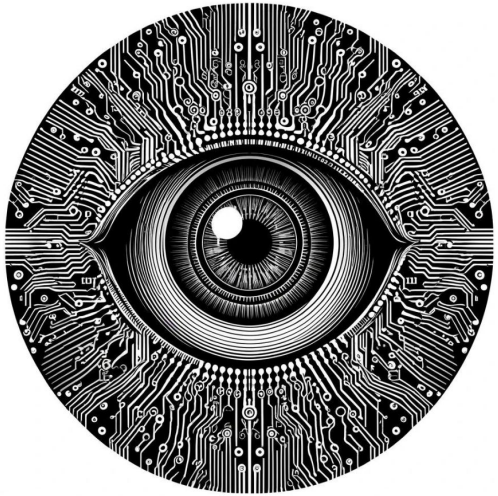
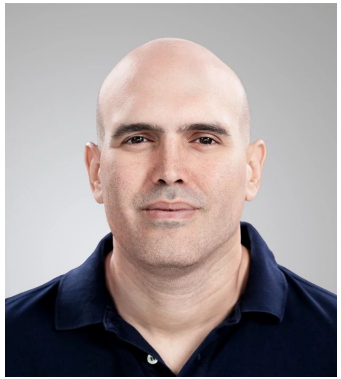
Workshop 9 - Regularization with Data Augmentation



Antonio Rueda-Toicen

About me

- AI Researcher at Hasso Plattner Institute, AI Engineer & DevRel for Voxel51
- Organizer of the Berlin Computer Vision Group
- Instructor at Nvidia's Deep Learning Institute and Berlin's Data Science Retreat
- Preparing a MOOC for OpenHPI (now open for registration)



LinkedIn

How to use our Discord channel during the workshop

- Our channel is **#practical-computer-vision-workshops.** Please ask all questions there instead of the Zoom chat. Through Discord we can have better and more detailed discussions.

- **Step 1 - Use the Discord invite on the Voxel51 website** https://discord.com/invite/fiftyone-community
- **Step 2 - Access channel** (use the direct link below or search) http://bit.ly/3YmvPXG

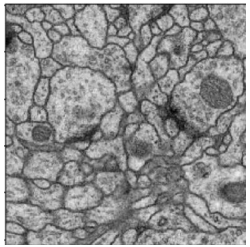
Agenda

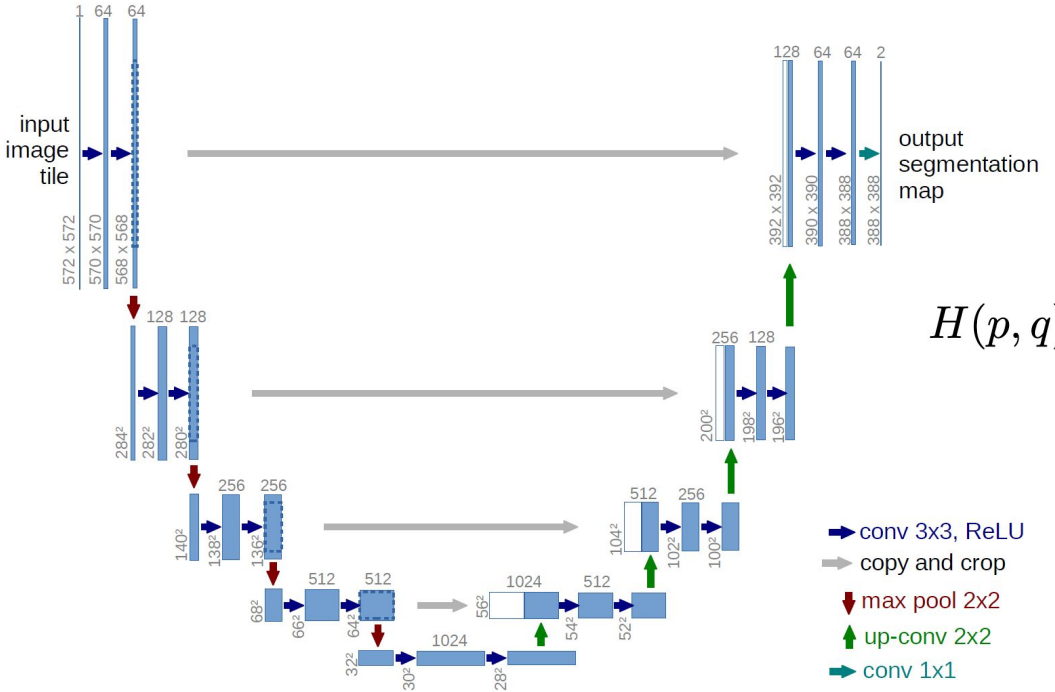
- Upsampling and channel mixing with convolutions (review)
- Image data augmentation

Notebooks

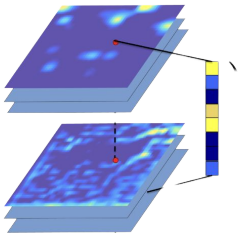
- Semantic segmentation with U-net (***now with augmentations! :D**)
  - Implementation with albumentations
  - Implementation with PyTorch's transforms.v2

Semantic segmentation with U-Net





$$H(p,q) = -\sum_i p(i) \log q(i)$$



[U-Net: Convolutional Networks for Biomedical Image Segmentation]

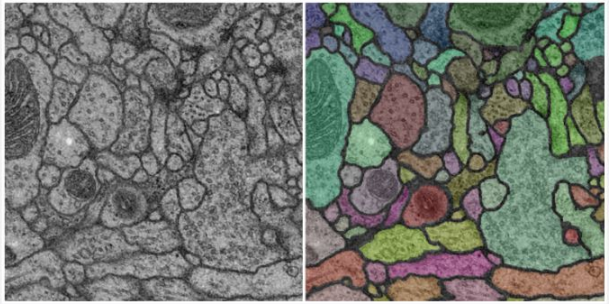
Segmentation of neuronal structures in EM stacks challenge - ISBI 2012

The ISBI 2012 challenge server has been decommissioned, but you can still download the training and test data!

The IEEE International Symposium on Biomedical Imaging (ISBI) is the premier forum for the presentation of technological advances in theoretical and applied biomedical imaging and image computing. In the context of the ISBI 2012 conference which was held in Barcelona, Spain, from 2 to 5 May 2012, we organized a challenge workshop on **segmentation of neuronal structures in EM stacks**.

In this **challenge**, a full stack of EM slices will be used to train machine learning algorithms for the purpose of **automatic segmentation of neural structures**.

The images are representative of actual images in the real-world, containing some noise and small image alignment errors. None of these problems led to any difficulties in the manual labeling of each element in the image stack by an expert human neuroanatomist. The aim of the challenge is to compare and rank the different competing methods based on their pixel and object classification accuracy.



