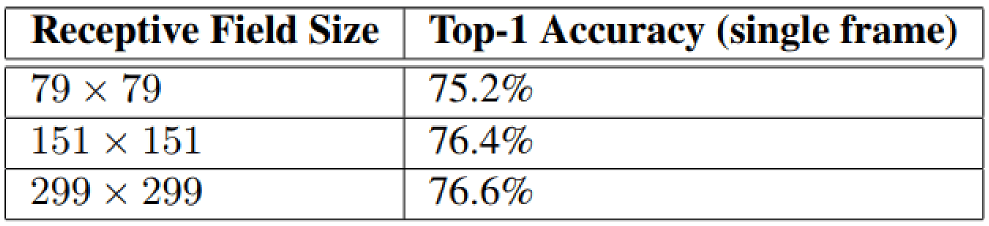
S9

Due No Due Date **Points** 0 **Available** after Mar 11 at 6:30am

DATA AUGMENTATION

One easy way of increasing accuracy is increasing the Receptive Field:



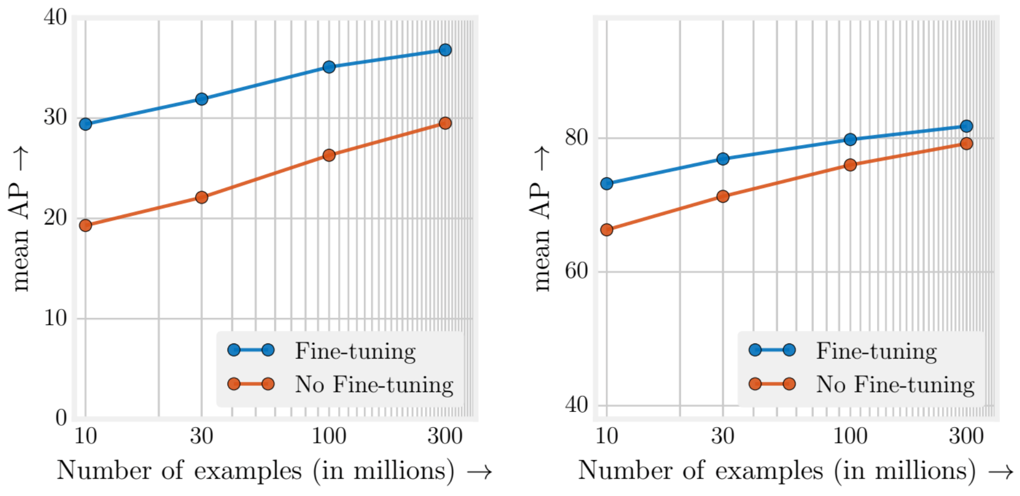
Receptive Field Size	Top-1 Accuracy (single frame)
79×79	75.2%
151×151	76.4%
299×299	76.6%

Table 2. Comparison of recognition performance when the size of the receptive field varies, but the computational cost is constant.

SOURCE (https://arxiv.org/pdf/1512.00567v3.pdf)

But **IF** you have done your last assignment, you'd realize that when we increase the receptive field of a network, we see heavy over-fitting.

RELATIONSHIP BETWEEN DATA AND VALIDATION ACCURACY



As you can see above it is exponential.

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_ICCV_2017_paper.pdf) *: 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

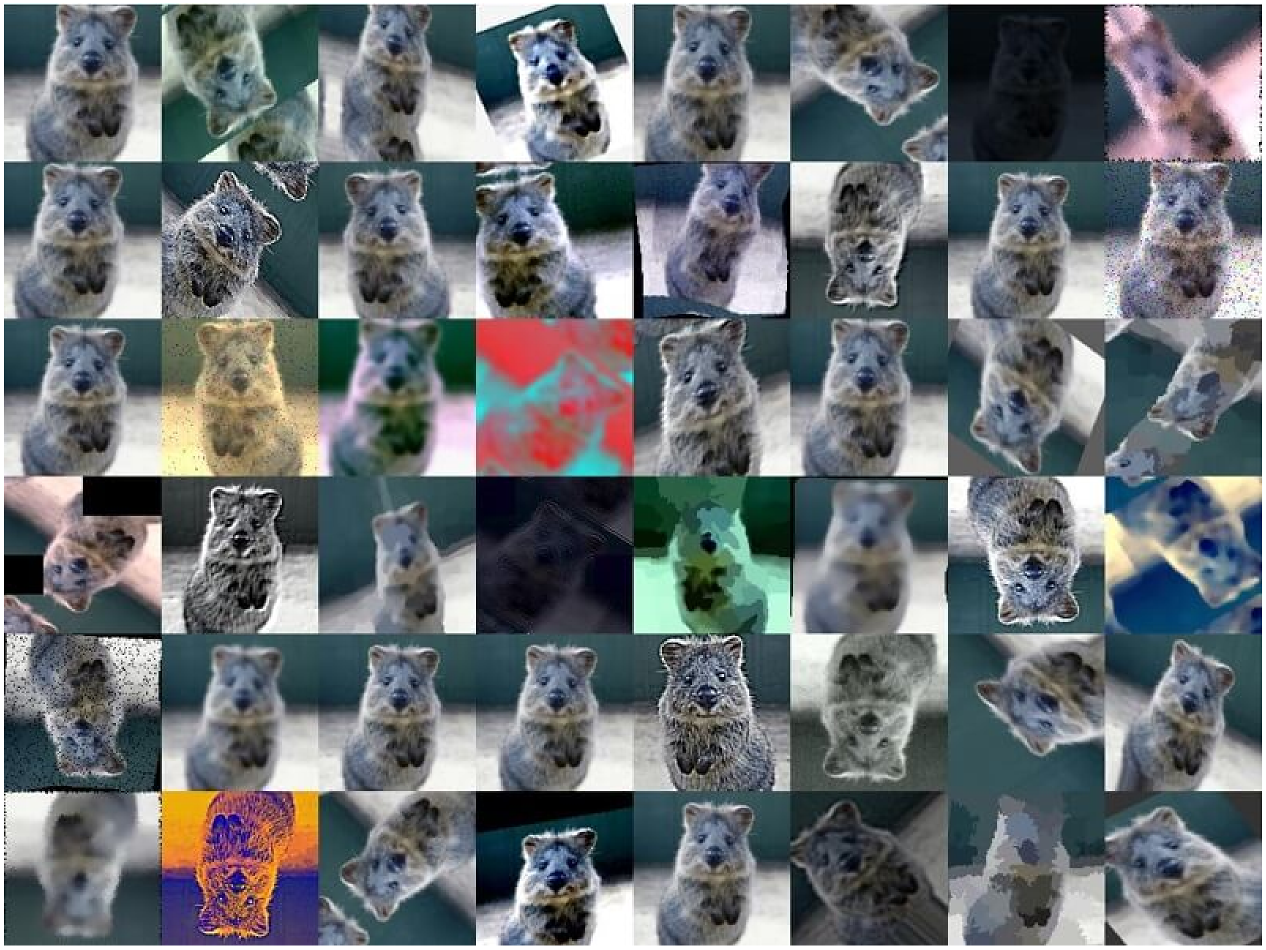
MMDA* - Middle-Class Man's Data Augmentation Strategies

