DeepWIFF: A Hybrid Machine Learning Parameterization of Wave-Induced Sea-Ice Floe Fracture

Meal Swipes for Late Days: Jacob DiChiacchio (jdichiac), Shreyas Rao (srao20), Tej Stead (tstead2)

GitHub Link: https://github.com/tejstead/deepwiff

Introduction

Sea ice plays an important role in the world's climate, covering ~12% of the ocean's surface. It is a major determinant of climate stability/instability, polar shipping routes, and it affects heat currents under the ocean's surface. In the past few decades, the rapidly changing climate has caused ice in the Arctic to thin out and disappear. Ice has become more of a seasonal occurrence in regions where it used to occur year round, and ice fractures have become far more common (Horvat and Tziperman, 2015). This has necessitated the development of more precise models to study sea ice-floe (ice sheets) fractures.

Unfortunately, the current sea-ice floe models are computationally expensive and can take days to run. The ice fracture process is related to the two-dimensional ocean wave height field, which is too computationally expensive to simulate in a climate model. Previously, a super-parametrized one-dimensional model (SP-WIFF) was used, but it is expensive to run as well (Horvat and Tziperman, 2015). Additionally, the SP-WIFF model has a stochastic component, so its results are not easily reproducible. To provide a more deterministic output, as well as reduce computation time, we use a classifier as well as a mixture density network to replicate the output of the SP-WIFF climate model when run to convergence.

Data

The data received from SP-WIFF is saved in the files "X_train.p", "X_test.p", "y_train.p", and "y_test.p", where the X_train and X_test correspond to the inputs in the SP-WIFF model, which are 26 dimensional vectors corresponding to input frequencies of forces that could cause fractures in ice floes. y_test and y_train correspond to the outputs of the SP-WIFF model. They are 49 dimensional vectors representing a discrete probability distribution of fracture sizes (scaled logarithmically).

Not all of the data is usable, since not all input frequencies are strong enough to create ice floe fractures. The SP-WIFF model automatically differentiates between usable and unusable data, and this process is learned by a 2 layer feed-forward neural network provided by Horvat with a 98% accuracy rate, deemed suitable enough for the model. Thus out of 1.8 million original inputs, only ~400 thousand inputs are deemed acceptable for the use of DeepWIFF, which are then saved in a 70%/30% test data/training data split to "X_train.p" and "X_test.p" respectively. The respective outputs of the usable inputs are saved to "y_train.p", and "y_test.p". This data is summarized in the table below.

Fig 1: DeepWIFF Input Data Summary

Total Inputs	Usable Inputs	Train Inputs (X_train)	Test Inputs (X_test)
1830900	404503	283152	121351

Model Architectures

A few different model architectures for DeepWIFF are tested, namely a feed-forward neural network and a mixture density network (MDN). Because the output of SP-WIFF is interpreted as a probability distribution, it is important for the models to produce valid discrete probability distributions in the output of their call functions. The feed-forward network acts as a control against which the results of the MDN are compared, and the specifics of each model are listed below.

Feed-Forward (Control Network)

The simplest model tested, which is also used as a control, is a multilayer feed-forward neural network (FFNN). Besides the inclusion of a LeakyReLU function between each of the 5 hidden layers, this control network is relatively simple and includes no surprise parts. A softmax layer is included at the end to ensure that the output is a valid probability distribution. The figures below represent the FFNN's architecture and important hyperparameters for training and testing.

