Abstract-摘要，

在GPU上加速卷积神经网络（CNN）通常包括两个阶段：训练和推理。传统上，这个两阶段过程部署在高端 GPU配备的服务器。在终端计算能力的提高和移动GPU的驱使下，人们对在各种各样的平台上performing inference的兴趣越来越大。在训练阶段要满足高吞吐量和精度的要求，与此相反的是，终端用户会面临与inference tasks相关的不同的要求。

为了解决这个兴起的趋势和新的需求，我们提出Pervasive CNN，一种用户满意的CNN inference 框架。P-CNN是由两个阶段组成的：跨平台离线编译和运行时管理**。**基于用户的需求，离线编译使用架构生成最佳内核独立技术，如自适应批量大小选择和协调精调。 运行时管理阶段 包括精度调节，执行和校准。 第一， 精度调节动态识别最快的内核同时保证可接受的精度。 接下来，运行时内核调度器给每个层划分最优计算资源，并且负责调度GPU线程块。 如果它的精度不是最终用户可以接受的，校准阶段选择较慢的但更精确的内核以提高精度。 最后，我们为CNN设计了一个用户满意度度量来评估我们的普适设计。我们的评估结果表明P-CNN可以为不同的推理任务提供最佳的用户满意度**。**

在GPU 上加速卷积神经网络通常包含两个步骤：训练和推理（Inference）。传统上，这两步处理都在含高端GPU配置的服务器中进行。台式机和手机GPU计算力的不断增长，使得在各种各样的平台上进行推理的兴趣不断地增长。与训练阶段所需要的高吞吐量和高精度不同，在与推理相关的任务中，终端用户面临着更多样化的要求。

为了跟进这一新兴趋势，满足新的要求，我们提出了 Pervasive CNN （P-CNN），这是一个注重用户满意度的CNN 推理框架。P-CNN 有两个组成部分：跨平台离线编译（compilation）和运行时间管理。根据用户的需求，离线编译使用架构独立技术，比如自适应批处理量大小选择（adaptive batch size selection ）和协同微调（coordinated fine-tuning）来生成最佳内核（optimal kernel）。运行时间管理阶段由准确率修正、执行和校正组成。首先，准确率修正会在可接受的准确率下动态地确定最快的内核。然后，运行时间内核调度器会为每一层神经网络分配出最优的计算资源，调度GPU 线程。如果获得的准确率对于终端用户来说是不能接受的，校正阶段会选择一个更慢但是更加精确的内核来提升准确率。最后，我们为 CNN 设计了一个用户满意度衡量指标，用于评估我们的 Pervasive 设计。我们的测试结果显示，P-CNN 在不同的推理任务中提供最佳的用户满意度。

# I.引子

近年来，深度学习based方法，特别是深度卷积神经网络，已经崛起，在很多领域都成为不可或缺的工具，从图像识别到自然语言处理。这主要是由于他们的增长的能力达到高准确率在许多挑战性的机器学习问题。例如，微软最近宣布，它的最新的深度学习网络（PReLU-nets）已经超过了人类水平的准确率在一个分类任务中，使用1000中ImageNet数据集。提高的准确率伴随着显著的计算量的增长。例如，VGGNet（在深度学习社区最受欢迎的CNN之一）要求1.5x（10\*\*10）浮点数乘法每个图片去执行物体识别。为了解决这个挑战，研究人员已经开始寻求CNN加速器的帮助。The GPGPU，拥有大规模并行处理能力，已经成为最受欢迎的深度学习加速器。例如，自从2013年，超过80%参加ImageNet图像识别竞赛的团队使用了GPU。

