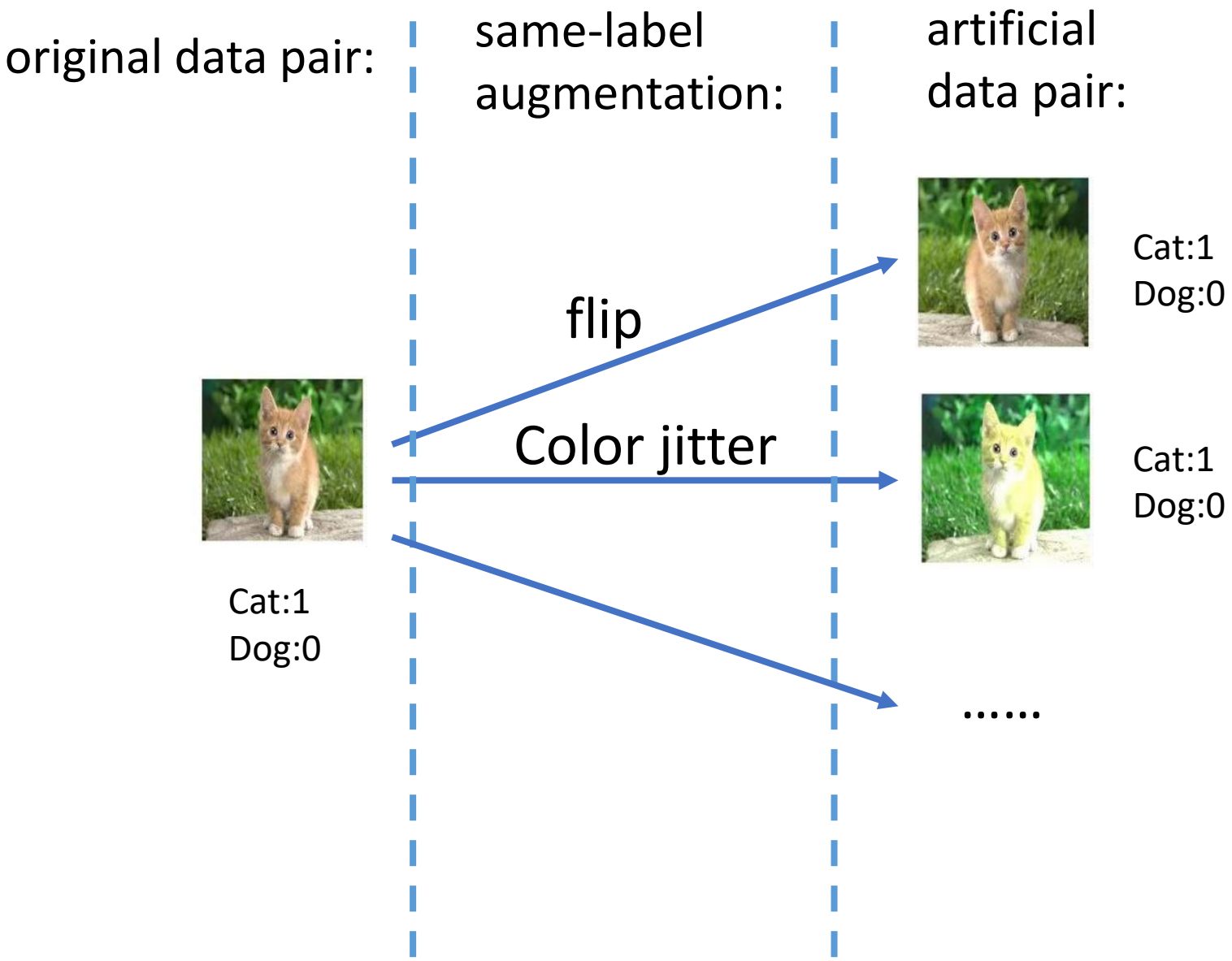
TransMix: Attend to Mix for Vision Transformers

CVPR 2021

Jie-Neng Chen[1*] Shuyang Sun[2*] Ju He[1] Philip Torr[2] Alan Yuille[1] Song Bai[3]
[1]Johns Hopkins University [2]University of Oxford [3]ByteDance Inc.