Fig 2: Feed Forward Network Architecture

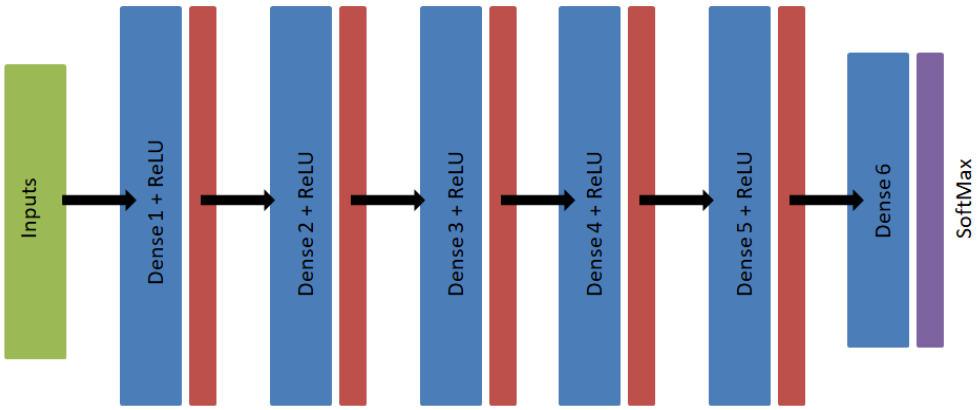

Fig 3: Feed Forward Network Parameters

Batch Size	256
Activation Function	Leaky ReLU
Learning Rate	.001
Hidden Layer Size	100
Epochs	100

Mixture Density Network

The main test network is a Mixture Density Network (MDN), a type of network that learns probability distributions. The justification for using this model is that SP-WIFF can essentially be interpreted as a model that outputs conditional probability distributions. Any type of deep neural network emulating this process is likely to benefit from also learning conditional probability distributions (Bishop, 1994).

The MDN produced consists of hidden dense layers and a custom MixtureDensity layer. After passing inputs through multiple dense layers with Leaky ReLU activations, the final dense layer output is passed through the MixtureDensity. Within the Mixture Density layer, DeepWIFF learns to approximate a probability distribution to match the SP-WIFF output using a sum of Gaussian (normal) distributions.

The sublayer within the MixtureDensity Layer that learns variances (sigma) for the Gaussian mixture is passed through a softplus activation function, to ensure that produced variances are always positive. The sublayer which learns the mixture parameters (alpha) is passed through a softmax layer, to ensure that the mixture ratios are positive and all sum to one. Because a sum of Gaussian distributions is a continuous distribution, a custom discretized function based on integer rounding and the distribution's cumulative distribution function (CDF) is implemented to allow the MDN to give data comparable to SP-WIFF's outputs. There is no need to softmax at the end of the network, as the discretized data is still a valid probability distribution. The two figures below document both the MDN's architecture and its important hyperparameters used in training and testing.

Fig 4: Mixture Density Network Architecture

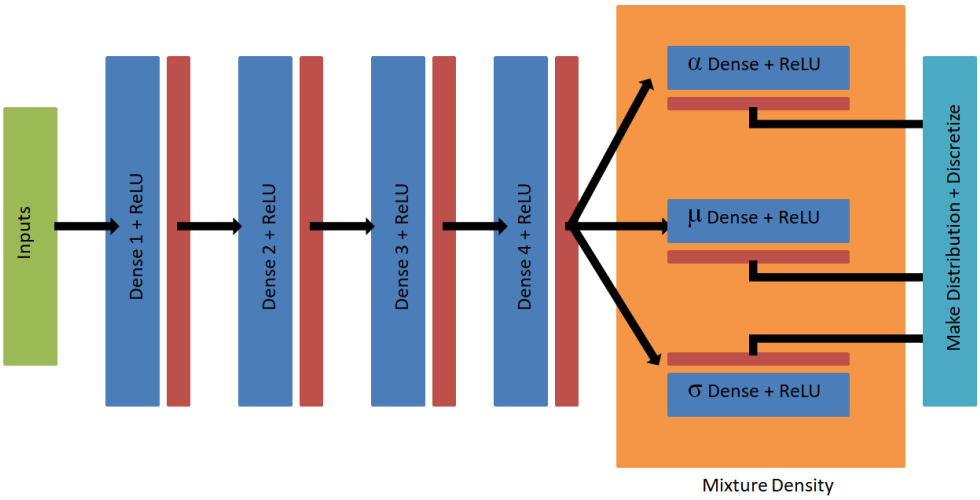

Fig 4: Mixture Density Network Parameters

Batch Size	256
Activation Function	Leaky ReLU
Learning Rate	.001
Mixture Type	Gaussian (Normal)
Mixture Components	12
Hidden Layer Size	100
Epochs	100