在GPU上加速CNN通常包括两个阶段。首先，一个CNN模型要在一个大规模的已经打好标签的训练数据集上进行训练。训练阶段，是最耗时的，（例如，在four high-end NVIDIA Titan Black GPUs上训练VGGNet花了3个星期），会交给高端的GPU设备去执行。一旦完成，网络就会被部署去运行inference，用它的先前已经被训练好的参数去分类、识别和处理未知输入。CNN的高准确率催生了一大批人工智能应用，比如DeepFace和Prisma。DeepFace，一款Facebook开发的深度学习面部识别系统，在识别人脸方面接近人工的准确率。Prisma能够把普通的照片转换成著名的艺术作品的风格。由Prisma生成的照片正在占领Instagram。通常来说，这些应用把从移动平台上获得的数据发送到数据中心，在那儿用高端GPU去perform the inference。最近，随着移动GPU的计算能力不断增长，人们对在移动平台上perform inference的兴趣越来越大（比如，loT sensors，移动智能手机）。这使得一些应用，像Facebook Moments（识别一张照片里有哪些朋友）和Google翻译app（通过手机摄像头的实时取景翻译菜单和标识）。结果是，以CNN为基础的应用正在所有的GPU平台上普及。但是，由于GPU硬件资源的差异，在一个GPU上的CNN的最佳结构用在另一个GPU上可能并不适用。这激发了我们研究CNN在GPU上的具体实现，以及制作一个分析的模型去预测在不同GPU架构上的CNN的性能。通过使用一个平台无关的分析模型，在高端GPU上训练得到的CNN模型可以被有效地部署在不同的平台上，不需要经过耗时的再训练。

需要指出很重要的一点，这两个阶段（训练和inference）的最优目标是不同的。训练阶段的目标是尽快获得高准确率。然而在The inference阶段，因为更接近终端用户，它的目标多种多样，要基于具体的任务。实时任务执行inference时延时要小，比如实时监控和自动驾驶。后台和批处理任务，比如部署在数据中心的DeepFace，对实时性不敏感，但是更关心能耗。一些互动任务，比如Google翻译app和Prisma，对实时性敏感，但是可以忍受一些延时。除了运行时间和能量，基于CNN的应用都有一个鲜明特征：准确率。有趣的是，在inference阶段并不总是追求高准确率，特别是当一个CNN的准确率可以超过人类水平的准确率的时候。举个例子，一些娱乐应用不需要高精度，比如Moments。降低准确度甚至可以提高用户体验，因为不显著的精度降低可以使得响应更快。对于一些在安全和科学研究方面的应用，高精度是必要的。因此，准确度对于用户体验的影响同样是由具体任务决定的。为了评估inference阶段的用户体验，我们首先研究了，，，然后提供了

为了保证在各种各样的平台上的理想的用户满意度，我们提出Pervasive CNN，a user satisfaction-aware CNN inference框架。P-CNN有两个阶段：跨平台线下编译和运行管理。基于使用了CNN的应用的规格，P-CNN可以推断出终端用户的需求。然后，根据这些需求，线下编译会针对要部署的平台生成最佳内核，还会为运行管理阶段提供调度信息。运行管理阶段包括精调，执行和和校准。首先，精调动态的标识出在可接受的精度范围的最快的内核。在此阶段，会为接下来的校准产生一系列调谐表。在运行态，当输入数据改变时，P-CNN监控的输出精度。如果终端用户不能接受这个精度，那么校准会选择一个l不太激进的调谐表（对应一个较慢但更精确的内核）来提高输出精度。

为了保证基于CNN的应用程序的实际响应时间在终端用户的忍受范围之内，我们首先开发了一个平台独立的时间模型来估计CNN的执行时间。然后用这个时间模型来指导跨平台的离线编译。对于特定的GPU架构生成最优内核，我们首先介绍深度学习库两个主要调优参数：子矩阵的大小和每个线程的寄存器消耗，这也决定了卷积核的性能。然后我们设计了一个分析指标并通过协调地微调这两个参数来选择最优内核。

通过与多个运行时调度器方案比较， P-CNN在各种GPU平台上执行不同的推理任务时获得了最佳用户满意度。 评价结果还表明我们的基于熵的近似方法达到相同的效果（即在10％精度损失内可以获得1.8倍加速），作为基于精度的方法。

总之，这项工作的主要贡献是：

我们提供普及CNN，一种提高用户满意度的 CNN推理框架，能够有效地在各种平台上部署经过CNN训练的模型而不需要耗时的再训练。

我们提出了平台独立的分析模型（即时间和资源模型），以指导跨平台的离线编译。

我们的第一步是帮助计算机架构团队去理解现有的GPU架构在支持基于CNN的应用程序的不足。

我们开发了一种运行时精度调节技术，可以动态的根据部署环境改变CNN的精度。

本文的其余部分安排如下。 第二节介绍背景。 第三部分深刻 基于学习的应用程序。 第四节描述了我们的普遍CNN，包括跨平台平台离线编译和runYtime管理。 第五节评估我们的普适设计。 相关工作和结论分别在第六节和第七节中讨论。

