

图像分类比赛中，你可以用如下方案举一反三



众所周知，Kaggle 是一个进行预测建模及数据分析的竞赛平台。在这个平台上，统计学家和数据科学家竞相构建最佳的模型，这些模型被用于预测、描述公司和用户上传的数据集。这种众包的方式之所以被广为接受，是因为对于同一个预测建模任务来说，可能存在无数种解决策略，但是想要事先知道哪种技术或分析方法是最有效的几乎不可能。[1]

任务概述

你能分清杂草和农作物幼苗吗？

如果我们能高效地区分农作物幼苗和杂草，那么就可以提升农作物的产量，更好地管理环境中的杂草。

Aarhus 大学信号处理研究小组与南丹麦大学合作，发布了一个用于该任务的数据包，其中包括处于不同生长阶段的 12 个物种（共计 960 种植物）的图像。[1][2]

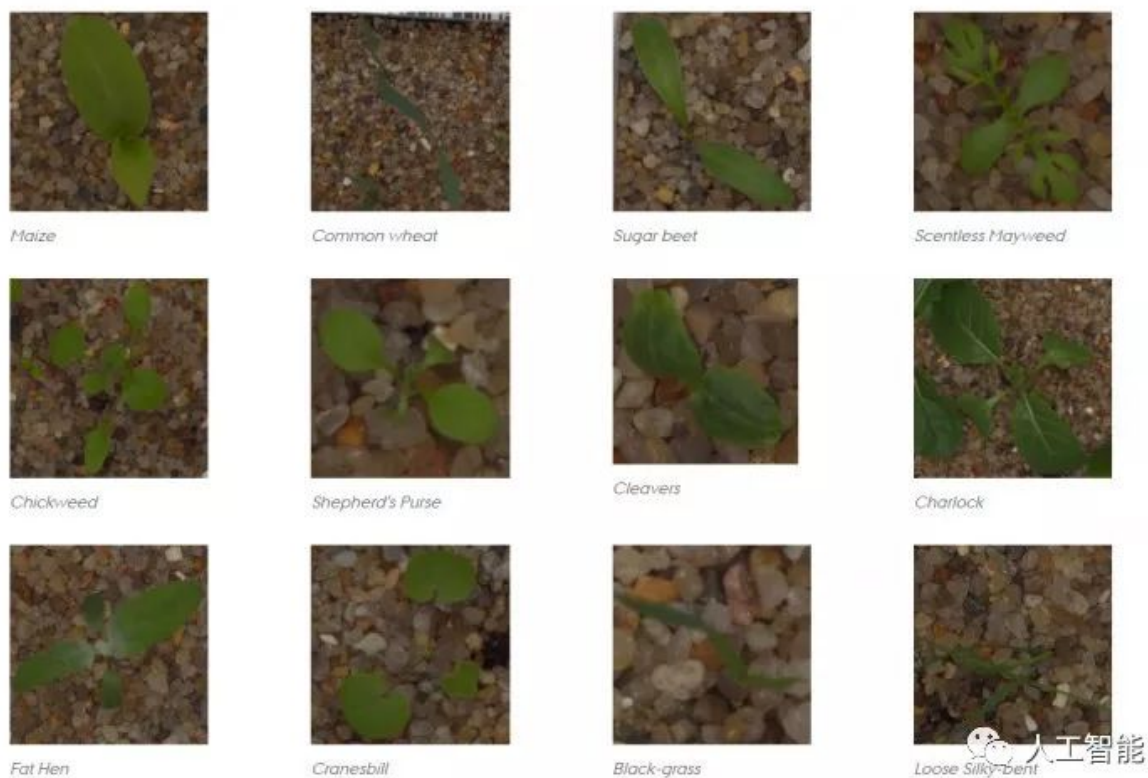


植物样本之一：「繁缕」样本

这个公开的数据库由注释的 RGB 图像组成，其物理分辨率大约为每毫米 10 像素。

为了对使用该数据库得到的分类结果进行标准化评估，组织者提供了基于 F1 值的对比基准，你可以通过如下链接获得这个数据集：<https://vision.eng.au.dk/plant-seedlings-dataset/>。[13]

