Deep Action Sequence Learning for Causal Shape Transformation

Kin Gwn Lore  
Mechanical Engineering, Iowa State University  
Ames, IA 50011  
kglore@iastate.edu

Daniel Stoecklein  
Mechanical Engineering, Iowa State University  
Ames, IA 50011  
stoeckd@iastate.edu

Michael Davies  
Computer Engineering, Iowa State University  
Ames, IA 50011  /mdavies@iastate.edu

Baskar Ganapathysubramaniam  
Mechanical Engineering, Iowa State University  
Ames, IA 50011  
baskarg@iastate.edu

Soumik Sarkar  
Mechanical Engineering, Iowa State University  
Ames, IA 50011  
soumiks@iastate.edu

Abstract

Deep learning (DL) became the method of choice in recent years for solving problems ranging from object recognition and speech recognition to robotic perception and human disease prediction. In this paper, we present a hybrid architecture of convolutional neural networks (CNN) and stacked autoencoders (SAE) to learn a sequence of actions that nonlinearly transforms an input shape or distribution into a target shape or distribution with the same support. While such a framework can be useful in a variety of problems such as robotic path planning, sequential decision-making in games and identifying material processing pathways to achieve desired microstructures, this paper focuses on controlling fluid deformations in a microfluidic channel by deliberately placing a sequence of pillars, which has a significant impact on manufacturing for biomedical and textile applications where highly targeted shapes are desired. We propose an architecture which simultaneously predicts the intermediate shape lying in the nonlinear transformation pathway between the undeformed and desired flow shape, then learns the causal action—the single pillar which results in the deformation of the flow—one at a time. The learning of stage-wise transformations provides deep insights into the physical flow deformation. Results show that under the current framework, our model is able to predict a sequence of pillars that reconstructs the flow shape which highly resembles the desired shape. Learning of a multistep topological transform has significant implications for rapid advances in material science and biomedical applications.
1 Introduction

Hierarchical feature extraction using deep neural networks has been very successful in accomplishing various tasks such as object recognition [1], speech recognition [2], scene understanding [3], multi-modal sensor fusion [4], prognostics [5], engineering design [6], policy reward learning [7], and relating DNA variants to diseases [8]. This paper proposes a novel application of deep learning by fusing multiple architectures in solving design engineering problems, which has a great potential in accelerating the development of various fields such as manufacturing, chemical engineering, and biology.

In the paper, we propose a deep learning architecture which simultaneously predicts the intermediate shape between two images and learns a sequence of causal actions contributing to desired shape transformation. This topological transformation framework has multiple implications. The architecture can be easily implemented in applications such as learning to transform the belief space for robotic path planning [9], sequential decision making in games [10], learning the material processing pathways to obtain desired microstructures starting from an initial microstructure [11], and learning a sequence of manufacturing steps in additive manufacturing [12], with fast-design being the main advantage. The major contributions of the paper are outlined below:

1. A formulation of learning causal shape transformations to predict a sequence of transformation actions is presented, in the setting where only an initial shape and a desired target shape are provided.
2. An integrated hierarchical feature extraction approach using stacked autoencoders (SAE) [13] with convolutional neural networks (CNN) [14] is proposed to capture transformation features to generate the associated sequence resulting in the transformation.
3. The proposed approach is tested and validated via numerical simulations on an engineering design problem (i.e. flow sculpting in microfluidic devices), with results showing competitive prediction accuracy over previously explored methods.

2 Problem setup and previous approach

In this section, we describe the problem setup and provide a brief background on the previous approach. To illustrate how the previous and current architectures can be implemented, we will focus our attention on microfluidic flow sculpting as the application.

2.1 Microfluidic flow sculpting

Inertial fluid flow sculpting via micropillar sequences is a recently developed method of fluid flow control with a wealth of applications in the microfluidics community [15]. This technique utilizes individual deformations imposed on fluid flowing past a single obstacle (pillar) in confined flow to create an overall net deformation from a rationally designed sequence of such pillars. If the pillars are spaced far enough apart (space > 6D, for a pillar diameter D), the individual deformations become independent, and can thus be pre-computed as building blocks for a highly efficient forward model for the prediction of sculpted flow given an input pillar sequence [16] (Fig. 1). Since its debut, flow sculpting via micropillar sequence design has been used in fundamental applications of novel particle fabrication [12], and solution transfer [17].

