Flow (2025), **5** E6
doi:10.1017/flo.2025.2

FLOW CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

DPD-v2: Generalised deep particle diffusometry for varied particle shapes and experimental conditions

Pranshul Sardana and Steven T. Wereley

School of Mechanical Engineering, Purdue University, West Lafayette, IN, USA
Corresponding author: Pranshul Sardana; Email: psardana@purdue.edu

Received: 31 August 2024; **Revised:** 13 December 2024; **Accepted:** 21 January 2025

Keywords: convolutional neural networks; deep learning; diffusion coefficient measurement; particle-based measurements; particle diffusometry

Abstract

Particle diffusometry (PD) is a technique of measuring the diffusion coefficient of a fluid sample by seeding it with tracer particles and observing their motion under a microscope. In microfluidic set-ups, the observed particles are often defocused and their motion is affected by factors such as fluid flow, which leads to high errors for conventional and deep learning-based PD (DPD) algorithms. This work improves the performance of DPD models by updating their architecture, avoiding temporal averaging in the input, and exploring the impact of various choices during training. These models provide state-of-the-art performance for generalised datasets regardless of particle shapes, concentration, flow or image noise and are called DPD-v2. These models provide a mean absolute error of $0.09\mu m^2 s^{-1}$ for Gaussian particles and $0.07\mu m^2 s^{-1}$ for defocused particles, which is 2x–4x lower errors as compared with the two following best methods. The performance of DPD-v2 models increases with crop size and the use of multiple stacks of images. The outputs of the DPD-v2 models were compared against the outputs from conventional algorithms on Gaussianised experimental no flow datasets, which provided $< 0.5\mu m^2 s^{-1}$ mean absolute difference. Hence, the DPD-v2 models can be used in real-world scenarios.

Impact Statement

Particle diffusometry (PD) methods have various applications in disease detection and nanoparticle characterisation. However, extending these applications beyond laboratory environments has been challenging due to the limitations of existing algorithms, which struggle with factors such as flow and particle defocus. This work provides a robust method for these real-world scenarios, which allows PD to realise its full potential in the mentioned applications. Once the diffusion coefficient is known, one can also calculate other physical properties of a fluid such as viscosity and temperature. The deep learning architectures created in the current work can also be trained to predict velocity in the presence of diffusion, which can enable the use of smaller particles for micro-particle image velocimetry.

1. Introduction

Diffusion is a natural phenomenon due to molecules random motion in a fluid. The diffusion coefficient governs the rate of diffusion, and its measurement is known as diffusometry. It has applications in domains such as disease detection (Clayton *et al.,* 2019), biological and material investigations

© The Author(s), 2025. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

(Zareh *et al.,* 2012) and drug development (Ganser *et al.,* 2009). For applications such as disease detection, the diffusometry devices can be used in a point-of-care setting, and diffusometry is performed optically (Colbert *et al.,* 2021). Here, the fluid sample is seeded with fluorescent micro-nano scale particles, and the motion of these particles is observed and imaged over multiple time stamps under high magnifications (Raffel *et al.,* 2007). When these particles are used for diffusometry, it is called Particle diffusometry (PD).

There are various methods for processing these sequences of particle images. Single particle tracking is a method where individual particles are located in each frame and tracked over time (Allan *et al.,* 2023; Crocker & Grier 1996). Next, the particles' mean square displacements (MSDs) between the frames are calculated and fit using a power law. The slope of this fit is used to calculate the diffusion coefficients. Single particle tracking methods show high performance if there is no flow in the underlying fluid. However, their performance degrades significantly when flow, a common experimental phenomenon, is present. The power laws cannot account for displacements because of arbitrary flows (Sardana & Wereley 2023). Additionally, the performance degrades significantly when the particles do not appear Gaussian shaped and are challenging to detect (Barnkob & Rossi 2021). This limitation of the particle shapes can be overcome by finding the MSDs using correlation-based methods (Liberzon *et al.,* 2020; Raffel *et al.,* 2007), where multiple particles in a spatial vicinity can be observed jointly, and the correlation between the frames can provide the net displacement. However, these methods still rely on a power law and underperform when the underlying fluid has a flow.

The diffusion coefficient can also be found without using the power laws (Olsen & Adrian 2000). This method determines the diffusion coefficient by subtracting the auto-correlation peak width from the cross-correlation peak width. However, this method relies on a strong assumption that the particles have a Gaussian shape, and the performance is under-documented when the particles do not fall under this assumption. An alternative way is to deconvolve the cross-correlation from the auto-correlation and find the two-dimensional probability density function (PDF) (Ahmadzadegan *et al.,* 2020). The standard deviation of the obtained PDF is used to calculate the diffusion coefficients. However, this method is only tested for particles having Gaussian and Bessel shape distributions, which are only model functions for experimental particle shapes. Hence, the existing PD algorithms fail to perform well, have limited testing in the presence of flow or have limited testing when the particles are defocused.

In the particle-based imaging community, the particle shapes are commonly assumed to be Gaussian. However, this assumption fails drastically when dealing with microfluidic set-ups under high magnifications. This is because the typical depth of microfluidic chips is of the order of 100μm, and it is challenging to create and align thinner light sheets. Hence, the measurement volumes typically have global illumination. Additionally, the high magnification causes a narrow depth of field. These effects combined lead to particles that are outside the depth of field but are still highly illuminated, which appear with an additional ring when images are taken because of the defocused location. Many recent works have focused on building algorithms to locate the three-dimensional position of these particles (Barnkob & Rossi 2021; Barnkob *et al.,* 2015; Rossi & Barnkob 2020). However, these methods typically have low information extraction ability and only work with a low particle concentration. Additionally, these methods only predict particle location and do not directly predict the diffusion coefficient, especially in the presence of flow.

Many recent works have utilised deep learning (DL) algorithms as an alternative to conventional algorithms (Barnkob *et al.,* 2021; Cai *et al.,* 2019; Newby *et al.,* 2018; Rabault *et al.,* 2017; Sardana & Wereley 2023; Zhong *et al.,* 2018). These works primarily use convolutional neural networks (CNNs), a family of DL algorithms. The CNNs are multi-layered neural networks, where each layer contains n-dimensional tensors that can extract spatio-temporal information in the particle images. Deep learning algorithms such as CNNs do not make any assumptions about the type of fluid flow or the shape of the particles and learn to solve the problem by training on large amounts of well-labelled datasets. When these deep learning algorithms are used for PD applications, they are called deep particle diffusometry (DPD) (Sardana & Wereley 2023). Even though the networks theoretically can extract more complex

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

data representations, the existing DPD method was only developed for Gaussian particles, limiting their applicability to real-world scenarios.

A big challenge in developing DL algorithms for particle-based imaging applications is the need for large, diverse, well-labelled datasets. Experimentally acquiring large labelled datasets is a very time consuming and error-prone task. The experimental conditions, such as fluid temperature and illumination, can differ over the experiments, creating unwanted biases in the dataset. Hence, it is a common practice to simulate (Barnkob *et al.,* 2021; Cai *et al.,* 2019; Newby *et al.,* 2018; Sardana & Wereley 2023; Zhong *et al.,* 2018) and synthetically generate (Dreisbach *et al.,* 2022) particle datasets as it is easy to control various experimental conditions. Works such as Mendes *et al.,* (2020) and Rossi (2019) have made it easier to create large datasets for modelling Gaussian and defocused particles. Even though Rossi (2019) can be used for defocused particle modelling, it does not include a change of particle location because of diffusion, limiting its usability for PD applications.

