Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project

Final Report

A Deep Learning Approach to Identify Maximum Stress Distribution Caused by Shock Wave in Geo-mechanical Materials

by

Hao Lu

hl1319@imperial.ac.uk

GitHub login: acse-hl1319

Supervisors:

Dr. Robert Bird

Dr. Adriana Paluszny

August 2020

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

Abstract

Wave propagation patterns in solids have been extensively researched in different areas within the geo-mechanical scope, among which the shock wave has been a concern, e.g. mining detonation tasks. Conventionally, the finite element method was used in these studies for its extensibility and accuracy. Recently, successes gained in deep learning have attracted geophysical scientists to explore ways of applying neural networks in the field of geo-mechanical analysis. In this project, ShockNet, an encoder-decoder neural network, was developed to predict maximum stress distributions caused by randomly initialised shock waves. Mean absolute errors (MAE) and mean squared errors (MSE) were used to calculate losses of predicted results. The dataset fed for ShocNet was generated by ANSYS Student Edition (ANSYS-SE). The result shows a satisfying convergence trend of MAE and MSE for ShockNets and its variance in a 2D domain, reaching at 0.084 and 0.024, respectively, after 250 epochs. Stress domains predicted by trained ShockNet can generally render the input into outputs that are ground truth-alike using only 1/400 of the time using ANSYS-SE. Transferred learning was found capable of decreasing the MSE and MAE to 0.021 and 0.081, respectively, within 100 epochs. Future work could be focusing on refining the input projection scales to increase the internal variance further and predicting the numerical value of maximum stress.

Keywords: finite element method, shock wave, deep learning, encoder-decoder.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

1. Introduction

Wave propagation in solids has been extensively researched in various engineering areas, e.g. waveform inversion for geophysical modelling (Mosser et al., 2020) and material property characterisation via ultrasonic waves (Lahivaara et al., 2018). Various wave types have been studied over the years: ultrasonic wave was used to determine gasoline volatility characteristics by measuring the wave velocity (Takahashi et al., 2000); acoustic emission wave classification was used to enable comprehensive rail crack monitoring (Li et al., 2020); lamb wave sensing network was used in structural health monitoring for aluminium skin(Sbarufatti et al., 2014).

Commented [LH1]: What does blosting mean?

Wave propagation problems can be described by partial differential equations (PDEs). When such PDEs are solvable, one may expect the analytical result that describes the actual problem. However, solutions to real-world wave propagation problems are likely to be unknown, and therefore numerical methods have been dominant for capturing features of wave propagation problems in solids. The most widely used numerical methods include finite difference method (FDM) and finite element method (FEM). FDM can be quickly developed into high order in space but is limited when addressing unstructured mesh (Nie et al., 2020; Lahivaara et al., 2018). By comparison, FEM can mesh arbitrary domains and return high-quality results if the node resolution is refined (Reddy, 2006). FEM has been used to analyse ultrasonic waves in heterogeneous media in 2D (Freed et al., 2016; Lhuillier et al., 2016; Nakahata et al., 2016) and elastic waves in 3D (Van Pamel et al., 2015 & 2017). Among different types of waves, shock propagation has been focused due to its sophisticated essence of irreversible thermodynamic processes and steeping gradients of velocity and temperature (Salas and Iollo, 1996; Mabssout and Pastor, 2003; Baty et al., 2007).

On the other hand, with the improvement of computational abilities, Neural network (NN) has gained considerable developments recently. NN simulates activities of neurons to “learn” features from large datasets to return concluding results like “instinct” without the knowledge of underlying physical equations in advance (Zhu et al., 2017; Kononenko et al., 2018).

Various NN architectures have been developed. Generative adversarial networks (GANs) had been developed for predicting wave propagating patterns in both homogeneous and heterogeneous domains using a multi-scaled generator (Zhu et al., 2017); however, the generator loss increased significantly after 2×10^5 epochs, and non-existing wavefronts were randomly generated at false locations. Sorteberg et al. (2018) applied a recursive NN (RNN) with long short-term memory (LSTM) to predict wavefront contours at different stages, which showed good convergence within the first ten timesteps.

For image generation and classification tasks, convolutional NNs (CNN) have been extensively used in various areas. Liang et al. (2018) developed autoencoders to adapt arbitrary aorta shapes in 3D and generate stress distribution. Nie et al. (2019) predicted stress fields for cantilevered structures through an encoder-decoder architecture, where the result showed good convergence and the visually negligible difference between predicted domains and the ground truths. CNNs were also applied to study wave propagating problems. Lahivaara et al. (2018) used an AlexNet-based CNN to characterise tortuosity and porosity. Li et al. (2019) developed a U-Net-based (Ronneberger et al., 2015) autoencoder to predict fracture locations by training it with 2D-seismic sections. Both studies successfully rendered the initial conditions to comparative predictions. Thus, this project focused on the encoder-decoder architecture to deliver good predicted results.

This project used ANSYS-SE (ANSYS-SE) to simulate the Von Mises stress field when the maximum stress overtime occurred. The domain was set as a rectangle in 2D of 10x20 mm^2 and assigned with six geo-mechanical materials. The lower boundary was fixed, while an instantaneous force was loaded on the upper boundary (UB) in a period of 10^{-9} – 10^{-8} s to simulate a shock wave. The jump in force was randomly allocated on the UB within the range of 0–2N. Initial conditions were projected based on boundary conditions, force allocations magnitudes and material densities.

An encoder-decoder NN, ShockNet, was developed in this project. ShockNet extract features from a projected initial condition image (the input) to predict the stress distribution field image (the output). During each training epoch, the output was optimised based on the ground-truth results simulated by ANSYS-SE. ShockNet was based on U-Net (Ronneberger et al., 2015), which enabled multi-scaled feature extractions at different depths. ShockNet also incorporates other state-of-the-art designs of modern CNNs. Kaiming et al. (2017) developed residual blocks which enable the NN to deepen without gradients vanishment problems, and therefore further increment learning capabilities; Hu et al. (2017) developed Squeeze and Excitation (SE) blocks which squeezed the feature map into a channel-wise numerical value and excited with the original convolutional block. The training process is governed based on image differences, where the criteria used in this project refer to MAE and MSE.

Commented [LH2]: what does the comment mean here? The following part should already include other "state-of-the-art" desighs e.g. resnet and se blocks.

Results showed that ShockNet could encode 2D input images projected from initial conditions to extract features, and decode the features to output predicted stress field. Good convergence trends were observed for ShockNet and other variant architectures. Minimum validation MSE and MAE reached at 0.018 and 0.075 respectively after 183 epochs, compared to 0.14 and 0.22 by StressNet after 5,000 epochs by Nie et al. (2019). Transferred learning was also studied, which showed the capability of ShockNet to generalise its learnt parameters to adapt higher-resolution dataset. ShockNet was also extended to predict maximum and minimum stress values of the output stress field (TShockNet).

