arXiv:2303.04554v1 [cs.CV] 8 Mar 2023

RADAM：通过深度激活图的随机聚合编码进行纹理识别

Leonardo Scabini[1,2], Kallil M. Zielinski[1], Lucas C. Ribas[3], Wesley N. Gonçalves[4], Bernard De Baets[2], 和 Odemir M. Bruno[1]

[1]圣卡洛斯物理研究所，圣保罗大学，邮编 13560-970，圣卡洛斯 - SP，巴西 [2]KERMIT，根特大学数据分析与数学建模系，Coupure links 653，邮编 9000，根特，比利时 [3]圣保罗州立大学生物科学、人文与精密科学研究所，邮编 15054-000，圣若泽杜里奥普雷图 - SP，巴西[4]南马托格罗索联邦大学计算学院，邮编 79070-900，大坎普 - MS，巴西

摘要

纹理分析是计算机视觉中一项经典但颇具挑战性的任务，深度神经网络正被积极应用于此。大多数方法都基于在预训练骨干网络周围构建特征聚合模块，再针对特定纹理识别任务对新架构进行微调。本文提出一种名为**随**机**聚**合**深**度**激**活**图**编码（RADAM）的新方法，无需对骨干网络做任何改动即可提取丰富的纹理表征。该技术利用随机自编码器（RAE）对预训练深度卷积网络不同深度的输出进行编码。RAE针对每幅图像单独训练，并采用闭式解；随后利用其解码器权重构成一维纹理表示，并将其输入线性支持向量机。这意味着无需微调或反向传播。我们在多个纹理基准数据集上评估 RADAM，并在不同计算预算下取得最先进的结果。结果表明，如果能更好地编码预训练骨干网络学习到的表征，那么这些骨干网络或许无需额外微调即可用于纹理识别。

1 引言

几十年来，纹理一直被《计算机》视觉领域视为多种应用中图像识别的一种基本视觉线索。尽管尚无广泛认可的理论定义，但通过观察周围的世界——从日常生活中的材料表面，到显微图像，甚至望远镜和遥感获取的宏观图像——我们都对纹理形成了直观认识。在数字图像中，一种抽象定义认为，纹理元素源于像素的局部强度恒定性和/或变化，并在不同尺度上大致独立地产生空间模式 [39]。

传统的纹理识别方法着重于对纹理模式进行数学描述，考察统计特征 [10, 15, 24], 频率 [1, 12], 复杂度/分形性 [2, 38], 等属性 [52]。即使在受控成像场景中，纹理的许多此类特征也难以建模。此外，数字图像本身的复杂多变也带来了额外的变化，使得现实应用中的任务更加复杂。

近年来，人们利用在大型自然图像数据集上预训练的模型，将深度神经网络的强大能力拓展到了纹理分析领域 [4–6，22，46–51]。这些迁移学习方法将预训练模型通用的视觉能力与专门用于捕捉额外纹理信息的技术相结合，在若干纹理识别任务上取得了最先进的性能。因此，近期大多数深度纹理识别研究都提出围绕预训练深度网络（骨干网络）构建新模块，并针对特定纹理分析任务重新训练新架构。然而，尽管新模块的计算复杂度相对较低，推理效率也因此较高，但重新训练骨干网络本身通常代价高昂。与此不同，随机神经网络 [13,25,26,40] 提出了一种用于训练神经网络的闭式解，以取代常见的反向传播方法，并具有多种潜在应用。例如，研究者分析了基于随机化的模型在 [17]MNIST 等数据集上的训练时间，结果显示训练速度最高可提升 150 倍。当使用数十万张图像训练模型时，这些提升尤为显著。

在这项工作中，我们提出了一种新模块，用于从预训练深度卷积神经网络（DCNN）中提取纹理特征。该方法名为**随**机编码**聚**合**深**度**激**活**图**（**RADAM**），**其研究方向有别于近期的深度纹理识别文献。**我们不通过增加骨干网络的复杂度再重新训练整个网络，而是提出一种利用新型随机化模块对骨干网络特征进行简单编码的方法。该方法将预训练卷积网络不同深度的深度激活图聚合起来，然后针对每张图像以像素为单位训练随机自编码器（RAE），并采用闭式解。该模块输出所训练随机自编码器的解码器权重，并将其作为输入图像的一维特征表示。这种方法简单易行，无需进行超参数调优或反向传播训练。我们提出在这些特征之上接入线性支持向量机，使用标准参数即可完成训练。我们的代码已公开，并存放在公共代码仓库中[1]。总之，我们的主要贡献如下：

(i) 我们提出了 RADAM 纹理特征编码技术，将其应用于预训练的深度卷积神经网络骨干，并与简单的线性支持向量机相结合。与以往工作提出的方法不同，该模型无需微调骨干网络即可取得令人瞩目的分类性能。(ii) 更大的骨干网络和更好的预训练能显著提升 RADAM 的性能，这表明我们的方法具有良好的扩展性。

2 背景

我们首先开展 关于深度学习和随机神经网络在纹理分析中的应用的文献综述。
此处介绍的方法 也纳入我们的实验进行比较。

2.1 深度神经网络纹理分析

在这项工作中，我们利用预训练的深度神经网络，重点研究基于迁移学习的纹理分析。若想全面了解不同的纹理分析方法，读者可参阅 [19]。已有大量研究将深度学习用于纹理识别，本文根据两种方法对这些研究进行回顾：特征提取或端到端微调。有些研究仅利用卷积神经网络进行纹理特征提取，并在模型架构之外使用专用分类器。Cimpoi 等人 [6] 是这一领域最早开展研究的团队之一，他们比较了两种不同卷积神经网络架构在特征提取方面的效率：FC-CNN 使用全连接（FC）层，而 FV-CNN 则使用费舍尔向量（FV） [5] 作为池化方法。他们证明，总体而言，全连接特征的效率并不高，因为其输出与像素的空间顺序高度相关。随后，Condori和Bruno [22]开发了一种名为 RankGP-3M-CNN 的模型，该模型采用全局平均池化（GAP），对三个组合卷积神经网络（VGG-19、Inception-V3 和 ResNet50）不同深度的激活图进行多层特征聚合，从而提取特征向量。他们提出了一种排序技术，根据训练数据集选出最佳激活图，在某些情况下取得了令人鼓舞的结果，但计算负载也随之增加，因为该方法需要三个骨干网络。Lyra 等人 [21] 同样提出了对多个卷积层进行特征聚合，但采用基于费舍尔向量的方法进行池化。

大量研究提出了端到端架构，使骨干网络能够针对纹理识别进行微调。Zhang 等人 [51]在深度卷积神经网络（DCNN）之上提出了一个名为深度纹理编码网络（Deep Texture Encoding Network，Deep-TEN）的无序编码层，该层支持任意尺寸的图像。Xue等人 [46] 提出了深度编码池化网络（Deep Encoding Pooling Network，DEPNet），将 Deep-TEN中的纹理编码层特征与全局平均池化（GAP）相结合，以同时探索图像的局部外观和全局上下文。随后，这些特征由双线性池化层进一步处理 [18]。在另一项工作中，Xue 等人 [47] 还将差分图像的特征与DEPNet的特征相结合，构建了一种新架构。Zhai 等人 [50] 采用不同的方法，提出了多属性感知网络（Multiple-Attribute-Perceived Network，MAP-Net），该网络在多分支架构中融入视觉纹理属性，并聚合不同层的特征。随后 [49]，，他们使用名为深度结构揭示网络（Deep Structure-Revealed Network，DSRNet）的模型，探索纹理基元之间的空间依赖关系，以捕获图像的结构信息。Chen 等人 [4] 提出了统计自相似网络的跨层聚合（Cross-Layer Aggregation of a Statistical Self-similarity Network，CLASSNet）。该卷积神经网络特征聚合模块采用差分盒计数池化层，用于刻画纹理图像的统计自相似性。近期，Yang 等人 [48] 提出了双阶知识融合与注意力编码网络（DFAEN），利用注意力机制聚合一阶和二阶信息，以编码纹理特征。这些方法都采用微调，使骨干网络适应新的架构以及新的分类头。

[1] https://github.com/scabini/RADAM

作为卷积神经网络的替代方案，视觉Transformer（ViTs）[8] 正在视觉识别领域的文献中兴起。一些研究通过可描述纹理数据集（DTD）对其在纹理分析中的潜力进行了初步探索，并取得了当前最先进的结果。首先，与卷积神经网络相比，视觉Transformer取得了具有竞争力的结果，但由于缺乏典型的卷积归纳偏置，通常需要更多的训练数据。为解决这一问题，一种有前景的替代方案是利用注意力机制直接从有关图像的文本描述中学习，例如，使用对比语言-图像预训练（CLIP）[31]。此外，还有研究提出了用于视觉Transformer预训练的更大规模数据集，例如Bamboo，[53], 这表明这些模型能够良好地扩展。另一种方法是优化多任务大规模视觉Transformer的构建，例如Gesmundo提出的 [9] 结合 μ2Net+ 方法。

