# Supplementary Material for "RaftMLP: How Much Can Be Done Without Attention and with Less Spatial Locality?"

Yuki Tatsunami[1,2][0000−0002−7889−8143] and Masato Taki[1][0000−0002−5375−7862]

[1] Rikkyo University, Tokyo, Japan
{y.tatsunami, taki_m}@rikkyo.ac.jp
[2] AnyTech Co., Ltd., Tokyo, Japan

## A More pseudocode

This section describes the pseudo-code for the methods discussed in this paper. The pseudo-code for Multi-scale Patch Embedding is detailed in Listing 1.1.

```
# b: size of mini-batch, h: height, w: width,
# kernels: list of kernel sizes for unfold.
#          e.g., [4, 8]

def __init__(self, in_channels, out_channels, kernels):
    mlp_in_channels = 0
    for k in kernels:
        mlp_in_channels += k ** 2
    mlp_in_channels *= in_channels
    self.embeddings = nn.ModuleList([
        nn.Sequential(*[nn.Unfold(
                kernel_size=k,
                stride=self.stride,
                padding=(k - self.stride) // 2),
            Rearrange("b c hw -> b hw c")
        ]) for k in kernels
    ])
    self.fc = nn.Linear(
        mlp_in_channels, out_channels
    )

def forward(self, input):
    b, _, h, w = input.shape
    outputs = []
    for emb in self.embeddings:
        output = emb(input)
        outputs.append(output)
    return self.fc(torch.cat(outputs, dim=2))
```

Listing 1.1: Pseudocode of multi-scale patch embedding (Pytorch-like)

## B Downstream Task Application

In this section, we discuss the application of our model to fownstream tasks. In order to apply our models to downstream tasks such as semantic segmentation, instance segmentation, and object detection, various resolutions need to be supported. Therefore, we insert bicubic interpolation before and after the raft-token-mixing block, as shown in Fig. 1. In the bicubic interpolation before the block, we convert the input to the resolution used for pre-training. The bicubic interpolation after the block restores the resolution before the first bicubic interpolation. Moreover, since the resolution of input images is not always divisible by the patch size, we apply bicubic interpolation to obtain the resolution before multi-scale embedding that is a factor of the patch size. This method can be applied to other global MLP-based models such as MLP-Mixer too.

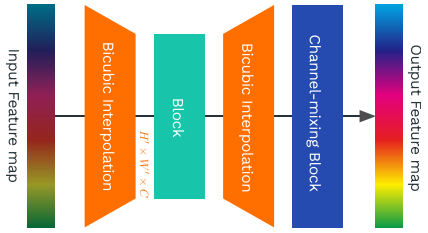



Fig. 1: Application of RaftMLP block utilizing bicubic interpolation

### B.1 Object Detetion

For the evaluation of object detection and instance segmentation, we compose a model in which the backbones of RetinaNet [8] and Mask R-CNN [3], which are both standard implementations on the object detection framework `mmdetection` [1], are replaced by RaftMLP and MLP-Mixer. For the dataset, we used MS COCO [9], which is one of the most popular benchmark datasets for object detection. The training setup is similar to ConvMLP [6], with AdamW as the optimizer, learning rate set to $10^{-4}$, weight decay set to $10^{-4}$, and 12 epochs of training with a batch size of 16. The results are compared with PureMLP [6], ResNet [4], and ConvMLP [6], and a summary is provided in Fig. 2. See Appendix B.4 for more details.

### B.2 Semantic Segmentation

We replace the backbone of Semantic FPN [5] implemented on `mmsegmention` [2] with RaftMLP and MLP-Mixer and evaluate their performances on the segmentation task. We adopt AdamW as the optimizer with a learning rate of $2.0 \times 10^{-4}$ and a weight decay of $10^{-4}$. The learning schedule follows the polynomial decay

learning rate policy with a power of 0.9. We use the famous ADE20K dataset [11] to train the model, with the input image randomly resized and cropped to a resolution of $512 \times 512$. The model had trained for 40000 iterations. The above settings follow ConvMLP [6]. A summary of the experimental results is shown in Fig. 2. See Appendix B.4 for more details.

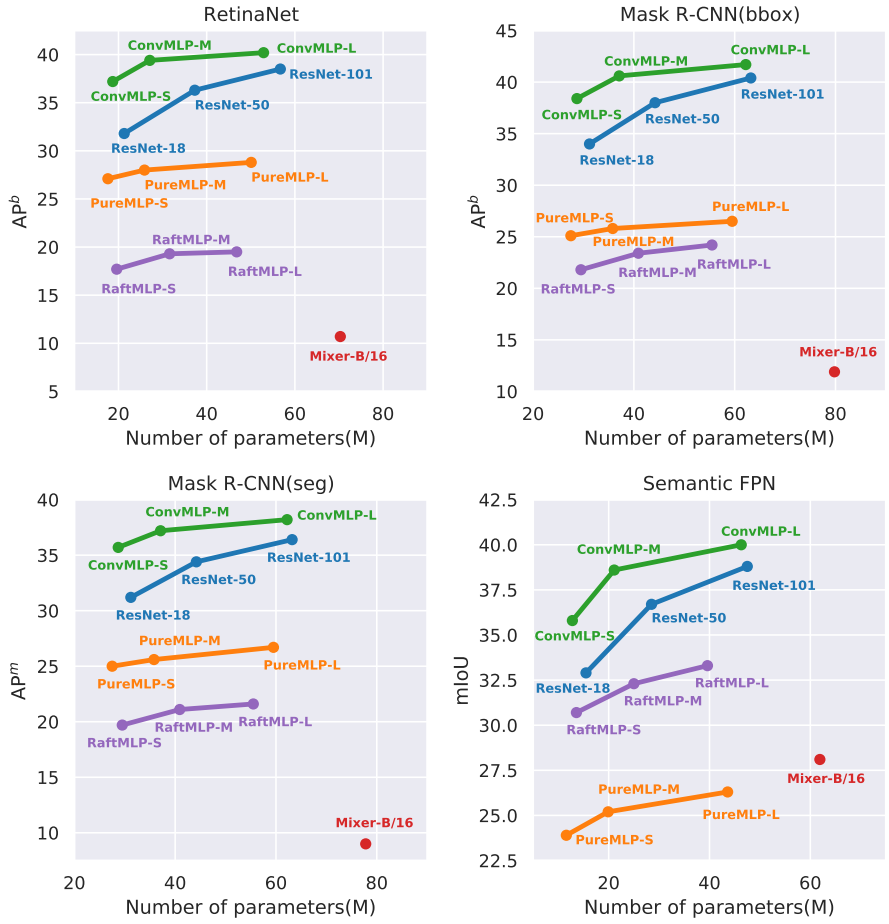



Fig. 2: The above compares the training results of RetinaNet and Mask R-CNN on MS COCO and Semantic FPN on ADE20K. We compare the results with ResNet, PureMLP, ConvMLP, Mixer, and RaftMLP as backbones in each case. RetinaNet uses AP for bounding boxes, Mask R-CNN AP for bounding boxes and segmentation, and Semantic FPN uses mIoU as their metric.

