Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

School of Electronic Engineering

CB54: Machine Learning Algorithms for EM Wave Scattering Problems

Appendix E: Testing & Results

Anthony James McElwee

ID Number: 20211330

August 2023

MEng in Electronic and Computer Engineering

Supervised by Dr Conor Brennan

Contents

Introduction

All statistical tests were performed using the free and open-source statistical analysis programme JASP [1]. The JASP programmes and the associated CSV datasets are available on GitHub. A simple Bayesian alternative to the frequentist approach resides within the JASP files too.

Description of Training Dataset

The generation of seeded samples was conducted in batches of 1000 samples per folder due to the input configuration used in Python arising from memory exhaustion. Five thousand samples were generated.All folders were scanned using Auslogics Duplicate File Finder 10 on the PNG files to search for duplicate geometry samples. These duplicates and their associated NumPy files were moved to a separate duplicate folder and excluded from all further experimental activity. Earlier runs of the experiment had used an even more minimal geometric scenerio generator where the main scatter was anchored at the origin. After generating 49000 samples, and extremely late in the project timeline, it was realised by the student that roughly 84% of the samples were duplicates. The training, validation and testing sets were all totally overlapping and the model was immediately overfitting on every run. After correcting for this issue, the student found that between 4000 and 5000 samples were enough to develop the model before the training loss curve started to indicate possible overfitting issues. The training set was split at 80% from the entire folder with the remaining 20% used for testing. A further split of 20% from the training set was used for validation at the end of each epoch leaving 640 samples for training in each session. Due to the removal of duplicates some folders had less than 640 training samples.

E1 Model Training Commentary

The final total model fitting time for the E1 component of the Prescient2DL deep learning model was 1117.3092517852783 seconds, which is roughly twenty minutes. This does not include the initial creation of the training/validation/testing splits which take time to process from the sample data to correct tensor format. The screen command line printout of the final two training data batches is available on GitHub for reference.

E1 Field Model Training Loss Curves

It should be noted that GitHub hosts all of the loss curves plotted at the end of each epoch. Tensorboard was also used to track the weight updates at each layer, however, analysis and improvement on weight behaviours will remain in the domain of future work due to time limitations.

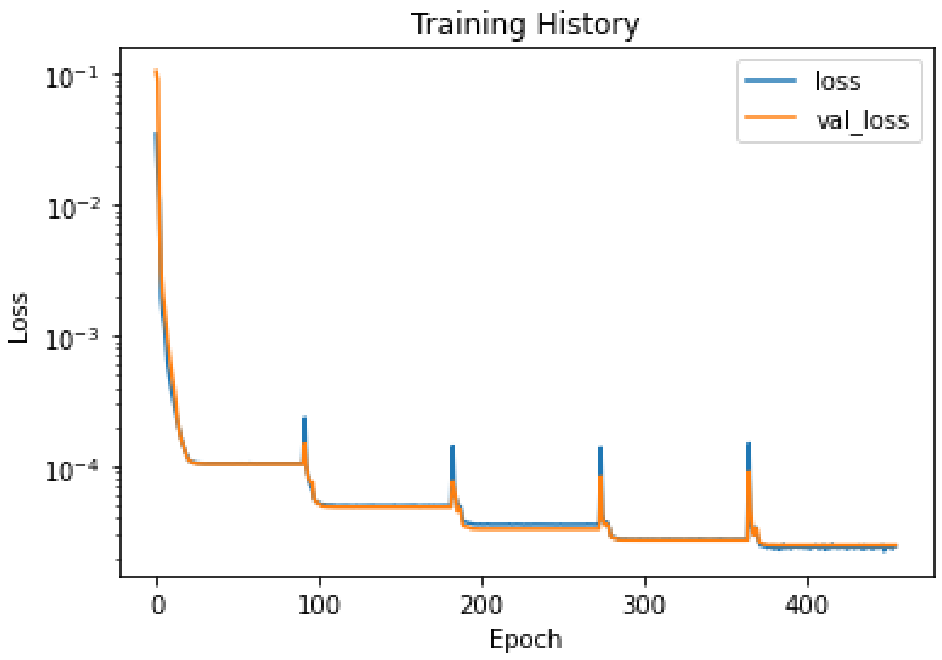

Figure 1Final Loss Curve Plot. Note the axis scale is logarithmic.

The loss curve is extremely severe when plotted on the normal axis and corresponds with loss curve plots found in the paper [2] that deals with physics deep learning applications to a laminar flow problem in the domain of fluid dynamics. Both training and validation curves track each other well In the final training session, the training curve starts to oscillate compared to the validation curve indicating possible overfitting. Although the loss curve generally follows the desired shape of a loss curve graph, albeit on a log scale, the loading spikes indicate a number of possible issues in the model design. These issues are discussed later in the appendix.

