LD-Pruner: Efficient Pruning of Latent Diffusion Models using Task-Agnostic Insights

Supplementary Material

1. Operator Modification

Fig. 1 shows the code used to replace the operators with different input and output number of channels and/or spatial dimensions.

2. Hyper-parameters

In Tab. 1, we give the hyper-parameters used for our experiments on Text-to-Image generation (T2I), Unconditional Image Generation (UIG) and Unconditional Audio Generation (UAG). ‘feat. KD coef.’ and ‘out KD coef.’ refer to the coefficient used in the knowledge-distillation loss applied at the feature-level and output-level, respectively.

3. Scoring Metric Composition: more results

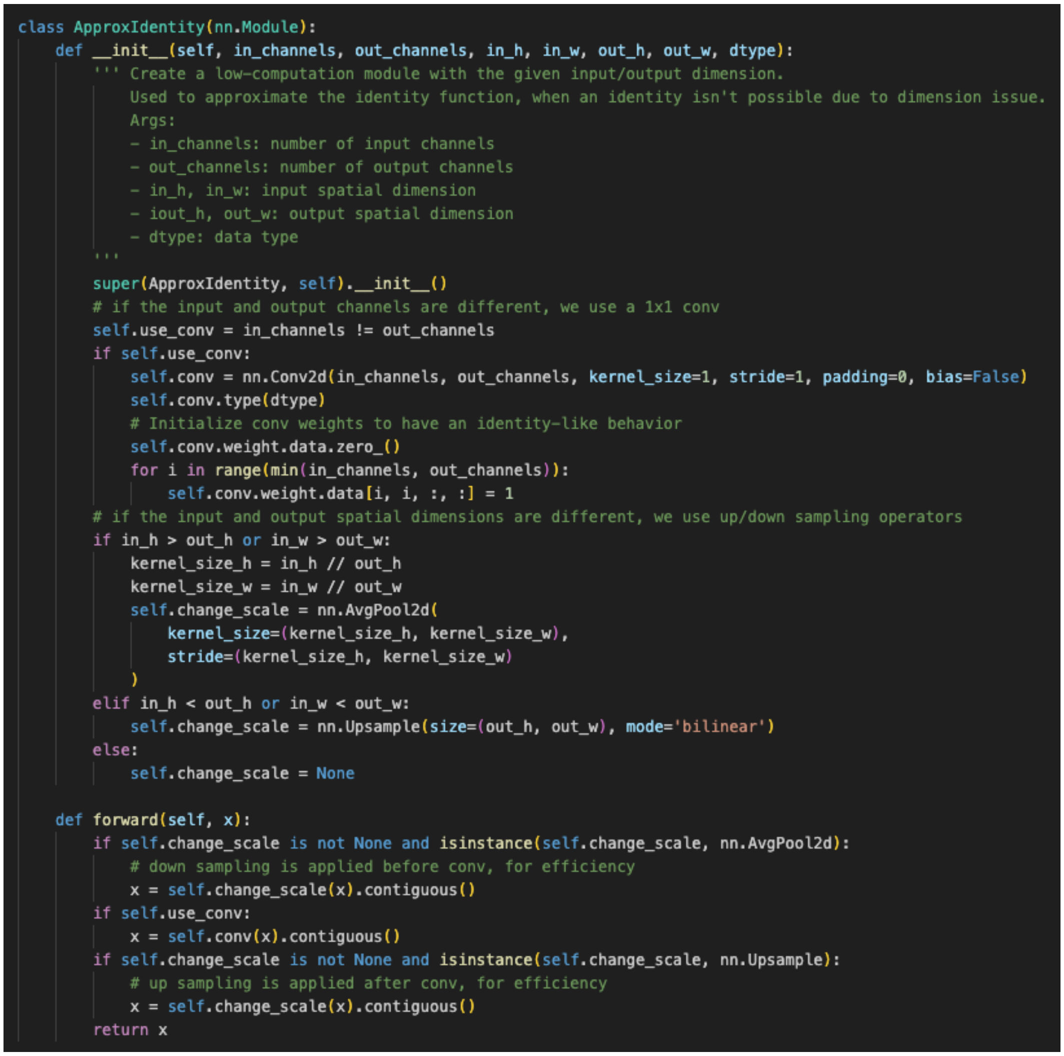
In Fig. 2, we provide additional qualitative comparison examples for the scoring metric composition comparison on T2I with SD. The results are without finetuning.

4. Importance Score vs Block

In Figs. 3, 4 and 5 , we show the relative importance of each operator in the Unet of SD, LDM-4 and AudioDiffusion, respectively. These scores provide insights into the relative contribution of individual operators to the overall models. The relationship between the BasicTransformerBlock, Transformer2D and Attention operators is illustrated in Fig. 9.

