Exploring the limits of deep learning in high-dimensional limited-sized data regimes using single- and multi-objective optimization strategies

Simon Jaxy[a], **Ann Nowé**[a] **and Pieter Libin**[a,b]

[a]Artificial Intelligence Lab, Department of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium
[b]Data Science Institute, Hasselt University, Martelarenlaan 42, B-3500 Hasselt, Belgium

Abstract. There is a substantial demand for deep learning methods that can work with limited, high-dimensional, and noisy datasets. Nonetheless, current research mostly neglects this area, especially in the absence of prior expert knowledge or knowledge transfer. In this work, we bridge this gap by studying the performance of deep learning methods on the true data distribution in a limited, high-dimensional, and noisy data setting. To this end, we conduct a systematic evaluation that reduces the available training data while retaining the challenging properties mentioned above. Furthermore, we extensively search the space of hyperparameters and compare state-of-the-art architectures and models build and trained from scratch to advocate for the use of multi-objective tuning strategies. Our experiments highlight the lack of performative deep learning models in current literature and investigate the impact of training hyperparameters. We analyze the complexity of the models and demonstrate the advantage of choosing models tuned under multi-objective criteria in lower data regimes to reduce the likelihood to overfit. Lastly, we demonstrate the importance of selecting a proper inductive bias given a limited-sized dataset. Given our results, we conclude that tuning models using a multi-objective criterion results in simpler yet competitive models when reducing the number of data points.

1 Introduction

In recent years, deep learning celebrated unprecedented success ranging from the rise of large language models [25, 40, 7] to the discovery of new materials [31, 8] and image generation [34, 36]. Pushing models further toward the boundary of computation enables a sheer unforeseeable range of possibilities [14], provided that the amount of data matches the demand of the steadily growing models and their hunger for data [6].

However, looking at the other end, where data is scarce but high-dimensional and noisy, we are still facing substantial obstacles. Research has long abstained from investigating deep learning models in these circumstances, although countless real-world applications require performative models to drive development and research. Rare diseases [3, 37] is a particular case of such an application where data can have very high-dimensional, noisy measurements and is limited by definition. Likewise, the discovery of new molecules to improve on existing drugs is a fundamentally low data problem as many newly proposed molecules are incompatible or even toxic [2]. Furthermore, archaeological discoveries rely on limited, highly complex data [23, 20]. Deep learning has made tremendous progress in the last few years, particularly regarding methodologies tailored for data in large quantities. In this work, we address settings where the data is limited, high-dimensional, and noisy by nature.

In recent work, Banerjee et al. [3] reflect on the state and applicability of deep learning methods for investigating rare diseases. To facilitate learning, they propose to increase the amount of data by combining data sets, injecting prior knowledge, or transferring the weights of a deep learning method trained on a related domain. However, in newly emerging fields such as diagnosing new rare diseases, simply merging datasets falls out of the question as we do not have relatable data. Therefore, using transfer learning to enable learning in high-dimensional yet scarce data environments is not an option since transferring weights from one domain to another might not always benefit performance [48].

We explore the limitations of deep learning when confronted with limited yet high-dimensional and noisy data without introducing prior knowledge or transferring weights from another domain. To do so, we conduct a systematic evaluation that mirrors the challenging data contexts and enables us to analyze deep learning and a deep Gaussian Process-based model in a controlled framework, by systematically decreasing the available data, starting from the entire data set, until only one percent of the original data remains. We then task the algorithms to predict using the complete test data to evaluate their abilities on the true data distribution. Furthermore, we extensively investigate the spaces of possible architectures and hyperparameters of several deep learning techniques to provide insights and guidelines to train models in this challenging setup. Specifically, we question the trade-off between performance and complexity in a multi-objective context. With our work, we demonstrate the capability of deep learning models to learn in this challenging framework.

The rest of the paper is structured as follows. We provide an overview of related work in section two. In sections three and four, we present our evaluation and deep learning models in detail before explaining our experimental procedures. Finally, we display and discuss our results in sections five and six respectively.

2 Related Work

Training small and efficient models that function correctly in challenging conditions remains an important challenge. Recently, the research community has recognized the lack of resources invested in this branch of deep learning, and part of it is shifting its focus away to reduce model complexity [5, 15, 30]. However, popular techniques to enable stable training in small data regimes, such as transfer learning [44, 47] or knowledge distillation [21, 43, 39], is not possible by definition of our problem framework as we are concerned with studying the behavior of deep learning methods for entirely new domains. Banerjee et al. [3] investigate the current state of machine learning in the context of rare diseases, where the authors propose several techniques to improve the dataset, such as harmonically combining data sets and reducing class imbalance with decision tree-based methods, as well as augmenting models, e.g., with knowledge graphs. Furthermore, applications of these techniques are summarized and analyzed. In our study, we take a general perspective on this difficult-to-train context, where the domain of rare diseases constitutes a specific case instead of focusing entirely on this field. Moreover, compared to [3], we do so without prior knowledge or the transfer of weights. Dou et al. [9] provide a broad review of machine learning methods facing small data challenges in molecular sciences. Our work differs since we create a synthetic evaluation in which we systematically control the quantity of information available to the learner. Additionally, we limit our exploration of deep learning models and abstain from other machine learning techniques as we seek to process high-dimensional data, a domain in which deep learning methods have been excelling in recent years, lifting the requirement to carefully design feature extraction techniques as it is frequent for traditional machine learning models. Brigato et al. [5] identified the need to find models that can work under small data availability. In their pioneering study, the authors reveal the benefit of using models with lower than state-of-the-art complexity by investigating convolutional neural networks under different image classification evaluations. Contrary to their work, we do not consider using sophisticated data augmentation techniques that require to be hand-designed and therefore sufficient prior knowledge [5]. In contrast to other works on limited data [5, 15], where the authors only present a few models per study, we expand our examinations to consider a broader range of deep learning models. As we search the space of deep learning architectures exhaustively, our work is related to Neural Architecture Search (NAS). NAS extends hyperparameter optimization by additionally searching architectural parameters [30, 10, 35]. It is concerned with finding the optimal network architecture without relying on a researcher's prior experience and freeing the process of required intuition by reducing the necessity of human intervention [35]. Our work builds on the principles of neural architecture search by considering a model's complexity and performance as two criteria for finding optimal architecture and tuning hyperparameters. We optimize hyperparameters using Bayesian and bandit-based search [13] to find suitable architectures and tuning hyperparameters. When performing our search concerning performance and complexity, we are interested in finding Pareto-optimal models resulting from the trade-off between these conflicting criteria. The Pareto frontier marks the boundary where we achieve an optimal trade-off between performance and model size such that we cannot find a model further optimized in one aspect without losing in the other [30].

3 Methods

We base our study on three pillars: the dataset, the models, and the search strategies. At first, we present our systematic evaluation which comprises the dataset. Then, we present the optimization strategies and lastly, the deep learning models.

3.1 Evaluation

In our experiments, we want to shed light on the performance and limitations of modern deep-learning techniques when faced with highly complex and noisy data under limited availability. To gain insights into the impact of the size of the dataset, we aim to show the limitations of such techniques when we systematically reduce the available data. More specifically, we consider the PTB-XL dataset [41], consisting of 21.837, 12-lead electrocardiogram recordings of 18.885 patients, having 71 labels in total [38] as our base data set, denoted as $\mathcal{X}$. The dataset contains an unbalanced label distribution, with possibly multiple labels per data point and noisy, high-dimensional time series data. We split $\mathcal{X}$ into a training $\mathcal{T}$, validation $\mathcal{V}$, and test set $\mathcal{U}$ in accordance with the stratified folds, as proposed in [41]. We use a downsampling rate, $\delta \in \{0.0, 0.2, 0.4, 0.6, 0.8, 0.85, 0.9, 0.95, 0.99\}$, to draw M subsets from $\mathcal{T}$ uniformly at random. For each δ, the random subset fulfills the following condition

$$|\mathcal{T}_\delta^i| = \lfloor |\mathcal{T}| * (1 - \delta) \rfloor \tag{1}$$

with $i = 1, \dots, M$, $| \cdot |$ denoting the size of a set and $\lfloor \cdot \rfloor$ the floor function. By training our models on the same fixed sets $\mathcal{T}_\delta^i$, for all values of i, and thus, avoiding training our models on different sub-samples of the data, we ensure comparability across the training runs of the different models. Further, we consider the complete validation set to tune the parameters according to the true data distribution. The validation in the original data split has been ensured to have high-quality samples and a balanced label distribution [41]. Thus, our tuning data set concerns:

$$\mathcal{D}_\delta^i = \{\mathcal{T}_\delta^i, \mathcal{V}\}. \tag{2}$$

We determine the performances and limitations of modern deep learning models in a two-fold procedure, where we define performance in terms of the macro-averaged area under the curve (macro AUC). To compute the macro AUC, each class AUC is calculated independently before averaging them. This accounts for a better judgment of the classifier than accuracy as it relieves the requirement to optimize for a threshold value [38], whereas micro-averaging, i.e., computing the average based on the combined true- and false positive rates of each class, would lead to an overrepresentation of highly populated classes [41].

3.2 Search

The task of automatically finding the optimal hyperparameters, be it architectural or training settings, is an active field of research [46, 13]. In hyperparameter optimization frameworks, the aim is to find the optimal setting of hyperparameters without having a human-in-the-loop to eliminate possible biases and the need for prior experience regarding specific machine learning models. For this, trials are sampled according to some heuristic and the black-box model is evaluated according to user-specified performance metrics. For our purposes, we differentiate between bandit-based optimization via the Asynchronous Successive Halving Algorithm (ASHA) [26], and Bayesian optimization using the Multiobjective Tree Parsen Estimator (MTPE) [32].

3.2.1 Asynchronous Successive Halving Algorithm

ASHA [26] can be best understood as a best-arm identification problem in a multi-armed bandit setup. The algorithm samples hyperparameter configurations where each configuration corresponds to an arm. It aims to identify the best-performing arm and, where in this setting this arm corresponds to the best-performing hyperparameter configuration. Formally, given a set of hyperparameter configurations Θ, we want to find the single best-performing arm according to the evaluation function f,

$$\begin{aligned} &maximize f(\theta) \\ &subject\, to\, \theta \in \Theta \end{aligned} \tag{3}$$

In doing so, the algorithm allocates a uniform computational budget to a predetermined number of configurations. After each evaluation, poorly performing configurations are eliminated, and their computational budget is redistributed to the remaining trials. This procedure is called successive halving [19]. In ASHA, this elimination process occurs asynchronously by evaluating each arm as soon as possible instead of waiting for the remaining arms, repeating the process until only one trial is left. Due to its asynchronous nature, the algorithm efficiently enables parallelization and scalability.

