CISC 867 Reproducibility Report: Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres

Ahmed Khaled[1], **Zhimu Guo**[2]
Department of Physics, Engineering Physics, and Astronomy
Queen's University
Kingston, ON K7L 3N6
[1]20ak41@queensu.ca, [2]15zg11@queensu.ca

Reproducibility Summary

Scope of Reproducibility

The scope is to verify all the six main claims made in the original paper: **The** physics of a multi-mode fibre can be mimicked by a complex-valued neural network to be able to restore images distorted during transmission through it, and their models can achieve average mean-squared error (MSE) as low as 960 over the dataset they presented, which they claim to be of equivalent performance to real-world optic devices in image restoration; **The** complex-weighted models trained faster with better accuracy than real-weighted ones; **Different** regularisation methods yielded minimal differences in MSE and minimal visual difference on image restoration; **Training** using perceptually-motivated structural similarity index measure based (SSIM) cost function did not improve the models performance noticeably; **Larger** speckle resolution improved the quality of the reconstructed image; **The** complex models generalized well to new datasets that were not included in the training datasets.

Methodology

We accessed and debugged the authors' code that used Keras and TensorFlow to understand the authors models better. Then, we created the entire pipeline of all of their models and experiments using PyTorch including building all the custom layers needed, and performed experiments to test their claims. We ran extra experiments with additional hyperparameter values to go beyond the original results. We used a single GPU of an NVIDIA RTX 3080 Ti running for 6 hours to complete all the original experiments, and about 19 extra hours to complete the additional experiments.

Results

We verified the six claims of the original paper, with minor comments. **The** complex-valued neural network models gave excellent visual performance and we got average testing MSE values that are within 3.6% of the reported values, and the results became almost similar (within 1%) when increasing the number of epochs beyond that of the authors'. **Complex-valued** models also gave better performance and converged faster than the real-valued ones. **Different** regularization methods had minimal visual difference on the results and only 0.37% difference on the MSE values, and **training** using SSIM-based loss function instead of MSE gave similar results. **Larger** speckle resolutions enhanced the performance of our models. **The** models were found to generalize well to new datasets not seen during training.

What was easy

The complex-valued neural network models were simple to understand, and we were able to recreate the exact same models using PyTorch. The authors' code had enough details on the models they implemented and they provided the dataset they used. Training and testing the model with all the trials and comparisons did not take much time to complete.

What was difficult

The authors proposed a fancy big pipeline but then actually implemented a much simpler one, which caused some confusion. In addition, the authors code was not well commented nor well organized. Moreover, they used many custom functions and custom layers for their complex-valued neural network models and some exclusive built-in functions from Keras. Therefore, we had to look at Keras API code to recreate those from ground up.

Preprint. Under review.

1 Introduction

This reproducibility report will be focusing on reproducing the published results in the paper "Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres" by Moran et al. [2018]. Multimode fibers can not be used alone to relay images, because optical modes mix strongly, and the images come out distorted at the receiver end. An example of this distortion is presented in Figure 1. Authors of this paper implemented complex-valued neural network models to reconstruct images from the speckle (distorted) images generated by a multimode fiber to replace the classical restoration methods. The classical ways includes using additional optical devices, which becomes the limitation for deploying multimode fibers for imaging applications, such as medical imaging.

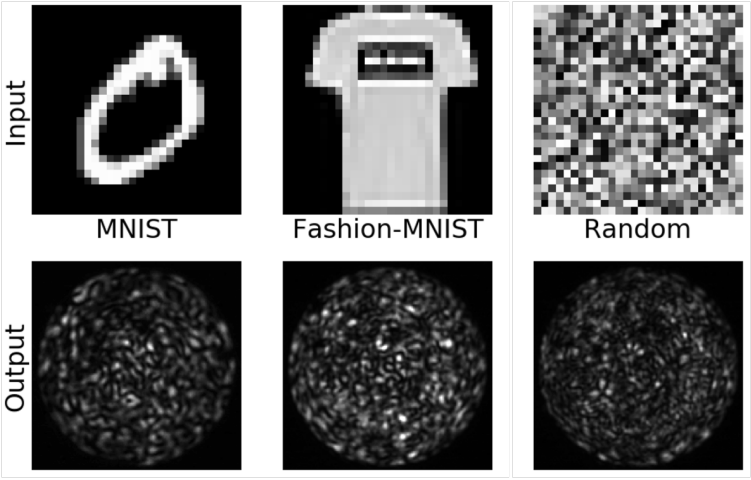


Figure 1: Examples of input and output from multi-mode fiber. We can see that it distorts images into speckles.

