

Designing Multi-Objective CNN Architectures for SQL Query Modeling with Evolution Strategies

Pablo Rivas¹ and Donald R. Schwartz²

¹ Department of Computer Science, Baylor University, Waco, TX 76798, USA
Pablo_Rivas@Baylor.edu

² Dept. of Computing Technology, Marist University, Poughkeepsie, NY 12601, USA
Donald.Schwartz@Marist.edu

Abstract. Automated evaluation of open-ended student work remains a challenge in educational technology. In the context of SQL query assessment, existing models often rely on rigid heuristics or underfit architectures that fail to generalize. Here we present a multi-objective neural model whose architecture and hyperparameters are optimized using evolution strategies (ES). Our model jointly predicts query correctness, diagnostic remarks, and numerical grades from raw student submissions. We show that this approach improves classification accuracy and robustness across underrepresented feedback classes, while maintaining interpretability. These findings demonstrate the utility of ES in discovering high-performing configurations for complex assessment tasks.

Keywords: evolution strategies, neural architecture search
Regular Research Paper

1 Introduction

In contemporary education, automated assessment technologies have become valuable tools for supporting student learning, particularly by providing immediate feedback on programming and query languages such as Structured Query Language (SQL). As the complexity and volume of student submissions grow, these systems help students assess and refine their work before it undergoes final evaluation by human instructors. Consequently, researchers and educators have turned to automated grading systems powered by machine learning to enhance the accuracy, efficiency, and reliability of assessment processes [38,47,18]. In this context, automated SQL grading systems offer the dual benefit of providing prompt, constructive feedback and supporting the development of essential database management skills among students [47,18]. Prior studies further demonstrate that such systems can move beyond binary correctness assessments to offer insightful explanations, thereby fostering deeper learning outcomes [18,9,16].

Despite these advances, automated grading of SQL statements remains a challenging task due to the syntactic intricacies and logical complexities inherent in query formulation. Effective solutions must incorporate models capable

of understanding not only the syntactic structure but also the semantic implications of SQL statements. In this regard, Rivas et al. [30] introduced the use of BERT, a transformer-based model known for its contextual representation capabilities, to improve the automatic grading of SQL statements, achieving notable gains in both accuracy and robustness. Building upon this foundation, Rivas and Schwartz [32] proposed attention-based Convolutional Neural Networks (CNNs) for modeling SQL statement correctness, demonstrating that deep learning architectures can successfully capture critical contextual dependencies often overlooked by simpler models. Additionally, Rivas [29] highlighted the critical role of explainable artificial intelligence (XAI) in this domain, advocating for grading models that not only produce correctness scores but also provide meaningful and actionable feedback to guide student learning.

Among the various machine learning models explored for SQL grading, CNNs have emerged as particularly effective due to their ability to extract hierarchical feature representations and their adaptability for processing sequential data [2,5]. The integration of attention mechanisms into CNN architectures further enhances their capacity to focus on the most relevant components of SQL queries, thereby improving the model’s comprehension of complex query structures [46]. This combination of CNNs and attention mechanisms enables the development of more sophisticated grading systems capable of delivering nuanced evaluations that extend beyond simplistic correctness assessments [9,46].

Parallel to advancements in model architectures, the optimization of hyperparameters remains a critical factor in improving machine learning model performance, including that of CNN-based grading systems. Careful tuning of hyperparameters such as learning rate, batch size, and network depth has a profound impact on predictive accuracy and model generalization [36,51]. Effective hyperparameter optimization not only enhances model performance but also mitigates overfitting, ensuring reliable predictions on previously unseen data [51,26]. Evolutionary strategies (ES), including the $(\mu/\rho + \lambda)$ approach, offer a compelling alternative to conventional heuristic methods by systematically navigating the hyperparameter search space through biologically inspired selection and adaptation mechanisms [28,22]. These strategies have consistently demonstrated their effectiveness in optimizing complex deep learning models, including CNNs, across a variety of application domains [44,25].

In this study, we present a comprehensive approach that applies the $(\mu/\rho + \lambda)$ evolution strategy to jointly optimize both the architecture and hyperparameters of a CNN augmented with attention mechanisms. The proposed framework aims to develop a robust automated grading system for SQL statements, capable of producing comprehensive correctness scores, detailed explanatory feedback, and final grade assignments. By synthesizing prior advances in automated SQL grading [30,32], CNN-based SQL understanding, and state-of-the-art hyperparameter optimization through evolutionary strategies, this research contributes a novel and scalable solution to the challenges of automated assessment. Ultimately, this work aspires to advance the state of AI-driven educational tools by delivering

reliable, explainable, and high-performance automated grading systems, thereby fostering effective and engaging learning environments in database education.

2 Related Work

The rapid advancement of artificial intelligence in education has led to significant research efforts aimed at improving automated assessment systems. This section reviews the most relevant prior work across three key areas central to the present study: automated systems for SQL grading, using CNNs for grading tasks in broader contexts, and using ES for hyperparameter optimization. By examining these areas, we establish the foundation upon which the proposed framework builds and identify the gaps this research aims to address.

2.1 SQL Automated Grading Systems

The evolution of automated SQL grading systems has advanced significantly over the past decade. Traditionally, these systems have taken one of two overall approaches: static analysis or dynamic analysis.

The static approach evaluates the structure of the query itself, without actually running the query against a dataset. Each submitted query is compared to a set of answer-key queries. This approach can involve using string similarity metrics [39], automated SQL provers [8], unbounded semirings [7], abstract syntax trees and cosine similarity functions [13], or graph-based approaches [19]. The static analysis approach is frequently good at identifying correct queries, but providing answer-key queries for all possible correct ways to construct a given query can be a huge undertaking. These models are not well-suited to provide partial credit for incorrect queries, although some provide feedback about how many changes would be required to correct the submitted query.

The dynamic approach executes queries against fixed datasets and compares the results with a set of correct answers. Early examples include [33,10,24]. More recent systems show improvements [17,41]. This approach is excellent at identifying incorrect queries, but these systems often struggle with accurately identifying correct queries. Since they are simply comparing the results of student-submitted queries with the answer-key results, they frequently provide inaccurate results. This is especially true when a query just happens to return a correct result (for example, “name all red parts” and “name all parts that were shipped from Norway” might produce the same results) or returns an empty table (an infinite number of queries can yield no answer records).

Given the shortcomings of each approach and recognizing the importance of awarding partial credit to incorrect queries, some systems combine the two in a hybrid approach. Most of these hybrid systems use dynamic analysis to identify the incorrect queries, then the static approach to try to award partial credit [18,45,6,23,11]. More recent examples include [48,12].

More recently, researchers are exploring how Artificial Intelligence can be used to provide feedback on student-submitted SQL queries. Weston, et al. [49]

describe a system that extracts features of students' SQL queries that instructors find significant. It then uses those features to cluster queries to allow instructors to identify trends that appear in the student submissions. Hamtini and Assaf [14] compared the feasibility of using ChatGPT, Gemini and Copilot to grade student SQL queries, comparing the results from the generative AI systems with the grading results of human experts. Initial findings showed that the GenAI approaches were more useful at assisting human experts than they were at taking over the grading tasks themselves but suggested that responses the system made about student errors could be tailored to help the systems learn better.

