目录

Introduction: 数据混合起来--MixUp!

Abstract

1. 深度神经网络非常强大，但对错误标签的记忆及对对抗样本的敏感性不理想。
2. 在成对样本及其标签的凸组合上训练神经网络。训练样本之间的线性表达。
3. 减少对错误标签的记忆，增加对于对抗样本的鲁棒性。

神经网络最小化均方误差，遵从ERM；神经网络的大小与训练样本的数量呈线性关系。

矛盾的地方：经典机器学习理论告诉我们，只要学习机的规模不随着训练数据数量的增加而增加，那么ERM的收敛性就是可以得到保证的。

即使在强正则化情况下，ERM 也允许大规模神经网络去记忆（而不是泛化）训练数据。

- 神经网络使用ERM 方法训练后，在训练分布之外的样本（对抗样本）上验证时会极大地改变预测结果。
- 即在测试分布与训练数据略有不同时，ERM 方法已不具有良好的解释和泛化性能。

Vicinal Risk Minimization(VRM)领域风险最小化原则，从训练样本邻域中提取附加的虚拟样本以扩充对训练分布的支持。

- 数据增强可以提高泛化能力，但这一过程依赖于数据集。
- 数据增强假定领域内样本都是同一类，且没有对不同类不同样本之间领域关系进行建模。
- 其他方法没有对标签进行相应的处理或者只对标签进行平滑处理。没有建立输入和输出的关系。

Main contribution of mixup

简而言之：mixup构建了虚拟的训练样本。

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

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$.

Mixup通过结合先验知识，即特征向量的线性插值应导致相关的标签的线性插值，来扩展训练分布。贡献在于提出一种通用的领域分布，称作mixup。

The contribution of this paper is to propose a generic vicinal distribution, called *mixup*:

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

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. In a nutshell, sampling from the *mixup* vicinal distribution produces virtual feature-target vectors