Loss

Because DeepWIFF is essentially a regression model, regression-type loss functions provide a more robust metric for measuring the network's performance. Following the idea that simpler is better, all the models employ mean squared error (MSE) as a loss function to train on. Although a few other loss functions, such as mean average error (MAE), log mean squared error (LMSE), and Kullback-Leibler divergence (KLD) were initially considered, MSE outperformed all other loss functions in testing. Based on these observations, and for the sake of standardization, it was deemed most effective to train the models using MSE. The formula for MSE is given below, where n represents the size of the output vector (distribution), y_i represents the actual ith value, and $\hat{y}_i$ represents the predicted ith value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Results

Because there are many different metrics for measuring the accuracy of each network's output, multiple different loss metrics are measured throughout the training. Three different loss functions are used: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Kullback-Leibler Divergence (KLD). Each loss is listed on the graphs and tables below in their abbreviated forms. The graphs below represent the validation loss (testing loss) measured at the end of each epoch. The first two tables also represent the exact measured training and validation losses on the final epoch of the model being run. To measure the number of outputs deemed close enough in valuation in the output SP-WIFF, the final table documents the ratio of validation data (testing data) within different percentages of the Squared Absolute Error (SAE).

Fig 4: MSE Validation Loss

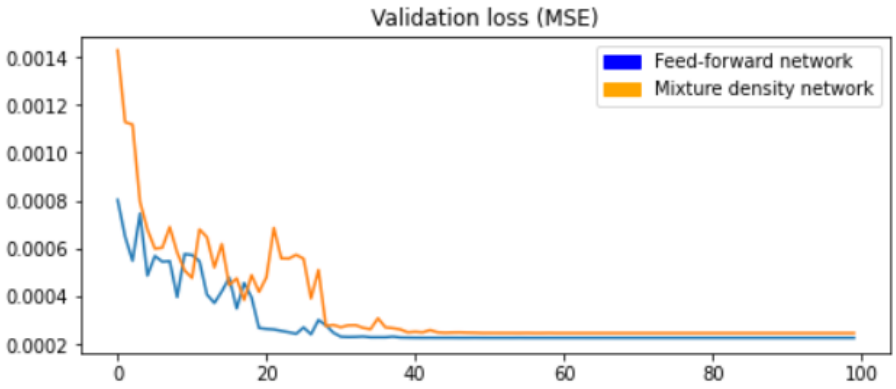

Fig 5: MAE Validation Loss

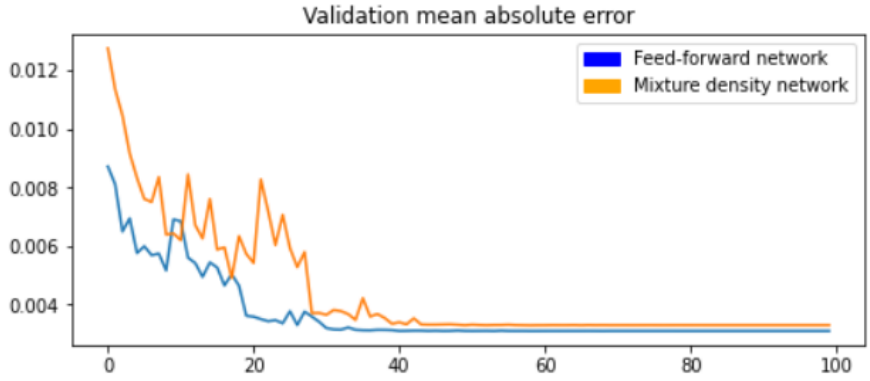

Fig 6: KLD Validation Loss

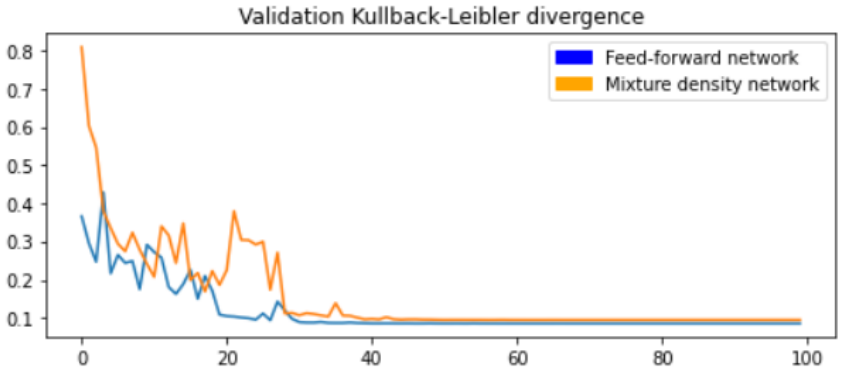

Fig 7: Training Losses Final Epoch

	Feed Forward Network	Mixture Density Network
KLD	.08596	.09559
MAE	3.083×10^{-3}	3.277×10^{-3}
MSE	2.372×10^{-4}	2.567×10^{-4}

Fig 8: Validation (Test) Losses Final Epoch

	Feed Forward Network	Mixture Density Network
KLD	.08552	.09477
MAE	3.092×10^{-3}	3.289×10^{-3}
MSE	2.265×10^{-4}	2.465×10^{-4}

Fig 9: Model Accuracy Based on SAE

	Feed Forward Network	Mixture Density Network
<25%	86.68%	84.46%
<10%	71.03%	65.03%
<5%	40.86%	32.72%

Discussion of Results

As the data points out, the control feed forward neural network outperformed the mixture density in every single metric, albeit just barely. Throughout training, both the feed forward network and mixture density network experience large fluctuations in loss within the first 30 epochs before steadily decreasing to a plateau at epoch 60. Throughout training, with the exception of a few spikes in the beginning of the training, the feed forward neural network consistently outperforms the mixture density network. If <10% Squared Absolute Error is taken the determinant for whether or not a prediction is deemed "acceptable" (i.e. close enough to the real prediction), it is observed that the feed forward neural network has a ~70% accuracy rate and that the mixture density network as a ~65% accuracy rate.

