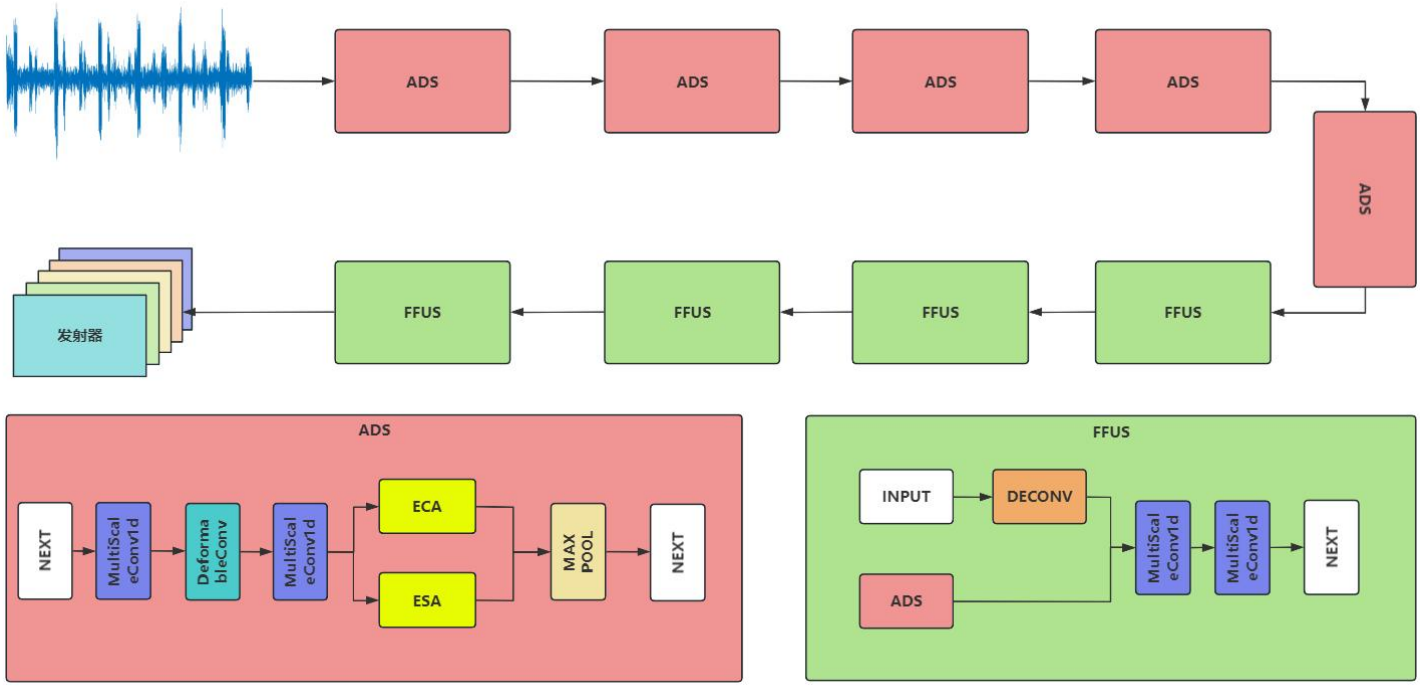
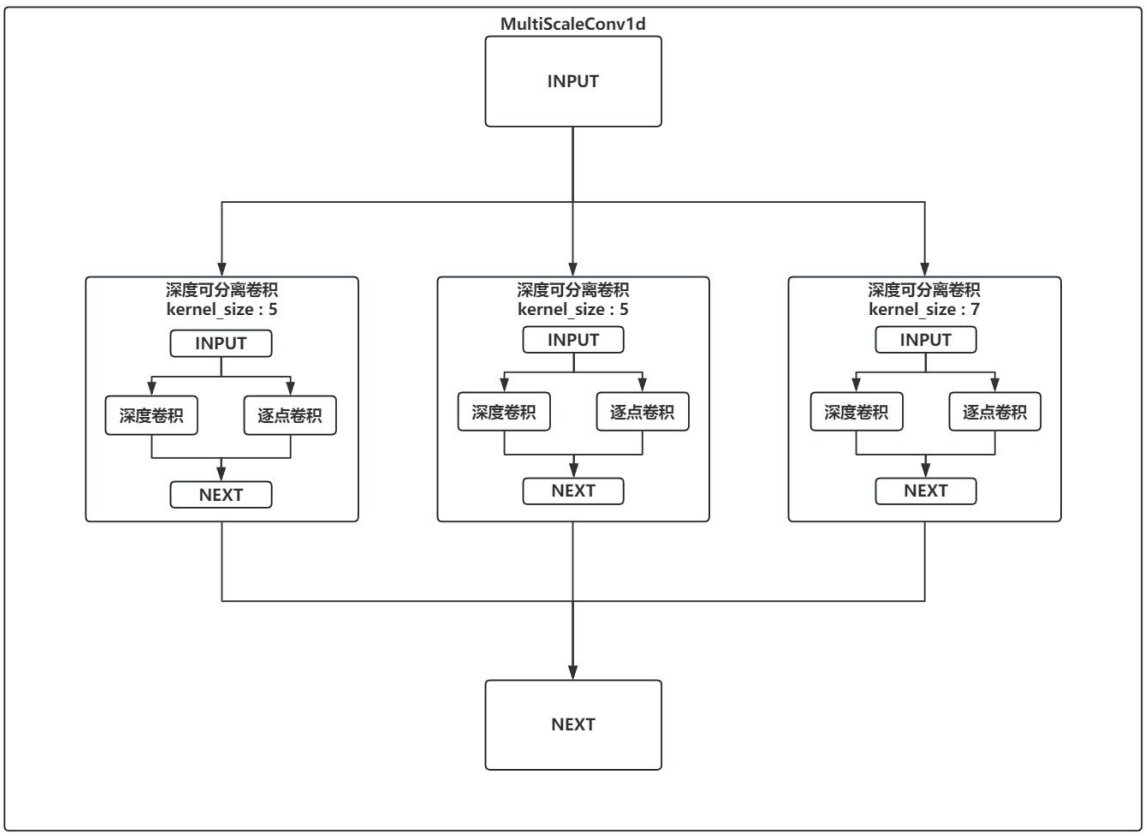
信号分选

