UNIVERSITY OF BATH

Department of Mechanical Engineering

FACULTY OF ENGINEERING AND DESIGN

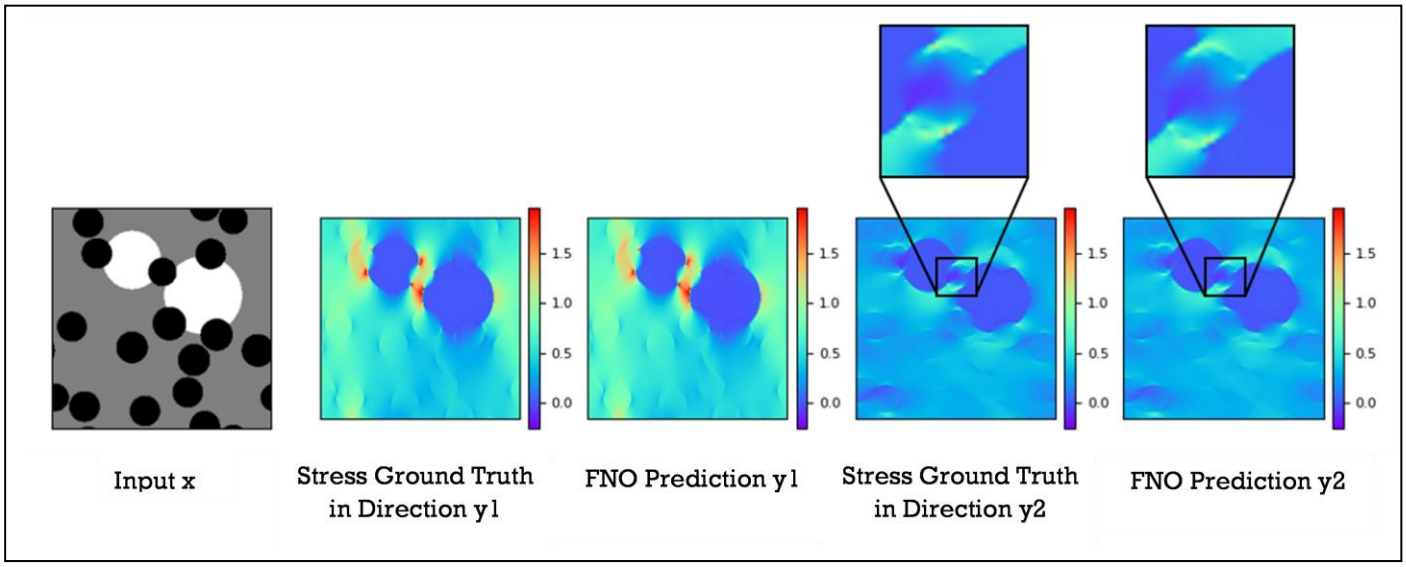
FINAL YEAR BEng PROJECT REPORT (ME30227)

Can Machines Learn to Predict the Behaviour of Carbon Fibre Composites?

Elliot Bancroft

Date of submission: 09/05/2023

Word count: 8147



"I certify that I have read and understood the entry in the Student Handbook for the Department of Mechanical Engineering on Cheating and Plagiarism and that all material in this assignment is my own work, except where I have indicated with appropriate references."

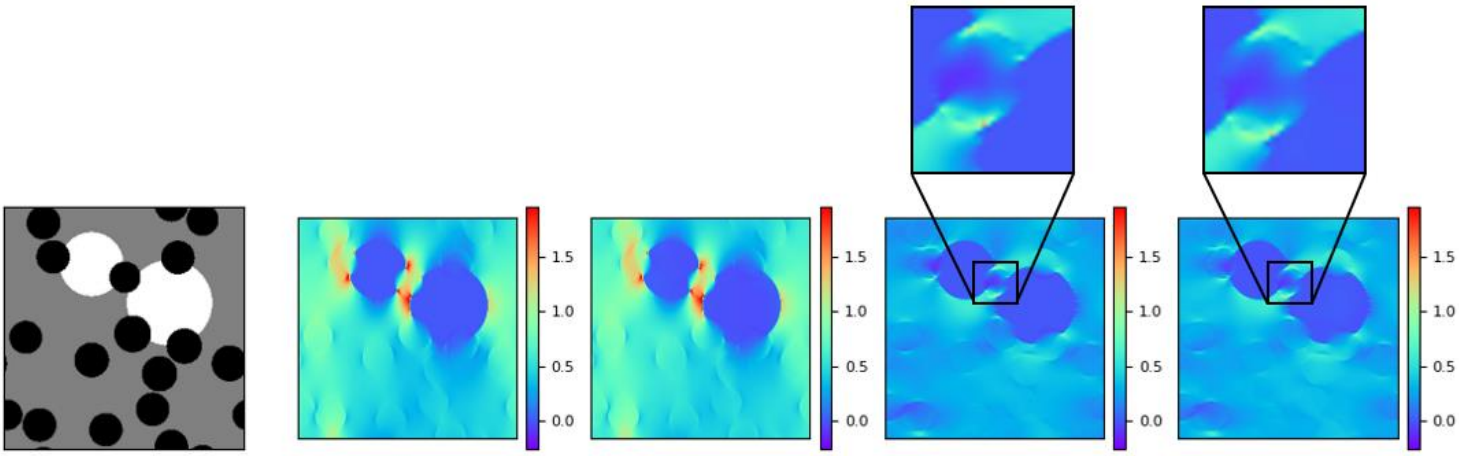
Author's signature: E Bancroft.....

Supervisor: *Prof. Yang Chen*

Assessor: *Prof. Richard Butler*

Department of Mechanical Engineering | UNIVERSITY OF BATH

Can Machines Learn to Predict the Behaviour of Carbon Fibre Composites?



Input x | Stress Ground Truth in Direction y1 | FNO Prediction y1 | Stress Ground Truth in Direction y2 | FNO Prediction y2

Author: Elliot Bancroft

Summary

The aim of this project was to implement a Fourier Neural Operator (FNO) in order to predict the stress, strain and damage field data for loaded cross-sections of carbon fibre composites. The work serves as an extension to the original work performed by Yang Chen et al. "Full-field prediction of stress and fracture patterns in composites using deep learning and self-attention" [1]. The data used in this work has been taken from the data used in the aforementioned report. The Fourier Neural Operator model was interpreted from the NeuralOperator GitHub Repository [2], [3]. The data was manipulated to fit the dimensions required by the FNO, and subsequently used to train the network.

The paper proceeds to demonstrate that the FNO can learn to predict stress and strain field behaviour with high accuracy even for unseen microstructural arrangements with varying characteristics. The model was not able to produce meaningful predictions for damage-field behaviour of the carbon fibre reinforced composite representative elements in the given project time, however, the model was successfully converging, and with more data it is argued that the model could become competent at predicting crack locations and geometries.

Although the research wasn't successfully able to evaluate the models on input images of different mesh sizes, it has been proven that Neural Operators have this capability [3]. In addition, the model was evaluated on lower information data which matched the dimensions of the original training data for the model, which the FNO was able to assess accurately.

Acknowledgements

The author appreciatively acknowledges the contribution of Prof. Yang Chen and Jacobo Ostos Bollman.

Professor Yang Chen, the project supervisor, supplied the data for the project as well as supplementary code for loading RVE data.

Jacobo Ostos Bollman is a fellow University of Bath Engineering student with whom elements of the research were discussed, and training of the Neural Operator models for the stress, strain and damage field data was split in order to alleviate time pressure.

Table of Contents

Report Layout

Given the context in which this report is being produced, contrary to typical academic paper structure in the field, the background and theory of the related problem will be investigated and explained extensively in order to aid the reader's understanding assuming zero knowledge of the subject area. The aim is to enable the reader to be able to follow the experimental procedures as well as the contribution of this paper in the field of work.

The paper will begin by introducing the applicable academic fields, namely relevant materials science, as well as solving methods for multiple non-linear partial differential equations (PDEs). It will explain the basic function of neural networks, as well as the modified neural networks which have been applied to similar problems to date.

The contribution of this paper to the subject area will be outlined and justified, before the paper progresses to analyse in more detail the previous work by academics in the area upon which this paper will be building.

Thereafter, a more technical section containing experimental and computational methodologies will be presented, having supplied sufficient context beforehand in section 1. This will include technical specifications of the models used to produce the results in this paper, with additional background found in the Appendices for the reader's interest.

Results will include diagrammatic predictions given by the model, as well as quantifications of the model's learning capabilities, and generalization of the model to lower resolution data.

1. Introduction

1.1 Problem Background

Predicting the dynamic behaviour of heterogeneous materials and, in particular, Carbon Fibre Composites is typically a complex problem to solve due to anisotropic behaviour caused by ply-orientation as well as structural irregularities and residual stresses and strains arising as a result of manufacturing processes [4], [7], [8]. This can lead to unreliable results from physical testing, which due to the minimal plastic performance of stiff carbon fibres in many cases, is also expensive [5], [6].

The prediction of carbon fibre-reinforced composite (CFRC) dynamic behaviour involves solving a very large number of non-linear partial differential equations (PDEs). There is a variety of contributors to the onset of non-linear behaviour in heterogeneous materials, such as the varying physical characteristics of

different materials in the microstructure, particularly across material boundaries, as well as in defects such as voids produced by manufacturing or other means [9], [10]. In recent times, numerical methods such as Finite Element Analysis (FEA), the Finite Difference method and the Newton-Raphson method, have been utilised to attempt to solve this problem [16], [17]. With recent advances and improvements to the FEA modelling process, dynamic behavioural characteristics have been successfully predicted at accuracies as high as 1.6% maximum load error compared to the physical tests [8].

The problem with such methods however is the vast amounts of computational power required for each test, as well as the exponential increase in computation time for complex geometries, and problems where the PDE needs to be solved at each timestep, causing computation time to take hours for highly complex systems [11], [12]. In some applications, such as use in the healthcare industry, image processing and autonomous cars, rapid responses are required from simulations, in which cases, Machine Learning (ML) algorithms have been found to be much faster prediction generation methods than purely FEA based approximations with little performance drop-off, if any [13]. Sun et al. found they were able to predict complex stress fields for composites in just a few seconds using adapted machine learning methods whereas it took 92.5 hours to run the same simulation using an FE model [14].

1.2 Neural Networks

In attempt to generate solutions to these types of problems more quickly, without any significant drop-off in accuracy, a ML approach using Neural Networks (NNs) has been applied to complex problems requiring the solution of PDEs [15]. Although basic NNs may be less suitable in terms of speed and accuracy than FEA in scenarios where a single type of PDE is involved in a simple geometry, 1 or 2-dimensional problem [18], the strength of NNs lies in their application to more complex problems - relating to complex geometries, higher dimensional problems and multi-scale modelling [19], [20]. Neural Networks can be trained on example data, which involves passing an input to the network, and allowing the network to compare the true result of the simulation (a series of PDEs) to its prediction of the outcome. The network is composed of a vast number of neurons and connections, where a neuron is simply a number stored at a point in the neural network. Between neurons, there are relationships defined by weights and biases (see equation 1) which are the variables that the neural network is attempting to optimise, by taking input data and comparing the output to the truth.

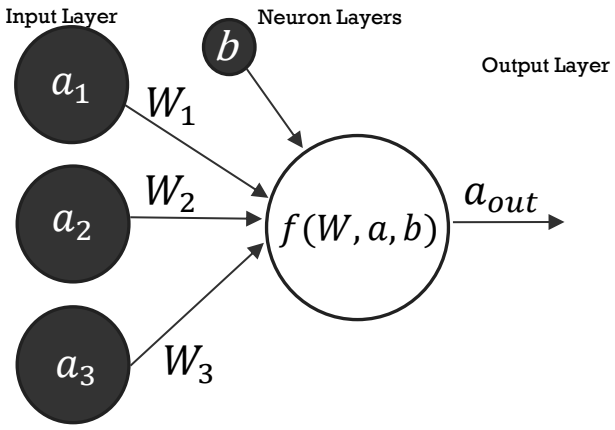


Figure 1: A simple Neural Network diagram demonstrating the function of weights, w, and biases, b, as network parameters in fully connected network layers.

$$f(W,a,b) = \left(\sum_{i=1}^{n} a_i \,.\, W_i\right) \; + \; b \; = \; a_{out} \tag{1}$$

Training can be terminated whenever the user is satisfied with the predictions produced by the neural network. At this point, the values of each of the weights and biases can be saved, and the trained model can be used to evaluate unseen input instances and produce a prediction very quickly.

In particular, Convolutional Neural Networks (CNNs) are popular in spatiotemporal problems with image inputs due to their ability to deal with 3D matrices, as well as their ability to extract features of the image from their grid structure using pooling and convolution layers [21], [22], [20]. These layers are used to extract features and subsamples the image inputs, ready to be passed to fully connected layers as seen in figure 1.

