What can we Learn by Predicting Accuracy?

Olivier RISSER-MAROIX *
LIPADE, Université Paris Cité
France
orissermaroix@gmail.com

Benjamin CHAMAND *
IRIT, Universite de Toulouse, CNRS,
Toulouse INP, UT3, Toulouse France
benjamin.chamand@irit.fr

Abstract

This paper seeks to answer the following question: "What can we learn by predicting accuracy?". Indeed, classification is one of the most popular tasks in machine learning, and many loss functions have been developed to maximize this non-differentiable objective function. Unlike past work on loss function design, which was guided mainly by intuition and theory before being validated by experimentation, here we propose to approach this problem in the opposite way: we seek to extract knowledge by experimentation. This data-driven approach is similar to that used in physics to discover general laws from data. We used a symbolic regression method to automatically find a mathematical expression highly correlated with a linear classifier's accuracy. The formula discovered on more than 260 datasets of embeddings has a Pearson's correlation of 0.96 and a r^2 of 0.93. More interestingly, this formula is highly explainable and confirms insights from various previous papers on loss design. We hope this work will open new perspectives in the search for new heuristics leading to a deeper understanding of machine learning theory.

1. Introduction

Most work in machine learning is done by building up and evaluating components from theoretical intuitions. Here we propose a different approach, which is to acquire insights from experimentation, in the same way that physicists have attempted to discover the analytical laws underlying physical phenomena in nature from observations. However, thanks to breakthroughs in artificial intelligence, a new trend to automate and assist research with *Machine Learning* (ML) tools is emerging. Some researchers started to use it in mathematics [9] and physics [12, 49].

In ML, the most similar setting would be the *meta-learning* one. In this *learning-to-learn* paradigm, a model gains experience over multiple learning episodes and uses this experience to improve its future learning performance. Hospedales et al. [23] reported successful applications of meta-learning on diverse tasks such as hyperparameter optimization, neural architecture search (NAS), etc. In this setting, the machine generally improves solutions without any human intervention. Although meta-learning has been widely explored and is actively involved in increasing the performance of machine learning models. The solutions found are generally non-interpretable. So surprisingly, including AI in the process hasn't caught the attention as a tool helping in the theoretical discoveries of ML studies. Hence, we investigate how machine learning can be integrated into the research process and lead us to better understand our discipline. As example, we propose here to tackle the problem of finding the key components of embeddings leading to better accuracy. This task could help us to better understand the intrinsic mechanisms of learning representations. Indeed, representation learning is often evaluated on benchmarks, such as [25] in NLP or [16, 61] in Computer Vision, where the task of classification is highly present. For example, self-supervised learning image representations are evaluated with a linear classifier. Classification performances are generally measured using the *accuracy*. To optimize this non-differentiable objective, researchers proposed proxy losses such as cross-entropy, hinge loss, and variants satisfying some properties and correcting several defaults of the previous ones. We can thus benefit from decades of research to validate the machine-generated function.

The task of predicting the future accuracy of a machine learning model has received little attention. While this question may look odd at first glance, answering it has multiple applications, such as: fastening NAS by being able to predict the performance of a random architecture without having to train it [29, 57]; evaluating the accuracy of a classifier on an unlabeled test set [10]; or measuring the difficulty of a dataset [8, 48]. Accuracy can thus be estimated from network weights [60], network architecture [57] or, as in our case, dataset statistics [4, 8, 10]. Previous works mostly rely on regression models such as neural

*Both authors contributed equally to this research.

networks or random forests, making solutions found non-explainable [10, 60]. While showing good performance for their respective use cases, those works did not focus on the interpretability of their solution.

In this paper, we provide a general formula by studying more than 260 datasets of embeddings with very different characteristics (size, dimension, number of classes, etc.). We propose to project those datasets into the same representation space by describing them as a set of statistics. From those statistical representations, we found a formula able to predict the future classification performance of a linear classifier with a strong Pearson's correlation and r^2 score. When comparing similar pipelines, we found our formula simpler and more explainable. Finally, we analyze it in light of decades of research.

2. Related Works

The scientific method requires understanding the mathematical relationships between variables in a given system. Symbolic Regression (SR) aims to find a function that explains hidden relationships in data without having any prior knowledge of the function's form. On the other hand, traditional regression imposes a single fixed model structure during training, frequently chosen to be expressive (e.g., neural network, random forest, etc.) at the expense of being easily interpretable. Because SR is believed to be an NP-hard problem [54], evolutionary methods have been developed to obtain approximate solutions [32, 31, 1, 42]. The symbolic regression challenge has recently regained popularity, and novel approaches combining classical genetic programming and modern deep reinforcement learning have emerged [45, 36, 44, 52, 53]. Indeed, when tested on 240 small datasets of 250 observations, SR was found to be both highly interpretable and competitive on small datasets [58].

To learn a model mapping any dataset to a predicted accuracy score, we must build a shared representation space among all datasets. For example, [43] used nine metrics of data complexity to characterize the behavior of several classifiers (linear, KNN, etc.) and thus found their respective domains of competence: where they perform best. In another work [22] found, by analyzing the twelve measures they proposed, that rich structures exist in such a measurement space, revealing the intricate relationship among the factors affecting the difficulty of a problem. However, they only examined the training sets' structures without generalizing to unseen points. More recently, [41] listed 22 complexity measures from past literature. Unfortunately, most have a complexity greater than $O(n^2)$, with n the number of points in the dataset. This makes those measures difficult to scale up to larger datasets. The task of finding statistical features for datasets representation is still considered an open question [10].