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

2. Methodology

In this project, ANSYS-SE was used to simulate random shock waves and generate 30,000 stress fields, where initial conditions have been saved in the file name. Initial conditions were then extracted to project input domains for ShockNet. ShockNet was developed for this project based on an encoder-decoder structure. The exact methodology used in this project will be structured as following three sections: dataset generation (2.1), ShockNet architecture development (2.2) and numeric stress value predictions (2.3), and module structure description (2.4).

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

2.1. Dataset Generation

This section describes general domain setup in ANSYS-SE, including the domain material assignments, mesh size selection and ANSYS-SE automation (2.1.1). Shock wave initialisation via Heaviside step function using randomised initial force was introduced in 2.1.2, while how sample inputs were projected from initial conditions was discussed in Initial Condition Projection2.1.3.

2.1.1. Domain Setup

ANSYS-SE was used to generate stress fields as ground truths numerically. The target domain was selected as 2D to generate the dataset as large as possible to develop ShockNet's learning ability. A 10x20 mm^2 rectangular domain was initialised in the XY plane. The mesh size of 1mm was used as it could maintain the capability of capturing sharp wavefront features without costing too long to solve the propagation problem (~20s per sample on AMD Ryzen 3600x). The generation process was fully automated with Python scripts using built-in scripting functionality. The initial condition was saved and reused to project corresponding input images for ShockNet. Domains were assigned with six common geo-mechanical materials: dolomite, sandstones, limestone, granite, basalt and chalks, with densities ranging from 2000 to 3000 kg/m^3 (Table 1). For each sample, the field was homogeneous and isotropic. The lower boundary was fixed during the simulation.

Table 1 Physical features of materials used for shock wave simulations.

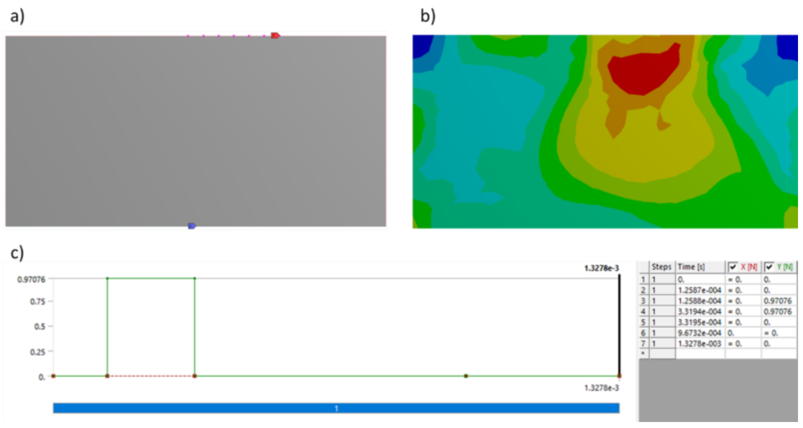
Material	Sandstones	Dolomite	Limestone	Chalks	Basalt	Granite
Density /kg m^{-3}	2600	2840	2160	2499	3000	2750
Young's Modulus /GPa	34.0	58.6	37.8	16.4	57.0	50.0
Poisson's Ratio	0.234	0.282	0.308	0.078	0.250	0.200

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

2.1.2. Shock Wave Approximation and Randomisation

As shock wave is produced by an impact which causes steep kinetic energy difference in a short period, one can expect the discontinuity of a shock wave to be approximated using a Heaviside step function (Villarreal 2006; Baty et al., 2007; Baty and Margolin, 2018):

$$H(E) = \begin{cases} E, t \in [t_{start}, t_{end}] \\ \quad 0, others \end{cases}$$



	Steps	Time [s]	✓ X [N]	✓ Y [N]
1	1	0.	= 0.	0.
2	1	1.2587e-004	= 0.	0.
3	1	1.2588e-004	= 0.	0.97076
4	1	3.3194e-004	= 0.	0.97076
5	1	3.3195e-004	= 0.	0.
6	1	9.6732e-004	0.	= 0.
7	1	1.3278e-003	= 0.	0.
*				

Figure 1: Force initialisation. a) and b) were examples for the initial condition and solved results from ANSYS-SE. The lower part c) refers to discontinuities in force.

where t_{start} and t_{end} define when the force was loaded on and removed from the UB and E stands for the kinetic energy of each node on UB. Random instantaneous forces were applied upon UB to simulate randomised shock waves. t_{start} and t_{end} were randomly generated during the entire simulation period. As the total simulation time is of 10^{-5}s, the time scale of the force jumping was randomly set between $10^{-9} – 10^{-8}$s to better the approximation of shock waves (Figure 1).

The Explicit Dynamics module from ANSYS-SE Mechanical platform was used in this project which applied a finite element (Lagrange) solver. Explicit Dynamics module was developed to address short timescale problems and sharp jumps in shock response by an improved arbitrary Lagrange-Euler algorithm (ANSYS Explicit Dynamics Brochure). Explicit Dynamics module supports applying instantaneous force to monitor a shock on the entire UB (boundary force) or selected nodes (nodal force).

The combination of randomised forces on UB and different materials helped to increase the contour shape varieties of the output stress domain. Two UB force randomisation methods were used: the first applied a randomised force on nodes from the entire UB, while the second applied forces on arbitrarily selected nodes on UB, which simulated different shock source lengths (Figure 1). Both randomisations applied a force within the range of 0-2N. Pseudocodes in blabla described how the randomisation was achieved automatically in ANSYS-SE.

Commented [LH3]: Will add informaation

Examples for both boundary force and nodal force were shown in Figure 2. During the automated simulation using ANSYS-SE, predicted stress fields and initial conditions were saved. Saved information included the time used for simulation, the maximum Von Mises stress over time and the minimum stress recorded at the same stage, and time slots for the force discontinuity. For forces cast on selected nodes on UB, indices of loaded nodes were recorded. Output stress fields were saved as EPS format and reshaped into 48x96 and 64x128.

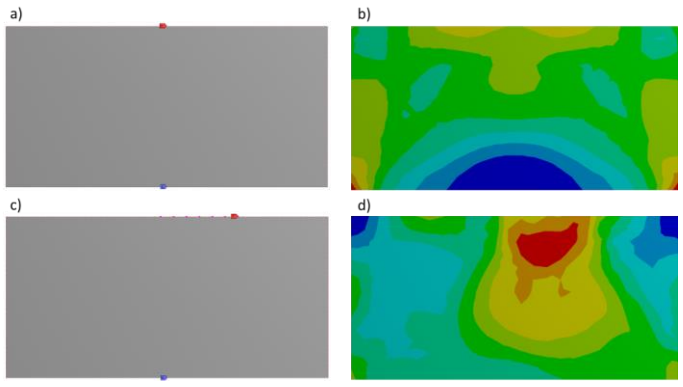


Figure 2: Initial force and supporting conditions (left) and corresponding numerical results (right). a) & b) are examples with forces cast on the entire upper boundary where the numerical result showed symmetric stress field, whereas for c) & d) the load was added on randomly selected nodes which led to asymmetricity.

