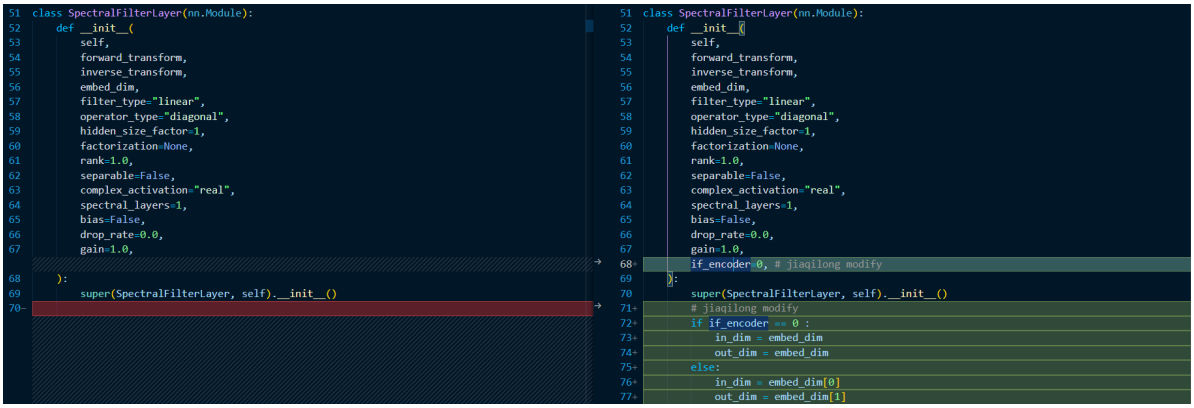
sfno 修改

models/networks/sfnonet.py

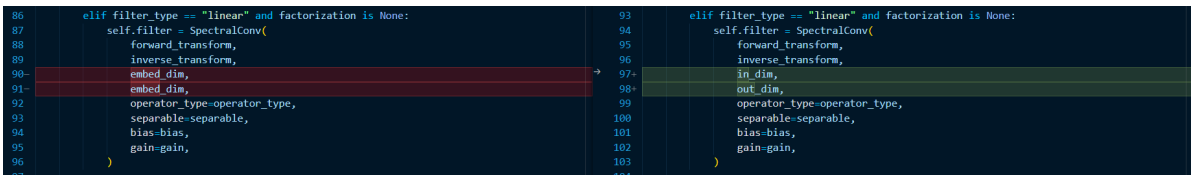
class SpectralFilterLayer

由于encoder中输入维度和输出维度不一样，故增加一个参数if_encoder，对该类作是否属于encoder的分类，并在spectral conv中修改输入输出维度



