Data Augmentation

Data Augmentation

- The technique of **artificially expanding a dataset** by applying various transformations or modifications to the existing data samples while preserving their labels or ground truth information.

- This technique is widely used in machine learning and deep learning to increase the diversity of training data, improve the generalization ability of models, and enhance their robustness against variations in input data.

1. Image Classification

- Augmenting images by applying transformations such as flipping (水平翻轉), rotation (旋轉), scaling (縮放), and cropping (裁切) to create additional training samples.

- This helps models learn to recognize objects from different viewpoints, orientations, and scales.

2. Object Detection

- Augmenting images by introducing variations in object positions (位置), sizes (尺寸), and occlusions (遮擋).

- This enables object detection models to better handle diverse scenarios and improve their ability to detect objects accurately under different conditions.

3. Semantic Segmentation (語義分割)

- Augmenting images by applying geometric transformations (幾何變換) and color variations (顏色變化) to generate more annotated pixel-level segmentation masks.
- This assists semantic segmentation models in learning to segment objects robustly across different scenes (不同場景) and lighting conditions (照明條件).

4. Natural Language Processing (NLP)

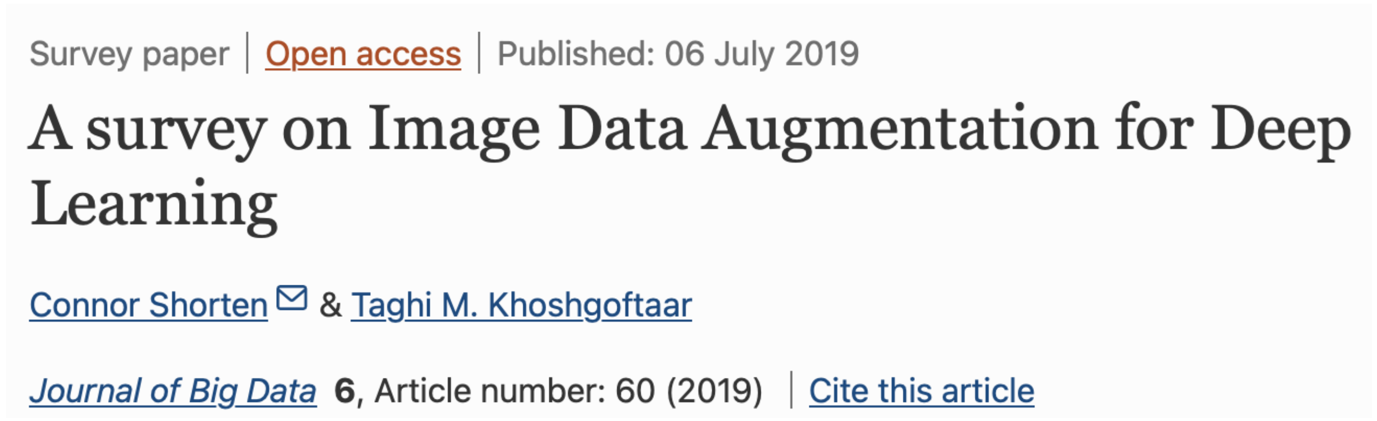
- Augmenting text data by introducing variations in word order (語句重排), synonyms (同義詞替換), paraphrases(改寫, 用不同的文字重新敘述一句話), and grammatical structures (句子重組).

- This aids NLP models in learning more robust language representations and improving their performance on tasks such as text classification, sentiment analysis, and machine translation.

5. Speech Recognition

- Augmenting audio data by adding background noise (背景噪聲), varying pitch (調整音調), speed (語速), and accent (口音).

- This helps speech recognition models become more resilient to environmental noise and accent variations, resulting in improved accuracy and robustness.


