

Dual-Head Learning-Based Candidate-Gate Pruning for Efficient Quantum Unitary Synthesis

S. Kalesha Vali¹, K. Venkata Ramana²

^{1,2}Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam-530003, A.P, India

ABSTRACT

Quantum unitary synthesis (QUS) commonly employs a scalar cost-to-go predictor to guide stochastic beam search. At each search step, however, every beam element is expanded over the complete candidate-gate set before the resulting states are evaluated, yielding a $(|G|)$ -way branching factor and making candidate evaluation a major computational bottleneck. This work augments the shared network trunk with a classification head that predicts the most likely next gates from a given residual state. The predicted gate probabilities are used to prune the candidate set prior to computationally expensive per-child cost-to-go evaluation. The proposed approach includes a data-generation and pre-processing pipeline, a multilabel classification formulation, and a safety mechanism that monitors the underlying cost-to-go model and automatically restores the most recent reliable model upon performance degradation. Experiments using a trained four-qubit ($n = 4$) model demonstrate approximately 87% next-gate prediction accuracy, with improved accuracy as the prediction-set size increases. Incorporating classification-based candidate pruning reduces search time by approximately 5times. Although aggressive pruning currently results in a reduction in overall synthesis success rate under the available training data, the observed classification performance indicates that increased training data and improved prediction accuracy can enable more effective pruning and further reduce synthesis cost. All reported results and figures are obtained from actual training and evaluation runs of the accompanying implementation.

Key words- Quantum unitary synthesis; Clifford+T; Minimum description length; Stochastic beam search; Multi-label classification; Quantum circuit synthesis; Learned heuristics; Pruning.

INTRODUCTION

Quantum computing provides a computational paradigm in which complex operations are represented and executed through quantum circuits. A fundamental problem in this domain is quantum unitary synthesis (QUS), which determines a sequence of elementary quantum gates that implements a given target unitary operation. The synthesis problem is computationally challenging because the number of possible gate sequences grows exponentially with circuit depth and the size of the available gate set. Consequently, practical QUS methods must balance synthesis quality, circuit complexity, and computational cost.

Several approaches have been developed to address this problem. Early circuit-synthesis methods focused on constructive and exact techniques. Shende et al. [1] investigated systematic synthesis of quantum logic circuits, establishing important decomposition procedures for converting unitary transformations into elementary gate sequences. For Clifford+T circuits, Kliuchnikov et al. [2] developed efficient exact synthesis methods for single-qubit unitaries, while Giles and Selinger [3] extended exact synthesis to multiqubit Clifford+T circuits. Gheorghiu et al. [4] further studied the fundamental cost of exact synthesis by characterizing the T-count and T-depth of arbitrary multiqubit unitaries. Although these methods provide strong guarantees, their computational requirements and reliance on exact constructions can limit their applicability to increasingly large and complex synthesis problems.

Heuristic and optimization-based methods provide an alternative by searching for high-quality circuits without requiring exhaustive exact constructions. Mosca and Mukhopadhyay [5] proposed a polynomial-time heuristic for reducing T-count, demonstrating that approximate optimization can substantially reduce circuit cost. Paradis et al. [6] introduced Synthetiq, which employs simulated annealing for fast and versatile circuit synthesis. Nagarajan et al. [7] formulated quantum circuit optimization as a mixed-integer linear programming problem through QuantumCircuitOpt, enabling provably optimal solutions for suitable problem instances. These approaches demonstrate the benefits of optimization-driven synthesis; however, iterative or exhaustive evaluation of candidate operations can become computationally expensive as the search space increases.

More recently, machine learning has been investigated as a means of learning useful structure within the quantum-circuit search space. Furrutter et al. [8] introduced diffusion models for quantum circuit synthesis, showing that generative models can learn distributions of useful circuit structures. Chen and Tang [9] proposed UDiTQC, a U-Net-style diffusion transformer for quantum circuit synthesis, further demonstrating the potential of learned generative models for constructing quantum circuits. Reinforcement learning has also been applied directly to unitary synthesis. Rietsch et al. [10] investigated reinforcement-learning-based synthesis of Clifford+T circuits, demonstrating that learned policies can guide the search toward useful gate sequences. These approaches establish that learned models can provide valuable guidance; however, the computational cost of evaluating a large number of candidate actions remains an important challenge during search.