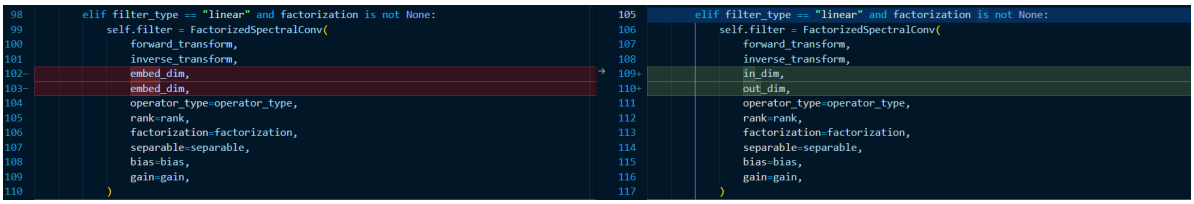
```
class SpectralFilterLayer(nn.Module):
    def __init__(
        self,
        forward_transform,
        inverse_transform,
        embed_dim,
        filter_type="linear",
        operator_type="diagonal",
        hidden_size_factor=1,
        factorization=None,
        rank=1.0,
        separable=False,
        complex_activation="real",
        spectral_layers=1,
        bias=False,
        drop_rate=0.0,
        gain=1.0,
    ):
        super(SpectralFilterLayer, self).__init__()
```

```
class SpectralFilterLayer(nn.Module):
    def __init__(
        self,
        forward_transform,
        inverse_transform,
        embed_dim,
        filter_type="linear",
        operator_type="diagonal",
        hidden_size_factor=1,
        factorization=None,
        rank=1.0,
        separable=False,
        complex_activation="real",
        spectral_layers=1,
        bias=False,
        drop_rate=0.0,
        gain=1.0,
        if_encoder=0, # jiaqilong modify
    ):
        super(SpectralFilterLayer, self).__init__()
        # jiaqilong modify
        if if_encoder == 0 :
            in_dim = embed_dim
            out_dim = embed_dim
        else:
            in_dim = embed_dim[0]
            out_dim = embed_dim[1]
```



```
        elif filter_type == "linear" and factorization is None:
            self.filter = SpectralConv(
                forward_transform,
                inverse_transform,
                embed_dim,
                embed_dim,
                operator_type=operator_type,
                separable=separable,
                bias=bias,
                gain=gain,
            )
```

```
        elif filter_type == "linear" and factorization is None:
            self.filter = SpectralConv(
                forward_transform,
                inverse_transform,
                in_dim,
                out_dim,
                operator_type=operator_type,
                separable=separable,
                bias=bias,
                gain=gain,
            )
```

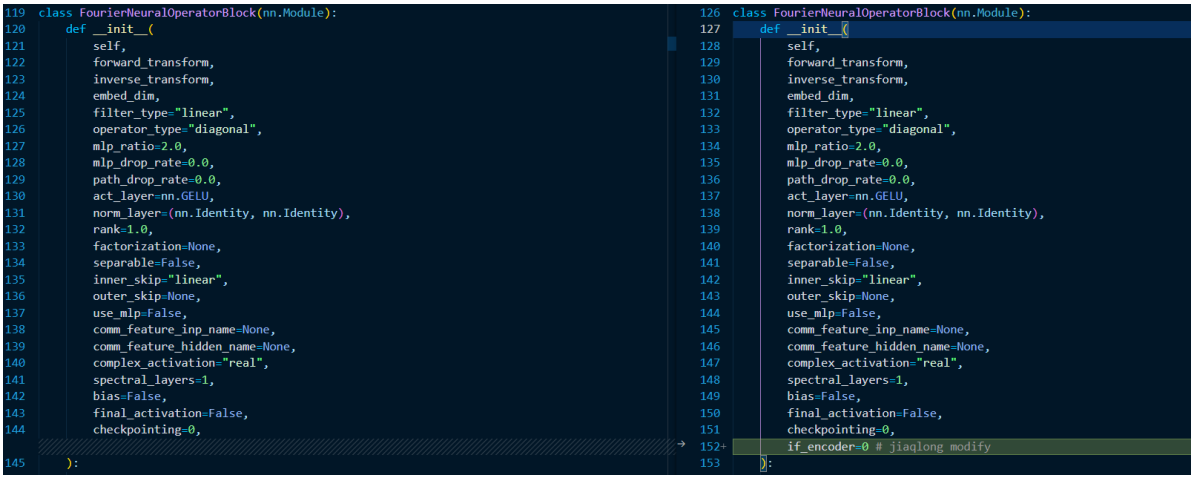


```
        elif filter_type == "linear" and factorization is not None:
            self.filter = FactorizedSpectralConv(
                forward_transform,
                inverse_transform,
                embed_dim,
                embed_dim,
                operator_type=operator_type,
                rank=rank,
                factorization=factorization,
                separable=separable,
                bias=bias,
                gain=gain,
            )
```

```
        elif filter_type == "linear" and factorization is not None:
            self.filter = FactorizedSpectralConv(
                forward_transform,
                inverse_transform,
                in_dim,
                out_dim,
                operator_type=operator_type,
                rank=rank,
                factorization=factorization,
                separable=separable,
                bias=bias,
                gain=gain,
            )
```

