SASNets: Classifying Small Angle Neutron Scattering Data Using Convolutional Neural Networks

Christopher Wang, Advisors: William Ratcliff & Paul Kienzle
NIST Centre For Neutron Research

Goal: Streamline the process of fitting small angle neutron scattering data to theory models.
Motivation: Fitting data can be difficult if the model is not known, especially for non-experts. This delays the process of analysis and increases labour.
Results: Implemented a convolutional neural network capable of 54.9% accuracy on a set of 71 models. Network was also able to distinguish between classes, such as cylinder and sphere.

Introduction

Small Angle Neutron Scattering(SANS) is a common technique used in the analysis of diverse materials such as crystals to proteins. SANS is used to study the internal structure of materials because of the neutron's unique scattering characteristics. Neutrons are scattered off of the nucleus, producing different intensity levels depending on the shape of the object.

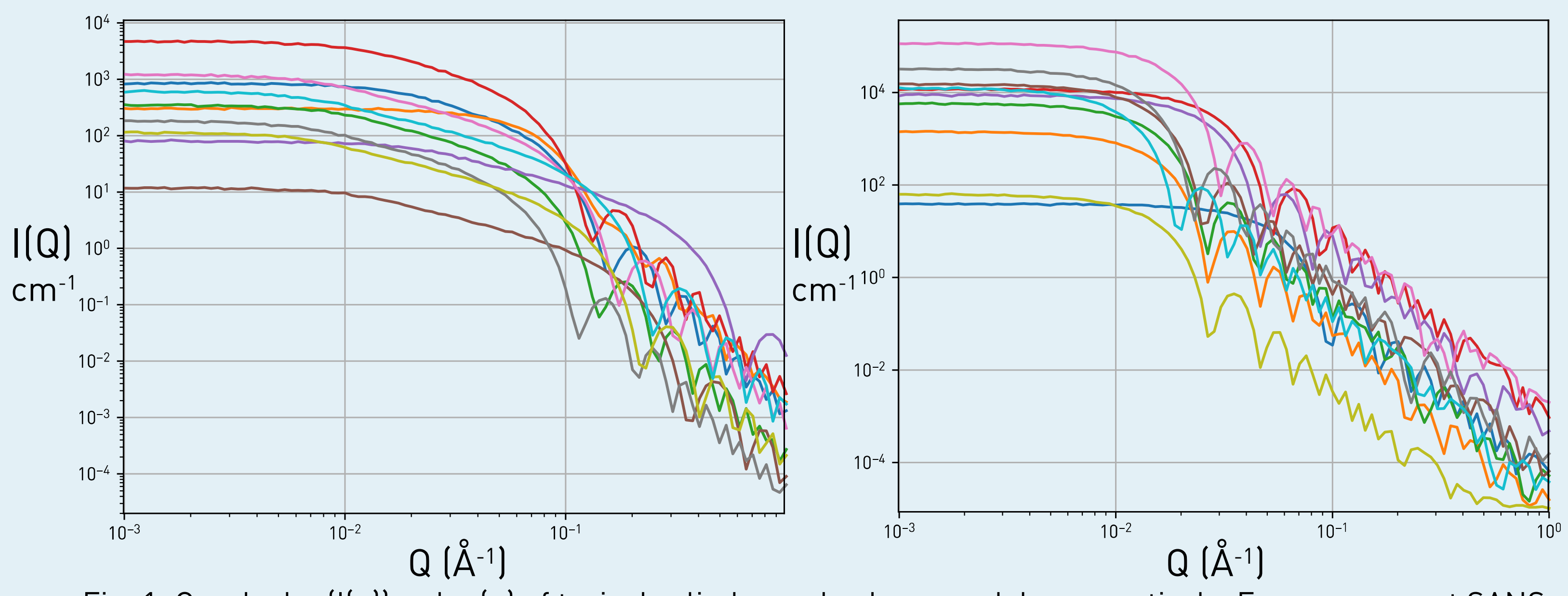
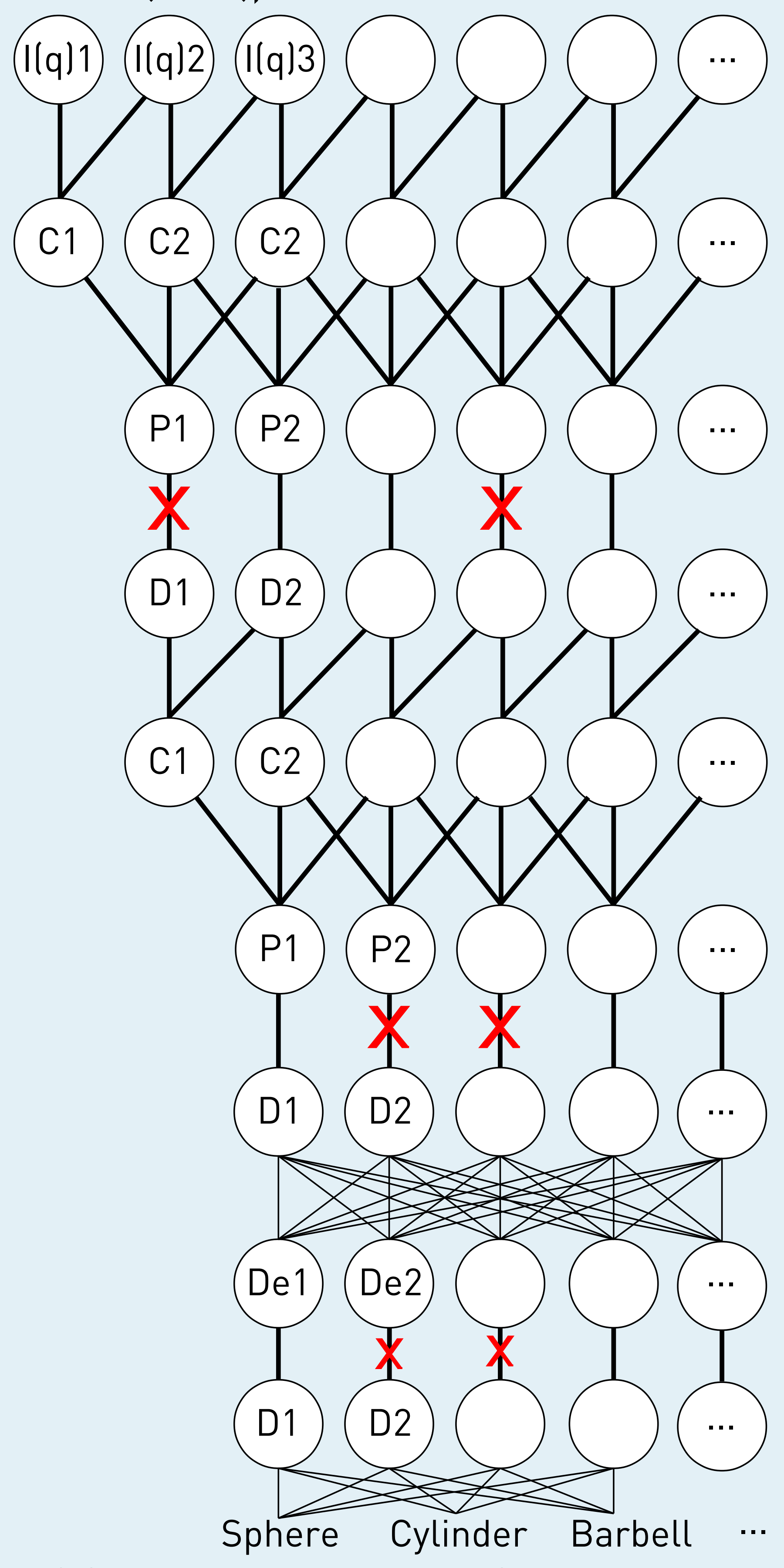


