Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Multitask GRU Learning for Channel Disturbance Detection, Cause Classification, and QBER Forecasting in Simulated Decoy-State BB84

Mayank Goswami

mayankfph@iitr.ac.in

Indian Institute of Technology Roorkee

Priyanshu Singh

Indian Institute of Technology Roorkee

Research Article

Keywords: Quantum Key Distribution, Quantum Bit Error Rate, QBER Forecasting

Posted Date: November 5th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11618502/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Multitask GRU Learning for Channel Disturbance Detection, Cause Classification, and QBER Forecasting in Simulated Decoy-State BB84

Priyanshu Singh[1] and Mayank Goswami,[1,2]

[1]Department of Physics, IIT Roorkee
[2]MF School of AI and DS, IIT Roorkee
mayank.goswami@ph.iitr.ac.in

***Abstract*—Supervising a QKD link involves identifying variations in channel behaviour, while also considering noise arising from finite sampling. They can also be used to predict future values of QBER. A PyTorch gated recurrent unit (GRU) model is explored which includes a shared sequence encoder taking the recently reported count and QBER values. This GRU model includes three heads, one for detecting disturbances, another for determining the cause of the disturbance (multi-label classification), and the third for predicting QBER. 800 training sessions, 160 validation sessions and 320 test sessions were generated using the decoy-state BB84 simulator. All models contain 5 observed features and 3975 trainable parameters. The average recall per interval was 0.9622 across the three training seeds, with a false positive rate of 0.0081. The macro-F1 score for determining the cause was 0.9837. The prediction error for horizon 1 was 0.4762, for horizon 5 was 0.7055 and for horizon 10 was 1.0066. The model raised at least one alarm for all test events but not all disturbed intervals were detected. The model was tested to ensure the integrity of the guarded calibration, logit based classification loss and bounded predictions. However, slow drifting and transition between events proved challenging. These results show that the model can be used for joint supervised supervision in this simulator. However, it cannot be guaranteed to perform well on physical links, causes not seen during training will not be detected and does not guarantee security of the QKD.**

I. Introduction

A. Need for monitoring

We cannot detect why the QBER has changed. The error rate could increase due to a misalignment, background light could cause additional detections and errors and we could have fewer observations due to the loss. This increases the uncertainty in the measured error rate. Using only the QBER to monitor could lead to confusion between the sample variance and actual changes to the channel. Useful information is gained by using counts from varying pulse intensities. In our simplified model the vacuum counts depend on the background yield. The signal and weak-decoy counts also depend on the transmission. By observing variances in these we can better determine if there is loss or background issues. The classifier will be able to detect the underlying mechanisms of the simulation. However we cannot tell if there was an accidental, environmental or adversarial interference based on the label. The forecasting task asks to predict what QBER we could expect to observe in the next few intervals. A new event can start at a random interval meaning we may not see any indication before the event begins in the telemetry. We measure the error in the forecast for multiple horizons. We also give the error for targets where the event state does not change between the input and target interval. B. Research Questions and Evaluation Criteria Can a small GRU be trained to perform all three tasks using previous and current count and QBER data over the simulated channel range? We measure the success of this using:

1) Detect intervals with active interference based on a threshold (set using validation non-events). Recall, false positive rate and event coverage for intervals.

2) Identify when there is a misalignment, high loss and/or background interference. F1 score per cause and exact agreement for event intervals.

3) Predict the measured QBER for 1, 5 and 10 intervals ahead. Mean absolute error (MAE) for multiple horizons and subsets of targets.

4) Observe performance differences between scenarios, fibres lengths and severities. Ensure results are correct based on the data pipeline we have implemented and predictions.

As we did not have any thresholds to pass before running the experiment we evaluate how well our model performed based on the results from the hold-out set. We do not claim our model achieved a certain target as we do not know how it would perform outside of our sampled range. C. Limitations This experiment uses supervised learning based on simulated telemetry. We do not have a complete QKD stack. We do not produce any final secret key. We do not evaluate any decoy-state yield bounds, error correction, privacy amplification or composable finite-key analysis. A separate analysis should be carried out to achieve finite-key security [3]. We are producing monitoring estimates rather than key security certificates.