Close to our work, [8] propose to understand the difficulty of a text classification task using textual statistics to describe datasets used and a genetic algorithm to find the summation of those statistics correlating best with the F1-score. However, their work is limited by the choice of features, such as n-grams, making it only usable for textual datasets. By searching, with a *Genetic Algorithm* (GA), an unweighted summation of a subset of proposed statistics, they could only cover features having the same magnitude, discarding pertinent other ones such as the dataset size. The choice of an unweighted summation is likely to perform worse than a weighted one learned by a linear regression model. However, our solution confirmed intuition from [22] suggesting that the relationship between statistics is highly non-linear.

In another interesting work, [4] proposed estimating the predictive accuracy of several classifiers to select the most suited for a given dataset. In their analysis, the authors studied only one linear model: the linear discriminant analysis (LDA).However, the current state-of-the-art use *softmax* based models require gradient descent approaches. In order to extract knowledge from a meta-dataset of tabular datasets, they used Cubist[1], a package producing models in the form of rulesets. However, by being numerous and formulated with hard-coded values, the generated rules are complex and difficult to generalize.

By being applied to specific data such as text or tabular ones, neither [8] nor [4] used the same set of statistics, making the results of their proposed pipeline not comparable. Here, we focus on general embeddings from datasets with a broader diversity in their characteristics, such as the range of the number of classes ([8] the biggest one being 115 while we generalize up to 1824 classes). In this work, we choose to describe our datasets with 19 domain agnostic statistics. Moreover, we compare the solution found by our pipeline with solutions found with previous ones [8, 4]. Using our set of general statistics, we found that our solution is more efficient than the others while being simpler.

3. Proposed Approach

As illustrated in Figure 3, our method is composed of two parts: (1) the creation of a meta-dataset $\mathcal{M}$ from the combination of different datasets and feature extractors, its representation, and ground-truth creation; (2) the discovery of an explainable heuristic by symbolic regression modeling. We detail each component in the following paragraphs.

Datasets and Feature Extractors To find a general law covering a wide range of cases for a classification task, we selected 12 datasets and 22 feature extractors. The number of classes ranges from 10 to 1854, while the dimen-

[1] https://cran.r-project.org/web/packages/Cubist/vignettes/cubist.html

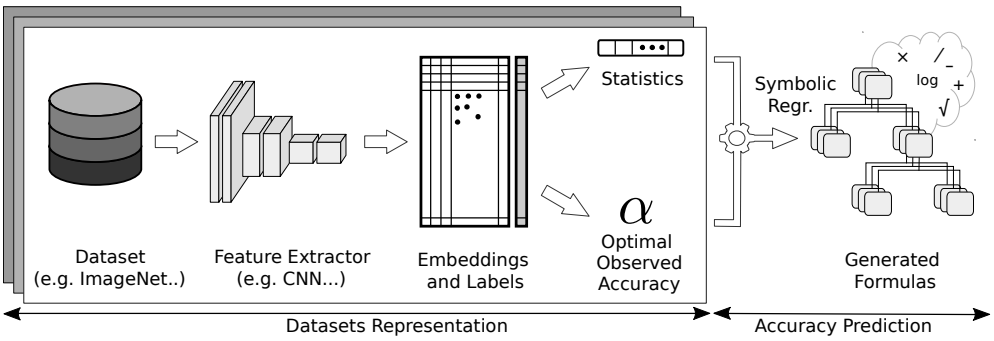

Figure 1. Proposed workflow

sionality of the features ranges from 256 to 2048. The selected datasets are MNIST [38], CIFAR10 [33], DTD [7], PhotoArt [59], CIFAR100 [33], 105-PinterestFaces [51], CUB200 [56], ImageNet-R [21], Caltech256 [17], FSS1000 [40], ImageNetMini [37],THINGS [20], containing respectively 10, 10, 47, 50, 100, 105, 200, 200, 256, 1000, 1000, 1854 classes. Regarding the feature extractors, different architectures have been selected with different pretraining to cover a large number of dimensions and difficulty levels of linear classification. For example, an architecture like FaceNet [50] is expected to perform poorly on CIFAR datasets since it is learned on a face recognition task while being a better feature extractor on this same dataset than a random initialized one. The ImageNet pretrained feature extractors used are: AlexNet [34], ResNet [19] (RN-{18, 50, 101}), DenseNet [27] (DN-{169, 201}), SqueezeNet [28], MobileNetv2 [47], MobileNetv3 [24] small and large versions. We also used FaceNet [50] pretrained on VGGFaces2 and CLIP-{RN50, ViT16b, ViT32b} [46] pretrained on millions of image-text pairs. As untrained feature extractors, we used: ResNet (RN-{34, 152}), DenseNet (DN-{169, 201}), SqueezeNet, MobileNetv2, MobileNetv3 small and large versions. All embedding dimensions represented here are: {256, 512, 576, 768, 960, 1024, 1280, 1664, 1792, 1920, 2048}. We refer to embeddings produced from the combination of all datasets of images by all feature extractors as a *dataset of embeddings*. We construct a meta-dataset $\mathcal{M}$ from those 260+ datasets of embeddings.

