UniConvNet: Expanding Effective Receptive Field while Maintaining Asymptotically Gaussian Distribution for ConvNets of Any Scale

Supplementary Material

Appendix

A. A proper Asymptotically Gaussian Distribution of small-scale pixels is more important than expanding the Effective Receptive Field

In dense prediction tasks (e.g., detection and segmentation), integrating contextual information via a large effective receptive field (ERF) [42] and distinguishing pixels of different scales are crucial. As shown in Fig. 6, the ERF is used to visualize the effectiveness of the proposed Three-layer RFA.

MogaNet-S [32] has similar asymptotically Gaussian distribution (AGD) at small-scale pixels, around the center, compared to UniConvNet-T, but UniConvNet-T exhibits a significantly larger ERF. This indicates that **expanding the ERF** while **maintaining the AGD** could help to generate a multi-scale impact, following AGD from center to edge, of a larger ERF, which consequently enhances the performance.

When comparing MogaNet-S [32] with ConvNeXt-T [40], MogaNet-S [32] has a better AGD at small-scale pixels with comparable ERF scale. This enables MogaNet-S [32] to have superior performance, demonstrating that the **AGD of small-scale pixels** is **more important** when the **ERF scale is comparable**.

Compared to UniConvNet-T, SLaK-T [37] achieves comparable ERF scale while disrupting the AGD of ERF. The top-1 accuracy on ImageNet increased by 1.7% point owing to the proper AGD on the area larger than small-scale area in SLaK-T [37]. UniRepLKNet-T [18] achieves much larger ERF, compared to SLaK-T [37], with inferior AGD, which benefits from extremely large ERF. It is constrained by high parameters and FLOPs costs compared with UniConvNet-T. This demonstrates that the sparsity [37] and re-parameterization [15, 16] techniques effectively enlarge the ERF but suffer from improper AGD of smaller-scale pixels (the dark gray area in UniRepLKNet-T [18]). Compared to UniConvNet-B, RepLKNet-31B [17] achieves a larger ERF but compromises small-scale AGD, with a 1.0% TOP-1 accuracy drop on ImageNet. These phenomena typically demonstrate our viewpoint that **a proper asymptotically Gaussian distribution of small-scale pixels is more important than expanding the Effective Receptive Field**.

As shown in Fig. 7, UniConvNet variants consistently

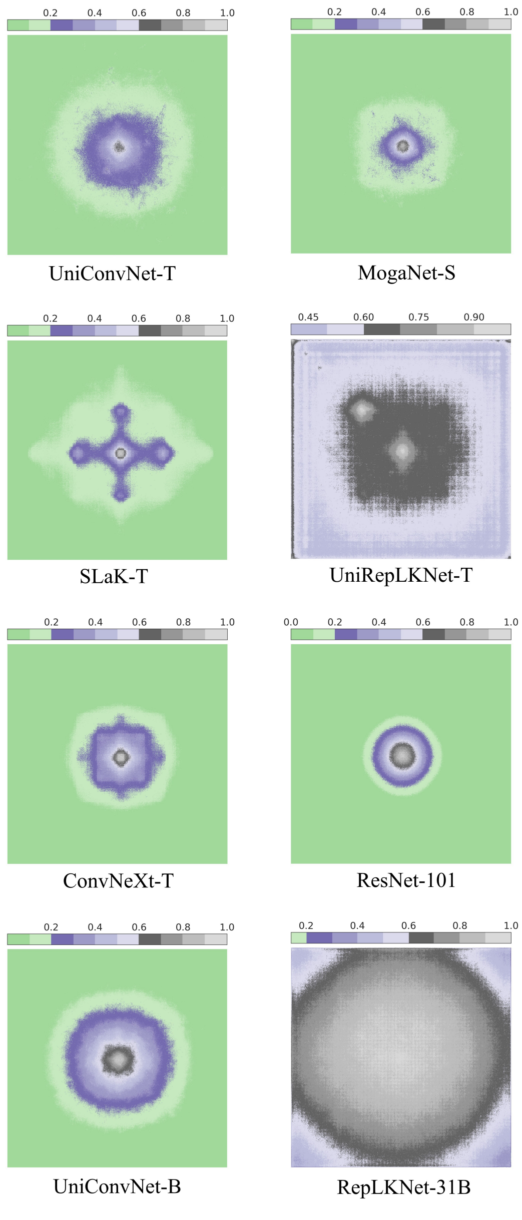

Figure 6. **Effective Receptive Field (ERF) of UniConvNet-T, MogaNet-S, SLaK-T, ConvNeXt-T, UniRepLKNet-T, ResNet-101 and UniConvNet-B, RepLKNet-31B.** The more stepped colour area indicates better AGD. The wider area indicates a larger ERF. Each ERF is based on an average of 1000 images with a resolution of 224 × 224.

