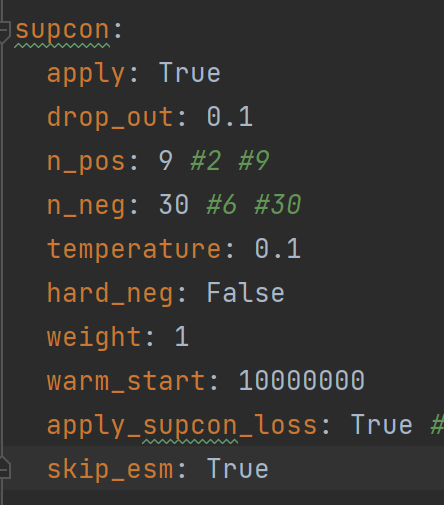
**[Yichuan0712/MyTarget2 (github.com)](https://github.com/Yichuan0712/MyTarget2)**

**这个链接是我已经修改好的, 当您遇到问题时可以参考这个**

**在config.yaml**

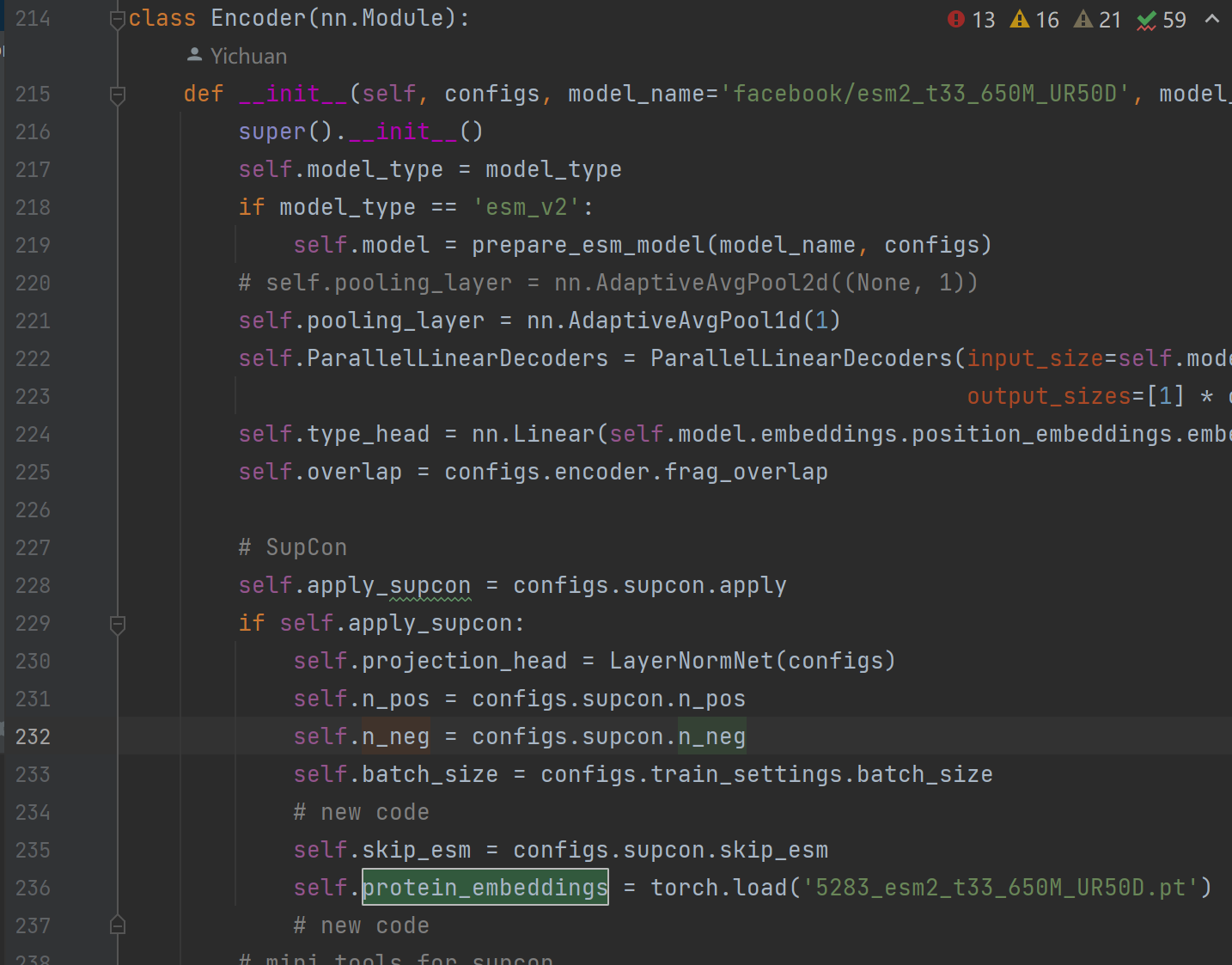
****

**在supcon里面增加一个skip\_esm, 设置为true, 这个用来跳过esm2 infer, 可以像clean一样直接用保存好的embedding来训练contrastive learning的部分**

**接下来实现一下这个skip功能,**

**首先, 把5283\_esm2\_t33\_650M\_UR50D.pt这个文件放到项目里, 直接跟其他代码放在同一个目录下就行**

**之后, 在model.py, 找到class Encoder的\_\_init\_\_()**

****

**找到if self.apply\_supcon: (在大约229行左右)**

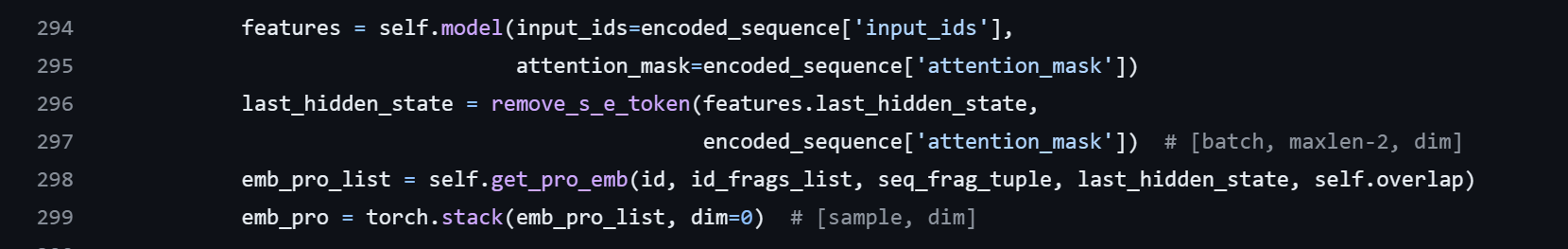
**在这个if条件的最末尾添加以下两行代码 (上图的行235和236已经添加)**

