TRANSFER LEARNING WITH DEEP TABULAR MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work on deep learning for tabular data demonstrates the strong performance of deep tabular models, often bridging the gap between gradient boosted decision trees and neural networks. Accuracy aside, a major advantage of neural models is that they are easily fine-tuned in new domains and learn reusable features. This property is often exploited in computer vision and natural language applications, where transfer learning is indispensable when task-specific training data is scarce. In this work, we explore the benefits that representation learning provides for knowledge transfer in the tabular domain. We conduct experiments in a realistic medical diagnosis test bed with limited amounts of downstream data and find that transfer learning with deep tabular models provides a definitive advantage over gradient boosted decision tree methods. We further compare the supervised and self-supervised pretraining strategies and provide practical advice on transfer learning with tabular models. Finally, we propose a pseudo-feature method for cases where the upstream and downstream feature sets differ, a tabular-specific problem widespread in real-world applications.

1 INTRODUCTION

Tabular data is ubiquitous throughout diverse real-world applications, spanning medical diagnosis (Johnson et al., 2016), housing price prediction (Afonso et al., 2019), loan approval (Arun et al., 2016), and robotics (Wienke et al., 2018), yet practitioners still rely heavily on classical machine learning systems. Recently, neural network architectures and training routines for tabular data have advanced significantly. Leading methods in tabular deep learning (Gorishniy et al., 2021; 2022; Somepalli et al., 2021; Kossen et al., 2021) now perform on par with the traditionally dominant gradient boosted decision trees (GBDT) (Friedman, 2001; Prokhorenkova et al., 2018; Chen and Guestrin, 2016; Ke et al., 2017). On top of their competitive performance, neural networks, which are end-to-end differentiable and extract complex data representations, possess numerous capabilities which decision trees lack; one especially useful capability is *transfer learning*, in which a representation learned on *pre-training* data is reused or fine-tuned on one or more *downstream* tasks.

Transfer learning plays a central role in industrial computer vision and natural language processing pipelines, where models learn generic features that are useful across many tasks. For example, feature extractors pre-trained on the ImageNet dataset can enhance object detectors (Ren et al., 2015), and large transformer models trained on vast text corpora develop conceptual understandings which can be readily fine-tuned for question answering or language inference (Devlin et al., 2019). One might wonder if deep neural networks for tabular data, which are typically shallow and whose hierarchical feature extraction is unexplored, can also build representations that are transferable beyond their pre-training tasks. In fact, a recent survey paper on deep learning with tabular data suggested that efficient knowledge transfer in tabular data is an open research question (Borisov et al., 2021). In this work, we show that deep tabular models with transfer learning definitively outperform their classical counterparts when auxiliary upstream pre-training data is available and the amount of downstream data is limited. Importantly, we find representation learning with tabular neural networks to be more powerful than gradient boosted decision trees with stacking – a strong baseline leveraging knowledge transfer from the upstream data with classical methods.

Some of the most common real-world scenarios with limited data are medical applications. Accumulating large amounts of patient data with labels is often very difficult, especially for rare conditions or hospital-specific tasks. However, large related datasets, e.g. for more common diagnoses, may be available in such cases. We note that while computer vision medical applications are common

(Irvin et al., 2019; Santa Cruz et al., 2021; Chen et al., 2018b; Turbé et al., 2021), many medical datasets are fundamentally tabular (Goldberger et al., 2000; Johnson et al., 2021; 2016; Law and Liu, 2009). Motivated by this scenario, we choose a realistic medical diagnosis test bed for our experiments both for its practical value and transfer learning suitability. We first design a suite of benchmark transfer learning tasks using the MetaMIMIC repository (Grzyb et al., 2021; Woźnica et al., 2022) and use this collection of tasks to compare transfer learning with prominent tabular models and GBDT methods at different levels of downstream data availability. We explore several transfer learning setups and lend suggestions to practitioners who may adopt tabular transfer learning. Additionally, we compare supervised pre-training and self-supervised pre-training strategies and find that supervised pre-training leads to more transferable features in the tabular domain, contrary to findings in vision where a mature progression of self-supervised methods exhibit strong performance (He et al., 2020).

Finally, we propose a pseudo-feature method which enables transfer learning when upstream and downstream feature sets differ. As tabular data is highly heterogeneous, the problem of downstream tasks whose formats and features differ from those of upstream data is common and has been reported to complicate knowledge transfer (Lewinson, 2020). Nonetheless, if our upstream data is missing columns present in downstream data, we still want to leverage pre-training. Our approach uses transfer learning in stages. In the case that upstream data is missing a column, we first pre-train a model on the upstream data without that feature. We then fine-tune the pre-trained model on downstream data to predict values in the column absent from the upstream data. Finally, after assigning pseudo-values of this feature to the upstream samples, we re-do the pre-training and transfer the feature extractor to the downstream task. This approach offers appreciable performance boosts over discarding the missing features and often performs comparably to models pre-trained with the ground truth feature values.

Our contributions are summarized as follows:

1. We find that recent deep tabular models combined with transfer learning have a decisive advantage over strong GBDT baselines, even those that also leverage upstream data.
2. We compare supervised and self-supervised pre-training strategies and find that the supervised pre-training leads to more transferable features in the tabular domain.
3. We propose a pseudo-feature method for aligning the upstream and downstream feature sets in heterogeneous data, addressing a common obstacle in the tabular domain.
4. We provide advice for practitioners on architectures, hyperparameter tuning, and transfer learning setups for tabular transfer learning.

2 Related Work

Deep learning for tabular data. The field of machine learning for tabular data has traditionally been dominated by gradient-boosted decision trees (Friedman, 2001; Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018). These models are used for practical applications across domains ranging from finance to medicine and are consistently recommended as the approach of choice for modeling tabular data (Shwartz-Ziv and Armon, 2022).

An extensive line of work on tabular deep learning aims to challenge the dominance of GBDT models. Numerous tabular neural architectures have been introduced, based on the ideas of creating differentiable learner ensembles (Popov et al., 2019; Hazimeh et al., 2020; Yang et al., 2018; Kontschieder et al., 2015; Badirli et al., 2020), incorporating attention mechanisms and transformer architectures (Somepalli et al., 2021; Gorishniy et al., 2021; Arık and Pfister, 2021; Huang et al., 2020; Song et al., 2019; Kossen et al., 2021), as well as a variety of other approaches (Wang et al., 2017; 2021; Beutel et al., 2018; Klambauer et al., 2017; Fiedler, 2021; Schäfl et al., 2021). However, recent systematic benchmarking of deep tabular models (Gorishniy et al., 2021; Shwartz-Ziv and Armon, 2022) shows that while these models are competitive with GBDT on some tasks, there is still no universal best method. Gorishniy et al. (2021) show that transformer-based models are the strongest alternative to GBDT and that ResNet and MLP models coupled with a strong hyperparameter tuning routine (Akiba et al., 2019) offer competitive baselines. Similarly, Kadra et al. (2021) find that carefully regularized MLPs are competitive. In a follow-up work, Gorishniy et al. (2022) show that

transformer architectures equipped with advanced embedding schemes for numerical features bridge the performance gap between deep tabular models and GBDT.

Transfer learning. Transfer learning (Pan and Yang, 2010; Weiss et al., 2016; Zhuang et al., 2020) has been incredibly successful in domains of computer vision and natural language processing (NLP). Large fine-tuned models excel on a variety of image classification (Dosovitskiy et al., 2020; Dai et al., 2021) and NLP benchmarks (Devlin et al., 2019; Howard and Ruder, 2018). ImageNet (Deng et al., 2009) pre-trained feature extractors are incorporated into the complex pipelines of successful object detection and semantic segmentation models (Chen et al., 2018a; Ren et al., 2015; Redmon and Farhadi, 2018; Redmon et al., 2016). Transfer learning is also particularly helpful in applications with limited data availability such as medical image classification (Alzubaidi et al., 2021; Heker and Greenspan, 2020; Chen et al., 2019; Alzubaidi et al., 2020).

In the tabular data domain, a recent review paper (Borisov et al., 2021) finds that transfer learning is underexplored and that the question of how to perform knowledge transfer and leverage upstream data remains open. In our work, we seek to answer these questions through a systematic study of transfer learning with recent successful deep tabular models.

Multiple works mention that transfer learning in the tabular domain is challenging due to the highly heterogeneous nature of tabular data (Jain et al., 2021; Lewinson, 2020). Several papers focus on converting tabular data to images instead (Sharma et al., 2019; Zhu et al., 2021; Sun et al., 2019) and leveraging transfer learning with vision models (Sun et al., 2019). Other studies explore designing CNN-like inductive biases for tabular models (Joffe, 2021), transferring XGBoost hyperparameters (Woźnica et al., 2022; Grzyb et al., 2021), and transferring whole models (Fang et al., 2019; Al-Stouhi and Reddy, 2011; Li et al., 2021) in the limited setting of shared label and feature space between the upstream and downstream tasks. Additionally, a concurrent work by Wang and Sun (2022) proposes a variable-column tabular neural network and applies it to transfer knowledge between datasets with partially-overlapping columns and same label space.

Stacking could also be seen as a form of leveraging upstream knowledge in classical methods (Wolpert, 1992; Ting and Witten, 1997).

Self-supervised learning. Self-supervised learning (SSL) aimed at harnessing unlabelled data through learning its structure and invariances has accumulated a large body of works over the last few years. Prominent SSL methods, such as Masked Language Modeling (MLM) (Devlin et al., 2019) in NLP and contrastive pre-training in computer vision (Chen et al., 2020) have revolutionized their fields making SSL the pre-training approach of choice (Devlin et al., 2019; Lan et al., 2019; Liu et al., 2019; Lewis et al., 2019; Chen et al., 2020; He et al., 2020; Caron et al., 2020; Bardes et al., 2021; Misra and Maaten, 2020). In fact, SSL pre-training in vision has been shown to produce more transferable features than supervised pre-training on ImageNet (He et al., 2020).

Recently, SSL has been adopted in the tabular domain for semi-supervised learning (Yin et al., 2020; Yoon et al., 2020; Ucar et al., 2021; Somepalli et al., 2021; Huang et al., 2020). Contrastive pre-training on auxilary unlabelled data (Somepalli et al., 2021) and MLM-like approaches (Huang et al., 2020) have been shown to provide gains over training from scratch for transformer tabular architectures in cases of limited labelled data. Finally, Rubachev et al. (2022) investigate benefits of supervised and unsupervised pretraining when applied to the same data without transferring knowledge between tasks.

3 Transfer Learning Setup in Tabular Domain

To study transfer learning in the tabular domain, we need to choose benchmark tasks and training pipelines. In this section, we detail both our upstream-downstream datasets as well as the tools we use to optimize transfer learning for tabular data.

3.1 MetaMIMIC Test Bed for Transfer Learning Experiments

MetaMIMIC repository. As medical diagnosis data often contains similar test results (i.e. features) across patients, and some diseases (i.e. tasks) are common while others are rare, this setting is a realistic use-case for our work. We thus construct a suite of transfer learning benchmarks using the MetaMIMIC repository Grzyb et al. (2021); Woźnica et al. (2022) which is based on the MIMIC-IV

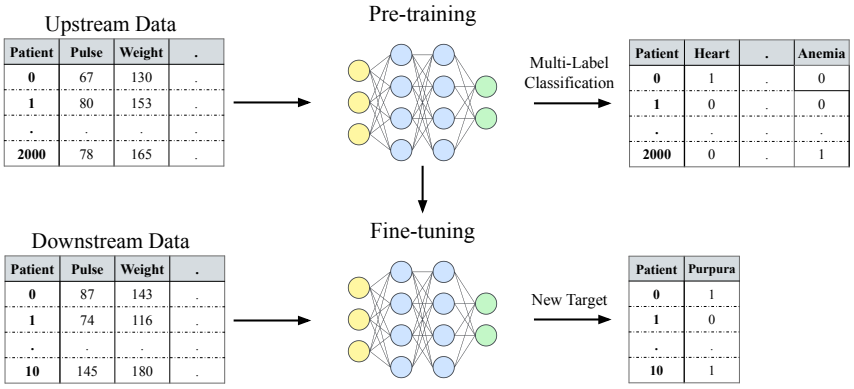

Figure 1: **Tabular transfer learning pipeline with MetaMIMIC.** We pre-train deep tabular neural networks on abundant upstream patient data with 11 diagnosis targets via multi-label classification. Then, we fine-tune the pre-trained models on limited downstream data with similar features to predict the new target diagnosis.