5. Visualization of Modified Operator Distribution

In Figs. 6, 7, and 8, we visualize the distribution by type of the modified operators within the Unet structures of SD, LDM-4, and AudioDiffusion, respectively. The operators are categorized by their type and the block they inhabit. The relationship between the BasicTransformerBlock, Transformer2D and Attention operators is illustrated in Fig. 9.



```
class ApproxIdentity(nn.Module):
    def __init__(self, in_channels, out_channels, in_h, in_w, out_h, out_w, dtype):
        ''' Create a low-computation module with the given input/output dimension.
            Used to approximate the identity function, when an identity isn't possible due to dimension issue.
            Args:
            - in_channels: number of input channels
            - out_channels: number of output channels
            - in_h, in_w: input spatial dimension
            - iout_h, out_w: output spatial dimension
            - dtype: data type
        '''
        super(ApproxIdentity, self).__init__()
        # if the input and output channels are different, we use a 1x1 conv
        self.use_conv = in_channels != out_channels
        if self.use_conv:
            self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1, padding=0, bias=False)
            self.conv.type(dtype)
            # Initialize conv weights to have an identity-like behavior
            self.conv.weight.data.zero_()
            for i in range(min(in_channels, out_channels)):
                self.conv.weight.data[i, i, :, :] = 1
        # if the input and output spatial dimensions are different, we use up/down sampling operators
        if in_h > out_h or in_w > out_w:
            kernel_size_h = in_h // out_h
            kernel_size_w = in_w // out_w
            self.change_scale = nn.AvgPool2d(
                kernel_size=(kernel_size_h, kernel_size_w),
                stride=(kernel_size_h, kernel_size_w)
            )
        elif in_h < out_h or in_w < out_w:
            self.change_scale = nn.Upsample(size=(out_h, out_w), mode='bilinear')
        else:
            self.change_scale = None

    def forward(self, x):
        if self.change_scale is not None and isinstance(self.change_scale, nn.AvgPool2d):
            # down sampling is applied before conv, for efficiency
            x = self.change_scale(x).contiguous()
        if self.use_conv:
            x = self.conv(x).contiguous()
        if self.change_scale is not None and isinstance(self.change_scale, nn.Upsample):
            # up sampling is applied after conv, for efficiency
            x = self.change_scale(x).contiguous()
        return x
```

Figure 1. Code used to replace the operators with different input/output dimension (typically, a convolution).

Task	lr	batch size	gradient accumulation	iterations	feat. KD coef.	out KD coef.
T2I Generation	$3e^{-5}$	64	4	50,000	0.7	0.7
UIG	$5e^{-6}$	32	4	50,000	300	300
UAG	$1e^{-4}$	64	2	12,000	10	10

Table 1. Hyper-parameters used in our experiments. We used the same hyper-parameters for all compression.

