计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202330085
61(7): 1836−1849, 2024

# 一种基于特征导向解耦网络结构的滤波器修剪方法

施瑞文[1] 李光辉[1] 代成龙[1] 张飞飞[2]
[1](江南大学人工智能与计算机学院 江苏无锡 214122)
[2](江苏邦融微电子有限公司 江苏昆山 215300)
(srw0109@qq.com)

# Feature-Oriented and Decoupled Network Structure Based Filter Pruning Method

Shi Ruiwen[1], Li Guanghui[1], Dai Chenglong[1], and Zhang Feifei[2]
[1](*School of Artificial Intelligence and Computer Science*, *Jiangnan University*, *Wuxi*, *Jiangsu* 214122)
[2](*Jiangsu Bangrong Microelectronics Co.*, *Ltd.*, *Kunshan*, *Jiangsu* 215300)

**Abstract** Many existing pruning methods for deep neural network models require modifying the loss function or embedding additional variables in the network, thus they can't benefit from the pre-trained network directly, and complicate the forward inference and training process. So far, most of the feature-oriented pruning work only use the intra-channel information to analyze the importance of filters, which makes it impossible to use the potential connections among channels during the pruning process. To address these issues, we consider the feature-oriented filter pruning task from an inter-channel perspective. The proposed method uses geometric distance to measure the potential correlation among channels, defines filter pruning as an optimization problem, and applies a greedy strategy to find an approximate solution to the optimal solution. The method achieves the decoupling of pruning from network and pruning from training, thus simplifying the pruning task. Extensive experiments demonstrate that the proposed pruning method achieves high performance for various network structures, for example, on CIFAR-10 dataset, the number of parameters and floating point operations of VGG-16 are reduced by 87.1% and 63.7%, respectively, while still has an accuracy of 93.81%. We also evaluate the proposed method using MobileFaceNets, a lightweight network, on CASIA-WebFace large dataset, and the evaluation results show that, when the number of parameters and floating-point operations are reduced by 58.0% and 63.6%, respectively, MobileFaceNets achieves an accuracy of 99.02% on LFW dataset without loss of inference accuracy (The code is available at: https://github.com/SSriven/FOAD).
**Key words** deep learning; model compression; model pruning; neural network acceleration; geometric distance

**摘 要** 现有的很多深度神经网络模型剪枝方法需要修改损失函数或在网络中嵌入额外的变量,无法直接受益于预训练网络,而且复杂化了前向推理和训练过程.到目前为止,大部分特征导向的剪枝工作仅利用通道内信息分析滤波器的重要性,使得剪枝过程无法利用通道间的潜在联系.针对上述问题,基于特征导向从通道间的角度考虑滤波器修剪任务,使用几何距离度量通道间的潜在相关性,将滤波器修剪定义为一个优化问题,并引入贪婪策略寻求最优解的近似解.该方法实现了剪枝与网络、剪枝与训练的解耦,从而简化了修剪任务.大量的实验证明了该方法对于各种网络结构都有良好的性能,例如在CIFAR-10数据集上,将VGG-16的参数量和浮点运算量分别降低了87.1%和63.7%,并且达到93.81%的高精度.还使

**收稿日期**: 2023-02-16; **修回日期**: 2023-10-12
**基金项目**: 国家自然科学基金项目(62072216);苏州市科技计划项目(SGC2021070)
This work was supported by the National Natural Science Foundation of China(62072216)and the Science and Technology Program of Suzhou(SGC2021070).

用轻量型网络 MobileFaceNets 和 CASIA-WebFace 数据集评估该方法的性能，结果显示使用该剪枝方法后，MobileFaceNets 在参数量和浮点运算量分别降低 58.0% 和 63.6% 的情况下，在 LFW 上的测试精度仍然达到 99.02%，而且推理精度几乎没有损失（源代码发布在：https://github.com/SSriven/FOAD）.

**关键词** 深度学习；模型压缩；模型剪枝；神经网络加速；几何距离

**中图法分类号** TP18

深度神经网络具有计算密集型和内存密集型特点，这导致在计算资源匮乏的边缘设备（嵌入式设备、移动平台等）上部署庞大且复杂的深度神经网络（deep neural network, DNN）极具挑战[1]. 因此，必须使用模型压缩技术来减少模型的参数量和计算量，从而降低模型的复杂度，使 DNN 模型能够应用于边缘设备. 常见的模型压缩方法包括参数量化[2-7]、知识蒸馏[8-11]和模型剪枝[12-14]等. 其中，模型剪枝技术在各种模型压缩方法中展现出了巨大潜力，大致可以分为权重剪枝[15-17]和滤波器剪枝 2 种. 权重剪枝即从网络中删除一些对输出影响较低的神经元或相应权重，是一种针对单个权重的细粒度剪枝方法. 然而，权重剪枝得到的稀疏权值矩阵无法在通用硬件上实现理想的加速效果[18]. 相比之下，滤波器剪枝是针对滤波器级别的剪枝算法，修剪后的网络依旧具有良好的组织结构，不像权重剪枝需要特殊硬件或库来加速推理，在一般通用处理器上就可以轻松实现加速效果.

滤波器剪枝的主要目的是删减冗余和不重要的滤波器，以降低模型复杂度并且不影响模型性能. 在最近提出的一些滤波器剪枝方法中，使用“L-范数”稀疏损失[13,16-17,19]和可学习掩码矩阵[20-21]表现出良好的修剪效果. 然而，由于这些方法需要修改损失函数或在网络中嵌入修剪相关变量，因此无法直接受益于预训练网络. 此外，不再直接使用滤波器本身信息，而是利用特征信息来修剪滤波器逐渐成为普遍使用的策略. 正如文献 [22] 中所述，特征图反映了滤波器与输入数据之间的关系，比滤波器本身包含更加丰富的信息. 基于这个理念，提出了几种特征导向的滤波器剪枝算法[22-26]，它们在修剪任务上的表现比滤波器导向更优. 然而，到目前为止，一些特征导向方法仅利用通道内特征信息探讨滤波器的重要性，这可能会使得通道间的潜在关系在剪枝过程无法被注意到，相似特征可能会获得相同分数，导致冗余特征依旧存活. 实际上，如果正确使用通道间特征信息，可以为滤波器修剪提供比通道内更丰富的知识. 具体来说，有 2 个原因：1）跨通道修剪可以更好地探索不同特征图和相应滤波器之间的潜在相关性，从而获得更好的修剪性能；2）如果仅由通道内特征信息评估相应滤波器的重要性，则它可能对输入数据敏感，而跨通道特征信息更加稳定可靠[25].

总而言之，现有的很多剪枝方法与网络模型本身结合太过紧密，例如修改损失函数或在网络中加入额外的变量，这复杂化了修剪任务，不易使用. 此外，正如上文所述，基于特征导向的跨通道修剪比通道内修剪能够带来更多的好处. 本文提出了一种新的基于特征导向的跨通道滤波器剪枝方法 FOAD（feature-oriented and decoupled）. 该方法使用几何距离和贪心策略，从已训练好的密集高冗余网络中识别出稀疏低冗余子网络. 与以往不同的是，该方法不再根据重要性排名选择滤波器. 具体来讲，FOAD 计算同一层内特征图通道间的几何距离，根据几何距离的特性，使用几何距离度量通道间的相关性，基于贪婪策略移除在几何距离附近的滤波器. FOAD 消除了额外的辅助约束和嵌入变量，从而简化了修剪过程. 此外，FOAD 不需要修改损失函数，也无需了解训练细节，对于任何训练有素的网络都有较强适应性.

本文的主要贡献主要有 3 个方面：

1）提出了一种新的滤波器剪枝方法 FOAD，该方法基于几何距离跨通道度量特征图的相关性，以删除冗余的特征图和其对应的滤波器. 从通道间的角度考虑滤波器修剪可以更稳定、更可靠地探索冗余滤波器，从而为滤波器修剪提供更精确的指导.

2）FOAD 实现了训练与修剪解耦以及网络结构与修剪解耦，将滤波器修剪定义为一个优化问题，并在修剪过程引入贪婪策略寻求最优解的近似解，设计了一种高鲁棒性、低成本的冗余滤波器修剪方案.

3）使用轻量且紧凑的神经网络 MobileFaceNets 和公开数据集验证 FOAD 的性能. 实验结果表明 FOAD 产生较高的压缩比，在参数量和计算量分别降低 58% 和 63.6% 时，MobileFaceNets 在 LFW 数据集上依旧保持 99.02% 的高精度，并且推理性能几乎没有损失.

## 1 相关工作

滤波器修剪直接针对整个滤波器进行修剪，修剪过后的网络依然具有良好的组织结构，不需要专用的硬件进行加速推理. 现有的滤波器剪枝算法大致可以被分为滤波导向和特征导向2种类型.

