Computers and Fluids 195 (2019) 104318

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Key time steps selection for CFD data based on deep metric learning

Yang Liu [a,b,*], Yutong Lu [c], Yueqing Wang [b], Dong Sun [b,d], Liang Deng [a,b], Yunbo Wan [b], Fang Wang [b]

[a] College of Computer, National University of Defense Technology, Changsha, China
[b] Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang, China
[c] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[d] State Key Laboratory of Aerodynamics, China Aerodynamics Research and Development Center, Mianyang, China

ARTICLE INFO

Article history:
Received 13 June 2019
Revised 25 September 2019
Accepted 9 October 2019
Available online 10 October 2019

Keywords:
Flow visualization
Key time steps selection
CFD flow field
Deep metric learning

ABSTRACT

As one of the main technologies of flow visualization, key time steps selection plays a key role in solving storage limit and has been intensively studied. In this paper, we introduce Deep Metric Learning (DML) into key time steps selection for Computational Fluid Dynamics (CFD) data and propose a local selection method based on DML. In specific, the proposed method samples small patches from CFD data, trains a Siamese deep neural network which has a symmetry structure with two Convolutional Neural Networks (CNN), and then selects the key time steps according to the similarities between consecutive time steps which are assessed by the networks. Compared with one of the existing local selection methods, the Myers's method, our method has advantages in accuracy, precision and recall, and the selection results are better. Experimental results also demonstrate the good generalization of the proposed method on CFD datasets.

1. Introduction

With the steady increase of computational performance, Computational Fluid Dynamics (CFD) researchers could use more complex models and larger grids to simulate flow fluids. For example, Parallel Hierarchic Adaptive Stabilized Transient Analysis (PHASTA) [1] uses more than 6 billion element grids to simulate flow over a vertical tail-rudder assembly for a geometry that exactly matches the configuration of an ongoing wind tunnel experiment [2]. For the unsteady simulation of such a scale, it is not possible to save the result data of all time steps to the storage system for flow visualization, and we have to keep a small fraction of data at some pre-selected time interval which is decided with simple heuristics [3]. However, there are two problems to consider. One is that the time interval is hard to choose, especially when the simulated phenomena are complex. The other is that because the time steps are selected with fixed time interval, they are probably not the most important ones which indicate the changing trends of flow fields. Thus, how to select the key time steps in a large-scale unsteady simulation, whose result datasets are time-varying, is a difficult problem to solve.

To solve this problem, methods of selecting key time steps are proposed. Methods can be roughly classified into two categories. The first one, called global method [3–5], is to use the entire set of time steps after the simulation is finished. These methods can take advantage of all the data to achieve a global optimization, but that is exactly what the storage system can not afford. The second one, named local method, identifies the key time steps while the simulation is running [6–10]. By this in-situ way, these methods only use several newly computed time steps and try to select the ones in which the flow fields begin to have a great change. For the large-scale unsteady simulation, the local methods reduce the need for transferring to storage and seem to be the only qualified way to do the selection.

To select key time steps with a local method is to extract the features of the flow field data of several newly computed time steps, compute the similarities between the features and then make the decision based on the similarities. For feature extraction, deep learning [11,12] shows great advantages in related fields, such as image recognition, face re-identification, etc. [13–17]. Besides, in the past few years, deep learning has achieved great success in CFD. More and more researchers tend to use deep learning methods such as Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN) to predict the flow field around physical models and learn the equations used in the numerical simulation [18–20]. Several researchers also have made some attempts in

* Corresponding author at: Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang, China.
E-mail addresses: liuyang777@nudt.edu.cn, 86821957@qq.com (Y. Liu).

https://doi.org/10.1016/j.compfluid.2019.104318
0045-7930/

CFD data compression[21] and feature extraction of flow field [22–24]. The significantly reduced computational cost and improved accuracy of simulation results show the potential of deep learning methods for processing CFD data. For the computation of the similarities between features, metric learning [25] shows its strength in relevant areas, such as computer vision, text analysis, etc. [26–28]. It learns a distance function tuned to a particular task and is useful in conjunction with nearest-neighbor methods that rely on similarities.

Therefore, Deep Metric Learning (DML) [29], which combines feature extraction and metric learning in a unified framework, may be used as a local method. Inspired by a Siamese neural network [30] which aims at signature verification, DML is proposed to solve the problem of person re-identification. Given two person images, DML uses a Siamese deep neural network to calculate their similarity and decide whether the persons from the two images are the same or not. Similarly, we can extract the features of the flow field data of time steps, use metric learning to assess the similarities between the features and select the key time steps based on the similarities.

However, to the best of our knowledge, no earlier research using a deep learning method to select key time steps for CFD data has been reported. Therefore, we introduce DML into key time steps selection for CFD data and propose a local selection method based on DML. Using small patches from CFD data, a Siamese deep neural network which has a symmetry structure with two subnetworks connected by a cosine similarity computation is trained. Each sub-network includes four convolutional layers and one fully connected layer. Binomial deviance [31] is used as the loss function to approximate the similarities between flow field data, and then the key time steps are selected based on the change of these similarities. Compared with one of the existing local methods, our method has advantages in selection results. Experimental results demonstrate the good generalization of the proposed method from following aspects: (1) the network trained with the preceding time steps can be applied to the following time steps; (2) the network trained with the data of a single case can be applied to similar cases. Therefore, our contributions in this work are:

- For CFD data, a local key time steps selection method based on DML is proposed.
- Compared with one of the existing local methods, our method yields better selection results.
- The good generalization of the proposed method is demonstrated.

The rest of the paper is organized as follows: Section 2 discusses related work of key time steps selection and deep learning methods; Section 3 presents our selection method for CFD time steps based on DML; Section 4 describes experimental setup and makes various analyses on the results; and finally, Section 5 summarizes our work and provides insights for future work.

2. Related work

2.1. Key time steps selection

Key time steps selection for CFD data is usually divided into global method and local method. The first category has been proven to be very successful in this area. Wang et al. [4] introduce an importance-driven approach to time-varying volume data visualization for enhancing that ability. Tong et al. [3] present a novel technique that can retrieve the key time steps from large scale time-varying data sets. Frey and Ertl [32] present an approach to adaptively select time steps from time-dependent volume data sets for an integrated and comprehensive visualization. Zhou et al. [5] propose an evaluation of selected time steps by computing the difference in the amount of information using variation of information from information theory. However, the global methods mentioned above and others [33,34] need the entire set of time steps which is unbearable for the storage system.

This problem can be addressed in the local methods which are able to detect change points in the simulations. Salloum et al. [6] propose a new indicator and trigger to detect sudden heat release in the simulations of turbulent combustion. Ling et al. [8] use feature importance metrics to detect events of interest and their method can be implemented in a high performance computing setting because of very little communication requirement between processors. Myers et al. [7] partition the simulation as it runs. They use the value of a single variable, or the mean of a variable, or a derived quantity to represent the flow field data. By using only several newly computed time steps, they utilize a statistical method, a modified F-statistic, to identify important time steps where the slope of partitions changes. They can also reconstruct a linear approximation of the simulation by saving sufficient statistics. Banesh et al. [9] expand on this work to present a statistical approach of change point detection in simulated ocean data. Others [10] are trying to make this kind of method more practical.

