An Analysis of Data Augmentation Techniques in Machine Learning Frameworks

Rajy Rawther
Advanced Micro Devices, Inc.
May 2021

Agenda

- Why do we need augmentation?
- Different types of augmentations
- Analysis of augmentations for classification vs object detection
- Random parameter adjustment to prevent overfitting: Common techniques
- How ML learning frameworks handle data augmentation in the training pipeline
- Challenges
- Closing remarks

Why Do We Need Augmentation?

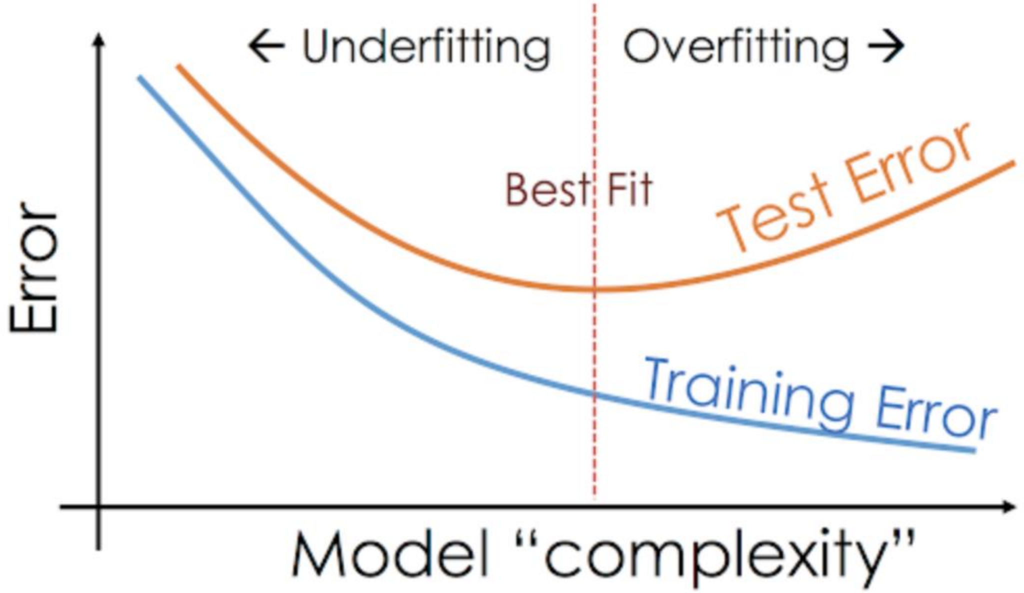
- In 2015, many of us have failed to correctly identify the color of this viral dress.
- However, today's neural networks can correctly predict the color of the dress with ~90% accuracy.
- This is achieved by training an image classification network with a large amount of data.
- Neural networks don't have misperceptions of data, but it can learn from poor data.
- The amount of data needed to train is linearly proportional to the complexity of your model.

How do I get more data if I don't have "more data"?

Overfitting

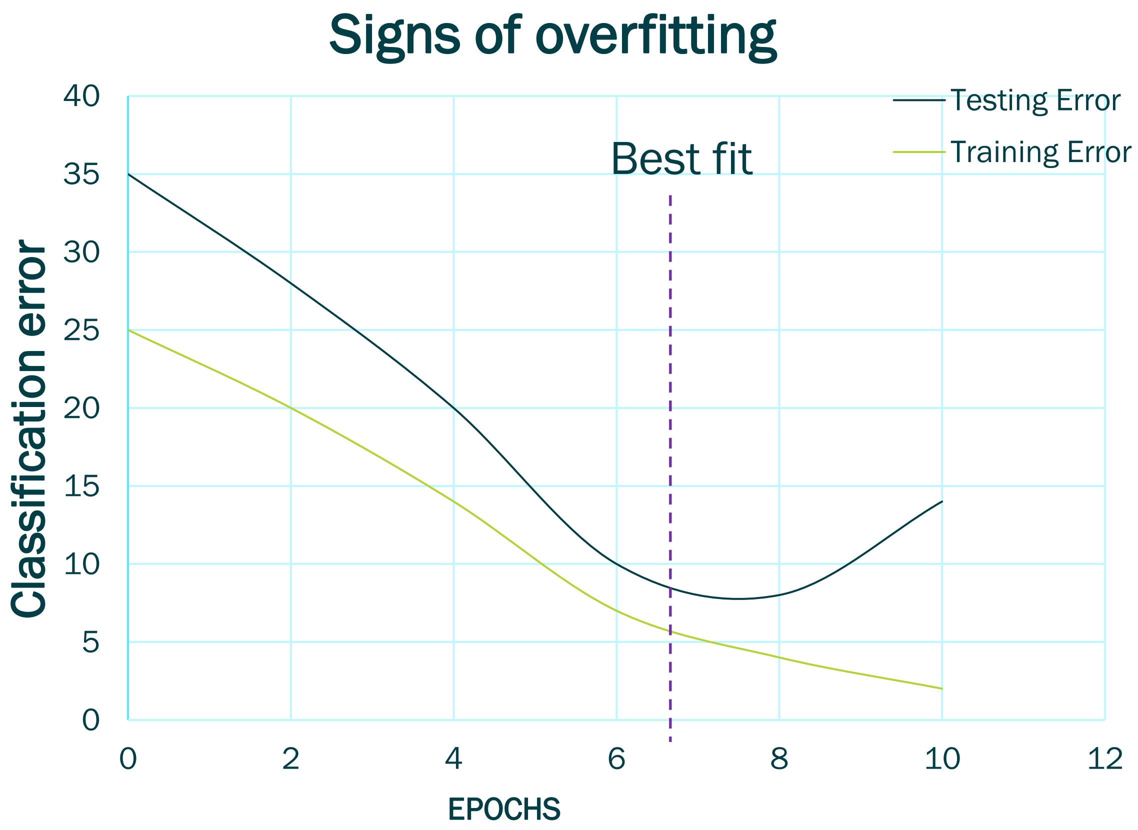
- ImageNet(ILSVRC 2012-2017) has 1.2 million training images, 50,000 validation images and 150,000 test images
- Typical image classification convolution network has millions of parameters and thousands of neurons to train

