset.iloc[36864*(i-1):36864*(j-1),:variable_number]
  labels = dataset.iloc[36864*(i-1):36864*(j-1),variable_number+label_number]
  high_2d = TSNE(perplexity=ply,learning_rate=lr).fit_transform(data)

  tsne_data = np.vstack((high_2d.T, labels)).T
  tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "labels"))
  list1 = labels.unique()
  list1.sort()
  sns.set_palette("Paired")

  sns.FacetGrid(tsne_df, hue="labels", hue_order = list1, size=10).map(plt.scatter, 'D
  st_date = int(np.floor((i-1)/24))+1
  st_time = (i-1)%24
  ed_date = int(np.floor((j-1)/24))+1
  ed_time = (j-1)%24
  plt.title(f'01-{st_date}␣{st_time}:00␣to␣01-{ed_date}␣{ed_time}:00␣(ply={ply},lr={lr
  plt.show()
  #plt.savefig(f'01{st_date}_{st_time}_01{ed_date}_{ed_time}_lr{lr}_{variable_number}v

select_tsne_2d(6,378,379,50,500,0)
select_tsne_2d(5,378,379,50,500,0)
select_tsne_2d(4,378,379,50,500,0)
select_tsne_2d(5,378,402,50,500,0)
select_tsne_2d(4,313,337,50,500,0)
select_tsne_2d(4,505,529,50,500,0)
```

Listing 1.3: Code for Question 3 - interpolation

```
import xarray as xr
import numpy as np
import matplotlib.pyplot as plt
sp_high_res = xr.open_dataset('/data_samples/perdigao_high_res_1H_2020_01.nc')
sp_low_res = xr.open_dataset('/data_samples/perdigao_low_res_1H_2020_01.nc')
df_low=sp_low_res.to_dataframe()
df_low.fillna(df_low.mean(),inplace=True)
df_low=df_low.drop(['height'],axis=1)

df_high=sp_high_res.to_dataframe()
df_high.fillna(df_high.mean(),inplace=True)
df_high=df_high.drop(['height'],axis=1)
order = df_high.columns
df_low=df_low[order]

#low-res
vel_first_1d_low=np.array(df_low.iloc[:9216].iloc[:,2]) #change last number for
vel_first_1d_high=np.array(df_high.iloc[:36864].iloc[:,2])
vel_first_2d_low=np.reshape(vel_first_1d_low,(96,96))
vel_first_2d_high=np.reshape(vel_first_1d_high,(192,192))

import cv2
import math
from skimage.metrics import structural_similarity as ssim
def upscaling_cv2(low_res,high_res,method,variable_name):
  scale_percent = 2        # percent of original size
  height = int(low_res.shape[0] * scale_percent)
  width = int(low_res.shape[1] * scale_percent)
  dim = (width, height)
  resized = cv2.resize(low_res, dim, interpolation = method)

  plt.subplot(1, 2, 1)
  plt.imshow(resized)
  plt.title(f"upscaled␣'{variable_name}'")
  plt.subplot(1, 2, 2)
  plt.imshow(high_res)
  plt.title(f"high-resolution␣'{variable_name}'")
  plt.suptitle(f"cv2_{method}")
  plt.show()
  # PSNR
  max1 = np.max(high_res)
  MSE1_sqrt = np.sqrt(np.mean(np.power(resized-high_res,2)))
  psnr = 20 * math.log10(max1/MSE1_sqrt)
  print(f"PSNR = {psnr}")

  #SSIM
  SSIM = ssim(resized, high_res)
  print(f"SSIM = {SSIM}")
  upscaling_cv2(vel_first_2d_low,vel_first_2d_high,cv2.INTER_NEAREST,"vel")
  upscaling_cv2(vel_first_2d_low,vel_first_2d_high,cv2.INTER_LINEAR
,"vel")
  upscaling_cv2(vel_first_2d_low,vel_first_2d_high,cv2.INTER_CUBIC  ,"vel")
```