



Inverse design of microfluidic concentration gradient generator using deep learning and physics-based component model

Seong Hyeon Hong¹ · Haizhou Yang¹ · Yi Wang¹

Received: 31 December 2019 / Accepted: 4 May 2020 / Published online: 16 May 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

This paper presents a new paradigm of deep neural network (DNN) for the inverse design of microfluidic concentration gradient generators (μ CGGs) with complex network topology. In this method, a concentration gradient (CG) and design parameters yielding the CG are, respectively, used as inputs and outputs of DNN, and the relationship between them is mapped. Several new elements are also proposed, including utilization of fast-running physics-based component model in the closed form to generate a large amount of data for DNN learning which otherwise is not available through computationally demanding computational fluid dynamics (CFD) simulation; and a divide-and-conquer strategy and DNN formulation combining classification and regression to mitigate many-to-one design complications for enhanced accuracy. Several DNN structures are investigated and developed, including single fully connected neural network (FCNN), convolutional neural network, and a new cascade FCNN for a divide-and-conquer implementation. Case studies are performed on a triple-Y μ CGG to evaluate design performance of the proposed method in a six-dimensional space that only includes sample concentrations at inlet reservoirs as design parameters, and in a nine-dimensional design space, to which inlet flow pressures are also added. It is verified in high-fidelity CFD simulation that widely used CGs can be produced using DNN-predicted design parameters accurately with average error $< 4\%$ and $< 8.5\%$ relative to the prescribed CGs, respectively, in the six- and nine-dimensional design space. The learned design rules are packaged into the DNN model that allows to generate accurate μ CGGs designs instantaneously (~ 3 ms) and eliminates requirements of simulation and optimization knowledge, facilitating distribution of the design capabilities to microfluidic end users.

Keywords Concentration gradient generator · Deep neural network · Microfluidics · Inverse design

1 Introduction

Generating concentration gradients (CGs) of biomolecules in a microfluidic environment is essential for studying biological processes. It has been found that CGs govern many biological processes, such as immune response, inflammation, wound healing, embryogenesis, and cancer metastasis (Keenan et al. 2008; Gupta et al. 2010; Meier et al. 2010; Kothapalli et al. 2011; Nandagopal et al. 2011). Therefore, numerous microfluidic concentration gradient generators (μ CGGs) have been developed and applied to in vitro systems to perform a variety of cell assays. Compared to their traditional counterparts at the macroscale, μ CGGs are

capable of generating gradients of fine resolution and providing highly controlled environment with salient flexibility. Additional advantages offered by μ CGGs include the generation of complex shapes and reduced sample consumptions (Dertinger et al. 2001; Wang et al. 2017).

There are various types of μ CGGs, including the tree-shaped, Y-shaped, flow splitter, pressure balance, membrane, droplet-based generators, among others (Toh et al. 2013; Hu et al. 2017; Wang et al. 2017). Each of them has its own merits and drawbacks, and the μ CGG needs to be selected and designed carefully to meet the criteria of a specific biological assay. The most common μ CGG is the tree-shaped network as reported in numerous literature studies (Vozzi et al. 2010; Gao et al. 2012; Wang et al. 2015; Chen et al. 2017; Hu et al. 2018), which has less stringent modeling and design requirement for a variety of concentration gradients (CGs), that is, it can be simulated and analyzed based on electric circuit analogy that does not involve species transport (Lee et al. 2010;

✉ Yi Wang
yiwang@cec.sc.edu

¹ University of South Carolina, Columbia, SC 29208, USA

Oh et al. 2012) because it relies on complete mixing of the target samples within all the branch channels before stream merging. As a result, it, in general, requires longer channels, more frequent flow split and merge, or more complex structures to induce chaotic mixing, which leads to larger footprint or more vulnerability to clogging and leakage (Wang et al. 2006). On the other hand, μ CGGs that utilize partial mixing through laminar flow diffusion, such as the Y-shaped μ CGG and their cascade combinations as reported by Wang et al. (2006) and Zhou et al. (2009), yield simpler microfluidic structures to enhance device reliability. A disadvantage of partial mixing-based μ CGG networks is that the CGs are generated via species transport in all channels rather than simply merging fully mixed streams with different average concentrations. Therefore, in addition to the concentrations at channel inlets, the CG also depends on sample's residence time within each channel, which introduces the flow velocity or inlet pressure as additional parameters for design. It is more challenging to simulate, predict, and design the CG produced by the CGG devices in a rigorous manner, which normally needs sophisticated computational fluid dynamics (CFD) simulation using the finite element or finite volume analysis package (Wang et al. 2006; Vozzi et al. 2010; Chen et al. 2017; Hu et al. 2018). In a previous effort, one of our coauthors presented a lego-like, system decomposition methodology along with a set of physics-based component models (PBCM) to analyze laminar flow μ CGGs based on partial mixing and species transport (Wang et al. 2006). The rationale is to break down a μ CGG network of complex topologies into a system of parameterized, commonly used, constituent components, including a microchannel, a combiner, a splitter, and a reservoir. The decomposition allows the closed-form solution of the governing equation of each component, and the component model then can be connected in correspondence to the device topology. Because of its analytical solution nature, the models offered speedup of three orders of magnitude over CFD simulation and were demonstrated to simulate a plethora of μ CGGs that exploit partial mixing and species transport to linear, sawtooth, and bell shapes, and the model was verified by both high-fidelity CFD simulation (Wang et al. 2006) and experiments (Zhou et al. 2009). Therefore, these physics-based models represent a viable solution to generate a massive amount of data for deep learning-based engineering modeling and design, which otherwise is not available through traditional, computationally demanding CFD simulations. Note that the component models are also applicable to the aforementioned CGGs based on complete mixing, e.g., the tree-shaped, as complete mixing is only a special case of species transport (Wang et al. 2006).

One of the foremost challenges in μ CGG development is to determine inlet parameters, such as the sample concentration and flow pressure/velocity at the inlet reservoir that can precisely produce target CGs prescribed by users. It is a

typical design optimization problem that requires numerous iterations to search the design parameters that minimize a cost function quantifying discrepancy between the CG generated by the candidate design and the prescribed CG. However, research to automate the design optimization process for μ CGG applications, in particular, for partial mixing and species transport-dominated μ CGGs is scarce, which may be attributed to several factors: (1) for CGs of simple profiles, e.g., a linear shape or an abrupt gradient, inlet parameters can be estimated fairly easily with a few iterations, i.e., trial and error, because of the small number of design parameters; and (2) automated optimization taking advantage of iterative simulation is often computationally prohibitive, which becomes even exacerbated for partial mixing-based CGGs where the simplified electric analogy model is not applicable due to dominant species transport. Therefore, it is necessary to use CFD simulation to resolve species transport in the network and utilize the global optimization method to alleviate the issue of local optimum caused by a large set of design parameters. Although the PBCM of CGGs above makes it possible to reduce the computational burden significantly, the process to replace the trial and error practice for automated design optimization based on PBCM is still unavailable and strongly desired, especially when GCs of more complex profiles are considered and the dimension of design parameter space grows dramatically. Along this frontier, Zhou et al. (2009) formulated an iterative simulation approach and applied the PBCM above to determine feasible concentrations at inlets to generate prescribed sawtooth (three line segments with discontinuity) and double-bell CGs using multi-stream laminar flow, in which, however, the iteration was performed manually without the optimization engine. Friedrich et al. (2012) combined the random search and the Nelder–Mead optimization algorithms to find the inlet conditions to produce linear and exponential (power) CGs, and the commercial finite element analysis package was utilized. Wang et al. (2019) applied the non-dominated sorting genetic algorithm II (Deb et al. 2002) upon their nearest neighbor approach to find the optimum design of the random mixer network. The efforts above included forward model simulation in the optimization loop. That is, given a candidate design the observables/measurements (CGs) predicted by the model will be used to derive another candidate design potentially yielding a closer match with the user-prescribed CG. It is a typical process for solving the forward design problem, and needs to be performed for each specific prescribed CG, which, in general, takes a long time to reach the optimal design and requires extensive modeling and optimization knowledge from designers.

To address the limitation, this paper presents a new paradigm for the inverse design of μ CGGs that harnesses deep neural network (DNN) architecture to learn design rules embedded in simulation data generated by the PBCM. More specifically,

in the inverse design, the CGs will be treated as the input, and the DNN will immediately produce the design parameters as the output that will produce a CG resembling the input most closely without incurring iterative simulation or optimization. This will not only enable fast and distributable design capabilities but also eliminate the need for pre-knowledge of modeling and optimization of μ CGGs from the designer/user. It should be pointed out that since the mapping relationship encoded in the DNN is essentially in the opposite direction relative to the forward model simulation above (that takes design parameters as input and outputs corresponding CGs), we use the terminology of inverse design throughout this paper. In addition, design rules will be learned by DNN using data generated prior to the actual design process, which is also different from traditional forward design optimization. Besides DNN, the database query-type approach, e.g., the nearest neighbor can also be used to select the optimum design from a library of possible designs that are precomputed by physics-based simulations. For instance, Wang et al. (2016) simulated thousands of random designs of microfluidic mixers using finite element analysis to build a library and selected the best design that produces the closest desired output by library search and match. However, building a library with FEA simulations is extremely computationally demanding for the case involving multiple continuous design variables, because it is infeasible to include all possible designs due to the curse of dimensionality. Therefore, it is a method for approximate design.

Recently, deep learning-based design has emerged as a dynamic research area and been investigated for several micro–nano-applications, including multiscale molecular dynamics simulation (Asproulis and Drikakis 2009, 2013; Drikakis and Frank 2015), nanophotonic devices (Malkiel et al. 2018; An et al. 2019; So et al. 2019), microfluidic droplet flow pattern recognition (Hadikhani et al. 2019), microfluidic flow sculpting (Stoecklein et al. 2017; Lore et al. 2018) and other microfluidic biotechnology applications (Riordon et al. 2018) because of its great promise to offer a rapid design solution in high-dimensional parameter space. Despite these seminal efforts, this paper presents several novelties, including (1) the utilization of system decomposition and fast-running, physics-based component models (PBCM) in the closed form to capture complex species transport, in general, μ CGGs and generate adequate training data, representing a different solution to bridging the gap between engineering design and deep learning. It is well known that a large volume of data is normally required to train the network of deep architecture, which precludes the use of high-fidelity, computationally demanding CFD simulations for training data generation within the resource-limited environment; (2) DNN-based inverse design of μ CGGs, in particular, those dominated by species transport has not been reported in the literature; and (3) a “divide and conquer” strategy to break down a design problem into a

series of DNN classification and regression tasks to tackle the many-to-one (non-unique) design issue and accommodate design parameters of different types.

The remainder of this paper is organized as follows. In Sect. 2, the methodology of the proposed inverse design framework, including the DNN formulation, μ CGG under consideration, physics-based component models, and the verification process, is thoroughly described. In Sect. 3, DNN implementation and design problem specification for two case studies are presented. The CGs obtained from DNN-predicted design parameters and their comparison with the prescribed CGs are given in Sect. 4 along with detailed discussion. Lastly, Sect. 5 concludes the paper with a summary of the accomplishments.

2 Methodology

In this section, we will first describe the deep neural network (DNN)-based inverse design, and then the physics-based component modeling to generate training, validation, and testing data. The performance of DNN testing and inverse design is also evaluated.