2.2 用于纹理分析的随机神经网络

最简单形式的随机神经网络 [13, 25, 26, 40], 是一种单隐层前馈神经网络，其输入权重为随机值，而输出层的权重则通过闭式解学习，这与基于梯度下降的学习形成对比。近期，有多项研究探讨了利用循环神经网络学习用于图像分析的纹理特征。Sá Junior 等人 [35] 将一幅图像中的小型局部区域作为循环神经网络的输入，并将该区域的中心像素作为目标。随后，将针对每幅图像训练得到的输出层权重用作纹理表示。Ribas 等人 [33] 在先前方法的基础上，引入图论对纹理图像进行建模，从而改进了该方法。其他研究 [16, 32] 也将这些概念扩展到了视频纹理分析（动态纹理）。

以往研究中采用的单层循环神经网络，其训练过程是在输出层求解最小二乘解。首先，将 $X \in \mathbb{R}^{n \times z}$ 视为包含 n 个训练样本和 z 个特征的输入矩阵，并将 $g = \phi(XW)$ 视为具有 Sigmoid 非线性函数的隐藏层前向传播，其中 $W \in \mathbb{R}^{z \times q}$ 表示 q 个神经元对应的随机输入权重。给定目标输出标签 Y，输出权重 f 可通过求解线性方程组的最小二乘解获得：

$$f = Y g^T (g g^T)^{-1}, \tag{1}$$

其中 $g^T(gg^T)^{-1}$ 是矩阵 g的摩尔-彭若斯伪逆 [23, 29]。

循环神经网络（RNN）的一个重要方面是为第一层生成随机权重。有证据表明，一旦权重固定，这种选择的影响就微乎其微。从这个意义上说，以往研究的一个常见趋势是使用线性同余生成器（LCG），这是一种简单的伪随机数生成器，其形式为 $x_{k+1} = (a x_k + b) \bmod c$。

将输入特征矩阵 X 视为目标输出 $Y = X$，即可将循环神经网络用作随机自编码器（RAE）[17] 。从这个意义上说，该模型由一个随机编码器和一个基于最小二乘法的解码器组成，后者可以映射输入数据。Kasun 等人 [17]还建议使用随机正交权重 [37] 来初始化编码器。这样，权重矩阵 f 就表示将投影后的随机空间变换回输入数据 X （输出）的过程。

3 RADAM：用于纹理特征编码

所提出的 RADAM 方法的核心思想是：对于经过骨干网络处理的输入图像，通过多深度特征聚合和随机逐像素编码，构成一个特征向量。首先，考虑将输入图像 $I \in \mathbb{R}^{w_0 \times h_0 \times 3}$ 输入骨干网络 $B = (d_1, \ldots, d_n)$，该网络由 n 个卷积层块组成。激活图，即给定 I 时任一卷积块的输出，是一个三维张量（为简洁起见，忽略批次维度） $X_i \in \mathbb{R}^{w_i \times h_i \times z_i}$。特征聚合过程包括组合不同深度处不同激活图的输出。为此，我们根据不同深度将骨干网络划分为固定数量的块。无论骨干网络架构的总深度如何，这种划分都旨在确保用于特征提取的块数量固定。

3.1 预训练深度卷积网络：骨干网络选择

大多数以往的纹理分析研究都将预训练残差网络 [11] (18 或 50) 作为骨干网络。本文根据 ResNet 架构，考虑五个层块的输出，也就是说，将五个激活图用于特征聚合。此外，我们还考虑了 ConvNeXt 架构 [20], , 这是一种较新的方法，在图像识别方面表现出良好潜力。对于这一骨干网络，我们根据原始研究中描述的架构，考虑来自四个层块的激活图。具体而言，使用以下 ConvNeXt 配置，以及各层块对应的通道数 (z_i)：

- ConvNeXt-nano [2]: $z_i = (80, 160, 320, 640)$.

[2]此变体未在原论文中介绍，但可在以下地址获取:
https://github.com/rwightman/pytorch-image-models/blob/main/timm/models/ConvNeXt.py

- ConvNeXt-T: $z_i = (96, 192, 384, 768)$。
- ConvNeXt-B: $z_i = (128, 256, 512, 1024)$
- ConvNeXt-L: $z_i = (192, 384, 768, 1536)$。
- ConvNeXt-XL: $z_i = (256, 512, 1024, 2048)$。

作为一种轻量级替代方案，我们也在实验中比较了 MobileNet V2 [36]（包含五个层块），分别采用原始架构或 1.4 的宽度乘数，即 $z_i = (16, 24, 32, 96, 320)$ 和$z_i = (24, 32, 48, 136, 448)$。

3.2 深度激活图准备

给定每个 对于深度激活图 X_i，我们应用按深度 l_p归一化（$p = 2$，即欧几里得范数 ）

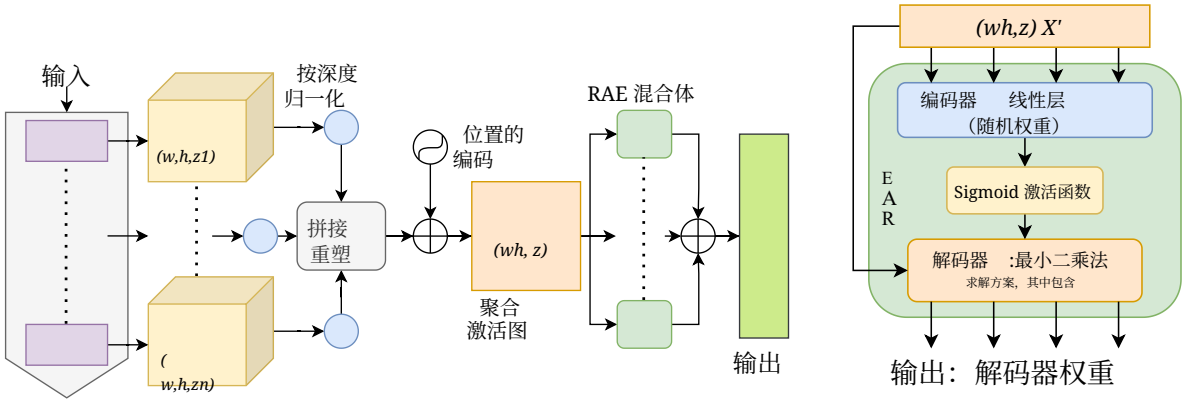
$$X_i(:,:,j) = \frac{X_i(:,:,j)}{\max(||X_i(:,:,j)||_2)}, \tag{2}$$

其中 $X_i(:,:\ j)$ 表示每个通道 $j \in z_i$ 上的二维激活图，其空间尺寸为（w_i, h_i） .
为进行特征聚合，我们提出沿第三维（z_i）拼接激活图。然而，每个图 X_i 的初始空间维度 w_i 和 h_i各不相同。为解决这一问题，我们直接使用双线性插值调整所有激活图的尺寸，并将骨干网络中间位置的空间维度 $X_{\frac{n}{2}}$（$w_{\frac{n}{2}}$, $h_{\frac{n}{2}}$）作为目标尺寸。换言之，我们将骨干网络中间位置的空间维度视为锚定尺寸，这意味着部分激活图需要上采样（若 $i > \frac{n}{2}$），其他则需要下采样（若 $i < \frac{n}{2}$）。自然地，考虑到上采样比下采样保留更多信息，因此更深层的激活图会获得更高优先级。这些假设基于卷积架构中最常见的结构，即空间尺寸随层深度增加而减小。尽管如此，我们的想法是让所有激活图保持固定的空间维度。从现在起，我们将所有 X_i的空间维度称为 $w = w_{\frac{n}{2}}$和 $h = h_{\frac{n}{2}}$。对于 224x224 的输入尺寸，这会使本文所研究的骨干网络得到 $w = h = 28$。随后，激活图的拼接方式如下

$$X' = [X_1; ...; X_n] \in \mathbb{R}^{w \times h \times z} \rightarrow X' \in \mathbb{R}^{wh \times z}, \tag{3}$$

其中 $[;.]$ 表示沿第三维进行拼接， $z = \sum_i z_i$ 是拼接后的通道数。考虑到常见的卷积架构中 $z_i < z_{i+1}$，更深层的激活图对整体 z 特征的影响更大。此外，每个通道z_i 上的二维激活图会被展平，从而重塑为大小为 wh-by-z的二维表示 X'，我们称之为聚合激活图。这些步骤如图所示。

1(a)，展示了所提出方法的整体结构。



(a) 所提出的特征编码模块（RADAM）。 (b) 随机自动编码器（RAE）。

图 1：所提出的 RADAM 架构示意图（a），展示了从输入图像到最终描述符的过程。RAE 的细节如（b）所示，它是一个通过最小二乘法求解的简单单层自动编码器，我们将解码器权重用作描述符（对 m 个随机自编码器求和）。

