A Deep Learning Approach to Identifying Shock Locations in Turbulent Combustion Tensor Fields

Mathew Monfort, Timothy Luciani, Jonathan Komperda,
Brian Ziebart, Farzad Mashayek, G. Elisabeta Marai

Abstract We introduce a deep learning approach for the identification of shock locations in large scale tensor field datasets. Such datasets are typically generated by turbulent combustion simulations. In this proof of concept approach, we use deep learning to learn mappings from strain tensors to Schlieren images which serve as labels. The use of neural networks allows for the Schlieren values to be accurately approximated more efficiently than calculating the values from the density gradient. In addition, we show that this approach can be used to predict the Schlieren values for both two-dimensional and three-dimensional tensor fields, potentially allowing for anomaly detection in tensor flows. Results on two shock example datasets show that this approach can assist in the extraction of features from reacting flow tensor fields.

1 Introduction

The design of efficient combustion systems requires an in-depth understanding of the underlying physics that occur within combustion chambers. The challenge is significantly escalated for supersonic combustion due to the presence of shocks and other waves. Considering that turbulent flows are inherently three-dimensional and shocks occur over very thin regions of the order of one mean free path, the presence of shocks introduces additional complexities to a flow that is characterized by a multitude of scales in time and space. Computational methods typically use artificial diffusion to smear the shocks such that they can be captured on a grid that is more coarse than the thickness of the shock [3]. Despite this approximation, the simulation of supersonic turbulent combustion is not only computationally demanding but also able to produce large quantities of data.

University of Illinois at Chicago, Chicago, IL 60607, e-mail: mathewmonfort@gmail.com, {tlucia2, jonk,bziebart,mashayek,gmarai}@uic.edu

As discussed in detail in our previous work [36], an important aspect of turbulence modeling and model validation involves analysis of the subgrid scale stress tensor or the stress tensor. Unfortunately, the scale of these datasets is very large; it is necessary to capture all the scales of turbulence, combustion, as well as the discontinuities. For example, Reynolds Averaged Navier-Stokes (RANS) simulations of the scramjet engine have used in excess of 33 million cells [15, 46, 16]. Meanwhile, Large eddy simulation (LES) of a simplified scramjet engine model have required upwards of 6 million cells [6], and preliminary LES simulations of a full scramjet at low Reynolds numbers have used 14 million cells [8]. It is expected that high-fidelity LES simulations of the full scramjet geometry at high Reynolds numbers will exceed 100 million cells. These datasets cannot be output frequently for visualization because of the computational cost of writing the files. When one considers that the computational cost of such a simulation can easily exceed 17,000 core-hours, it becomes necessary to limit disk-intensive operations to preserve core-hours [29].

Feature extraction can be a useful approach to reducing the size of the dataset to visualize. In this approach, a small subset of points are identified as features of interest and output for visualization. While multiple filtering techniques exist for flow shock feature extraction, some of these methods incorrectly identify regions of turbulence as shocks, and conversely some turbulence models incorrectly identify shocks as turbulence [17]. Most importantly, filtering techniques generally do not have the ability to identify abnormalities in the flow in the absence of expert knowledge input. As a result, it is necessary to investigate more flexible approaches to analyzing discontinuities in the flow that may correspond to shockwaves, including approaches which can identify normal shock behavior and abnormalities. Such anomaly detection tasks tend to require a statistical approach.

In this work we investigate the potential and limitations of deep learning [5, 30]—a machine learning technique based on learning representations of the data—with respect to shock feature extraction from strain tensor values. Deep learning has been successfully used in a variety of applications, including image analysis and speech recognition, and, beyond feature extraction, has the potential to capture abnormalities in the data. Here we take an exploratory first step in this direction by investigating the feasibility of using deep learning to retrieve shock locations. Future work will investigate the ability of deep learning with respect to anomaly detection.

2 Background and Related Work

2.1 *Stress and Strain Tensors*

A tensor is an extension of the concept of a scalar and a vector to higher orders. Scalars and vectors are 0-th and 1-st order tensors, respectively. In general, a k-th order tensor can be represented by a k-dimensional array, *e.g.* a second order tensor is a two-dimensional array (a matrix). For example, while a stress *vector* is the force acting on a given unit surface, a stress *tensor* is defined as the components of stress vectors acting on each coordinate surface; thus stress can be described by a symmetric 2-nd order tensor.

The velocity stress and strain tensor fields are manifested in the transport of fluid momentum, which is a vector quantity governed by the following conservation equation:

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}, \qquad \text{for } i = 1,2,3 \tag{1}$$

where the Cartesian index notation is employed in which the index $i = 1,2,3$ represents spatial directions along the x, y, and z Cartesian coordinates, respectively; and the repeated index j implies summation over the coordinates. t is time, ρ is the fluid density, $\mathbf{u} \equiv [u_1, u_2, u_3]$ is the Eulerian fluid velocity, p is the pressure, and τ is the stress tensor defined as:

$$\tau_{ij} = 2\mu \left(S_{ij} - \frac{1}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k} \right) \tag{2}$$

where μ is the dynamic viscosity coefficient (a fluid-dependent parameter) and S is the velocity strain tensor defined as:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{3}$$

As indicated by the definitions above, both the stress and strain tensor are two-dimensional, symmetric, positively-defined arrays. Density, along with the three velocity components and total energy, are the primary variables calculated in the code. All other information, including the stress tensor and velocity strain tensor, are secondary variables calculated from these primary variables.

