Adapting Data-Driven Techniques to Improve Surrogate Machine Learning Model Performance

DOI:
10.1109/ACCESS.2023.3253429

Document Version
Final published version

Link to publication record in Manchester Research Explorer

Citation for published version (APA):

Published in:
IEEE Access

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y66o] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Adapting Data-Driven Techniques to Improve Surrogate Machine Learning Model Performance

HUW RHYS JONES, ANDREI C. POPESCU, YUSUF SULEHMAN, AND TINGTING MU
Department of Computer Science, The University of Manchester, M13 9PL, Manchester, U.K.
Corresponding author: Huw Rhys Jones (huw.jones@manchester.ac.uk)
This work was supported in part by the U.K. Research and Innovation, and in part EDF Energy.

ABSTRACT We demonstrate the adaption of three established methods to the field of surrogate machine learning model development. These methods are data augmentation, custom loss functions and fine-tuning of pre-trained models. Each of these methods have seen widespread use in the field of machine learning, however, here we apply them specifically to surrogate machine learning model development. The machine learning model that forms the basis behind this work was intended to surrogate a traditional engineering model used in the UK nuclear industry. Previous performance of this model has been hampered by poor performance due to limited training data. Here, we demonstrate that through a combination of additional techniques, model performance can be significantly improved. We show that each of the aforementioned techniques have utility in their own right and in combination with one another. However, we see them best applied when used to fine-tune existing models. Five pre-trained surrogate models produced prior to this study were further trained using an augmented dataset and with our custom loss function. Through the combination of all three techniques, we see an improvement of at least 38% in performance across the five models.

INDEX TERMS Advanced gas-cooled reactor, convolutional neural network, data analysis, data augmentation, data science, graphite, loss function, machine learning, nuclear, regression, supervised learning, surrogate model, pre-trained models, fine-tuning.

I. INTRODUCTION
A machine learning surrogate (MLS) is a model which aims to explain natural or mathematical phenomena which can already be explained using an existing model. Using data from the original model, machine learning techniques are used to produce an optimised MLS model. The advantages of an MLS include increased computational efficiency when generating model outputs, with the trade-off being reduced accuracy. Once developed and trained, machine learning models (including an MLS) can produce new data instances almost instantly using a standard computer, whereas generating the same information using the original model and equivalent hardware may require hours or days of computational effort. The reduction in accuracy between an MLS and an original model must be quantified on a case-by-case basis and assessed on whether it is acceptable for practical use.

Previous research works have dealt with the production of MLS in areas such as material properties prediction [1] and [2]. A recent work [3] focuses on seismic analysis for nuclear graphite cores, with the MLS model from this research work forming the basis of this research work. In the aforementioned studies, a strong focus on neural networks [4] is seen, including convolutional neural networks (CNNs).

Despite the motivation for the production of MLS models being to reduce the need for expensive production of data, a large amount of this data is required to train such a model. A machine learning model trained on an insufficient number of data instances may result in overfitting [5]. Some techniques were employed in the aforementioned paper, including randomised layer dropout [10], to counteract the effects of overfitting.
In summary, we have three key research questions to be investigated and answered by this work:

1) Can data augmentation can be applied to problems such as machine learning surrogates?
2) Can a custom loss function can be developed specifically for the task at hand?
3) Can pre-trained machine learning models be fine-tuned using the two aforementioned methods?

To this end, experiments will be performed that utilise each of these methods in turn to measure their effectiveness. In addition, we will combine all three approaches to determine if we can aggregate the benefits of each.

II. BACKGROUND
A. ADVANCED GAS-COOLED REACTORS AND THE PARMEC MODEL

The advanced gas-cooled reactor (AGR) is the most common design of nuclear power station within the UK. The reactor core of the AGR is composed of stacked graphite bricks of two individual types: (i) fuel bricks, which house the fuel assemblies which power the fission reaction and (ii) interstitial bricks, which provide structural support and importantly allow the insertion of control rods for safety purposes. The seismic response of the interstitial bricks is of paramount interest to safety analysts, as it determines the ability to safely shutdown the reactor during an severe event. As viewed from above, the two types of graphite brick are alternately arranged as per Figure 2.

The computational model Parmec [14] is the underlying model which was the subject of a MLS model in [3] and the same problem and base dataset is considered in this work. Parmec is employed to simulate the seismic response of the graphite core within the AGR. This model consists of a simplified 3-dimensional representation of the AGR graphite core, including the positional arrangement of the graphite bricks and other components. Parmec can be used to simulate a range of different seismic scenarios with the resulting component translation, rotation etc. being calculated by the model.

Due to years of exposure to high temperatures and irradiation, some of the fuel bricks within the reactor are cracking, causing them to break into two pieces. The presence and configuration of these cracks has an impact on the reaction of the core to seismic loading. It is possible that up to 40% of the fuel bricks will eventually crack, although it is difficult to determine or predict where and when cracks will occur.

The main input variable to the Parmec model is a pattern of cracked fuel bricks within the graphite core. As we do not know the where the cracks will occur, standard practice is to generate random configurations of cracked bricks within the core. As there are 1998 fuel bricks in the AGR core and we are assuming 40% of them will crack, this tool outputs a representation of the position of 795 cracked bricks.

Based on the input of a pattern of cracked bricks within the core, the Parmec software simulates an earthquake and...
generates a time-history of the earthquake response for all of the thousands of components within the core. The software outputs files representing the movement of the core for each of a set number of time frames. For a worst-case scenario earthquake, this represents approximately 6.8 million outputs, representing displacements in every direction for all bricks at all time frames.

To summarise the process of generation of a single instance of the Parmec model:
1) A random pattern of cracked fuel bricks is generated using an industry standard tool.
2) The cracked brick pattern is then fed into the Parmec model, which then proceeds to simulate an earthquake.
3) Output files are generated at a set number of time frames, each representing a point during the earthquake. It continues to do this until the earthquake simulation ends.

The aforementioned process is repeated until a satisfactory sample size has been produced. There are over 10^{2500} possible permutations of crack configuration, assuming 40% cracking. With each configuration requiring around 2 hours to compute the seismic response via Parmec, it is clearly impractical to generate data for even a small percentage of them. Instead, industry practice is to generate random configurations of cracks, passing each through Parmec in order to build up a stochastic distribution of the seismic response.

The relationship between crack configuration and seismic response of core components is complex, hence the Parmec model consists of many thousands of parameters and equations. On a low level, the interaction between core components during an earthquake is well understood - hence the ability to produce a model such as Parmec. On a high level, the overall trend involving a statistical understanding of the problem space is less well understood. The ability to generate equivalent data using fewer computational resources (i.e. bypassing the need for Parmec) and to expose relationships within the data is a key motivation behind the application of machine learning surrogate to this problem.

