SCTE•ISBE CABLE-TEC EXPO® 2020 VIRTUAL EXPERIENCE » OCTOBER 12-15

2020 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS®

Convolutional Neural Networks for Proactive Network Management

Developing Machine Learning Models to Detect and Classify Impairments in D3.1 OFDM Channels

A Technical Paper prepared for SCTE•ISBE by

Jude Ferreira
Principal Data Scientist
Comcast
215.286.4070
jude_ferreira@cable.comcast.com

Maher Harb
Director, Data Science
Comcast
267.260.1846
maher_harb@comcast.com

Karthik Subramanya
Research Engineer
Comcast
267.260.2289
karthik_subramanya@comcast.com

Bryan Santangelo
Executive Director, Data Eng and Science
Comcast
918.640.8936
bryan_santangelo@cable.comcast.com

Dan Rice
Vice President, HFC Architecture
Comcast
720.512.3730
daniel_rice4@comcast.com

Table of Contents

List of Figures

List of Tables

1. Introduction

The signal quality on HFC networks can degrade over time, from an impairment perspective, if not proactively maintained. Comcast manages hundreds of thousands of miles of network throughout the world in which we experience a wide array of conditions that can degrade the performance of the network. From connections loosening and cracks forming, to lines getting cut, destructive energy and signal impediments are part of maintaining modern networks. Early detection, mitigation, and routing fix agents efficiently, to the right location, can not only improve customer experience but also enable operational efficiency.

Adaptive Profile Management Application (PMA) systems continue to be deployed to manage downstream and upstream network capacity and network stability. However, and perhaps ironically, PMA systems can mask the degradation of the network, as an inherent function of the optimization and mitigation process. It therefore has become increasingly important to develop systems that can support automated Proactive Network Maintenance (PNM) to reduce the impact of network impairments on customer experience and enable the highest possible capacity and performance.

In this regard, Comcast has invested heavily in data platforms and data science functions across organizations, to become more data driven and to incorporate Machine Learning (ML) approaches into the network. In this paper, we will describe the use of Convolutional Neural Networks (CNNs) to identify network impairments within DOCSIS 3.1 (D3.1) channels with a high degree of accuracy.

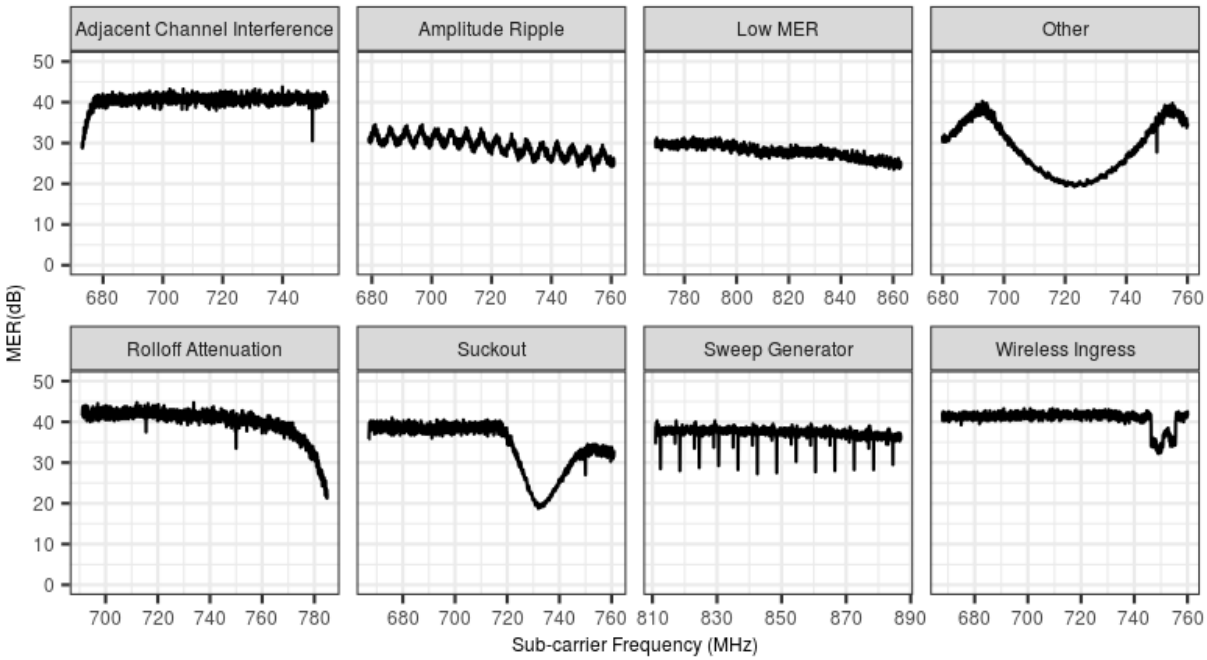

Figure 1. Different types of impairments seen in D3.1 OFDM Channels

It is beneficial to classify the various network impairments (shown in Figure 1) as they may warrant different responses from techs in the field to enable fastest possible Mean Time To Repair (MTTR). In

addition, clustering of these impairments across geographic locations and network topology may be exploited to identify the root cause impacting multiple customers that share common points in the network. Note that the latter requires a second layer model to be built on top of the classification model described in the paper.