II. Problem, Objectives, and Motivation

A. Why channel monitoring is needed

The change in QBER alone does not provide information into why the change has occurred. There could be an alignment issue causing an increase in error fraction. There could be stray photons causing extra detections and errors. Or there could just be loss causing a reduction in useful samples. This will result in a less certain measurement in error rate. If only using the QBER to monitor then there could be confusion between statistical fluctuations and the channel changing.

Using multiple pulse intensities provides extra information. With our simple model we see that the vacuum sample count depends on the background yield. Signal and weak decoy samples depend on the transmissions. Fluctuations in any of these can help us differentiate between loss and background issues. The classifier is trained to know the underlying causes from our simulation. However, knowing the label does not tell us if the disturbance was due to random, environmental or adversarial causes.

In forecasting we want to know what QBER could be seen in the near future. The event can start at any random time so we may not have any information on when it started from the telemetry. We measure the error in our forecasts over multiple look aheads. We also measure the error for targets where the event state does not change between the input and target.

B. Questions and Evaluation Criteria

Can a small GRU provide all three tasks using historical and current sample and QBER measurements for the full range of simulated channels? To answer this we have four objectives: Detect intervals of active disturbance with a threshold set based on validation intervals with no events. Measured by interval recall, false positive rate and event coverage. Identify if there are alignment issues, high loss, background issues and any combination. Measured by F1 score for each cause and perfect agreement during event intervals. Make forecasts for the measured QBER 1, 5 and 10 intervals ahead. Measured by MAE over different look ahead intervals and subsets of targets. Investigate how the model performs across different scenarios, fibre lengths and severity groups. Ensure that the results provided are due to the data pipeline and predictions implemented. As we did not set any numerical thresholds for passing the experiment we are using the holdout results to evaluate our model. We do not claim that our model passed any specific target. We have shown that within the sampled range it performed well but do not know how it would have performed outside of this range.

C. Limitations

This experiment uses supervised learning with simulated telemetry. We do not have a complete QKD stack. There is no final secret key generated. We do not evaluate any of the following: decoy state yield bound, error correction, privacy amplification or composable finite key analysis. We would need to perform a separate analysis for finite key security [3]. The results provided are monitoring estimates not key security guarantees.

III. Theoretical Background

A. BB84 and decoy-intensity observations

In the BB84 protocol the sender and receiver choose preparation and measurement bases. When the sender and receiver choose the basis those matched-basis outcomes help the sender and receiver estimate error and in the full protocol produce key generation. The simulator in this case picks Z-basis matches with a probability of 0.25 then takes half of those Z-basis matches to estimate the QBER. The simulator only shows counts and does not keep track of individual quantum states through the complete protocol.

A phase randomized coherent pulse with mean photon number μ has a Poisson photon number distribution as,

$$P(n \mid \mu) = e^{-\mu}\frac{\mu^n}{n!}. \tag{1}$$

Different intensity sources are used for decoy-state analysis [2], so that we can have three different monitoring streams for signal, weak-decoy, and vacuum intensities of means (0.5, 0.1 and 0). These are chosen with probabilities (0.7, 0.2, 0.1). This setup is consistent with a decoy-state protocol, however it is not performing a decoy-state security estimation.

B. Transmission, clicks, and errors

Let L be fiber length in kilometers, η_d detector efficiency, and ℓ_t additional time-varying loss in decibels. Overall transmission is

$$\eta_t = \eta_d 10^{-(0.2L+\ell_t)/10}. \tag{2}$$

For intensity k, the probability of a signal-induced click is

$$s_{t,k} = 1 - \exp(-\eta_t \mu_k). \tag{3}$$

While background yield $Y_{0,t}$ and alignment error $e_{d,t}$, the implemented click and error probabilities are

$$Q_{t,k} = s_{t,k} + (1 - s_{t,k})Y_{0,t}, \tag{4}$$

$$E_{t,k} = \frac{e_{d,t}s_{t,k} + 0.5(1 - s_{t,k})Y_{0,t}}{Q_{t,k}}. \tag{5}$$

The background will be modeled to generate random bit results. The $1 - s_{t,k}$ term removes the simulated background term from registering a second click when the signal is registering a single one. These equations are the counting model that the simulator will compute, they do not include dead time, after pulsing or other device effects.

