Dynamic Convolution Neural Network

Team ^^ | 21기 김지엽, 박세진, 22기 조해진

KUBIG
DATA SCIENCE & AI

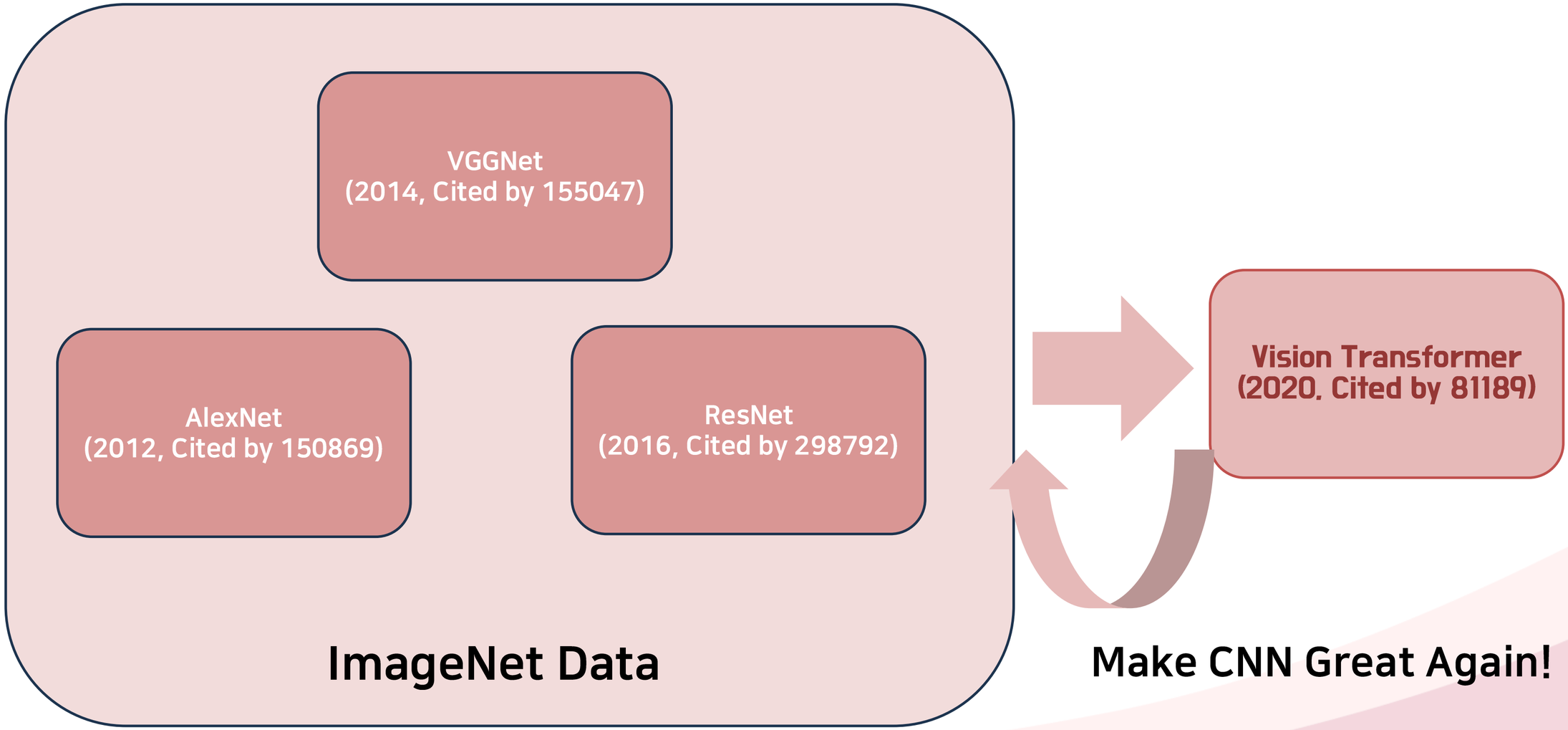
KUBIG
DATA SCIENCE & AI

CONTENTS

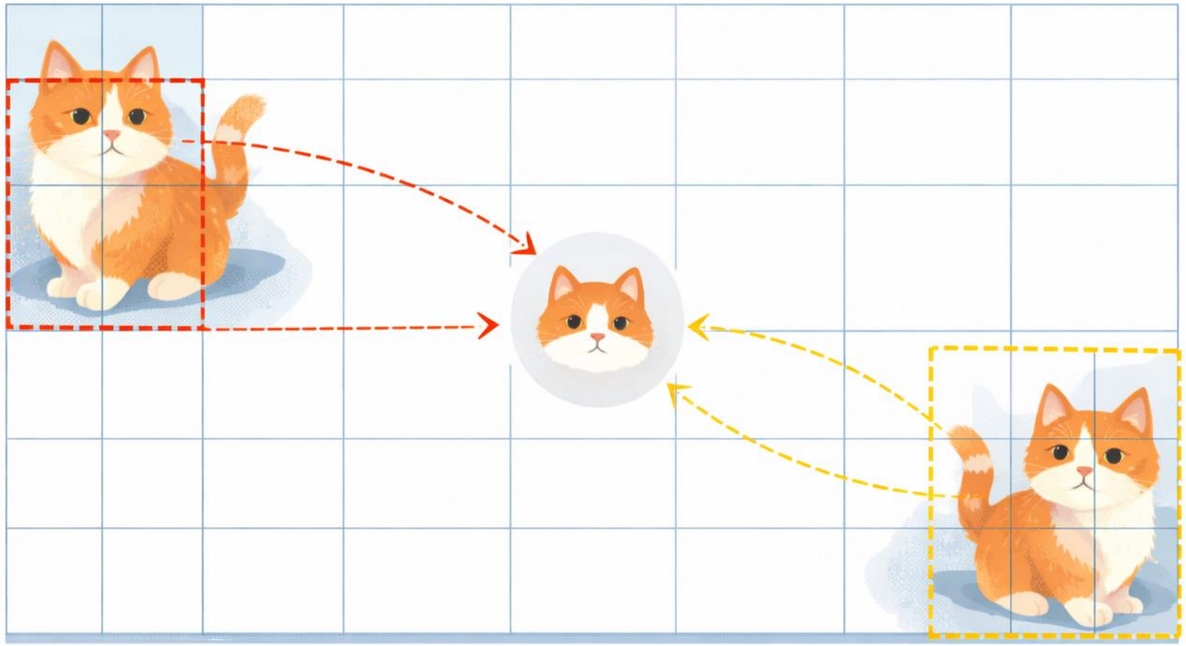
KUBIG
DATA SCIENCE & AI

01. Introduction