Example of ssTEM image and its corresponding segmentation

Relevant dates

- **Deadline for submitting results**: ~~February 1st~~ March 1st, 2012
- **Notification of the evaluation**: ~~February 21st~~ March 2nd, 2012

Semantic, Instance, and Panoptic Segmentation

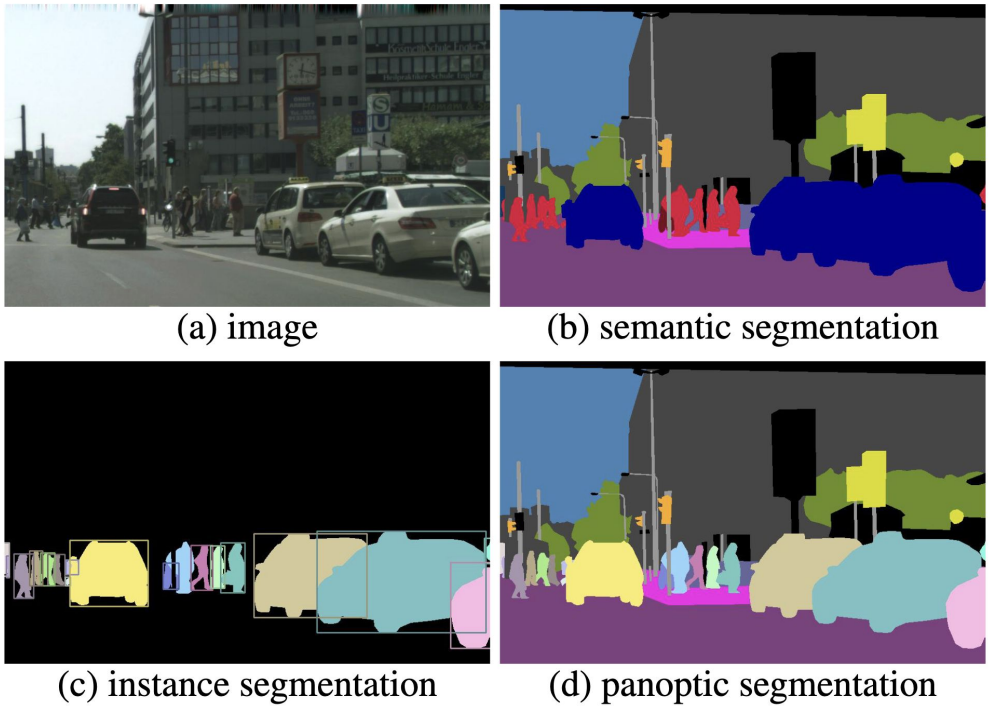
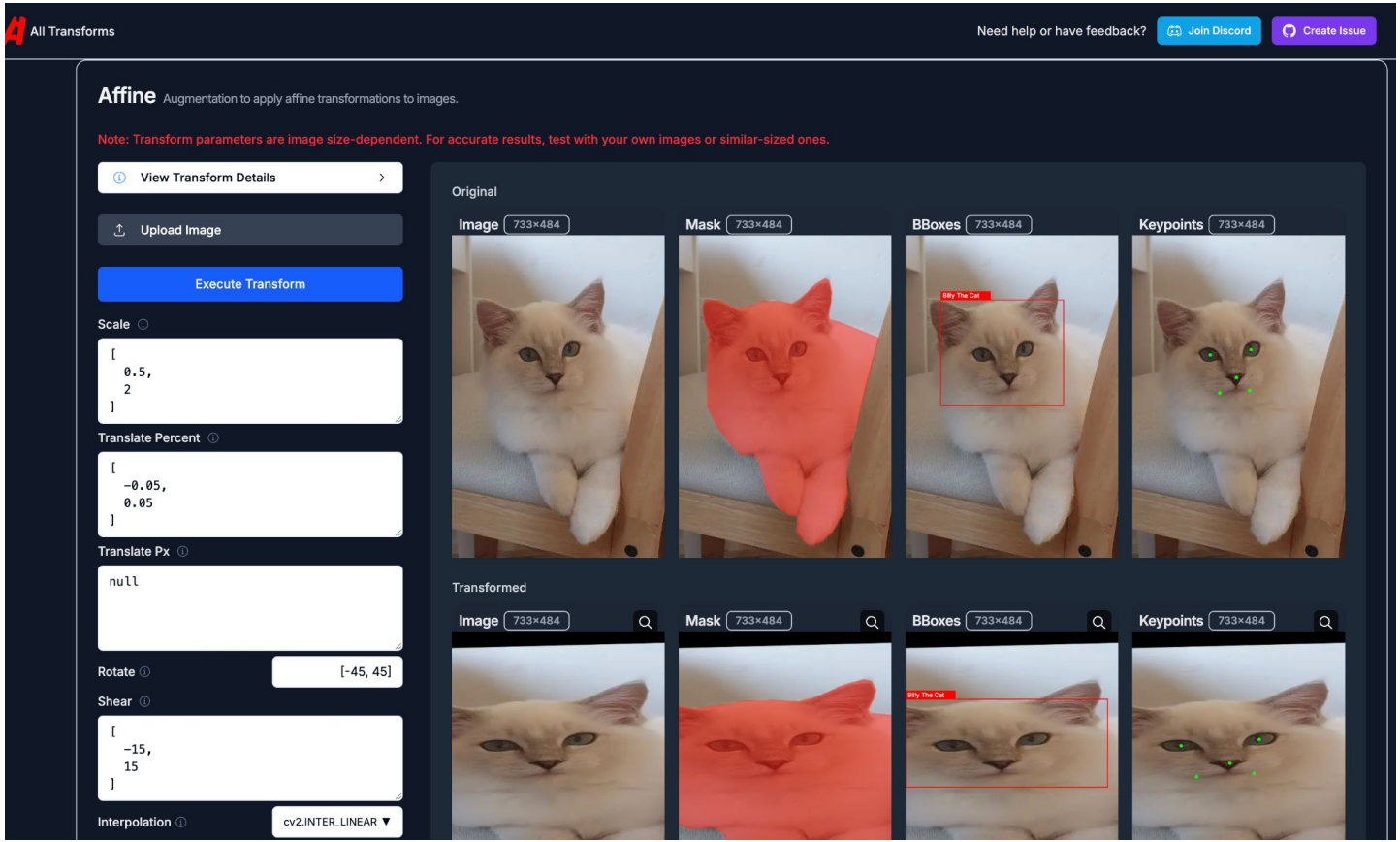
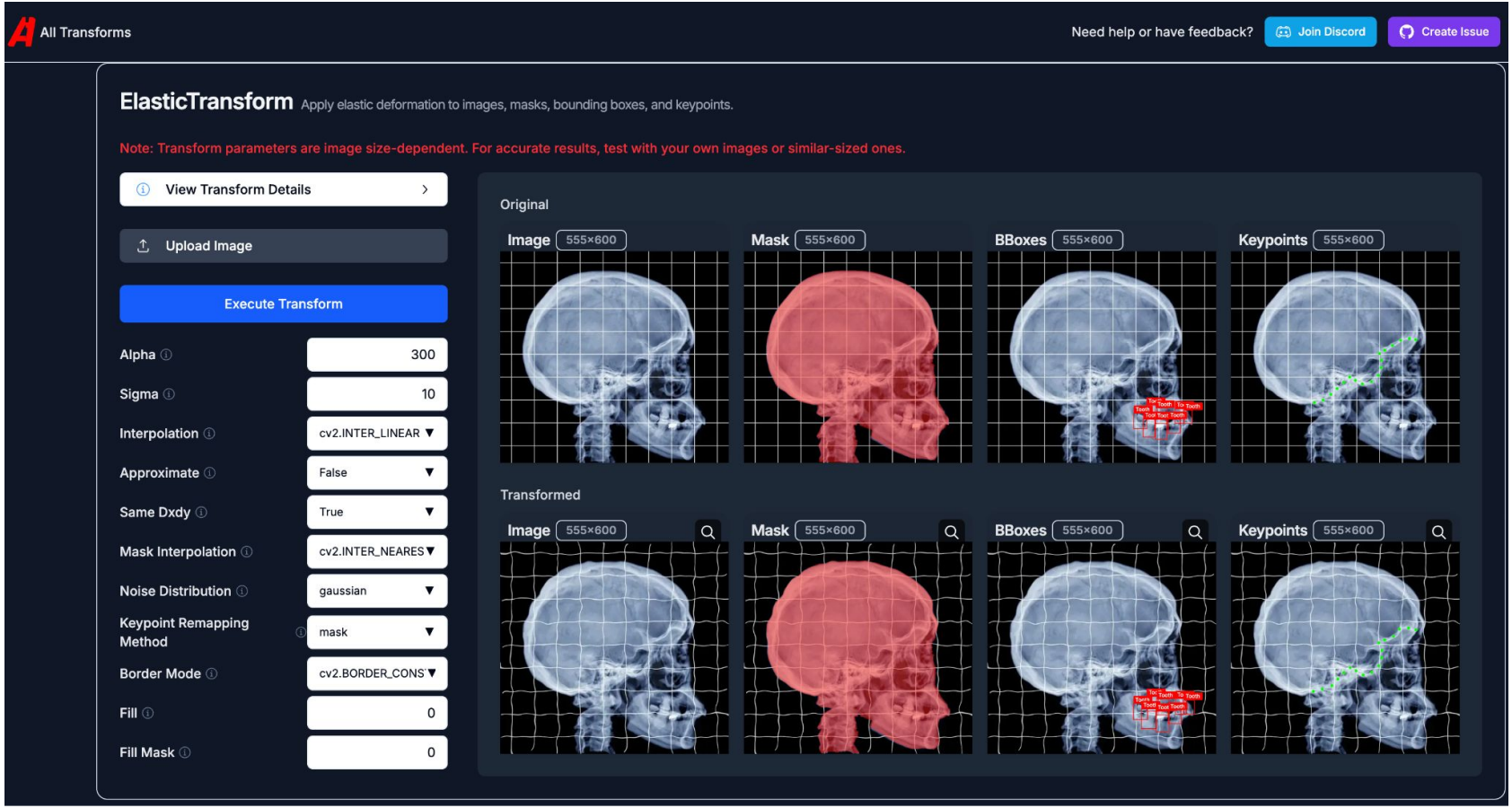


