AbS-Q: An Effective Analysis-by-Synthesis Framework for Recovering Signals from Noisy Quantum Outputs

Phat Thanh Tran
Department of Computer Science
Florida State University
Tallahassee, Florida, USA
ptt17@fsu.edu

Ryan Carmichael
Department of Computer Science
Florida State University
Tallahassee, Florida, USA
rcc22e@fsu.edu

Jake Scally
Department of Physics
Florida State University
Tallahassee, Florida, USA
jscally@fsu.edu

Austin Myers
Department of Mathematics
Florida State University
Tallahassee, Florida, USA
atm22b@fsu.edu

Huynjin Yi
Department of Computer Science
Florida State University
Tallahassee, Florida, USA
hy22c@fsu.edu

Xiuwen Liu
Department of Computer Science
Florida State University
Tallahassee, Florida, USA
xliu@fsu.edu

Bayaner Arigong
FAMU-FSU
College of Engineering
Tallahassee, Florida, USA
barigong@eng.famu.fsu.edu

***Abstract*—Quantum computing is advancing rapidly, driven by the increasing capabilities of quantum hardware. However, as devices scale to hundreds of qubits, the presence of high error rates during execution poses a substantial challenge to obtaining reliable and high-fidelity results. Recent approaches have utilized the Hamming spectrum to reconstruct noisy output distributions, improving the proximity of measurement results to the correct solution. In this work, we propose AbS-Q: an effective post-processing model for signal recovery from noisy quantum outputs. AbS-Q integrates structural information from the Hamming spectrum with convolutional deep learning to enhance the ranking and recovery of correct results. In our evaluations, we demonstrate that AbS-Q outperforms existing Hamming-based post-processing techniques in recovering accurate outputs, highlighting its potential as a scalable error mitigation strategy.**

***Index Terms*—AbS-Q Analysis-by-Synthesis Recovering Signals Hamming-based Hamming spectrum**

I. Introduction

Quantum computing is rapidly progressing, with hardware now supporting hundreds of qubits. These systems show promise for accelerating tasks such as optimization, quantum chemistry, and machine learning. However, current devices operate in the Noisy Intermediate-Scale Quantum (NISQ) era and remain prone to noise and hardware errors, limiting their reliability and preventing consistent quantum advantage due to the impracticality of full-scale error correction.

To address the challenges posed by noise in quantum computations, prior research has proposed numerous techniques aimed at reducing gate counts [1]–[3], enabling error-aware computations [4]–[6], and mitigating specific sources of noise such as measurement errors, idle errors, and crosstalk [7], [8].

Quantum error management methods are typically categorized into Quantum Error Correction (QEC) and Quantum Error Mitigation (QEM). While QEC provides real-time error correction, it remains infeasible for near-term devices due to its high resource demands. QEM, in contrast, is a practical post-processing approach that exploits structural patterns such as circuit symmetries, statistical relationships, or machine learning [7], [8]. Techniques like zero-noise extrapolation (ZNE), probabilistic error cancellation (PEC), measurement error mitigation [7], Clifford data regression (CDR), and Hamming spectrum methods [9], [10] are widely used. Hamming-based approaches are particularly attractive for their low overhead but often struggle to consistently rank the correct output at the top of the distribution.

Recent works such as HAMMER [9] and QBEEP [10] have demonstrated the utility of the Hamming spectrum in post-processing error mitigation. These methods exploit the structure of the noisy output distribution by analyzing the Hamming distances between observed bitstrings and known or estimated solutions. Using this information, they aim to reconstruct a more accurate output distribution and enhance the overall fidelity of the results. Specifically, HAMMER [9] uses a reweighting strategy to amplify bitstrings closer in Hamming distance to a reference, while QBEEP [10] applies a binning technique that aggregates output probabilities based on Hamming similarity. While these methods improve probability mass concentration near the correct output, they often fail to consistently elevate its rank within the measured distribution. This limitation motivates the development of low-overhead techniques that more effectively promote correct outputs.

