

The 1st International Workshop “Feature Extraction: Modern Questions and Challenges”

Hierarchical Feature Extraction for Efficient Design of Microfluidic Flow Patterns

Kin Gwn Lore

Daniel Stoecklein

Michael Davies

Baskar Ganapathysubramanian

Soumik Sarkar

Department of Mechanical Engineering

Iowa State University

Ames, IA 50010, USA

KGLORE@IASTATE.EDU

STOECKD@IASTATE.EDU

MDAVIES@IASTATE.EDU

BASKARG@IASTATE.EDU

SOUMIKS@IASTATE.EDU

Editor: Afshin Rostamizadeh

Abstract

Deep neural networks are being widely used for feature representation learning in diverse problem areas ranging from object recognition and speech recognition to robotic perception and human disease prediction. We demonstrate a novel, perhaps the first application of deep learning in mechanical design, specifically to learn complex microfluidic flow patterns in order to solve inverse problems in fluid mechanics. A recent discovery showed the ability to control the fluid deformations in a microfluidic channel by placing a sequence of pillars. This provides a fundamental tool for numerous material science, manufacturing and biological applications. However, designing pillar sequences for user-defined deformations is practically infeasible as the current process requires laborious and time-consuming design iterations in a very large, highly nonlinear design space that can have as large as 10^{15} possibilities. We demonstrate that hierarchical feature extraction can potentially lead to a scalable design tool via learning semantic representations from a relatively small number of flow pattern examples. The paper compares the performances of pre-trained deep neural networks and deep convolutional neural networks as well as their learnt features. We show that a balanced training data generation process with respect to a metric on the output space improves the feature extraction performance. Overall, the deep learning based design process is shown to expedite the current state-of-the-art design approaches by more than 600 times.

1. Introduction

Learning multiple levels of abstraction (Deng and Dong 2014) using deep neural network architectures has experienced great success in various domains such as object recognition (Larochelle and Bengio 2008), speech recognition (Hinton et al. 2012), scene understanding (Sarkar et al. 2015; Couprie et al. 2013) and multi-modal sensor fusion (Srivastava and Salakhutdinov 2012). Aside from these traditional applications, deep learning has also made significant impact in biology by relating DNA variants to diseases (Leung et al. 2014) and detecting mitosis from cancer histology images (Cireşan et al. 2013). In this paper, the application of deep learning in design engineering (specifically, microfluidic device or lab-on-a-chip design) is explored, which has a large implication on manufacturing processes,

chemical engineering, and biology. The main goal of this paper is to investigate the potential of hierarchical feature extraction techniques to become scalable design tools via learning semantic representations from a relatively small number of training examples. Apart from this objective, we compare supervised and unsupervised schemes and explore the impact of training data generation technique on the feature extraction process.

Controlling the shape and location of a fluid stream provides a fundamental tool for creating structured materials, preparing biological samples, and engineering transport of heat and mass. Previous methods to manipulate the cross-sectional shape and flow deformation of fluids have focused on creating chaos and mixing by fluid twisting instead of ordering and structuring streams with precise sequences of fluid perturbations. The concept of sculpting fluid streams in a microchannel using a set of pillars that individually deform a flow has been recently discovered and demonstrated with great success Amini et al. (2013). Pre-computed deformations from individual pillars can be used in sequence as operations on fluid flow, enabling rapid simulation of arbitrary microfluidic devices. In this way, an elegant mathematical operation can yield the final flow shape for a sequence without an experiment or additional numerical simulation.

Although this approach has allowed for the sculpting of complex fluid shapes (Stoecklein et al. 2014), creating user-defined flow shapes for practical applications currently requires laborious and time-consuming trial and error design iterations. The ability to create a user-defined flow shape and automatically determine a sequence of pillars that yields this shape is a significant and impactful advance. While standard techniques that reformulate this inverse design problem into an unconstrained optimization problem (Stoecklein et al. 2015) have been successful, they are invariably time-consuming. This makes their utility in real-time design suspect.

With this motivation, the applicability of deep learning models is explored to serve as a map between user-defined flow shapes and the corresponding sequences of pillars. Previous works have used evolutionary algorithms and neural nets to solve forward computational fluid problems (Müller et al. 1999; Duriez et al. 2014; Baymani et al. 2010; Ashforth-Frost et al. 1995), but to the best of the authors' knowledge, this is the first application of deep learning based hierarchical feature extraction to solve the inverse problem specifically in fluid mechanics.