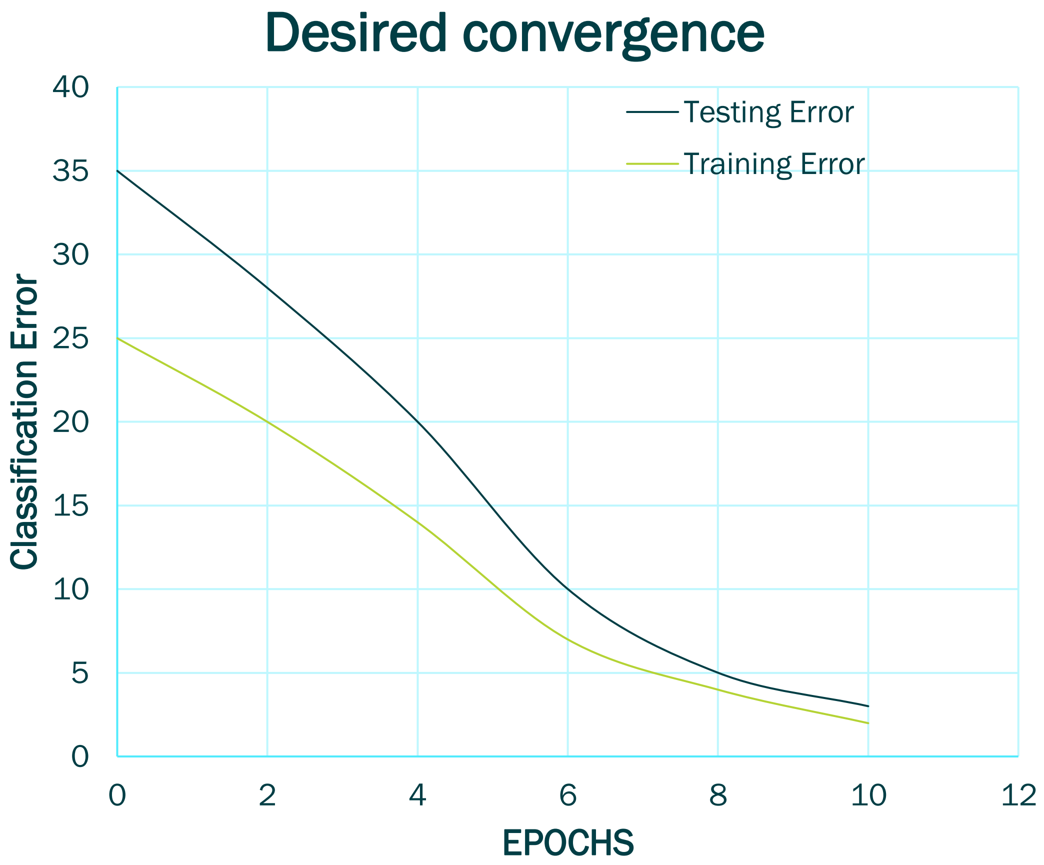
Overfitting!



Overfitting occurs when the model fits too much to the training data to the extent that it performs poorly on unseen data.

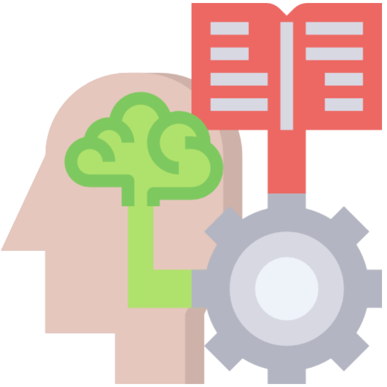
Example Of Overfitting





Data Augmentation Advantages

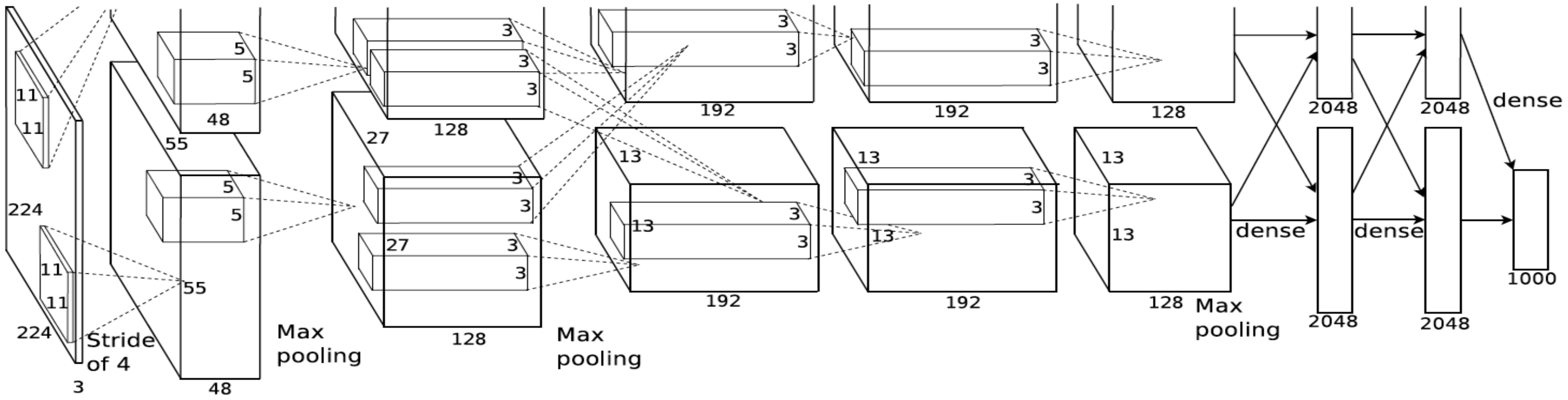
- Reduce overfitting –
  - You don’t want the network to memorize the features of training data.
- Faster training
- Increased accuracy
- Increased dataset size
- Makes your trained neural network invariant to different aspects
  - Translation
  - Size
  - Illumination
  - location
  - Mask
- Add hard-to-get or rare variations to the dataset



History: AlexNet Augmentation for Classification

- Dataset size is increased to 2048x by
  - Randomly cropping 224x224 patches
  - Doing color augmentations
  - Randomly flipping the images horizontally
  - Randomly resizing them

Resulted in 1% error rate reduction

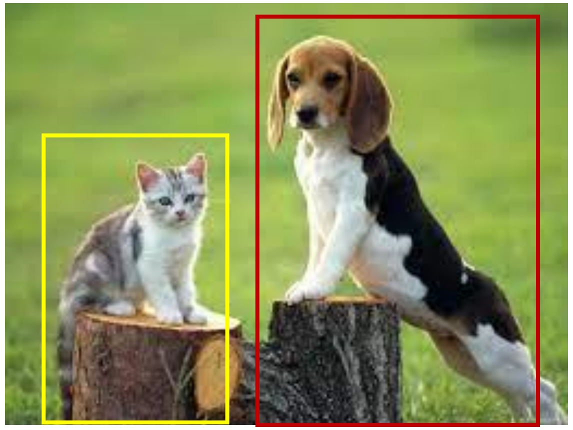


Understanding Different Use Cases For Augmentation

Image Classification

Object Detection

Segmentation



Cat or Dog? Entire Image

Classify Objects with Location

Classify each pixel to a class

Data Augmentation Categories

Color	Effects	Geometric	Filtering	Mixing
Brightness	Rain	Flip	Box	Blend
Contrast	Snow	Resize	Sobel	Non_linear-Blend
Saturation	Glitch	Crop	Gaussian	Crop And Patch
Blur	Noise	Warp Affine	Noise	Erase
Hue	Fog	Scale	Median	RICAP
Color-Temp	Gamma-Correction	Rotate		
Vignette	Water	Warp Perspective		