Johnson et al. (2021); Goldberger et al. (2000) clinical database of anonymized patient data from the the Beth Israel Deaconess Medical Center ICU admissions. MetaMIMIC contains 12 binary prediction tasks corresponding to different diagnoses such as hypertensive diseases, ischematic heart disease, diabetes, alcohol dependence and others. It covers 34925 patients and 172 features, including one categorical feature (gender), related to the medical examination of patients hand-selected by the authors to have the smallest number of missing values (Grzyb et al., 2021; Woźnica et al., 2022). The features include mean, maximum and minimum statistics of lab test results as well as general features such as height, weight, age and gender. The 12 medical diagnosis targets are related tasks of varied similarity and make MetaMIMIC a perfect test bed for transfer learning experiments.

Upstream and downstream tasks. A medical practitioner may possess abundant annotated diagnosis data corresponding to a number of common diseases and want to harness this data to assist in diagnosing another disease, perhaps one which is rare or for which data is scarce. To simulate this scenario, we create transfer learning problems by splitting the MetaMIMIC data into upstream and downstream tasks. Specifically, we reserve 11 targets for the upstream pre-training tasks and one target for the downstream fine-tuning tasks, thus obtaining 12 upstream-downstream splits – one for each downstream diagnosis. Additionally, we limit the amount of downstream data to 4, 10, 20, 100, and 200 samples corresponding to several scenarios of data availability.

In total, we have 60 combinations of upstream and downstream datasets for our transfer learning experiments. We pre-train our models as multi-label classifiers on the upstream datasets with 11 targets and then transfer the feature extractors onto the downstream binary diagnosis tasks, Figure 1 presents a diagram illustrating the pipeline.

3.2 Tabular Models

We conduct transfer learning experiments with four tabular neural networks, and we compare them to two GBDT implementations.

For neural networks, we use transformer-based architectures found to be the most competitive with GBDT tabular approaches (Gorishniy et al., 2021; Huang et al., 2020; Gorishniy et al., 2022). The specific implementations we consider include the recent FT-Transformer (Gorishniy et al., 2021) and TabTransformer (Huang et al., 2020). We do not include implementations with inter-sample attention (Somepalli et al., 2021; Kossen et al., 2021) in our experiments since these do not lend themselves to scenarios with extremely limited downstream data. In addition, we use MLP and ResNet tabular architectures as they are known to be consistent and competitive baselines (Gorishniy et al., 2021).

The TabTransformer architecture comprises of an embedding layer for categorical features, a stack of transformer layers and a multi-layer perceptron applied to concatenation of processed categorical

features and normalized numerical features. In contrast, FT-Transformer architecture transforms all features (including numerical) to embeddings and applies a stack of Transformer layers to the embeddings.

For GBDT implementation, we use the popular Catboost (Prokhorenkova et al., 2018) and XGBoost libraries (Chen and Guestrin, 2016).

3.3 Transfer Learning Setups and Baselines

In addition to a range of architectures, we consider several setups for transferring the upstream pre-trained neural feature extractors onto downstream tasks. Specifically, we use either a single linear layer or a two-layer MLP with 200 neurons in each layer for the classification head. We also either freeze the weights of the feature extractor or fine-tune the entire model end-to-end. To summarize, we implement four transfer learning setups for neural networks:

- Linear head atop a frozen feature extractor
- MLP head atop a frozen feature extractor
- End-to-end fine-tuned feature extractor with a linear head
- End-to-end fine-tuned feature extractor with an MLP head

We compare the above setups to the following baselines:

- Neural models trained from scratch on downstream data
- CatBoost and XGBoost with and without stacking

We use stacking for GBDT models to build a stronger baseline which leverages the upstream data (Wolpert, 1992; Ting and Witten, 1997). To implement stacking, we first train upstream GBDT models to predict the 11 upstream targets and then concatenate their predictions to the downstream data features when training downstream GBDT models.

3.4 Hyperparameter Tuning

The standard hyperparameter tuning procedure for deep learning is to randomly sample a validation set and use it to optimize the hyperparameters. In contrast, in our scenario we often have too little downstream data to afford a sizeable validation split. However, we can make use of the abundant upstream data and leverage hyperparameter transfer which is known to be effective for GBDT (Woźnica et al., 2022; Grzyb et al., 2021).

We tune the hyperparameters of each model with the Optuna library (Akiba et al., 2019) using Bayesian optimization. In particular, for GBDT models and neural baselines trained from scratch, we tune the hyperparameters on a single randomly chosen upstream target with the same number of training samples as available in the downstream task, since hyperparameters depend strongly on the sample size. The optimal hyperparameters are chosen based on the upstream validation set, where validation data is plentiful. We find this tuning strategy to be especially effective for GBDT and provide comparison with default hyperparameters in Appendix C. The benefits of this hyperparameter tuning approach are less pronounced for deep baselines.

For deep models with transfer learning, we tune the hyperparameters on the full upstream data using the available large upstream validation set with the goal to obtain the best performing feature extractor for the pre-training multi-target task. We then fine-tune this feature extractor with a small learning rate on the downstream data. As this strategy offers considerable performance gains over default hyperparameters, we highlight the importance of tuning the feature extractor and present the comparison with default hyperparameters in Appendix C as well as the details on hyperparameter search spaces for each model.

4 Results for Transfer Learning with Deep Tabular Models

In this section, we compare transfer learned deep tabular models with GBDT methods at varying levels of downstream data availability. We note that here we present the aggregated results in the

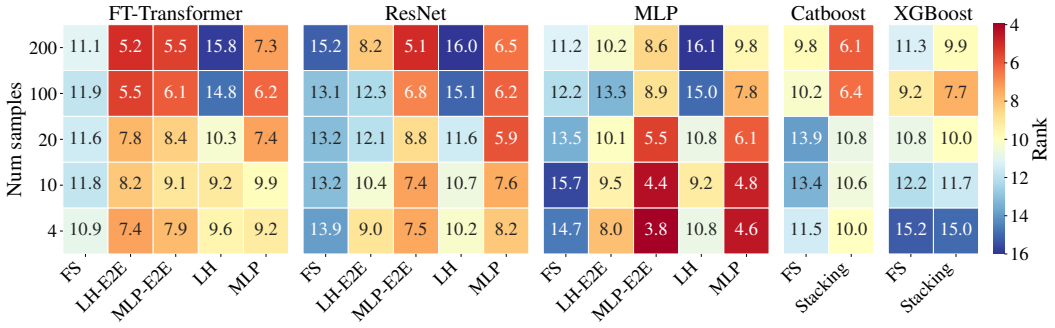

Figure 2: **Average model ranks across all downstream tasks.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks. FT-Transformer fine-tuned end-to-end outperforms all GBDT models, including GBDT with stacking, at all data levels. MLP is highly competitive in low data regimes.

form of the average rank of the models across all of the twelve downstream tasks at each data level. We choose this rank aggregation metric since it does not allow a small number of high-variance tasks to dominate comparisons, unlike, for example, average accuracy. Ranks are computed taking into account statistical significance of the performance differences between the models. We further report the detailed results for all of the models on all datasets and results for TabTransformer in Appendix D.

Figure 2 presents average model ranks on the downstream tasks as a heatmap where the warmer colors represent better overall rank. The performance of every model is shown on the dedicated panel of the heatmap with the results for different transfer learning setups presented in columns. First, noting the color pattern in the Catboost and XGBoost columns, we observe that deep tabular models pre-trained on the upstream data outperform GBDT at all data levels and especially in the low data regime of 4-20 downstream samples.

We emphasize that knowledge transfer with stacking, while providing strong boosts compared to from-scratch GBDT training (see Stacking and FS columns of GBDT), still decisively falls behind the deep tabular models with transfer learning, suggesting that representation learning for tabular data is significantly more powerful and allows neural networks to transfer richer information than simple predictions learned on the upstream tasks.

We summarize the main practical takeaways of Figure 2 below:

- Simpler models such as MLP with transfer learning are competitive, especially in extremely low data regimes. More complex architectures like FT-Transformer offer consistent performance gains over GBDT across all data levels and reach their peak performance in higher data regimes.
- Representation learning with deep tabular models provides significant gains over strong GBDT baselines leveraging knowledge transfer from the upstream data through stacking. The gains are especially pronounced in low data regimes.
- Regarding transfer learning setups, we find that using an MLP head with a trainable or frozen feature extractor is effective for all deep tabular models. Additionally, a linear head with an end-to-end fine-tuned feature extractor is competitive for FT-Transformer.

To verify that transfer learning with deep tabular models works well beyond the medical domain, we conduct experiments on other datasets: Yeast functional genomics data from the biological sciences domain (Elisseeff and Weston, 2001) and Emotions data from the music domain (Trohidis et al., 2008). Both datasets are multilabel, and we treat each classification label as a separate task. Similarly to the experiments with MetaMIMIC, we split the tasks into downstream and upstream by reserving

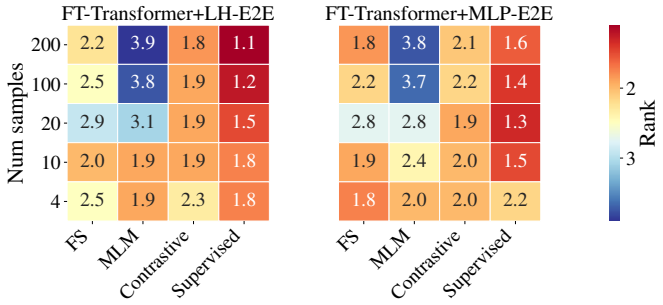

Figure 3: **Comparison of supervised and self-supervised pre-training strategies for FT-Transformer.** The left panel illustrates end-to-end fine-tuning with linear head and the right plot illustrates end-to-end fine-tuning with MLP head, the two most effective strategies for FT-Transformer. Within each panel, columns represent pre-training strategies and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. Contrastive pre-training outperforms from-scratch trained models, while MLM pre-training is not effective. Supervised pre-training outperforms all self-supervised pre-training strategies in our experiments.

$n-1$ labels for the upstream task and the n-th label for the downstream task. We report results of these experiments in Figures 8, 9 in Appendix D.5. We observe similar trends and in particular that deep tabular models pre-trained on upstream data and finetuned with MLP head outperform deep baselines trained from scratch and Catboost models leveraging stacking.

5 Self-Supervised Pre-training

In domains where established SSL methods are increasingly dominant, such as computer vision, self-supervised learners are known to extract more transferable features than models trained on labelled data (He et al., 2020; 2021). In this section, we compare supervised pre-training with unsupervised pre-training and find that the opposite is true in the tabular domain. We use the Masked Language Model (MLM) pre-training recently adapted to tabular data (Huang et al., 2020) and the tabular version of contrastive learning (Somepalli et al., 2021). Since both methods were proposed for tabular transformer architectures, we conduct the experiments with the FT-Transformer model. The inferior performance of self-supervised pre-training might be a consequence of the fact that SSL is significantly less explored and tuned in the tabular domain than in vision or NLP.

5.1 Tabular MLM Pretraining

Masked Language Modeling (MLM) was first proposed for language models by Devlin et al. (2019) as a powerful unsupervised learning strategy. MLM involves training a model to predict tokens in text masked at random so that its learned representations contain information useful for reconstructing these masked tokens. In the tabular domain, instead of masking tokens, a random subset of features is masked for each sample, and the masked values are predicted in a multi-target classification manner (Huang et al., 2020). In our experiments, we mask one randomly selected feature for each sample, asking the network to learn the structure of the data and form representations from $n-1$ features that are useful in producing the value in the n-th feature. For more detail, see Appendix B.

5.2 Contrastive Pre-Training

Contrastive pre-training uses data augmentations to generate positive pairs, or two different augmented views of a given example, and the loss function encourages a feature extractor to map positive pairs to similar features. Meanwhile, the network is also trained to map negative pairs, or augmented views of different base examples, far apart in feature space. We utilize the implementation of contrastive learning from Somepalli et al. (2021). In particular, we generate positive pairs by applying two data augmentations: CutMix (Yun et al., 2019) in the input space and Mixup (Zhang et al., 2017) in the embedding space. For more details, see Appendix B.

