PUFFERFISH : COMMUNICATION-EFFICIENT MODELS AT NO EXTRA COST

MLSys 2021

DML GROUP MEETING
12.10

OUTLINE

- Introduction
- Effective deep factorized network training
- Strategies for mitigating training accuracy
- Experiments

Introduction

- 通信开销导致数据并行的分布式训练无法达到最佳的加速效果
- 通信开销归因于频繁的梯度更新
- 模型更高的准确率也导致了参数规模的扩大，加剧了通信开销
- 现有解决办法：低精度训练、梯度稀疏化
- 梯度压缩存在的问题：

1) 梯度压缩的计算代价很大，例如对每个批次的梯度做SVD矩阵分解

2) 现有的梯度压缩方法要么没有充分利用，要么需要额外内存

3) 在现有的深度学习框架上整合这些梯度压缩方法需要大量的工作

Introduction

- 能否将梯度压缩步骤的纳入模型架构本身?
- 通信效率可以在没有额外成本的情况下得以提升
- 基于此想法设计了Pufferfish
- 设计思想

  1) 将模型架构分解，训练分解后的模型，一劳永逸

  2) 对分解后的模型采用额外的训练策略提高准确率

Effective deep factorized network training

Low-rank factorization for FC layers

- 两层的FC network可表示为：

$$h(x) = \sigma(W_1 \sigma(W_2 x))$$

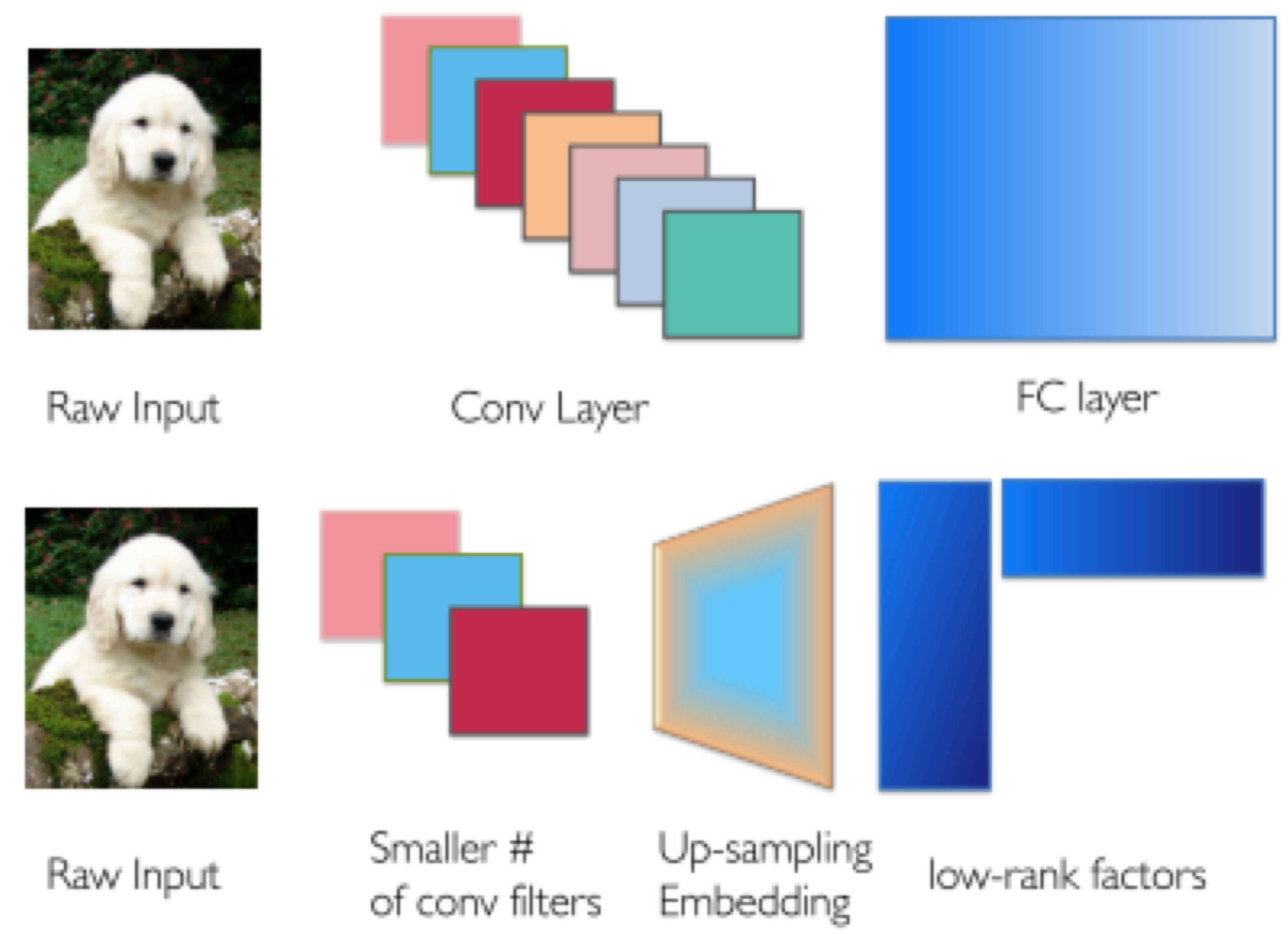
- 将每层的参数分解为

$$W_l \xrightarrow{\text{分解}} U_l V_l^T$$

Effective deep factorized network training

Low-rank factorization for convolution layers

- 原始参数形式：$W \in \mathbb{R}^{c_{\text{in}} \times c_{\text{out}} \times k \times k}$
- 整理：$W_{\text{unrolled}} \in \mathbb{R}^{c_{\text{in}} k^2 \times c_{\text{out}}}$
- 分解：$U \in \mathbb{R}^{c_{\text{in}} k^2 \times r}, V^\top \in \mathbb{R}^{r \times c_{\text{out}}}$
- 转换：$U \in \mathbb{R}^{c_{\text{in}} \times r \times k \times k}$ $V_l^\top \in \mathbb{R}^{r \times c_{\text{out}} \times 1 \times 1}$



Effective deep factorized network training