### B. PREVIOUS MACHINE LEARNING SURROGATE MODEL OF PARMEC

In previous machine learning assessments of AGR graphite core seismic analysis [3], each crack configuration is considered an individual data instance, with the pattern of cracked bricks being the input features and the response of core components to the earthquake being the output labels. For the sake of simplicity and focus, the MLS model was trained to predict the earthquake response for a single interstitial brick at a single time frame - see Figure 2.

To outline the features of the MLS, each instance has an input size of 1988 with this being the number of fuel bricks within the AGR graphite. This input was arranged into a 3D tensor which retains physical positional relationships within the actual AGR graphite core (Figure 3). Each element is either a 1, -1 or 0 representing a cracked brick, uncracked brick or ‘empty’ position. The 3-dimensional encoding of the input features also allows the dataset to be used with a convolutional neural network [15] which was found to be the best performing type of machine learning model.

To outline the labels of the MLS, each instance has a single output representing the displacement of a single core interstitial brick. This output is a measure of displacement in the horizontal direction at the time during the earthquake when seismic acceleration is at its peak. The single interstitial brick chosen is the central brick on the upper most core level (Figure 2). This output is a scalar value which is normalised to the range 0 to 1, with 0 representing the most extreme easterly displacement in the dataset and 1 representing the equivalent in the westerly direction.

For the aforementioned study, a dataset of approximately 8300 instances was created using the random crack pattern generator and the Parmec software, simulating a worst-case scenario earthquake. Each instance represents an input/output pair, as outlined above. Out of these instances, 6300 (75%) were used for training with the remaining 2000 samples retained for testing. This dataset will be imported for the study discussed in this research paper, with the training set being the basic dataset used to optimise models discussed in section IV and the testing set being used to evaluate models in section V.

Note that as the MLS produced in [3] is a regression, rather than classification model. Classification is in contrast to regression problems, with the later producing a continuous output variable, rather than discrete value as per the earlier.

### C. DATA AUGMENTATION

Data augmentation is frequently employed in classification problems within the field of machine learning [16], where the model predicts a discrete category for each dataset instance. A classic example of classification is in computer vision, where a 2D or 3D tensor representing an image is used to predict a category that is depicted. For example, models trained on the ImageNet dataset [17], which contains millions of images each representing one of 1000 discrete classifications, attempt to categorise the image depicted in an instance it is presented with.

When dealing with problems such as ImageNet classification, performance is constrained by the size of the dataset. Model performance tends to improve with a larger number of training examples as the model has more information to learn from. Conversely, a dataset of insufficient size tends to cause the phenomenon known as overfitting [5], where the model optimises too closely to the training data, including any noise or unrelated variability. With publicly available datasets being of a standard size, three options are available:

1) **Manually generate more data** In the case of image classification, this may mean obtaining more images and labelling them according to what class of object is depicted in them. This is often a time and labour intensive process.

2) **Model refinement** The model used to make predictions from data can be refined to improve performance.

In the case of the AGR graphite core, data augmentation can be used to augment the training set with additional crack patterns that are not represented in the original dataset. This can help to improve the generalisation of the model, allowing it to handle a wider range of crack configurations and seismic responses.

By generating additional crack patterns and simulating the seismic response for each, the model can be trained on a larger and more varied dataset. This can help to improve the accuracy and robustness of the model, allowing it to make more accurate predictions for real-world scenarios.

**Figure 3**: Example of a crack pattern in an AGR graphite core, generated using the random crack pattern generator.

**Figure 4**: Visualization of the 3-dimensional encoding of the input features for a single instance of the MLS.

**Figure 5**: Graph showing the performance of the MLS model on the training and testing datasets, with the best performing type of machine learning model indicated.
H. Rhys Jones et al.: Adapting Data-Driven Techniques to Improve Surrogate Machine Learning Model Performance

FIGURE 2. A top down diagram of the AGR graphite core Parmec model. Bricks are arranged into channels of two different types: fuel (blue) and interstitial (grey). Both types of channel are the same height, with fuel bricks being stacked seven high and the shorter interstitial bricks being stacked 12 high. The cracking status of all 1988 fuel bricks is included in the input features (whether the brick is cracked or not) of the surrogate machine learning model. For the output labels, only the earthquake response of the upper most interstitial brick (orange) is predicted by the surrogate machine learning model.

FIGURE 3. Visualisation of a 3-dimensional feature encoding. This example represents a single instance with each data-point representing a fuel brick. Yellow and black data points represent uncracked and cracked bricks, respectively.

This may include refining the selection of model metrics, parameters, architecture and randomised nodal dropout [10]. This is again a resource intensive process and is likely to have diminishing returns on effort invested as the model reaches the maximum possible optimisation.

3) Data augmentation Using data manipulation techniques to generate more data. A comparatively cheap process compared to other methods.

Figure 4 shows an example of a data augmentation process on a single image instance. The image on the left of this figure would correctly be classified as a bird. This image can be used to generate five additional instances of the same class: three by rotation and two by mirroring. By applying this process to all images within a dataset, the number of available training instances can be multiplied by a factor of six.

From the example, it can be seen that data augmentation techniques are highly suited to problems where the data is structured as a 2D or 3D tensor (such as greyscale or colour images, respectively). By extension, data augmentation is highly effective for applications in which localised or spatial patterns are of importance, for example where CNNs are employed. It should be noted at this point that the research work that we attempt to build on in this research work employs 3D encoding of data as well as CNNs.

In the field of machine learning regression, where a continuous variable is predicted (in contrast to classification discussed above), several studies have looked at using data augmentation to improve model performance [6], [7], [8]. In [6] it is noted that the focus in research concerning data augmentation has been heavily centred around classification, mainly concerning image analysis, with considerably less looking at regression problems. All three of the aforementioned research works use augmentation techniques such as adding noise to labels and use of generative networks [9]. The application of image manipulation techniques and exploitation of symmetry has not yet been attempted within the field of machine learning regression.

D. CUSTOM LOSS FUNCTION

One of the most important model metrics to be selected during machine learning model development is the loss function. This function is used during training to calculate the difference between the ground truth and model prediction (known as the loss rate). Further, the derivative of the loss rate is used to iteratively update the model weights during the training and optimisation process.

Typically, one of only a few loss functions will be selected for regression model training. A common selection is the mean squared error [18] or one of its derivatives such as root mean squared error. Other options include mean absolute error [19] and Huber loss [20] which was found to be optimal in the preceding research work on this topic [3].

An issue encountered in the aforementioned preceding research was that the data was not evenly distributed throughout the data space. The output data generated by the Parmec model tends to be distributed around a central modal value, with increasingly fewer examples towards the extremes of the data space. This in turn results in a model which tends to over-predict values in the lower part of the data space, and under-predict those in the upper part (Figure 5).

This problem can be compared with the issue of class imbalance encountered in the field of machine learning classification [21]. Much literature has been written on the subject of correcting for data imbalance in classification, with a recent work [22] using a weighted loss function.