2.1.3. Initial Condition Projection

Initial conditions were unpacked and then processed by linear scalings. Figure 3 illustrates examples of the projected initial conditions as inputs for ShockNet. The size (number of pixels in width and height) of the input image was the same as stress field image. The magnitude of shock forces and material densities were scaled into RGB channelled colours to present initial conditions. A scale of 0.085 was set for projecting the density into greyscale pixels, which resulted in 183-255 corresponding to the density range of 2160-3000 kg/m^3.

Similarly, a scale of 127 was set to visualise projected forces (darker yellow denotes larger force magnitude). The lower 1/10 of the image was set as red to represent the fixed supporting lower boundary. Other areas were assigned with blue (B: 255) to maintain the 2:1 ratio of width to height and stand out from other scaled sections.

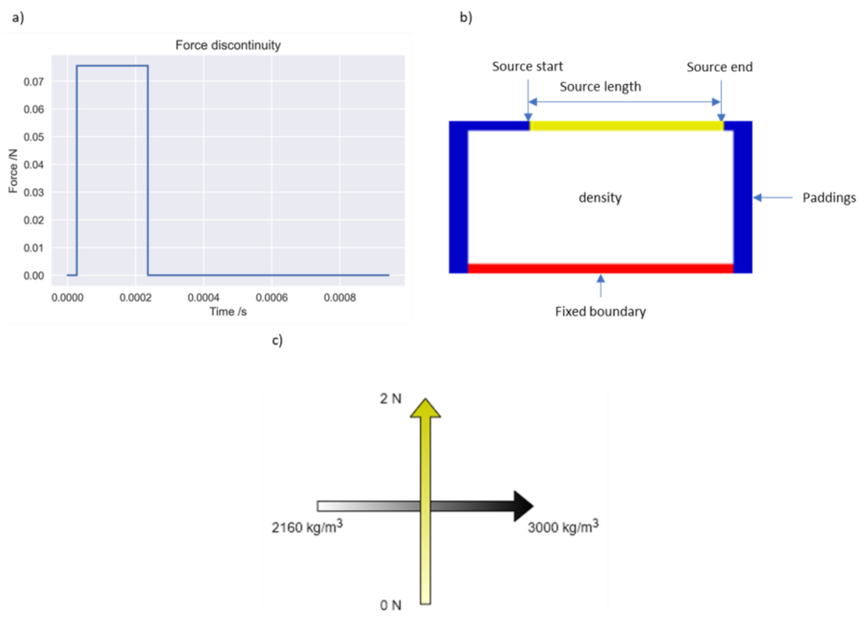


Figure 3: Initial condition projection examples. Figure a) denotes the force discontinuity cast on the source (dark yellow) in figure b). b) illustrates the components in an initial condition image and c) explains how the colour scale related to the physical parameters.

2.2. ShockNet

In recent studies, CNNs have been widely used for image classification and image generation tasks (Bai et al., 2018; Liang et al., 2018; Balu et al., 2019; Nie et al., 2019; Gao et al., 2020). CNNs contain convolutional blocks which apply feature extracting kernels to capture information from the input. The encoder-decoder structure contains CNN blocks to encode features and transpose CNN layers to decode extracted information. Encoder-decoder structures have been extensively applied for image generation and classification tasks. U-Net (Ronneberger et al., 2015) is one of the most successful encoder-decoder structures which have shown practical value in biomechanical (Bai et al., 2018; Liang et al., 2018; Balu et al., 2019) and geo-mechanical areas (Nie et al., 2019; Gao et al., 2020).

2.2.1. Image generation architecture: ShockNet

The original U-Net contains, however, only 12 layers from the input to the bridge block, which is shallow compared to more recent CNN architectures. One of the reasons that the original U-Net cannot be extended to further depth is gradient vanishment (Kaiming et al., 2017). When the NN architecture becomes deep, the small gradient will decrease further under the chain rule through propagations. Kaiming et al. (2017) introduced the residual network (ResNet), which enables the NN to be as deep as 150 layers by preserving the information from previous blocks. The residual block helps to enable the flow of memory/information from the initial layer to the final layer.

At each SL, a Squeeze and Excitation (SE) block was added. SE structure was introduced by Hu et al. (2017) which extended the advantage of ResNet. SE blocks enable channel-wise multiplication with a numerical value, which has shown the strength to decrease the top-1 error ranging from 0.61 to 0.85 (depending on the depth) with negligible additional computational cost in classification problems (Hu et al., 2017).

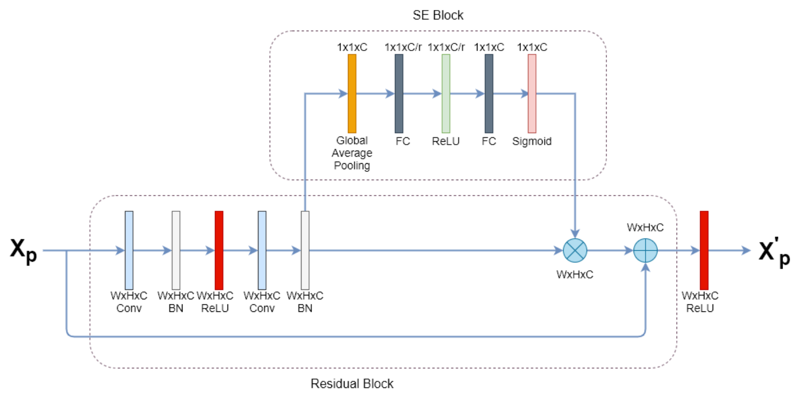


Figure 4: A ShockNet block at one SL. Each residual block contains two convolutional blocks, followed by a batch normalisation for each. A copy of X_p was reserved. The output from the second batch normalisation layer would be (1) sent to the SE block and (2) saved as a copy. Channel-wise scales are output by the SE block and multiply with the saved copy. The multiplied result would then be added with the reserved copy of X_p.

Based on previous successful improvements, ShockNet was developed. Figure 4 presented a single ShockNet level (SL), where each SL contains a basic two-layer residual block and an SE block. Before activated by the last ReLU layer, the last batch normalisation layer would be sent to the SE block for channel-wise multiplication.

The overall ShockNet architecture was shown in Figure 5. ShockNet down-sampled the input image to extract initial condition features. At the end of the SE block at each depth, the tensor would be going in two directions: one would be sent to a max-pooling layer for downsampling side, and another would be preserved for later concatenation to the upsampling side. Such extracted features would be reserved at each level (ShockNet Level, SL) for concatenation with upsampling blocks, enabling multi-scaled information additions at each SL. A max-pooling layer was added to extract the most prominent features from the last level.

ShockNet3 and ShockNet4 were implemented in this project corresponding to 3 SLs and 4 SLs, respectively. After a specified number of SLs, a bridge block connects the downsampling part with the upsampling part. The upsampling part contains the same SLs but is connected by transposed convolutional layers.

