Course Notes: Deep Learning for Visual Computing

Peter Wonka

August 30, 2021

Contents

1 Data Augmentation **4**

1.1 What is Data Augmentation? . . . 5
1.2 Why Data Augmentation? . . . 6
1.3 Example Operations / Transformations . . . 7
1.4 How to fill new pixels . . . 10
1.5 Common Data Augmentation Examples . . . 12
1.6 Cutout . . . 17
1.7 Sample Pairing . . . 19
1.8 Mixup . . . 21
1.9 How to combine Operations / Transformations . . . 23
1.10 PyTorch Comments . . . 25
1.11 Test Time Augmentation . . . 26
1.12 AutoAugment Overview . . . 27
1.13 AutoAugment Search Algorithm . . . 29
1.14 AutoAugment Discussion . . . 31

1.15 Population Based Augmentation 32
1.16 RandAugment 33
1.17 Various Strategies for Learning Data Augmentation 35
1.18 Further Reading 36

1 Data Augmentation

1.1 What is Data Augmentation?

- **Data Augmentation**: is a data processing step that increases the amount of available samples / data by performing transformations on the original data
- Examples: cropping, padding, flipping of images
- Challenge: the transformations should **preserve** the **original labels** / annotations of the data.
  - Counter-example: image of a number with label 6, data augmentation flips the image upside down, new image looks like 9.

→

→

1.2 Why Data Augmentation?

- **Data augmentation** should improve performance of a machine learning framework, especially if not a lot of data is available
- Data augmentation teaches a model about **invariances** in the data domain
- Data augmentation should **reduce overfitting**
- Data augmentation should **improve generalization**

1.3 Example Operations / Transformations

- List of common image transformations that are used for data augmentation.
- Image transformations can have parameters. The parameter is denoted as *magnitude* below.

Operation Name	Description
H-Flip	Flip the image horizontally
V-Flip	Flip the image vertically
Zoom / Crop	Zoom into the image / crop a smaller region and resize
ShearX(Y)	Shear the image along the horizontal (vertical) axis with rate *magnitude*.
TranslateX(Y)	Translate the image in the horizontal (vertical) direction by *magnitude* number of pixels.
Rotate	Rotate the image *magnitude* degrees.
AutoContrast	Maximize the the image contrast, by making the darkest pixel black and lightest pixel white.
Invert	Invert the pixels of the image.
Equalize	Equalize the image histogram.
Solarize	Invert all pixels above a threshold value of *magnitude*.
Posterize	Reduce the number of bits for each pixel to *magnitude* bits.
Contrast	Control the contrast of the image. A *magnitude*=0 gives a gray image, whereas *magnitude*=1 gives the original image.

Color	Adjust the color balance of the image, in a manner similar to the controls on a colour TV set. A *magnitude*=0 gives a black & white image, whereas *magnitude*=1 gives the original image.
Brightness	Adjust the brightness of the image. A *magnitude*=0 gives a black image, whereas *magnitude*=1 gives the original image.
Sharpness	Adjust the sharpness of the image. A *magnitude*=0 gives a blurred image, whereas *magnitude*=1 gives the original image.
Cutout	Set a random square patch of side-length *magnitude* pixels to gray.
Sample Pairing	Linearly add the image with another image (selected at random from the same mini-batch) with weight *magnitude*, without changing the label.

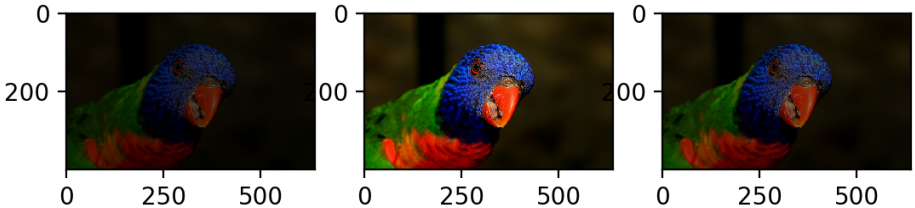
1.4 How to fill new pixels

- Several image transformations can introduce new pixels into the image, e.g. rotation
- We need to specify how to fill in the new pixels
- Example, from left to write
  - set new pixels to a constant value. e.g. 0
  - set new pixel colors to the nearest boundary pixel color
  - reflect pixel colors at the boundary
  - symmetric reflect: similar to reflect, except for the fact that, at the boundary of reflection, a copy of the edge pixels are made
  - wrap: image is repeated as if tiled