下图是一个表示了该数据集中 12 类植物的样本：



图片来源: <https://vision.eng.au.dk/plant-seedlings-dataset/>

下面为大家介绍这一图像分类任务，该任务可以分为五个步骤

第一步：

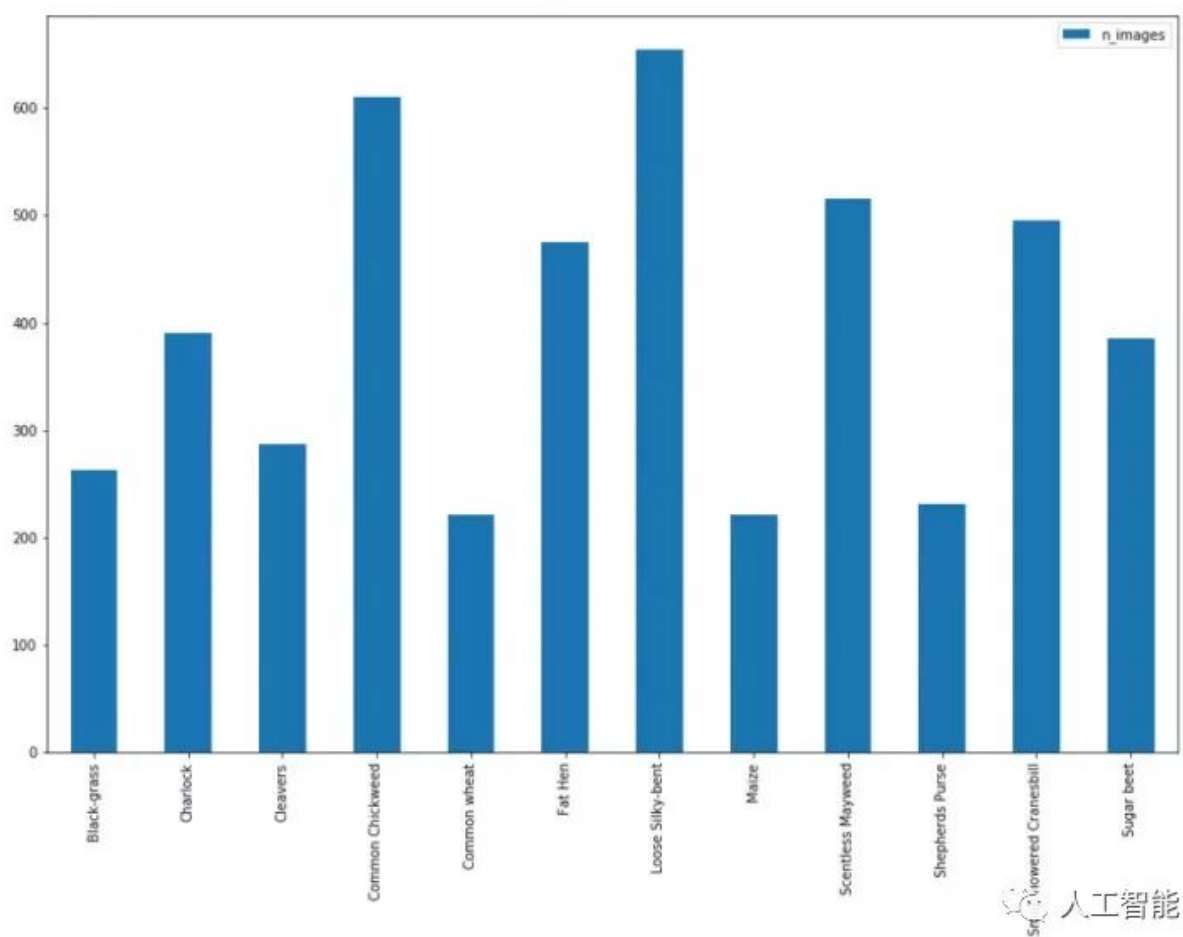
在大多数机器学习任务中，我们首先要做的（也是最重要的任务）就是在使用算法之前分析数据集。这一步骤之所以重要，是因为它能够让我们对数据集的复杂度有深入的了解，这最终将有助于算法的设计。

图像和类别的分布情况如下：

```
Black-grass 263 images
Charlock 390 images
Cleavers 287 images
Common Chickweed 611 images
Common wheat 221 images
Fat Hen 475 images
Loose Silky-bent 654 images
Maize 221 images
Scentless Mayweed 516 images
Shepherds Purse 231 images
Small-flowered Cranesbill 496 images
Sugar beet 385 images
```



正如文中所提到的，该数据集共包含 4750 张从属于 12 个类别的植物图片。然而，如上图所示，这种分布是不均匀的，各种类别的植物分布从最多 654 张图像到最少 221 张图像。很显然数据是不平衡的，我们需要对数据进行平衡处理，以便获得最佳的分类效果。本文将在第三步中讨论这个问题。