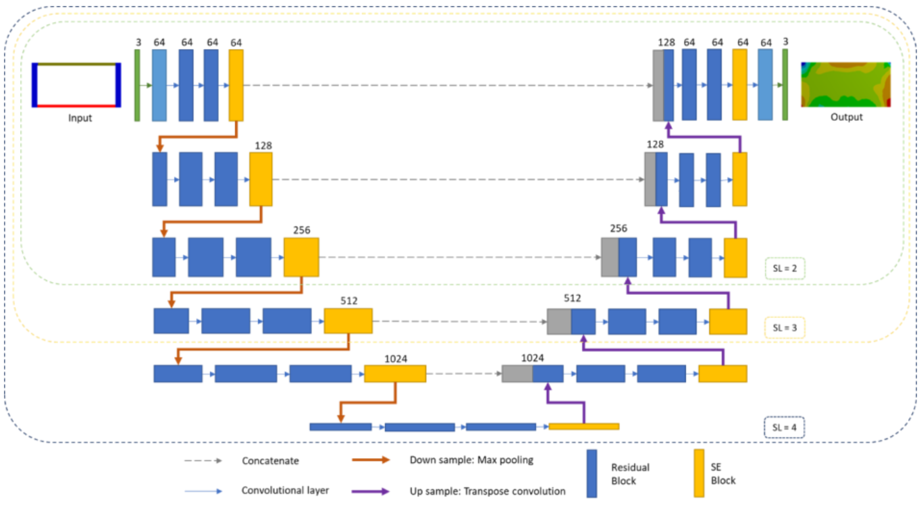


Figure 5: ShockNet architecture. SL numbers were labelled with the defined scope. Red and purple arrows represent max pooling and transpose convolution layers.

2.2.2. ShockNet Training Strategies

This project used MAE and MSE to track the training and validation losses of images:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{3}(y_{i,c} - \hat{y}_{i,c})$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{3}(y_{i,c} - \hat{y}_{i,c})^2$$

where $y_{i,c}$ refers to the predicted value of a pixel at each RGB channel whereas $\hat{y}_i$ refers to the ground truth; n refers to the total number of pixels (8192 and 4608 for 48x96 and 64x128, respectively). ADADELTA was used as the optimiser to automatically changing the learning rate without aggressively decreasing it (Zeiler, 2012). Both ShockNet3 and ShockNet4 were trained to compare and determine how depths would affect the training and validation process.

Data augmentation was frequently used to feed NNs with more diversified data if the dataset size was limited. However, initial conditions and numerical stress fields in this project were strictly corresponded, suggesting that augmenting only the input would possibly compromise the exact relation between the initialised domain and the ground truth result.

As ShockNet uses ReLU as its primary activation function, Kaiming initialisations were used in this project (He et al., 2015). In this project, Kaiming normal initialisation (KNI) was the primary initialisation method, whereas Kaiming uniform initialisation (KUI) was also applied as a comparison to the former.

2.3. Image generation with stress value predictions: TShockNet

The author also developed a two-branched architecture to enable ShockNet not only to generate stress domain images but also to predict numeric values of maximum and minimum stresses (TShockNet). TShockNet would encode the input image first as in ShockNet during down-sampling, but in addition to decoding the low-level features and generate images, TShockNet would also direct to fully connected layers to predict maximum and minimum stress values. As the range of maximum and minimum stress varied substantially (10^3 – 10^{10} Pa), TShockNet predicted the logarithm of the stress:

$$I_{max} = log_{10}P_{max}$$

$$I_{min} = log_{10}P_{min}$$

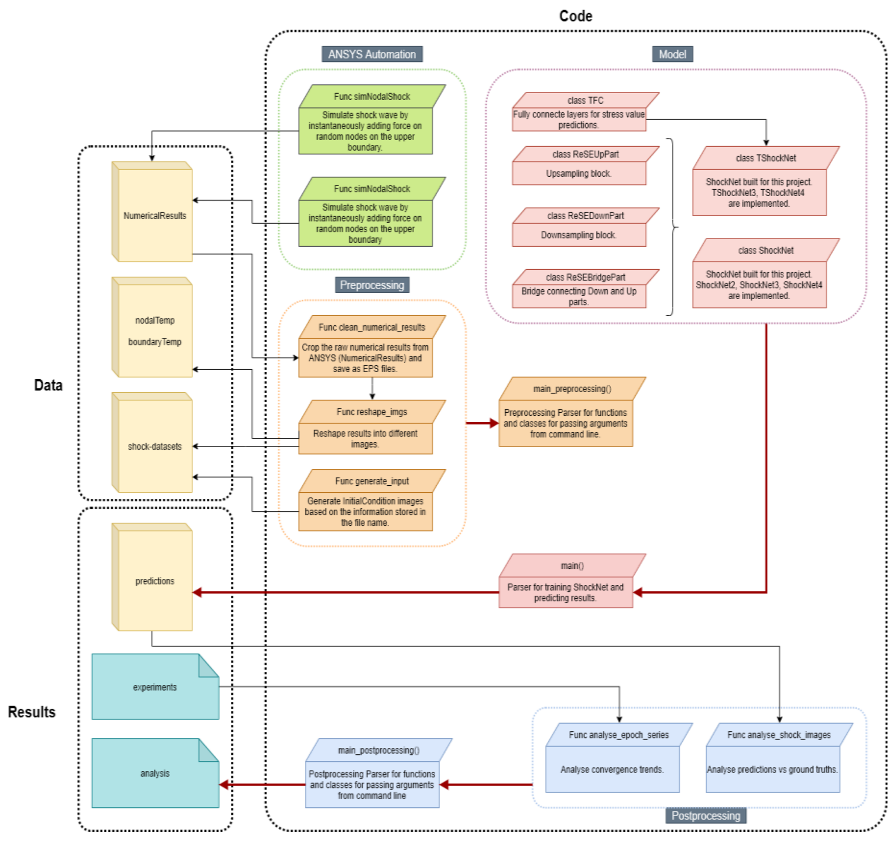
instead of the actual stress value, where P_{max} and P_{min} are maximum and minimum stress, respectively. The L1 loss was used to calculate the summation of maximum and minimum stress losses deviated from the ground truth:

$$L1loss_{stress} = \sum_{i=1}^{n}(I_{max} - I_{max,true}) + (I_{min} - I_{min,true})$$

where $n = 48 \times 96 = 4608$ for TShockNet.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

2.4. Code Structure



Code
ANSYS Automation
Model
Func simNodalShock
Simulate shock wave by instantaneously adding force on random nodes on the upper boundary.
Func simNodalShock
Simulate shock wave by instantaneously adding force on random nodes on the upper boundary
class TFC
Fully connecte layers for stress value predictions.
class ReSEUpPart
Upsampling block.
class ReSEDownPart
Downsampling block.
class ReSEBridgePart
Bridge connecting Down and Up parts.
class TShockNet
ShockNet built for this project. TShockNet3, TShockNet4 are implemented.
class ShockNet
ShockNet built for this project. ShockNet2, ShockNet3, ShockNet4 are implemented.
Data
NumericalResults
nodalTemp
boundaryTemp
shock-datasets
Preprocessing
Func clean_numerical_results
Crop the raw numerical results from ANSYS (NumericalResults) and save as EPS files.
Func reshape_imgs
Reshape results into different images.
Func generate_input
Generate InitialCondition images based on the information stored in the file name.
main_preprocessing()
Preprocessing Parser for functions and classes for passing arguments from command line.
main()
Parser for training ShockNet and predicting results.
Results
predictions
experiments
analysis
main_postprocessing()
Postprocessing Parser for functions and classes for passing arguments from command line
Func analyse_epoch_series
Analyse convergence trends.
Func analyse_shock_images
Analyse predictions vs ground truths.
Postprocessing

Figure 6: Module structure for codes.