E1 Field Model Training Test Scores

For the final training dataset, the following scores were achieved:

Metric	**#**
Training Mean Squared Error Loss	2.3042e-05
Validation Mean Squared Error Loss	2.5000e-05
Final Learning Rate	1.0000e-22

Key Metric: For the final training dataset, the mean squared error test score after training was 2.628892798384186e-05. Since no cross-contamination of samples exist in each split, the validation error here is the correct indicator of model performance from the perspective of the deep learning model compared to the final solved solution provided as the prediction target. The final learning rate frequently was reduced to this order of magnitude throughout each training dataset run and this is reflected in the loss curves.

E1 Field Model Training Visual Prediction Results Absolute Component

The student collected prediction plots of a sample test case at the end of each training set loading stage to illustrate the improvement in the prediction power of the model as it was being trained. Each dataset contained roughly 640 samples of training data, depending on duplicates, after the testing and validation splits were removed. The plots are shown below in chronological order and can be found on GitHub. They illustrate the input geometry contrast values and the absolute field assembled form the two predicted real and imaginary component fields. The rapid convergence to low mean squared error loss is clearly captured in the difference plot on the bottom left corner, however, the source of truth and predicted fields really only become comparable as the model starts to overfit in the final two plots.

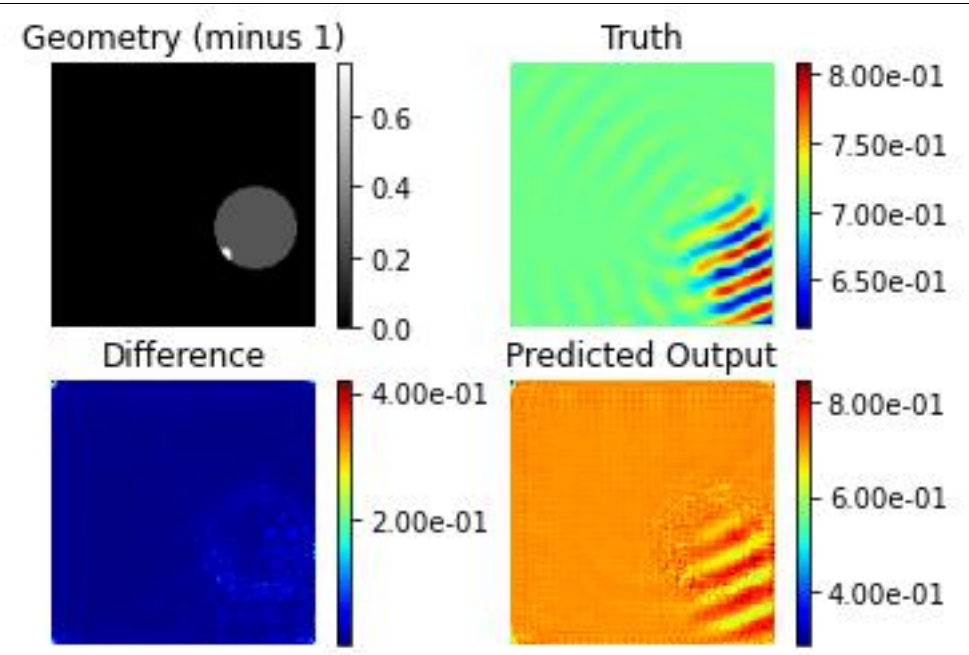

Figure 2End of first training session for E1.

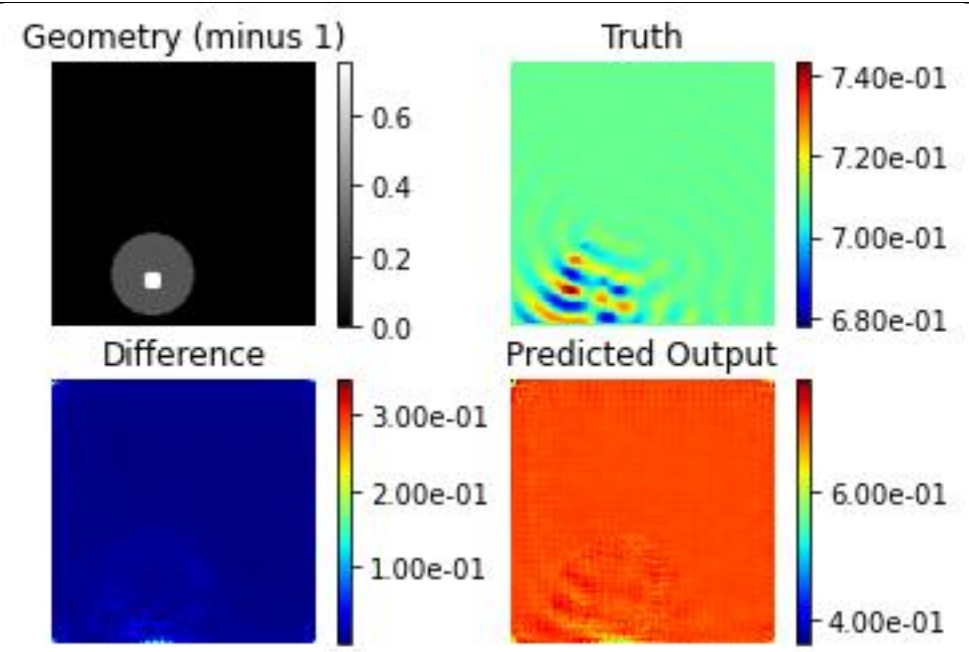

Figure 3. End of second training session for E1.