1)滤波导向. 滤波导向的剪枝算法根据滤波器本身信息进行修剪. 例如在先前的工作中：Li等人[13]和He等人[19]移除掉范数低的滤波器；Liu等人[20]和Zhuang等人[27]使用BN层的参数$\gamma$作为评判通道的重要性程度；You等人[28]使用泰勒展开来估计全局滤波器的重要性排序. 一些文献通过研究滤波器之间的关系进行修剪：He等人[29]根据几何中值进行滤波器剪枝；Dubey等人[30]构造滤波器的核心集，用较小的核心集逼近原始集；Wang等人[31]将被删除的滤波器权重参数集成到存活的滤波器上. 也有研究者考虑跨层修剪，例如：Luo等人[32]根据下一层的输出信息剪枝当前层的卷积核；He等人[33]通过基于LASSO回归的通道选择和最小二乘重建来有效地剪枝每一层. 此外，Zhuang等人[34]在网络中引入额外的识别感知损失，以提高中间层的识别能力；Meng等人[21]将一个滤波器拆分成多个1×1的滤波器；Li等人[35]中提出了一种将滤波器剪枝与矩阵低秩分解融合的方法；Tiwari等人[36]引入预算约束进行滤波器修剪；Chen等人[37]将可训练参数划分为零不变组，并设计了一种新的优化算法. 上述方法着眼于找到更重要的滤波器修剪卷积神经网络(convolution neural network, CNN). 有研究者从寻找最优网络结构的角度去修剪CNN，例如：Lin等人[38]提出了通道剪枝算法ABCPruner，他们认为网络结构比权重更重要，因此该算法是寻找每层的通道数，而不像以往算法着眼于选择更重要的通道；Dong等人[39]提出可转换架构搜索，其目标是搜索网络的最佳大小，通过最小化计算成本来调整网络. Edouard等人[40]提出了一种新的标量DNN权重分布密度的自适应哈希方法来合并冗余神经元；Liu等人[41]从修剪可塑性角度分析修剪效果，在渐进修剪过程中重新生成神经元.

2)特征导向. 正如文献[22]中所述，特征图反映了卷积核与输入信息之间的关系，比滤波器本身包含更加丰富的信息. 基于这个理念，近年学者们提出了基于特征导向的剪枝算法，例如Lin等人[22]认为具有低秩的特征图包含的信息更少，滤波器相对不太重要，基于这个原则，提出利用一种基于高秩特征图的滤波器剪枝方法. Tang等人[23]引入仿冒特征作为对照组，以减少潜在的不相关因素的干扰，帮助挖掘冗余滤波器. Suau等人[24]提出PFA算法，通过对特征图进行PCA来确定各层的剪枝比例，通过对特征图进行相关性分析来评估滤波器的重要性. Sui等人[25]从特征图通道间的角度出发，提出使用通道独立性来执行高效的滤波器修剪，独立程度低的特征图被认为包含的信息较少，相对应的通道可以被移除. 此外Jiang等人[26]利用图卷积神经网络(graph convolutional neural network, GCN)和强化学习进行修剪CNN，该方法对特征图进行处理之后将其作为GCN的输入，然后使用强化学习训练GCN，由于要训练GCN和CNN，这需要大量的时间成本.

特征导向是一种数据驱动的剪枝算法，可能对输入数据较为敏感，与此相反的是滤波导向直接使用模型参数评估滤波器，不需要再输入数据进行前向传播，修剪结果稳定. 然而特征导向方法获取到的信息更加丰富，对于评判滤波器的重要性提供了更强有力的依据. 本文所提出的方法属于特征图导向，与Sui等人[25]方法类似，FOAD也是从特征图通道间的角度出发，不同的是Sui等人删除令特征图核范数变化最小的特征所对应的滤波器，而FOAD删除与某一个特征相关性最强的若干个特征所对应的滤波器，因此可以保留更多不同特征，同时删除冗余特征.

## 2 本文方法FOAD的介绍

### 2.1 符号说明

给定一个已训练好的CNN模型，该CNN模型包含$L$个卷积层，$L^i$表示第$i$层.$L^i$的参数可以表示为$\boldsymbol{W}_i=(\boldsymbol{w}_i^1,\boldsymbol{w}_i^2,\cdots,\boldsymbol{w}_i^{n_i})$，其中$n_i$表示第$i$层的滤波器数量，$\boldsymbol{w}_i^j\in\mathbb{R}^{n_{i-1}\times k_h\times k_w}$表示第$i$层的第$j$个滤波器，$k_h$和$k_w$分别表示滤波器的高和宽. 令$L^i$的输入为$X_i=(\boldsymbol{x}_i^1,\boldsymbol{x}_i^2,\cdots,\boldsymbol{x}_i^N)$，其中$N$表示输入的样本数，$\boldsymbol{x}_i^p\in\mathbb{R}^{n_{i-1}\times h_i\times w_i}$表示第$p$个样本，$h_i$和$w_i$分别表示第$i$层的输入图像的高和宽. 令第$i$层的输出特征图为$\boldsymbol{Y}_i=(\boldsymbol{y}_i^1,\boldsymbol{y}_i^2,\cdots,\boldsymbol{y}_i^N)$，$\boldsymbol{y}_i^p\in\mathbb{R}^{n_i\times h_{i+1}\times w_{i+1}}$，其中$\boldsymbol{y}_i^p=\boldsymbol{x}_{i+1}^p$. 经过修剪之后，$\boldsymbol{W}_i$被分成2个子集$R_i$和$K_i$. $R_i=\{r_i^1,r_i^2,\cdots,r_i^{n_{ir}}\}$表示要移除的滤波器索引集合，$r_i^j$表示第$i$层中要移除的第$j$个滤波器的索引. $K_i=\{k_i^1,k_i^2,\cdots,k_i^{n_{ik}}\}$表示要保留的滤波器索引集合，$k_i^j$表示第$i$层中要保留的第$j$个滤波器的索引. 其中，$n_{ir}+n_{ik}=n_i$，$R_i\cap K_i=\varnothing$，$R_i\cup K_i=\{1,2,\cdots,n_i\}$. 本文使用

模型参数量(*params*)和浮点运算量(*FLOPs*)评估模型的大小和计算开销，下文中的修剪率 *Pr* 指的是减少的 *FLOPs* 与原始 *FLOPs* 的比值，*Pr* 的数学公式为

$$Pr = 1 - \frac{FLOPs_{\text{pruned}}}{FLOPs_{\text{original}}}, \tag{1}$$

其中 $FLOPs_{\text{pruned}}$ 与 $FLOPs_{\text{original}}$ 分别表示修剪后与修剪前的浮点运算量.

### 2.2 方法论述

滤波器剪枝的目标是找到一个滤波器子集，使该子集对 CNN 的影响忽略不计. 本文要找到子集 $R_i$ 和 $K_i$，$R_i$ 中的滤波器所产生的特征图的几何距离逼近 $K_i$ 中的滤波器. 换句话说，本文的目的是尽可能多地保留不同特征，移除冗余特征，故可将滤波器剪枝问题定义为优化问题：

$$\min_{R_i,K_i} \sum_{i=0}^{L-1} \sum_{m=0}^{N-1} \sum_{p\in R_i} \sum_{q\in K_i} \varphi(\boldsymbol{y}_i^{m,p}, \boldsymbol{y}_i^{m,q}), \tag{2}$$

其中 $\varphi(\cdot,\cdot)$ 用来计算 2 张特征图之间的几何距离，表示相关性. $\boldsymbol{y}_i^{m,c} \in \mathbb{R}^{h_{i+1}\times w_{i+1}}$ 表示第 $i$ 层输出特征图中第 $m$ 个特征图的第 $c$ 通道. 在本文中 $\varphi(\cdot,\cdot)$ 被定义为

$$\varphi(\boldsymbol{y}_i^{m,p}, \boldsymbol{y}_i^{m,q}) = \sqrt{\left\|\boldsymbol{y}_i^{m,p} - \boldsymbol{y}_i^{m,q}\right\|_{\text{F}}^2}. \tag{3}$$

其中 $\|\cdot\|_{\text{F}}$ 表示 F-范数. 为了将几何距离映射到 (0, 1) 区间，通道间的几何距离越接近 1 表示距离越近，相关性也就越强. 式(3)可以表示为

$$\varphi(\boldsymbol{y}_i^{m,p}, \boldsymbol{y}_i^{m,q}) = \frac{1}{1 + \sqrt{\left\|\boldsymbol{y}_i^{m,p} - \boldsymbol{y}_i^{m,q}\right\|_{\text{F}}^2}}. \tag{4}$$

式(4)将几何距离映射到(0,1)区间. 然而特征图对输入图像比较敏感，需要同时输入多个图像，然后计算所有输入图像所产生的特征图中相同通道间的几何距离的平均值. 因此式(4)可以进一步被修改为

$$\varphi^i(p,q) = \frac{1}{1 + \dfrac{1}{N}\displaystyle\sum_{m=0}^{N-1} \sqrt{\left\|\boldsymbol{y}_i^{m,p} - \boldsymbol{y}_i^{m,q}\right\|_{\text{F}}^2}}, \tag{5}$$

其中 $\varphi^i(p,q)$ 表示第 $i$ 层输出特征图第 $p$ 与 $q$ 通道之间的几何距离. 此外，当 $p=q$ 时，按照式(5)计算的 $\varphi^i(p,q)=1$，为了避免通道自身的影响，本文约束当 $p=q$ 时 $\varphi^i(p,q)=0$. 几何距离的公式变为：

$$\psi^i(p,q) = \begin{cases} \varphi^i(p,q), p \neq q, \\ 0, p = q. \end{cases} \tag{6}$$

最终滤波器修剪优化问题可以表示为

$$\max_{R_i,K_i} \sum_{i=0}^{L-1} \sum_{p\in R_i} \sum_{q\in K_i} \psi^i(p,q). \tag{7}$$

然而求解式(7)是一个 NP-Hard 问题，因此本文采用贪婪策略解决式(7)所描述的优化问题，即 FOAD 每次选择一个通道，并在当前迭代中移除与该通道相关性最强的 $t$ 个通道和其对应的滤波器. 显然，使用贪婪策略得到的解是次优解，但是这种差距可以通过微调来弥补.