For a sampled signal count n_t, the error count follows

$$B_t \sim \text{Binomial}(n_t, q_t), \qquad \widehat{q}_t = B_t/n_t, \tag{6}$$

where $q_t = E_{t,\text{signal}}$. Measured QBER is marked missing when $n_t < 10$. QBER is represented internally as a fraction; a forecast error of 0.005 corresponds to 0.5 percentage points (pp), not a relative error of 0.5%.

C. Sampling variation and calibration

For a binomial estimate with known probability q, the standard deviation is $\sqrt{q(1-q)/n}$. The implementation substitutes the mean QBER from a clean calibration interval, q_{ref}, to construct

$$z_t = \frac{\widehat{q}_t - q_{\text{ref}}}{\sqrt{q_{\text{ref}}(1-q_{\text{ref}})/\max(n_t, 1)}}. \tag{7}$$

This scaling adjusts for how the number of samples changes under a reference model. It doesn't completely remove the effect of fiber length on the feature distribution. Also it doesn't include any uncertainty, in the estimated calibration mean. The calibration mean is kept within the range $[10^{-4}1 - 10^{-4}]$. If all calibration QBER values are missing the system uses 0.02 as a fallback. These steps help keep the math from going to infinity.. The fallback value of 0.02 should be checked and confirmed for real-world links.

D. GRU sequence encoding

Gated Recurrent Units (GRUs) store and update memory states using gates [4], where for an input vector of x_t and previous hidden state h_{t-1}.

$$z_t^g = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

$$\tilde{h}_t = \tanh(W_h x_t + b_{ih} + r_t \odot (U_h h_{t-1} + b_{hh}))$$

$$h_t = z_t^g \odot h_{t-1} + (1 - z_t^g) \odot \tilde{h}_t$$

Where superscripts denote that the update gate z_t^g is different to the QBER discrepancy z_t in equation 7. The reset gate applies to the projected recurrent signal in line with PyTorch GRUs [5]. Finally, h_t provides a summary of the current 20 interval window. Note that all windows start with a zero hidden state, so no hidden state information is passed between sessions or windows.

A GRU was selected as this is a simple recurrent network which can learn from time-series data. As architecture comparison is not performed in this study, it is not shown whether the GRU would perform better than other architectures such as LSTM [6], CNNs, and attention mechanisms.

IV. Previous Work

ML has also been employed for multiple separate QKD problems. Zhang et al. [7] examine LSTM-assisted calibration reference-frame in measurement-device-independent QKD. They analyze a calibration protocol while the current work predicts a disturbance state and estimated QBER in a future round. Both involve time-dependent data, but protocols, outputs, and conditions are quite different.

Maragkopoulos et al. [8] analyze channel-impairment diagnosis from QBER and secret-key-rate time series. They serve as a relevant baseline in reading the operational telemetry. The current experiment employs intensity-specific counts and measured QBER obtained using a known, controlled simulator, with explicit identification of three injected mechanisms, and extends the supervised monitoring formulation with a joint forecasting head, without asserting novelty in learning-based QKD diagnosis.

Luo et al. [9] explore deep-learning-based attack detection in continuous-variable QKD. Liu et al. [10] train one-class anomaly detection classifiers for active attacks on a QKD receiver. These papers tackle security-related monitoring, but in environments a present with a different i.i.d. input disturbance problem than our labeled disturbance problem. Specifically, learning a normal class is not the same as training on each category of the disturbance set.