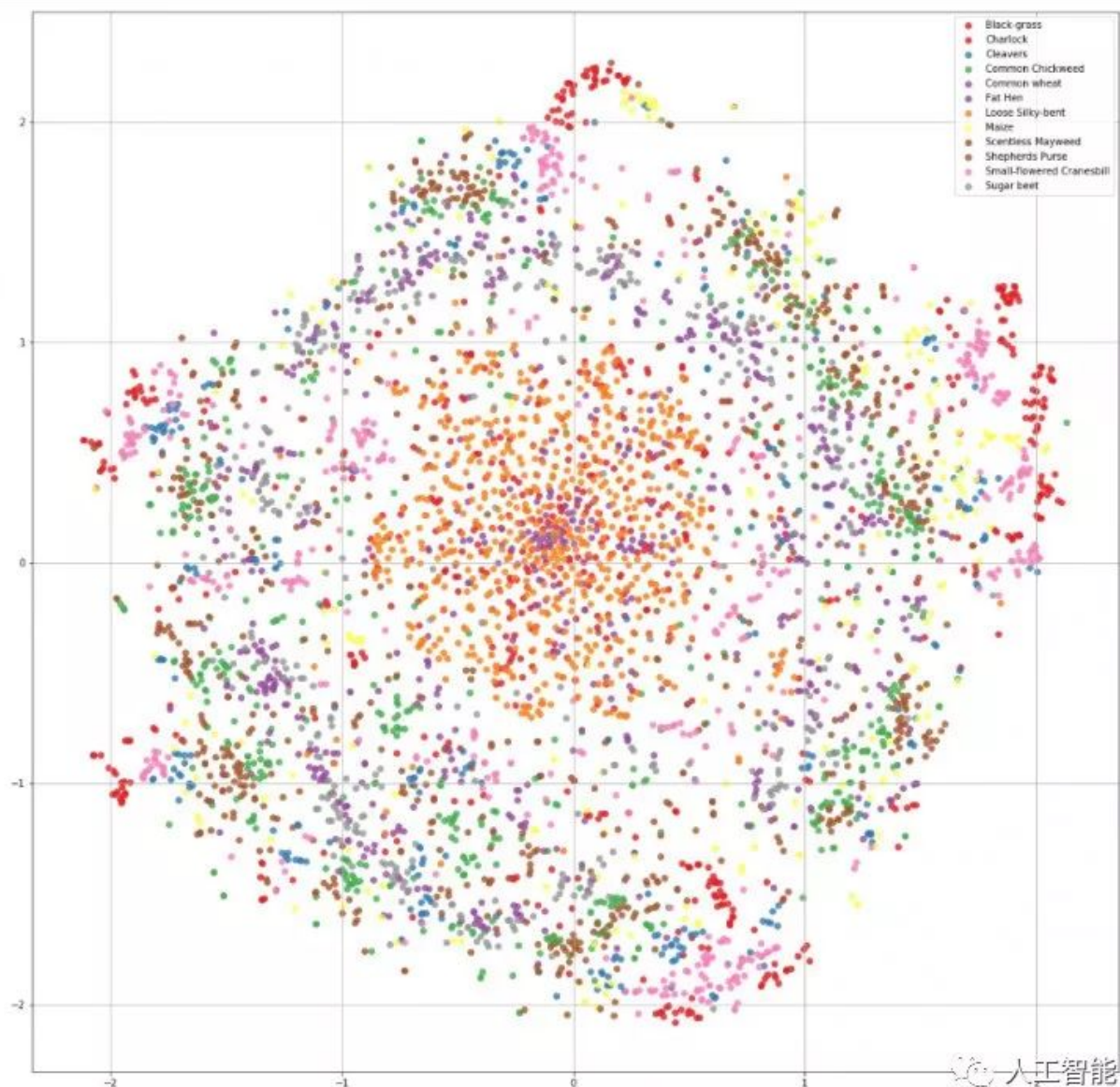
每个类的图像分布

为了更好地理解数据，对图像进行可视化处理十分重要。因此，我们将每类植物的示例图片展示了出来，以便看到图像之间的差异。



上面这些图片看上去实在太像了，以至于我们不能直接看出什么信息。因此，我决定使用 t 分布随机邻域嵌入（<https://lvdmaaten.github.io/tsne/>）可视化技术来查看图片的分布。

t 分布随机邻域嵌入（t-SNE）是一种特别适合对高维数据集进行可视化的降维技术。这种技术可以通过「Barnes-Hut」近似算法来实现，这使得它能够被应用于大型的真实数据集。
[14]



数据集的t-SNE可视化结果

仔细观察之后，我们几乎看不出类别之间的区别。因此，知道仅仅是人类难以区分这个数据，还是机器学习模型也很难区分这个数据很重要。所以，我们将为此做一个基本的对比基准。

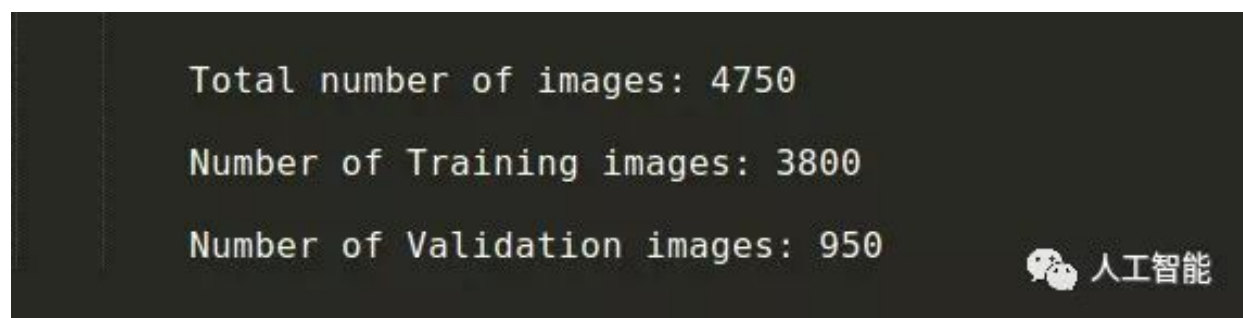
训练集和验证集

在开始建立模型的对比基准前，我们需要将数据划分为训练数据集和验证数据集。在原始测试集上测试模型之前，验证集起到了测试数据集的作用。所以，一个模型基本上是在训练数据集上进行训练，在验证集上进行测试，随着时间的推移，模型在验证集上的性能将会提升。

