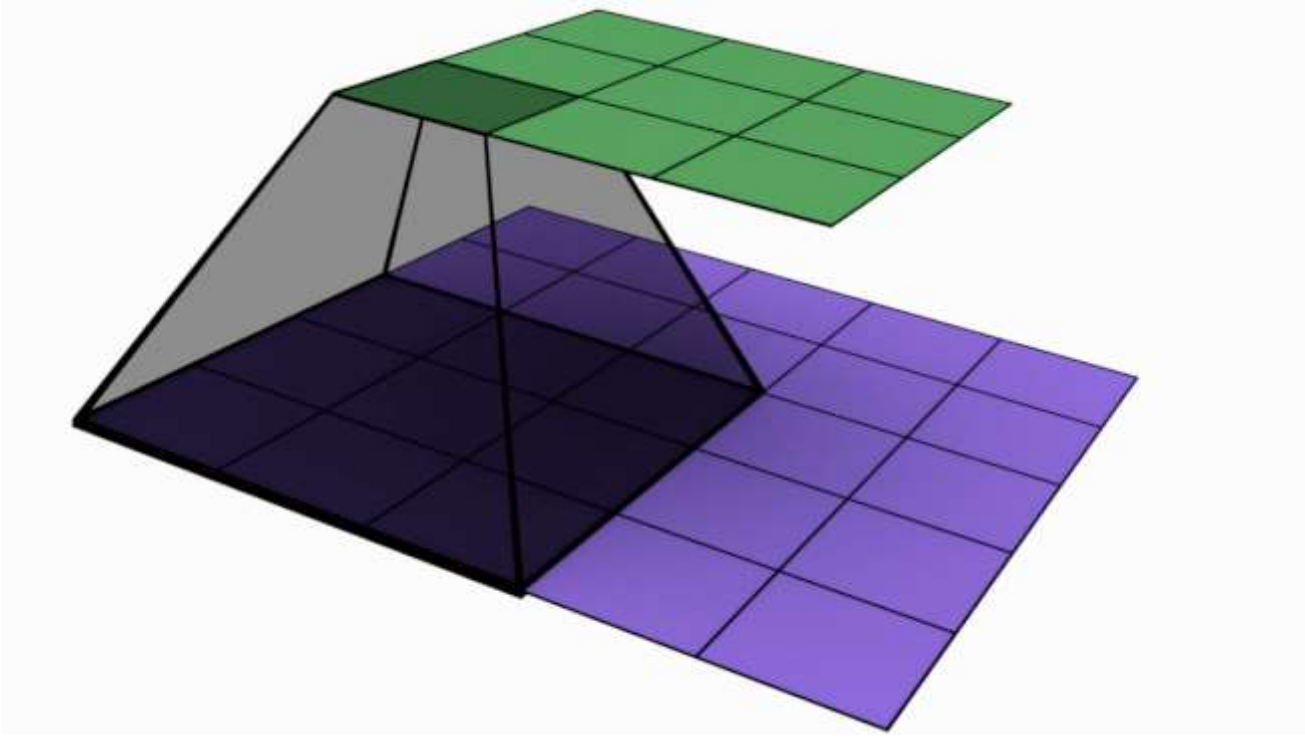
Session 9 - Advanced Convolutions, Data Augmentation and Visualization;

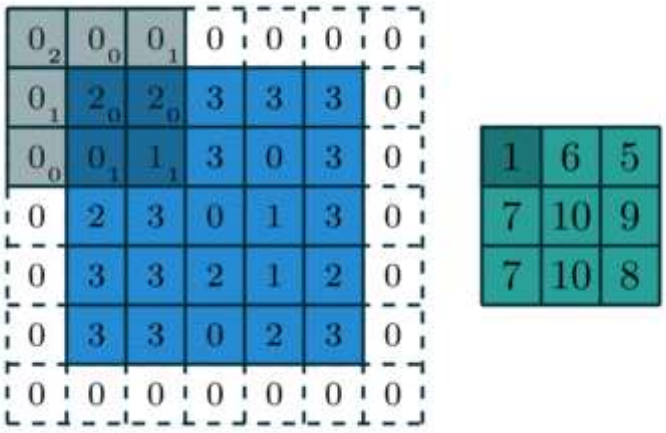
- Due Saturday by 9am
- Points 0
- Available after Mar 23 at 10am

SESSION 9 - Advanced Concepts, Data Augmentation & Visualizations

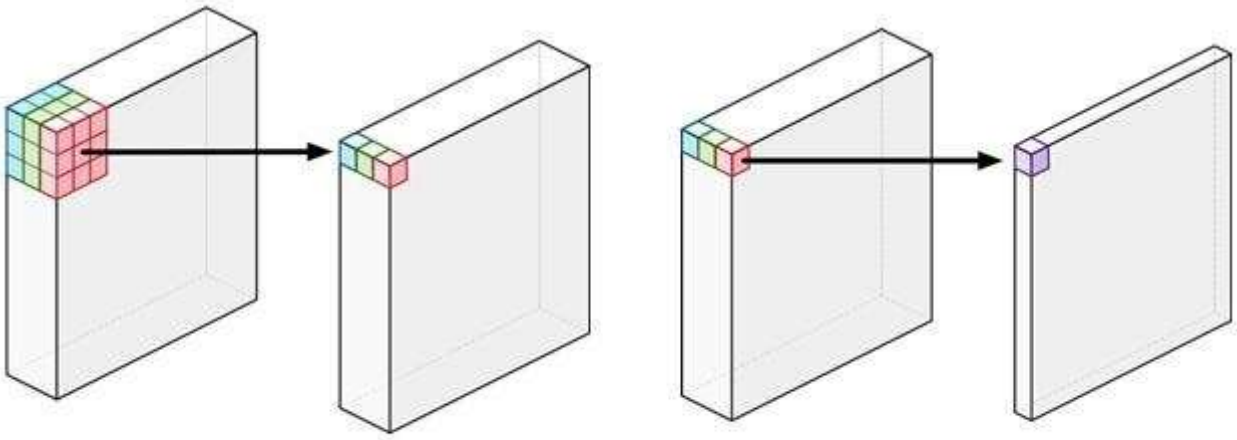
Advanced Convolutions

Normal Convolutions

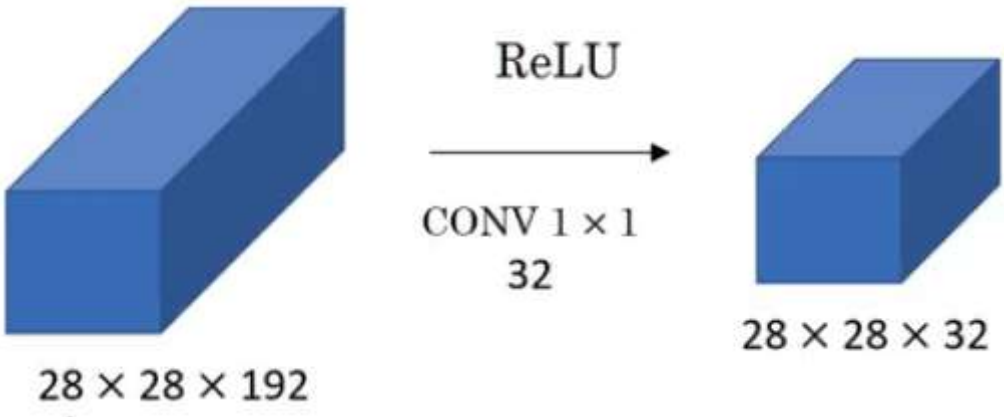



0 0 0 0 0 0 0
0 2 2 3 3 3 0
0 0 1 3 0 3 0
0 2 3 0 1 3 0
0 3 3 2 1 2 0
0 3 3 0 2 3 0
0 0 0 0 0 0 0
1 6 5
7 10 9
7 10 8

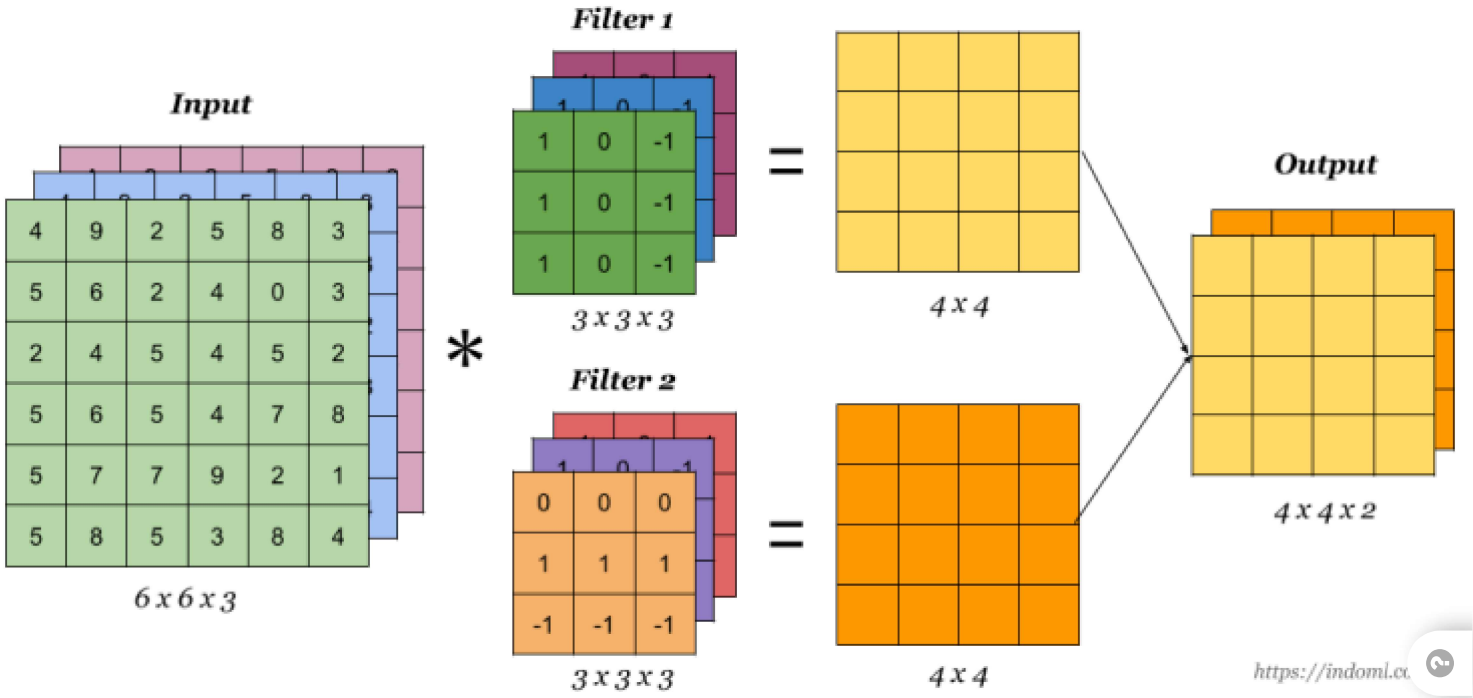
Pointwise Convolutions



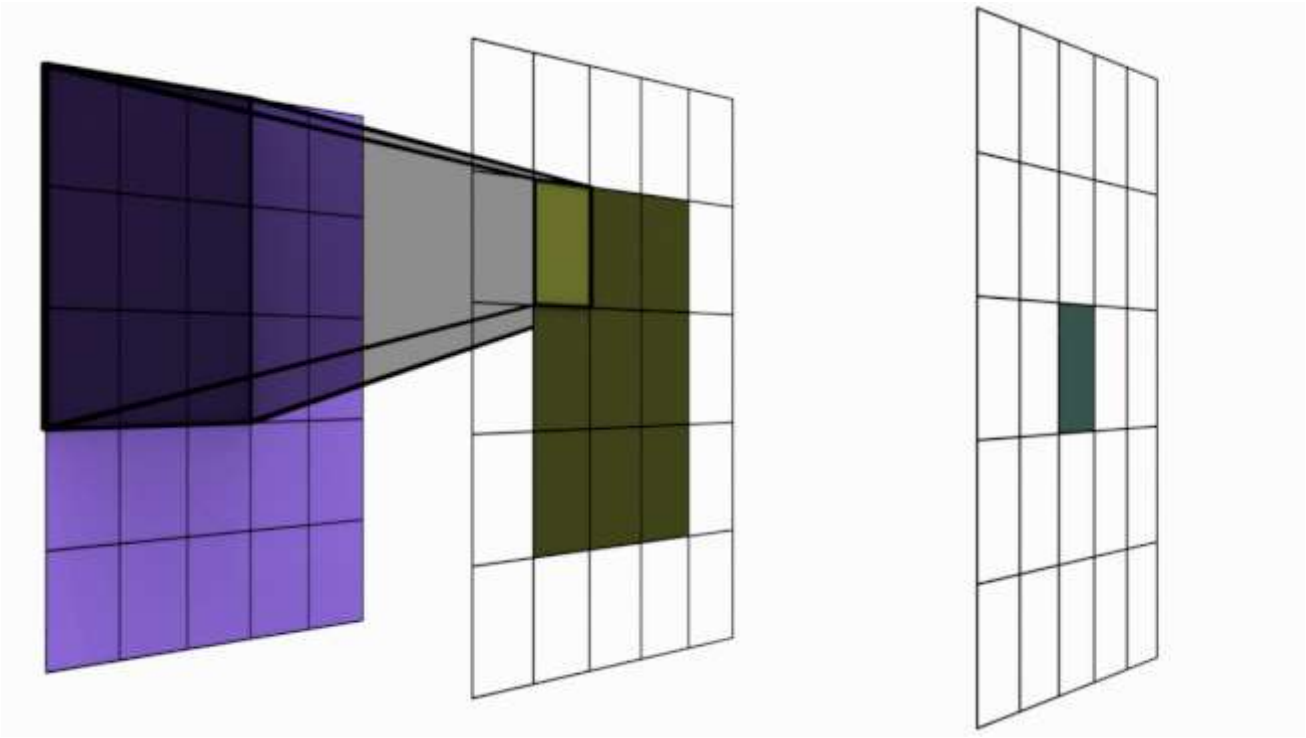
a) depthwise convolution b) pointwise convolution

