ModelGiF: Gradient Fields for Model Functional Distance

Jie Song, Zhengqi Xu, Sai Wu, Gang Chen, and Mingli Song*
Zhejiang Univerisy
{sjie,xuzhengqi,wusai,cg,brooksong}@zju.edu.cn

Abstract

*The last decade has witnessed the success of deep learning and the surge of publicly released trained models, which necessitates the quantification of the model functional distance for various purposes. However, quantifying the model functional distance is always challenging due to the opacity in inner workings and the heterogeneity in architectures or tasks. Inspired by the concept of "field" in physics, in this work we introduce **Model Gradient Field** (abbr. ModelGiF) to extract homogeneous representations from the heterogeneous pre-trained models. Our main assumption underlying ModelGiF is that each pre-trained deep model uniquely determines a ModelGiF over the input space. The distance between models can thus be measured by the similarity between their ModelGiFs. We validate the effectiveness of the proposed ModelGiF with a suite of testbeds, including task relatedness estimation, intellectual property protection, and model unlearning verification. Experimental results demonstrate the versatility of the proposed ModelGiF on these tasks, with significantly superiority performance to state-of-the-art competitors. Codes are available at https://github.com/zju-vipa/modelgif.*

1. Introduction

The last decade has witnessed the great progress of deep learning in various fields, and a plethora of deep neural networks are developed and released publicly, with either their architectures and trained parameters (*e.g.*, Tensorflow Hub[1], Pytorch Hub[2]) for research, or the prediction API (*e.g.*, BigML, Amazon Machine Learning) as ML-as-a-Service (MLaaS) for commercial purposes. These off-the-shelf pre-trained models become extremely important resources for not only practitioners to solve their own problems, but also researchers to explore and exploit the huge potential underlying these pre-trained models.

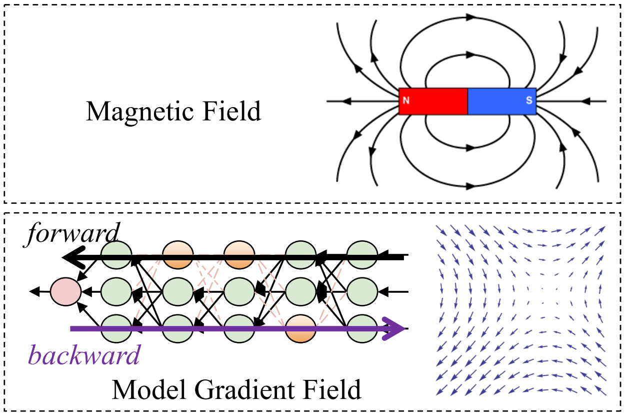


Figure 1. An illustrative diagram of the *magnetic field* and the proposed *model gradient field* (ModelGiF) defined on the input space.

With the surge of pre-trained models released, quantifying the model relationship emerges as an important question. The large-scale open-sourced deep models, heterogeneous in architectures and tasks and trained isolatedly or dependently, are related to each other in various manners. For example, a student model trained by knowledge distillation [19] should behave more similarly with the teacher than an independently trained identical model. Likewise, a fine-tuned model should be more closely related to its pre-trained model than the one trained from scratch. More generally, models trained in isolation on heterogeneous tasks should inherit the intrinsic task relatedness [53] as task-specific features are extracted by these models. Broadly speaking, there exists a *model metric space* where models with similar functional behaviors are clustered together and dissimilar ones are far apart. The model functional distance between models, if left unresolved, leaves existing model repositories still simple unstructured collections of many isolated open-sourced models, hindering the exploitation of their great value as a whole.

Despite the ever-increasing number of publicly available pre-trained models, the study on the model functional distance, *i.e.*, structure of the model metric space, lags far behind. This phenomenon can be largely attributed to the great challenge of computing the functional distance between any deep models, where the barriers are three-folds: (1) *Hetero-*

*Corresponding author
[1] https://www.tensorflow.org/hub
[2] https://pytorch.org/hub/

geneity: deep models usually differ significantly in architectures. Even with the same architecture, models trained on different datasets or tasks can also behave quite differently; (2) *Opacity*: the opaque inner workings of deep models render computing the model similarity extremely difficult; (3) *Efficiency*: as the cost of computing pairwise distance grows quadratically with the number of models, the computation of the functional distance should be efficient.

A few prior works have been devoted to computing model distance, in either weight space [1, 24] or representation space [28, 12]. For example, Task2Vec [1] computes task or model representations based on estimates of the Fisher information matrix associated with the probe network parameters, which provides a fixed-dimensional embedding of the model agnostic to the task variations. However, Task2Vec assumes all probe models share the same architecture, which is highly restrictive in practice. ModelDiff [28] and representation similarity analysis (RSA) [12], on the other hand, adopt the similarity of representations on a small set of reference inputs as the model representations, and the model distance is thus calculated resorting to these model representations, which is shown effective for model reuse detection and transferability estimation. However, these methods only capture the point-wise behavior of the models at a small number of chosen reference points, which is limited in representation capacity and fails at harder tasks such as detecting model extraction [28]. Recently, DEPARA [43, 44] and ZEST [22] are proposed as pilot studies by applying explanation methods for computing model distance. However, they are validated on disjoint downstream tasks, and the performance on comprehensive tasks remains unclear. Moreover, these methods also use a small number of chosen reference points to extract the model representations, which makes them suffer from low representation capacity.

