浙江大学

本科实验报告

课程名称：人工智能与生命科学

姓 名：谢集

学 院：竺可桢学院

专 业：混合班

学 号：3220103501

指导教师：周如鸿

报告日期：2023 年 12 月 10 日

AttABseq 智能抗体筛选

一、实验背景.

在现代生物技术领域，蛋白质工程和抗体药物开发是研究的热点。蛋白质和抗体之间的相互作用对于疾病治疗和生物医学研究至关重要。因此在未来的生产中，如何快速而准确的筛选抗体，是当前研究的热点。

本次实验通过数据集 $\mathrm{AB\text{-}Bind}$，使用 `CNN`+`transformer` 的架构来进行智能抗体筛选的研究。

二、实验目的.

1、通过深度学习，使用 AI 来预测氨基酸突变对抗体和蛋白质的结合自由能（$\mathrm{binding\ free\ energy}$）的影响。

2、完成模型的训练和相关结果参数的图表绘制。

3、通过 `transformer` 的训练结果，找出什么样的氨基酸位置是关键的，并给出相关分析。

三、实验方法和实验材料.

3.0 实验环境.

> - 操作系统：Windows 2019.
> - 电脑配置：8 核，26GB 内存.
> - 显卡配置：TeslaT4，16GB 显存.
> - cuda 版本：11.2
> - python 版本：3.6

3.1 数据集.

本次实验用到的数据集是 $\mathrm{AB\text{-}Bind}$，这是一个全面的抗体结合突变数据库，其中包括一系列不同的抗体结合数据和结构，可用于抗体相互作用的评估方法的建立。$\mathrm{AB\text{-}Bind}$ 数据库包括 1101 个突变体，这些突变体的结合自由能变化（$\mathrm{dd}G$）是通过实验确定的，涉及了 32 个复合物。其公式如下：

$$\mathrm{dd}G = \mathrm{d}G_{\mathrm{mutant}} - \mathrm{d}G_{\mathrm{wet}}(\mathrm{kcal/mol})$$

其中，$\mathrm{d}G_{\mathrm{mutant}}$ 是指突变型蛋白质的自由能，而 $\mathrm{d}G_{\mathrm{wet}}$ 是指野生型蛋白质的自由能。根据自由能的定义，如果 $\mathrm{dd}G$ 为正，则表明突变降低了蛋白质的稳定性；如果为负，则表明突变增加了蛋白质的稳定性。

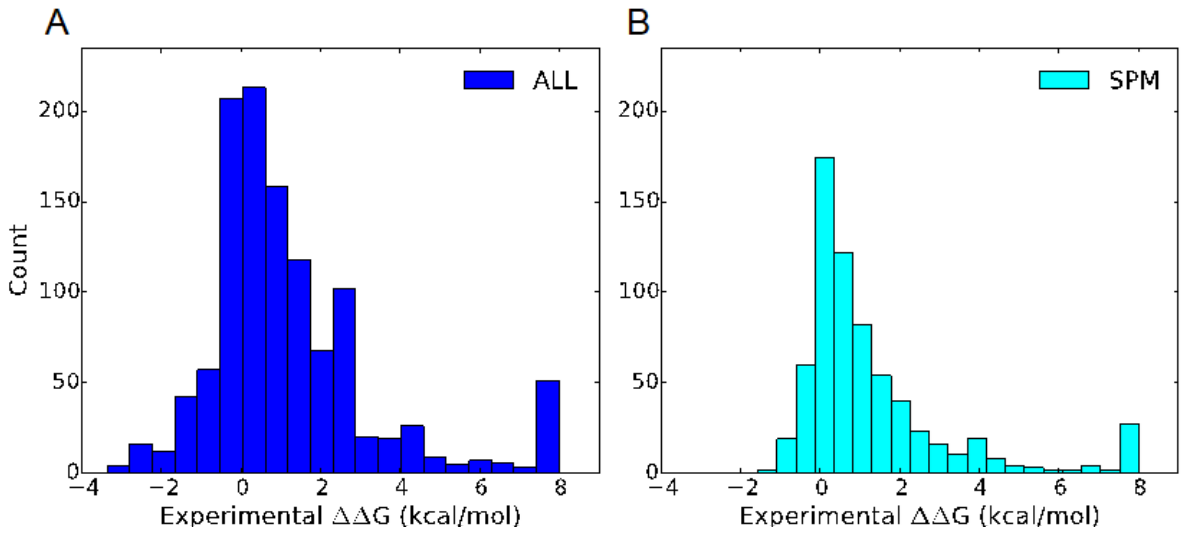

图 1. 经过突变后的结合自由能的变化。第一个直方图展示所有突变的数据，第二个直方图展示单点突变（SPMs）的数据。

3.2 蛋白质信息的存储和表示.

本实验通过位置特异性评分矩阵（PSSM）来研究存储、表示蛋白质信息，并研究这些突变。PSSM 是一种生物信息学工具，又称位置特异性权重矩阵（PSWM）或位置权重矩阵（PWM），是生物序列的常用表示方法。

PSSM 的每一行代表字母表中的每个符号（DNA 序列中的核苷酸为 4 行，蛋白质序列中的氨基酸为 20 行），每一列代表模式中的每个位置。用于表示特定位置氨基酸的出现频率。如图：

```
A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
[[-2 -2 -3 -4 -2 -1 -3 -4 -2  0  1 -2  8 -1 -3 -2 -2 -2 -2  0]
 [ 2  1 -1 -2 -2  0  0  3 -2 -2 -2 -1 -2 -3 -2  0  0 -3 -2 -1]
 [ 3 -2 -2 -3 -1 -2 -2 -2 -3  1  0 -2  0 -2 -2  0  1 -3 -2  3]
 [ 2 -2 -2 -3 -2 -1 -2 -2 -2  0  0 -2  5 -2  3 -1  0 -3 -2  0]
 [ 4  1 -2 -3 -1 -1 -2 -1 -2 -1  0 -1  0 -2 -2  0 -1 -3 -2  0]
 [-1 -3 -2 -2 -3 -2 -2 -3 -3 -4 -4 -2 -3 -4  8  2 -1 -5 -4 -3]
 [ 0  4 -2 -2  1  1 -1 -1 -2 -3 -3  0 -2 -3  3 -1 -2  5 -2 -3]
 [ 2 -2 -2 -3 -2 -2 -2 -2 -3 -1  2 -2  2 -2  1  0  3 -3 -2  0]
 [-2 -3 -4 -4 -2 -3 -3 -4 -3  1  4 -3  1  2 -3 -3 -2  4 -1  1]
 [-1 -3 -4 -4 -2 -3 -4 -4 -3  1  4 -3  1  4 -4 -3 -2 -2  0  0]]
```