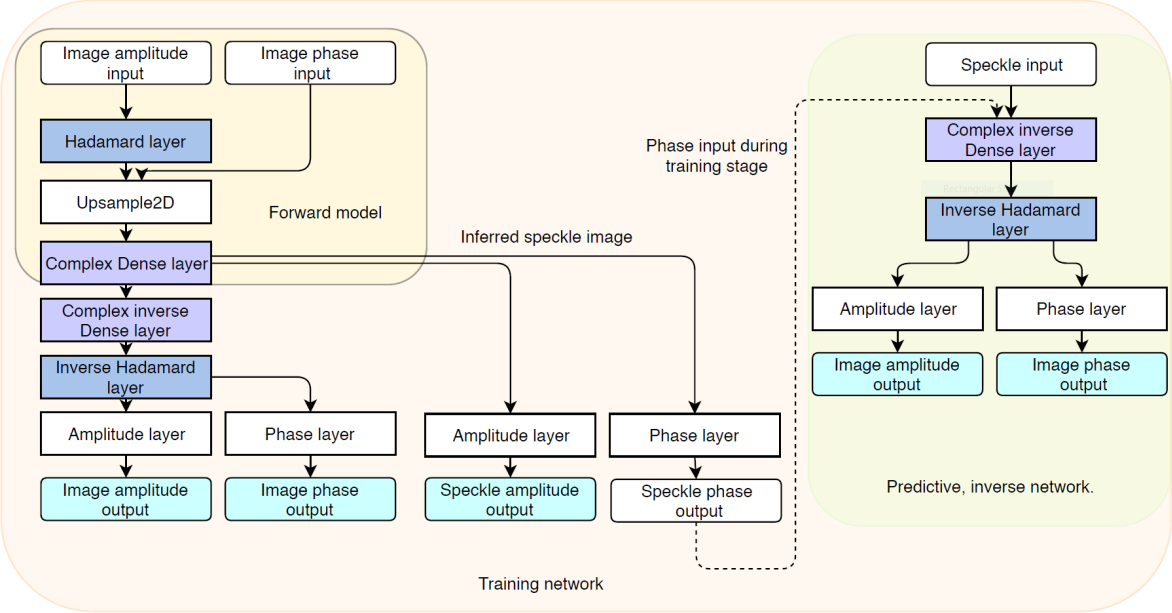


Figure 2: Model structure of the forward and inverse model. Cyan outputs are those for which we have target training data. Coloured layers indicate tied weights between associated forward and inverse layers. During training, inferred speckle phases from the forward model can potentially be used to augment training via the inverse model. Controlling image phase input (much easier than sensing phase output) can also augment learning in this approach.

The reason for using complex-valued neural networks comes from the physics of light propagation in multimode fiber; the fiber can be represented as a complex unitary transmission matrix. The complex-valued neural networks took complex inputs, and separated the real and imaginary parts of the complex numbers into two channels, each operating as if the numbers were purely real. The pipeline proposed by the authors is presented in Figure 2. It contains two models connected together in an autoencoder-like structure, a forward model that focused on simulating the distortion applied by the multimode fiber on the input images to generate the speckle images, and an inverse model to recover the images, that can be trained in conjunction with the forward model. Alternatively, you can ditch that autoencoder-like pipeline and just train the inverse model directly to recover images from distorted images. We discovered later that the authors opted for the later option, and they only implemented variants of the inverse model and trained it directly on pairs of input and speckle images, just like training any feedforward neural network with many layers, except that in this case, they included complex-valued dense layers and hadamard multiplication layers.

2 Scope of reproducibility

The scope of this reproducibility report is to verify all the claims made in the original paper:

Claim 1 The physics of a multi-mode fibre can be mimicked by a complex-valued neural network to be able to restore images distorted during transmission through it, and their models can achieve MSE as low as 960 over the dataset they presented (shown in Table 1), which they claim to be of equivalent performance to real-world optic devices in image restoration;

Claim 2 The complex-weighted model trained faster and achieved better MSE than real-weighted ones;

Claim 3 Different regularisation methods yielded minimal differences in MSE and minimal visual difference;

Claim 4 Training using perceptually-motivated SSIM-based cost function did not improve the models performance noticeably;

Claim 5 Larger speckle resolutions improved the quality of the final estimate of the reconstructed image in terms of MSE;

Claim 6 The complex models generalized well to new datasets that were not included in the training datasets.

Model		
Name	Regularization	MSE
Real-valued	l_2 weight regularisation	1034.62
Real-valued	None	1025.96
Complex	Unitary regularisation	989.28
Complex	Multiscale no regularisation	988.10
Complex	l_2 weight regularisation	962.33
Complex	None	960.31

Table 1: Model comparisons where each single layer model with 19,669,776 parameters (9,835,280 for real-valued models) was trained for 300 epochs, or until convergence, on a speckle resolution of 112×112, with regularization constant of 0.03.

As an important note, the pipeline proposed by the authors contained an inverse model and a forward model (shown in Figure 2), but they only included the inverse model in their implementation and their results in the paper. They mentioned that the forward model could be included to achieve better results, but we could not find neither its implementation in their code nor its results in the paper. They tested different versions of the inverse model with both real and complex-valued layers, as well as different regularization methods and cost functions. To verify and reproduce the results claimed in this paper, the scope of our reproducibility report included building the networks with the inverse model as described by the authors to include different layer types (complex and real), regularization methods, cost functions, speckle image resolutions, and with additional new datasets that were not seen during training.

