树莓派深度学习模型优化

Salem Ameen 科学、工程与环境学院 索尔福德大学 曼彻斯特，英国
s.a.ameen1@salford.ac.uk

Kangaranmulle Siriwardana 科学、工程与环境学院 索尔福德大学 曼彻斯特，英国
k.p.s.k.siriwardana@edu.salford.ac.uk

Theo Theodoridis 科学、工程与环境学院 索尔福德大学 曼彻斯特，英国
t.theodoridis@salford.ac.uk

摘要— 深度学习模型已在众多应用领域日益普及，包括计算机视觉、自然语言处理和语音识别。然而，这些模型通常需要大量计算资源，因此难以在树莓派等低功耗设备上运行。解决这一挑战的一种方法是采用剪枝技术来缩小深度学习模型的规模。剪枝是指从模型中移除不重要的权重和连接，从而得到更小、更高效的模型。剪枝可以在训练过程中进行，也可以在模型训练完成后进行。另一种方法是针对树莓派架构专门优化深度学习模型。这包括优化模型的架构和参数，以充分利用树莓派的硬件能力，例如其中央处理器和 GPU。此外，还可以通过尽量减少所需的计算量来优化模型的能效。对树莓派上的深度学习模型进行剪枝和优化，有助于克服低功耗设备在计算能力和能耗方面的限制，使深度学习模型能够在更广泛的设备上运行。在接下来的章节中，我们将更详细地探讨这些方法，并讨论它们在优化树莓派深度学习模型方面的效果。

关键词—深度学习、卷积神经网络、剪枝、压缩、优化、树莓派。

I. 引言

近年来，在低功耗设备[1] 上运行深度学习模型变得越来越普遍。然而，树莓派等低功耗设备在计算资源和能效方面面临诸多挑战与限制。这些限制使得在不进行大量优化的情况下，很难将深度学习模型部署到低功耗设备上。优化低功耗设备上的深度学习模型的一种方法是剪枝，即从神经网络中移除不重要的权重。剪枝可以显著减少运行深度学习模型所需的计算资源，同时保持甚至提高其准确率。另一种方法是优化神经网络架构，即改变网络层的数量和排列方式，在保持准确率的同时降低计算需求。本文重点研究面向树莓派这一常用低功耗设备的深度学习模型剪枝与优化。我们研究剪枝对深度学习模型计算性能和能效的影响，并比较不同剪枝技术的性能。此外，我们还探讨优化树莓派神经网络架构的优势，并提出一种经过优化的适用于该设备的架构。实验结果表明，这些技术能够有效支持在低功耗设备上部署深度学习模型。

本研究旨在探讨剪枝和压缩优化技术对深度图像分类模型的影响，研究对象是在树莓派上运行实时推理的深度图像分类模型。

在以下章节中，我们将讨论在低功耗设备上优化深度学习模型的现有技术，包括剪枝。随后，我们将介绍实验设置，包括树莓派的硬件和软件配置。接下来，我们将详细说明所使用的数据集和深度学习模型架构，并解释剪枝过程以及其如何应用于深度学习模型。之后，我们将讨论实验结果并进行分析。最后，我们将结合文献综述对结果进行解读，说明研究发现对深度学习模型在低功耗设备上实际部署的影响，总结本文的主要贡献和研究发现，并阐述本研究对低功耗设备深度学习领域的意义。

II. 文献综述

以下综述介绍了不同的架构及其性能。研究人员借助 Raspberry Pi Model 3B+，使用树莓派在野外进行动物检测 [2] 。他们采用自定义卷积神经网络，该网络在 ImageNet [3] 和 VIPeR [4] 数据集上训练，并使用雪豹数据集重新训练，以识别野外的雪豹。对于预先下载的图像，模型识别雪豹的准确率达到 0.97；对于实时图像，准确率为 0.74。研究中报告的推理时间为处理 347 张图像用时 29.2 秒，即每张图像 84.15 毫秒。考虑到硬件成本和推理速度，结果颇有前景，因为该设备能够捕捉并识别实时视频。DeepEdgeBench [5] 讨论了多种用于深度神经网络模型推理的边缘设备的基准测试。DeepEdgeBench 对多款主流设备（片上系统）进行了基准测试，包括英伟达 Jetson Nano、Asus Tinker Edge R、Google Coral Dev、树莓派，以及 Arduino Nano 微控制器。测试使用 MobileNetV2、MobileNetV2 Lite、量化版 MobileNetV2 和量化版 MobileNetV1 模型，对 5000 张图像进行处理。每张图像的推理速度以及测试图像的 Top-1 准确率汇总于表1。