- Motivation
- Preliminary
- Methodology
- Experiment

Motivation: Image Augmentation



Motivation: Image Augmentation

original data pair:

cross-label augmentation:

artificial data pair:

Cat:1
Dog:0

Cat:0
Dog:1

Mixup

Cut Mix

Cat:0.5
Dog:0.5

$$\lambda = \frac{Strength_A}{Strength_{total}}$$

Cat:0.8
Dog:0.2

$$\lambda = \frac{Area_A}{Area_{total}}$$

.....

Motivation: Image Augmentation

Not all pixels are created equal!

Cat:0
Dog:1

Cat:1
Dog:0

Cat:0.3
Dog:0.7

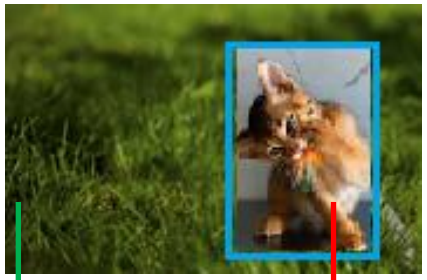
But actually this is almost an image of merely cat.

Motivation: Image Augmentation

A new method considering the importance of a pixel is needed

Cat:0
Dog:1

Cat:1
Dog:0



Cat: 0.95
Dog: 0.05

Background
Not important

Object
Important

Preliminary: How to measure importance?

ViT divides an image to patches, and applies attention to those patches.

Source: Google AI blog

Preliminary: How to measure importance?

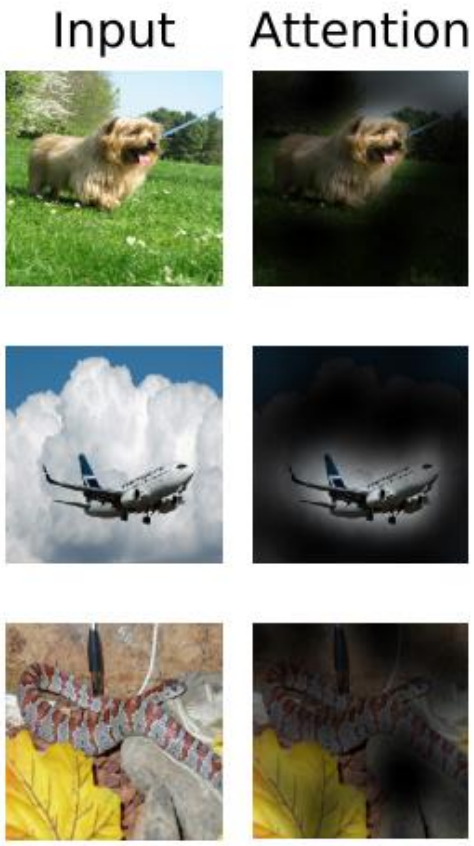


Figure 6: Representative examples of attention from the output token to the input space. See Appendix D.7 for details.

Remember ViT could assign an attention map to the picture?

Preliminary: How to measure importance?

Use the attention value to represent the importance

Cat:1
Dog:0

Cat:0
Dog:1

Cat:0.8
Dog:0.2

$$\lambda = \frac{Attn_A}{Attn_{total}}$$

Methodology

Mix Image → Use ViT model to get **label prediction** and **attention matrix** → Mix Label → Back Propagation

ViT

Attention matrix

$$\lambda = \sum Attn_A$$

Label
Cat: 1- λ
Dog: λ

Label prediction
Cat: 0.3662
Dog: 0.6338

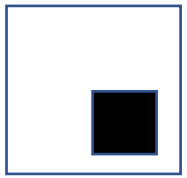
Cross Entropy Loss

Methodology

Algorithm 1 Pseudocode of TransMix in a PyTorch-like style.

```
# H, W: the height and width of the input image
# p: number of patches
# M: 0-initialized mask with shape (H,W)
# downsample: downsample from length (H*W) to (p)
# (bx1, bx2, by1, by2): bounding box coordinate

for (x, y) in loader: # load a minibatch with N pairs
   # CutMix image in a minibatch
   M[bx1:bx2, by1:by2] = 1
   x[:,:,M==1] = x.flip(0)[:,:,M==1]
   M = downsample(M.view(-1))

   # attention matrix A: (N, p)
   logits, A = model(x)

   # Mix labels with the attention map
   lam = matmul(A, M)
   y = (1-lam) * y + lam * y.flip(0)

   CrossEntropyLoss(logits, y).backward()
```

