Equiangular Basis Vectors

Yang Shen Xuhao Sun Xiu-Shen Wei*
School of Computer Science and Engineering, Nanjing University of Science and Technology, China
{shenyang_98,sunxh,weixs}@njust.edu.cn

CVPR 2023

2023.6.25

Motivation

Task: classification tasks （Large categories）

Learning objective: 1. general k-way **classification layers**
2. classical deep **metric learning**

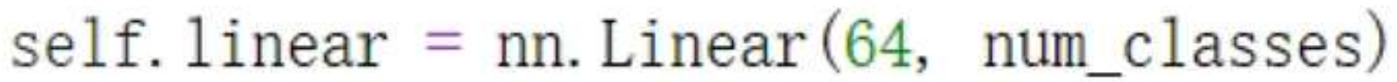
Problem:

1. The **trainable parameters** of the classifier rise as the **number of categories** becomes **larger**.

2. **Metric learning** requires a significant amount of **extra computation** for large-scale datasets, especially for those **pre-training tasks**.

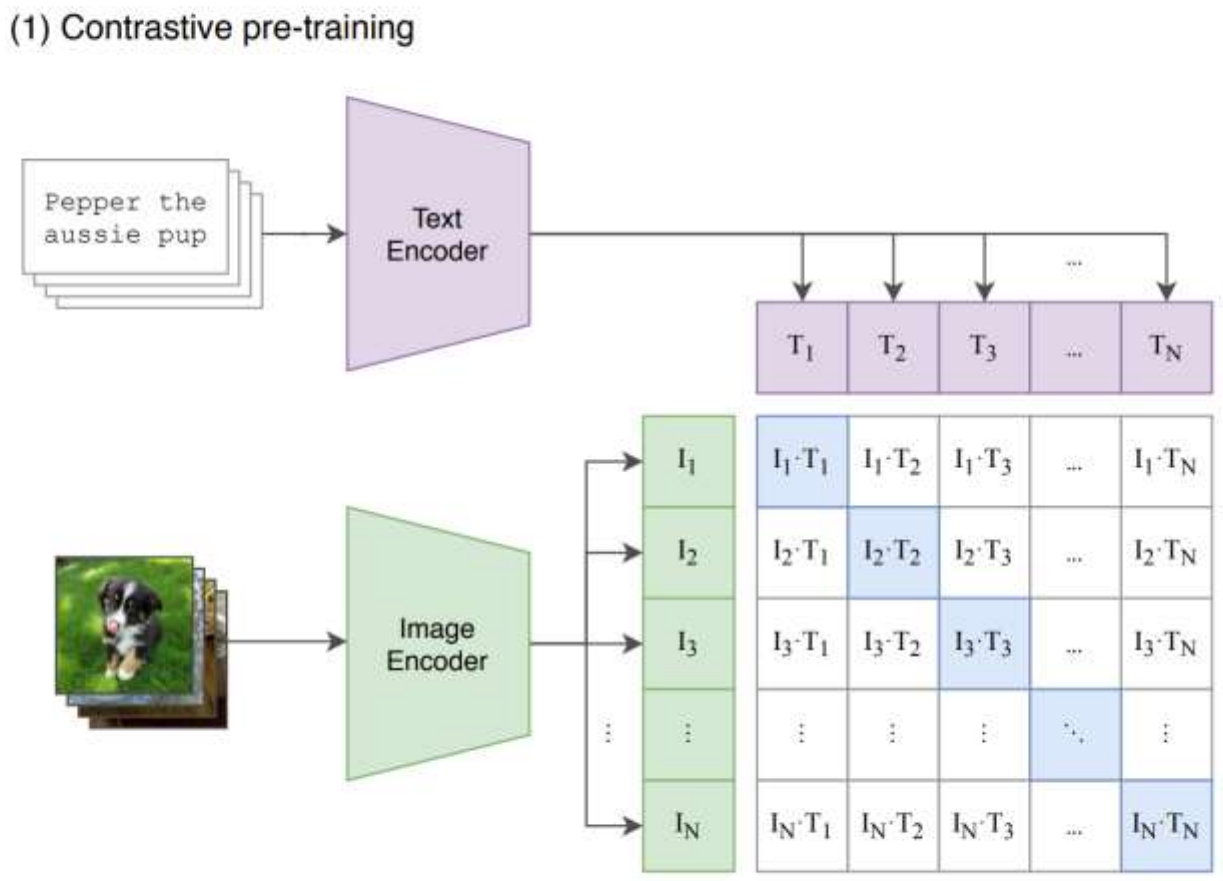
Table 1. Experiments on the dataset with 100,000 classes. “Params.” denotes parameters need to be optimized. “Top-1 Acc” represents Top-1 accuracy.

