NETWORK COMPRESSION FOR MACHINE-LEARNT FLUID SIMULATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-scale, multi-fidelity numerical simulations form the pillar of scientific applications related to numerically modeling fluids. However, simulating the fluid behavior characterized by the non-linear Navier Stokes equations are often times computational expensive. Physics informed machine learning methods is a viable alternative and as such has seen great interest in the community [refer to Kutz (2017); Brunton et al. (2020); Duraisamy et al. (2019) for a detailed review on this topic]. For full physics emulators, the cost of network inference is often trivial. However, in the current paradigm of data-driven fluid mechanics models are built as surrogates for complex sub-processes. These models are then used in conjunction to the Navier Stokes solvers, which makes ML model inference an important factor in the terms of algorithmic latency. With the ever growing size of networks, and often times overparameterization, exploring effective network compression techniques becomes not only relevant but critical for engineering systems design. In this study, we explore the applicability of pruning and quantization (FP32 to int8) methods for one such application relevant to modeling fluid turbulence. Post-compression, we demonstrate the improvement in the accuracy of network predictions and build intuition in the process by comparing the compressed to the original network state.

1 INTRODUCTION

Overparameterized neural networks have recorded state-of-the-art performances in applications such as computer vision and natural language processing [Neill (2020); Huang et al. (2018); Brown et al. (2020)] in recent years. These networks however, lead to a larger footprint for inference as well as huge memory and storage requirements. While there exists techniques to improve model inference by network compression, network pruning and quantization are emerging as important techniques that are algorithm and hardware-efficient [Tanaka et al. (2019); Arora et al. (2018)]. Such network compression techniques can substantially reduce the computational demands of the inference when conducted in a fashion amenable to the hardware or hardware designed to explore sparsity [Rocki et al. (2020)]. While there have been studies that explore pruning at network initialization [Blalock et al. (2020); Frankle et al. (2020)] to reduce cost of training itself, the scope of this work is limited to exploring feasibility of pruning methods for scientific applications, post-training.

While magnitude pruning [Mozer & Smolensky (1988); Hassibi & Stork (1993); Janowsky (1989); Reed (1993); Han et al. (2015)] has shown impressive results, they involve significantly more computational cost to iteratively train and prune network parameters [Tanaka et al. (2019)]. In addition, these methods in some cases undergo the phenomena of *layer collapse*, rendering the network untrainable. Therefore a more appropriate approach in pruning is based on conserving gradient based scores at each neuron and layer of the network [Tanaka et al. (2019); LeCun et al. (1989)]. In this study, the Synaptic Flow (SynFlow) pruning algorithm is used. Furthermore, the effect of quantization by storing information in fewer bits (FP32 to int8 precision) are explored. The compressed network performance is measured against a hold-out test dataset, taken from the same distribution as the training set. For scientific surrogate model applications, learning OOD performance is not critical. This is especially relevant to building models for complex physical processes that provide source terms to the larger numerical model as in the case of weather modeling [Watt-Meyer et al. (2021)] and combustion [Bode et al. (2021)].

1.1 MACHINE LEARNING FOR FLUIDS

Scale bridging is a critical need in computational sciences, where the modeling community has developed accurate physics models from first principles, of processes at lower length and time scales that influence the behavior at the high scales of interest. However, it is not computationally feasible to incorporate all of the lower length scale physics directly into up-scaled models. This is an area where machine learning has shown promise, in building emulators of the lower length scale models which incur a mere fraction of the computational cost of the original higher fidelity models. Some related efforts can be found in Portwood et al. (2019; 2020); Raissi et al. (2020); Ling et al. (2016); Maulik et al. (2019); Shankar et al. (2020). We extend this effort to emulate a data-driven turbulence sub-grid closure term for compressible flows relevant to modeling advanced propulsion systems - refer to Appendix C for more details.