**self.skip\_esm = configs.supcon.skip\_esm**

**self.protein\_embeddings = torch.load('5283\_esm2\_t33\_650M\_UR50D.pt')**

**接下来找到class Encoder的forward()**

**在大约294行的位置, 有四行代码, 如下**

****

**把上面这段代码删掉, 换成下面的代码**

**if self.skip\_esm:**

**emb\_pro\_list = []**

**for i in id:**

**emb\_pro\_list.append(self.protein\_embeddings[i])**

**emb\_pro = torch.stack(emb\_pro\_list, dim=0)**

**else:**

**features = self.model(input\_ids=encoded\_sequence['input\_ids'],**

**attention\_mask=encoded\_sequence['attention\_mask'])**

**last\_hidden\_state = remove\_s\_e\_token(features.last\_hidden\_state,**

**encoded\_sequence['attention\_mask']) # [batch, maxlen-2, dim]**

**""""add something here? the 1+n\_pos+n\_neg"""**

**emb\_pro\_list = self.get\_pro\_emb(id, id\_frags\_list, seq\_frag\_tuple, last\_hidden\_state, self.overlap)**

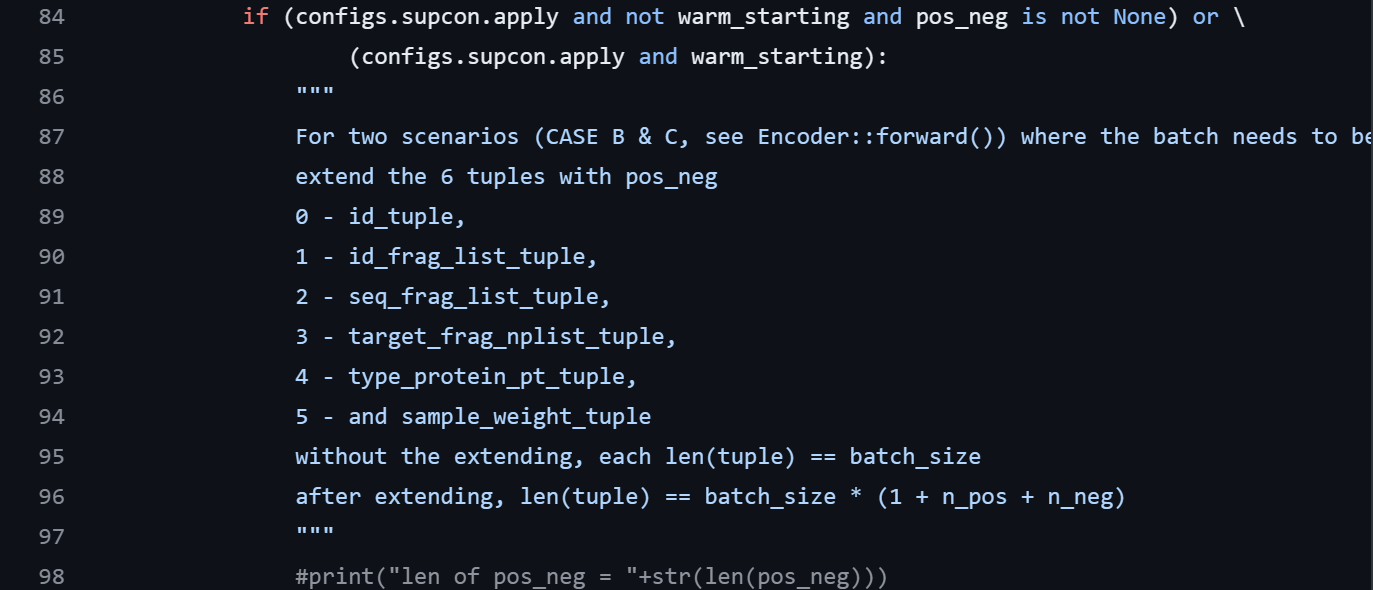
**emb\_pro = torch.stack(emb\_pro\_list, dim=0) # [sample, dim]**