Meta-Dataset Representation To be able to find the hidden relationship between a given dataset and the associated optimal accuracy, we need to describe each of those datasets by a feature vector s in a shared representation space $\mathcal{S}$. Inspired by [22, 41, 43, 5] we selected various features s_i (the more diverse and uncorrelated are the statistics, the better): the dimensionality of embeddings (*dim*), the number of output classes (*n_classes*), the trace of the average matrix of all intra-class covariance matrices (*sb_trace*), the trace of the average of all inter-classes covariances matrices

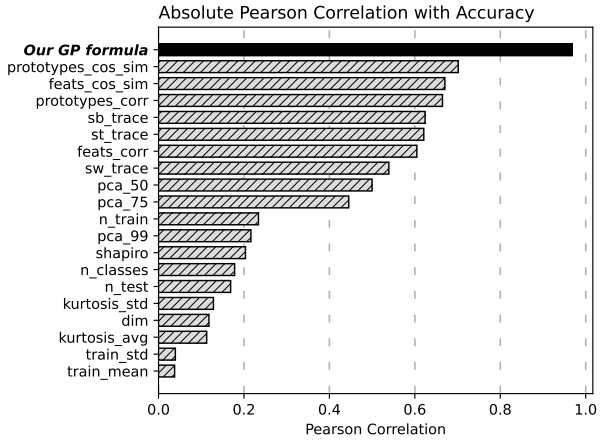

Figure 2. Absolute value of Pearson correlation between each dataset statistic and accuracy.

(*sw_trace*), the sum of the two previous traces (*st_trace*), the mean squared deviation (MSD) between the features' correlation matrix and the identity (*feats_corr*), the mean cosine similarity between each pair of dimensions (*feats_cos_sim*), the percentage of dimensions to be retained for a given explained variance of 50%, 75% and 99% (*pca_XX%*) to capture information about the dataset intrinsic dimension, the average of all embedding values (*train_mean*) and the standard deviation (*train_std*), the average and the standard deviation of the kurtosis computed on each dimension (*kurtosis_avg*, *kurtosis_std*), and the average Shapiro-Wilk value testing the normality of each dimension (*shapiro*). The two variables (*prototypes_corr*, *prototype_cos_sim*) refer respectively to information about the correlation and the cosine similarity between prototypes. Here, the term prototype denotes the average embedding for each class. Finally, we added number of samples in the training set (*n_train*) and the testing set (*n_test*). The correlations of each statistic with accuracy are reported in Figure 2.

Ground Truth Creation Once we have extracted the embeddings from various datasets with feature extractors, we

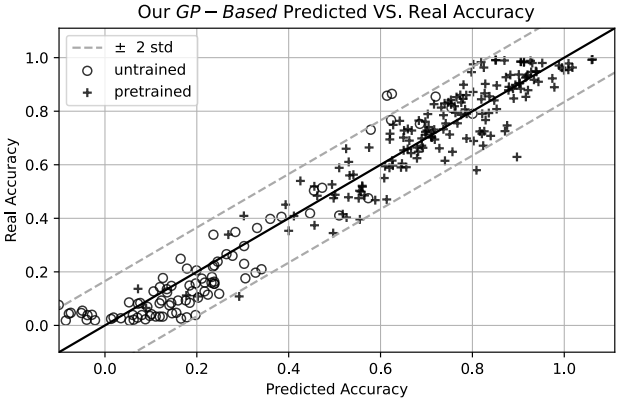

Figure 3. We can observe a strong linear relationship between our predicted accuracy compared to the real one.

need to find the best reachable accuracy by a softmax classifier for each case. To do so, we applied the same procedure as [5]. We split each dataset of embeddings into training and testing sets and trained the model during 1000 epochs with a batch size of 2048. All embeddings and classifier weights are ℓ_2-normalized. The test sets are the usual ones for datasets with a specific split, such as CIFAR. We used a 66/33 split for few-shot datasets, such as THINGS, to ensure that the train/test split proportion left at least 10 images per class. The other ones were split with a ratio of 75/25. By tracking the accuracy on the test set, we can observe the best-reached accuracy α that we will consider as a good approximation of the best accuracy reachable α^*. We used Adam optimizer. Our meta-dataset $\mathcal{M} = \{(s_i, \alpha_i)\}_{i=1}^{D}$ corresponds to all the pairs of statistical representation $s_i \in \mathcal{S}$ of each dataset d_i of the D datasets and the observed optimal accuracy $\alpha_i \in \mathcal{A}$. Those tuples are then our inputs and targets.

Symbolic Regression Recovering hidden algebraic relationship between variables in order to describe a given phenomenon is the objective that symbolic regression (SR) seeks to optimize. We search a prediction function $p : \mathbb{R}^S \rightarrow \mathbb{R}$ from our meta-dataset $\mathcal{M}$, with S the number of statistical features representing each dataset d_i. As previously mentioned, different approaches have been developed for symbolic regression. By benchmarking SR frameworks and ML models, it has been found that DSO [45], a deep learning-based approach, and *gplearn*, a genetic programming (GP) framework, are two of the top-5 methods compared [35]. When trying with code provided by DSO [45] on our task, solutions found under-performed the *gplearn* with more complex formulas and longer training time. Thus, we focus here on the *gplearn* implementation[2] because of the compactness of the solutions found, speed of execution, and easiness of use. In GP-based symbolic regression, a population $\mathcal{P}$ of randomly generated mathematical expressions is "evolved" using evolutionary operations like *selection*, *crossover* and *mutation* to improve a fitness function $\mathcal{F}$. The individuals p in the population $\mathcal{P}$ are represented as hierarchical compositions of primitive functions τ and terminals appropriate to the particular problem domain. Here, $\tau = \{\log, e, \sqrt{;}+, -, \times, \div\}$ and the set of terminals corresponds to the statistics s_i describing the dataset d_i. We evolved a population of 5000 individuals for 20 steps and tested 3 different fitness functions : the first one corresponds to the r^2 between the predicted formula and the expected result. This fitness function produced poor results both on training and testing sets. The second one measures Pearson's correlation between the expected and predicted accuracies. While being easier to optimize than the first one, we found this one to be surprisingly inefficient since it tends to group the pretrained representations in a compact cluster, and the untrained ones in another one such that a line passes through the two centroids. Indeed, the Pearson correlation between model accuracies and the variable specifying whether a pretrained or untrained model is used for embedding extraction is already at 0.77. To overcome this effect, we designed a simple fitness function such that both pretrained and untrained extracted embeddings independently have a linear correlation with accuracy. For a given individual, here a GP predictor formula $p(\cdot)$, we assess its fitness score $\mathcal{F}$:

$$\mathcal{F} = \min \Big[\Big| \text{pearsonr}\Big(p(\mathcal{S}_{pretrained}), \mathcal{A}_{pretrained}\Big) \Big|, \Big| \text{pearsonr}\Big(p(\mathcal{S}_{untrained}), \mathcal{A}_{untrained}\Big) \Big| \Big] \quad (1)$$

with $\mathcal{S}_{subset}, \mathcal{A}_{subset}$ corresponding respectively to the sets of statistical representations and target accuracies α of the given $subset \in \{pretrained, untrained\}$. While this constraint does not enforce to have both pretained and untrained sets to be correlated with the same tendency, we can, however, experimentally observe the benefits on Figure 3, where pretrained and untrained networks are not separated in very distinct clusters but are distributed around a line. We split our meta-dataset in a fixed 75/25-train/test fashion and repeated each experiment 1000×. Since $\mathcal{F}$ only seeks for correlation, a linear transformation of the output value is learned on the training set in order to predict the accuracy: $\hat{\alpha} = a \cdot p(\cdot) + b$.

4. Results

Baselines To evaluate the performance of our GP solution, we compare it with popular regression methods, including linear regression, decision tree regression, and random forest regression. The same training/test split has been used for all those methods. All variables are used simul-

[2] https://gplearn.readthedocs.io/

Table 1. Our formula has a better correlation and higher predictive power with only 5 variables, while the other models used the 19 variables (all p-value < 0.01).

Method	Pearson$r$	r^2
Linear Regression	0.9042	0.8011
Decision Tree	0.9472	0.8868
Random Forest (10 trees)	0.9643	0.9246
Our GP formula (GPF)	**0.9671**	**0.9319**

taneously. Performances on the test set are reported in Table 1. With a substantial gap of r^2 score between the linear regressor and our formula, we can conclude that the task of predicting the accuracy requires a complex non-linear combination of only a few variables. Furthermore, we compare with non-linear regressors such as decision trees and random forests. We choose those because of their performances and the widespread belief suggesting those models are among the most interpretable ones. We used *sklearn* implementations. Our formula outperformed the decision tree and performed similarly to the random forest while being much more explainable.

Symbolic Regression Formula We ran our GP pipeline 1000 times on the same training set and serialized their respective solutions and scores for analysis. The solution having the best test r^2 score was found $6\times$. We compare on Figure 4 the test performances to the complexity of solutions found. Our formula has a complexity of 6 nodes. We will refer to this ***G**enetic **P**rogramming **F**ormulas* as:

$$GPF = \log\left(\frac{sb\_trace/st\_trace}{\sqrt{n\_classes \cdot feats\_corr \cdot prototypes\_cos\_sim}}\right) \quad (2)$$

We can easily rewrite : $GPF = SEP - COR$ with:

$$\begin{aligned} SEP &= \log\left(\frac{sb\_trace}{st\_trace}\right) \\ COR &= \frac{1}{2}\log\left(n\_classes \cdot feats\_corr \cdot prototypes\_cos\_sim\right) \end{aligned} \quad (3)$$

SEP may correspond to a **SEP**arability criterion while COR may correspond to **COR**relation information. Section. 5 delves deeper into each formula component. By ablating GPF and considering each part independently, we found they were complementary. Indeed, SEP has only a Pearson's correlation of 0.65 and COR of -0.87 while the combination of the two parts reached 0.96. Finally, we found that other best-performing GP formulas have a similar structure and variables. We report in Figure 5 how many times each statistic was used during the 1000 runs.

Ablation As a first ablation test, we propose demonstrating how the variables selected by our GPF influence the results for different baselines. Table 2 shows a minor decrease in performance of those methods when compared to

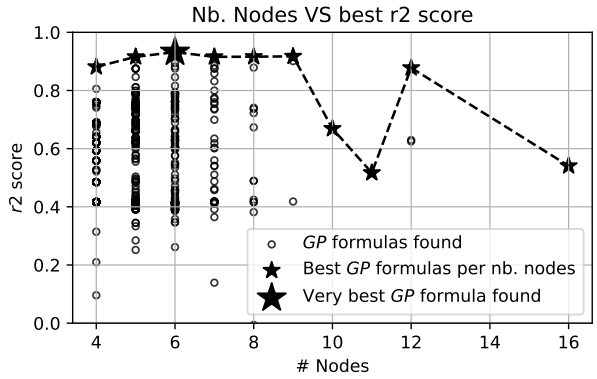

Figure 4. Performance versus the complexity of GP-Formulas.

