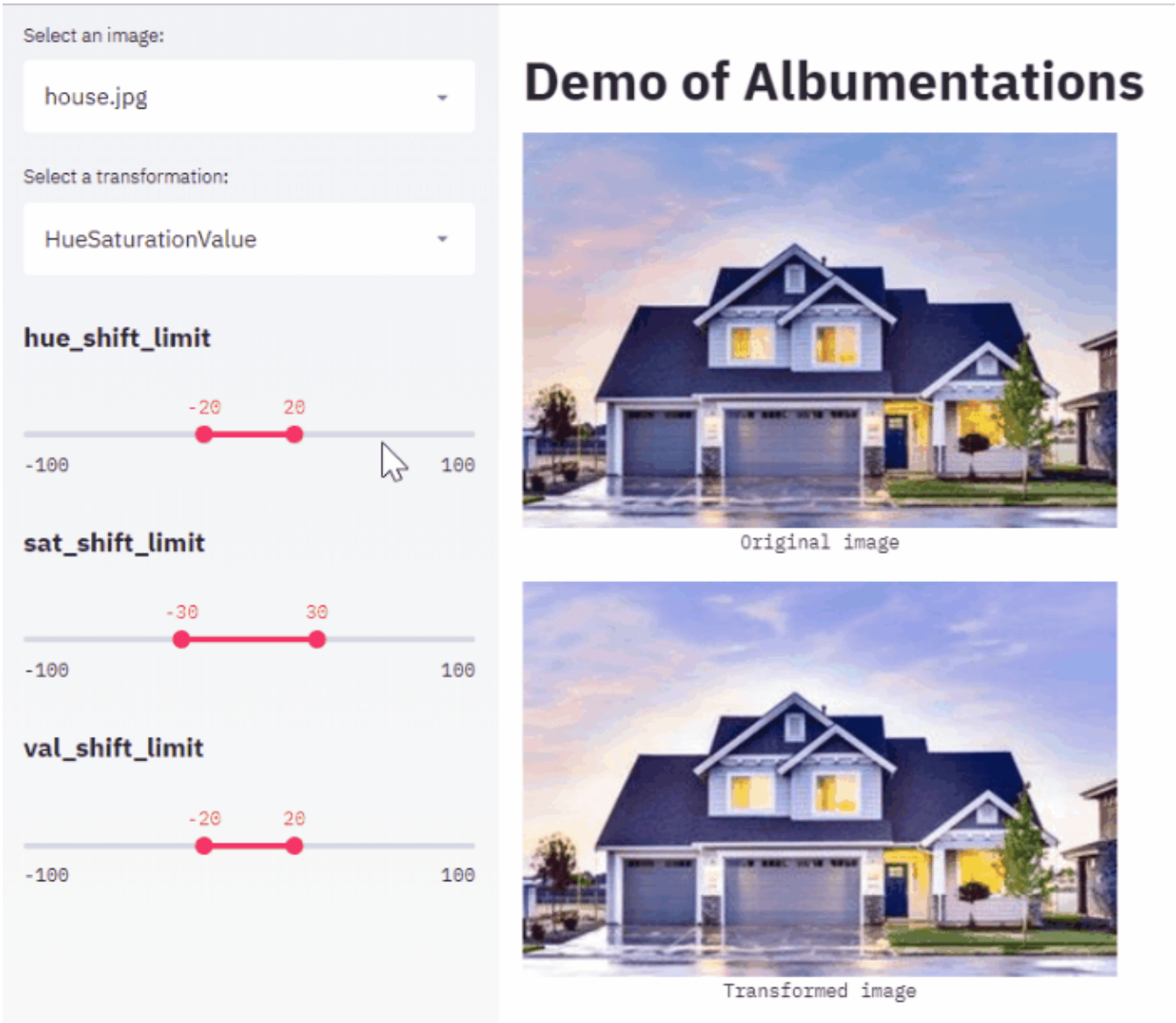
Explore image augmentations using a convenient tool

The easy way to understand how image transformations work powered by Albumentations and Streamlit

Ilia Larchenko

Dec 10, 2019 · 9 min read ★



The interface of the service. Original photo by Binyamin Mellish from Pexels

Image augmentation is an essential concept that became an industry-standard in Computer Vision. It makes ML models more precise and robust, improves generalization, and helps to create high accuracy models with minimal data.

There are ready to use tools to apply different transformations to your training pipeline. My choice is Albumentations library

But to use it effectively, you should understand how it works. In this article, I will tell you how to choose the right set of transformations and present you with a simple tool for the visualization and exploration of different augmentation techniques.

If you are familiar with the “augmentations” concept and already use Albumentations, you can follow this link right now; there you can play with different transformations in the interactive interface and see the result in real-time (like on a gif above), or you can even take the source code and run it locally.

Otherwise, I advise you to continue reading and check it afterward.

Original photo by Jose Mueses from Pexels

. . .

What is augmentation?

Merely speaking, augmentation is a technique of generating additional training samples from the existing ones.

Let's imagine you are training a CV model to classify images with 'cats' and 'dogs.' But for some reason, you have only a few training images.