Figure 6 showed the primary code structure, including ANSYS SE automation, ShockNet model designs and pre- and post-processing. ANSYS-SE automation codes enabled fast randomised shock wave simulations, where the numerically calculated stress fields were stored and processed by Preprocessing codes. Preprocessing codes cleaned and reshaped the original output from ANSYS-SE for projecting initial conditions as inputs for ShockNet.

ShockNet codes were composed of basic ShockNet blocks, where TShockNet had a new branch of fully connected layers for numeric stress predictions. ShockNets and TShockNets were parsed in the main() function where training and validation losses were recorded in *experiments,* and ShockNet-predicted stress fields were stored in *predictions*.

Postprocessing codes would then pick up loss records per epoch from *experiments* and *predictions* from ShockNets to generate results for analysis and discussions in 3.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

3. Results and Discussion

Two types of datasets had been trained in this project: datasets with forces initialised on the entire boundary (BDs) and randomly selected nodes (NDs). The dataset size was 30,000 for each image pixel resolution, which had been divided into 22,000 and 8,000 as training and validation subsets, respectively. ANSYS-SE approximately spent 15–22s on simulating each shock wave case and about 24 hours on running all 30,000 samples (AMD Ryzen 5 3600X CPU). Data were generated by simultaneously running six ANSYS-SE processes on the desktop which consumed over 16GB RAM (ANSYS-SE's Explicit Module does not support parallelised computations for 2D problems within a single workbench).

All codes were implemented based on the open-source framework, Pytorch, and run on Google's Colaboratory, which provided server-level backend GPUs. Training and validation sessions were run on NVIDA's Tesla P100-PCIE-16GB.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

3.1. Training and Validation Losses

This section shows the results of convergence comparisons between ShockNet4 trained on NDs and BDs (3.1.1). As a result, NDs provided more variable stress domains which thus improved the learning capability of ShockNet4. In 3.1.2, comparisons between ShockNet3 and ShockNet4 using KUI and KNI were shown and discussed.

3.1.1. NDs vs BDs

Table 2 showed the mean and standard deviation difference of RGB channels between the two datasets and denoted the validation MAE and MSE of ShockNet4 after trained for 200 epochs. Comparing to BDs, NDs showed more considerable internal variance, which improved ShockNet's learning capability. When trained with BDs, the symmetric instantaneous load led to higher losses for both MAE and MSE. Figure 7 compared the trend of convergence when trained on BDs and NDs. ShockNet4 trained with BDs converged faster in a more stable pattern but stagnated after only ten epochs. By comparison, when trained with NDs, the ShockNet4 showed the consistent potential to reach lower loss for both MSE and MAE. In this case, in the following experiments, NDs were primarily used for training the ShockNet and predicting stress fields.

Table 2: Input internal variance and validation results after trained for 200 epochs for NDs and BDs of 48x96 resolution.

Dataset	Mean	Std dev	MAE	MSE
NDs	R: 180.15178385 G: 204.8688151 B: 181.6319987	R: 84.79378607 G: 61.2799176 B: 89.29453818	0.0843	0.0230
BDs	R: 200.43989413 G: 229.30037593 B: 200.44750417	R: 70.55906642 G: 11.49446858 B: 81.78639965	0.0873	0.0317

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

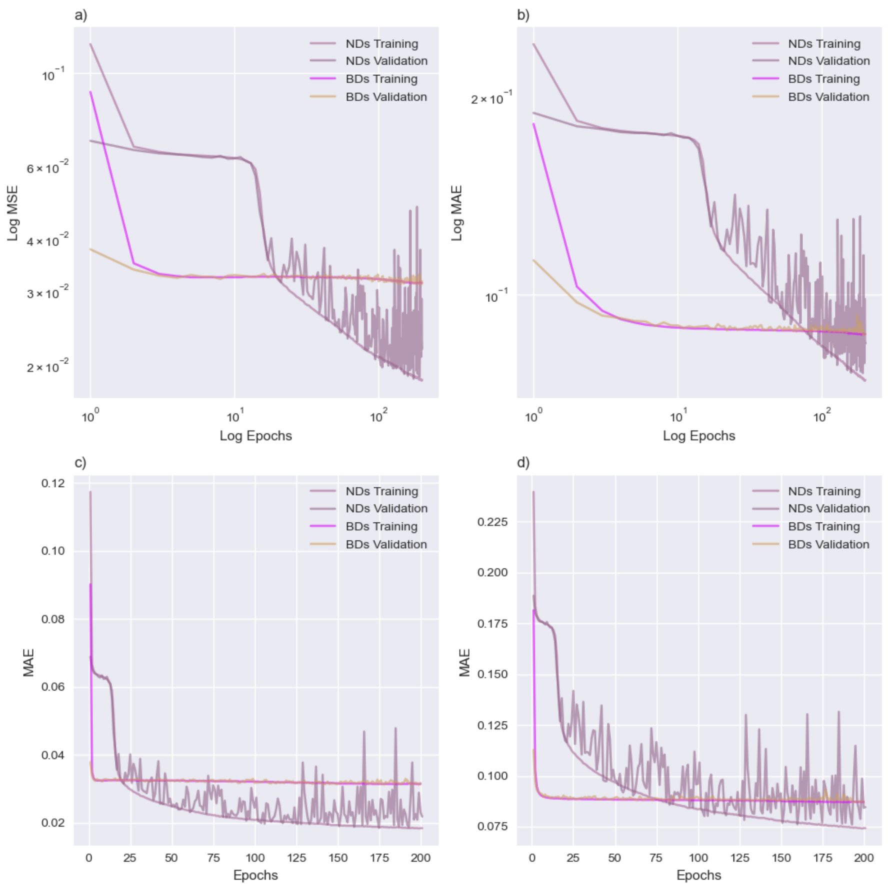


Figure 7: Comparison of ShockNet4 between training with BDs and training with BDs. Figure a) and b) refer to log-log plots.