The major contributions of the paper are outlined below:

1. A novel 'hierarchical feature extraction for design' approach using deep neural networks (DNN) (Hinton 2009; Hinton et al. 2006) and convolutional neural networks (CNN) (Krizhevsky et al. 2012) is proposed to capture multi-scale patterns of a deformed fluid flow to solve a simultaneous multi-class classification problem. We show that such models can learn good representative features even from a very small subset of the entire design space (can be as skewed as $150k$ out of 10^{15} possibilities) leading to a scalable design tool.
2. Using domain knowledge regarding the massive nonlinear design space, a quasi-random sampling method is proposed based on a metric on the output space to generate robust training data for improved feature extraction.
3. The feasibility of the proposed approach is tested and validated via experiments with different network structures (supervised and unsupervised), with results showing com-

petitive prediction accuracy and significant reduction in execution time compared to the state-of-the-art genetic algorithm based design tool.

4. Hierarchical features are visualized to extract coherent structures in flow which may lead to better understanding of the physics of complex flow patterns.
5. The proposed method provides near real time design that greatly enables design exploration by the end user. As conventional state-of-the-art (based on genetic algorithm) take dozens of hours to identify one appropriate device design, multiple design iterations result in a design time of weeks for a microfluidic device. This wait time makes CAD like design/analysis of microfluidic devices infeasible and bottlenecks creative exploration especially for biomedical applications. We show that hierarchical feature extraction is able to address this issue.

Our collaborators intend to utilize this framework for concrete biomedical applications that require microfluidic device design, for example: (a) Device design to move fluid surrounding cells (lymphoid leukemia cells) against a far wall of the microfluidic channel where it can be collected at high purity while cells are maintained in the channel center. High purity collection can allow reuse of valuable reagents that are used to stain cells for diagnostics. (b) Device design to wrap fluid around the microchannel surface. This will be used to characterize binding of p24 (an HIV viral capsid protein) to anti-p24 antibody immobilized on the microchannel surface. The idea of flow sculpting here is to enhance reaction of low abundance proteins that can improve diagnostic limits of detection for various diseases.

Therefore, this novel study may open up many new application areas for the machine learning community related to thermo-fluid sciences and design engineering.

2. Problem setup

Deep neural networks (with pre-training using deep belief networks (Erhan et al. 2010a)) and deep convolutional neural networks are both used to extract pertinent nonlinear features from flow shape images as well as learn a classifier with pillar configurations as the target classes. Deep learning tools are ideal in this problem as configuring hand-crafted features for the flow shape images is an extremely nonintuitive and tedious process. As the forward problem of generating simulated flow shapes from a sequence is computationally inexpensive and can be done in a fraction of a second, a large set of labeled images (in the order of thousands/millions) can be generated for training. The class labels (in the order of hundreds) represent different pillar sequences determined by the number, size, and location of pillars. Training is performed using Theano platform (Bergstra et al. 2010) on NVIDIA TITAN Black GPU.

2.1 Simultaneous multi-class classifications (SMC)

One approach to solve the inverse design problem is to assign class indices to pillars with different specifications. For example, a pillar at position 0.0 and diameter 0.375 will be assigned an index of 1, whereas another pillar at position 0.125 and diameter 0.375 will be assigned an index of 2. Index assignment is performed over a finite combination of pillar positions and diameters that has been obtained by discretizing the design space. In this study, there are 32 possible indices that describes the diameter and position of a single pillar.

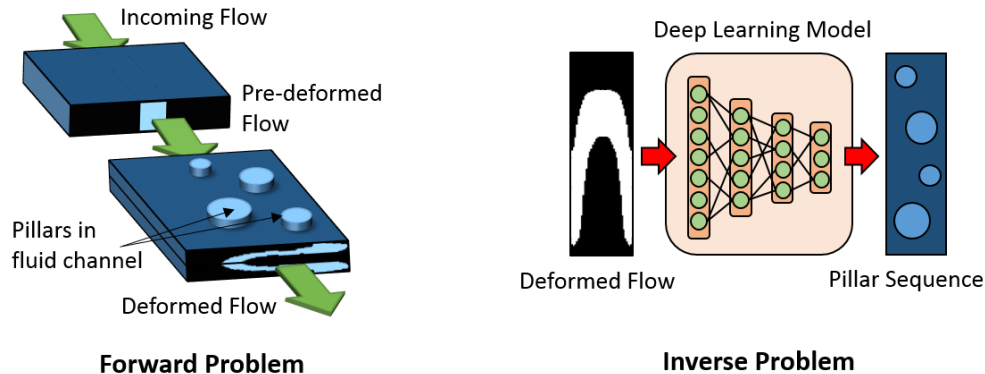


Figure 1: Illustration of the forward and the inverse problem

Consequently, this becomes a 32-class classification problem for one pillar. In general, when presented with a flow shape that is generated from an n_p -pillar sequence, the trained model needs to simultaneously predict the class for each of the n_p pillars to form a sequence of indices from which the positions and diameters for each pillar are retrieved. Essentially, instead of solving a single classification problem just like what that has been typically done, the model actually solves one problem for each pillar based on the same weights and biases that have been learned by the model.

