MULTIFIDELITY DATA FUSION IN CONVOLUTIONAL ENCODER/DECODER ASSEMBLY NETWORKS FOR COMPUTATIONAL FLUID DYNAMICS

A PREPRINT

Lauren Partin*
ACMS Department
University of Notre Dame
South Bend, IN
lhensley@nd.edu

Gianluca Geraci
Sandia National Laboratories
Albuquerque, NM
ggeraci@sandia.gov

Ahmad A. Rushdi
Sandia National Laboratories
Albuquerque, NM
arushdi@sandia.gov

Michael S. Eldred
Sandia National Laboratories
Albuquerque, NM
mseldre@sandia.gov

Daniele E. Schiavazzi
ACMS Department
University of Notre Dame
South Bend, IN
dschiavazzi@nd.edu

May 23, 2022

ABSTRACT

We analyze the regression accuracy of convolutional neural networks assembled from encoders, decoders and skip connections and trained with multifidelity data. These networks benefit from a significant reduction in the number of trainable parameters with respect to an equivalent fully connected network. These architectures are also versatile with respect to the input and output dimensionality. For example, encoder-decoder, decoder-encoder or decoder-encoder-decoder architectures are well suited to learn mappings between input and outputs of any dimensionality. We demonstrate the accuracy produced by such architectures when trained on a few high-fidelity and many low-fidelity data generated from models ranging from one-dimensional functions to Poisson equation solvers in two-dimensions. We finally discuss a number of implementation choices that improve the reliability of the uncertainty estimates generated by a dropblock regularizer, and compare uncertainty estimates among low-, high- and multi-fidelity approaches.

1 Introduction

The analysis of physical phenomena through their mathematical or numerical replicas is a common practice in engineering and science, providing the analyst with the ability to predict the behavior of a system outside a number of observed outputs. Simulation of complex phenomena, for example characterized by multiple interacting physics, may require a substantial computational effort, and the availability of sufficient resources may be a key factor in the ability to answer the scientific questions of interest. However, it is often possible to combine accurate but expensive high-fidelity simulations with lower-fidelity simulations that provide approximations at lower cost, in order to optimize efficiency while retaining accuracy.

This study focuses on generating multifidelity (MF) surrogate models designed to combine information from a few high-fidelity (HF) model solutions and many low-fidelity (LF) approximations of varying accuracy. More specifically, we focus on data-driven multifidelity surrogates in the machine learning context. Existing approaches in the literature include student-teacher networks with the ability to handle datasets with variable annotation quality [4], surrogates

*

trained using transfer learning between two model fidelities [3], and fully connected neural networks combining three sub-networks designed to learn a LF representation, the correlation between a LF and a HF representation, and to minimize a physics-based residual loss [16]. Other approaches utilize Bayesian neural networks [15], or combine convolutional and fully connected networks to learn the discrepancy between increasingly fine discretizations of a given PDE solution, projected on a common mesh [24].

Our approach is inspired by the recent successes in image classification and segmentation tasks shown by deep convolutional encoder-decoder networks (see, e.g., [17]). While multifidelity data fusion has been mainly demonstrated for fully connected networks or for ensembles of hybrid convolutional and fully connected networks [24], no approach has focused on convolutional networks assembled from encoders, decoders and skip connections, where the model fidelities are learned simulataneously, following an *all-at-once* training paradigm. Note how convolutions are essential to reduce the number of weights with respect to fully connected networks when the input, the output or both are high-dimensional, as discussed in our recent work [19].

We also focus on quantifying the *predictive uncertainty* in the network outputs, i.e., we want to characterize the variability of the predicted quantities of interest, a paradigm commonly referred to as "UQ for ML", analyzing the uncertainty in predictions that are inherent when using a machine-learned surrogate model. We see this as a *model form* uncertainty that only relates to how the information flows through the selected multifidelity network, and, in traditional uncertainty propagation studies, complements the *aleatoric* uncertainty arising from a set of stochastic model inputs. We refer instead to the situation where a machine learning surrogate is employed as an inexpensive surrogate for uncertainty quantification studies as "ML for UQ". In this paper, we endow our multifidelity network predictions with uncertainty estimates, to facilitate future work in effective surrogate model management for forward uncertainty quantification studies.

Many different approaches have been proposed in the literature to quantify predictive uncertainty in neural network outputs [8, 11, 1]. Among these, dropout layers [23] offer a simple and computationally appealing solution to drop neurons according to some probability, providing, at the same time, regularization and variance estimates. Their interpretation in terms of ensemble networks also helps to understand their theoretical propoerties [2, 5]. However, their performance has been mainly assessed on neural networks with fully connected layers. In this study, we use dropblocks [6], i.e., adaptations of dropout layers showing improved performance on convolutional architectures.

This paper is organized as follows: §2 introduces the problems of interest including one-dimensional function approximation from MF datasets, and prediction of high-dimensional responses from computational fluid dynamics solvers. §3 provides the details of the convolutional networks used in the study, how the training, testing and validation datasets were selected, the loss function formulation, and the augmentation of such networks with LF predictors. Uncertainty estimates through MC-dropblock are discussed in §4. Results are summarized in §5, whereas conclusions and plans for future work are provided in §6.

2 Problem description

We focus on *dense* regression for both low- and high-dimensional inputs and outputs, and examine how low-fidelity representations can be leveraged to accelerate training and improve the accuracy of high-fidelity predictions from limited data.

2.1 One-dimensional multifidelity function approximation

We analyze multiple examples in one-dimensional regression, each consisting of two linearly and/or nonlinearly correlated LF and HF functions. Fewer training examples are available for the HF model than for the LF model [16]. The first example consists of two linearly correlated continuous functions, defined as

$$y_L(x) = (1/2)(6x-2)^2\sin(12x-4) + 10(x-1/2) - 5 \quad (1)$$

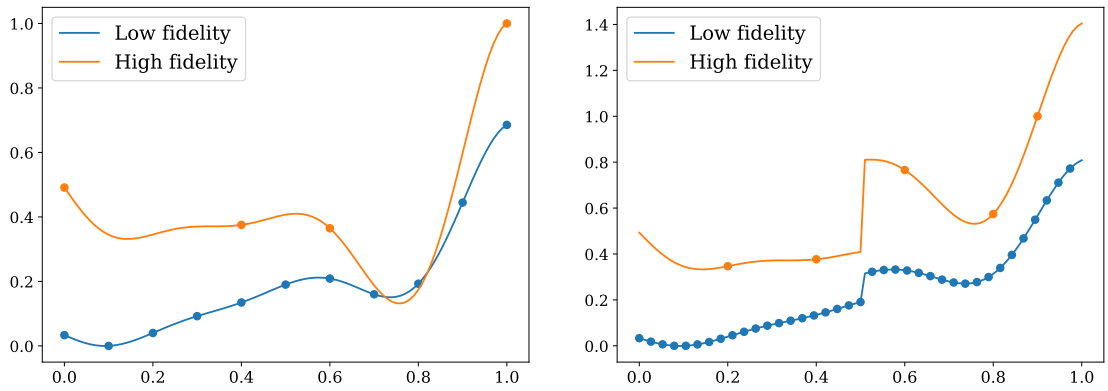
$$y_H(x) = (6x-2)^2\sin(12x-4), \quad (2)$$

where 11 and 4 samples, are provided for y_L and y_H, respectively, as shown in Fig. 1(a). In the second example, we consider two linearly correlated discontinuous functions expressed as

$$y_L(x) = \begin{cases} l(x) = 0.5(6x-2)^2\sin(12x-4) + 10(x-0.5) - 5 & 0 \le x \le 0.5 \\ 3 + l(x) & 0.5 < x \le 1 \end{cases} \quad (3)$$

$$y_H(x) = \begin{cases} h(x) = 2y_L(x) - 20x + 20 & 0 \le x \le 0.5 \\ 4 + h(x) & 0.5 < x \le 1, \end{cases} \quad (4)$$

with 38 and 5 training samples for y_L and y_H, respectively, as shown in Fig. 1(b).



(a) (b)

Figure 1: LF and HF functions and their training locations from (a) Eqs. (1), (2) and (b) Eqs. (3), (4). Function values are rescaled such that $y_H(x), y_L(x) \in [0, 1]$ for x in the training set.

2.2 High-dimensional dense regression

For the high-dimensional dense regression case, we focus on an application in computational fluid dynamics, where we measure a noisy realization of the indicator function (referred to as a scalar *concentration*) for a fluid domain Ω_f and a noisy velocity field, and we are interested in predicting the fluid pressure distribution in Ω_f. The pressure up to a constant can be computed through a reformulation of the imcompressible Navier Stokes equations as a Poisson pressure equation with appropriate boundary conditions [22]. This approach, however, may require the solution of a partial differential equation on a large computational grid and training a neural network to perform such task could provide a much faster and computationally attractive alternative. Note that, unlike physics-informed neural networks (PINN [20]), we would like to learn a relation between concentration/velocity and pressure, rather than pressure as a function of space and time.