However, practical applications require solving the inverse problem, that is, to generate a sequence of pillars given a user-defined flow shape. Without intelligent computer algorithms, such tasks require time-consuming trial and error design iterations. The automated determination of pillar sequences that yields a custom shape is a significant and impactful advance. Although researchers have tried to frame this inverse problem as an unconstrained optimization problem [16], they are invariable time-consuming. While many methods are used to solve the forward problem [18-21], only a limited amount of effort has been done in solving the inverse problem [6]. For practical and time-efficient applications, deep learning methods are explored to map user-defined flow shapes and the corresponding pillars of sequence. Additionally, this application serves as a very impactful advance towards learning topological transformations.

This framework can be utilized for concrete biomedical applications that require the design of microfluidic devices. Possible applications include: (a) designing a device to move fluid surrounding...
cells (e.g. lymphoid leukemia cells) against a far wall of the microfluidic channel where it can be collected at high purity while the cells are maintained at the channel center. High purity allows the reuse of valuable reagents for staining cells during diagnosis, (b) wrapping a fluid around the microchannel surface to characterize binding p24 (an HIV viral capsid protein) to anti-p24 antibody immobilized on the microchannel surface. Flow sculpting can enhance reaction of low abundance proteins that can improve diagnostic limits of detection for various diseases. Hence, this study may promise new application areas for the machine learning community related to thermo-fluid sciences and design engineering.

2.2 Discretization of the design space

To approach the inverse problem, class indices are assigned to pillars with different specifications. For instance, a pillar located at position 0.0 with a diameter of 0.375 is an index of 1, whereas another pillar at position 0.125 and diameter 0.375 is assigned an index of 2. Diameter and position values are represented as ratios with respect to the channel size and locations, so they are dimensionless quantities to help enable scalability of the fluid channel. Index assignment is performed over a finite combination of pillar positions and diameters that has been obtained by discretizing the design space. In the study, there are 32 possible indices (or classes) that describe the diameter and position of a single pillar.

2.3 Previous approach: Simultaneous multi-class classification (SMC)

Simultaneous multi-class classification is a method proposed in [6] to solve a similar problem. Instead of solving a single classification problem, the model solves a sub-problem for each pillar using the parameters learned by the CNN. This formulation requires a slight modification in the loss function: for pillar sequence with length \( n_p \), the loss function to be minimized for a data set \( D \) is the negative log-likelihood defined as:

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

where \( D \) denotes the training set, \( \theta \) 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 image. The total loss is computed by summing the individual losses for each pillar.

3 Proposed architectures

While CNN-SMC is capable of predicting a large number of different sequences in a total time of just seconds, the drawback is that the sequence length is constrained because a new model needs to be trained to generate a sequence with different lengths. Furthermore, the sequence which deforms into the target flow shape is predicted jointly and does not provide sufficient insight on the causal interplay between pillars causing the deformation. In this context, recurrent neural network (RNN)-like architectures [22] are, although scalable, deemed unsuitable because the elements in the output...
vector in our problem are generally independent of each other, unlike words in a sentence. A related concept is the spatial transformer network [23] where the localization network outputs geometrical transformation parameters; however we desire an exact class attributed to an arbitrary transformation function. In this work, we predict the pillar sequence one pillar at a time without disregarding causality, such that the produced sequences deform the flow into one that resembles more closely to the ultimately desired shape.

### 3.1 Pillar prediction network (PPN)

To predict the sequence of pillars that results in the desired flow shape, this paper introduces the notion of transformation learning. Fig. 3 shows the learning approach by supplying juxtaposed flow shapes, one before deformation, and one after, into a CNN that extracts relevant features and predicts the class of the pillar causing the deformation. In the training data generation procedure, the input is comprised of three parts: (i) the pre-deformed flow shape, which is produced from a randomly generated sequence of varying length with values up to the number of classes of a single pillar; (ii) a padding to prevent the convolutional kernels from picking up interfering features from the juxtaposed images; and (iii) the post-deformed shape produced from adding a random pillar index to the previous sequence. This newly-added pillar index will become the label to train the CNN. Essentially, the pillar prediction network, as the name suggests, predicts the index of the pillar (which describes its position and diameter in the flow channel) given a pre- and post-deformed shape.

