Transferring Knowledge from Large Foundation Models to Small Downstream Models

Shikai Qiu [1 2 3] **Boran Han** [1] **Danielle C. Maddix** [1] **Shuai Zhang** [1] **Yuyang Wang** [1] **Andrew Gordon Wilson** [1 2]

Abstract

How do we transfer the relevant knowledge from ever larger foundation models into small, task-specific downstream models that can run at much lower costs? Standard transfer learning using pre-trained weights as the initialization transfers limited information and commits us to often massive pre-trained architectures. This procedure also precludes combining multiple pre-trained models that learn complementary information. To address these shortcomings, we introduce *Adaptive Feature Transfer* (AFT). Instead of transferring weights, AFT operates purely on features, thereby decoupling the choice of the pre-trained model from the smaller downstream model. Rather than indiscriminately compressing all pre-trained features, AFT adaptively transfers pre-trained features that are most useful for performing the downstream task, using a simple regularization that adds minimal overhead. Across multiple vision, language, and multi-modal datasets, AFT achieves significantly better downstream performance compared to alternatives with a similar computational cost. Furthermore, AFT reliably translates improvement in pre-trained models into improvement in downstream performance, even if the downstream model is over $50\times$ smaller, and can effectively transfer complementary information learned by multiple pre-trained models.

1. Introduction

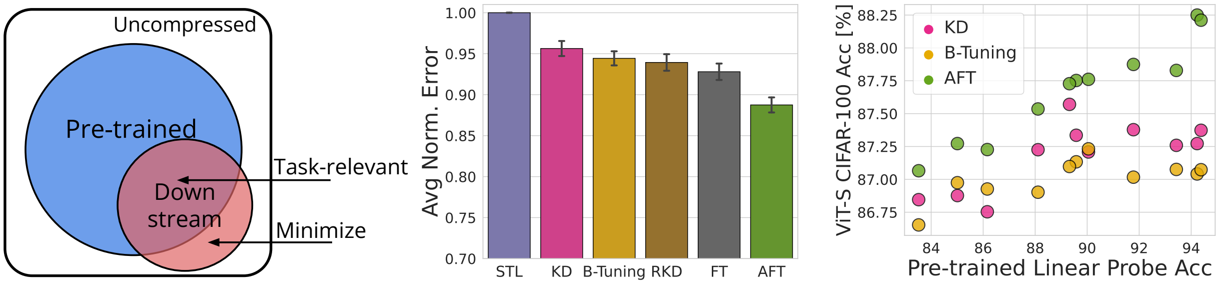
Despite the growing importance of transfer learning, it remains standard practice to simply start with some pre-trained weights as an initialization for fine-tuning on downstream data. This procedure only transfers generic and limited information and the computational burden of fine-tuning and deploying pre-trained models is quickly becoming prohibitive with increases in model size (Bommasani et al., 2021; Brown et al., 2020; Dosovitskiy et al., 2020; Zhai et al., 2022). Furthermore, this approach precludes transferring from multiple pre-trained models that learn complementary information due to different pre-training strategies, when a variety of distinctly pre-trained models have become available, especially in domains like computer vision (Oquab et al., 2023; Radford et al., 2021; Kolesnikov et al., 2020; Chen et al., 2020b).

In principle, however, this transfer from large foundation models to small downstream models should not only be possible but also natural, since the downstream models need not indiscriminately compress all knowledge learned by pre-training, but only inherit the task-revelant knowledge. Leveraging this insight, we propose *Adaptive Feature Transfer* (AFT), illustrated in Figure 1a, a simple, general, and efficient method to adaptively transfer task-relevant knowledge from a set of pre-trained models into a small downstream model, with negligible cost compared to standard training. Viewing pre-trained features as a compressed representation of the input containing highly relevant information for downstream predictions, AFT steers the downstream model to prioritize learning the task-relevant subset of pre-trained features over entirely new features representing information about the raw input but not preserved by pre-training. Crucially, recognizing not all pre-trained features are relevant for a specific downstream task, AFT discourages the downstream model from learning irrelevant features.

Across multiple vision, language, and multi-modal datasets, we show AFT delivers a substantial performance improvement when transferring from some of the strongest open-source vision and language foundation models, compared to alternatives with a similar computational cost: direct fine-tuning of the downstream model with standard transfer learning, B-Tuning (You et al., 2022), an efficient method multi-source and cross-architecture transfer learning, and knowledge distillation from the pre-trained to the downstream model (Hinton et al., 2015; Romero et al., 2014; Park et al., 2019; Kim et al., 2018). Moreover,

[1] AWS AI Labs, Santa Clara, CA, USA [2] Department of Computer Science, New York University, NYC, USA [3] Work done during an internship at AWS. Correspondence to: Shikai Qiu <sq2129@nyu.edu>, Boran Han <boranhan@amazon.com>, Andrew Gordon Wilson <andrewgw@cims.nyu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) Information diagram for AFT

(b) Aggregated performance

(c) Using stronger pre-trained models

Figure 1: **Adaptive Feature Transfer (AFT) transfers knowledge from large foundation models into small downstream models, improving downstream performance with minimal cost**. (**a**) AFT regularizes the downstream model to prioritize learning the task-relevant subset of pre-trained features $(\text{blue} \cap \text{red})$ over entirely new features $(\text{red} \setminus \text{blue})$. The blue region represents information in pre-trained features, red represents information in downstream features, and inside the square boundary represents all information in the raw, uncompressed input. (**b**) Over 6 vision datasets and 8 NLP datasets, AFT significantly outperforms standard transfer learning (STL), knowledge distillation (KD) (Hinton et al., 2015; Romero et al., 2014), including its more sophisticated variants relational knowledge distillation (RKD) (Park et al., 2019) and factor transfer (FT) (Kim et al., 2018), and B-Tuning (You et al., 2022). Error is normalized by STL error and averaged over datasets and downstream models, including ViT-S, MLP Mixer-B, ResNet-50, BERT-S, and DistillBERT. Error bars show standard errors across models and datasets. (**c**) AFT is the most effective at translating improvements in pre-trained models to improvements in downstream performance. See Section 4 for experiment details.

we find AFT is particularly effective at translating improvements in pre-trained models into improvements in downstream performance (Figure 1). Our code is available at `https://github.com/amazon-science/adaptive-feature-transfer`.

2. Related Work

We review the standard transfer learning approach and methods that enable efficient transfer learning from multiple sources and across architectures.

Transfer learning. Standard transfer learning (STL) proceeds by loading a pre-trained parameter vector as the initialization for parameters θ of a downstream model with the same architecture, followed by updating θ by minimizing the downstream loss $L(\theta)$, known as fine-tuning (Zhuang et al., 2019). This simple approach has enabled state-of-the-art performances on a wide range of vision (Dosovitskiy et al., 2020; Oquab et al., 2023; He et al., 2015) and language tasks (Devlin et al., 2018; Touvron et al., 2023).

