GRADIENTS AS FEATURES FOR DEEP REPRESENTATION LEARNING

Fangzhou Mu, Yingyu Liang
Department of Computer Sciences
University of Wisconsin-Madison
{fmu, yliang,}@cs.wisc.edu

Yin Li
Departments of Biostatistics & Computer Sciences
University of Wisconsin-Madison
yin.li@wisc.edu

ABSTRACT

We address the challenging problem of deep representation learning—the efficient adaption of a pre-trained deep network to different tasks. Specifically, we propose to explore gradient-based features. These features are gradients of the model parameters with respect to a task-specific loss given an input sample. Our key innovation is the design of a linear model that incorporates both gradient and activation of the pre-trained network. We demonstrate that our model provides a local linear approximation to an underlying deep model, and discuss important theoretical insights. Moreover, we present an efficient algorithm for the training and inference of our model without computing the actual gradients. Our method is evaluated across a number of representation-learning tasks on several datasets and using different network architectures. Strong results are obtained in all settings, and are well-aligned with our theoretical insights[1].

1 INTRODUCTION

Despite tremendous success of deep learning, training deep neural networks requires a massive amount of labeled data and computing resources. The recent development of representation learning holds great promises for improving data efficiency of training, and enables an easy adaption to different tasks using the same feature representation. These features can be learned via either unsupervised learning using deep generative models (Kingma & Welling, 2014; Dumoulin et al., 2016), or self-supervised learning with "pretext" tasks and pseudo-labels (Noroozi & Favaro, 2016; Zhang et al., 2016; Gidaris et al., 2018), or transfer learning from another large-scale dataset (Yosinski et al., 2014; Oquab et al., 2014; Girshick et al., 2014). After learning, the per-sample activation of the network is considered as generic features. By leveraging these features, simple classifiers such as linear models can be learned for different tasks. However, given sufficient amount of training data, the performance of representation-learning methods lags behind fully supervised deep models.

As a step towards bridging this gap, we propose to make use of gradient-based features from a pre-trained network, i.e., gradients of the model parameters with respect to a task-specific loss given an input sample. Our key intuition is that these per-sample gradients contain task-relevant discriminative information. More importantly, we design a novel linear model that accounts for both gradient-based and activation-based features. The design of our linear model stems from the recent advances in the theoretical analysis of deep models. Specifically, our gradient-based features are inspired by the neural tangent kernel (NTK) (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019b), and adapt NTK in the setting of finite-width networks. Therefore, our model provides a local approximation of fine-tuning an underlying deep model, and the accuracy of the approximation is controlled by the semantic gap between the representation-learning and the target tasks. Finally, the structure of the gradient features and the linear model allows us to derive an efficient and scalable algorithm for training and inference.

To evaluate our method, we focus on visual representation learning in this paper, although our model can be easily modified for natural language or speech data. To this end, we consider a number of learning tasks in vision, including unsupervised, self-supervised and transfer learning. Our method

[1]Project webpage at http://pages.cs.wisc.edu/~fmu/gradfeat20

is evaluated across tasks, datasets and network architectures and compared against a set of baseline methods. We observe empirically that our model with the gradient features outperforms the traditional activation-based logistic regressor by a significant margin in all settings. Moreover, our model compares favorably against fine-tuning of network parameters.

Our main contributions are thus summarized as follows.

- We propose a novel representation-learning method. At the core of our method lies in a linear model that builds on gradients of model parameters as the feature representation.
- From a theoretical perspective, we claim that our linear model provides a local approximation of fine-tuning an underlying deep model. From a practical perspective, we devise an efficient and scalable algorithm for the training and inference of our method.
- We demonstrate strong results of our method across various representation-learning tasks, different network architectures and several datasets. Furthermore, these empirical results are well-aligned with our theoretical insight.

2 Related Work

Representation Learning. Learning good representation of data without expensive supervision remains a challenging problem. Representation learning using deep models has been recently explored. For example, different types of deep latent variable models (Kingma & Welling, 2014; Higgins et al., 2017; Berthelot et al., 2019; Dumoulin et al., 2016; Donahue et al., 2016; Dinh et al., 2017; Kingma & Dhariwal, 2018; Grathwohl et al., 2019) were considered for representation learning. These models were designed to fit to the distribution of data, yet their intermediate responses were found useful for discriminative tasks. Another example is self-supervised learning. This paradigm seeks to learn from a discriminative pretext task whose supervision comes almost for free. These pretext tasks for images include predicting rotation angles (Gidaris et al., 2018), solving jigsaw puzzles (Noroozi & Favaro, 2016) and colorizing grayscale images (Zhang et al., 2016). Finally, the idea of transfer learning hinges on the assumption that features learned from a large and generic dataset can be shared across closely related tasks and datasets (Girshick et al., 2014; Sharif Razavian et al., 2014; Oquab et al., 2014). The most successful models for transfer learning so far are those pre-trained on the ImageNet classification task (Yosinski et al., 2014).

