## 代表值和梯度下降追踪分析训练数据对模型的影响

1、Chih-Kuan Yeh, Joon Sik Kim, Ian E.H. Yen, et al. Representer Point Selection for Explaining Deep Neural Networks [C]// NeurIPS 18: 9311-9321.

2、Garima, Frederick Liu, Satyen Kale, et al. Estimating Training Data Influence by Tracing Gradient Descent [C]// NeurIPS 20.

## Representer Point Selection for Explaining Deep Neural Networks

## (Carnegie Mellon University)

# Importance of the application

* 随着深度学习开始得到更广泛的应用，我们开始关注的不仅仅是模型预测的准确率和速度，更应该去理解为什么它会做出特定的预测。虽然我们可能并不需要在使用一个复杂的系统（比如汽车）时完全理解它，只需要信任它的原理并掌握使用方法即可，因为我们的信任建立在会有相应的专家对这个系统有透彻的理解。然而在深度学习中，即使是专家也也不清楚为什么说深度神经网络为什么会做出特定的预测。因此，提升人们对模型的理解显得尤为重要，通过这些理解能够对模型做出更合理的调优（如数据集和模型的调试），或者为模型的训练开阔思路以获得更好的性能。
* 在之前的可解释性方法中以影响函数（Influence Function, IF）最为流行，旨在理解每个训练样本对模型的影响，通过删去或扰动某个训练样本后，衡量最后模型对做出特定预测的变化有多大。但是IF的计算需要计算海森矩阵，开销难以接收。

# Problems cannot be solved with existing methods

* 影响函数的计算开销过大，需要一个更轻量级的方法。
* 影响函数在解释模型时的出发点是通过删去某个训练样本，再次训练模型后对特定预测的置信度的改变情况，本质上也并没有深入去理解模型的架构问题。

# Formal definition of the problem

对于一个分类问题，输入空间为（即图片），其对应的输出空间（即标签），给定训练集及相应的标签。采用一个神经网络作为预测模型：，其中，是对应于样本的，在模型最后一层的输入特征。是一个矩阵（最后一层），是剩余的所有参数（特征提取）用来生成最后一层的输入特征。所以就是模型的全部参数。将模型拆成两部分，就可以把模型看作是特征提取层和以为参数的预测层，其中特征提取层可能是很深或者很宽的，只需把它看作是一个功能函数即可。

目标是理解，一个特定的训练点是如何影响模型做出预测的。使用作为损失函数。假设我们已经获得了充分收敛的模型参数，其中为正则化项。我们将把最终的输出重写为线性形式：。就可以看作是训练点对模型做出输入测试点后做出预测的贡献值。

# What are the technical challenges with this approach?

* 如何推导并计算出代表值，即？
* 如何确保如何适用于各种类型的模型？

# Proposed Methods for these challenges

* 拆分模型为特征提取层及最后的预测层，训练样本经过特征提取层的处理后，再将最后一层的预测重写成线性组合形式，得到最后的重要性代表（Representer value）。
* 提出两个假设，假设模型已得到充分收敛，这是为了公式推导的必须前提；而是需要正则项，推荐使用L2正则项，对于没有正则项的模型，也可以通过加入正则项再微调即可，这是为了简化计算的一个技巧。

本文提出代表值Representer Value 分析对于一个测试样本来说，各个训练样本对模型做出决策的贡献程度。本文将预测分解为训练样本的加权重要性组合，权重就反应了训练样本的重要程度。根据各个训练样本对模型预测结果的影响是好还是坏分成两类：Positive，支持模型做出决策；Negative，抑制了模型做出预测的置信度。其框架如图1所示，将模型拆成了两部分，特征提取层视作黑盒（这里的目标是为了理解样本对模型预测的影响，暂时先不考虑模型自身），其输出的特征与代表值进行线性组合后做出最终预测，这里的权重即为所需要求解的样本重要程度，再给定一个特定测试样本，就可以算出每个训练样本对该测试样本的代表值。

其推导如图2所示。假设的充分收敛保证了，已达到最优的模型参数，则最后一层的损失梯度等于0，移项后左边为最后一层的模型参数，右边变形，成为特征的线性加权组合，将测试点送入模型，最终的预测结果就是各个训练点特征测试点特征的线性加权组合，然后一项一项的看就是每个训练点对测试点的代表值了。最终的输出可以分解为训练点特征与测试点特征的加权乘积和。而引入正则项后方便能够用移项来计算，如果没有正则项可以通过微调最后的预测层参数即可，不用花费较大的代价。