Fig. 1. Graphs log(I(q)) vs log(q) of typical cylinder and sphere models respectively. For a nonexpert SANS user, these can be difficult to distinguish.

Artificial neural networks are computer systems designed based on the human brain. They are composed of a network of nodes (axons) connected by edges (synapses). A neuron takes its inputs, multiplies them by a weight matrix, and adds a bias matrix. If the output quantity exceeds a threshold, the neuron outputs a value. Different neuron types, modelled after various biological functions, can be selected. In this research, we use a convolutional neural network(CNN), which are based on the animal visual cortex.



Input layer of 100 I(q) points.

Convolutional layers with filters of size 2.

Pooling layer that outputs the max of its three inputs.

Dropout temporarily removes a node with probability p.

Second convolutional layer with filter size 2.

Pooling layer with size 3 pools.

Dropout with probability p.

Densely connected layer where each neuron has an input form all preceding neurons.

Dropout of probability p.

Densely connected output layer.

Fig. 2. Simplified representation of our CNN structure.

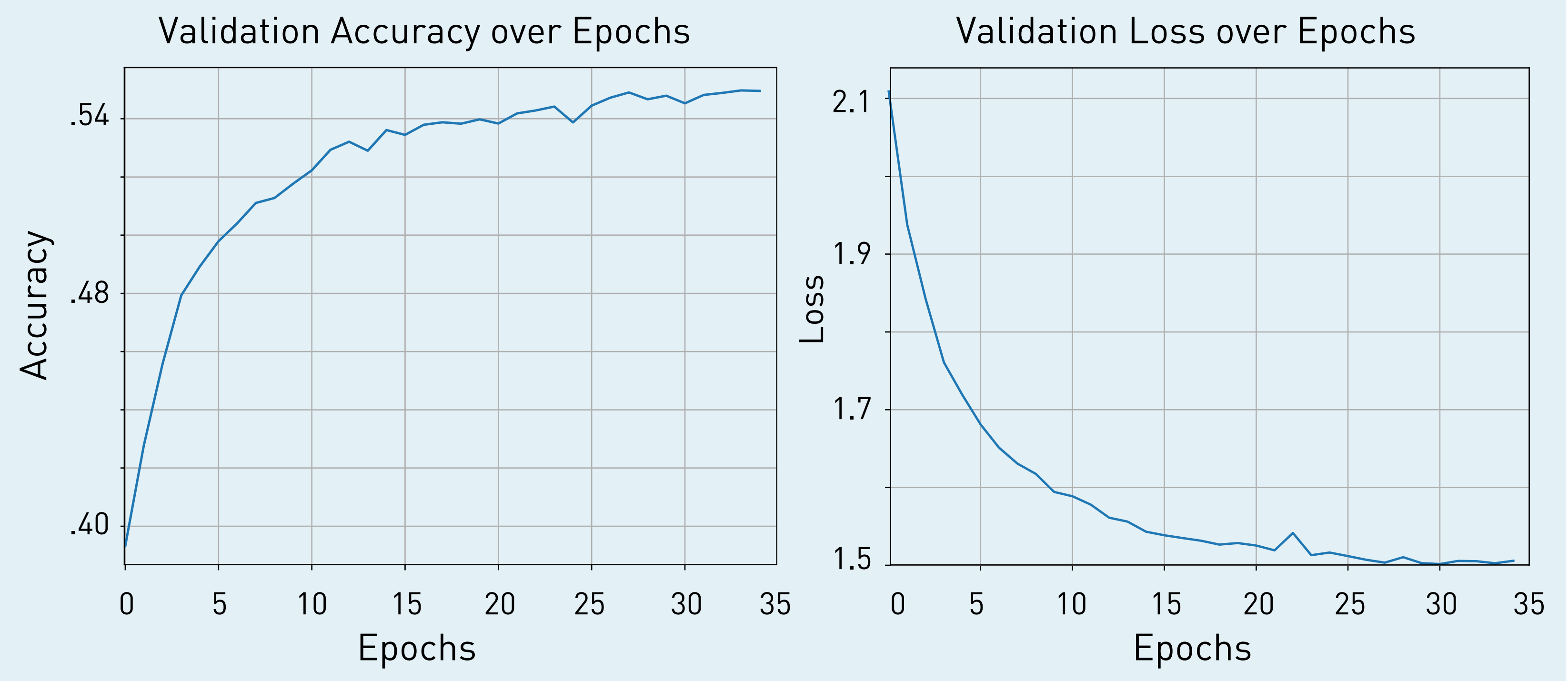
N.B. For a more complete mathematical introduction to SANS, see [2]. For more about softmax, see [5].