As opposed to proposing new representation-learning tasks, our work studies how to get the most out of the existing tasks. Hence, our method is broadly applicable – it offers a generic framework that can be readily combined with any representation-learning paradigm.

Gradient of Deep Networks. Our method makes use of the Jacobian matrix of a deep network as feature representation for a downstream task. Gradient information is traditionally employed for visualizing and interpreting convolutional networks (Simonyan et al., 2013), and more recently for generating adversarial samples (Szegedy et al., 2013), crafting defense strategies (Goodfellow et al., 2015), facilitating knowledge distillation (Sinha et al., 2018; Srinivas & Fleuret, 2018), and boosting multi-task and meta learning (Sinha et al., 2018; Achille et al., 2019).

Our work draws inspiration from Fisher vectors (FVs) (Jaakkola & Haussler, 1999)–gradient-based features from a probabilistic model (e.g., mixture of Gaussians). The FV pipeline has demonstrated its success for visual recognition based on hand-crafted local descriptors (Perronnin & Dance, 2007). More recently, FVs have shown promising results with deep models, first as an ingredient of a hybrid system (Perronnin & Larlus, 2015), and then as task embeddings for meta-learning (Achille et al., 2019). Our method differs from the FV approaches in two folds. First, it is not built around a probabilistic model, hence has distinct theoretical motivation as we describe later. Second, our method enjoys *exact* gradient computation with respect to network parameters and allows scalable training, whereas Achille et al. (2019) employs heuristics in their method to aggressively approximate the computation of FVs.

Another line of relevant research is from Zinkevich et al. (2017). They also explored similar gradient features of deep networks, which they call holographic features, by drawing inspiration from generalized linear models (GLM). They showed that a linear model trained on the gradient features can faithfully recover the output of the original network for the same task, and also showed some other desired properties of gradient features. In contrast, our work is motivated by the NTK theory,

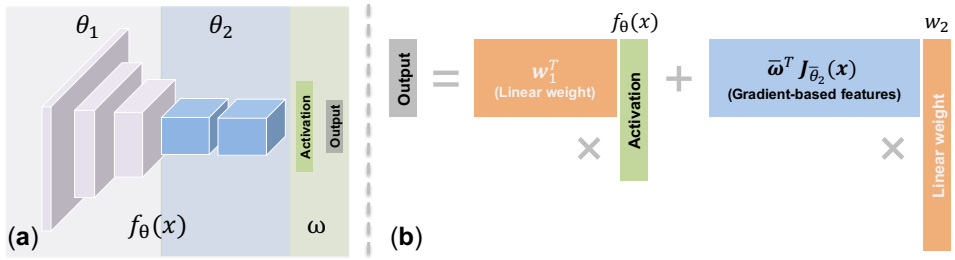

Figure 1: (**a**) An illustration of our parameterization. We consider a deep network $F(\boldsymbol{x};\theta,\boldsymbol{\omega}) \triangleq \boldsymbol{\omega}^T f_\theta(\boldsymbol{x})$ that consists of a ConvNet f_θ with its parameters $\theta \triangleq (\theta_1, \theta_2)$ and linear weights $\boldsymbol{\omega}$. (**b**) An overview of our proposed model. Our model takes the activation $f_\theta(\boldsymbol{x})$ and the gradient $J_{\bar{\theta}_2}(\boldsymbol{x})$ as input (see illustration in (a)), and learns linear weights $\boldsymbol{w}_1$ (matrix) and w_2 (vector) for prediction.

and we study the more challenging problem of *adapting* the features for a different task. It would be interesting to investigate the connection between GLM and NTK.

Neural Tangent Kernel (NTK) for Wide Networks. Jacot et al. (2018) established the connection between deep networks and kernel methods by introducing the neural tangent kernel (NTK). Lee et al. (2019) further showed that a network evolves as a linear model in the infinite width limit when trained on certain losses under gradient descent. Similar ideas have been used to analyze wide networks (see, e.g., Arora et al. (2019b;a); Li & Liang (2018); Allen-Zhu et al. (2019a); Du et al. (2019); Allen-Zhu et al. (2019b); Cao & Gu (2020); Mei et al. (2019)). Our method is, to our best knowledge, the first attempt to materialize the theory in the regime of practical networks. Instead of assuming random initialization of network parameters as all the prior works do, we for the first time empirically evaluate the implication of pre-training on the linear approximation theory.