The current work proposes an improved version of regression CNNs proposed by Sardana & Wereley (2023). The previous version of DPD models used temporally averaged frames as inputs, where multiple frames were averaged in a single image before feeding to the CNN, leading to temporal information overlap in a single image. However, when tested on defocused particles in the presence of flow, this leads to poor performance because the underlying data are more complex. The DPD models proposed in the current study were designed to use stacks of frames as the input to the CNNs, hence avoiding temporal information overlap in the input space. This leads to better performing models for both Gaussian and defocused particles. Similar to Sardana & Wereley (2023), the current study uses hyperparameter optimisation and architecture search to obtain better performing models. This study also provides a detailed insight into various factors improving the performance of trained models, such as criteria used for the outer loop deciding the next set of hyperparameters, size of image crop, number of CNN layers, etc. The proposed CNN models were benchmarked against the CNNs from Sardana & Wereley (2023) and four conventional methods. As the current work improves the performance of the original DPD models, these are referred to as DPD-v2 for easier differentiation. Additionally, inference runs of the CNN models were made on experimental data with pure diffusion, and the predictions were compared against predictions from conventional methods.

The rest of the paper is organised as follows. Section 2 details the different types of simulated and experimentally generated and used in this work. Section 3 provides an overview of DPD, details on the DPD-v2 architecture, the training and testing process and details on hyperparameter search. Section 4 provides the failure of DPD-v1 models on more defocused datasets. Next, it provides the impact of various architectural and training workflow choices and ways to maximise performance for particle datasets. Next, the section provides the impact of particle concentration and noise on the performance of the trained models. Additionally, the section provides a benchmark against conventional methods and results for operating the networks on experimental data. Finally, section 5 concludes the work and provides future direction.

2. Creating datasets

Deep learning algorithms learn to solve a problem by minimising a generic loss function on large, labelled datasets designed for that task. These algorithms are often trained in a supervised fashion to learn how to map the input space and its corresponding label. For particle-based imaging applications, the input space is often a single frame (Barnkob & Rossi 2021) or a sequence of frames (Cai *et al.,* 2019; Sardana & Wereley 2023), and the labels can represent information about the fluid or particles such as particle locations (Barnkob & Rossi 2021), particle diffusion (Sardana & Wereley 2023) or flow velocity (Cai *et al.,* 2019).

As this work focused on developing CNN models that can work with any particle shape, two simulated datasets were created with different particle shapes: Gaussian and defocused. These datasets were created using MATLAB. The dataset with Gaussian particles had no motion in the out-of-plane direction,

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

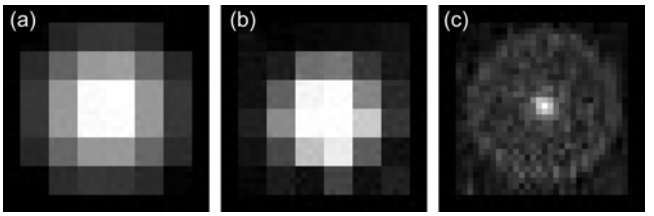


Figure 1. *Particle shapes obtained from Gaussian and ray-tracing simulations: (a) particle from Gaussian simulation, (b) In-focus particle obtained from the ray-tracing simulation, (c) out-of-focus particle at a distance of 20μm from the focal plane obtained from the ray-tracing simulation. The images are zoomed in to improve visualisation.*

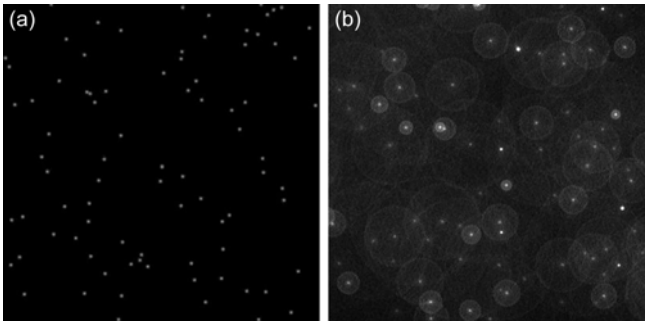


Figure 2. *The 400 px × 400 px crops of images with (a) Gaussian particles and (b) defocused particles.*

and the particle simulated diameter remained 6 px. However, for the defocused case, the particle shape and simulated diameter depend on multiple factors, including the particle distance from the focal plane (Rossi 2019). As this distance changes during diffusion, the particle shapes and diameters also change per frame and were simulated using a ray tracing approach (Rossi 2019). The particles had a diameter of 400nm, and the maximum depth of the measurement volume was set to be 100μm. The lens system had a focal length of 350μm, a numerical aperture of 0.65 and a 40x magnification. The lens and the medium had refractive indices of 1.5 and 1, respectively. The pixel size was set to 7.4μm with 20 points per pixel and 500 rays per point. The gain was set to 1, and the cylindrical focal length was 0 to avoid astigmatism. Figure 1 shows the particle shapes obtained from Gaussian and ray tracking defocus simulations.

As diffusion has a temporal component, each data point in datasets developed and used in the current study had a sequence of 101 single-channel frames. These frames were grey scale, as colour information is irrelevant for diffusion coefficient prediction. Each frame was 1024 px × 1024 px and had 600 particles without background noise unless otherwise specified. The particles were randomly seeded in the first frame, and their location in the next frames depended on their current location, flow and diffusion coefficient values. More details about the process can be found in Sardana & Wereley (2023). Figure 2 shows 400 px × 400 px crops of images with Gaussian and defocused particles.

For both datasets, the particle position changes because of both flow and diffusion. The diffusion coefficients varied from 0.3 to 3$\mu m^2 s^{-1}$ with a 0.01$\mu m^2 s^{-1}$ step size, leading to 271 different possibilities. These diffusion conditions were superimposed with one of the four flow conditions: no flow (pure diffusion), Poiseuille flow, Couette flow and uniform flow. The maximum flow velocity was 2ms^{-1} leading to a Péclet numbers (*Pe*) number ranging from 0 to 2.67 $\times 10^6$ and the flow was from left to right. The four possible flows and 271 diffusion coefficient values for each flow lead to 1084 sequences, referred to as a simulation batch. For each particle shape, three simulation batches were created. Adding more simulation batches increases the training time but does not significantly improve the performance (Sardana & Wereley 2023). The particles were randomly seeded in each simulation batch. In all the experiments,

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

the first simulation batch was used for training, the second for validation and the third for testing and benchmarking the CNN models against other algorithms. As a minor contribution, this work combines the defocused particle-image generation with particle diffusion and fluid flow workflows to create the defocused particle diffusion under arbitrary flow datasets.

The study also includes inference of the trained CNNs on experimental data under the no flow condition (Lee *et al.,* 2022). The experiments were done at 15 frames per second with 500nm particles under 40x magnification. The camera's focal length was 4.2 mm, but the focal length of the entire optical system was unknown. The first 3 minutes of the experiments were used, and the videos were subsampled to obtain multiple effective diffusion coefficient values (Sardana & Wereley 2023). The outputs from CNN models were compared against the outputs from the conventional algorithms to obtain prediction differences with the experimental datasets. It is important to note that the experimental datasets and the defocused simulations differed in different parameters. As many microscope and experimental parameters are unknown, this simulated defocused dataset was used to prove that CNNs can extract interesting fluid properties such as diffusion from defocused particles. This study does not focus on fine tuning these defocused datasets and the models trained on those for a single experimental set-up. Hence, the experimental dataset was Gaussianised before feeding these images to the CNNs. This was done by identifying individual particle locations in a frame and putting Gaussian particles in a new image.

