Particle image velocimetry analysis with simultaneous uncertainty quantification using Bayesian neural networks

Mia C Morrell, Kyle Hickmann and Brandon M Wilson
Los Alamos National Laboratory, XCP-8, Los Alamos, NM 87545, United States of America
E-mail: mcm8850@nyu.edu

Received 29 November 2020, revised 29 March 2021
Accepted for publication 13 April 2021
Published 14 June 2021

Abstract
Particle image velocimetry (PIV) is an effective tool in experimental fluid mechanics for extracting flow fields from images. Recently, convolutional neural networks (CNNs) have been used to perform PIV analysis with accuracy on par with classical methods. Here we extend the use of CNNs to analyze PIV data while providing simultaneous uncertainty quantification on the inferred flow field. The method we apply in this paper is a Bayesian convolutional neural network (BCNN) which learns distributions of the CNN weights through variational Bayes. In order to demonstrate the utility of BCNNs for the PIV task, we compare the performance of three distinct BCNN models with simple architectures. The first network estimates flow velocity from image interrogation regions only. Our second model learns to infer velocity from both the image interrogation regions and interrogation region cross-correlation maps. Finally, our best performing network infers velocities from interrogation region cross-correlation maps only. We find that BCNNs using interrogation region cross-correlation maps as inputs perform better than those using interrogation windows only as inputs and discuss reasons why this may be the case. Additionally, we test the best performing BCNN on a full synthetic test image pair and a real image pair from the 1st International PIV Challenge. We show that $\sim 98\%$ of true particle displacements from the full synthetic image pair can be captured within the BCNN’s 95\% confidence intervals, and that the BCNN’s performance on the real image pair is quantitatively similar to that of algorithms tested in the 1st International PIV Challenge. Finally, we show that BCNNs can be generalized to be used with multi-pass PIV algorithms with a moderate loss in accuracy, which may be overcome by future work on finetuning and training schemes. To our knowledge, this is the first use of Bayesian neural networks to perform PIV.

Supplementary material for this article is available online

Keywords: particle image velocimetry, uncertainty, Bayesian neural networks

(Some figures may appear in colour only in the online journal)
Classically, PIV algorithms infer the local fluid velocity from the correlation of consecutive image subregions called interrogation windows (IW). Full flow fields can then be determined for the entire image domain. Methods based on utilizing cross-correlation of consecutive images to estimate flow fields have been extremely successful in performing PIV analysis [15]. During the past 20 years, many algorithmic changes have improved the accuracy of calculated PIV velocities, including sub-pixel estimation, window shifting [16], various correlation methods [17–20], and window deformation [21, 22].

More recently, machine learning (ML) methods have been designed to replace the classical cross-correlation algorithm. Instead, convolutional neural networks (CNNs) applied to the image pair have been demonstrated to calculate the full flow field [23–26]. These CNN methods rival classical PIV methods in accuracy.

Whether using classical cross-correlation or advanced CNN methods, imaging conditions and algorithmic choices contribute to complex errors and uncertainties in PIV results. The prevalent use of PIV in research and industry has recently encouraged the development of many new methods of characterizing PIV errors and uncertainties. However, neither classical PIV methods nor CNN-based PIV methods have a generally accepted framework for uncertainty quantification (UQ).

Multiple methods are found in the literature for quantifying PIV uncertainties [27–32]. Strengths, limitations, and algorithms for these methods can be found in [33]. In summary, the existing methods map uncertainty metrics from information contained in the image pairs (e.g. particle image information, noise, etc) or correlation map (e.g. correlation distribution, intensity, noise, etc) to a measure of confidence in the particle velocity prediction. The input-uncertainty mapping is informed empirically from synthetic data or analytically based on a limited number of predetermined features.

In this work, we introduce a method of simultaneous PIV analysis and corresponding uncertainty quantification via deep neural networks. Deep machine learning (DML) offers the possibility of learning complex, non-linear relationships between input and inferred velocity without the reliance on features designed through expert opinion. This flexibility of DML allows inference from a variety of features including pre-determined features commonly used in classical PIV UQ and much more subtle features in the data. Our specific approach is to train Bayesian convolutional neural networks (BCNNs) to map correlation maps and image pairs to particle displacement. The uncertainty prediction is then an inherent feature of the structure of the BCNN inference.

To our knowledge, this work is the first instance of PIV analysis with automated UQ and the first application of Bayesian neural networks to the PIV task. BCNNs have been used successfully to identify complex mappings for other applications, including segmenting computed tomography (CT) scans, analyzing magnetic resonance imaging (MRI) data, detecting the location of light sources, and facial recognition [34–37].

In the following, we describe the methodology of applying BCNNs to PIV uncertainty, including outlining the training data (section 2.1) and providing an overview of convolutional and Bayesian neural networks applied to PIV (sections 2.2 and 2.3). We then present the CNN and BCNN results for three inputs (section 3.1): image pairs (IW), correlation map and image pairs (CM+IW), and correlation map (CM). Next, we present results applying a BCNN to a synthetic image pair and a real image pair from the 1st International PIV Challenge [38] to show how BCNNs may be used in practice (section 3.2). Finally, we present results applying a BCNN to a multi-pass PIV algorithm in order to show how BCNNs can be used in conjunction with existing PIV algorithms (appendix E).

We emphasize the purpose of this work: we aim to demonstrate the feasibility of using BCNNs to predict both particle velocities and their uncertainties simultaneously from image data. We note that all network architectures presented in this work are close to the simplest possible and are likely not the optimal architecture for performing the PIV task. In future work, we aim to expand upon these simple architectures to create deeper, more complex BCNNs for PIV which approach state-of-the-art PIV algorithms in accuracy. Thus, architecture optimization, network generality, sweeps of hyperparameters, and network finetuning are left for future work.

![Figure 1. Sketch of data flow for the neural networks in this work (BCNNs and CNNs). Interrogation windows are specific regions of an image pair, and are used as inputs to some of our neural networks. The other possible input is the cross-correlation map of each interrogation window pair. In this work, we investigate using correlation maps only as inputs (CM), correlation maps and interrogation windows as inputs (CM+IW), and interrogation windows only as inputs (IW) to our neural networks. These neural networks output a probability distribution function (PDF) of particle displacement predictions between members of a given image pair, indicated as PDF(u), PDF(v) in this sketch. We compare the results of each of these neural networks to that of OpenPIV, which uses correlation maps of interrogation window pairs to estimate the particle displacement u, v between image frames.](Image)
2. Methodology

2.1. Synthetic PIV image training data

In order to evaluate the performance of our BCNNs, we train and test BCNNs of nearly identical architecture but different data inputs on the same training and testing data sets. Three network types are trained with the input type changing between each network. The different types of input studied are (a) interrogation windows only (IW), (b) correlation maps and interrogation windows (CM+IW), and (c) correlation maps only (CM).

The training data are 32×32 pixel interrogation region pairs extracted from 16,000 128×128 pixel synthetic image pairs which vary error-contributing parameters (particle displacement, particle diameter, particle density, shear, out-of-plane motion, and background noise). One interrogation region pair is extracted per image pair, yielding a training set of 16,000 interrogation region pairs. For the IW input, the input shape is 32×32×2 pixels, while the CM+IW shape and CM input shapes are 32×32×3 and 64×64×1, respectively. This is because the interrogation windows themselves are 32×32 pixels and their cross-correlation map is 64×64 pixels, which we size down to 32×32 pixels to stack neatly with the 32×32×2 interrogation window pair for the CM+IW input. The parameters used to generate the training data are tabulated in table 1. Throughout this work, background noise level is represented by Gaussian white noise added to the synthetic image with variance which is a given fraction of the standard deviation of the original image.

In the field of deep machine learning, small amounts of background noise are often added to prevent neural networks from overfitting the training data. Additionally, real PIV images have natural noise. We find that adding small amounts of background noise to the training data limits the CM-BCNN’s tolerance of ramps in error contributing parameters compared to its clean counterpart (see appendix C). This most likely occurs because adding very small amounts of noise to image pairs can result in large fluctuations in their correlation maps, making it difficult to learn a mapping between correlation maps and their corresponding particle displacements. For these reasons, we choose to omit adding noise to the training data.

| Parameter                          | Sample range       |
|------------------------------------|--------------------|
| Diameter of particles (pixels)     | [1.0, 5.0]         |
| Particle density (particles/square pixel) | [0.012, 0.117]   |
| Strength of shear (multiplier)     | [0, 0.02]          |
| Angle of shear (degrees)           | [0, 360]           |
| x-displacement (pixels)            | [−4, 4]            |
| y-displacement (pixels)            | [−4, 4]            |
| Out-of-plane displacement (pixels) | [0, 0.3]           |
| Noise level (fraction of standard deviation of image) | 0.0 |

See appendix A for more information concerning synthetic image generation. In brief, particles are assumed to have Gaussian profiles, and their equations of motion contain a linear displacement term and a shear term. A useful reference concerning synthetic particle image generation is [39], which explains such methods in-depth.

