Model Lineage Closeness Analysis

Chen Tang[1], Lan Zhang[1,2*], Qi Zhao[1], Xirong Zhuang[1], Xiangyang Li[1]

[1]University of Science and Technology of China, China
[2]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, China
{chentang1999, zq2021, xirongz}@mail.ustc.edu.cn, zhanglan@ustc.edu.cn

Abstract

As machine learning model modification techniques are extensively employed to obtain well-performing models at reduced costs, several studies have emerged to determine the presence of a modification relationship (i.e., lineage) between models. However, these methods are not robust to high-impact modification techniques, and none of them have addressed the measurement of lineage closeness, which quantifies the degrees of modification. In this work, we visualize the changes in model decision boundaries resulting from different modification techniques and conclude that differences in decision boundaries serve as a precise metric of lineage closeness. Building upon this insight, we propose a modification-type agnostic and task-agnostic method to measure model lineage closeness by calculating mean adversarial distances from data points to decision boundaries and the matching rate of data points, with data points selected through an efficient sampling method to reduce computational overhead. Moreover, we propose a novel indirect measurement approach to support lineage closeness measurement for models with different tasks. Finally, comprehensive experiments show that our design achieves an impressive 97% accuracy in lineage determination and can precisely measure model lineage closeness for different modifications.

Introduction

Machine learning models are increasingly utilized across a diverse range of applications with remarkable success. However, generating a well-trained model is a challenging task that requires significant hardware resources and training data. Consequently, various model modification techniques, such as fine-tuning, pruning, adversarial training, quantization, transfer learning, and distillation, have been proposed to generate models based on existing models, thereby saving training costs. These techniques have gained immense popularity among model developers due to their convenience and impressive performance.

With the advent of model modification techniques, a new question arises: how can we measure the similarity between models with a modification relationship? To distinguish this modification similarity from the similarity of model parameters or structures, here, **we define model lineage as models with a modification relationship and use the model lineage closeness to indicate the model modification degree.** A higher lineage closeness denotes a lower model modification degree.

Measuring model lineage closeness is a crucial task. First, lineage closeness between models can provide valuable insights. By measuring lineage closeness, we can better quantify the effects of different model modification techniques, thereby deepening our understanding of neural networks. Second, precise metrics for model lineage closeness can benefit various downstream applications, such as model copyright protection, by helping us determine the source of a model and identify potential infringement. However, there is still a lack of precise metrics for measuring model lineage closeness.

It is non-trivial to measure model lineage closeness, which faces two key challenges. **The diversity and complexity of model modification methods make it difficult to determine whether there is a lineage between two models.** Model lineage is not determined by training dataset, model structure, or hyperparameters, but by the modification relationships between models. Characterizing such modification relationships poses a considerable challenge. Moreover, models with similar performance on the same task but no lineage can lead to misclassifications in lineage determination. Recently, several works (Cao, Jia, and Gong 2021; Li et al. 2021; Yang, Wang, and Wang 2022; Peng et al. 2022; Lukas, Zhang, and Kerschbaum 2021; Guan, Liang, and He 2022) have explored this problem. The key observation is that models with lineage tend to exhibit more similar decision boundaries, while models without lineage display greater dissimilarities in their decision boundaries. Based on this insight, these studies propose generating data points near the decision boundaries of models. Since models with similar decision boundaries will produce the same outputs for these data points, they determine lineage based on whether the models have the same outputs for these data points. However, modifications to the model lead to shifts in decision boundaries, resulting in changes to the outputs of these data points, thus leading to incorrect lineage determination. Besides, some modifications, such as transfer learning, result in different model output dimensions, invalidating these lineage determination methods as they require the same output dimensions between models. **There is a lack**

*Corresponding Author

of effective metrics to measure the lineage closeness between models. Different modifications affect models differently, resulting in varying lineage closeness. Intuitively, a slight modification to a model results in high lineage closeness, whereas a substantial modification results in lower lineage closeness. However, measuring lineage closeness requires precisely capturing the degree of modification, and no such metric is currently available.

To address the aforementioned challenges, we first investigate six popular model modification techniques and categorize them based on whether they maintain the same task for the modified model and whether the modification is performed in-place. We then visualize and analyze the impact of different modification types on model decision boundaries. We conclude that differences in models' decision boundaries serve as a precise metric of lineage closeness between models. Following this, we design a model lineage closeness measurement method that is agnostic to both modification types and tasks. This innovative method integrates the mean adversarial distance and matching rate to ascertain the similarity of model decision boundaries as their lineage closeness. Mean adversarial distance represents the mean distances from data points along the gradient direction to the model decision boundary, and the matching rate denotes the proportion of data points exhibiting consistent outputs on models. Since data points are sampled from the training dataset rather than generated near the decision boundaries as in existing works, our method is not affected by shifts of decision boundaries and is applicable to any modification type, i.e. modification-type agnostic. Additionally, we design an indirect measurement method to align inconsistent decision space dimensions to support lineage closeness measurement for models with different tasks, i.e. task-agnostic. Ultimately, we design a data points sampling method that greatly reduces the computational overhead of lineage closeness measurement without compromising the accuracy of lineage closeness measurement.