3. Deep particle diffusometry-v2

Deep particle diffusometry is a method of predicting diffusion coefficient from a sequence of particle images (Sardana & Wereley 2023). The technique uses a sequence of images as input without identifying individual particles. This sequence of images is transformed into a diffusion coefficient by passing it through a multi-layered CNN. The CNN learns this transformation in a supervised fashion during the training process, during which it learns to map the sequence of images to a given diffusion coefficient value.

As diffusion is a spatio-temporal process, the DPD algorithms must process information from multiple frames. Hence, alternative ways, such as three-dimensional convolutions (Zolfaghari *et al.,* 2018), CNNs with Long Short-Term Memory (LSTM) (Ullah *et al.,* 2017) and hybrid CNN approaches (Tran *et al.,* 2018), can also be applied to process these data. The input sequence of frames can also be converted to a single frame by temporally averaging the frames before feeding to the network (Sardana & Wereley 2023). This approach works well when the input space is sparse, such as when Gaussian particles are used. Alternatively, the frames can be fed individually to avoid any information loss before feeding the data to the CNNs and the network can extract the useful information.

To avoid this information loss associated with the existing DPD models, this work introduces a CNN-based PD method that can use multiple frames as inputs. This is done by changing the input layer of the architecture to account for multiple frames while keeping the same output. This provides the ability to train the network on an arbitrarily high number of images without significantly increasing the computational cost. The current models were trained and tested on up to 64 frames and it is possible to train them for an even higher number of frames. To get better generalisation for PD applications, the new models were trained and tested on simulated Gaussian and defocused datasets to avoid any narrow results on a single dataset. As this work improves the performance of existing DPD models, the proposed models are also referred to as DPD-v2 models for ease of differentiating between the two.

3.1. Networks created

The CNN architectures in the current work are inspired by ResNet (He *et al.,* 2016) and DPD (Sardana & Wereley 2023). For DPD-v2, the first layer was a multi-channel two-dimensional convolution, where the number of channels depended on the number of input frames. The output of this layer was fed to n-residual blocks with two sequential convolution layers and a skip connection. This allowed the network

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press



Figure 3. *Main components of the DPD-v2 architecture: the first convolution layer (conv), n-repeating residual blocks (each with two convolution layers), an average pool layer and a fc layer.*

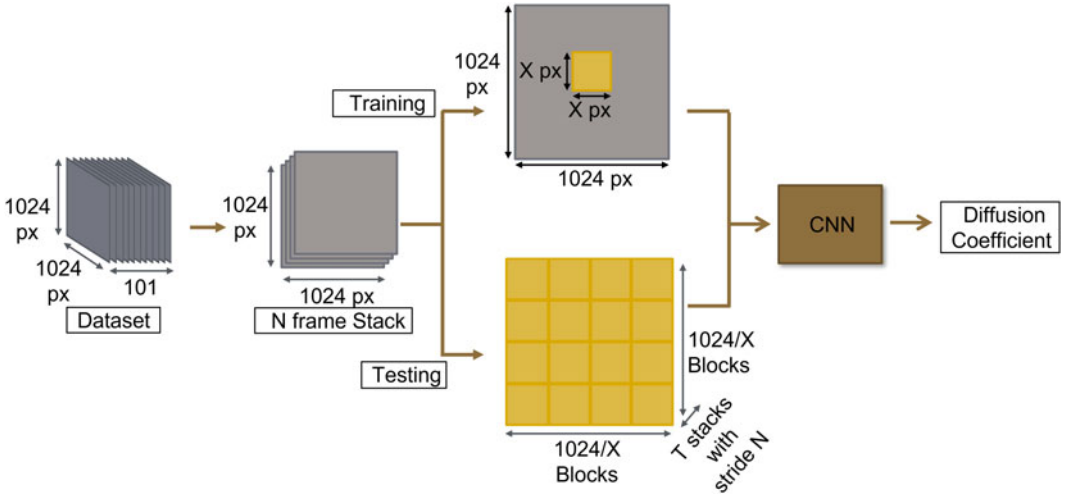


Figure 4. *Data flow during training and testing.*

to have a deeper structure and extract complex information from the underlying image stack. All the convolution layers were followed by batch normalisation and rectified linear units. All the convolution filters in the residual blocks had a kernel size 3. The output of the last convolution layer was fed to an average pool layer that provides a low-dimensional representation using the features in the fully connected (fc) layer. The effect of the size of this pool is explored in the current work. The DPD-v1 architectures had an additional max-pool layer before the first convolution layer. However, this layer was removed from DPD-v2 to reduce the information loss before the convolution layers. The last layer in the CNN architectures was the fc layer, which had a single output representing the diffusion coefficient. The number of neurons in the input of this layer depended on various factors, such as the input image crop size, the size of the first convolution layer and the size of the average pool. This number was automatically adjusted based on the other architectural choices. Figure 3 shows the main components of the DPD-v2 architecture.

3.2. Network training, testing and hyperparameter search

The CNNs in the current work performed a video regression task. The networks were trained in a supervised fashion to predict diffusion coefficients from an N-frame stack as input. The CNNs were primarily trained and tested on 256 px $\times$ 256 px crops obtained from four-frame stacks. Sizes smaller than this led to cases where there were no particles in the crops for the Gaussian particle dataset. During training and validation, these crops were made on random spatial locations to promote generalisation, whereas, during testing, the entire stack was broken down into smaller crops to be data efficient and remove spatial stochasticity. During testing, the temporal stochasticity was reduced using multiple stacks with a temporal stride (Sardana & Wereley 2023). The data flow is shown in figure 4. The current CNN architecture provides the flexibility to be trained and tested on crop size (X) and stack size (N).

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

Table 1. *Exhaustive hyperparameter and architectural parameter space used for the joint search of training configuration. Some experiments reduced this space to fit the large models on the GPUs and avoid training instabilities*

Search parameter	Search type	Range	Initial value
Convolutional layers	Choice	7, 13, 18, 25, 31, 37, 43, 49, 97	13
First convolution layer size	Choice	3, 5, 7	5
Kernel size of average pool	Choice	1, 2, 4, 8, 16, 32, 64, 128, 256	32
Learning rate	Log uniform	[1e-6 – 1e-3]	1e-4
Batch size	Choice	2, 4, 8, 16, 32	8

Data augmentation of the image stacks further promoted the generalisation of the networks. These include random vertical and horizontal flips and rotations. It is important to note that all the augmentations were applied jointly to the entire stack to avoid any unreal changes in spatial information within the stack.

A big challenge while training a neural network is to figure out a good set of hyperparameters. However, this set of hyperparameters is highly dependent on the network architecture and the datasets it is being trained on. So, a hyperparameter search is required for the current task. Additionally, a good network architecture further depends on the underlying datasets. As the original ResNet architectures were not developed for particle images, their architectural choices (such as the number of layers) are not an ideal starting point for particle image datasets. Hence, a neural architecture search is also required. These two search problems can be simplified to a single search problem by considering neural architecture search as a hyperparameter search task and performing a joint search. For each experiment, 500 different combinations were used. The exhaustive hyperparameters and architectural parameter space used for the joint search are provided in table 1. Some experiments reduced this space to fit the large models on the GPUs and avoid training instabilities.