一旦我们对验证集上的测试结果感到满意，我们就可以在真实的测试集上应用该模型。通过这种方式，可以看出模型是否在验证集上发生欠拟合或过拟合现象，这可以帮助我们更好地拟合模型。

我们将包含 4750 张图片的数据集的 80% 作为训练集，另外 20% 作为验证集。

```
Total number of images: 4750
Number of Training images: 3800
Number of Validation images: 950
```



训练集和验证集的划分

第二步：

当获得了训练集和验证集之后，我们将开始使用数据集的对比基准。正如所见，这是一个分类问题：给定测试集，我们需要将图片归类到 12 个类别中的某一类。我们将使用卷积神经网络（CNN）来完成这项任务。

「如果你是一位初学者，需要对深度学习术语有更深入的了解，请访问如下博客：

<https://medium.com/@shridhar743/a-beginners-guide-to-deep-learning-5ee814cf7706>」

事实上，有许多方法可以创建卷积神经网络（CNN）模型，我们将使用 Keras 深度学习程序库来实现第一个对比基准。我们还将使用 Keras 中提供的预训练好的模型，这些模型已经利用 ImageNet 数据集训练过，我们将对其进行调优以满足任务需求。

从头开始训练一个卷积神经网络（CNN）的效率相当低下，我们将利用在包含 1000 类图像的 ImageNet 上预训练好的卷积神经网络（CNN）的权重，然后将某些层保持为「冻结」状态，再将一些层解冻并进行训练，从而进行调优。这是因为，最上面的层学习到简单的基本特征，而我们不需要对其进行训练，可以直接将它们应用到我们的任务中。需要注意的一点是，我们要检查数据集是否与 ImageNet 类似，以及我们的数据集规模有多大。这两

个特征将决定我们如何进行调优。如果你想了解更多的细节，请参阅 Andrej Karpathy 的博客 (<https://medium.com/@karpathy>)。

在植物幼苗检测比赛的环境下，数据集规模很小，但是与 ImageNet 有些相似之处。因此，我们可以首先直接使用 ImageNet 的权重，仅仅在对比基准的基础上添加一个能够对 12 个类进行分类的最终输出层。接着，我们将逐渐解冻一些模型底部的层，并仅仅对这些解冻的层进行训练。

由于 Keras 库提供了大量预训练好的模型，我们采用 Keras 为对比基准进行初始化。具体而言，我们将使用 ResNet50 和 InceptionResNetV2 这两个模型。很重要的是，我们需要用一个简单模型和一个非常高端的模型对数据集进行基准测试，以便发现给定模型是否产生了欠拟合和过拟合。

Available models

Models for image classification with weights trained on ImageNet:

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- DenseNet
- NASNet
- MobileNetV2



Keras 库提供的在 ImageNet 上预训练好的模型

图片来源: <https://keras.io/applications/>

此外，我们还可以检测这些模型在 ImageNet 数据集上的性能，查看 Keras 中每个模型 (<https://keras.io/applications/>) 的参数个数，从而选择对比基准模型。

Documentation for individual models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

201 人工智能

图片来源: <https://keras.io/applications/>

在第一个基准测试中,我删除了最后的输出层,并添加了一个能够对 12 个类别进行分类的最终输出层。此外,我将总结出的模型运行结果和参数的个数打印了出来,下图是最后几层网络的信息截图:

Layer (type)	Output Shape	Param #	Connected to
conv_7b_ac (Activation)	(None, None, None, 1)	0	conv_7b_bn[0][0]
avg_pool (GlobalAveragePooling2D)	(None, 1536)	0	conv_7b_ac[0][0]
dense_1 (Dense)	(None, 12)	184440	avg_pool[0][0]
Total params: 54,521,176			
Trainable params: 184,440			
Non-trainable params: 54,336,736			

201 人工智能

我在模型的最后添加了一个全连接层,构建了第一个对比基准


我共将该模型运行了 10 轮,而实验结果在第 6 轮之后就饱和了。训练的准确率为 88%,验证的准确率则为87%。

Epoch 8/10	149/149	[=====]	- 269s 1s/step - loss: 0.6423 - acc: 0.8663 - val_loss: 0.4201 - val_acc: 0.8802
Epoch 9/10	149/149	[=====]	- 264s 1s/step - loss: 0.5307 - acc: 0.8787 - val_loss: 0.4050 - val_acc: 0.8751
Epoch 10/10	149/149	[=====]	- 261s 1s/step - loss: 0.4487 - acc: 0.8826 - val_loss: 0.3976 - val_acc: 0.8700