RMDA* - Rich Man's Data Augmentation Strategies

PMDAs

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

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



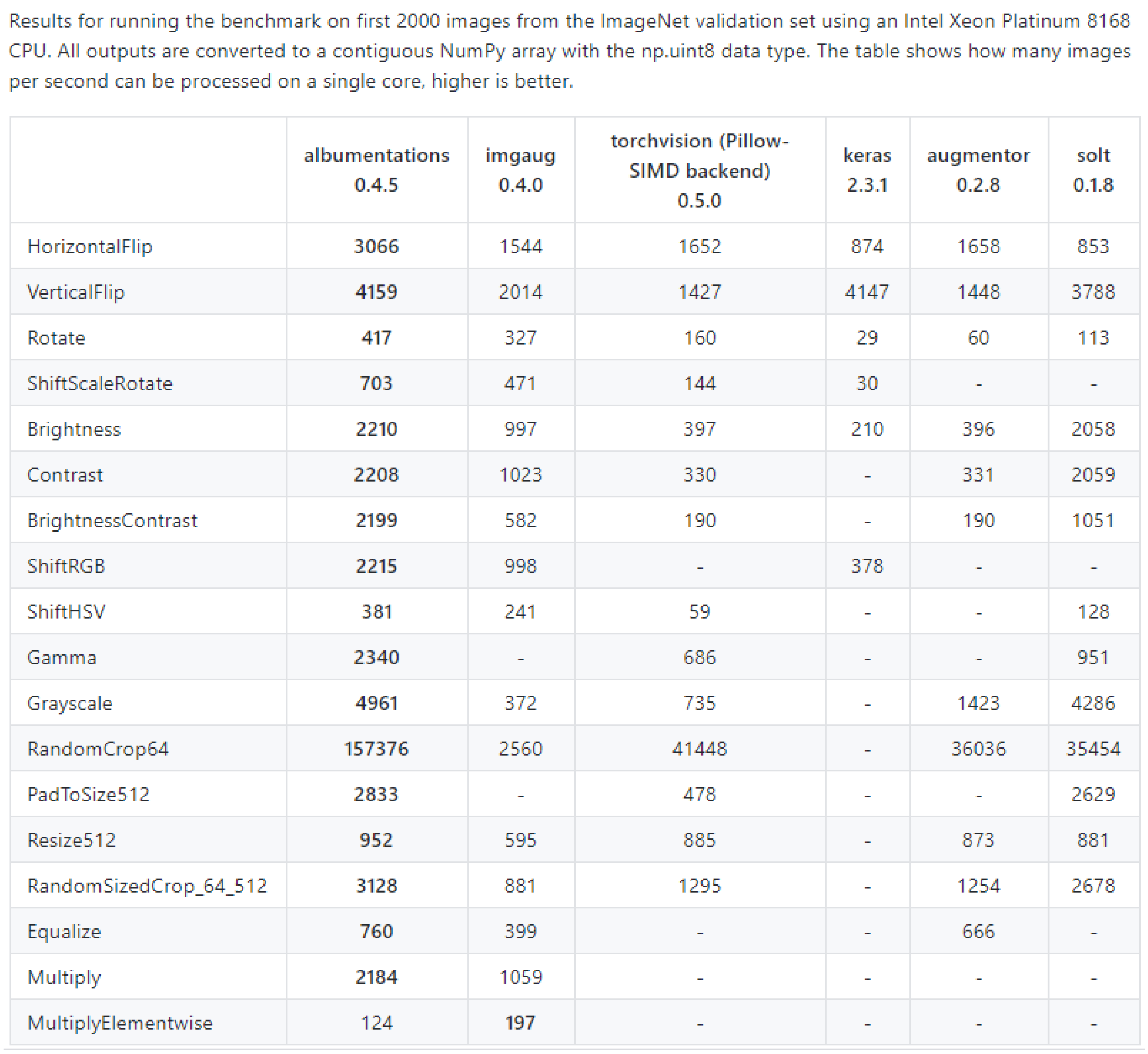
MMDA

We have 2 beautiful Libraries for MMDAs:

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

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

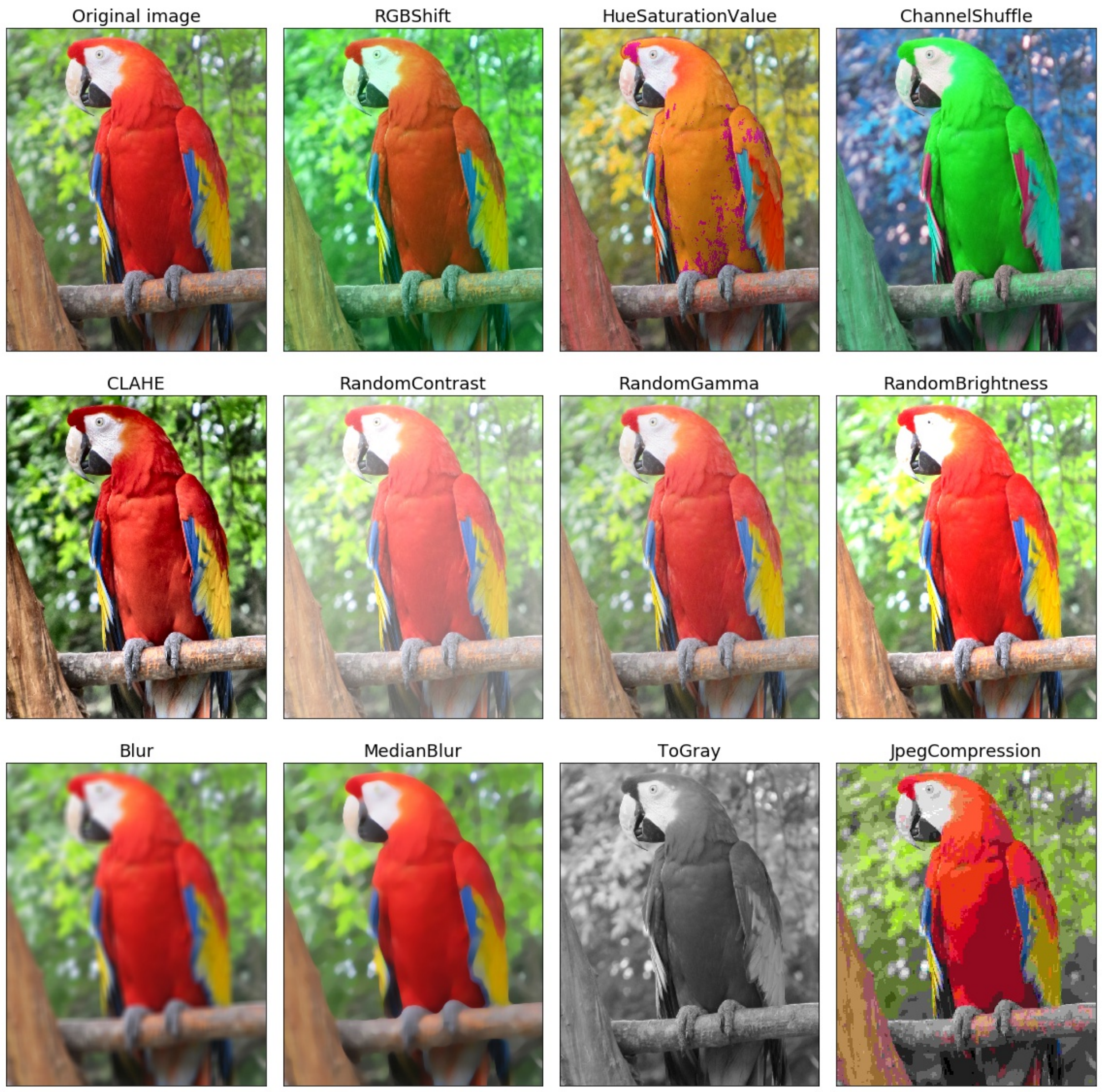
We are going to Focus on ALBUMENTATIONS because of it's easy integration with PyTorch.