Image from Panoptic Segmentation

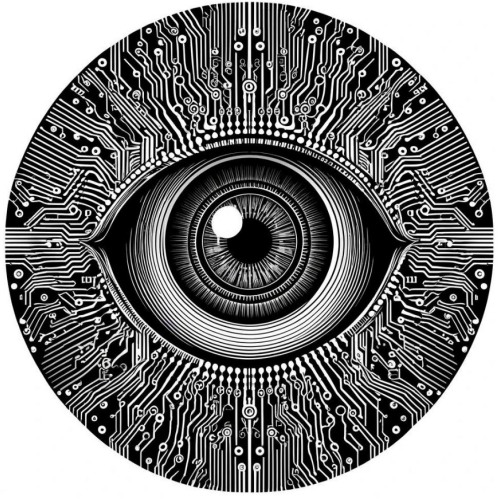


https://explore.albumentations.ai/transform/Affine



https://explore.albumentations.ai/transform/ElasticTransform

Upsampling and Channel Mixing with Convolutions

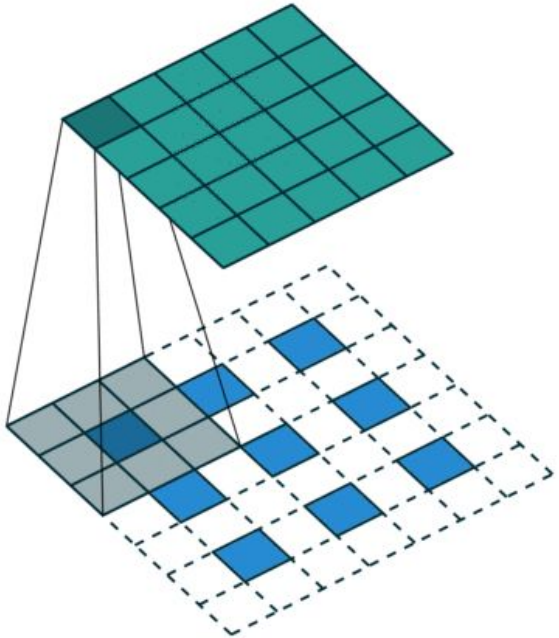


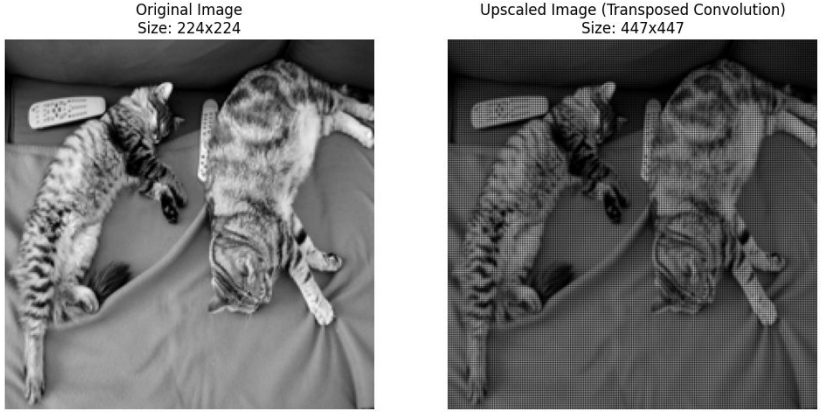
Antonio Rueda-Toicen

Learning goals

- Master upsampling and channel mixing with convolutions
- Use bilinear interpolation to resize images
- Understand hypercolumns and feature combination for image generation

Upsampling filters (aka transposed convolution / upconvolution / atrous convolution / deconvolution)