# II。背景

A．CNN

我们以AlexNet为例介绍CNN。如图1所示。它由五个卷积层，三个max池化层，和两个分类器层组成。 卷积层执行卷积滤波器和输入特征图的局部区域的点积。 这些操作决定了CNN计算的执行时间。 卷积层中的卷积操作受益于优化的矩阵乘法库。 cuBLAS [9]是一个这样的库，实现基本线性代数子程序（BLAS），在Nvidia CUDA运行时库的顶部。

我们在图2中展示了一种典型的基于矩阵乘法的卷积运算。

在步骤1中，一个被称为 im2col的操作伸出本地

区域在输入图像中成columnYmajor矩阵（Dm），类似地，在步骤2中，CONV层的权重为 伸出到过滤器矩阵 （Fm ）。 然后原来的卷积运算可以转换成步骤3中的矩阵乘法（Fm\*Dm）。在图2中，过滤器矩阵Fm的维度是——，而数据矩阵Dm的维度是。输出矩阵Om的维度是——。所以我们可以通过Fm\*Dm的乘累加操作计算出一个卷积层的浮点运算量（即FLOPs）

Conv = 2Nf，，，，

一个乘累加操作计为2个浮点运算量。该转化率通常用来衡量卷积层的计算强度。

B.用户对CNN的满意度

虽然我们知道用户对CNN的满意度是和运行时、能耗和精度相关的，但是这三者对用户满意度的影响在不同的任务中是不同的。为了准确的反映出这三个因素如何决定用户满意度，我们将CNN应用程序分为三类——交互式任务，实时任务，和后台任务。首先，我们分别用这三类任务来分析运行时和能耗对用户满意度的影响。然后我们将深入讨论CNN应用的独特因素（即精度）。

1,。交互式任务

由于大多是交互式任务是面向用户的，，比如人工智能的手机应用（例如，Prisma和Google翻译），他们对运行时敏感。如果应用程序的响应时间超过人类可接受的限度，用户会放弃使用该应用程序。然而，用户也可以容忍一些延迟，因为他们

理解人工智能应用程序涉及复杂计算。 我们在图3中阐释了用户 满意度，运行时间和能耗之间的关系。运行时间在x坐标轴上从左到右增加。之前的工作[10]表明，交互式的应用程序的用户满意度可以分为三个不同的状态：随着应用程序运行时增加：感觉不到的、可容忍的和不可用的。

在感觉不到的区域，运行时增加不会危害用户满意度，但可以节约更多的能量。因此，在这一阶段没有必要去追求最快的响应速度。相反，我们应该努力

通过降低性能来最小化能耗，使运行时接近Ti。超过Ti之后，用户满意度进入可容忍区域，用户满意度会不断降低因为用户开始注意到缓慢的响应时间。然而，运行时在此区域仍然可以接受。当运行时进一步增加，用户满意度最终在Tt时被摧毁。在Tt和之后，用户会放弃这个任务。结果是，用户满意度在不可用区域降到0。我们应该努力避免在这个区域运行任务。

2.实时任务

实时任务有严格的运行时要求。例如，自动驾驶车辆的行人检测有严格的执行时间限制，如果不满足这一限制，可能会导致严重的交通事故。因此，实时任务的曲线没有图3中那样的可容忍区域。它的感觉不到的和不可用的区域几乎和交互式任务相同。但是，在感觉不到区域的结束（即Ti）应小于最后期限。

3.后台任务

后台任务通常对运行时不敏感，因为用户不期望结果具有更长的持续时间。相反，这些任务更关心能耗。例如，在获取许多照片之后，用户会切换到其他应用程序，如网页浏览等，而Moments正在识别在这些照片里有哪些朋友。虽然用户不需要立刻发送照片给他们的朋友，他们更加在意Moments的耗电量因为他们想要更长的电池寿命。因此，后台任务的运行时间不会影响用户满意度。在图3中，后台任务的整个区域都是感觉不到的区域。注意，随着运行时间的增加，能耗首先降低，然后平稳。这是因为能耗是product of power and runtime。在Tc和之后的时间里，power的decrease被runtime的increase抵消了。

4.基于CNN的应用的精度

除了运行时和能耗，CNN应用程序有一个独特的标识：精度。有趣的是，高精度在推理中不总是受青睐的，特别是当最先进的CNN的精度可以超越人类水平的精度。 例如，一些娱乐应用不需要高精度，如Moments。 降低准确度甚至可以提高用户体验。不明显的精度下降可以带来更快的反应时间。在这种情况下，找到最佳精度很重要。低精度会影响用户满意度。高精度会导致更重的计算负担。