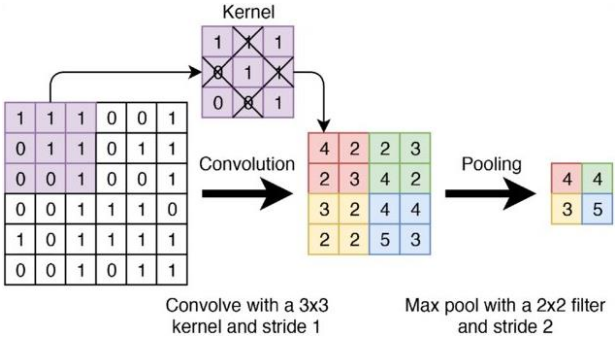


Figure 2: A visualisation from [100] of convolution using a kernel, and the pooling operation.

A CNN is composed of layers of neurons which perform different operations. Instead of applying standard matrix multiplication to data, they perform a convolution in at least one of the layers [23]. A convolution allows an image to be down sampled, since it converts all the pixels in the convolutional kernel field into one value (see figure 2), by discarding irrelevant detail and retaining the important information [24].

A kernel is a matrix of values which is typically smaller in size than the image matrix over which it is passed. It moves over the image at specified intervals, and performs a dot product with the pixel values in its field of view, in order to output one value for each position it passes over [25], [26].

In this way, CNNs allow weights and biases to be related to only a portion of the receptive field as opposed to the entirety of the input data as is the case in standard neural networks. Down sampling to produce these smaller representative images are also known as feature maps, since they contain information from the original input.

The other operation characteristic of CNNs is known as pooling, which extracts a representative value from each feature map, in order to further reduce the size of the data. Most commonly, max pooling is used, whereby the maximum value is extracted from the feature map, although other popular types of pooling include average pooling, stochastic pooling and mixed pooling amongst others. This method has been shown to reduce overfitting of the model, as well as reducing the computational cost associated with the network by reducing the number of weights and biases in the network [27], [28].

In previous work by Yang Chen et al. [1], a finite dimensional operator approach was used whereby the solution operator was parameterized using a convolutional network operating between finite-dimensional Euclidean spaces [29], [30], [31]. This means that the method is mesh dependent, hence the network hyperparameters will need to be tuned in order to produce accurate predictions for input images of different resolutions and sizes [32]. These types of networks are constrained to a specific input domain, hence images with different sizes will not be able to be accurately processed without retraining the network with a different set of image data with the desired characteristics [31], [33]. Another problem with finite dimensional CNNs, and indeed CNNs in general, is the volume of computations required in the convolutional steps, exacerbated by the fact that convolutions are widely accepted to be the most computationally costly element of training a neural network [34], [35], [36], [37].

1.3 Neural Operators

Therefore, an important step to further the work performed by Yang Chen et al. is to create a model which is discretization-invariant. This will enable the network to be used to predict CFRC stress, strain and damage behaviour for a much wider range of image data [38]. In this work, the use of Fourier Neural Operators (FNOs) is proposed. Neural operator networks are discretization invariant since their neural network is comprised of neural operators as layers which are activated by non-linear activation functions, such as the Rectified Linear Unit (ReLU) function (figure 3) [33], [39]. Given that the neural operator layers have non-linear activation functions, even if the layers are linear operators, it can be said that neural networks are universal approximators – 'for any series of inputs and outputs, the neural operator can reasonably approximate the function which maps the former to the latter' [40], [41]. The architecture of the neural operator is displayed in figure 4.

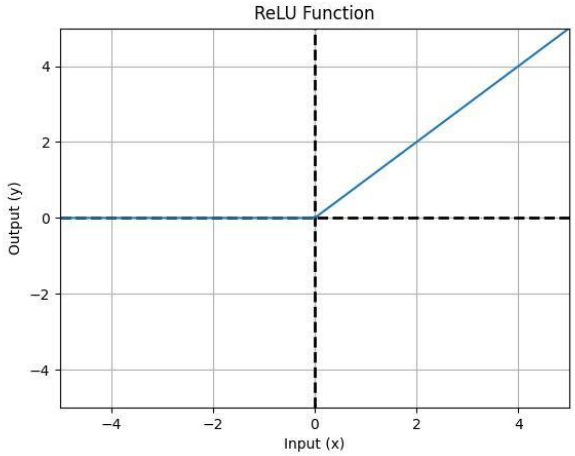


Figure 3: Graphical representation of the non-linear ReLU function

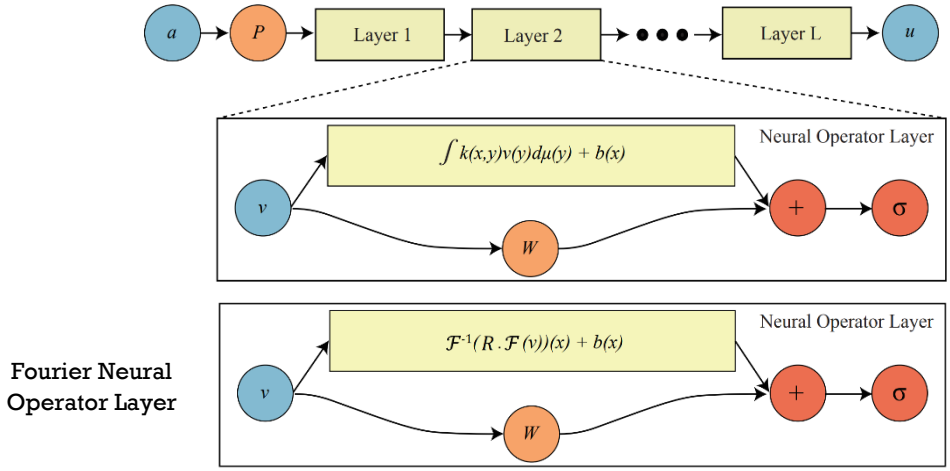


Figure 4: The Neural Operator architecture proposed by Kovachki et al. [3].

One of the major drawbacks of the neural operator is the computational cost of solving integral operators, however it has been shown that the use of fast Fourier transform (FFT) can moderate this problem by solving differentiation and integration operators in the Fourier domain [33]. This is because differentiation and integration of functions in the time domain, are equivalent to the multiplication and division of the Fourier transforms of the functions in the frequency domain respectively [42].

In this work, a Fourier Neural Operator architecture will be trained to predict the stress, strain and fracture behaviour of a carbon fibre composite under load. This will build on previous work since this network will only be required to be trained once before having the ability to generate approximations of performance for other CFRC image data of different sizes and resolutions.

The trained network's ability to produce predictions on lower resolution input data will also be investigated.

2. Literature Review

Literature relating to the application of machine learning to predict carbon fibre composite and, more generally, other material behaviour will be reviewed chronologically in order to assess the contribution of this paper in furthering completed work in the field. Since this paper builds directly upon the paper of Yang Chen et al., the reader is urged to consult paper [1], for further underpinning literature. This literature

review section will give a more in depth overview on some of the work discussed in [1] as well as work published following the publication of paper [1].

2.1 Using deep learning to predict fracture patterns in crystalline solids [43]

Hus et al. successfully used a long-short term memory (LSTM) convolutional network to predict the fracture behaviour of simple crystalline solids. The Conv LSTM was able to accurately predict crack propagation for very simple cases, however some inaccuracies began to occur even when introducing an additional crystalline region of different orientation to the test sample. This highlights that the method was still not competent enough to be able to predict the fracture behaviour of more irregular materials, especially composite materials such as CFRCs which have far more complex microstructures.

2.2 Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods [44]

David Montes de Oca Zapiain et al. successfully implemented an LTSM to predict the behavioural evolution of two-phase mixtures. This work was promising in that it could be directly transferred to microstructure evolution given the similarity of the image data used to train the network. One limitation of this work is that the accuracy of the predictions at mixture boundaries and other localised features was limited. In addition, the LTSM model has not been tested or proven on more data sets with more impulsive characteristics such as crack initiation or propagation. This may be a limitation of the model since the randomness of crack-initiation means the model will have to learn data on irregular temporal fields, where LTSM typically expects and learns on data with consistent spaciotemporal properties [45], [46].

2.3 End-to-end prediction of multi-material stress fields and fracture patterns using cycle-consistent adversarial and transformer neural networks [47]

Buehler and Buehler have successfully been able to predict stress fields surrounding cracks with reasonable agreement. For more complex samples with multiple cracks, the local information contained in the prediction began to lack accuracy.

Buehler and Buehler also successfully managed to implement transfer learning to the network in order to attempt to use the pre-trained network to predict the dynamic crack propagation. This prevented the network from having to be trained from scratch by simply adjusting some of the training samples. The crack propagation prediction was limited however, with the model not having the capability to predict much further in the time domain past the initiation of the crack propagation.

Further to this study, the model was then tested on samples involving both soft and rigid microstructures, which was an additional level of complexity above previous works, and closer to the level of complexity of CFRCs. Transfer learning was again used based on the initial model, and Von Mises stresses within the samples were predicted with an impressive degree of accuracy. The model was even able to reconstruct the original sample mesh using the Von Mises stress field data as an input.

Overall, the GAN model shows impressive versatility to transfer learn other field data, albeit with lower stress prediction accuracy for more complex microstructures and also suffered in performance for more complex crack propagation prediction, for which it was only able to predict a very short expansion of the crack.

GAN models are also prone to mode collapse and can be hard to train as they can take a long time to converge [48], [49].

2.4 Predicting Mechanical Properties from Microstructure Images in Fiber-reinforced Polymers using Convolutional Neural Networks [14]

Sun et al. produced a CNN-based architecture called StressNet [50] which was trained on 32x32 pixel samples taken from an FE simulation. Their work demonstrated success in predicting stress-fields for fibre-reinforced polymers, however beyond this application, its ability to predict other physical characteristics such as damage and strain remains to be tested.

In addition, the network is not able to generalize well, meaning the network will only be able to predict the stress-field behaviour for samples of similar sizes and will assume the same load conditions as the samples it is trained on [51], [52].

2.5 A Data-Driven Approach to Full-Field Damage and Failure Pattern Prediction in Microstructure-Dependent Composites using Deep Learning [97]

Two stacked, CNNs were used sequentially by Sepasdar et al. in order to predict failure patterns for 2D single fibre reinforced polymer representative volume elements (RVEs). The first CNN is trained on post-failure full-field stress distribution data, and the second SNN subsequently is trained to produce a failure pattern using the full-field stress prediction from the first CNN as an input.

The RVEs were generated using a simple stress condition whereby the fibre reinforced polymer is constrained in the x-direction at one edge, with a load applied in the x-direction at the opposite edge (see figure 5).

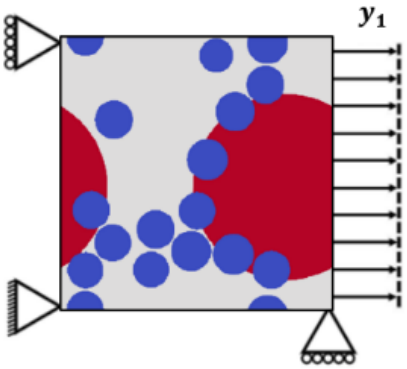


Figure 5: The periodic boundary conditions placed on the RVEs during the data generation process.

The input images were of higher resolution than in much of the previous work - 256x256 pixels in size, however the method was only able to correctly predict the crack pattern with 90% accuracy even for a simple stress condition. More complex, multi-fibre composite microstructures were also out of the scope of this work.

2.6 Predicting mechanically driven full-field quantities of interest with deep learning-based metamodels [53]

Mohammedzadeh and Lejeune used a novel "MultiRes-WNet" CNN-based architecture to predict displacement, strain and crack propagation behaviour on 28x28 pixel input images. The network was much more accurate than previous work at predicting crack propagation paths, and was subsequently shown to be successful at predicting strain and displacement behaviour.

The success of the network to approximate behaviour for much higher resolution images was not investigated in the work. Mohammadzadeh and Lejeune also indicate the need for a model which can achieve the same accuracy based off a far smaller training dataset in order to reduce computation costs.

2.7 Stress field prediction in fiber-reinforced composite materials using a deep learning approach [54]

Badhuri et al. use a U-net arrangement of a CNN, in order to test the performance of a neural network trained on cheaper, low-resolution data, when tested on data of higher resolution. The main problem with this work is that the U-net CNN is not mesh independent, and hence, if the model is to make predictions based on higher resolution data, transfer learning must be performed, and high resolution must be produced and used to train the network further.

This is computationally inefficient compared to Neural Operators, which learn function to function mapping rather than vector to vector mapping, giving them the property of mesh-invariance [55], [56], [57].