AMD

Color & Illumination Examples

Original

Brightness

Contrast

Saturation

Color Temp-

Vignette

Geometric and Displacement Distortion Examples

Original

Horizontal Flip

Crop

Resize

Rotate

Vertical Flip

Warp

Fish-Eye Effect

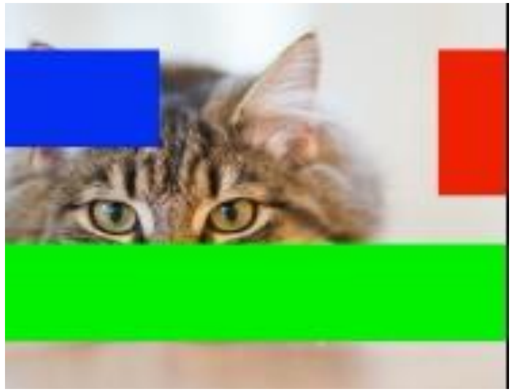
Mixing Augmentations: Disruptive

Original

Blend

ColorTwist

Erase



Nonlinear Blend

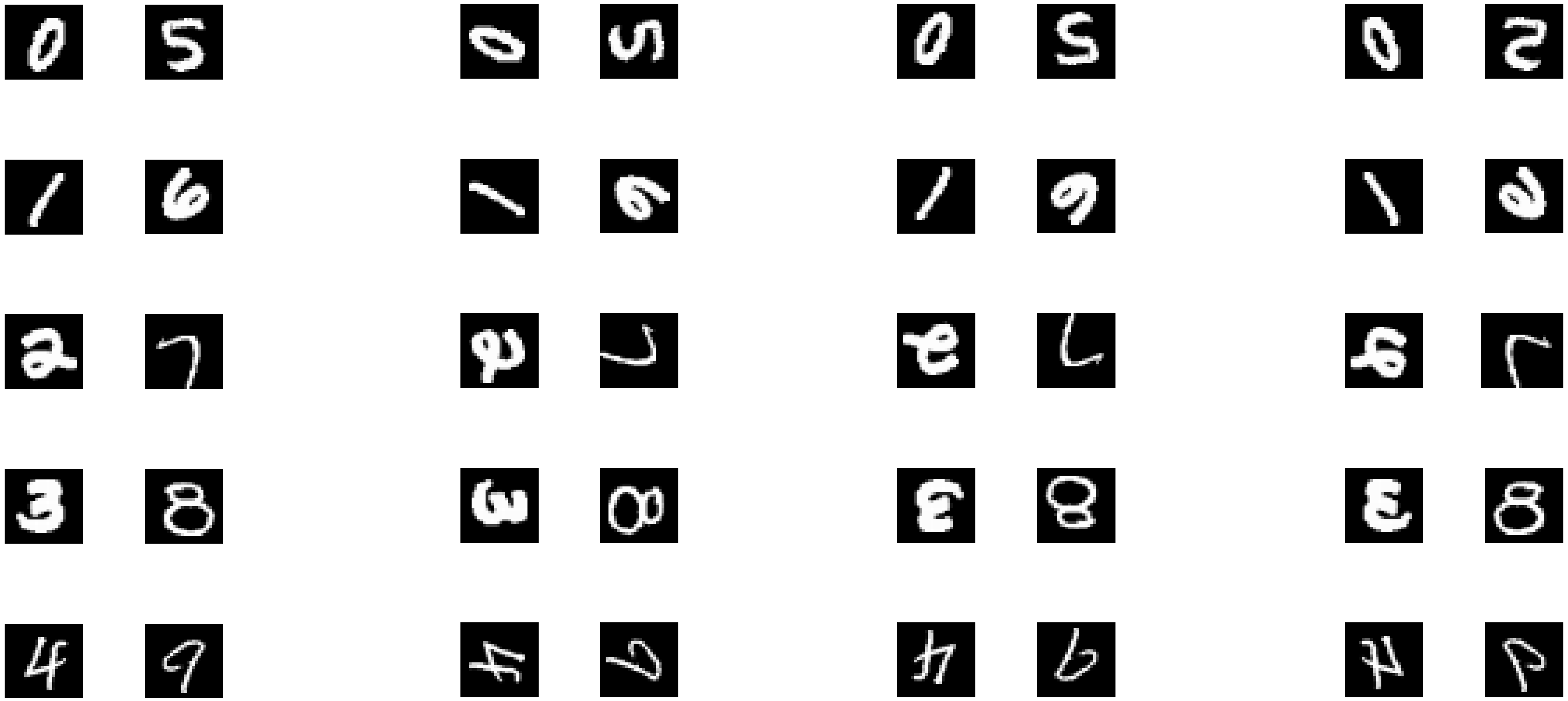
Crop And Patch

Glitch

Water

Not All Augmentations Apply To All Datasets

Original | Rotate 90 | Rotate 180 | Flip (mirror)



Random Parameter Adjustments To Prevent Overfitting

- Randomness in color augmentation
  - Real world data can exist in a variety of conditions, like low lighting, grasslands, rain, snow, etc.
  - Random parameter adjustments can help to overcome this by generating new data on the fly.

```
α = 1.0 + random.uniform(-strength, strength)
Image *= α
```

Strength to control brightness

Brightness Variation

Noise variation

Random Geometric Distortion

By doing random geometric augmentations like scale, resize, rotate, flip, etc., you are training your network to be invariant to geometric distortions.