We generate two test sets. The first test set (Test Set I) consists of 2000 image pairs, resembles the training data, and is generated using the parameters tabulated in table 1. One interrogation region pair is extracted per image pair, yielding a data set size of 2000 interrogation region pairs. The purpose of Test Set I is to determine how well the BCNNs perform on data similar to their training data.

The second test set (Test Set II) varies each error-contributing parameter, one at a time, while holding all other error-contributing parameters at a constant value, and consists of 2000 image pairs per parameter sweep. One interrogation region pair is extracted per image pair, yielding a data set size of 2000 interrogation region pairs per parameter sweep. The parameters used to generate Test Set II are shown in table 2. The purpose of Test Set II is to evaluate how the BCNNs respond to uncertainty introduced by variation in single error-contributing parameters.

| Parameter                          | Default value | Sweep range/value |
|------------------------------------|---------------|-------------------|
| Diameter of particles (pixels)     | 3.0           | [0.5, 5]          |
| Particle density (particles/square pixel) | 0.098     | [0.0098, 0.146]   |
| Strength of shear (multiplier)     | 0.0           | [0.0, 0.1]        |
| Angle of shear (degrees)           | 0.0           | 0.0               |
| x-displacement (pixels)            | −1.0          | [−5.2, 5.20]      |
| y-displacement (pixels)            | 1.0           | [−5.2, 5.20]      |
| Out-of-plane displacement (pixels) | 0.0           | [0.0, 0.4]        |
| Noise level (fraction of standard deviation of image) | 0.0 | [0.0, 0.2] |

2.2. Convolutional neural network

Convolutional neural networks (CNNs) have been widely used in computer vision tasks, including that of PIV [23–26]. CNNs are a type of deep neural network designed to exploit structures such as locality and translational invariance found in images [40]. The basic building block of deep neural networks (including CNNs) is a *neuron* *a* mapping \( \mathbb{R}^n \) to \( \mathbb{R}^m \) which takes in an input vector \( X \in \mathbb{R}^n \) and outputs a nonlinearly transformed vector, \( Y = a(X) = \sigma(WX + b) \) in \( \mathbb{R}^m \) (see figure 2). Here \( \sigma \) is a nonlinear function applied element-wise, \( b \) is a vector of biases, and \( W \) is a weight matrix. Neurons of the network are arranged into layers in which the out-
Figure 2. (A) Basic operation of a single neural network layer. Inputs $X$ to the neural network undergo a linear transformation $WX + b$ (where $W$ and $b$ are scalar matrices referred to as weights and biases, respectively) and are passed through a nonlinear function, or activation to produce an output. (B) A sketch of the operation of a convolutional layer with a single filter of kernel size $3 \times 3$ with stride of 1. Stride indicates how far the kernel slides between each step of the convolution. The kernel applies a transformation to the shaded blue area, outputting a convolutional result, denoted in red for each step of the convolution.

Figure 3. (A) Sketch of a convolutional neural network (CNN). At the input layer, image data is fed into the CNN. Within the hidden layers, the image passes through convolutional layers, the operation of which is described in figure 2. In this sketch, the image passes through three convolutional layers. The first, second, and third convolutional layers all have two filters with kernel size of $2 \times 2$ and stride of 1. Transparent squares represent filter locations on each layer input. Each color square represents a different filter location. After the final convolutional layer, an output of two vectors $Y_1$ and $Y_2$ is obtained at the output layer. (B) Diagram showing training process for a neural network. Inputs are fed into the neural network, producing predictions. The error of these predictions is calculated. Via backpropogation, an algorithm which computes the neural network’s gradient in weight space with respect to a loss function, the weights of the neural network (illustrated in figure 2(A)) are updated. This process continues, minimizing the error between the predicted and target outputs via the loss function.

The use of convolutional filters in CNNs has two main advantages. First, it drastically reduces the number of trainable parameters in the network and second, it creates translation invariant nonlinear filters in the network allowing the network to learn feature structure independent of where a feature occurs in the image.

Input to the CNN (IW, IW + CM, or CM in this paper) is fed into the input layer and passed through each layer of the
2.2.1. Convolutional layers. Convolutional layers apply a set of convolutional filters to the input in each layer. A 2D convolutional layer consists of neurons arranged in 3-dimensional blocks and is specified by its height, width, and depth. Each of these neurons computes an output value from its input in the previous layer. The height and width indicate the height and width of the layer in the spatial plane (in neurons), while the depth indicates the number of filters in the layer. Neurons which belong to a given filter share weights and biases. Filters use a small window of the previous layer as inputs, and the convolution is computed by running these filters over the entire spatial plane, as shown in figure 2(B). In this work, we use only linear, $\sigma_i(x) = x$, and rectified linear unit (ReLU), $\sigma_i(x) = \max(0, x)$, activations.

2.2.2. Batch normalization layers. Batch normalization, a widely used regularization scheme for neural networks, scales and shifts the output of a neural network layer such that the result has approximately zero mean and unit variance [43]. Batch normalization is used to increase the speed and stability of neural networks. Although batch normalization was originally recommended for use after convolutional layers and before activations, in this work we apply batch normalization layers after activations. We use this order of convolutional layers, activations, and batch normalization since it produces more stable neural networks in the case of our chosen problem and network architecture.

2.2.3. Training. During training, the CNN learns by iteratively optimizing its biases and weights in order to minimize a loss function that quantifies how well the network fits the training data. Optimizers used are usually based off of gradient descent algorithms with the key difference that each evaluation of the loss function is only performed on a small subset, known as a batch, of the full training set. After each evaluation of the loss function and performance of the gradient descent step a new batch is chosen. This allows the network to learn from huge training sets but also implies that the loss function being reduced changes from step to step of the optimization. This type of optimization problem is referred to as a stochastic optimization.

Gradients of the loss are computed through a forward pass and a backward pass algorithm. During the forward pass, inputs are propagated through the CNN and outputs are returned. During the backward pass gradients are computed using the chain rule, the loss function is evaluated, and neuron weights are updated using gradient descent. Backpropagation, an algorithm which computes the neural network’s gradient in weight space with respect to a loss function, is a key tool used during the backward pass [44]. See figure 3(B) for an illustration of the training process.

In this work, we minimize the mean squared error loss function (MSE) using the Adam optimizer [45] while training our CNNs. This has become a standard training procedure for deep neural networks.

2.2.4. CNN structure. For the CM input (which has a size of $64 \times 64 \times 1$ pixels), we apply 29 convolutional layers with 16 filters each and a kernel size of $3 \times 3$ pixels followed by a convolutional layer with two filters and kernel size of $6 \times 6$. Between each convolutional layer we apply an activation, either ReLU or linear, and after each activation we apply a batch normalization layer (except for the last activation). See figure 4 for an example diagram of the CM-CNN. Complete diagrams of the IW and CM+IW CNNs can be found in figures S1 and S2 of the supplement.

2.3. Bayesian neural network

Bayesian neural networks offer a solution to the deterministic neural network’s lack of pointwise uncertainty quantification. In this work, we follow the approach of [46] to Bayesian neural networks. Bayesian neural networks account for the
uncertainty of neural network weights by estimating the posterior prediction distribution

\[ p(y|x, D) = \int p(y|x, w)p(w|D)dw \]  

(1)

where \( x \) is the input to the neural network with label \( y \), \( D = \{(x_i, y_i)\} \) is the training data, \( w \) are the weights of the neural network, and \( p(D|w) = \prod_i p(y_i|x_i, w_i) \) is the likelihood function, or the probability of predicting \( \{y_i\} \) given inputs \( \{x_i\} \) and weights \( \{w_i\} \). Thus for a given input \( x \), a Bayesian neural network returns the probability of observing its corresponding label \( \hat{y} \):

\[ p(\hat{y}|x) = E_{p(w|D)}p(\hat{y}|x, w). \]  

(2)

2.3.1. Deriving a loss function. In order to approximate the true posterior \( p(w|D) \), we must apply variational inference by minimizing the Kullback-Leibler (KL) divergence between the true posterior \( p(w|D) \) and a distribution of known form \( q_0(w) \). We do this by minimizing the variational free energy as a function of \( \theta \),

\[ \mathcal{F}(D, \theta) = KL(q_0(w)||p(w)) - E_{q_0(w)}\log p(D|w) \]  

(3)

where the first term is the KL divergence between the variational posterior \( q_0(w) \) and the prior distribution on the weights, \( p(w) \), and the second term is the expected value of the log-likelihood of data \( D \) given weights \( w \) distributed by the variational posterior. We can approximate equation (3) via sampling \( w^{(j)} \) from \( q_0(w) \):

\[ \mathcal{F}(D, \theta) \approx \frac{1}{N} \sum_{j=1}^{N} [\log q_0(w^{(j)}) - \log p(w^{(j)}) - \log p(D|w^{(j)})]. \]  

(4)

Since we are performing regression, we can substitute the mean squared error function for the negative log likelihood \(-\frac{1}{2} \sum_{j=1}^{N} \log p(D|w^{(j)})\), assuming Gaussian \( p(y_i|x_i, w_i) \) with fixed standard deviation (see appendix B).