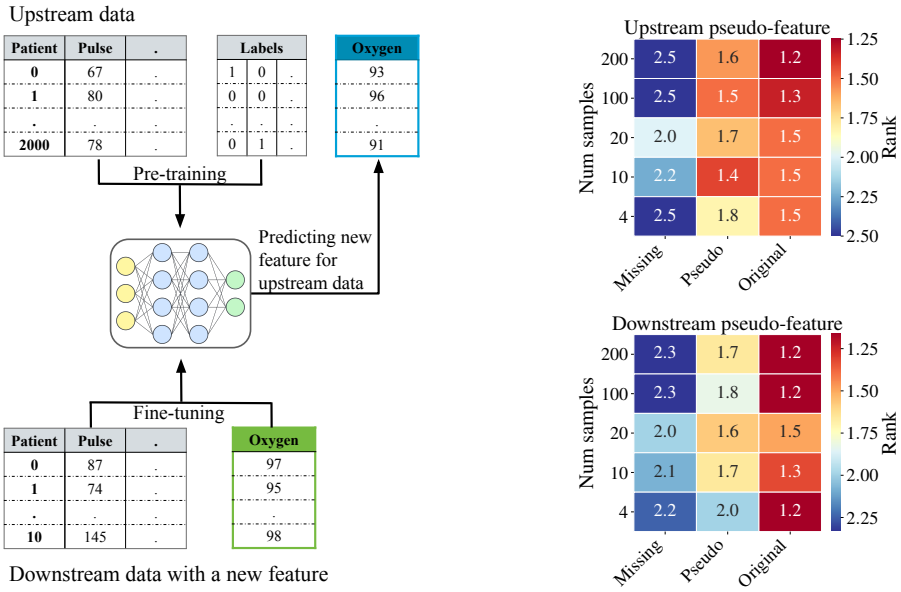

Figure 4: **Pseudo-Feature method for aligning upstream and downstream feature sets.** Left: Diagram illustrating strategy for handling mismatches between the upstream and downstream features. When upstream data is missing a feature present in downstream data, it is predicted with a model pre-trained on upstream data and fine-tuned to predict the new feature on the downstream data. Right top: Scenario with missing feature in the upstream data. Comparison of ranks of FT-Transformer model trained on data with missing feature, with pseudo-feature and with the original feature. Right bottom: Scenario with missing feature in the downstream data. Comparison of ranks of FT-Transformer model trained and fine-tuned on data with missing feature, fine-tuned with pseudo and with original feature. In both scenarios, using the pseudo-feature is better than training without the feature but worse than worse than the original ground truth feature values.

5.3 Comparing Supervised and Self-Supervised Pre-training

While self-supervised learning makes for transferable feature extractors in other domains, our experiments show that supervised pre-training is consistently better than the recent SSL pre-training methods we try that are designed for tabular data. In Figure 3, we compare supervised pre-training with contrastive and MLM pre-training strategies and show that supervised pre-training always attains the best average rank. Contrastive pre-training produces better results than training from scratch on the downstream data when using a linear head, but it is still inferior to supervised pre-training. Tabular MLM pretraining also falls behind the supervised strategy and performs comparably to training from scratch in the lower data regimes but leads to a weaker downstream model in the higher data regimes.

6 Aligning Upstream and Downstream Feature Sets with Pseudo-Features

While so far we have worked with upstream and downstream tasks which shared a common feature space, in the real world, tabular data is highly heterogeneous and upstream data having a different set of features from downstream data is a realistic problem (Lewinson, 2020). In our medical data scenario, downstream patient data may include additional lab tests not available for patients in the upstream data. In fact, the additional downstream feature may not even be meaningful for the upstream data, such as medical exams only applicable to downstream patients of biological sex different from the upstream patients. In this section, we propose a pseudo-feature method for aligning the upstream and downstream data and show that the data heterogeneity problem can be addressed more effectively than simply taking the intersection of the upstream and downstream feature sets for tabular transfer learning, which would throw away useful features. The proposed pseudo-feature

method is related to missing data imputation Zhang et al. (2008); Sefidian and Daneshpour (2019); Yoon et al. (2018) and self-training ideas Tanha et al. (2017). In particular, misalignment of feature sets in upstream and downstream data can be seen as an extreme scenario with features having all values missing.

Suppose our upstream data (X_u, Y_u) is missing a feature x_{new} present in the downstream data (X_d, Y_d). We then use transfer learning in stages. As the diagram on the left panel of Figure 4 shows:

1. Pre-train feature extractor $f : X_u \to Y_u$ on upstream data (X_u, Y_u) without feature x_{new}.
2. Fine-tune the feature extractor f on the downstream samples X_d to predict x_{new} as the target and obtain a model $\hat{f} : X_d \setminus \{x_{\text{new}}\} \to x_{\text{new}}$.
3. Use the model $\hat{f}$ to assign pseudo-values $\hat{x}_{\text{new}}$ of the missing feature to the upstream samples: $\hat{x}_{\text{new}} = \hat{f}(X_u)$ thus obtaining augmented upstream data $(X_u \cup \{\hat{x}_{\text{new}}\}, Y_u)$.
4. Finally, we can leverage the augmented upstream data $(X_u \cup \{\hat{x}_{\text{new}}\}, Y_u)$ to pre-train a feature extractor which we will fine-tune on the original downstream task (X_d, Y_d) using all of the available downstream features.

Similarly, in scenarios with a missing feature in downstream data, we can directly train a feature predictor on the upstream data and obtain pseudo-values for the downstream data.

This approach offers appreciable performance boosts over discarding the missing features and often performs comparably to models pre-trained with the ground truth feature values as shown in the right panel of Figure 4. The top heatmap represents the experiment where downstream data has an additional feature missing from the upstream data. The bottom heatmap represents the opposite scenario of the upstream data having an additional feature not available in the downstream data. To ensure that the features we experiment with are meaningful and contain useful information, for each task we chose important features according to GBDT feature importances. We observe that in both experiments, using the pseudo feature is better than doing transfer learning with without the missing feature at all. Additionally, we observe that in some cases, the pseudo-feature approach performs comparably to using the original ground truth feature (10-100 samples on the top heatmap and 20 samples on the bottom heatmap). Interestingly, the pseudo-feature method is more beneficial when upstream features are missing, which suggests that having ground-truth values for the additional feature in the downstream data is more important.

7 Discussion

In this paper, we demonstrate that deep tabular models with transfer learning definitively outperform strong GBDT baselines with stacking in a realistic scenario where the target downstream data is limited and auxillary upstream pre-training data is available. We highlight that representation learning with neural networks enables more effective knowledge transfer than leveraging upstream task predictions through stacking. Additionally, we present a pseudo-feature method to enable effective transfer learning in challenging cases where the upstream and downstream feature sets differ. We provide suggestions regarding architectures, hyperparameter tuning, and setups for tabular transfer learning and hope that this work serves as a guide for practitioners.

8 Ethics Statement

We conduct experiments with the MetaMIMIC medical dataset (Grzyb et al., 2021; Woźnica et al., 2022), which is based on the publicly accessible MIMIC database (Goldberger et al., 2000; Johnson et al., 2016; 2021). Regarding the patient consent to collect this data, as stated in (Johnson et al., 2016): "The project was approved by the Institutional Review Boards of Beth Israel Deaconess Medical Center (Boston, MA) and the Massachusetts Institute of Technology (Cambridge, MA). Requirement for individual patient consent was waived because the project did not impact clinical care and all protected health information was deidentified". The MIMIC database is freely available to any person upon completion of the credentialing process found at the following link: `https://mimic-iv.mit.edu/docs/access/` and is distributed under Open Data Commons Open Database License v1.0, please see the following link for details: `https://physionet.org/content/mimic-iv-demo-omop/view-license/0.9/`.

9 REPRODUCIBILITY STATEMENT

We include the code for reproducing our results in the supplementary materials. We also describe strategies for data preprocessing, training the models, choosing the best epoch as well as hyperparameter ranges in Appendix B and C.

REFERENCES

B. Afonso, L. Melo, W. Oliveira, S. Sousa, and L. Berton. Housing prices prediction with a deep learning and random forest ensemble. In *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional*, pages 389–400. SBC, 2019.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

S. Al-Stouhi and C. K. Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 60–75. Springer, 2011.

L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, and Y. Duan. Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis. *Electronics*, 9(3):427, 2020.

L. Alzubaidi, M. Al-Amidie, A. Al-Asadi, A. J. Humaidi, O. Al-Shamma, M. A. Fadhel, J. Zhang, J. Santamaría, and Y. Duan. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, 13(7):1590, 2021.

S. O. Arık and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, pages 6679–6687, 2021.

K. Arun, G. Ishan, and K. Sanmeet. Loan approval prediction based on machine learning approach. *IOSR J. Comput. Eng*, 18(3):18–21, 2016.

S. Badirli, X. Liu, Z. Xing, A. Bhowmik, K. Doan, and S. S. Keerthi. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54, 2018.

V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018a.

S. Chen, A. Qin, D. Zhou, and D. Yan. U-net-generated synthetic ct images for magnetic resonance imaging-only prostate intensity-modulated radiation therapy treatment planning. *Medical physics*, 45(12):5659–5665, 2018b.

S. Chen, K. Ma, and Y. Zheng. Med3d: Transfer learning for 3d medical image analysis. *arXiv preprint arXiv:1904.00625*, 2019.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. *arXiv preprint arXiv:2106.04803*, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. *Advances in neural information processing systems*, 14, 2001.

W. Fang, C. Chen, B. Song, L. Wang, J. Zhou, and K. Q. Zhu. Adapted tree boosting for transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 741–750. IEEE, 2019.

J. Fiedler. Simple modifications to improve tabular neural networks. *arXiv preprint arXiv:2108.03214*, 2021.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. *arXiv preprint arXiv:2106.11959*, 2021.

Y. Gorishniy, I. Rubachev, and A. Babenko. On embeddings for numerical features in tabular deep learning. *arXiv preprint arXiv:2203.05556*, 2022.

M. Grzyb, Z. Trafas, K. Woźnica, and P. Biecek. metamimic: analysis of hyperparameter transferability for tabular data using mimic-iv database, 2021. URL `https://github.com/ModelOriented/metaMIMIC/blob/main/preprint.pdf`.

H. Hazimeh, N. Ponomareva, P. Mol, Z. Tan, and R. Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

M. Heker and H. Greenspan. Joint liver lesion segmentation and classification via transfer learning. *arXiv preprint arXiv:2004.12352*, 2020.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 590–597, 2019.

V. Jain, M. Goel, and K. Shah. Deep learning on small tabular dataset: Using transfer learning and image classification. In *International Conference on Artificial Intelligence and Speech Technology*, pages 555–568. Springer, 2021.

L. Joffe. Transfer learning for tabular data. 2021.

A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark. Mimic-iv, 2021. URL `https://physionet.org/content/mimiciv/1.0/`.

A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3 (1):1–9, 2016.

A. Kadra, M. Lindauer, F. Hutter, and J. Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. *arXiv preprint arXiv:2106.11189*, 2021.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

J. Kossen, N. Band, C. Lyle, A. N. Gomez, T. Rainforth, and Y. Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

M. Y. Law and B. Liu. Dicom-rt and its utilization in radiation therapy. *Radiographics*, 29(3): 655–667, 2009.

E. Lewinson. *Python for Finance Cookbook: Over 50 recipes for applying modern Python libraries to financial data analysis*. Packt Publishing Limited, 2020.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Z. Li, D. Ding, X. Liu, P. Zhang, Y. Wu, and L. Ma. Ttnet: Tabular transfer network for few-samples prediction. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 293–301, 2021.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

I. Rubachev, A. Alekberov, Y. Gorishniy, and A. Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

B. G. Santa Cruz, M. N. Bossa, J. Sölter, and A. D. Husch. Public covid-19 x-ray datasets and their impact on model bias–a systematic review of a significant problem. *Medical image analysis*, 74: 102225, 2021.

B. Schäfl, L. Gruber, A. Bitto-Nemling, and S. Hochreiter. Hopular: Modern hopfield networks for tabular data. 2021.

A. M. Sefidian and N. Daneshpour. Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model. *Expert Systems with Applications*, 115:68–94, 2019.

A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific reports*, 9(1):1–7, 2019.

R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.11.011. URL https://www.sciencedirect.com/science/article/pii/S1566253521002360.

G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.

B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong. Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

J. Tanha, M. Van Someren, and H. Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370, 2017.

K. M. Ting and I. H. Witten. Stacked generalization: when does it work? 1997.

K. Trohidis, G. Tsoumakas, G. Kalliris, I. P. Vlahavas, et al. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

V. Turbé, C. Herbst, T. Mngomezulu, S. Meshkinfamfard, N. Dlamini, T. Mhlongo, T. Smit, V. Cherepanova, K. Shimada, J. Budd, et al. Deep learning of hiv field-based rapid tests. *Nature Medicine*, 27(7):1165–1170, 2021.

T. Ucar, E. Hajiramezanali, and L. Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34, 2021.

R. Wang, B. Fu, G. Fu, and M. Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7. 2017.

R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, and E. Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, pages 1785–1797, 2021.

Z. Wang and J. Sun. Transtab: Learning transferable tabular transformers across tables. *arXiv preprint arXiv:2205.09328*, 2022.

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

J. Wienke, D. Wigand, N. Koster, and S. Wrede. Model-based performance testing for robotics software components. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 25–32. IEEE, 2018.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. URL `http://www.jstor.org/stable/3001968`.