When inferencing, the right portion (part iii. in Fig. 3b) of the input image is replaced by the final target shape. The input image is fed into the CNN to obtain a pillar index, which is added to an initially empty pillar sequence. Because the forward model (i.e. going from sequence to flow shape) is not at all computationally expensive and takes merely milliseconds, this pillar sequence is used to regenerate the current shape which replaces the left portion (part i.) of the input, while the right
portion (part iii., the target) remains the unchanged. The input enters the CNN again to obtain a second pillar index, and is subsequently added to the previously obtained pillar sequence. At this point, we have a sequence of length 2, where the sequence will then again used to regenerate a new current shape. The process is repeated until the current shape matches the target shape, or until a user-defined stopping criterion is met.

However, this method only works well for simpler target shapes. For more complex flow shapes (e.g. shapes with many sharp angles, jagged edges, swirls and curls; see Fig. 8 sample 20A for example), the transformation path may be highly nonlinear-- the current shape may never converge to the final desired shape. Furthermore, the training data covers a vanishingly small fraction of the design space, with coverage shrinking exponentially as the sequence length increases (i.e., an \( n_p \) sequence will result in \( 32^{nP} \) different combinations), so it is necessary to learn the transformations in a meaningful way. To help alleviate this issue, we will introduce the Intermediate Transformation Network (ITN) in the next subsection.

An alternative PPN is to feed the pre- and post-deformed shapes into the CNN separately with isolated channels before merging them together in the fully connected layer, which may or may not result in better classification performance. For the simplicity of analysis, simple model, and efficient error backpropagation, we decided to focus our analysis solely on the aforementioned formulation where flow shape images are in juxtaposition.

**PPN parameters:** The input image is comprised of two 12 × 100 px flow shape image with a 5 × 100 px padding in between, resulting in a final dimension of 29 × 100 px. The model contains two convolutional layers with 40 and 100 kernels respectively (sizes of 5 × 5 and 3 × 3 px), each followed by a 2 × 2 maxpooling layer. The fully connected layer has 500 hidden units. Training is done with 250,240 training examples and 60,000 validation examples in minibatches of 50 with a learning rate of 0.01. The training procedure employs the early stopping algorithm where training stops when validation error ceases to decrease. The hyperparameters are chosen via cross validation.

### 3.2 Intermediate transformation network (ITN)

The ITN attempts to construct a flow shape that bridges between the purely undeformed flow shape (i.e. shape generated with an empty sequence) and the final desired flow shape in the nonlinear transformation path (See Fig. 4). We used a deep autoencoder to extract hierarchical features from the desired final shape, and output an approximated bridging shape. To generate the training data for this network, the input image is generated with a random pillar sequence with a varying length. The corresponding bridging shape is generated by truncating the same pillar sequence by half, thus the shape lies in the middle of the transformation pathway (Fig. 4b). Formally, if a sequence has length \( n \), it is truncated to \( (n + 1)/2 \) if \( n \) is odd, and to \( n/2 \) if \( n \) is even. This pair of images is used to train a deep autoencoder, where the mean squared error (MSE) between the model outputs and the desired bridging shape is backpropagated to finetune the weights and biases of the deep autoencoder.

During inference, the edges in the outputs of the ITN may appear blurry because the bridging shape is only an approximation (Fig. 4c). To obtain pure black and white images, we threshold the pixel values where pixel intensities larger or equal to 0.5 are set to 1, 0 otherwise.

Figure 4: The ITN addresses the question: ‘Given the final shape, what is the possible shape that lies in the middle of the nonlinear deformation pathway?’
The ITN may be extended to obtain several *waypoints* instead of only the midpoint. Doing so will allow a smoother transformation pathway, but will require some changes in the training data generation procedure. We call this the recursive ITN (rITN) and leave this as a future work.