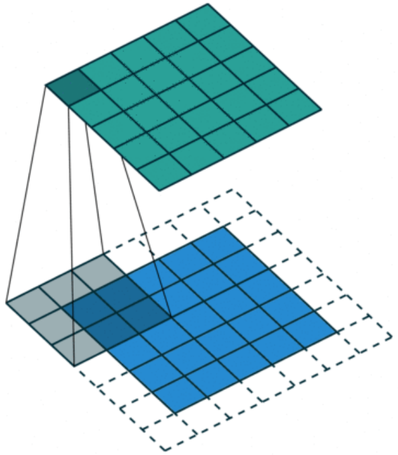
01. Beyond CNN



01. Inductive Bias of CNN: Locality



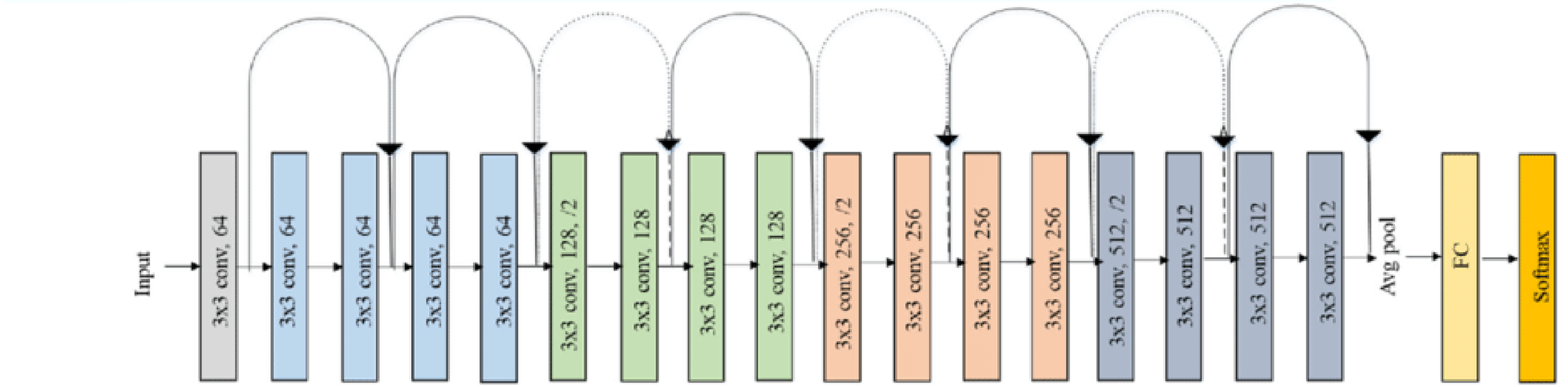
CNN's Inductive Bias: Locality



출처: https://deeplizard.com

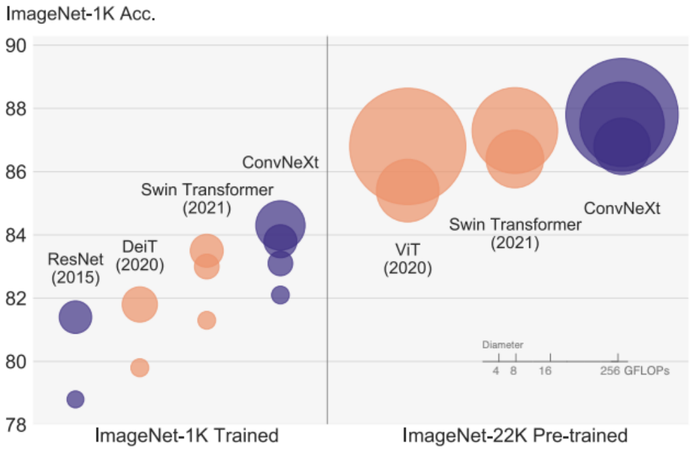
동일한 Filter 를 합성곱 형식으로 이미지 전체에 Sliding 시키며 구현