In this work, we use take \( \theta = (\mu, \sigma) \) and approximate the true posterior \( p(w|D) \) with a Gaussian \( q_0(w) \) where \( \mu \) is the mean and \( \sigma \) is the standard deviation. Therefore, compared to a deterministic neural network of the same structure, our Bayesian neural network has twice the number of parameters.

2.3.2. Training. Training iterations consist of a forward and a backward pass, as in deterministic neural networks (described previously in section 2.2.3). The forward pass is performed by drawing a sample from \( q_0(w) \), which is then used to evaluate equation (3). The backward pass updates values of weight distribution parameters \( \mu \) and \( \sigma \) via backpropagation, which is made possible by the Flipout Monte Carlo estimator [47].

2.3.3. Predicting particle displacement vectors. A Bayesian neural network effectively constructs an ensemble of neural networks. By sampling this network ensemble, we obtain an ensemble of outputs for a single input from which statistics such as mean and standard deviation can be computed. To make predictions using the Bayesian neural network, we draw a sample from \( q_0(w) \) and use it to evaluate the output of the neural network. For any given input, we draw 2000 samples from \( q_0(w) \) and compute the mean and standard deviation of the resulting 2000 neural network outputs. We therefore can capture the center and spread of each prediction made by the Bayesian neural network.

2.3.4. BCNN structure. To perform PIV using a BCNN, we use the exact neural network structures and inputs used in our CNNs (shown in figures S1, S2, and 4). As described in the context of our CNNs, the inputs to the BCNN are either (a) interrogation window pairs (IW), (b) interrogation window pairs and their correlation maps (CM+IW), or (c) the cross-correlation of interrogation window pairs (CM). This input is fed into the BCNN, producing a prediction of the particle displacement vectors between the interrogation window pairs at the output layer.

Our BCNNs differ from our CNNs since they use 2D convolutional variational layers instead of 2D convolutional layers. Convolutional variational layers differ from convolutional layers, as convolutional variational layers use distributions for weights as explained in the previous section. For all convolutional variational layers, the Flipout algorithm [47] is used to estimate gradients for backpropagation and each layer is initialized with the standard normal prior.

3. Results

3.1. BCNN and CNN performance

Here we demonstrate that the BCNN and CNN using the correlation maps only as inputs (CM-BCNN and CM-CNN) perform the best on both Test Sets I and II out of the three BCNNs and three CNNs tested. We show that the CM-BCNN is capable of performing similarly to a simple single pass PIV algorithm implemented using OpenPIV [48, 49] in accuracy during sweeps of many of the tested error-contributing parameters. These results are followed by a discussion of the relation between the tested BCNNs, their corresponding CNNs, and the OpenPIV algorithm. Next, we perform full PIV on entire synthetic and real image pairs using the best performing network, the CM-BCNN. Finally, in appendix E, we implement a ‘double pass’ algorithm using the CM-BCNN to evaluate generalizability to correlation maps generated via slightly different algorithms.

Note that throughout this work, we use OpenPIV without any of its error-correction or window deformation capabilities for a fair comparison to and preliminary demonstration of BCNNs applied to PIV. For use with error-correction or window deformation, inputs generated from these advanced
Figure 5. Predicted particle displacements (pixels) versus true displacements in the x-direction (pixels). Predicted versus true particle displacement in the y-direction are similar and are shown in the supplement (figure S3). (A), (C) and (E) show particle displacements predicted by Bayesian neural networks with interrogation windows only as inputs (IW-BCNN, purple), correlation maps and interrogation windows as inputs (CM+IW-BCNN, brown), and correlation maps only as inputs (CM-BCNN, red), respectively, versus true particle displacements. Error bars are standard deviations. (B), (D) and (F) show particle displacements predicted by the deterministic versions of (A), (C) and (E) (IW-CNN, CM+IW-CNN, CM-BCNN), respectively versus true particle displacements. (G) shows particle displacements predicted by OpenPIV (PIV, blue) versus true particle displacements. We observe that the CM-BCNN and the CM-CNN perform the best out of the BCNNs and CNNs tested, respectively. All of the tested BCNNs and CNNs decrease the number of large errors in displacement, or spurious displacements, compared to the OpenPIV algorithm (F).

3.1.1 Test Set I. First, we compare how well our BCNNs learn to extract displacement vectors from interrogation window and/or correlation map information by evaluating their performance on Test Set I, a data set similar to the training set. Figure 5 shows the performance of the BCNN (left column), CNN (center column), and OpenPIV algorithm (right column) in predicting the x displacement. Predictions of y displacement are similar to those shown in figure 5 and are shown in the supplement (figure S3). The three network types, varying input classes, are also compared (IW (top), CM+IW (middle), and CM (bottom)).

The BCNN and CNN models trained on CM+IW (CM+IW-BCNN, CM+IW-CNN) and CM (CM-BCNN, CM-CNN) show similar performance in predicting displacements to OpenPIV. However, both BCNN and CNN methods predict fewer spurious vectors, or particle displacement predictions with large errors, than the OpenPIV algorithm. This suggests BCNN and CNN methods may reduce the need for complex spurious vector detection algorithms. Additionally, BCNN and CNN methods may provide a rigorous means to identify and replace spurious vectors.

An advantage of the BCNN method is the accompanying predicted uncertainties. Predicted uncertainties (95% confidence intervals) capture ≈ 67%, ≈ 93% and ≈ 98% of the true displacements for the IW, CM+IW and CM training sets, respectively. We see that the BCNN using the correlation maps algorithms can be incorporated into the BCNN training following the methods outlined in this paper.
only as inputs (CM-BCNN) performs the best out of the three BCNNs tested on Test Set I, with an $R^2$ value of 0.993. This is most likely due to the additional feature extraction that cross-correlating the interrogation window pairs provides, and the omission of potentially uninformative information originating from the raw interrogation window pair.

Here we note some interesting phenomena that can be observed in figure 5. First, we see that the BCNN and CNN predictions are inferior when trained on IW input alone. However, the BCNN predicted uncertainties (95% confidence intervals) capture ≥67% of the true displacements, which is a surprisingly significant proportion given that no feature extraction (i.e. cross-correlation) has been applied. While, in this work, we have found poor performance using CNN architectures on IW input, it is always possible that variations in training hyperparameters or network architecture could allow for improved performance. Due to the simplicity of the architectures presented in this work, it is likely more optimal architectures can be found for performing the PIV task. We believe an exciting line of future work is the search for and application of these more optimal architectures.

Upon inspection, both the CM+IW-CNN and CM+IW-BCNN networks may be impacted by pixel-locking phenomena. This is indicated by clustering of increased errors near integer true displacements. This may be due to certain patterns of operations learned by both the CM+IW-BCNN and CM+IW-CNN in the interrogation window features. Further investigation of the activations of these networks, or the result of each network layer, may allow us to discern whether the CM+IW-BCNN and CM+IW-CNN are learning patterns which contribute to pixel-locking. This is outside the scope of the present report but will be explored in future work.

Furthermore, we observe that the BCNNs seem to predict larger uncertainties and/or predict less accurate predictions at the boundaries of their training regime ([-4, 4]) pixels. This is because BCNNs are sensitive outside the training data limits. See figure A6 for an example of a CM-BCNN trained on particle displacements ranging between [-12, 12] pixels displaying the same phenomenon around the edges of its training regime.

Finally, we note that the results displayed by each BCNN and its corresponding CNN (i.e. CM-BCNN and its deterministic version, CM-CNN) in figure 5 are quantitatively similar. This similarity emerges because BCNNs are effectively ensembles of CNNs with identical architectures. Thus, any appropriately trained CNN should return predictions contained within the prediction distribution of an appropriately trained BCNN of identical architecture. We observe this behavior in Test Sets I (figure 5) and II, as discussed in the next section.

### 3.1.2. Test Set II

Next, we evaluate how each BCNN responds to changes in error contributing parameters by performing predictions on Test Set II, a data set which varies one error contributing parameter at a time while holding all other error contributing parameters constant.

In figures 6–11, we show the root mean square error (RMSE) of each BCNN’s particle displacement predictions in solid circles and the RMSE of each CNN’s predictions in hollow circles as a function of the error contributing parameters: particle displacement, particle density, particle diameter, particle shear, out of plane motion, and background noise level. For reference, similar analytic trends are demonstrated in Raffel et al [39].

The RMSE is calculated using the following equation:

$$ \text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[ (x_i^{(p)} - x_i^{(t)})^2 + (y_i^{(p)} - y_i^{(t)})^2 \right]} $$

where $x_i^{(p)}$ and $y_i^{(p)}$ indicate predicted $x$ and $y$ displacements, and $x_i^{(t)}$ and $y_i^{(t)}$ indicate true $x$ and $y$ displacements. Error bars indicate standard deviations from the RMSE particle displacement of each BCNN, which is calculated by propagating the error appropriately according to equation (5). Additionally,
Figure 7. RMS error of particle displacements (pixels) predicted by BCNNs and CNNs using interrogation windows only (IW, (A)), correlation maps and interrogation windows (CM+IW, (B)), and correlation maps only as input (CM, (C)) versus particle density. Results from BCNNs are marked in solid circles with error bars representing standard deviations about the particle displacement prediction RMS error, while results from CNNs are marked in open circles. Each data point, solid or open, has $N = 100$ samples, and is generated according to equation (5). Error bars are generated by propagating the uncertainty of particle displacement predictions appropriately in response to passing particle displacements through equation (5). Blue dashed lines are the RMS error of predictions made by OpenPIV, also generated via equation (5) with $N = 100$.