Method	Optimizer	EBVs Dim.	Params. (M)	Top-1 Acc (%)
FC	SGD	–	228.4	1.29
FC	AdamW	–	228.4	**30.25**
EBVs	SGD	5000	**33.8**	29.99

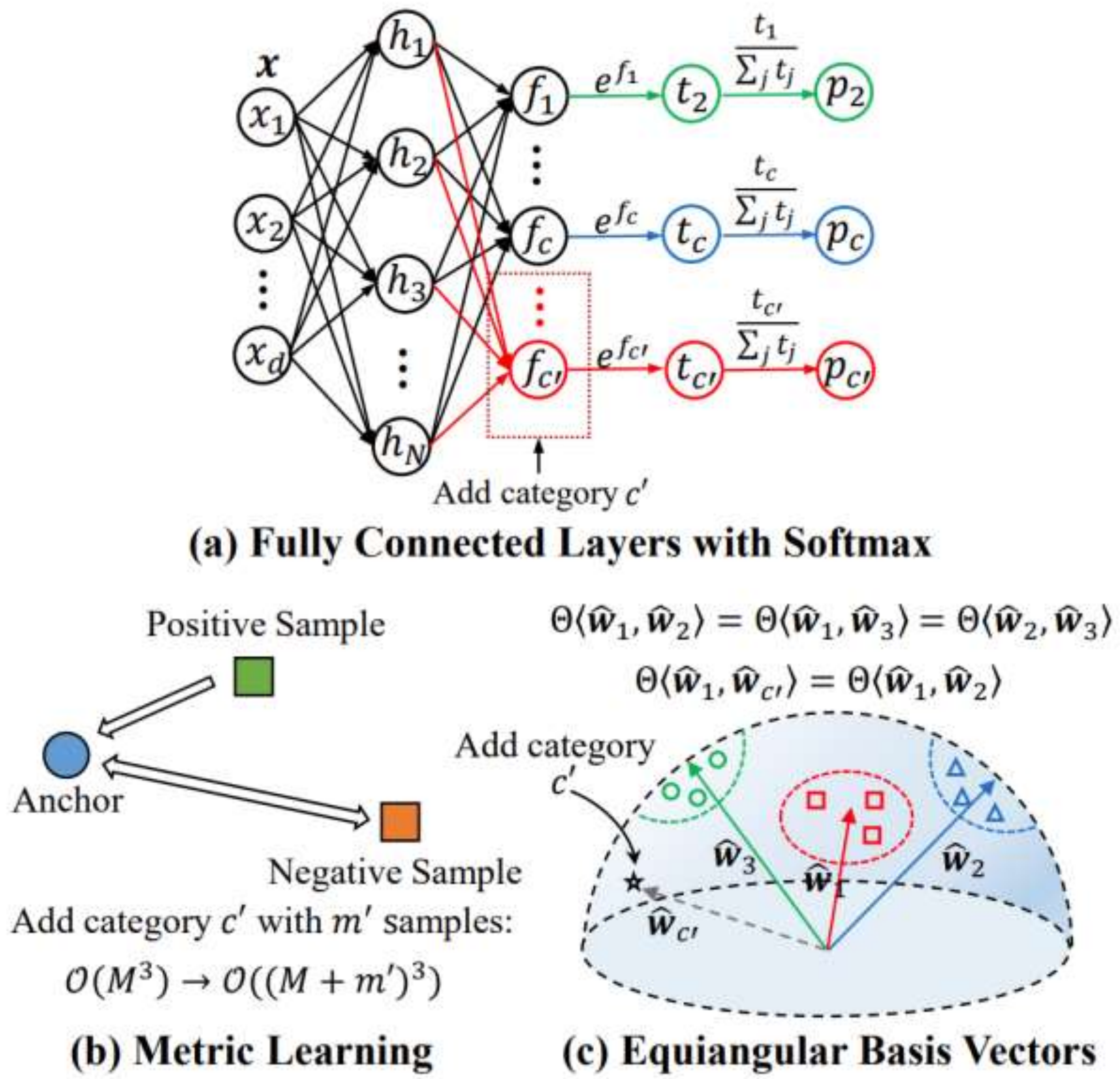


```
self.linear = nn.Linear(64, num_classes)
```

Metric learning: learn a **transformation function** that maps training data points from the original space to a new space where **similar points are closer** while **dissimilar points become farther apart**.