demonstrate large ERF while maintaining AGD. These findings suggest that the Three-layer RFA can extend the ERF with proper combination of smaller kernels (e.g., 7×7, 9×9, 11×11).

B. Throughput Analysis Configuration

We use an A100 40GB GPU to benchmark throughput on several classic and relevant models. The software environment is PyTorch 1.13, CUDA 11.7, cuDNN 8.5. The hardware and software configurations align with InternImage [69] to ensure a fair comparison. The overall throughput of each model, measured as the number of images processed per second, is reported in FP32/FP16 data formats.

C. Illustration of UniConvNet Block

C.1. Stem & Downsampling Block

Similar to ConvNeXt [40] and InternImage [69], our model adopts a pyramid architecture with a stem block and three downsampling blocks to generate multi-scale feature maps. As shown in Fig. 5, The stem block, positioned before the first stage, reduces the input resolution by a factor of 4. The stem block employs a bottleneck design, comprising two stacked 3×3 convolution layers and LayerNorm layers, interspersed with a GELU activation function to introduce nonlinearity to input images. The 3×3 convolutions have strides of 2 and padding of 1. The first convolution output channels are half those of the second. The downsampling block between stages reduces the input resolution by a factor of 2. It consists of a LayerNorm layer followed by a 3×3 convolution with a stride of 2 padding of 1.

C.2. Basic Block

Inspired by the state-of-the-art CNN model InternImage, which integrates LayerNorm [2], feed-forward networks [65], and GELU [25], UniConvNet incorporates three stacked residual components in basic blocks. Each residual component begins with a LayerNorm layer to normalize input features, followed sequentially by the Three-layer RFA, modified DCNV3 [69], and a feed-forward network. The basic block initially employs the Three-layer RFA residual component to extract multi-scale features from small- and large-scale patterns, establishing long-range and multi-scale dependencies. Following the Three-layer RFA residual component, a 3×3 (modified DCNV3 [69]) residual convolution block and a feed-forward network are used for densely local perception, akin to conventional ConvNets.

D. Training Settings

D.1. ImageNet-1K/22K Training

We adopt the commonly used training recipes from state-of-the-art methods [18, 40, 62, 64, 69, 72, 81] and remove some tune-ups for fair comparisons and to better represent the effectiveness of the proposed UniConvNet. Additionally, we apply a weak training recipe, following EMO [81], to improve performance on smaller models (UniConvNet-A/P0/P1/P2), and a strong training recipe, based on common practice [40], for larger variants (UniConvNet-N0/N1/N2/N3/T/S/B). All experiments are conducted on the ImageNet-1K [14] dataset, comprising 1000 object classes and 1.2 million training images.