实验表明，Representer Value 相对于影响函数IF来说能够更合理的分辨出训练样本对模型做出特定预测的贡献，并且在计算开销上具有数量级的优势

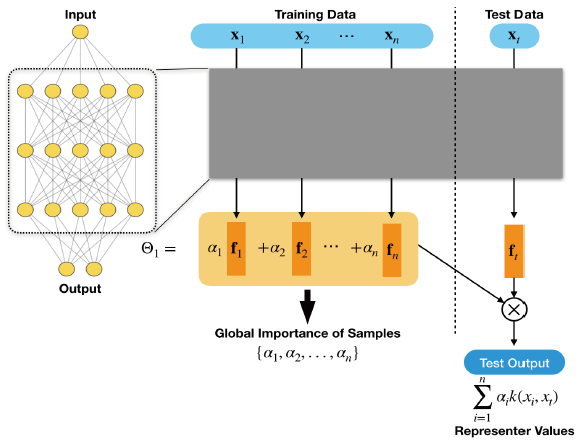


图1. Representer Value的框架理解。

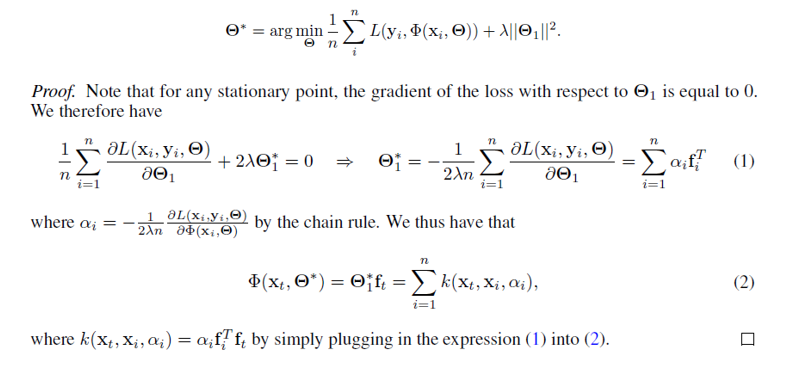


图2. Representer Value的公式推导

**Experiments and Datasets**

* 数据集

论文采用了Animals with Attributes、CIFAR10两个常用的数据集。

* 对比方法

主要与影响函数IF对比，部分实验中也与随机策略对比。

* 实验
  + 数据集纠错

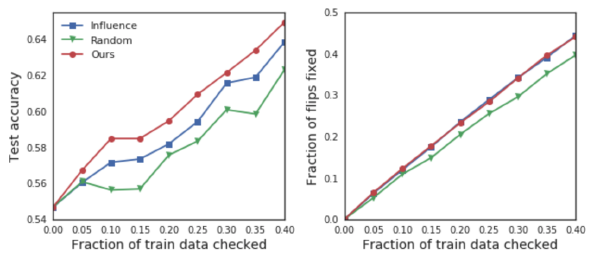


图3.与IF相比的数据集纠错实验

为了验证代表值是否真的能得到训练样本的重要程度，从CIFAR10中抽取马和汽车的两分类，其中有40%训练数据的标签是错的（因为是两分类任务，只要翻转标签就正确了）。用本文方法、IF、随机方法计算样本点的影响，排序后翻转部分标签后的结果。左图中，通过使用代表值检查训练点，与IF相比，代表值能够在检查更少的数据样本的情况下更正确的找到标错的样本（因为错误标记的样本可以视作对模型具有坏的影响，而正确标记的样本则对模型有正面影响，检查绝对值最大的代表值就很可能找出误标的数据）。而右图中，随着被纠错的数据增多，模型的精度也逐渐增大。

* + 样例可视化测试

给定某个测试样本，使用代表值分别挑选出三个具有Positive和 Negative的训练样例来，检验方法是否真的最初了合理的打分。

从图4中可以看到，代表值寻找到了与测试样本很像且标签一致的样本，这对模型的预测做出了支持。而一些图像看起来像测试图像，但其实标签并不是，这就给模型带来一定的困扰，使得模型对作出预测的置信度下降。而在图5中，影响函数在正例的寻找表现良好，但是在负例寻找时却只是找到了一些不是那么有意义的样本，而代表值仍表现良好，寻找到了图像内容相似但标签实际不一致的负例，鲁棒性更好。