### B.3 Details of Architectures

*RaftMLP* The details of the architectures of RaftMLP-S, RaftMLP-M, and RaftMLP-L used in this paper are details in Table 1.

*Object Detection, Instance Segmentation and Semantic Segmentation* For the RetinaNet [8] and Mask R-CNN [3] and Semantic FPN [5] we used, we consulted the results of [6], which is the same setup for ResNet, PureMLP, and ConvMLP, which are our comparison. The backbones we have experimented with are RaftMLP and Mixer-B/16. All of the architectures we have arranged use Feature Pyramid Network [7]. Therefore, we must clearly state what feature pyramid was input to these architectures from the backbones RaftMLP and Mixer-B/16. RaftMLP utilizes the output immediately after the first multi-scale patch embedding and the outputs of Level-2 to Level-4 as feature maps to be input to the detector and segmentor. Similarly, Mixer-B/16, along with RaftMLP, uses the output immediately after the patch embedding and the outputs of Block-4, Block-10, and Block-12 as before-mentioned feature maps.

### B.4 Details of Quantitative Results

*Object Detection and Instance Segmentation* Table 2 contains the detailed results of the experiment for RetinaNet performed in Subsection B.1, Table 3 includes the detailed results of the experiment for Mask R-CNN worked in Subsection B.1. The results of RetinaNet are not doing as well overall as PureMLP, even with RaftMLP, which guarantees some spatial structure. In particular, it struggles to detect small objects. This result can be seen in B.6, where RaftMLP adds artifacts to the feature map, harming object detection.

*Semantic Segmentation* Table 4 contains the detailed results of the experiment performed in Subsection B.2.

### B.5 Qualitative Results

*Object Detection and Instance Segmentation* Fig. 3a shows the ground truth for an sample of MS COCO validation dataset. Fig. 3b shows the inference result for RetinaNet with ResNet-50 as the backbone to be installed in `mmdetection`, and Fig. 3c, 3d, 3e, and 3f inference results for the four RetinaNets trained in Subsection B.1. Fig. 3g shows the inference result for Mask R-CNN with ResNet-50 as the backbone to be installed in `mmdetection`, and Fig. 3h, 3i, 3j, and 3k inference results for the four Mask R-CNNs trained in Subsection B.1. Despite the lack of precision, the figures reveal that results of Global MLP-based models for the object detection and instance segmentation tasks are satisfactory.

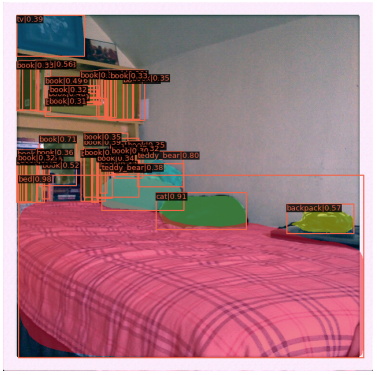
*Semantic Segmentation* Fig. 4a presents an image with ADE20k validation dataset overlaid with its ground truth. Fig. 4b shows the inference results of the model applying ResNet-50, which `mmsegmentation` provides, as the backbone of the Semantic FPN. Fig. 4c, 4d, 4e, and 4f show the inference results for the four models we trained in Subsection B.2 experiment.

(a) Ground Truth (b) RetinaNet|ResNet-50

(c) RetinaNet|Mixer-B/16 (d) RetinaNet|RaftMLP-S

(e) RetinaNet|RaftMLP-M (f) RetinaNet|RaftMLP-L

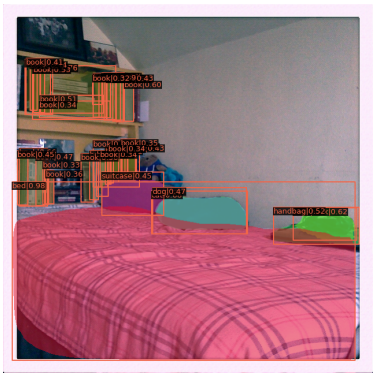
Fig. 3: Qualitative results of object detection and instance segmentation