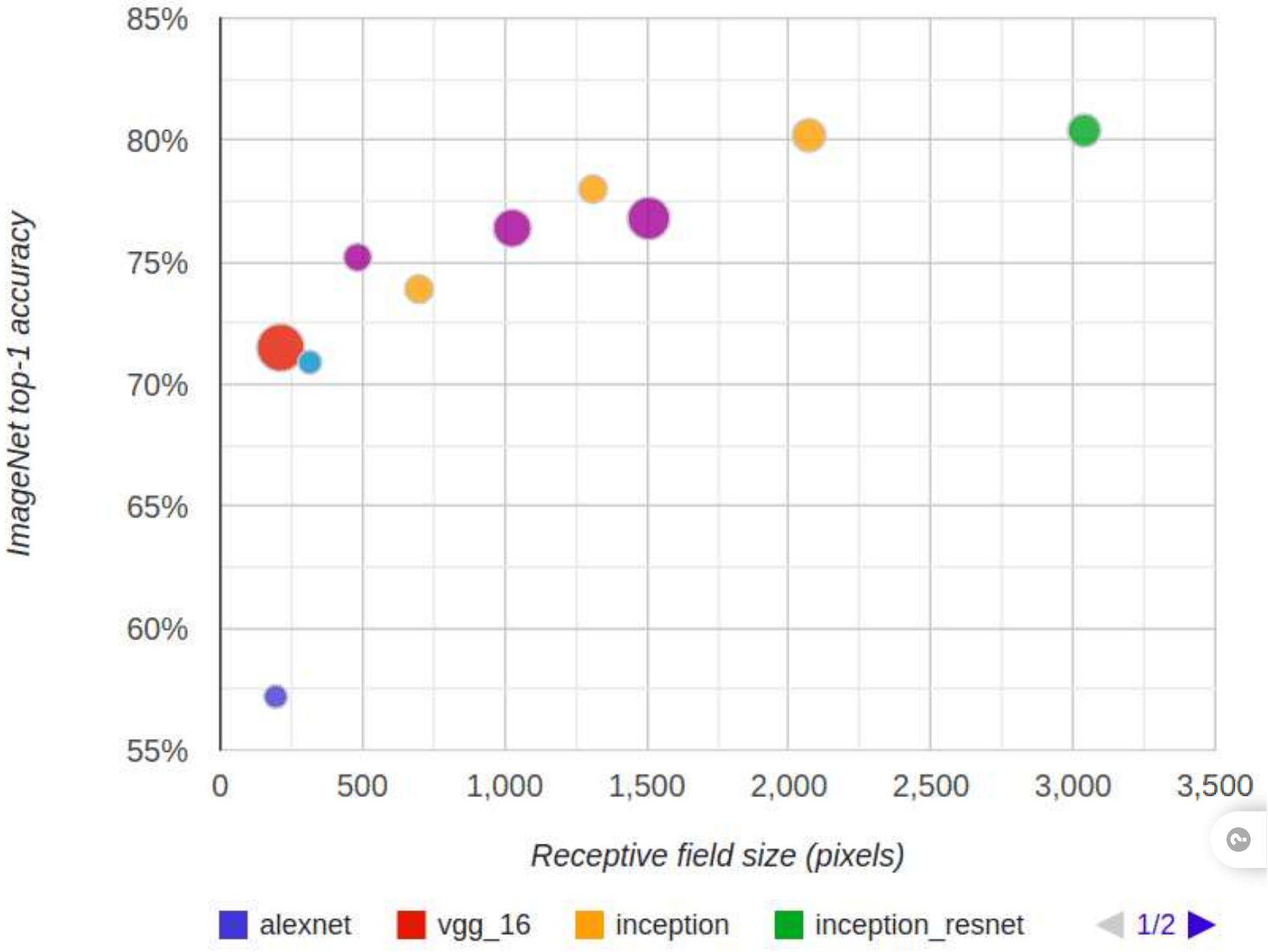


Concept of Channels



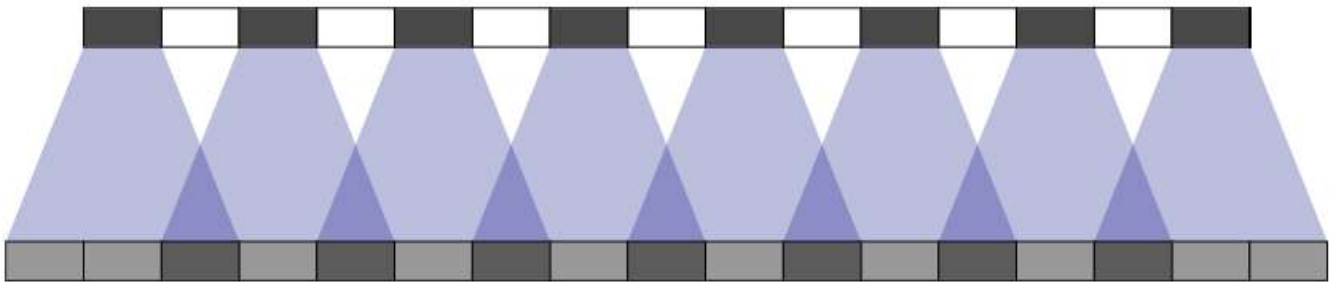
Receptive Fields

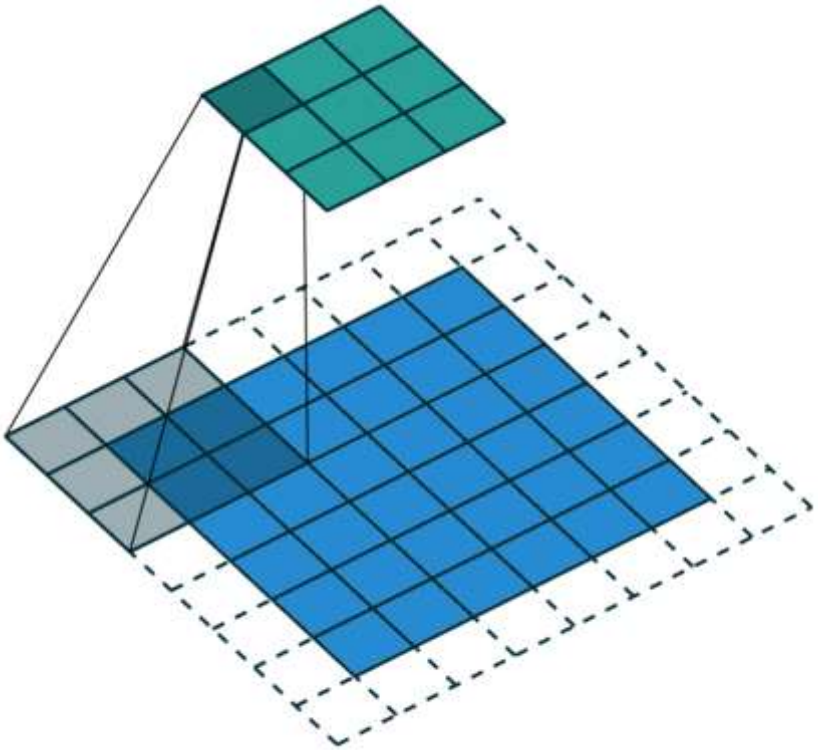




We observe a logarithmic relationship between classification accuracy and receptive field size, which suggests that large receptive fields are necessary for high-level recognition tasks, but with diminishing rewards. For example, note how MobileNets achieve high recognition performance even if using a very compact architecture: with depth-wise convolutions, the receptive field is increased with a small compute footprint. In comparison, VGG-16 requires 27X more FLOPs than MobileNets, but produces a smaller receptive field size; even if much more complex, VGG's accuracy is only slightly better than MobileNet's. This suggests that networks that can efficiently generate large receptive fields may enjoy enhanced recognition performance.[REF (https://distill.pub/2019/computing-receptive-fields/)]

Strides & Checkerboard Issue





How do we get 5x5 RF with one 3x3 Kernel?

or

Atrous or DIlated Convolutions

We get a receptive field of 3x3 when we convolve by 3x3.

What should we do to get a receptive field of 5x5 when we convolve by 3x3?

Dilated convolution increases the receptive view (global view) of the network exponentially and linear parameter accretion.

With this purpose, it finds usage in applications cares more about integrating the knowledge of the broader context with less cost.

The key application the dilated convolution authors have in mind is a **dense prediction**: vision applications where the predicted object has a similar size and structure to the input image.

For example,

panoptic/semantic segmentation with one label per pixel;

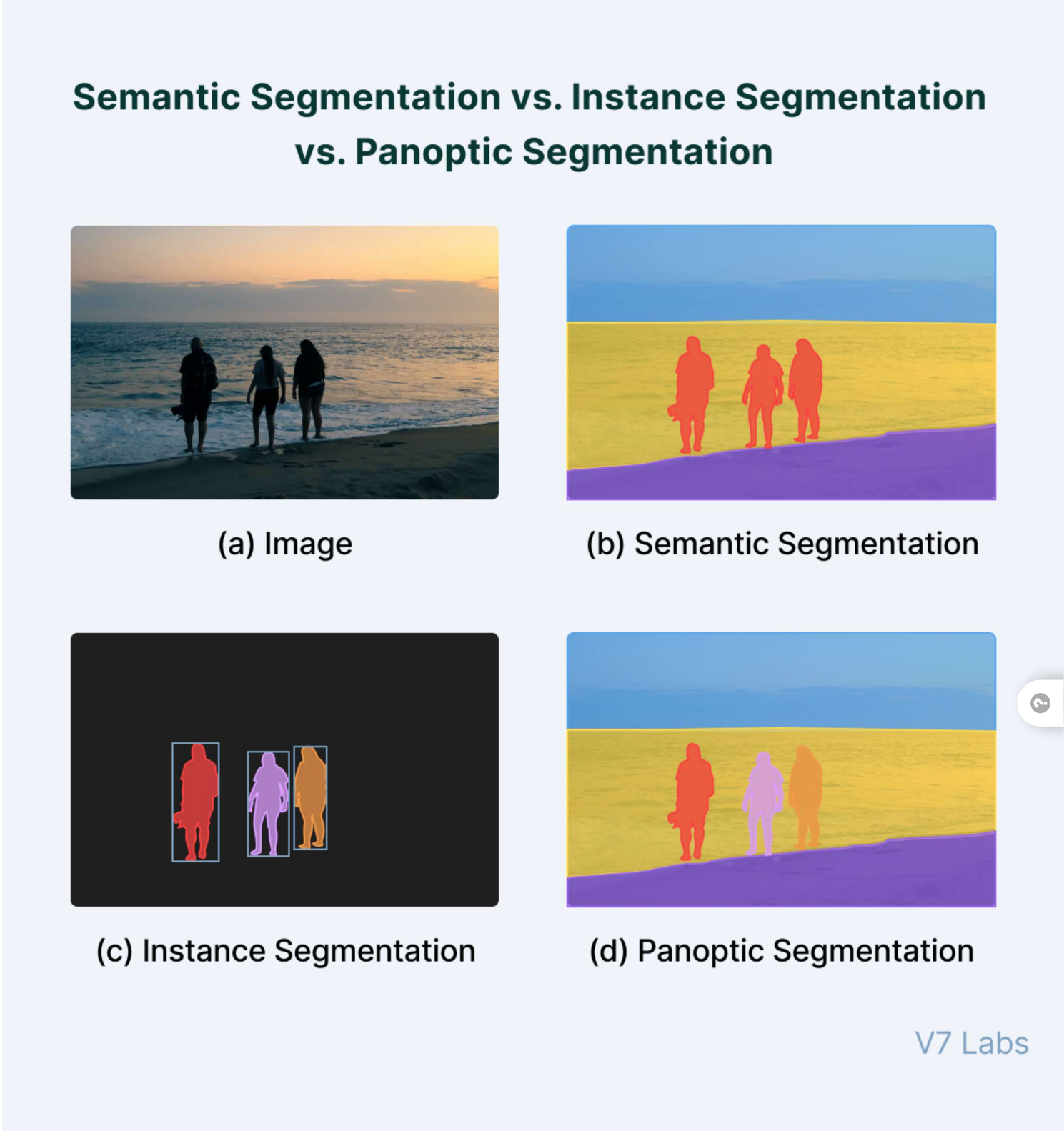
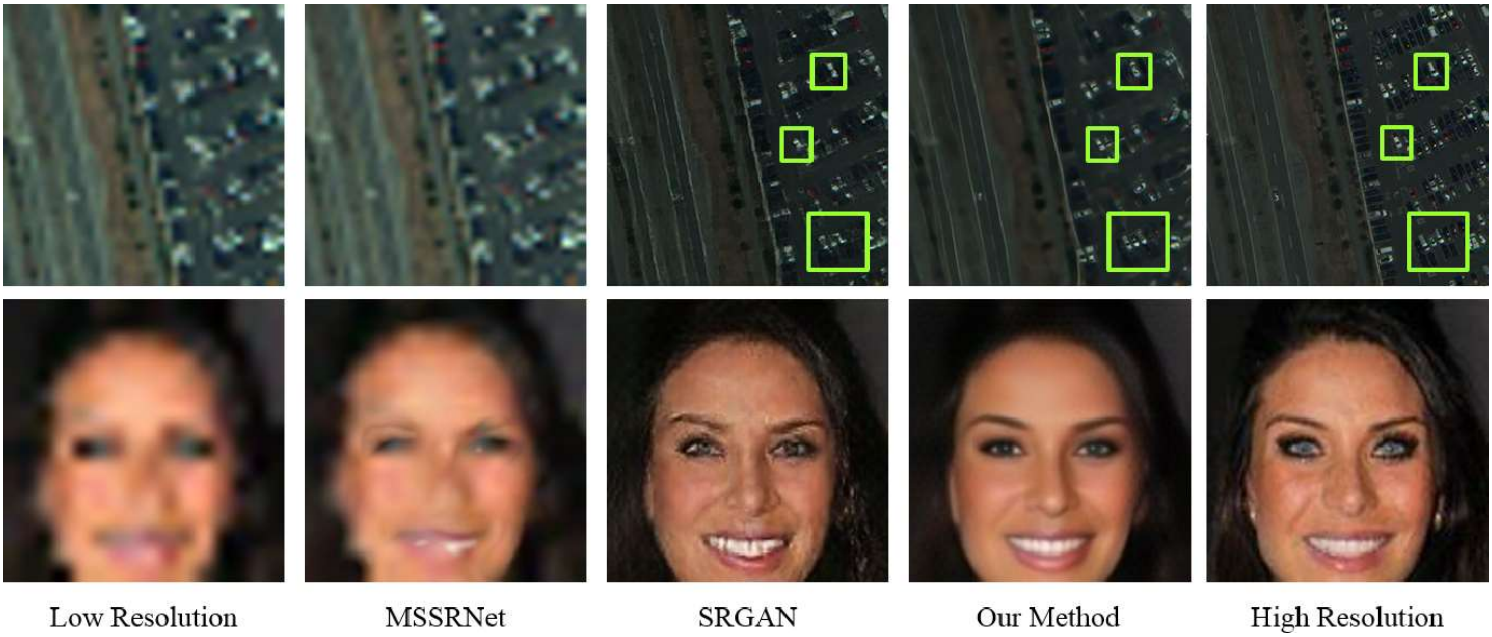