2.8 Learning the stress-strain fields in digital composites using Fourier neural operator [58]

Mehran et al. were able to successfully implement a Fourier Neural Operator to predict stress and strain field data for simplified 48x48 pixel composite geometries. The Neural Operator was also successfully tested on inputs with upscaled resolutions of 200x200 pixels with reasonable accuracy.

When confronted with inputs which contain less random assortments of pixels than the training set, the accuracy of the model deteriorated slightly posing the question of how well the model is able to generalise, as well as the lack of variation in the training dataset.

The paper does not venture into the investigation of crack propagation behaviour within the images. In addition, the complexity of the features within the images is low, with the input images containing random arrangements of soft and stiff pixels, giving an unrealistic representation of a composite where material

boundaries are square and perpendicular, excluding any complex overlapping or boundaries between microstructures such as voids, resin and fibres.

2.9 Full-field prediction of stress and fracture patterns in composites using deep learning and self- attention [1]

The work conducted by Yang Chen et al. in this paper is the foundation of the work carried out in this project. Like the work of Sepasdar et al., two sequential CNN models were used in order to predict full-field stress and fracture patterns of composites. The model also incorporated a self-attention layer, which are characteristically good at recognising feature boundaries [59], [60], [61].

RVEs containing pore, matrix and fibre details were used to train the model, with the addition of pores into the images of the composites adding a novel level of complexity compared to similar previous work. The RVEs also varied in fibre volume fraction as well as porosity.

High agreement was found between the CNN predictions and the numerical simulations however, the limitation of the paper is again that the model is not mesh or discretization invariant. This is why in this paper we propose the use of an FNO.

3. Experimental and Computational Methods

3.1 Data

The data used in this project is a section of data used in the work by Yang Chen et al. [1]. It consists of RVEs containing fibre, matrix and pore information. The RVEs were algorithmically generated to have microstructural randomness using rules which produced realistic cross-sectional geometries. For this research, two different groups of RVEs were used with volume fibre ratios of 0.3 and 0.5, each group contained 150 samples with different fibre arrangements, within which each fibre arrangement has 6 different porosity variations, equating to a total of 1800 RVEs. An example of a couple of the RVEs can be seen in figure 6.

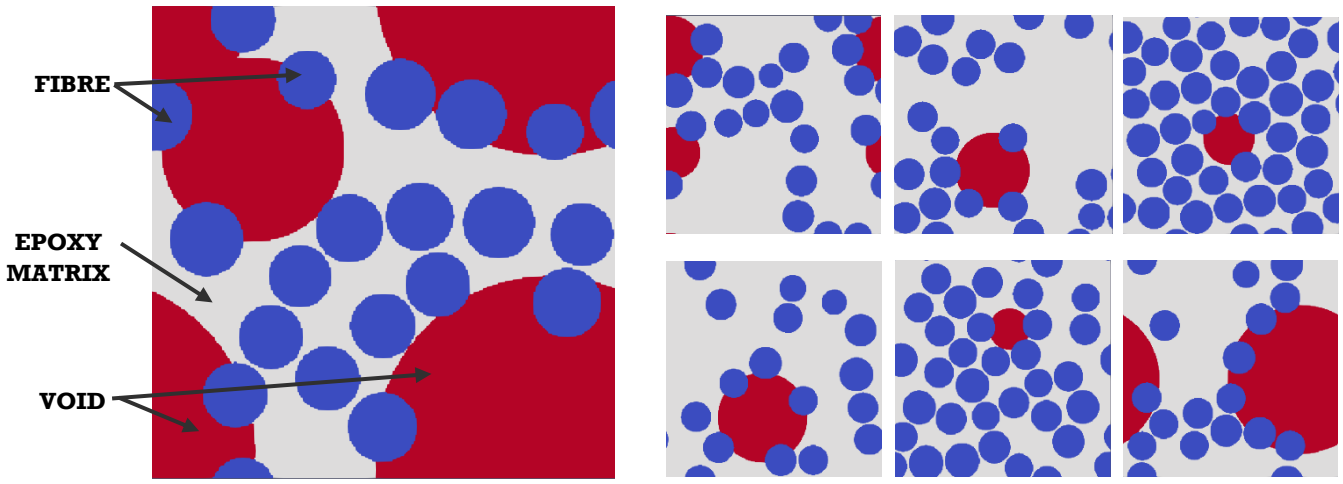


Figure 6: Some examples of the RVEs used in testing with different fibre volume ratios as well as microstructural differences such as void sizes, void locations and fibre arrangements.

In order to test the trained network's capability, it was tested on previously unseen RVEs with volume fibre ratios of 0.25 and 0.6. This will be discussed in more detail in sections 4 and 5.

Periodic boundary conditions [62] are applied to the RVEs and a simulation of transverse tension is applied in increments in the x-direction (denoted as y1 in this paper). A complex, multi-field system defined by PDEs arises as a result of the tension, which was solved using an FFT method [63], [64]. FFT solving methods are a good alternative to FEM methods since they eliminate the need for meshing, and use a simple grid to

break up the image input [65], [66]. This means digital images can be used as a direct input to the solver without the need to apply meshing to the images.

FFT solvers alleviate the computational intensity of traditional FE-based solvers not only by making use of their inherent periodicity to aid homogenisation of the underlying equations, but also by making integral computations in the Fourier domain [67], [68]. This leads to much faster computation times.

Since the data used is the same as that in [1], the physical properties used to model the carbon fibres and epoxy matrix are the same. Nevertheless, for the benefit of the reader, they are repeated below.

The transverse properties of carbon fibres and epoxy matrix are used:

- 'Elastic moduli, $\lambda = 4.167\ GPa$, $\mu = 6.25\ GPa$ for the fibres'
- 'For the epoxy matrix, $\lambda = 4.795\ GPa$, $\mu = 1.353\ GPa$'

'The fracture parameters are $gc = 1\ N/mm$ for fibres and $gc = 0.011\ N/mm$ for matrix. The characteristic length is $lc = 0.5\ \mu m$ for all phases.'

The simulation takes place over a pre-determined set of iterations which ensure that fracture occurs in each of the RVEs. At each iteration, stress, strain and damage field data over the RVE is obtained.

3.2 Model

From the Neural Operator GitHub repository [2], [3], a Tensorized Fourier Neural Operator (TFNO) model is used as the basis for the network. The data was loaded using a modified script used in previous work by Yang Chen [1], which ensured the data was in the dimensions expected by the FNO.

As previously mentioned, the FNO was trained using 1800 different mesh cases (0.3 fibre volume and 0.5 fibre volume) and the rest of the data was withheld in order to test the network's ability to make predictions for cross-sectional microstructures it hadn't previously been challenged with.

The structure of the Fourier Neural Operator implemented in this work can be seen in figure 7 [3], where P and Q represent neural networks consisting of convolutional layers as well as fully-connected layers, however the implemented layers are up to the discretion of the user.

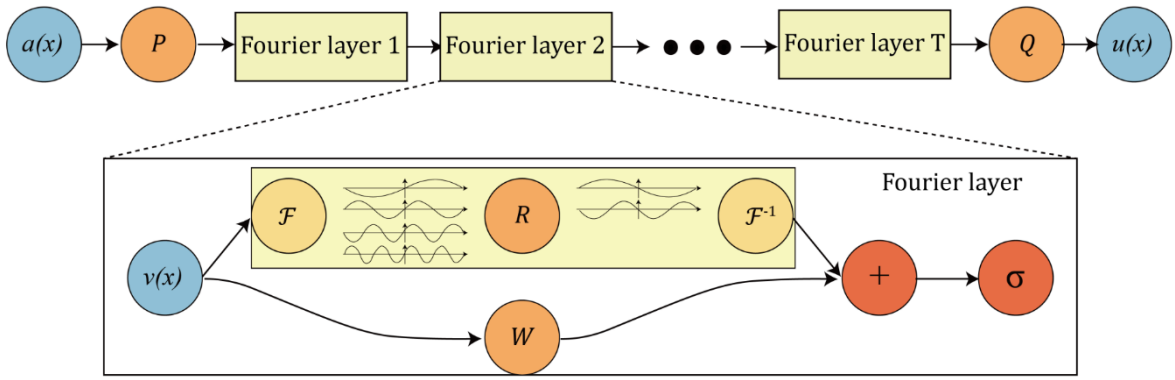


Figure 7: Fourier Neural Operator structure as proposed by Kovachki et al. [3]. The top layer shows the overarching structure of the FNO, and the bottom layer shows the architecture of the Fourier layers.

The function of Neural Network P is primarily to lift the input image to a higher dimensional channel using a pointwise lifting operator. In simple terms, the 2D image is segmented, and the segmentations are appended depth wise to create a new array or tensor, introducing an additional dimension to the data. This involves mapping pixels to a point in a different Euclidian space. This comes with the advantage that, by using lifting operators such as the kernel lifting operator, multiple inputs can be mapped to single outputs. The reverse is also true, which is useful in Neural Network Q.

Within the Fourier layer, the data is first converted to the frequency domain, where pooling and convolutions are able to be applied, before passing through a linear function, for example a weight tensor, which can be used to parameterize the network as well as filter out higher Fourier modes to reduce computational complexity. Ultimately an inverse Fourier transform is applied to the data, before it is

combined with data which has bypassed the Fourier transform phase, and passed through some linear, or non-linear transform.

3.2.1 Factorisation

Tensor factorisation extremely important in machine learning applications. Decomposing tensors allows for their dimensionality to be reduced, resulting in the tensor being expressed as a product of smaller factors. This eases the storage requirement for data computations in processing, in this instance, allowing the training of the neural operator to be faster and more efficient [69], [70], [71].

In this work, Tucker Decomposition was used, which has been found to require less storage to produce a given performance compared to traditional multilinear models [72].

3.2.2 Optimiser

Within the neural operator, the optimiser is used in order to reduce the loss function. The loss function relates the difference between the network prediction and the true value [73]. It is necessary to minimise this cost function by incremental gradient descent [74].

The model used for the research conducted in this paper incorporated an Adaptive Moment Estimation (ADAM) optimiser, which uses adaptive learning rates and momentum amongst other features in order to better the performance of standard stochastic gradient descent [75], [76]. Relevant and important features with respect to this paper are discussed further in section 5.

3.2.3 Scheduler

The scheduler enables the learning rate of the gradient descent to decay incrementally over a series of epochs, allowing for faster convergence as the learning process progresses.

3.3 Model Training

The model was trained with stress, strain and damage data separately on a 12th Gen Intel® Core™ i7-1260P processor with 16GB of RAM. The total running training times for each of the data fields were, 96, 130 and 182 minutes respectively. Without time constraints, it would be recommended to train the model on a high-memory GPU, or even investigate the use of parallelism across multiple GPUs to train the network.

Parallelism can involve running different layers of the model on different GPUs or splitting up batches to be processed on GPUs in parallel. It is even possible to split individual computations such as matrix multiplications, which are known to be computationally intensive for large amounts of data [77], [78].

It is important to note that finding the correct training parameters for the network also took several attempts, as the learning rate, weight decay, scheduler setting, training and test sample numbers, and batch sizes all affected the training accuracy.

The weight decay was set at a constant 1e-6 and learning rate updating was the chosen method in order to minimise the effect of overshooting during gradient descent as the model converged.

The learning rate was set to 1e-3 and the scheduler was configured to reduce the learning rate by a factor of 0.7 every 10 epochs. This enables the model not to overshoot in gradient descent, making convergence more difficult by not being sensitive enough to small changes in gradient near local minima. In addition, the learning rate reduction factor had to be great enough that the gradient descent steps did not become too small early in the training procedure, as this would have increased the time taken to reach reasonable model accuracy. The full stress and strain model training setups can be found in the appendices.

3.4 Experimental Methods

The model will be tested first on the base set of data to assess its ability to predict the stress, strain and damage characteristics of the RVEs. The trained FNO will then be assessed on lower resolution data which will be obtained using tensor interpolation functions which are part of the Pytorch Python package.

3.4.1 Low Resolution Data

Since the underlying architecture of the TFNO model from the Neural Operator GitHub repository [2], [3] does not allow a trained model to be assessed on data of different size to the training data, the problem was circumnavigated by interpolating the data at a ratio of 0.5, thereby decreasing the image size by half to 126x126 pixels. A second operation was then performed in order to expand each pixel to a larger pixel,

which was comprised of 4 smaller pixels for the benefit of the TFNO model, in order to effectively reduce the information contained in the input images by half, whilst retaining the original image dimensions of 251x251 pixels.

3.4.2 Normalisers

The effect of using different data normalisers on model convergence was also investigated as part of the strain data model. One 100 epoch training procedure was carried out using a range normaliser and another was carried out using a unit Gaussian normaliser. The results and potential factors affected said outcomes are displayed in sections 4 and 5.