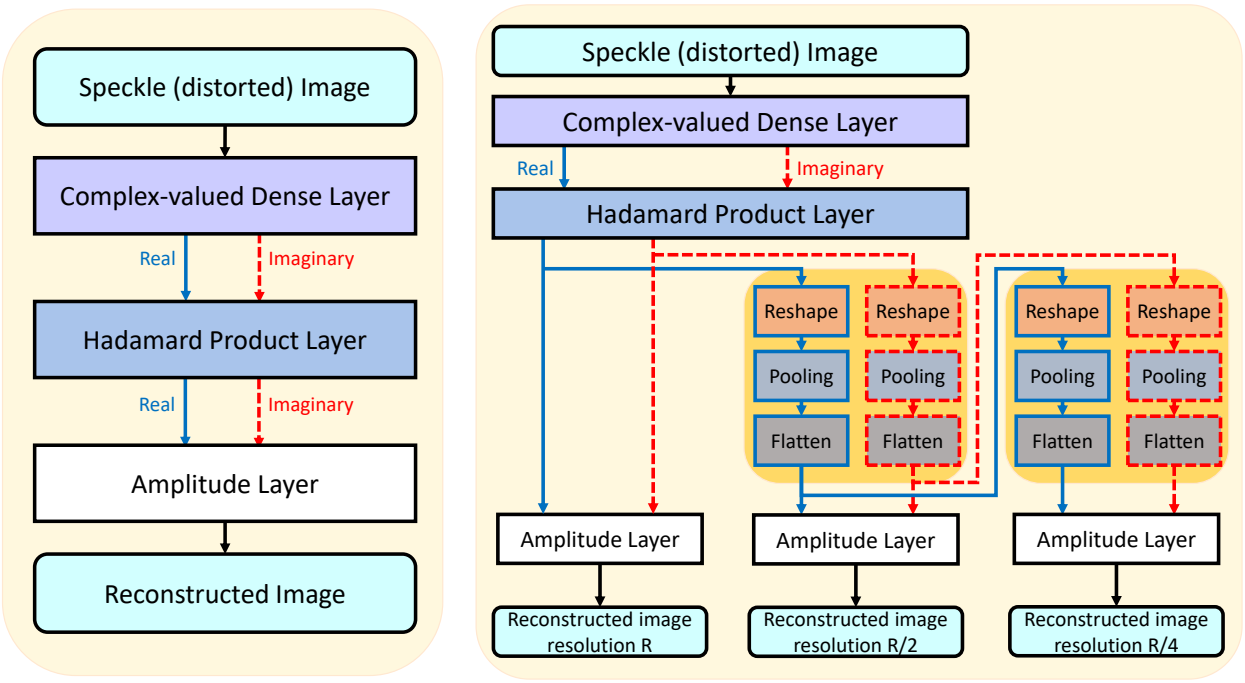
3 Methodology

We inspected the code published by the authors to better understand their implementation of the models. The original code used Keras and TensorFlow, and it had errors that required some debugging. Then, we created the entire pipeline of all of their models and experiments using PyTorch including building all the necessary custom layers. Then we used our models to perform all the experiments needed to test the claims made in the paper.

3.1 Model descriptions

As mentioned before, the authors' proposed pipeline, shown in Figure 2, contains two models connected together in an autoencoder-like structure; a forward model that focused on simulating the distortion applied by the multimode fiber on the input images to generate the speckle images, and an inverse model to recover the images, that can be trained in conjunction with the forward model. Alternatively, you can just train the inverse model directly with speckle images as its inputs and reconstructed images as its outputs. The authors opted for the latter option, and they only included the inverse model in their code implementation and their results in the paper. Thus, that was what we were building, verifying, and using for testing their claims.

The models we implemented had the pipelines shown in Figure 3. Both pipelines were inverse pipelines as we mentioned, and they were trained by flattened examples of speckle (distorted) images and the corresponding ground truth of the reconstructed image. In inference, we inputted the flattened speckle image to the model and it gave us the flattened reconstructed image then we reshaped it for viewing. Even though the model was simple, it worked well because it mimicked the physics of the multi-mode fibre by including a complex-valued dense layer that reflected the nature of the transmission matrix of the fibre (which is complex), and by including a hadamard product layer that mimicked laser amplitude drop effects. However, the authors only used linear activation in their models in their code, despite the fact that they mentioned in the paper that they used ReLU. Nonetheless, we included some trials with ReLU activation in Sec 4.2 to explore if it enhances the performance.



(a) (b)

Figure 3: (a) Pipeline A: The pipeline of the models that train with single resolution of the reconstructed image. (b) Pipeline B: The pipeline of the models that train with multiple resolutions of the reconstructed image.

The pipeline in Figure 3(a) (pipeline A) was trained with single resolution of the reconstructed image. However, the pipeline in Figure 3(b) (pipeline B) was trained with three resolutions: the original one, half the original one, and quarter the original one, and the cost function of this pipeline during training was the combined MSE across the three resolutions. However, in the inference/test stage, we only looked at the reconstructed image with the original resolution, as this was the intended task of the network. The pooling layer in this model was an average pooling with a window size of two pixels and a stride of two pixels, so each pooling layer halved the resolution of the image inputted to it.

The exact models we built were based on these two pipelines is shown in Table 2. They were exactly the same ones as the authors built, and included all authors' models.

Model No.	Pipeline	Dense layer type	Regularization	Training loss	No. of parameters
1	A	Real-valued	None	MSE	9,835,280
2	A	Real-valued	l_2	MSE	9,835,280
3	A	Complex-valued	None	MSE	19,669,776
4	A	Complex-valued	l_2	MSE	19,669,776
5	A	Complex-valued	l_2	SSIM-Based	19,669,776
6	A	Complex-valued	Unitary	MSE	19,669,776
7	B	Complex-valued	None	MSE	19,669,776

Table 2: Details of the models built.

