Multi-Level Selection for OOD Ensembles

Scott T. Merrill
Department of Computer Science
University of North Carolina
smerrill@unc.edu

Hung-Tien Huang
Department of Computer Science
University of North Carolina
hungtien@unc.edu

Abstract

Traditional machine learning models are trained under the assumption that the data used for training the model will not change when the model is used for inference. However, such assumption might not hold during inference as distribution shift might occur. The Out-of-Distribution (OOD) generalization task aims to train trustworthy worthy models under any hypothetical distribution shift. Prior works tackle OOD generalization from different perspective, including but not limited to representation learning, self-supervised learning, and model ensembeling [Liu et al., 2021]. In this work, we apply genetic algorithms to construct ensembles where the the training data and features of the underlying models are jointly optimized. Given an ensemble produced by genetic algorithm, we further propose a post-hoc mixture of expert training procedure that further improve OOD generalization performance.

1 Introduction

Machine learning (ML) have shown outstanding performance in variety of applications. Yet, traditional ML models are trained under the assumption that the data used for training the model will not change when the model is used for inference. One approach to maintain robustness under distribution shifts is to ensemble a diverse collection of models. Given each individual model is reasonably accurate and produces independent errors with respect to the other models, mistakes may cancel out leading to an ensemble with good generalization ability Johansson et al. [2007]. There are many ways such diversity can be induced in ensembles. We consider creating this diversity by modifying the training data and features used by each individual model; this approach is similar to the way ensembles are created in many classical ML models such as XGBoost [Friedman, 2001] and Random Forests [Breiman, 2001]. We apply an Evolutionary Algorithm (EA) to perform feature and training data selection. The EA optimizes a novel fitness function that we demonstrate is a good proxy of OOD generalization. Finally, given an ensemble produced by a genetic algorithm, we propose a post-hoc mixture of expert training procedure that further improves OOD generalization performance. To the best of our knowledge, no evolutionary algorithm has been applied to select features for OOD tasks.

2 Related Work

2.1 Sparsely-Gated Mixture of Experts

In ensemble methods, the predictions made by each expert are often averaged. Similar to ensemble methods, mixture of experts (MoE or ME) model also maintain a collection of models. The difference is that MoE assumes each individual expert specializes in different "aspect" of the task we want to solve; hence, the output of individual experts should be mixed "differently" depending on the current instance instead of weighing expert opinions uniformly as in ensemble methods. Shazeer et al.

Preprint. Work in progress.

proposed a Sparsely-Gated MoE (SMoE) Layer that has a learnable gating function that generates a sparsely distributed mixing coefficient to encourage specialties Shazeer et al. [2017]. This designed sparsity allows massive expert models to be trained efficiently.

2.2 Feature Selection (FS)

FS is a dimension reduction technique where the goal is to identify an optimal subset of features for a specific task. Ladla and Deepa demonstrates how reducing redundant, irrelevant, or noisy features can improve classification efficiency and performance. Two categories of FS methods are filter method and wrapper methods [Ladla and Deepa, 2011]. Filter methods, typically exploit some correlation measure in the dependent variable to pre-process features for a classifier Yu and Liu [2003]. Wrapper methods consider the performance of the downstream classifier and feature subset together. Filter methods may be problematic in OOD problems as the correlations in the training dataset may not persist in the testing dataset due to possible distribution shifts. Wrapper methods may suffer similar issues in OOD settings if the classifiers' performance on in-distribution data isn't a good proxy for true OOD accuracy. However, wrapper methods can work well if the objective of the classifier correlates with true OOD performance. Our work attempts to identify a metric that aligns with true OOD accuracy to improve the of wrapper based FS selection.

Wrapper methods can be thought of as a search problem where the classifier tells how well a particular feature subset performs. The large search space, interaction effects among variables and latency in training a classifier make the search problem particularly difficult. To reduce the search space, greedy algorithms such as Sequential Forward Selection and Sequential Backward Elimination can be quite effective Ladla and Deepa [2011]. However, these greedy strategies are deterministic and are sensitive to local optima. In contrast, there's a class of gradient free evolutionary approaches which consider a population of solutions simultaneously and thus are less sensitive to local optima. In recent years various Genetic Algorithms (GAs), Genetic Programming (GPs), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) have been successfully applied to the problem of feature selection Xue et al. [2016].

To the best of our knowledge, no evolutionary algorithm has been applied to select features for OOD tasks. We speculate the reason for this may because the difficulty in coming up with a fitness function that acts as a proxy of true OOD accuracy. We attempt to combat this difficulty by evaluating model fitness based on out-of-bag (OOB) Error. We show how this OOB Error is, in certain problems, a good indication of true OOD accuracy and moreover how it can be used to evolved feature sets that perform well OOD.

