

# 241102

## 模型改进

### Funnel Transformer的渐进式压缩策略

借鉴Funnel-Transformer (NeurIPS 2020), 采用类似Funnel Transformer的渐进式压缩策略, 对DeepDXF模型的序列长度从512调整为4096

#### 渐进式降维:

- 不是一次性从4096降到64
- 每次只降低一半, 保留更多信息

#### 局部特征提取:

- 使用Conv1D捕获局部上下文
- `kernel_size=3`保证每个token都能看到左右邻居

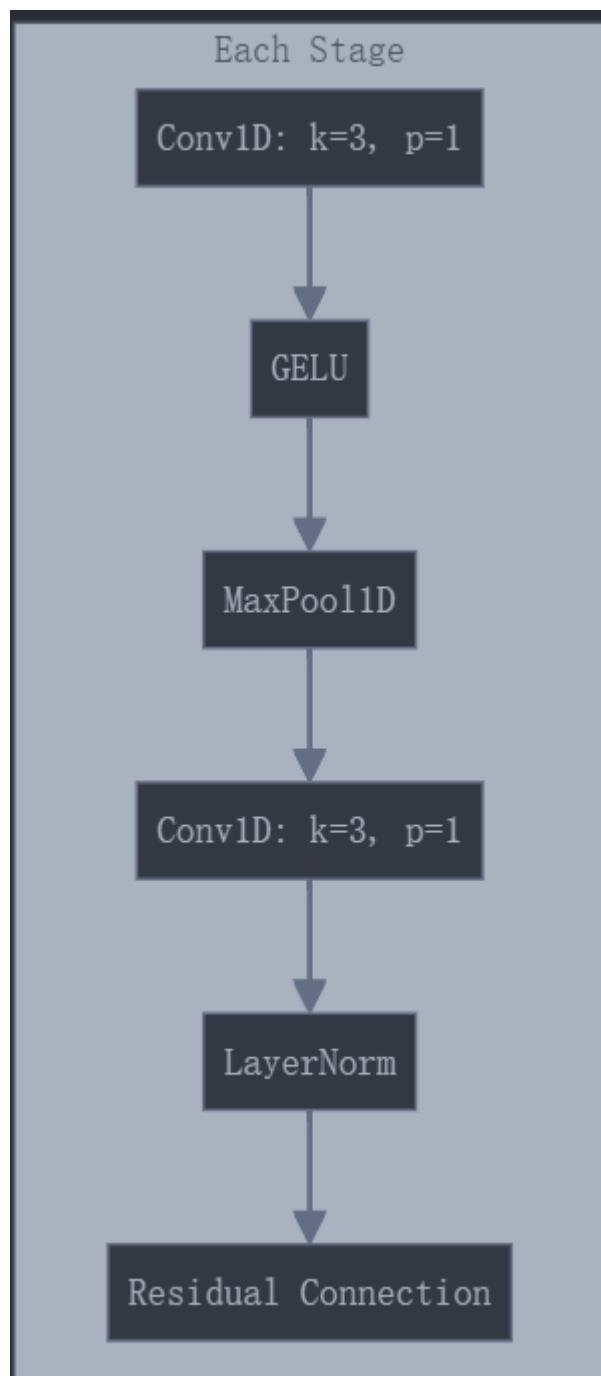
#### 残差连接:

- 缓解梯度消失问题
- 保留原始信息

#### 特征保持:

- 虽然序列长度从4096降到64
- 但特征维度256始终保持不变

损失降低了一点, h5文件数量从49到79



## 实验指标

### CGMN相关指标

1. AUC (Area Under Curve)分数:通过ROC曲线计算得到

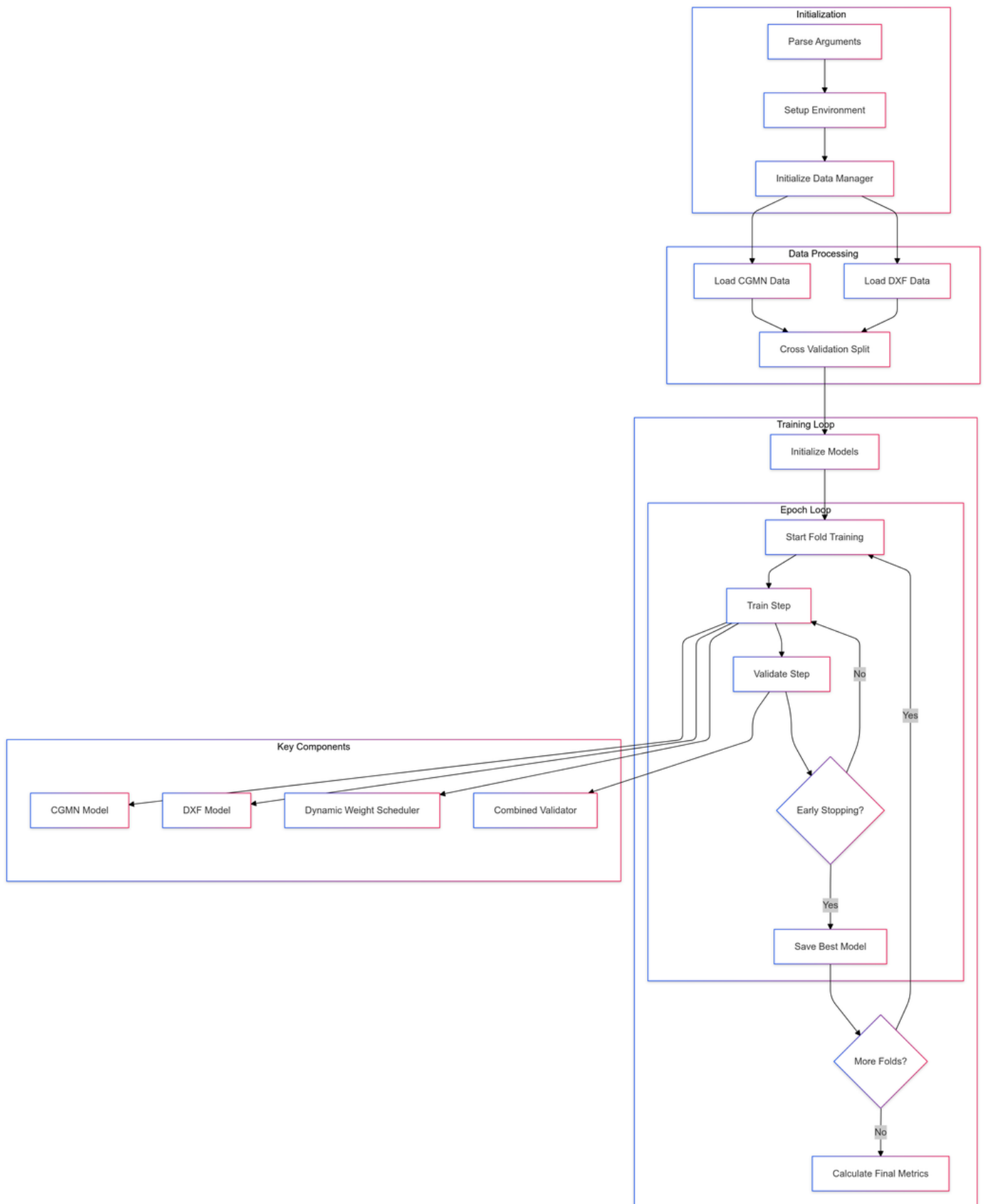
```
1 fpr, tpr, _ = roc_curve(truth, (1 - diff) / 2) model_auc = auc(fpr, tpr)
```