**ITN parameters:** The autoencoder has 3 layers of 500 hidden units each and accepts flow shape images of $12 \times 100$ px. Training was done on 500,000 training examples and 20,000 validation examples in minibatch of size 1,000 with a learning rate of 0.01. Training is also done using the early-stopping algorithm. The hyperparameters are chosen via cross validation.

### 3.3 The integrated pipeline (PPN+ITN)

![Diagram of the integrated pipeline combining both PPN and ITN.](image)

With the roles of the PPN and ITN clearly described, we can now integrate both networks to form an integrated pipeline. A schematic is shown in Fig. 5. At the very beginning, the ITN guesses a candidate for the bridging flow shape, which is then considered as the temporary target shape, and is concatenated with the undeformed shape placed to its left with a padding. The concatenated input is supplied into the PPN to predict the first pillar causing the deformation, and then added to the sequence which was initially empty, hence resulting in a sequence of length 1. The current shape is regenerated with the updated 1-pillar sequence and replaces the left portion of the input image for the next PPN iteration to obtain the second pillar index. The process is repeated as ‘Stage A’ (with the bridging shape as a temporary target) until the current shape is sufficiently similar to the bridging shape, or until an iteration limit is reached. Then, the right portion of the input image (which was the bridging shape) is replaced with the final desired shape as the target. This process is continued as ‘Stage B’ which undergoes the same process as ‘Stage A’, except the target shape is now the final desired shape. After each iteration, the predicted pillar index is added to the sequence until the reconstructed flow shape matches the desired shape or a stopping criterion is achieved. The resulting sequence will vary in length for different desired shapes.

### 4 Results and discussions

In this section we first present the evaluation metrics used in the study, then show the results comparing CNN-SMC against our method using PPN and the PPN+ITN hybrid architecture.

#### 4.1 Performance evaluation metrics

To quantify the effectiveness of both approaches, we evaluated the *pixel match rate* (PMR) on 20 target flow shapes and computed appropriate statistics. 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. The PMR, defined in [6], is computed as...
follows:

$$PMR = 1 - \frac{\|p - \hat{p}\|_1}{|p|}$$

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

As a supplementary metric, the structural similarity index (SSIM) [24] is used to compare how structurally similar are the regenerated flow shape images (from predicted sequence) to the target flow shape. SSIM explores the change in image structure and incorporates pixel inter-dependencies. SSIM is expressed as:

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

where $\mu_x$ is the average of window $x$, $\mu_y$ is the average of window $y$, $\sigma_x^2$ is the variance of $x$, $\sigma_y^2$ is the variance of $y$, $\sigma_{xy}$ is the covariance of $x$ and $y$, $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$ are two variables to stabilize the division with weak denominator with $k_1 = 0.01$ and $k_2 = 0.03$ by default, and $L$ is the dynamic range of pixel values.

Figure 6: Four examples of sequence prediction using (i) PPN-only without bridging and (ii) PPN+ITN with bridging. By predicting a bridging shape, the resulting predicted sequence is able to reconstruct flow shapes that are more similar to the target shape. Each frame shows the deformation on the flow shape with each additional predicted pillar added into the sequence.

4.2 Predicting sequences with PPN and ITN

In all of our tests, the target flow shape is generated from a 10-pillar sequence which is sufficiently complex. Fig. 6 shows four example target shapes with the performance using only PPN (without bridging) and PPN+ITN (with bridging). In most cases, the PPN-only formulation produces pillar sequences resulting in flow shapes that do not resemble the target shape as close as using PPN+ITN. This can be clearly seen for cases C and D in Fig. 6. The prediction performance saw great improvements by using the bridging shape as a temporary target. In addition, we see that most shapes are able to converge to both the bridging shape as well as the target in the PPN+ITN formulation.

In realistic applications, the sequence may be stored in memory and post-processed to remove the
Figure 7: Sample-wise (a) PMR and (b) SSIM comparison using CNN-SMC, PPN, and PPN+ITN. 20 test target shapes are randomly generated with a 10-pillar sequence.

redundant pillars, thus producing a shorter sequence. This is beneficial to microfluidic applications due to a smaller on-device footprint, less diffusive blurring from fluid time-of-flight, and lower pressure requirements to drive flow through the microchannel [15].