The model we describe improves on the rule-based approaches currently being used to identify Mobile Wireless Ingress and Sweep Generator patterns. Notifications from the current rule-based model for detecting Mobile Wireless Ingress are sent across a notification bus to other Comcast OSS tools, to ensure that technicians are dispatched to the right hubs, network segments, and homes to remediate issues. The rule-based approach also provides a baseline for evaluation for the ML approaches. We also developed a real time version of the algorithm that techs can use to check that issues have been fixed after remediation. The same workflow described in this paper could also be used with alternate CNN-based models.

2. Data Collection: OFDM Receive Modulation Error Ratio (RxMER)

The underlying data for the model comes from a data collection platform, which acquires telemetry from cable modems and CMTSs for various performance-related measures, as well as other characteristics, such as make, model, hardware and software versions. The API platform runs both on pre-determined intervals as well as on-demand to collect data from the entire access network. Real time use cases are typically focused to specific groups of cable modems or interfaces with higher periodic rates. The collection platform also provides synchronous and asynchronous API requests so that response from cable modems and CMTSs can be returned back to the consumer, sent on a message bus for multiple consumers, and captured into our data lake.

Comcast supports millions of D3.1 devices. The data collection captures high resolution Receive Modulation Error Rate (RxMER) data from these devices at regular intervals. The methods described to identify various impairments for OFDM Channels in this paper are based on this high resolution RxMER measure per subcarrier (as shown in Figure 1). Modulation Error Ratio is an important measure for consideration, as it not only picks up on core signal-to-noise (SNR) characteristics but also all signal imperfections. Therefore, high resolution (high frequency and per-OFDM subcarrier) captures of RxMER is the primary measure and focus for characterization of impairments.

3. Labels and Supervised Learning

Methods for training models with well labeled data sets are well established and vastly varietal. While initial attempts focused on non-supervised machine learning methods, such as clustering, to separate out impairments into similar groups, the results were not promising. Thus, we recognized the need to label a set of training data to be applied to supervised machine learning methods. Labeling can be labor-intensive activity that, in this paradigm, would also rely on subject matter experts to examine and classify the impairments based on their expertise. In order to make the labeling process robust, we developed a Pattern Labeling user interface (UI) that makes RxMER samples available to labelers. To help capture impairments, and given that a majority of RxMER samples do not have impairments, the sampling strategy focused on capturing samples with high variance over OFDM subcarriers. When presented, the labelers can examine and submit their assessment for the impairments within a few mouse clicks.