In this work, inspired by the concept of "field" in physics, we propose **Model Gradient Field** (as shown in Figure 1), abbreviated as ModelGiF, as the proxy to extract homogeneous representations from pre-trained heterogeneous models to derive their functionality distance. Specially, the proposed ModelGiF is defined on the input space, *i.e.*, every point in the ModelGiF denotes the gradient vector of the model output *w.r.t.* the input on the same point. The main assumption underlying ModelGiF is that each pre-trained deep model uniquely determines a ModelGiF over the input space. The functional distance between any two models can thus be measured by the similarity between their ModelGiFs. Unlike prior methods where the point-wise features are adopted for representing the model, ModelGiFs represents these models by their gradient on the whole input space, which makes it more capable of differentiating highly related models (*i.e.*, model extraction detection). Moreover, we provide theoretical insights into the proposed ModelGiFs for model functional distance, and make extensive discussions on different implementation details. We validate the effectiveness of the proposed ModelGiF with a suite of testbeds, including transferability estimation, intellectual property protection, and model unlearning verification. Experimental results demonstrate the versatility of the proposed ModelGiF on these tasks, with significantly superiority to state-of-the-art (SOTA) methods.

To sum up, we make the following contributions: (1) we propose the concept of "model gradient field", a novel method for quantifying the functionality similarity between pre-trained deep models; (2) we provide theoretical insights into the proposed ModelGiFs for model functional distance, and make extensive discussions on different implementation details; (3) extensive experiments demonstrate the effectiveness and the superiority of ModelGiF on various tasks, including transferability estimation, intellectual property protection, and model unlearning verification.

2. Related Work

This section briefly reviews some related topics to model functional distance, including task relatedness estimation, intellectual property protection, and model unlearning.

Task Relatedness Estimation. Recent studies [53, 43, 44, 5, 51, 3, 6, 26] reveals the existence of task relatedness or task structure among visual tasks. It is the concept underlying transfer learning and provides a principled way to seamlessly reuse supervision among related tasks or solve many tasks in one system [53]. Existing works to obtain task relatedness can be roughly divided into empirical and analytical approaches. Taskonomy [53] is the most representative work of empirical methods. It proposes a fully computational approach to obtain task relatedness by exhaustively computing the actual transfer learning performance, which is thus usually taken as the ground truth for evaluating other estimators. Despite the indisputable results, the computational cost of empirical methods is extremely high. On the other hand, analytical methods, *e.g.*,, DEPARA [44, 43], Task2Vec [1] and RSA [12], try to estimate the task relatedness without conducting the actual transfer learning. Analytical methods generally significantly reduce the computation overhead, but can be highly restrictive in model architectures or estimation performance. The proposed ModelGiF relaxes these restrictions and achieves task relatedness more consistent to taskonomy than existing approaches.

Intellectual Property Protection. Since training deep models usually consumes expensive data collection and large computation resources, the trained models constitute valuable *intellectual property (IP)* that should be protected in cases where reusing is not allowed without authorization. However, model IP can be infringed in a variety of forms, such as fine-tuning [2, 48], pruning [29, 33], transfer learning [39, 49, 50] and model extraction [34, 47, 25, 21], which poses great challenge to IP protection. Existing IP protection approaches can be roughly categorized into *watermarking* [23, 48, 13, 54, 2] and *fingerprinting* [30, 8, 28, 17]. Watermarking methods usually leverage weight regularization [48, 13] to put secret watermark in the model parameters or train models on a triggered set to leave backdoor [2, 54] in them. While being able to provide exact ownership verification, these techniques are invasive, *i.e.*, they need to tamper with the training process, which may affect the model utility or introduce new security risks into the model. Fingerprinting, on the contrary, extracts a unique identifier, *i.e.*, fingerprint, from the owner model to differentiate it from other models. Latest fingerprinting (*e.g.*, ModelDiff [28], IPGuard [8], SAC [17]) on deep learning models, though being non-invasive, also fall short when facing the diverse and ever-growing attack scenarios [10]. In this work, we propose ModelGiF as the homogeneous representations for heterogeneous pre-trained models, which can be naturally adopted as the fingerprint for IP protection and yields superior IP protection performance to existing approaches.

Model Unlearning Verification. Model unlearning aims to remove the effects of data points from the trained model, which has been attracting increasing attentions in recent years due to legislation and privacy ethics. Cao *et al.* [9] are dedicated to making learning systems forget and present an unlearning approach capable of forgetting specific data. Graves *et al.* [16] propose an effective method to eliminate the influence of personal data from the model while maintaining the validity. Bourtoule *et al.* [7] particularly propose a model unlearning method to decrease the time it takes to retrain exactly. However, how to identify whether the impact of data has been eliminated is an essential but rarely studied problem. Recently ZEST [22] verifies the unlearning of data by comparing the Local Interpretable Model-Agnostic Explanations (LIME) [37]. In this work, we adopt the proposed ModelGiF for unlearning verifacation, which yields competitive performance in our experiments.

3. Methodology

3.1. Problem Setup

Assume there is a model repository consisting of N pre-trained models $\mathcal{M} = \{M_1, M_2, ..., M_N\}$. With mild assumptions, these models are defined on the same input space $\mathbb{R}^D$ ($D = WHC$, where W, H and C denotes the width, height and channel of the input space[3]), yet trained with data sampled from different data distributions $\mathcal{P} = \{p_1, p_2, ..., p_N\}$. Note that here we made no assumptions on the model architectures and the tasks, which means that these models can be different in architectures (*e.g.*, ResNet [18] and VGG [42]), and tasks (*e.g.*, visual classification [27], detection [36] or segmentation [11]). The goal in this work is to construct a model metric space where models with similar functional behaviors are clustered together and dissimilar ones are far apart, where the vital step is quantifying the functional distance between these models.

