A fast and efficient deep learning procedure for tracking droplet motion in dense microfluidic emulsions

Mihir Durve1,5, Fabio Bonaccorso1,2,3, Andrea Montessori1, Marco Lauricella1, Adriano Tiribocchi1,2 and Sauro Succi1,2,4

1Center for Life Nano Science@La Sapienza, Istituto Italiano di Tecnologia, Viale Regina Elena, 291, 00161 Roma, Italy
2Istituto per le Applicazioni del Calcolo CNR, via dei Taurini 19, Rome, Italy
3Department of Physics and INFN, University of Rome Tor Vergata, Via della Ricerca Scientifica, 1 00133, Rome, Italy
4Institute for Applied Computational Science, John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, USA
5Quantitative Life Sciences Unit, The Abdus Salam International Centre for Theoretical Physics (ICTP), Trieste 34151, Italy

We present a deep learning-based object detection and object tracking algorithm to study droplet motion in dense microfluidic emulsions. The deep learning procedure is shown to correctly predict the droplets’ shape and track their motion at competitive rates as compared to standard clustering algorithms, even in the presence of significant deformations. The deep learning technique and tool developed in this work could be used for the general study of the dynamics of biological agents in fluid systems, such as moving cells and self-propelled microorganisms in complex biological flows.

This article is part of the theme issue ‘Progress in mesoscale methods for fluid dynamics simulation’.

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.5490991.

© 2021 The Author(s) Published by the Royal Society. All rights reserved.
1. Introduction

In recent times, machine learning (ML) has gained enormous attention within the scientific community [1], as it provides a general automated framework to accomplish highly complex computational tasks through a self-improving trial-and-error minimization procedure [2].

Although often overhauled (for a critical assessment see [3,4]), ML proves indeed successful in a number of tasks, such as object classification and image recognition [5,6], handwriting recognition [7], text analysis [8,9], voice recognition [10] and more recently also physical [11] and biological applications such as protein folding [12].

ML algorithms perform these tasks by realizing a model that outputs decisions or predictions starting from (big) data inputs that are used to train the network by recursive optimization of its parameters (the weights connecting nodes of the network across two subsequent layers). A particularly successful instance of ML, called deep learning (DL), uses multi-layer artificial neural networks, as opposed to the so-called shallow networks, which consist of just a few hidden layers [13]. Deep learning-based approaches have been successfully employed in a number of applications from medical diagnosis [14–17] to physics [11,18]. In microfluidics, deep learning has been used to study physical systems and quality control processes. Mahdi et al. [19] used neural networks to predict the size of the droplets in an emulsion. Khor et al. [20] developed a convolutional autoencoder model to discover a low-dimensional representation to describe droplet shapes within a concentrated emulsion and study the stability of droplets in an emulsion. Hadikhani et al. [21] trained neural networks to estimate fluid and flow parameters by analysing digital images of fluid in microchips.

In recent years, high internal phase emulsions (HIPE) and soft flowing crystals, namely ordered states of flowing matter characterized by a highly ordered collection of liquid droplets arranged in crystal-like structures, are showing their potential in many fields of science and engineering. Indeed, due to the typical porous polymeric matrix (see figure 1 for two examples of emulsions in microfluidic channels (a) during the early-stage formation and (b) at high packing fractions), they can be efficiently employed in a wide range of advanced applications, such as catalyst supports, ion-exchange modules, separation media, electrochemical sensing, to name but a few. Moreover, they are very fit for bioinspired applications such as scaffolds in tissue engineering, where the interconnected pore structure is essential for their function. Besides their relevance for a host of applications in the fields mentioned above, soft porous materials also raise a fundamental challenge to non-equilibrium thermodynamics, since they display properties that cannot be traced back to any of the three basic states of matter. For instance, HIPEs are binary mixtures of immiscible liquids, each featuring linear Newtonian rheology, that exhibit highly non-Newtonian mechanical and rheological properties. Thus, a full understanding of the rich dynamics underlying these classical many-body systems is required to optimize the microfluidic devices employed for their production and to fine-tune their properties for different purposes.