2. LR AUC:使用LogisticRegression分类器评估的AUC分数

```
1 aucc = self.evaluation((1 - diff) / 2, truth, ratio=0.1)
```

## 联合训练框架的改进

- 1.自适应损失权重
- 2.DeepDXF数据加载全部拿来训练了，没有验证和测试
- 3.早停机制只是基于CGMN



## 待解决

- 1.数据集再补充(cx的尾巴数据，task1的数据)
- 2.dwg->dxf工具统一，看看不同提取工具的效果


[ODAFileConverter-DWG2DXF] <https://www.opendesign.com/guestfiles/TeighaFileConverter>

利用ODAFileConverter转化有的提取不了信息，提取为2018 ASCII DXF类型

但是用AutoCAD却可以

> srtp > dwg > 2018\_ASCII\_DXF

↓ 排序 ▾    ≡ 查看 ▾    ...

名称	修改日期	类型	大小
 8-040116-SOP1(ETCH) REV A.dxf	2024/10/27 15:05	AutoCADDrawin...	2,292 KB

3.实体类别，实体参数需要再计算整理

4.文本的内容后序继续优化，把文字部分提取出来（后面单独比较也是可以的，数组？）

5.dxf\_CGMN\_process.py需要处理，从dxf->不同类，不同区域的json，而不是所有dxf->一份json

6.dxf\_CGMN\_process.py中HATCH的处理部分需要修正

## 想法

1.损失函数有没有正则化项，加入正则化项，数量作为损失函数的加权

$$\min_{\xi_R} \mathcal{L}_{\text{NCACF-R}}(\xi_R) := \sum_{u,i} c_{u,i} (r_{u,i} - \psi_{\gamma}(w_u, h_i))^2 + \lambda_W \sum_u \|w_u\|^2 + \lambda_H \sum_i \|h_i - \phi_{\theta}(x_i)\|^2$$

- 参数集合：  
 $\xi_R = \{\theta, \gamma, W, H\}$  是要优化的全部参数集合。

目标函数的组成：

1. 损失函数部分：

- 使用加权平方误差  $\sum_{u,i} c_{u,i} (r_{u,i} - \psi_{\gamma}(w_u, h_i))^2$  衡量模型预测  $\psi_{\gamma}(w_u, h_i)$  与真实交互  $r_{u,i}$  之间的误差。

2. 正则化项：

- $\lambda_W \sum_u \|w_u\|^2$ ：控制用户嵌入  $w_u$  的复杂度，防止过拟合。
- $\lambda_H \sum_i \|h_i - \phi_{\theta}(x_i)\|^2$ ：控制项目嵌入与内容特征的差异，确保项目嵌入  $h_i$  与内容特征提取器的输出  $\phi_{\theta}(x_i)$  保持一致。

$$\min_{\xi_R} \mathcal{L}_{\text{MF-R}}(\xi_R) := \sum_{u,i} c_{u,i} (r_{u,i} - w_u^{\top} h_i)^2 + \lambda_W \sum_u \|w_u\|^2 + \lambda_H \sum_i \|h_i - \phi_{\theta}(x_i)\|^2$$

7.对于CGMN数据集数量不需要限制试一试

8.联合训练的代码debug

9.根据文档的检索方式来考虑

10.在CGMN那条线，考虑不同类别的重要性，比如椭圆比LINE更重要，因为出现次数少，所含的信息量大，

当对每一份dxf文件，得到每一类的相似度，对每一类的相似度赋予权重，权重计算方式为idf权重的计算方式或者是**tf-idf的计算方式（最出名的经典排序方法）**得到权重，最后再对同一份DXF文件内各类的权重归一化

## idf 权重

- $df_t$  是出现词项 $t$ 的文档数目
- $df_t$  是和词项 $t$ 的信息量成反比的一个值
- 于是可以定义词项 $t$ 的**idf权重**:

$$idf_t = \log_{10} \frac{N}{df_t}$$

(其中 $N$  是文档集中文档的数目)

- $idf_t$  是反映词项 $t$ 的信息量的一个指标
- 实际中往往计算 $[\log N/df_t]$ 而不是  $[N/df_t]$ ，这可以对idf的影响有所抑制
- 值得注意的是，对于tf 和idf我们都采用了对数计算方式