经过求解式(7)，得到集合 $R_i$ 和 $K_i$，$R_i$ 中的滤波器可以被安全地移除. 此外，还要考虑 2 个问题：首先，有些通道所表示的特征冗余度较低，它们与其他通道的相关性都很弱，即使相关性最强的通道，它们之间的几何距离也是一个极低的数值(例如小于 0.1). 其次，先被保存到 $K_i$ 中的通道也可能与其他通道的相关性相对较强. 如果贸然地移除相关性弱或者早已被保存到 $K_i$ 中的通道，可能会导致网络学习到的特征变少，从而使得网络的分类性能下降. 为了解决上述 2 个问题，引入一个阈值 $s$ 和一个约束条件，即相关性大于 $s$ 的通道并且该通道不属于 $K_i$ 才可以被安全地移除，被移除的通道也不再参与后续计算. 图 1 展示了修剪 1 层的大致过程，其中 $t$=1, $s$=0，每次移除 1 个通道，被移除的通道不再参与后续的相关性计算，被保留的通道则不被移除.

FOAD 修剪过程大致为：

Step 1. 取出 1 个通道.

Step 2. 如果该通道已经被移除，则转到 Step1，否则将该通道放入集合 $K$ 中.

Step 3. 使用式(6)计算该通道与其他所有通道之间的几何距离，若某个通道已被移除，则跳过这个通道，最终得到该通道的相关性集合.

Step 4. 对该通道的相关性集合降序排序.

Step 5. 移除掉相关性集合中前 $t$ 个不在集合 $K$ 中，且几何距离大于等于阈值 $s$ 的通道.

Step 6. 转至 Step1 直至所有通道都已计算完.

算法 1 展示了详细的 FOAD 修剪细节.

**算法 1.** FOAD.

输入：特征图 $\boldsymbol{Y}_i$，每次修剪去除的滤波器个数 $t$，相似度阈值 $s$；

输出：修剪之后保留的卷积核集 $K$；

初始化：$c$ = 特征图通道数，$U=\{0,1,\cdots,c\}$，$R=\{\}$，$K=\{\}$.

① for $i=0 \rightarrow c-1$ do

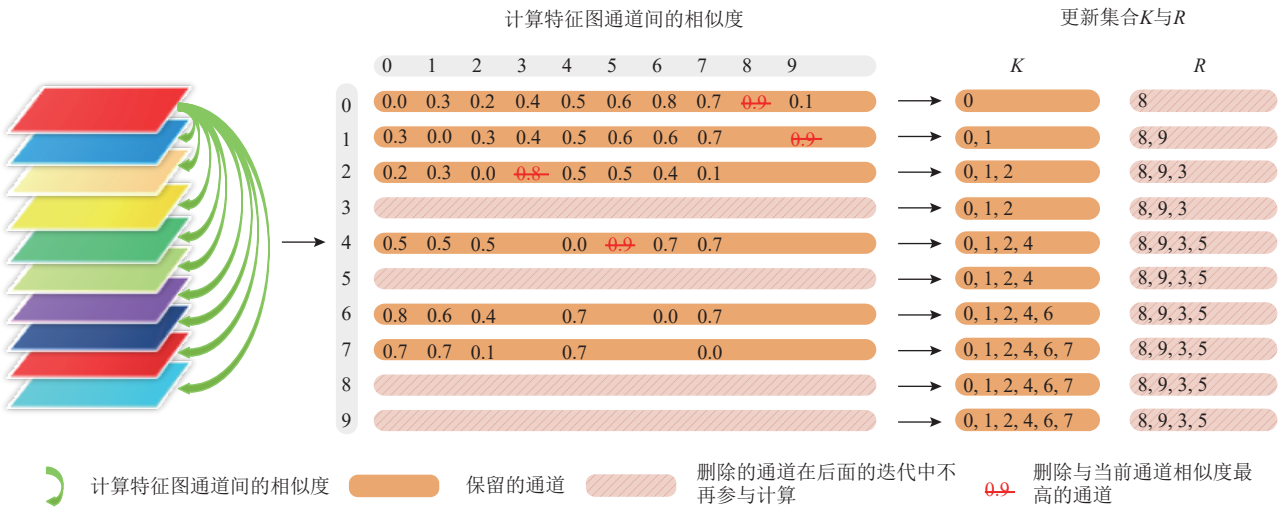
② if(第 $i$ 通道已经被移除了) then

③ continue;

④ end if

⑤ *K.append*($i$); /*将第 $i$ 通道放入集合 $K$ 中*/

⑥ *Scores* = {};




注：计算特征图每个通道之间的几何距离，若某个通道在前面的迭代中已被移除，那么就不参与后续的计算. 例如第 8 通道最先被移除，那么后续的通道不再与第 8 通道计算，后续也不会被移除，例如第 0 通道和第 1 通道.

Fig. 1 Greedy pruning of the filter based on the geometric distance between feature channels

图 1 根据特征通道间的几何距离贪婪地修剪滤波器

⑦ for $j = 0 \rightarrow c-1$ do
⑧ 使用式(6)计算 $i$ 和 $j$ 通道的几何距离 *score*;
⑨ 将几何距离 *score* 与索引 $j$ 放入集合 *Scores* 中;
⑩ end for
⑪ 对集合 *Scores* 按照几何距离进行降序排序;
⑫ for $k = 1 \rightarrow t$ do
⑬ 取出集合 *Scores* 中第 $k$ 个元素的通道索引 *index* 以及对应的几何距离 *score*;
⑭ if(如果通道 *index* 不在集合 $K$ 中，且 几何距离 *score* 大于等于阈值 $s$) then
⑮ *R.append*(*index*); /*移除通道 *index*/
⑯ *U*[*index*] = none; /*在 $U$ 中移除*/
⑰ end if
⑱ end for
⑲ end for
⑳ return *K*.

### 2.3 算法鲁棒性分析

2.2 节详细阐述了 FOAD 剪枝算法的主要思想，其中修剪过程中的一个关键点在于特征图通道间几何距离的计算，然而特征图可能对输入图像比较敏感，因为对于不同的输入图片，产生的特征图也不同，特征图通道之间的几何距离也就不同，这就无法保证对同一网络每次剪枝后产生的网络结构的一致性，这一问题为 FOAD 带来了巨大的挑战. 然而，本文在实验中观察到，单个卷积层对不同的输入图片剪枝之后保留的滤波器集具有鲁棒性. 为了说明这一点，本文设置了 Batch Size 分别为 16, 32, 64, 128, 512 的输入图片，且所有数据均随机打乱. 令 $u$ 为 2 个集合的交并比，使用式(8)计算不同 Batch Size 的输入数据经过剪枝之后保留的卷积核集的交并比:

$$u = \frac{a \cap b}{a \cup b}, \tag{8}$$

其中 $a$ 和 $b$ 分别是输入不同 Batch Size 的图片集所产生的滤波器集.图 2 展示了不同 Batch Size 之间的交并比. 可以清楚地看到，即使输入不同的数据，经过 FOAD 之后，每个卷积层所保留的滤波器集的交并比都大于 0.65，且大部分都在 0.75 以上. 换句话说，即使输入不同的图片，经过 FOAD 之后所保留的滤波器有 75% 以上是相同的，这表明，FOAD 仅需要一小部分输入数据即可高效且准确地完成剪枝任务. 为了进一步验证我们的观点，本文在 3.9 节中针对不同 Batch Size 对模型性能的影响做了详细分析.

### 2.4 修剪策略

通过 2.2 节和 2.3 节，我们阐述了跨通道修剪 CNN 的算法 FOAD 的主要思想和特点，算法 1 展示了修剪 1 层卷积层的详细过程. 对 CNN 中每一层卷积层都执行算法 1 之后得到了一个稀疏的网络，再对该网络微调以恢复精度. 本文考虑了 2 种修剪策略，即一次性修剪和渐进式修剪. 一次性修剪指的是只修剪 1 次达到目标修剪率；渐进式修剪指的是每次修剪一部分，修剪、微调迭代 $n$ 次直至达到目标修剪率，渐进式修剪在工业场景下是一种普遍使用的修剪策略. 文献 [16] 证明了渐进式修剪比一次性修剪