The aim of this work is to demonstrate that general deep learning-based tools can be adapted, tuned and profitably used to extract trajectories of droplets starting from digital frames of the fluid system, as obtained either by experiment or via computer simulations. The deep learning procedure is here applied to the case of dense flowing emulsions in micro-channels to compute the paths followed by the droplets, as obtained via lattice Boltzmann simulations. This represents a very first step toward a more complex task, consisting in the characterization of the dynamics of this many-body soft system in terms of effective equations of motion of the droplets flowing in the continuum phase.

The results are encouraging, in that the procedure proves capable of fast (about 400 frames/second) recognition and tracking of droplets with significant departures from the spherical shape. Given its flexibility, it is expected that the procedure developed in this paper can be readily generalized to the study of the dynamics of moving agents in micro- and nanoflows of biological interest [22].
The paper is structured as follows. In the next section, we describe the ML procedure and, shortly, the lattice Boltzmann method used to produce the validation data. Section 3 is dedicated to a discussion of the numerical results and some conclusive remarks close the paper.

2. Methods

In the following, we provide details of the ML procedure as well as of the lattice Boltzmann method, which is the simulation technique employed to produce the validation data. We begin by discussing the latter.

(a) Lattice Boltzmann method for dense emulsions

In this section, we briefly describe the numerical model, namely an extended colour-gradient approach with repulsive near-contact interactions, previously employed in refs [23–27]. In the multicomponent LB model, two sets of discrete distribution functions evolve via the usual streaming-collision algorithm (see [28,29]):

\[
\begin{align*}
    f_i^k(x + c_i \Delta t, t + \Delta t) &= f_i^k(x, t) + \Omega_i [(f_i^k(x, t)] + S_i(x, t). \\
    \end{align*}
\]

In the above equation, \( f_i^k \) is the discrete distribution function, representing the probability of finding a particle of the \( k \)th component at position \( x \), time \( t \) with discrete velocity \( c_i \) and \( S_i \) is a source term coding for the effect of external forces (such as gravity, near-contact interactions, etc).

In equation (2.1), the time step is taken as \( \Delta t = 1 \), and the index \( i \) runs over the discrete lattice directions \( i = 1, \ldots, b \), being \( b = 9 \) for a two-dimensional nine-speed lattice (D2Q9). The density \( \rho^k \) of the \( k \)th component and the total linear momentum of the mixture \( \rho u = \sum_k \rho^k u^k \) are obtained, respectively, via the zeroth- and the first-order moment of the lattice distributions \( \rho^k(x, t) = \sum_i f_i^k(x, t) \) and \( \rho u = \sum_i \sum_k f_i^k(x, t) c_i. \) The collision operator splits into three parts [30,31]:

\[
\Omega_i^k = (\Omega_i^k)^{(3)} + (\Omega_i^k)^{(1)} + (\Omega_i^k)^{(2)},
\]

where \( (\Omega_i^k)^{(1)} \) stands for the standard collisional relaxation [28], \( (\Omega_i^k)^{(2)} \) codes for the perturbation step [30], contributing to the buildup of the interfacial tension while \( (\Omega_i^k)^{(3)} \) is the recolouring step [30,32], which promotes the segregation between the two species, minimizing their mutual diffusion.