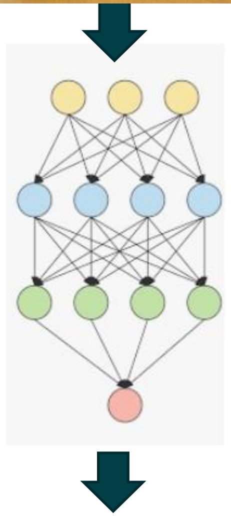
Tesla

Ford

Dataset

Tesla Car – Label 0
Ford Car – Label 1

Ford



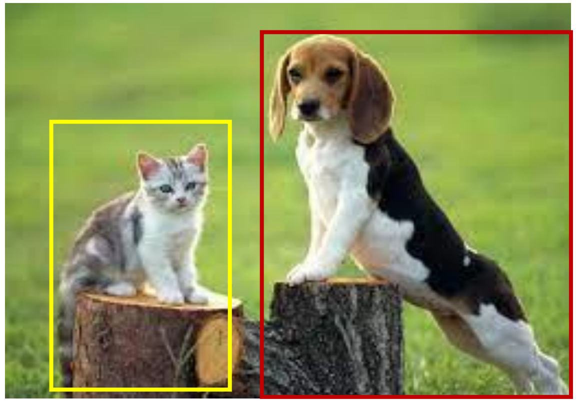
Trained Neural Net

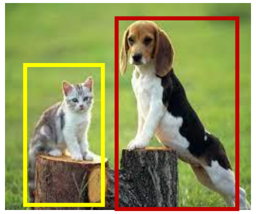
Label - 0

Tesla (wrong)

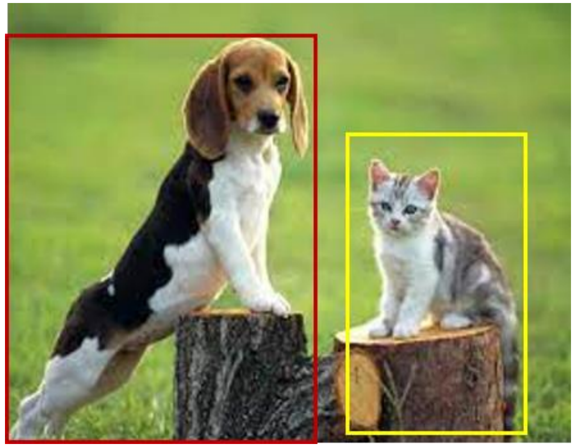
Bounding Box Augmentations

Color Augmentations don’t impact the locations of bounding boxes whereas geometric operations alters bounding box locations.

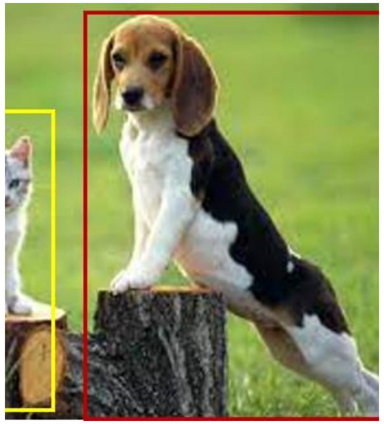




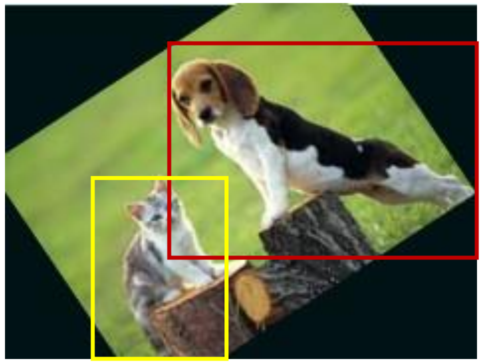
Resize



Flip



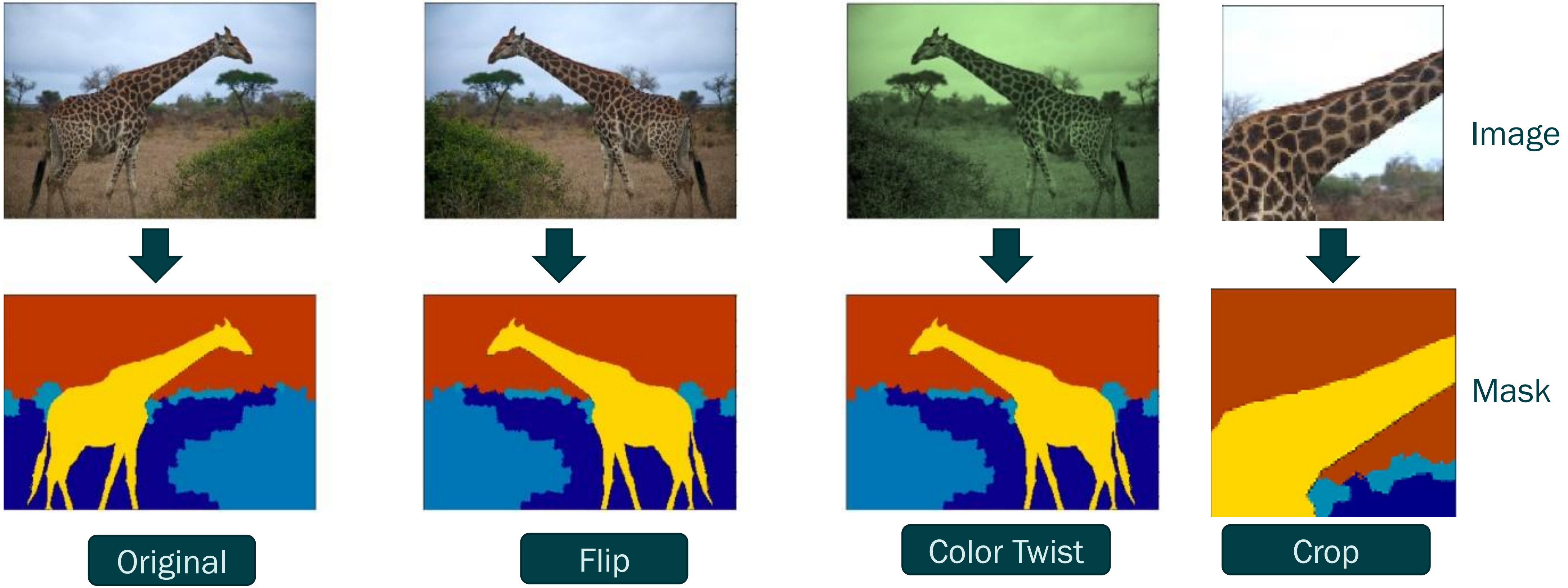
Crop



Rotate

Segmentation Mask Augmentations

- Examples of augmentations applied to base image and mask image.



How To Build A Training Pipeline With Augmentation?