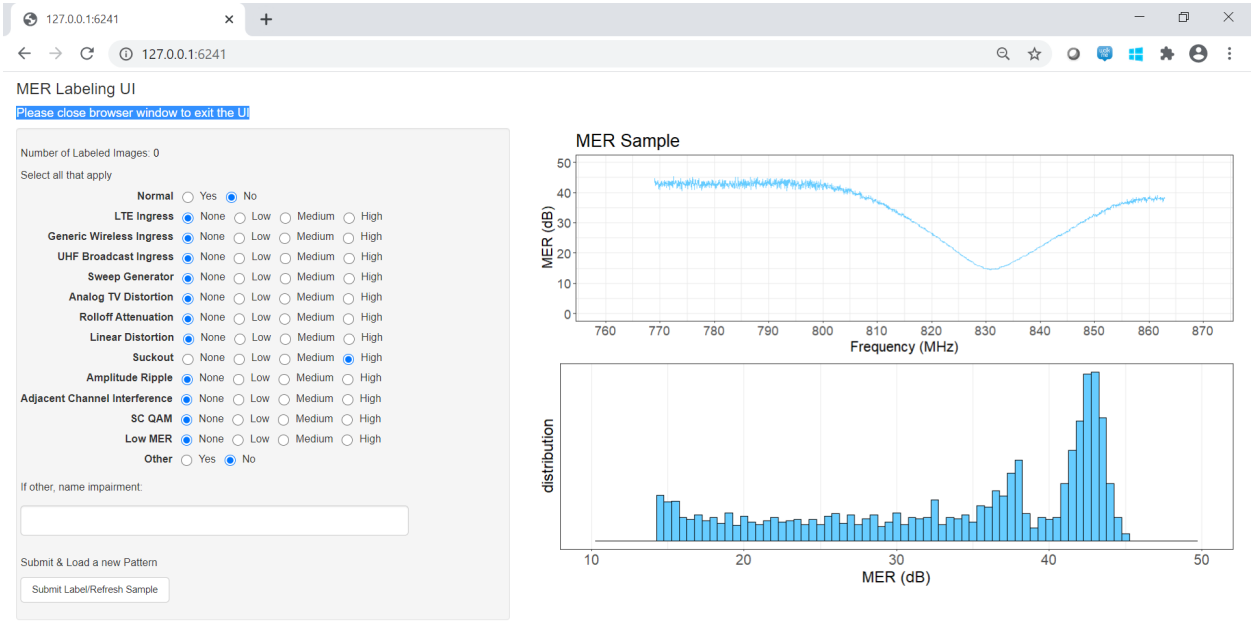

Figure 2. Screenshot of the RxMER Pattern Labeling UI

Labels are a critical component to building a good performing classification model. Given the significant value of enhancing capacity and improving customer experience by being able to identify and characterize specific impairments, we aimed for a crowdsourcing approach involving field technicians and other SMEs to generate labeled data using this UI. Once initial models are developed, we plan to pre-populate the Labeling UI with the impairments predicted by the different models to make the process more efficient.

D3.1 OFDM Channels are configured to extend from 24 to 192 MHz and often placed in the highest spectrum of the Hybrid Fiber Copper (HFC) network above the video and D3.0 channels. In many service groups, some portion of the OFDM channel is outside the HFC design, in what is often referred to as the "roll-off" spectrum. Given the size of the channels and where the OFDM channel may be located, it is possible that some cable modems could experience multiple impairments. Thus, the Labeling UI allows a sample to be labeled with multiple impairments. Also, when impairments exist, the level of severity (low, medium, high) needs to be specified. Future versions may allow for more nuanced severities. Note that characterizing severity will likely increase the number of training samples needed to build good models. The labeling effort is designed such that each sample would get multiple responses from different experts, to resolve conflicting labels and to build confidence in the label value.

For the models described in this paper, we labeled approximately 4,000 RxMER samples. Severity of the impairments was not considered while building these proof-of-concept models.

4. Data Preprocessing

OFDM channels in Comcast typically vary in width from 48-96 MHz. This results in an approximate array of 940-1880 RxMER sub-carrier measurements per cable modem, per poll. Since the algorithms in consideration of this paper require a fixed input shape, the following options were evaluated:

- Fixing the width of the spectra to 1,880 by padding the end point with the following options:
  - Zeros
  - The average MER value
  - The last MER value
- Fixing the width of the spectra to a set value (e.g. 900) and applying a smoothing function to transform the raw input to the fixed length input.

As will be seen in the results section, the option of fixed spectra width of 1,880 with average MER value performed the best in our experiments.

5. Model Evaluation

The model evaluation strategy uses the typical train/test split, with 80% of the approximately 4k labeled RxMER samples to be used for training, and 20% kept out for validation. Figure 3 shows the distribution of samples by impairment. About 1,400 of the sample are normal -- i.e. they contain no visible impairment -- while the rest are distributed over several categories. Figure 4 shows the distribution of number of impairments by sample. Approximately 60% of the samples have a single impairment.