Conversely, for regression based problems, research attention has been scarce by comparison. Some recent works [23], [24] note the lack of research on imbalanced regression have proposed solutions to problems caused by gaps or rarefactions in the data space. The solutions proposed involve the application of label distribution smoothing, where artificial data is generated to fill gaps, or undersampling,
FIGURE 4. An example of image manipulation techniques to perform data augmentation. The base instance of an image depicting a bird is shown in image (a). The additional images show examples of two types of augmentation. Images (b), (c) and (d) show image (a) rotated by 90, 180 and 270 degrees, respectively. Similarly, images (e) and (f) show image (a) reflected about the vertical and horizontal axis, respectively. Despite being manipulated in this way, each image still effectively depicts an example of a bird and can be treated as such in the training of a machine learning model. Data augmentation can be used to expand a dataset without labelling additional examples, potentially improving model performance and reducing overfitting.

where instances are removed from regions where the dataset is highly biased.

Whereas the data distribution in the aforementioned research papers contain discontinuities and irregular gaps, our dataset follows a regular pattern (See bottom image of Figure 5). This makes the methods explored previously in this area potentially unsuitable for the research problem at hand. If we wish to counteract the data imbalance problem in the research at hand, a bespoke custom loss function will have to be developed.

E. FINE TUNING OF PRE-TRAINED MODELS

Often in machine learning, models are trained from scratch, a process that consumes significant resources in terms of time and computation to achieve optimal performance. By fine-tuning models that were pre-trained for similar purposes, we can make the task of model development for a certain problem more efficient and less resource intensive.

In [3], an optimal model architecture was developed for the purpose of surrogation of a nuclear engineering model. Using this architecture and the training data, the parameters of the model were optimised from random starting weights. As training started from random weights, the process of model optimisation was repeated multiple times. The best performing models from this study, along with their pre-trained weights, can be transferred to this study as a starting point for exploitation of the methods described in subsections II-C & II-D.

The process of fine-tuning pre-trained models with an expanded database has precedent in existing research works [25]. It can be compared with the closely related field of transfer learning [26].

III. PREPARATION

A. IMAGE MANIPULATION TECHNIQUES

Recall from Figure 3 that the input features for this research problem is a 3D tensor representing the position of cracked fuel bricks within a given instance. This tensor is comparable to that of a colour image such as the one shown in Figure 4. Being a tensor of a similar encoding to that in the aforementioned figure, the same image manipulation techniques can also be applied to the tensor for this research problem.

Recall also from subsection II-B that the output labels for this research problem are normalised continuous values representing displacement in the central brick of the core. Note also from Figure 2 that the overall data space from which the output variables are extracted is planar and can be expressed as a 2D tensor. For a given instance i.e. input/output pair, we can apply any of the rotational or mirroring techniques shown in Figure 4. Note that if we rotate or mirror about the vertical axis at the centre-point of the core, the encoding order of the input features will change, but the output label will not (as it represents the top brick in the central channel). Hence, data augmentation for the dataset in this study will create new instances with restructured feature tensors, but the same output label value - see Figure 6.

As mentioned in subsection II-C, the motivation behind using image manipulation techniques to create an augmented dataset is based on the conjecture that symmetries do exist
in the Parmec data space. What is the justification behind the belief that rotational and symmetric manipulations of Parmec inputs would yield similarly transformed outputs? There are three lines of evidence that can be used to support the validity of this process:

1) **AGR Design:** The design of the AGR (and the Parmec model that is based on it) contains four-fold symmetry [27]. This effectively means that each quarter of the AGR (and Parmec) model is a rotation or mirror of the others. Therefore, we could hypothesise that mirroring and rotating the randomly generated crack patterns as exampled in Figure 6 would yield the equivalent transformation in the output tensor.

2) **Dataset Observation:** Observe Figure 7 which shows the average output value for each brick in the top layer of the Parmec model. It can be seen that there is a symmetry across both the vertical and horizontal centre-lines. This pattern is of the same four-fold symmetry as mentioned in the first bullet-point.

3) **Augmented Equivalent Data:** Through the Parmec software package, we have the benefit of creating ground truth equivalents of augmented data. For a given ground truth example, we can manipulate the inputs according to one of the transformations shown in Figure 6 and then feed them through the Parmec model. Then we can compare the outputs generated by Parmec and our non-Parmec data augmentation technique.

Despite symmetry existing in the AGR and the Parmec model as mentioned in bullet point 1, the inputs are generated randomly, meaning that the symmetry is not exploited by default. If rotation (90, 180 and 270 degrees) and mirroring (vertical and horizontal) is applied to all examples in our dataset, the available number of instances can be multiplied by a factor of six. Therefore, should all proposed methods of data augmentation be fully exploited, it would be possible to obtain a total of 37800 samples from the original training set of 6300 instances (see subsection II-B). Just as each of the images produced in Figure 5 through augmentation can be considered new individual training examples in a image classification application, each of the additional 31500 samples generated through the processes exampled in Figure 6 can be considered new instances. This is because the input configurations are generated randomly and it is very unlikely such symmetries would occur by chance (recall from subsection II-A that there are $10^{2500}$ possible configurations).

It should be noted that bullet point 1 is an incomplete justification, as the simulated earthquake always impacts the model on one particular point on its periphery i.e. the Parmec model is not fully symmetric as boundary conditions are different on one side. In order to get a preliminary evaluation of the veracity of this augmentation technique, we performed the process described in bullet point 3 for two base instances, applying each of the five manipulation techniques on the inputs and then using these to generate labelled examples using Parmec. Simultaneously, we apply all five of our non-Parmec data augmentation techniques to the inputs and outputs of both examples. Comparing the outputs of both techniques, we notice agreement when applying rotation by 180 degrees and when mirroring about the horizontal axis.

The validity of the augmentation technique discussed here will ultimately be tested through an experimental machine.
FIGURE 6. An example of image manipulation techniques applied to Parmec data to facilitate data augmentation. i: a slice from the randomly generated feature inputs for an example instance: the dark blue spots represent intact fuel bricks, with the yellow spots being cracked bricks. This image is the equivalent of image (a) from Figure 4, i.e. it is an original, unaltered instance. Note also the position of the interstitial brick at the centre of the core (O) which is the source of the output labels (see Figure 2) ii: the same example instance as shown in (i), but it has been mirrored about the vertical centre-line. This image is the equivalent of image (e) from Figure 4. Note that the position of the central brick (O) is the same as it resides on the line about which the mirroring operation took place. iii: again, the same example as (i), but this time it has been rotated by 90 degrees - the equivalent of image (b) from Figure 4. In each case, the output label (O) effectively remains the same, as it represents the central brick of the core, about which the rotation or mirroring is performed.

FIGURE 7. Parmec outputs for all bricks in the top level of the core. The colour of each brick data-point represents the average across the entire dataset (blue to red represents low to high, respectively). Note the symmetry about the vertical and horizontal centre-lines.

learning process. We can train two machine learning models of the exact same architecture and parameters: one with a dataset augmented with and one with the base dataset. A separate testing set of only non-augmented instances will be retained for testing both models which both models can best tested against. The performance of each image manipulation method can be evaluated in this way.