2.1 Inverse design

The method and framework of DNN-based inverse design and physics-based models are illustrated in Fig. 1. A DNN is an artificial neural network (ANN) with multiple layers of neurons that can approximate a complex relationship between two distinct spaces. Figure 1 shows that the inputs and the outputs of the DNN are, respectively, the prescribed CG and the inlet parameters. More specifically, the DNN input is normalized CG values along the microchannel width at the detector location. In this paper, the CG is measured with 100 uniformly distributed probes along the width direction, leading to a resolution of 12 μ m by each probe. Thus, the input of the DNN must be a vector with the values between 0 and 1. The output of the DNN is the inlet parameters used to create the CG supplied at the input, which include the normalized sample concentrations and the pressures at the in-flow/inlet reservoirs. On the contrary, the input and output of PBCM above are, respectively, the inlet parameters and the corresponding CG. Since the inputs and outputs for PBCM simulation and DNN-based design are reversed, the proposed method is termed inverse design. It should be noted that our DNN is formulated as a regression problem or a hybrid problem of regression and classification rather than classification only and, hence, allowing prediction of continuous design parameters at the inlets.

To generate adequate data for training, validation, and testing of DNN, a large number of simulations representing various combinations of the inlet sample concentrations and the

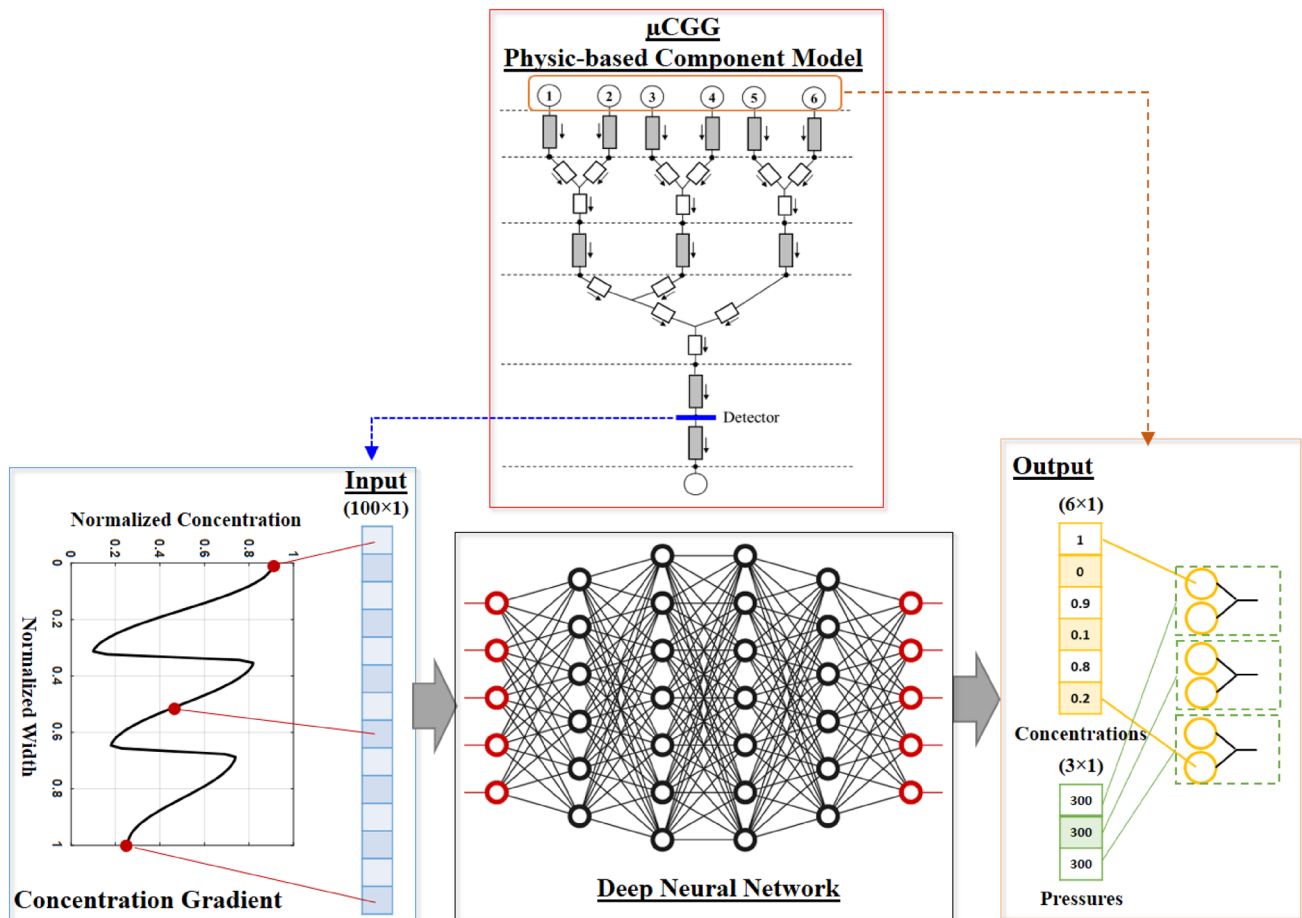


Fig. 1 Schematic of the deep neural network-based inverse design framework

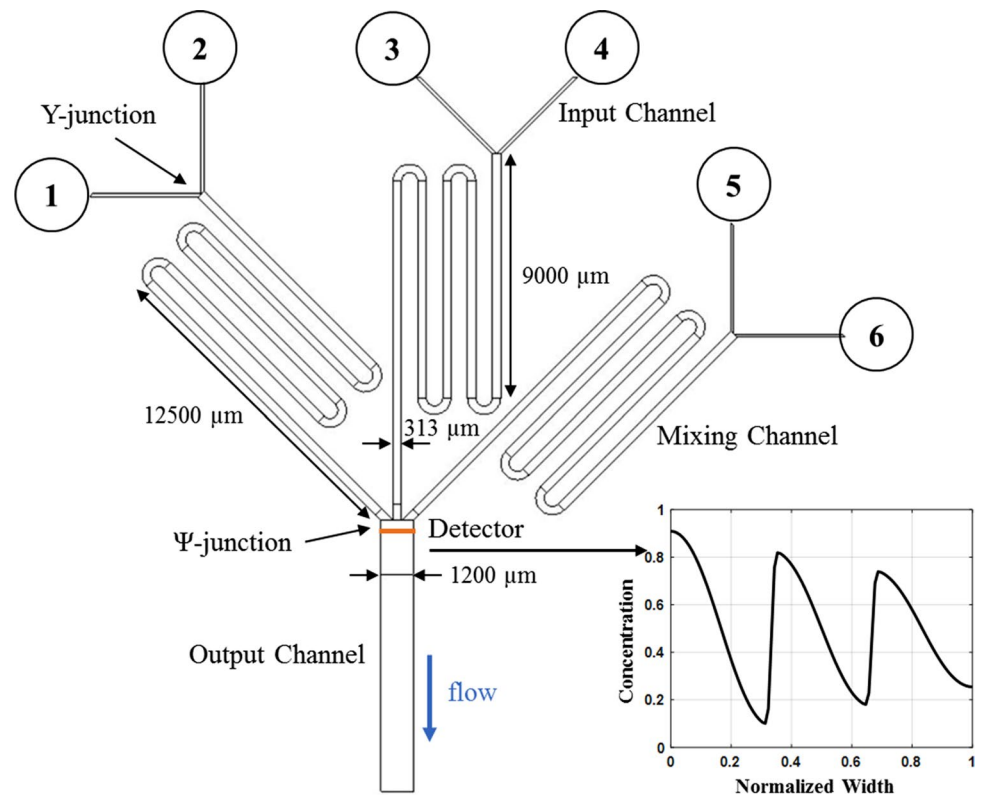
pressures in design space are created by the Latin hypercube sampling (LHS). Each combination of these parameters is then used as the input of PBCM simulation to predict the corresponding CG at the detector location, which as discussed above are reversed in DNN-based inverse design. In the present effort, DNNs of various architectures are trained using the Pytorch (Paszke et al. 2019), a machine learning library in the Python programming language. Given the massive data and the large number of network weight parameters associated with DNN, GPU computing is performed on NVIDIA GeForce GTX 1080 Ti to accelerate the training process, which turns out to be almost four times faster than CPU for this particular work.

2.2 Microfluidic concentration gradient generator and physics-based component modeling

As shown in Fig. 2, a triple-Y μ CGG that was presented in our previous papers (Wang et al. 2006; Zhou et al. 2009) and comprised of three Y-shaped mixers connected in parallel capable of generating complex CGs is adopted for analysis and design verification. From the figure, all the

microchannels and mixers have a consistent height of 60 μm . In each Y-shaped mixer, there are two inlets and one outlet. Two flow streams with different sample concentrations enter the Y-junction via the two inlets and exit the mixing microchannel as a single stream. Due to the small Reynolds number of microfluidics, molecular diffusion is the dominant mechanism to mix sample species from both streams. Thus immediately after the junction, a step-like CG is generated as the sample from both streams has no time yet to diffuse transversely. At the end of the mixing channel, a sigmoid or an approximately linear concentration profile is generated because of inter-stream diffusion that smears out the discontinuity of sample concentrations between the two streams. In addition, a flat concentration profile can also form as a result of rapid mixing or minor difference in the sample concentration between the two streams. As reported before (Wang et al. 2006; Zhou et al. 2009), the central portion of the formed CG exhibits good linearity, and those close to the side walls are slightly bent due to the impermeability of walls to species transport. The constituent linear CG from each Y-shaped mixer is then juxtaposed laterally, resulting in

Fig. 2 Schematic of the triple-Y microfluidic concentration gradient generator with a consistent height of 60 μm



temporally and spatially stable CGs of more complex shapes around the Ψ -junction, viz., the entrance of the main output channel. For example, by combining approximately linear CGs from the Y-shaped mixers, bell-shaped, sawtooth-shaped, or exponential-shaped CG can be created. It should be noted that first, the inlet concentration and pressure (or flow rate) of each Y-shaped mixer can be used to tune the constituent CG at its end, including the slope, peak, and mean concentration values. The effect of the pressure on the CG is manifested through the sample's residence time within the mixing channel downstream of the Y-junction. Second, the juxtaposed CG, in particular, the width of each constituent CG and the discontinuous interface between them observed at the detector is determined collectively by all the inlet concentrations and the inlet pressure. Third, the present effort focuses on the triple-Y μCGG , which has six inlets numbered 1 through 6 in Fig. 2. Six streams with possibly different sample concentrations are pumped into the three Y-shaped mixers, generating three constituent linear profiles, which are then combined at the Ψ -junction of the main output channel. For this particular device, three pressures, one for each Y-shaped mixer are employed to control the flow rate, the extent of mixing, and the width of the constituent CG emanating from each mixer. That is, the pressure at inlets 1 and 2 is the same, which can be different from that at inlets 3 and 4 and that at inlets 5 and 6. When all three pressures are set equal, the three constituent CGs have the

same stream width in the combined one as shown by the inset on the right of Fig. 2. Nonetheless, if they are different, their widths are different depending on the flow rate of each stream. A stream driven by a higher pressure, viz., a larger flow velocity will occupy a larger portion of the width of the main output channel and vice versa. One caveat is that if the pressure differences among the three Y-shaped mixers are overly large, backflow, that is, flow from one Y-shaped mixer to another may occur, causing the fluid flow to exit through inlets. In the present work, backflow is prohibited by constraining the pressure differences. In summary, there are nine design variables for the triple-Y CGG, including six sample concentrations and three pairs of pressures at six inlets (1–2, 3–4, 5–6), and the design goal is to seek design parameters that could produce the CG at the detector matching the prescribed/target CG. Other geometric parameters of the triple-Y CGG are also given in Fig. 2, and the detector is located at 400 μm downstream of the inlet of the main output channel.

2.3 Physics-based component model

As reported extensively in the literature, CFD simulation serves as the main physics-based modeling for CGG analysis. The issue with CFD is that although effective for verification, it is ill suited for design optimization due to its demanding computational costs, which can be even

exacerbated for DNN-based inverse design where a large volume of data is needed for network training, validation, and testing.