3.2. The Proposed ModelGiF

As aforementioned, quantifying the functional distance between pre-trained models is challenging due to the heterogeneity, opacity and efficiency issues. Our main idea is extracting homogeneous descriptors from these heterogeneous models to make them comparable. To this end, we proposed the concept ModelGiF, as shown in Figure 1, to derive the comparable identifier of the pre-trained models for quantifying model distance.

ModelGiF is inspired by the concept "field" in physics where a field is a mapping from space to some physical quantity, represented by a scalar, vector, or tensor, that has a value for each point in the space [32]. For example, the magnetic field describes the magnetic influence on moving electric charges, electric currents, and magnetic materials [14]. Likewise, we define ModelGiF as a mapping from the input space to the gradient space such that each point in the input space is assigned to a gradient vector. The formal definition is provided as follows.

Definition 3.1 (Model Gradient Field) *Let M be a deep model trained on the data which is sampled according to some distribution p from the data space $\mathcal{X}$, outputting a scalar*[4] *prediction in the label space $\mathcal{Y}$. A point $\mathbf{x}$ can be described as $\mathbf{x} = (x_1, x_2, ..., x_D)$. We define the model gradient field of M as the gradient of every possible input point in $\mathcal{X}$ w.r.t. the model output:*

$$\textsc{ModelGiF}(M) \triangleq \nabla_{\mathbf{x}} M, \tag{1}$$

which is a mapping from the input space $\mathbb{R}^D$ to the gradient space $\mathbb{G}^D$, $\textsc{ModelGiF} : \mathbb{R}^D \rightarrow \mathbb{G}^D$. Note that the ModelGiF is defined on the whole input space (which is shared by all the models) rather than some discretely points sampled from the data distribution p specific to the model M. As the architecture of M and the tasks (*i.e.*, the label space $\mathcal{Y}$) can be heterogeneous, the ModelGiF of a model can be seen as a projection of the model to the common input space.

3.3. Field Curves for ModelGiF

With the proposed ModelGiF, the model functional distance can be quantified by the ModelGiF similarity. However, as ModelGiF is defined on the whole input space that is usually extremely huge, how to extract its representation

[3] Without losing generality, we use vector instead of tensor for notation simplicity.

[4] For vector or tensor predictions, we simply take their l_2 norm as the scalar output.

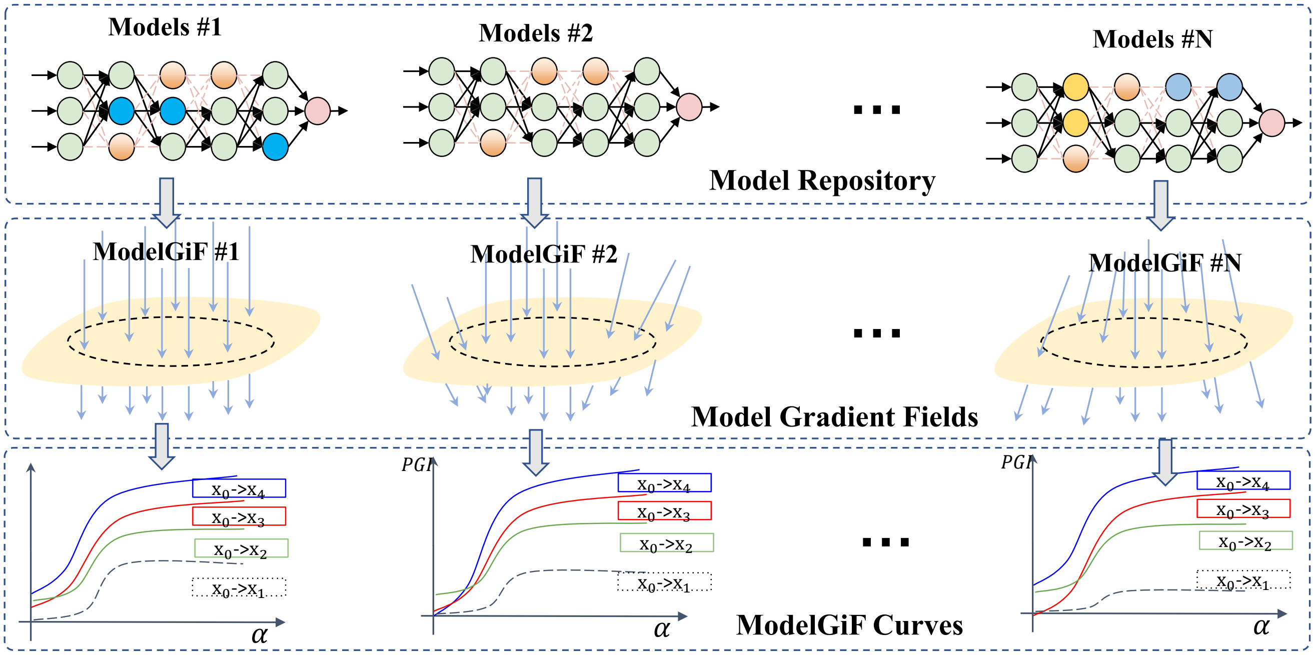


Figure 2. An illustrative diagram of the overall pipeline of obtaining ModelGiF curves. The more similar the ModelGiFs, the more similar the corresponding trained models. Note that ModelGiF Curves are in high-dimensional space in reality.

or signature becomes the vital step for quantifying model distance. In physics, a field is usually depicted by the *field curves* (as shown in Figure 1), *i.e.*, the integral curves for the field, and can be constructed by starting at a point and tracing a curve through space that follows the direction of the vector field, by making the field curve tangent to the field vector at each point [46]. Inspired by this, we propose *ModelGiF Curves* as the descriptors of ModelGiF to measure the model functional distance.