The work in [11] addresses part of this challenge through a learning-based QUS framework that uses a neural network to predict the minimum description length (MDL), or cost-to-go, of candidate Clifford+T circuits. The predicted cost is incorporated into a stochastic beam-search procedure to guide synthesis toward promising circuit states. By replacing repeated exhaustive optimization with a learned cost estimator, the framework improves the scalability of guided synthesis. Nevertheless, its search procedure still evaluates every candidate gate for each beam element before the resulting states can be scored. If the beam width is (B) and the gate vocabulary contains $(|G|)$ possible actions, approximately $(B|G|)$ candidate expansions are generated at each search step. Thus, although the learned regression model reduces the cost of evaluating candidate quality, the full branching of the action space remains a major source of computational overhead.

This observation motivates the development of a mechanism that can identify promising gates before expensive candidate-state evaluation is performed. In this work, the framework in [11] is extended by introducing a second classification head on the same shared neural-network trunk. Instead of evaluating all possible gates, the classification head predicts the gates that are most likely to provide useful progress from the current residual unitary. The predicted candidates are then used to prune the search space before candidate-state evaluation and cost-to-go prediction are performed. If only (k) candidate gates are retained, where $(k \ll |G|)$, the effective number of candidate expansions is reduced from approximately $(B|G|)$ to (Bk) per search step. This reduces the effective branching factor while preserving the learned cost-guidance mechanism of the original framework.

The proposed classification task is formulated as a multi-label prediction problem rather than a conventional single-label classification problem. Multiple gates may represent valid or similarly promising next actions because of equivalent synthesis paths, commuting operations, and alternative circuit constructions. A multi-label formulation therefore enables the model to retain several plausible candidates rather than restricting the search to a single predicted gate. The classifier uses the same residual-state representation as the original cost-to-go predictor, allowing both tasks to benefit from a shared learned representation. A multi-task training strategy is employed to introduce the classification capability while preserving the effectiveness of the pretrained regression model. The resulting dual-head architecture jointly provides cost estimation for candidate evaluation and gate-selection information for search-space pruning.

Experimental evaluation demonstrates that the proposed classification head can identify promising next-gate candidates with high top- (k) recall. For the evaluated four-qubit synthesis setting with a 24-action gate vocabulary, the classifier achieves approximately 87% top-8 recall, with recall exceeding 95% for larger candidate sets. When integrated into stochastic beam search, the proposed pruning mechanism produces approximately a five-fold reduction in wall-clock search time in the evaluated configuration. The results also reveal a tradeoff between pruning aggressiveness and synthesis success rate, indicating that classification accuracy and candidate-set size are important factors in determining the achievable reduction in search cost.

The main contributions of this work are the introduction of a classification-based search-pruning mechanism as an extension of the learned QUS framework in [11]; a shared multi-task neural architecture that jointly learns cost-to-go regression and next-gate classification from a common residual-state representation; a multi-label gate-prediction strategy that retains multiple plausible synthesis actions rather than restricting the search to a single gate; and the integration of the predicted candidate set into stochastic beam search to reduce expensive candidate evaluations from approximately $(B|G|)$ to (Bk) . Experimental results demonstrate high top- (k) gate-selection recall and substantial reductions in search time, while also characterizing the tradeoff between pruning aggressiveness and synthesis success.

PROBLEM FORMULATION

Let n be the qubit count and G the fixed-order action vocabulary $[H_0, \dots, H_{n-1}, S_0, \dots, S_{n-1}, T_0, \dots, T_{n-1}, CX_{0,1}, \dots, CX_{n-1,n-2}]$. The action-set size is $|G| = 3n + n(n-1) = n(n+2)$.

For $n = 4, |G| = 24$. The three single-qubit gate types contribute $3n$ actions, while the ordered control-target CNOT pairs contribute $n(n - 1)$ actions.

Given a residual unitary $R \in \mathcal{C}^{2^n \times 2^n}$, the classification head predicts a vector $\hat{p}(R) \in [0,1]^{|G|}$, with one independent probability per action. The formulation is multi-label because several gates can be valid next actions owing to commuting operations and alternative synthesis paths.