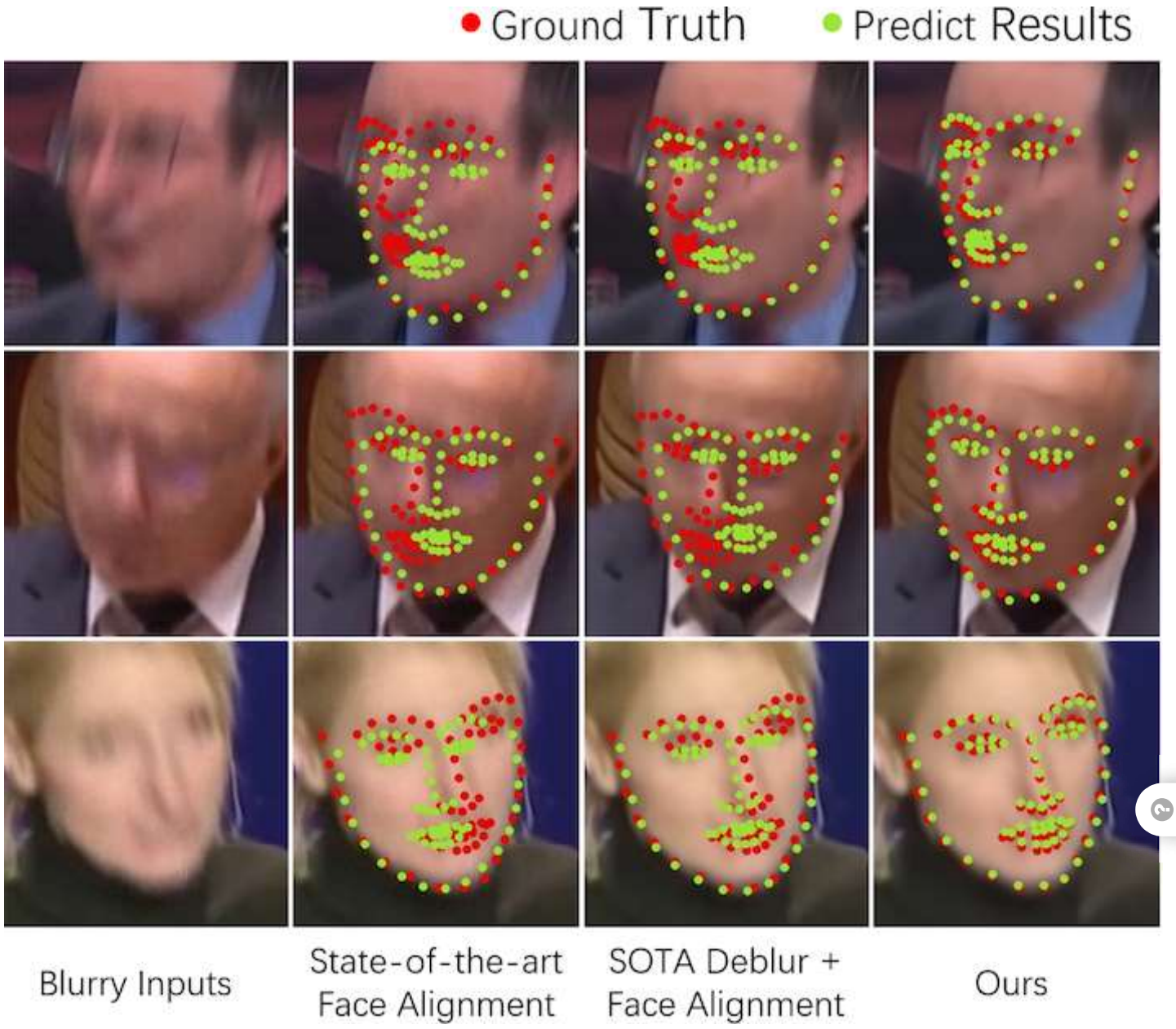
image super-resolution,



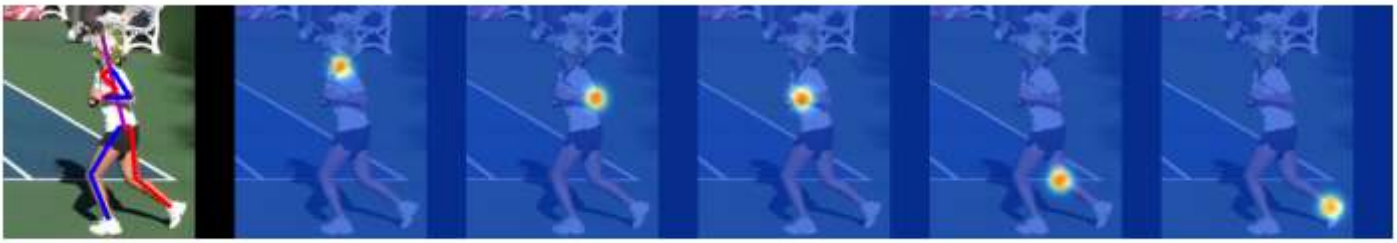
denoising,

Generative Art with Reference Images

keypoint detection or Facial Landmarks



Pose Estimation



and a lot more!

What is the output above?

"Dense"

The Problem: "convolution is a rather local operation" Improving the resolution of segmentation results Dilated convolutions or atrous convolutions present a potential solution to this problem: they are cap of capturing global context without reducing the resolution of the segmentation map.

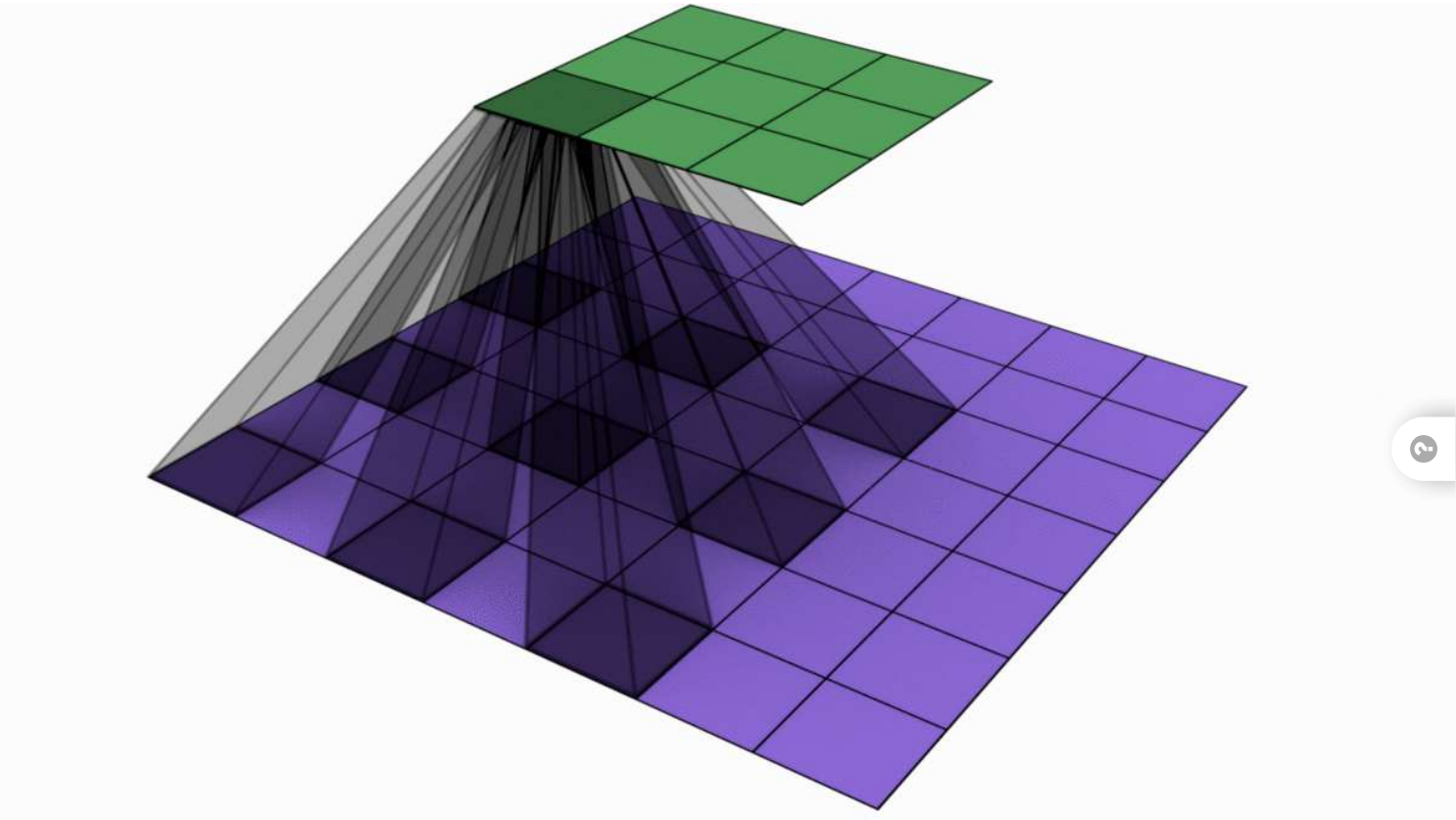
Normal convolutional neural networks (CNNs) are based on the fact that each convolutional layer collects information from a larger neighborhood around each pixel/voxel. This means that in the upper layers of the network, each pixel of a feature map could potentially hold information about a large region of the image.

However, experiments show that during learning this is usually not the case. Rather, the information in each pixel stays localized [source] and the network “does not live up to its full potential”.

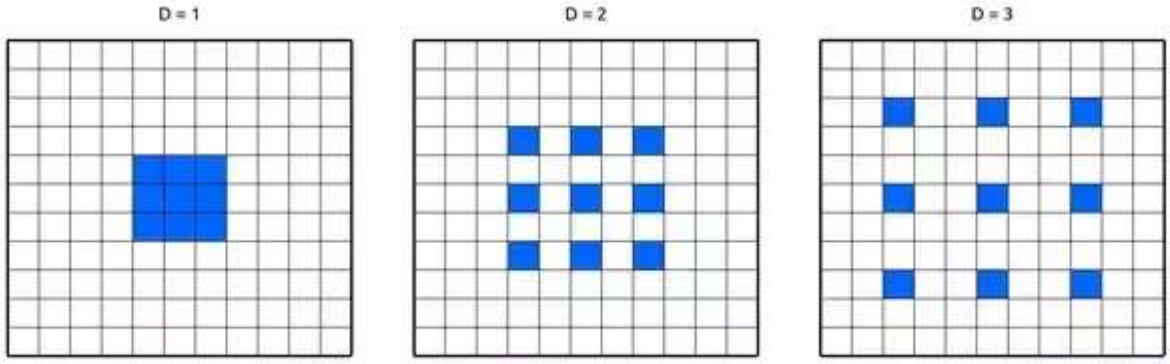
Dilated convolutions change the rules of the game by using kernels of the same size as normal convolutional layers but spread out over a larger area on the image.