Figure 8. RMS error of particle displacements (pixels) predicted by BCNNs and CNNs using interrogation windows only (IW, (A)), correlation maps and interrogation windows (CM+IW, (B)), and correlation maps only as input (CM, (C)) versus particle diameter. Results from BCNNs are marked in solid circles with error bars representing standard deviations about the particle displacement prediction RMS error, while results from CNNs are marked in open circles. Each data point, solid or open, has $N = 100$ samples, and is generated according to equation (5). Error bars are generated by propagating the uncertainty of particle displacement predictions appropriately in response to passing particle displacements through equation (5). Blue dashed lines are the RMS error of predictions made by OpenPIV, also generated via equation (5) with $N = 100$.

Figure 9. RMS error of particle displacements (pixels) predicted by BCNNs and CNNs using interrogation windows only (IW, (A)), correlation maps and interrogation windows (CM+IW, (B)), and correlation maps only as input (CM, (C)) versus particle shear. Results from BCNNs are marked in solid circles with error bars representing standard deviations about the particle displacement prediction RMS error, while results from CNNs are marked in open circles. Each data point, solid or open, has $N = 100$ samples, and is generated according to equation (5). Error bars are generated by propagating the uncertainty of particle displacement predictions appropriately in response to passing particle displacements through equation (5). Blue dashed lines are the RMS error of predictions made by OpenPIV, also generated via equation (5) with $N = 100$. 
Figure 10. RMS error of particle displacements (pixels) predicted by BCNNs and CNNs using interrogation windows only (IW, (A)), correlation maps and interrogation windows (CM+IW, (B)), and correlation maps only as input (CM, (C)) versus particle out of plane motion. Results from BCNNs are marked in solid circles with error bars representing standard deviations about the particle displacement prediction RMS error, while results from CNNs are marked in open circles. Each data point, solid or open, has \( N = 100 \) samples, and is generated according to equation (5). Error bars are generated by propagating the uncertainty of particle displacement predictions appropriately in response to passing particle displacements through equation (5). Blue dashed lines are the RMS error of predictions made by OpenPIV, also generated via equation (5) with \( N = 100 \).

Figure 11. RMS error of particle displacements (pixels) predicted by BCNNs and CNNs using interrogation windows only (IW, (A)), correlation maps and interrogation windows (CM+IW, (B)), and correlation maps only as input (CM, (C)) versus background noise level. Results from BCNNs are marked in solid circles with error bars representing standard deviations about the particle displacement prediction RMS error, while results from CNNs are marked in open circles. Each data point, solid or open, has \( N = 100 \) samples, and is generated according to equation (5). Error bars are generated by propagating the uncertainty of particle displacement predictions appropriately in response to passing particle displacements through equation (5). Blue dashed lines are the RMS error of predictions made by OpenPIV, also generated via equation (5) with \( N = 100 \).

recall that the mean and standard deviation of each predicted particle displacement distribution generated by the BCNNs is estimated by drawing 2000 samples.

As observed previously on Test Set I, out of the three classes of BCNNs and CNNs tested, the BCNN and CNN using correlation maps only as inputs (CM-BCNN, CM-CNN) perform the best on Test Set II. The mean particle displacement predictions made by the CM-BCNN are similar in accuracy to those predicted by OpenPIV throughout ramps of particle shear (figure 9) and out of plane motion (figure 10). During sweeps of particle displacement (figure 6), particle density (figure 7), and particle diameter (figure 8), the mean displacement predictions made by the CM-BNN are also similar in accuracy to those made by OpenPIV in accuracy for the majority of the error contributing parameter range tested. As expected, the CM-BCNN performs poorly when presented with particle displacements outside of its training regime of \( x, y \in [-4,4] \) pixels. Additionally, while ramping the error contributing parameter of background noise level (figure 11), the mean particle displacement predictions made by the CM-BCNN are less accurate but within \( \sim 0.2 \) pixels of those made by OpenPIV.

The standard deviations of predicted RMS particle displacements output by the BCNNs in figures 6–11 range from from 0.01 to 0.04 for the IW-BCNN, from 0.03 to 0.8 for the CM+IW-BCNN, and from 0.005 to 0.7 for the CM-BCNN. Since the uncertainties output from the BCNN are computed by taking the standard deviation of 2000 samples drawn from the BCNN, a smaller uncertainty indicates a narrower distribution of possible particle displacement predictions and a larger uncertainty indicates a wider distribution.

Note that it is possible for a BCNN to be over-confident (i.e. predict an unrealistically small uncertainty) about a particle displacement prediction. This occurs when a BCNN is unable to learn the appropriate weight distributions required to predict accurate particle displacements, thus outputting wildly incorrect displacement prediction distributions with unrealistic mean and spread. Examples of this phenomenon can be...
seen in panel (A) of figures 6–11, as the IW-BCNN is suboptimal, yet places small error bars on its particle displacement predictions. It is therefore important to construct BCNNs with appropriate architectures and provide appropriate training data in order to receive meaningful particle displacements predictions and corresponding uncertainty predictions. One can diagnose an extremely suboptimal neural network such as the IW-BCNN by noting its performance on validation and test data, and looking for suspicious signs such as predicted uncertainties which remain unrealistically small despite displacement vector magnitude and other error contributing parameter levels on real data.

It is important to again note the quantitative similarities between the results for each BCNN and its corresponding CNN (i.e. CM-BCNN and its deterministic version, CM-CNN) in figures 6–11. As discussed previously in Results: Test Set I, the reason for this similarity is that a BCNN acts as an ensemble of CNNs of identical architecture. This is particularly apparent in figures 6–10, where most RMS displacement errors generated by CM-CNN and CM+IW-CNN models fall within roughly three standard deviations of the RMSE displacement of their corresponding BCNN model. Inconsistencies in this trend can be found in most of the IW-BCNN models. This anomaly occurs due to the following: a BCNN represents an ensemble of CNNs, some of which may be more accurate or less accurate than ±3 standard deviations away from the mean output of the CNN ensemble.

### 3.2. Performing PIV using the CM-BCNN on full synthetic and real images

We have established that the CM-BCNN is the most accurate BCNN out of the three tested (IW, CM+IW, CM). We apply the CM-BCNN and OpenPIV on two full image pairs: (a) a synthetic image—generated using the parameters and equations of motion detailed in table 3—and (b) Case A of the International PIV Challenge which consists of a real experimental image pair.

In figure 12 we show the true flow field (figure 12(A)) of the synthetic images, the flow field estimated by OpenPIV (figure 12(B)), and the mean flow field predicted by the CM-BCNN (figure 12(C)), from left to right. Different colors in figure 12 represent flow vector magnitude.

Slight variations from the true flow field in vector direction and magnitude can be observed in both the OpenPIV estimated flow field and the CM-BCNN estimated flow field (figures 12(D) and (E), respectively). Note that the error magnitude is comparable between OpenPIV and the CM-BCNN.

The CM-BCNN performs well in capturing the true flow, upon visual inspection, with performance comparable to that of OpenPIV. In figures 13 and S6 we show the particle displacements predicted by OpenPIV and the CM-BCNN versus the true particle displacements between the image pair whose true flow field is shown in figure 12(A). The CM-BCNN is able to capture 98.44% of the true particle displacements within its 95% confidence interval (figures 13(A) and S6(A)). The accuracy of the CM-BCNN is similar to that of OpenPIV, whose particle displacement predictions are shown in figure 13(B).

Next, we test the CM-BCNN and OpenPIV on a real image pair from the 1st International PIV Challenge [38]. Specifically, we complete Case A: a pair of experimental images of the wake behind an aircraft taken by the DLR (German Aerospace Center). Case A features many challenges for PIV algorithms, including strong velocity gradients, particle density loss, and varying particle sizes. We reproduce the plots (figures 2 and 4) associated with Case A shown in the results from the 1st International PIV Challenge [38] using our CM-BCNN and OpenPIV, and then compare these results to those of the algorithms participating in the PIV Challenge.

To adapt to the large particle displacements displayed in the real image, we trained a new ‘expanded’ CM-BCNN on displacements ranging from −12 pixels to +12 pixels and use this neural network to generate the results of figures 14–16, setting all other parameters in this expanded training set to the values reported in table 1. Results on Test Sets I and II for this expanded CM-BCNN are reported in appendix D.

First, we generate the full estimated flow field for the real image pair, shown in figure 14. Comparing to the corresponding figure (figure 2) in the results from the 1st International PIV Challenge [38], we find that upon visual inspection our CM-BCNN and OpenPIV perform similarly to the average of the DUTAE, PURDUE, CORIA1, CORIA2, UC3M and LAVISION algorithms the authors present as a reference for Case A. Both the CM-BCNN and OpenPIV struggle to predict accurate vectors in the challenging central region, just as many of the algorithms presented in the results from the 1st International PIV Challenge [38]. In figure 15, we can see that the CM-BCNN is more uncertain about the central vortex region than the rest of the image (with the exception of occasional vectors with large uncertainties outside the central vortex). We also observe that the CM-BCNN performs about as well as OpenPIV in figure S19, capturing ~96.48% and ~95.47% of x and y displacements predicted by OpenPIV, respectively within its predicted 95% confidence intervals, and displaying $R^2$ values of 0.958 and 0.951 for x and y displacements for the CM-BCNN vs OpenPIV curve.

