



# mixup: BEYOND EMPIRICAL RISK MINIMIZATION

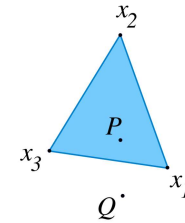
## Index

- [Abstraction](#)
- [Introduction](#)
- [Contribution](#)
- [From Empirical Risk Minimization To mixup](#)
- [Experiments](#)
  - 1. [ImageNet classification](#)
  - 2. [CIFAR-10 and CIFAR-100](#)
  - 3. [Speech Data](#)
  - 4. [Memorization of Corrupted Labels](#)
  - 5. [Robustness to Adversarial Examples](#)
  - 6. [Tabular Data](#)
  - 7. [Stabilization of Generative Adversarial Networks \(GANs\)](#)

## Abstraction

mixup은 data와 label을 **convex combination**해서 새로운 데이터를 생성

- ▼ convex combination



$$\alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n$$

where the real numbers  $\alpha_i$  satisfy  $\alpha_i \geq 0$  and  $\alpha_1 + \alpha_2 + \dots + \alpha_n = 1$ .<sup>1</sup>

점들을 선형 결합(linear combination)할 때

- 계수가 양수 + 계수의 합을 1로 제한
- (그림) 주어진 지점을 서로 연결한 도형 안에 있는 P와 같은 지점들

## Introduction

### 신경망의 2가지 특징

#### 1 ERM(Empirical Risk Minimization)

→ 학습 데이터에 대한 평균 에러 최소화 ⇒ 신경망 최적화

#### 2 SOTA는 학습 데이터의 크기에 선형적으로 비례하게 신경망의 크기가 커짐

##### ▼ SOTA



SOTA는 State-of-the-art의 약자로 사전학습된 신경망들 중 현재 최고 수준의 신경망이라는 뜻이다

### ERM 기반 학습의 단점

- classical result (Vapnik & Chervonenkis, 1971)에 따르면, 모델의 크기가 학습 데이터 수에 따라 비례해서 증가하지 않아야 ERM의 수렴이 보장 → 신경망의 2가지 특징과 모순
- (강한 규제(regularization) 방법을 써도) 학습 데이터를 기억(memorize) ⇒ 과적합 현상 발생
- ▼ memorization

모델이 학습을 진행할 때, 정답만을 기억하고 내리는 행동  
즉, 데이터 분포를 학습하는 것이 아니라 해당 데이터가 어떤 라벨에 해당하는지 기억하게 되는 것  
⇒ test distribution에 대한 generalization 불가능

## VRM(Vicinal Risk Minimization)의 등장

- ERM의 차이점
  - 학습 데이터 + 학습 데이터 근방의(vicinal) 분포까지도 학습 ⇒ 추가적인 data에 대해 결론 도출 가능
- 근방의 분포를 얻는 방법
  - 데이터 증강(Data Augmentation)을 사용
  - ⇒ VRM 기반의 학습 C 데이터 증강을 통한 학습
- 결론
  - 데이터 증강(Data augmentation) → 신경망의 일반화(generalization)에 좋다 (= 과적합 ↓)
- ▼  학습 데이터 근방의 분포로 어떻게 학습하는가
  - 가상의 데이터를 근방의 분포로부터 샘플링 ⇒ 이를 학습에 이용
  - 인접한 examples는 같은 클래스에 속한다고 가정
- ▼  Data Augmentation
  - 적은 양의 데이터를 바탕으로 다양한 알고리즘을 통해 데이터의 양을 늘리는 기술


## Contribution

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

(단,  $(x_i, y_i)$ 과  $(x_j, y_j)$ 는 training data에서 랜덤으로 추출한 example)

## mixup의 training 분포 확장 원리

- ▼  선형 보간법(linear interpolation)

통계적 혹은 실험적으로 구해진 데이터로부터, 주어진 데이터를 만족하는 근사 함수  $f(x)$ 를 구하고, 이 식을 이용하여 주어진 변수에 대한 함수값을 구하는 과정

ex. (0, 0), (1, 10), (2, 20)이 주어졌을 때, 이들에 대한 근사 함수를  $f(x) = 10x$ 로 구하고, 1.5에 대한 함수 값으로 15를 구하는 것이다.