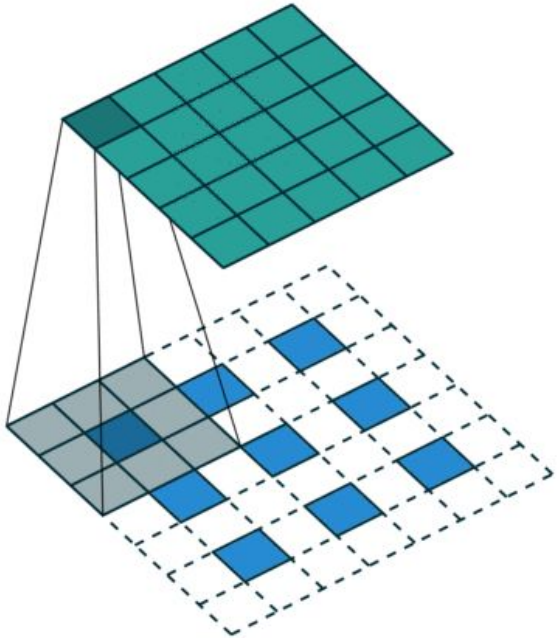


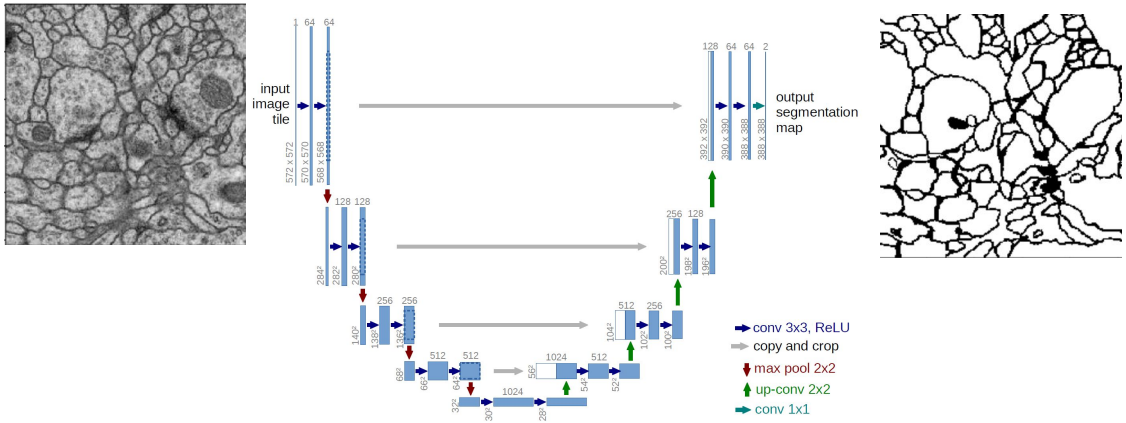
Upconvolution of 3x3 and stride = 1 padded with zeros

Effect: the output is a 5x5 image with ‘synthetic’ pixels

Activity: Conv2D in Pytorch (Colab notebook)

Usage of ‘up-convolution’: decoder paths in image generation models





Upconvolution of 3x3 and stride = 1 padded with zeros

Effect: the output is a 5x5 image with ‘synthetic’ pixels

Activity: Conv2D in Pytorch (Colab notebook)

The checkerboard artifact from “deconvolution”

Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

All layers use resize-convolution.
No artifacts.

Image from [Deconvolution and Checkerboard artifacts]

Original Image
Size: 224x224

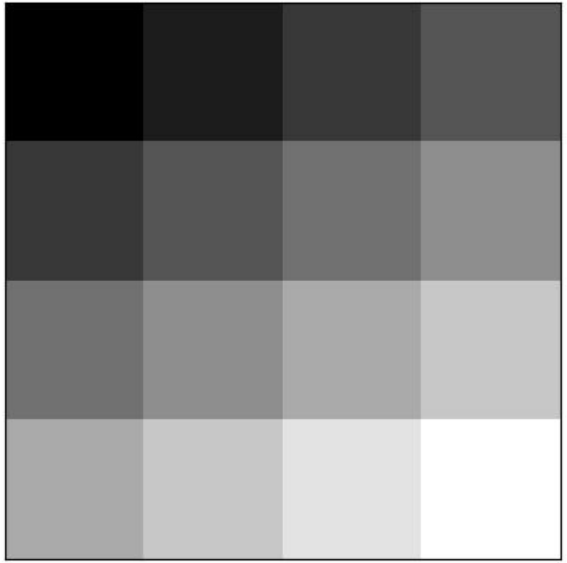
Upscaled Image (Transposed Convolution)
Size: 447x447

Problem: “checkerboard” artifacts

Bilinear interpolation

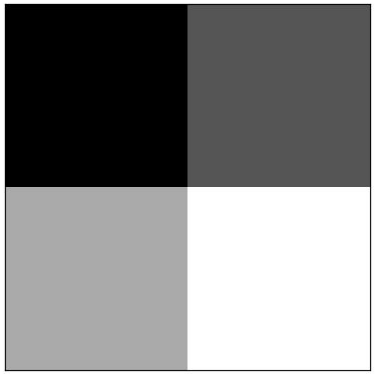
$$w_x = \frac{x - x_1}{x_2 - x_1}$$

output



$$w_y = \frac{y - y_1}{y_2 - y_1}$$

input



$(x_1, y_1) = (0, 0)$ coordinates of Q_{11}

$(x_2, y_2) = (1, 1)$ coordinates of Q_{22}

$(x, y) = (0.6, 0.7)$ target point

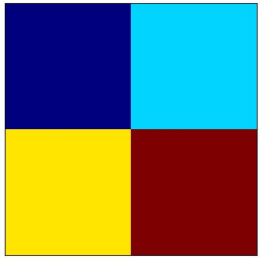
$$Q = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

$$f(x, y) = \begin{bmatrix} 1 - w_x & w_x \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} 1 - w_y \\ w_y \end{bmatrix}$$

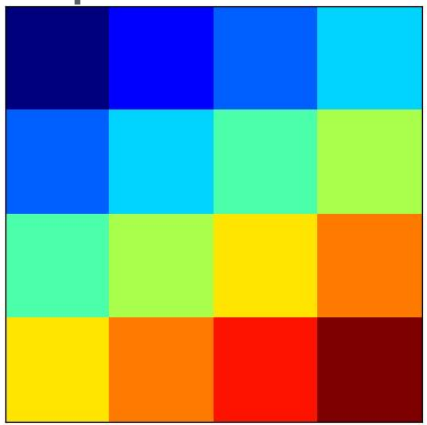
Bilinear interpolation in PyTorch

```
import torch
import torch.nn.functional as F

# Create a 1x1x2x2 tensor (batch_size x channels x height x width)
# Notice that the batch size and channel dimensions are created by wrapping
# the height and width tensor with two pairs of extra square brackets
input = torch.tensor([[[[10, 20],
                        [30, 40]]]],
                     dtype=torch.float32)

# Upscale to 4x4
output = F.interpolate(input, size=(4, 4), mode='bilinear',
align_corners=True)

import matplotlib.pyplot as plt
# The colormap is just for illustration of corner alignment
plt.imshow(input.squeeze(), cmap="jet")
plt.imshow(output.squeeze(), cmap="jet")
```

