Parity in Evolutionary Neural Architecture Search

Cameron Bennett
CS 8803 SMR
Georgia Institute of Technology
Atlanta, GA 30332
`cbennett49@gatech.edu`

Patrick Crawford
CS 8803 SMR
Georgia Institute of Technology
Atlanta, GA 30332
`pcrawford9@gatech.edu`

Rahul Bhethanabotla
CS 8803 SMR
Georgia Institute of Technology
Atlanta, GA 30332
`rmb8@gatech.edu`

Manav Agrawal
CS 8803 SMR
Georgia Institute of Technology
Atlanta, GA 30332
`magrawal62@gatech.edu`

Abstract

Neural architecture search (NAS) is an automated method used to design deep learning networks without the need for expert knowledge-based hand-designing of architectures, which can be both expensive and time-consuming. Evolutionary NAS is one optimization approach for neural architecture search that mimics natural selection and biological processes like selection to identify high-performing designs for subsequent iteration. There has been pre-existing research into NAS for natural language processing (NLP) problems like sentiment classification but no such research to our knowledge has considered the problem of social fairness and parity in the context of the models produced by the NAS search. We introduce metrics for social fairness and parity as objectives in our multi-objective search space for Evolutionary NAS to see if their inclusion can produce models that have equivalent performance for all subgroups along sensitive features. In addition to beating our BERT and LSTM baseline in categorical variance, our evaluation shows that the inclusion of these metrics produces models that have equivalent performance on these subgroups, increasing the categorical parity of our output models.

1 Introduction

Over the past decade, deep learning (DL) techniques have become extremely ubiquitous with products for every domain leveraging deep models and ML technologies for their functions. Present in products and platforms ranging from smart thermostats to self-driving cars to online shopping platforms, DL techniques are slowly becoming normal, common-place components in day-to-day life. Some companies and businesses have even begun to leverage DL models when interacting and managing human data. Unfortunately, the use of DL technologies in these domains with minimal consideration or regard for how these models might consume input data and the latent patterns in them has led to unfortunate and even downright appalling results. For example, in 2014, Amazon's recruiting AI learned to be strongly biased against women candidates and actively down-weighted samples when the word "woman" or "women" was present in a resume [Das18]. Unfairness and bias towards groups along identity factors such as gender, sexuality, or faith can be extremely problematic for a company or group using the model. Thus, concerns about models' integrity and fairness are usually managed and measured by specific teams of ML and domain experts. For example, Meta has many teams that manage the various models created to monitor speech or information on the platform. However, having multiple teams to monitor these types of models that directly interact with humans and, indeed, hiring the talent capable of designing these models optimally while considering these factors in the first place, can be an inordinately expensive task especially for companies with limited resources.

The advent of neural networks, through frameworks such as Keras and PyTorch, as methods of function approximation has widely expanded the solution space readily available to developers of all backgrounds. However, the majority of software developers are unable to immediately take advantage of these technologies due to the previously-mentioned expanding complexities associated with constructing, training and evaluating a network for a given task. Bridging the gap between developers lacking thorough backgrounds in machine learning is a task that has been taken up by select cloud providers such as Azure's LUIS (Language Understanding Intention System) [1] and Google Cloud Platform's Auto-ML Vision resource [2]. Though these resources can be useful, they ultimately put a cost on an otherwise automate-able task whereas it would be preferable to see an open-source alternative of some sort for researchers and entrepreneurs to engage with. Neural Architecture Search (NAS) is one variant of AutoML that searches a space of neural network architectures in order to find novel or optimal ones without necessarily requiring domain-expert hand-designing or intervention in the model development process. However, this lack of a need for domain expertise can lead to the unfairness and bias issues discussed previously.

Though research has been done to construct a multitude of Neural Architecture Search methods, a majority of these studies skew their domain of evaluation towards image classification (which precludes many of the situation where bias or fairness can play a role) and all of their implementations lack an easily accessible code-base for use by other researchers. While researching for this paper, it was discovered that there is a surprisingly minimal presence of Auto-ML frameworks available on GitHub and only a fraction of these frameworks have a hand in the process of automating architecture design in addition to hyper-parameters. The framework we found that was most comparable to our desired task was the Auto-Keras framework; however, this framework has been notorious for being slow, having a limited layer search space and not providing direct considerations for any parity metrics. This paper seeks to introduce a framework that enables researchers to perform a multi-objective evolutionary random search for the task of sentiment analysis with consideration to metrics of fairness. In particular, the researchers can apply this framework for the task of generating a sentiment analysis model with equivalent performance on varying metrics across various subcategories of the data-set.

Explicitly, the goal of our project is to investigate the following research hypothesis:

> *The explicit inclusion of "social fairness" metrics in NAS will lead to greater categorical parity in NAS-produced models without significant sacrifices to model F1 score performance overall.*

2 Background and Related Work

2.1 Types of fairness

Fairness in the context of NAS tends to relate the sampling of blocks and resultant children produced and the likelihood of cells being picked. Our work focuses on fairness from the perspective of social computing; that is, we focus on reducing discrepancies in performance across subcategories on what are called sensitive attributes or features. Common examples of such attributes are race, gender, sexuality, faith, or health-status. Sensitive attributes are rarely the output labels of models themselves but can play a role in learning patterns during model training. Due to the potential for serious discrimination when sensitive attributes are not properly considered, researchers have come up with and continue to develop fairness metrics and methodologies for various ML models and methods.

Prior work has focused on fairness metrics along two different avenues: group- and individual-centric metrics. The former works to establish that different sensitive attributes all have equitable performance and/or likelihood for some resultant label output while the latter focuses on ensuring two individuals, regardless of sensitive attribute differences, can achieve the same result if their

[1]https://learn.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis
[2]https://cloud.google.com/vision/automl/docs

other features are similar. We focus on the group-centric approach in our methods. Categorical parity, which determines whether the results of a model's classification are not dependent on a given sensitive attribute or category, is one example of a group-centric metric. Equalized odds, another metric, checks if, for any particular label and attribute/category, a classifier predicts that label equally well for all values of that category [HPS16].

Mainly model statisticians have developed a statistical non-discrimination criteria. Generally they view fairness through independence, separation and sufficiency. Independence is ensuring the classifier predicts labels regardless of sensitive attributes. This is often referred to as demographic parity, statistical parity, and group fairness. Second criterion is separation. This essentially wants to equate the true positive and negative rate for all the groups equally. The idea of equalizing error rates is called into critique and sometimes can lead to lower accuracy and models that perform badly. But the model becomes fairer across each of the categories. Third criterion is sufficiency or calibration. Equalized opportunity and equalized odds are common metrics which satisfies separation. Calibration wants correct predicted probability for all the groups. Essentially the categories must be explicitly encoded and the classifier must detect class membership otherwise it fails. Newer models such as predictive parity and darlington parity fall in calibration [BHN17].

