Deep Forest: Towards an Alternative to Deep Neural Networks

Zhi-Hua Zhou and Ji Feng

摘要

本文提出了一种能在许多任务中能与深度神经网络相竞争的决策树集成方法：gcForest。相比于需要大量精力进行超参数调整的深度神经网络，gcForest 更容易训练；在实验中，相同的超参数应用在不同领域的不同数据上，都获得了优异的效果。同时，gcForest 的训练效率较高，用户可以根据可用的计算资源来控制训练成本，加上 gcForest 可并行的特点，训练效率可能会进一步提高。此外，与需要大规模训练数据的深度神经网络相比，gcForest 在小规模训练数据上也能很好地工作。

关键词：深度森林；深度神经网络

1 引言

近年来，深度神经网络在各种应用领域都取得了巨大的成功，特别是在视觉和语音信息领域上[1][2]，导致了深度学习的热潮。

虽然深度神经网络很强大，但它们也有明显的缺陷。首先，深度神经网络的训练通常需要大量的训练数据，这使得它无法直接应用于数据规模较小的任务中。即使在大数据时代，许多真实任务由于标记成本高而仍然缺乏足够的标记数据，这导致深度神经网络在这些任务中的性能较差。其次，深度神经网络是一种非常复杂的模型，在训练过程中通常需要强大的计算设施，从而使得大公司以外的个人无法充分地使用它。同时，深度神经网络有太多超参数，而只有仔细调整超参数才能获得较好的学习效果。例如，即使一些作者都使用卷积神经网络[3][4]，由于卷积层结构等不同的选择，他们实际上使用不同的学习模型，实际上使用不同的学习模型。这一事实不仅使深度神经网络的训练非常棘手，像艺术而不是科学，而且深度神经网络的理论分析非常困难，因为太多的干扰因子几乎无限的构型组合。

表征学习能力对深度神经网络至关重要。深度神经网络为了利用大量的训练数据，其表征学习能力应该很强，这解释了为什么深度神经网络非常复杂，比支持向量机等普通的学习模型要复杂得多。因此，如果我们能将这种强的表征学习能力应用于其他学习模型，也许我们就能够实现与深度神经网络竞争的性能，且减少前面提过的神经网络的缺陷。

在本文中，我们提出了一种新的决策树（gcForest：多粒度扫描级联森林）集成方法。该方法生成一个深度集成森林，具有级联结构，使 gcForest 能够进行表示学习。当输入具有高纬度时，多种粒度扫描可以进一步增强其表征学习能力，使得 gcForest 可能具有上下文或结构感知能力，可以自适应地确定级联级别的数量，从而自动设置模型的复杂性，使得 gcForest 可以在小规模数据上出色地执行，因此用户可以根据可用的计算资源来控制培训成本。gcForest 的超参数比深度神经网络要少得多，而且它对超参数不敏感，因此对于不同领域的不同数据，它在大多数情况下都能够在默认设置下获得优异的性能。这些特性不仅使 gcForest 的训练更方便，而且使理论分析也比深度神经网络更容易（树形

