Motivation
The training process of Convolutional Neural Networks (CNN) can be lengthy. Being able to predict the performance of a certain set of hyperparameters on a specific CNN architecture can save both time and computation cost.

This project focuses on predicting the final accuracy of a CNN using only the validation accuracy of the initial 100 epochs. With our best model, we are able to achieve an average error of 0.38%.

Data
We collected validation accuracy curves from LeNet and Vgg-19 architecture with 11 different sets of hyperparameters and activation layers.

The data are normalized under two schemes:

\[
x' = \frac{x - \text{mean}(x)}{\text{max} x - \text{min} x}
\]

The dataset consists of 34 curves and is split into 22/6/6 (train/val/test).

Models
(K-)Nearest Neighbor:
- New input resembles known shapes in the training set.
- Loss for curve fitting: \[ L = \sum (y - \hat{y})^2 \]

Clustering:
- Time-series probabilistic distribution assumption.
- Loss: MSE = \[ \frac{1}{T} \sum (y - \hat{y})^2 \]

Markov Chain Monte Carlo:
- Time-series probabilistic distribution and first order Markov assumption \[ y_{epoch} \sim P_{epoch}(y_{epoch} | y_{epoch-1}) \].
- Gaussian prior distribution and Gaussian white noises.
- Metropolis-Hasting sampling.

Sliding Window Prediction:
- Capture the pattern in the change rate of the curve.
- Sliding window approximation of the curve.
- Three-layer neural network for capturing pattern with early stopping and dropout.

Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Curve Fitting</td>
<td>0.1761%</td>
<td>0.7523%</td>
<td>0.6331%</td>
<td>0.5264%</td>
</tr>
<tr>
<td>(K-)Nearest Neighbor</td>
<td>0.2250%</td>
<td>0.6753%</td>
<td>0.4276%</td>
<td>0.3824%</td>
</tr>
<tr>
<td>Clustering-6</td>
<td>0.1176%</td>
<td>1.8678%</td>
<td>0.8139%</td>
<td>0.7335%</td>
</tr>
<tr>
<td>Markov Chain Monte Carlo</td>
<td>0.1466%</td>
<td>0.7043%</td>
<td>0.3853%</td>
<td>0.3825%</td>
</tr>
<tr>
<td>Sliding Window Prediction</td>
<td>0.3218%</td>
<td>0.8258%</td>
<td>0.5953%</td>
<td>0.5967%</td>
</tr>
</tbody>
</table>

Models
Normalization:
- Helps generalize the curve shape.
- Both schemes achieve approximately the same level of performance.

Model Performance:
- Direct curve fitting can often fail to predict the decay rate of the curve in later epochs using the prior epochs.
- MCMC achieves the best performance as it takes time-series analysis into account, but can be unstable due to sampling.
- Nearest Neighbor achieves good performance by using generative methods to account for uncertainty. It takes significantly less computation than MCMC (2 secs vs. 45 mins).
- Sliding window prediction does not capture the increase rate as expected due to the noisy nature of the data.

Future Work
- Generate data using more advanced architectures (ResNet/MobileNetV2).
- Use 1-D convolution / averaging to capture the trend of the curve.

References