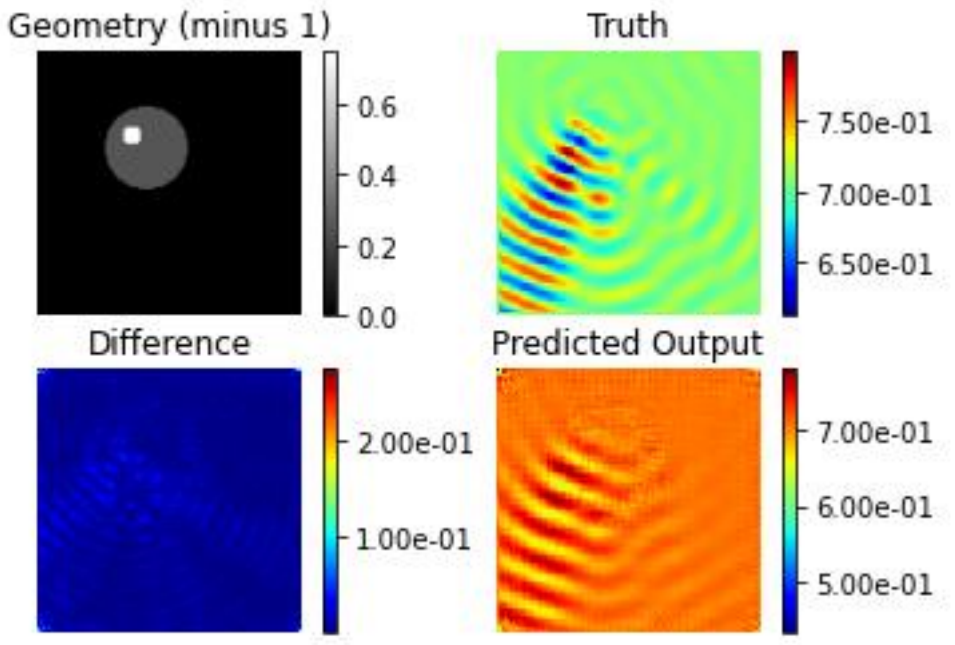

Figure 4. End of third training session. For E1.

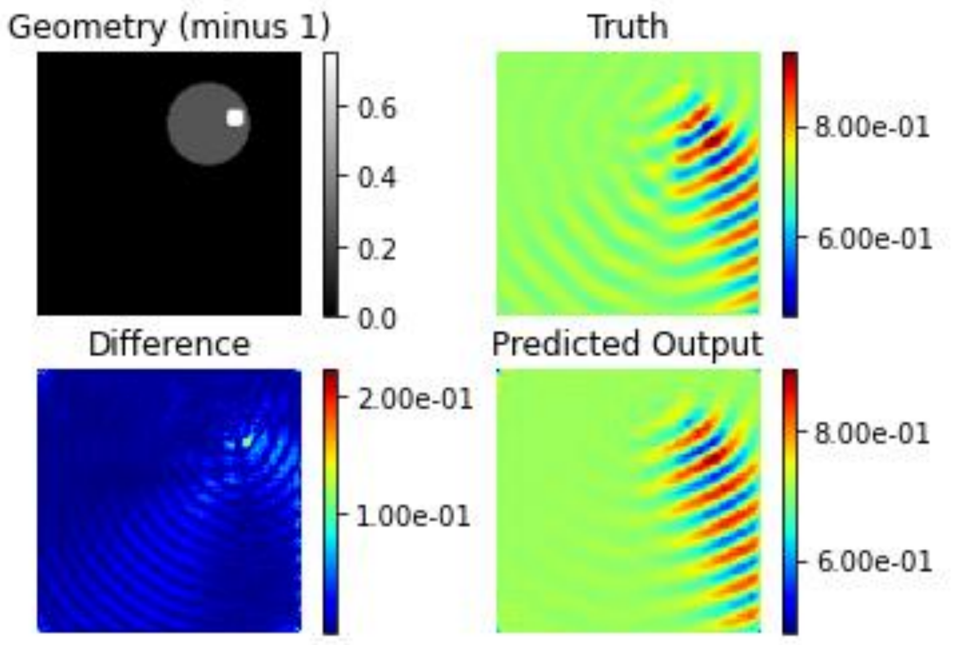

Figure 5. End of fourth training session for E1.

