PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies

— Supplementary Material —

In this appendix, we provide additional content to complement the main manuscript:

• Appendix A: A detailed description of Tab. 7.
• Appendix B: Comparisons of training strategies for prior representative works and PointNeXt.
• Appendix C: Qualitative comparisons on S3DIS and ShapeNetPart.
• Appendix D: The architecture of PointNeXt for classification.
• Appendix E: Societal impact.

A Detailed Description for Manuscript Tab. 7

Naive width scaling increases the channel size of PointNet++ from 32 to 256 to match the throughput of the baseline model, PointNeXt-XL. Naive depth scaling refers to appending more SA blocks ($B = (3, 6, 3, 3)$, the same as PointNeXt-XL) in PointNet++. Furthermore, naive compound scaling doubles the width of naive depth scaled model to the same as PointNeXt-XL ($C = 64$). Compared to the PointNet++ trained with improved training strategies ($63.2\%$ mIoU, 186 ins./sec.), naive depth scaling ($63.4\%$ mIoU, 53 ins./sec.) and naive width scaling ($59.4\%$ mIoU, 43 ins./sec.) only lead to a large overhead in throughput with insignificant improvement in accuracy. In contrast, our proposed model scaling strategy achieves much higher performance than the naive scaling strategies while being much faster. This can be observed by comparing PointNeXt-XL ($70.5\%$ mIoU, 45 ins./sec.) to the naive compound scaled PointNet++ ($62.3\%$ mIoU, 24 ins./sec.).

B Training Strategies Comparison

In this section, we summarize the training strategies used in representative point-based methods such as DGCNN [8], KPConv [6], PointMLP [4], Point Transformer [10], Stratified Transformer [3], PointNet++ [5], and our PointNeXt on S3DIS [1] in Tab. I, on ScanObjectNN [7] in Tab. II, on ScanNet [2] in Tab. III, and on ShapeNetPart [9] in Tab. IV, respectively.

C Qualitative Results

We provide qualitative results of PointNeXt-XL for S3DIS (Fig. II) and PointNeXt-S ($C = 160$) for ShapeNetPart (Fig. III). The qualitative results of PointNet++ trained with the original training strategies are also included in the figures for comparison. On both datasets, PointNeXt produces predictions closer to the ground truth compared to PointNet++. More specifically, on S3DIS shown in (Fig. II), PointNeXt is able to segment hard classes, including doors (1st, 3rd, and 4th rows), clutter (1st and 3rd rows), chairs (2nd row), and the board (4th row), while PointNet++ fails to segment properly to some extent. On ShapeNetPart (Fig. III), PointNeXt precisely segments wings of an airplane (1st row), microphone of an earphone (2nd row), body of a motorbike (3rd row), fin of a rocket (4th row), and bearing of a skateboard (5th row).

D Classification Architecture

As illustrated in Fig. I, the classification architecture shares the same encoder as the segmentation one. The output features of the encoder are passed to a global pooling layer (i.e., global max-pooling) to acquire a global shape representation for classification. Note that the points are only downsampled by a factor of 2 in each stage, since the number of input points in classification tasks is usually small, e.g., 1024 or 2048 points.
### Table I: Training strategies used in different methods for S3DIS segmentation.

<table>
<thead>
<tr>
<th>Method</th>
<th>DGCNN</th>
<th>KPConv</th>
<th>PointTransformer</th>
<th>PointNet++</th>
<th>PointNeXt (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs</td>
<td>101</td>
<td>500</td>
<td>100</td>
<td>32</td>
<td>100</td>
</tr>
<tr>
<td>Batch size</td>
<td>12</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>SGD</td>
<td>SGD</td>
<td>AdamW</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-2}$</td>
<td>0.5</td>
<td>$1 \times 10^{-3}$</td>
<td>0.01</td>
</tr>
<tr>
<td>LR decay</td>
<td>step</td>
<td>step</td>
<td>multi step</td>
<td>step</td>
<td>cosine</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0</td>
<td>$10^{-8}$</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Label smoothing $\varepsilon$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>0.2</td>
</tr>
<tr>
<td>Entire scene as input</td>
<td>$\times$</td>
<td>$\times$</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
</tr>
<tr>
<td>Random rotation</td>
<td>$\times$</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Random scaling</td>
<td>$\times$</td>
<td>[0.8,1.2]</td>
<td>[0.9,1.1]</td>
<td>$\times$</td>
<td>[0.9,1.1]</td>
</tr>
<tr>
<td>Random translation</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>Random jittering</td>
<td>$\times$</td>
<td>0.001</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>Height appending</td>
<td>$\times$</td>
<td>✓</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>Color drop</td>
<td>$\times$</td>
<td>0.2</td>
<td>$\times$</td>
<td>$\times$</td>
<td>0.2</td>
</tr>
<tr>
<td>Color auto-contrast</td>
<td>$\times$</td>
<td>$\times$</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
</tr>
<tr>
<td>Color jittering</td>
<td>$\times$</td>
<td>$\times$</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
</tr>
<tr>
<td>mIoU (%)</td>
<td>56.1</td>
<td>70.6</td>
<td>73.5</td>
<td>54.5</td>
<td>74.9</td>
</tr>
</tbody>
</table>

