Fast-NTK: Parameter-Efficient Unlearning for Large-Scale Models

Guihong Li[1,*], **Hsiang Hsu**[2], **Chun-Fu (Richard) Chen**[2], **Radu Marculescu**[1]
[1]The University of Texas at Austin, USA
[2]Global Technology Applied Research, JPMorgan Chase, USA
{lgh, radum}@utexas.edu
{hsiang.hsu, richard.cf.chen}@jpmchase.com

Abstract

The rapid growth of machine learning has spurred legislative initiatives such as "the Right to be Forgotten," allowing users to request data removal. In response, machine unlearning proposes the selective removal of unwanted data without the need for retraining from scratch. While the Neural-Tangent-Kernel (NTK) based unlearning method excels in performance, it suffers from significant computational complexity, especially for large-scale models and datasets. To improve this situation, our work introduces "Fast-NTK," a novel NTK-based unlearning algorithm that significantly reduces the computational complexity by incorporating parameter-efficient fine-tuning methods, such as fine-tuning batch normalization layers in a CNN or visual prompts in a vision transformer. Our experimental results demonstrate scalability to really large neural networks and datasets (e.g., 88M parameters and 5k images), surpassing the limitations of previous full-model NTK-based approaches designed for smaller cases (e.g., 8M parameters and 500 images). Notably, our approach maintains a performance comparable to the traditional methods of retraining on the retain set alone. Fast-NTK can thus enable practical and scalable NTK-based unlearning in deep neural networks.

1. Introduction

The surge of machine learning applications has prompted legislative actions, notably "the Right to be Forgotten," allowing individuals to request the removal of their online information [28]. However, the privacy challenge remains as erasing data from databases may persist in machine learning models, particularly in deep neural networks (DNNs), which are recognized for their efficient training data memorization [30]. To address this issue, machine unlearning has emerged to enable selective removal of unwanted "forget samples" without the need of retraining the model from scratch [25].

Among various unlearning algorithms [2, 3, 5, 11, 24, 29], NTK-based unlearning stands out for its state-of-the-art performance [8, 9]. However, NTK-based unlearning algorithms are challenging due to the need of computing kernel matrices with respect to all samples and model weights. This computational complexity grows polynomially with the number of samples and model weights, thus resulting in intensive computation costs and memory consumption. Consequently, the effectiveness of NTK-based unlearning algorithms is often limited only to small-scale models and datasets (e.g., 8M parameters and 500 images).

In this work, we draw inspiration from parameter-efficient fine-tuning (PEFT) [4, 18, 22, 33] and leverage the NTK-based unlearning algorithms — specifically, the computation of kernel matrices — to work with a limited set of important parameters, such as those used in batch normalization layers and visual prompts. We term this approach "Fast-NTK," as shown in Figure 1. Unlike the conventional application of NTK-based unlearning algorithms, Fast-NTK significantly reduces the parameter count (cf. Table 2) of the standard implementation of the entire model. Remarkably, our experimental results, e.g., vision transformers (ViTs) on the ImageNet-R dataset, demonstrate indistinguishable performance compared to the commonly used baseline that retrains the model from scratch only on the remaining data. Consequently, we believe our approach provides a practical and scalable solution for the NTK-based unlearning approaches.[1]

2. Background and Related Work

Consider a training dataset $\mathcal{D}$ that can be divided into two disjoint subsets: a forget set $\mathcal{D}_f$ which is the target for unlearning, and a retain set $\mathcal{D}_r$ which contains the remaining samples. The objective of machine unlearning is to eliminate the knowledge from the forget samples in $\mathcal{D}_f$

*Work done during an internship at JPMorgan Chase Bank, N.A.

[1]Codes to reproduce our experiments are public at GitHub.