2.2 CNN-Based Grading Methods

The application of CNNs in automated grading extends beyond SQL, finding success in programming assignment evaluation, natural language response assessment, and structured content analysis. CNNs excel at learning hierarchical feature representations, enabling the capture of both local and global dependencies essential for accurate grading [2,5]. Recent advancements have integrated attention mechanisms into CNN architectures, further improving contextual understanding by directing the model's focus toward semantically relevant input [46].

In programming education, CNNs have been applied to analyze code structure and syntax across various languages. Yang [52] demonstrated the effectiveness of CNNs with recurrent neural filters in processing structured data sequences, highlighting their utility for code analysis tasks. Attention-enhanced CNNs have also shown promise in improving feedback quality by identifying critical code segments that impact correctness [53]. In natural language grading, CNNs have been used to assess text responses, offering nuanced evaluations of coherence and argument completeness. Yin et al. [53] introduced attention-based CNN models for sentence pair modeling, demonstrating their superiority over traditional feature engineering approaches in capturing semantic relationships.

The Convolutional Block Attention Module (CBAM) proposed by Woo et al. further exemplifies the refinement of CNN-based models through attention integration, allowing adaptive feature selection that enhances model interpretability and grading accuracy [50]. Additionally, hybrid architectures combining CNNs with Recurrent Neural Networks (RNNs) have been explored to improve the modeling of sequential dependencies in student submissions, particularly in programming and response generation tasks [20]. Our study uses multi-headed attention heads instead, following a more modern approach similar to transformers.

2.3 Evolutionary Strategies for Hyperparameter Optimization

Evolutionary strategies (ES) have been used for hyperparameter optimization in machine learning, addressing the challenges posed by complex model architectures and large hyperparameter spaces. Inspired by natural selection, ES offer robust search capabilities through adaptive selection, mutation, and recombination [1,4]. Among these, the $(\mu/\rho + \lambda)$ -ES stands out for its efficiency in identifying high-performing configurations, particularly in deep learning models [21,35].

Compared to grid search and random search, which suffer from inefficiency and lack of guidance, ES provide a structured exploration of the search space, avoiding the pitfalls of exhaustive or purely stochastic approaches [55]. While Bayesian optimization offers probabilistic modeling, it can be limited by its dependency on initial samples and assumptions about the search landscape [1,21]. ES methods, in contrast, adaptively refine populations of solutions, yielding competitive results even in high-dimensional optimization problems [35,40].

CMA-ES, a leading ES variant, leverages covariance matrices to capture interdependencies among parameters, facilitating efficient navigation of complex search spaces [1,21]. Applications of ES extend to neural architecture search, where strategies like those proposed by Suganuma et al. successfully optimize network structures for task-specific performance gains [40]. Additionally, the integration of ES with reinforcement learning and multi-objective optimization frameworks further enhances their versatility in discovering architectures that balance performance with computational efficiency [4,27]. Despite higher computational costs, the flexibility and effectiveness of ES make them a valuable tool for hyperparameter and architecture optimization in modern AI systems [54].

3 Methodology

This section outlines the proposed approach for automated SQL grading using a CNN architecture enhanced with attention mechanisms. We also detail the application of the $(\mu/\rho + \lambda)$ Evolutionary Strategy for hyperparameter optimization and provide a comprehensive description of the dataset used for evaluation.

3.1 Neural Network Architecture

We proposed a parameterized CNN to assess SQL query submissions through multi-task learning. The model simultaneously predicts three outputs: (1) query correctness, (2) explanatory remarks, and (3) a numerical grade. Formally, given an input query $\mathbf{x} \in \mathcal{Z}^T$, the model computes a shared latent representation $\mathbf{h} \in \mathbb{R}^d$, from which the outputs are derived through separate prediction heads.

Embedding Layer. The input queries are tokenized and converted into sequences of integers corresponding to a learned vocabulary \mathcal{V} , with size $|\mathcal{V}| = v$. An embedding matrix $\mathbf{E} \in \mathbb{R}^{v \times d_e}$ maps each token to a d_e -dimensional continuous vector, where d_e is the embedding dimension, leading to a dense representation:

$$\mathbf{X}_e = \text{Embedding}(\mathbf{x}) \in \mathbb{R}^{T \times d_e}. \quad (1)$$

Convolutional Feature Extractors. The embedded sequence is processed by multiple 1D convolutional layers with varying kernel sizes k_1, \dots, k_n and F filters per layer. For each kernel size k_i , the convolutional transformation is given by:

$$\mathbf{C}^{(i)} = \text{ReLU}(\text{Conv1D}_{k_i}(\mathbf{X}_e)). \quad (2)$$

The outputs of the convolutional layers are concatenated along the feature dimension, producing a comprehensive local feature representation.

Attention Mechanisms. To capture long-range dependencies, the model uses either standard additive attention or Multi-Head Attention (MHA) [42], depending on the hyperparameter used. When MHA is used, the attention output is:

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_H) \mathbf{W}^O, \quad (3)$$

where \mathbf{Q} , \mathbf{K} , and \mathbf{V} are query, key, and value matrices derived from the input features, and H is the number of attention heads.

Pooling and Bottleneck Layer. To reduce the variable-length sequence representations to fixed-size vectors, the model applies either global average pooling, global max pooling, or a concatenation of both. The pooled features are passed through a bottleneck layer implemented as a fully connected layer with dimensionality d_b and activation function σ_b :

$$\mathbf{h} = \sigma_b(\mathbf{W}_b \mathbf{z} + \mathbf{b}_b), \quad (4)$$

where \mathbf{z} is the concatenated pooled representation, and $\mathbf{W}_b \in \mathbb{R}^{d_b \times \dim(\mathbf{z})}$.

Output Heads. The shared latent representation \mathbf{h} is connected to three separate prediction heads:

1. Correctness: A sigmoid-activated neuron computes the binary correctness probability:

$$\hat{y}_{\text{correct}} = \sigma(\mathbf{w}_c^\top \mathbf{h} + b_c). \quad (5)$$

2. Remarks: A softmax layer predicts one of four feedback remarks:

$$\hat{\mathbf{y}}_{\text{remarks}} = \text{softmax}(\mathbf{W}_r \mathbf{h} + \mathbf{b}_r). \quad (6)$$

3. Grade: A regression output predicts the normalized grade using a sigmoid activation:

$$\hat{y}_{\text{grade}} = \sigma(\mathbf{w}_g^\top \mathbf{h} + b_g). \quad (7)$$

Loss Function. The model is trained using a weighted multi-objective loss:

$$L = \lambda_c L_{\text{BCE}}(\hat{y}_{\text{correct}}, y_{\text{correct}}) + \lambda_r L_{\text{BCE}}(\hat{\mathbf{y}}_{\text{remarks}}, \mathbf{y}_{\text{remarks}}) + \lambda_g L_{\text{MSE}}(\hat{y}_{\text{grade}}, y_{\text{grade}}),$$

where L_{BCE} denotes the binary cross-entropy loss, L_{MSE} the mean squared error loss, and $\lambda_c = 0.142$, $\lambda_r = 0.740$, $\lambda_g = 0.118$ are empirically determined weights.

