7 Supplementary Material

7.1 Converting Parallel Pixal Difference Module (PPDM) to Vanilla Convolution

The goal of the conversion is to make PPDM as fast and memory efficient as the vanilla convolution. The process can be finished in two steps: firstly, we convert the pixel difference convolutions into vanilla convolutions, and secondly, we can obtain a 5×5 vanilla convolution by reparameterizing the multi-branch structure, as illustrated in Fig. 5.

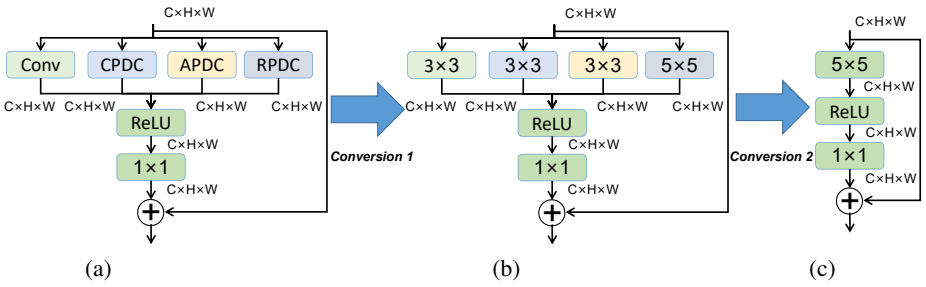

Fig. 5. Convertion of PPDM.

Next, we describe the implementation process in detail.

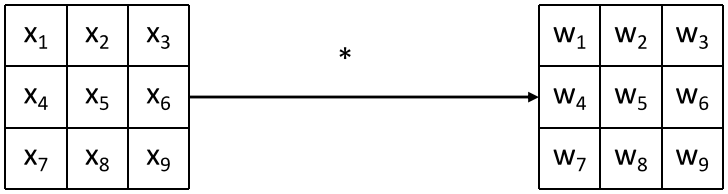

Fig. 6. The process of convolution.

As shown in Fig. 6, vanilla convolution is achieved by:

$$\begin{aligned} \boldsymbol{Y} &= Conv(\boldsymbol{X}) \\ &= x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 \\ &\quad + x_4 \cdot w_4 + x_5 \cdot w_5 + x_6 \cdot w_6 \\ &\quad + x_7 \cdot w_7 + x_8 \cdot w_8 + x_9 \cdot w_9 \end{aligned} \tag{2}$$

Where $\boldsymbol{X}$, $\boldsymbol{Y}$ represent the feature matrices of the input and output. x_i is the single value in $\boldsymbol{X}$ and w_i is learnable parameter in convolutional kernel. Unless otherwise stated, these letters have the same meaning in the following formula, and we will not repeat them.

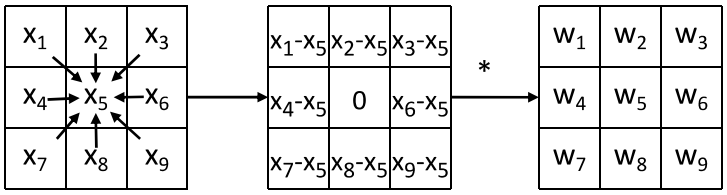

Fig. 7. The process of CPDC.

As shown in Fig. 7, For CPDC:

$$
\begin{aligned}
\boldsymbol{Y} &= CPDC(\boldsymbol{X}) \\
&= (x_1 - x_5) \cdot w_1 + (x_2 - x_5) \cdot w_2 + (x_3 - x_5) \cdot w_3 \\
&\quad + (x_4 - x_5) \cdot w_4 + (x_5 - x_5) \cdot w_5 + (x_6 - x_5) \cdot w_6 \\
&\quad + (x_7 - x_5) \cdot w_7 + (x_8 - x_5) \cdot w_8 + (x_9 - x_5) \cdot w_9 \\
&= x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 \\
&\quad + x_4 \cdot w_4 + x_6 \cdot w_6 \\
&\quad + x_7 \cdot w_7 + x_8 \cdot w_8 + x_9 \cdot w_9 \\
&\quad + x_5 \cdot (- \sum_{i=\{1,2,3,4,6,7,8,9\}} \cdot w_i)
\end{aligned}
\tag{3}
$$

