Expanded Report of DREAM

Due to the space limitations, in the `Design` section of our paper, we only provide a brief introduction to the calculation of conditional probabilities in DREAM. This report provides detailed experiments and results, search priorities, and a feedback-driven search example to elaborate on our design.

The Detailed Experiment to Calculate Conditional Probability

Based on summarized observations from feedback data, DREAM selects an optimal action that is the most possible to improve the current search by calculating the conditional probability $\mathbb{P}(Action | A, C, G, W)$. Although the existing studies have mentioned that several actions can improve the model performance in some situations, it is still difficult to directly measure the actual effect of each action under different conditions.

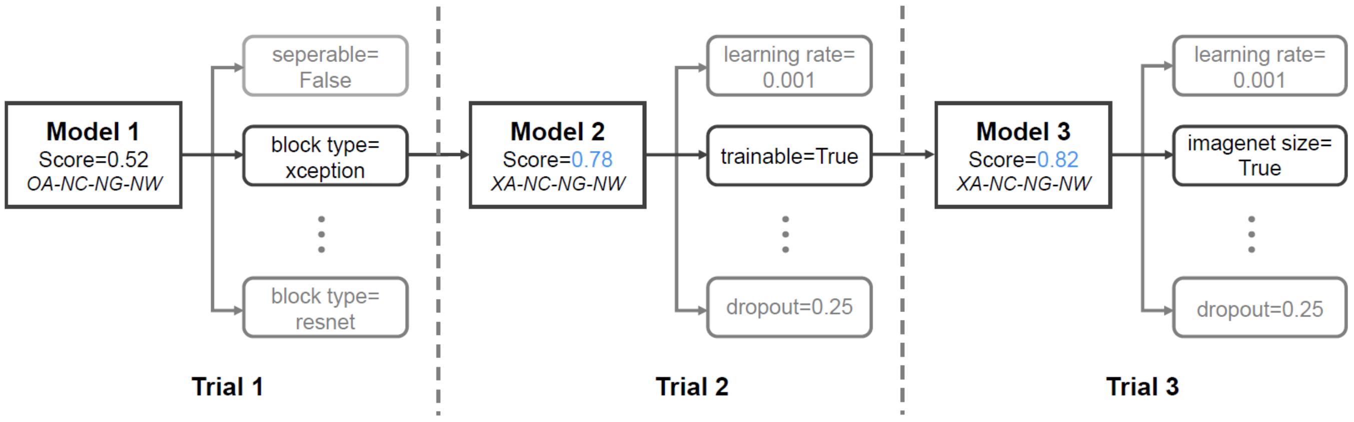
To solve this problem, we construct large-scale experiments to evaluate the search priority of each action under different conditions in the feedback-driven search. We randomly generate and train 200 different models on two datasets (i.e., CIFAR-10, CIFAR-100) and record the conditions summarized from the feedback of these models. We record the model accuracy in training as Acc_{ori}. Next, we search all feasible actions on each model one by one and generate new models to evaluate these actions. Each search action can be written as $O-V$, where O represents an object (e.g., model architecture, hyperparameter) this action will change, and V represents the newly assigned value of the object. Then we train these new models and record accuracy as Acc_{O-V}. All the training processes mentioned here include 15 epochs, and the batch size is 32. After these training processes are finished, we calculate the change of accuracy from the pair $O-V$, which is written as $\Delta Acc_{O-V} = Acc_{O-V}-Acc_{ori}$. Finally, we calculate the conditional probability $\mathbb{P}(Action | A, C, G, W)$. $$ \mathbb{P}(Action | A, C, G, W) = \frac{\sum_{i = 0}^{n} \Delta Acc^{i}_{O-V}} {n} $$

where n represents the amount of models under the same *A-C-G-W* condition. Larger $\mathbb{P}(Action | A, C, G, W)$ means that the action with the object O and value V are more prioritized in search and have more possibility to improve the current search performance. We implement these evaluated priorities as the default setting in the feedback-driven search to fix the AutoML pipeline and guide the pipeline to search effectively. In each trial, we choose the action with the highest probability (under observed A-C-G-W) to help generate new models. The whole priority table is shown here. In the table, the left side of the equal sign for each action is the object, that is, the hyperparameter or model architecture that will be changed, and the right side is the newly assigned value.

During the search, if the action with the highest priority has already been applied in the current model, the feedback-driven search will select the action with the next highest priority, and so on. Additionally, it is worth mentioning that we have met 17 sets of *A-C-G-W* conditions in the evaluation of 200 models, which is not all the potential conditions. For other conditions not triggered in experiments, we have manually analyzed and found that some conditions are too hard to coexist. For instance, when the EW condition is met, the NaN weights will cause the model to not update properly in the backpropagation. Therefore, the VG and EG conditions can't occur with EW at the same time. For such conditions that are not covered in our search priority, DREAM will choose random actions in search.

The Demo Case of Feedback-driven Search

Here is a demo case to illustrate how the feedback-driven search performs with the search priorities.