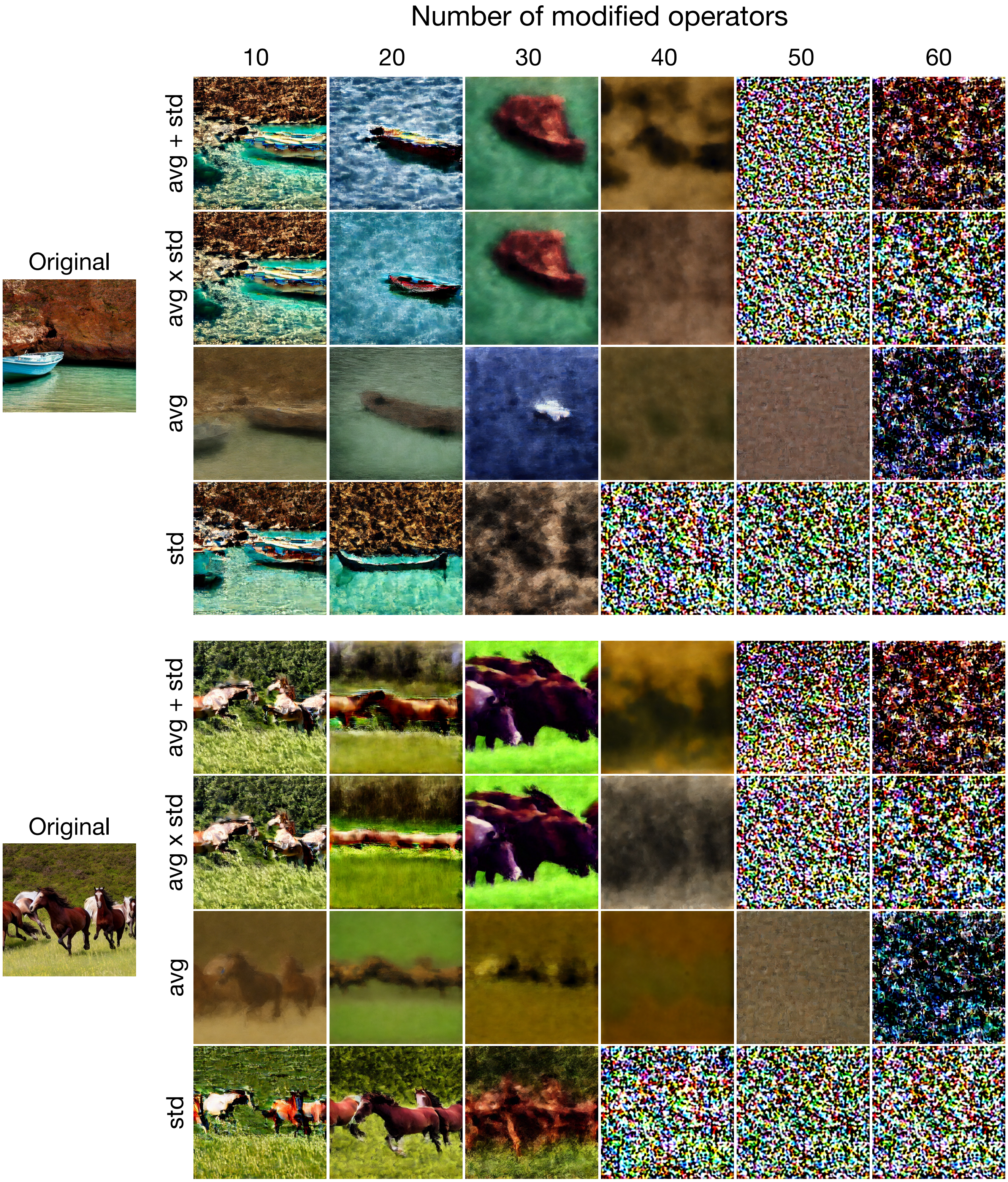


Figure 2. Qualitative comparison of the impact of various combination methods for average and standard deviation in our proposed scoring metric, with SD. The results are without finetuning. Prompts: "boat on a beautiful sea", "group of wild horses galloping through a meadow".