Optimization. Training is performed using either the Adam or RMSprop optimizer, with a learning rate η selected through hyperparameter tuning. Early stopping and learning rate reduction strategies are employed to prevent overfitting and ensure convergence.

3.2 Hyperparameter Optimization Using Evolutionary Strategies

To optimize the hyperparameters of the proposed neural architecture, we employ the $(\mu/\rho + \lambda)$ -ES, a population-based stochastic optimization algorithm inspired by natural selection. This method is particularly suitable for exploring complex, high-dimensional, and mixed-type hyperparameter spaces such as ours [31].

Algorithmic Framework. At each generation, a parent population of size μ is selected based on the highest validation Area Under the Curve (AUC) scores. Offspring are generated by recombining ρ randomly selected parents from this elite pool, producing λ new candidate solutions. The evolutionary cycle consists of selection, recombination, mutation, and evaluation steps.

Recombination. Given a set of ρ parents $\hat{f}_{\mathbf{p}_1}, \dots, \mathbf{p}_\rho \mathcal{G}$, the offspring \mathbf{o} is produced by parameter-wise recombination:

$$o_j = \begin{cases} \frac{1}{\rho} \sum_{i=1}^{\rho} p_{i,j} & \text{if } \theta_j \in \mathbb{R} \text{ (continuous parameter),} \\ \text{round} \left(\frac{1}{\rho} \sum_{i=1}^{\rho} p_{i,j} \right) & \text{if } \theta_j \in \mathbb{Z} \text{ (integer parameter),} \\ \text{random_choice}(\hat{f}_{p_{1,j}}, \dots, p_{\rho,j} \mathcal{G}) & \text{if } \theta_j \in \text{categorical.} \end{cases}$$

Here, θ_j denotes the j -th hyperparameter, and the offspring parameter o_j inherits its value based on the type of parameter.

Mutation. Each offspring undergoes probabilistic mutation with rate γ . Mutation strategies are adapted based on parameter types:

- Continuous Parameters: Additive Gaussian noise:

$$o_j = |o_j + \mathcal{N}(0, \sigma^2)|.$$

- Integer Parameters: Random integer offset within a predefined range:

$$o_j = \max(2, o_j + \Delta), \quad \Delta \sim \text{Uniform}[k, k].$$

- Categorical Parameters: Random reassignment with probability γ .

Mutation parameters, including the mutation rate γ and standard deviation σ , are adaptively adjusted using the 1/5th success rule [3]:

$$\gamma = \begin{cases} \gamma/a & \text{if } P_s > \frac{1}{5}, \\ \gamma \cdot a & \text{if } P_s < \frac{1}{5}, \end{cases}$$

where P_s is the success rate over G generations, and $a \in [0.85, 1)$ is a predefined and well-known adaptation factor.

Evaluation and Selection. Each offspring is evaluated by training the neural network with the proposed hyperparameters. Performance is assessed using the validation AUC. Individuals with previously evaluated configurations retrieve their performance scores directly, avoiding redundant evaluations.

Parallel Execution. To efficiently manage computational resources, the evaluation of candidate solutions is parallelized across multiple GPUs using a job scheduling mechanism. This enables simultaneous model training runs.

Termination Criterion. The algorithm iterates for a maximum of G_{\max} generations or until convergence, defined by no significant improvement in the average validation AUC across successive generations.

Summary. The $(\mu/\rho + \lambda)$ Evolutionary Strategy offers an effective and adaptive mechanism for hyperparameter optimization, enabling the discovery of high-performing configurations in a complex search space. By integrating adaptive mutation rates, intelligent recombination, and parallel evaluation, the approach balances exploration and exploitation, leading to improved model performance.

3.3 Dataset Description

The dataset used in this study is a curated and augmented collection of SQL query submissions from undergraduate coursework, designed to support model training and evaluation in structured query understanding. The core dataset, publicly available in [15], initially comprised 675 annotated submissions. These samples were labeled with correctness, numeric grades, and explanatory remarks (*Correct*, *Partially Correct*, *Non-Interpretable*, and *Cheating*) to facilitate supervised learning [32,34]. To expand coverage and improve model generalization across a broader range of query styles and schema complexities, we augmented the original dataset with thousands of additional SQL submissions sourced from a college-level introductory database course. After preprocessing and filtering, the final training and validation corpus consists of 4,918 student-submitted SQL queries. Of these, approximately 55% are labeled as correct, and the mean grade is 84.3 out of 100, indicating a moderate class imbalance and a skew toward higher-performing submissions.

Each query in this corpus was preprocessed using word-level tokenization [43], resulting in a vocabulary of 1,480 tokens. All sequences were padded to a fixed length of 219 tokens to ensure uniformity during batch training.

In addition to the training corpus, a separate test set of 2,577 previously unseen SQL submissions was held out for final model evaluation. This set exhibits slightly higher correctness rates, with 59% of the queries labeled correct and an average grade of 87.4. The inclusion of this test set ensures rigorous validation on student samples not encountered during model development.

An overview of the dataset is shown in Table 1, and an example SQL query from the dataset is provided below.

```
SELECT DISTINCT s.SNum FROM Snacks s, Vendors v, Delivers d, Parks p
WHERE v.HQLoc = 'LosAngeles' AND p.Location = 'LosAngeles'
AND v.VNum = d.VNum AND s.SNum = d.SNum AND p.PNum = d.PNum;
```

4 Experiments and Results

4.1 Experimental Setup

The experimental evaluation focuses on optimizing the neural network architecture and its hyperparameters using the $(\mu/\rho + \lambda)$ -ES. The objective is to

Table 1: Sample Data Extracted From Expanded Dataset

Submitted Answer	Correct?	Remark	Grade
SELECT * FROM parks WHERE location = 'LA';	1	Correct	100
Select * From Delivers Where Amnt >= 300 <= 750;	0	Partially	20
⋮	⋮	⋮	⋮
Total count: 5,398	Avg: 0.58	Total: 4	Avg: 85

maximize the validation AUC for the correctness prediction task while simultaneously producing meaningful feedback remarks and grade estimations.