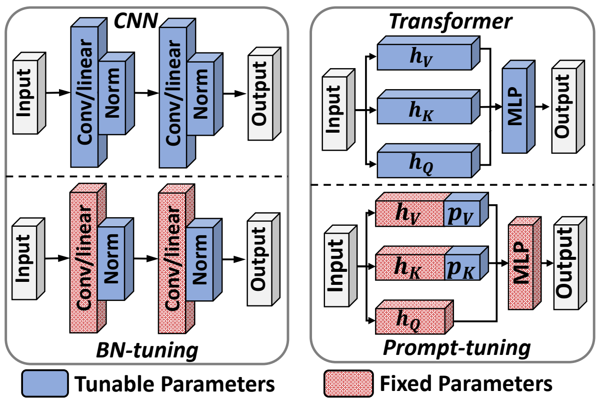

Figure 1. A schematic of parameter-efficient fine-tuning and unlearning. For CNNs (left), instead of updating the entire model, we conduct the fine-tuning and NTK-based unlearning on batch normalization (BN) layers. For transformers (right), we only modify the appended prompts ($\boldsymbol{p}_K$ and $\boldsymbol{p}_V$).

of a model trained with $\mathcal{D}$, while minimizing the performance degradation of the retain samples in $\mathcal{D}_r$ [31]. One simple strategy is to retrain the entire model from scratch, utilizing only the samples in $\mathcal{D}_r$. However, this process is time-consuming, particularly when dealing with large-scale datasets and models. Consequently, current research endeavors to directly erase the knowledge associated with the forget samples from the model, without necessitating a complete retraining.

There exist three distinct strategies for accomplishing machine unlearning: data partitioning [2], mimicking differential privacy [11], and adjusting the model weights [3, 5, 24, 29]. Our work delves into the intricacies of updating the model weights, hence targeting machine unlearning through the computation of NTKs [16, 21].

Consider a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^d$, where $\mathcal{X}$ and $\mathcal{Y}$ are the support sets of the input and output, respectively. The NTK matrix of the two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as:

$$\boldsymbol{\Theta}(\mathcal{D}_1, \mathcal{D}_2) \triangleq \nabla_\theta f_\theta(\mathcal{D}_1) \nabla_\theta f_\theta(\mathcal{D}_2)^\top \quad (1)$$

Let θ and θ_r be the weights from training with the entire training set $\mathcal{D}$ and the retain set $\mathcal{D}_r$ alone, respectively[2]. By linearizing the outputs of f_θ, we can approximate θ and θ_r in closed forms, and directly move the model weights from θ to θ_r by an optimal one-shot update:

$$\theta_r = \theta + \boldsymbol{P} \nabla_\theta f_\theta(\mathcal{D}_f)^\top \boldsymbol{M} \boldsymbol{V}, \quad (2)$$

where $\boldsymbol{P} = \boldsymbol{I} - \nabla_\theta f_\theta(\mathcal{D}_r)^\top \boldsymbol{\Theta}(\mathcal{D}_r, \mathcal{D}_r)^{-1} \nabla_\theta f_\theta(\mathcal{D}_r)$ is the matrix that projects the gradients of the samples to forget $\nabla_\theta f_\theta(\mathcal{D}_f)$ to a space that is orthogonal to the space spanned by the gradients of all retain samples; $\boldsymbol{M} = \left[\boldsymbol{\Theta}(\mathcal{D}_f, \mathcal{D}_f) - \boldsymbol{\Theta}(\mathcal{D}_r, \mathcal{D}_f)^\top \boldsymbol{\Theta}(\mathcal{D}_r, \mathcal{D}_r)^{-1} \boldsymbol{\Theta}(\mathcal{D}_r, \mathcal{D}_f)\right]^{-1}$ and $\boldsymbol{V} = (\mathbf{y}_f - f_\theta(\mathcal{D}_f)) + \boldsymbol{\Theta}(\mathcal{D}_r, \mathcal{D}_f)^\top \boldsymbol{\Theta}(\mathcal{D}_f, \mathcal{D}_f)^{-1}(\mathbf{y}_r - f_\theta(\mathcal{D}_r))$ are the re-weighting matrices, while $\mathbf{y}_f$ and $\mathbf{y}_r$ are the ground truth labels for the forget set and retain set, respectively.

[2]Note that directly obtaining θ_r from θ is the goal of machine unlearning by updating model weights.