3.3 逐像素随机编码

使用单张图像的聚合激活图来训练 RAE，其中每个空间点（即像素（X'的一行））作为一个样本，每个通道（ X' 的一列）作为一个特征。从这个角度来看，该方法也适用于任意输入尺寸（只要骨干网络支持），因为空间维度只会影响 RAE 的训练样本数量。直观而言，较大的输入尺寸有助于 RAE 训练，但也会显著增加骨干网络的计算成本。因此，在本研究中，我们仅考虑固定的 224x224 输入尺寸（强制调整大小），因为这是各种骨干网络最常见的配置。此外，考虑到展平操作 X'会丢失像素的空间布局，我们添加了位置编码，该编码由不同频率的正弦和余弦函数组成，维度为 z，如 [42], 所提出，并按照 [43]扩展至 2 个空间维度：

$$\begin{aligned}
\mathrm{PE}(x, y, 2i) &= \sin(\frac{x}{10000^{4i/z}}),\\
\mathrm{PE}(x, y, 2i+1) &= \cos(\frac{x}{10000^{4i/z}}),\\
\mathrm{PE}(x, y, 2j+z/2) &= \sin(\frac{y}{10000^{4j/z}}),\\
\mathrm{PE}(x, y, 2j+1+z/2) &= \cos(\frac{y}{10000^{4j/z}}),
\end{aligned}$$

其中 $x \in w$ 和 $y \in h$，随后通过逐元素求和将其加到聚合激活图中：

$$X' = X' \oplus PE. \tag{4}$$

将位置编码加到 X' 后，RAE 的第一步是使用权重为 $W_k \in \mathbb{R}^{z\times q}$ 的随机全连接层对输入进行投影，然后经过 Sigmoid 非线性函数。为简化操作并提高可复现性，权重使用线性同余生成器生成，随后进行标准化（零均值、单位方差）和正交化 [37]。这些配置是根据先前的研究 [17, 33] 选定的。对于线性同余生成器的参数，我们使用 $a = 75$、$b = 74$ 和 $c = 2^{16}+1$，并以 $x = 0$ 作为起始值；这是根据 1981 年的 ZX81 计算机采用的经典配置。这里的 k 类似于随机采样的种子，表示在使用给定配置生成的线性同余生成器空间中的起始索引。补充材料中提供了有关线性同余生成器权重的更多细节，例如对不同线性同余生成器配置影响的消融实验。随后，所有样本的编码器 $g_k \in \mathbb{R}^{wh\times q}$ 前向传播可表示为

$$g_k = \phi(X'W_k)\,, \tag{5}$$

而解码器权重 $f_k \in \mathbb{R}^{z\times q}$ 则通过式（1）中所述的最小二乘解获得，将目标值改为Y： X':

$$f_k = X' g_k^T (g_k g_k^T)^{-1}\,. \tag{6}$$

为每幅图像采用一个独立的随机神经网络，其核心思想是将输出权重本身用作表征。对于随机自编码器（RAE），输出层与输入层的维度相同。因此，为了保持维度不变，使用单个隐藏神经元（$q = 1$）。由此，得到的解码器权重表示为

$$f_k = (\nu_1, \ldots, \nu_z)\,, \tag{7}$$

其中，ν_i 表示单个隐藏神经元与输出层 i之间的连接权重，对应于特征$i \in z$。

单神经元 RAE 在编码深度激活图中包含的足够信息方面可能存在局限。因此，我们提出采用模型集成，或者如近期提出的 [45], “模型混合”，即组合 m 并行模型的权重。在这里，每个模型都是一个使用不同随机编码器（采用不同的线性同余生成器种子）的 RAE，而组合方式是将解码器权重相加。

$$\varphi_m = (\sum_{k=1}^{m} f_k(\nu_1), \ldots, \sum_{k=1}^{m} f_k(\nu_z))\,. \tag{8}$$

需要注意的是，这些编码器 g_k 中每个 m RAE 的随机权重初始化都各不相同。为此，我们生成一个大小为 mz 的线性同余生成器序列，从而为每个 z RAE 设置 m 个权重。RAE 的结构如图 1(b)所示。按照图 1(a)所示的完整 RADAM 流程处理后，可获得输入图像的纹理表示，即特征向量 φ_m。所有这些步骤的代码均可在补充材料和我们的在线代码库 [1]中找到。随后，使用特征向量 φ_m 训练线性分类器，以完成给定的纹理识别任务（有关分类器的更多细节，请参见第 4.2.1节）。

4 实验与结果

4.1 实验设置

我们的模型使用 PyTorch [27] 实现（分类步骤除外），这使得 RADAM 更容易与该库中实现的多种方法结合。分类步骤使用 Scikit-learn [28]完成。我们以平均分类准确率及相应的标准差衡量结果（视数据集情况而定）。对于骨干网络，我们采用 PyTorch Image Models 库 [44]（版本 0.6.7），其中包含多种经过预训练的计算机视觉方法。补充材料中给出了 RADAM 的主要代码；完整实现（包括实验脚本）可在我们的 GitHub 代码库 [1]中查看。

本文使用七个纹理数据集进行评估，其中以下两个 Outex 数据集变体 [14] 仅用于分析 RADAM 方法：

- *Outex10*：由 24 种不同纹理类别中的 4320 张灰度图像组成。该数据集侧重于旋转不变性；
- *Outex13*：该数据集包含 68 种纹理类别中的 1360 张 RGB 图像，用于评估彩色纹理识别。

以下五个数据集用于与其他方法进行比较：

- 可描述纹理数据集（*DTD*） [5]：由 5640 张图像组成，涵盖 47 种不同的纹理类别，并通过提供的 10 个划分进行训练、验证和测试；
- *FlickrMaterialDataset*（*FMD*） [41]：包含 1000 张图像，代表 10 种材料类别，并通过重复 10 次的 10 折交叉验证进行验证；
- *KTH-TIPS2-b* [3]：包含 4752 张图像，涵盖 11 种不同的材料。该数据集有一组固定的 4 个划分，用于 4 折交叉验证；
- 户外场景地面纹理（*GTOS*） [47]：该数据集包含 34105 张图像，分为 40 类户外地面材料。此外，还提供了一组固定的 5 个训练/测试划分；
- *GTOS-Mobile* [46]：包含使用手机拍摄的 100011 张图像，涵盖 31 种不同的户外地面材料，并采用单一训练/测试划分。

4.2 RADAM 属性分析

我们的首次实验评估关注所提出的 RADAM 方法的若干方面。在补充材料中，我们进一步分析了不同随机权重（LCG 配置）造成的影响，并得出结论：这些影响极小，这与以往研究的结果相符。接下来，我们将评估并讨论 RADAM 的其他方面。

4.2.1 位置编码与不同分类器

我们评估 RADAM 流水线的两项设计选择：是否使用位置编码，以及分类器的选择。我们在两种不同的分类器下比较使用和不使用编码的情况：线性判别分析（LDA） [34]和支持向量机（SVM） [30]。对于 LDA，我们采用使用 Ledoit–Wolf 引理进行自动收缩的最小二乘解；对于 SVM，则采用线性核和 $C = 1$ 。由于位置编码的评估涉及纹理的空间属性，我们采用侧重于旋转不变性的 Outex10 基准。如表1 所示，位置编码在所有情况下都能提升或保持性能，尤其是在使用 LDA 分类器时。另一方面，SVM 在所有情况下均取得最佳结果，但从位置编码中获得的提升较小。尽管如此，我们仍在架构中保留位置编码，因为与其潜在收益相比，额外开销可以忽略不计。后续所有实验也都使用 SVM 作为分类器。

4.2.2 组合规模

所提出的 RADAM 方法唯一的自由参数是要组合的随机自编码器数量，即 m。我们评估了m 从 1 到 32 的取值，结果如图所示。2 展示了 Outex13 数据集中不同骨干网络的结果。我们观察到，当 m 从 1 增加到 4 时，性能显著提升；而取值更大时，性能趋于稳定。这些结果表明，$4 \leq m \leq 8$ 是平衡性能与成本的良好选择，因为超过该值后性能没有显著提升。本文后续所有实验均采用 $m = 4$。

表1：采用不同骨干网络的 RADAM（$m = 1$）在位置编码和不同分类器方面的消融实验。结果根据 Outex10 的单一训练/测试划分，以分类准确率衡量。

编码	骨干网络	Outex10 LDA	Outex10 SVM
none	ResNet18	77.5	83.4
位置编码	ResNet18	79.5	84.7
none	ConvNeXt-nano	82.4	89.6
位置编码	ConvNeXt-nano	85.9	89.6
none	ResNet50	86.0	87.4
位置编码	ResNet50	87.4	87.4
none	ConvNeXt-T	87.8	91.1
位置编码	ConvNeXt-T	89.4	91.1