2.2. Deep learning in CFD

With the rise of artificial intelligence, more and more researches emerge in applying deep learning to CFD. Guo et al. [18] succeed to use CNN to predict steady flow field around bluff objects with significantly reduced computational cost. Lin et al. [19] use a deep neural network (DNN) to model Reynolds stress anisotropy tensor in Reynolds-averaged Navier Stokes (RANS) simulations, and significantly improve the accuracy of simulation results. Lee et al. [20]. predict unsteady laminar vortex shedding behind a cylinder using GAN and particularly emphasize the ability of learning the solution of N-S equations. Liu et al. [21] introduce deep learning into CFD data compression and propose a novel in-situ compression method based on GAN. Several researchers [22–24] manage to use CNN to extract the features of flow field. These studies also show the potential of deep learning methods for processing CFD data.

2.3. Deep metric learning

The local method to select key time steps includes feature extraction and similarities computation. Feature extraction is a key technique of deep learning which has achieved big success in the areas of application namely computer vision, speech recognition, and natural language processing [13]. Farias et al. [14] introduce a special neural network known as Sparse Auto-Encoder for two classification problem of TJ-II fusion database. Jiang et al. [15] propose a real time Internet cross media retrieval method which handles for the initial evaluation of image recognition samples to detect the needs of recognition algorithm. Other studies [16,17] also prove that deep learning is quite qualified for feature extraction.

Metric learning [25] aims at the problem that how to assess the similarity or distance between the pairs of images, and learns task-specific distance functions in a supervised manner. In the area of computer vision, Kulis et al. [26] perform metric learning over a database of 300,000 image patches and try to retrieve matching image patches from other images to a query image patch. For the Caltech-101 data set, Frome et al. [27] adopt a local metric approach to produce state-of-the-art results at the time, and led to further work in the vision community on applications of metric learning. In the area of text analysis, Davis et al. [28] show that metric learning could achieve gains in terms of recall over standard similarities. Besides, metric learning is also applied in other areas, such as music analysis, automated program debugging, etc.

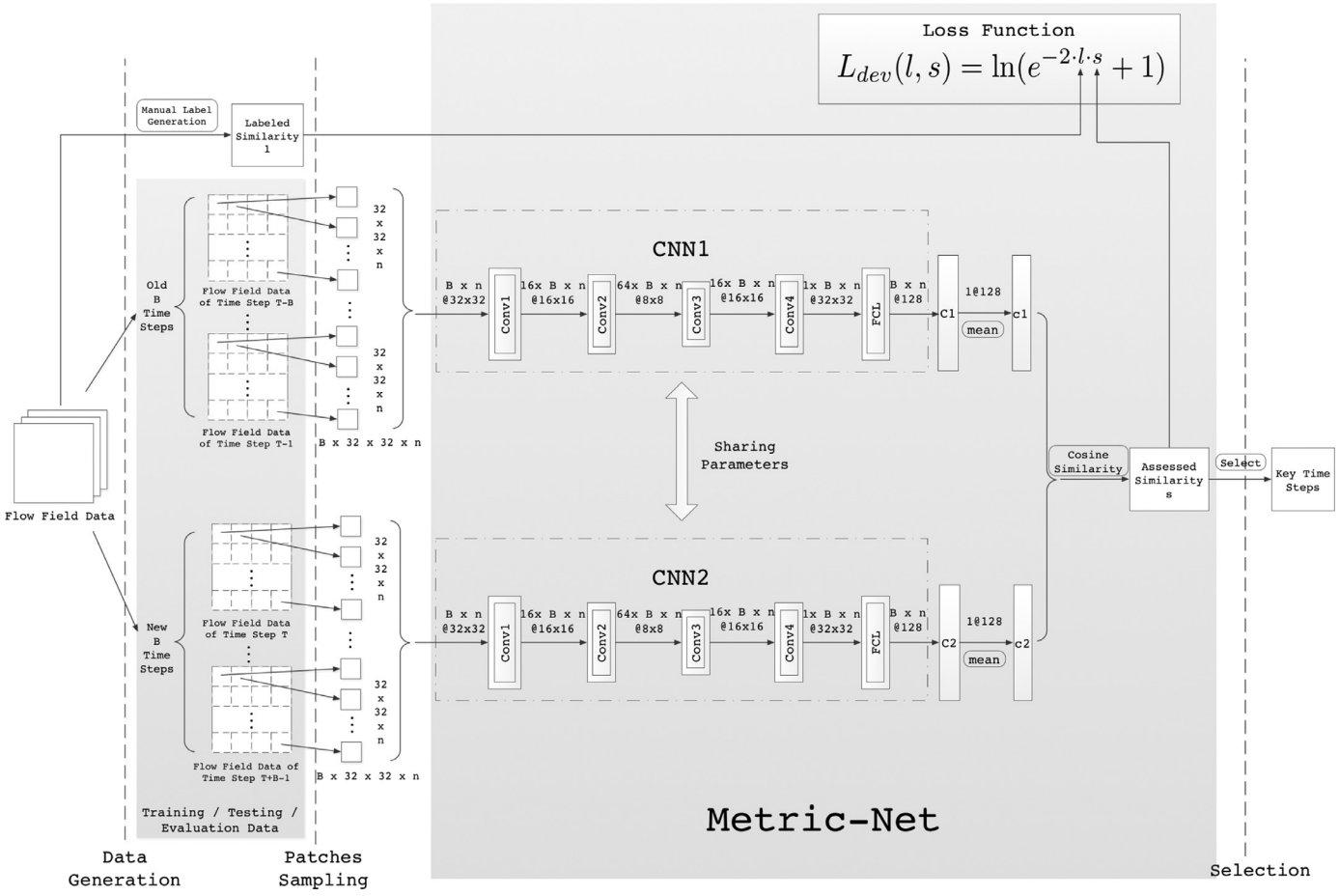

Fig. 1. The architecture of the proposed key time steps selection method. The Metric-Net is used to compute the similarities and the Selection selects the key time steps.

DML [29] combines feature extraction and metric learning in a unified framework. Inspired by Siamese neural network [30] which is originally proposed for signature verification, DML applies deep learning in the person re-identification problem. Siamese deep neural network is used to assess the similarity between different person images. Unlike the original work, DML does not share the same weights and biases in the two sub-networks. The cosine similarity is used to compute the similarity between the features generated by deep neural networks and the DML is trained with binomial deviance as the loss function. Extensive results illustrate that the network can switch flexibly between two modes, which shows its good generalization.

However, as far as we know, there is currently no relevant research on using deep metric learning methods to select key time steps for CFD data.