4. Results and Analysis

4.1 Training Convergence

Due to the lack of access to a high-performance workstation, as well as the time constraints for the project, training for the strain and damage models took place over 100 epochs, with the stress training carried out over 60 epochs. The training sessions were run in five sets of 20 epoch and 3 sets for the stress model, due to the overheating experienced within the machine used.

Each epoch consisted of 10 iterations, with 160 training samples used in each epoch in batches of 16. The models were also tested on 32 samples in batches of 16.

Figure 8 displays the convergence of the strain model, in terms of training and validation errors. The zoomed section displays that the model was continuing to converge past the 100-epoch training phase, as the training error continued to decrease. The validation continues to follow this downward trend, and becomes lower than the training error at a couple of instances, signalling the model is beginning to overfit to the training data and encoding the noise in individual samples [79], [80].

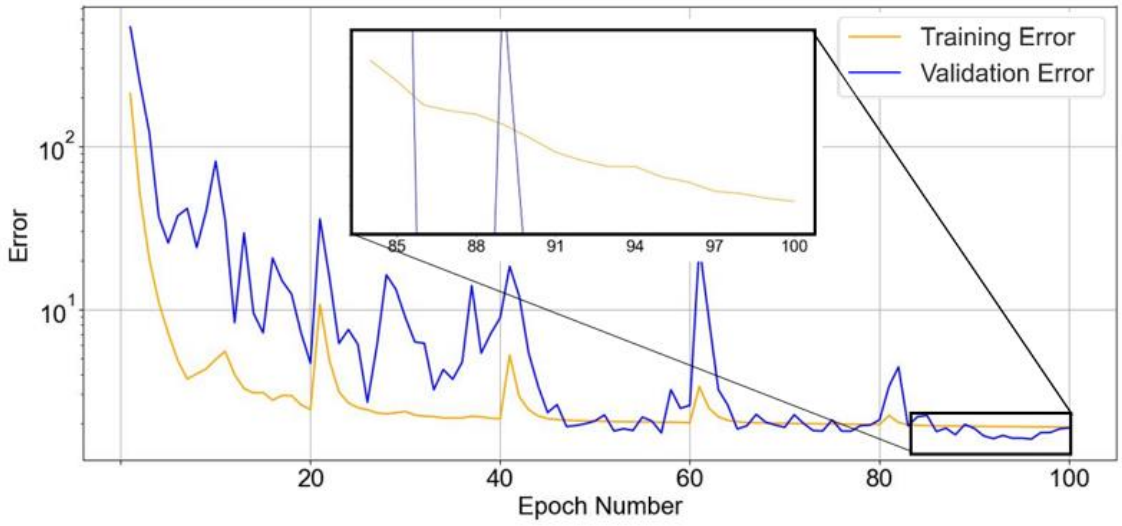


Figure 8: The strain model training and validation error relationship. The validation dropping below the training error around epoch 90 is a sign of overfitting.

A comparison of normalisers was carried out, also displayed in figure 9, which determines the Unit Gaussian Normaliser to lead the model to converge faster than the Range Normaliser which will be discussed further in section 5.

The periodic jumps in error coincide with the intermittent breaks during the training process. This caused the model to take several epochs in each 20-epoch training segment to reach the error at which the previous segment concluded on. The potential cause of this phenomenon is discussed in section 5.

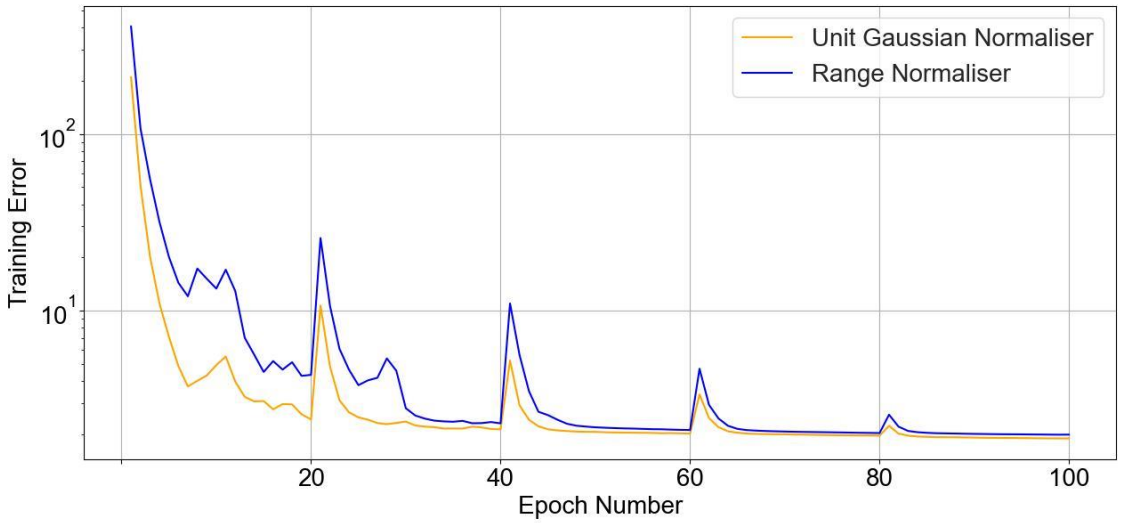


Figure 9: A comparison of strain model training errors using a range normaliser and a unit Gaussian normaliser to normalise the strain data.

The stress training procedure (figure 10) led to the fastest convergence of any of the 3 data fields, with the training error reaching 0.65 after just 60 epochs, compared to 1.89 and 7.85 after 100 epochs for the strain and damage models respectively. Curiously, the validation error was lower than the training error for a portion of the training in the first 10 epochs. Data imbalance is a suggested cause of this unusual training phenomenon, which will be discussed further within section 5.

The training was halted at 60 epochs for the stress model, since the validation and training errors were beginning to diverge, indicating that the model was overfitting to the training data causing it to lose generalisation capability.

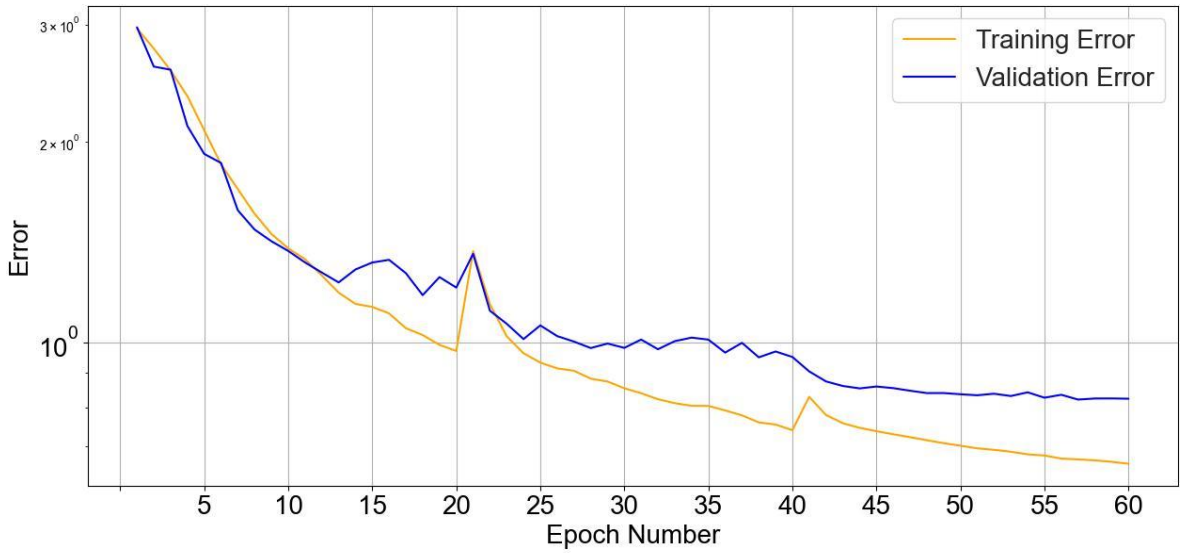


Figure 10: Training and validation error curves for the stress model, demonstrating rapid convergence towards 0 even over the course of 60 epochs.

The damage training proved to be the most challenging for the FNO, since the model is attempting to predict very localised information. In 100 epochs, the model was not able to converge on a meaningful result, however as seen in figure 11, the training error did continue to converge through the 100-epoch cut-off.

At this point, a divergence appears to be forming between the training and validation errors however, signalling that the model may be beginning to overfit to the training data at this stage.

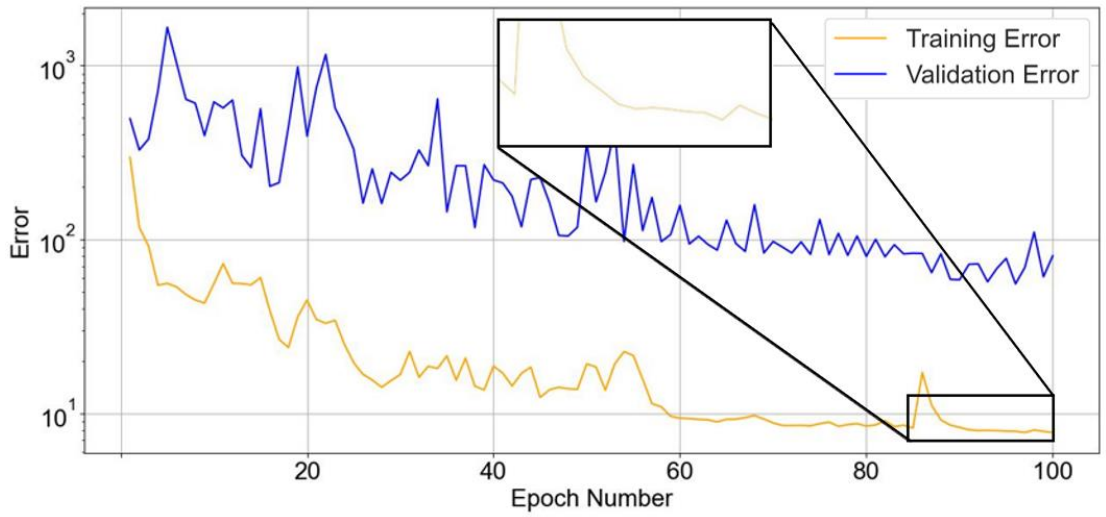


Figure 11: A comparison of damage model training and validation errors which show the reluctance of the damage model to converge past a training error of 7.84. The zoomed section demonstrates that the model was continuing to become more accurate beyond the 100-epoch training limit, however overfitting is thought to be a contributing factor

4.2 Strain Model

Figure 12 displays the model's ability to make strain field predictions firstly based on the data with fibre volume fractions of 0.3 and 0.5, which were notably used to train the model. Even with a relatively short training process, the model is able to predict strain fields well, with relatively little information loss. The figure also demonstrates the model's predictions for unseen volume fibre ratios of 0.25 and 0.6. The performance is not significantly distinguishable between training and unseen data, although it is apparent that the model is less accurate in areas containing large amounts of information (see Figure 12 (c)).