3.1.2. Comparisons between ShockNet and its variant

Figure 8 showed the performance of ShockNet for different SL numbers and initialisation methods when trained with NDs. ShockNet4 was initialised with both KUI and KNI methods (ShockNet4-uni and ShockNet4-norm, respectively), and ShockNet3 initialised with Kaiming normal (ShockNet3-norm) was also added for parallel comparison. All three curves in Figure 8 showed good convergence trends where the optimisers were working to reach the global minimum. The platform-like pattern showed on all curves was a typical period when the optimiser was temporarily trapped within the local minimum. ShockNet3-norm appeared to be able to escape from its first local minimum faster than the ShockNet4 variants (10 epochs for ShockNet3-norm comparing to 15 and 20 epochs for ShockNet4-norm and ShockNet4-uni respectively). MAE appeared to be more sensitive to the training process, showing a steeper decrease with epochs

ShockNet4-norm presented the lowest MAE and MSE within 200 epochs, where ShockNet4-uni showed similar training losses yet unstable validation errors. It appeared that the KUI method would cause the losses to explode. The increase started at 125th epoch, where the validation MSE reached 0.10 and MAE at 0.19. The amplitude continued to soar to 0.16 and 0.23 for MSE and MAE, respectively.

ShockNet3-norm, on the other hand, presented the most considerable losses, ending at 0.024 (MSE) and 0.085 (MAE) after 200 epochs, whereas ShockNet4-norm reached at 0.019 (MSE) and 0.075 (MAE). The curve shape was similar to it of ShockNet4-norm, though, which had been continuously decreasing stably until 200 epochs. This suggested that it might worth further experimenting on both ShockNet4 and ShockNet3 to discover the effects of NN layers on such shock-wave simulation problems.

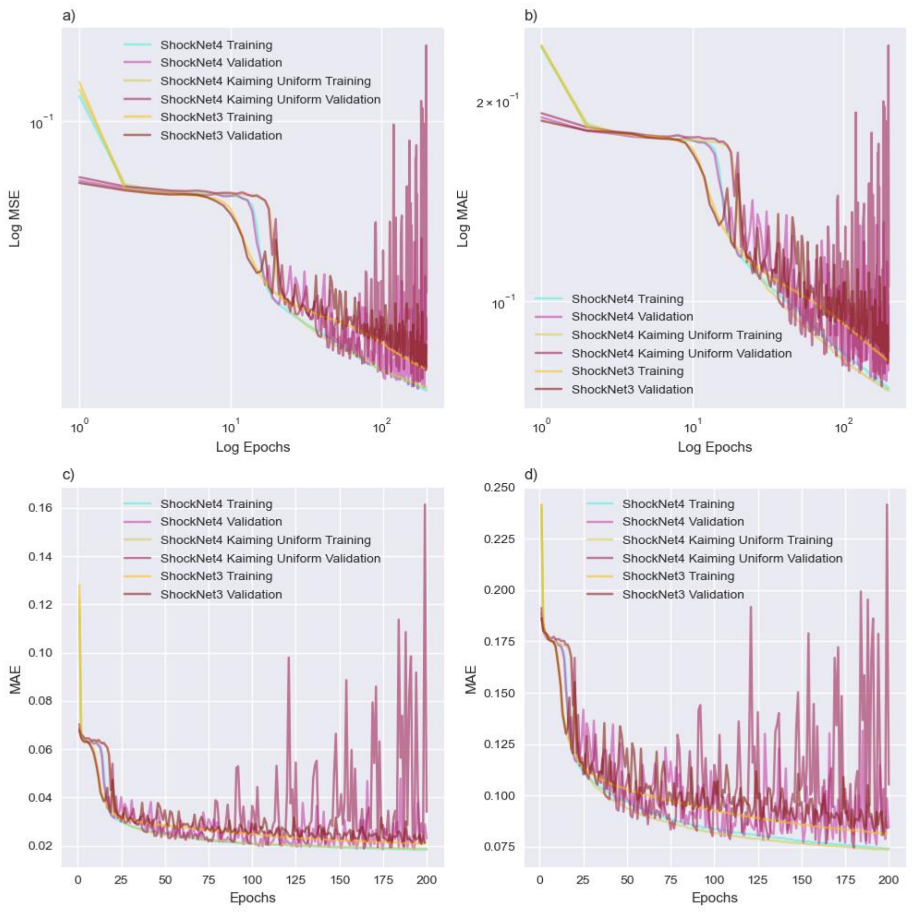


Figure 8: Convergence comparisons. Each graph contains training and validation MSE and MAE of experimenting with (1) ShockNet4 using KNI, (2) ShockNet4 using KUI, and (3) ShockNet3 using KNI. Figure a) and b) refer to log-log plots.

3.2. Predictions vs Ground Truths

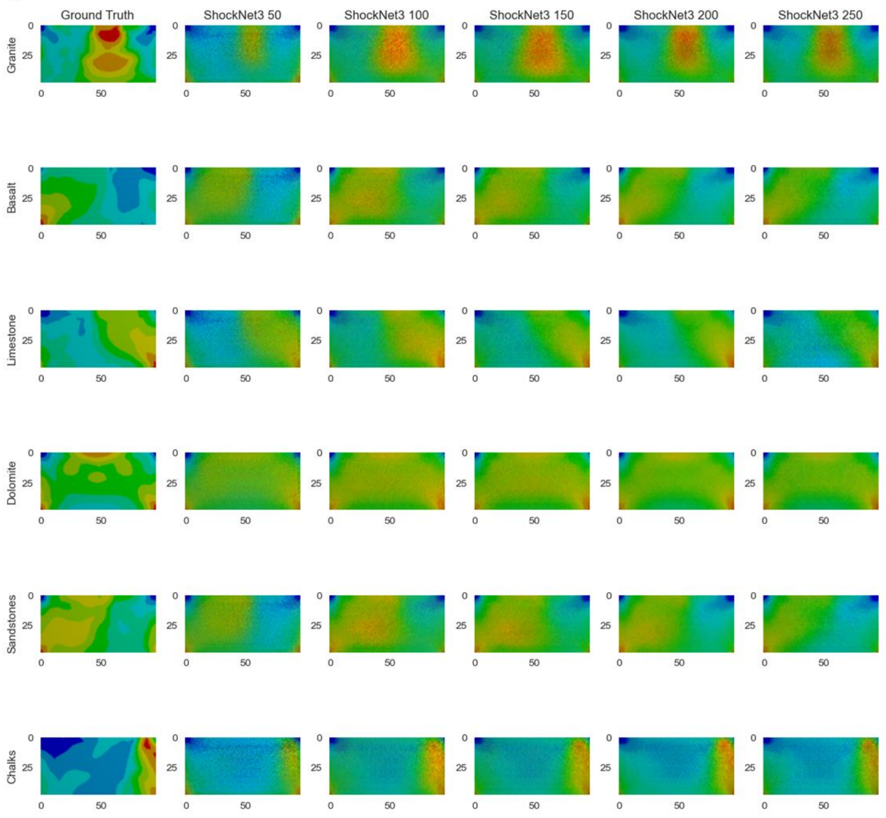
ShockNet3 and ShockNet4 trained with different epochs were saved and reused to predict stress fields (Figure 9). Samples were randomly extracted from the prediction subset for each material initialised in the dataset. It took an average of 0.051s and 0.063s to generate a stress field prediction for ShockNet3 and ShockNet4 (on NVIDA's Tesla P100-PCIE-16GB), respectively, comparing to the average time consumption of 20s for ANSYS-SE.