Survey paper | Open access | Published: 06 July 2019

A survey on Image Data Augmentation for Deep Learning

Connor Shorten & Taghi M. Khoshgoftaar

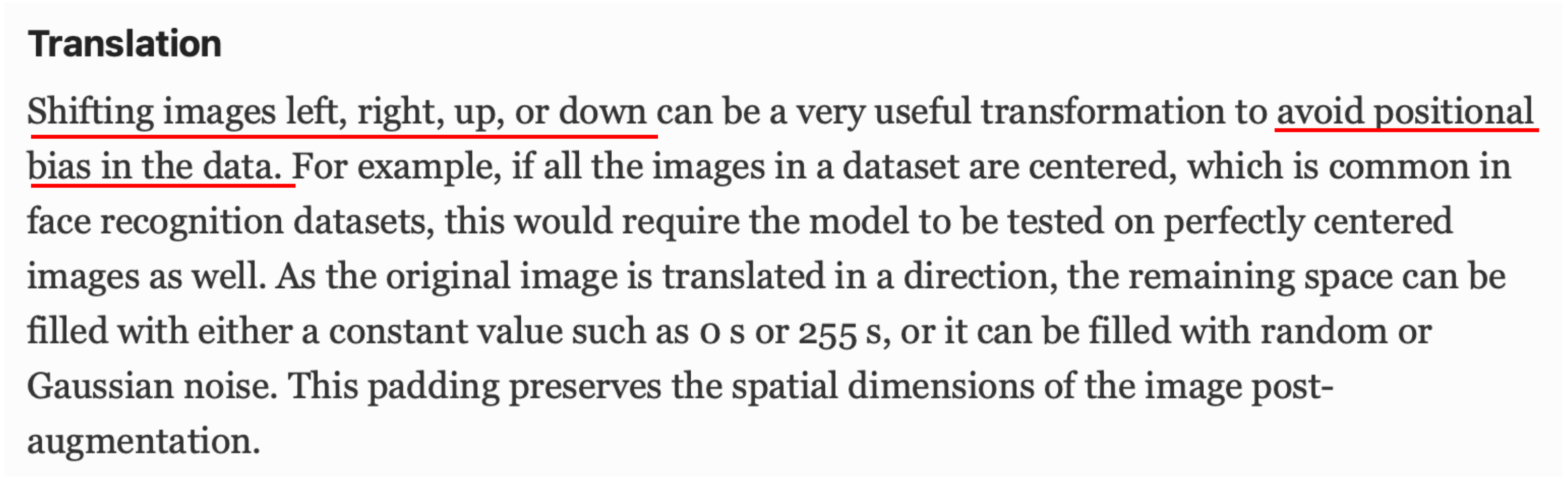
Journal of Big Data **6**, Article number: 60 (2019) | Cite this article

- **Data warping (數據扭曲):** transform existing images such that their label is preserved. (改變現有)
- **Oversampling:** create synthetic instances and add them to the training set. (合成新的)

A. Data Augmentations based on basic image manipulations

- **Flipping**
- **Color space**
- **Cropping**
- **Rotation**
- **Translation**
- **Noise injection**
- **Color space transformations**
- **Kernel filters**
- **Mixing images**
- **Random erasing**

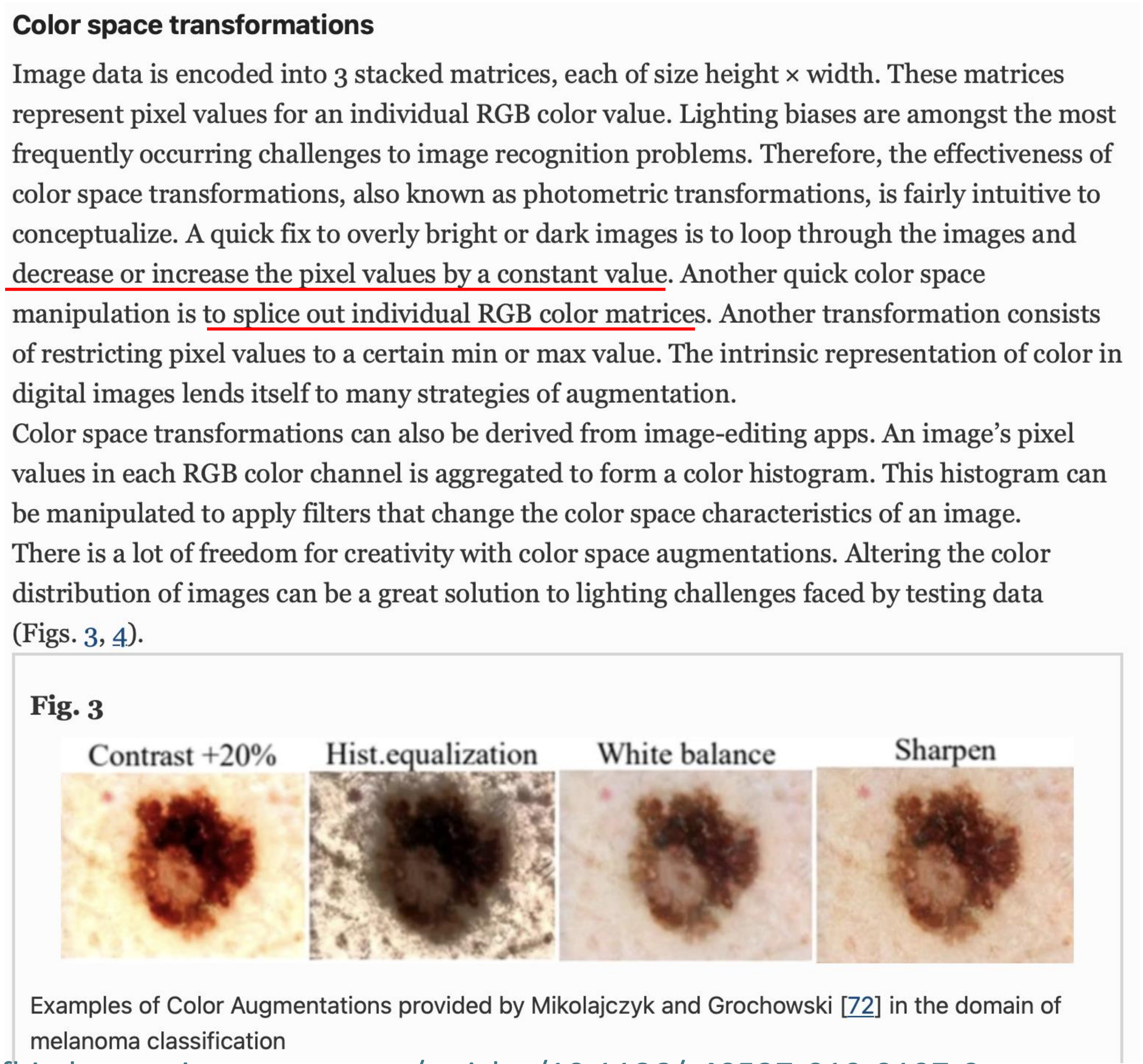
Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0