Optimization Objectives. The hyperparameter optimization process focused on tuning both architectural and training-related parameters. On the architectural side, the vocabulary size v was varied in the range [2, 65536], and the embedding dimension d_e was explored within [2, 4096]. The number of convolutional filters F spanned values from 2 to 2048. Kernel size configurations included both single and multi-kernel combinations selected from the set $\{3, 4, 5, 6, 7, 8\}$. When multiple kernel sizes were specified, such as 3|4|5, the model applied parallel convolutional layers, one per kernel size, and concatenated their outputs along the feature dimension. The use of multi-head attention was treated as a binary decision (True or False), with the number of attention heads H ranging from 1 to 52. Three pooling strategies were considered: average pooling (avg), max pooling (max), and a hybrid of both (avg_max). The bottleneck layer dimension d_b ranged from 1 to 1024, and the activation functions examined included relu, tanh, gelu, and selu. Dropout rates γ were sampled continuously between 0.0001 and 0.9990. For the training configuration, the optimizer was chosen from adam and rmsprop, and the learning rate η was optimized in the interval $[3.32 \cdot 10^{-7}, 5.41 \cdot 10^{-3}]$. These hyperparameters were explored using the ES discussed earlier.

Search Process. The ES optimization was performed over a population of $\mu = 16$ parents and $\lambda = 16$ offspring per generation, for a total of 25 generations. Recombination and mutation operators were applied to explore the hyperparameter space, and candidate configurations were evaluated based on the validation AUC. Mutation rates and perturbation magnitudes were adaptively adjusted using the 1/5th success rule to balance exploration and exploitation of the parameter space. This process was repeated with different initial populations.

Evaluation Protocol. Each individual configuration was evaluated using a stratified train-validation split, and models were trained until convergence using early stopping and learning rate scheduling. Validation AUC scores were recorded and used to guide the evolutionary search. Parallel GPU resources were leveraged to efficiently evaluate multiple configurations concurrently.

4.2 Results from $(\mu/\rho + \lambda)$ -ES

Table 2 summarizes the top 10 hyperparameter configurations identified through evolutionary search, ranked by validation AUC on the correctness prediction task. Each row corresponds to a unique model instance defined by architectural and training-related settings, including vocabulary size, embedding dimension, number of convolutional filters, kernel size combinations, use of multi-head attention (MHA), pooling strategy, bottleneck layer characteristics, optimizer type, dropout rate, and learning rate. These configurations reflect the most performant trade-offs between expressiveness and regularization observed during training.

To visualize the structure of the high-dimensional hyperparameter space, we projected the results of all evaluated configurations into two dimensions using principal component analysis (PCA). Each hyperparameter configuration, including both numeric and categorical features, was encoded and standardized prior to projection. Categorical variables were transformed via one-hot encoding, and the full feature matrix was scaled using z-score normalization. We then applied PCA to reduce the 12-dimensional configuration space to two principal components, capturing the dominant axes of variation among configurations.

The resulting 2D coordinates were used to fit a smooth interpolation surface over the validation AUC values using radial basis function (RBF) interpolation. This surface approximates the underlying error landscape of the model’s performance across the search space. As shown in Fig. 1, regions with higher AUC are visible with a different color on the surface. The star indicates the best-performing configuration discovered during search.

Notably, the initial grid search was used to populate a diverse set of configurations across the landscape, and this served as the initialization for the subsequent evolution strategy procedure described earlier in Section 3.2. This visualization illustrates how the ES builds upon broad initial coverage and adapts search toward local optima in performance.

4.3 Analysis of Hyperparameter Sensitivity

To better understand the influence of individual hyperparameters on model performance, we performed a targeted analysis using Gaussian Process Regression (GPR) [37]. For each continuous or integer hyperparameter, we extracted its

Table 2: Top 10 Hyperparameter Configurations (Ranked by Validation AUC)

Vocab	Emb.	Filters	K. Sizes	MHA	Heads	Pool	Bneck	A.F.	Drop	Opt	LR
4596	321	246	3,4,5,7	Y	4	avg_max	92	gelu	0.0538	adam	5.03e-5
4440	347	242	3,4,5,6,7,8	N		max	86	gelu	0.0412	adam	7.18e-5
4130	296	220	3,4,5,6,7,8	N		avg_max	85	gelu	0.0543	adam	6.34e-5
4562	320	243	3,4,5,7	N		avg_max	90	selu	0.0593	rmsprop	6.17e-5
4440	348	243	3,4,5,6,7,8	N		avg_max	85	gelu	0.0563	rmsprop	8.66e-5
4452	359	238	3,4,5,6,7,8	Y	5	avg_max	89	relu	0.0576	adam	6.28e-5
2190	134	104	3,4,5,6,7,8	N		avg_max	36	tanh	0.0840	adam	8.86e-5
4504	336	243	3,4,5,6,7,8	Y	6	avg_max	85	gelu	0.0531	adam	8.31e-5
4436	348	242	3,4,5,6,7,8	N		max	81	gelu	0.0524	adam	5.91e-5
4000	384	300	3,4,5,6,7,8	N		max	85	relu	0.0500	rmsprop	5.00e-5

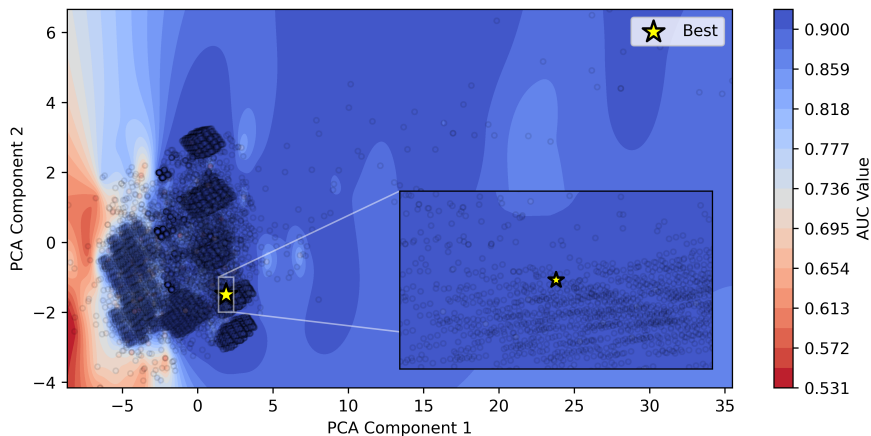


Fig. 1: PCA projection of the hyperparameter space with RBF-interpolated validation AUC values. The contours indicate estimated model performance, with cooler regions representing higher AUC. The star marks the best parameters.

values across all trials, computed the corresponding validation AUC scores, and fit a GPR model to estimate the relationship between the parameter and model performance. Each parameter was log-transformed to improve the fit and plotted against the validation AUC. We also highlighted the top three configurations yielding the highest AUC to identify optimal regions in the parameter space.

This approach allowed us to isolate the effect of each parameter while accounting for noise and variance in other dimensions. Shaded regions around each regression curve indicate the 68% and 95% confidence intervals, helping to visualize uncertainty and performance trends. These plots, shown in Fig. 2–5, support the analysis of sensitivity to learning rate, filter count, vocabulary size, attention heads, bottleneck dimensionality, dropout rate, and embedding size.