重点：

Mixup超参数α控制了在特征-目标向量之间插值的强度，当α趋向于0时恢复为ERM原则。

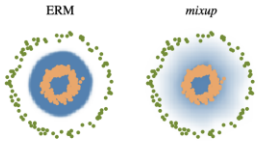
Main contribution of mixup

Mixup仅需要几行代码即可实现。图1显示决策边界从一个类到另一个类线性的转变，提供了一个更平滑的不确定性估计。

Mixup超参数α控制了在特征-目标向量之间插值的强度，当α趋向于0时恢复为ERM原则。

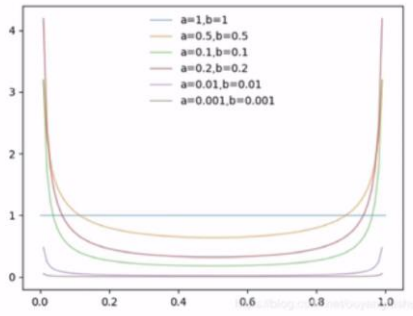
```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



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

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \to 0$.

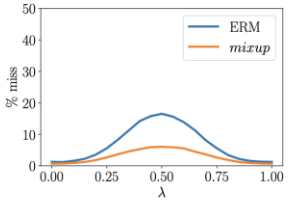


tips

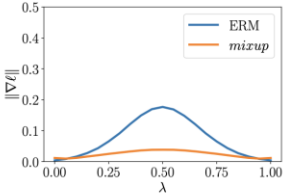
- 三个和三个以上从Dirichlet分布采样的权重样本的凸组合的性能表现也不错，但计算成本增加；
- 当前的mixup使用了一个单一的loader获取minibatch，对其随机打乱后，mixup对同一个minibatch内的数据做混合。这样的策略和在整个数据集随机打乱效果是一样的，而且还减少了IO的开销。
- 在同种标签的数据中使用mixup不会造成结果的显著增强。

Mixup insight

mixup做了什么？



(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.



(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.

Figure 2: *mixup* leads to more robust model behaviors in-between the training data.

两个模型有相同的结构，使用相同的训练过程，在同一个从训练数据里随机抽样而来的样本上来评估。
用mixup训练的模型在预测训练数据之间的数据时更稳定。
对离散样本空间进行连续化，提高邻域内的平滑性。

一种数据增强方式，在不同类别之间学会一种线性的过度，这样会导致模型在预测过程中不容易在某些类别之间发生震荡。
这种线性建模减少了在预测训练样本以外的数据时的不适应性。
从奥卡姆剃刀的原理出发，线性是一个很好的归纳偏见。

Experiments Results

IMAGENET CLASSICATION

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

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

MEMORIZATION OF CORRUPTED LABELS

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
	mixup ($\alpha = 8$)	**5.9**	6.4	2.27	86.32
	mixup + dropout ($\alpha = 4, p = 0.1$)	6.2	**6.2**	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
	mixup ($\alpha = 32$)	11.3	12.7	5.84	85.71
	mixup + dropout ($\alpha = 8, p = 0.3$)	**10.9**	**10.9**	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
	mixup ($\alpha = 32$)	25.3	30.9	18.92	85.44
	mixup + dropout ($\alpha = 8, p = 0.3$)	**24.0**	**24.8**	19.70	87.67

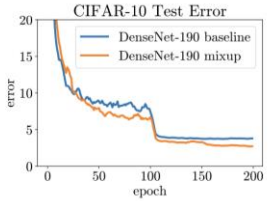
Table 2: Results on the corrupted label experiments for the best models.

Experiments Results

CIFAR-10 AND CIFAR-100

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

(a) Test errors for the CIFAR experiments.



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

Figure 3: Test errors for ERM and *mixup* on the CIFAR experiments.

STABILIZATION OF GAN

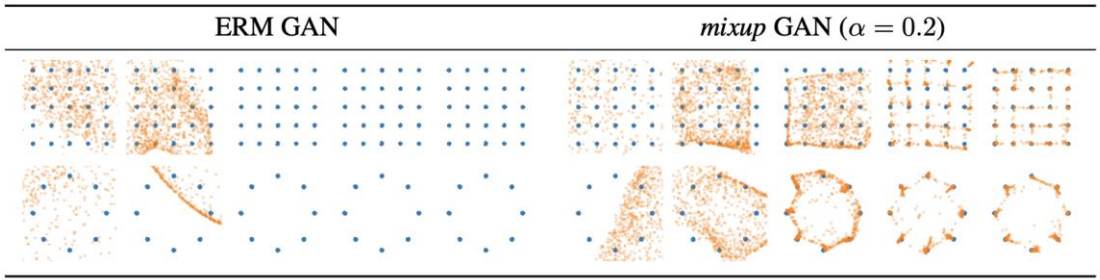


Figure 5: Effect of *mixup* on stabilizing GAN training at iterations 10, 100, 1000, 10000, and 20000.

Experiments Results

ABLATION STUDIES

Method	Specification	Modified		Weight decay	
		Input	Target	10^{-4}	5×10^{-4}
ERM		✗	✗	5.53	5.18
mixup	AC + RP	✓	✓	**4.24**	4.68
	AC + KNN	✓	✓	4.98	5.26
mix labels and latent representations (AC + RP)	Layer 1	✓	✓	4.44	**4.51**
	Layer 2	✓	✓	4.56	4.61
	Layer 3	✓	✓	5.39	5.55
	Layer 4	✓	✓	5.95	5.43
	Layer 5	✓	✓	5.39	5.15
mix inputs only	SC + KNN (Chawla et al., 2002)	✓	✗	5.45	5.52
	AC + KNN	✓	✗	5.43	5.48
	SC + RP	✓	✗	5.23	5.55
	AC + RP	✓	✗	5.17	5.72
label smoothing (Szegedy et al., 2016)	$\epsilon = 0.05$	✗	✓	5.25	5.02
	$\epsilon = 0.1$	✗	✓	5.33	5.17