3. Our proposed method

The architecture of the proposed DML-based selection method is presented in Fig. 1. A time step T is regarded as a key one if it indicates the changing trends of flow fields. Usually, the flow field changes a lot at T. Thus, the similarity between the data before T and the data after T needs to be assessed. As flow fields usually change slowly in consecutive time steps, the corresponding data are similar. To accurately reflect the changing trends of flow fields, the data of B (B $\geq$ 3) time steps are used to calculate the similarity instead of only one time step. That is, to judge if the time step T is a key one or not, the data which are used to compare are of time steps from T-B to T-1 (named Old B Time Steps in Fig. 1) and the ones from T to T+B-1 (named New B Time Steps in Fig. 1). The size of B is decided according to the size of data and available memory resources.

Our method includes three parts: the data processing, the network, named Metric-Net, and the selection. Accordingly, the method is divided into three steps. Firstly, in the data processing, input data are generated and sampled. Then, Metric-Net takes these data as input and trains a DML model to assess the similarities. Finally, the selection is responsible for selecting the key time steps, that is, deciding whether a time step is important or not.

3.1. The data processing

The data processing is composed of data generation and patches sampling. For the data generation, as supervised learning is adopted, four kinds of data need to be generated: training data, testing data, evaluation data and labeled data. For training, testing and evaluation data, primary variables of the flow field are used, i.e. pressure, density and three components of velocity (u, v, w). As gradient descent is used for network learning, all the input data are max-min normalized respectively by the maximum and minimum from the seen data. Training data is generated from a series of time steps from the beginning of simulation to the moment when physical phenomenon is completely presented. As mentioned above, to evaluate the importance of time step T, the data of B time steps before T and B time steps after T (including T) are used.

To train neural networks to learn a mapping from training data to the similarities, labeled data, i.e. true data, are generated to be approximated in the network training. We observe the whole simulation process and manually select the key time steps which indicate the changing trends of flow fields. Asymmetry labels are as-

signed to the time step pairs to tune the network. For example, if time step T is labeled as a key time step, it is considered that flow field starts to change at time step T, that is, the flow field data of old B time steps is much different from the ones of new B time steps. Therefore, this time step pair (T-B, T) is taken as a dissimilar pair and its similarity is set as -1. If T is not labeled as a key time step, this pair is regarded as a similar pair and its similarity is set as 1.

The second step of the data processing is to sample the patches. For 2D data, small patches of size 32×32 from the data of Old B Time Steps and New B Time Steps are sampled and taken into Metric-Net as input. In this way, no matter what the scale and the shape of the flow field is, the shape of the input layer of Metric-Net is the same, that is $B \times n \times 32 \times 32$, where n denotes the number of the patches. Therefore, the network which is trained with one flow field case is qualified to be applied to other cases with different scales and shapes.

3.2. The metric-net

The middle of Fig. 1 illustrates the structure of Metric-Net which is based on a Siamese deep neural network. This network has a symmetry structure with two sub-networks which are connected by a cosine similarity computation. Each sub-network consists of four convolutional layers (Conv1 to Conv4 in Fig. 1) and one fully connected layer (FCL in Fig. 1).

The four convolutional layers are responsible for extracting features of the flow field data. Multiple layers are used to dig a deeper feature map and the numbers of output channels of these layers are 16, 64, 16, and 1, respectively. All of the convolutional layers use 3×3 convolutional kernels which need to be fixed through the training stage. Because the mode of convolution is set as SAME in our method, the size of output is only related to the strides of the convolutional layer. By setting the strides, we are able to adjust the shape of convolutional layers. Rectified Linear Unit (ReLU) is used as the activation function in convolutional layers.

Take CNN1 for example, after four convolutions, the fully connected layer connects all the features and outputs the representation ($C1$ in Fig. 1) of $B \times n$ patches. Then a mean value of $C1$ (of size $B \times n \times 128$) is calculated to represent the feature of old B time steps, i.e. $c1$ in Fig. 1, which is of size 1×128. Similarly, CNN2 outputs the feature representation $c2$ of new B time steps. To evaluate the similarity between old and new B time steps, the cosine similarity, given in Eq. (1), is adopted. As the labeled similarities are set as -1 or 1, tanh, which can output negative numbers, is used as the activation function of the last layer, i.e. the fully connected layer.

$$s = cos\_sim(c1, c2) = \frac{c1 \cdot c2^T}{\sqrt{c1 \cdot c1^T} \cdot \sqrt{c2 \cdot c2^T}} \tag{1}$$

In the training stage, the time step pairs are randomly selected. Sample patches and corresponding labeled data of these pairs are taken as input and Metric-Net is trained with Adam and back-propagation. Based on Siamese, the two sub-networks, CNN1 and CNN2, share the same weights and biases. In this way, the parameters needed are relatively simple and easy to tune. In order to make the assessed similarity s closer to the real similarity value l, binomial deviance L_{dev} is used as the loss function, given in Eq. 2. Thus, the network is optimized by minimizing L_{dev}. A complete algorithm for training Metric-Net is given in Algorithm 1. The input is flow field data and labeled data, and the output is the well-trained Metric-Net.

$$L_{dev}(l, s) = \ln(e^{-2 \cdot l \cdot s} + 1) \tag{2}$$

In the testing stage, time step pairs are handled according to the sequence of simulation. The well-trained Metric-Net takes the data of each pair as input and generates the corresponding similarity value which is sent to the selection part.

Algorithm 1 MetricNetworkTraining().

```
/* d is the original data. */
/* old is the data of old B time steps. */
/* new is the data of new B time steps. */
/* l is the labeled data. */
/* C1 and C2 are the outputs of CNN1 and CNN2, respectively. */
/* c1 and c2 are the mean values of C1 and C2, respectively. */
/* s is the cosine similarity of c1 and c2. */
/* E is the training epochs. */
1: for each e ∈ E do
2:    old, new ← RandomSample(d)
3:    C1 ← CNN1(old)
4:    C2 ← CNN2(new)
5:    c1 ← mean(C1)
6:    c2 ← mean(C2)
7:    s ← cos_sim(c1, c2)
8:    Update Metric-Net by minimizing L_dev(l, s) loss with Equation
2
9: end for
```

3.3. The selection

As mentioned above, if a time step T is a key one, the similarity of the time step pair (T-B, T) is set as -1. Therefore, the direct way to select a key time step is to find the pair whose similarity is negative, and the corresponding T is a key time step. However, the result may be not good enough in two cases.

- Case One: flow fields may change frequently in a period of time, leading to a large number of negative similarity values. In this case, our selection method will continuously select adjacent time steps which may be redundant.
- Case Two: flow fields may change sightly in a long period of time, leading to successive positive similarity values. In this case, no time steps can be selected, which could not reflect the changes of flow fields.

Therefore, we divide the whole time sequence into several intervals. For any time interval, the time step which has the smallest similarity in this interval is chosen as the key one of this interval. In specific, as the simulation is running in a time interval I, Metric-Net generates an assessed similarity s for each time step. The s is used to update the smallest similarity $s_{\min}$ of I if s is smaller than $s_{\min}$, and the time step X whose s equals to $s_{\min}$ is recorded. When the counter i reaches the length L of I, that is the simulation finishes this time interval, the time step X who has the smallest similarity $s_{\min}$ will be selected as the key time step of I if $s_{\min}$ is smaller than a pre-set threshold θ, and the corresponding flow field data is output to the storage system. The threshold θ is used to avoid that $s_{\min}$ is too large to represent a change point. For Case One, the time interval separates the frequently changing period and the redundant selection is avoided. For Case Two, even if the similarities in a time interval are all positive numbers, we are able to select the key time step who has the local minimum.