By performing a Chapman–Enskog multiscale expansion, it can be shown that the hydrodynamic limit of equation 2.1 is a set of equations for the conservation of mass and linear momentum (i.e. the Navier–Stokes equations), with a capillary stress tensor of the form:

\[
\Sigma = -\tau \sum_i \sum_k (\Omega_i^k)^{(2)} c_i c_i = -\frac{\sigma}{2|\nabla \rho|} (|\nabla \rho|^2 I - \nabla \rho \otimes \nabla \rho),
\]

\( \tau \) being the collision relaxation time, controlling the kinematic viscosity via the relation \( \nu = c_s^2 (\tau - 1/2) (c_s = 1/\sqrt{3} \) the sound speed of the model) and \( \sigma \) is the surface tension [28,29].
In equation (2.3), the symbol $\otimes$ denotes a dyadic tensor product. The stress-jump condition across a fluid interface is then augmented with an intra-component repulsive term aimed at condensing the effect of all the repulsive near-contact forces (i.e. Van der Waals, electrostatic, steric and hydration repulsion) acting on much smaller scales ($\sim O(1 \text{ nm})$) than those resolved on the lattice (typically well above hundreds of nanometres). It takes the following form:

$$T^1 \cdot n - T^2 \cdot n = -\nabla (\sigma I - \sigma (n \otimes n)) - \pi n,$$

where $\pi[h(x)]$ is responsible for the repulsion between neighbouring fluid interfaces and $h(x)$ being the distance between interacting interfaces along the normal $n$.

The additional term can be readily included within the LB framework, by adding a forcing term acting only on the fluid interfaces in near-contact, namely:

$$F_{\text{rep}} = -A_h[h(x)] n \delta I.$$  \hspace{1cm} (2.5)

In the above, $A_h[h(x)]$ is the parameter controlling the strength (force per unit volume) of the near-contact interactions, $h(x)$ is the distance between the interfaces, $n$ is a unit vector normal to the interface and $\delta I \propto \nabla \phi$ is a function, proportional to the phase-field $\phi = (\rho^1 - \rho^2)/(\rho^1 + \rho^2)$, employed to localize the force at the interface.

To conclude, the addition of the repulsive force (added to the right-hand side of equation (2.1)) naturally leads to the following (extended) conservation law for the momentum equation:

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot \rho uu = -\nabla p + \nabla \cdot [\rho v(\nabla u + \nabla u^T)] + \nabla \cdot (\Sigma + \pi I),$$  \hspace{1cm} (2.6)

namely the Navier–Stokes equation for a multicomponent system, augmented with a surface-localized repulsive term, expressed through the gradient of the potential function $\pi$.

(b) Deep learning procedure

The deep learning approach employed in this work consists of three basic steps: (1) data preparation, (2) network training and set-up and (3) inference (see figure 2), which we now discuss in some detail.

In order to extract trajectories of droplets from the digital images, the deep learning-based application must accomplish two tasks: (i) identify the droplets and find their location in a given image and (ii) track the droplet displacements between successive frames. In our implementation, these two tasks are performed via two different algorithms: droplet detection makes use of YOLO [33,34], a state-of-the-art, deep neural network algorithm for real-time object detection. The YOLO algorithm has been shown to be significantly faster than most other techniques for object detection and classification [34,35]. The YOLO algorithm can be implemented on the CPU as well as GPUs to gain a higher rate of image analysis (inference speed) making it a good choice for real-time video analysis. YOLO takes an image as an input and outputs a list of detected objects with their classes, confidence of detection and their locations. This information is then passed to DeepSORT [36], an effective multi-object tracker, whose task is precisely to keep track of the detected objects as they move in time.

(i) Data preparation

In order to identify droplets in a given image, the YOLO network must be trained with several examples, called training dataset, which should have a wide enough statistical significance. Its preparation is arguably the most labour intensive task of the training phase and will have a high impact on the validity of the whole procedure. For supervised learning algorithms, training data typically consist of inputs, such as images, and the expected output of the network called labels. In order to train a network to identify droplets in an image, the training data must include several examples, each consisting of an image of a system with droplets, along with a text file containing
Figure 2. The three basic steps involved in the deep learning-based application for trajectory extraction of droplets in dense emulsions. From left to right: (1) synthetic data preparation for the network training, (2) network training and (3) inference and validation based on the data provided by the LB simulation (top). The output is a set of droplets along with their trajectories (bottom). (Online version in colour.)