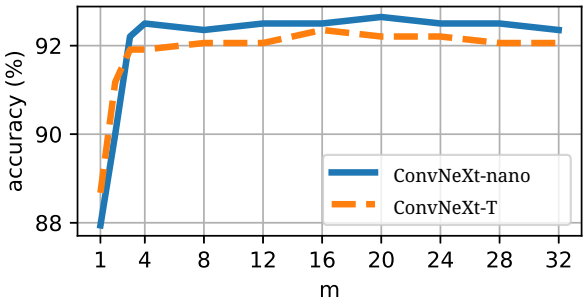


图 2：在 Outex13 数据集上，针对不同骨干网络，改变所提出方法中 RAE 的数量（m）所产生的影响。本文的所有其他实验均采用 $m = 4$。

增加 m 时观察到的效果符合模型集成（或“模型汤”）中通常出现的情况，[45], 即分别训练的模型组合起来可能会带来益处。另一方面，我们的编码器是随机的，且各自具有不同的权重。不过，即使每个编码器都会对输入进行不同的随机投影，解码器仍会学习将投影变换回相同的特征空间。换言之，随机自编码器会针对相同输入学习不同的编码—解码函数；在我们的特征提取应用场景中，将这些函数组合起来能够得到更好的表征。

4.3 与其他池化技术的比较

为展示 RADAM 相较于常见池化方法的优势，我们使用与 RADAM 相同的配置，采用全局平均池化（GAP）结合支持向量机（SVM）进行了补充实验。我们对骨干网络最后一层的输出应用 GAP（这是大多数卷积神经网络架构中常用的方法），并且还使用 GAP 聚合，即聚合每个 n 特征块的 GAP，并返回一个包含 z 个特征的图像表示（与 RADAM 相同）。结果见表2。可以看到，在所有骨干网络和数据集上，RADAM 都以显著优势胜过这两种方法。GAP 聚合通常优于仅对最后一个卷积层使用常规 GAP，这符合预期，因为它加入了更多特征。然而，有时常规 GAP 的表现优于这种简单的拼接聚合。另一方面，使用 RADAM 进行聚合带来的提升更加显著，并在我们考察的所有基准测试中达到最先进水平，下一节将对此进行讨论。

4.4 与文献对比

最后，我们在五个具有挑战性的纹理识别数据集上，将 RADAM 与几种达到最先进水平的方法进行比较；所有结果均列于表 3 中。表格根据不同骨干网络的计算预算分成若干行。我们注明了骨干网络所用的预训练数据集；若未特别说明，则使用 ImageNet-1K [7]。首先，我们展示使用 RADAM 和 MobileNet V2（标准版本或宽度乘数为 1.4 的版本）作为轻量级替代方案所获得的结果（其开销将在第 4.5 节中深入讨论）。第一部分比较了使用 ResNet18 的方法；考虑到 ConvNeXt-nano 的计算预算与之相近，我们也将搭配 ConvNeXt-nano 的 RADAM 纳入这一部分

表 2：使用不同技术从骨干网络中提取图像特征时的分类准确率（线性支持向量机）：GAP、骨干网络各个 n 特征块的 GAP 聚合（采用与 RADAM 相同的处理方式），以及所提出的 RADAM。我们在两个具有挑战性的纹理基准数据集（DTD 和 FMD）上进行评估。

方法	骨干网络	DTD	FMD
GAP	ResNet18	64.1±1.1	77.1±0.7
GAP 聚合	ResNet18	64.6±1.2	77.2±0.6
RADAM	ResNet18	68.1±1.0	77.7±0.5
GAP	ConvNeXt-nano	61.9±1.0	73.8±0.9
GAP 聚合	ConvNeXt-nano	71.7±0.6	84.4±0.4
RADAM	ConvNeXt-nano	74.9±0.7	87.1±0.4
GAP	ResNet50	72.6±0.9	84.5±0.6
GAP 聚合	ResNet50	74.6±0.6	83.4±0.4
RADAM	ResNet50	75.6±1.1	85.3±0.4
GAP	ConvNeXt-T	66.0±1.0	79.0±0.4
GAP 聚合	ConvNeXt-T	73.7±0.9	83.6±0.5
RADAM	ConvNeXt-T	77.0±0.7	88.7±0.4
GAP	ImageNet-21K 上的 ConvNeXt-T	78.4±0.7	91.4±0.3
GAP 聚合	ImageNet-21K 上的 ConvNeXt-T	77.3±0.9	89.9±0.5
RADAM	ImageNet-21K 上的 ConvNeXt-T	81.4±0.7	93.0±0.3
GAP	ConvNeXt-B	69.2±1.3	83.6±0.5
GAP 聚合	ConvNeXt-B	73.5±1.0	86.7±0.5
RADAM	ConvNeXt-B	76.4±0.9	90.2±0.2
GAP	ImageNet-21K 中的 ConvNeXt-B	79.2±0.6	91.9±0.4
GAP 聚合	ImageNet-21K 上的 ConvNeXt-B	80.4±1.0	92.3±0.3
RADAM	ImageNet-21K 上的 ConvNeXt-B	82.8±0.9	94.0±0.2
GAP	ConvNeXt-L	70.9±0.9	84.6±0.4
GAP 聚合	ConvNeXt-L	73.4±0.6	86.4±0.5
RADAM	ConvNeXt-L	77.4±1.1	89.3±0.3
GAP	ImageNet-21K 上的 ConvNeXt-L	80.4±0.9	92.2±0.4
GAP 聚合	ImageNet-21K 上的 ConvNeXt-L	81.9±0.6	93.4±0.4
RADAM	ImageNet-21K 上的 ConvNeXt-L	84.0±1.0	95.2±0.4
GAP	ImageNet-21K 中的 ConvNeXt-XL	81.3±1.0	92.4±0.3
GAP 聚合	ImageNet-21K 中的 ConvNeXt-XL	82.0±1.1	93.9±0.4
RADAM	ImageNet-21K 上的 ConvNeXt-XL	83.7±0.9	95.2±0.3

。在这种情况下，使用 ResNet18 的 RADAM 在 KTH 上取得了有竞争力的表现，但在其他数据集上的效果较差。另一方面，使用 MobileNetV2 1.4 的 RADAM 在除 GTOS-Mobile 之外的所有情况下都优于 ResNet18，并且在 DTD、FMD 和 KTH 上胜过所有采用 ResNet18 的文献方法，证明它是一种出色的低成本方法。使用 ConvNeXt-nano 时，RADAM 在除 GTOS 和 GTOS-Mobile 之外的所有数据集上也都取得了最佳结果。

考虑 ResNet50 的计算预算下的结果，RADAM 的表现远优于 ResNet18。在大多数数据集上，使用 ResNet50 都取得了有竞争力的结果，并且在 KTH 上取得了最佳结果。使用 ConvNeXt-T 时，RADAM 在 FMD 上达到了最先进水平，在 KTH 上也超过了所比较的方法。若考虑在 ImageNet-21K 上预训练的 ConvNeXt-T，性能会进一步提升，并在大多数数据集上取得最佳结果。这些结果表明，对所使用的骨干网络进行更好的预训练具有提升潜力。

表 3 最后几行所示的结果涉及与前述方法相比成本更高的方法。这里，我们将使用 ConvNeXt-B、L 和 XL 的 RADAM 与若干研究进行比较，其中包括 ViTs 等非常新近的方法。在这种情况下，我们的方法在所有数据集上都取得了最先进的结果，尤其是在采用 ConvNeXt-L/XL 骨干网络时。再次可以看到，使用 ImageNet-21K 进行更好的预训练能够显著提升性能。

4.5 特征提取成本与性能

我们还进行了额外分析，以更好地理解所比较的纹理识别方法在分类性能与计算预算之间的平衡。我们根据 GFLOPs 衡量推理成本，并

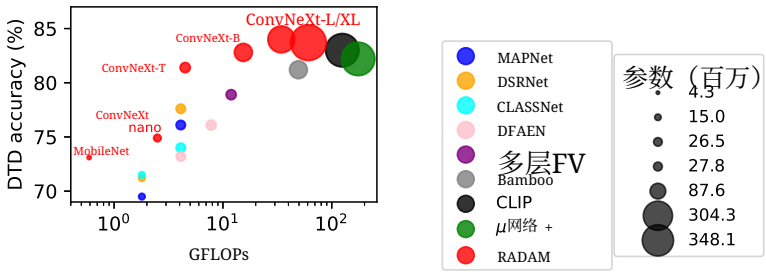
表3：不同方法在纹理基准数据集上的分类准确率。根据计算预算，将所用骨干网络分组排列在不同的行块中；输入尺寸在括号中注明（未注明时为 224x224），每个行块中表现最好的两项结果以粗体突出显示。结果中蓝色 表示各数据集此前的最先进水平，红色 则表示我们的结果达到或超过了该水平。