学习器通常比神经网络更利于解释）。在我们的实验中，gcForest 具有对深度神经网络的高度竞争力，而 gcForest 的训练时间成本小于深度神经网络。

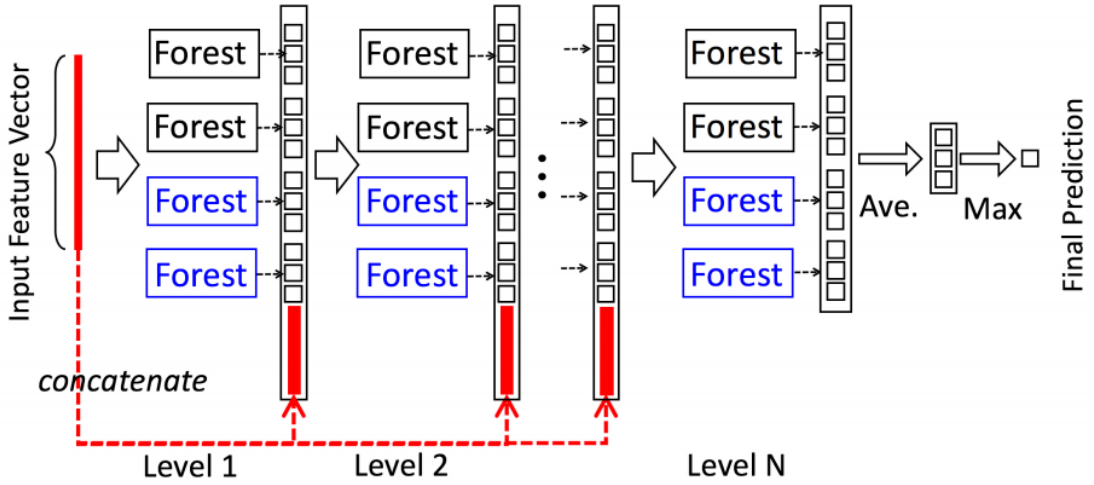

图 1: 梯联森林结构图。假设级联的每个级别由两个随机的森林（黑色）和两个完全随机的树森林（蓝色）组成。假设有三个类要预测；因此，每个森林将输出一个三维的类向量，然后将其连接起来以重新表示原始输入。

2 相关工作

为了解决复杂的学习任务，学习模型的层数很可能必须够深。然而，目前的深度模型总是一种神经网络，即多层参数化的、可以通过反向传播进行训练的可微非线性模块。思考深度学习是否可以与其他模块一起实现是一件很有趣的事情，因为它们有自己的优势，如果深度足够，可能会显示出巨大的潜力。本文致力于解决这一问题，并说明如何构建深度森林，这可能为许多任务打开一扇替代深度神经网络的大门。

3 本文方法

3.1 本文方法概述

在本节中，我们将首先介绍级联森林结构、多粒度扫描、总体架构和对超参数的注释。

3.2 级联森林结构

深度神经网络中的表示学习主要依赖于对原始特征的逐层处理。受这种识别的启发，gcForest 采用了一个级联结构，如图 1所示，每个级联级别接收其上一级处理的特征信息，并将其处理结果输出到下一级。

每一层都是决策树森林的集合，即一个集合的集合。在这里，我们包括了不同类型的森林来鼓励多样性，因为多样性对集合建设至关重要[5]。简单起见，假设我们使用两个完全随机森林和两个随机森林[6]，每个完全随机森林包含 500 棵完全随机的树[7]，在树的每个节点上随机选择一个特征进行分割，直到每个叶节点只包含相同的实例类。类似地，每个随机森林包含 500 棵树，通过随机选择 $\sqrt{d}$ 数量的特征作为候选特征（d 是输入特征的数量），并选择具有最佳基尼指数的特征作为分割特征。每个森林中的树数是一个超参数。