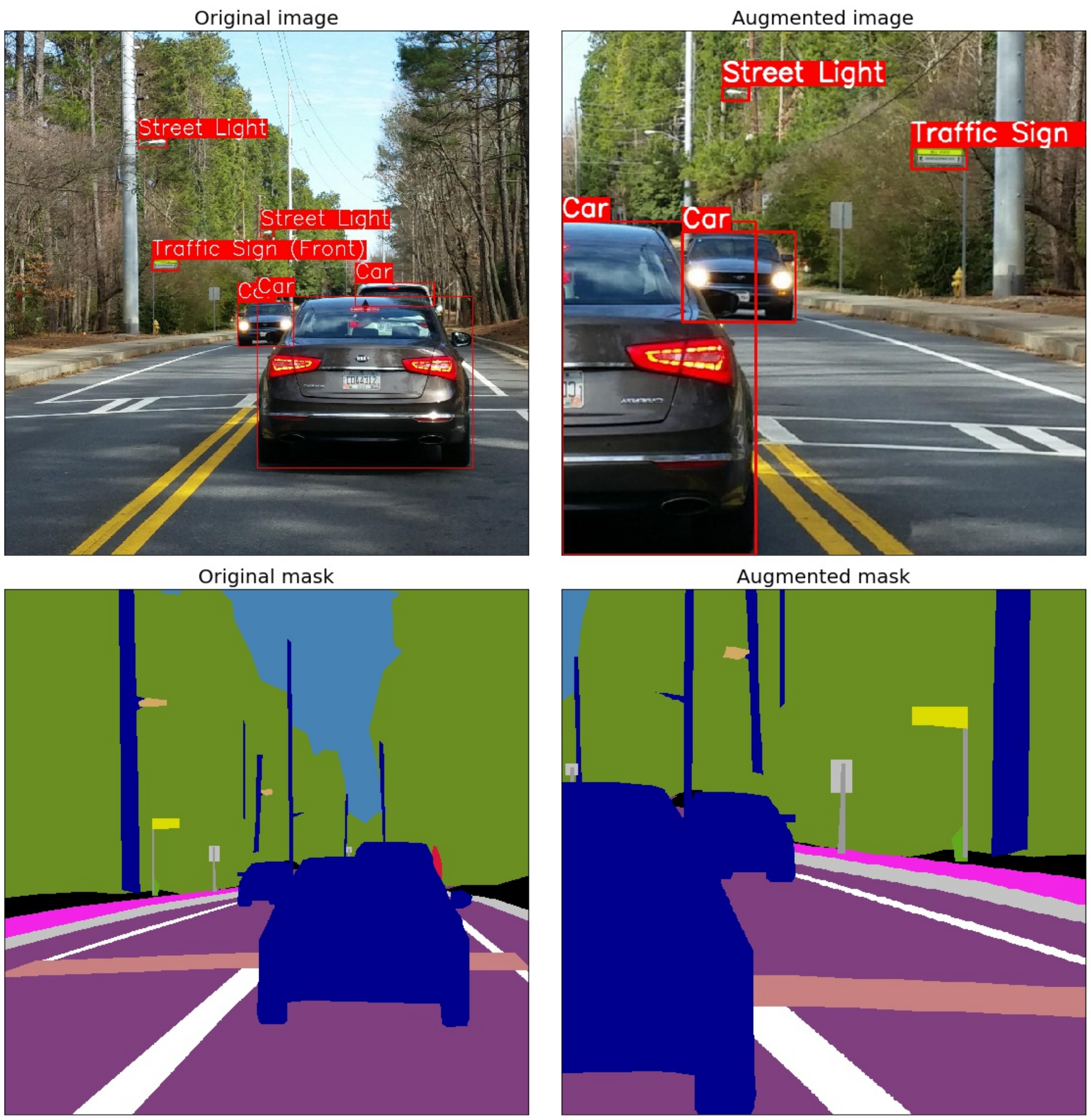
Results for running the benchmark on first 2000 images from the ImageNet validation set using an Intel Xeon Platinum 8168 CPU. All outputs are converted to a contiguous NumPy array with the np.uint8 data type. The table shows how many images per second can be processed on a single core, higher is better.

	albumentations 0.4.5	imgaug 0.4.0	torchvision (Pillow-SIMD backend) 0.5.0	keras 2.3.1	augmentor 0.2.8	solt 0.1.8
HorizontalFlip	**3066**	1544	1652	874	1658	853
VerticalFlip	**4159**	2014	1427	4147	1448	3788
Rotate	**417**	327	160	29	60	113
ShiftScaleRotate	**703**	471	144	30	-	-
Brightness	**2210**	997	397	210	396	2058
Contrast	**2208**	1023	330	-	331	2059
BrightnessContrast	**2199**	582	190	-	190	1051
ShiftRGB	**2215**	998	-	378	-	-
ShiftHSV	**381**	241	59	-	-	128
Gamma	**2340**	-	686	-	-	951
Grayscale	**4961**	372	735	-	1423	4286
RandomCrop64	**157376**	2560	41448	-	36036	35454
PadToSize512	**2833**	-	478	-	-	2629
Resize512	**952**	595	885	-	873	881
RandomSizedCrop_64_512	**3128**	881	1295	-	1254	2678
Equalize	**760**	399	-	-	666	-
Multiply	**2184**	1059	-	-	-	-
MultiplyElementwise	124	**197**	-	-	-	-

It has some nearly everything you'd want:



And smart way of handling Annotations



Some of the MMDAs are already included in Albumentations. But let's see the ones we are interested in.

[ELASTIC DISTORTION - 2003 (http://cognitivemedium.com/assets/rmnist/Simard.pdf)](http://cognitivemedium.com/assets/rmnist/Simard.pdf)

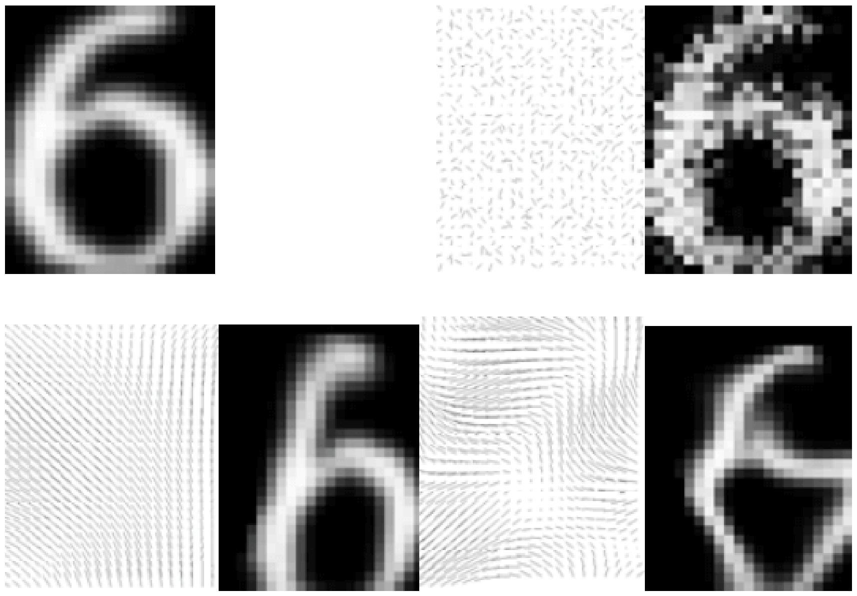
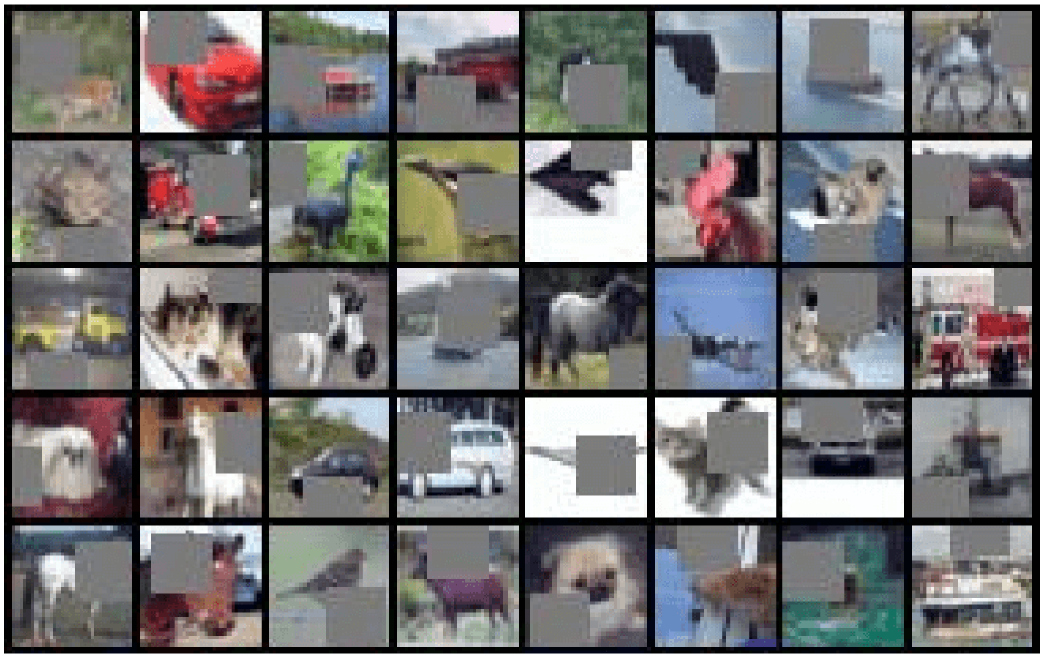


Figure 2. Top left: Original image. Right and bottom: Pairs of displacement fields with various smoothing, and resulting images when displacement fields are applied to the original image.