In summary, our main contributions are three-fold: 1). Our work is the first to address model lineage closeness, exploring the impact of various model modification techniques on the model and proposing the similarity of model decision boundaries as a metric for measuring model lineage closeness. 2). We propose a modification-type agnostic and task-agnostic method to measure model lineage closeness. This innovative method overcomes the limitations of previous methods that support precise lineage closeness measurement between models with different modification types and different tasks. 3). We construct a comprehensive benchmark that covers 13 distinct types of model modifications and contains 136 models. Based on this benchmark, our method achieves a high accuracy of 97% for lineage determination and can precisely measure model lineage closeness for any two models. We make our code and this benchmark open-source.[1]

Related Work

Since 2019, several methods have been proposed to determine model lineage. Most of these approaches involve generating model-specific inputs known as fingerprints, such as adversarial examples (Cao, Jia, and Gong 2021), conferrable adversarial examples (Lukas, Zhang, and Kerschbaum 2021), universal adversarial perturbations (Peng et al. 2022) and noise images generated through meta-training (Yang, Wang, and Wang 2022). These works determine whether a suspect model has lineage with the model based on whether it predicts the same labels for a majority of these fingerprints. Besides, other works focus on evaluating differences in models' performance. For instance, (Li et al. 2021) propose to calculate the cosine similarity of models' decision distance vectors to assess whether the suspect model shares lineage with the model. (Chen et al. 2022) suggest using a diverse set of testing metrics to determine model lineage. (Guan, Liang, and He 2022) propose to calculate the correlation matrix between models' outputs and compare them with a preset threshold for lineage determination.

However, these methods are sensitive to high-impact modifications, which can cause significant shifts in decision boundaries, rendering the generated fingerprints invalid or inaccurately scored. In addition, most of these methods require consistency in the models' output dimensions, making them ineffective for tasks involving different output dimensions. Furthermore, these methods do not account for the degree of modification, limiting their effectiveness as metrics for model lineage closeness. In this paper, we explore the effects of various modification types on models and conduct an in-depth analysis of model lineage closeness in Sec 3. This analysis informs the design of a modification-type agnostic and task-agnostic method for measuring model lineage closeness in Sec 4.

Exploring Model Lineage

In this section, we first state the problem and categorize various modification techniques, then use the decision boundary visualization method (Somepalli et al. 2022) to analyze the impact of these modifications on model lineage closeness.

Problem Statement and Design Space

In model lineage closeness measurement, we consider two classification models: the source model f and the compared model g, as well as the training dataset d of f. The objective is to use d to calculate a score that represents the degree of modifications from f to g. Formally, the model lineage closeness measurement involves a two-step process:

Step 1: A test set generation $G(f, d) = d_t$, takes the source model f and the dataset $d = \{\boldsymbol{x_i}, y_i\}$ as inputs, outputs the test set d_t.

Step 2: A lineage closeness score calculation $L(f, g, d_t) = s$, takes the source model f, compared model g, test set d_t as inputs, and outputs the lineage closeness score $s \in [0, 1]$, which indicates the degree of modifications from f to g.

In our design space, we consider six common modification techniques, which include: (1) Fine-tuning (Adi et al. 2018) alters the model's parameters by training for a few additional epochs on a subset of the model's parameters using