Note, it may seem that assigning classes to target pillars suffers from under-specification. Actually, this was just a user-defined design constraint as there are limited configurations for individual pillar placements based on manufacturability.

This formulation is applicable on both DNN and CNN architectures (CNN with SMC shown in Figure 2) and requires only a slight modification in the loss function. For an n_p -pillar problem, the loss function to be minimized for a data set \mathcal{D} is the negative log-likelihood defined as:

$$\ell_{total}(\theta = \{W, b\}, \mathcal{D}) = - \sum_{j=1}^{n_p} \sum_{i=0}^{|\mathcal{D}|} \left[\log \left(P(Y = y^{(i)} | x^{(i)}, W, b) \right) \right]_j$$

where \mathcal{D} denotes the training set, θ is the model parameters with W as the weights and b for the biases. y is predicted pillar index whereas x is the provided flow shape test image. The total loss is obtained by summing the losses computer for each pillar.

2.2 Genetic algorithm

Previous works of Stoecklein et al. (2015) has attempted to identify pillar sequence(s) for a given fluid flow shape using the current state-of-the-art genetic algorithm (GA). The GA is an evolutionary search heuristic that solves optimization problems using techniques inspired from natural evolution. Broadly, a random initial population of pillar sequences is generated. This population is evolved through crossover and mutation toward a targeted flow shape by penalizing sequences that produce flow shapes different from the target, and rewarding those that are similar. These punishments and rewards are borne out in the breeding process between generations. After many generations of breeding, the population

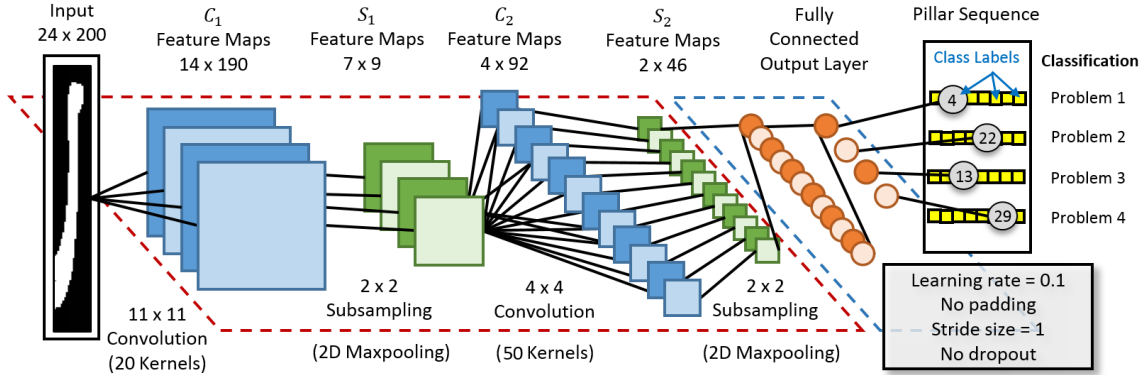


Figure 2: CNN with the SMC problem formulation.

should produce a larger subset of flow shapes similar to the target shape than the initial population.

3. Training data generation using a metric on the output space

As the number of pillars increases, number of design possibilities increases exponentially. Although training data generation process is fairly quick, the design process has to be efficient with a relatively much smaller number of training examples in order to be scalable. To investigate the scalability of the proposed technique, the number of training data examples is kept constant even as the number of pillars increases. In this study, each training dataset consists of 150,000 flow shape images generated from pillar sequences spanning 4-10 pillars. However, this means that each set of training data covers a vanishingly small fraction of the possible design space, with coverage shrinking exponentially as the number of pillars in a sequence increases (see Table 1 and Figure 8 (a)). Ideally, each dataset would uniformly sample the design space and capture a wide variety of features, but high dimensionality with longer pillar sequences makes this difficult for a uniform distribution. Randomly sampling would likely have clustering or gaps in coverage as the number of pillars in a sequence increases. We therefore use quasi-random Sobol sequences to more uniformly cover the high dimensional space (Sobol et al. 2011). Unlike typical psuedo-random number generators, a Sobol sequence enforces low discrepancy sampling. Discrepancy, D_N , is a measure of coverage for a sampled set P of size N within a set of intervals B spanning the available space, as defined by Kuipers (2006):

$$D_N(P) = \sup_B \left| \frac{A(B; P)}{N} - \lambda(B) \right| \quad (1)$$

where $A(B; P)$ is the number of sampled points that fall into the intervals defined by B , and $\lambda(B)$ is the Lebesgue measure of B (i.e., the size of the design space). By minimizing D_N (low discrepancy), the points sampled will fill the space defined by the intervals in B more evenly than uncorrelated random points (Press 1992). Thus, low discrepancy sampling helps to prevent crowding or large gaps in the selected pillar sequences, which should create a more balanced and varied dataset of pillar sequences as illustrated in Figure 3.

