The Alan Turing Institute

Low-Cost Model Selection for Transformers (LoCoMoSeT)

March 2024
Edmund Dable-Heath, Data Scientist
Phil Swatton, Data Scientist
Jack Roberts, Lead Data Scientist
James Bishop, Principal Data Scientist

1 Executive Summary

The wide availability of performant pre-trained models in online repositories, such as the HuggingFace model hub, has led to a paradigm shift in applied data science away from training new models from scratch and towards transfer learning – fine-tuning existing pre-trained models on a new downstream task of interest. Given hundreds or even thousands of pre-trained models are available, and the unreasonably large computational cost required to fine-tune a model, it is difficult to identify which model should be selected for fine-tuning. To address this problem, several low-cost metrics for predicting the downstream fine-tuned model performance have been developed in the literature. We assess the performance of these metrics in the case of image classification with a pool of vision transformers, with all models in the pool using the common ImageNet pre-training dataset. We find simple metrics with essentially zero computational cost, such as the number of model parameters or the reported ImageNet validation accuracy, are competitive with higher- (but still low-) cost metrics that require inference on the fine-tuning dataset. There is some evidence that combining a zero-cost metric and a low-cost metric requiring inference on a small number of samples from the new downstream task performs best overall. These recommendations may differ if considering pools of models pre-trained on different datasets.

Contents

1 Executive Summary **2**

2 Introduction **5**

3 Problem Statement **5**

3.1 Model Development 5

3.2 Model Selection 6

3.3 Sources of Fine-Tuning Success 7

4 Candidate Metrics **7**

4.1 Metric Categorisation 7

4.2 Zero-Cost, Task-Agnostic Metrics 8
4.3 Low-Cost, Task Dependent Metrics 9
4.4 Combinations of Metrics 12
4.5 Assessing Metric Performance 12
4.6 Reported Metric Performance 13

5 Metric Selection **13**

6 Experiment Design **14**
6.1 Datasets 14
6.2 Models 17
6.3 Fine-Tuning Strategy 19
6.4 Metrics Computation 19
6.5 Hardware 19

7 Results **19**
7.1 Performance Evaluation 20
7.2 Combining Metrics 21
7.3 Impact of Dataset Size on Performance 23
7.4 Time Saving Relative to Fine-Tuning 24

8 Limitations **25**

9 Conclusions **27**

10 Recommendations for Future Work **28**

A Reported Metric Performance **33**

B Excluded Models **34**

C Initial Metric Exploration **34**
C.1 Number of Model Parameters 34
C.2 Renggli 35
C.3 PARC 36
C.4 Topological Data Analysis 36
C.5 LogME 36

D Additional Results **37**

D.1 Raw Data Examples 37

D.2 Results at Each Training Set Size 40

D.3 Combining Metrics 44

2 Introduction

The task of effective model architecture search has been a key part of deep learning since its inception - with a particular explosion in research in recent years [1, 2]. Whilst there are multiple proposed methods for developing architectures from scratch, there has been a paradigm shift away from designing novel architectures for each new task given the proliferation of large pre-trained models. Instead, data scientists now perform transfer learning by fine-tuning a pre-trained model on a downstream task

As a consequence of this shift, there now exists a 'marketplace' of models that one can take off the shelf and fine-tune. Given the wide array of pre-trained models available, identifying the best model to fine-tune on a new dataset and task is a time-consuming problem. In particular, the computational time and costs required to fine-tune models make it prohibitively expensive to consider more than a few candidate models for further development. In this project, we have explored time and compute-efficient metrics by which suitable pre-trained models for a given fine-tuning dataset can be identified, with the aim of both saving time in model development and improving ultimate performance on the downstream task by allowing more candidate models to be considered.

Although model selection is a general problem relevant to any downstream task, we focus on image classification as a common example use case. As described in more detail in later sections, we also focus our experiments on models using the vision transformer architecture (ViT) [3], inspiring the project name LoCoMoSeT - **Lo**w-**Co**st **Mo**del **Se**lection for **T**ransformers, and we will refer to the selection metrics as the 'LoCoMoSeT metrics'. We start by formally defining the metrics selection problem, before describing prior work on low-cost selection strategies from the literature. We then describe our experiments and results in the remainder of the report.

3 Problem Statement

In this section, we formally describe a typical development process for adapting a pre-trained model to a new downstream task. In particular, we focus on the computational cost involved in model selection as part of this process, and where the metrics we explore as part of this project could assist in reducing that cost.

3.1 Model Development

In supervised machine learning tasks[1], given a target dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N}$ consisting of N samples of input features x_i (such as images) and associated labels y_i, the task is to learn a model, ϕ_t, to predict the labels given the input features, $y_i \sim \phi_t(x_i)$. An optimisation (training) process is followed to learn the parameters of ϕ_t to minimise the difference between the predicted labels and the ground truth labels, guided by a loss function or evaluation metric $\mathcal{L}\left(\mathcal{D}_t, \phi_t\right)$.

Model training can be expensive in terms of human development time and computational costs, especially for current state-of-the-art deep learning approaches where models can have millions or billions of parameters and training datasets millions of images, or trillions of tokens in the case of text datasets. In most applied data science settings the data, computational, and human resource requirements to develop a model competitive with the state of the art from scratch are not available.

Instead, practitioners build on models available in online repositories, or 'model hubs', such as the HuggingFace model hub [4]. These provide both implementations of performant model architectures (removing the need to design or implement an architecture from scratch), and of pre-trained weights for those models – learnt weights using a pre-training or 'source' dataset, $\mathcal{D}_s$, which in most circumstances will be different to, and much larger than, the target dataset ($\mathcal{D}_t \neq \mathcal{D}_s$ and $|\mathcal{D}_s| \gg |\mathcal{D}_t|$). Given a pre-trained model selected from the model hub, ϕ_s, the goal is to update the weights of the model by fine-tuning them on the target dataset to create the model $\phi_t = T\left(\phi_s, \mathcal{D}_t\right)$, transferring knowledge from the pre-training dataset to achieve higher performance on the target dataset compared to starting with a randomly initialised model. We notate

[1]While we focus on supervised learning in providing our definitions, this is without loss of generality for the points raised in this section.

pre-trained models from a model hub as ϕ_s, models fine-tuned on the target dataset as ϕ_t, and use $T\left(\phi_s, \mathcal{D}_t\right)$ to denote the fine-tuning process (which requires a pre-trained model and target dataset as input).

3.2 Model Selection

The first step of any model development pipeline using transfer learning is pre-trained model selection, namely, given a model hub $\mathcal{M}$, which pre-trained model (or models) $\phi_s \in \mathcal{M}$ is the best choice for the new downstream task?

A brute-force selection process would be to pick a subset of models, $\mathcal{S} \in \mathcal{M}$, fine-tune them all, and pick the best-performing model, ϕ_t^*, so that:

$$\phi_t^* = \underset{\phi_s \in \mathcal{S}}{\arg\max}\, \mathcal{L}\left(\mathcal{D}_t, \phi_t\right) \tag{1}$$

where $\mathcal{L}$ is assumed to be an evaluation metric to be maximised, such as the achieved validation accuracy for a classification task. Note that a pre-trained model ϕ_s is selected from the pool but the evaluation metric on the right-hand side must be computed using the fine-tuned model $\phi_t = T\left(\phi_s, \mathcal{D}_t\right)$, where T represents the model fine-tuning process.

Although fine-tuning a model is much cheaper than training a model from scratch, it can still be computationally expensive. To stick within a computational budget, the brute-force strategy is likely to require a compromise between how many models are considered ($|\mathcal{S}|$) and to what extent the fine-tuning of each model can be optimised (by scanning over training hyperparameters).

Rather than the brute-force strategy, the model selection process can be generalised to:

$$\phi_t^m = \underset{\phi_s \in \mathcal{S}}{\arg\max}\, m\left(\mathcal{D}_t, \phi_s\right) \tag{2}$$

where m is a metric function that acts as a proxy for the usual model evaluation metric $\mathcal{L}$ in Equation 1, with the critical difference being that m is computed using the pre-trained model ϕ_s as input, whereas $\mathcal{L}$ requires the fine-tuned model ϕ_t.

The goal of this report is to explore whether there are metrics m that well approximate the brute-force selection strategy, so that in the ideal case $\phi_t^m = \phi_t^*$, whilst being much faster than having to fine-tune each model. Alternatively, for a given computational budget, a low-cost selection strategy would allow more candidate models to be considered ($|\mathcal{S}|$ larger than for the brute-force strategy with an equivalent budget), meaning ϕ_t^m may identify a different, better-performing model than ϕ_t^*. Further, if a metric can predict the best candidate model using only a subset of the target dataset, $\mathcal{D}_m \subset \mathcal{D}_t$ (replacing $\mathcal{D}_t$ with $\mathcal{D}_m$ in Equaton 2), this would amplify the reduction in the computational cost of m, compared to the brute-force strategy, even further. We refer to $\mathcal{D}_m$ as the metrics or 'probe' dataset.

One aspect of the model selection process not considered in either search strategy above is the time taken to pick a suitable subset of candidate models from the model hub ($\mathcal{S} \in \mathcal{M}$). At the time of writing, there are half a million models available on the HuggingFace hub, including everything from state-of-the-art models uploaded by big tech to toy models uploaded by students following tutorials. There may be many suitable models that match the target task type and pass some simple search criteria based on the popularity of the model and the availability of documentation describing its origin. This may still be too many models to consider in practice, even if using a low-cost selection strategy. Identifying a good set of candidate models $\mathcal{S}$ may involve scanning the literature and checking any available benchmarks or model leaderboards, for example, which can be time-consuming in terms of their human, rather than computational, costs. We do not consider this further in the report, rather focusing on the performance and computational costs of a model selection strategy given a known pool of candidate models.

3.3 Sources of Fine-Tuning Success

Given a pool of models $\mathcal{S}$ on which fine-tuning will be performed, it is likely not entirely random as to which models within that pool will perform best. In particular, the wider literature on transfer learning suggests some clear sources of downstream fine-tuning success. These sources are what the candidate model selection metrics discussed in the remainder of the report must aim to capture.

A well-established finding in the literature is that the similarity of the source dataset $\mathcal{D}_s$ and target dataset $\mathcal{D}_t$ is predictive of successful transfer learning [5, 6]. This relationship is likely due to the fact that where both datasets are similar, the model learned from the source dataset will already be a closer fit to the downstream task. Where all models in $\mathcal{S}$ were trained on the same source dataset however, this will hold constant variation in downstream fine-tuning performance from dataset similarity and thus any variation will come from other sources.

Second, it is likely that architectural differences will play a role in successful transfer. If we consider the focus of this paper - transformers - this architecture has known and desirable scaling properties. This has enabled larger architectures which can take into account more features - it is likely that just as this architecture has resulted in better pre-trained models, these better pre-trained models will in turn produce better downstream performance. If the macro architectural choice of a transformer-based model is held constant in $\mathcal{S}$, then other differences such as the depth or width of the model, number of parameters, relative balance of attention versus convolutional layers will still exist and may still be sources of downstream fine-tuning success.

A third and final difference between the models in $\mathcal{S}$ is the pre-training regime, such as using a different training objective (as in self-supervised approaches) or different optimiser hyperparameters.

4 Candidate Metrics

Several metrics are proposed in the literature on pre-trained model selection. In this section, we provide some categorisations that are useful for differentiating the ways in which the metrics capture downstream fine-tuning performance. We then summarise the most promising metrics from the literature. We conclude this section with discussions on how metrics have been assessed in the literature, which feeds into a summary of the reported performance of the metrics based on these assessment methods.

4.1 Metric Categorisation

The general problem assessed in the literature is to provide a metric that allows for model selection without the need to run expensive training or fine-tuning experiments. These metrics can be usefully classified in many ways which help to differentiate the ways in which they achieve this goal. Here we discuss the various categorisations.

4.1.1 Zero-Cost vs. Low-Cost

The goal of using such metrics is to save on the computational cost of training multiple models. However, there is a distinction between zero (or almost zero) computational cost and low computational cost metrics. Zero-cost metrics are typically heuristics for model performance which require no computation, such as the number of parameters in a model or the reported test accuracy of a model on a pre-training dataset. Low-cost metrics by contrast typically require inference through the model of some or all of a dataset to compute them.

We note that there are other auxiliary costs often associated with zero-cost metrics, such as collecting the test accuracy results for the pre-training dataset, even so we don't consider these computational costs for the purposes of the definition of zero-cost.

4.1.2 Task-Agnostic vs. Task-Dependent

Another way in which metrics differ is in whether they assess models with or without the context of the particular task they will be applied to once trained. Task-agnostic metrics do not take into account the downstream task and so are based on ranking the models without further context. Task-dependent metrics instead rank the models while considering the downstream task, so there is some reason to expect they will capture more sources of fine-tuning success and outperform task-agnostic metrics.

In practice, task-agnostic metrics are zero-cost metrics (such as the number of parameters or the reported test accuracy on the pre-training dataset), while task-dependent metrics are low-cost metrics and as before are often based on performing inference through parts of the model to compute.

4.1.3 Features in the last layer only vs. throughout the Model

Many of the task-dependent metrics in the literature are computed by extracting features from the model having run some kind of inference using the target dataset. These features are representations of the data in some latent space that the model is working in, with the features at different layers of the model containing different information about how the model is processing the data.

Which layer of the model to use to extract feature representations can be considered a design choice. Commonly, the last layer (i.e. the final layer before the classification head) is used. This layer is the most directly related to how the model is attempting to classify an image, and using it doesn't require any architecture-dependent assumptions about the structure of the model in intermediate layers.

4.1.4 Dataset Similarity Metrics

Dataset similarity metrics could be used to compare the pre-training and target datasets, which has been shown to be predictive of downstream fine-tuning performance as discussed in Section 3.3. This can include both model-agnostic metrics (such as those proposed in [5, 7]), which measure the similarity of the pre-training and target datasets only without considering the model architecture, or metrics that compare the behaviour of a pre-trained model on the pre-training and target datasets (such as NCE, which is briefly discussed in Section 4.3.5).

There are several potential pitfalls with this method, however: the pre-training dataset is not always available - for example in the case of models trained on Google's JFT dataset [8, 9]. In this case, dataset similarity metrics cannot be computed. Additionally, model-agnostic dataset similarity metrics do not capture sources of variation related to the model architecture and pre-training, and cannot be used to discriminate between models pre-trained on the same dataset. We therefore choose not to use dataset similarity metrics in our work in this report.

4.1.5 Convolutional vs. Transformer Architectures

The choice of metric is not necessarily dependent on the choice of model architecture. However, taking the example of image classification, many metrics proposed in the literature were developed for and tested on convolutional neural networks (CNNs). When tested on vision transformers, they have either been found to not perform well on transformers, or not apply to transformer architectures due to their construction [10].

This further motivates the LoCoMoSeT project however, as vision transformer models have superseded CNNs in many applications, and as such extending the assessment of the metrics to vision transformers is a novel application.

4.2 Zero-Cost, Task-Agnostic Metrics

4.2.1 Number of Model Parameters

As discussed in [11, 12], the number of parameters of a model can be used as a proxy for the capacity of the model to learn, and its likely achievable performance on, any new task. This has become especially true with the emergence of transformer architectures which have known and desirable scaling properties [13, 14, 15].

Selecting the largest candidate model ignores other architectural differences that impact performance. Further, it does not consider the pre-training regime of the model, such as the pre-training dataset or pre-training objective. Finally, using the largest model in a pool ignores any computational constraints, a factor considered in the literature (although when selecting models from a pool, it makes sense to constrain this pool to only contain plausible options in the first place). We do not consider this constraint in the project, though make a note of this among other constraints to consider in Section 10.