The results are rather surprising, as it was expected that the mixture density network would outperform the feed forward neural network by a considerable margin. The justification for this belief was that mixture density networks are specifically designed to learn dependent probability distributions, whereas feed forward networks hold no such specialization. While the data may point to the feed forward neural network as the model architecture for DeepWIFF, it may be less desirable simply because its results are less interpretable. The mixture density network has the added advantage of easily being interpreted as a mixture of gaussian distributions being used to approximate an arbitrary probability distribution. It also should be noted that an accuracy of 70% would not be considered acceptable enough to properly replace SP-WIFF in climate models. An accuracy of >95% is necessary for this, which neither model came close to achieving. That being said, both neural networks are very successful in terms of training time, taking less than a minute to run data that would normally take 9+ hours on SP-WIFF.

Challenges

Because this project is inherently exploratory in nature, the biggest obstacle for us was that we had no point of reference to make sure our model was maximizing its effectiveness. As such, it was quite difficult to settle on a model of choice that could effectively address the problem at hand. After testing a few networks and playing around with a few potential ideas (partially connected networks, seq2seq models, etc.), we found that the mixture density network

would be most suitable for our problem at hand. Many other network we tested models simply failed to train properly or converge at a realistic speed

Within our mixture density model, our biggest challenge was figuring out how to compare the continuous output of our network (a Gaussian mixture) to a discrete probability distribution, which was the form our testing/training labels used. Searching online and reading articles provided little help, as the articles describing mixture density networks always deemed the continuous outputs of the MDNs as acceptable. The other problem we encountered with this was making a differentiable "discretize" function to backpropagate on, as methods such as randomly sampling from a continuous distribution don't necessarily allow differentiation/backpropagation. We finally overcame this issue by creating a custom discretize function based on calculating differences in values from each Gaussian mixture's cumulative distribution function.

