MindSpore

MindSpore自动数据增强

主讲人：Drew

目录

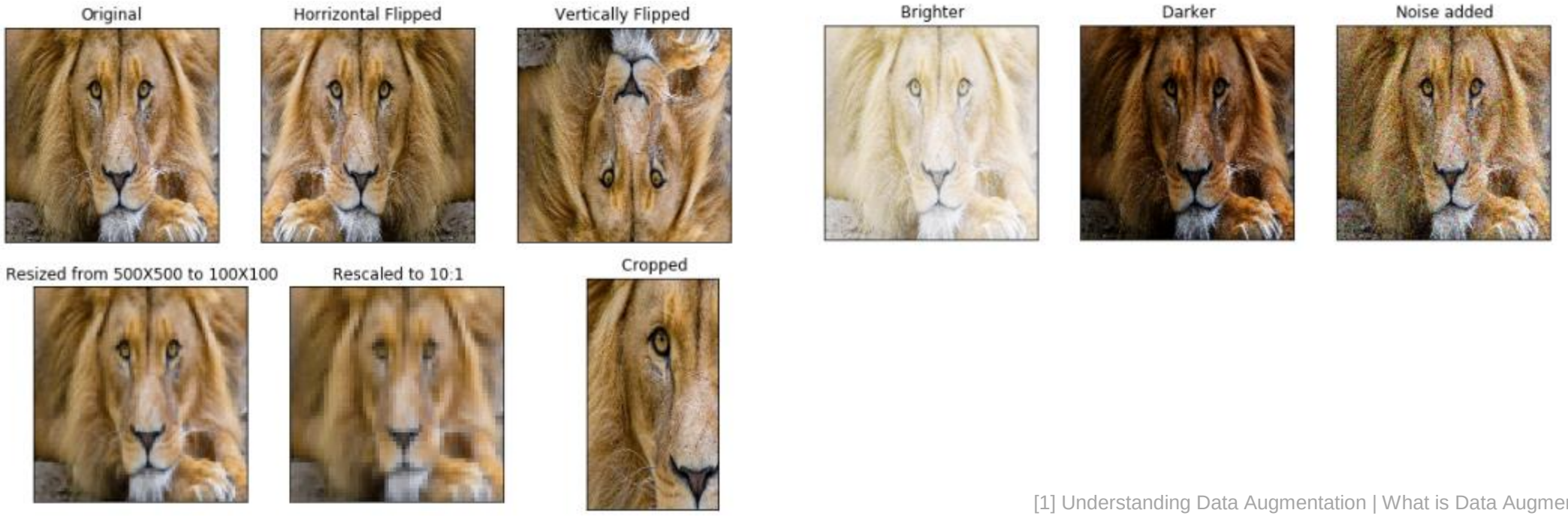
contents

MindSpore

数据增强背景介绍

MindSpore

- **什么是数据增强?**
  - 从可用数据中合成新数据的这种方法称为“数据增强”，有助于解决模型训练中数据有限的问题。
- **常用的方法**
  - 几何变换（例如：翻转、旋转、平移、裁剪、缩放）
  - 颜色空间变换（例如：色偏，亮度调整、噪声注入）



[1] Understanding Data Augmentation | What is Data Augmentation & how it works? , https://www.mygreatlearning.com/blog/understanding-data-augmentation/

数据增强背景介绍

MindSpore

➢ 常用的方法

- GAN（生成对抗网络）

学习实际图片中的特性并生成逼近真实的数据

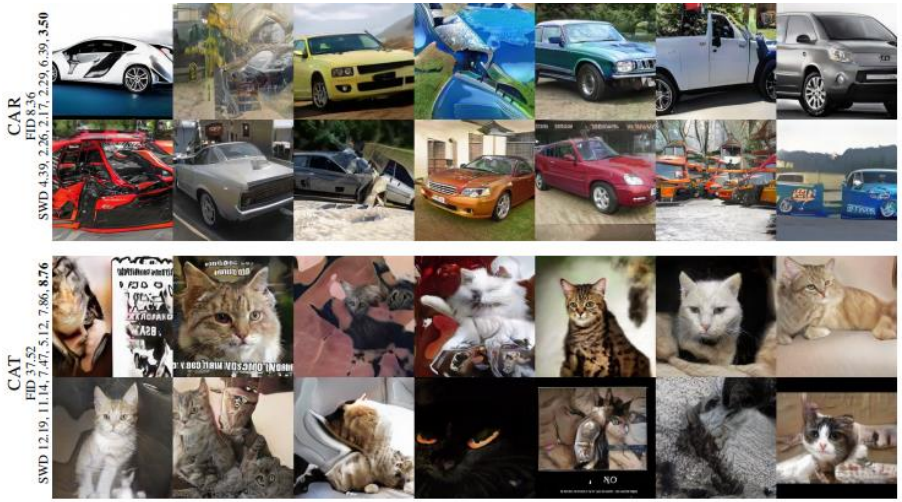
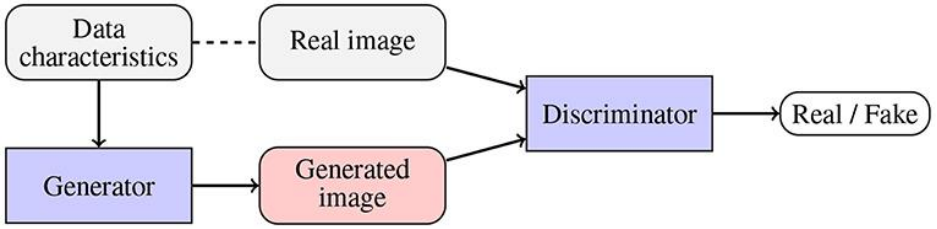
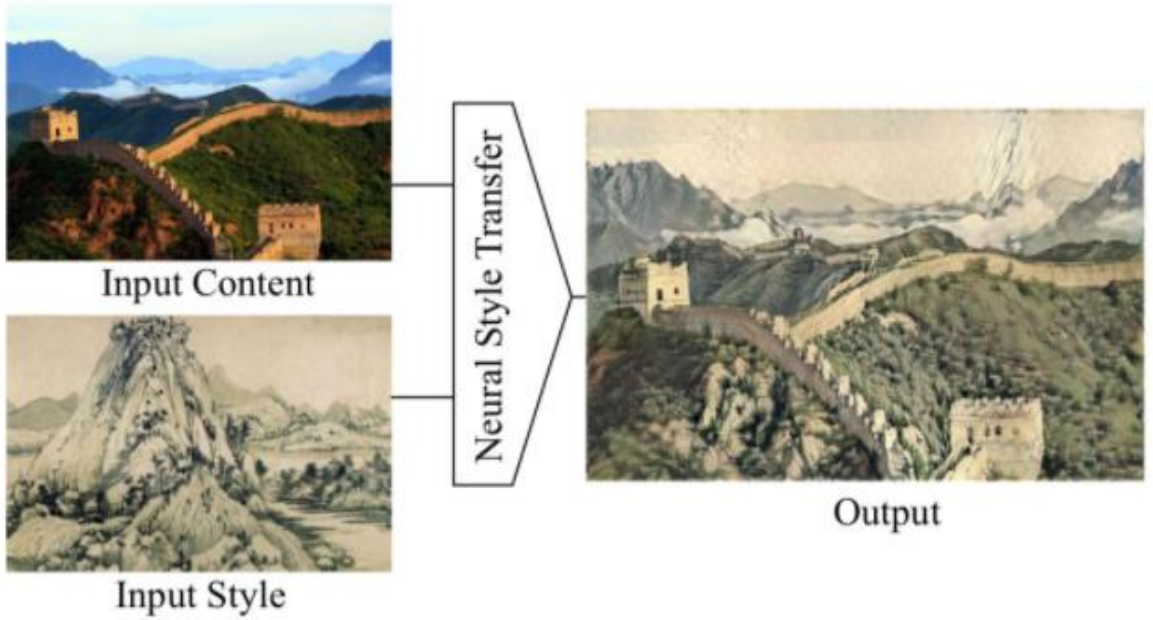


Figure 13: Example images generated at 256 × 256 from LSUN categories. Sliced Wasserstein Distance (SWD) $\times 10^3$ is given for levels 256, 128, 64, 32 and 16, and the average is bolded. We also quote the Fréchet Inception Distance (FID) computed from 50K images.