**这样就增加了一个if self.skip\_esm条件, 当条件满足时, 直接从.pt文件获取esm2 infer的embedding, 否则就执行老版本的代码**

**接下来到train.py, 在84行左右, 有一个if条件**

**if (configs.supcon.apply and not warm\_starting and pos\_neg is not None) or \**

**(configs.supcon.apply and warm\_starting):**

****

**把这个if 条件里的内容全部删除, 替换为以下代码**

**"""**

**For two scenarios (CASE B & C, see Encoder::forward()) where the batch needs to be extended,**

**extend the 6 tuples with pos\_neg**

**0 - id\_tuple,**

**1 - id\_frag\_list\_tuple,**

**2 - seq\_frag\_list\_tuple,**

**3 - target\_frag\_nplist\_tuple,**

**4 - type\_protein\_pt\_tuple,**

**5 - and sample\_weight\_tuple**

**without the extending, each len(tuple) == batch\_size**

**after extending, len(tuple) == batch\_size \* (1 + n\_pos + n\_neg)**

**"""**

**flag\_batch\_extension = True**

**for one\_in\_a\_batch in range(b\_size):**

**# pos\_neg[one\_in\_a\_batch][0]**

**for one\_of\_pos in range(configs.supcon.n\_pos):**

**# pos\_neg[one\_in\_a\_batch][0][one\_of\_pos]**

**id\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][0],)**

**id\_frag\_list\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][1],)**

**seq\_frag\_list\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][2],)**

**target\_frag\_nplist\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][3],)**

**type\_protein\_pt\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][4],)**

**sample\_weight\_tuple += (pos\_neg[one\_in\_a\_batch][0][one\_of\_pos][5],)**

**for one\_in\_a\_batch in range(b\_size):**

**# pos\_neg[one\_in\_a\_batch][1]**

**for one\_of\_neg in range(configs.supcon.n\_neg):**

**# pos\_neg[one\_in\_a\_batch][1][one\_of\_neg]**

**id\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][0],)**

**id\_frag\_list\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][1],)**

**seq\_frag\_list\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][2],)**

**target\_frag\_nplist\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][3],)**

**type\_protein\_pt\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][4],)**

**sample\_weight\_tuple += (pos\_neg[one\_in\_a\_batch][1][one\_of\_neg][5],)**

**type\_protein\_pt\_tuple\_temp = [torch.from\_numpy(x) if isinstance(x, np.ndarray) else x for x in type\_protein\_pt\_tuple]**