As it is unknown what network depth can extract the features from the particle images better, the search was performed for different network depths. The deepest networks (97 layers) had roughly 14x more convolutional layers than the shallowest ones (7 layers). Increasing the network depth increases the spatial information the network sees before making the final decision, but this also increases the number of model parameters, making the model relatively harder to train. The kernel size of the first convolution layers controls the initial spatial receptive field of the network. A higher initial receptive field provides more spatial information to the CNN, which can lead to shallower networks. However, it can also make information extraction more difficult. The kernel size of the average pool controls the amount of information that can be averaged out without degrading the network performance. The larger this kernel, the smaller the network can be obtained, but the network would need to use a smaller latent space to make accurate predictions. The learning rate (LR) is one of the most important hyperparameters. It controls the amount of change in a network parameter during the backpropagation process. If LR is too small, the learning can get stuck to a local minimum. If LR is too large, the learning can overshoot the optimal point. Another important hyperparameter during the backpropagation is batch size. It defines the number of samples the network sees per iteration. If the batch size is too small, the underlying loss space will be too noisy, leading to a difficult convergence. However, if the batch size is too large, it will lead to the models overfitting to the training dataset. The space was searched with the tree-structured Parzen estimator approach (Bergstra *et al.,* 2011) in a Bayesian manner.

The hyperparameter search was done to optimise the performance of the models on the validation datasets. This work compares various criteria: maximising R^2, minimising the mean absolute error (MAE), minimising the mean square error (MSE) and minimising the mean root absolute error (MRAE). The validation dataset was also used to provide an early stop during training and to select the best-performing models. In the inner loop, the MSE (L2 loss) between the true and predicted diffusion coefficients was primarily used as the loss function. This work explores the MAE (L1 loss) as the

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

Table 2. *Mean absolute error ($\mu m^2 s^{-1}$) for DPD-v1 and DPD-v2 models on Gaussian and defocused test datasets*

Dataset	DPD-v1	DPD-v2
Gaussian	0.24	0.12
Defocused	0.44	0.10

alternative loss function. Both loss functions are distance metrics, and CNN's task was to minimise this distance. Stochastic gradient descent with momentum was used as the optimiser. As the DPD-v2 models are trained for a particle-image application, these can serve as better pre-trained checkpoints for other particle-image applications such as Particle Image Velocimetry (PIV).

4. Results and discussion

This study provides CNN architectures and training workflows specialised for particle datasets. This is achieved by performing the architecture and hyperparameter search on image sequences containing defocused and Gaussian-shaped particles. This work also analyses the impact of different loss functions during backpropagation. Next, the impact of different architectural choices, such as the network depth and pool size, is presented. The study also contains CNNs trained on different crop sizes and different numbers of images in a stack, hence exploring the impact of varying input spatial and temporal information. Next, the performance of the DPD-v2 models was compared against the performance of the DPD-v1 models and four conventional methods. Finally, the outputs from DPD-v2 models and conventional methods are compared for experimental data, and the difference between the two sets of predictions is presented.

All the tests were made on the test dataset, which was never used for training or validation. The tests had 256x256 crops from the image stacks unless specified otherwise. The performance is compared with the MAE of the best model obtained after training 500 different hyperparameters and architectural combinations. The MAE was used as the evaluation matrix because it provides the results in the same units as diffusion coefficients, making the evaluation physically meaningful. As all the numbers are averaged in the original units, it is a more reliable unit than RMSE. The spatio-temporal stochasticity of the predictions is reduced by using five 4-frame stacks unless specified otherwise. This leads to 20 frames used during testing. While comparing different models, an MAE above $0.03 \mu m^2 s^{-1}$ was considered as a significant difference. This is 10 % of the minimum diffusion coefficient used in this study.

4.1. Failure of temporal averaging

The previous version of DPD models (DPD-v1) used temporally averaged images as input. While this data modality can effectively represent the multi-frame problem within a single frame, it remained uncertain whether the performance of these models would degrade when handling complex data, such as defocused particles. The proposed DPD models (DPD-v2) can use multiple frames as inputs and hence have more information to process. Table 2 compares the performance of DPD-v1 and DPD-v2 models on Gaussian and defocused datasets.

It can be seen that DPD-v2 models provide 2x–4x less MAE when compared with DPD-v1 models. The DPD-v1 models perform the worst when the particles are defocused, and flow is present along with diffusion. As the only difference here is the input data modality, it can be seen that temporal averaging fails when the datasets get complex.

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

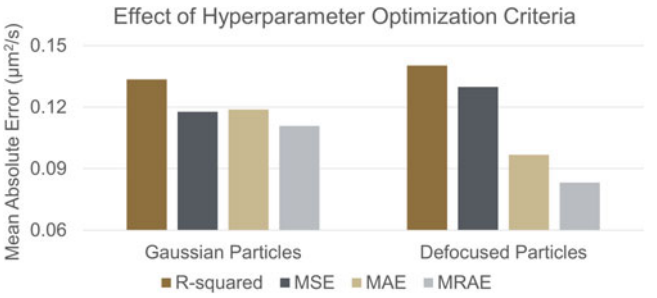


Figure 5. *Mean absolute errors ($\mu m^2 s^{-1}$) obtained from the best DPD-v2 models obtained using the following hyperparameter optimisation criteria: maximising the R^2, minimising the MSE, minimising the MAE and minimising the MRAE.*

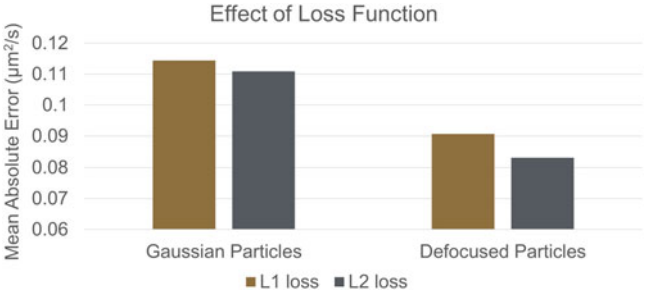


Figure 6. *Mean absolute errors ($\mu m^2 s^{-1}$) obtained from the best DPD-v2 models trained with the L1 and L2 loss functions.*

4.2. Hyperparameter optimisation criterion

Hyperparameter optimisation is one of the primary components in this work that provides state-of-the-art CNNs. These hyperparameters were selected in a Bayesian manner to improve the performance according to the hyperparameter optimisation criteria. This study compares four criteria between the true and predicted diffusion coefficients: maximising the R^2, minimising MSE, minimising MAE and minimising the MRAE. Figure 5 compares the MAE obtained from the best models when the four criteria were used for hyperparameter optimisation.

It can be seen that the choice of optimisation criterion changes the performance of the trained models regardless of the dataset type; R^2 was the worst criterion in both cases. It is because the error terms are squared with R^2, which decreases the magnitude of the error when the error is less than 1. The smaller the magnitude is, the harder it is to differentiate between two models. This also makes MSE a bad criterion. The MAE does not impact the error's magnitude, leading to higher-performing models. Finally, a good hyperparameter optimisation criterion will encourage large magnitudes for small changes in error. As the magnitudes of the MAEs are less than 1, the root of the absolute errors will lead to higher magnitudes. Hence, the model obtained using MRAE performs the best. This performance can be further increased by using higher-order roots.

4.3. Comparing loss functions

The choice of the loss function in the inner loop also impacts the performance of the trained models. The choice of loss function defines the underlying loss landscape. Figure 6 compares the MAEs obtained using the L1 and L2 loss functions.

It can be seen that the models trained with the L2 loss function perform better than models trained with the L1 loss function. This is because the gradient of L2 loss is continuous, which makes the

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

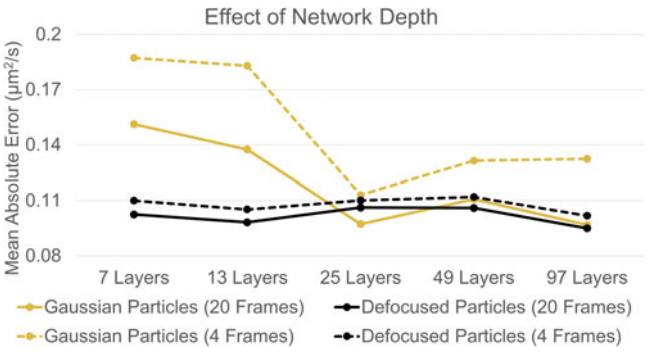


Figure 7. *Mean absolute errors ($\mu m^2 s^{-1}$) obtained for DPD-v2 models with different depths. These models were tested on 1 and 5 stacks of images, leading to 4-frame and 20-frame results, respectively.*

convergence process easier. Even though the performance gap is insignificant, L2 loss is a better choice. Similar results were obtained for DPD-v1 models.