Shwartz-Ziv et al. (2022) note that STL merely transfers an initialization, and that our knowledge of the source task should affect the shapes and locations of optima on the downstream task. To transfer additional information, Shwartz-Ziv et al. (2022) propose a Bayesian transfer learning approach by regularizing the downstream model with a Gaussian prior centered at the pre-trained weights, with a covariance matrix such that θ is allowed large variance in directions where pre-training loss increases slowly.

Efficient multi-source transfer learning. To transfer from multiple sources without fine-tuning many pre-trained models, Lee et al. (2019) propose to learn a classifier defined as a weighted combination of frozen pre-trained features, where the weights are derived from non-linear maximal correlation analysis. Chang et al. (2022) uses a mixture-of-experts model to combine complementary information across different models and datasets in material sciences. Shu et al. (2021) develops Zoo-Tuning to aggregate the parameters from multiple pre-trained models into a single downstream model, all assumed to have the same architecture. In addition, several works propose to rank and select in advance a subset of pre-trained models or features for transferring to a specific downstream task (You et al., 2022; Fumero et al., 2023; Deshpande et al., 2021), thus reducing the cost of exploration when a large number of pre-trained models are available. As these methods still reuse the pre-trained architecture for the downstream task, they are only useful for reducing the cost of training, but not the cost of deploying large pre-trained architectures. Moreover, methods such as Zoo-Tuning cannot be applied to transfer across architectures, limiting the choice of pre-trained models.

Cross-architecture transfer learning. B-Tuning (You et al., 2022) is a recently proposed method that enables cross-architecture transfer by regularizing the downstream model with a prior defined by the approximate posterior of a linear model conditioned on pre-trained features. Unlike the prior in Shwartz-Ziv et al. (2022), this prior is defined in function space rather than parameter space, and can therefore be used for downstream models of any architecture. On transferring from multiple pre-trained vision models, You et al. (2022) shows B-Tuning outperforms both knowledge distillation and Zoo-Tuning.

An alternative approach to cross-architecture transfer is knowledge distillation (KD) (Hinton et al., 2015). While the original KD trains the student to perform the same task as the teacher, feature-based KD can be applied to transfer the knowledge learned by a teacher pre-trained on a different but related task to a downstream student model, by training it to predict the teacher's features rather than logits (Romero et al., 2014; Heo et al., 2019a; Huang & Wang, 2017; Heo et al., 2019b; Gu et al., 2023; Yim et al., 2017; Ahn et al., 2019; You et al., 2022). In this approach, the student is usually trained to minimize a regression objective $\mathbb{E}_x\left[\|\phi_T(x) - V\phi_S(x)\|_2^2\right]$, where ϕ_S and ϕ_T denote the student and teacher features, and V is a learned transformation that can account for the difference in dimensionality and the arbitrariness of the choice of coordinates. Many works have proposed more sophisticated version of feature-based KD, such as relational knowledge distillation (RKD) (Park et al., 2019) that aims to capture the relation between the features of different inputs rather than their absolute values, and factor transfer (Kim et al., 2018), which trains the student to predict a compressed version of the teacher features learned through an autoencoder. Other works, such as Jang et al. (2019); Ji et al. (2021), focus on incorporating features from many intermediate layers.

Difference between AFT and prior works. As we shall explain in detail in Section 3, AFT is conceptually distinct from B-Tuning and KD, though they all use pre-trained features to regularize the downstream model. The main difference between our approach and B-Tuning is that 1) we regularize the downstream model's features rather than predictions, which allows more information to be transferred into the downstream model (features are often higher dimensional than the outputs), and 2) we learn the importance of each pre-trained feature during training on the downstream task rather than determining it ahead of time based purely on the posterior predictive mean of pre-trained models, which fails to take into account any property of the downstream model. In contrast to KD, AFT does not penalize the downstream model (student) from forgetting some of the pre-trained (teacher) features, and only penalizes learning extra features not extracted from pre-training.

3. Adaptive Feature Transfer

We now introduce Adaptive Feature Transfer (AFT), a method that adaptively transfers task-relevant knowledge from large foundation models to a small downstream model with negligible overhead compared to standard training.

3.1. An informative prior from pre-trained features

The core intuition behind AFT is that we want the downstream model to prefer making predictions based on information already present in the pre-trained features, as they are highly likely to contain useful knowledge for the downstream task, but without necessarily using all pre-trained features, since not all of them will be relevant to the downstream task. We now formalize this simple intuition mathematically by defining a prior for downstream learning. Let $\theta \in \mathbb{R}^P$ be the downstream model parameters, the random variable $X \in \mathbb{R}^{d_{\text{in}}}$ be the downstream inputs, $\Phi = \phi_\theta(X) \in \mathbb{R}^{d_\phi}$ be the features of the downstream model, $Y = W\Phi \in \mathbb{R}^{d_{\text{out}}}$ be the downstream model outputs, and $\Psi = \psi(X) \in \mathbb{R}^{d_\psi}$ be a list of frozen pre-trained features, formed by concatenating the last layer features from an arbitrary number of pre-trained models. To encourage the desired behavior, we define a prior that favors low mutual information between downstream features Φ and the input X conditioned on the pre-trianed features Ψ,

$$p(\theta) \propto \exp(-\beta I(\Phi; X|\Psi)), \tag{1}$$

where the $I(\Phi; X|\Psi)$ measures the amount of information about X encoded in downstream features Φ but not in the pre-trained features Ψ, visualized in Figure 1 as the area of red \ blue, and $\beta > 0$ controls the strength of this prior. The mutual information is given by

$$\begin{aligned} I(\Phi; X|\Psi) &= H(\Phi|\Psi) - H(\Phi|X, \Psi) & (2)\\ &= \mathbb{E}_{\Phi,\Psi}[-\log p(\Phi|\Psi)] + c & (3)\\ &\le \min_{\mu} \mathbb{E}_{\Phi,\Psi}[-\log q_\mu(\Phi|\Psi)] + c, & (4)\end{aligned}$$

where H denotes the conditional entropy. $H(\Phi|X, \Psi)$ is some constant c since Φ is deterministic given X and we used a variational distribution $q_\mu(\Phi|\Psi)$ with variational parameters μ to approximate the inaccessible conditional density $p(\Phi|\Psi)$ and thus bound the mutual information.