图4. 用代表值找出具有最大正负影响的样本例子

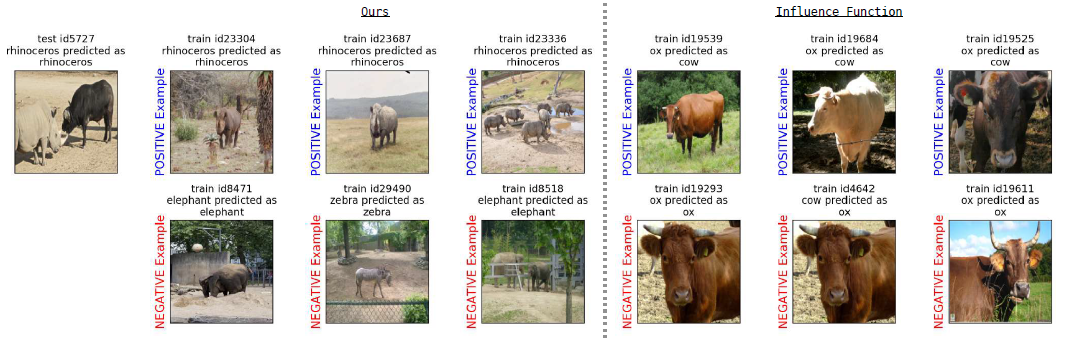


图5. 用代表值找出具有最大正负影响的样本例子

* + 误分类理解

为了加深对模型为什么做出了误分类的理解，本文给出了一个羚羊被错分成鹿的例子，希望通过代表值来解释。图6给出了四个Negative Representer Value最大的训练样本 ，在这些样本中除了被标记的动物外（主体），还出现了羚羊，这类样本使得标签强迫模型关注主体，而抑制了模型对输出羚羊的预测。

181幅羚羊测试图像中，误分类实例总数为15个，其中12个误分类为鹿，在这12个误分类的羚羊测试图像中，都有图6所示的4个训练图像，说明这四个图对误分类的影响最大。在标记为斑马或大象的图像中存在羚羊。训练数据的标签中的这些噪音使模型感到困惑，模型看到大象和羚羊时，标签迫使模型只关注大象。因此，给定一幅包含小羚羊和其他大型物体的图像，该模型学会了抑制羚羊类。

因此可以使用多标签预测来训练网络，或者清理数据集来删除这些同样会让人也感到困惑的训练示例。有趣的是，模型在显示的第二个训练图像(中间的图片)上犯了同样的错误(预测鹿而不是羚羊)，这表明对于训练点，我们应该预计大多数错误分类也是鹿。实际测试中，在863张羚羊训练图像中，有8张被误分类，其中6张被误分类为鹿。



图6. 误分类理解

* + 计算性能测试

从图7以看到，Representer Value 比 Influence Function计算速度快了很多，其中CIFAR10更是具有几个数量级的提升，具有明显的性能优势，因为省去了IF中的海森矩阵计算。但只选了50个测试样本做平均得到的结果，方差显得有些大，尤其是CIFAR-10。

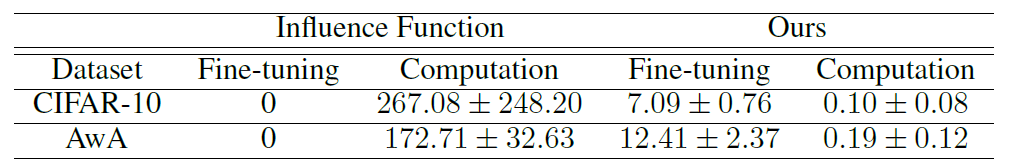


图7 相较于IF的计算效率比较

* + 在NLP领域的应用

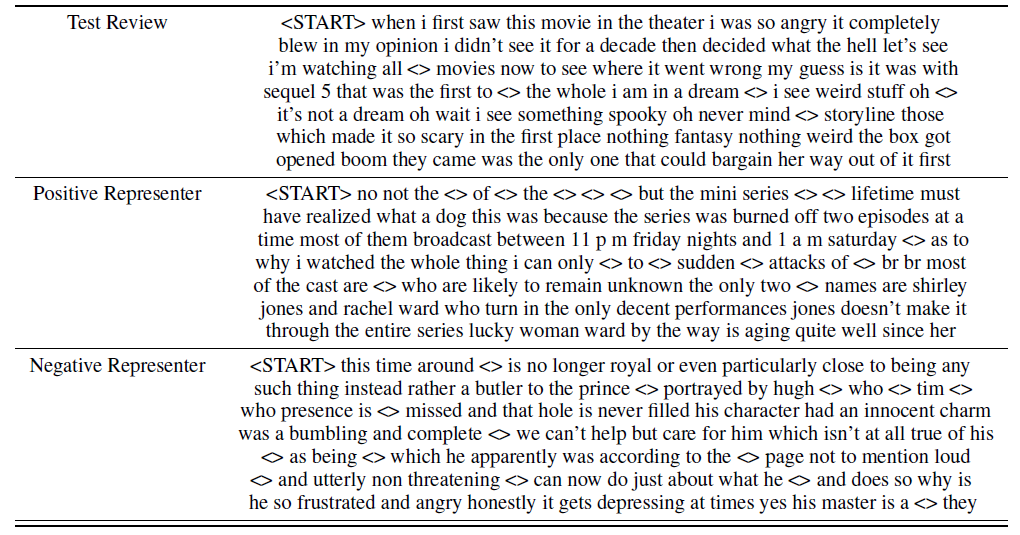


图8在NLP领域中的简单应用实例

# Differences of this Method compared to others

代表值虽然继承了IF希望计算出某个训练点是如何影响模型对特定测试点做出预测的思想，但是走出了完全不同的思路，通过将模型拆分的方式，将模型的预测看作是每个训练样本特征和代表值的线性组合，从侧面计算出来了训练点对测试样本的重要程度，而不再需要通过去掉训练样本后重新训练模型来计算影响程度，本文的方法具有相当的计算优势。