The base stochastic beam search expands each of B beam elements using all $|G|$ actions, giving approximately $B|G|$ candidate expansions per search step. The proposed classifier retains only the k highest-scoring actions, where $k \ll |G|$, reducing the candidate set to approximately Bk before the more expensive residual construction and cost-to-go evaluation.

DATA GENERATION AND PREPROCESSING

The training data for the classification head are generated from randomly sampled Clifford+T circuits and processed through circuit optimization, canonicalization, and residual-state construction to obtain consistent training examples. The resulting data are converted into multi-label gate-prediction targets and phase-normalized real-valued feature vectors, as illustrated in Fig. 1.

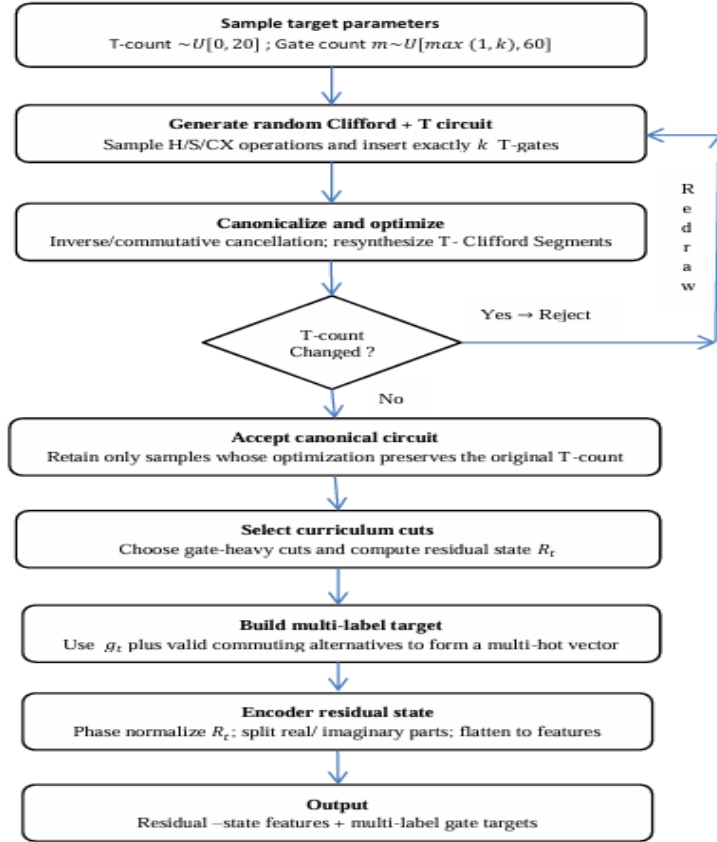


Fig. 1. Data-generation and pre-processing flow used to construct training examples for the classification head.

A. Circuit Sampling and Canonicalization

Training examples originate from randomly sampled Clifford+T circuits generated with the same underlying procedure used for the regression head. A target T -count k is sampled uniformly from $[0, 20]$, and a gate count m is sampled uniformly from $[\max(1, k), 60]$. A circuit of m gates is generated using H, S , and CX operations on randomly selected qubits or ordered qubit pairs, with exactly k T gates inserted at random positions.

Peephole optimization first performs inexpensive inverse and commutative cancellations. The circuit is then divided into maximal T -free Clifford segments separated by T gates; each segment is resynthesized using Qiskit's Clifford-tableau synthesis, and the shorter of the original and resynthesized segments is retained. Samples for which the optimization changes T -count are rejected and redrawn.

B. Curriculum Cuts and Multi-Label Targets

An accepted canonicalized circuit is represented as $C = (g_1, g_2, \dots, g_M)$, where g_1 is applied first and g_M last. Cutting after gate t gives a residual state R_t . The regression target is the number of gates already represented by the prefix, $y_t^{reg} = t$.

For classification, the correct search label is gt , the last gate already present in the prefix, because the synthesis search proceeds backward from the target and undoes the most recently applied gate first. Using g_{t+1} would instead describe forward construction and would not match the search transition.

More than one gate may be a valid label. The implementation scans backward from gt and adds earlier gates whose qubits remain independent of the intervening operations, provided the commutation is valid. The result is a multi-hot target vector over the fixed action vocabulary.

To support smaller targets with an n -qubit model, 15% of training circuits are sampled at $n' < n$ and padded with identity operations. Actions touching padding qubits are masked at inference time. Because the independent-sigmoid formulation does not require probability renormalization after masking, padded actions can simply be excluded.