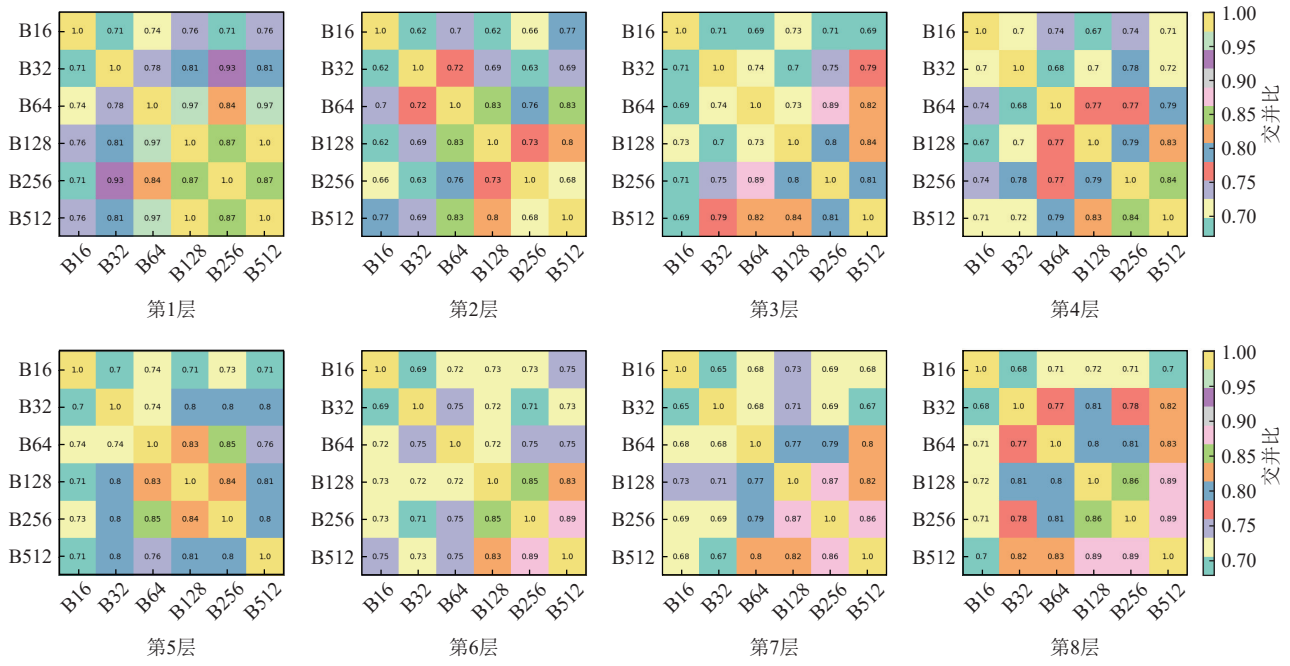



Fig. 2 Robustness analysis

图 2 鲁棒性分析

能够更容易找到比较好的子网络. 本文在实验中也表明了, 渐进式修剪比一次性修剪效果更好.

## 3 实 验

### 3.1 数据集和基线模型

为了验证方法 FOAD 的有效性和实用性, 基于不同的基线模型以及不同的图像分类数据集, 评估 FOAD 的修剪性能. 具体来说, 本文在 CIFAR-10 和 CIFAR-100[42] 数据集上对 3 种 CNN(VGG-13, VGG-16, ResNet-56)进行了实验. CIFAR-10 数据集包含 10 个类别, 总共有 60 000 张图片, 其中训练集 50 000 张, 测试集 10 000 张. CIFAR-100 数据集则包含 100 个类别, 训练集和测试集的图片数量与 CIFAR-10 相同. 此外, 为了更进一步验证 FOAD 的有效性, 针对更大规模的 CASIA-WebFace 数据集在 MobileFaceNets[43] 上进行剪枝实验, 并在 LFW 数据集[44] 上进行评估. CASIA-WebFace 数据集包含了 10 575 个人的 494 414 张图片.

### 3.2 实验设置

本文使用 Pytorch-1.11 框架实现 FOAD, 并在 NVIDIA RTX 3050Ti GPU 上进行实验. 对于基线模型的训练, 在 CIFAR-10 和 CIFAR-100 数据集上训练 160 个周期, 并选择随机梯度下降作为优化器, Batch Size、初始学习率、动量和权值衰减系数分别为 64, 0.1, 0.9, 1E−4.此外, 在第 80 和 120 个周期将学习率分别设为 0.01 和 0.001.在 CASIA-WebFace 数据集上, 参考文献 [43] 中的训练策略进行基线模型的训练, 使用随机梯度下降优化器, 训练总周期为 60, Batch Size 设为 128, PReLU 的权值衰减系数为 0, MobileFaceNets 中最后一层全连接层的权值衰减系数设为 4E−4, 其他层的权值衰减系数为 4E−5, 初始学习率和动量分别设为 0.1 和 0.9, 并且在训练迭代为第 25 000, 36 000, 48 000 次时, 将学习率分别调整为 0.01, 0.001, 0.0001. 对于微调过程, 采用的训练策略和基线模型的训练策略一致. 为了测试模型剪枝性能, 使用模型参数量和浮点运算量评估模型的大小和计算开销, 并且将剪枝之后的精度与基线模型精度进行比较. 本文还比较了一次性修剪和渐进式修剪的差异性, 为了区别, 我们用 FOAD-G(gradual, G)表示渐进式修剪, FOAD-O(once, O)表示一次性修剪. 对于 VGG-16, 取 $t$=1, $s$=0.3 进行渐进式修剪; 对于 ResNet-56, 考虑到残差块的特殊性, 较小的 $t$ 会导致修剪率太低, 需要迭代修剪多次才能达到目标修剪率, 故在初次修剪时取 $t$=3, $s$=0.2, 在后续的渐进式修剪中取 $t$=1, $s$=0 对 ResNet-56 进行渐进式修剪.

### 3.3 在 CIFAR-10 数据集上的评估结果

1)VGG-16. 使用 VGG-16 在 CIFAR-10 数据集上训练, 再对训练好的 VGG-16 进行修剪和微调. 原始的 VGG-16 由 13 个卷积层和 3 个全连接层组成, 由

于本文方法主要针对卷积层，因此在本文中使用的VGG-16均只有1个全连接层. 修剪结果如表1所示.

VGG-16在CIFAR-10数据集上的基线精度为93.84%，当修剪掉53.2%的参数量和37.3%的浮点运算量后，微调之后的精度能够达到94.00%，这比基线模型精度还要更高. 当修剪掉70.0%的参数量和45.9%的浮点运算量，微调之后的精度能够达到93.82%，相比基线模型的精度，仅仅下降了0.02%. 与具备同级别修剪率的其他方法相比，FOAD也展现出了一定的优势，例如L1, FPGM, SSS, GAL等方法，在修剪率为

**Table 1 Pruning Results of VGG-16 and ResNet-56 on CIFAR-10 Dataset**

**表1 VGG-16和ResNet-56在CIFAR-10数据集上的修剪结果**

| 模型 | 方法 | 精度/% | *params* | *params.drop*/% | *FLOPs* | *FLOPs.drop*/% |
|---|---|---|---|---|---|---|
| VGG-16 | L1[13]（ICLR 2017） | 93.40 | $9.69\times10^6$ | 34.2 | $0.412\,80\times10^9$ | 34.2 |
| | SSS[45]（ECCV 2018） | 93.02 | $3.86\times10^6$ | 73.8 | $0.366\,38\times10^9$ | 41.6 |
| | GAL[46]（CPVR 2019） | 93.42 | $2.62\times10^6$ | 82.2 | $0.343\,79\times10^9$ | 45.2 |
| | FPGM[29]（CVPR 2019） | 93.54 | | | $0.412\,80\times10^9$ | 34.2 |
| | Hinge[35]（CVPR 2020） | 93.59 | $2.93\times10^6$ | 80.1 | $0.382\,06\times10^9$ | 39.1 |
| | FOAD-O（$t$=1，$s$=0.3） | **94.00** | **$6.89\times10^6$** | **53.2** | **$0.393\,31\times10^9$** | **37.3** |
| | FOAD-O（$t$=2，$s$=0.4） | **93.82** | **$4.42\times10^6$** | **70.0** | **$0.339\,32\times10^9$** | **45.9** |
| | ThiNet[32]（ICCV 2017） | 90.76 | $5.30\times10^6$ | 64.0 | $0.225\,85\times10^9$ | 64.0 |
| | NS[20]（ICCV 2017） | 93.80 | $1.69\times10^6$ | 88.5 | $0.307\,41\times10^9$ | 51.0 |
| | HRank[22]（CVPR 2020） | 92.34 | $2.63\times10^6$ | 82.1 | $0.217\,69\times10^9$ | 65.3 |
| | Di[26]（IJCAI 2022） | 93.27 | $2.13\times10^6$ | 85.5 | $0.267\,88\times10^9$ | 57.3 |
| | FOAD-G | **93.81** | **$1.90\times10^6$** | **87.1** | **$0.227\,78\times10^9$** | **63.7** |
| | FOAD-O（$t$=3，$s$=0.3） | **93.36** | **$2.73\times10^6$** | **81.5** | **$0.246\,84\times10^9$** | **60.7** |
| | COP[31]（IJCAI 2019） | 93.31 | $1.06\times10^6$ | 92.8 | $0.166\,25\times10^9$ | 73.5 |
| | ABCPruner[38]（IJCAI 2020） | 93.08 | $1.66\times10^6$ | 88.7 | $0.164\,99\times10^9$ | 73.7 |
| | OTO[37]（NeurIPS 2021） | 93.30 | $0.81\times10^6$ | 94.5 | $0.168\,13\times10^9$ | 73.2 |
| | CHIP[25]（NeurIPS 2021） | 93.18 | $1.87\times10^6$ | 87.3 | $0.134\,25\times10^9$ | 78.6 |
| | Di[26]（IJCAI 2022） | 93.08 | $1.02\times10^6$ | 93.1 | $0.168\,76\times10^9$ | 73.1 |
| | FOAD-G | **93.41** | **$0.74\times10^6$** | **95.0** | **$0.178\,80\times10^9$** | **71.5** |
| | FOAD-O（$t$=3，$s$=0.2） | **93.08** | **$2.56\times10^6$** | **82.6** | **$0.165\,62\times10^9$** | **73.6** |
| ResNet-56 | L1[13]（ICLR 2017） | 93.06 | $0.59\times10^6$ | 27.6 | $0.212\,32\times10^9$ | 13.7 |
| | PFA[24]（CVPR 2018） | 92.49 | $0.49\times10^6$ | 40.4 | $0.151\,30\times10^9$ | 38.5 |
| | NISP[47]（CVPR 2018） | 93.01 | $0.47\times10^6$ | 42.4 | $0.158\,68\times10^9$ | 35.5 |
| | GAL[46]（CPVR 2019） | 92.98 | $0.72\times10^6$ | 11.8 | $0.163\,52\times10^9$ | 37.6 |
| | FOAD-O（$t$=3，$s$=0.2） | **93.50** | **$0.40\times10^6$** | **51.2** | **$0.151\,86\times10^9$** | **38.3** |
| | He[33]（ICCV 2017） | 91.80 | | | $0.123\,01\times10^9$ | 50.0 |
| | FOAD-G | **93.06** | **$0.23\times10^6$** | **72.0** | **$0.118\,09\times10^9$** | **52.0** |
| | FOAD-O（$t$=3，$s$=0.1） | **92.59** | **$0.38\times10^6$** | **53.7** | **$0.123\,01\times10^9$** | **50.0** |
| | CHIP[25]（NeurIPS 2021） | 92.05 | $0.23\times10^6$ | 71.8 | $0.068\,15\times10^9$ | 72.3 |
| | FOAD-G | **92.45** | **$0.21\times10^6$** | **74.4** | **$0.076\,27\times10^9$** | **69.0** |
| | FOAD-O（$t$=4，$s$=0） | **91.20** | **$0.29\times10^6$** | **64.6** | **$0.078\,97\times10^9$** | **67.9** |
| | HRank[22]（CVPR 2020） | 90.72 | $0.26\times10^6$ | 68.1 | $0.063\,72\times10^9$ | 74.1 |
| | FilterSketch[48]（TNNLS 2021） | 91.20 | $0.23\times10^6$ | 71.8 | $0.062\,98\times10^9$ | 74.4 |
| | FOAD-G | **91.34** | **$0.15\times10^6$** | **81.7** | **$0.053\,14\times10^9$** | **78.4** |
| | FOAD-O（$t$=5，$s$=0） | **90.38** | **$0.24\times10^6$** | **70.7** | **$0.064\,95\times10^9$** | **73.6** |