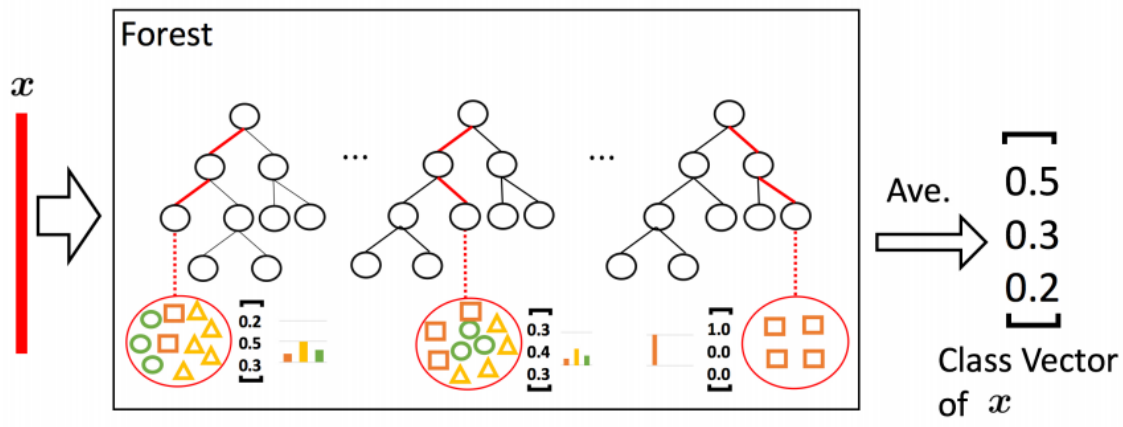

图 2: 梯联森林结构图。假设级联的每个级别由两个随机的森林（黑色）和两个完全随机的 1 类向量生成的说明。叶节点中的不同标记意味着不同的类。

给定一个实例，每个森林将产生一个类分布的估计，计算实例所落在的叶节点上不同类的百分比，然后在同一森林中的所有的树木中求平均值，如图 2所示，其中，红色路径为实例遍历到叶节点的路径。

估计的类分布形成一个类向量，然后将其与原始特征向量连接起来，输入到下一级级联。例如，假设有三个类，那么四个森林的每个森林将产生一个三维类向量；因此，下一级级联将接收 12（= 3×4）个增强特征。

为了减少过拟合的风险，通过 k 倍交叉验证生成每个森林产生的类向量。具体来说，每个实例将被用作 k−1 次的训练数据，得到 k−1 类向量，然后平均产生最终的类向量作为下一级级联的增强特征。在扩展一个新的级层后，将在验证集上估计整个级层的性能，如果没有显著的性能增益，训练过程将终止，从而自动确定级联级层的数量。与大多数复杂度固定的深度神经网络模型相比，gcForest 通过在适当时终止训练的方式来自适应地决定其模型复杂度，来使得它能够适用于不同规模的训练数据，而不限于大规模的训练数据。

3.3 多粒度扫描

深度神经网络在处理特征关系方面非常强大，例如，卷积神经网络对以原始像素之间的空间关系为关键的图像数据有效[8]；递归神经网络对以序列关系为关键的序列数据有效[9][10]。受这种认知的启发，我们通过多粒度扫描的过程来增强级联森林。