Translation

Shifting images left, right, up, or down can be a very useful transformation to avoid positional bias in the data. For example, if all the images in a dataset are centered, which is common in face recognition datasets, this would require the model to be tested on perfectly centered images as well. As the original image is translated in a direction, the remaining space can be filled with either a constant value such as 0 s or 255 s, or it can be filled with random or Gaussian noise. This padding preserves the spatial dimensions of the image post-augmentation.

Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0



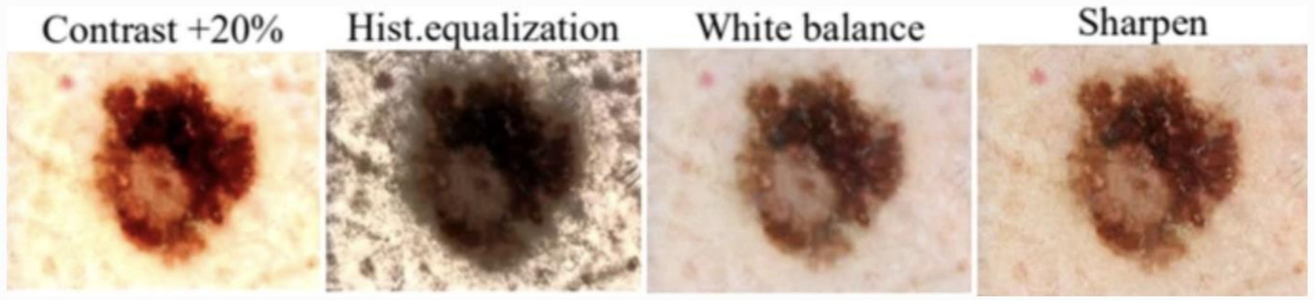
Color space transformations

Image data is encoded into 3 stacked matrices, each of size height × width. These matrices represent pixel values for an individual RGB color value. Lighting biases are amongst the most frequently occurring challenges to image recognition problems. Therefore, the effectiveness of color space transformations, also known as photometric transformations, is fairly intuitive to conceptualize. A quick fix to overly bright or dark images is to loop through the images and decrease or increase the pixel values by a constant value. Another quick color space manipulation is to splice out individual RGB color matrices. Another transformation consists of restricting pixel values to a certain min or max value. The intrinsic representation of color in digital images lends itself to many strategies of augmentation.

Color space transformations can also be derived from image-editing apps. An image's pixel values in each RGB color channel is aggregated to form a color histogram. This histogram can be manipulated to apply filters that change the color space characteristics of an image.

