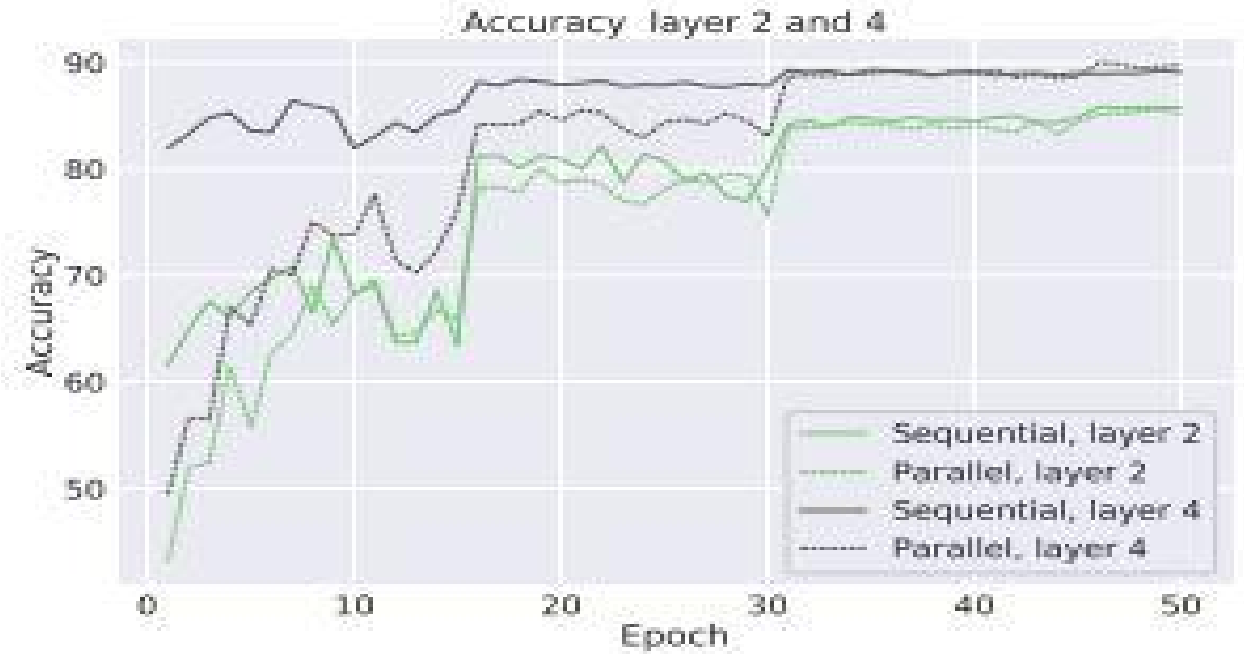
The methodology that is implemented here is a decoupled greedy algorithm of Convolutional Neural Network with the MNIST dataset. The system takes an image as an input for the training set, the model trains itself and then predicts the output for the testing data. The aim of this is to handle the backward locking, update locking and forward updating problems of the CNN. There have been previous attempts to solve the backward locking problems but the update locking as well as the forward locking problems were not addressed before. The sequential nature of the gradient processing introduces some of these problems. Optimized procedure for Decoupling greedy algorithm is performed to achieve update unlocking in a more efficient way than the previously proposed methods. The same is then extended to asynchronous environments with replay buffers to achieve forward unlocking with minimized errors.

The reason for using the MNIST(Modified National Institute of Standards and technology) dataset was it's a huge dataset consisting of handwritten digits. It is commonly used for image processing systems. Largely used for machine learning in regards to testing and training. It contains about 60,000 training images and 10,000 testing images. During training of the model, a sample of data in the form of "minibatches" is passed. The data is then reshuffled at every epoch so that it does not cause overfitting and then use feature of multiprocessing from Python to speed up the process of data retrieval.
DataLoader is used for abstraction of complexity in the form of an API. In order to preserve the size of the image I try to use zero padding.

Algorithm:

Input: Stream $\mathcal{S} \triangleq \{(x_0^t, y^t)\}_{t \leq T}$; Distribution of the delay $p = \{p(j)\}_j$; Buffer size M.
Initialize: Buffers $\{B_j\}_j$; params $\{\theta_j, \gamma_j\}_j$.
while training do
  Sample j in $\{1, ..., J\}$ following p.
  if $j = 1$ **then**
    $(x_0, y) \leftarrow \mathcal{S}$
  else
    $(x_{j-1}, y) \leftarrow B_{j-1}$.
  end
  $x_j \leftarrow f_{\theta_{j-1}}(x_{j-1})$.
  Compute $\nabla_{(\gamma_j, \theta_j)} \hat{\mathcal{L}}(y, x_j; \gamma_j, \theta_j)$.
  $(\theta_j, \gamma_j) \leftarrow$ Update parameters (θ_j, γ_j).
  if $j < J$ **then** $B_j \leftarrow (x^j, y)$.
end

To get rid of the update locking problem and also allow parallel processing, we use relay buffers. This allows the layer to use the previously sampled data and not wait for the layer to completely finish the processing before it can update.



References:

- https://towardsdatascience.com/deep-q-network-dqn-ii-b6bf911b6b2c
- Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. International Conference of Machine Learning, 2017.
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In Advances in neural information processing systems, pp. 153–160, 2007.
- Eugene Belilovsky, Micheal Eickenberg, Edouard Oyallon. Decoupled Greedy Learning of CNNs. International conference on Machine learning 2020.
- https://medium.com/@nutanbhogendrasharma/pytorch-convolutional-neural-network-with-mnist-dataset-4e8a4265e118