3.2.2 Multi-objective Tree Parsen Estimator

MTPE [32] is a multi-objective hyperparameter optimization algorithm in the form of

$$\begin{aligned} &maximize\, f(\theta) = (f_1(\theta), ..., f_M(\theta)) \\ &subject\, to\, \theta \in \Theta \end{aligned} \tag{4}$$

that, similar to ASHA, tries to find the best hyperparameter configuration, but instead of a single objective evaluation, it considers multiple objective functions. For this, a metric vector is constructed given an evaluation, $\zeta = f(\theta)$. Using this metric vector, containing the respective evaluations of each objective function, we can compare two hyperparameter settings by the concept of domination. Following[1][32], a vector ζ dominates a set of vectors Z if and only if for every $\zeta' \in Z : \forall\, i\, \zeta_i \geq \zeta'_i$ and $\exists\, j$ such that $\zeta_j > \zeta_j$, denoted as $\zeta \succ Z$, meaning that the metric vector ζ performs better in at least one metric $j \in \{1 \dots M\}$, while performing better or equal in the other metrics. Similarly, weak domination is defined as $\forall \zeta' \in Z : \forall\, i\, \zeta_i \geq \zeta'_i$ and denoted with $\zeta \succeq Z$. With the concept of domination, we can define two densities, $l(\theta)$ and $g(\theta)$,

$$p(\theta|\zeta) = \begin{cases} l(\theta) \text{ if } \zeta \succ Z^* \cup \zeta || Z \\ g(\theta) \text{ if } \zeta \preceq Z^* \end{cases} \tag{5}$$

where $Z^* = p(\zeta \succ Z^* \cup \zeta || Z^*) = \gamma$ with $\zeta || Z$ denoting that two vectors are incomparable (i.e., neither $\zeta \succeq Z$ nor $\zeta \preceq Z$). Thus, Z^* contains trials that are inferior or incomparable to ζ, while γ constitutes a threshold parameter that can be set by the user. Essentially, γ, divides the hyperparameter configuration according to their goodness, $l(\theta)$, i.e., trials that dominate older trials, and badness, $g(\theta)$, respectively. A new configuration is sampled by evaluating an acquisition function. In the case of MTPE, the corresponding acquisition function is the Expected Hypervolume Improvement (EHVI) given by

$$EHVI_{Z^*}(x) \propto (\gamma + \frac{g(\theta)}{l(\theta)}(1-\gamma))^{-1} \tag{6}$$

giving more emphasis to samples with a high probability under $l(\theta)$ and a low probability under $g(\theta)$. Subsequently, after evaluating the new hyperparameter configuration in terms of the objective functions $\{f_m\}_{m=1}^M$, the probability densities are updated, and a proceeding sample is chosen according to the acquisition function.

[1] Please note that instead of following the original notation of [32], we decided to switch the signs to emphasize that we are dealing with a maximization problem.

3.3 Models

We test a broad range of models to demonstrate the state of deep learning under limited data availability. We investigate the current best-performing models reported for the original PTB-XL dataset. Next, we question the performance of popular deep learning methods by extensively searching possible architectural settings.

3.3.1 State of the Art

As a baseline, we consider the models reported in [28, 29] consisting of a state space model (S4) [28, 16], a one-dimensional XResnet model [18], and an LSTM-based model (CPC) [29]. Each of these models comes with a fixed architecture. First, we train the models on each $\mathcal{T}_\delta^i$ with their default training hyperparameters as described in [28, 29]. The training hyperparameters under investigation are the learning rate and weight decay as these have a crucial impact on a deep learning model's performance [13], and batch size, which has shown to directly correlate with the ability to generalize [24, 17, 22]. Next, we tune each of these hyperparameters using the bandit-based hyperparameter optimization ASHA to analyze their impact on the performance of smaller data samples. For our purposes, we do not tune the SOTA models on the multi-objective criterion as their model size does not change since we consider their architectures fixed.

3.3.2 Base models

Additionally to the state-of-the-art models, we investigate a range of deep learning models, which we refer to as Base models, comprised of a convolutional neural network architecture (CNN), an LSTM-based architecture (LSTM), a transformer encoder (ENC), and a state-space model (S4) [16]. Furthermore, we explore the performance of deep Gaussian Process approximations using Random Fourier Features and convolutional layers (ConvRFF) [42].
We tune these models regarding architectural hyperparameters, such as the width and depth of the layers, activation functions, and dropout (see the Supplementary Information for a detailed overview). Further, we tune them for the same tuning hyperparameters as the SOTA models using ASHA [26] to gain insights over the preferred architectures in low but high-dimensional data settings. Additionally, we tune the hyperparameters of the Base models using a multi-objective search procedure concerning performance and complexity realized by the MTPE.

4 Results

We search the space of training and architectural hyperparameters using the single objective ASHA and the multi-objective TPE. We describe the experimental setup in the Supplementary Information B and present an overview of the associated experiments in Table 1. Each model is tuned for every down sample rate for each of the five data samples and 100 trials per optimization strategy, resulting in $|\delta|$xMx100 $= 450$ trials per optimization strategy. In the case of ASHA, we select the top-performing model of each data sample to train it on the respective data sample. Contrarily, as the multi-objective search does not result in a single best model but rather in a Pareto front of model configurations, we train each member of the

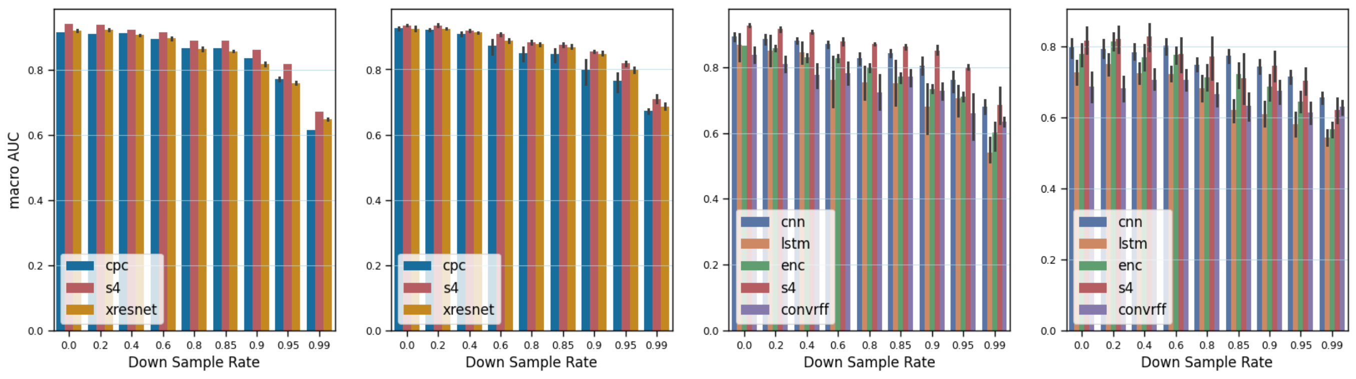

Figure 1. Performance is measured as macro AUC for all models, starting from the SOTA models (left), single-objectively tuned SOTA models (middle-left), single-objectively tuned Base models (middle-left), and multi-objectively tuned models (right) across all down sample rates.

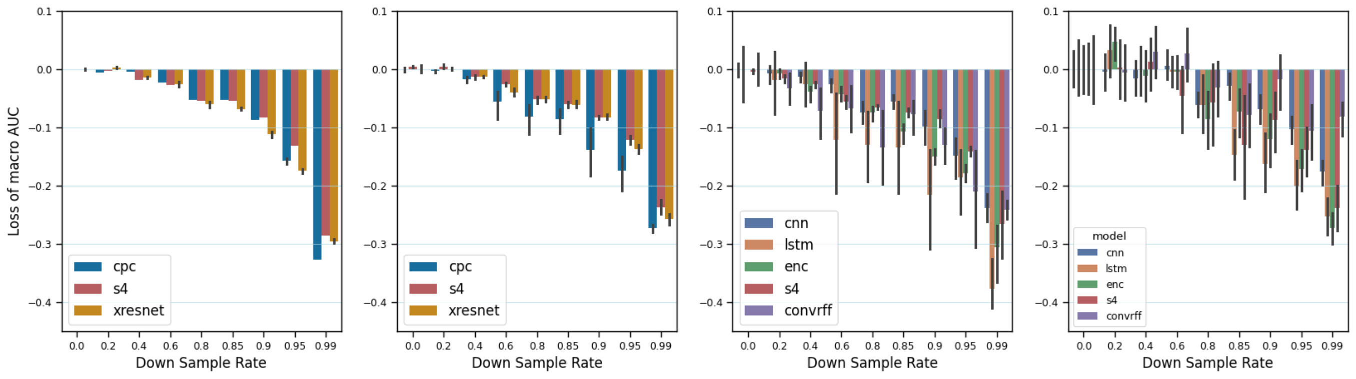

Figure 2. Loss of performance when comparing each down sample rate to the average performance on the entire data set, measured as macro AUC for all models, starting from the SOTA models (left), single-objectively tuned SOTA models (middle-left), single-objectively tuned Base models (middle-left), and multi-objectively tuned models (right) across all down sampling rates. The performance remains close to the baseline up to a down sample rate of 40%.

Model	Configurations	Pareto Trials	Full Runs
S4	900	216	351
CPC	450	-	90
XResnet	450	-	90
CNN	900	312	357
ConvRFF	900	248	293
LSTM	450	207	252
ENC	450	294	339
Total	4500	1277	1772

Table 1. An overview of the experiments. Each SOTA model has precisely one configuration, whereas each tuning run search ($5 \text{x} 9 \text{x} 100 = 450$) models, given 9 different down sample, 5 data samples, and 100 trials per optimization strategy. The number of Pareto trials varies for each model and each down sample rate. In total, we conduct 1772 training runs.

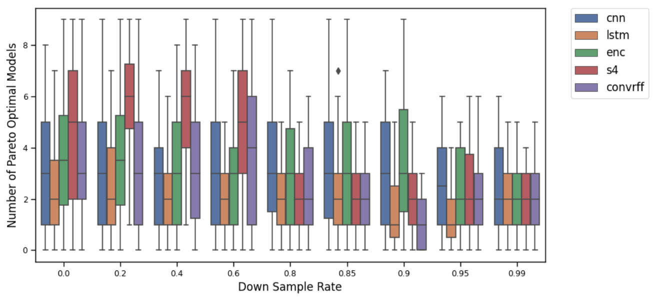

Figure 3. Number of Pareto optimal models per down sample rate.

Pareto front. In total, we train 1.772 models over 100 epochs, ranging over 9 different down sample rate settings and five data samples per down sample rate.
Figure 3 displays the number of Pareto models found per model and down sample rate. Notably, the number of Pareto optimal models decreases together with the data.

Performance

At first, we compare the performances on the test data, $\mathcal{U}$, of each model, trained for each $\mathcal{T}^i_\delta$. We depict their results in Figure 1 starting with the SOTA models on the left, the best-performing retuned models via the single-objective ASHA, and lastly, the models found by the multi-objective Bayesian search. In the case of the single objective hyperparameter tuning, we report that the performance of the SOTA models is higher than that of the Base models. However, the higher the down sample rate, the smaller the gap becomes, resulting in a marginal performance advantage for the S4 (0.68 ± 0.07) compared to the CNN (0.68 ± 0.03) for a down sample rate of 99%. In the multi-objective case, the CNN surpasses the S4 for a down sample rate of 60%, 95%, and 99%. For a down sample rate of 99%, the Gaussian process-based model ConvRFF significantly closes the gap between the two models based on point estimates and even beats the S4 model regarding macro AUC. Notably, the performance for all models remains stable upon a loss of 40 percent of the sample size, after which it starts to decline, as shown in Figure 2. Overall S4 is the superior model for most of the down sample rates.
Figure 2 shows the difference in the average performance of each model given a down sample rate of 0%, i.e., the complete data set, depicting the SOTA models (left), the single objective models (middle), and the multi-objective models (right). Remarkably, comparing all three configurations, we see that the multi-objectively tuned models perform stable even up to a down sample rate of 60%, remaining superior for all remaining down sample rates.