Most neural networks have inductive biases and Mitchell (1980) argued that inductive biases constitute the heart of generalization and the key basis for learning itself [Finlayson (2020)]. A key challenge of machine learning, therefore, is to design systems whose inductive biases align with the structure of the problem at hand. The effect of such efforts is not merely to endow the model with the capacity to learn key patterns, but also – somewhat paradoxically – to deliberately hamper the capacity of the model to learn other (presumably less useful) patterns, or at least to drive the model away from learning them. In other words, inductive biases stipulate the properties that we believe our model should have in order to generalize to future data; they thus encode our key assumptions about the problem itself. Unlike applications in computer vision where models are deployed on edge devices (ex. FPGAs), our choice of network is not limited to reducing number of parameters due to memory limitations. It is however dependent on reducing number of total operations - more details in Figure 3. Therefore our chosen network architecture is an eight-hidden layer feed-forward dense neural network that has similar inductive biases of our problem of interest – i.e. to determine functional relationship between parameters to predict a series of outputs – and lower number of operations compared to a spatial relationship aware Convolutional Neural Network (CNN).

In this study, we explore a previously well trained network developed expressly for this task. In order to automate the process of network design, a Bayesian optimization based autoML method is used - more details can be found in previous work [Mitra et al. (2020)]. Once the best performing setting for network architecture and hyper-parameters are identified, they are coupled to an open source PDE solver, OpenFOAM [Jasak et al. (2007)]. While Appendix A and C describe the approach briefly, for a detailed review refer to the original manuscript [c.f. Mitra et al. (2021)].

1.2 THE NEED FOR NETWORK COMPRESSION

The machine-learnt subgrid model, now coupled to the C++ solver, OpenFOAM [Jasak et al. (2007)], is used for inference - in a workflow schematic as shown in Appendix A. This inference is obtained at each node point, for each time-step. Typically for a engineering level fluid simulation, the number of meshpoints and timesteps are in the order of a few million. This essentially means the number of network inference function calls are of the same order. For a deep neural network that amounts to $O * T * N$ operation calls, where O is the number of matrix operations, T is the number of overall simulation time-steps and N are the total number of node points. Even if the well-trained surrogate model alleviates the need to numerically solve for PDEs for specific tasks, these repetitive costs accumulate and become expensive for reasonably sized problem. This also leads to a latency between different components of the numerical solver that incorporates the inference calls, which slows down overall computation.

A useful strategy in alleviating this would be to use a reduced precision, that will improve the arithmetic and memory bandwidths for inference. For example, half-precision math throughput in recent GPUs is 2× to 8× higher than for single-precision [Al Ghadani et al. (2020)]. In addition to speed improvements, reduced precision formats also reduce the amount of memory required for training and inference. Therefore, pruning and quantization can play an important role in network compression and improving the inference throughput.

2 NETWORK COMPRESSION

2.1 PRUNING APPROACH

Conventional pruning algorithms assign scores to parameters in neural networks after training and remove the parameters with the lowest scores. Popular scoring metrics include weight magnitudes, its generalization to multi-layers, first- and second-order Taylor coefficients of the training loss with respect to the parameters, and more sophisticated variants. While these pruning algorithms can indeed compress neural networks at test time, there is no reduction in the cost of training. On the other hand the SynFlow approach [Tanaka et al. (2019)] uses synaptic saliency as a metric for pruning defined as:

$$S(\theta) = \frac{\partial \mathrm{R}}{\partial \theta} \odot \theta; S^{in} = \frac{\partial \mathrm{R}}{\partial \theta^{in}} \odot \theta^{in}; S^{out} = \frac{\partial \mathrm{R}}{\partial \theta^{out}} \odot \theta^{out} \tag{1}$$

where R is a scalar loss function of the output y of a feed-forward network parameterized by θ. When R is the training loss L, the resulting synaptic saliency metric is equivalent (modulo sign) to - $\frac{\partial \mathrm{L}}{\partial \theta} \odot \theta$, the score metric used in skeletonization LeCun et al. (1989), one of the first network pruning algorithms. The resulting score metric is also closely related to $|\frac{\partial \mathrm{R}}{\partial \theta} \odot \theta|$. In the SynFlow approach, Tanaka et al. (2019), the neuron-wise conservation of synaptic saliency is maintained such $S^{in} = S^{out}$.