To train the downstream model, we seek the most likely parameters conditioned on the data under this prior, by minimizing the bound on the negative log posterior, equal to $L(\theta) + \beta R(\theta)$, where $L(\theta)$ is the unregularized loss (e.g. cross-entropy loss) and $R(\theta)$ is the bound on the mutual information given by

$$R(\theta) = \min_{\mu} \mathbb{E}_{\Phi,\Psi}[-\log q_\mu(\Phi|\Psi)], \tag{5}$$

where the expectation can only be estimated using training samples. The effect of optimizing this objective is to maximize the downstream data fit while minimizing the information in downstream features Φ that cannot be decoded from the pre-trained features Ψ via the map $q_\mu(\Phi|\Psi)$, after optimizing for variational parameters μ. We consider a simple Gaussian parameterization $q_\mu(\Phi|\Psi) = \mathcal{N}(\Phi|\mu\Psi, I)$, where $\mu : \mathbb{R}^{d_\psi} \to \mathbb{R}^{d_\phi}$ is an affine transformation, which leads to:

$$R(\theta) = \min_{\mu} \mathbb{E}_{\Phi,\Psi}\left[\|\Phi - \mu\Psi\|^2\right], \tag{6}$$

after ignoring some $\theta-$independent constants. Since the minimization over the offsets in the affine transformation is equivalent to subtracting the mean from both Φ and Ψ, we will henceforth assume that Φ and Ψ have been preprocessed to have zero-mean and assume $\mu \in \mathbb{R}^{d_\phi \times d_\psi}$ to be a linear transformation.

By comparison, the KD objective is equivalent to

$$R_{\mathrm{KD}}(\theta) = \min_{V} \mathbb{E}_{\Phi,\Psi}\left[\|V\Phi - \Psi\|^2\right], \tag{7}$$

with $V \in \mathbb{R}^{d_\psi \times d_\phi}$. The regularization we introduce moves the learnable transformation to act on the pre-trained features instead of the downstream features. This simple modification makes the objective more suitable for transfer learning. While minimizing the KD objective requires the downstream Φ features to contain all information needed to predict the pre-trained features Ψ, even if some are irrelevant or harmful to the downstream task, our objective $R(\theta)$ only requires the downstream features Φ to lie in the span of the pre-trained features Ψ, allowing Φ to encode only a subset of information in Ψ. With this simple but significant change to the knowledge distillation objective, we incentivize an adaptive transfer of pre-trained features to the downstream task. As we will show, this objective leads to significant performance gains for transfer learning with almost no additional cost and is particularly effective at translating improvements in pre-trained models to downstream performance.

3.2. Improving the objective using kernels

While conceptually straightforward, evaluating and minimizing the regularization $R(\theta)$ in Eq. 6 introduces both optimization and statistical challenges: 1) since evaluating $R(\theta)$ requires finding the optimal variational parameters μ, which changes every time we update θ, we want to simplify the optimization problem for μ to minimize its computational overhead, and 2) since we wish to estimate the true $R(\theta)$ whose exact value is given by an expectation over the true rather than empirical distribution of Φ and Ψ, we want to avoid over-fitting to the training data when optimizing for μ once we replace the expectation in Eq. 6 with its empirical estimate, especially since transfer learning often involves small downstream datasets.

We now show how to exploit a kernel formulation of the objective to further mitigate both challenges. Recall that the behavior of a linear model $f(\cdot) = w^\top \phi(\cdot)$ is completely characterized by its kernel $k_\Phi(x, x') = \phi(x)^\top \phi(x')$. From a kernel perspective, the existence of $\mu \in \mathbb{R}^{d_\phi \times d_\psi}$ such that $\Phi = \mu\Psi$ is equivalent to the existence of $\tilde{\mu} \in \mathbb{R}^{d_\phi \times d_\psi}$ such that $k_\Phi = k_{\tilde{\mu}\Psi}$. Therefore, we replace the ℓ_2 distance between the features with a distance between their kernel functions,

$$R_{\mathrm{AFT}}(\theta) = \min_{\mu} \sqrt{\mathbb{E}\left[\left(k_\Phi(X, X') - k_{\mu\Psi}(X, X')\right)^2\right]}, \tag{8}$$

where X and X' are drawn from the input distribution. As with the previous objective in Eq. 6, this objective achieves a minimum value of 0 if and only if each $\phi_i(\cdot), i = 1, ..., d_\phi$, is in the span of $\{\psi_i(\cdot)\}_{i=1}^{d_\psi}$. However, the kernel formulation has the key advantage that part of the optimization problem over μ is done automatically since the kernel is invariant under any orthogonal transformation of the features, implying that we only need to optimize μ up to an orthogonal transformation, significantly reducing the complexity of the inner optimization. This reduction of complexity simply reflects the fact there is no substantive difference between two models whose features only differ by an orthogonal transformation, e.g. a permutation or rotation of the feature dimensions.

To prevent over-fitting the variational parameters μ to the empirical distribution of the features, we parameterize μ as a diagonal matrix $\mathrm{diag}(\sigma(s))$, i.e. $\mu_{ii} = \sigma(s_i)$, where σ is the sigmoid function and s is a d_ψ-dimensional vector. Doing so greatly reduces the number of variational parameters to optimize, while retaining the ability for the model to weigh each dimension of the pre-trained features differently. Note that choosing a diagonal μ is always admissible in the kernel formulation, which does not require the features to have the same dimensions. Furthermore, due to the invariance of the kernel under orthogonal transformations, we are effectively searching over all $\mu' = U\mu = U\mathrm{diag}(s) \in \mathbb{R}^{d_\psi \times d_\psi}$, where $U \in \mathbb{R}^{d_\psi \times d_\psi}$ is any orthogonal matrix, without actually optimizing the dense matrix U which has significantly more parameters than μ. Finally, we normalize the features to have unit ℓ_2 norm before computing the respective kernels, i.e., $k_\Phi(x, x') := \phi(x)^\top \phi(x') / \|\phi(x)\| \|\phi(x')\|$, to reduce the variance in the kernel entries.

In Section 5.3, we compare AFT with its other variants and show that both using the kernel formulation and learning a diagonal μ are essential to its performance (Section 5.2). We also verify that the learned μ indeed places higher weights on more informative features (Figure 6c), allowing AFT to achieve robust performance even when a significant fraction of the pre-trained features is noise (Figure 6b).