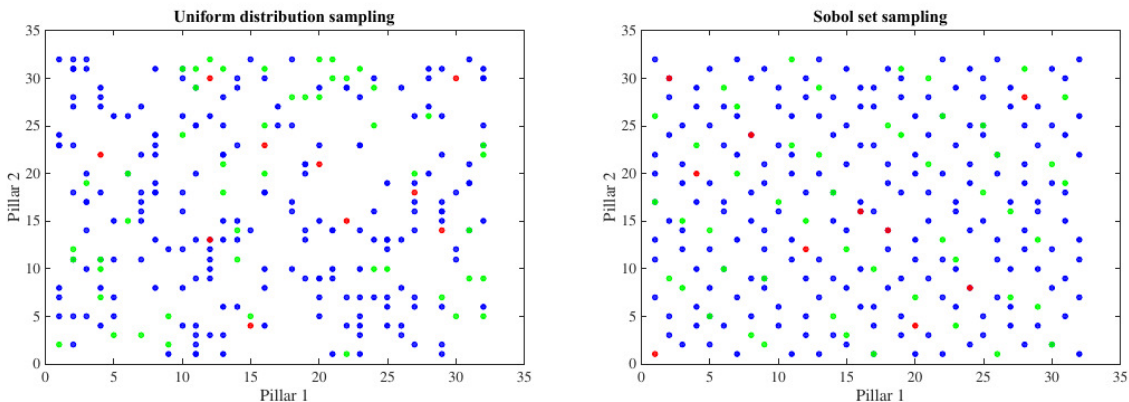


Figure 3: Uniform sampling vs. Quasi-random sampling. A set of 160 2-pillar sequences are formed using uniform (left) and Sobol (right) sampling methods. The first 10 sampled sequences are shown in red, the next 50 sampled sequences in green, and the final 100 sampled sequences in blue.

Table 1: Number of possible sequences, n_s in an n_p -pillar sequence.

n_p	4	5	6	7	8	9	10
n_s	32^4	32^5	32^6	32^7	32^8	32^9	32^{10}
n_s	$1.05(10^6)$	$3.36(10^7)$	$1.07(10^9)$	$3.43(10^{10})$	$1.10(10^{12})$	$3.52(10^{13})$	$1.12(10^{15})$

Sobol sequences create intervals by successively halving the size of the domain as the number of sampled points increase. Each new point in the sequence is therefore based on the previous point, which leads to the quasi-random nature of the Sobol set, as the sampling method is ultimately deterministic. The sampling used here chooses different pillar configurations from a set of 32 pillar types, with 4 diameters and 8 locations in the channel. Multiple pillars in a sequence means that the space being sampled is a hypercube of dimension n_p . Training, validation, and testing data are formed by disjoint Sobol sequences. The flow shape images are then formed using the same method as in Stoecklein et al. (2015), which reduces fluid flow simulation to simple matrix multiplication.

Remark 3.1 *In this training data generation process, we use the domain knowledge of the output space i.e., the space of the pillar sequence arrangements. With a simple Euclidean metric on the output space, the proposed quasi-random sampling process helps extract representative flow shape features from the massive design space. Such a method can have a broader implication for learning hierarchical features from physical datasets (both simulated and experimental) in general.*

4. Experiments and results

This section reports the parameters of the network and evaluates the performance of both DNN (with unsupervised pre-training using a deep belief network (DBN)) and CNN in solving the problem.

4.1 Network parameters and description

The DNN contains 5 hidden layers with 2,000 hidden units each. The input is an image of the flow shape with dimensions of 24×200 pixels flattened into a 1×4800 row vector with binary pixel intensity values of 0 (black) and 1 (white). The CNN consists of 2 convolutional layers (20 s of size 11×11 for first layer, 50 filters of size 4×4 for the second layer), 2 pooling layers (downsampled by 2×2 maxpooling), and one fully-connected layer with 500 hidden units. Training was performed on 150,000 samples (with an additional 20,000 validation samples) while testing was performed on 20,000 samples for each set of different sequence length. The learning rate is 0.1. All hyperparameters mentioned above have been cross-validated and chosen based on repeated experiments.