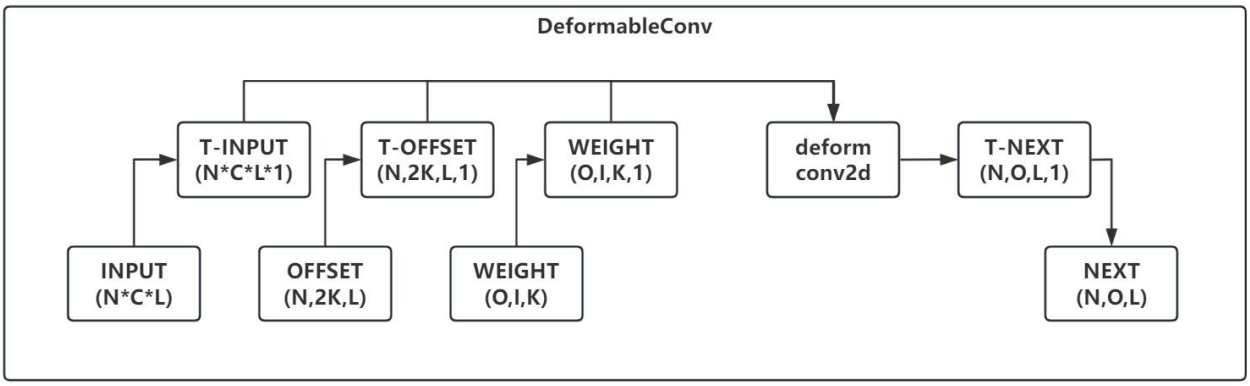
对于接收信号的 stft 图像加上位置信息这个点，目前还是不知道改如何正确的加入。

但是针对极端重叠信号的处理，我采用了可变卷积，通过自适应调整感受野，来进一步学习极端重叠情况下各个信号之间的不同特征，然后再通过深度可分离卷积和多尺度卷积来降低整个训练过程的参数量和浮点运算次数，训练得到的 D-CAU 模型结果如下：