传统上，研究人员使用标记的测试数据集来评价CNN应用的准确性。 然而，不可能创建一个标记数据集包含所有的应用程序。一个可行的选择是在CNN应用中实现自我诊断。在运行时，他们可以利用其输出来诊断自己。由于缺少标记的测试数据集，我们使用网络不确定性[12]而不是精度作为度量。我们以一个分类任务为例。每个输入数据应该属于一个由Y {1，...，N}表示的特定类别。但是，在运行时我们不知道正确的类标签。CNN分类器层（Y）的输出是概率分布P。例如，P（0.6,0.3,0.1）表示输出属于class1的可能性是60％，属于class2的可能性是30%。我们用输出的熵去衡量输出的不准确性。

熵值表达式是

H（Y）

更高的熵值意味着分布P具有更多的不确定性。例如，P1（0.4,0.4,0.2）的熵高于P2（0.7,0.2,0.1），因为P1在class1和class2两者中高度混淆，而P2更加确定输出属于class1。高不确定性（即高熵值）意味着低精度。如表1所示，网络的精度随着熵的减小而提高。因此，我们利用网路熵作为精度的度量来判断一个CNN的输出是否可靠。

# III。.characterization

A．实验设置和方法

为了表征CNN的实际部署和覆盖所有可能你的应用场景，我们部署了三个ImageNet获胜者模型（AlexNet [1]，GoogLeNet [13]和VGGNet [4]）到四个不同的平台（服务器，台式机 ，笔记本，和移动端）。详细参数列于表二。

我们使用Nvidia Visual Profiler收集GPU运行时信息。CNN模型由Caffe [15]训练，一个学术界和工业界都广泛使用的开源深度学习框架。我们characterize三个最新的深度学习库：cuBLAS（Caffe采用），cuDNN [16]（由NVIDIA开发的主要工业库）和Nervana [17]（最快的实现），在CNN应用的加速上。测试集是从ImageNet [3]上选取的。

B．inference prefers a small batch size

对于训练阶段，通常是批量输入几百个数据并同时处理。这种批处理方法不仅防止过拟合，而且通过摊销从GPU内存加载权重的开销，能显著提高GPU效率。例如，Karen et al. [4]使用随机梯度下降法（SGD）和一批256的样本量来训练VGGNet。批处理方法也有利于推理时的处理吞吐量，它显著增加网络延迟（即网络端到端响应时间），特别是在移动设备平台上。 表III比较了不同的网络延迟.

考虑到移动GPU的计算能力有限，我们在训练时选择比256小的样本量。AlexNet用128，GoogLeNet用64，VGGNet用32。如表三所示，尽管Nervana在这三种最先进的深度学习库中获得了最快的性能，它也需要大约400ms在TX1上使用小批量训练幼稚的CNN（即AlexNet）。

这远远无法满足最终用户的实际需求。更糟的是，cuDNN和Nervana不能够在移动GPU上使用批处理方法运行GoogLeNet和VGGNet，因为out-of-memory问题（表三中列为x）。

更重要的是，没有足够的数据。。。例如，大多数互动式任务一次只提交一个请求，然后等待响应。因此，推理倾向于较小的批处理量，因为要求响应时间快而且只有有限可用的输入数据，特别是对实时任务和交互式任务。

C．当前GPU实现的低效率

尽管推理优选小的批处理量来获得较好的用户满意度，高度优化的深度学习库，比如Nervana和cuDNN，对较小的批量表现不佳。例如，Nervana的批处理大小必须是32的倍数。因此，批处理（即，表III中的fontYbold）对应到支持的最小批量大小：32。支持最小批量大小，GoogLeNet的最小延迟TX1仍然大于500ms。 虽然nonY的延迟在笔记本电脑和desNtop平台上的批处理方法可以满足最终用户的要求（例如，<30ms），它们的吞吐量（即每秒处理的图像的数量）最佳。图4显示了没有配料的生产率比到使用分批方法的。

D．现有的提升Util的方法

1）小的子矩阵大小

虽然cuDNN采用小的subYmatrix大小（即32x32）在TX1中增加资源利用率性能仍然劣于cuBLAS，如图所示。 5。 这是因为cuDNN中的小子矩阵导致