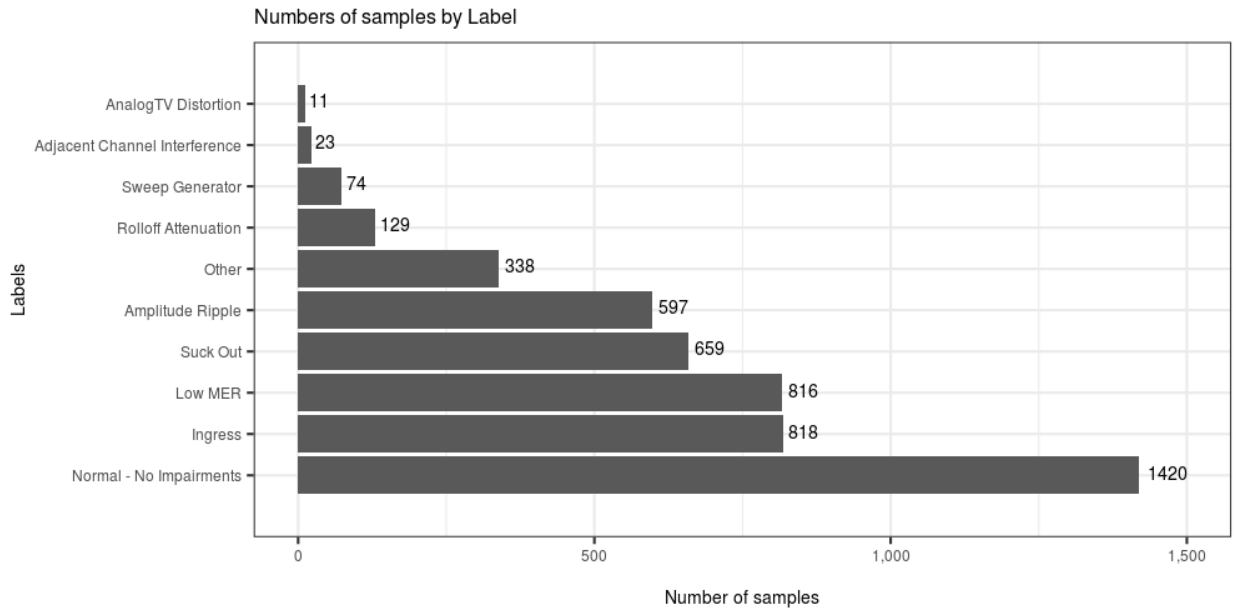

Figure 3. Number of samples by label (impairment type).

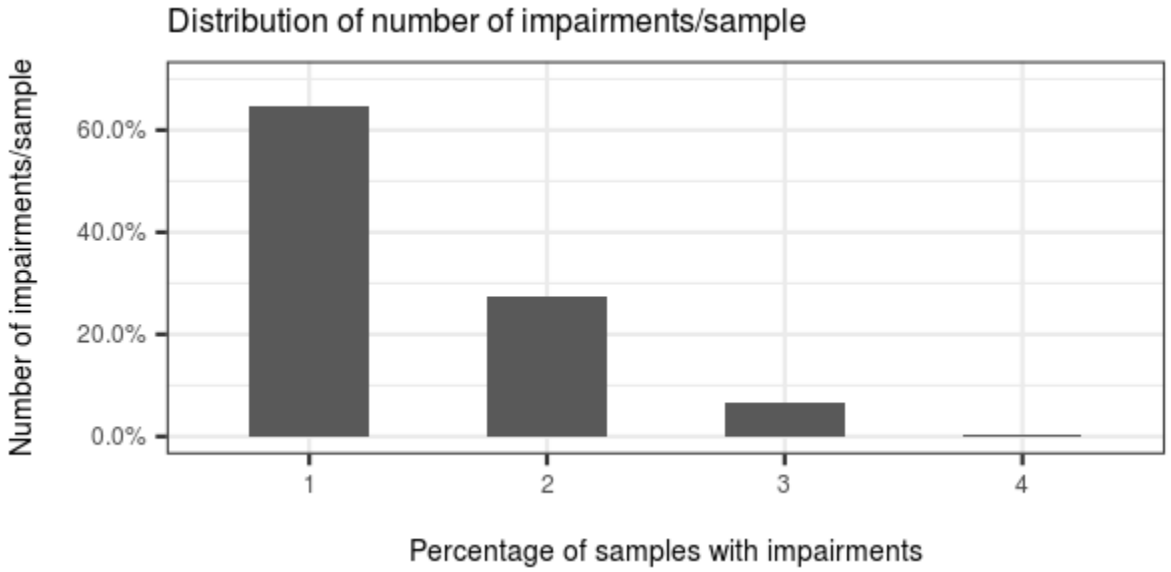

Figure 4. Distribution of number of impairments/samples

Since an RxMER instance for an OFDM Channel can have multiple impairments, detecting impairments is a multi-label classification problem. Predictions for an instance is a set of labels, and therefore the prediction can be fully correct, partially correct or fully incorrect. This makes model evaluation more challenging than binary classification problems, where accuracy, precision, recall and receiver operating characteristic (ROC) curves are typically used as evaluation criteria. In addition to determining accuracy, precision, recall and ROC for individual classes, we will be using the following evaluation criteria:

1. Exact Match Ratio (subset accuracy) – This indicates the percentage of samples that have all their labels classified correctly, given by:

$$Exact\ Match\ Ratio = \frac{1}{n}\sum_{i=1}^{n} I(Yi = Zi)$$

2. Hamming Loss – This indicates the fraction of labels that are incorrectly predicted, given by:

$$Hamming\ Loss\ = \frac{1}{n}\sum \mathrm{i} = 1^{|\mathrm{n}|}\frac{\mathrm{xor}(Yi, Zi)}{|L|}$$

In the formulas above,

- n is the number of multi-label samples
- Yi is the ground truth
- Zi is the prediction
- L is the number of labels

The following are the definitions of Accuracy, Precision, Recall and F1 Score that will be used to evaluate the predictions of individual classes by a model:

$$Accuracy\ = \frac{(TP\ +\ TN)}{(TP\ +\ TN\ +\ FP\ +\ FN)}$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

In the formulas above,

- True Positive (TP) – A true positive is an outcome where the model correctly predicts the positive class
- True Negative (TN) – A true negative is an outcome where the model correctly predicts the negative class
- False Positive (FP) – A false positive is an outcome where the model incorrectly predicts the positive class
- False Negative (FN) – A false negative is an outcome where the model incorrectly predicts the negative class.