Under the assumption that no body force is acting on the fluid, the Poisson equation for the pressure p can be written as

$$\Delta p = \boldsymbol{\nabla} \cdot \mathbf{f} = \boldsymbol{\nabla} \cdot \left[-\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) + \mu \, \Delta \mathbf{u} \right], \tag{5}$$

and only Neumann boundary conditions are applied equal to the flux of $\mathbf{f}$ across the smooth boundary $\partial\Omega_f$ of Ω_f [22].

The problem is discretized through a structured grid Ω with $m_x \cdot m_y \cdot m_z = m$ cells, where the fluid region is identified as $c_b = 1$ in a binary *concentration* map $c_b : \mathbb{R}^3 \to \{0, 1\}$, but, in practice, noise makes this function non-binary, with *measured* concentration expressed as $c : \mathbb{R}^3 \to \mathbb{R}$. Our objective is to train a neural network surrogate to quickly evaluate the map

$$f(\mathbf{c}, \mathbf{u}) = \mathbf{p}, \text{ where } f : \mathbb{R}^m \times \mathbb{R}^{m \times 3} \to \mathbb{R}^m, \tag{6}$$

which, given the concentrations and velocity distributions over Ω, returns the spatial pressure distribution on Ω_f. We also want to investigate the possibility to increase training efficiency and accuracy using a multifidelity dataset containing pressure representations with increasingly coarser resolution. Multifidelity training examples are obtained by combining a small number of HF examples, resulting from a Poisson pressure equation finite element solver, with a large number of solutions from the same solver, but evaluated on coarser discretizations.

2.3 Low- to high-dimensional dense regression for CFD

We consider a problem with the same pressure outputs as that in §2.2, but with only two inputs, i.e., the radius r of the cylindrical fluid domain and the maximum velocity v_{max}, as these parameters are sufficient to fully specify the geometry and velocity distribution in a Hagen-Poiseuille flow, as shown in Fig. 5(b). In other words, this case represents the surrogate construction for a random field (the pressure) given only two input parameters. This low-to-high dimensional regression problem may not be easy to solve for traditional UQ approaches. For example, techniques like generalized polynomial chaos [25], could be used to obtain a pressure estimate at each pixel in the fluid domain, but additional structure would need to be specified to account for the spatial correlation of the resulting pressure field.

3 Network architectures

For all problems discussed above, we employ a network architecture assembled from convolutional encoders, decoders and skip connections. A convolutional encoder [14] is composed of alternating layers of convolutions and pooling (i.e., downsampling), which generate a compressed feature representation. A convolutional decoder, on the other hand, is composed of alternating layers of convolutions and upsampling. For dense regression problems, the encoder and decoder are symmetric, so that the input and output dimensionality of the network is the same. A detailed description of three network architectures (used for the problems discussed in §2.1, §2.2 and §2.3, respectively) and their hyperparameters is offered next.

3.1 Decoder-encoder architecture for one-dimensional regression

For the one-dimensional regression problem presented in §2.1, the LF predictor is generated using a decoder-encoder network with skip connections. The HF predictor is obtained as a sum of a convolution between the LF and x, and by processing the LF through two additional convolutional layers, designed to capture the non linear correlation between the HF and LF outputs. Note that the one-dimensional regression network in Fig. 2 only predicts a single low-fidelity estimator LF.

The number of kernels per convolutional layer is equal to 16, 16, 8, 8 (4 layers) in the decoder, 8, 8, 16, 16, 8 (5 layers) in the encoder and 8, 1 (2 layers) in the nonlinear correlation network between the LF and HF. The selected kernel sizes are 1, 1, 2, 2 in the decoder, 2, 2, 1, 1 in the encoder, 2, 1 in the nonlinear correlation, and 1 in the linear correlation. A stride of 1 and padding are used to keep inputs and outputs of the same size. Training is performed using the Adam optimizer [12] utilizing a step learning rate scheduler with decay 0.9, where the step size and initial learning rate is determined by optimizing for the best fit of the given training dataset. A batch size of 1 is used due to the limited number of training examples. In addition, the functions are rescaled to the range $[0, 1]$ based on the maximum and minimum value in the training set, which allows for a consistent use of the same optimizer across functions. The network layout is illustrated in Fig. 2.

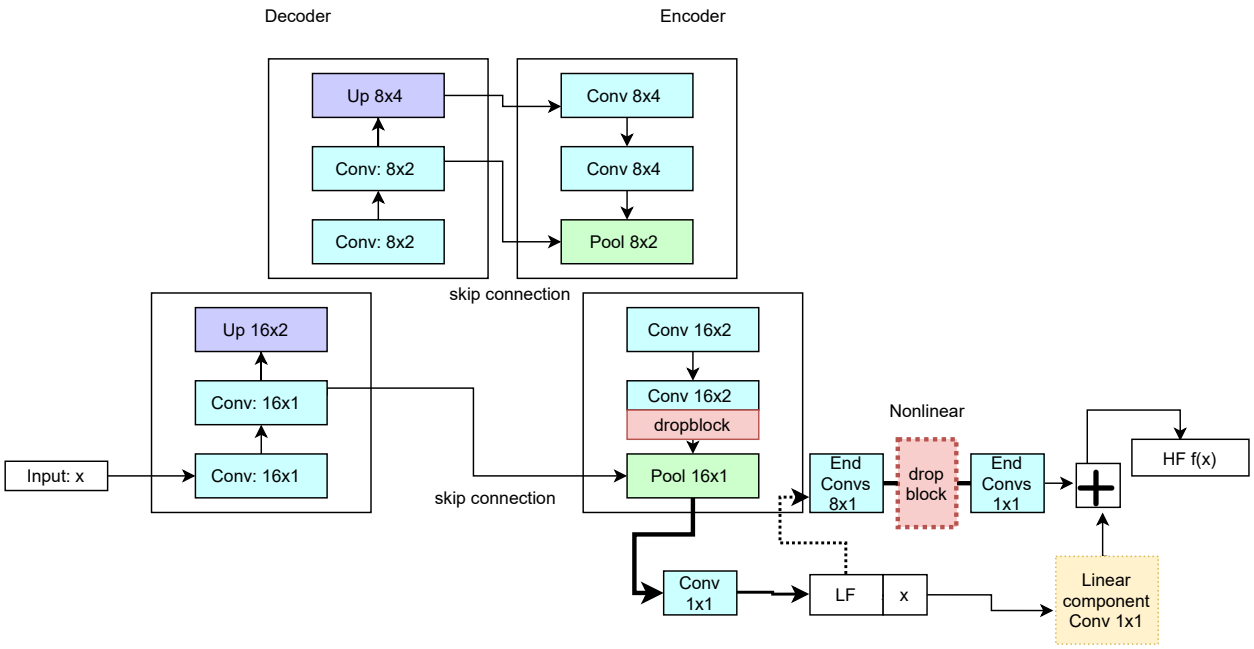


Figure 2: Multi-fidelity decoder-encoder convolutional network architecture for one-dimensional regression with optional explicit feedback. The HF output is a linear combination of the nonlinear and linear portion of the network, $HF = \gamma Linear(LF, x) + (1 - \gamma)Nonlinear(LF, x)$, where γ is a parameter learned by the network.

3.2 Encoder-decoder architecture for high-dimensional regression

The selected architecture for high-dimensional dense regression resembles the popular U-Net [21]. The U-Net architecture and its variants have shown great performance in terms of accuracy and training speed for segmentation tasks, even under limited training data [10]. Our implementation predicts the relative pressure on a two-dimensional structured grid from velocity and concentration inputs provided on each pixel.

More specifically, we consider both input and output images with 64×64 pixels. The number of kernels per convolution layer in the encoder are 16, 16, 32, 32, 64, 64, 128, 128 (8 layers), respectively, while those in the decoder are instead 64, 64, 32, 32, 16, 16, 32, 32, 16, 1 (10 layers). Each convolution layer is followed by a batch normalization layer and ReLU activation, which becomes a simple identity after the final convolution layer. Additionally, each convolution is performed with a kernel of size 3, padding 1 and stride 1 (i.e. same size output), followed by max pooling. Training is performed using the Adam optimizer with learning rate 0.01, using a step learning rate scheduler, which decays by a factor 0.9 every 200 epochs, with a batch size of 16. We utilize the Xavier weight initialization scheme following a normal distribution [7].

A multifidelity network is obtained by extracting a LF representation of increasing resolution at each stage of the decoder. A term for each LF predictor is then added to the loss function, so these LF representations are accurately learned. This is depicted in Fig. 3 where the models are ordered as $LF1$, $LF2$, $LF3$, HF, i.e., from the coarsest to the finest resolutions.