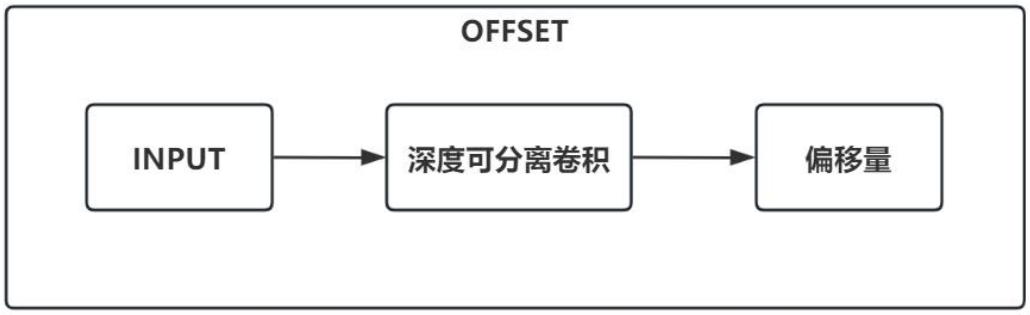
在整个模型 **FLOPs 为原模型的 0.48** 和**浮点运算次数为原来的 0.504** 的情况下，训练得到的模型具有比 CAUNet 更好的性能，**相似性（SIM）提高百分之 6.5，交并比（IOU）提高百分之 16**。得到一个更加轻量化，性能更好的模型。







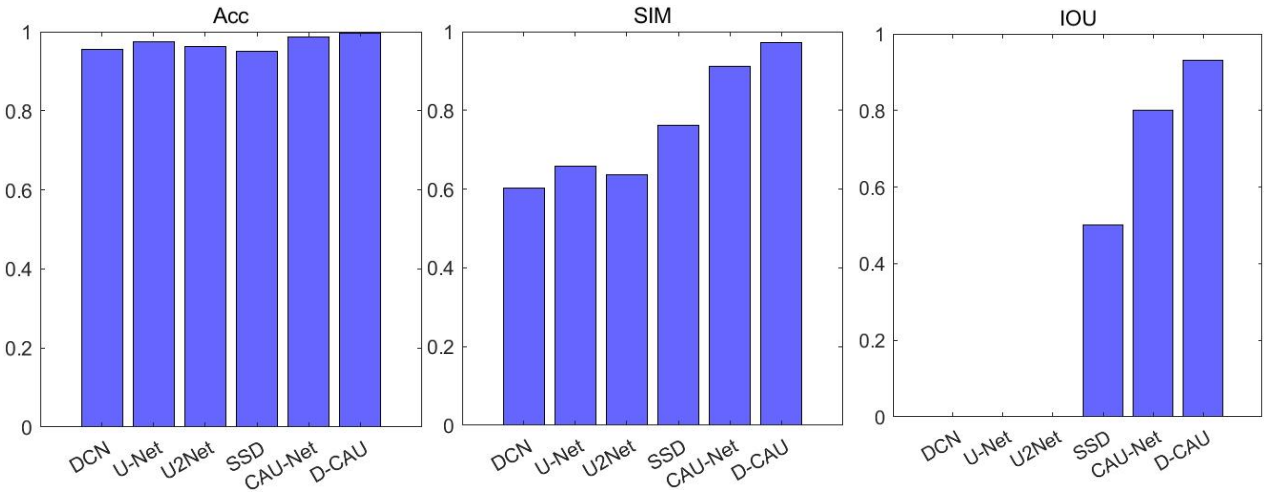
Deformconv2d　输出为(N , O , L , 1)
INPUT　N 是批量大小，C 是通道数，L 是序列长度
OFFSET　N 是批量大小，K 是卷积核，L 是序列长度 **偏置初始化**(从均匀分布中采样偏置值)
WEIGHT　O 是输出，I 是输入，K 是卷积核　**权重初始化**(使用 Kaiming 初始化方法)