[1] https://github.com/chentangUSTCCS/Model_Lineage_Closeness

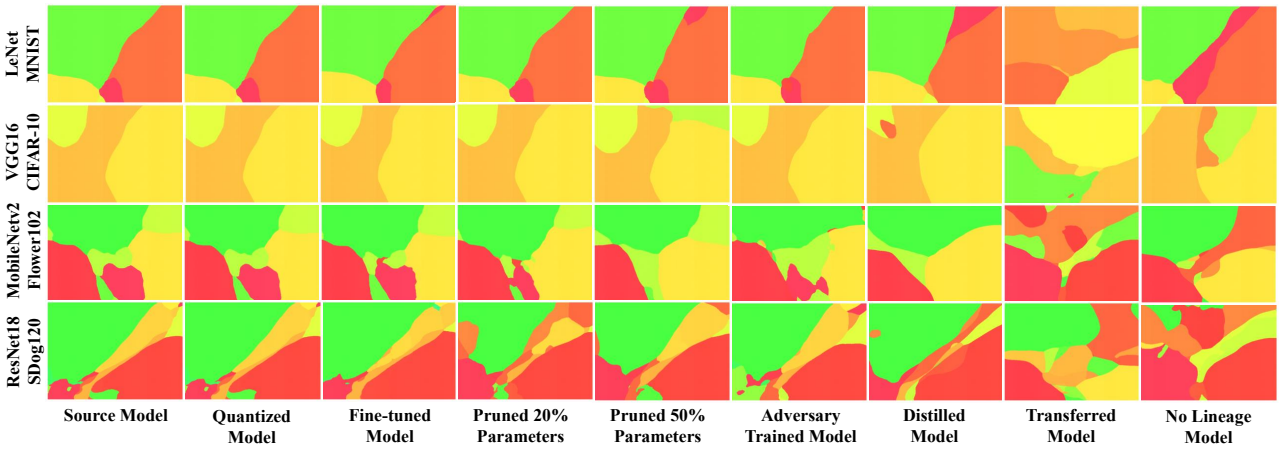


Figure 1: An example of decision boundary visualization results of the source model, lineage models, and no lineage model. Different colors indicate different decision areas, between which are decision boundaries.

	Same task	Different task
In-place	Fine-tune, Quantization, Prune, Adversarial Training	Transfer Learning
Out-place	Distillation	-

Table 1: Dividing different modification techniques based on whether the task remains the same and whether the modification is performed in-place. In-place modifications involve direct alterations to the source model, e.g., prune, while out-place indicates generating a new model, e.g., distillation.

the training dataset; (2) Pruning (Han et al. 2015) reduces the size of less important parts of the model's parameters; (3) Adversarial training (Madry et al. 2018a) improves robustness against adversarial attacks by training with adversarial samples; (4) Quantization (Han, Mao, and Dally 2016) compresses the model by converting its parameters to integer or lower-precision floating-point types; (5) Transfer learning (Simonyan and Zisserman 2015a) transfers the knowledge from the model to a different but related task by training with the dataset of the related task; (6) Feature-based knowledge distillation (Hinton, Vinyals, and Dean 2015) distills knowledge from the model to train a new model. Feature-based distillation requires the model to have the same structure as the source model to utilize the intermediate layers' outputs. For the rest of this paper, distillation refers to the feature-based distillation.

We divide six common modification techniques into three categories based on the task and the in-place modification of the model, as shown in Tab.1. In the first category, we identify four modification techniques: fine-tuning, pruning, adversarial training, and quantization. The second category includes only transfer learning as the sole technique. Finally, distillation is the only technique in the third category.

Exploring Model Lineage

Leveraging the decision boundary visualization technique, we explore the differences among different categories to further explore how different modification techniques affect the model lineage closeness. Specifically, we used four datasets: MNIST (LeCun et al. 1998), CIFAR-10 (Krizhevsky, Hinton et al. 2009), Flower102 (Nilsback and Zisserman 2008) and SDog120 (Khosla et al. 2011), four model structures: LeNet (LeCun et al. 1998), VGG16 (Simonyan and Zisserman 2015b), MobileNetv2 (Sandler et al. 2018) and ResNet18 (He et al. 2016), to train two sets of models. Each set of models consists of three types: the source model, lineage models (generated by modifying the source model), and no-lineage models (trained with different hyperparameters or structures than the source model). Then we visualized different regions of their decision boundaries multiple times. Fig.1 is an example of decision boundary visualization results. And we have the following observations:

•**Observation #1**: Models with lineage display a significant similarity in their decision boundaries, whereas models without lineage exhibit notable differences in their decision boundaries.

Theoretically, during model training, a model can converge to multiple local optima with similar performance levels. In contrast, variations in hyperparameters or structural settings typically lead the model to converge to different local optima (Ainsworth, Hayase, and Srinivasa 2023). Our experiments reveal that models converging to different local optima (i.e., no lineage models) often exhibit widely varying decision boundaries. Conversely, lineage models, being direct modifications of the source model, are less likely to diverge into other local optima, resulting in substantial similarity in their decision boundaries, as illustrated in Fig.1.

•**Observation #2**: In the first category of modification techniques, different modification techniques have varying degrees of influence on the model decision boundary, e.g., adversarial training impacts the decision boundary more significantly than fine-tuning. Moreover, different settings of the same modification technique also produce varying degrees of influence, e.g., pruning 50% of the parameters has a greater influence on the decision boundary than pruning 20%.