图 2. PSSM 实例

通过分析一个蛋白质序列的 PSSM，可以了解哪些位置的氨基酸突变更可能影响蛋白质的功能或结构，从而预测这些变化如何影响抗原和抗体之间的亲和力。

3.3 数据预处理，PSSM 特征矩阵生成.

PSSM 矩阵的生成需要用到蛋白质处理软件 `PSI-BLAST`。`BLAST` 是一个基本局部比对搜索工具，可将输入的蛋白质序列与数据库中的已知序列进行比较，以获得相关序列的相似性和其他信息，从而确定序列的来源或进化情况；而 `PSI-BLAST` 是其基础上的 PSSM 生成工具。它每次使用已有的 PSSM 搜索数据库后会用搜索结果重建 PSSM，然后使用新的 PSSM 再次搜索数据库，直到没有新结果产生为

止。在本次实验中，我们的下载文件里包括了这个软件。

我们先下载实验所需的文件 `AttABseq`，将 AB-Bind 数据集（`AB1001order.csv`）复制到根目录下的 `data` 文件夹，然后打开 `script` 文件夹，用命令行运行代码：

```
python data_preprocessing.py C:/Users/Administrator/Desktop/AttABseq/
```

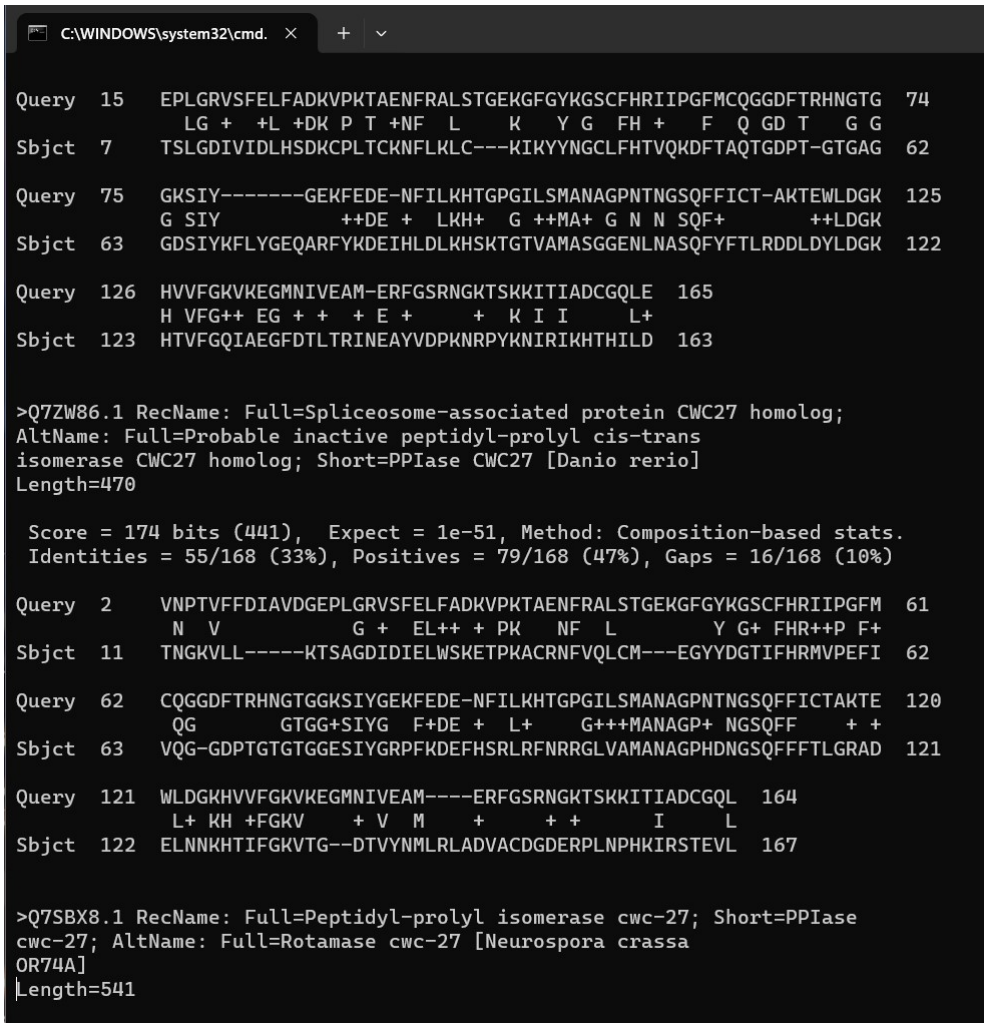

图 3. preprocessing 预处理过程图

等待几个小时后 `data` 文件夹下产生 4 个 `numpy` 文件作为预处理结果，其分别是 `antibody.npy`，`antibody_mut.npy`，`antigen.npy`，`antigen_mut.npy`。

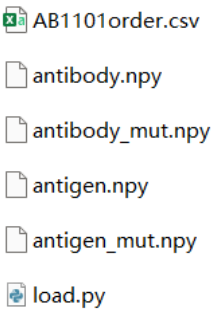

图 4. npy 文件 **3.4 模型训练.**

将 `main.py` 中 `iteration` 变量改为 `15`，`batch_size` 改成 `6`，然后使用命令行运行以下代码以开启训练：

```
python main.py
```