3 Methods

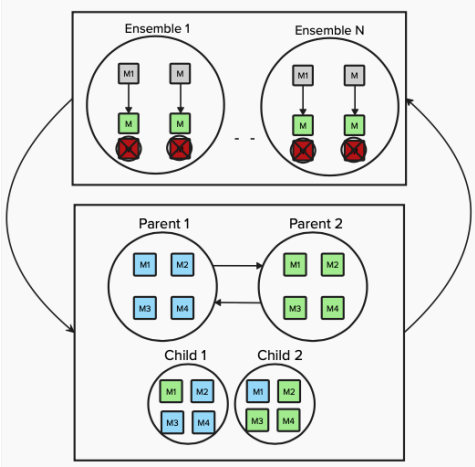
In this section we will explain our method to evolve ensembles of XGB Trees Chen and Guestrin [2016] for OOD prediction tasks. Evolution will identify optimal subsets of data and subset of features used to train each individual model. A custom fitness function which judges how well all the models work together in classifying OOD examples will be used to guide this optimization. We empirically validate that this fitness function is a good proxy of OOD accuracy.

3.1 Evolutionary Framework

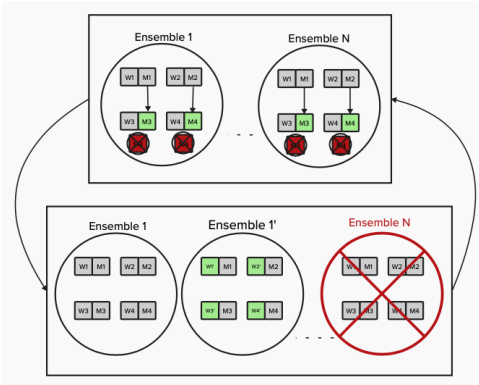
We next define our genetic encoding and the evolutionary operators that modify these encodings. Note that our encodings rely on training examples being first pre-processed into distinct clusters. This pre-processing step can be achieved, for example, through K-Means, Decorr Liao et al. [2022] or any hard-clustering algorithm. Figure 1 gives a pictorial description for our proposed framework.

Encoding: Each XGB Tree model m is represented fully by two vectors; a cluster vector $C = [c_1, ..., c_k]$ and a feature vector $F = [f_1, ..., f_d]$. Each are binary indicator vectors where $c_i = 1$ implies the m was trained on cluster i and $f_j = 1$ indicates the model has access to feature j.

Mutation: Mutations operate the same on both the cluster and feature vector. We define two mutation operators. The first selects n elements from one of these vectors as candidates for mutation. These n indices are each flipped independently with probability p. We also consider a simpler



(a) Illustration of EVE-OOD. Models are mutated within each ensemble based on fitness. Mutated models replace the least fit models. Top performing models then undergo crossover. Two children replace the worst performing ensembles.



(b) Illustration of WEE-OOD. Models are mutated within each ensemble based on fitness. Mutated models replace the least fit models. The weights of most fit ensembles are mutated and worst performing ensemble is replaced.

Figure 1: Illustration of our framework.

operation where all k clusters and all d features are considered candidates for mutations and flipped with probabilities p_k and p_d respectively.

We perform mutations over either clusters or features but never both simultaneously. The intuition behind this is it is easier to optimize one set of parameters at a time. Consider for example, a favorable mutation to the feature vector that is matched with an unfavorable mutation to the cluster vector. If the model's classification accuracy improves or degrades, it is impossible to isolate the source of improvement or degradation.

Crossover: We define two crossover methods which operate on ensembles of models. Each ensemble E contains a collection of N models each $[m_1, ..., m_n]$. The first crossover method selects a random integer i between 1 and n. Child 1 inherits a sample of i models from Parent 1 and a sample of the remaining $n - i$ models from Parent 2. Conversely, Child 2 inherits a sample of i models of Parent 2 and sample of the remaining $n - i$ models from Parent 1. We define a second crossover operation in which we select i random models in Parent 1 and swap them with i models in Parent 2. This operation thus requires an additional hyperparameter i representing the number of models to be swapped.

Fitness Function: We borrow an idea from training random forests and define the fitness function of our algorithm as the out-of-bag (OOB) error. The fitness of a model is defined by the average accuracy when predicting on clusters the model was not trained on. The fitness of an ensemble is a voting classifier of all models not trained on a particular cluster.