4.4. Optimising the architecture

One major factor influencing the performance of CNNs is their architecture. The two main aspects of the architectural choice in this work were the depth of the CNN and the pool size. In the current experiment, these architectural parameters were fixed during the hyperparameter search to test their performance. First, the impact of network depth was tested. Figure 7 shows the performance of DPD-v2 models with different network depths. These models were tested on 1 and 5 stacks of images, leading to 4-frame and 20-frame results, respectively.

It can be seen that the network's performance depends on the number of layers used. The shallower networks have a higher loss for Gaussian particles than the deeper networks. Meanwhile, for defocused particles, the network depth does not drastically affect the performance. This shows that relatively shallow networks can efficiently extract the diffusion coefficients from the complex but information-rich defocused particles. Moreover, unlike the conventional algorithms, networks trained on defocused particles overall perform better than those trained on Gaussian particles, highlighting the capability of neural networks to effectively process complex information. The networks trained on defocused particles also perform well with only a single stack of images (4 frames). Even though the 97 layered networks have the best performance, it is less than 1 % better than the second-best networks in both categories, and the training time increases by at least 3x. The second-best networks had 25 layers for models trained for Gaussian particles and 25–49 layers for models trained for defocused particles. A maximum network depth of 49 layers was used in the rest of the experiments so these experiments can use deeper networks if needed without significantly increasing the computation costs. Next, the impact of the average pool kernel size was evaluated. Figure 8 shows the performance of DPD-v2 models with different average pool kernel sizes.

It can be seen that the performance of models trained on defocused particles improves with larger pool sizes. Meanwhile, for models trained on Gaussian particles, increasing the pool size increases performance to a certain point. The performance was best for pool sizes between 16 and 64, used in further experiments. A larger pool size helps reduce the trainable parameters in the fc layer, showing that the CNNs can extract the diffusion coefficient information with fewer parameters (hence, more efficiently) when the particles are defocused.

Optimising the CNN architecture and training workflow helped achieve the best-performing DPD-v2 models. Figure 9 shows the performance of the best DPD-v2 models trained on the defocused dataset.

It can be seen that the model has a good overall performance with a small degree of under-prediction for higher diffusion coefficients. It is because diffusion is a stochastic process and the particles can

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

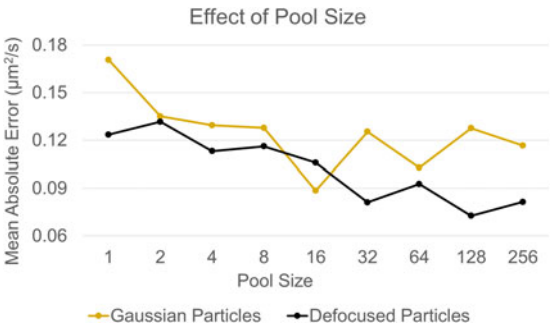


Figure 8. *Mean absolute errors ($\mu m^2 s^{-1}$) obtained for DPD-v2 models with different average pool kernel sizes.*

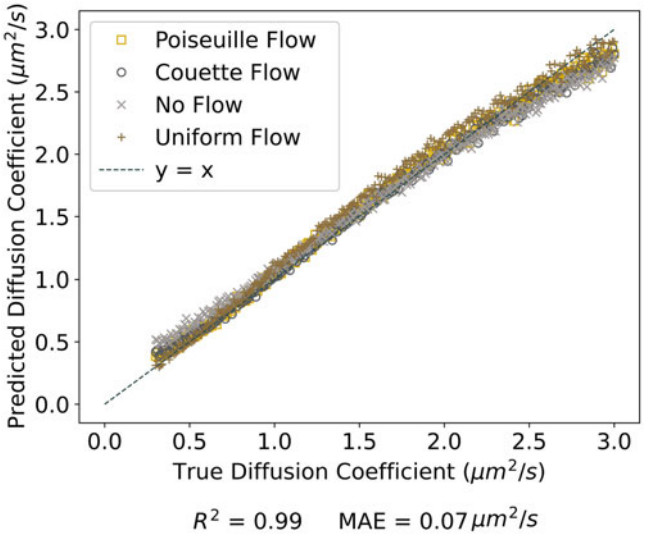


Figure 9. *Performance of best DPD-v2 models trained on defocused dataset.*

show a smaller motion even if the diffusion coefficient is high. Hence, the algorithms need more spatial and temporal information to predict better the diffusion coefficient, which would reduce the spatial and temporal resolution of the algorithms. The best model trained on Gaussian particles had slightly higher MAE ($0.09 \mu m^2 s^{-1}$) and lower R^2 (0.98) values, which are still quite similar to defocused particles. Next, table 3 shows the effect of different flows on the performance of the models trained for Gaussian and defocused particles.

It can be seen that the models' performance is better on flows with a gradient (Couette and Poiseuille flows) than on flows without a gradient (uniform and no flows). This is because both flows with a gradient produce similar motion patterns and are, hence, overrepresented in the dataset. Conversely, flows without a gradient produce different motion patterns and are, therefore, seen less frequently by the models during training. Still, the MAEs produced for each flow are significantly smaller than the least value of diffusion coefficients used in this work.

4.5. Impact of crop sizes

The performance of the CNNs can be affected by the amount of spatial information provided. For diffusometry, a higher amount of spatial information leads to more particles, reducing the problem's inherent

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

Table 3. *Effect of different flows on the performance of the models trained for Gaussian and defocused particles*

Particle shape	Gaussian		Defocused	
Flow	Mean absolute error ($\mu m^2 s^{-1}$)	R^2	Mean absolute error ($\mu m^2 s^{-1}$)	R^2
Couette flow	0.08	0.99	0.06	0.99
No flow	0.08	0.99	0.09	0.99
Poiseuille flow	0.07	0.98	0.06	0.99
Uniform flow	0.12	0.99	0.08	0.99

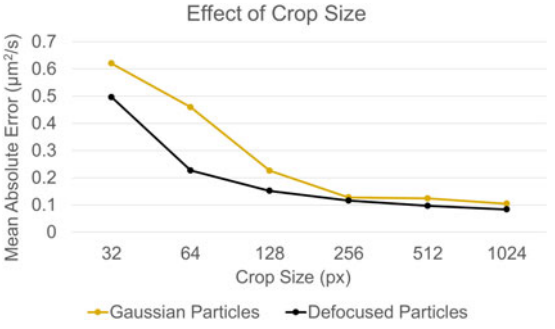


Figure 10. *Mean absolute errors ($\mu m^2 s^{-1}$) of DPD-v2 models trained on different crop sizes. In all the cases, the same amount of spatial information by using multiple crops for smaller crop sizes.*

stochasticity. The spatial stochasticity can be reduced during testing using multiple crops of any given size. It can also be reduced by training the networks on bigger crops. Figure 10 compares the performance of DPD-v2 models trained on different crop sizes. During testing, the models used all the spatial information present by breaking down the 1024x1024 images into smaller crops, feeding the individual crops to the models to obtain diffusion coefficient, and averaging these values to get a single diffusion coefficient. Five 4-frame stacks were used in all the cases.