4.2 Performance of deep learning models

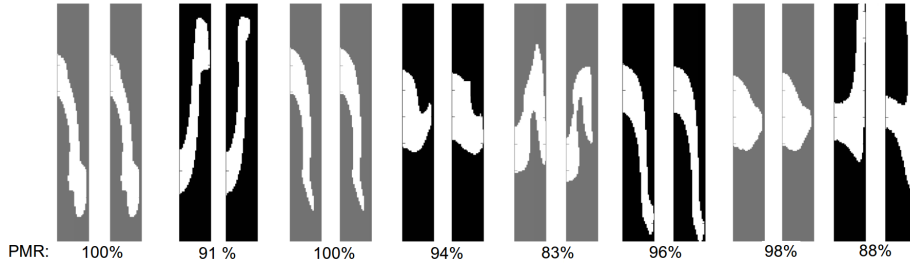


Figure 4: Eight example predictions ($n_p = 4$). The left side of each pair is the target flow shape and the right side is the flow generated from the predicted pillar sequences.

After the network makes a prediction, the resulting sequence is used to generate the flow shape which is then compared with the target shape as shown in Figure 4. The pixel match rate (PMR) is used as a performance metric. In simpler terms, given an original flow shape image and an image generated from the predicted pillar sequences using the DL model, two pixels at corresponding locations match if they both have the same color. Formally,

$$PMR(\%) = 100\% \left(1 - \frac{\|\mathbf{p} - \hat{\mathbf{p}}\|_1}{|\mathbf{p}|} \right)$$

where \mathbf{p} is the target image vector, $\hat{\mathbf{p}}$ is the predicted image vector, and $|\mathbf{p}|$ denotes the number of elements in the vector (i.e., the total number of pixels in the image).

Table 2 shows the PMR using both DNN and CNN architectures for a 4-pillar sequence. CNN clearly achieves a higher average similarity compared with DNNs and is able to predict the pillar sequence that generates a perfectly matching flow shape with the desired flow shape (i.e., $PMR = 100\%$). With CNN, 84.33% of the test samples scored a PMR of 80% and above which is much higher than using DNN. As the PMR threshold is increased, the performance of CNN remains superior than the other by a significant margin.

Table 3 shows the comparison of PMR score when the CNN is trained with a quasi-randomly generated training data (using Sobol sequences) with the expected PMR score with randomly generated training set. The model trained with quasi-randomly generated training data is able to generalize better and predict more accurate sequences.

Table 2: PMR-based performance for DNN vs CNN. The appended number 4 is the number of pillars in the sequence.

PMR(%)	DNN-4	CNN-4	Test Samples (%)	DNN-4	CNN-4
Min	56.90	52.83	\geq PMR 80	65.17	84.33
Mean	82.60	88.09	\geq PMR 85	49.59	70.89
Median	84.88	89.92	\geq PMR 90	27.20	49.50
Max	100.00	100.00	\geq PMR 95	5.81	18.47

Table 3: PMR-based performance for CNN with 7-pillar sequence that is trained with randomly-generated training data vs. quasi-randomly generated training data.

PMR (%)	CNN-7, Random	CNN-7, Quasi-random
Min	44.62	50.36
Mean	76.97	78.44
Median	77.46	79.44
Max	98.11	97.58

Remark 4.1 *As feature extraction concerns extracting informative, non-redundant features that facilitate the generalization procedure, efforts are usually placed to improve training algorithms or reframing the problem such that a different feature extraction technique can be applied. Better performance from using quasi-random training data suggests that domain knowledge should not be overlooked and can be incorporated into the data generation procedure to aid effective learning even with datasets severely limited in size.*

Remark 4.2 *What is a good PMR? Finding out the percentage of samples higher than a specific PMR threshold is particularly useful for design purposes. In the context of manufacturing, 84.33% of the predictions from the CNN is satisfactory if the design error tolerance is 20%. For biomedical applications as described in the introduction, a PMR of at least 80-85% is good enough to make reasonably efficient devices. Minor variations in shapes are allowable, as fluid diffusion anyway results in shape distortion. Essentially, this tool serves as the first step in determining pillar sequences within acceptable criteria without investing too much time in design iterations.*