**Limitation of this work or your discovery**

* 关于代表值缺乏量化的比较，即，究竟多大的值算好或者不好。比如选择一个模型预测正确的测试样本，按代表值的绝对值排序训练数据(包括负的和正的)，然后开始删除不重要的前5%、10%，……，重新训练模型，看看什么时候测试样本被错误分类了。这样就可以间接衡量出到底由多少训练样本对模型做出该预测是决定性的。
* 部分实验，尤其是样例展示实验，未免有些主观了。

## Estimating Training Data Influence by Tracing Gradient Descent

## (Google Research)

# Importance of the application

* 深度学习已经被用于解决各种现实问题。机器学习的一种常见形式是监督学习，在这种范式中，模型根据标记数据进行训练。控制模型的训练数据输入是提高深度学习模型质量的主要问题之一。比如，这种技术可以用来识别和修复错误标记的数据。
* 为了对模型做出特定预测有更好的理解，需要有针对性的分析每个训练数据，由于影响函数具有局限性（计算开销，出发点），因此需要提出更实用的技术来更有效的对训练数据进行分析。

# Problems cannot be solved with existing methods

* 影响函数IF扰动输入点后带来的影响，但开销过大。
* 代表值将预测分解为训练样本的加权重要性组合，但需要额外的充分收敛和正则化项假设。
* 都是从IF出发，希望计算出某个训练点是如何影响模型对特定测试点做出预测的思想，但之前的方法的角度都是从收敛后的模型出发，本文希望在训练的过程中去追踪训练的情况来评价任意时刻某个训练点对模型的影响。

# Formal definition of the problem

对于一个分类问题，输入空间为（即图片），用分别代表训练样本和测试样本。训练一个用表示的模型来做预测，我们借助损失函数来衡量模型的表现。给定数据集，我们通过寻找最优的参数集合来最小化损失。使用特定的优化方法迭代优化模型参数，比如随机梯度下降，在第次迭代时，使用了一个样本。那么，在模型参数从更新为的过程中，该训练样本对特定的测试样本的影响就可以被定义为前后的损失变化大小，即，直观理解就是，当某个训练样本加入训练并帮助模型优化参数前后，分别使用迭代更新前后计算出测试样本的损失值，评估前后的损失变化大小，就可以知道该训练样本对测试样本的重要程度。

# What are the technical challenges with this approach?

* 如何在批次梯度下降时应用？
* 如何处理实际训练中测试样本不可见的问题？
* 如何缓解开销较大的问题？

# Possible methods for this challenges

* 虽然训练过程中测试数据不可见，但通过定期存下检查点模型，可以达到按需追溯任意时刻模型的类似效果（如果存下检查点的间隔足够小）。极端的，当每次更新都存下检查点模型时，就等同于理想的情况，当然这样的代价过高，还需要折衷考虑。
* 算法通常会同时访问多个点，而不是单个点，这就需要一种方法来区分每个训练样本的相对贡献。认为一个检查点模型在该阶段记住了所有训练样本参与的“贡献”，使得TracIn能够面向批SGD。对于各个检查点模型来说，其见过的每个训练数据只会出现一次，即两个检查点模型的间隔小于1个遍历完全部数据的epoch。

本文提出了一个能够在对于给定测试样本前提下，跟踪训练过程，捕获各个训练样本被SGD用于更新模型后，对模型预测（或损失）的变化。 从而有效地从各种数据集中找到错误标记的样本或离群值，并为每个训练样本计算影响力分数，有助于理解训练样本（而不是特征）的预测。示例中模型的任务是预测左侧测试图像的主体西葫芦（同时也有安全带，因此很可能被误分类为安全带）。随着训练的进行，各个训练样本都会影响模型对测试图像的损失（预测分数于实际标签的距离）。西葫芦的预测分数越高，损失越低。减少损失的训练样本，如西葫芦图像，被称为 支持者(Proponents)。而增加损失的训练样本，如安全带的图像，被称为 反对者 (Opponents)。在示例中，标有“太阳镜”的图像也是一个支持者，因为图像中虽然有安全带，但被标注“太阳镜”，以便促使模型更好地区分西葫芦和安全带。

而理想情况显然难以实现，因为实际场景中，一般采用的是批次梯度下降而不是单样本的随机梯度下降；还有一个最大的问题就是训练至测试样本不可见；最后一个问题是计算开销。为了解决上述问题，本文提出了一个近似计算方法和一个启发式的替代计算方案。在近似计算方法中，先将理想情况下的原式一阶近似（已经足够有效了，不需代价高昂的二阶近似），然后与SGD更新的公式合并，最后得到近似计算方案。在启发式的替代计算方案中，引入“存档”，也就是检查点模型，不过每次存下的检查点最好是保证当所有数据都被用到一轮后（即每个数据只被用到一次），因为在该检查点模型中，恰好包含了每个数据使用且仅使用一次所带来的影响。由于学习率会衰减，所以要把学习率大小作为权重考虑进去。