It can be seen that the performance of the models increases with larger crop sizes. For both particle shapes, the performance improvement is higher for the smaller window sizes. This is because the small window size reduces the amount of spatial information in the data, increasing the prediction error. This can be tackled by increasing the particle concentration and increasing the spatial information in the dataset. For Gaussian particles, the smaller crop sizes can even lead to crops with no particles, contributing to the lower performance when the crops were smaller than 256 px × 256 px. Even though the performance at these smaller sizes can be improved by removing crops with no information, models trained on larger crops will still perform better as in the later part of the figure. The figure also shows the neural networks can better extract information rich for defocused particles, hence providing lower MAE with all crop sizes compared with Gaussian particles.

Training a model on larger crop sizes is also challenging because it leads to larger models that are difficult to fit on a GPU, increasing the training time. In this work, 8 models were trained in parallel for crops 256 px × 256 px and smaller, and the training time for 500 hyperparameter configurations was under 1 day. However, only a single model can be fit on the GPU for 1024 px × 1024 px, which took around 10 days to train for the 500 hyperparameter configurations.

4.6. Impact of stack size

The performance of the CNNs can also be affected by the amount of temporal information provided. A higher amount of temporal information leads to longer particle trajectories, reducing the problem's

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

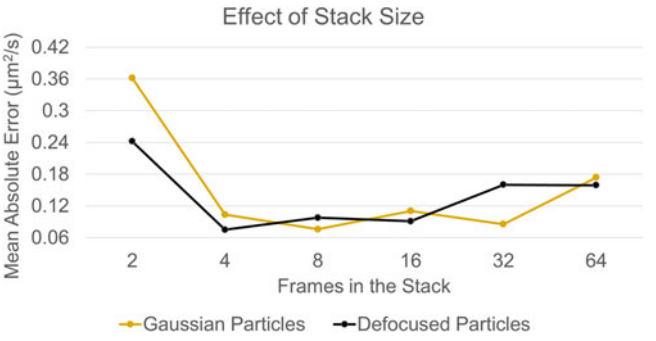


Figure 11. *Mean absolute errors ($\mu m^2 s^{-1}$) of DPD-v2 models trained on different numbers of frames in the stack. In all cases, the same amount of temporal information is used by using multiple stacks for smaller stack sizes.*

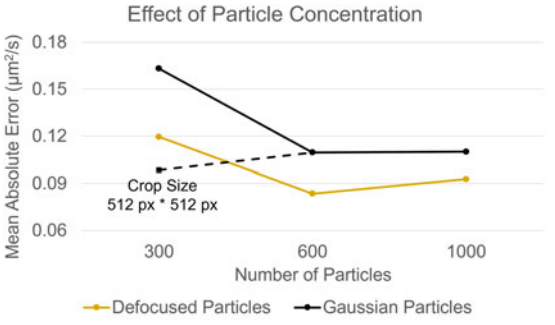


Figure 12. *Mean absolute errors ($\mu m^2 s^{-1}$) of DPD-v2 models trained on datasets with varying particle concentrations: 300, 600 and 1000 particles per frame. The models were trained and tested with a crop size of 256 × 256 pixels. A crop size of 512 × 512 pixels was also used for the 300-particle Gaussian case to avoid crops with no particles.*

inherent stochasticity. The temporal stochasticity can be reduced during testing using multiple stacks of images and averaging the diffusion coefficient to get a single value. It can also be reduced by training the networks on bigger stacks, as long as the networks can extract information from longer temporal sequences. Figure 11 compares the performance of DPD-v2 models trained on different numbers of frames in the stack. The experiment was conducted with 64 frames. The largest stack contained 64 frames, and the multiple stacks were used during testing when the stack size was less than 64. The models were trained on crops of size 256 px × 256 px to save training time while maintaining high performance.

It can be seen that the networks performed the worst when two frames were used. The particles in the underlying datasets move because of flow and diffusion, making it difficult to extract this information from two frames. The networks have a good performance when 4 to 16 frames are used. At higher frames per stack, it gets difficult for the current DPD-v2 architectures to extract useful information. High temporal information can still be included by using multiple smaller stacks. Models trained on 4 frames in the stack provide a good performance and a temporal resolution.

4.7. Impact of particle concentration

The performance of deep learning models can also be affected by experimental factors such as the concentration of particles. A higher concentration of particles can result in increased overlap, leading to

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

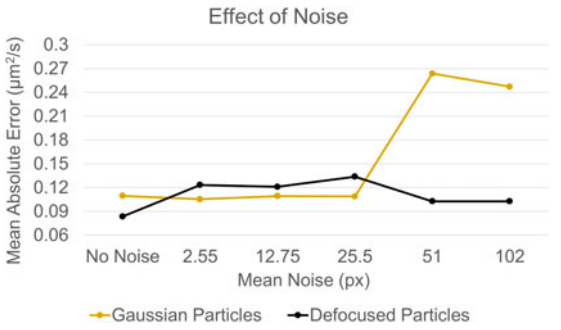


Figure 13. *Mean absolute errors ($\mu m^2 s^{-1}$) of DPD-v2 models trained on datasets with varying levels of Gaussian noise. The standard deviation was set to 25.5 pixels.*

challenges in distinguishing between the particles, whereas a lower concentration may result in insufficient information for the stochastic diffusion process. To evaluate the impact of particle concentration, DPD-v2 models were trained and tested on datasets with particle concentrations of 300, 600 and 1000 particles per frame for both particle shapes. The models were trained and tested with a crop size of 256 × 256 pixels. A crop size of 512 × 512 pixels was also used for the 300-particle Gaussian case to avoid crops with no particles. Figure 13 shows the results.

It can be seen that the performance of DPD-v2 models degrades at lower particle concentrations. This is due to an increased likelihood of crops containing few or no particles, particularly in the case of Gaussian particles. The figure also shows that using a larger crop size mitigates this issue by reducing the probability of low-information crops, thereby improving model performance.

4.8. Impact of noise

Another factor impacting the performance of CNNs is the presence of noise in the images. Noise reduces the quality of the acquired image by introducing random variations in pixel intensities, making it challenging for the models to extract meaningful information. To evaluate this effect, DPD-v2 models were trained and tested on datasets with varying levels of Gaussian noise with mean 2.55, 12.75, 25.5, 51 and 102 pixels and a standard deviation of 25.5 pixels for the 8-bit (256-pixel level) images. Figure 13 compares the performance of these models in terms of MAEs ($\mu m^2 s^{-1}$).

It can be seen that the DPD-v2 models trained on Gaussian particles perform well with no and low noise levels (under 51 pixels). However, their performance degrades at higher noise levels. In contrast, DPD-v2 models trained on defocused particles maintain consistent performance across the tested noise levels, demonstrating their robustness and ability to generalise despite the noise.

4.9. Benchmarking against other methods

The performance of DPD-v2 models was compared against the performance of DPD-v1 and four conventional methods. The implementations of Ahmadzadegan *et al.,* (2020), Crocker & Grier (1996) and Olsen & Adrian (2000) were referred to as iPED, trackPy and Olsen–Adrian, respectively. The correlation-based alternative of trackPy was referred to as correlation-EMSD. These algorithms were compared for both particle shapes with 20 frames. The DPD-v1 and DPD-v2 models were tested on 256 px × 256 px crops. Figure 14 compares the performance of various methods on Gaussian and defocused datasets under the no noise condition.