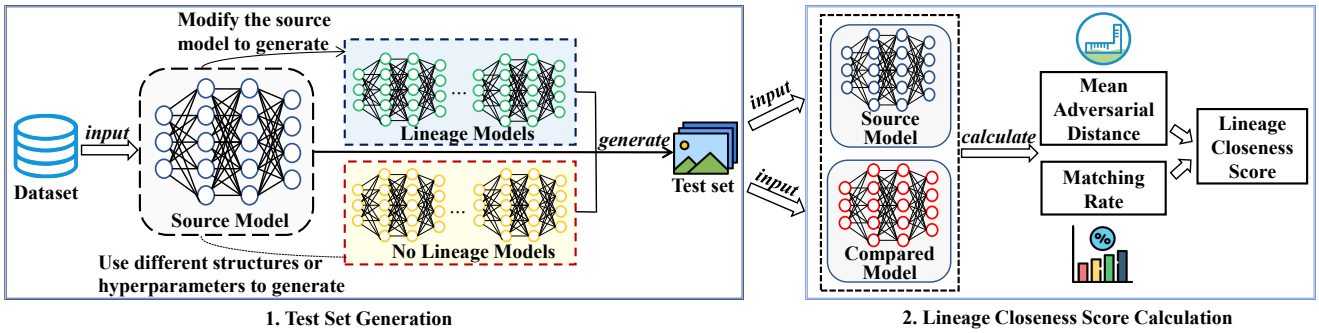


Figure 2: Model lineage closeness measurement approach workflow.

•**Observation #3**: In the second category of modification techniques, transferred models with different tasks have inconsistent decision space dimensions with the source model.

•**Observation #4**: In the third category of modification techniques, the decision boundaries of distilled models are less similar to the source model than other modified models.

In distillation, since model outputs of the intermediate layers for different local optimal solutions are inconsistent (Ainsworth, Hayase, and Srinivasa 2023), feature-based distillation makes the model's outputs consistent in the intermediate layers so that the models converge to the same local optimal solution with high probability, thus resulting in weakly similar decision boundaries.

•**Observation #5**: Some parts of decision boundaries are highly sensitive to modifications, with even small changes causing significant shifts. Conversely, some parts of the decision boundaries are commonly present in the source model, lineage models, and no-lineage models.

When calculating the similarity of models' decision boundaries, these two types do not contribute to the result. This is because one type is overly sensitive to modifications, and the other is ubiquitous, both of which distract from accurately calculating the similarity of decision boundaries.

In summary, based on the above observations, we found that models with lineage have similar decision boundaries, and the similarity of decision boundaries is a suitable metric for model lineage closeness, reflecting the degree of model modification. However, we need to further address the following challenges: 1) How can we quantify the difference in decision boundaries to measure the degrees of modifications between models? 2) How can we overcome the inconsistency in the dimensions of decision space brought by different tasks? 3) How can we avoid the impact of these two types of decision boundaries on the result of the lineage closeness measurement? To address these challenges, we design a modification-type agnostic and task-agnostic model lineage closeness measurement method, which we describe in the next section.

Model Lineage Closeness Measurement

Approach Design

The workflow of our design is shown in Fig.2. The key components include a test set generator and a lineage closeness score calculator. The test set generator efficiently generates a test set for lineage closeness score calculation, while the lineage closeness score calculator takes the test set as input to both the source model and the compared model to calculate the lineage closeness score. Since the data points used to measure lineage closeness are far from the decision boundary, our method is not affected by shifts in the model decision boundary due to modifications, i.e., modification-type agnostic. Moreover, we propose an indirect measurement method that transfers different tasks to the same task to measure lineage closeness, thereby achieving task-agnostic measurement. The details are as follows.

Lineage Closeness Score Calculation

Based on the above observations $\#1, \#2,$, we adopt decision boundary similarity to measure model lineage closeness. Specifically, for the source model f and the compared model g, we calculate the similarity of their decision boundaries by computing the distance from data points to the decision boundaries of different models. Here, we use the adversarial distance (Tramèr et al. 2017), which is the shortest distance from the data point along the gradient direction to the decision boundaries of the models. As shown in case 1 of Fig.3, for each data point $\boldsymbol{x}$, we first calculate its gradient direction by:

$$\boldsymbol{grad} = sign(\nabla f(\boldsymbol{x})). \tag{1}$$

Then we calculate the adversarial distance along the gradient direction from the data point $\boldsymbol{x}$ to decision boundaries of different models separately, that is:

$$d_f = \arg\min_{d_f}[f(\boldsymbol{x} + d_f * \boldsymbol{grad}) \neq f(\boldsymbol{x})], \tag{2}$$

$$d_g = \arg\min_{d_g}[g(\boldsymbol{x} + d_g * \boldsymbol{grad}) \neq g(\boldsymbol{x})]. \tag{3}$$

