Algorithm1 Permute Feedforward

def PermuteFeedforward(protein\_feature):

/\* protein\_feature size: (batch\_size, 2500, feat\_space\_dim) \*/

/\* dense = Linear(2500, num\_steps) \*/

1: protein\_feature <- protein\_feature.permute(0, 2, 1)

2: protein\_feature <- ReLU(dense(protein\_feature))

3: protein\_feature <- protein\_feature.permute(0, 2, 1)

4: return protein\_feature

Algorithm 2 PALACE

class PALACE:

def \_\_init\_\_(self):

1: self.seed /\* set the backend of cuda and random \*/

2: self.feat\_space\_dim /\* d\_model in pytorch \*/

3: self.prot\_blks /\* number of protein encoder blocks \*/

4: self.smi\_blks /\* number of SMILES encoder blocks \*/

5: self.cross\_blks /\* number of cross encoder blocks \*/

6: self.dec\_blks /\* number of decoder blocks \*/

7: self.dropout /\* dropout ratio \*/

8: self.batch\_size /\* batch size \*/

9: self.num\_steps /\* number of steps per sequence \*/

10: self.lr /\* learning rate \*/

11: self.num\_epochs /\* number of epochs \*/

12: self.ffn\_num\_hiddens /\* number of hidden units in linear layer \*/

13: self.num\_heads /\* number of heads in multi-head self attention \*/

14: smi\_encoder <- encoder(SMILES\_vocab\_size, self.feat\_space\_dim, self.ffn\_num\_hiddens, self.num\_heads, self.smi\_blks, self.dropout)

15: prot\_encoder <- encoder(protein\_vocab\_size, self.feat\_space\_dim, self.ffn\_num\_hiddens, self.num\_heads, self.prot\_blks, self.dropout)

16: cross\_encoder <- encoder(SMILES\_vocab\_size, self.feat\_space\_dim, self.ffn\_num\_hiddens, self.num\_heads, self.cross\_blks, self.dropout)

17: decoder <- decoder(SMILES\_vocab\_size, self.feat\_space\_dim, self.ffn\_num\_hiddens, self.num\_heads, self.dec\_blks, self.dropout)

18: Linear <- Linar(self.feat\_space\_dim, SMILES\_vocab\_size)

19: softmax <- LogSoftmax

20: optimizer <- NAdm

21: loss <- CrossEntropyLoss

22: scheduler <- ReduceLROnPlateau

Algorithm 3 Training

def train(protein, SMILES):

/\* encoder input shape: (batch\_size, num\_steps) \*/

/\* encoder output shape: (batch\_size, num\_steps, feat\_space\_dim) \*/

/\* decoder input shape: (batch\_size, num\_steps) \*/

/\* decoder output shape: (batch\_size, num\_steps, SMILES\_vocab\_size) \*/

1: deepnorm initiation

2: protein <- prot\_encoder(protein)

3: SMILES <- smi\_encoder(SMILES)

3: mix <- protein + SMILES

4: mix <- Deepnorm(FeedForward(mix), ( mix + protein + SMILES) / 3)

5: encoder\_outputs <- cross\_encoder(mix)

6: decoder\_outputs <- decoder(mix)

7: PALACE\_outputs <- softmax(Linear(decoder\_outputs))

8.loss.backward(PALACE\_outputs, labels)