4.3 Comparison with CNN-SMC

Figure 8: 20 test shapes with reconstructed flow shapes generated from sequences predicted using different methods. Column A is the target shape, B for the reconstruction for CNN-SMC, C for PPN, and D for PPN+ITN.

20 sample-wise comparison on the performance of CNN-SMC, PPN and PPN+ITN are shown in Fig. 7 and Table 1. The reconstructed flow shapes from the predicted sequences are shown in Fig. 8. In both Fig. 7a and Fig. 7b, the PMR and SSIM for PPN and PPN+ITN are clearly higher than the CNN-SMC approach. We observe that for some target flow shapes, the predicted sequence for CNN-SMC may result in an entirely dissimilar shape (e.g. sample 2, 19, 20). In some cases (e.g. samples 14, 15, 16) PPN fared better than the hybrid PPN+ITN model. However, PPN+ITN is consistently more superior in terms of both PMR and SSIM than the PPN-only architecture. This shows that having a bridging shape generally helps in producing sequences that generate complex flow shapes. The inferior performance of CNN-SMC is due the model being constrained to always generate a sequence with 10 pillars, thus the resultant flow shapes often become overcomplicated.

A clear advantage of using both PPN and ITN together is that the model does not need to be retrained for variable sequence lengths, unlike in the CNN-SMC model where the number of pillars in the output sequence is constrained. Therefore, only a single model is needed for deployment, rather than multiple models accounting for different pillar sequence lengths. This method is highly scalable, and has an enormous room for extension into sculpting more highly complex flow shapes.
Table 1: PMR and SSIM of regenerated flow shapes using different enhancement methods. The numbers reported are in the format of [PMR/SSIM]. Asterisk (*) denotes our architecture.

| Target Flow Shapes | CNN-SMC | PPN* | PPN+ITN* |
|--------------------|---------|------|----------|
| Test Shape 1       | 0.5600 / 0.1829 | 0.8317 / 0.4349 | 0.8475 / 0.4521 |
| Test Shape 2       | 0.5133 / 0.0578 | 0.9375 / 0.7160 | 0.9117 / 0.5899 |
| Test Shape 3       | 0.9367 / 0.6622 | 0.9342 / 0.6437 | 0.9558 / 0.7481 |
| Test Shape 4       | 0.5883 / 0.1813 | 0.8833 / 0.5303 | 0.8950 / 0.6082 |
| Test Shape 5       | 0.8275 / 0.4539 | 0.9650 / 0.8108 | 0.9725 / 0.8407 |
| Test Shape 6       | 0.6992 / 0.2705 | 0.9208 / 0.6765 | 0.9358 / 0.7346 |
| Test Shape 7       | 0.6300 / 0.1412 | 0.9567 / 0.7895 | 0.9558 / 0.8167 |
| Test Shape 8       | 0.6117 / 0.2355 | 0.8125 / 0.4687 | 0.9075 / 0.6330 |
| Test Shape 9       | 0.7767 / 0.3743 | 0.9117 / 0.5776 | 0.9292 / 0.6105 |
| Test Shape 10      | 0.7883 / 0.4407 | 0.9500 / 0.7745 | 0.9625 / 0.8299 |
| Test Shape 11      | 0.5892 / 0.2046 | 0.8608 / 0.4137 | 0.9275 / 0.6511 |
| Test Shape 12      | 0.6058 / 0.1430 | 0.9025 / 0.5527 | 0.9567 / 0.6314 |
| Test Shape 13      | 0.7200 / 0.2126 | 0.9442 / 0.7458 | 0.9433 / 0.7233 |
| Test Shape 14      | 0.5983 / 0.2229 | 0.9433 / 0.6804 | 0.8850 / 0.5466 |
| Test Shape 15      | 0.6808 / 0.3412 | 0.8933 / 0.5195 | 0.8583 / 0.4332 |
| Test Shape 16      | 0.5508 / 0.1135 | 0.9433 / 0.6919 | 0.9200 / 0.5813 |
| Test Shape 17      | 0.7792 / 0.4134 | 0.9033 / 0.5882 | 0.9617 / 0.7655 |
| Test Shape 18      | 0.6133 / 0.2677 | 0.9075 / 0.6612 | 0.9267 / 0.6416 |
| Test Shape 19      | 0.6075 / 0.0464 | 0.9367 / 0.7540 | 0.9567 / 0.8044 |
| Test Shape 20      | 0.5300 / 0.0134 | 0.8983 / 0.6039 | 0.9183 / 0.6753 |
| Average Score      | 0.6603 / 0.2500 | 0.9118 / 0.5290 | 0.9331 / 0.6609 |

