Resnet 实现图像分类

1. 根据以上过程使用华为云计算平台实现 Resnet 并进行训练，并打印 train loss 和 test loss 随 Epoch 的变化。

Train loss 和 test loss 随 Epoch 的变化如下

```
================ epoch: 0 ================
Epoch: 0, Step 0, Loss: 4.730635166168213
Epoch: 0, Step 10, Loss: 3.9417178630828857
Epoch: 0, Step 20, Loss: 3.524763822555542
Test Accuracy: 9.78%

================ epoch: 1 ================
Epoch: 1, Step 0, Loss: 3.5577056407928467
Epoch: 1, Step 10, Loss: 3.248241424560547
Epoch: 1, Step 20, Loss: 2.9388880729675293
Test Accuracy: 20.7%

================ epoch: 2 ================
Epoch: 2, Step 0, Loss: 2.922029972076416
Epoch: 2, Step 10, Loss: 2.716867685317993
Epoch: 2, Step 20, Loss: 2.442538261413574
Test Accuracy: 31.25%

================ epoch: 3 ================
Epoch: 3, Step 0, Loss: 2.451678991317749
Epoch: 3, Step 10, Loss: 2.26495361328125
Epoch: 3, Step 20, Loss: 2.1220316886901855
Test Accuracy: 31.319999999999997%

================ epoch: 4 ================
Epoch: 4, Step 0, Loss: 2.10208797454834
Epoch: 4, Step 10, Loss: 1.9281272888183594
Epoch: 4, Step 20, Loss: 1.8231682777404785
Test Accuracy: 35.339999999999996%
```

```
================ epoch: 53 ================
Epoch: 53, Step 0, Loss: 0.00012660498032346368
Epoch: 53, Step 10, Loss: 0.00013054149167146534
Epoch: 53, Step 20, Loss: 0.0010247941827401519
Test Accuracy: 49.120000000000005%

================ epoch: 54 ================
Epoch: 54, Step 0, Loss: 0.00012247252743691206
Epoch: 54, Step 10, Loss: 0.00012570906255859882
Epoch: 54, Step 20, Loss: 0.0009695711196400225
Test Accuracy: 49.2%

================ epoch: 55 ================
Epoch: 55, Step 0, Loss: 0.00012051373778376728
Epoch: 55, Step 10, Loss: 0.0001242623693542555
Epoch: 55, Step 20, Loss: 0.0009941192111000419
Test Accuracy: 49.18%

================ epoch: 56 ================
Epoch: 56, Step 0, Loss: 0.00011589555651880801
Epoch: 56, Step 10, Loss: 0.00011857132631121203
Epoch: 56, Step 20, Loss: 0.0009313642513006926
Test Accuracy: 49.19%

================ epoch: 57 ================
Epoch: 57, Step 0, Loss: 0.00011213924153707922
Epoch: 57, Step 10, Loss: 0.00011491074110381305
Epoch: 57, Step 20, Loss: 0.0009983424097299576
Test Accuracy: 49.18%
```

```
================ epoch: 109 ================
Epoch: 109, Step 0, Loss: 4.1588933527236804e-05
Epoch: 109, Step 10, Loss: 4.3319378164596856e-05
Epoch: 109, Step 20, Loss: 0.0007726078038103878
Test Accuracy: 49.14%

================ epoch: 110 ================
Epoch: 110, Step 0, Loss: 4.13572124671191e-05
Epoch: 110, Step 10, Loss: 4.2967101762769744e-05
Epoch: 110, Step 20, Loss: 0.0007494341698475182
Test Accuracy: 49.27%

================ epoch: 111 ================
Epoch: 111, Step 0, Loss: 4.057879050378688e-05
Epoch: 111, Step 10, Loss: 4.216436718706973e-05
Epoch: 111, Step 20, Loss: 0.0007421904010698199
Test Accuracy: 49.16%

================ epoch: 112 ================
Epoch: 112, Step 0, Loss: 4.044344314024784e-05
Epoch: 112, Step 10, Loss: 4.2228857637383044e-05
Epoch: 112, Step 20, Loss: 0.000678455107845366
Test Accuracy: 49.24%

================ epoch: 113 ================
Epoch: 113, Step 0, Loss: 3.970618490711786e-05
Epoch: 113, Step 10, Loss: 4.153386180405505e-05
Epoch: 113, Step 20, Loss: 0.0007117451750673354
Test Accuracy: 48.949999999999996%
```

```
================ epoch: 207 ================
Epoch: 207, Step 0, Loss: 0.00011275532597210258
Epoch: 207, Step 10, Loss: 0.00011362785880919546
Epoch: 207, Step 20, Loss: 0.0008148489287123084
Test Accuracy: 48.75%