Super Prameters	**UniConvNet-A /P0/P1/P2(W)** ImageNet-1K	**UniConvNet-N0/N1 /N2/N3/T/S/B(S)** ImageNet-1K	**UniConvNet-L(S)** ImageNet-22K
Imput Scale	224^2	224^2	192^2
Training Epochs	300	300	90
Batch Size	4096	4096	4096
Optimizer	AdamW	AdamW	AdamW
Optimizer Momentum	$\beta_1, \beta_1 = 0.9, 0.999$	$\beta_1, \beta_1 = 0.9, 0.999$	$\beta_1, \beta_1 = 0.9, 0.999$
Base Learning Rate	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$
Learning Rate Schedule	cosine	cosine	cosine
Learning Rate Decay	$5e^{-2}$	$5e^{-2}$	$5e^{-2}$
Layer-wise Learning Rate Decay	✗	✗	✗
Warmup Epochs	20	20	20
Warmup Schedule	linear	linear	linear
Label Smoothing ε	0.1	0.1	0.1
Dropout Rate	✗	✗	✗
Drop Path Rate	0.05/0.05/0.05/0.08	0.08/0.1/0.1/0.1/0.2/0.4/0.6	0.2
Layer Scale	$1e^{-6}$	$1e^{-6}$	$1e^{-6}$
RandAugment	(9,0.5)	(9,0.5)	(9,0.5)
Color Jitter	0.4	0.4	0.4
Horizontal Flip	✗	✗	✗
Random Resized Crop	✗	✗	✗
Repeated Augment	✗	✗	✗
Head Init Scale	✗	✗	✗
Mixup Alpha	✗	0.8	0.8
Cutmix Alpha	✗	1.0	1.0
Erasing Probability	✗	0.25	0.25
Gradient Clip	✗	✗	✗
Loss	Cross Entropy	Cross Entropy	Cross Entropy
Exp. Mov. Avg. (EMA)	0.9999	0.9999	✗

Table 11. **(Pre-)Training settings for various model variants on ImageNet-1K/22K.** The training recipes adhere to standard practices [18, 40, 62, 64, 69, 72, 81], with certain tune-ups removed. Multiple stochastic depth drop rates (e.g., 0.08/0.1/0.1/0.1/0.2/0.4/0.6) are assigned to UniConvNet-N0/N1/N2/N3/T/S/B, respectively. "W" and "S" indicate that the UniConvNet variants are trained using the weak and strong training recipes, respectively.

Using the weak training recipe, we train UniConvNet-A/P0/P1/P2 models from scratch with 224×224 inputs for 300 epochs. The AdamW optimizer is used with a learning rate of 4×10^{-3}. Training begins with a 20-epoch linear warmup, followed by a cosine decay learning rate schedule. A batch size of 4096 and a weight decay of 0.05 are employed. RandAugment [12] is applied for data augmentation in the weak training recipe. Regularization techniques, including Stochastic Depth [28] and Label Smoothing [57], are employed. A Layer Scale [63] with an initial value of 1×10^{-6} is used. Exponential Moving Average (EMA) [48] is employed to reduce overfitting in larger models.

For UniConvNet-N0/N1/N2/N3/T/S/B, the strong training recipe is applied, incorporating additional data augmentation techniques such as Mixup [79], Cutmix [77], and Random Erasing [84], to enhancethe dataset for training on larger models. For UniConvNet-L, we follow the strong training recipe and change the input image size to 192×192.