但是出现以下问题：

```
The code uses GPU...
Training...
Epoch   Time(sec)       Loss_train      Loss_val        pearson mae     mse     rmse    r2
Epoch: 1
C:\Users\Administrator\Desktop\AttABseq\AttABseq\script\Radam.py:53: UserWarning: This overload of addcmul_ is deprecated:
        addcmul_(Number value, Tensor tensor1, Tensor tensor2)
Consider using one of the following signatures instead:
        addcmul_(Tensor tensor1, Tensor tensor2, *, Number value) (Triggered internally at  ..\torch\csrc\utils\python_arg_parser.cpp:882.)
  exp_avg_sq.mul_(beta2).addcmul_(1 - beta2, grad, grad)
Traceback (most recent call last):
  File "main.py", line 145, in <module>
    pccs_val, mae_val, mse_val, rmse_val, r2_val, loss_val_fold, y_val_true, y_val_predict = tester.test(dataset_val, epoch)  # pccs_dev && loss_val_fold are for an epoch
  File "C:\Users\Administrator\Desktop\AttABseq\AttABseq\script\model.py", line 544, in test
    T = torch.cat([T,correct], dim=-1)
RuntimeError: All input tensors must be on the same device. Received cpu and cuda:0

(pl) C:\Users\Administrator\Desktop\AttABseq\AttABseq\script>
```

图 5. 训练报错

这是因为 `T` 和 `correct` 不在一个设备上。使用 `.to` 函数修改代码：

```
532         with torch.no_grad():
533             for data in dataset:
534                 ag_s, ab_s, ag_m_s, ab_m_s, labels = [], [], [], [], []
535                 ag, ab, agm, abm, label = data
536                 ag_s.append(ag)
537                 ab_s.append(ab)
538                 ag_m_s.append(agm)
539                 ab_m_s.append(abm)
540                 labels.append(label)
541                 data = pack(ag_s, ab_s, ag_m_s, ab_m_s, labels, self.model.device)
542                 correct, predicted = self.model(data, itera, train=False)
543                 correct = correct.view(1,-1)
544                 predicted = predicted.view(1,-1)
545                 T = torch.cat([T.to(correct.device),correct], dim=-1)
546                 Y = torch.cat([Y.to(correct.device),predicted], dim=-1)
547         T = T.squeeze()
548         Y = Y.squeeze()
549         T=T.to(torch.device('cpu'))
550         Y=Y.to(torch.device('cpu'))
```

图 6. 修改代码

就可以开始训练了。大约三个小时候训练结束，生成六个文件夹：

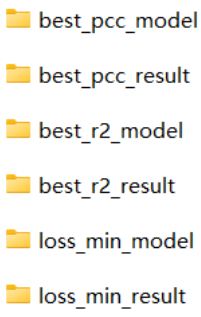

图 7. 模型训练结果

四、实验数据记录和处理.

1、数据获取和蛋白质序列的 PSSM 特征生成.