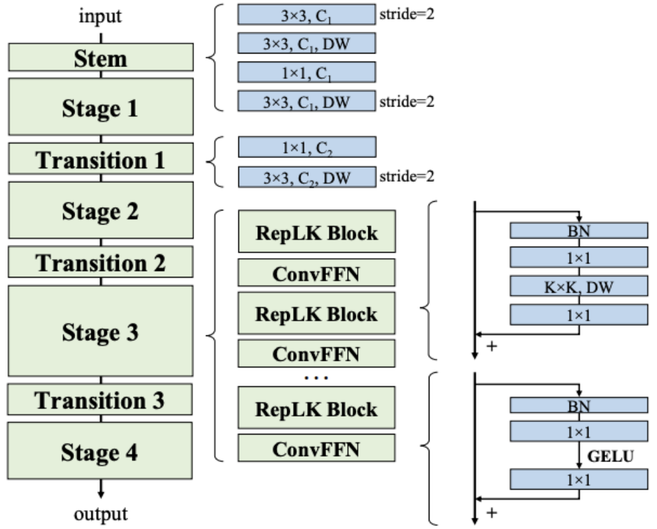
01. Limitation of CNN: Small ERF

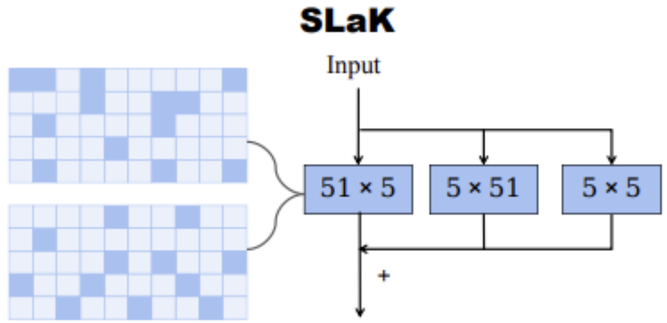


- ERF(Effective Receptive Field): 후반부 레이어가 참조하는 입력 피처맵의 그래디언트 분포
- **CNN**은 layer 가 깊어질수록 ERF가 점진적으로, 규칙적으로 (Gaussian) 커진다
- **ViT**는 self-attention 을 통해 ERF가 이미지 전체가 되어, 멀리 있는 픽셀도 바로 상호작용 가능

01. ConvNeXt and Large Kernels







ConvNeXT
:ViT의 구조를 CNN 에 접목

RepLKNet
:31x31 kernel Reparametrization

SLaK
:51x51 kernel Sparse 하게 분해하여 사용

01. Adaptive and Shiftwise Convolution

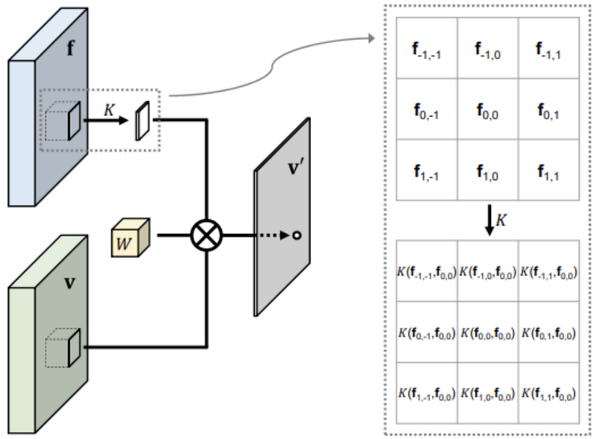
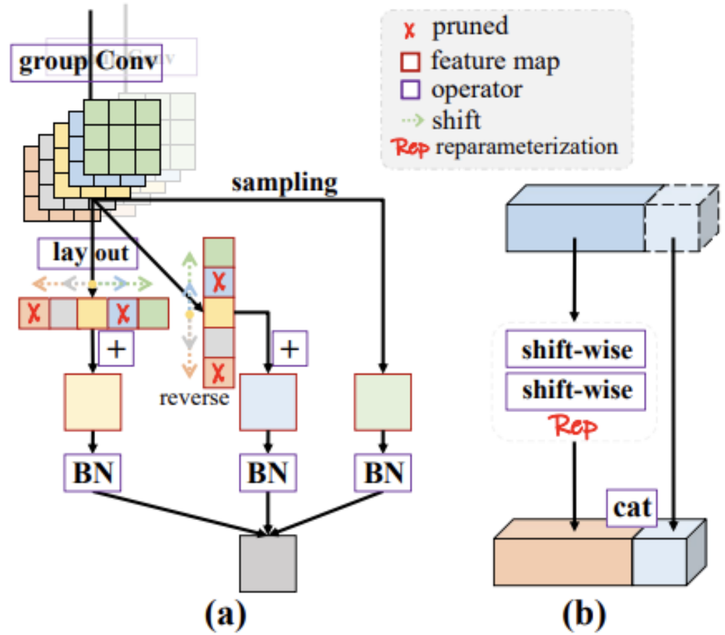


Figure 1: **Pixel-Adaptive Convolution.** PAC modifies a standard convolution on an input $\mathbf{v}$ by modifying the spatially invariant filter $\mathbf{W}$ with an adapting kernel K. The adapting kernel is constructed using either pre-defined or learned features $\mathbf{f}$. $\otimes$ denotes element-wise multiplication of matrices followed by a summation. Only one output channel is shown for the illustration.