Both ShockNet3 and ShockNet4 had been generating progressively more precise outputs when trained with more epochs. Contours were indistinguishable at 50th and 100th epoch yet became generally sharper after 250 epochs (Figure 9). The area of maximum stress was becoming more appropriately proportioned with the contours refined every 50 epochs. Noise features were dominant at 50th epoch but substantially suppressed at 250th epoch. Comparing to ShockNet3, ShockNet4 showed more promising ability to render more details and reduce noises in the output stress domain, which, however, cost about 26.3% more time to train with NDs per epoch than ShockNet3 (~19 minutes for ShockNet3 and ~24 minutes for ShockNet4).

Figure 10 showed six basalt results generated from ShockNet4 after trained for 250 epochs. Among six samples, Basalt 1 showed minimum MSE (0.0112) and MAE (0.0630) whereas Basalt 5 showed maximum MSE (0.0347) and MAE (0.1109). MSE and MAE increased when the maximum stress area (i.e. red area) became more substantial. All samples indicated that ShockNet4 was able to identify the maximum and minimum stress areas at corners. The comparison between Basalt 1 and Basalt 4 suggested the effect of the internal variance of the ground truth image: the large minimum (blue) stress area in Basalt 1 was sharply differentiated from the medium stress (green) area whereas, in Basalt 4, the area of minimum stress was much smaller which reduced ShockNet4's ability to refine the stress boundary.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

a)



GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

b)

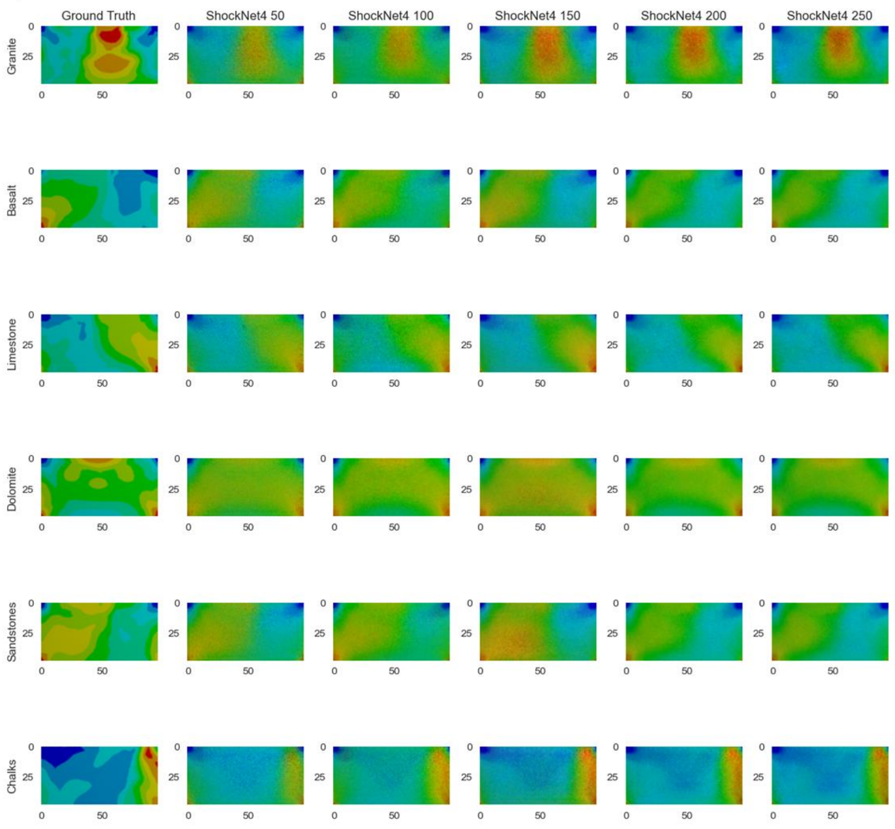


Figure 9: Visual comparisons between predicted maximum stress field and the ground truth result (first column). 3-SL (a) and 4-SL (b) ShockNets were used to generate the predictions. Rows represent different material samples, whereas the last five columns refer to the different number of epochs of which 3-SL and ShockNet4s had been trained.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

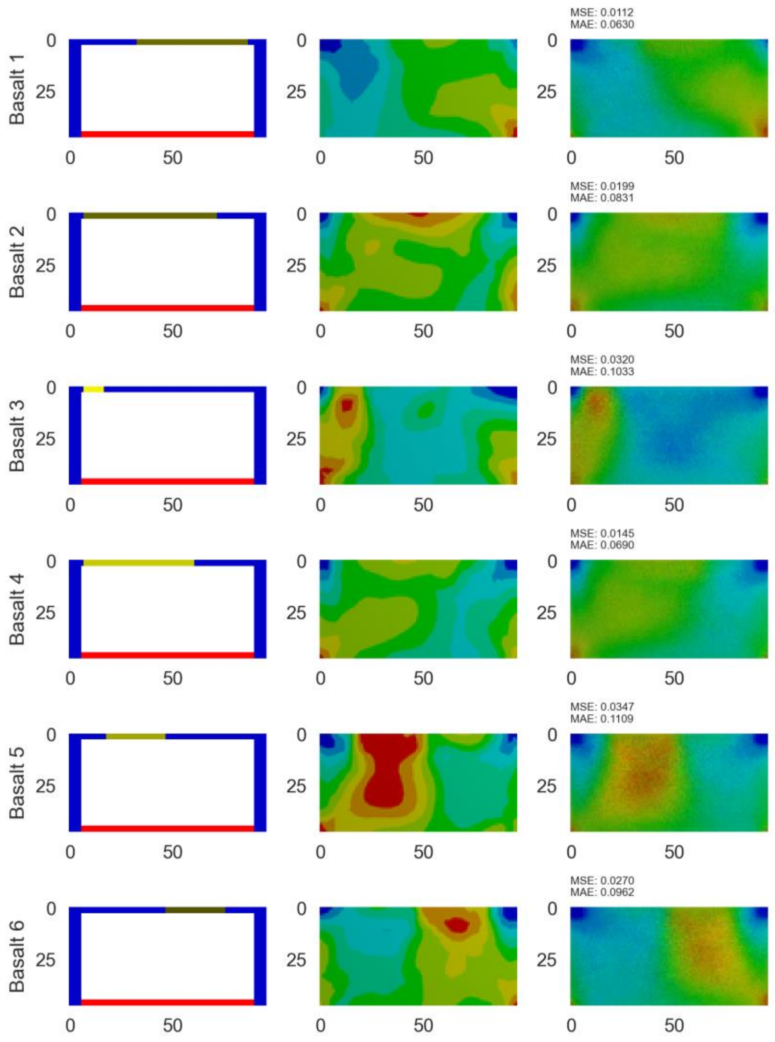


Figure 10: Results generated from ShockNet4-norm where corresponding initial conditions and ground truths are presented. MSE and MAE are calculated. Basalt was selected as the sample material.