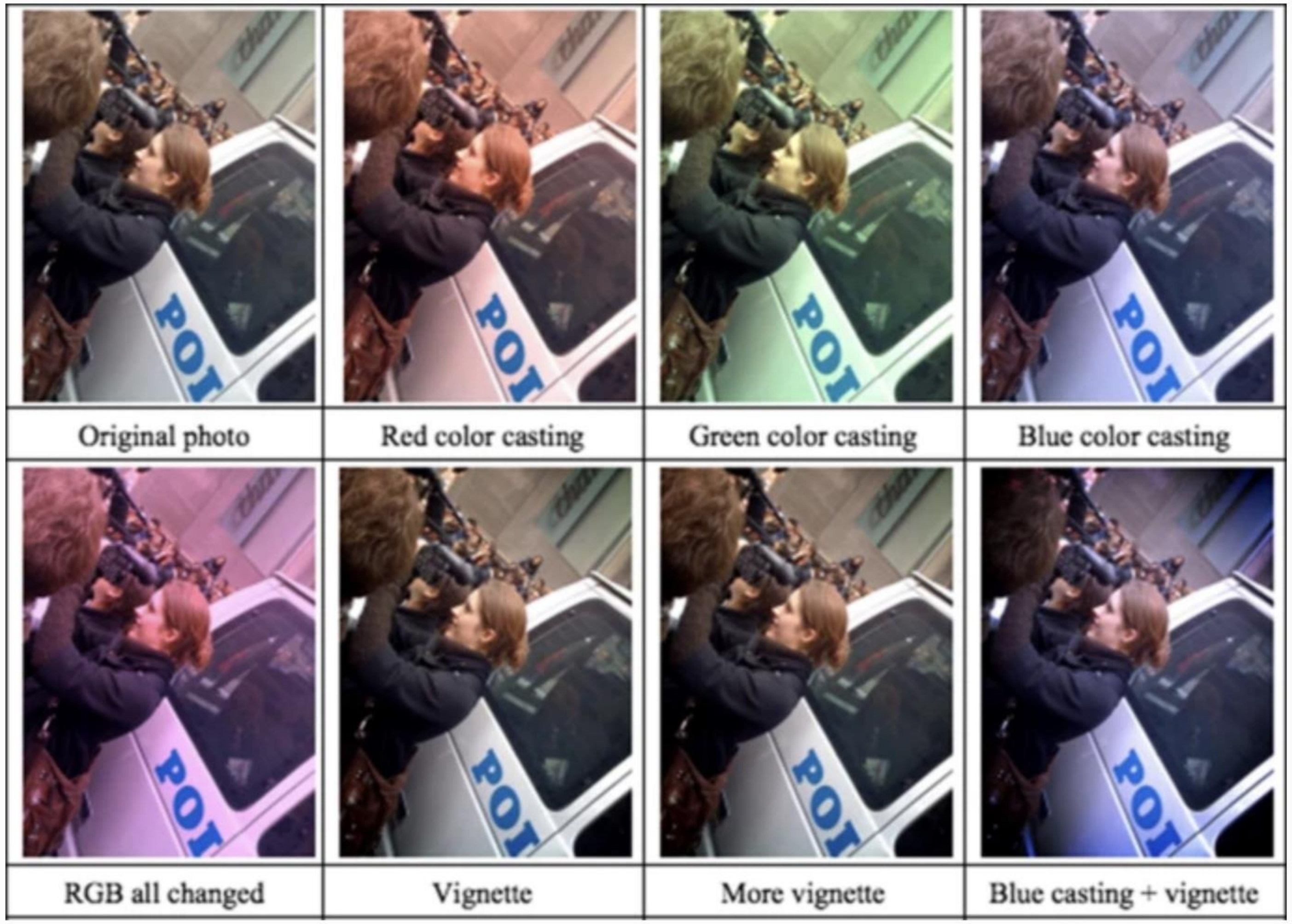
There is a lot of freedom for creativity with color space augmentations. Altering the color distribution of images can be a great solution to lighting challenges faced by testing data (Figs. 3, 4).

Fig. 3



Examples of Color Augmentations provided by Mikolajczyk and Grochowski [72] in the domain of melanoma classification

Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

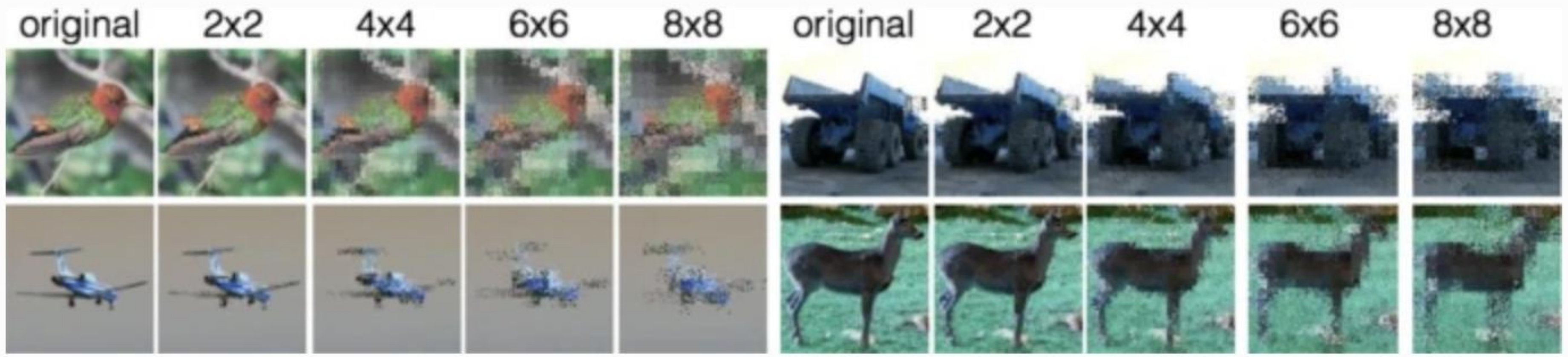


Examples of color augmentations tested by Wu et al. [127]

Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

Kernel filters (濾波器)

Fig. 6



blurring images for Data Augmentation could lead to higher resistance to motion blur during testing.
模糊處理 增強 運動模糊 抵抗力