2.2 Improving fairness

There are various ways to improve fairness. They happen in three stages, pre-processing, in-training, and post-processing. The first way consists of cleaning, sampling and feature blinding. This consists of cleaning the data or removing explicit categories as an input. But this is very insufficient as the categories and features are correlated with other features since we cannot assume independence in real world scenarios. It is also very data-specific and cannot be generalized. Another method is modification of objective function. This will change how our model is optimized and trained and instead for accuracy, is fine-tuned for fairness metrics. There are other ways such as adversarial classification and post processing optimization but this paper focuses on objective function trained for social fairness [Kle+18].

2.3 Existing Frameworks

Frameworks such as Fairlearn measure and mitigate risk of bias in certain groups and focuses on group, subgroup, and individual fairness. For mitigating fairness problems, this framework is limited to simple binary classification, regression, etc. Other frameworks designed such as Zafar et al. [Zaf+17] focus on convex margin-based classifiers like Logistic Regressions and SVMs. They added a disparate treatment criterion and mathematically established that the ratio of probability based on a particular feature should be greater than a threshold value. The work focused on SVM classification on numerical bank data. Agarwal et al. [Aga+18] utilized a combination of demographic parity and equalized odds with binary classification. Liu and Vicente [LV21] focused on stochastic multi-objective optimization exploring the tradeoff space between model accuracy and fairness metrics. The metrics used were equal opportunity and ran on the income data-set. Cruz et al. [Cru+21] combines accuracy and fairness into a convex function and it tries to optimize on that, but it also introduces new hyper-parameters as the function requires a prior.

Auto-tune by AutoML is a new framework for creation of fair and accurate models. This framework can use any user-defined search model to optimize over the demographic parity and accuracy tradeoff. This hybrid search strategy uses an array of methods such as, Latin Hypercube sample, Genetic Algorithm, Generating Set Search, etc. The purpose was to create models that can include different fairness mitigations hyper-parameters in this multi-objective. There are other optimization frameworks such as FairBO [Per+21] and FairHO [Cru+21]. FairAutoML [WW21] creates a pipeline that integrates fairness by optimizing AutoML hyper-parameters and testing it on Adult, Bank, and Compas dataset.

2.4 Evolutionary Search

Evolutionary Search (ES) is a popular continual learning technique that enables optimization of solutions in open-ended search spaces. In particular, ES draws upon natural selection as a means to simulate an environment that evaluates some array of proposed solutions (referred to as "individuals") on a set of performance metrics and then uses these evaluation scores to assign dominance of solutions within a tradeoff space and then objectively select the most performative individuals produced.

Evolutionary search can be done on single objectives but typically gains benefit over other search methods when done on a multi-objective space due to the increased accumulation of information that relates solutions to both objectives at the same time. ES has the benefit over other optimization techniques of being both continual and cumulative in its process of generating solutions, which is why it was explored for this particular paper. At a high level, ES typically consists of 4 stages: Generation, Mating/Mutation, Selection and Reproduction. Generation is the process of spawning some set of valid solutions, that can be evaluated on all of the proposed metrics. Mating/Mutation is the process of co-mingling individuals with some random chance in order to compound solution effectiveness. Selection is the combined process of evaluating and ranking individuals using any method of choice. Some common selection methods proposed for evolutionary search include Tournament Selection, NSGA(Non-dominated Sorting Genetic Algorithm) I or II, Stochastic Selection or Lexicase. The applicability of these metrics vary by context but as a whole, the space for ES has enabled creation of methods that can vary greatly in inclusivity. Finally, the process of reproduction sees ES regenerating the pool of individuals to be evaluated for a following iteration(referred to as "generation").

3 Framework: PEARs

In order to run our desired experiments, we had to construct our own framework, that we will be hosting as an open-source playground for NAS-specific AutoML. Unlike any other existing framework, we're granting the ability to test evolutionary NAS search on multi-objective spaces for NLP tasks. The framework constructed consists of two main components: the model generator and model selector. The model generator and individual construction is a modified version of a method used in the DEAP(Distributed Evolutionary Algorithms in Python)[For+12] framework. The DEAP framework also directly provided us with a method for selection in NSGAII [Deb+02]. The remainder of our check-pointing and scoring system was built using python's pickler and torch-metrics. Though it is common for mating and mutation methods to exist in evolutionary frameworks, this task seemed a bit much for this paper's focus and was therefore left as a future area of work to explore. The repository for the code-base is available at `https://github.gatech.edu/cs8803smr-f22/repo-team11`

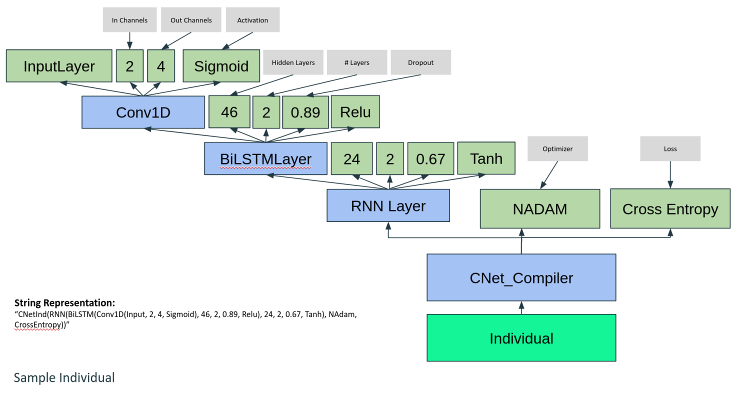
3.1 Model Generator

DEAP utilizes a tree-based method of encoding it's individuals. A handful of it's base classes specialize in wrapping primitive objects in such a way that enables them to be attached via linked list connections and dynamically compiled for evaluation. DEAP randomly generates individuals by maintaining a global primitive set object that is initialized with a desired input type, output type and a wide variety of functions and ephemerals that (when combined) comply with the desired rules to create a set of nested functions that produce a valid solution for some given input and output. For example, our implementation possesses a class for "Optimizers" and this class is the associated type of all optimizers (i.e. ADam, NADam, etc.) available for compile-able network individuals. The benefit of this structure is that it is easy to interpret and enables control over both network architecture as well as hyperparameters in our search space. An example individual can be seen below in Figure 1 (specifics regarding primitive set can be seen in code-base).