Pixel-Adaptive CNN

:Kernel의 가중치를 픽셀별로 학습



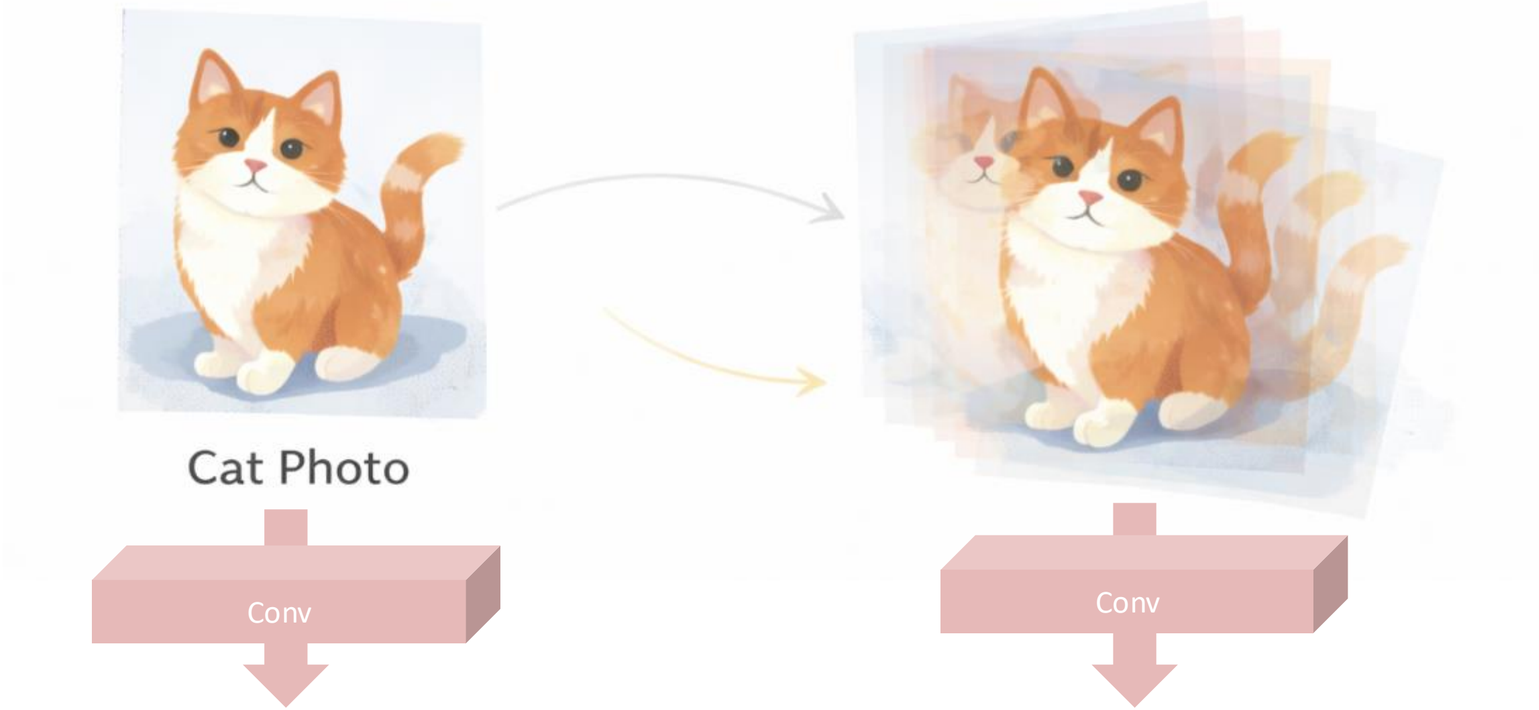
Shiftwise Convolution

:small kernel들을 shift시켜 large kernel 근사

KUBIG
DATA SCIENCE & AI

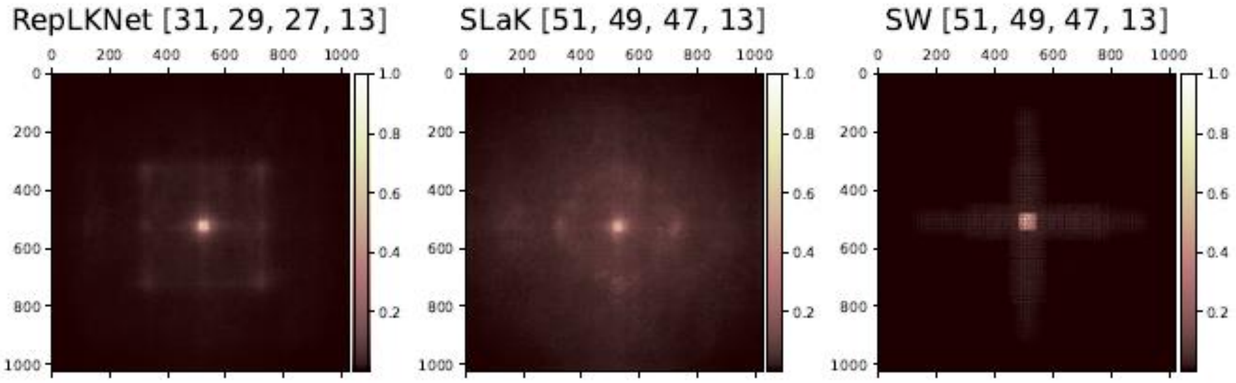
02. Motivation

02. Idea 1: Jittering



- 채널별 Jittering(흔드는 과정)을 통해 Noise 를 주면 ERF 가 더 커지지 않을까?
- Random하게 Feature map을 흔들면 강한 augmentation없이도 더 robust(강건)한 feature을 학습할 수 있지 않을까?

02. Idea 2: Rotation



ShiftWise Convolution은 작은 커널로 큰 커널의 효과를 냄
→ ERF가 확장되긴 하지만 가로세로 방향으로만 분포함
→ERF의 분포를 다양화 해보자!
→ 대각선 방향의 특성이 중요한 데이터
→ 예를 들어 분자 구조나 텍스쳐, 패턴 데이터 등에 대해 성능 향상 예측