In this work, we introduce AbS-Q (Analysis-by-Synthesis for Quantum computers), a post-processing framework that integrates the structural insights offered by the Hamming spectrum with a convolutional deep learning architecture. By encoding the Hamming spectrum as a spatial prior, the model learns the relative importance of neighboring bitstrings, effectively enhancing the probability of recovering the correct output. In particular, the model is trained only once and can be applied across different datasets without retraining, while still maintaining its ability to improve output fidelity. Comparative evaluations demonstrate that AbS-Q consistently outperforms existing Hamming-based techniques such as HAMMER [9] and QBEEP [10], particularly in improving the ranking of the correct output across varying noise levels, including both high and low error regimes. To assess the need to incorporate the Hamming spectrum into the model's convolutional layers, we also conducted ablation studies using random and He initialization schemes. The results indicate that using the Hamming spectrum leads to a more stable and effective learning process with quality results. We validate our approach using 9-qubits Bernstein-Vazirani (BV) [11] circuits executed on IBMQ hardware. To facilitate reproducibility, the source code is publicly available at https://github.com/ptran1999/AbS-Q.[1]

II. Background

A. NISQ Paradigm

In the current Noisy Intermediate-Scale Quantum (NISQ) era, quantum computers are constrained by limited qubit counts, short coherence times, gate infidelities, and various hardware-level challenges that hinder large-scale, fault-tolerant quantum computation [12], [13]. Gate-based quantum systems are characterized by a range of hardware-specific parameters, such as native gate sets, qubit connectivity (topology), and calibration statistics derived from frequent benchmarking [4]. Despite steady improvements, these systems remain susceptible to noise: average two-qubit gate error rates on state-of-the-art devices (e.g., IBM and Google) still range from 0.001% to 1%, significantly limiting the depth and size of executable quantum circuits.

While QEC provides a route to fault tolerance, its substantial resource overhead—often requiring thousands of physical qubits per logical qubit—renders it impractical for near-term devices [5], [6], [14]. As a result, NISQ systems remain valuable for targeted tasks like variational optimization and quantum simulation [1], [2]. This has led to increased interest in alternative strategies such as improved hardware, noise-aware compilation, and QEM techniques that reduce noise impact without full-scale QEC overhead [7], [8], [10].

B. Quantum Error Structure

Recent research has shown that the structure of noise in quantum output distributions exhibits patterns that can be exploited for error mitigation. The HAMMER [9] framework demonstrates that, for many quantum circuits, errors tend to be locally clustered in the Hamming spectrum; that is, incorrect bit strings often lie close (in Hamming distance) to the correct output. This is supported by empirical evidence showing that the observed expected Hamming distance (EHD) among output states is often lower than the theoretical value of $n/2$ for uniformly random noise, where n is the number of qubits. HAMMER [9] leverages this observation by applying a reweighting function that assigns higher probabilities to bitstrings closer to others in Hamming space, based on an inverse distance function. Moreover, the clustering effect persists even in the presence of entanglement, and as circuit fidelity deteriorates, the output distribution tends toward randomness, with EHD approaching $n/2$.

Building on this insight, QBEEP [10] introduces a more nuanced observation: errors are not always locally clustered. Instead, they can exhibit non-local clustering, where erroneous outputs may form distant clusters in Hamming space rather than simply surrounding the correct result. This phenomenon is empirically supported through extensive experiments on both IBM's superconducting processors and IonQ's trapped-ion devices, using randomized benchmarking and Bernstein-Vazirani [11] circuits. By computing the Index of Dispersion (IoD)—a statistical measure defined as the variance-to-mean ratio of the Hamming distance distribution—QBEEP [10] quantifies the degree of clustering. An IoD close to 1 suggests Poisson-like behavior, which was consistently observed across platforms, confirming the presence of structured error distributions beyond local clustering.