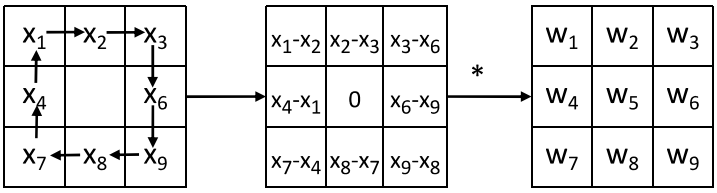

Fig. 8. The process of APDC.

As shown in Fig. 8, For APDC:

$$
\begin{aligned}
\boldsymbol{Y} &= APDC(\boldsymbol{X}) \\
&= (x_1 - x_2) \cdot w_1 + (x_2 - x_3) \cdot w_2 + (x_3 - x_6) \cdot w_3 \\
&\quad + (x_4 - x_1) \cdot w_4 + (x_6 - x_9) \cdot w_6 \\
&\quad + (x_7 - x_4) \cdot w_7 + (x_8 - x_7) \cdot w_8 + (x_9 - x_8) \cdot w_9 \\
&= x_1 \cdot (w_1 - w_4) + x_2 \cdot (w_2 - w_1) + x_3 \cdot (w_3 - w_2) \\
&\quad + x_4 \cdot (w_4 - w_7) + x_6 \cdot (w_6 - w_3) \\
&\quad + x_7 \cdot (w_7 - w_8) + x_8 \cdot (w_8 - w_9) + x_9 \cdot (w_9 - w_6)
\end{aligned}
\tag{4}
$$

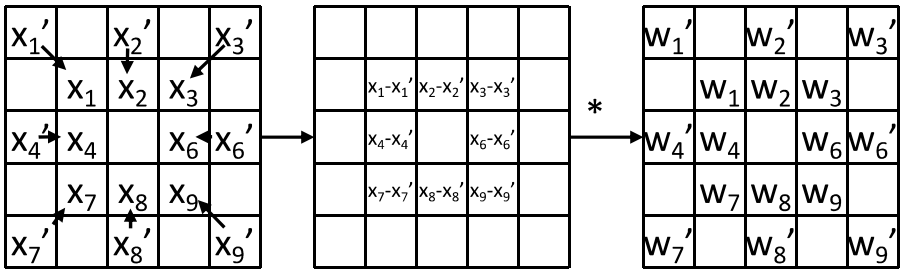

Fig. 9. The process of RPDC.

As shown in Fig. 9, For RPDC:

$$
\begin{aligned}
\boldsymbol{Y} &= RPDC(\boldsymbol{X}) \\
&= (x'_1 - x_1) \cdot w'_1 + (x'_2 - x_2) \cdot w'_2 + (x'_3 - x_3) \cdot w'_3 \\
&\quad + (x'_4 - x_4) \cdot w'_4 + (x'_6 - x_6) \cdot w'_6 \\
&\quad + (x'_7 - x_7) \cdot w'_7 + (x'_8 - x_8) \cdot w'_8 + (x'_9 - x_9) \cdot w'_9;; \\
&= x'_1 \cdot w'_1 + x'_2 \cdot w'_2 + x'_3 \cdot w'_3 \\
&\quad + x_1 \cdot (-w'_1) + x_2 \cdot (-w'_2) + x_3 \cdot (-w'_3) \\
&\quad + x'_4 \cdot w'_4 + x_4 \cdot (-w'_4) + x_6 \cdot (-w'_6) + x'_6 \cdot w'_6 \\
&\quad + x_7 \cdot (-w'_7) + x_8 \cdot (-w'_8) + x_9 \cdot (-w'_9) \\
&\quad + x'_7 \cdot w'_7 + x'_8 \cdot w'_8 + x'_9 \cdot w'_9
\end{aligned} \tag{5}
$$

