進捗報告

1　今週やったこと

- NAS の実装

2　NAS

DARTS 系の NAS の実験をする. ネットワークにはまずエッジとセルが必要となる. コード 1, 2 に Pytorch で実装したクラスを示した. ネットワークの構造上逆伝播が可能が心配だったが, 逆伝播ができることも確認した. ただし期待通りの挙動かは分からないため, 実験をしながらテストしたい.

エッジに渡す演算子の多くは既に用意されているが, 存在しない零写像の仕様は逆伝播するときのボトルネックにならないようにテストする.

3　今後の予定

実装したクラスでネットワークを構築して実験する. 対象とする問題は Cifar-10 を考えている.

4　ソースコード

動作することを第一に実装した. operators で演算子の候補を受け取る. セルは 3 つのノードで 1 入力 1 出力と暫定的に定めた.

https://github.com/tatsuya-sugiyama/WeeklyReport/blob/2020_0717/report/2020_07_17/NAS_test.ipynb

Listing 1: Edge

```
class Edge(nn.Module):
  def __init__(self, operators):
    super(Edge, self).__init__()
    self.operators = operators

    rand = torch.randn(len(operators),
        requires_grad=True)
    self.theta = rand / torch.sum(rand)

  def forward(self, input: Tensor) ->
      Tensor:
    output = torch.zeros(input.shape,
        requires_grad=True)
    for (theta_i, operator) in zip(self.
        theta, self.operators):
      if operator == None:
        continue
      output = output + theta_i *
          operator(input)

    return output
```

Listing 2: Cell

```
class Cell(nn.Module):
  def __init__(self, operators):
    super(Cell, self).__init__()

    self.node_num = 3

    self.ref = [(0, 1), (0, 2), (1, 2)]
    self.edges = [Edge(operators) for _
        in self.ref]

  def forward(self, input) -> Tensor:
    nodes = [torch.zeros(*list(input.
        shape), requires_grad=True) for _
         in range(self.node_num)]
    nodes[0] = input

    for idx, (inref, outref) in enumerate
        (self.ref):
      nodes[outref] = nodes[outref] +
          self.edges[idx](nodes[inref])

    return nodes[-1]
```