在 `script` 产生的 PSSM 文件如图：

```
Last position-specific scoring matrix computed, weighted observed percentages rounded down, information per position, and relative weight of gapless real matches to pseudocounts
            A   R   N   D   C   Q   E   G   H   I   L   K   M   F   P   S   T   W   Y   V   A   R   N   D   C   Q   E   G   H   I   L   K   M   F   P   S   T   W   Y   V
    1 D    -4  -4   3   6  -7  -2   5  -5  -4  -6  -6  -3  -6  -7  -5  -3  -4  -7  -6  -6    0   0  12  42   0   0  46   0   0   0   0   0   0   0   0   0   0   0   0   0  1.64 0.66
    2 I     1  -4  -3  -5  -3  -3  -4  -3  -5   4   2  -4  -1  -1  -3  -2   1  -5  -3   3   13   0   1   0   0   1   0   1   0  27  17   0   1   2   1   3   8   0   0  26  0.59 0.36
    3 V    -3  -3  -5  -2  -6   6  -1  -1  -5  -1  -4   1  -2  -6  -5  -3  -2  -2  -3   5    1   1   0   2   0  34   4   6   0   2   1  10   1   0   0   1   2   1   1  36  1.01 0.64
    4 L    -5  -7  -7  -7  -5  -6  -7  -8  -7   2   4  -6   4  -1  -7  -4  -3  -6  -5   5    0   0   0   0   0   0   0   0   0   9  34   0   8   2   0   1   1   0   0  45  1.39 0.70
    5 T    -3  -3  -3  -5  -5   1   1  -5  -3   0  -5  -3  -2  -6  -3   0   6  -6  -4   1    1   1   1   0   0   7   8   0   1   5   0   1   1   0   1   8  54   0   0  10  1.13 0.70
    6 Q    -6  -2  -4  -5  -8   8   0  -7   0  -3  -5  -2  -3  -4  -4  -4  -5  -8  -4  -4    0   2   0   0   0  76   6   0   2   2   1   2   0   1   1   1   1   0   1   2  2.32 1.18
    7 S    -1  -2   0  -1  -5   0  -1  -3   0  -4  -3  -3  -2  -3   3   4   3  -2  -5  -2    4   2   3   4   0   4   4   2   2   1   2   1   1   1  16  31  18   1   0   2  0.62 0.48
    8 P    -5  -6  -5  -3  -8  -1   1   0  -5  -5  -5  -3  -7  -8   8  -2  -3  -4  -8  -5    1   0   0   2   0   3  10   7   0   1   1   1   0   0  67   3   2   0   0   1  1.99 1.02
    9 A     1   2  -1  -2  -4   2   0   1  -1  -2  -2   1  -2  -3   1   2   0  -5  -4  -2   12  11   2   2   0  10   5   9   1   2   4   9   1   1   8  15   6   0   0   2  0.25 0.24
   10 T    -1  -2   0   0  -3  -2   0  -1   3  -2  -1  -1  -4   2   0   3   0  -4   1  -2    4   2   4   6   0   2   6   4   7   2   6   3   0   9   6  26   6   0   5   3  0.28 0.28
   11 L    -1  -3  -5  -5  -5   1  -1  -5  -2   1   3  -1   1  -3  -6  -3  -2  -6  -4   4    5   1   0   1   0   8   4   1   1   6  31   5   3   1   0   2   3   0   0  30  0.70 0.45
   12 S    -1  -1  -1   0  -3  -1   0  -4  -2   1  -1  -1  -1  -3  -1   3   3  -3  -3   2    3   3   3   5   1   3   7   1   1   9   5   3   2   1   2  21  17   0   1  13  0.29 0.27
   13 V     2   0  -6  -7  -2  -5  -4  -2  -3   1   0   0  -2  -4  -4  -3   0  -7  -3   5   15   5   0   0   1   0   1   3   1   6   7   6   1   1   1   2   6   0   1  45  0.83 0.53
   14 T     0   0  -2  -4  -5   3   0  -2  -2  -2   0   2  -1   0   0   1   1  -1  -2   0    8   4   2   0   0  15   6   3   1   2   8  12   2   4   5  12   9   1   1   6  0.21 0.25
   15 P    -1   0  -4  -3  -6  -1   3  -6  -5  -2   1  -1  -1  -6   4  -2   1  -2  -6   2    5   6   1   1   0   2  21   0   0   2  16   4   2   0  19   2   7   1   0  12  0.54 0.43
   16 G    -4  -4  -1  -4  -8  -8  -4   7  -5  -9  -7  -4  -8  -9  -8  -2  -2  -9  -9  -9    1   1   3   1   0   0   1  83   0   0   1   1   0   0   0   3   3   0   0   0  2.29 1.10
   17 D     0   0  -1   2  -6   4   3   2  -3  -6  -4   1  -3  -5  -5   1  -2  -7  -6  -5    9   4   3  10   0  18  17  13   1   0   2   9   1   0   0   9   2   0   0   0  0.57 0.44
   18 S    -2   1   1   1  -1   0  -1  -6  -3  -3  -5  -1  -1  -4   0   4   4  -7  -6  -3    2   6   6   7   1   3   5   0   1   1   1   4   2   1   5  32  21   0   0   2  0.66 0.52
   19 V     4  -6  -7  -8  -3  -7  -7  -6  -5   1   1  -4   0  -3  -6  -4  -2  -8  -5   5   28   0   0   0   1   0   0   0   0   6  13   1   2   1   0   1   2   0   0  43  1.20 0.69
   20 S    -2   2  -1  -4  -6  -2   0  -6  -1   0  -2   2   0  -3  -6   1   5  -7  -3   0    3   9   3   0   0   2   6   0   2   6   5  10   2   1   0  11  32   0   1   7  0.58 0.46
   21 L    -7  -8  -9 -10  -7  -8  -9 -10  -9   4   6  -8   3   2  -9  -8  -6  -8  -7  -1    0   0   0   0   0   0   0   0   0  18  64   0   7   7   0   0   0   0   0   4  1.82 0.90
   22 S    -3   1   1  -1  -6  -1   0  -5   0  -4  -3   0  -4  -5   1   3   4   0  -2  -4    1   8   7   3   0   3   5   1   2   1   3   6   0   0   8  24  24   1   2   1  0.59 0.49
   23 C    -8 -11 -11 -12  12 -11 -12  -8 -11  -6  -6 -11  -9 -10 -11  -6  -9  -3  -6  -5    0   0   0   0  96   0   0   0   0   0   1   0   0   0   0   1   0   1   0   1  4.83 1.90
   24 R     1   2   0   0  -5   1   1  -2  -1  -2  -2   1  -3  -3  -3   1   2  -4  -3  -1   10  11   5   4   0   6  10   3   1   2   4   9   0   1   1  11  16   0   1   4  0.25 0.26
   25 A     3  -6  -5  -5  -5  -1  -5  -2  -1  -1   1  -5  -2   1   1  -1  -1  -5   4   3   25   0   0   1   0   3   0   4   1   2  11   0   1   6   6   4   4   0  12  19  0.54 0.42
   26 S    -2  -2   1   0  -5   0   0  -4  -1   0  -4  -1  -3  -3  -2   5   2  -7  -5  -3    3   2   6   6   0   3   5   1   2   5   2   4   1   1   2  45  11   0   0   2  0.78 0.58
   27 Q    -2  -2   1  -1  -5   2  -1   4  -1  -2  -2  -1   0  -5  -2   2   1  -3  -2  -1    3   2   6   3   0   8   4  28   1   2   3   4   3   0   2  13   9   0   2   4  0.39 0.33
   28 S    -1  -3   2   1  -2  -2  -1   0   0  -3  -1  -3  -2   3  -1   3   1  -5  -1  -3    6   1   8   7   1   2   3   6   2   1   7   2   1  11   4  25  10   0   2   1  0.34 0.32
   29 I    -1  -2  -1  -1  -5  -1  -1   1   0   1  -1  -2  -3  -4   4   0   2  -3   0  -1    5   2   3   4   0   3   3  11   2   7   5   3   0   0  23   8  11   0   4   5  0.35 0.30
   30 S    -2  -2   1   0  -4  -1   0   0   4  -2  -3  -1  -4   2   1   3   0  -3   0  -1    3   2   7   5   0   2   5   8   9   3   2   4   0   8   7  21   6   0   3   5  0.27 0.27
   31 N    -1  -3   2   0  -6  -1  -2   0   1  -3  -2  -3  -2  -3   4   2   0  -6   3  -2    4   1   9   6   0   2   2   6   3   1   5   1   1   1  20  16   6   0  10   4  0.43 0.38
   32 N    -1   1   2   0  -5  -2   0  -2   1  -2  -2  -3  -3   1  -2   1   1   2   5  -2    5   7  10   5   0   2   6   3   4   2   3   1   0   5   3  13   9   2  18   3  0.36 0.31
   33 L    -4  -5  -7  -4  -1  -6  -5  -4  -7   5   3  -7   5   0  -4  -6  -5   0  -3   4    1   1   0   1   1   0   1   2   0  25  24   0  13   3   1   0   1   1   1  23  1.11 0.60
   34 H     0  -1   1  -2  -4   2  -1  -2   3  -1  -2  -2  -2   2  -3   2   1  -3   3  -3    6   3   5   2   0  10   5   2   8   4   3   2   1   8   1  17   8   0  12   1  0.35 0.32
   35 W    -8  -7  -6  -9  -8  -6  -5  -7  -4  -8  -6  -8  -8  -7  -9  -5  -4  13  -6  -5    0   0   1   0   0   0   1   1   0   0   1   0   0   0   0   1   1  90   0   1  4.91 2.01
   36 Y    -2  -2  -5  -5  -3  -3  -3  -5  -1  -1   0  -2  -1   3  -5  -3   0   2   7   2    3   3   1   1   1   1   2   1   1   4   8   3   1  12   1   2   6   2  35  14  1.00 0.52
   37 Q    -3   5  -2  -3  -5   4  -2  -3  -2  -3  -1   3  -4  -2  -4  -2  -2  -2  -2  -1    2  34   2   1   0  16   2   3   1   1   6  14   0   2   1   2   3   1   2   5  0.83 0.48
```