2.2 QUANTIZATION

Post-training quantization is an important tool for optimizing deep learning models, as it helps accelerate inference. To accomplish this goal of int8 quantization, the Deep Learning Quantizer app within MATLAB (2020) is used. The important aspects of quantization is the precision loss due underflow and the overflow of the stored information. Appendix B discusses this briefly.

3 RESULTS

We use two thresholds to compare the performance of the pruning algorithm. First is a threshold of pruning 50% parameters and the second a threshold of pruning 90% parameters. These thresholds although somewhat arbitrary, will demonstrate the possible compression of the network and the effect on the inference performance. The pruning is done on all the network layers, fc_1 to fc_9 in an iterative fashion. The resulting outcome from each iteration is described in detail in Appendix B while the final state is discussed in Figure 1. Tables 1 and 2 shows the relative number of parameters pruned in each setting and the corresponding Mean Squared Error (MSE) measure on the unseen validation dataset, randomly sampled from the original dataset. Additionally, the effects of quantization were explored and reported in Table 2. It follows that for both the pruning thresholds, the network performance (MSE) improves over the baseline, determined by the original network MSE on the same data. This outcome is not entirely surprising, as overparameterization and the propensity of deep neural networks to learn noise for regularization, and better generalization can lead to lower baseline scores [Bao et al. (2019)]. Pruning can often remove the learnt 'noise' which improves performance at the cost of robustness and generalizability. Figure 2 shows the histogram of the original network and the min/max range of the data is within the overflow and underflow range of the int8 precision (-128 to 127) which leads to insignificant precision losses.

For scientific surrogate models, while the network inference is key in accelerating compute, the most important metric is the accuracy of the model output. This gains relevance as the ML models are coupled to non-linear PDE solvers which makes the overall system extremely sensitive to the accuracy of these data-driven sub-models. The 50% pruned network shows remarkable consistency with the original network characteristics (see Figure 4) with the added advantage of an atleast 5x speedup [refer Table 4]. The loss in the saliency score is about an order of magnitude for the 50% threshold, compared to three orders of magnitude for the 90% pruning (refer Table 3). This makes the former a stronger candidate for pruning thresholds for the given problem.

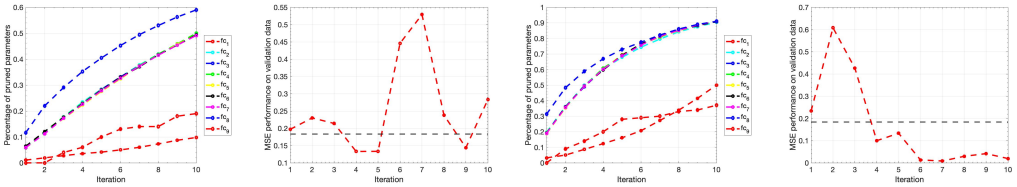

Figure 1: [L-R] As the pruning iteration increases, the larger share of parameters are pruned from deeper layers preventing layer collapse. Pruning 50% parameters from the network shows a slight improvement over original MSE (marked in dotted horizontal line) of the full network. Pruning 90% parameters show significant improvement over original MSE baseline. This can be attributed to the removal of the noise in the learning process. The gain in speedup is compensated by the loss in generalizability for these models. However, for many surrogate modeling applications this may be acceptable.