B. WEIGHTED LOSS FUNCTION

In [3], the effectiveness of using three alternative loss functions were compared. These were the mean squared error (MSE), mean absolute error (MAE) and the Huber loss [20]. A model trained using the Huber loss function was found to produce the best performance. However, the other two loss functions produced a similar, albeit poorer, performance. Each of the three loss functions (Figure 8) each have their own strengths and weaknesses, with MSE heavily weighting outlying values, MAE proportionately weighting outliers and Huber being somewhere in between.

As mentioned in subsection II-D, our base dataset is highly centred around a central value with a double tailed distribution which is particularly elongated at the upper end. The fact that the dataset has important regions in both the centre and the tails may explain why the Huber loss function, which behaves as a compromise between MSE and MAE, provided the best performance.

Regardless of the loss function used, the resulting model is biased towards the central region, resulting in difficulty predicting at the upper and lower extremes (Figure 5).

We propose a loss function tailored to this dataset which applies an adjustment factor that is a function of model prediction distance from a central value.

$$\text{Loss} = \frac{\alpha^2}{n\beta^2} \sum_{i=1}^{n} Z_i^2 (y_i - \tilde{y}_i)^2. \quad (1)$$

$$Z_i = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(\tilde{y}_i - \mu)^2}{2\sigma^2} \right). \quad (2)$$

$$\beta := \max\{Z_i : i = 1, 2, \ldots, n\}. \quad (3)$$

As can be seen from Eq. (1), the regular loss, which takes the mean of the square difference between the ground truth $y_i$ and the model prediction $\tilde{y}_i$, is adjusted by the factor $Z_i$, as given by Eq. (2) and is calculated for each instance $i$. This term is based on the probability density function which in this case describes a Gaussian distribution. This Gaussian
distribution is fit to that of our data distribution (Figure 9) by use of the mode $\mu$ and standard deviation $\sigma$. To normalise relative to the maximum value across the dataset, the function is scaled through division by the square of $\beta$, which denotes the maximum such $Z_i$. Also included is the magnitude coefficient $\alpha$ which will be optimised experimentally.

This function results in a strong adjustment for predictions made near the mode, quickly dropping off to one as we move away in either direction. This adjusted loss function penalises predictions made near the region where there is a large concentration of training data and instead pushes it towards the extremes.

This loss function was designed with the intention of countering the bias in the data distribution. It was developed by an experimental process of adding and refining various terms of the equation. This experimental process began with defining a basic loss function inspired by the Gaussian distribution, training a model using this loss function and then evaluating using the testing set. Through a process of directed trial and error, we were able to obtain the function defined by Eq. (1) and Eq. (2).

C. PRE-TRAINED MODEL TRANSFER

In [3], multiple machine learning models were developed and trained on the dataset. In the best performing case, a convolutional neural network was utilised with a refined architecture. Optimal performance was achieved when including inputs representing the cracking status of the top 3 levels of the AGR core. The model was used to predict displacements in a single brick on the top level of the core at a single time point during an earthquake, which we will keep as the target model prediction in this research work also. The background to these models as well as the dataset used to train them is given in subsection II-B.

Out of all of the models produced as part of [3], the best performing five models were obtained and imported into this research study, as well as their optimised weights. The architecture and general parameters of each of these models is the same, with the only difference between each being the hyper-parameters optimised during training.

We evaluated each of these models, numbered M1 to M5, against the testing dataset with the results shown in Table 1. These models will form the basis of pre-trained model fine-tuning experiments through refinement using the methods described in subsections III-A & III-B. As mentioned in subsection II-E, fine-tuning pre-trained models can reduce the time and computational resources required compared to starting from randomised weights. Transferring models and weights in this case will not only reduce resource requirements but also be interesting from a research perspective.

IV. EXPERIMENTAL EVALUATION PROCESS

A. AUGMENTATION

The experimental evaluation of augmentation began by evaluating the effectiveness of each image manipulation technique mentioned in subsection II-C individually. To do this, we augmented the base dataset (obtained from a previous study as mentioned in subsection II-B) using each of the five...
image manipulation techniques. This resulted in six available datasets: three using rotation, two using mirroring and the original unaugmented set. A summary of the datasets is shown in Table 2.

For the purposes of comparison, a model design was selected which is highly simplified compared to that utilised in [3]. This simplification was made in order to reduce computational demands and to allow obtaining of results quickly. As our intention at this point is to compare the effectiveness of using different datasets and not overall optimisation, a simplified model is acceptable for this purpose.

The neural network architecture that was selected for this part of the research can be seen in Figure 10. Between the input and output layers, there is one convolutional layer followed by two dense layers. Activation functions and the use of dropout was utilised based on previous experience. The Huber loss function was used for back propagation and optimisation during training.

Six experiments were performed which involved training the model shown in Figure 10 individually with the datasets listed in Table 3. A 10% sample of the unaugmented dataset (D0 from Table 2) was retained for validation and the model was trained until reaching convergence in terms of validation loss. For each experiment, the training process was repeated 32 times, each time initialising with randomised starting weights and dropout nodes. Each model was then evaluated using a separate testing dataset with the results given in subsection V-A1.

The datasets from the first phase were combined incrementally in the order of effectiveness as per Table 4. Again, each experiment was repeated 32 times. The results of this experiment are given in subsection V-A2.

### B. CUSTOM LOSS FUNCTION

The purpose of this experiment is to evaluate the bespoke loss function defined in subsection III-B and shown in Eq. (1) & Eq. (2). The simplified architecture used in the previous section and shown in Figure 10 was again used in the experiments in this section.

Initially, the unaugmented dataset (D0 from Table 2) was used for training (E3.1). We then investigated the use of our custom loss function in combination with the augmented datasets as detailed in subsection IV-A. E3.2 uses a training set which includes all augmented datasets as per E2.4 and the custom loss function used in E3.1. This experiment is summarised in Table 5.
Next, we adjust the alpha coefficient in the remaining parts of this experiment, with α having a value of unity in experiment 3. E4.1, E4.2 & E4.3 each increment α by unity in turn, with a summary given in Table 6.

As per experiments 1 & 2, we will repeat each experiment 32 times to account for the stochastic nature of weight initialisation. The results are reported in subsection V-B.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Dataset</th>
<th>Loss Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4.1</td>
<td>Base dataset and all augmentation datasets</td>
<td>As per E3.2 with α of 2</td>
</tr>
<tr>
<td>E4.2</td>
<td>Base dataset and all augmentation datasets</td>
<td>As per E3.2 with α of 3</td>
</tr>
<tr>
<td>E4.3</td>
<td>Base dataset and all augmentation datasets</td>
<td>As per E3.2 with α of 4</td>
</tr>
</tbody>
</table>

### C. PRE-TRAINED MODEL FINE-TUNING

Subsection III-C discusses five pre-trained models obtained using the methodology of a previous research work in this field [3]. The performance of these models against a testing dataset is summarised in Table 1.