================ epoch: 208 ================
Epoch: 208, Step 0, Loss: 0.00010938670311588794
Epoch: 208, Step 10, Loss: 0.00010930907592410222
Epoch: 208, Step 20, Loss: 0.0008495545480400324
Test Accuracy: 48.72%

================ epoch: 209 ================
Epoch: 209, Step 0, Loss: 0.00010694810771383345
Epoch: 209, Step 10, Loss: 0.00010780262527987361
Epoch: 209, Step 20, Loss: 0.0007734188693575561
Test Accuracy: 48.83%

================ epoch: 210 ================
Epoch: 210, Step 0, Loss: 0.00010390315583208576
Epoch: 210, Step 10, Loss: 0.0001038362315739505
Epoch: 210, Step 20, Loss: 0.0008485581493005157
Test Accuracy: 48.77%

================ epoch: 211 ================
Epoch: 211, Step 0, Loss: 0.00010158854274777696
Epoch: 211, Step 10, Loss: 0.00010262909199809656
Epoch: 211, Step 20, Loss: 0.0007662546704523265
Test Accuracy: 48.85%

================ epoch: 316 ================
Epoch: 316, Step 0, Loss: 1.7784652300179005e-05
Epoch: 316, Step 10, Loss: 1.7739723261911422e-05
Epoch: 316, Step 20, Loss: 2.458603739796672e-05
Test Accuracy: 48.39%

================ epoch: 317 ================
Epoch: 317, Step 0, Loss: 1.738094761094544e-05
Epoch: 317, Step 10, Loss: 1.7351205315208063e-05
Epoch: 317, Step 20, Loss: 2.4130500605679117e-05
Test Accuracy: 48.39%

================ epoch: 318 ================
Epoch: 318, Step 0, Loss: 1.7019436199916527e-05
Epoch: 318, Step 10, Loss: 1.6980437067104504e-05
Epoch: 318, Step 20, Loss: 2.3453992980648763e-05
Test Accuracy: 48.339999999999996%

================ epoch: 319 ================
Epoch: 319, Step 0, Loss: 1.6632837287033908e-05
Epoch: 319, Step 10, Loss: 1.6609090380370617e-05
Epoch: 319, Step 20, Loss: 2.2884971258463338e-05
Test Accuracy: 48.339999999999996%
```

```
================ epoch: 401 ================
Epoch: 401, Step 0, Loss: 2.178302383981645e-05
Epoch: 401, Step 10, Loss: 2.4028255211305805e-05
Epoch: 401, Step 20, Loss: 4.6815708628855646e-05
Test Accuracy: 49.25%

================ epoch: 402 ================
Epoch: 402, Step 0, Loss: 2.1449408450280316e-05
Epoch: 402, Step 10, Loss: 2.369090543652419e-05
Epoch: 402, Step 20, Loss: 4.6903445763746276e-05
Test Accuracy: 49.27%

================ epoch: 403 ================
Epoch: 403, Step 0, Loss: 2.1104258848936297e-05
Epoch: 403, Step 10, Loss: 2.3283009795704857e-05
Epoch: 403, Step 20, Loss: 4.608129529515281e-05
Test Accuracy: 49.230000000000004%

================ epoch: 404 ================
Epoch: 404, Step 0, Loss: 2.060674705717247e-05
Epoch: 404, Step 10, Loss: 2.2735745005775243e-05
Epoch: 404, Step 20, Loss: 4.514198371907696e-05
Test Accuracy: 49.21%


================ epoch: 496 ================
Epoch: 496, Step 0, Loss: 4.141505996813066e-06
Epoch: 496, Step 10, Loss: 4.5985480028321035e-06
Epoch: 496, Step 20, Loss: 6.1100950006220955e-06
Test Accuracy: 48.99%

================ epoch: 497 ================
Epoch: 497, Step 0, Loss: 4.082833584106993e-06
Epoch: 497, Step 10, Loss: 4.533472747425549e-06
Epoch: 497, Step 20, Loss: 6.020619366609026e-06
Test Accuracy: 48.980000000000004%

================ epoch: 498 ================
Epoch: 498, Step 0, Loss: 4.022647772217169e-06
Epoch: 498, Step 10, Loss: 4.466535301617114e-06
Epoch: 498, Step 20, Loss: 5.929457074671518e-06
Test Accuracy: 48.96%

================ epoch: 499 ================
Epoch: 499, Step 0, Loss: 3.966943950217683e-06
Epoch: 499, Step 10, Loss: 4.404428182169795e-06
Epoch: 499, Step 20, Loss: 5.835685897181975e-06
Test Accuracy: 48.96%
```