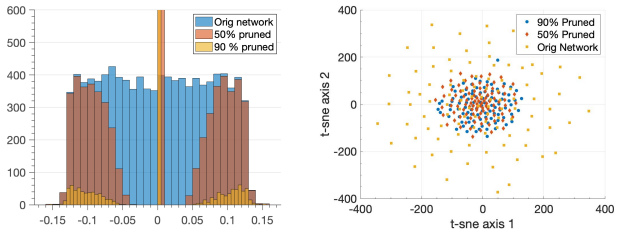

Figure 2: Comparison of the layer connections for fc_4 between the original and pruned networks shows the pruning is done primarily for very low values of the weights, preserving the representational value of the network and removing 'noise'. The low-dimensional representation, obtained from t-SNE, shows the overlapping function-space similarities in the original to the compressed networks, which explains the generally good inference performance for latter with fewer parameters.

3.1 Exploring similarities within low rank representation

To better understand the predictions, we plot the histogram of the original network, and the two pruned networks for the layer 4 connections (fc_4), in Figure 2, as it is a representative sample of the neural connections in the whole network. The result shows that the pruned networks only retain the most consequential information, i.e. at the edges of the histogram, whereas the noise in the form of small weight values are not retained. This supports the discussion in the previous section. Visualizing the distribution of network weights in a low-dimensional space for the same layer, using t-SNE [Van der Maaten & Hinton (2008)], shows a similar trend of overlapping regions of similarity, whereas the original network also retains some sparse noise. Overall, for the cost of pruning, and the efficiency gained the 50% pruned network overall appears a good candidate for the network compression.

4 Conclusions

Network pruning, or the compression of neural networks by removing parameters, has been an important subject both for reasons of practical deployment and for theoretical understanding of artificial [Arora et al. (2018)] and biological [Liu et al. (2018)] neural networks. Pruning shows an immediate effect on the network predictions by improving the performance of the network on the original dataset in Figure 1. This is not a surprising result as previous studies [Tanaka et al. (2019); Arora et al. (2018)] have reported minor improvements in accuracy during post-training pruning exercises. These pruned models are expected to have comparatively poorer generalizability characteristics. However, for applications related to building surrogate models the networks are expected to have good performance for datasets from within the training distribution, therefore sacrificing generalizability to achieve faster network inference might be relevant to many applications of scientific interest relevant to climate/weather modeling and building robust digital twins, among others.

REFERENCES

Ahmed Khamis Abdullah Al Ghadani, Waleeja Mateen, and Rameshkumar G Ramaswamy. Tensor-based cuda optimization for ann inferencing using parallel acceleration on embedded gpu. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 291–302. Springer, 2020.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pp. 254–263. PMLR, 2018.

Zhenshan Bao, Jiayang Liu, and Wenbo Zhang. Using distillation to improve network performance after pruning and quantization. In *Proceedings of the 2019 2nd International Conference on Machine Learning and Machine Intelligence*, pp. 3–6, 2019.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

Mathis Bode, Michael Gauding, Zeyu Lian, Dominik Denker, Marco Davidovic, Konstantin Kleinheinz, Jenia Jitsev, and Heinz Pitsch. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. *Proceedings of the Combustion Institute*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.

Sam Finlayson. Induction, Inductive Biases, and Infusing Knowledge into Learned Representations. `https://sgfin.github.io/2020/06/22/Induction-Intro/#InduBias`, 2020. [Online; accessed 19-February-2020].

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.

Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pp. 1–20. IUC Dubrovnik Croatia, 2007.

J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *NIPs*, volume 2, pp. 598–605. Citeseer, 1989.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.

W Malalasekera and HK Versteeg. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Prentice Hall Upper Saddle River, NJ, 2007.

MATLAB. *version 9.90.0 (R2020a)*. The MathWorks Inc., Natick, Massachusetts, 2020.

Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.

M Meldi, Didier Lucor, and P Sagaut. Is the smagorinsky coefficient sensitive to uncertainty in the form of the energy spectrum? *Physics of Fluids*, 23(12):125109, 2011.