Strain Model Inputs, Ground Truths and Model Predictions in y1 and y2

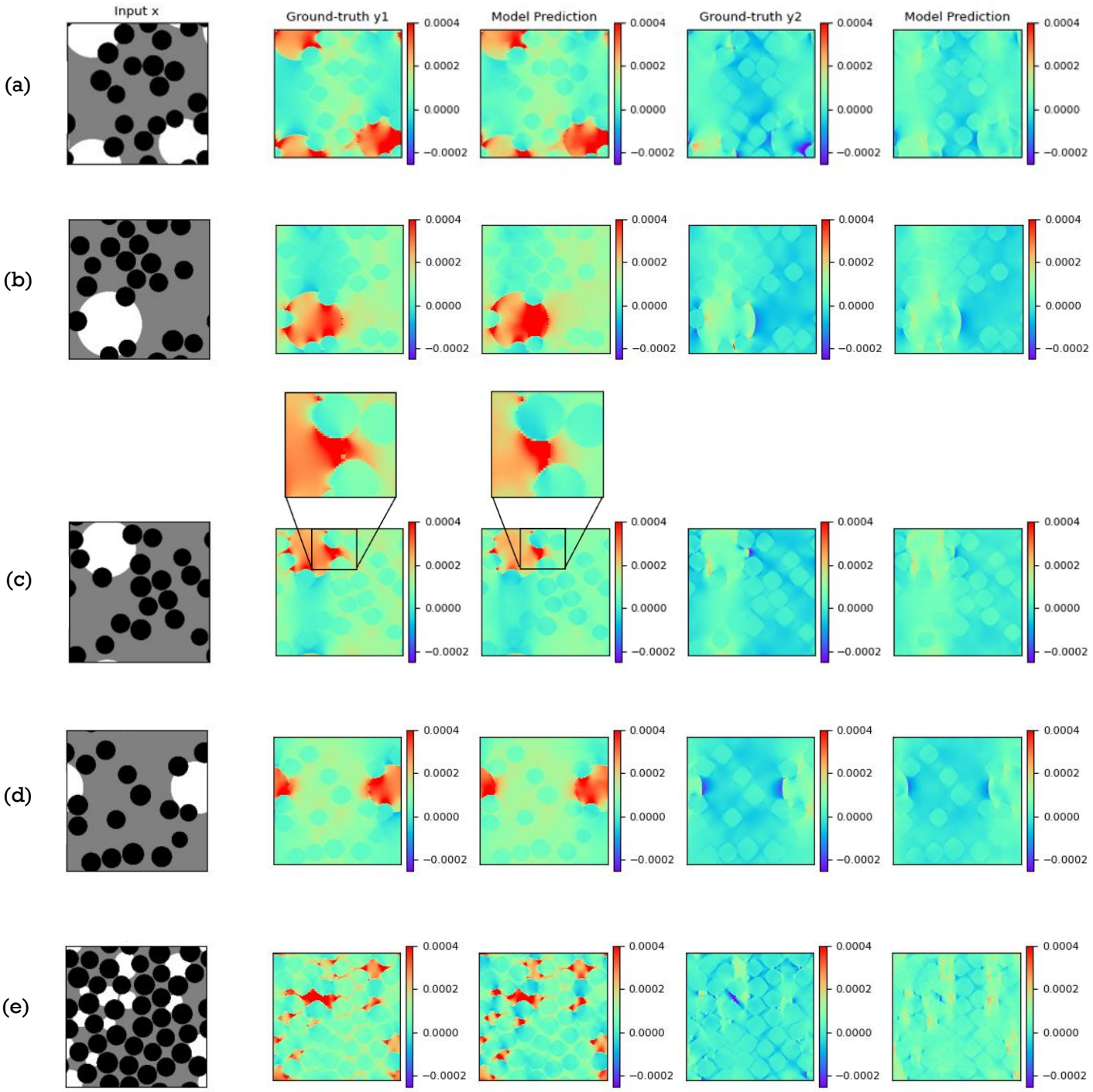


Figure 12: Predictions made by the strain model on a series of RVEs. (a), (b) and (c) are predictions based on meshes which were part of the training set. (c) also includes a zoomed image of a high information area in the ground-truth as well as the model's attempt to predict the same outcome. (d) and (e) are image inputs with fibre volume 0.25 and 0.6 respectively which were previously unseen by the model.

4.3 Stress Model

The stress model results are presented in a similar format, demonstrating the impressive performance of the stress model after just 60 training epochs to make predictions based on input cross-sections with fibre volume ratios of 0.3 and 0.5, as well as previously unseen cross-sections with varying microstructures, and fibre volume ratios of 0.25 and 0.6.

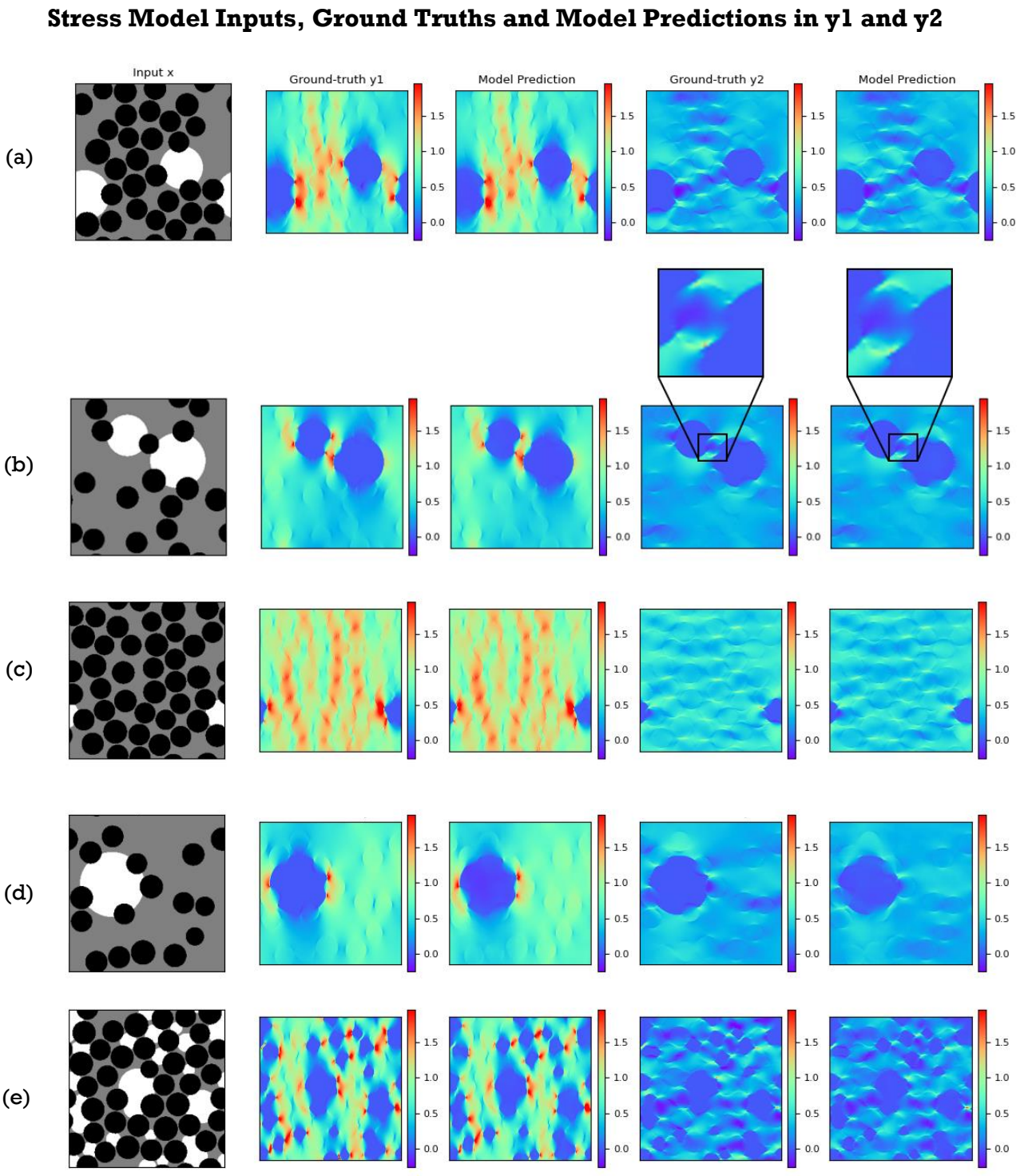


Figure 13: Predictions made by the strained model on a series of RVEs. (a) and (b) are predictions based on meshes which were part of the training set. (b) also includes a zoomed image for comparison, showing that the trained FNO is able to capture extremely localised information, displaying a stress concentration of just a few pixels in size. (c), (d) and (e) are image inputs with fibre volumes 0.25 and 0.8 which were previously unseen by the model yet the model still exhibits a high degree of prediction accuracy even for extremely localised behaviour.

Very fine localised stress concentrations are predicted successfully by the model, even for the previously unseen data. This contradicts the notion that the model may have been overfitting to the training data, but is useful in order to generate an explanation for the validation loss behaviour which is presented in section 5.

4.4 Low Resolution Data Evaluation

Input data was down sampled by a scale factor of 2, and reintroduced to the trained strain model as seen in figure 14. Notably, the TFNO model infrastructure provided in the aforementioned GitHub repository was not equipped to load a model trained on data of one size, to be used to evaluate input data of another size. Given the time restrictions, the underlying model code was not altered, however the issue was circumnavigated by interpolating the input data by a scale factor of 0.5, effectively halving the image size. Following this process, individual pixels were projected to larger pixels, which were 2x2 pixels each containing the same information in order to satisfy the data size needs of the trained TFNO.

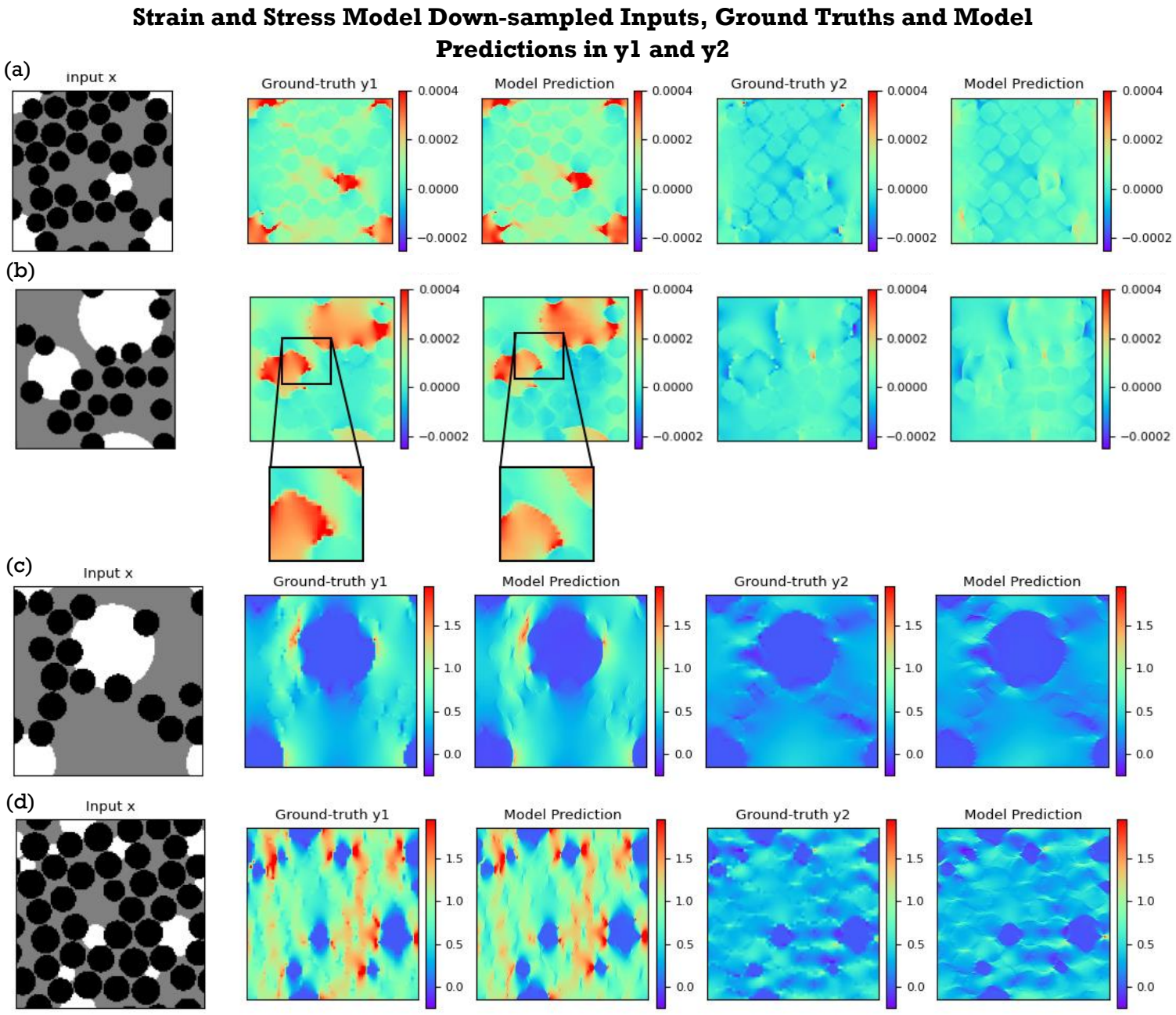


Figure 14: (a) and (b) display the strain model's predictions on the down sampled seen and unseen meshes respectively. (c) and (d) are the stress model predictions for unseen volume fibre ratios

Both the trained stress and strain models were able to deal with the down-sampled data well, and projected it back to the higher resolution upon which it was trained.

5. Discussion

The FNO is able to generalize quickly and accurately for both stress and strain data. It can predict global features of loaded carbon fibre reinforced composites even trained over a period of 100 epochs with 1800 training samples. Some information loss is evident at very localised features, such as small stress and strain concentrations within the microstructure.

Since the number of parameters present in the FNO used in this research is of the order of magnitude 10^4 times greater than the number of training samples, it can be reasonably determined that with a longer training period, the model would become more precise at reproducing very localised features, as more of the model parameters are fine-tuned to better estimate the underlying non-linear PDEs. This is reinforced by figures 8, 10 and 11, which show that the training and validation errors are continuing to reduce beyond the designated training period.