C. Residual-State Representation

A residual R is physically meaningful only up to a global phase. Therefore, the matrix is phase-normalized by locating the first entry with magnitude greater than 10^{-6} in row-major order and dividing the entire matrix by that entry's phase. The normalized matrix is split into real and imaginary parts and flattened into a real feature vector of length $2(2^n)^2$; this gives 512 features for $n = 4$ and 2048 for $n = 5$.

The classification target is a multi-hot vector in $\{0,1\}^{|G|}$. At the trivial cut, $t = 0$, no next action is defined, so that example contributes only to the regression loss.

PROPOSED DUAL-HEAD ARCHITECTURE

The overall organization of the proposed dual-head network, including the shared residual-state representation, common neural-network trunk, and task-specific regression and classification heads, is illustrated in Fig. 2.

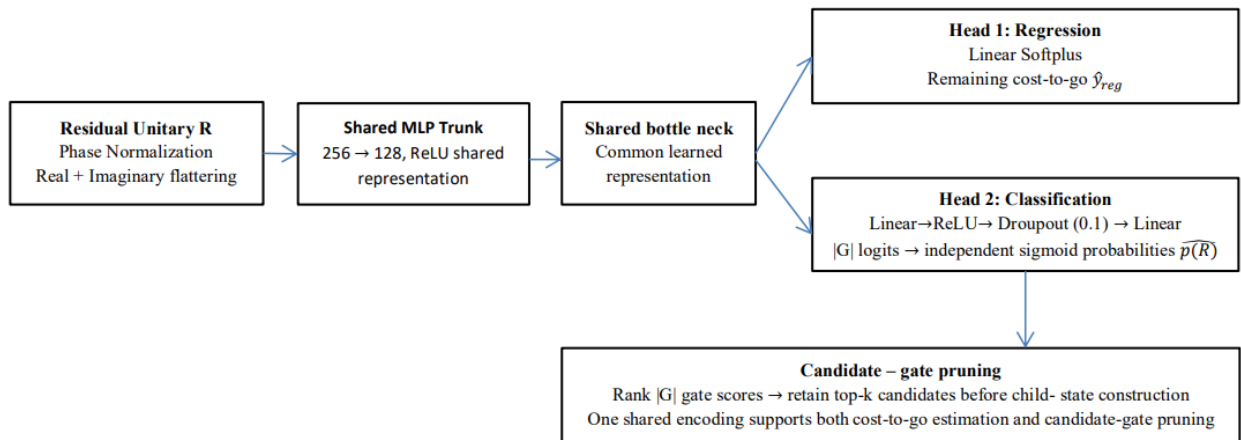


Fig. 2. Proposed dual-head Neural architecture. The regression and classification heads share the same residual-state representation and trunk.

A. Shared Representation and Heads

The classification head branches from the bottleneck representation already used by the cost-to-go regression head. In the reported $n = 4$ implementation, the shared MLP trunk uses hidden dimensions (256,128). The regression head is Linear→Softplus with one nonnegative output, while the classification head is Linear→ReLU→Dropout(0.1)→Linear with $|G|$ logits followed by independent sigmoid activations.

Independent sigmoids are preferred to softmax because multiple actions may be simultaneously correct and because padded-qubit actions can be masked without renormalizing the remaining probability mass.

B. Training and Safety Mechanism

Training starts from an already-trained regression-only checkpoint. First, the trunk is frozen while the new classification head is trained. Next, joint fine-tuning unfreezes the trunk using a learning rate 20 times smaller than that of the classification head. The combined objective is $L = \frac{L_{reg}}{S_{reg}} + \frac{L_{cls}}{S_{cls}}$ where S_{reg} and S_{cls} are exponential moving averages of the corresponding loss magnitudes.

A regression safeguard monitors validation MSE against the pretrained baseline. If the validation MSE degrades by more than 2%, the model is rolled back to the most recent reliable checkpoint and the trunk learning rate is halved. This prevents the added classification objective from substantially degrading the original cost-to-go predictor.

END-TO-END SEARCH AND PRUNING

The complete integration of the dual-head network into the stochastic beam-search procedure, from residual-state encoding and candidate-gate prediction to candidate pruning, cost-to-go evaluation, and beam selection, is shown in Fig. 3.