Neural Style Transfer(图像风格化)

定义:将一张图片的语义内容与不同风格融合起来的过程被称为神经风格迁移



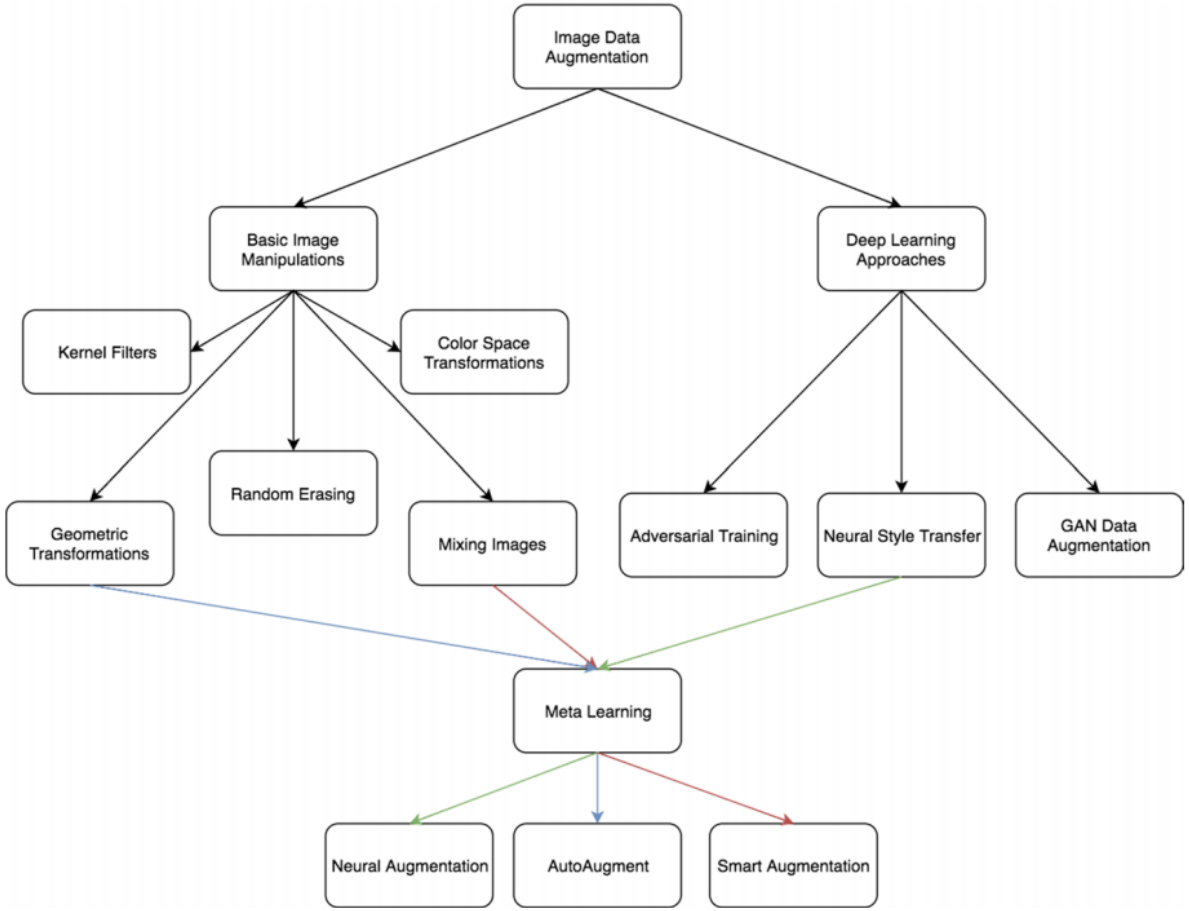
[1] Karras T , Aila T , Laine S , et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation[J]. 2017.

[2] 综述：图像风格化算法最全盘点 | 内附大量扩展应用, https://mp.weixin.qq.com/s/iV-OXiKF1jgAhSmX4QUIXw

数据增强背景介绍

MindSpore

➢ 数据增强方法概览



方法	优点	缺点
几何/颜色空间变换	快速	依赖经验，需手动配置好入参等
(DL Approaches)深度学习方法	自动从数据集学习增强策略	GAN较难训练. Style transfer使用场景有限.
Meta Learning(元学习/学会学习)	依赖原始数据进行数据增强，从中学习最佳的数据增强策略，且可泛化到同类数据集	学习成本高昂，需要大量数据集及学习时间等

[1] Shorten C , Khoshgoftaar T M . A survey on Image Data Augmentation for Deep Learning[J]. Journal of Big Data, 2019, 6(1).

数据增强背景介绍

MindSpore

➢ **数据增强的作用**

- **比喻：** *The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms. — Andrew Ng(吴恩达)*
- **作用：**数据增强可用于，扩充训练数据的多样性和数据量，提高模型泛化性。且还可以用于解决分类任务中的类别不平衡问题。
- **效果：**数据增强前后精度对比:

Application	Performance without Augmentation	Performance with Augmentation	Augmentation method
Image classification	57%	78.6%	Simple Image based
Image classification	57%	85.7%	GAN based
Text classification	79%	87%	Easy Data Augmentation

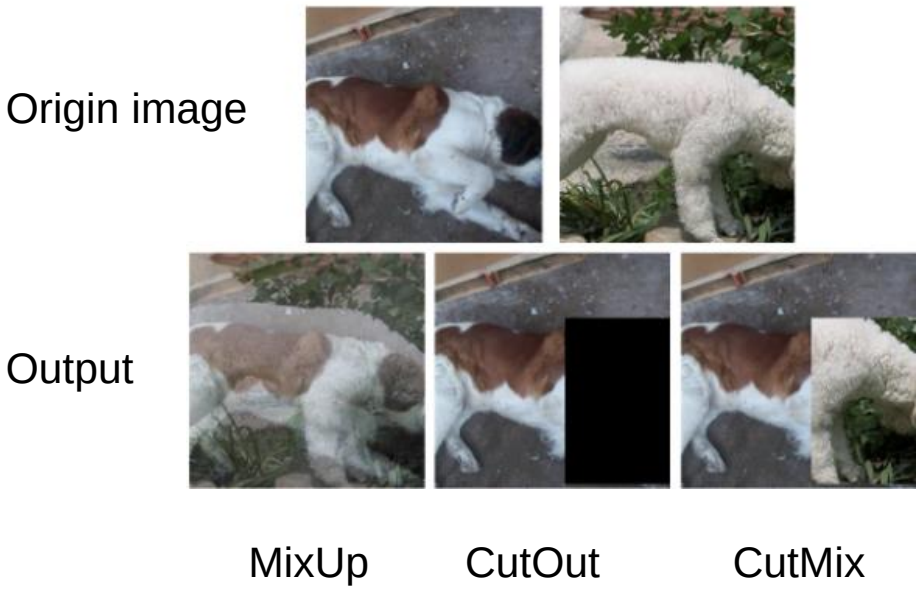
[1] Understanding Data Augmentation | What is Data Augmentation & how it works? , https://www.mygreatlearning.com/blog/understanding-data-augmentation/

数据增强背景介绍

MindSpore

➢ YOLOV4相比YOLOV3的改进之一：数据增强

CutMix



Mosaic(镶嵌）：将四张图片拼接在一起

Figure 3: Mosaic represents a new method of data augmentation.

label smoothing（标签平滑）

通常标签会表示为[0，1，1，0，0]等，然而这显得过分确定适当平滑表示成[0.1，0.9，0.9，0.1，0.1]显得更符合预测概率

Self-Adversarial Training（SAT）：自我对抗训练

依据模型的状态来通知训练漏洞以转换输入图像
图片正向训练后，使用损耗信号量以对模型最不利的方式改变输入图像，使得模型被迫学习这一困难图像（特征）

[1] Bochkovskiy A，Wang C Y，Liao H . YOLOv4: Optimal Speed and Accuracy of Object Detection[J]. 2020.