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

Peetak Mitra, Niccolò Dal Santo, Majid Haghshenas, Shounak Mitra, Conor Daly, and David P Schmidt. On the effectiveness of bayesian automl methods for physics emulators. 2020.

Peetak Mitra, Majid Haghshenas, Niccolò Dal Santo, Mateus Dias Ribeiro, Shounak Mitra, Conor Daly, and David Schmidt. Analysis and interpretation of data-driven closure models for large eddy simulation of internal combustion engines. Technical report, SAE Technical Paper, 2021.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, pp. 107–115, 1988.

James O' Neill. An overview of neural network compression. *arXiv preprint arXiv:2006.03669*, 2020.

Gavin D Portwood, Peetak P Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T Nadiga, Juan A Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, et al. Turbulence forecasting via neural ode. *arXiv preprint arXiv:1911.05180*, 2019.

Gavin D Portwood, Balasubramanya T Nadiga, Juan A Saenz, and Daniel Livescu. Analysis and interpretation of out-performing neural network residual flux models. *arXiv preprint arXiv:2004.07207*, 2020.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

Mateus Dias Ribeiro, Alex Mendonca Bimbato, Mauricio Araujo Zanardi, Jose Antonio Perrella Balestieri, and David P Schmidt. Large-eddy simulation of the flow in a direct injection spark ignition engine using an open-source framework. *INTERNATIONAL JOURNAL OF ENGINE RESEARCH*, 2020.

Kamil Rocki, Dirk Van Essendelft, Ilya Sharapov, Robert Schreiber, Michael Morrison, Vladimir Kibardin, Andrey Portnoy, Jean Francois Dietiker, Madhava Syamlal, and Michael James. Fast stencil-code computation on a wafer-scale processor. *arXiv preprint arXiv:2010.03660*, 2020.

Varun Shankar, Gavin Portwood, Arvind Mohan, Peetak Mitra, Venkat Viswanathan, and David Schmidt. Rapid spatiotemporal turbulence modeling with convolutional neural odes. *Bulletin of the American Physical Society*, 2020.

Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

Charles G Speziale. Galilean invariance of subgrid-scale stress models in the large-eddy simulation of turbulence. *Journal of fluid mechanics*, 156:55–62, 1985.

Edward A Spiegel and G Veronis. On the boussinesq approximation for a compressible fluid. *The Astrophysical Journal*, 131:442, 1960.

Hidenori Tanaka, Aran Nayebi, Niru Maheswaranathan, Lane McIntosh, Stephen A Baccus, and Surya Ganguli. From deep learning to mechanistic understanding in neuroscience: the structure of retinal prediction. *arXiv preprint arXiv:1912.06207*, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Oliver Watt-Meyer, Noah Domino Brenowitz, Spencer Koncius Clark, Brian Henn, Anna Kwa, Jeremy J McGibbon, Walter A Perkins, and Christopher S Bretherton. Correcting weather and climate models by machine learning nudged historical simulations. 2021.

A Overall approach

Fluid turbulence is a multi-scale phenomenon and is an essential component of modeling engineering-relevant flows. While solving the full Navier Stokes using Direct Numerical Simulation (DNS) results in the most accurate representation of the complicated, non-linear, non-local, multi-scale phenomenon, DNS is often computationally intractable. Engineering level solutions based on Reynolds Averaged Navier-Stokes (RANS) and Large Eddy Simulations (LES) alleviate this issue by resolving the larger integral length scales and modelling the smaller unresolved scales by introducing a linear operator to the Navier-Stokes equation to reduce the simulation complexity. These models however suffer from the curse of turbulence closure. The linear eddy-viscosity model represents one of the most popular methods for Reynolds stress closure for two-equation RANS as well as Smagronisky-LES models Germano et al. (1991); Smagorinsky (1963). However, these approximates models are commonly phenomenological/heuristic in nature and thus require fitting to high fidelity DNS datasets for idealized flows in anycase Portwood et al. (2019).