表1：DeepEdgeBench 的结果

模型	MobileNetV2	MobileNetV2 TF Lite	MobileNetV2 量化	MobileNetV1 量化
推理时间/图像（毫秒）	207.54	193.02	128	4.4
前1名准确率	0.733	0.733	0.726	0.364

文献[6]讨论了一种降低延迟和模型大小的方法：不使用较大的模型，而是采用神经网络树。该方法使用较小的架构对输入进行细分，并通过多个架构对其分类，直到输入归入特定类别。此过程可降低延迟、内存使用量和功耗，并已在树莓派 3 和树莓派 Zero 上进行测试。表2列出了在 CIFAR-100 数据集上使用树莓派 3 时不同模型的推理时间对比。

表2 [6] 获得的结果

模型	VGG-Pruned	CondenseNet	MNN-Tree
推理时间每张图像（毫秒）	*934*	*7196*	*506*

在 [7] 开展的测试中，研究人员使用了一种更简单的卷积神经网络，并采用树莓派 3 测量属于猫和狗类别的 100 张图像的推理时间。该模型是一个简单的 5 层卷积神经网络，训练了 90 个训练轮数。记录的每张图像推理时间为 16.37 毫秒。在 [8] 中，研究人员采用了类似的方法，使用 3 个卷积神经网络模型对包含 27 个类别的德国交通标志识别基准数据集进行分类。使用这些模型进行分类的准确率为 98.57%，并在树莓派 2B 上进行了测试。每张图像分类的平均时间为 0.72 秒。[9] 取得了更有前景的结果，其中使用卷积神经网络模型对从车辆品牌和型号识别数据集中提取的图像进行分类，该数据集包含 10 个类别。研究人员使用 InceptionV3、VGG16 和 MobileNet 架构训练了三个模型。测试硬件采用树莓派 3B+，并配备英特尔神经计算棒；后者是一种用于神经网络的硬件加速器。研究人员使用模型优化器对训练好的模型进行优化，以提高推理速度。

表 3 [9] 获得的结果

	InceptionV3	VGG16	MobileNet
模型加载时间（秒）	52	50	71

III. 方法

本节将讨论实验设置的描述，包括树莓派的硬件和软件配置、所使用数据集的详细信息、深度学习模型架构，以及对剪枝和优化过程的说明及其应用于深度学习模型的方式。

A. 树莓派

树莓派已发布多个版本，最新版本是树莓派 4 Model B。可以利用树莓派提供的端口连接各种外围设备。树莓派的一项主要特性是配备了通用输入输出（GPIO），这些接口可以与传感器或执行器进行电气连接，并可根据用户程序进行控制。该计算机由 5V直流电源供电，功耗低于现代计算机。

树莓派能够运行各种软件，包括多个操作系统。广泛使用的操作系统是树莓派基金会的官方操作系统，即基于 Debian Linux 的树莓派操作系统。

与机器学习和深度学习相关的软件及应用程序接口，例如 Python、TensorFlow、OpenCV 等，都可以安装在树莓派上，并将其用作运行推理的边缘设备。由于资源有限，在树莓派上实现模型构建和训练尤其困难。

应对树莓派进行配置，使其能够运行深度学习模型。尽管模型是在功能强大得多的系统（例如个人计算机）上创建和训练的，但在树莓派上运行这些模型需要进行预处理并加载模型，而这需要原生 TensorFlow。

用于运行模型的树莓派 4 Model B 上运行的操作系统的详细信息如下。

- 型号：树莓派 4 Model B Rev 1.5
- 操作系统名称：Debian GNU/Linux，版本 11
- 版本代号：bullseye
- 架构：64位
- 中央处理器：Broadcom BCM2711，四核 Cortex-A72（ARM v8）64位片上系统 @ 1.5GHz