One trade-off in this regard is the balance of training data size to localised feature representation. Whilst an increased data size would decrease overfitting for a given number of training epochs, it would also lead to the neural network taking more time to accurately represent very localised information, since it will learn to generalise over a larger set of inputs. However, provided that the Neural Network is sufficiently deep, containing a large enough number of parameters (weights and biases), with sufficient training, this increased difficulty in predicting localised information has been shown that it can be overcome [81].

The poor convergence of the damage field model is perhaps not surprising given the complexity of the problem. For image data containing very localised information, particularly where the features are very complex, such as crack propagation, Neural Networks characteristically require more training data [82]. This is because variations in the data around the complex features are finer and less noticeable. It is therefore rational to suppose that the underlying PDEs in these areas are far more non-linear than in other regions where stress and strain variations are more global. Consequently, there are a large amount of network parameters that need to be modified in these areas, in order to be able to generalise well.

Linked to the above point, the disparity between training error and validation error may also suggest that in each iteration, the model is overfitting to the training data, reducing its capability to make predictions on the test samples. This is likely to be a by-product of the lack of training data, leading the model not to a see sufficient variety of features to learn from.

The Unit Gaussian Normaliser was also shown to be a more optimal optimiser in terms of convergence time for this particular application, compared to a Range Normaliser. This is likely because of the Gaussian Normalisation method's inherent increased resilience to outlier values, since it fits data based on the mean and standard deviation of the overall dataset, concentrating the data around the mean value between 0 and 1 in the case of the Unit Gaussian Normaliser [83].

Gaussian normalisation of data within the network can also help to smooth the gradient of the cost function compared to range normalisation, where the gradient magnitude can vary more sporadically for different features, if one feature has much greater values than another. This may lead the model to take longer to converge on local minima.

On this note, the reason for the faster convergence of the stress model compared to the strain model may also be gradient related. It has been found to be the case that models with a lower resolution in input data values will take longer to converge since the gradient updates to the neural network parameters are smaller and more random for shallower gradients.

The strain data resolution is naturally lower than the stress data resolution since in the RVE generation stage, stress, strain and damage field data is provided to double precision, or 64 bits. In double precision floats, the relative resolution of the data is determined by the number of significant digits. Double precision enables 15-17 digits of precision, however for the strain data, many of the leading digits are 0. This results in double precision effectively giving strain data to 9 or 10 significant digits of precision in some cases.

Logically, this results in lower relative information contained in the strain data after normalisation compared to the stress data, particularly for small strain values, potentially causing gradients to become less informative and more erratic as a minimum point is approached in the loss function [84], [85].

Regarding the stress model convergence, the training error – validation error relationship was worthy of note. Since the model adapts its parameters based on training data, one would typically expect the model to always perform better on data which is part of the training set. This was not the case in the stress model training procedure, however. From the outset, the model temporarily performed better on the validation data (test samples). This can be difficult to explain, especially given that, when retrained on exactly the same training parameters, the phenomenon ceased to occur.

Due to the shuffling of the training data, there is potential that, during the first 10 epochs, the training data was far more representative of one volume fibre ratio than another, and the test samples were largely comprised of data containing characteristics which were disproportionately represented in the training samples. This could have the effect that, at the early stage in the training, since one microstructure group was seen more often than the other, if the test sample belonged to the same volume fibre microstructure group, it will have been accurately evaluated by the model.

It is also important to address the periodic spikes in training and validation error, which coincided with breaks in training, during which the learning rate of the model was decreased by the predetermined factor. One possible trigger for this behaviour is a loss of momentum within the model, associated with the use of ADAM optimiser [86], [87].

ADAM optimiser incorporates a velocity term into the gradient descent process, in order to accelerate the convergence process. The velocity term is typically related to the exponential moving average (EMA) of the previous gradient terms, and at each iteration, the network parameters are updated based on both the gradient and momentum terms. Higher momentum will result in a greater inertial behaviour of the gradient descent process. As a result, pausing the network training and altering the learning rate can result in the loss of the momentum term within the gradient descent process, causing the following iterations to make changes to weights which don't follow the same trajectory as the optimum path calculated with the momentum term [88], [89], [87].

5.1 Overfitting

Overfitting becomes an issue where the model becomes very accurate at solving problems with the same geometries and, in this case, microstructural characteristics that it has been trained on, however it is not able to generalise well to variations in input images. For example, changes in volume fibre ratios [98], [99]. Typically, the neural network becomes so competent at solving the training data set, that noise and other anomalous features are 'memorised' by the neural network [90]. This causes the accuracy of predictions for unseen datasets to drop, often an undesirable feature for a neural network, which is certainly the case for materials behaviour prediction models.

Arguably the most significant cause for the overfitting phenomenon, is overparameterization [91]. Small datasets are typically more susceptible to overfitting than larger datasets, since the model has a far greater number of parameters than there are training samples. This gives a degree of over-malleability, whereby the neural network is able to alter weights and biases to generate solutions which are extremely fitted to the training data. In simple terms, the input data does not contain enough information to warrant representation by so many parameters [92], [93].

A key indication that a Neural Network is overfitted is given by the training error versus validation error relationship. The training error is given by a function of the difference of the model's prediction on a training data sample and the true result, similarly the validation error, is a function of the difference between the model's prediction on a sample separated from the training samples (usually referred to as test samples) and the true result.

If the two loss types begin to diverge, it insinuates that the model is becoming more accurate at evaluating training data samples, and less capable of generalising to solve unseen cases, at this point it can be deduced that the model is beginning to fit to noise and other redundant features of the input data [94].

As seen in figures 8 and 11, this phenomenon does appear to occur towards the ends of the training procedure.

5.2 Optimiser

As mentioned in section 3, this model uses a scheduler to incrementally decrease the learning rate at predetermined intervals. Often, this method yields good convergence rates, due to its ability to reduce over-shooting in the gradient descent process as the model reaches a local minimum [95]. In the case of the damage model, it is fair to comment that, the performance of this method for data with extremely complex, localised features, is not ideal.

More recently, extensions to the ADAM optimiser have been developed in order to improve convergence speed. One of these is YOGI [96], [101]. YOGI has been shown to outperform ADAM for many tasks, by enabling the model to adjust the learning rate for individual weight and bias parameters within the network.

6. Conclusions

A Fourier Neural Operator was successfully applied and trained to evaluate stress and strain-field characteristics for carbon fibre reinforced composite representative microstructures. With just 60 and 100 epochs required respectively to produce the predictions on unseen data presented in section 4 of the paper, it can be deduced that, with larger data sizes, and a longer training period, the model could become extremely proficient at making predictions for a wide range of carbon fibre composite meshes on stress and strain field data. Once trained, unseen meshes could be evaluated by the model in as little as 3.4 seconds on a CPU.

The effect of normalisation techniques on the input data was investigated, introducing theory as to why the unit Gaussian normaliser performs better than the range normaliser in this application.

The performance of the damage-field trained model was likely limited by a lack of training data, which restricted its ability to update sufficient network parameters to be able to predict the extremely complex, localised crack patterns in the mesh. It has been concluded that additional training data is required, in order for the model to see more instances of crack patterns, and utilise and update the abundance of network parameters in order to predict the complex crack formations.

A suggestion was also proposed as to why convergence takes longer for data containing small values, such is strain in this instance. In future research, alternative normalisation techniques should be considered in order not to lose relative resolution for small data points.

Given time constraints, the FNO code was not able to be modified sufficiently to evaluate the model using image data with different size meshes, however in an attempt to circumnavigate this issue, the input samples were able to be down-sampled to 126x126 pixel images and then each pixel was able to be expanded to a 2x2 array, with each element containing the original pixel data, enabling the inputs to retain the characteristic resolution loss of the down-sampled images, whilst fitting the 251x251 pixel dimensions required by the trained model. It was interesting to observe that the model appeared able to make predictions on the low-resolution inputs, in the high resolution upon which it was trained with high accuracy.

7. Future Work

Future work in this field should begin with a more extensive training of the Fourier Neural Operator on a larger dataset across a larger number of epochs. As demonstrated in this paper, this would alleviate small inaccuracy issues, particularly for data containing very localised features.

Investigation should be made into alternatives to double precision floats for small data values, in order to allow for increased relative resolution to aid model convergence.

In addition, it would be recommended to attempt to test the network on data of different resolutions, in order to assess its capability to generalise to different meshes. It would be particularly interesting to investigate the effect super-high and super-low resolution data might have on the outcomes of the network prediction, and indeed how far each way this capacity can be pushed whilst maintaining a suitable degree of accuracy.

Further still, the problem should be extended to 3D geometries, or 2D geometries with more complex transverse loading conditions, potentially including dynamically varying loads.

8. Appendices

Appendix I – Stress Model Setup

```
import torch
import sys
import os
from neuralop.models import TFNO
from neuralop import Trainer
from neuralop.utils import count_params
from neuralop import LpLoss, H1Loss
from yc_utils.LOAD_fracture import load_fracture_mesh_stress
import matplotlib.pyplot as plt
import numpy as np
import datetime

inference = False

# list data directories
lst_dir0 = list()
lst_idx_UC = list()
dir_mesh = list()

if inference==True: # unseen data directories
    shuffle = False

    lst_dir0.append( r"D:\data\unseen\L0.05_vf0.25\h0.0002" )
    lst_dir0.append( r"D:\data\unseen\L0.05_vf0.6\h0.0002" )
    dir_mesh = r"D:\data\unseen\mesh"

else:  # training data directories
    shuffle = True

    lst_dir0.append( r"D:\data\L0.05_vf0.3\h0.0002" )
    lst_dir0.append( r"D:\data\L0.05_vf0.5\h0.0002" )
    dir_mesh = r"D:\data\mesh"

fieldtype = "stress"
# load the data
train_loader, test_loaders, output_encoder = load_fracture_mesh_stress(
                        lst_dir0=lst_dir0,
                        dir_mesh=dir_mesh,
                        n_train=160, n_tests=[32],
                        batch_size=16, test_batch_sizes=[16],
                        test_resolutions=[251],
                        train_resolution=251,
                        grid_boundaries=[[0,1],[0,1]],
                        positional_encoding=True,
                        encode_input=False,
                        encode_output=False,
                        encoding='channel-wise')


# create a tensorised FNO model
model = TFNO( n_modes=(251, 251),
              in_channels=3, out_channels=3,
              hidden_channels=32, projection_channels=64,
              factorization='tucker', rank=0.42)
```

```
#load the trained model for the evaluation stage
#model.load_state_dict(torch.load('C:/Users/Elliot/Documents/fyp/fno/stressmodel.pth'))
##############################################################################################
#model.eval()
##############################################################################################

n_params = count_params(model)
print(f'\nOur model has {n_params} parameters.')
sys.stdout.flush()

# create the optimizer
optimizer = torch.optim.Adam(model.parameters(),
                             lr=1e-3,
                             weight_decay=1e-6)
#scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=30)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.7)

# creating the losses
l2loss = LpLoss(d=2, p=2, reduce_dims=[0,1])
h1loss = H1Loss(d=2, reduce_dims=[0,1])

train_loss = h1loss
eval_losses={'h1': h1loss, 'l2': l2loss}


# %%
print('\n### MODEL ###\n', model)
print('\n### OPTIMIZER ###\n', optimizer)
print('\n### SCHEDULER ###\n', scheduler)
print('\n### LOSSES ###')
print(f'\n * Train: {train_loss}')
print(f'\n * Test: {eval_losses}')
sys.stdout.flush()

# Create the trainer
trainer = Trainer(model, n_epochs=20,
                  device=device,
                  mg_patching_levels=0,
                  wandb_log=False,
                  log_test_interval=1,
                  use_distributed=False,
                  verbose=True)

# Train the model
i = trainer.train(train_loader, test_loaders,
                      output_encoder,
                      model,
                      optimizer,
                      scheduler,
                      regularizer=False,
                      training_loss=train_loss,
                      eval_losses=eval_losses)
# save the model
torch.save(model.state_dict(), 'C:/Users/Elliot/Documents/fyp/fno/stressmodel.pth')
```