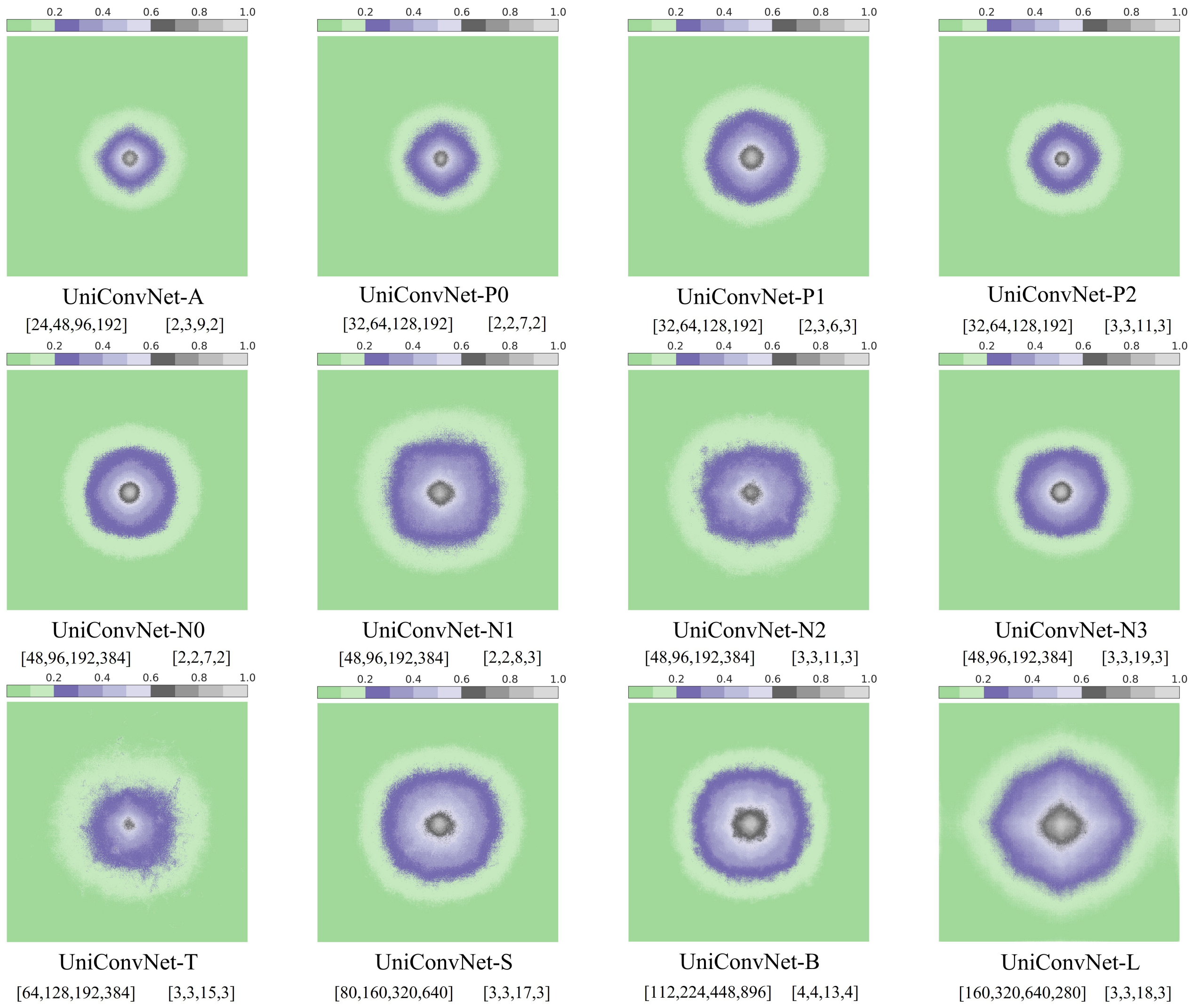

Figure 7. **Effective Receptive Field (ERF) of all UniConvNet variants.** The more stepped colour area indicates better AGD. The wider area indicates a larger ERF. Each ERF is based on an average of 1000 images with a resolution of 224×224.

Detailed training configurations for different model variants are provided in Tab. 11.

D.2. ImageNet-1K Fine-tuning

For ImageNet-1K fine-tuning, compared to the strong training recipes for ImageNet-1K/22K, the base learning rate of the AdamW optimizer is set to 5×10^{-5}. The learning rate decay is set to 1×10^{-8}. ImageNet-1K fine-tuning is performed with a batch size of 512, without requiring warm-up. Different layer-wise learning rate decay factors are applied: 0.7 for UniConvNet-T/S/B and 0.8 for UniConvNet-L. Data augmentation techniques, Mixup [79] and Cutmix [77], are removed to improve fine-tuning results. Additionally, ImageNet-1K pre-trained UniConvNet-T/S/B and ImageNet-22K pre-trained UniConvNet-L are fine-tuned at an increased resolution of 384×384.

D.3. Training Recipes for Classification

We evaluate the performance of UniConvNet-A/P0/P1/P2/N0/N1/N2 on ImageNet-1K to conduct an ablation study comparing weak and strong training recipes. As illustrated in Tab. 13, the weak training recipe exhibits overfitting with models having $10.2M$ parameters, while models with $13.2M$ parameters start benefiting from the strong training recipe. Consequently, the choice of training recipes is straightforward: UniConvNet-A/P0/P1/P2/N0 adopts the weak training recipe to leverage smaller-scale datasets. UniConvNet-N1/N2 and larger models employ the strong