已安装的应用程序接口和模块取决于树莓派的架构和操作系统。不过，在安装应用程序接口之前，我们会在系统中创建一个虚拟环境，这样外部软件包就不会影响我们要进行的测试。虚拟环境中只包含为满足需求而安装的软件包，并且可以避免预装软件包引发的依赖问题。

B. 数据集

- MNIST 和 SVHN 数据集：两个数据集都被划分为训练集和验证集，比例分别为 70% 和 30%。由于图像并不复杂，因此未使用数据增强。我们使用 TensorFlow 的内置层创建了一个自定义卷积神经网络模型。
- 马或人、cats_vs_dogs 和花卉数据集：与之前的模型不同，此处使用迁移学习来训练模型。具体来说，分别使用 VGG16、MobileNetV2 和 ResNet 50 预训练模型。

C. 优化技术

- 权重剪枝 [10, 11]：我们使用基于幅值的权重剪枝技术，以不同稀疏度对权重进行剪枝，稀疏度范围为 S = [0.25、0.50、0.60、0.70、0.80、0.90、0.95、0.97、0.99]。随后，我们对剪枝后的模型进行少量训练轮数的微调，以恢复准确率。
- 模型转换：转换过程是一组用于将原始模型和剪枝模型转换为 TensorFlow Lite 的工具。TensorFlow Lite 是一组工具，可让机器学习模型在移动设备、嵌入式设备和边缘设备上运行。

- 量化[12]：我们使用训练后量化技术，将权重的精度从 float32 降低至 int8。然后，我们对量化模型进行少量训练轮数的微调，以恢复准确率。
- TensorFlow Lite 委托：硬件加速
  借助委托，可以通过利用设备上的 GPU 和处理器等加速器来加速 TensorFlow 模型。
  TensorFlow Lite 使用针对 ARM Neon 指令集优化的中央处理器内核，但中央处理器是多用途的，并未专门针对机器学习模型进行优化。由于不同加速器架构带来的复杂性，每种架构都无法执行神经网络中的所有操作。TensorFlow Lite 委托 API 在 TFLite 运行时与底层 API 之间搭起桥梁，以解决这一问题。

D. 模型基准测试

基准测试是通过在特定系统上运行多项测试，衡量计算机程序相对性能的方法。供模型进行推理的数据或图像存储在与各模型对应的文件夹中。为进行推理测试，每个数据集选取 100 张图像作为样本，以便计算公平的指标。对于每个模型，图像从磁盘逐张加载，而不是按批次加载；计时范围从图像加载开始，到类别预测结果生成结束。每次推理所用的时间都会记录下来，并用于计算平均推理时间。不过，数据集中第一张图像的推理时间会单独测量，因为在这次运行中，加载 TensorFlow 等软件包所需的时间比其他图像更长，而其余推理过程则快得多。因此，在公式中，推理从第 2nd 张图像开始，而不是从第一张开始。

$$t_{infer_1} = t_{end_infer_1} - t_{start_infer_1} \quad (1)$$

$$t_{infer_i} = t_{end_infer_i} - t_{start_infer_i} \quad \text{对于 i = 2 至 N,} \quad (2)$$

其中，

$t_{end_infer_i}$ 是第 i^{th} 次推理的结束时间
$t_{start_infer_i}$ 是第 i^{th} 次推理的开始时间
t_{infer_i} 是第 i^{th} 次推理所用的时间
N 是测试图像的总数

平均推理时间 $\bar{t}_{infer}$ 可计算如下：

$$\bar{t}_{infer} = \frac{t_{infer}}{N-1} \quad (3)$$

标准差 $std\left(t_{infer}\right)$ 可计算为

$$std\left(t_{infer}\right) = \frac{\Sigma_{i=2}^{N}\left(t_{infer_i} - \bar{t}_{infer}\right)^2}{N-1} \quad (4)$$

标准差 standard error, $ste\left(t_{infer}\right)$ 为 calculate

$$ste\left(t_{infer}\right) = \frac{std\left(t_{infer}\right)}{\sqrt{N-1}} \quad (5)$$

i.e. standard deviation given.