4.2.2 Performance of Models on Pre-Training Dataset

Another task-agnostic metric discussed in [11] is the performance of the models on the pre-training dataset [16, 17]. This can typically be found without access to the pre-training dataset as these values are often reported upon in the literature associated with the model.

Models pre-trained on different datasets cannot be meaningfully compared by the reported test accuracies, and testing on a single dataset's test set (such as ImageNet) is unlikely to be a fair comparison. It is therefore only effective when all models in the pool are pre-trained on the same dataset. However, if all the candidate models are pre-trained on the dataset, reported test accuracy may capture more sources of variation than number of parameters.

4.3 Low-Cost, Task Dependent Metrics

4.3.1 Renggli Metric

We refer to the task-dependent metric in [11] as the Renggli metric. This should not be confused with the fact this paper also discusses task-agnostic metrics. The task-dependent Renggli metric performs a forward pass of the new dataset through a model, and collects the output features prior to the classification head. It then fits a linear or k-nearest neighbours classifier on top of these embeddings, and evaluates them on the target labels. For our implementation of the Renggli metric we use a linear classifier (as recommended by the original paper as both choices performed similarly). The accuracy[2] of this classification is the Renggli metric.

4.3.2 Pairwise Annotation Representation Comparison (PARC)

The Pairwise Annotation Representation Comparison (PARC) metric [12] builds on a previous metric, RSA [18], which compares the features from the last layer of a model to labels predicted by a probe model (typically a model with a retrained classification head) trained on the data.

In PARC the predicted labels are replaced by the ground truth labels, allowing for this metric to be applied with no training necessary. Formally, given a labelled dataset $\mathcal{D} = \{x_i, y_i\}$ - for data x_i and labels y_i - PARC takes a subset of the full dataset, $\mathcal{D}_m \subset \mathcal{D}$ - where $|\mathcal{D}_m| = n$ - of n (data, label) pairs and computes the following $n \times n$ distance matrices:

$$D_x := 1 - \text{corrcoef}(f_i), \qquad D_y := 1 - \text{corrcoef}(g_i), \tag{3}$$

where $f_i = \phi(x_i)$ represents the extracted features from the last layer of a model ϕ, g_i represents the label y_i projected to a one hot vector, and corrcoef$(\cdot)$ is the Pearson correlation coefficient for every pair of features (f_i, f_j) or labels (g_i, g_j) in the metric data subset $\mathcal{D}_m$. These distance matrices describe the geometry of the features and the labels, which PARC then compares using Spearman's rank correlation:

$$\text{PARC}(\mathcal{D}_m) = \text{spearmanr}(\{D_z[i,j] \mid i < j\}, \{D_c[i,j] \mid i < j\}). \tag{4}$$

This metric suggests that the model is likely to perform well if the geometry of the features extracted via inference prior to training correlate strongly with the geometry of the ground truth labels. The PARC paper also suggests implementing dimensionality reduction (PCA) on the features, in an effort to have a more consistent measure over a set of models that may vary widely in the dimension of their features.

To extract the features from the model for the computation for the PARC metric (equation (4)) the original authors use global pooling [19] on the final layer of the CNN models considered. In our implementation, we extract the features from the last layer (prior to the classification head) of the ViT models, and then reduce them to common dimension with PCA.

[2]Other performance metrics could plausibly be used here.

4.3.3 Topological Data Analysis (TDA) of Decision Boundaries

Topological data analysis looks to compute a measure called the topological complexity for both the labelled target dataset $\mathcal{D} = \{(x_i, y_i)\}$, for data x_i and labels y_i, and the outcome of a model $\phi(\cdot)$, $M = \{(x_i, \phi(x_i))\}$. This approach is particularly powerful when the topology of the data is a key part of the analysis, for example when considering problems on networks. The methods can be applied to other domains, but are typically considered experimental in this case [20].

In the case of model selection, the suggestion is that if the topological complexity of the features (or labels) extracted from the model for a given target dataset is similar to topological complexity of the dataset itself then the model will be a good choice for transfer learning [21, 22].

The process that we use for our TDA approach is as follows:

1. Preprocess a subset of the dataset images and extract the pixel values. For each array of pixel values create the set $\mathcal{D}$ by appending a one-hot vector representing a label y_i to the pixel array. For each model extract either the features of the last layer prior to the classification head, or the predicted labels, via inference and similarly append the pixel arrays with the relevant one hot vectors to create the set Z_s.
2. Use a topological method known as the Cubical Homology [23] to create a set of homology diagrams for each set, giving the information necessary to compute the topological complexity.
3. Using these diagrams a vector representation of the topological complexity is computed for both the dataset and each model (using either the predicted labels or the features).
4. Finally, the Euclidean distance between the dataset vector and each model vector is computed, with models that have a topological complexity vector closer to the dataset vector being predicted to have stronger post-fine-tuning performance.

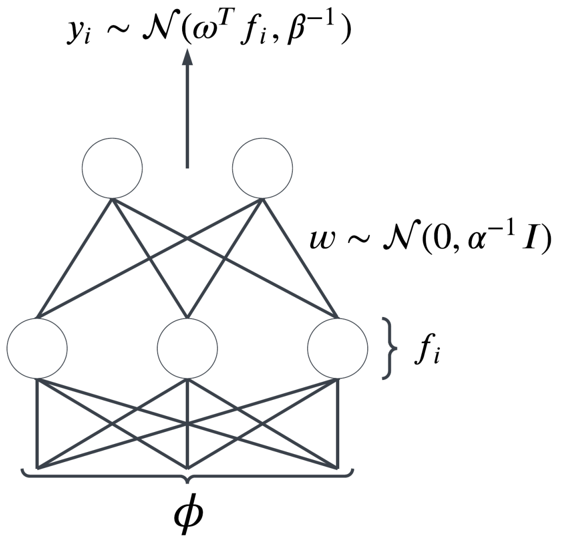
This differed slightly to the original approach in [21]. First, we used an alternative, more computationally efficient, implementation of the homology computation method [23]. Second, the original paper considered only binary classification, so we had to use a one-hot vector to represent the label information rather than a singular value.

4.3.4 Log Maximum Evidence (LogME)

The LogME (or Log Maximum Evidence) metric forms part of a larger scheme found in papers [24, 25] to utilise model hubs (or model zoos as it is referred to in the papers), with the metric being part of the assessment pipeline. Whilst the primary application of the LogME metric is to image classification tasks, it is also applied to regression and natural language processing tasks in the literature.

Starting with a collection of models, $\{\phi_m\}_{m=1}^{M}$ and a target dataset $\mathcal{D} = \{(x_i, y_i)\}_{i \in [n]}$ LogME returns a score for each model, by which they can be ranked. It should be noted that LogME is not on an absolute scale (unlike that of the Renggli or PARC metrics) and as such the scores only make sense within the context of ranking a collection of models.

Given the dataset and a chosen model, LogME extracts the features $f_i = \phi(x_i) \in \mathbb{R}^d$, where d is the dimension of the feature vectors, from the last layer of the model, combining them in a feature matrix $F \in \mathbb{R}^{n \times d}$, where n is the size of the dataset. It then looks to estimate a value for $p(y \mid F)$, the probability that the vector containing all the labels, $y \in \mathbb{R}^n$, will be returned given the features F, a direct measure of the compatibility of the two. This is done by estimating the performance of a linear classifier head, a graphical representation of which can be seen in figure 1.



$y_i \sim \mathcal{N}(\omega^T f_i, \beta^{-1})$
$w \sim \mathcal{N}(0, \alpha^{-1} I)$
f_i
ϕ

Figure 1: A schematic of how the LogME metric estimates the performance of a model on a downstream task. Here f_i are the output features of the model ϕ for each datapoint i, , with F being the feature matrix containing all features, and corresponding ground truth labels y_i. The weights, w, corresponding to the classification head are treated as being picked from a normal distribution with a variance parameterised by α, with the labels treated as being picked from a normal distribution centred on $w^T f_i$, with a variance parameterised by the parameter β.

Typically the likelihood, $p(y \mid F, \omega^*)$, would be used to estimate this quantity, however this has been shown to overfit, so instead LogME turns to the evidence:

$$p(y \mid F) = \int p(w)p(y \mid F, w)\mathrm{d}w. \tag{5}$$

By assuming that the values of the parameters w are picked from a normal distribution, with variance controlled by a parameter α, and the labels from a normal distribution centred on $w^T f_i$, with variance controlled by a parameter β, a closed form for $p(w)$ and $p(y \mid F, w)$ can be given:

$$p(w) = \mathcal{N}(0, \alpha^{-1}\mathbb{I}), \qquad p(y_i \mid f_i, w) = \mathcal{N}(w^T f_i, \beta^{-1}). \tag{6}$$

From this an expression for the value $\mathcal{L}(\alpha, \beta) = \log p(y \mid F, \alpha, \beta)$ can be derived. By the use of a fast-converging fixed-point iteration algorithm, LogME efficiently estimates the maximum evidence, returning the following as a final metric score:

$$\text{LogME}(\phi, \mathcal{D}) = \max_{\alpha,\beta} \mathcal{L}(\alpha, \beta). \tag{7}$$

By doing this LogME doesn't ever implement a linear classification head, instead estimating the performance of one. It should be noted that the original paper only considers using LogME on an entire dataset; however they do not make strong claims that this is necessary.

The original authors of the LogME paper provide an efficient implementation of the LogME metric [26, 24, 25], which we used.

4.3.5 Other Task Dependent Metrics

There are many more metrics found in the literature that were discovered during this project, here we briefly mention several that were potential candidates.

NCE The negative conditional entropy (NCE) metric [7] uses a measure of conditional entropy between the pre-training dataset for a given model and the target dataset. This can be considered as falling into the category of dataset similarity measures [5]. The method employed includes retraining a new classifier head on the given model and comparing the entropy of the labels from the pre-training dataset and the target dataset, conditioned on the predicted labels for the target dataset from the new model. This metric would be of particular use if the pool of models had variation in the pre-training dataset, and access to the pre-training datasets were available in each case.

LEEP The log expected empirical prediction (LEEP) metric [27] is widely considered in the literature, frequently as a point of comparison for the performance of subsequent metrics. To compute this metric a distribution by extracting the softmax layer from the original classification head of the model for each image, which is then combined with the ground truth labels of the target dataset to produce the empirical estimator.

H_α-score H-score [28] is an information-theoretic approach to assessing transferability compatibility between models and target datasets. It captures the ability of a model to discriminate between classes by extracting the features from the last layer of a model and comparing the covariance of the features and the features conditioned on the labels. The authors of [29] claim that the poor performance of H-score found in the literature is not due to a fault in the method of the metric, but instead a lack of robustness in estimating the metric leading to numerical errors. They propose a regularisation technique to overcome this, with an adapted metric H_α-score claiming much-improved results.

4.4 Combinations of Metrics

Several papers in the literature additionally consider whether there is any benefit in combining metrics to improve performance.

[11] find that neither task-agnostic nor task-dependent metrics are consistently better across all scenarios, and so recommend combining both approaches as they capture different information. They suggest doing this by selecting multiple models to fine-tune, such as the best model according to the task-agnostic metric and the best model from the task-dependent metric.

[12] scale the PARC metric score according to the depth of the (CNN) models they examine. This serves as a proxy for the capacity of the models to learn, much in the way the number of parameters above serves as a task-agnostic metric. In this sense, this is also a process of combining two metrics, albeit in this case instead of selecting multiple models the metrics are merged to produce a new score. They find that this improves the performance of both PARC and other task-dependent metrics considered in the paper.

4.5 Assessing Metric Performance

Within the literature, several methods of assessing metric performance are proposed, with the majority being based on the correlation between the metric scores and post-fine-tuning validation or test accuracy scores.

4.5.1 Correlation

Where the correlation methods are concerned, typically either Pearson's, Spearman's rank or Kendall's tau rank correlation is employed, with Kendall's tau used especially in examples with small sample sizes since the other two become numerically unstable in these regimes [30, 31].

Here, the correlation will capture how well the metric captures not only the downstream fine-tuning performance of the models on the dataset, but how well the metric captures the differences between models. Where Spearman's rank or Kendall's tau correlation are used [32], this becomes an assessment of how well the metrics capture the rank ordering of the model's downstream fine-tuning performances.

The PARC paper [12] introduces the idea of the mean Pearson correlation to average the correlation between the metric scores and the fine-tuning validation accuracy across different datasets.

4.5.2 Regret

Renggli et. al. [11] examine a slightly different task to the one considered here, looking at search strategies (metrics) that return a set number of models, B, as opposed to a ranking or returning just the predicted best model. With this in mind, for a given set of models to search over, $\phi_i \in \mathcal{M}$, a dataset $\mathcal{D}$, and a search strategy $m(\mathcal{M}, \mathcal{D}) = \mathcal{S}$ which returns a subset of the best models predicted by a metric, $\phi_i \in \mathcal{S} \subset \mathcal{M}$, they introduce the notion of 'regret', defined as the difference in performance between the best model overall and the best-selected model:

$$\max_{\phi_i \in \mathcal{M}} \mathbb{E}_{FT}\left[t(\phi_i, \mathcal{D})\right] - \mathbb{E}_m\left[\max_{\phi_i \in \mathcal{S}} t(\phi_i, \mathcal{D})\right], \tag{8}$$

where $t(\phi_i, \mathcal{D})$ denotes the post fine-tuning test accuracy of a model ϕ_i on the dataset $\mathcal{D}$. The expectations are taken over different distributions, with $\mathbb{E}_{FT}$ taking into account the randomness inherent in fine-tuning, and $\mathbb{E}_m$ taking into account any randomness apparent in the computation of the metric m.

The authors claim that regret is more applicable for assessing the efficacy of metrics than simply comparing correlation values as it still functions at various edge cases, for example when the fine-tuned test accuracy scores are all comparable and therefore uncorrelated with the metric ranking. This can be seen by the fact that perfect correlation would imply zero regret, however zero regret does not imply perfect correlation.

Regret captures how well the metric in question selects the model with the best downstream fine-tuning performance, whereas correlations examine the overall relationship between the metric value and downstream fine-tuning performance across all the models.

4.6 Reported Metric Performance

There are clear limitations on what we can draw from the reported performance in the original publications of the metrics presented in Sections 4.2 and 4.3. Many of these papers assess different pools of models and different downstream fine-tuning tasks. Moreover, there is no standardisation in which metrics are used, and so results cannot necessarily be compared to one another even if the former concern were ignored.

It should also be highlighted that many of the metrics in the literature were tested on CNN architectures and are thus untested against transformer architectures. Nonetheless, it is still useful to understand how the metrics have performed in various settings. Whilst task-dependent metrics are typically computationally costlier than their task-agnostic counterparts they outperform the task-agnostic metrics in the literature. NCE [7] and LEEP [27] are both frequently compared to, and beaten by, other metrics in the literature.

A longer discussion of the reported performance of the metrics can be found in Appendix A.

5 Metric Selection

Before implementing the full fine-tuning experiments we chose a subset of the candidate metrics based on an initial assessment of their efficacy. To do this, we selected a pool of models pre-trained on ImageNet-1k, computed the metrics for each of those models on the ImageNet-1k validation set, and compared the metric scores to the reported ImageNet-1k top-1 accuracy in each model's documentation.

This enabled us to examine the viability of the metrics without needing to fine-tune each model on a new target dataset. If a metric is unable to rank models by their performance on the task they have already been pre-trained on, it's unlikely they can be predictive of performance on a different target dataset. Details of these exploratory checks can be found in Appendix C.

The final selection of metrics that we decided to take forward to the full fine-tuning experiments on new target datasets can be found in Table 1 with details of their metric classification. Alongside the number of model parameters, we also include the validation accuracy of the models on their pre-training dataset (ImageNet-1k for all of the models we selected) as a further task-agnostic metric as this provided a convenient benchmark to compare the other metrics against.