减少寄存器消耗，如表四所示将进一步导致更多的全局内存事务

降低计算密度。 图。 图6示出 计算密度的比较：浮点的比率

指令对不同subY中的总指令矩阵大小。 因此，在更高的价格之间存在交易

资源利用（即，更小的子矩阵）和更高计算密度（即，更大的子矩阵）。 实际上层的性能由子Y矩阵的大小确定和寄存器使用。 我们可以通过获得最佳吞吐量使用crossYplatform离线编译fineYtune subY矩阵大小和每个线程的寄存器消耗跨不同平台的卷积层。

*2）多进程机制*

虽然多进程机制，如MPS [21]可以提高GPU资源利用率，它没有控制权 如何调度TB（即，Thread BlocNs）。 有可能的更多的TB被分配给其他并发tasN使用MPS。 因此，在运行时没有时间Y的保证敏感的CNNY应用。 注意虽然

技术如SpatialYMultitasNing GPU [22]，可以通过专门分配满足最终用户的要求足够短信CNNYbased应用， *的Util* 多样性挑战这种方法。 如果我们分配最大值SM的数量（即，在层中使用的SM资源最高 *的Util）*的所有层，仍有资源利用不足与低 *的Util* 层 。 相反，我们应根据 *的Util*各层分配SM的 。具体来说，我们首先尝试找到SM的最佳数量满足runYtime要求，然后利用空间分配技术来向每个层分配最优SM电源门以其它方式降低能量

消费。

*3）大批量*

对于bacNground tasNs，我们可以通过提高GPU利用率增加批量大小。 然而，配料方法与a 大批量大小容易用完GPU内存，因为基于CNNY的应用程序是记忆性的[23]。 我们应注意批量大小选择。 什么时候 GridSize等于maxBlocNs，GPU资源完全并且此时的批量大小是最佳的。 然而，由于硬件资源的变化（例如 *NSMS*和*R），*不同的平台有不同的maxBlocNs，这导致了

最佳批量大小在不同的GPU平台之间变化（在图8中标记为红色）。 更糟糕的是，由于网络参数对于CNN的，每一层不同，如 *W 0至H 2 O*和*N 楼* 各种层具有不同的最佳批量，即使是同一平台。 因此，我们必须制造一个

分析模型来预测CNN的性能不同的GPU架构。 有了这个模型，我们可以很容易

推断每个基于CNNY的应用程序的最佳批量大小在不同的GPU平台上。

# IV。 普适CNN

我们提出普及性CNN（PYCNN）有效将基于CNNY的应用程序部署到不同平台

保持最高的用户满意度。 如讨论 第二节，响应时间，能量消耗和精度

确定用户的满意度。 在这三个因素中，endy用户对响应时间和精度更敏感。从而，我们的调度策略给出的时间和精度最高优先。 满足响应时间的要求和精度，PYCNN试图通过最佳使用来节省能量GPU资源。 可以导出最佳GPU资源。通过资源模型。 以确保响应时间在下用户容忍度，PYCNN特征的时间模型来预测响应时间。 基于预测的性能，我们可以调整离线编译期间的响应时间。 允许基于CNNY的应用的动态适应精度到realYworld场景，PYCNN执行精度调整 在runYtime阶段。 因此，我们的PYCNN框架包括两个主要步骤：跨平台离线编译和runYtime管理，如图2所示。 10.基于部署GPU架构和用户需求，离线

编译生成每个层的最佳内核提供调度信息（即调度配置），包括最佳占用数SM的（即*，optSM）*及各层的*optTLP，*对于runYtime管理。 runYtime管理阶段包括精度调整，执行和校准。 基于调度配置，动态精确调整

识别速度最快Nernels可接受 *CNN 熵*生成用于以下的一系列调谐表

校准。 在调优和执行阶段，runY

1. 用户输入

用户输入模块是。但是，对于最终用户来说，每提交一次请求都要清晰的描述他们的需求是很烦的。因此，我们设计了一种推理机制去自动推测最终用户的需求。准确地说，我们首先根据应用的特点把应用按照不同的任务类型分类，如实时任务、后台任务和交互式任务。然后我们写了一个look-up表去记录每个任务的时间要求。时间要求可以从调查报告[24][25]中的一些用户体验的统计信息中得到。在未来，我们可以创建一个更细粒度的时间要求表，为每个用户使用机器学习技术来学习用户体验。 最后，我们可以使用应用程序的任务类型来查找此表，并找到每个应用程序的要求。