这有助于了解推理时间的分布情况。我们可以了解各次推理与平均推理时间之间的偏离程度。数值较低表示各图像的推理时间非常接近。

另一方面，标准误差是统计样本总体中样本的近似标准差。当总体规模较大时，标准误差往往较小，因为标准误差与总体规模成反比。

每个模型的数据集都存放在单独的文件夹中，属于不同类别的数据则被划分到子文件夹中。该脚本使用树莓派中的 Python 解释器运行。

IV. 结果与分析

本节将介绍以下内容：

- 实验所得结果的展示
- 比较剪枝模型与未剪枝模型在准确率、速度和内存使用量方面的性能
- 讨论通过剪枝在准确率与效率之间实现的权衡

A. 剪枝权重 图 1 展示了

在不同数据集上，不同稀疏度 S 下，剪枝后的模型与原始模型在参数数量及准确率/损失方面的比较。

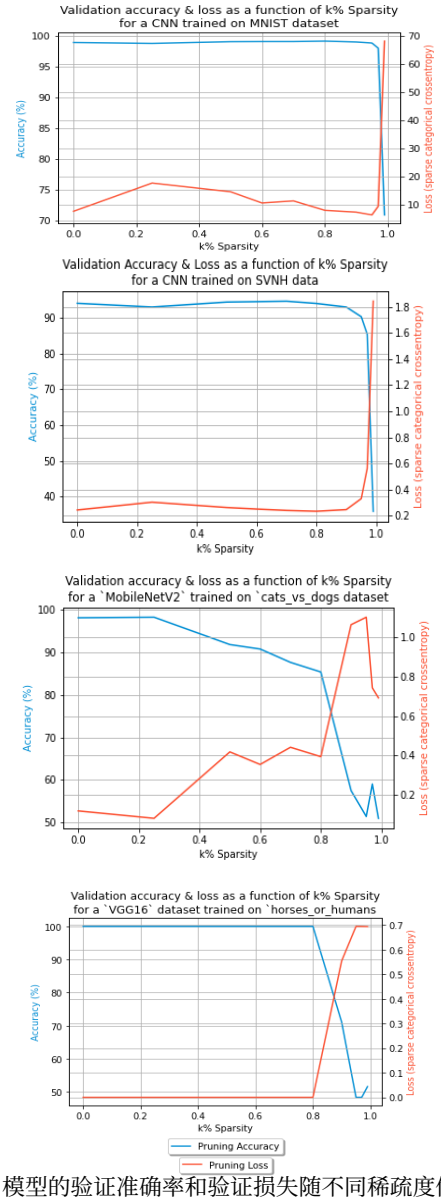

图 1：模型的验证准确率和验证损失随不同稀疏度值的变化

B. 为获得最佳准确率而剪枝模型

按照上一节所述对模型进行剪枝，以观察准确率和损失随不同稀疏度变化的情况后，我们必须为每个模型选择一个稀疏度，以便进一步优化。一个重要的观察结果是，这五个模型对剪枝的反应各不相同，准确率变化因模型而异，甚至可能因未经过测试的数据集而异。我们再次对这五个模型进行剪枝，将其稀疏度调整到某一水平，同时不会显著影响准确率，如表5所示。

表5 用于进一步优化模型的选定稀疏度

模型			验证准确率	
			之前 剪枝（%）	之后 剪枝(%)
MNIST	0.7	5	98.97	100
SVHN	0.8	10	94.32	96.87
马/人	0.8	20	99.67	100
猫与狗	0.5	20	97.91	93.75
花卉	0.7	20	92.55	90.62

在剪枝和重新训练过程中，使用了相同的编译和拟合超参数，仅改变了模型训练的轮数。剪枝后，MNIST、‘SVHN’和‘马或人’数据集的模型准确率有所提高。由此可以得出结论，模型中的大多数参数对模型准确率没有影响，因此可以将其移除。然而，‘猫与狗’和‘花卉’数据的模型在剪枝后的准确率低于原始模型。接下来，应通过将模型转换为适合的格式，准备在树莓派上使用，相关内容将在后文讨论。