Besides, similar decision boundaries should produce the same output on both sides. Therefore, we need to confirm whether the outputs of the data points on both sides of the decision boundary are equal, i.e.,

$$f(\boldsymbol{x}) = g(\boldsymbol{x}), \;\; f(\boldsymbol{x}+d_f*\boldsymbol{grad}) = g(\boldsymbol{x}+d_g*\boldsymbol{grad}). \tag{4}$$

As shown in case 2 of Fig.3, it can be observed that data points yield different outputs on different models. In this case, the computed adversarial distance only reflects the distance of the data points to the decision boundary of different

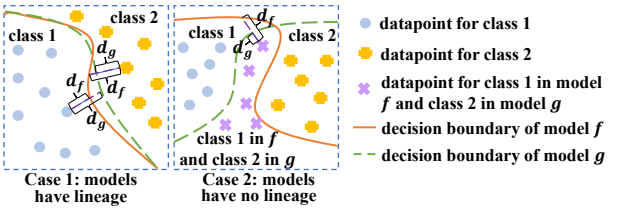


Figure 3: Adversarial distance along the gradient direction from the data point to different models' decision boundaries.

models and cannot be used to measure model lineage closeness. To address this limitation, we introduce an additional metric, the matching rate, to complement the measurement of model lineage closeness. Let N represent the total number of data points. We compute the adversarial distances for all data points and count the number of data points that satisfy Eq.(4), denoted as k. The matching rate Mr is calculated by:

$$Mr = \frac{k}{N}, Mr \in [0, 1]. \tag{5}$$

We call a match successful when the outputs of the data points are the same on both sides of the decision boundary of the source and the compared model, i.e., satisfy Eq.(4). The matching rate corresponds to the proportion of successfully matched data points out of the total number of data points. A higher Mr indicates greater similarity in the decision boundaries of models. In addition, considering the definition domain of data points, we set a maximum value of 1 for the adversarial distance. Any adversarial distance exceeding 1 will be deemed a failed matching. After identifying all the successfully matched data points, denoted as M, we calculate the mean adversarial distance differences from each data point to different decision boundaries, as follows:

$$\overline{d} = \frac{1}{|M|} \sum_{x \in M} |d_f - d_g|, \overline{d} \in [0, 1]. \tag{6}$$

Finally, we combine the above two metrics to calculate the model lineage closeness s:

$$s = (1 - \overline{d}) * Mr, s \in [0, 1]. \tag{7}$$

Unlike existing efforts that are sensitive to shifts in model decision boundaries due to modifications, s directly captures the distances and changes of model decision boundaries, which are unaffected by modifications, i.e., modification-type agnostic.

Test Set Generation

Utilizing the entire dataset to measure lineage closeness entails a significant computational burden. Furthermore, as discussed in observation #5, there are two special types of decision boundaries: those sensitive to model modifications and those that are common across models. These decision boundaries do not contribute to accurate lineage closeness measurements. To address these issues, we propose a two-step process for generating the test set, described as follows:

1. Remove invalid samples: First, to mitigate the impact of sensitive decision boundaries on our results, we modify (e.g., fine-tuning) the source model to generate a set of models with high lineage closeness. Then for each data point, we verify whether it matches successfully with both the source model and the lineage models, removing those that do not match. Second, to avoid the effect of common decision boundaries, we generate no lineage models using different training settings. We then check whether the remaining data points match successfully with both the source model and the no lineage models, removing those that do match.

2. Sample discrepancy data points: After removing invalid samples, a large number of data points remain. To further reduce computational overhead, we aim to sample a representative subset of data points for lineage closeness measurement. Given that measuring lineage closeness involves calculating the adversarial distance based on the gradient of data points, greater differences among data points lead to more pronounced variations in gradient directions, which better capture the similarity of decision boundaries. Therefore, we sample K data points that exhibit the greatest differences from each other. However, selecting these data points is an NP-complete problem and cannot be solved in polynomial time. To address this, we use an approximation approach: calculating the sum of differences between each data point and all other points, ranking these differences, and selecting the top K data points as the final test set d_t for measuring model lineage closeness. Here, K is the number of data points predetermined for lineage closeness measurement.

