BHCal Energy Calibration PyTorch Proposed Plan

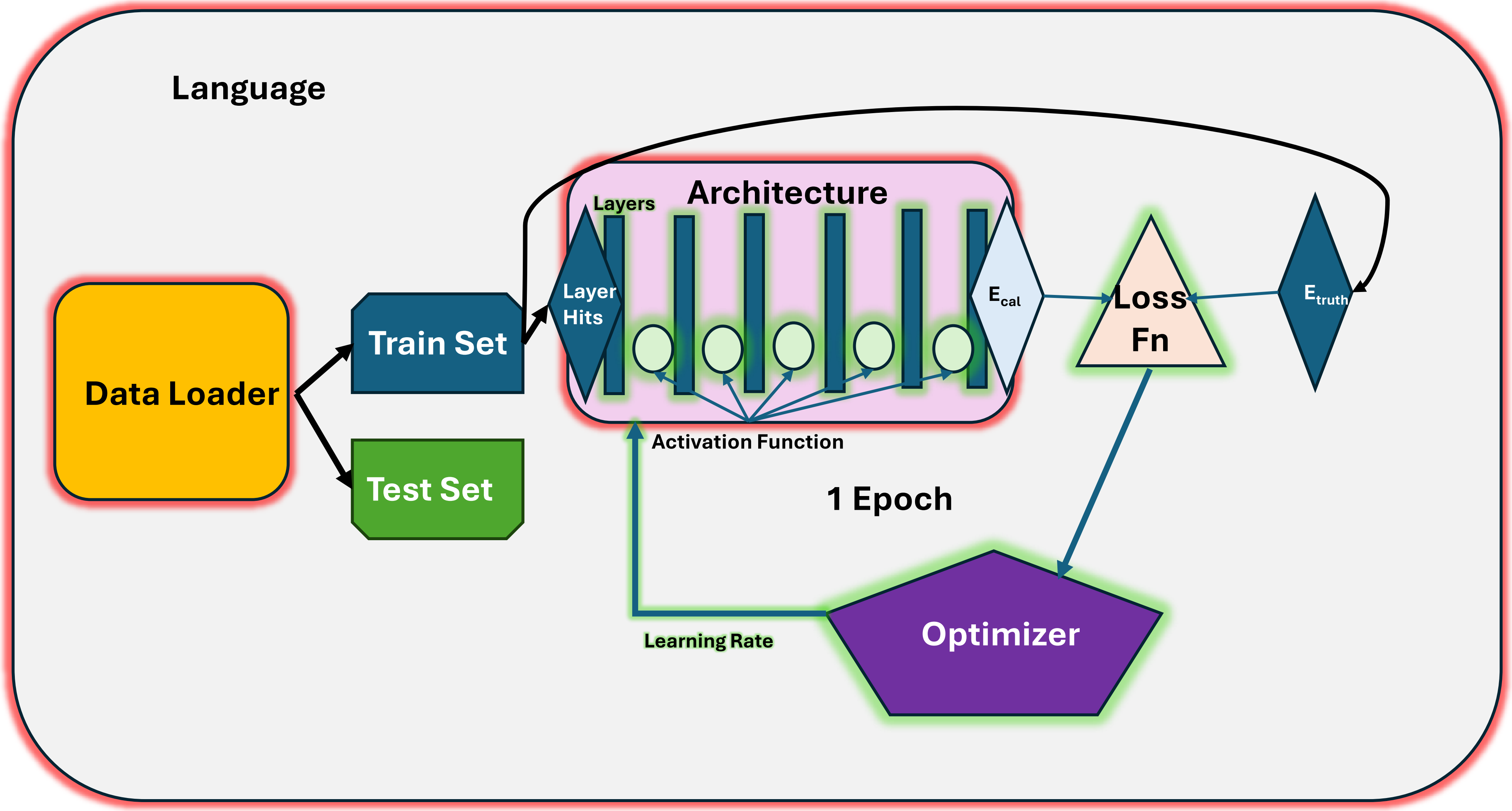
David Ruth

Framework

- Several possible frameworks (PyTorch, Keras, Tensorflow)
- PyTorch is ‘implicitly supported’ by the EIC software package

- Benefits:
  - Better opportunities to optimize and use computational power than TMVA
  - Ability to schedule learning rate decay
  - Ability to automatically tune hyperparameters
  - Use of dropout
  - More flexible

PyTorch



Red Glow – Larger Procedure Decision

Green Glow - Hyperparameter

Language Options

- Python ← **My Suggestion**
  - Language the architecture was designed for
  - Very straightforward, could have a basic working model within a few weeks

- C++
  - Easier to interface with ROOT
  - Can connect to existing simulation codes more easily

Data Loader Options

- Generate a Text File for Training, Etc.
  - Slightly easier to do
  - Less efficient for large trainsets
  - One row for each event, one column for each layer and one for E_{truth}