The quantitative figures of those papers are not an immediate baseline for our experiment. Variations in protocol, measure, description of the attack, sampling condition, or split, make an accuracy comparison alone insufficient to reflect model quality. The function of related work is to set the usage context and to identify the supervised count-and-QBER formulation. The claim of no novelty does not rely on the absence of other multitask or recurrent QKD work.

V. Proposed Method and Evaluation Design

A. Session generation

Each session consists of 240 intervals each containing 2 million emitted pulse samples. For each intensity level, the number of emitted counts is sampled from a multinomial distribution. Detected counts follow a binomial distribution, and matching and error samples are taken. The fiber length is uniformly distributed between 15 and 70 km, the detector efficiency is between 0.10 and 0.20 and initial error is between 0.01 and 0.02. The background count rate prior to any disturbance is 10^{-5}.

A slow varying background is generated. The alignment has a retention factor of 0.97 and innovation variance of 0.0005, while the loss has a retention factor of 0.98 and innovation variance of 0.03 dB. This represents assumptions about how the data would be generated.

There are 8 scenarios, each having equal weighting. These are: normal, slow alignment drift, sudden alignment jump, high loss, increased background, loss with misalignment, background with misalignment and all mechanisms. The first 30 intervals are clean calibration data. One disturbance event occurs within the session, starting at an integer between 60 and 149 and of length between 40 and 79 intervals. The last disturbance must occur before the end of the session.

The severity (v) is uniformly distributed between 0.2 and 1.0. Active mechanisms have their base amplitude values sampled independently and multiplied by v. Alignment is increased by a factor of v where the value is between 0.03 and 0.08, loss is increased by a factor of v where 8-16 dB is added, and the background count rate is

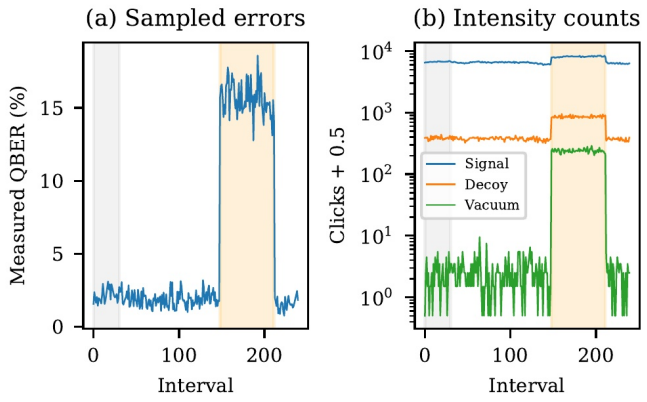


Fig. 1. Held-out background-plus-misalignment session. Gray indicates clean calibration and orange the event. Panel (b) adds 0.5 to counts for logarithmic display. This session belongs to the same test split used in the evaluation.

TABLE I
DATA PARTITIONS AND EFFECTIVE EXAMPLES.

Partition	Sessions	Per scenario	Windows
Train (seed 1)	800	100	168,000
Validation (seed 2)	160	20	33,600
Test (seed 3)	320	40	67,200

increased by a factor of v where the value is between 0.001 and 0.003. Alignment has a linear increase between 0 and the sampled alignment increment. All other mechanisms have a sudden increase. Labeling of events means that the start of the alignment increase will be labelled, even if the physical increase starts at 0. This can cause some events to appear almost identical to normal data. An example of a combination disturbance is shown in figure 1.

B. Independent-session splits

The split can be seen in Table I. Each window from a session is in the same partition. The simulator has different seeds for generating the training, validation and test sets. This keeps the scenarios balanced at session level. There are fewer event intervals as every session containing disturbance also contains clean intervals. Around 25.1% of windows in the training data finish during an event.

The training and test datasets share a consistent fiber-length range. The test, thus, measures performance on independent sessions drawn from a familiar range; it does not measure extrapolation to substantially longer links. Among the testing sessions, there are 280 with disturbance and 40 without disturbance.