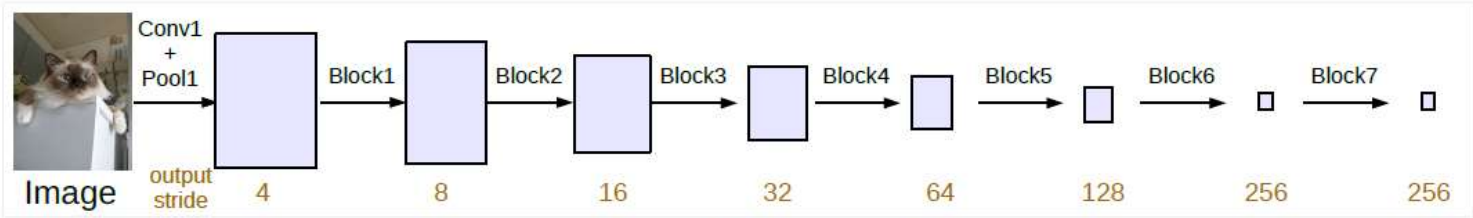
In many such applications, one wants to integrate information from different spatial scales and balance two properties:

- local, pixel-level accuracy, such as precise detection of edges, and
- integrating the knowledge of the wider, global context

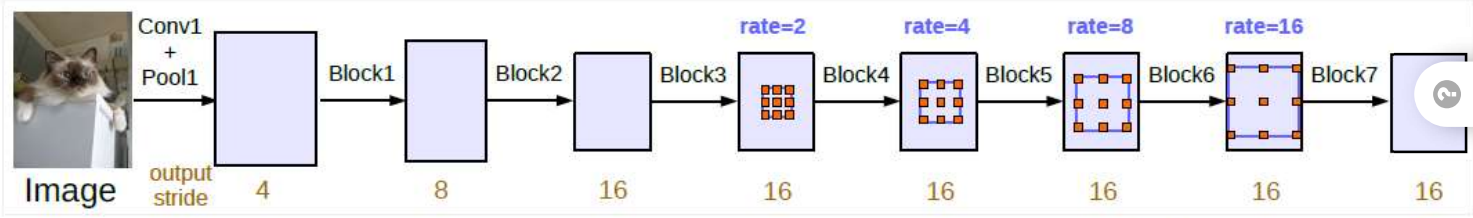


Dilation





(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

How do we increase channel size after Convolution

or

Transpose Convolution

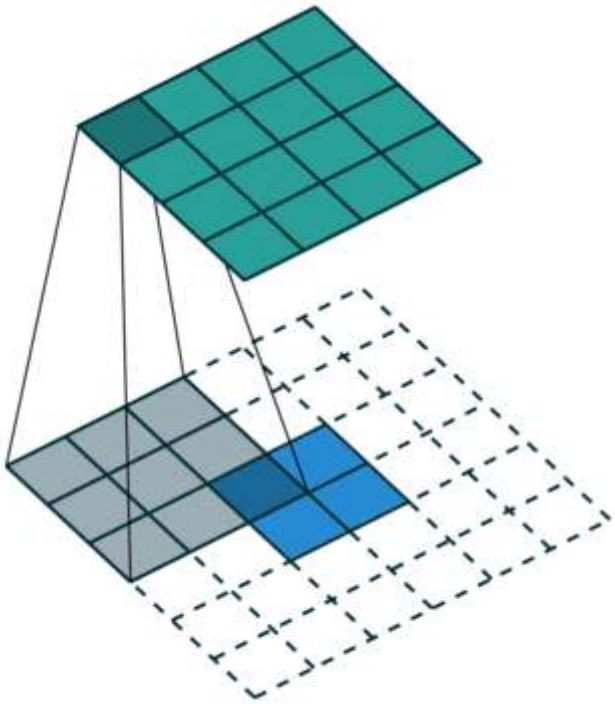
or

Deconvolution or Fractionally Strided Convolution

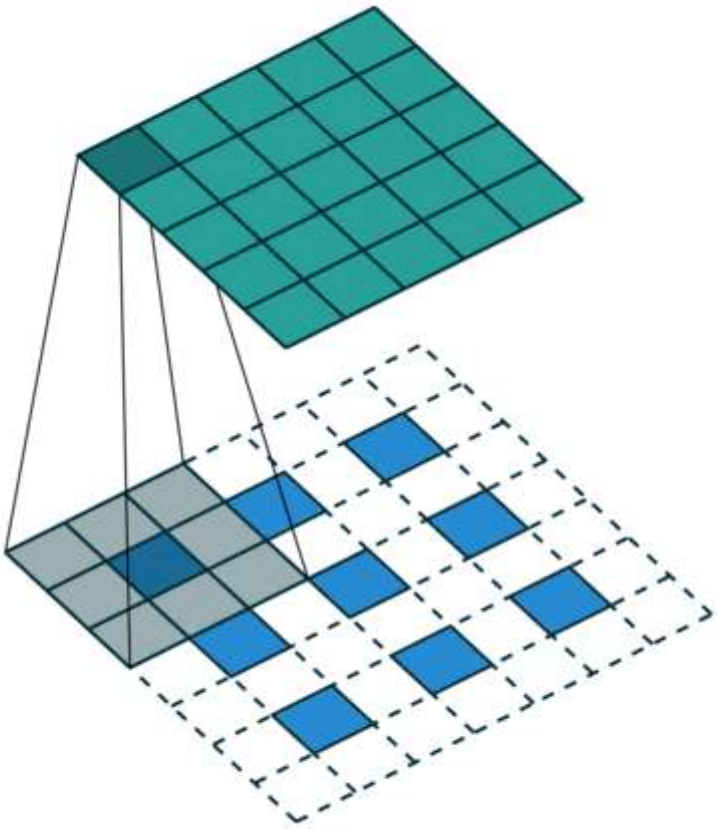
We have a kernel of size 3x3.

If we convolve on 5x5 we get 3x3. What do we do to get 7x7? Or get a larger channel size than we started with?

A simple approach, but inefficient.



Deconvolution or Transpose Convolution is a better approach, but...

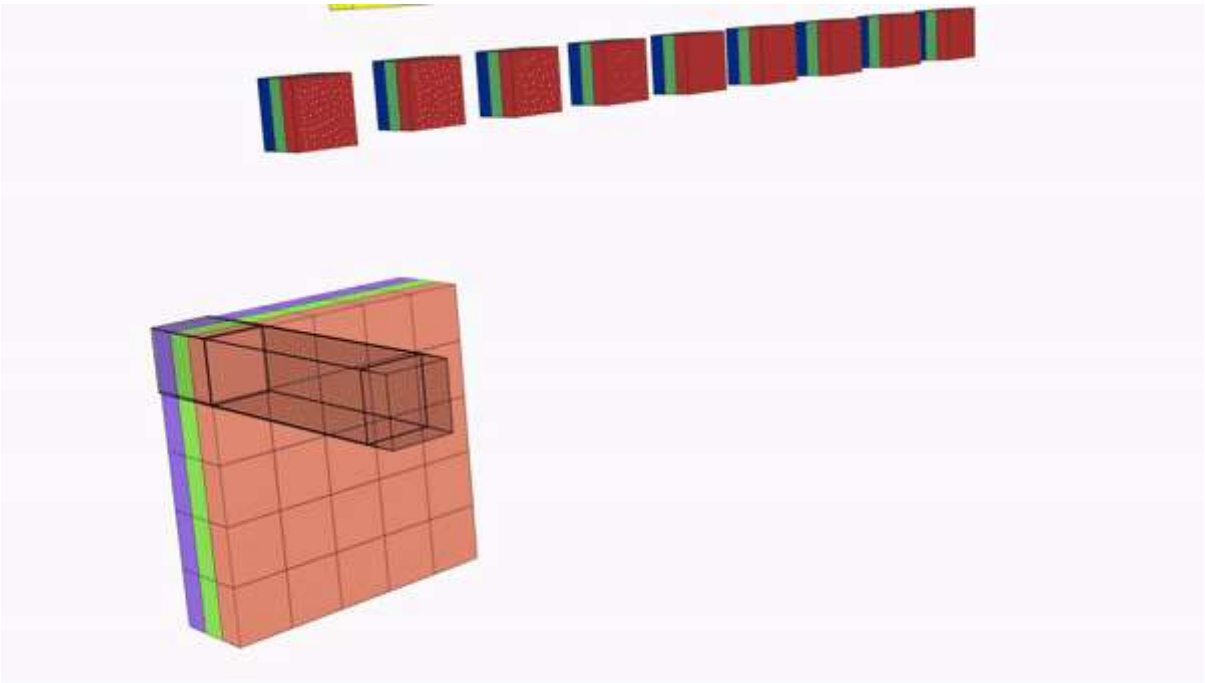


it introduces the checkerboard issue:

Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Pixel Shuffle

We'll learn about PS algorithm to understand that we are free to manipulate the pixels the way we want! Nothing more, nothing less.

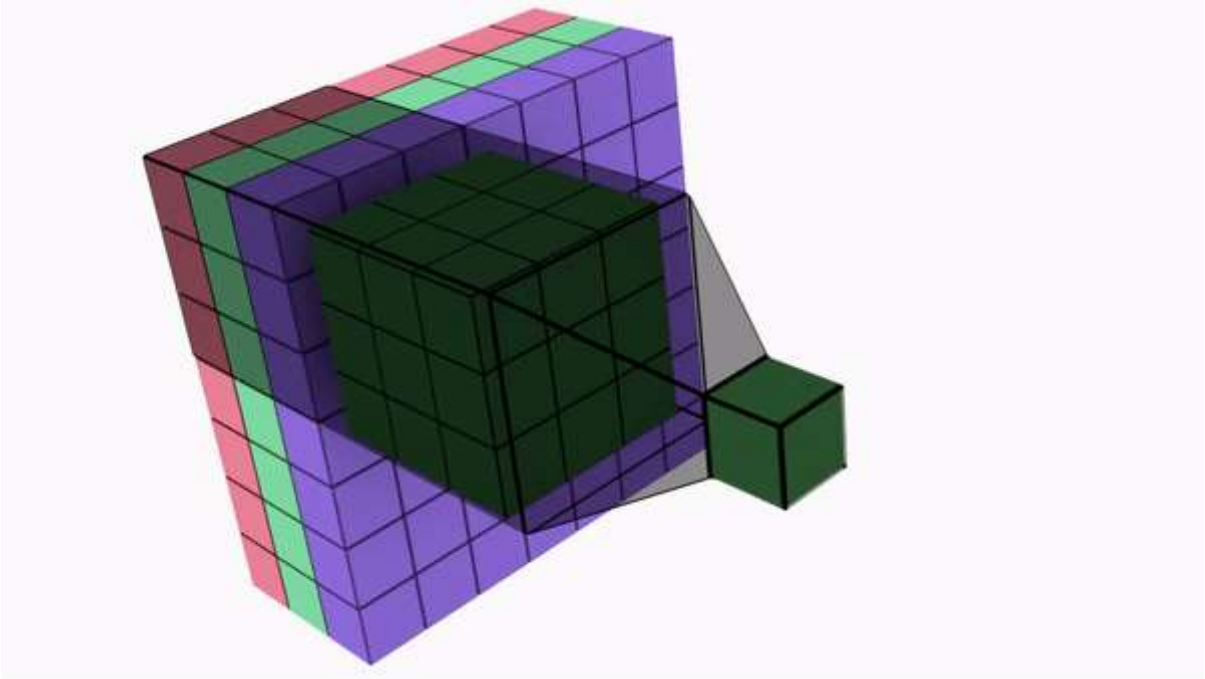


Jeremy Howard's Explanation (https://www.youtube.com/embed/nG3tT31nPmQ?start=2128)

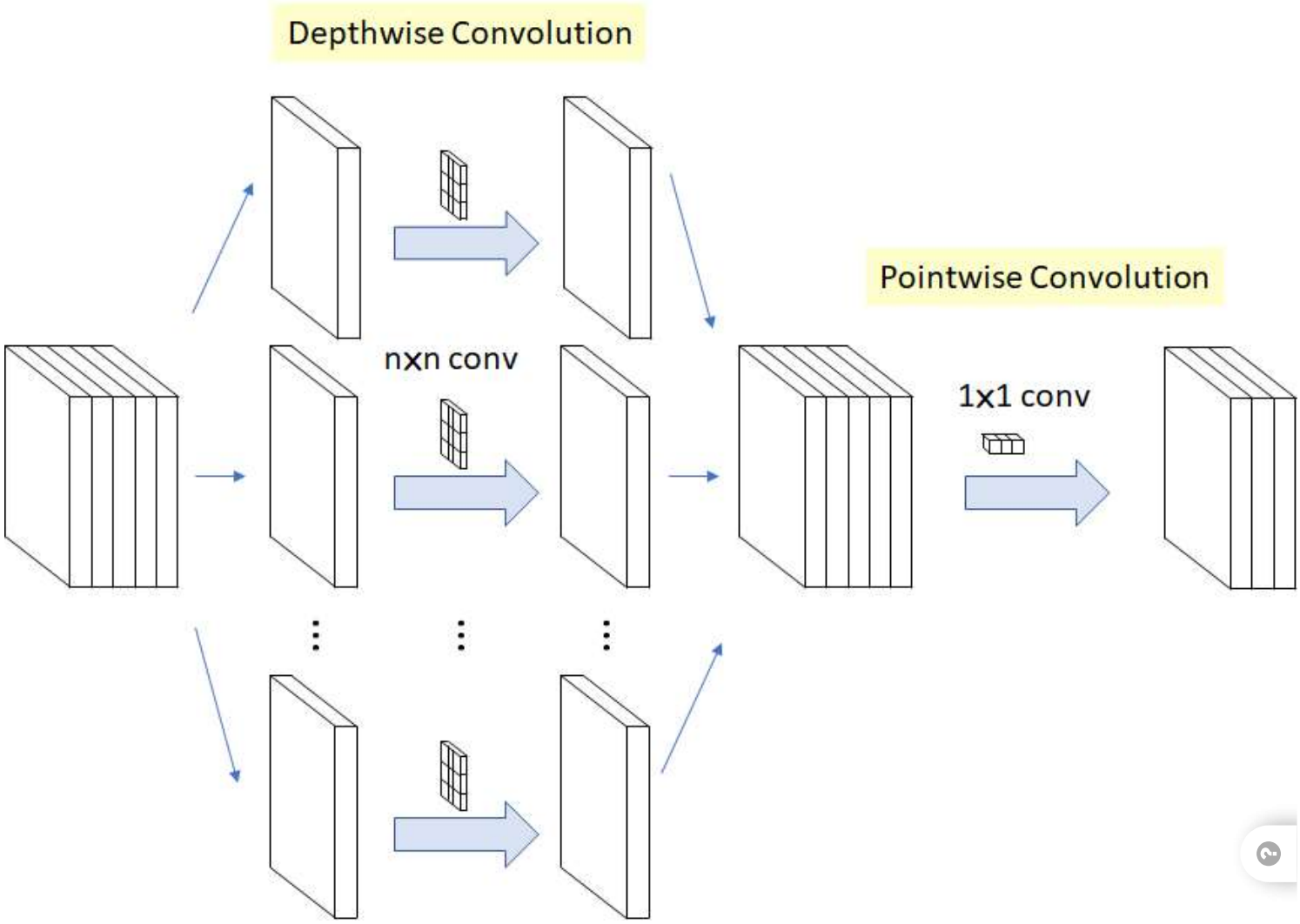
Pixel Shuffle Paper (https://arxiv.org/pdf/1609.05158.pdf)