2.2 Shock Feature Extraction

Most feature extraction techniques fall into one of three basic categories. The most widespread method uses feature attributes such as mass, centroid, volume, texture, or moment of inertia [50, 51, 52, 10]. A number of filtering techniques specifically for detecting and visualizing shocks waves have been developed, including using shock surface alignment to the pressure gradient vector [33], and using the density gradient in the direction of the velocity [44, 34]. A second approach to feature extraction uses isosurfacing in higher dimensions [24]. A third class of approaches uses various machine learning techniques to aid in feature tracking. Tzeng and Ma [58] utilize neural networks to learn which transfer functions are most appropriate in tracking the features of interest. Ozer et al. [43] use a clustering algorithm to group features based on similarity measures. In our previous work [36], we introduced a large scale K-Means clustering approach to define and track regions of interest. Our approach is similar to these last category approaches in that we also utilize machine learning.

The approach we use to generate labels for the tensor data is based on the Schlieren filter [35]. The density-gradient of combustion datasets relates indirectly to the stress tensor through the conservation equation [35]. Such density-gradient descriptors can be used to generate flow visualizations in the style of Schlieren images [19], and have been shown to accurately reflect shock boundaries [35].

The use of the Schlieren computation is intended as an exploratory first step in investigating the ability of deep learning to identify shockwaves and other features in CFD datasets. While the computation of the Schlieren itself is not costly enough to justify a learning alternative, more accurate shock prediction methods are significantly more costly and difficult to pose numerically. The ultimate goal of the future work would be to employ a deep learning approach to directly pinpoint and differentiate different phenomena at a lower computational cost than the many sensors currently available in literature.

2.3 Deep Learning

In 1943 McCulloch and Pitts [39] introduced a set of simplified computational models of biological networks. These ideas were soon extended to include models of how these networks might learn including Hebbian learning [20], multilayer perceptrons [25] and eventually backpropagation [49]. However, the extensive computational complexity of training large networks [41] prohibitively limited their usefulness.

Deep architectures yield a greater expressive power than shallow networks since functions that can be compactly represented by an architecture of depth k, would require an exponential number of computational elements to be represented by an architecture with a depth of $k-1$ [5]. This breadth for depth trade-off allows deep architectures to represent a wide family of functions with reduced complexity and improved generalization. Each succeeding layer of the network combines the features of the previous layer forming a higher level abstraction of the features in the preceding layer. This increasing level of abstraction from layer to layer allows deep networks to produce strong generalizations for highly varying functions. The difficulty lies in efficiently training the large number of parameters needed to form the network [41].

Convolutional networks are specific types of deep architectures inspired by the structure of the visual cortex [37, 30] which do not suffer from the typical convergence issues of other deep architectures [5]. The defining characteristic of convolutional networks is the use of local receptive fields with shared parameters. These fields are used to scan input features with a two dimensional structure and form feature maps that capture low-level (e.g., edges and corners) representations of the input. This process is then repeated in each succeeding layer allowing for the formation of progressively higher-level abstractions. This two-dimensional representational power of convolutional networks has allowed them to dominate the field of computer vision [28, 55].

Volumetric convolutional networks [40, 38, 47] are an extension of convolutional surface-based architectures to input features with three dimensional structure. They have successfully been applied to the area of object recognition [38] and MRI segmentation [40]. The ability of these architectures to take advantage of three dimensional structure directly translates to the problem of inferring Schlieren features in combustible fluids.

2.4 Application Domain Background

Supersonic combustors, such as scramjet engines, are prime examples of flows where turbulence and combustion interact with shock waves [46, 15, 48]. However, the concurrent presence of shock waves and turbulence in the simulation of supersonic turbulent flow presents additional challenges compared to subsonic flows; the numerical methods designed to treat these properties must predict the presence and capture these features accurately [17]. The inability to accurately predict these features contributes to the many unresolved fundamental issues that surround supersonic combustors, such as the scramjet. Flames in the presence of shocks are known to become distorted, generate vorticity, and break up or stretch [26, 27]. Additionally, when turbulence is seen in the presence of a shock, there is an amplification of velocity fluctuations [8]. The inability to accurately predict these behaviors affects the remainder of the solution domain, thereby causing a failure to compare well against experiment.

While numerical simulations can provide an acceptable prediction of turbulence behavior, either through the use of large eddy simulation or direct numerical simulation, there is still a significant challenge in the simulation of flows involving flow discontinuities, such as shocks. Methods for numerically modeling supersonic turbulent combustion[23, 4] rely on the accurate prediction of the shock location. Current methods for shock capturing may rely on an artificial viscosity to dissipate the shock, to smear it across many solution points so that it may be captured [1, 2, 3]. However, incorrectly predicting the location of the shock may have unintended side effects, such as adding the artificial viscosity in the incorrect regions, thereby dissipating other regions of the flow. Research has been performed in developing sensors to accurately predict the shock location [14, 3, 17]. However, some of these methods incorrectly identify regions of turbulence as shocks, and conversely some turbulence models incorrectly identify shocks as turbulence [17]. As a result, it is necessary to investigate more flexible approaches to analyzing discontinuities in the flow that may correspond to shockwaves, including approaches which can identify normal shock behavior and abnormalities.