## mixup의 장점

- 1 CIFAR-10, CIFAR100(실험2) 및 ImageNet-2012(실험1) 이미지 분류 데이터 셋에서 성능 좋음
- 2 손상된 레이블로부터 학습하거나(실험4) adversarial example에서도(실험5) 강력함
- 3 음성 데이터(실험3)와 표 형식(실험6) 데이터에 대한 일반화(generalization)를 개선하고, GAN(실험7)의 학습을 안정화하는 데 사용 가능

## CIFAR-10 실험에 필요한 소스 코드

<https://github.com/facebookresearch/mixup-cifar10>

## From Empirical Risk Minimization To mixup

### ERM 기반의 학습

$$R(f) = \int \ell(f(x), y) dP(x, y).$$


### $f \in \mathcal{F}$

- random feature vector X와 random target vector Y 사이의 관계를 설명하는 함수(즉, 딥러닝 모델)

$l$

- $f(x)$ 와  $y$ 의 차이를 모델링하는 손실 함수


⇒ 전체 학습 데이터에 대한 손실 함수의 평균을 최소화시켜서  $f(= (x,y)$  사이의 관계를 모델링하는 함수)를 찾는다

- ▼  전체 학습 데이터 분포에서 손실 함수에 대한 평균을 정의
  - 적분을 활용 ⇒ 손실 함수의 기댓값 정의


## R(f)

- 기대 위험


→ 기대 위험을 계산하기 위해서는  $P(x,y)$ 를 알고 있어야 하지만, **데이터의 분포 P**를 모른다

- ▼  학습 데이터 D로부터 데이터의 분포 P를 근사
  - 훈련 데이터를  $D=\{(x_i, y_i)\}_{i=1}^n$ , 라고 했을 때,  $(x_i, y_i) \sim P$ 를 따른다고 하자
  - D를 기반으로 **경험적 분포(Empirical Distribution)**를 정의하면,

$$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \delta(x = x_i, y = y_i),$$

- ▼  경험적 분포
  - 반복된 시행을 통해 확률 변수가 일정 값을 넘지 않을 확률을 유추하는 함수

## $\delta(x=x_i, y=y_i)$

- 다이라-델타 함수(Dirac-Delta function)
  - $(x_i, y_i)$ 에서만 1이고 나머지 점에서는 0인 함수
  - ⇒ 이를 통해서 **기대 위험(Expected Risk)**를 도출
    - ▼  기대 위험(Expected Risk)
      - 우리가 알지 못하는 실제 데이터 분포를 일반화할 때 발생하는 error의 기대치

## 기대 위험(Expected Risk)

$$R_\delta(f) = \int \ell(f(x), y) dP_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i).$$

- $R_\delta(f)$ 를 최소화 ⇒ **ERM** 기반의 학습
- 딥러닝과 같이 파라미터 수가 많은 모델을 학습할 경우, empirical risk를 최소화하는 과정에서 학습 데이터를 기억(memorize)할 수 있다 (단점)
  - VRM 기반의 학습으로 확장시킬 필요가 있다

- ▼  수식

$\delta(x=x_i, y=y_i)$ 에 의해,  $i$ 번째 항만 남고 나머지는 0으로 사라짐

## VRM 기반의 학습

데이터의 분포 P에 VRM을 적용하면, P를 아래와 같이 나타낼 수 있다 (단,  $\nu$ 는 근방의 분포)

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^n \nu(\tilde{x}, \tilde{y} | x_i, y_i),$$