	$\epsilon = 0.2$	✗	✓	5.34	5.06
mix inputs + label smoothing (AC + RP)	$\epsilon = 0.05$	✓	✓	5.02	5.40
	$\epsilon = 0.1$	✓	✓	5.08	5.09
	$\epsilon = 0.2$	✓	✓	4.98	5.06
	$\epsilon = 0.4$	✓	✓	5.25	5.39
add Gaussian noise to inputs	$\sigma = 0.05$	✓	✗	5.53	5.04
	$\sigma = 0.1$	✓	✗	6.41	5.86
	$\sigma = 0.2$	✓	✗	7.16	7.24

Table 5: Results of the ablation studies on the CIFAR-10 dataset. Reported are the median test errors of the last 10 epochs. A tick (✓) means the component is different from standard ERM training, whereas a cross (✗) means it follows the standard training practice. AC: mix between all classes. SC: mix within the same class. RP: mix between random pairs. KNN: mix between k-nearest neighbors (k=200). Please refer to the text for details about the experiments and interpretations.

From ERM to Mixup(理论推导)

理论推导

2 From Empirical Risk Minimization to *mixup*

In supervised learning, we are interested in finding a function $f \in \mathcal{F}$ that describes the relationship between a random feature vector X and a random target vector Y, which follow the joint distribution $P(X, Y)$. To this end, we first define a loss function ℓ that penalizes the differences between predictions $f(x)$ and actual targets y, for examples $(x, y) \sim P$. Then, we minimize the average of the loss function ℓ over the data distribution P, also known as the *expected risk*:

$$R(f) = \int \ell(f(x), y) \mathrm{d}P(x, y).$$

Unfortunately, the distribution P is unknown in most practical situations. Instead, we usually have access to a set of training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $(x_i, y_i) \sim P$ for all $i = 1, \dots, n$. Using the training data $\mathcal{D}$, we may approximate P by the *empirical distribution*

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

where $\delta(x = x_i, y = y_i)$ is a Dirac mass centered at (x_i, y_i). Using the empirical distribution P_δ, we can now approximate the expected risk by the *empirical risk*:

$$R_\delta(f) = \int \ell(f(x), y) \mathrm{d}P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i$$

However, the naïve estimate P_δ is one out of many possible choices to approximate the true distribution P. For instance, in the *Vicinal Risk Minimization* (VRM) principle (Chapelle et al., 2000), the distribution P is approximated by

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

where ν is a *vicinity distribution* that measures the probability of finding the *virtual* feature-target pair $(\tilde{x}, \tilde{y})$ in the *vicinity* of the training feature-target pair (x_i, y_i). In particular, Chapelle et al. (2000) considered Gaussian vicinities $\nu(\tilde{x}, \tilde{y} | x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2) \delta(\tilde{y} = y_i)$, which is equivalent to augmenting the training data with additive Gaussian noise. To learn using VRM, we sample the vicinal distribution to construct a dataset $\mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$, and minimize the *empirical vicinal risk*:

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

The contribution of this paper is to propose a generic vicinal distribution, called *mixup*:

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

半监督学习背景及假设

研究背景及目的

合理利用更多的无标注数据，辅助监督学习。虽然只有input，没有label，但它的分布，却可以告诉我们一些事情。

Before presenting an overview of the techniques of SSL, we first introduce the notations. To illustrate the DSSL framework, we limit our focus on the single-label classification tasks which are simple to describe and implement. We refer interested readers to [32], [33], [34] for multi-label classification tasks. Let $X = \{X_L, X_U\}$ denote the entire data set, including a small labeled subset $X_L = \{x_i\}_{i=1}^{L}$ with labels $Y_L = (y_1, y_2, \ldots, y_L)$ and a large scale unlabeled subset $X_U = \{(x_i)\}_{i=1}^{U}$, and generally we assume $L \ll U$. We assume that the dataset contains K classes and the first L examples within X are labeled by $\{y_i\}_{i=1}^{L} \in (y^1, y^2, \ldots, y^K)$. Formally, SSL aims to solve the following optimization problem,

$$\min_{\theta} \underbrace{\sum_{x \in X_L, y \in Y_L} \mathcal{L}_s(x, y, \theta)}_{\text{supervised loss}} + \alpha \underbrace{\sum_{x \in X_U} \mathcal{L}_u(x, \theta)}_{\text{unsupervised loss}} + \beta \underbrace{\sum_{x \in X} \mathcal{R}(x, \theta)}_{\text{regularization}}, \tag{1}$$

low-density假设

smoothness假设

MixMatch: A Holistic Approach to Semi-Supervised Learning

Introduction: 全家桶MixMatch!

Related work

1. Consistency Regularization

方案1：自洽正则化对未标记数据进行数据增强，产生的新数据输入分类器，预测结果应保持自洽。即同一个数据增强产生的样本，模型预测结果应保持一致。此规则被加入到损失函数中

$$\left\| p_{model}\left(y \mid Augment(x);\theta\right) - p_{model}\left(y \mid Augment(x);\theta\right) \right\|_2^2$$

2. Entropy Minimization

方案2：最小化熵的做法就是许多半监督学习方法都基于一个共识，即分类器的分类边界不应该穿过边际分布的高密度区域。因此强迫分类器对未标记数据作出低熵预测。实现方法是在损失函数中增加一项，最小化 $p_{model}(y|x)$ 对应的熵

3. Traditional Regularization

方案3：传统正则化的做法就是为了让模型泛化能力更好，一般的做法对模型参数做 L2 正则化进行权重衰减

主要步骤：

diagram

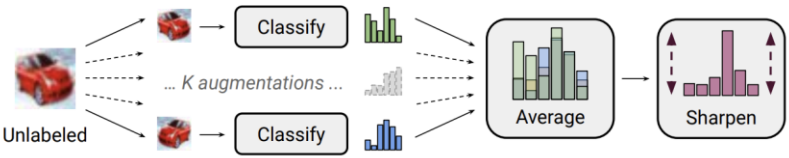


Figure 1: Diagram of the label guessing process used in MixMatch. Stochastic data augmentation is applied to an unlabeled image K times, and each augmented image is fed through the classifier. Then, the average of these K predictions is "sharpened" by adjusting the distribution's temperature. See algorithm 1 for a full description.

method

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha) \tag{2}$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta)) \tag{3}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2 \tag{4}$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} \tag{5}$$

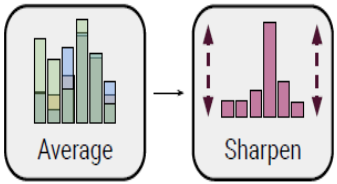
主要步骤：

- **数据增广** crop, flip
- **标签猜测**

$$\bar{q}_b = \frac{1}{K}\sum_{k=1}^{K} \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$$

- **锐化（sharpening）**

$$\mathrm{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$



- **数据混合(Mix-up)**

$$\begin{aligned}
\lambda &\sim \mathrm{Beta}(\alpha, \alpha) \\