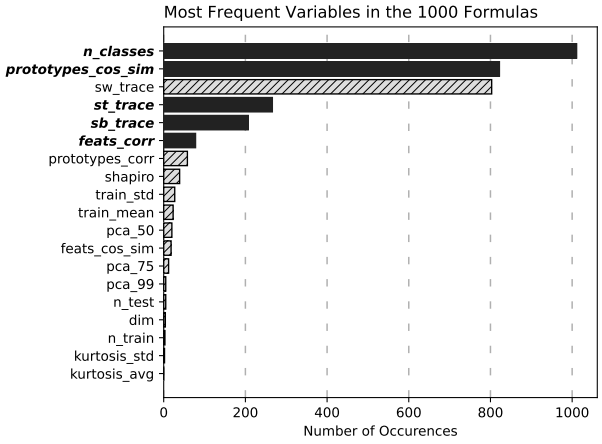

Figure 5. Frequency of the most used variables in the 1000 GP-formulas found. Dark bars indicate the variables present in the best GP formula (one variable can be used several times per formula).

Table 2. The slight variation in baseline precision when only using the five variables chosen by our genetic formula suggests that those five variables are the most important for all methods. (all p-value < 0.01).

Method	Pearson$r$	r^2
Linear Regression	0.8796	0.7689
Decision Tree	0.9538	0.8937
Random Forest (10 trees)	0.9532	0.9057
Our GP formula (GPF)	**0.9671**	**0.9319**

the original baseline from Table 1. For example, Pearson's correlation of the random forest falls from 0.9643 to 0.9532, while the decision tree rises from 0.9472 to 0.9538. On the other hand, the linear regression model needs to suffer from this feature selection step by dropping from 0.9042 to 0.8796. All p-values < 0.01 suggests that selected variables could be sufficient to correlate with accuracy. However, a non-linear transformation of those variables is still required.

One can note that the log properties can still be ap-

Table 3. We evaluate the performance of our formula after replacing each variable with its mean value. All p-values < 0.01, except for Sb_trace with a p-value of 0.6882. Bold scores correspond to the worst one making the variable the most important one.

Ablated Variable	Pearson$r$	r^2
Sb_trace	**−0.0503**	**−2.1806**
n_classes	0.7918	0.4341
St_trace	0.8028	−1.2761
feats_corr	0.8530	0.5818
prototypes_cos_sim	0.9420	0.8764
No variable ablated	<u>0.9671</u>	<u>0.9319</u>

plied to our GPF to reduce our formula as a linear combination of our selected variables. With coefficients being $[1, -1, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}]$, our GPF could thus be written as:

```
GPF = log(Sb_trace) - log(St_trace)
      - 0.5 log(n_classes)
      - 0.5 log(feats_corr)
      - 0.5 log(prototypes_corr)
```
(4)

Given that no prior on the structure of the GPF was imposed during the search phase, this discovery is particularly interesting. With this finding arise two closely related questions: 1) does learning those five coefficients improve performance? 2) what would be the performances if all variables were used? To answer the first one, we learned a linear regression model on the five statistics after passing them to the log. Doing so increased the performance of the linear model without the log transform from a Pearson's coefficient of 0.9042 in Table 1 to 0.9607 after log transforming inputs as reported in Table 4. While being more efficient on the train set, the linear model performed worse on the test set than our GPF. However, when comparing the learned weights, we found that signs and magnitudes were highly similar to weights or our GPF with a cosine similarity of 0.9923. By not requiring any re-weighting of our five variables, our formula in its original form (Equation. 2) is thus more interesting. On the other hand, we learned a linear model on the log-transformed statistics using the same procedure. Due to negative values in the original ones, only 17 of 19 are kept. At the expense of being significantly less explainable, the model's scores are reported in Table 4 outperformed our GPF ones. Ones can note that the relative difference of reported correlations between the models with 17 variables and the GPF with 5 is smaller than the difference of the GPF with the 5 variables and the GPF with the 4 most important variables reported in Table 3.

Finally, our last ablation study seeks to determine which components of our formula are the most important ones. We can determine how much each variable influences scores by freezing each variable and replacing it with its mean value.

Table 4. We compare our solution to different methods. Our GPF offers the best compromise between accuracy and simplicity without having any prior on the structure of the solution. Our extension of the GPF using 17 variables offers the best Pearsons's correlation and r^2 score. Here *log* stands for log-transformed variables and *org* for the original ones.

Method	Nb. Var	Pearson$r$	r^2
GA unweighted sum	19 org	0.7763	0.5744
GA unweighted sum	17 log	*0.9621*	*0.9254*
Cubist Rules	19 org	0.9666	0.9343
Cubist Rules	5 log	0.9642	0.9276
Cubist Rules	17 log	*0.9772*	*0.9525*
Our Linear Regression	5 log	0.9607	0.9206
Our Linear Regression	17 log	**0.9795**	**0.9586**
Our GP formula	5 org	<u>*0.9671*</u>	<u>*0.9319*</u>

The more significant the drop, the greater the variable's significance. Table 3 allows us to see that freezing each variable results in a decrease in score. All p-values are significant (< 0.01), with the exception of *Sb_trace*. Indeed, freezing it removes all correlation between our GPF and the expected accuracy with a p-value of 0.6882. On the other hand, while having a minimal positive impact on correlating GPF with accuracy, *prototypes_cos_sim* appears to be still important to have a good r^2 prediction score.

Comparison to Related Work As previously mentioned, [8, 4] used neither similar datasets nor statistics for describing selected datasets making their work hard to compare them and with them. However, we propose to compare our solutions found by applying their pipeline on our meta-dataset. Results are reported in Table 4.