input (with colormap)
shape: 2x2



output (with colormap)
shape: 4x4



1x1 convolutions mix image channels

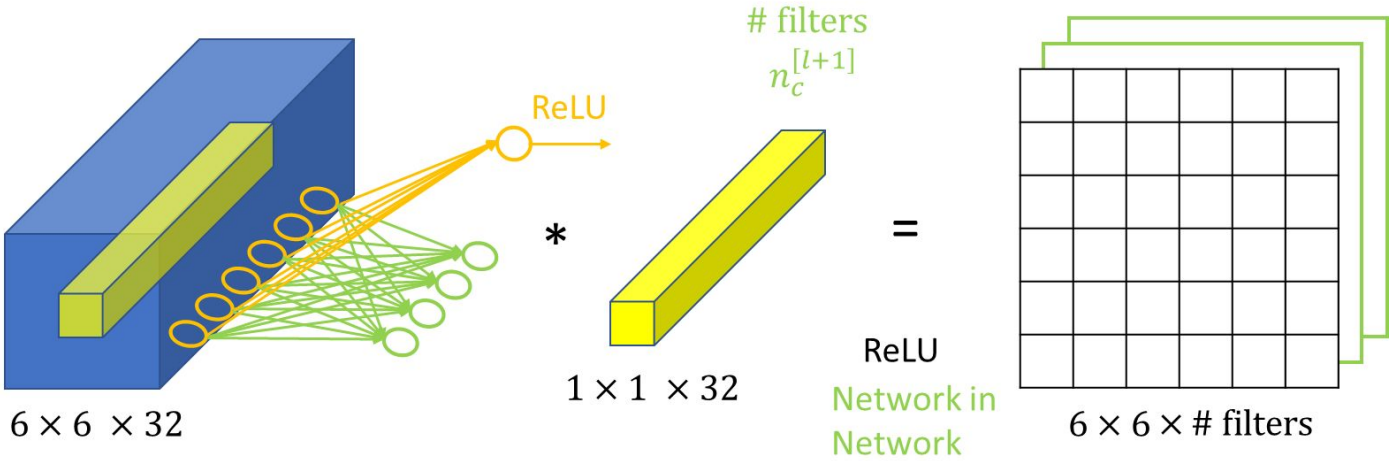


Image from "What does a 1x1 convolution do?" by Andrew Ng

Upsampling and channel mixing with 1x1 convs

Original Image
Size: 224x224

Sample Convolution Outputs (4 of 64)
Size: 224x224

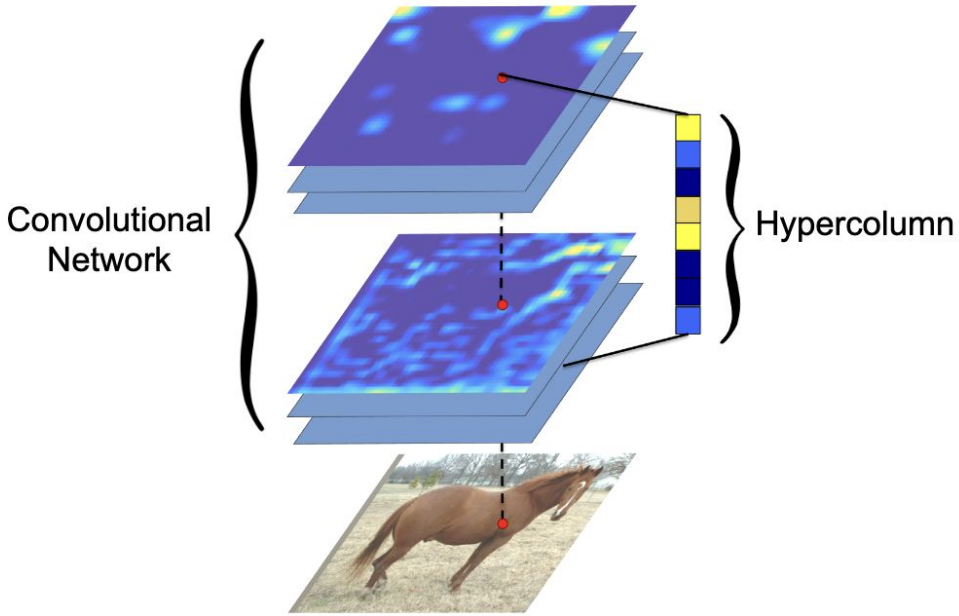
Sample Upsampled Outputs
Size: 448x448

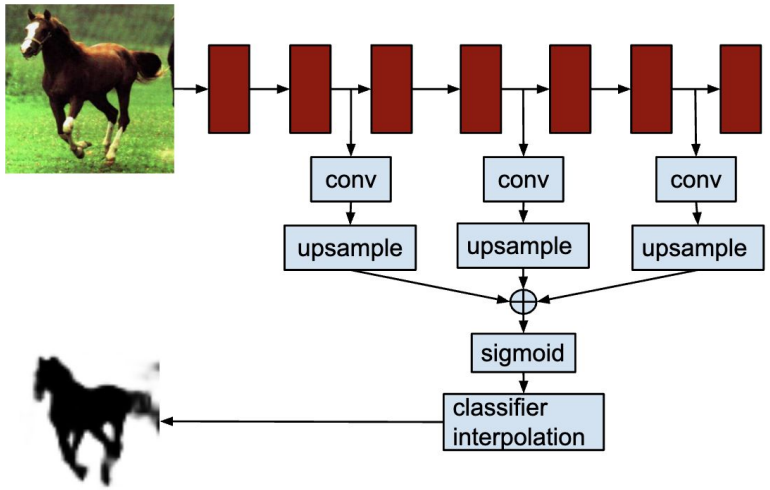
Final Output (After 1x1 Conv)
Size: 448x448

```
layers = nn.Sequential(
    nn.Conv2d(
        in_channels=1,
        out_channels=64,
        kernel_size=3,
        padding=1,
        bias=False
    ),
    nn.Upsample(
        scale_factor=2,
        mode='bilinear',
        align_corners=True
    ),
    nn.Conv2d(
        in_channels=64,
        out_channels=1,
        kernel_size=1,
        bias=False
    )
)
```

Note: no checkerboard artifacts

Combining “hypercolumns” is a common use of 1x1 convolutions





Images from “Hypercolumns for Object Segmentation and Fine-grained Localization”

Summary

1x1 convolutions mix channel information

- Mixing channels allows us to create new images by combining learned features

Transposed convolutions enable upsampling (with checkerboard artifacts)

- Bilinear interpolation and channel mixing is a better alternative

Further reading and references

A guide to convolution arithmetic for deep learning

- https://arxiv.org/abs/1603.07285

Network in network (1x1 convolutions)

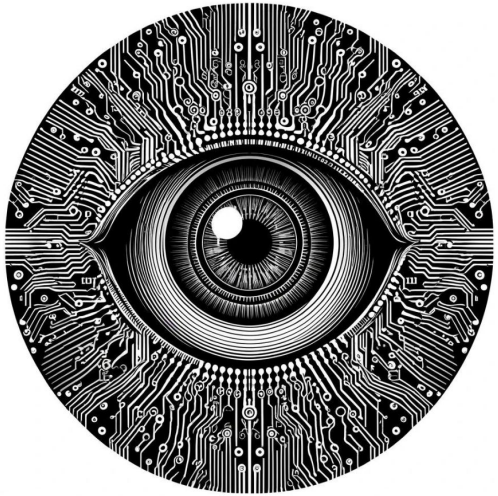
- https://arxiv.org/abs/1312.4400

Hypercolumns for object segmentation and fine-grained localization

- https://openaccess.thecvf.com/content_cvpr_2015/papers/Hariharan_Hypercolumns_for_Object_2015_CVPR_paper.pdf