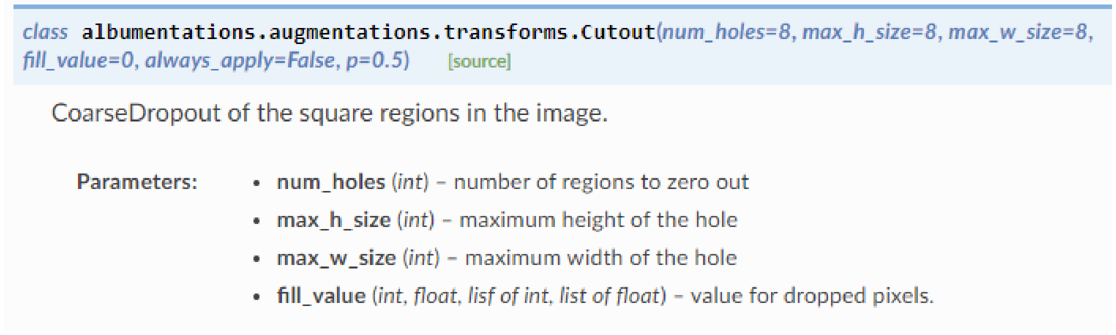
A very nice article on how to use this on Bengali (or others) is **here (https://www.kaggle.com/corochann/bengali-albumentations-data-augmentation-tutorial)**.

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

CutOut: Image speaks better on what it is:



What color is that cut-out grey?



class **albumentations.augmentations.transforms.Cutout**(*num_holes=8, max_h_size=8, max_w_size=8, fill_value=0, always_apply=False, p=0.5*) [source]

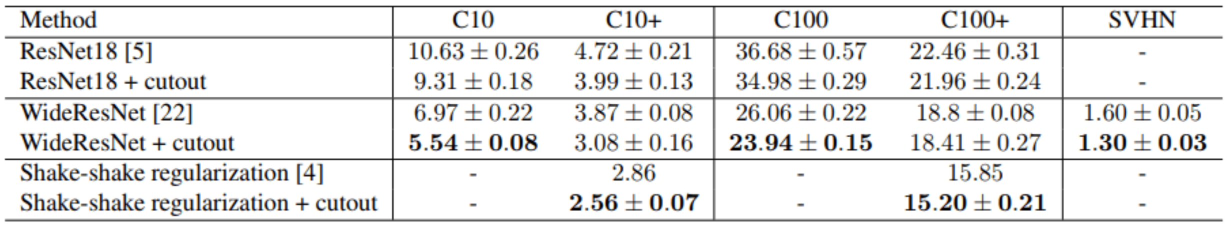
CoarseDropout of the square regions in the image.

Parameters:

- **num_holes** (*int*) – number of regions to zero out
- **max_h_size** (*int*) – maximum height of the hole
- **max_w_size** (*int*) – maximum width of the hole
- **fill_value** (*int, float, lisf of int, list of float*) – value for dropped pixels.

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

Why CutOut?

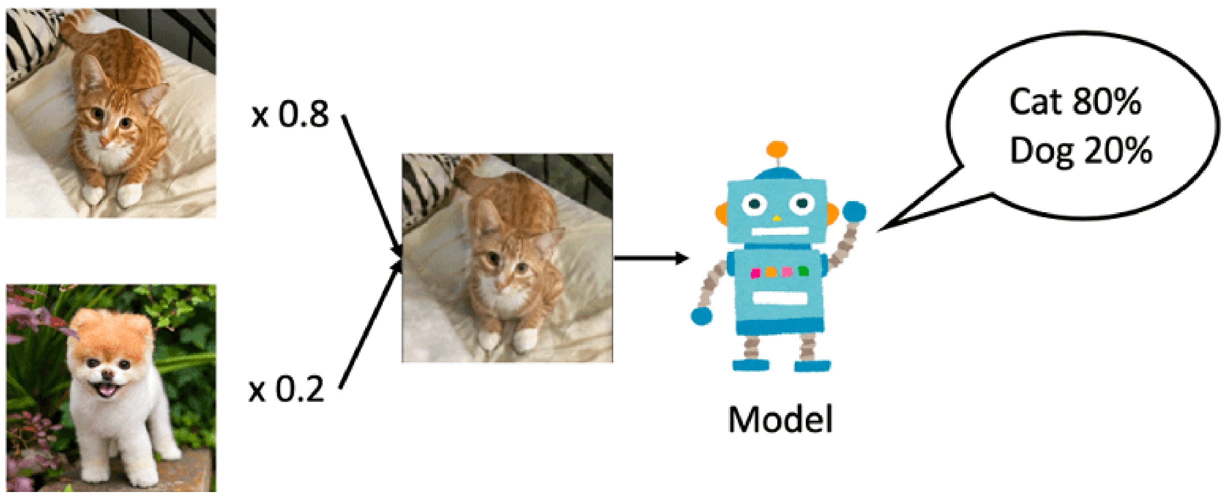


Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	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 - 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 similar to class label smoothing by mixing class labels with the ratio $\lambda : 1 - \lambda$.



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 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.