D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

K. Woźnica, M. Grzyb, Z. Trafas, and P. Biecek. Consolidated learning–a domain-specific model-free optimization strategy with examples for xgboost and mimic-iv. *arXiv preprint arXiv:2201.11815*, 2022.

Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

J. Yoon, J. Jordon, and M. Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pages 5689–5698. PMLR, 2018.

J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

S. Zhang, J. Zhang, X. Zhu, Y. Qin, and C. Zhang. Missing value imputation based on data clustering. In *Transactions on computational science I*, pages 128–138. Springer, 2008.

Y. Zhu, T. Brettin, F. Xia, A. Partin, M. Shukla, H. Yoo, Y. A. Evrard, J. H. Doroshow, and R. L. Stevens. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports*, 11(1):1–11, 2021.

F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A LIMITATIONS AND IMPACT

In this section we discuss limitations of our work. We note that transfer learning with deep tabular models requires more computational resources than training XGBoost or CatBoost, especially when the hyperparameter tuning is used thus incurring additional costs on practitioners adopting our approach. In addition, while we handle the cases of differing upstream and downstream feature sets thus allowing for transfer learning with heterogeneous data, our approach relies on having reasonably similar (but not identical) upstream and downstream feature sets and related upstream and downstream tasks and we do not propose a foundation model for tabular data.

B EXPERIMENTAL DETAILS.

B.1 HARDWARE

We ran our experiments on NVIDIA GeForce RTX 2080 Ti machines.

B.2 IMPLEMENTATION LICENSES

For the model implementations and training routines we adapt the code from the following publicly available repositories and libraries:

- RTDL [1] under MIT License
- TabTransformer [2] under MIT License
- Catboost [3] under Apache License
- XGBoost [4] under Apache License

B.3 DATA PREPROCESSING

We preprocess numerical features with quantile transformation with standard output distribution for neural networks and we use original features for GBDT models. Quantile transformer is first fit on upstream data and then applied to downstream data to preserve similar feature distribution in upstream and downstream data. We note, that using the same normalization for upstream and downstream data is a very important step for transfer learning. We impute missing values with mean values for numerical features and with a new category for categorical features.

B.4 TRAINING DETAILS

B.4.1 SUPERVISED PRE-TRAINING

All deep models are trained with AdamW optimizer (Loshchilov and Hutter, 2017). We pre-train models on upstream data for 500 epochs with patience set to 30, meaning that we continue training until there are 30 consecutive epochs without improvement on the validation set, but no longer than 500 epochs. Since we assume limited data availability for downstream task, we do not sample a validation set for early stopping. Instead, we fine-tune/train models from scratch for 200 epochs and choose an "optimal" epoch as discussed in Section B.4.3. We use learning rate $1e-4$ for training from scratch on downstream data and learning rate $5e-5$ for fine-tuning pre-trained models. For pre-training, learning rate and weight decay are tunable hyperparameters for each model and each pre-training dataset (collection of $n-1$ upstream tasks). Batch size was set to 256 in all transfer-learning experiments.

[1] https://github.com/Yura52/tabular-dl-revisiting-models
[2] https://github.com/lucidrains/tab-transformer-pytorch
[3] https://catboost.ai/
[4] https://xgboost.ai/

B.4.2 SELF-SUPERVISED PRE-TRAINING

MLM Pre-training

To implement MLM pre-training for tabular data, for each training sample in a batch we randomly select a feature to replace with a masking token (in the embedding space), which is unique for every feature. We also initialize a distinct fully connected layer, or head, for each column which is used to predict the masked values of that feature. We compute the appropriate loss, cross-entropy for categorical features and MSE for numerical ones, using the output from the head corresponding to the masked feature.

Our masked pre-training routine has very no additional hyperparameters. The only deviations form the pre-training hyperparameters listed above are that we use larger batch sizes of 512 and we do not employ patience. In this setting we pre-training for a full 500 epochs.

Since MLM pre-training requires training additional classification heads for every feature in the data, it dramatically increases the memory requirements. Because of that we limit the experiments with self-supervised pre-training to using the default FT-transformer configuration, including the setups we compare to, that is training from scratch and using supervised pre-training.

Contrastive Learning

We adapt the implementation of contrastive learning for tabular data from Somepalli et al. (2021). The pre-training loss consists of two components, the first is the InfoNCE contrastive loss, which minimizes the distance between the representation of two views of the same data point (original and augmented samples), while maximizing the distance in the feature space between different samples. The second component is denoising loss, which is used to train an MLP head to predict the original sample from a noisy view of that sample. For a detailed explanation of contrastive pre-training for tabular data we refer the reader to Somepalli et al. (2021). To construct positive pairs for contrastive loss and noisy samples for denoising loss we use two data aumentations: CutMix in the input space and Mixup in the embedding space Yun et al. (2019); Zhang et al. (2017).

Formally, let a batch of data be represented by $X = \{x_i\}_{i=0}^{n}$, where each x_i has q features. Let m denote a binary mask (over the features of any x_i) with each entry being a one with probability p. Then CutMix augmentation of a sample in the input space is computed as

$$x_i^{\text{CutMix}} = m \times x_i + (1 - m) \times x_j$$

where j is a random index chosen from $[0, n]$. To define Mixup in the embedding space, let $\hat{X}^{\text{CutMix}}$ be the set of the embeddings of the entire CutMix-ed batch. Then, Mixup augmentation is computed as a convex combination

$$\hat{x_i}^{\text{Mixup}} = \mu \hat{x}_i^{\text{CutMix}} + (1 - \mu) \hat{x}_k^{\text{CutMix}}$$

where k is again a random index chosen from $[0, n]$.

The hyperparameters are similar to supervised pre-training with only several modifications. First, we use smaller batch sized of 200 and we train for a full 500 epochs (no patience). Also, contrastive learning has additional hyperparameters m and μ, both of which we set to 0.9.

B.4.3 EPOCH SELECTION

Since we work with limited downstream data, sampling a validation set for early stopping is not always possible. Therefore, we select the number of fine-tuning epochs for deep models as follows. For the data levels of 4 and 10 samples, we simply select 30 fine-tuning epochs. For more data of 20 samples, we select 60 fine-tuning epochs. In the larger data levels of 100 and 200 samples, we sample 20% of the data as a validation set to perform early stopping with the flexible end-to-end fine-tuned transfer learning setups prone to overfitting. For early stopping, we terminates training if no improvement in the validation score is observed for more than 30 epochs. In the less flexible transfer learning setups with a frozen feature extractor, we select 100 fine-tuning epochs for the MLP head atop a frozen feature extractor and 200 fine-tuning epochs for the linear head atop a frozen feature extractor. Finally, for the deep baselines with the hyperparameters tuned on a small subsample of the upstream data, we select the best epoch from the small upstream subsample.

B.4.4 STACKING FOR GBDT MODELS

For XGBoost models we incorporate stacking by training 11 additional XGBoost classifiers on upstream tasks and stack their predictions as additional features for downstream data. For Catboost we apply the same strategy, but we train a single multi-label Catboost classifier predicting all 11 upstream labels.

B.4.5 STATISTICAL SIGNIFICANCE

We compute ranks taking into account statistical significance of the performance differences between the models; on a given task, top models without statistically significant performance difference all receive rank 1. To compute statistical significance we use the one-sided Wilcoxon Rank-Sum test Wilcoxon (1945); Mann and Whitney (1947) with $p = 0.05$. We run each experiment with 10 random seeds for the experiments in Section 4 and 5 random seeds for the experiments in Sections 5, 6.

C HYPERPARAMETER TUNING

In this section we provide hyperparaeter search spaces and distributions for Optuna for each model, which are adapted from the original papers. For Catboost and XGBoost models we use search spaces proposed in Gorishniy et al. (2021). We run 50 Optuna trials to tune hyperparameters for each model. In our experiments we tune the hyperparameters on full upstream data for deep tabular models with transfer learning, and on upstream data of the same size as downstream data for deep baselines trained from scratch and for GBDT models.

C.0.1 FT-TRANSFORMER

The hyperparameter search space and distributions as well as the default configuration are presented in Table 1. The number of heads is always set to 8 as recommended in the original paper.

Table 1: Optuna hyperparameter search space and default configuration for FT-Transformer

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 4]$	3
Feature embedding size	UniformInt$[64, 512]$	192
Residual dropout	$\{0,$ Uniform$[0, 0.2]\}$	0.0
Attention dropout	Uniform$[0, 0.5]$	0.2
FFN dropout	Uniform$[0, 0.5]$	0.1
FFN factor	Uniform$[2/3, 8/3]$	$4/3$
Learning rate	LogUniform$[1e-5, 1e-3]$	$1e-4$
Weight decay	LogUniform$[1e-6, 1e-3]$	$1e-5$

C.0.2 RESNET

The hyperparameter search space and distributions as well as the default configuration are presented in Table 2

C.0.3 MLP

The hyperparameter search space and distributions as well as the default configuration are presented in Table 3

C.0.4 TABTRANSFORMER

The hyperparameter search space and distributions as well as the default configuration are presented in Table 4.

Table 2: Optuna hyperparameter search space and default configuration for ResNet

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 8]$	5
Category embedding size	UniformInt$[64, 512]$	128
Layer size	UniformInt$[64, 512]$	200
Hidden factor	Uniform$[1, 4]$	3
Hidden dropout	Uniform$[0, 0.5]$	0.2
Residual dropout	$\{0$, Uniform$[0, 0.5]\}$	0.2
Learning rate	LogUniform$[1e-5, 1e-2]$	$1e-4$
Weight decay	$\{0$, LogUniform$[1e-6, 1e-3]\}$	0.0

Table 3: Optuna hyperparameter search space and default configuration for MLP

Parameter	Search Space	Default
Number of layers	UniformInt$[1, 8]$	3
Category embedding size	UniformInt$[64, 512]$	128
Layer size	UniformInt$[1, 512]$	$[300, 200, 300]$
Dropout	$\{0$, Uniform$[0, 0.5]\}$	0.2
Learning rate	LogUniform$[1e-5, 1e-2]$	$1e-4$
Weight decay	$\{0$, LogUniform$[1e-6, 1e-3]\}$	$1e-5$

Table 4: Optuna hyperparameter search space and default configuration for TabTransformer

Parameter	Search Space	Default
Number of heads	UniformInt$[2, 8]$	8
Number of layers	UniformInt$[1, 12]$	6
Category embedding size	UniformInt$[8, 128]$	32
Attention Dropout	Uniform$[0.0, 0.5]$	0.0
FF Dropout	Uniform$[0.0, 0.5]$	0.0
Learning rate	LogUniform$[1e-6, 1e-3]$	$1e-4$
Weight decay	LogUniform$[1e-6, 1e-3]$	$1e-5$

C.0.5 CATBOOST

The hyperparameter search space and distributions are presented in Table 5. For default configuration we use default parameters from the Catboost library (Prokhorenkova et al., 2018).

Table 5: Optuna hyperparameter search space for Catboost

Parameter	Search Space
Iterations	UniformInt$[2, 1000]$
Max depth	UniformInt$[3, 10]$
Learning rate	LogUniform$[1e-5, 1]$
Bagging temperature	Uniform$[0, 1]$
L2 leaf reg	LogUniform$[1, 10]$
Leaf estimation iterations	UniformInt$[1, 10]$

C.0.6 XGBOOST

The hyperparameter search space and distributions as well as the default configuration are presented in Table 6. For default configuration we use default parameters from the XGBoost library (Chen and Guestrin, 2016).

Table 6: Optuna hyperparameter search space for XGBoost

Parameter	Search Space
Num estimators	UniformInt$[2, 1000]$
Max depth	UniformInt$[3, 10]$
Min child weight	LogUniform$[1e-8, 1e5]$
Subsample	Uniform$[0.5, 1]$
Learning rate	LogUniform$[1e-5, 1]$
Col sample by level	Uniform$[0.5, 1]$
Col sample by tree	Uniform$[0.5, 1]$
Gamma	$\{0$, LogUniform$[1e-8, 1e2]\}$
Lambda	$\{0$, LogUniform$[1e-8, 1e2]\}$
Alpha	$\{0$, LogUniform$[1e-8, 1e2]\}$

C.1 Tuning GBDT Models

In Table 7 we explore the effectiveness of our hyperparameter tuning strategy for GBDT models. In particular, we compute average ranks for models with tuned and default configurations. Recall, because of the limited data availability in downstream tasks, we tune the hyperparameters on upstream data of the same size as the downstream data using full-size validation set. We find that using upstream data for tuning hyperparameters is effective for XGBoost model while for catboost tuned configuration outperforms default configuration at three out of five data availability levels.

Table 7: **Comparison of tuned and default configurations of GBDT models.** The table displays ranks computed pair-wise for default/tuned configurations of XGBoost and Catboost models.