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ... IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
IN·T1 IN·T2 IN·T3 ... IN·TN

$$C(I,T) = \frac{1}{n}\sum_{i=1}^{n} -\log \frac{\exp(\mathrm{sim}(I_i,T_i)/\tau)}{\sum_{j=1}^{n}\exp\left(\mathrm{sim}(I_i,T_j)/\tau\right)}$$


x
x_1
x_2
x_d
h_1
h_2
h_3
h_N
f_1
f_c
$f_{c'}$
e^{f_1}
e^{f_c}
$e^{f_{c'}}$
t_2
t_c
$t_{c'}$
$\frac{t_1}{\sum_j t_j}$
$\frac{t_c}{\sum_j t_j}$
$\frac{t_{c'}}{\sum_j t_j}$
p_2
p_c
$p_{c'}$
Add category c'
(a) Fully Connected Layers with Softmax
Positive Sample
Anchor
Negative Sample
Add category c' with m' samples:
$\mathcal{O}(M^3) \rightarrow \mathcal{O}((M+m')^3)$
(b) Metric Learning
$\Theta\langle \hat{w}_1, \hat{w}_2 \rangle = \Theta\langle \hat{w}_1, \hat{w}_3 \rangle = \Theta\langle \hat{w}_2, \hat{w}_3 \rangle$
$\Theta\langle \hat{w}_1, \hat{w}_{c'} \rangle = \Theta\langle \hat{w}_1, \hat{w}_2 \rangle$
Add category c'
$\hat{w}_3$
$\hat{w}_1$
$\hat{w}_2$
$\hat{w}_{c'}$
(c) Equiangular Basis Vectors

1. **Definition EBVs**
2. **Generate EBVs**
3. **Optimize EBVs**

Figure 1. Comparisons between typical classification paradigms and our proposed Equiangular Basis Vectors (EBVs). **(a)** A general classifier ends with k-way fully connected layers with `softmax`. When adding more categories, the trainable parameters of the classifier grow linearly. **(b)** Taking triplet embedding [62] as an example of classical metric learning methods, the complexity is $\mathcal{O}(M^3)$ when given M images and it will grows to $\mathcal{O}((M+m')^3)$ when adding a new category with m' images. **(c)** Our proposed EBVs. EBVs predefine fixed normalized vector embeddings for different categories and these embeddings will not be changed during the training stage. The trainable parameters of the network will not be changed with the growth of the number of categories while the complexity only grows from $\mathcal{O}(M)$ to $\mathcal{O}(M+m')$.

3. Optimize EBVs

Implementations Suppose we have M sample-label pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$ in N classes and their d-dimensional features $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_M$ with $\boldsymbol{v}_i = f_{\boldsymbol{\theta}}(x_i)$, where $f_{\boldsymbol{\theta}}(\cdot)$ represents a feature extractor. A straightforward way to achieve the learning objective of EBVs is to directly optimize the cosine similarity between each $\boldsymbol{v}_i$ and the corresponding basis vector $\hat{\boldsymbol{w}}_{y_i} \in \mathcal{W}$.

$$P_{\mathrm{EBVs}}(y = y_i | \boldsymbol{v}_i) = \frac{\exp(\hat{\boldsymbol{w}}_i \hat{\boldsymbol{v}}_i / \tau)}{\sum_{j=1}^{N} \exp(\hat{\boldsymbol{w}}_j \hat{\boldsymbol{v}}_i / \tau)}, \qquad (6)$$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{M} \log P_{\mathrm{EBVs}}(y = y_i | f_{\boldsymbol{\theta}}(x_i)) \,. \qquad (7)$$

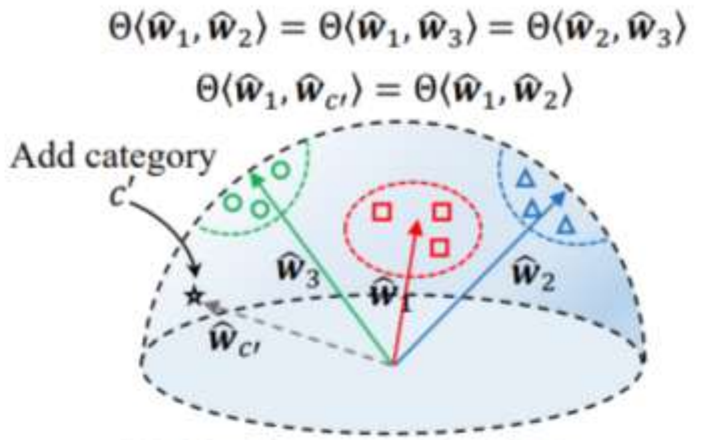
$$C(I, T) = \frac{1}{n} \sum_{i=1}^{n} -\log \frac{\exp(\mathrm{sim}(I_i, T_i)/\tau)}{\sum_{j=1}^{n} \exp\left(\mathrm{sim}(I_i, T_j)/\tau\right)}$$