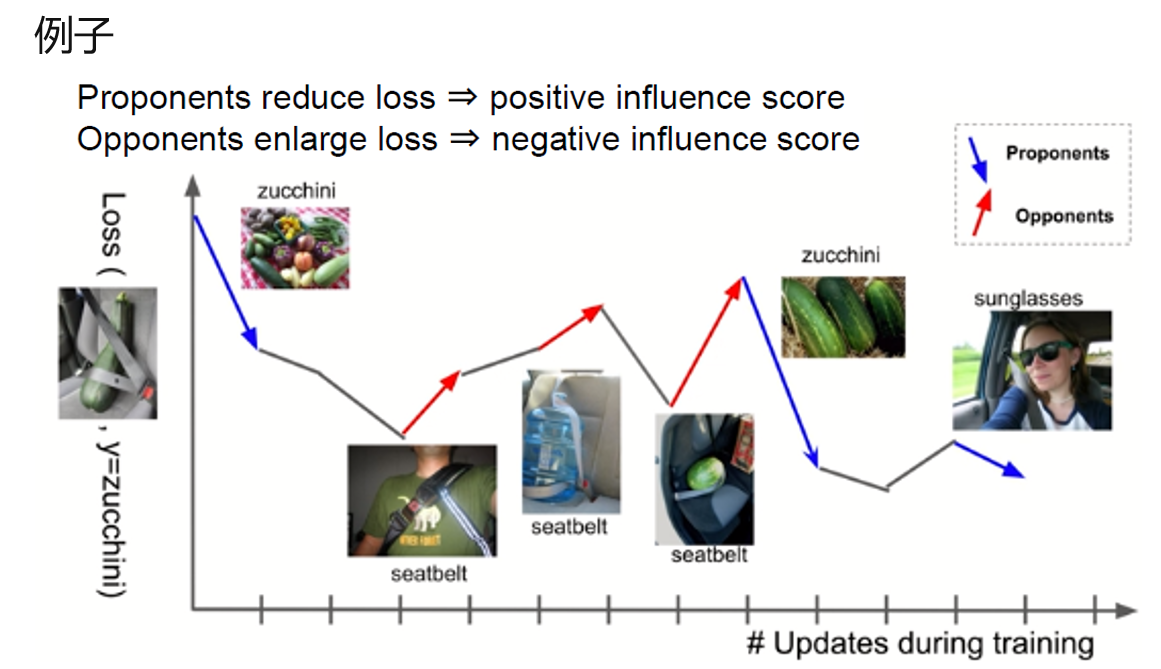


图9. 一个助于理解的样例

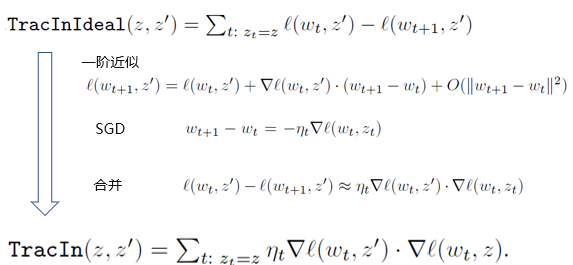


图10. 从理想情况推导的一阶近似 TracIn Influence Score

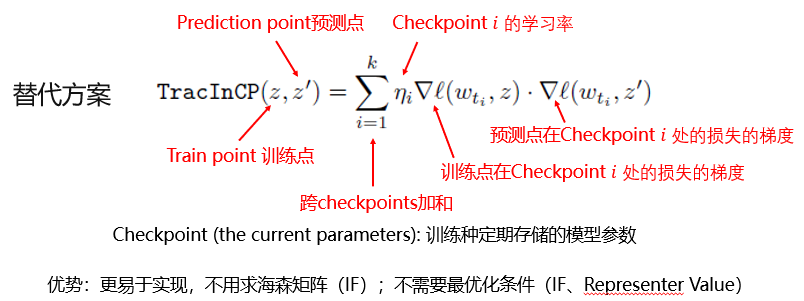


图11. 可以付诸实际应用的替代方案

**Experiments and Datasets**

* 数据集

论文采用了常用的数据集：此CIFAR10，MNIST，ImageNet。

* 对比方法

影响函数IF，代表值Representer Value。

* 数据集纠错

论文选用了ResNet-56 网络去进行 CIFAR-10分类任务，这里只用最后一层的模型参数来计算海森矩阵（IF）、微调（Representer Value）。本文的方法的设置为，从第30个检查点开始存档模型，每隔30个检查点取一个，每个检查点都是在1个epoch刚结束后存下的。 图12的左图随着被检查的数据增多，精度逐渐上升；右图，随着检查的数据增多，错标签的数据被验证的越多，从曲线的走势来看，本文的方法明显由于基线的影响函数和代表值。红色的线是忽略学习率权重，让训练前后的权重值都相等，用来验证学习率对本文方法的影响大小，可以看出其实忽略与否对最终效果的影响不明显。