3 Methods

To study the applicability of deep learning to the feature identification problem, we trained convolutional neural networks to learn a mapping from strain tensors to Schlieren values [35] for each time-step in a turbulent flow. To accomplish this we form a regression network similar to an auto-encoder [9, 21] where instead of learning to replicate the strain tensors used as input, we learn to construct the associated Schlieren values for each time step. The strain tensor is calculated with Equation (3) above [35]. Additionally, the Schlieren value for each pixel is derived with the following equation:

$$\text{Schlieren}(x,y,z) = \beta e^{-\frac{k|\nabla\rho|}{|\nabla\rho|_{\max}}}, \tag{4}$$

where x, y, and z are the position coordinates, β and k are rendering parameters set to 0.8 and 20 respectively, and Δp is the gradient of the density field.

We will examine this approach on two datasets, a three-dimensional Sod dataset and a two-dimensional blast dataset. The differing spatial dimensions of both of these domains requires the formation of two different network architectures. In this section we will describe the methods used to construct and train both of these networks.

3.1 Data Processing

3.1.1 Three-Dimensional Sod Dataset

The three-dimensional Sod problem is one form of a shock tube problem, which is frequently considered a benchmark test for shock capturing methods. It is also commonly used for testing compressibility terms in numerical codes due to its inclusion of spatial pressure variation [32]. The initial condition contains a driver and driven gas separated by a diaphragm in the center. When the diaphragm breaks, at time zero, a discontinuity forms and travels to the end of the tube. The final solution, which is available as an analytical solution, consists of rarefaction, contact, and shock waves [57].

For each time-step (of a total of 1,775 steps in our experiments) we discretize the position coordinates of the Sod dataset into a 804x4x4 volume and for each position calculate the strain tensor S and the Schlieren using Equations (3) and (4).

We then represent each of the six unique values of the strain tensor as a different channel of a three dimensional image (6x804x4x4) and the Schlieren value as a single channel image (1x804x4x4). Representing the data in this form allows for us to use computer vision techniques such as volumetric convolutional networks to learn a mapping from the strain tensors to the Schlieren values.

With the data in this format we place every tenth time-step into a test set (177 total) and randomly separate the 90% remaining time-steps into a training set (1,416 total) and 10% into an evaluation set (182 total) that is used to determine when the network training has converged.

3.1.2 Two-Dimensional Blast Dataset

The second problem considered is a two-dimensional explosion. The initial condition consists of a high-density, high-pressure region located inside of a circle in the center of the geometry and low-density, low-pressure in the remainder of the computational domain. The two regions are joined by a discontinuity, which travels outwards in time, forming shock, contact, and rarefaction waves. Comparison along the radial directions gives virtually identical results due to the problem's symmetry, and the resolution of discontinuities that travel in all directions is the same as that in the one-dimensional Sod problem [57]. The solution of this problem requires high resolution throughout the domain due to the sharp discontinuities traveling in all spatial directions.

For each time-step of a total of 158 steps we discretize the position coordinates into a 70x70 surface and for each position calculate the strain tensor S using Equation (3) and the Schlieren value with Equation (4).

We then represent each of the six unique values of the strain tensor as a different channel of an image (6x70x70) and the Schlieren value as a single channel image (1x70x70). As with the three-dimensional dataset we place every tenth time-step into a test set (15 total) and randomly separate the 90% remaining time-steps into a training set (133 total) and 10% into an evaluation set (ten total) that is used to determine when the network training has converged.

3.2 Network Architecture

For both the three-dimensional and the two-dimensional dataset we constructed an eight-layer all convolutional network [53] that is divided into two parts; a feature extractor that learns a function for condensing the input features into a low-dimensional feature vector, and an image constructor that learns a function that transforms the low-dimensional feature vector calculated in the previous part into a schlieren image. We construct the network in this way in order to improve generalization by forcing the network to learn a sparse feature representation with useful (general) features of the strain tensors. The ultimate structure is chosen by tuning the hyper-parameters on the evaluation set [7]. The main difference between the two networks is the use of volumetric convolutions for the three-dimensional dataset and regular convolutions for the two-dimensional dataset.