The intention of this experiment is to further train these models using the methods detailed in the previous two subsections (IV-A & IV-B). We begin by further training models M1 to M5 using our dataset enlarged by all augmentation methods as per E2.4. We then combine both augmentation and the use of our custom loss function as defined in Eq. 1 & Eq. 2. Finally, we perform the same further training of the transferred models using the conditions of the previous experiment but with an α coefficient of two.

This experiment is summarised in Table 7. We repeat the training process six times for each part of the experiment as opposed to the 32 times performed in earlier experiments. This is as only the dropout nodes are randomly selected and we are not initialising the weights. The results of this experiment will be presented in subsection V-C.

### V. RESULTS

This section outlines several experiments used to test the hypotheses described in previous sections as well as a summary of the results. The results are presented in tables and graphical figures.

In the tables, the loss between the predictions of the model and the ground truth from the test dataset are reported in mean squared error, giving an equivalent comparator regardless of the loss function used to train the model. We report the optimal (lowest) and mean loss achieved for all repeats of the experiment.

In the figures, the performance of the model is represented in four sub-figures, each plotting the test set predictions of the model (vertical axis) against ground truth labels (horizontal axis). In the top sub-figure, the predictions are delineated by lines representing a 10 and 20 percentage point margin, with predictions that fall into each margin highlighted by colour. In the lower three sub-figures, the predictions and ground truth are delineated by a margin at a certain point along the data continuum. These allow predictions to be classified into true or false, depending on if they are predicted in the correct region. The delineations should be compared with those in the lower sub-figure of Figure 5 where they are indicated.

#### 1) INDIVIDUAL AUGMENTATION

The results of this experiment are summarised in Table 8. The experiment which produced the optimal performance (i.e. lowest test loss) out of 32 repeats was E1.4 with 7.10E-3. This experiment also had the lowest mean loss (8.10E-3). All experiments which used an augmented dataset (E1.1 to E1.5) had a lower optimal performance than when using the unaugmented set only (E1.0). The mean values for all augmented experiments excluding E1.5 are below that of E1.0.

#### 2) MULTIPLE AUGMENTATION

The results of this experiment are summarised in Table 9. All experiments which combine augmented datasets (E2.1 to E2.4) have improved optimal and mean test performances over that of the unaugmented experiment (E1.0). As datasets are incrementally added, both the optimal and mean test performances see improvement.

In Figure 11 we see a visualisation and comparison of the test performance of the optimal model from E1.0 & E2.4.
TABLE 8. Results Summary of Experiment 1. Experiment 1.0 includes only the unaugmented dataset (D0). Experiments 1.1 to 1.5 combine datasets D0 and one of the augmented datasets (D1 to D5). The models produced during each experiment were tested against the testing set with the results reported in mean squared error (MSE). Out of the 32 models trained for each experiment, the optimal result (i.e. the lowest) is reported as well as the overall mean.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1.0</td>
<td>1.06E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E1.1</td>
<td>7.70E-3</td>
<td>8.60E-3</td>
</tr>
<tr>
<td>E1.2</td>
<td>2.02E-3</td>
<td>8.20E-3</td>
</tr>
<tr>
<td>E1.3</td>
<td>7.90E-3</td>
<td>9.00E-3</td>
</tr>
<tr>
<td>E1.4</td>
<td>7.10E-3</td>
<td>8.10E-3</td>
</tr>
<tr>
<td>E1.5</td>
<td>1.02E-2</td>
<td>1.12E-2</td>
</tr>
</tbody>
</table>

TABLE 9. Results Summary of Experiment 2. Each experiment combines the base unaugmented dataset (D0) with two or more augmented datasets as shown in Table 4. Experiment E1.10 is included for comparison and context. The models produced during each experiment were tested against the testing set with the results reported in mean squared error (MSE). Out of the 32 models trained for each experiment, the optimal result (i.e. the lowest) is reported as well as the overall mean.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1.0</td>
<td>1.06E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E1.1</td>
<td>7.30E-3</td>
<td>8.40E-3</td>
</tr>
<tr>
<td>E1.2</td>
<td>6.60E-3</td>
<td>7.70E-3</td>
</tr>
<tr>
<td>E1.3</td>
<td>6.60E-3</td>
<td>7.70E-3</td>
</tr>
<tr>
<td>E1.4</td>
<td>6.50E-3</td>
<td>7.50E-3</td>
</tr>
</tbody>
</table>

B. CUSTOM LOSS FUNCTION

An experimental approach to the development of a loss function customised to the needs of the data problem at hand was discussed in subsection IV-B with a summary of proposed experiments shown in Tables 5 & 6. The results are summarised in Tables 10 & 11 with a visual summary of E4.1 shown in Figure 12.

Using our custom loss function (E3.1) produces a model with similar mean squared error to that of the baseline case (E1.0) with little variation. The addition of all augmented datasets to the training set (E3.2) yields similar model performance to that of experiment E2.4 which uses the baseline loss function. Looking next at experiment 4, it appears that increasing the $\alpha$ coefficient above unity appears to increase mean squared error.

C. PRE-TRAINED MODEL FINE-TUNING

We discussed in subsection IV-C that our five existing models with their pre-trained weights (M1 to M5) were further trained using methods developed in this research work. The performance of these models further trained on our augmented training set is shown in Table 12. Comparing these results with the original model performance (Table 1) it can be seen that a significant improvement has been achieved.

Looking next at pre-trained models fine-tuned with the aforementioned augmented dataset and also using our custom loss function, very similar performance results are seen (Table 13). It is possible that the similarity between the results of experiments 5.1 and 5.2 is due to the fact that the optimal model performance has been achieved i.e. the model has generalised to the data in both of these experiments as closely as is theoretically possible.

Finally, performing the same process as the aforementioned experiment but with the $\alpha$ set to two, we see the performance of the models summarised in Table 14 with a visual comparison with the original M3 model seen in Figure 13.
TABLE 10. Results Summary of Experiment 3. E3.1 involved training the model using the custom loss function as described in subsection III-B. E3.2 involves again using our custom loss function, but also involves training with the full augmented dataset as per E2.4. The results of E1.0, which involved training with a standard Huber loss function, are shown for context. The models produced during each experiment were tested against the testing set with the results reported in mean squared error (MSE). Out of the 32 models trained for each experiment, the optimal result (i.e. the lowest) is reported as well as the overall mean.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1.0</td>
<td>1.06E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E3.1</td>
<td>1.08E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E3.2</td>
<td>6.54E-3</td>
<td>7.50E-3</td>
</tr>
</tbody>
</table>

TABLE 11. Results Summary of Experiment 4. Each part of this experiment involved training on the full augmented dataset as per E2.4 and the loss function from E3.1. In this experiment, we increment the value of alpha as described in subsection IV-B. The models produced during each experiment were tested against the testing set with the results reported in mean squared error (MSE). Out of the 32 models trained for each experiment, the optimal result (i.e. the lowest) is reported as well as the overall mean.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>E4.1</td>
<td>7.31E-3</td>
<td>7.88E-3</td>
</tr>
<tr>
<td>E4.2</td>
<td>8.14E-3</td>
<td>9.65E-3</td>
</tr>
<tr>
<td>E4.3</td>
<td>4.17E-2</td>
<td>4.41E-2</td>
</tr>
</tbody>
</table>

TABLE 12. Results Summary of Experiment 5.1. This experiment involves the further training of the pre-trained models transferred from a previous study with an training set enlarged by data augmentation. The model performance from this experiment is compared with that obtained previously in [3].