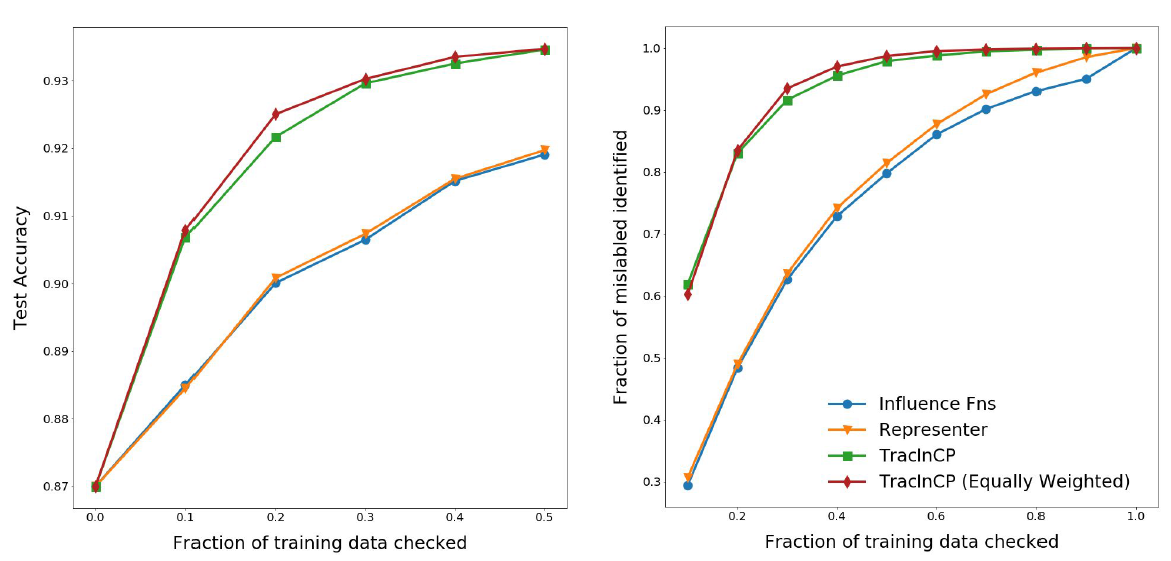


图12. 数据集纠错

* 对比TracIn / TracInIdeal /

为了评估理想方法TracInIdeal与近似方法TracIn和启发式替代方案TracInCP究竟有多大差距，用了三层的全连接简单模型(240k参数)，使得计算整个模型和所有训练步骤的检查点成为可能。图13左图：每个点对应一个测试点（共100个）的一个参数更新步骤，纵坐标为一阶近似计算出来的分数，而横坐标是损失变化的大小（也就是之前提到的想要得到的理想情况），两个量之间的Pearson相关系数为0.978，这个值已经可以使得一阶近似方案足够应对一般场景下的需求了。

右图：分析所选取的检查点所计算出来的TracInCP与一阶近似相比的差距（随机选取了100个测试点），纵坐标越大，越相近。1、选择能大幅降低损失的检查点(第4个)，比选择相同数量等间距检查点(第3个)更好，也比较符合直观，因为当采用等间隔检查点时，后期的检查点损失趋于平缓，所能捕获到的信息不多。2、使用中间检查点（第1个）的效果比使用最后一个检查点(第2个)更好。3、总的来看，拥有更多检查点的TracInCP提高了相关性，如果所选择的检查点具有较高的损失降低率，这种相关性就更强了。4、RP代表随机投影降维，降维越多，信息丢失越多，效果越差。