目录

contents

MindSpore

常规数据增强方法

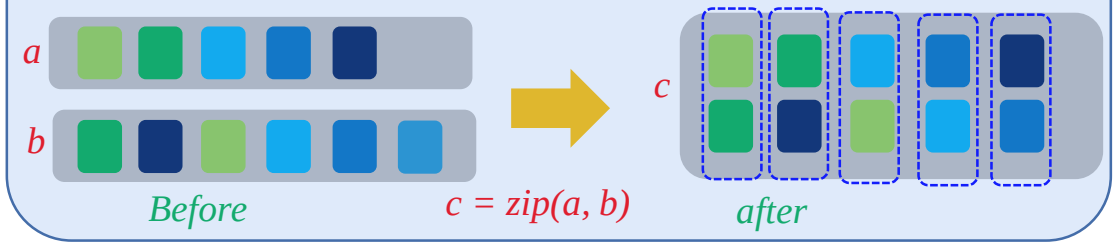
MindSpore

➢ 常用数据集处理操作

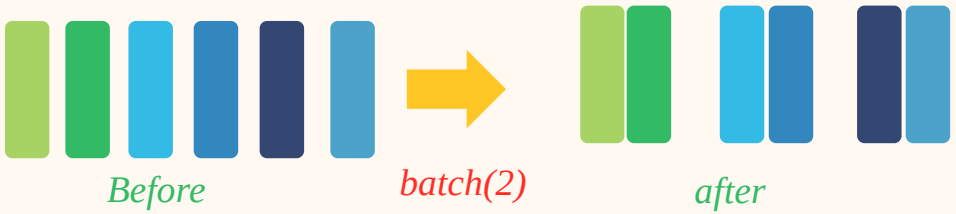
- 常用的数据集处理操作包含：shuffle(打乱顺序）、batch（分批）、zip(列合并）、map(数据增强批处理)

Shuffle： 用来打乱数据集中的数据排序。越大的buffer_size意味着更高的混洗度，但是这也意味着会花费更多的时间和计算资源

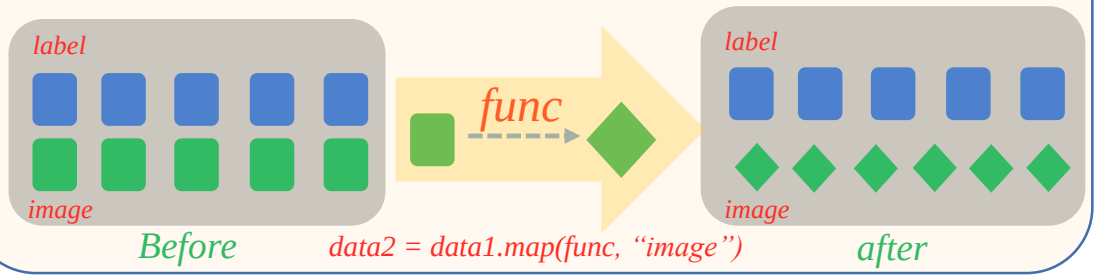
zip: 将多个数据集合并成一个。



batch： 在训练时，数据可能需要分批batch处理。



map:将指定的数据增强操作作用于数据集的指定列数据，实现数据映射操作。



常规数据增强方法

MindSpore

➢ 常用数据集处理操作

- 其他数据处理操作

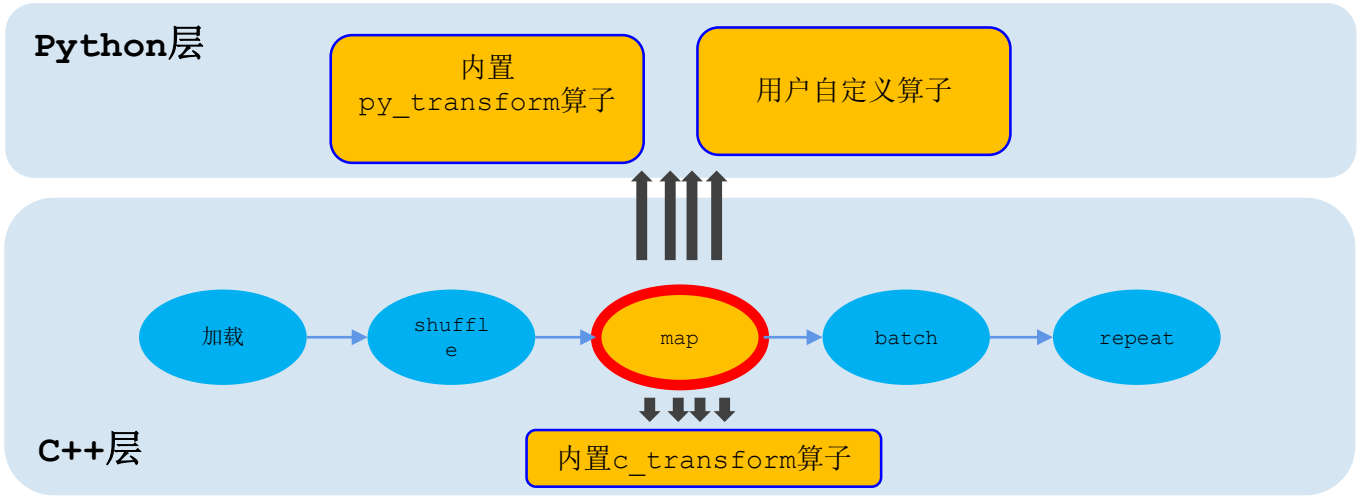
算子名称	说明
concat	将两个数据集连接起来，形成一个更大的数据集
filter	按条件过滤数据集
save	保存数据集至MindRecord格式
skip	跳过数据集的前N条记录
take	只取用数据集的前N条
split	将数据集拆分成训练集、验证集
sync_wait/sync_update	同步等待和更新，主要用于自动数据增强，依据loss等进行策略调整

常规数据增强方法

MindSpore

➢ 数据增强方法

- 数据增强在数据处理管道的位置及使用方式



- **特点**

1）数据增强以map算子方式插入到pipeline中，map算子再调用具体的数据增强算子。

2）对给定的列进行操作

3）能够调整线程数实现高并发

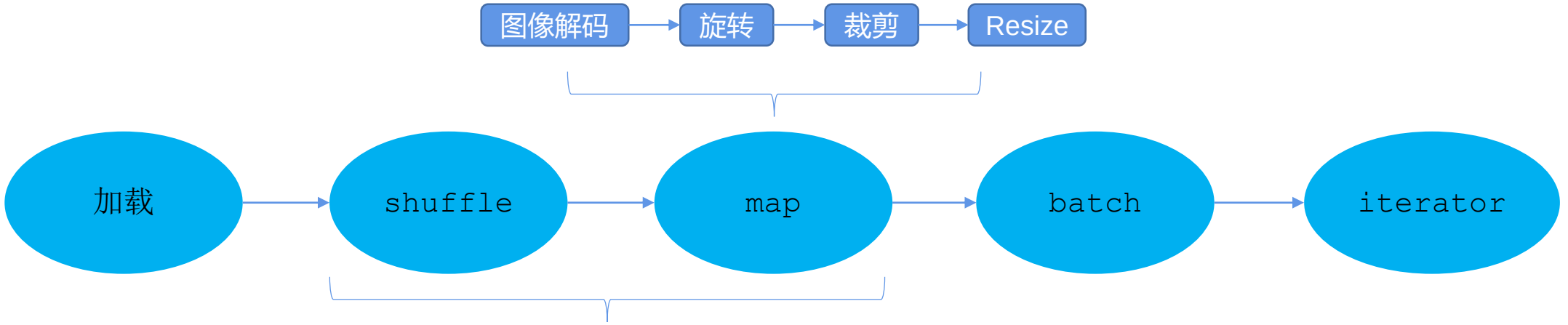
4）灵活及高性能 ①数据处理的pipeline在C++层维护内存管理 ② 内置C算子（c_transform）提供更高的性能