<table>
<thead>
<tr>
<th>Model</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
<th>Original Test Performance</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>5.70E-3</td>
<td>6.20E-3</td>
<td>9.28E-3</td>
<td>38.5</td>
</tr>
<tr>
<td>M2</td>
<td>5.70E-3</td>
<td>6.20E-3</td>
<td>9.25E-3</td>
<td>38.0</td>
</tr>
<tr>
<td>M3</td>
<td>5.70E-3</td>
<td>6.10E-3</td>
<td>9.22E-3</td>
<td>38.0</td>
</tr>
<tr>
<td>M4</td>
<td>5.70E-3</td>
<td>6.20E-3</td>
<td>9.68E-3</td>
<td>41.0</td>
</tr>
<tr>
<td>M5</td>
<td>5.80E-3</td>
<td>6.20E-3</td>
<td>9.48E-3</td>
<td>39.0</td>
</tr>
</tbody>
</table>

Comparing E5.1 & E5.2 with E5.3, it initially appears to perform slightly worse in terms of MSE test loss. From the comparison figure, multiple improvements over the original model performance can be seen, including a closer fit with fewer examples falling outside of both the 10 and 20 point margins. Also, the model from E5.3 performs considerably better than its original counterpart at the upper and lower bounds of the data space.

D. ANALYSIS AND DISCUSSION

We have performed five experiments to test the veracity of the adaption of three methods to the field of machine learning surrogates. A summary of all these experiments is given in Table 15.

FIGURE 12. Visual summary of the performance of the optimal model produced during E4.1. The simplified model architecture shown in Figure 10 is trained on an expanded dataset which includes augmentation as described in subsection III-A. The custom loss function as per Eq. (1) & Eq. (2) is used during training with an $\alpha$ of value of 2. This process was repeated 32 times and evaluated against the test set. The predictions of the model with the lowest test loss are plotted against ground truth values and presented in four ways. In the top image, bounding lines are placed parallel to perfect prediction/ground truth agreement line (black), demarcating a 10 & 20 percentage point margin. The lower three images split the data space into segments and quantify the proportion of samples which are correctly placed.
TABLE 13. Results Summary of Experiment 5.2. This experiment involves the further training of the pre-trained models transferred from a previous study with an training set enlarged by data augmentation and also the use of a custom loss function. The model performance from this experiment is compared with that obtained previously in [3].

<table>
<thead>
<tr>
<th>Model</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
<th>Original Test Performance</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>5.70E-3</td>
<td>6.10E-3</td>
<td>9.28E-3</td>
<td>38.5</td>
</tr>
<tr>
<td>M2</td>
<td>5.70E-3</td>
<td>6.10E-3</td>
<td>9.28E-3</td>
<td>38.0</td>
</tr>
<tr>
<td>M3</td>
<td>5.70E-3</td>
<td>6.20E-3</td>
<td>9.28E-3</td>
<td>38.0</td>
</tr>
<tr>
<td>M4</td>
<td>5.70E-3</td>
<td>6.30E-3</td>
<td>9.68E-3</td>
<td>41.0</td>
</tr>
<tr>
<td>M5</td>
<td>5.80E-3</td>
<td>6.20E-3</td>
<td>9.48E-3</td>
<td>39.0</td>
</tr>
</tbody>
</table>

TABLE 14. Results Summary of Experiment 5.3. Like E5.2, this experiment involves the further training of the pre-trained models transferred from a previous study with an training set enlarged by data augmentation and the use of a custom loss function. However, this time a custom loss function with an $\alpha$ of two is used. The model performance from this experiment is compared with that obtained previously in [3].

<table>
<thead>
<tr>
<th>Model</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
<th>Original Test Performance</th>
<th>Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>6.00E-3</td>
<td>6.50E-3</td>
<td>9.28E-3</td>
<td>35.0</td>
</tr>
<tr>
<td>M2</td>
<td>6.20E-3</td>
<td>6.50E-3</td>
<td>9.28E-3</td>
<td>33.0</td>
</tr>
<tr>
<td>M3</td>
<td>6.40E-3</td>
<td>7.00E-3</td>
<td>9.28E-3</td>
<td>30.5</td>
</tr>
<tr>
<td>M4</td>
<td>6.40E-3</td>
<td>7.00E-3</td>
<td>9.68E-3</td>
<td>34.0</td>
</tr>
<tr>
<td>M5</td>
<td>6.40E-3</td>
<td>8.00E-3</td>
<td>9.48E-3</td>
<td>32.5</td>
</tr>
</tbody>
</table>

1) DATA AUGMENTATION

We developed five techniques for augmenting our dataset, each involving a different way of exploiting the symmetry in our underlying model. It can be seen from a comparison of the performance of models trained with any of the augmented datasets (E1.1 - E1.5) with a model trained only on the base dataset (E1.0) that each augmentation technique yields at least some benefit. The results show that the addition of some augmented datasets has more of an effect than others. For example, experiments E1.4 & E1.2 (rotation by 180 degrees and vertical mirroring) both see about a one third reduction in mean squared error in comparison to the baseline (E1.0). It is interesting that the two best performing augmentations both make adjustments about the vertical axis. Conversely, the least performing augmented dataset (D5, included in E1.5) makes its adjustment about the horizontal axis. It was noted in subsection refmanipulation that the underlying Parmec model is not truly symmetric, with the simulated earthquake impacting the model on one side. This asymmetry may be explain the differences in performance between the augmented datasets.

Experiment 2 involved in the incremental addition of the augmented datasets. Starting with the two best performing experiments from experiment 1 (180 degree rotation and vertical mirroring) we actually see a slight reduction in performance compared with applying either augmentation technique individually. However, each subsequent addition of augmentation datasets either sees a greater or equal model performance, with the inclusion of all augmentation datasets (E2.4) providing the best overall performance.

It can be seen that a model trained on all augmented datasets produces an overall better fit than when using no augmentation at all (Figure 11). Key indicators include the percentage of data within 20 points of the ground truth - 98% for the best performing model from E2.4 compared with 94% for a similar model from E1.0. The inclusion of all augmented datasets also appears to improve model prediction in the upper part of the data space - compare the second section from the top in Figure 11.