图 8. pssm 文件部分内容截图

2、抗体优化模型训练和模型测试.

训练过程中，我多次发现在第二个 epoch 时，`predict` 的值都是 0，然后接下来的训练里，它都卡在 0 处没有动弹。经过了 15 轮的迭代后它有所变化，但 loss 还是非常之高。

```
predict: tensor([1.0092, 1.0066, 0.9879, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.000
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        1.4533, 1.5108, 1.4785, 1.4926, 1.4366, 1.5514, 1.3691, 1.5241, 1.1877,
        1.1894, 0.0000, 0.0000, 0.7900, 0.7748, 0.7725, 0.7749, 0.7744, 0.7904,
        1.2231, 1.2366, 1.2236, 1.2190, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 2.5828, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.8711, 1.8803, 1.8777,
        1.8689, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000])
tensor(-0.0475) 3.3342385 17.508463 4.184311521132693 -1.7827707079128938
Validation loss decreased (17.812300 --> 17.508465).  Saving model ...
15      8263.4707351    2.799717426300049       17.508464813232422      -0.04752319306
```

图 9. 训练结束后，predict 基本都还是 0

我经过多次 `debug` 无果后，开始尝试调整 learning rate，最终把它缩小十倍后，很好的解决了这个问题，在第三轮的时候 loss 就小了很多。

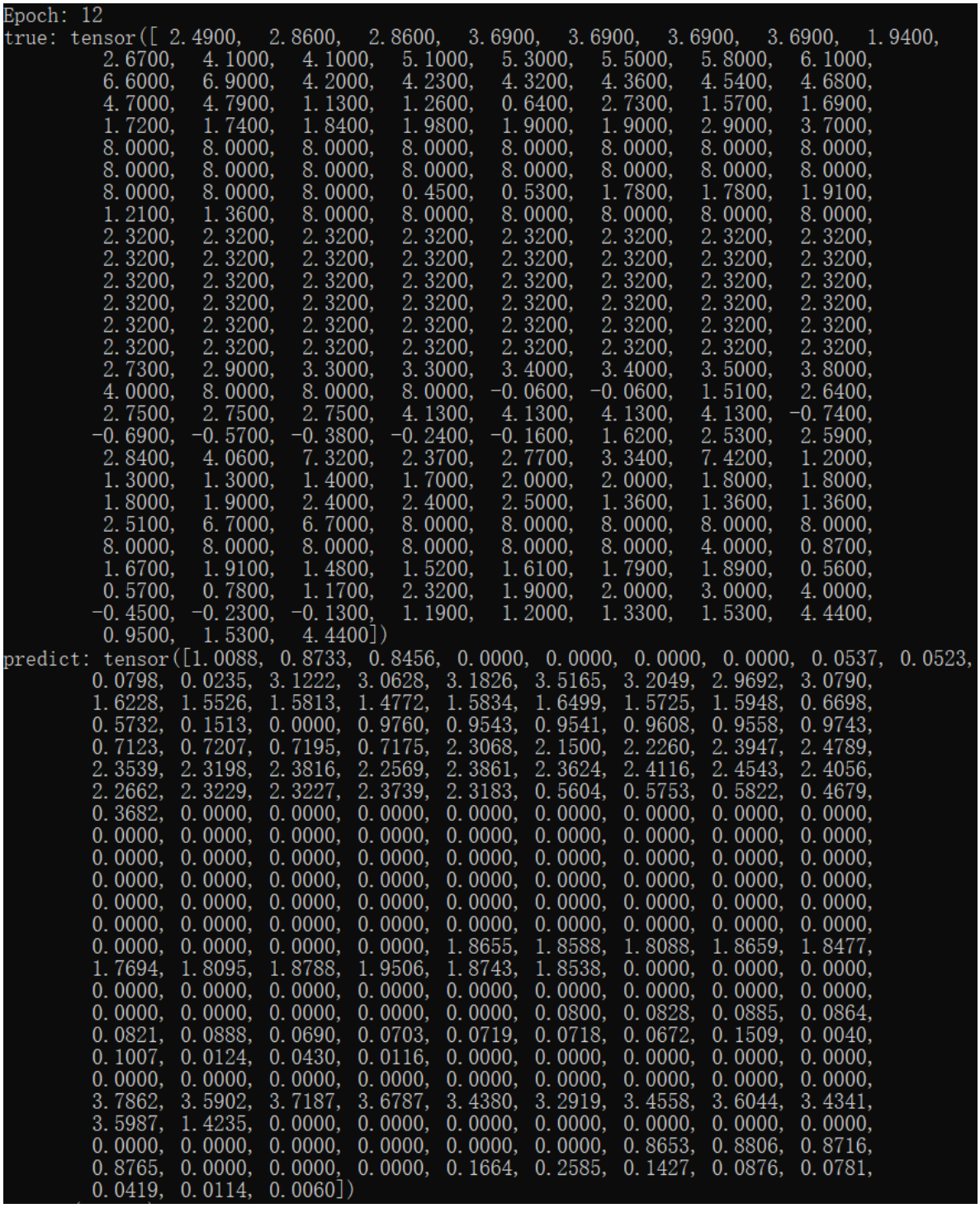

图 10. 正常训练

最终训练结果表格如下（保留了 Loss 更优的）：

当您想要在文本中为数字添加数学公式时，您可以使用两个美元符号（$$）将数字括起来，以指示这是一个数学公式。以下是带有数学公式的表格：

Epoch	Time(sec)	Loss_train	Loss_val	Pearson	R2
1	473.7687003	2.7946441173553	18.0148220062255	0.168525502	−1.8632504
2	956.9852347	2.7945592403411	17.7121467590332	0.414589226	−1.8151434
3	1450.539040	2.7590229511260	15.6290102005004	0.551896452	−1.4840527
4	1953.859212	2.7219347953796	15.2135953903198	0.600119292	−1.4180271