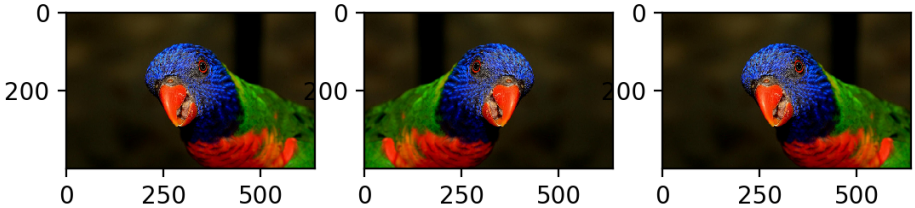
1.5 Common Data Augmentation Examples

- Input Image:

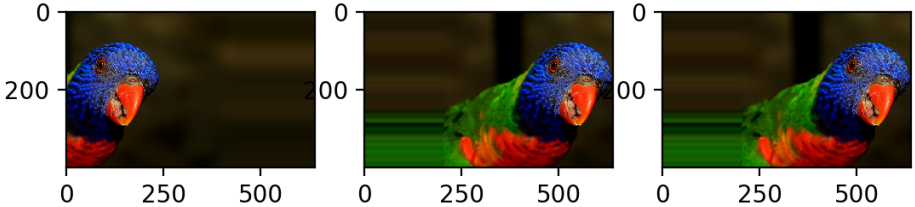
- Brightness:



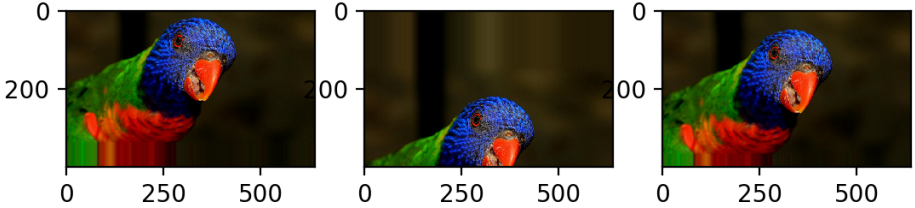
- Horizontal Flip



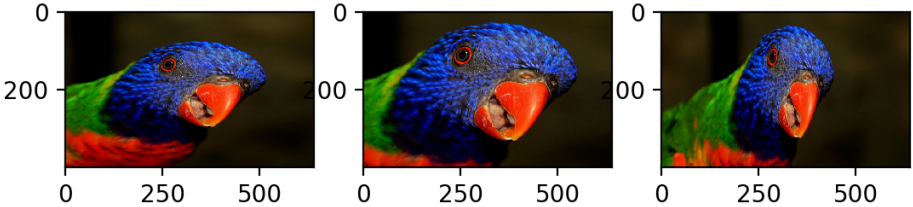
- Horizontal shift



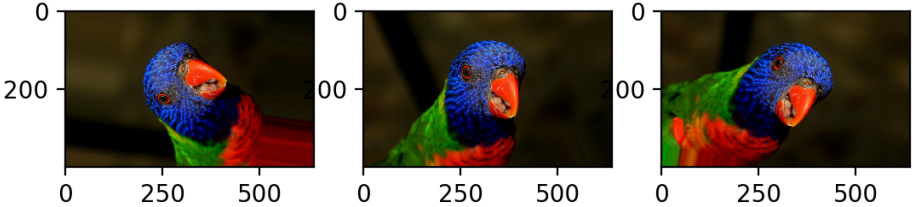
- Vertical shift



- Zoom

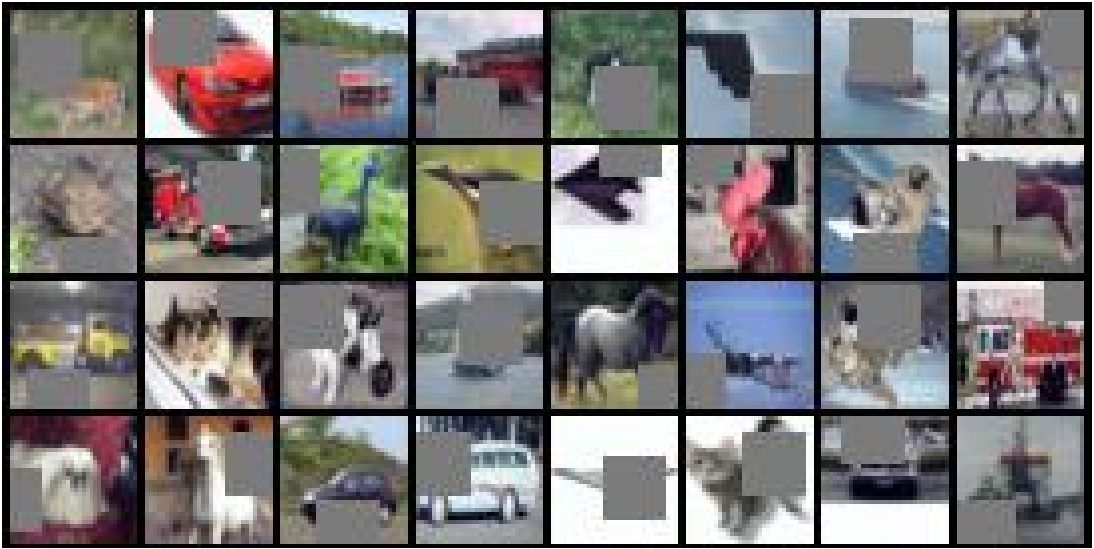


- Rotation



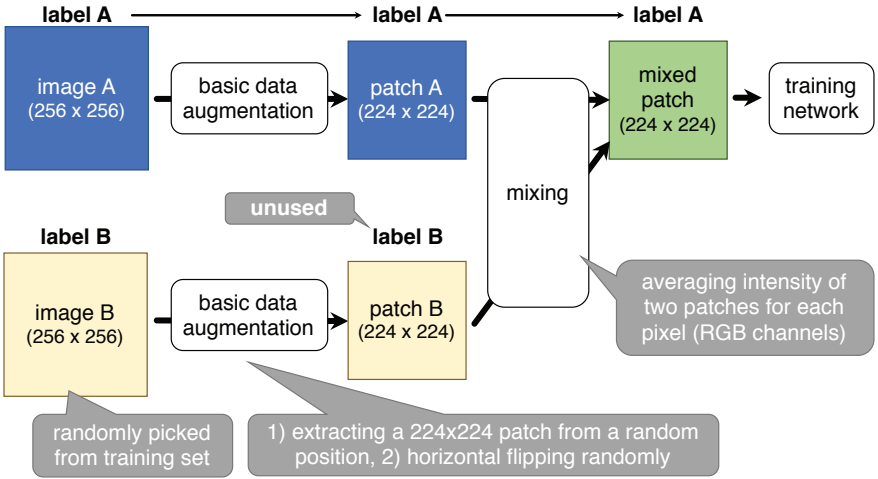
1.6 Cutout

- Literature:
  - T. DeVries, G. Taylor, Improved regularization of convolutional neural networks with cutout
  - Z. Zhong, Random erasing data augmentation
- Cutout a random rectangle / square from the input image
  - square region is set to 0 in the transformed space (should be a grayish color)
- Authors claim the shape of the region is much less important than the size
  - $\rightarrow$ therefore authors proposed the simplest possible region, a square
- Examples:



1.7 Sample Pairing

- Literature:
  - Hiroshi Inoue, Data Augmentation by Pairing Samples for Images Classification
- Combine two images I_1 and I_2 by linear blending: $\lambda I_1 + (1-\lambda) I_2$
  - Paper uses $\lambda = 0.5$ and simply keeps the label of I_1 as the correct label
- Image pairs are selected randomly from the data set without considering class labels
  - Pairing only images with identical or similar labels degrades performance
- Paper uses a schedule:
  - interleave epochs with sample pairing and without pairing
  - fine-tune without sample pairing
- Illustration:


label A
label A
label A
image A
(256 x 256)
basic data
augmentation
patch A
(224 x 224)
mixing
mixed
patch
(224 x 224)
training
network
unused
label B
label B
image B
(256 x 256)
basic data
augmentation
patch B
(224 x 224)
averaging intensity of
two patches for each
pixel (RGB channels)
randomly picked
from training set
1) extracting a 224x224 patch from a random
position, 2) horizontal flipping randomly