It can be seen that the DPD-v2 algorithms provide the least error for both particle shapes; iPED and DPD-v1 were the two next-best algorithms with similar performances. All the other algorithms have high MAE at least for defocused datasets. The MAE for trackPy on defocused particles was so large

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

Table 4. *Mean absolute difference ($\mu m^2 s^{-1}$) between the outputs from DPD-v2 models and outputs from trackPy and iPED on Gaussianised experimental datasets*

Concentration (mg/ml)	trackPy	iPED
20	0.12	0.48
15	0.09	0.41
15	0.14	0.29

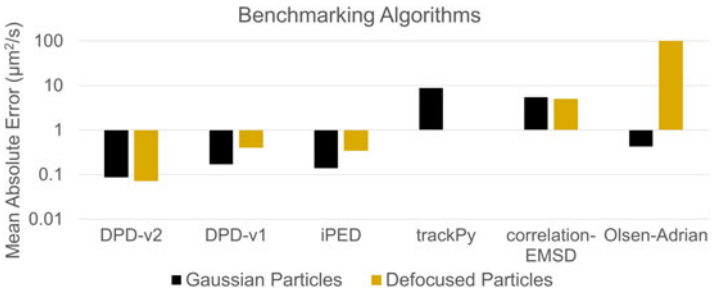


Figure 14. *Mean absolute errors ($\mu m^2 s^{-1}$) of various methods on Gaussian and defocused datasets. The experiments were done with 20 frames.*

that it is not even included in the figure; iPED and DPD-v1 have also performed better for the Gaussian datasets because these are simpler representations. Due to the DPD-v2 models' ability to extract diffusion coefficients from information-rich data, they perform slightly better on defocused particles, where changes in defocus during diffusion provide additional information.

4.10 Evaluation on experimental data

Finally, the performance of the DPD-v2 models was evaluated using experimental datasets, which lack ground truth values for diffusion coefficients. So, the outputs of DPD-v2 models were compared against the outputs of two conventional methods, and the mean absolute difference (MAD) was calculated. It is important to note that this evaluation does not assume DPD-v2 is a better estimator. Rather, it examines how closely its estimates align with those of existing models on experimental data. Table 4 shows these results.

It can be seen the MAD between trackPy and DPD-v2 models on the experimental datasets was comparable to the MAEs observed on simulated datasets, indicating consistent performance. In contrast, the MAD between the iPED and DPD-v2 models was four times higher but remained within the lower range of regression values, suggesting reasonable agreement.

5. Conclusion and future work

Predicting physical properties, such as the diffusion coefficient, from particle images in a microfluidic set-up is a challenging task. This is because the particles are often defocused and do not follow a Gaussian shape. The motion of these particles is often dependent on multiple factors such as diffusion and flow and it is a hard task to isolate the underlying phenomenon from particle motion. Conventional algorithms are often developed with many assumptions and cannot be used in a generalised microfluidic setting. Deep learning can be used as an alternative. However, the performance of deep learning algorithms can be severely affected by their architecture, the way these are trained, and how the input data are fed. The current work optimises the performance of DPD-v1 models on these factors to produce

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

state-of-the-art PD models named DPD-v2. These models were trained and tested on Gaussian and defocused datasets to ensure that the results are not limited to a single dataset type.

The study shows the failure of the temporal averaging input modality used in DPD-v1 models when the datasets are complex. This failure was removed in DPD-v2 models by using image stacks in the input which had 2x–4x less MAE than DPD-v1 models. Next, the work shows the impact of the hyperparameter optimisation criterion, the loss function and architectural choices to optimise the training process. The performance can be improved by using the L2 loss function for the inner loop and MRAE criterion for the hyperparameter optimisation. Twenty-five CNN layers were enough for both particle shapes to create good-performing models with fast training times. This led to MAEs of $0.07\mu m^2 s^{-1}$ for defocused particles and $0.09\mu m^2 s^{-1}$ for Gaussian particles. The work also shows the impact of changing crop sizes and the number of frames in the input stack. The networks perform better with larger crop sizes and can extract information efficiently with 4–16 frames in the stack. Next, the performance of DPD-v2 models was tested for various particle concentrations. The results show a decline in performance at lower concentrations due to an increased likelihood of crops with low or no particle information. However, this issue can be resolved by increasing the crop size. Additionally, the performance of DPD-v2 models was tested under varying noise levels. The results show that the models trained on defocused particles are robust against noise, while the performance of models trained on Gaussian particles degrades at higher noise levels.

The DPD-v2 models were benchmarked against DPD-v1 models and four conventional methods on both particle shapes in the simulated datasets. The DPD-v2 models had 2x–4x lower errors than the next two best-performing models (DPD-v1 and iPED). Finally, the outputs of the DPD-v2 models were compared against the outputs of iPED and trackPy with Gaussianised experimental images under a no flow condition, which provided mean absolute differences with magnitude 1x–4x compared with the MAEs in the simulated cases.

Even though the current CNNs can account for various numbers of frames in the stack, their performance decreases when the stack size increases above 16 frames. The temporal information can be better accounted for by using a hybrid CNN-Recurrent Neural Network (RNN) architecture. The current workflow also does not have a way to reject inputs with no particles or low-information content which can be included in future work. Finally, the experimental evaluation of the current models depends on the Gaussianising of the datasets. The current Gaussianising algorithm can only detect particles closer to the focal plane. Even though algorithms such as that of Ratz *et al.,* (2023) can detect defocused particles, these are developed on astigmatic datasets that do not work well for the current case. Similar algorithms can be created for defocused particles to improve the Gaussianising process.

Data availability. The DPD-v2 code can be found here: https://github.itap.purdue.edu/psardana/DPD-v2. The sample datasets can be visualised here: https://engineering.purdue.edu/microfluidics/DPDWebsite.html. The full dataset with Gaussian particle shape can be found here: https://osf.io/j2xwa/, and with defocused particle shapes can be found here: https://osf.io/qych9/. The code to generate the defocused datasets can be found here: https://github.itap.purdue.edu/psardana/defocused-PD-generation. The code for DPD-v1 and benchmarking algorithms can be found here: https://github.itap.purdue.edu/psardana/deepParticleDiffusometry.

Funding. This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

Competing interests. The authors declare no conflict of interest.

Appendix A. Distance Metrics

The distance metrics can be used to represent the error in regression problems. Error is defined as the difference between the true and predicted value of the diffusion coefficient. The true value of the diffusion coefficient was a parameter used to create the particle diffusion videos. The predicted value of the diffusion coefficient is an output of the algorithms that use these image sequences as inputs.

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

The MAE is a single term defining the performance of the algorithms and is given by A.1

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \,, \tag{A.1}$$

here, y_i is the true diffusion coefficient, $\hat{y}_i$ is the predicted diffusion coefficient, i is the index of a data point and n is the total number of predictions. Another commonly used way of representing distance is the MSE, which is given by A.2

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \,. \tag{A.2}$$

Even though this is a commonly used metric, it squares the numbers and hence can overestimate the distance when it is greater than 1 and underestimate the error when it is less than 1. In contrast, the root of a number can overestimate the distance when it is less than 1 and underestimate the error when it is greater than 1. This can be used as a distance with MRAE and is given by A.3

$$\mathrm{MRAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^{1/k} \,, \tag{A.3}$$

here, k represents the kth root of the error. When k is 2, it represents the mean square root absolute error. The goodness of fit of the predicted values to the true value can also be given by R^2 and is given by A.4

$$r = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}}_i)}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}}_i)^2}}, \tag{A.4}$$

where r is Pearson's r, $\bar{y}$ is the mean of the true diffusion coefficients and $\bar{\hat{y}}$ is the mean of the predicted diffusion coefficients in the dataset. For experimental data, the real value of the diffusion coefficient is not known. However, we can still evaluate the closeness of outputs of any two algorithms using the MAD, which is given by A.5

$$\mathrm{MAD} = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - \hat{z}_i| \,, \tag{A.5}$$

here, $\hat{y}_i$ and $\hat{z}_i$ are predicted diffusion coefficient from the two algorithms.