Although the NTK-based unlearning provides state-of-the-art performance in comparison to other methods [17], there are concerns regarding its numerical instability and scalability for models with many parameters [8, 9]. The inherent computational complexity has spurred efforts to enhance the efficiency of NTK-based unlearning algorithms, especially in large-scale setups. One approach to mitigate the computational costs involves the utilization of sketching techniques to approximate the tensor products associated with NTK [32]. This method not only scales linearly with data sparsity, but also efficiently truncates the Taylor series of arc-cosine kernels. Additionally, improvements in the spectral approximation of the kernel matrix are achieved through leveraging the score sampling, or introducing a distribution that efficiently generates random features by approximating scores of arc-cosine kernels [32]. Further strides in computational efficiency are made by novel algorithms employing mixed-order or high-order automatic differentiation [26]. It is important to note that these methods are often tailored to specific types of deep neural networks, thus limiting their widespread applicability. Moreover, their efficiency may still fall short for some larger deep networks [26]. Consequently, our objective is to propose a parameter-efficient and practical implementation of NTK-based unlearning methods, as discussed next.

3. Proposed Method

3.1. Fast-NTK

The major barrier in NTK-based unlearning arises from the computation of the Jacobian matrix $\nabla_\theta f_\theta(\mathcal{D})$, defined in Eq. (1) and (2), with dimensions $|\mathcal{Y}||\mathcal{D}_f| \times d$. In the context of deep neural networks, the parameter count d spans a vast range, from millions to trillions [7, 27]. This abundance of parameters poses a formidable challenge due to the prohibitive costs in computation and storage, and has indeed been a primary impediment in applying NTK-based unlearning algorithm on large scale models. To mitigate the computational and storage burdens, the concept of PEFT has been recently proposed in Houlsby et al. [14]. PEFT selectively fine-tunes only a small subset of (additional) model parameters. Recent empirical findings indicate that state-of-the-art PEFT techniques achieve performance comparable to that of full fine-tuning (i.e., tuning all parameters) [33], but with a lower computational cost.

Drawing inspiration from PEFT, we extend the approach to NTK-based unlearning by selectively focusing on a subset of model parameters—this combined technique is referred to as "Fast-NTK." As illustrated in Fig. 1, in the case

Table 1. Prompt-based FAST-NTK on ViTs with CIFAR-10. All results are averaged over 5 runs with different seeds. The results closest to RETRAIN are considered as the best results and shown in **bold**.

Dataset	Architectures	ViT-Small			ViT-Base		
	#Images per class	100	200	500	100	200	500
CIFAR-10	#Params ratio (%)	0.11	0.11	0.11	0.05	0.05	0.05
Accuracy on $\mathcal{D}_r$	FULL	95.78±0.52	94.93±1.06	94.78±0.48	84.18±1.09	85.67±0.62	87.07±0.24
	RETRAIN	96.02±0.43	95.71±0.53	94.29±0.20	84.36±1.16	86.47±0.58	88.19±0.32
	MAX LOSS	87.18±1.19	86.53±0.79	83.47±0.40	78.04±0.60	**84.26±0.83**	**87.39±0.08**
	RANDOM LABEL	93.87±0.86	93.72±0.55	93.32±0.35	76.56±0.83	83.83±0.82	87.28±0.17
	Fast-NTK	**93.91±0.77**	**94.84±1.25**	**94.59±0.03**	**87.60±1.16**	89.13±0.51	89.30±0.12
Accuracy on $\mathcal{D}_f$	FULL	97.00±1.55	96.20±1.36	95.73±1.67	84.80±6.31	90.40±1.11	92.00±0.00
	RETRAIN	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	MAX LOSS	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	RANDOM LABEL	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	Fast-NTK	0.20±0.40	0.20±0.24	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	0.20±0.20
Accuracy on Hold-Out set	FULL	86.62±1.42	87.51±0.93	89.29±0.18	82.06±0.77	84.95±0.95	86.78±0.28
	RETRAIN	78.94±1.11	79.61±0.44	80.64±0.10	73.76±0.43	76.56±0.87	78.59±0.27
	MAX LOSS	71.90±0.45	73.38±0.72	73.09±0.24	68.14±0.71	74.60±0.86	77.73±0.21
	RANDOM LABEL	**78.12±0.91**	**79.01±0.59**	80.30±0.07	66.80±1.23	74.34±0.83	77.73±0.35
	Fast-NTK	77.78±1.14	78.87±0.97	**80.63±0.23**	**70.62±2.10**	**75.37±0.45**	**78.47±0.15**
#Relearning Epochs	RETRAIN	2.60±0.49	1.40±0.49	1.00±0.00	>100	>100	46.50±0.50
	MAX LOSS	9.20±0.40	8.00±0.00	6.00±0.00	**>100**	**>100**	**47.50±0.50**
	RANDOM LABEL	2.20±0.40	**1.20±0.40**	**1.00±0.00**	>100	>100	45.00±1.00
	Fast-NTK	**2.60±0.49**	1.00±0.00	**1.00±0.00**	**>100**	**>100**	53.50±1.50