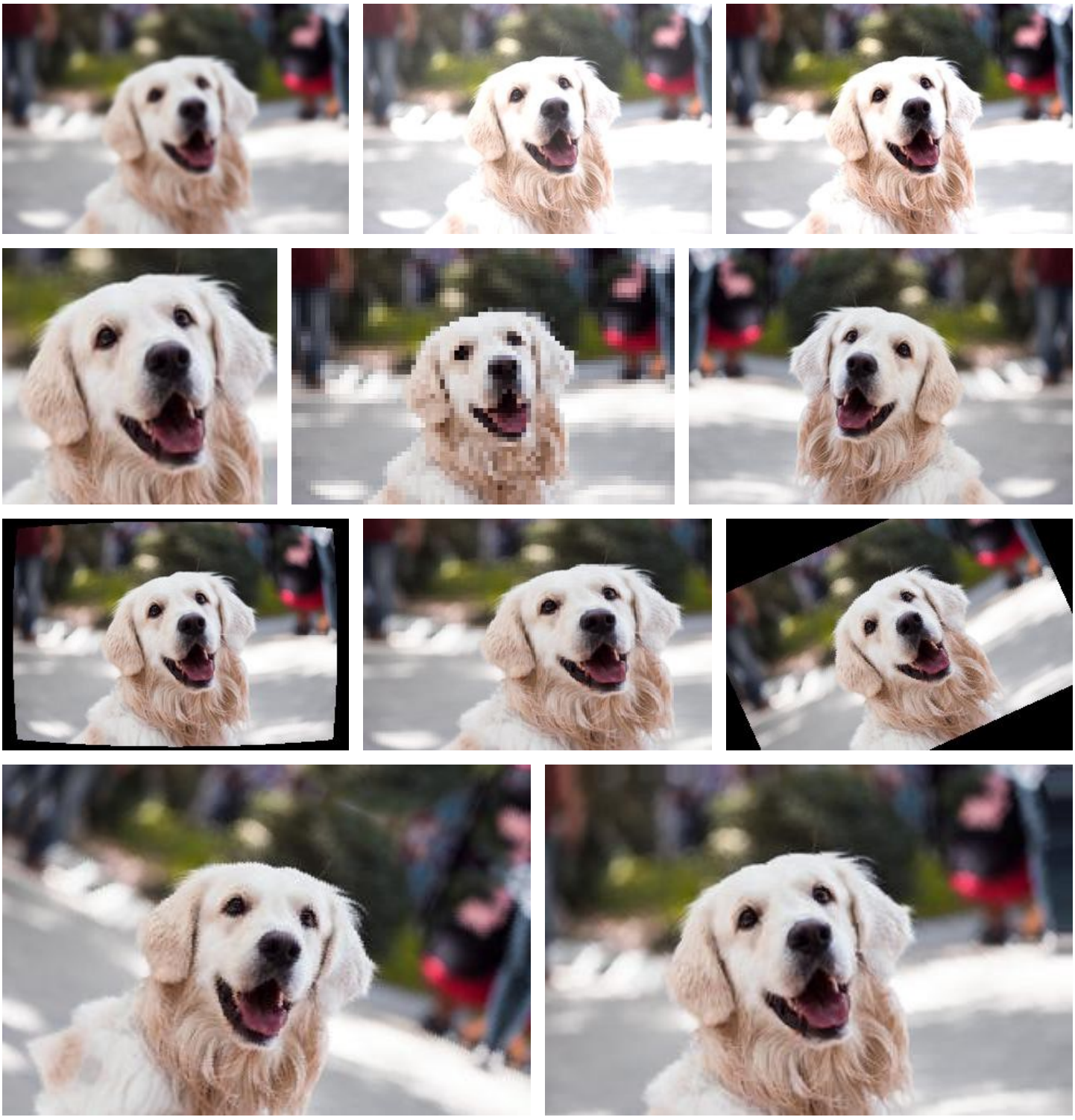
Dog image (Photo by Svetozar Milashevich from Pexels)

But it is always good to show your model as much training image as possible. The easiest way to get new samples without making new photos is to change the existing image a bit:

- Flip
- Rotate
- Scale and crop
- Add some noise or blur
- Change brightness and contrast
- Or alter color channels a little bit