Furthermore, the analysis reveals a positive correlation between circuit complexity and EHD, suggesting that as quantum circuits grow deeper and more complex, the mean Hamming distance of errors increases. Notably, even in more complex circuits, errors do not become uniformly random but instead retain a clustered structure, albeit at larger Hamming distances. Unlike HAMMER [9], which primarily addresses local clustering, QBEEP [10] incorporates a Bayesian state graph model that adapts to both local and non-local error patterns, offering improved flexibility across varying circuit types and hardware platforms.

Given the effectiveness and empirical grounding of both HAMMER [9] and QBEEP [10], we include them as key baselines for comparison in our evaluation. Their insights into the Hamming spectrum serve as the basis for our method, which addresses key limitations—most notably the inconsistent promotion of the correct output—while preserving the low-overhead benefits of spectrum-based postprocessing.

III. Methodology

In this section, we present the design of AbS-Q, a post-processing framework that integrates the structural insights of the Hamming spectrum with the feature-extraction capabilities of convolutional neural networks (CNNs) [15]. The key idea behind AbS-Q is to leverage the tendency of quantum output distributions to exhibit local and non-local clustering in Hamming space [9], [10], and to enhance this by allowing a neural network to learn patterns in these clustered structures. While traditional Hamming-based techniques such as HAMMER and QBEEP rely on fixed distance metrics to reweight outputs [9], [10], AbS-Q utilizes convolutional layers to learn the relative importance of neighboring bitstrings modeled by their Hamming distances directly from data. This approach allows the model to generalize beyond static heuristics and adapt

[1]Code available at https://github.com/ptran1999/AbS-Q

to more complex error behaviors across different circuits and noise regimes. In the following subsections, we describe the architecture of the model, the role of the Hamming spectrum in its input representation, and how the convolutional layers contribute to improving the ranking of the correct output in noisy quantum results. Our evaluation of the Hamming spectrum reveals a consistent relationship between the correct output bitstring and those with lower Hamming distances to it, aligning with prior observations in the HAMMER [9] framework that highlight local clustering of errors in Hamming space. Specifically, we observe that the empirical expected Hamming distance is often lower than the theoretical average of n/2—where n denotes the number of qubits—indicating that errors tend to be localized rather than uniformly distributed. Our experiments are primarily conducted by executing the Bernstein-Vazirani [11] algorithm on IBM's Sherbrooke and Brisbane quantum processors. Notably, we do not observe evidence of non-local error clustering, which is consistent with findings reported in the QBEEP [10] study. That work similarly concluded that such nonlocal phenomena are not well captured by current noise models. Based on these observations, we focus on using the local error clustering behavior as the foundational principle for designing our proposed error mitigation approach.

A. CNNs with Hamming Spectrum

Convolutional neural networks (CNNs) are a class of deep learning models particularly well-suited for extracting local patterns from structured data [15]. Originally developed for image and signal processing, convolutional layers apply learnable filters across an input to detect spatially correlated features. These filters slide over the input, aggregating information from neighboring values within a fixed receptive field, allowing the network to learn both local dependencies and hierarchical representations. In the context of quantum error mitigation, we adopt this principle by applying convolutional layers over the Hamming spectrum of quantum output distributions. Each position in the input corresponds to a bitstring's probability, indexed by its Hamming distance from the expected solution. The convolutional layer is thus able to model how errors tend to cluster across nearby Hamming distances—capturing patterns that reflect the localized structure of noise in real quantum systems. This enables the model to emphasize regions of the spectrum likely to contain or support the correct output, effectively learning a generalized, data-driven analogue to classical Hamming-based correction techniques.

B. Architecture of AbS-Q Model

In this section, we describe the design of our proposed algorithm, which combines structural insights from the Hamming spectrum with the representational power of deep learning. In deep learning, convolutional neural networks (CNNs) are widely used to extract meaningful features by aggregating information from local neighborhoods [15]. We adopt this principle and adapt it to quantum output distributions by leveraging the locality property observed in the Hamming spectrum—where bitstrings with lower Hamming distance to the correct result tend to exhibit higher probabilities [9], [10].