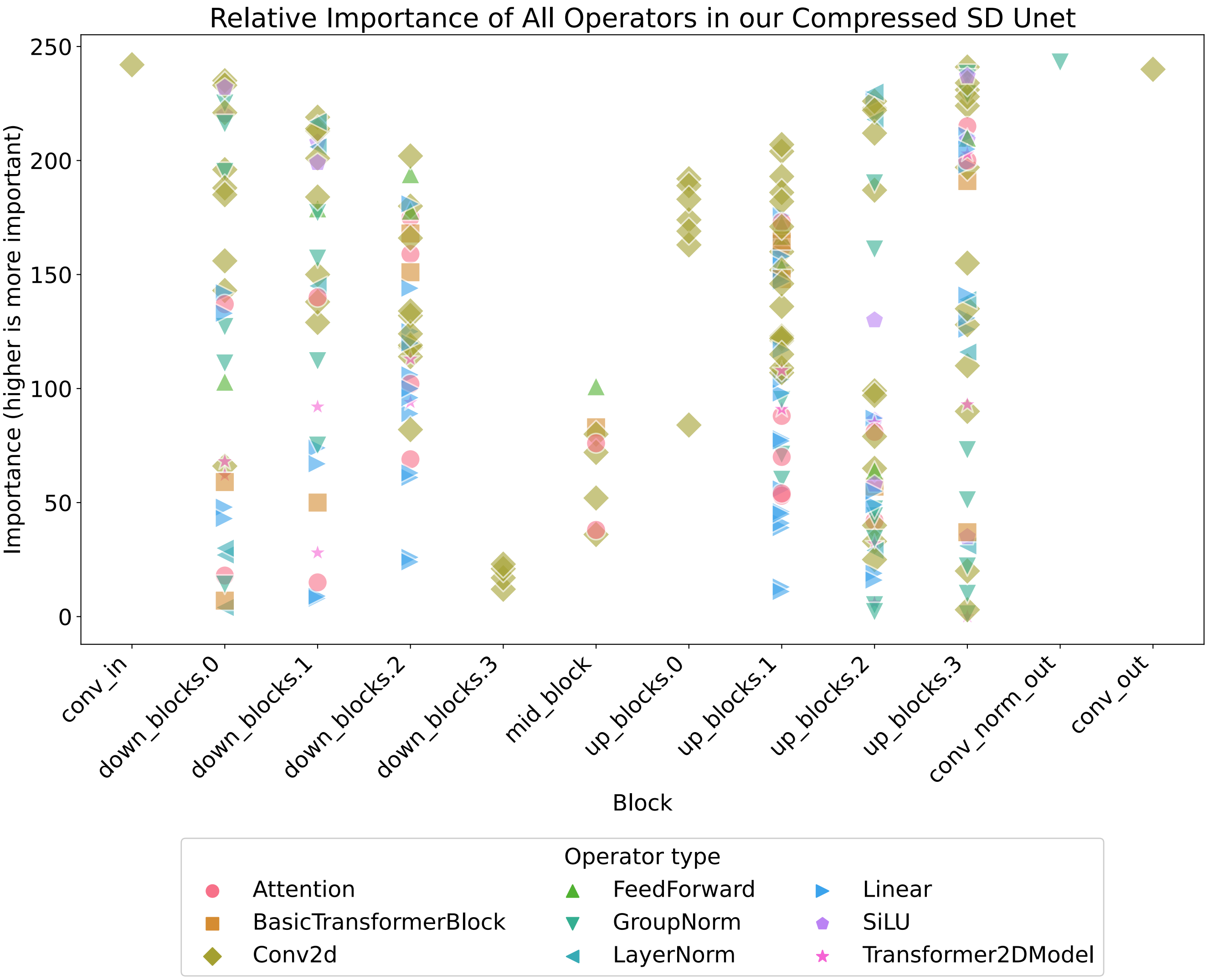


Figure 3. Importance ranking of the operators in SD Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

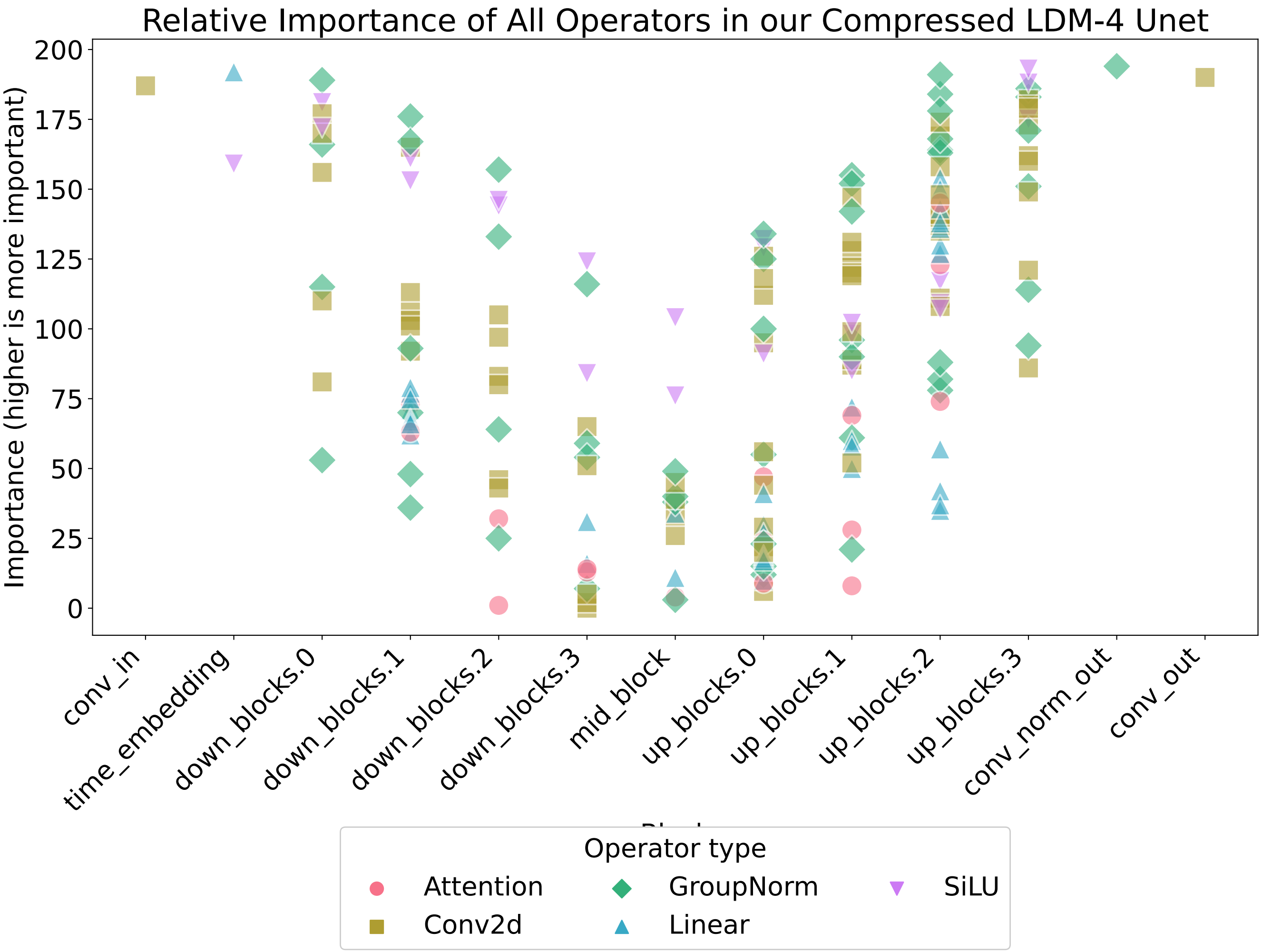


Figure 4. Importance ranking of the operators in LDM-4 Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