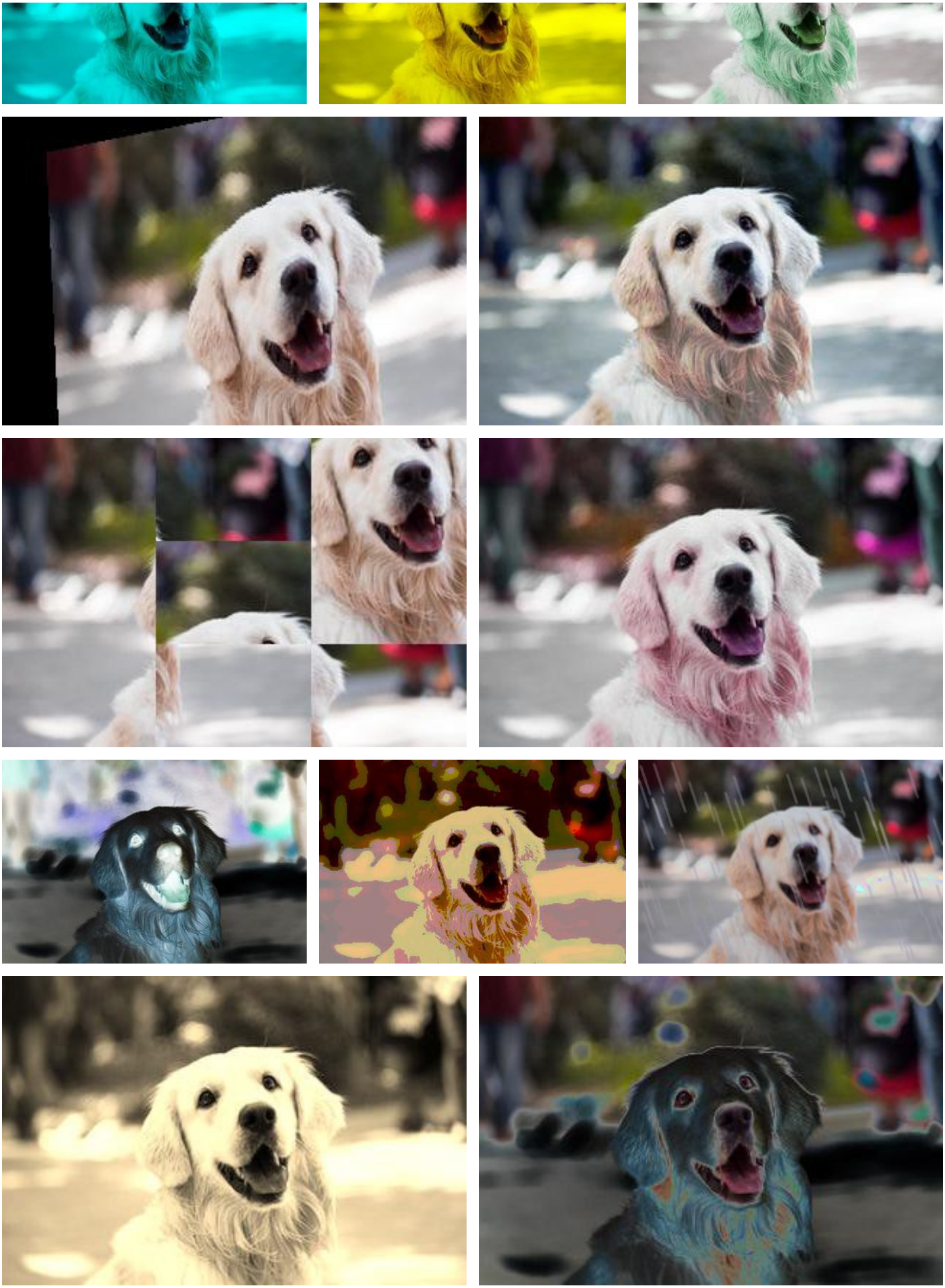
As a result, you will get a lot of new images that will still be images of the dog and will tell the model a piece of important information: "the rotated image of the dog is still an

image of the dog.”



Light transformations made using Albumentations library

Finally, you can apply some stronger transformations that in some cases also can be useful (as far as we can recognize the dog on these images):



Stronger transformations made using Albumentations library

. . .

How to apply this augmentation techniques to practice?

Nowadays, you don't need to make all these transforms by your hands; there are some libraries with ready to use implementations. And my favorite one is **Albumenatations.**

Albumenatations is a widely used image augmentations library created by Computer Vision specialists, Kaggle Masters and Grandmasters. You can find dozens of different transforms implemented in a very computationally efficient way.

You can start using it right now (if you don't use it yet) following the examples from the showcase notebook.

. . .

Which transformations are the best?

There are some standard sets of transformations that work quite well on most tasks (flips, rotations, resize), but there is no silver bullet, and every particular job has its optimal combination.

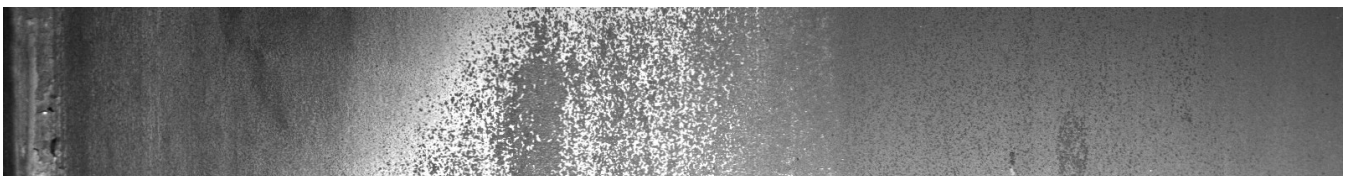
Let's look at the recent Kaggle competitions and see what augmentations were used by the winners.

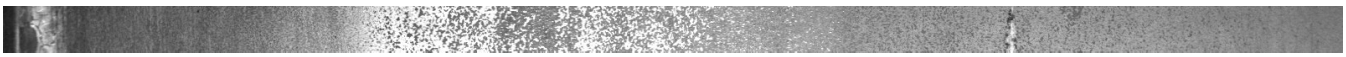
Note: below I list summary from the writeups of 1st place solutions, a lot of them have used Albumentations but not all. For consistency, I have used Albumentations names to describe transforms, where it is applicable.

. . .

Severstal: Steel Defect Detection

Task: find different types of defects on steel plates





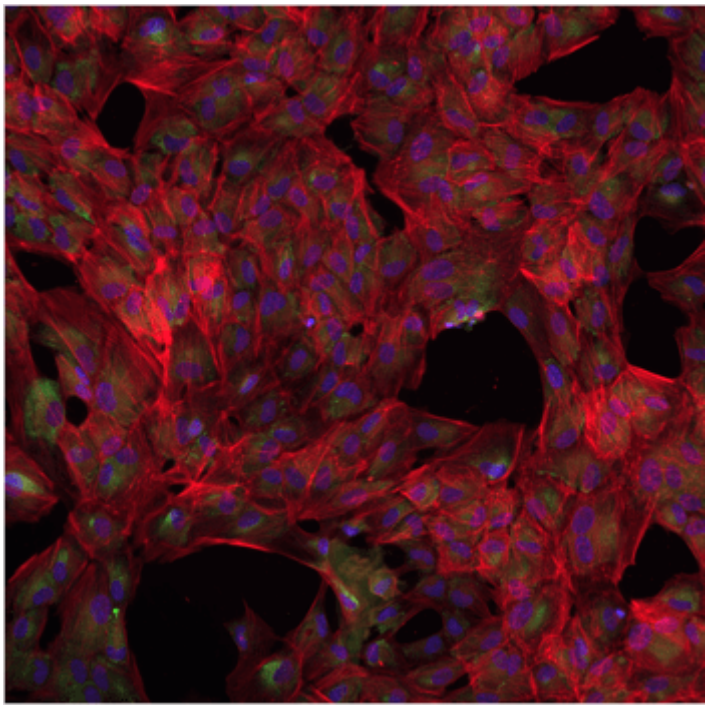
Source: https://www.kaggle.com/c/severstal-steel-defect-detection/data