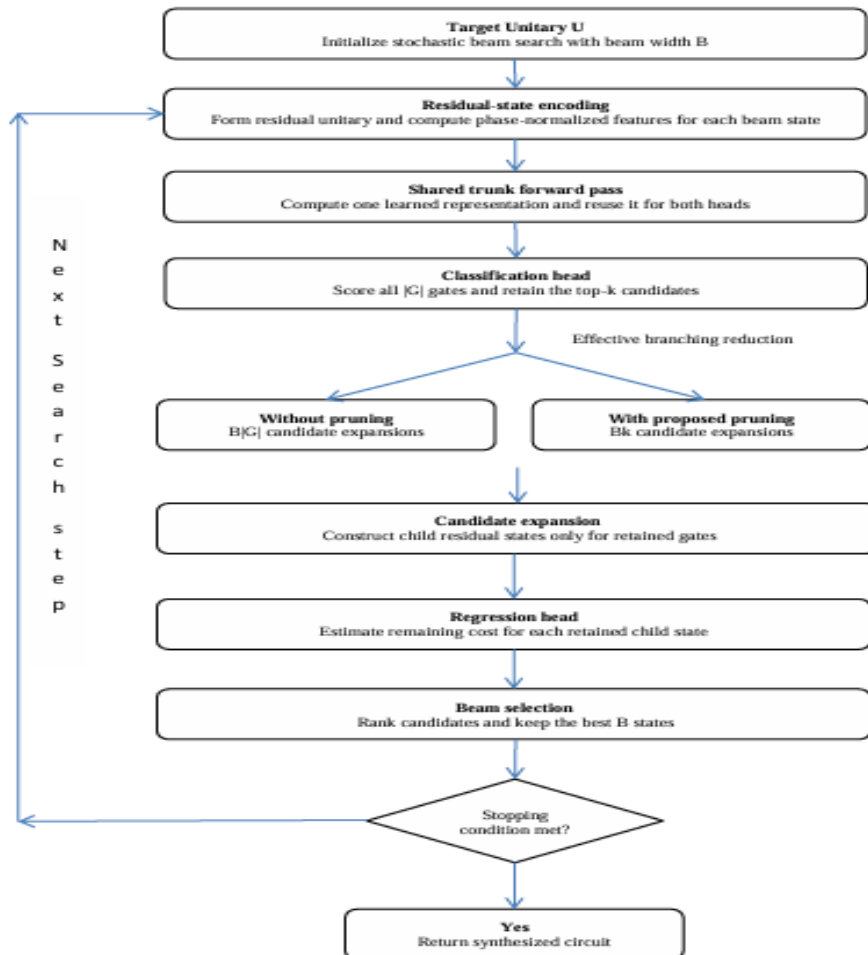


Fig. 3. End-to-end flow of dual-head QUS with classification-based candidate pruning.

At each search step, the residual state is encoded once and passed through the shared trunk. The classification head scores the action vocabulary and selects the top-k candidates. Only those candidates are expanded into child residual states and evaluated by the regression head. The stochastic beam search then retains the best B states. Thus, the dominant candidate-scoring workload changes from approximately $B|G|$ to Bk per step, while the classifier adds only one inexpensive head to the shared representation.

EXPERIMENTAL SETUP

The experimental setup describes the software environment, model configuration, training procedure, and evaluation protocol used to implement and assess the proposed dual-head QUS framework.

A. Software and Implementation Environment

The experiments were implemented in Python. The quantum-circuit canonicalization and Clifford-tableau resynthesis stages use Qiskit, including the Qiskit Clifford-tableau synthesis routines *synth_clifford_greedy* and *synth_clifford_ag* described in the implementation report. The reported experimental results were generated from the actual training and evaluation workflow rather than from a purely conceptual simulation. The implementation report does not specify an exact Python or Qiskit version; therefore, no version number is asserted here.

For reproducibility, the implementation report states that the reported numbers and figures are generated by *paper/generate_results.py* from checkpoints produced by *qus.train baseline* and *qus.train joint*, with the corresponding commands documented in the project README.