4.3 Visualization of learned features

This section presents the visualization of learned features for both network structures used. Visualizing features shows that the network can extract meaningful features in the flow shapes that can be interpreted by the domain experts. For the DNN, network weights are visualized by converting the weight values in the first layer to pixel intensities and plotting the weights for each hidden unit in a rasterized manner. To obtain filter-like representations of learned features in the deeper layers, recently proposed activation maximization (AM) is used (Erhan et al. 2010b). This technique seeks to find inputs that maximize the activation of a specific hidden unit in a particular layer. Let θ denotes the parameters of the network (weights and biases) and $h_{ij}(\theta, \mathbf{x})$ be the value of the activation function $h_{ij}(\cdot)$ (usually the logistic sigmoid function) of hidden unit i in layer j on input \mathbf{x} . As the network has been

trained, θ remains constant. Therefore, the optimization process becomes:

$$\mathbf{x}^* = \underset{\mathbf{x} \text{ s.t. } \|\mathbf{x}\|=\rho}{\operatorname{argmax}} h_{ij}(\theta, \mathbf{x})$$

where \mathbf{x}^* denotes the inputs that maximizes the hidden unit activation. In general, the problem is a non-convex optimization problem. However, it is still useful to find the local optimum by performing a simple gradient ascent along the gradient of $h_{ij}(\theta, \mathbf{x})$, because the solution may be able to visualize the patterns of the inputs learned by the hidden unit.

Figure 5 visualizes the input image vectors that maximize activations of the hidden units. While lower hidden layers present a collection of structures that contributes to the overall flow shape, the final hidden layer shows identifiable flow shapes that are captured by the hidden units.

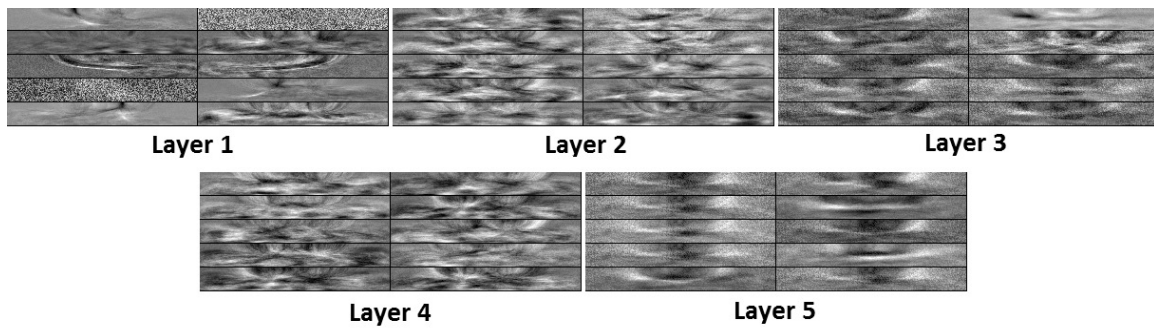


Figure 5: Visualization of hidden layer activations in the DNN.

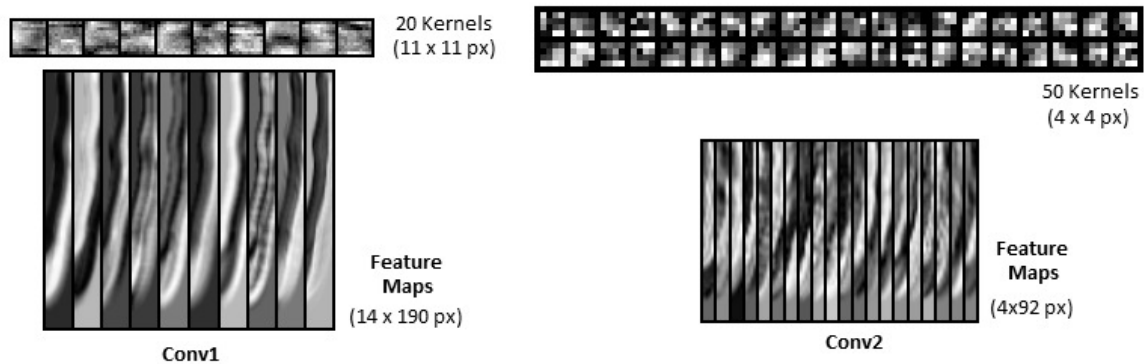


Figure 6: Visualization of CNN filters and feature maps of a test sample from convolutional layers 1 and 2.

In CNNs, learned features can be visualized by plotting the weight matrix of the feature maps in the same manner as obtaining filter-like representations from DNNs. Figure 6 shows the filters and convolved feature maps from both convolutional layers. Filters, with corresponding feature maps, are chosen at random to reveal the edge-like features that are learned by the model.