Augmentations used by the winner :

- Randomcrop
- HorizontalFlip
- VerticalFlip
- RandomBrightnessContrast
- Customized defect blackout

. . .

Recursion Cellular Image Classification



Source: https://www.kaggle.com/c/recursion-cellular-image-classification/overview

Task: disentangle experimental noise from real biological signals on special 6-channels images

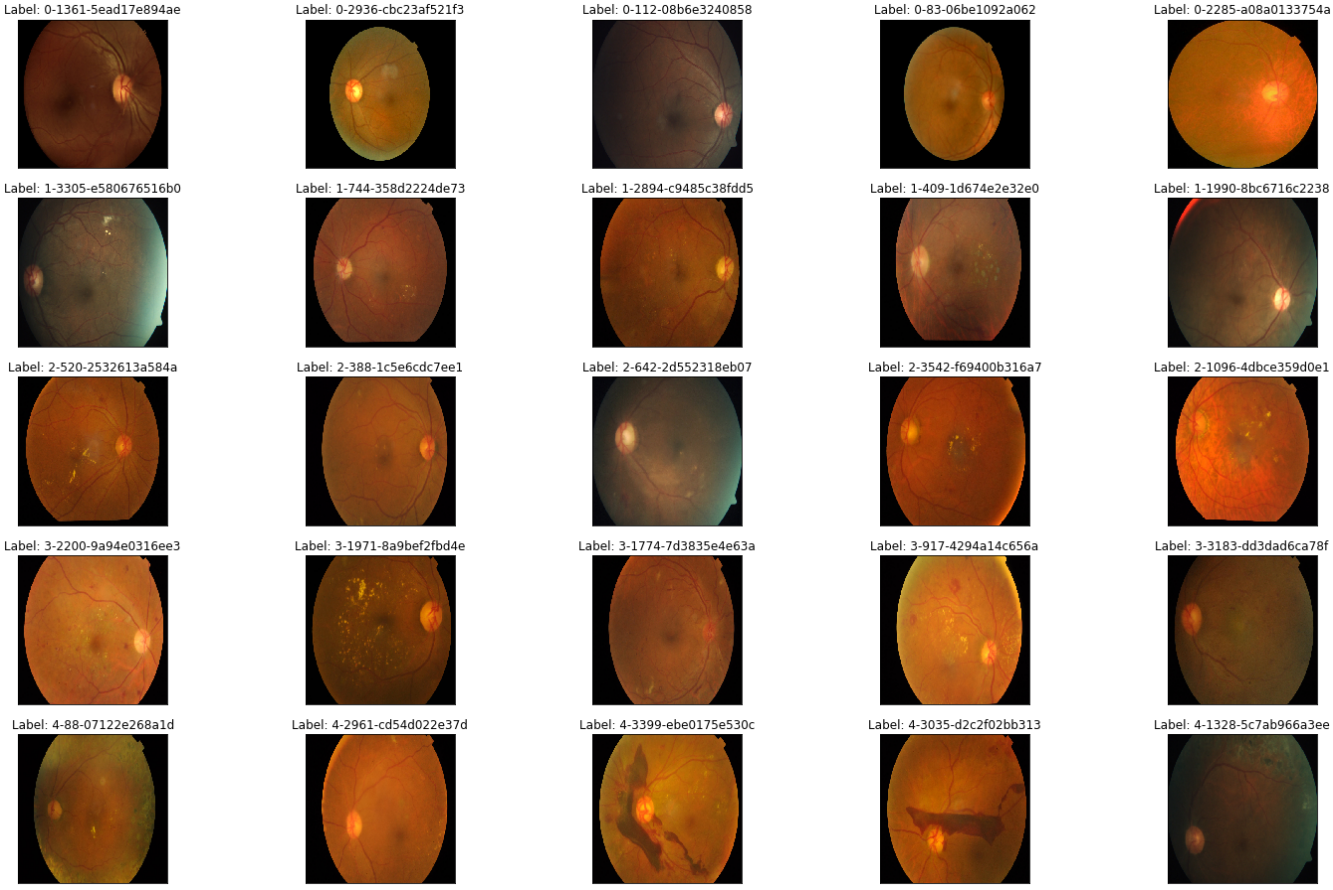
Augmentations used by the winner :

- Random resized crop
- HorizontalFlip
- VerticalFlip
- RandomRotate90
- Normalizing each image channel to N(0, 1)
- For each channel: channel = channel * a + b, where a ~ N(1, 0.1), b ~ N(0, 0.1)

. . .