Super Prameters	**UniConvNet-T/S/B** ImageNet-1K pt ImageNet-1K ft	**UniConvNet-L** ImageNet-22K pt ImageNet-1K ft
Imput Scale	384^2	384^2
Training Epochs	30	30
Batch Size	512	512
Optimizer	AdamW	AdamW
Optimizer Momentum	$\beta_1, \beta_1 = 0.9, 0.999$	$\beta_1, \beta_1 = 0.9, 0.999$
Base Learning Rate	$5e^{-5}$	$5e^{-5}$
Learning Rate Schedule	cosine	cosine
Learning Rate Decay	$1e^{-8}$	$1e^{-8}$
Layer-wise Learning Rate Decay	0.7	0.8
Warmup Epochs	✗	✗
Warmup Schedule	✗	✗
Label Smoothing ε	0.1	0.1
Dropout Rate	✗	✗
Drop Path Rate	0.4/0.6/0.8	0.3
Layer Scale	pre-trained	pre-trained
RandAugment	(9,0.5)	(9,0.5)
Color Jitter	0.4	0.4
Horizontal Flip	✗	✗
Random Resized Crop	✗	✗
Repeated Augment	✗	✗
Head Init Scale	0.001	0.001
Mixup Alpha	✗	✗
Cutmix Alpha	✗	✗
Erasing Probability	0.25	0.25
Gradient Clip	✗	✗
Loss	Cross Entropy	Cross Entropy
Exp. Mov. Avg. (EMA)	0.9999	0.9999

Table 12. **Fine-tuning settings for various model variants on ImageNet-1K.** The training recipe follows common practices [18, 40]. Multiple stochastic depth drop rates (e.g., 0.4/0.6/0.8) are for UniConvNet-T/S/B, respectively. "ImageNet-1K pt", "ImageNet-1K ft", and "ImageNet-22K pt" represent ImageNet-1K pre-training, ImageNet-1K fine-tuning and ImageNet-22K pre-training, respectively.

UniConvNet	#Params	ACC-W(%)	ACC-S(%)
UniConvNet-A	3.4M	**77.0**	✗
UniConvNet-P0	5.2M	**79.1**	✗
UniConvNet-P1	6.1M	**79.6**	78.8
UniConvNet-P2	7.6M	**80.5**	79.9
UniConvNet-N0	10.2M	**81.7**	81.6
UniConvNet-N1	13.1M	81.8	**82.2**
UniConvNet-N2	15.0M	✗	**82.7**

Table 13. **Explorations of Training Recipes for Classification.** "ACC-W(%)" and "ACC-S(%)" are the TOP-1 accuracy trained by weak traininig recipe and strong traininig recipe, respectively.

training recipe, utilizing augmented [77, 79, 84] datasets to optimize performance with larger parameter sizes.

D.4. Object Detection and Instance Segmentation Fine-tuning

Following EMO [81], we utilize the standard MMDetection [6] library and the AdamW [41] optimizer to train the heavy RetinaNet and light SSDLite models with a batch size of 16 on 8 A100 GPUs.

Following standard practices [40, 69, 72], we further fine-tune the scaled-up UniConvNet using batch sizes of 16 and 8, respectively, for fair comparisons. Under the $1\times$ schedule, images are resized so that the shorter side is 800 pixels and the longer side does not exceed 1333 pixels. During testing, the shorter side is fixed at 800 pixels. Under the $3\times$ schedule, the longer side remains capped at 1333 pixels, while the shorter side is resized to a range of 480–800 pixels. We also employ the standard MMDetection [6] library and the AdamW [41] optimizer for training, using a base learning rate of 1×10^{-4}.

D.5. Semantic Segmentation Fine-tuning

We fine-tune DeepLabv3 [7] and PSPNet [83] using the ImageNet-1K pre-trained UniConvNet on the ADE20K [85] dataset. Following EMO [81], we use the MMSegmentation [11] library and the AdamW [41] optimizer to train DeepLabv3 [7] and PSPNet [83] for 160k iterations on 8 A100 GPUs, ensuring fair comparisons.

The scaled-up UniConvNet is fine-tuned with the UperNet framework on ADE20K for 160k iterations, using a batch size of 16. The AdamW [41] optimizer is used for training. The base learning rates are set to 6×10^{-5} for UniConvNet-N2/N3/T/S/B and 2×10^{-5} for UniConvNet-L. A polynomial decay schedule with a power of 1.0 is applied for learning rate decay. Following common practices [38, 40, 69, 72], images are cropped to 512×512 for UniConvNet-N2/N3/T/S/B and 640×640 for UniConvNet-L to ensure fair comparisons.