Two techniques are used here. The first one is the automatic adjustment of the length L of the time interval. For the first interval, L is initialized with simple heuristics. For later intervals, L is automatically changed to the distance from the time step P who has the smallest similarity in the last interval (the one before this interval) to the time step X who has the smallest similarity in this interval. In this way, the changing trend of flow fields is captured. If X is close to P, which means that flow field is changing quickly,

Algorithm 2 KeyTimeStepSelect().

/* s is the assessed similarity by Metric-Net. */
/* $s_{\min}$ is the local minimal similarity value. */
/* MAX is a large number. */
/* L is the length of the time interval. */
/* i is a counter for the time interval. */
/* P is the time step who has the smallest similarity in the last interval. */
/* X is the time step who has the smallest similarity in this interval. */
/* θ is the pre-set threshold. */
/* K is the set of key time steps. */
/* T is the whole time sequence. */
$s_{\min} \leftarrow MAX$, $i \leftarrow 0$, $P \leftarrow 0$, $L \leftarrow initial\_value$
for each $t \in T$ **do**
  $s \leftarrow MetricNet(t)$
  if $(s < s_{\min})$ **do**
    $s_{\min} \leftarrow s$, $X \leftarrow t$
  end if
  $i \leftarrow i + 1$
  if (i equals to L) **do**
    if (s is negative) **do**
      $i \leftarrow i - 1$
    else
      $s_{\min} \leftarrow MAX$, $i \leftarrow 0$
      $L \leftarrow X - P$
      $P \leftarrow X$
      if $(s < \theta)$ **do**
        output the flow field data of X
        append X to K
      end if
    end if
  end if
end for

L is accordingly changed to a small value to find more key time steps. If X is far from P, which means that flow field is changing smoothly, L is accordingly changed to a large value to find less key time steps. Note that P may not be the last key time step because its similarity may be larger than θ. Similarly, X may not be this key time step. The reason why we do not use the distance from the last key time step to this key time step to set L is that the last key time step may be too far from this key time step. A very large L could lead to mistakes in the following selection.

The second technique is the extension of the length L of the interval. The similarity value s of the final time step of a time interval I may be a negative number, which means the flow field is still changing greatly at the end of I. To avoid the redundant selection in Case One, we extend L by reducing the counter i until the similarity value becomes positive.

The algorithm for selection is given in Algorithm 2. The input is an initialized length L of time interval and the output is a set of key time steps K.

4. Experiments

In this section, KTS-DML is short for Key Time steps Selection based on DML which is proposed in Section 3. In order to evaluate the performance of KTS-DML, the Myers's method [7] is chosen for contrast, referred to Myers's, as it is successfully used in similar fields and has been applied to detect change points [9]. A series of experiments are conducted on a machine which equips an Intel Xeon CPU i7-4790K@4.00GHz and a memory of 32GB. All codes are written with Python 3.5.2 under Windows 7 operating system. To build Metric-Net, TensorFlow v1.4 is used. The following experimental results are the average of multiple tests. In the implementation of Myers's, we use the mean of a variable to represent the flow field data of a time step. As several parameters are needed, e.g. α and δ, the results of Myers's below are the best that we can get after tuning.

Table 1
Results in statistics.

Case	Method	TP	FP	TN	FN
Cylinder	Myers's	5	1	313	1
	KTS-DML	6	0	314	0
Triangle	Myers's	12	11	11,966	11
	KTS-DML	19	1	11,976	4
Plane	Myers's	9	5	4480	6
	KTS-DML	11	3	4482	4

4.1. Evaluation metrics

The following four metrics are used to compare the performance of KTS-DML and Myers's.

- $Accuracy = (TP + TN)/(TP + FP + FN + TN)$, the ratio of correctly prediction to the total time steps, which is the most intuitive performance measure. However, considering that the key time steps occupies a very small population of whole time steps, which means TN is relatively quite large, a high *Accuracy* may be achieved even if the selection results are all wrong. Thus, we have to look at other parameters for more details.
- $Precision = TP/(TP + FP)$, the ratio of correctly selected key time steps to the total selected key time steps, which shows that how many key time steps are real in the selected ones.
- $Recall = TP/(TP + FN)$, the ratio of correctly selected key time steps to the total labeled key time steps, which shows that how many key time steps are selected in the real ones.
- $F1 = 2 * (Recall * Precision)/(Recall + Precision)$, the weighted average of Precision and Recall. This score takes both FP and FN into account and is usually more useful than *Accuracy*, especially when an uneven class distribution occurs.

Above, TP is the number of the correctly selected key time steps, TN is the number of the correctly predicted non-key time steps, FP is the number of selected key time steps which are actually not, and FN is the number of the predicted non-key time steps which are actually key ones. As flow fields in consecutive time steps are usually similar, if the selected time step T is near the labeled key time step L, it will be manually checked if it is a correct one.

4.2. The performance comparison with Myers's

Our method is evaluated on three 2D cases: flow past cylinder (Cylinder), triangle (Triangle) and plane (Plane). For these three cases, the dimensions of the grids are all 381×101, and computational conditions are similar. For convenience, only the v component of velocity data which can accurately reflect the changes of flow fields is kept. For the first case, Cylinder, the data of 1600 time steps are collected in which the first 60% is the train set, the middle 20% is the evaluation set and the remaining 20% is the test set. For each of the latter two cases, 12,000 time steps are collected as the test set.

As shown in Tables 1 and 2 which report the results of KTS-DML and Myers's for the three cases, KTS-DML has advantages in every evaluation metric for all cases. As mentioned above, TN is