The primary area of focus with model generation involved ensuring valid connections between primitive layers such as GRUs and LSTMs (regardless of their hyper-parameters) in order to improve sample efficiency of this evolutionary algorithm. For this, we implemented a custom network generator that takes advantage of PyTorch's eager execution by passing input data shapes into randomly generated layers to ensure any appended network layers would maintain the validity of the generated network. Essentially, we provide the network generator with a set of typed functions and values along with a max network size and it will loop through a process of randomly generating a layer, testing that layer's ability to maintain a valid network and cautiously piece together an individual.

3.2 Model Selection

We decided to utilize NSGAII as our selection method for this implementation. NSGAII has had a long standing presence as the preferred method of multi-objective selection for most evolutionary algorithms research papers since 2002. In particular, this method of selection takes advantage of crowding distance sorting to maintain clusters of pareto optimal individuals with high performance in



String Representation:
"CNetInd(RNN(BiLSTM(Conv1D(Input, 2, 4, Sigmoid), 46, 2, 0.89, Relu), 24, 2, 0.67, Tanh), NAdam, CrossEntropy))"

Figure 1: An example individual produced by our model generator.

any one objective as well as all objectives. At a high level, this selection method consists of 2 steps (visible in Figure 2): Non-Dominated Sort and Crowd Distance Sorting. The Non-Dominated sort mechanism consists of computing a dominance count for the top k individuals of a given population for some k objectives. An individual is said to dominate another if it outperforms that individual on a specified metric. This process of non-dominated sorting generates k frontiers of individuals which then all have their crowd distances computed. Crowd distance is the distance from one point in the pareto front to it's nearest 2 neighbors(assuming a 2 dimensional front). In the event that 2 individuals both appear on the pareto front and do not dominate one another, the one with a greater crowd distance is prioritized. This two step selection mechanism has shown an effectiveness at maintaining a solid spread of pareto optimal individuals with respect to both metrics in a multi-objective space. Our implementation utilizes DEAPs implementation of this algorithm.

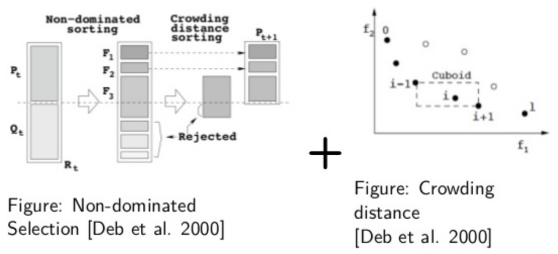

Non-dominated sorting
Crowding distance sorting
P_t
Q_t
R_t
F_1
F_2
F_3
P_{t+1}
Rejected
Figure: Non-dominated Selection [Deb et al. 2000]
+
f_2
0
i-1
Cuboid
i
i+1
l
f_1
Figure: Crowding distance [Deb et al. 2000]

Figure 2: The two steps of the NSGAII Selection Algorithm

4 Fairness Metrics

A fitness function in an evolutionary NAS algorithm suggests how well the model is doing. A single-objective NAS for a classification task might focus on accuracy. The aim for this paper is to make this into a multi-objective problem. The primary objective will be overall model accuracy but Table 1 shows the different secondary objective metrics and what it entails.

Table 1: Objective Metrics for Evaluation

Name of Experiment	Secondary Objective
Baseline	Maximize overall F1
Accuracy-Mean	Minimize mean difference in per-category accuracies
Accuracy-Variance	Minimize variance in per-category accuracies
F1-Mean	Minimize mean difference in per-category F1 scores
F1-Variance	Minimize variance in per-category F1 scores

Our falsifiable hypothesis is that the explicit inclusion of "social fairness" metrics in NAS will lead to greater categorical parity in NAS-produced models without significant sacrifices to model accuracy score performance overall. 3 different baselines of comparison were used based on appropriateness to our use case as well as our NAS trials. We've decided to use the variance in categorical accuracies and range of categorical accuracy's as our metrics for validating our hypothesis. This is because our definition of parity values distributions that are closer to uniform with a variance of 0. We ran 2 trials of NAS using PEARs which consisted of 25 generations of selection over a population size of 25 individuals with a pareto front of size 10. Additional configurations for our NAS trials are laid out in Appendix A. Non-NAS baselines shared the same training loop configurations and data loading pipelines as those provided within NAS trials.

5 Experimental Setup and Results

To demonstrate effective consideration for parity and fairness metrics, the following experiment has been provided to measure the effectiveness of custom parity metrics on the search space.

5.1 Data Used

The data-set utilized for this experimentation was sampled from the amazon product data-set [3]. In particular, a data-set consisting of summary product reviews was generated for topics: Office Products, Beauty, CDs and Vinyls, Sports and Outdoors, Baby, and Amazon Instant Video. This data-set consists of labels 0 and 1 corresponding to a 1-star review and 5-star review respectively. The split of these labels was 50:50 such that there were 25K rows of data for each target category. The topics selected were also used as the category of concern and the distribution of this category dimension is shown in Table 2 below.

Table 2: Distribution of categories

Name	Count
CDs and Vinyl	29984
Sports and Outdoors	7189
Beauty	6029
Baby	4677
Amazon Instant Video	1085
Office Products	1036

The reason for maintaining this imbalance of categories was to provide some non-label-based bias in training with the hopes of identifying bias within models produced and selecting those less impacted by it.

5.2 Baseline Results

The first baseline considered based on the use case is a model consisting of 2 LSTM layers followed by 2 linear layers and a classification head. This seemed the lightweight and intuitive solution for our particular sequential task. The chosen hyper-parameters of this network are included in the Appendix. After training this model (see Figure 3), we measured an overall accuracy of 85.39%, a categorical variance of 16.045, a categorical range of 8.02% and the categorical accuracy distribution in figure 3.

The next baseline was a more complex model that we also figured might be applicable to this task at the expense of more resource-intensive training. Our second baseline is a pretrained Distil-BERT model from PyTorch. We simply attached a classification head for our task. We trained for the same number of epochs with the same learning rate as the prior baseline which reaped an overall accuracy of 88.01%, a categorical variance of 20.22, and a categorical range of 11.74% and the categorical accuracy distributions in Figure 4.

[3]https://jmcauley.ucsd.edu/data/amazon/

Finally, we thought it appropriate to include one NAS baseline with a previously applied metric relating to precision and recall. We decided to go with F1 Score as the objective for measuring our NAS baseline's performance, and the variance and range of categorical accuracy. After running 2 trials with these objectives in our NAS, we averaged the categorical accuracies of our last generation of pareto optimal individuals which produced a categorical variance of 16.53 and range of 10.52% and a distribution of the following categorical accuracies.