APTOS 2019 Blindness Detection

Task: find diabetic retinopathy on retina images



Source: https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy

Augmentations used by the winner:

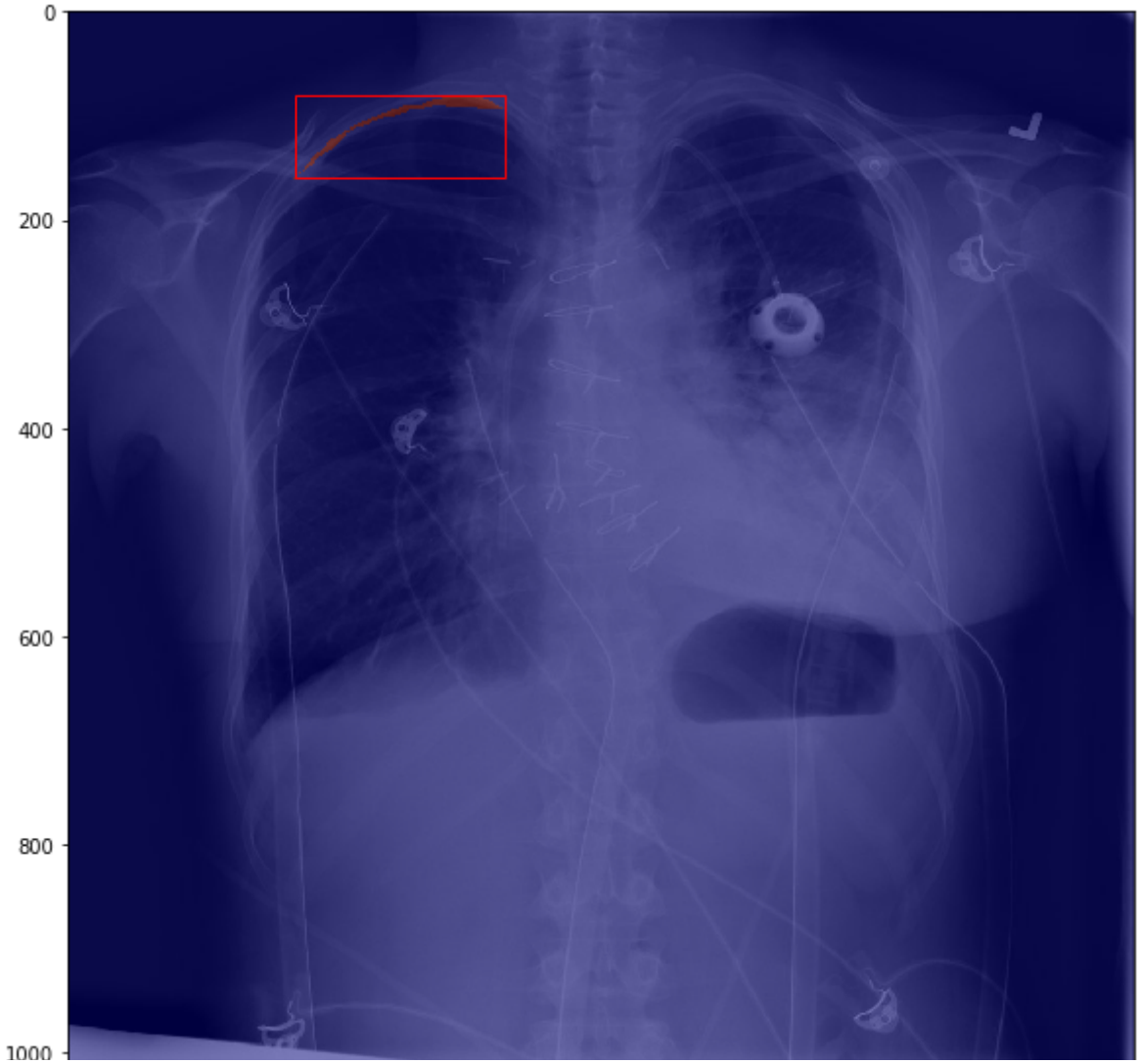
- RandomContrast

- Flip
- RandomScale
- RandomBrightness
- HueSaturationValue
- RandomRotation
- Blur
- ElasticTransform

. . .