201 人工智能

为了进一步提升模型的性能，我们解冻了一些模型底部的层，并且令其学习率呈指数形式递减，像这样，我们训练了更多的层。这个操作使模型性能提高了 2%。

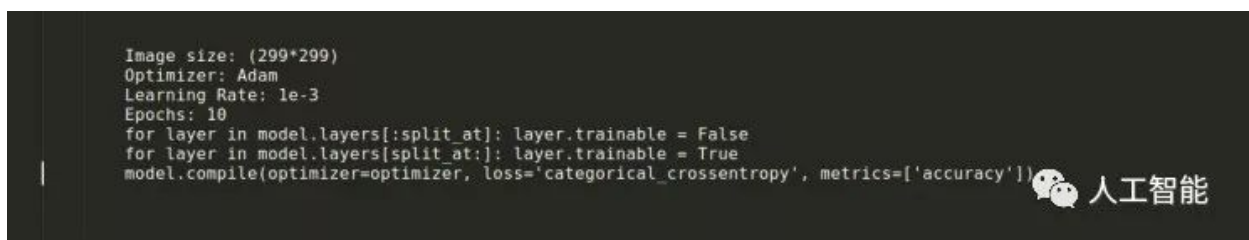
```
Epoch 10/10  
149/149 [=====] - 261s 1s/step - loss: 0.3721 - acc: 0.8991 - val_loss: 0.3376 - val_acc: 0.9011
```