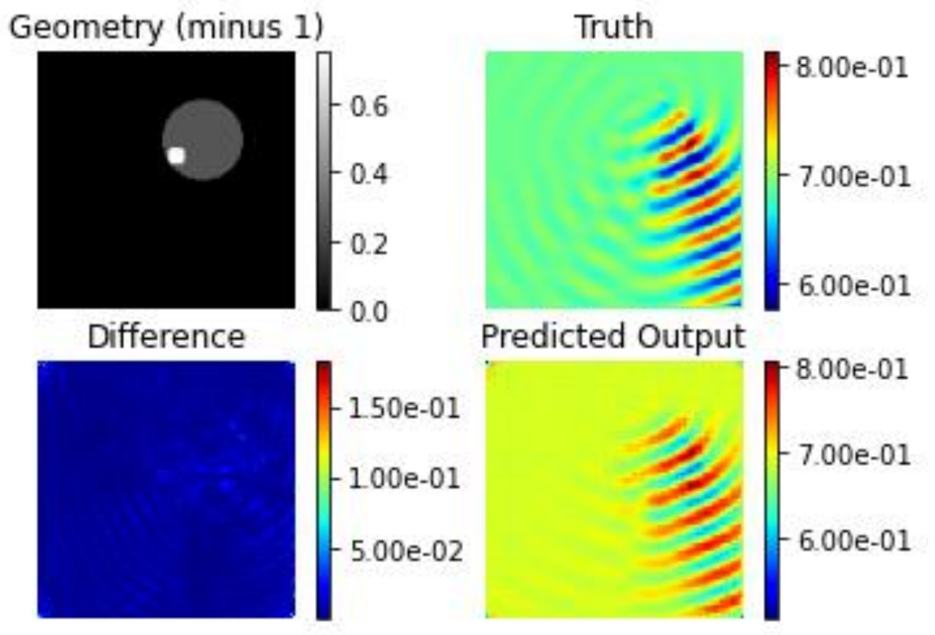

Figure 6. End of fifth training session for E1.

E2 Model Training Commentary

The final total model fitting time for the E2 component of the Prescient2DL deep learning model was 1183.819656610489 seconds, which is roughly twenty minutes. This is almost the same as the E1 component training time and this is to be expected as the same architecture was used in both cases. This does not include the initial creation of the training/validation/testing splits which take time to process from the sample data to correct tensor format. The screen command line printout of the final two training data batches is available on GitHub for reference.

E2 Field Model Training Loss Curves

It should be noted that GitHub hosts all of the loss curves plotted at the end of each epoch. Tensorboard was also used to track the weight updates at each layer, however, analysis and improvement on weight behaviours will remain in the domain of future work due to time limitations.

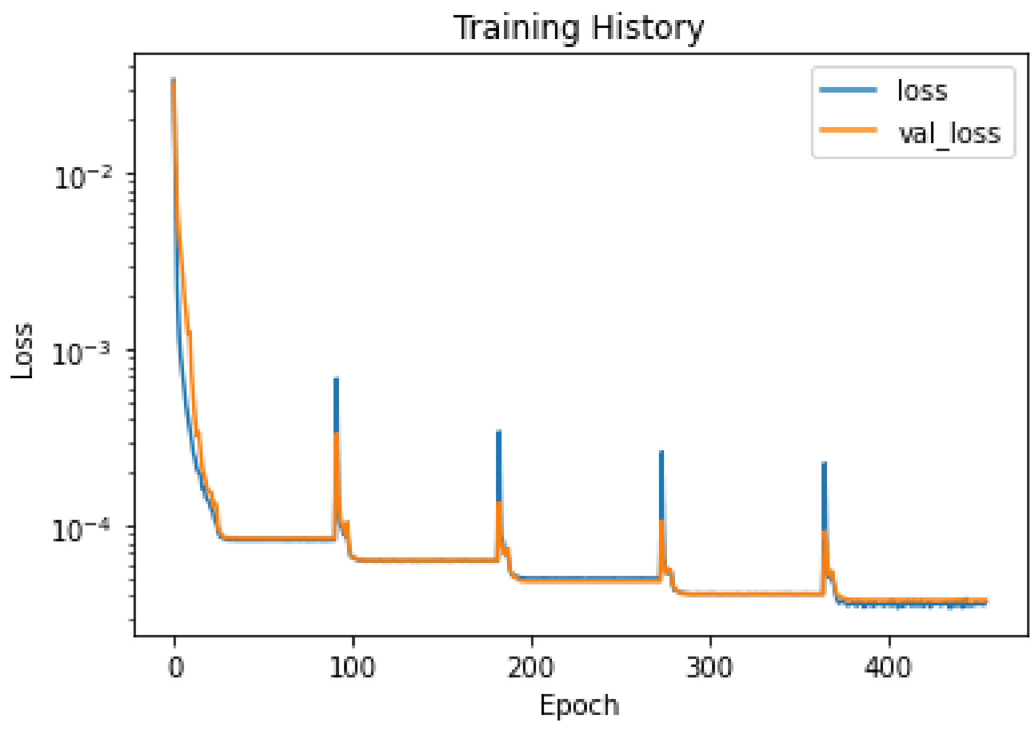

Figure 7. Final Loss Curve Plot E2. Note the axis scale is logarithmic.