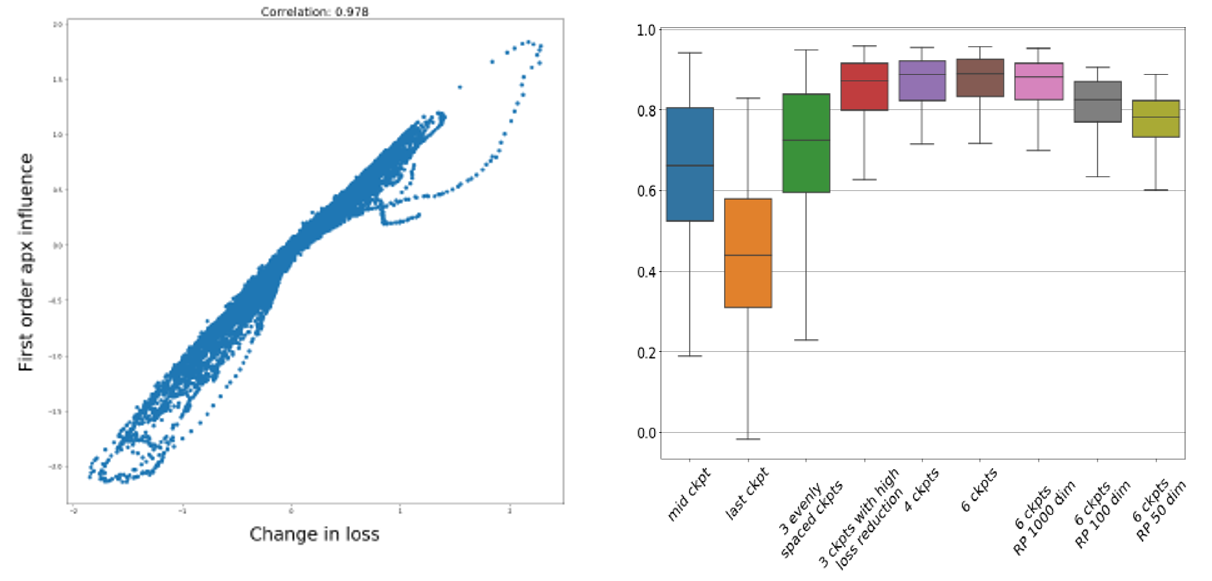


图13. 在MNIST上对比TracIn和TracInIdeal

* Similarity vs. TracIn

Similarity是基于特征相似度的方法。为了展示TracIn认为的重要的图象是否正确，与Similarity做了对比，给定一个测试样本后，分别选出支持样本和反对样本，以及Similarity所认为的最相似的和最不相似的训练样本。本文方法找到的支持者都是变色龙，而反对者与变色龙长得一样，但却是其它标签的动物，这使得模型强制向其它方向训练，进而成为了反对者。至于Similarity在选出最相似样本时则只是选出了相似的图像（而不在一他们是不是同一类），而且Similarity所选出的最不相似图像则完全是不想关的，不具备太大参考价值。

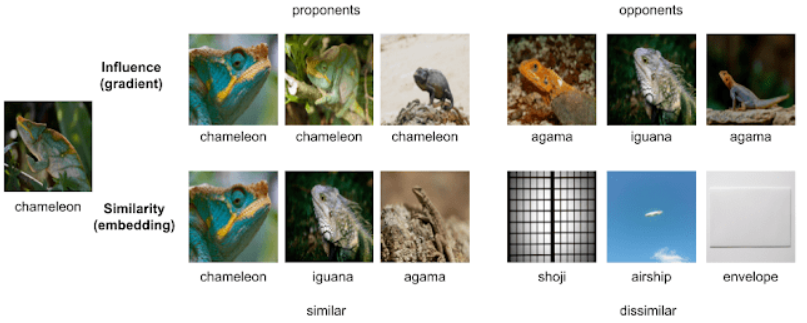


图14. Similarity vs. TracIn

* 自我影响力

除了测试点数据的分析以外，还能对训练点本身分析，即自我影响力（Self-Influence），当时，训练样本对自身的影响力。图15中，一个仅带有振荡的罕见示波器样本，表现出很高的自我影响力。而其他常见的示波器图像，带有旋钮和电线。可能会被预测为别的东西（电视），较低的自我影响力，对模型的预测有一定的坏处。高自我影响力更有可能代表了正确的记忆信息（indicative of memorization.），记忆是合理的;相反，低自我影响力记忆会损伤模型性能。

对于图16中复杂的场景，如多主体：网球和球拍，颈托和拐杖，西葫芦和安全带；整体和局部：商店和蟹，理发店和理发椅，护膝和排球。高自我影响力的样本也可能会带来坏处，在这些复杂场景下，高自我影响力的样本促使模型抓住了其他信息而走向了错误方向。