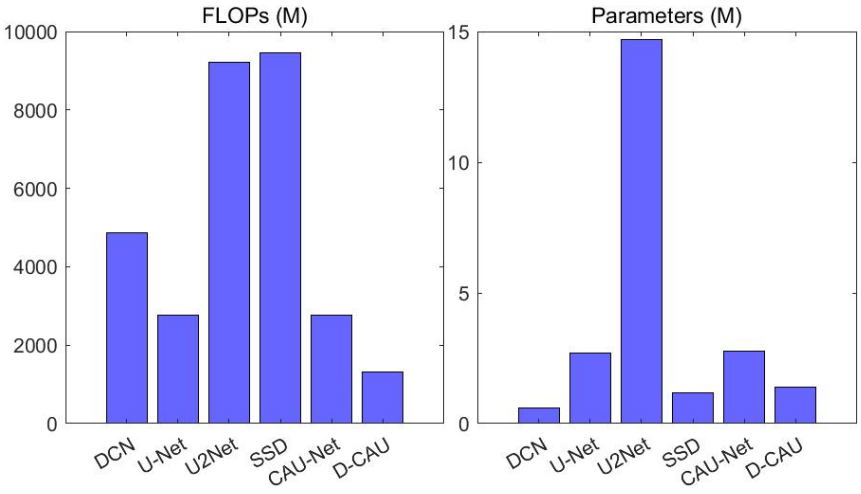
CAU 模型在 **SIM** 和 **IOU** 这两个参数上的表现不是很好，尤其是在面对**多发射器**的情况下 **IOU 性能很差**，这是因为面对多发射器，**重叠**的情况也是增加，该模型不能很好的完成分选。于是针对这一点，采用**可变卷积**，进一步学习特征，得到更好的性能表现。