E2 Field Model Training Test Scores

For the final training dataset, the following scores were achieved:

Metric	**#**
Training Mean Squared Error Loss	3.7006e-05
Validation Mean Squared Error Loss	3.7967e-05
Final Learning Rate	1.0000e-22

Key Metric: For the final training dataset, the mean squared error test score after training was 4.160570097155869e-05. Since no cross-contamination of samples exist in each split, the validation error here is the correct indicator of model performance from the perspective of the deep learning model compared to the final solved solution provided as the prediction target. This is notably higher than the E1 score, however, the complexity of the y-dimension is more apparent since the incident dipole wave has greater magnitude of change compared to the x-dimension co-ordinate. The final learning rate frequently was reduced to this order of magnitude throughout each training dataset run and this is reflected in the loss curves.

E2 Field Model Training Visual Prediction Results Absolute Component

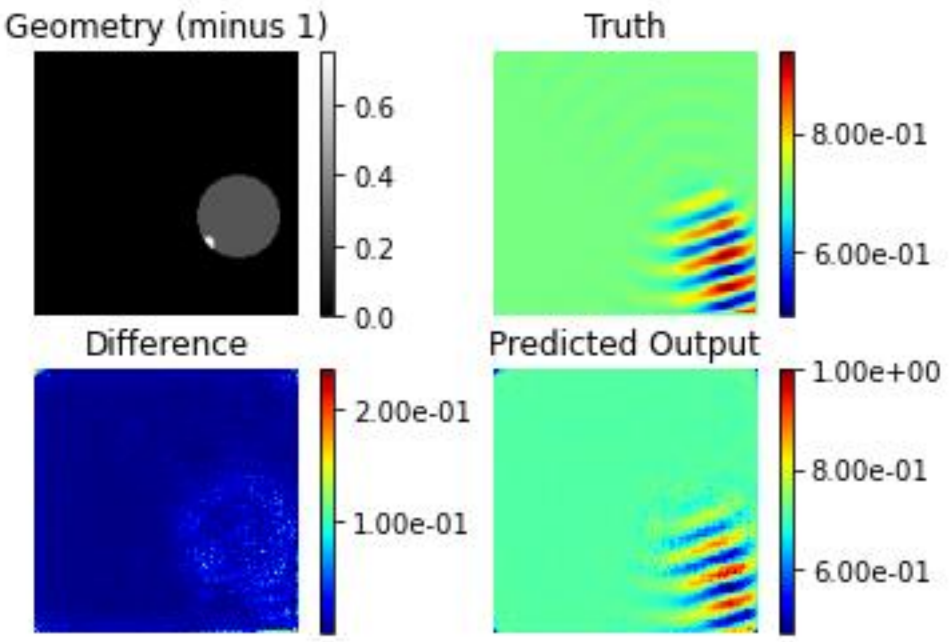

Figure 8. End of first training session for E2.

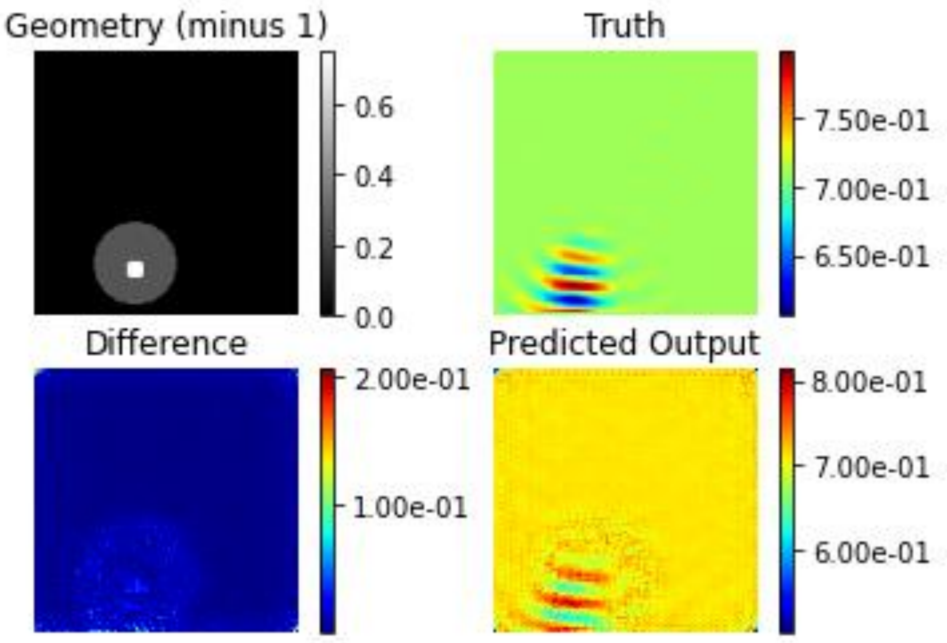

Figure 9. End of second training session for E2.