方法	骨干网络	DTD	FMD	KTH-2-b	GTOS	GTOS-Mobile
RADAM light（我们的）	MobileNet V2	69.3±0.8	80.5±0.6	84.8±0.5	81.0±1.3	75.1
RADAM light（我们的方法）	MobileNet V2 1.4	73.1±0.9	82.6±0.5	86.8±3.1	81.7±1.7	78.2
DeepTEN	ResNet18 (352)					76.1
DEPNet	ResNet18					82.2
MAPNet	ResNet18	69.5±0.8	80.8±1.0	80.9±1.8	80.3±2.6	83.0
DSRNet	ResNet18	71.2±0.7	81.3±0.8	81.8±1.6	81.0±2.1	**83.7**
CLASSNet	ResNet18	**71.5**±0.4	**82.5**±0.7	**85.4**±1.1	**84.3**±2.2	**85.3**
RADAM（我们的方法）	ResNet18	68.1±1.0	77.7±0.5	84.7±3.6	80.6±1.7	79.5
RADAM（我们的方法）	ConvNeXt-纳米型	**74.9**±0.7	**87.1**±0.4	**89.6**±3.8	**83.7**±1.5	81.8
DeepTEN	ResNet50 (352)	69.6	80.2±0.9	82.0±3.3	84.5±2.9	
MAPNet	ResNet50	76.1±0.6	85.2±0.7	84.5±1.3	84.7±2.2	86.6
DSRNet	ResNet50	**77.6**±0.6	86.0±0.8	85.9±1.3	85.3±2.0	**87.0**
CLASSNet	ResNet50	74.0±0.5	86.2±0.9	87.7±1.3	**85.6**±2.2	85.7
DFAEN	ResNet50	73.2	86.9	86.3		**86.9**
RADAM（我们的方法）	ResNet50	75.6±1.1	85.3±0.4	88.5±3.2	81.8±1.1	81.0
RADAM（我们的方法）	ConvNeXt-T	77.0±0.7	**88.7**±0.4	**90.7**±4.0	84.2±1.7	85.3
RADAM（本文方法）	ImageNet-21K 中的 ConvNeXt-T	**81.4**±0.7	**93.0**±0.3	**91.0**±4.9	**85.4**±1.6	86.5
DFAEN	DenseNet161	76.1	87.6	86.6		86.9
多层FV	EfficientNet-B5（512）	78.9	88.7	82.9		
RankGP-3M-CNN++	（3 个骨干网络）		86.2±1.4	91.1±4.5		
微调	Bamboo 中的 ViT B/16 69m	81.2				
CLIP 零样本	WIT 400m 中的 ViT L/14 (336)	83.0				
μ2Net+	ImageNet-21K 中的 ViT-L/16 (384)	82.2				
RADAM（我们的方法）	ConvNeXt-B	76.4±0.9	90.2±0.2	87.7±5.6	84.1±1.6	82.2
RADAM（我们的方法）	ImageNet-21K 上的 ConvNeXt-B	82.8±0.9	94.0±0.2	**91.8**±4.1	**86.6**±1.7	87.1
RADAM（本文方法）	ConvNeXt-L	77.4±1.1	89.3±0.3	89.3±3.4	84.0±1.8	85.8
RADAM（我们的方法）	ImageNet-21K 上的 ConvNeXt-L	**84.0**±1.0	**95.2**±0.4	91.3±4.1	85.9±1.6	**87.3**
RADAM（我们的方法）	ImageNet-21K 上的 ConvNeXt-XL	**83.7**±0.9	**95.2**±0.3	**94.4**±3.8	**87.2**±1.9	**90.2**

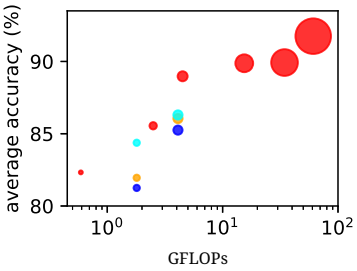
根据各方法所使用的骨干网络计算参数数量，因为这是每条流水线中资源需求最高的步骤。这里的一个重要方面是，输入尺寸会显著影响各方法的浮点运算次数（请查看表 3 中括号内的输入尺寸）。大多数研究采用 224x224 的输入（与 RADAM 使用的输入尺寸相同）；作者未说明时，我们也假设使用这一尺寸。在此分析中，我们也未考虑骨干网络的准备工作，无论是预训练成本还是所选数据集，也未考虑那些采用微调的方法。结果见图 3，首先是仅针对 DTD 数据集的结果（图 3(a)），因为这是最具挑战性的任务，且可供比较的方法更多；随后是其余数据集的平均结果（图3(b)）。可以看出，RADAM 在不同预算下均表现出优势，尤其是在使用 MobileNet V2 1.4 和 ConvNeXt-T 时。此外，结果也会随骨干网络复杂度的提高而提升。

还需要特别指出，由于缺乏可用结果，本分析未涵盖其他方法的骨干网络成本。例如，多层FV曾在FMD上使用输入尺寸为512像素的EfficientNet-B5取得此前最先进水平，其推理成本约为12 GFLOPs。这一成本明显高于ConvNeXt-T的成本（4.5 GFLOPs）；相比多层FV，RADAM使用ConvNeXt-T将绝对性能提升了4.3%。在KTH上，RankGP-3M-CNN++ 使用三个骨干网络，推理总成本约为30 GFLOPs，取得了91.1%的准确率（此前的最先进水平）；而RADAM使用ConvNeXt-T取得了可比的结果（-0.1%），并且使用ConvNeXt-B取得了最先进的结果（91.8%，推理成本为15 GFLOPs），使用ConvNeXt-XL也取得了最先进的结果（94.4%，推理成本为61 GFLOPs）。对于GTOS和GTOS-Mobile，RADAM使用ConvNeXt-T同样实现了具有竞争力的成本和性能，并使用ConvNeXt-B、L和XL取得了最先进的结果。

为了补充成本分析，我们在表4 中展示了 RADAM（搭配 $m = 4$ 和支持向量机）的实际运行时间，并与 ResNet 的成本进行对比。我们的思路是比较 RADAM 的使用成本（它只需进行一次骨干网络前向传播并使用支持向量机）与以往工作中对骨干网络进行微调的方法。我们测量了 ResNet18 和 ResNet50 对单张 224x224 RGB 图像运行 100 次的平均时间（毫秒），考察 RADAM 模块、ResNet 前向传播和反向传播，以及微调骨干网络时的相应耗时



（a）DTD 数据集上的最佳方法。图中的文字表示与 RADAM 配合使用的骨干网络。



（b）在其他四个数据集上有结果的方法的平均性能。

图3：不同方法采用的骨干网络对应的推理时计算预算。

表4：在配备 GTX 1080ti、Intel Core i7-7820X 3.60GHz 处理器（使用 8 个线程）和 64GB 内存的机器上，使用一张 224x224 RGB 图像进行 100 次重复运行的平均运行时间（毫秒）。

方法	骨干网络	时间（毫秒） CPU	GPU
仅 RADAM 模块	ResNet18	2.7±0.2	2.9±0.04
骨干网络的前向传播	ResNet18	15.0±1.6	4.3±0.1
骨干网络的反向传播	ResNet18	50.8±5.3	9.9±1.5
骨干网络的一个微调周期	ResNet18	65.8±6.0	14.2±1.6
骨干网络的前向传播 + RADAM +支持向量机推理	ResNet18	19.8±1.6	8.1±0.1
仅 RADAM 模块	ResNet50	7.0±0.4	5.2±0.4
骨干网络的前向传播	ResNet50	34.0±2.6	11.1±0.8
骨干网络的反向传播	ResNet50	107.6±5.9	21.5±2.2
骨干网络的一个微调周期	ResNet50	141.7±7.6	32.6±2.3
骨干网络的前向传播 + RADAM + 支持向量机推理	ResNet50	50.3±2.6	18.1±0.8

使用 RADAM 进行特征提取并执行支持向量机推理。需要注意的是，RADAM 的成本会随骨干网络而变化，因为聚合特征的数量（z）会发生变化，从而影响 RAE 训练成本。根据表中的结果，在 GTOS-Mobile 上使用批量大小为 1、仅训练 10 轮，对 ResNet50 进行微调，在 CPU 上约需 40 小时，在 GPU 上约需 9 小时。相比之下，使用 RADAM 提取特征，然后在整个 GTOS-Mobile 数据集上进行支持向量机推理，在 CPU 上约需 1.3 小时，在 GPU 上约需半小时。在整个数据集上训练支持向量机（使用 CPU）平均还需额外 15 分钟，且未进行超参数调优。结果表明，无论在推理速度方面，还是在比较特征提取后接支持向量机与训练/微调的耗时方面，RADAM 模块都明显快于 ResNet 骨干网络。对于 ConvNeXt-nano 和 ConvNeXt-T，结果同样成立，因为它们的成本分别与 ResNet18 和 ResNet50 相当，这印证了我们的主张：RADAM 不仅能大幅节省训练时间，还能以相近的推理成本取得最先进的结果。此外，对于 ConvNeXt-L 和 XL 等成本更高的骨干网络，训练时间上的优势可能更加显著。