2. 回答下列问题

（1）解释什么是 batch size

Batch size 是一次训练所选取的样本数，影响模型的优化程度和速度。

（2）什么是一个 epoch

一个 epoch 表示将所有数据送入网络中，完成一次前向计算+反向传播的过程。

（3）什么是过拟合，如何克服过拟合现象？

过拟合指训练的数据量不充足或是模型搭建有一定问题导致网络无法得到泛化特征，模型在训练集上误差很小，在测试集误差很大。

我们可以通过提升训练集的数据量或是简化模型，减少参数个数，从而避免出现过拟合现象。

（4）你认为 Resnet 引入的残差块为什么有效？

提出了拟合残差的思路而非直接拟合输出值，从而构造出了恒等映射，保证新添加的网络曾不会使效果比原来差。也避免了梯度消失的问题，可以训练几百甚至上千层的网络。

（5）有哪些常用的激活函数？分析这些激活函数的特点。

Sigmod 函数——输出在(0,1)之间，输出范围有限，优化稳定，可以用作输出层。连续函数，便于求导。但不是 0 均值的，导致后层的神经元的输入是非 0 均值的信号，会对梯度产生影响。

Tanh 函数——是 Sigmod 函数的变形，但相对于 Sigmod 函数，Tanh 函数是 0 均值的，实际应用中会比 Sigmod 好。

ReLU 函数——在 x＞0 的区域不会出现梯度饱和，梯度消失的问题，计算复杂度低的同时又带来了非线性性。但在 x＜0 的区域，梯度为 0，若某个神经元处在该区域，则之后的神经元梯度上于该神经元相关的误差传输的导数量为 0

Leaky ReLU 函数——ReLU 函数的变形，在 x＜0 区域，梯度为一个较小的常数，但不为 0，相应参数可以被更新。

Maxout 函数——函数拟合能力非常强，可以拟合任意的凸函数。具有线性，不饱和性。但是增加了参数量。

（6）常用的优化器 optimizer 有哪些？这些优化器各有什么特点？

GD——沿梯度方向不断减小模型参数，从而最小化 cost funtion。但训练速度慢且容易陷入局部最优解。

BGD——采用整个数据集的数据计算 cost function 对参数的梯度，导致在一次更新中，就对整个数据集计算梯度，计算非常慢。

SGD——相对于对整个数据集计算梯度，SGD 从样本中抽出一组进行更新，之后再抽出一组至损失值在可接受范围内。相对于 BGD，避免了对相似样本重复计算带来的冗余，提示效率。但是更新频繁，cost function

Momentum SGD——定义了速度，参数的更新不光取决于当前梯度，还保持了其惯性。

MBGD——每次利用一小批样本进行计算，降低参数更新时的方差，收敛更稳定，充分利用深度学习库中高度优化的矩阵操作进行更有效的梯度计算。但不能很好保证收敛性。

AdaGrad——是一种自适应学习率算法，独立地适应所有模型参数的学习率，缩放每个参数反比于其所有梯度历史平均值总和的平方根。具有代价函数最大梯度的参数相应地有个快速下降的学习率，而具有小梯度的参数在学习率上有相对较小的下降。但随着迭代次数增多，学习率会越来越小最终趋于 0。

3. 尝试对以上模型进行修改，以提高基于 CIFAR-100 数据集的分类精度。可以从一下角度进行：数据预处理（数据增强等）、改变模型结构、改变训练策略等。提供你的思路和源代码，用记录训练结果以证明你的方法有效。

尝试后未成功对模型以及数据进行改变，尝试改变超参数，但未能实现明显的优化效果。