3 Gradient-based Features for Representation Learning

We start by introducing the setting of representation learning using deep models. Consider a feed-forward deep neural network $F_{\theta,\boldsymbol{\omega}}(\boldsymbol{x}) \triangleq \boldsymbol{\omega}^T f_\theta(\boldsymbol{x})$ that consists of a backbone $f_\theta(\boldsymbol{x})$ with its vectorized parameters θ and a linear model defined by $\boldsymbol{\omega}$ (*italic* for vectors and **bold** for matrix). Specifically, f_θ encodes the input $\boldsymbol{x}$ into a vector representation $f_\theta(\boldsymbol{x}) \in \mathbb{R}^d$. $\boldsymbol{\omega} \in \mathbb{R}^{d\times c}$ are thus linear weights that map a feature vector into c output dimensions. For this work, we focus on convolutional networks (ConvNets) for classification tasks. With trivial modifications, our method can easily extend beyond ConvNets and classification, e.g., for a recurrent network as the backbone and/or for a regression task.

Following the setting of representation learning, we assume that a *pre-trained* $f_{\bar{\theta}}$ is given with $\bar{\theta}$ as the learned parameters. The term representation learning refers to a set of learning methods that do not make use of discriminative signals from the task of interest. For example, f can be the encoder of a deep generative model (Kingma & Welling, 2014; Dumoulin et al., 2016; Donahue et al., 2016), or a ConvNet learned by using proxy tasks (self-supervised learning) (Goyal et al., 2019; Kolesnikov et al., 2019) or from another large-scale labeled dataset such as ImageNet (Deng et al., 2009). Given a target task, it is common practice to regard $f_{\bar{\theta}}$ as a fixed feature extractor (activation-based features) and train a set of linear weights, given by

$$g_{\bar{\boldsymbol{\omega}}}(\boldsymbol{x}) = \bar{\boldsymbol{\omega}}^T f_{\bar{\theta}}(\boldsymbol{x}), \tag{1}$$

We omit the bias term for clarity. Note that $\bar{\boldsymbol{\omega}}$ and $\bar{\theta}$ are instantiations of $\boldsymbol{\omega}$ and θ, where $\bar{\boldsymbol{\omega}}$ is the solution of the linear model and $\bar{\theta}$ is given by representation learning. Based on this setup, we now describe our method, discuss the theoretical implications and present an efficient training scheme.

3.1 Method Outline

Our method assumes a partition of $\theta \triangleq (\theta_1, \theta_2)$, where θ_1 and θ_2 parameterize the bottom and top layers of the ConvNet f (see Figure 1(a) for an illustration). Importantly, we propose to use *gradient*-based features $\nabla_{\bar{\theta}_2} F_{\bar{\theta},\bar{\omega}}(\boldsymbol{x}) = \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x})$ in addition to *activation*-based features $f_{\bar{\theta}}(\boldsymbol{x})$. Specifically, $\boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) \in \mathbb{R}^{d\times|\theta_2|}$ is the Jacobian matrix of $f_{\bar{\theta}}$ with respect to the pre-trained parameter

$\bar{\theta}_2$ from the *top* layers of f. Given the features $(f_{\bar{\theta}}(\boldsymbol{x}), \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}))$ for $\boldsymbol{x}$, our linear model $\hat{g}$, hereby considered as a classifier for concreteness, takes the form

$$\hat{g}_{\boldsymbol{w}_1, w_2}(\boldsymbol{x}) = \boldsymbol{w}_1^T f_{\bar{\theta}}(\boldsymbol{x}) + \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2 = g_{\boldsymbol{w}_1}(\boldsymbol{x}) + \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2, \tag{2}$$

where $\boldsymbol{w}_1 \in \mathbb{R}^{d \times c}$ are linear classifiers initialized from $\bar{\boldsymbol{\omega}}$, $w_2 \in \mathbb{R}^{|\theta_2|}$ are shared linear weights for gradient features, and $|\theta_2|$ is the size of the parameter θ_2. Both $\boldsymbol{w}_1$ and w_2 are our model parameters that need to be learned from a target task. An overview of the model is shown in Figure 1(b).

Our model subsumes the linear model in Eq. (1) as the first term, and includes a second term that is linear in the gradient-based features. We note that this extra linear term is different from standard linear classifiers as in Eq. (1). In this case, the gradient-based features form a matrix and the weight vector w_2 is multiplied to each row of the feature matrix. Therefore, w_2 is shared for all output dimensions. Similar to standard linear classifiers, the output of $\hat{g}$ is further normalized by the softmax function and trained with the cross-entropy loss using labeled data from the target dataset.

Conceptually, our method can be summarized into three steps.

- **Pre-train the ConvNet $f_{\bar{\theta}}$.** This is accomplished by substituting in any existing representation-learning algorithm. Examples include unsupervised, self-supervised and transfer learning.
- **Train linear classifiers $\bar{\boldsymbol{\omega}}$ using $f_{\bar{\theta}}(\boldsymbol{x})$.** This is a standard step in representation learning by fitting a linear model using "generic features".
- **Learn the linear model $\hat{g}_{\boldsymbol{w}_1, w_2}(\boldsymbol{x})$.** A linear model in the form of Eq. (2) is learned using gradient- and activation-based features. Note that the features are obtained when $\theta = \bar{\theta}$ is kept fixed, hence our method *requires no extra tuning of the parameter* $\bar{\theta}$.