Low-rank factorization for LSTM layers

$$\begin{aligned}
i_t &= \sigma(\boxed{W_{ii}}x_t + b_{ii} + \boxed{W_{hi}}h_{t-1} + b_{hi}) \\
f_t &= \sigma(\boxed{W_{if}}x_t + b_{if} + \boxed{W_{hf}}h_{t-1} + b_{hf}) \\
g_t &= \tanh(\boxed{W_{ig}}x_t + b_{ig} + \boxed{W_{hg}}h_{t-1} + b_{hg}) \\
o_t &= \sigma(\boxed{W_{io}}x_t + b_{io} + \boxed{W_{ho}}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t).
\end{aligned} \tag{1}$$

分解 →

$$\begin{aligned}
i_t &= \sigma(\boxed{U_{ii}V_{ii}^\top}x_t + b_{ii} + \boxed{U_{hi}V_{hi}^\top}h_{t-1} + b_{hi}) \\
f_t &= \sigma(\boxed{U_{if}V_{if}^\top}x_t + b_{if} + \boxed{U_{hf}V_{hf}^\top}h_{t-1} + b_{hf}) \\
g_t &= \tanh(\boxed{U_{ig}V_{ig}^\top}x_t + b_{ig} + \boxed{U_{hg}V_{hg}^\top}h_{t-1} + b_{hg}) \\
o_t &= \sigma(\boxed{U_{io}V_{io}^\top}x_t + b_{io} + \boxed{U_{ho}V_{ho}^\top}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t).
\end{aligned} \tag{2}$$

Effective deep factorized network training

Low-rank network factorization for Transformer

- Multi-Head Attention：
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_p)W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

- FFN：
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- 可分解参数：
$$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{pd\times d} \quad W^O \in \mathbb{R}^{pd\times pd} \quad W_1 \in \mathbb{R}^{pd\times 4pd}, W_2 \in \mathbb{R}^{4pd\times pd}$$