1.8 Mixup

- Literature: H.Zhang, mixup: Beyond empirical risk minimization
- Form a convex combination of samples, e.g. image tensors I_1, I_2

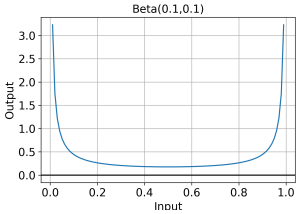
$$I_{new} = \lambda I_1 + (1-\lambda) I_2 \tag{1.1}$$

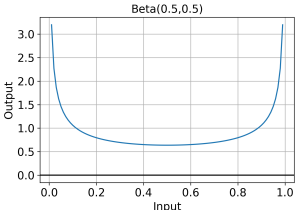
- Form a convex combination of one hot ground truth labels, using the same λ:

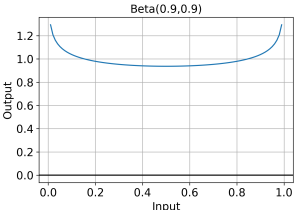
$$l_{new} = \lambda l_1 + (1-\lambda) l_2 \tag{1.2}$$

- Sample λ from a Beta distribution:

$$\lambda \sim Beta[\alpha, \alpha], \quad \alpha \in (0, \inf) \tag{1.3}$$


Beta(0.1,0.1)
Output
Input
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0


Beta(0.5,0.5)
Output
Input
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0


Beta(0.9,0.9)
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.2
0.4
0.6
0.8
1.0

1.9 How to combine Operations / Transformations

- Apply one transformation per sample (e.g. image) vs. applying multiple transformations in sequence
- Apply the same transformation (sequence) for all samples in a mini-batch vs. sampling a new transformation for each sample
- Assign probabilities to individual transformations / assign probabilities to chains of operations
- Use a discrete set of transformation parameters / use continuous set of parameters
  - e.g. rotation angle
- Autoaugment solution:
  - A sub-policy is a combination of k transformations, each with probability of being executed and parameters. $(k = 2)$
  - A policy is a combination of sub-policies. Each subpolicy has a probability of

being selected.

1.10 PyTorch Comments

- We just give some example commands
- torchvision.transforms.Compose(transforms)
  - Composes several transforms together
- torchvision.transforms.CenterCrop(size)
  - Crops the given PIL Image at the center
- torchvision.transforms.ColorJitter(brightness=0, contrast=0, saturation=0, hue=0)
  - Randomly change the brightness, contrast and saturation of an image
- torchvision.transforms.RandomAffine(degrees, translate=None, scale=None, shear=None, resample=False, fillcolor=0)
  - Random affine transformation of the image keeping center invariant

1.11 Test Time Augmentation

- Data Augmentation can also be applied at test time
  - e.g. an image can be classified with k different transformation (sequences) and the results can be aggregated
- Important to compare methods using the same test time augmentation

1.12 AutoAugment Overview

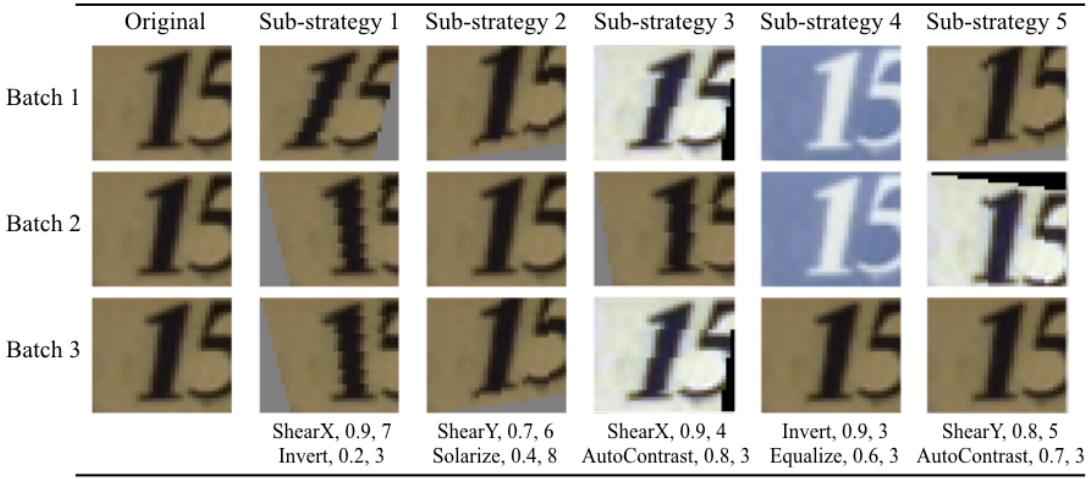
- Literature: Ekin Cubuk et al., AutoAugment: Learning Augmentation Strategies from Data
- Results from the paper:
  - Error rates (%) from this paper compared to the best results so far on five datasets (Top-5 for ImageNet, Top-1 for the others).
  - See paper for exact description of competitors
  - GPU hours are estimated for an NVIDIA Tesla P100