In the original work, a data-driven turbulence closure model is built. The full details can be found in the original work [Mitra et al. (2021)]. Once fully trained the network is then converted to C++ source code using MATLAB [MATLAB (2020)] Code generator tools, which is then fully integrated into the OpenFOAM Navier-Stokes solver. During runtime for the coupled framework [see Figure 3], the network inference function along with various libraries in OpenFOAM are called at each timestep and at each node. Therefore the latency in the network inference function call, potentially slows down the overall simulation. This paradigm is similar to applications in weather forecasting as well as combustion [Bode et al. (2021)].

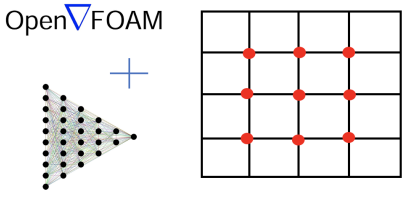

Figure 3: The current modeling paradigm includes using the neural network as a surrogate model within the non-linear PDE solver. The neural network is used as an inference engine within the OpenFOAM Jasak et al. (2007) code and the inferences scales with the nodes (indicated here as the red dot on a simplified mesh structure) at every iteration of the simulation. A typical CFD problem involves millions of mesh points and thousands of timesteps before a solution is reached, thereby exacerbating the challenge in slow network inference.

B NETWORK COMPRESSION, ADDITIONAL RESULTS

The final state for the layer wise pruning, in Table 1, indicates the shallower layers lose fewer parameters while the deeper layers lose more parameters as a percentage of the total per-layer connections. This is consistent with the SynFlow algorithm and essentially ensures there is no layer collapse in the network.

Layer	**Total parameters**	**50% pruned network**	**90% pruned network**
fc_1	1400	137	700
fc_2	10000	4949	9045
fc_3	10000	4956	9099
fc_4	10000	5000	9082
fc_5	10000	4938	9109
fc_6	10000	4938	9074
fc_7	10000	4912	9108
fc_8	10000	5901	9096
fc_9	100	19	37
MSE	0.1874	0.133	0.029

Table 1: The total number of pruned parameters for each threshold (50% and 90%) at the end of the final iteration. The MSE performance improves over the baseline (original network).

B.1 COMPUTATIONAL FOOTPRINT OF THE COMPRESSED NETWORKS

Apart from the discussions regarding the compressed network inference performance in terms of validation dataset accuracy and the speed up, Table 2 discusses the footprint of the original network compared to the compressed networks in terms of space occupied by the weights and the run time memory. As expected, with a higher compression ratio, the memory occupied by weights and the runtime memory requirements reduce.

Network	**# Weights**	**Memory (Weights)**	**# Operations**	**Runtime memory**	**MSE***
Original	71500	0.27 MB	142199	1.63 MB	0.1840
Unpruned Quantized	71500	0.14 MB	142199	0.55 MB	**0.1680**
50% Pruned	35750	0.14 MB	70699	0.8 MB	0.2694
50 % Pruned Quantized	35750	0.07 MB	70699	0.27 MB	**0.1053**
90% Pruned	14300	0.05 MB	27799	0.32 MB	**0.0339**
90 % Pruned Quantized	35750	0.03 MB	70699	0.11 MB	**0.0518**

Table 2: The effect in terms of runtime memory and space occupied by network weights is inversely proportional to the compression ratio. The MSE is tested on an unseen validation dataset containing 10,000 samples. The **bolded** networks indicate better than baseline performing settings.

B.2 CONSERVATION OF SALIENCY SCORE

The conservation of the saliency score (refer equation 1) is shown in Table 3 for both pruning thresholds. The 50% pruned network shows a loss of an order of magnitude compared to three for the higher compression ratio setting. This is consistent to the singular value observation seen in Figure 4.