\lambda' &= \max(\lambda, 1-\lambda) \\
x' &= \lambda' x_1 + (1-\lambda') x_2 \\
p' &= \lambda' p_1 + (1-\lambda') p_2
\end{aligned}$$

伪代码：

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \mathrm{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \mathrm{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \mathrm{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \mathrm{Shuffle}\big(\mathrm{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\mathrm{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\mathrm{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

实验结果：

B.1 CIFAR-10

Training the same model with supervised learning on the entire 50000-example training set achieved an error rate of 4.13%.

Methods/Labels	250	500	1000	2000	4000
PiModel	53.02 ± 2.05	41.82 ± 1.52	31.53 ± 0.98	23.07 ± 0.66	17.41 ± 0.37
PseudoLabel	49.98 ± 1.17	40.55 ± 1.70	30.91 ± 1.73	21.96 ± 0.42	16.21 ± 0.11
Mixup	47.43 ± 0.92	36.17 ± 1.36	25.72 ± 0.66	18.14 ± 1.06	13.15 ± 0.20
VAT	36.03 ± 2.82	26.11 ± 1.52	18.68 ± 0.40	14.40 ± 0.15	11.05 ± 0.31
MeanTeacher	47.32 ± 4.71	42.01 ± 5.86	17.32 ± 4.00	12.17 ± 0.22	10.36 ± 0.25
MixMatch	11.08 ± 0.87	9.65 ± 0.94	7.75 ± 0.32	7.03 ± 0.15	6.24 ± 0.06

Table 5: Error rate (%) for CIFAR10.

B.2 SVHN

Training the same model with supervised learning on the entire 73257-example training set achieved an error rate of 2.59%.

Methods/Labels	250	500	1000	2000	4000
PiModel	17.65 ± 0.27	11.44 ± 0.39	8.60 ± 0.18	6.94 ± 0.27	5.57 ± 0.14
PseudoLabel	21.16 ± 0.88	14.35 ± 0.37	10.19 ± 0.41	7.54 ± 0.27	5.71 ± 0.07
Mixup	39.97 ± 1.89	29.62 ± 1.54	16.79 ± 0.63	10.47 ± 0.48	7.96 ± 0.14
VAT	8.41 ± 1.01	7.44 ± 0.79	5.98 ± 0.21	4.85 ± 0.23	4.20 ± 0.15
MeanTeacher	6.45 ± 2.43	3.82 ± 0.17	3.75 ± 0.10	3.51 ± 0.09	3.39 ± 0.11
MixMatch	3.78 ± 0.26	3.64 ± 0.46	3.27 ± 0.31	3.04 ± 0.13	2.89 ± 0.06

Table 6: Error rate (%) for SVHN.

Introduction: 全家桶MixMatch!

Related work

1. Consistency Regularization

方案1：自洽正则化对未标记数据进行数据增强，产生的新数据输入分类器，预测结果应保持自洽。即同一个数据增强产生的样本，模型预测结果应保持一致。此规则被加入到损失函数中

$$\left\| p_{model}\left(y \mid Augment(x);\theta\right) - p_{model}\left(y \mid Augment(x);\theta\right) \right\|_2^2$$

2. Entropy Minimization

方案2：最小化熵的做法就是许多半监督学习方法都基于一个共识，即分类器的分类边界不应该穿过边际分布的高密度区域。因此强迫分类器对未标记数据作出低熵预测。实现方法是在损失函数中增加一项，最小化 $p_{model}(y|x)$ 对应的熵

3. Traditional Regularization

方案3：传统正则化的做法就是为了让模型泛化能力更好，一般的做法对模型参数做 L2 正则化进行权重衰减

主要步骤：

diagram

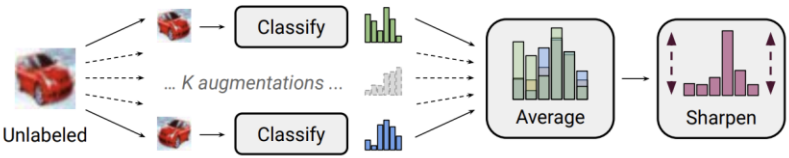


Figure 1: Diagram of the label guessing process used in MixMatch. Stochastic data augmentation is applied to an unlabeled image K times, and each augmented image is fed through the classifier. Then, the average of these K predictions is "sharpened" by adjusting the distribution's temperature. See algorithm 1 for a full description.

method

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha) \tag{2}$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta)) \tag{3}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2 \tag{4}$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} \tag{5}$$