Num samples	4	10	20	100	200
XGBoost Tuned	1.17	1.17	1.25	1.33	1.42
XGBoost Default	1.42	1.83	1.67	1.50	1.58
Catboost Tuned	1.33	1.08	1.17	1.25	1.42
Catboost Default	1.25	1.25	1.42	1.33	1.33

C.2 Tuning Deep Baselines

In Table 8 we evaluate hyperparameter tuning routine for deep baselines, i.e. deep neural networks trained from scratch. We find that unlike GBDT models, hyperparameters tuned on upstream data do not transfer to downstream tasks at lower data regimes (i.e. 4-20 samples) for deep models. However, tuning helps deep baselines in higher data regimes.

Table 8: **Comparison of tuned and default configurations of deep tabular baselines.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular neural networks.

Num samples	4	10	20	100	200
FT-Transformer Tuned	1.33	1.25	1.00	1.33	1.33
FT-Transformer Default	1.00	1.08	1.58	1.417	1.50
ResNet Tuned	1.25	1.17	1.33	1.25	1.75
ResNet Default	1.00	1.08	1.25	1.00	1.00
MLP Tuned	1.42	1.67	1.58	1.17	1.33
MLP Default	1.00	1.08	1.33	1.67	1.50
TabTransformer Tuned	1.25	1.33	1.17	1.08	1.17
TabTransformer Default	1.17	1.25	1.33	1.50	1.67

C.3 Tuning Deep Models for Transfer Learning

In Table 9 we evaluate the hyperparameter tuning strategy for deep tabular neural networks with transfer learning. Recall, we tune the hyperparameters on the full upstream data which is then used for pre-training. We find this strategy helpful for most models, especially for FT-Transformer and TabTransformer, and for most transfer learning setups.

Table 9: **Comparison of tuned and default configurations of deep tabular neural networks with transfer learning.** The table displays ranks computed pair-wise for default/tuned configurations of deep tabular models fine-tuned with 4 different transfer-learning setups.

Num Samples	4	10	20	100	200
FT-Transformer + LH-E2E Tuned	1.00	1.00	1.00	1.08	1.17
FT-Transformer + LH-E2E Default	1.25	1.25	1.17	1.42	1.17
FT-Transformer + MLP-E2E Tuned	1.00	1.00	1.08	1.00	1.08
FT-Transformer + MLP-E2E Default	1.08	1.00	1.25	1.33	1.58
FT-Transformer + LH Tuned	1.00	1.00	1.00	1.00	1.00
FT-Transformer + LH Default	1.17	1.25	1.33	1.58	1.67
FT-Transformer + MLP Tuned	1.08	1.00	1.00	1.00	1.00
FT-Transformer + MLP Default	1.17	1.08	1.33	1.58	1.75
ResNet + LH-E2E Tuned	1.00	1.17	1.33	1.25	1.33
ResNet + LH-E2E Default	1.08	1.00	1.00	1.17	1.25
ResNet + MLP-E2E Tuned	1.00	1.00	1.08	1.08	1.00
ResNet + MLP-E2E Default	1.17	1.00	1.08	1.58	1.42
ResNet + LH Tuned	1.00	1.00	1.08	1.00	1.00
ResNet + LH Default	1.17	1.25	1.25	1.50	1.50
ResNet + MLP Tuned	1.00	1.00	1.00	1.00	1.08
ResNet + MLP Default	1.17	1.17	1.33	1.58	1.75
MLP + LH-E2E Tuned	1.17	1.25	1.42	1.42	1.17
MLP + LH-E2E Default	1.08	1.08	1.08	1.33	1.50
MLP + MLP-E2E Tuned	1.00	1.08	1.33	1.25	1.42
MLP + MLP-E2E Default	1.08	1.00	1.25	1.25	1.33
MLP + LH Tuned	1.00	1.25	1.17	1.17	1.08
MLP + LH Default	1.08	1.08	1.17	1.17	1.25
MLP + MLP Tuned	1.08	1.08	1.08	1.00	1.00
MLP + MLP Default	1.00	1.00	1.17	1.17	1.42
TabTransformer + LH-E2E Tuned	1.00	1.00	1.08	1.08	1.00
TabTransformer + LH-E2E Default	1.17	1.25	1.25	1.50	1.25
TabTransformer + MLP-E2E Tuned	1.00	1.00	1.08	1.08	1.00
TabTransformer + MLP-E2E Default	1.17	1.25	1.25	1.42	1.25
TabTransformer + LH Tuned	1.00	1.00	1.00	1.00	1.00
TabTransformer + LH Default	1.25	1.17	1.25	1.25	1.17
TabTransformer + MLP Tuned	1.00	1.00	1.00	1.00	1.00
TabTransformer + MLP Default	1.25	1.33	1.25	1.25	1.25

D ADDITIONAL RESULTS

D.1 RESULTS FOR TABTRANSFORMER

Figure 5 is equivalent to Figure 2, but also includes the results for TabTransformer model, a tabular neural network consisting of transformer block for categorical features and an MLP block on top for numerical features. We find that TabTransformer performs poorly compared to other deep tabular neural networks, which might be explained by the fact that the original paper Huang et al. (2020) only suggests to tune the hyperparameters of transformer block, but not the MLP part, while Meta-MIMIC data has only one categorical features and 171 numerical features.

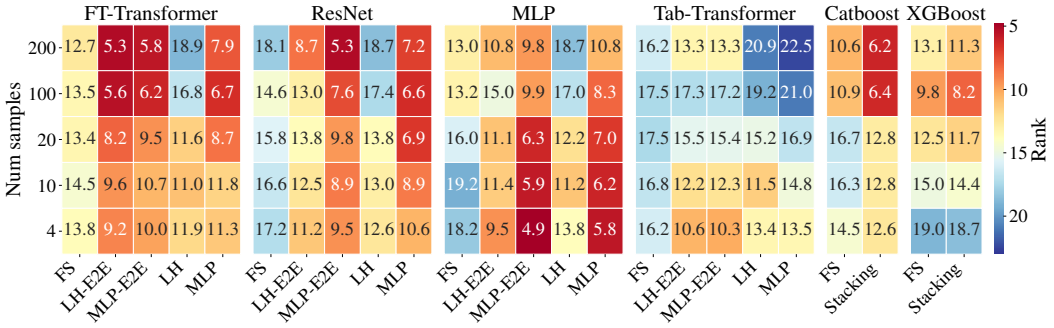

Figure 5: **Average model ranks including TabTransformer.** Deep tabular models and GBDT performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks.

D.2 MODEL-WISE RANKS

In Figure 6 we compare different transfer learning setups for each deep tabular model.

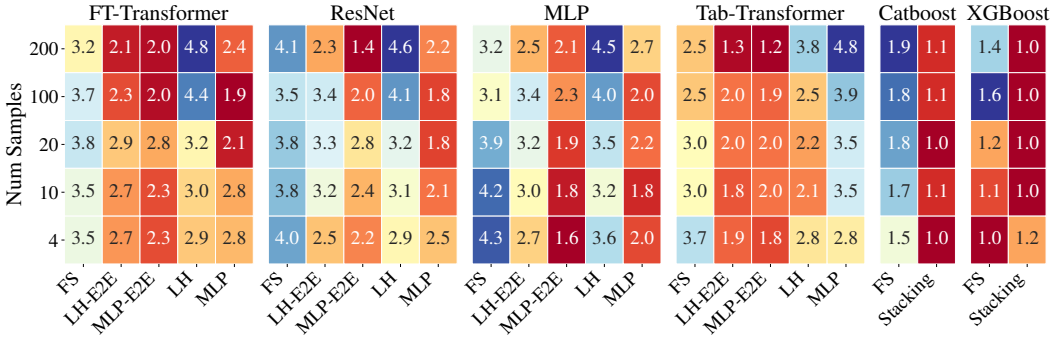

Figure 6: **Model-wise ranks for GBDT and neural networks.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is computed across training setups for each model and is averaged across all downstream tasks.

D.3 AVERAGE ROC-AUC IMPROVEMENT OVER CATBOOST

In Figure 7 we display ROC-AUC improvements over Catboost baseline averaged across all downstream tasks. We observe trends similar to ones with ranks; MLP model performs best in low data regimes, while FT-Transformer offers consistent gains across all data levels.

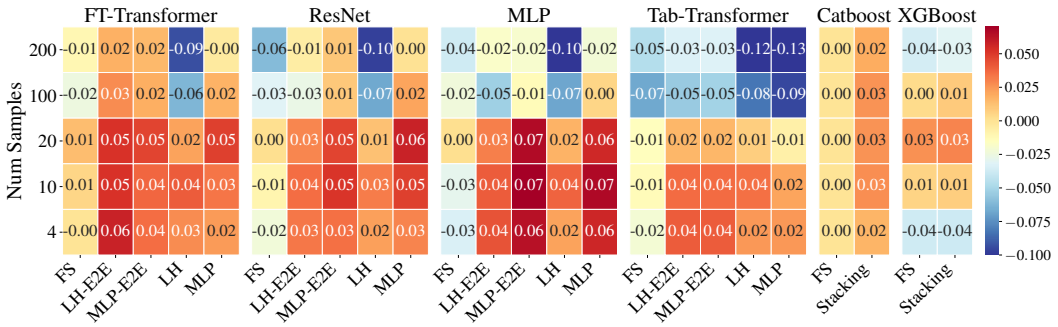

Figure 7: **Average ROC-AUC improvements over Catboost baseline.** Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. ROC-AUC improvement is computed as difference in ROC-AUC with Catboost model and is averaged across all downstream tasks.

D.4 DATASET-LEVEL RESULTS

In Tables 10, 11, 12, 13 we report ROC-AUC measurements for each model on each downstream task. We include the results for GBDT models, neural baselines and neural networks with transfer learning.

D.5 EXPERIMENTS ON ADDITIONAL DATASETS

To expand the experimental evaluation of tabular transfer learning beyond the medical domain, we include experiments on two additional multi-label datasets: Yeast functional genomics data Elisseeff and Weston (2001) and Emotions data Trohidis et al. (2008) . Yeast contains micro-array expression data and phylogenetic profiles of genes with labels corresponding to different gene functional classes. Each gene is associated with a set of functional classes, therefore each task aims to predict if a gene is associated with a particular gene functional group. The dataset contains 2417 samples (genes) with 103 numerical features and 14 labels for each sample. Emotions data aims to detect emotions in music, it contains rhythmic and timbre features of 593 songs and labels corresponding to 6 emotions. In our experimental setup, we treat each classification label as a separate task. Similarly to the experiments with MetaMIMIC, we split the tasks into downstream and upstream by reserving $n-1$ labels for the upstream task and the n-th label for the downstream task. We further limit the amount of downstream data to 4, 10, 20, 100 and 200 samples. This strategy results in 70 combinations of upstream and downstream datasets constructed from the Yeast database and 30 combinations from the Emotions database. We conduct experiments with FT-Transformer, MLP, and Catboost models which demonstrated the best results in our previous experiments. We tune the hyperparameters for deep models and Catboost as described in Appendix C. Figures 8, 9 present average model ranks of two tabular neural network and Catboost models on downstream tasks as a heatmap with warmer colors representing better results.

We observe that deep tabular models pre-trained on upstream data and finetuned with MLP head outperform models trained from scratch and Catboost models leveraging stacking. Interestingly, MLP model performs better on the Yeast dataset, while FT-Transformer performs better on the Emotions dataset.

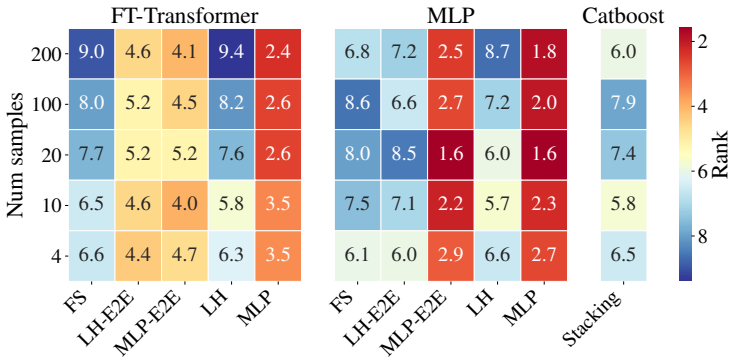

Figure 8: **Average model ranks across downstream tasks in Yeast data.** Deep tabular models and Catboost performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks.