Mixing images

Fig. 7

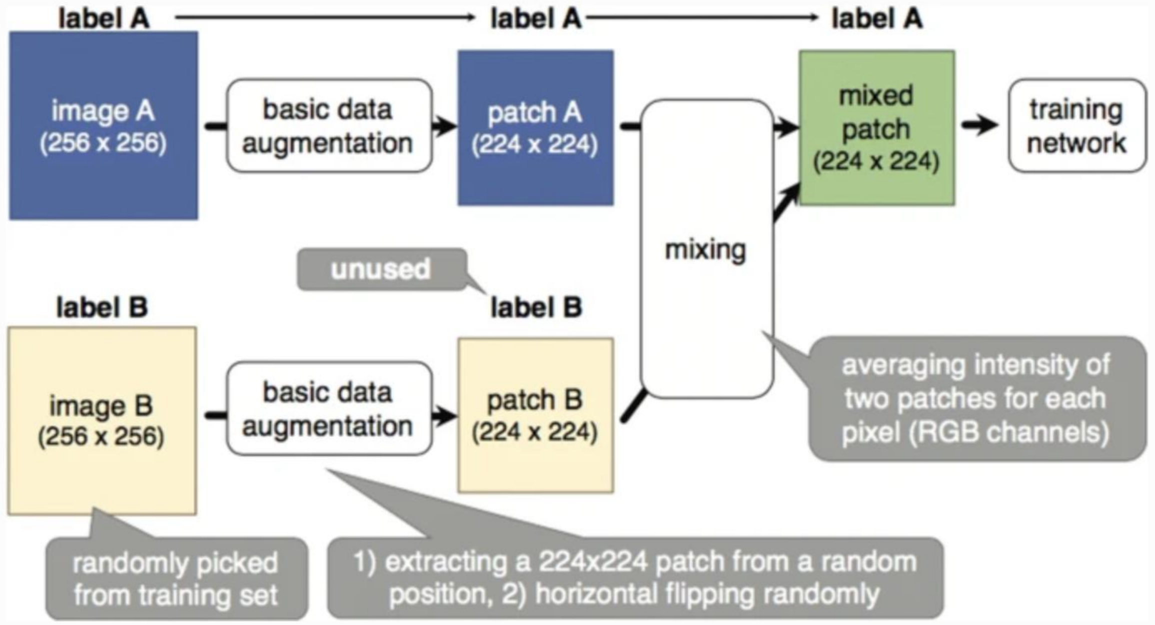
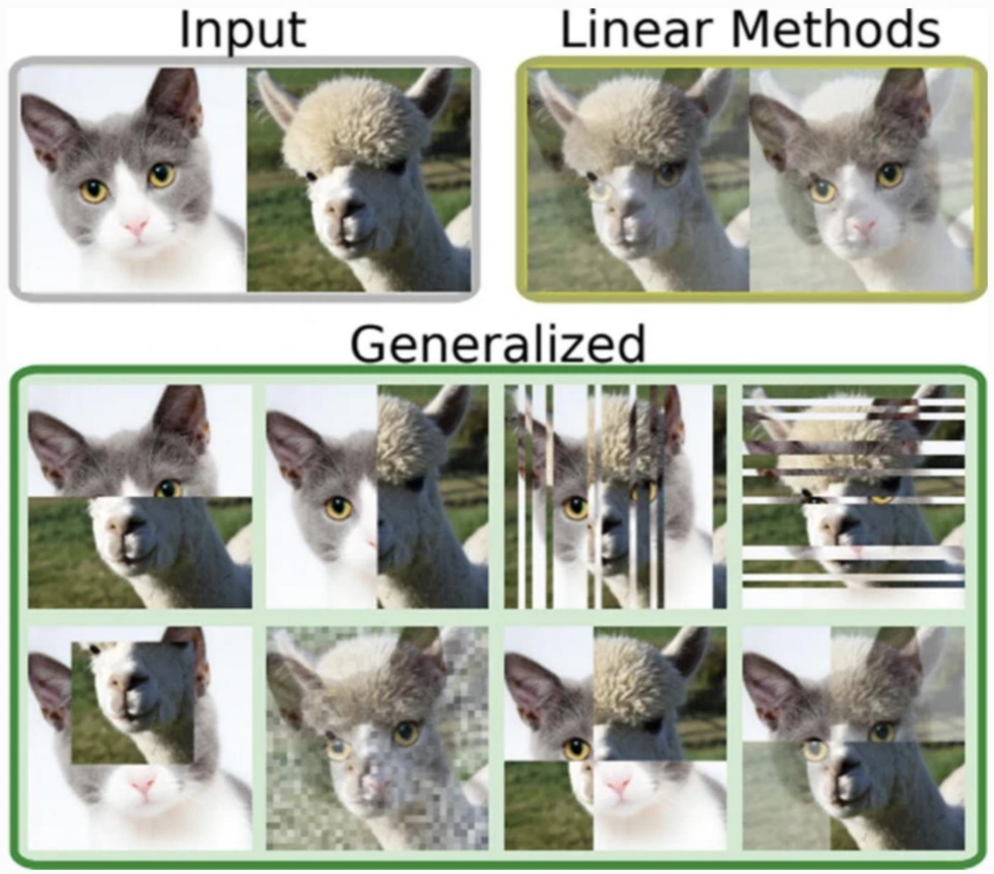