02. Idea 3: Pixel Mix

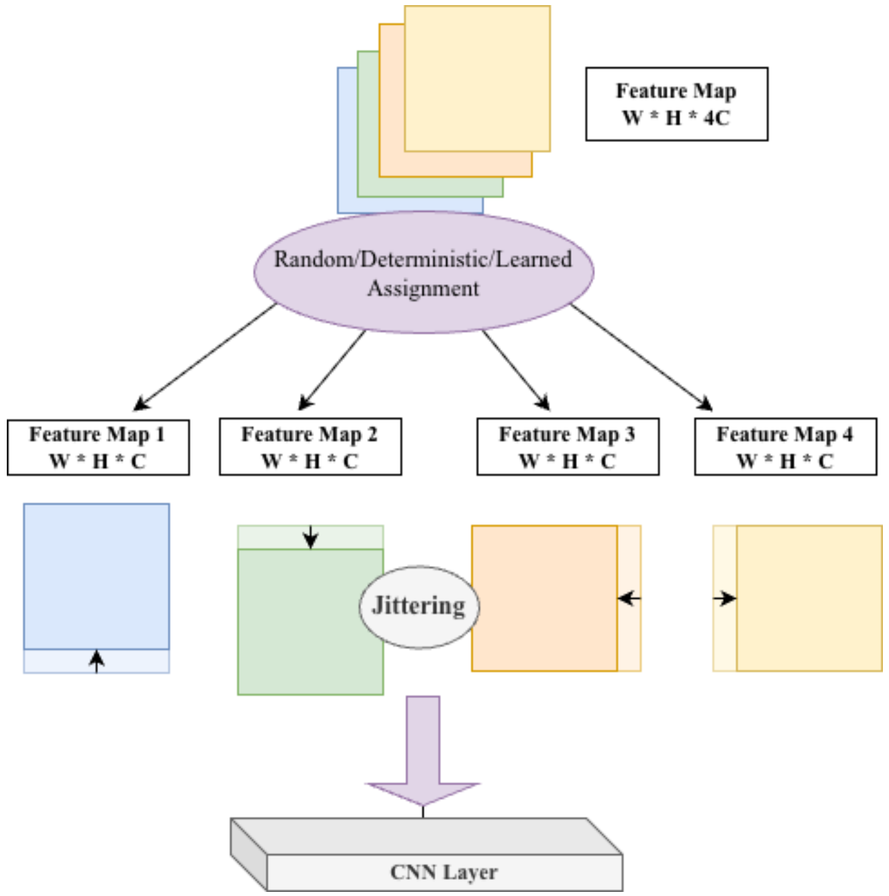
현실에 Edge Device들에 들어갈 수 있는 네트워크들은 보통 MLP 8-9층을 넘어가지 못함
→ 또한, Cache에서 돌아가기 위해선 수십 KB가 한계
→ 작아지는 RF Size, 그로 인해 성능 하락 → 이걸 Dynamic Conv로 해결해보자 !

	Method	Runtime	Size	Set5		Set14		BSDS100		Urban100		Manga109	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Interpolation	Nearest	4ms	-	26.25	0.7372	24.65	0.6529	25.03	0.6293	22.17	0.6154	23.45	0.7414
	Bilinear	16ms	-	27.55	0.7884	25.42	0.6792	25.54	0.6460	22.69	0.6346	24.21	0.7666
	Bicubic	60ms	-	28.42	0.8101	26.00	0.7023	25.96	0.6672	23.14	0.6574	24.91	0.7871
SR-LUT	Ours-V	**15ms**	1MB	29.22	0.8304	26.65	0.7258	26.33	0.6880	23.68	0.6852	26.30	0.8246
	Ours-F	34ms	**77KB**	29.77	0.8429	26.99	**0.7372**	**26.57**	**0.6990**	23.94	0.6971	**26.87**	0.8367
	Ours-S	91ms	1.274MB	**29.82**	**0.8478**	**27.01**	0.7355	26.53	0.6953	**24.02**	**0.6990**	26.80	**0.8380**
Sparse coding	NE + LLE [6]	7016ms*	1.434MB	29.62	0.8404	26.82	0.7346	26.49	0.6970	23.84	0.6942	26.10	0.8195
	Zeyde *et al.* [50]	8797ms*	1.434MB	26.69	0.8429	26.90	0.7354	26.53	0.6968	23.90	0.6962	26.24	0.8241
	ANR [42]	1715ms*	1.434MB	29.70	0.8422	26.86	0.7368	26.52	0.6992	23.89	0.6964	26.18	0.8214
	A+ [43]	1748ms*	15.171MB	30.27	0.8602	27.30	0.7498	26.73	0.7088	24.33	0.7189	26.91	0.8480
DNN	FSRCNN [9]	371ms	12K†	30.71	0.8656	27.60	0.7543	26.96	0.7129	24.61	0.7263	27.91	0.8587
	CARN-M [2]	4955ms	412K†	31.82	0.8898	28.29	0.7747	27.42	0.7305	25.62	0.7694	29.85	0.8993
	RRDB [44]	31717ms	16698K†	32.68	0.8999	28.88	0.7891	27.82	0.7444	27.02	0.8146	31.57	0.9185