C. Observable features and causal windows

Calibration provides q_{ref} and the mean count $C_{\text{ref},k}$ for each intensity. The input vector contains five features:

$$x_t = \left[\frac{\widehat{q}_t^{\text{fill}}}{0.05}, \operatorname{asinh}(z_t), \left\{ \log \frac{C_{t,k} + 0.5}{C_{\text{ref},k} + 0.5} \right\}_{k=1}^{3} \right]. \quad (8)$$

QBER is forward filled from the last non-null value, with the deviation feature being set to zero. The count offset ensures the log does not result in a null. Features are mean and standard deviation scaled using only the training window.

For prediction at interval $t \geq 30$, we observe from $t-19$ to t. All calibration is complete before we make any decisions to evaluate. As the values from calibration may be present in the past part of the input window, all values will be known by the time we make our decision. No future observations, causes or true probabilities from the simulator are included in the model inputs.

Targets are made up of the event indicator at t, the three active cause indicators at t and the observed QBER at $t+h$ for $h \in \{1, 5, 10\}$. If we have a forecast target outside the session or there is a missing QBER then the target will be masked. In practice this will just be stored as a zero however it will be masked out of the loss and metrics. We have 196,480 valid forecast targets in the test set. None of the sampled sessions in the recorded experiment had any missing QBER. There are masks due to the targets being at the end of the session.

D. Shared network and training objective

The model uses `torch.nn.GRU` with five inputs, 32 hidden units and one recurrent layer. Including the three output heads the model has 3,975 parameters. The final hidden state of the model produces one event logit, three independent cause logits and three forecast logits. The model applies sigmoid transformations to produce event. Cause probabilities. Independent cause outputs in the model allow mechanisms; the model does not impose softmax competition.

The forecast, at horizon h is $\widehat{q}_{t+h} = \sigma(a_{th})$. Therefore the forecast produced by the model lies in $[0, 1]$. The model initializes forecast bias using the valid training-target QBER clipped away from the endpoints. This step uses training information. A prediction that falls within the interval is not necessarily accurate so the model reports forecast validity and forecast error separately.

The objective is

$$\mathcal{L} = \mathcal{L}_{\text{event}} + \mathcal{L}_{\text{cause}} + \mathcal{L}_{\text{forecast}}. \quad (9)$$

The classification terms use PyTorch's binary cross-entropy with logits. For logit a and binary target y, this computes the stable expression

$$\ell(a, y) = \max(a, 0) - ay + \log(1 + e^{-|a|}). \quad (10)$$

Event loss is averaged over examples and cause loss over example–cause pairs. Forecast loss is a masked Huber

penalty with unit transition point on the scaled error $e = (\widehat{q} - q)/0.05$:

$$H(e) = \begin{cases} e^2/2, & |e| < 1, \\ |e| - 1/2, & |e| \geq 1. \end{cases} \tag{11}$$

Only true future targets are considered. This means that although we have equal loss coefficients, the contribution of each will be affected by how they are scaled. We make no single task or loss weighted ablations.

The gradients are calculated using PyTorch autograd [pytorch]. We use Adam optimizer [kingma2015] with a learning rate of 0.003 and train in batches of 256 windows. Gradients are clipped by global norm of 1. We train for a maximum of 25 epochs with patience of 4. We save a checkpoint if the validation loss increases by over 10^{-4}. We seed the initializations and batching with 42, 43 and 44 respectively. The test set is not used to tune parameters or choose checkpoints.

E. Thresholds and Baselines

Event thresholds are calculated as the 99th percentile of the model's probabilities for non-event intervals in the validation set. These are then compared to the test set. Thresholds for cause detection are 0.5. There is no guarantee that the false positive rate will exactly be 1% due to our target of having 1% false alarms in the validation set.

In this experiment we only train and evaluate the multi-task GRU. Control charts and persistence were previously trained for different implementation so these values only indicate how well they perform in the simulator setting but cannot indicate any improvements over alternative predictors. Comparisons to other neural sequence models is future work.