3.2 Theoretical Insight

The key insight is that our model provides a local linear approximation to $F_{\theta_2, \boldsymbol{\omega}}(\boldsymbol{x})$. This approximation comes from Eq. (2)–the crux of our approach. Importantly, our linear model is mathematically well motivated –it can be interpreted as the 1st-order Taylor expansion of $F_{\theta, \boldsymbol{\omega}}$ with respect to its parameters $(\theta_2, \boldsymbol{\omega})$ around the point of $(\bar{\theta}_2, \bar{\boldsymbol{\omega}})$. More formally, we note that

$$\begin{aligned} F_{\theta, \boldsymbol{\omega}}(\boldsymbol{x}) &\approx \bar{\boldsymbol{\omega}}^T f_{\bar{\theta}}(\boldsymbol{x}) + \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x})(\theta_2 - \bar{\theta}_2) + (\boldsymbol{\omega} - \bar{\boldsymbol{\omega}})^T f_{\bar{\theta}}(\boldsymbol{x}) \\ &= \boldsymbol{\omega}^T f_{\bar{\theta}}(\boldsymbol{x}) + \bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x})(\theta_2 - \bar{\theta}_2) \\ &= \hat{g}_{\boldsymbol{\omega}, \theta_2 - \bar{\theta}_2}(\boldsymbol{x}). \end{aligned} \tag{3}$$

With $\boldsymbol{\omega} = \boldsymbol{w}_1$ and $\theta_2 - \bar{\theta}_2 = w_2$, Eq. (2) provides a linear approximation of the deep model $F_{\theta_2, \boldsymbol{\omega}}(\boldsymbol{x})$ around the initialization $(\bar{\theta}_2, \bar{\boldsymbol{\omega}})$. Our key intuition is that given a sufficiently strong base network, training our model approximates fine-tuning $F_{\theta_2, \boldsymbol{\omega}}$.

The quality of the linear approximation can be theoretically analyzed via the recent neural tangent kernel approach (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019b) or some related ideas (Arora et al., 2019a; Li & Liang, 2018; Allen-Zhu et al., 2019a; Du et al., 2019; Allen-Zhu et al., 2019b; Cao & Gu, 2020; Mei et al., 2019) when the base network $F_{\theta, \boldsymbol{\omega}}$ is sufficiently wide and at random initialization. Unlike prior works, we apply the linear approximation on pre-trained networks of practical sizes. We argue that such an approximation is useful in practice for the following reasons:

- **The pre-trained network provides a strong starting point.** Thus, the pre-trained network parameter $\bar{\theta}$ is *close* to a good solution for the downstream task. The key to good linear approximation is that the network output is stable with respect to small changes in the network parameters. The pre-trained base network also has such stability properties, which are supported by empirical observations. For example, the pre-trained network has similar predictions for a significant fraction of data in the downstream task as a fine-tuned network.
- **The network width required for the linearization to hold decreases as data becomes more structured.** An assumption made in existing analysis is that the network is sufficiently or even infinitely wide compared to the size of the dataset, so that the approximation can hold for *any* dataset. We argue that this is not necessary in practice, since the practical datasets are *well-structured*, and theoretically it has been shown that as long as the trained network is sufficiently

wide compared to the effective complexity determined by the structure of the data, then the approximation can hold (Li & Liang, 2018; Allen-Zhu et al., 2019a). Our approach thus takes advantage of the bottom layers to reduce data complexity in the hope that linearization of the top (and often the widest) layers can be sufficiently accurate.

3.3 Scalable Training

Moving beyond the theoretical aspects, a practical challenge of our method is the high cost of computing $\hat{g}$ during training and inference. A naïve approach requires evaluating and storing $\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x})$ for all $\boldsymbol{x}$. This is computationally expensive and can become infeasible as the output dimensionality c and the parameter size $|\theta_2|$ grow. Inspired by Pearlmutter (1994), we design an efficient training and inference scheme for $\hat{g}$. Thanks to this scheme, the complexity of training our model using gradient features is of the same magnitude as training a linear classifier on network activation.

Central to our scalable approach is the inexpensive evaluation of the Jacobian-vector product (JVP) $\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2$, whose size is the same as the output dimension c. First, we note that

$$F_{\bar{\theta}+rw_2,\bar{\omega}}(\boldsymbol{x}) = F_{\bar{\theta},\bar{\omega}}(\boldsymbol{x}) + r\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2 + o(r^2) \tag{4}$$

by 1st-order Taylor expansion around a scalar $r = 0$. Rearrange and take the limit of r to 0, we get

$$\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2 = \lim_{r \to 0} \frac{F_{\bar{\theta}+rw_2,\bar{\omega}}(\boldsymbol{x}) - F_{\bar{\theta},\bar{\omega}}(\boldsymbol{x})}{r} = \left.\frac{\partial}{\partial r}\right|_{r=0} F_{\bar{\theta}+rw_2,\bar{\omega}}(\boldsymbol{x}), \tag{5}$$

which can be conveniently evaluated via forward-mode automatic differentiation.