5 Conclusions and future work

This paper proposes a deep learning based approach to learn a sequence of actions that carries out the desired transformation over an input 2D shape, which has potentially high impact on the innovation of manufacturing processes, material sciences, biomedical applications, decision planning, and many more. In the paper, we specifically focused in engineering microfluidic channels for flow sculpting. We demonstrated that creative integration of DL based tools can tackle the inverse fluid problem and achieve the required design accuracy while expediting the design process compared to laborious trial-and-error design iterations. The training data generation and hierarchical feature extraction processes together prove to be quite useful for a scalable design tool. Current efforts are primarily focusing on optimizing the tool-chain (e.g. rITN), defining and quantifying the complexity of a flow shape, as well as tailoring for specific application areas such as manufacturing and biology.

References

[1] 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.

[2] 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.

[3] Soumik Sarkar, Vivek Venugopalan, Kishore Reddy, Michael Giering, Julian Ryde, and Navdeep Jaitly. Occlusion edge detection in rgb-d frames using deep convolutional networks. Proceedings of IEEE High Performance Extreme Computing Conference, Waltham, MA, 2015.

[4] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In Advances in neural information processing systems, pages 2222–2230, 2012.

[5] Soumalya Sarkar, Kin Gwn Lore, and Soumik Sarkar. Early detection of combustion instability by neural-symbolic analysis on hi-speed video. In Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCo@ NIPS 2015), Montreal, Canada, 2015.

[6] Kin Gwn Lore, Daniel Stoecklein, Michael Davies, Baskar Ganapathysubramanian, and Soumik Sarkar. Hierarchical feature extraction for efficient design of microfluidic flow patterns. In Proceedings of The 1st International Workshop on “Feature Extraction: Modern Questions and Challenges”, NIPS, pages 213–225, 2015.
[7] Kin Gwn Lore, Nicholas Sweet, Kundan Kumar, Nisar Ahmed, and Soumik Sarkar. Deep value of information estimators for collaborative human-machine information gathering. *International Conference on Cyber-physical Systems (ICCPs). Vienna, Austria.*, 2016.

[8] 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.

[9] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[10] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[11] Olga Wodo, Jaroslaw Zola, Balaji Sesharath Pokuri, Pengfei Du, and Baskar Ganapathysubramanian. Automated, high throughput exploration of process–structure–property relationships using the mapreduce paradigm. *Materials Discovery*, 2015.

[12] Kevin S. Paulsen, Dino Di Carlo, and Aram J. Chung. Optofluidic fabrication for 3D-shaped particles. *Nature Communications*, 6:6976, 2015.

[13] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[14] 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.

[15] 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.

[16] Daniel Stoecklein, Chueh-Yu Wu, Donghyuk Kim, Dino Di Carlo, and Baskar Ganapathysubramanian. Optimization of micropillar sequences for fluid flow sculpting. *Physics of Fluids, vol.28*, 2016.

[17] Elodie Sollier, Hamed Amini, Derek E. Go, Patrick a. Sandoz, Keegan Owsley, and Dino Di Carlo. Inertial microfluidic programming of microparticle-laden flows for solution transfer around cells and particles. *Microfluidics and Nanofluidics*, 19:53–65, 2015.

[18] S. D. Müller, M. Milano, and Petros Koumoutsakos. Application of machine learning algorithms to flow modeling and optimization. 1999.

[19] 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.

[20] Modjtaba Baymani, Asghar Kerayechian, and Sohrab Effati. Artificial neural networks approach for solving stokes problem. *Applied Mathematics*, 1(04):288, 2010.

[21] 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*, page 6. IEEE, 1995.

[22] Tomas Mikolov, Martín Karafiát, Lukas Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2008–2016, 2015.
[24] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.