Definition 3.2 (Field Curves) *For a field F:* $\mathbb{R}^D \rightarrow \mathbb{G}^D$*, a curve* $\mathbf{x}(t) = \big(x_1(t), x_2(t), ..., x_D(t)\big)$ *is called a field curve of the field F if the following condition is satisfied: For all points* $P \in \mathbf{x}(t)$*, the tangent vector of the curve in the point* P *has the same direction as the vector* $F(P)$*:*

$$\frac{dx_1(t)}{dt} = F\big(\mathbf{x}(t)\big)_1, \frac{dx_2(t)}{dt} = F\big(\mathbf{x}(t)\big)_2, ... \tag{2}$$

The field curve can be described as the solution of the system of differential equations in Eqn. 2, and it plays an important role for both analysis and visualization of vector fields [46]. Unfortunately, as the ModelGiF is usually complicated for a pre-trained deep model, there is no closed solution. The field curves are in general not describable as parameterized curves, which hinders the comparisons between different ModelGiFs. To resolve this issue, we introduce the definition of ModelGiF curves as follows.

Definition 3.3 (ModelGiF Curves) *For a ModelGiF F:* $\mathbb{R}^D \rightarrow \mathbb{G}^D$*, a curve* $\mathbf{g}(t) = \big(g_1\left(t\right), g_2\left(t\right), ..., g_D\left(t\right)\big)$ *is called the ModelGiF curve of F if the following condition is satisfied:*

$$\frac{dg_1(t)}{dt} = F(\mathbf{x}(t))_1, \frac{dg_2(t)}{dt} = F(\mathbf{x}(t))_2, ... \tag{3}$$

Note that different from the definition of field curves, the ModelGiF curves are defined on the gradient space $\mathbb{G}^D$ instead of the input space $\mathbb{R}^D$.

Proposition 1 *For a ModelGiF F:* $\mathbb{R}^D \rightarrow \mathbb{G}^D$*, and two points* $\mathbf{x}_0$*,* $\mathbf{x}_1$ *in* $\mathbb{R}^D$*, the gradient integral along the straight line from* $\mathbf{x}_0$ *to* $\mathbf{x}_1$ *is a ModelGiF curve of F:*

$$\mathbf{g}(t) = \int_{\alpha=0}^{t} F\big(\mathbf{x}_0 + \alpha(\mathbf{x}_1 - \mathbf{x}_0)\big) d\alpha, \quad t \in [0, 1] \tag{4}$$

With the proposition, we can make comparisons between ModelGiFs by simply comparing their ModelGiF curves between some predefined pairs of points. Now we provide the detailed pipeline of quantifying the Model functional distance with the proposed ModelGiF.

3.4. ModelGiF for Model Distance

Provided with the trained deep models as described in Section 3.1, we compute the model distance with the proposed ModelGiF in two steps (in Figure 2): *1) Sampling reference points*, and *2) Computing the model distance*.

Sampling Reference Points. As described in Proposition 1, the first step before obtaining ModelGiF curves is determining the reference points (*i.e.*, $\mathbf{x}_1$[5] in Eqn. 4.), which determines the domain of these curves. Let K be the number of reference points to be sampled. Intuitively, these reference points should *representative*. However, as the models in $\mathcal{M}$ can be heterogeneous in architectures and tasks and trained on data sampled from different distributions, it is hard to determine a common set of reference points which are representative for all these models. In this work, we investigate three types of reference points as follows.

1) *Random samples* drawn from $\mathcal{P}$. Each data point is sampled in two stages: first randomly sampling the distribution p from $\mathcal{P}$ and then randomly sampling the reference point from p.

2) *Augmented samples* using CutMix [52]. Let $\mathbf{x}$ and $\mathbf{x}^*$ be two randomly sampled points in $\mathcal{P}$. CutMix generates a new sample $\tilde{\mathbf{x}}$ using the two samples by cutting off and pasting patches among them: $\tilde{\mathbf{x}} = \mathbf{m} \odot \mathbf{x} + (1 - \mathbf{m}) \odot \mathbf{x}^*$, where $\mathbf{m}$ represents the binary mask to combine images with different parts, and $\odot$ the element-wise multiplication.

3) *Adversarial samples* generated with PGD [31]. PGD attack is a multi-step variant of Fast Gradient Sign Method (FGSM) [15]: $\mathbf{x}_{t+1} = \prod\left(\mathbf{x}_t + \alpha \mathrm{SGN}(\nabla \mathcal{J}(\mathbf{x}_t))\right)$, where $\prod$ denotes the projection to the allowed space.

The performance of these types of reference points are discussed in Section 4.1. We also provide the sensitivity analysis of the number of reference points K, which demonstrates that the performance of ModelGiF quickly becomes superior to existing methods as K increases.