Figure 3. A sample image from training dataset with its label in the format needed to train the YOLO network, namely the sequential number of the object, the class of the object (droplet), the coordinates of its centre (\(x\) and \(y\)), the width (\(w\)) and height (\(h\)) of the bounding box. (Online version in colour.)

| Obj | Class | \(x\) | \(y\) | \(w\) | \(h\) |
|-----|-------|-------|-------|-------|-------|
| 1   |       | 0.16  | 0.38  | 0.06  | 0.22  |
| 2   |       | 0.26  | 0.51  | 0.06  | 0.22  |
| 3   |       | 0.36  | 0.47  | 0.06  | 0.22  |
| 4   |       | 0.46  | 0.58  | 0.08  | 0.17  |
| 5   |       | 0.66  | 0.52  | 0.08  | 0.16  |
| 6   |       | 0.76  | 0.29  | 0.10  | 0.13  |
| 7   |       | 0.84  | 0.41  | 0.07  | 0.19  |

their sizes (height and width) and centre positions (\(x\) and \(y\) coordinates) within the image as labels.

To generate the training dataset, we prepared several images by randomly placing a fixed number (7) of droplets with varying aspect ratios in a uniformly dark background of size 1040 × 450 pixels, since this is the typical output of the LB simulations. For each sample, we build the image and note the positions and sizes of all the droplets that we placed. An example from the training dataset is shown in figure 3, along with the text file containing the labels of each droplet, namely the coordinate of its centre and the height and width of its bounding box. The code to generate the training data is given in the data access sheet.

(ii) Network training and setup

The network setup is performed by using Darknet [37], an open-source neural network framework written in C and CUDA, and developed to support YOLO on both CPU and GPU
devices: it supports the most common neural network operations such as convolution, max pooling and various activation functions. Once the network is trained, the same framework can then be used as an inference engine.

For our task, we selected a predefined, ready-to-implement network architecture, known as yolov3-tiny, consisting of 24 layers, branching out in many convolutional and pooling layers with a total of approximately 8 million trainable parameters (weights). The training of the YOLO network was carried out by employing 10,000 training images as described above. For the training, we used GPU implementation of Darknet on a system consisting of NVIDIA RTX 2060 Super GPU, common in middle-level desktop configurations for gaming. The Github repository outlines the details to carry out training of the network [38] as well as in the data access sheet. The training operation is realized in many batches. In each batch, a fixed number of images (called batch size) are given as input to the network. Based on the predicted output of the network, a loss function is calculated and the parameters of the network are then updated to minimize it. The YOLO algorithm uses a multi-component loss function [33,34] accounting for the errors in classifying detected objects, locating the objects, as well as error in prediction confidence for the detected objects. The chosen values of parameters for the training (batches, batch size, etc.) can be found in [39]. As the training progresses, the total loss decreases to a near-zero value, as shown in figure 4.

The accuracy of a trained network for object detection is measured by a metric called average precision (AP), a figure of merit between 0 and 1. The AP of a network can be computed using a separate validation dataset, and in our case, we reserved a fraction (5%) of the training dataset as validation dataset. In order to compute the AP, all the images of the validation dataset are given as input to the network and all detected objects (droplets) are tabulated in descending order of their individual detection confidence score. Starting from the top of the table, for each detected object, precision and recall values are computed as follows:

\[
P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}. \tag{2.7}
\]

Here, TP is the number of true positive detection (droplet does exist and it is detected), FP is the number of false-positive detection (i.e. a droplet is detected but it doesn’t exist) and FN is the number of false-negative detection (i.e. a droplet does exist but it is not detected).