# tf-idf权重计算

- 词项的tf-idf权重是tf权重和idf权重的乘积

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$

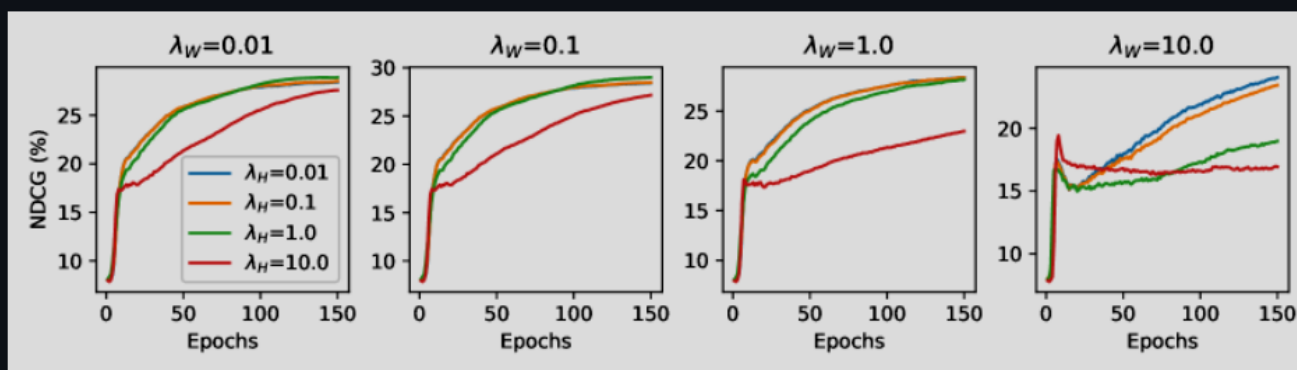
- 信息检索中最出名的权重计算方法
- 注意：上面的“-”是连接符，不是减号
- 其他叫法：tf.idf、tf x idf

## 实验

主要参考DeepCAD的实验方式

### 1.transformer的隐藏层数

### 2.epoch，损失函数的参数等



(a) Relaxed

(一) 放松

### 3.各种初始化的方式

### 4.损失函数的选择可以关注下哪种更好

### 5.数据集的划分（过拟合可能是训练，测试，验证数据集划分的原因）

### 3. 模型训练与评估：10折交叉验证 (10-fold Cross-Validation)

- 验证集与测试集划分：
  - 选取 **20%** 的歌曲作为**验证集**。
  - 剩余的歌曲被随机划分为 **10个大小相等的子集**，分别用于训练和测试。
  - 在**冷启动场景**中，模型无法访问验证集和测试集中歌曲的播放次数。
- 超参数调整：
  - 为减少计算开销，超参数只在**第一次训练/测试划分**上进行调整。
  - 调整后的最优值会用于剩余的9次划分。