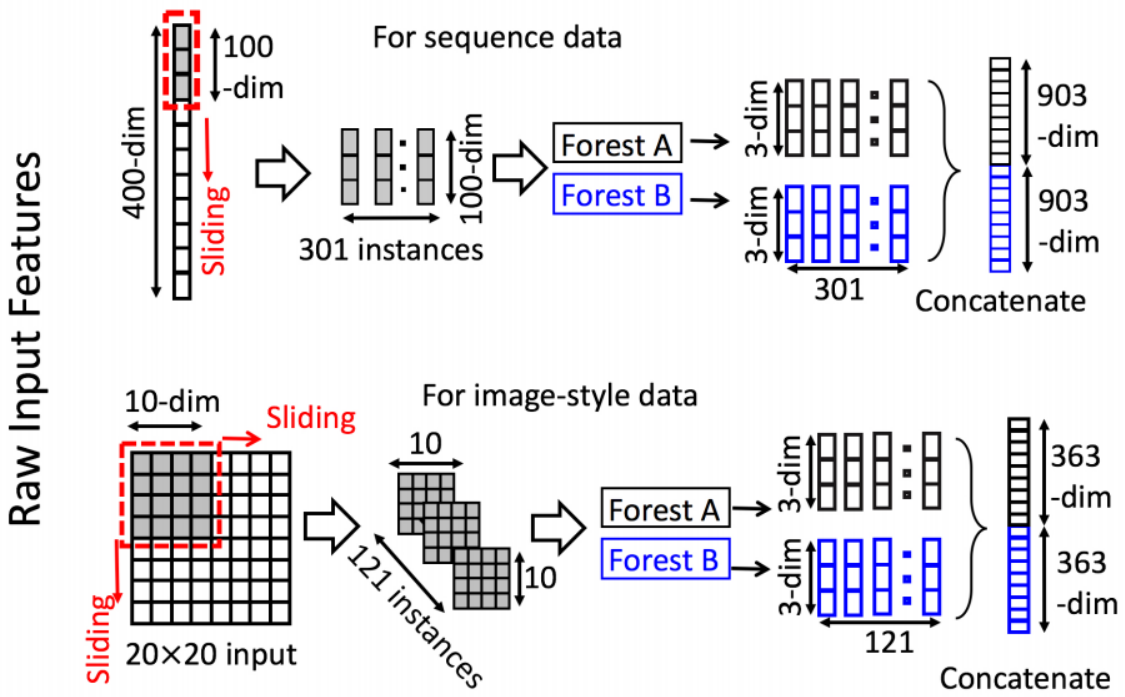

图 3: 通过滑动窗口扫描嵌入特征的例子。假设有三个类，原始特征为 400 维，滑动窗口为 100 维。

如图 3所示，滑动窗口用于扫描原始特征。假设有 400 个原始特征，并使用了 100 个特征的窗口大小。对于序列数据，通过滑动一个特征的窗口，将生成一个 100 维的特征向量；总共产生了 301 个特征向量。如果原始特征具有空间关系，例如一个包含 400 个图像像素的 20×20 面板，那么一个 10×10 窗口将产生 121 个特征向量。

所有从积极/消极的训练例子提取的特征向量被视为积极/消极的实例，然后将用于生成类向量，就像在 3.2 节中：相同大小的窗口提取的实例将被用于训练一个完全随机森林和随机森林，然后生成类向量并连接为转换后的特征。如图 3所示，假设有 3 个类，使用一个 100 维窗口；然后每个森林产生 301 个三维类向量，得到一个对应于原始 400 维特征向量的 1806 维变换后特征向量。当转换后的特征向量太长而无法容纳时，可以执行特征采样，例如，通过对滑动窗口扫描生成的实例进行子采样，因为完全随机森林不依赖于特征分裂选择，而随机森林对不准确的分裂特征选择相当不敏感。

图 3只显示了一个滑动窗口的大小。通过使用多个大小的滑动窗口，将生成不同粒度的特征向量，如图 4所示。

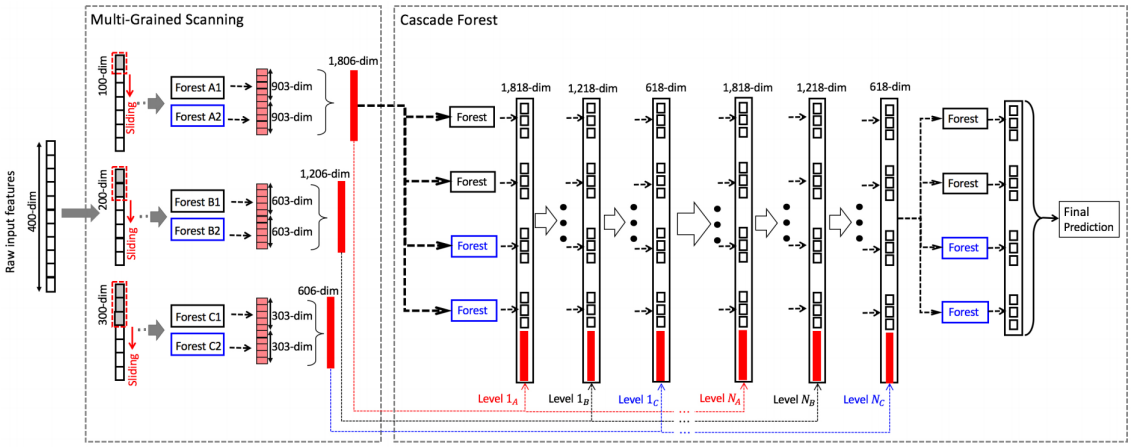

图 4: gcForest 的总体过程。假设有三个类可以预测，原始特征是 400 维的，并使用三种大小的滑动窗口。

3.4 总体程序和超参数

图 4总结了 gcForest 的总体过程。假设原始输入为 400 个原始特征，并且有三个窗口大小用于多粒度扫描。对于 m 个训练实例，一个大小为 100 个特征的窗口将生成 301×m 个 100 维训练示例的数据集。这些数据将用于训练一个完全随机森林和一个随机森林，每个森林都包含 500 棵树。如果有三类需要预测，则将得到一个 1806 维的特征向量，如第 3.2 节所述。转换后的训练集将用于第一级级联森林的训练。类似地，大小为 200 和 300 个特征的滑动窗口将为每个原始训练示例分别生成 1206 维和 606 维的特征向量。变换后的特征向量，加上上级生成的类向量，分别用于对二级和三级森林进行训练。这个过程将重复，直到验证性能收敛。换句话说，最终的模型实际上是一个级联森林的级联，其中级联中的每个级联都由多个级联（级联森林）组成，每个内层级联对应于一个扫描粒度，如图 4所示。其中，对于困难的任务，如果计算资源允许，用户可以尝试更多的扫描粒度。