Network/Epoch	1	2	3	4	5	6	7	8	9	10
50% pruned	$1.09e^8$	$1.06e^8$	$9.92e^7$	$8.80e^7$	$7.52e^7$	$6.20e^7$	$4.98e^7$	$3.90e^7$	$3.00e^7$	$2.27e^7$
90% pruned	$1.09e^8$	$8.40e^7$	$4.26e^7$	$1.69e^7$	$6.09e^6$	$2.46e^6$	$1.13e^6$	$5.20e^5$	$2.45e^5$	$1.10e^5$

Table 3: The saliency score for both pruned networks show that the 50% pruning conserves much of the original information, consistent with observations in Figure 4.

Network	Sample size: 1000	Sample size: 1000000	Speedup	Overall speedup
Original	0.19 s	1.5 s	1x	3-4x
50 % pruned	0.14 s	1.1 s	1.5x	4.5-6 x
90 % pruned	0.06 s	0.47 s	3x	9-12x

Table 4: The inference speed-up post pruning for different data sizes shows up to 5-10x overall improvement in network inference without appreciable loss in generalizability. The results from different samples are shown here to demonstrate consistency in inference throughput.

B.3 INFERENCE SPEEDUP

Among the primary motivation to explore the feasibility of network compression is to improve network inference. Table 4 shows the performance of the inference engine in a simulation-relevant environment. Since the quantization involves storing information in lower bits, from FP32 to int8 precision, an additional 3-4x speedup is expected for the networks.

C PHYSICS OF THE PROBLEM

The LES-filtered governing equations (using Favre-averaging) for the balance of mass and momentum are as below:

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u_j}) = 0 \tag{2}$$

$$\frac{\partial(\bar{\rho}\tilde{u_i})}{\partial t} + \frac{\partial \bar{\rho}\tilde{u_i}\tilde{u_j}}{\partial x_j} = \frac{\partial}{\partial x_j}[\bar{\rho}\bar{\nu}(\frac{\partial \tilde{u_j}}{\partial x_i} + \frac{\partial \tilde{u_i}}{\partial x_j}) - \frac{2}{3}\bar{\rho}\bar{\nu}\frac{\partial \tilde{u_k}}{\partial x_k}\delta_{ij} - \rho\tau_{ij}^{sgs}] - \frac{\partial \bar{p}}{\partial x_i} + \bar{p}g_i \tag{3}$$

where u represents the velocity, p is the pressure, ρ the fluid density, ν the dynamic the viscosity and τ the subgrid stress term. The effect of the sub-grid scale appears on the right hand side of the governing equations through the sub-grid scale stresses, τ_{ij}, which are modelled using the Boussinesq approximation Spiegel & Veronis (1960), and the assumption by Smagorinsky that the smallest scales are isotropic Smagorinsky (1963). Based on Prandtl mixing length theory, the subgrid viscosity can be derived in terms of characteristic length and one velocity scale Malalasekera & Versteeg (2007) as follows, therefore helping to close the Reynolds stress term

$$\tau_{ij}{}^{sgs} - \frac{1}{3}\tau_{kk}{}^{sgs}\delta_{ij} = -\mu_{sgs}S_{ij} \tag{4}$$

$$\mu_{sgs} = \rho(C_s\Delta)^2|\bar{S}| \tag{5}$$

where the superscript sgs stands for subgrid scale terms, Δ is the filter width, and C_s is the Smagorinsky constant, and S_{ij} is the strain rate tensor and is calculated by taking off-diagnal gradients of velocities. In conclusion, the above approximation for the eddy viscosity assumes that changes in the resolved fields are slow, so that subgrid eddies can adjust themselves quickly to the rate-of-strain tensor. Thus, a closure based on a single constant is not universally true and the constant value may have to be adjusted Meldi et al. (2011), based on fitting the model parameters to high-fidelity data. Since some form of data-fitting is needed to optimize the subgrid scale model parameters even for this simplified approach, one can envisage a purely data-driven method to optimally approximate this changing constant based on large scale resolved terms, motivating our approach. More details about the LES implementation as well as the accuracy of results is reported in Ribeiro et al. (2020).