Stochastic kernel distance estimation. For an efficient implementation, we estimate the kernel distance $\sqrt{\mathbb{E}\left[\left(k_\Phi(X, X') - k_{\mu\Psi}(X, X')\right)^2\right]}$ with a mini-batch estimate $\sqrt{\frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B}\left(k_\Phi(x_i, x_j) - k_{\mu\Phi}(x_i, x_j)\right)^2} =$

Algorithm 1 Adaptive Feature Transfer (AFT)

```
Require: Pre-computed pre-trained features, downstream data, downstream model $f_\theta = W \circ \phi_\theta$,
    downstream loss function $L$, batch size $B$, learning rates $(\eta_1, \eta_2)$, regularization coefficient $\beta$
for each mini-batch $X_{\text{batch}} \in \mathbb{R}^{B \times d_{\text{in}}}, Y_{\text{batch}} \in \mathbb{R}^{B \times d_{\text{out}}}, \Psi_{\text{batch}} \in \mathbb{R}^{B \times d_\psi}$ do
    Compute features $\Phi_{\text{batch}} = \phi_\theta(X_{\text{batch}}) \in \mathbb{R}^{B \times d_\phi}$ and outputs $\hat{Y}_{\text{batch}} = \Phi_{\text{batch}} W^\top$
    Scale pre-trained features $\Psi_{\text{batch}} \leftarrow \Psi_{\text{batch}} \mu^\top$
    Subtract the mini-batch mean from $\Phi_{\text{batch}}$ and $\Psi_{\text{batch}}$ and normalize each row
    Compute $B \times B$ mini-batch kernels $K^{\Phi}_{\text{batch}} = \Phi_{\text{batch}} \Phi^\top_{\text{batch}}, K^{\mu\Psi}_{\text{batch}} = \Psi_{\text{batch}} \Psi^\top_{\text{batch}}$
    Compute mini-batch loss $\hat{L}(\theta) = L(\theta, Y_{\text{batch}}, \hat{Y}_{\text{batch}})$ and the kernel distance estimate:
        $\hat{\delta}(\theta, \mu) = \frac{1}{B} \left\| K^{\Phi}_{\text{batch}} - K^{\mu\Psi}_{\text{batch}} \right\|_F$
    Update $\theta$ and $\mu$:
        $\theta \leftarrow \theta - \eta_1 \nabla_\theta \left( \hat{L}(\theta) + \beta \hat{\delta}(\theta, \mu) \right), \quad \mu \leftarrow \mu - \eta_2 \nabla_\mu \hat{\delta}(\theta, \mu)$
end for
```

$\frac{1}{B} \left\| K^{\Phi}_{\text{batch}} - K^{\mu\Psi}_{\text{batch}} \right\|_F$, where K^{Φ}_{batch} and $K^{\mu\Psi}_{\text{batch}}$ are kernel matrices evaluated on a batch of B inputs. We then perform gradient-descent over (θ, μ) jointly. Algorithm 1 details the training procedure, simplifying the update expression assuming SGD.

Negligible training overhead. We compute and cache the pre-trained features on the training set once and simply retrieve them during training without spending additional time to compute them. Table 1 compares the runtime on an NVIDIA A100 GPU for training ViT-S/16 (22M parameters) for one epoch on CIFAR-100 using STL and AFT, where AFT uses pre-trained features from OpenCLIP ViT-L/14 (303M parameters) (Cherti et al., 2023). As expected, the overhead of retrieving pre-computed features and computing the kernel distance is negligible compared to standard training. Pre-computing the features incurs only a one-time cost, which takes about 9 minutes for OpenCLIP ViT-L/14 on the CIFAR-100 training set.

Table 1: AFT has negligible training overhead compared to standard transfer learning. We report 1 epoch training time on CIFAR-100 for ViT-S/16 with STL and AFT, where AFT transfers features from OpenCLIP ViT-L/14.

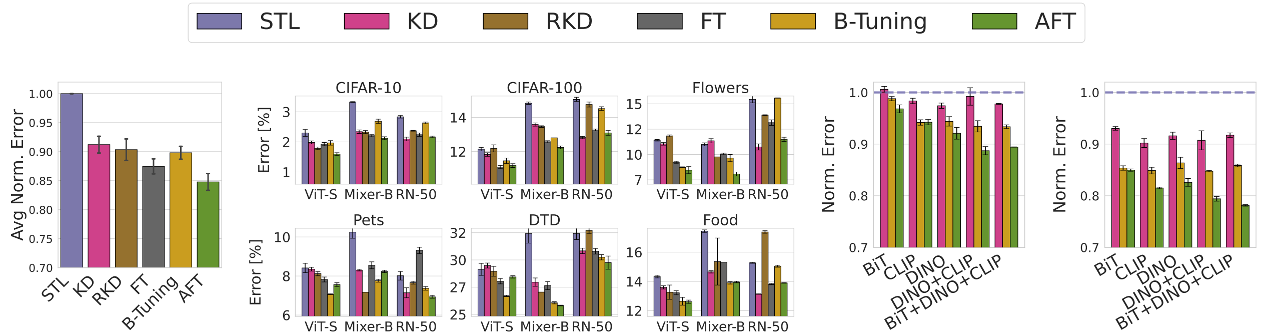
Method	Pre-trained (ψ)	Downstream (ϕ)	Time (min)
STL	N/A	ViT-S/16	1.74
AFT	OpenCLIP ViT-L/14	ViT-S/16	1.77

4. Experiments

We evaluate the proposed method **Adaptive Feature Transfer (AFT)** across a variety of vision, language, and multi-modal datasets. To probe the effectiveness of the method in the most impactful and practically relevant scenario, we transfer from some of the largest and strongest open-source pre-trained vision and language models such as ViT-G/14 trained with DINOv2 (Oquab et al., 2023) and LLaMA-2 (Touvron et al., 2023). For AFT, we start with a pre-trained version of the downstream architecture and optimize the training loss plus the regularization term in Eq. 8. We compare AFT against the following methods with comparable computational costs:

- **Standard Transfer Learning (STL)**. STL simply transfers an initialization from the pre-trained model for fine-tuning on the downstream task. This approach prevents the use of any additional pre-trained models that either differ in architecture or size from the downstream model. Therefore we transfer from a pre-trained version of the same downstream architecture with standard fine-tuning.
- **B-Tuning** (You et al., 2022). In addition to initializing with a pre-trained version of the downstream architecture, B-Tuning uses an approximate posterior predictive distribution of a linear model on top of the features from all other additional pre-trained models as a prior. This method demonstrated state-of-the-art performance when transferring from multiple pre-trained vision models up to ResNet-152 (He et al., 2015) size. Its effectiveness has yet to be tested for modern massively pre-trained vision foundation models such as Vision Transformers (Dosovitskiy et al., 2020).
- **Knowledge distillation (KD)**. In addition to initializing with a pre-trained version of the downstream architecture, we optimize the feature-based KD objective, which trains the downstream model (student) to fit the pre-trained (teacher) features (Romero et al., 2014), with the objective given by Eq. 7. We also include two more sophisticated variants of KD, relational knowledge distillation (RKD) (Park et al., 2019), which aims to capture the relation between the features of different inputs rather than their absolute values, and factor transfer (Kim et al., 2018), which trains the student to predict a highly compressed version of the teacher features, where the compression is learned by training an unsupervised autoencoder on the teacher features.