KI Service Zentrum by Hasso-Plattner-Institut HPI

Image Data Augmentation



Antonio Rueda-Toicen

SPONSORED BY THE
Federal Ministry of Education and Research

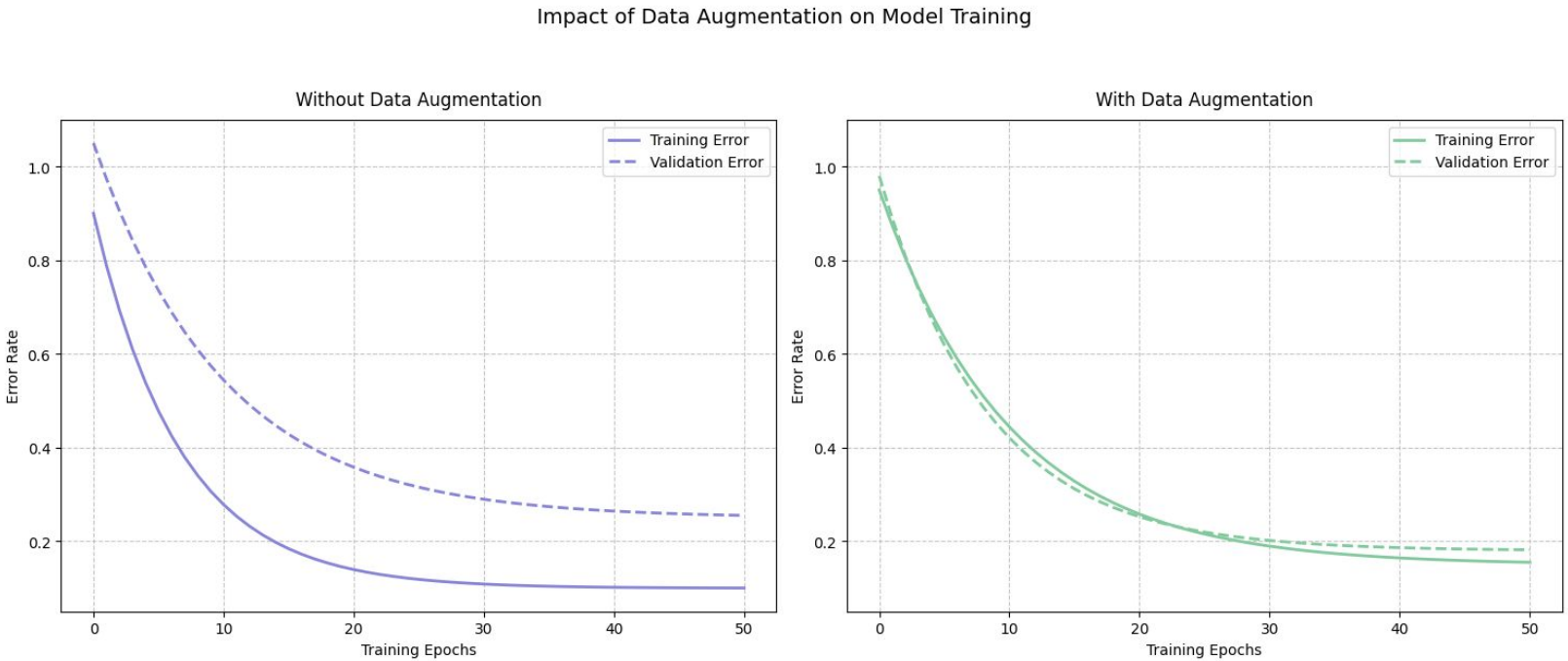
Learning goals

- Use data augmentation to extend training datasets
- Gain familiarity with PyTorch’s transforms.v2: affine transformations, vertical and horizontal flipping, and random crops

Data augmentation

Image from https://docs.fast.ai/vision.augment

Data augmentation as regularization

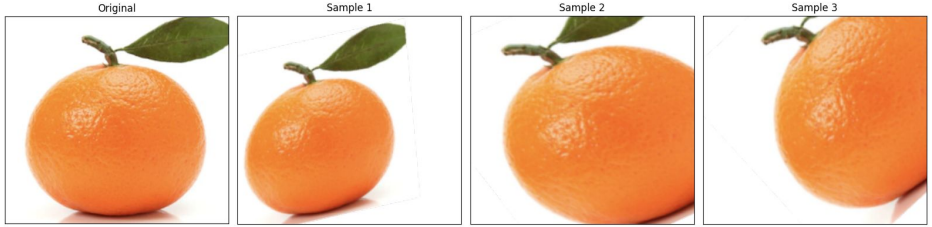


Affine transformations

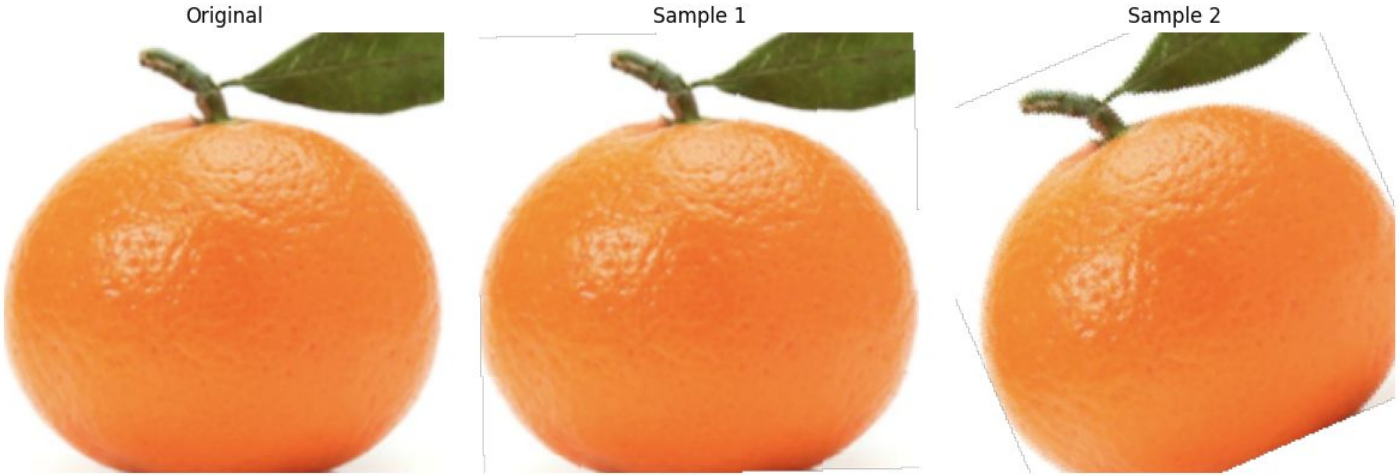
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta_x \\ \Delta_y \end{pmatrix}$$

Linear transformation
(Rotation, scaling, shear)

Translation
(Moving along the axes)