③内置Python算子(py_transform)更丰富的算子支持④ 支持用户自定义python数据增强函数

常规数据增强方法

MindSpore

➢数据增强流程



将用户的训练数据加载至内存缓存中，不同的数据集的读取需要采用不同的xxDataset。

shuffle操作用来打乱数据集中的数据排序。一般情况下，越多数据的一起打乱，会得到更好的训练效果。

调用数据处理相关算子，如：大小缩放、图像旋转、归一化等操作，实现对训练的预处理，以期望达到更好的训练效果。

在训练时，数据可能需要分批batch处理。比如：传给训练过程是100个batch，每batch 32个样本。

迭代输出数据用于进一步训练。

常规数据增强方法

MindSpore

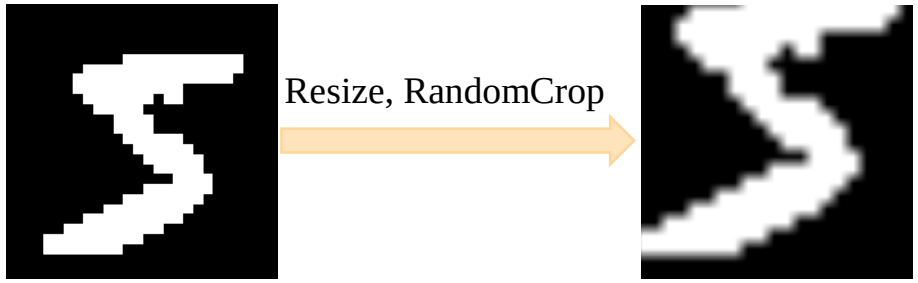
➢ 数据增强算子使用方式

- **c_transform算子使用方式**

```
from mindspore.dataset.vision import Inter
import mindspore.dataset.vision.c_transforms as c_vision

DATA_DIR = './datasets/MNIST_Data/train'

mnist_dataset = ds.MnistDataset(DATA_DIR, num_samples=6, shuffle=False)
resize_op = c_vision.Resize(size=(200,200), interpolation=Inter.LINEAR)
crop_op = c_vision.RandomCrop(150)
transforms_list = [resize_op, crop_op]
mnist_dataset = mnist_dataset.map(operations=transforms_list, input_columns=["image"])
mnist_dataset = mnist_dataset.create_dict_iterator()
data = next(mnist_dataset)
```



自定义数据增强函数使用方式

```
func = lambda x : x*2 # Define the lambda function
ds2 = ds1.map(input_columns="data", operations=func)
for data in ds2.create_dict_iterator():
    print(data["data"])
```

```
[[0 2 4]
 [2 4 6]]
[[4 6 8]]
```

参考：https://www.mindspore.cn/doc/programming_guide/zh-CN/master/augmentation.html#c_transforms

目录

contents

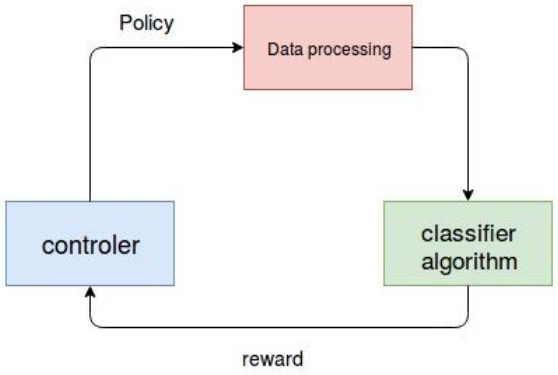
MindSpore

自动数据增强开山之作

[1] Cubuk E D , Zoph B , Mane D , et al. **AutoAugment: Learning Augmentation Strategies From Data**[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.

AutoAugment：旨在**自动化地为目标数据集寻找有效数据增强策略。**

摘要：提出了一种简单的基于搜索的数据增强方法AutoAugment. 作者的主要思想**是创建离散的数据增强策略的搜索空间，并直接在一些数据集上评估特定策略的优劣。**作者设计了一个搜索空间，每一种策略由许多子策略组成，在每一个batch中都会为每一张图像随机选择一种子策略。子策略包含两个操作，每个操作都是一种图像处理方法，如平移，旋转或剪切等，对于每一个操作都有一组概率和幅度来表征这个操作的使用性质。本文使用搜索算法，搜索使神经网络在目标数据集上获得了最高的验证集精度的**最佳数据增强策略**



搜索算法有两部分：一部分是**控制器**(controller,一个递归神经网络)，另一部分则是**训练算法PPO**(Proximal Policy Optimization)算法。 每一步操作中，控制器对softmax输出的结果预测产生决策；然后将决策作为下一步操作的嵌入向量。控制器拥有30个softmax来分别预测5个子策略的决策，每个子策略又具有2个操作，而每个操作又需要操作类型，幅度和概率。

奖励信号定义为某个策略对于原始模型泛化能力的优化程度。实验中预留了一个验证集来衡量评估子模型的精度，评测的结果被用作控制器的奖励信号。

自动数据增强开山之作

近日，来自谷歌大脑的研究者在 arXiv 上发表论文，提出一种自动搜索合适数据增强策略的方法 AutoAugment，该方法创建一个数据增强策略的搜索空间，利用搜索算法选取适合特定数据集的数据增强策略。此外，从一个数据集中学到的策略能够很好地迁移到其它相似的数据集上。

- 在 CIFAR-10 上误差率降低 0.65%，在 ImageNet 数据集上，达到83.54% 的 Top-1 准确率。在 SVHN 上，误差率从 1.30% 降低到了 1.02%。

CIFAR 10

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [56]	3.87	3.08	2.68
Shake-Shake (26 2x32d) [58]	3.55	3.02	2.47
Shake-Shake (26 2x96d) [58]	2.86	2.56	1.99
Shake-Shake (26 2x112d) [58]	2.82	2.57	1.89
AmoebaNet-B (6,128) [21]	2.98	2.13	1.75
PyramidNet+ShakeDrop [59]	2.67	2.31	**1.48**

ImageNet

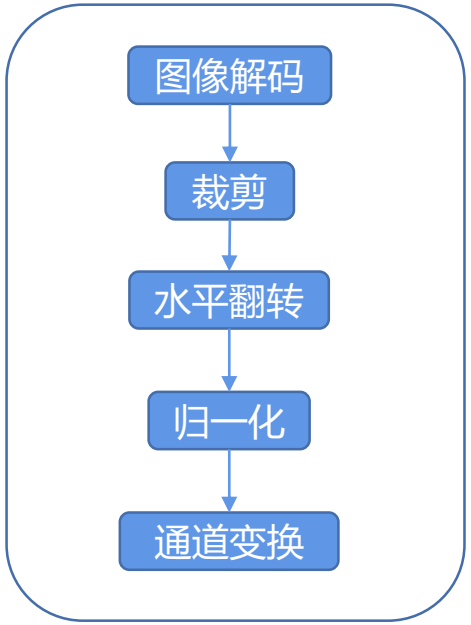
Model	Baseline	Inception Pre-processing [14]	AutoAugment
ResNet-50 [15]	24.70 / 7.80	23.69 / 6.92	22.37 / 6.18
ResNet-200 [15]	-	21.52 / 5.85	20.00 / 4.99
AmoebaNet-B (6,190) [21]	-	17.80 / 3.97	17.25 / 3.78
AmoebaNet-C (6,228) [21]	-	16.90 / 3.90	**16.46 / 3.52**