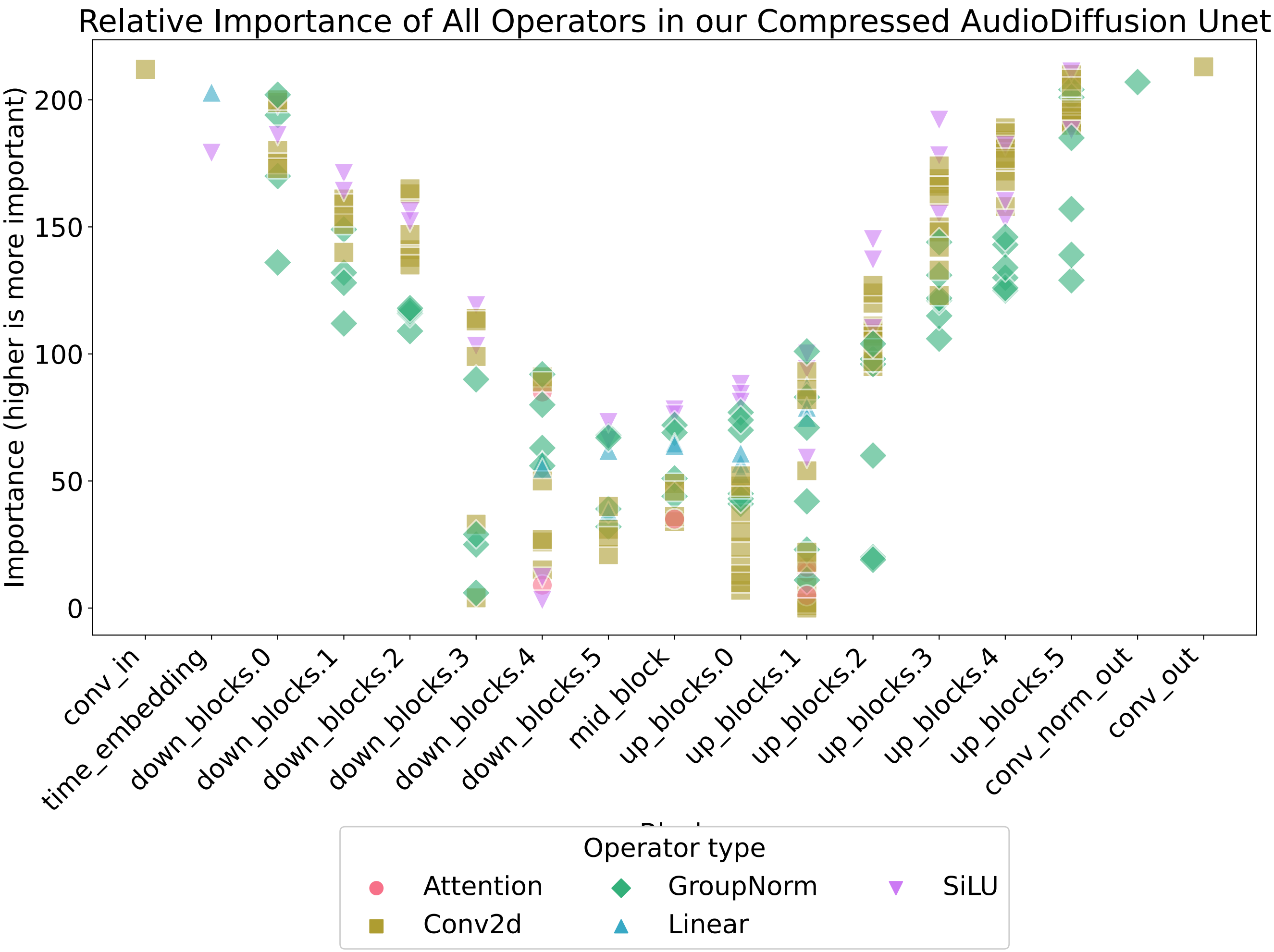


Figure 5. Importance ranking of the operators in AudioDiffusion Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