然后再用**深度可分离的多尺度卷积**，代替其 ADS 模块和 FFUS 模块中的卷积操作，在增加模型性能的同时，减少模型所需要的参数量。

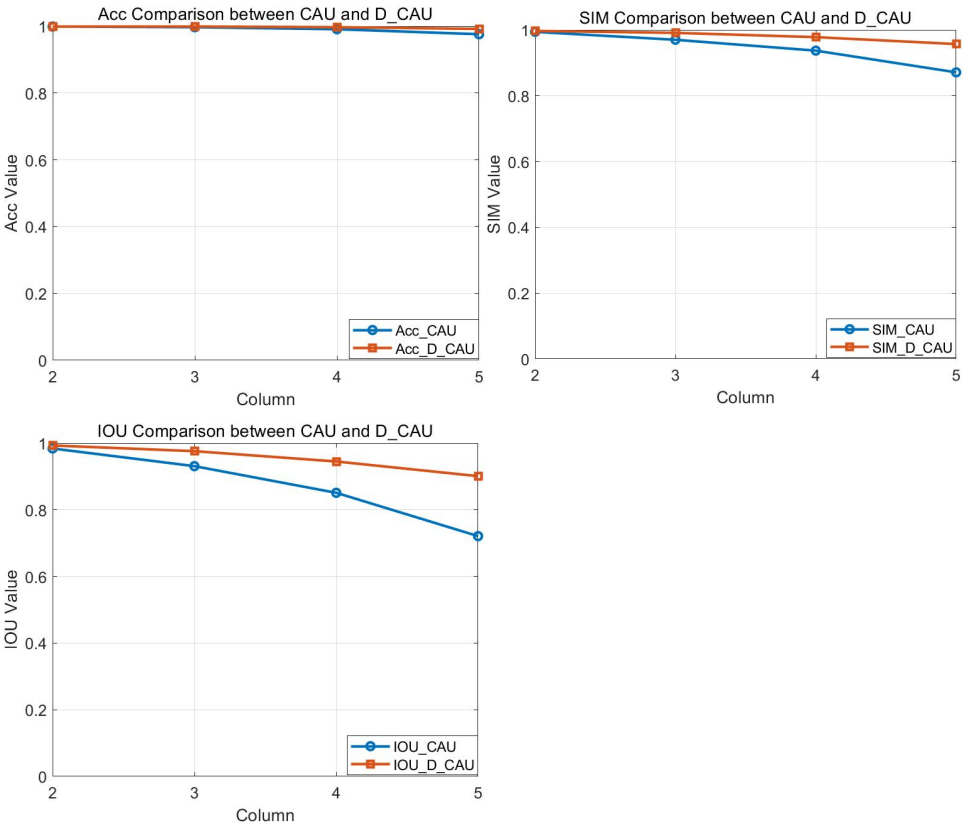
性能比较：



参数比较：



与 CAUNet 对比：



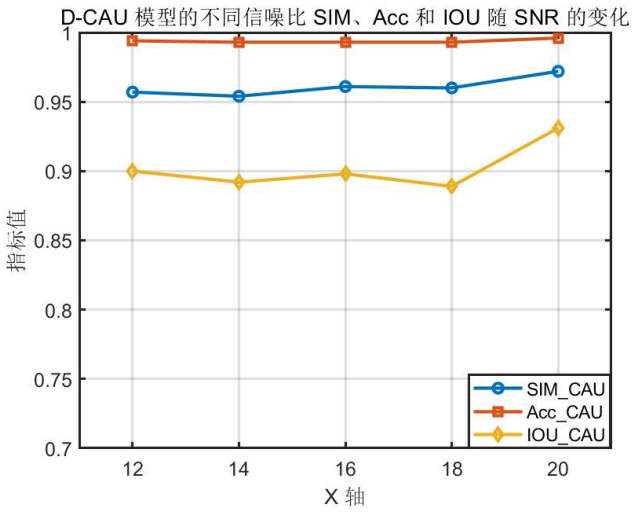
ACC	2	3	4	5
CAU	0.999	0.999	0.997	0.992
D-CAU	0.999	0.997	0.991	0.976
提升(%)	0	0.2	0.6	1.6

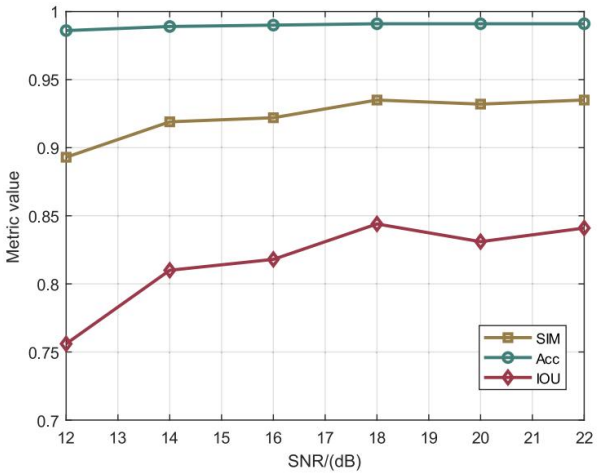
SIM	2	3	4	5
CAU	0.997	0.990	0.978	0.957
D-CAU	0.994	0.969	0.937	0.871
提升(%)	0.3	2.1	4.4	9.9

IOU	2	3	4	5
CAU	0.994	0.975	0.945	0.901
D-CAU	0.984	0.931	0.851	0.721
提升(%)	1.0	4.7	11.0	25.0

CAUNet 和 D-CAU 在不同信噪比下的性能

D-CAU　　CAUNet（目前没跑，用论文中数据）





IOU 目前最差为 0.889。

可以看出，在不同信噪比下，其模型的识别率都高于 CAUNet。

附结果图：

D-CAU

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9717178344726562
acc: 0.995626413690476
iou: 0.9307803386032693
```

CAU

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9130073189735413
acc: 0.986514384920635
iou: 0.8007820603285815
```

D-CAU 在**发射器**为 2，3，4，5 情况下的结果

```
sim: 0.9974414706230164
acc: 0.9998138392857143
iou: 0.9938004985461901
```

```
sim: 0.9903396368026733
acc: 0.9988504464285715
iou: 0.97500272916797
```

```
sim: 0.9784048199653625
acc: 0.9966319375
iou: 0.9446590987599476
```

```
sim: 0.9574646353721619
acc: 0.992358125
iou: 0.9008952695144445
```

CAUNet 在**发射器**为 2，3，4，5 情况下的结果

```
sim: 0.993655264377594
acc: 0.9995265624999999
iou: 0.9841859024127244
```

```
sim: 0.9693989753723145
acc: 0.9967245535714284
iou: 0.930633848880513
```

```
sim: 0.9367785453796387
acc: 0.9905092500000001
iou: 0.8512789395195278
```

```
sim: 0.8713444471359253
acc: 0.9763553125000003
iou: 0.7212486832694543
```

可以看出在多发射器的情况下模型的性能领先更明显。

CAUNet 参数

```
FLOPs: 2756.657288
Params: 2.755883
```

D-CAU 参数

```
FLOPs: 1327.07932
Params: 1.390019
```

不同 SNR

D-CAU

SNR12

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9574299454689026
acc: 0.9936514880952382
iou: 0.899551197141492
```

SNR14

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9543184638023376
acc: 0.9930695436507937
iou: 0.8916755518433845
```

SNR16

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9607882499694824
acc: 0.9934440228174605
iou: 0.8983459840795672
```

SNR 18

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9619441628456116
acc: 0.9941798363095236
iou: 0.9084416563833342
```

SNR 20

```
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([16, 1, 8000])
torch.Size([8, 1, 8000])
sim: 0.9717178344726562
acc: 0.995626413690476
iou: 0.9307803386032693
```