[1] DeepVoltaire/AutoAugment, https://github.com/DeepVoltaire/AutoAugment

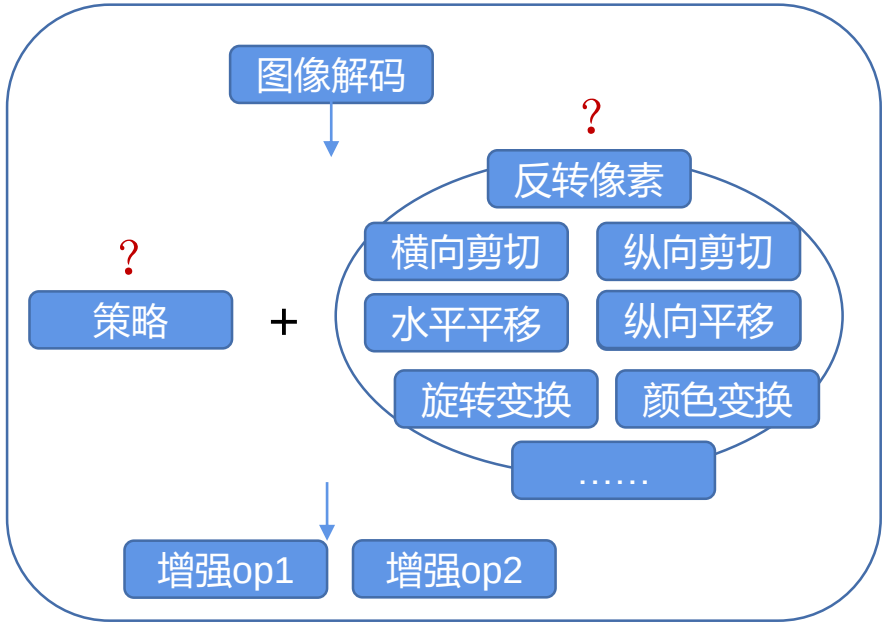
基于概率自动数据增强

在 ImageNet 上，Alex 等人在 2012 年引入的数据增强方法仍然是现在的标准操作，仅有微小的改变

针对 ImageNet 数据集，最终搜索出来的数据增强方案包含 25 个子策略组合，每个子策略中都包含两种变换，针对每幅图像都随机的挑选一个子策略组合，然后以一定的概率来决定是否执行子策略中的每种变换。



ResNet + ImageNet标准处理流程



AutoAugment处理流程

引用自：谷歌《一种自动搜索合适数据增强策略的方法》

基于概率数据增强 - 策略&算子

OP1	OP2
- [[7, 0.8, 1],	[1, 0.8, 4]]
- [[12, 0.4, 9],	[7, 0.6, 3]]
- [[12, 0.4, 1],	[4, 0.6, 8]]
- [[8, 0.8, 3],	[7, 0.4, 7]]
- [[8, 0.4, 2],	[8, 0.6, 2]]
- [[12, 0.2, 0],	[7, 0.8, 8]]
- [[7, 0.4, 8],	[27, 0.8, 3]]
- [[0, 0.2, 9],	[4, 0.6, 8]]
- [[12, 0.6, 1],	[7, 1.0, 2]]
- [[6, 0.4, 9],	[4, 0.6, 0]]
- [[7, 1.0, 9],	[1, 0.6, 3]]
- [[12, 0.4, 7],	[7, 0.6, 0]]
- [[9, 0.4, 6],	[5, 0.4, 7]]
- [[8, 0.6, 8],	[12, 0.6, 9]]
- [[8, 0.2, 4],	[4, 0.8, 9]]
- [[4, 1.0, 7],	[3, 0.8, 9]]
- [[0, 0.0, 0],	[8, 0.8, 4]]
- [[1, 0.8, 0],	[12, 0.6, 4]]
- [[12, 1.0, 0],	[4, 0.6, 2]]
- [[7, 0.8, 4],	[7, 0.0, 8]]
- [[7, 1.0, 4],	[5, 0.6, 2]]
- [[1, 0.4, 7],	[27, 0.6, 7]]
- [[9, 0.8, 2],	[8, 0.6, 10]]
- [[8, 0.6, 8],	[7, 0.6, 1]]
- [[12, 0.8, 6],	[4, 0.4, 5]]

算子ID，概率，参数

算子ID	算子名称	算子ID	算子名称	算子ID	算子名称
0	ShearXImpl	11	CutoutImpl	22	FlipLRImpl
1	ShearYImpl	12	ColorImpl	23	FlipUDImpl
2	TranslateXImpl	13	ConstrastImpl	24	BlurImpl
3	TranslateYImpl	14	BrightnessImpl	25	CropImpl
4	RotateImpl	15	SharpnessImpl	26	CutoutSpecific…
5	AutoContrastImpl	16	AdditiveGaussi…	27	SolarizeAddImpl
6	InvertImpl	17	Dropout		
7	EqualizeImpl	18	Multiply		
8	SolarizeImpl	19	CoarseDropout		
9	PosterizeImpl	20	GammaContrast		
10	SampleParingImpl	21	LinearContrast		

基于imgaug库

基于概率数据增强 - MindSpore算子

增强算子	MindSpore算子名称	描述
shearX	**RandomAffine**	横向剪切
shearY	**RandomAffine**	纵向剪切
translateX	**RandomAffine**	水平平移
translateY	**RandomAffine**	垂直平移
rotate	**RandomRotation**	旋转变换
color	**RandomColor**	颜色变换
posterize	**RandomPosterize**	减少颜色通道位数
solarize	**RandomSolarize**	指定的阈值范围内，反转所有的像素点
contrast	**RandomColorAdjust**	调整对比度
sharpness	**RandomSharpness**	调整锐度
brightness	**RandomColorAdjust**	调整亮度
autocontrast	**AutoContrast**	最大化图像对比度
equalize	**Equalize**	均衡图像直方图
invert	**Invert**	反转图像

MindSpore算子可以查看链接：https://www.mindspore.cn/api/zh-CN/master/api/python/mindspore/mindspore.dataset.vision.html

基于概率数据增强 - MindSpore策略

- **Random select sub policy：基于概率的策略选择算子，随机选择其中一个执行**

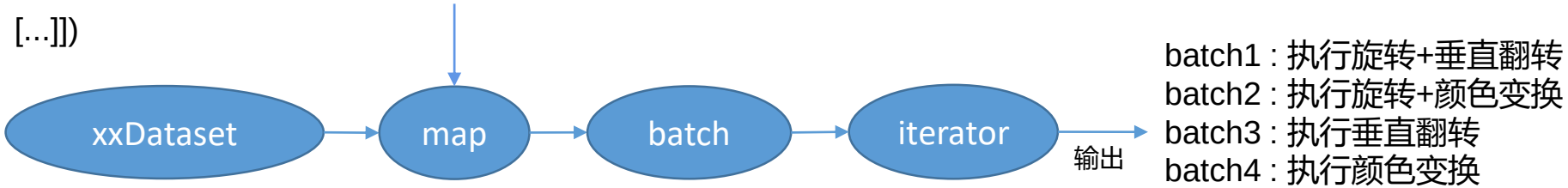
class mindspore.dataset.vision.c_transforms.RandomSelectSubpolicy(policy) [source]

Choose a random sub-policy from a list to be applied on the input image. A sub-policy is a list of tuples (op, prob), where op is a TensorOp operation and prob is the probability that this op will be applied. Once a sub-policy is selected, each op within the subpolicy with be applied in sequence according to its probability.

Parameters

policy (list(list(tuple(TensorOp,float))) – List of sub-policies to choose from.

```
op_list = C.RandomSelectSubpolicy([
    [(c_vision.RandomRotation((45, 45)), 0.5), (c_transforms.RandomVerticalFlip(), 1)],
    [(c_vision.RandomRotation((90, 90)), 1), (c_transforms.RandomColorAdjust(), 0.2)],
    [...]])
```



基于概率数据增强 - MindSpore策略

- **Random choice：多个算子选择其中一个执行**

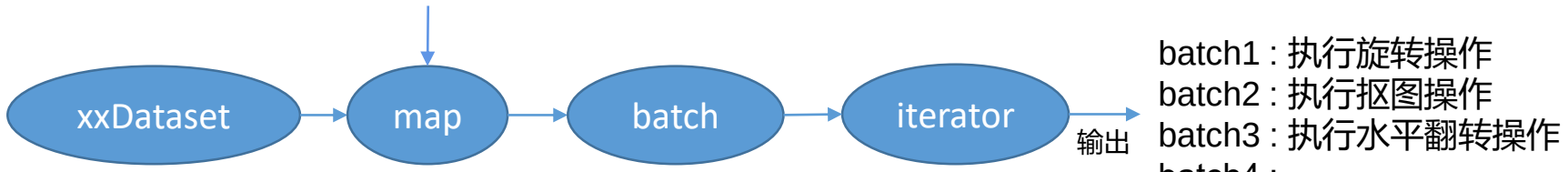
class `mindspore.dataset.vision.py_transforms.RandomChoice(transforms)` [source]

Randomly select one transform from a series of transforms and applies that on the image.