6. Convolutional Neural Networks(CNNs)

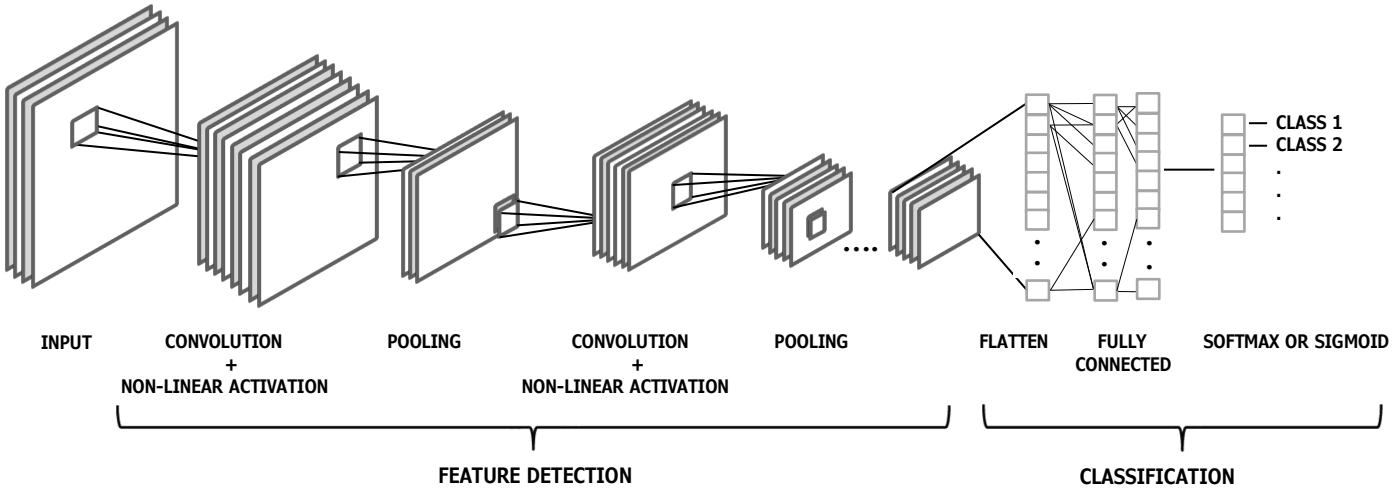

Figure 5. Convolution Neural Network (CNN) components

CNNs are a class of neural networks that have proven to be extremely effective in recent years in the field of perceptual problems, specifically image recognition. They can be used to process data that has a spatial structure. In addition to being used for images that have a 2-D grid, they can also be used for 1-D structures, such as time series. In our use case, we use CNNs to classify impairments based on the 1-D RxMER per subcarrier array for OFDM Channels.

The model architecture is shown in Figure 5. CNNs contain the following components in addition to the input and output layers:

- Convolutional layers: Convolution layers use filters that perform convolution operations to scan the input layers with respect to dimensions. Convolution layers perform several convolutions in parallel, to produce a set of linear activations. Then, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function, to generate a feature or activation map.
- Pooling layers: Pooling layers are typically applied after a convolution layer and perform a down sampling operation. They replace the output of the feature map with a summary statistic, such as max or average of the nearby outputs.
- Fully connected layers: Fully connected layers are typically present toward the end of a CNN architecture, and operate on a flattened input where each input is connected to all neurons.

There is no single network architecture, and understanding the network architecture is an area of great study. CNNs typically have a series of convolutional and pooling layers that are stacked together. The features that the convolutional/pooling layers detect increase in complexity as we go further down the network[1]. The convolutional and pooling operations provide translation invariance that uniquely distinguishes CNNs from other types of neural networks. Invariance refers to being able to recognize an object even when its appearance varies in some way.

7. Modeling

We experimented with CNN architectures that had between 1-3 convolution blocks and 1-3 fully connected layers. A grid search was performed on the following hyper-parameters before selecting the final model:

Table 1. Hyperparameters, Ranges evaluated during training

Hyperparameter	Range	Hyper-parameter Type
Number of filters in convolutional layers	[32, 64, 96, 128]	Network Structure
Kernel size in convolutional layers	[3,5,7,9]	Network Structure
Pooling Size	[2,3,4]	Network Structure
Fully connected hidden layer size	[100, 150, 200, 250]	Network Structure
Dropout	[0.3, 0.4, 0.5]	Network Structure
L2 Regularization	[0, 0.0001, 0.0005, 0.001, 0.005, 0.01]	Network Structure
Learning Rate	[0.0001, 0.0003, 0.0005, 0.001, 0.003, 0.005, 0.01, 0.03]	Network Training
Batch Size	[16, 32, 64, 128]	Network Training

In addition, we evaluated the different methods to get an input of fixed shaped described in the data pre-processing section.