(g) Mask R-CNN|ResNet-50

(h) Mask R-CNN|Mixer-B/16

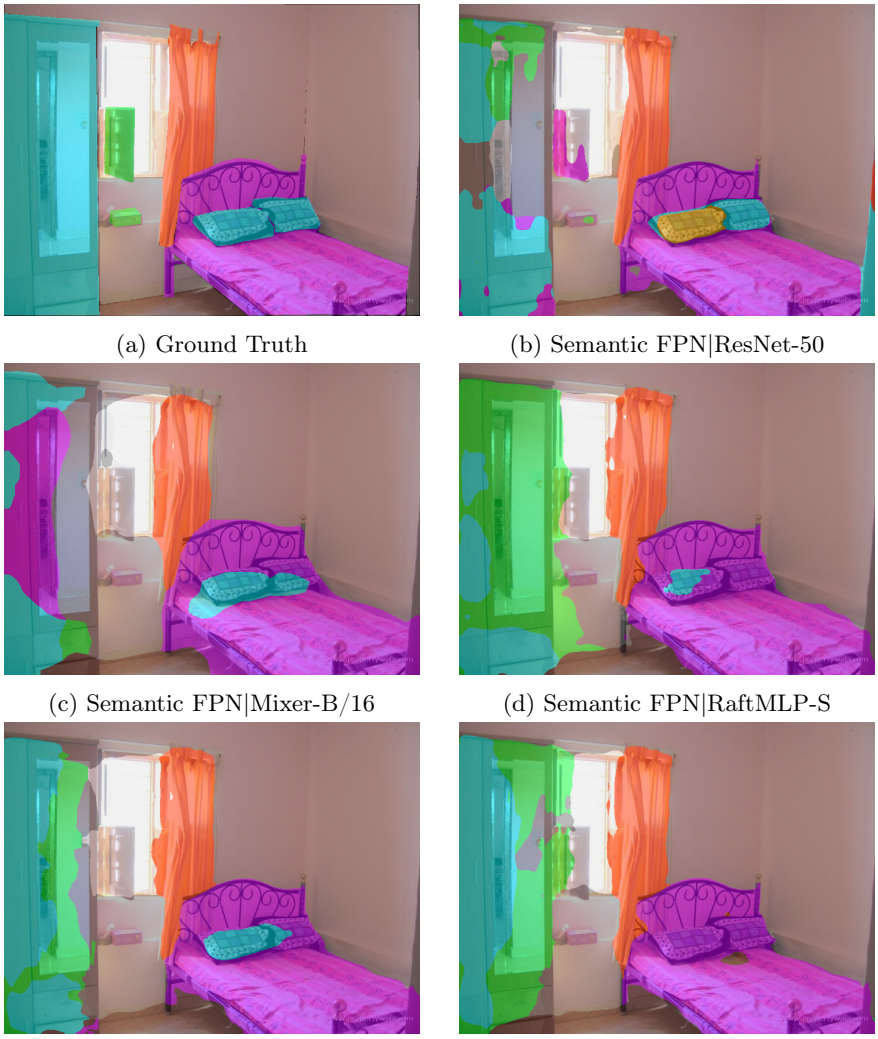
(i) Mask R-CNN|RaftMLP-S



(j) Mask R-CNN|RaftMLP-M

(k) Mask R-CNN|RaftMLP-L

Fig. 3: Qualitative results of object detection and instance segmentation



(a) Ground Truth (b) Semantic FPN|ResNet-50

(c) Semantic FPN|Mixer-B/16 (d) Semantic FPN|RaftMLP-S

(e) Semantic FPN|RaftMLP-M (f) Semantic FPN|RaftMLP-L

Fig. 4: Qualitative results of semantic segmentation

### B.6 Visualization

We used an image with ImageNet to visualize and compare its feature map. The image used as input was the ferret image on the left of Fig. 5, which was input to pre-trained ResNet-50, Mixer-B/16, and RaftMLP-M. Some of the outputs of the intermediate layers are summarized on the right side of Fig. 5. For ResNet-50, we used the output of layers 1 through 4; for Mixer-B/16, we used the output of Blocks 2, 4, 10, and 12; for RaftMLP-M, we used the output of each Level. We have also included further intermediate layer outputs for the three models,

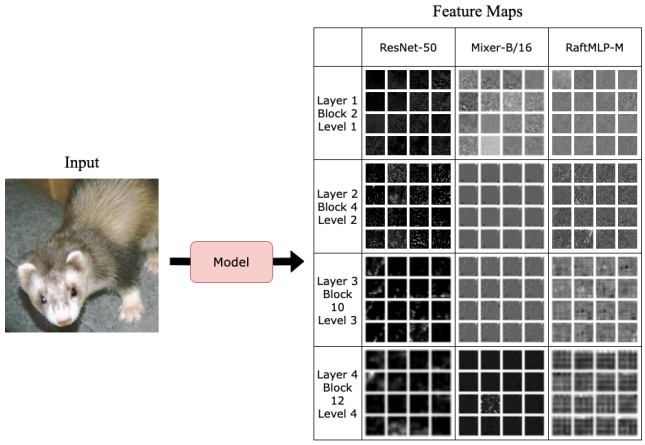



Fig. 5: Summary of the comparison of ResNet-50, Mixer-B/16, and RaftMLP-M intermediate layer feature maps

see Fig. 6, 7, 8, and 9 for Resnet-50, Fig. 10, 11, 12 and 13 for Mixer-B/16, and Fig. 14, 15, 16, and 17 for RaftMLP-M.

As mentioned in Section 5, the appearance of features in the middle layer of global MLP-based models is different from that of the convolutional base represented by ResNet. We believe this is why global MLP-based models do not perform well when selected as the backbone of existing architectures for object detection, instance segmentation, and semantic segmentation. The feature map of RaftMLP-M is different from that of ResNet in that the lower layers have feature maps that capture the features of the ferret. In contrast, the upper layers have feature maps with visible artifacts of vertical and horizontal lines. The feature maps of Mixer-B/16 do not capture the features of the ferret, and they are overall shuffled and have many similar feature maps. Tasks such as object detection, semantic segmentation, or even image generation will require innovations specific to global MLP-based models. The occurrence of artifacts might have a minor impact on classification, where global average pooling is used. However, for tasks such as segmentation and image generation, it becomes a severe problem. Hence, it will be necessary to design architectures and loss functions that do not emit this artifact. Or else, convolution-based methods such as RetinaNet, Mask R-CNN, and Semantic FPN may be insufficient to recover the whole shuffled information by global MLP-based models. To recover the global shuffled information by the global MLP-based model, global MLP-based models may lack a module that can capture the global relations, such as self-attention modules and token-mixing blocks.

Table 1: Specific settings on the model architectures of hierarchy RaftMLP in different scales. $l$ denotes level and $c'_l$ denotes the number of basic channels in RaftMLP for level $l$.

| Model | RaftMLP-S | | RaftMLP-M | | RaftMLP-L | |
|---|---|---|---|---|---|---|
| Level | Block | Setting | Block | Setting | Block | Setting |
| $l=1$ | RaftMLP×2 | $c'_1=64$ | RaftMLP×2 | $c'_1=96$ | RaftMLP×2 | $c'_1=128$ |
| $l=2$ | RaftMLP×2 | $c'_2=128$ | RaftMLP×2 | $c'_2=192$ | RaftMLP×2 | $c'_1=192$ |
| $l=3$ | RaftMLP×6 | $c'_3=256$ | RaftMLP×6 | $c'_3=384$ | RaftMLP×6 | $c'_1=512$ |
| $l=4$ | RaftMLP×2 | $c'_4=512$ | RaftMLP×2 | $c'_4=768$ | RaftMLP×2 | $c'_1=1024$ |

Table 2: Comparison of RetinaNet metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP, and Mixer as the backbone.

| Backbone | #MParams | $AP^b$ | $AP^b_{50}$ | $AP^b_{75}$ | $AP^b_S$ | $AP^b_M$ | $AP^b_L$ |
|---|---|---|---|---|---|---|---|
| ResNet-18 [4, 6] | 21.3 | 31.8 | 49.6 | 33.6 | 16.3 | 34.3 | 43.2 |
| PureMLP-S [6] | 17.6 | 27.1 | 44.2 | 28.3 | 13.6 | 29.2 | 36.4 |
| ConvMLP-S [6] | 18.7 | 37.2 | 56.4 | 39.8 | 20.1 | 40.7 | 50.4 |
| RaftMLP-S | 19.6 | 17.7 | 33.3 | 16.5 | 4.5 | 14.1 | 32.4 |
| ResNet-50 [4, 6] | 37.7 | 36.3 | 55.3 | 38.6 | 19.3 | 40.0 | 48.8 |
| PureMLP-M [6] | 25.9 | 28.0 | 45.6 | 29.0 | 14.5 | 29.9 | 37.8 |
| ConvMLP-M [6] | 27.1 | 39.4 | 58.7 | 42.0 | 21.5 | 43.2 | 52.5 |
| RaftMLP-M | 27.1 | 19.3 | 36.3 | 17.8 | 5.2 | 15.9 | 35.1 |
| ResNet-101 [4, 6] | 56.7 | 38.5 | 57.8 | 41.2 | 21.4 | 42.6 | 51.1 |
| PureMLP-L [6] | 50.1 | 28.8 | 46.8 | 29.9 | 15.0 | 31.0 | 38.4 |
| ConvMLP-L [6] | 52.9 | 40.2 | 59.3 | 43.3 | 23.5 | 43.8 | 53.3 |
| RaftMLP-L | 52.9 | 19.5 | 36.8 | 18.1 | 5.0 | 16.1 | 35.4 |
| Mixer-B/16 [10] | 70.3 | 10.7 | 20.0 | 10.1 | 0.1 | 6.7 | 25.8 |