3.2 Datasets

The datasets used for this reproducibility report were the same two datasets as used by the authors of the original paper. The first dataset was used for training and testing of all of our models, and each training example of this dataset consisted of a speckle (distorted) image and the corresponding image that was input to the fiber (i.e. the ground truth of the reconstructed image). The input images to the fiber were taken from MNIST, fashion-MNIST and random-pixels datasets. After training, the task was reconstructing the original image (MNIST, fashion-MNIST or random-pixels image) given the speckle image. Examples of the dataset are shown in Figure 1. The training set included 44000 training examples, and the testing set included 6000 examples. Each input image has 28×28 pixels with amplitudes ranging from 0 to 255, and each speckle image had 112×112 pixels with amplitudes ranging from 0 to 255. The speckle images originally were at 224×224 resolution, but the authors scaled them down to 112×112 using average pooling for faster training speed and lower memory capacity requirement.

The second dataset was only used for testing, and it consisted of some images of Muybridge's historic stop-motion films (Muybridge [1955 - 1887]). This dataset was not included in the training, and the purpose was to verify that the network could generalize to unknown datasets, meaning that the network truly represents the inverse of the fiber transmission matrix.

3.3 Hyperparameters

The hyperparameters used by the authors of the original paper were extracted from the code they provided, and they are summarized in Table 3. We used the exact same hyperparameter values to test their claims in the original paper. We also performed extra experiments for different hyperparameter values to improve the results beyond those they reported.

Name	Value
Number of hidden layers	1
Number of neurons in hidden layer	784
Number of epochs	300 or until convergence
Optimizer	SGD
Learning rate	10^{-5}
Batch size	64
Regularization constant	0.03

Table 3: The hyperparameters the authors used in their study.

3.4 Experimental setup and code

We performed three experiments to verify all of the six claims of the authors.

3.4.1 Test Setup 1

In this setup, we built and trained the models in Table 2 then we tested them using the testing portion of the training dataset (the first dataset described in Sec 3.2) and reported the average MSE over the testing portion of the training dataset. The resolution of the speckle images for all the experiments was 112×112. We also used the trained model to reconstruct some example speckle images to be judged visually. This setup was used to verify **claims 1 to 4**, which were verifying the reported performance of the authors' models (in terms of MSE), verifying the fact the complex-weighted models train faster and give better results than the real-weighted ones, different regularization methods provided

minimal difference on MSE and minimum visual difference on image reconstruction, and changing the loss function to be based on SSIM did not improve the performance noticeably.

3.4.2 Test Setup 2

In this setup, we focused on Model No. 4 from Table 2 (complex-valued model with l_2 regularization) and trained/tested it with five different speckle image resolutions: 224×224, 112×112, 56×56, 28×28, and 14×14. These different resolutions were obtained by average pooling of the speckle images of the dataset. This setup was used to verify **claim 5** which says that larger speckle resolutions improved the quality of the reconstructed image in terms of MSE. The number of model parameters changes with the resolution of the speckle image, because the size of dense layer depends on it. The number of parameters in Table 2 was calculated for 112×112 resolution.

3.4.3 Test Setup 3

In this setup, we used the pre-trained Model No. 4 from Table 2 (complex-valued model with l_2 regularization) to reconstruct images from an additional dataset that was not used in training (Muybridge's historic stop-motion dataset described in Sec 3.2). The images in that dataset were actually from a movie (GIF image). The input pictures to the fiber (original images), speckle images, and reconstructed images were assembled into short movies (GIF images), and these can be seen in the front page of our GitHub repo[1]. This setup was used to test **claim 6**, which says that the complex models generalized well to new datasets that were not included in the training datasets. The authors evaluated the reconstructed movie based on its visual presentation, but did not provide MSE values. Therefore, we also used the same visual evaluation method for our recreated model. It is to be noted that the authors' included experiments for a fiber length of 1 meter and 10 meter, but only they only provided the data for 1 meter. However, the 1 meter data was enough to validate their claim.

3.5 Computational requirements

The authors trained their models using an Nvidia Titan X GPU within several hours, without mentioning the exact number. Therefore, we also used a single GPU (NVIDIA RTX 3080 Ti), a single CPU (Ryzen 9 5900X), as well as 32 GB of system memory which was important to support the dataset with larger speckle resolutions.

It took us 3 hours to run **Test Setup 1**, which included training/testing all the models in Table2 for 300 epochs. Then, it took us another three hours to run **Test Setup 2**, which included training/testing Model No. 4 from Table 2 (complex-valued model with l_2 regularization) for five different speckle image resolutions. Finally, it took us very negligible time to run **Test Setup 3** as there is no training involved, it was just using Model No. 4 (inference) from Table 2 (complex-valued model with l_2 regularization) to reconstruct images from an additional dataset that was not used in training (Muybridge's historic stop-motion dataset described in Sec 3.2). Finally, it took us 19 hours for the additional trials such as different number of epochs and different activation function.