给定一个测试实例，它将通过多粒度扫描过程得到相应的转换特征表示，然后通过级联直到最后一级。最终的预测将通过在最后一级对四个三维类向量进行聚合并取最大聚合值的类的方式来得到。

Table 1: Summary of hyper-parameters and default settings. Boldfont highlights hyper-parameters with relatively larger influence; "?" indicates default value unknown, or generally requiring different settings for different tasks.

Deep neural networks (e.g., convolutional neural networks)	gcForest
Type of activation functions: Sigmoid, ReLU, tanh, linear, etc. Architecture configurations: **No. Hidden layers**: ? **No. Nodes in hidden layer**: ? **No. Feature maps**: ? **Kernel size**: ? Optimization configurations: **Learning rate**: ? Dropout: {0.25/0.50} **Momentum**: ? **L1/L2 weight regularization penalty**: ? Weight initialization: Uniform, glorot_normal, glorot_uni, etc. Batch size: {32/64/128}	Type of forests: Completely-random tree forest, random forest, etc. Forest in multi-grained scanning: **No. Forests**: {2} **No. Trees in each forest**: {500} Tree growth: till pure leaf, or reach depth 100 **Sliding window size**: $\{\lfloor d/16 \rfloor, \lfloor d/8 \rfloor, \lfloor d/4 \rfloor\}$ Forest in cascade: **No. Forests**: {8} **No. Trees in each forest**: {500} Tree growth: till pure leaf

图 5: 总结了深度神经网络和 gcForest 的超参数，其中给出了我们实验中使用的默认值

4 复现细节

4.1 与已有开源代码对比

单层级联的架构参考代码：https://github.com/onism/gcForest，而数据的处理，表格数据、序列数据、图像数据的多粒度扫描和多层级联森林部分的代码没有参考代码，纯属自己编写。

本次论文复现按照论文里的模型，对表格数据、序列数据、图像数据都进行了模型重现和数据实验。

4.2 复现实验与结果

4.2.1 表格数据

数据集用的是 sklearn 自带的鸢尾花数据集，算法模型只需用到后面的级联部分，不需要用到前面的多粒度扫描。跑出来测试集准确率为 0.94。

4.2.2 序列数据

数据集描述：用的是论文中的 gtzan，该数据集是音乐流派分类相关的，它包含十个文件夹，对应十种音乐流派；每个文件夹里面有 100 个.wav 后缀的音频文件，时长都是 30s。