KUBIG
DATA SCIENCE & AI

03. Method

03. Method 1



Settings

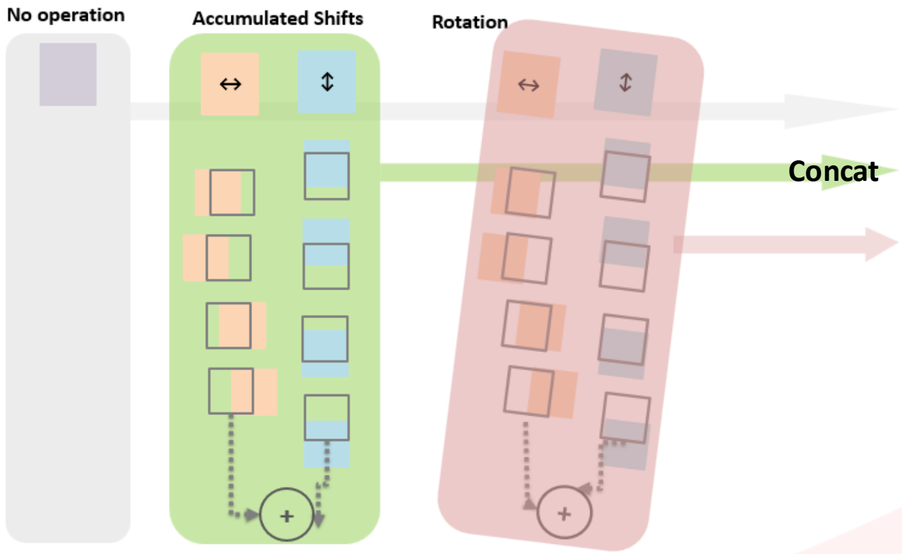
- ResNet-18 구조 사용,
  Stem(filter: 3x3, Channel: 3 → 64) 이후
  다양한 jittering method 적용
- Jittering 제외한 모든 구조, 학습조건, Random Seed는
  모두 동일하게 설정하여 공정하게 성능 비교

Jittering Methods

1. 각 이미지마다 Random 하게 1-pixel Jittering
   (상, 하, 좌, 우 각 16개씩 등)
2. Index(1~64)에 따라 deterministic 하게 1 pixel
   Jittering (상, 하, 좌, 우 각 16개씩)
3. Random 하게 0.5 pixel Jittering, 선형
   보간(interpolation)을 통해 구현
4. Learnable Jittering: 각 channel 이 어디로 jittering
   될지를 weight로 두고 학습

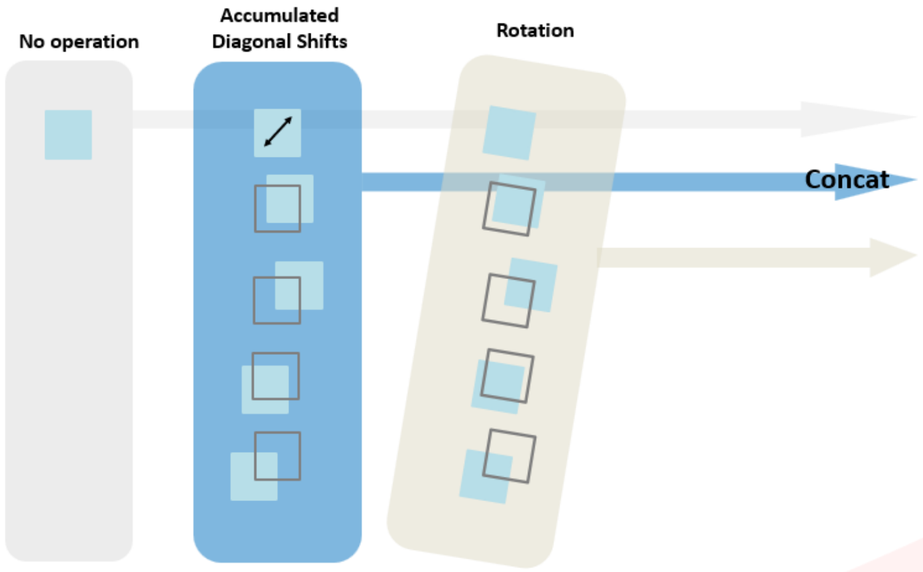
03. Method 2

Stem (3x3 Conv2d)	1차 압축
Channel 확장 (5x5 Conv2d)	기초 locality feature 추출 & 채널 수 확장
Shift Operation	상하좌우로 shift한 후 summation **커널 확장 효과**
Rotational Operation	회전 param 따라 **회전** **대각 방향 dependency 학습 기대**
Pointwise Convolution	PW Conv으로 채널간 믹싱



03. Method 2

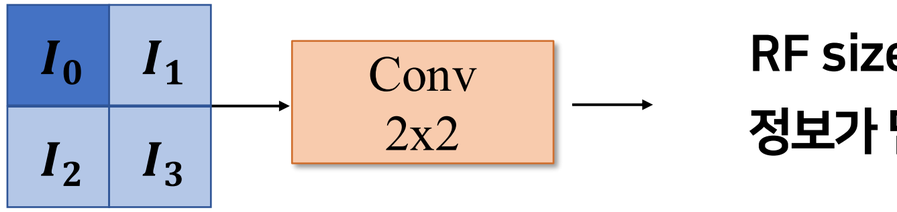
Stem (3x3 Conv2d)	1차 압축
Channel 확장 (5x5 Conv2d)	기초 locality feature 추출 & 채널 수 확장
Diagonal Shift Operation	**y=x 방향으로 shift** 후 summation **대각 방향 커널 확장 효과**
Rotational Operation	이전 단계 복제 후 회전. Output을 다시 Summation
Pointwise Convolution	PW Conv으로 채널간 믹싱



03. Method 3

Efficient SR 네트워크들의 구조

→첫 입력 Conv 2x2 → 그 다음 부턴 1x1



RF size 만큼의 locality에 대한 정보가 담긴 feature map

여기부터는 서로 다른 픽셀 사이의 관계가 학습되지 않음

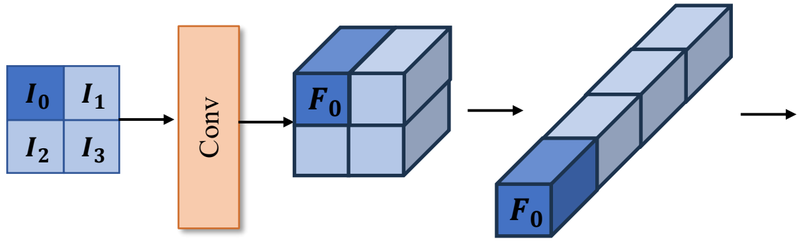
Q. 입출력은 동일하되 서로 간의 관계도 학습하고..