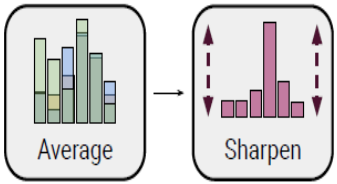
主要步骤：

- **数据增广** crop, flip
- **标签猜测**

$$\bar{q}_b = \frac{1}{K}\sum_{k=1}^{K} \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$$

- **锐化（sharpening）**

$$\mathrm{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$



- **数据混合(Mix-up)**

$$\begin{aligned}
\lambda &\sim \mathrm{Beta}(\alpha, \alpha) \\
\lambda' &= \max(\lambda, 1-\lambda) \\
x' &= \lambda' x_1 + (1-\lambda') x_2 \\
p' &= \lambda' p_1 + (1-\lambda') p_2
\end{aligned}$$

伪代码：

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \mathrm{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \mathrm{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \mathrm{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \mathrm{Shuffle}\big(\mathrm{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\mathrm{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\mathrm{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

实验结果：

B.1 CIFAR-10

Training the same model with supervised learning on the entire 50000-example training set achieved an error rate of 4.13%.

Methods/Labels	250	500	1000	2000	4000
PiModel	53.02 ± 2.05	41.82 ± 1.52	31.53 ± 0.98	23.07 ± 0.66	17.41 ± 0.37
PseudoLabel	49.98 ± 1.17	40.55 ± 1.70	30.91 ± 1.73	21.96 ± 0.42	16.21 ± 0.11
Mixup	47.43 ± 0.92	36.17 ± 1.36	25.72 ± 0.66	18.14 ± 1.06	13.15 ± 0.20
VAT	36.03 ± 2.82	26.11 ± 1.52	18.68 ± 0.40	14.40 ± 0.15	11.05 ± 0.31
MeanTeacher	47.32 ± 4.71	42.01 ± 5.86	17.32 ± 4.00	12.17 ± 0.22	10.36 ± 0.25
MixMatch	11.08 ± 0.87	9.65 ± 0.94	7.75 ± 0.32	7.03 ± 0.15	6.24 ± 0.06

Table 5: Error rate (%) for CIFAR10.

B.2 SVHN

Training the same model with supervised learning on the entire 73257-example training set achieved an error rate of 2.59%.

Methods/Labels	250	500	1000	2000	4000
PiModel	17.65 ± 0.27	11.44 ± 0.39	8.60 ± 0.18	6.94 ± 0.27	5.57 ± 0.14
PseudoLabel	21.16 ± 0.88	14.35 ± 0.37	10.19 ± 0.41	7.54 ± 0.27	5.71 ± 0.07
Mixup	39.97 ± 1.89	29.62 ± 1.54	16.79 ± 0.63	10.47 ± 0.48	7.96 ± 0.14
VAT	8.41 ± 1.01	7.44 ± 0.79	5.98 ± 0.21	4.85 ± 0.23	4.20 ± 0.15
MeanTeacher	6.45 ± 2.43	3.82 ± 0.17	3.75 ± 0.10	3.51 ± 0.09	3.39 ± 0.11
MixMatch	3.78 ± 0.26	3.64 ± 0.46	3.27 ± 0.31	3.04 ± 0.13	2.89 ± 0.06

Table 6: Error rate (%) for SVHN.

DivideMix: Learning with Noisy Labels as Semi-supervised Learning

标签噪声学习

Introduction

核心思想：模型将它认为是噪声标签的样本分离出来，把它们作为无标签样本，然后用半监督学习（semi-supervised learning, SSL）来对有标签+无标签样本进行学习。

整体流程：

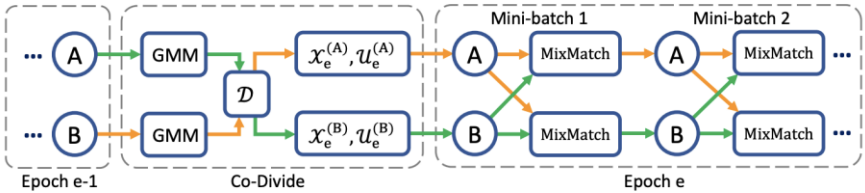


Figure 1: DivideMix trains two networks (A and B) simultaneously. At each epoch, a network models its per-sample loss distribution with a GMM to divide the dataset into a labeled set (mostly clean) and an unlabeled set (mostly noisy), which is then used as training data for the other network (*i.e.* co-divide). At each mini-batch, a network performs semi-supervised training using an improved MixMatch method. We perform label co-refinement on the labeled samples and label co-guessing on the unlabeled samples.