To complement the analysis of continuous hyperparameters, we also examined the distribution of categorical parameters among the top-performing configurations. Fig. 6 (left) shows the validation AUC distributions for each optimizer using a boxplot, emphasizing the relative performance of adam and rmsprop. The right plot displays a histogram of kernel size combinations among the top 0.5% of configurations, revealing a strong preference for multi-scale convolution (e.g., 3|4|5|6|7|8). These categorical analyses help identify non-numeric settings that consistently appeared in the best solutions.

4.4 Discussion of Results and Model Improvements

We now present updated results for the final model configurations optimized through evolutionary strategies. As before, the evaluation is organized into three learning tasks: correctness classification (Model C), remark classification (Model R),

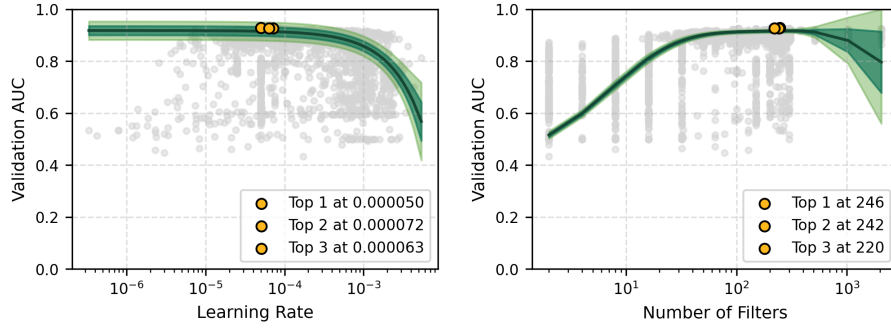


Fig. 2: Effect of learning rate (left) and number of convolutional filters (right) on validation AUC. Top-performing configurations are marked with a circle.

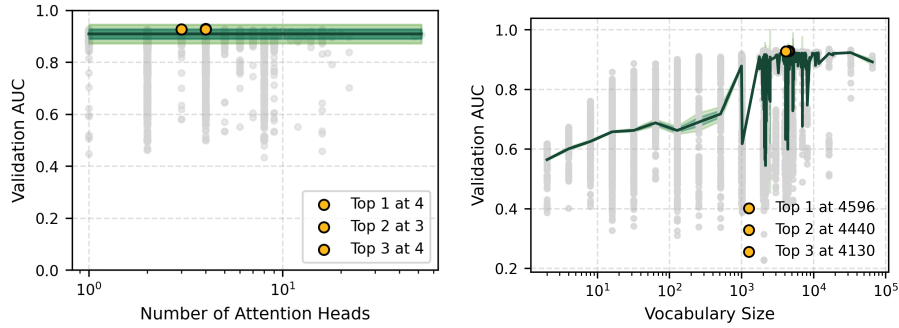


Fig. 3: Impact of number of attention heads (left) and vocabulary size (right) on model performance.

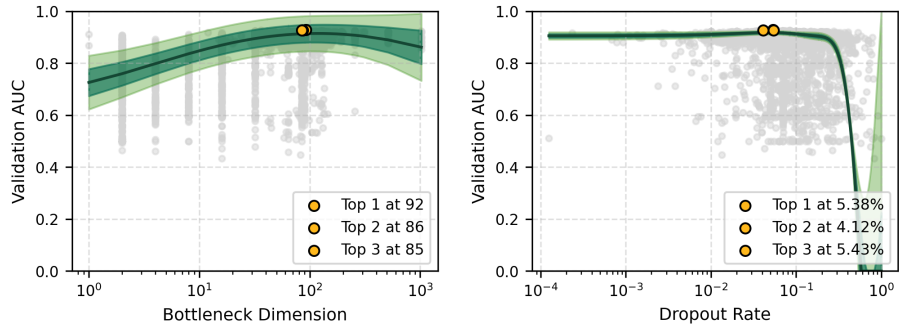


Fig. 4: AUC sensitivity to bottleneck layer dimension (left) and dropout rate (right). A low dropout value consistently outperformed higher rates.

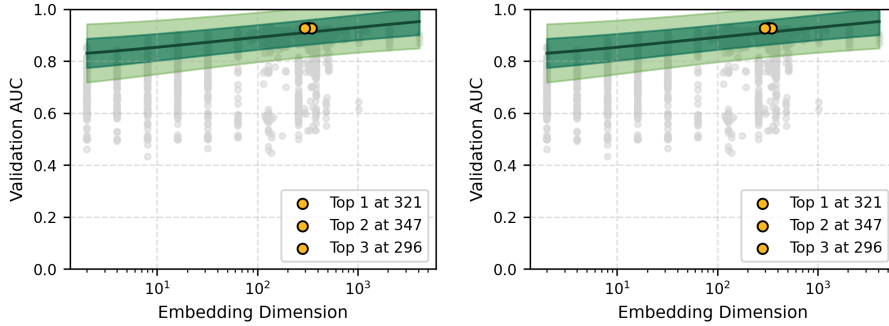


Fig. 5: Relationship between embedding dimension and validation AUC. Larger embedding dimensions were correlated with improved performance.

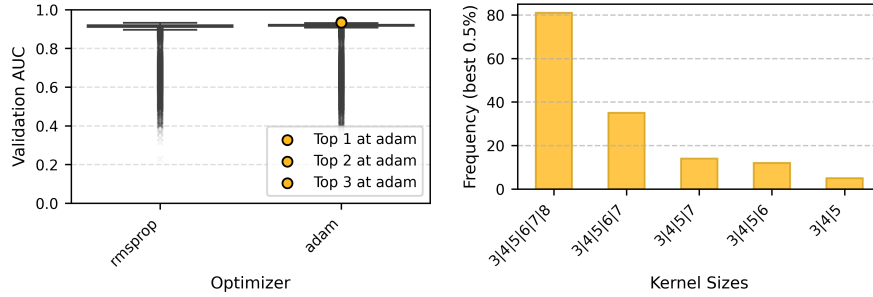


Fig. 6: Left: Distribution of validation AUCs by optimizer type; Right: Frequency of kernel size combinations among the top 0.5% of configurations.

and grade regression (Model G). All models were evaluated using leave-one-out (LOO) cross-validation, for comparability with prior benchmarks [29,32,34].

Correctness Classification (Model C). Fig. 7 displays the confusion matrix and ROC curve for the binary correctness prediction task. The model achieves a balanced and overall accuracy of 88%, improving over earlier baselines. Precision and recall are symmetric across both classes (Correct and Incorrect), with F_1 -scores of 0.89 and 0.85, respectively. These gains reflect improved discrimination and calibration despite the underlying class imbalance shown earlier in Table 1.