Conclusion

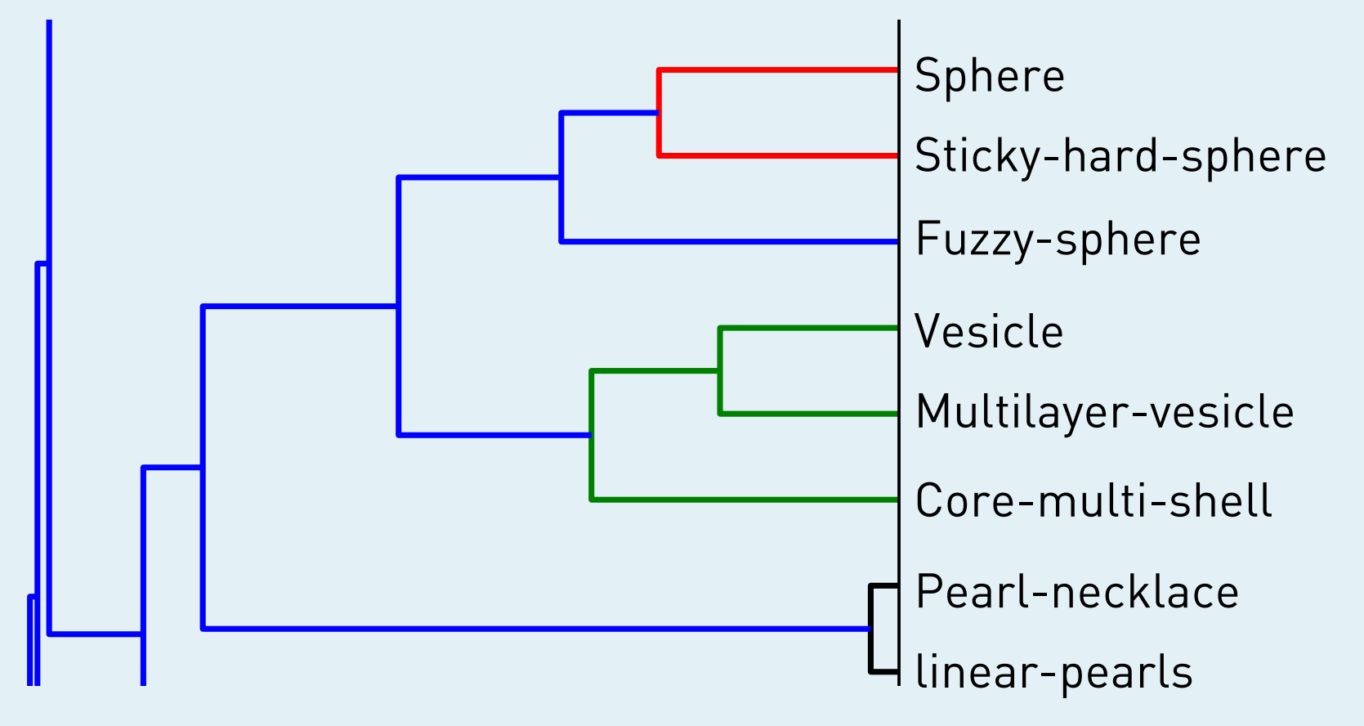
We demonstrate an implementation of a convolutional neural network which classifies SANS data into a set of 71 models with 54.9% accuracy. We also find that the network can group related models together based solely on the input. Development on the project continues; currently we are working on generating more realistic data and refining the model to be more robust to incomplete or noisy data. In addition, we are working on integrating model prediction with SASView, which will automatically fit to the theory model.

Results

The model was run for 34 epochs, where each epoch trained the model repeatedly on random samples of training data of size five until all training data had been used. At the end of each epoch, the model was evaluated with the validation data. The model achieved 54.9% accuracy at the end of the training run with a loss of 1.505 on the evaluation data. Training took a total of 2 hours and 36 minutes.