4 Results

4.1 Results reproducing original paper

4.1.1 Results of Test Setup 1

In this section, we ran the first test setup in Sec3.4.1, where we built and trained the models in Table 2. Then we tested them using the testing portion of the training dataset (the first dataset described in Sec 3.2) and reported the average MSE over the testing portion of the training dataset. This setup was used to verify **claims 1 to 4**, which were verifying the reported performance of the authors' models (in terms of MSE), the fact that the complex-weighted models train faster and give better results than the real-weighted ones, the fact that different regularization methods provided minimal difference on MSE and minimum visual difference on image reconstruction, and the fact that changing the loss function to be based on SSIM did not improve the performance noticeably.

To verify **claim 1** and the part of **claim 2** that stated that the complex-weighted models gave better results than the real-weighted ones, our final MSE results from the testing portion of the dataset was reported in Table 4 and compared to those of the authors. In addition, we showed some testing examples for all the complex-valued model in Figure 4. Visually speaking, the performance was excellent and we can reconstruct the distorted image. As for the numerical results, we can see that complex-weighted models out performed real-weighted ones, verifying that part of **claim 2**.

[1] `https://github.com/ZhimuG/CISC867`

We can also see that their results at 300 epochs were close to ours at 300 epochs; but their MSE loss was 3.6% less than ours on average. 3.6% testing accuracy differences of a classification model is a big significant number, but 3.6% difference in MSE is a small difference to the point that the difference in reconstructed images was not distinguishable by the human eye. Therefore, this was an acceptable difference and we can consider **claim 1** to be verified. However, this intrigued us to train the models further, so we trained for another 300 epochs (so now the models are trained for 600 epochs), and as we can see in Table 2, the author's results and our results at 600 epochs were almost identical, the average absolute difference was less than 1%. This indicated that the authors may have trained for 600 epochs but mistakenly reported 300 epochs instead. Finally, we also trained the models for 1000 epochs, and we obtained better numbers than they claimed in the paper as shown in Table 2, which hinted at the fact that they did not realize the best performance of their models.

Model No.	Pipeline	Dense layer type	Regularization	Author's MSE (300 epochs)	Our MSE (300, 600, 1000 epochs)
1	A	Real-valued	None	1025.96	1063.84, 1027.31, 995.55
2	A	Real-valued	l_2	1034.62	1065.78, 1024.71, 995.49
3	A	Complex-valued	None	960.31	1015.38, 962.51, 928.98
4	A	Complex-valued	l_2	962.33	1018.81, 964.39, 926.43
5	A	Complex-valued, SSIM	l_2	-	954.73, 896.03, 855.98
6	A	Complex-valued	Unitary	989.28	1017.23, 966.95, 925.64
7	B	Complex-valued	None	988.1	1004.61, 969.83, 938.27

Table 4: Our MSE results at 300, 600, and 1000 epochs compared to those of the authors at 300 epochs. All the models have MSE training loss except for Model No. 5 having SSIM-based loss, and the authors did not report MSE for it and just compared it to the other models qualitatively.

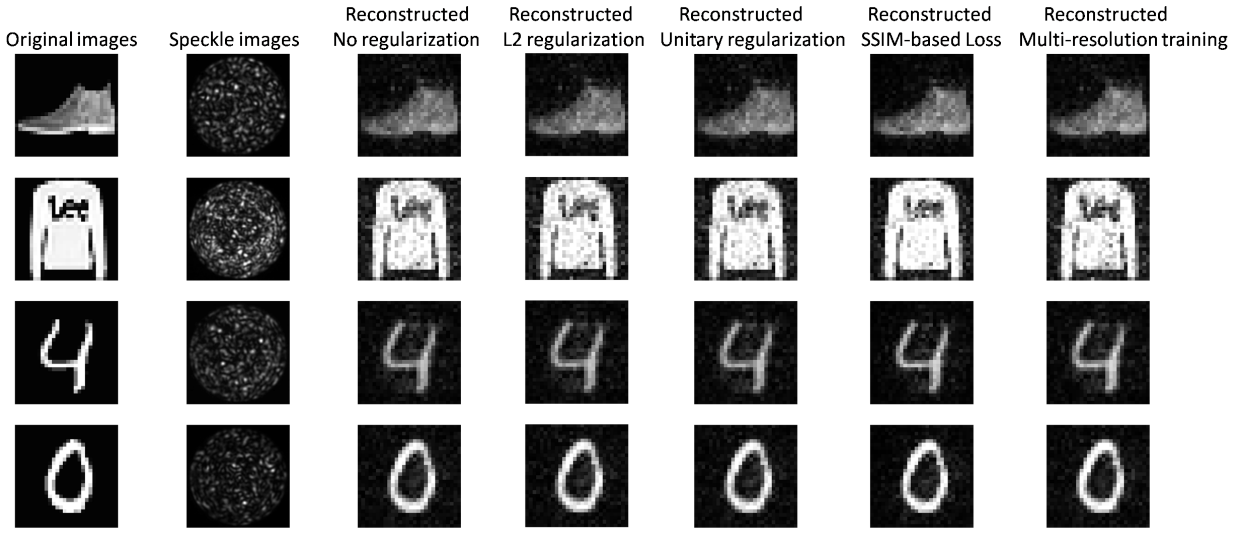


Figure 4: Reconstructed images using the complex-valued models. As we can see, all the models give excellent performance and the slight difference in their testing MSE isn't really distinguishable visually.

To verify the part of **claim 2** related to the convergence speed of the complex-weighted models compared to the real-weighted ones, the MSE over the training loss was reported for different epochs in Table 5. As shown here, the complex-valued models consistently learn faster than the real-valued one, thus verifying their claim.