B. Model Configuration and Training Procedure

The reported experiments use $n = 4$ qubits and an action vocabulary of $|G| = 24$. The shared representation is an MLP with hidden dimensions (256,128), followed by two task-specific heads. The regression head consists of a linear layer followed by Softplus and predicts the remaining-gate cost-to-go. The classification head uses Linear $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1) $\rightarrow$ Linear and produces one logit per action, followed by independent sigmoid activations. Because multiple gates can be valid at a residual state, the classification objective is multi-label binary cross-entropy rather than softmax classification.

Training follows the staged procedure described in the implementation report. Stage 0 trains the regression-only baseline for 20 epochs with 150 steps per epoch. For the dual-head model, the classification head is first trained with the shared trunk frozen for 8 epochs, followed by 25 joint fine-tuning epochs with 150 steps per epoch. The classification-head hidden width is 96. A replay buffer of 2000 examples is used, with freshly sampled circuits covering T-counts in $[0,10]$ and gate counts in $[1,25]$. Fifteen percent of training examples use a lower-qubit padded slice, with actions touching padding qubits masked during inference. A regression-safety mechanism is retained during joint fine-tuning. Validation regression MSE is monitored against the pretrained regression baseline; if the validation MSE degrades by more than 2%, the model is rolled back to the most recent reliable checkpoint and the trunk learning rate is halved. This prevents the addition of the classification objective from substantially damaging the original cost-to-go predictor.

C. Evaluation Metrics and Experimental Protocol

The classification head is evaluated using top- k recall because the target is multi-label: an example is counted as correctly covered when at least one valid next-gate label occurs among the k highest-scoring actions. Classification performance is measured on 3000 freshly sampled held-out examples. The reported top- k values are evaluated for $k \in \{1,2,3,5,8,12,18,24\}$, with $k = 24$ corresponding to full expansion.

The evaluation also measures the effect of pruning on the downstream search process. The effective branching factor is estimated from the number of regression forward passes per beam element per search step, and mean wall-clock synthesis time per instance is recorded for different pruning widths. End-to-end synthesis success rate is compared with the full-expansion baseline to expose the tradeoff between computational savings and search reliability. The reported search experiments use beam width 10. The principal architectural and training-related components of the dual-head network are summarized in Table I, providing a concise reference for the configuration used in the reported four-qubit experiments.

Table I. Network component configuration used in the reported $n=4$ experiments.

Component	Configuration
Shared trunk	MLP, hidden dims (256,128) , ReLU (evaluated $n = 4$ configuration)
Regression head	Linear $\rightarrow$ Softplus (1 output; remaining gate count)
Classification head	Linear $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1) $\rightarrow$ Linear ($ G $ logits)
Output activation	Independent sigmoids (multi-label), not softmax
Classification loss	Binary cross-entropy per action, averaged over examples with a label

RESULTS AND DISCUSSION

A. Learning Curve and Safety Check

The original learning curve shows top-8 recall increasing from 87.0% at the first frozen-trunk epoch to approximately 88–90% after the trunk is unfrozen. One rollback occurred at joint epoch 13 (global epoch 21): validation MSE increased from the pretrained baseline of 7.535 to 7.761 (+3.0%), exceeding the 2% tolerance. The safeguard restored the last good checkpoint, after which MSE recovered to 6.955 on the next epoch.

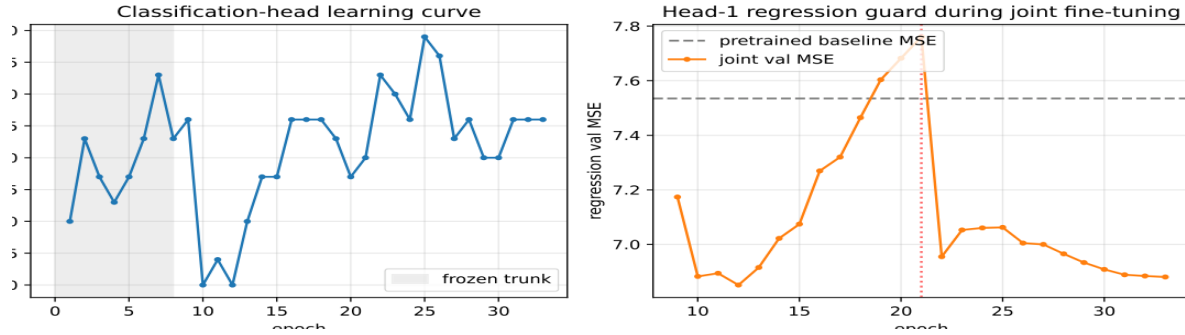


Fig. 4. Left: top-8 recall vs. epoch (shaded region: trunk frozen). Right: regression validation MSE during joint fine-tuning, against the pretrained baseline (dashed) and the automatic rollback event (dotted vertical line) at epoch 21.