In this context, PBCM and associated systematic simulation method developed by us (Wang et al. 2006; Zhou et al. 2009) is exploited. Similar to a lego-like approach, it decomposes a complex CGG into several commonly used components, such as the microchannel, the combiner, the splitter, and the reservoir. Although not available to the entire CGG network, the analytical, closed-form solution of the governing species transport equation can be obtained for these simple components. These PBCMs then can be linked to represent desired μ CGG network topology as shown in Fig. 3.

For the sake of completeness, PBCM is briefly introduced herein. The details for all component models above are given elsewhere (Wang et al. 2006; Zhou et al. 2009). The pressure and flow in the μ CGG are solved using the electric circuit analogy method (Oh et al. 2012). For a microchannel, the pressure (p) at its inlet and outlet and its fluidic resistance are given by

$$p_{\text{out}} = p_{\text{in}} + Rq, R = \frac{12\beta l\alpha}{w^4 \left[1 - \frac{192\beta}{\pi^5} \sum_{i=1,3,5,\dots}^{\infty} \frac{\tanh(i\pi/2\beta)}{i^5} \right]}, \quad (1)$$

where q is the volumetric flow rate, R is the flow resistance, β is the channel's aspect ratio (width/depth), l is the channel length, α is the dynamic viscosity, and w is the channel width. The combiner and splitter, which are essentially junctions, are treated as a point-wise component with zero physical size, and flow resistance between the inlet and outlet are assumed negligible. To initiate the circuit simulation, a pressure (or flow velocity) is set at the inlet reservoirs, and the outlet reservoir at the end of the main output channel is grounded, both serving as the boundary conditions.

The CG of the sample is obtained by the analytical solution of the convection–diffusion equation with assumptions, and expressed in the form of the Fourier series coefficients

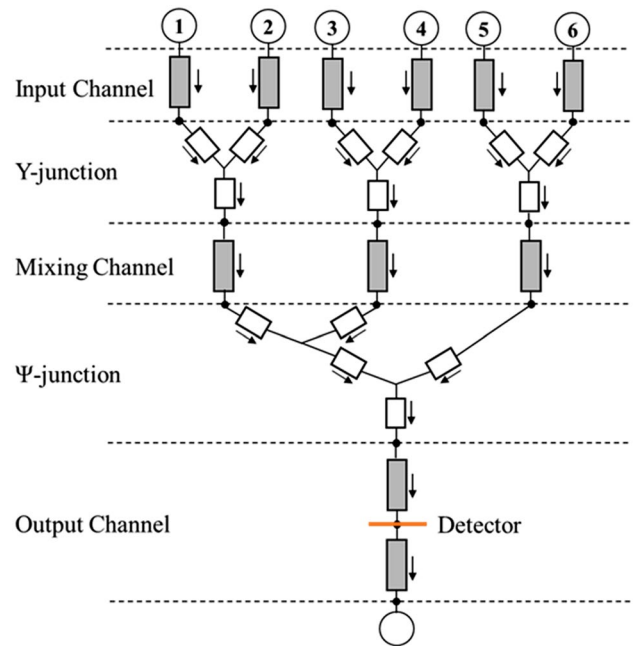


Fig. 3 Schematic of PBCM assembly to represent the triple-Y μ CGG

at the inlet(s) $d_n^{(\text{in})}$ and the outlet(s) $d_n^{(\text{out})}$ of a component, where n is the index of the Fourier series coefficient. In other words, the model is essential to establish an input–output relationship between the Fourier series coefficient $d_n^{(\text{in})}$ and $d_n^{(\text{out})}$. For instance, the relationship of a straight microchannel is as follows:

$$d_n^{(\text{out})} = d_n^{(\text{in})} e^{-(n\pi)^2 \tau} \text{ and } \tau = \frac{LD}{Uw^2}, \quad (2)$$

where D is the molecular diffusivity and U is the average flow velocity. In the combiner, two incoming streams at the inlets carrying different CGs are combined as a single stream exiting the outlet. The equation to relate the Fourier coefficients at its outlet $d_n^{(\text{out})}$ to those at its inlets $d_n^{(L)}$ and $d_n^{(R)}$ is given by

$$d_n^{(\text{out})} = \begin{cases} d_0^{(L)}s + d_0^{(R)}(1-s), & n = 0 \\ s \sum_{m=0}^{\infty, m \neq ns} d_m^{(L)} \frac{f_1 \sin(f_2) + f_2 \sin(f_1)}{f_1 f_2} + s \sum_{m=0}^{\infty, m=ns} d_m^{(L)} + (1-s) \sum_{m=0}^{\infty, m=n(1-s)} (-1)^{n-m} d_m^{(R)} \\ + 2(-1)^n(1-s) \sum_{m=0}^{\infty, m \neq n(1-s)} d_m^{(R)} \left(\frac{\cos(F_2/2) \sin(F_1/2)}{F_1} + \frac{\cos(F_1/2) \sin(F_2/2)}{F_2} \right), & n \geq 1 \end{cases}, \quad (3)$$

$$f_1 = (m - ns)\pi, f_2 = (m + ns)\pi \\ F_1 = (m + n - ns)\pi, F_2 = (m - n + ns)\pi$$

where L and R denote the inlet at the left and right, respectively; $s = q^{(L)} / (q^{(L)} + q^{(R)})$ is the flow ratio at the combiner junction, i.e., the location of the interface between the two incoming streams, and m is the index of the Fourier series coefficient at the inlets. For the reservoir containing samples that will be pumped into the μ CGG network, the Fourier coefficients are given by

$$d_n^{(out)} = \begin{cases} C_s, & n = 0 \\ 0, & n \geq 1 \end{cases}, \tag{4}$$

where C_s is the sample concentration specified at the reservoir.

All PBCMs above are implemented in MATLAB (MathWorks 2017) in a modular, parameterized manner, and can be linked to construct μ CGGs of various topology (Wang et al. 2006; Zhou et al. 2009), although the present effort focuses on the triple-Y μ CGG only. The assembled network model will be computed sequentially in two steps. First, the pressure distribution and the flow rate through each component within the entire network will be computed based on the Kirchhoff’s law given the boundary conditions, i.e., the pressure and/or flow rate specified at the inlet reservoir. Next, the Fourier coefficients of the sample concentration for each component in the network will be computed. It is undertaken along the flow direction determined in the previous step starting from the upstream-most inlet reservoir using specified sample concentrations, i.e., boundary conditions. The coefficients $\{d_n^{(out)}\}^j$ at the outlet of component j are computed using those at its inlet(s), and then assigned to those at the inlet of the component immediately downstream.

It should be noticed that first, PBCM above is developed in two spatial dimensions, i.e., the longitudinal and width direction of the channel, with an assumption that the channel depth is small relative to the width. Gorman and Wikswa (2008) indicated that the CG results based on 2D approximation are similar to those obtained by 3D simulation, especially when the aspect ratio β is small. Second, our PBCM was previously verified by both experiments (using microfabricated μ CGGs) and CFD simulation with very minor difference among them (Wang et al. 2006; Zhou et al. 2009), and can be considered trustworthy for DNN data generation. Finally, because of its closed-form nature, PBCM can be simulated at extremely fast speed in MATLAB, typically a fraction of a second on a regular PC, in distinct contrast to CFD that normally takes around 10–30 min for a single simulation. Thus, our PBCM can be utilized to produce a huge amount of data for DNN training, validation, and testing.

2.4 Deep neural network and inverse design verification

To comprehensively evaluate the proposed DNN-based inverse design method, three performance criteria are defined, including DNN testing error ϵ_t , design error relative to PBCM ϵ_{PBCM} and design error relative to CFD ϵ_{CFD} , which are defined by

$$\begin{aligned} \epsilon_t &= \frac{|CG_t - g_{PBCM}(f_{DNN}(CG_t))|}{CG_t} \\ \epsilon_{PBCM} &= \frac{|CG_p - g_{PBCM}(f_{DNN}(CG_p))|}{CG_p} \\ \epsilon_{CFD} &= \frac{|CG_p - g_{CFD}(f_{DNN}(CG_p))|}{CG_p}, \end{aligned} \tag{5}$$

where CG_t refers to the set-aside testing data, simply a portion of the LHS data generated above, CG_p is the prescribed CG that may not be physically achievable, e.g., a strictly smooth linear profile is not physically attainable through μ CGGs, and f_{DNN} is the function form of the trained DNN for inverse design, which predicts design parameters (inlet concentrations and pressures) based on CG_p . Finally, g_{PBCM} and g_{CFD} represent PBCM and CFD simulation, respectively, which takes the design parameters from f_{DNN} as inputs and outputs the CGs. It clearly shows that when evaluating ϵ_t , the CGs generated by our PBCMs, i.e., CG_t are used as the inputs to DNN-based design to obtain μ CGG design parameters. These design parameters are then entered to our PBCM again to predict corresponding CGs. The produced CGs and those as the input to DNN are compared quantitatively.

In contrast, the evaluation of ϵ_{PBCM} uses a CG_p as the input to DNN rather than the CG produced by our PBCM. The only difference between ϵ_{PBCM} and ϵ_{CFD} is that in the latter, the optimal parameters predicted by the DNN-based inverse design are used in CFD simulation, and the CG profile predicted by CFD is compared with the prescribed/target CG. The detailed procedure for calculating ϵ_{CFD} is illustrated in Fig. 4, and includes the following steps: (1) specify a prescribed/target CG and supply it to the trained DNN; (2) DNN predicts the design parameters of inlet concentrations and pressures; (3) design parameters are used to configure the CFD model; (4) CFD simulation predicts the CG at the detector given the design parameters; and (5) CFD-computed CG is compared with the prescribed/target CG. When the CFD simulation in the third step is replaced with the PBCM above, the procedure above produces ϵ_{PBCM} . When the CG produced by PBCM and PBCM simulation are, respectively, used in the first step and the third step in Fig. 4, the error evaluation reduces to ϵ_t . It should be pointed out that ϵ_t , ϵ_{PBCM} , and ϵ_{CFD} are used to inspect various

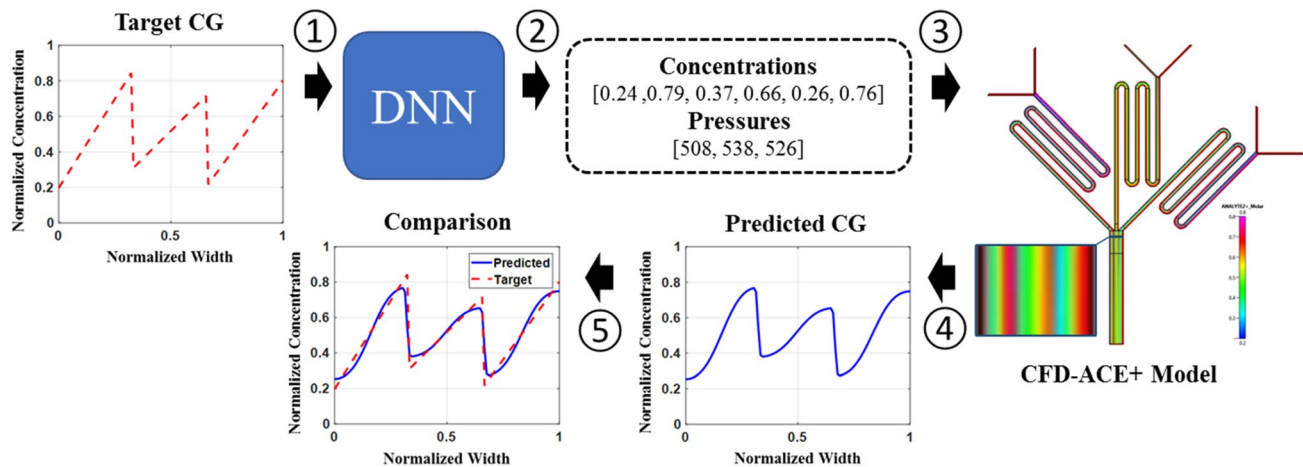


Fig. 4 Verification of the trained DNN and inverse design using high-fidelity CFD simulation

aspects of the entire DNN-based inverse design process. ε_t will only examine the DNN learning performance, such as accuracy and generality. $\varepsilon_{\text{PBCM}}$ quantifies the resemblance between the prescribed CG and the CG computed by PBCM using inverse design-predicted parameters. Again the prescribed/target CG may not be physically achievable through μCGGs . Therefore, $\varepsilon_{\text{PBCM}}$ characterizes not only learning performance above but also feasibility of generating the prescribed CG. More broadly, ε_{CFD} will also scrutinize the discrepancy between PBCM and high-fidelity CFD arising from the assumptions used in PBCM, and therefore, CFD simulation is employed to produce CGs using design parameters provided by DNN for verification. Since our DNN is trained using PBCM-generated data, it is expected that ε_{CFD} is worse than $\varepsilon_{\text{PBCM}}$ given the same prescribed/target CG.