of convolutional neural networks (CNNs), our approach involves fine-tuning the batch normalization (BN) layers, which has proven to be an effective strategy for adapting a trained model to new data domains [4, 22]. Meanwhile, for vision transformers (ViTs), success is achieved by fine-tuning several prompts appended to the attention blocks [18, 23, 33]. To elaborate, given a pre-trained CNN or ViT, we perform fine-tuning on the downstream dataset $\mathcal{D}$ by using BN-based adjustments (for CNNs) or prompt-based modifications (for ViTs).

Subsequently, when provided with a forget set $\mathcal{D}_f$, we execute NTK-based unlearning using Eq. (2) exclusively on the fine-tuned parameters. This streamlined Fast-NTK approach significantly reduces the parameters subjected to fine-tuning, down to a range of $0.05\% \sim 4.88\%$ of the full model parameters. Remarkably, Fast-NTK achieves a performance comparable to tuning all parameters, as demonstrated in the next section.

3.2. Parameter Reduction of Fast-NTK

Fine-tune/unlearn CNNs with BN layers. As shown in Fig. 1, a convolutional layer is typically followed by a batch normalization layer in a CNN. For a typical convolutional layer with C_o output channels, C_i input channels, kernel size $K \times K$, and g separable groups, the total number of parameters (weights) in this layer is $\text{Parameters}_{\text{conv}} = \frac{C_o \times C_i \times K^2}{g}$. In contrast, for a batch normalization (BN) layer, the only learnable parameters are the scaling (γ) and shifting (β) terms for each channel. Hence, the total number of learnable parameters in a BN layer is then $\text{Parameters}_{\text{BN}} = 2 \times C_o$. Usually, $C_i \times K^2 \gg 2$ and $g = 1$; therefore

$$\frac{\text{Parameters}_{\text{conv}}}{\text{Parameters}_{\text{BN}}} = \frac{C_i \times K^2}{2g} \gg 1. \tag{3}$$

Fine-tune/unlearn ViTs with Prompts. In a ViT, the embedding layer transforms the input image into a sequence-

Table 2. BN-based FAST-NTK on CNNs with CIFAR-10. All results are averaged over 5 runs with different seeds. The results closest to RETRAIN are considered as the best results and shown in **bold**.

Dataset	Architectures	MobileNet-V2			ResNet-110		
	#Images per class	100	200	500	100	200	500
CIFAR-10	#Params ratio (%)	4.88	4.88	4.88	0.51	0.51	0.51
Accuracy on $\mathcal{D}_r$	FULL	74.42±2.17	78.54±0.62	84.12±0.24	66.87±1.03	72.28±1.39	77.22±1.27
	RETRAIN	75.56±2.36	79.50±0.55	85.27±0.26	69.02±1.65	74.13±1.54	78.98±0.43
	MAX LOSS	**71.13±1.91**	68.24±1.40	14.12±1.59	56.64±2.17	49.17±2.19	13.49±1.89
	RANDOM LABEL	69.58±2.21	69.02±1.72	66.94±2.84	58.76±1.58	66.58±1.71	72.58±2.42
	Fast-NTK	70.80±2.04	**73.70±0.68**	**80.76±0.40**	**65.60±4.36**	**71.04±1.65**	**76.84±0.21**
Accuracy on $\mathcal{D}_f$	FULL	68.40±5.28	75.00±4.17	84.80±2.07	67.20±3.06	73.70±1.81	75.20±0.98
	RETRAIN	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	MAX LOSS	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	RANDOM LABEL	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	Fast-NTK	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
Accuracy on Hold-Out set	FULL	65.00±1.11	71.63±1.25	77.91±0.21	54.12±0.72	62.29±1.12	71.02±0.71
	RETRAIN	60.50±1.15	66.41±0.46	71.80±0.19	50.36±1.17	57.57±0.71	65.58±1.51
	MAX LOSS	58.14±0.95	58.28±1.22	12.51±1.27	43.18±0.34	41.61±2.06	12.06±2.81
	RANDOM LABEL	57.04±1.15	63.36±0.48	69.27±0.15	43.38±0.91	52.53±0.64	**61.18±0.60**
	Fast-NTK	**58.54±0.88**	**63.96±1.67**	**69.96±3.64**	**50.80±5.57**	**59.88±1.59**	60.58±0.63
#Relearning Epochs	RETRAIN	21.20±0.40	11.00±0.00	4.80±0.40	12.60±0.49	6.20±0.40	3.00±0.00
	MAX LOSS	28.80±0.40	22.20±0.40	77.20±6.01	24.00±0.89	25.20±2.48	22.00±0.93
	RANDOM LABEL	19.80±0.40	10.00±0.00	**4.00±0.00**	10.80±0.40	**6.00±0.00**	**3.00±0.49**
	Fast-NTK	**21.00±0.63**	**10.80±0.40**	**4.00±0.00**	**12.40±0.80**	**6.00±0.00**	2.80±0.40