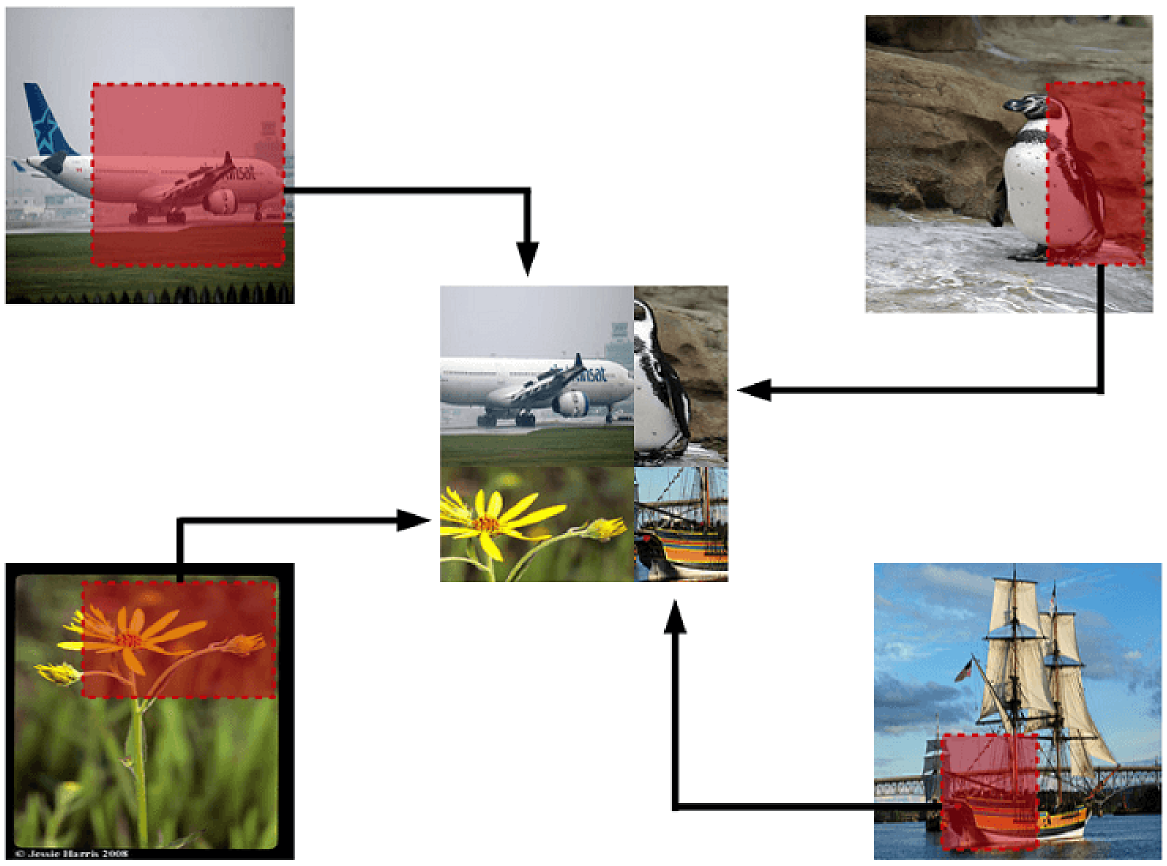
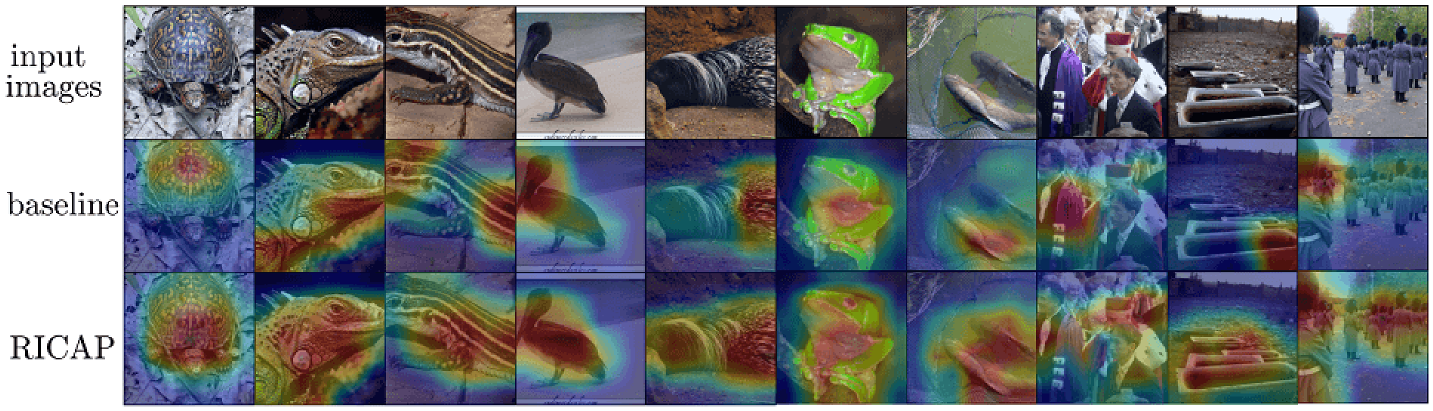


TABLE I
TEST ERROR RATES USING WIDERESNET ON THE CIFAR DATASET.

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

† indicates the results of our experiments.



Above we are seeing something called GradCAM's output. That would be our last topic of the day.

RMDA

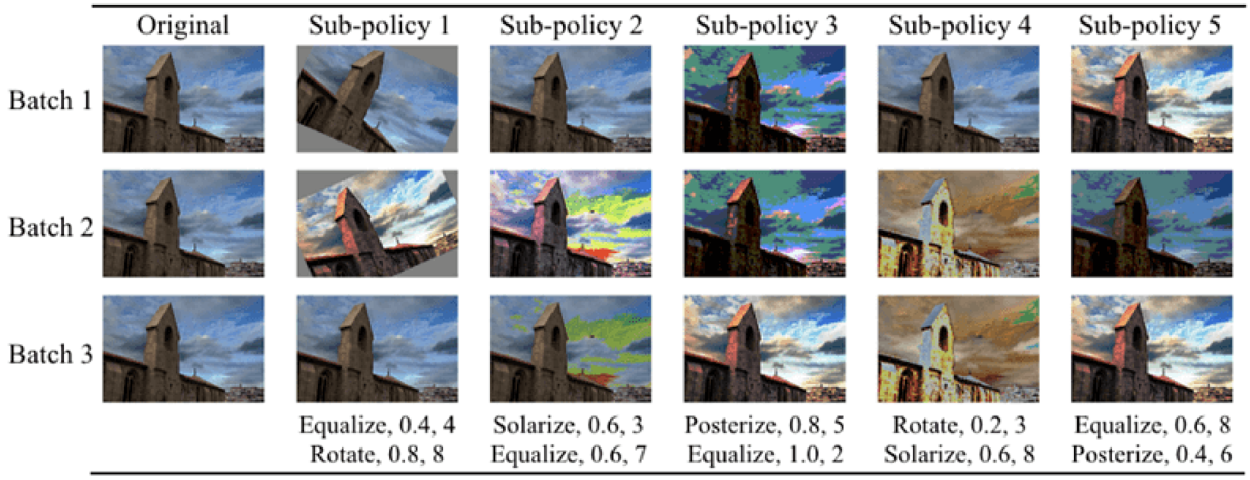
Reinforcement learning or AutoAugment 24th May 2018 (https://arxiv.org/abs/1805.09501)

Pros:

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

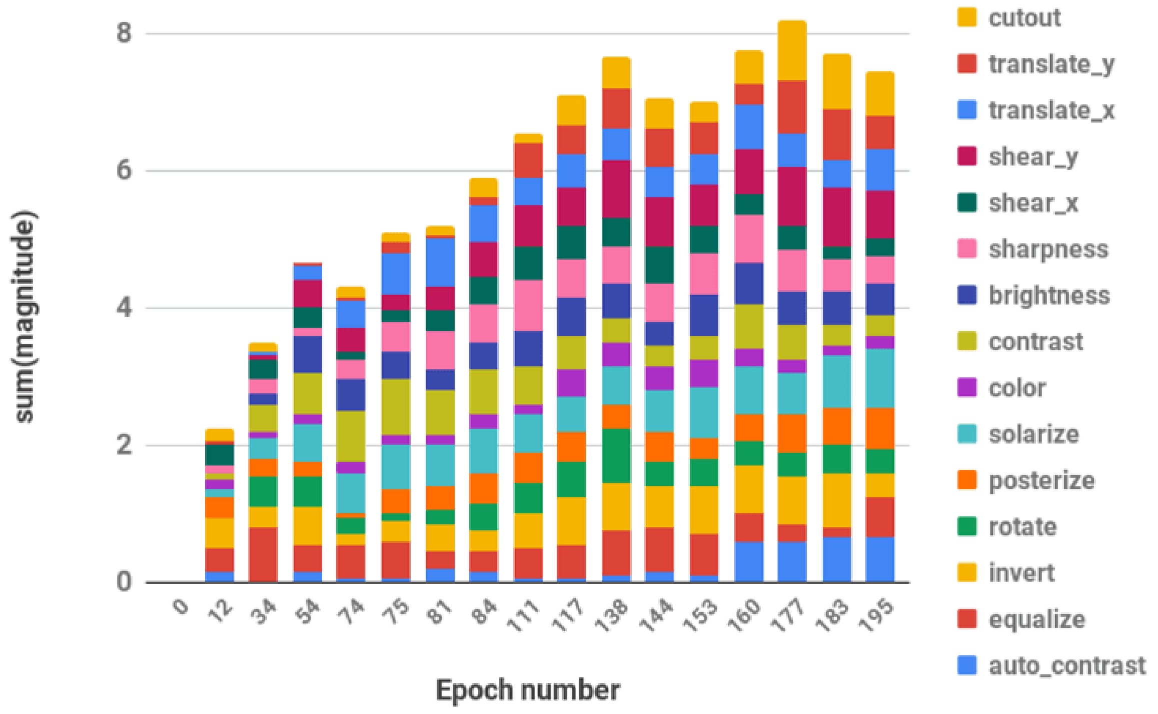
Cons:

1. Takes 5000 P100 GPU hours for CIFAR and 15000 GPU hours for ImageNet



[Population-Based Augmentation (https://arxiv.org/pdf/1905.05393v1.pdf)](https://arxiv.org/pdf/1905.05393v1.pdf)

(PBA), which generates nonstationary augmentation policy schedules instead of a fixed augmentation policy.