3.2 Evolutionary Voting Ensemble (EVE-OOD)

We show a visual representation of our first approach, which we refer to as EVE-OOD in in Figure 1a. We first initialize a population of M ensembles each containing N models. The encoding of each model is initialized to be trained on a random set of clusters using a random set of features. The random binary encodings for each are guided by a Gaussian distribution with their own parameters. For each ensemble we first compute the OOB fitness of its constituent models. We then identify the most fit models and perform a feature mutation with probability p_f and a cluster mutation with probability $1 - p_f$. The new mutated model then replaces the least fit model in the ensemble based on OOB fitness. After performing selection at the level of models, selection then occurs at the level of ensembles. We identify the best performing ensembles based on their OOB fitness. These then undergo a crossover creating new candidate ensembles that replace the worst performing members of

the population. We repeat this process for some fixed number of generations. We also considered several variants of this formulation which we describe briefly in the appendix.

3.3 Weighted Evolutionary Ensemble (WEE-OOD)

WEE-OOD attempts to improve on the voting-based approach of EVE-OOD by evolving a weighted ensemble. The pseudocode for WEE-OOD is shown in Algorithm 2 and a visual representation of this approach in Figure 1b. WEE-OOD begins by initializing N models and N weights. The model mutations are performed similarly to EVE-OOD. After these mutations, the OOB fitness of each weighted ensemble is calculated. We then identify the top ensembles, mutate only the weight vectors, and use these mutations to replace the worst performing ensembles. Thus, we have again defined a multi-level selection algorithm which first evolves good candidate models and proceeds to evolve weightings of those models that work well together.

EVE-OOD and WEE-OOD both will produce a population of M ensembles at the end of g generations of evolution. It's common in evolutionary algorithms to maintain the most fit individuals as the best solution. We consider this classical approach of taking the champion as well as an alternative approach of combining all models in the ensemble in the final generation. We try bagging all the ensembles predictions in the final generation as well as weighting each of their predictions by the ensemble fitness.

3.4 Post-hoc Mixture of Experts

Suppose we use the abovementioned algorithm to come up with an ensemble model of size N that is already doing a pretty good job at generalizing to out-of-distribution data. Weighing each expert in the ensemble uniformly is simple yet naive way of aggregating expert outputs. However, prior works suggest that ensemble method could benefit from learning an additional gating function $g(\cdot)$ that maps an input x to a weight vector $\langle w_i \rangle_{i=1}^{N}$ [Shazeer et al., 2017, Jacobs et al., 1991, Jordan and Jacobs, 1993]. The gating function is parameterized as a neural network and is optimized with the following objective function:

$$\mathrm{L}\left(g\left(x\right), \langle m_i\left(x\right)\rangle_{i=1}^{N}, y\right) = \alpha \mathrm{L}_{\text{in-bag}} + \beta \mathrm{L}_{\text{out-of-bag}} + \gamma \mathrm{L}_{\text{all}} \tag{1}$$

$$\mathrm{L}_{\text{in-bag}} = \mathrm{softmax}\left(\langle w_i \rangle_{x \text{ is in-bag w.r.t } m_i}\right) \langle \mathcal{H}\left(m_i\left(x\right), y\right)\rangle_{x \text{ is in-bag w.r.t } m_i} \tag{2}$$

$$\mathrm{L}_{\text{out-of-bag}} = \mathrm{softmax}\left(\langle w_i \rangle_{x \text{ is out-of-bag w.r.t } m_i}\right) \langle \mathcal{H}\left(m_i\left(x\right), y\right)\rangle_{x \text{ is out-of-bag w.r.t } m_i} \tag{3}$$

$$\mathrm{L}_{\text{all}} = \mathrm{softmax}\left(\langle w_i \rangle_{i=1}^{N}\right) \langle \mathcal{H}\left(m_i\left(x\right), y\right)\rangle_{i=1}^{N} \tag{4}$$

For a given instance x, we want the gating function to allocate most weight to the best performing expert within the in-bag experts, which is entailed by $L_{\text{in-bag}}$; meanwhile, instead of forbidding out-of-bag experts from contributing to the final inference, we want the gating function to allocate some weights to the potential good-performing out-of-bag experts, which is encouraged by $L_{\text{out-of-bag}}$; lastly, L_{all} attempts to encourage the gating function to weigh the contribution of all experts jointly to achieve better final inference.

4 Experiments and Results

4.1 Real-World Dataset

We consider adult [Becker and Kohavi, 1996] dataset from UCI machine learning repository. We arbitrarily choose one of the predictor variables and make in-distribution and out-of-distribution sets. Instances with attribute "workclass" equal to "federal-gov", "local-gov", "state-gov" are considered as out-of-distribution and the rest being in-distribution. The in-distribution dataset is split into training set, validation in-distribution set, and test in-distribution set following the 80/10/10 splitting ratio. The out-of-distribution dataset is split into validation out-of-distribution set and test out-of-distribution set with 50/50 ratio.