## References


1. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open MMLab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019), https://github.com/open-mmlab/mmdetection
2. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020)
3. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV. pp. 2961–2969 (2017)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
5. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR. pp. 6399–6408 (2019)

Table 3: Comparison of Mask R-CNN metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP and Mixer as the backbone.

| Backbone | #MParams | $AP^b$ | $AP^b_{50}$ | $AP^b_{75}$ | $AP^m$ | $AP^m_{50}$ | $AP^m_{75}$ |
|---|---|---|---|---|---|---|---|
| ResNet-18 [4, 6] | 31.2 | 34.0 | 54.0 | 36.7 | 31.2 | 51.0 | 32.7 |
| PureMLP-S [6] | 27.5 | 25.1 | 45.1 | 25.1 | 25.0 | 42.8 | 26.0 |
| ConvMLP-S [6] | 28.7 | 38.4 | 59.8 | 41.8 | 35.7 | 56.7 | 38.2 |
| RaftMLP-S | 29.5 | 21.8 | 40.2 | 21.0 | 19.7 | 36.5 | 19.1 |
| ResNet-50 [4, 6] | 44.2 | 38.0 | 58.6 | 41.4 | 34.4 | 55.1 | 36.7 |
| PureMLP-M [6] | 35.8 | 25.8 | 46.1 | 25.8 | 25.6 | 43.5 | 26.5 |
| ConvMLP-M [6] | 37.1 | 40.6 | 61.7 | 44.5 | 37.2 | 58.8 | 39.8 |
| RaftMLP-M | 40.9 | 23.4 | 42.5 | 22.7 | 21.1 | 38.8 | 20.8 |
| ResNet-101 [4, 6] | 63.2 | 40.4 | 61.1 | 44.2 | 36.4 | 57.7 | 38.8 |
| PureMLP-L [6] | 59.5 | 26.5 | 45.0 | 27.4 | 26.7 | 47.5 | 26.8 |
| ConvMLP-L [6] | 62.2 | 41.7 | 62.8 | 45.5 | 38.2 | 59.9 | 41.1 |
| RaftMLP-L | 55.5 | 24.2 | 43.9 | 23.7 | 21.6 | 39.7 | 23.7 |
| Mixer-B/16 [10] | 79.8 | 11.9 | 22.8 | 11.2 | 9.5 | 19.1 | 8.5 |

Table 4: Comparison of Semantic FPN metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP and Mixer as the backbone.

| Backbone | #MParams | mIoU |
|---|---|---|
| ResNet-18 [4, 6] | 15.5 | 32.9 |
| Pure-MlP-S [6] | 11.6 | 23.9 |
| ConvMLP-S [6] | 12.8 | 35.8 |
| RaftMLP-S | 13.6 | 30.7 |
| ResNet-50 [4, 6] | 28.5 | 36.7 |
| Pure-MlP-M [6] | 19.9 | 25.2 |
| ConvMLP-M [6] | 21.1 | 38.6 |
| RaftMLP-M | 25.0 | 32.3 |
| ResNet-101 [4, 6] | 47.5 | 38.8 |
| Pure-MlP-L [6] | 43.6 | 26.3 |
| ConvMLP-L [6] | 46.3 | 40.0 |
| RaftMLP-L | 39.6 | 33.3 |
| Mixer-B/16 [10] | 63.9 | 28.1 |

6. Li, J., Hassani, A., Walton, S., Shi, H.: ConvMLP: Hierarchical convolutional MLPs for vision. arXiv preprint arXiv:2109.04454 (2021)
7. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. pp. 2117–2125 (2017)
8. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)
9. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. pp.

740–755 (2014)
10. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al.: Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601 (2021)
11. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR. pp. 633–641 (2017)