C. Keras 模型[1] 和 TensorFlow Lite 的基准测试结果

使用 Keras 模型对树莓派的性能进行了基准测试，结果如表6所示，Lite 模型的结果见表7。

表6 使用 Keras 模型对测试图像进行推理的结果

模型 / 时间（毫秒）	MNIST	**svhn**	马或 人类	猫与 **dogs**	**TF** 花卉
t_{infer_1}	539.58	527.06	1861.52	2520.03	3886.11
$\bar{t}_{infer}$	114.26	120.55	1386.21	241.58	1079.45
$std\ (t_{infer})$ 6		6.90	36.71	8.57	26.19
$ste\ (t_{infer})$ 0.	66	0.70	2.61	0.87	2.65

表 7 剪枝/未剪枝和（或）量化的 TensorFlow Lite 模型的推理结果

剪枝的 TensorFlow Lite 模型					
模型 / 时间（毫秒）	MNIST	svhn	马 or 人类	猫对比 dogs	TF 花卉
t_{infer_1}	21.32 2	01.86 364	8.91 123.	36 1699.8	7
$\bar{t}_{infer}$	3.73 9.				
$std\ (t_{infer})$	0.26 0.	05 4 1.91	3.24		
$ste\ (t_{infer})$	0.03 0.				
未剪枝的 TensorFlow Lite 模型					
t_{infer_1}	38.62	124.79	3559.50	38.30	1090.45
$\bar{t}_{infer}$	3.74	84.57	3475.11	9.54	1009.91
$std\ (t_{infer})$	0.24	2.07	8.86	0.75	8.51
$ste\ (t_{infer})$	0.02	0.21	0.63	0.08	0.82
剪枝和量化的 Lite 模型					
t_{infer_1}	28.90	155.24	1976.78	36.08	2006.02
$\bar{t}_{infer}$	3.06	103.59	1765.95	7.21	1580.21
$std\ (t_{infer})$	0.21	1.98	1.65	0.21	42.71
$ste\ (t_{infer})$	0.02	0.20	0.12	0.02	4.31
未剪枝和量化的 Lite 模型					
t_{infer_1}	29.87	156.41	1951.04	32.78	1832.24
$\bar{t}_{infer}$	3.09	103.49	1768.14	7.21	1595.60
$std\ (t_{infer})$	0.29	1.97	1.59	0.34	61.37
$ste\ (t_{infer})$	0.03	0.20	0.11	0.03	6.20

我们注意到，MNIST 和 SVHN 模型的初始推理时间存在差异，其中 TF Lite 模型以及量化模型的初始推理时间要低得多。在这两种情况下，初始推理时间的缩短幅度几乎是 Keras 模型的 10 倍。

另一方面，与 Keras 模型相比，TF Lite 和量化的“猫与狗”模型的初始推理时间减少了近 25 倍。对于 TF flowers 模型，使用 TF Lite 和量化模型时，初始推理时间减少了一半。马人与人分类表现出不同于其他模型的特征：TF Lite 的初始图像推理时间长于 Keras 模型，但量化模型的延迟与 Keras 模型几乎相同。

考虑模型的平均推理时间时，每次推理所需的时间少于初始推理时间。MNIST 和 SVHN 模型的平均推理时间表现出相似的特征，其中量化模型每次图像推理所需的时间更短。MNIST TF Lite 模型的推理时间比 Keras 模型减少了 30 倍，而对于 SVHN 模型，减少幅度接近 10 倍。与 Keras 模型相比，TF flowers 量化模型的推理时间有所增加。猫与狗模型的平均推理时间低于 Keras 模型，减少了超过一半。马或人模型的平均推理时间在使用 TF Lite 模型时增加，但在模型量化后减少。