More precisely, let us consider the basic building block of f–convolutional layers. These layers are defined as a *linear* function $h(\boldsymbol{z}_c; \boldsymbol{w}_c, b_c) = \boldsymbol{w}_c^T \boldsymbol{z}_c + b_c$, where $\boldsymbol{w}_c$ and b_c, which are part of θ, are the weight and bias respectively, and $\boldsymbol{z}_c$ is the layer input. We denote the counterparts of $\boldsymbol{w}_c$ and b_c in w_2 as $\tilde{\boldsymbol{w}}_c$ and $\tilde{b}_c$, i.e., $\tilde{\boldsymbol{w}}_c$ and $\tilde{b}_c$ are the linear weights applied to $\boldsymbol{w}_c$ and b_c. It can be shown that

$$\frac{\partial h(\boldsymbol{z}_c; \boldsymbol{w}_c + r\tilde{\boldsymbol{w}}_c, b_c + r\tilde{b}_c)}{\partial r} = h(\boldsymbol{z}_c; \tilde{\boldsymbol{w}}_c, \tilde{b}_c) + h(\frac{\partial \boldsymbol{z}_c}{\partial r}; \boldsymbol{w}_c, 0), \tag{6}$$

where $\frac{\partial \boldsymbol{z}_c}{\partial r}$ is the JVP coming from the upstream layer.

When a nonlinearity is encountered, we have, using the ReLU function as an example,

$$\frac{\partial ReLU(\boldsymbol{z}_c)}{\partial r} = \frac{\partial \boldsymbol{z}_c}{\partial r} \odot \mathbf{1}_{\mathrm{z_c} \geq 0}, \tag{7}$$

where $\odot$ is the element-wise product, $\mathbf{1}$ is the element-wise indicator function and $\boldsymbol{z}_c$ is the layer input. Other nonlinearities as well as pooling layers can be handled in the same spirit. For batch normalization, we fold them into their preceding convolutions.

Importantly, Eq. (6) and (7) provide an efficient approach to evaluating the desired JVP in Eq. (5) by successively evaluating a set of JVPs *on the fly*. This chain of evaluations starts with the seed $\frac{\partial z_0}{\partial r} = 0$, where z_0 is the output of the section of f parameterized by θ_1 and can be pre-computed. $\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2$ can be computed along with the standard forward propagation through f. Moreover, during the training of $\hat{g}$, its parameters $\boldsymbol{w}_1$ and w_2 can be learned via standard back-propagation. In summary, our approach only requires a *single* forward pass through the fixed f for evaluating $\hat{g}$, and a *single* backward pass for updating the parameters $\boldsymbol{w}_1$ and w_2.

Complexity Analysis. We further discuss the complexity of our method in training and inference, and contrast our method to the fine-tuning of network parameters θ_2. Our forward pass, as demonstrated by Eq. (6) and (7), is a chain of linear operations intertwined by element-wise multiplications. The second term in Eq. (6) forms the "main stream" of computation, while the first term merges into the main stream at every layer of the ConvNet f. The same reasoning holds for the backward pass. Overall, our method requires twice as many linear operations as fine-tuning θ_2 of the ConvNet. Note, however, that half of the linear operations by our method are slightly cheaper due to the absence of the bias term. Moreover, in the special case where θ_2 only includes the topmost layer of f, our method carries out the same number of operations as fine-tuning since the second term in Eq. (6) can be dropped. For memory consumption, our method requires to store an additional "copy" (linear weights) of the model parameters compared to fine-tuning. As the size of θ_2 is typically small, this minor increase of computing and memory cost puts our method on the same page as fine-tuning.

4 EXPERIMENTS

Our experimental results are organized into two parts. We first perform ablation studies to understand the representation power of the gradient features. Next, we evaluate our method on three representation-learning tasks: learning deep generative models, self-supervised learning using a pretext task, and transfer learning from ImageNet.

We report results on several datasets and using different base networks to demonstrate the strength of our method. In all cases, our method achieves significant better performance than the standard logistic regressor on network activation. In the most interesting scenario where the pre-training task is similar to the target task, our method achieves comparable or even better performance than fine-tuning. Before we present our results, we summarize our implementation details.

Implementation Details. We adopt the NTK parametrization (Jacot et al., 2018) for θ_2 and fold batch normalization into their preceding convolutional layers prior to training our model. For the SVHN and CIFAR-10/100 experiments, We train the models for 80K iterations with initial learning rate 1e-3, halved every 20K iterations. For the VOC07 and COCO2014 experiments, we train the models for 50 epochs with initial learning rate 1e-3, halved every 20 epochs. All models are trained with the Adam optimizer (Kingma & Ba, 2015) with batch size 64, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and weight decay 1e-6. In addition to Adam, we also use the SGD optimizer with weight decay 5e-5, momentum 0.9 and the same learning rate schedule for fine-tuning, and we report the better result between the two runs.