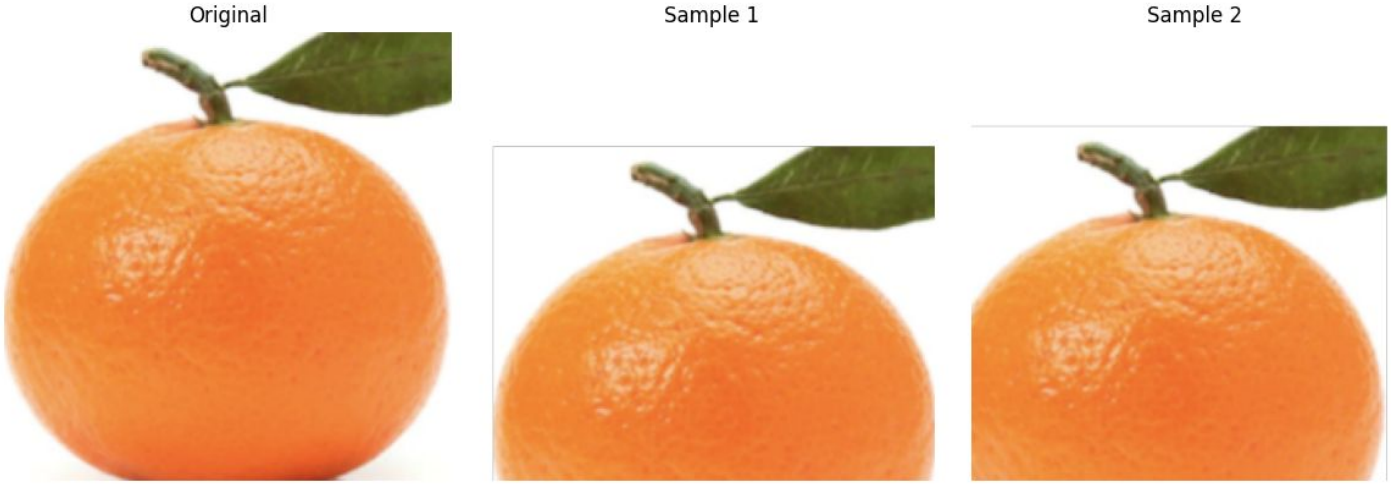
The RandomAffine transformation - rotation



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

```
transforms.RandomAffine(degrees=45)
```

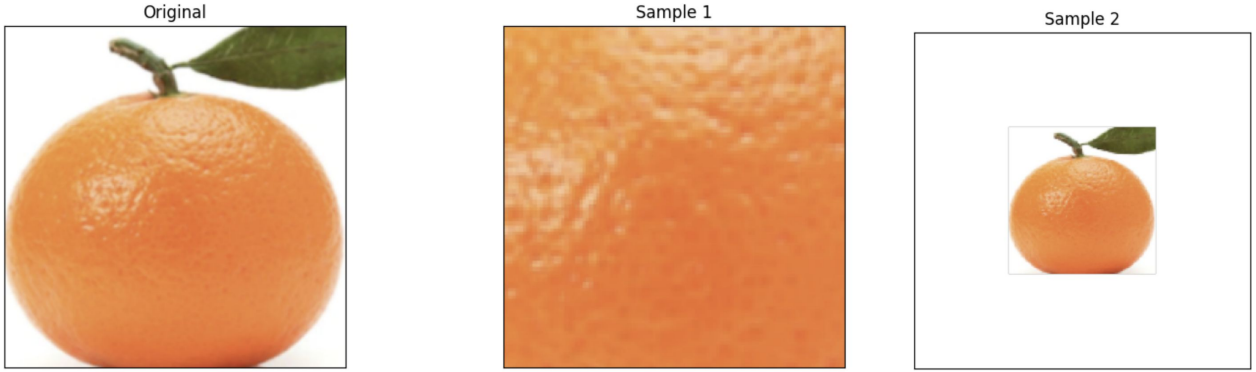
The RandomAffine transformation - translation



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

```
transforms.RandomAffine(degrees=0, translate=(0.4, 0.4))
```

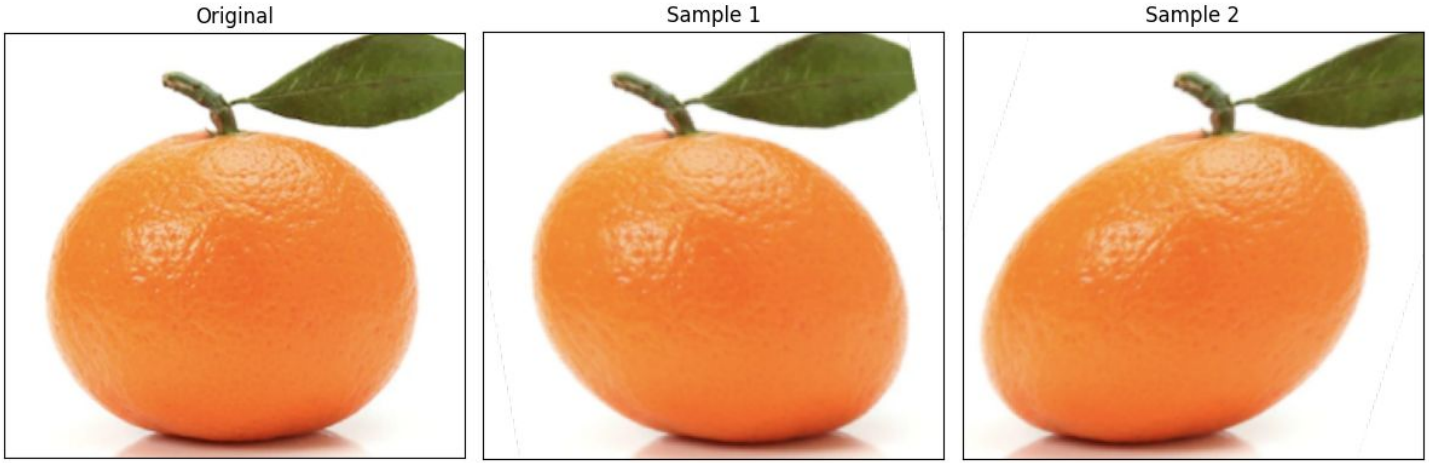
The RandomAffine transformation - scaling



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

```
transforms.RandomAffine(degrees=0, scale=(0.25, 2.5))
```

The RandomAffine transformation - shear



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & s_x \\ s_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

```
transforms.RandomAffine(degrees=0,shear=45)
```