By performing convolutional kernel calculations, multiple branch structures can be merged. The entire process of reparameterization is shown in Eq. 6. To distinguish the parameters of the four kinds of convolution kernels, we denote vanilla convolution, CPDC, APDC and RPDC by v, c, a, and r, respectively.

$$
\begin{aligned}
\boldsymbol{Y} &= AVE(Conv(\boldsymbol{X}), CPDC(\boldsymbol{X}), APDC(\boldsymbol{X}), RPDC(\boldsymbol{X})) \\
&= (Conv(\boldsymbol{X}) + CPDC(\boldsymbol{X}) + APDC(\boldsymbol{X}) + RPDC(\boldsymbol{X}))/4 \\
&= (x_1 \cdot w_1^v + x_2 \cdot w_2^v + x_3 \cdot w_3^v + x_4 \cdot w_4^v + x_5 \cdot w_5^v + x_6 \cdot w_6^v \\
&\quad + x_7 \cdot w_7^v + x_8 \cdot w_8^v + x_9 \cdot w_9^v \\
&\quad + x_1 \cdot w_1^c + x_2 \cdot w_2^c + x_3 \cdot w_3^c + x_4 \cdot w_4^c + x_6 \cdot w_6^c \\
&\quad + x_7 \cdot w_7^c + x_8 \cdot w_8^c + x_9 \cdot w_9^c + x_5 \cdot (- \sum_{i=\{1\sim4,6\sim9\}} \cdot w_i^c) \\
&\quad + x_1 \cdot (w_1^a - w_4^a) + x_2 \cdot (w_2^a - w_1^a) + x_3 \cdot (w_3^a - w_2^a) \\
&\quad + x_4 \cdot (w_4^a - w_7^a) + x_6 \cdot (w_6^a - w_3^a) \\
&\quad + x_7 \cdot (w_7^a - w_8^a) + x_8 \cdot (w_8^a - w_9^a) + x_9 \cdot (w_9^a - w_6^a) \\
&\quad + x_1' \cdot w_1^{r\prime} + x_2' \cdot w_2^{r\prime} + x_3' \cdot w_3^{r\prime} + x_1 \cdot (-w_1^{r\prime}) + x_2 \cdot (-w_2^{r\prime}) \\
&\quad + x_3 \cdot (-w_3^{r\prime}) + x_4' \cdot w_4^{r\prime} + x_4 \cdot (-w_4^{r\prime}) + x_6 \cdot (-w_6^{r\prime}) + x_6' \cdot w_6^{r\prime} \\
&\quad + x_7 \cdot (-w_7^{r\prime}) + x_8 \cdot (-w_8^{r\prime}) + x_9 \cdot (-w_9^{r\prime}) \\
&\quad + x_7' \cdot w_7^{r\prime} + x_8' \cdot w_8^{r\prime} + x_9' \cdot w_9^{r\prime})/4 \\
&= x_1' \cdot w_1^{r\prime}/4 + x_2' \cdot w_2^{r\prime}/4 + x_3' \cdot w_3^{r\prime}/4 + x_1 \cdot (w_1^v + w_1^c + w_1^a - w_4^a - w_1^{r\prime})/4 \\
&\quad + x_2 \cdot (w_2^v + w_2^c + w_2^a - w_1^a + w_2^a - w_1^a - w_2^{r\prime})/4 \\
&\quad + x_3 \cdot (w_3^v + w_3^c + w_3^a - w_2^a - w_3^{r\prime})/4 \\
&\quad + x_4' \cdot w_4^{r\prime}/4 + x_4 \cdot (w_4^v + w_4^c + w_4^a - w_7^a - w_4^{r\prime})/4 \\
&\quad + x_5 \cdot (w_5^v - \sum_{i=\{1,2,3,4,6,7,8,9\}} \cdot w_i^c)/4 \\
&\quad + x_6 \cdot (w_6^v + w_6^c + w_6^a - w_3^a - w_6^{r\prime})/4 + x_6' \cdot w_6^{r\prime}/4 \\
&\quad + x_7 \cdot (w_7^v + w_7^c + w_7^a - w_8^a - w_7^{r\prime})/4 + x_8 \cdot (w_8^v + w_8^c + w_8^a - w_9^a - w_8^{r\prime})/4 \\
&\quad + x_9 \cdot (w_9^v + w_9^c + w_9^a - w_6^a - w_9^{r\prime})/4 + x_7' \cdot w_7^{r\prime}/4 + x_8' \cdot w_8^{r\prime}/4 + x_9' \cdot w_9^{r\prime}/4 \\
&= \sum x_i \cdot \hat{w}_i
\end{aligned}
\tag{6}
$$

After these two steps of processing, the core of multi-branch structure can be converted to a single 5×5 vanilla convolution, as shown in Fig. 5 (c).