Parameters

transforms (list) – List of transformations to be chosen from to apply.

```
op_list = C.RandomChoice([C.RandomCrop((32, 32), (4, 4, 4, 4)),
                          C.RandomHorizontalFlip(prob=0.5),
                          C.RandomRotation((90, 90)])
```



batch1 : 执行旋转操作
batch2 : 执行抠图操作
batch3 : 执行水平翻转操作
batch4 : …

基于概率数据增强 - MindSpore策略

- **Random apply：基于某个概率执行这批算子**

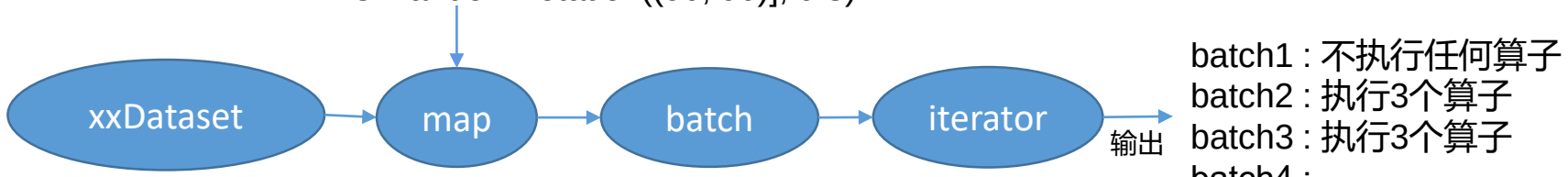
class `mindspore.dataset.vision.py_transforms.RandomApply(transforms, prob=0.5)` [source]

Randomly perform a series of transforms with a given probability.

Parameters

- **transforms** (list) – List of transformations to be applied.
- **prob** (float, optional) – The probability to apply the transformation list (default=0.5).

op_list = **C.RandomApply**([C.RandomCrop((32, 32), (4, 4, 4, 4)),
C.RandomHorizontalFlip(prob=0.5),
C.RandomRotation((90, 90)], **0.8**)



batch1 : 不执行任何算子
batch2 : 执行3个算子
batch3 : 执行3个算子
batch4 : …

基于概率数据增强 - MindSpore策略

- **Compose：多个算子合并执行**

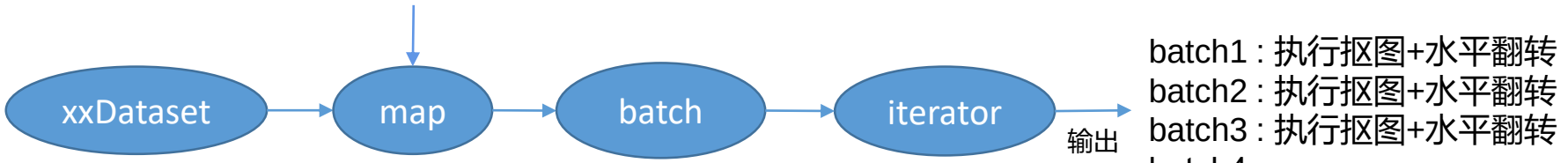
class mindspore.dataset.transforms.c_transforms.Compose(transforms) [source]

Compose a list of transforms into a single transform.

Parameters

transforms (list) – List of transformations to be applied.

```
op_list = C.Compose([C.RandomCrop((32, 32), (4, 4, 4, 4)),
                     C.RandomHorizontalFlip(prob=0.5)])
```



基于概率数据增强 - 代码演示

```
ds = de.ImageFolderDataset(data_dir, num_parallel_workers, True, …)
transform_img = [V_C.RandomCropDecodeResize(image_size, scale=(0.08, 1.0), ratio=(0.75, 1.333)),
                 V_C.RandomHorizontalFlip(prob=0.5)]
ds = ds.map(input_columns="image", operations=transform_img)
policies = […,
          [(C.RandomRotation(), 0.5), (C.RandomVerticalFlip(), 0.8)],
          [(C.RandomColorAdjust(), 0.2), (C.RandomRotation(), 1)],
          …]
ds = ds.map(input_columns=[ “image”], operations= C.RandomSelectSubpolicy(policies))
ds = ds.batch(batch_size, drop_remainder=True)
```

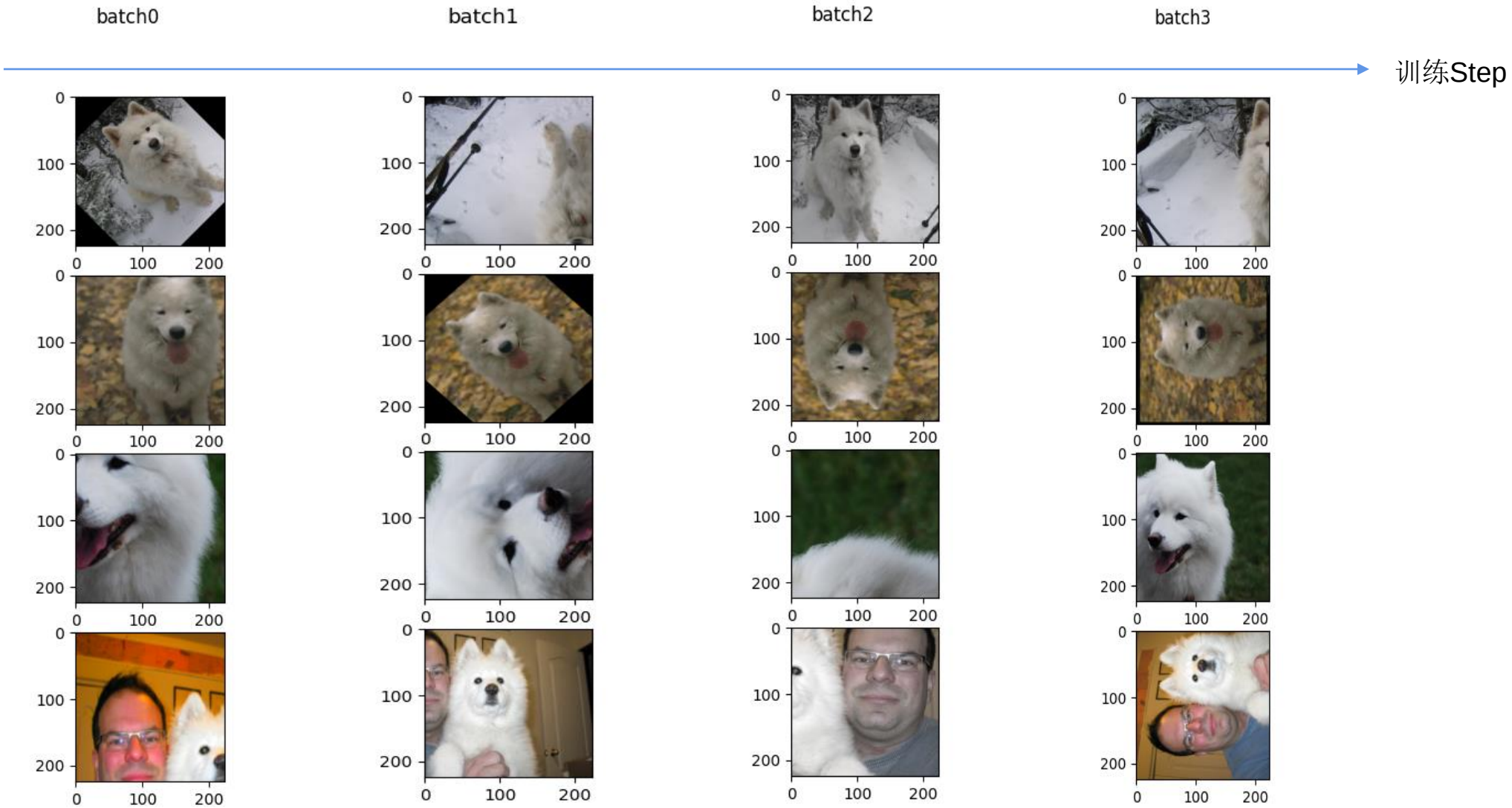
Step1: 加载数据集，解码、裁剪、翻转

Step2: 基于概率数据增强

Step3: Batch数据

[1]**应用自动数据增强,** https://www.mindspore.cn/tutorial/training/zh-CN/master/advanced_use/enable_auto_augmentation.html