8. Results

We compared our models to 2 dummy models – one that has all labels assigned the most frequent class and another that randomly assigns labels based on the distribution in the training data. These 2 data points serve as benchmarks to compare against. In addition, we also compared the performance to a 3-layer conventional neural network.

Table 2. Subset Accuracy, Hamming Loss for models

Model	Subset Accuracy	Hamming Loss
Model that predicts most frequent class	0.319	0.163
Model that randomly assigns labels based on distribution	0.146	0.202
Neural Network – 3 dense layers	0.645	0.047
CNN – Zero padding at end to get 1880 Subcarriers	0.778	0.031
CNN – Average MER padding at end to get 1880 Subcarriers	0.852	0.018
CNN – Last MER seen padding at end to get 1880 Subcarriers	0.824	0.022
CNN – Fixed width of 900 Subcarriers	0.809	0.024

As seen above, the CNN-based models significantly outperform the dummy and the neural network-based models on the validation dataset. While all the CNN based models had results in the same range, the model with average MER padding performed the best in our experiments.

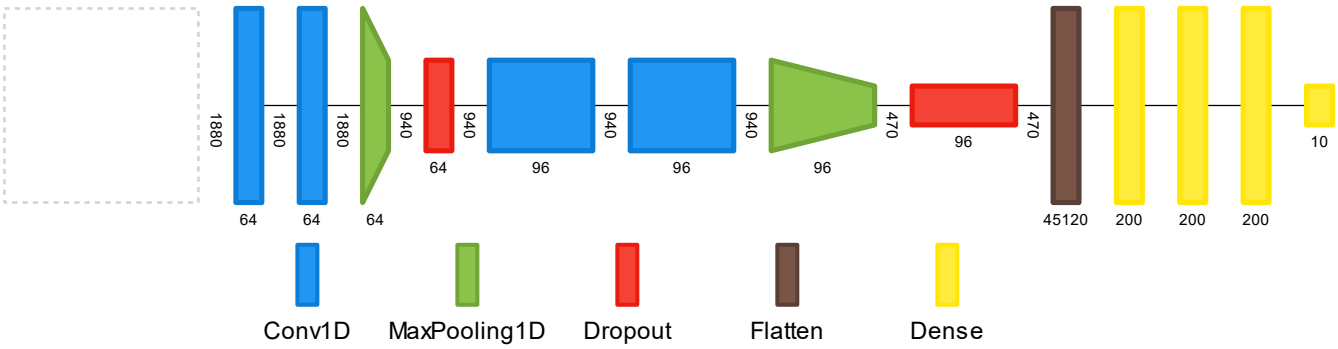

Figure 6. Network Architecture CNN model that had best performance on validation dataset [2]

All the classes have accuracy above 90% on the validation dataset. Apart from amplitude ripple, they also have very good precision, recall and F1 scores. Due to its fine-grained nature, amplitude ripple does not perform as well and probably needs more training samples to get better scores. Another observation from the results is that incorrect classifications were often attributable to multiple impairments being present. This also indicates the need for additional labeled data.

Analog TV distortion and adjacent channel interference had fewer than 5 samples in the validation dataset and were not evaluated.

Table 3. CNN model with average MER padding - Individual classes performance metrics on validation data

Impairment	Accuracy	Precision	Recall	F1 Score
Normal – No Impairment	0.989	0.976	0.992	0.984
Ingress	0.980	0.975	0.935	0.954
Sweep	0.997	0.933	0.933	0.933
Roll-off	0.996	0.938	0.968	0.952
Suck out	0.974	0.939	0.911	0.925
Amplitude Ripple	0.939	0.912	0.715	0.802
Low MER	0.974	0.958	0.924	0.940