SIIM-ACR Pneumothorax Segmentation

Task: identify Pneumothorax disease in chest x-rays



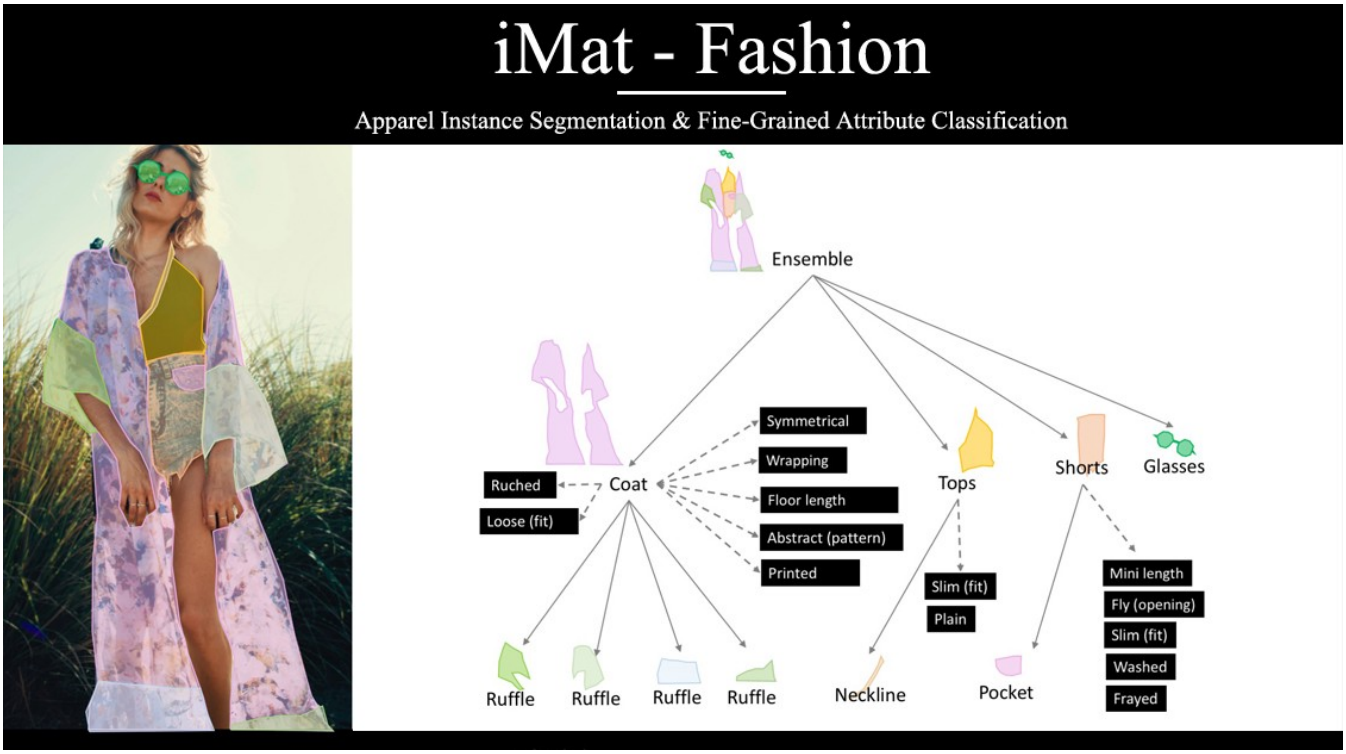
Source: https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/data

Augmentations used by the winner :

- HorizontalFlip
- RandomContrast
- RandomGamma
- RandomBrightness
- ElasticTransform
- GridDistortion
- OpticalDistortion
- ShiftScaleRotate

. . .

iMaterialist (Fashion) 2019 at FGVC6



Source: https://www.kaggle.com/c/imaterialist-fashion-2019-FGVC6/overview

Task: segment apparel and classify attributes

Augmentations used by the winner:

- Horizontal Flip
- RandomBrightnessContrast
- JpegCompression

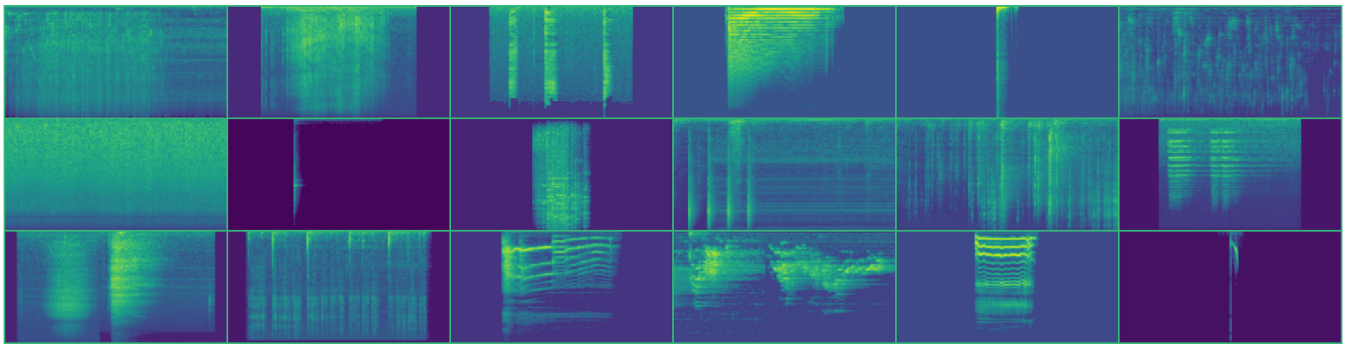
. . .

Bonus! Non-image competition

Computer vision models can be applied to other domains, and so do augmentations. For example, in audio processing, most of the SOTA algorithms convert raw audio into Mel-spectrograms and process them as normal images.