In addition to opting not to continue with the PARC and TDA metrics for both performance and theoretical reasons based on the exploratory checks, we decided not to continue with NCE, LEEP or H_α-score either. LEEP and NCE are out-performed by the other metrics in the literature, and NCE additionally requires access to the pre-training dataset which we had stipulated as something to be avoided. H_α-score did not have an implementation available.

Metric	Zero vs. Low	Agnostic vs. Dependent	Features	Pre-training Dataset
Renggli	Low cost	Task dependent	Last layer	No
LogME	Low cost	Task dependent	Last Layer	No
No. Parameters	Zero cost	Task agnostic	N/A	No
ImageNet-1k val. acc.	Zero cost*	Task agnostic	N/A	No*

Table 1: Final metric selection, with details on which type of metrics these are. *As discussed in Section 4.2.2, the pre-training dataset validation accuracy is only zero-cost if it is reported in the model's documentation (otherwise it would be high-cost and require access to the pre-training dataset to compute).

6 Experiment Design

To evaluate the performance of the LoCoMoSeT metrics we have run a set of experiments in which we fine-tune a pool of models on several target image classification datasets, and compare their fine-tuned accuracy to their predicted rankings via our chosen metrics. The details of the experimental setup are described in this section.

6.1 Datasets

We have selected four target datasets with implementations available on the HuggingFace datasets hub [4] – Oxford Pets [33], WikiArt [34], RVL-CDIP [35], and Visual Genome [36], chosen to capture a variety of domains, dataset sizes, and numbers of classes. Table 2 lists some high-level details for each dataset, with Figure 2 showing examples from each dataset.

The task on the Oxford Pets dataset is to classify an image of one of 37 breeds of dog or cat. As the style of imagery and some of the classes overlap with ImageNet (the pre-training dataset for the models), we may expect this to be the most straightforward dataset for the metrics to be predictive of model performance after fine-tuning.

The remaining datasets are more distinct from ImageNet. The WikiArt dataset contains images of artworks with the task being to classify it as belonging to one of 129 artists. RVL-CDIP (Ryerson Vision Lab Complex Document Information Processing) contains greyscale scans of one of sixteen types of documents, such as letters, advertisements, and invoices. Our use of Visual Genome is described in the next subsection.

Name	**Classes**	**Total Images**	**Train/Metrics Subsets**	**Test Size**	**Source**
Oxford Pets	37	7390	500, 1000, 4803	1848	[33]
WikiArt	129	81444	500, 1000, 5000, 10000, 50000	20361	[34]
RVL-CDIP	16	400000	500, 1000, 5000, 10000, 50000	40000	[35]
Visual Genome Tree	2	40005	500, 1000, 5000, 10000, 26004	10001	[36]
Visual Genome Faucet	2	3543	500, 1000, 2303	886	[36]
Visual Genome Watch	2	4345	500, 1000, 2825	1086	[36]

Table 2: The datasets used in the experiments. The columns are: **Name:** The name used to describe the dataset in the report, **Classes:** The number of possible image class labels in the dataset, **Total Images:** The total number of images in the whole dataset, **Train/Metrics Subsets:** The sizes of the subsets that we have run experiments on for each dataset, both for fine-tuning models (the number of images in the training set) and computing metrics (the number of images used to compute the metric), **Test Size:** The number of images in the test set used to evaluate the fine-tuned models, **Source:** Citation for the original dataset author and the HuggingFace dataset.

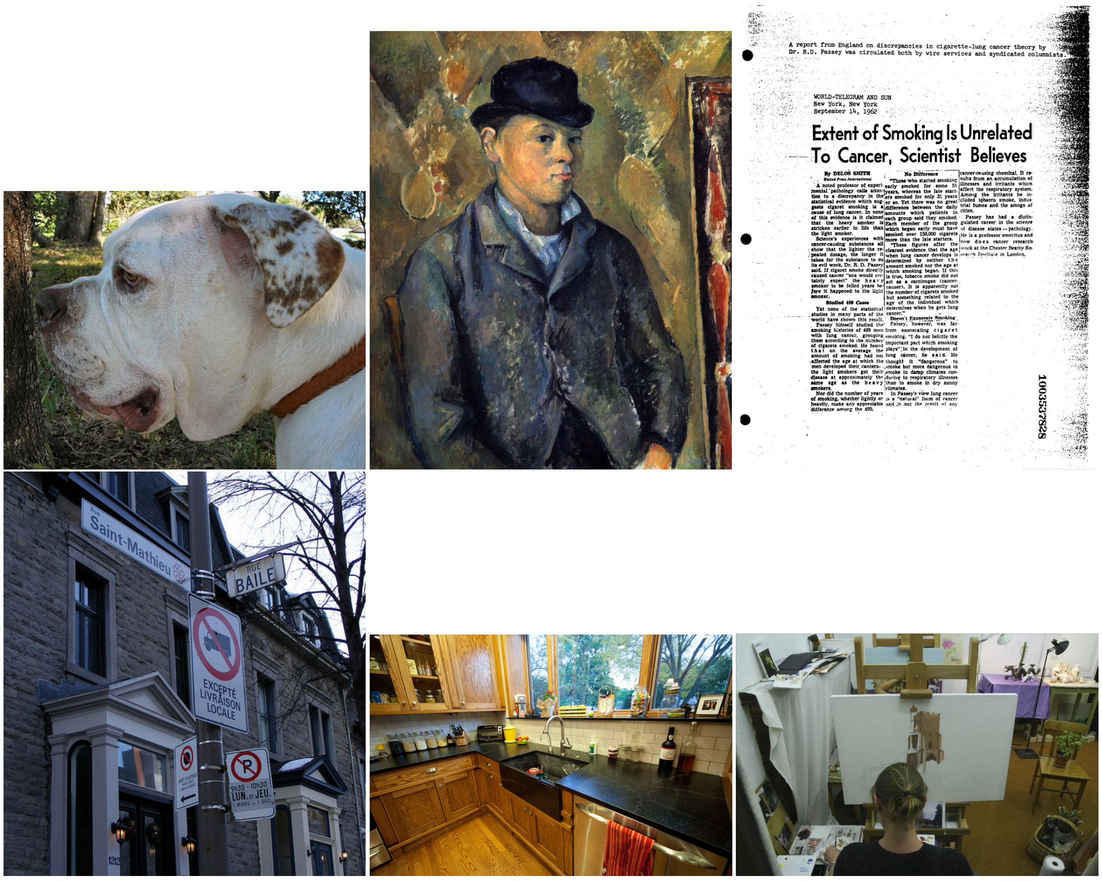

A report from England on discrepancies in cigarette-lung cancer theory by Dr. R.D. Passey was circulated both by wire services and syndicated columnists.
WORLD-TELEGRAM AND SUN
New York, New York
September 14, 1962
Extent of Smoking Is Unrelated To Cancer, Scientist Believes
1003537828
Saint-Mathieu
RUE BAILE
EXCEPTE LIVRAISON LOCALE

Figure 2: Example images from each dataset. Along the top row, from left to right, are examples from Oxford Pets [33], WikiArt [34] and RVL-CDIP [35]. Along the bottom row are examples from the tree, faucet and watch binary datasets created from Visual Genome [36].

6.1.1 Binary Classification Datasets

The Visual Genome dataset (VG) [36] is a common object detection dataset, with photographs of urban and natural environments annotated with an average of 35 object bounding boxes per image, as well as further object attributes and relationships between objects in the image. We created three binary classification datasets from VG, as described below. Compared to the Oxford Pets dataset, where the class of interest is always the focus of the image, the class to be identified in the VG-derived datasets may only be a small object in the background, see Figure 2 for a comparison. They therefore represent datasets with an overlap in the style of imagery with ImageNet (photography of natural and urban environments), but where the task is quite different and potentially more difficult.

Three objects were selected to create binary classification datasets, with a variety in prominence of the objects in the image (where prominence here is defined as the percentage area of the whole image that the bounding box of the target object fills), and a variety in the number of images containing the objects. The chosen objects were: watch (average prominence of 1.32%), tree (average prominence: 25.8%), and faucet (average prominence 2.67%). Further details for the binary datasets are given in Table 2. To create the final datasets for each object, we combine all the images containing that object with an equal number of images not containing the object, to create balanced binary classification datasets.

6.1.2 Training Dataset Splitting & Subsetting Strategy

For the Oxford Pets, WikiAt, and Visual Genome-derived datasets we have used a 65% / 10% / 25% train/validation/test split. The validation set is used only as a debugging tool to monitor the progress of the model fine-tuning, so we favoured a larger test set. We use a stratified split for all random sampling so each subset has a similar distribution of classes to the original parent dataset, but only run the experiments with subsets created from a single random seed. The RVL-CDIP dataset comes with pre-defined training, validation, and test splits so we use those directly.

In our experiments, we further subset the training datasets to obtain a range of results at different training dataset sizes. For each dataset, we fine-tune models at training subset sizes of 500, 1000, 5000, 10000, and 50000 images, and also with the whole training set if it is smaller than 50000 images (see Table 2). We have not run experiments with training set sizes larger than 50000 samples within the scope of this project. All experiments on each dataset use the same test set to compute the achieved fine-tuned test accuracy for each model.

6.1.3 Metrics Dataset Subsetting Strategy

The LoCoMoSeT metrics are computed using images from the training subsets only, excluding images from the validation and test sets. As the computational cost of the task-dependent metrics is determined by the number of images used to compute them, we additionally compute the metrics using subsets of images from the training datasets, thereby allowing an analysis of how many images each metric requires to be predictive of the model's fine-tuned performance (see Section 7.3.2).

For example, for a training dataset size of 10000 images, we compute the metrics with 500, 1000, 5000, and all 10000 of those images. Notating the whole parent training dataset as $\mathcal{D}_{\text{parent}}$, the training subset used for an experiment as $\mathcal{D}_t$, and the dataset used to compute the metrics in an experiment as $\mathcal{D}_m$, the different datasets relate to each other as $\mathcal{D}_m \subseteq \mathcal{D}_t \subseteq \mathcal{D}_{\text{parent}}$.

A visual representation of this can be seen in Figure 3. Here the white boxes represent the whole parent training dataset, $\mathcal{D}_{parent}$, the blue boxes represent the training subset $\mathcal{D}_t$ used for an individual experiment, and the yellow the further subset of the dataset used to compute the metrics $\mathcal{D}_m$. For example, the top right box in the figure represents the experiment where the model is fine-tuned on the whole parent training set, whilst the metrics are computed on a much smaller subset. This is the configuration where using the metrics would give the greatest reduction in computational cost. Experiments were run for every permutation of the train and metrics subset sizes listed in Table 2.

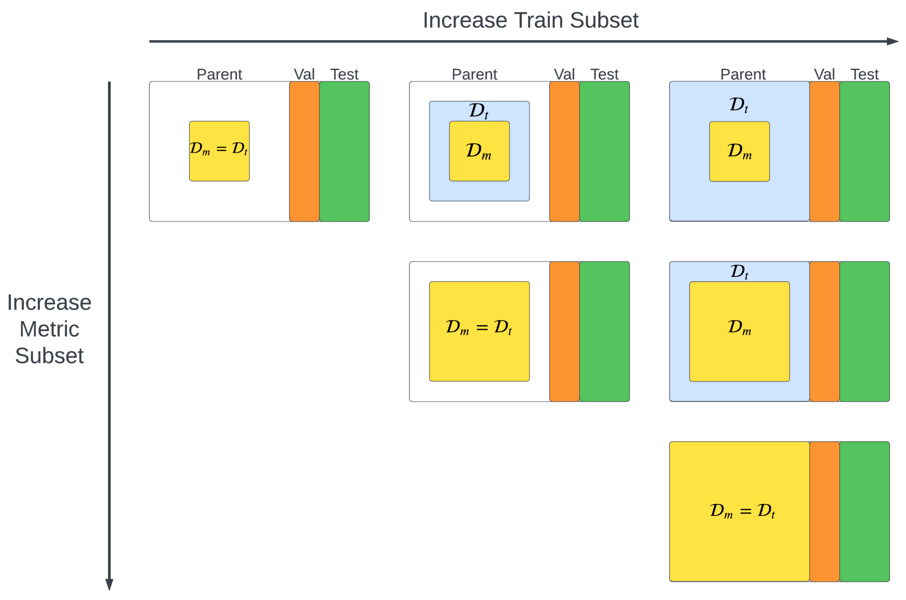


Figure 3: Schematic for dataset subsetting for both training and metric dataset subsets. The training subset (blue) is always a subset of the parent training dataset, with the metric subset (yellow) always a subset of the training subset. An experiment was run for subsetting strategies represented by the different boxes.

Name	**Resolution**	**Params**	**ImageNet**	**Ref**
apple/mobilevit-xx-small	256	1	69.0	[37]
apple/mobilevit-small	256	5	78.4	[37]
facebook/deit-tiny-patch16-224	224	6	72.2	[38]
facebook/deit-small-patch16-224	224	22	79.9	[38]
facebook/deit-base-patch16-224	224	86	81.8	[38]
facebook/deit-base-patch16-384	384	86	82.9	[38]
facebook/dinov2-small-imagenet1k-1-layer	224	22	81.1	[39]
facebook/dinov2-base-imagenet1k-1-layer	224	87	84.5	[39]
google/vit-base-patch16-384	384	86	85.5	[3]
google/vit-large-patch16-224	224	303	84.0	[40]
microsoft/beit-base-patch16-224	224	86	85.2	[41]
microsoft/beit-base-patch16-384	384	86	86.8	[41]
microsoft/beit-large-patch16-512	512	305	88.6	[41]
microsoft/cvt-13	224	20	81.6	[42]
microsoft/cvt-21-384	384	31	83.3	[42]
microsoft/swin-base-patch4-window12-384	384	87	86.4	[43]
microsoft/swin-large-patch4-window12-384-in22k	384	195	87.3	[43]
microsoft/swinv2-base-patch4-window8-256	256	87	84.2	[44]
microsoft/swinv2-large-patch4-window12to16-192to256-22kto1k-ft	256	195	86.9	[44]
Zetatech/pvt-large-224	224	61	81.7	[45]

Table 3: The 20 models used for the experiments. The meanings of the columns are: **Name:** The path to the model on the HuggingFace model hub [4], **Resolution:** The image input size for the model (the side length in pixels, all models use square, 3 channel inputs), **Params:** The number of parameters in the model in millions, **ImageNet:** The claimed ImageNet-1k top-1 accuracy of the model, **Ref**: Citation to the original publication for the model architecture.

6.2 Models

We focused our experiments on models available in the HuggingFace model hub that are based on the transformer architecture (i.e. vision transformers) and pre-trained on ImageNet. ImageNet is the most common pre-training dataset for image classification models, so our experiments with this fixed likely relate to the most common scenario for transfer learning. However, this restriction means we do not consider the impact of similarity between the pre-training dataset and the target dataset in model selection, which we leave as a possible extension of the project (see Section 10). We also don't include models based on other architectures, notably convolutional neural networks (although some of the included models do have convolutional components), as the LoCoMoSeT metrics have been tested on these in the prior literature. However, we cannot make claims about the ability of the metrics to generalise to mixed-architecture model pools based on our experiments.

With these conditions, there was a remaining pool of around 80 models to consider with the appropriate pre-training dataset and model library tags to appear in search results. From these, we selected a subset of 20 models that a) compatible for fine-tuning with an unmodified template HuggingFace transformers pipeline (see Appendix B for further details on models we excluded after initial testing), and b) captured a variety of model families, sizes (by number of parameters) and input image resolutions. The model pool is listed in Table 3. Where multiple users or organisations have uploaded models from the same family we favoured those from the original publisher. The models vary in size from 1–305 million parameters, have input resolutions between 224 x 224 and 512 x 512 pixels, and have claimed ImageNet-1k top-1 accuracies between 69% and 87%. They come from eight main families, as described below:

ViT (Google) [3, 40] The first publication of the vision transformer architecture. Images are split into square patches, each of which is flattened, transformed via a learnt linear embedding, and combined with learnt position embeddings. The patch embeddings, prepended with a class token embedding, are passed to multiple layers of conventional transformer encoder blocks. The output embedding of the prepended class token is passed to a classification head (fully-connected network with a single hidden layer in the original implementation).