like feature representation with the embedding dimension of E. Next, the representation is processed by several transformer block, consisting of a multi-head self-attention (MSA) block and two multi-layer perceptron (MLP) layers to obtain the outputs. Within each block, each MLP layer has $E \times rE$, where r is usually 4; so two MLP layers have $8E^2$ parameters. Besides, each attention head has three weight matrices of size $\frac{E}{m} \times E$, where m is the number of attention heads in a given MSA. Hence, MSA has $3E^2$ parameters, and, in total:

$$\text{Parameters}_{\text{Block}} = 8E^2 + 3m \times \frac{E}{m} \times E = 11E^2$$

As shown in Fig. 1, the prompt-based fine-tuning inserts the prompt parameters $\boldsymbol{p}_K$ and $\boldsymbol{p}_V$ to the Key and Value $\boldsymbol{h}_K$ and $\boldsymbol{h}_V$ of an MSA.

As a contrast to tuning the entire MSA, the prompt-based method fine-tunes only $L_p \times E$ parameters, where L_p is the number of appended prompts. Typically, the embedding dimensions E is much higher than the prompt length L_p (in our experimental setup, $L_p = 10$); therefore:

$$\frac{\text{Parameters}_{\text{block}}}{\text{Parameters}_{\text{prompt}}} = \frac{11E}{L_p} \gg 1. \tag{4}$$

4. Empirical results

4.1. Setup

Our method starts with the CNNs and ViTs pre-trained on the CIFAR-100 and ImageNet-1K datasets, respectively. We fine-tune these pre-trained models on the CIFAR-10 [20] and ImageNet-R [13] datasets and then assess the performance of FAST-NTK. In the case of CIFAR-10, we designate one class as $\mathcal{D}_f$, while considering the remaining classes as $\mathcal{D}_r$. Similarly, for the ImageNet-R dataset, we randomly choose one class as $\mathcal{D}_f$ and select either 19 or 49 classes from the 200 classes as $\mathcal{D}_r$ (i.e., resulting in 20 or 50 total classes in $\mathcal{D}$) to demonstrate the scalability of our approach. Besides, we vary the number of the images per

Table 3. Prompt-based FAST-NTK on ViTs with ImageNet-R. All results are averaged over 5 runs with different seeds. The results closest to RETRAIN are considered as the best results and shown in **bold**.