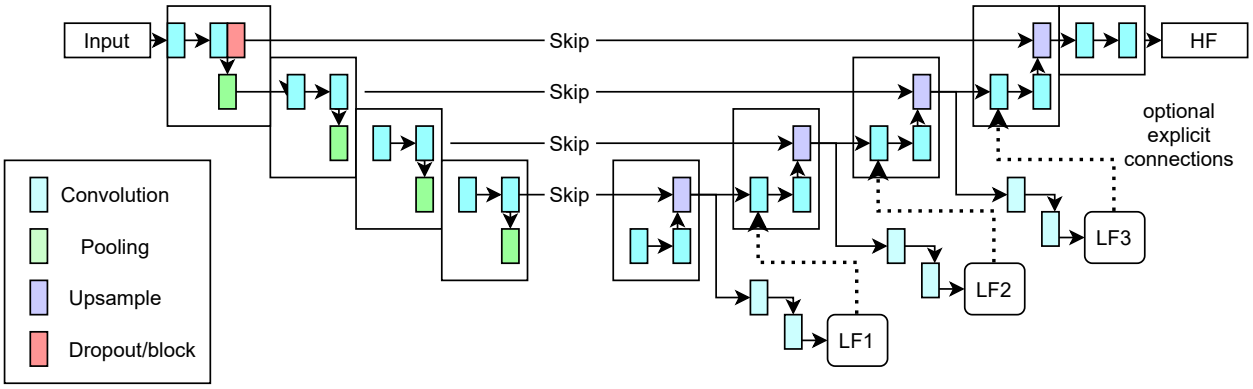


Figure 3: Multi-fidelity decoder-encoder convolutional network architecture for high-dimensional dense regression. For explicit LF-HF feedback, the connections drawn with dotted lines are included, while they are omitted when the LF-HF feedback is implicit. Outputs $LF1$, $LF2$, $LF3$, HF are ordered in terms of resolution from coarsest to finest.

3.3 Encoder architecture for low- to high-dimensional regression

The network selected for low- to high-dimensional dense regression is shown in Fig. 4. The two dimensional input is significantly upsampled to generate 64×64 output images. This is achieved with a single decoder, choosing the network depth to enforce the correct output dimensions without padding, i.e. 6 upsampling layers, each with a scale factor of 2. The number of kernels per convolution layer are 320, 320, 160, 160, 80, 80, 40, 40, 20, 20, 10, 10, 5, 5, (14 layers) which decrease by a factor of 2 at each decoder block.

Similar to the network in the previous section, a LF prediction is generated at each decoder stage, as shown in Fig. 4. The LF representations are ordered as $LF1$, $LF2$, $LF3$, HF, i.e., from the coarsest to the finest resolutions.

3.4 High- to low-fidelity representation coupling

For each of these three networks, we consider an implicit and an explicit coupling between the LF and the HF representations. In the first *implicit* case, the LF predictors are not directly propagated towards the network output, as shown in Fig. 3 and 4, where the dotted arrows are omitted. However, forcing the upstream stages to learn accurate coarse pressure representations clearly affects the accuracy of the high-fidelity prediction. Propagation of information through the dotted arrows is instead allowed for the *explicit* feedback mechanism, meaning that the LF predictions are propagated through the following stages of the decoder. Finally, the network selected for one-dimensional regression is shown in Fig. 2, where implicit and explicit feedback is again obtained by omitting or including the information transfer through the dotted arrow. When the HF and LF truth belong to the same space and are correlated, an explicit

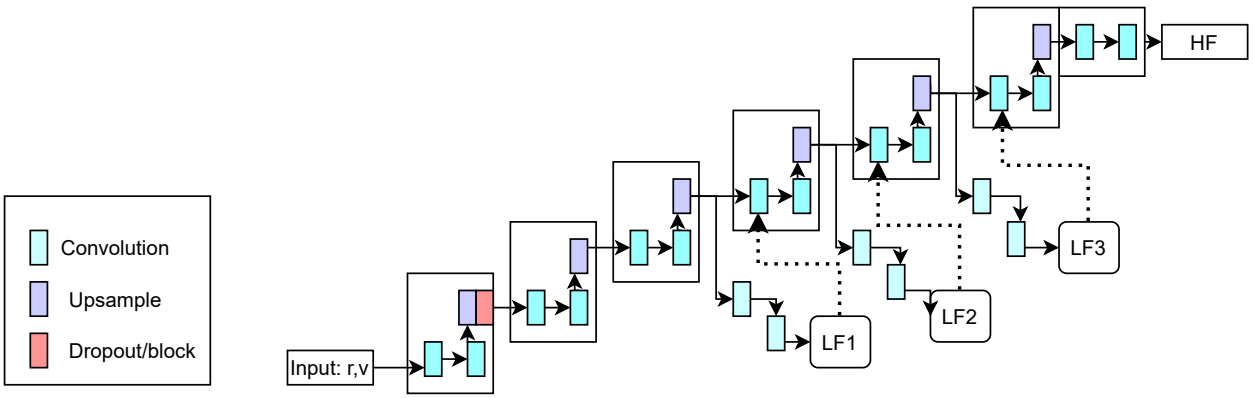


Figure 4: Multi-fidelity convolutional decoder architecture for low- to high-dimensional dense regression. For explicit LF-HF feedback, the connections drawn with dotted lines are included, while they are omitted when the LF-HF feedback is implicit. Outputs $LF1$, $LF2$, $LF3$, HF are ordered in terms of resolution from coarsest to finest.

connection helps in capturing the relationship between the LF and HF, such as in the two one-dimensional regression problems discussed in Section 3.1. However, in the high- and low- to high-dimensional regression problems examined in this work, it's unclear that an explicit connection would be beneficial, since the LFs and HF live on different spaces. In such a case, an implicit connection appears to force the network between two successive LFs to learn some implicit discrepancy between their corresponding feature channels, ultimately leading to improved HF predictions.

3.5 Training datasets and multifidelity loss

For one-dimensional regression, we use the same training data as detailed in [16]. For the high-dimensional and low- to high-dimensional cases, the dataset consists of two-dimensional slices from a Hagen-Poiseuille flow in a cylindrical fluid domain Ω_f. The solution is axisymmetric and therefore equal with respect to any plane that includes the cylinder axis; therefore a two-dimensional slice is sufficient to fully describe the flow. The Hagen-Poiseuille flow dataset (denoted as *HF 116/0*) consists of 116 HF realizations corresponding to random values of the maximum velocity and cylinder radius parameters (see Fig. 5). Training, validation and testing datasets are obtained using 60/20/20 split ratios, resulting in 116, 49, 35 HF images, respectively. A second dataset (*HF 32/0*) was also generated by randomly subsampling 32 of the 116 HF realizations.

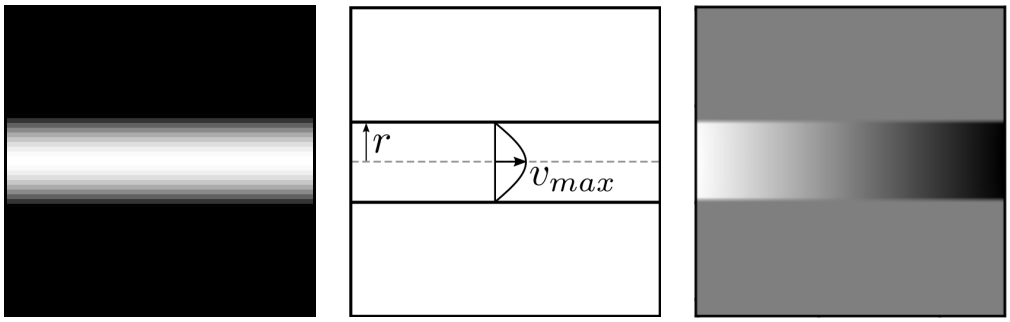
The multifidelity training dataset consists of 32 HF and 116 LF representations, each of which contains noisy, subsampled images of dimensions 32×32, 16×16, and 8×8, corresponding to a HF sample from the full 116 HF training set, for a total of $116 \cdot 3 + 32 = 380$ images.

The training loss consists of the integral of the Mean Square Error (MSE), assembled from equal contributions (penalty 1/4) of all four fidelities. The integral here is obtained by multiplying each pixel's contribution to the MSE by its size. We also tested a loss formulation where larger penalties were applied to the high-fidelity samples. While this showed some improvements in the final high-fidelity accuracy, the results were not consistently better than with equal penalties across different network weight initializations.

In addition, we select the best hyperparameters by optimizing for the best fit of the validation set; in addition, we choose the model from the epoch during training with the lowest validation loss. In the low-dimensional case, since a validation set doesn't exist, we optimize for the best fit of the training set. These hyperparameters include the learning rate, step size of the learning rate scheduler, number of filters, regularization penalty, weight initialization scheme, batch size, optimizer, dropblock location and drop probability.

After training, every dataset's accuracy is evaluated on the same validation and test sets described above, i.e., 49 and 35 HF images, respectively. Final accuracy results are reported for predictions on the test set, using the model with the lowest validation loss during training.

For the one-dimensional case, an L_2 regularization penalty is carefully selected to constrain the relationship between the LF and HF and therefore, in general, its value may depend on the dataset. For Eqs. 2-1, we use L_2 regularization with penalty 10^{-4} only on the weights following the LF prediction. For Eqs. 4-3, we compare the cases where we use regularization on all weights with penalty 10^{-5} versus the case with no regularization. For the high-dimensional case, no regularization was used.