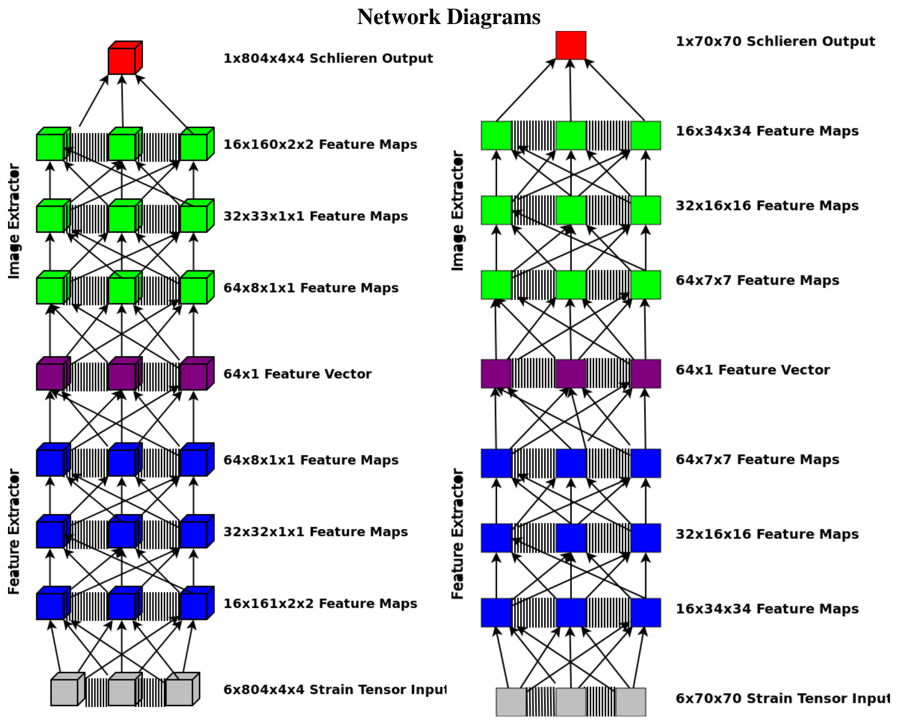

Fig. 1 Networks for three-dimensional Sod dataset (left) and the two-dimensional Blast dataset (right). Each network comprises two parts; a feature extractor that learns a function for condensing the input features into a low-dimensional feature vector, and an image constructor that learns a function that transforms the low-dimensional feature vector calculated in the previous part into a Schlieren image.

The feature extractor for both networks consist of four strided volumetric convolutional layers that reduce the input strain tensors into a 64 neuron feature vector. Strided convolutions incorporate regularization into the convolutional layers while improving the efficiency in network performance [54] when compared to standard max-pooling based sub-sampling. The image extractor consists of an inverse mapping, sometimes referred to as deconvolutional layers [42], with matching strides and kernel sizes as the feature extractor. The key difference is that the image extractor outputs a single channel image compared to the six channel input of the feature extractor. Figure 1 details the structure of both of the networks with a feature extractor reducing the input strain tensors to a low dimensional vector of 64 features and the image extractor building the Schlieren from the feature vector. Additionally, we use Exponentiated Linear Units (ELU) [11] as our activation function for both networks which leads to improved efficiency and performance while addressing the vanishing gradient problem in training deep networks.

3.3 Optimization

We train our network parameters to minimize the mean squared error between the predicted Schlieren values and the ground truth values and optimize the weights with an adaptive learning rate calculated using the adadelta optimization function [59]. To avoid large gradient updates that may be caused from learning regression values, we incorporate gradient clipping [45] to constrain the gradient norm to lie within a specific threshold (i.e., $[0,1]$).

In order to improve the efficiency of our training routine we use spatial batch normalization [22] to normalize the feature maps generated after each convolutional layer. This ensures that the input distribution for each layer is consistent (zero mean and unit variance) which greatly improves learning performance.

Convergence of the network training is determined using a five-step average windowed delta loss on the evaluation set. When the average delta loss (mean squared error) on the evaluation set is positive, meaning the network is not improving its ability to construct the Schlieren, learning is stopped. This ensures that we do not over-train on the training set and preserve the networks ability to generalize.

3.4 Regularization

When training a system on a limited amount of data (or an unbalanced dataset) there is a risk of over-training on the training set and losing the ability to generalize to new instances. For this reason we incorporate spatial dropout [56] after each of the convolutional layers in the network. During training this randomly drops entire feature maps in forward propagation (by setting all values to 0). This forces the network to learn a sparsified representation of the feature vector (condensed low-dimensional feature representation) leading to improved generalization.

4 Results

We evaluate our approach on two datasets, a three-dimensional Sod dataset and a two-dimensional Blast dataset as described in Section 3.1. In this section we describe our results in training the networks and examine the features the networks learned.

It is important to note that while calculating the Schlieren for the three-dimensional Sod dataset via Equation (4) takes an average of 550 ms per time step on an Intel Xeon E5-2697 2.6 GHz processor, our trained network generates the Schlieren in less than 107 ms on the same CPU processor (respectively less than 1 ms on a GeForce GTX 1080 GPU) running the Torch framework [12], an open source machine learning library and scientific computing framework that provides a range of algorithms for deep machine learning. Torch uses the scripting language LuaJIT and an underlying C implementation. This increase in efficiency becomes extremely significant as datasets grow to include hundreds of thousands of time-steps. In total the three-dimensional network trained for about 35 seconds on the Sod dataset and the two-dimensional network trained for about six seconds on the Blast dataset before convergence. Nevertheless, the main strength of the deep learning approach lies in its potential for anomaly detection, which the Schlieren filter cannot do.

4.1 Training

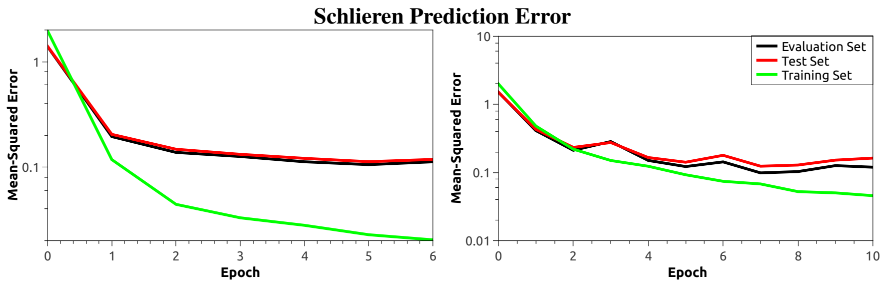

Fig. 2 The average error of the network generated Schlieren for both the three-dimensional Sod dataset (left) and the two-dimensional Blast dataset (right) after each training epoch. The Y axis represents the mean squared error of the Schlieren value and it is plotted in log-scale for clarity. A standard Schlieren carries units of density gradient (kg/m^4), however the flow simulation code uses a normalized function and as such in this case the Schlieren is unitless.