All methods start with the same pre-trained initialization of the downstream architecture. AFT, B-Tuning, and KD additionally optimize their respective regularization objective weighted by a hyperparameter $\beta > 0$, which is tuned



(a) Aggregated error (b) Error across models and datasets (c) ViT-S (d) Mixer-B

Figure 2: **Evaluation on 6 vision datasets using ViT-S, MLP-Mixer-B, and ResNet-50 as downstream models**. (**a**) AFT achieves the lowest normalized error, averaged across all 6 datasets, 3 downstream models, and 3 seeds when transferring from DINOv2 ViT-G/14. The error is normalized by the STL error before averaging. Error bars show standard errors of the aggregated performance. (**b**) Breakdown of unnormalized error for each downstream model and dataset. Error bars show standard errors across 3 seeds. (**c**, **d**) On CIFAR-100, AFT further improves from combining multiple pre-trained models.

on the validation set. We will use the term "pre-trained models" to refer to models whose features ψ are used to define the regularization objectives, rather than being used as the initialization for the downstream model. We include full experiment details, including hyperparameters, in Appendix A. We report the mean and standard errors computed across 3 runs for each method.

4.1. Image Classification

Effective transfer from SOTA vision foundation models. We evaluate AFT's ability to transfer from state-of-the-art vision foundation models into commonly used downstream architectures, including ViT-S (Dosovitskiy et al., 2020), MLP-Mixer-B (Tolstikhin et al., 2021), and ResNet-50 (He et al., 2015). We initialize the downstream models with ImageNet-1K checkpoints for all methods. In Figure 2a and 2b, we show performance when transferring from DINOv2 ViT-G/14, the largest model in the DINOv2 family with over a billion parameters, on CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), Oxford Flowers-102 (Nilsback & Zisserman, 2008), Oxford-IIIT Pets (Parkhi et al., 2012), Describable Textures Dataset (DTD) (Cimpoi et al., 2014) and Food-101 (Bossard et al., 2014) datasets. We find AFT significantly boosts the performance of all three models, reducing the error by an average of over 15% relative to STL performance (Figure 2a), and outperforms alternatives in most cases. The main exception is ResNet-50, where KD tends to slightly outperform AFT.

Transfer from multiple pre-trained models In Figure 2c and 2d, we show the performance on CIFAR-100 when transferring from various vision foundation models, including BiT ResNet-101x3 (Kolesnikov et al., 2020) (denoted BiT), OpenCLIP ViT-G (Cherti et al., 2023; Radford et al., 2021) (denoted CLIP) and DINOv2 ViT-G/14 (Oquab et al., 2023) (denoted DINO). AFT significantly outperforms competing methods. Moreover, AFT consistently achieves the best performance by transferring from multiple pre-trained models such as DINO + CLIP or BIT + DINO + CLIP. This result shows AFT can effectively combine complementary features learned by these models due to different inductive biases, pre-training objectives, and pre-training data.

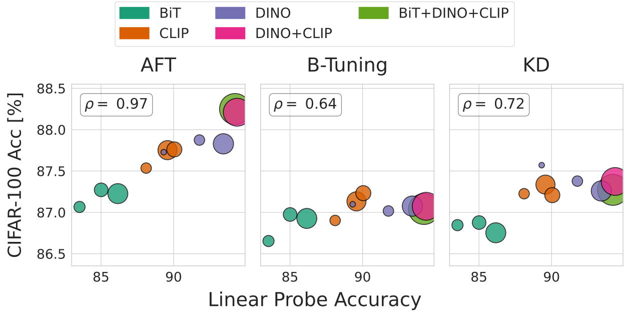


Figure 3: **CIFAR-100 downstream accuracy vs linear probe accuracy of pre-trained features, averaged across 3 downstream models**. AFT most effectively translates improvements in pre-trained models to improvements in downstream performance. Marker size is proportional to the number of parameters in the pre-trained models, ranging from 87 million to 2.7 billion.

Performance improves with stronger pre-trained models. With an effective method, we wish the downstream performance to consistently improve by transferring from stronger pre-trained models. A method that successfully transfers from large to small models at a particular scale may fail to translate further improvements in pre-trained models to improvements in downstream performance.

To test the scalability with respect to pre-trained model quality, we compare the downstream performance achieved by each method to the linear probe accuracy of the pre-trained features, i.e., the accuracy achieved by logistic regression on the pre-trained features. We use linear probe accuracy as it measures the amount of useful information we can extract from large pre-trained models on the downstream task without expensive fine-tuning, and is widely used as a metric to estimate the quality of pre-traiend representations as the models are scaled up (Radford et al., 2021; Oquab et al., 2023; Chen et al., 2020a; Dosovitskiy et al., 2020). Figure 3 shows AFT is significantly more effective than alternatives at translating improvements in pre-trained models to improvements in downstream performance, with the highest correlation (0.97) between the downstream accuracy and pre-trained linear probe accuracy. By comparison, other methods' performance saturates early and correlates less well with the linear probe accuracy, showing the unique scalability of AFT with respect to pre-trained model quality.

Inference time savings. Table 2 shows the inference time on CIFAR-100 test set using an NVIDIA A100 GPU for various ViT models. We have shown that AFT effectively transfers from pre-trained models as large as DINOv2 ViT-G/14 to ViT-S/16, which has 50× fewer parameters and 100× faster inference time.

While the linear probe accuracy of a sufficiently large pre-trained model can exceed the accuracy of AFT, the linear probe is only efficient to train (via logistic regression) but still expensive to deploy, as it requires inference with the original pre-trained model, and is therefore not a viable alternative to the methods considered here. For example, the linear probe accuracy of OpenCLIP ViT-L/14 roughly matches AFT accuracy when transferred to ViT-S/16 on CIFAR-100 (Figure 3), but OpenCLIP ViT-L/14 is 20× larger than ViT-S/16 and is 4.4× slower to run.

Table 2: Inference times on CIFAR-100 test set. Transferring from DINOv2 ViT-G/14 to ViT-S/16 reduces inference times by 100×.

Model	Params (M)	Inference time (min)
ViT-S/16	22	0.33
OpenCLIP ViT-L/14	303	1.45
DINOv2 ViT-G/14	1136	34.2

4.2. Natural Language Processing

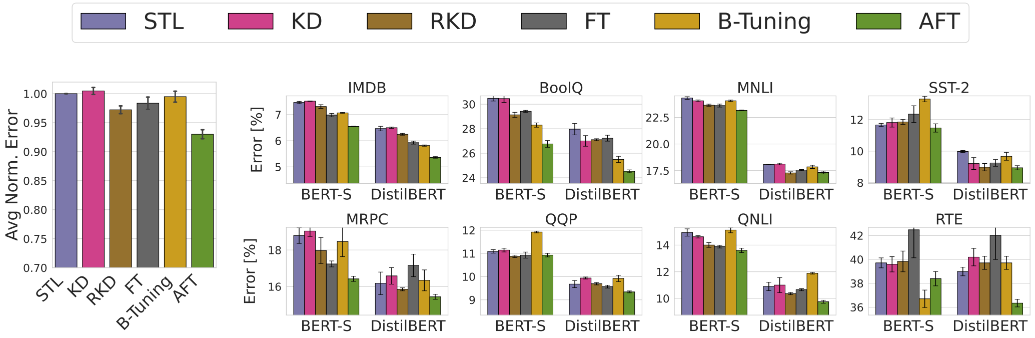
We explore transferring from larger open-source large language models, such as GPT-2 (Radford et al., 2019), Flan-T5 (Chung et al., 2022), and LLaMA 2 (Touvron et al., 2023), into much smaller language models, namely BERT Small (Devlin et al., 2018) and DistillBERT (Sanh et al., 2020). We follow common practices for extracting input-level features: using the embedding of the [CLS] token for BERT models and the decoder's embedding of the last token for GPT-2, Flan-T5, and LLaMA. In Appendix A.2, we provide details on input formatting and discuss memorization concerns.