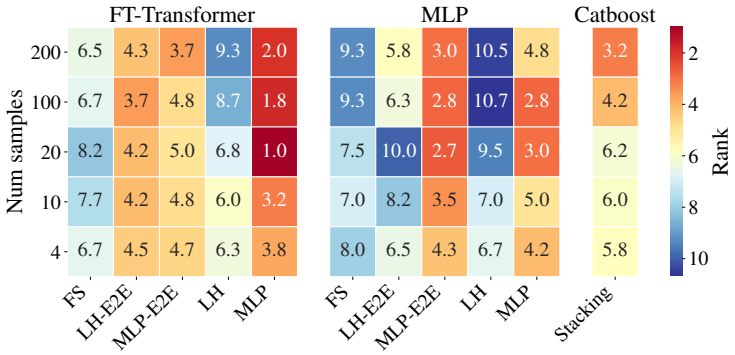

Figure 9: **Average model ranks across downstream tasks in Emotions data.** Deep tabular models and Catboost performance is presented on the corresponding panels. Within each panel, columns represent transfer learning setups, and rows correspond to the number of available downstream samples. Warmer colors indicate better performance. FS denotes training from scratch (without pre-training on upstream data), LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training. Rank is averaged across all downstream tasks.

Target Task	Diabetes					Hypertensive					Ischematic				
Num Samples	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**
FT-FS	0.511±0.008	0.540±0.004	0.567±0.004	0.596±0.009	0.802±0.010	0.453±0.010	0.477±0.007	0.528±0.006	0.658±0.008	0.722±0.005	0.620±0.009	0.656±0.004	0.644±0.011	0.678±0.005	0.740±0.006
FT-FS-2	0.514±0.008	0.550±0.006	0.570±0.003	0.666±0.006	0.779±0.013	0.464±0.006	0.482±0.007	0.525±0.004	0.666±0.005	0.701±0.004	0.655±0.002	0.656±0.005	0.596±0.011	0.740±0.003	0.759±0.002
FT-LH-E2E	0.661±0.007	**0.692**±0.005	**0.710**±0.004	0.770±0.006	0.821±0.003	0.669±0.005	0.713±0.003	0.707±0.002	0.746±0.002	0.760±0.002	0.675±0.004	0.664±0.005	0.701±0.002	0.771±0.002	0.786±0.001
FT-MLP-E2EE	0.524±0.011	0.606±0.009	0.654±0.012	0.781±0.007	**0.830**±0.004	0.648±0.009	0.721±0.003	0.709±0.003	**0.759**±0.003	**0.768**±0.002	0.702±0.006	0.704±0.008	0.708±0.004	0.781±0.002	0.807±0.001
FT-LH	**0.690**±0.005	**0.689**±0.004	**0.704**±0.005	0.700±0.005	0.702±0.005	0.658±0.008	0.713±0.005	0.708±0.004	0.714±0.003	0.716±0.003	0.637±0.003	0.679±0.002	0.688±0.001	0.716±0.002	0.726±0.002
FT-MLP	0.534±0.012	0.598±0.011	0.662±0.011	0.736±0.002	0.755±0.001	0.648±0.012	0.724±0.004	0.725±0.003	**0.760**±0.002	**0.767**±0.001	0.691±0.006	0.707±0.007	0.740±0.003	0.792±0.001	0.813±0.001
ResNet-FS	0.498±0.014	0.481±0.011	0.501±0.006	0.627±0.009	0.677±0.006	0.464±0.011	0.492±0.012	0.568±0.004	0.666±0.003	0.689±0.003	0.608±0.008	0.622±0.006	0.631±0.009	0.707±0.003	0.726±0.003
ResNet-FS-2	0.484±0.008	0.514±0.005	0.539±0.006	0.677±0.003	0.776±0.002	0.500±0.007	0.494±0.008	0.538±0.006	0.644±0.007	0.724±0.002	0.627±0.006	0.646±0.005	0.631±0.003	0.736±0.002	0.752±0.002
ResNet-LH-E2E	0.643±0.003	0.660±0.003	0.686±0.002	0.705±0.002	0.789±0.001	0.684±0.004	0.683±0.002	0.697±0.001	0.681±0.002	0.762±0.001	0.631±0.003	0.663±0.002	0.687±0.003	0.767±0.001	0.778±0.001
ResNet-MLP-E2E	0.445±0.008	0.542±0.009	0.616±0.007	0.633±0.009	0.775±0.002	**0.704**±0.009	0.719±0.003	0.716±0.002	0.737±0.003	**0.767**±0.001	0.677±0.008	0.724±0.006	0.715±0.005	0.786±0.003	0.814±0.001
ResNet-LH	0.672±0.003	0.649±0.003	0.692±0.002	0.700±0.002	0.709±0.002	0.686±0.004	0.719±0.002	0.719±0.001	0.729±0.001	0.738±0.001	0.581±0.003	0.636±0.003	0.678±0.002	0.707±0.002	0.712±0.003
ResNet-MLP	0.457±0.007	0.546±0.009	0.635±0.009	0.720±0.002	0.752±0.001	**0.704**±0.009	0.730±0.002	**0.739**±0.002	0.757±0.001	**0.768**±0.001	0.663±0.009	0.715±0.007	0.752±0.004	**0.799**±0.001	**0.818**±0.001
MLP-FS	0.503±0.007	0.491±0.005	0.515±0.002	0.652±0.003	0.706±0.002	0.442±0.003	0.470±0.004	0.578±0.001	0.670±0.002	0.699±0.002	0.648±0.004	0.643±0.008	0.626±0.002	0.728±0.002	0.760±0.001
MLP-FS-2	0.507±0.005	0.529±0.006	0.543±0.004	0.550±0.012	0.671±0.005	0.500±0.010	0.492±0.011	0.518±0.005	0.638±0.004	0.692±0.002	0.639±0.004	0.667±0.005	0.668±0.003	0.729±0.003	0.761±0.003
MLP-LH-E2E	0.675±0.002	0.682±0.002	0.689±0.002	0.698±0.002	0.766±0.002	0.661±0.003	0.674±0.002	0.682±0.001	0.692±0.002	0.753±0.001	0.657±0.003	0.662±0.003	0.689±0.002	0.726±0.002	0.785±0.001
MLP-MLP-E2E	0.536±0.005	0.589±0.004	0.626±0.004	0.618±0.008	0.623±0.006	**0.706**±0.006	**0.736**±0.002	0.732±0.002	0.753±0.002	0.753±0.002	**0.713**±0.004	**0.745**±0.004	**0.768**±0.002	0.788±0.001	0.811±0.000
MLP-LH	0.685±0.002	0.665±0.002	0.697±0.002	0.701±0.002	0.707±0.002	0.658±0.003	0.711±0.001	0.704±0.002	0.718±0.001	0.724±0.001	0.626±0.004	0.671±0.002	0.675±0.002	0.696±0.003	0.699±0.003
MLP-MLP	0.533±0.005	0.590±0.004	0.627±0.003	0.719±0.001	0.744±0.001	**0.706**±0.006	**0.738**±0.002	**0.739**±0.002	**0.762**±0.001	0.764±0.001	**0.715**±0.005	**0.738**±0.004	0.758±0.002	0.789±0.001	0.807±0.001
Tab-FS	0.488±0.007	0.487±0.002	0.517±0.003	0.616±0.004	0.691±0.004	0.496±0.007	0.465±0.004	0.570±0.004	0.667±0.004	0.701±0.003	0.461±0.006	0.659±0.003	0.620±0.004	0.711±0.002	0.739±0.002
Tab-FS-2	0.495±0.007	0.507±0.004	0.541±0.004	0.563±0.012	0.713±0.003	0.513±0.005	0.496±0.011	0.538±0.003	0.564±0.007	0.659±0.004	0.472±0.003	0.632±0.011	0.571±0.006	0.721±0.001	0.746±0.001
Tab-LH-E2E	**0.693**±0.006	**0.692**±0.006	0.694±0.005	0.704±0.006	0.727±0.005	0.658±0.004	0.665±0.004	0.685±0.003	0.659±0.004	0.727±0.003	0.664±0.005	0.668±0.005	0.675±0.004	0.708±0.004	0.772±0.002
Tab-MLP-E2E	**0.693**±0.006	**0.692**±0.006	0.694±0.005	0.704±0.006	0.725±0.005	0.658±0.004	0.665±0.004	0.685±0.003	0.659±0.004	0.727±0.003	0.664±0.005	0.668±0.005	0.675±0.004	0.706±0.004	0.770±0.002
Tab-LH	**0.698**±0.006	**0.691**±0.005	**0.703**±0.005	0.705±0.005	0.705±0.006	0.653±0.005	0.684±0.004	0.678±0.004	0.666±0.004	0.664±0.004	0.625±0.006	0.652±0.007	0.655±0.006	0.668±0.006	0.664±0.006
Tab-MLP	**0.698**±0.006	**0.698**±0.006	**0.703**±0.006	0.703±0.006	0.702±0.006	0.653±0.005	0.658±0.004	0.657±0.004	0.656±0.004	0.654±0.004	0.625±0.006	0.624±0.006	0.628±0.006	0.642±0.006	0.639±0.006
Catboost	0.495±0.004	0.514±0.008	0.585±0.003	0.782±0.001	0.797±0.001	0.524±0.005	0.501±0.004	0.491±0.001	0.721±0.001	0.738±0.001	0.657±0.004	0.641±0.005	0.651±0.003	0.735±0.001	0.734±0.001
Catboost+stacking	0.507±0.004	0.555±0.006	0.623±0.004	**0.807**±0.002	**0.826**±0.001	0.604±0.012	0.602±0.008	0.576±0.014	0.732±0.001	0.756±0.001	0.671±0.004	0.688±0.010	0.734±0.004	0.768±0.002	0.766±0.002
XGBoost	0.532±0.006	0.525±0.003	0.593±0.006	**0.805**±0.001	0.824±0.000	0.466±0.003	0.514±0.005	0.514±0.004	0.733±0.001	0.756±0.001	0.619±0.006	0.656±0.003	0.699±0.003	0.747±0.001	0.752±0.000
XGBoost+stacking	0.524±0.011	0.529±0.004	0.599±0.005	**0.806**±0.000	**0.826**±0.001	0.458±0.003	0.510±0.004	0.517±0.005	0.740±0.001	0.759±0.000	0.628±0.007	0.659±0.003	0.722±0.004	0.770±0.001	0.776±0.001