3 Case studies and DNN structures

Two different case studies are performed to verify and demonstrate DNN-based inverse design of the triple-Y μCGG . In the first case study, various CGs are generated by tuning only inlet concentrations of the sample while equal and constant pressures are applied at all the inlets. Therefore, all the three constituent CGs from Y-shaped mixers have the same width in the combined one. The constant pressure at all the inlets is set to 300 Pa. In the second case study, as discussed above the pressures are organized into three pairs with different values, each for one Y-shaped mixer, resulting in different widths of the constituent CGs in the combined one. This scenario adds a critical challenge to DNN training and inverse design since many combinations of design parameters can yield the same CG, i.e., many-to-one mapping. In other words, the design problem becomes non-unique, and

as a result, DNNs of more sophisticated architectures are needed in contrast to the first case study.

3.1 Case study 1: inverse design of inlet concentrations

In the first case study, two types of DNNs are developed, which are fully connected neural network (FCNN) and convolutional neural network (CNN). In total, 200,000 instances for various combinations of design parameters are simulated by PBCM, among which 16,000 and 4000 are used for validation and testing, respectively, and the rest are fully utilized for training. FCNN is the most common neural network, it is relatively straightforward to construct as there are less hyperparameters of architecture to select. The number of hidden layers and the neurons for each hidden layer are the only choices that need to be made for FCNN construction. In the present effort, 4 hidden layers with neuron numbers of [256, 128, 64, 24] are chosen empirically. On the other hand, CNN is more laborious to construct because of more hyperparameters, such as the number and size of the convolution and pooling layers needed to be determined. Selecting hyperparameters to establish an optimal DNN architecture is a general challenge in DNN development. Although the meta-optimization technique (Curteanu et al. 2011; Zhang et al. 2016; Hong et al. 2019) can automatically narrow down the choice of hyperparameters, it involves extremely demanding computation and training of a large number of DNN instances, in particular, when the evolutionary algorithms are used, which is out of the scope of the present study. Therefore, CNN hyperparameters and architecture are also selected empirically. As shown in Fig. 5, CNN used in this work is comprised of three convolution and pooling layers

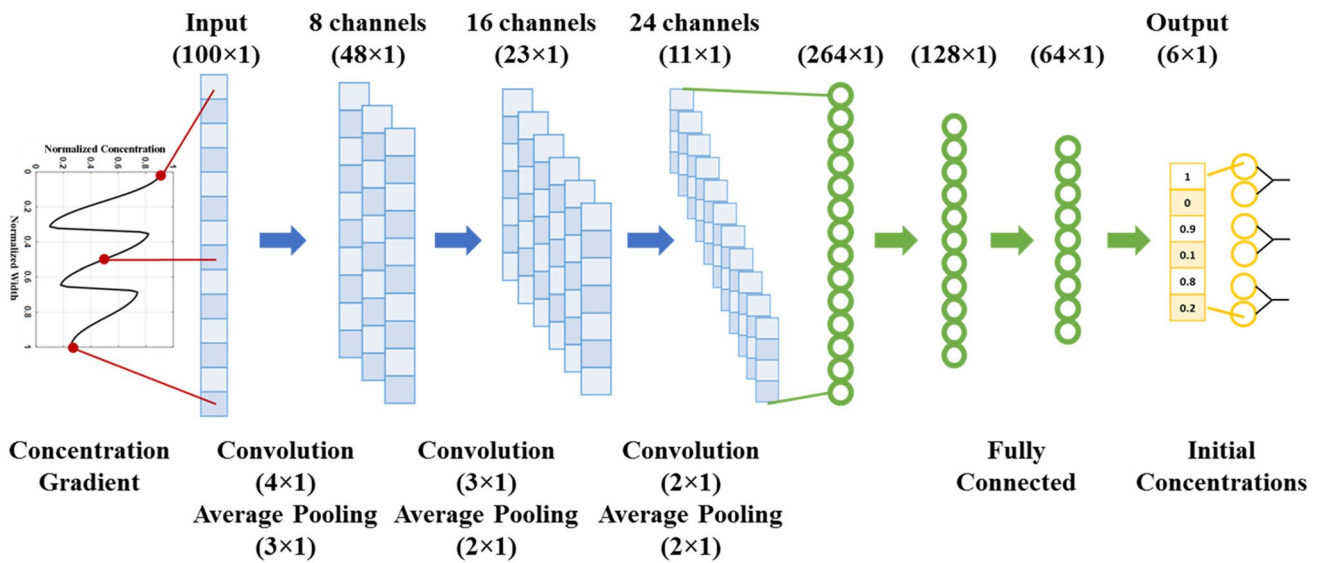


Fig. 5 Schematic of the convolutional neural network (CNN)

Table 1 Comparison between FCNN and CNN in terms of the number of trainable weights, number of layers, and testing accuracy

	Number of trainable weights	Number of layers	Testing error (ϵ_t)		
			Min (%)	Mean (%)	STD (%)
FCNN	68,718	6	0.51	0.67	0.11
CNN	43,798	7	0.54	0.67	0.13

followed by two fully connected hidden layers. The input vector is one-dimensional and represents the discretized CGs along the width direction, and convolution and pooling operations of various sizes are performed on the one-dimensional vector as shown in the figure. Moreover, no padding is applied for all layers and the stride size for the convolution and average pooling are 1 and 2, respectively. For both FCNN and CNN training, the Adam optimizer is applied with a learning rate of $1e-3$ with the hard tangent activation function to the output layer followed by rescaling, and the computed concentration gradients are bounded between 0 and 1.

The comparison between FCNN and CNN in terms of the number of trainable weights, layer size, and testing error (ϵ_t) is shown in Table 1. The advantage of using CNN in this case study is clear, i.e., with a smaller number of parameters more layers can be constructed to refine feature filters. Usually, for problems with highly complex input–output mapping relationship, a large number of layers in the network structure is necessary. Both DNNs exhibit testing accuracy greater than 99% ($\epsilon_t < 1\%$) which ensures that the DNN structure selected is effective to capture the design rules hidden in the data. It is important to

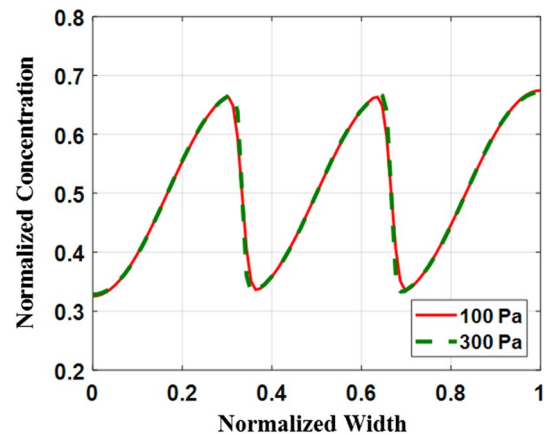


Fig. 6 Nearly identical concentration gradients with different pressures

note that there may be uncertainties in ANN training due to random initial weights. In other words, given different initial weights, ANN may not converge to the same solution. To address this issue, ANN training is repeated multiple times (10 times for this work) and the most accurate ANN is selected. In the table, we can observe minimum, mean and standard deviation (STD) of testing errors (ϵ_t) for both DNNs. It has been found that the uncertainties for these two DNNs are very small since both STDs are close to 0.1%. This ensures that the DNN structure, training method, and hyperparameters that we currently use are robust for this particular problem.

3.2 Case study 2: inverse design of inlet concentrations and pressures

Adding inlet pressures of the triple-Y μ CGG into inverse design significantly increases the difficulty in DNN formulation and prediction, which can be attributed to two facts. First, as discussed above the optimal solution becomes non-unique, i.e., many design parameters lead to the same CG (many-to-one). As displayed in Fig. 6, two almost identical CGs are generated with different inlet pressures. This is because even if the pressures are different, inlet concentrations can be tuned to produce nearly same CGs. This implies that given their continuous variations, a large number of inlet pressure and inlet concentration combinations can be found to generate a desired CG, which renders it difficult to formulate and train an accurate and reproducible DNN. The second challenge is associated with the issue of backflow. When the pressures applied to the Y-shaped mixers are varied, it also changes the flow rates through them and relative widths of the constituent CGs in the combined one. For instance, if the flow rate of the left mixer is larger than the other two, then the constituent CG formed in the former will occupy a larger portion along the width of the main output channel. Although allowing to generate CGs of complex

and asymmetric shapes, it may cause backflow of the fluid from the left mixer to the other two when their pressure difference becomes overly large. That is, the pressure at the Ψ -junction is higher than the inlets of the two mixers in the middle and on the right. A strategy needs to be developed in DNN formulation to avoid backflow occurrence first. Therefore, estimating inlet concentrations and pressures is handled separately in this case study.

We propose a “divide-and-conquer” strategy and a new architecture with a cascade connection of three FCNNs in a stair-like arrangement as shown in Fig. 7. It breaks down inverse design into several steps to enable sequential determination of the pressure and concentration parameters by individual FCNNs, although they are trained together as a whole. The hidden layer structures of each FCNNs are identical to the FCNN introduced in a previous section, while the sizes of the input and output layer are different.