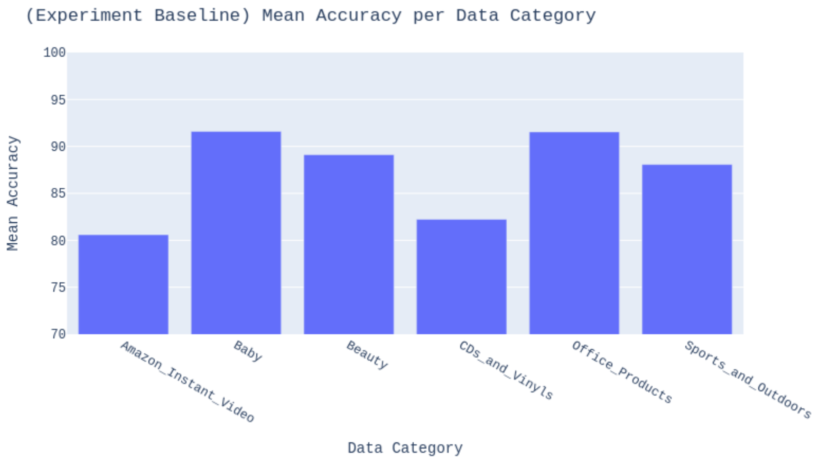


Figure 3: LSTM Baseline Categorical Accuracy Distribution

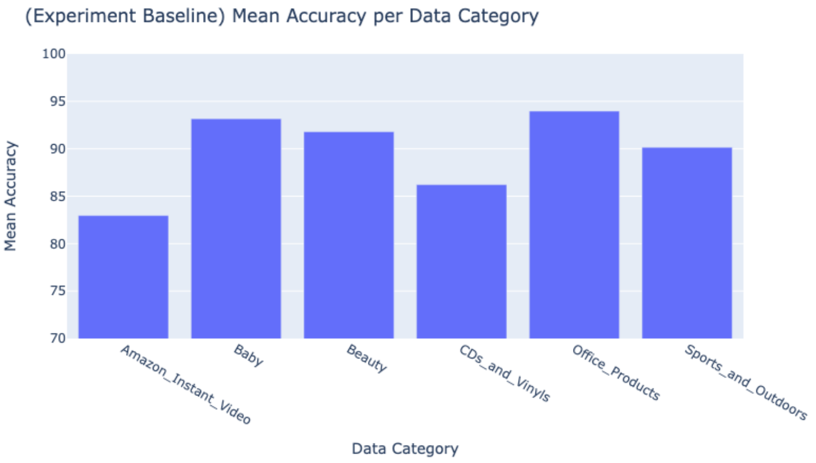


Figure 4: DistilBERT Baseline Categorical Accuracy Distribution

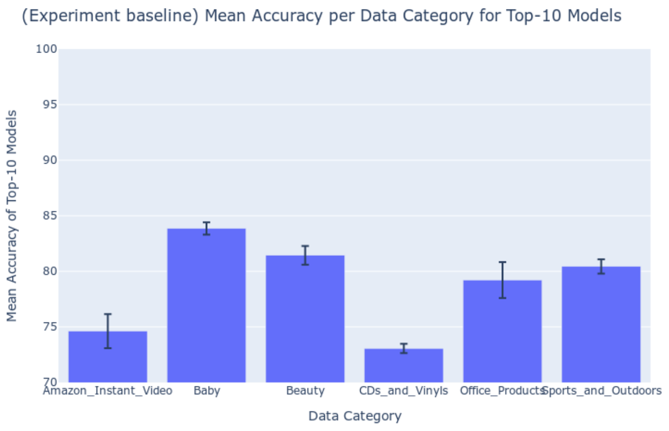


Figure 5: NAS with Accuracy and F1 objectives

5.3 Experimental Results

After determining the baselines, we ran NAS with various secondary objectives as discussed earlier. Around 300 models were evaluated per trial and it should remain consistent with the experiments of this scale and larger. Firstly, it is important to evaluate the pareto frontier of category accuracy and variance. In Appendix C, we can see the pareto fronts and the first graph shows the models generated by NAS throughout all the generations. It can be observed that there is a slight curve and there are various models with low variance but low accuracy or considerable variance with better accuracy, but not a big trade-off.

Table 3: Baselines and Experiments Accuracy, F1, and Category Variance Table

Name of Experiment	Type	Category Variance	Category Range	Model Accuracy (%)	Model F1
DistilBert	Transformer: Baseline	20.22	12.26	**88.01**	**88.01**
LSTM	LSTM: Baseline	16.05	**8.61**	85.39	85.47
Basic F1	NAS: Baseline	16.53	10.52	77.10	77.06
Accuracy-mean	**NAS: Experiment**	**14.15**	10.08	76.40	76.30
Accuracy-variance	NAS: Experiment	17.17	10.98	76.70	76.70
F1-mean	NAS: Experiment	16.43	10.65	76.51	76.50
F1-variance	NAS: Experiment	17.04	10.92	76.23	76.21

As seen in Table 3, a graphical representation of categorical parity variables can be created to show how our baselines perform when compared to the metrics introduced in NAS.

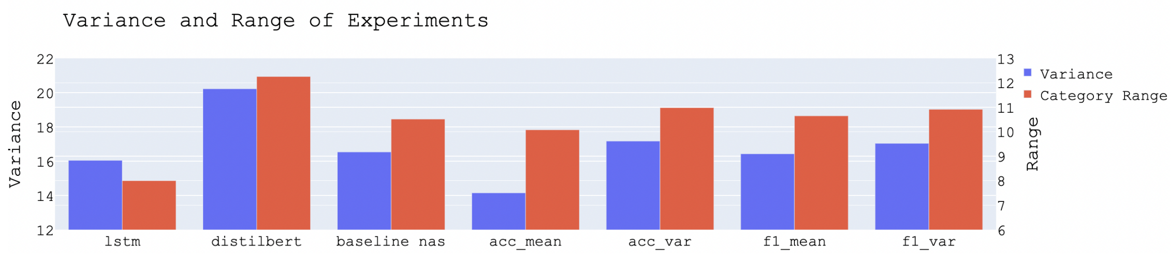


Figure 6: Baselines and Experiments against Categorical Variance and Range

From Figure 4 and 5, the secondary objective of minimizing difference in per-category accuracies (acc-mean) yielded the lowest variance and one of the lower categorical ranges. Since variance is more adept at estimating the spread of datapoints from mean, it can be said that incorporating this improves parity. When compared to baseline of NAS, there is a 0.7% model accuracy drop with an improvement of 2 in variance and 0.5%. This is approximately 14.4% drop in variance.