DeiT (Facebook) [38] Data-efficient image transformers (DeiT) uses the same architecture as ViT but adds a learnt distillation token in addition to the class token from ViT (but appended to the end rather than the beginning of the input sequence). The final output embedding of the distillation token is passed to a secondary classification head which is used to predict the label assigned by a teacher model (chosen to be a convolutional neural network in this case). At inference time the secondary classification head can optionally be used in combination with the usual head for the class token from ViT. The implementation of DeiT that we are using uses the class token only.

PvT (Nanjing University) [45] Pyramid vision transformers (PvT) modify the ViT architecture to consider features at different scales in a series of stages. The early stages use small (e.g. 4 x 4 pixel) input patches to learn fine-grained local features in the image, which are particularly useful in the context of pixel-level tasks (such as object detection). The patch size increases and patch sequence lengths decrease in each subsequent stage of the model to reduce computational cost (whilst retaining the information from the fine-grained features from earlier stages).

Swin and Swin V2 (Microsoft) [43, 44] The **S**hifted **Win**dow transformers (Swin) architecture modifies the self-attention mechanism so it is only computed between local regions of patches (windows), not between all input patches. In each subsequent layer, the windows are shifted so that regions overlap and information can be shared across the whole image. Patches are also merged at several points in the model to create a hierarchy whereby early layers are at fine-grained resolution (4 x 4 pixel patches like PvT) with the output resolution decreasing (and patch size increasing) in later layers. Rather than using a separate class token, Swin pools the representation of all output patches and passes this to a classification head. Swin V2 makes improvements to training stability and adds a self-supervised pre-training regime to reduce the quantity of labelled data required.

CvT (Microsoft) [42] Convolutional vision transformers (CvT) apply 2D convolutional operations to the patch embeddings and in the self-attention blocks, which usually operate only with flat sequences of patches, aiming to create an architecture merging the strengths of CNNs and ViTs. The use of convolutions re-introduces the inductive biases of CNNs and means CvT does not require the use of position embeddings. The embedding of a class token is still used as the input to a classification head as in ViT, but the class token is not introduced until the last block of the model rather than at the start.

DINOv2 (Facebook) [46, 39] Self-**di**stillation with **no** labels (DINO) uses student and teacher networks, like DeiT, but in DINO the teacher network is created from previous iterations of the student network. Both networks use the original ViT architecture. DINO uses a self-supervised training regime, in which different augmentations (crops, distortions, and similar) of an unlabelled image are passed to each network and the student's parameters are updated to minimise the difference between its output representation of the image and the teacher network's representation. DINOv2 makes modifications to improve training stability and efficiency at larger dataset and model scales.

BEiT (Microsoft) [41] BERT Pre-Training of Image Transformers (BEiT) employs a different self-supervised pre-training approach using masked image modelling. Before pre-training, an image tokenizer is learnt that represents an image as a number of discrete tokens. During pre-training, a ViT architecture is trained to predict the tokens associated with the image but with some of the input image patches masked. Like Swin, when fine-tuned for downstream tasks BeiT uses the average output representations of all patches as input to the classification head, rather than a separate class token.

MobileViT (Apple) [37] MobileViT is another hybrid architecture drawing inspiration from both transformers and convolutional neural networks, but optimised for running within the constrained resources of mobile devices. Where CvT could be described as a ViT with some added convolutional components, MobileViT is primarily a CNN architecture but with some convolutional layers modified to use self-attention to be able to learn global image representations in shallower and narrower networks than conventional CNNs.

6.3 Fine-Tuning Strategy

As the experiments require fine-tuning 20 models on 6 datasets, each at multiple data subset sizes, they have a high computational cost and it was not feasible to do hyperparameter tuning in this project. Although this risks us not achieving the optimal performance possible for each model, fine-tuning is expensive and it often is not possible to do extensive hyperparameter tuning in practice. By keeping the training strategy fixed our experiments partially capture the ability of the metrics to select a model that is 'easy' to train to a high level of performance, which may be more suitable than a model that has higher potential but is difficult to train in some circumstances.

We use the HuggingFace transformers trainer [47] with default arguments. These include the use of the AdamW optimiser [48], a learning rate of 5×10^{-5} with a linear scheduler (linearly decreasing during training), and a batch size of 8. We made one exception to this for DINOv2, which we found would not train successfully with the default learning rate for all the variants of DINOv2 we used. For DINOv2 we therefore reduced the initial learning rate to 5×10^{-6}, keeping all other parameters the same (including the linear scheduler on the learning rate).

Training is run for 10 epochs with evaluation on the validation set at the end of each epoch and once on the test set at the end of training. All images are pre-processed and cached locally before starting training so any operations (such as resizing) required for compatibility with the model's expected inputs are run only once.

In all cases, we are fine-tuning all parameters of the models, which is usually the suggested strategy for transformers in the literature but may not be optimal for the smaller training dataset sizes [49]. We also have not included a data augmentation strategy, which similarly may impact model performance on particularly the smaller training datasets. Model fine-tuning strategies for vision transformers is another area where we feel there is scope for follow-on work.

6.4 Metrics Computation

The metrics are computed on identical hardware to the model fine-tuning. Extracting image features in the penultimate layer of each model is run on GPU using a modified HuggingFace inference pipeline for image classification, and uses the same batch size as fine-tuning. The LoCoMoSeT metrics themselves are computed on CPU (using the extracted image features as input).

6.5 Hardware

All the fine-tuning and metrics experiments are run using a quarter of one compute tray of the Baskerville HPC [50], namely one Nvidia A100 GPU, 128 GB DDR4 RAM, and 18 cores of an Intel Xeon Platinum 8360Y CPU.

7 Results

We assess the performance of the metrics using two main criteria, the Kendall's Tau rank correlation and the regret score, which are defined more formally in Section 4.5. The Kendall's Tau correlation measures the ability of the metrics to successfully rank the whole pool of model scores by their achieved fine-tuned test accuracies (higher correlation values correspond to better metric performance), while the regret score measures the difference in performance between the model that would have been selected using one of the metrics and the actual best model overall (lower regret values correspond to better metric performance).

For reference, Appendix D.1 includes plots of the raw data for the experiments with the highest and lowest correlations for each metric, which can help to give further context to interpret the results presented in the main text.

7.1 Performance Evaluation

7.1.1 Results at Maximum Dataset Sizes

We begin by exploring the subset of experiments using the largest version of each dataset, and with the metrics also being computed using the whole training dataset. For example, for the Oxford Pets dataset, this means considering the case where the training dataset has 4803 images and the metrics are also computed using all of those 4803 images (see Table 2 for the other dataset sizes). This likely represents the best-case scenario for the task-dependent metrics, as it maximises the access they have to the target dataset.

In Table 4a the datasets are varied across rows, the metrics across columns, and Kendall's Tau rank correlation between the metrics and fine-tuned test accuracy for each combination in the cells. The metric with the strongest correlation for each dataset is highlighted in bold. In addition, we present each correlation with a bootstrapped 95% confidence interval to give some sense of the uncertainty. Note that there is large uncertainty on correlation coefficients computed on a small number of samples (20 points in this case, one per model).

With some exceptions, the metrics show a significant positive correlation with the achieved fine-tuned test accuracies across all the datasets. Renggli appears to be the least stable and has zero correlation with fine-tuned accuracy for the Visual Genome Watch dataset as well as three confidence intervals for other datasets that have negative lower bounds. Although the confidence intervals overlap so it's not possible to make a fully substantiated claim, the task-agnostic metrics (number of parameters and ImageNet validation accuracy) report the highest correlations on five out of the six datasets.

Table 4b shows the regret values of the same experiments, expressed as the difference in the test accuracy percentage of the best model and the model that would have been selected by the metric. As the largest model in our model pool (microsoft/beit-large-patch16-512) also has the highest reported ImageNet validation accuracy, the number of parameters and ImageNet validation accuracies always select the same model and so share the same regret value. They also select BEiT large across all datasets as, by definition, the task-agnostic metrics do not depend on the target dataset.

For regret, it appears the task-dependent metrics select the best model (regret value of zero) most often, wtih LogME correctly identifying the best model for 3 out of the 6 datasets, for example. However, they also appear more culpable of making large mistakes, with a maximum regret value of 4.93 for the task-dependent metrics but only 1.62 for the task-agnostic metrics. Overall, there is no evidence to favour any of the metrics over any of the others based on the regret values for this subset of experiments.

7.1.2 Aggregated Results

Table 5 summarises the mean correlation and regret values across all the datasets and at all training subset sizes. For example, for the Oxford Pets dataset, this means aggregating three results - where the models are fine-tuned using 500 images, 1000 images, and 4803 images. The values in Table 5 are the mean across the 24 different subset sizes (across the 6 datasets) listed in the dataset details in Table 2.

Overall, comparing Table 5 to Table 4a, the results are similar to when considering only the maximum dataset sizes. There is evidence that the task-agnostic metrics outperform the task-dependent metrics in recovering the overall ranking of the models (by Kendall's Tau rank correlation), and also that LogME outperforms Renggli amongst the task-dependent metrics. Alternatively, in terms of regret, there is no evidence to favour any of the metrics over the others, although it's interesting to note the Renggli score has the best (lowest) mean regret despite it having the lowest mean correlation.

Given there is no evidence to suggest favouring the higher-cost task-dependent metrics overall, these findings suggest recommending a model selection strategy using a zero-cost task-agnostic metric, with no strong preference between number of parameters and ImageNet validation accuracy.

For reference, Appendix D.2 contains results at each training subset size for each dataset (again limited to the case where the metrics are computed on the whole training subset).

Table 4: Metric Performance per Dataset at Largest Training and Metrics Dataset Sizes

(a) Metric Performance Assessed by Correlation

Dataset	Renggli	LogME	N. Params	ImageNet Acc.
VG Faucet	0.25	**0.52**	0.38	0.38
	(-0.19, 0.60)	(0.09, 0.79)	(0.05, 0.66)	(-0.02, 0.69)
VG Tree	0.38	0.32	0.50	**0.50**
	(-0.06, 0.65)	(-0.11, 0.62)	(0.12, 0.72)	(0.22, 0.72)
VG Watch	0.04	0.50	0.59	**0.71**
	(-0.41, 0.43)	(0.12, 0.73)	(0.20, 0.79)	(0.45, 0.86)
RVL-CDIP	0.55	0.49	0.65	**0.73**
	(0.18, 0.76)	(0.12, 0.74)	(0.35, 0.82)	(0.47, 0.86)
WikiArt	0.62	0.66	**0.72**	0.71
	(0.32, 0.80)	(0.37, 0.82)	(0.40, 0.89)	(0.43, 0.84)
Oxford Pets	0.40	0.49	0.43	**0.54**
	(-0.01, 0.64)	(0.24, 0.69)	(-0.06, 0.72)	(0.25, 0.77)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

(b) Metric Performance Assessed by Regret

Dataset	Renggli	LogME	N. Params	ImageNet Acc.
VG Faucet	0.23	**0.00**	1.13	1.13
VG Tree	**0.03**	3.19	1.56	1.56
VG Watch	4.93	**0.00**	0.46	0.46
RVL-CDIP	**0.12**	4.28	**0.12**	**0.12**
WikiArt	**0.00**	**0.00**	1.62	1.62
Oxford Pets	**0.49**	**0.49**	**0.49**	**0.49**

Largest value for each row presented in bold.

Table 5: Mean Metric Performance When Metrics Computed on Whole Training Dataset

Metric	Renggli	LogME	N. Params	ImageNet Acc.
Kendall's Tau	0.35	0.49	**0.60**	0.58
	(0.27, 0.43)	(0.44, 0.54)	(0.56, 0.64)	(0.54, 0.62)
Regret	**1.60**	2.16	1.79	1.79
	(0.82, 2.37)	(1.38, 2.94)	(1.10, 2.47)	(1.10, 2.47)

Values presented with 95% confidence intervals.
Largest value for each row presented in bold.

7.2 Combining Metrics

Within the literature, there are several methods for combining different metrics to improve upon performance. In particular, the Renggli paper [11] finds task-agnostic and task-dependent metrics perform well in different circumstances, and so recommends selecting two models – the best model according to a task-dependent metric (what we call the Renggli metric) and the best model according to a task-agnostic metric (number of parameters). Alternatively, the PARC paper [12] linearly combines a task-agnostic metric with a task-dependent metric to produce a single, ensemble score.

To assess the performance of each possible combination of the four metrics we have considered in our experiments, we take the PARC approach of linearly combining them. We first scale each metric value using a

Z-score normalisation as the metrics produce values on large-scale differences.

Results averaged across all datasets (in the same manner as Section 7.1.2) are shown in Table 6. In terms of Kendall's Tau correlation, there is no evidence to suggest combinations of metrics out-performs using an individual metric, with number of parameters retaining the highest mean performance overall (although not significantly higher than the other metrics).

Alternatively, when considering regret, some combinations do appear to outperform individual metrics, particularly combinations including LogME and a task-agnostic metric, and with LogME combined with ImageNet validation accuracy best overall. This is despite LogME being the worst metric to use on its own in terms of regret. Although these results are also not significant considering the size of the confidence intervals, they hint that there may be circumstances where incorporating a task-dependent metric in the model selection process is worthwhile.

Appendix D.3 includes tables of results for metric combinations on individual datasets.

Table 6: Performance of metric combinations by mean Kendall's Tau correlation and mean Regret across all datasets and data subset sizes, assuming the metrics are computed on the whole training dataset.

Metric Combos	**Kendall's Tau**	**Regret**
Renggli	0.35 (0.27, 0.43)	1.6 (0.82, 2.37)
LogME	0.49 (0.44, 0.54))	2.16 (1.38, 2.94)
N. Params	**0.60** (0.56, 0.64)	1.79 (1.1, 2.47)
ImageNet Acc.	0.58 (0.54, 0.62)	1.79 (1.1, 2.47)
Renggli& LogME	0.45 (0.39, 0.51)	1.5 (0.72, 2.27)
Renggli & N. Params	0.46 (0.39, 0.53)	1.64 (0.96, 2.33)
Renggli & ImageNet Acc.	0.48 (0.43, 0.54)	1.26 (0.59, 1.92)
LogME & N. Params	0.53 (0.48, 0.58)	1.49 (0.93, 2.04)
LogME & ImageNet Acc.	0.55 (0.51, 0.6)	**0.83** (0.46, 1.2)
N. Params & ImageNet Acc.	0.55 (0.51, 0.59)	1.79 (1.1, 2.47)
Renggli & LogME & N. Params	0.48 (0.42, 0.54)	1.15 (0.56, 1.73)
Renggli & LogME & ImageNet Acc.	0.52 (0.47, 0.57)	1.01 (0.47, 1.54)
Renggli & N. Params & ImageNet Acc.	0.52 (0.47, 0.58)	1.82 (1.1, 2.53)
LogME & N. Params & ImageNetAcc.	0.56 (0.52, 0.6)	1.37 (0.88, 1.86)
Renggli & LogME & N. Params & ImageNet Acc.	0.52 (0.47, 0.58)	1.1 (0.57, 1.63)

7.3 Impact of Dataset Size on Performance

7.3.1 Training Dataset Size

Here we explore whether the available training dataset size impacts the performance of the metrics (if still computing the metrics on the whole training dataset), and in particular whether they are still predictive if only a smaller quantity of labelled data is available.