Remark 4.3 *The visualizations show that the hierarchical feature extraction processes can learn coherent structures in the flow shapes that can lead to better understanding of ingre-*

dients of the overall flow and hence the physics of complex flow patterns. For example, for different pillar sequences, changes in activation patterns (projected down to the visible layer) can signify fundamental implications of pillars on the flow shapes. In this context, although fully supervised CNN has a higher performance accuracy, it is quite difficult to project activation patterns down to visible layer as it involves (memory and computationally) expensive de-convolution process (Zeiler and Fergus 2014). On the other hand, DNN visualizations shown here demonstrate various features learned at different levels without large complexity. For instance, visualization of the 5th layer provides valuable insights to the diversity of flow shapes achievable through different pillar sequences.

4.4 Comparison with genetic algorithm

This section compares the PMR and execution time between the state-of-the-art GA and deep learning (DL) methods (using only CNN-SMC) for sequences with more than four pillars. Among the four examples shown in Figure 7, GA gives marginally better predictions in terms of quantitative accuracy. However, the DL outcomes are generally acceptable as the errors fall within the typical design threshold for this application.

On the other hand, as an evolutionary heuristic, the GA requires many fluid flow shapes to be produced and evaluated during a search. Therefore, the execution time for identifying a pillar sequence for a single image can take hours (Stoecklein et al. 2015). However, a trained deep network will give results with a runtime on the order of seconds. Furthermore, the choice of cost function will greatly impact the effectiveness of a GA search, and therefore requires laborious tuning. Given the time required for a single run, this tuning process can be considerably more complex and difficult than the training of a deep network. While the algorithm runtime using DL methods mentioned in the previous section includes the time needed to load the model and predict six different pillar sequences ($n_p = 5, 6, \dots, 10$) given an input, prediction actually occurs almost instantaneously since most of the time spent is used to load the model weights. Hence, DL methods allow for near real-time flow sculpting. More importantly in this case, the user could manipulate a flow shape using a graphical interface that would elicit key expert knowledge.













Target	GA Best	DL Best	Target	GA Best	DL Best	Target	GA Best	DL Best	Target	GA Best	DL Best
											
PMR	98.79%	95.88%	PMR	97.60%	93.15%	PMR	98.63%	95.81%	PMR	98.44%	92.48%
Time (s)	11,222.3	11.3	Time (s)	11,654.6	11.3	Time (s)	12,198.0	11.3	Time (s)	11,561.5	11.3

Figure 7: Comparison of pixel match rate and algorithm runtime for GA versus DL methods (using CNN-SMC) to generate pillar sequences that has the best shape reconstruction closest to the target flow shape. Four examples are provided.

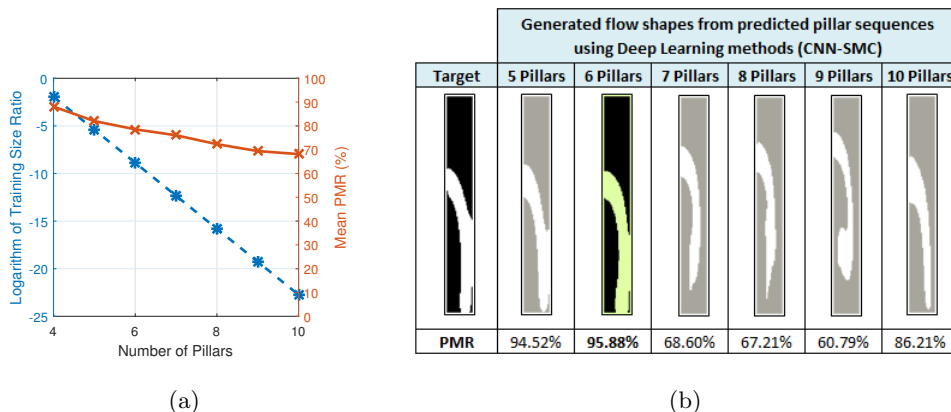


Figure 8: (a) Variation of the *average* PMR and training size ratio (ratio of training samples to the number of all possible pillar sequences, on a logarithmic scale) with increasing number of pillars in a sequence. (b) Flow shapes generated from predicted pillar sequences.

Another interesting aspect is that the number of possible pillar configurations (hence the size of the search space) increases exponentially with the number of pillars. With the training set kept constant at 150,000 samples in this current study, the ratio of training samples to the number of all possible pillar sequences shrinks very quickly with increasing pillar sequence length. Figure 8 (a) shows the capability of the hierarchical model under this “curse-of-dimensionality” challenge. Even with an exponential decrease in coverage of the design space, the PMR rate only degrades linearly. For example, 68% PMR is achieved (may be slightly less than acceptable) using only 150,000 training examples out of 10^{15} possible pillar configurations for the 10-pillar sequence. Figure 8 (b) shows the flow shape generated from predicted pillar sequences based on an example input flow shape. The PMR values for this example are consistent with the results obtained from Figure 8 (a). In the design context, the user will be presented with a selection of outputs to choose from. In this use case, the user may accept a small decrease in accuracy (95.88% to 94.52%) to gain possible financial savings by using fewer pillars (5 instead of 6).