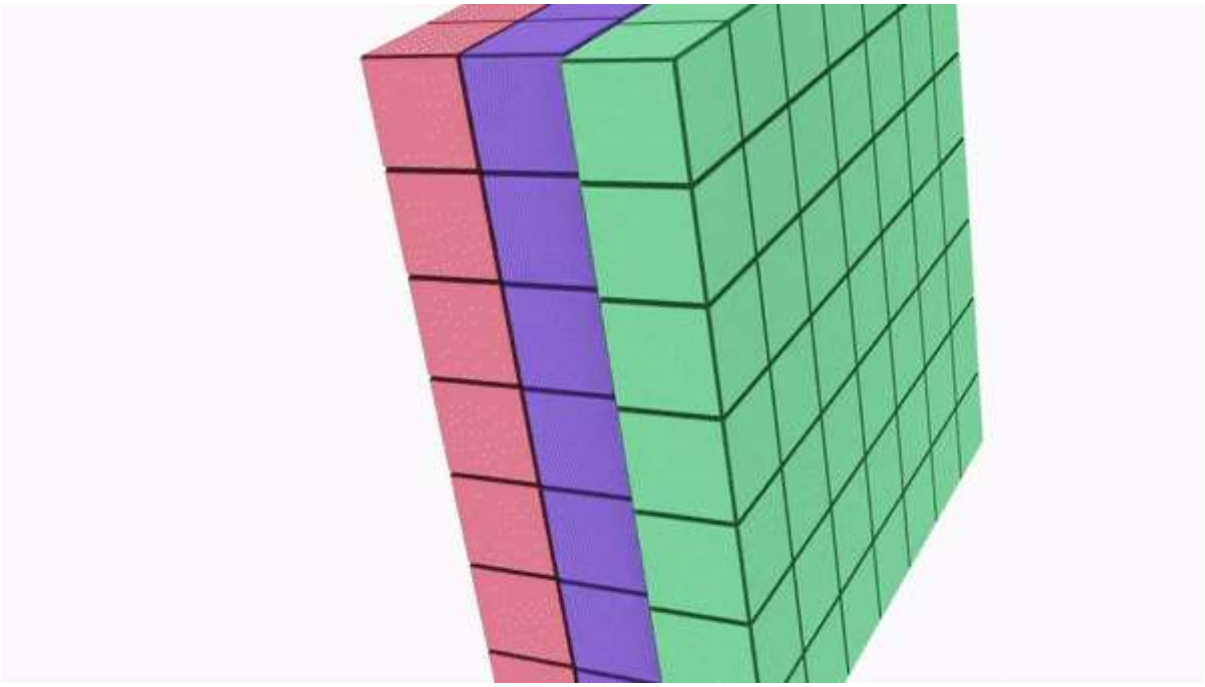
Depthwise Separable Convolution

What is wrong with this convolution?



Depthwize Separable Convolution

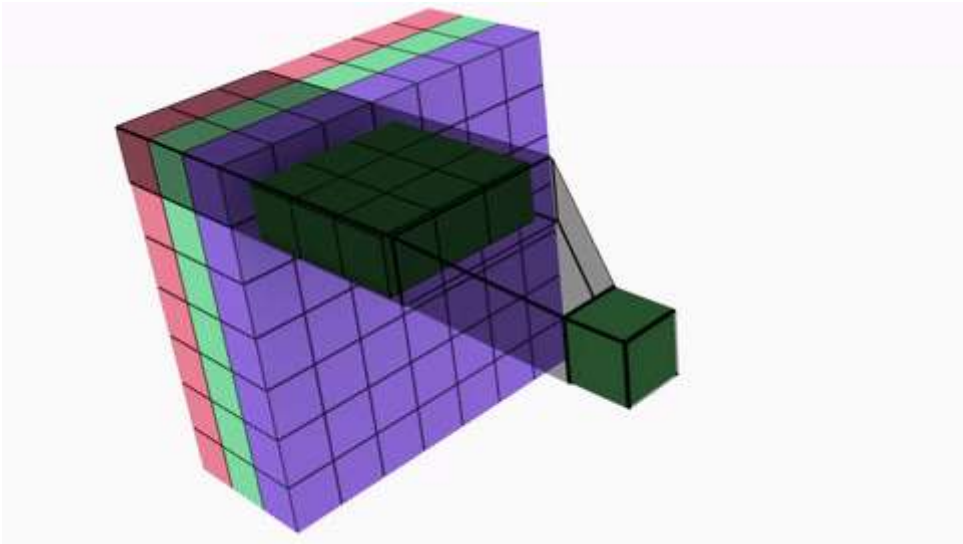

Depthwise Convolution
nxn conv
Pointwise Convolution
1x1 conv



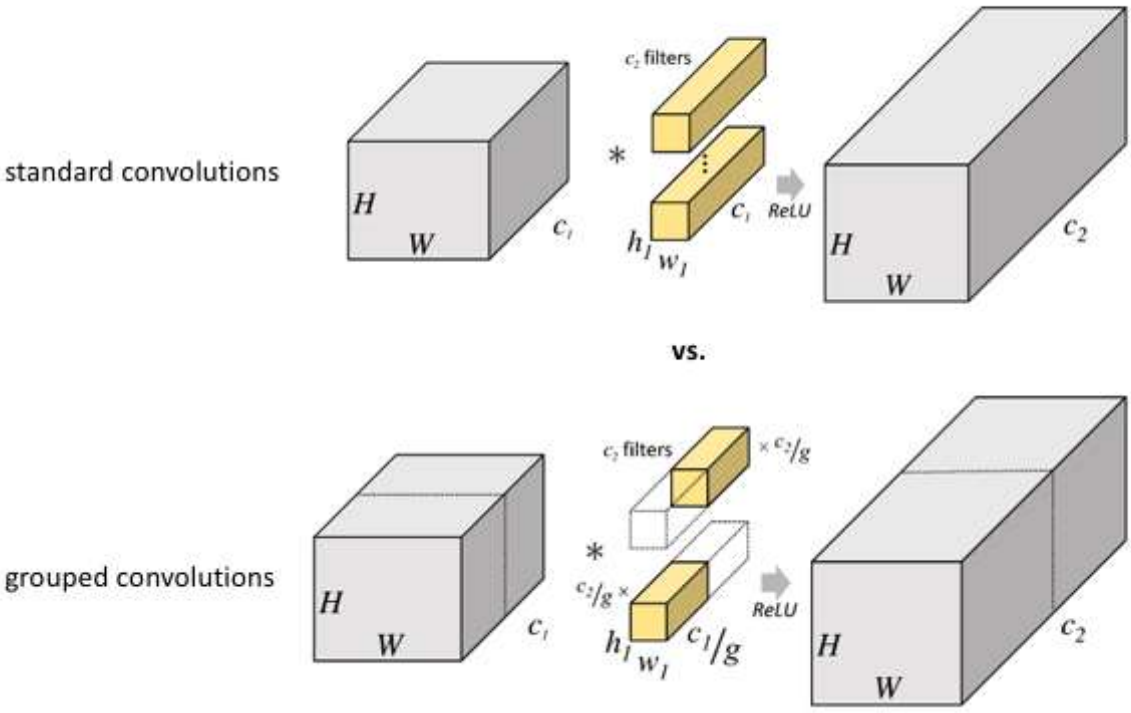
For a depthwise separable convolution on the same example, we traverse the 16 channels with 1 3x3 kernel each, giving us 16 feature maps.
Now, before merging anything, we traverse these 16 feature maps with 32 1x1 convolutions each and only then start to them add together.
This results in 656 (16x3x3 + 16x32x1x1) parameters as opposed to the 4608 (16x32x3x3) parameters from above. [REF (https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/)]

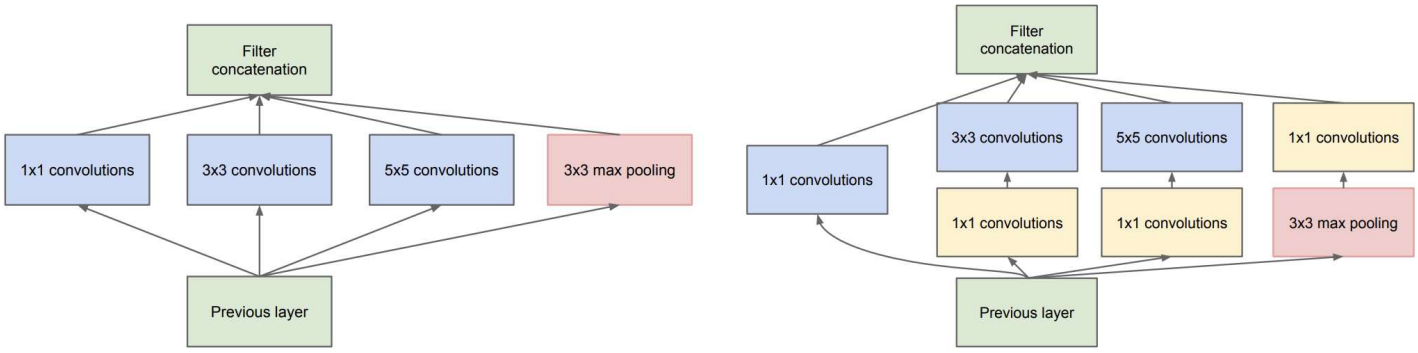
Spatially Separable Convolutions

Was used immensely in different variants of Xception-Inception Networks as well as in MobileNets.\