Figure 4 (top) shows the rank correlation for each metric at each training dataset size. Confidence intervals are not shown but are similarly large to the ranges shown in Table 4a.

Similar to previous findings, it appears Renggli is the least stable metric and the most likely to benefit from larger training dataset sizes. For the other metrics, it's not clear whether the quantity of training data matters for correlation – for some datasets and metric pairs (such as LogME on WikiArt) performance appears to improve with dataset size, whereas for others (such as no. of parameters and Oxford Pets) there doesn't appear to be any advantage for larger datasets.

Alongside the correlation, Figure 4 (bottom) plots the regret score against the dataset size for each metric. From the regret, there are signs that the metrics are more likely to select the best model when a larger training dataset is available, particularly for the task-agnostic metrics, though LogME and Renggli may also be more stable overall for larger datasets, with LogME, in particular, having a large regret value on the 500-sample Oxford Pets subset.

Overall, the results suggest the metrics may be better at identifying the best model when larger training datasets (more than 1000 images) are available, though the results are not conclusive (particularly in terms of the achieved correlation values) and a wider set of experiments would be needed to make claims about the overall relationship with dataset size.

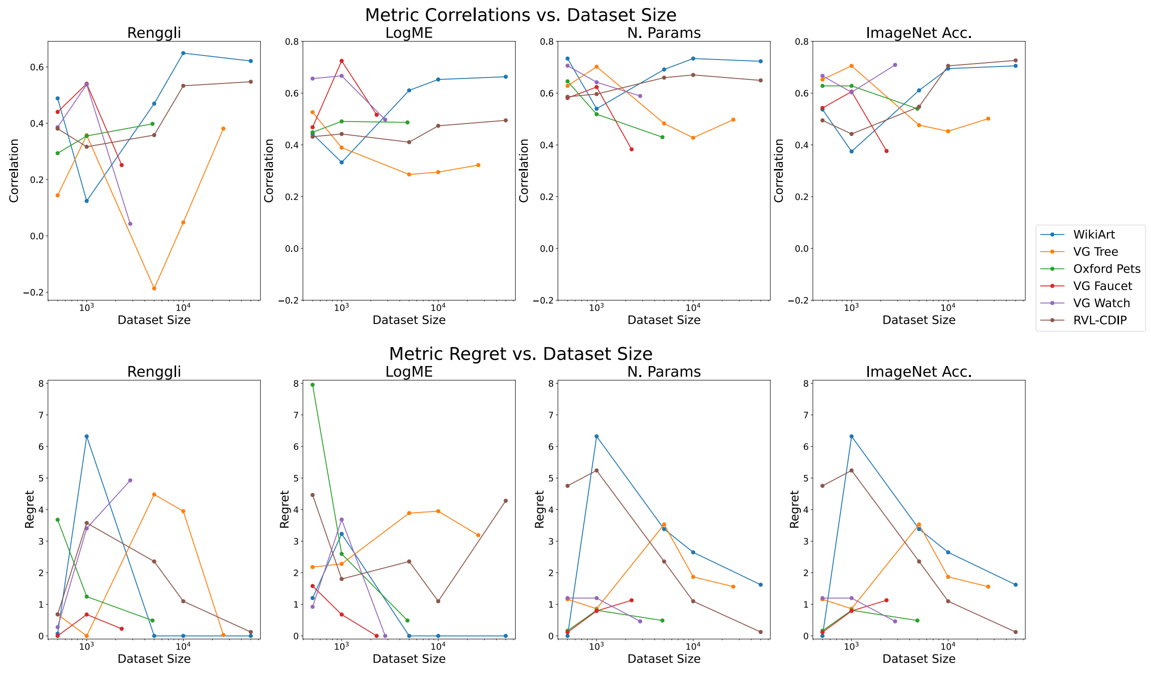


Figure 4: Correlation (top row) and regret scores (bottom row) between metric scores and fine-tuning accuracy against a varying dataset size. Here the metric is computed using the whole training dataset at each size. Note the log scale on the x-axis.

7.3.2 Metrics Dataset Size

All the results discussed previously have been under the assumption that the metrics are computed using the whole training dataset. As the number of images used to compute the task-dependent metrics dominates their computation time (as discussed in the next section), there is a clear benefit if the metrics can be computed on a small subset of the training data and still be performant. The preliminary experiments (see Section 5 and Appendix C, Figures 7 and 9) showed that a relatively large metrics dataset size may be needed before the metric values stabilise, but here we explore whether that has an impact on their ability to rank the models.

Table 7 shows the performance results for individual and combinations of metrics, but fixing the metrics dataset size at its smallest value of 500 samples at all training dataset sizes. This can be compared with Table 6), which shows the same results but computed with the metrics using the whole training dataset. There is no evidence that using a smaller metrics dataset reduces the reported correlation for any of the individual metrics or metrics combinations. Alternatively, in terms of mean regret, there is a reduction of around 1 percentage point in the achieved accuracy of the selected model via the Renggli and LogME metrics (the first two rows of the table). The results for number of parameters and ImageNet accuracy (third and fourth rows) don't change as they don't depend on the metrics dataset.

Interestingly, using combinations of metrics (row five onwards) appears to mitigate the performance loss from using a small metrics dataset, and particularly combinations of LogME with a task-agnostic metric. LogME with 500 samples combined with the number of model parameters achieves a mean regret less than 0.1 percentage points higher than the best achieved when using a larger metrics dataset (an insignificant difference considering the confidence intervals), for example. Overall, we would therefore recommend either using only a zero-cost metric, due to their simplicity, or a combination of a zero-cost metric and LogME with a small metrics dataset, which may out-perform the single-metric approach whilst remaining low-cost (as discussed in the next section).

Figure 5 shows the correlation and regret values plotted at all metrics dataset sizes, whilst leaving the training datasets fixed at their maximum size for each target dataset. Similarly to above, for regret there is some evidence that performance improves at metrics dataset sizes above 1000 samples. However, this is not consistent across all datasets and it is much less clear whether there is an improvement in correlation at larger dataset sizes. Further experiments would be needed to fully establish the relationship with metrics dataset size.

7.4 Time Saving Relative to Fine-Tuning

The goal of LoCoMoSeT is to reduce the time taken to select models. The zero-cost, task-agnostic metrics we've considered (no. parameters and ImageNet validation accuracy) are the best choice in this regard as they have zero computational cost (or at least much lower cost than task-dependent metrics). Nevertheless, if deciding to use one of the task-dependent metrics (LogME or Renggli), the question becomes how much time do you save relative to fine-tuning the models?

Table 8 shows the time taken to fine-tune each model and to compute the LoCoMoSeT metrics. The time taken for the metrics is dominated by the number of forward passes (number of images) of the model required. For example, on the RVL-CDIP dataset (and using 50,000 images for the metrics) for the slowest metrics job loading the model took 4 seconds, the Renggli metric took 12 seconds to compute, and the LogME metric 1 second, but extracting the features from the model (needed as a precursor to computing LogME and Renggli) took one and a half hours. The recorded metrics times in the table are the total time to extract the features once and then to compute LogME and Renggli. Training times are for the fine-tuning strategy described in Section 6.3.

If computing the metrics using the whole training set, they are roughly 15-20 times faster to calculate than fine-tuning job a model (not accounting for an unexplained exception for WikiArt where this is reduced to below 10 times). This is equivalent to roughly half an epoch of fine-tuning. However, as the time taken is dominated by the number of forward passes of the models, the time advantage of using the task-dependent metrics will be much greater if using only a subset of the training data to compute them, meaning the previous result represents the worst-case scenario. The results in Section 7.3.2 suggest using LogME at a metrics dataset size of 500 images combined with a zero-cost metric may be sufficient to achieve similar, or even better, performance than computing LogME alone on the whole training dataset. Using LogME with 500 images would be roughly 150-200 times faster than fine-tuning a model on a 5000-image training dataset (or approximately equivalent to 5% of an epoch of fine-tuning), for example.

Metric	Correlation	Regret
Renggli	0.27, (0.19, 0.35)	2.49, (1.38, 3.61)
LogME	0.45, (0.41, 0.49)	3.19, (2.3, 4.09)
N. Params	**0.6**, (0.56, 0.64)	1.79, (1.1, 2.47)
ImageNet Acc.	0.58, (0.54, 0.62)	1.79, (1.1, 2.47)
Renggli & LogME	0.42, (0.37, 0.47)	1.98, (1.18, 2.77)
Renggli & N. Params	0.44, (0.38, 0.49)	1.66, (0.88, 2.45)
Renggli & ImageNet Acc.	0.46, (0.4, 0.52)	1.63, (0.8, 2.47)
LogME & N. Params	0.55, (0.51, 0.58)	**0.91**, (0.57, 1.25)
LogME & ImageNet Acc.	0.56, (0.53, 0.6)	1.04, (0.69, 1.4)
N. Params & ImageNet Acc.	0.55, (0.51, 0.59)	1.79, (1.1, 2.47)
Renggli & LogME & N. Params	0.47, (0.43, 0.52)	1.89, (1.03, 2.76)
Renggli & LogME & ImageNet Acc.	0.51, (0.47, 0.56)	1.15, (0.64, 1.65)
Renggli & N. Params & ImageNet Acc.	0.5, (0.46, 0.55)	1.62, (0.86, 2.39)
LogME & N. Params & ImageNet Acc.	0.56, (0.52, 0.6)	1.79, (1.1, 2.47)
Renggli & LogME & N. Params & ImageNet Acc.	0.53, (0.49, 0.58)	1.25, (0.48, 2.03)

Table 7: 500 metric samples, averaged over all other consideration

8 Limitations

As laid out in sections 4 and 6 there were a series of design choices made for this project for reasons including focusing the scope and computational constraints. Here we explore the limitations to the project that occurred due to these choices, how they potentially affected the outcomes, and if applicable what could be done to mitigate these effects.

We focused on image classification tasks and our findings may not translate to other domains. Whilst the focus on image classification affected the metrics considered in the project, it should be noted that the task-dependent metrics (Renggli and LogME) are not strictly limited to the visual domain in their construction. Both can be applied to classification tasks in other domains - with LogME already applied to NLP classification tasks in the literature. LogME also has an adaptation for both multi-label classification and regression tasks, both of which the Renggli metric could be adapted for. Metrics for selecting pre-trained models for tasks such as object detection, image manipulation, and NLP tasks such as translation, text generation and captioning would require alternative constructions.

The choice of models included in the experiments imposes several restrictions that limit the generalisabiltiy of our findings. We consider only models based on the transformer architecture, as they are the current dominant architecture for the tasks that we considered. The metrics we consider were all initially tested on CNNs, but are architecture-agnostic in their construction. However, other architectures remain untested and we do not compare models based on different architectures. We further select only models pre-trained on ImageNet,

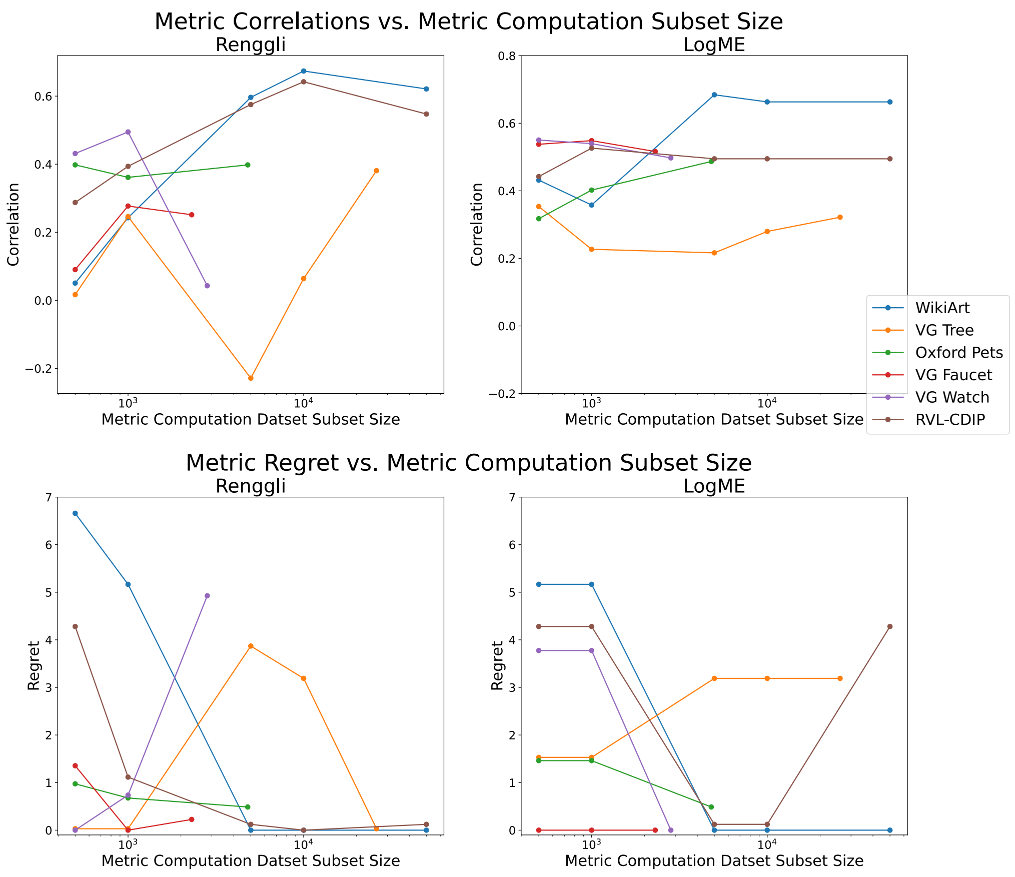


Figure 5: Correlation (top row) and regret (bottom row) of the task-dependent metrics with post-fine-tuning test accuracy versus the number of images in the metrics dataset. The training dataset is held at its maximum size for each dataset. Note the log scale on the x-axis. Here the task-agnostic metrics are dropped as they don't require a metrics dataset.

which means that the similarity between the pre-training dataset and the target dataset is held constant across all models. Holding both dataset similarity and large-scale architectural differences between models constant in our experiments may inflate the performance of task-agnostic metrics (number of parameters and ImageNet validation accuracy) relative to scenarios where these variables are not held constant.

As discussed in Section 6.3, we used a default fine-tuning strategy without hyperparameter tuning (with one exception for the learning rate used with the DINOv2 models), including training all the parameters in each model rather than leaving any of them frozen, for example. This was necessary to limit the computational time and costs of the project but risks our achieved fine-tuned accuracies being sub-optimal for each model. Scanning over optimiser hyperparameters, freezing/unfreezing different parts of the model, exploring data augmentation strategies, and similar could achieve different levels of improvement for each model and so change the relationship between the ranking predicted by the LoCoMoSeT metrics and the actual model rankings.

The goal of the application of these metrics is to select the best, or most appropriate, pre-trained model for the target dataset. For the purposes of this project, we consider the best model as the one with the highest post-fine-tuning test accuracy. However, using the test accuracy alone omits other potential constraints, such as computational constraints which larger models may not fulfil. Approaches such as [51] use a Pareto curve approach [52] to develop model architectures that solve this multi-dimensional optimisation problem for novel architecture search.

Finally, there is an emerging paradigm of using large language models, among other foundation models, without fine-tuning for many tasks [53]. This may require a different model selection process.