Following figure 4 in the results from the 1st International PIV Challenge [38], we then plot the particle displacement

| Parameter                          | Value/range |
|-----------------------------------|-------------|
| Diameter of particles (pixels)    | 3.0         |
| Particle density (particles/pixel) | 0.098       |
| Strength of shear (multiplier)    | 0.01        |
| Angle of shear (degrees)          | 0.0         |
| x-displacement (pixels)           | [−1.0, 1.0] |
| y-displacement (pixels)           | [−1.0, 1.0] |
| Out-of-plane displacement (pixels)| 0.05        |
| Noise level (fraction of standard deviation of image) | 0.001 |
Figure 12. Performance of OpenPIV and CM-BCNN on a simulated test image pair with parameters listed in table 3. Color indicates the magnitude of each interrogation window displacement vector (in pixels). In pixels, (A) shows the true flow field, (B) shows the flow field estimated by OpenPIV, and (C) shows the mean flow field estimated by the CM-BCNN. (D) and (E) show the errors of the OpenPIV generated flow field and the CM-BCNN generated flow field, respectively. Errors generated by OpenPIV (shown in (D)) fall in $[-0.50, 0.60]$ pixels in the $x$-direction and $[-0.48, 0.45]$ pixels in the $y$-direction. Errors generated by the CM-BCNN (shown in (E)) are bounded by $[-0.52, 0.63]$ pixels in the $x$-direction and $[-0.63, 0.51]$ pixels in the $y$-direction.

Figure 13. (A) Particle displacements (in pixels) within the flow field shown in figure 12 predicted by the CM-BCNN versus true particle displacements within the flow field shown in figure 12 in the $x$-direction. Note that 98.44% of the true particle displacements are contained within the 95% confidence interval predicted by the CM-BCNN. Error bars are standard deviations. (B) Particle displacements (in pixels) within the flow field shown in figure 12 predicted by OpenPIV vs true particle displacements in the $x$-direction. Results for the equivalent of (A) and (B) in the $y$-direction are similar and are shown in the supplement (figure S6).

profiles estimated by the CM-BCNN and OpenPIV along the $y = 496$ pixel line in figure 16 and compare these to the profiles shown by algorithms participating in the PIV Challenge [38]. We again find that both the CM-BCNN and OpenPIV perform similarly to the algorithms participating in the 1st International PIV Challenge. Specifically, the $x$ and $y$ displacement profiles along the $y = 496$ pixel line of both the CM-BCNN and OpenPIV display the same qualitative shape and similar minima and maxima as the algorithms tested in the PIV Challenge.

In summary, we have found that the CM-BCNN performs well on both the full synthetic image pair and the full real image pair. Performance on the full synthetic image pair is comparable to that of OpenPIV, and the expanded CM-BCNN’s performance on the real image pair from Case A of the 1st International PIV Challenge [38] is comparable to the algorithms participating in the PIV Challenge. A key takeaway from these tests on synthetic and real images is that BCNNs can be constructed for particular tasks, such as recognizing large particle displacements, by adjusting their training data.

In addition to the results presented in this section, we also provide results and discussion concerning the application of the CM-BCNN to multi-pass algorithms in appendix E. In brief, we show that with a moderate loss in accuracy, the CM-BCNN generalizes successfully to correlation maps generated by multi pass algorithms. We expect that this loss in accuracy may be overcome with both algorithmic improvements and adjustments of neural network architecture and training data.
Figure 14. Particle displacement prediction (in pixels) made by OpenPIV (A) and mean particle displacement prediction (in pixels) made by CM-BCNN (B) for a real test image pair. The test images are from Case A of the First International PIV Challenge [38] and feature strong gradients, particle density loss, and varying particle sizes. The most challenging region of these images is the center of the vortex, where error-contributing effects such as particle density loss and strong gradients are strongest. Color indicates the magnitude of each interrogation window displacement vector. Both OpenPIV and the CM-BCNN produce flow fields similar to the ‘ground truth’ flow field displayed in figure 2 of the First International PIV Challenge’s results [38], which is calculated by averaging predictions from the DUTAE, PURDUE, CORIA1, CORIA2, UC3M and LAVISION PIV algorithms.

Figure 15. Performance of CM-BCNN on the real test image pair whose estimated flow field (in pixels) is shown in figure 14, with arrow color indicating the sum of predicted standard deviations predicted by the CM-BCNN, $\sqrt{\sigma_x^2 + \sigma_y^2}$ (in pixels). As in figure 14, arrows indicate the mean CM-BCNN particle placement predictions. Note that in the challenging center of the vortex, the uncertainties output by the CM-BCNN are much larger ($\sim$4 pixels) than outside of the vortex ($\sim$0.3 pixels, with the exception of sporadic vectors with large uncertainties).

4. Discussion

In this work, for the first time, we have shown that Bayesian convolutional neural networks are effective for simultaneous PIV calculation and uncertainty quantification. First, we found that correlation maps (CMs) are the best inputs for both our BCNN and CNN models out of the inputs tested (IW, CM+IW, and CM). Furthermore, by testing each of our neural networks on Test Sets I and II, we demonstrated that the CM-BCNN is able to both output accurate predictions rivaling OpenPIV and provide simultaneous uncertainty quantification. Upon application to Test Set I, it was seen that CM-BCNN predicted uncertainties (95% confidence intervals) capture $\approx$98% of true displacements. Additionally, via testing on Test Set II, we observed that the CM-BCNN is fairly robust to ramping error contributing parameters. Finally, the quantitative similarity between BCNN and corresponding CNN predictions across tests on Test Sets I and II are consistent with the theoretical expectation that predictions from a given CNN
should fall within the distribution of predictions generated by a BCNN of identical architecture.

Once we established that the CM-BCNN was the best performing neural network out of those tested, we applied it to the test problem of a full synthetic image pair with a simple flow field. The CM-BCNN successfully captured 98% of true particle displacements in its 95% confidence interval, with accuracy comparable to that of OpenPIV.

Next, we evaluated the CM-BCNN on real images from Case A of the 1st International PIV Challenge. To detect the large particle displacements present in these images, we trained a CM-BCNN on a training set with parameters identical to those shown in table D1. We found that this expanded CM-BCNN performed well on the real images, with accuracy comparable to those of both OpenPIV and the algorithms tested in the 1st International PIV Challenge.

As a final test, in appendix E we evaluated the generalizability of the CM-BCNN to correlation maps generated by multi-pass algorithms. With a moderate loss of accuracy, we found that the CM-BCNN was able to generalize to the multi-pass correlation maps. By adjusting training datasets and finetuning the CM-BCNN, future work may yield a CM-BCNN which can be applied to both single and multi pass algorithms with no loss of accuracy.

The network architectures used in this work are very simple, lending themselves to the application of interpretability studies to understand the features in correlation maps being used for inference of particle displacement. However, all network architectures presented in this work are close to the simplest possible and are most likely sub-optimal for performing the PIV task. We leave architecture optimization, network generality, sweeps of hyperparameters, and network finetuning for future work. Indeed, one indication that there may be future improvements in this application is the universally poor performance of BCNNs and CNNs trained on just the flow field interrogation windows. The information to compute correlation maps is certainly contained in the interrogation windows and therefore should be able to be utilized by the CNNs. However, the transformation from an interrogation window to a correlation map is a very specific non-linear transformation and therefore the simple architectures explored here are insufficient. In future work, we will incorporate this transformation into the network architecture.

Our primary future direction of research is to include more complex structures in the BCNN architecture, i.e. the pyrimidal feature extraction used in state-of-the-art CNNs such as LiteFlowNet [50]. Constructing deeper, more complex BCNNs is the most promising avenue to BCNNs that can rival state-of-the-art PIV algorithms in accuracy. By expanding the network architecture, we aim to be able to recognize more nuanced features in the image data and therefore be able to train the BCNNs on more challenging image data such as images featuring turbulence and complex flows.

Besides improving the network architecture, application of deep ML to PIV analysis opens many avenues for inferring fluid flow information. In many applications there are multiple flow diagnostics. It is a simple task to construct networks that can take in highly varied sets of simultaneous diagnostic input. Understanding how to construct effective, stable, network architectures which couple multiple diagnostics is an exciting area of future research.

Finally, we stress the novelty and utility of this work. By using BCNNs to perform PIV tasks, we have demonstrated that predicted particle displacements and their uncertainties can be generated simultaneously from the same algorithm. We emphasize that the uncertainties produced from the BCNNs are directly produced from the algorithm that generates particle displacement predictions, which is not the case for many existing PIV uncertainty quantification methods. We suggest that using BCNNs to perform PIV is a solution to the absence of a universally accepted uncertainty quantification method for classical and CNN-based PIV algorithms and anticipate that understanding how to apply BCNNs to PIV will become a compelling field of study.