5 结论

我们提出了 RADAM，一种用于纹理分析的新型特征编码模块。该方法使用 RAE 对预训练深度卷积神经网络提取的聚合深度激活图进行随机编码。这些自编码器以其 z 维像素作为样本点进行训练，从而学习将激活图汇聚为一维表示。随后，利用从该纹理图像激活图中学习到的解码器权重对纹理图像进行编码。该过程不考虑顺序，但通过使用二维位置编码纳入像素的空间信息。与以往工作相比，我们的方法无需对骨干网络进行微调，且编码模块相当简单。我们使用支持向量机对描述符进行线性分类，并在多个纹理基准数据集上取得了最先进水平的表现。对于计算预算各异的骨干网络，RADAM 在推理成本与性能方面也实现了最佳效率。考虑到与其他方法相比，RADAM 无需微调骨干网络，因此训练成本也更低，这些结果尤为出色。

我们的工作证实了一种更简单的纹理识别方法：无需对成本高昂的骨干网络进行微调，也能获得较强的判别能力。未来的工作可以探索不同的骨干网络，或尝试对我们的 RAE 采用不同的设计，例如使用多个层、增加隐藏神经元，或进行其他改进。另一方面，如果计算资源充足，也可以采用一种更接近以往工作的方案，以端到端方式探索我们的模块。由于 RADAM 是确定性的，且具有闭式解，一种替代方案是添加线性层而非支持向量机，并与骨干网络一同对其进行优化。

致谢

L. Scabini 感谢圣保罗研究基金会（FAPESP）提供资助（资助编号：#2019/07811-0 和 #2021/09163-6），并感谢国家科学技术发展委员会（CNPq）提供资助（资助编号：#142438/2018年- 9）。K M. Zielinski 感谢圣保罗研究基金会（FAPESP）提供支持（资助编号：#2022/03668-1），并感谢高等教育人才培养协调机构（CAPES）提供支持（资助编号：# 88887.631085/2021年-00）。L. C. Ribas 感谢圣保罗研究基金会（FAPESP）提供支持（资助编号：#2021/07289-2 和 # 2018/22214-6）。W. N. Gonçalves 感谢 CNPq 提供支持（资助编号：#405997/2021-3 和 #305296/2022-1）。O. M. Bruno 感谢 CNPq 提供支持（资助编号：#307897/2018-4）和 FAPESP 提供支持（资助编号：#2018/22214-6 和 # 2021/08325-2）。作者还感谢 NVIDIA GPU资助计划。本研究获得佛兰德政府通过“佛兰德人工智能研究计划”项目提供的资助。

参考文献

[1] Robert Azencott、Jia-Ping Wang 和 Laurent Younes。使用窗函数傅里叶滤波器进行纹理分类。IEEE模式分析与机器智能汇刊，19 2 148 153, 1997 1 ( ): –。André Ricardo Backes、Dalcimar Casanova 和 Odemir Martinez Bruno。基于[2]分形描述符的彩色纹理分析。模式识别，45 5 1984 1992, 2012 1 ( ): –。B. Caputo、E. Hayman 和 P. Mallikarjuna。特定类别的材料分类。载于第十届IEEE国际计算机视觉会议（ICCV’[3] *05* *1*, 2, 1597 1604 2, 2005 6 ），第卷，页码 –，第 卷。Zhile Chen、Feng Li、Yuhui Quan、Yong Xu 和 Hui Ji[4]。通过利用跨层统计自相似性进行深度纹理识别‡。载于2021年IEEE/CVF计算机视觉与模式识别会议（CVPR），第5227–5236页，2021年。1, 2[5] Mircea Cimpoi、Subhransu Maji、Iasonas Kokkinos、Sammy Mohamed 和 Andrea Vedaldi。描述真实场景中的纹理。载于 *2014* 3606 3613, 2014 1, 2, 6 IEEE计算机视觉与模式识别会议，第 –页。Mircea Cimpoi[6]、Subhransu Maji 和 Andrea Vedaldi。用于纹理识别与分割的深度滤波器组。载于2015年IEEE计算机视觉与模式识别会议（CVPR），第 3828 3836, 2015 1, 2 –页。Jia Deng、Wei Dong、Richard Socher、Li-Jia Li、Kai Li[7] 和 Li 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，第248–255页。IEEE，2009年。7[8] Alexey Dosovitskiy、Lucas Beyer、Alexander Kolesnikov、Dirk Weissenborn、Xiaohua Zhai、Thomas Unterthiner、Mostafa Dehghani、Matthias Minderer、Georg Heigold、Sylvain Gelly、Jakob Uszkoreit 和 Neil Houlsby。一张图像胜过16×16个词：用于大规模图像识别的变换器。CoRR，abs/ 2010 11929, 2020 3 。。Andrea Gesmundo。大规模多任务动态机器学习系统的持续开发方法。[9] arXiv预印本 arXiv:2209.07326，2022年。3[10] 67 5 786 804, 1979 1 Robert M Haralick[11]。纹理的统计方法和结构方法。IEEE论文集，( ): –。Kaiming He、Xiangyu Zhang、Shaoqing Ren 和 Jian Sun。用于图像识别的深度残差学习。载于IEEE计算机视觉与模式识别会议论文集，第 770 778, 2016 3 –页。Minh A Hoang、Jan-Mark Geusebroek 和 Arnold WM Smeulders[12]。彩色纹理测量与分割。信号处理，85 2 265 275, 2005 1 ( ): –。Guang-Bin Huang[13]、Qin-Yu Zhu 和 Chee-Kheong Siew。极限学习机：理论与应用。神经计算，70(1):489–501，2006年。1, 3[14] S. Huovinen、M. Pietikäinen、T. Ojala、J. Kyllönen、J. Viertola 和 T. Mäenpää。Outex——用于纹理分析算法经验评估的新框架。载于第16届国际模式识别会议论文集，第1卷，第10701页，美国加利福尼亚州洛斯阿拉米托斯，8月 2002 6 。IEEE计算机学会。Bela Julesz。视觉模式判别。IRE 信息论汇刊，[15] 8 2 84 92, 1962 1 ( ): –。Jarbas Joaci de Mesquita Sá J unior、Lucas Correia Ribas[16] 和 Odemir Martinez Bruno。用于动态纹理分类的基于随机神经网络的签名。Expert Systems with Applications，135 194 200, 2019 3 : –。Liyanaarachchi Lekamalage Chamara Kasun、Hongming Zhou[17]、G-B Huang 和 Chi Man Vong。面向大数据的极限学习机表征学习。IEEE智能系统，28 31 34, 2013 1, 3, 5 : –。Tsung-Yu Lin、Aruni RoyChowdhury 和 Subhransu Maji。用于细粒度视觉识别[18]的双线性卷积神经网络模型。载于2015年IEEE国际计算机视觉会议（ICCV），第 1449 1457, 2015 2 –页。Li Liu、Jie Chen、Paul Fieguth、Guoying Zhao、Rama Chellappa[19] 和 Matti Pietikäinen。从词袋到卷积神经网络：纹理分类中纹理表示的二十年发展。国际计算机视觉杂志，127(1):74–109，2018年11月。2

[20] Zhuang Liu、Hanzi Mao、Chao-Yuan Wu、Christoph Feichtenhofer、Trevor Darrell 和 Saining Xie。面向2020年代的卷积神经网络。载于IEEE/CVF计算机视觉与模式识别会议论文集，第

11976 11986, 2022 3 –页。Lucas O. Lyra、[21] Antonio Elias Fabris 和 Joao B. Florindo。用于视觉纹理识别的多层深度特征提取，2022年。2[22] Rayner H. M. Condori 和 Odemir M. Bruno。通过全局池化测量分析用于纹理分类的激活图。信息科学，

555 260 279, 2021 1, 2 : –。[23] E. H. Moore。论一般代数矩阵的逆。美国数学会通报，26 395，1920 394 3：–。[24] Timo Ojala、Matti Pietikainen 和 Topi Maenpaa。使用局部二值模式进行多分辨率灰度及旋转不变纹理分类。IEEE模式分析与机器智能汇刊，

24 7 971 987, 2002 1 ( ): –。Yoh-Han Pao、[25] Gwang-Hoon Park 和 Dejan J Sobajic。随机向量函数连接网络的学习与泛化特性。神经计算，