训练了模型底部几层之后得到的实验结果

此外，在这个过程中，我们使用的超参数总结如下：

```
Image size: (299*299)  
Optimizer: Adam  
Learning Rate: 1e-3  
Epochs: 10  
for layer in model.layers[:split_at]: layer.trainable = False  
for layer in model.layers[split_at:]: layer.trainable = True  
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```



第三步：

一旦准备好了对比基准，我们就需要开始对其进行改进。首先，我们可以进行数据增强处理，增加数据集中的图像数。

没有数据，就没有机器学习！

但是正如上文所述，我们得到的数据集是不平衡的，我们需要对其进行平衡化处理，从而使用于训练模型的每一批的数据中都有均匀分布的 12 类图像。

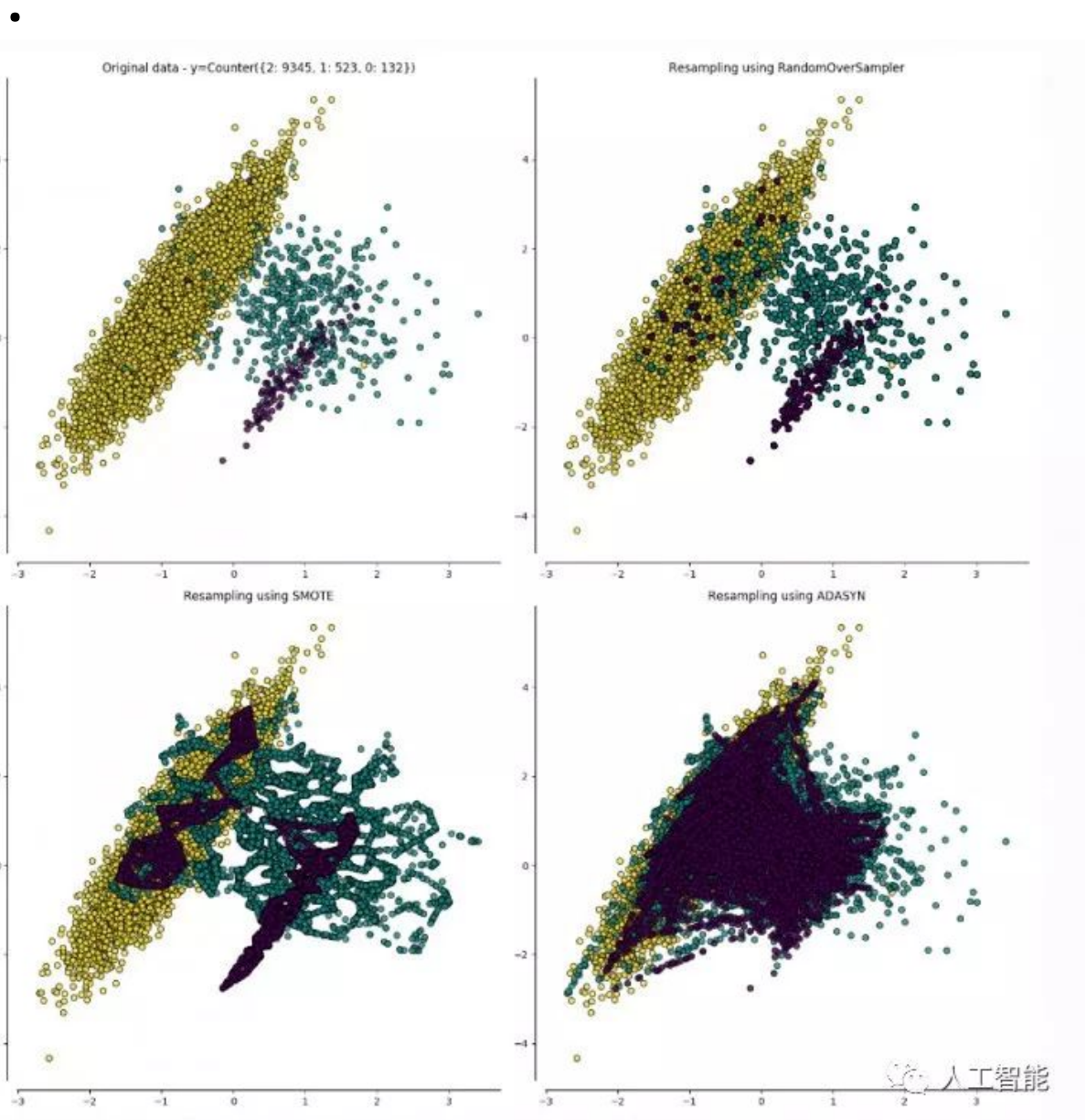
现实生活中的数据集往往都是不平衡的，而模型在样本数量较少的类别上的性能并不太好。所以，将一个具有少数样本的类误分类为一个样本数量较多的类的成本通常要比将数量较多的类误分类高得多。

由此，我们尝试使用两种方法来平衡数据：

- 针对不平衡学习的自适应样本合成方法 (ADASYN)：ADASYN 为样本较少的类生成合成的数据，这种方法会生成更多较难学习的数据集样本。

- ADASYN 的核心思想是，根据学习的困难程度，对样本数少的类别实例使用加权分布。ADASYN 通过两种方法提高了对数据分布的学习效果：（1）减少类别的不平衡所带来的偏差。（2）自适应地将分类的决策边界转换为更困难的样本。[5]
- 少数类过采样技术（SMOTE）：SMOTE 包括对少数类的过采样和多数类的欠采样，从而得到最佳抽样结果。

我们对少数（异常）类进行过采样并对多数（正常）类进行欠采样的做法可以得到比仅仅对多数类进行欠采样更好的分类性能（在 ROC 空间中）。[6]



重抽样结果示意图[7]

在此用例中，可以证明 SMOTE 算法的结果更好，因此 SMOTE 的性能优于 ADASYN 算法。当数据集处于平衡状态后，我们就可以继续进行数据增强工作。

我们可以通过许多途径实现数据增强，其中最重要的一些方法如下：

缩放

裁剪

反转

旋转

平移

加入噪音

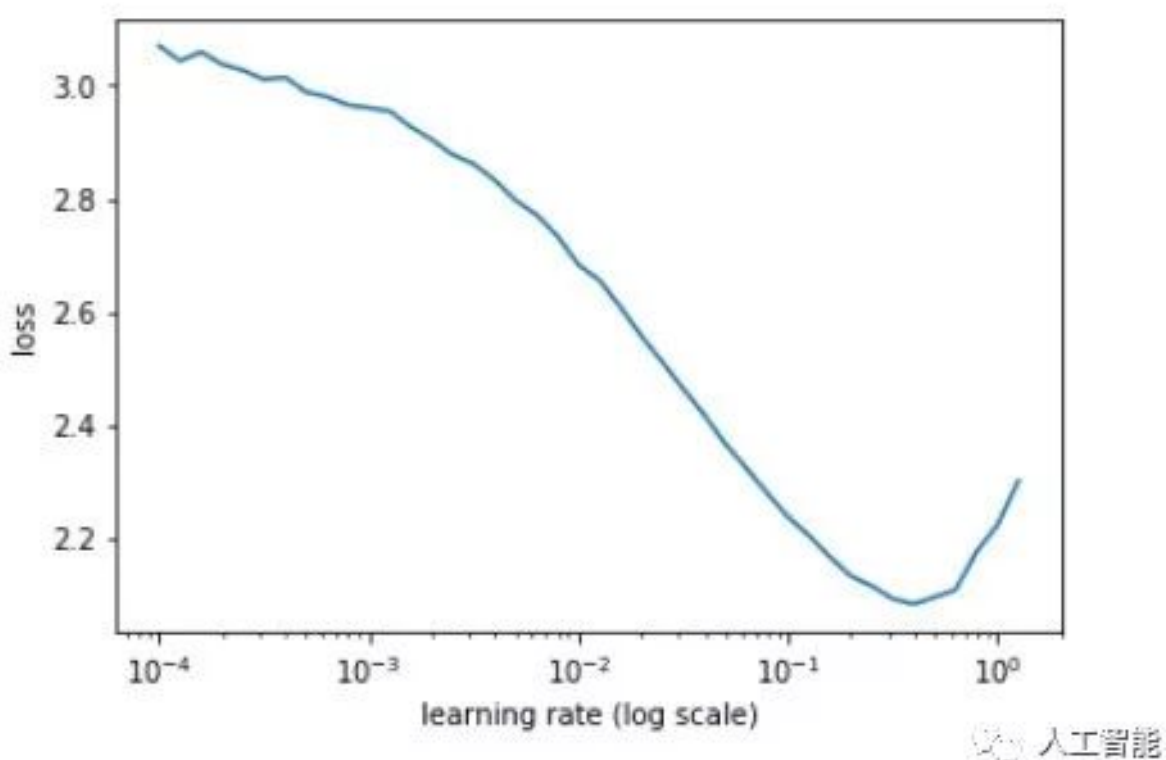
改变光照条件

使用 GAN 这样的先进技术

第四步：

接下来，我们将进一步提升模型结果。在这里，我们将对学习率进行优化，这里涉及到周期性学习率（cyclical learning rate）和带热重启的学习率（learning rate with warm restarts）技术。在此之前，我们需要为模型找到可能的最佳学习率。这是通过绘制学习率和损失函数的关系图来实现的，这一图像用来查看损失函数值从哪里开始下降。

本文描述了一种设定学习率的新方法——周期性学习率，它实际上让我们不必再通过大量实验找到全局学习率的最优值和最佳学习计划。这种方法并不是单调地减小学习率，而是让学习率周期性地在合理的边界值之间变化。利用周期性学习率代替固定的学习率进行训练，能够有效地在不用进行调优的情况下提升分类准确率，需要的迭代次数往往也更少。[11]



学习率和损失函数的关系示意图