Dataset	No. Train Images	Train Time	No. Metric Images	Metric Time	Metric Adv.	Max Met. Adv.
VG Faucet	500	2 - 19	500	0 - 1	15	62
	1000	4 - 38	1000	0 - 1	18	38
	2303	7 - 84	2303	1 - 3	17	17
VG Tree	500	2 - 19	500	0 - 1	15	735
	1000	4 - 38	1000	0 - 2	17	399
	5000	16 - 189	5000	1 - 7	19	93
	10000	35 - 385	10000	3 - 14	19	47
	26004	80 - 948	26004	8 - 35	18	18
VG Watch	500	2 - 19	500	0 - 1	16	82
	1000	3 - 38	1000	0 - 1	18	45
	2825	9 - 103	2825	1 - 4	18	18
RVL-CDIP	500	2 - 18	500	0 - 1	13	1226
	1000	4 - 38	1000	0 - 1	14	689
	5000	17 - 189	5000	2 - 7	15	156
	10000	36 - 383	10000	4 - 14	15	79
	50000	187 - 1923	50000	19 - 91	15	15
WikiArt	500	2 - 19	500	0 - 1	7	680
	1000	4 - 38	1000	1 - 2	7	359
	5000	19 - 192	5000	4 - 9	8	77
	10000	41 - 381	10000	8 - 20	8	39
	50000	198 - 1884	50000	38 - 104	8	8
Oxford Pets	500	2 - 18	500	0 - 1	17	140
	1000	3 - 38	1000	0 - 1	18	86
	4803	15 - 169	4803	1 - 7	20	20

Table 8: Computation times for both training and metrics. All times in minutes. A time of 0 means the computation took less than half a minute. The ranges shown in the Train Time and Metric Time columns are the elapsed time for the fastest and slowest models to fine-tune or compute the metrics (across the 20 candidate models included in our experiments). The metric advantage is how much faster the metrics are to compute than full fine-tuning as a multiple of the time taken. The max metric advantage is the difference in computation time between the metrics with the stated number of metric images per row and the maximum number of training images for fine-tuning each dataset.

9 Conclusions

In this project, we have explored several metrics for predicting the performance of pre-trained vision transformers after fine-tuning on a target image classification dataset. We explored the existing literature and found that some existing metrics have been shown to have poor performance in different contexts, to the extent of being anti-correlated with fine-tuned accuracy in some cases, or have been tested only on convolutional neural networks and not vision transformers (the current dominant architecture for classification problems).

We therefore ran experiments to fine-tune a pool of twenty vision transformers on six different downstream classification tasks. Out of the six metrics we initially considered, two did not pass our sanity checks of whether the metric is predictive of a model's performance on its pre-training dataset (i.e. the metric could not predict the ImageNet-1k accuracy of a model pre-trained on ImageNet-1k). This and the prior literature review highlight the difficulty both in conceiving appropriate metrics and in evaluating their performance.

We finally selected two task-dependent metrics (Renggli and LogME), which predict fine-tuned performance using feature representations in the penultimate layer of the model, and two task-agnostic metrics (number of parameters and reported ImageNet-1k validation accuracy), which are essentially zero cost to compute but do not take into account anything about the target dataset/task.

Overall, in our experiments, all four metrics have significant positive correlations with fine-tuned model accuracy. The task-dependent metrics require roughly the time of half an epoch of fine-tuning to run if they are computed on the whole training dataset, but can be computed on much smaller subsets of images to reduce the cost further. However, we found no evidence that the task-dependent metrics (LogME, Renggli) out-perform the zero-cost metrics (no. parameters, ImageNet accuracy) overall. As the task-dependent metrics are more expensive to compute, if using only one metric we would recommend using either no. of parameters or ImageNet validation accuracy as the proxy for likely fine-tuned model performance. However, we found some evidence to suggest that a strategy combining LogME with a zero-cost metric may outperform using a zero-cost metric alone. This strategy was also effective when computing LogME with only a small subset of the training data.

All results should be interpreted with caution about their generalisability as they relate to a single choice of candidate models and a small number of target datasets. In particular, we do not vary the pre-training dataset in our experiments, only considering models pre-trained on ImageNet-1k. We also only consider image classification, and while the metrics could be applied to other tasks (and have been in the case of the LogME metric) they may perform differently.

10 Recommendations for Future Work

The most notable simplification we have made for the experiments run in this project is the decision to use models pre-trained on a single dataset (ImageNet-1k) and for a single downstream task type (image classification). A natural extension would be to explore the role of similarity between the pre-training and target datasets in determining performance after fine-tuning, and to what extent the LoCoMoSeT metrics capture this effect. This could draw on previous ARC work on model and dataset similarities [54], and particularly the use of the optimal transport data similarity metric which is shown to correlate with transferability in the literature [5]. This would also be highly relevant for generalisability to other tasks, which may be more likely to have diversity in pre-training datasets compared to the prevalence of ImageNet for image classification.

More generally, an area for future work could be exploring the best strategies for transfer learning with vision transformers. There is some previous literature on the best approaches for training ViTs (such as [40, 49]), which discuss architecture design and pre-training approaches but also include some experiments for fine-tuning pre-trained ViTs on new datasets. This project has focused on reducing the time consumed by the initial model selection process; exploring fine-tuning strategies would aid in reducing the cost in the next step of training the selected model itself. Other training regimes, such as self-supervised training, parameter-efficient fine-tuning, and knowledge distillation, could also be explored. In the LoCoMoSeT context, it would be interesting to explore whether there are metrics that could be used to recommend how a model should be fine-tuned. For example, could computing LogME or Renggli scores on feature representations in earlier blocks of the model give insights about which parameters should be frozen or unfrozen during fine-tuning?

There are also further metrics that could be explored (such as the H_α metric), different approaches for combining metrics, and different ways of using the metrics (such as computing them after a small amount of fine-tuning on the target dataset, as in [55]).

Acknowledgements

The computations described in this research were performed using the Baskerville Tier 2 HPC service [50]. Baskerville was funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

References

[1] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*, 2023.

[2] Krishna Teja Chitty-Venkata, Murali Emani, Venkatram Vishwanath, and Arun K Somani. Neural architecture search for transformers: A survey. *IEEE Access*, 10:108374–108412, 2022.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. arXiv:2010.11929 [cs].

[4] Hugging Face. Model hub. `https://huggingface.co/models`, 2023. Accessed: 2024-01-31.

[5] David Alvarez-Melis and Nicolo Fusi. Geometric Dataset Distances via Optimal Transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 21428–21439. Curran Associates, Inc., 2020.

[6] Robin Hirt, Akash Srivastava, Carlos Berg, and Niklas Kühl. Sequential transfer machine learning in networks: Measuring the impact of data and neural net similarity on transferability. *arXiv preprint arXiv:2003.13070*, 2020.

[7] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1395–1405, 2019.

[8] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[9] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[10] Qinqin Zhou, Kekai Sheng, Xiawu Zheng, Ke Li, Xing Sun, Yonghong Tian, Jie Chen, and Rongrong Ji. Training-free transformer architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10894–10903, 2022.

[11] Cedric Renggli, André Susano Pinto, Luka Rimanic, Joan Puigcerver, Carlos Riquelme, Ce Zhang, and Mario Lučić. Which model to transfer? finding the needle in the growing haystack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9205–9214, 2022.

[12] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. Scalable diverse model selection for accessible transfer learning. *Advances in Neural Information Processing Systems*, 34:19301–19312, 2021.

[13] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers, January 2022. arXiv:2109.10686 [cs].

[14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. arXiv:2001.08361 [cs, stat].

[15] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling Vision Transformers to 22 Billion Parameters, February 2023. arXiv:2302.05442 [cs].

[16] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

[17] Amiel Meiseles and Lior Rokach. Source model selection for deep learning in the time series domain. *IEEE Access*, 8:6190–6200, 2020.

[18] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12387–12396, 2019.

[19] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[20] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists. *Frontiers in artificial intelligence*, 4:108, 2021.

[21] Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In *International Conference on Machine Learning*, pages 5351–5360. PMLR, 2019.

[22] Weizhi Li, Gautam Dasarathy, Karthikeyan Natesan Ramamurthy, and Visar Berisha. Finding the homology of decision boundaries with active learning. *Advances in Neural Information Processing Systems*, 33:8355–8365, 2020.

[23] Seungho Choe and Sheela Ramanna. Cubical homology-based machine learning: An application in image classification. *Axioms*, 11(3):112, 2022.

[24] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. LogME: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*, pages 12133–12143. PMLR, 2021.

[25] Kaichao You, Yong Liu, Ziyang Zhang, Jianmin Wang, Michael I Jordan, and Mingsheng Long. Ranking and tuning pre-trained models: a new paradigm for exploiting model hubs. *The Journal of Machine Learning Research*, 23(1):9400–9446, 2022.

[26] Kaicho You. Logme. `https://github.com/thuml/LogME/tree/main`, 2021.

[27] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*, pages 7294–7305. PMLR, 2020.

[28] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *2019 IEEE international conference on image processing (ICIP)*, pages 2309–2313. IEEE, 2019.

[29] Shibal Ibrahim, Natalia Ponomareva, and Rahul Mazumder. Newer is not always better: Rethinking transferability metrics, their peculiarities, stability and performance. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 693–709. Springer, 2022.

[30] Felix D Schönbrodt and Marco Perugini. At what sample size do correlations stabilize? *Journal of Research in Personality*, 47(5):609–612, 2013.

[31] Mohamad Adam Bujang and Nurakmal Baharum. Sample size guideline for correlation analysis. *World*, 3(1):37–46, 2016.

[32] Sorana-Daniela Bolboaca and Lorentz Jäntschi. Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences*, 5(9):179–200, 2006.

[33] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[34] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

[35] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015.

[36] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2017.

[37] Sachin Mehta and Mohammad Rastegari. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer, March 2022. arXiv:2110.02178 [cs].

[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, January 2021. arXiv:2012.12877 [cs].

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, April 2023. arXiv:2304.07193 [cs].

[40] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers, June 2022. arXiv:2106.10270 [cs].

[41] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers, September 2022. arXiv:2106.08254 [cs].

[42] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing Convolutions to Vision Transformers, March 2021. arXiv:2103.15808 [cs].

[43] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, August 2021. arXiv:2103.14030 [cs].

[44] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer V2: Scaling Up Capacity and Resolution, April 2022. arXiv:2111.09883 [cs].

[45] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions, August 2021. arXiv:2102.12122 [cs].

[46] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers, May 2021. arXiv:2104.14294 [cs].

[47] HuggingFace trainer. `https://huggingface.co/docs/transformers/main_classes/trainer`.

[48] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. arXiv:1711.05101 [cs, math].

[49] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about Vision Transformers, March 2022. arXiv:2203.09795 [cs].

[50] Baskerville Tier 2 HPC. `https://www.baskerville.ac.uk`.

[51] Yong Guo, Yaofo Chen, Yin Zheng, Qi Chen, Peilin Zhao, Junzhou Huang, Jian Chen, and Mingkui Tan. Pareto-aware neural architecture generation for diverse computational budgets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2247–2257, 2023.

[52] Thomas Blanchet, Juliette Fournier, and Thomas Piketty. Generalized pareto curves: theory and applications. *Review of Income and Wealth*, 68(1):263–288, 2022.

[53] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[54] Phil Swatton, Jo Knight, and James Bishop. Model similarity phase 2: Dataset similarity. *Applied Research Centre for Defence and Security*, 2023.

[55] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

[56] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[57] Intel/vit-base-patch16-224-int8-static-inc on Hugging Face. `https://huggingface.co/Intel/vit-base-patch16-224-int8-static-inc`, December 2023.

[58] Ali Hassani and Humphrey Shi. Dilated Neighborhood Attention Transformer, January 2023. arXiv:2209.15001 [cs].

[59] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6185–6194, 2023.

[60] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021.

[61] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. arXiv:1802.03426 [cs, stat].

A Reported Metric Performance

Here, we collate the reported performance of the metrics in Sections 4.2 and 4.3, based on the performance metrics in subsection 4.5. Table 9 presents the reported metric performances from the literature:

Metric Name	Performance	Performance metric	Ref
Renggli	0.8 – 0	relative regret	[11]
PARC	0.53 – 0.70	mean Pearson's correlation	[12]
TDA	0.06 – 0.1	absolute difference in accuracy	[21]
LogME	0.5 – 0.82	Kendall's tau correlation	[24]
NCE	-0.35 – 0.68	Kendall's tau correlation	[7, 24]
LEEP	-0.06 – 0.72	Kendall's tau correlation	[27, 24]
H_α-score	0.67 – 0.89	Pearson's correlation	[29]

Table 9: Collected metric performance scores from the literature. The NCE and LEEP scores are from the LogME paper [24]. Regret as a measure (Renggli) is optimal if at 0. See below for details.

Renggli The Renggli metric (which we are using to refer to the task-aware linear classifier evaluation metric from [11]) has strong experimental results from the original paper, evaluated on several pools of CNN models with regret used to evaluate the efficacy of the metric. The values given in table 9 are the best and worst regret for the datasets considered in the paper, with average regret being ~ 0.1 (a value of 0 regret is the best possible). They also report an improvement in performance if a task-dependent and task-agnostic metric are combined.

PARC The PARC metric [12] performs strongly on a range of CNNs and target datasets in the paper, with a measure of mean Pearson's correlation averaging the Pearson's correlation between the PARC metric scores and post fine-tuning validation accuracies for each model across all the datasets considered. The range of values in table 9 encompasses PARC without any additional methods (0.53), PARC with feature reduction (0.59), and PARC with both feature reduction and scaling by the capacity to learn proxy for each model (0.70). The values suggest that it outperforms all other metrics it is compared to in the paper.

TDA The topological data analysis methods of [21] examine a slightly different problem to the general case considered in this project. They take a dataset (e.g MNIST or CIFAR-10) and construct binary classification datasets based on each pair of classes, training a classifier on each binary dataset. Then the topological complexity is compared for each of these classifiers to each other binary dataset, using this as a measure to pick the best pre-trained classifier for the target binary dataset. They evaluate the approach via the difference in accuracy between models with similar complexity to the dataset, and models with dissimilar complexity to the dataset.

LogME The LogME metric [24, 25] has experimental results in several settings. Table 9 reports the range of Kendall's tau correlation between the LogME metric values and fine-tuned accuracy for the pools of CNN models considered in the paper for the image classification task, across all the datasets considered. Kendall's tau correlation is considered to be strong for values above 0.5 [32], suggesting LogME to be an effective metric. In the regression setting LogME has a reported correlation of 0.79 with the MSE of the fine-tuned models. LogME has also been tested on vision transformers in a limited set of experiments (similar to the exploratory checks we used as part of our metric selection process in Appendix C).

Alternative Metrics NCE [7] and LEEP [27] are both frequently compared to other metrics in literature as benchmarks. The correlation values in table 9 are taken from the LogME paper [24], which reports NCE and LEEP underperforming in comparison to LogME on the tasks considered. Additionally, the PARC paper [12] reports a mean Pearson's correlation of 0.021 and 0.1, for NCE and LEEP respectively, without the capacity to change modification, and 0.28 and 0.26 with it.

The original H-score [28] similarly performs poorly, however the adapted form H_α-score [29] suggests a strong performance, with the Pearson's correlation scores for the image classification tasks considered in the paper given in table 9. In this paper they also consider small balanced and unbalanced datasets, for which H_α-score outperforms NCE, LEEP and LogME.

B Excluded Models

While curating the model pool used for our experiments we considered several other model families which were then dropped as they did not perform as expected or would have required customised code for fine-tuning or inference. Examples include:

- **Models using other frameworks:** A model being in the HuggingFace model hub does not guarantee that it is designed to be used with HuggingFace transformers. The most notable example is models from timm (torch image models) [56].
- **Models with 3rd party dependencies:** Some models in the hub (that are in the HuggingFace transformers format) have other third-party dependencies. Examples include a quantized Intel model [57], that required the use of a separate quantization library, and a family of models (DiNAT [58]) requiring a library implementing neighbourhood attention [59]. Where we attempted to use models in this category we often found the dependencies were difficult to install or the resulting model performed unexpectedly.
- **(Some) distilled models:** Distilled models are trained with two loss heads, one to compute losses compared to ground truth labels and the other with respect to a teacher model. Some distilled models in the HuggingFace hub would require modifications to be fine-tuned conventionally with a single loss head, either to the model loading code or the model base classes. One example was the LeViT family of models [60]. Other distilled models are ready out of the box for conventional fine-tuning with transformers (such as the DeiT family listed above).