To determine whether the detection is a true positive or a false-positive, we use a quantity called Intersection over Union (IoU) value of the detection. In geometric terms, IoU is given by the area of the intersection divided by the area of the union of the ground truth bounding box, with

---

**Figure 4.** Total loss function as the training progresses. In the inset, we show the mean average precision on the validation data computed during various stages of the training. (Online version in colour.)
predicted bounding box (see figure 5). We set the IoU threshold to 0.5, i.e. if the IoU is $\geq 0.5$ the prediction is classified as a true positive.

After computing Precision and Recall for all the predictions in the validation dataset, the Precision–Recall curve is plotted and the AP is computed as, $AP = \int_0^1 p(r) \, dr$. We also measured the mean average precision (mAP), defined as the average precision among all the classes of objects in the dataset, as reported in the inset of figure 4. We observe that the mAP value monotonically increases as the training progresses and saturates to about 0.98 after a sufficient number of training batches, indicating that the training procedure achieves nearly perfect accuracy. At the end of the training, the optimized parameters of the network are saved and used for the inference step, which we discuss next.

(iii) Inference step

In the inference step, the input data, i.e. the density maps produced by the LB simulations at each time step, are presented as an input to the trained network so that YOLO detects the droplets in the simulated image and passes this information along to the object tracking algorithm (DeepSORT). Together, YOLO and DeepSORT detect and track the droplets at each time step, producing all the trajectories.

The particular network architecture can produce accurate trajectories even if the dimension in pixel of the input image is not exactly the same used during the training phase: training is a slow process that scales with the image size. It is a good advantage to reuse the coefficients of the YOLO network for images even $2 \times$ or $3 \times$ bigger, for which the training would have been significantly slower.

In the next section, we apply the network to two microfluidic cases simulated using the lattice Boltzmann method described in §2a, namely a straight and tapered channel, respectively. The first scenario should be easy for the detection (droplets are almost always circles) and tracking (droplets do not merge or split). The second case is a more interesting one, since the nozzle forces the droplets to aggregate and squeeze to fit the central channel: the shape of each droplet changes in time and some break-ups are present, which makes it a good test case.

3. Results and discussion

To test our trained network, we analysed video data generated for the two aforementioned applications. For the details, see the data access sheet. The first case (Simple channel) consists of a thin inlet channel in which droplets are continuously produced and flow downstream in the main microchannel (see figure 6). Sequences of images of size $850 \times 250$ pixels are generated and stacked to prepare a video file which serves as an input to the trained YOLO network for droplet detection.

The YOLO network analyses each frame of the video file, computing the bounding boxes for each detected droplet. The information of the bounding boxes, i.e. their size and location, is then passed to DeepSORT to track the droplets in the sequential images using a unique ID.
associated with each detected droplet. In figure 7, the inferred trajectories, as the centres of the bounding boxes, of the tracked droplets are shown. A video file (electronic supplementary material, simple_channel.mp4) showing the detection and tracking of the droplets is provided in the electronic supplementary material.

The second system (Tapered channel) consists of a dense emulsion, flowing inside a tapered channel connected to a thin straight channel (see figure 8). Simulation output images of size 950 × 500 pixels were stacked to prepare input for the YOLO network. The predicted bounding boxes are shown in figure 9a, and the trajectories of the droplets, i.e. the centres of the predicted bounding boxes, compiled by analysing all the frames are shown in figure 9b. A video file (electronic supplementary material, tapered_channel.mp4) of the detection and tracking process is also provided in the electronic supplementary material.

(a) Computational performance

The computational time required to identify and track objects is an important factor for many applications. In table 1, we report the performances relative to the full track-and-detect process on different computers, measuring the inference speed in frames per second (fps). We report the speed on two machines using only general purpose CPUs (in a typical configuration for a notebook and for a computing server) and using a middle-level GPU for a desktop (NVIDIA RTX 2060 Super). From this table, it is clear that the inference proceeds at the significant speed of 400 frames per second on the machine powered by GPUs.