Effective deep factorized network training

Computational complexity and model size

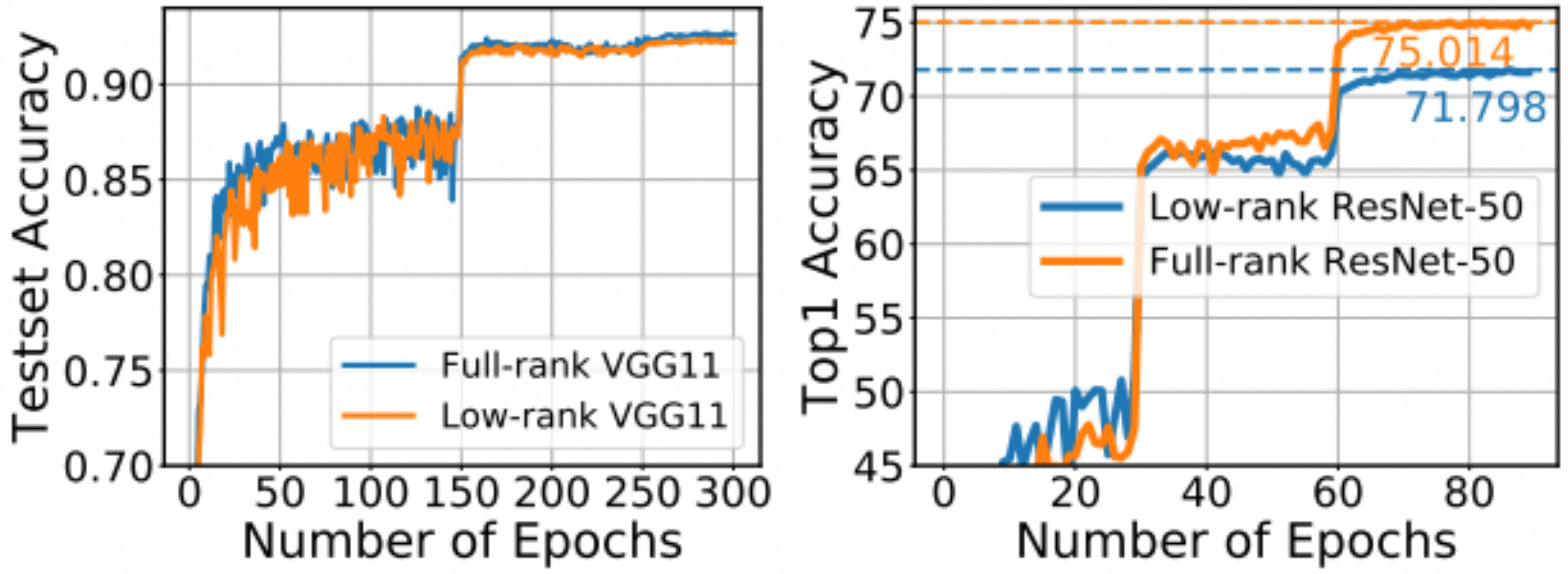
Networks	# Params.	Computational Complexity
Vanilla FC	$m \times n$	$\mathcal{O}(mn)$
Factorized FC	$r(m+n)$	$\mathcal{O}(r(m+n))$
Vanilla Conv.	$c_{\text{in}} \times c_{\text{out}} \times k^2$	$\mathcal{O}(c_{\text{in}} c_{\text{out}} k^2 HW)$
Factorized Conv.	$c_{\text{in}} r k^2 + r c_{\text{out}}$	$\mathcal{O}(r c_{\text{in}} k^2 HW + rHW c_{\text{out}})$
Vanilla LSTM	$4(dh + h^2)$	$\mathcal{O}(dh + h^2)$
Factorized LSTM	$4dr + 12hr$	$\mathcal{O}(dr + hr)$
Vanilla Attention	$4p^2d^2$	$\mathcal{O}(Np^2d^2 + N^2d)$
Factorized Attention	$(3p+5)prd$	$\mathcal{O}\big(rpdN + N^2d\big)$
Vanilla FFN	$8p^2d^2$	$\mathcal{O}\big(p^2d^2N\big)$
Factorized FFN	$10pdr$	$\mathcal{O}\big(rpdN\big)$