Fig. 9

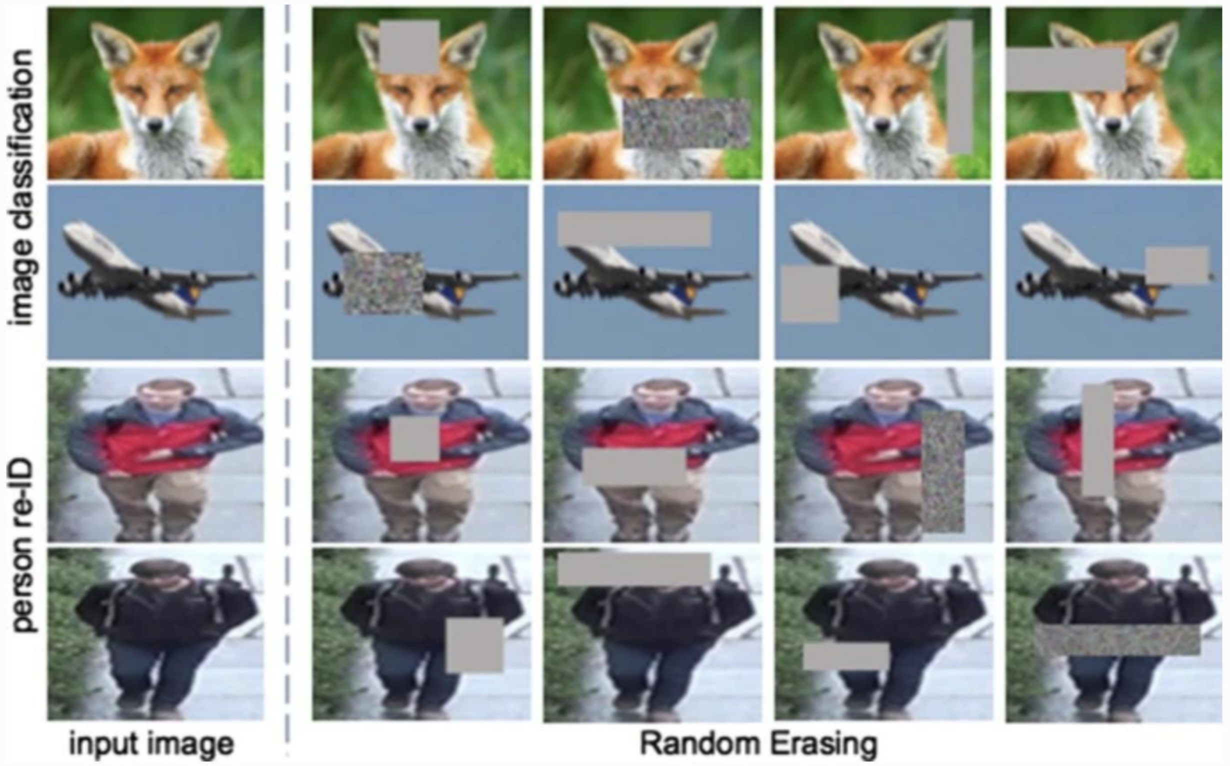


Mixing images together by averaging their pixel values is a very counterintuitive approach to Data Augmentation.

Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

Random erasing

Fig. 11



This technique was specifically designed to combat image recognition challenges due to occlusion.