(a) (b) (c)

Figure 5: Poiseuille flow test case. Velocity profile (a). Test case parameterization in terms of the fluid region radius and maximum velocity (b). Pressure result (c).

4 Estimating prediction uncertainty with dropblocks

Dropout layers [23] are a widely used form of network regularization, designed to avoid overfitting during training. These layers operate by dropping neurons at random, simulating an ensemble of architectures without the extra computational burden of testing them individually. However, dropout layers are ineffective for convolutional architectures, due to the spatial correlation present in images, so that features typically consist of multiple correlated pixels. For this reason, DropBlock [6] layers are designed to drop a continuous group of pixels. As these layers are still parameterized in terms of drop probability p, the relation between such hyperparameter and the actual ratio of features being dropped should be first clarified. A Bernoulli mask is generated in [6], using a probability γ expressed as

$$\gamma = \frac{pF}{b(F-b+1)} \text{ for 1D, and } \gamma = \frac{pF^2}{b^2(F-b+1)^2} \text{ for 2D.} \tag{7}$$

The drop probability p may be an inaccurate representation of the percentage of elements dropped. As an example, for $p = 1$, γ may be less than 1 whenever $F \neq b$, meaning that, on average, not all features will be dropped despite the deceptively high drop probability. As seen in Table 1, even when γ is equal to the dropout probability, the actual drop ratio may not be equal to the dropout probability, since the Bernoulli mask generated with γ is expanded by the block size. In [6], a distinction is made between the dropblock mask being independent or shared across feature channels; we considered both approaches and chose the one producing the best accuracy for each test case; we also compared the implications of both choices in §5.4.

When used during the evaluation of an optimally trained network, dropblocks layers can also be used to *inject stochasticity* in the network predictions, and, combined with Monte Carlo sampling, provide a tool to quantify output uncertainty. This technique is commonly referred to as *MC dropout*, see [5]. As noted in §1, it is important to emphasize that this notion of uncertainty reflects the variability introduced in the network by hyper-parameters (in this case changes in the network architecture induced by randomly dropping groups of pixels within features) and not the impact of any uncertainty either in the network weights (as in Bayesian neural networks, see [13, 15]) or its inputs.

For the one-dimensional regression network discussed in §3.1 we added dropblock regularization with a drop probability of 0.2 and a block size of 3. The drop probability is increased by equal increments for 5000 steps, until it reaches 0.2, or, in other words, through a linear scheduler, recommended in [6] to achieve more accurate predictions. The dropblock mask is independent across feature channels due to the small dimensionality of the layers. Each convolutional layer is followed by a tanh activation, with the exception of the final layer where we use a linear activation. Finally, for the two other networks discussed in §3.2 and §3.3, we have investigated dropblock layers that are shared or independent across feature channels, and settled on shared masks, using a block size of 3, with a linear scheduler for 300 steps from a drop probability of 0 to 0.25.

Note that predictions in §5.1, §5.2, and §5.3 utilize a single dropblock layer (or 2 layers in §5.1) during training, and final predictions were calculated in the absence of dropblock layers. This setup generally produced more accurate final predictions with dropblock off than multiple dropblock layers during training. However, as we discuss in §5.4, this setup is not conducive for uncertainty quantification, and we're able to produce equally accurate results for the *mean prediction*, if we utilize multiple dropblock layers during training, with different dropblock hyperparameters.

Block size	p	Feature size	γ	Actual dropped	Block size	p	Feature size	γ	Actual dropped
3	0.2	3	0.200	0.451	5	0.2	8	0.080	0.294
3	0.9	3	0.900	0.997	5	0.9	8	0.360	0.840
3	0.2	4	0.133	0.318	5	0.2	16	0.053	0.208
3	0.9	4	0.600	0.912	5	0.9	16	0.240	0.678
3	0.2	8	0.088	0.227	7	0.2	8	0.114	0.481
3	0.9	8	0.400	0.747	7	0.9	8	0.514	0.974
3	0.2	16	0.076	0.197	7	0.2	16	0.046	0.227
3	0.9	16	0.342	0.680	7	0.9	16	0.206	0.713

Table 1: Dropout probability vs. the average ratio of features dropped. This latter is computed across 1000 dropblock realizations and averaged. This is evaluated for a one-dimensional layer, where each channel has a length of 8, and masks are independent across feature channels.

5 Results

5.1 One-dimensional regression

To ensure there was no detriment to using a convolutional network as opposed to a fully-connected network, we started training the network from LF data only. After obtaining accurate results on the LF, we focused on the multifidelity predictions. The explicit network outperformed the implicit network, which seems reasonable given the significant correlation between the LF and HF models..

The multifidelity network is able to correctly leverage the LF data to influence the HF prediction to more closely resemble the true HF function, as shown in Fig. 6(b), compared to the predictions from the network trained with HF data only in Fig. 6(a). Similarly, in Fig. 7, the multifidelity network is able to capture the discontinuity by extracting this information from the LF data, since this feature could not be learned from the limited HF data. Equally accurate predictions in Fig. 6(a) and Fig. 7(a) result from different initial choices for the weights and biases, drawn from $U(-s, s)$, with $s = nk_0k_1$, n is the number of input channels, and $k = (k_0, k_1)$ is the kernel shape. These results were found consistent across other weight initialization schemes, such as Xavier [7].

Including x as an additional input downstream of the LF predictor was a necessary adjustment needed to separate the LF into a linear and a non linear contribution, facilitating their combination into an optimal HF predictor; in this regard, note that in Eq. (4), $y_H(x) = cy_L(x) + F_l(x)$, where c is a constant and $F_l(x) = -20x + 20$ is linear in x.

Although similar, the two problem sets for one-dimensional regression differ in one important aspect, i.e., the x coordinate for the HF samples is shared across fidelities for Eq. (1) and (2), whereas these locations are different for Eqs. (3) and (4). For different LF and HF x values, spikes appear in the multifidelity network predictions, as shown in Fig. 7(c). The LF prediction overcompensated by creating spikes at the locations of the HF data, to improve the HF training loss, without altering the training loss at the LF points. Therefore, the LF portion of the network required more regularization to prevent this behavior.

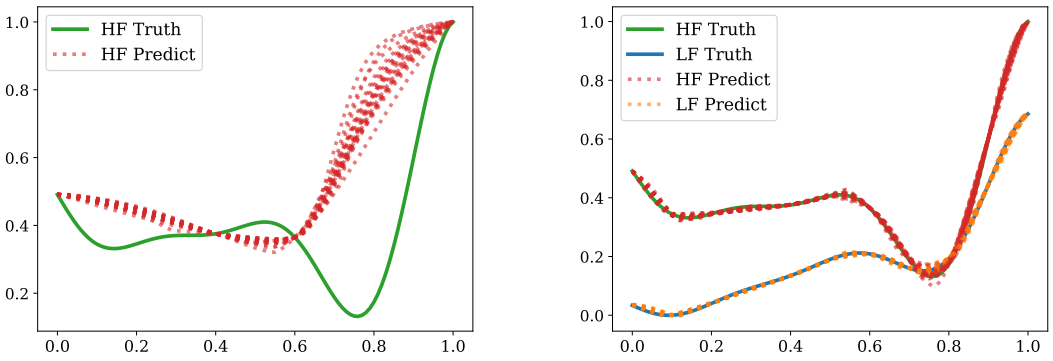
This does not happen instead when using the same x values for the LF and HF, since a spike reducing the HF loss would necessarily increase the LF loss. Finally, although [15] reported robust results, their network required the regularization penalty and the network size to be carefully selected to capture the true underlying HF-LF correlation; since no validation set was included, this would require some degree of manual tuning, which might not be possible in a realistic application, since HF data may not be readily available. Therefore, the sensitivity of our network to the regularization penalty for the example in Fig. 1(b) does not appear to be a limitation of this specific architecture.

5.2 High-dimensional dense regression

We trained the multifidelity networks with implicit and explicit feedback using 32 HF and 116 LF pressure examples (this network configuration is denoted as *MF 32/116*), and compared its outputs with those from the HF 116/0 and HF 32/0 networks, respectively. Results are shown in Table 2 where we report the final validation accuracy as R^2

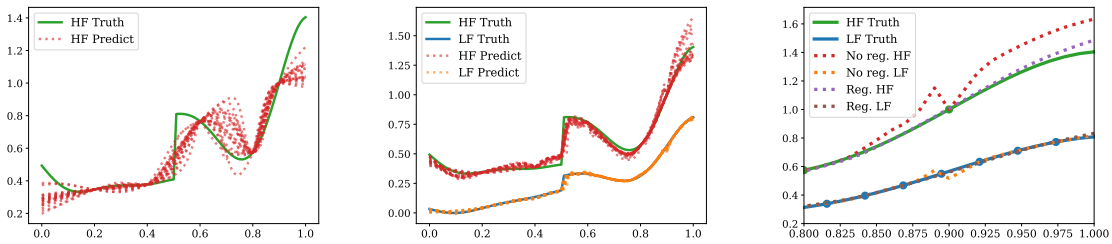
$$R^2 = 1 - RMSE = 1 - \frac{N-1}{N} \cdot \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{8}$$

where $RMSE$ is the relative mean squared error and N is calculated across the entire validation set (i.e., the number of pixels in a single image times the number of available validation samples). We also report a normalized accuracy



(a) (b)

Figure 6: Predictions from (a) HF only network, (b) MF network, for different random initializations. True function values are from Eqs. (1), (2).