基于概率数据增强 - 效果



基于概率数据增强 - 精度提升

- No augmentation

top1_acc=76.35%

```
[ mind_solver.py: 276][ INFO]  run in evaluate
[ mind_solver.py: 298][ INFO]  use automl dateset
[ mind_solver.py: 401][ INFO]  use automl network model
[ mind_solver.py: 688][ INFO]  use automl eval
260440896,MainProcess):2020-08-19-02:27:00.458.967 [mindspore/train/serialization.py:330] Remove parameter prefix name: network., continue to load
[ mind_solver.py: 708][ INFO]  img_tot 49984, self.config.data.batch_size 32, top1_correct 38161
[ mind_solver.py: 714][ INFO]  results=[[38161.]


[ mind_solver.py: 720][ INFO]  eval: top1_correct=38161.0, tot=49984.0, acc=76.35%
[ mind_solver.py: 721][ INFO]  eval: top5_correct=46557.0, tot=49984.0, acc=93.14%
```

- Auto data augmentation

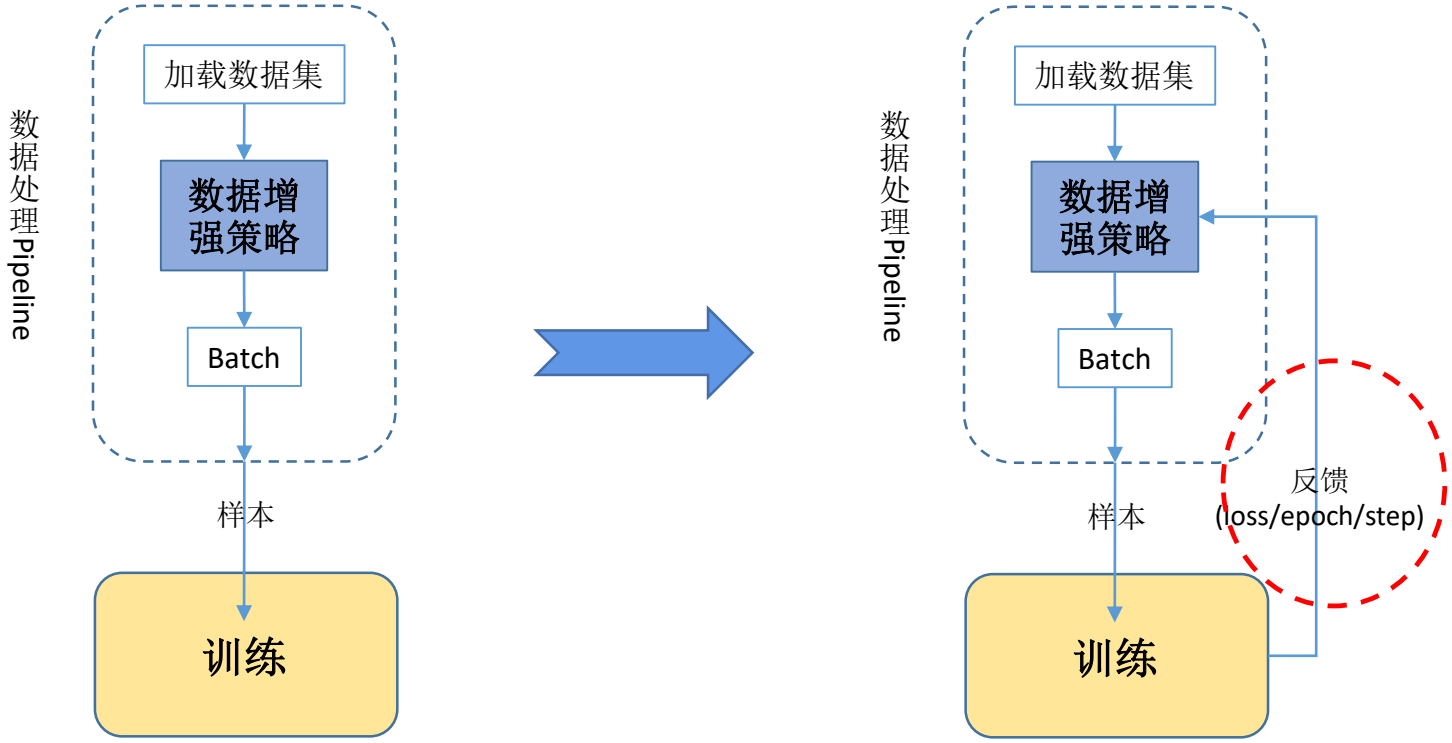
top1_acc=77.49%

```
[2020-08-29 01:03:17,564][ mind_solver.py][line: 266][    INFO] ms_output_dir: /home/lee/local_root
[2020-08-29 01:03:17,819][ mind_solver.py: 270][ INFO]  run in evaluate
[2020-08-29 01:03:17,819][ mind_solver.py: 292][ INFO]  use automl dateset
[2020-08-29 01:04:22,773][ mind_solver.py: 456][ INFO]  use mindspore zoo network model
[2020-08-29 01:04:25,953][ mind_solver.py: 764][ INFO]  use automl eval
[WARNING] ME(16662:139768010073920,MainProcess):2020-08-29-01:04:26.487.385 [mindspore/train/serialization.py:331] Remov
[2020-08-29 01:05:51,276][ mind_solver.py: 784][ INFO]  img_tot 49984, self.config.data.batch_size 32, top1_correct 387
[2020-08-29 01:05:51,332][ mind_solver.py: 790][ INFO]  results=[[38734.]
 [46854.]
 [49984.]]
[2020-08-29 01:05:51,332][ mind_solver.py: 796][ INFO]  eval: top1_correct=38734.0, tot=49984.0, acc=77.49%
[2020-08-29 01:05:51,332][ mind_solver.py: 797][ INFO]  eval: top5_correct=46854.0, tot=49984.0, acc=93.74%
```

演示视频

基于反馈的数据增强

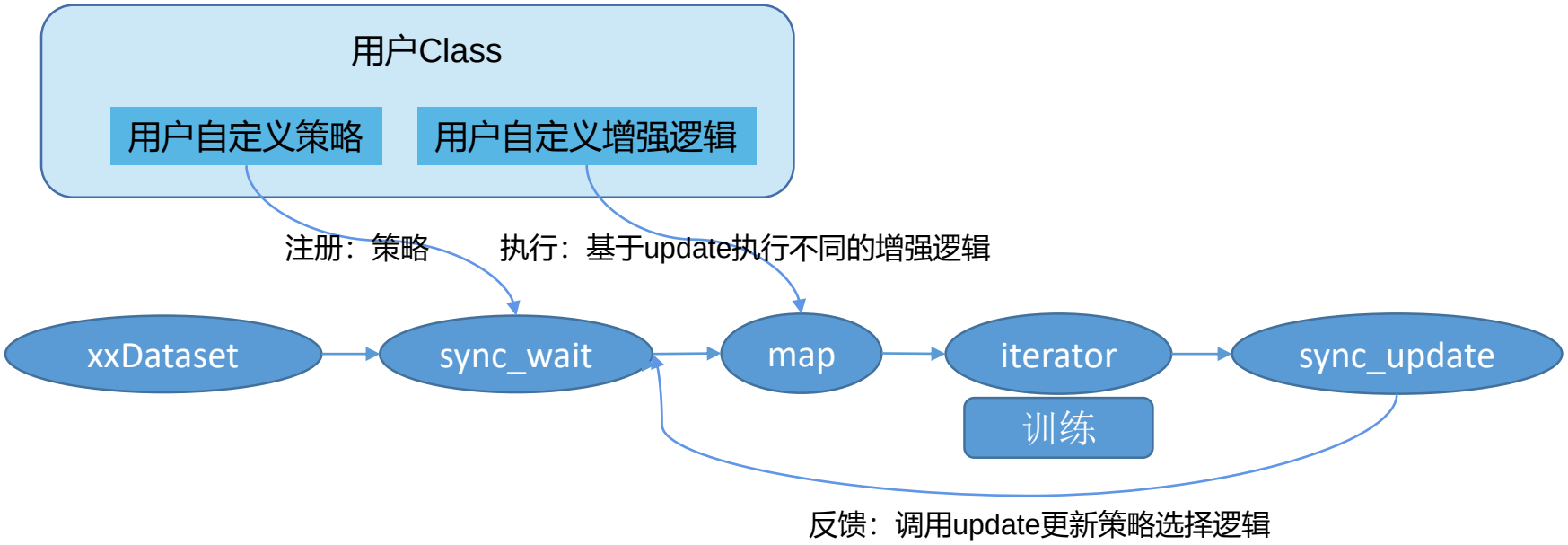
思考如何基于训练流程中的loss值/epoch数/step数，反过来影响数据增强过程，提供给用户更多操作数据的空间，进而提供更加丰富多样的数据用于训练。