B. Classification Accuracy

Because the task is multi-label, top- k recall is the appropriate metric: an example is counted as successful if at least one valid label appears among the k highest-scoring actions. On 3000 freshly sampled held-out examples, the measured recalls are 0.306, 0.479, 0.603, 0.721, 0.865, 0.952, 0.972, and 1.000 for $k = 1, 2, 3, 5, 8, 12, 18, 24$, respectively.

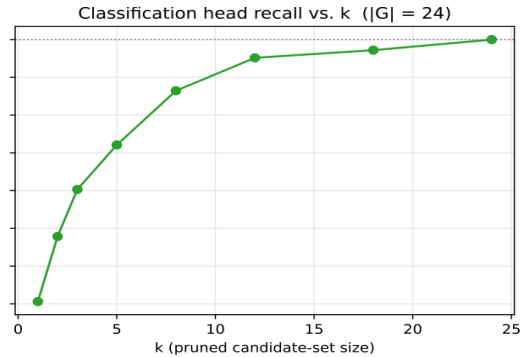


Fig. 5. Top- k recall versus k . The original Stage-3 acceptance gate requires top-8 recall ≥ 0.90 ; this checkpoint reaches 0.865.

The corresponding top- k recall values for the different candidate-set sizes are summarized in Table II, highlighting the improvement in gate coverage as the number of retained candidates' increases.

Table II. Classification-head top- k recall versus k on 3000 held-out examples.

k	1	2	3	5	8	12	18	24 ($= G $)
Top- k recall	0.306	0.479	0.603	0.721	0.865	0.952	0.972	1.000

C. Effect on Search Cost and Success Rate

The purpose of pruning is to reduce the number of regression forward passes per beam element per step. The original measurements show that the effective branching factor tracks the configured k closely: approximately 1.47 passes at $k = 1$ and 22.6 at $k = 24$. Wall-clock time falls from 0.55s per instance at $k = 24$ to 0.11s at $k = 1$, corresponding to an approximately five-fold reduction in the evaluated configuration.

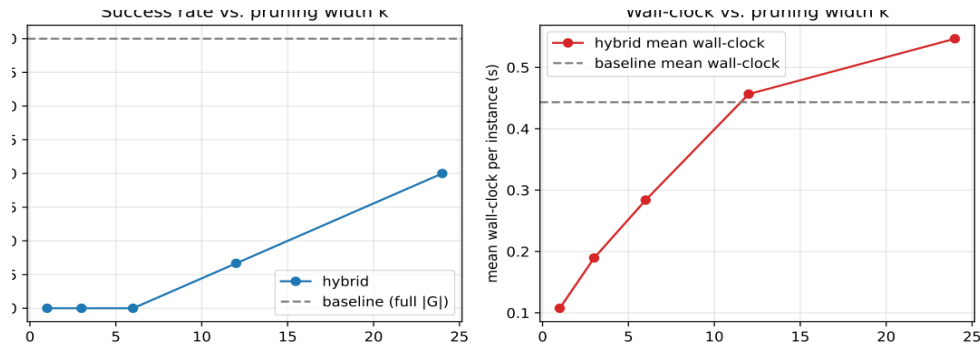


Fig. 6. Left: end-to-end synthesis success rate versus pruning width k , against the full-expansion baseline (dashed). Right: mean wall-clock time per instance versus k .

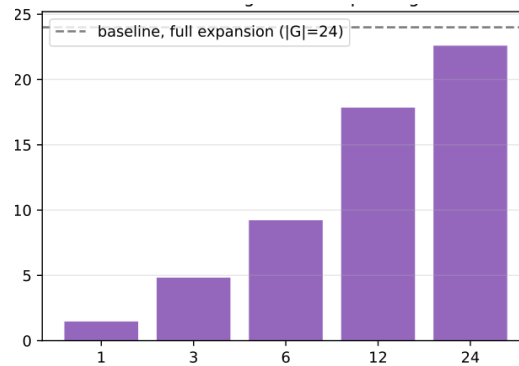


Fig. 7. Effective branching factor (regression forward passes per beam element per step) versus configured k .