(a) (b) (c)

Figure 7: Predictions from (a) HF only network (b) MF network trained with LF and HF data, for different random initializations. (c) Effect of regularization on LF predictor.

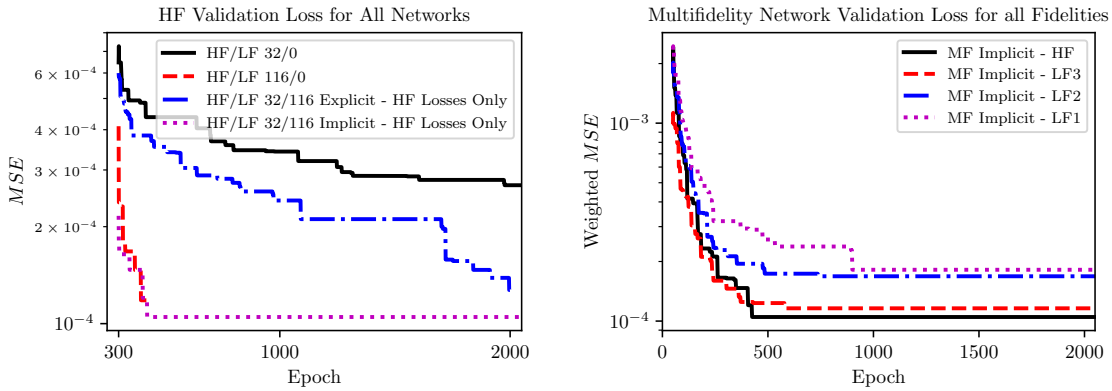
with respect to the cost of generating the training data set, quantified, in this study, as the total number of pixels in the data. Thus, 116 HF images with resolution 64×64 have a cost of $475,136$ pixels, while 32 HF images have a cost of $131,072$ pixels (a cost ratio of 0.276 to the 116 HF case). The multifidelity dataset with 32 HF images and 116 images for each LF would instead result in a cost of $286,976$ pixels (a cost ratio of 0.604 to the 116 high-fidelity case).

The multifidelity network with implicit feedback significantly improves the accuracy with respect to the HF 32/0 network, and its normalized accuracy is also superior to the HF 116/0 network. High-fidelity and multifidelity validation loss profiles are shown in Fig. 8. Fig. 8(a) compares the HF contribution to the validation loss for four networks, i.e., HF 116/0, HF 32/0 and MF 32/116 with both implicit and explicit multifidelity coupling. The plot emphasizes the acceleration in convergence produced by training networks with multifidelity data. Figure 8(b), on the other hand, demonstrates the ability of the network to learn multiple LF representations during training.

The approximation accuracy of the HF 32/0 and MF 32/116 networks is compared for two pressure examples from the test set in Fig. 9 and for slices of the pressure field parallel to the cylinder generator in Fig. 10. The multifidelity network shows more consistently accurate predictions across the entire image instead of a localized region. The larger pressure errors noticeable near the fluid boundary appear typical of convolutional neural networks, which often report lower accuracy near the boundary (see, e.g., [9]). This effect is magnified in our case as the boundary changes with every sample. The original U-Net architecture overcomes this through reflective padding the input layer and not padding any subsequent layers [21] (whereas we zero pad each convolutional layer), although other approaches have been proposed in the convolutional network literature to overcome this limitation (see, e.g., [9]).

Skip Conn.	Network Type	HF/LF	R^2	Normalized R^2
Concat	MF, explicit feedback	32/116	0.924898	3.223e-06
Add	MF, explicit feedback	32/116	0.930846	3.244e-06
Concat	MF, implicit feedback	32/116	0.917597	3.197e-06
Add	MF, implicit feedback	32/116	0.940792	3.278e-06
Concat	HF	32/0	0.909241	6.937e-06
Add	HF	32/0	0.874646	6.673e-06
Concat	HF	116/0	0.94422	1.987e-06
Add	HF	116/0	0.935619	1.969e-06

Table 2: Comparison of HF and MF network regression performance. Accuracy is computed for the test set. The cost for multifidelity training is 286,976 pixels (32 high fidelity images, 116 samples of each of the 3 coarse low fidelity sets). The cost for the 116 HF samples is 116×64×64=475,136. The normalized accuracy is R^2/C where C is the cost. The terms *Concat* and *Add* refer to how the information from a skip connection is assembled into the decoder.



(a) (b)

Figure 8: Validation loss profiles for Poiseuille test case (a). The plot compares the losses resulting from HF only and MF training, using the best model for each category, i.e. with the highest test accuracy. The profiles for the weighted mean squared losses integrated over the fluid domain are shown in (b) for the best performing MF approach. Only the decreasing losses are shown.

5.3 Low- to High-dimensional dense regression

The results from the low- to high-dimensional encoder architecture reported in Table 3 show competitive test set accuracy for the MF 32/116 network with respect to the HF 116/0 network. Although the MF network has lower accuracy, its normalized accuracy is larger than the HF network due to the significant smaller cost. The predictions shown in Fig. 11 are mostly accurate, with only localized errors present in cylinders of larger radius as shown in Fig. 11(c) and Fig. 11(d) which may be due to significant upsampling of the input space. Inclusion of gradient information in the loss could potentially be used to smooth the surrogate representation in these regions.

Skip Conn.	Network Type	HF/LF	R^2	Normalized R^2
Concat	MF, implicit feedback	32/116	0.935599	3.260e-6
Concat	High-fidelity only	116/0	0.951893	2.003e-6

Table 3: Comparison of HF and MF network performance in low- to high-dimensional regression.

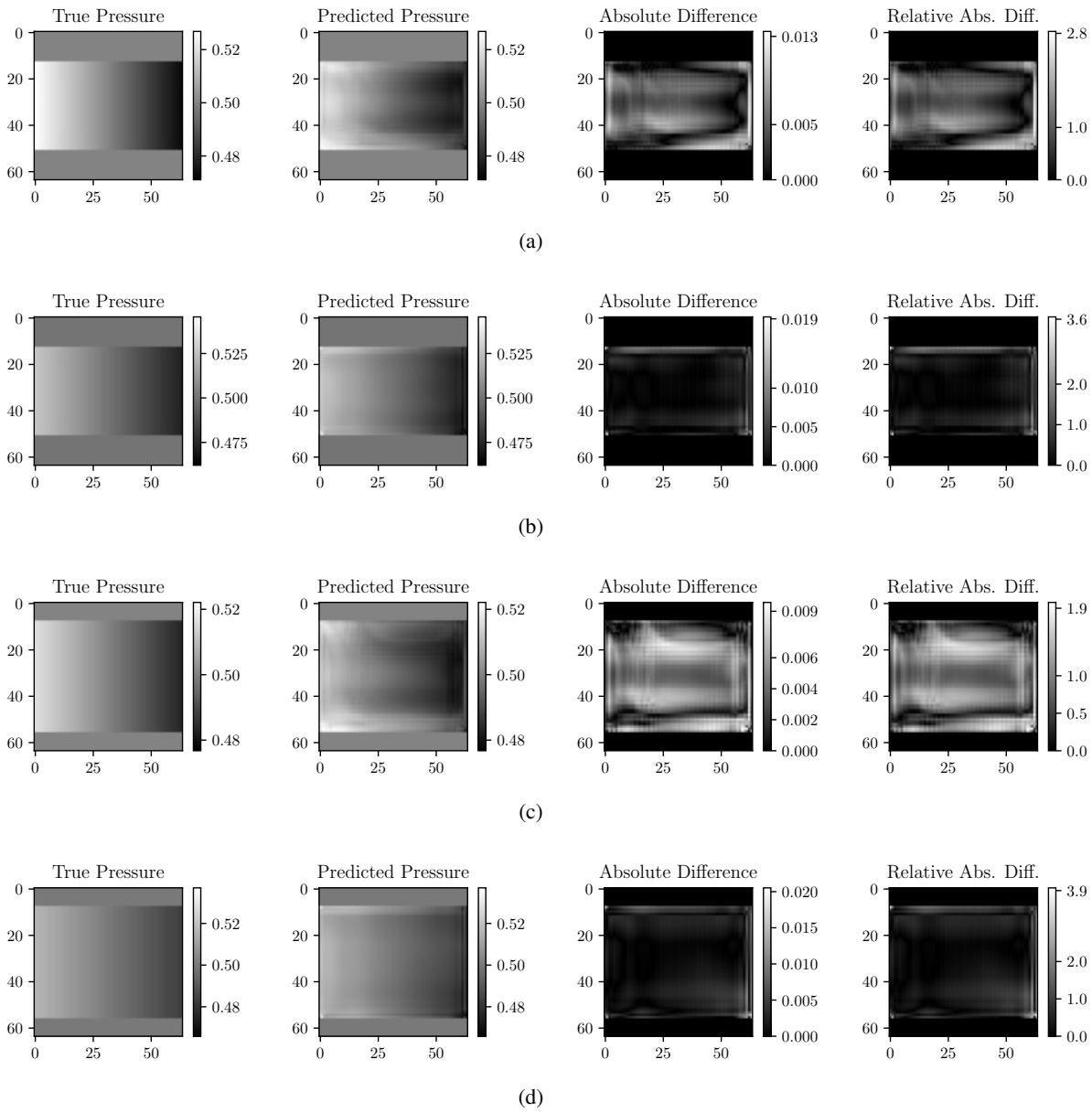
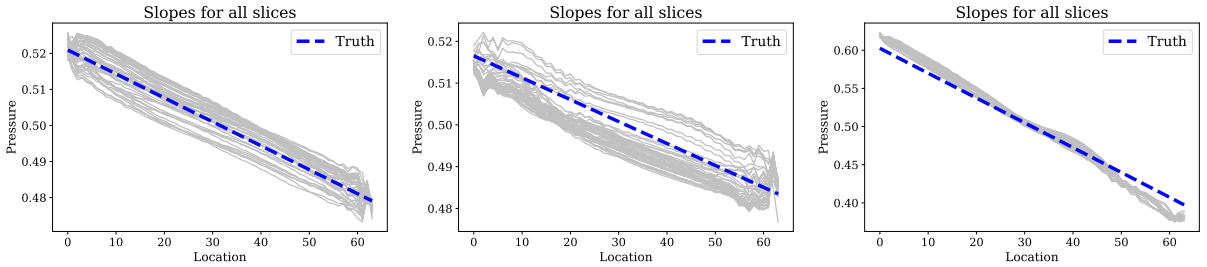