### Table II: Training strategies used in different methods for ScanObejectNN classification.

<table>
<thead>
<tr>
<th>Method</th>
<th>DGCNN</th>
<th>PointMLP</th>
<th>PointNet++</th>
<th>PointNeXt (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs</td>
<td>250</td>
<td>200</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>SGD</td>
<td>Adam</td>
<td>AdamW</td>
</tr>
<tr>
<td>LR</td>
<td>$1 \times 10^{-3}$</td>
<td>0.01</td>
<td>$1 \times 10^{-3}$</td>
<td>$2 \times 10^{-3}$</td>
</tr>
<tr>
<td>LR decay</td>
<td>step</td>
<td>cosine</td>
<td>step</td>
<td>cosine</td>
</tr>
<tr>
<td>Weight decay</td>
<td>10$^{-4}$</td>
<td>10$^{-4}$</td>
<td>10$^{-4}$</td>
<td>0.05</td>
</tr>
<tr>
<td>Label smoothing $\varepsilon$</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>Point resampling</td>
<td>$\times$</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Random rotation</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Random scaling</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Random translation</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Random jittering</td>
<td>✓</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Height appending</td>
<td>$\times$</td>
<td>$\times$</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OA (%)</td>
<td>78.1</td>
<td>85.7</td>
<td>77.9</td>
<td>87.7</td>
</tr>
</tbody>
</table>

### E Societal Impact

We do not see an immediate negative societal impact from our work. We notice that the way we discover the improved training and scaling strategies may consume a little more computing resources and affect the environment. Nevertheless, the improved training and scaling strategies will make researchers pay more attention to aspects other than architectural changes, which in the long term makes research in computer vision more diverse and generally better.

### References


### Table III: Training strategies used in different methods for ScanNet segmentation.

<table>
<thead>
<tr>
<th>Method</th>
<th>KPCov</th>
<th>PointTransformer</th>
<th>Stratified Transformer</th>
<th>PointNet++</th>
<th>PointNeXt (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Epochs</strong></td>
<td>500</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td><strong>Batch size</strong></td>
<td>10</td>
<td>16</td>
<td>8</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td><strong>Optimizer</strong></td>
<td>SGD</td>
<td>SGD</td>
<td>AdamW</td>
<td>Adam</td>
<td>AdamW</td>
</tr>
<tr>
<td><strong>LR</strong></td>
<td>$1 \times 10^{-2}$</td>
<td>$5 \times 10^{-1}$</td>
<td>$6 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td><strong>LR decay</strong></td>
<td>step</td>
<td>multi step</td>
<td>multi step with warm up</td>
<td>step</td>
<td>multi step</td>
</tr>
<tr>
<td><strong>Weight decay</strong></td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$5 \times 10^{-2}$</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td><strong>Entire scene as input</strong></td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Random rotation</strong></td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Random scaling</strong></td>
<td>[0.9,1.1]</td>
<td>[0.9,1.1]</td>
<td>[0.8,1.2]</td>
<td>×</td>
<td>[0.8,1.2]</td>
</tr>
<tr>
<td><strong>Random translation</strong></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td><strong>Random jittering</strong></td>
<td>0.001</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td><strong>Height appending</strong></td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Color drop</strong></td>
<td>✓</td>
<td>×</td>
<td>0.2</td>
<td>×</td>
<td>0.2</td>
</tr>
<tr>
<td><strong>Color auto-contrast</strong></td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Color jittering</strong></td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td><strong>Test mIoU (%)</strong></td>
<td>68.6</td>
<td>-</td>
<td>73.7</td>
<td>55.7</td>
<td>71.2</td>
</tr>
</tbody>
</table>

### Table IV: Training strategies used in different methods for ShapeNetPart segmentation.

<table>
<thead>
<tr>
<th>Method</th>
<th>DGCNN</th>
<th>KPCov</th>
<th>PointNet++</th>
<th>PointNeXt (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Epochs</strong></td>
<td>201</td>
<td>500</td>
<td>201</td>
<td>300</td>
</tr>
<tr>
<td><strong>Batch size</strong></td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td><strong>Optimizer</strong></td>
<td>Adam</td>
<td>SGD</td>
<td>Adam</td>
<td>AdamW</td>
</tr>
<tr>
<td><strong>LR</strong></td>
<td>$3 \times 10^{-3}$</td>
<td>$1 \times 10^{-2}$</td>
<td>$1 \times 10^{-3}$</td>
<td>0.001</td>
</tr>
<tr>
<td><strong>LR decay</strong></td>
<td>step</td>
<td>step</td>
<td>step</td>
<td>multi step</td>
</tr>
<tr>
<td><strong>Weight decay</strong></td>
<td>0.0</td>
<td>$10^{-3}$</td>
<td>0.0</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td><strong>Label smoothing</strong></td>
<td>ε</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td><strong>Random rotation</strong></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Random scaling</strong></td>
<td>[0.9,1.1]</td>
<td>✓</td>
<td>×</td>
<td>[0.8,1.2]</td>
</tr>
<tr>
<td><strong>Random translation</strong></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Random jittering</strong></td>
<td>0.001</td>
<td>✓</td>
<td>✓</td>
<td>0.001</td>
</tr>
<tr>
<td><strong>Normal Drop</strong></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Height appending</strong></td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>mIoU (%)</strong></td>
<td>85.2</td>
<td>86.4</td>
<td>85.1</td>
<td>87.0</td>
</tr>
</tbody>
</table>
Figure I: **PointNeXt architecture for classification.** The classification architecture shares the same encoder as the segmentation architecture.
Figure II: Qualitative comparisons of PointNet++ (2\textsuperscript{nd} column), PointNeXt (3\textsuperscript{rd} column), and Ground Truth (4\textsuperscript{th} column) on S3DIS semantic segmentation. The input point cloud is visualized with original colors in the 1\textsuperscript{st} column. Differences between PointNet++ and PointNeXt are highlighted with red dash circles. Zoom-in for details.
Figure III: Qualitative comparisons of PointNet++ (left), PointNeXt (middle), and Ground Truth (right) on ShapeNetPart part segmentation.