Figure 2 shows the prediction (mean-squared) error on the training, evaluation, and test set for both datasets after each training epoch. We can clearly see that the networks perform very well on the training sets and converge quickly on the evaluation sets. The test error matches closely with the evaluation error indicating that the evaluation error is a strong representation of the networks ability to generalize to the test set. In fact we were able to achieve an average mean-squared error of 0.14 on the three-dimensional Sod test set and 0.12 on the two-dimensional Blast test set. These results suggest that an adequately trained network has the ability to detect anomalies in the tensor field by comparing the predictive error of the network to the true Schlieren values for each time step. A large error would signal a possible deviation from the regular tensor flow indicating anomalous behavior.

4.2 Learned Network Features

In this section we show examples of the features that the two-dimensional network learns for the Blast dataset. We restrict this section to the two-dimensional dataset for the sake of visual clarity as viewing images of the low resolution volumetric features of the Sod dataset in two dimensions is not very informative. However, the conclusions from the two-dimensional dataset match those of the three-dimensional dataset in that the network learns strong feature representations of the input strain tensors allowing for accurate constructions of the associated three-dimensional Schlieren images.

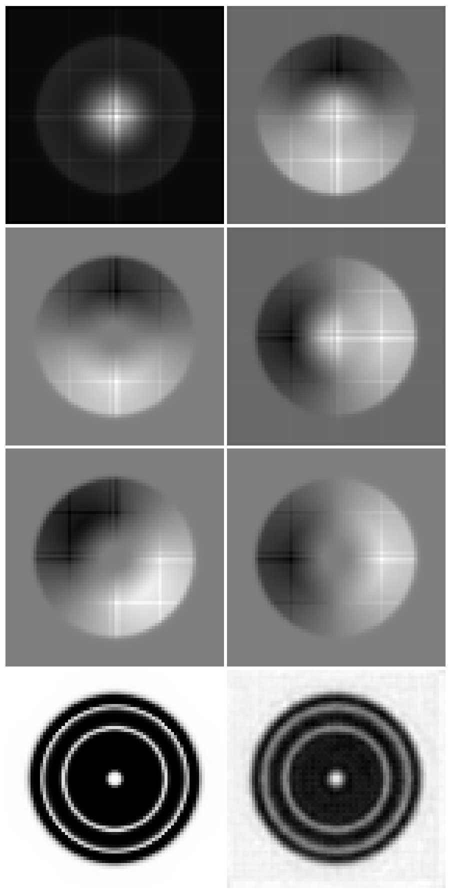

Fig. 3 The six channels of the strain tensor used as input for step 120 followed by the ground truth Schlieren (bottom left) and the Schlieren generated by the network (bottom right).

As an example from the test set, Figure 3 displays the six channel strain tensor (calculated via Equation (3)) used as input into the network for time step 120 followed by the true Schlieren image (calculated via Equation (4) and the Schlieren image generated by the network. We can see that the output of the network (bottom right) matches closely with the calculated Schlieren (bottom left) with a clear separation between high and low-density regions. The fact that we can develop such a close approximation of the desired Schlieren for a time step in the test set is very impressive. Deep networks tend to require large amounts of data to generate strong results indicating that increasing the size of our training set will further improve our predictions.

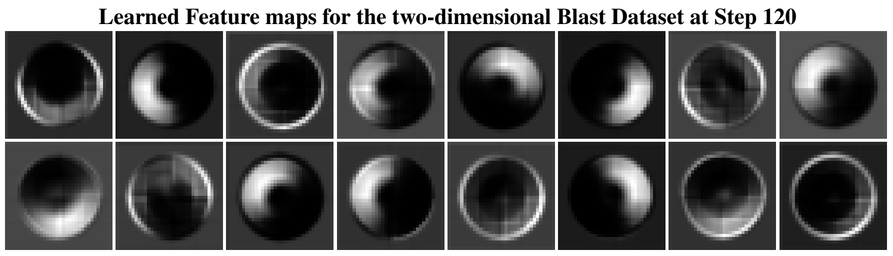
Learned Feature maps for the two-dimensional Blast Dataset at Step 120

Fig. 4 Feature maps in the first layer of the feature extractor.

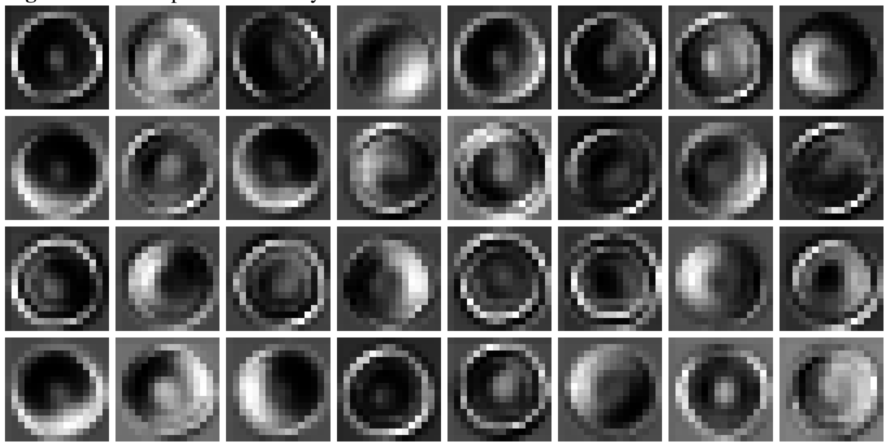

Fig. 5 Feature maps in the second layer of the feature extractor.