We evaluate the performance of AFT and competing methods at transferring from Flan-T5 Large to BERT Small and DistillBERT on 8 datasets: Large Movie Review (IMDB)(Maas et al., 2011), BoolQ (Wang et al., 2019), MNLI (Williams et al., 2018), SST-2 (Socher et al., 2013), MRPC (Dolan & Brockett, 2005), QQP (Wang et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE (Wang et al., 2018). In Figures 4a and 4b, we show that AFT significantly outperforms the competing methods. As in the vision datasets, AFT most effectively translates improvements in pre-trained models to improvements in downstream performance. In Figure 5, we observe that using AFT with instruction-tuned pre-trained language models like Flan-T5 and LLaMA Chat leads to the best post-transfer performance, aligning with their superior zero-shot question answering capabilities (Chung et al., 2022).

In Figure 5, unlike in vision datasets, we find that combining multiple pre-trained models often does not improve their linear probe accuracy or the accuracy achieved by AFT, suggesting little complementary information is learned between these pre-trained language models. This may be due to the high similarity in pre-training datasets, objectives, and architectures among these transformer-based generative models, which are predominantly trained with next or masked token prediction on similar distributions of internet text.

4.3. Multi-modality

AFT's ability to efficiently transfer from multiple models makes it well-suited for multi-modal applications. In these settings, modality-specific sub-components, such as image and text encoders in CLIP (Radford et al., 2021), can benefit from transferring complementary features learned by pre-trained models in each modality. We demonstrate this on SNLI-VE (Xie et al., 2019; 2018), a visual entailment dataset where the goal is to determine if a text corresponds to an image. Using ResNet-50 CLIP as the downstream model, we construct a classifier $f_\theta(x_I, x_T) = W\phi(x_I, x_T)$ with features $\phi(x_I, x_T)$ given by the tensor product $\phi_I(x_I) \otimes \phi_T(x_T)$, representing pairwise interactions between image and text features. Table 3 shows that AFT improves CLIP's performance by simultaneously transferring from a ViT-L/14 trained with DINOv2 and LLaMA 13B.



(a) Aggregated error

(b) Error across models and datasets

Figure 4: **Evaluation on 8 language dataset using BERT Small and DistillBert as downstream models**. (**a**) AFT achieves the lowest normalized error, averaged across 6 datasets, 2 downstream models, and 3 seeds, when transferring from Flan-T5 Large. The error is normalized by the STL error before averaging. The error is normalized by the STL error before averaging. Error bars show standard errors of the aggregated performance. (**b**) Breakdown of unnormalized error for each downstream model and dataset. Error bars show standard errors across 3 seeds.

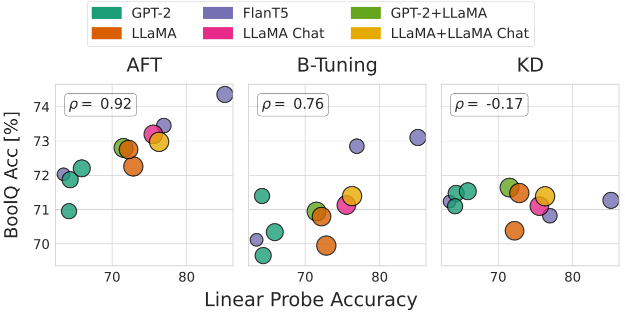


Figure 5: **BoolQ downstream accuracy v.s. linear probe accuracy of pre-trained features, averaged across two downstream models on BoolQ**. AFT most effectively translates improvements in pre-trained models to improvements in downstream performance. Marker size is proportional to the log of the number of parameters in the pre-trained models, ranging from 61 million to 14 billion.

Table 3: AFT improves CLIP's accuracy on SNLI-VE by transferring from DINOv2 and LLaMA 13B.

Method	STL	KD	AFT
SNLI-VE Acc.	$73.69_{\pm 0.28}$	$74.05_{\pm 0.05}$	$\mathbf{74.39}_{\pm \mathbf{0.18}}$

5. Analyzing Why AFT works

Having demonstrated AFT as a highly effective method, we now perform experiments to verify our understanding of why AFT works and reveal which design decisions are important.

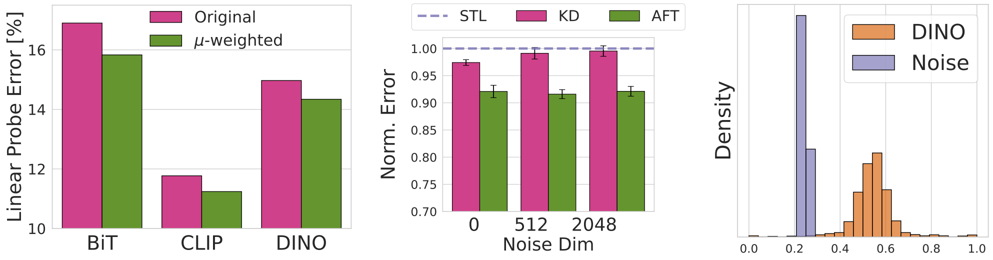
5.1. AFT upweights features that generalize better

If the learned weights μ in AFT indeed upweight the more informative features, then we expect a linear probe trained on the weighted features $\mu\psi$ should outperform one trained on the original features ψ. In Figure 6a, we show the linear probe error on CIFAR-100 with the original pre-trained features ψ from BiT 50x3, OpenCLIP ViT-G, or DINOv2 ViT-G, and on the weighted features $\mu\psi$, where the weights μ are learned by AFT when transferring to ViT-S. We find weighing the pre-trained features by the AFT weights improves the linear probe performance for all pre-trained models, showing that AFT indeed identifies and upweights pre-trained features that leads to better generalization on the downstream task.