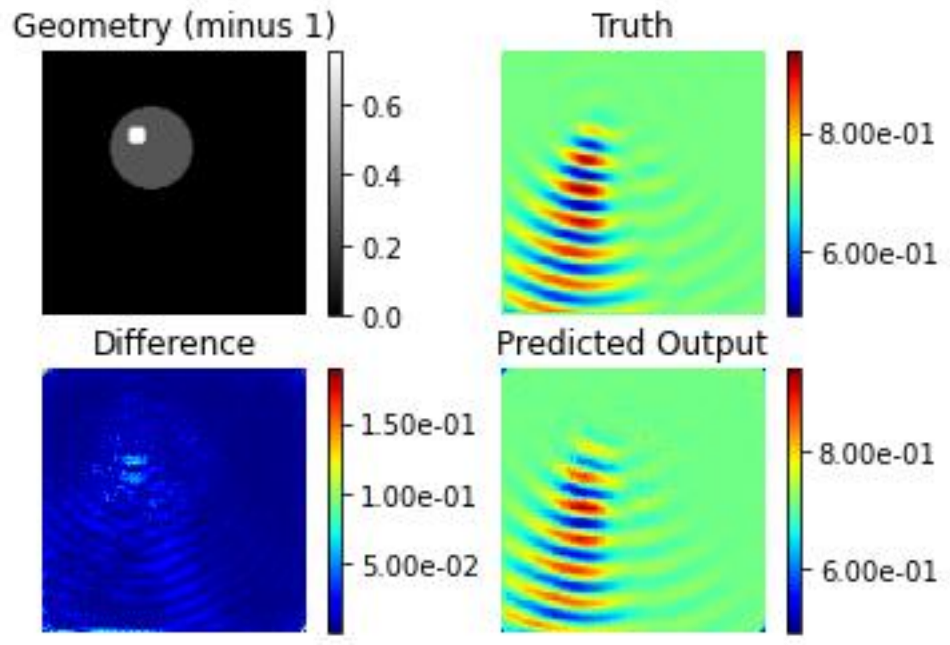

Figure 10. End of third training session for E2.

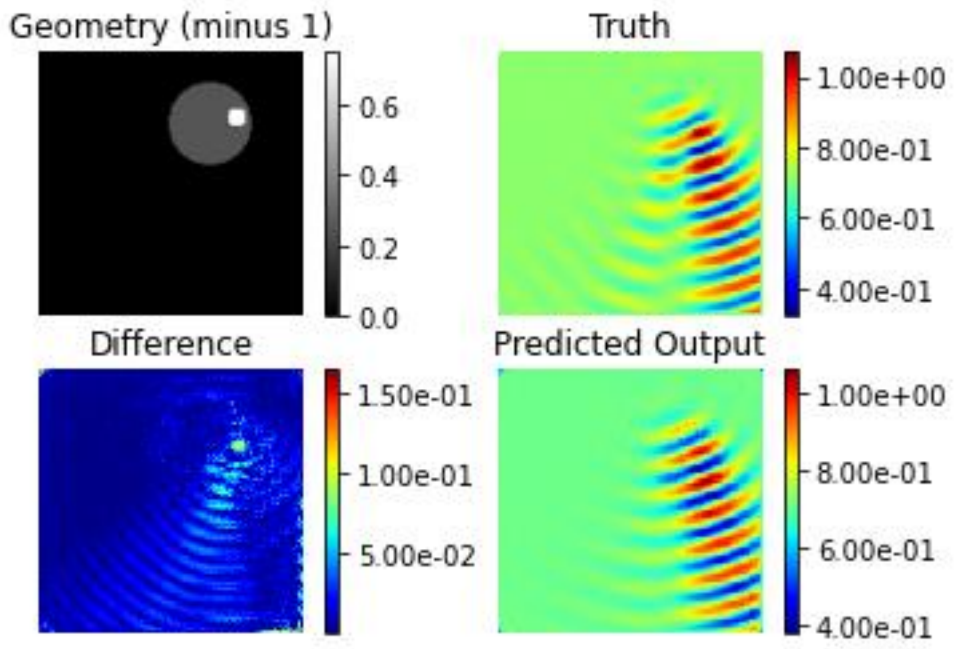

Figure 11. End of fourth training session for E2.