Horizontal and vertical flipping

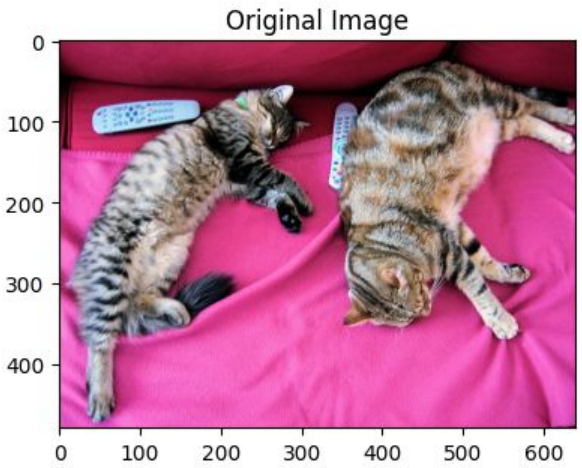
Original

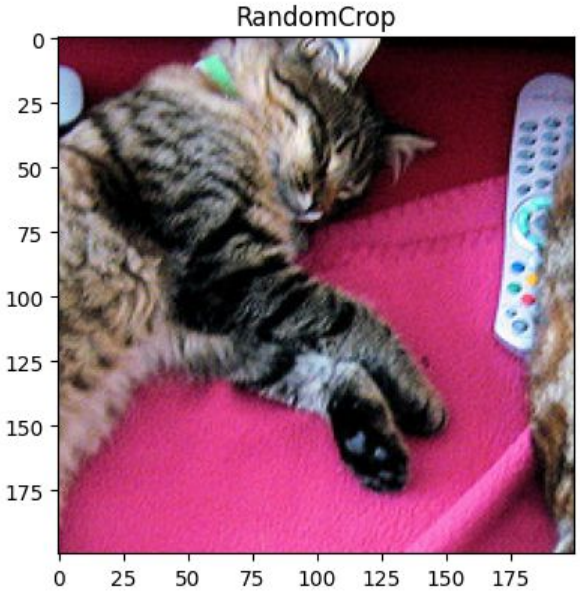
Sample 1

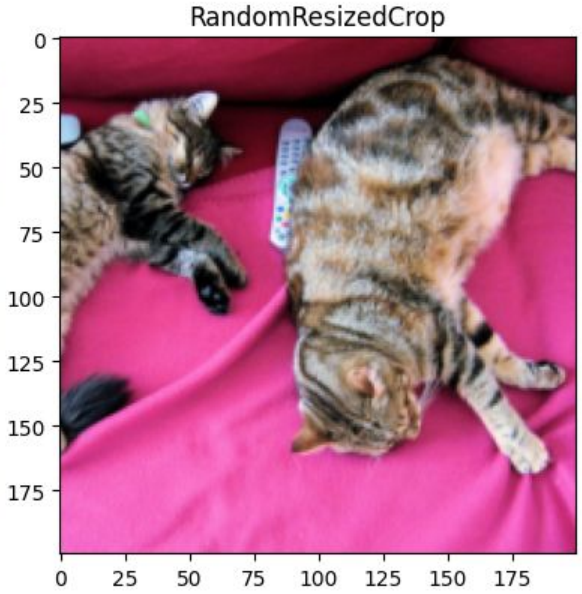
Sample 2

```
transforms.Compose([
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomVerticalFlip(p=0.5)
    ])
```

RandomCrop and RandomResizedCrop








```
transforms.RandomCrop(size=224)
```

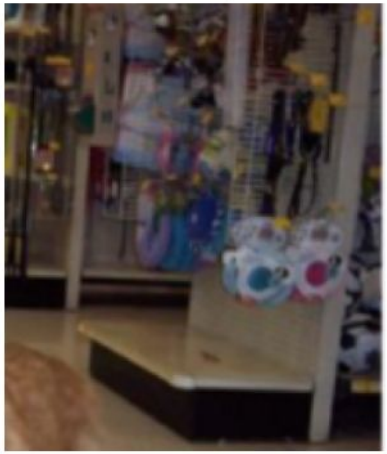
```
transforms.RandomResizedCrop(
        size=224,
        scale=(0.08, 1.0),  # Range of size of crop
        ratio=(3/4, 4/3)    # Range of aspect ratio
    )
```

Blind data augmentation can damage your model

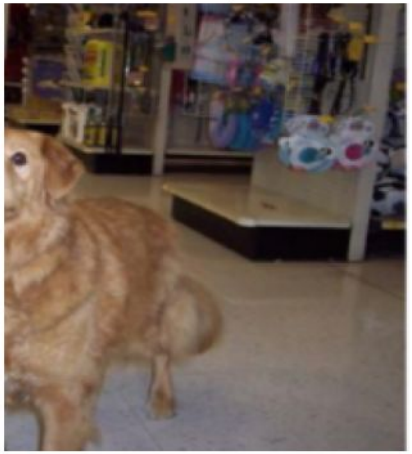
$\hat{y} = \text{golden retriever (unit 207 on Imagenet)}$

Original

Sample 1



Sample 2



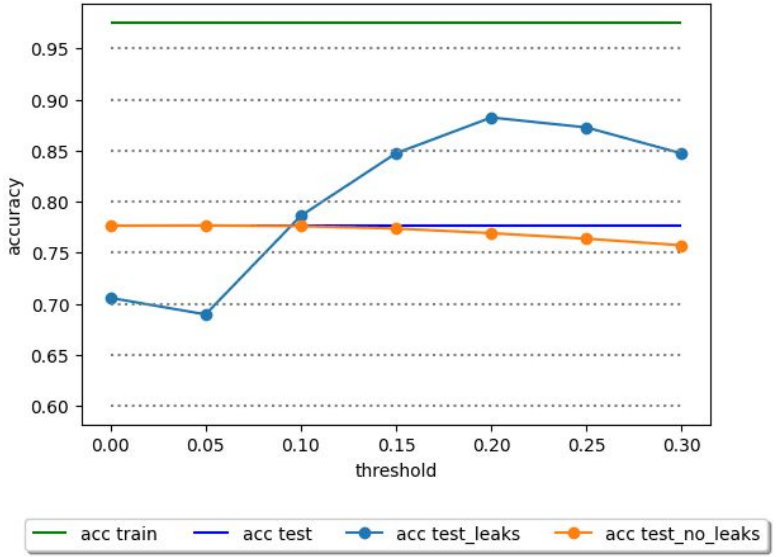
$$\text{Cross Entropy Loss} = -\log(0.003) = 5.809$$

$$-\log(0.6) = 0.511$$

The validation set should not be augmented during training


```python
import torchvision.transforms.v2 as T

imagenet_mean = [0.485, 0.456, 0.406]
imagenet_std  = [0.229, 0.224, 0.225]

transform = T.Compose([

# Resize to 224x224
T.Resize((224, 224)),

# Convert to torch Image
T.ToImage(),

# Convert to scaled float tensor
T.ToDtype(torch.float32, scale=True),

# Apply ImageNet normalization
T.Normalize(mean=imagenet_mean,
            std=imagenet_std)
])
```

Augmentation should happen after train, validation, and test splitting



Images from "On Leaky Datasets and a Clever Horse"

Summary

Data augmentation creates synthetic input data

- Data augmentation can improve the robustness of a model and prevent overfitting

PyTorch transforms are powerful and composable

- `transforms.v2` provides rotation, scale, crop, and flip (+other) operations
- Each transform has specific parameters for fine control

Avoiding data leakage: best practices

- Apply augmentations after train, validation, and test split. Never augment validation data. Test augmentation effects through observation and loss values

Further reading and references

Data augmentation is still data curation

- https://voxel51.com/blog/data-augmentation-is-still-data-curation/

Getting started with PyTorch's transforms v2

- https://pytorch.org/vision/main/auto_examples/transforms/plot_transforms_getting_started.html

Illustration of transforms

- https://pytorch.org/vision/main/auto_examples/transforms/plot_transforms_illustrations.html#sphx-glr-auto-examples-transforms-plot-transforms-illustrations-py

Resources

- Github Repository
- YouTube playlist
- Discord channel
  #practical-computer-vision-workshops