Overall, the results of experiments 1 & 2 show that the adaption of image manipulation augmentation techniques to our machine learning surrogate models is highly effective.

FIGURE 13. Visual summary and comparison of the performance of the original model M3 (Left) & M3 After E5.3 (Right). The model obtained from [3] is trained using our custom loss function Eq. (1), with an $\alpha$ value of 2. This process was repeated six times and evaluated against the test set. The predictions of the model with the lowest test loss are plotted against ground truth values and presented in four ways. In the top image, bounding lines are placed parallel to perfect prediction/ground truth agreement line (black), demarcating a 10 & 20 percentage point margin. The lower three images split the data space into segments and quantify the proportion of samples which are correctly placed.
TABLE 15. Summary of All Experiments. We have performed five experiments to determine the veracity of three machine learning methods. Experiments 1 & 2 each deal with data augmentation, with the earlier experiment applying a range of augmentation methods individually and the latter applying them in combination. In experiment 3, a machine learning model is trained using a custom loss function, with E3.1 training on the base dataset and E3.2 training on the dataset expanded through augmentation. Experiment 4 again involves training a model using a model using an augmented dataset and custom loss function, with a user defined parameter being adjusted. Experiment 5 involved fine-tuning the training of five models obtained from a previous research work. E5.1 involves fine-tuning using the dataset expanded through augmentation and using the loss function previously used (Huber). E5.2 involves fine-tuning with both the dataset expanded through augmentation and the use of our custom loss function. E5.3 uses the same method as E5.2 but involves adjust a parameter in the custom loss function. All values are in MSE loss, hence lower values represent a better performing model.

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Model</th>
<th>Dataset</th>
<th>Loss Function</th>
<th>Optimal Test Performance</th>
<th>Mean Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1.0</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset</td>
<td>Huber</td>
<td>1.06E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E1.1</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset &amp; 90 degrees rotation augmentation dataset</td>
<td>Huber</td>
<td>7.70E-3</td>
<td>8.0E-3</td>
</tr>
<tr>
<td>E1.2</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset &amp; 180 degrees rotation augmentation dataset</td>
<td>Huber</td>
<td>7.20E-3</td>
<td>8.20E-3</td>
</tr>
<tr>
<td>E1.3</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset &amp; 270 degrees rotation augmentation dataset</td>
<td>Huber</td>
<td>7.90E-3</td>
<td>9.00E-3</td>
</tr>
<tr>
<td>E1.4</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset &amp; Vertically Mirrored augmentation dataset</td>
<td>Huber</td>
<td>7.10E-3</td>
<td>8.10E-3</td>
</tr>
<tr>
<td>E1.5</td>
<td>Simplified Model (Figure 10)</td>
<td>Base Dataset &amp; Horizontally Mirrored augmentation dataset</td>
<td>Huber</td>
<td>1.02E-2</td>
<td>1.12E-2</td>
</tr>
<tr>
<td>E2.1</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset, 180 degrees rotation &amp; Vertically Mirrored augmentation datasets</td>
<td>Huber</td>
<td>7.30E-3</td>
<td>8.40E-3</td>
</tr>
<tr>
<td>E2.2</td>
<td>Simplified Model (Figure 10)</td>
<td>Datasets used in E2.1 &amp; 90 degrees rotation augmentation dataset</td>
<td>Huber</td>
<td>6.60E-3</td>
<td>7.70E-3</td>
</tr>
<tr>
<td>E2.3</td>
<td>Simplified Model (Figure 10)</td>
<td>Datasets used in E2.2 &amp; 270 degrees rotation augmentation dataset</td>
<td>Huber</td>
<td>6.60E-3</td>
<td>7.70E-3</td>
</tr>
<tr>
<td>E2.4</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>Huber</td>
<td>6.50E-3</td>
<td>7.50E-3</td>
</tr>
<tr>
<td>E3.1</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset</td>
<td>As per Eqs. (1) &amp; (2)</td>
<td>1.08E-2</td>
<td>1.11E-2</td>
</tr>
<tr>
<td>E3.2</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per Eqs. (1) &amp; (2)</td>
<td>6.54E-3</td>
<td>7.50E-3</td>
</tr>
<tr>
<td>E4.1</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per E3.2 $\alpha = 2$</td>
<td>7.31E-3</td>
<td>7.88E-3</td>
</tr>
<tr>
<td>E4.2</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per E3.2 $\alpha = 3$</td>
<td>8.14E-3</td>
<td>9.65E-3</td>
</tr>
<tr>
<td>E4.3</td>
<td>Simplified Model (Figure 10)</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per E3.2 $\alpha = 4$</td>
<td>4.17E-2</td>
<td>4.41E-2</td>
</tr>
<tr>
<td>E5.1</td>
<td>Pre-trained models from [3]</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>Huber</td>
<td>5.70E-3 (M3)</td>
<td>6.10E-3 (M3)</td>
</tr>
<tr>
<td>E5.2</td>
<td>Pre-trained models from [3]</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per Eqs. (1) &amp; (2)</td>
<td>5.70E-3 (M3)</td>
<td>6.10E-3 (M2)</td>
</tr>
<tr>
<td>E5.3</td>
<td>Pre-trained models from [3]</td>
<td>Base dataset &amp; all augmentation datasets</td>
<td>As per E5.2 $\alpha = 2$</td>
<td>6.00E-3 (M1)</td>
<td>6.50E-3 (M2)</td>
</tr>
</tbody>
</table>
We see improved model performance when comparing any of our experiments involving data augmentation (E1.1 - E2.4) with a model trained on the base dataset (E1.0). The performance of the models from experiments 1 & 2 even show improvements over the original models from the previous study - compare for example the results of E2.4 with any of the model results from Table 1. This is particularly noteworthy when considering experiments 1 & 2 involve a model that is highly simplified compared to M1 - M5.

2) CUSTOM LOSS FUNCTION
We defined a custom loss function specially designed to fit our dataset distribution as outlined by Eq. (1) & (2). We first applied this custom loss function in the training of models for experiment 3, first using the base dataset (E3.1) and then using the fully augmented dataset (E3.2).

We can compare the results of E3.1 directly to our base experiment (E1.0), with both experiments using the same base dataset and the only difference being the loss function. Making this comparison, it appears that using the custom loss function reduces optimal model performance slightly (1.08E-2 against 1.06-2) and no change in the mean performance.

We can make the equivalent comparison between E3.2 and E2.4, with both experiments using the fully augmented dataset and the only difference being the loss function. Again we see a slight performance reduction when using the custom loss function (6.54E-3 against 6.50E-3) and no change in the mean performance.

In experiment 4, we repeat the method of E3.2 but increment the value of the parameter \( \alpha \). We can see that for each increment, the model performance worsens in terms of a rising optimal and mean MSE loss.