Mark the replaced part



Reverse in batch dimension

X:

……

X.flip(0):

……

Convert the mark to token format
[original patch, original patch, ……,replaced patch,……]

Calculate the attention
value of the replaced part

Experiment

- Basic Classification
- Downstream Task
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation
- Robustness

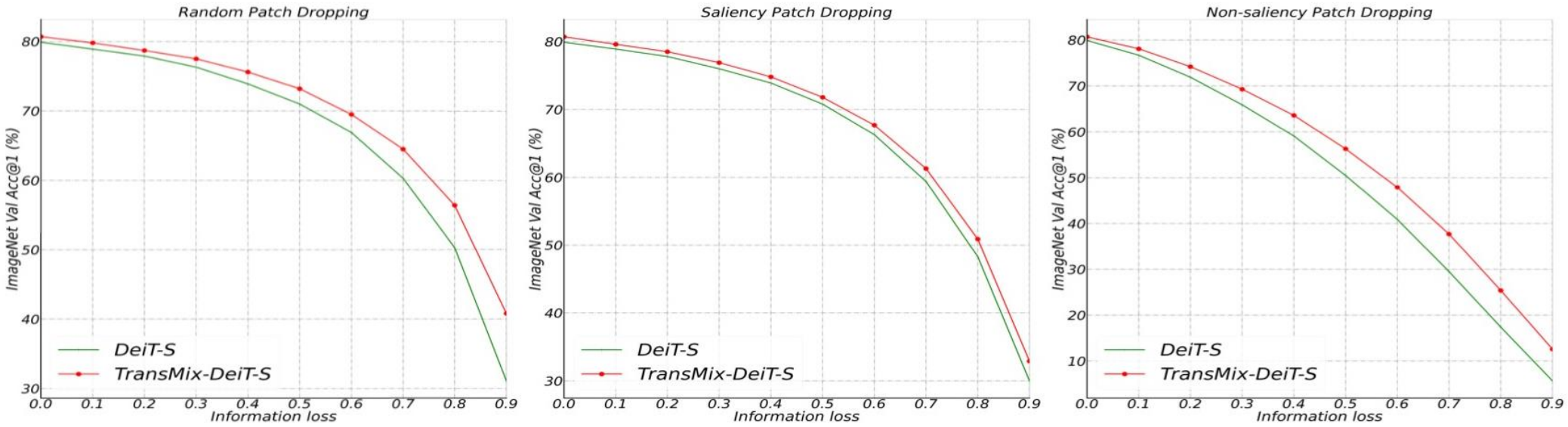
Experiment

Dataset: ImageNet1k

Basic classification

Models	Params	FLOPs	Top-1 Acc (%)	+TransMix Top-1 Acc (%)
DeiT-T [42]	5.7M	1.6G	72.2	**72.6**
PVT-T [51]	13.2M	1.9G	75.1	**75.5**
XCiT-T [15]	12M	2.3G	79.4	**80.1**
CaiT-XXS[43]	17.3M	3.8G	79.1	**79.8**
DeiT-S [42]	22.1M	4.7G	79.8	**80.7**
PVT-S [51]	24.5M	3.8G	79.8	**80.5**
XCiT-S [15]	26M	4.8G	82.0	**82.3**
PVT-M [51]	44.2M	6.7G	81.2	**82.1**
PVT-L [51]	61.4M	9.8G	81.7	**82.4**
XCiT-M [15]	84M	16.2G	82.7	**83.4**
DeiT-B [42]	86.6M	17.6G	81.8	**82.4**
XCiT-L	189M	36.1G	82.9	**83.8**

Experiment



Occlusion