Data availability statement

Any data that support the findings of this study are included within the article.

Acknowledgment

This work was supported by the U.S. Department of Energy through the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). This work has been approved for release under LA-UR-20-29836.

Appendix A. Synthetic image generation

Synthetic particle image pairs are generated by uniformly seeding an initial image with particle locations and then updating these locations to create the second image in the pair. Particle locations \((x[t], y[t], z[t])\) are updated using the following equation of motion:

\[
x[t] = x[t-1] + \Delta x + (-X[t-1] \sin \alpha_g + Y[t-1] \cos \alpha_g) \\
\times g \cos \alpha_g
\]

\[
y[t] = y[t-1] + \Delta y + (-X[t-1] \sin \alpha_g + Y[t-1] \cos \alpha_g) \\
\times g \sin \alpha_g
\]

\[
z[t] = z[t-1] + \Delta z
\]

where \((x[t], y[t], z[t])\) is the particle’s updated position at time \(t\), \((\Delta x, \Delta y, \Delta z)\) is the particle displacement vector, and \(g\) is a multiplier which tunes the magnitude of shear displacement with direction \(\alpha_g\). The coordinates \((X[t-1], Y[t-1])\) are a transformation of the image coordinates \((x[t-1], y[t-1])\) and represent the particle’s distance...
from the center of the image minus the overlap term \( lw \), where \( l = 0.75 \) is a multiplier of interrogation window size \( w \). For each image frame in the image pair, each synthetic particle is given a Gaussian intensity profile. More detailed discussion of synthetic particle image generation is given in [39].

**Appendix B. Negative log-likelihood is equivalent to MSE for Gaussian distributions with fixed standard deviation**

Here we show that for Gaussian \( p(y|x, w) \) with fixed standard deviation \( s_i = s \), the negative log likelihood is interchangeable with the mean squared error of data \( D \).

Suppose \( p(y|x, w) \) is a Gaussian

\[
p(y|x, w) \propto e^{-\frac{(y - m)^2}{2s^2}}.
\]

Then the log likelihood function is

\[
\log p(D|w) = \log \prod_i p(y_i|x_i, w_i) = C - \sum_i \frac{(y_i - m_i)^2}{2s_i^2},
\]

where \( C \) is a constant. Assuming that the standard deviation of each \( p(y_i|x, w) \) is equal,

\[
\log p(D|w) = C - \frac{1}{2s^2} \sum_i (y_i - m_i)^2.
\]

Plugging equation (B3) into equation (4), we can drop the constant \( C \) since equation (4) will be minimized:

\[
F(D, \theta) \approx \frac{1}{N} \sum_{j=1}^{N} [\log q(w^{(j)}|\theta) - \log p(w^{(j)})] + \frac{1}{2s^2} \sum_i (y_i - m_i)^2.
\]

Note that the second term of equation (B4) is proportional to the mean squared error between data \( \{y_i\} \) and mean \( \{m_i\} \):

\[
\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - m_i)^2.
\]

Thus, we can replace the second term in equation (B4) with the following:

\[
F(D, \theta) \approx \frac{1}{N} \sum_{j=1}^{N} [\log q(w^{(j)}|\theta) - \log p(w^{(j)})] + \text{MSE}.
\]

**Appendix C. Training BCNNs on noisy data**

Here we show results from a CM-BCNN trained on noisy data to address why we did not choose to inject background noise into the training data of the neural networks discussed in the main text.

It is a common practice in the field of deep learning to add background noise to training images to avoid overfitting and improve generalizability to other data sets. To see if this approach was beneficial for training the neural networks discussed in this text, we generated a new training set with the same parameters as the clean training set (see table 1), but with a very small amount of added background noise (see table C1). We also generated Test Set III, a data set generated using the same parameters as this noisy training set. These parameters are recorded in table C1.

After training the CM-BCNN on the noisy training data set (generated using the parameters in table C1), we test this ‘noisy’ CM-BCNN on Test Sets I and II in order to compare it to its clean counterpart. We also test the noisy CM-BCNN on Test Set III and compare its performance to that of OpenPIV.

We find that on Test Set I, the noisy CM-BCNN slightly outperforms the clean CM-BCNN on \( x \)-displacements (figure A1, \( R^2 = 0.994 \) compared to 0.993, 98.90% of true...
Figure A2. RMSE (pixels) for the CM-BCNN trained on noisy data versus error contributing parameters: particle displacement magnitude (A), particle shear (B), particle density (C), out of plane motion (D), particle diameter (E), and background noise level (F). Comparing these results to the equivalent results from the CM-BCNN trained on clean data (rightmost column, figures 6–11), the CM-BCNN trained on noisy data is far less accurate in response to ramping the error contributing parameters of shear, density, out of plane motion, and diameter, and slightly less accurate in response to ramping particle displacement magnitude. We suspect this lack of accuracy when ramping most error-contributing parameters is due to added background noise during training destroying patterns in the image data originating from particle shear, out of plane motion, etc. As expected, the CM-BCNN is slightly more robust to ramping background noise (F) than its clean counterpart (figure 11).

In the first test set, the clean CM-BCNN achieves slightly higher accuracy on the x-displacements contained in 95% confidence interval compared to 97.65%, but is slightly worse on the y-displacements (figure S3 and S7, $R^2 = 0.994$ compared to 0.996, 98.85% of true x-displacements contained in 95% confidence interval compared to 99.15%).

On Test Set II, the noisy CM-BCNN is unsurprisingly slightly more accurate than the clean CM-BCNN throughout ramping background noise. However, the noisy CM-BCNN is slightly less accurate than the clean CM-BCNN on zero noise, which may be a result of the stochastic nature of training procedures. Additionally, the noisy CM-BCNN is much less accurate than the clean CM-BCNN during ramps of other error-contributing parameters. The clean CM-BCNN especially outperforms the noisy CM-BCNN over ramps of particle shear (figure A2(B)), particle density (figure A2(C)), out of plane motion (figure A2(D)), and particle diameter (figure A2(E)). This suggests that adding small amounts of background noise to the training data is sufficient to prevent the CM-BCNN from recognizing patterns, especially when inputs are subject to shear, out of plane motion, etc. As stated in the main text, the CM-BCNN’s intolerance to background noise during training likely originates because the addition of small amounts of background noise to images can result in large variations in their correlation map compared to that of the ‘clean’ images. This intolerance of noise in the training data may be rectified by using more complex architectures which include the preprocessing step of feature extraction via cross correlation, and which may be able to detect more nuanced features in image data despite the presence of noise. This is a topic for future study.

On Test Set III, the noisy CM-BCNN outperforms OpenPIV in both the x and y directions (figures A3 and S8), with far less large outliers. This is unsurprising, as OpenPIV is not particularly robust to background noise.

Figure A3. Predicted vs true particle x-displacements (pixels) on Test Set III for the CM-BCNN trained on image data with added background noise (A) and OpenPIV (B). The CM-BCNN performs much better than OpenPIV on the noisy dataset, with an $R^2$ value of 0.969 as opposed to −1.231. Additionally, the CM-BCNN predicts 97.45% of true particle displacements in its 95% confidence interval, a percentage comparable to the CM-BCNN trained on clean training data (figure 5).
Due to the noisy CM-BCNN’s larger errors in response to ramping error-contributing parameters, we choose to present neural networks trained on noise-free data in the main text. As stated previously, future work includes adjusting network architectures to be more robust to noisy training data.

Appendix D. Training BCNNs on expanded particle displacement range

Here we show results from the ‘expanded’ CM-BCNN, or the CM-BCNN trained on data with particle displacements expanded to \([-12, 12]\) pixels from \([-4, 4]\) pixels. The expanded CM-BCNN was used predict particle displacements in figures 14–16 from a pair of real images. These real images are from Case A of the 1st International PIV Challenge [38].

We trained the expanded CM-BCNN in order to (a) demonstrate how a CM-BCNN can be trained to predict larger particle displacements, and (b) respond appropriately to a real image task with potentially large particle displacements. To do this, we generated a new training set with the same parameters as the clean training set (see table), but with particle displacements ranging between \([-12, 12]\) pixels (see table D1). This training set consists of 32,000 interrogation window pairs extracted from 32,000 image pairs. We also generated Test Set IV, a data set generated using the same parameters as this expanded training set. These parameters are recorded in table D1.

Table D1. Parameters for ‘expanded’ training data and Test Set IV.

| Parameter                        | Sample range          |
|----------------------------------|-----------------------|
| Diameter of particles (pixels)   | [1.0, 5.0]            |
| Particle density (particles/square pixel) | [0.012, 0.117] |
| Strength of shear (multiplier)   | [0.0, 0.02]           |
| Angle of shear (degrees)         | [0, 360]              |
| x-displacement (pixels)          | \([-12, 12]\)         |
| y-displacement (pixels)          | \([-12, 12]\)         |
| Out-of-plane displacement (pixels) | [0, 0.3]            |
| Noise level (fraction of standard deviation of image) | 0.0 |

After training the expanded CM-BCNN on the training dataset with expanded particle displacements, we test the expanded CM-BCNN on Test Sets I and II in order to compare it to its counterpart trained on particle displacements ranging between \([-4, 4]\) pixels. We also test the expanded CM-BCNN on Test Set IV and compare its performance to that of OpenPIV.