主要步骤：

method

核心思想：模型将它认为是噪声标签的样本分离出来，把它们作为无标签样本，然后用半监督学习（semi-supervised learning, SSL）来对有标签+无标签样本进行学习。

Co-divide by loss modeling

$$\ell(\theta) = \{\ell_i\}_{i=1}^{N} = \{ -\sum_{c=1}^{C} y_i^c \log(\mathrm{p}_{\mathrm{model}}^{\mathrm{c}}(x_i;\theta))\}_{i=1}^{N},$$

where $\mathrm{p}_{\mathrm{model}}^{\mathrm{c}}$ is the model's output softmax probability for class c.

Confidence Penalty for asymmetric noise

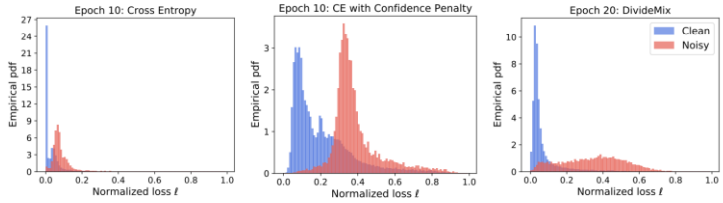


Figure 2: Training on CIFAR-10 with 40% asymmetric noise, warm up for 10 epochs. (a) Standard training with cross-entropy loss causes the model to overfit and produce over-confident predictions, making ℓ difficult to be modeled by the GMM. (b) Adding a confidence penalty (negative entropy) during warm up leads to more evenly-distributed ℓ. (c) Training with DivideMix can effectively reduce the loss for clean samples while keeping the loss larger for most noisy samples.

主要步骤：

Co-refinement+Co-guess+MixMatch

To account for label noise, we make two improvements to MixMatch which enable the two networks to teach each other. First, we perform label co-refinement for labeled samples by linearly combining the ground-truth label y_b with the network's prediction p_b (averaged across multiple augmentations of x_b), guided by the clean probability w_b produced by the other network:

$$\bar{y}_b = w_b y_b + (1 - w_b) p_b. \tag{3}$$

Then we apply a sharpening function on the refined label to reduce its temperature:

$$\hat{y}_b = \text{Sharpen}(\bar{y}_b, T) = \bar{y}_b^{c\,\frac{1}{T}} \Big/ \sum_{c=1}^{C} \bar{y}_b^{c\,\frac{1}{T}}, \text{ for } c = 1, 2, ..., C. \tag{4}$$

Second, we use the ensemble of predictions from both networks to "co-guess" the labels for unlabeled samples (algorithm 1, line 20), which can produce more reliable guessed labels.

Under high levels of noise, the network would be encouraged to predict the same class to minimize the loss. To prevent assigning all samples to a single class, we apply the regularization term used by Tanaka et al. (2018) and Arazo et al. (2019), which uses a uniform prior distribution π (*i.e.* $\pi_c = 1/C$) to regularize the model's average output across all samples in the mini-batch:

$$\mathcal{L}_{\text{reg}} = \sum_c \pi_c \log \left(\pi_c \Big/ \frac{1}{|\mathcal{X}'| + |\mathcal{U}'|} \sum_{x \in \mathcal{X}' + \mathcal{U}'} \text{p}_{\text{model}}^{c}(x; \theta) \right). \tag{11}$$

Finally, the total loss is:

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_u \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\text{reg}}. \tag{12}$$

In our experiments, we set λ_r as 1 and use λ_u to control the strength of the unsupervised loss.

伪代码：

Algorithm 1: DivideMix. Line 4-8: co-divide; Line 17-18: label co-refinement; Line 20: label co-guessing.

Input: $\theta^{(1)}$ and $\theta^{(2)}$, training dataset $(\mathcal{X}, \mathcal{Y})$, clean probability threshold τ, number of augmentations M, sharpening temperature T, unsupervised loss weight λ_u, Beta distribution parameter α for MixMatch.