Computing the model distance. Let $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, , \mathbf{x}_K\}$ be the reference points obtained from the last step. For the i-th model M_i, we can get K ModelGiF curves by substituting each point in $\mathcal{S}$ for the $\mathbf{x}_1$ in Eqn. 4. These curves are used as the identifier of the ModelGiF, and thus serve as the representations of the trained deep model. There are several distance metrics to measure the similarity between curves, *e.g.*, Hausdorff distance and Fréchet distance. As the method for curve distance is not our contribution in this work, we simply adopt the integrated point-wise cosine distance to validate the performance of the proposed method:

$$d(M_i, M_j) = \sum_{k=1}^{K} \int_{t=0}^{1} \left(1 - \cos(\mathbf{g}^{i,k}(t), \mathbf{g}^{j,k}(t))\right) dt, \quad (5)$$

where $\mathbf{g}^{i,k}$ denotes the k-th curve of the i-th model in $\mathcal{M}$, and $\cdot$ denotes the inner product between $\mathbf{g}^{i,k}(t)$ and $\mathbf{g}^{i,k}(t)$. In practice, we approximate the integral with summation by sampling with some fixed intervals.

3.5. Theoretical Analysis of ModelGiF

From the definition of model distance in Eqn. 5, we can see that the defined distance can be seen as the summation of point-wise similarities over all the curves. Here we provide some theoretical insights into ModelGiF by dissecting the model-level distance into point-level distance.

Proposition 2 *For the point $\mathbf{g}(t_1)$ along the ModelGiF curves $\mathbf{g}(t)$ defined from $\mathbf{x}^*$ to $\mathbf{x}$, then*

$$\sum_i t_1(x_i - x_i^*) g_i(t_1) = M(\mathbf{x}) - M(\mathbf{x}^*), \quad (6)$$

where the left side is actually the summation over Integrated Gradients [45]. Eqn. 6 tells us that the every point in proposed ModelGiF curve is strongly related to the model prediction differences from this point and the baseline point, thus the proposed model distance is a powerful distance metric to quantifying the model function distance.

4. Experiments

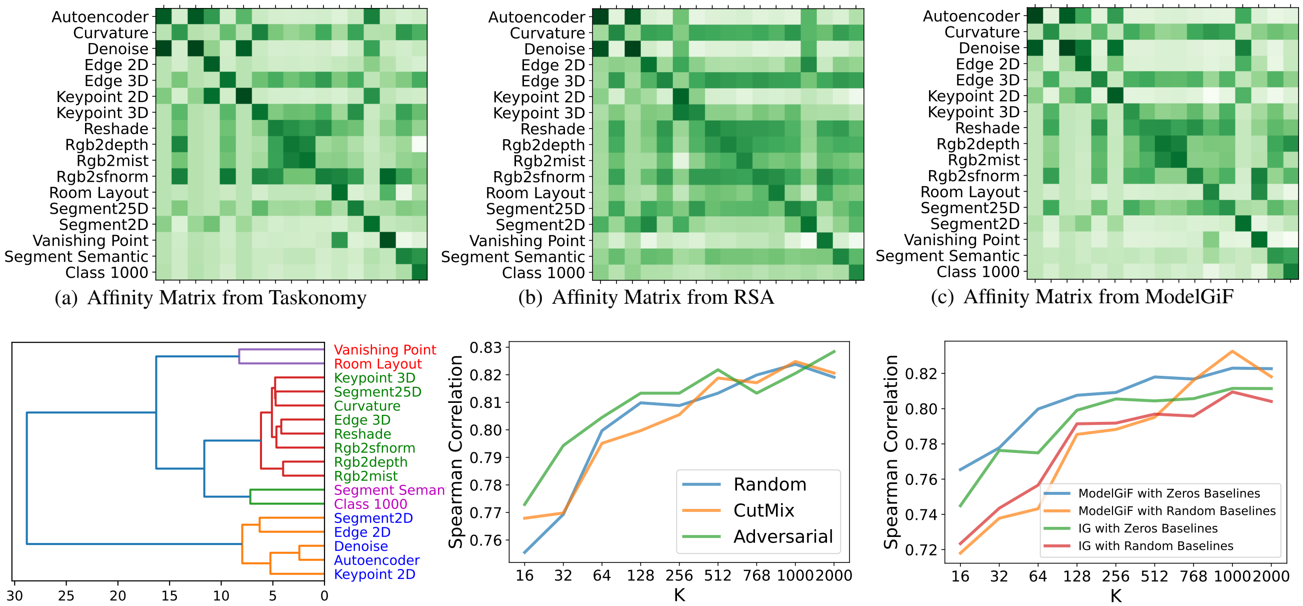
In this section, we verify the effectiveness of ModelGiF with a suite of testbeds, including task relatedness estimation, intellectual property protection and model unlearning verification. Details are provided as follows.

4.1. Application: Task Relatedness Estimation

Experimental Setup. We adopt 17 pre-trained heterogeneous models from Taskonomy [53] to compare their functionality similarity. These models are trained on various tasks (including Autoencoder, Curvature, Denoise, Edge 2D, Edge 3D, Keypoint 2D, Keypoint 3D, Reshade, Rgb2depth, Rgb2mist, Rgb2sfnorm, Room Layout, Segment25D, Segment2D, Vanishing Point, Segmentation, and Classification) and are different in architectures. Generally speaking, the architectures of these models follows an encoder-decoder scheme, in which the encoder is implemented by fully convolutional layers and the decoder varies according to the tasks. Please refer to [53] for more detailed information. In our experiments, only the encoders of these models are adopted for generating their ModelGiFs.