6 2 163 180, 1994 1, 3 ( ): –。Y-H Pao 和 Yoshiyasu Takefuji。函数连接网络计算：理论、系统架[26] 构与功能。《计算机》，25(5):76–79，1992年。1, 3[27] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Kopf、Edward Yang、Zachary DeVito、Martin Raison、Alykhan Tejani、Sasank Chilamkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。PyTorch：一种命令式风格、高性能的深度学习库。载于神经信息处理系统进展32，第8024–8035页。Curran Associates, Inc.，2019年。6[28] F. Pedregosa、G. Varoquaux、A. Gramfort、V. Michel、B. Thirion、O. Grisel、M. Blondel、P. Prettenhofer、R. Weiss、V. Dubourg、J. Vanderplas、A. Passos、D. Cournapeau、M. Brucher、M. Perrot 和 E. Duchesnay。Scikit-learn：Python机器学习。机器学习研究杂志，

12 2825 2830, 2011 6 : –。R. Penrose。矩阵[29] 的广义逆。剑桥哲学学会数学会报，51(3):406— -413，1955年。3[30] John Platt 等。支持向量机的概率输出及与正则化似然方法的比较。大间隔分类器进展，10(3):61–74，1999年。6[31] Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sandhini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark、Gretchen Krueger 和 Ilya Sutskever。通过自然语言监督学习可迁移视觉模型，

2021 3 。Lucas C Ribas、Jarbas Joaci de Mesquita Sá Junior、Antoine Manzanera 和[32] Odemir M Bruno。使用随机神经网络学习用于动态纹理分类的图表示。应用软计算，

114 108035, 2022 3 :。Lucas C. Ribas、Jarbas[33] Joaci Mesquita Sá Junior、Leonardo F.S. Scabini 和 Odemir M. Bruno。复杂网络与随机神经网络融合用于纹理分析。模式识别，

103 107189, 2020 3, 5 :。Brian D Ripley。模式识别与神经网络。[34] 2007 6 剑桥大学出版社，。[35] Jarbas Joaci Mesquita Sá Junior 和 André Ricardo Backes。基于极限学习机的纹理分类签名。模式识别，51:395–401，2016年。3[36] Mark Sandler、Andrew Howard、Menglong Zhu、Andrey Zhmoginov 和 Liang-Chieh Chen。MobileNetV2：倒置残差与线性瓶颈。载于IEEE计算机视觉与模式识别会议论文集，第4510 4520页，2018年4 –。[37] Andrew M Saxe、James L McClelland 和 Surya Ganguli。深度线性神经网络学习非线性动力学的精确解。arXiv预印本 arXiv:

1312 6120, 2013 3, 5 。。Leonardo FS Scabini、[38] Rayner HM Condori、Wesley N Gonçalves 和 Odemir M Bruno。用于颜色纹理表征的多层复杂网络描述符。信息科学，

491 30 47, 2019 1 : –。Leonardo FS Scabini、Lucas C Ribas 和[39] Odemir M Bruno。用于颜色纹理分析的时空谱网络。信息科学，515:64–79，2020年。1[40] Wouter F Schmidt、Martin A Kraaijveld 和 Robert P W Duin。具有随机权重的前馈神经网络。载于第 11 届 IAPR 国际模式识别会议论文集。第 II 卷。B 会议：模式识别方法论与系统，第

1 4, 1992 1, 3 –页。Lavanya Sharan、Ruth Rosenholtz 和 EH Adelson。材料感知：一瞥之间，你能看到什么？[41] 视觉杂志 - J VISION，9:784–784，2010年8月。6[42] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力机制就是你所需要的一切。神经信息处理系统进展，

30, 2017 5 。Zelun Wang 和 Jyh-Charn Liu。使用[43] 序列级训练的深度神经网络将数学公式图像转换为LaTeX序列。国际文档分析与识别期刊（IJDAR），

24 1 63 75, 2021 5 ( ): –。Ross Wightman。[44] 2019 6 PyTorch图像模型。https://github.com/rwightman/pytorch-image-models，。[45] Mitchell Wortsman、Gabriel Ilharco、Samir Ya Gadre、Rebecca Roelofs、Raphael Gontijo-Lopes、Ari S Morcos、Hongseok Namkoong、Ali Farhadi、Yair Carmon、Simon Kornblith 等。模型汤：对多个微调模型的权重取平均，无需增加推理时间即可提高准确率。载于国际机器学习会议，第23965–23998页。PMLR，2022年。5, 7[46] Jia Xue、Hang Zhang 和 Kristin Dana。用于地面地形识别的深度纹理流形。载于2018年IEEE/CVF计算机视觉与模式识别会议。IEEE，2018年6月

2018 1, 2, 6 。Jia Xue、Hang Zhang、Ko[47] Nishino 和 Kristin Dana。用于地面地形材料识别的差分视点。IEEE模式分析与机器智能汇刊，PP，2020年9月。1, 2, 6

[48] Zhijing Yang、Shujian Lai、Xiaobin Hong、Yukai Shi、Yongqiang Cheng 和 Chunmei Qing。DFAEN：用于纹理识别的双阶知识融合与注意力编码网络。*Expert Systems with Applications*，209 118223, 2022 1, 2：。Wei Zhai、Yang Cao、Zheng-Jun Zha、HaiYong Xie 和 Feng Wu。用于纹理识别的深度结构揭示网络。发表于*2020* 11007 11016, 2020 1, 2 *IEEE/CVF*计算机视觉与模式识别会议（*CVPR*），页码 –。[49] Wei Zhai、Yang Cao、Jing Zhang 和 Zheng-Jun Zha。用于真实场景纹理识别的深层多属性感知网络。发表于*2019* 3612 3621, 2019 1, 2 *IEEE/CVF*国际计算机视觉会议（*ICCV*），页码 –。H. Zhang、J. Xue 和 K. Dana。Deep ten：纹理编码网络。发表于*IEEE*计算机视觉与模式识别会议（*CVPR*），页码 2896 2905, 2017 1, 2 – 美国加利福尼亚州洛斯阿拉米托斯，7月。IEEE计算机学会。[51] Jianguo Zhang 和 Tieniu Tan。不变纹理分析方法简述。模式识别，（）：–747，2002。I[52] Yuanhan Zhang、Qinghong Sun、Yichun Zhou、Zexin He、Zhenfei Yin、Kun Wang、Lu Sheng、Yu Qiao、Jing Shao 和 Ziwei Liu。Bamboo：利用人机协同持续构建超大规模视觉数据集，2022年。3 35 3 735

RADAM 补充材料：通过深度激活图的随机聚合编码进行纹理识别

1 RADAM 实现

我们给出了 RADAM 模块及其所有内部机制的精确 Python 代码。未来，我们将通过公共代码仓库公开此实现及相关实验。首先，RADAM 应用于使用 *timm* 库构建的骨干网络的输出端。具体调用如下：

```
导入timm2 网络 = timm.创建_模型（模型，特征_仅=真，预训练=真，输出_步幅=8）
```

这里，模型 是一个字符串，其内容为库中骨干网络的名称（例如 resnet18、convnext nano 等）。仅特征选项适用于大多数卷积架构，会将模型的输出转换为不同深度的激活图列表。由此，可以使用以下操作获取这些激活图的一些属性：

```
1 导入 numpy as np 2 导入 PyTorch 3 device = "cuda" if
torch.cuda.is_available() else "cpu" 4 net.to(device) 5 act_
maps =net(torch.zeros(1,3,224,224)) 6 z = sum([d.size()[1]for
d in act_maps]) 7 half_depth =int(np.round(len(act_maps)/2))
8 w = act_maps[half_depth].size()[2]9 h = act_maps[half_
depth].size()[3]
```

这些属性分别是每个激活图的特征数量之和 z，以及架构中大致位于中间位置的空间维度 (w,h)。请注意，只要骨干网络支持，这些属性就可以针对除 (224,224) 之外的任意输入图像尺寸计算。随后，这些属性将用于创建 RADAM 的主类：

```
1 导入 torchvision 2 从 einops.layers.torch 导入 Rearrange 3 类 RADAM(torch.nn.Module): 4 def __
init__(self, 设备, z, spatial_dims =(28,28), m=4, pos_encoding=True, dim_norm=(2,3)): →5
super(RADAM, self).__init__() 6 self.q=17 self.z=z 8 self.device = device 9 self.rearange =
torch.nn.Sequential(lp_norm_layer(p=2.0, dim=dim_norm, eps=1e-10), 10
torchvision.transforms.Resize(spatial_dims), 11 Rearrange('b c h w ->b c (h w)') 12 ) 13 self.RAEs = []
14 for model in range(m): 15 self.RAEs.append(RAE(q=self.q, z=z, w=spatial_dims[0] ,h=spatial_dims
[1] ,设备=device, seed=model*(self.q*z), pos_encoding=pos_encoding)) →
```

```
    定义 前向传播(自身 , x):
        输出 = []
        如果 不是实例(x , 列表):
            x = [x]
        设备 = 自身.设备
        批次 , _, _, _ = x[0].size()
        对于 深度 在 range(len(x)):
            x[深度] =自身。rearange(x[深度])
        对于 样本在 range(batch):
            聚合_激活_图 = PyTorch。vstack([x[i][样本] 对于 i 在 range(len(x))])
            池化后的_特征 =PyTorch.zeros(self.q,self.z).to(设备)
            遍历 随机自编码器 中self.RAE:
                池化后的_特征 +=随机自编码器.fit_AE(agg_激活_图)
            输出.append(池化后的_特征)
        返回 PyTorch.stack(out)

类lp_范数_层(PyTorch.nn.Module):
    def __初始化__(自身, p=2.0, 维度=(1, 2), eps=1e-10):
        父类().__初始化__()
        自身。段落=段落
        自身.维度=维度
        self.eps=eps
    定义 前向传播(自身 , x):
        返回 PyTorch.nn.functional.normalize(x, p=自身.p, dim=自身.dim, eps=自身.eps)
```