These issues are not insurmountable in most cases, but their inclusion would have come at greater cost and were therefore considered out of scope. Using models sourced only from the HuggingFace hub is not sufficient to guarantee they will work out of the box within the HuggingFace ecosystem of libraries.

C Initial Metric Exploration

Here we give the results of the exploratory checks run on the various metrics, in which we computed the metrics on the ImageNet validation set for each model, and then compared those metric scores to the reported ImageNet test accuracy of each model.

C.1 Number of Model Parameters

Estimating the post-fine-tuning performance of these models by the number of parameters within the model presented an efficient and well-correlated metric within our initial exploratory checks. This can be seen by Figure 6, which plots the number of parameters in the model against the reported performance on Imagenet-1k.

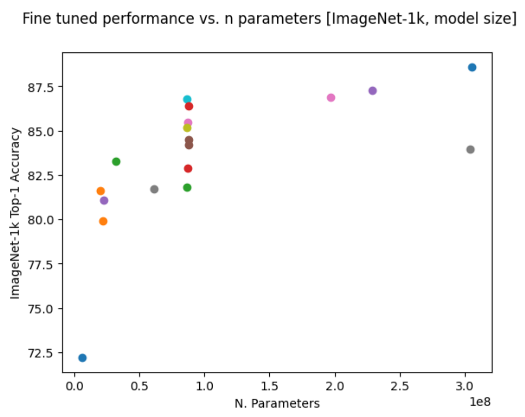


Figure 6: Number of model parameters compared to the reported performance of each model on Imagenet-1k. Here each colour represents a different model.

As a zero-cost, task-agnostic metric, the number of parameters serves as a useful baseline for low-cost task-dependent metrics. As seen in Figure 6, over the entire pool of models, it has a strong association with reported ImageNet-1k test accuracy. In particular, it accurately predicts the best and worst model.

However, notably, it does not predict the position of the second-largest model correctly. Moreover, there is a vertical line of models with similar sizes but a large variation in validation accuracy, showing there can be a large level of ambiguity for models of similar size.

C.2 Renggli

It should be noted that the Renggli metric should be expected to return strong results under these exploratory checks. This is due to the fact the final combination of model and classifier head should be a close approximation of the original model, having been trained and then evaluated on the same dataset.

We present the results for the Renggli metric in Figure 7.

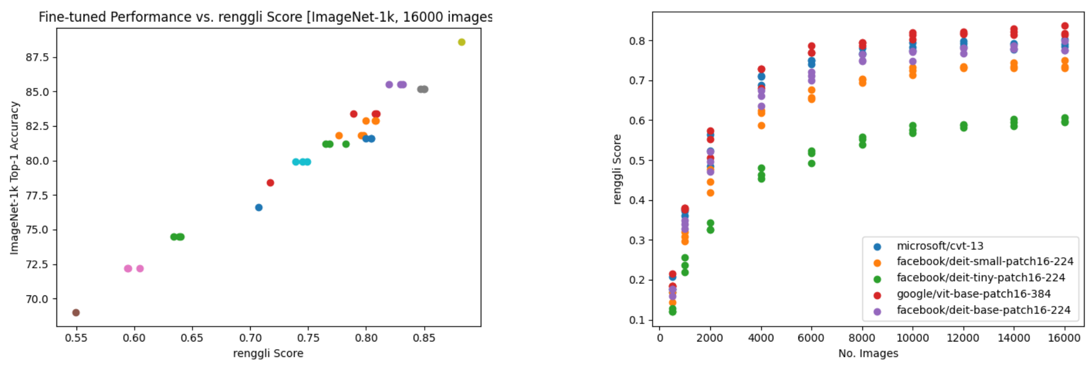


Figure 7: The leftmost plot shows Renggli metric scores compared to the reported performance of each model on Imagenet-1k. The rightmost plot shows how the Renggli score changes as the number of images used to compute it increases. In both plots, colours represent different models.

The Renggli metric returns a ranking of models with ImageNet-1k as the downstream task that is strongly associated with the ranking of models by their reported validation accuracy, as seen in Figure 7 (left).

Additionally, as can be seen by Figure 7 (right), the Renggli metric score converges to a stable score when computed on roughly 10,000 images. We expect that this value is likely to be dependent on the number of classes and their representation within the dataset. For ImageNet-1k, this corresponds to around 10 images per class and, whilst being a large number of images to have to run inference on, it is much smaller than the

50,000 images in the validation set or 1.3 million images in the training set. It also may not be necessary for the absolute value of the metric to have converged to be able to obtain a reliable ranking (this is explored in the full experiments in Section 7).

These positive results led us to decide to include the Renggli metric in our full experiments.

C.3 PARC

Despite the strong performance reported in the literature, the PARC metric did not return a well-correlated ranking of the models with their performance on Imagenet-1K, as seen by Figure 8.

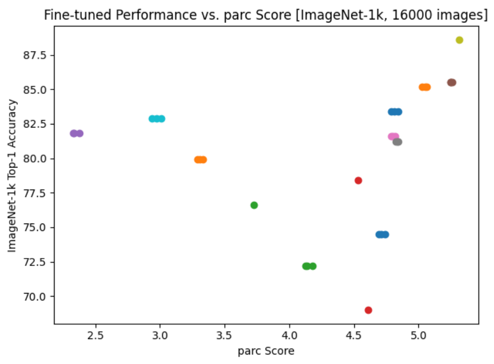


Figure 8: PARC metric scores compared to the reported performance of each model on Imagenet-1k. Here each colour represents a different model.

The results of the figure are typical across variations of PARC that we tried: varying the size of the metrics dataset, and varying the dimension of feature reduction (PCA) (applied to the features in the penultimate layer before being input to PARC). A further potential pitfall of PARC is that the distance matrices (Equation (3)) have to be computed in memory, which presents a large memory constraint if the number of model features or the metrics dataset size are large.

Several possible improvements to PARC could be made that we opted not to try for this project. Specifically, there are many alternatives to the feature reduction method that could be implemented, such as UMAP [61] (although alternatives to PCA were not considered by the original authors), and the inclusion of the 'capacity to learn' adaptation discussed in Section 4.4.

Overall, this early failure led us to decide against including PARC in our full set of experiments.

C.4 Topological Data Analysis

We experimented with an implementation of the TDA-based metric but chose not to continue with it for both a lack of convincing early results and fundamental problems with this usage of TDA. We saw little evidence of correlation between the rankings model returned by TDA and the ImageNet-1k validation accuracies (results can be shared on request).

We consider this use of TDA immature in its development. In the literature, binary classifiers are typically considered, with the generalisation to a greater range of classes not seen. Whilst it may be possible to use the topological complexity of the underlying data and labels that are gained through an untrained classifier head, or with the features instead, this can not be seen in our experiments thus far and it is unclear how one would adapt the current approach to ensure this is accurate. The development of such an approach is beyond the scope of this project.

C.5 LogME

Figure 9 presents the results of our exploratory checks with the LogME metric.

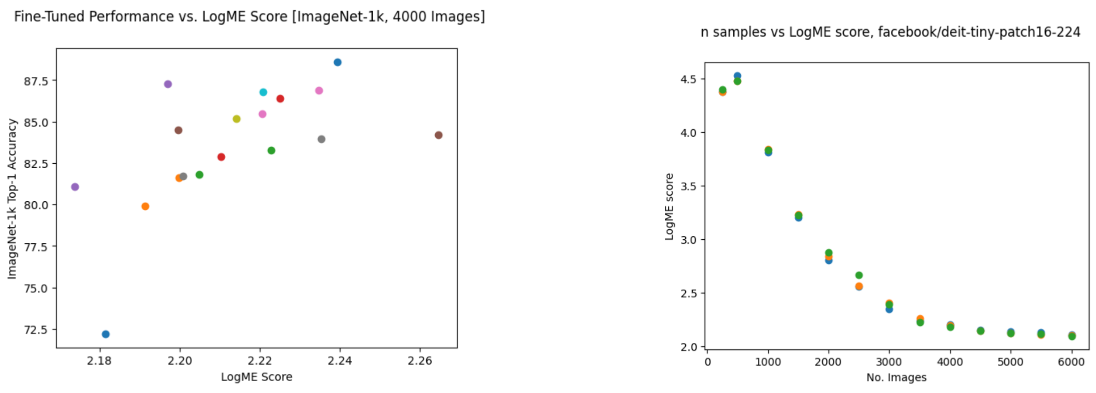


Figure 9: The leftmost plot shows LogME scores compared to the reported performance of each model on Imagenet-1k. Here points are coloured by model. The rightmost plot shows how the LogME score changes as the number of images used to compute it increases.

The LogME metric performed strongly in the sanity checks, with a clear association with the validation accuracy of the pool of models considered, which can be seen in Figure 9 (left). These results, alongside strong theoretical and empirical results from the literature, give LogME a strong backing as a candidate metric. Further, there is already an efficient implementation of LogME available, provided by the authors of the original paper.

In the original LogME paper [24] the entire target dataset is used to compute a metric value, however, as can be seen by Figure 9 (right) the value of the LogME metric converges for a relatively small subset of images taken from the overall dataset. This allows for it to be computed more efficiently as the bottleneck in computation is the inference through the model to obtain the features.

Overall, the evidence presented here was sufficiently positive to lead us to keep LogME for our full set of experiments.

D Additional Results

D.1 Raw Data Examples

Figure 10 shows raw data for the experiments with the best and worst correlation between the metric score and post-fine-tuning accuracy, for each metric. It should be noted that the y-axis here is not scaled uniformly, with uncorrelated results often arising because of uniformity in the test accuracy achieved. The figure also notes the model that the metric would have picked in each instance (orange cross), and the model that performed best under fine-tuning (green cross). In the case of the worst-correlated results, the difference in post-fine-tuning test accuracy of the picked models can be large. However, it should be noted that the best fine-tuned model would not have been selected by the metric in any of the best-correlated results either, which shows the correlation between the two does not give the full picture. Figure 11 gives the same data with the models ranked by the metric scores and post fine-tuning accuracy.

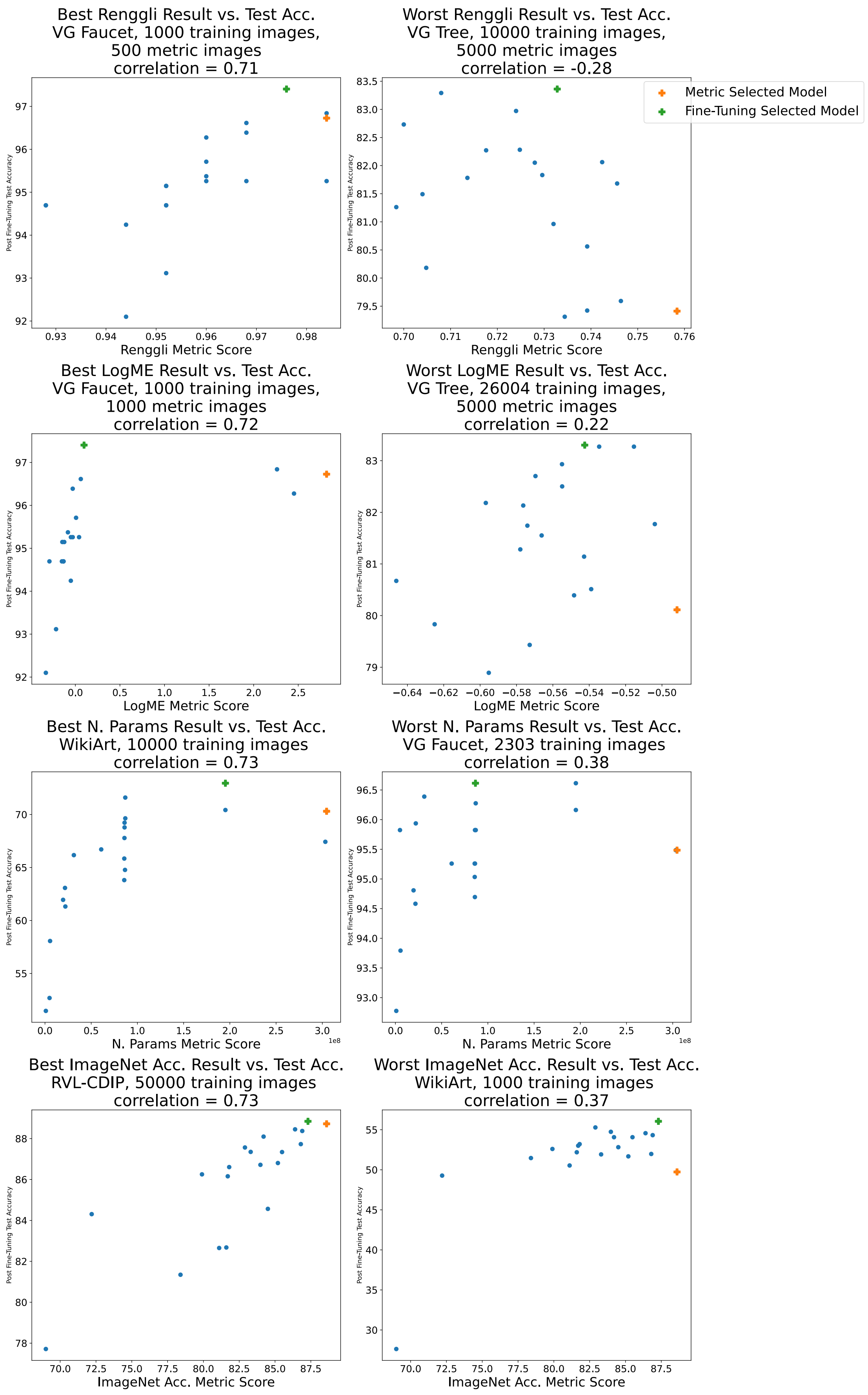


Figure 10: Best and worst performing (by correlation) metric scores against the post-fine-tuning test accuracy for each metric. The model that the metric would have picked in each instance is shown with an orange cross, with the best fine-tuned model shown with a green cross.

