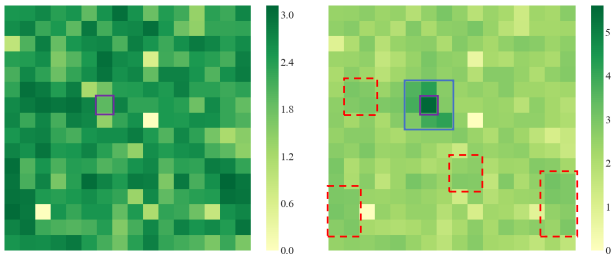
RepMLPNet: Hierarchical Vision MLP with Re-parameterized Locality Appendix



(A) Original FC kernel (B) FC kernel with locality injected

Figure 3. FC kernel before and after Locality Injection.

Appendix A: Visualizing Locality Injection

We demonstrate the locality is injected into the FC kernel by showing the kernel weights.

Specifically, we visualize the weights of FC3 kernel sampled from the 10th RepMLP Block of the 3rd stage of RepMLPNet-D256 trained on ImageNet. We reshape the kernel into $\bar{\mathrm{W}}(s, h, w, 1, h, w)$, which is $(16, 16, 16, 1, 16, 16)$, then sample the weights related to the first input channel and the (7,7) point (marked by a purple square) on the first output channel, which is $\bar{\mathrm{W}}_{1,7,7,1,:,:}$ (indices starting from 1). Then we take the absolute value, rescale by the minimum value of the whole matrix, and take the logarithm for the better readability.

As shown in Fig. 3, a point with darker color indicates the FC considers the corresponding position on the input channel more related to the output point at (7,7). Obviously, the original kernel has no locality as the marked point and the neighbors have no larger values than the others.

Then we merge the parallel conv layers into the kernel via Locality Injection. The resultant kernel has larger values around the marked point, suggesting that the model focuses more on the neighbors, which is expected. Besides, the kernel's capacity to model the long-range dependencies is not lost as some points (marked by red dashed rectangles) outside the largest conv kernel (3×3 in this case, marked by a blue square) still have larger values than some points inside.

Appendix B: Details of Semantic Segmentation

We solve the problem of using MLP as the backbone of a semantic segmentation framework (*e.g.*, UperNet) by **1)** the hierarchical design, **2)** splitting feature maps into non-overlapping patches and **3)** communications between patches.

Fig. 4 shows an example of RepMLPNet + UperNet.

1) UperNet requires feauture maps of 4 different levels, which fits our hierarchical architecture. **2)** Since RepMLP Block works with a fixed input feature map size, we split the feature maps into non-overlapping patches each with the required size. **3)** The original 2× embedding cannot realize inter-patch communication, so we replace it with 3×3 conv. To reduce the computational cost of 3×3 conv, we decompose it into a 1×1 conv for expanding the channels and a 3×3 stride-2 depth-wise conv for downsampling. Fig. 5 shows the difference between a 2× embedding and a 3×3 conv.

Interestingly, as the input is split into non-overlapping patches, one may expect that the predictions would be less accurate at the edges of patches, but we observe no such phenomenon. Fig. 6 shows that the predictions at the edges are as good as the internal pixels within patches. This discovery suggests that the dependencies across patches have been well established and that the representational capacity of MLP is strong enough for such a dense prediction task.

We hope our results spark further research on the application of MLP on downstream tasks.

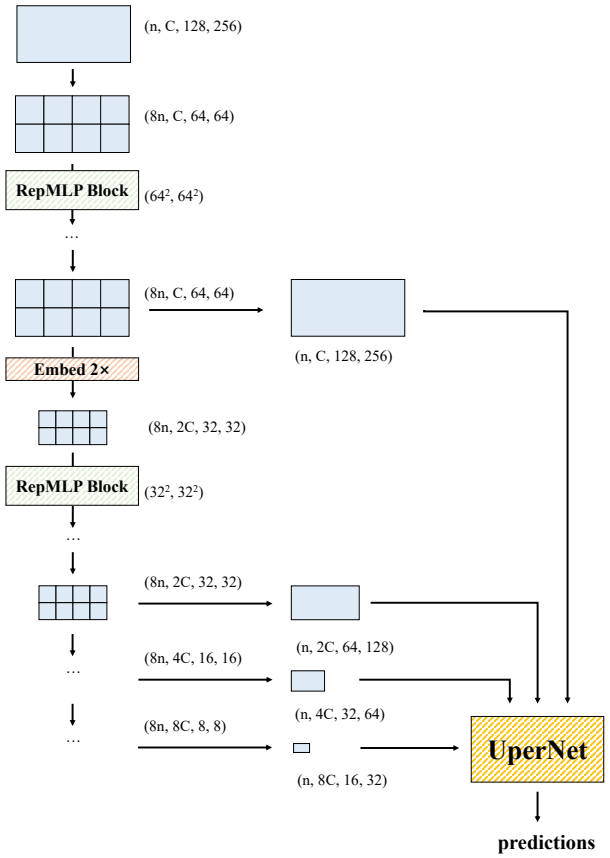
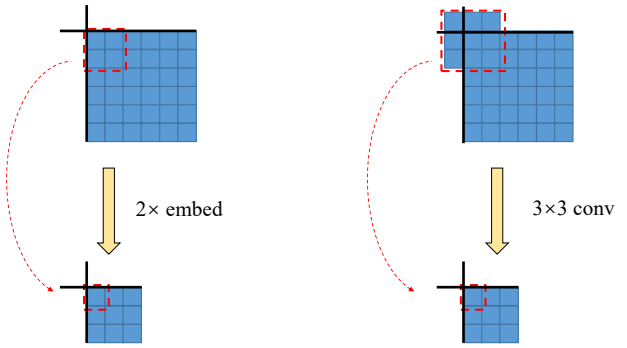