The other major point of difficulty in our models was hyperparameter tuning. We spent a large portion of our time tweaking the hyperparameters of our models to minimize MSE throughout training. Within our mixture density network, we often had to deal with numerical overflow errors when calculating gradients. To sidestep this issue, we changed learning rates, imposed regularity conditions, and started working with float64 rather than float32 values. While overcoming these challenges was difficult, we learned a great deal about neural networks and how to properly work with the nuances of different neural network architectures.

Reflection

Overall, we feel that our project was relatively successful. While we were disappointed to see that the control feed-forward neural network outperformed our mixture density network slightly, it provided important insight on what networks might be good for learning the data that SP-WIFF outputs. We were also disappointed to see that the models were not completely accurate, only correctly predicting outputs with a ~70% success rate. This would not be enough to hold up in the real world. We feel that while we may not have reached our stretch goals, we did realize our target goal to analyze the effectiveness of at least 2 different networks learning the SP-WIFF model.

Surprisingly, our data was contradictory to our original assumptions. Given that mixture density networks are specifically designed with the purpose of learning probability distributions, we expected the feed forward network to underperform in comparison to our mixture density network. This held true even after tuning our MDN's hyperparameters. We were also quite surprised to see that mean square error was the most effective loss function to train our model against. Although we performed relatively few runs using Kullback-Leibler divergence for training loss, our initial tests strongly suggest that it would be a weaker loss function, despite being a more technically backed measure of distribution similarity.

Reflecting on our overall work process, we realized that our approach changed greatly over time. Our initial strategy was to try complex architectures that failed, rather than sticking to the basics and building up. As we progressed through the project, we learned that tuning network

hyperparameters could provide far more improvement than adding newer or fancy layers. Given more time, we would approach our network designs more systematically, and we would explore other network architectures with known success. We would also train our MDN with far more mixture components (>25 rather than just 12), as we suspect this might also significantly improve the accuracy of our MDN.

It seems that the biggest takeaway from our paper is that "simpler is better". We found that the simpler models and simpler loss functions were much more effective than complex models. We suspect that with the addition of complexity to a model, unless utilized properly, the complexity acts as a buffer which prevents a model from achieving maximal success. Since we would like to continue working on this project, we will take this approach in the future and hopefully create a robust deep learning model which accurately predicts the data given by SP-WIFF. This work on our DeepWIFF model will hopefully be one of the first in a long line of neural networks used to learn models in climate science.

Ethical Considerations

While the DeepWIFF model experienced relative success in emulating the outputs of the SP-WIFF model, these results should not be taken at face value. As is the case with many other deep learning models, DeepWIFF must contend with the Black Box problem, a set of problems in which deep learning models lack interpretability (Rudin, 2019). On the broadest level, the network architectures of models like Mixture Density Networks are understood; one is simply approximating an arbitrary probability distribution using a mixture of Gaussians distributions. However, the calculations within the layers, especially in the feed-forward layers, offer little insight into why DeepWIFF outputs the distributions it does. Without further calculations to support using DeepWIFF in place of SP-WIFF, blindly using the output of DeepWIFF could potentially be dangerous, especially in a climate model where incorrect calculations could lead to incorrect conclusions about environmental phenomena.

Furthermore, DeepWIFF is finely tuned to specific climate parameters of SP-WIFF at a specific moment in time. As the climate changes and the parameters used in SP-WIFF change, the DeepWIFF network would need to be retrained, otherwise outputs of the DeepWIFF model could be outdated and possibly incorrect. As the process of constantly retraining the network might be costly, one should consider the environmental and temporal ramifications of using the DeepWIFF network.

References

Bishop, C. (1994). Mixture density networks. Technical Report. Aston University, Birmingham. (Unpublished). http://publications.aston.ac.uk/id/eprint/373/

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1, 206–215. https://doi.org/10.1038/s42256-019-0048-x

Horvat, C., & Tziperman, E. (2015). A prognostic model of the sea-ice floe size and thickness distribution. The Cryosphere, 9(6), 2119–2134. https://tc.copernicus.org/articles/9/2119/2015/