First of all, to reduce complexity, the inlet pressure is limited to one of the six values [100, 200, 300, 500, 1000, 3000] (Pa), which is selected under the guidance of PBCM simulation, and essentially switches the problem of pressure determination from regression to classification, and hence, alleviating the many-to-one issue. It is important to note that even with such a restriction, the variety of CGs that can be

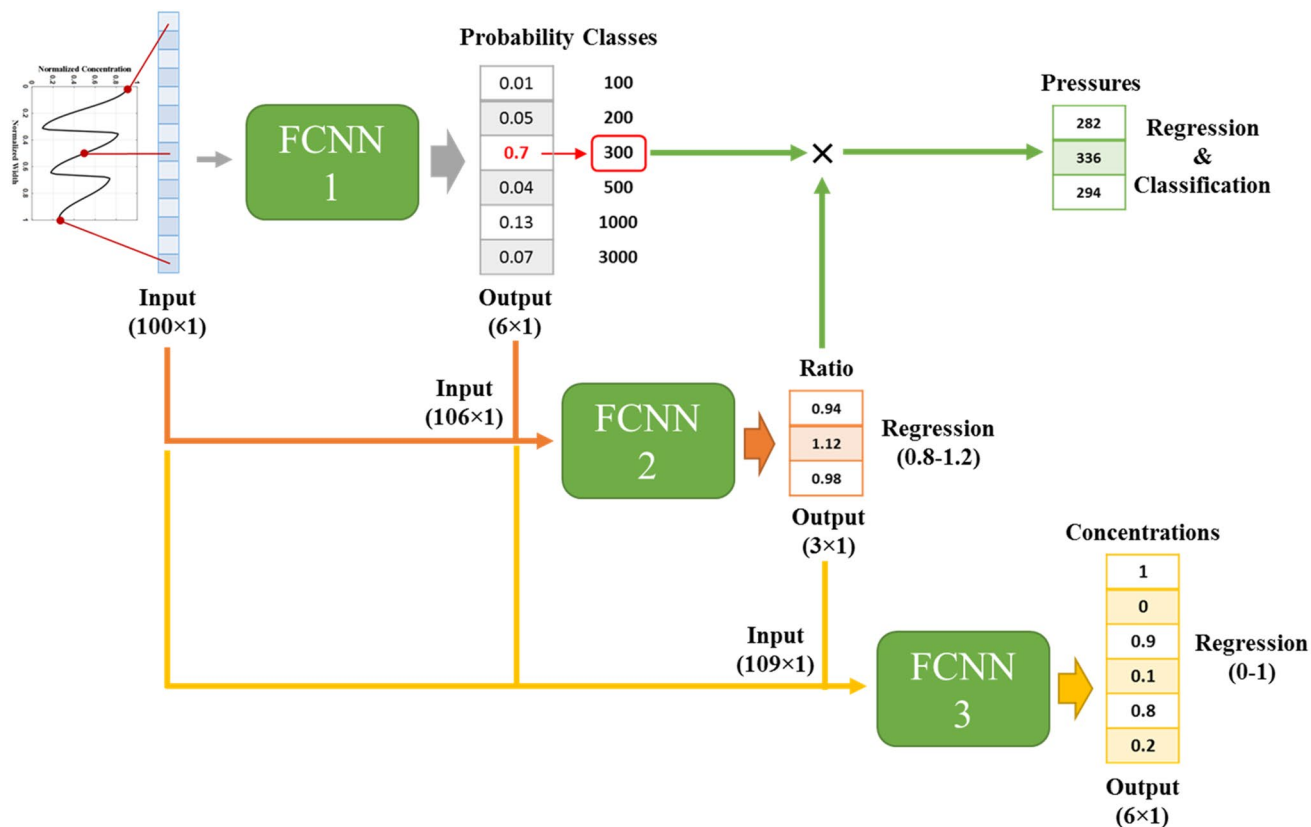


Fig. 7 Schematic of the cascade FCNN

achieved by tuning the inlet concentrations will remain the same. Given a target CG, the first FCNN assigns one of the six values to the inlet pressure according to the computed probabilities of all classes, and the value serves as a baseline pressure value for the three Y-shaped mixers.

Next, the inlet pressure applied to each Y-shaped mixer will be determined individually by tuning the baseline value, which allows to change the relative width of each constituent CG in the combined one without causing backflow. This is accomplished by the second FCNN that computes three ratios in the range of [0.8, 1.2]. The ratio is defined by the inlet pressure of each Y-shaped mixer to the baseline value predicted by the first FCNN. In other words, the baseline pressure value multiplied by the three ratios yields three inlet pressures applied to the Y-shaped mixers. The use of a narrow range of the ratio effectively precludes the occurrence of backflow phenomenon. For enhanced accuracy, the probabilities computed from the first FCNN are supplied to the second FCNN along with the target/prescribed CG. Consequently, the range of the output pressures will be computed in the piecewise manner [80–120, 160–240, 240–360, 400–600, 800–1200, 2400–3600] (Pa).

Finally, the probabilities and the ratio computed above are supplied to the third FCNN, in conjunction with the target/prescribed CG to predict inlet concentrations. Two points need to be noted: (1) the error introduced by pressure estimation in the first FCNN can be compensated by refining the inlet concentration. Therefore, it is rational to supply the ratio of the inlet pressure to inlet concentration prediction; and (2) once the pressure is determined by the two preceding FCNNs, the many-to-one (non-unique) issue is tackled, and the third FCNN is able to determine the inlet concentrations more reliably. In short, our analysis indicates that by systematically dividing the tasks into a series of sub-tasks, and directing information to each individual FCNN in the given order, DNN accuracy and design performance can be substantially improved compared to a single DNN that predicts inlet pressures and concentrations simultaneously.

Since predicting inlet pressures and concentrations exhibit more complex input–output mapping relationship for DNN learning, 9 million instances of PBCM simulation are generated, which is 45 times more compared to the previous case study. There are six classes of baseline pressure values, and for each pressure, 1.5 million instances are created using LHS that randomly selects three ratios and six inlet concentrations. Among the entire data set, 85,500 and 4500 instances of the data set are utilized for validation and testing. Similar to the previous case study, DNN training has been repeated multiple times. The minimum and the average test error ϵ_t , of the proposed cascade FCNN is found to be 0.74% and 1.00%, respectively. Again, STD of the test errors is found to be small with the

actual value of 0.18%, confirming the robustness of our DNN training.

4 Results and discussion

In this section, the results of the case studies using FCNN, CNN, and cascade FCNN will be presented, and the errors of the CG produced using inverse design parameters relative to the target/prescribed CG are also quantitated.

4.1 Case study 1: inverse design of inlet concentrations

First, we present the results of case study 1, in which only inlet concentrations of the three Y-shaped mixers are predicted while holding all inlet pressures the same, viz., 300 Pa. The trained DNNs, including FCNN and CNN, are verified with a set of prescribed CGs under different categories, including (a) sawtooth, (b) linear, and (c) exponential shapes that are not available in the training data due to the physical limitation of the tripe Y-shaped μ CGGs (detailed description is given below). Among them, three specific ones are selected for presentation in Figs. 8 and 9, which show the results obtained by FCNN- and CNN-based inverse design. The left column illustrates the comparison between the predicted CGs and the prescribed CGs. The predicted CGs include one computed by PBCM and the other by CFD that are, respectively, denoted by the green markers and the solid blue line. Briefly, given the target/prescribed CG, the trained DNN determines the inlet concentration parameters, which are then entered to both PBCM and CFD simulation, and the CGs predicted by both models are then compared against the target CG. The details for inverse design verification are elucidated in Sect. 2.3. The right column shows the contour plot of the normalized concentration at the detector in CFD simulation. Figures 8 and 9 show that inlet concentration parameters predicted by both FCNN- and CNN-based inverse design are able to generate almost identical CGs, and resemble well all three prescribed CGs. The discrepancy between PBCM- and the CFD-predicted CG is almost negligible.

It should be pointed out that due to the physical limitation, the predicted CGs cannot exactly match the prescribed/target CGs. Bending of CGs at both sides in all plots is attributed to the impermeability of channel walls (Wang et al. 2006), and may also occur in the complete mixing-based CGGs. For the same reason, two small bumps are also observed at the interfaces between the constituent CGs, i.e., at the 1/3 and 2/3 of the normalized width (Fig. 8b, c), which are caused by impermeability of the wall of the upstream mixing channel.

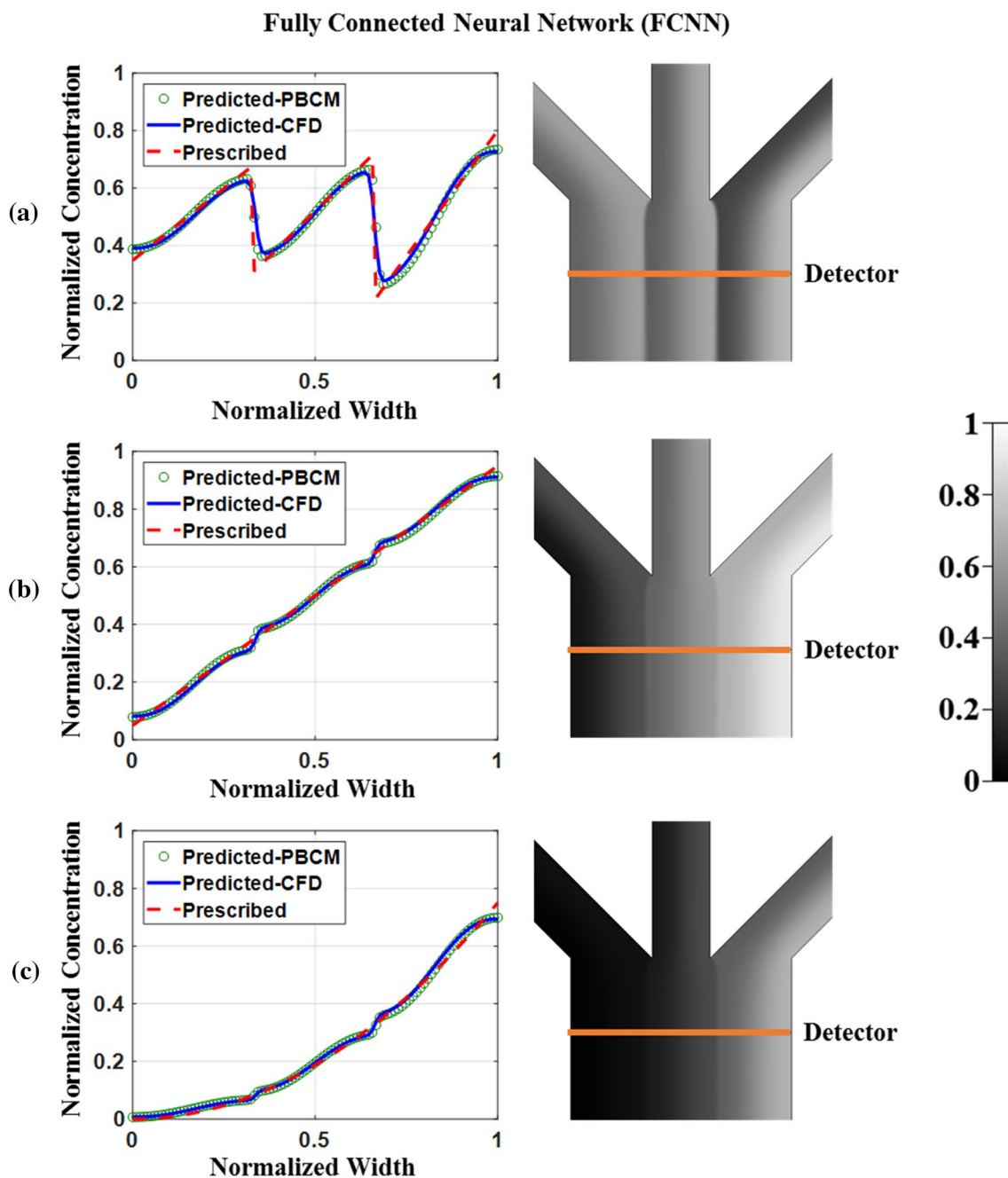


Fig. 8 Predicted CGs using FCNN-predicted design parameters for same constituent widths. **a** Sawtooth CG, **b** linear CG, and **c** exponential CG. Left column: the comparison between the prescribed/tar-

get CG, PBCM-predicted CG, and CFD-predicted CG. Right column: concentration contour plot simulated by CFD

For more elaborate analysis, the percent errors of CGs computed by PBCM and CFD, i.e., ϵ_{PBCM} and ϵ_{CFD} using the inlet concentrations predicted by FCNN and CNN relative to the prescribed CGs are shown in Table 2. It shows that CGs generated using FCNN- and CNN-predicted design parameters agree with the prescribed CG quantitatively, and both exhibit similar percent errors. The percent errors of

CFD-computed CGs (ϵ_{CFD}) are slightly worse than PBCM (ϵ_{PBCM}) despite the minor difference. As explained above, this is because the inverse design rules learned by FCNN and CNN are based on the data generated by PBCM rather than CFD, and therefore, the CGs computed by PBCM is closer to the prescribed CG. On the other hand, the minor difference between ϵ_{PBCM} and ϵ_{CFD} ($<0.6\%$) also confirms