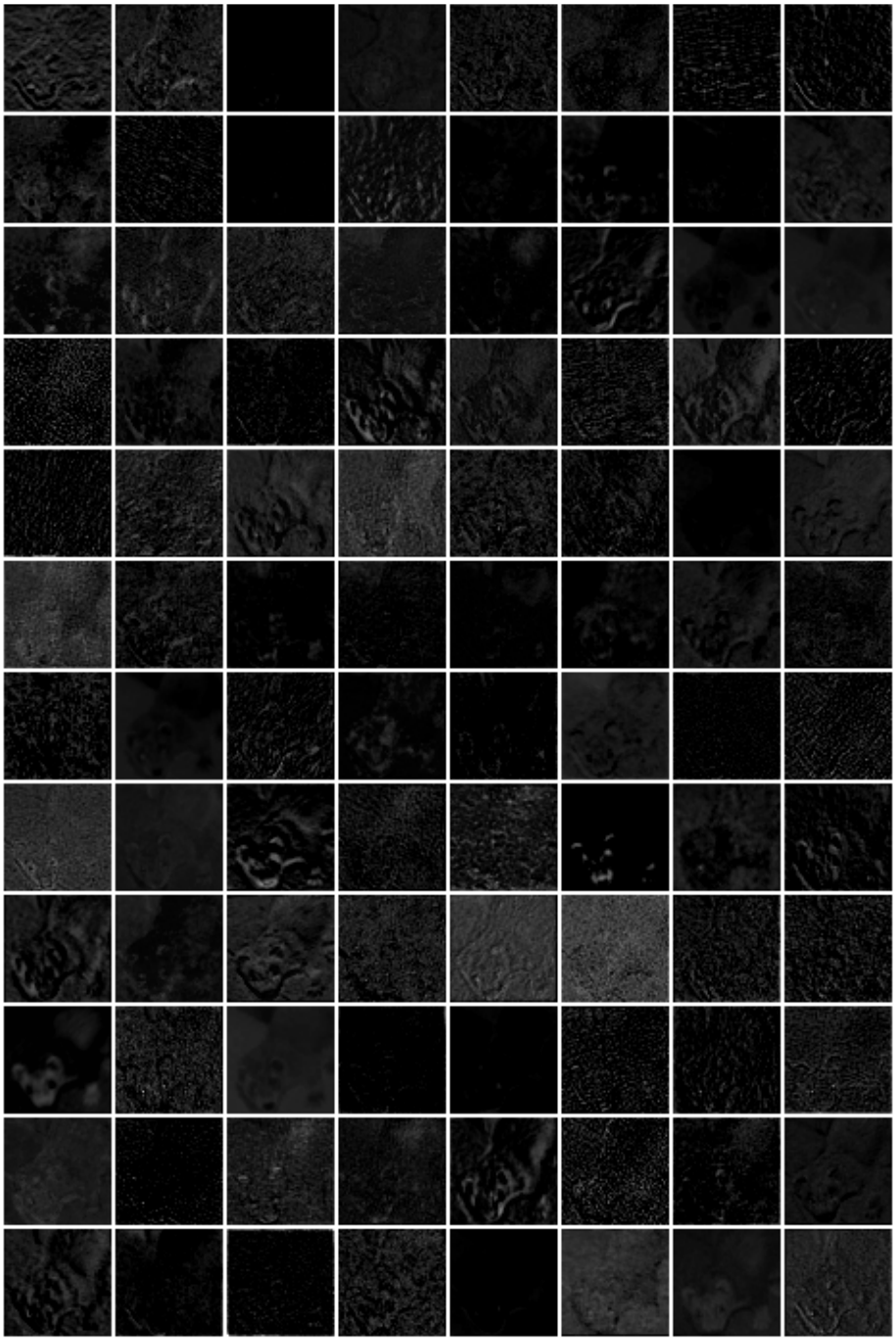


Fig. 6: Part of the feature maps output from Layer-1 of ResNet-50 with the ferret images as input

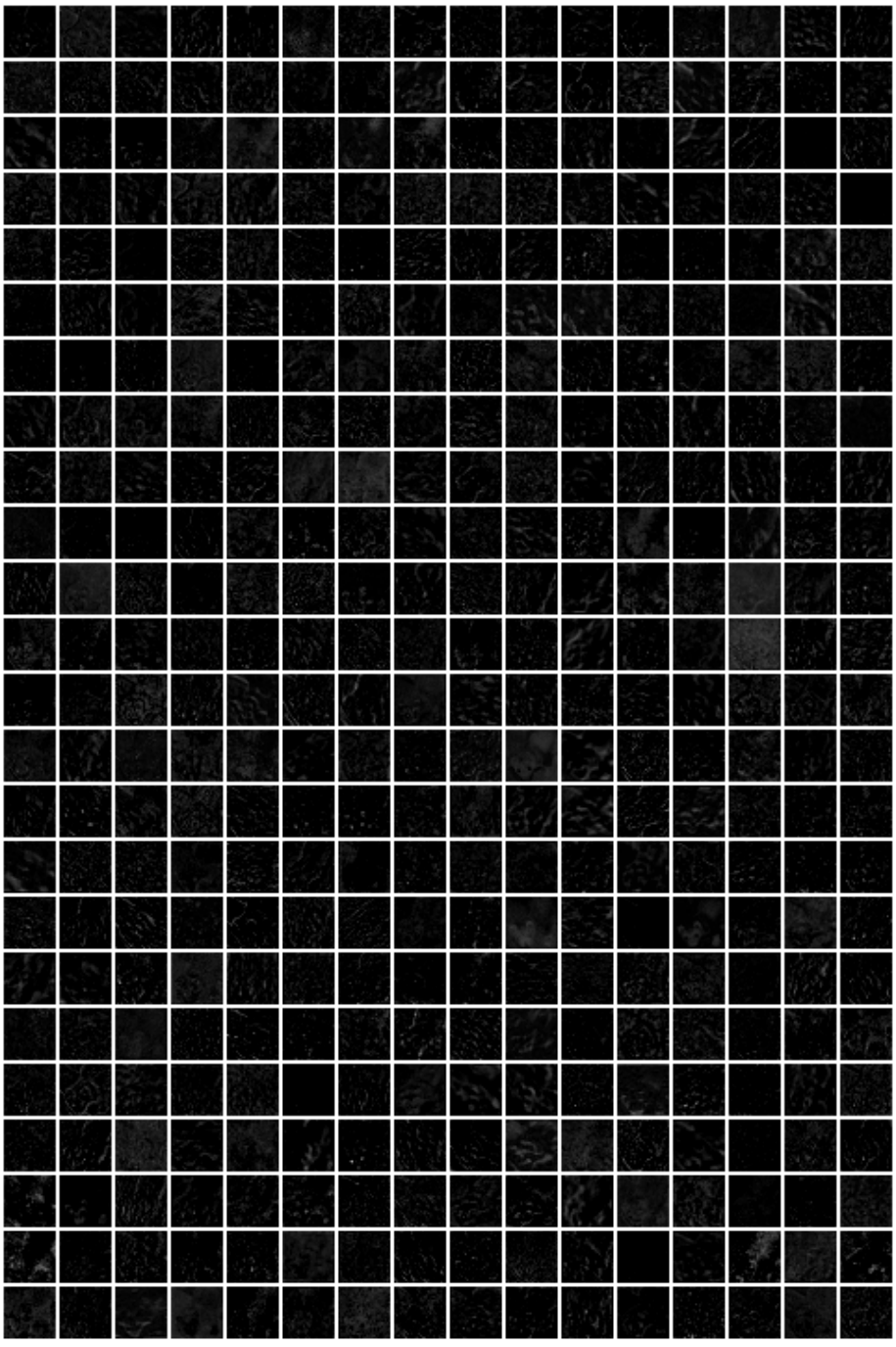


Fig. 7: All the feature maps output from Layer-2 of ResNet-50 with the ferret images as input