D. 模型的准确率变化

现在我们将查看模型在剪枝和转换后的验证准确率，如表 8 所示。

[1] 我们使用 Keras 模型这一术语，以区别于前文讨论的轻量模型（TensorFlow Lite）

表 8 模型准确率 Keras、TF Lite 及量化剪枝和未剪枝模型模型

准确率（%）	MNIST	SVHN	马或人类	猫对比 dogs	TF 花卉
Keras 模型	98.97	97.91	99.67	94.32	92.55
未剪枝 TFLite	100.00	87.50	100.00	96.87	87.5
剪枝 TFLite	100.00	87.50	100.00	96.87	87.5
未剪枝量化	100.00	93.75	96.87	18.75	87.5
已剪枝量化	100.00	93.75	96.87	18.75	87.5

MNIST 模型在转换为 TF Lite 格式并进行量化后，模型准确率有所提高。‘猫与狗’模型转换为 TF Lite 格式后，模型准确率下降，但量化后有所提高，不过准确率略低于 Keras 模型。‘马或人’模型转换为 TF Lite 模型后，模型准确率有所提高，但量化后降低。对于‘SVHN’模型，将模型转换为 TF Lite 模型后，准确率有所提高，但模型量化后准确率大幅下降。TF flowers 模型在量化或转换为 TF Lite 格式后，模型准确率保持不变，并且略低于 Keras 模型。

E. 使用 TensorFlow Lite 委托时的性能：

Arm NN [2] 提供 Tflite 委托库，可加速 Arm 硬件上的某些 TensorFlow Lite 运算符。与 Arm NN 委托兼容的操作将由其执行，不兼容的操作则会卸载到默认操作或 TensorFlow Lite 操作，并在中央处理器上执行。

应在树莓派上单独安装 Arm NN 库并将其启用，使其在执行推理时加载。使用 Arm NN 委托对测试图像进行推理的结果如表 9 所示。由于 MNIST 模型较小且本身推理速度很快，因此使用委托不会对其产生影响。使用 Arm NN 委托时，SVHN 模型的初始图像延迟略有降低。使用委托后，“猫与狗”模型的推理时间大幅缩短；而“马或人”和“TF 鲜花”模型的推理时间则显著减少，几乎缩短至 Keras 模型的一半。

使用委托时，也可以看到类似的平均推理时间效果。使用委托后，马或人和 TF flowers 模型的平均推理时间显著下降。将剪枝模型和未剪枝模型转换为 TF Lite 格式后，这两个模型的推理时间有所增加，但得益于 Arm NN 委托，推理时间缩短了，性能优于 Keras 模型。

表 9 使用未剪枝和未剪枝 TF Lite 模型及 Arm NN 委托对测试图像进行推理的结果

未剪枝 TF Lite 模型					
模型 / 时间（毫秒）	MNIST	svhn	马或人类	猫与 dogs	花卉
t_{infer_1}	31.99	95.40	596.41	39.06	561.15
$\bar{t}_{infer}$	3.75	54.23	458.53	6.45	487.95
$std\ (t_{infer})$	0.56	1.90	4.24	0.12	3.46
$ste\ (t_{infer})$	0.06	0.19	0.30	0.01	0.35
剪枝版 TF Lite					
t_{infer_1}	31.05	92.60	594.59	38.62	554.80
$\bar{t}_{infer}$	3.62	54.80	456.16	6.47	485.54
$std\ (t_{infer})$	0.83	2.45	4.30	0.16	4.33
$ste\ (t_{infer})$	0.08	0.25	0.31	0.02	0.44

V. 结论

根据获得的结果，我们可以得出以下结论：

- 剪枝不会显著改变深度图像分类模型的模型准确率。在某些情况下，剪枝会提高模型准确率，而在另一些情况下，准确率会略有下降。自定义卷积神经网络的验证准确率在这两种情况下都会提高，但使用预训练层时，准确率会降低。
- 当模型转换为 TF Lite 格式时某些模型的表现优于 Keras 模型。自定义卷积神经网络模型的推理时间短于其 Keras 模型。然而，使用 ResNet50 和 VGG16 预训练层的模型表现甚至不如其 Keras 模型。使用这些预训练层的模型，其推理时间有所增加。由于 MobileNet 是为边缘设备设计的，使用该预训练层的模型表现优于其 Keras 模型。
- 当模型被量化（剪枝并未剪枝）自定义卷积神经网络的推理时间有所缩短，甚至低于 TF Lite 版本。然而，与 TF Lite 版本类似，基于 ResNet50 和 VGG16 的模型推理时间有所增加。基于 MobileNet 的模型性能甚至优于其 Keras 模型，但不如 TF Lite 版本。
- 最佳推理结果是在使用 Arm NN 委托在树莓派上运行时获得的。与之前的测试相比，所有模型的推理时间均有所改善，并且显著缩短。Arm NN 委托只能用于模型的 TF Lite 版本，无法用于量化模型。
- 模型大小不会发生显著变化，当与 Keras 格式相比，模型转换为 TF Lite 格式后，模型大小并未发生显著变化。
- 剪枝不会影响模型大小。剪枝后的未剪枝模型转换为 TF Lite 版本或量化版本后，与剪枝模型的模型大小相同。