B.跨平台离线编译

1．批量大小的选择

a．后台任务：因为后台任务不关心响应时间，我们可以使用批处理方法实现最大处理吞吐量和减少能耗。但是，我们应该注意批量尺寸选择，因为基于CNN的应用程序是内存密集型的，大批量很容易耗尽GPU内存。因此，最佳批量大小应该是最小批量，允许所有层充分利用GPU资源。如表V所示，最后一层显示最小的Util。因此，最佳批量大小应确保最后一层的Util等于1。当批量大小大于最佳批量大小，处理吞吐量将不能进一步提高，因为所有的计算资源都已经被利用。  
b）其他任务：由于这些任务（例如，实时和交互任务）对响应时间敏感，在时间要求（T）内又没有生成足够多的可用数据，我们将初始批量大小设置为在T期间产生的可用数据的数量（即，T / data生成速率）。  
2）内核优化  
在确定初始批量大小后，我们开始优化每个层的内核。如第三章节的D部分所述，内核的性能主要取决于子矩阵大小和每个线程的寄存器消耗。对于CNN来说，常见的子矩阵大小是128x128,128x64和128x32 [17]。子矩阵的选择不仅决定着块的数量（即网格尺寸），也影响每个线程的寄存器使用（即curReg）。对于寄存器密集的内核，如sgemm，降低寄存器的使用可以增加线程级并行（即TLP）如图9所示。每个线程的最小寄存器使用（即minReg）是由SM的寄存器资源和SM支持的的最大线程数的比值确定的。因此，寄存器数量的调整范围是从minReg到curReg。

为了实现最佳性能，我们应该协调微调子矩阵的大小和每个线程的寄存器。图9显示了不同的寄存器数在TLP的子矩阵大小的影响（即128î128）对K20。这里curReg是127个，minReg是32个。在图9中，每个楼梯由几个设计点相同的TLP不同数量的寄存器，每个线程。在相同的线程级并行的情况下，单线程使用最多的寄存器的设计具有最好的性能。因此，我们只需要考虑每个楼梯上的最右点（以红色标记）。 通过修剪的设计空间，我们可以轻松地探索候选点的性能。 这里我们使用寄存器溢出技术[26]来减少寄存器到这些候选点的数量。

1. 运行时管理
2. 如第二部分的B部分所述，精度是基于CNN的应用程序的一个独特的标识，尽管最先进的CNN可以得到比人类水平还要高的精度，但是没有必要总是让CNN应用运行在如此高精度水平。降低精度甚至可以提高用户体验，因为不明显的精度降低可以缩短响应时间。因此，为每个CNN应用程序找到最优精度（即optAccuracy，不是最高精度）对我们来说是很重要的。然而，不同的应用对精度会有不同的要求。即使对同一个任务，不同的应用环境可能会有不同的optAccuracy。例如，部署在一个热闹的广场上的人脸识别系统比部署在一个安静的房间里的需要更高的精度。由于训练CNN是一个耗费时间的过程，为每个应用去训练一个新的CNN模型以获得不同的最优精度的要求是不现实的。相反的，我们引入一个基于熵的近似方法来解决这个挑战，使得CNN的精度在运行时动态地适应不同的环境。

因为CNN主要专注于图像或视频处理

# V．评估

1. 在本节中，我们使用真正的基于GPU的系统和基于模拟器的框架来全面评估我们的P-CNN设计。我们的模拟器框架是基于GPGPU-sim [28]实现的。GPUWatch [29]用于测量能量消耗。我们首先介绍我们的度量，CNN的满意度（即SoC），来评估我们的P-CNN，然后描述基线方案，在调度基于CNN的应用程序时，只考虑一个或两个因素，如节能调度器，性能优先调度器和每能量的QoS（即QPE）调度器[10]。接下来，我们将它们与我们的面向SoC的P-CNN进行比较，并在不同平台的三个重要场景上对我们的P-CNN进行综合评估。我们还使用标记的测试数据证明基于熵的精度调整的程序。
2. .用于评估用户对CNN的满意度的度量