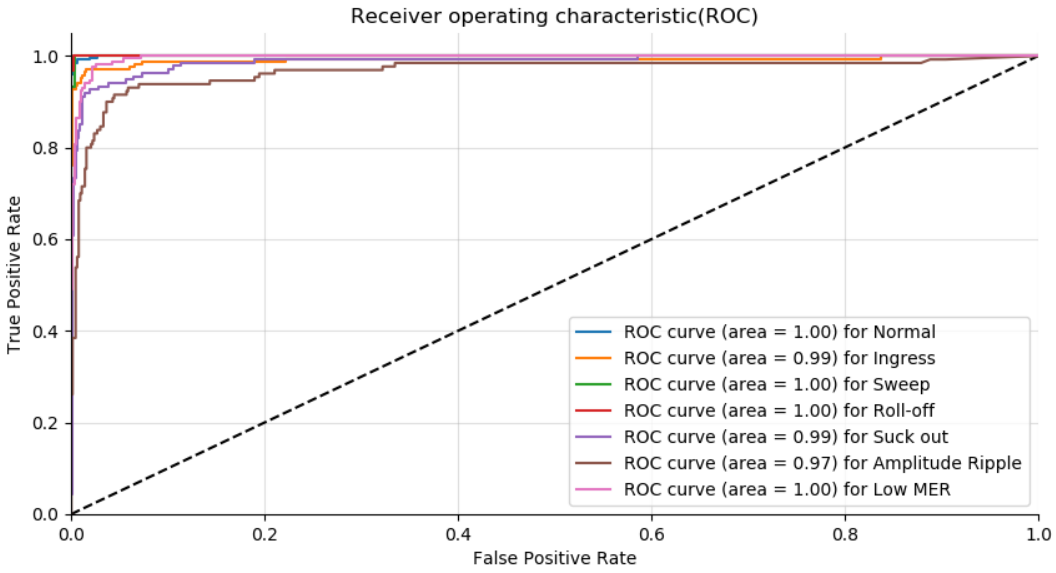

Figure 7. ROC Curves

9. Machine Learning Pipeline

At a high level, the platform driving proactive network maintenance consists of 3 layers – Data Lake, Compute Engine and Integration Layer, as depicted in Figure 8.

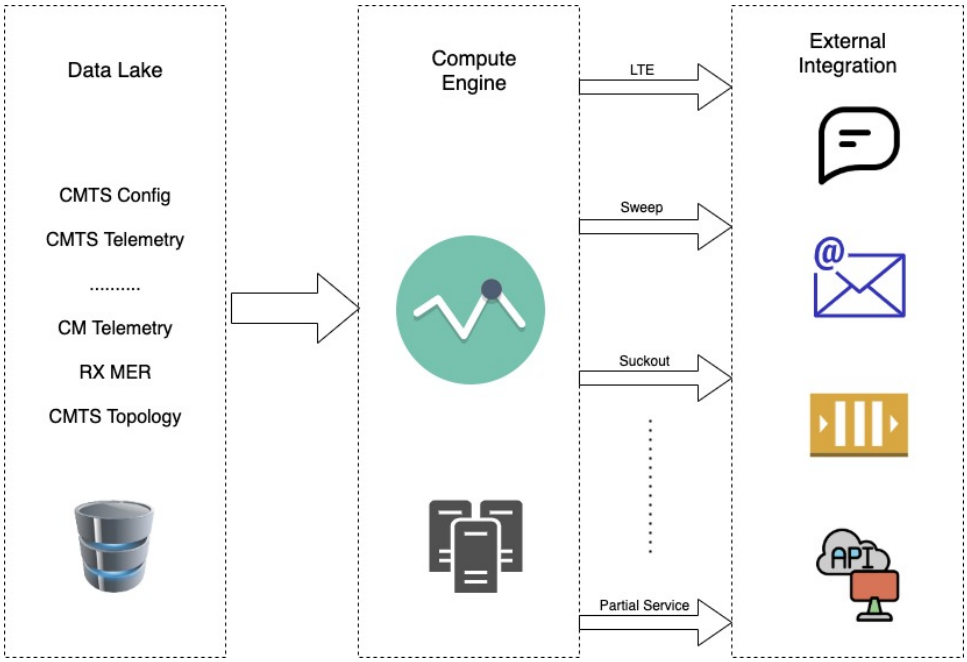

Figure 8. Machine Learning Pipeline Layers

9.1. Data Lake

Comcast's cloud-based data lake storage solution plays a pivotal role in our efforts to apply data science and develop analytical solutions to better understand and build a highly dynamic, resilient access network. The data lake also acts as a source of truth for a wide variety of data streams across Comcast. While the primary application of this data lake is to drive ML and analytical applications, custom integration and other solutions built on top of this data lake help power a wide variety of use cases. The platform is highly elastic, reliable and offers a rich set of tools for our analysts, engineers and data scientists to easily access this data and form a unified view of our network, customers and devices. It helps them use this data for analysis, visualization and ML applications, without the technical barrier of knowing the underlying infrastructure. The queries on the underlying data run on a Spark-based distributed computation engine, which is purpose-built to handle large data sources.