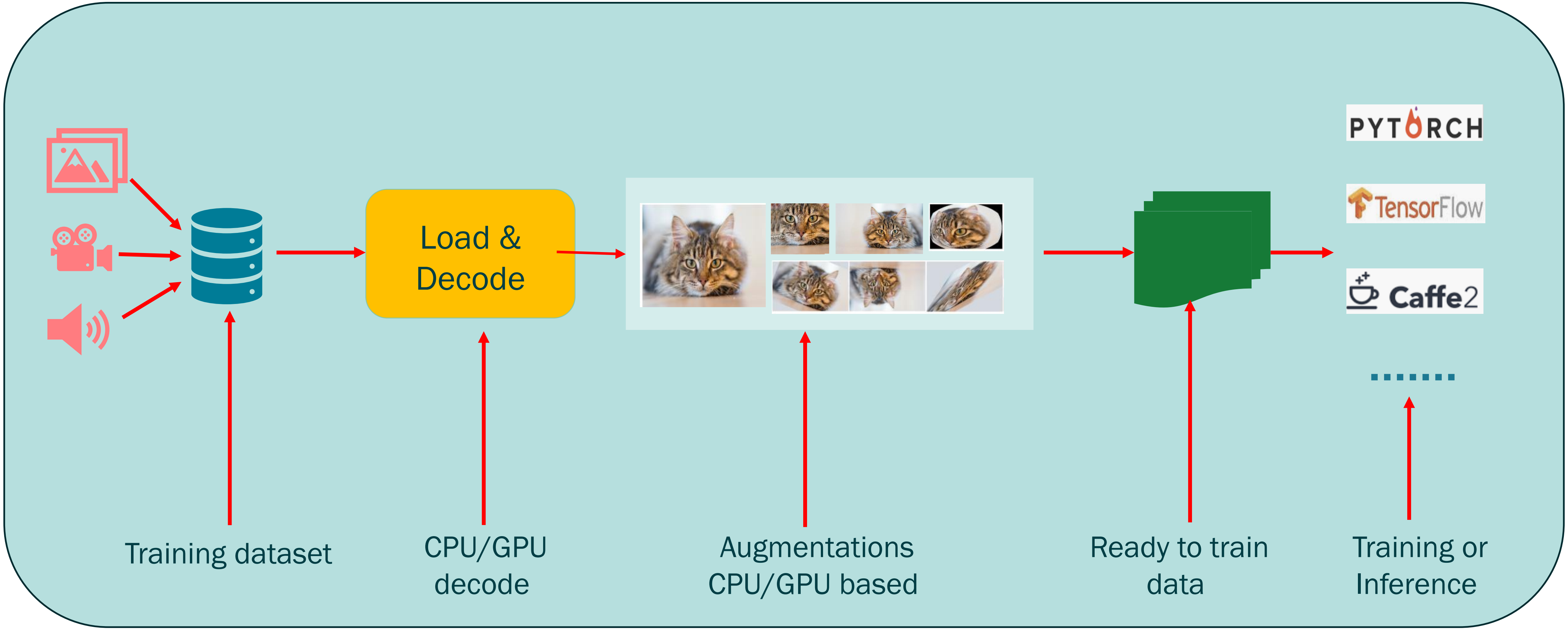
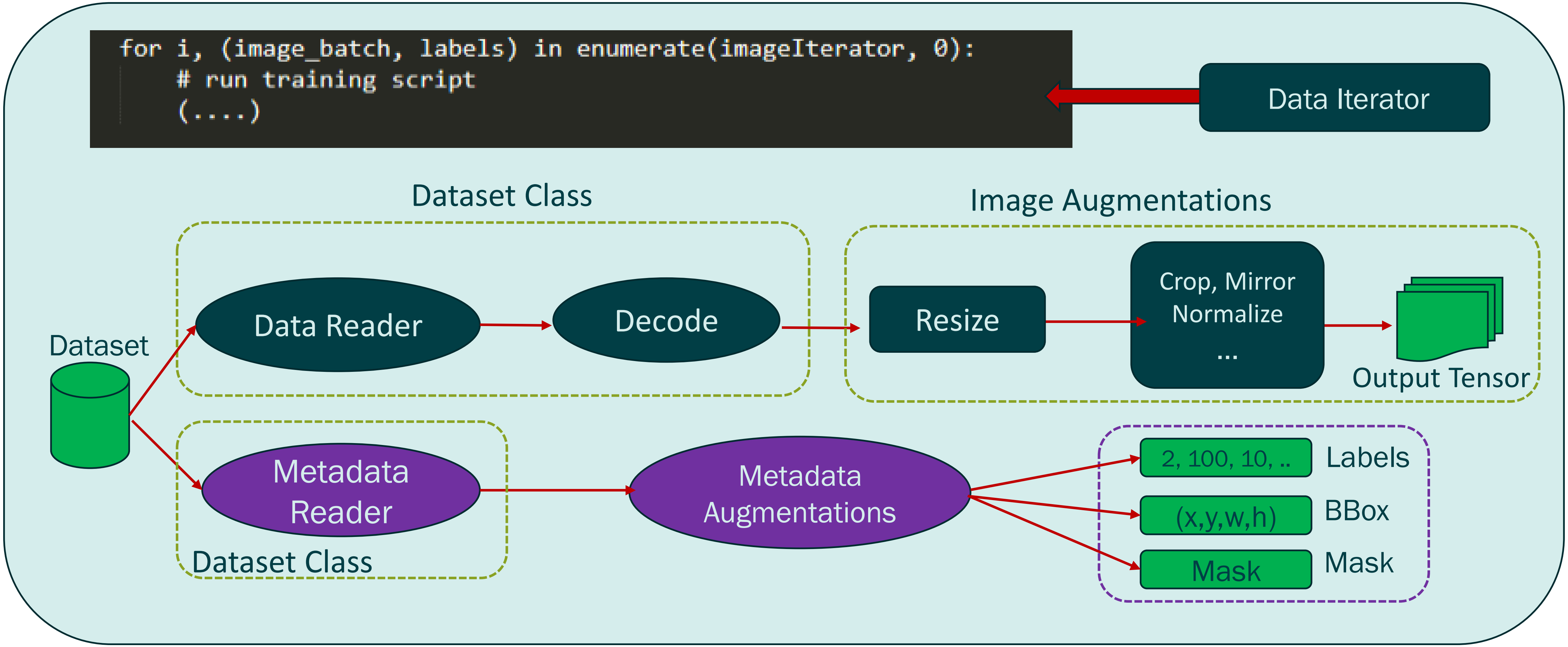