Patch Gaussian Augmentation 6 June 2019 (https://arxiv.org/pdf/1906.02611v1.pdf)

Prior work has argued that there is an inherent trade-off between robustness and accuracy, which is exemplified by standard data augment techniques such as Cutout, which improves clean accuracy but not robustness, and additive Gaussian noise, which improves robustness but hurts accuracy. To overcome this trade-off, we introduce Patch Gaussian, a simple augmentation scheme that adds noise to randomly selected patches in an input image. Models trained with Patch Gaussian achieve state of the art on the CIFAR-10 and ImageNet.

`Patch Gaussian` works by adding a WxW patch of Gaussian noise to the image (Figure 3)[3]. As with `Cutout`, the center of the patch is sampled to be within the image. By varying the size of this patch and the maximum standard deviation of noise sampled σ_{max}, we can interpolate between `Gaussian` (which applies additive Gaussian noise to the whole image) and an approximation of Cutout (which removes all information inside the patch). See Figure 9 for more examples.

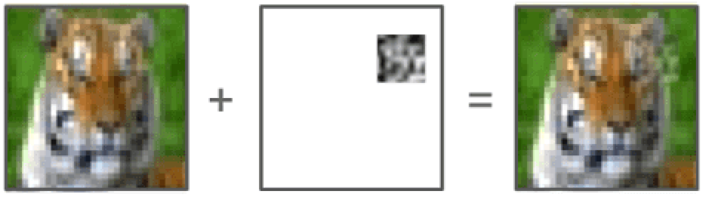
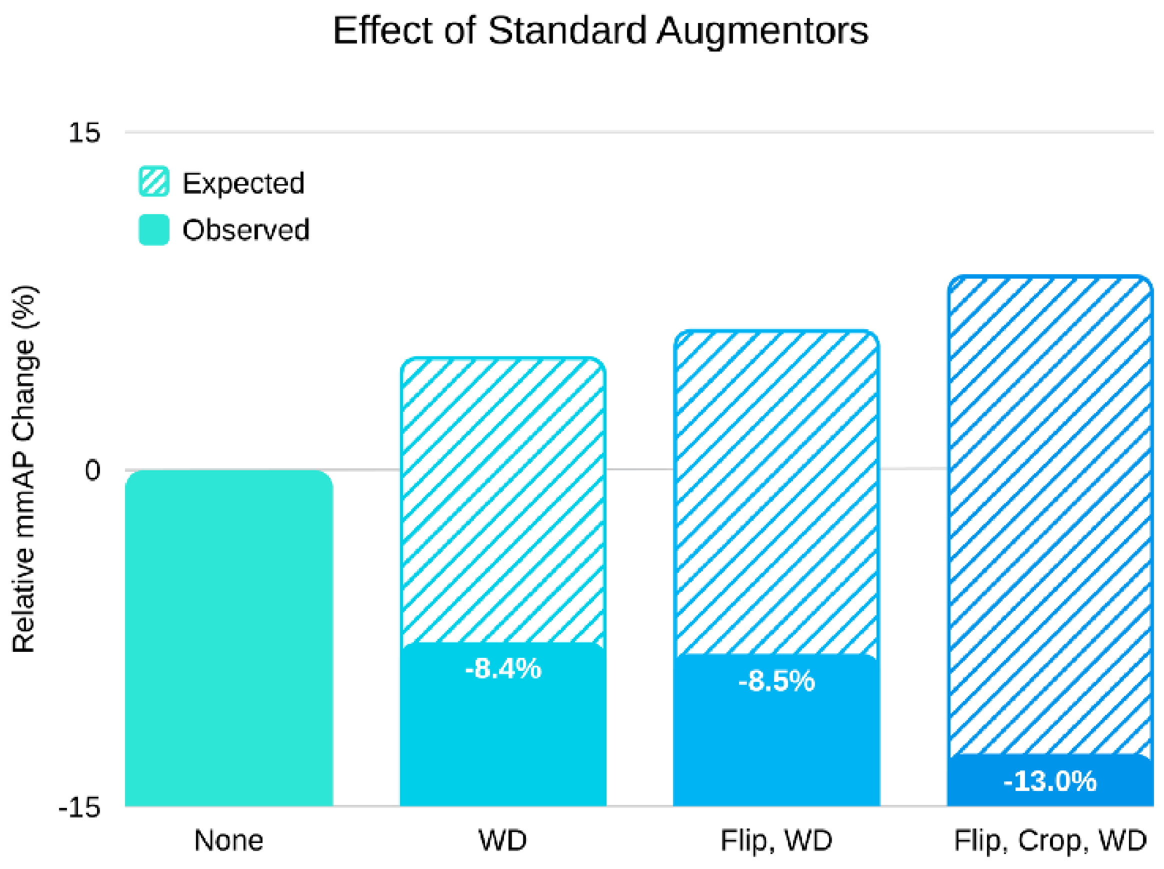
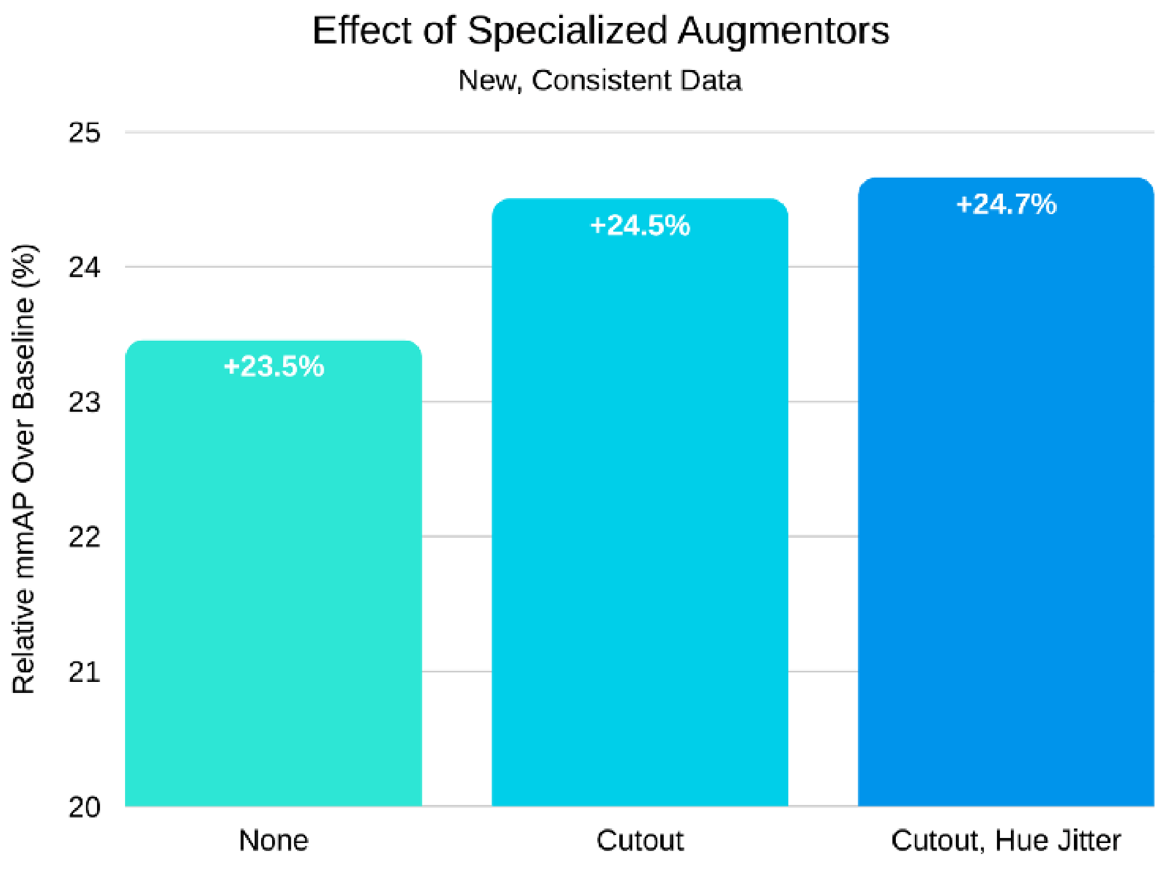


Figure 3: `Patch Gaussian` is the addition of Gaussian noise to pixels in a square patch. It allows us to interpolate between `Gaussian` and `Cutout`, approaching `Gaussian` with increasing patch size and `Cutout` with increasing σ.