From these measurements, it would initially appear that we have been unable to demonstrate any advantage to using our custom loss function. Has this experiment been a failure? To answer this question, we must return to the motivation behind this method as detailed in subsections II-D. We discuss the fact that our data space is concentrated in a central region with increasing rarefaction as we move away from it. Consequently, our models trained on this data performs poorly near the extremes as can be seen from the left part of Figure 11. Comparing the aforementioned figure with a visualisation of the optimal model from experiment E4.1 (\( \alpha = 2 \)), we can see significant improvements in model performance at the upper and lower extremes of the data space (Figure 12). This advantage comes at the expense of minor reductions in overall performance metrics, for example percentage of examples that are predicted outside of 20 percentage points of the ground truth (rising from 2.0% in E2.4 to 2.6% in E4.1). In addition, the trade-off appears to become more intense as the value of \( \alpha \) increases. Whether or not this trade-off is of benefit and which value of \( \alpha \) to select will depend on any practical application of the model. Nonetheless, it is likely that analysis using any such model would value good performance near the extremes as events in these regions are most likely to impact safety. Therefore, we highlight the potential of this method and carry it forward into experiment 5.

3) PRE-TRAINED MODEL FINE-TUNING
We can see from the results of experiment 5.1 (Table 12) that further training our pre-trained models with an augmented dataset results in a significant performance uplift. For example, M1 sees its test mean squared error drop from 9.28E-3 to a minimum of 5.70E-3 - a performance improvement of about 38.5%. Combining the augmented dataset with our custom loss function and further training our base models appears to have little effect beyond what we saw in E5.1 (Table 13). Repeating E5.3 with an alpha coefficient of 2 initially appears to worsen performance, with test MSE rising from 5.70E-3 to between 6E-3 and 6.40E-3. However, on inspection of the visual results of E5.3 and comparison with those of E5.1 & E5.2, we see significantly improved performance at the upper and lower boundaries (above 0.6 and below 0.2 on the normalised scale of ground truth). Hence, we present the optimal results from E5.3 in Figure 13. Some notable improvements include a halving in the number of predictions that fall outside of the 20 percentage point margin (4.0% in the original M3 against 1.8% post E5.3), an increasing percentage of samples correctly placed over the 0.6 boundary (40.9% in the original M3 against 54.8% post E5.3) and finally a similar improvement below the 0.2 boundary (16.1% against 35.1%). As touched on earlier, in the field of machine learning for nuclear energy, this represents noteworthy progress as safety decisions are likely to concern events at the extremes i.e. very high or very low values.

VI. CONCLUSION
In this research work, we demonstrate the adaption of three established approaches to the field of surrogate machine learning model development. The methods are data augmentation, custom loss functions and pre-trained model fine-tuning. Each of these approaches have seen widespread use in the field of machine learning; however, here we apply them specifically to surrogate machine learning model development.

The machine learning model that forms the basis behind this work was intended to surrogate a traditional engineering model used in the UK nuclear industry. This model was built with the intention of increasing computational efficiency over the original model it surrogated. The performance of this model was hampered by poor performance due to limited training data. Here, we demonstrate that through a combination of additional techniques, model performance can be significantly improved.

We have demonstrated the adaption of image manipulation techniques, used frequently in image classification problems, for use in regression based surrogate machine learning models. Whereas previous research regarding the use of data augmentation in regression problems had used techniques such as adding noise to labels, we exploit symmetry in the
underlying model to generate additional data. We demonstrate that using this technique to create an augmented dataset, we can improve model performance significantly over previous works, even when using a far simpler model architecture.

The second method we demonstrate the adaption of is the custom loss function. We show that a loss function designed specifically around the distribution of our data can compensate for label bias. The design and intent of the loss function designed for this study is in contrast to those designed for other research works, where they were designed for compensation for gaps or rarefactions in the data distribution.

We show that each of the aforementioned techniques have utility in their own right and in combination with one another. However, we see them best applied as part of a fine-tuning operation on existing pre-trained models. Five pre-trained surrogate models produced prior to this research were further trained with the augmented dataset and with our custom loss function. Through the combination of all three techniques, we see an improvement of at least 38% in performance across the five models.

VII. FUTURE WORK

Further work will aim to build on the methods developed in this research work. This includes refinement and fine-tuning of the custom loss function developed in this paper. In addition, we will look at other techniques that could be used to improve model performance. One area with potential application to our research problem is active learning [29]. This technique uses additional feedback within the learning process and is often employed in situations where training data is in short supply. As we have established in this research work, techniques which compensate for a lack of training data or gaps in training data can be used to improve performance. Therefore, active learning is a worthwhile area to study next.

ACKNOWLEDGMENT

The authors would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility, The University of Manchester. They would also like to thank the teams behind the open-source projects NumPy [30] and Matplotlib [31], used widely throughout this project. They would also like to acknowledge the help and support given by Kirstie Dibdib Bruce and James Andrew Macgregor for their efforts in proof reading.

REFERENCES


[6] K. Dibdib Bruce and James Andrew Macgregor for their efforts in proof reading.


HUW RHYS JONES was born in Swansea, South Wales, in 1986. He received the B.S. degree in mechanical engineering from Swansea University, U.K., in 2011, and the M.S. degree in nuclear engineering from the University of Birmingham, U.K., in 2012. He is currently pursuing the Ph.D. degree in computer science with The University of Manchester, U.K.

He worked in the nuclear sector, from 2012 to 2018. During this time, he worked in many areas of the nuclear industry including safety assessments, inspections, outage support, and the computer modeling of reactor components. He recently accepted a position working in nuclear decommissioning with The University of Manchester. He recently had a research article published in the journal of Nuclear Engineering and Design concerning the application of advanced data science techniques to the nuclear safety field.

ANDREI C. POPESCU was born in Ploieşti, Romania, in 2001. He is currently pursuing the B.Sc. degree in computer science with The University of Manchester, U.K.

YUSUF SULEHMAN was born in Manchester, U.K., in 1998. He received the master’s degree in mathematics from The University of Manchester, U.K., in 2020, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include improving the adversarial robustness of deep neural networks, with a particular focus on Lipschitz estimation for robustness certification and Lipschitz regularization as a method of adversarial defense.

TINGTING MU received the B.S. degree from the School of the Gifted Young, University of Science and Technology of China, in 2004, and the Ph.D. degree from the University of Liverpool, in 2008.

She is currently a Senior Lecturer (Associate Professor) in machine learning with The University of Manchester. She is also an Honorary Research Fellow with The University of Manchester at Harwell and a Visiting Scientist with the Science and Technology Facilities Council. Her research interests include representation learning: how to extract refined information from raw data and knowledge and encode it in representation spaces for use in pattern recognition and prediction tasks. She works on algorithm design and the theoretical understanding of a diverse range of representation learning methodologies, e.g., manifold learning, matrix factorization, and neural representation learning. Her application interests include image, text, speech data analysis, and practical AI solutions.