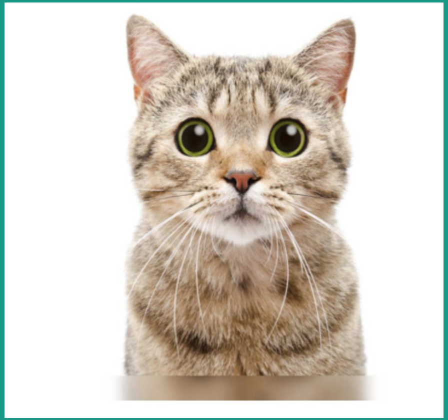
Mixup: Beyond Empirical Risk Minimization



New (Virtual) Labels

$$0.6 * \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.4 * \begin{bmatrix} 0 \\ 1 \end{bmatrix} =>> \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

Dog image , Cat image

Multiclass:

$$0.6 * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 0.4 * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} =>> \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.4 \\ 0 \end{bmatrix}$$

New hyperparameter (**alpha**)
alpha = 0.6
(1-alpha) = 0.4

Mixup

```
for (x1, y1), (x2, y2) in zip (loader1, loader2):

    lam = numpy.random.beta(alpha, alpha)
    mix =lam * x1 + (1. - lam) * x2
    y =lam * y1 + (1. - lam) * y2

    optimizer.zero_grad()
    output = net(mix)
    loss(output, y).backward()
    optimizer.step()
```

Mixup

- Better accuracy. (next slides)
- Increases the robustness to adversarial examples. (next slides)
- Reduces the memorization of corrupted labels.
- Stabilizes the training of generative adversarial networks.

Mixup: Classification

- If the dataset contains incorrect labels (labeling a cat as dog), mixup can help reducing the effect of incorrect labels.

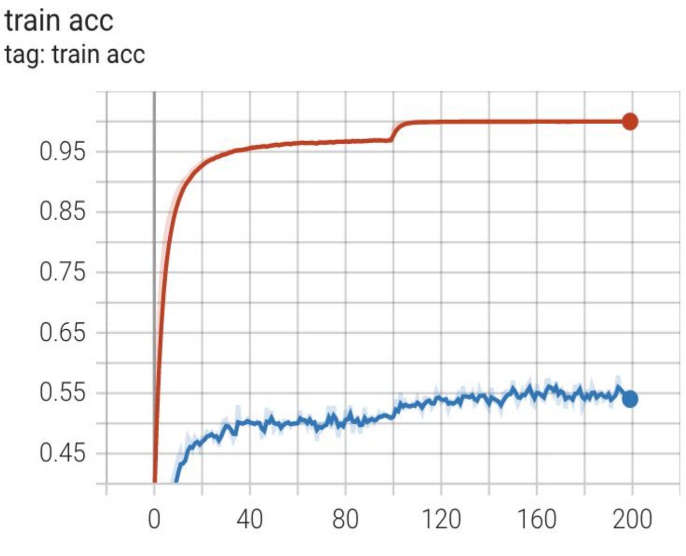
- The model will not see a pure cat nor a pure dog during training!! (the training acc will be low, but we consider the val acc only).

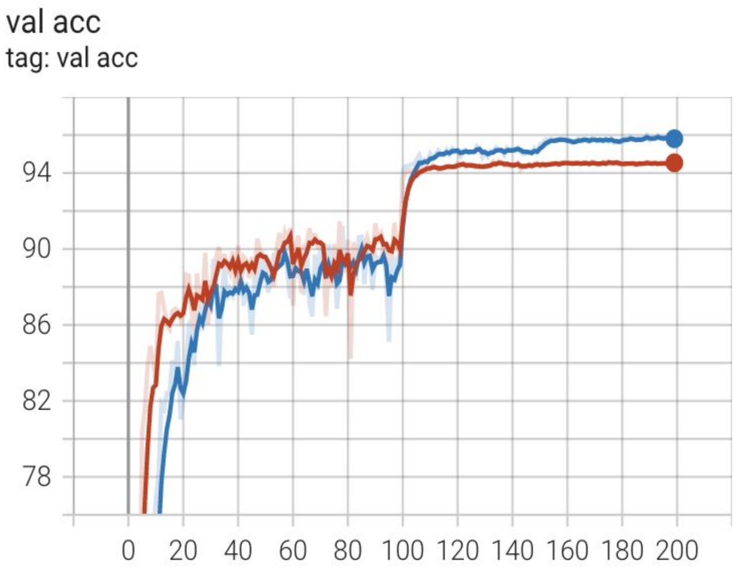
- Don't mix images from the same class!

Results on CIFAR10 (200 epochs)

	Dataset	Model	ERM	mixup
paper	CIFAR10	ResNet18 (cifar10 variant)	94.4	95.8
ours	CIFAR10	ResNet18 (cifar10 variant)	**94.8**	**95.9**

P100 GPU , ~4 hours/model, 200 epochs





RED: ERM **BLUE: Mixup**

Robustness To Adversarial Examples

- Mixup can significantly improve the robustness of the neural network to adversarial examples..
- This gradient is important to find the adversarial examples (gradient ascend /next slide).
- (**Hypothesis**) It makes the norm gradient of the loss w.r.t the input small.
- FAST GRADIENT SIGN METHOD: computes the gradients of a loss function w.r.t the input image and then takes the direction that ascends the gradients to create a new image (not weights) that *maximizes* the loss.

Robustness To Adversarial Examples

Iterative Fast Gradient Sign Method: <u>update the image not the weights</u>

```
for n_steps:
    for inputs , y in data:
        input.reqiures_grad=True   #very important
        output = net(input)
        loss(output, y).backward()
        inputs = inputs + eps* inputs.grad.sign() #gradient ascend
```

- **Test the model against the new data**

Goodfellow, et el: Explaining and harnessing adversarial examples

	Attack	Model	eps	ERM	mixup
ours	**White box attack**	ResNet18	0.007 0.5 4	**51%** **20%** **9%**	46.% 16% 9%
ours	**Black box attack**	ResNet18	0.007 0.5 4	**52%** **36%** **27%**	46% 32% 24%
paper	**White box attack**	ResNet101	4	9.3%	**14.8%**
paper	**Black box attack**	ResNet101	4	43%	**54%**

Results After one Iteration

Different results from the paper! We used ResNet18 not ResNet101! **Shallower model**?

Attack	Model	eps	ERM	mixup
White box attack	ResNet18	0.007	**0.5%**	1.25%
Black box attack	ResNet18	0.007	**3.5%**	%

Results After 10 Iterations

Mixup Stabilizes the training of GANs

- Train the generator as usual.
- Mix a real image and a fake image and feed the new image to the discriminator

ERM :

$$\max_g \min_d \mathop{\mathbb{E}}_{x,z} \ell(d(x), 1) + \ell(d(g(z)), 0),$$

Mixup:

$$\max_g \min_d \mathop{\mathbb{E}}_{x,z,\lambda} \ell(d(\lambda x + (1-\lambda) g(z)), \lambda).$$

Conclusion

- **mixup** is a data augmentation method that consists of only two parts: 1-**random convex combination of raw inputs**, and correspondingly, 2-**convex combination of one-hot labels.**
- Another idea that gives worse results is to mix the **feature maps** instead of mixing the the raw inputs.
- Another idea also is to mix the **inputs only** (worse results).
- Using the combination of more than 2 images doesn't provide better results (2 images is enough).
- Not only images!!

The End

Questions😄