3.3. Transferred Learning Capabilities

Transferred learning showed that the ability of ShockNet to generalise what has been learnt to a new dataset of different resolution. Experiments on the performance of transferred learning were shown in Figure 11. Unlike ShockNet4, transferred 4-SL ShockNet (ShockNet4-transferred) did not display the second time drop in losses The oscillation pattern of validation losses, however, showed a more stable convergence trend towards training losses. Although only trained with less than 100 epochs, ShockNet4-transferred reached approximately the same training and validation losses as ShockNet4 trained for 250 epochs. This showed the promising capability of ShockNet4 to be applied on high-resolution images when it had only been trained with coarse datasets.

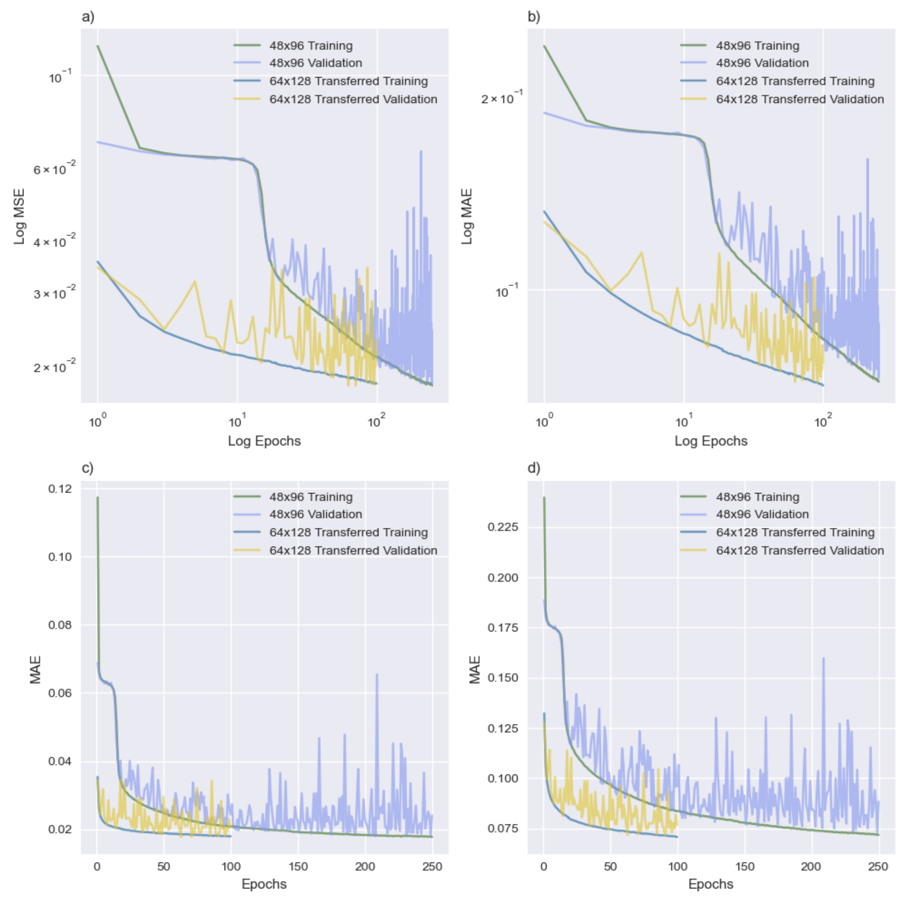


Figure 11: Comparison between ordinary learning using 48x96 dataset and transferred learning using 64x128 dataset. Parameters were inherited from the ShockNet4 after trained for 250 epochs with 48x96 dataset.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

3.4. Maximum stress predictions

TShockNet was developed to enable ShockNet to learn from both the geometric distribution of stresses and the numeric value of maximum and minimum stresses. TShockNet, however, took ~4 hours to finish a training & validating cycle on NVIDA’s Tesla P100-PCIE-16GB, and therefore only ten epochs had been trained. The convergence trend was present for all three criteria but required further experiments (Figure 12).

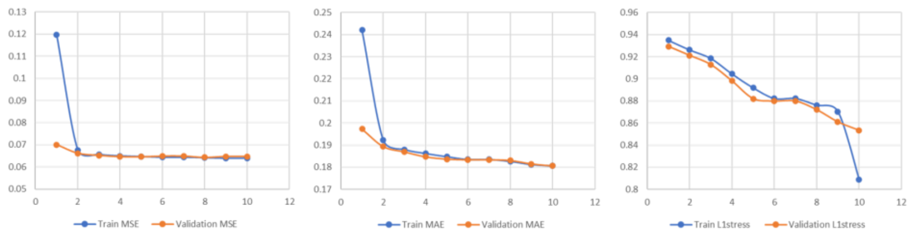


Figure 12: TShockNet training results. L1Stress is the L1 loss of the numerical stress exponent.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

4. Conclusions

In this project, ShockNet, an encoder-decoder-based image generation architecture, had been built to explore its potential to replace numerical shock stress predictions. NDs and BDs were generated for training, where NDs could improve the learning capability of ShockNet as it. Results showed that, for ShockNet architecture, KNI worked as the best initialisation method. The effect of depth of ShockNets was studied by comparing two similarly structured architectures with different SL numbers. Comparing to ShockNet4, ShockNet3 showed faster convergence and comparable prediction qualities which might be more practical to apply to real-world problems.

ShockNets were able to predict results that depicted the general contour of the stress field, which was a less time-consuming way comparing to ANSYS-SE. The resolution of predicted images became clearer when trained with more epochs. Noises had been substantially suppressed but still existed. One reason for this was the insufficient training epochs. From previous successful studies, about 1,000-20,000 epochs were trained to achieve relatively low losses and visually satisfying qualities (Zhu et al., 2017; Nie et al., 2019), about 10^1 - 10^2 times larger than what had been done in this project.

Transferred learning was applied on another dataset of higher resolution, and the result showed the promising potential of ShockNet to be generalised and applied on datasets of various resolutions. Stress prediction architecture was implemented, but the training time was limited.

Though ShockNet has shown its primary ability and value to be further studied as an alternative way to analyse shock wave-induced stress distribution, further efforts are required to improve the performance of it. Future work could focus on the following directions: further training on ShockNet3 and ShockNet4 to explore the actual stop point of the training process (where the validation loss soars); continuing the transferred learning process for higher resolutions to examine the generalisation capabilities; more experiments on TShockNet to explore the accuracy and performance of it to predict actual stress range.

GitHub repository: https://github.com/acse-2019/irp-acse-hl1319

5. Bibliography