The success-rate trade-off is important. At the current checkpoint, top-8 recall is 86.5%, so a missed valid action can derail a synthesis trajectory. The original evaluation reports 0% success for $k \in \{1,3,6\}$, 3.3% at $k = 12$, and 10% at $k = 24$ with the confidence-based widening rule, compared with 20% for the full-expansion baseline on the small 6-T-count $\times$ 10-trial evaluation. These results indicate that the pruning mechanism itself behaves as intended, but the classifier requires more training or a wider/adaptive candidate set before aggressive pruning can be relied upon for synthesis success.

CONCLUSION

This work extends learned quantum unitary synthesis with a second classification head that predicts plausible next gates before the expensive per-child cost-to-go evaluation. The resulting dual-head architecture reuses the shared residual-state representation, formulates gate selection as a multi-label problem, and reduces the effective branching factor from approximately $B|G|$ to Bk . The data-generation, canonicalization, multi-label target construction, phase-normalized feature extraction, and safety-monitoring procedures provide a complete pipeline for training and deploying the pruning head.

The evaluated $n = 4$ model confirms the central computational benefit: reducing the candidate set from 24 gates to a small top- k set directly reduces regression evaluations and lowers wall-clock search time by approximately five times at $k = 1$. At the same time, the current checkpoint shows that runtime reduction and synthesis success are not interchangeable objectives. Top-8 recall of 86.5% is below the 90% acceptance threshold used by the evaluation procedure, and aggressive pruning can therefore reduce synthesis success. The appropriate next step is larger-scale training at the intended $n = 5$ setting, together with confidence-adaptive widening and repeated measurement of top- k recall, search time, effective branching factor, and end-to-end success rate. Thus, the proposed architecture should be viewed as a practical search-cost reduction mechanism whose full benefit depends on sufficiently accurate gate prediction.

REFERENCES

1. V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum logic circuits," in Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC), 2005, pp. 272–275.

2. V. Kliuchnikov, D. Maslov, and M. Mosca, “Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates,” *Quantum Inf. Comput.*, vol. 13, no. 7–8, pp. 607–630, 2013.
3. B. Giles and P. Selinger, “Exact synthesis of multiqubit Clifford+T circuits,” *Phys. Rev. A*, vol. 87, no. 3, Art. no. 032332, 2013.
4. V. Gheorghiu, M. Mosca, and P. Mukhopadhyay, “T-count and T-depth of any multi-qubit unitary,” *npj Quantum Inf.*, vol. 8, no. 1, Art. no. 141, 2022.
5. M. Mosca and P. Mukhopadhyay, “A polynomial time and space heuristic algorithm for T-count,” *Quantum Sci. Technol.*, vol. 7, no. 1, Art. no. 015003, 2021.
6. A. Paradis, J. Dekoninck, B. Bichsel, and M. Vechev, “Synthetiq: Fast and versatile quantum circuit synthesis,” *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, pp. 55–82, 2024.
7. H. Nagarajan and Z. Szabó, “Provably Optimal Quantum Circuits with Mixed-Integer Programming,” *arXiv:2510.00649*, 2025.
8. F. Furrutter, G. Muñoz-Gil, and H. J. Briegel, “Quantum circuit synthesis with diffusion models,” *Nat. Mach. Intell.*, vol. 6, pp. 515–524, 2024.
9. Z. Chen and H. Tang, “UDiTQC: U-Net-Style Diffusion Transformer for Quantum Circuit Synthesis,” *arXiv:2501.16380*, 2025.
10. S. Rietsch, A. Y. Dubey, C. Ufrecht, M. Periyasamy, A. Plinge, C. Mutschler, and D. D. Scherer, “Unitary Synthesis of Clifford+T Circuits with Reinforcement Learning,” in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2024, pp. 824–835.
11. L. Theißinger, T. Gerlach, D. Berghaus, and C. Bauckhage, “Beyond Reinforcement Learning: Fast and Scalable Quantum Circuit Synthesis,” *arXiv:2602.15146*, 2026.