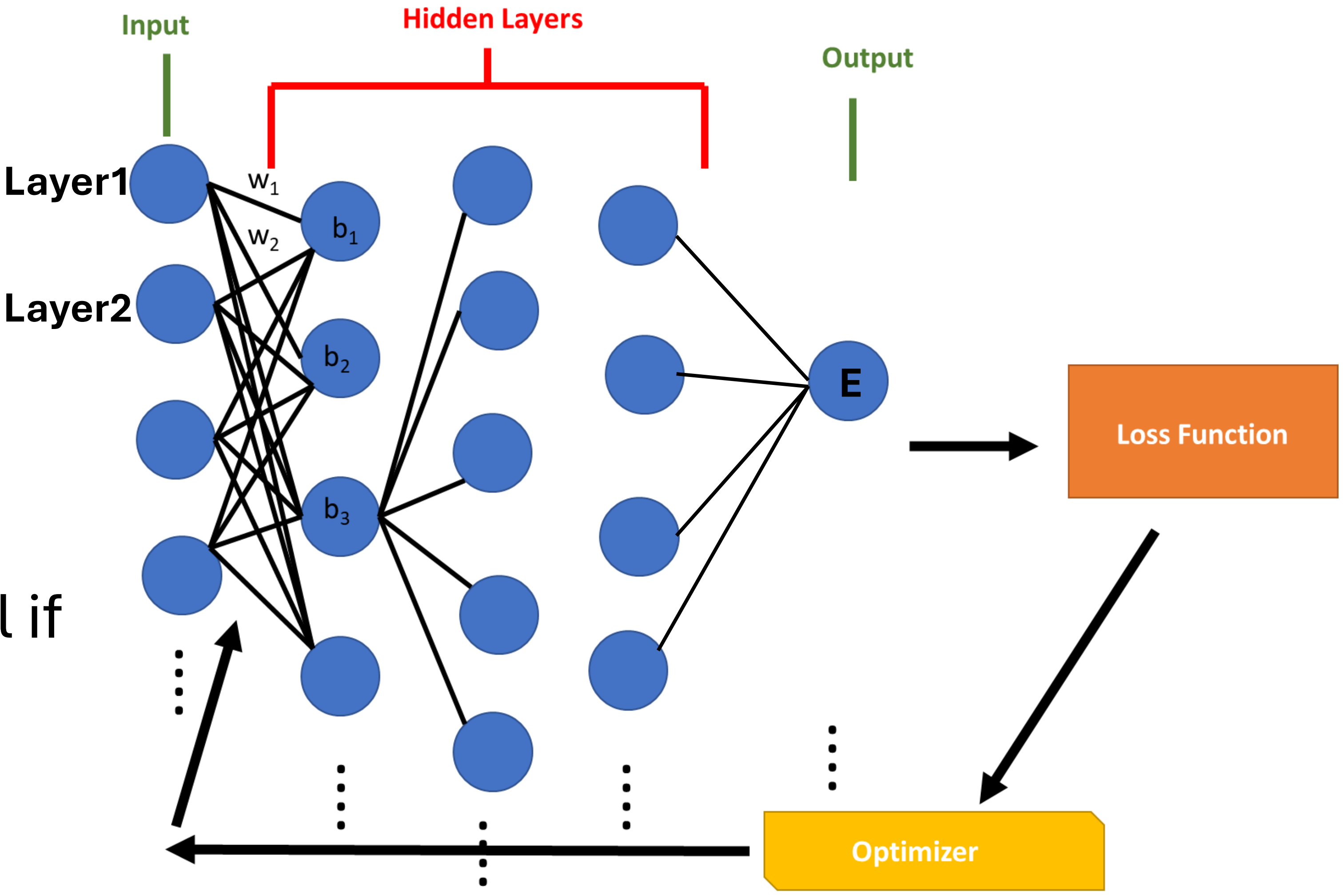
- Use PyRoot (or native C++ support) to Convert Root TTrees to PyTorch Tensors
  - Significantly more time consuming to figure out
  - More efficient for large trainsets
  - Easier to retrain model in future

Architecture Options

$$E = C(w_1L_1 + w_2L_2 \ldots + b_1) + \ldots$$

Multilayer Perceptron

- Simplest possible neural network
- Fully connected
- I have the most experience with it
- Less computationally efficient, but practical if you have computing power to spare