Freesound Audio Tagging 2019

Task: recognize sounds and apply tags of varying natures



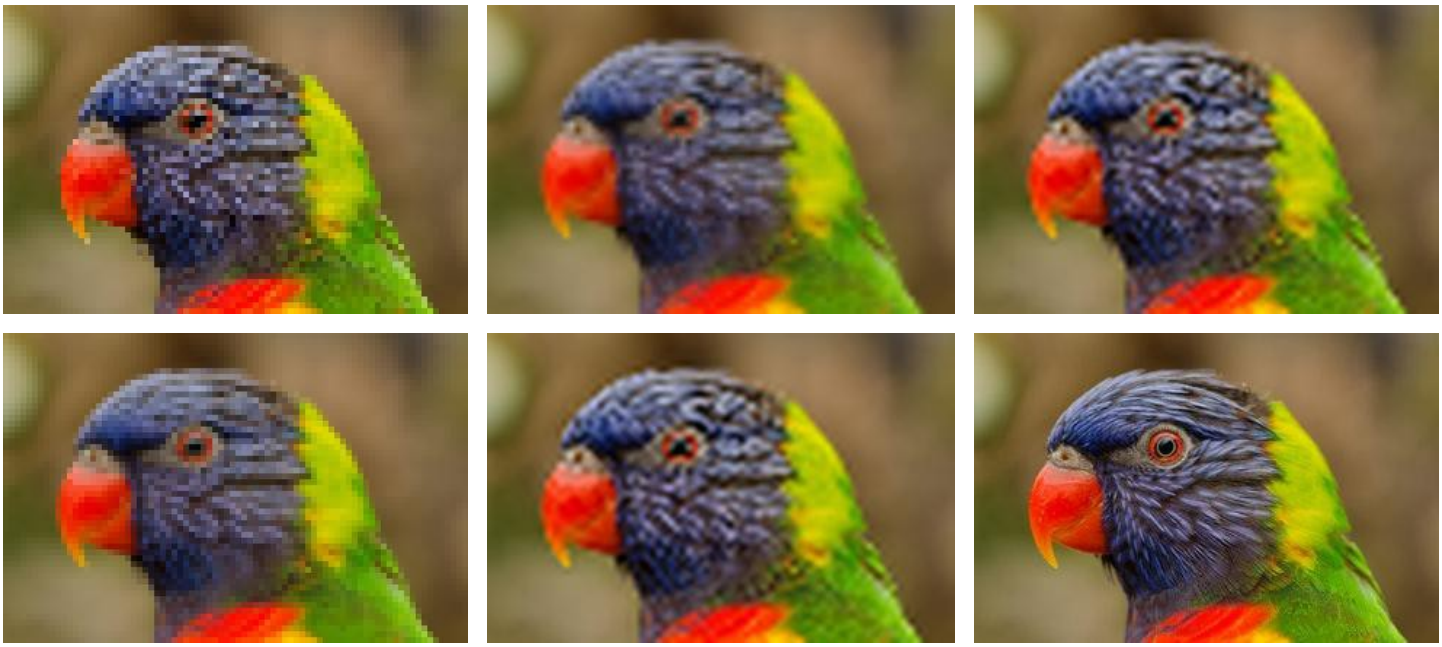
Source https://github.com/lRomul/argus-freesound

Augmentations used by the winner:

- RandomCrop
- RandomResizedCrop
- SpecAugment (~ Cutout for spectrograms)

. . .

As you can see in most of the examples, similar sets of basic transformations were used, but more complex ones are specific for each particular task. Moreover, not only transformations but also their parameters are important, look at these images of parrot.



Original photo by Skitterphoto from Pexels

All of them (except the original one) are the results of Downscale transformation with the same scale (0.25) but different interpolation methods — the parameter that doesn't seem to be very important at first glance. But the results are quite different.

. . .

So, how to chose transformations and tune their parameters?

Sometimes the logic behind choosing transforms is clear: when you work with images from a microscope you can easily rotate it 360 degrees, when you work with people's photos in the most cases rotation should be much more moderate. Sometimes unexpected things can work like RandomResizedCrop in the Freesound competitions as it contradicts the physical meaning of the spectrogram. Some cases are questionable from a common-sense standpoint and not obvious. For example, should we apply Horizontal flip to x-rays images and use the resulting photos with unusual sides of organs on it?

All these questions should be answered for each particular task independently. Even when you work with different datasets from the same domain, you can end up with different best sets of transformation.

There are some tries to automate this process. Just check out this paper about AutoAugment. But now it is mostly scientific exercises but not the production techniques. And the hard work should be done manually.

The best way to chose augmentation policy is by trials and error — the set of transformations and their parameters that leads you to the best metric values on the Cross-Validation or the Test set is the best one.

But there are rules of thumb on how to do it faster. The common augmentation search approach consists of 3–4 steps:

1. [Optionaly] train your model without augmentations to have a reliable baseline. It is useful for debugging, but sometimes step 2 can be used as a baseline as well.
2. Try some light transforms (shift, rotate, flip, crop, brightness, contrast, etc.) following common sense. If a result you get is very similar to original images in the dataset — you most probably can go with it.
3. Try to add some dataset-specific augmentations. For example, if you know that on the production stage, you will work with low-quality images — it worth trying some kind of Blur, Compression, or Noise. If you work with street photos and have a lot of pictures with different weather conditions, try experimenting with RandomFog, Show, or Rain. But don't rely on the visual assessment; compare the metrics of your model before and after using these augmentations. If they become better on a CV — keep this transformation otherwise drop it.
4. Try some strong transformations. Consider them as some hyperparameters you want to tune. Transforms as Cutout, Solarize, or ChannelDropout can produce very unusual non-natural images, but sometimes, it can help you to train a better model. Now, you should solely rely on your CV metrics — test some combinations and find the best one. At the same time, you can also try more aggressive params of the transforms from steps 2 and 3.

Finally, you will come up with the best combination of transforms.

The final choice is always made based on metrics, but your experience, intuition, and common sense can help to find the solution faster.