Figures 4 and 5 show examples of feature maps in the first two layers of the feature extractor for time step 120 from the test set (we restrict our view to the first two layers due to the low resolution of feature maps in deeper layers). We observe that the network has learned the informative features of each unique strain tensor and combined them to form feature maps with the necessary information for constructing the Schlieren image. The specialization of these feature maps allows for the network to learn a general representation of the input data which is very important for accurately predicting the Schlieren values in unseen time steps. Additionally, these feature maps serve as a set of dictionary references with increasing abstraction (by layer) that allow the network to properly condense the input tensors into a general low-dimensional feature vector as shown in Figure 1.

4.3 Output Visualization

The output of the network produces a reconstructed volume for every tenth time-step of both datasets (test cases), where each point corresponds to the floating-point Schlieren value at that location in the discretized grid. In our experiments, we evaluated 177 and 15 output Schlieren volumes for the three-dimensional and two-dimensional datasets, respectively.

To analyze the results, we visualize each volume as a two-dimensional pseudocolor image that encodes the Schlieren value between two diverging colors – red and black. Similar to grayscale photos, pseudocolor images map intensity to a color scale that ranges from the minimum and maximum value of a data sample [13].

We create three pseudocolor images for each reconstructed Schlieren volume to compare it against the input data and its ground truth. Figure 6 shows the analysis of the output corresponding to the three time-steps of the Blast dataset. Note how the distinct dark areas in the reconstructed images (right) match those of the ground truth (left). These areas signify the occurrence of large density gradients across data samples that correlate with potential shock locations. Figure 7 shows the similar ground truth and reconstruction for the three-dimensional dataset. In this figure, note the three distinct shock zones.

The noise in the Schlieren generated from the network in Figure 6 is a symptom of the low amount of training data in this particular dataset. We trained on data from a single combustion run for both the three-dimensional and two-dimensional settings. Expanding the training data will likely improve results, as is the case in most deep learning applications. This paper is meant as a proof of concept that deep learning is a feasible tool for generating the Schlieren from strain tensors and had potential for identifying anomalies in the combustion fields. Additional image processing may create a clearer Schlieren, as would an increase in the amount of training data.

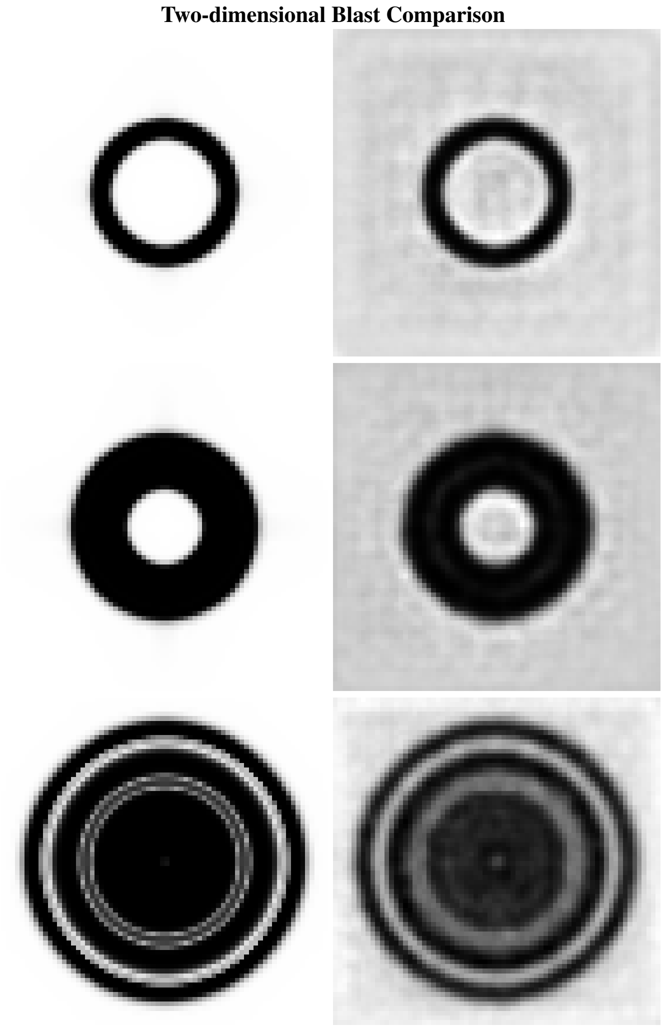

Fig. 6 The network comparison between the network ground truth (left) and the final reconstruction (right) for the 1st, 50th, and 150th time-step of the two-dimensional Blast dataset. The dark areas seen in the images indicate large density gradients corresponding to potential shock regions.