如上图所示，0.1 看上去似乎是一个不错的学习率。但是随着越来越接近全局最小值，我们希望用更小的步长对最佳学习率进行探索。「学习率退火」是一种实现这种方法，受到这篇论文 (<https://arxiv.org/pdf/1608.03983.pdf>) 的影响，我选择使用带热重启的学习率。同时，我们将优化器从 Adam 变为随机梯度下降 (SGD)，实现了带重启策略的随机梯度下降 (SGDR)。

接下来，我们可以使用上述技术训练一些模型架构，然后将这些模型得到的结果进行合并。这也就是所谓的模型集成，如今它已经成为一种流行技术，但是这种技术也将带来高昂的计算开销。

因此，我决定使用一种名为快照集成 (snapshot ensembling) 的技术，通过训练一个单一神经网络来达到集成目的，并且沿着优化路径收敛到几个局部最小值，最终将模型参数保存下来。

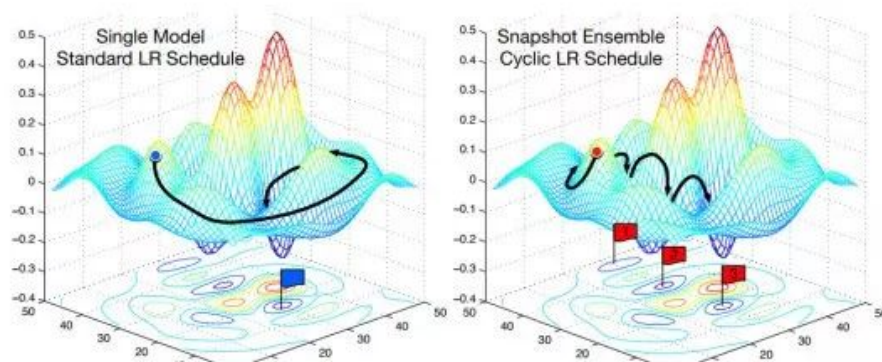


Figure 1: **Left:** Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. **Right:** Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling.

左图：使用传统学习率计划的随机梯度下降优化示意图。在训练结束时，模型收敛到一个最小值处

右图：快照集成示意图。模型经历了几个学习率退火周期，在从多个局部最小值中逃离出来后，收敛到某最小值处。我们为测试时集成的每一个最小值建立了一个快照。

图片来源：<https://arxiv.org/abs/1704.00109>

当学习率被固定下来后，我开始调整图像的大小。我训练了一个针对于 64*64 图像大小的模型（在 ImageNet 上对其进行调优），解冻某些层，对其应用周期性学习率和快照集成技术，获得该模型的权重。将图像的尺寸改为 299*299，并且再次利用图像大小为 64*64 的权重对其进行调优，并采用快照集成技术和带热重启的学习率。

如果改变图像的大小，需要再次采用周期性学习率、快照集成、热重启等技术寻找学习率和损失函数之间的关系，获得最佳学习率。

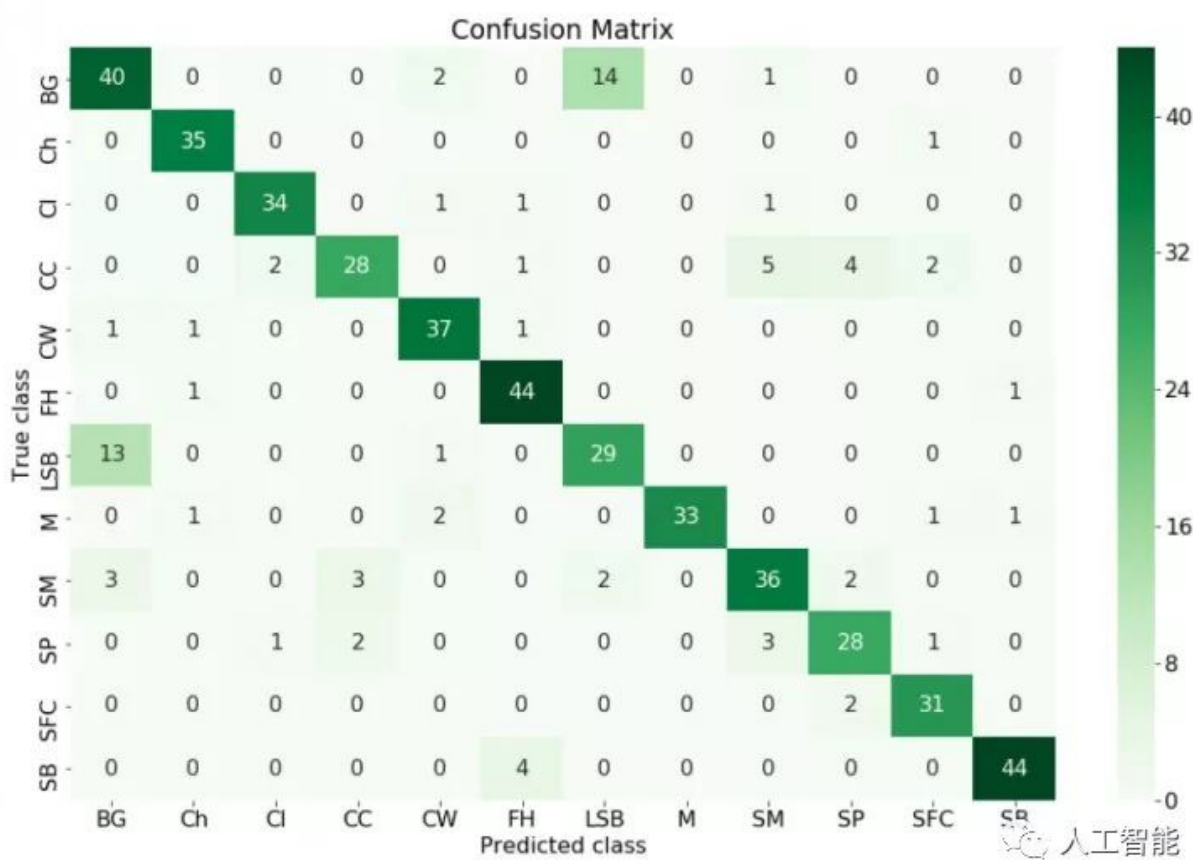
第五步：

在最后一步，我们将对结果进行可视化，看看模型对哪个类别的预测结果最好、对哪个类别的预测结果最差，并且我们还可以采取必要的措施进一步改进结果。

构造一个混淆矩阵是理解模型结果的好方法。

在机器学习领域，特别是统计分类问题中，混淆矩阵（也称为误差矩阵）是一个特定的表格，它能够将算法的性能可视化，这种算法通常是监督学习算法，在非监督学习领域它通常

被称为匹配矩阵。矩阵中的每一行代表预测类别中的一个实例，而每一列则代表真实类别中的一个实例（反之亦然）。这个矩阵之所以被称为「混淆矩阵」，是因为它能够让人很容易地看到系统是否混淆了两个类（即通常将一个类错误标记为另一个类）。



混淆矩阵中真正的类别和预测出的类别

从混淆矩阵中我们可以看到所有的模型预测类别和真实类别不符的情况，我们可以采取措施去改进模型。例如，可以做更多的数据增强工作，试着让模型更好地学习到分类规则。

最后，我们将验证集与训练数据合并，并通过已经得到的超参数，对模型进行最后一次训练，在最终提交结果之前对测试数据集进行评估。

#	Team Name	Kernel	Team Members	Score	Entries	Last
1	Kumar Shridhar			0.99496	22	7m
Your Best Entry Your submission scored 0.99496, which is an improvement of your previous score of 0.99244. Great job!						