基于反馈的数据增强

基于反馈的数据增强

通过dataset sync_wait & sync_update机制，实现用户自定义数据增强逻辑，用户可根据实时loss值/step数/epoch数来自动调整数据增强的逻辑，进而提高loss收敛效率，实现精度提升。



基于反馈的数据增强 - sync_wait & sync_update

基于反馈的数据增强 - sync_wait

- **sync_wait：注册用户自定义增强函数**

```
sync_wait(condition_name, num_batch=1, callback=None)
```

Add a blocking condition to the input Dataset.

Parameters

- **num_batch** (int) – the number of batches without blocking at the start of each epoch.
- **condition_name** (str) – The condition name that is used to toggle sending next row.
- **callback** (function) – The callback funciton that will be invoked when sync_update is called.

Raises

RuntimeError – If condition name already exists.

```
class Augment:
    def __init__(self, loss):
        self.loss = loss
    def preprocess(self, input_):
        return input_
    def update(self, data):
        self.loss = data["loss"]
…
aug = Augment(0)
dataset = dataset.sync_wait(condition_name="policy", callback=aug.update)
```

基于反馈的数据增强 - sync_update

- **sync_update：回调自定义数据增强策略**

```
sync_update(condition_name, num_batch=None, data=None)
```

Release a blocking condition and trigger callback with given data.

Parameters

- **condition_name** (str) – The condition name that is used to toggle sending next row.
- **num_batch** (Union[int, None]) – The number of batches(rows) that are released. When num_batch is None, it will default to the number specified by the sync_wait operator (default=None).
- **data** (Union[dict, None]) – The data passed to the callback (default=None).

```
…
for data in dataset.create_dict_iterator():
        loss += context.loss
        data = {“loss”: loss}          # 可以感知loss/epoch/step
        dataset.sync_update(condition_name="policy", data=data)
```

基于反馈的数据增强 - 示例

```
class AugmentTest:          # 用户自定义
    def __init__(self):
        self.loss = {}
    def preprocess(self, input_):
        return input_           # 基于self.loss实现自己的增强逻辑
    def update(self, data):
        # data: {"epoch": 2, "loss": 5.231}
        if data:
            if len(self.loss) == 10 and data['epoch'] not in self.loss:
                min_loss = 10000.0
                for key in self.loss:
                    if self.loss[key] < min_loss:
                        min_loss = self.loss[key]
                print("Every 10 epoch, the min loss is: {}".format(min_loss))
                self.loss = {}
            self.loss[data['epoch']] = data['loss']
```

```
ds = de.ImageFolderDataset(data_dir, num_parallel_workers, True, …)
ds = ds.map(…)
augment_test = AugmentTest()
ds = ds.sync_wait(condition_name="policy", callback=augment_test.update)
ds = ds.batch(…)

epoch = 0
for e in range(10):             # 总共训练10个epoch
    epoch += 1                  # 当前epoch数
    for data in ds.create_dict_iterator(num_epochs=-1):
        # 训练过程
        loss = context.loss     # 获取迭代loss值
        data = {“loss”: loss, “epoch”: epoch}
        ds.sync_update(condition_name="policy", data=data)
```

目录

contents

MindSpore

自动数据增强--总结

- 基于概率的自动数据增强
  - 通过提供多种算子支持，丰富数据增强操作，并借助于基于概率的RandomSelectSubpolicy实现AutoAugment，实现~1%左右的精度提升；
- 基于反馈的数据增强
  - 提供sync_wait & sync_update反馈机制，方便用户根据实时loss/epoch/step实现自己的数据增强逻辑，以达到更好的训练效果；
- 欢迎大家使用及参与
  - 提供丰富多样的数据集加载、增强类操作
  - 欢迎大家参与数据部分开发，sig/data主页包含资料，主要issue等：https://gitee.com/mindspore/community/tree/master/sigs/data

THANK YOU

MindData数据增强实现

定义

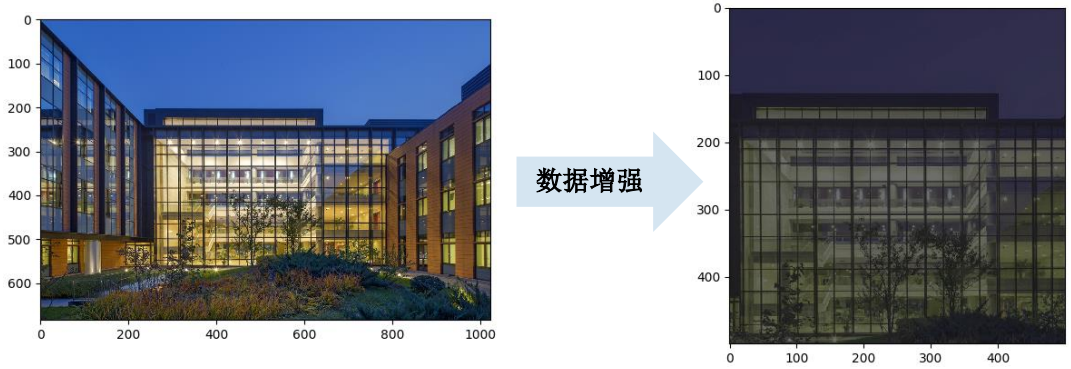
数据增强是一种**创造**有着不同方向的**“新”数据的方法**

目的

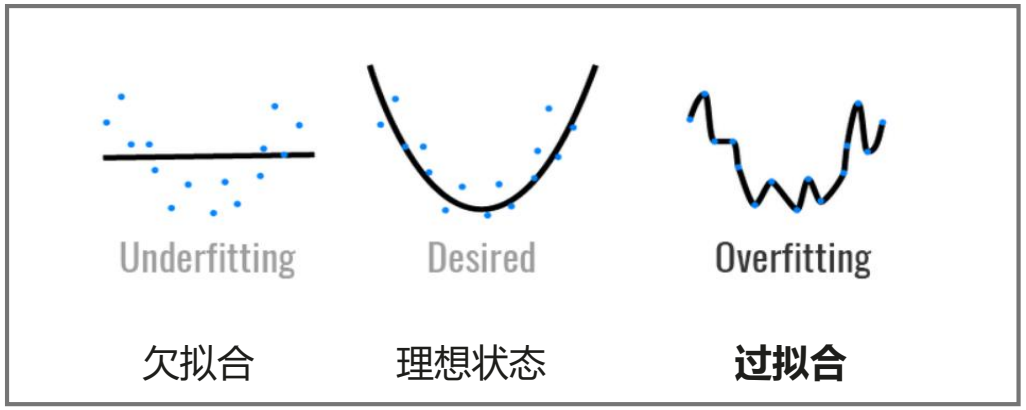
① 从有限数据中**生成“更多数据”**

② **防止过拟合**

当前只做了大部分的CV类数据增强算子实现。



例1 有限数据和复杂模型→过拟合



例2 数据不均衡影响模型效果