Indirect Measurement

Based on insight #3, transfer learning leads to inconsistency in the decision space dimensions, which renders our proposed method for measuring model lineage closeness impractical. However, examination reveals that transfer learning primarily affects the model's classification dimensions while preserving its feature extraction capability. Essentially, the feature extraction ability of the model remains largely unaffected. Therefore, we propose an indirect measurement, approach for assessing lineage closeness between models with different tasks. This approach adds an extra step to the model lineage closeness measurement process described above. Specifically, we transfer the source model using the training dataset of the compared model to align the decision space dimensions. The hyperparameters for transfer learning (epochs, learning rate, etc) only need to be set appropriately. Subsequently, we measure the lineage closeness score between the transferred source model and the compared model. This calculated score is then considered as the lineage closeness between the source model and the compared model. In this way, our method achieves task-agnostic.

The effectiveness of this method is largely attributed to the preservation of the model's feature extraction ability. Although the dimensions of the feature representations may differ, utilizing the same dataset ensures that the feature representations of the models remain as similar as possible.

Experiments

In this section, we construct a comprehensive benchmark based on the ModelReuse benchmark (Li et al. 2021), cover-

Baselines and our method			#Models	IPGuard	CAE	ModelDiff	SAC	MeTaFinger	DEEPJUDGE	**Ours**
Model types	Lineage models	Fine-tune	28	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)
		Prune	28	18(64%)	21(75%)	28(100%)	28(100%)	23(82%)	21(75%)	28(100%)
		Adversarial training	14	11(78%)	12(86%)	14(100%)	14(100%)	13(93%)	12(86%)	14(100%)
		Quantization	28	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)	28(100%)
		Transfer learning	14	-	-	14(100%)	14(100%)	-	-	14(100%)
		Distillation	14	0(0%)	9(64%)	14(100%)	13(93%)	10(71%)	4(29%)	9(64%)
	No lineage models		72	72(100%)	72(100%)	21(29%)	7(10%)	72(100%)	72(100%)	72(100%)
Determination accuracy overall			198	157(79%)	170(86%)	147(74%)	125(63%)	174(88%)	165(83%)	**193(97%)**
Ability to determine transferred model lineage				×	×	√	√	×	×	√
Ability to measure lineage closeness				×	×	×	×	×	×	√

Table 2: Evaluation results of the benchmark. The '#Models' column is the number of model pairs.

ing common modification techniques. Based on the benchmark, we compare the correctness and efficiency with baselines and demonstrate the precise lineage closeness measurement of our method.

Benchmark and Experiments Setup

To develop a more comprehensive benchmark for exploring model lineage closeness, we extend the existing benchmark (Li et al. 2021) to cover all modification types. In this extended benchmark, we maintain the model structures used in the original benchmark, namely MobileNetv2 and ResNet18, as well as the dataset: Oxford Flowers 102, Stanford Dogs 120, and ImageNet (Deng et al. 2009). The updated benchmark now encompasses 136 models, including 2 pretrained models, 108 lineage models(12 transferred + 24 fine-tuned + 24 pruned + 12 adversarial trained + 24 quantized + 12 distilled), and 26 retrained models. Each transferred model is built from one of the pretrained models and each lineage model is built from one of the transferred models, and the 26 retrained models are trained from scratch.

For comparison, we use IPGuard (Cao, Jia, and Gong 2021), CAE (Lukas, Zhang, and Kerschbaum 2021), ModelDiff (Li et al. 2021), SAC (Guan, Liang, and He 2022), MeTaFinger (Yang, Wang, and Wang 2022) and DEEPJUDGE (Chen et al. 2022) as our baselines for comparison. For IPGuard, we leverage PGD (Madry et al. 2018b) to generate 100 adversarial samples as its fingerprint. For CAE, we generate 3 lineage models and take 2 no lineage models to generate 100 conferrable adversarial examples as its fingerprint. For ModelDiff, we generate 100 test inputs to compare models' behavior patterns. For SAC, we generate 100 test inputs to calculate the correlation distance. For MeTaFinger, we generate 100 samples by meta-training as its fingerprint. For DEEPJUDGE, we leverage PGD to generate 100 fingerprints and take their black-box evaluation method to determine the lineage. We set the determination threshold to the lowest value of all baselines, i.e. 0.5. Moreover, all experiments are conducted on a Linux Server with 1 Tesla P100 GPU and implemented with PyTorch 1.5 using Python 3.7.

Correctness

To compare the correctness of our method with baselines, we generate 100 samples for lineage determination and set a determination threshold δ. Specifically, a lineage is identified between models if their lineage closeness exceeds δ. To determine the threshold, we generate 4 lineage models and 2 no lineage models for each source model and calculate the lineage closeness score. Then we set the threshold δ to 0.35 which can well distinguish generated lineage models and no lineage models. The set of models to set up thresholds (24 models) is different from the set of models for evaluation (198 models) in experiments.