$\theta^{(1)}, \theta^{(2)} = \text{WarmUp}(\mathcal{X}, \mathcal{Y}, \theta^{(1)}, \theta^{(2)})$ *// standard training (with confidence penalty)*

while $e < \text{MaxEpoch}$ **do**

  $\mathcal{W}^{(2)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(1)})$ *// model per-sample loss with $\theta^{(1)}$ to obtain clean proabability for $\theta^{(2)}$*

  $\mathcal{W}^{(1)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(2)})$ *// model per-sample loss with $\theta^{(2)}$ to obtain clean proabability for $\theta^{(1)}$*

  for $k = 1, 2$ **do** *// train the two networks one by one*

    $\mathcal{X}_e^{(k)} = \{(x_i, y_i, w_i) | w_i \geq \tau, \forall (x_i, y_i, w_i) \in (\mathcal{X}, \mathcal{Y}, \mathcal{W}^{(k)})\}$ *// labeled training set for $\theta^{(k)}$*

    $\mathcal{U}_e^{(k)} = \{x_i | w_i < \tau, \forall (x_i, w_i) \in (\mathcal{X}, \mathcal{W}^{(k)})\}$ *// unlabeled training set for $\theta^{(k)}$*

    for $\text{iter} = 1$ **to** num_iters **do**

      From $\mathcal{X}_e^{(k)}$, draw a mini-batch $\{(x_b, y_b, w_b); b \in (1, ..., B)\}$

      From $\mathcal{U}_e^{(k)}$, draw a mini-batch $\{u_b; b \in (1, ..., B)\}$

      for $b = 1$ **to** B **do**

        for $m = 1$ **to** M **do**

          $\hat{x}_{b,m} = \text{Augment}(x_b)$ *// apply m^{th} round of augmentation to x_b*

          $\hat{u}_{b,m} = \text{Augment}(u_b)$ *// apply m^{th} round of augmentation to u_b*

        end

        $p_b = \frac{1}{M} \sum_m \text{p}_{\text{model}}(\hat{x}_{b,m}; \theta^{(k)})$ *// average the predictions across augmentations of x_b*

        $\bar{y}_b = w_b y_b + (1 - w_b) p_b$

        // refine ground-truth label guided by the clean probability produced by the other network

        $\hat{y}_b = \text{Sharpen}(\bar{y}_b, T)$ *// apply temperature sharpening to the refined label*

        $\bar{q}_b = \frac{1}{2M} \sum_m \left(\text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(1)}) + \text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(2)})\right)$

        // co-guessing: average the predictions from both networks across augmentations of u_b

        $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// apply temperature sharpening to the guessed label*

      end

      $\hat{\mathcal{X}} = \{(\hat{x}_{b,m}, \hat{y}_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented labeled mini-batch*

      $\hat{\mathcal{U}} = \{(\hat{u}_{b,m}, q_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented unlabeled mini-batch*

      $\mathcal{L}_\mathcal{X}, \mathcal{L}_\mathcal{U} = \text{MixMatch}(\hat{\mathcal{X}}, \hat{\mathcal{U}})$ *// apply MixMatch*

      $\mathcal{L} = \mathcal{L}_\mathcal{X} + \lambda_u \mathcal{L}_\mathcal{U} + \lambda_r \mathcal{L}_{\text{reg}}$ *// total loss*

      $\theta^{(k)} = \text{SGD}(\mathcal{L}, \theta^{(k)})$ *// update model parameters*

    end

  end

end

Thanks