4.1 ABLATION STUDY

We conduct ablation studies to address the following two questions: Does pre-training encourage more powerful gradient features? What is the optimal size of the gradient features? We answer the first question by comparing gradient features derived from various combinations of random and pre-trained network parameters, and the second by varying the number of layers that contribute to the gradient features.

Baselines. We compare our linear model in Eq. (2), referred to as the **full** model in the sequel, against two baselines. The first baseline $g_{\boldsymbol{w}_1}(\boldsymbol{x})$, or the **activation** model, is the first term in the full model. It is a logistic regressor on network activation and is widely used for representation learning in practice. The second baseline $\bar{\boldsymbol{\omega}}^T \boldsymbol{J}_{\bar{\theta}_2}(\boldsymbol{x}) w_2$, or the **gradient** model, is the second term in the full model. It is linear in the gradient features and can serve as a linear classifier on its own. Moreover, we compare our method against fine-tuning in order to assess the validity of our theoretical insight.

Settings. For ablation study, we consider both unsupervised and supervised learning settings and report classification accuracy. In the *unsupervised setting* (Table 1), we transfer features learned from a deep generative model for image classification. Specifically, we train a BiGAN on CIFAR-10 (Krizhevsky et al., 2009) and use its encoder as the base network for classification of the same dataset. In the *supervised setting* (Table 2), we consider fine-tuning ImageNet pre-trained models on a different classification tasks. Concretely, we use the PyTorch (Paszke et al., 2017) distribution of ImageNet pre-trained ResNet-18 (He et al., 2016) as the base network for VOC07 (Everingham et al., 2010) object classification. For both settings, activation features are *always* output from the pre-trained base network. In contrast, gradient features are evaluated with respect to all possible combinations of random and pre-trained network parameters. In this way, we ensure that variation in performance of the full model can be solely attributed to the gradient features.

Results and Conclusions. Our results are shown in Tables 1 (unsupervised) and 2 (supervised). We summarize our main conclusions from the ablation study.

- **The discriminative power of the gradient features is a consequence of pre-training**. In particular, the full model's performance gain is not a consequence of an increase in parameter size. The full model supplied with random gradients is no better than the activation baseline.
- **Gradient from the topmost layer of a pre-trained network suffices to ensure a reasonable performance gain**. Further inflating the gradient features from bottom layers has diminishing returns, introducing little accuracy gain and sometimes hurting the performance. The results in Zinkevich et al. (2017) also suggest that gradient features from top layers are the most effective in terms of approximating the underlying network.

Table 1: **Ablation study in the unsupervised setting**. We train a BiGAN on CIFAR-10 and use its encoder as the base network for classification of the same dataset. All results are reported using classification accuracy. We obtain the pre-trained $\bar{\theta}_1$ and $\bar{\theta}_2$ from the pre-trained base network, and $\bar{\boldsymbol{\omega}}$ by first training the activation baseline for the target task. The specifications of θ_1, θ_2 and $\boldsymbol{\omega}$ only affects gradient evaluation. Activation features are always output from the pre-trained base network.

			θ_2: conv5		θ_2: conv4-5		θ_2: conv3-5	
			grad	**full**	**grad**	**full**	**grad**	**full**
Random $\bar{\theta}_1$	random θ_2	random $\boldsymbol{\omega}$	23.09	62.83	26.30	62.83	27.52	62.85
		pre-trained $\bar{\boldsymbol{\omega}}$	40.23	63.11	43.36	63.42	44.36	63.38
	pre-trained $\bar{\theta}_2$	random $\boldsymbol{\omega}$	32.50	62.98	33.20	63.10	35.13	63.10
		pre-trained $\bar{\boldsymbol{\omega}}$	44.07	63.99	45.87	64.21	48.88	64.51
Pre-trained $\bar{\theta}_1$	random θ_2	random $\boldsymbol{\omega}$	53.39	63.74	56.24	64.30	57.20	64.60
		pre-trained $\bar{\boldsymbol{\omega}}$	59.98	65.46	61.47	65.38	59.89	64.73
	pre-trained $\bar{\theta}_2$	random $\boldsymbol{\omega}$	70.28	69.84	**71.08**	70.10	**71.60**	70.64
		pre-trained $\bar{\boldsymbol{\omega}}$	70.14	**70.51**	70.89	70.78	**71.55**	71.37
activation			62.87					
fine-tuning			*71.78*		*73.18*		*74.30*	

Table 2: **Ablation study in the supervised setting**. We use the PyTorch distribution of ImageNet pre-trained ResNet18 as the base network for VOC07 object classification. All results are reported using mean average precision (mAP). Predictions are based on a single center crop at test time.