Remark Classification (Model R). Fig. 8 shows precision-recall curves and the confusion matrix for Model R. Accuracy for this task reached 88%, a strong result given the increased complexity of multi-class prediction. Performance was especially strong on the dominant Correct and Partially Correct classes, with F_1 -scores of 0.90 and 0.85, respectively. Notably, smaller classes such as Non Interpretable and Cheating saw improvements compared to prior work,

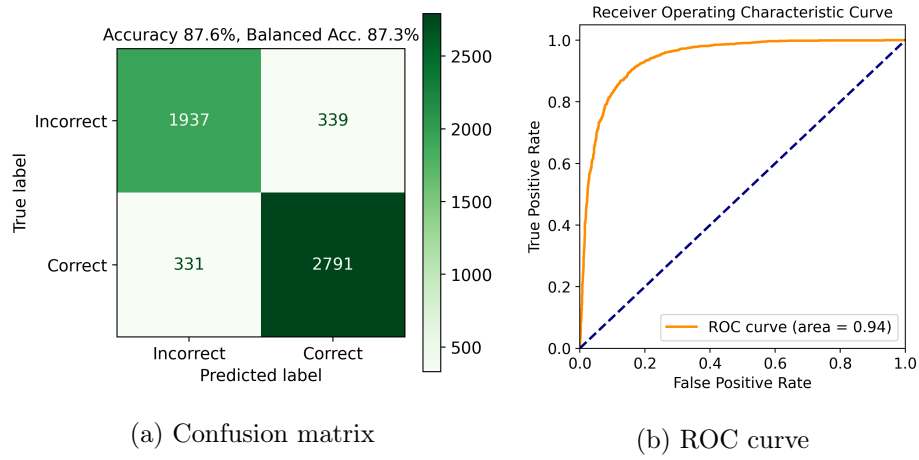


Fig. 7: Correctness classification model (Model C): (a) Confusion matrix and (b) ROC curve.

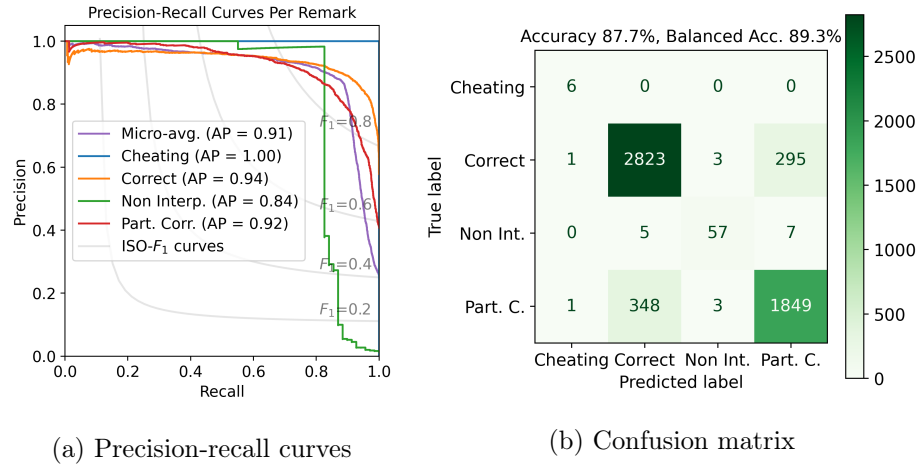


Fig. 8: Remark classification model (Model R): (a) Precision-recall curves and (b) Confusion matrix.

achieving F_1 -scores of 0.86 and 0.86. These gains indicate that the revised model generalizes better across both high- and low-frequency labels.

Grade Prediction (Model G). Fig. 9 presents results for the grade regression model. The updated model exhibits stronger variance capture, with an explained variance (EV) of 0.472, indicating improved sensitivity to signal fluctuations in grade labels. However, the R^2 metric is now negative (-0.015), which suggests that while the model approximates the mean reasonably well, it struggles with

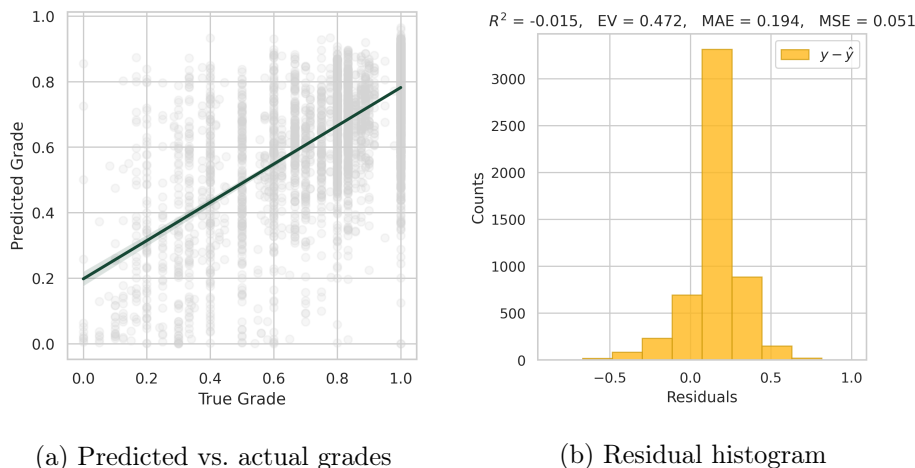


Fig. 9: Grade prediction model (Model G): (a) True vs. predicted grades and (b) distribution of residuals $y - \hat{y}$.

precise grade-level regression. Mean absolute error (MAE) increased to 0.194, and MSE increased to 0.051. These outcomes reflect a more cautious regressor that sacrifices precision to avoid overfitting on noisy or imbalanced labels.

Summary of Performance Improvements. Table 3 reports the full set of evaluation metrics for all models. The right-hand Δ columns indicate gains relative to the previous baseline. Binary and multi-class classification metrics indicate consistent gains, especially in terms of macro-averaged F_1 -scores and precision across minority labels. The grade regression task shows mixed results, with gains in variance capture but a degradation in R^2 , underscoring the challenge of modeling numerical grades with limited supervision. Overall, the combination of architectural search and objective-specific tuning yields robust classification improvements with moderate trade-offs in regression fidelity.

5 Conclusion

We presented a principled framework for optimizing deep neural network architectures using evolutionary strategies for the task of automated SQL grading. The approach integrated architectural and training hyperparameters into a unified search space, guided by a $(\mu/\rho + \lambda)$ -ES algorithm initialized through grid-based sampling. This method enabled efficient navigation of high-dimensional parameter combinations, yielding robust multi-task models.