Compression vs. Performance

The trade-off between complexity and performance for the multi-objectively optimized models is shown in Figures 4, where we depict the Pareto front of a CNN model trained on $\mathcal{T}^3_{0.85}$ after tuning it for 10 epochs. In this example, we demonstrate that a model's complexity directly influences the performance in the early epochs. Two additional multi-objective search results are shown in the Supplementary Material C for a S4 and a ConvRFF model that culminated in two Pareto fronts that are less promiennt concerning their spread and amount compared to Figure 4.
Figure 5 shows the performances of fully trained models using $\mathcal{T}_{0.85}$, and more down sample rates are found in the Supplementary Information D. From this Figure, we can observe that given a fixed down sample rate the model size can be reduced without severely impacting the performance in return by carefully investigating the Pareto front.

Training Hyperparameters

Next, we investigate the role of the training hyperparameters. More specifically, we study whether the learning rate, weight decay, dropout, or batch size is impacted by the reduction of data samples. We depict the identified hyperparameters of each of the three searches, i.e., the search for the SOTA models, the single-objective, and the multi-objective Base models in Figures 15, 16, 17 (found in the Supplementary Information E). We note that the weight decay has no particular tendency except for the Bayesian optimisation procedure that saw an increase from 0.004 ± 0.01 for a down sample rate of 0% to 0.02 ± 0.03 given only 1% of the data set. The optimization algorithms set the batch size at a slightly lower value compared to the original batch size of 32 as reported in [28], particularly for higher down sample rates (26.11 ± 31.77 to 21.30 ± 24.52

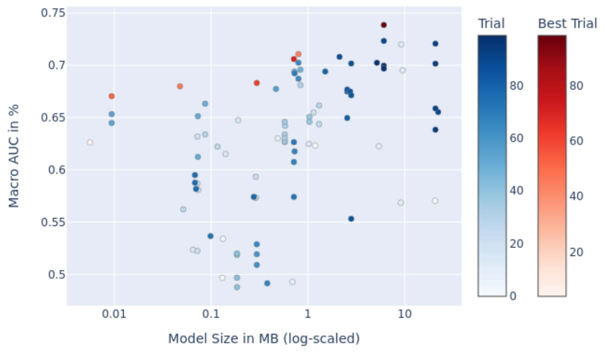

Figure 4. Pareto front of a CNN tuned on $\mathcal{T}^4_{0.85}$ while being optimized for performance and complexity.

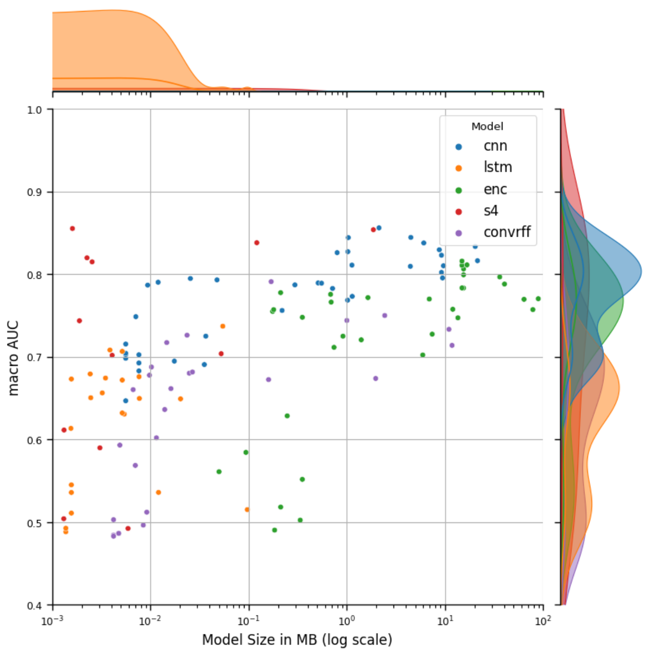

Figure 5. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.85}$ for performance and complexity simultaneously. Displayed for all $i = 1 \dots 5$.

for a down sample rate of 0% and 99% averaged over all search results). Overall, all three searches slightly increased the learning rates in their configurations compared to the learning rate used on the entire data set (0.005 ± 0.008 to 0.018 ± 0.022 again for $\delta = 0$ and $\delta = 0.99$ averaged over all searches). Lastly, the Bayesian multi-objective search resulted in higher dropout rates compared by roughly 15% (0.24 ± 0.19 to 0.40 ± 0.22, $\delta = 0$ and $\delta = 0.99$ respectively).

Overfit

Subsequently, we inspect the models for their susceptibility to overfit when reducing the amount of training data. Here, we determine overfit as the training loss minus the validation loss, and likewise training macro AUC minus the validation macro AUC. We show the respective values for all training epochs, averaged over each down sample rate in Figure 6, while displaying the non-averaged results for each search in Figure 18 and Figure 20 in the Supplementary Information F. Surprisingly, the SOTA models are well-tuned on average concerning overfit in both loss and performance, having a slight overfit on the loss. The models found by ASHA show a larger average overfit with respect to to the performance values. In contrast, the multi-objective models demonstrate comparably marginal overfit in terms of averaged loss and macro AUC. Figure 7 displays single trajectories of overfit and loss per model tuned with single-objective and multi-objective search, respectively. In both cases, the trajectories indicate that for a subset of models, the overfit diverges more strongly from the average case. Overall, the multi-objective search criterion results in models that are well-tuned on average.

Ability to capture classes

Lastly, we investigate the performance of each model for each class for a fixed down sample rate of 99% to reflect the most extreme case, meaning that only one percent of the original data is available. Figures 22-24, given in the Supplementary Information G, show the prediction of each model for each of the classes in the PTB-XL data set [41]. For all models, most of the classes are correct, i.e., they achieved a macro AUC of 50% or higher during evaluation on the complete test set $\mathcal{U}$.

5 Discussion

Our study compared the performance of state-of-the-art models and basic architectures via a systematic evaluation that tests a model's ability to deal with limited, highly dimensional, and noisy data. Furthermore, we extensively explored the space of architectural and training hyperparameters.
Each model suffers in performance loss when reducing the amount of data points. However, some models, such as the transformer encoder, are more affected than others (e.g., the state space model S4). We found that the models are surprisingly robust regarding overfit. Overall, we state that the models that are tuned with respect to performance and complexity are well-tuned (i.e., no over- nor underfit) compared to their single-objective counterparts.
Training hyperparameters, such as the learning rate and batch size, have a clear tendency to be impacted by a given a lower number of data points corroborating earlier findings in the literature [17], particularly for the lower batch sizes [24, 22]. Weight decay only plays

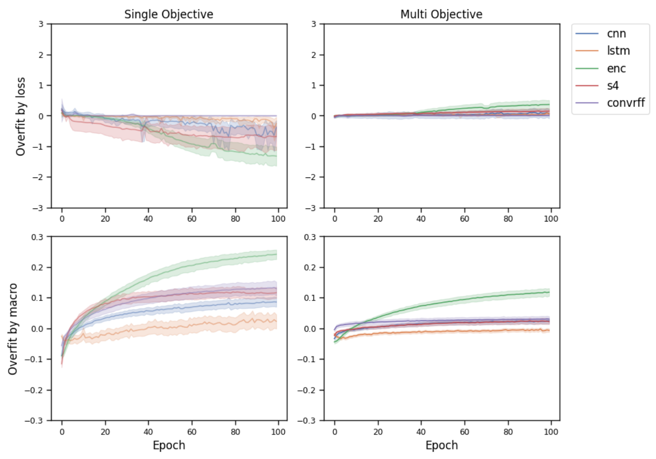

Figure 6. Overfit in terms of loss and macro AUC, calculated by subtracting the validation value from the training value, averaged over all down sample rates, and displayed for all epochs. In the case of the loss, we normalized the training and validation values before subtracting the latter from the former. The plots compare single-objectively (left) and multi-objectively (right) tuned models.

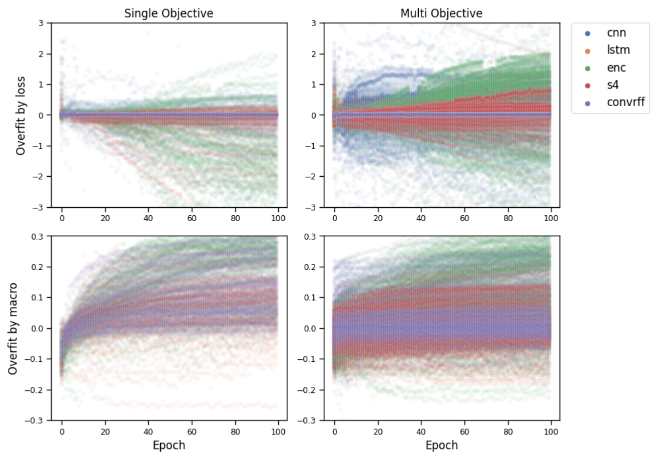

Figure 7. Overfit in terms of loss and macro AUC, calculated by subtracting the validation value from the training value, averaged over all down sample rates, and displayed for all epochs as single trajectories. In the case of the loss, we normalized the training and validation values before subtracting the latter from the former. The plots compare single-objectively (left) and multi-objectively (right) tuned models.

a marginal role in the single objective search, while displaying a tendency to be increased when considering performance and complexity. Similarly, dropout gains importance when the number of data points is decreased and it becomes more important to balance performance and model complexity, in accordance with the reports made by Brigato et al. [4].

We postulate that the superior performance of the state space model and the convolutional neural network-based models arise due to their inductive biases, specific for a given data type. This difference is noticeable in the single-objective models depicted in the mid-column of Figure 1, where the remaining models fall off quickly for higher down sample rates. The ability of the state space model to capture long-range signals [16] appears to be beneficial to classify electrocardiogram signals, as demonstrated on the original PTB-XL evaluation [28]. Thus, one way of augmenting the inferior models would be to expand their inductive biases, e.g., as proposed by Yin et al., who introduce a convolutional bias into their transformer architecture [45].

We highlighted the importance of balancing performance and model size by advocating for a multi-objective model selection when confronted with limited yet high-dimensional and noisy data. Our experiments demonstrated that the performance of our multi-objectively models remains more stable when compared to single-objectively tuned models, and are robust regarding overfit. Furthermore, we have shown that, in low data regimes, a model's complexity can be significantly reduced in low data regimes without sacrificing its performance, as seen in Figure 5. When ready-made state-of-the-art models are not yet available in the corresponding literature, we propose tuning the architectural and training hyperparameters of a model from scratch, as it can result in a performance close to the best-performing models in the respective domain, especially in lower data set sizes. Albeit the accuracy may not be the best possible, the reduction in overfit is a strong argument in favor. Additionally, the resulting Pareto front allows for a case-sensitive model selection in which we favor one objective over the other.