## $V(\tilde{x}, \tilde{y} | x_i, y_i)$

- 근방의 분포(vicinal distribution)
  - 학습 데이터의 feature-target 쌍  $(x_i, y_i)$  근방에서 가상의 feature-target 쌍을 찾을 확률의 분포
- $(\tilde{x}, \tilde{y})$  → (Data Augmentation으로부터 만들어진) 가상의 feature-target vector
- $(x_i, y_i)$  → 학습 데이터의 feature-target 쌍

## 새로운 기대 위험

$$R_\nu(f) = \frac{1}{m} \sum_{i=1}^m \ell(f(\tilde{x}_i), \tilde{y}_i).$$

- $R_\nu(f)$  를 최소화  $\Rightarrow$  VRM 기반의 학습

## mixup

- vicinal distribution을 일반화한 식을 제시  $\Rightarrow$  mixup 데이터 증강 기법  $\subset$  VRM 학습

$$\mu(\tilde{x}, \tilde{y} | x_i, y_i) = \frac{1}{n} \sum_j \mathbb{E}_\lambda [\delta(\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)],$$

$\rightarrow$  두 데이터  $((x_i, y_i), (x_j, y_j))$ 를  $\lambda$ 를 이용해서 적절하게 섞은 후 모델의 학습에 활용

$\rightarrow \lambda(\text{lambda})$  값은 **Beta(a,a) 분포**를 따른다

- mixup 모델은 모든 학습 데이터셋을 통해 vicinity를 구함  
(단,  $(x_i, y_i)$ 과  $(x_j, y_j)$ 는 training data에서 랜덤으로 추출한 example)

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j, \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j,\end{aligned}$$

$\rightarrow$  vicinal distribution에서 virtual feature-target vector를 추출

### ▶ 베타 분포

확률에 대한 확률 분포

성공 횟수( $\alpha-1$ )와 실패 횟수( $\beta-1$ )가 고정된 것이고, 확률이 확률변수

cf. 이항 분포

확률  $p$ 가 고정된 것이고, 성공 횟수 및 실패 횟수가 확률변수

$$\begin{aligned}\text{beta} &: x^{\alpha-1}(1-x)^{\beta-1} \\ \text{binomial} &: p^x(1-p)^{n-x}\end{aligned}$$

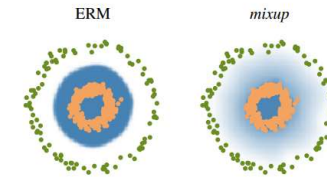
Figure 1 (a)

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of mixup training in PyTorch.

- Pytorch로 구현한 코드

Figure 1 (b)



(b) Effect of mixup ( $\alpha = 1$ ) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates  $p(y = 1|x)$ .

- ERM (좌측) VS. mixup (우측)
    - $\rightarrow$  파란색 부분은 해당 데이터  $x$ 가 주어졌을 때, 클래스가 1일 확률
    - $\rightarrow$  ERM을 보면 파란색 부분이 클래스 1가 가까운 것과 먼 것 사이의 차이를 나타내지 못함
    - $\rightarrow$  mixup은 가까운 부분은 더 짙은 파란색으로 나타내어, uncertainty를 부드럽게 (smoother) 표시
    - $\rightarrow$  ERM은 두 클래스 간의 decision boundary가 뚜렷  $\rightarrow$  mixup은 decision boundary가 부드러움
- $\Rightarrow$  mixup이 ERM에 비해 과적합이 덜 발생한다 + mixup이 **regularization** 역할