Competitors and Evaluation Metric. The proposed ModelGiF is compared with several prior approaches to task relatedness estimation, including RSA [12], AttributionMaps [43], and ZEST [22]. Note that AttributionMaps adopts saliency [41], DeepLIFT [40] and ϵ-LRP [4] to generate attributions for task relatedness estimation. We compare ModelGiF with all these variants to demonstrate its superiority in task relatedness prediction. As the affinity matrix from Taskonomy is obtained based on the actual transfer learning performance, it is used as the ground truth of the affinity among these tasks. In this experiment, we will

[5] Here we simply set $\mathbf{x}_0$ to the zero vector. Other settings do not yield significantly superior performance in our experiments.



(a) Affinity Matrix from Taskonomy (b) Affinity Matrix from RSA (c) Affinity Matrix from ModelGiF

(d) Similarity Tree from ModelGiF (e) Different reference points (f) Different implementations

Figure 3. Results of task related estimation: (a) affinity matrix from Taskonomy; (b) affinity matrix from RSA; (c) affinity matrix from ModelGiF; (d) the similarity tree derived from ModelGiF; (e) performance with varying K of different reference points; (e) performance with varying K of different implementations.

Method	**Spearman's Correlation**
Zest [22]	0.359
AttributionMaps$_{\text{SALIENCY}}$ [43]	0.619
AttributionMaps$_{\text{DEEPLIFT}}$ [43]	0.685
AttributionMaps$_{\epsilon\text{-LRP}}$ [43]	0.682
RSA [12]	0.777
ModelGiF$_{\text{RANDOM}}$	0.834
ModelGiF$_{\text{AUGMENT}}$	0.835
ModelGiF$_{\text{ADVERSARIAL}}$	0.830

Table 1. Comparison between the proposed ModelGiF and existing methods. Here $1,000$ reference points are sampled in ModelGiF.

verify that the functionality similarity obtained by ModelGiF positively correlates with that from taskonomy. In order to quantify the similarity between the affinity matrix representing functionality similarity obtained by ModelGiF and the affinity matrix obtained by Taskonomy, we use Spearman's correlation as the metric for evaluation.

Results. We first test the proposed ModelGiF with by randomly sampling $1,000$ reference points. The visualization of the affinity matrix (as well as that from Taskonomy [53] and RSA [12]) and the similarity tree are provided in Figure 3. This similarity tree is constructed by agglomerative hierarchical clustering based on the affinity matrix. It can be seen that the affinity matrix from the proposed ModelGiF is visually highly similar to that from Taskonomy in most regions. The similarity tree derived from ModelGiF perfectly matches the results from taskonomy where 3D (in green font) tasks, 2D (in blue font), geometric (in red font) tasks, and semantic (in purple font) tasks cluster into corresponding groups as expected.

Table 1 provide a quantitative comparison between the proposed ModelGiF with existing works in Spearman's correlation. It can be easily seen that the proposed ModelGiF yields significantly superior performance to existing methods, improving the SOTA Spearman's correlation from 0.777 to 0.835. Furthermore, all the three types of reference points produce Spearman's correlation more than 0.83, which implies that the proposed method is quite robust to the choice of the reference points. To make a more comprehensive study, we test ModelGiF with varying number of reference points. The correlation curves are depicted in Figure 3e, where different types of reference points are compared. It can be seen that as the number of reference points increases, the correlation steadily grows. Surprisingly, with only 16 points, ModelGiF already achieves comparable performance to RSA [12] in Spearman's correlation. It is attractive as the computation overhead of the proposed method grows linearly with the the number of reference points. The results indicate that we can safely reduce the computation cost by decreasing the number of reference points, and we can also strive for higher performance by adding more reference points, which makes ModelGiF flexible and applicable in a variety of scenarios. Another conclusion we can draw from Figure 3e is that the adversarial samples generally yields superior performance to other ref-

	CIFAR					tiny-ImageNet				
Attack	IPGuard [8]	CAE [30]	EWE [23]	SAC [17]	ModelGiF (Ours)	IPGuard [8]	CAE [30]	EWE [23]	SAC [17]	ModelGiF (Ours)
Finetune-A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.48	1.00	1.00
Finetune-L	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Pruning	1.00	0.95	0.87	1.00	1.00	1.00	1.00	0.58	1.00	1.00
Extract-L	0.81	0.83	0.97	1.00	1.00	0.97	1.00	1.00	1.00	1.00
Extract-P	0.81	0.90	0.97	1.00	1.00	0.97	1.00	1.00	1.00	1.00
Extract-Adv	0.54	0.52	0.91	0.92	1.00	0.65	0.78	1.00	0.91	1.00
Transfer-10C	1.00	1.00	1.00	1.00	1.00	N/A	N/A	N/A	N/A	N/A
Transfer-A	–	–	–	1.00	1.00	–	–	–	1.00	1.00
Transfer-L	–	–	–	1.00	1.00	–	–	–	1.00	1.00

Table 2. Comparison between the proposed ModelGiF and existing methods for IP proctection. The performance is measured in AUC value under the AUC-ROC curve. "–" represent the IP protection method can not detect this kind of attack. "N/A" denotes not applicable.

erence points, which gives implications of future work to strive for higher performance of task relatedness estimation.

In Figure 3f, we make comparisons between some different implementations of ModelGiF. "ModelGiF with random baselines" denotes $\mathbf{x}_0$ in Eqn. 4 is fixed to be zero, and "ModelGiF with random baselines" denotes $\mathbf{x}_0$ is randomly sampled. We also include Integrated Gradient (IG, with zero as the baseline) and IG with random baselines for comparisons. Albeit bearing some similarity with IG, it can be seen that ModelGiF with zero baselines significantly outperforms these baselines in most cases. ModelGiF with random baselines, however, is more promising when the number of reference points becomes sufficiently larger.