In certain cases, however, the Pareto front can have a sub-optimal shape, resulting in a set of models forming a concave curve. By averaging over these models, we obtain a new model that dominates the set of Pareto optimal models [12], meaning that it performs better in terms of performance while being of smaller complexity than the models of the Pareto front found to be optimal. Concretely, this means that if the Pareto front exhibits a concave shape, the overfit curves presented in Figure 6 overestimate the performance of the models. This is confirmed by, by analyzing Figure 7, from which it is evident that some model trajectories diverge stronger than the average, e.g., the transformer-based model. However, the ConvRFF and S4 models are well-represented by their respective averages. Figures 19 and 21 of the Supplementary Information F display the overfit in loss and macro AUC per down sample rate using single trajectories and averages over each $i = 1 \dots M$ with a 95% confidence interval per model. Overall, our results indicate that the trajectories exhibit a reduction in overfit, albeit with the possibility of degenerate cases.

Given the plethora of different hyperparameter settings, our work only explored the tip of the iceberg, but nonetheless parameters that are most relevant for model creation. However, we used efficient search strategies to navigate the space of possible settings.

Lastly, we want to address why we have chosen to tune our models on the entire validation set for every down sample rate instead of reducing the amount of validation similar to the training data. With this study, we want to reflect on the state of current deep learning methods when faced with reduced data in a high-dimensional and noisy setting in terms of capturing the true data distribution. While downsampling the validation data would be tailored specific to a case, we chose to emphasize the general capabilities of the deep learning methods across limited data samples. Using the entire validation data allows us to optimize the models according to the true data distributions and thus gives a better reflection of the model performances for low data regimes.

6 Conclusion

In this study, we have analyzed the current state of deep learning methods in limited, high-dimensional, and noisy data settings. Specifically, we have introduced a novel approach that allows to investigate models when reducing the data in a systematic manner. Furthermore, we have extensively searched the space of architectural and training hyperparameters using single-objective and multi-objective search. Our experiments revealed the importance of model selection according to multi-objective criteria to yield well-tuned models, competitive with state of the art methods. Moreover, we demonstrated impact of the inductive bias of a model in limited, high-dimensional and noisy data sets.

7 Acknowledgement

S.J. gratefully acknowledges support from Fonds Wetenschappelijk Onderzoek (FWO) via FWO PhD Fellowship strategic basic research, Belgium 1SHHV24N. P.J.K.L. wishes to express gratitude for the support received from FWO via postdoctoral fellowship, Belgium 1242021N and the research council of the Vrije Universiteit Brussel (OZR-VUB via grant number OZR3863BOF). This research is supported through the IMAGIca project by the Interdisciplinary Research Program of the Vrije Universiteit Brussel (reference IRP8_b). Lastly, we want to thank the HPC administration and support service of Vrije Universiteit Brussel that helped tremendously during the experimental phase and Bart Bogaerts for providing us with essential feedback and guidance throughout the development of this research.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[2] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017.

[3] J. Banerjee, J. N. Taroni, R. J. Allaway, D. V. Prasad, J. Guinney, and C. Greene. Machine learning in rare disease. *Nature Methods*, pages 1–12, 2023.

[4] L. Brigato and L. Iocchi. A close look at deep learning with small data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2490–2497, 2021.

[5] L. Brigato and L. Iocchi. A close look at deep learning with small data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2490–2497. IEEE, 2021.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[7] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[8] C. Chen, D. T. Nguyen, S. J. Lee, N. A. Baker, A. S. Karakoti, L. Lauw, C. Owen, K. T. Mueller, B. A. Bilodeau, V. Murugesan, and M. Troyer. Accelerating computational materials discovery with artificial intelligence and cloud high-performance computing: from large-scale screening to experimental validation, 2024.

[9] B. Dou, Z. Zhu, E. Merkurjev, L. Ke, L. Chen, J. Jiang, Y. Zhu, J. Liu, B. Zhang, and G.-W. Wei. Machine learning methods for small data challenges in molecular science. *Chemical Reviews*, 123(13):8736–8780, 2023.

[10] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[11] W. Falcon and The PyTorch Lightning team. PyTorch Lightning, Mar. 2019.

[12] F. Felten, E.-G. Talbi, and G. Danoy. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework, 2024. URL https://arxiv.org/abs/2311.12495.

[13] M. Feurer and F. Hutter. Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33, 2019.

[14] K. Grace, H. Stewart, J. F. Sandkühler, S. Thomas, B. Weinstein-Raun, and J. Brauner. Thousands of ai authors on the future of ai, 2024.

[15] S. Greydanus and D. Kobak. Scaling down deep learning with mnist-1d, 2024.

[16] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[17] F. He, T. Liu, and D. Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. *Advances in neural information processing systems*, 32, 2019.

[18] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. *CoRR*, abs/1812.01187, 2018.

[19] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR, 2016.

[20] A. H. Jamil, F. Yakub, A. Azizan, S. A. Roslan, S. A. Zaki, and S. A. Ahmad. A review on deep learning application for detection of archaeological structures. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 26(1):7–14, Jan. 2022.

[21] P. Kaliamoorthi, A. Siddhant, E. Li, and M. Johnson. Distilling large language models into tiny and effective students using pqrnn. *CoRR*, abs/2101.08890, 2021.

[22] I. Kandel and M. Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4):312–315, 2020.

[23] A. Karamitrou, F. Sturt, P. Bogiatzis, and D. Beresford-Jones. Towards the use of artificial intelligence deep learning networks for detection of archaeological sites. *Surface Topography: Metrology and Properties*, 10(4):044001, oct 2022.

[24] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[25] T. Le Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2022.

[26] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-Tzur, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2: 230–246, 2020.

[27] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[28] T. Mehari and N. Strodthoff. Advancing the state-of-the-art for ecg analysis through structured state space models. *arXiv preprint arXiv:2211.07579*, 2022.

[29] T. Mehari and N. Strodthoff. Self-supervised representation learning from 12-lead ecg data. *Computers in biology and medicine*, 141: 105114, 2022.

[30] G. Menghani. Efficient deep learning: A survey on making deep learning models smaller. *Faster, and Better. arXiv*, 2106, 2021.

[31] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

[32] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 533–541, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[34] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation, 2021.

[35] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[36] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[37] J. Schaefer, M. Lehne, J. Schepers, F. Prasser, and S. Thun. The use of machine learning in rare diseases: a scoping review. *Orphanet journal of rare diseases*, 15:1–10, 2020.

[38] N. Strodthoff, P. Wagner, T. Schaeffter, and W. Samek. Deep learning for ecg analysis: Benchmarks and insights from ptb-xl. *IEEE Journal of Biomedical and Health Informatics*, 25(5):1519–1528, 2021.

[39] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou. Mobilebert: a compact task-agnostic BERT for resource-limited devices. *CoRR*, abs/2004.02984, 2020.

[40] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[41] P. Wagner, N. Strodthoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):154, 2020.

[42] T. Wang, L. Xu, and J. Li. Sdcrkl-gp: Scalable deep convolutional random kernel learning in gaussian process for image recognition. *Neurocomputing*, 456:288–298, 2021. ISSN 0925-2312.

[43] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[44] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[45] M. Yin, Z. Chang, and Y. Wang. Adaptive hybrid vision transformer for small datasets. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 873–880. IEEE, 2023.

[46] T. Yu and H. Zhu. Hyper-parameter optimization: A review of algorithms and applications. *CoRR*, abs/2003.05689, 2020.

[47] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[48] W. Zhang, L. Deng, L. Zhang, and D. Wu. A survey on negative transfer. *IEEE/CAA Journal of Automatica Sinica*, 10(2):305–329, 2023.

Appendices

A Hyperparameters Settings

In the following, we list the training and the architectural hyperparameters considered during our model tuning.

A.1 *Training Hyperparameters Settings*

The training hyperparameters contain the learning rate, weight decay, and batch size. We use either a randomly uniformed value from a range of values, denoted as $(\cdot)$, or a sample from a list of values, denoted as $[\cdot]$:

- learning rate: $(1e-5, 1e-1)$
- weight decay: $(1e-5, 1e-1)$
- batch size: $[2^2, 2^3, 2^4, 2^5, 2^6, 2^7]$

A.2 *Architectural Hyperparameters Settings*

The architectural hyperparameters consist of hyperparameters specifying the depth and width of a neural network layer. Further, they include the activation function, dropout, and other hyperparameters that may be specific to certain architectural types. Similar to the training hyperparameters, we use either a randomly uniformed value from a range of values or a sample from a list of values. We set architectural hyperparameters such that each model's size complies with the memory provided by one NVIDIA A100 or P100 GPU. If we sample an architecture that overshoots this requirement, its trial is declared invalid.

A.2.1 *CNN*

The CNN architecture contains hyperparameters specifying the convolutional, normalization, and pooling layers. Furthermore, the final classification head receives either the averaged feature dimensions as input or the original output of the last convolutional layer.

- Number of layers: $[1, 2, 3, 4, 5]$
- Feature Sizes: $[2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8]$
- Kernel Sizes: $[3, 5, 7]$
- Padding: $[0, 1]$
- Strides: $[1, 2]$
- Pooling Function: $[AVG, MAX, None]$
- Normalization: $[BatchNorm, LayerNorm, None]$
- Activation Function: $[RELU, GELU, ELU]$
- Global Pooling: $[True, False]$
- Dropout: $(0.0, 0.75)$

A.2.2 *LSTM*

The LSTM's architecture determines the setup for each of the LSTM-specific layers.

- Number of layers: $[1, 2, 3, 4, 5]$
- Hidden Size: $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9]$
- LSTM bias: $[True, False]$
- LSTM bidirectional: $[True, False]$
- Activation Function: $[RELU, GELU, ELU]$
- Normalization: $[BatchNorm, LayerNorm, None]$
- Dropout: $(0.0, 0.75)$

A.2.3 *Transformer Encoder*

As before, the transformer encoder's architectural hyperparameters determine its layer-specific settings. Before passing the input to the final classification layer, the input is either pooled via Self Attention Pooling, Adaptive Pooling, or passed through linearly.

- Number of layers: $[1, 2, 3, 4, 5]$
- Hidden Size: $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9]$
- Number of Heads: $[1, 2, 4, 8]$
- Classification Layer: $[SelfAttentionPooling,$ $AdaptiveConcatPooling, None]$
- Dropout: $(0.0, 0.75)$

A.2.4 *State Space Model (S4)*

We define the architecture of the S4 [16] as the number of layers, the sizes of each layer, and the number of hidden states. Further, it may or may not be bidirectional.

- Number of layers: $[1, 2, 3, 4, 5]$
- Hidden Size: $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9]$
- Number of Hidden States: $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9]$
- S4 bidirectional: $[True, False]$
- Normalization: $[BatchNorm, LayerNorm, None]$
- Dropout: $(0.0, 0.75)$

A.2.5 *Convolutional RFF*

Here, we list the hyperparameters of the convolutional RFF [42]. The architectural choices are similar to those of the CNN except for the activation functions and the RFF-specific MC samples.