注：黑体数值表示本文方法的实验结果；“*drop*”表示降低.

30%~50%时，它们微调后的精度都比FOAD要低. 在修剪率为50%以上时，FOAD同样具备一些优势，例如当参数量和浮点运算量降低87.1%和63.7%时，微调之后的精度为93.81%，相较于GAL，微调之后的精度更高，浮点运算量降低更多. 相较于NS，浮点运算量降低得更多，微调后的精度也比NS高，而对比HRank，在参数量和浮点运算量降低得差不多的情况下，它们微调之后的精度只有92.34%，而FOAD的精度是93.81%. 对于更高的修剪率，即浮点运算量降低至70%以上时，FOAD在微调之后的精度能够达到93.41%，对比同级别修剪率的其他方法，FOAD精度也更高. 例如，CHIP将VGG-16的参数量和浮点运算量分别降低87.3%和78.6%时，修剪后的模型精度为93.08%. 而FOAD将VGG-16的参数量和浮点运算量分别降低95.0%和71.5%时，修剪后的模型精度为93.41%，即使浮点运算量降低没有CHIP多，但是参数量的降低(95.0%对比87.3%)和修剪后的模型精度(93.41%)对比CHIP(93.08%)有较大优势. 这是因为FOAD能够有效地去除冗余的通道，且使用了更细粒度的修剪策略(渐进式修剪)，渐进式修剪在高剪枝率下可以发挥很好的作用. 而CHIP在剪枝率比较高的时候，容易误删一些对模型贡献大的通道，导致模型性能下降. 通过这些实验数据也证明了FOAD能够有效地加速和压缩CNN，合理地利用特征图通道间的相关性更有助于识别冗余的特征以及滤波器.

2)ResNet-56. ResNet-56由多个残差块堆叠而成，由于残差块中存在短连接操作，因此不能像修剪VGG-16一样修剪ResNet-56.在修剪ResNet-56时，对于残差块中的最后一层不进行修剪，如果修剪了该层将无法进行短连接操作. 修剪结果如表1所示，ResNet-56在CIFAR-10数据集上训练的基线模型精度为93.31%，FOAD将ResNet-56的参数量和浮点运算量分别降低了51.2%和38.3%，微调之后的精度比基线模型精度提升了0.19个百分点(93.50%)，对比L1，PFA，NISP，GAL，FOAD有更高的修剪率和精度. 在浮点运算量减少50.0%以上时，对比同级别修剪率的方法，FOAD仍然表现出了更优异的性能. 例如当参数量和浮点运算量分别减少81.7%和78.4%时，微调之后的精度能够达到91.34%，对比FilterSketch和HRank，FOAD有更高的剪枝率和精度. 从实验结果来看，FOAD对于具有短连接操作的网络模型也能够有效地修剪，且表现出了较佳的性能.

### 3.4 在CIFAR-100数据集上的评估结果

1)VGG-16. CIFAR-100数据集有更多的类别，分类难度更大，本文使用VGG-16在CIFAR-100上进行了评估，评估结果如表2所示，VGG-16在CIFAR-100上的基线模型精度为73.12%. FOAD与一些先进的方法在参数量和浮点运算量以及微调之后的精度这3个评价指标上进行了比较.FOAD可以将VGG-16的参数量和浮点运算量分别减少79.4%和48.0%，而精度没有下降太多.COP将VGG-16的参数量和浮点运算量分别降低了73.2%和43.1%，在精度上从72.59%降低至71.77%，而FOAD在精度上从73.12%提升至

**Table 2 Pruning Results of VGG-16 and ResNet-56 on CIFAR-100 Dataset**

**表2 VGG-16和ResNet-56在CIFAR-100数据集上的修剪结果**

| 模型 | 方法 | 精度 | *params.drop* | *FLOPs.drop* |
|---|---|---|---|---|
| VGG-16 | PFA[24]（CVPR 2018） | 70.00 | 66.9 | 42.9 |
| | COP[31]（IJCAI 2019） | 71.77 | 73.2 | 43.1 |
| | DPES[49]（2021） | 67.06 | | 19.9 |
| | CHIP[25]（NeurIPS 2021） | 72.15 | 39.9 | 43.0 |
| | Di[26]（IJCAI 2022） | 72.00 | 80.4 | 56.5 |
| | FOAD-G | **73.30** | **79.4** | **48.0** |
| | FOAD-O（$t$=3，$s$=0.3） | **73.21** | **72.9** | **44.0** |
| ResNet-56 | PFA[24]（CVPR 2018） | 69.22 | 18.5 | 20.6 |
| | PFA[24]（CVPR 2018） | 68.05 | 26.4 | 33.3 |
| | DPES[49]（IS 2021） | 57.81 | | 16.19 |
| | CHIP[25]（NeurIPS 2021） | 69.00 | 38.2 | 41.4 |
| | PGMPF[50]（AAAI 2022） | 70.21 | | 52.6 |
| | FOAD-O（$t$=2，$s$=0.1） | **71.00** | **43.4** | **42.2** |

注：黑体数值表示本文方法FOAD的实验结果；“*drop*”表示降低.

73.30%, 不管是精度下降的幅度还是微调之后的精度(73.30% 对比 71.77%, +0.18% 对比−0.82%), FOAD 都要比 COP 更具优势. 与 PFA 相比, FOAD 也有更高的参数量下降(−79.4% 对比−66.9%)、浮点运算量下降(−48.0% 对比−42.9%)以及更高的精度(73.30% 对比 70.00%). 与 CHIP 相比, 在参数量和浮点运算量降低得相近时, FOAD 具有更高的精度(73.30% 对比 72.00%).

2)ResNet-56. 与此同时, 本文也使用 ResNet-56 在 CIFAR-100 数据集上进行了评估, ResNet-56 在 CIFAR-100 上的基线模型精度为 71.58%, 评估结果展示在表 2 中. 对比 PFA 的 *FLOPs*, 其分别下降了 20.6% 和 33.3%, FOAD 将 *FLOPs* 降低了 42.2%, 微调之后的精度也更高(71.00% 对比 69.22%, 68.05%). 对比 CHIP, PGMPF, DPES 这 3 种方法, FOAD 也有更大的优势. 这进一步证明了 FOAD 对于类别更多的数据集也具有一定的效果.