Locality도 공유하고 그럴 수 없을까?

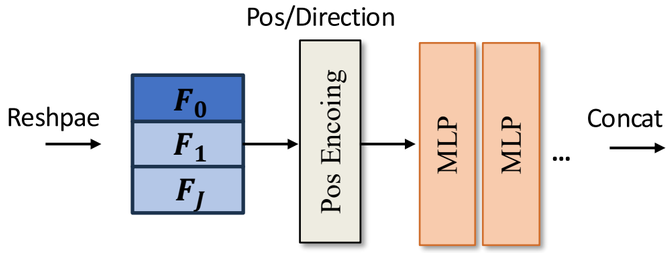
03. Method 3

Pixel-level Mix

• 첫 입력 Conv → Feature : Kernel size만큼의 Locality를 파악한 Feature Vector



Feature Vector Shift & concat
→ 중심 좌표 기준 주변 픽셀에서 뽑힌
Locality를 간접적으로 학습



중심 좌표 기준에서 서로 다른 픽셀에서 뽑힌
Locality를 직접적으로 학습
→ Position 정보와 Feature vector를 통해
한 픽셀에서 다른 픽셀을 바라보는 관계 학습

Pooling이 없어서 Translation invariance가 사라진 구조에서
Transformer같이 위치 정보를 활용 + 일종의 Attention같은 효과

KUBIG
DATA SCIENCE & AI

04. Results & Conclusion

04. Result 1 (Accuracy)

Train / Validation : **CIFAR-100**

Metric: 각 Jittering method별로 동일한 4개 seed에 대해 120 epoch 학습, Test Accuracy 평균과 표준편차 제공

Method	Mean Accuracy	Std
baseline	77.62	0.16
shift_det	77.18	0.24
shift_learnable	77.01	0.21
shift_float_0.5	76.65	0.31
shift_int	76.17	0.21

결과 및 해석

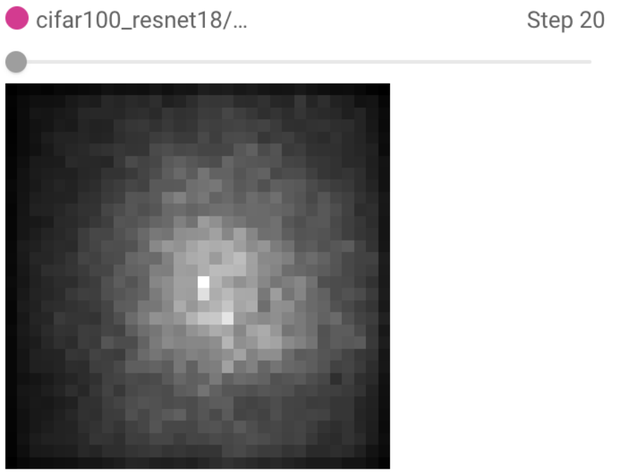
→ Jittering 적용하지 않은 Baseline ResNet-18이 가장 좋은 성능을 보임.

→ 채널별로 jittering 되는 방향을 정해주는 Deterministic Shift가 매번 shift 되는 방향이 달라지는 Random Shift 보다 좋은 결과를 보였는데, 이미지별로 jittering 되는 방향이 달라지면 일관된 학습이 이루어지지 못하기 때문.

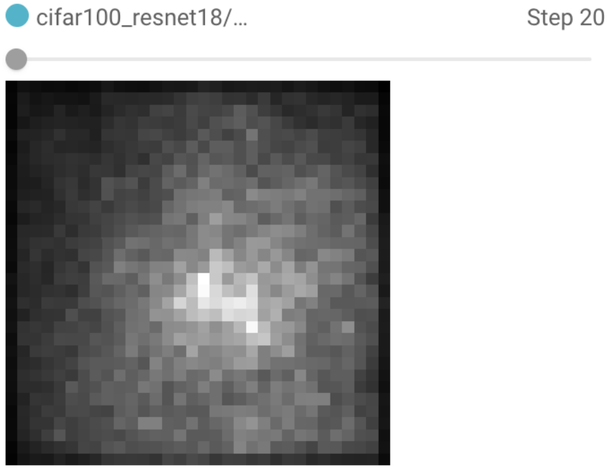
→ Interpolation을 적용해야 하는 0.5-pixel shift는 낮은 성능을 보임

→ Learnable Shift는 identity를 포함하지만, 학습이 어려워져 성능이 악화됨.

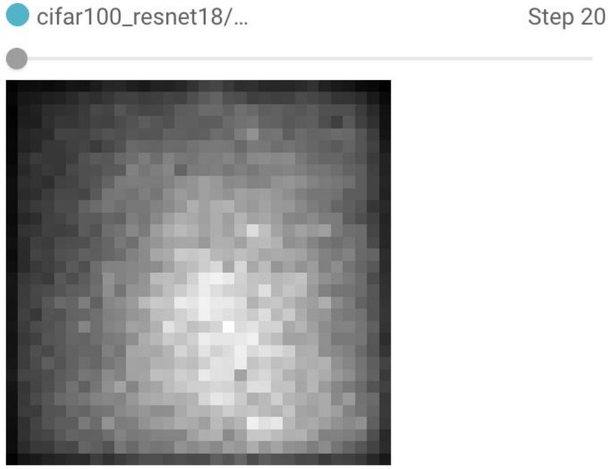
04. Result 1 (ERF)