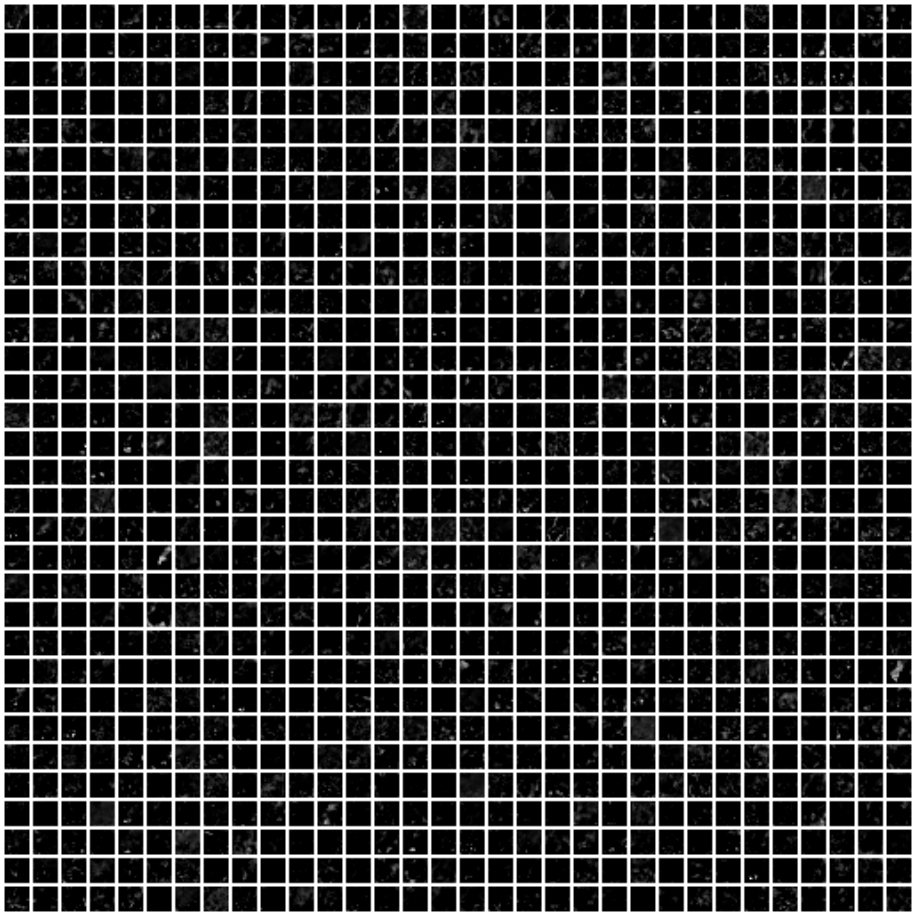


Fig. 8: All the feature maps output from Layer-3 of ResNet-50 with the ferret images as input

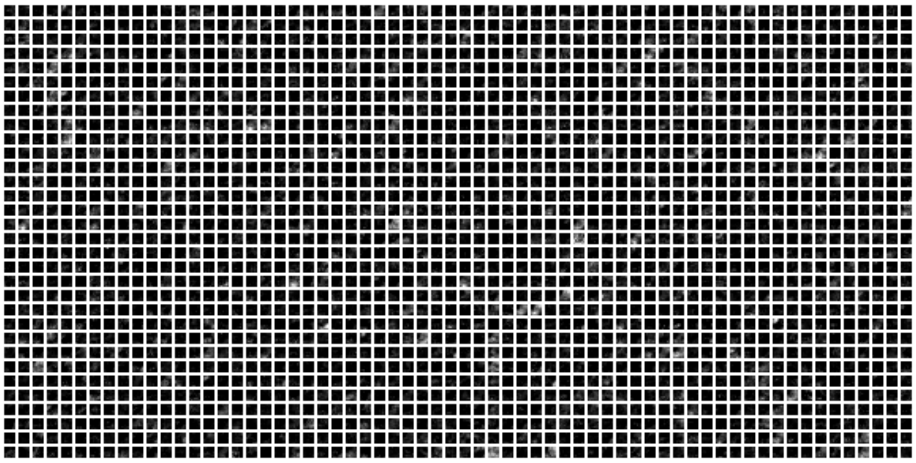


Fig. 9: All the feature maps output from Layer-4 of ResNet-50 with the ferret images as input

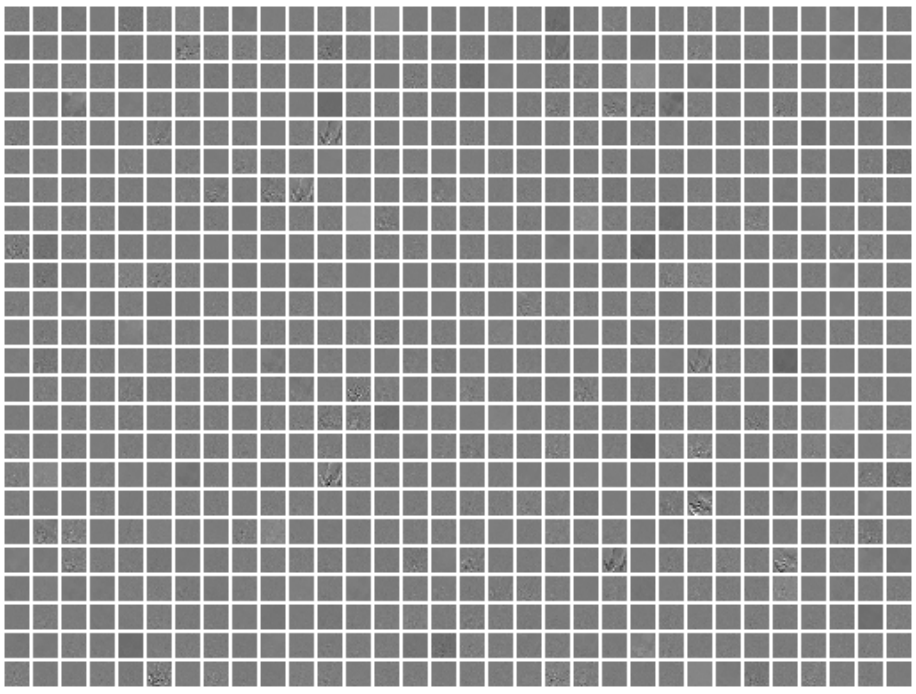


Fig. 10: All the feature maps output from Block-2 of Mixer-B/16 with the ferret images as input