F. Metrics and Uncertainty

We calculate precision, recall, F1, FPR and AUROC for our detection. Coverage is how many injected events are covered by our models, this means that if any alerts are raised during the event interval then the event is counted as being covered. Novel coverage means that the model must transition from false to true during the event. Detection delay is how long it takes to raise the first alert after the start of the event, given that it was covered. It is possible to have 0 detection delay if the alert was already raised at the start of the event.

F1 for causes is calculated per label for all decision intervals, giving a macro average. MAE for forecast is calculated using true targets. These are presented in percentage points (pp). Quiet targets are those where there is no event state change between the decision and target. Here we only need to consider the endpoint state as there is a maximum of one event per session and each event will have a minimum length of over 10 intervals. Quiet targets can still have drift and sampling noise.

TABLE II
Interval detection performance: mean and sample SD across three seeds.

Metric	Mean	SD
recall	0.962247	0.001184
fpr	0.008077	0.001001
f1	0.968574	0.000895
auroc	0.991481	0.000110

TABLE III
Event-level coverage. Any alarm and new alarm use different counting rules.

Seed	Any alarm	New alarm	Delay
42	280/280	279/280	0
43	280/280	279/280	0
44	280/280	279/280	0

We present averages and sample standard deviation across 3 seeds for training. This indicates how variable the model is when training with 1 data split. This is not an indicator of confidence intervals for new data distributions or hardware. Length, severity and scenario breakdowns are done using seed 42.

VI. Results and Verification

A. Disturbance detection

The mean test recall is 0.9622, the precision is 0.9750, F1 is 0.9686, and AUROC is 0.9915. The mean FPR is 0.0081. Table II reports training variation. The threshold was selected before the evaluation of the test, using only normal validation intervals.

All three models have alarms for all 280 injected events. However, it doesn't imply that all disturbed periods are identified or that the model anticipates the start of the event. The new alarm count shown in Table III excludes instances where there was no new transition in the event. The false positive rate in normal sessions is 0.00766 since only the 40 sessions with no events were considered. The false positive rate given in the table is in reference to all the non-event periods.

B. Cause Classification

Overall macro F1 score is 0.9837. The average F1 score per cause is 0.9520 for alignment, 0.9991 for excess loss and 1.0000 for background causes. For all three causes, the labels will be identical for 93.83% of event periods. The causes included are those present in both the training and test set. For seed 42, the gradual drift has a lower recall than the abrupt alignment steps with a value of 0.7584 compared to 0.9923. Gradual drift starts with a small increase in the physical value and the event label will have been turned on at this point. It can be seen that the early drift periods can be difficult to identify due to how the labelling was applied.

TABLE IV
FORECAST MAE IN PERCENTAGE POINTS, MEAN ± SAMPLE SD.

Horizon	All targets	Quiet targets
1	0.4762 ± 0.0069	0.4228 ± 0.0066
5	0.7055 ± 0.0023	0.4465 ± 0.0046
10	1.0066 ± 0.0077	0.5010 ± 0.0075

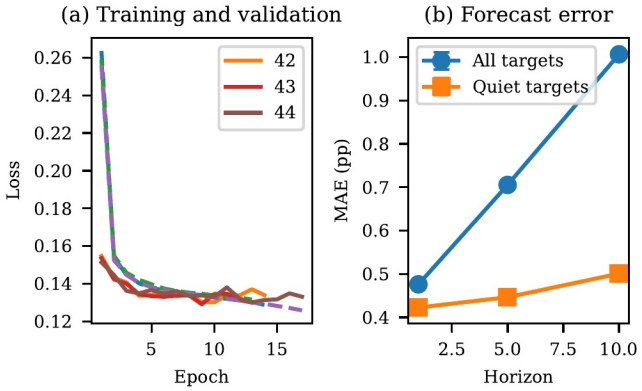


Fig. 2. (a) Training (dashed) and validation (solid) loss for the three seeds. (b) Forecast MAE with sample-SD error bars. The curves summarize the same data split.