### 3.5 在 CASIA-WebFace 数据集上的评估结果

MobileFaceNets 是一个在人脸识别领域被广泛使用的卷积神经网络, 它具有优秀的特征提取能力, 并且是一个适合部署在嵌入式设备上的轻量型的网络, 模型大小仅仅只有 3.99 MB. 然而在一些计算资源十分有限的嵌入式设备端, 该模型依旧比较大, 随意裁剪网络的通道数和网络深度, 将极大地影响网络的特征提取能力, 因此可以使用 FOAD 自动修剪 MobileFaceNets.MobileFaceNets 由多个瓶颈块组成, 每个瓶颈块由 3 个卷积层组成, 由于瓶颈块中存在短连接操作, 因此每个瓶颈块中的最后一层卷积层在本文中不执行剪枝操作, 而瓶颈块中的第 2 个卷积是深度卷积, 深度卷积的输入通道和输出通道必须一致, 故在修剪时瓶颈块中的第 1 层卷积和第 2 层深度卷积共享同一个 $K$.

修剪结果如表 3 所示, 基线模型在 LFW 上的精度为 99.17%. 在参数量和浮点运算量分别下降 29% 和 34% 时, 在 LFW 上的精度仅仅下降了 0.03 个百分点(99.17% 对比 99.14%). 即使将模型压缩至 1.75 MB、*FLOPs* 降低 63.6% 以及参数量减少 58%, 在这种极致修剪下, 微调之后的模型在 LFW 上的测试精度依然能够达到 99.02%. 因此, FOAD 在复杂数据集上也有较好的效果, 而且也说明了 FOAD 对于有瓶颈块的紧凑型网络可以很好地工作.

### 3.6 渐进式修剪与一次性修剪的对比分析

与此同时, 本文在相同的修剪率下对比了渐进式修剪和一次性修剪的效果, 对比结果展示在表 1 和表 2 中. 可以清楚地看到, 渐进式修剪比一次性修剪能够更好地恢复精度, 例如在表 1 中, 当将 VGG-16 的浮点运算量同样降低 60% 左右时, 渐进式修剪比一次性修剪的精度更高(93.81% 对比 93.36%). 在浮点运算量降低 70% 左右时, 渐进式修剪也有更好的表现(93.41% 对比 93.08%), 这种情况在对 ResNet-56 进行修剪时也可以观察到. 同样地, 在表 2 中也有类似的结果. 渐进式修剪追求的是精细地修剪, 而一次性修剪追求一步到位. 精细地修剪意味着时间成本更高, 但是能够得到更高的精度, 而一次性修剪则与之相反. 上述对比表明 FOAD 适用于渐进式修剪策略, 并且进一步验证了在高剪枝率下渐进式修剪有更高的性能.

### 3.7 FOAD 变体分析

本文对 FOAD 的变体进行了研究, 提出了 2 种变体来证明保留特征多样性的同时移除冗余特征是合理的.2 种变体包括: 1)反向. 保留相关性高的特征, 移除相关性低的特征. 2)随机. 修剪之后随机初始化参数从头训练. 为了保证公平性, 在消融实验中均使用一次性修剪, 且令 FOAD 以及其变体的修剪率保持在同一水平线, 阈值 $s$=0.FOAD 将 VGG-16 在 CIFAR-10 上的浮点运算量降低了 72.6%, 反向和随机 2 种变体减少了 64.7% 的浮点运算量. 本文在图 3 中报告了各种变体的修剪效果, 可以看到, FOAD 明显比其变体的性能更优. 根据分析, 反向变体降低了特征的多样性, 保留了更多相关性高的特征. 这说明了特征多样性有益于网络分类, 也体现出 FOAD 的优越性.

**Table 3 Pruning Results of MobileFaceNets on CASIA-WebFace Dataset**

**表 3 MobileFaceNets 在 CASIA-WebFace 数据集上的修剪结果**

| 模型 | LFW 精度/% | *params* | *params.drop*/% | *FLOPs* | *FLOPs.drop*/% |
|---|---|---|---|---|---|
| MobileFaceNets | 99.17 | $1.00\times10^6$ | 0 | $0.38\times10^9$ | 0 |
| FOAD-O ($t$=1, $s$=0) | 99.14 | $0.71\times10^6$ | 29.0 | $0.25\times10^9$ | 34.0 |
| FOAD-O ($t$=2, $s$=0) | 99.07 | $0.54\times10^6$ | 46.0 | $0.18\times10^9$ | 52.6 |
| FOAD-O ($t$=3, $s$=0) | 99.02 | $0.42\times10^6$ | 58.0 | $0.14\times10^9$ | 63.6 |

注: "*drop*"表示降低.

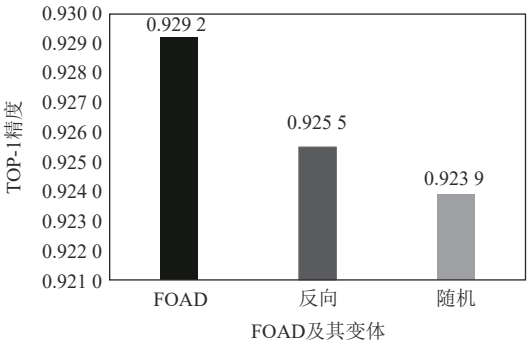



Fig. 3 Pruning results for FOAD and its variants

图 3 FOAD 及其变体的剪枝结果

### 3.8 对比不同度量特征图相似性的方法

我们想要探索不同的度量方法对剪枝后的模型精度是否有影响，本节主要对比几何距离和余弦相似度 2 种度量相似性的方法. 实验结果如表 4 所示，在 CIFAR-10 和 CIFAR-100 两个数据集上，使用几何距离进行修剪都要比余弦相似度更好. 这是因为几何距离更加注重多维空间中各个点之间的绝对距离，而余弦相似度更加注重 2 个向量在方向上的差异. 在通道数比较大的网络中存在大量冗余的、相似的特征图，使用几何距离度量相似度会更加合适.

### 3.9 不同 Batch Size 对模型剪枝性能的影响分析

在 2.3 节中，本文对输入不同 Batch Size 的数据进行剪枝产生的滤波器集合的交并比做了简单分析. 在图 2 中清楚地看到，即使输入不同的图片，经过 FOAD 之后所保留的滤波器有 75% 以上是相同的. 为了进一步验证，本节针对不同 Batch Size 在 VGG-16 和 CIFAR-10 上进行了相关实验，将 Batch Size 分别设置为 16, 32, 64, 128, 256, 512. 为了保证各实验组的公平性，每组实验中 $t$=1, $s$=0. 实验结果如表 5 所示，无论 Batch Size 取多少，在参数量和浮点运算量几乎相同的情况下，剪枝后模型的精度相差不大. 由此可以得出结论：本文方法 FOAD 对于数据是鲁棒的，即不管输入多少数据，FOAD 都可以高效地修剪网络.

### 3.10 超参数 $t$ 和 $s$ 的分析

FOAD 存在 2 个可调节的超参数 $t$ 和 $s$，用来控制修剪率. $t$ 越大，压缩率越高；而 $s$ 越大，压缩率越低. $s$ 的作用是抑制过度修剪. 在剪枝的过程中，可以根据压缩率的需求调整超参数 $t$ 和 $s$，不同的超参数 $t$ 和 $s$ 组合所产生的修剪效果也不一样. 为此，本文使用不同的超参数组合，采用控制变量法进行剪枝实验，实验结果如图 4 所示，图左侧表示固定 $s$=0.3 时 $t$ 分别取 1, 2, 3, 4，右侧表示固定 $t$=2 时 $s$ 分别取 0, 0.1, 0.2, 0.3. 其中图 4(a)展示的是 VGG-13 在 CIFAR-10 上的剪枝结果，图 4(b)展示的是 VGG-16 在 CIFAR-100 上的剪枝结果. 可以看到，随着 $t$ 的增大，参数量和浮点运算量减少越多，精度损失越大；而随着 $s$ 的增大，参数量和浮点运算量减少越少，精度损失越小. 经过观察发现，参数量和浮点运算量的减少对于 $t$ 的变化更加敏感，这是因为 $s$ 的作用是抑制过度修剪，当修剪率太高而导致精度下降太多时，可以适当加入 $s$ 来抑制某些过度修剪. 例如图 4(a)右侧所示，当

**Table 4 Methods for Comparing the Similarity of Different Metric Feature Maps**

**表 4 对比不同度量特征图相似性的方法**

| 数据集 | 度量方法 | 精度/% | *params* | *params.drop*/% | *FLOPs* | *FLOPs.drop*/% |
|---|---|---|---|---|---|---|
| CIFAR-10 | 余弦相似度 | 93.49 | $6.19\times10^6$ | 57.9 | $0.29\times10^9$ | 53.8 |
| | 几何距离（本文） | **93.52** | **$6.79\times10^6$** | **53.9** | **$0.30\times10^9$** | **52.3** |
| CIFAR-100 | 余弦相似度 | 71.26 | $4.35\times10^6$ | 70.5 | $0.20\times10^9$ | 68.1 |
| | 几何距离（本文） | **71.67** | **$4.75\times10^6$** | **67.8** | **$0.18\times10^9$** | **70.8** |

注：黑体数值表示本文方法 FOAD 的实验结果；“*drop*”表示降低.