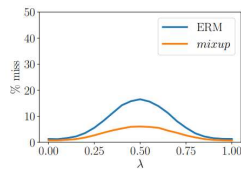
## regularization

훈련 데이터에만 특화되지 않고 일반화(generalization)가 가능하도록 규제(penalty)를 가하는 기법

## what is mixup doing?

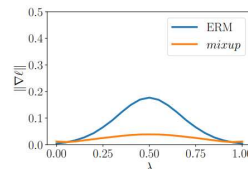
- 모델이 training data 간에 **선형적인 양상 (linear behavior)**을 가지게끔 해준다  
 $\Rightarrow$  학습 데이터가 아닌 데이터에서 예상치 못한 결과(undesirable oscillations)를 내는 것을 줄여준다  
 (생각) 수식에서  $x$ 와  $y$ 를  $\lambda$ 와 함께 **선형의 식**으로 표현한 부분을 말하는 것 같다  $\rightarrow$  선형의 식으로 표현해서 학습 데이터 외에 새로운 데이터를 만들어 낼 수 있음  $\Rightarrow$  과적합 피하게 된다

Figure 2 (a)



(a) Prediction errors in-between training data. Evaluated at  $x = \lambda x_i + (1 - \lambda)x_j$ , a prediction is counted as a "miss" if it does not belong to  $\{y_i, y_j\}$ . The model trained with *mixup* has fewer misses.

Figure 2 (b)



(b) Norm of the gradients of the model w.r.t. input in-between training data, evaluated at  $x = \lambda x_i + (1 - \lambda)x_j$ . The model trained with *mixup* has smaller gradient norms.

- model prediction
  - 성능 : ERM < mixup
- 학습 데이터 간의 **gradient norm**  $\rightarrow$  작게 나타날수록 더 안정적인 학습
  - 안정적인 학습 : ERM < mixup

## gradient norm

많이 틀릴 수록 기울기가 가팔라지므로, 크게 나타날수록 학습이 불안정

## Experiments

## 1. ImageNet classification

### 학습 데이터 셋

- 1,000개 클래스 + 130만 개 학습 이미지 + 50,000개의 테스트 이미지

### Data Augmentation

- 크기(size) 및 가로 세로(aspect) 비율 왜곡
- 랜덤 크롭(random crop)
- 수평 플립(horizontal flips)

### learning rate

- 처음 5 epochs 동안) 0.1  $\rightarrow$  0.4로 선형적으로(linearly) 증가
- 90 epochs 동안 training 시) 30, 60, 80 epochs 이후에 0.1배
- 200 epochs 동안 training 시) 60, 120, 180 epochs 이후에 0.1배

### 테스트 데이터 셋

- 224 x 224의 크기로 중간 부분이 소실된 이미지로 테스트

Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM (Goyal et al., 2017)	90	23.5	-
	<i>mixup</i> $\alpha = 0.2$	90	<b>23.3</b>	<b>6.6</b>
ResNet-101	ERM (Goyal et al., 2017)	90	22.1	-
	<i>mixup</i> $\alpha = 0.2$	90	<b>21.5</b>	<b>5.6</b>
ResNeXt-101 32*4d	ERM (Xie et al., 2016)	100	21.2	-
	ERM	90	21.2	5.6
	<i>mixup</i> $\alpha = 0.4$	90	<b>20.7</b>	<b>5.3</b>
ResNeXt-101 64*4d	ERM (Xie et al., 2016)	100	20.4	5.3
	<i>mixup</i> $\alpha = 0.4$	90	<b>19.8</b>	<b>4.9</b>
ResNet-50	ERM	200	23.6	7.0
	<i>mixup</i> $\alpha = 0.2$	200	<b>22.1</b>	<b>6.1</b>
ResNet-101	ERM	200	22.0	6.1
	<i>mixup</i> $\alpha = 0.2$	200	<b>20.8</b>	<b>5.4</b>
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	<i>mixup</i> $\alpha = 0.4$	200	<b>20.1</b>	<b>5.0</b>

Table 1: Validation errors for ERM and *mixup* on the development set of ImageNet-2012.