4.2. Application: Intellectual Property Protection

We evaluate the performance of ModelGiF for IP protection against different model stealing attacks. To make fair comparisons with SOTA methods, we follow the experimental settings of [17] to conduct our experiments.

Experimental Setup. Five categories of stealing attacks are considered here to test the protection performance. including finetuning, pruning, transfer learning, model extraction and adversarial model extraction. For finetuning, Finetune-L denotes fine-tuning only the last layer and leaves the other layers unchanged. Finetune-A fine-tunes all the layers in the model. For transfer learning, the CIFAR10 model is transferred to CIFAR10-C and CIFAR100. The tiny-ImageNet model (trained with the front 100 labels in Tiny-ImageNet) is transferred to the 100 labels left behind in tiny-ImageNet dataset. In model extraction, the victim model can be extracted in two manners: probability-based model extraction (Extract-P), and label-based model extraction (Extract-L). However, the attacker can evade the detection by applying adversarial training after the label-based model extraction [30]. In our experiment, adversarial model extraction (Extract-adv) adopts the predicted label to evade the detection by adversarial training.

Models and Competitors. Different IP protection methods are evaluated on most of the common model architectures, including VGG [42], ResNet [18], DenseNet [20] and MobielNet [38]. To demonstrate the superiority of the proposed ModelGiF, we make comparisons with SOTA IP protection methods, including IPGuard [8], CAE [30], EWE [23] and SAC [17]. IPGuard and CAE utilizes the transferability of adversarial examples and test the attack success rate of these adversarial examples on the suspect models. A model will be recognized as a stolen model if its attack success rate is larger than a threshold. SAC propose to leverage the pairwise relationship between samples as the model fingerprint. EWE, on the contrary, trains the source model on backdoor data and leaves the watermark in the model. Please refer to [17] for more detailed experimental settings.

Results. To validate the effectiveness and the superiority of the proposed ModelGiF, we conduct experiments on different datasets for the defender and the attacker. We we leverage AUC-ROC curve and use AUC value between the fingerprinting scores of the irrelevant models and the stolen models to measure the fingerprinting effectiveness. Results are listed in Table 2. It can be seen that the proposed ModelGiF, although not tailored for IP protection, achieve superior performance in all the attack scenarios. The AUC value is 1.0 across all attacks (including the challenging "Extract-adv" attack), which means it perfectly recognize all the attacks to protect the IP, outperforming existing SOTA approaches like SAC [17]. We acknowledge the existing benchmark for IP protection methods is not sufficient large and challenging to provide thorough comparisons between IP methods. However, the superiority of ModelGiF to SOTA methods on this benchmark still provides us a strong confidence of the proposed method for quantifying model functional distance.

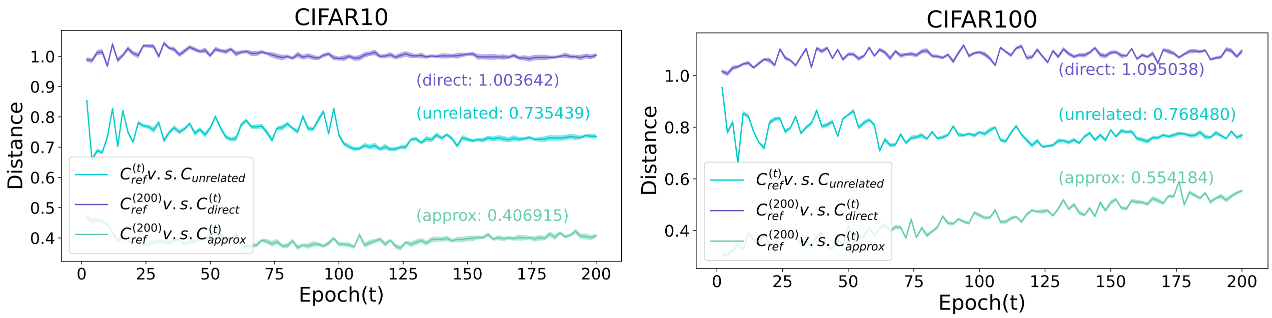


Figure 4. Cosine distances between the reference classifier C_{ref} and unrelated classifier $C_{unrelated}$, the directly unlearned classifier C_{direct} and the approximately unlearned classifier C_{approx}.

4.3. Application: Model Unlearning Verification

Machine unlearning studies how we can efficiently delete data points used to train models without retraining from scratch [7]. In this section, we demonstrate the effectiveness of the proposed ModelGiF for model unlearning verification, *i.e*., ModelGiF can distinguish the models trained with and without the data points that is requested to be unlearned.

Experimental Setup. Following the experimental settings of [22], we training four classifiers to evaluate ModelGiF for model unlearning verification. The first classifier is called the *reference classifier* $C_{ref}^{(t)}$, where t denotes that the classifier is obtained after t epochs training. The reference classifier serves as the original classifier trained on all the training data, including the data points which are requested to be deleted later. The second classifier is called the *unrelated classifier*, which is another classifier trained on all the training data, but from different initialization. The third classifier is called the *exactly unlearned classifier*, which is trained on the remaining data after removing the data points requested to be deleted. Note that the exactly unlearned classifier is trained from scratch and has never seen those data points which are requested to be removed, and it thus can be seen as the exactly unlearned classifier [7]. The last classifier is called *approximately unlearned classifier*, which is obtained by directly optimizing the original reference classifier to remove the knowledge learned from those data points requested to be unlearned [16]. We compare the ModelGiFs of the unrelated classifier, the exactly unlearned classifier and the approximately unlearned classifier to that of the reference classifier. The goal is to test whether the proposed ModelGiF can recognize the exactly unlearned classifier from the unrelated classifier.