**type\_protein\_pt\_tuple = type\_protein\_pt\_tuple\_temp**

**上面的代码处理了旧代码在进行维度转换时的错误**

**完成上面的修改后, 还需要修改class Encoder, 请回到model.py**

**首先在class Encoder中增加下面这个类方法, 这个类方法用于进一步处理数据, 转换为(batch\_size, 1+n\_pos+n\_neg, len\_of\_embedding)的格式**

**def reorganize\_emb\_pro(self, emb\_pro):**

**n\_batch = int(emb\_pro.shape[0] / (1 + self.n\_pos + self.n\_neg))**

**bch\_anchors, bch\_positives, bch\_negatives = torch.split(emb\_pro,**

**[n\_batch, n\_batch \* self.n\_pos, n\_batch \* self.n\_neg],**

**dim=0)**

**emb\_pro\_ = []**

**for i in range(n\_batch):**

**anchor = bch\_anchors[i].unsqueeze(0)**

**positive = bch\_positives[(i \* self.n\_pos):(i \* self.n\_pos + self.n\_pos)]**

**negative = bch\_negatives[(i \* self.n\_neg):(i \* self.n\_neg + self.n\_neg)]**

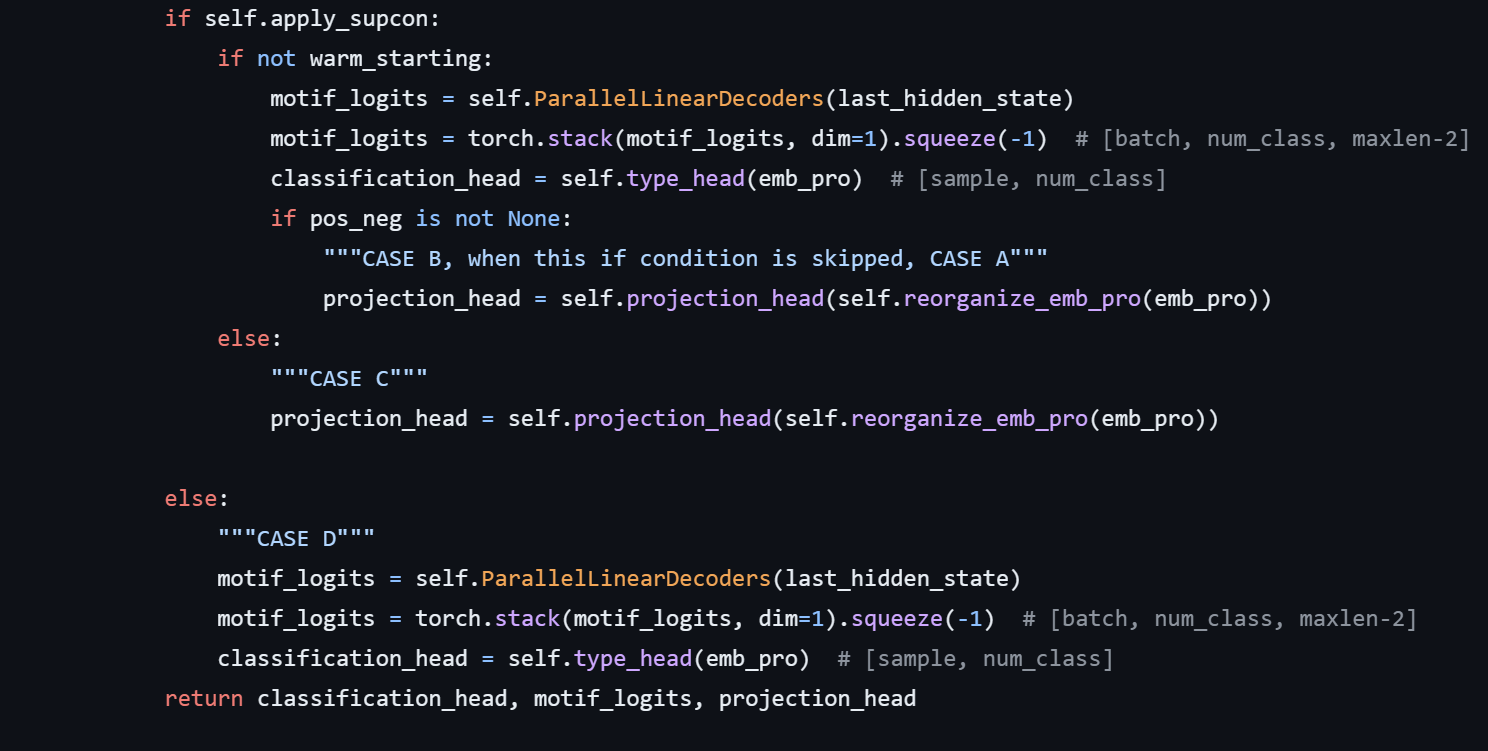