### 실험 내용

- mixup과 ERM을 사용  $\rightarrow$  SOTA ImageNet-2012 분류 모델을 학습

### 실험 결과

- hyperparameter  $\alpha$

→ 0.1~0.4에서 우수한 성능 + 이보다 더 클 경우, underfitting

- 90 epochs

- ResNet-101과 ResNeXt-101 모델은 ResNet-50 (0.2%)보다 훨씬 개선(0.5% ~ 0.6%)됨

- 200 epochs

- mixup은 ResNet-50의 top-1 error가 90 epochs에 비해 1.2% 더 감소하지만, ERM은 ResNet-50의 top-1 error가 90 epochs와 비슷하게 유지

⇒ ERM과 비교했을 때, mixup은 **higer capacities**와 longer training run의 효과

#### ▼ higher capacities

→ higher capacity (= parameter 개수 = hidden layer 개수) ⇒ 더 high-level feature를 추출할 수 있게 된다

→ 구체적으로 hidden layer가 쌓이는 과정에서 비선형 함수를 사용하며, 함수들이 합성되면서 훨씬 더 복잡한 함수를 표현 가능 ⇒ 더 high-level feature 학습 가능 ⇒ 더 어려운 문제를 해결 가능 ⇒ representative(표현력)이 더 좋아짐

## 2. CIFAR-10 and CIFAR-100

- mixup의 generalization performance를 평가하기 위해서 실험

#### ▼ setting

- 200 epochs + 128 minibatch

#### ▼ learning rate

- 0.1에서 시작
- 100, 150 epochs에서 0.1배 (단, WideResNet은 제외)
- WideResNet은 60, 120, 180 epochs에서 0.1배

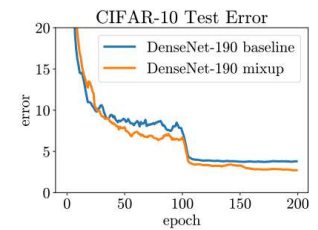
Figure 3 (a)

Figure 3 (b)

Dataset	Model	ERM	mixup
CIFAR-10	PreAct ResNet-18	5.6	<b>4.2</b>
	WideResNet-28-10	3.8	<b>2.7</b>
	DenseNet-BC-190	3.7	<b>2.7</b>
CIFAR-100	PreAct ResNet-18	25.6	<b>21.1</b>
	WideResNet-28-10	19.4	<b>17.5</b>
	DenseNet-BC-190	19.0	<b>16.8</b>

(a) Test errors for the CIFAR experiments.

- 성능 : ERM < mixup



(b) Test error evolution for the best ERM and mixup models.

- ERM과 mixup은 비슷한 속도로 각각의 best test error로 수렴

## 3. Speech Data

- 음성 인식 실험을 수행

#### ▼ 데이터 셋

- Google commands dataset
- 65,000개의 발화를 포함
- 각 발화는 약 1초 + 30개 클래스(수천 명의 사람이 발음하는 yes/no/down/left 같은 음성 명령)

#### ▼ 전처리

- 원래 파형에서 16kHz의 샘플링 속도로 정규화된 spectrogram을 추출
- 160 × 101로 크기를 동일하게 하기 위해 spectrogram을 zero-padding
- 음성 데이터의 경우 파형과 spectrogram에서 mixup을 적용하는 것이 좋으나, 여기서는 spectrogram에 mixup을 적용

#### ▼ setting

- 30 epochs + 100 minibatch

#### ▼ learning rate

- $3 \times 10^{-3}$ 에서 시작

- 10 epochs 마다 0.1배

Model	Method	Validation set	Test set
LeNet	ERM	<b>9.8</b>	<b>10.3</b>
	<i>mixup</i> ( $\alpha = 0.1$ )	10.1	10.8
	<i>mixup</i> ( $\alpha = 0.2$ )	10.2	11.3
VGG-11	ERM	5.0	4.6
	<i>mixup</i> ( $\alpha = 0.1$ )	4.0	3.8
	<i>mixup</i> ( $\alpha = 0.2$ )	<b>3.9</b>	<b>3.4</b>

Figure 4: Classification errors of ERM and *mixup* on the Google commands dataset.