Experimental Details and Results. Experiments are conducted on CIFAR10 and CIFAR100. On CIFAR10, all classifiers are implemented by ResNet20. On CIFAR100, all classifiers are implemented by ResNet50. We randomly sample 128 data points from the training data to be unlearned, and use these data as the reference points to compute the distance between ModelGIFs of these classifiers. Experimental results are provided in Figure 4. It can be seen that with the proposed ModelGiF, exactly unlearned classifier gets significantly higher distance with the reference classifier than the unrelated classifier. It implies that ModelGiF can also be used as a tool to verify the unlearning performance of existing unlearning methods. Another observation from Figure 4 is that the distance between the approximately unlearned classifier and the reference classifier is much lower than unrelated classifier, which implies that existing approximately unlearning methods can not delete the data from the model thoroughly. However, as the training for unlearning continues, the information can be gradually forgotten, as implied by the increasing distance shown in Figure 4. These results also accord with prior finding that continual learning easily lead to catastrophic forgetting problem [35], which validate the rationality of the results from the proposed ModelGiF.

5. Conclusion and Future Work

In this work, we propose ModelGiF to quantify model functional distance. The main assumption underlying ModelGiF is that each pre-trained deep model uniquely determines a ModelGiF over the input space. The distance between models can thus be measured by the similarity between their ModelGiFs. We apply the proposed ModelGiF to task relatedness estimation, intellectual property protection, and model unlearning verification. Experimental results demonstrate the versatility of the proposed ModelGiF on these tasks, with significantly superiority performance to state-of-the-art competitors. There are several directions for future work with the proposed ModelGiF. For example, exploring more scenarios where ModelGiF can be applied. Another interesting research direction is proposing more informative reference points to extract stronger representations of ModelGiF. Finally, how to further speed up the computation of the proposed method is also important to make it easier to use.

Acknowledgement. This work is funded by the National Key Research and Development Project (Grant No: 2022YFB2703100), National Natural Science Foundation of China (62106220, 61976186, U20B2066), and Ningbo Natural Science Foundation (2021J189).

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6430–6439, 2019. 2

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1615–1631, 2018. 2, 3

[3] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2015. 2

[4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015. 5

[5] 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. 2

[6] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning theory and kernel machines*, pages 567–580. Springer, 2003. 2

[7] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021. 3, 8

[8] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 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*, pages 14–25, 2021. 3, 7

[9] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015. 3

[10] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 824–841. IEEE, 2022. 3

[11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3

[12] 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. 2, 5, 6

[13] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32, 2019. 3

[14] Richard Phillips Feynman, Robert B. Leighton, Matthew Sands, and Everett M. Hafner. The feynman lectures on physics; vol. i. *American Journal of Physics*, 33:750–752, 1965. 3

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 5

[16] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021. 3, 8

[17] Jiyang Guan, Jian Liang, and Ran He. Are you stealing my model? sample correlation for fingerprinting deep neural networks. *arXiv preprint arXiv:2210.15427*, 2022. 3, 7

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 7

[19] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 1

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE con-*

ference on computer vision and pattern recognition, pages 4700–4708, 2017. 7

[21] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1345–1362, 2020. 3

[22] Hengrui Jia, Hongyu Chen, Jonas Guan, Ali Shahin Shamsabadi, and Nicolas Papernot. A zest of lime: Towards architecture-independent model distances. In *International Conference on Learning Representations*, 2021. 2, 3, 5, 6, 8

[23] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *USENIX Security Symposium*, pages 1937–1954, 2021. 3, 7

[24] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1039–1056. IEEE, 2021. 2

[25] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019. 3

[26] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004. 2

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 3

[28] Yuanchun Li, Ziqi Zhang, Bingyan Liu, Ziyue Yang, and Yunxin Liu. Modeldiff: testing-based dnn similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 139–151, 2021. 2, 3

[29] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, pages 273–294. Springer, 2018. 2

[30] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019. 3, 7

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 5

[32] Ernan McMullin. The origins of the field concept in physics. *Physics in Perspective*, 4:13–39, 2002. 3

[33] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019. 2

[34] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019. 3

[35] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks : the official journal of the International Neural Network Society*, 113:54–71, 2018. 8

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 3

[37] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 3

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 7

[39] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 3

[40] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016. 5

[41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 5

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 7

[43] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen, and Mingli Song. Deep model transferability from attribution maps. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 5, 6

[44] Jie Song, Yixin Chen, Jingwen Ye, Xinchao Wang, Chengchao Shen, Feng Mao, and Mingli Song. Depara: Deep attribution graph for deep knowledge transferability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3922–3930, 2020. 2

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017. 5

[46] Holger Theisel. *Vector field curvature and applications*. PhD thesis, University of Rostock, Germany, 1995. 4

[47] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016. 3

[48] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017. 2, 3

[49] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016. 3

[50] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *International Conference on Machine Learning*, pages 5085–5094. PMLR, 2018. 3

[51] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 2

[52] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 5

[53] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 1, 2, 5, 6

[54] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018. 3