模型描述：见图 6：

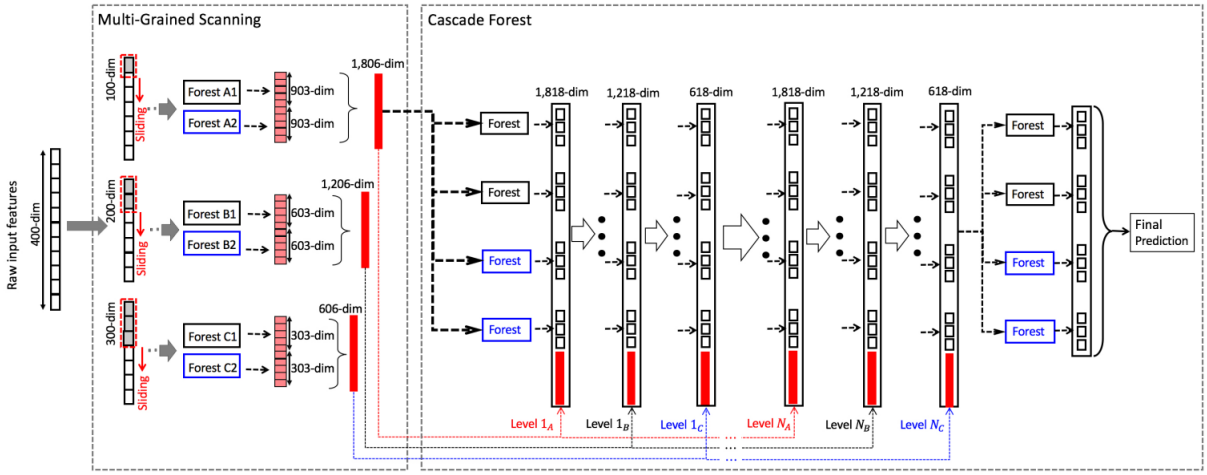

图 6: 序列数据所用模型

数据处理：我用了 Librosa 包的方法提取了.wav 音频文件的一些关键特征，具体特征和代码见图 7：

```
# 色度频率   12 ,1292
chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
shape = np.shape(chroma_stft)
for i in range(shape[0]):
    label_features_line = np.concatenate([label_features_line, chroma_stft[i, :].reshape(1, -1)], axis=1)

# 光谱质心
spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
label_features_line = np.concatenate([label_features_line, spec_cent], axis=1)

spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
label_features_line = np.concatenate([label_features_line, spec_bw], axis=1)

# 光谱衰减
rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
label_features_line = np.concatenate([label_features_line, rolloff], axis=1)

# 过零率
zcr = librosa.feature.zero_crossing_rate(y)
label_features_line = np.concatenate([label_features_line, zcr], axis=1)

# 梅尔频率倒谱系数  20.1292
mfcc = librosa.feature.mfcc(y=y, sr=sr)
for j in range(np.shape(mfcc)[0]):
    label_features_line = np.concatenate([label_features_line, mfcc[j, :].reshape(1, -1)], axis=1)
```

图 7: 音频文件特征提取代码段

发现复现结果和论文中作者给出的实验结果还有一定差距，分析原因可能是：1. 实验条件限制（我是用实验室的电脑跑的，还没有服务器资源供我使用），所以我的采样率设置的非常低（过高会导致我电脑内存溢出），比默认采样率小了一百倍左右。2. 音频数据的采样本身导致采样后的数据形式是特征数远远大于样本数的表格数据，也就是形成了宽矩阵。

解决方法：

解决问题 2，进行数据增强，为了进一步探索数据增强对于实验结果提升的影响，进行了两次数据增强，发现数据增强确实能提升准确率。增强代码见图 8：

```
#噪声增强，0.004是噪声因子
y_noise=y+0.004*np.random.normal(loc=0,scale=1,size=len(y))
y_noise2 = y + 0.002 * np.random.normal(loc=0, scale=1, size=len(y))
#波形移位
y_roll=np.roll(y,int(sr//2))
y_roll2 = np.roll(y, int(sr // 4))
#音高修正
y_high=librosa.effects.pitch_shift(y,sr,n_steps=3,bins_per_octave=24)
y_high2 = librosa.effects.pitch_shift(y, sr, n_steps=-3, bins_per_octave=24)
```

图 8: 数据增强代码段

实验结果：

	增强前	增强一次	增强两次
1	56.82%	61.65%	61.42%
2	48.11%	60.51%	64.45%
3	48.48%	62.31%	65.15%