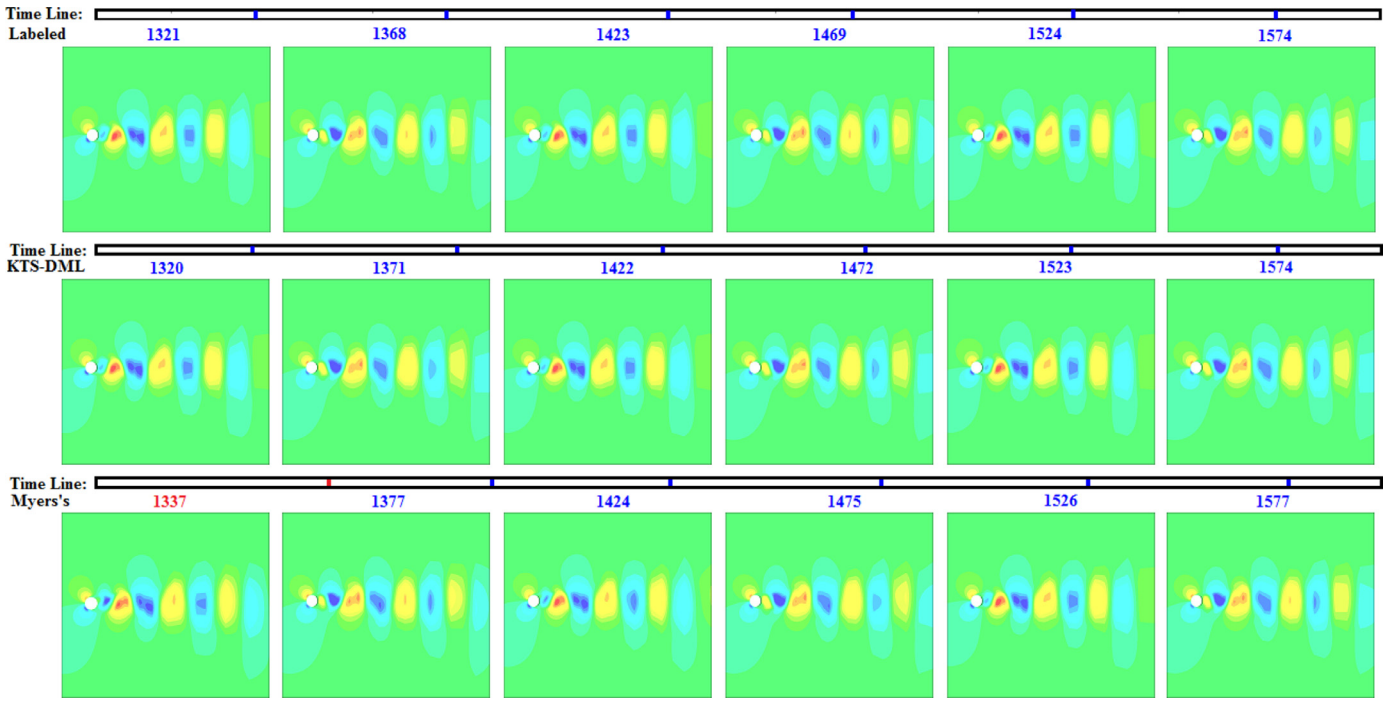

Fig. 2. The selection results of labeled, KTS-DML and Myers's for case Cylinder. In each picture with the index number of the corresponding time step, the red shows the regions with high velocity, while the blue indicates that the velocity in those regions is low. The time line shows the distribution of selected key time steps in the whole sequence, and a block or index number in red means this selected time step is not a labeled one while a blue block or index number represents a correctly selected key time step. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Results in evaluation metrics.

Case	Method	Accuracy	Precision	Recall	F1
Cylinder	Myers's	99.38%	83.33%	83.33%	83.33%
	KTS-DML	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Triangle	Myers's	99.82%	52.17%	52.17%	52.17%
	KTS-DML	**99.96%**	**95.00%**	**82.61%**	**88.37%**
Plane	Myers's	99.76%	64.29%	60.00%	62.07%
	KTS-DML	**99.84%**	**78.57%**	**73.33%**	**75.86%**

much larger than *TP, FP* and *FN* for each case, leading to very high accuracy. Therefore, the following analysis mainly focuses on the other three metrics. As our selection method aims at flow visualization, to better explain the numbers in the tables, the visualization effect of several time steps is presented to make an intuitive comparison among labeled key time steps and the ones selected by KTS-DML and Myers's for Cylinder in Fig. 2, Triangle in Fig. 3 and Plane in Fig. 4. Limited to pages, only 6 time steps in the same period of time for each method are shown, and the index numbers of time steps are given above the pictures accordingly. For each picture, Tecplot is used to exhibit the contour of the v component of velocity and we mainly focus on the changes in the structure of primary features and flow trend in flow fields. Meanwhile, the time lines which show the distribution of selected key time steps in the whole sequence are also presented. The small blocks on these lines denote the selected time steps. A block or index number in red means this selected time step is not a labeled one, while the blue block or index number represents a correctly selected key time step.

(1) Case Cylinder. As the last 20% of this dataset is used as the test set, the range of index numbers is from 1281 to 1600. In this range, 6 key time steps are labeled, which are number 1321, 1368, 1423, 1469, 1524, 1574, respectively. The selection of KTS-DML is different. But as mentioned above, if the selected time step is near the labeled one, a manual check will be made. For example, the first selected time step is 1320 which is different but very close to the labeled 1321. Seeing that the visualization effect of 1320 and 1321 are very similar, it is regarded as a correctly selection. Therefore, the selection of KTS-DML matches all the 6 labeled time steps, resulting in 100% accuracy, precision and recall. The result of Myers's is slightly worse. Though it also selects 6 key time steps, one of them is actually not. The F1 score consequently falls to 83.33%.

(2) Case Triangle. In this case, the whole time sequence can be divided into three stages according to the flow trend. The flow field goes smoothly from the beginning to step 3600 (Stage Smooth), then starts to change violently from step 3601 to 4700 (Stage Violent), and finally enters into the periodical state (Stage Period). Therefore, only 3 key time steps at Stage Smooth and twice the quantity at Stage Violent are labeled. Afterwards, key time steps are labeled about every 500 steps at Stage Period. To better show the performance of the two methods, 6 time steps presented in visualization are selected in the range from step 2000 to 5000, where the state of flow field is from beginning changing to changing greatly.

A 88.37% F1 score shows the good performance of KTS-DML. 19 of 23 labeled time steps are correctly selected and only one wrong selection is made. At Stage Smooth, KTS-DML misses one key time step, number 650. In fact, the flow field in this step is similar to the one in step 2190. The reason why it is labeled is that when we are labeling key time steps, the flow field in this step seems to reach a steady state in visualization. If we come back to do the labeling again, this step may not be selected. At Stage Violent, KTS-DML selects a wrong step, i.e. number 4026, and misses two steps which are 4200 and 4426. The miss of step 4200 dues to the wrong selection of 4026. As they are in the same time interval, our selection method may find 4026 is the one with local minimal similarity. The reason for the miss of step 4426 may lie in that the flow field is about to enter into Stage Period at this step and the