To model this, we construct a Hamming-aware convolutional layer in which each node in the input is connected to its 1 Hamming distance neighbors in the subsequent layer. The output of each node is computed as a weighted sum of its own value and the values of all bitstrings that differ by a single bit, with weights defined by a learnable kernel. Mathematically, for an input vector $x \in \mathbb{R}^N$ and a kernel $k \in \mathbb{R}^{n+1}$ (where n is the number of qubits), the output at index i is given by:

$$y_i = \text{ReLU}\left(x_i \cdot k_0 + \sum_{j=1}^{n} x_{h_j(i)} \cdot k_j \right),$$

where $h_j(i)$ denotes the j-th bitstring differing from index i by exactly one bit. The kernel is initialized such that k_0 corresponds to the average observed probability of the correct result (self-connection), while the remaining k_j terms correspond to the average probabilities of the nearest neighbors. For an n-qubit system, this results in a kernel of size $n+1$, accounting for one self-connection and n Hamming neighbors. After the weighted sum, we apply a ReLU activation function to produce the node's output. At the final layer, we apply the softmax function to transform the outputs into a probability distribution, which is then used to determine the predicted result.

After constructing the convolutional architecture, we assemble the full model by sequentially stacking the individual layers. The input to the model is a tensor comprising normalized probability distributions generated by executing quantum circuits on actual hardware. For instance, in the dataset of bitstrings with Hamming weight 4, we have 126 samples, each represented by a 512-dimensional feature vector that accounts for all possible 9-bit string variations. As a result, the input tensor has a shape of $(126, 512)$. The target labels are also structured as tensors of shape $(126, 512)$, where each row is a one-hot encoded vector representing the correct bitstring. Thus, both the input and output dimensions of the model are 512 in this configuration. Each convolutional layer in the network is structured such that nodes are selectively connected to previous-layer nodes based on their Hamming distance, allowing for localized feature extraction across bitstring indices. At the final layer, we apply the softmax function to convert the raw output values into a valid probability distribution over all bitstrings [16]. This predicted distribution is then compared to the ground-truth one-hot encoded target distribution using the Kullback–Leibler (KL) divergence loss function [17]. The model is trained using backpropagation and optimized with the Adam optimizer [18], employing a learning rate of 0.01 over the course of 50 training epochs. We explore various architectural configurations and find that a network consisting of four convolutional layers offers the best balance between convergence speed, predictive accuracy, and fidelity.

Algorithm 1: Hamming-Convolution Layer

```
Input: Input tensor $\mathbf{x}$, kernel $\mathbf{k} = [k_0, k_1, \dots, k_n]$
Output: Output tensor $\mathbf{y}$
foreach index $i$ in input dimension do
    Compute Hamming-distance-1 neighbors of $i$:
      $\{h_1(i), h_2(i), \dots, h_n(i)\}$;
    $y_i \leftarrow x_i \cdot k_0$;
    for $j \leftarrow 1$ to $n$ do
        if $h_j(i)$ is valid then
            $y_i \leftarrow y_i + x_{h_j(i)} \cdot k_j$;
    $y_i \leftarrow \mathrm{ReLU}(y_i)$;
return $\mathbf{y}$
```

IV. Evaluation

In this section, we present the experimental results of our AbS-Q model, highlighting its promising performance across various configurations. We also detail the methodology used to conduct our experiments. All experiments are based on output distributions obtained by executing the Bernstein–Vazirani (BV) [11] algorithm on IBM's Brisbane and Sherbrooke quantum processors. Specifically, we focus on 9-qubit BV [11] instances, evaluating performance across different target bitstrings that vary in Hamming weight, as higher numbers of ones are known to introduce greater susceptibility to quantum noise.

To assess the effectiveness of AbS-Q, we evaluate the impact of different kernel initialization strategies, including random, He and Hamming spectrum-based initialization, to determine whether our initialization method contributes meaningful prior structure. Furthermore, we investigate the effect of varying the number of convolutional layers in our architecture to identify the optimal model depth. Lastly, we benchmark it against existing error mitigation techniques such as HAMMER [9] and QBEEP [10].