## 실험 결과

- 용량이 더 큰 모델인 VGG-11에서 *mixup*이 ERM을 능가

## 4. Memorization of Corrupted Labels

- memorization을 검증하기 위해서 손상된(corruption) label로 학습

### 가정

- hyperparameter  $\alpha$ 가 커질수록 memorization이 어려워진다

( $\because$   $\alpha$ 가 커질수록 virtual example을 training data로부터 멀리에서 생성하기 때문에)

### ▼ 데이터 셋

- 각 레이블의 20%, 50%, 80%에 랜덤 노이즈 발생
- 모든 테스트 라벨은 그대로 유지

### ▼ 비교할 모델

- Dropout  $\rightarrow$  손상된 label을 사용한 학습에 대해 SOTA 모델
- *mixup* / Dropout / *mixup*+Dropout / ERM을 비교

### ▼ setting

- *mixup*)  $\alpha \in \{1, 2, 8, 32\}$ 을 선택
- Dropout)  $p \in \{0.5, 0.7, 0.8, 0.9\}$ 을 선택

- *mixup*+Dropout)  $\alpha \in \{1, 2, 4, 8\}$ 과  $p \in \{0.3, 0.5, 0.7\}$ 을 선택

Label corruption	Method	Test error		Training error	
		Best	Last	Real	Corrupted
20%	ERM	12.7	16.6	0.05	0.28
	ERM + dropout ( $p = 0.7$ )	8.8	10.4	5.26	83.55
	<i>mixup</i> ( $\alpha = 8$ )	<b>5.9</b>	6.4	2.27	86.32
	<i>mixup</i> + dropout ( $\alpha = 4, p = 0.1$ )	6.2	<b>6.2</b>	1.92	85.02
50%	ERM	18.8	44.6	0.26	0.64
	ERM + dropout ( $p = 0.8$ )	14.1	15.5	12.71	86.98
	<i>mixup</i> ( $\alpha = 32$ )	11.3	12.7	5.84	85.71
	<i>mixup</i> + dropout ( $\alpha = 8, p = 0.3$ )	<b>10.9</b>	<b>10.9</b>	7.56	87.90
80%	ERM	36.5	73.9	0.62	0.83
	ERM + dropout ( $p = 0.8$ )	30.9	35.1	29.84	86.37
	<i>mixup</i> ( $\alpha = 32$ )	25.3	30.9	18.92	85.44
	<i>mixup</i> + dropout ( $\alpha = 8, p = 0.3$ )	<b>24.0</b>	<b>24.8</b>	19.70	87.67

Table 2: Results on the corrupted label experiments for the best models.

## 표

- best error와 200 epochs의 error를 기록
- memorization 정도를 정량화하기 위해서 실제 label과 손상된 label에 대한 마지막 epochs에서의 error도 평가

## 실험 결과

- 작은 learning rate로 진행되면서, ERM 모델은 손상된 label에 과적합하기 시작함
- 큰 확률  $p$ (ex. 0.7, 0.8)를 사용하는 경우, Dropout으로 과적합을 줄일 수 있음
- 큰  $\alpha$ (ex. 8, 32)를 사용하는 경우, *mixup*은 best/last error 모두 Dropout보다 성능이 좋음
- *mixup*+Dropout 이 가장 좋은 성능을 보임

## 5. Robustness to Adversarial Examples

### Adversarial Example

$\rightarrow$  모델의 성능을 저하시키기 위해, 틀린 클래스로 인식하도록 작은 노이즈를 냄 (단, 노이즈를 포함한 사진이 사람이 보기에는 원래 사진과 구분되지 않아야 함)

### ▼ setting

- **FGSM(Fast Gradient Sign Method)** / Iterative-FGSM(I-FGSM) 방법을 사용  
 $\rightarrow$  Adversarial Example 생성