9.2. Compute Engine

The compute engine refers to the actual implementation of feature pipeline and ML models on polling data from the data collection framework. The standard ML life cycle involves model development, experimentation, training, test, validation and supporting model evolution. These ML models could be custom implementations developed from scratch or based on popular ML frameworks. During the development phase, the Compute Engine provides support to track and compare metrics for various experiments involving different models, features, tuning hyperparameters, etc. It also helps track code, data and model lineage while supporting promotion of models between different stages. To build the Compute Engine, we use open source ML frameworks and enhance it with custom functionality to suit our pipeline.

9.3. Integration layer:

The integration layer provides support to successfully integrate the recommendations from the pipeline to various OSS tools or directly to SMEs and technicians, through a broad set of interfaces and tools. The more seamless programmatic integration involves APIs and streaming events that would integrate with other OSS tools within Comcast. In severe cases, notifications could also be evented through Comcast's IM application, SMS and emails so they can be attended with greater urgency.

As a standard route, consumers would subscribe to the streams through Comcast's streaming data platform, which would event out our impairment notifications at pre-defined intervals. Once these events are consumed, they result in tickets/work orders being created automatically. This event-driven architecture enables any number of tools to easily integrate with our ML platform and track network events in real time. The following data are provided as part of notifications to assist with event prioritization and triangulation:

- Interfaces details such as PLC Location, Start/End Frequency, Total/Impacted cable modem counts
- List of impacted nodes
- List of impacted cable modems with severity of impairments
- Interface impairment rankings at the national, divisional and regional levels
- Reference to the API for historical data stored in the data lake related to the event
- Reference for API to enable fix agents to collect real-time on demand data, to confirm the issue is still present and to assist in isolation and confirm mitigation
- Other data sources to enrich the event, such as the mobile wireless carriers that overlap with the OFDM Channel

We also provide a real-time API that OSS tools integrate with and can be invoked on an on-demand basis. Once the impairments are attended to and a fix is identified, the network technicians can invoke the API through the UI of the OSS tools. Once a request is made to our API, the devices on the relevant interfaces/nodes are polled in real-time. Those polling results analyzed and scored through the model and a response is sent back to the UI, which indicates if the identified fix resulted in clearing the impairment. If the impairment is no longer seen, the OSS tool automatically clears the ticket. It is important to note that the compute engine handles model management for both offline scoring of the models for the entire footprint, and real-time scoring based on live polling data.

10. Conclusion/Next Steps

We've seen very promising results in being able to classify impairments for OFDM channels using CNNs on our validation dataset, with a high degree of accuracy. However, we need to continue to iterate and use additional labeled samples to address the following:

- The samples we've used may not cover the gamut of RxMER curves for the entire footprint
- There are some impairments that had very few examples
- Impairments that were incorrectly classified were often due to multiple impairments being present
- Detecting severity in addition to the impairment

A crowdsourcing approach to labeling that involves field technicians and other SMEs would be beneficial and increase confidence in the models being developed. SMEs are also needed for addressing edge cases, where the impairments were not always well-defined.

In the models/data pre-processing phase, there are a couple of areas that can be explored further:

- Increase the number of training samples by using data augmentation techniques such as flipping the RxMER curves horizontally or scaling the entire sample. We'll need to consider that some augmentations cannot be applied to all impairments, as it would modify the RxMER curve in a way that makes the impairment no longer applicable. For example, samples with roll-offs cannot be flipped horizontally.
- Impact of data pre-processing techniques on models needs to be understood better. For example, a sample having roll-off may be incorrectly classified as having suckout, due to the average MER padding added to get to the standard width.

We will also be extending this effort beyond OFDM channels to cover upstream, video and D3.0 channels. For OFDM channels specifically, building models for classifying impairments that include both RxMER and RxPower will be explored.

Abbreviations

API	Application Programming Interface
CMTS	Cable Modem Termination System
CNN	Convolutional Neural Network
D3.0	DOCSIS 3.0
D3.1	DOCSIS 3.1
FN	False negative
FP	False positive
HFC	Hybrid Fiber-Coaxial
MER	Modulation Error Rate
ML	Machine learning
MTTR	Mean Time To Repair
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OSS	Operational Support Systems
PMA	Profile Management Application
PNM	Proactive Network Maintenance
RxMER	Receive Modulation Error Rate
ROC	Receiver Operating Characteristic
QAM	Quadrature Amplitude Modulation
SCTE	Society of Cable Telecommunications Engineers
SME	Subject Matter Expert
SNR	Signal-to-noise ratio
TN	True Negative
TP	True Positive
UI	User Interface

Bibliography & References

1. *Visualizing and Understanding Convolutional Networks* - https://arxiv.org/pdf/1311.2901.pdf

2. *Net2Vis -- A Visual Grammar for Automatically Generating Publication-Ready CNN Architecture Visualizations, Alex Bäuerle, Christian van Onzenoodt, Timo Ropinski* https://arxiv.org/abs/1902.04394