- Number of layers: $[1, 2, 3, 4, 5]$
- Kernel: $[RBF, ARCCOS]$
- Number of MC samples: $[1, 5, 10, 15, 20]$
- Feature Sizes: $[2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9]$
- Global Pooling: $[True, False]$
- Kernel Sizes: $[3, 5, 7]$
- Padding: $[0, 1]$
- Strides: $[1, 2]$

B Experimental Setup

We implement the experiments using PyTorch [33] and PyTorch Lightning [11]. We use RayTune [27] to implement ASHA and Optuna [1] for the MTPE. Our tuning is setup such that each sample of architectural or training hyperparameters constitutes a trial. An experiment consists of 100 trials per model, both SOTA, and Base, for each $\mathcal{D}_\delta^i$. We run each trial for 10 epochs and fix $M = 5$ to report the mean and variances per model and per down sample rate. After completing the tuning, we select the best hyperparameter according to their respective performances in the single objective optimization and every model that is part of the Pareto front for the multi objective search. We then train each of the best-performing models for 100 epochs. Subsequently, the models are tested on the complete testing data as we judge its quality to capture the underlying data distribution. All experiments (training, tuning and testing) are run on Nvidia A100 and P100 GPUs.

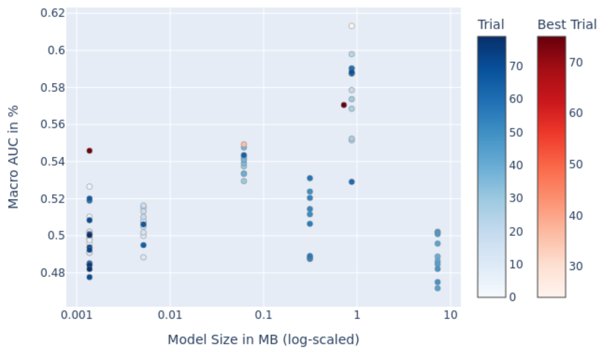

Figure 8. Pareto front of a S4 trained on $\mathcal{T}^2_{0.99}$ while being optimized for performance and complexity.

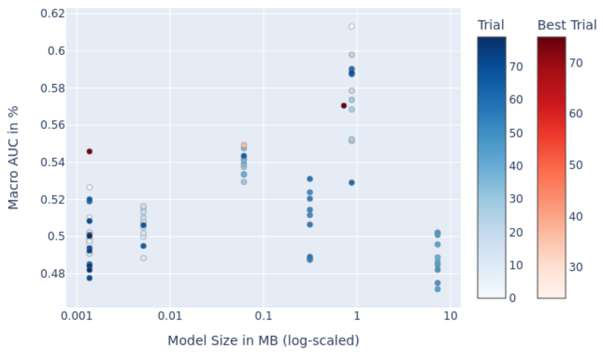

Figure 9. Pareto front of a ConvRFF trained on $\mathcal{T}^1_{0.8}$ while being optimized for performance and complexity.

C Pareto Front

Figures 8 and 9 show two Pareto fronts resulting from tuning a S4 model and a ConvRFF model on $\mathcal{T}^2_{0.99}$ and $\mathcal{T}^1_{0.8}$ respectively. Compared to Figure 4 the two searches returned a Pareto front that is much less elaborated resulting in only a few proposals.

D Compression vs. Performance

We display the model size (in MB) and the performance as measured in macro AUC for $\delta = \{0.0, 0.4, 0.8, 0.9, 0.99\}$ in Figures 10-fig:size-vs-score-ap5. We can see that for all models, the complexity can be reduced without loosing drastically in performance.

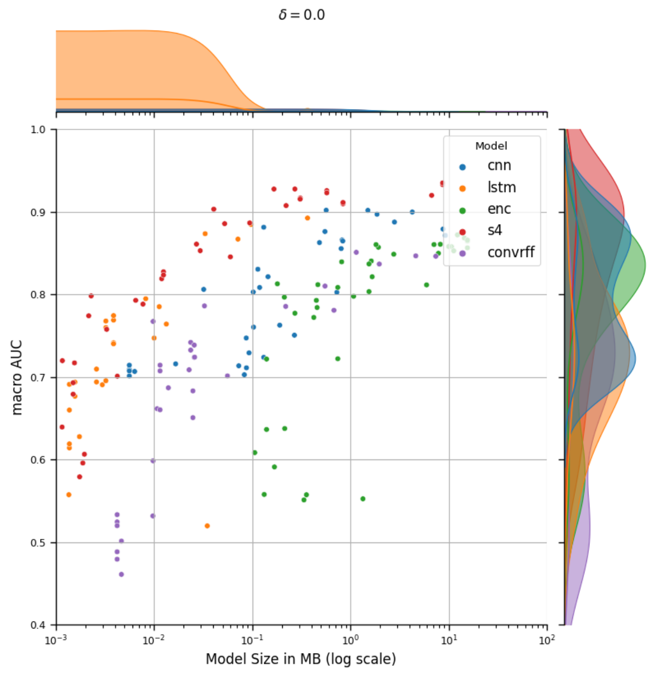

Figure 10. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.0}$ for performance and complexity simultaneously. Displayed for all $i = 1 \ldots 5$.

E Training Hyperparameters

Next, we investigate the role of the training hyperparameters. For this, we plot the mean and standard deviation of every training hyperparameter across all down sample rates. Figures 15-17 show the role of each hyperparameter in each setting, separated by model and search strategies.

F Overfit

In this section, we display the overfit as measured by the difference between training and validation loss and macro AUC for all models across every down sample rate. The result can be seen in Figure 18 and Figure 20. Further, we display single trajectories in Figure 21 and Figure 21.

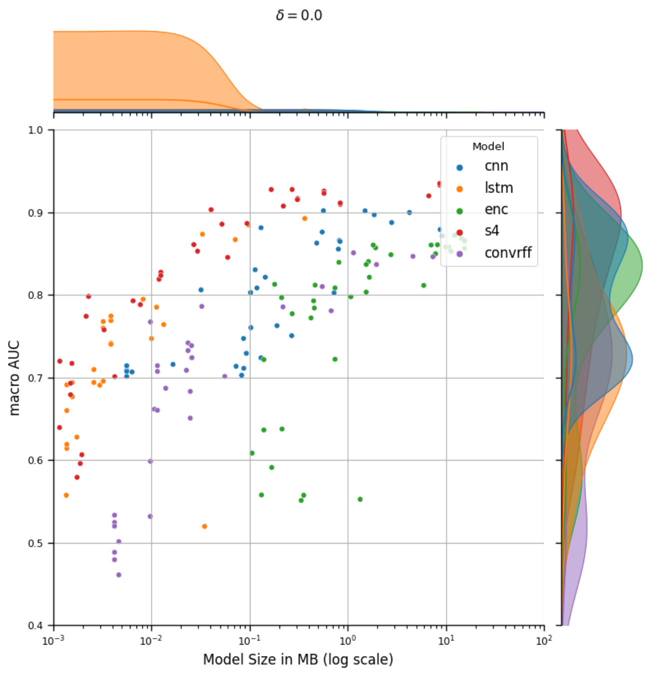

Figure 11. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.4}$ for performance and complexity simultaneously. Displayed for all $i = 1 \ldots 5$.

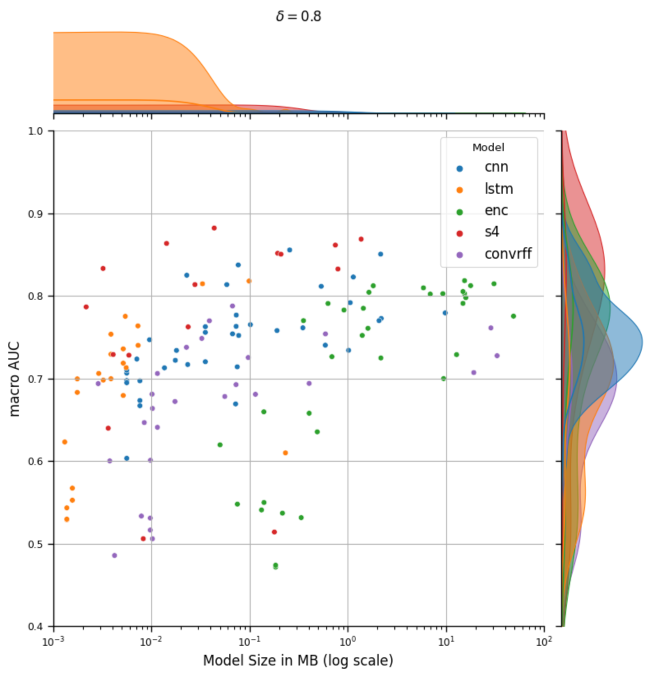

Figure 12. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.8}$ for performance and complexity simultaneously. Displayed for all $i = 1 \ldots 5$.

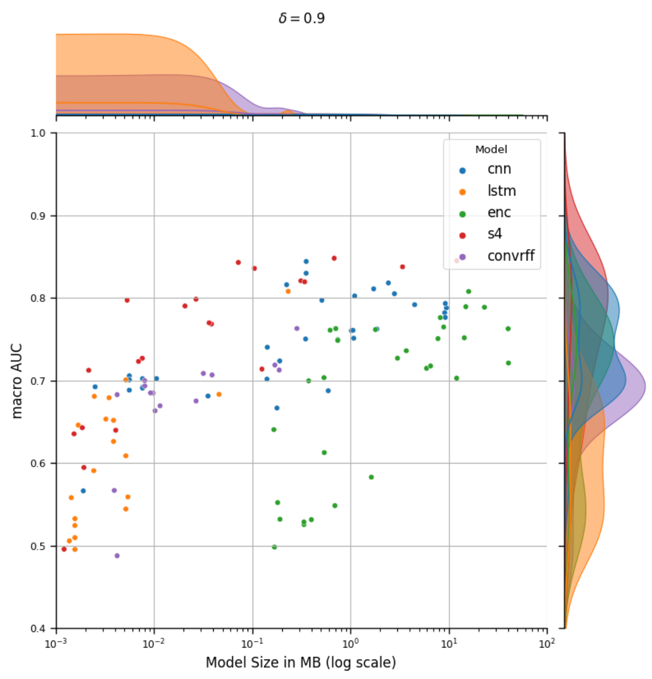

Figure 13. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.9}$ for performance and complexity simultaneously. Displayed for all $i = 1 \dots 5$.

G Class Performance

We study the performance of our models for each given class. Specifically, we focus on the case when only one percent of the data is available (see Figure 22-24). All models capture most of the classes correctly, even though the sample size for each label is severely lowered. Some, such as the ConvRFF tuned via the single-objective criterion, neglect one class in particular for which it may not have seen any data samples during training. In the case of the Base models tuned via both the single- and the multi-objective criterion, only the CNN performs better in the single-objective case. Especially the best performing SOTA model S4 fails to predict six classes on average by eliciting a performance less than 50%, whereas its multi-objectively tuned counterpart only neglects two classes. The overall best-performing model S4 significantly lowers class neglect in the multi-objective case by only missing two classes compared to five and six for the untuned and single-objective case.