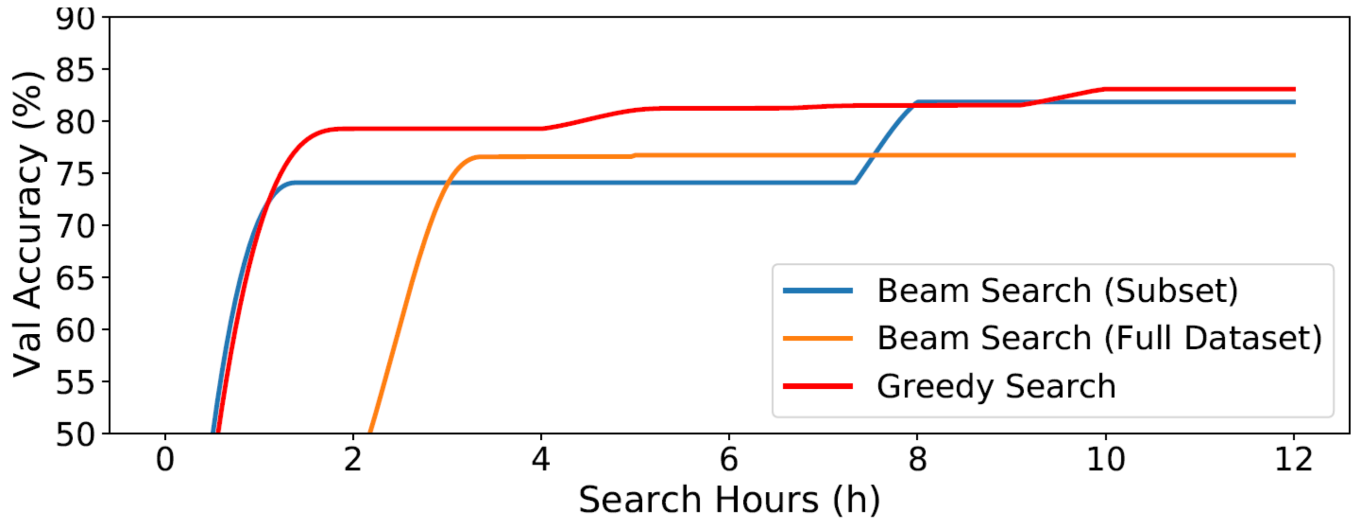
For the `Model 1` with an initial score of 0.52 and the conditions as `OA-NC-NG-NW`, as shown in the 14th column of the priority table, the action with the highest priority is `block type=xception`, which means that the object in this action is `block type` and the new value for it is `xception`. This action will change the model architecture to *XceptionNet*. After applying this action and generating the `Model 2`, the search score in training has increased to 0.78, and the conditions have turned to `XA-NC-NG-NW`, whose priority is listed in the last column of the priority table. The action with the highest priority in current conditions is `pretrained=True`. However, when building `Model 2`, the hyperparameter `pretrained` has already been set to `True`. Therefore, `pretrained=True` is skipped, and `trainable=True`, which has the next highest priority under the conditions, is selected as the action for this search. After applying this action, the score of `Model 3` increases to 0.83. At this time, the current conditions keep `XA-NC-NG-NW`, and the next action to be searched is `imagenet size=True`. The search process described above is highlighted in the above figure. The feedback-driven search in DREAM selects the actions one by one based on the conditions and the built-in priorities until the search is terminated.

Beam Search VS Greedy Search

The feedback-driven search in DREAM is an improved greedy search strategy. Specifically, when searching for a new model, it picks the best performing model in the search history and modifies this model by selecting actions that are most likely to improve the model through the feedback-driven search.

In fact, beam search, which is a common heuristic greedy search strategy, can be an alternative search method in DREAM. It builds the search tree with a breadth-first search over the search space and generates all successors of the states at each level of the tree and sorts them with their performance. The number of the stored states is equal to the width of the beam, which means that the larger the beam width, the fewer states to prune and the higher the search overhead. When the beam width is 1, the search process will be equivalent to the greedy search. With the above properties, beam search can explore the search space by expanding the most promising results in a limited set and is widely used in NLP and speech recognition models to choose the best output.

To determine the actual effect of beam search as an alternative search method, we construct a comparative experiment on the CIFAR-100 dataset. The experiment ensures that the search priority and search space of the two search strategies are consistent. In order to control the overhead of beam search, the beam width is set to 3. In addition, with the intention of improving the search efficiency of beam search and covering more possible models in search, we implement a two-step search. In the first step, we train nine searched models with the highest priorities, which are generated from the previous search, on a 10% subset of the CIFAR-100 dataset. Then in the second step, three models with the best performance in the first step are reserved for the complete training. In addition, in order to avoid beam search converging in local optimum, all hyperparameters in the models in searches have a chance of 0.01 to randomly mutate.



The above figure shows the comparison results of beam search and greedy search on the CIFAR-100 dataset, where the X-axis is the GPU hours spent in the search, and the Y-axis is the best validation accuracy the search reached. We implement two kinds of beam search in the experiments, namely a search trained in two steps with a 10% subset of the dataset as described above (i.e., `Beam Search (Subset)`), and a search trained directly on the full dataset without the subset (i.e., `Beam Search (Full Dataset)`). The experimental results demonstrate the superiority of greedy search in search efficiency and effectiveness. From this figure, greedy search improves accuracy faster than beam search using subsets. And beam search trained on the full dataset performs worst in these search methods. We analyze the specific process of the search manually and find that the main reason for the performance difference is that the beam search trains more models, and most of them help little in improving the search performance. This results in that the search efficiency of beam search is worse than that of greedy search in the case of the same search space and search priorities. Even for the search that most of the models are training on the subset of the CIFAR-100 dataset, the time cost of beam search to reach 80% accuracy is still nearly 6 hours higher than the time spent of greedy search due to the time overhead of training additional models.

Therefore, for the consideration of efficiency and effectiveness in search, we select the greedy method as the default strategy of the feedback-driven search. We believe that beam search has the potential to achieve better results when computing resources are sufficient and training can be deployed in parallel. It will be our future work to improve the effectiveness of the feedback-driven beam search.