6.我的那条线也保存，不要删除（邻接表的建立方式，需要更改下，有接触就建立边，特别是直线部分），和cl作为对照

7.CGMN最后可以逐个原始相乘或拼接

8.时间打印，其它的删除，结构很清晰，科学

```
already load adj matrix (70839, 70839) 3.482644557952881
Epoch 0 [108.6s]: train==[439.13647=439.09998 + 0.03576]
Epoch 1 [106.9s]: train==[233.95430=233.91814 + 0.03616]
Epoch 2 [106.6s]: train==[195.93034=195.89395 + 0.03641]
Epoch 3 [105.4s]: train==[173.43889=173.40219 + 0.03662]
```

```
n_interactions=1027370
n_train=810128, n_test=217242, sparsity=0.00084
already load adj matrix (70839, 70839) 3.482644557952881
Epoch 0 [108.6s]: train==[439.13647=439.09998 + 0.03576]
Epoch 1 [106.9s]: train==[233.95430=233.91814 + 0.03616]
Epoch 2 [106.6s]: train==[195.93034=195.89395 + 0.03641]
Epoch 3 [105.4s]: train==[173.43889=173.40219 + 0.03662]
Epoch 4 [106.3s]: train==[153.79224=153.75539 + 0.03680]
Epoch 5 [107.5s]: train==[140.98706=140.95001 + 0.03695]
Epoch 6 [106.5s]: train==[134.26880=134.23181 + 0.03708]
Epoch 7 [107.0s]: train==[130.25456=130.21735 + 0.03720]
Epoch 8 [107.2s]: train==[127.38524=127.34794 + 0.03732]
Epoch 9 [106.8s + 200.7s]: train==[124.83270=124.79528 + 0.03745], recall=[0.03819, 0.12521], precision=[0.01223, 0.00798], hit=[0.18923, 0.43382], ndcg=[0.02714, 0.05298]
Save the weights in path: ../weights/gowalla_epoch_9.pkl
Epoch 10 [104.3s]: train==[121.82255=121.78493 + 0.03758]
Epoch 11 [105.1s]: train==[119.27834=119.24073 + 0.03773]
```

9.拆分完后需要把块分解，去除INSERT，多段线（看看autoCAD的能不能自动拆分，自己再去了解下DIMENSION里面的引用块有没有算在INSERT内）



Model\_Space显示:

- INSERT: 65个      # 这里的INSERT是块引用, 每个引用只算作1个实体
- LINE: 2207个      # 这是直接在Model\_Space中绘制的线条
- CIRCLE: 277个      # 这是直接在Model\_Space中绘制的圆
- ...

SH1块定义包含:

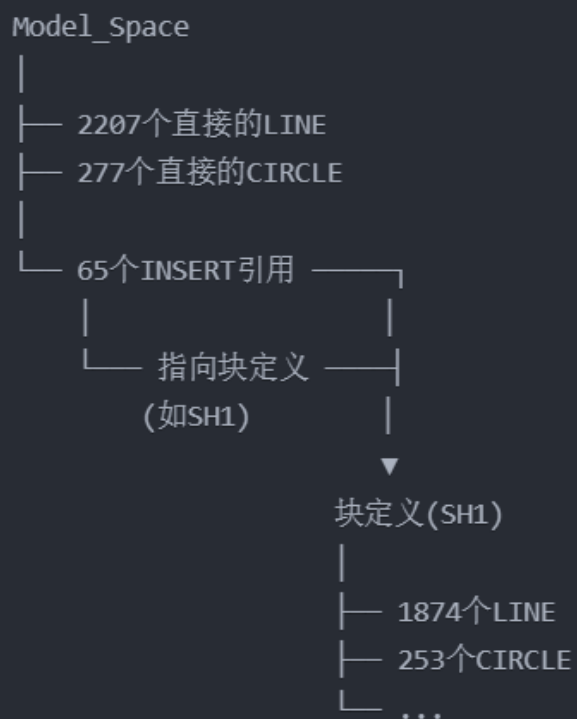
- LINE: 1874个      # 这些线条是块定义的一部分
- CIRCLE: 253个      # 这些圆是块定义的一部分
- ...

2. 举例说明: 假设Model\_Space引用了一次SH1块:

- 在Model\_Space中只会计算为1个INSERT实体
- 不会把SH1中的1874个LINE加到Model\_Space的2207个LINE中
- 实际显示时你能看到所有图形, 但在实体计数时它们属于不同的层级

3. 类比解释: 这就像文件系统中的快捷方式:

- 一个快捷方式 (INSERT) 只占用很小空间
- 但它可以引用到包含大量内容的文件夹 (块定义)
- 计算快捷方式数量时, 不会把目标文件夹中的文件数量加进来



所以：

- Model\_Space中的实体数量只计算直接存在于Model\_Space中的实体
- 块引用（INSERT）被计算为单个实体
- 块定义中的实体数量不会加到Model\_Space的计数中