As for **claim 3 and 4**, they can also be verified directly from Table 4 and Figure 4. As for **claim 3**, the complex-valued models (with MSE loss) that used different regularization (Models No. 3, 4 and 6 from Table 2) gave close MSE results, and the greatest difference was between no regularization and l_2 regularization, and it equaled 0.37% which was negligible. In addition, we saw no visual difference by looking at their results in Figure 4. Same exact observations applied for **claim 4** about the differences between using SSIM-based loss (Model no. 5) and using MSE loss while keeping the model structure constant (Model no. 4), thus verifying these two claims of the authors.

Epoch	Model 1 loss (real)	Model 2 loss (real)	Model 3 loss (complex)	Model 4 loss (complex)	Model 6 loss (complex)	Model 7 loss (complex)
50	1229.23	1230.83	1222.42	1226.4	1226.22	1151.02
100	1134.47	1139.19	1111.70	1114.42	1115.01	1075.69
150	1104	1107.95	1071.89	1073.59	1073.79	1049.31
200	1088.77	1086.53	1047.21	1048.9	1047.61	1032.06
250	1074.02	1074.35	1028.52	1030.70	1031.92	1015.98
300	1063.47	1065.41	1015.21	1019.02	1017.44	1004.83

Table 5: Average MSE over training portion of the dataset for different models at different epochs; we found out that the complex-valued models consistently learn faster. Model No.5 is not included in this table because it has SSIM-based training loss, not MSE like the rest of the models, so we can not compare its training loss to those of others models.

4.1.2 Results of Test Setup 2

In this section, we ran the second setup described in Sec 3.4.2. In this setup, we trained/tested Model No. 4 from Table 2 (complex-valued model with l_2 regularization) five different speckle image resolutions: 224×224, 112×112, 56×56, 28×28, and 14×14 and reported MSE from the testing portion dataset for all resolutions. This setup was used to verify **claim 5** which says that larger speckle resolutions improved the quality of the reconstructed image in terms of MSE. The MSE results were reported in Table 6, and visual examples were shown in Figure 5. As we expected, decreasing the speckle resolution had detrimental impact on the quality of the reconstructed image and increased the testing MSE significantly, thus verifying their claim.

Speckle resolution used in training	MSE
224×224	913.3
112×112	1017
56×56	1229.8
28×28	1941.8
14×14	3229

Table 6: Average MSE over testing portion of the dataset when Model No. 4 (complex-valued with l_2 regularization) was trained with different speckle image resolution.

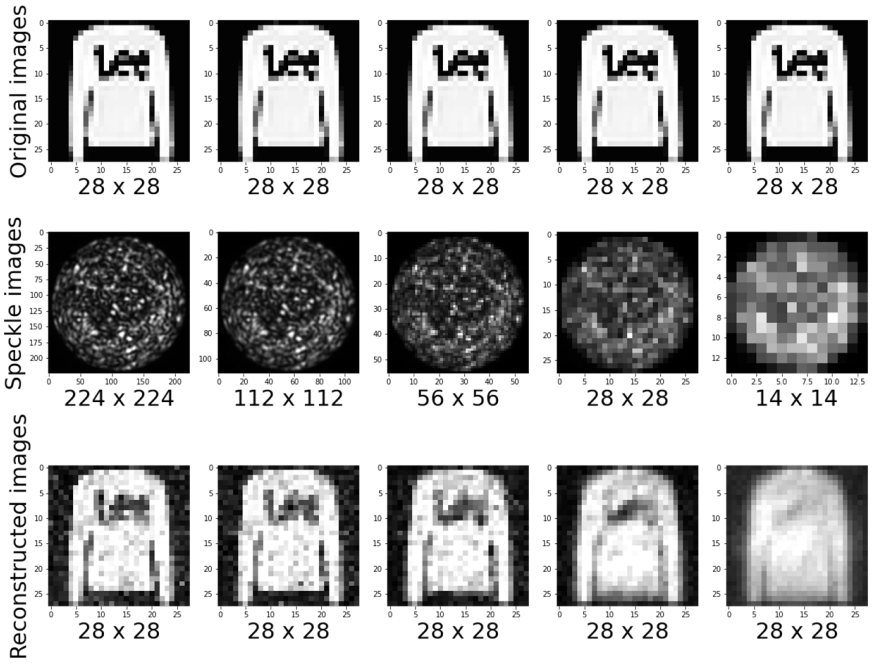


Figure 5: Reconstructing an image using the model but trained with five different speckle resolutions.

4.1.3 Results of Test Setup 3

In this section, we ran the third setup described in Sec 3.4.3. In this setup, we used the pre-trained Model No. 4 from Table 2 (complex-valued model with l_2 regularization) to reconstruct images from an additional dataset that was not used in training (Muybridge's historic stop-motion dataset described in Sec 3.2). This setup was used to test **claim 6**, which says that the complex models generalized well to new datasets that were not included in the training datasets.

The images in that dataset were actually just a few images that compose a movie (GIF image). The input pictures to the fiber (original images), the speckle (distorted) images, and the reconstructed images were all assembled into short movies (GIF images), and these can be seen in the front page of our GitHub repo. Example shots from that short movie is shown in Figure 6. However, they did not mention any quantitative measure for this claim, instead they evaluated the results visually. Therefore, we also verified their claim using the same method. As shown here, we can clearly see that the reconstructed movie had very good visual quality, and this verified their claim.