In another approach, feature extraction and clustering algorithms have been used to track moving objects at around 35 fps [40] on multi-core CPUs. Even though it is not fair to compare the inference speed for algorithms with datasets of different complexity and using different computing machines, it is worthwhile to note that, with the method demonstrated in this work,
Figure 8. A snapshot of the tapered channel. Droplets enter the convergent nozzle from the left, pushing to enter the end of the channel at the right. The motion is now fully two-dimensional, with significant droplet deformations, thus setting a more demanding target to the deep learning procedure. (Online version in colour.)

Figure 9. (a) Droplet detected for the Tapered channel by the YOLO network for the same snapshot shown in figure 8. (b) DeepSort uniquely labels each droplet to compute all the trajectories. (Online version in colour.)

Table 1. Average inference speed for the two cases on various types of devices in frame per second (fps). Performances on GPU are almost $100 \times$ the values obtained with a full CPU implementation.

| machine                      | inference speed |
|------------------------------|-----------------|
|                              | simple channel  | tapered channel |
| CPU (1X Intel i3-2328M)      | 1.61 fps        | 0.69 fps        |
| CPU (16X Intel Xeon Platinum 8176) | 8.14 fps         | 5.10 fps        |
| GPU (NVIDIA RTX 2060 Super)  | 400 fps         | 370 fps         |

The complete process of identification and tracking of droplets can take place at much higher rates (few hundred fps) without trading-off the accuracy, and taking advantage of the existing ML frameworks highly optimized for the GPUs.

The application developed by training a cutting-edge deep learning model can be efficiently used to compute the trajectories of droplets in dense emulsion systems simulated by the lattice Boltzmann method or any other computational method capable of capturing the relevant physics. It is worth noting that in both cases explored, and especially in the case of the Tapered channel,
the data used to train the network depart noticeably from those acquired via simulations. The fact that the network is nevertheless able to identify the droplets and efficiently track them over sequential frames bodes well for its generalization to more complex applications, particularly for tracking deformable biological bodies in complex fluid flows. This will make the subject of future work.

4. Conclusion

In this work, we demonstrated an automated procedure to adapt a cutting edge deep learning-based application, called YOLO and DeepSORT, to infer trajectories of droplets in dense emulsions in microchannels. We have shown that the deep network can effectively be trained with synthetic data, thereby bypassing the labour-intensive process of acquiring the training data. The present tool performs competitively with state-of-the-art object tracking and detection algorithms in terms of inference speed.

Extracting the information about the trajectories could also provide useful insights in the study of several other complex many-body systems, such as moving groups of animals and biological microorganisms. We also plan to extend the ML procedure not only to detect droplets and track their trajectories but also to exploit the latter data to infer the effective equations of motion of this complex many-body system, possibly using generative adversarial networks [41].

Data accessibility. We provided further ESM for data access.

(i) training python code: make_training_data.py
(ii) Data access sheet.pdf: explanation of the files
(iii) yolov3-customs.weights: weights file

Authors’ contributions. M. Durve: wrote the paper, implemented the neural network, analysed the data. A. Montessori: wrote the paper, performed lattice Boltzmann simulations. F. Bonaccorso: implemented the neural network and analysed the data. S. Succi: wrote the paper and analysed the data. All the authors contributed to the writing process of the manuscript.

Competing interests. We declare we have no competing interests.

Funding. The research leading to these results has received funding from the European Research Council under the Horizon 2020 Programme Grant Agreement no. 739964 (‘COPMAT’).

Acknowledgements. The research leading to these results has received funding from the European Research Council under the Horizon 2020 Programme Grant Agreement n. 739964 (‘COPMAT’). A.M. acknowledges the CINECA Computationsal Grant ISCRA-C IsC83 - ‘SDROMOL’, id. HP10CZXK6R under the ISCRA initiative, for the availability of high performance computing resources and support.