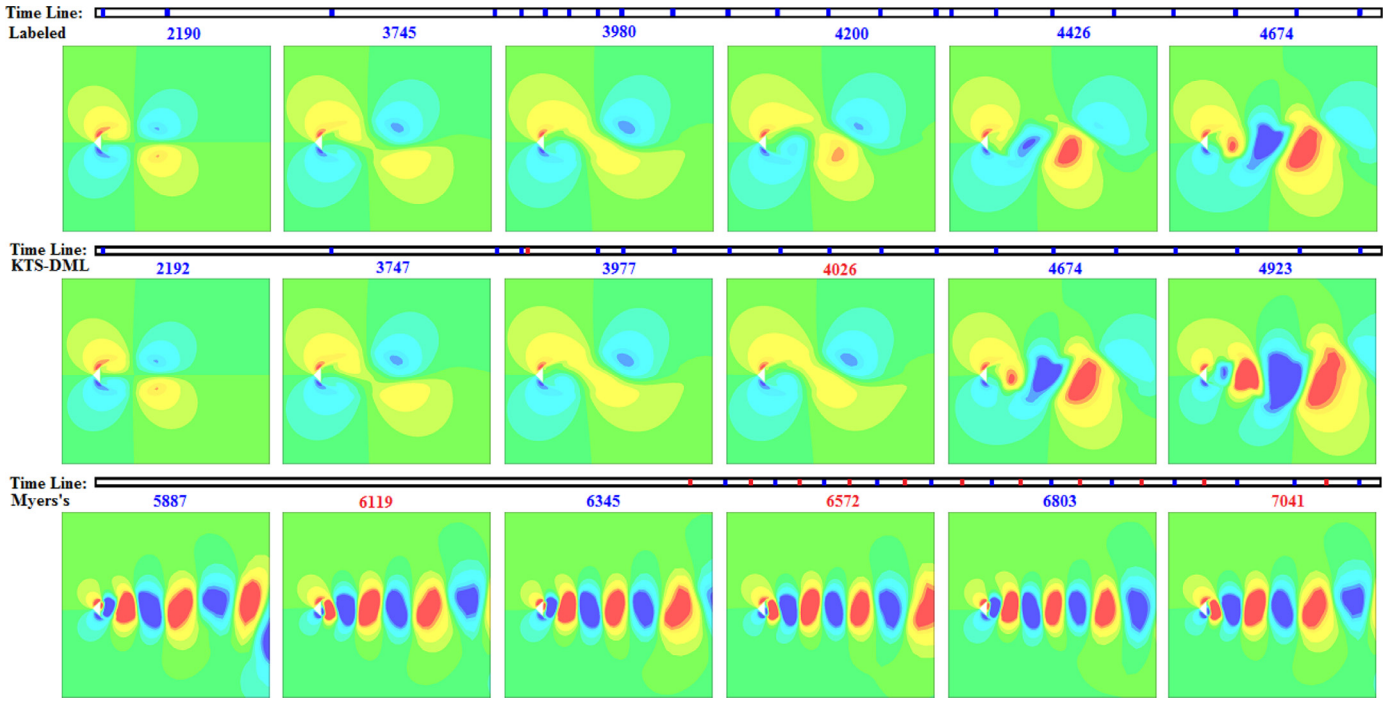

Fig. 3. The selection results of labeled, KTS-DML and Myers's for case Triangle. The labeled and KTS-DML present the selected time steps at Stage Smooth and Violent. With no selection at these two stages, Myers's has to show the ones selected from step 5800 to 7100 at Stage Period.

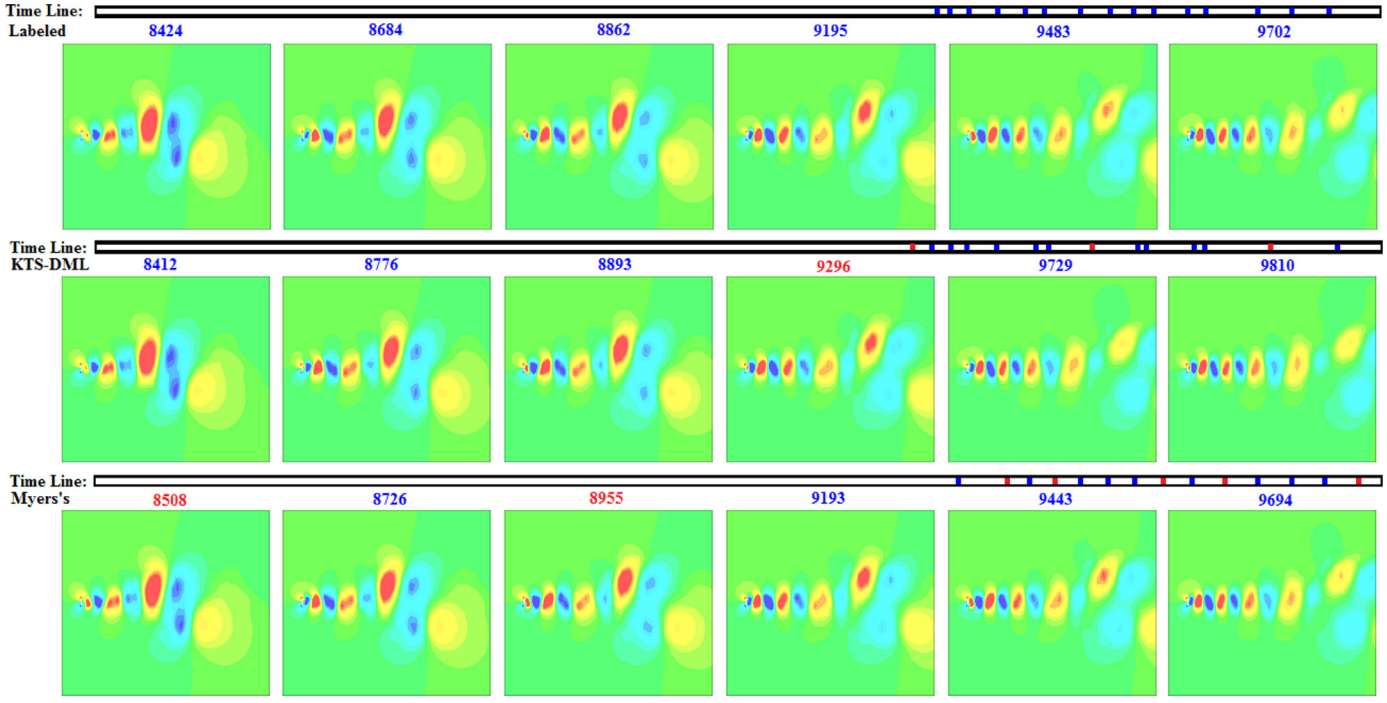

Fig. 4. The selection results of labeled, KTS-DML and Myers's for case Plane. All time steps presented in this case are from Stage Period.

change becomes too small to be detected. There is another miss at Stage Period and that may due to the imperfect selection algorithm proposed in Section 3.3. At Stage Period, the change of flow field is periodical and the pattern is fixed. Therefore, the time interval between two local minimal similarities is relatively fixed and the unknown sudden disturbance at step 7977 is difficult to catch.

Myers's performs not well in this case and gets a 52.17% F1 score. The determinants, Recall and Precision, are both unsatisfactory. The low Recall mainly dues to many misses of labeled key time steps. It can be seen from the time line shown in Fig. 3 that Myers's selects none at Stage Smooth and Violent. The result does not become better after tuning parameters to capture more time steps. Thus, we have to show 6 selected key time steps at Stage Period instead of in the range from step 2000 to 5000 as the labeled and KTS-DML do. The time line also shows that the low Precision of Myers's in this case lies in many wrong selection at Stage Period.