5	2463.576936	2.6044981479644	14.6178426742553	0.578223168	−1.3233392
6	2976.563974	2.5839071273803	13.2556400299072	0.595439076	−1.1068327

Epoch	Time(sec)	Loss_train	Loss_val	Pearson	R2
7	3492.522385	2.4024815559387	12.6484003067016	0.636978745	−1.0103190
8	4013.974228	2.2620975971221	12.0521955490112	0.652012407	−0.9155591
9	4538.806935	2.2232568264007	11.7755603790283	0.678327500	−0.8715908
11	5608.104569	2.2055249214172	11.5423021316528	0.684090971	−0.8345171
12	6145.399454	2.0732476711273	11.1424255371093	0.709782063	−0.7709613
13	6683.613938	1.8927793502807	11.0604333877563	0.700327634	−0.7579297

表 1. 训练结果(其中第 10、14、15 次训练没有获得更小的 loss)

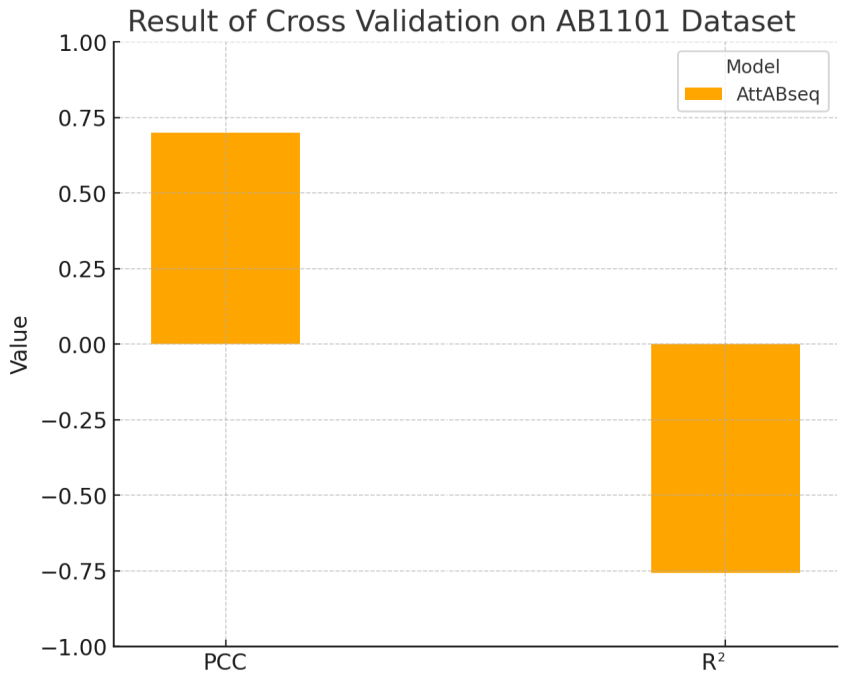

图 11. 训练结果

五、实验分析.

1、数据获取和蛋白质序列的 PSSM 特征生成.

PSSM 的生成大约花费了我十个小时时间。这说明，尽管数据集只有 1101 个蛋白质，但其包含的信息量还是相当巨大的。也许使用更多的数据量，可以训练出更高精度、鲁棒性更强的模型。

2、抗体优化模型训练和模型测试.

2.1 训练过程

据我的猜测，预测结果卡在全 0 点，很可能是因为全零点是一个驻点。我猜测是 loss 函数可能在全零点梯度过小导致的。但是调整 learning rate 的代价是，它 loss 的下降变得比较慢。在 epoch 3 到 epoch 4 时它只下降了 0.5。最后根据图 2，也可以观察到 loss 有一定的收敛。

不过，通过阅读源码，我发现 `class predict` 里前向传播的 `final` 结果都使用了 ReLU，这大概率是导致结果没有负数且有很多 0 的原因。

2.2 训练结果

由于调整了学习率，所以 loss 下降比较慢，后期出现了不降反升的情况。一个很有趣的观察是，训练集的 loss 远远小于验证集的 loss，这说明模型本身存在过拟合。

通过观察图 11，我们可以发现，负数无法被预测，而且预测结果与真实结果差异很大，不过其相对大小关系是可以明显地看出来的，比如真实值是 8，那么预测值大概达到了 3；如果真实值是 $1, 2$ 或者负数，那么预测值很小甚至是 0。因此可以认为，模型学习到了一些关联的信息和特征。**相关系数 pcc 也很直观的说明了这一点。**

R^2 的值都是负数，说明模型的拟合效果完全不够，而后续越来越接近 0，也可以很直观的看出其效果在好。很可惜由于时间原因和经费限制（服务器是自己租的），我只训练到了 15 个 epoch。相信通过一些调整超参数的技巧和更多的资源，我也可以把 R^2 抬到 0 以上。

3、训练结果的可解释性分析.