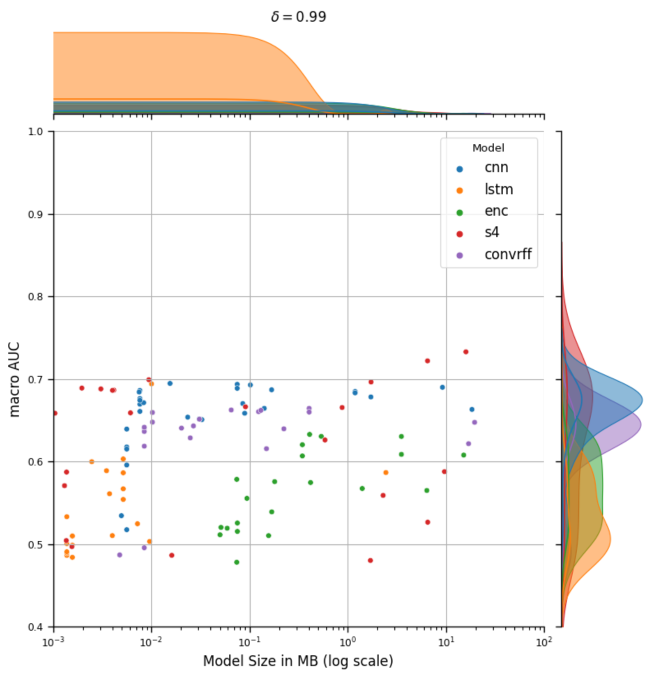

Figure 14. Size vs. performance of fully trained models after being optimized on $\mathcal{T}^i_{0.99}$ for performance and complexity simultaneously. Displayed for all $i = 1 \dots 5$.

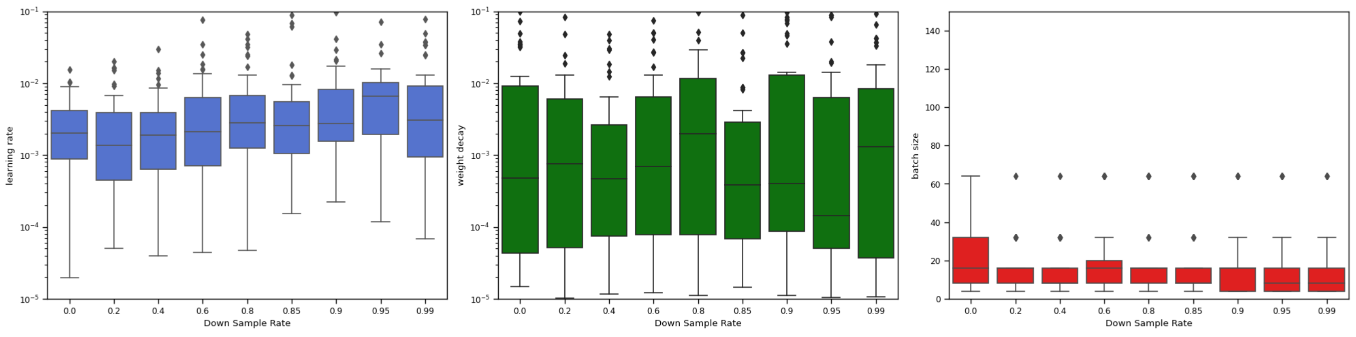

Figure 15. The training hyperparameters found by the single-objective hyperparameter tuning of the SOTA models using Asynchronous Successive Halving Algorithm. The hyperparameters depicted from left to right are the learning rate, weight decay, and batch size.

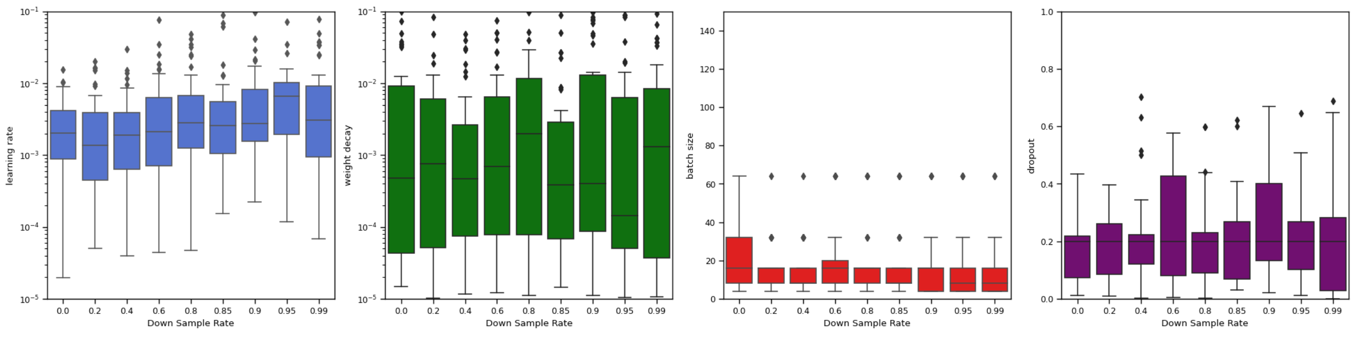

Figure 16. The training hyperparameters found by the single-objective hyperparameter tuning of the Basic models using Asynchronous Successive Halving Algorithm. The hyperparameters depicted from left to right are the learning rate, weight decay, batch size, and dropout.

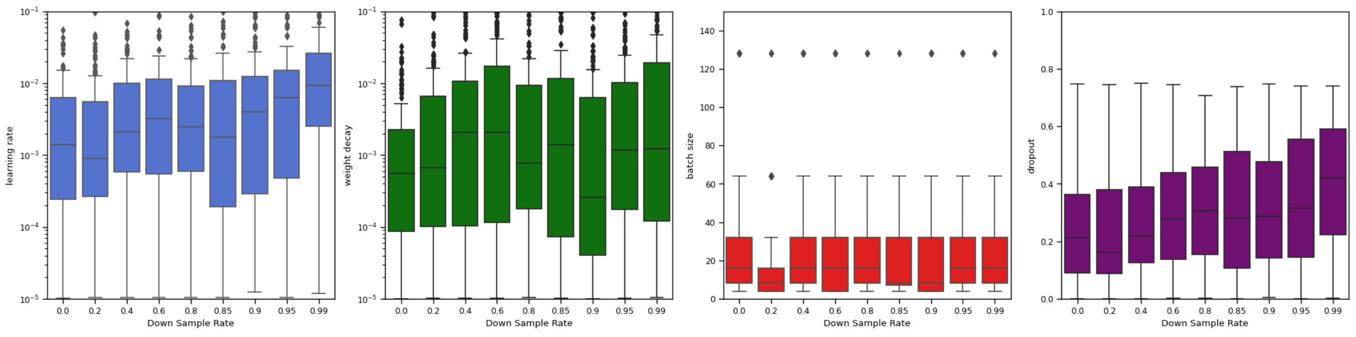

Figure 17. The training hyperparameters found by the multi-objective hyperparameter tuning of the Basic models using Multiobjective Tree Parsen Estimator. The hyperparameters depicted from left to right are the learning rate, weight decay, batch size, and dropout.

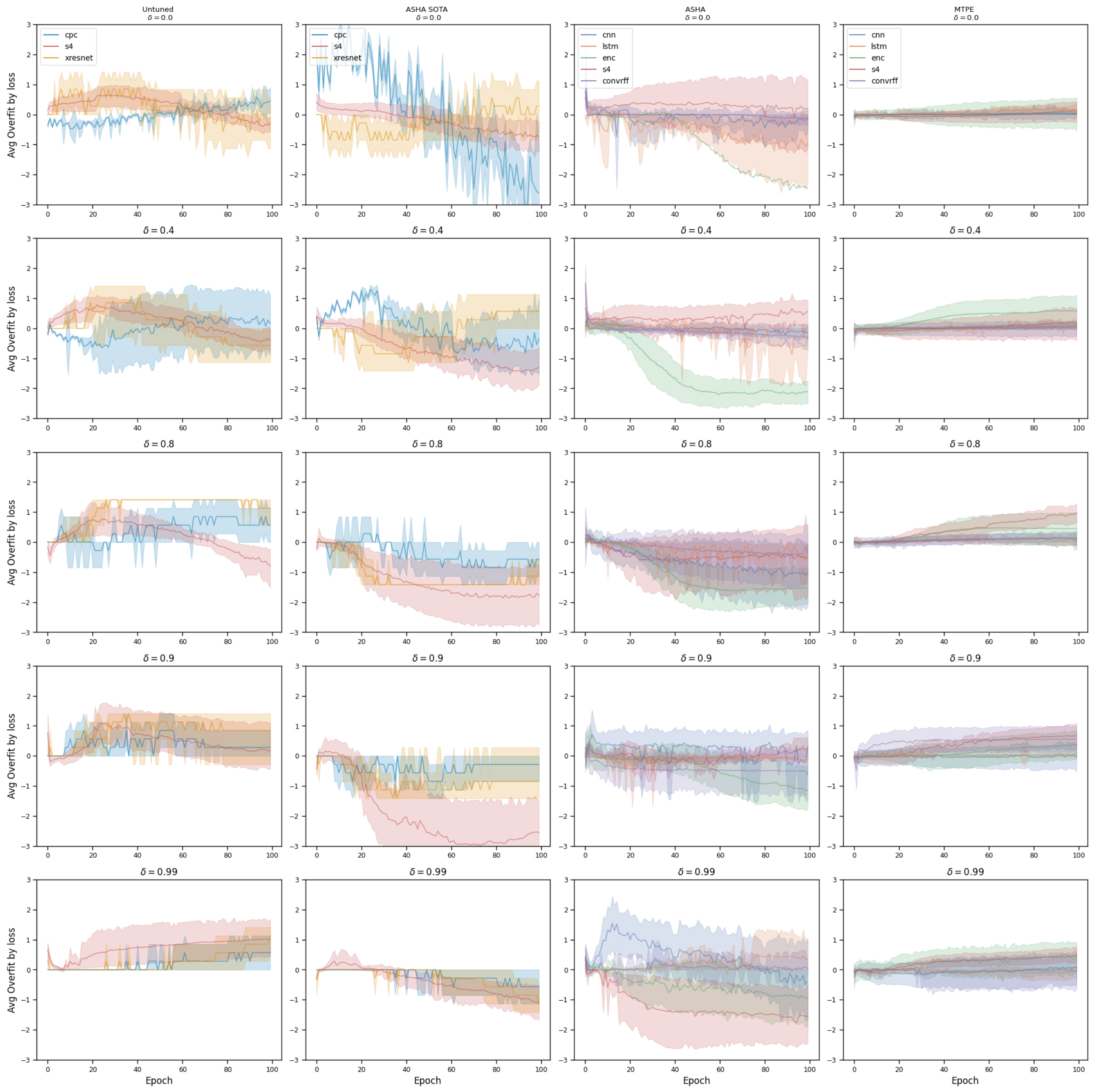

Figure 18. Overfit as measured by subtracting the normalized validation loss from the normalized training loss, for each down sample rate. From left to right the SOTA models, single-objective Basic models split into retuned SOTA Models (middle left) and tuned Base Models (middle right), and multi-objective Basic models are shown. We display the average trajectories with 95% confidence intervals.