BUT


Effect of Standard Augmentors
15
0
-15
Relative mmAP Change (%)
Expected
Observed
-8.4%
-8.5%
-13.0%
None
WD
Flip, WD
Flip, Crop, WD

ALSO



Let's cover a critical concept for our next step.

[Global Average Pooling (https://arxiv.org/pdf/1312.4400.pdf)](https://arxiv.org/pdf/1312.4400.pdf)

the fully connected layers are prone to overfitting, thus hampering the generalization ability of the overall network. Dropout is proposed by Hinton et al. [5] as a regularizer which randomly sets half of the activations to the fully connected layers to zero during training. It has improved generalization ability and largely prevents overfitting.

Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer.

1. it is more native to the convolution structure by enforcing correspondences between feature maps and categories, thus, the feature maps can be easily interpreted as categories-confidence maps.
2. No parameter to optimize in the global average pooling, thus overfitting is avoided at this layer
3. GAP sums out the spatial information, this is more robust to the spatial translation of the input.
4. GAP has ... parameters.

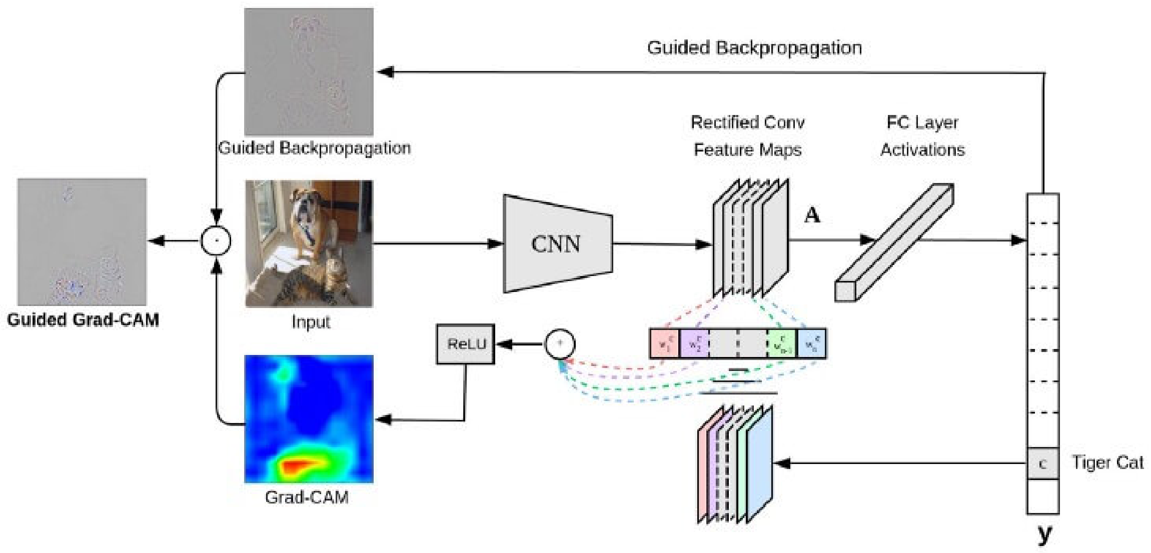
GradCAM

1. applicable to **any** CNN based network without requiring any changes (like GAP requirement)
2. can be applied to classification, captioning or Visual QA problems

Let's look at some **demos (http://gradcam.cloudcv.org/)**

Gradient-weighted Class Activation Mapping (GradCAM) uses the gradients of any target concept (say logits for 'dog' or even a caption), flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

We take the final convolutional feature map, and then we **weight** every channel in that feature with the gradient of the class with respect to the channel. It tells us how intensely the input image activates different channels by how important each channel is with regard to the class. It does not require any re-training or change in the existing architecture.



Steps:

1. Load a pre-trained model
2. Load an image which can be processed by this model (224x224 for VGG16 why?)
3. Infer the image and get the topmost class index
4. Take the output of the final convolutional layer
5. Compute the gradient of the class output value w.r.t to L feature maps
6. Pool the gradients over all the axes leaving out the channel dimension
7. Weigh the output feature map with the computed gradients (+ve)
8. Average the weighted feature maps along channels
9. Normalize the heat map to make the values between 0 and 1

DIAGNOSING DNN

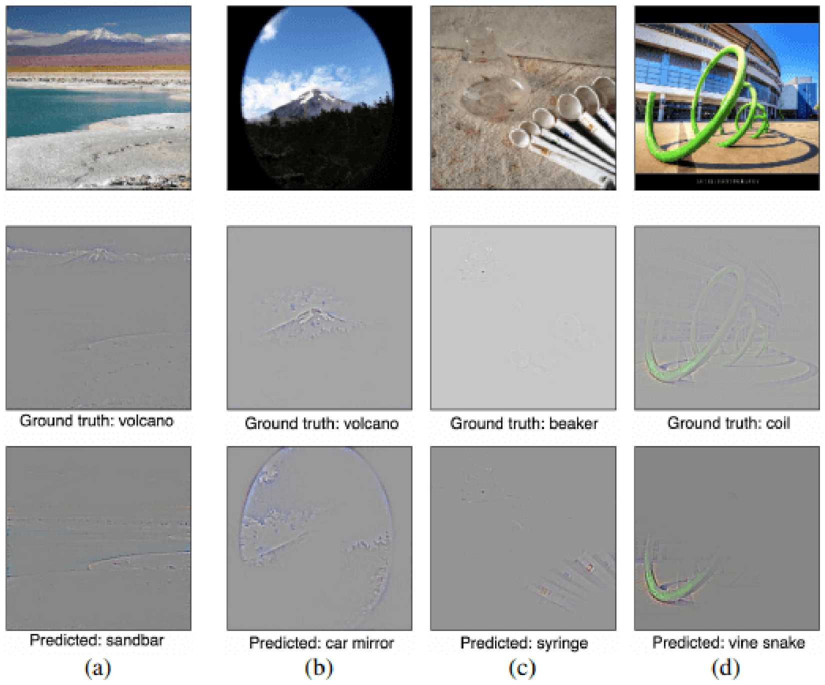


Figure 4: In these cases the model (VGG-16) failed to predict the correct class in its top 1 (a and d) and top 5 (b and c) predictions. Humans would find it hard to explain some of these predictions without looking at the visualization for the predicted class. But with Grad-CAM, these mistakes seem justifiable.