Figure 9: Predictions from HF 32/0 on two pressure configurations from the test set (a,c). Predictions from MF 32/116 on the same test examples (b,d).

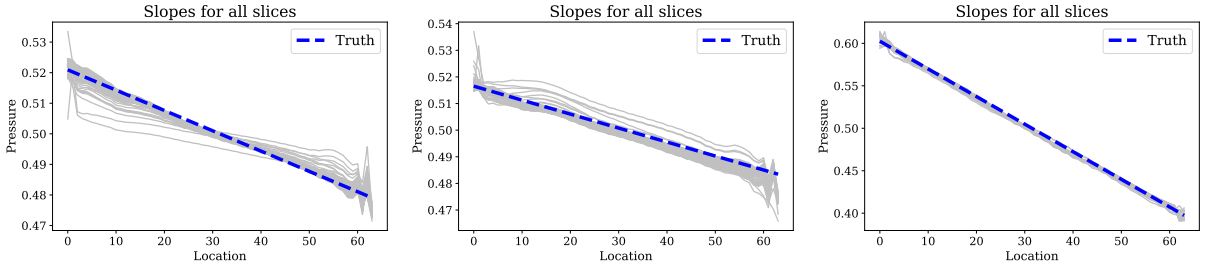
5.4 Uncertainty quantification of network predictions

In the previous sections we investigated the network configurations and parameters leading to optimal accuracy. In this section, we instead focus on the most appropriate architectural choices to quantify uncertainty in the network predictions using MC dropblock. This consists in feeding the same input to the network $N_{UQ} = 1000$ times, generating an ensemble of N_{UQ} predictions induced by randomness in the dropblock layers.

Our goals is to quantify the uncertainty associated with the largest possible ensemble of sub-networks generated by dropblock layers. Therefore we insert a dropblock layer after every convolution layer before the LF outputs, as this configuration is expected to generate more variability in the predictions. However, we would like to maintain a shared parameterization for all HF and LF outputs. To do so, we choose not to include dropblock layers at locations which would induce stochasticity only to a subset of the HF or LF predictors, i.e. we omit dropblock after any convolution



(a) Sample 1 (b) Sample 2 (c) Sample 3



(d) Sample 1 (e) Sample 2 (f) Sample 3

Figure 10: Pressure predictions of test data on multiple slices parallel to the cylinder generator in the fluid region. Note that more slices are available for a samples associated with a cylinder of larger radius r. (a)-(c) Predictions from HF 32/0 network with highest test accuracy. (d)-(f) Predictions from MF 32/116 network with highest test accuracy.

layers following the LF network outputs. Due to this constraint, we apply the same binary mask both prior to the skip connection and prior to the pooling layer at each stage of the encoder of the network in Fig. 3.

We also wanted our network to promote accuracy in *each* MC-dropblock samples rather than only on their mean. To do so, we activated dropblock layers both in training and validation. Keeping all dropblock layers off (e.g. *evaluation* mode in PyTorch) is often performed in validation for regularization purposes. However, this leads to accurate *mean* predictions, but nothing stops the single dropout sample to be inaccurate or have large oscillations, leading to a significant increase of the prediction uncertainty.

5.4.1 One-dimensional regression

Our network for one-dimensional regression includes dropblocks after the first eight convolution layers (see Fig. 2), since these layers precede the LF prediction. For the dropblock layers, we use a drop probability of 0.1 and a block size of 1, due to the low dimensionality of the layers, ultimately amounting to a dropout layer. Therefore, we consider different schedulers which result in more accurate predictions for dropout layers, such as an exponential decay as described in [18] and a delayed start to a linear scheduler. For the results in Figs. 12 and 14, we use the exponential scheduler $p\exp(-gt)+(1-p)$ with $g=3\times10^{-5}$, where t is the current step in the scheduler updates. Alternatively, for the result in Fig. 13, we further optimize the learning rate, learning rate scheduler, and batch size to achieve a better fit of the LF training data, and use a linear scheduler with steps equal to the number of epochs (i.e., 5×10^4).

The 5%-95% confidence interval computed by MC dropblock for the function in Eq. (2) is shown in Fig. 12 and Eq. (4) in Fig. 14. Use of tanh activation functions produce relatively narrow uncertainty intervals both within and beyond the training locations, as shown in Figs. 12(a) and 14(a). By replacing the activations with ReLU, the intervals widen, particularly away from the training examples, as shown in Figs. 12(b) and 14(b). This behavior is consistent with the observations in [5] for dropout layers in fully connected architectures, where they observed bounded and unbounded intervals with tanh and ReLU activation functions, respectively. In general, the ReLU activation functions produce wider uncertainty intervals, which are more likely to capture the true HF model.