References

1. LeCun Y, Bengio Y, Hinton G. 2015 Deep learning. Nature 521, 436–444. (doi:10.1038/nature14539)
2. Mitchell TM 1997 Machine learning, 1st edn. McGraw-Hill series in computer science. McGraw-Hill.
3. Coveney PV, Dougherty ER, Highfield RR. 2016 Big data need big theory too. Phil. Trans. R. Soc. A 374, 20160153. (doi:10.1098/rsta.2016.0153)
4. Succi S, Coveney PV. 2019 Big data: the end of the scientific method? Phil. Trans. R. Soc. A 377, 20180145. (doi:10.1098/rsta.2018.0145)
5. Wu J, Yu Y, Huang C, Yu K. 2015 Deep multiple instance learning for image classification and auto-annotation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3460–3469.
6. Rawat W, Wang Z. 2017 Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. 29, 2352–2449. (doi:10.1162/neco_a_00990)
7. Darmatasia, Fanany MI. 2017 Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM). In 2017 5th International Conference on Information and Communication Technology (ICoIC7), pp. 1–6.
8. Hasan A, Moin S, Karim A, Shamshirband S. 2018 Machine learning-based sentiment analysis for twitter accounts. *Math. Comput. Appl.* 23, 11. (doi:10.3390/mca23010011)

9. Ramadhani AM, Goo HS. 2017 Twitter sentiment analysis using deep learning methods. In *2017 7th International Annual Engineering Seminar (InAES)*, pp. 1–4.

10. Tandel NH, Prajapati HB, Dabhi VK. 2020 Voice recognition and voice comparison using machine learning techniques: a survey. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 459–465.

11. Carleo G, Cirac I, Cranmer K, Daudet L, Shuld M, Tishby N, Vogt-Maranto L, Zdeborová L. 2019 Machine learning and the physical sciences. *Rev. Mod. Phys.* 91, 045002. (doi:10.1103/RevModPhys.91.045002)

12. Senior AW et al. 2020 Improved protein structure prediction using potentials from deep learning. *Nature* 577, 706–710. (doi:10.1038/s41586-019-1923-7)

13. Goodfellow I, Bengio Y, Courville A. 2016 *Deep learning*. Cambridge, MA: MIT Press.

14. Chen CL, Mahjoubfar A, Tai LC, Blaby IK, Huang A, Niazi KR, Jalali B. 2016 Deep learning in label-free cell classification. *Sci. Rep.* 6, 21471. (doi:10.1038/srep21471)

15. Guo Z, Li X, Huang H, Guo N, Li Q. 2019 Deep learning-based image segmentation on multimodal medical imaging. *IEEE Trans. Radiat. Plasma Med. Sci.* 3, 162–169. (doi:10.1109/TRPMS.2018.2890359)

16. Fujiyoshi H, Hirakawa T, Yamashita T. 2019 Deep learning-based image recognition for autonomous driving. *IATSS Res.* 43, 244–252. (doi:10.1016/j.iatssr.2019.11.008)

17. Xie J, Liu R, Luttrell J, Zhang C. 2019 Deep learning based analysis of histopathological images of breast cancer. *Front. Genet.* 10, 80. (doi:10.3389/fgene.2019.00080)

18. Sharir O, Levine Y, Wies N, Carleo G, Shashua A. 2020 Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.* 124, 020503. (doi:10.1103/PhysRevLett.124.020503)

19. Mahdi Y, Daoud K. 2017 Microdroplet size prediction in microfluidic systems via artificial neural network modeling for water-in-oil emulsion formulation. *J. Dispersion Sci. Technol.* 38, 1501–1508. (doi:10.1080/01932691.2016.1257391)

20. Khor JW, Jean N, Luxenberg ES, Ermon S, Tang SKY. 2019 Using machine learning to discover shape descriptors for predicting emulsion stability in a microfluidic channel. *Soft Matter* 15, 1361–1372. (doi:10.1039/C8SM02054J)

21. Hadikhani P, Borhani N, Hashemi S, Psaltis D. 2019 Learning from droplet flows in microfluidic channels using deep neural networks. *Sci. Rep.* 9, 8114. (doi:10.1038/s41598-019-44556-x)