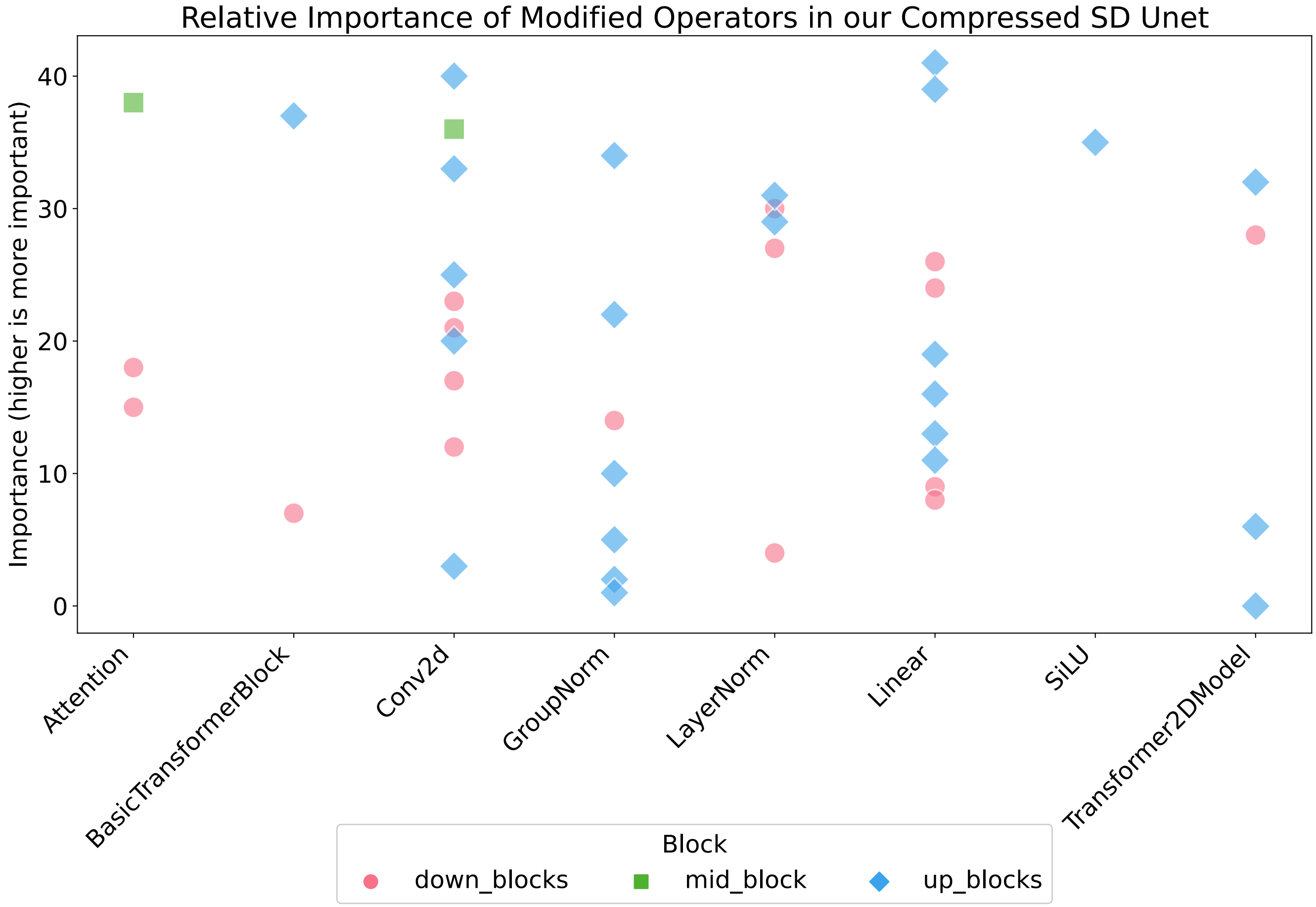


Figure 6. Importance ranking of the modified operators in SD Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