经过两次增强，复现结果基本和论文中作者给出的实验结果相近了。

4.2.3 图像数据

数据集描述：由于实验条件限制，我找了较小的图像数据集，用的是 sklearn 自带的手写数字，数据规模为（1797，8，8）。

模型描述：将图 6的左半部分换成图 9，右半部分保持不变。也就是改变多粒度扫描部分的模型，后面级联的模型保持不动：

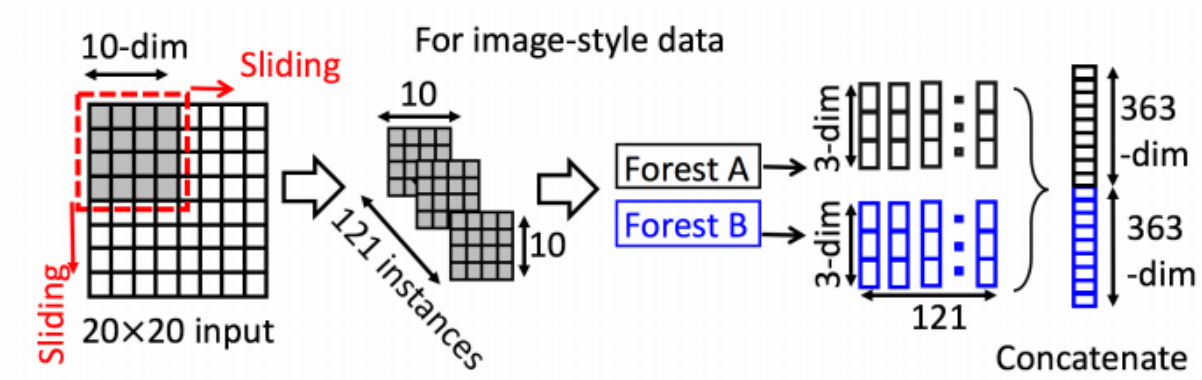

图 9: 图像数据所用的多粒度扫描模型

实验结果：重复了三次分别为：93.60，83.67，91.25。

参数设置：由于这个数据集比较小，所以可以采用论文超参数设置：扫描的森林：2 个森林为一组 500 颗树/森林树停止生长条件：最大深度达到 100。级联的森林：8 个森林为一组 500 颗树/森林树停止生长条件：无（完全不剪枝）。

4.3 创新点

图 10中绿色框线部分：论文里的方法是对所有窗口滑动得到的子特征集进行纵向拼接得到一个很高的矩阵，相当于扩充了样本量，再用拼接后的数据集训练森林。我改进的方法是对所有窗口滑动得到的子特征集都单独训练森林。