			θ_2: layer4 block2		θ_2: layer4 block1-2	
			grad	**full**	**grad**	**full**
Random θ_1	random θ_2	random $\boldsymbol{\omega}$	15.63	82.84	17.96	82.85
		pre-trained $\bar{\boldsymbol{\omega}}$	17.43	82.79	19.90	82.82
	pre-trained $\bar{\theta}_2$	random $\boldsymbol{\omega}$	18.15	82.85	20.15	82.21
		pre-trained $\bar{\boldsymbol{\omega}}$	20.78	82.76	23.57	82.73
Pre-trained $\bar{\theta}_1$	random θ_2	random $\boldsymbol{\omega}$	62.35	82.84	69.34	82.82
		pre-trained $\bar{\boldsymbol{\omega}}$	65.75	82.84	71.73	82.86
	pre-trained $\bar{\theta}_2$	random $\boldsymbol{\omega}$	80.74	83.15	80.30	82.97
		pre-trained $\bar{\boldsymbol{\omega}}$	83.05	**83.50**	83.12	**83.40**
activation			82.65			
fine-tuning			*82.97*		*82.50*	

- **Our results improve as the dataset and base network grow in complexity**. Our method *always* outperforms the activation baseline by a significant margin when the gradient features are derived from *pre-trained* parameters. Moreover, our method consistently beats the gradient baseline and even fine-tuning on the more challenging VOC07 dataset when using a more complicated residual network.

Remarks. We obtain the pre-trained $\bar{\theta} = (\bar{\theta}_1, \bar{\theta}_2)$ from representation learning, and the pre-trained $\bar{\boldsymbol{\omega}}$ by first training the standard logistic regressor for the target task. We follow the same training procedure in later experiments. According to the ablation study, a good heuristic for our method is to use the gradients from the *topmost* convolutional layer (or residual block) of a pre-trained network. A more principled strategy for selecting gradient-contributing layers is left for future work.

4.2 Results on Representation Learning

We now present results on three different representation-learning tasks: unsupervised learning using generative modeling, self-supervised learning using pretext tasks and transfer learning from pre-trained models. For all experiments in this section, we contrast our proposed linear classifier (i.e., the full model) to the activation and gradient baselines as well as fine-tuning θ_2 for the target task.

Table 3: **Unsupervised Learning Results**. We consider two base networks, namely the encoder of a BiGAN and that of a VAE, trained on three datasets. Our target task is image classification on the same datasets. All results are reported using classification accuracy.

		SVHN	CIFAR-10	CIFAR-100
BiGAN (θ_2: conv5)	**activation**	82.04	62.87	34.30
	gradient	**89.91**	70.14	38.99
	full	**89.96**	**70.51**	**39.37**
	fine-tuning	*91.15*	*71.78*	*41.05*
VAE (θ_2: conv8)	**activation**	81.95	52.05	29.20
	gradient	**91.44**	**61.47**	**35.10**
	full	90.90	61.16	34.83
	fine-tuning	*93.61*	*65.16*	*37.86*

Table 4: **Self-supervised and Transfer Learning Results**. We consider a ResNet-50 pre-trained on the jigsaw pretext task, and a ResNet18 pre-trained on the ImageNet classification task. Our target task is VOC07 and COCO2014 object classification. All results are reported using mean average precision (mAP). Predictions are averaged over ten random crops at test time.

		VOC07	COCO2014
Self-supervised (Jigsaw)	**activation**	57.83	42.10
	gradient	57.73	41.02
	full	**61.70**	**45.62**
	fine-tuning	*67.88*	*52.66*
Zhang et al. (2016)		52.3	n/a
Doersch et al. (2015)		44.7	n/a
Agrawal et al. (2015)		31.2	n/a
Wang & Gupta (2015)		29.4	n/a
Transfer (ImageNet)	**activation**	83.59	62.98
	gradient	84.63	66.33
	full	**84.95**	**66.45**
	fine-tuning	*84.14*	*63.43*

Following our conclusion from the ablation study, we make use of gradient features only from the topmost layer (or residual block) of a pre-trained base network.

Unsupervised Learning. Similar to our ablation study, we consider unsupervised representation learning using deep generative models. We train both BiGAN and VAE, and use their encoders as the pre-trained ConvNet f. Our BiGAN and VAE models follow the architecture and training setup from Dumoulin et al. (2016) and Berthelot et al. (2019). We average-pool the ConvNet output for activation features, and obtain gradient features from the topmost convolutional layer. We train on the `train` split of CIFAR-10/100 and the `extra` split of SVHN, and report classification accuracy on their `test` splits.

Unsupervised Learning Results. Our results are summarized in Table 3. Our model consistently improves the activation baseline by over 10% on all three datasets. Moreover, we observe good agreement of performance between our model and fine-tuning. Finally, we notice that our model does not have significant advantage over the gradient baseline. We think this is because the datasets and base networks used here are so simple that the gradient features alone can be equally effective.