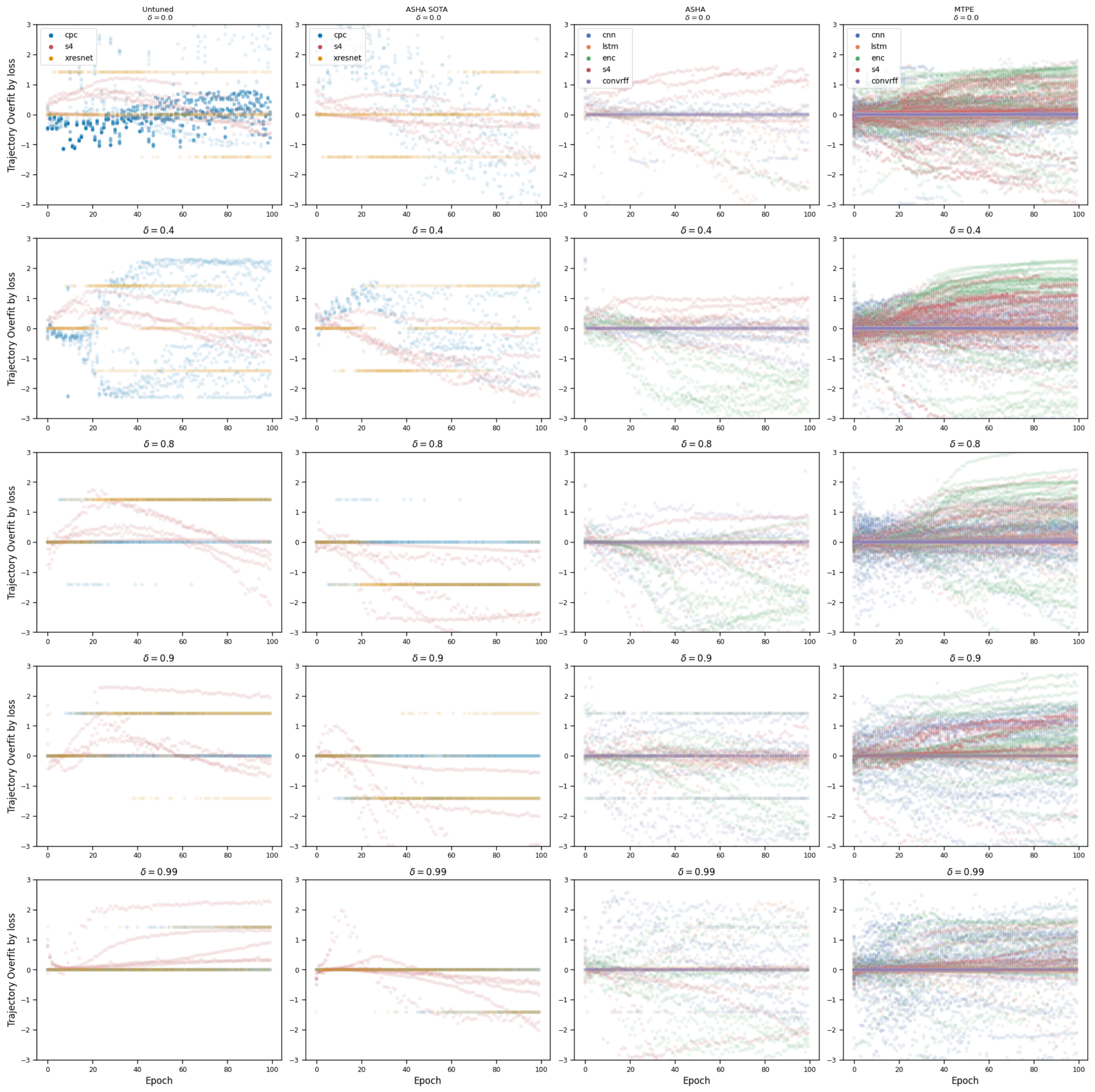

Figure 19. Overfit as measured by subtracting the normalized validation loss from the normalized training loss, for each down sample rate. From left to right the SOTA models, single-objective Basic models split into retuned SOTA Models (middle left) and tuned Base Models (middle right), and multi-objective Basic models are shown. We display single trajectories.

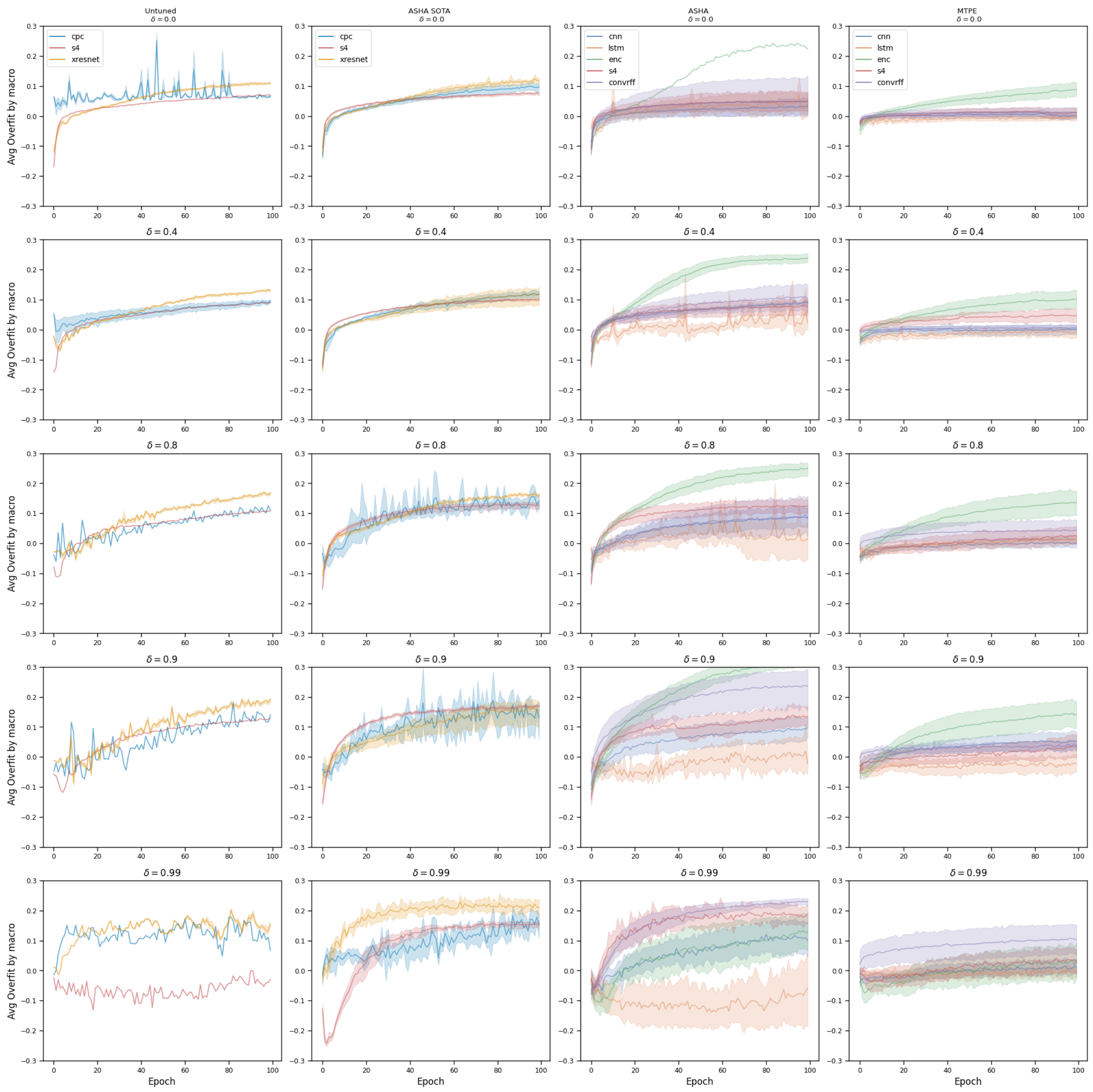

Figure 20. Overfit as measured by subtracting the validation macro AUC from the training macro AUC, for each down sample rate. From left to right the SOTA models, single-objective Basic models split into retuned SOTA Models (middle left) and tuned Base Models (middle right), and multi-objective Basic models are shown. We display the average trajectories with 95% confidence intervals.

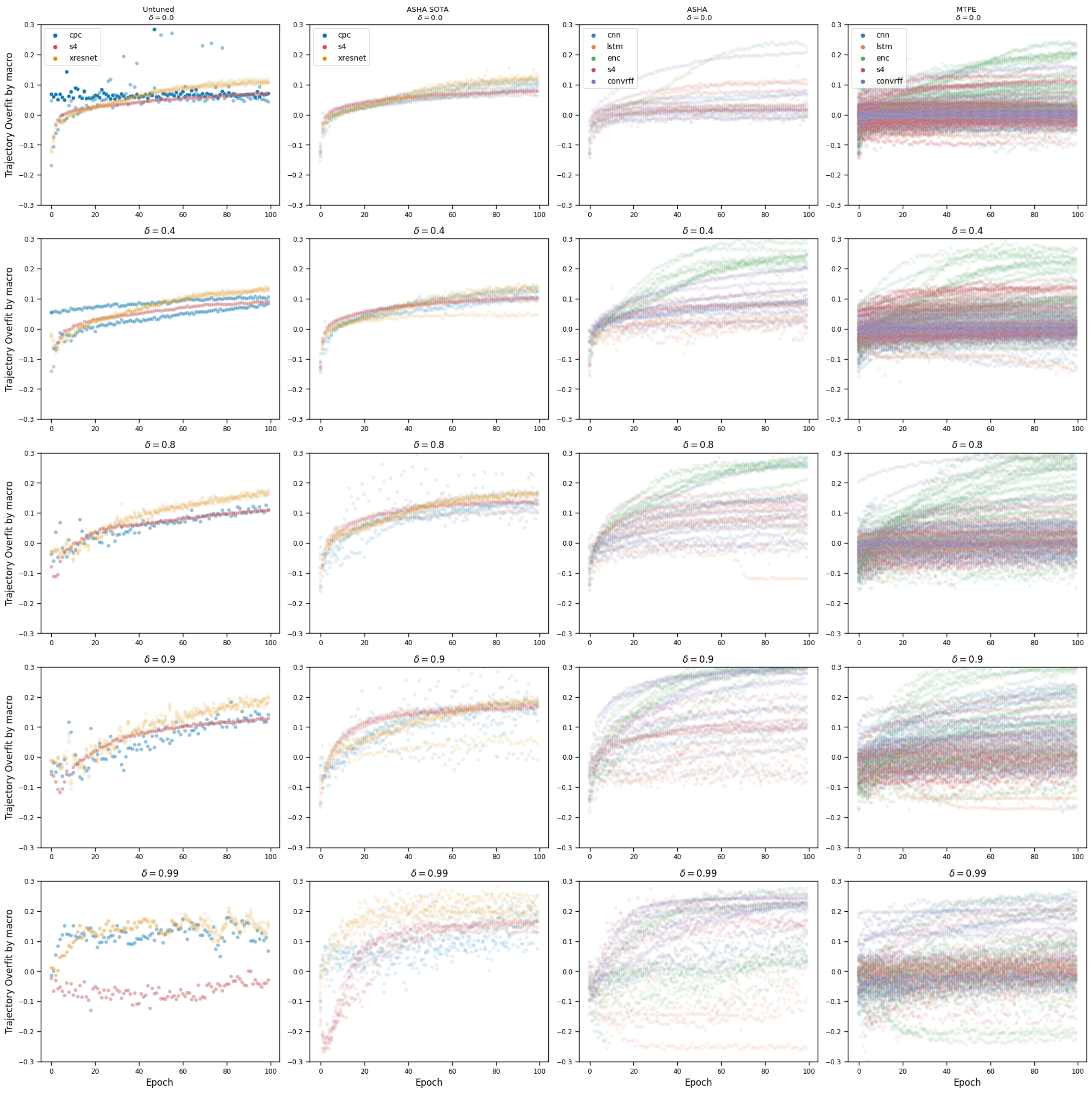

Figure 21. Overfit as measured by subtracting the validation macro AUC from the training macro AUC, for each down sample rate. From left to right the SOTA models, single-objective Basic models split into retuned SOTA Models (middle left) and tuned Base Models (middle right), and multi-objective Basic models are shown. We display single trajectories.