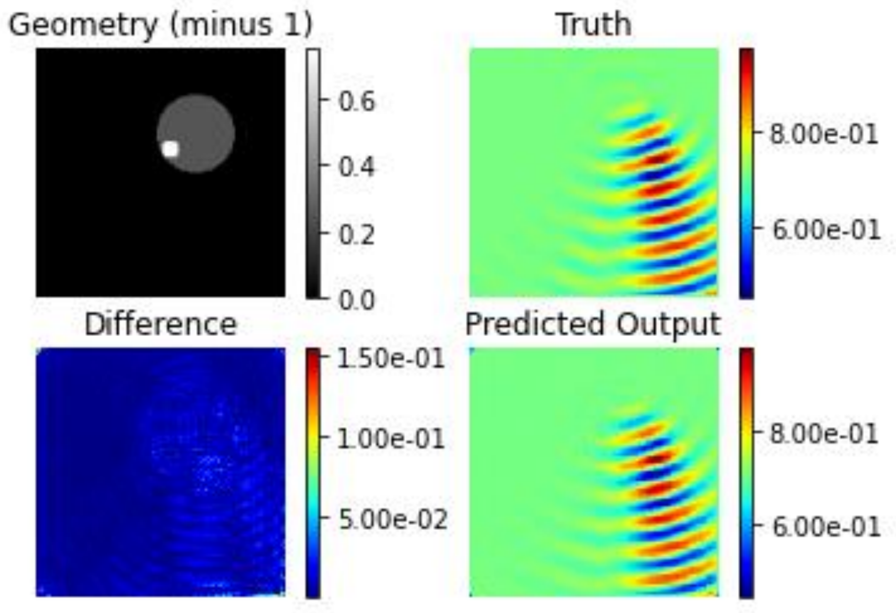

Figure 12. End of fifth training session for E2.

Collected comments on the E1 & E2 Training Process

It appears that both models are not learning in a manner usually desired in deep learning development processes. The student will now provide some commentary on this topic.

The search space of possible geometric configurations for this problem can be approximated with an upper bound on the number of cells in the grid multiplied by the possible starting positions for the smaller scatterer in the scenario. This can be roughly esitmate to be (128^2 * 1600) which gives 26 million. Five-thousand geometric samples is roughly 2% of the total possible coverage. The means squared error results indicate that the deep learning models are successfully capturing a compressed version of the fields in their weights but only up to a certain degree. Based on the oscillating training curve in the final run of the training, it is unlikely that there is insufficient training data. Indeed, more training data could be generated and up to 49000 data samples had been generated in earlier conceptions of the project development. The use of training augmentations is also eliminated since medical applications require pre-designated incident directions. With the exception of horizontal mirroring, data augmentation would shrink the possible permutations in the scene configuration space and increase the probability of duplicates between the training/test/validation sets.

The student finds it much more likely, based on the literature review, that the U-net architecture in its current form is sub-optimal or inappropriate for the problem at hand. The student has reflected on the use of meta-architecture in the project research log and this would be a route that research could follow off the back of the findings made in this project.

At time of writing, the student has begun to interpret the general application of deep learning to the forward problem using U-Net as an implicit recasting of the problem to an inverse one where the inputs are basically noisy inputs that need to be recovered backwards to their scattered fields. This had been the students long term outlook when conducting the literature review, except using different prior information gleamed from Neumann integral expansions in the higher frequency approximations to the wave scattering configurations and then applying denoising to complete the scene. There is a conceptual connection in this idea to the GANs approach documented in [3].

Although the student cannot rule out the possibility that the model has received inadequate training time, the rapid convergence to a similar order of magnitude of loss through all training sessions implies that this is not a training time duration issue. The student also believes increasing the training data size to cover more of the possible permutations would not scale well when higher-contrast problems are tackled since this would dramatically increase the amount of training data required to achieve results, in direct contradiction to developing the model in the first instance.

While the predicted target data was pre-processed and then post-processed to bring the values within a smaller range suitable for deep learning, this was not done for the input data. For this low-contrast problem where the incident waves are already tightly bounded in value, it is unlikely that this is causing a problem. However, the student notes that in the high-contrast scenarios such processing would also need to be carried out on the input data in order for the model weights to adjust quickly and avoid losing permutations to bias adjustment. As already commented, the model incorporates a bias term in each convolution layer but this bias term still needs to learn so processing the input data further may help this delivery.

The student has adapted a learning rate that reduces over time as the loss curve saturates. The use of an increased batch size compared to [4] up to where the computer memory would allow should also have aided the improved training performance. The model also had two extra final linear layers that aimed to provide a blurring effect on the arrays, given the smoothness assumption for the scattered fields. The student also included a Dropout layer need the lower end of the decoding side of the model. The student notes that the use of batch layer normalisation may help the model to train more smoothly, however, implicit regularisation may also be arising from these layers and the model may be struggling to reduce the loss values because of their inclusion at so many levels. The student also did not use hyper-parameter tuning or meta-search libraries such as AutoKeras. In general, the student would suggest experimenting with regularisation on the activity of some of the convolution layers should further research be conducted on this specific model architecture.

Although not formally reported, the student tried to use Xavier weight initialisation as suggested in [4]. This did not positively impact the learning curves.

Finally, the student wants to raise the possibility that the application of deep learning to this problem may not be suitable beyond the generation of emulation. This project has failed to find evidence that infusing deep learning models into conventional Krylov based solvers can improve their convergence properties. While the model has been shown to provide a decent estimation of the target fields, substantial benefits to existing methodologies can only be claimed if either the initial error arising from using the deep model lower the iteration count of the Krylov solver or help the Krylov solver to converge at a faster rate to a solution that meets the error criterion of the simulation. Evidence for either of these goals was not found in the experiments conducted in this project.