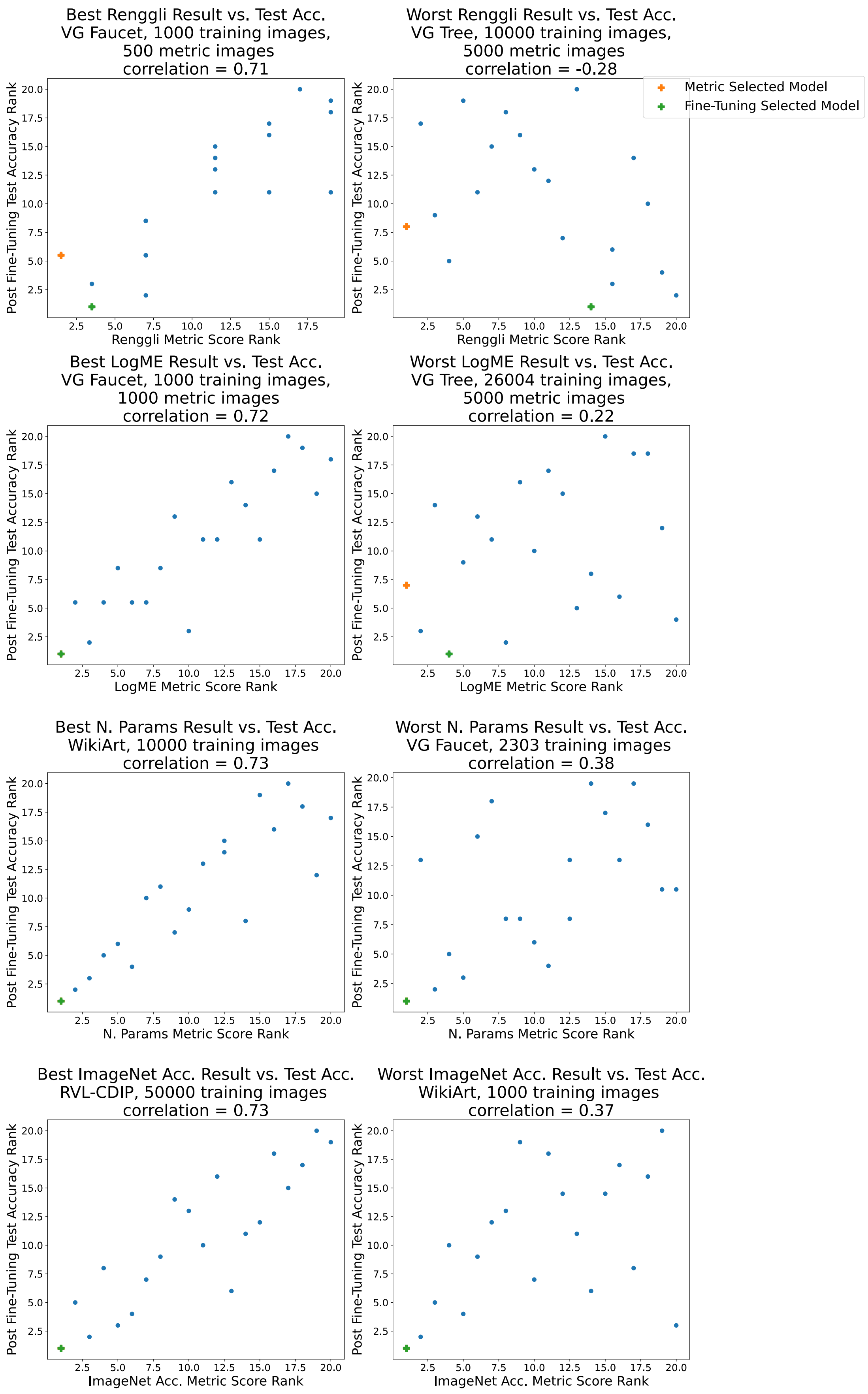


Figure 11: Best and worst performing (by correlation) ranked metric scores against the ranked post-fine-tuning test accuracy for each metric. The model that the metric would have picked in each instance is shown with an orange cross, with the best fine-tuned model shown with a green cross.

D.2 Results at Each Training Set Size

D.2.1 Kendall's Tau

To give a fuller picture of the results here we compare metric performances for varied subsets of the datasets. As in Table 4a, Tables 16 - 21 have the metrics across the columns, with Kendall's tau rank correlation given in the cells alongside a bootstrapped 95% confidence interval, with the strongest correlation in each case in bold. Across the rows the size of the training subset used in each computation is given.

Splitting the results in this way gives a very similar picture to that of Table 4a, with the task-agnostic metrics typically out-performing the task-dependent metrics, in addition to LogME outperforming Renggli.

Table 10: Metrics Performance Across Dataset Sizes: VG Faucet

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.44 (0.00, 0.73)	0.47 (0.13, 0.73)	**0.58** (0.34, 0.80)	0.54 (0.28, 0.76)
1000	0.54 (0.27, 0.75)	**0.72** (0.51, 0.86)	0.62 (0.39, 0.80)	0.61 (0.27, 0.81)
2303	0.25 (-0.19, 0.60)	**0.52** (0.09, 0.79)	0.38 (0.05, 0.66)	0.38 (-0.02, 0.69)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 11: Metrics Performance Across Dataset Sizes: VG Tree

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.14 (-0.20, 0.48)	0.53 (0.22, 0.75)	0.63 (0.41, 0.79)	**0.65** (0.35, 0.83)
1000	0.36 (-0.03, 0.67)	0.39 (0.03, 0.66)	0.70 (0.47, 0.84)	**0.71** (0.42, 0.86)
5000	-0.19 (-0.52, 0.15)	0.29 (-0.15, 0.54)	**0.48** (0.03, 0.73)	0.48 (0.06, 0.72)
10000	0.05 (-0.37, 0.40)	0.29 (-0.17, 0.63)	0.43 (0.04, 0.64)	**0.45** (0.10, 0.66)
26004	0.38 (-0.06, 0.65)	0.32 (-0.11, 0.62)	0.50 (0.12, 0.72)	**0.50** (0.22, 0.72)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 12: Metrics Performance Across Dataset Sizes: VG Watch

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.39	0.66	**0.71**	0.67
	(-0.01, 0.62)	(0.42, 0.82)	(0.42, 0.86)	(0.40, 0.83)
1000	0.54	**0.67**	0.64	0.60
	(0.27, 0.74)	(0.41, 0.82)	(0.24, 0.81)	(0.19, 0.78)
2825	0.04	0.50	0.59	**0.71**
	(-0.41, 0.43)	(0.12, 0.73)	(0.20, 0.79)	(0.45, 0.86)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 13: Metrics Performance Across Dataset Sizes: Oxford Pets

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.29	0.45	**0.65**	0.63
	(-0.07, 0.59)	(0.09, 0.69)	(0.37, 0.83)	(0.32, 0.82)
1000	0.35	0.49	0.52	**0.63**
	(0.03, 0.60)	(0.12, 0.72)	(0.11, 0.75)	(0.35, 0.80)
4803	0.40	0.49	0.43	**0.54**
	(-0.01, 0.64)	(0.24, 0.69)	(-0.06, 0.72)	(0.25, 0.77)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 14: Metrics Performance Across Dataset Sizes: RVL-CDIP

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.38	0.43	**0.59**	0.49
	(0.10, 0.62)	(0.16, 0.63)	(0.22, 0.78)	(0.16, 0.75)
1000	0.32	0.44	**0.60**	0.44
	(-0.15, 0.67)	(0.07, 0.69)	(0.17, 0.80)	(0.03, 0.69)
5000	0.36	0.41	**0.66**	0.55
	(-0.02, 0.63)	(0.04, 0.66)	(0.42, 0.83)	(0.17, 0.76)
10000	0.53	0.47	0.67	**0.71**
	(0.19, 0.72)	(0.10, 0.71)	(0.41, 0.84)	(0.46, 0.87)
50000	0.55	0.49	0.65	**0.73**
	(0.18, 0.76)	(0.12, 0.74)	(0.35, 0.82)	(0.47, 0.86)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 15: Metrics Performance Across Dataset Sizes: WikiArt

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.49 (0.19, 0.69)	0.44 (0.20, 0.63)	**0.73** (0.49, 0.89)	0.54 (0.20, 0.77)
1000	0.12 (-0.31, 0.50)	0.33 (-0.01, 0.59)	**0.54** (0.03, 0.76)	0.37 (-0.09, 0.70)
5000	0.47 (0.11, 0.71)	0.61 (0.32, 0.78)	**0.69** (0.42, 0.86)	0.61 (0.28, 0.80)
10000	0.65 (0.35, 0.81)	0.65 (0.39, 0.82)	**0.73** (0.47, 0.88)	0.69 (0.41, 0.84)
50000	0.62 (0.32, 0.80)	0.66 (0.37, 0.82)	**0.72** (0.40, 0.89)	0.71 (0.43, 0.84)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

D.2.2 Regret

Similar to the correlation results, we also present regret values for each combination of metric and dataset, with the dataset subset used for the metric computation varied. This again gives a similar picture to the top level results. Although there is greater variance in the measure of regret in some instances, there is no strong evidence overall for any of the metrics over the others considered.

Table 16: Metrics Regret Across Dataset Sizes: VG Faucet

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	**0.00**	1.58	0.11	0.11
1000	**0.68**	**0.68**	0.79	0.79
2303	0.23	**0.00**	1.13	1.13

Largest value for each row presented in bold.

Table 17: Metrics Regret Across Dataset Sizes: VG Tree

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	**0.68**	2.18	1.16	1.16
1000	**0.00**	2.28	0.86	0.86
5000	4.48	3.89	**3.53**	**3.53**
10000	3.95	3.95	**1.87**	**1.87**
26004	**0.03**	3.19	1.56	1.56

Largest value for each row presented in bold.

Table 18: Metrics Regret Across Dataset Sizes: VG Watch

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	**0.28**	0.92	1.20	1.20
1000	3.41	3.68	**1.20**	**1.20**
2825	4.93	**0.00**	0.46	0.46

Largest value for each row presented in bold.

Table 19: Metrics Regret Across Dataset Sizes: Oxford Pets

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	3.68	7.95	**0.16**	**0.16**
1000	1.24	2.60	**0.81**	**0.81**
4803	**0.49**	**0.49**	**0.49**	**0.49**

Largest value for each row presented in bold.

Table 20: Metrics Regret Across Dataset Sizes: RVL-CDIP

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	**0.68**	4.46	4.75	4.75
1000	3.58	**1.80**	5.24	5.24
5000	**2.36**	**2.36**	**2.36**	**2.36**
10000	**1.10**	**1.10**	**1.10**	**1.10**
50000	**0.12**	4.28	**0.12**	**0.12**

Largest value for each row presented in bold.

Table 21: Metrics Regret Across Dataset Sizes: WikiArt

N. Images	Renggli	LogME	N. Params	ImageNet Acc.
500	0.07	1.20	**0.00**	**0.00**
1000	6.32	**3.23**	6.32	6.32
5000	**0.00**	**0.00**	3.38	3.38
10000	**0.00**	**0.00**	2.65	2.65
50000	**0.00**	**0.00**	1.62	1.62

Largest value for each row presented in bold.

D.3 Combining Metrics

Here we include ancillary tables showing the results for metric combinations for each dataset (at the maximum training and metrics subset sizes used in our experiments).

Table 22: Metric Combination Performance for Each Dataset, Assessed by Correlation

Metric Combinations	VG Faucet	VG Tree	VG Watch	RVL-CDIP	WikiART	Oxford Pets
Renggli	0.25 (-0.18, 0.60)	0.38 (-0.05, 0.65)	0.04 (-0.41, 0.43)	0.55 (0.19, 0.76)	0.62 (0.32, 0.79)	0.40 (0.00, 0.64)
LogME	**0.52** (0.09, 0.78)	0.32 (-0.10, 0.61)	0.50 (0.14, 0.74)	0.49 (0.12, 0.74)	0.66 (0.38, 0.82)	0.49 (0.25, 0.68)
N. Params	0.38 (0.05, 0.66)	**0.50** (0.09, 0.71)	0.59 (0.18, 0.79)	0.65 (0.34, 0.82)	**0.72** (0.40, 0.89)	0.43 (-0.03, 0.72)
ImageNet Acc.	0.38 (-0.02, 0.70)	**0.50** (0.22, 0.72)	**0.71** (0.45, 0.86)	**0.73** (0.46, 0.86)	0.71 (0.44, 0.84)	0.54 (0.24, 0.76)
Renggli & LogME	0.43 (0.01, 0.71)	0.33 (-0.08, 0.61)	0.49 (0.03, 0.76)	0.55 (0.18, 0.77)	0.63 (0.35, 0.79)	0.49 (0.19, 0.71)
Renggli & N. Params	0.29 (-0.07, 0.60)	0.35 (-0.02, 0.60)	0.22 (-0.21, 0.53)	0.57 (0.25, 0.77)	0.63 (0.32, 0.80)	0.35 (-0.13, 0.67)
Renggli & ImageNet Acc.	0.35 (-0.07, 0.68)	0.44 (0.10, 0.66)	0.47 (0.07, 0.70)	0.67 (0.35, 0.84)	0.68 (0.43, 0.84)	0.46 (0.10, 0.68)
LogME & N. Params	0.39 (0.01, 0.70)	0.29 (-0.11, 0.58)	0.54 (0.12, 0.77)	0.53 (0.20, 0.73)	0.63 (0.33, 0.77)	0.43 (-0.04, 0.70)
LogME & ImageNet Acc.	0.45 (0.03, 0.74)	0.42 (0.06, 0.64)	0.69 (0.29, 0.87)	0.60 (0.25, 0.80)	0.69 (0.41, 0.85)	**0.55** (0.28, 0.75)
N. Paramas & ImageNet Acc.	0.32 (-0.05, 0.65)	0.41 (0.01, 0.66)	0.59 (0.19, 0.80)	0.65 (0.36, 0.82)	0.61 (0.26, 0.78)	0.42 (-0.03, 0.72)
Renggli & LogME & N. Params	0.37 (-0.01, 0.66)	0.30 (-0.08, 0.58)	0.47 (0.01, 0.73)	0.56 (0.27, 0.75)	0.66 (0.37, 0.81)	0.42 (-0.01, 0.71)
Renggli & LogME & ImageNet Acc.	0.45 (0.01, 0.75)	0.37 (-0.03, 0.62)	0.61 (0.20, 0.82)	0.60 (0.24, 0.79)	0.69 (0.40, 0.84)	0.51 (0.19, 0.74)
Renggli & N. Params & ImageNet Acc.	0.37 (0.01, 0.68)	0.36 (0.02, 0.62)	0.55 (0.13, 0.78)	0.62 (0.31, 0.80)	0.64 (0.33, 0.81)	0.38 (-0.07, 0.67)
LogME & N. Params & ImageNet Acc.	0.40 (0.01, 0.70)	0.37 (0.02, 0.61)	0.60 (0.16, 0.81)	0.59 (0.27, 0.79)	0.64 (0.33, 0.81)	0.46 (0.03, 0.72)
Renggli & LogME & N. Params & ImageNet Acc.	0.40 (0.01, 0.70)	0.35 (-0.02, 0.60)	0.58 (0.15, 0.79)	0.58 (0.28, 0.77)	0.67 (0.40, 0.82)	0.42 (-0.02, 0.70)

Values presented with bootstrapped 95% confidence intervals.
Largest value for each row presented in bold.

Table 23: Metric Combinations Assessed by Regret

Metric Combinations	**VG Faucet**	**VG Tree**	**VG Watch**	**RVL-CDIP**	**WikiART**	**Oxford Pets**
Renggli	0.23	**0.03**	4.93	**0.12**	**0.00**	**0.49**
LogME	**0.00**	3.19	**0.00**	4.28	**0.00**	**0.49**
N. Params	1.13	1.56	0.46	**0.12**	1.62	**0.49**
ImageNet Acc.	1.13	1.56	0.46	**0.12**	1.62	**0.49**
Renggli & LogME	0.45	3.19	**0.00**	**0.12**	**0.00**	**0.49**
Renggli & N. Params	1.13	**0.03**	0.46	**0.12**	1.62	**0.49**
Renggli & ImageNet Acc.	**0.00**	**0.03**	0.46	**0.12**	**0.00**	**0.49**
LogME & N. Params	1.13	2.91	5.99	**0.12**	1.62	**0.49**
LogME & ImageNet Acc.	0.34	**0.03**	**0.00**	**0.12**	**0.00**	**0.49**
N. Params & ImageNet Acc.	1.13	1.56	0.46	**0.12**	1.62	**0.49**
Renggli & LogME & N. Params	1.13	**0.03**	0.46	**0.12**	**0.00**	**0.49**
Renggli & LogME & ImageNet Acc.	**0.00**	**0.03**	**0.00**	**0.12**	**0.00**	**0.49**
Renggli & N. Params & ImageNet Acc.	1.13	1.56	0.46	**0.12**	1.62	**0.49**
LogME & N. Params & ImageNet Acc.	1.13	1.56	0.46	**0.12**	1.62	**0.49**
Renggli & LogME & N. Params & ImageNet Acc.	1.13	**0.03**	0.46	**0.12**	1.62	**0.49**

Largest value for each row presented in bold.