However when compared to LSTM models and a transformer architecture, model accuracies are inferior. However, these NAS models were not fully trained and were run for 25 generations. If the best performing model from acc-mean be taken and trained for additional five epochs (Appendix B), these results are measured in Figure 6.

Table 4: Baselines and Experiments Accuracy, F1, and Category Variance Table

Name of Experiment	Type	Category Variance	Category Range	Model Accuracy (%)	Model F1
DistilBert	Transformer: Baseline	20.22	12.26	**88.01**	**88.01**
LSTM	LSTM: Baseline	16.05	8.61	85.39	85.47
Trained Acc-Mean	**NAS: Experiment**	**7.26**	**7.05**	83.03	83.06

Table 4 shows that if the model is fully trained, the accuracies approach LSTM models' accuracies. However, the variance of category accuracy is extremely low. This is almost a 54% decrease in variance with only with 2.8% decrease in model accuracy. This validates our hypothesis that using an explicit parity objective improves categorical accuracy. Now BERT model was included as a baseline because transformer architectures are more appropriate for sentiment analysis related tasks. However, there is high variance but as it can be seen in Figure 6, the trained acc-mean does not significantly compromise accuracy.

Some additional things we noticed were that the pareto-fronts produced in all experiments generally conveyed that were was not a big tradeoff between parity and accuracy. We speculate this has to do with the small range of convergence seen in our experiments. Future work will seek to expand the numeric range of our metrics and see how that affects the pareto curves produced. Additionally, the difference in category accuracies across experiments could only be seen in measurements for standard deviation because most category averages were very similar in value. We hope expanding the scale of our experiment might make disparities in categorical accuracies more obvious between trials but for sake of this paper, our measurement of categorical variance and range is satisfactory for identifying relationships between experiments. Finally, we noticed through some trial and error that the reasons for overall accuracy disparity between our baseline LSTM and the models produced from NAS has to do with the fact that our NAS models are on average more complex than a simple LSTM. Usually this would be beneficial given a more complex task however for the task of sentiment analysis, our NAS models were not able to converge as quickly as the baseline LSTM due to the added number of parameters and layers being trained. However we did demonstrate that, given some fine-tuning our model could be comparable in accuracy to both of our baselines but with a lower variance and range of categorical accuracy.

6 Conclusion

The completion of this work provides some affirmations around the importance of accounting for fairness in both deep learning and Auto-ML. In particular, this paper targets the vacant intersection of study regarding Neural Architecture Search, NLP and fairness. We hypothesized that the inclusion of **some** explicit fairness proxy metrics in the search space of neural networks would help in identifying networks that can more effectively perform across subsets of data not directly linked to the labels being trained on. Our experiment demonstrated that this was in fact the case. However, there is still more room to experiment with these metrics. In setting up and evaluating experiments, the range of scoring for our particular custom metrics was rather narrow(see Appendix C). For future work, we'd explore ways of expanding the range of scores such that trade-offs between models are more spread out and impactful. Additionally, we'd like to experiment with weighted selection methods to see how to effectively spread out the trade-off space of dominant individuals. Ultimately, however, our contributions serve as a first step towards accounting for fairness in larger systems of AutoML.

7 Contributions

- Patrick (RB, MA, CB): implemented custom per-category metrics functions, implemented NAS model checkpointing and logging, contributed to evolutionary framework development (e.g. helping fix bugs, adding some features, integrating checkpointing), helped with baseline model training/analysis/visuals as well as best NAS-produced model fine-tuning/analysis/visuals
- Cameron (RB, MA, PAC): Coordinated team meetings and presentation rehersals, pitched initial idea behind paper, pushed initial implementation of the genetic search used in repository(as well as implemented majority of primitive layer methods used), created custom generator function being used to make neural networks, created compiler function for training, evaluating and scoring individuals, engineered residual networks for framework use, hosted majority of NAS runs for both data collection and debugging, wrote initial draft of paper, assisted in generating quantitative and qualitative plots(majority Appendix).
- Manav (RB, CB, PAC): Baseline model training/analysis/visuals, worked on improving the balance of datasets and other EALib bugs, resposible for setting up trains on clusters and fixed environment problems, helped with data analysis of experiments (generating category variance, range)
- Rahul (MA, CB, PAC): worked on setting up primitives for the NAS search space, worked on training dataset construction/collection/modifications, worked on NAS checkpointing/logging, contributed to evolutionary framework development (e.g. helping fix bugs, generally adding some features), helped with baseline model and NAS training/analysis, identified evaluation/logging bugs, wrote initial/final draft of paper, worked on and voiced over script for demo
- **Note** (MA, CB, PAC, RB): Over the course of this semester, all members of our team all contributed to the codebase considerably and meaningfully. However, this is not represented well in our commit history. This is because a lot of our coding was done in a pair-programming (or rather quad-programming) style on VS Code Live Share where all 4 of us would sit in a Discord/Zoom call and work through code and debug together. However, only one person would commit this code. As a result, the commit history does not accurately reflect on the work that both Manav and I did as both Patrick and Cameron would often make these official and final commits to the repo. We wanted to reach out to you to make sure this information was made clear, so that members of the team aren't penalized due to a seeming lack of contribution to the codebase. This has been communicated over the email.

References

[Deb+02] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: `10.1109/4235.996017`.

[For+12] Félix-Antoine Fortin et al. "DEAP: Evolutionary Algorithms Made Easy". In: *Journal of Machine Learning Research* 13 (July 2012), pp. 2171–2175.

[HPS16] Moritz Hardt, Eric Price, and Nati Srebro. "Equality of opportunity in supervised learning". In: *Advances in neural information processing systems* 29 (2016).

[BHN17] Solon Barocas, Moritz Hardt, and Arvind Narayanan. "Fairness in machine learning". In: *Nips tutorial* 1 (2017), p. 2.

[Zaf+17] Muhammad Bilal Zafar et al. "Fairness constraints: Mechanisms for fair classification". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 962–970.

[Aga+18] Alekh Agarwal et al. "A reductions approach to fair classification". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 60–69.

[Das18] Jeffrey Dastin. "Amazon scraps secret AI recruiting tool that showed bias against women." In: *Reuters* (Oct. 10, 2018). URL: `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G10` (visited on 12/03/2022).