Appendix B. Recommendations for End Users

The current work provides many DL models for PD. Even though DL workflows can produce models that can predict diffusion coefficients regardless of any particle shape, this was not the current focus. The current models can make reliable predictions if the domain of the new dataset is included in the domain of the training dataset. Hence, the models need to be finetuned for data from the new optical system to account for the difference in particle shapes. Pre-processing the images to create Gaussian particles is a more robust way that does not require such fine tuning of the trained models. So, the choice of the right model first depends on the particle shapes in the images. Next, the choice of the right model depends on the desired spatial and temporal resolutions, which are user-defined criteria. Larger crop sizes lead to higher performance but lower spatial resolutions. However, more temporal information does not always lead to higher performance as it gets difficult for the current models to extract information from long sequences. Hence, multiple stacks of relatively shorter stack sizes (4–16 frames) should be used for longer video sequences. Finally, as the DPD models are trained for a particle-based application, these models can be used as better checkpoints for other particle-based applications (such as PIV) when compared with other models that are trained for non-particle datasets. An end user should prefer using hyperparameter optimisation while using these models to ensure their performance is not dependent on a non-intuitive user-defined hyperparameter.

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

References

Ahmadzadegan, A., Ardekani, A. M., & Vlachos, P. P. (2020). Estimation of the probability density function of random displacements from images. *Physical Review E*, *102*(3), 033305.

Allan, D. B., Caswell, T., Keim, N. C., van der Wel, C. M., & Verweij, R. W. (2023). soft-matter/trackpy: v0.6.1 (v0.6.1). Zenodo. https://doi.org/10.5281/zenodo.7670439

Barnkob, R., Cierpka, C., Chen, M., Sachs, S., Mader, P., & Rossi, M. (2021). Defocus particle tracking: A comparison of methods based on model functions, cross-correlation, and neural networks. *Measurement Science and Technology*, *32*(9), 094011.

Barnkob, R., Kähler, C. J., & Rossi, M. (2015). General defocusing particle tracking. *Lab on a Chip*, *15*(17), 3556–3560.

Barnkob, R., & Rossi, M. (2021). Defocustracker: A modular toolbox for defocusing-based, single-camera, 3D particle tracking. *Journal of Open Research Software*, *9*(1), 1–8.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24. Curran Associates.

Cai, S., Zhou, S., Xu, C., & Gao, Q. (2019). Dense motion estimation of particle images via a convolutional neural network. *Experiments in Fluids*, *60*(4), 1–16.

Clayton, K. N., Moehling, T. J., Lee, D. H., Wereley, S. T., Linnes, J. C., & Kinzer-Ursem, T. L. (2019). Particle diffusometry: An optical detection method for vibrio cholerae presence in environmental water samples. *Scientific Reports*, *9*(1), 1–12.

Colbert, A. J., Co, K., Lima-Cooper, G., Lee, D. H., Clayton, K. N., Wereley, S. T., John, C. C., Linnes, J. C., & Kinzer-Ursem, T. L. (2021). Towards the use of a smartphone imaging-based tool for point-of-care detection of asymptomatic low-density malaria parasitaemia. *Malaria Journal*, *20*(1), 1–13.

Crocker, J. C., & Grier, D. G. (1996). Methods of digital video microscopy for colloidal studies. *Journal of Colloid and Interface Science*, *179*(1), 298–310.

Dreisbach, M., Leister, R., Probst, M., Friederich, P., Stroh, A., & Kriegseis, J. (2022). Particle detection by means of neural networks and synthetic training data refinement in defocusing particle tracking velocimetry. *Measurement Science and Technology*, *33*(12), 124001.

Ganser, A., Roth, G., Galen, J. C. V., Hilderink, J., Wammes, J. J., Müller, I., Leeuwen, F. N. V., Wiesmüller, K. H., & Brock, R. (2009). Diffusion-driven device for a high-resolution dose-response profiling of combination chemotherapy. *Analytical Chemistry*, *81*(13), 5233–5240.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016-December: 770–778. IEEE, New Jersey, USA.

Lee, D. H., Madsen, E. A., Linnes, J. C., & Wereley, S. T. (2022). Temporally and spatially resolved micro-rheometry of a transient viscous polymer formation. *Measurement Science and Technology*, *34*(3), 035301.

Liberzon, A., Lasagna, D., Aubert, M., Bachant, P., Käufer, T., jakirkham, Bauer, A., Vodenicharski, B., Dallas, C., Borg, J., tomerast, & ranleu (2020). OpenPIV/openpiv-python: OpenPIV - Python (v0.22.2) with a new extended search PIV grid option (0.22.2). Zenodo. https://doi.org/10.5281/zenodo.3930343

Mendes, L., Bernardino, A., & Ferreira, R. M. (2020). piv-image-generator: An image generating software package for planar piv and optical flow benchmarking. *SoftwareX*, *12*, 100537.

Newby, J. M., Schaefer, A. M., Lee, P. T., Forest, M. G., & Lai, S. K. (2018). Convolutional neural networks automate detection for tracking of submicron-scale particles in 2d and 3d. *Proceedings of the National Academy of Sciences of the United States of America*. 115: 9026–9031

Olsen, M. G., & Adrian, R. J. (2000). Brownian motion and correlation in particle image velocimetry. *Optics & Laser Technology*, *32*(7-8), 621–627.

Rabault, J., Kolaas, J., & Jensen, A. (2017). Performing particle image velocimetry using artificial neural networks: A proof-of-concept. *Measurement Science and Technology*, *28*(12), 125301.

Raffel, M., Willert, C. E., Wereley, S. T., & Kompenhans, J. (2007). *Particle image velocimetry* (2 edn). Springer. Berlin Heidelberg.

Ratz, M., Sachs, S., König, J., & Cierpka, C. (2023). A deep neural network architecture for reliable 3d position and size determination for lagrangian particle tracking using a single camera. *Measurement Science and Technology*, *34*(10), 105203.

Rossi, M. (2019). Synthetic image generator for defocusing and astigmatic piv/ptv. *Measurement Science and Technology*, *31*, 017003.

Rossi, M., & Barnkob, R. (2020). A fast and robust algorithm for general defocusing particle tracking. *Measurement Science and Technology*, *32*(1), 014001.

Sardana, P., & Wereley, S. T. (2023). Deep particle diffusometry: Using deep learning to measure diffusion coefficient from particle images. *Measurement Science and Technology*, *35*, 034002.

Tran, D., Wang, H., Torresani, L., Ray, J., Lecun, Y., & Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6450–6459. IEEE, New Jersey, USA.

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press

Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, *6*, 1155–1166.

Zareh, S. K., Desantis, M. C., Kessler, J. M., Li, J. L., & Wang, Y. M. (2012). Single-image diffusion coefficient measurements of proteins in free solution. *Biophysical Journal*, *102*(7), 1685–1691.

Zhong, Y., Li, C., Zhou, H., & Wang, G. (2018). Developing noise-resistant three-dimensional single particle tracking using deep neural networks. *Analytical Chemistry*, *90*(18), 10748–10757.

Zolfaghari, M., Singh, K., & Brox, T. (2018). Eco: Efficient convolutional network for online video understanding. *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, Cham, Switzerland.

Cite this article: Sardana P. and Wereley S. T. (2025). DPD-v2: Generalised deep particle diffusometry for varied particle shapes and experimental conditions. *Flow*, *5*, E6. https://doi.org/10.1017/flo.2025.2

https://doi.org/10.1017/flo.2025.2 Published online by Cambridge University Press