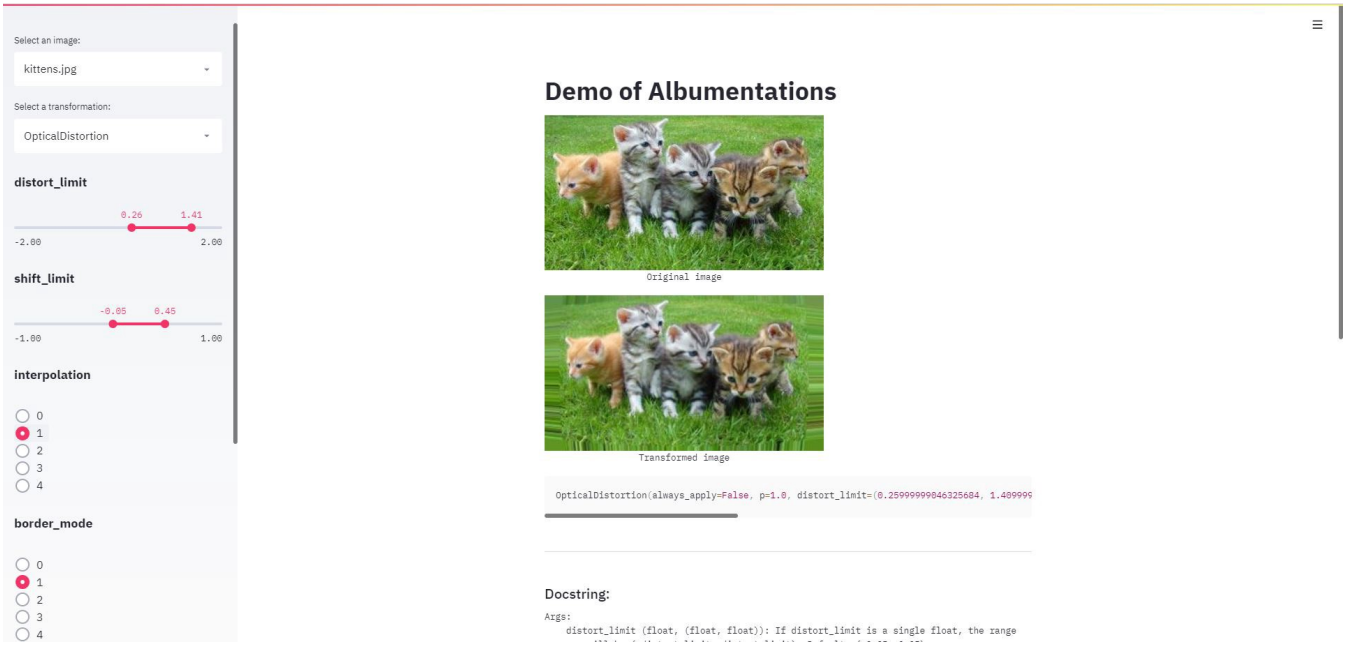
. . .

But how can I get an intuitive understanding of how different transformations work?

Some transforms are intuitively clear, for example, Crop “crops the central part of the input.” And it has only two parameters “ height” and “width” of the crop.

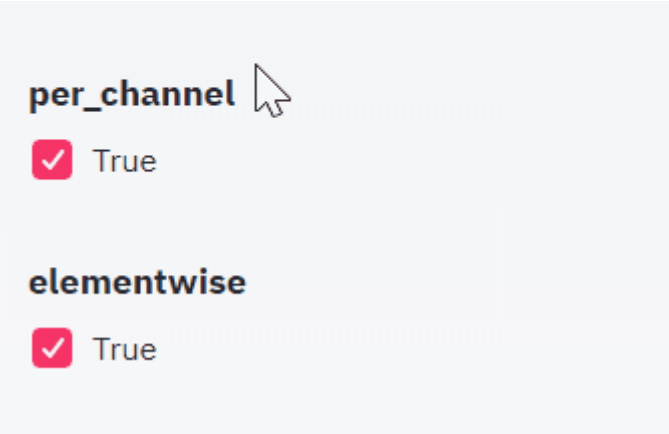
But the complex ones like RandomRain are much less intuitive and may have up to 10 different params. It is tough to imagine the result of altering them in your head.

That is why I have created the service for experimenting with different transformations, online version of it is available here **https://albumentations-demo.herokuapp.com/**



Screenshot of the service

I have used the excellent Steamlit framework to create a convenient interface.



Params of the MultiplicativeNoise transform

You can choose any transformation you want and tweak the parameters with visual control elements to see the resulting transformed image.

For convenience, you can also see the code to call the chosen transformation with selected parameters — copy-paste it into your Python script and use it. You will also see the original docstring from the source code of this transformation, where you can find explanations and details about this transform.

```
MultiplicativeNoise(always_apply=False, p=1.0, multiplier=(0.9, 1.1), per_channel=True, elementwise=True)
```

Docstring:

```
Multiply image to random number or array of numbers.

    Args:
        multiplier (float or tuple of floats): If single float image will be multiplied to this number.
            If tuple of float multiplier will be in range `[multiplier[0], multiplier[1])`. Default: (0.9, 1.1).
        per_channel (bool): If `False`, same values for all channels will be used.
            If `True` use sample values for each channels. Default False.
        elementwise (bool): If `False` multiply multiply all pixels in an image with a random value sampled once.
            If `True` Multiply image pixels with values that are pixelwise randomly sampled. Defaule: False.

    Targets:
        image

    Image types:
        Any
```

The code to call the transformation and the docstring

Playing with parameters and understanding of how they work on different images is the best way to get the intuition you need for the fast augmentations tunning.

. . .

Can I use my images in this service?

Yes, you can do it if you run the service locally.

```
git clone https://github.com/IliaLarchenko/albumentations-demo
cd albumentations-demo
pip install -r requirements.txt
streamlit run src/app.py -- --image_folder PATH_TO_YOUR_IMAGE_FOLDER
```

The source code of this service is available on GitHub (https://github.com/IliaLarchenko/albumentations-demo) you can clone it and do whatever you want.

While the online version is useful when you want to explore all the transformation available or just quickly check how they work, the local one is more flexible and can be customized for your needs.

. . .

I hope this tool and the article will be useful for you! Don't hesitate to ask me anything or propose any improvements. Good luck!

Computer Vision Augmentation Data Science Image Processing Interfaces

About Help Legal