[Kle+18] Jon Kleinberg et al. "Algorithmic fairness". In: *Aea papers and proceedings*. Vol. 108. 2018, pp. 22–27.

[Cru+21] André F Cruz et al. “Promoting fairness through hyperparameter optimization”. In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 1036–1041.

[LV21] Suyun Liu and Luis Nunes Vicente. “The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning”. In: *Annals of Operations Research* (2021), pp. 1–30.

[Per+21] Valerio Perrone et al. “Fair bayesian optimization”. In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 2021, pp. 854–863.

[WW21] Qingyun Wu and Chi Wang. “Fair AutoML”. In: *arXiv preprint arXiv:2111.06495* (2021).

8 Appendix A: Hyper-parameters

In order to test the effectiveness of our metrics, we ran 2 trials of a search for 25 generations with the following parameters for Evolutionary Search.

Table 5: Evolutionary Search Parameters

Max Tree Size	5
Population Size	25
Pareto Pool	10

Table 6: Training Loop Hyper-parameters

Tokenizer	DistilBert
Learning Rate	1E-3
Batch Size	6
Epochs	15
Embedding Dimension	300
Criterion	Cross Entropy Loss
Max Sequence Size	128

Figure 7: These hyper-parameters were shared amongst NAS and non-NAS individuals

Table 7: Primitive Layers Used

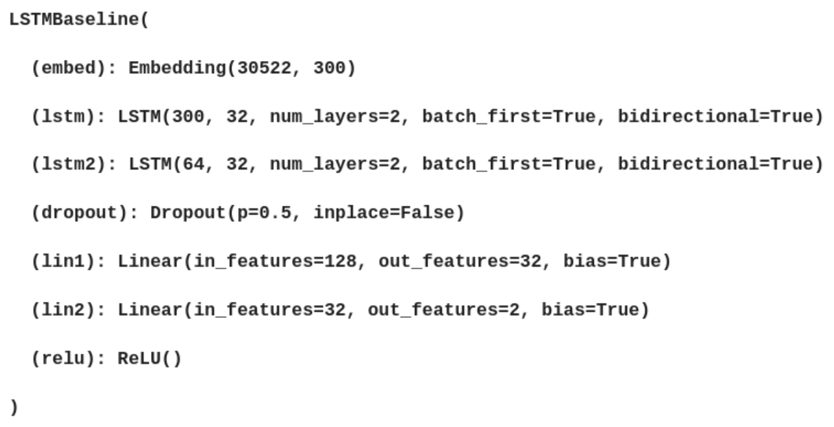
BiLSTM
LSTM
RNN
GRU
TransformerEncoder
LinearLayer
Conv1D
Residual Connection

Table 8: Terminals and Range of values

Optimizer	["ADAM", "NADAM", "RADAM"]
Size of Residual Connection	[1,3](layers)
Number of Recurrent Layers	[1,3]
Hidden Layer Size	[10,50]
Dropout Value	[0-1]
Channel Size	[1,5](Conv1D Layers)
Kernel Size	[1,5](Conv1D Layers)
Activation Functions	[Relu, Sigmoid, Tanh]

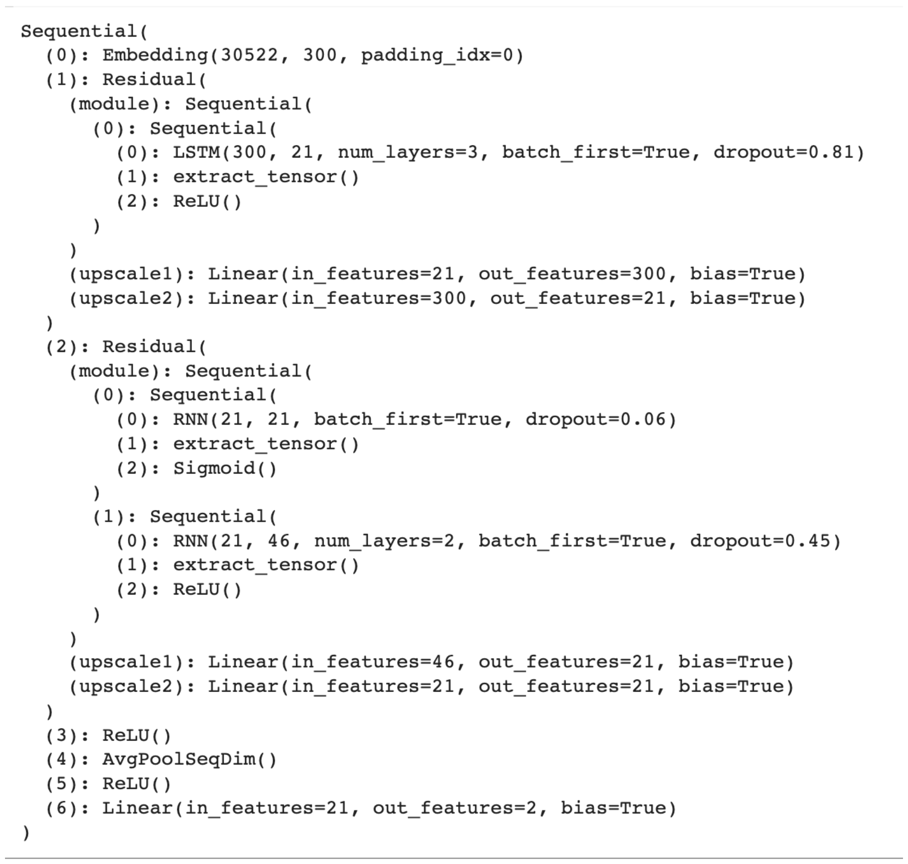
9 Appendix B: Mentioned Architectures

9.1 LSTM Architecture



```
LSTMBaseline(

  (embed): Embedding(30522, 300)

  (lstm): LSTM(300, 32, num_layers=2, batch_first=True, bidirectional=True)

  (lstm2): LSTM(64, 32, num_layers=2, batch_first=True, bidirectional=True)

  (dropout): Dropout(p=0.5, inplace=False)

  (lin1): Linear(in_features=128, out_features=32, bias=True)

  (lin2): Linear(in_features=32, out_features=2, bias=True)

  (relu): ReLU()

)
```