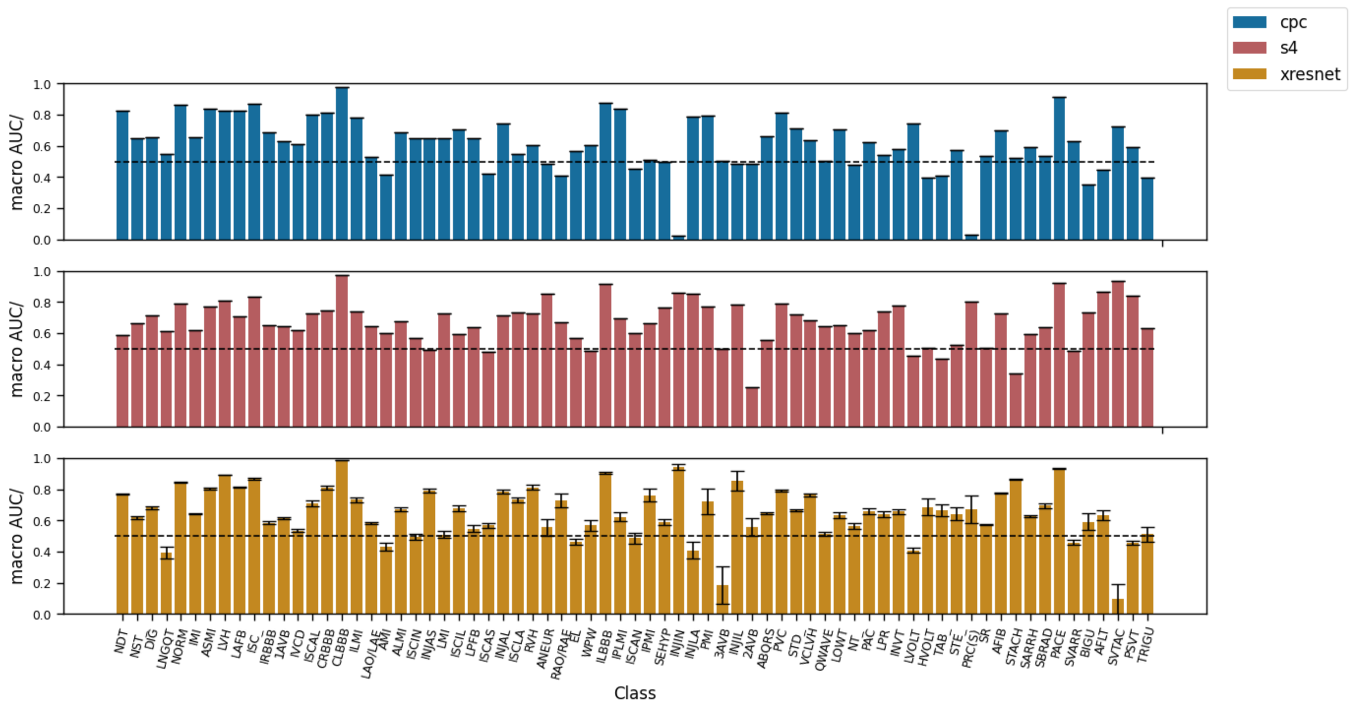

Figure 22. Performance per label for untuned SOTA models for a down sample rate of 99%.

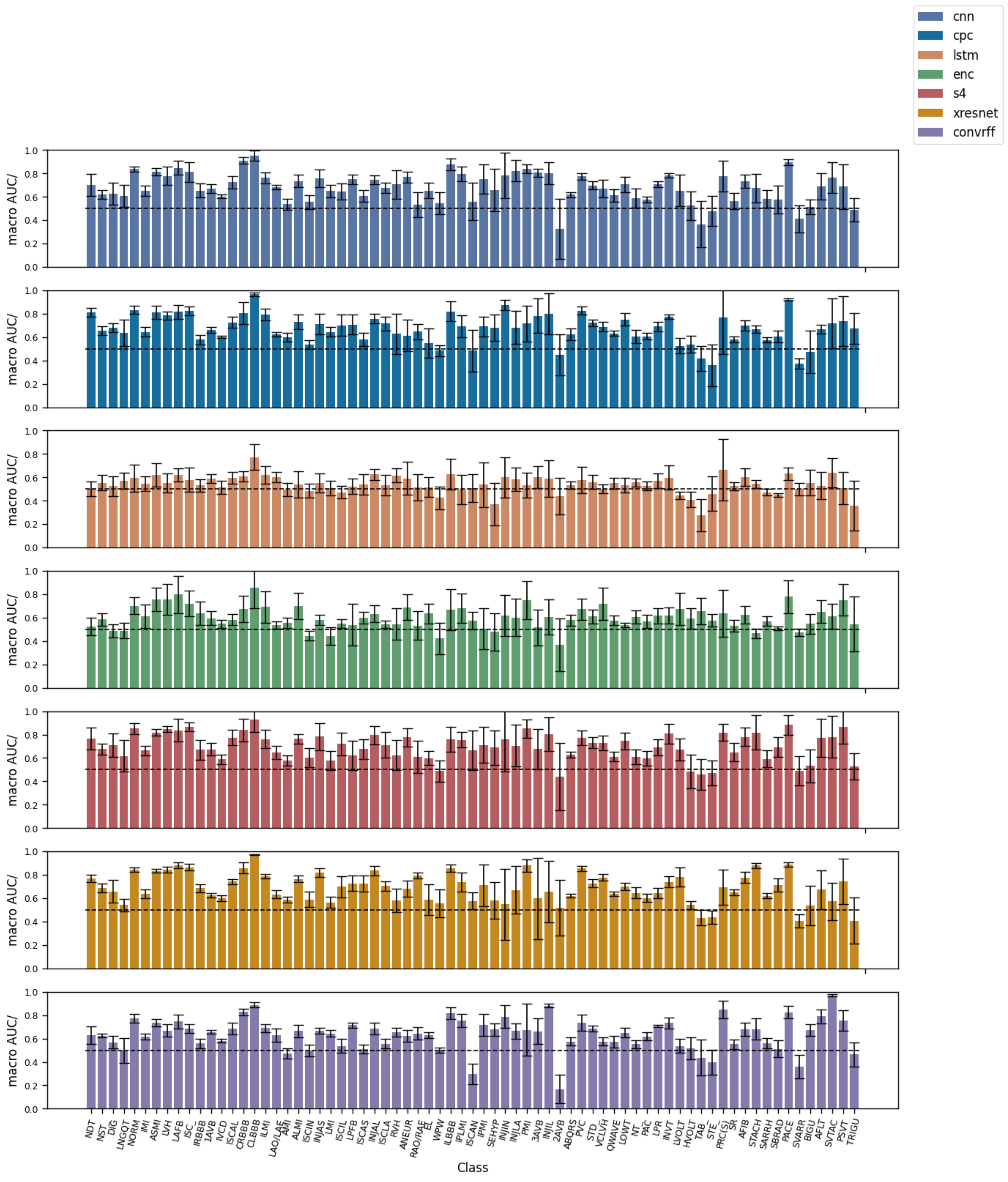

Figure 23. Performance per label tuned by the single-objective ASHA for a down sample rate of 99%.

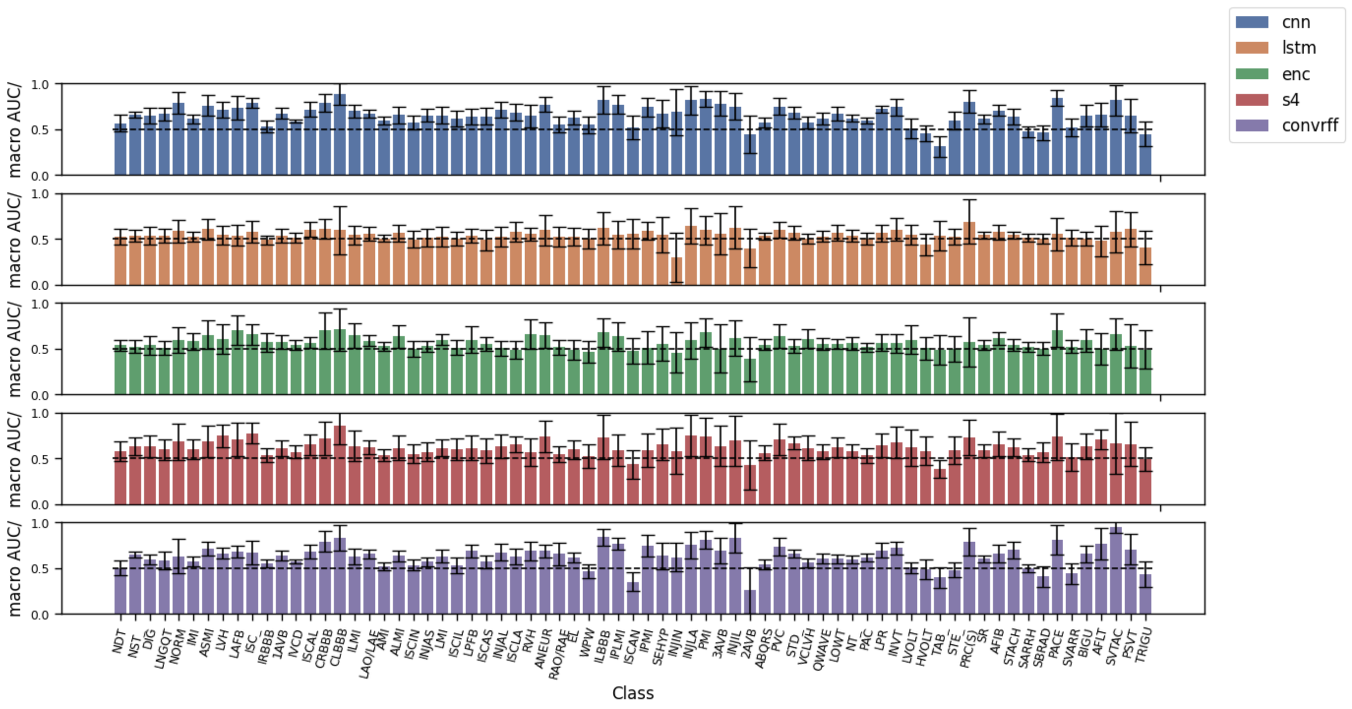

Figure 24. Performance per label tuned by the multi-objective TPE for a down sample rate of 99%.