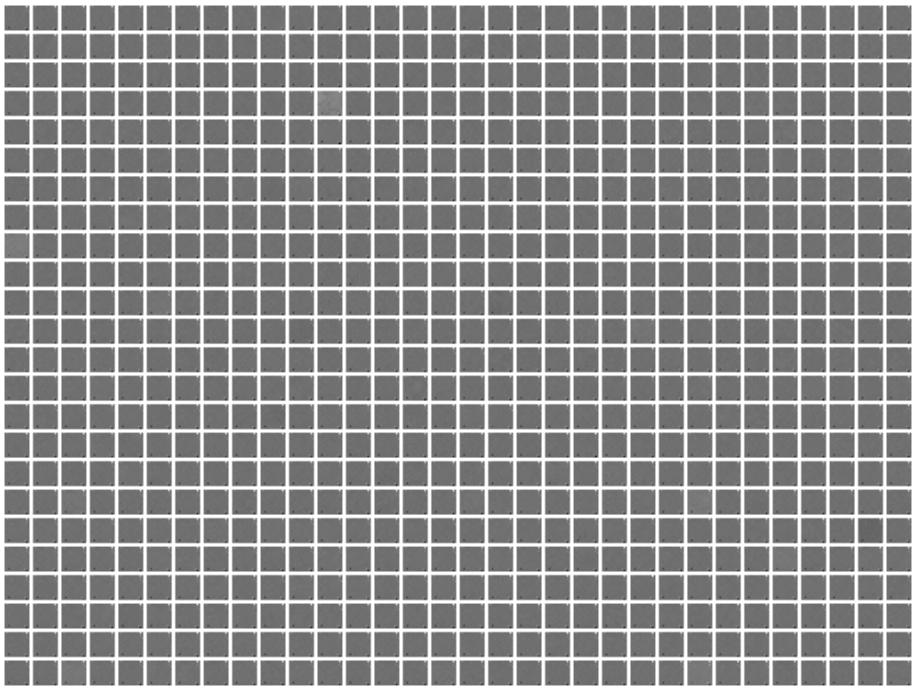


Fig. 11: All the feature maps output from Block-4 of Mixer-B/16 with the ferret images as input

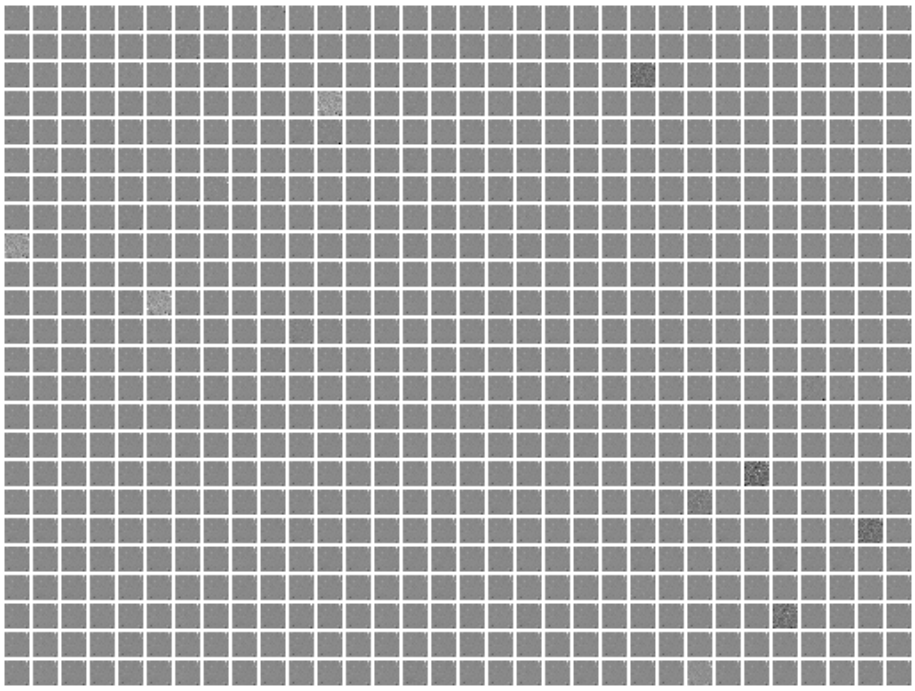


Fig. 12: All the feature maps output from Block-10 of Mixer-B/16 with the ferret images as input

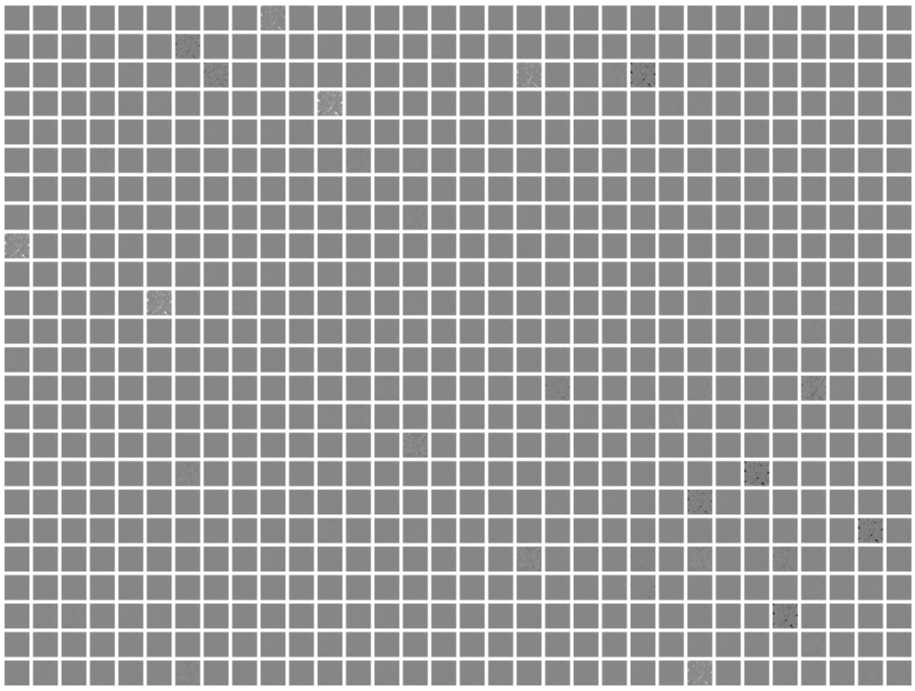


Fig. 13: All the feature maps output from Block-12 of Mixer-B/16 with the ferret images as input