4.2 Fitness function Alignment Case Study

While there are several critical hyperparameters in our algorithm, we recognize the clustering algorithm and number of clusters as arguably the most important. These parameters directly influence

our fitness function. Since evolution will be optimizing this fitness objective, it's important for it to be representative of true OOD accuracy. We design an experiment to study the alignment of our proposed fitness function with the true OOD accuracy. We perform a grid search over the number of clusters to use and different clustering algorithms. Specifically, we use K-Means, hierarchical clustering, DBSCAN and Decorr Liao et al. [2022]. We then create 100 ensembles each randomly initialized with 25 models. Finally, we compute the OOB fitness of each ensemble and the true OOD accuracy of each ensemble. We define alignment as the r^2 obtained by regressing the OOB fitness onto true OOD accuracy.

We show a heatmap of these r^2 values for different parameterization in Figure 2a. We notice that there appears to be a number of clusters where alignment is maximized. And, importantly, we also notice that the relationship seems continuous; similar cluster bins produce similar alignment scores. Finally, we are excited to see that with 40 bins and using K-Means to bin our data, we can achieve an r^2 of 0.68. Given the alignments under certain parameterization, we believe this verifies OOB error as a logical metric to optimize for.

We design one more case study which to analyze the alignment of our fitness function. This study was inspired by analyzing the meta-data on many runs of our algorithm. We noticed an interesting relationship. Specifically we realized that the size of the cluster encoding vectors were quite strongly correlated with the fitness function alignment. To better understand this relationship, we perform a similar experiment to the previous one. However, instead of random initialization, each model is initialized to be trained on at most 2 clusters. We compare this to an initialization where models are trained on all but 2 clusters. We report the results in Figure 2b.

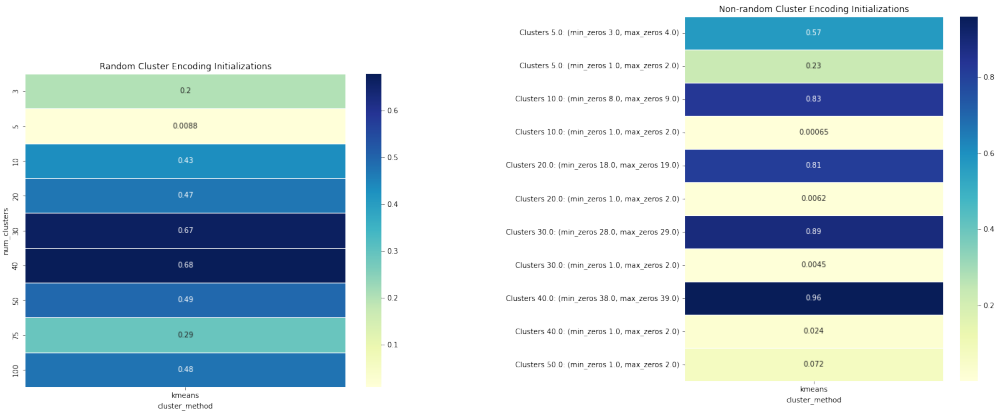
From this we see that training on fewer clusters seems to be really strongly related to alignment. Importantly, these alignment scores are less sensitive to the number of clusters as we originally observed. This is exciting from a practical standpoint and might signify robustness of this fitness function to these hyperparameters. The correlations seem so high it's almost too good to be true. At first, we believed the classifiers may be learning nothing and the fitness and OOD accuracies may just have both been around 50 percent. However, both fitness and OOD accuracies appear substantial and this isn't the case. We use these insights to guide our hyperparameter selection in our proceeding experiments. Specifically, we make two modifications to our algorithms. Instead of randomly initializing our cluster encodings according to a Gaussian distribution, we initialize them to be trained on at most 2 clusters; we hope evolution will slowly complexifying models. We also update our fitness function to add a term which encourages models to use fewer clusters and a term to encourage diversity. In our experiments we compare different parameterizations of the below fitness functions which generalizes the original fitness function.

$$\begin{aligned} \text{fitness} = &\ \text{Accuracy}_{\text{OOB}} \\ &+ \alpha\,(1 - \text{mean cosine distance between all other models in ensemble}) \\ &+ \beta\left(1 - \frac{\text{number of clusters model used}}{\text{total clusters}}\right) \end{aligned} \tag{5}$$

4.3 Analysis

We ran EVE-OOD and WEE-OOD for 30 generations on the adult dataset on clusters of sizes 10, 30 and 50 produced by K-Means. We considered ensemble sizes of sizes 15 and 25. We finally varied alpha and beta to be 0, 0.1, and 0.5 . We held all other parameters constant and save the careful understanding of these parameters as a key area of future work.

