**C100 Fault Classification Workflow**

*October 7, 2022*

**Step 1: Data Extraction**

In the data extraction process we collect 4 signals from each cavity [GMES, CRFP, DETA2, GASK]. We collect a total of 32 signals for each fault (4 signals/cavity × 8 cavities). We extract data for each run separately. We extract data and separate them based on the label file provided by a subject matter expert. The label file is in .txt format. We use rfwtools (<https://github.com/JeffersonLab/rfwtools>) to extract the data.

**Step 2: Data Preprocessing**

A signal from a single event contains 8,192 time samples and spans approximately 1.64 seconds. In our experiments we consider a window 7,680 time samples long ranging from 1533.4 ms before fault to 2.4 ms after fault (this corresponds to entries 13 to 7,693). After taking 7,680 time samples we resample to 4,096 time samples. We do standard normalization (z-score) of each signal.

**Step 3: Data Merge and Split**

For the classification models (cavity and fault), we use data starting from November 25, 2019 through February 8, 2022 to build the network. Data collected between May 1, 2022 and July 11, 2022 is used to fine-tune the network. We use data from July 12, 2022 to September 14, 2022 to evaluate the network (i.e., simulate performance as if it was deployed in the machine).

We perform two classifications:

1. Cavity Identification: we consider all the faulty data
2. Fault Classification: we exclude faults due to multi cavity turn-off

We split the dataset into training (60%), validation (20%), and testing (20%) to build the network. We perform a similar data split for layer freezing or transfer learning experiments.

**Step 4: Model Building**

To build the model, we train the network using training data, and validate the model using validation data, and test. The model is saved. Finally, the model is evaluated using a standalone dataset. Figure 1 shows the block diagram of the implemented model.

Diagram, schematic

Description automatically generated

FIGURE 1: Model architecture for classification (same architecture is used for cavity identification and fault classification).

**Step 5: Fine-Tuning**

If the model is trained using the dataset from one operational run and we want to implement the model for a different run, we need to fine-tune the network to improve performance. To fine-tune the network we need to load the trained model. We perform fine-tune using two different techniques:

1. Layer Freezing
2. Transfer Learning

|  |  |
| --- | --- |
| **Layer Freezing** | **Transfer Learning** |
| Train a network using 1st dataset | Train a network using 1st dataset |
| Freeze some layers of the trained network | Load the model of the trained network |
| Re-train the network using a part of the 2nd dataset | Re-train the network using a part of the 2nd dataset |
| Test using another part of the 2nd dataset | Test using another part of the 2nd dataset |

In layer freezing technique, we freeze all layers of the trained model except fully connected (FC) layer. We re-train the model using training dataset of the current run and validate the model using current run validation data. After finishing the training process, the model is saved and tested using test data.

In transfer learning techniques, we re-train the loaded saved model using the model using training dataset of the current run and validate the model using current run validation data. After finishing the training process, the model is saved and test using test data.

**Step 6: Test**

We load the latest model for deployment into the system. We select best performing model among normal model (no fine-tuning) and fine-tuned model (layer freeze or transfer learning). For the October 2022 deployment, the best performing cavity classification model used transfer learning. The best performing fault classification model did not utilize any fine-tuning.

**Step 7: Test Results Before Deployment**

***Cavity Identification***

**Calendar

Description automatically generated with medium confidence**

****

***Fault Classification***

Table

Description automatically generated with low confidence