Performance gains were most pronounced in the classification of correctness and feedback remarks. Correctness classification achieved a balanced accuracy of 85%, with consistent improvements in F_1 -score for both the Correct and Incorrect classes. For the remark classification task, macro-level performance

Table 3: Classification and Regression Performance Analysis

Class Evaluated	Evaluation Metrics			
	<i>Prec.</i> Δ	<i>Recall</i> Δ	<i>F₁-score</i> Δ	<i>Support</i> Δ
Incorrect	0.88 -0.03	0.77 $+0.08$	0.82 $+0.03$	2276 $-$
Correct	0.85 $+0.04$	0.93 -0.04	0.89 $+0.01$	3122 $-$
Accuracy			0.86 $+0.02$	5398 $-$
Balanced Accuracy			0.85 $+0.02$	5398 $-$
Cheating	0 $+0.75$	0 $+1.00$	0 $+0.86$	6 $-$
Correct	0.79 $+0.10$	0.97 -0.07	0.87 $+0.03$	3122 $-$
Non Interpretable	0 $+0.90$	0 $+0.83$	0 $+0.86$	69 $-$
Partially Correct	0.91 -0.05	0.64 $+0.20$	0.65 $+0.20$	2201 $-$
Accuracy			0.82 $+0.06$	5398 $-$
Balanced Accuracy			0.41 $+0.46$	5398 $-$
Regression	<i>R</i> ² Δ	<i>EV</i> Δ	<i>MAE</i> Δ	<i>MSE</i> Δ
\hat{y} = Grade	0.427 -0.442	0.429 $+0.043$	0.113 $+0.081$	0.029 $+0.022$

saw a significant rise in balanced accuracy, from 0.41 to 0.87, supported by large F_1 -score gains in underrepresented classes such as Non Interpretable and Cheating. These results suggest that the model architecture learned more discriminative representations under sparse supervision and noisy inputs.

The grade regression task showed mixed results. While explained variance improved slightly, the R^2 value declined, indicating poorer alignment with fine-grained grade targets. Nevertheless, the model maintained low error metrics, with a mean absolute error (MAE) of 0.113 and a mean squared error (MSE) of 0.029, suggesting its utility in producing coarse but reliable grade predictions.

Overall, the experimental evidence supports the use of evolutionary strategies as a viable mechanism for model discovery in supervised learning contexts characterized by heterogeneous outputs and label sparsity. Future work will explore extensions to multitask learning and reinforcement-based feedback integration, with the aim of developing more adaptive assessment systems that align with real-world educational constraints.

Acknowledgments. Part of this work was funded by the National Science Foundation under grant CNS-2136961.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bartz-Beielstein, T., Branke, J., Mehnen, J., Mersmann, O.: Evolutionary algorithms. Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery **4**, 178–195 (2014). <https://doi.org/10.1002/widm.1124>

2. Bellot, P., Campos, G., Pérez-Enciso, M.: Can deep learning improve genomic prediction of complex human traits? *Genetics* **210**, 809–819 (2018). <https://doi.org/10.1534/genetics.118.301298>
3. Beyer, H.G.: *The Theory of Evolution Strategies*. Natural Computing Series, Springer Berlin, Heidelberg, 1 edn. (2001). <https://doi.org/https://doi.org/10.1007/978-3-662-04378-3>
4. Bucheli, V., Pabón, O., Ordóñez, H.: Evolutionary algorithms guided by erdős-rényi complex networks. *Peerj Computer Science* **10**, e1773 (2024). <https://doi.org/10.7717/peerj-cs.1773>
5. Cai, R., Xu, B., Zhang, Z., Yang, X., Li, Z., Liang, Z.: An encoder-decoder framework translating natural language to database queries pp. 3977–3983 (2018). <https://doi.org/10.24963/ijcai.2018/553>
6. Chandra, B., Banerjee, A., Hazra, U., Joseph, M., Sudarshan, S.: Edit based grading of sql queries. In: 8th ACM IKDD CODS and 26th COMAD, pp. 56–64 (2021)
7. Chu, S., Murphy, B., Roesch, J., Cheung, A., Suci, D.: Axiomatic foundations and algorithms for deciding semantic equivalences of sql queries. *Proceedings of the VLDB Endowment* **11**, 1482–1495 (2018). <https://doi.org/10.14778/3236187.3236200>
8. Chu, S., Li, D., Wang, C., Cheung, A., Suci, D.: Demonstration of the cosette automated sql prover. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. pp. 1591–1594 (2017). <https://doi.org/10.1145/3035918.3058728>
9. Coyne, J., Summers, S., Wood, D.: Enhancing faculty grading productivity using robotic process automation (rpa): the development of a structured query language (sql) automated grading tool. *Issues in Accounting Education* **39**, 55–68 (2024). <https://doi.org/10.2308/issues-2023-037>
10. Dekeyser, S., de Raadt, M., Lee, T.Y.: Computer assisted assessment of sql query skills. In: *Proceedings of the 18th Australasian Database Conference (ADC 2007)*. vol. 63, pp. 53–62. Australian Computer Society Inc. (2007)
11. Fabijanic, M., Dambic, G., Fulanovic, B.: A novel system for automatic, configurable and partial assessment of student SQL queries. In: *43rd International Convention on Information, Communication and Electronic Technology (MIPRO 2020)*. pp. 832–837. IEEE (2020). <https://doi.org/10.23919/MIPRO48935.2020.9245264>, <https://doi.org/10.23919/MIPRO48935.2020.9245264>
12. Fabijanić, M., Mekterović, I.: Partial sql query assessment. In: *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*. pp. 1317–1322 (2023). <https://doi.org/10.23919/MIPRO57284.2023.10159706>, <https://api.semanticscholar.org/CorpusID:259299956>
13. Ganesan, S., Gong, T., Lee, J.: Sqllearn: automated sql statement assessment using structure-based analysis. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education v. 2* pp. 1644–1645 (2024). <https://doi.org/10.1145/3626253.3635607>
14. Hamtini, T., Assaf, A.J.: Exploring the efficacy of genai in grading sql query tasks: a case study. *Cybernetics and Information Technologies* **24**, 102–111 (2024). <https://doi.org/10.2478/cait-2024-0027>
15. Jinshui, W.: Combining dynamic and static analysis for automated grading SQL statements (May 2022). <https://doi.org/10.5281/zenodo.6526769>, <https://doi.org/10.5281/zenodo.6526769>
16. Kanchan, S., Kalsekar, S., Dubey, N., Fernandes, C., Hamdare, S.: Automated sql grading system pp. 701–708 (2021). https://doi.org/10.1007/978-981-33-4543-0_74