Based on the benchmark, we generated 198 model pairs: 126 model pairs with lineage and 72 model pairs without lineage. This setup allows us to comprehensively evaluate the determination accuracy of both the baseline methods and our proposed approach. The results are shown in Tab.2. Our method achieves a remarkable accuracy rate of 97% in determining model lineage, surpassing the accuracy of IPGuard (79%), CAE (86%), ModelDiff (74%), SAC (63%), MeTaFinger (88%), and DEEPJUDGE (83%). In our method, only five model pairs were misclassified, all of which are distilled models. Upon visualizing their decision boundaries, we observed substantial disparities between the decision boundaries of the distilled models and their respective source models. This discrepancy in decision boundaries indicates a low lineage closeness, leading to the inability to determine the lineage accurately. Conversely, other baselines frequently struggle to determine lineage accurately when the source model undergoes substantial modifications such as pruning and adversarial training. While ModelDiff and SAC exhibit strong performance in determining lineage models, they demonstrate a higher misjudgment rate for no lineage models.

Moreover, compared to other baselines, our method offers the advantage of supporting lineage determination for models with different tasks, as well as the ability to measure the lineage closeness between any two models. In contrast, the outcomes produced by baselines lack clear distinction when applied to models with various modification types. As a result, these baselines are insufficient for precisely measuring model lineage closeness.

Efficiency

We compare the time efficiency of our method with that of baseline approaches. The time overhead includes model generation, test set generation, and lineage measurement. The results are presented in Tab.3. First, our method generates models with high lineage closeness through fine-tuning and pruning, which is less time-consuming compared to the approaches used by CAE and MeTaFinger. Consequently, the time overhead for model generation in our method is lower than that of these baselines. Second, unlike the baselines,

	Models generation	Test set generation	Lineage measurement
IPGuard	-	7	1
CAE	>10000	103	2
ModelDiff	-	27	0.58
SAC	-	24	28
MeTaFinger	>4000	152	2
DEEPJUDGE	-	7	3
Ours	2000	800	10

Table 3: Comparison of the average time costs(sec.).

our method selects the most suitable 100 samples from all available samples for measuring model lineage closeness, rather than generating 100 samples that merely meet certain requirements (e.g., adversarial samples). This additional step incurs approximately 800 seconds of extra time overhead. Third, our method performs a fine-grained approach for measuring lineage closeness, which requires a longer time (approximately 10 seconds) during the lineage measurement phase.

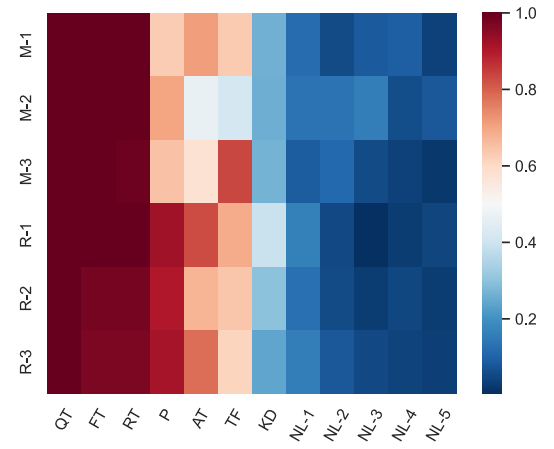
In conclusion, while our method reduces time overhead in model generation, it incurs additional time overhead in test set generation. However, both the model and test set can be pre-generated and saved in advance. During the lineage measurement phase, our method can directly utilize the pre-generated test set, which takes only 10 seconds and is acceptable in practice.

Quantization Analysis for Model Modification

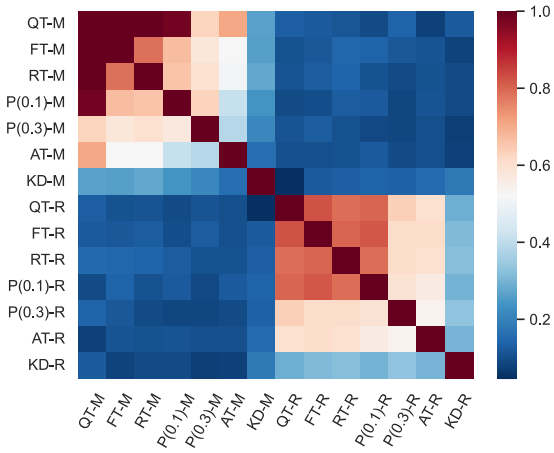
In this section, we perform fine-grained quantitative analysis for different types of lineage models, including lineage and indirect lineage. The results are as shown in Fig.4(a) and Fig.4(b), in which M and R are short of MobileNetv2 and ResNet18. Modification types include fine-tuning the last layer(FT for short), retraining the last layer(RT for short), pruning (P for short), adversarial training(AT for short), quantization(QT for short), and distillation(KD for short).