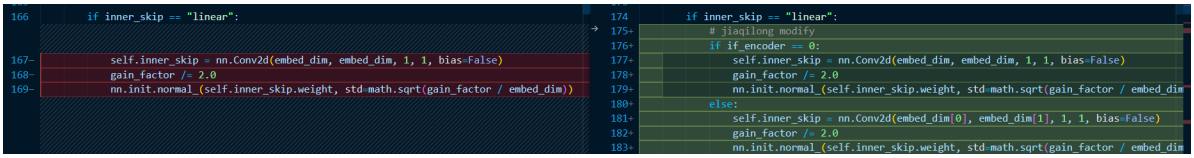
class FourierNeuralOperatorBlock

修改原因和方式同上。



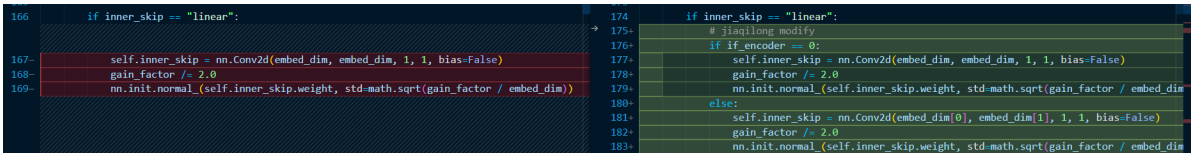
```
class FourierNeuralOperatorBlock(nn.Module):
    def __init__(
        self,
        forward_transform,
        inverse_transform,
        embed_dim,
        filter_type="linear",
        operator_type="diagonal",
        mlp_ratio=2.0,
        mlp_drop_rate=0.0,
        path_drop_rate=0.0,
        act_layer=nn.GELU,
        norm_layer=(nn.Identity, nn.Identity),
        rank=1.0,
        factorization=None,
        separable=False,
        inner_skip="linear",
        outer_skip=None,
        use_mlp=False,
        comm_feature_inp_name=None,
        comm_feature_hidden_name=None,
        complex_activation="real",
        spectral_layers=1,
        bias=False,
        final_activation=False,
        checkpointing=0,
    ):
```

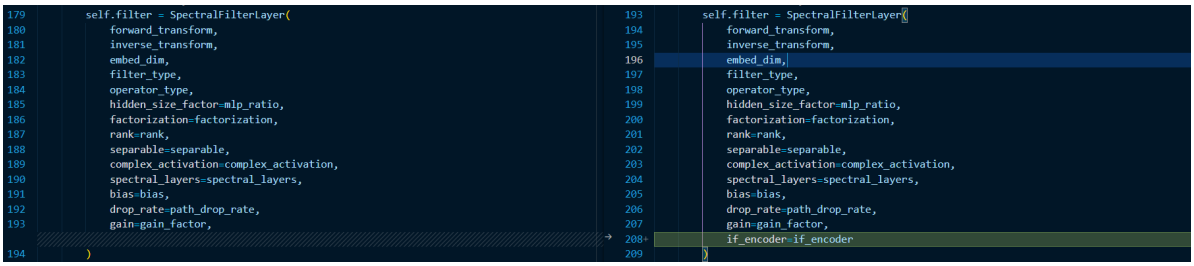
```
class FourierNeuralOperatorBlock(nn.Module):
    def __init__(
        self,
        forward_transform,
        inverse_transform,
        embed_dim,
        filter_type="linear",
        operator_type="diagonal",
        mlp_ratio=2.0,
        mlp_drop_rate=0.0,
        path_drop_rate=0.0,
        act_layer=nn.GELU,
        norm_layer=(nn.Identity, nn.Identity),
        rank=1.0,
        factorization=None,
        separable=False,
        inner_skip="linear",
        outer_skip=None,
        use_mlp=False,
        comm_feature_inp_name=None,
        comm_feature_hidden_name=None,
        complex_activation="real",
        spectral_layers=1,
        bias=False,
        final_activation=False,
        checkpointing=0,
        if_encoder=0 # jiaqlong modify
    ):
```