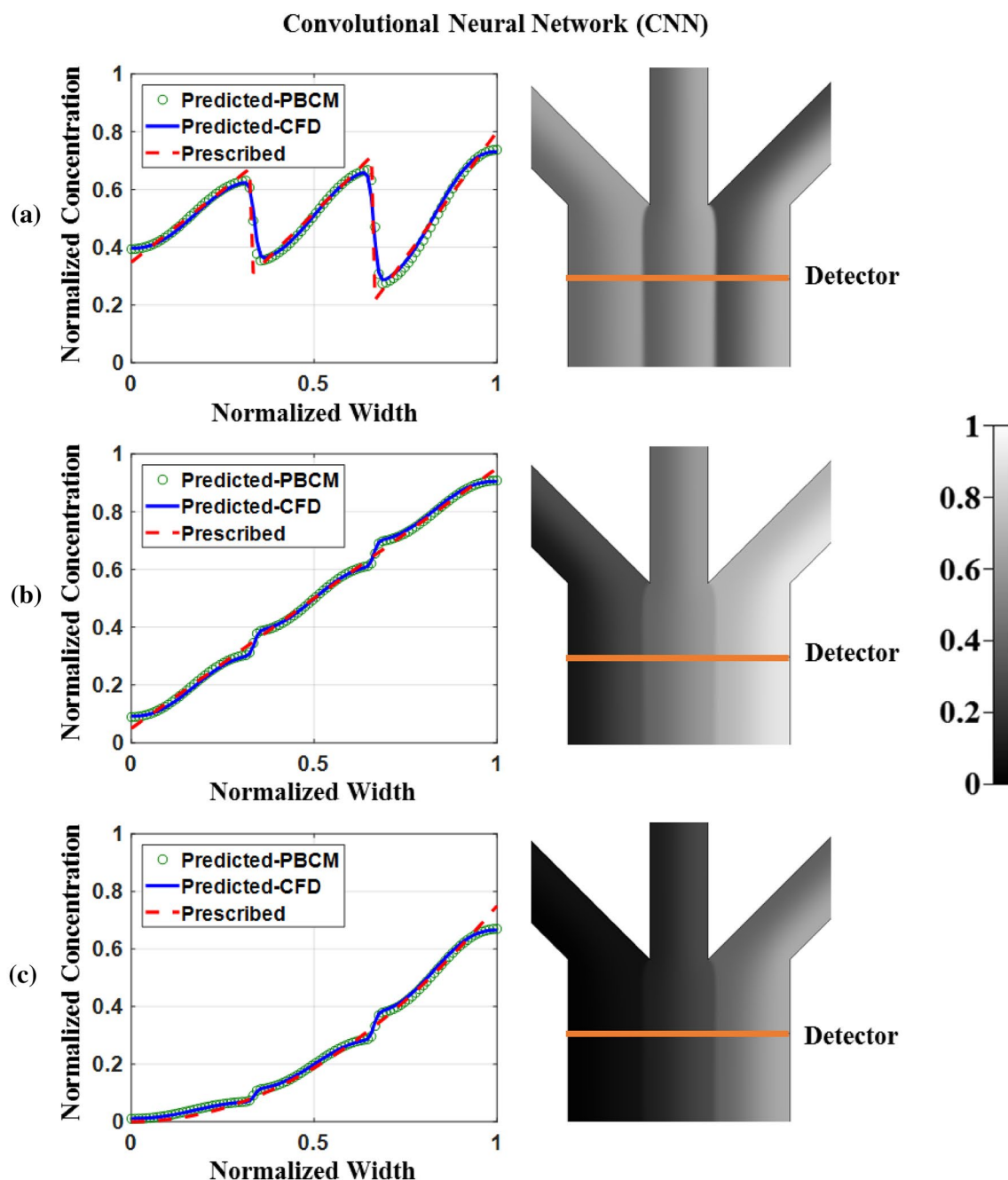


Fig. 9 Predicted CGs using CNN-evaluated design parameters for same constituent widths. **a** Sawtooth CG, **b** linear CG, and **c** exponential CG. Left column: the comparison between the prescribed/target CG, PBCM-predicted CG, and CFD-predicted CG. Right column: concentration contour plot simulated by CFD

the accuracy of PBCM in this case study where the stream width from each Y-shaped mixer is the same.

4.2 Case study 2: inverse design of inlet concentrations and pressures

In this case study, the use of DNN to design more complex sawtooth shapes containing constituent linear segments of

different widths and orientations are investigated. In addition to the single FCNN above, the new cascade FCNN is also interrogated. The CGs computed by PBCM and CFD using design parameters predicted by FCNN and cascade FCNN are shown in Figs. 10 and 11, respectively, in which the left column compares CGs predicted by PBCM (labeled by green markers) and CFD (blue line) against the prescribed/target CG (red dashed line), and the right column

Table 2 Predicted inlet concentrations and CG percentage errors of FCNN and CNN

Shapes	DNN	Concentrations						Error (%)	
		1	2	3	4	5	6	ϵ_{PBCM}	ϵ_{CFD}
Sawtooth (a)	FCNN	0.3597	0.6626	0.3260	0.7019	0.2062	0.7868	4.0	4.6
	CNN	0.3669	0.6602	0.3139	0.7072	0.2165	0.7895	3.9	4.4
Linear (b)	FCNN	0.0525	0.3401	0.3579	0.6386	0.6539	0.9405	1.8	1.9
	CNN	0.0649	0.3293	0.3598	0.6386	0.6733	0.9316	1.8	2.4
Exponential (c)	FCNN	0	0.0730	0.0746	0.3163	0.3194	0.7366	4.4	4.3
	CNN	0.0036	0.0767	0.0923	0.3076	0.3444	0.7023	5.2	5.7

depicts the contour plot of the normalized concentration at the Ψ -junction merging all Y-shaped mixers, which also includes the detector location.

First, we examine design performance of the single FCNN, which is similar to the one in the previous case study with the only difference that this one predicts both the inlet pressures and the inlet concentrations, while the previous one only predicts inlet concentrations while holding the pressure applied to each Y-shaped mixer the same. From Fig. 10, we can clearly see that although exhibiting the trend of the profile, the predicted CGs do not match the prescribed/target CGs, which as discussed above is attributed to the difficulty of single FCNN to address the many-to-one (non-unique) issue of inverse design when it is applied to complex CGs in higher dimensional design space. In contrast, it is expected that cascade FCNN has better capability of predicting both the inlet pressures and concentrations for each Y-shaped mixer and generating a complex CG including constituent linear profiles of different widths. It is evident in all the three CGs shown in Fig. 11 that by adding new design parameters, diversity and flexibility of CG generation is enhanced compared to the previous study. Next, CGs computed by PBCM and CFD using the nine design parameters estimated by cascade FCNN, in general, match very well the prescribed/target CGs of various shapes except at the stream interface and side walls due to the mass impermeability of all the channels.

The cascade FCNN-predicted inlet pressures and concentrations, and the percent errors of the produced CGs relative to the prescribed/target CGs are all presented in Table 3. The average percent error ϵ_{CFD} of FCNN and cascade FCNN are also calculated, which are 15.0% and 8.2%, respectively. Therefore, by breaking down the nine-parameter inverse design into sequential steps through cascade FCNN, the non-unique issue can be effectively settled and more complex CGs can be generated as desired.

It is also interesting to note that in Table 3 although minor, the differences in the percent error between PBCM- and CFD-computed CGs, i.e., ϵ_{PBCM} vs. ϵ_{CFD} is more appreciable than the first case study. This is because when the detector is located in the developing flow region after the merging Ψ -junction, the unequal flow rates from the three Y-shaped mixers cause noticeable transverse flow along the cross section, which,

however, is less critical when all flow rates are the same. This can be confirmed by observing the variation of the stream interface. For example, the contour line corresponding to the stream interface in the concentration plot in Figs. 10 and 11 (unequal flow rate) expands more dramatically than those in Figs. 8 and 9 (equal flow rate). Since PBCM is the point-wise model assuming transversely uniform flow within the channel for analytical solution, the species transport under developing flow is not taken into account, leading to larger discrepancy between PBCM and CFD-computed CGs. However, considering the extremely fast simulation speed of PBCM relative to CFD (at least three orders of magnitude), the former is adopted to generate a large volume of data for DNN training that however cannot be accomplished through CFD.

Although cascade FCNN can generate complex CGs with different constituent widths, it would be desirable if it can also be used to design μCGGs for simple shapes as well. Next, we will investigate the backward compatibility of cascade FCNN to the low-dimensional inverse design problem, i.e., downgrading its use from the nine-dimensional to six-dimensional (inlet concentrations only) problem, while it is originally trained within nine-dimensional space including both the inlet pressures and the inlet concentrations. In other words, given a target CG comprised of constituent profiles with the same width, is it possible to use cascade FCNN to produce equal inlet pressures along with appropriately selected inlet concentrations. The results of cascade FCNN and corresponding CG prediction in the reduced dimension are depicted in Fig. 12 and Table 4. The left column in Fig. 12 compares the CGs predicted by PBCM (labeled by green markers) and CFD (blue line) against the prescribed/target CGs (red dashed line) that include the sawtooth, linear, and exponential profiles.

In Fig. 12a, it is clear to show that cascade FCNN produces sawtooth CG that resembles the prescribed/target CG, similar to CG produced by FCNN in Fig. 8a. The corresponding table values for the three predicted pressures, 502, 523 and 515 Pa, are almost identical. However, for the linear and exponential CGs shown in Fig. 12b, c, respectively, cascade FCNN exhibits poor design performance indicated by appreciable discrepancy between the CFD-computed CGs and the prescribed CGs. The error ϵ_{CFD} up to 13% is observed in Table 4, which may arise

Fully Connected Neural Network (FCNN)