To find an unweighted sum of a few variables, we used a similar *genetic algorithm*[3] (GA) with the Pearson's correlation between predicted and real score as fitness function, such as in [8]. By setting variable type being integer and bounded between $[-1, 1]$, the 3 possibles values are $\{-1, 0, 1\}$. With an initial population of 5000 and 300 iterations, we found results to be stable. We used the exact same train/test split as us. To compute the r^2 score, we employed the same procedure of linearly re-calibrating the formula with (a, b) learned on the training set. As expected, the results on the 19 variables are significantly worse than the weighted summation of our baseline linear regression model. Thus, we tested the same pipeline after keeping and log transforming 17 variables (due to 2 out of the 19 variables having negative values). Consistently with the discovery of the ablation study suggesting only to log transforming variables as pre-processing, results increased. The best solution found with the genetic algorithm (GA) is reported

[3] https://github.com/rmsolgi/geneticalgorithm

```
if
  percentage_dims_exp_var_99 > 0.9316406
  feats_corr <= 0.05039461
  kurtosis_std <= 9.284021
  n_test > 1880
  n_test <= 4384
then
  outcome = 5.4992777
     - 0.031372 kurtosis_std
     - 5.02 percentage_dims_exp_var_99
     + 1.31 feats_corr
     + 0.83 Sb_trace
     - 0.19 prototypes_cos_sim
     + 0.085 prototypes_corr
     + 8e-06 n_test + 1.6 train_mean
     - 3.2e-05 n_classes
     - 1.8 train_std
```

Figure 6. Example of one of the ten rules output by *Cubist* when learning on the 19 variables.

in Equation. 5:

```
GA = log(Sb_trace) + log(shapiro)
     + log(dim) - log(feats_corr)
     - log(Sw_trace) - log(kurtosis_avg)
     - log(prototypes_corr)
```
(5)

Solution found used 7 variables while our GPF used only 5. With only 3 variables shared with our GPF, we find it difficult to understand the interaction with the selected variables.

To compare our solution to the pipeline proposed by [4], we used the R package implementing Cubist, the software the authors used to find an interpretative set of rules. As reported in Table 4 we experimented with three set of input variables. The first one corresponds to our 19 original variables without any transformation. While having scores comparable to our GPF using 5 variables, we can note that the rules are highly complex. Indeed, it produced 10 rules, using many coefficients, which are hard to read. One example of rule is reported in Figure. 6. In a second experiment, we used only the top 5 variables selected by our GPF after log transforming them. It helped the Cubist system in outputting comparable results by using only two rules. Each rule predicts the accuracy as a linear combination of all 5 log variables. We compare the coefficients of those rules with those of our original GPF (as in Equation. 4. Interestingly, they have a cosine similarity of 0.9467 and 0.9791 with our GPF ones. With only 5 variables, our GPF is simpler and performs better. Finally, we found that our log pre-processing also benefited Cubist. When giving the 17 log-transformed variables as input, Cubist proposed a solution based on 6 rules while being significantly more efficient than the 10 rules outputted from the 19 original variables. However, our extension of the GPF to the linear combination of the 17 log-variable still performs better while being much more straightforward than the Cubist's solutions.

Table 5. We examine our formula's transferability of baseline regressors with five variables. The best reachable GPF is indicated with the * symbol. All p-values < 0.01.

Method	Pearson$r$	r^2
Linear Regression	0.6191	0.3052
Decision Tree	0.7944	0.1928
Random Forest (10 trees)	0.7231	-0.0722
Our GP formula (GPF)	**0.8618**	**0.4565**
Our GP formula (GPF) *	**0.8618**	**0.7428**

Using Ockham's principle, those findings are evidence making our GPF a better choice. Furthermore, our GPF can be easier to explain because of its conciseness. We propose to discuss our GPF in Section. 5.

Generalization Our method is applicable to any dataset describable with a set of a few statistics. Because we obtained our GPF using statistics from embedding datasets extracted only from vision datasets and feature extractors, the generalization of our GPF discovered on those datasets to other domains, such as text, can be questioned. Thus, we used 7 text datasets[4], and 4 pretrained text features extractors from the *sentence-transformers* package[5] to test our formula's ability to transfer to new modalities. Combining all those datasets and feature extractors, we applied the same process to extract dataset statistics and accuracies, yielding 28 points for our analysis. Table 5 compares how our formula transfers to this new set of points with classical regressors. All reported correlations have a statistically significant p-value (< 0.01). We can observe a significant drop in Pearson's correlation and r^2 scores for all methods. However, our GPG still outperforms other methods with a strong Pearson's correlation of 0.8618. Two r^2 scores for our GPF were reported; the first one corresponds to our formula linearly transformed with the coefficient a, b learned on the training set of the vision dataset. The second one, obtained by linearly translating our GPF, refers to the best possible r^2 score on the text meta-dataset. To find the *oracle* coefficients, we evaluated the formula after learning the parameters a, b on the text meta-dataset (here train=test). The oracle gives us the best score reachable on this meta-dataset of 28 points. With $(a, b) = (0.2417, 1.0327)$ for the vision meta-dataset and $(a, b) = (0.2508, 0.9121)$ for the text meta-dataset, we can see that the parameters from the text and the vi-

[4]WOS5736 (11 classes), 20NewsGroup, Ohsumed-23, New "Year's Resolutions" (two datasets are build from two different columns, 10 and 115 classes), "Quotes with Authors" (1043 classes after filtering classes with at least 20 samples) and the Reuters21578-Apte-90Cat (90 classes).