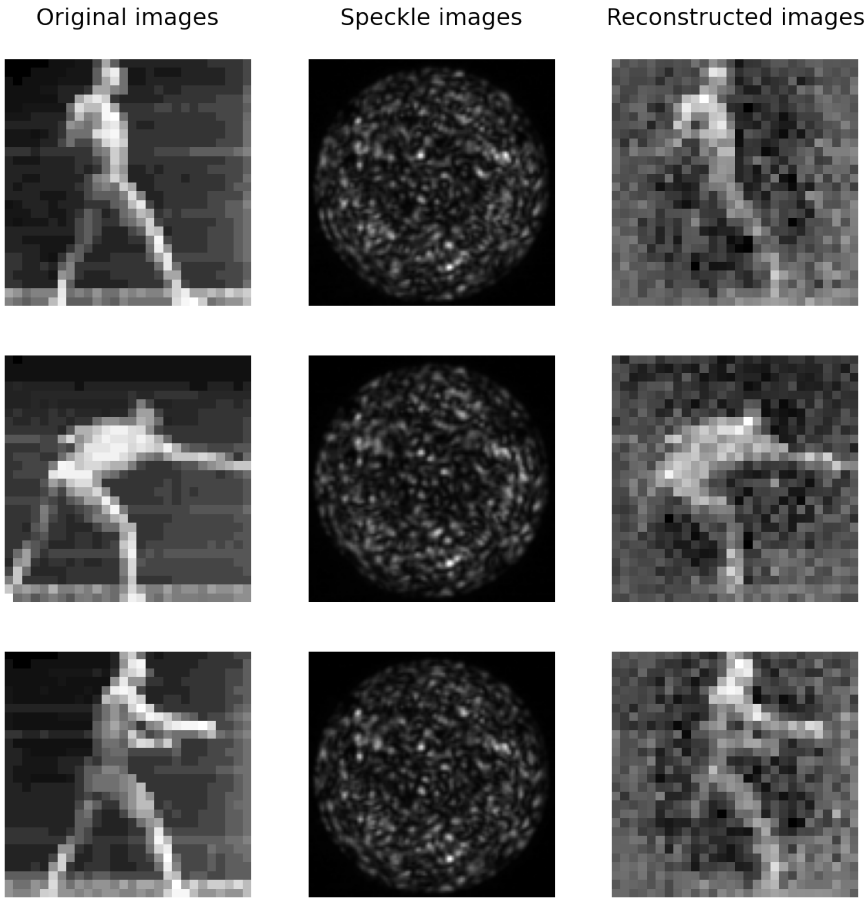


Figure 6: Screenshots from the movie (taken from Muybridge's historic stop-motion dataset), speckle (distorted) movie and reconstructed movie. The movie is in the front page of our GitHub repo.

4.2 Results beyond original paper

We noticed that the authors did not train their models with enough epochs (only 300 epochs), and they only used linear activation. So, we tried **three extra experiments**: **Experiment 1**, which was increasing the number of epochs to 1000 instead of 300 while keeping the linear activation, **Experiment 2**, which was using ReLU activation while keeping the number of epochs at 300 like the authors did, and **Experiment 3**, which was using ReLU activation while increasing the number of epochs to 1000 instead of 300. The results are reported in Table 7. For the first experiment, increasing the number of epochs to 1000 up from 300 while keeping the same linear activation enhanced the average of MSE of the models by 4.2% over the authors' results. For the second experiment, using ReLU activation while keeping the number of epochs at 300 enhanced the average of MSE of the models by 8.12% over the authors' results. The highest improvement in this experiment was for Model no. 7, and the MSE improvement was a significant 15.47%. For the

third experiment, using ReLU activation while increasing the number of epochs to 1000 instead of 300 enhanced the average of MSE of the models by 13.49% over the authors' results. The highest improvement in this experiment was for Model no. 6, and the MSE improvement was a significant 20.47%

Model No.	Authors' MSE	Experiment 1 MSE	Experiment 2 MSE	Experiment 3 MSE
1	1025.96	995.55	1094.6	1036.9
2	1034.62	995.49	913.6	1004.26
3	960.31	928.98	878.77	779.34
4	962.33	926.43	875.96	773.89
5	-	855.98	858.562	773.89
6	989.28	925.64	879.1	786.77
7	988.1	938.27	835.29	788.87

Table 7: MSE results of our models (model description is in Table 2) for the extra experiments versus those of the authors. Authors' MSE: at 300 epochs and linear activation, Experiment 1: at 1000 epochs and linear activation, Experiment 2: at 300 epochs and ReLU activation, and Experiment 3: at 1000 epochs and ReLU activation.

5 Discussion

As discussed in detail in the Results section, our experiments confirmed the claims of the original paper with two major high-level comments and some minor comments regarding the details of the claims. **The first major comment** was that the authors were self-contradictory when describing goals in their paper. They proposed a sophisticated pipeline that contained both forward and inverse models connected in an autoencoder-like structure (shown in Figure 2), but the final implementation was much simpler and only included the direct inverse models (shown in Figure 3). **The second major comment** was that they mentioned using ReLU as the activation function, but they only used linear activation in their code. We demonstrated in Sec 4.2 that we could achieve enhance MSE results up to 20.47% when using ReLU and 1000 epochs instead of 300, which is very significant. Therefore, this could be another error in the original paper.