5.2. AFT is robust to uninformative features

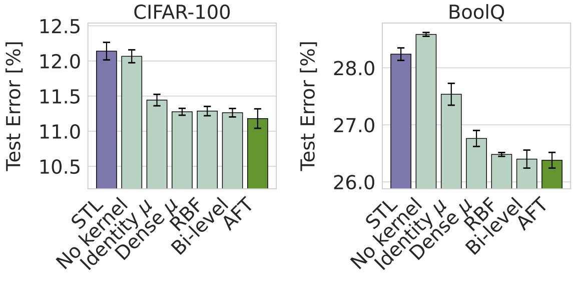
As the adaptive nature of AFT enables it to automatically downweight irrelevant features without any intervention, we expect it to perform well even when a large number of pre-trained features are completely uninformative of the downstream task. To test this hypothesis, we transfer from DINOv2 ViT-G/14 and a random noise model whose features are drawn from $\mathcal{N}(0, I_{d_{\text{noise}}})$, where $d_{\text{noise}} \in \{0, 512, 2048\}$ is its feature dimension, into ViT-S/16 on CIFAR-100.

Results in Figure 6b clearly illustrate the limitations of compression-based objectives like KD, whose performance quickly degrades to near STL level as we introduce the noise features, since the downstream model is trained to learn many useless features. By constrast, AFT performance is nearly unaffected by the presence of noise features. In Figure 6c, we show this robustness because the learned weights μ_i in AFT are much smaller for the noise features.



(a) AFT upweights informative features (b) Error v.s. d_{noise} (c) Distribution of μ_i

Figure 6: **Analysis of AFT's properties on CIFAR-100**. (**a**) Linear probe error is improved when applying the learned AFT weights μ to the pre-trained features, indicating that AFT effectively upweights informative features for the downstream task. (**b**) AFT's performance remains stable as an increasing number of noise features (d_{noise}) are appended to the pre-trained features, demonstrating its robustness to uninformative features. (**c**) The learned μ_i values effectively separate noise features from useful features, with noise features assigned much smaller weights.



(a) DINOv2 ViT-G/14 to ViT-S (b) Flan-T5 Large to BERT-S

Figure 7: **Ablation experiments**. Using the kernel and learning μ is essential for AFT's performance, whereas using an RBF kernel and bi-level optimization over (μ, θ) barely impacts performance. Making μ dense slightly hurts performance.

5.3. Ablation experiments

We investigate the impact of key design choices in AFT on its performance on CIFAR-100 and BoolQ. We compare AFT with four other variants where we a) do not use the kernel formulation and instead use the ℓ_2 objective in Eq. 6 (No kernel), b) disable the ability to learn μ and fix it to be the identity (Identity μ), c) Use a dense rather than diagonal μ (Dense μ), d) replace the linear kernel $k(x, x') = \phi(x)^\top \phi(x')$ with radial basis function (RBF) kernel $k(x, x') = \exp\left(-\|\phi(x) - \phi(x')\|^2\right)$ (RBF), and e) use bi-level optimization over θ and μ by performing 5 inner updates for μ per update of θ (Bi-level).

We find using the kernel formulation and learning the feature weights μ are essential to AFT's performance, while the use of alternative kernels such as the RBF kernel and bi-level optimization does not impact the performance in any significant way. Learning a dense rather than diagonal μ slightly hurts performance.

6. Discussion

Transfer learning — pre-training then fine-tuning — is becoming the mainstream paradigm for deploying deep learning models. However, the default approach to transfer learning remains surprisingly naive, transferring limited and generic information: simply use the pre-trained weights as an initialization for the downstream loss optimization. There is therefore a great need to develop transfer learning procedures more tailored to the task at hand.

Through AFT, we have shown that a simple, general, and computationally efficient approach exists for transferring knowledge from large models to small models. An important takeaway from AFT is that aligning what is transferred to the small downstream model with the specific downstream task is crucial for effective transfer learning, showing this large-to-small transfer fundamentally differs from just model compression. As future works uncover even more effective methods for large-to-small transfer, our fundamental understanding of transfer learning will further advance.

AFT offers a trade-off between reducing the cost of transfer learning and the potential performance improvements. AFT is inherently limited by the reduced representational capacity of small downstream models. This limitation can be mitigated by selecting more expressive downstream models, albeit at the cost of diminished savings in training and inference. Furthermore, the current formulation of AFT prioritizes simplicity, generality, and computational efficiency by restricting the transfer to only the last layer features. Expanding and optimizing the set of features transferred via AFT is an exciting direction for future work that may significantly further enhance performance.

Acknowledgements

We thank Micah Goldblum, Nate Gruver, and Daohan Lu for helpful discussions. This work is supported by NSF CAREER IIS-2145492, NSF CDS&E-MSS 2134216, NSF HDR-2118310, BigHat Biosciences, Capital One, and an Amazon Research Award.

Impact Statement

The goal of this work is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9163–9171, 2019.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Rees Chang, Yu-Xiong Wang, and Elif Ertekin. Towards overcoming data scarcity in materials science: unifying models and datasets with a mixture of experts framework. *npj Computational Materials*, 8(1):242, 2022.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020b.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2829, 2023.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv preprint arXiv:2102.00084*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Marco Fumero, Florian Wenzel, Luca Zancato, Alessandro Achille, Emanuele Rodolà, Stefano Soatto, Bernhard Schölkopf, and Francesco Locatello. Leveraging sparse and shared feature activations for disentangled representation learning. *arXiv preprint arXiv:2304.07939*, 2023.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1921–1930, 2019a. doi: 10.1109/ICCV.2019.00201.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019b. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33013779. URL https://doi.org/10.1609/aaai.v33i01.33013779.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer, 2017.

Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Learning what and where to transfer. In *International conference on machine learning*, pp. 3030–3039. PMLR, 2019.

Mingi Ji, Byeongho Heo, and Sungrae Park. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7945–7952, 2021.

Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. *Advances in neural information processing systems*, 31, 2018.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Joshua Lee, Prasanna Sattigeri, and Gregory Wornell. Learning new tricks from old dogs: Multi-source transfer learning from pre-trained networks. *Advances in neural information processing systems*, 32, 2019.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3967–3976, 2019.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Yang Shu, Zhi Kou, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. Zoo-tuning: Adaptive transfer from a zoo of models. In *International Conference on Machine Learning*, pp. 9626–9637. PMLR, 2021.

Ravid Shwartz-Ziv, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew Gordon Wilson. Pre-train your loss: Easy bayesian transfer learning with informative priors. *arXiv preprint arXiv:2205.10279*, 2022.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. Visual entailment task for visually-grounded language learning. *arXiv preprint arXiv:1811.10582*, 2018.

Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. Visual entailment: A novel task for fine-grained image understanding. *arXiv preprint arXiv:1901.06706*, 2019.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4133–4141, 2017.

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

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

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. URL http://arxiv.org/abs/1911.02685.