[5]`https://www.sbert.net` (sentence-t5-base, allenai-specter, paraphrase-MiniLM-L6-v2, bert-base-uncased)

sion meta-dataset are similar. However, the r^2 score appears extremely sensitive, perhaps due to the small number of points. This score drop can be explained by the discrepancy between the text and vision meta-datasets. For example, while datasets with more than 200 classes are common in vision, text classification tasks typically have a much lower number of classes, such as 2 for sentiment analysis or 20 for topics modeling. We measured this discrepancy by performing Student's t-test on each of the five selected variables. We found three of the five variables have a p-value < 0.01 (*Sb_trace, n_classes, prototypes_cos_sim*), giving evidence against the null hypothesis of equal population means. While not perfect, our results appear promising. However, they would benefit from incorporating more meta-datasets from other domains, such as audio, video, graph-based, or tabular data classification datasets.

5. Discussion

As seen previously, GPF can be written as a summation of two components. With a closer look, one can observe that the first element SEP is close to the Fisher's criterion used in the *Linear Discriminant Analysis* (LDA) [14] where the objective is to find a linear projection that maximizes the ratio of between-class variance and the within-class variance. Thus, SEP corresponds to a separability measure of classes. Interestingly, this criterion has been used successfully as a loss function in deep learning [11, 15]. The choice of an LDA-based loss function remains marginal in deep learning, the cross-entropy (CE) being a more popular choice. However, strong similarities between the LDA and the CE allow us to swap this first separability measure for the latter. Indeed, [55] noticed that one of the most widely studied technical routes to overcome certain deficiencies of the softmax in the cross-entropy-based loss is to encourage stronger intra-class compactness and larger inter-class separability, analogously to Fisher's criterion.

The second part, COR, is negatively correlated to the accuracy. This is easily understandable by looking at each variable composing this part of the formula. The first is the number of classes ($n\_classes$). Indeed, when a machine learning model is trained on a dataset, it is natural to expect that scores will decrease as the number of classes grows. [18] discuss how, as the number of classes in a dataset grows, it gets harder to distinguish between them, making the dataset increasingly challenging to classify. This intuition may be empirically verified on datasets with different class granularities. For example, [6] observed a drop in accuracy from 0.97 to 0.82 on the CUB200 dataset [56] when changing the number of classes from a coarse level (13) to a fine-grained one (200). The two other variables ($feats\_corr$, $prototype\_cos\_sim$) correspond to orthogonality and decorrelation information. By looking at the literature, we can easily explain the importance of both decorrelation terms. In defense of the weights decorrelation term (*prototypes_cos_sim*), [2] found on several state-of-the-art CNN that they could achieve better accuracy, more stable training, and smoother convergence by using orthogonal regularization of weights. Previous works on features decorrelation heavily justify the presence of our features decorrelation variable (*feats_corr*) [3, 13, 26, 30, 39, 55, 62]. Indeed, [39] found that correlated input variables usually lead the eigenvectors of the Hessian to be rotated away from the coordinate axes leading to slower convergence. Thus, several propositions were developed to better decorrelate variables such as PCA or ZCA [30]. More recently, decorrelation played an essential role in the performance increase of self-supervised methods [3, 13, 26, 62]. For example, [13] recently introduced a whitening step in their self-supervised loss, and [3] included a decorrelation part in their loss. They argue that this term decorrelates the variables and prevents collapse.

6. Conclusion

In this paper, we showed that a simple pipeline could help us to extract theoretical intuitions from experimentation. To do so, we conducted experiments on a meta-dataset of more than 260 datasets of embeddings extracted from the combination of a wide range of datasets and feature extractors. To solve the problem of expressing such disparate datasets, we proposed combining them into a single space by creating a representation using a set of general statistics that can be computed on any dataset. As a result, our work applies to computer vision and all other areas of machine learning. Finally, an heuristic able of predicting the accuracy of a linear classifier was discovered automatically, with a Pearson's correlation of 0.96 and an r^2 of 0.93. Interestingly, other systems with similar performances tend to confirm our GPF by having highly correlated weights. Furthermore, this formula is highly explainable and is consistent with decades of research. This successful example of AI-assisted research encourages us to use it in other areas, such as predicting and understanding hyperparameters (regularization, temperature, tree depth, etc.).

7. Acknowledgment

This work was partially financed by *Smiths Detection*. Authors would like to thank all peoples involved in the proof-reading and contributed to substantially improving this document. Listed in alphabetical order: Thibault ALEXANDRE, Ihab BENDIDI, Mohamed CHELALI, Philippe JOLY, Celia KHERFALLAH, Camille KURTZ, Amine MARZOUKI, Julien PINQUIER, Guillaume SERIEYS.

References

[1] Douglas Adriano Augusto and Helio J. C. Barbosa. Symbolic regression via genetic programming. In *SBRN, Procs.*, pages 173–178, 2000.

[2] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *NIPS, Procs.*, volume 31, 2018.

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR, Procs.*, 2022.

[4] Hilan Bensusan and Alexandros Kalousis. Estimating the predictive accuracy of a classifier. In *ECML, Procs.*, pages 25–36, 2001.

[5] Benjamin Chamand, Olivier Risser-Maroix, C. Kurtz, P. Joly, and N. Loménie. Fine-tune your classifier: Finding correlations with temperature. In *ICIP, Procs.*, 2022.

[6] Dongliang Chang, Kaiyue Pang, Yixiao Zheng, Zhanyu Ma, Yi-Zhe Song, and Jun Guo. Your" flamingo" is my" bird": Fine-grained, or not. In *CVPR, Procs.*, pages 11476–11485, 2021.