Lineage Closeness Measurement We generate 6 source models: 3 using MobileNetv2 and 3 using ResNet18. For each source model, we generate 7 lineage models and 5 no lineage models. Then we measure the lineage closeness between these models and the source model. The results are shown in Fig.4(a). The horizontal and vertical coordinates indicate the source model and the compared model(lineage model or no lineage model), where each lattice indicates the lineage closeness score s between them. We observe there is a significant difference of s between models with and without lineage, and modifying the model with a larger degree has a lower s. These results confirm the soundness of our observations and the lineage closeness measurement method.

Indirect Lineage Indirect lineage refers to the lineage between models achieved through different modification techniques applied to a single source model. For example, if model A is respectively modified to models B (A$\rightarrow$ B) and C (A$\rightarrow$ C), then there is an indirect lineage between B and C. In the experiment, We generated 2 source models and 7 lineage models for each source model. We computed the s between these 14 models two by two and the results are shown in Fig.4(b). Each lattice indicates the s between the corresponding two models. We observed that models with and without indirect lineage still show significant differences of s and the indirect lineage closeness has lower s than the lineage closeness. These results show that our method accurately captures the differences in model modifications.



(a) Lineage closeness measurement.



(b) Indirect lineage closeness measurement.

Figure 4: 1-3 are the index of source models. NL is the short of no lineage and 1-5 is the index of no lineage models. P(0.1) is the short of prune 10% weight, and P(0.3) is the short of prune 30% weight.

Conclusion

In this work, we visualize the decision boundaries of different types of lineage models and propose a modification-type agnostic and task-agnostic method to measure lineage closeness based on the similarity of decision boundaries. In addition, we reduce computational cost by an efficient test set generation method. Our comprehensive experiments show that our method provides a precise measurement solution for model lineage closeness.

Acknowledgments

The research is partially supported by China National Natural Science Foundation with No. 61932016, National Key R&D Program of China under Grant No. 2021ZD0110400, Innovation Program for Quantum Science and Technology 2021ZD0302900 and China National Natural Science Foundation with No. 62132018, 62231015, "Pioneer" and "Leading Goose R&D Program of Zhejiang", 2023C01029, and 2023C01143, "the Fundamental Research Funds for the Central Universities" WK2150110024, Science and Technology Tackling Program of Anhui Province No.202423k09020016.

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 1615–1631.

Ainsworth, S. K.; Hayase, J.; and Srinivasa, S. S. 2023. Git Re-Basin: Merging Models modulo Permutation Symmetries. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Cao, X.; Jia, J.; and Gong, N. Z. 2021. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 14–25.

Chen, J.; Wang, J.; Peng, T.; Sun, Y.; Cheng, P.; Ji, S.; Ma, X.; Li, B.; and Song, D. 2022. Copy, right? A testing framework for copyright protection of deep learning models. In *2022 IEEE Symposium on Security and Privacy (SP)*, 824–841. IEEE.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 248–255. IEEE Computer Society.

Guan, J.; Liang, J.; and He, R. 2022. Are you stealing my model? sample correlation for fingerprinting deep neural networks. *Advances in Neural Information Processing Systems*, 35: 36571–36584.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In Bengio, Y.; and LeCun, Y., eds., *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Li, F.-F. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *University of Toronto*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, Y.; Zhang, Z.; Liu, B.; Yang, Z.; and Liu, Y. 2021. ModelDiff: testing-based DNN similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 139–151.

Lukas, N.; Zhang, Y.; and Kerschbaum, F. 2021. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018a. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018b. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729. IEEE.

Peng, Z.; Li, S.; Chen, G.; Zhang, C.; Zhu, H.; and Xue, M. 2022. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13430–13439.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Simonyan, K.; and Zisserman, A. 2015a. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Simonyan, K.; and Zisserman, A. 2015b. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Somepalli, G.; Fowl, L.; Bansal, A.; Yeh-Chiang, P.; Dar, Y.; Baraniuk, R.; Goldblum, M.; and Goldstein, T. 2022. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13699–13708.

Tramèr, F.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*.

Yang, K.; Wang, R.; and Wang, L. 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 776–782.