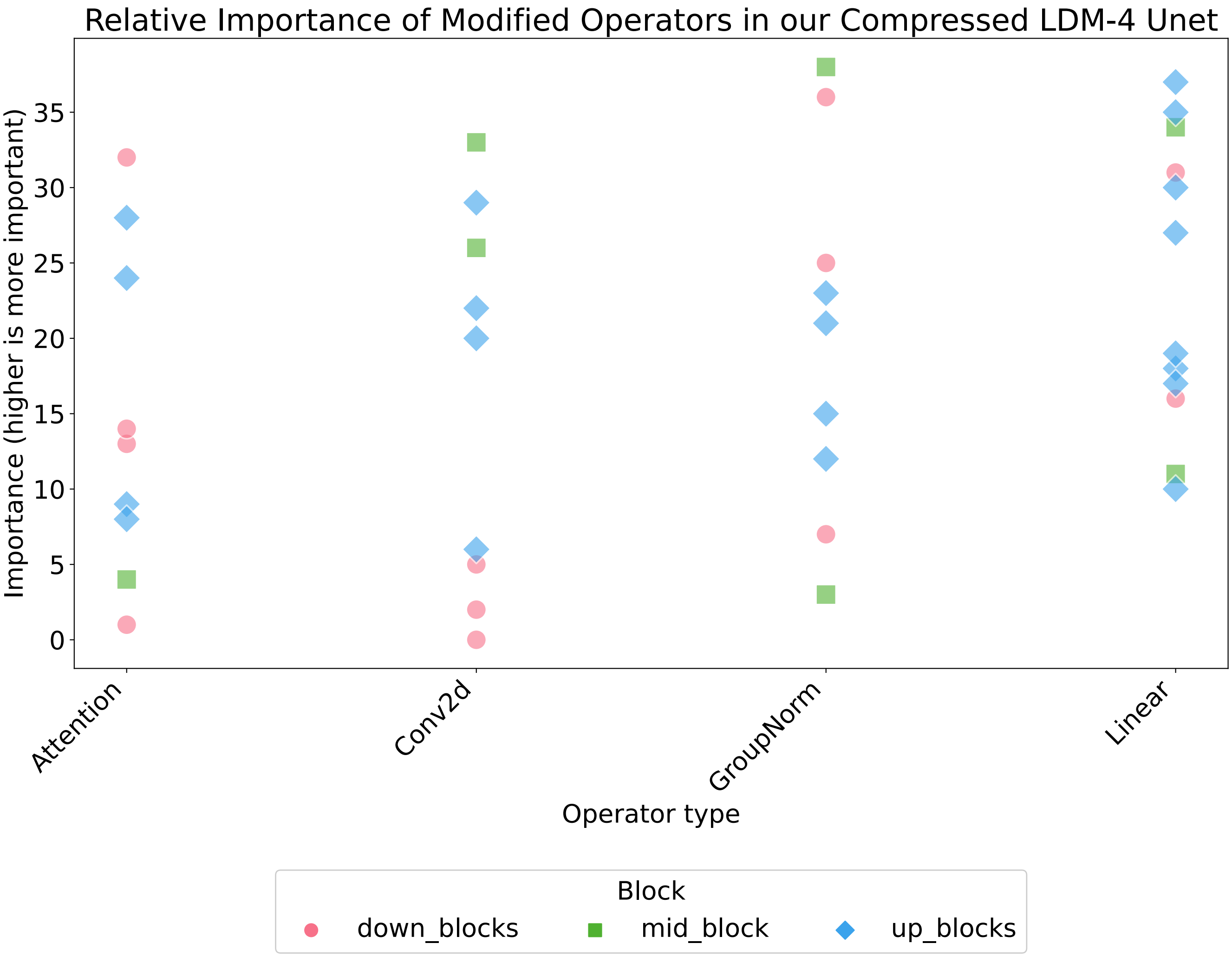


Figure 7. Importance ranking of the modified operators in LDM-4 Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