为了评估P-CNN如何使基于CNN的应用的用户满意度受益，我们提出了一种新的度量，CNN满意度（即SoC）：  
  
SoC时间是响应时间如何影响用户满意度的度量，如第II.B节所述。察觉不到的SoC时间被定义为1。不可用的SoC时间被定义为0。在可容忍区域中，SoC时间随着响应时间 [30]线性下降。我们使用SoC准确度来描述准确度对用户满意度的影响。如果CNN熵小于预先定义的阈值（即CNN阈值），则SoC精度为1。否则，SoC精度定义为CNN阈值/ CNN熵。

1. 基线调度器

我们将P-CNN与以下五种基线方案进行比较：  
1）性能优先调度器：为了获得快速响应时间，该方案通常采用非批处理方法来执行基于CNN的应用。  
2）节能调度器：该方案仅考虑能量消耗。它在训练阶段利用批处理方法。由于大批量处理的高处理吞吐量，它可以消耗更少的能量来执行任务。它与我们的P-CNN相同，在处理后台任务时没有精度调整。

3）每能量QoS（QPE）调度器：当调度基于CNN的应用时，该方案考虑响应时间和能量消耗。它试图在给定的运行时要求下消耗最少的能量。  
4）QPE +调度器：与QPE相比，QPE +可以利用最优计算资源进行推理。当Util低时可以节省能源。 QPE +与我们的P-CNN类似，没有精确度调整。  
5）理想调度器：它是对最终用户的所有需求具有先验知识的oracle调度器。它可以通过分析所有调谐点获得最大的SoC。

1. P-CNN的综合评价  
   我们在三个典型任务下评估各种调度：年龄检测（交互式任务），视频监控（实时任务）和图像标记（后台任务）。在用户提交自拍之后，年龄检测可以使用CNN估计他/她的年龄。由于可用数据有限，此交互式任务通常采用非批处理方法。由于人机交互研究表明，对于某些交互任务，用户可以忍受100毫秒的延迟，并且放弃需要超过3秒的加载的应用[32]，Ti和Tt分别被定义为100毫秒和3秒用于年龄检测任务。对于实时视频监控任务，处理帧的最后期限是帧速率。例如，如果监控视频的帧速率为60FPS，则截止时间为1 / 60s。对于背景图像标记，没有时间限制。  
     
   我们使用最新版本的GPGPU-sim来比较这五个调度器。表VI中的仿真参数来自Nvidia K20c和TX1的规格。如第二节A中所讨论的，卷积运算是通过矩阵乘法来实现的。因此，我们运行矩阵乘法的内核（即sgemm [33]）来执行GPGPU-sim中的卷积运算。  
     
   我们首先比较P-CNN与其他调度器的运行时间和能耗。然后我们使用SoC来总结我们的P-CNN的用户满意度。图13，14和15分别展示了在不同调度方案下的运行时间，能量消耗和SoC分数。为了与在某个度量中表现最佳的基线调度器进行比较，将运行时标准化为性能优选调度器，而将能量消耗归一化为节能调度器。  
     
   我们首先讨论K20c的结果。 由于我们的时间模型可以准确地预测响应时间，所以三个应用的响应时间在配备有时间模型的调度器（即除了节能调度器之外的所有调度器）的不可感知区域内。 因此，在图13（a）中，除了节能调度器，所有调度器都实现了最大的SoC时间。 对于实时任务，节能调度器的运行时间超过截止时间，这导致其SoC时间和SoC（即，图15（a）中的‘x’）为0。 如图14（a）所示，P-CNN在这些调度器中消耗最少的能量（几乎等于理想调度器）。 我们逐步解释这种现象。对于交互式任务，QPE +比QPE消耗更少的能量，因为QPE +可以关闭空闲SM以节省能量。然而，对于实时和后台任务，QPE和QPE +的能量是相同的，因为GPU资源已经在QPE中使用并且没有空闲SM可以被QPE +关闭。对于精确的非敏感任务，如年龄检测和图像标记，P-CNN可以通过选择具有可接受的精度（5％损失）的最快的内核（1.5X加速）来节省比QPE +更多的能量。然而，对于需要高精度（即视频监控）的应用，P-CNN消耗与QPE +类似的能量。  
   具有理想的SoC时间和最小的能量消耗，P-CNN获得了图15（a）中的高SoC。然而，它仍然没有实现与理想调度器一样高的SoC，特别是对于交互式任务（即，年龄检测）。这是因为娱乐应用（例如年龄检测）可以以低精度很好地工作，而P-CNN采取保守的准确度来执行该任务。在未来，我们可以通过学习用户体验来改进对交互式任务的准确度要求的预测。然后我们在Nvidia TX1上比较这五个调度器。评估结果表明，P-CNN在TX1上也实现了理想的SoC时间（图13（b）），最低能量消耗（图14（b））和高SoC（图15（b））。注意，对于实时任务，除了我们的P-CNN和理想调度器，所有调度器都不能满足用户满意度（即，图15（b）中的“x”）。如图13（b）所示，尽管使用了非批量方法，实时任务的运行时间仍然超过TX1的最后期限，这是最终用户不能接受的。我们的P-CNN可以进一步加速在TX1上的推理过程，通过使用近似方法以满足用户满意度。
2. 精调