Figure 4. An example of using RepMLPNet as the backbone of UperNet. After 4× downsampling on the 512×1024 inputs, the feature map size becomes 128×256. Then we split the feature maps into 2×4 non-overlapping patches, each of 64×64, because the first RepMLP Block maps inputs of 64×64 into 64×64 (*i.e.*, the FC kernel is $(64^2,64^2)$). Similarly, the outputs of the four stages are reshaped back and fed into the UperNet.



(A) Downsample with 2× embed (B) Downsample with 3×3 conv

Figure 5. The difference between 2× embedding and 3×3 conv is that the latter realizes communications across patch edges. In this figure, a square denotes a pixel on a feature map and the thick lines denote the edges of patches. We take the upper left corner of a patch as an example

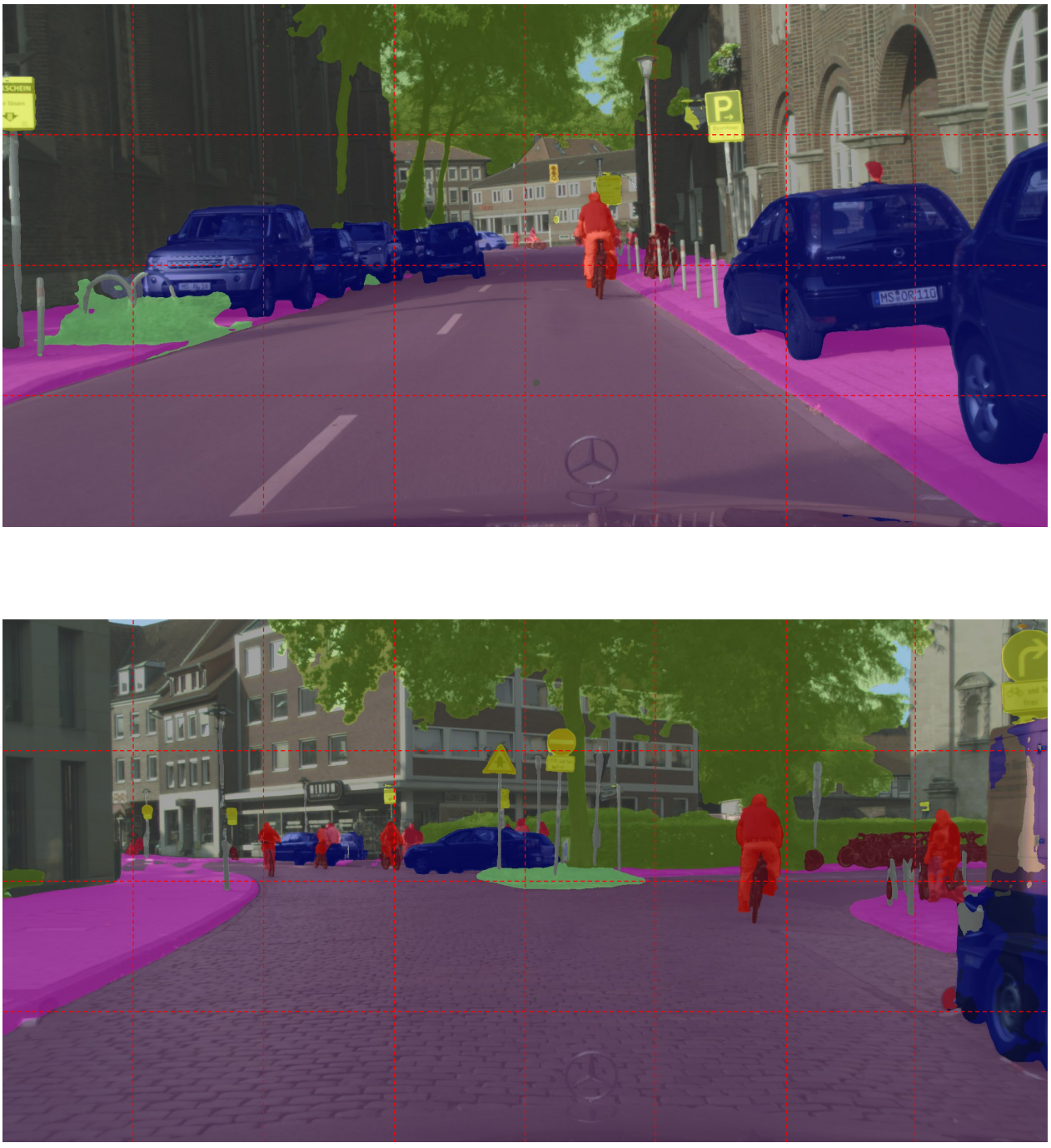


Figure 6. The predictions at the edges of patches are no observably worse. We show two images from the Cityscapes validation set as examples. As the test resolution is 1024×2048, the input to the first RepMLP Block is 256×512 (after the beginning 4× downsampling), so that the input is split into 32 non-overlapping patches and then fed into RepMLP Blocks. We use red dashed lines to denote the edges of patches and it is observed that the predictions at the edges are almost as good as the other parts.