Descriptive Statistics of Testing Datasets

The following table gives the descriptive statistics for each statistical test set used to evaluate the impact of Prescient2DL on SolverEMF2. Each set consisted of 100 original samples solved using the naïve initial guess of the incident wave as the scattered field. After training the models for predicting the two scattered fields, a second run of SolverEMF2 was used on the same original samples, allowing for direct comparison across duration of calculation, iteration count and initial error between the original sample information ("_o") and the model-assisted sample information ("_m").

Metric	N	Mean	SD	SE	Coefficient of variation
DS1					
Duration_o	100	1.106213	0.055213	0.005521	0.0499113
Duration_m	100	1.010293	0.093377	0.009338	0.0924252
Iteration_Count_o	100	22.57	0.655282	0.065528	0.0290333
Iteration_Count_m	100	22.05	0.479373	0.047937	0.0217402
Error_Initial_o	100	0.004857	0.002712	0.000271	0.5583008
Error_Initial_m	100	0.001102	0.000492	4.92E-05	0.4466384
DS2					
Duration_o	100	0.772	0.132	0.013	0.171
Duration_m	100	0.72	0.071	0.007	0.099
Iteration_Count_o	100	19.57	0.573	0.057	0.029
Iteration_Count_m	100	19.35	0.52	0.052	0.027
Error_Initial_o	100	0.003	0.002	1.963×10^{-4}	0.619
Error_Initial_m	100	8.014×10^{-4}	3.702×10^{-4}	3.702×10^{-5}	0.462
DS3					
Duration_o	100	2.218	0.198	0.02	0.089
Duration_m	100	2.308	0.246	0.025	0.106
Iteration_Count_o	100	56.65	1.048	0.105	0.019
Iteration_Count_m	100	56.58	0.955	0.096	0.017
Error_Initial_o	100	0.03	0.019	0.002	0.611
Error_Initial_m	100	0.02	0.011	0.001	0.533

Paired t-Tests of Testing Datasets

After training the models for predicting the two scattered fields, a second run of SolverEMF2 was used on the same original samples, allowing for direct comparison across duration of calculation, iteration count and initial error between the original sample information ("_o") and the model-assisted sample information ("_m").

Paired Samples T-Test								
Measure 1	**Measure 2**	**t**	**df**	**p**	**Mean Difference**	**SE Difference**	**Cohen's d**	**SE Cohen's d**
				DS1				
Duration_o	Duration_m	8.9132305	99	< .001	0.0959198	0.0107615	0.8913231	0.1656468
Iteration_Count_o	Iteration_Count_m	7.2478005	99	< .001	0.52	0.0717459	0.7247801	0.1394524
Error_Initial_o	Error_Initial_m	16.5942404	99	< .001	0.0037544	0.0002263	1.659424	0.0587952
				DS2				
Duration_o	Duration_m	3.394	99	< .001	0.052	0.015	0.339	0.15
Iteration_Count_o	Iteration_Count_m	4.2	99	< .001	0.22	0.052	0.42	0.1
Error_Initial_o	Error_Initial_m	14.493	99	< .001	0.002	1.634×10^{-4}	1.449	0.061
				DS3				
Duration_o	Duration_m	-3.3	99	0.001	-0.09	0.027	-0.33	0.125
Iteration_Count_o	Iteration_Count_m	0.572	99	0.569	0.07	0.122	0.057	0.122
Error_Initial_o	Error_Initial_m	12.829	99	< .001	0.01	8.095×10^{-4}	1.283	0.017

Paired t-Tests of Testing Datasets Commentary

Between the descriptive statistics of the testing datasets and the t-Tests conducted on the variables, the impact of Prescient2DL on Solver EMF2 can be established. In the attached appendix Project Plan Proposal, the Primary and Secondary research hypothesis were postulated. The Primary Research question was orientated around using Prescient2DL to improve the Krylov Iterative solver at the heart of SolverEMF2.

CHECK: NOTE Cohen's D.

CHECK: NOT FINISHED

Bibliography

[1] JASP Team, “JASP (Version 0.17.3).” 2023. [Online]. Available: https://jasp-stats.org/

[2] T.-T.-H. Le, H. Kang, and H. Kim, “Towards Incompressible Laminar Flow Estimation Based on Interpolated Feature Generation and Deep Learning,” *Sustainability*, vol. 14, no. 19, Art. no. 19, Jan. 2022, doi: 10.3390/su141911996.

[3] Z. Ma, K. Xu, R. Song, C.-F. Wang, and X. Chen, “Learning-Based Fast Electromagnetic Scattering Solver Through Generative Adversarial Network,” *IEEE Trans. Antennas Propag.*, vol. 69, no. 4, pp. 2194–2208, Apr. 2021, doi: 10.1109/TAP.2020.3026447.

[4] Q. Ren, Y. Wang, Y. Li, and S. Qi, *Sophisticated Electromagnetic Forward Scattering Solver via Deep Learning*. Singapore: Springer, 2022. doi: 10.1007/978-981-16-6261-4.