Image Classification Training With Augmentation Pipeline



Augmentations In PyTorch

PyTorch uses torchvision.transforms library to apply image transformations

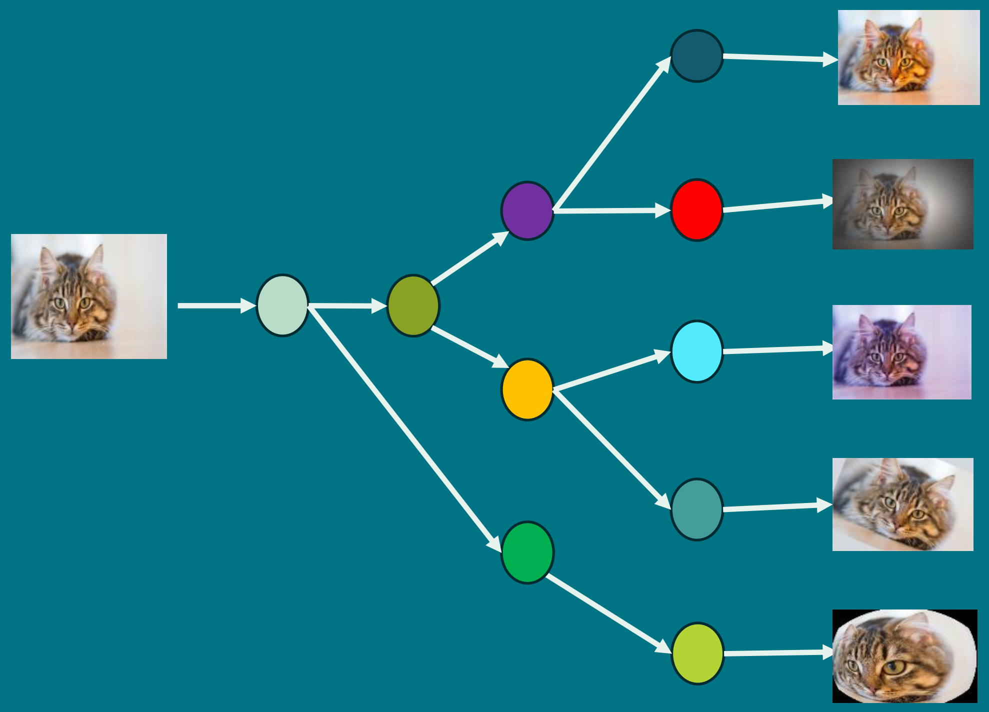
```
import torch
import torchvision
import torchvision.transforms as transforms

# define pytorch transforms
transform = transforms.Compose([
     transforms.Resize((300, 300)),
     transforms.RandomCrop((224, 224)),
     transforms.RandomHorizontalFlip(p=0.5),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
     ])

trainset = torchvision.datasets.Imagenet(root='path/to/imagenet_root/', train=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size, shuffle=True)

# get some random training images
dataiter = iter(trainloader)
for i, (image_batch, labels) in dataiter.next()
     (...) #run training script
```

Augmentation Functions In Pytorch And TensorFlow

PyTorch (torchvision.transforms)	PyTorch (torchvision.functional)	TensorFlow (lambda functions)
CenterCrop	adjust_brightness	Center_crop
Normalize	adjust_hue	Random_brightness
Resize, Scale	crop	Random_contrast
RandomCrop	equalize	Random_hue
ColorJitter	hflip	Random_flip_left_right
RandomAffine	vflip	Random_flip_up_down
RandomRotate, RandomFlip	pad	Resize_and_rescale

Data Augmentation Pipeline: Offline vs On The Fly

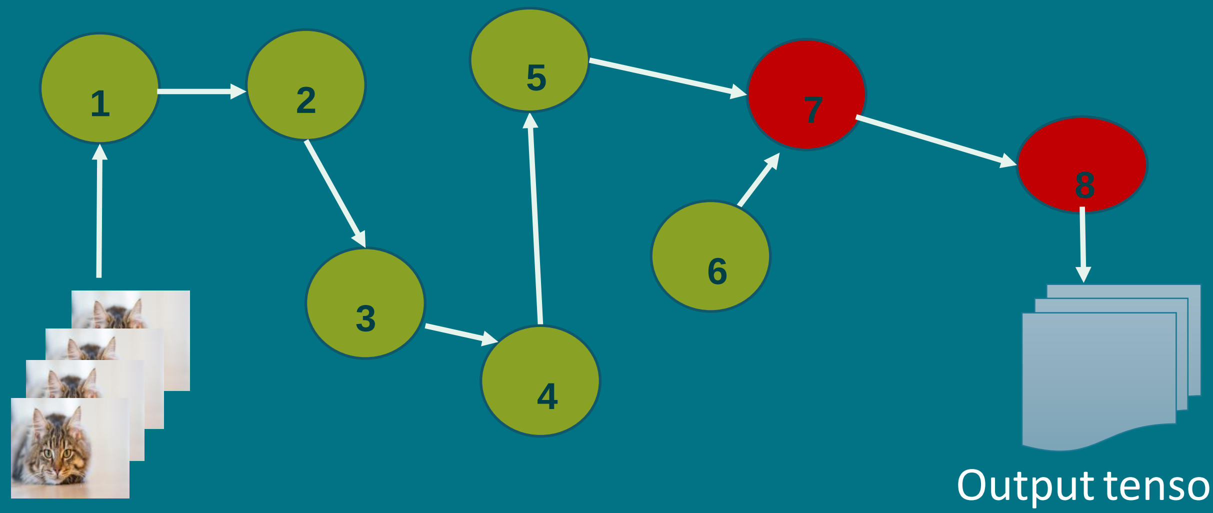
A batch of images are generated from a single image using a pipeline



Offline

Augmentations

A batch of images are processed in a pipeline using CPU/GPU to generate output tensor



On the fly

CPU Aug

GPU Aug

Example: SSD Object Detection Training

- Object detection and classification are done in a single forward pass of the network
- Bounding boxes need to be processed along with images to compute the loss function

$$L_{Total} = L_{confidence} + \alpha L_{loc}$$

- Known to perform worse on smaller objects since they disappear in some feature maps, because priors were precomputed at different feature maps.
- SSD uses VGG-16 as the base network for classification
- To alleviate this, SSDRandomCrop augmentation is used.

Example: SSD Object Detection Training Augmentations

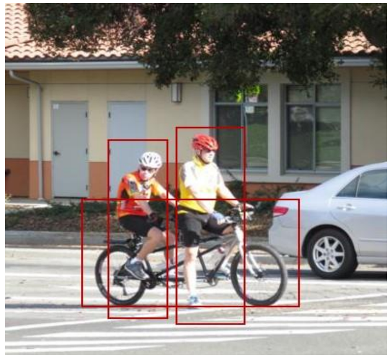
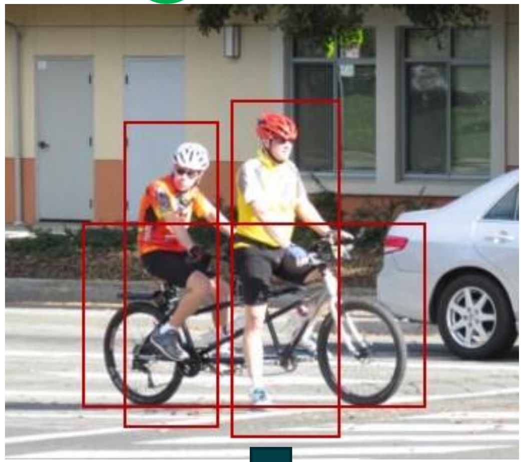