**Table 5 Analysis of the Effect of Different Batch Sizes on Model Pruning Performance**

**表 5 不同 Batch Size 对模型剪枝性能的影响分析**

| Batch Size | 精度/% | *params* | *FLOPs* |
|---|---|---|---|
| 16 | 93.69 | $6.74\times10^6$ | $0.297\,93\times10^9$ |
| 32 | 93.56 | $6.58\times10^6$ | $0.292\,00\times10^9$ |
| 64 | 93.52 | $6.79\times10^6$ | $0.299\,29\times10^9$ |
| 128 | 93.48 | $6.89\times10^6$ | $0.303\,72\times10^9$ |
| 256 | 93.70 | $6.76\times10^6$ | $0.302\,35\times10^9$ |
| 512 | 93.46 | $6.78\times10^6$ | $0.301\,25\times10^9$ |

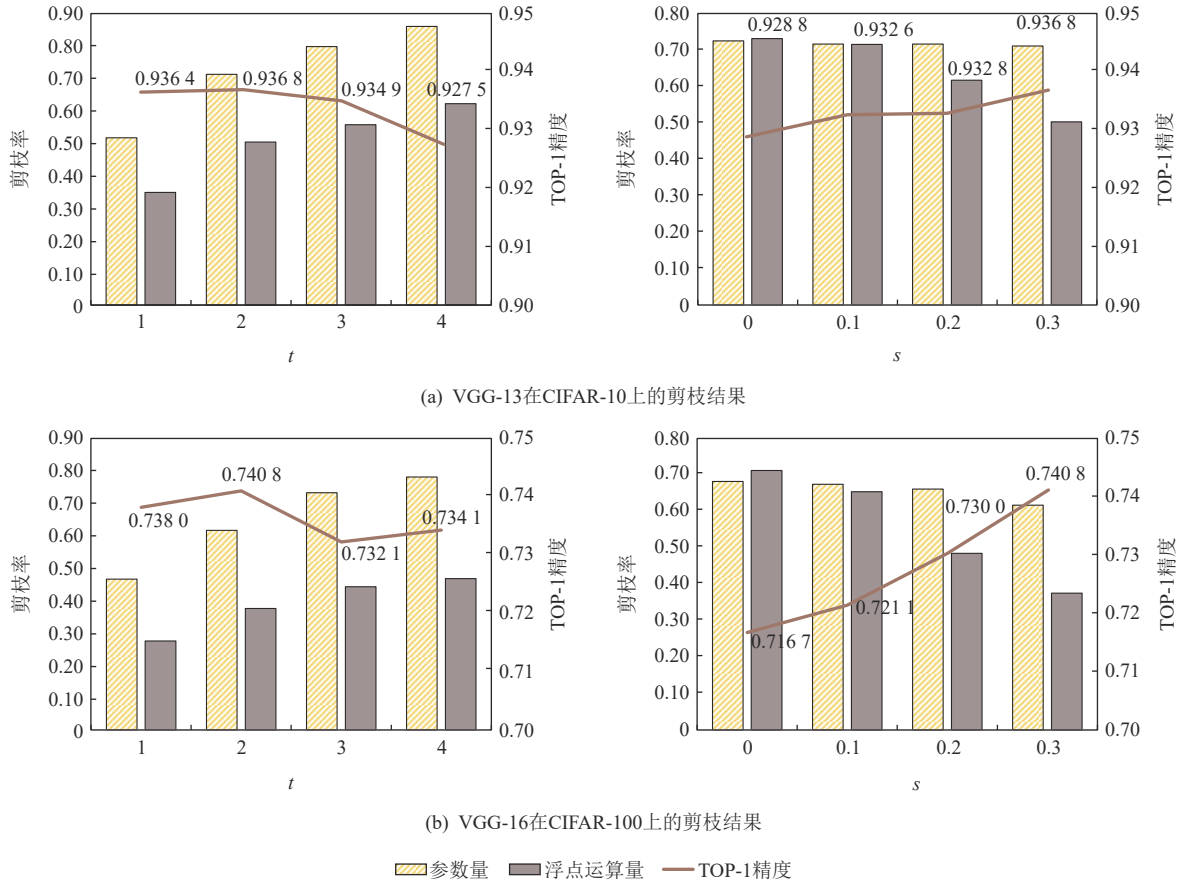



Fig. 4 Analysis experimental results of hyperparameters *t* and *s*

图 4 超参数 *t* 和 *s* 的分析实验结果

不对 *t* 进行限制, 即 *s*=0 时, 参数量和浮点运算量都减少了 70%, 而精度却下降到 92.88%. 当对 *t* 进行限制, 例如 *s*=0.3 时, 微调后的精度可以达到 93.68%. 这个现象也可以在图 4(b)中观察到. 经过大量的实验发现, 当 $t\in\{2,3\}$, $s\in\{0.2,0.3\}$时, 可达到较好的修剪效果.

## 4 结 论

本文提出了一种新的滤波器修剪方法 FOAD. 该方法充分利用了特征通道间的潜在信息, 通过研究特征通道间的潜在相关性来评估滤波器的冗余程度, FOAD 将网络结构和训练过程与修剪过程进行解耦, 极大地简化了剪枝任务. 此外, 通过实验观察到, 单个卷积层对不同的输入图片修剪之后保留的卷积核集具有鲁棒性. 在不同数据集以及不同网络结构上的大量实验结果表明, 本文提出的方法在显著降低模型的计算成本和存储成本的同时, 几乎不损失模型精度, 证明了渐进式修剪比一次性修剪更具有优势.

**作者贡献声明:** 施瑞文提出了算法思路和实验方案, 完成实验并撰写论文; 李光辉和代成龙提出了指导意见并修改论文; 张飞飞参与了实验验证.

## 参 考 文 献

[1] Pohlen T, Hermans A, Mathias M, et al. Full-Resolution residual networks for semantic segmentation in street scenes [C] // Proc of the 2017 IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2017: 3309−3318

[2] Dettmers T. 8-bit approximations for parallelism in deep learning[J]. arXiv preprint, arXiv: 1511.04561, 2016

[3] Hwang K, Sung W. Fixed-point feedforward deep neural network design using weights +1, 0, and −1[C] // Proc of the 2014 IEEE Workshop on Signal Processing Systems. Piscataway, NJ: IEEE, 2014: 174−179

[4] Courbariaux M, Bengio Y, David J. BinaryConnect: Training deep neural networks with binary weights during propagations [C] // Proc of the Annual Conf on Neural Information Processing Systems 2015. Piscataway, NJ: IEEE 2015: 3123−3131

[5] Courbariaux M, Bengio Y. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1 [J]. arXiv

preprint, arXiv: 1602.02830, 2016

[6] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks [C] // Proc of the 14th European Conf on Computer Vision. Berlin: Springer, 2016: 525−542

[7] Gong Cheng, Lu Ye, Dai Surong, et al. Ultra-low loss quantization method for deep neural network compression[J]. Journal of Software, 2021, 32(8): 2391−2407 (in Chinese)
(龚成, 卢冶, 代素蓉, 等. 一种超低损失的深度神经网络量化压缩方法[J]. 软件学报, 2021, 32(8): 2391−2407)

[8] Romero A, Ballas N, Kahou S, et al. FitNets: Hints for thin deep nets[J]. arXiv preprint, arXiv: 1412.6550, 2015

[9] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint, arXiv: 1503.02531, 2015

[10] Zhang Jing, Wang Ziming, Ren Yonggong. A3C deep reinforcement learning model compression and knowledge extraction[J]. Journal of Computer Research and Development, 2023, 60(6): 1373−1384 (in Chinese)
(张晶, 王子铭, 任永功. A3C 深度强化学习模型压缩及知识抽取[J]. 计算机研究与发展, 2023, 60(6): 1373−1384)

[11] Lin Zhenyuan, Lin Shaohui, Yao Yiwu, et al. Multi-teacher contrastive knowledge inversion for data-free distillation[J/OL]. Journal of Frontiers of Computer Science and Technology, 2022[2023-09-13]. http://fcst.ceaj.org/CN/10.3778/j.issn.1673-9418.2204107 (in Chinese)
(林振元, 林绍辉, 姚益武, 等. 多教师对比知识反演的无数据模型压缩方法[J/OL]. 计算机科学与探索, 2022[2023-09-13]. http://fcst.ceaj.org/CN/10.3778/j.issn.1673-9418.2204107)

[12] Han Song, Mao Huizi, William J, et al. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding[J]. arXiv preprint, arXiv: 1510.00149, 2016

[13] Li Hao, Kadav A, Durdanovic I, et al. Pruning filters for efficient ConvNets [C/OL] // Proc of the 5th Int Conf on Learning Representations. Berlin: Springer, 2017[2023-09-13]. https://openreview.net/forum?id=rJqFGTslg

[14] Lin Tao, U. Stich S , Barba L, et al. Dynamic model pruning with feedback [C/OL] // Proc of the 8th Int Conf on Learning Representations. Berlin: Springer, 2020[2023-09-13]. https://openreview.net/forum?id=SJem8lSFwB

[15] Zhu M, Gupta S. To prune, or not to prune: Exploring the efficacy of pruning for model compression [C/OL] // Proc of the 6th Int Conf on Learning Representations. Berlin: Springer, 2018[2023-09-13]. https://openreview.net/forum?id=Sy1iIDkPM

[16] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks [C/OL]// Proc of the 7th Int Conf on Learning Representations. Berlin: Springer, 2019[2023-09-13]. https://openreview.net/forum?id=rJl-b3RcF7