	Architectures	ViT-Tiny		ViT-Small		ViT-Base	
Dataset	#Classes/#IPC	20/50	50/20	20/50	50/20	20/50	50/20
ImageNet-R	#Params ratio (%)	0.24	0.35	0.12	0.18	0.06	0.09
Accuracy on $\mathcal{D}_r$	FULL	66.40±0.91	65.48±0.85	87.60±0.89	85.56±1.58	36.82±2.55	15.36±1.17
	RETRAIN	68.21±1.50	65.92±0.93	87.77±0.42	86.58±0.15	37.45±2.03	16.02±1.02
	MAX LOSS	57.71±1.33	51.80±0.70	77.35±1.23	71.17±0.66	24.78±2.74	8.52±0.77
	RANDOM LABEL	58.29±1.70	51.50±0.97	78.51±1.52	71.43±0.20	23.56±2.99	7.60±0.77
	Fast-NTK	**66.53±0.63**	**65.24±0.48**	**87.03±1.49**	**85.31±2.14**	**40.84±1.84**	**17.40±1.89**
Accuracy on $\mathcal{D}_f$	FULL	77.20±5.60	56.67±20.95	91.60±3.44	87.50±2.50	56.80±11.91	20.00±5.00
	RETRAIN	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	MAX LOSS	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	RANDOM LABEL	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
	Fast-NTK	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**	**0.00±0.00**
Accuracy on Hold-Out set	FULL	47.73±0.45	31.43±1.70	68.06±1.12	52.75±0.75	32.53±2.40	12.05±0.05
	RETRAIN	46.54±1.26	30.67±2.25	64.69±0.90	51.60±0.90	30.29±2.24	11.60±0.10
	MAX LOSS	41.56±1.05	26.13±0.87	59.41±1.03	45.15±0.05	19.67±3.20	6.55±0.55
	RANDOM LABEL	45.02±1.92	26.03±1.31	64.03±1.23	45.40±0.10	19.96±3.60	6.55±0.25
	Fast-NTK	**45.44±0.93**	**31.17±1.28**	**64.03±1.03**	**52.40±0.50**	**23.26±2.40**	**9.15±0.35**
#Relearning Epochs	RETRAIN	5.00±0.00	4.67±0.47	6.40±0.49	6.50±0.50	>100	>100
	MAX LOSS	17.00±0.00	13.67±0.47	18.00±0.00	15.00±0.00	>**100**	>**100**
	RANDOM LABEL	4.20±0.40	3.67±0.47	**6.40±0.49**	**6.00±0.00**	>**100**	>**100**
	Fast-NTK	**4.40±0.49**	**4.00±0.00**	5.80±0.40	**6.00±0.00**	>**100**	>**100**

Table 4. Linear probing on the ImageNet-R dataset. All results are averaged over 5 runs with different seeds.

	Network	ViT-Small			ViT-Base		
	#Classes/#IPC	20/20	20/50	50/20	20/20	20/50	50/20
Acc on $\mathcal{D}_r$	PRE-TRAINED	60.39±1.27	58.24±1.49	53.70±2.36	99.93±0.11	99.32±0.24	99.87±0.08
	RANDOM-INIT	35.66±1.69	26.40±1.06	22.32±0.60	32.31±0.74	19.30±0.66	17.33±0.69
	Fast-NTK	60.25±3.76	53.71±2.45	47.24±0.00	86.58±2.13	87.66±1.01	87.24±0.00
Acc on $\mathcal{D}_f$	PRE-TRAINED	72.50±9.01	79.50±6.22	66.25±5.45	100.00±0.00	99.00±1.00	98.75±2.17
	RANDOM-INIT	54.53±2.46	33.33±17.00	43.33±11.12	49.40±4.42	15.00±8.16	17.33±7.72
	Fast-NTK	0.00±0.00	0.00±0.00	0.00±0.00	2.50±2.50	0.00±0.00	0.00±0.00

class (IPC) from 20 to 500. For example, for ImageNet-R dataset, we can set the number of classes as 20 (one forget class plus 19 retain classes) and set IPC as 200, then in total we have 4,000 images in $\mathcal{D}$; we can set IPC as 500 for CIFAR-10, we have 5,000 images in $\mathcal{D}$.

We consider the following three metrics. First, we measure accuracy on both $\mathcal{D}_r$ and $\mathcal{D}_f$ — an unlearning algorithm should maintain high accuracy on $\mathcal{D}_r$ while minimizing accuracy on $\mathcal{D}_f$. Second, we calculate accuracy on a hold-out set to ensure consistent performance on unseen data. Note that the hold-out set may contain samples from classes present in both $\mathcal{D}_f$ and $\mathcal{D}_r$. The accuracy on the hold-out set should remain unaffected by the unlearning algorithm. Third, we incorporate relearning time [8], representing the number of epochs to achieve a training loss below 0.05 on the forget set[3]. Relearning time serves as a measure of the difficulty in recovering knowledge from the forget set. If the model fails to achieve a loss below 0.05 within 100 epochs, we denote it as '>100'.

We compare FAST-NTK against the following baselines:

- FULL: The original model fine-tuned on $\mathcal{D} = \mathcal{D}_f \cup \mathcal{D}_r$ without unlearning, serving as the reference model.
- MAX LOSS [12]: This baseline maximizes the training loss with respect to the ground truth labels of the samples in the forget set $\mathcal{D}_f$.
- RANDOM LABEL [10, 19]: This baseline minimizes the training loss by assigning uniformly random labels to the samples in the forget set $\mathcal{D}_f$.
- RETRAIN: The model trained only on the retain set $\mathcal{D}_r$.