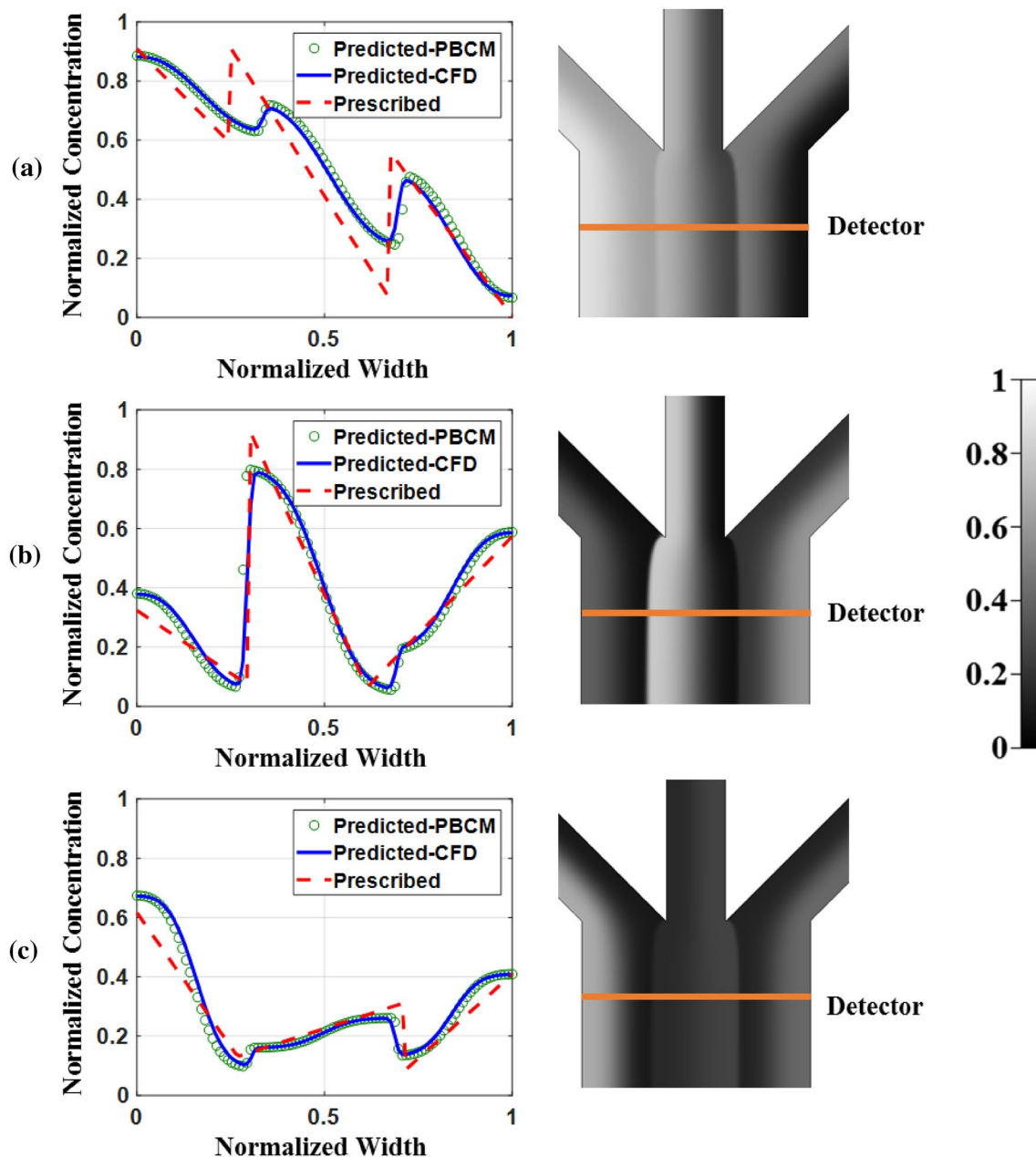


Fig. 10 Predicted CGs using FCNN-evaluated design parameters for different constituent widths. **a–c** Complex sawtooth CGs. Left column: the comparison between the prescribed/target CG, PBCM-

predicted CG, and CFD-predicted CG. Right column: concentration contour plot simulated by CFD

from several factors. First, the training data may not have adequate instances corresponding to linear or exponential CGs. Cascade FCNN is trained with the LHS data sampled in the nine-dimensional parameter space. Therefore, there is a low probability to have three equal inlet pressures in the data as they are treated as separate dimensions for sampling. However, the training data of single FCNN is LHS sampled in six-dimensional parameter space for inlet

concentrations only and is inherently assigned equal inlet pressures. Clearly, the accuracy of the predictions will be compromised if relevant data for training are scarce. Second, the many-to-one issue may be exacerbated for linear and exponential shapes because of their simple shapes. This is intuitively comprehensible since it is easy to narrow down the choice of inlet pressures for the sawtooth CG because its shape provides richer information

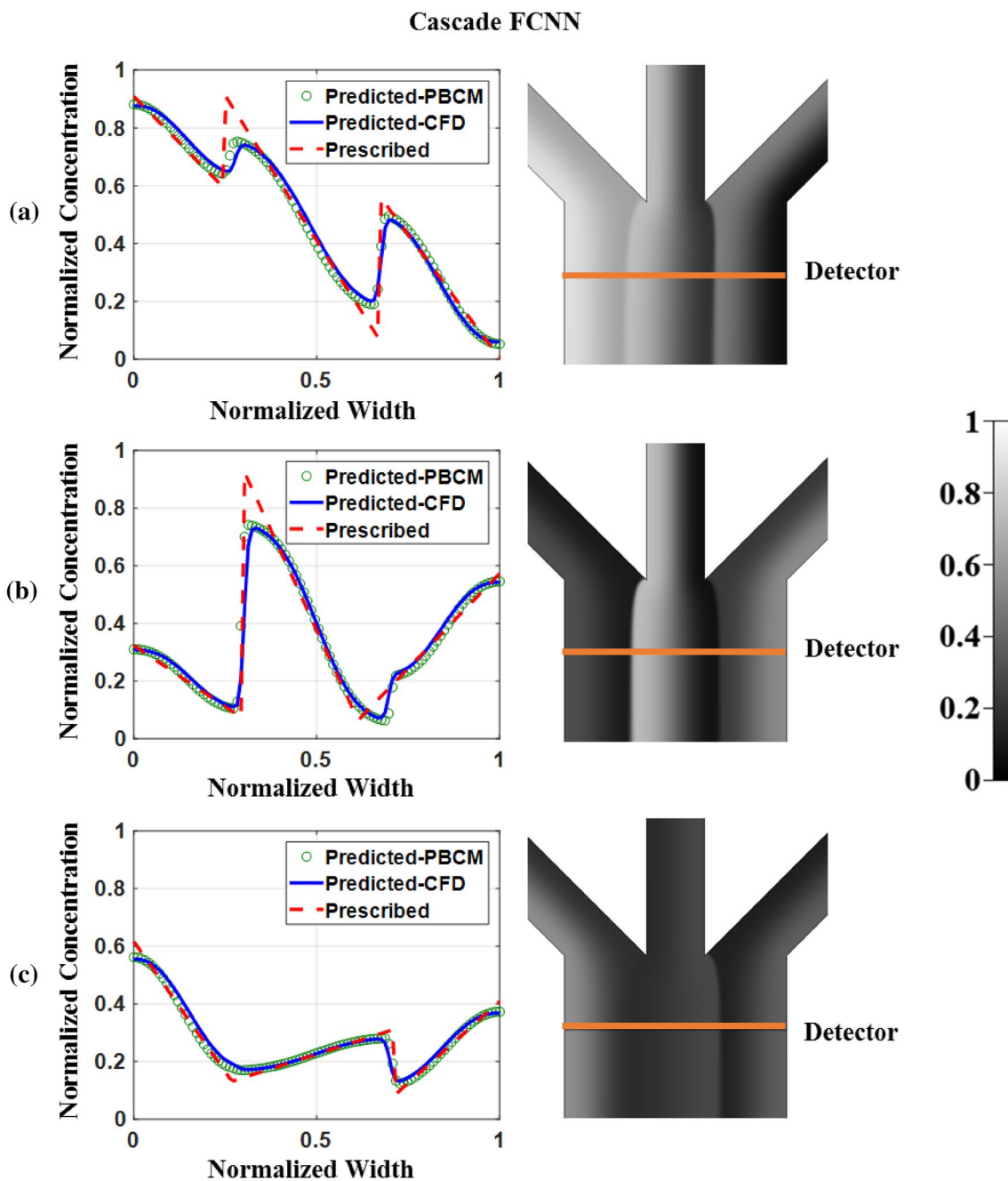


Fig. 11 Predicted CGs using cascade FCNN-evaluated design parameters for different constituent widths. **a–c** Complex sawtooth CGs. Left column: the comparison between the prescribed/target CG,

PBCM-predicted CG, and CFD-predicted CG. Right column: concentration contour plot simulated by CFD

for determining design parameters. However, to generate the linear and exponential profiles, there are more possible combinations of inlet pressures and concentrations. As a result, although intended to mitigate the non-unique issue, cascade FCNN trained in the nine-dimensional parameter

space cannot uniquely determine the simple linear and exponential CGs (with the same constituent widths) in six-dimensional design space. One potential workaround to be investigated in the future is to add more data representative of simple CGs separately prior to LHS sampling.

Table 3 Predicted inlet concentrations and pressures, and percentage errors for complex CGs

Shapes	DNN	Pressures (Pa)			Concentrations						Error (%)	
		1–2	3–4	5–6	1	2	3	4	5	6	ϵ_{PBCM}	ϵ_{CFD}
(a)	FCNN	298	324	264	0.91	0.60	0.77	0.20	0.54	0	16.5	14.8
	Cascade	240	360	292	0.93	0.59	0.80	0.15	0.55	0	5.8	7.9
(b)	FCNN	476	668	511	0.39	0.05	0.81	0.05	0.18	0.60	14.5	13.2
	Cascade	411	553	415	0.32	0.09	0.76	0.05	0.20	0.56	8.5	9.5
(c)	FCNN	751	971	774	0.68	0.09	0.16	0.26	0.13	0.41	15.3	16.9
	Cascade	244	360	256	0.63	0.11	0.16	0.29	0.08	0.41	4.5	7.3

5 Conclusions

In this paper, a new framework based on deep neural network (DNN) is developed for the inverse design of microfluidic concentration gradient generator (μCGG) of complex network topology. It takes CGs as input and outputs optimal design parameters, and hence, operating in an opposite manner relative to forward simulation used in the traditional design process. DNN is trained using simulation data with many parameter combinations generated before the design process to learn the design rules. The new method includes several key elements: (1) experimentally verified, fast-running physics-based component model (PBCM) in the closed form to capture convection–diffusion species transport in a variety of μCGGs that cannot be modeled by the electric circuit analogy technique and to produce a large amount of data efficiently for DNN learning; (2) the “divide-and-conquer” strategy and hybrid formulation of classification and regression within a single DNN platform to mitigate many-to-one (non-unique) design complications; and (3) the learned DNN model allows rapid, distributable inverse design. Several DNN network structures and formulation are developed, including single fully connected neural network (FCNN), convolutional neural network (CNN), and a new cascade FCNN. The cascade FCNN divides the design process into several subtasks to determine design parameters of various types in tandem, offering a salient solution to tackle the non-unique design issue.

Case studies are performed on a species transport-dominated triple Y-shaped μCGG to evaluate design performance of the proposed method. First, single FCNN and CNN are examined in a six-dimensional space that only includes sample concentrations at inlet reservoirs as the design parameters. It is verified in high-fidelity CFD simulation that the widely used CGs, such as sawtooth, linear, and exponential can be produced using FCNN- and CNN-predicted design parameters, which exhibit excellent agreement with the prescribed CGs with the average relative error $< 4\%$. Second, cascade FCNN is examined in nine-dimensional design space including both the inlet

concentrations and inlet pressures and the latter introduces significant complexities not only due to higher dimensionality but also the many-to-one (non-unique) complications. The cascade FCNN is able to effectively mitigate these issues, yielding enhanced design for CGs (average error $< 8\%$ relative to prescribed CGs).

