9 Appendix

In this appendix, we provide the pseudo code implementation for SAC variants discussed in Section 4.2, including SAC-S, SAC-IS, SAC-SK and SAC-ISK, and the figure illustration of prior works, including SE [14], CBAM [51], CAM [58] and PAC [42].


```python
def SAC_S(input_feature, coordi_map):
    # Note: Pseudo code for SAC-S.
    attention_map = Conv_attention7x7(coordi_map)  # (N, 1, H, W)
    input_feature = input_feature * attention_map  # (N, C, H, W)
    feature = Conv_feature3x3(input_feature)  # (N, C, H, W)
    output_feature = feature + input_feature  # (N, C, H, W)
    return output_feature  # (N, C, H, W)

def SAC_IS(input_feature, coordi_map):
    # Note: Pseudo code for SAC-IS.
    attention_map = Conv_attention7x7(coordi_map)  # (N, C, H, W)
    input_feature = input_feature * attention_map  # (N, C, H, W)
    feature = Conv_feature3x3(input_feature)  # (N, C, H, W)
    output_feature = feature + input_feature  # (N, C, H, W)
    return output_feature  # (N, C, H, W)

def SAC_SK(input_feature, coordi_map):
    # Note: Pseudo code for SAC-SK.
    unfold_feature = unfold(input_feature, kernel_size=K,
                             padding=K//2)  # (N, C*K*K, H, W)
    attention_map = Conv_attention7x7(coordi_map)  # (N, K*K, H, W)
    attention_map = attention_map.repeat(1, 1, 1, 1)  # (N, C*K*K, H, W)
    input_feature = input_feature * attention_map  # (N, C*K*K, H, W)
    feature = Conv_feature1x1(input_feature)  # (N, C, H, W)
    feature = Conv_feature3x3(feature)  # (N, C, H, W)
    output_feature = feature + input_feature  # (N, C, H, W)
    return output_feature  # (N, C, H, W)

def SAC_ISK(input_feature, coordi_map):
    # Note: Pseudo code for SAC-ISK.
    unfold_feature = unfold(input_feature, kernel_size=K,
                             padding=K//2)  # (N, C*K*K, H, W)
    attention_map = Conv_attention7x7(coordi_map)  # (N, C*K*K, H, W)
    input_feature = input_feature * attention_map  # (N, C*K*K, H, W)
    feature = Conv_feature1x1(input_feature)  # (N, C, H, W)
    return output_feature  # (N, C, H, W)
```

SqueezeSegV3 19
Fig. 5: Variants of spatially-adaptive convolution from previous work.

feature = Conv_feature3x3(feature) # (N, C, H, W)
output_feature = feature+input_feature # (N, C, H, W)
return output_feature # (N, C, H, W)