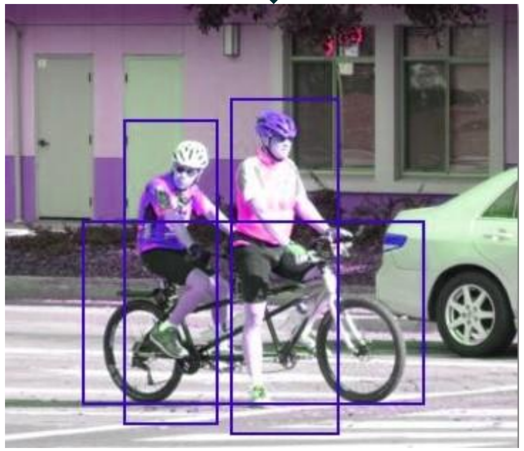
Image with bboxes

SSDRandomCrop → ColorTwist → Resize → RandomMirror

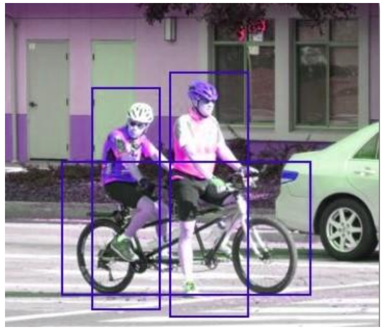
Bad crop

Good crop

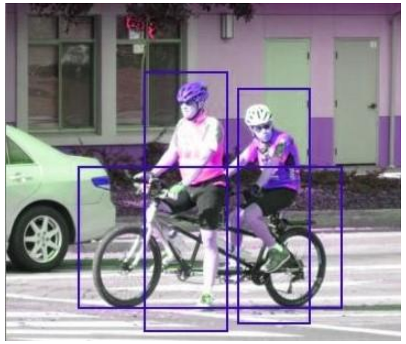




Color Twisted



Resized



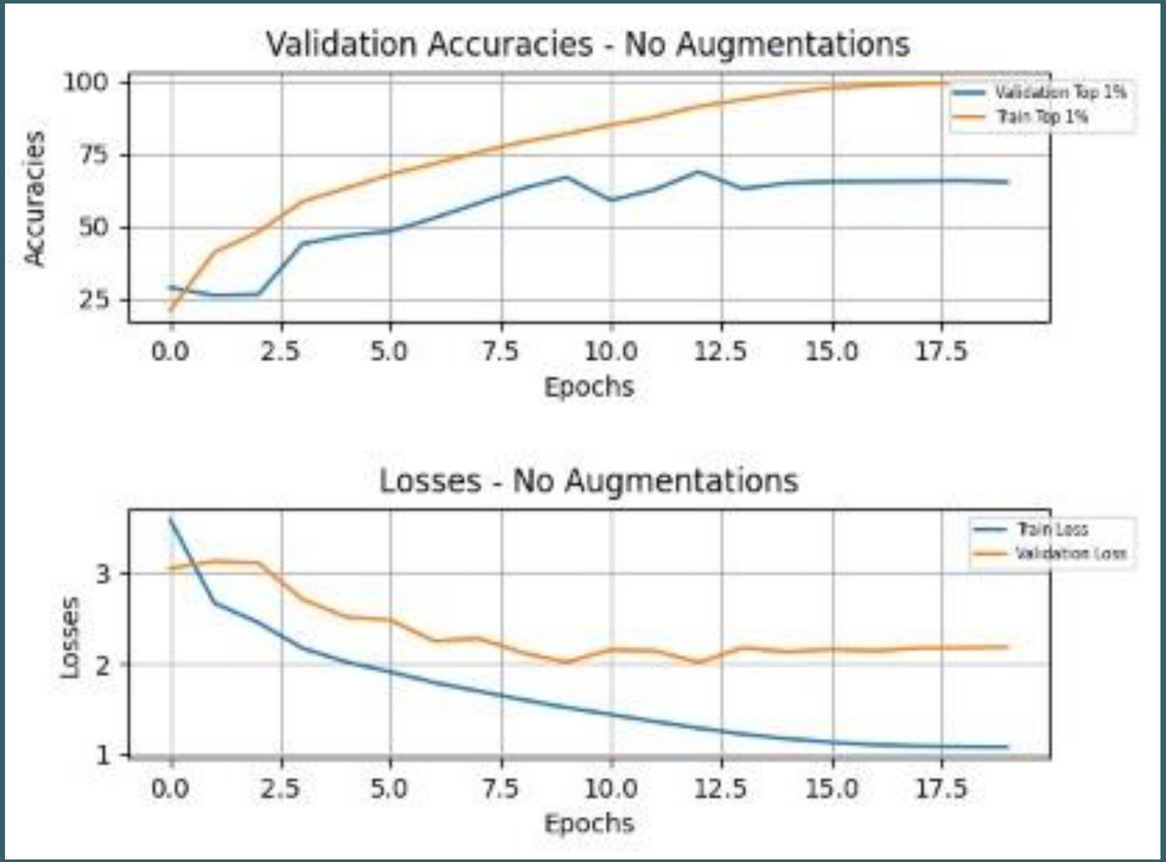
Randomly flipped

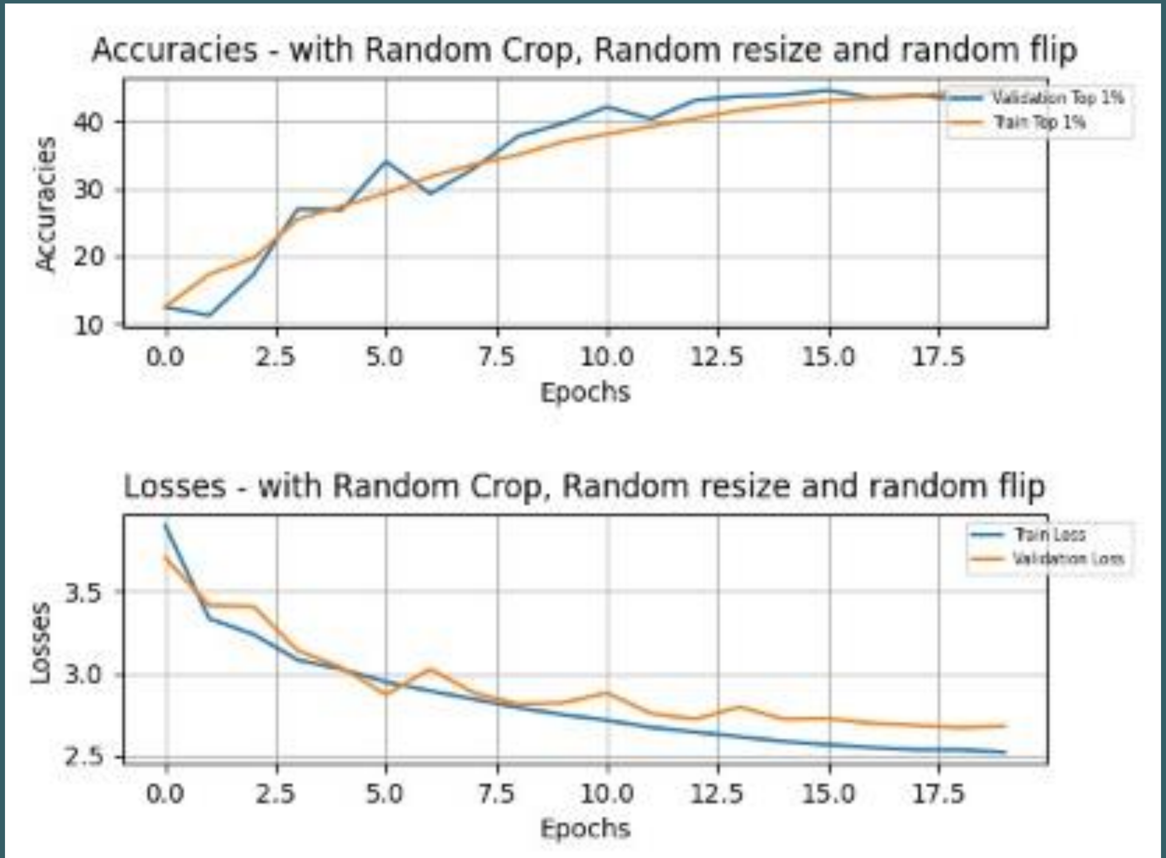
Data Augmentation Results

Augmentation Variation	Train Accuracy (top1)	Validation Accuracy (top 1)	Train Loss	Validation Loss
Almost no augmentation	91.5	65.3	1.08	2.17
Normalization	91.5	71.78	1.109	2.04
Random Resize + Random Crop + Normalization	97.7	77.2	1.5	1.65
Random Resize + Random CMN*	97.8	76.7	1.49	1.62

*CMN: Random **C**rop and **M**irror with **N**ormalization
Table generated for ResNet50 training on a smaller subset of ImageNet dataset using PyTorch

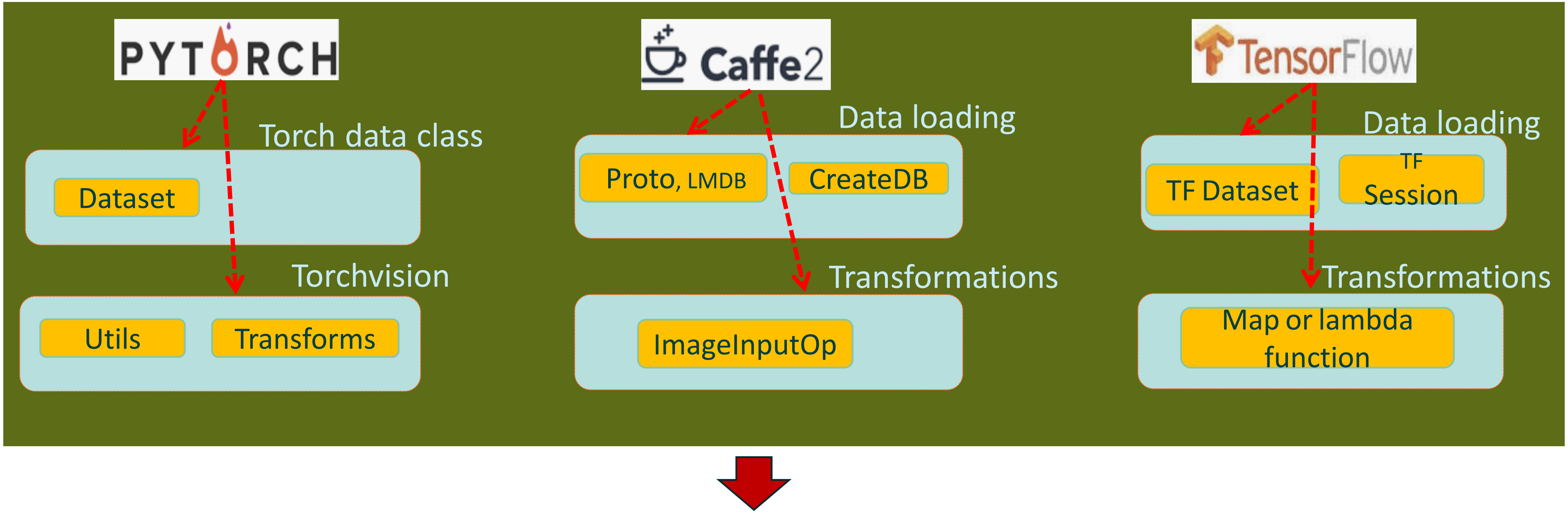
Training Results





Data Augmentation Framework Challenges

- Each Framework has its own data loader and augmentation pipelines.
- Extra effort to optimize them individually: not portable



Need a unified library which can work across all the frameworks.

Data Augmentation Algorithm Challenges

- Designing an ideal augmentation strategy is heuristic and can result in sub-optimal training outcomes
- Data augmentation techniques can greatly depend on the dataset: e.g., face detection dataset
    - Component invariant (hairstyle, makeup, accessory)
    - Attribute (pose, expression, age)
- Each of the augmentation use cases has many challenges when it comes to choosing the ideal augmentation strategy.
- A unified augmentation library that can work across all frameworks can provide both performance and flexibility

Wrap it up

- Data Augmentation: To prevent overfitting and expand dataset
- Data Augmentation pipelines can greatly vary based on the use case
- Choosing the ideal augmentation pipeline is tricky and needs some automation
- Neural Style Transfer and GANs bring an artistic approach to augmentation and can provide automation

References

ImageNet

http://www.image-net.org/challenges/LSVRC/2012/

PyTorch Transforms

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

TensorFlow Augmentations

https://www.tensorflow.org/tutorials/images/data_augmentation

MIVisionX

https://github.com/GPUOpen-ProfessionalCompute-Libraries/MIVisionX

A survey on Image Data Augmentation for Deep Learning

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

rocAL (ROCm Augmentation Library)

https://github.com/GPUOpen-ProfessionalCompute-Libraries/MIVisionX/tree/master/rocAL

AMD

Disclaimer

- The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.
- THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
-
- AMD, the AMD Arrow logo, Epyc, Radeon, ROCm and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

2021 embedded vision summit

VIRTUAL | MAY 25-28

AMD