Table 10: **ROC-AUC scores for all models for "Diabetes", "Hypertensive" and "Ischematic" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Heart					Overweight					Anemia				
Num Samples	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**
FT-FS	0.532±0.013	0.557±0.009	0.653±0.005	0.687±0.008	0.746±0.003	0.516±0.012	0.506±0.003	0.623±0.006	0.582±0.015	**0.822**±0.009	0.643±0.013	0.709±0.003	0.741±0.006	0.764±0.003	0.722±0.007
FT-FS-2	0.517±0.005	0.589±0.004	0.571±0.011	0.762±0.002	0.748±0.003	0.491±0.008	0.490±0.009	0.610±0.004	0.638±0.005	0.679±0.004	**0.728**±0.006	0.749±0.002	0.735±0.006	0.782±0.001	**0.794**±0.002
FT-LH-E2E	0.589±0.002	0.642±0.004	0.667±0.002	0.770±0.002	0.782±0.002	0.673±0.004	0.620±0.008	**0.728**±0.012	**0.759**±0.015	**0.818**±0.004	0.639±0.003	0.646±0.003	0.665±0.004	0.756±0.001	0.762±0.001
FT-MLP-E2EE	0.542±0.008	0.642±0.007	0.658±0.002	0.768±0.005	0.784±0.003	0.540±0.015	0.469±0.011	0.581±0.024	0.720±0.018	0.768±0.013	0.662±0.009	0.682±0.004	0.725±0.003	0.750±0.002	0.773±0.001
FT-LH	**0.626**±0.003	0.664±0.002	0.672±0.002	0.689±0.003	0.713±0.002	0.669±0.002	0.675±0.003	0.682±0.002	0.675±0.002	0.674±0.002	0.557±0.003	0.580±0.003	0.597±0.004	0.610±0.003	0.617±0.003
FT-MLP	0.516±0.016	0.640±0.014	0.667±0.005	**0.777**±0.001	**0.790**±0.001	0.532±0.015	0.469±0.016	0.507±0.016	0.656±0.006	0.718±0.004	0.610±0.013	0.642±0.006	0.687±0.003	0.705±0.003	0.712±0.003
ResNet-FS	0.489±0.028	0.532±0.012	0.644±0.006	0.683±0.004	0.730±0.005	0.515±0.015	0.502±0.011	0.642±0.006	0.642±0.008	0.665±0.007	0.553±0.019	0.653±0.009	0.698±0.004	0.750±0.003	0.767±0.002
ResNet-FS-2	0.497±0.020	0.551±0.010	0.650±0.004	0.750±0.004	0.772±0.002	0.495±0.009	0.504±0.008	0.615±0.004	0.666±0.004	0.739±0.003	0.644±0.019	0.752±0.004	0.747±0.002	0.757±0.002	0.780±0.001
ResNet-LH-E2E	0.587±0.005	0.609±0.004	0.644±0.002	0.656±0.002	0.760±0.002	**0.699**±0.003	**0.691**±0.003	0.700±0.002	0.714±0.001	0.770±0.002	0.607±0.003	0.652±0.003	0.665±0.002	0.736±0.001	0.748±0.001
ResNet-MLP-E2E	0.497±0.011	0.618±0.008	0.641±0.004	0.761±0.002	**0.789**±0.001	0.547±0.008	0.545±0.011	0.568±0.013	0.674±0.009	0.765±0.005	0.608±0.009	0.697±0.008	0.727±0.004	0.760±0.001	0.767±0.002
ResNet-LH	0.612±0.005	0.657±0.004	0.668±0.002	0.661±0.003	0.677±0.003	**0.698**±0.003	0.683±0.003	0.679±0.002	0.687±0.001	0.685±0.001	0.563±0.004	0.593±0.004	0.594±0.003	0.635±0.003	0.639±0.003
ResNet-MLP	0.488±0.012	0.627±0.008	**0.683**±0.003	0.768±0.002	0.783±0.001	0.548±0.009	0.544±0.012	0.525±0.010	0.669±0.004	0.726±0.002	0.582±0.009	0.666±0.011	0.710±0.004	0.739±0.002	0.739±0.003
MLP-FS	0.426±0.007	0.512±0.019	0.646±0.002	0.710±0.004	0.765±0.001	0.533±0.004	0.499±0.008	0.635±0.003	0.651±0.005	0.698±0.002	0.641±0.006	0.689±0.002	0.708±0.000	0.763±0.002	**0.796**±0.001
MLP-FS-2	0.570±0.010	0.643±0.006	0.669±0.003	0.756±0.002	0.751±0.001	0.528±0.008	0.489±0.005	0.573±0.003	0.566±0.006	0.646±0.003	**0.733**±0.005	0.753±0.003	0.738±0.001	0.768±0.001	0.790±0.001
MLP-LH-E2E	**0.626**±0.003	0.644±0.003	0.665±0.002	0.677±0.002	0.766±0.001	0.680±0.002	0.675±0.002	0.697±0.002	0.683±0.002	0.718±0.002	0.610±0.003	0.615±0.003	0.657±0.002	0.706±0.002	0.749±0.001
MLP-MLP-E2E	**0.629**±0.007	**0.692**±0.003	**0.685**±0.002	0.719±0.003	0.780±0.001	0.599±0.005	0.574±0.008	0.609±0.004	0.614±0.004	0.626±0.003	0.690±0.004	0.695±0.004	0.723±0.002	0.757±0.001	0.758±0.001
MLP-LH	0.610±0.003	0.650±0.002	0.656±0.002	0.665±0.003	0.679±0.003	0.675±0.002	0.688±0.001	0.685±0.001	0.685±0.002	0.686±0.002	0.551±0.004	0.567±0.004	0.592±0.004	0.597±0.003	0.600±0.004
MLP-MLP	**0.620**±0.006	0.681±0.003	**0.684**±0.002	0.755±0.002	0.767±0.001	0.597±0.005	0.576±0.008	0.582±0.004	0.646±0.003	0.669±0.003	0.655±0.005	0.666±0.004	0.694±0.003	0.703±0.003	0.699±0.002
Tab-FS	0.501±0.009	0.545±0.002	0.645±0.003	0.731±0.003	0.741±0.004	0.510±0.012	0.506±0.003	0.582±0.005	0.578±0.002	0.654±0.005	0.548±0.009	0.677±0.003	0.708±0.003	0.762±0.002	0.770±0.001
Tab-FS-2	0.517±0.006	0.565±0.006	0.614±0.004	0.709±0.006	0.736±0.004	0.518±0.009	0.499±0.007	0.547±0.004	0.564±0.012	0.612±0.008	0.602±0.004	0.689±0.006	0.729±0.002	0.727±0.005	0.779±0.001
Tab-LH-E2E	0.602±0.006	0.613±0.005	0.636±0.004	0.661±0.006	0.760±0.004	0.676±0.004	0.679±0.004	0.684±0.004	0.684±0.003	0.726±0.003	0.594±0.004	0.603±0.004	0.650±0.003	0.701±0.004	0.742±0.003
Tab-MLP-E2E	0.602±0.006	0.613±0.005	0.636±0.004	0.705±0.005	0.757±0.004	0.676±0.004	0.679±0.004	0.684±0.004	0.684±0.003	0.733±0.003	0.594±0.005	0.603±0.004	0.650±0.003	0.698±0.004	0.745±0.003
Tab-LH	0.599±0.007	0.619±0.006	0.621±0.006	0.631±0.006	0.636±0.007	0.673±0.004	0.681±0.004	0.681±0.004	0.679±0.004	0.679±0.004	0.561±0.005	0.567±0.006	0.591±0.004	0.592±0.004	0.590±0.003
Tab-MLP	0.599±0.007	0.595±0.008	0.600±0.007	0.616±0.007	0.617±0.007	0.673±0.004	0.674±0.004	0.674±0.004	0.676±0.004	0.676±0.004	0.561±0.005	0.561±0.006	0.572±0.004	0.576±0.004	0.574±0.004
Catboost	0.554±0.005	0.581±0.004	0.652±0.001	0.712±0.001	0.744±0.001	0.605±0.007	0.499±0.008	0.625±0.004	0.764±0.002	0.765±0.001	0.560±0.009	**0.765**±0.003	0.760±0.001	0.775±0.001	0.778±0.000
Catboost+stacking	0.555±0.005	0.624±0.009	0.674±0.001	0.756±0.001	0.777±0.001	0.614±0.006	0.539±0.007	0.672±0.005	**0.792**±0.002	0.797±0.001	0.557±0.009	0.754±0.003	0.763±0.001	0.781±0.001	0.783±0.000
XGBoost	0.470±0.016	0.564±0.004	0.646±0.005	0.735±0.001	0.757±0.001	0.500±0.005	0.540±0.004	**0.715**±0.010	0.789±0.001	0.766±0.001	0.527±0.015	0.754±0.001	**0.773**±0.002	**0.788**±0.000	0.783±0.000
XGBoost+stacking	0.458±0.014	0.567±0.004	0.649±0.004	0.742±0.001	0.767±0.001	0.511±0.008	0.538±0.005	**0.711**±0.009	**0.789**±0.001	0.767±0.002	0.504±0.009	0.754±0.002	**0.772**±0.002	**0.788**±0.000	0.783±0.000

Table 11: **ROC-AUC scores for all models for "Heart", "Overweight" and "Anemia" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Respiratory					Hypotension					Lipoid				
Num Samples	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**
FT-FS	0.500±0.009	0.475±0.005	0.514±0.006	0.572±0.009	0.561±0.004	0.526±0.008	0.518±0.009	0.546±0.006	0.601±0.009	0.644±0.002	0.502±0.006	0.552±0.011	0.564±0.006	0.640±0.012	0.675±0.004
FT-FS-2	0.495±0.005	0.469±0.005	0.484±0.006	0.562±0.010	0.577±0.003	0.574±0.003	0.552±0.005	0.539±0.012	0.546±0.011	0.641±0.001	0.520±0.004	0.530±0.002	0.574±0.003	0.598±0.003	0.660±0.004
FT-LH-E2E	0.545±0.004	0.498±0.004	0.517±0.002	0.578±0.002	0.585±0.001	0.564±0.002	0.564±0.003	0.584±0.002	0.592±0.002	0.635±0.001	0.584±0.006	0.565±0.008	0.630±0.004	0.727±0.002	**0.745**±0.002
FT-MLP-E2EE	0.531±0.004	0.462±0.007	0.496±0.006	0.576±0.004	0.562±0.004	0.595±0.002	0.595±0.004	0.612±0.001	0.627±0.002	0.647±0.002	0.429±0.009	0.475±0.012	0.633±0.003	0.714±0.004	0.737±0.001
FT-LH	0.556±0.002	0.536±0.002	0.543±0.002	0.565±0.002	0.564±0.002	0.511±0.002	0.533±0.002	0.545±0.002	0.538±0.002	0.553±0.002	**0.631**±0.004	0.582±0.006	0.655±0.003	0.707±0.001	0.724±0.001
FT-MLP	0.547±0.004	0.490±0.008	0.518±0.004	0.575±0.002	0.569±0.001	0.590±0.003	0.593±0.007	0.604±0.002	0.612±0.001	0.637±0.001	0.415±0.009	0.531±0.014	0.689±0.002	0.735±0.002	0.740±0.001
ResNet-FS	0.499±0.007	0.500±0.004	0.520±0.010	0.564±0.004	0.550±0.004	0.516±0.008	0.502±0.011	0.506±0.011	0.572±0.006	0.578±0.011	0.503±0.007	0.574±0.003	0.581±0.004	0.595±0.004	0.649±0.002
ResNet-FS-2	0.479±0.008	0.460±0.005	0.465±0.004	0.565±0.003	0.586±0.003	0.552±0.010	0.535±0.006	0.539±0.004	0.589±0.004	0.638±0.002	0.505±0.009	0.551±0.007	0.574±0.003	0.523±0.018	0.664±0.003
ResNet-LH-E2E	0.536±0.003	0.530±0.002	0.517±0.002	0.577±0.001	0.581±0.001	0.521±0.005	0.535±0.004	0.531±0.003	0.547±0.002	0.590±0.001	0.561±0.004	0.571±0.005	0.610±0.004	0.635±0.002	0.725±0.001
ResNet-MLP-E2E	0.524±0.006	0.521±0.003	0.511±0.003	0.583±0.003	0.578±0.002	**0.614**±0.004	0.603±0.004	0.601±0.004	0.624±0.003	0.638±0.001	0.442±0.006	0.545±0.010	0.657±0.005	0.720±0.003	**0.744**±0.001
ResNet-LH	0.552±0.002	**0.551**±0.001	**0.550**±0.001	0.570±0.001	0.565±0.001	0.503±0.004	0.543±0.004	0.527±0.003	0.524±0.003	0.535±0.003	0.563±0.005	0.552±0.007	0.620±0.004	0.697±0.002	0.714±0.002
ResNet-MLP	0.520±0.006	0.532±0.003	0.530±0.003	0.573±0.001	0.566±0.001	**0.616**±0.004	0.603±0.004	0.621±0.002	0.626±0.001	0.649±0.001	0.442±0.008	0.551±0.014	0.684±0.005	**0.742**±0.001	**0.746**±0.001
MLP-FS	0.492±0.006	0.494±0.004	0.503±0.006	0.564±0.003	0.576±0.002	0.504±0.004	0.513±0.005	0.529±0.005	0.621±0.001	0.648±0.002	0.507±0.006	0.531±0.015	0.574±0.005	0.598±0.004	0.559±0.034
MLP-FS-2	0.500±0.003	0.447±0.004	0.458±0.003	0.542±0.004	0.574±0.002	0.568±0.003	0.545±0.005	0.587±0.002	0.571±0.007	**0.652**±0.001	0.510±0.005	0.577±0.005	0.574±0.002	0.633±0.003	0.665±0.003
MLP-LH-E2E	0.545±0.002	0.535±0.001	0.531±0.001	0.578±0.001	**0.591**±0.001	0.532±0.002	0.530±0.002	0.560±0.002	0.563±0.001	0.595±0.001	0.602±0.003	**0.607**±0.003	0.624±0.002	0.624±0.003	0.713±0.001
MLP-MLP-E2E	**0.564**±0.003	0.533±0.005	0.535±0.003	0.583±0.002	0.583±0.002	**0.608**±0.004	**0.628**±0.002	**0.633**±0.001	**0.635**±0.001	**0.654**±0.000	0.430±0.009	0.592±0.007	0.684±0.004	0.708±0.003	0.741±0.001
MLP-LH	0.551±0.002	0.544±0.002	**0.553**±0.001	0.567±0.002	0.567±0.002	0.513±0.003	0.550±0.002	0.557±0.002	0.548±0.002	0.555±0.002	0.608±0.003	0.577±0.004	0.644±0.003	0.696±0.001	0.707±0.001
MLP-MLP	**0.571**±0.003	**0.545**±0.005	**0.554**±0.003	0.577±0.001	0.575±0.002	**0.611**±0.004	**0.629**±0.002	**0.631**±0.001	0.628±0.001	0.645±0.001	0.429±0.009	0.594±0.007	**0.701**±0.004	**0.740**±0.001	0.743±0.001
Tab-FS	0.463±0.002	0.459±0.003	0.472±0.004	0.525±0.010	0.567±0.003	0.547±0.004	0.535±0.006	0.544±0.007	0.561±0.008	0.635±0.001	0.536±0.002	0.510±0.005	0.505±0.003	0.538±0.007	0.656±0.002
Tab-FS-2	0.461±0.004	0.454±0.003	0.458±0.002	0.494±0.008	0.526±0.003	0.541±0.004	0.525±0.006	0.540±0.002	0.566±0.008	0.632±0.001	0.537±0.003	0.489±0.005	0.487±0.002	0.513±0.009	0.561±0.009
Tab-LH-E2E	0.531±0.004	0.522±0.004	0.504±0.003	0.551±0.004	0.582±0.002	0.526±0.004	0.523±0.004	0.540±0.003	0.544±0.003	0.577±0.004	0.615±0.005	**0.614**±0.006	0.611±0.006	0.625±0.006	0.670±0.007
Tab-MLP-E2E	0.531±0.004	0.522±0.004	0.504±0.003	0.552±0.004	0.582±0.002	0.526±0.004	0.524±0.004	0.540±0.003	0.544±0.003	0.572±0.004	0.615±0.005	**0.614**±0.006	0.611±0.006	0.625±0.006	0.668±0.007
Tab-LH	0.550±0.005	**0.548**±0.004	**0.551**±0.004	0.562±0.004	0.562±0.004	0.511±0.004	0.534±0.004	0.537±0.004	0.533±0.004	0.537±0.004	0.617±0.006	0.595±0.006	0.619±0.007	0.647±0.008	0.651±0.008
Tab-MLP	0.550±0.005	**0.547**±0.005	**0.552**±0.004	0.558±0.004	0.558±0.004	0.511±0.004	0.512±0.004	0.520±0.004	0.523±0.004	0.526±0.004	0.617±0.006	**0.618**±0.006	0.621±0.006	0.634±0.007	0.635±0.007
Catboost	0.482±0.001	0.492±0.003	0.496±0.003	0.554±0.002	0.572±0.002	0.523±0.001	0.490±0.004	0.531±0.005	0.552±0.001	0.635±0.001	0.523±0.014	0.536±0.002	0.547±0.005	0.618±0.006	0.693±0.001
Catboost+stacking	0.488±0.003	0.498±0.002	0.503±0.003	**0.592**±0.002	0.584±0.001	0.568±0.002	0.501±0.004	0.545±0.004	0.583±0.001	0.643±0.001	0.515±0.017	0.536±0.002	0.555±0.007	0.669±0.005	0.718±0.002
XGBoost	0.493±0.004	0.493±0.003	0.494±0.001	0.571±0.001	0.496±0.003	0.507±0.004	0.509±0.002	0.553±0.001	0.539±0.001	0.517±0.009	0.390±0.008	0.483±0.004	0.539±0.001	0.558±0.002	0.695±0.001
XGBoost+stacking	0.499±0.005	0.495±0.004	0.497±0.002	0.578±0.001	0.499±0.006	0.507±0.006	0.511±0.003	0.553±0.002	0.543±0.001	0.516±0.008	0.394±0.006	0.487±0.005	0.545±0.002	0.576±0.001	0.723±0.001