2	Michael_			0.99244	25	18d
3	Thiago			0.98992	9	1mo
4	SM			0.98992	40	20d
5	James Requa			0.98866	4	1mo
6	Motiur Rahman			0.98740	3	24d
7	Abdelrahman Ahmed			0.98740	9	1mo
8	Ankit Goila			0.98740	13	22d
9	Vikrant Raj Behal			0.98614	10	1mo
10	Vitaly Bushaev			0.98614	24	17d
11	Michal Zurek			0.98614	36	9d
12	Alex Rass fast.ai			0.98488	10	1mo
13	fastai partha-ramesh			0.98488	2	1mo
14	HOOYAO			0.98488	1	25d
15	ManishChablani			0.98236	27	19d
16	Krishna fast.ai			0.98236	3	19d
17	hlic			0.98236	12	7d
18	Dipiyoti Bisharad			0.98236		
19	Jeremy Howard			0.98110	4	1mo

本文模型在最终提交后名列第一

请注意：训练中使用的数据增强结果需要在测试集中显示出来，以获得可能得到的最佳结果。

参考文献：

- [1] <https://www.kaggle.com/c/plant-seedlings-classification>
- [2] <https://arxiv.org/abs/1711.05458>
- [3] <https://vision.eng.au.dk/plant-seedlings-dataset/>
- [4] <https://keras.io/applications/>
- [5] <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4633969&tag=1>
- [6] <https://jair.org/index.php/jair/article/view/10302>

- [7] http://contrib.scikit-learn.org/imbalanced-learn/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html#sphx-glr-auto-examples-over-sampling-plot-comparison-over-sampling-py
- [8]<https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>
- [9]<https://medium.com/y medialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9>
- [10] <https://arxiv.org/pdf/1608.03983.pdf>
- [11] <https://arxiv.org/pdf/1506.01186.pdf>
- [12] <https://arxiv.org/abs/1704.00109>
- [13]<https://vision.eng.au.dk/plant-seedlings-dataset/>
- [14] <https://lvdmaaten.github.io/tsne/>

想要了解更多资讯，请扫描下方二维码，关注机器学习研究会



机器学习研究云订阅号

转自： 人工智能