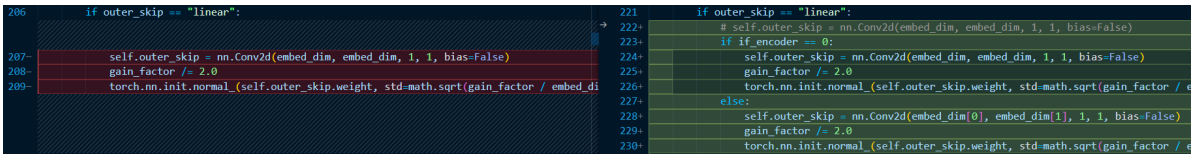


```
        if inner_skip == "linear":
            self.inner_skip = nn.Conv2d(embed_dim, embed_dim, 1, 1, bias=False)
            gain_factor /= 2.0
            nn.init.normal_(self.inner_skip.weight, std=math.sqrt(gain_factor / embed_dim))
```

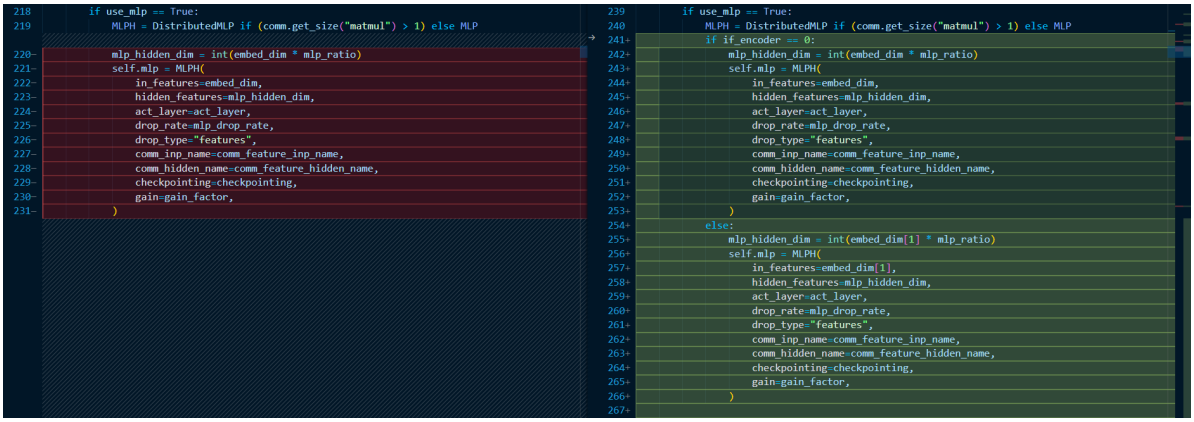
```
    if inner_skip == "linear":
        # jiaqilong modify
        if if_encoder == 0:
            self.inner_skip = nn.Conv2d(embed_dim, embed_dim, 1, 1, bias=False)
            gain_factor /= 2.0
            nn.init.normal_(self.inner_skip.weight, std=math.sqrt(gain_factor / embed_dim
        else:
            self.inner_skip = nn.Conv2d(embed_dim[0], embed_dim[1], 1, 1, bias=False)
            gain_factor /= 2.0
            nn.init.normal_(self.inner_skip.weight, std=math.sqrt(gain_factor / embed_dim
```







230行 参数初始化，sqrt中的分母选择我并无依据，还需斧正。



718行起，到800行，class Encoder_sfno，为单个变量的encoder。

801行起，到890行，class EncoderWrapper_sfno，为整个encoder，包含

892行起，到文件末，class SphericalFourierNeuralOperatorNetSfnoEnc，为替换encoder后的模型。