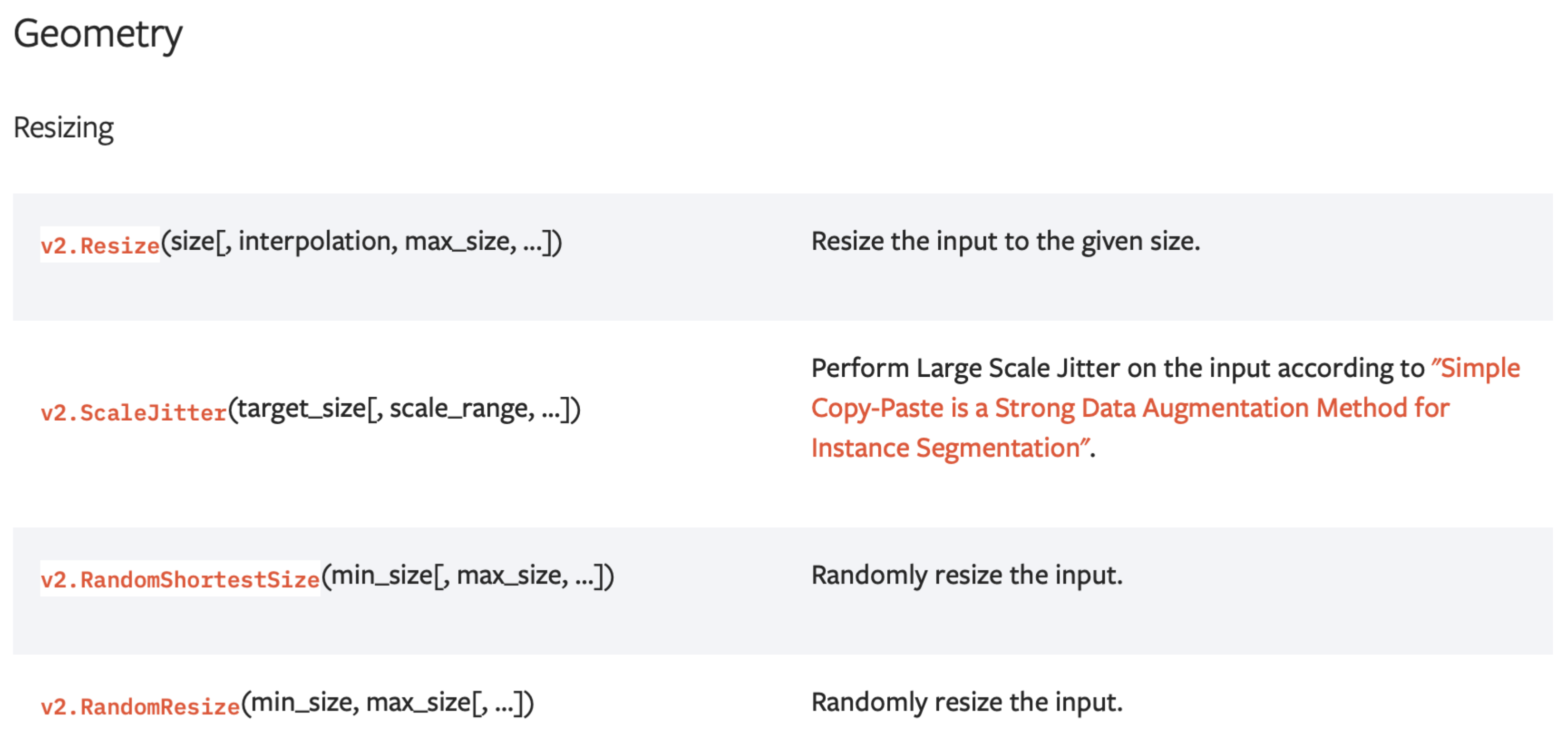
Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

B. Data Augmentations based on Deep Learning

- **Feature space augmentation**
- **Adversarial training**
- **GAN-based Data Augmentation**
- **Neural Style Transfer**
- **Meta learning Data Augmentations**
- **Neural augmentation**
- **Smart Augmentation**
- **AutoAugment**