[7] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR, Procs.*, 2014.

[8] Edward Collins, Nikolai Rozanov, and Bingbing Zhang. Evolutionary data measures: Understanding the difficulty of text classification tasks. In *CoNLL*, 2018.

[9] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[10] Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *CVPR, Procs.*, pages 15069–15078, 2021.

[11] Matthias Dorfer, Rainer Kelz, and Gerhard Widmer. Deep linear discriminant analysis. In *ICLR, Procs.*, 2016.

[12] Michael R Douglas. Machine learning as a tool in theoretical science. *Nature Reviews Physics*, pages 1–2, 2022.

[13] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *ICML, Procs.*, pages 3015–3024. PMLR, 2021.

[14] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[15] Benyamin Ghojogh, Milad Sikaroudi, Sobhan Shafiei, Hamid R Tizhoosh, Fakhri Karray, and Mark Crowley. Fisher discriminant triplet and contrastive losses for training siamese networks. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

[16] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. *arXiv preprint arXiv:1905.01235*, 2019.

[17] Greg Griffin, Alex Holub, and Pietro Perona. Caltech256 image dataset. 2006.

[18] Maya R Gupta, Samy Bengio, and Jason Weston. Training highly multiclass classifiers. *The Journal of Machine Learning Research*, 15(1):1461–1492, 2014.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR, Procs.*, pages 770–778, 2016.

[20] Martin N. Hebart, Adam H. Dickter, Alexis Kidder, Wan Y. Kwok, Anna Corriveau, Caitlin Van Wicklin, and Chris I. Baker. THINGS: A database of 1, 854 object concepts and more than 26, 000 naturalistic object images. *PLOS One*, 14(10):e0223792, 2019.

[21] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *CoRR*, abs/2006.16241, 2020.

[22] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):289–300, 2002.

[23] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV, Procs.*, pages 1314–1324, 2019.

[25] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *ICML, Procs.*, pages 4411–4421. PMLR, 2020.

[26] Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *ICCV, Procs.*, pages 9598–9608, 2021.

[27] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR, Procs.*, pages 4700–4708, 2017.

[28] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[29] Roxana Istrate, Florian Scheidegger, Giovanni Mariani, Dimitrios Nikolopoulos, Constantine Bekas, and Adelmo Cristiano Innocenza Malossi. Tapas: Train-less accuracy predictor for architecture search. In *AAAI, Procs.*, volume 33, pages 3927–3934, 2019.

[30] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

[31] John R. Koza. *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1993.

[32] John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2), 1994.

[33] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS, Procs.*, 2012.

[35] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[36] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *ICML, Procs.*, pages 5979–5989, 2021.

[37] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[38] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database. `http://yann.lecun.com/exdb/mnist`, 2010.

[39] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[40] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. FSS-1000: A 1000-class dataset for few-shot segmentation. In *CVPR, Procs.*, pages 2866–2875, 2020.

[41] Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin Kam Ho. How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys*, 52(5):1–34, 2019.

[42] Qiang Lu, Jun Ren, and Zhiguang Wang. Using genetic programming with prior formula knowledge to solve symbolic regression problem. *Computational Intelligence and Neuroscience*, 2016:1021378:1–1021378:17, 2016.

[43] Ester Bernadó Mansilla and Tin Kam Ho. On classifier domains of competence. In *ICPR, Procs.*, volume 1, pages 136–139. IEEE, 2004.

[44] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel M. Faissol, and Brenden K. Petersen. Symbolic regression via neural-guided genetic programming population seeding. In *NIPS, Procs.*, 2021.

[45] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *ICLR, Procs.*, 2021.

[46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning transferable visual models from natural language supervision. In *ICML, Procs.*, pages 8748–8763, 2021.

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR, Procs.*, pages 4510–4520, 2018.

[48] Florian Scheidegger, Roxana Istrate, Giovanni Mariani, Luca Benini, Costas Bekas, and Cristiano Malossi. Efficient image dataset classification difficulty estimation for predicting deep-learning accuracy. *The Visual Computer*, 37(6):1593–1610, 2021.

[49] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[50] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR, Procs.*, pages 815–823, 2015.

[51] Burak Toy. Pins Face Recognition — kaggle.com. `https://www.kaggle.com/hereisburak/pins-face-recognition`. [Accessed 13-Dec-2021].

[52] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):2631, 2020.

[53] Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter AN Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation*, 29(2):211–237, 2021.

[54] Marco Virgolin and Solon P Pissis. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

[55] Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *CVPR, Procs.*, pages 9117–9126, 2018.

[56] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, Caltech, 2010.

[57] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *ECCV, Procs.*, pages 660–676. Springer, 2020.

[58] Casper Wilstrup and Jaan Kasak. Symbolic regression outperforms other models for small data sets. *ArXiv*, abs/2103.15147, 2021.

[59] Qi Wu, Hongping Cai, and Peter Hall. Learning graphs to model visual objects across different depictive styles. In *ECCV, Procs.*, pages 313–328, 2014.

[60] Yasunori Yamada and Tetsuro Morimura. Weight features for predicting future model performance of deep neural networks. In *IJCAI*, pages 2231–2237, 2016.

[61] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. The visual task adaptation benchmark. 2019.

[62] Shaofeng Zhang, Feng Zhu, Junchi Yan, Rui Zhao, and Xiaokang Yang. Zero-cl: Instance and feature decorrelation for negative-free symmetric contrastive learning. In *ICLR, Procs.*, 2022.