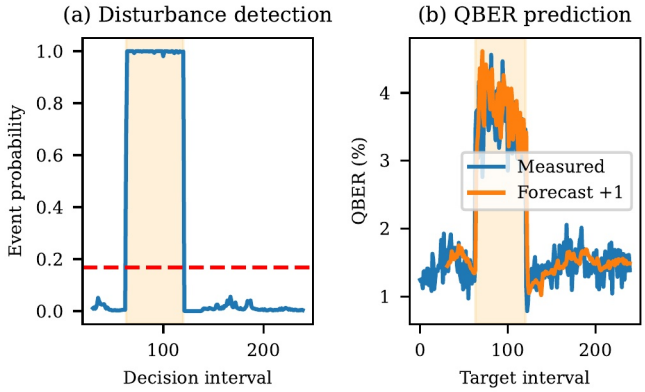


Fig. 3. Held-out alignment-step session, seed 42. (a) Event probability and validation threshold. (b) Measured QBER and one-step forecast aligned to target time. Orange denotes the injected event.

TABLE V
SEED-42 LENGTH GROUPS; MAE IS FOR ONE-STEP PREDICTION.

Length (km)	Recall	FPR	MAE (pp)
15-30	0.9707	0.0126	0.3161
30-50	0.9543	0.0084	0.4069
50-70	0.9659	0.0074	0.6313

C. QBER Prediction

As shown in table IV the prediction error increases as the prediction window increases. The quiet target has lower errors as it excludes the onset and recovery transitions. However, the target still has noise and changes during events. A prediction aligned to a future target shouldn't be seen as having seen that target. An example of single step predictions aligned to their prediction period can be seen in figure 3.

D. Fiber-length and severity breakdowns

Table V looked at the groups seed-42 predictions by fiber length. The sessions were not used to fit the model. The sessions lengths remain within the training range. When the received counts are low the sessions uncertainty, in measured QBER increases. The table is descriptive because the channel length, the detector efficiency and the event severity all vary across sessions..

E. Implementation verification and adverse findings

The tool assesses finite features for calibration QBER values - zero, nominal and missing. It tests causal inputs, so that its perturbations under future-observation evaluations do not affect verification. It evaluates input robustness by confirming labels, and latent true QBER do not enter features. Checks of shape target-mask gradient-flow and extreme-logit pass. The evaluation of checkpoint replay is done separately from retraining. This is because loading saved weights is used to test inference reproducibility while fresh training can differ due to differences in software versions and platforms.

The run, as reported, ran PyTorch 2.11.0 on CPU, NumPy 2.1.3. Re-running the tests with the same version of NumPy reproduced the exact saved test labels. The results obtained by loading the checkpoints with PyTorch 2.6.0 show a maximum absolute difference of 3×10^{-7} across our tests this check across versions shows numerical agreement, not bitwise identity. A new run of local training, with a different version of NumPy, produced other data and score, which is kept separately.

The new prediction arrays let you compute detection, classification and forecast metrics independently. The evaluated outputs do not contain out-of-range QBER prediction. Since the sigmoid enforces the range of numbers, this check doesn't mean that the prediction error is small in all regions. The previous infinite output and probability-space BCE defects are historical defects observed in the NumPy implementation. They aren't reported as failures in the present PyTorch run.

The remaining adverse findings are performance limitations. Gradual drift is harder to detect future changes make predictions accurate and high average scores do not prove reliable behavior for unknown causes. There is no architecture comparison or real-link validation, in this experiment.

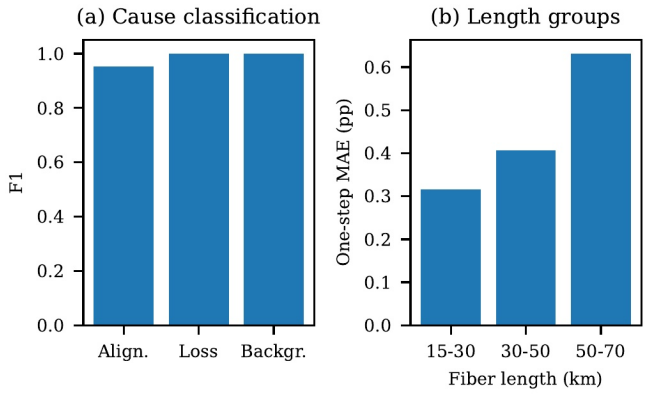


Fig. 4. (a) Per-cause F1 averaged across training seeds. (b) One-step MAE by length group for seed 42. These panels describe different aspects of the same held-out dataset.