17. Kleerekoper, A., Schofield, A.: Sql tester: an online sql assessment tool and its impact. In: Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education. pp. 87–92 (2018). <https://doi.org/10.1145/3197091.3197124>
18. Kleiner, C., Tebbe, C., Heine, F.: Automated grading and tutoring of sql statements to improve student learning p. 161–168 (2013). <https://doi.org/10.1145/2526968.2526986>, <https://doi.org/10.1145/2526968.2526986>
19. Köberlein, L., Probst, D., Lenz, R.: Graph-based qss: A graph-based approach to quantifying semantic similarity for automated linear sql grading (2025). <https://doi.org/10.18420/BTW2025-13>, <https://dl.gi.de/handle/20.500.12116/45875>
20. Latif, S., Rana, R., Khalifa, S., Jurdak, R., Epps, J.: Direct modelling of speech emotion from raw speech (2019). <https://doi.org/10.21437/interspeech.2019-3252>
21. Loshchilov, I.: Cma-es with restarts for solving cec 2013 benchmark problems (2013). <https://doi.org/10.1109/cec.2013.6557593>
22. Loshchilov, I., Hutter, F.: Cma-es for hyperparameter optimization of deep neural networks (2016). <https://doi.org/10.48550/arxiv.1604.07269>
23. Nayak, S., Agarwal, R., Khatri, S.K., Mohammadian, M.: Student outcome assessment on structured query language using rubrics and automated feedback generation. *International Journal of Advanced Computer Science and Applications* **15**(3) (2024). <https://doi.org/10.14569/IJACSA.2024.0150374>, <http://dx.doi.org/10.14569/IJACSA.2024.0150374>
24. Prior, J.C., Lister, R.: The backwash effect on sql skills grading. *ACM SIGCSE Bulletin* **36**(3), 32–36 (2004). <https://doi.org/10.1145/1007996.1008008>
25. Puentes-Garzón, D., Barrios-Hernandez, C., Navaux, P.: Hyperparameter optimization for convolutional neural networks with genetic algorithms and bayesian optimization pp. 1–5 (2022). <https://doi.org/10.1109/la-cci54402.2022.9981104>
26. Pérez-Enciso, M., Zingaretti, L.: A guide on deep learning for complex trait genomic prediction. *Genes* **10**, 553 (2019). <https://doi.org/10.3390/genes10070553>
27. Randriambololona, A., Shaeri, M., Sarabi, S.: Prediction accuracy of artificial neural networks in thermal management applications subject to neural network architectures (2022). <https://doi.org/10.11159/htff22.175>
28. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning* **87**, 357–380 (2012). <https://doi.org/10.1007/s10994-012-5286-7>
29. Rivas, P.: Explainable ai for sql grading: a practical approach with multi-task cnns pp. 57–73 (2025). https://doi.org/10.1007/978-3-031-86623-4_5
30. Rivas, P., Schwartz, D., Quevedo, E.: Bert goes to sql school: improving automatic grading of sql statements pp. 83–90 (2023). <https://doi.org/10.1109/csce60160.2023.00019>
31. Rivas, P.: Deep learning evolved: Overcoming sub-optimal local minima with $(\mu/\rho + \lambda)$ -evolution strategies. In: 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE). pp. 37–45 (2023). <https://doi.org/10.1109/CSCE60160.2023.00012>
32. Rivas, P., Schwartz, D.R.: Modeling sql statement correctness with attention-based convolutional neural networks. In: 2021 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 64–71 (2021). <https://doi.org/10.1109/CSCI54926.2021.00086>

33. Sadiq, S., Orlowska, M., Sadiq, W., Lin, J.: Sqlator: an online sql learning workbench. In: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. pp. 223–227 (2004)
34. Schwartz, D.R., Rivas, P.: An automated sql query grading system using an attention-based convolutional neural network. In: The 18th International Conference on Frontiers in Education: Computer Science and Computer Engineering. pp. 1–12 (2022)
35. Shirakawa, S., Iwata, Y., Akimoto, Y.: Dynamic optimization of neural network structures using probabilistic modeling (2018). <https://doi.org/10.48550/arxiv.1801.07650>
36. Smith, L.: A disciplined approach to neural network hyper-parameters: part 1 – learning rate, batch size, momentum, and weight decay (2018). <https://doi.org/10.48550/arxiv.1803.09820>
37. Smith, M., Lawrence, N.: gaussian process regression for binned data (2018). <https://doi.org/10.48550/arxiv.1809.02010>
38. Sokac, M.: Automated grading through contrastive learning: a gradient analysis and feature ablation approach. *Machine Learning and Knowledge Extraction* **7**, 41 (2025). <https://doi.org/10.3390/make7020041>
39. Štajduhar, I., Mauša, G.: Using string similarity metrics for automated grading of sql statements. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 1250–1255. IEEE (2015). <https://doi.org/10.1109/mipro.2015.7160467>
40. Suganuma, M., Shirakawa, S., Nagao, T.: A genetic programming approach to designing convolutional neural network architectures pp. 497–504 (2017). <https://doi.org/10.1145/3071178.3071229>
41. Trongratsameethong, A., Vichianroj, P.: Asqlag - automated sql assignment grading system for multiple dbms. *Journal of Technology and Innovation in Tertiary Education Siam Technology College* **vol.1**, no.42–62 (2018). <https://doi.org/10.14456/jti.2018.4>
42. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017). <https://doi.org/10.48550/arxiv.1706.03762>
43. Vijayarani, S., Janani, R., et al.: Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)* **3**(1), 37–47 (2016)
44. Vincent, A., Jidesh, P.: An improved hyperparameter optimization framework for automl systems using evolutionary algorithms (2022). <https://doi.org/10.21203/rs.3.rs-1781731/v1>
45. Wang, J., Zhao, Y., Tang, Z., Xing, Z.: Combining dynamic and static analysis for automated grading sql statements. *J Netw Intell* **5**(4), 179–190 (2020)
46. Wanjiru, B., Bommel, P., Hiemstra, D.: Towards a generic model for classifying software into correctness levels and its application to sql pp. 37–40 (2023). <https://doi.org/10.1109/seeng59157.2023.00012>
47. Wanjiru, B., Bommel, P., Hiemstra, D.: Sensitivity of automated sql grading in computer science courses pp. 283–299 (2024). https://doi.org/10.1007/978-3-031-65522-7_26
48. Wanjiru, B., Bommel, P.v., Hiemstra, D.: Dynamic and partial grading of sql queries. *Journal of Engineering Research and Sciences* **3**, 1–14 (2024). <https://doi.org/10.55708/js0308001>

49. Weston, M., Sun, H., Herman, G.L., Benotman, H., Alawini, A.: Echelon: an ai tool for clustering student-written sql queries. 2021 IEEE Frontiers in Education Conference (FIE) pp. 1–8 (2021). <https://doi.org/10.1109/fie49875.2021.9637203>
50. Woo, S., Park, J., Lee, J., Kweon, I.: Cbam: convolutional block attention module pp. 3–19 (2018). https://doi.org/10.1007/978-3-030-01234-2_1
51. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: theory and practice. *Neurocomputing* **415**, 295–316 (2020). <https://doi.org/10.1016/j.neucom.2020.07.061>
52. Yang, Y.: Convolutional neural networks with recurrent neural filters (2018). <https://doi.org/10.18653/v1/d18-1109>
53. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* **4**, 259–272 (2016). https://doi.org/10.1162/tacl_a_00097
54. Zelinka, I., Davendra, D., Roman, S., Jašek, R.: Do evolutionary algorithm dynamics create complex network structures? *Complex Systems* **20**, 127–140 (2011). <https://doi.org/10.25088/complexsystems.20.2.127>
55. Zhou, J., Zheng, H., Li, S., Hao, Q., Zhang, H., Gao, W., Wang, X.: A knowledge-guided competitive co-evolutionary algorithm for feature selection. *Applied Sciences* **14**, 4501 (2024). <https://doi.org/10.3390/app14114501>