We find that on Test Set I, the expanded CM-BCNN slightly underperforms the non-expanded CM-BCNN on x-displacements (figure A4, \(R^2 = 0.992\) compared to 0.993, 96.30% of true x-displacements contained in 95% confidence interval compared to 97.65%), and is slightly worse than the non-expanded CM-BCNN on y-displacements (figure S9, \(R^2 = 0.994\) compared to 0.996, 96.70% of true x-displacements contained in 95% confidence interval compared to 99.15%).

On Test Set II, the expanded CM-BCNN is much less accurate than the non-expanded CM-BCNN throughout ramping background noise. We suspect this may occur because the range of particle displacements in the training data has been expanded. When a neural network trained on clean data with particle displacements ranging between \([-4, 4]\) pixels is presented with a noisy image, it will likely guess an output ranging between \([-4, 4]\) pixels despite not ‘knowing’ how to accurately assign a particle displacement (since the neural network was not exposed to background noise during training). We therefore assume that this output is uniformly drawn from the range \([-4, 4]\) pixels, and is probably incorrect. If the true particle displacement of the noisy image is small (\(\sim 1\) pixel), the error of this output is not large.

However, when a neural network trains on clean data with particle displacements ranging between \([-12, 12]\) pixels and it is presented with a noisy image with small particle displacements (\(\sim 1\)), it will likely predict an output ranging between \([-12, 12]\) pixels, thus increasing the RMS error in particle displacement by roughly a factor of \(3\sqrt{2}\) (for images with a sufficient amount of noise). Indeed, we can see in figure A5 that the RMS error in particle displacement versus background noise level curve for the expanded CM-BCNN asymptotes to \(\sim 2\), while its non-expanded equivalent asymptotes to 0.5 (figure 11(C)). This is roughly consistent with the relationship \(0.5 \times 3\sqrt{2} \sim 2\). This statistical artifact is potentially important when evaluating the accuracy of neural networks’ response to background noise. As stated in the main text, this lack of background noise tolerance may be remedied by the combination of expanding the network architecture and training on noisy data, and is the subject of future work.

However, the expanded CM-BCNN is only slightly less accurate or roughly as accurate as its non-expanded counterpart (see figures A5(B)–(E) versus figures 9 and 10) while...
Figure A5. RMSE (pixels) for the ‘expanded’ CM-BCNN, or CM-BCNN trained on data with an expanded particle displacement range versus error contributing parameters: particle displacement magnitude (A), particle shear (B), particle density (C), out of plane motion (D), particle diameter (E), and background noise level (F). Comparing these results to the equivalent results from the CM-BCNN trained on data with particle displacements ranging between $[-4, 4]$ pixels (rightmost column, figures 6–11), the expanded CM-BCNN is far less accurate in response to ramping the error contributing parameters of background noise level, while only slightly less accurate or around the same accuracy in response to ramping the other error-contributing parameters of particle displacement, shear, out of plane motion, density, and diameter. See appendix D for an explanation of why this may occur. The expanded CM-BCNN may be made more tolerant to background noise by the combination of expanding the neural network architecture and training on noisy data. This is a topic for future study.

Figure A6. Predicted vs true particle $x$-displacements (pixels) on Test Set IV for the ‘expanded’ CM-BCNN, or CM-BCNN trained on image data with expanded particle displacement range (A) and OpenPIV (B). The expanded CM-BCNN performs much better than OpenPIV on the dataset with expanded particle displacements, with an $R^2$ value of 0.980 as opposed to 0.884. Additionally, the expanded CM-BCNN predicts far less large errors than OpenPIV and captures 84.05% of true particle displacements in its 95% confidence interval.

On Test Set IV, the expanded CM-BCNN outperforms OpenPIV in both the $x$ and $y$ directions (figure A6 and S10), with far less large outliers. This is unsurprising, as OpenPIV is prone to large outliers and increased errors with increased particle displacement range.

Appendix E. CM-BCNN generalizability

In main text, we have demonstrated that the CM-BCNN is an effective tool for simultaneously producing accurate particle displacement predictions and quantifying their uncertainty. A tempting use for the CM-BCNN is to apply it to correlation maps generated by existing, perhaps state-of-the-art PIV codes. It is therefore important to determine whether (a) the CM-BCNN is sufficiently general to estimate particle displacements and uncertainties from correlation maps originating from existing PIV algorithms and (b) if the CM-BCNN is sufficiently general, whether it is practical to do so. In this section, we test CM-BCNN’s ability to generalize to existing PIV algorithms by evaluating its performance on correlation maps generated by a simple multi-pass algorithm. The simplest case of a multi-pass PIV algorithm is a double pass algorithm. This algorithm follows the steps enumerated in table E1.

In a typical multi-pass PIV algorithm, steps (2) and (3) in table E1 are iterated many times such that the final estimate of
particle displacement converges to a stable value. Additionally, a typical multi-pass PIV algorithm involves removal of spurious displacements, or large errors in predicted particle displacement, in step (2). We accomplish this by defaulting to a single pass algorithm for particle displacements estimated in step (2) greater than 10 pixels. Similarly, we default to the single pass algorithm for second order particle displacements estimated in step (3) greater than 5 pixels.

Many widely used PIV codes \[51, 52\] utilize either multi-pass or multi-grid methods, both which involve iterative cross-correlation of interrogation window pairs, shifting an interrogation window by the estimated particle displacement found in the last algorithm iteration. This is because multi-pass PIV algorithms have been shown to reduce errors in estimated particle displacement by optimizing the number of particles contained within each interrogation window \[39, 53\].

Additionally, we note that window deformation is another commonly used method in popular PIV codes: we did not implement it in this work, as window deformation also requires the iterative use of interrogation windows offset by estimated particle displacements \[21\]. As a baseline test of whether correlation maps generated by PIV algorithms involving interrogation window offsets can be analyzed by the CM-BCNN, we implement the double pass algorithm enumerated in table \(E1\) using both the CM-BCNN and OpenPIV for comparison.

First, we implement the double pass algorithm described in table \(E1\) using OpenPIV and run this algorithm on Test Set I. In figure \(A7(B)\), we show the results of this calculation for displacements in the \(x\)-direction. The double pass PIV algorithm performs slightly better than the single pass algorithm on Test Set I, as its predicted vs true particle displacement has an \(R^2\) value of 0.8517 as opposed to the \(R^2\) of 0.8515 found for the single pass equivalent (figure \(5(G)\)). Results for displacements in the \(y\)-direction are similar and are shown in the supplement (figure \(S11(B)\)): the predicted vs true particle displacement curve for the double pass algorithm has an \(R^2\) value of 0.8854 while the single pass equivalent has an \(R^2\) value of 0.8850. Similarly, on Test Set II, the double pass PIV algorithm performs at about the same accuracy as its single pass counterpart (blue dashed lines, figure \(A10\)).

Next, we use the CM-BCNN to perform the double-pass algorithm. Note that we still use OpenPIV to calculate the interrogation window offsets. We choose to do this because when implementing the CM-BCNN to simultaneously provide particle displacement estimates and uncertainties from correlation maps calculated via an existing algorithm, it is more practical to not make any modifications to the existing algorithm and simply introduce the CM-BCNN afterwards.

In table \(E2\), we outline the double pass algorithm used with the CM-BCNN, step by step. We run this algorithm on Test Set I and show the results of this calculation for displacements in the \(x\)-direction in figure \(A7(A)\). The double pass CM-BCNN captures 90.60% of true particle displacements in its 95% confidence interval, a result not as accurate as that obtained by the single pass CM-BCNN, but still sufficient to suggest that the CM-BCNN can indeed be generalized to correlation maps generated through multi-pass algorithms, with some sacrifice of accuracy. Results for displacements in the \(y\)-direction are similar and are shown in the supplement (figure \(S11(A)\)).