Among these baselines, RETRAIN is commonly referred to as the **golden baseline**. This designation stems from its lack of prior knowledge about the samples in the forget set $\mathcal{D}_f$, making it an ideal reference point for comparing any unlearning algorithms. By evaluating FAST-NTK against RETRAIN, we aim to ensure that the unlearned model closely approximates the ideal scenario. This comparison helps ascertain that the unlearning process effectively eliminates unwanted data without causing significant performance degradation on $\mathcal{D}_r$. Essentially, an ideal unlearned model should exhibit indistinguishability in terms of the specified evaluation metrics to the golden baseline RETRAIN (see [25, Section 3.2]).

4.2. Evaluation of Fast-NTK

We perform BN-based fine-tuning on ViTs, MobileNet-v2 and ResNet-110 using a subset of the CIFAR-10 dataset, followed by unlearning algorithms that involves forgetting the class labeled "0." To showcase the scalability of our approach, we vary the number of images per class (#IPC). The results in Table 1 and Table 2 reveal that our method requires less than **4.88%** of the parameters involved in fine-tuning the entire model, thus making the unlearning process practical and achievable for these large models. Notably, FAST-NTK exhibits negligible or no accuracy degradation on the retain set compared to the golden baseline RETRAIN. In contrast, the accuracy on the forget set is indistinguishable from RETRAIN (drops to "0") across various setups, with a similar number of relearning epochs needed as RETRAIN. Compared to the other baselines, MAX LOSS and RANDOM LABEL, FAST-NTK effectively preserves accuracy on the retrain set $\mathcal{D}_r$ and the general test set, highlighting the robustness and efficiency of our proposed technique for CNNs.

[3]Here, we use 0.05 but it can be other values.

Additionally, we extend the same setting to ViTs on the ImageNet-R dataset. As demonstrated in Table 3, our approach requires less than **0.4%** of the parameters compared to tuning the entire model, thus making it practical unlearning feasible for these large models. Comparisons with RETRAIN, MAX LOSS, and RANDOM LABEL show that FAST-NTK effectively preserves accuracy on the retain set $\mathcal{D}_r$ and the general test set, achieving close accuracy to RETRAIN on the retain set. These results confirm the effectiveness and practicality of our unlearning approach for ViTs. Importantly, our method scales up to ViTs, representing a significant advancement compared to previous approaches like [8], which are confined only to toy networks and small datasets (e.g., 8M parameters and less than 200 samples).

5. Discussion

Risk of using pre-trained models. It is crucial to emphasize that FAST-NTK starts with a pre-trained model rather than one initialized randomly. Despite the increasing popularity of leveraging pre-trained foundation models [1], these pre-trained models may possess some knowledge of classes from $\mathcal{D}_f$. This prior knowledge introduces an inherent risk for the unlearning process, as erasing all information and concepts associated with the classes in $\mathcal{D}_f$ solely through the use of forget samples becomes a challenging task.

To assess this risk, for the pre-trained models used in our evaluation (PRE-TRAINED), we conduct fine-tuning of the classification head (i.e., linear probing) on $\mathcal{D}_r \cup \mathcal{D}_f$, while keeping the parameters in the remaining layers frozen. We also conduct the linear probing on the randomly initialized model (RANDOM-INIT) and the unlearned model obtained by FAST-NTK (cf. Section 4).

As illustrated in Table 4, the accuracy of PRE-TRAINED on $\mathcal{D}_r$ and $\mathcal{D}_f$ is much higher than RANDOM-INIT (very close to 100%), indicating that the pre-trained model already possesses some level of knowledge about $\mathcal{D}_r$ and $\mathcal{D}_f$. As expected, FAST-NTK effectively removes the knowledge on $\mathcal{D}_f$ as the accuracy on $\mathcal{D}_f$ is zero. This finding underscores the need for further investigation into the interplay between unlearning and PEFT on pre-trained models.

Future work. Our current implementation to obtain the NTK matrix relies on exact computations. To further improve the efficiency of FAST-NTK, one future direction is to explore approximate computation of the NTK matrix, e.g., by low-rank approximation or factorization [6, 15].