Model performance is assessed using three metrics: accuracy, fidelity, and rank change of the target bitstring. Accuracy refers to the proportion of test cases where the bitstring with the highest model-predicted probability matches the true target string. This is computed by selecting the most probable output from the predicted distribution and checking it against the ground truth for each test instance. Since our model works based on different random train/test splits, we record our results using the average results of the models.

Fidelity is a commonly used metric in quantum computing to quantify the similarity between two quantum states. It provides a measure of how close a given (possibly noisy or approximate) quantum state ρ is to an ideal target state σ. For classical probability distributions, the fidelity can be computed using the following formula:

$$F(\rho, \sigma) = \left(\sum_i \sqrt{\rho_i \sigma_i} \right)^2 \tag{1}$$

Here, ρ_i and σ_i represent the probabilities of outcome i in the respective distributions. The fidelity value ranges from 0 to 1, where 1 indicates identical distributions and 0 indicates completely orthogonal distributions. This metric is particularly useful for evaluating the performance of quantum algorithms and error mitigation techniques.

In addition to accuracy and fidelity, we introduce a third evaluation metric: the number of rank increases and decreases of the correct bitstring. For each test sample, we track the rank position of the correct bitstring in the predicted distribution before and after applying our model. A rank increase is recorded if the correct bitstring improves its position in the sorted list of predicted probabilities (e.g., from rank 4 to rank 2), regardless of whether it reaches the top rank. Conversely, a decrease is recorded if its rank worsens. This metric offers complementary insight into model performance, especially when accuracy remains static. An increase in the rank of the correct bitstring, even if it is not the top prediction, suggests the model is consistently shifting probability mass toward the correct solution. Such rank improvements are useful in practice, where techniques like repeated sampling can further amplify the desired output.

A. Experimental Setup and Performance Evaluation

We conduct our experiments using four distinct datasets. One of these datasets is used for both training and testing, while the remaining three serve exclusively as test sets. The training/testing dataset consists of 149 samples, where each target bitstring represents a decimal value ranging from 0 to 148. As these values are encoded using 9-bit strings, the corresponding Hamming weights in this dataset span from 0 to 7. The number of samples per Hamming weight is distributed as follows: {1, 8, 26, 44, 40, 22, 7, 1}. For each sample, we collect 10,240 measurement shots from the quantum device, resulting in a normalized probability distribution with 512 features (since $2^9 = 512$).

This selective sampling approach allows us to train a model that can generalize across varying Hamming weights rather than specialize in a specific weight class. Due to quantum hardware time constraints, we are not able to include all possible 512 bitstrings in the training data. The dataset is split into training and testing subsets using an 80:20 ratio, and the model is trained and evaluated multiple times to compute the average performance.

The three additional test datasets consist exclusively of bitstrings with Hamming weights of 4, 6, and 7, containing 126, 84, and 36 samples respectively. Each of these samples also comprises 10,240 measurement shots. These Hamming weights are selected to assess model performance across scenarios representing moderate error levels (weight 4), high error levels (weight 6), and maximal observed errors (weight 7).

Figure 1 illustrates the performance of our proposed model. Our training and testing losses both begin near 6, which is expected due to the use of KL divergence over 512 output classes. As training progresses, the losses steadily decline