7.2 Visualization on BIPED Dataset

A qualitative comparison of PiDiNeXt's outputs with some other methods are shown in Fig 10. The results of PiDiNeXt-tiny-L have significantly less irrelevant cue than the results of PiDiNeXt-tiny-L, but due to the pursuit of efficiency, the quality of the generated edges still has great space for improvement. The visual effect of PiDiNeXt's results are much better and still competitive even compared to the results of the VGG-based method DeXiNed [18]. Precision-Recall curves of PiDiNeXt and PiDiNet on BIPED dataset is demonstrated by Fig. 11. On the whole, PiDiNeXt is better than PiDiNet.

And we can see that the solid blue line representing PiDiNeXt also exceeds the dashed red line representing PiDiNet, even though the numerical difference between the two in Table 2 is inconspicuous.

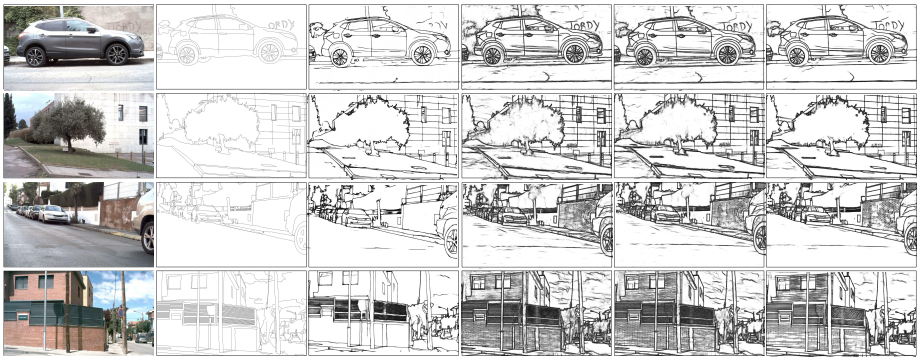

Fig. 10. A qualitative comparison of PiDiNeXt's outputs with some other methods.

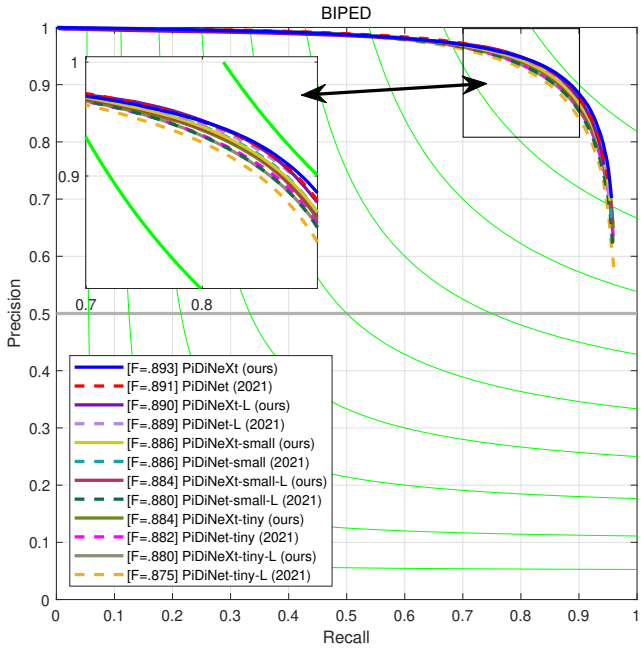

Fig. 11. Precision-Recall curves of PiDiNet and PiDiNeXt on BIPED dataset.