ERF Baseline(No Jittering)



ERF Deterministic Shift



ERF Learnable Shift

- 반면, 20-epoch ERF를 비교해보면, Shift를 적용한 모델들에서 ERF가 훨씬 넓게 나타남
- 즉, 멀리 있는 pixel과의 상호작용을 더 잘 인식하며 CNN locality의 한계를 일부 극복.
- 하지만 성능 향상은 보이지 못했기 때문에, ERF가 넓더라도 학습이 불안정하거나 데이터 왜곡이 발생하면 성능이 떨어진다는 것을 알 수 있음.
- Jittering의 강도를 조절하거나, learnable shift를 더 안정적으로 학습시키는 방법 고민 필요

04. Result 2

Train / Validation : **CIFAR-100**

Zero-shot test: 텍스쳐 데이터셋으로 흔히 쓰이는 DTD Dataset 이용

Method	CIFAR100	DTD
SW+Erasing	84.04	41.67
RotSW+Cutmix	84.04	40.34
RotSW+Erasing	84.57	39.89
DiaRotSW+mixup	84.22	**42.02**
DiaRotSW+earsing	**84.67**	41.67
DiaSW+erasing	83.86	41.40

모델 성능

→ CIFAR-100 validation과 DTD 모두 Kernel을 대각선 방향으로 움직이는 operation과 회전 operation을 포함한 모델(DiaRotSW)이 성능이 좋았음.

ERF 다양화

→ 대각선 방향의 ERF 분포 다양화는 DiaRotSW에서만 확인할 수 있었음.

→ 최고 활성화 영역이 중앙부와 주변부에 연속적으로 나타나는 경우 DTD에서 좋은 성능을 보였음.

04. Result 3

Train: DIV2K 800/100 image set

Test: Set5, Set14와 같은 Benchmark Dataset 활용

→ 실제로 Shift 아이디어 적용시 성능 향상을 확인

→ 4배 Super-Resolution 기준

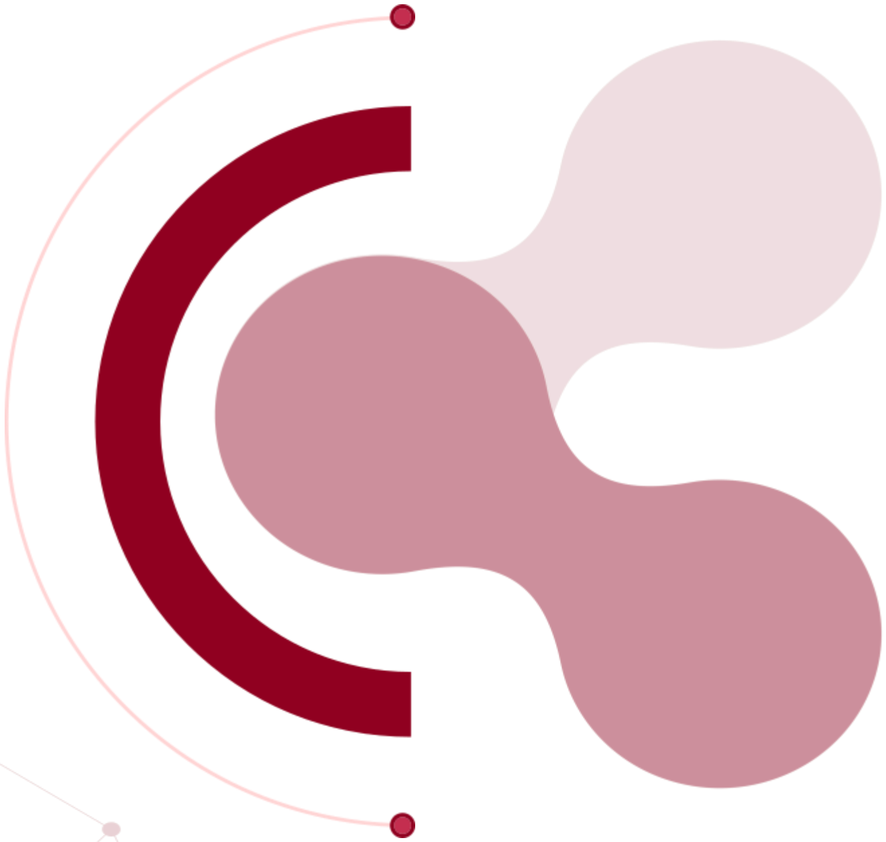
Method	X4 SR, Set5	X4 SR, Set14
SR-LUT	29.90	27.01
SR-LUT + Shift	**30.03**	**27.29**
MuLUT	30.40	27.60
MuLUT+Shift	**30.50**	**27.72**

04. Conclusion

- CNN의 장점이자 한계인 Locality라는 Inductive Bias를 유지하면서, 다양한 convolution 구조와 변형을 활용하여 ERF를 늘리고 CNN의 한계를 극복하는 방법을 고민
- 기존에 있는 모델을 그대로 쓰지 않고, 함께 논문들을 읽고 아이디어를 얻어서 이를 직접 구현

- Pixel-level Mix CNN의 경우 확실한 성능 향상을 보임.
- DiaRotSW의 경우 텍스쳐 데이터에 특화된 operator를 제안하고 성능을 확인했으나 모델과 ERF간 관계, ERF와 모델 성능 간 관계를 이론적으로 명확히 규정하지 못함.
- ConV 전 Jittering의 경우 ERF는 분명하게 늘어남을 확인할 수 있었으나, Jittering 과정에서 학습이 불안정해지고 성능에 한계를 보임.

KUBIG
DATA SCIENCE & AI



Thank you