如第IV.C节所述，我们使用网络不确定性（即CNN熵）来指导运行时精度调整。这里我们使用标记的测试数据集来评估基于熵的精度调整过程中的精度变化。图16显示了在该调谐过程期间的加速（橙色条），熵（蓝线）和精度（绿线）。我们观察到加速单调增加，直到网络不确定性大于预定义的阈值。同时，网络不确定性（即E）的增加对应于网络精度（即，A）的减小，这表明CNN熵是评估CNN的准确性的有效度量。精度调整后，我们实现了1.8倍的加速，精度有一点点下降（10%）。图16中还包括了基于精度的近似，其中使用标记的数据来测量CNN精度，然后近似由精度指导。 图16阐释了我们的基于熵的近似如何获得了与基于精度的方法相同的效果（即，类似的加速和精确度）。与基于精度的方法相比，我们的基于熵的方法是无监督的（没有标记的数据来测量精度），这适用于运行时精度。

# VI．相关工作

深度学习：sirius [34]是一个基于DNN服务的开放端到端智能个人助理。 DjiNN [35]进一步使社区表征运行DNN服务的GPU加速服务器系统，并为DNN服务的未来仓库规模的计算机体系结构的设计提供了见解。 D3NN [23]还为高端服务器开发了高吞吐量深度学习系统。不仅关注数据中心平台，我们的P-CNN可以在保证高用户满意度的情况下高效地将CNN应用部署到不同平台。

GPU：最近，已经提出了用于GPU多任务的架构扩展。 Adriaens et al。表明不是所有的GPU应用程序都充分利用GPU资源[22]。他们提出了一些启发式空间多任务，并演示了相较于使用GPGPU-sim的顺序内核执行的性能提升。 Liang et al进一步提高GPU的多任务处理能力。他们提出了GPU空间-时间多任务[20]，允许不同的内核在空间和时间上共享GPU资源。然而，不同层之间的CNN的不同GPU资源利用率挑战了这些多任务机制，这导致在各个层之间不适当的SM分配。此外，它们的方法在分配SM时不考虑用户满意度。虽然我们的P-CNN也基于空间划分，我们首先使用我们的分析模型来评估每层的Util，然后根据每层中的Util分配最优SMs，这可以使用更少的SM实现高用户满意度。

近似计算：近来，深度学习社区的研究人员已经开发了几种加速CNN的方法，包括低秩近似[36]和网络修剪[37]。其中，网络修剪主要集中在减少CNN的存储，使其易于部署到移动平台。基于修剪的方法主要是压缩全连接层（大多数CNN参数驻留在其中）的参数，而我们的方法主要集中于加速卷积层（其中花费大多数处理时间）。此外，它需要修改CNN架构，这导致耗时的重新训练。因此，网络修剪不能用于运行时精度调整。穿孔CNNs [38]也利用循环穿孔技术做近似。虽然其近似过程被引导为监督方法（即，损失函数），但是其仍然不适用于运行时精度调谐。

# VII．结论

这项工作提出了Pervasive CNN，一个用户满意的CNN推理框架。 P-CNN包括跨平台离线编译和运行时管理。首先，P-CNN基于基于CNN的应用的规范推断最终用户的要求。根据这个要求，离线编译为部署的平台生成最佳内核。然后，精度调整找到此部署场景的最佳精度。在执行期间，运行时内核管理将最佳SM资源分配给每个层，然后调度GPU线程块。如果最终用户不能容忍其精度，则校准选择较慢但更精确的内核以提高精度。特别是，P-CNN具有准确的时间模型以保证响应时间的要求。我们的评估结果表明，与其他运行时调度器方案相比，P-CNN能够以最好的用户满意度有效地将基于CNN的应用部署到各种平台。