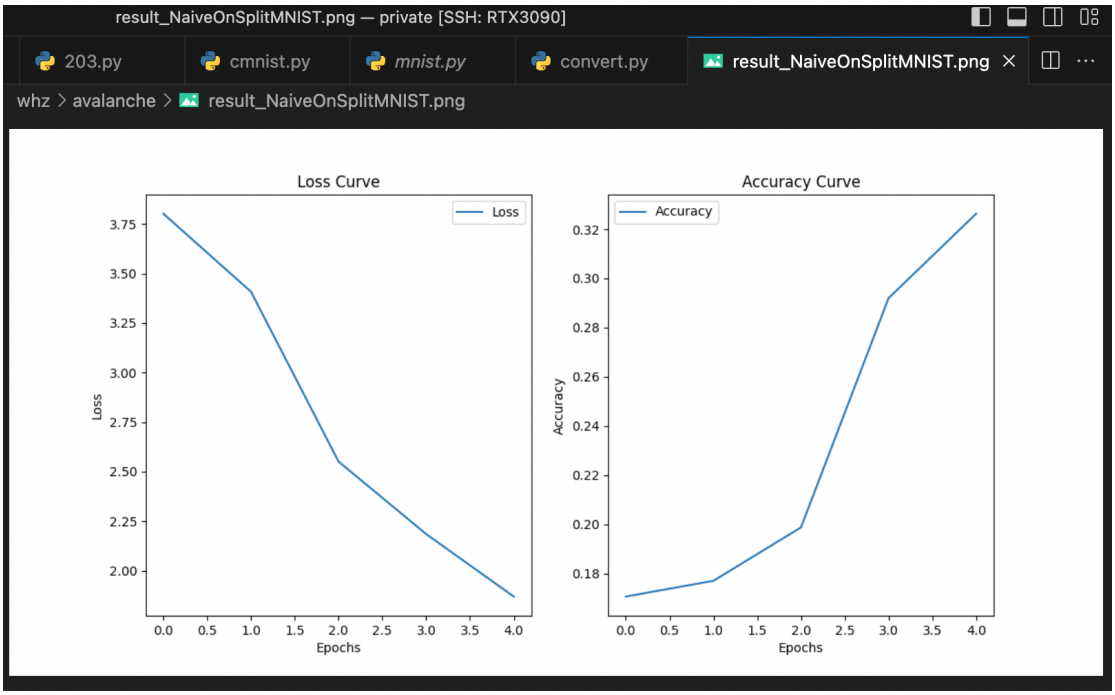

目前能画出来随着 experience 增加单调下降的 Loss 曲线和单调上升的 acc 曲线
策略是 Naïve，数据集是 SplitMNIST

周一开会时候说的每一次训练所抽取的样本，记样本编号，所有不同的策略。
经过我这几天的尝试，发现这一步实现比较难。
查看源代码发现因为 avalanche 在方便使用的同时采用 dataloader 封装了数据集。

```python
class TensorMNIST(MNIST):
    def __getitem__(self, index: int):
        """
        Args:
            index (int): Index

        Returns:
            tuple: (image, target) where target is index of the target class.
        """
        img = self.data[index].float().unsqueeze(0) / 255.
        target = int(self.targets[index])

        if self.transform is not None:
            img = self.transform(img)

        if self.target_transform is not None:
            target = self.target_transform(target)

        return img, target


def get_mnist_dataset(dataset_root):
    if dataset_root is None:
        dataset_root = default_dataset_location("mnist")

    train_set = TensorMNIST(root=str(dataset_root), train=True, download=True)

    test_set = TensorMNIST(root=str(dataset_root), train=False, download=True)

    return train_set, test_set
```

上图是 avalanche load MNIST 数据集的过程。

这具体来看，不是从图片文件直接读入数据的过程，而是先载入到内存。所以对于 cl 策略选择到的样本，

```
for i in range(strategy.mb_x.size(0)):  # Iterate through the batch
    x, y, task = strategy.mb_x[i], strategy.mb_y[i], strategy.mb_task_id[i]
    image = x.cpu().detach()  # Image is already a 3D tensor
    label = y.cpu().detach().item()
    task_id = task.cpu().detach().item()
```

里面每一个 batch 的数据只有 image（图像），label 以及对应的 task_id。

如果想要获得对应样本的原 filename，大致解决方案：
（一）遍历数据集，在对应的 label 下找对应图像的，匹配到了记录一下文件名，但是这个复杂度大概是$O(n^2)$
（二）重写 dataloader，如下图所示。主要是记录一下这个 sample indice 和文件路径之间的映射关系。

```
# Load the CIFAR-10 dataset into an ImageFolder
dataset = datasets.CIFAR10(root='./data', train=True, download=True

# Create a DataLoader to iterate over the dataset
dataloader = data.DataLoader(dataset, batch_size=32, shuffle=True)

# Get the mapping between sample indices and file paths
filepath_mapping = dataset.samples

# Iterate over batches in the DataLoader
for batch_idx, (images, targets) in enumerate(dataloader):
    # Iterate over samples in the current batch
    for sample_idx in range(images.shape[0]):
        # Get the file path of the sample with index sample_idx in
        sample_filepath = filepath_mapping[(batch_idx * dataloader.
        print(f'Sample with index {sample_idx} in batch {batch_idx}
```