We show our results and compare them to three baselines. An XGB using all features, an ensemble of size 25 XGB models where features are selected randomly, and an EoA model. We also show our best models in our approach during the midterm report. A key distinction between these models we presented earlier in the semester is that they attempt to create diverse ensembles by optimizing a fitness function for quality diversity (QD) Pugh et al. [2016]; the hope is by optimizing QD models will create independent errors that will cancel out and the ensemble will be able to generalize well OOD. In this report, we instead try to optimize a function to approximate true OOD accuracy. We draw this comparison to emphasize the fact that approximating true OOD accuracy directly seems like a more promising approach than to optimize for QD. We speculate this may not be a unique quality of



(a) Gaussian Distributed Cluster Encodings

(b) Cluster Encodings have between min_zeros and at most max_zeros

Figure 2: Heatmap of fitness alignment.

EAs but rather apply to ensemble based OOD generalization approaches in general. We specifically see Table 1 that optimizing an approximate of OOD accuracy to provide great benefit. We also note that the best models for both EVE-OOD and WEE-OOD were found using 10 clusters. This is likely because with fewer clusters, the search space for which we optimize over is smaller. We speculate, with more generations of evolution, the performance of the models with more clusters could surpass the results we provide below. We lastly point out that we also explored making predictions based on all of the ensembles in the final generation rather than just the "champion" with the highest fitness. We studied both an un-weighted voting prediction of all ensembles and a fitness weighted scheme. We don't report the performance below, and only note that all accuracy metrics can be improved very marginally by performing this additional ensembling step.

Classifier	Train	Test-ID	Test-OOD
Baseline 1: Logistic Regression	83.32	82.64	77.79
Baseline 2: MoE	84.70	84.61	78.28
Baseline 3: EoA	85.65	85.55	80.66
Baseline 4: XGB	**89.52**	87.27	83.75
Baseline 5: Ensemble XGB	89.33	87.14	83.62
EVO(l, n, f)	86.44	85.57	79.84
CoEVO(l, n, f)	84.25	84.18	78.03
EVE-OOD	85.96	87.19	**84.00**
WEE-OOD	86.07	**87.33**	83.87

Table 1: Accuracy Comparison on Adult Dataset

To further understand our fitness function we plot both, the fitness alignment, mean number of clusters used and the true OOD accuracy. We show the results when 10 clusters are used. We see higher values of α and β do indeed seem to improve fitness alignments which is presumabley driven by these parameters encouraging training on fewer clusters. However, maintaining this better alignment doesn't seem to improve OOD performance. This is a strange result and perhaps indicates we need to perform more generations of evolution.

Another interesting result that merits investigation is the large initial drop in alignment in the early generations of evolution. We hypothesize this is a result of our mutation and crossover scheme. Specifically, greedily mutating the best models and using them to replace the worst performing models greatly reduces cluster diversity. After just a few generations, many of the models in each ensemble will contain models with similar cluster encodings. We demonstrate this claim in Figure 6a of the appendix. We demonstrate on a simple example with only 10 clusters. We plot the models in a single ensemble and show the number of models trained on cluster i; we track this distribution every 5 iterations. We notice the distribution begins relatively uniform which should be expected as this is how we initialized it. However, after just a few generations, the distribution becomes multi-modal.

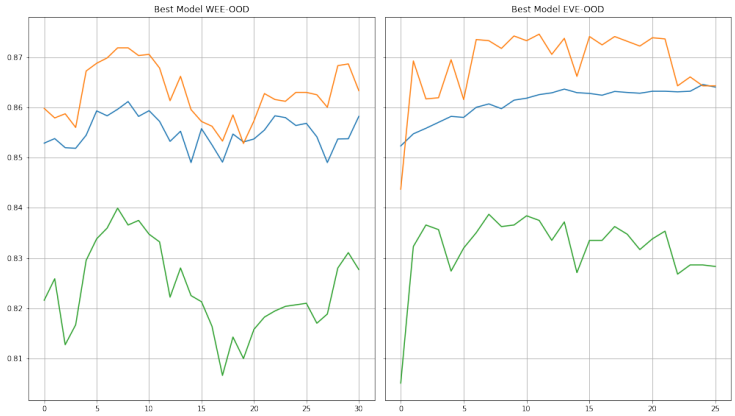


Figure 3: Best models found from experiments

As the best performing models are mutated to replace the worst performing once, cluster diversity vanishes. After 20 generations, the total number of clusters being used for training has more than doubled which perhaps explains the drop in fitness alignments. Interestingly we found that many ensembles converge to a very similar distribution after 20 generations regardless of initialization. We show another cluster distribution in Figure 6b to emphasize this point. This is likely a result of our crossover operations which encourages information sharing between ensembles. This analysis brings up an important direction of future work; how can we modify our mutation operation and model replacement scheme to prevent certain clusters to be used for training on nearly all models? We've tried randomly replacing models rather than performing our greedy mutation; while this slows the process of converging to high density clusters, it doesn't seem to prevent the behavior entirely.