22. Bernaschi M, Melchionna S, Succi S. 2019 Mesoscopic simulations at the physics-chemistry-biology interface. *Rev. Mod. Phys.* 91, 025004. (doi:10.1103/RevModPhys.91.025004)

23. Montessori A, Lauricella M, Tirelli N, Succi S. 2019 Mesoscale modelling of near-contact interactions for complex flowing interfaces. *J. Fluid Mech.* 872, 327–347. (doi:10.1017/jfm.2019.372)

24. Montessori A, Lauricella M, Tiribocchi A, Succi S. 2019 Modeling pattern formation in soft flowing crystals. *Phys. Rev. Fluids* 4, 072201. (doi:10.1103/PhysRevFluids.4.072201)

25. Montessori A, Tiribocchi A, Lauricella M, Bonaccorso F, Succi S. 2021 Wet to dry self-transitions in dense emulsions: From order to disorder and back. *Phys. Rev. Fluids* 6, 023606. (doi:10.1103/PhysRevFluids.6.023606)

26. Montessori A, Tiribocchi A, Lauricella M, Bonaccorso F, Succi S. 2021 Mesoscale modelling of droplets’ self-assembly in microfluidic channels. *Soft Matter* 17, 2374–2383. (doi:10.1039/DOSM02047H)

27. Montessori A, Tiribocchi A, Bonaccorso F, Lauricella M, Succi S. 2020 Lattice Boltzmann simulations capture the multiscale physics of soft flowing crystals. *Phil. Trans. R. Soc. A* 378, 20190406. (doi:10.1098/rsta.2019.0406)

28. Succi S. 2018 *The lattice Boltzmann equation: for complex states of flowing matter*. Oxford, UK: Oxford University Press.

29. Krüger T, Kusumaatmaja H, Kuzmin A, Shardt O, Silva G, Viggen EM. 2017 The lattice Boltzmann method. *Springer International Publishing* 10, 978–973. (doi:10.1007/978-3-319-44649-3)

30. Gunstensen AK, Rothman DH, Zaleski S, Zanetti G. 1991 Lattice Boltzmann model of immiscible fluids. *Phys. Rev. A* 43, 4320. (doi:10.1103/PhysRevA.43.4320)
31. Leclaire S, Reggio M, Trépanier JY. 2012 Numerical evaluation of two recoloring operators for an immiscible two-phase flow lattice Boltzmann model. *Appl. Math. Model.* 36, 2237–2252. (doi:10.1016/j.apm.2011.08.027)

32. Latva-Kokko M, Rothman DH. 2005 Diffusion properties of gradient-based lattice Boltzmann models of immiscible fluids. *Phys. Rev. E* 71, 056702. (doi:10.1103/PhysRevE.71.056702)

33. Redmon J, Divvala S, Girshick R, Farhadi A. 2016 You only look once: unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788.

34. Redmon J, Farhadi A. 2018 YOLOv3: an incremental improvement. (http://arxiv.org/abs/1804.02767v1)

35. Sanchez SA, Romero HJ, Morales AD. 2020 A review: comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. *IOP Conference Series: Materials Science and Engineering* 844, 012024. (doi:10.1088/1757-899X/844/1/012024)

36. Wojke N, Bewley A, Paulus D. 2017 Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649.

37. Redmon J Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/, 2013–2016.

38. https://github.com/AlexeyAB/darknet.

39. Batches = 10000, batch size = 64, learning rate = 0.001.

40. Keivani A, Tapamo J, Ghayoor F. 2017 Motion-based moving object detection and tracking using automatic K-means. In *2017 IEEE AFRICON*, pp. 32–37.

41. Yang L, Daskalakis C, Karniadakis GE. 2020 Generative ensemble-regression: learning stochastic dynamics from discrete particle ensemble observations. (http://arxiv.org/abs/2008.01915)