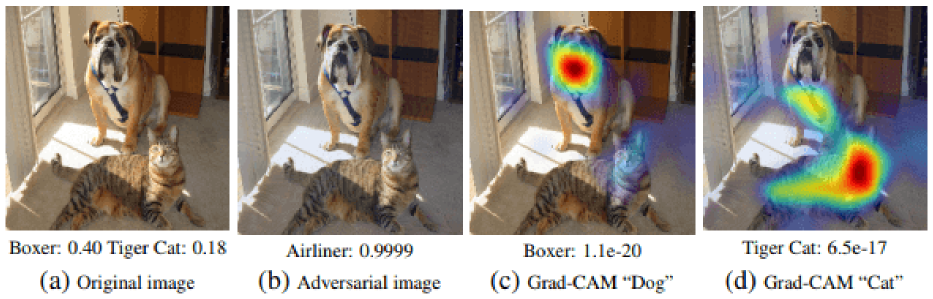
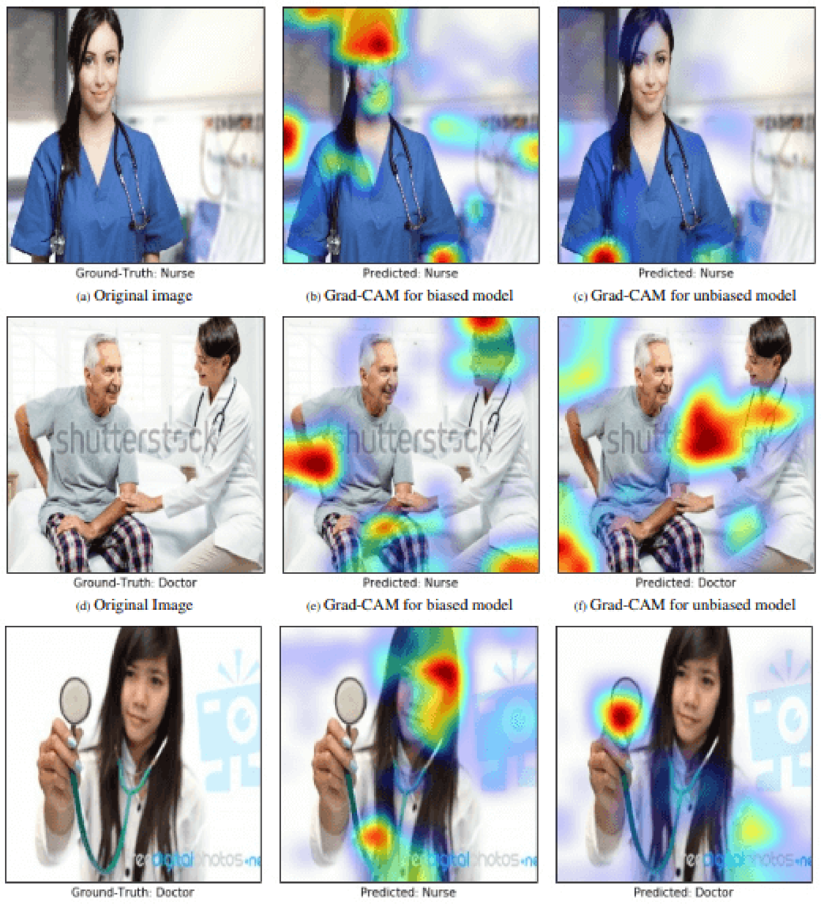


Figure 5: (a-b) Original image and the generated adversarial image for category "airliner". (c-d) Grad-CAM visualizations for the original categories "tiger cat" and "boxer (dog)" along with their confidence. Inspite of the network being completely fooled into thinking that the image belongs to "airliner" category with high confidence (>0.9999), Grad-CAM can localize the original categories accurately.

In this section we provide qualitative examples showing the explanations from the two models trained for distinguishing doctors from nurses- model1 which was trained on images (with an inherent bias) from a popular search engine, and model2 which was trained on a more balanced set of images from the same search engine.

As shown in Fig. A4, Grad-CAM visualizations of the model predictions show that the model had learned to look at the person's face / hairstyle to distinguish nurses from doctors, thus learning a gender stereotype.

Using the insights gained from the Grad-CAM visualizations, we balanced the dataset and retrained the model. The new model, model2 not only generalizes well to a balanced test set, it also looks at the right regions.



(a) Original image (b) Grad-CAM for biased model (c) Grad-CAM for unbiased model

(d) Original Image (e) Grad-CAM for biased model (f) Grad-CAM for unbiased model

(g) Original Image (h) Grad-CAM for biased model (i) Grad-CAM for unbiased model

Figure A4: Grad-CAM explanations for model1 and model2. In (a-c) we can see that even though both models made the right decision, the biased model (model1) was looking at the face of the person to decide if the person was a nurse (b), whereas the unbiased model, was looking at the short sleeves to make the decision (c). For example image (d) and example (g) the biased model made the wrong prediction (misclassifying a doctor as a nurse) by looking at the face and the hairstyle (e, h), where as the unbiased model made the right prediction looking at the white coat, and the stethoscope (f, i).

GradCAM At different Stages in a Conv Network

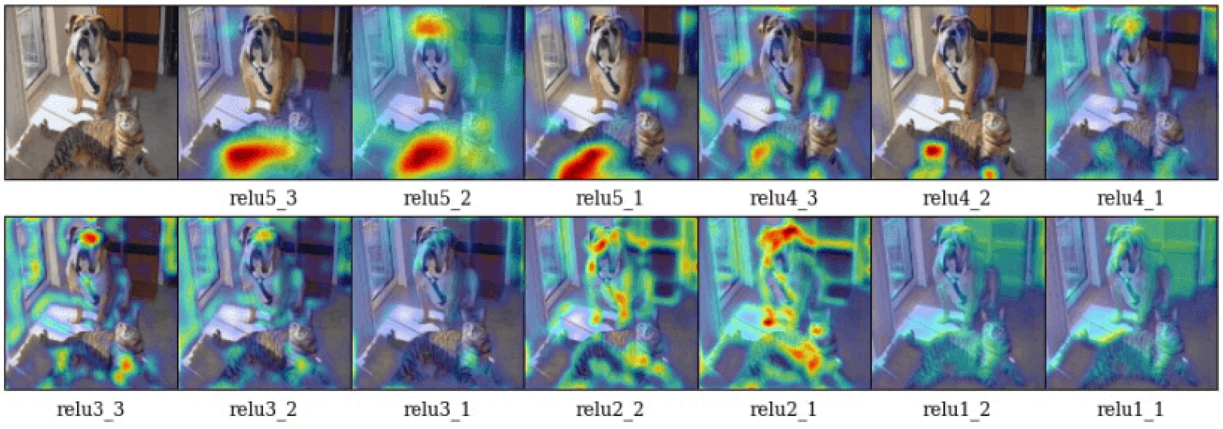


Figure A6: Grad-CAM at different convolutional layers for the 'tiger cat' class. This figure analyzes how localizations change qualitatively as we perform Grad-CAM with respect to different feature maps in a CNN (VGG16 [45]). We find that the best looking visualizations are often obtained after the deepest convolutional layer in the network, and localizations get progressively worse at shallower layers. This is consistent with our intuition described in Section 3 of main paper.

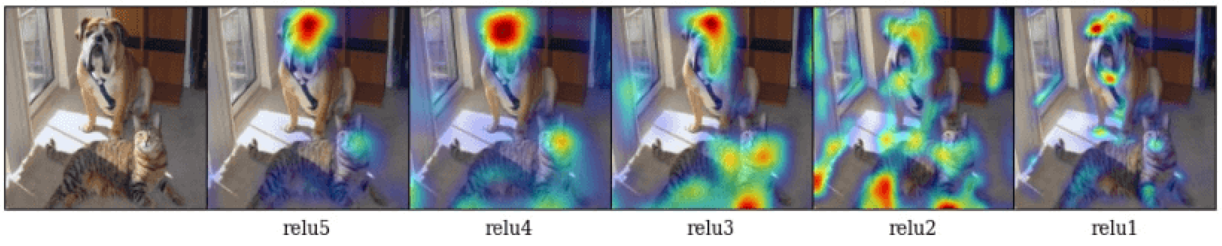


Figure A7: Grad-CAM localizations for "tiger cat" category for different rectified convolutional layer feature maps for AlexNet.

Assignment:

1. Move your last code's transformations to Albumentations. Apply ToTensor, HorizontalFlip, Normalize (at min) + More (for additional points)
2. Please make sure that your test_transforms are simple and only using ToTensor and Normalize
3. Implement GradCam function as a module.
4. Your final code (notebook file) must use imported functions to implement transformations and GradCam functionality
5. Target Accuracy is 87%
6. Submit answers to S9-Assignment-Solution.

Questions asked in S9-Assigment-Solution are:

1. Paste your Albumentation File Code
2. Paste your GradCam Module's code
3. Paste your notebook file code
4. What is your final Validation Accuracy?
5. Share the link to your Repo.

If there are errors/mis-representations in Repo, whole submission would be marked 0.

Q9: -Coding Exercise - 2 Hours - 1000 Pts.

1. You have 2 hours to solve this quiz.
2. Please make sure you have a proper Internet connection and a laptop before you proceed.
3. This is a coding exercise.
4. Even if you are working in a group, separate submissions are required, and you cannot just name your group member.

Video - Wednesday