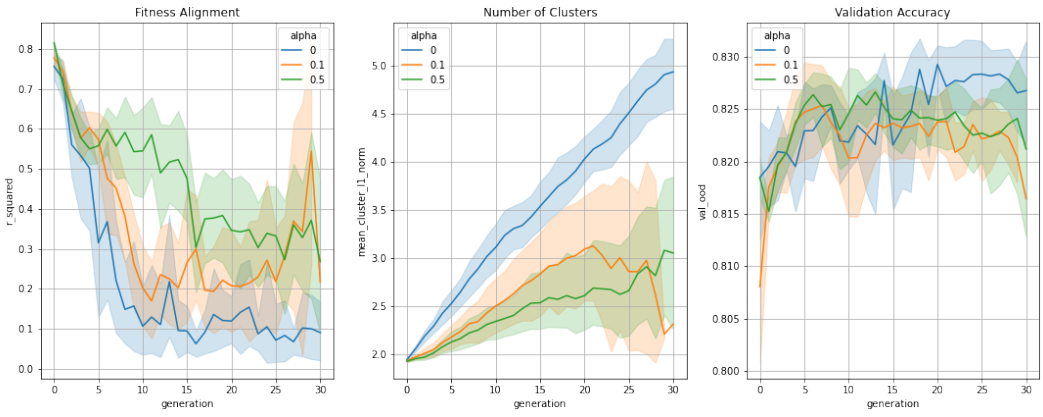


Figure 4: Alpha's impact on Fitness Alignment, Mean Cluster Encoding and OOD Accuracy

In Figure 8, we show the mean feature vector size of the models through several generations. The dataset we considered has 14 total features and our models seem to select more than half the features on average. It's reassuring to see that the more fit models indeed have larger feature subsets than worse performing models. We mentioned previously, that multiple ensemble populations converged to similar cluster distributions after 20 generations. We find that ensembles with similar cluster distributions after 30 generations also have similar feature distributions after 30 generations. We show an example of this in Figure 7a and Figure 7b. It's possible that this is again result of crossover operations. It's also possible that evolution is identifying the optimal feature subsets over the given similar cluster distributions, implying our methods are performing FS appropriately.

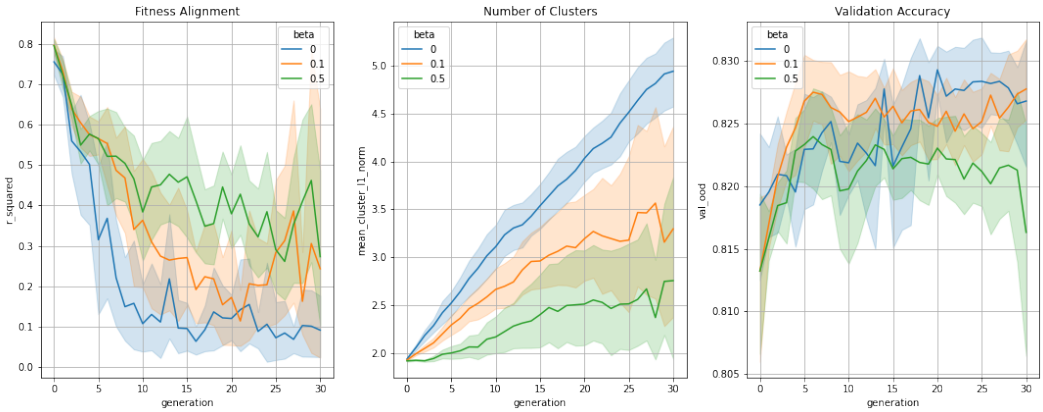


Figure 5: Beta's impact on Fitness Alignment, Mean Cluster Encoding and OOD Accuracy