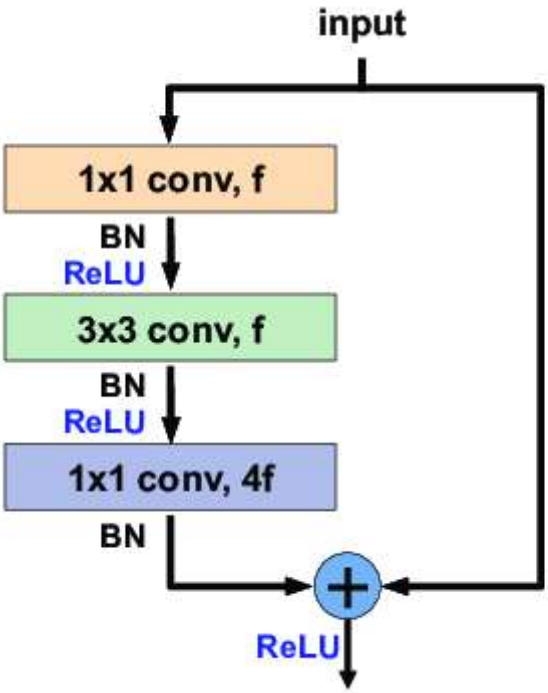
Grouped Convolutions

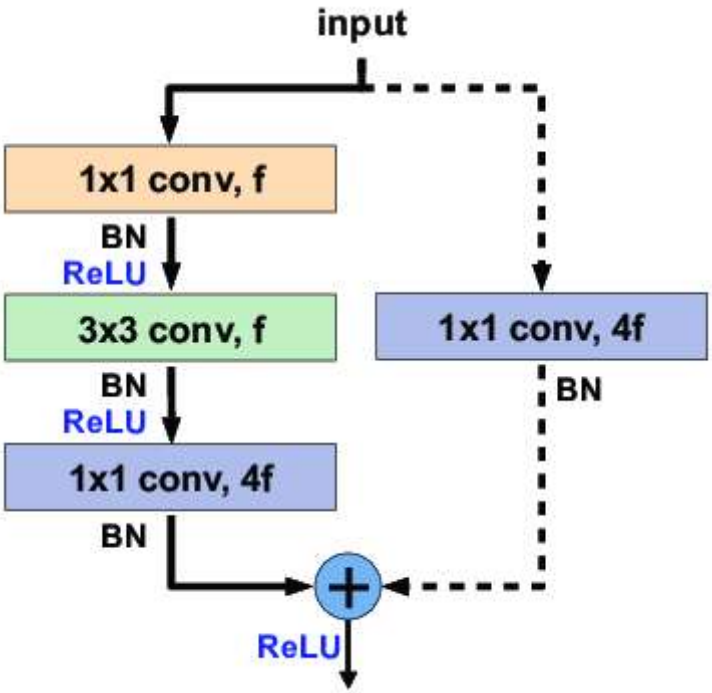


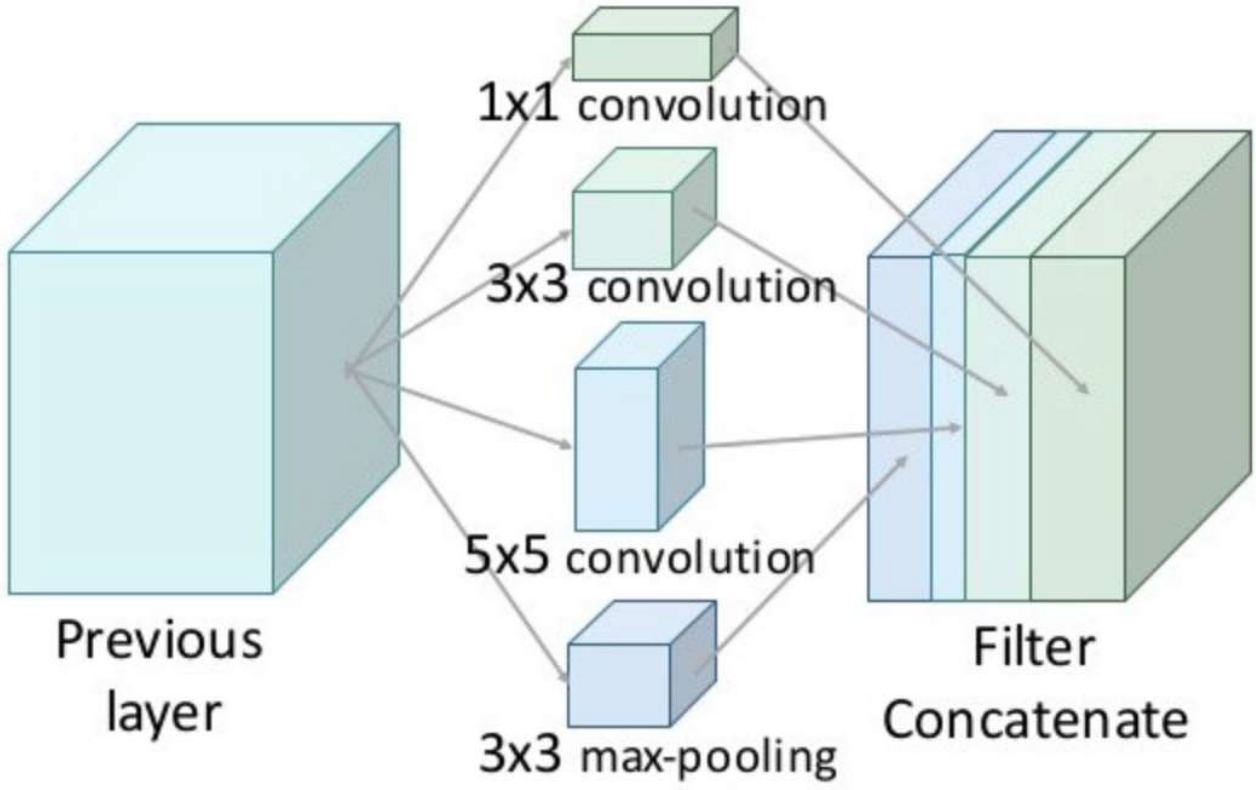


(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module


input
1x1 conv, f
BN
ReLU
3x3 conv, f
BN
ReLU
1x1 conv, 4f
BN
ReLU

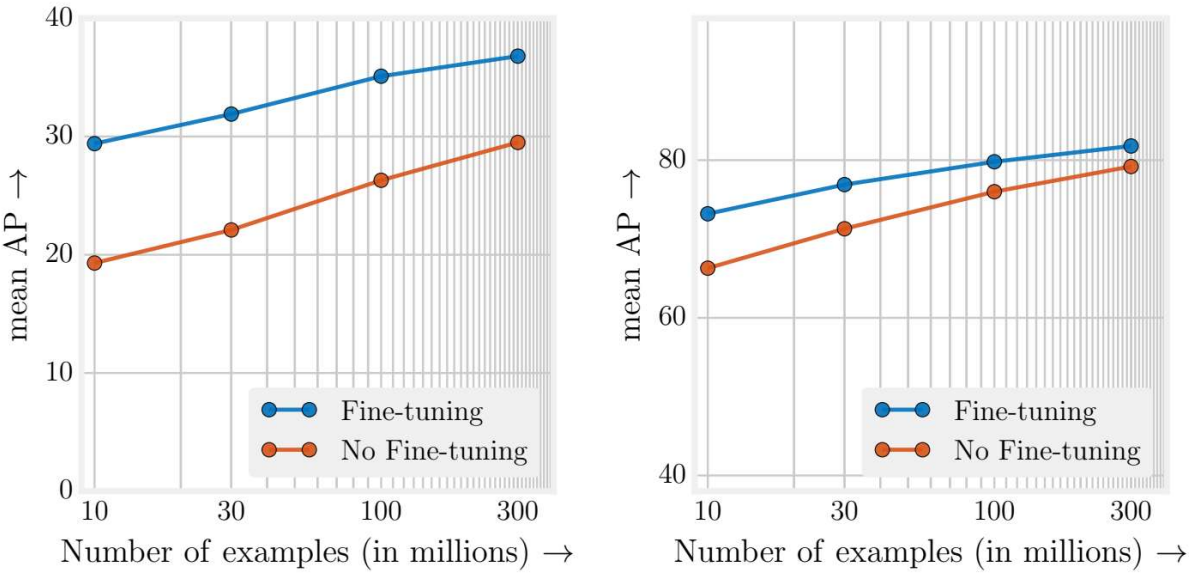

input
1x1 conv, f
BN
ReLU
3x3 conv, f
BN
ReLU
1x1 conv, 4f
BN
1x1 conv, 4f
BN
ReLU


1x1 convolution
3x3 convolution
5x5 convolution
3x3 max-pooling
Previous layer
Filter Concatenate

Data Augmentation

Any why we should fall in love with it!

Relationship between Data and Validation Accuracy



Data Augmentation

Data augmentation (DA) is an effective alternative to obtaining valid data, and can generate new labeled data based on existing data using "label-preserving transformations".

Designing appropriate DA policies requires a lot of expert experience and is time-consuming, and the evaluation of searching the optimal policies is costly. Moreover, the policies generated using DA policies are usually not reusable.

Generally, the more data, the better deep neural networks can do. Insufficient data can lead to model over-fitting, which will reduce the generalization performance of the model on the test set.

Source (http://openaccess.thecvf.com/content_ICCV_2017/papers/Sun_Revisiting_Unreasonable_Effectiveness_ICC : we find that the performance on vision tasks increases logarithmically based on the volume of training data size

Techniques such as:

- DropOut
- Batch Normalization
- L1/L2 regularization Layer normalization

have been proposed to help combat over-fitting, but they will fall short if data is limited.

What options do we have?

PMDA - Poor Man's Data Augmentation Strategies

MMDA* - Middle-Class Man's Data Augmentation Strategies

RMDA* - Rich Man's Data Augmentation Strategies

PMDA

Scale; Translation; Rotation; Blurring; Image Mirroring; Color Shifting / Whitening.

Simple Stuff which most probably is in-built in Pytorch.

MMDA

We have 2 beautiful Libraries for MMDAs:

IMGAUG - not going to use (https://github.com/aleju/imgaug)

(https://github.com/aleju/imgaug) (https://github.com/albumentations-team/albumentations)

ALBUMENTATIONS - going to use (https://github.com/albumentations-team/albumentations)

We are going to Focus on ALBUMENTATIONS because of its easy integration with PyTorch.
(https://github.com/albumentations-team/albumentations)

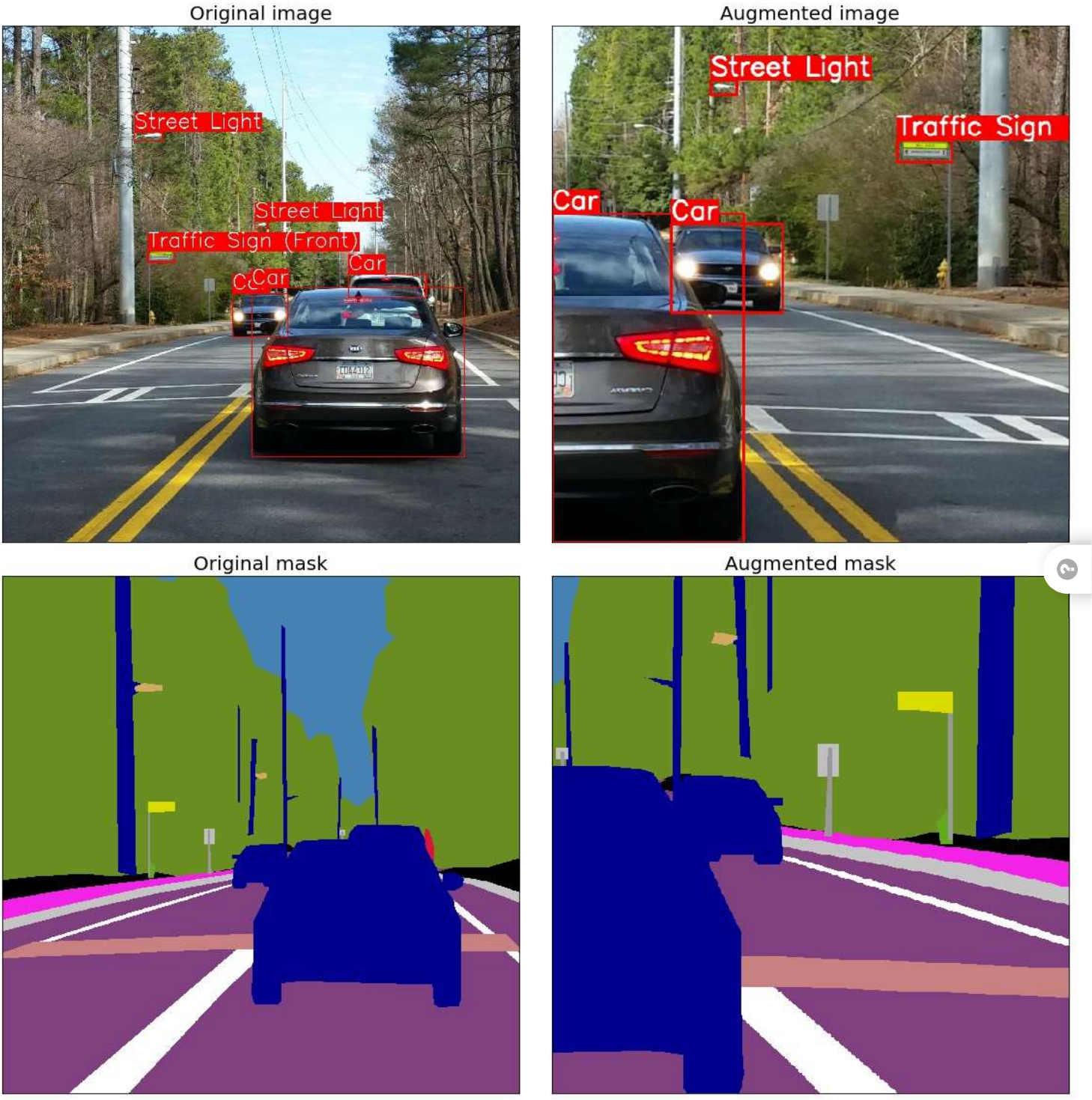
(https://github.com/albumentations-team/albumentations)

	albumentations 1.1.0	imgaug 0.4.0	torchvision (Pillow-SIMD backend) 0.10.1	keras 2.6.0	augmentor 0.2.8	solt 0.1.9
HorizontalFlip	**10220**	2702	2517	876	2528	6798
VerticalFlip	**4438**	2141	2151	4381	2155	3659
Rotate	**389**	283	165	28	60	367
ShiftScaleRotate	**669**	425	146	29	-	-
Brightness	**2765**	1124	411	229	408	2325
Contrast	**2767**	1137	349	-	346	2311
BrightnessContrast	**2746**	629	190	-	189	1196
ShiftRGB	**2758**	1093	-	360	-	-
ShiftHSV	**598**	259	59	-	-	144
Gamma	**2849**	-	388	-	-	933
Grayscale	**5219**	393	723	-	1082	1309
RandomCrop64	**163550**	2562	50159	-	42842	22260
PadToSize512	**3609**	-	602	-	-	3097
Resize512	1049	611	**1066**	-	1041	1017
RandomSizedCrop_64_512	**3224**	858	1660	-	1598	2675
Posterize	**2789**	-	-	-	-	-
Solarize	**2761**	-	-	-	-	-
Equalize	647	385	-	-	**765**	-
Multiply	**2659**	1129	-	-	-	-
MultiplyElementwise	111	**200**	-	-	-	-
ColorJitter	**351**	78	57	-	-	-

(https://github.com/albumentations-team/albumentations)

(https://github.com/albumentations-team/albumentations)

It has everything you'll ever need (conditions apply)