Appendix II – Strain Model Setup

```
import torch
import sys
import os
from neuralop.models import TFNO
from neuralop import Trainer
from neuralop.utils import count_params
from neuralop import LpLoss, H1Loss
from yc_utils.LOAD_fracture import load_fracture_mesh_stress
import matplotlib.pyplot as plt
import numpy as np
import datetime

inference = False

# list data directories
lst_dir0 = list()
lst_idx_UC = list()
dir_mesh = list()

if inference==True: # unseen data directories
    shuffle = False

    lst_dir0.append( r"D:\data\unseen\L0.05_vf0.25\h0.0002" )
    lst_dir0.append( r"D:\data\unseen\L0.05_vf0.6\h0.0002" )
    dir_mesh = r"D:\data\unseen\mesh"

else:  # training data directories
    shuffle = True

    lst_dir0.append( r"D:\data\L0.05_vf0.3\h0.0002" )
    lst_dir0.append( r"D:\data\L0.05_vf0.5\h0.0002" )
    dir_mesh = r"D:\data\mesh"

fieldtype = "stress"
# load the data
train_loader, test_loaders, output_encoder = load_fracture_mesh_strain(
                          lst_dir0=lst_dir0,
                          dir_mesh=dir_mesh,
                          n_train=160, n_tests=[32],
                          batch_size=16, test_batch_sizes=[16],
                          test_resolutions=[251],
                          train_resolution=251,
                          grid_boundaries=[[0,1],[0,1]],
                          positional_encoding=True,
                          encode_input=False,
                          encode_output=False,
                          encoding='channel-wise')


# create a tensorised FNO model
model = TFNO( n_modes=(251, 251),
              in_channels=3, out_channels=3,
              hidden_channels=32, projection_channels=64,
              factorization='tucker', rank=0.42)

#load the trained model for the evaluation stage

#model.load_state_dict(torch.load('C:/Users/Elliot/Documents/fyp/fno/strainmodel.pth'))
##############################################################################################
#model.eval()
##############################################################################################
```

```
n_params = count_params(model)
print(f'\nOur model has {n_params} parameters.')
sys.stdout.flush()

# create the optimizer
optimizer = torch.optim.Adam(model.parameters(),
                             lr=1e-3,
                             weight_decay=1e-6)
#scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=30)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.7)

# creating the losses
l2loss = LpLoss(d=2, p=2, reduce_dims=[0,1])
h1loss = H1Loss(d=2, reduce_dims=[0,1])

train_loss = h1loss
eval_losses={'h1': h1loss, 'l2': l2loss}


# %%
print('\n### MODEL ###\n', model)
print('\n### OPTIMIZER ###\n', optimizer)
print('\n### SCHEDULER ###\n', scheduler)
print('\n### LOSSES ###')
print(f'\n * Train: {train_loss}')
print(f'\n * Test: {eval_losses}')
sys.stdout.flush()

# Create the trainer
trainer = Trainer(model, n_epochs=20,
                  device=device,
                  mg_patching_levels=0,
                  wandb_log=False,
                  log_test_interval=1,
                  use_distributed=False,
                  verbose=True)

# Train the model
i = trainer.train(train_loader, test_loaders,
                      output_encoder,
                      model,
                      optimizer,
                      scheduler,
                      regularizer=False,
                      training_loss=train_loss,
                      eval_losses=eval_losses)
# save the model
torch.save(model.state_dict(), 'C:/Users/Elliot/Documents/fyp/fno/strainmodel.pth')
```

9. References

[1] Chen, Yang & Dodwell, Tim & Chuaqui, Tomás & Butler, Richard. (2022). Full-field prediction of stress and fracture patterns in composites using deep learning and self-attention. 10.13140/RG.2.2.17744.17924.

[2] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar A., ''Fourier Neural Operator for Parametric Partial Differential Equations'', ICLR, 2021. doi:10.48550/arXiv.2010.08895.

[3] Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar A., ''Neural Operator: Learning Maps Between Function Spaces'', JMLR, 2021. doi:10.48550/arXiv.2108.08481.

[4] Aziz Shah, Mohamad & Yunus, Mohd Azmi & Abdul Rani, Muhamad Norhisham & Wan Iskandar Mirza, Wan Imaan Izhan & Sani, Shahrir. (2021). An Improved Method for Dynamic Behaviour Prediction of Carbon Fibre Reinforced Epoxy (CFRE) using Finite Element Model Updating. IOP Conference Series: Materials Science and Engineering. 1041. 012064. 10.1088/1757-899X/1041/1/012064.

[5] Zhenchao Qi, Nanxi Zhang, Yong Liu, Wenliang Chen, Prediction of mechanical properties of carbon fiber based on cross-scale FEM and machine learning, Composite Structures, Volume 212, 2019, Pages 199-206, ISSN 0263-8223, https://doi.org/10.1016/j.compstruct.2019.01.042.

[6] S. Dey, T. Mukhopadhyay, S.K. Sahu, G. Li, H. Rabitz, S. Adhikari, Thermal uncertainty quantification in frequency responses of laminated composite plates, Composites Part B: Engineering, Volume 80, 2015, Pages 186-197, ISSN 1359-8368, https://doi.org/10.1016/j.compositesb.2015.06.006.

[7] M A S A Shah et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1041 012064, DOI 10.1088/1757-899X/1041/1/012064.

[8] Marco Monti, Marta Palenzona, Francesca Fiorino, Felix Baudach, Andrea Onnis, Andrea Romeo, Design, manufacturing and FEA prediction of the mechanical behavior of a hybrid-molded polycarbonate / continuous carbon fiber reinforced composite component, Composites Part B: Engineering, Volume 238, 2022, 109891, ISSN 1359-8368, https://doi.org/10.1016/j.compositesb.2022.109891.

[9] Kenneth L. Reifsnider, Vitauts Tamuzs, Shinji Ogihara, On nonlinear behavior in brittle heterogeneous materials, Composites Science and Technology, Volume 66, Issue 14, 2006, Pages 2473-2478, ISSN 0266-3538, https://doi.org/10.1016/j.compscitech.2006.04.003.

[10] Borzeszkowski, Bartosz & Lubowiecka, Izabela & Sauer, Roger. (2022). Nonlinear material identification of heterogeneous isogeometric Kirchhoff–Love shells. Computer Methods in Applied Mechanics and Engineering. 390. 114442. 10.1016/j.cma.2021.114442.

[11] Li, Angran & Chen, Ruijia & Barati Farimani, Amir & Zhang, Yongjie. (2020). Reaction diffusion system prediction based on convolutional neural network. Scientific Reports. 10. 3894. 10.1038/s41598-020-60853-2.

[12] Kononenko, O., & Kononenko, I. (2018). Machine Learning and Finite Element Method for Physical Systems Modeling. Retrieved April 24, 2023, from https://arxiv.org/pdf/1801.07337.

[13] Phellan Aro, Renzo & Hachem, Bahe & Clin, Julien & Duong, Luc. (2020). Real-time biomechanics using the finite element method and machine learning: Review and perspective. Medical Physics. 48. 10.1002/mp.14602.

[14] Sun, Y., Hanhan, I., Sangid, M.D., & Lin, G. (2020). Predicting Mechanical Properties from Microstructure Images in Fiber-reinforced Polymers using Convolutional Neural Networks. ArXiv, abs/2010.03675.

[15] Jamie M. Taylor, David Pardo, Ignacio Muga, A Deep Fourier Residual method for solving PDEs using Neural Networks, Computer Methods in Applied Mechanics and Engineering, Volume 405, 2023, 115850, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2022.115850.

[16] Ben-Israel, Adi. (1965). A Newton-Raphson method for the solution of systems of equations. Journal of Mathematical Analysis and Applications. 3. 94-98. 10.1007/BF02760034.

[17] Lagaris, Isaac & Likas, Aristidis & Fotiadis, Dimitrios. (1998). Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks. 9. 987-1000. 10.1109/72.712178.

[18] Sacchetti, A. & Bachmann, Benjamin & Loffel, Kaspar & Kunzi, Urs-Martin & Paoli, Beatrice. (2022). Neural Networks to Solve Partial Differential Equations: A Comparison With Finite Elements. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3160186.

[19] Gao, Han & Sun, Luning & Wang, Jian-Xun. (2020). PhyGeoNet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parametric PDEs on Irregular Domain.

[20] Jiaqi Ding, Junhai Xu, Jianguo Wei, Jijun Tang, Fei Guo, A multi-scale multi-model deep neural network via ensemble strategy on high-throughput microscopy image for protein subcellular localization, Expert Systems with Applications, Volume 212, 2023, 118744, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2022.118744.

[21] Rafidison, Maminiaina & Ramafiarisona, Hajasoa & Randriamitantsoa, Paul & Rafanantenana, Sabine & Toky, Faniriharisoa & Rakotondrazaka, Lovasoa & Rakotomihamina, Andry. (2023). Image Classification Based on Light Convolutional Neural Network Using Pulse Couple Neural Network. Computational Intelligence and Neuroscience. 2023. 1-17. 10.1155/2023/7371907.

[22] Analysis of convolutional neural network image classifiers in a hierarchical max-pooling model with additional local pooling, Journal of Statistical Planning and Inference, Volume 224, 2023, Pages 109-126, ISSN 0378-3758, https://doi.org/10.1016/j.jspi.2022.11.001.

[23] Montesinos-López, Osval & Montesinos, Abelardo & Crossa, Jose. (2022). Convolutional Neural Networks. 10.1007/978-3-030-89010-0_13.

[24] Kim, Hwajoon. (2019). The Definition of Convolution in Deep Learning by using Matrix. Journal of Engineering and Applied Sciences.

14. 2272-2275. 10.36478/jeasci.2019.2272.2275.

[25] Mairal, Julien & Koniusz, Piotr & Harchaoui, Zaid & Schmid, Cordelia. (2014). Convolutional Kernel Networks. Advances in neural information processing systems.

[26] Sun, Zhun & Ozay, Mete & Okatani, Takayuki. (2015). Design of Kernels in Convolutional Neural Networks for Image Classification.

[27] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

[28] Gholamalinejad, Hossein & Khosravi, Hossein. (2020). Pooling Methods in Deep Neural Networks, a Review.

[29] KHOO, Y., LU, J., & YING, L. (2021). Solving parametric PDE problems with artificial neural networks. European Journal of Applied Mathematics, 32(3), 421-435. doi:10.1017/S0956792520000182

[30] Li, Zongyi & Kovachki, Nikola & Azizzadenesheli, Kamyar & Liu, Burigede & Bhattacharya, Kaushik & Stuart, Andrew & Anandkumar, Animashree. (2020). Fourier Neural Operator for Parametric Partial Differential Equations.

[31] Guo, X., Li, W., & Iorio, F. (2016). Convolutional Neural Networks for Steady Flow Approximation. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[32] Kabri, Samira & Roith, Tim & Tenbrinck, Daniel & Burger, Martin. (2023). Resolution-Invariant Image Classification based on Fourier Neural Operators.

[33] Raonić, Bogdan & Molinaro, Roberto & Rohner, Tobias & Mishra, Siddhartha & Bezenac, Emmanuel. (2023). Convolutional Neural Operators. 10.48550/arXiv.2302.01178.

[34] Yang, B., Bender, G., Le, Q.V., & Ngiam, J. (2019). CondConv: Conditionally Parameterized Convolutions for Efficient Inference. Neural Information Processing Systems.

[35] E. Georganas et al., "Anatomy of High-Performance Deep Learning Convolutions on SIMD Architectures," SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, 2018, pp. 830-841, doi: 10.1109/SC.2018.00069.

[36] Liu, Albert & Law, Oscar. (2021). Convolution Optimization. 10.1002/9781119810483.ch5.

[37] Y. Ma, Y. Cao, S. Vrudhula and J. -s. Seo, "Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 7, pp. 1354-1367, July 2018, doi: 10.1109/TVLSI.2018.2815603.

[38] Wang, C.J., & Golland, P. (2022). Discretization Invariant Learning on Neural Fields.

[39] Hatano, Naoya & Ikeda, Masahiro & Ishikawa, Isao & Sawano, Yoshihiro. (2021). A Global Universality of Two-Layer Neural Networks with ReLU Activations. Journal of Function Spaces. 2021. 1-3. 10.1155/2021/6637220.

[40] Felix Voigtlaender, The universal approximation theorem for complex-valued neural networks, Applied and Computational Harmonic Analysis, Volume 64, 2023, Pages 33-61, ISSN 1063-5203, https://doi.org/10.1016/j.acha.2022.12.002.

[41] Woochul Jung, A. Rojas, Universal approximation on metric spaces, Journal of Mathematical Analysis and Applications, Volume 526, Issue 1, 2023, 127157, ISSN 0022-247X, https://doi.org/10.1016/j.jmaa.2023.127157.

[42] Campos, Juan & Yaroslavsky, Leonid & Moreno, A & Yzuel, Maria. (2002). Integration in the Fourier domain for restoration of a function from its slope: Comparison of four methods. Optics letters. 27. 1986-8. 10.1364/OL.27.001986.

[43] Hsu, Y.-C., C.-H. Yu, and M.J. Buehler, Using deep learning to predict fracture patterns in crystalline solids. Matter, 2020. 3(1): p. 197-211

[44] Montes de Oca Zapiain, D., J.A. Stewart, and R. Dingreville, Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods. npj Computational Materials, 2021. 7(1): p. 1-11

[45] Bidokhti, Amir & Ghaemmaghami, Shahrokh. (2022). Compound short- and long-term memory for memory augmented neural networks. Engineering Applications of Artificial Intelligence. 116. 105450. 10.1016/j.engappai.2022.105450.