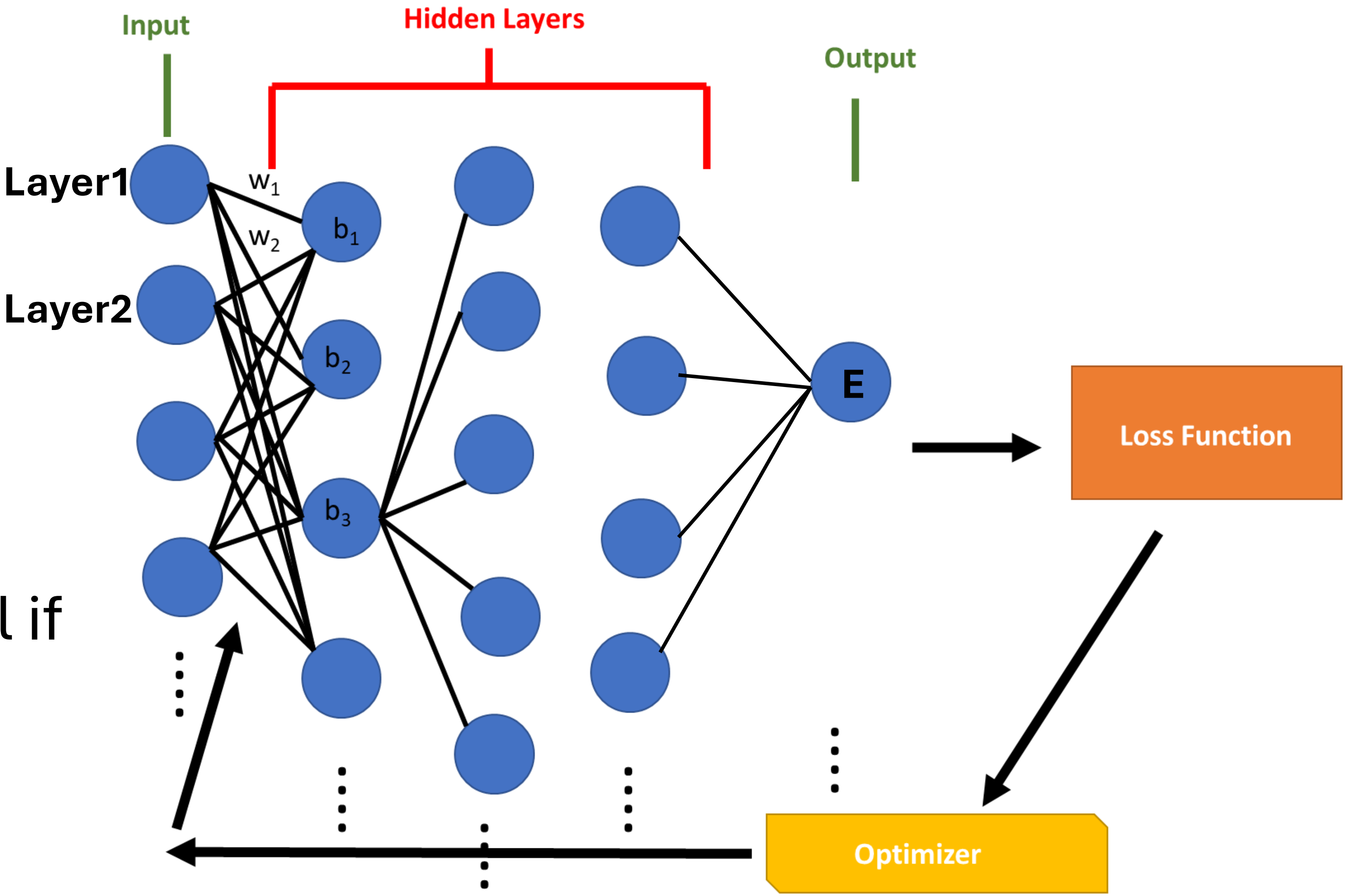
Architecture Options

$E = C(w_1L_1 + w_2L_2 \ldots + b_1) + \ldots$

My Suggestion

Multilayer Perceptron

- Simplest possible neural network
- Fully connected
- I have the most experience with it
- Less computationally efficient, but practical if you have computing power to spare



Architecture Options

- Convolutional Neural Network or Graph Neural Network
  - Both good for “images” or trainsets with a complex relationship between input nodes
  - In my opinion, not suited for this relatively simple problem
- K-Nearest Neighbor
  - Classifier Network that Groups Like Elements
  - **Only viable** for heavily discretized energies
- Linear Discriminant Analysis
  - Also a classifier, but very straightforward
  - Should be up and running quickly but would also break down for a continuous energy spectrum
  - “Best” performance of the TMVA networks

Proposed Timeline / Division

- Python
- Text File Data Loader (to start)
- Multilayer Perceptron
- **David:** Design neural network and establish basic functionality (3-4 weeks)
- **Derek:** Supervise integration with existing simulation codes
- **Olaiya (+Nathaly? +Anyone else interested?):** Hyperparameter optimization (4-8 weeks)
  - **N_{layers}**
  - **$N_{neurons}$ per layer**
  - **Learning Rate**
  - **N_{events} for training**
  - **Learning Decay**
  - **Loss Function**
  - **Optimizer**
  - **Activation Function**
  - **Dropout**
  - **Epochs**
- Estimate around total ~3 months to perfect the new neural network