Table 12: **ROC-AUC scores for all models for "Respiratory", "Hypotension" and "Lipoid" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.

Dataset	Atrial					Purpura					Alcohol				
Num Samples	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**	**4**	**10**	**20**	**100**	**200**
FT-FS	0.603±0.008	0.594±0.015	0.617±0.007	0.702±0.008	0.690±0.008	0.711±0.007	0.703±0.010	0.737±0.005	**0.870**±0.007	0.851±0.009	0.593±0.008	0.570±0.010	0.600±0.005	0.666±0.006	0.640±0.005
FT-FS-2	0.606±0.003	0.628±0.003	0.382±0.005	0.665±0.005	0.703±0.001	**0.763**±0.003	**0.745**±0.003	0.451±0.012	0.778±0.004	0.852±0.006	0.603±0.004	0.584±0.003	0.588±0.007	0.635±0.006	0.633±0.005
FT-LH-E2E	0.579±0.005	0.603±0.005	0.604±0.004	0.725±0.002	**0.788**±0.002	0.643±0.002	0.641±0.002	0.664±0.004	0.784±0.005	0.777±0.005	0.585±0.005	0.573±0.004	0.616±0.005	0.667±0.003	0.700±0.005
FT-MLP-E2EE	0.605±0.009	0.627±0.006	0.636±0.005	0.753±0.002	0.752±0.002	0.705±0.007	0.672±0.006	0.695±0.005	0.629±0.014	0.798±0.003	0.658±0.007	0.589±0.006	0.622±0.003	0.677±0.004	0.700±0.002
FT-LH	0.543±0.003	0.568±0.003	0.577±0.002	0.610±0.002	0.612±0.002	0.511±0.002	0.538±0.002	0.543±0.002	0.561±0.004	0.552±0.004	0.462±0.003	0.520±0.003	0.513±0.004	0.477±0.004	0.475±0.004
FT-MLP	0.587±0.015	0.637±0.016	**0.667**±0.007	0.766±0.002	0.772±0.001	0.647±0.010	0.563±0.014	0.615±0.008	0.715±0.004	0.723±0.003	0.685±0.005	**0.606**±0.011	**0.666**±0.006	0.685±0.003	0.703±0.003
ResNet-FS	0.595±0.007	0.570±0.009	0.599±0.006	0.691±0.004	0.625±0.019	0.643±0.007	0.708±0.007	0.689±0.022	0.755±0.007	0.659±0.021	0.560±0.015	0.550±0.004	0.611±0.009	0.680±0.006	0.700±0.007
ResNet-FS-2	0.585±0.010	0.573±0.009	0.629±0.005	0.705±0.002	0.749±0.002	0.684±0.010	0.702±0.006	0.730±0.004	0.769±0.004	0.775±0.003	0.567±0.011	0.573±0.006	0.616±0.006	**0.702**±0.003	**0.745**±0.003
ResNet-LH-E2E	0.609±0.006	0.523±0.004	0.587±0.002	0.681±0.001	0.758±0.001	0.562±0.006	0.599±0.004	0.618±0.002	0.687±0.002	0.715±0.002	0.473±0.004	0.504±0.003	0.576±0.002	0.603±0.003	0.704±0.003
ResNet-MLP-E2E	**0.671**±0.013	0.627±0.010	0.650±0.005	0.732±0.005	0.748±0.005	0.687±0.009	0.652±0.011	0.697±0.006	0.729±0.005	0.761±0.004	**0.715**±0.007	**0.621**±0.009	0.630±0.005	0.689±0.003	0.720±0.002
ResNet-LH	0.582±0.007	0.557±0.003	0.570±0.002	0.587±0.002	0.595±0.002	0.533±0.008	0.560±0.005	0.535±0.003	0.542±0.004	0.546±0.004	0.436±0.005	0.471±0.005	0.488±0.005	0.464±0.003	0.463±0.003
ResNet-MLP	**0.669**±0.014	0.625±0.009	**0.677**±0.004	**0.770**±0.002	0.766±0.002	0.665±0.011	0.617±0.014	0.659±0.008	0.740±0.003	0.751±0.003	**0.719**±0.008	**0.617**±0.009	0.649±0.003	0.661±0.002	0.689±0.002
MLP-FS	0.535±0.015	0.529±0.010	0.603±0.005	0.704±0.001	0.707±0.001	0.559±0.026	0.570±0.022	0.689±0.007	0.754±0.002	0.749±0.003	0.518±0.013	0.504±0.011	0.584±0.009	0.618±0.002	0.632±0.003
MLP-FS-2	0.588±0.004	0.594±0.008	0.635±0.003	0.577±0.010	0.726±0.001	**0.764**±0.003	0.696±0.006	0.733±0.003	0.716±0.006	0.750±0.002	0.514±0.008	0.505±0.006	0.531±0.005	0.626±0.003	0.556±0.005
MLP-LH-E2E	0.563±0.004	0.584±0.004	0.606±0.003	0.646±0.003	0.733±0.002	0.594±0.003	0.595±0.002	0.642±0.003	0.595±0.003	0.723±0.002	0.483±0.003	0.473±0.002	0.496±0.002	0.492±0.002	0.639±0.002
MLP-MLP-E2E	**0.662**±0.007	**0.689**±0.003	**0.681**±0.001	0.714±0.001	0.753±0.001	0.699±0.004	0.688±0.003	0.725±0.002	0.692±0.002	0.772±0.001	0.639±0.008	0.517±0.006	0.555±0.004	0.581±0.004	0.658±0.003
MLP-LH	0.535±0.005	0.595±0.004	0.588±0.004	0.598±0.004	0.593±0.004	0.503±0.003	0.561±0.003	0.564±0.004	0.565±0.004	0.563±0.004	0.460±0.004	0.473±0.002	0.459±0.002	0.457±0.002	0.454±0.003
MLP-MLP	**0.653**±0.007	**0.682**±0.003	0.679±0.002	0.745±0.001	0.740±0.002	0.660±0.005	0.647±0.003	0.662±0.003	0.728±0.002	0.738±0.002	0.642±0.008	0.501±0.007	0.523±0.006	0.580±0.006	0.590±0.006
Tab-FS	0.593±0.007	0.563±0.007	0.608±0.009	0.608±0.011	0.696±0.003	0.688±0.007	0.697±0.007	0.672±0.014	0.673±0.016	0.672±0.006	0.626±0.004	0.536±0.006	0.552±0.016	0.536±0.012	0.672±0.003
Tab-FS-2	0.530±0.004	0.547±0.004	0.583±0.002	0.688±0.002	0.692±0.002	0.649±0.003	0.681±0.009	0.661±0.004	0.654±0.003	0.723±0.003	0.623±0.007	**0.573**±0.019	0.459±0.003	0.558±0.025	0.666±0.005
Tab-LH-E2E	0.561±0.008	0.573±0.007	0.591±0.005	0.607±0.008	0.716±0.005	0.572±0.005	0.579±0.005	0.617±0.004	0.599±0.006	0.680±0.005	0.506±0.007	0.495±0.006	0.476±0.005	0.614±0.006	0.666±0.006
Tab-MLP-E2E	0.561±0.008	0.573±0.007	0.591±0.005	0.607±0.008	0.720±0.005	0.572±0.005	0.579±0.005	0.617±0.004	0.599±0.006	0.687±0.005	0.506±0.007	0.495±0.006	0.476±0.005	0.613±0.006	0.668±0.006
Tab-LH	0.540±0.008	0.578±0.008	0.575±0.007	0.587±0.008	0.584±0.008	0.508±0.005	0.565±0.006	0.562±0.005	0.557±0.005	0.552±0.006	0.473±0.008	0.502±0.009	0.489±0.009	0.494±0.009	0.489±0.008
Tab-MLP	0.540±0.008	0.555±0.008	0.557±0.008	0.568±0.008	0.567±0.008	0.508±0.005	0.529±0.005	0.535±0.005	0.537±0.005	0.534±0.005	0.473±0.008	0.474±0.008	0.467±0.007	0.484±0.008	0.478±0.008
Catboost	0.598±0.001	0.588±0.007	0.541±0.004	0.624±0.001	0.705±0.002	0.651±0.002	0.618±0.005	0.677±0.006	0.834±0.003	0.875±0.001	0.541±0.002	0.568±0.004	0.611±0.004	0.640±0.002	0.706±0.001
Catboost+stacking	0.610±0.001	0.610±0.007	0.555±0.005	0.678±0.001	0.749±0.002	0.681±0.003	0.641±0.005	0.680±0.009	0.836±0.003	**0.878**±0.001	0.545±0.002	0.566±0.007	0.599±0.005	0.625±0.003	0.692±0.001
XGBoost	0.573±0.009	0.632±0.002	0.586±0.001	0.626±0.001	0.583±0.001	0.572±0.006	0.646±0.002	**0.762**±0.003	0.804±0.001	**0.877**±0.003	0.618±0.006	0.548±0.001	0.613±0.002	0.659±0.001	0.502±0.002
XGBoost+stacking	0.584±0.007	0.636±0.004	0.584±0.002	0.627±0.001	0.574±0.008	0.568±0.004	0.652±0.002	**0.763**±0.003	0.802±0.001	**0.878**±0.003	0.623±0.004	0.546±0.001	0.613±0.002	0.659±0.001	0.505±0.003

Table 13: **ROC-AUC scores for all models for "Atrial", "Purpura" and "Alcohol" downstream tasks.** FT denotes FT-Transformer, Tab denotes TabTransformer. The first two rows in each section correspond to training from scratch, where FS corresponds to deep baseline architecture (tuned on subsample of upstream data), and FS-2 shows the results for the same architecture as one used with transfer learning. LH and MLP denote linear and MLP heads correspondingly, E2E denotes end-to-end training.