5 Discussion and Conclusion

As previously discussed, the goal of this project was to examine the potential of using deep learning on tensor field data generated by turbulent combustion simulations. First, we were interested in finding out whether a supervised approach can detect structures in the data, and whether these structures correlate with the regions

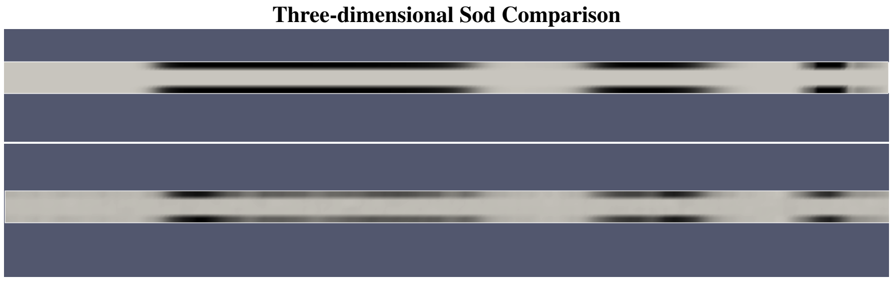

Fig. 7 The network comparison between the network ground truth (top) and the final reconstruction (bottom) for the 100th time-step of the three-dimensional Sod dataset. The dark narrow bands seen in the two images indicate large density gradients, and correspond to potential shock regions. Note the three distinct shock zones in the images.

of interest. Second, we wanted to examine whether this problem is computationally feasible. The answer to both questions is affirmative.

In summary, we found that the deep learning approach can effectively capture and construct shock features. The results indicate that deep networks have the ability to identify anomalies in the tensor flow. Furthermore, the efficiency of the machine learning algorithm exceeds that of calculating the Schlieren via Equation (4). With a greater than 5x improvement on time complexity for the three-dimensional Sod dataset. The efficiency of the machine learning algorithm leads to a 500x improvement when running on the GPU, while optimized CPU to GPU throughput-computing transfers lead to only a 2.5x improvement on average [31]. A small deviation in the predicted values shows that deep learning has the potential to be an effective tool for efficiently generating visualizations of large tensor fields.

It is important to note that this work is exploratory and is meant to be a proof of concept for the use of deep networks in visualizing large tensor fields. While the results are strong, we trained and tested the networks on the same time sequences (with separate training and testing time steps). Predicting time steps in sequences that were not previously trained on may be a more difficult task, However, the results in this chapter were generated with a very limited dataset, 1,775 and 158 time-steps for the Sod and Blast datasets respectively, and deep networks usually require very large amounts of data to be effectively trained [55]. This implies that while the predictive performance may drop with separate training and testing sequences, it may also improve with the inclusion of more data. Additionally, the datasets used in this chapter are sequential in nature lending to the notion that the results may improve with deep architectures that can take advantage of sequential features in their predictions such as recurrent neural networks [18]. Further exploration of this area is needed in order to form a decisive conclusion.

In conclusion, we have introduced a supervised machine learning approach for the segmentation of shocks in large scale tensor field datasets generated by computational turbulent combustion simulations. The approach employs a deep learning architecture based on volumetric convolutional networks. Our evaluation on two rich combustion datasets shows this approach can assist in the visual analysis of the combustion tensor field and that it is more effective than direct filtering calculations. Most importantly, the approach has potential for the detection of anomalies in the data.

Acknowledgements This work was partially supported by the National Science Foundation through award NSF CAREER IIS-1541277. We gratefully acknowledge the NSF Graduate Research Fellowship program for supporting Tim. We thank Adrian Maries and Shiwangi Singh for the initial literature search, and the Electronic Visualization Lab for their feedback and support. We further thank the Peyman Lab for the original motivation behind this line of work, and the Dagstuhl 16142 and 14082 seminars run by the Leibniz Center for Informatics for the many useful discussions regarding tensor field analysis and visualization.

References

1. H. Abbassi, J. Komperda, F. Mashayek, and G. Jacobs. Application of entropy viscosity method for supersonic flow simulation using discontinuous spectral element method. *AIAA Paper 2012-1115*, 2012.
2. H. Abbassi, F. Mashayek, and G. Jacobs. Entropy viscosity approach for compressible turbulent simulations using discontinuous spectral element method. *AIAA Paper 2014-0947*, 2014.
3. H. Abbassi, F. Mashayek, and G. Jacobs. Shock capturing with entropy-based artificial viscosity for staggered grid discontinuous spectral element method. *Computers & Fluids*, 98:152–163, 2014.
4. A. Banaeizadeh, Z. Li, and F. Jaberi. Compressible scalar filtered mass density function model for high-speed turbulent flows. *AIAA Journal*, 49(10):2130–2143, 2011.
5. Y. Bengio and Y. Lecun. *Scaling Learning Algorithms toward AI*, pages 321–359. MIT Press, 2007.
6. M. Berglund and C. Fureby. Les of supersonic combustion in a scramjet engine model. In *Proceedings of the Combustion Institute 31*, pages 2497–2504. The Combustion Institute, Pittsburgh, PA, 2007.
7. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, Feb. 2012.
8. I. Bermejo-Moreno. Subgrid-scale modeling of shock-turbulence interaction for large-eddy simulation. Ann. res. briefs, Center for Turbulence Research, Stanford University, Stanford, CA, 2009.
9. H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4):291–294, 1988.
10. J. Caban, A. Joshi, and P. Rheingans. Texture-based feature tracking for effective time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1472–1479, Nov. 2007.
11. D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.
12. R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.
13. L. Dimitrov. Pseudo-colored visualization of eeg activities on the human cortex using mri-based volume rendering and delaunay interpolation. 1995.
14. F. Ducros, V. Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, and T. Poinsot. Large-eddy simulation of the shock/turbulence interaction. *Journal of Computational Physics*, 152:517–549, 1999.