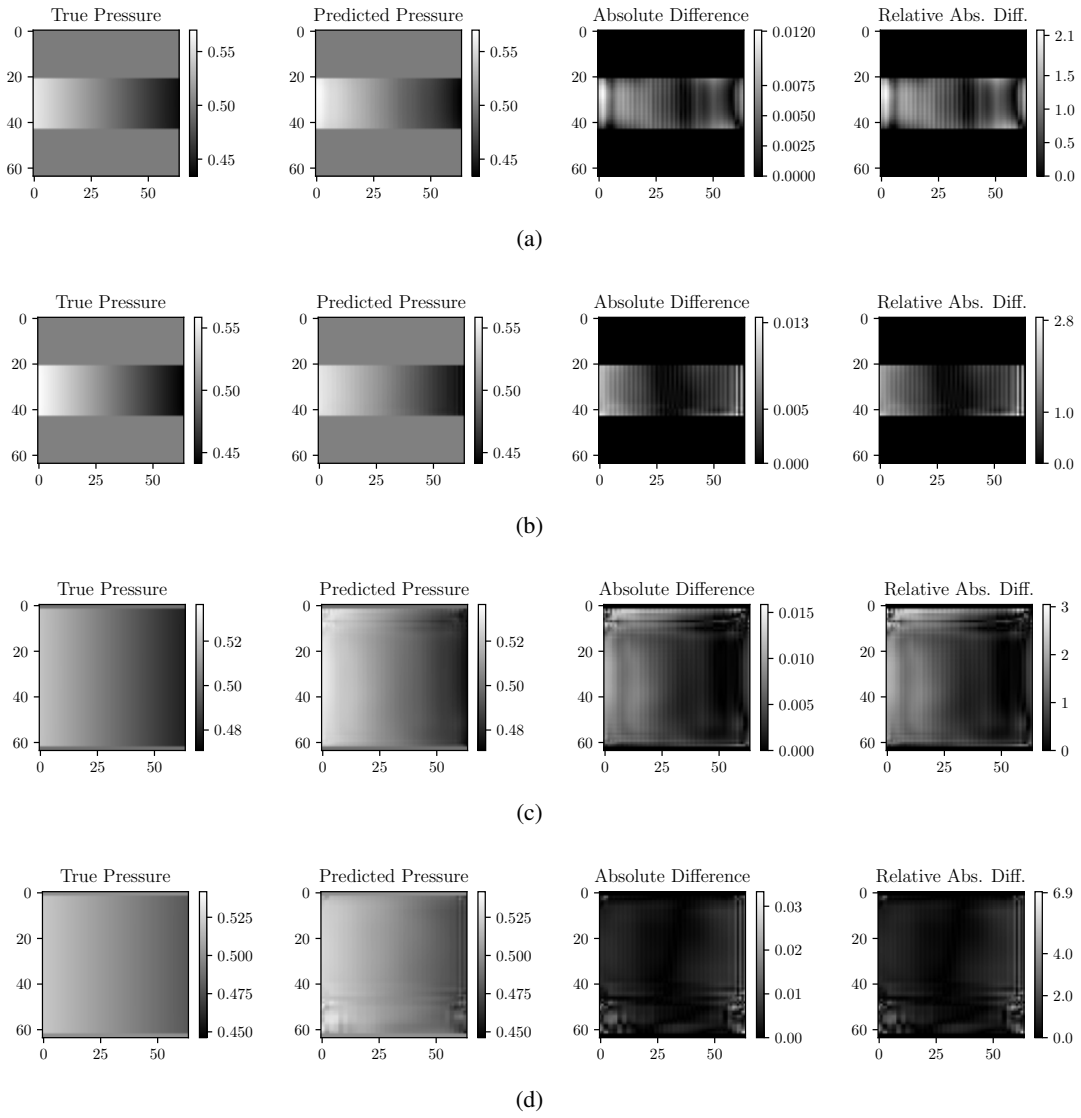


Figure 11: Predictions from the decoder network for the low- to high-dimensional regression test case. (a,c) Prediction from HF 116/0 network. (b,d) Prediction from MF 32/116 network.

In addition, we investigated the effect of the scheduler; by choosing a different drop probability scheduler, or varying the speed of the dropblock scheduler, the uncertainty interval varies, as shown in Fig. 13. Generally, the slower the scheduler, the wider the confidence interval, although this will need to be investigated further.

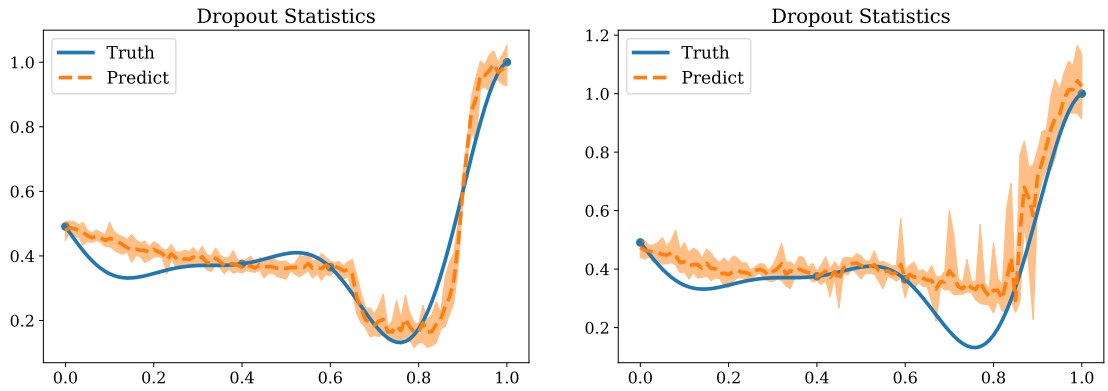
5.4.2 High-dimensional dense regression

For high-dimensional regression, we apply dropblock after the first ten convolution layers of Fig. 3 preceding the LF1 output. We use a drop probability of 0.1, a block size of 3, and a linear scheduler going from $p = 0$ to $p = 0.1$ in 300 steps. This setup produces similar accuracy results for the mean prediction to those in §5.2, obtained with dropblock off. The mean MF prediction has an accuracy equal to approximately 0.94, compared to 0.9 for the HF 32/0 network. In addition, the uncertainty estimates for the high-dimensional case are plotted in Fig. 15 for the centerline of the fluid region, where we use $N_{UQ} = 1000$ realizations. In Fig. 16 we plot the mean square error (MSE) and standard deviation of the MC-dropblock realizations across test samples versus their location along the axis of the cylindrical fluid domain. MF network tends to produce lower variance than HF networks. The HF 32/0 produces the highest variance and highest error, which is consistent with the limited amount of data during training. In addition, both the variance and MSE appear parabolic. This is consistent with the pressure results being approximately 0.5

for all samples at the center of the fluid domain, due to the way samples are normalized. The increase in error and uncertainty away from the center is also expected due to the change in slope of the true pressure profiles.

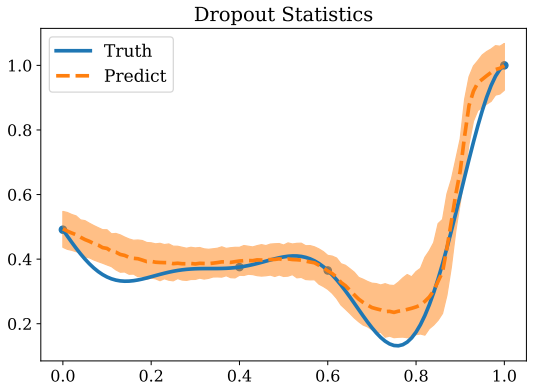
We also compare the results from dropblock masks being shared or independent across feature channels. The study in [6] concludes that independent dropblock masks work better, but it only analyzes the resulting network accuracy and not prediction uncertainty. Our results in Fig. 17 show comparable accuracy for shared and independent masks. However, shared masks resulted in more uncertainty for HF 116/0 and MF 32/116 networks, as expected, and comparable uncertainty for HF 32/0. In addition Fig. 16 shows that dropblock masks applied independently on feature channels lead to more error and uncertainty for networks trained with fewer data (i.e., HF 32/0), as expected.

We finally investigated how specific dropblock placements affect prediction accuracy and uncertainty, even though placing dropblock after every convolution layer provides the most detailed representation of uncertainty due to the network architecture. Dropblock layers placed further downstream in the network (after the 4th or 6th convolution) tends to produce an adverse effect on the multifidelity prediction which is not as apparent in the high-fidelity networks, reducing the accuracy and increasing the variance. This could indicate an enforced reliance on earlier skip connections, which would worsen the LF predictions.



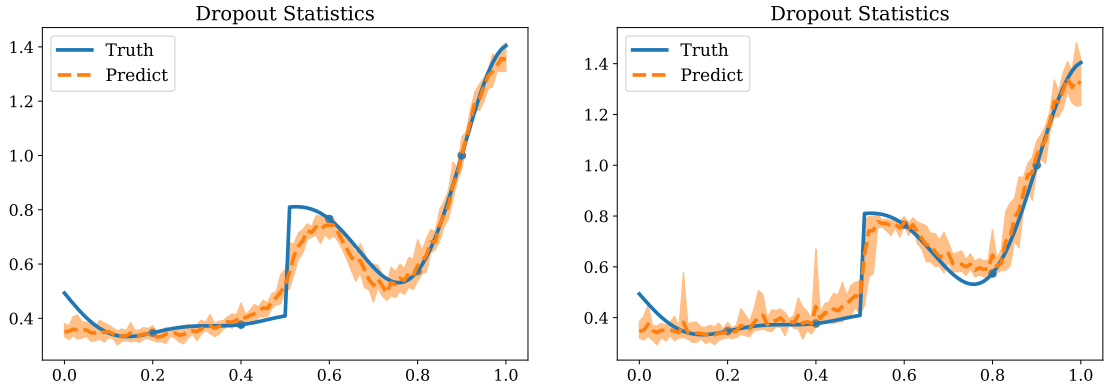
(a) (b)

Figure 12: Exponential dropblock scheduler. Mean prediction and estimated 5%-95% confidence interval for Eq. (2), from a network with multiple dropblock realizations and (a) tanh and (b) ReLU activation functions.



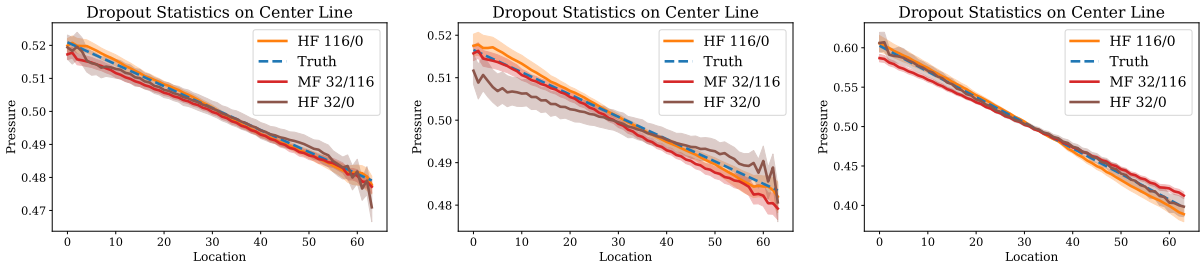
(a)

Figure 13: Linear dropblock scheduler, learning rate 3×10^{-3}, learning rate scheduler with step size 1200, and batch size including all LF and HF samples. Mean prediction and estimated 5%-95% confidence interval for Eq. (2), from a network with multiple dropblock realizations for the HF predictor.