Elastic Distortions

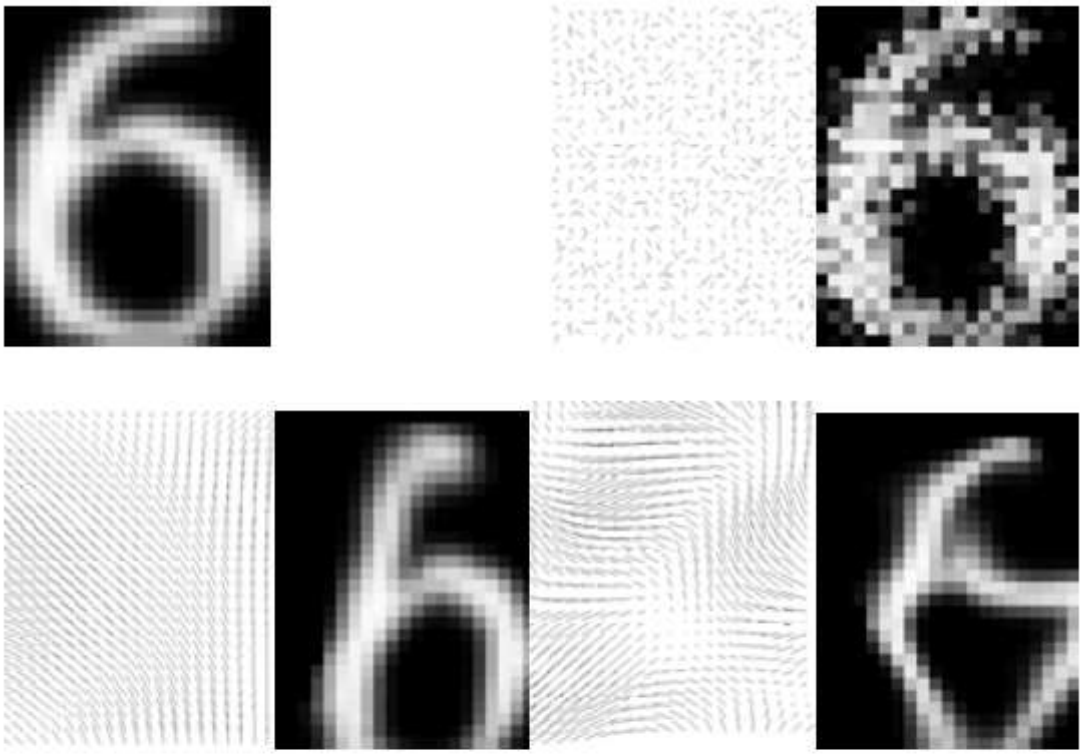
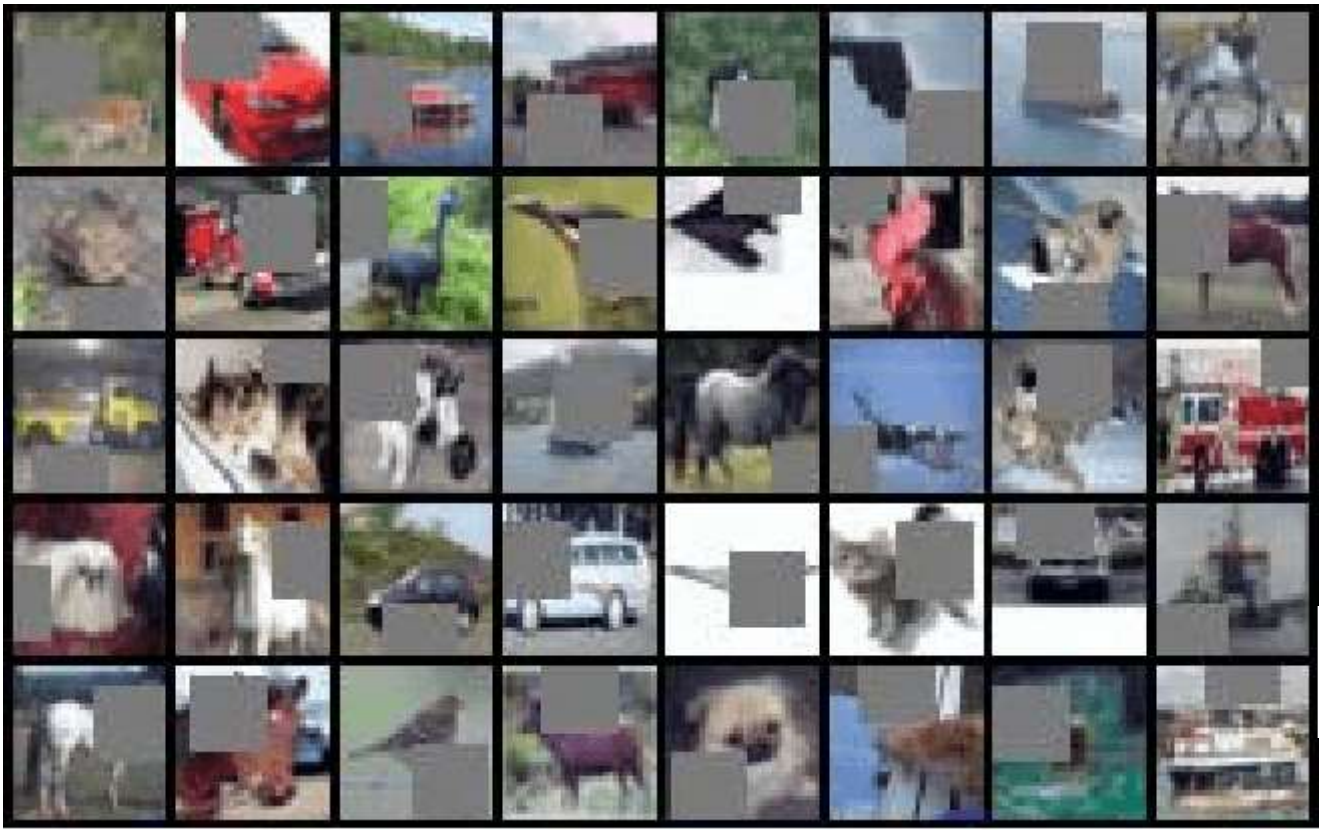


Figure 2. Top left: Original image. Right and bottom: Pairs of displacement fields with various smoothing, and resulting images when displacement fields are applied to the original image.

Elastic Distortions (2003) (http://cognitivemedium.com/assets/rmnist/Simard.pdf)

A very nice article on how to use this on Bengali (or others) is **here (https://www.kaggle.com/corochann/bengali-albumentations-data-augmentation-tutorial)**

CutOut



CutOut - 29 Nov 2017 (https://arxiv.org/pdf/1708.04552.pdf)

What color is that cut-out grey?

`class CoarseDropout (max_holes=8, max_height=8, max_width=8, min_holes=None, min_height=None, min_width=None, fill_value=0, mask_fill_value=None, always_apply=False, p=0.5)` [view source on GitHub]

CoarseDropout of the rectangular regions in the image.

Parameters:

Name	Type	Description
`max_holes`	`int`	Maximum number of regions to zero out.
`max_height`	`int, float`	Maximum height of the hole.
`max_width`	`int, float`	Maximum width of the hole.
`min_holes`	`int`	Minimum number of regions to zero out. If `None`, `min_holes` is be set to `max_holes`. Default: `None`.
`min_height`	`int, float`	Minimum height of the hole. Default: None. If `None`, `min_height` is set to `max_height`. Default: `None`. If float, it is calculated as a fraction of the image height.
`min_width`	`int, float`	Minimum width of the hole. If `None`, `min_height` is set to `max_width`. Default: `None`. If float, it is calculated as a fraction of the image width.
`fill_value`	`int, float, list of int, list of float`	value for dropped pixels.
`mask_fill_value`	`int, float, list of int, list of float`	fill value for dropped pixels in mask. If `None` - mask is not affected. Default: `None`.

(https://albumentations.ai/docs/api_reference/augmentations/dropout/coarse_dropout/)

What is the maximum cut-out size you can use?

Why CutOut?

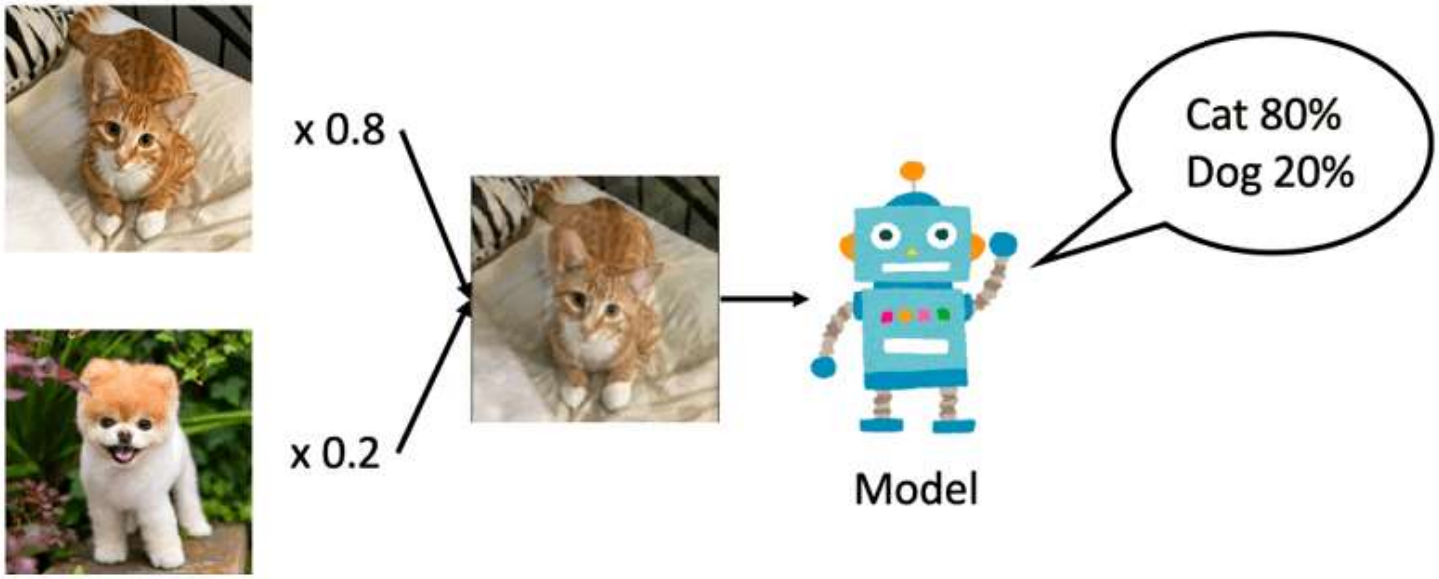
Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

Table 1: Test error rates (%) on CIFAR (C10, C100) and SVHN datasets. "+" indicates standard data augmentation (mirror + crop). Results averaged over five runs, with the exception of shake-shake regularization which only had three runs each. Baseline shake-shake regularization results taken from [4].

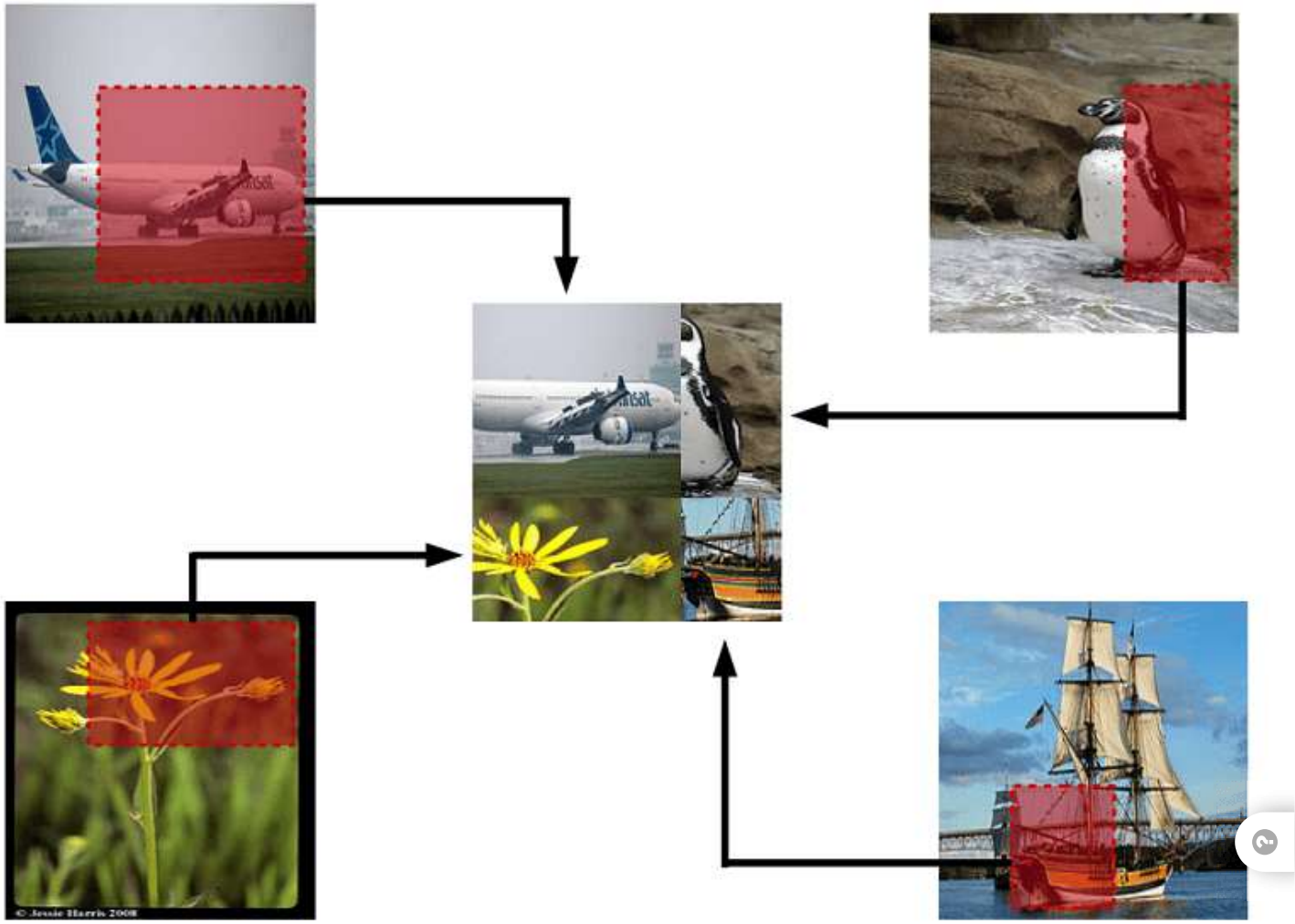
MixUp

MixUp 27 April 2018 (https://arxiv.org/pdf/1710.09412.pdf)

Mixup alpha-blends two images to construct a new training image. Mixup can train deep CNNs on convex combinations of pairs of training samples and their labels and enables deep CNNs to favor a simple linear behavior in between training samples. This behavior makes the prediction confidence transit linearly from one class to another class, thus providing smoother estimation and margin maximization. Alpha-blending not only increases the variety of training images but also works like adversarial perturbation. Thereby, mixup makes deep CNNs robust to adversarial examples and stabilizes the training of generative adversarial networks. In addition, it behaves similarly to class label smoothing by mixing class labels with the ratio $\lambda : 1 - \lambda$.



RICAP



RICAP 22 Nov 2018 (https://arxiv.org/pdf/1811.09030v1.pdf)

RICAP crops four training images and patches them to construct a new training image; it selects images and determines the cropping sizes randomly, where the size of the final image is identical to that of the original image. RICAP also mixes class labels of the four images with ratios proportional to the areas of the four images like **label smoothing (https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html) [REF (https://daydreamatnight.github.io/2022/03/04/Intro-and-Pytorch-Implementation-of-Label-Smoothing-Regularization-LSR/)**] in mixup. Compared to mixup, RICAP has three clear distinctions: it mixes images spatially, it uses partial images by cropping, and it does not create features that are absent in the original dataset except for boundary patching.

TABLE I
TEST ERROR RATES USING WIDERESNET ON THE CIFAR DATASET.

Method	CIFAR-10	CIFAR-100
Baseline	3.89	18.85
+ dropout ($p = 0.2$)	4.65 $\pm 0.08^{\dagger}$	21.27 $\pm 0.19^{\dagger}$
+ cutout (16×16)	3.08 ± 0.16	18.41 ± 0.27
+ random erasing	3.08 ± 0.05	17.73 ± 0.15
+ mixup ($\alpha = 1.0$)	3.02 $\pm 0.04^{\dagger}$	17.62 $\pm 0.25^{\dagger}$
+ RICAP ($\beta = 0.3$)	**2.85** ± 0.06	**17.22** ± 0.20

† indicates the results of our experiments.

