Deep Learning and Physics - Project Report
SANS: Using CNN on intensity arrays to predict the model type

Hassan Habib, Cagil Benibol

20.4.25

---

1 Introduction

This project attempts to predict the type and name of nuclear model based on the application of deep learning methods on 2D intensity arrays. These intensity arrays were obtained by measurements in virtual experiments (VE) simulated by Monte Carlo neutron tracing, performed by the Global Neutron Scientists (GNeuS) project. VEs were performed under the conditions observed at the FRM-II reactor in Garching, Germany. The corresponding dataset contains 46 form factor models that will be predicted using a three different DL vision models on the array.

2 Methods

We used python library PyTorch to apply all three models for the purposes mentioned above.

2.1 Data Processing

In the dataset, each image (a 2D intensity array) is a floating point value pixel of size 256 x 144. The total database consists of 259,328 PSD images. The training dataset consists of 181,531 PSD images (80%); the testing dataset consists of 51,866 PSD images (20%); and the validation dataset consists of 25,934 PSD images (10%). The data set obtained was originally in the format `.h5`. To utilize the torch library, we first transform it into the `torch.Tensor` format.

Regarding the models metadata, they were obtained in a separate dataset in the `.pkl` file type. This dataset included information such as instrument parameters, model names, and other metadata for the parameters. The data were then parsed and transformed into a relevant datatype within the code. Afterwards, the images and label were loaded as a `(torch) Dataloader` for further modeling.

2.2 Models

2.2.1 A simple CNN

As mentioned above, we used PyTorch to implement our DL model. The implemented neural network is a convolutional neural network (CNN) designed for a classification task with 46 output classes.
Architecture: The network starts with three convolutional layers, each followed by batch normalization and a ReLU activation function to introduce non-linearity. The first convolutional layer applies 64 filters with a 3×3 kernel, the second applies 128 filters, and the third applies 256 filters, all using a stride of 1 and padding of 1. Each convolutional layer is also followed by a max-pooling operation with a 2×2 kernel, which reduces spatial dimensions and prevents overfitting.
After the convolutional feature extraction, the output is flattened and passed through three fully connected layers. The first fully connected layer has 512 neurons, the second has 128 neurons, and the final output layer has 46 neurons, corresponding to the number of classes. ReLU activation is applied between the fully connected layers, and dropout with a probability of 0.5 is used to reduce overfitting.
Optimizer and Loss function: The model is trained using the cross-entropy loss function, which is suitable for multi-class classification. The optimizer used is Adam with a learning rate of 0.01 and momentum of 0.8 to improve convergence during training. This CNN model effectively extracts hierarchical features from input images and classifies them into one of 46 categories.

2.2.2 VGG-19

The second model we built is the classic VGG-19 model, which won the ILSVRC in 2014.
Architecture: We implemented model a modified VGG-19 architecture designed for grayscale image classification. It consists of five convolutional blocks, each followed by max-pooling, progressively increasing the number of filters (64 to 4096). The feature extractor is followed by three fully connected layers with ReLU activation and dropout for regularization. The output layer has 46 units for multi-class classification.
Optimizer and Loss function: The model uses Cross-Entropy Loss and the Adam optimizer with a learning rate of 0.001. We trained on a GPU-enabled device for efficient computation.

2.2.3 ResNet-34

Next up, we used a pre-built model from the PyTorch library to compare against the rest. However, we did not use their pre-trained weights.
Architecture: This model is based on a pre-trained ResNet-34 architecture, modified for grayscale image classification. To adapt the network for single-channel input, the original first convolutional layer (designed for 3-channel RGB images) is replaced with a custom 2D Convolution layer for a single channel. We made sure that the architecture retains ResNet-34's core structure of 34 layers with residual connections and batch normalization, which stabilize training and mitigate vanishing gradients. The final fully connected layer is

replaced with a linear classifier to produce outputs for the 46-class classification task.
Optimizer and loss function: Our custom model is optimized using the Adam optimizer with a learning rate of 0.001 and trained with Cross-Entropy Loss.

3 Results

The following are our results for the three models that we implemented. This includes one-time run on the test dataset to check out the generalization abilities of each model.

3.1 Simple CNN

After running a simple-CNN model with 25 epochs and bin size of 32, we have the following result:
Final loss: 2.3556
Training accuracy: 18.52 %

3.2 VGG-19

After running a VGG-19 model with 25 epochs and bin size of 16, we have the following result:
Final loss: 1.8388
Training accuracy: 38.12 %

3.3 ResNet-34

After running ResNet-34 model with 25 epochs and bin size of 32, we have the following result:
Final loss: 1.1888
Training accuracy: 55.56 %

4 Discussion

In general, we tried to obtain better results. We have tried three different models of increasing depth to improve the accuracy. The best results were obtained by the ResNet-34 model, the biggest model of them all (as expected). It can be concluded here that, until a certain depth, the models are bound to improve more as their size is increased. Therefore, we expect, say ResNet-50, to perform even better than models implemented by us. All in all, our approach was to try out different models and model components (like optimizers, activation functions etc.), starting from a simple one like an unmodified CNN to more complex models like ResNet-34.