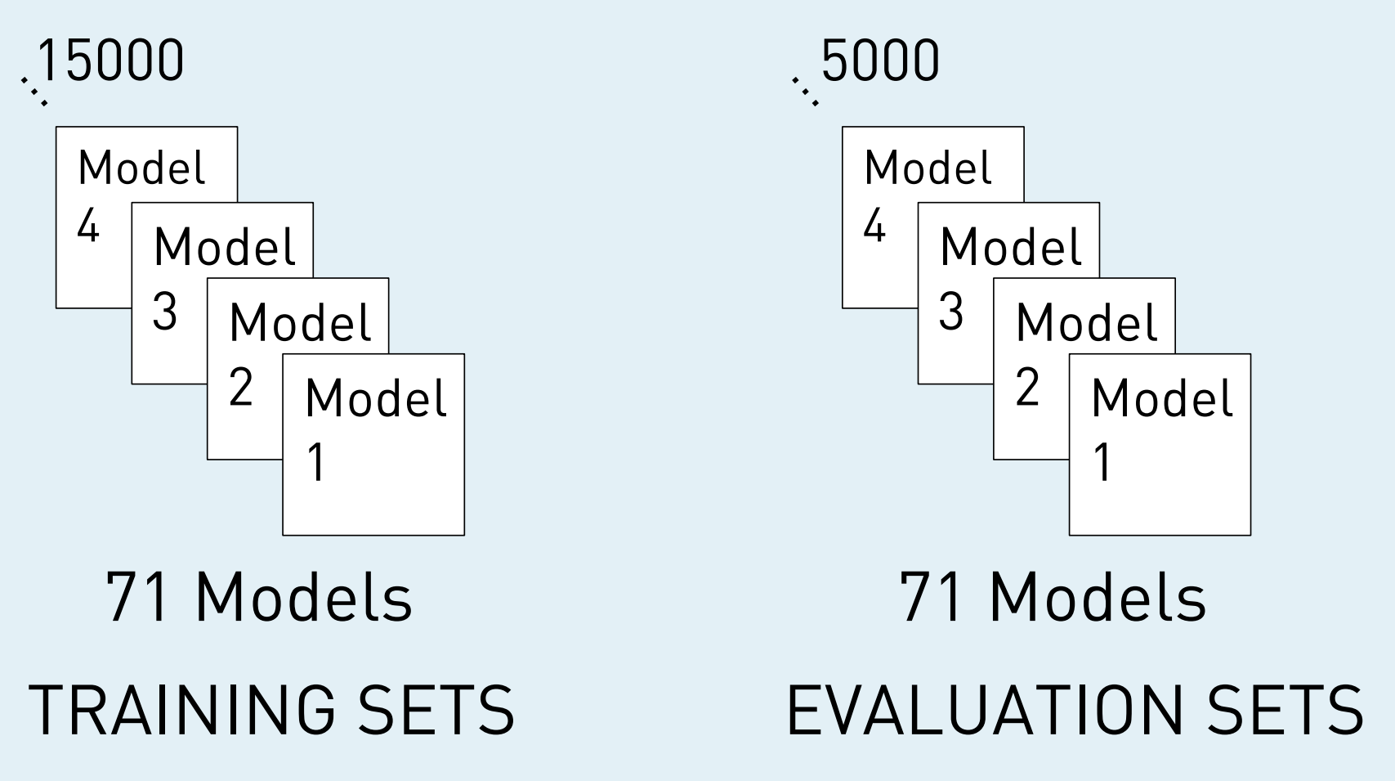
As there were 71 distinct models, an accuracy of 54.9% represents a 40× increase over random guessing. This demonstrates that a CNN has the potential to learn the correct model classifications very well.



In addition, the network tended to misclassify models within related groups. For example, a sphere would sometimes be misclassified as a sticky-hard-sphere, but never as a lamellar.

Fig. 4. An excerpt from a dendrogram, which clusters related models together based on how often they were confused with each other. Here, the grouping of spheres is shown. A full dendrogram in SVG form can be found on the website.

Experimental Setup



- Each model has 100 I(q) points
- Train for 34 epochs
- Adam[3] and softmax
- Separate evaluation set provides independent statistics
- Constructed using Keras, Tensorflow and Python.

The model is trained using I(q) as the predictor and the model name as the expected output. The system specs were: i7-7700, Nvidia Founders Edition GTX 1080 Ti, 512GB M.2 SSD, and 16GB DDR4 2400MHZ RAM.

N.B. More information about Tensorflow can be found at [1].

References

[1] Google. Tensorflow Documentation. (2017).
[2] Jackson, A.J. (2008). *Introduction to Small-Angle Neutron Scattering and Neutron Reflectometry.*
[3] Kingma, D.P., Ba, J. (2014). Adam: A method for stochastic optimisation. arXiv:1412.6980
[4] LeCun, Y., Bottou, L., Bengio Y., Haffner P. (1998). Gradient-based learning applied to document recognition. *In proceedings of the IEEE*, 86(11):2278-2324, November.
[5] Menard, S. (2002). *Applied logistic regression analysis* (Vol. 106). Sage.

Acknowledgements

This research was supported by funding and material support from the NIST Centre for Neutron Research and its Centre for High Resolution Neutron Scattering and by code developed by the SANS subproject of NSF-funded DANSE DMR-0520547. Finally, we thank the many colleagues who have provided advice, data, and material, without whom this project would not have been possible.

Further Info

For more information about this project, scan a QR code from the right.

Website

Github

Contact

Chris Wang: wangch00@icloud.com, Paul Kienzle: paul.kienzle@nist.gov, William Ratcliff: william.ratcliff@nist.gov