3) Case Plane. In this case, Stage Smooth is from step 1 to 5900, Stage Violent is from step 5901 to 7800, and the rest is Stage Period. A similar situation happens, that is, Myers's finds no key time steps at Stage Smooth and Violent. Therefore, to make a better comparison, only Stage Period is concerned, and the data for case Plane in Table 1, Table 2 and Fig. 4 are only collected at this stage. Even so, Myers's performs worse than KTS-DML. For the 23 labeled key time steps, Myers's misses 6 while KTS-DML misses 4. Besides, these two methods both select 14 time steps, of which 5 are wrong selection for Myers's while only 3 for KTS-DML. Consequently, compared with KTS-DML, Myers's gets a lower precision and a lower recall, which decides the lower F1 score.

What KTS-DML has in common with Myers's is to find the change points which indicate the changing trend of flow fields. But, KTS-DML emphasizes the great changes of flow field data as the labeled key time steps do. For Myers's, the key time steps they select are not expected to be the ones where flow fields vary greatly. This may be the main reason why Myers's performs not well. Though the selection results are better, KTS-DML consumes more time and memory than Myers's. The amount of computation in Myers is so small that the time overhead is negligible. In our method, there is a lot of computation in CNNs which are used to produce better features. The overhead of KTS-DML is about 7.3% of the compute time of the CFD simulation, which is deemed reasonable. For memory requirements, KTS-DML needs about 150MB for Case Cylinder while Myers's only needs 32MB. Similar situations occur in other two cases because their input data is the same size. The disadvantage of KTS-DML is due to the large memory requirements for the model and the layer output in CNNs.

4.3. The generalization of KTS-DML

Training the neural networks is time-consuming. It may take about hours to train a relatively good network for large data with one CPU thread. As a result, from the perspective of time, it is unrealistic to train a network for each case. Therefore, the generalization of KTS-DML need to be demonstrated and we hope that the network trained by one case can be applied to other cases directly. By this way, other cases can avoid long training and be qualified for practical applications. The good generalization of Metric-Net will be shown from following aspects.

(1) When a simulation stops, it may be found that flow field is not completely converged or statistical data are not enough. Then they tend to continue the simulation to achieve a steady state or to get more data. In this situation, the time steps are collected before the stop and used to train the network. Then the network is applied to the time steps after the continuation, as we do in case Cylinder. The time step 960 can be regarded as the stop of the simulation and the continuation produces the following data. As time steps from number 961 to 1280 are used as evaluation set, the test starts from step 1281. The 100% accuracy, precision and recall is a good verification of this generalization, which shows the network trained with the preceding time steps can be applied to the following time steps.

(2) For cases under the same or similar computational conditions, we hope to train the network with the data from a single case and then extend the network to similar cases. In this way, the key time steps of similar cases can be directly in-situ selected with the trained network and the time-consuming training stage is avoided. This can be verified by the results of case Triangle and Plane. After trained with the data which are the 960 time steps from case Cylinder, the network is extended to case Triangle and Plane. The statistical numbers in Table 1, Table 2 and the visualization effect in Fig. 3, Fig. 4 all demonstrate that the network trained with the data of a single case can be applied to similar cases.

Table 3
Results of the validation in statistics.

Case	Method	TP	FP	TN	FN
Cylinder	sel-disa	0	0	314	6
	auto-int-disa	6	0	314	0
	int-ext-disa	6	0	314	0
	KTS-DML	6	0	314	0
Triangle	sel-disa	16	336	11,641	7
	auto-int-disa	19	3	11,974	4
	int-ext-disa	19	7	11,970	4
	KTS-DML	19	1	11,976	4
Plane	sel-disa	10	273	4212	5
	auto-int-disa	9	3	4482	6
	int-ext-disa	10	3	4482	5
	KTS-DML	11	3	4482	4

Table 4
Results of the validation in metrics.

Case	Method	Accuracy	Precision	Recall	F1
Cylinder	sel-disa	98.13%	NAN	0.00%	NAN
	auto-int-disa	100.00%	100.00%	100.00%	100%
	int-ext-disa	100.00%	100.00%	100.00%	100%
	KTS-DML	100.00%	100.00%	100.00%	100%
Triangle	sel-disa	97.14%	4.55%	69.57%	8.53%
	auto-int-disa	99.94%	86.36%	82.61%	84.44%
	int-ext-disa	99.91%	73.08%	82.61%	77.55%
	KTS-DML	**99.96%**	**95.00%**	**82.61%**	**88.37%**
Plane	sel-disa	93.82%	3.53%	66.67%	6.71%
	auto-int-disa	99.80%	75.00%	60.00%	66.67%
	int-ext-disa	99.82%	76.92%	66.67%	71.43%
	KTS-DML	**99.84%**	**78.57%**	**73.33%**	**75.86%**

4.4. The validation of the selection

To validate the effect of the Selection method proposed in Section 3.3, comparisons are conducted between the full functional KTS-DML and the KTS-DML with one technique disabled, and the results are given in Tables 3 and 4. (1) The whole selection is disabled, referred to sel-disa. the key time steps are selected in the most intuitive way, that is, the one with negative similarity is selected. (2) The time interval is fixed during the whole process, instead of being automatically adjusted, referred to auto-int-disa. The intervals are set as 50, 250, 250 respectively according to the computational condition of each case. (3) The interval is not extended though the similarity is negative when the counter reaches the interval, referred to int-ext-disa. Possible problems caused by these disabling have been discussed in Section 3.3 and the results shown in Tables 3 and 4 confirm that these techniques have played a good role in our method.

5. Conclusions and future work

In this paper, a local key time steps selection method for CFD data based on DML is proposed. With a Siamese deep neural network which is composed of two CNNs, Metric-Net is trained to extract features of flow field data to evaluate the similarities between time steps as accurate as possible. A algorithm for key time steps selection is presented to yield a better result. Compared with Myers's, the proposed method achieves higher accuracy, precision and recall. Meanwhile, the good generalization shown in different CFD cases proves the applicability of this method in practical applications.

Considering that only the 2D cases are used in this paper, in future work, the proposed method will be applied to more CFD cases, especially large 3D cases, to test the generalization of neural networks. If the 3D data are so large that the storage system can not afford, it is very meaningful to demonstrate the generalization in which the network trained with small-scale data can be applied

to the large-scale data. Besides, the way to label the similarities may be improved to get a better training result.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (# 2017YFB0202201), the National Nature Science Foundation of China (# 61806205) and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (# 2016ZT06D211).

References

[1] Bauer AC, Abbasi H, Ahrens J, Childs H, Geveci B, Klasky S, Moreland K, O'Leary P, Vishwanath V, Whitlock B, Bethel EW. In situ methods, infrastructures, and applications on high performance computing platforms. Comput Gr Forum 2016;35(3):577–97. doi:10.1111/cgf.12930.

[2] Ayachit U, Bauer A, Duque E, Eisenhauer G, Ferrier N, Gu J, Jansen K, Loring B, Lukic Z, Menon S, Morozov D, O'Leary P, Ranjan R, Rasquin M, Stone C, Vishwanath V, Weber G, Whitlock B, Wolf M, Bethel EW. Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures. In: Proceedings of the international conference for high performance computing, networking, storage and analysis; 2016. p. 921–32. doi:10.1109/SC.2016.78.