[2] GitHub - ARM-software/armnn：Arm NN 机器学习软件。此处的代码是 https://review.mlplatform.org/admin/repos/ml/armnn 的只读镜像

- 模型准确率不受剪枝影响，而取决于格式。剪枝版和未剪枝版 TF Lite 的准确率相同。量化模型的准确率相同，且不受剪枝影响。

- 就准确率和速度而言，使用 Arm NN 委托的 TF Lite 模型在树莓派上取得了最佳效果。

参考文献

[1] S. Ameen 和 S. Vadera，《一种从深度图像和彩色图像中对美国手语指拼进行分类的卷积神经网络》，Expert Systems，第 34 卷第 3 期，文章编号 e12197，2017 年。[2] B. H. Curtin 和 S. J. Matthews，《在树莓派上进行野生动物低成本图像分类的深度学习》，载于 2019 年 IEEE 第十届普适计算、电子与移动通信会议（UEMCON），2019 年：IEEE，第 0082-0087 页。[3] J. Deng、W. Dong、R. Socher、L.-J. Li、K. Li 和 L. Fei-Fei，《ImageNet：大规模分层图像数据库》，载于 2009 年 IEEE 计算机视觉与模式识别会议，2009 年：IEEE，第 248-255 页。[4] D. Gray 和 H. Tao，《具有视点不变性的行人识别：局部化特征集成》，欧洲计算机视觉会议（1），第 2008 卷，第 262-275 页，2008 年。[5] S. P. Baller、A. Jindal、M. Chadha 和 M. Gerndt，《DeepEdgeBench：对边缘设备上的深度神经网络进行基准测试》，载于 2021 年 IEEE 云工程国际会议（IC2E），2021 年：IEEE，第 20-30 页。[6] A. Goel、S. Aghajanzadeh、C. Tung、S.-H. Chen、G. K. Thiruvathukal 和 Y.-H. Lu，《用于嵌入式设备低功耗图像分类的模块化神经网络》，ACM电子系统设计自动化汇刊（TODAES），第 26 卷第 1 期，第 1-35 页，2020 年。[7] H. A. Shiddieqy、F. I. Hariadi 和 T. Adiono，《在单板计算机上实现基于深度学习的图像分类》，载于 2017 年电子与智能设备国际研讨会（ISESD），2017 年：IEEE，第 133-137 页。[8] C. F. Silva 和 C. A. Siebra，《卷积神经网络在嵌入式系统图像分类中的应用研究》，载于 2017 年 IEEE 拉丁美洲计算智能会议（LA-CCI），2017 年：IEEE，第 1-6 页。[9] E. Kristiani、C.-T. Yang 和 C.-Y. Huang，《iSEC：一种面向边缘计算图像分类的优化深度学习模型》，IEEE Access，第 8 卷，第 27267-27276 页，2020 年。[10] S. Han、J. Pool、J. Tran 和 W. Dally，《学习权重和连接以实现高效神经网络》，神经信息处理系统进展，第 28 卷，2015 年。[11] S. Vadera 和 S. Ameen，《深度神经网络的剪枝方法》，IEEE Access，第 10 卷，第 63280-63300 页，2022 年。[12] S. Han、H. Mao 和 W. J. Dally，《深度压缩：通过剪枝、训练量化和霍夫曼编码压缩深度神经网络》，arXiv 预印本 arXiv:1510.00149，2015 年。