15. J. R. Edwards, A. Potturi, and J. A. Fulton. Large-eddy / reynolds-averaged navier-stokes simulations of scramjet combustor flow fields. AIAA Paper 2012-4262, 2012.
16. J. A. Fulton, J. R. Edwards, H. A. Hassan, J. C. McDaniel, C. P. Goyne, R. D. Rockwell, A. D. Cutler, C. T. Johansen, and P. M. Danehy. Large-eddy/reynolds-averaged navierstokes simulations of reactive flow in dual-mode scramjet combustor. *Journal of Propulsion and Power*, 30(3):558–575, 2013.
17. Z. Ghiasi, J. Komperda, D. Li, and F. Mashayek. Simulation of supersonic turbulent non-reactive flow in ramp-cavity combustor using a discontinuous spectral element method. *AIAA Paper 2016-0617*, 2016.
18. A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, May 2009.
19. A. HADJADJ and A. KUDRYAVTSEV. Computation and flow visualization in high-speed aerodynamics. *Journal of Turbulence*, 6:16, 2005.
20. D. O. Hebb. *The Organization of Behavior*. John Wiley, New York, 1949.
21. G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
22. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
23. A. Irannejad, F. Jaberi, J. Komperda, and F. Mashayek. Large eddy simulation of supersonic turbulent combustion with fmdf. *AIAA Paper 2014-1188*, 2014.
24. G. Ji, H.-W. Shen, and R. Wenger. Volume tracking using higpher dimensional isosurfacing. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 28–, Washington, DC, USA, 2003. IEEE Computer Society.
25. O. JJ. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *Archives of General Psychiatry*, 7(3):218–219, 1962.
26. Y. JU, A. SHIMANO, and O. INOUE. Vorticity generation and flame distortion induced by shock flame interaction. In *Proceedings of the Combustion Institute 27*, pages 735–741. The Combustion Institute, Boulder, CO, 1998.
27. A. M. Khokhlov, E. S. Oran, and G. O. Thomas. Numerical simulation of deflagration-to-detonation transition: The role of shockflame interactions in turbulent flames. *Combustion and Flame*, 117(12):323 – 339, 1999.
28. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.
29. J. Larsson, R. Vicquelin, and I. Bermejo-Moreno. Large eddy simulations of the hyshot ii scramjet. Ann. res. briefs, Center for Turbulence Research, Stanford University, Stanford, CA, 2011.
30. Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–258. MIT Press, Cambridge, MA, USA, 1998.
31. V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. *ACM SIGARCH Computer Architecture News*, 38(3):451–460, 2010.
32. Z. Li, A. Banaeizadeh, S. Rezaeiravesh, and F. Jaberi. Advanced modeling of high speed turbulent reacting flows. *AIAA Paper 2012-116*, 2012.
33. D. Lovely and H. Haimes. Shock detection from computational fluid dynamics results. In *14th Computational Fluid Dynamics Conference*, pages 255–258, Cambridge, MA, USA, 1999. MIT Press.
34. K.-L. Ma, J. V. Rosendale, and W. Vermeer. 3d shock wave visualization on unstructured grids. In *VVS*, pages 87–, 1996.
35. G. E. Marai, T. Luciani, A. Maries, S. L. Yilmaz, and M. B. Nik. Visual descriptors for dense tensor fields in computational turbulent combustion: A case study. In *Visualization and Data Analysis*, pages 1–11. Ingenta, 2016.

36. A. Maries, T. Luciani, P. Pisciuneri, M. Nik, S. Yilmaz, P. Givi, and G. Marai. *A Clustering Method for Identifying Regions of Interest in Turbulent Combustion Tensor Fields*. Visualization and Processing of Higher Order Descriptors for Multi-Valued Data. Editors: Ingrid Hotz and Thomas Schultz, Springer, 2015. In Press.
37. M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.
38. D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.
39. W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
40. F. Milletari, N. Navab, and S. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, abs/1606.04797, 2016.
41. M. Minsky and S. Papert. Perceptrons : an introduction to computational geometry, 1969.
42. H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.
43. S. Ozer, J. Wei, D. Silver, and P. Martin. Group dynamics in scientific visualization. in large data analysis and visualization (ldav). In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, page 97104, Oct. 2012.
44. H.-G. Pagendarm and B. Walter. Feature detection from vector quantities in a numerically simulated hypersonic flow field in combination with experimental flow visualization. In *Proceedings of the Conference on Visualization '94*, VIS '94, pages 117–123, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
45. R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.
46. D. M. Peterson, M. Hagenmaier, C. D. Carter, and S. G. Tuttle. Hybrid reynolds-averaged and large-eddy simulations of a supersonic cavity flameholder. AIAA Paper 2013-2483, 2013.
47. C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. *CoRR*, abs/1604.03265, 2016.
48. C. J. Roy and J. R. Edwards. Numerical simulation of a three-dimensional flame/shock wave interaction. *AIAA Journal*, 38(5):745–754, 2000.
49. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.
50. R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.
51. D. Silver. Object-oriented visualization. *IEEE Computer Graphics and Applications*, 15(3):54–62, 1.
52. D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997.
53. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.
54. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.
55. S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu. A taxonomy of deep convolutional neural nets for computer vision. *Frontiers in Robotics and AI*, 2:36, 2016.
56. J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014.
57. E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer-Verlag, Berlin, 3rd edition, 2009.
58. F.-Y. Tzeng and K.-L. Ma. Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 6–, Washington, DC, USA, 2005. IEEE Computer Society.
59. M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.