The minor comments were related to their claims. For **claim 1**, we verified the faithful restoration of images visually as shown in Figure 4, and on average our MSE values (in Table 4) came within 3.6% of the claimed values in the paper after 300 epochs of training (they also used 300 epochs). **One minor comment** here was that the average difference between our results and their results became less than 1% when we increased the number of epochs to 600, and based on the almost identical numbers we believe that the authors may have mistakenly reported 300 epochs instead of 600. **Another minor comment** here was that we could improve the MSE performance by 4.2% on average if we trained for 1000 epochs instead of 300, and we could enhance the MSE performance by 8.12% on average when using ReLU activation instead of linear activation, and we could enhance the MSE performance by 15.47% on average when doing both (1000 epochs and ReLU activation).

As for **claim 2**, we confirmed that the complex-valued models provided better results than the real-valued ones (Table 4) and verified that they converge faster as well (Table 5).

As for **claim 3**, we confirmed that regularization methods had very minor effect on MSE results (0.37 % maximum) and yielded no visual difference on the results as seen in Figure 4. As for **claim 4**, same exact observations were made and confirmed that using SSIM-based training loss yielded no significant improvements on MSE or visual results.

As for **claim 5**, we confirmed that the performance was significantly improved with increasing speckle resolution. Finally, we confirmed **claim 6** that the model generalized well to new datasets not seen during training by reconstructing a movie, available on the front page of our GitHub repo, and the results were visually excellent (the authors had no numerical results for it as well).

The strengths of our reproducibility approach were: faithful recreation of the original complex and real-valued neural network models with all the discussed pipelines, using completely different machine learning framework, exactly same datasets, more sophisticated investigation into hyperparameter values, and better hardware for more extensive testing. In this reproducibility report, we recreated exactly the same complex and real-valued neural network models proposed by the authors of the paper, with the same hyperparameters and network structures. We also used PyTorch as the framework for our recreated model to code all the custom network layers and functions needed, whereas the original model was built using Keras and TensorFlow. We went through all the details of the original models, including network specifications, cost functions, regularizations, and all the custom layers and functions used in the original models. Taking advantage of the better hardware on hand, we also extended the trials and tests to include more hyperparameter values and different activation function to further validate the results and claims made in the paper and we even got beyond them.

We saw no apparent weakness to our approach, however minor critiques may come from how well we recreated all the custom layers and functions using a completely different framework. The authors took advantage of many built-in functions from Keras to handle data imports, cost functions, regularizations, and convergence checking and testing, many of which were not available in PyTorch. Therefore, it was necessary to dive into the Keras API and dig up these function definitions to recreate them in PyTorch. We tested those step by step and we believe that every aspect of the models, custom code, training and testing was faithfully recreated, and this was the only part that involved translation between different frameworks.

5.1 What was easy

First, accessing the code and datasets was easy, because the authors published their entire GitHub repository along with their datasets. The link to their repository was included in the paper, and on their repository page they included another link to a Dropbox folder that has the file for almost all the datasets. We were able to download the original code and datasets without an issue, and we were able to run the original code after installing all required packages and debugging it. The authors also provided sufficient description and diagrams to help us understand their complex-valued neural network models conceptually, and all the hyperparameters can be found in their code. After recreating the model in PyTorch, we were able to train and test our model and obtain similar results as stated in the paper. Training the models using the original dataset was simple and quick, we were able to finish all the trials and tests done by the authors within 6 hours. We used a single NVIDIA RTX 3080 Ti, and the authors mentioned in their code notebook that they used a single NVIDIA Titan X GPU. The authors did not mention how many hours were spent on training, however, with better hardware we have, we were able to extend the trials and tests to include more hyperparameter values and different activation function to further validate the results and claims made in the paper and even get beyond them.

5.2 What was difficult

The majority of the reproducibility of the paper went smoothly, but there were few challenges we faced. The authors also had a few contradictions between the goals stated in their paper and the actual implementation in their code as mentioned before in our report, where some of the goals listed in their paper, such as using an autoencoder-like pipeline with connected forward and inverse models, were not present in their models. They did not specify in their paper that they did not achieve those goals, nor did they provide any explanations why certain methods or goals were discarded. This caused several confusions, but we were able to sort things out after thoroughly reading through both their paper and their code.

In addition, the code published by the authors was not well commented, and its structure was not well organized either. Therefore, it took us extra time to carefully read through their code and find all the hyperparameter values they used. They also used many built-in functions from Keras, which were not available in PyTorch. Therefore, we had to go through Keras API and learn all the details of those functions they used to reimplement these functions to maintain the integrity of the recreation of their model. Finally, they did not provide the data of the 10 meter fiber for their third experiment discussed in Sec 3.4.3, instead they only provided the 1 meter one. However, the 1 meter data was enough to validate their claim in this experiment.

References

Oisín Moran, Piergiorgio Caramazza, Daniele Faccio, and Roderick Murray-Smith. Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/148510031349642de5ca0c544f31b2ef-Paper.pdf`.

Eadweard Muybridge. *The human figure in motion.* Dover Publications, New York, 1955 - 1887.