通过修改并运行下发的 `interpre.py`，获得大量的 `dis` 开头的 `csv` 文件和 `png` 文件。由于时间限制，我选择其中的 13 组数据进行分析：

```
pdb name: 1AK4
mutation: D:P490V
seq mutation: agP_89V
pdb name: 1AK4
mutation: D:P493A
seq mutation: agP_92A
pdb name: 1AK4
mutation: D:V486A
seq mutation: agV_85A
pdb name: 1BJ1
mutation: W:Q87A
seq mutation: agQ_181A
pdb name: 1BJ1
mutation: W:Q89A
seq mutation: agQ_183A
pdb name: 1BJ1
mutation: W:R82A
seq mutation: agR_176A
pdb name: 1BJ1
mutation: W:Y45A
seq mutation: agY_139A
pdb name: 1CZ8
mutation: W:Q87A
seq mutation: agQ_167A
pdb name: 1CZ8
mutation: W:Q89A
seq mutation: agQ_169A
pdb name: 1CZ8
mutation: W:R82A
seq mutation: agR_162A
pdb name: 1CZ8
mutation: W:Y45A
seq mutation: agY_125A
pdb name: 1DQJ
mutation: B:Y50A
seq mutation: abY_263A
pdb name: 1DQJ
mutation: B:Y53A
seq mutation: abY_266A
```

图 12. 运行过程

图 13. 文件夹截图

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		P	I	V	Q	N	L	Q	G	Q	M	V	H	Q	A	I	S	P	R
2	M	0.51837	0.492359	0.437006	0.455198	0.461217	0.467237	0.57008	0.468858	0.492537	0.46486	0.53112	0.412617	0.494536	0.45884	0.378943	0.504709	0.411351	0.467614
3	V	0.510529	0.484518	0.429165	0.447357	0.453376	0.459396	0.562239	0.461017	0.484696	0.457019	0.523279	0.404776	0.486695	0.451	0.371102	0.496868	0.40351	0.459774
4	N	0.478121	0.45211	0.396757	0.414949	0.420968	0.426988	0.529831	0.42861	0.452288	0.424611	0.490871	0.372368	0.454287	0.418592	0.338694	0.46446	0.371102	0.427366
5	P	0.560729	0.534718	0.479365	0.497557	0.503576	0.509596	0.612439	0.511217	0.534896	0.507219	0.573478	0.454976	0.536895	0.501199	0.421302	0.547068	0.453709	0.509973
6	T	0.597534	0.571524	0.516171	0.534362	0.540382	0.546402	0.649245	0.548023	0.571702	0.544025	0.610284	0.491781	0.573701	0.538005	0.458108	0.583874	0.490515	0.546779
7	V	0.514216	0.488205	0.432852	0.451044	0.457064	0.463083	0.565926	0.464705	0.488383	0.460706	0.526966	0.408463	0.490382	0.454687	0.374789	0.500555	0.407197	0.463461
8	F	0.827988	0.801977	0.746624	0.764816	0.770835	0.776855	0.879698	0.778476	0.802155	0.774478	0.840737	0.722235	0.804154	0.768458	0.688561	0.814327	0.720969	0.777232
9	F	0.692581	0.66657	0.611217	0.629409	0.635429	0.641448	0.744291	0.64307	0.666748	0.639072	0.705331	0.586828	0.668747	0.633052	0.553154	0.678921	0.585562	0.641826
10	D	0.595691	0.56968	0.514327	0.532519	0.538538	0.544558	0.647401	0.546179	0.569858	0.542181	0.608441	0.489938	0.571857	0.536162	0.456264	0.58203	0.488672	0.544936
11	I	0.588072	0.562061	0.506708	0.5249	0.53092	0.536939	0.639782	0.538561	0.562239	0.534562	0.600822	0.482319	0.564238	0.528543	0.448645	0.574411	0.481053	0.537317
12	A	0.421879	0.395868	0.340515	0.358707	0.364727	0.370746	0.473589	0.372368	0.396046	0.36837	0.434629	0.316126	0.398045	0.36235	0.282452	0.408219	0.31486	0.371124
13	V	0.570036	0.544025	0.488672	0.506864	0.512883	0.518903	0.621746	0.520524	0.544203	0.516526	0.582785	0.464283	0.546202	0.510506	0.430609	0.556375	0.463016	0.51928
14	D	0.437739	0.411728	0.356375	0.374567	0.380586	0.386606	0.489449	0.388227	0.411906	0.384229	0.450489	0.331986	0.413905	0.37821	0.298312	0.424078	0.33072	0.386984
15	G	0.595691	0.56968	0.514327	0.532519	0.538538	0.544558	0.647401	0.546179	0.569858	0.542181	0.608441	0.489938	0.571857	0.536162	0.456264	0.58203	0.488672	0.544936
16	E	0.536562	0.510551	0.455198	0.47339	0.479409	0.485429	0.588272	0.48705	0.510729	0.483052	0.549311	0.430809	0.512728	0.477032	0.397135	0.522901	0.429543	0.485806
17	P	0.718947	0.692936	0.637583	0.655775	0.661795	0.667814	0.770657	0.669436	0.693114	0.665438	0.731697	0.613194	0.695113	0.659418	0.57952	0.705287	0.611928	0.668192
18	L	0.518192	0.492181	0.436828	0.45502	0.46104	0.467059	0.569902	0.468681	0.492359	0.464682	0.530942	0.412439	0.494358	0.458663	0.378765	0.504531	0.411173	0.467437
19	G	0.507619	0.481608	0.426255	0.444447	0.450466	0.456486	0.559329	0.458108	0.481786	0.454109	0.520369	0.401866	0.483785	0.44809	0.368192	0.493958	0.4006	0.456864
20	R	0.499445	0.473434	0.418081	0.436273	0.442292	0.448312	0.551155	0.449933	0.473612	0.445935	0.512195	0.393692	0.475611	0.439916	0.360018	0.485784	0.392426	0.448689
21	V	0.590249	0.564238	0.508885	0.527077	0.533096	0.539116	0.641959	0.540737	0.564416	0.536739	0.602999	0.484496	0.566415	0.53072	0.450822	0.576588	0.48323	0.539494
22	S	0.48303	0.457019	0.401666	0.419858	0.425877	0.431897	0.53474	0.433518	0.457197	0.42952	0.49578	0.377277	0.459196	0.423501	0.343603	0.469369	0.376011	0.432275
23	F	0.669236	0.643225	0.587872	0.606064	0.612083	0.618103	0.720946	0.619725	0.643403	0.615726	0.681986	0.563483	0.645402	0.609707	0.529809	0.655575	0.562217	0.618481
24	E	0.433541	0.40753	0.352177	0.370369	0.376388	0.382408	0.485251	0.384029	0.407708	0.380031	0.446291	0.327788	0.409707	0.374012	0.294114	0.41988	0.326522	0.382785
25	L	0.522745	0.496735	0.441382	0.459574	0.465593	0.471613	0.574456	0.473234	0.496913	0.469236	0.535495	0.416992	0.498912	0.463216	0.383319	0.509085	0.415726	0.47199
26	F	0.830142	0.804131	0.748778	0.76697	0.77299	0.779009	0.881853	0.780631	0.804309	0.776633	0.842892	0.724389	0.806308	0.770613	0.690715	0.816482	0.723123	0.779387
27	A	0.639738	0.613727	0.558374	0.576566	0.582586	0.588605	0.691448	0.590227	0.613905	0.586228	0.652488	0.533985	0.615904	0.580209	0.500311	0.626077	0.532719	0.588983
28	D	0.562039	0.536028	0.480675	0.498867	0.504887	0.510906	0.613749	0.512528	0.536206	0.50853	0.574789	0.456286	0.538205	0.50251	0.422612	0.548378	0.45502	0.511284