5. Conclusions

This paper proposes a deep learning based approach to solve complex design exploration problems, specifically design of microfluidic channels for flow sculpting. We have demonstrated that DL based tools can achieve the required design accuracy. The state-of-the-art GA based method performs slightly better quantitatively, but it remains a less desirable solution due very long execution time. DL based methods can expedite the design process by more than 600 times and present a real-time design alternative. A quasi-random training data generation process was developed based on a metric on the output space. Such a process can be used for learning features from simulation and experimental data in many physical problems in general. The training data generation and hierarchical feature extraction processes together proves to be quite useful for a scalable design tool. Finally, feature visualization is performed to investigate flow pattern aspects at various scales. Current efforts are primarily focusing on optimizing the tool-chain as well as tailoring for specific application areas such as manufacturing and biology.

6. Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce GTX TITAN Black GPU used for this research.

References

- Hamed Amini, Elodie Sollier, Mahdokht Masaeli, Yu Xie, Baskar Ganapathysubramanian, Howard A. Stone, and Dino Di Carlo. Engineering fluid flow using sequenced microstructures. *Nature Communications*, 2013.
- S. Ashforth-Frost, V. N. Fontama, K. Jambunathan, and S. L. Hartle. The role of neural networks in fluid mechanics and heat transfer. In *Instrumentation and Measurement Technology Conference, 1995. IMTC/95. Proceedings. Integrating Intelligent Instrumentation and Control., IEEE*, page 6. IEEE, 1995.
- Modjtaba Baymani, Asghar Kerayechian, and Sohrab Effati. Artificial neural networks approach for solving stokes problem. *Applied Mathematics*, 1(04):288, 2010.
- James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.
- Dan C. Cireşan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2013*, pages 411–418. Springer, 2013.
- Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.
- Li Deng and Yu Dong. Foundations and trends in signal processing. *Signal Processing*, 7: 3–4, 2014.
- Thomas Duriez, Vladimir Parezanović, Laurent Cordier, Bernd R Noack, Joël Delville, Jean-Paul Bonnet, Marc Segond, and Markus Abel. Closed-loop turbulence control using machine learning. *arXiv preprint arXiv:1404.4589*, 2014.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010a.
- Dumitru Erhan, Aaron Courville, and Yoshua Bengio. Understanding representations learned in deep architectures. *Department d’Informatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep*, 1355, 2010b.
- Geoffrey E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.
- Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

- Geoffrey E. Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6), 2012.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- Lauwerens Kuipers. *Uniform distribution of sequences*. Dover Publications, Mineola, N.Y., 2006. ISBN 978-0486450193.
- Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008.
- Michael K. K. Leung, Hui Y. Xiong, Leo J. Lee, and Brendan J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.
- S. D. Müller, M. Milano, and Petros Koumoutsakos. Application of machine learning algorithms to flow modeling and optimization. 1999.
- William Press. *Numerical recipes in C the art of scientific computing*. Cambridge University Press India, New Delhi, 1992. ISBN 978-8185618166.
- Soumik Sarkar, Vivek Venugopalan, Kishore Reddy, Michael Giering, Julian Ryde, and Navdeep Jaitly. Occlusion edge detection in rgb-d frames using deep convolutional neural networks. *Proceedings of IEEE High Performance Extreme Computing (HPEC) Conference, (Waltham, MA)*, 2015.
- Ilya M. Sobol, Danil Asotsky, Alexander Kreinin, and Sergei Kucherenko. Construction and comparison of high-dimensional sobol generators. *Wilmott Journal*, page 6479, 2011.
- Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.
- Daniel Stoecklein, Chueh-Yu Wu, Keegan Owsley, Yu Xie, Dino Di Carlo, and Baskar Ganapathysubramanian. Micropillar sequence designs for fundamental inertial flow transformations. *Lab on a Chip*, 2014.
- Daniel Stoecklein, Chueh-Yu Wu, Donghyuk Kim, Dino Di Carlo, and Baskar Ganapathysubramanian. Optimization of micropillar sequences for fluid flow sculpting. *arXiv preprint arXiv:1506.01111*, June 2015. URL <http://arxiv.org/abs/1506.01111>.
- M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014.