Strategies for mitigating training accuracy

- 模型分解一定程度上导致准确率下降
- 采取训练策略尽可能减少对准确率的影响

1) 混合网络架构 (hybrid network architecture)

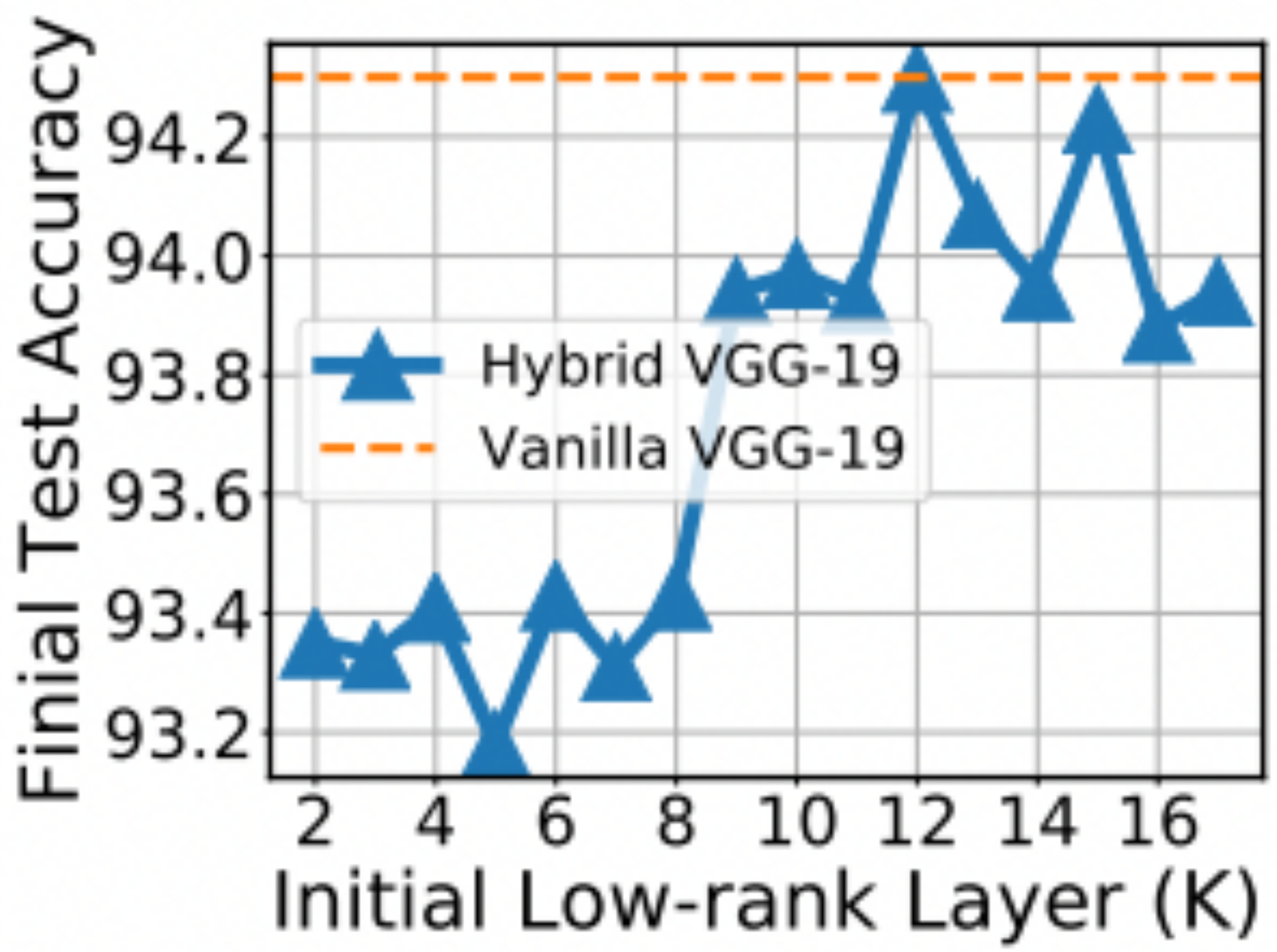
2) vanilla warm-up training



(a) VGG-11 on CIFAR-10 (b) ResNet-50 on ImageNet

hybrid network architecture

- 矩阵分解不可避免地带来了误差
- 误差会随着前向传播逐渐累积
- 深度神经网络中，后层的参数数量决定整个网络的规模
- 解决方案：仅对后层layer都做分解
- 注意：最后一层FC layer通常和最后的输出结果有关，最好不要分解该层
- 缺点：引入超参数K，需要对所有类型的layer调整K