[17] Guo Yiwen, Yao Anbang, Chen Yurong. Dynamic network surgery for efficient DNNs [C]// Proc of the Annual Conf on Neural Information Processing Systems 2016. Piscataway, NJ: IEEE, 2016: 1379−1387

[18] Han Song, Liu Xingyu, Mao Huizi, et al. EIE: Efficient inference engine on compressed deep neural network [C] // Proc of the 43rd ACM/IEEE Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 243−254

[19] He Yang, Kang Guoliang, Dong Xuanyi, et al. Soft filter pruning for accelerating deep convolutional neural networks [C] // Proc of the 27th Int Joint Conf on Artificial Intelligence. Berlin: Springer, 2018: 2234−2240

[20] Liu Zhuang, Li Jianguo, Shen Zhiqiang, et al. Learning efficient convolutional networks through network slimming [C]// Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2017: 2755−2763

[21] Meng Fanxu, Cheng Hao, Li Ke, et al. Pruning filter in filter [C/OL] // Proc of the Annual Conf on Neural Information Processing Systems 2020. Piscataway, NJ: IEEE, 2020[2023-09-13]. https://proceedings.neurips.cc/paper/2020/hash/ccb1d45fb76f7c5a0bf619f979c6cf36-Abstract.html

[22] Lin Mingbao, Ji Rongrong, Wang Yan, et al. HRank: Filter pruning using high-rank feature map [C] // Proc of the 2020 IEEE/CVF Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 1526−1535

[23] Tang Yehui, Wang Yunhe, Xu Yixing, et al. SCOP: Scientific control for reliable neural network pruning [C/OL] // Proc of the Annual Conf on Neural Information Processing Systems 2020. Piscataway, NJ: IEEE, 2020[2023-09-13]. https://proceedings.neurips.cc/paper/2020/hash/7bcdf75ad237b8e02e301f4091fb6bc8-Abstract.html

[24] Suau X, Zappella L, Apostoloff N, et al. Network compression using correlation analysis of layer responses [C/OL] // Proc of the 2018 IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ:IEEE,2018[2023-09-13].https://readpaper.com/pdf-annotate/note?noteId=1959569544894668800

[25] Sui Yang, Yin Miao, Xie Yi, et al. CHIP: Channel independence-based pruning for compact neural networks [C] // Proc of the Annual Conf on Neural Information Processing Systems 2021. Piscataway, NJ: IEEE , 2021: 24604−24616

[26] Jiang Di, Cao Yuan, Yang Qiang. On the channel pruning using graph convolution network for convolutional neural network acceleration[C] // Proc of the 21st Int Joint Conf on Artificial Intelligence. Berlin: Springer, 2022: 3107−3113

[27] Zhuang Tao, Zhang Zhixuan, Huang Yuheng, et al. Neuron-level structured pruning using polarization regularizer [C/OL] // Proc of the Annual Conf on Neural Information Processing Systems 2020. Piscataway, NJ: IEEE, 2020[2023-09-13]. https://proceedings.neurips.cc/paper/2020/hash/703957b6dd9e3a7980e040bee50ded65-Abstract.html

[28] You Zhonghui, Yan Kun, Ye Jinmian , et al. Gate Decorator: Global filter pruning method for accelerating deep convolutional neural networks [C] // Proc of the Annual Conf on Neural Information Processing Systems 2019. Piscataway, NJ: IEEE, 2019: 2130−2141

[29] He Yang , Liu Ping , Wang Ziwei , et al. Filter pruning via geometric median for deep convolutional neural networks acceleration [C] // Proc of the Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2019: 4340−4349

[30] Dubey A, Moitreya C, Ahuja N, et al. Coreset-based neural network compression [C] // Proc of the 15th European Conf Computer Vision. Piscataway, NJ: IEEE, 2018: 469−486

[31] Wang Wenxiao, Cong Fu, Guo Jishun, et al. COP: Customized deep model compression via regularized correlation-based filter-level pruning [C] // Proc of the 28th Int Joint Conf on Artificial Intelligence. Berlin: Springer, 2019: 3785−3791

[32] Luo Jianhao, Wu Jianxin, Lin Weiyao, et al. ThiNet: A filter level pruning method for deep neural network compression [C] // Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2017: 5068−5076

[33] He Yihui, Zhang Xiangyu, Sun Jian. Channel pruning for accelerating very deep neural networks [C] // Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2017: 1398−1406

[34] Zhuang Zhuangwei, Tan Mingkui, Zhuang Bohan, et al. Discrimination-aware channel pruning for deep neural networks [C] // Proc of the Annual Conf on Neural Information Processing Systems 2018. Piscataway, NJ: IEEE, 2018: 883−894

[35] Li Yawei, Gu Shuhang, Mayer C, et al. Group sparsity: The hinge between filter pruning and decomposition for network compression [C] // Proc of the 2020 IEEE/CVF Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 8015−8024

[36] Tiwari R, Bamba U, Chavan A, et al. ChipNet: Budget-aware pruning with heaviside continuous approximations [C/OL] // Proc of the 9th Int Conf on Learning Representations. Berlin: Springer, 2021[2023-09-13]. https://openreview.net/forum?id=xCxXwTzx4L1

[37] Chen Tianyi, Ji Bo, Ding Tianyu, et al. Only train once: A one-shot neural network training and pruning framework [C] // Proc of the Annual Conf on Neural Information Processing Systems 2021. Piscataway, NJ: IEEE, 2021: 19637−19651

[38] Lin Mingbao, Ji Rongrong, Zhang Yuxin, et al. Channel pruning via automatic structure search [C] // Proc of the 29th Int Joint Conf on Artificial Intelligence. Berlin: Springer, 2020: 673−679

[39] Dong Xuanyi, Yang Yi. Network pruning via transformable architecture search [C] // Proc of the Annual Conf on Neural Information Processing Systems 2019. Piscataway, NJ: IEEE, 2019: 759−770

[40] Edouard Y, Arnaud D, Matthieu C, et al. RED : Looking for redundancies for data-free structured compression of deep neural networks[J]. arXiv preprint, arXiv: 2105.14797, 2021

[41] Liu Shiwei, Chen Tianlong, Chen Xiaohan, et al. Sparse training via boosting pruning plasticity with neuroregeneration [C] // Proc of the Annual Conf on Neural Information Processing Systems 2021. Piscataway, NJ: IEEE, 2021: 9908−9922

[42] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J/OL]. Handbook of Systemic Autoimmune Diseases, 2009[2023-09-13]. https://xueshu.baidu.com/usercenter/paper/show?paperid=c55665fb879e98e130fce77052d4c8e8&site=xueshu_se

[43] Chen Sheng, Liu Yang, Gao Xiang, et al. MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices [C] // Proc of the 13th Chinese Conf on Biometric Recognition. Berlin: Springer, 2018: 428−438

[44] Huang G, Mattar M, Berg T, et al. Labeled faces in the wild: A database for studying face recognition in unconstrained environments [C/OL] // Proc of the Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition. 2008[2023-09-13]. https://cs.brown.edu/courses/csci1430/2011/proj4/papers/lfw.pdf

[45] Huang Zehao, Wang Naiyan. Data-driven sparse structure selection for deep neural networks [C] // Proc of the 15th European Conf on Computer Vision. Piscataway, NJ: IEEE, 2018: 317−334

[46] Lin Shaohui, Ji Rongrong, Yan Chenqian, et al. Towards optimal structured CNN pruning via generative adversarial learning [C] // Proc of the IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2019: 2790−2799

[47] Yu Ruichi, Li Ang, Chen Chunfu, et al. NISP: Pruning networks using neuron importance score propagation [C] // Proc of the 2018 IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2018: 9194–9203

[48] Lin Mingbao, Cao Liujuan, Li Shaojie, et al. Filter sketch for network pruning[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 33(12): 7091−7100

[49] Fernandes F, Yen G. Pruning deep convolutional neural net-works architectures with evolution strategy [C/OL] // Proc of the Information Sciences. Amsterdam: Elsevier, 2021[2023-09-13].https://doi.org/10.1016/j.ins.2020.11.009

[50] Cai Linhang, An Zhulin, Yang Chuanguang, et al. Prior gradient mask guided pruning-aware fine-tuning [C/OL] // Proc of the AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2022[2023-09-13]. http://dx. doi.org/10.1609/aaai.v36i1.19888

**Shi Ruiwen**, born in 1999. Master. His main research interests include model compression and deep learning.

**施瑞文**, 1999 年生. 硕士. 主要研究方向为模型压缩、深度学习.

**Li Guanghui**, born in 1970. PhD, professor and PhD supervisor. Senior member of CCF. His main research interests include wireless sensor network, model compression and intelligent nondestructive detection technology.

**李光辉**, 1970 年生. 博士，教授，博士生导师. CCF 高级会员. 主要研究方向为无线传感网、模型压缩、智能无损检测技术.

**Dai Chenglong**, born in 1992. Lecturer. His main research interests include electroencephalogram processing, electroencephalogram analyzing, and model compression.

**代成龙**, 1992 年生. 讲师. 主要研究方向为脑电图处理、脑电图分析、模型压缩.

**Zhang Feifei**, born in 1982. Master. His main research interests include the hardware acceleration implementation of image processing algorithm and SoC chip design.

**张飞飞**, 1982 年生. 硕士. 主要研究方向为图像处理算法的硬件加速、SoC 芯片设计.