4.4 Post-hoc Mixture of Experts

We report the performance of our proposed gating function optimized with Equation 1 on one of the ensemble produced by our method in Table 2. We attempted 4 combinations of α, β, and γ to experiment with the contributions of each terms in Equation 1. In Table 2, uniform refers to all model receive equal weights; vanilla refers to $\alpha = \beta = 0$ and $\gamma = 1$, ib-oob refers to $\alpha = \beta = 1$ and $\gamma = 0$, oob refers to $\alpha = \gamma = 0$ and $\beta = 1$, and ib-oob-va refers to $\alpha = \beta = \gamma = 1$. We can evidently determine that the ensemble benefits from having a gating function that allocates weights to each expert dynamically during inference. However, it seems that decomposing Equation 1 into $L_{\text{in-bag}}$ and $L_{\text{out-of-bag}}$ does not provide further boost in performance compare to that of optimized with plain L_{all}.

ensemble 1	val-id	val-ood	test-id	test-ood
vanilla	86.44	82.40	87.68	82.68
ib-oob	85.99	82.43	87.24	82.68
oob	85.32	81.58	86.19	81.00
ib-oob-vanilla	86.44	82.62	87.63	82.90
uniform	85.10	80.23	80.35	80.37
ensemble 2	val-id	val-ood	test-id	test-ood
vanilla	86.32	82.43	87.68	87.62
ib-oob	86.32	81.85	87.24	82.35
oob	85.76	82.46	86.19	82.35
ib-oob-vanilla	86.55	82.59	87.63	82.90
uniform	85.05	80.45	80.75	80.30

Table 2: Accuracy Comparison on Adult Dataset for Mixture of Experts Experiments

5 Conclusion and Future Work

While this research to still be incomplete, we believe our work and early results show signs of encouragement. In particular, the alignment of OOB fitness with true OOD accuracy is exciting and gives us hope of the feasibility of applying evolutionary algorithms for OOD generalization. Moreover, our method contributes to the field by providing a fitness function and framework for evolution algorithms to be applied to OOD problems. By comparing the performance we reported earlier this semester, we also contribute the idea that optimizing approximations OOD accuracy may be more beneficial than optimizing for QD Pugh et al. [2016] in constructing OOD ensembles. We hope our work encourages others to explore variants of OOB fitness. Another really exciting direction of future work would be to define a crossover operation on the individual model level. There has been work to incorporate crossover type operations to decision trees so incorporating a similar framework may enable a more directed and efficient optimization in the space of models Papagelis and Kalles [2001].

References

Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 34(2):550–570, 2021.

Ulf Johansson, Tuve Lofstrom, and Lars Niklasson. The importance of diversity in neural network ensembles - an empirical investigation. In *2007 International Joint Conference on Neural Networks*, pages 661–666, 2007. doi: 10.1109/IJCNN.2007.4371035.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL `https://doi.org/10.1214/aos/1013203451`.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/a:1010933404324.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=B1ckMDqlg`.

L. Ladla and Deepa. Feature selection methods and algorithms. *International Journal on Computer Science and Engineering*, 3, 2011.

Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. volume 2, pages 856–863, 01 2003.

Bing Xue, Mengjie Zhang, Will N. Browne, and Xin Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016. doi: 10.1109/TEVC.2015.2504420.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016. doi: 10.1145/2939672.2939785. URL `http://dx.doi.org/10.1145/2939672.2939785`.

Yufan Liao, Qi Wu, and Xing Yan. Decorr: Environment partitioning for invariant learning and ood generalization, 2022.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers Robotics AI*, 3:40, 2016. URL `https://api.semanticscholar.org/CorpusID:21713708`.

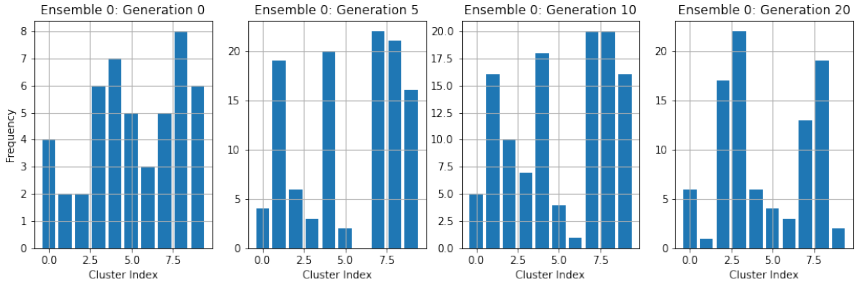
Athanasios Papagelis and Dimitris Kalles. Breeding decision trees using evolutionary techniques. pages 393–400, 01 2001.