9.2 Fine-tune Best NAS-Produced Model



```
Sequential(
  (0): Embedding(30522, 300, padding_idx=0)
  (1): Residual(
    (module): Sequential(
      (0): Sequential(
        (0): LSTM(300, 21, num_layers=3, batch_first=True, dropout=0.81)
        (1): extract_tensor()
        (2): ReLU()
      )
    )
    (upscale1): Linear(in_features=21, out_features=300, bias=True)
    (upscale2): Linear(in_features=300, out_features=21, bias=True)
  )
  (2): Residual(
    (module): Sequential(
      (0): Sequential(
        (0): RNN(21, 21, batch_first=True, dropout=0.06)
        (1): extract_tensor()
        (2): Sigmoid()
      )
      (1): Sequential(
        (0): RNN(21, 46, num_layers=2, batch_first=True, dropout=0.45)
        (1): extract_tensor()
        (2): ReLU()
      )
    )
    (upscale1): Linear(in_features=46, out_features=21, bias=True)
    (upscale2): Linear(in_features=21, out_features=21, bias=True)
  )
  (3): ReLU()
  (4): AvgPoolSeqDim()
  (5): ReLU()
  (6): Linear(in_features=21, out_features=2, bias=True)
)
```

10 Appendix C: Qualitative Plots

10.1 Pareto Fronts

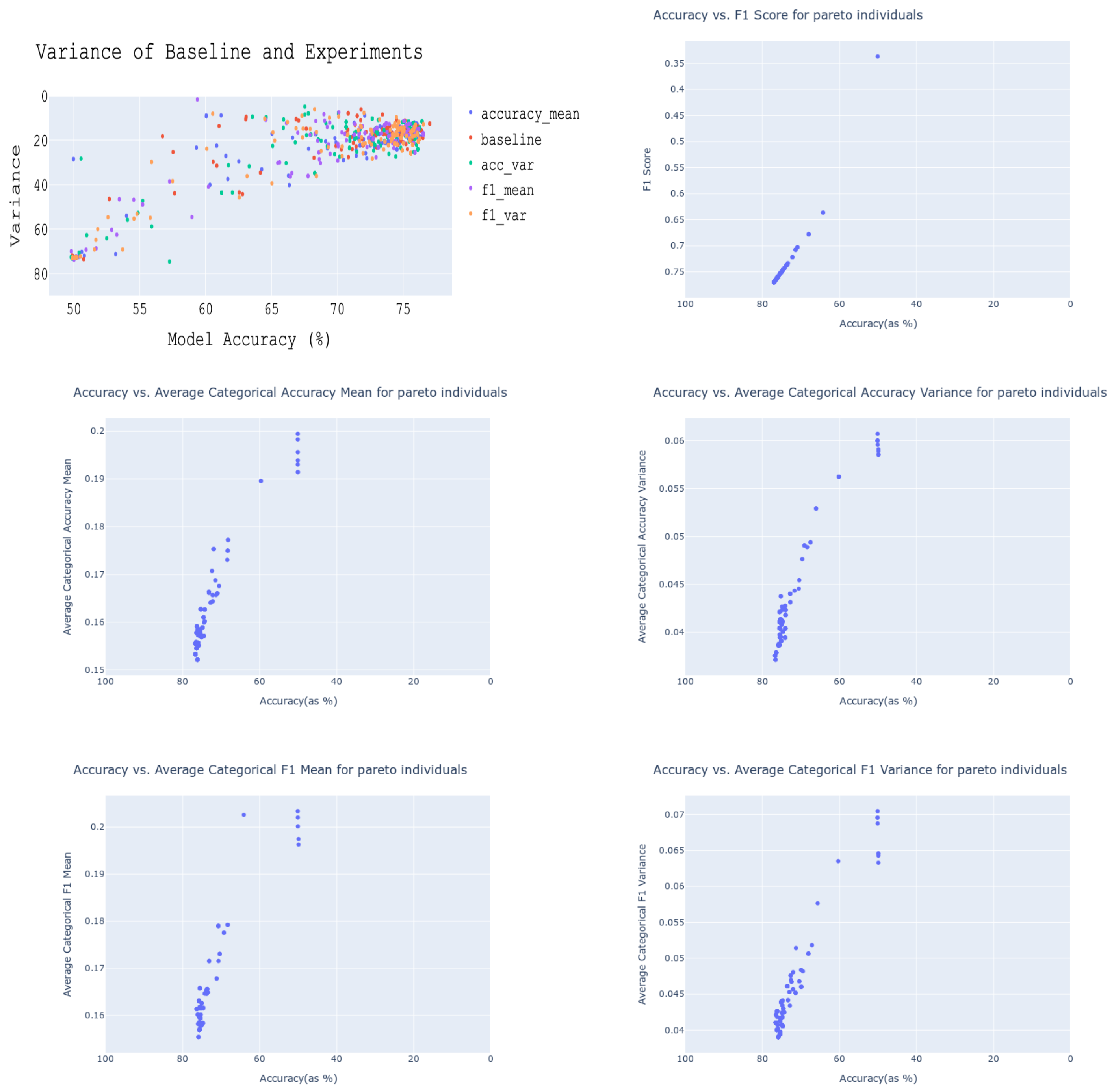


Table 9: Noticed little tradeoff between the introduced metrics and accuracy

10.2 Distribution of Categorical Accuracies

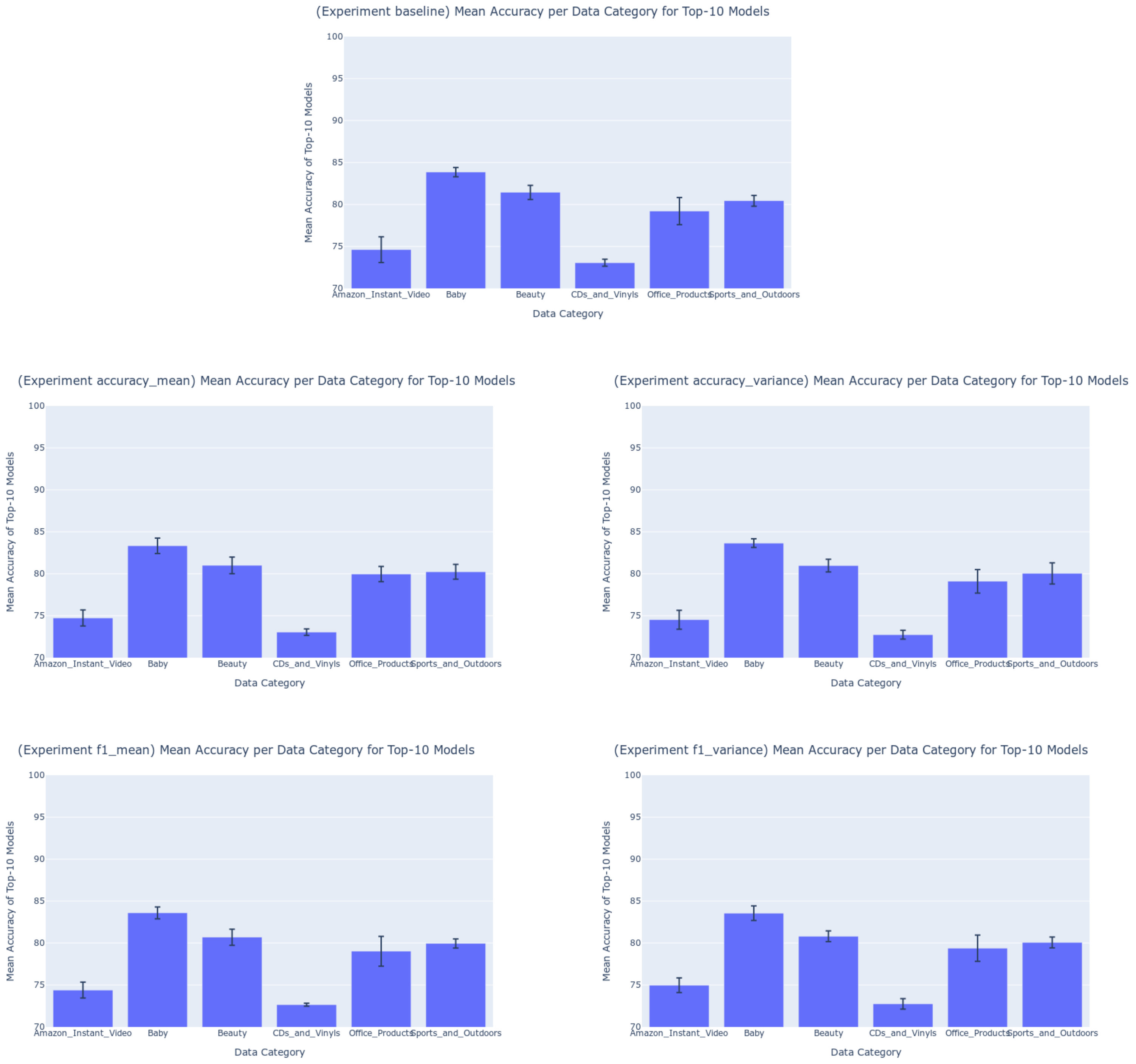


Table 10: Categorical accuracies were shared amongst all search spaces only varying on standard deviations per search space

10.3 Metric Convergences

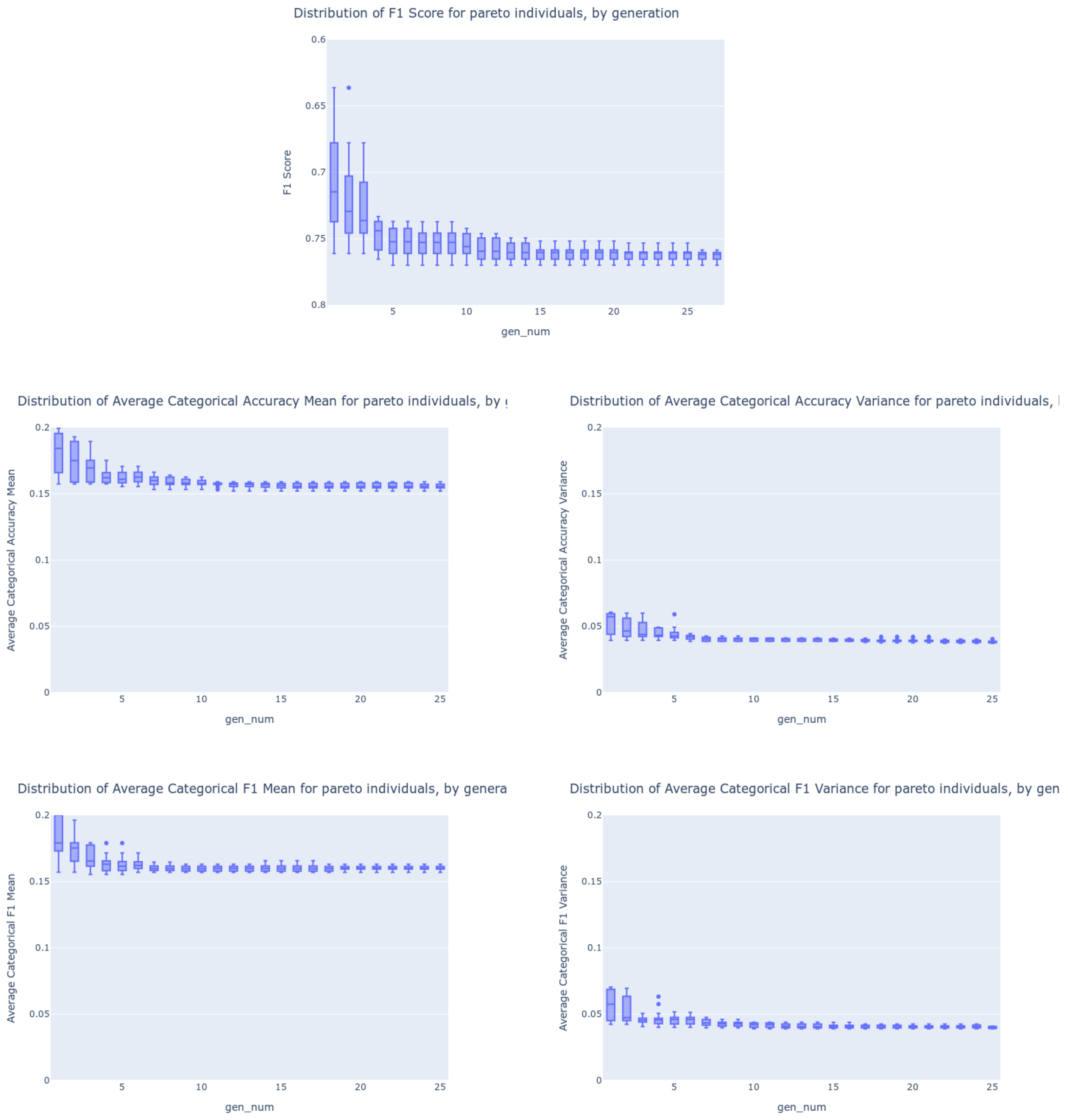


Table 11: Accuracy mean metric had the largest range of convergence and potentially presents the largest tradeoff space available for search.