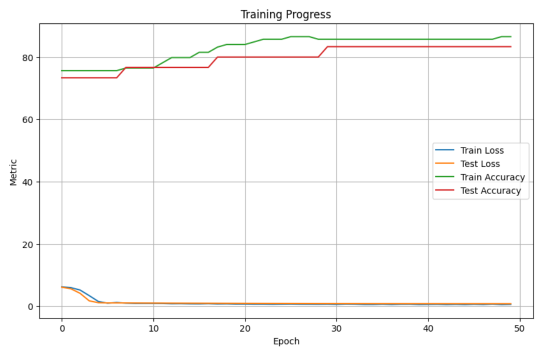

Fig. 1: AbS-Q with 4 layers data performance

and converge to approximately 0.5. The training and testing accuracy typically start around 70% and reach approximately 86% by the end of training. In addition, the model consistently achieves precision, recall, and F1-scores in the range of 0.75 to 0.80. These metrics are commonly used to evaluate classification models, where precision indicates the proportion of correct positive predictions, recall measures the model's ability to identify all relevant instances, and the F1-score provides a harmonic mean of both [19]. Figure 2 shows a typical example where the AbS-Q method achieves a significant improvement. In the original raw measurements, the correct string has a lower count than the first but erroneous state. After applying the AbS-Q method, the correct state has the highest probability and much higher than the second highest.

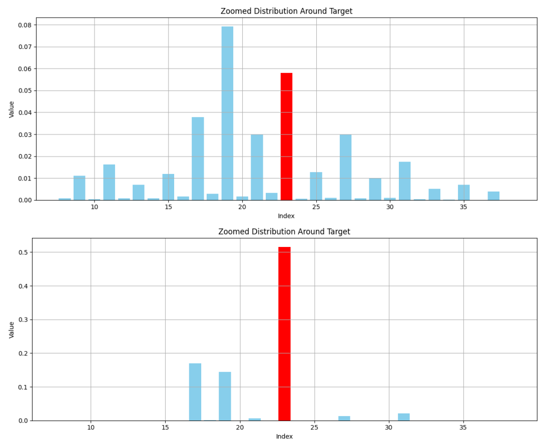

Fig. 2: Top: Distribution before going through AbS-Q; bottom: distribution after going through AbS-Q. Here the red line represent the target bit string

These results suggest that the model is effectively learning and improving over time. We anticipate that further improvements in performance could be achieved by training on a more diverse dataset that includes a broader range of Hamming weight variations.

B. Effects of Different Filter Initialization

In this experiment, we investigate the impact of Hamming spectrum-based initialization on the performance of our convolutional model. Specifically, we initialize the kernel weights using the observed Hamming distribution of the quantum output and compare this approach to two commonly used initialization methods in deep learning: random initialization and He initialization, the latter being particularly designed for ReLU-activated networks. Table I shows the results on the four datasets using the three initialization methods. Clearly, the Hamming spectrum method gives the best results. For Hamming weight 4 dataset, it improves the rank of 65 samples where only 7 have lower ranks.

Our findings indicate that initializing the kernel using the Hamming spectrum provides notable advantages. This approach not only accelerates convergence but also results in smoother and more stable training dynamics. In contrast, both random and He initialization methods exhibit inconsistent performance, particularly in terms of accuracy. In several cases, these methods even led to a decrease in accuracy compared to the original unprocessed quantum output, suggesting poor generalization.

Nevertheless, both random and He initialization methods did show improvements in fidelity across all datasets and, in some instances, outperformed traditional quantum error mitigation techniques such as HAMMER [9] and QBEEP [10]. Furthermore, the rank analysis revealed a favorable trend, with more rank increases than decreases, indicating some effectiveness in amplifying correct bitstrings.

However, in comparison, our Hamming spectrum initialized model consistently outperformed the alternatives across all metrics. These results emphasize the critical role of incorporating domain-specific statistical structures—such as the Hamming spectrum—in developing effective quantum error mitigation strategies.

C. Effects of Number of Layers

Here we evaluate the impact of varying the number of convolutional layers in our model to determine the optimal configuration that balances performance with computational efficiency. We conduct experiments with models containing between one and seven layers. The results are shown in Tab. II. Our findings indicate that performance consistently improves from one to four layers, with the most significant gains observed when increasing from one to two layers. Beyond four layers, however, the improvements become marginal, and diminishing returns are observed. Notably, models with seven layers occasionally fail to generalize well on certain datasets, likely due to over smoothing. Moreover, deeper models demand greater computational resources and longer training times. Based on these observations, we identify the four-layer architecture as the most effective trade-off between accuracy and efficiency, and adopt it as the default configuration for subsequent experiments.