A Model Variants

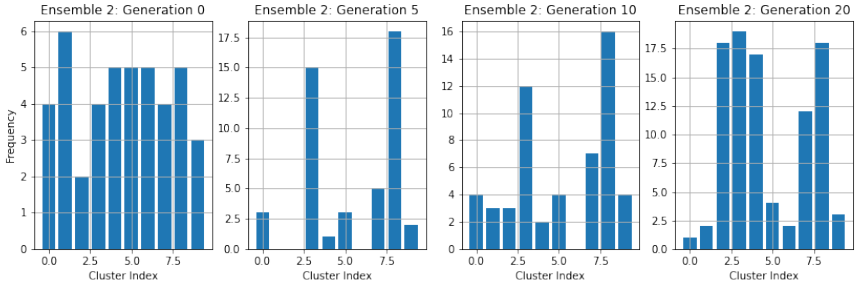
Variant 1 Rather than mutating only the feature or the clusters. A single model is mutated twice; once modifying only the features and once modifying only the clusters. Moreover, two new models are created; 1 in which the clusters are the same as the original model but the features are different, and 1 in which the features are the same as the original model but the clusters are mutated.

Variant 2 Rather than replacing the worst performing model, we also consider replacing a model at random. We found this helpful in maintain within ensemble diversity.

B Cluster and Feature Distributions of a Single Ensemble

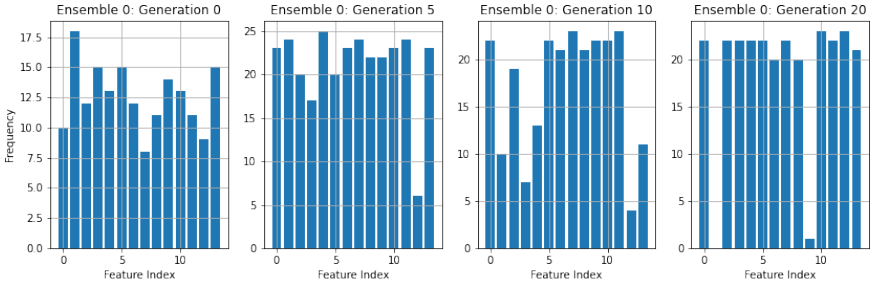


(a) Cluster Distribution of Ensemble Population 0

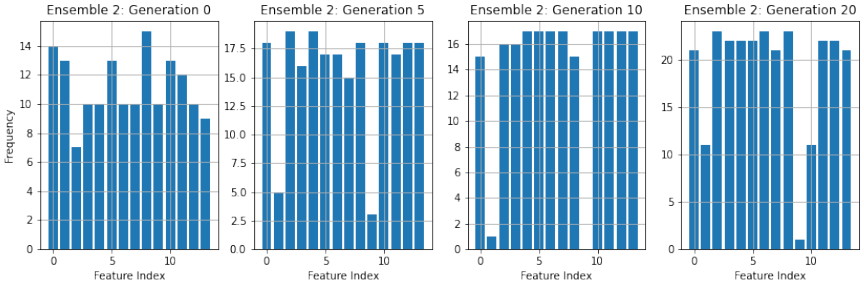


(b) Cluster Distribution of Ensemble Population 2

Figure 6: Cluster distributions.



(a) Feature Distribution of Ensemble Population 0



(b) Feature Distribution of Ensemble Population 2

Figure 7: Feature distributions.

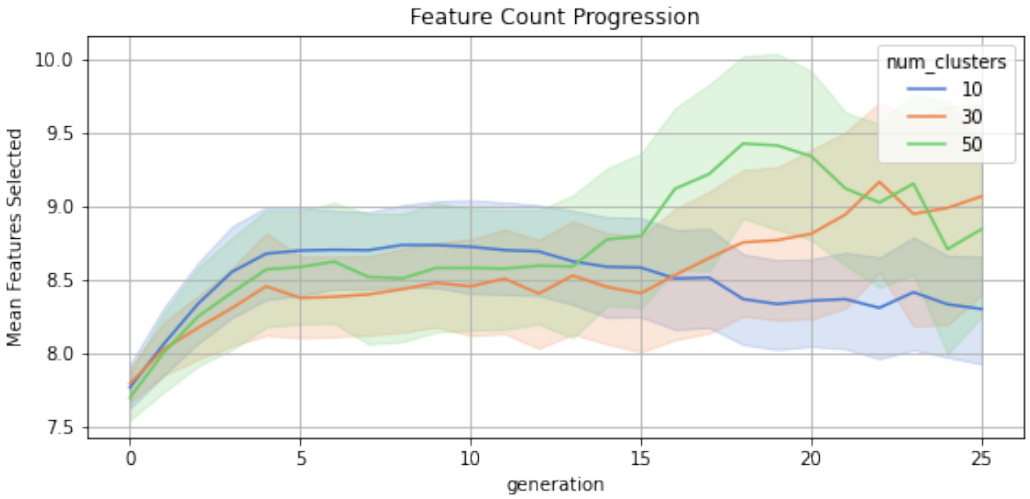


Figure 8: Feature Count Progression