- I-FGSM의 경우 동일한 step 크기로 10회 반복
- 모든 픽셀에 대해 최대 4까지의 노이즈(perbutions)를 허용
- ▼ FGSM
 

입력 이미지에 대한 **gradient(기울기) 정보**를 추출하고, 이를 왜곡하여 원본 이미지에 더하는 방법

반복된 학습  $X$  + 공격 목표를 정할 수  $X$ (= **non-targeted** 방식) + white box attack
- ▼ gradient(기울기)
 

모델이 학습할 때 각 픽셀이 미치는 영향
- ▼ targeted vs. non-targeted
 

targeted → 원하는 정답으로 유도 가능

non-targeted → 원하는 정답으로 유도 불가능
- ▼ one-shot vs. iterative
 

one-shot → 잡음을 생성하기 위한 반복된 학습(최적화)가 필요  $X$  (ex. FGSM)

iterative → 잡음을 생성하기 위한 반복된 학습(최적화)가 필요  $O$  (ex. I-FGSM)
- ▼ White-box attack
 

모델의 정보(ex. gradient)를 토대로 노이즈 생성
- ▼ Black-box attack
 

모델 정보 없이 노이즈 생성

Metric	Method	FGSM	I-FGSM
Top-1	ERM	90.7	99.9
	<i>mixup</i>	<b>75.2</b>	99.6
Top-5	ERM	63.1	93.4
	<i>mixup</i>	<b>49.1</b>	95.8

(a) White box attacks.

Metric	Method	FGSM	I-FGSM
Top-1	ERM	57.0	57.3
	<i>mixup</i>	<b>46.0</b>	<b>40.9</b>
Top-5	ERM	24.8	18.1
	<i>mixup</i>	<b>17.4</b>	<b>11.8</b>

(b) Black box attacks.

Table 3: Classification errors of ERM and *mixup* models when tested on adversarial examples.

## 실험 결과

- (a) FGSM의 경우, mixup 모델이 Top-1 error에서 ERM 모델보다 2.7배 더 강력하다
- (b) FGSM의 경우 mixup 모델이 Top-1 error에서 ERM 모델보다 1.25배 더 강력하다

- (a) mixup과 ERM 모두 I-FGSM에 강하지 않지만, (b) I-FGSM에서 mixup은 ERM보다 40% 더 강력하다

ex. 2.7배 강력하다 =  $(100 - \text{Top-1 mixup error}(= 75.2)) / (100 - \text{Top-1 ERM error}(= 90.7)) = 2.7$

- white box attack에 대해 강력한 정도 : ERM < mixup
- black box attack에 대해 강력한 정도 : ERM < mixup

## 6. Tabular Data

- 비이미지 데이터에 대한 mixup의 성능을 추가로 탐구하기 위해 6가지 임의 분류 문제 실험

### setting

- minibatch size 16 + 10 이상의 epochs

Dataset	ERM	<i>mixup</i>	Dataset	ERM	<i>mixup</i>
Abalone	74.0	73.6	Htru2	2.0	2.0
Arcene	57.6	<b>48.0</b>	Iris	21.3	<b>17.3</b>
Arrhythmia	56.6	<b>46.3</b>	Phishing	16.3	15.2

Table 4: ERM and *mixup* classification errors on the UCI datasets.

### 실험 결과

- 성능 : ERM < mixup

## 7. Stabilization of Generative Adversarial Networks (GANs)

### GAN의 이미지 생성 원리

- GAN에서 생성자(generator)와 판별자(discriminator)는 분포  $P$ 를 모델링하기 위해 경쟁한다
  - 생성자( $g$ )는 노이즈 벡터  $z \sim Q$ 를 실제 샘플  $x \sim P$ 와 유사한 가짜 샘플  $g(z)$ 로 변환하기 위해 경쟁