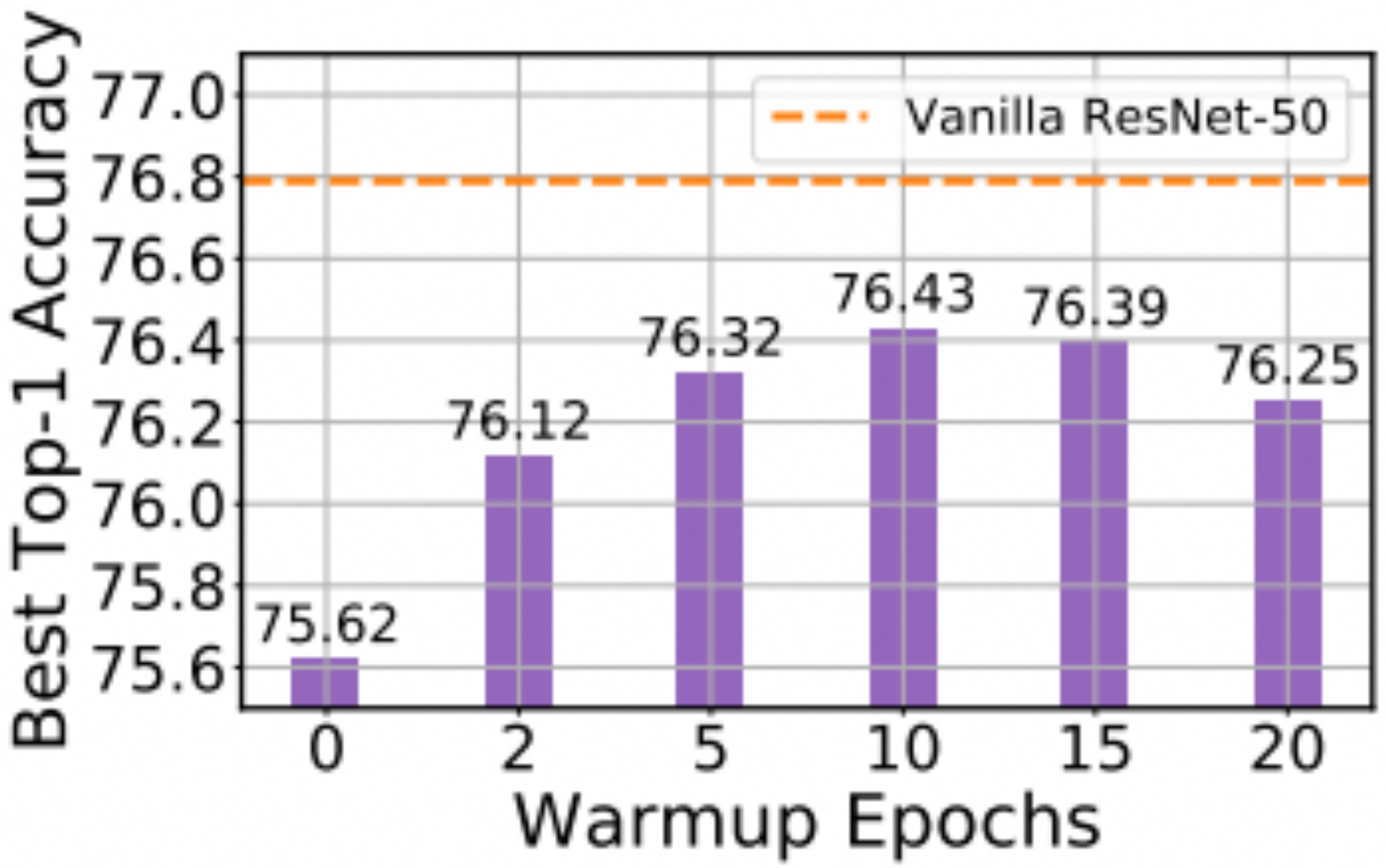
(a) Hybrid network

$$\{W_1, W_2, \cdots, W_L\} \;\not\rightarrow\; \{U_1, V_1^\top, U_2, V_2^\top, \cdots, U_L, V_L^\top\}$$

$$\{W_1, W_2, \cdots, W_L\} \rightarrow \{W_1, W_2, \cdots, W_{K-1}, U_K, V_K^\top, \cdots, U_L, V_L^\top\}$$

vanilla warm-up training

- 训练早期的epoch对最终模型的准确率至关重要
- 直接训练分解后的模型会导致精度损失，并且这种损失不能在后续训练阶段减轻
- 解决方案：使用部分训练的原始满秩模型来初始化分解后的低秩模型
- 缺点：引入超参数Warmup Epochs



(b) Vanilla warm-up training

Algorithm

Algorithm 1 PUFFERFISH Training Procedure

Input : Randomly initialized weights of vanilla N-layer architectures $\{W_1, W_2, \ldots, W_L\}$, and the associated weights of hybrid N-layer architecture $\{W_1, W_2, \ldots, W_{K-1}, U_K, V_K^\top, \ldots, U_L, V_L^\top\}$, the entire training epochs E, the vanilla warm-up training epochs E_{wu}, and learning rate schedule $\{\eta_t\}_{t=1}^{E}$

Output : Trained hybrid L-layer architecture weights $\{\hat{W}_1, \hat{W}_2, \ldots, \hat{W}_{K-1}, \hat{U}_K, \hat{V}_K^\top, \ldots, \hat{U}_L, \hat{V}_L^\top\}$

for $t \in \{1, \ldots, E_{wu}\}$ **do**