D. Comparison with HAMMER and QBEEP

To demonstrate the effectiveness of the proposed method, we have compared our method to HAMMER [9] and QBEEP [10] on the datasets. Table III summarizes our comparative

TABLE I: Comparison of model performance using different initialization methods and spectrum-based corrections.

	Original Error Data		**Random**			**He**			**Hamming Spectrum**		
	Accuracy	Fidelity	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank inc/dec
Train/Test Data	0.7517	0.5968	0.5234	0.8542	35/2	0.7517	0.7640	8/13	0.8590	0.8760	35/2
Hamming weight 4	0.4206	0.3300	0.3190	0.5812	59/22	0.2300	0.6104	57/17	0.6507	0.6886	65/7
Hamming weight 6	0.0952	0.1162	0.0666	0.1477	41/40	0.0238	0.1484	41/38	0.1357	0.1925	48/30
Hamming weight 7	0.0000	0.0593	0.0222	0.0755	23/12	0.0166	0.0739	23/11	0.0444	0.0785	19/16

TABLE II: Comparison of model performance with 2, 4, and 6 layers.

	Original Error Data		**2 Layers**			**4 Layers**			**6 Layers**		
	Accuracy	Fidelity	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank
Train/Test Data	0.7517	0.5968	0.8268	0.8198	30/5	0.8590	0.8760	35/2	0.1798	0.8602	35/2
Hamming weight 4	0.4206	0.3300	0.5460	0.5955	44/20	0.6507	0.6886	65/7	0.0936	0.5361	53/26
Hamming weight 6	0.0952	0.1162	0.0880	0.1582	29/42	0.1357	0.1925	48/30	0.0166	0.1280	48/31
Hamming weight 7	0.0000	0.0593	0.0277	0.0562	24/12	0.0444	0.0785	19/16	0.0111	0.0713	18/16

TABLE III: Comparison of accuracy, fidelity, and rank improvements across different methods.

	Original Error Data		**HAMMER [9]**			**QBEEP [10]**			**ABS**		
	Accuracy	Fidelity	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank inc/dec	Accuracy	Fidelity	Rank inc/dec
Train/Test Data	0.7517	0.5968	0.7517	0.7802	2/5	0.7517	0.7640	8/13	0.8590	0.8760	35/2
Hamming weight 4	0.4206	0.3300	0.4206	0.5100	2/0	0.4286	0.4900	9/24	0.6507	0.6886	65/7
Hamming weight 6	0.0952	0.1162	0.0952	0.1399	26/15	0.0833	0.1185	24/48	0.1357	0.1925	48/30
Hamming weight 7	0.0000	0.0593	0.0000	0.0450	18/5	0.0278	0.0466	13/22	0.0444	0.0785	19/16

results. Our comparative evaluations show that neither HAMMER nor QBEEP method significantly improves the accuracy of the original quantum output distributions. However, both techniques yield noticeable gains in fidelity, indicating a closer alignment with the ideal quantum state. A deeper analysis using rank increase/decrease statistics reveals a critical limitation of QBEEP [10]: across all datasets, QBEEP [10] consistently resulted in a higher number of rank decreases than increases for the correct bitstring. This suggests a potential confidence regression for correct predictions.

In contrast, our proposed AbS-Q model outperformed both HAMMER [9] and QBEEP [10] in all three evaluation metrics: accuracy, fidelity, and rank dynamics. AbS-Q not only achieved consistent gains in accuracy and fidelity in all test scenarios, but also demonstrated favorable rank progression, even under noisy or error-prone input conditions. Specifically, the number of rank increases for the correct bitstring consistently exceeded the number of decreases, highlighting the robustness and reliability of the model.

V. Related work

Reducing quantum errors is essential for advancing quantum computing, and two main strategies have emerged: quantum error correction (QEC) and quantum error mitigation (QEM). QEC typically relies on surface codes [12]–[14], where logical qubits are constructed from multiple physical qubits and protected via stabilizer measurements. Several works [4]–[6] have addressed the practical challenges of implementing QEC on near-term quantum hardware, including ancilla placement and syndrome decoding in cryogenic conditions.