In this work we aim to derive a functional relationship between the large scale resolved flow features and the sub-grid scale unresolved terms, and specifically to approximate the subgrid scale viscosity

$$\mu_{sgs} = f(Re_c, S, \Omega, K, Y) \tag{6}$$

where Re_c is the Cell Reynolds number, S is the Strain-rate tensor and has six components, Ω is the rotation-rate tensor, and has three components, K is the Kinetic energy gradient, and Y is

a non-dimensional term that is a measure of the mesh resolution. The non-dimensionalized input features are chosen in order to impose Galliean-invariance Speziale (1985); Ling et al. (2016). This functional mapping between a set of inputs to an output corresponds to the inductive biases of the dense fully connected network, hence the choice of the said architecture.

D ADDITIONAL ANALYSIS

D.1 STRUCTURAL SIMILARITY

Structural Similarity Index Measure (SSIM) is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms Wang et al. (2004). Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the data in the matrix. This metric applied to two similar-sized matrix, x and y, can be defined mathematically as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7}$$

where μ_x is the average of x_i, μ_y is the average of y_i, c_1 and c_2 are some constants, σ_{xy} is the covariance of x and y, σ_x^2 and σ_y^2 are the variances of x and y.

D.2 SINGULAR VALUE DECOMPOSITION

In addition to the SSIM, another measure to compare matrices is using Singular Value Decomposition (SVD). A primary application of this in the current scenario is to compare the singular values, in decreasing order, to measure the so called 'energies' of the matrix. Mathematically for a matrix $\mathbf{M}$, this can be defined as:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{8}$$

where $\mathbf{U}$ is a $m \times m$ complex unitary matrix, $\mathbf{\Sigma}$ is a $m \times n$ rectangular diagonal matrix with non-negative numbers on the diagonal (also known as singular values), and $\mathbf{V}$ is a $n \times n$ complex unitary matrix. The singular values $\mathbf{\Sigma 1} \geq \mathbf{\Sigma 2} \geq ... \geq \mathbf{\Sigma m} \geq \mathbf{0}$ are in descending order along the main diagonal of $\mathbf{\Sigma}$. The most important information or the highest 'energetic modes' are stored in the first few columns.

In the world of image processing, singular values have been used to compress the information in fewer bits using the first few singular values. In comparing the singular values for layer 4 connections in the different networks, the close similarities in the original network and the 50% pruned network is encouraging. This essential means that the 50% network retains most of the relevant information from the original network as evidenced in Figure 2. This in conjunction with a high degree of conservation of the saliency score (Table 3) shows that the 50% pruning is most appropriate for network compression.

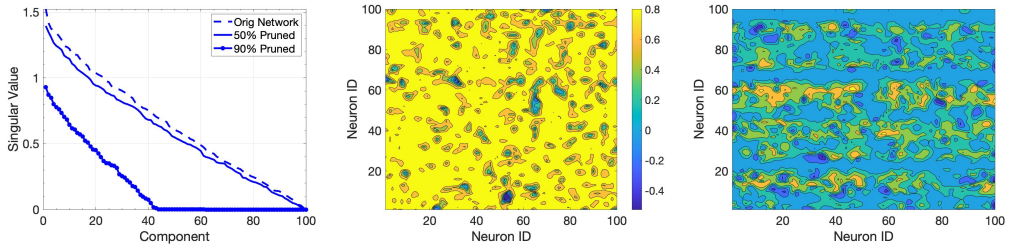

Figure 4: The similarity in the information retained as shown from the singular value plot on the left and the structural similarity between original-50% pruned networks and original-90% pruned networks reveal important characteristics. The 50% pruning effectively retains a lot of the information in the origina network, in a compressed space.