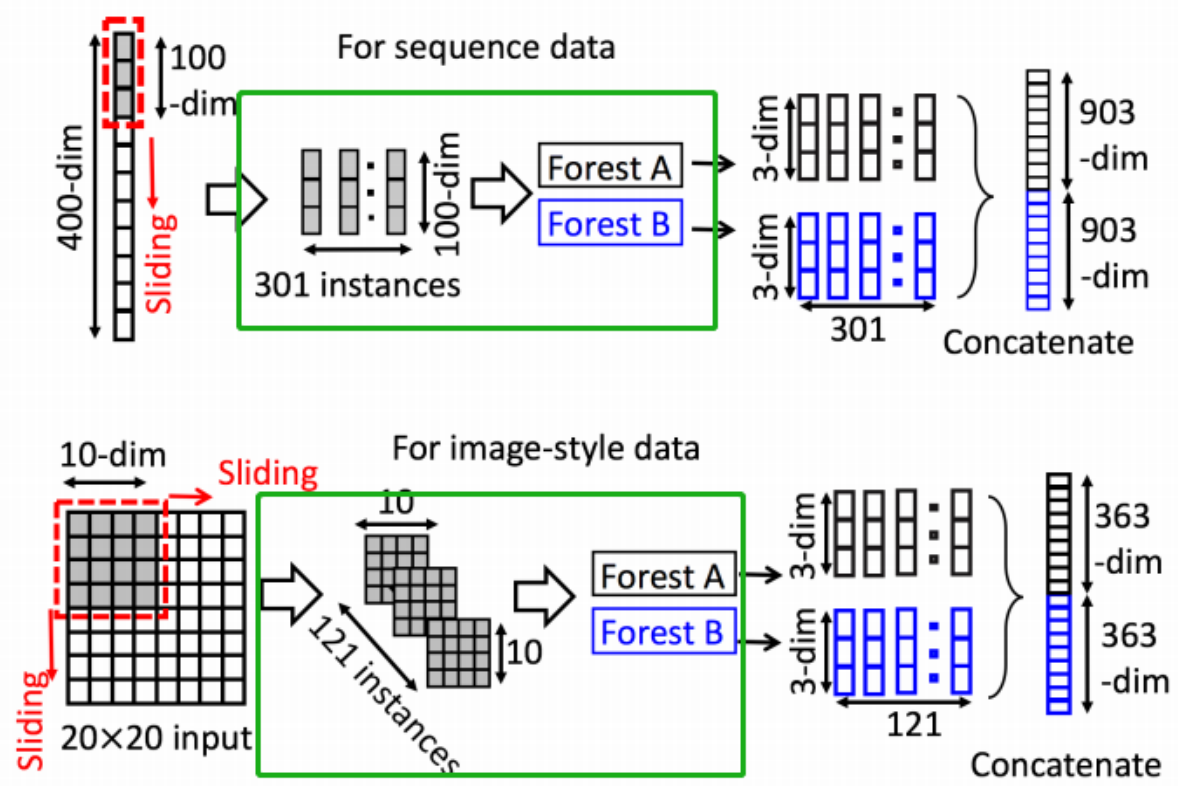

图 10: 创新部分

5 实验结果分析

创新部分是针对多粒度扫描的，所以只有序列数据和图像数据可以看到创新前后的性能提升。

5.1 序列数据

	增强前（原版）	增强前（改进版）	增强一次	增强一次（改进版）	增强两次	增强两次（改进版）
1	56.82%	53.79%	61.65%	62.97%	61.42%	62.66%
2	48.11%	57.95%	60.51%	63.07%	64.45%	65.42%
3	48.48%	56.44%	62.31%	64.58%	65.15%	65.91%

可见，其他条件相同的情况下，对于音频数据，改进后的方法的平均准确率更高。

分析：改进版在性能上是以空间换时间的：1. 原版比改进版省内存空间：改进版在扫描阶段，窗口每滑动一次都要训练出一个森林并且保存起来，所以森林非常多，内存消耗大。原版是滑动到最后，才训练森林，所以要保存的森林非常少。2. 改进的比原版的训练速度更快：原版是每滑动一次，都增加一行样本，滑动到最后才训练森林，所以用来训练森林的样本量非常大，因此耗时长。

5.2 图像数据

	原版	改进版
1	93.60%	95.79%
2	83.67%	95.62%
3	91.25%	95.45%

可见，超参数相同的情况下，对于该图像数据集，改进后的方法的平均准确率更高。

6 总结与展望

本次论文复现按照论文里的模型，对表格数据、序列数据、图像数据都进行了模型重现和数据实验。对多粒度扫描部分进行了改进，但是仅限于序列数据和图像数据才适用，不能对表格数据也进行改进，希望能探索出对表格数据的改进方案，也许后续可以尝试对表格数据进行编码。

参考文献

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[J/OL]. Commun. ACM, 2017, 60(6): 84-90. https://doi.org/10.1145/3065386. DOI: 10.1145/3065386.

[2] HINTON G E, DENG L, YU D, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups[J]. IEEE Signal Processing Magazine, 2012, 29: 82-97.

[3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[J/OL]. Commun. ACM, 2017, 60(6): 84-90. https://doi.org/10.1145/3065386. DOI: 10.1145/3065386.

[4] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition [J]. Proc. IEEE, 1998, 86: 2278-2324.

[5] VAN D. Ensemble Methods : Foundations and Algorithms[C]//. 2012.

[6] BREIMAN L. Random Forests[J]. Machine Learning, 2001, 45: 5-32.

[7] LIU F T, TING K M, YU Y, et al. Spectrum of Variable-Random Trees[J]. J. Artif. Int. Res., 2008, 32(1): 355-384.

[8] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[J/OL]. Commun. ACM, 2017, 60(6): 84-90. https://doi.org/10.1145/3065386. DOI: 10.1145/3065386.

[9] GRAVES A, MOHAMED A R, HINTON G. Speech recognition with deep recurrent neural networks[C]//2013 IEEE International Conference on Acoustics, Speech and Signal Processing. 2013: 6645-6649. DOI: 10.1109/ICASSP.2013.6638947.

[10] CHO K, van MERRIËNBOER B, GULCEHRE C, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation[C/OL]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1724-1734. https://aclanthology.org/D14-1179. DOI: 10.3115/v1/D14-1179.