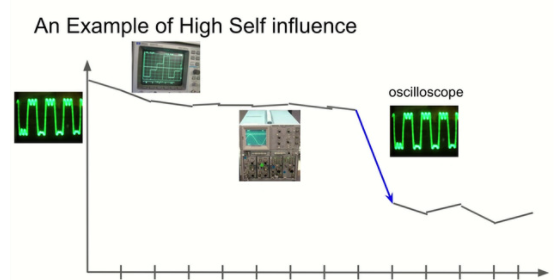


图15. 自我影响力在示波器样例上的测试

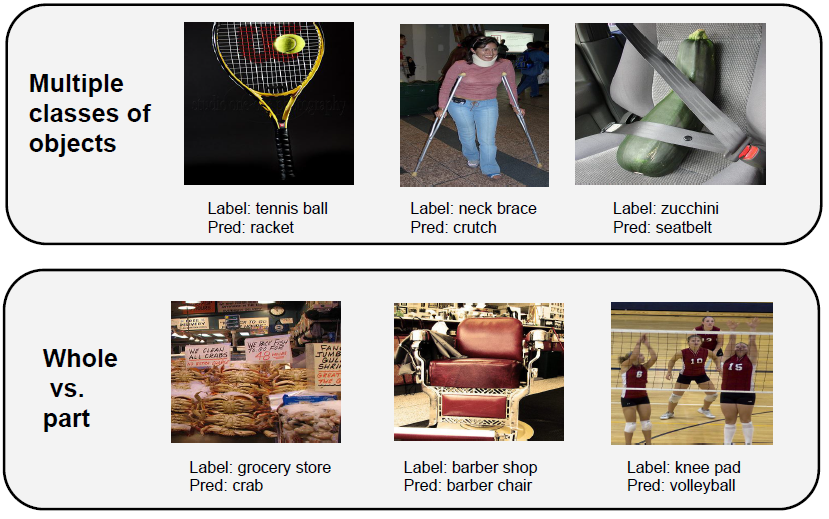


图16. 自我影响力在复杂样本上的应用

* 应用示例

将本文的方法在ImageNet的应用，最左列时测试样本，第二到四列时找到的影响最大的支持者，最后三则是负面影响最大的反对者。对于所找到的支持者，跟测试图像标签一致，让模型更加坚定自己的预测结果；或者跟虽然跟测试图像不一致，但包含了测试图像中其他的元素，促使模型可以更有效的分辨图像中各个元素与标签的关系（比如安全带和葫芦）而对于反对者，虽然跟测试图像相似，但标签却不一致，这就给了模型迷惑，导致模型对测试样本的预测表现出了怀疑，我真的预测对了吗？比如第二列中的城堡，在外形上与测试样本的教堂同属一个风格，看上去非常相似，这就对模型造成困扰。

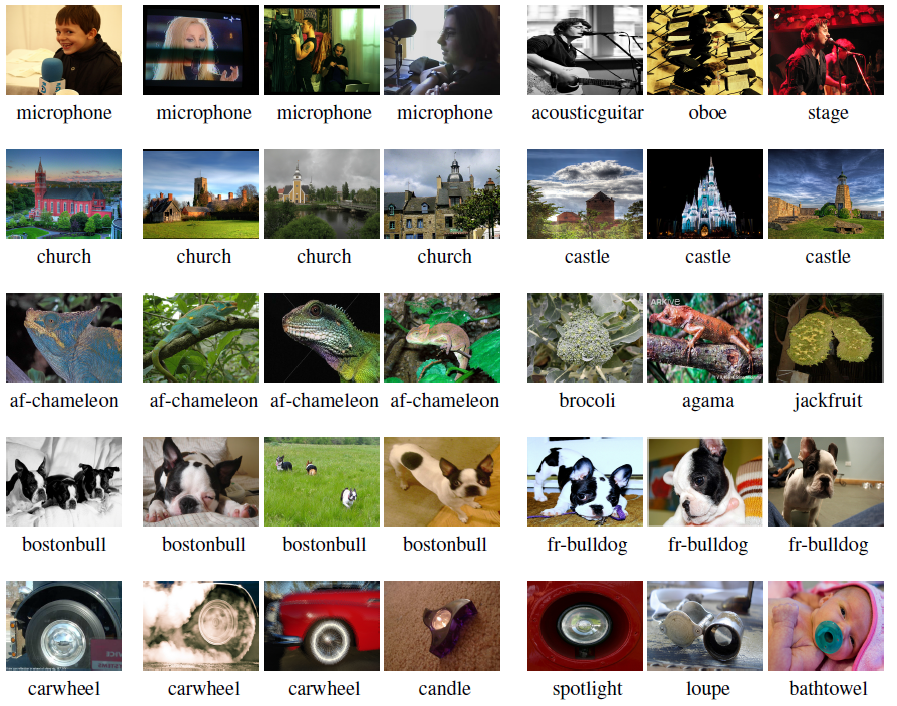


图17. ImageNet上的应用

# Differences of this Method compared to others

不同于影响函数中估计扰动输入点后带来的影响，以及代表值中将预测分解为训练样本的加权重要性组合，本文尝试了从另一个角度出发理解训练过程中训练样本对测试样本的影响。这样的做法开拓了一个全新的思路，同时还不需要付出大量计算代价，也几乎没有额外限制假设。对于给定的测试样本，跟踪训练过程。在训练期间，访问特定的训练样本将改变模型的参数，这种改变会修改测试样本上的预测/损失。如果能够全程跟踪训练样本，那么测试样本上损失或预测的变化即可归因于相关训练样本，其中训练样本的影响力可以被视作训练样本各次访问的累计。

**Limitation of this work or your discovery**

* 检查点模型的存档无疑带来了额外的存储开销，尤其在追求更接近理想情况下的时候（缩小检查点存档的间隙），检查点模型的额外开销将会几何倍数增加。
* 根据图13左图中的结果，一阶近似的做法实际上与理想情况还是有一些偏差，而且实验是在简单的全连接模型和简单的MNIST数据集上的结果，类似的结果能否很有效的扩展到复杂数据集下还未可知（当然从图17.在ImageNet上的应用来看，表现还不错）。