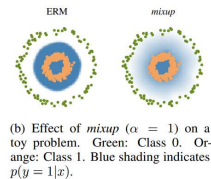
- 판별자(d)는 실제 표본 x와 가짜 표본 g(z)를 구별하기 위해 경쟁
- 수학적으로, GAN을 학습시키는 것은 최적화 문제를 해결하는 것과 같다

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

- $\ell$ 은 이진 교차 엔트로피 손실(binary cross entropy loss)
- 판별자는 생성자에 vanishing gradient 문제를 일으키므로, min-max 방정식을 해결하는 것은 매우 어려움
- ▼ 이진 교차 엔트로피 손실(binary cross entropy loss)  
이진 분류(= 데이터가 주어졌을 때, 해당 데이터를 두 가지 정답 중 하나로 분류하는 것)의 목적 함수로 사용
- ▼ “경찰과 위조지폐범” 비유  
위조지폐범(generator)은 위조지폐를 진짜 지폐와 거의 비슷하게 만듭니다. 경찰(discriminator)은 이 지폐가 진짜 지폐인지 위조지폐인지 구분하려 합니다. 위조지폐범과 경찰은 적대적인 관계에 있고, 위조지폐범은 계속 위조지폐를 생성하고 경찰은 진짜를 찾아내려고 하는 쫓고 쫓기는 과정이 반복됩니다.  
⇒ 위조지폐범(Generator)과 경찰(Discriminator)가 서로 위조지폐를 생성하고 구분하는 것을 반복하는 minmax game

## mixup 모델이 GAN 학습을 안정화

- mixup이 판별자의 gradient 정규화(regularizer) 역할 (cf. Figure 1 (b)) → GAN의 학습을 안정화
- 판별자의 smoothness(부드러움)은 생성자에 안정적인 gradient 정보를 보장 → vanishing gradient 문제 해결
- ▼ smoothness가 안정적인 gradient 정보를 보장  
(생각) ERM처럼 클래스 1/클래스 0으로 뚜렷하게 나뉘면 오차가 급격하게 줄어드는데 반해, smoothness를 가진 mixup의 경우, 오차가 급격하게 줄어들지 않으므로 vanishing gradient 문제를 해결할 수 있다
- ▼ vanishing gradient



오차가 크게 줄어들어 학습(ex. 가중치 업데이트)이 되지 않는 현상

## mixup을 적용한 GANs

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

Figure 5

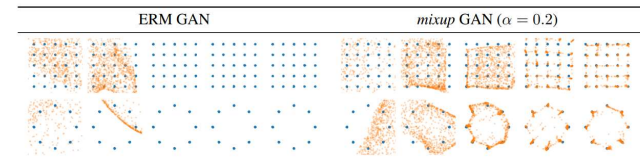


Figure 5: Effect of *mixup* on stabilizing GAN training at iterations 10, 100, 1000, 10000, and 20000.

## setting

- 120,000 minibatch + minibatch size 128
- 모든 생성자의 학습을 시작하기 전에 판별자가 5회 반복해서 학습된다

## 실험 결과

- 두 개의 데이터 셋(파란색 샘플)을 모델링할 때, GAN(주황색 샘플)의 학습이 안정화된 정도 : ERM < mixup

## 참고자료

- [선형결합?](#)
- [convex combination? \(1\).](#)
- [convex combination? \(2\).](#)
- [convex combination? \(3\).](#)
- [SOTA?](#)
- [mixup 논문? \(1\).](#)
- [선형 보간법?](#)
- [경험적 분포?](#)
- [베타 분포?](#)
- [regularization?](#)
- [gradient norm?](#)
- [higher capacities? \(1\).](#)

- mixup 논문? (2).
- mixup 논문? (3).
- mixup 논문? (4).
- mixup 논문? (5).
- Data Augmentation?
- higher capacities? (2).
- white box attack vs. black box attack?
- FGSM? (1).
- FGSM? (2).
- GAN의 이미지 생성 원리?
- 이진 교차 엔트로피?