Reference: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

Transforming and augmenting images

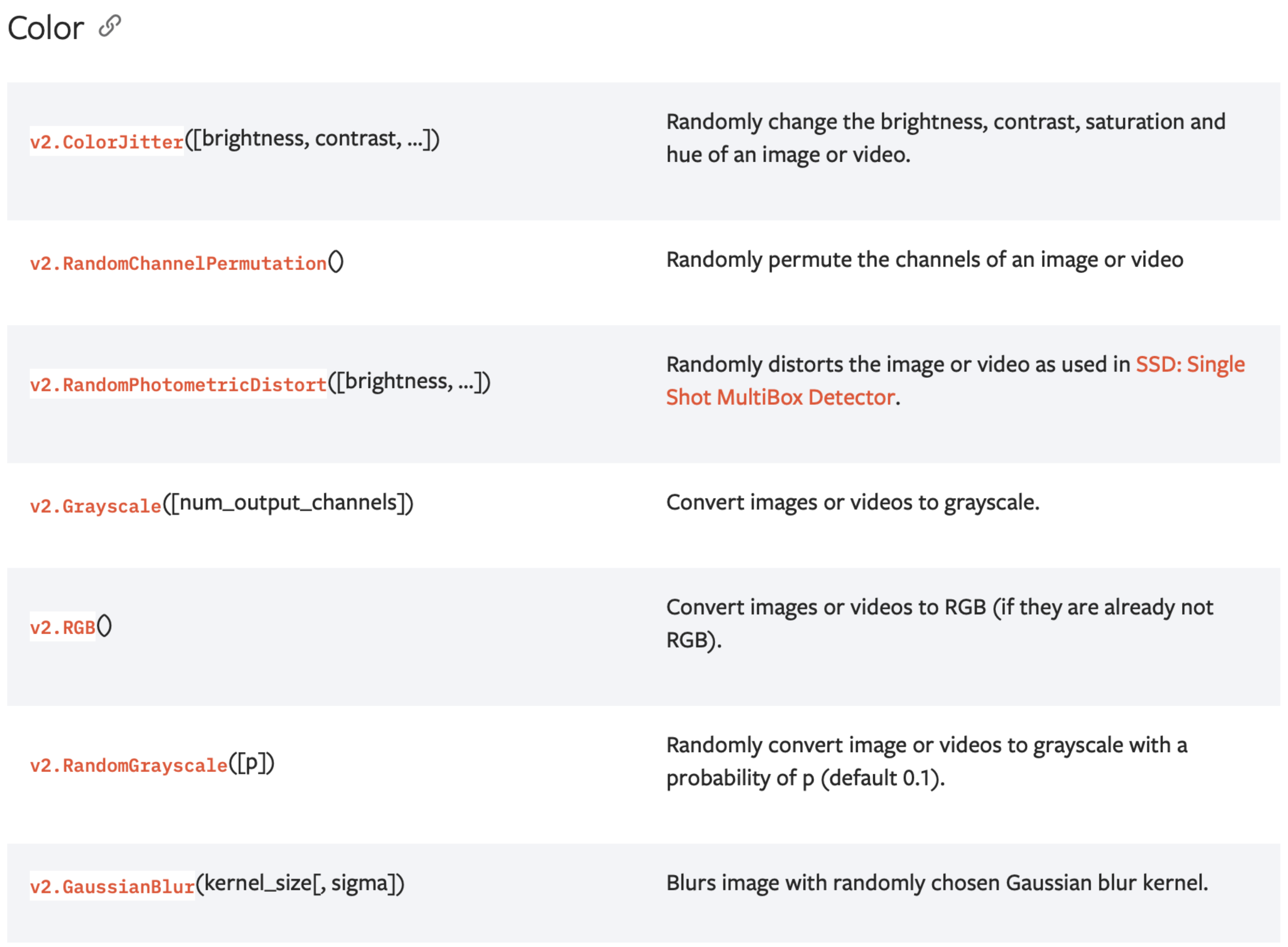


Geometry

Resizing

`v2.Resize`(size[, interpolation, max_size, ...])	Resize the input to the given size.
`v2.ScaleJitter`(target_size[, scale_range, ...])	Perform Large Scale Jitter on the input according to "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation".
`v2.RandomShortestSize`(min_size[, max_size, ...])	Randomly resize the input.
`v2.RandomResize`(min_size, max_size[, ...])	Randomly resize the input.

https://pytorch.org/vision/stable/transforms.html



Color

`v2.ColorJitter`([brightness, contrast, ...])	Randomly change the brightness, contrast, saturation and hue of an image or video.
`v2.RandomChannelPermutation`()	Randomly permute the channels of an image or video
`v2.RandomPhotometricDistort`([brightness, ...])	Randomly distorts the image or video as used in SSD: Single Shot MultiBox Detector.
`v2.Grayscale`([num_output_channels])	Convert images or videos to grayscale.
`v2.RGB`()	Convert images or videos to RGB (if they are already not RGB).
`v2.RandomGrayscale`([p])	Randomly convert image or videos to grayscale with a probability of p (default 0.1).
`v2.GaussianBlur`(kernel_size[, sigma])	Blurs image with randomly chosen Gaussian blur kernel.

https://pytorch.org/vision/stable/transforms.html

Auto-Augmentation

AutoAugment is a common Data Augmentation technique that can improve the accuracy of Image Classification models. Though the data augmentation policies are directly linked to their trained dataset, empirical studies show that ImageNet policies provide significant improvements when applied to other datasets. In TorchVision we implemented 3 policies learned on the following datasets: ImageNet, CIFAR10 and SVHN. The new transform can be used standalone or mixed-and-matched with existing transforms:

`v2.AutoAugment`([policy, interpolation, fill])	AutoAugment data augmentation method based on ”AutoAugment: Learning Augmentation Strategies from Data”.
`v2.RandAugment`([num_ops, magnitude, ...])	RandAugment data augmentation method based on ”RandAugment: Practical automated data augmentation with a reduced search space”.
`v2.TrivialAugmentWide`([num_magnitude_bins, ...])	Dataset-independent data-augmentation with TrivialAugment Wide, as described in ”TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation”.
`v2.AugMix`([severity, mixture_width, ...])	AugMix data augmentation method based on ”AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty”.

```
import torchvision.transforms as transforms

# Image flipping
flip_transform = transforms.RandomHorizontalFlip(p=0.5) # Horizontal flip with a probability of 0.5

# Rotation
rotation_transform = transforms.RandomRotation(degrees=30) # Random rotation within ±30 degrees

# Scaling
scale_transform = transforms.RandomResizedCrop(size=(224, 224), scale=(0.8, 1.0))

# Translation
translate_transform = transforms.RandomAffine(degrees=0, translate=(0.2, 0.2)) # Translation within ±0.2
times image width and height

# Brightness adjustment
brightness_transform = transforms.ColorJitter(brightness=0.2) # Adjust brightness within ±0.2 times

# Contrast adjustment
contrast_transform = transforms.ColorJitter(contrast=0.2) # Adjust contrast within ±0.2 times

# Noise injection
noise_transform = transforms.Compose([
transforms.ToTensor(),
transforms.Lambda(lambda x: x + 0.1 * torch.randn_like(x)) # Add Gaussian noise with a standard
deviation of 0.1
])
```