VII. Conclusion: Extent to Which the Problem Is Solved

A small PyTorch GRU model does disturbance detection identifies known causes. Predicts QBER using the same 20-interval history. The measurements that were not used during training up this combined approach for the synthetic channels shown. There is a lot of event coverage but the interval recall is still than one; this difference is important when talking about how well detection works. Forecast errors get bigger as the time increases and near event changes. These findings show that this is good for a class project that demonstrates supervised sequence learning not, for a working or secure system.

VIII. Limitations and Future Work

A. Simulation and disturbance coverage

The simulator works with aggregate counts and includes one disturbance episode and a clean calibration period in each affected session. It does not model hardware imperfections, repeated short events, contaminated startup, or changing receiver behavior. The training data include every cause category, so the current evaluation does not test causes the model has not seen. Future work should evaluate measured telemetry and disturbance families held out from training.

B. Statistical and comparative limits

Three initialization seeds share one simulated split. Their SDs do not measure uncertainty across independent datasets. Interval observations overlap in time, so treating each window as an independent replicate would exaggerate precision. Future experiments should vary simulator seeds and channel distributions, compare other neural architectures, and include task ablations. Without those comparisons, the present study cannot attribute performance specifically to multitask learning or establish GRU superiority.

C. Numerical repairs and operational evaluation

Guarded calibration, stable classification losses and bounded forecasts remove the particular numerical failure modes found in the earlier implementation. The calibration fallback and forecast parameterization still require assessment under realistic missingness and extreme conditions. An operational monitor would also need an alarm policy specifying persistence, recovery and intervention costs. None of these monitoring outputs replaces finite-key security analysis [3].

IX. Video Resource

https://youtu.be/81A$_n$$dN1XZU$

References

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, 2014, doi: 10.1016/j.tcs.2014.05.025. Reprint of the 1984 paper.

[2] X. Ma, B. Qi, Y. Zhao, and H.-K. Lo, "Practical decoy state for quantum key distribution," *Physical Review A*, vol. 72, Art. 012326, 2005, doi: 10.1103/PhysRevA.72.012326.

[3] M. Tomamichel, C. C. W. Lim, N. Gisin, and R. Renner, "Tight finite-key analysis for quantum cryptography," *Nature Communications*, vol. 3, Art. 634, 2012, doi: 10.1038/ncomms1631.

[4] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734, doi: 10.3115/v1/D14-1179.

[5] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: NeurIPS 2019 proceedings.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[7] S. Zhang *et al.*, "Machine learning-assisted measurement device-independent quantum key distribution on reference frame calibration," *Entropy*, vol. 23, no. 10, Art. 1242, 2021, doi: 10.3390/e23101242.

[8] G. Maragkopoulos, A. Mandilara, T. Nikas, and D. Syvridis, "Real-time diagnostics on a QKD link via QBER time-series analysis," *Entropy*, vol. 26, no. 11, Art. 922, 2024, doi: 10.3390/e26110922.

[9] H. Luo *et al.*, "Beyond universal attack detection for continuous-variable quantum key distribution via deep learning," *Physical Review A*, vol. 105, Art. 042411, 2022, doi: 10.1103/PhysRevA.105.042411.

[10] J. Liu, B. Huang, J. Su, Q. Peng, and A. Huang, "Deep anomaly detection for active attacks on the receiver in quantum key distribution," *Optics Express*, vol. 33, no. 22, pp. 47137–47150, 2025, doi: 10.1364/OE.575434.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980.