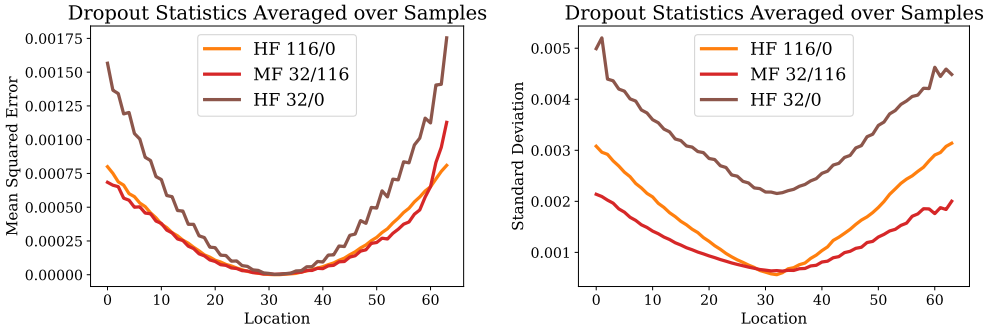
(a) (b)

Figure 14: Exponential dropblock scheduler. Mean prediction and estimated 5%-95% confidence interval for Eq. (4), from a network with multiple dropblock realizations and (a) tanh and (b) ReLU activation functions.



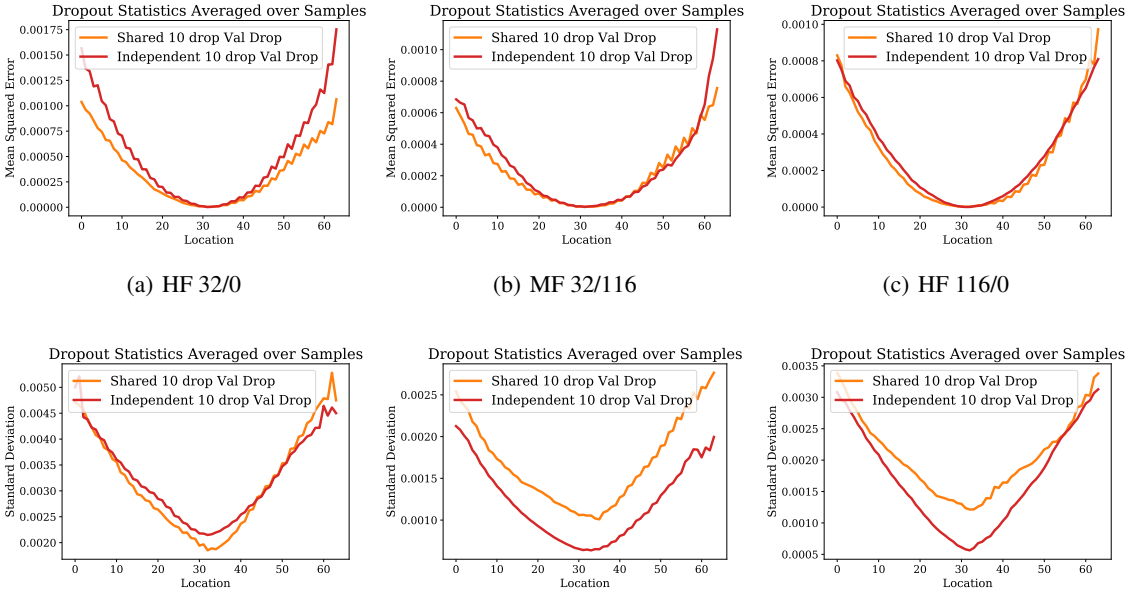
(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 15: Mean prediction and 5%-95% confidence interval from an ensemble of 1000 dropblock pressure realizations for three test data examples sliced along the cylinder centerline.



(a) Drop 10 layers (b) Drop 10 layers

Figure 16: Mean square error and standard deviation resulting from 1000 network evaluations with 10 dropblock layers in each network. Quantities are calculated along the axis of the cylindrical fluid domain. The optimal network is selected using the minimum validation loss calculated by activating all dropblock layers.



(a) HF 32/0 (b) MF 32/116 (c) HF 116/0

(d) HF 32/0 (e) MF 32/116 (f) HF 116/0

Figure 17: Mean squared errors (top row) and standard deviations (bottom row) from 1000 dropblock realizations along the axis of the cylindrical fluid domain for networks containing 10 dropblock layers, The optimal network is selected using the minimum validation loss calculated by activating all dropblock layers.

6 Conclusions and future work

We show that multifidelity networks trained from a few expensive HF samples and a large collection of inexpensive LF examples can be used to improve the accuracy in dense regression. In this work, we focus on convolutional neural networks, since they require a significantly smaller number of parameters with respect to fully connected layers having the same number of neurons. Additionally, architectures resulting from an assembly of encoders and decoders have the flexibility to predict the results from HF physics-based solvers having either high-dimensional inputs, high-dimensional outputs or both, where the number of parameters resulting from fully connected networks would simply be too large. Convolutional networks offer significant computational savings for high-dimensional inputs/outputs, and their performance is comparable to fully-connected multifidelity networks, even for one-dimensional multifidelity function approximation.

Using two examples in one-dimensional functional approximation and the solution of the pressure Poisson equation, we show that multifidelity networks produce, at a reduced cost, a validation accuracy comparable to that of networks trained from a much larger number of high-fidelity realizations. Use of datasets containing examples from multiple fidelities also accelerates training, leading to significant loss reductions early on during the training process.

We also focus on quantifying the variability in the network predictions using dropblock layers where the drop probability is selected by a linear scheduler. Keeping the dropblock active during testing and validation improves the accuracy in single MC-dropblock realizations improving the robustness of the uncertainty estimates. In addition, use of dropblock masks that are independently applied rather than shared across feature channels leads to reduced uncertainty estimates. Finally, we investigate how the location of the dropblock layer affect prediction uncertainty. Adding multiple dropblocks after each convolutional layer, while still providing a shared parameterization across all fidelities, appear to maximize the uncertainty due to variability in the network architecture. Localized dropblock layers may negatively affect the accuracy of some LF predictors, leading to less accurate and more uncertain MF outputs.

Future work will be devoted to investigate the performance of additional network layouts, testing new encoder-decoder configurations and on comparing the variance reduction of MC-dropblock with more traditional MF Monte Carlo estimators.

Acknowledgments

This work was supported by a NSF CAREER award #1942662, a NSF CDS&E award #2104831 (PI DES) and used computational resources provided through the Center for Research Computing at the University of Notre Dame. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government. The authors would like to thank Ishani Aniruddha Karmarkar for her assistance with testing fully connected multi-fidelity network implementations proposed in the literature.

References

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U.R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *arXiv preprint arXiv:2011.06225*, 2020.

[2] P. Baldi and P.J. Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26:2814–2822, 2013.

[3] S. De, J. Britton, M. Reynolds, R. Skinner, K. Jansen, and A. Doostan. On transfer learning of neural networks using bi-fidelity data for uncertainty propagation. *International Journal for Uncertainty Quantification*, 10(6), 2020.

[4] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf. Fidelity-weighted learning. *arXiv preprint arXiv:1711.02799*, 2017.

[5] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[6] G. Ghiasi, T-Y. Lin, and Q.V. Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018.

[7] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[8] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[9] Carlo Innamorati, Tobias Ritschel, Tim Weyrich, and Niloy J. Mitra. Learning on the edge: Explicit boundary handling in cnns. *CoRR*, abs/1805.03106, 2018.

[10] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

[11] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[13] H. Langseth and L. Portinale. Bayesian networks in reliability. *Reliability Engineering & System Safety*, 92(1):92–108, 2007.

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[15] X. Meng, H. Babaee, and G.E. Karniadakis. Multi-fidelity bayesian neural networks: Algorithms and applications. *Journal of Computational Physics*, 438:110361, 2021.

[16] X. Meng and G.E. Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[17] S. Minaee, Y.Y. Boykov, F. Porikli, A.J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[18] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. Curriculum dropout. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3564–3572, 2017.

[19] Lauren Partin, G. Geraci, Ahmad Rushdi, M.S. Eldred, and Daniele Schiavazzi. Multifidelity data fusion in convolutional encoder/decoder assembly networks for computational fluid dynamics applications. In *CSRI Summer Proceedings 2021 (in press)*, pages 102–119. Sandia National Laboratories, 2021.

[20] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[21] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[22] D.E. Schiavazzi, A. Nemes, S. Schmitter, and F. Coletti. The effect of velocity filtering in pressure estimation. *Experiments in Fluids*, 58(5):50, 2017.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[24] Yous van Halder, Benjamin Sanderse, and Barry Koren. Multi-level neural networks for pdes with uncertain parameters. *arXiv preprint arXiv:2004.13128*, 2020.

[25] D. Xiu and G.E. Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.