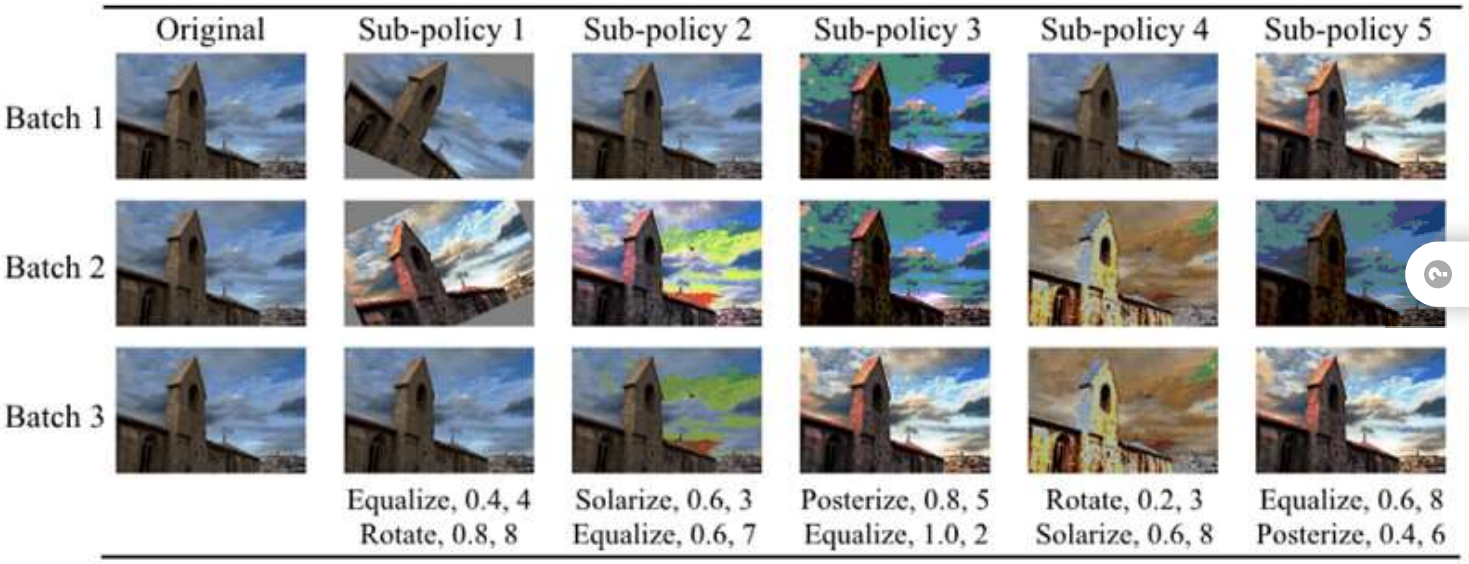
Paper	Year	Highlight
SamplePairing [27]	2018	Combine two images with a single label.
Mixup [28]	2018	Linearly fuse images and their labels. Figure 5.
BC Learning [29]	2018	Combine two images and their labels. Treat the image as a waveform, and declare that image mixing makes sense for machines.
CutMix [30]	2019	Spatially fuse two images and linearly fuse the labels. Figure 5.
Mosaic [22]	2020	Spatially mix four images and their annotations, thereby enriching the context for each class.
AugMix [31]	2020	One image undergoes several basic augmentations, and the results are fused with the original image.
PuzzleMix [32]	2020	Optimize a mask for fusing two images to utilize the salient information and underlying statistics.
Co-Mixup [33]	2021	Maximize the salient signal of input images and diversity among the augmented images.
SuperMix [34]	2021	Optimize a mask for fusing two images to exploit the salient region with the Newton iterative method, 65x faster than gradient descent.
GridMix [35]	2021	Split two images into patches, spatially fuse the patches, and linearly merge the annotation.
Cut, Paste and Learn [36]	2017	Cut object instances and paste them onto random backgrounds. Figure 6.
Scale and Blend [37]	2017	Cut and scale object instances, and blend them in meaningful locations.
Context DA [38]	2018	Combine object instances using context guidance to obtain meaningful images.
Simple Copy-Paste [39]	2021	Randomly paste object instances to images with large-scale jittering.
Continuous Copy-Paste [40]	2021	Deploy Cut, Paste and Learn to videos.

Reinforcement Learning or AutoAugment 24th May 2018 (https://arxiv.org/abs/1805.09501)

Pros:

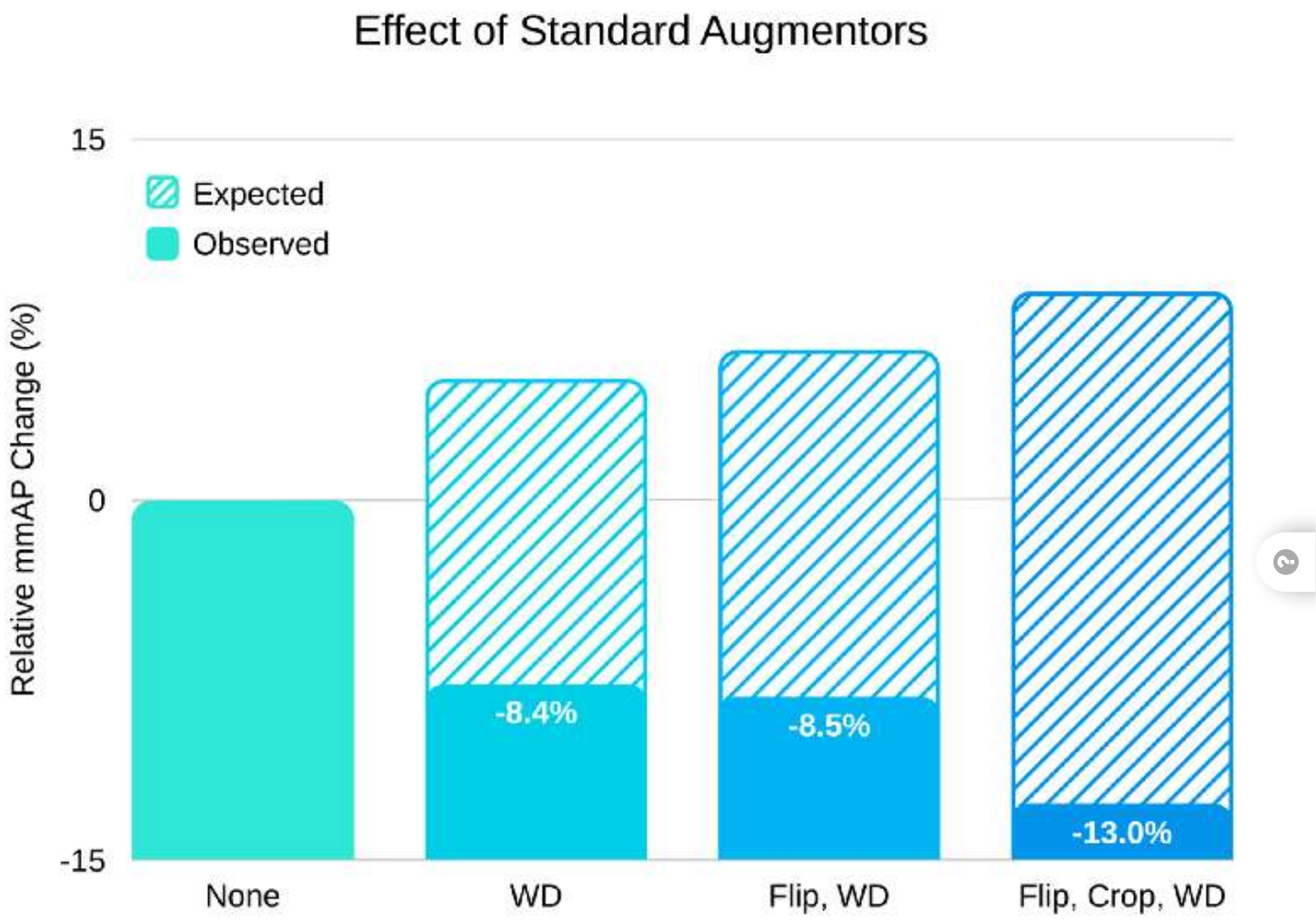
- Can be applied directly on the dataset
- Learned policies can be transferred to a new dataset
- Achieves State-of-art for ImageNet, CIFAR10, and CIFAR100
- NAS took CIFAR10 error to 2% (cannot be beaten by humans), AutoAugment with NAS took this number to 1.5%

Cons: Takes 5000 P100 GPU hours for CIFAR and 15000 GPU hours for ImageNet

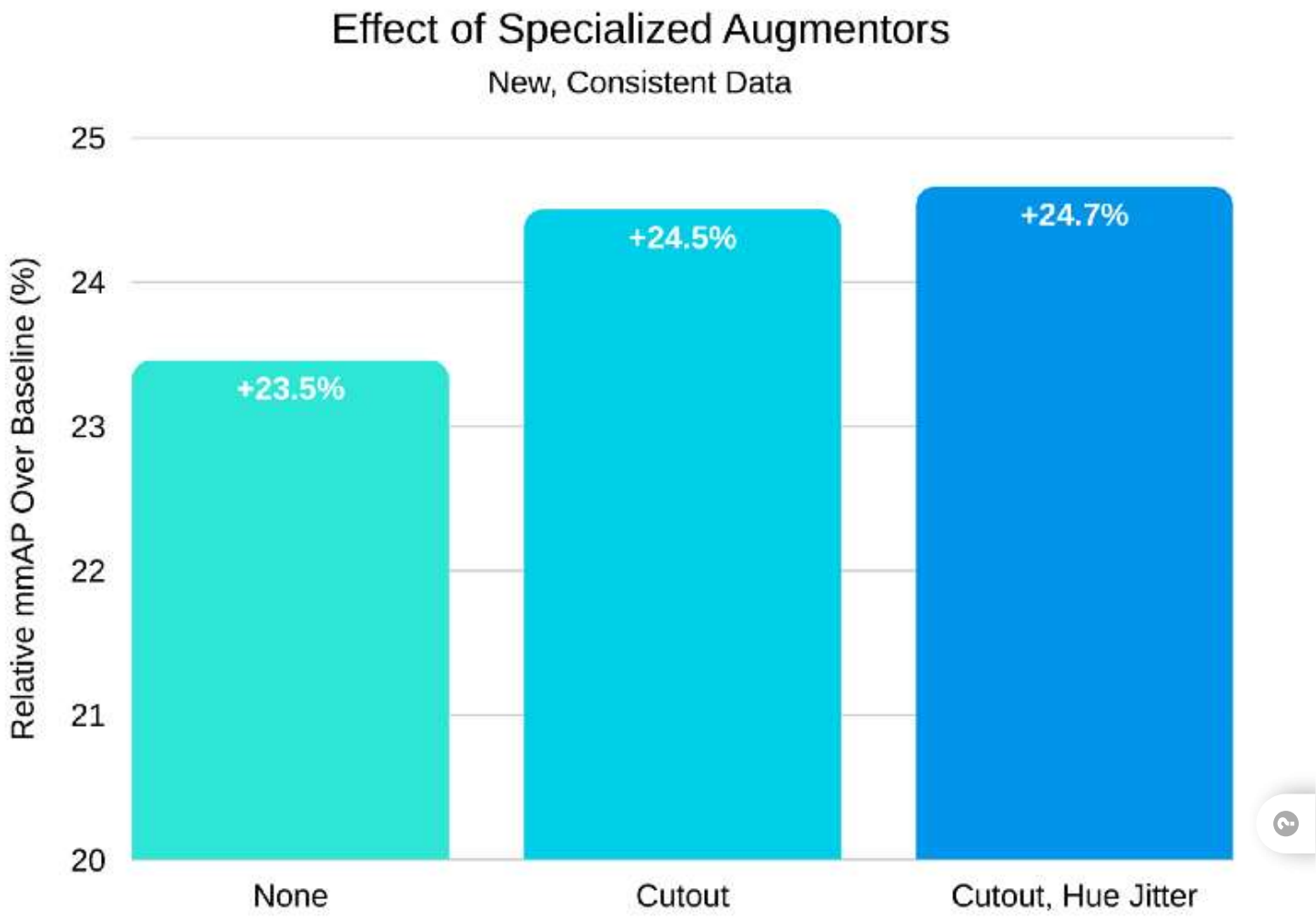


Strategy

But



Also (https://meet.google.com/linkredirect?authuser=2&dest=https%3A%2F%2Ftowardsdatascience.com%2Fwhen-conventional-wisdom-fails-revisiting-data-augmentation-for-self-driving-cars-4831998c5509)



A Comprehensive Survey of Image Augmentation

Check out here - 23 Nov 2022 (https://arxiv.org/pdf/2205.01491.pdf)

Assignment

Solution File for Assignment 8 (https://canvas.instructure.com/courses/8491182/files/248904593?wrap=1)
(https://canvas.instructure.com/courses/8491182/files/248904593/download?download_frd=1)

1. Write a new network that
    1. has the architecture to C1C2C3C40 (No MaxPooling, but 3 convolutions, where the last one has a stride of 2 instead) (If you can figure out how to use Dilated kernels here instead of MP or strided convolution, then 200pts extra!)
    2. total RF must be more than 44
    3. one of the layers must use Depthwise Separable Convolution
    4. one of the layers must use Dilated Convolution
    5. use GAP (compulsory):- add FC after GAP to target #of classes (optional)
    6. use albumentation library and apply:
        1. horizontal flip
        2. shiftScaleRotate
        3. coarseDropout (max_holes = 1, max_height=16px, max_width=16, min_holes = 1, min_height=16px, min_width=16px, fill_value=(mean of your dataset), mask_fill_value = None)
    7. achieve 85% accuracy, as many epochs as you want. Total Params to be less than 200k.
    8. make sure you're following code-modularity (else 0 for full assignment)
    9. upload to Github
    10. Attempt S9-Assignment Solution.
    11. Questions in the Assignment QnA are:
        1. copy and paste your model code from your model.py file (full code) [125]
        2. copy paste output of torch summary [125]
        3. copy-paste the code where you implemented albumentation transformation for all three transformations [125]
        4. copy paste your training log (you must be running validation/text after each Epoch [125]
        5. Share the link for your README.md file. [200]

Videos

Studio

GM