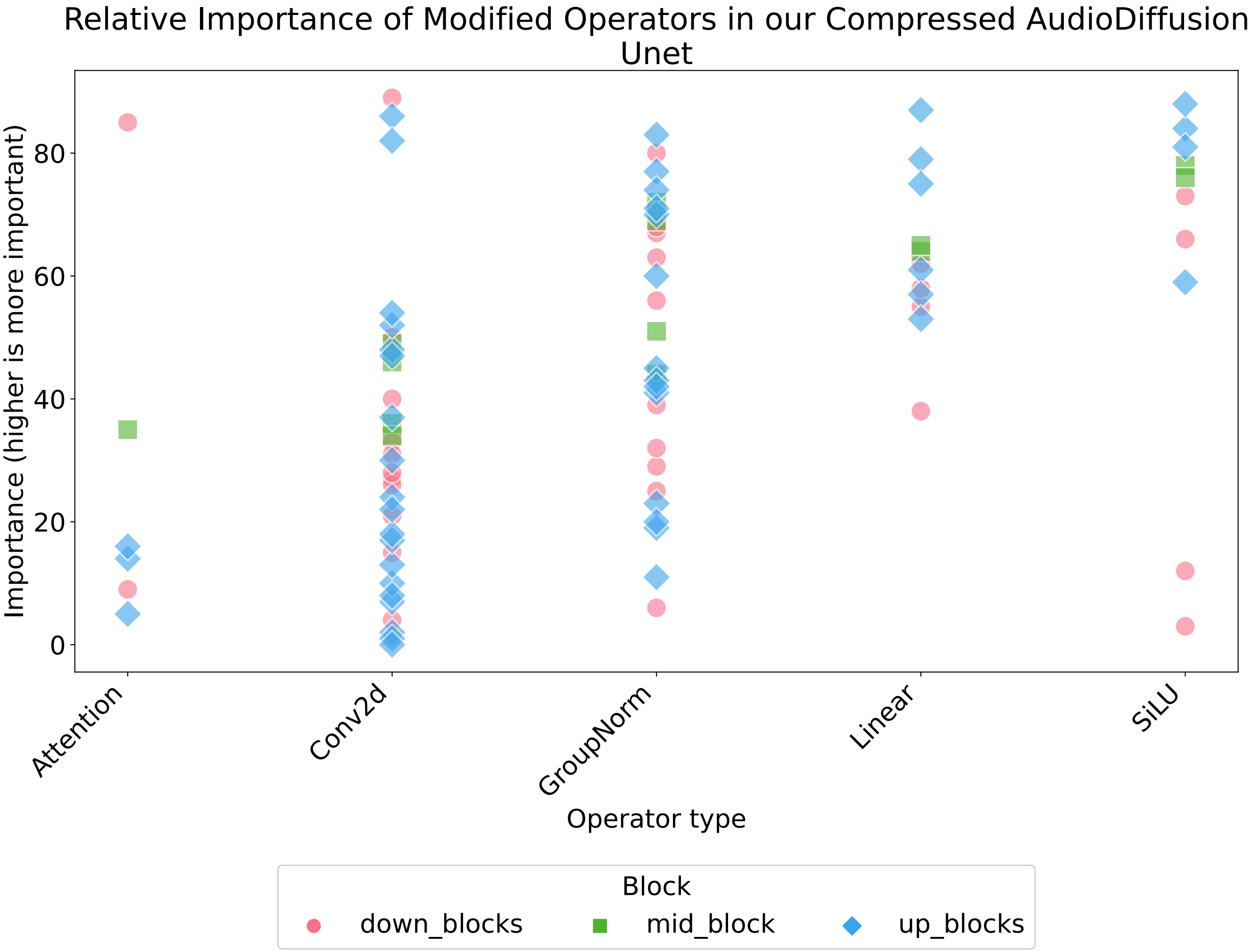


Figure 8. Importance ranking of the modified operators in AudioDiffusion Unet, as determined by LDPruner. Lower values indicate less importance to the Unet output.

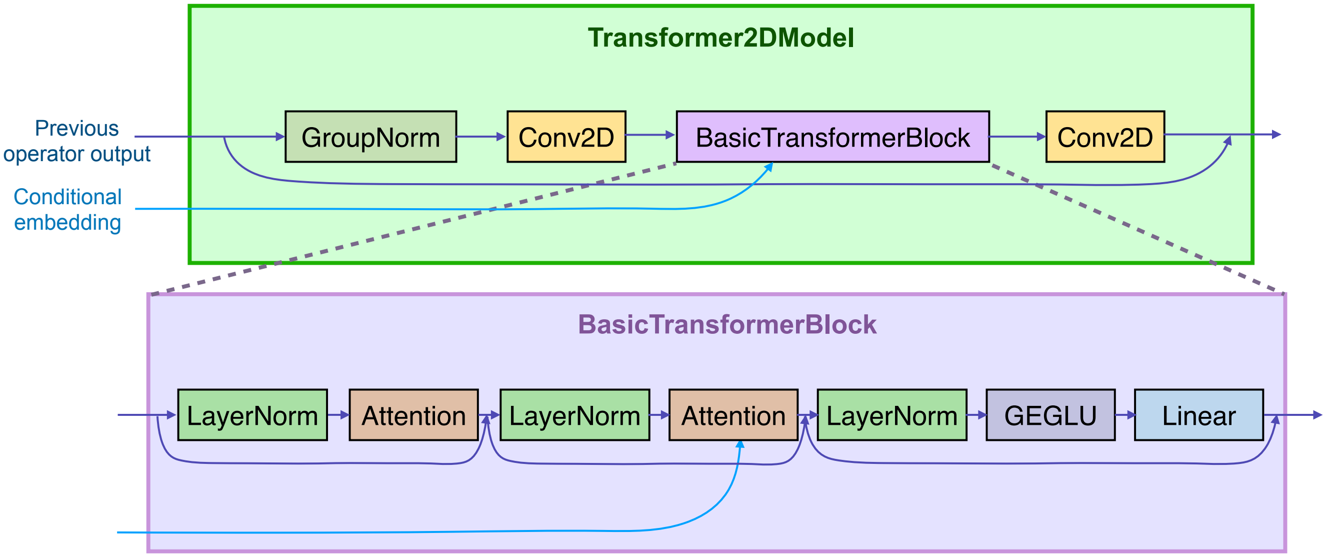


Figure 9. A simplified view of the Transformer2DModel operator, illustrating the relationship between the Transformer2DModel, BasicTransformerBlock, and Attention operators.