On the other hand, QEM focuses on reducing error impact without requiring full error correction overhead. This includes pre-circuit techniques like transpiler optimization [1]–[3] and domain-specific cancellation [2]. Post-circuit techniques include Bayesian inference [7], circuit inversion [8], and spectrum-based reclassification such as HAMMER [9] and QBEEP [10]. HAMMER models Hamming-local error clusters for reweighting bitstring probabilities, while QBEEP utilizes qubit-specific error priors to boost fidelity across multiple devices.

These methods highlight the growing importance of noise-aware processing in the NISQ era, offering complementary solutions to QEC that are better suited for today's hardware.

VI. Conclusion and Future Work

In this paper, we have presented AbS-Q, a convolutional neural network framework that leverages Hamming-distance-based kernels to mitigate quantum errors in noisy output distributions. Our approach was evaluated on datasets generated from the Bernstein–Vazirani algorithm executed on IBM quantum hardware, targeting 9-bit strings [11]. Compared to existing methods such as HAMMER [9] and QBEEP [10], AbS-Q demonstrates consistent improvements in accuracy, fidelity, and rank positioning of the correct bitstring. We further show that initializing the kernel using the Hamming spectrum not only enhances performance but also accelerates convergence compared to random and He initialization. An ablation study on the number of convolutional layers reveals that a four-layer configuration offers the best trade-off between accuracy and computational cost. These findings suggest that AbS-Q is a robust and efficient method for learning error patterns in quantum circuits and can serve as a foundation for future advancements in quantum error mitigation using deep learning techniques [20]. Beyond empirical improvements, we would like to understand the theoretical properties of the proposed method and provide theoretical justifications of the empirical findings; this is being investigated.

References

[1] L. Gushu *et al.*, "Gflow: Efficient quantum circuit mapping through graph-based search and scheduling," in *ASPLOS*, 2022.

[2] Z. Peng *et al.*, “2qan: Efficient quantum circuit optimization for 2-local hamiltonians,” in *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023.

[3] T. Itoko *et al.*, “Qsearch: Greedy search for optimal qubit mapping,” *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–12, 2022.

[4] X. Wu *et al.*, “Stitching surface codes to near-term quantum devices,” in *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*, 2022.

[5] A. Sharma *et al.*, “Liliput: Lightweight lookup table-based syndrome decoder for quantum error correction,” in *Proceedings of the 54th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2021.

[6] C. Li *et al.*, “Qulatis: Quantum lookup table for adaptive and timely in-situ syndrome decoding,” *arXiv preprint arXiv:2104.14793*, 2021.

[7] S. Zheng *et al.*, “Bayesian inference-based quantum error mitigation,” *arXiv preprint arXiv:2202.12194*, 2022.

[8] D. Patel *et al.*, “Reversal-based error mitigation for quantum circuits,” *arXiv preprint arXiv:2203.10049*, 2022.

[9] S. Tannu, P. Das, R. Ayanzadeh, and M. Qureshi, “Hammer: boosting fidelity of noisy quantum circuits by exploiting hamming behavior of erroneous outcomes,” in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 529–540. [Online]. Available: https://doi.org/10.1145/3503222.3507703

[10] S. Stein, N. Wiebe, Y. Ding, J. Ang, and A. Li, “Q-beep: Quantum bayesian error mitigation employing poisson modeling over the hamming spectrum,” in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA ’23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589043

[11] E. Bernstein and U. Vazirani, “Quantum complexity theory,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[12] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, “Surface codes: Towards practical large-scale quantum computation,” *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[13] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, “Topological quantum memory,” *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4452–4505, 2002.

[14] H. Bombin and M. A. Martin-Delgado, “Topological quantum distillation,” *Physical Review Letters*, vol. 97, no. 18, p. 180501, 2006.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” *MIT Press*, 2016.

[17] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[18] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in python,” pp. 2825–2830, 2011. [Online]. Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.