[3] Tong X, Lee T-y, Shen H-W. Salient time steps selection from large scale time-varying data sets with dynamic time warping. In: Proceedings of the IEEE symposium on large data analysis and visualization (LDAV). IEEE; 2012. p. 49–56. doi:10.1109/LDAV.2012.6378975.

[4] Wang C, Yu H, Ma K-L. Importance-driven time-varying data visualization. IEEE Trans Vis Comput Graph 2008;14(6):1547–54. doi:10.1109/TVCG.2008.140.

[5] Zhou B, Chiang Y-j. Key time steps selection for large-scale time-varying volume datasets using an information-theoretic storyboard. In: Proceedings of the Eurographics Conference on Visualization (EuroVis), 37; 2018. p. 37–49. doi:10.1111/cgf.13399.

[6] Salloum M, Bennett JC, Pinar A, Bhagatwala A, Chen JH. Enabling adaptive scientific workflows via trigger detection. In: Proceedings of the first workshop on in situ infrastructures for enabling extreme-scale analysis and visualization (ISAV); 2015. p. 41–5. doi:10.1145/2828612.2828619.

[7] Myers K, Lawrence E, Fugate M, Bowen CM, Ticknor L, Woodring J, Wendelberger J, Ahrens J. Partitioning a large simulation as it runs. Technometrics 2016;58(3):329–40.

[8] Ling J, Kegelmeyer WP, Reed KA, Shead TM, Iv WLD. Using feature importance metrics to detect events of interest in scientific computing applications. In: Proceedings of the IEEE international symposium on signal processing and information technology (ISSPIT); 2017. p. 55–63. doi:10.1109/ISSPIT.2018.8642690.

[9] Banesh D, Wendelberger J, Petersen M, Ahrens J, Hamann B. Change point detection for ocean eddy analysis. In: Proceedings of the Workshop on Visualisation in Environmental Sciences, EnvirVis; 2018. p. 27–33.

[10] Larsen M, Woods A, Marsaglia N, Harrison C, Childs H. A flexible system for in situ triggers. In: Proceedings of the workshop on in situ infrastructures for enabling extreme-scale analysis and visualization (ISAV); 2018. p. 1–6. doi:10.1145/3281464.3281468.

[11] Bengio Y. Learning deep architectures for ai. Found Trends Mach Learn 2009;2(1):1–127. doi:10.1561/2200000006.

[12] Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw 2015;61:85–117. doi:10.1016/j.neunet.2014.09.003.

[13] Dara S, Tumma P. Feature extraction by using deep learning: a survey. In: Proceedings of the 2nd international conference on electronics, communication and aerospace technology, ICECA; 2018. p. 1795–801. doi:10.1109/ICECA.2018.8474912.

[14] Farias G, Dormido-Canto S, Vega J, Ratt G, Vargas H, Hermosilla G, Alfaro L, Valencia A. Automatic feature extraction in large fusion databases by using deep learning approach. Fusion Eng Des 2016;112:979–83. doi:10.1016/j.fusengdes.2016.06.016.

[15] Jiang B, Yang J, Lv Z, Tian K, Meng Q, Yan Y. Internet cross-media retrieval based on deep learning. J Vis Commun Image Represent 2017;48:356–66. doi:10.1016/j.jvcir.2017.02.011.

[16] Hayakawa Y, Oonuma T, Kobayashi H, Takahashi A, Chiba S, Fujiki NM. Feature extraction of video using deep neural network. In: Proceedings of the IEEE 15th international conference on cognitive informatics cognitive computing (ICCI*CC); 2016. p. 465–70. doi:10.1109/ICCI-CC.2016.7862078.

[17] Mohsen H, El-Dahshan E-SA, El-Horbaty E-SM, Salem A-BM. Classification using deep learning neural networks for brain tumors. Future Comput Inform J 2018;3(1):68–71. doi:10.1016/j.fcij.2017.12.001.

[18] Guo X, Li W, Iorio F. Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. In: KDD '16. New York, NY, USA: ACM; 2016. p. 481–90. doi:10.1145/2939672.2939738.

[19] Ling J, Kurzawski A, Templeton J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J Fluid Mech 2016;807:155–66. doi:10.1017/jfm.2016.615.

[20] Lee S, You D. Prediction of laminar vortex shedding over a cylinder using deep learning, 2017, arXiv:1712.07854.

[21] Liu Y, Wang Y, Deng L, Wang F, Liu F, Lu Y, Li S. A novel in situ compression method for CFD data based on generative adversarial network. J Vis 2019;22(1):95–108. doi:10.1007/s12650-018-0519-x.

[22] Liu Y, Lu Y, Wang Y, Sun D, Deng L, Wang F, Lei Y. A CNN-based shock detection method in flow visualization. Comput Fluids 2019;184:1–9. doi:10.1016/j.compfluid.2019.03.022.

[23] Deng L, Wang Y, Liu Y, Wang F, Li S, Liu J. A CNN-based vortex identification method. J Vis 2019;22(1):65–78. doi:10.1007/s12650-018-0523-1.

[24] Monfort M, Luciani T, Komperda J, Ziebart B, Mashayek F, Marai GE. A deep learning approach to identifying shock locations in turbulent combustion tensor fields. Model Anal Vis Anisotropy 2017:375–92.

[25] Kulis B. Metric learning: a survey. Found Trends® Mach Learn 2012;5(4):287–364. doi:10.1561/2200000019.

[26] Kulis B, Jain P, Grauman K. Fast similarity search for learned metrics. IEEE Trans Pattern Anal Mach Intell 2009;31(12):2143–57. doi:10.1109/TPAMI.2009.151.

[27] Frome A, Singer Y, Sha F, Malik J. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: Proceedings of the IEEE 11th international conference on computer vision; 2007. p. 1–8. doi:10.1109/ICCV.2007.4408839.

[28] Davis JV, Dhillon IS. Structured metric learning for high dimensional problems. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. In: KDD. New York, NY, USA: ACM; 2008. p. 195–203. doi:10.1145/1401890.1401918.

[29] Yi D, Lei Z, Li ISZ. Deep metric learning for practical person re-identification. In: Proceedings of the 22nd international conference on pattern recognition (ICPR); 2014. p. 34–9.

[30] Bromley J, Guyon I, LeCun Y, Sackinger E, Shah R. Signature verification using a siamese time delay neural network. Int J Pattern Recognit Artif Intell 1993;7(4):669–88. doi:10.1159/000463903.

[31] Baldi P, Chauvin Y. Neural networks for fingerprint recognition. Neural Comput 1993;5:402–18. doi:10.1162/neco.1993.5.3.402.

[32] Frey S, Ertl T. Flow-based temporal selection for interactive volume visualization. Comput Gr Forum 2017;36(8):153–65. doi:10.1111/cgf.13070.

[33] Woodring J, Shen H. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. IEEE Trans Vis Comput Graph 2009;15(1):123–37. doi:10.1109/TVCG.2008.69.

[34] Lee T, Shen H. Visualization and exploration of temporal trend relationships in multivariate time-varying data. IEEE Trans Vis Comput Graph 2009;15(6):1359–66. doi:10.1109/TVCG.2009.200.