Self-supervised Learning. Moving beyond unsupervised learning, we also consider self-supervised setting, where the representation is learned via a pretext task and pseudo labels. In this case, we consider a ResNet50 pre-trained on the jigsaw self-supervising task (Goyal et al., 2019). See Noroozi & Favaro (2016) for details on the jigsaw task. We average-pool the ConvNet output for activation features, and use gradient features from the last residual block. We train on the `trainval` split of

VOC07 and the `train` split of COCO2014 for object classification, and report the mean average precision (mAP) scores on their respective `test` and `val` splits.

Self-supervised Learning Results. Our results are summarized in the top part of Table 4. We again observe a significant improvement of up to 4.5% over the two baselines on both datasets. Our method also outperforms a large set of self-supervised learning methods using network activation. Note that our method still lags behind fine-tuning in this setting with a large gap (over 6%).

Transfer Learning. Finally, we consider the most widely used transfer learning setting, where the representation is learned by pre-training on a large scale labelled dataset e.g., ImageNet. We experiment with an ImageNet pre-trained ResNet18 from PyTorch. We follow the same setting as self-supervised learning and report results on VOC07 and COCO datasets.

Transfer Learning Results. Our results are summarized in the bottom part of Table 4. Not surprisingly, our results consistently outperforms the two baselines across both datasets. However, the gap between our method and gradient baseline is much smaller. To our surprise, our method outperforms fine-tuning on both datasets by a large margin. In comparison to fine-tuning, our method has a gain of +0.8% and +3.0% on VOC07 and the larger COCO dataset.

4.3 Further Discussions

We provide further discussions of our results. Specifically, we analyze the performance gap between out method and fine-tuning under three different settings. Moreover, we compare our method to the recent effort of learning infinitely wide networks using exact NTK (Arora et al., 2019b).

The performance gap between our method and fine-tuning. We argue that this gap is controlled by the degree of semantic overlap between the representation-learning and target tasks. The jigsaw pretext task, though proven effective for learning features, is still semantically distant to an image classification task, and hence results in the large performance gap we observe in the self-supervised setting. On the other hand, it is reasonable to expect classifiers for datasets with overlapping categories to share common semantics. This may explain why our method outperforms fine-tuning in the transfer-learning experiments, where the source and target datasets are conceptually close.

Comparing our method to learning infinitely wide networks from scratch. Arora et al. (2019b). recently proposed exact NTK computation algorithms for learning infinitely wide ConvNets, and proved that such networks are equivalent to kernel regression using NTK. They obtaind 66% accuracy on CIFAR-10 classification using a vanilla network and achieved an extra 11% boost after including customized network modules. Their approach, at its current stage, only supports full-batch learning and is limited to single output dimension. Our method, though not directly comparable with theirs, does not suffer from these limitations and produces competitive results (e.g., we obtain 70.5% accuracy on CIFAR-10 using a four-layer BiGAN encoder as the base network, see Table 3). Hence, out work highlights another promising direction to capitalize on the insights generated by the theoretical analysis of NTK.

5 Conclusion

We presented a novel method for deep representation learning. Specifically, given a pre-trained deep network, we explored as features the per-sample gradients of the network parameters relative to a task-specific loss, and constructed a linear model that combines network activation with the gradient features. We showed that our model can be very efficient in training and inference, and may be understood as a local linear approximation to an underlying deep model by an appeal to the neural-tangent-kernel (NTK) theory. We empirically demonstrated that the gradient features are highly discriminative for downstream tasks, and our method can significantly improve over the baseline method of representation learning across pre-training tasks, network architectures and datasets. We believe that our work offers a novel perspective to improving deep representation learning. Future research directions include using NTK as a distance metric for measuring sample similarity in low-shot classification and assessing model similarity in knowledge distillation.

Acknowledgement: This work was supported in part by FA9550-18-1-0166. The authors would also like to acknowledge the support provided by the University of Wisconsin-Madison Office of

the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *ICCV*, 2019.

Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, 2015.

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *NeurIPS*, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019b.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, 2019a.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *NeurIPS*, 2019b.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *ICLR*, 2019.

Yuan Cao and Quanquan Gu. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *AAAI*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2016.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *ICML*, 2019.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2016.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *ICLR*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *NeurIPS*, 1999.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019.

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

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *NeurIPS*, 2019.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *NeurIPS*, 2018.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *COLT*, 2019.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS*, 2017.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *CVPR*, 2015.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.

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

Ayan Sinha, Zhao Chen, Vijay Badrinarayanan, and Andrew Rabinovich. Gradient adversarial training of neural networks. *arXiv preprint arXiv:1806.08028*, 2018.

Suraj Srinivas and Francois Fleuret. Knowledge transfer with Jacobian matching. In *ICML*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

Martin A Zinkevich, Alex Davies, and Dale Schuurmans. Holographic feature representations of deep networks. In *UAI*, 2017.