图 14. dis-1AK4-D_P490V-agP_89V-0.csv

图 16. dis-1AK4-D_P490V-agP_89V-0.png 由于时间原因，我选择了最简单的一个解释分析：考察关键位置蛋白质。我们对抗体上每一个位点的氨基酸，统计其对抗原上每一个位置的权重和，取每一张 `csv` 里和最大的六个氨基酸进行统计，使用以下代码：

```
import pandas as pd
file_name=[
    'dis-1AK4-D_P490V-agP_89V-0.csv',
    'dis-1AK4-D_P493A-agP_92A-1.csv',
    'dis-1AK4-D_V486A-agV_85A-2.csv',
    'dis-1BJ1-W_Q87A-agQ_181A-3.csv',
    'dis-1BJ1-W_Q89A-agQ_183A-4.csv',
    'dis-1BJ1-W_R82A-agR_176A-5.csv',
    'dis-1BJ1-W_Y45A-agY_139A-6.csv',
    'dis-1CZ8-W_Q87A-agQ_167A-7.csv',
    'dis-1CZ8-W_Q89A-agQ_169A-8.csv',
    'dis-1CZ8-W_R82A-agR_162A-9.csv',
    'dis-1CZ8-W_Y45A-agY_125A-10.csv',
    'dis-1DQJ-B_Y50A-abY_263A-11.csv',
    'dis-1DQJ-B_Y53A-abY_266A-12.csv'
```

```
]
dict={}
for name in file_name:
    data = pd.read_csv(name)
    #print(data['Unnamed: 0'])'
    L=len(data['Unnamed: 0'])
    answer=[]
    for id in range(L):
        A=''
        B=0
        for i in data.iloc[id]:
            if (type(i)!=type('')):
                B+=i
            else:
                A=i
        answer.append((B,A))
    answer.sort()
    for i in range(1,7):
        if dict.get(answer[-i][1])==None:
            dict[answer[-i][1]]=1
        else:
            dict[answer[-i][1]]+=1
print(dict)
```

得到结果：

`{'K': 10, 'Y': 8, 'F': 5, 'S': 14, 'T': 1, 'C': 1, 'P': 5, 'M': 1, 'N': 3, 'V': 8, 'G': 10, 'L': 5, 'A': 2, 'Q': 3, 'H': 2}`

可以发现，抗体上影响较大的氨基酸还是非常有规律的。出现次数较多的分别是：`S`、`K`、`G`、`Y`、`V`。经过查询这分别是：丝氨酸、赖氨酸、甘氨酸、酪氨酸、缬氨酸。其大多为中性或者中性偏碱，但是和亲水疏水没有太大的关系。

六、讨论和心得.

这次的实验让我深刻体会到了人工智能在蛋白质工程领域的重要性。通过分析蛋白质氨基酸突变对结合自由能的影响，我不仅对模型的建模方法有了一定的了解（比如在矩阵上直接使用 `CNN`），还学会了很多生物信息领域的相关知识，比如 `blast`，PSSM，以及了解到了使用 `pdb` 数据库来查询蛋白质结构信息。我还完成了对重要氨基酸位置的查询和分析，并且结合学过的生物学和化学知识，对此进行了自己的猜测，虽然不一定准确，但是非常有趣。

收获最大的，还是训练模型的 `debug` 过程。我认识到了 `torch` 版本和 `numpy` 版本对应的重要性，以及明白了 `AI` 领域中“配置环境”的难点所在。我通过对错误信息的分析，找到了修改方式，并让模型成功的开始训练。当然，在遇到 `predict` 值全是 0 的情况下，我通过自行的探究，通过调整超参数，很好地解决了这个问题。这让我的自主学习能力和科研素养得到了提高了磨炼。