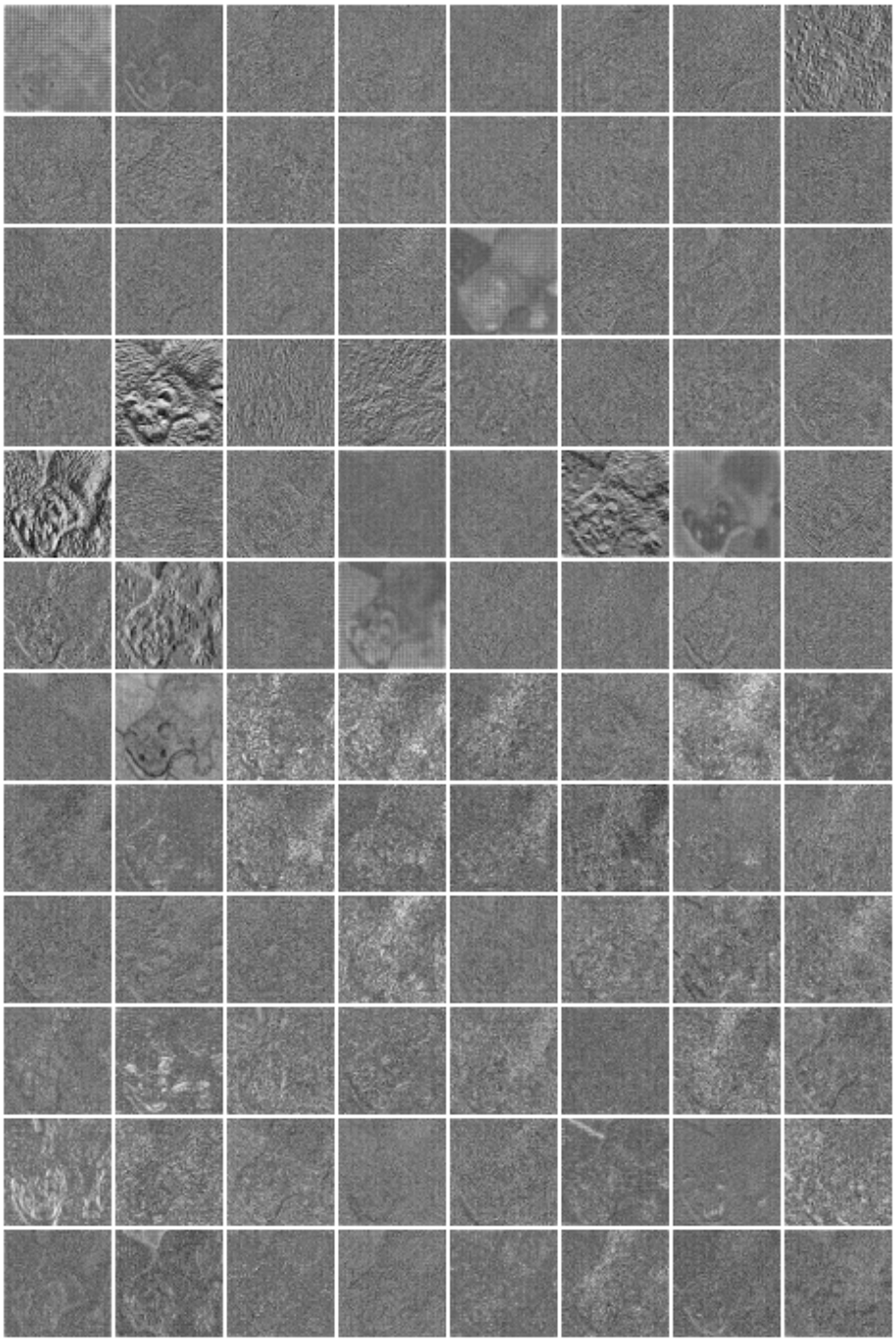


Fig. 14: All the feature maps output from Level-1 of RaftMLP-M with the ferret images as input

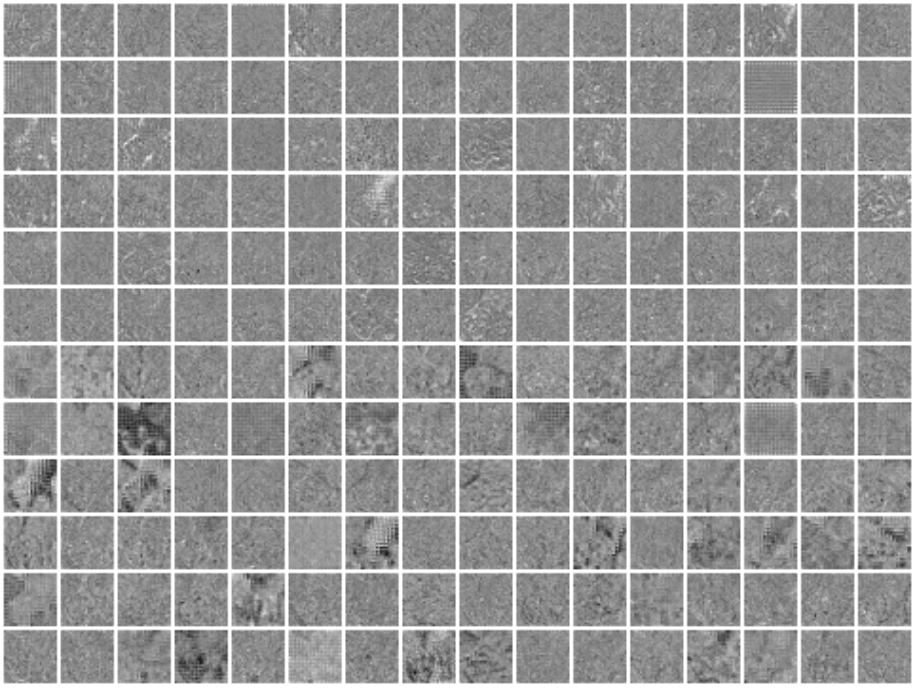


Fig. 15: All the feature maps output from Level-2 of RaftMLP-M with the ferret images as input

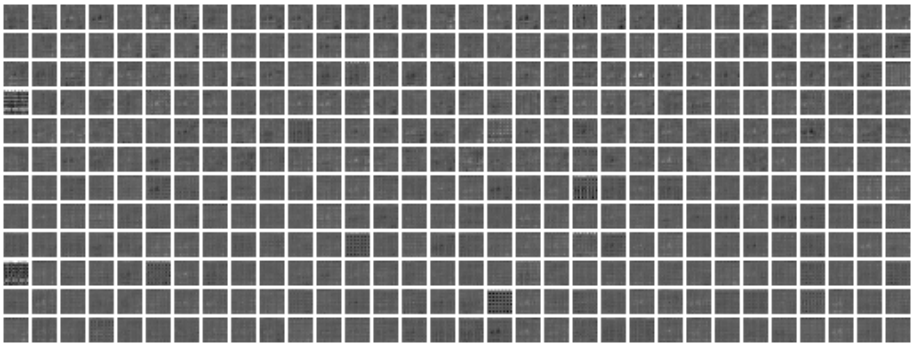


Fig. 16: All the feature maps output from Level-3 of RaftMLP-M with the ferret images as input

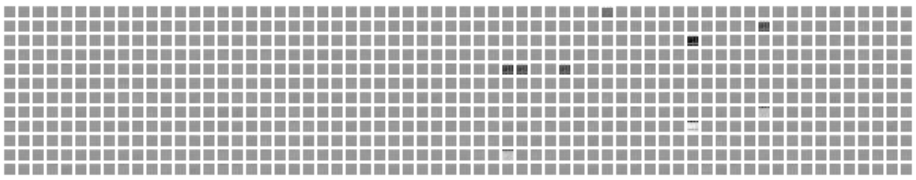


Fig. 17: All the feature maps output from Level-4 of RaftMLP-M with the ferret images as input