A. Experiment details

We tune the hyperparameter β for AFT, KD, and B-Tuning in all experiments by holding out 10% of the original training set and selecting the β value that yields the highest accuracy on this holdout set. Once the optimal β is determined, we train the models on the entire training set using this value. Our implementations of relational knowledge distillation (RKD) and B-Tuning are based on their original implementations, available at `https://github.com/lenscloth/RKD` and `https://github.com/thuml/LogME`, respectively. Following Park et al. (2019), we weigh the angle loss and the distance loss in RKD at a 2:1 ratio. For Factor Transfer, we replace the original CNN-based paraphraser and translator networks with MLPs, as we work with the last layer features, which lack spatial dimensions, instead of the intermediate CNN feature maps used in the original paper (Kim et al., 2018).

A.1. Vision experiments

We use the timm (Wightman, 2019) implementation for all vision models, their pre-trained checkpoints, and data preprocessing pipelines. We do not use data augmentation in any experiment.

We use the Adam optimizer in all experiments and train for 5000 steps (rounded up to whole epochs) with a batch size of 128 and a cosine lr decay schedule. We use a base learning rate of $1e-4$ for ViT-S/16 and MLP Mixer-B, and $1e-3$ for ResNet-50. We tune $\beta \in \{3, 10, 30\}$ for AFT, $\beta \in \{0.1, 1, 10, 100\}$ for KD, RKD, FT, and $\beta \in \{1, 1e2, 1e3, 1e4\}$ for B-Tuning. We use the Adam optimizer and a learning rate of $1e-2$ for updating the vector s parameterizing the diagonal elements of μ.

A.2. Language experiments

We use the Hugging Face implementation of all the language models. We use the Adam optimizer in all experiments and train for 5000 steps (rounded up to whole epochs) with a batch size of 64 and a cosine lr decay schedule. We use a base learning rate of $2e-5$ for both BERT Small and DistilBERT. We tune $\beta \in \{1, 3, 10\}$ for AFT, $\beta \in \{0.01, 0.1, 1, 10\}$ for KD, RKD, FT, and $\beta \in \{1, 1e2, 1e3, 1e4\}$ for B-Tuning. We use the Adam optimizer and a learning rate of $1e-2$ for updating the vector s parameterizing the diagonal elements of μ.

We format each example as follows before feeding it into the language model:

- IMDB (Maas et al., 2011): ⟨review⟩ Overall, the sentiment of my review is
- BoolQ (Wang et al., 2019): Question: ⟨question⟩\n Reference: ⟨passage⟩\n Answer:
- MNLI (Williams et al., 2018): Premise: ⟨premise⟩\n Hypothesis: ⟨hypothesis⟩\n Does the premise entail the hypothesis? Answer:
- SST-2 (Socher et al., 2013): Review: ”⟨sentence⟩”\n Sentiment:
- MRPC (Dolan & Brockett, 2005): Sentence 1: ⟨sentence1⟩\n Sentence 2: ⟨sentence2⟩\n Is Sentence 1 equivalent to Sentence 2? Answer:
- QQP (Wang et al., 2018): Question 1: ⟨question1⟩\n Question 2: ⟨question2⟩\n Are Question 1 and Question 2 equivalent? Answer:
- QNLI (Rajpurkar et al., 2016): Question: ⟨question⟩\n Sentence: ⟨sentence⟩\n Does the sentence answer the question? Answer:
- RTE (Wang et al., 2018): Sentence 1: ⟨sentence1⟩\n Sentence 2: ⟨sentence2⟩\n Does Sentence 1 entail Sentence 2? Answer:

On memorization concerns. Language models are pre-trained on internet-scale data, making it difficult to rule out the possibility that the benchmarks we evaluated on are not in their training set. However, this concern is irrelevant for us as our experiments aim only to compare each method’s effectiveness in transferring knowledge from the pre-trained models rather than establishing some absolute level of downstream performance on these benchmarks.

A.3. SNLI-VE experiments

We use the official OpenAI implementation of CLIP ResNet-50 (Radford et al., 2021). We use the Adam optimizer in all experiments and train for 1 epoch with a batch size of 64. We use a base learning rate of $1e-5$ for CLIP ResNet-50. We tune $\beta \in \{1, 3, 10\}$ for AFT, and $\beta \in \{0.01, 0.1, 1\}$ for KD. We use the Adam optimizer and a learning rate of $1e-2$ for updating the vector s parameterizing the diagonal elements of μ.

B. Extended results

Table 4: Unnormalized results for transfer to ViT-S/16 in Figure 2c.

Method	BiT	CLIP	DINO	DINO+CLIP	BiT+DINO+CLIP
KD	$87.79_{\pm 0.07}$	$88.06_{\pm 0.06}$	$88.17_{\pm 0.06}$	$87.96_{\pm 0.21}$	$88.13_{\pm 0.01}$
B-Tuning	$88.01_{\pm 0.05}$	$88.57_{\pm 0.06}$	$88.54_{\pm 0.11}$	$88.66_{\pm 0.13}$	$88.67_{\pm 0.04}$
AFT	$88.25_{\pm 0.09}$	$88.56_{\pm 0.06}$	$88.88_{\pm 0.06}$	$89.23_{\pm 0.10}$	$89.14_{\pm 0.00}$

Table 5: Unnormalized results for transfer to MLP-Mixer in Figure 2d.

Method	BiT	CLIP	DINO	DINO+CLIP	BiT+DINO+CLIP
KD	$86.21_{\pm 0.05}$	$86.63_{\pm 0.13}$	$86.42_{\pm 0.11}$	$86.55_{\pm 0.27}$	$86.40_{\pm 0.06}$
B-Tuning	$87.34_{\pm 0.06}$	$87.42_{\pm 0.10}$	$87.20_{\pm 0.16}$	$87.43_{\pm 0.02}$	$87.27_{\pm 0.04}$
AFT	$87.40_{\pm 0.03}$	$87.92_{\pm 0.02}$	$87.76_{\pm 0.11}$	$88.23_{\pm 0.07}$	$88.42_{\pm 0.02}$

Table 6: Unnormalized results for transfer to ResNet-50.

Method	BiT	CLIP	DINO	DINO+CLIP	BiT+DINO+CLIP
KD	$86.64_{\pm 0.15}$	$87.32_{\pm 0.16}$	$87.18_{\pm 0.10}$	$87.62_{\pm 0.07}$	$87.29_{\pm 0.14}$
B-Tuning	$85.57_{\pm 0.10}$	$85.42_{\pm 0.04}$	$85.49_{\pm \text{NaN}}$	$85.06_{\pm 0.05}$	$85.19_{\pm 0.11}$
AFT	$86.17_{\pm 0.05}$	$86.78_{\pm 0.07}$	$86.91_{\pm 0.09}$	$87.18_{\pm 0.04}$	$87.08_{\pm 0.10}$