  Train $\{W_1, W_2, \ldots, W_L\}$ with learning rate schedule $\{\eta_t\}_{t=1}^{E_{wu}}$; `// vanilla warm-up training`

end

for $l \in \{1, \ldots, L\}$ **do**

  if $l < K$ **then**

    copy the partially trained W_l weight to the hybrid network;

  else

    $\tilde{U}_l \Sigma_l \tilde{V}_l^\top = \text{SVD}(W_l)$; `// Decomposing the vanilla warm-up trained weights`

    $U_l = \tilde{U}_l \Sigma_l^{\frac{1}{2}}, V_l^\top = \Sigma^{\frac{1}{2}} \tilde{V}_l^\top$

  end

end

for $t \in \{E_{wu}+1, \ldots, E\}$ **do**

  Train the hybrid network weights, *i.e.*, $\{W_1, W_2, \ldots, W_{K-1}, U_K, V_K^\top, \ldots, U_L, V_L^\top\}$ with learning rate schedule $\{\eta_t\}_{t=E_{wu}}^{E}$; `// consecutive low rank training`

end

Experiments

- Pufferfish可以训练得到一个比其他方法小3.35倍的模型
- 与POWERSGD、SIGNUM和Vanilla SGD相比，在CIFAR-10上训练的ResNet-18的端到端加速比分别为1.22倍、1.52倍和1.74倍，同时达到与Vanilla SGD相同的精度。
- 在ImageNet数据集上，与EB Train相比，Pufferfish模型的参数减少了1.3M，而Top-1测试精度提高了1.76%
- 与LTH相比，在CIFAR-10上为VGG-19实现相同的模型压缩率但加快了5.67倍

Conclusion

- 缺点：

1）引入超参数，r、K、Warmup Epochs，部分超参数需要按layer类型调整

2）每种模型如何进行模型分解需要提前分析，在完全了解模型后才能设计出好的分解方案

THANKS!