Dataset	GPU hours	Best published results	Our results
CIFAR-10	5000	2.1	1.5
CIFAR-100	0	12.2	10.7
SVHN	1000	1.3	1.0
Stanford Cars	0	5.9	5.2
ImageNet	15000	3.9	3.5

- Approach: discrete search problem; Requires:
  - Search Algorithm
  - Search Space

1.13 AutoAugment Search Algorithm

- Search Algorithm: Reinforcement Learning. Components:
  - Controller: recurrent neural network
  - Training algorithm: Proximal Policy Optimization
  - Alternatives would be other search algorithms, e.g. genetic algorithms, random search
- Search Space
  - a policy consists of 5 sub-policies
  - A sub-policy consists of two image transformations applied in sequence
  - hyperparameters: probability of applying the transformation and the parameters for the transformation
  - parameters and probabilities are discretized into 10 and 11 values respectively
  - Example policy is shown below:


Original
Sub-strategy 1
Sub-strategy 2
Sub-strategy 3
Sub-strategy 4
Sub-strategy 5
Batch 1
Batch 2
Batch 3
ShearX, 0.9, 7
Invert, 0.2, 3
ShearY, 0.7, 6
Solarize, 0.4, 8
ShearX, 0.9, 4
AutoContrast, 0.8, 3
Invert, 0.9, 3
Equalize, 0.6, 3
ShearY, 0.8, 5
AutoContrast, 0.7, 3

1.14 AutoAugment Discussion

- AutoAugment is a very expensive algorithm
- Generating enough samples for reinforcement learning is difficult
- No computation is shared between samples
- For example, if using Google Cloud on-demand P100 GPUs in 2019, it would cost about $7,500 to discover a CIFAR policy, and $37,500 to discover an ImageNet one
- Common application seems mainly to transfer a learned policy, e.g. the imagenet policy to other datasets
  - Works relatively well in experiments

1.15 Population Based Augmentation

- Literature: Ho et al., Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules
- Generates an augmentation policy schedule which can adapt based on the current epoch of training.
- Population of workers evaluating a schedule
  - Explore: modifying hyperparameters of workers
  - Exploit: copy weights of high performing workers to lower performing ones

1.16 RandAugment

- Literature: Ekin et al., RandAugment: Practical automated data augmentation with a reduced search space
- Use **K = 14** different transformations: identity, autoContrast, equalize, rotate, solarize, color, posterize, contrast, brightness, sharpness, shear-x, shear-y, translate-x, translate-y
- Perform N transformations per training image
  - For each transformation, select it with probability $\frac{1}{K}$
- Configure the transformations with an integer strength from 0 (min) to 10 (max)
- Pick a single parameter M to select the strength **globally** (same M for all transformations)
- Experiment with four schedules:
  - constant magnitude

- random magnitude
- linearly increasing magnitude
- random magnitude with increasing upper bound

1.17 Various Strategies for Learning Data Augmentation

- Use **Generative Adversarial Networks** (GANs) to generate additional data
- Use a **Variational Autoencoder** to generate additional data
- Use **GANs** generate **augmentation strategies** (selecting transformations and parameters)
- Use another neural network to **learn data transformations**

1.18 Further Reading

- Lemley, Smart Augmentation - Learning an Optimal Data Augmentation Strategy
- Albumentations library for transforming images