[46] Bolboaca, Roland & Piroska, Haller. (2023). Performance Analysis of Long Short-Term Memory Predictive Neural Networks on Time Series Data. Mathematics. 11. 1432. 10.3390/math11061432.

[47] Eric L. Buehler, Markus J. Buehler, End-to-end prediction of multimaterial stress fields and fracture patterns using cycle-consistent adversarial and transformer neural networks, Biomedical Engineering Advances, Volume 4, 2022, 100038, ISSN 2667-0992, https://doi.org/10.1016/j.bea.2022.100038.

[48] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., & Bengio, Y. (2014). Generative Adversarial Nets. NIPS.

[49] Xuewei Li, Jinming Ma, Jian Yu, Mankun Zhao, Mei Yu, Hongwei Liu, Weiping Ding, Ruiguo Yu, A structure-enhanced generative adversarial network for knowledge graph zero-shot relational learning, Information Sciences, Volume 629, 2023, Pages 169-183, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2023.01.113.

[50] Zhenguo Nie, Haoliang Jiang, and Levent Burak Kara. Stress field prediction in cantilevered structures using convolutional neural networks. Journal of Computing and Information Science in Engineering, 20(1), 2020

[51] Wang, Yinan & Oyen, Diane & Guo, Weihong & Mehta, Anishi & Scott, Cory & Panda, Nishant & Fernández-Godino, M. Giselle & Srinivasan, Gowri & Yue, Xiaowei. (2021). StressNet - Deep learning to predict stress with fracture propagation in brittle materials. npj Materials Degradation. 5. 10.1038/s41529-021-00151-y.

[52] Nie, Zhenguo & Jiang, Haoliang & Kara, Levent. (2019). Stress Field Prediction in Cantilevered Structures Using Convolutional Neural Networks. 10.1115/DETC2019-98472.

[53] Mohammadzadeh, S. and E. Lejeune, Predicting mechanically driven full-field quantities of interest with deep learning-based metamodels. Extreme Mechanics Letters, 2022. 50: p.101566

[54] Anindya Bhaduri, Ashwini Gupta, Lori Graham-Brady, Stress field prediction in fiber-reinforced composite materials using a deep learning approach, Composites Part B: Engineering, Volume 238, 2022, 109879, ISSN 1359-8368, https://doi.org/10.1016/j.compositesb.2022.109879.

[55] Wei, Xiong & Ma, Muyuan & Huang, Xiaomeng & Sun, Pei & Tian, Yang. (2023). KoopmanLab: machine learning for solving complex physics equations. 10.48550/arXiv.2301.01104.

[56] Lee, S. (2022). Mesh-Independent Operator Learning for Partial Differential Equations. ICML 2022 2nd AI for Science Workshop. Retrieved from https://openreview.net/forum?id=JUtZG8-2vGp

[57] Lee, S. (2023). Mesh-Independent Operator Learning for PDEs using Set Representations. Retrieved from https://openreview.net/forum?id=7d-d0BFz6Hf

[58] Mehran, Meer & Pittie, Tanu & Chakraborty, Souvik & Krishnan, N M Anoop. (2022). Learning the stress-strain fields in digital composites using Fourier neural operator. iScience. 25. 105452. 10.1016/j.isci.2022.105452.

[59] Cordonnier, Jean-Baptiste & Loukas, Andreas & Jaggi, Martin. (2019). On the Relationship between Self-Attention and Convolutional Layers.

[60] Gao, Changxia & Zhang, Ning & Li, Youru & Bian, Feng & Wan, Huaiyu. (2022). Self-attention-based time-variant neural networks for multi-step time series forecasting. Neural Computing and Applications. 34. 10.1007/s00521-021-06871-1.

[61] He, Haoyu & Liu, Jing & Pan, Zizheng & Cai, Jianfei & Zhang, Jing & Tao, Dacheng & Zhuang, Bohan. (2021). Pruning Self-attentions into Convolutional Layers in Single Path.

[62] Osborne, Alfred. (2010). Periodic Boundary Conditions. 10.1016/S0074-6142(10)97040-X.

[63] Chen, Y., et al., FFT phase-field model combined with cohesive composite voxels for fracture of composite materials with interfaces. Computational Mechanics, 2021. 68(2): p. 433-457.

[64] Chen, Y., et al., A FFT solver for variational phase-field modelling of brittle fracture. Computer Methods in Applied Mechanics and Engineering, 2019. 349: p. 167-190.

[65] S. Brisard, L. Dormieux, Combining Galerkin approximation techniques with the principle of Hashin and Shtrikman to derive a new FFT-based numerical method for the homogenization of composites, Computer Methods in Applied Mechanics and Engineering, Volumes 217–220, 2012, Pages 197-212, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2012.01.003.

[66] T.W.J. de Geus, J. Vondřejc, J. Zeman, R.H.J. Peerlings, M.G.D. Geers, Finite strain FFT-based non-linear solvers made simple, Computer Methods in Applied Mechanics and Engineering, Volume 318, 2017, Pages 412-430, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2016.12.032.

[67] Cruzado, Aitor & Segurado, Javier & Hartl, Darren & Benzerga, Amine. (2021). A variational fast Fourier transform method for phase-transforming materials. Modelling and Simulation in Materials Science and Engineering. 29. 10.1088/1361-651X/abe4c7.

[68] Ladecký, Martin & Leute, Richard & Falsafi, Ali & Pultarova, Ivana & Pastewka, Lars & Junge, Till & Zeman, Jan. (2022). Optimal FFT-accelerated Finite Element Solver for Homogenization. 10.48550/arXiv.2203.02962.

[69] Yingyue Bi, Yingcong Lu, Zhen Long, Ce Zhu, Yipeng Liu, Chapter 1 - Tensor decompositions: computations, applications, and challenges, Editor(s): Yipeng Liu, Tensors for Data Processing, Academic Press, 2022, Pages 1-30, ISBN 9780128244470, https://doi.org/10.1016/B978-0-12-824447-0.00007-8.

[70] Rabanser, Stephan & Shchur, Oleksandr & Günnemann, Stephan. (2017). Introduction to Tensor Decompositions and their Applications in Machine Learning.

[71] Liu, Y., Liu, J., Long, Z., Zhu, C. (2022). Tensor Decomposition. In: Tensor Computation for Data Analysis. Springer, Cham.

[72] Thi Hieu Luu, Yvon Maday, Matthieu Guillo, Pierre Guérin, A new method for reconstruction of cross-sections using Tucker decomposition, Journal of Computational Physics, Volume 345, 2017, Pages 189-206, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2017.05.019.

[73] Janocha, Katarzyna & Czarnecki, Wojciech. (2017). On Loss Functions for Deep Neural Networks in Classification. Schedae Informaticae. 25. 10.4467/20838476SI.16.004.6185.

[74] Ee, Weinan & Ma, Chao & Wu, Lei. (2019). Analysis of the Gradient Descent Algorithm for a Deep Neural Network Model with Skip-connections.

[75] Ruder, Sebastian. (2016). An overview of gradient descent optimization algorithms.

[76] Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.

[77] Weng, L., & Brockman, G. (2022, June 9). Techniques for training large neural networks. https://openai.com/research/techniques-for-training-large-neural-networks.

[78] Pourghassemi, Behnam & Zhang, Chenghao & Lee, Joo & Chandramowlishwaran, Aparna. (2020). Brief Announcement: On the Limits of Parallelizing Convolutional Neural Networks on GPUs.

[79] Salman, Shaeke & Liu, Xiuwen. (2019). Overfitting Mechanism and Avoidance in Deep Neural Networks.

[80] Kadhim, Zahraa & Abdullah, Hasanen & Ghathwan, Khalil. (2023). Automatically Avoiding Overfitting in Deep Neural Networks by Using Hyper-Parameters Optimization Methods. International Journal of

Online and Biomedical Engineering (iJOE). 19. 146-162. 10.3991/ijoe.v19i05.38153.

[81] Linjordet, T., Balog, K. (2019). Impact of Training Dataset Size on Neural Answer Selection Models. In: Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D. (eds) Advances in Information Retrieval. ECIR 2019. Lecture Notes in Computer Science(), vol 11437. Springer, Cham. https://doi.org/10.1007/978-3-030-15712-8_59

[82] M. Worthington, H.B. Chew, Crack path predictions in heterogeneous media by machine learning, Journal of the Mechanics and Physics of Solids, Volume 172, 2023, 105188, ISSN 0022-5096, https://doi.org/10.1016/j.jmps.2022.105188.

[83] Zhang, Xinhua. (2016). Gaussian Distribution. 10.1007/978-1-4899-7502-7_107-1.

[84] Goldberg, D. (1991). What Every Computer Scientist Should Know about Floating-Point Arithmetic. ACM Comput. Surv., 23(1), 5–48. doi:10.1145/103162.103163.

[85] Greenbaum, A., & Chartier, T. P. (2012). Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms. USA: Princeton University Press.

[86] Bernstein, Jeremy & Mingard, Chris & Huang, Kevin & Azizan, Navid & Yue, Yisong. (2023). Automatic Gradient Descent: Deep Learning without Hyperparameters.

[87] Zhao, Xiaoyi & Xie, Ping & Xing, Ling & Zhang, Gaoyuan & Ma, Huahong. (2023). Clustered Federated Learning Based on Momentum Gradient Descent for Heterogeneous Data. Electronics. 12. 1972. 10.3390/electronics12091972.

[88] Kan, T. & Gao, Z. & Yang, C.. (2020). Stochastic Gradient Descent Method of Convolutional Neural Network Using Fractional-Order Momentum. Moshi Shibie yu.

[89] Abdulkadirov, Ruslan & Lyakhov, Pavel. (2023). A new approach to training neural networks using natural gradient descent with momentum based on Dirichlet distributions. Computer Optics. 47. 160-169. 10.18287/2412-6179-CO-1147.

[90] Allamy, Haider. (2014). METHODS TO AVOID OVER-FITTING AND UNDER-FITTING IN SUPERVISED MACHINE LEARNING (COMPARATIVE STUDY).

[91] Salman, Shaeke & Liu, Xiuwen. (2019). Overfitting Mechanism and Avoidance in Deep Neural Networks.

[92] Maity, Subha & Roy, Saptarshi & Xue, Songkai & Yurochkin, Mikhail & Sun, Yuekai. (2022). How does overparametrization affect performance on minority groups?. 10.48550/arXiv.2206.03515.

[93] Whittaker, Gerald & Confesor, Remegio & Luzio, M. & Arnold, Jeff. (2010). Detection of Overparameterization and Overfitting in an Automatic Calibration of SWAT. Transaction of ASABE. 53. 10.13031/2013.34909.

[94] Gavrilov, Andrei & Jordache, Alex & Vasdani, Maya & Deng, Jack. (2018). Preventing Model Overfitting and Underfitting in Convolutional Neural Networks. International Journal of Software Science and Computational Intelligence. 10. 19-28. 10.4018/IJSSCI.2018100102.

[95] Li, Yuanyuan & Zhang, Qianqian & Won, Daehan & Lu, Fake & Yoon, Sang Won. (2020). Learning Rate Optimization in Convolutional Neural Networks for Medical Images Classification.

[96] Zaheer, M., Reddi, S. J., Sachan, D., Kale, S., & Kumar, S. (2018).

[97] Sepasdar, Reza & Karpatne, Anuj & Shakiba, Maryam. (2021). A Data-Driven Approach to Full-Field Damage and Failure Pattern Prediction in Microstructure-Dependent Composites using Deep Learning.

[98] Lawrence, Steve & Giles, C.Lee. (2000). Overfitting and neural networks: Conjugate gradient and backpropagation. Proceedings of the International Joint Conference on Neural Networks. 1. 114-119 vol.1. 10.1109/IJCNN.2000.857823. https://doi.org/10.1007/978-3-030-74386-4_2.

[99] Li, Zhu & Zhou, Zhi-Hua & Gretton, Arthur. (2021). Towards an Understanding of Benign Overfitting in Neural Networks.

[100] Verschoof-van der Vaart, Wouter & Lambers, Karsten. (2019). Learning to Look at LiDAR: The Use of R-CNN in the Automated Detection of Archaeological Objects in LiDAR Data from the Netherlands. 2. 31-40. 10.5334/jcaa.32.

[101] Adaptive Methods for Nonconvex Optimization. Proceedings of the 32nd International Conference on Neural Information Processing Systems, 9815–9825. Presented at the Montréal, Canada. Red Hook, NY, USA: Curran Associates Inc.