This work represents an initial effort to develop DNN-based inverse design method/tool for a specific microfluidic application, i.e., μCGGs , although we expect it potentially could be extended to others. There are several interesting findings. First, a massive amount of data (e.g., up to 9 million) instances is used in this work to train the network of deep structures for inverse design, in particular, for high-dimensional space with parameters of different types. Existing high-fidelity simulation (CFD, FEM, and others) and experimental tests are too costly to produce the data at this level and, therefore, models with simplification/assumption could potentially fill the gap for DNN learning at the cost of reasonable accuracy loss. Second, it may be problematic to downgrade the use of cascade FCNN trained in a high-dimensional parameter space to a low-dimensional one due to different data sampling and representation. In this work, cascade FCNN exhibits superior performance for the design of complex CGs involving different inlet pressures, but not the one with equal inlet pressures, which, however, can be better predicted by single FCNN or CNN. Third, the DNN model is trained on a large volume of data to improve the generality of inverse design for various prescribed CGs. The data generated before the design process are generally relevant to but not exactly the same as that for a specific, prescribed CG. Therefore, it is anticipated that inverse design cannot exceed the performance of the traditional optimization method that aims to meet the “specificity” of each prescribed CG. The key advantage of inverse design is that once learned, the design rules are packaged into the DNN model and the design can be obtained in a similar manner as database query, which will significantly accelerate the design process, and eliminate the need for pre-knowledge of simulation and design, both facilitating distribution of the consolidated design capabilities to microfluidic end

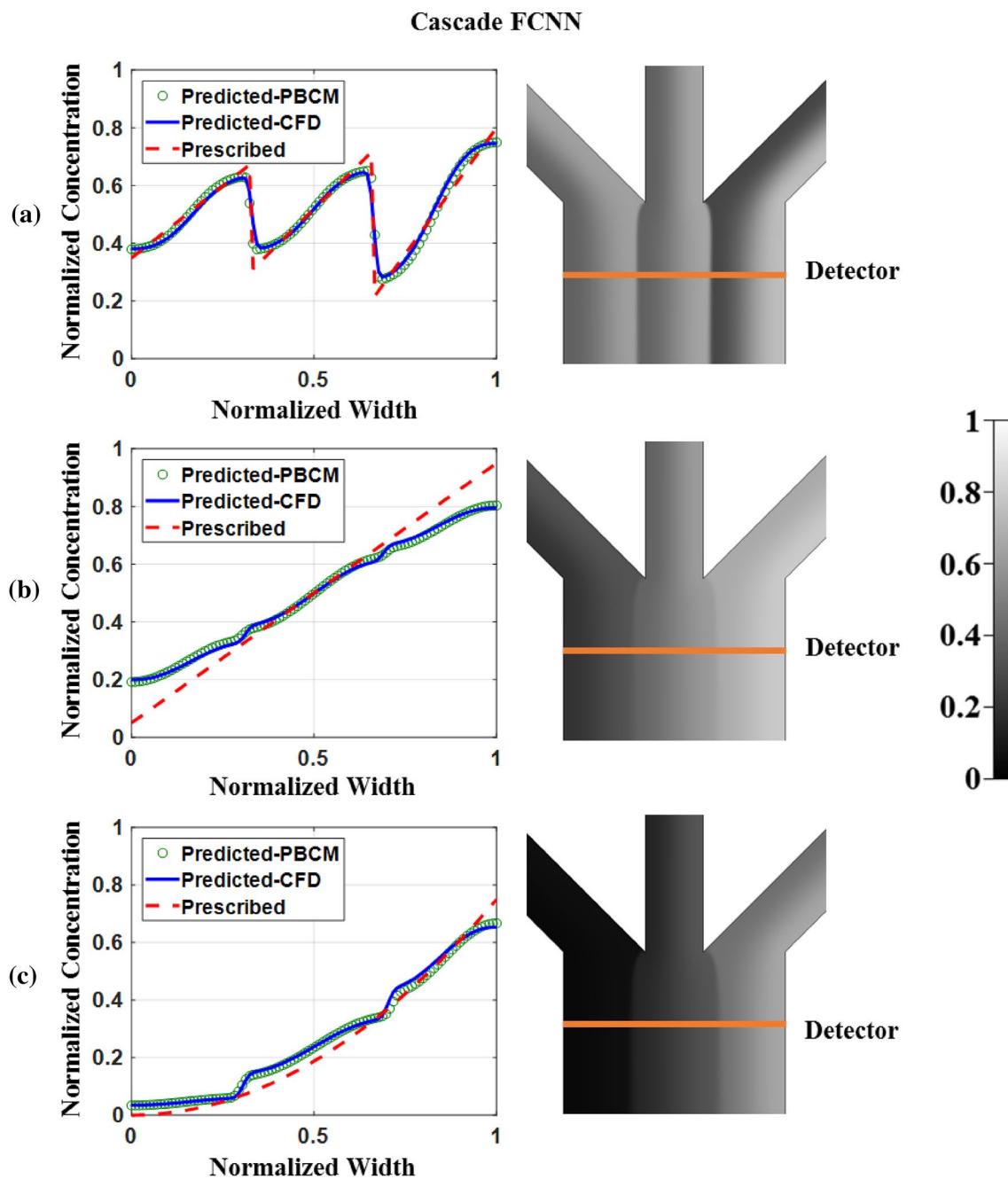


Fig. 12 Predicted CGs using cascade FCNN-evaluated design parameters for different constituent widths. **a** Sawtooth CG, **b** linear CG, and **c** exponential CG. Left column: the comparison between the

prescribed/target CG, PBCM-predicted CG, and CFD-predicted CG. Right column: concentration contour plot simulated by CFD

Table 4 Predicted inlet pressures and concentrations of cascade FCNN for simple CGs

Shapes	Pressures (Pa)			Concentrations						Error (%)	
	1–2	3–4	5–6	1	2	3	4	5	6	ϵ_{PBCM}	ϵ_{CFD}
(a) Sawtooth	502	523	515	0.37	0.64	0.37	0.66	0.26	0.76	4.4	4.8
(b) Linear	85	106	84	0	0.53	0.18	0.82	0.46	1.00	10.8	13.1
(c) Exponential	87	116	84	0	0.09	0	0.48	0.09	1.00	11.3	12.9

users. The future work will follow along two directions: improvement of data diversity to increase backward-compatibility of DNN models trained in high-dimensional design space, DNN hyperparameter meta-optimization to enhance model accuracy and extension of the method to other microfluidic applications.

Acknowledgements The authors acknowledge the faculty startup grant from the University of South Carolina for partial funding of this research.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

- An S, Fowler C, Zheng B, Shalaginov MY, Tang H, Li H, Richardson KA (2019) A novel modeling approach for all-dielectric metasurfaces using deep neural networks. arXiv preprint: arXiv:1906.03387
- Asproulis N, Drikakis D (2009) Nanoscale materials modelling using neural networks. *J Comput Theor Nanosci* 6(3):514–518
- Asproulis N, Drikakis D (2013) An artificial neural network-based multiscale method for hybrid atomistic-continuum simulations. *Microfluid Nanofluid* 15(4):559–574
- Chen X, Hu Z, Zhang L, Yao Z, Chen X, Zheng Y, Song H (2018) Numerical and experimental study on a microfluidic concentration gradient generator for arbitrary approximate linear and quadratic concentration curve output. *Int J Chem React Eng* 16(1)
- Curteanu S, Cartwright H (2011) Neural networks applied in chemistry. I. Determination of the optimal topology of multilayer perceptron neural networks. *J Chemom* 25(10):527–549
- Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6(2):182–197
- Dertinger SK, Chiu DT, Jeon NL, Whitesides GM (2001) Generation of gradients having complex shapes using microfluidic networks. *Anal Chem* 73(6):1240–1246
- Drikakis D, Frank M (2015) Advances and challenges in computational research of micro- and nanoflows. *Microfluid Nanofluid* 19(5):1019–1033
- Friedrich D, Please CP, Melvin T (2012) Design of novel microfluidic concentration gradient generators suitable for linear and exponential concentration ranges. *Chem Eng J* 193:296–303
- Gao Y, Sun J, Lin WH, Webb DJ, Li D (2012) A compact microfluidic gradient generator using passive pumping. *Microfluid Nanofluid* 12(6):887–895
- Gorman BR, Wikswo JP (2008) Characterization of transport in microfluidic gradient generators. *Microfluid Nanofluid* 4(4):273
- Gupta K, Kim DH, Ellison D, Smith C, Kundu A, Tuan J, Levchenko A (2010) Lab-on-a-chip devices as an emerging platform for stem cell biology. *Lab Chip* 10(16):2019–2031
- Hadikhani P, Borhani N, Hashemi SMH, Psaltis D (2019) Learning from droplet flows in microfluidic channels using deep neural networks. *Sci Rep* 9(1):8114
- Hong SH, Cornelius J, Wang Y, Pant K (2019) Fault compensation by online updating of genetic algorithm-selected neural network model for model predictive control. *SN Appl Sci* 1(11):1488
- Hu C, Liu J, Chen H, Nie F (2017) Microfluidic platforms for gradient generation and its applications. *Biochem Anal Biochem* 6(320):2161–1009
- Hu Z, Chen X, Wang L (2018) Design and fabrication of concentration-gradient generators with two and three inlets in microfluidic chips. *Chem Eng Technol* 41(3):489–495
- Keenan TM, Folch A (2008) Biomolecular gradients in cell culture systems. *Lab Chip* 8(1):34–57
- Kothapalli CR, Van Veen E, De Valence S, Chung S, Zervantonakis IK, Gertler FB, Kamm RD (2011) A high-throughput microfluidic assay to study neurite response to growth factor gradients. *Lab Chip* 11(3):497–507
- Lee K, Kim C, Jung G, Kim TS, Kang JY, Oh KW (2010) Microfluidic network-based combinatorial dilution device for high throughput screening and optimization. *Microfluid Nanofluid* 8(5):677–685
- Lore KG, Stoecklein D, Davies M, Ganapathysubramanian B, Sarkar S (2018) A deep learning framework for causal shape transformation. *Neural Netw* 98:305–317
- Malkiel I, Mrejen M, Nagler A, Arieli U, Wolf L, Suchowski H (2018) Plasmonic nanostructure design and characterization via deep learning. *Light Sci Appl* 7(1):60
- MATLAB version 9.3.0.713579 (R2017b) (2017) Natick. The MathWorks Inc., Massachusetts
- Meier M, Lucchetta EM, Ismagilov RF (2010) Chemical stimulation of the *Arabidopsis thaliana* root using multi-laminar flow on a microfluidic chip. *Lab Chip* 10(16):2147–2153
- Nandagopal S, Wu D, Lin F (2011) Combinatorial guidance by CCR7 ligands for T lymphocytes migration in co-existing chemokine fields. *PLoS ONE* 6(3):e18183
- Oh KW, Lee K, Ahn B, Furlani EP (2012) Design of pressure-driven microfluidic networks using electric circuit analogy. *Lab Chip* 12(3):515–545
- Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Desmaison A (2019) PyTorch: an imperative style, high-performance deep learning library. In: *Advances in neural information processing systems*, pp 8024–8035
- Riordon J, Sovilj D, Sanner S, Sinton D, Young EW (2018) Deep learning with microfluidics for biotechnology. *Trends Biotechnol* 37(3):310–324
- So S, Mun J, Rho J (2019) Simultaneous inverse-design of material and structure via deep-learning: demonstration of dipole resonance engineering using core-shell nanoparticles. *ACS Appl Mater Interfaces* 11(27):24264–24268
- Stoecklein D, Lore KG, Davies M, Sarkar S, Ganapathysubramanian B (2017) Deep learning for flow sculpting: insights into efficient learning using scientific simulation data. *Sci Rep* 7:46368
- Toh AG, Wang ZP, Yang C, Nguyen NT (2014) Engineering microfluidic concentration gradient generators for biological applications. *Microfluid Nanofluid* 16(1–2):1–18
- Vozzi G, Mazzei D, Tirella A, Vozzi F, Ahluwalia A (2010) Finite element modelling and design of a concentration gradient generating bioreactor: application to biological pattern formation and toxicology. *Toxicol In Vitro* 24(6):1828–1837
- Wang Y, Mukherjee T, Lin Q (2006) Systematic modeling of microfluidic concentration gradient generators. *J Micromech Microeng* 16(10):2128
- Wang H, Chen CH, Xiang Z, Wang M, Lee C (2015) A convection-driven long-range linear gradient generator with dynamic control. *Lab Chip* 15(6):1445–1450
- Wang J, Brisk P, Grover WH (2016) Random design of microfluidics. *Lab Chip* 16(21):4212–4219
- Wang X, Liu Z, Pang Y (2017) Concentration gradient generation methods based on microfluidic systems. *RSC Adv* 7(48):29966–29984
- Wang J, Zhang N, Chen J, Rodgers VG, Brisk P, Grover WH (2019) Finding the optimal design of a passive microfluidic mixer. *Lab Chip* 19(21):3618–3627
- Zhang R, Tao J, Gao F (2016) A new approach of Takagi-Sugeno fuzzy modeling using an improved genetic algorithm optimization for oxygen content in a coke furnace. *Ind Eng Chem Res* 55(22):6465–6474

Zhou Y, Wang Y, Mukherjee T, Lin Q (2009) Generation of complex concentration profiles by partial diffusive mixing in multi-stream laminar flow. *Lab Chip* 9(10):1439–1448

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.