This lack of accuracy is surprising, as one would expect that the double pass CM-BCNN would be more accurate than the single pass CM-BCNN, as was the case for the OpenPIV single and double pass algorithms. The reason for double pass CM-BCNN’s inaccuracy compared to its single pass equivalent is found in figures \(A8\) and \(A9\). In figure \(A8\), we plot the 1st order term of the predicted particle displacement on Test Set I estimated by step (4) of the double pass CM-BCNN algorithm detailed in table \(E2\) versus the OpenPIV 1st order particle displacement term (the interrogation window shift generated by

---

**Table E1.** Double pass PIV algorithm.

| Step | Operation |
|------|-----------|
| 1    | Select Interrogation Window A and Interrogation Window B on the sequential image pair Image Frame A and Image Frame B. During the first pass, Interrogation Window A and Interrogation Window B are centered about the same coordinates on both Image Frame A and Image Frame B. |
| 2    | Compute the cross-correlation map of Interrogation Window A and Interrogation Window B and extract an integer estimate of the particle displacement between them via OpenPIV. |
| 3    | Shift Interrogation Window B by the integer estimate of particle displacement calculated in (2). Compute the cross-correlation map of Interrogation Window A and Interrogation Window B and again extract the particle displacement between them via OpenPIV. |
| 4    | Add the particle displacements found in (2) and (3) together, yielding the final estimate of particle displacement. |

---

**Figure A7.** (A) Particle displacements (pixels) predicted by the CM-BCNN using a double pass algorithm versus true particle displacements in the \(x\)-direction. The double pass algorithm used with the CM-BCNN is detailed in table \(E2\). Although not as accurate as the single pass CM-BCNN shown in figure \(5\), the double pass CM-BCNN implementation shows good accuracy on Test Set I, suggesting that the CM-BCNN is robust enough to generalize to correlation maps generated via multiple algorithms. (B) Particle displacements (pixels) predicted by OpenPIV using a double pass algorithm vs true particle displacements in the \(x\)-direction. Results for the equivalent of (A) and (B) in the \(y\)-direction are similar and are shown in the supplement (figure \(S11\)).
Table E2. Double pass CM-BCNN algorithm.

| Step | Operation |
|------|-----------|
| 1    | Select Interrogation Window A and Interrogation Window B on the sequential image pair Image Frame A and Image Frame B. During the first pass, Interrogation Window A and Interrogation Window B are centered about the same coordinates on both Image Frame A and Image Frame B. |
| 2    | Compute the cross-correlation map of Interrogation Window A and Interrogation Window B and extract an integer estimate of the particle displacement between them via OpenPIV. |
| 3    | Shift Interrogation Window B by the integer estimate of particle displacement calculated in (2). Compute the cross-correlation map of Interrogation Window A and Interrogation Window B and extract the particle displacement between them via the CM-BCNN, yielding a mean and standard deviation prediction. |
| 4    | Use the CM-BCNN to extract a mean and standard deviation particle displacement from the correlation map calculated in (2). Discard the non-integer part of the mean particle displacement prediction. |
| 5    | Add the particle displacements found in (3) and (4) together, yielding the final estimate of particle displacement. Propagate the uncertainties found in (3) and (4) to obtain the uncertainty of the final particle displacement prediction. |

In figure A9(A), we plot the 2nd order term of the predicted particle displacement on Test Set I generated by step (4) of the double pass CM-BCNN algorithm detailed in table E2 versus the true 2nd order particle displacement term (pixels): the difference between the true x-displacement and the interrogation window shift generated by step (2) of the algorithm described in table E2. Error bars are standard deviations. (B) The OpenPIV equivalent of (A): the 2nd order term of the predicted particle displacement (in pixels) on Test Set I generated by step (3) of the double pass OpenPIV algorithm detailed in table E1 versus the true 2nd order particle displacement term (pixels): the difference between the true x-displacement and the interrogation window shift generated by step (2) of the algorithm described in table E1. Note that the CM-BCNN struggles to match the accuracy of OpenPIV on correlation maps generated from multi-pass algorithms, such as those used to generate this figure. Results for the y-direction are similar and are shown in the supplement (figure S12).
in RMS mean displacement below 1 pixel for most ranges of error contributing parameters tested, implying that although some accuracy was lost, the CM-BCNN is still able to recognize features of correlation maps generated by double pass algorithms.

We conclude that with a moderate loss in accuracy, the CM-BCNN generalizes successfully to correlation maps generated by multi-pass algorithms. In future work, techniques such as augmenting the training set (i.e. rotating and adding background noise to training data), optimizing the network architecture, fine-tuning the CM-BCNN with correlation maps produced by multi-pass algorithms, and/or including multi-pass correlation maps in the training data could be implemented to decrease the loss in accuracy observed upon applying the CM-BCNN to multi-pass correlation maps.

We end our discussion of the application of the CM-BCNN to multi-pass algorithms by noting a mathematical consideration. Since the uncertainty of each particle displacement prediction made on each correlation map must be propagated to yield the final particle displacement prediction uncertainty as described in step (5) of the double pass CM-BCNN algorithm detailed in table E2, with every additional pass the final particle displacement uncertainty grows monotonically. Thus, if the spread is large on particle displacement distributions predicted by the CM-BCNN for a given correlation map, it may be impractical to implement a multi-pass algorithm.

**ORCID ID**

Mia C Morrell [https://orcid.org/0000-0002-1811-1337](https://orcid.org/0000-0002-1811-1337)

**References**

[1] Violato D and Scarano F 2011 Three-dimensional evolution of flow structures in transitional circular and chevron jets *Phys. Fluids* 23 124104

[2] Violato D and Scarano F 2013 Three-dimensional vortex analysis and aeroacoustic source characterization of jet core breakdown *Phys. Fluids* 25 015112

[3] Kühler C, Scholz U and Ortmanns J 2006 Wall-shear stress and near-wall turbulence measurements up to single pixel resolution by means of long-distance micro-PIV *Exp. Fluids* 41 327–41

[4] Lindken R, Silvestro F, Westerweel J and Nieuwstadt F 2002 Turbulence measurements with micro-PIV in large-scale pipe flow *Proc. 11th Int. Symp. Applications of Laser Techniques to Fluid Mechanics* (Lisbon, Portugal, 2002) 12.1

[5] Harirahan P et al 2018 Inter-laboratory characterization of the velocity field in the FDA blood pump model using particle image velocimetry (PIV) *Cardiovasc. Eng. Technol.* 9 623–40

[6] Kheradvar A and Falahatpisheh A 2012 The effects of dynamic saddle annulus and leaflet length on transmural flow pattern and leaflet stress of a bileaflet bioprosthetic mitral valve *J. Heart Valve Dis.* 21 225–33
Wieneke B 2015 PIV uncertainty quantification from Smith B L and Oberkampf W L 2014 Limitations of and Raffel M, Willert C and Kompenhans J 1998 Adrian R 2005 Twenty years of particle image velocimetry Timmins B, Wilson B, Smith B and Vlachos P 2012 A method Nguyen T, Goth N, Jones P, Vaghetto R and Hassan Y 2018 Scarano F 2001 Iterative image deformation methods in PIV Zhu W, van Hout R and Katz J 2007 PIV measurements in the Faraday M and Refsgaard J 2017 Advanced PIV Measurements of Atmospheric Boundary Layer Flow Lecun Y and Bengio Y 1995 Convolutional networks for estimating time-averaged velocity fields Exp. Fluids 39 159–69. Guı̈ L and Wereley S 2002 A correlation-based continuous window-shift technique to reduce the peak-locking effect in digital PIV image evaluation Exp. Fluids 32 506–17 Meinhart C, Wereley S and Santiago J 2000 A PIV algorithm for estimating time-averaged velocity fields J. Fluids Eng. 122 285–9. Westerweel J 1999 Fundamentals of digital particle image velocimetry Meas. Sci. Technol. 8 1379. Sciacchitano A, Scarano F and Wieneke B 2012 Multi-frame pyramid correlation for time-resolved PIV Exp. Fluids 53 1087–105. Eckstein A and Vlachos P 2009 Digital particle image velocimetry (DPIV) robust phase correlation Meas. Sci. Technol. 20 055401. Huang H T, Fiedler H and Wang J 1993 Limitation and improvement of PIV Exp. Fluids 15 168–74. Scarano F 2001 Iterative image deformation methods in PIV Meas. Sci. Technol. 13 R1. Cai S, Zhou S, Chao X and Gao Q 2019 Dense motion estimation of particle images via a convolutional neural network Exp. Fluids 60 73 Cai S, Jiang L, Gao Q, Xu C and Runjie W 2019 Particle image velocimetry based on a deep learning motion estimator IEEE Trans. Instrum. Meas. 69 3538–54. Rabault J, Kolaas J and Jensen A 2017 Performing particle image velocimetry using artificial neural networks: a proof-of-concept Meas. Sci. Technol. 28 125301 Lee Y, Yang H and Yin Z 2017 PIV-DCNN: cascaded deep convolutional neural networks for particle image velocimetry Exp. Fluids 58 171.
[47] Wen Y, Vicol P, Jimmy B, Tran D and Grosse R 2018 Flipout: efficient pseudo-independent weight perturbations on mini-batches Int. Conf. on Learning Representations

[48] Gida H B, Gurka R, and Liberzon A 2020 OpenPIV-Matlab—an open-source software for particle image velocimetry; test case: birds’ aerodynamics (arXiv:2007.12967)

[49] Liberzon A et al 2020 OpenPIV/openpiv-python: OpenPIV—Python (v0.22.2) with a new extended search PIV grid option

[50] Hui T-W, Tang X and Loy C C 2018 LiteFlowNet: a lightweight convolutional neural network for optical flow estimation (arXiv:1805.07036)

[51] Scarano F and Riethmuller M L 2012 Advances in iterative multigrid PIV image processing Exp. Fluids 29 S051–60

[52] Michel Stanislas K. Okamoto C K, Westerweel J and Scarano F 2008 Main results of the third international PIV challenge Exp. Fluids 45 27–71

[53] Adrian R J and Westerweel J 2011 Particle Image Velocimetry (New York: Cambridge University Press)