**triple = torch.cat((anchor, positive, negative), dim=0)**

**emb\_pro\_.append(triple)**

**emb\_pro\_ = torch.stack(emb\_pro\_, dim=0)**

**return emb\_pro\_**

**之后, 找到Encoder的forward()方法**

****

**找到CASE B和CASE C, 做如上图所示的修改**

**将原来的**

**~~emb\_pro\_ = emb\_pro.view((self.batch\_size, 1 + self.n\_pos + self.n\_neg, -1))~~**

**~~projection\_head = self.projection\_head(emb\_pro\_)~~**

**替换为**

**projection\_head = self.projection\_head(self.reorganize\_emb\_pro(emb\_pro))**

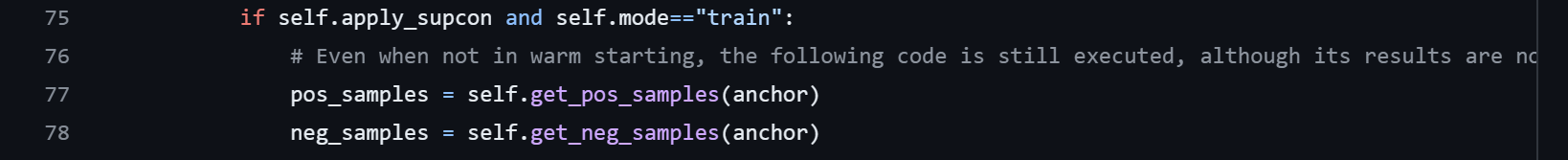
**旧代码还有一个bug, 如果选择 ’clean’ 模式, batch size将会是8, 分别属于8个不同的类型**

**但是旧代码误用了错误的index导致取出的8个anchor不满足 ’属于8个不同的类型’**

**在data\_clean.py中**

**找到class LocalizationDataset的\_\_getitem\_\_()方法**

**在约75行左右的这个位置做如下的修改**

****

**将get\_pos\_samples(idx)和get\_neg\_samples(idx)改成**

**get\_pos\_samples(anchor)和get\_neg\_samples(anchor)**