6. Conclusion

In this work, we have proposed "Fast-NTK", an innovative approach to machine unlearning that addresses the computational challenges associated with Neural-Tangent-Kernel-based (NTK-based) methods. By integrating the parameter-efficient fine-tuning techniques, Fast-NTK significantly reduces computational complexity, making it an efficient and practical solution for large-scale models and datasets. Our experimental results demonstrate that Fast-NTK not only significantly improve the scalability of prior full-model NTK-based strategies but also achieves comparable accuracy with the classical retraining-based methods. Our approach paves the way for practical and scalable NTK-based unlearning in deep neural networks.

Disclaimer

This paper was prepared for informational purposes by the Global Technology Applied Research center of JPMorgan Chase & Co. This paper is not a product of the Research Department of JPMorgan Chase & Co. or its affiliates. Neither JPMorgan Chase & Co. nor any of its affiliates makes any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, without limitation, with respect to the completeness, accuracy, or reliability of the information contained herein and the potential legal, compliance, tax, or accounting effects thereof. This document is not intended as investment research or investment advice, or as a recommendation, offer, or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. Guihong Li's and Radu Marculescu's contributions were made as part of Guihong Li's internship at the Global Technology Applied Research center of JPMorgan Chase & Co.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 6

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy*, pages 141–159. IEEE, 2021. 1, 2

[3] Yuantao Chen, Jie Xiong, Weihong Xu, and Jingwen Zuo. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing*, 22:7435–7445, 2019. 1, 2

[4] Hung-Yueh Chiang, Natalia Frumkin, Feng Liang, and Diana Marculescu. MobileTL: on-device transfer learning with inverted residual blocks. In *Proceedings of the AAAI*, 2023. 1, 3

[5] Rishav Chourasia and Neil Shah. Forget unlearning: Towards true data-deletion in machine learning. In *Proceedings of ICML*. PMLR, 2023. 1, 2

[6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of ICLR*, 2021. 2

[8] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of CVPR*. IEEE, 2020. 1, 2, 6

[9] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Proceedings of ECCV*. Springer, 2020. 1, 2

[10] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI*, pages 11516–11524, 2021. 6

[11] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. In *Advances in NeurIPS*, 2021. 1, 2

[12] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl? *CoRR*, abs/2207.05521, 2022. 6

[13] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020. 4

[14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*. PMLR, 2019. 2

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. 6

[16] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generaliza-

tion in neural networks. In *Advances in NeurIPS*, 2018. 2

[17] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsification can simplify machine unlearning. *CoRR*, abs/2304.04934, 2023. 2

[18] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of ECCV*. Springer, 2022. 1, 3

[19] Zhifeng Kong and Kamalika Chaudhuri. Data redaction from conditional generative models. *CoRR*, abs/2305.11351, 2023. 6

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 4

[21] Jaehoon Lee et al. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in NeurIPS*, 2019. 2

[22] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory. In *Advances in NeurIPS*, 2022. 1, 3

[23] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 3

[24] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of ALT*. PMLR, 2021. 1, 2

[25] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *CoRR*, abs/2209.02299, 2022. 1, 6

[26] Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Fast finite width neural tangent kernel. In *Proceedings of ICML*. PMLR, 2022. 2

[27] 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. In *Proceedings of ICML*. PMLR, 2021. 2

[28] General Data Protection Regulation. General data protection regulation (gdpr). *Intersoft Consulting, Accessed in October*, 24(1), 2018. 1

[29] Ayush Kumar Tarun, Vikram Singh Chundawat, Murari Mandal, and Mohan S. Kankanhalli. Deep regression unlearning. In *Proceedings of ICML*. PMLR, 2023. 1, 2

[30] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger B. Grosse. On the quantitative analysis of decoder-based generative models. In *Proceedings of ICLR*, 2017. 1

[31] Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. Machine unlearning: A survey. *ACM Comput. Surv.*, 56(1), 2023. 2

[32] Amir Zandieh, Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Scaling neural tangent kernels via sketching and random features. In *Advances in NeurIPS*, 2021. 2

[33] Zangwei Zheng, Xiangyu Yue, Kai Wang, and Yang You. Prompt vision transformer for domain generalization. *arXiv preprint arXiv:2208.08914*, 2022. 1, 2, 3