代码中的参数与我们在正文实验中使用的参数相同。此外，我们使用 *einops* 库来高效地重组激活图，并额外使用一个类，以 PyTorch 层的形式对批次进行 L_p 归一化。

随机自编码器（RAE）的实现如下。它接收模型的各项维度作为参数，这些维度此前已传入 RADAM 类。构造函数会生成随机编码器和位置编码，而拟合自编码器方法则计算解码器权重，即执行最小二乘训练：

```
类 RAE():
    def __初始化__(自身,q, z, w,  h, 设备, pos_encoding, 种子):
        自身._设备 = 设备
        自身。位置编码_编码=位置编码_编码
        自身。W = 生成_orthogonal(线性同余生成器(q, z, seed))。转至(设备)
        如果自身。位置编码_编码:
            自身。编码 = 二维位置编码(z, w, h)
            自身。编码 = PyTorch。重塑(自身。编码, (z, w*h)).to(设备)
        自身。_激活 = PyTorch。sigmoid

    def fit_AE(self, x):
        如果 self.位置编码: _编码:
            x = PyTorch.add(x,self.encoding)
        g = 自身。_激活(PyTorch。mm(自身。W,x))
        返回 PyTorch.linalg.lstsq(g.t(), x.t()).solution
```

其余用于权重初始化和位置编码的函数如下：

```
导入pickle
定义 线性同余生成器(n , n, 种子):
    升 = 米*n
    # To generate this weight file, use:
        #V = torch.zeros(2**18, dtype=torch.float)
        #V[0], a, b, c = 1, 75, 74, (2**16)+1
        #for x in range(1, (2**18)):
        #   V[x] = (a*V[x-1]+b) % c
        #with open('RAE_LCG_weights.pkl', 'wb') as f:
            #pickle.dump(V, f)
    使用 open('RAE_线性同余生成器_weights.pkl', 'rb') 作为 f:
```

```
12 V =pickle.load(f) 13 f.close() 14 V =V[种子:L+种子]15 V =torch.divide(torch.subtract(V, t
orch.mean(V)), torch.std(V)) 16 return V.reshape((m,n)) 17 18 def make_orthogonal(tensor): 19 rows =
tensor.size(0) 20 cols = tensor.numel() // rows 21 flattened = torch.reshape(tensor, (rows, cols)) 22 if
rows < cols: 23 flattened.t_() 24 q, r = torch.linalg.qr(flattened) 25 d =torch.diag(r, 0) 26 ph = d.sign()
27 q *= ph 28 if rows < cols: 29 q.t_() 30 return q 31 32 import math 33 def positionalencoding2d(d_
model, height, width): 34 pe = torch.zeros(d_model, height, width) 35 d_model = int(d_model / 2) 36 div_
term = torch.exp(torch.arange(0., d_model, 2) * 37 -(math.log(10000.0) / d_model)) 38 pos_w =
torch.arange(0., width).unsqueeze(1) 39 pos_h = torch.arange(0., height).unsqueeze(1) 40 pe[0:d_
model:2, :, :] =torch.sin(pos_w * div_term).transpose(0, 1).unsqueeze(1).repeat(1, height, 1) →41 pe[1:d_
model:2, :, :] =torch.cos(pos_w * div_term).transpose(0, 1).unsqueeze(1).repeat(1, height, 1) →42 pe[d_
model::2, :, :] = torch.sin(pos_h * div_term).transpose(0, 1).unsqueeze(2).repeat(1, 1, width) →43 pe[d_
model + 1::2, :, :] =torch.cos(pos_h * div_term).transpose(0, 1).unsqueeze(2).repeat(1, 1, width) →44
return pe
```

最后，对于给定的图像或图像批次，只需将主*RADAM* 类应用于*net*的输出，即可获得纹理表征：

```
纹理_表示 = RADAM(device,z,(w,h))(net(input_batch))
```

如正文中所述 论文中，这些表征随后用于训练线性支持向量机，以进行纹理分类。F
为此，我们使用 scikit-l 库来构建分类器，如下所示：

```
来自 sklearn 导入svm
支持向量机 = svm.SVC(kernel='linear')
```

2 使用RADAM进行补充实验

这里介绍正文中未呈现的补充实验。

2.1 随机初始化导致的方差

表 1 展示了改变线性同余生成器配置（a、 b和 c）对 RAE 权重初始化所导致的 RADAM 性能方差。我们从 [1] 中提供的表格里选取了 10 组配置。总体而言，我们观察到这些参数引起的性能方差极小，尤其是在使用更大的骨干网络时，这印证了随机权重的选择并不关键。我们采用了参数 $a = 75$、 $b = 74$和 $c = 2^{16} + 1$，从$x = 1$（表格第一行）开始，因为它使用的整数最小，而且根据 ZX81 的说法，这也是一种经典配置

[1] www.wikipedia.org/wiki/线性_同余_发生器 器

计算机可追溯至 1981 年。然而，细微差异可能会影响 RADAM 与其他方法之间的比较。尽管如此，值得注意的是，我们选择的线性同余生成器参数不会影响主论文中报告的 RADAM 最先进的结果。事实上，针对每个数据集和骨干网络仔细选择表 1 中的特定配置，可以取得更高的结果，但我们更倾向于在所有情况下采用单一且一致的配置。此外，为便于应用，我们提供了代码计算出的所有线性同余生成器配置的精确值文件（这些文件将在我们的代码仓库中提供）。

表1：改变用于为编码器生成随机权重的 LCG 配置时，RADAM 分类准确率的方差。对于表格的第二部分（ConvNeXt 结果），LCG 配置对应于与 ResNet18 相同的行。

LCG 参数				ResNet18			
来源	a	b	c	DTD	FMD	KTH	Outex13
ZX81	75	74	$2^{16}+1$	68.1	77.7	84.7	89.0
《Numerical Recipes》一书	1664525	1013904223	2^{32}	68.0	77.9	85.0	90.0
Borland C/C++	22695477	1	2^{32}	66.7	79.7	86.0	88.2
glibc（由 GCC 使用）	1103515245	12345	2^{31}	68.0	78.0	84.7	90.0
Borland Delphi、Virtual Pascal	134775813	1	2^{32}	67.9	79.2	85.6	90.1
Microsoft Visual/Quick C/C++	214013	2531011	2^{32}	68.1	78.2	85.1	89.3
Microsoft Visual Basic	1140671485	12820163	2^{24}	68.1	79.2	85.4	88.7
Native API 中的 RtlUniform	2147483629	2147483587	2^{31}-1	68.0	78.9	85.1	89.6
Java 的 java.util.Random	25214903917	11	2^{48}	67.9	77.4	85.4	89.7
cc65	16843009	826366247	2^{32}	68.0	78.7	85.6	89.0
			平均值	67.9±0.4	78.5±0.8	85.3±0.4	89.4±0.6

	ConvNeXt-nano				ImageNet-21K 中的 ConvNeXt-XL			
	DTD	FMD	KTH	Outex13	DTD	FMD	KTH	Outex13
	74.9	87.1	89.6	92.5	83.7	95.2	94.4	92.5
	75.0	86.9	89.2	92.4	83.9	95.7	94.5	91.8
	75.1	86.8	89.5	91.2	83.7	95.3	94.6	92.4
	75.0	86.6	89.8	92.2	83.9	95.4	94.1	92.6
	75.1	86.4	89.5	92.4	83.8	95.6	94.5	92.4
	75.1	87.1	89.5	92.5	83.8	95.5	94.6	92.8
	74.9	87.7	89.4	92.2	83.7	95.3	94.6	92.4
	75.1	86.9	89.6	91.9	83.8	95.2	94.6	91.9
	74.9	86.9	89.3	92.6	83.8	95.5	94.8	92.2
	75.0	87.6	89.0	92.2	83.8	95.5	94.5	92.8
平均值	75.0±0.1	87.0±0.4	89.4±0.2	92.2±0.4	83.8±0.1	95.4±0.2	94.5±0.2	92.4±0.3