1. Definition EBVs

1. Equiangular
2. as orthogonal as possible

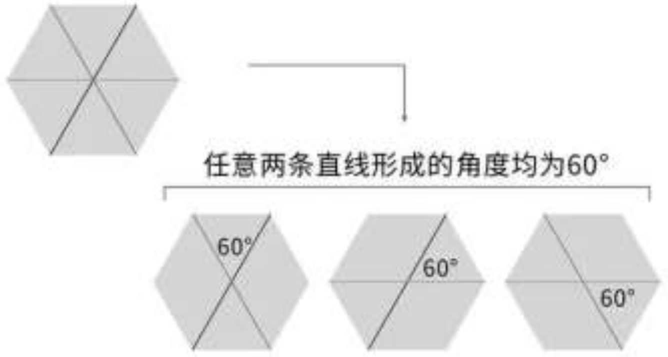
↓

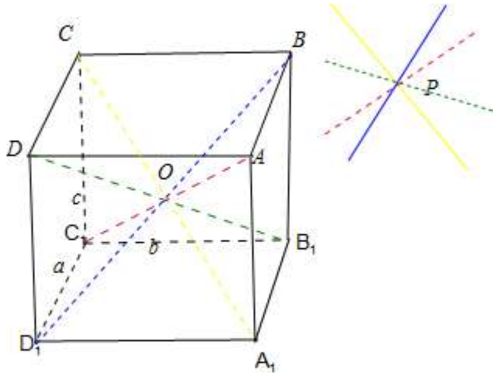
空间中存在多条直线成角两两相等且交于一点


$\Theta\langle\hat{w}_1,\hat{w}_2\rangle=\Theta\langle\hat{w}_1,\hat{w}_3\rangle=\Theta\langle\hat{w}_2,\hat{w}_3\rangle$
$\Theta\langle\hat{w}_1,\hat{w}_{c'}\rangle=\Theta\langle\hat{w}_1,\hat{w}_2\rangle$
Add category c'
$\hat{w}_3$
$\hat{w}_1$
$\hat{w}_2$
$\hat{w}_{c'}$

(c) Equiangular Basis Vectors

等角线（Equiangular lines），通俗地说就是，空间中**相交于同一个点的任意两条直线的夹角（非钝角）都相等**的一组直线，更严谨地说，在常见的情形中，这个等角线所处的空间是 d 维实欧氏空间 $\mathbb{R}^d$ 。





In Section 3.1 of the paper, we make the definition of the proposed EBVs, where $\alpha \in [0, 1)$ represents the maximum value of the absolute value of the cosine of the angle between any two vectors, d denotes the dimension of each coordinate vector while N denotes the number of the basis vectors. Specifically, for the EBVs set $\mathcal{W}$, each $\boldsymbol{w} \in \mathbb{R}^d$ in $\mathcal{W}$ should satisfies:

$$\forall \boldsymbol{w}_i, \boldsymbol{w}_j \in \mathcal{W}, i \neq j, \quad -\alpha \leq \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \, \|\boldsymbol{w}_j\|} \leq \alpha \,, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm and $\mathrm{card}(\mathcal{W}) = N$.

clidean norm. Let ϕ denote the spherical distance function, which can also be replaced by the cosine similarity function. EBVs produce a distribution over classes for a query point $\boldsymbol{v} \in \mathbb{R}^d$ based on `softmax` over cosine similarity to the N fixed coordinate vectors in the embedding space:

$$p(y = k|\boldsymbol{v}) = \frac{\exp(-\phi(\boldsymbol{v}, \boldsymbol{w}_k))}{\sum_{k'} \exp(-\phi(\boldsymbol{v}, \boldsymbol{w}_{k'}))} \,, \tag{2}$$

where $y = k \in \{1, 2, \ldots, N\}$ denotes the corresponding coordinate vector, which can also be seen as the corresponding label. While k' represents the associated basis vectors

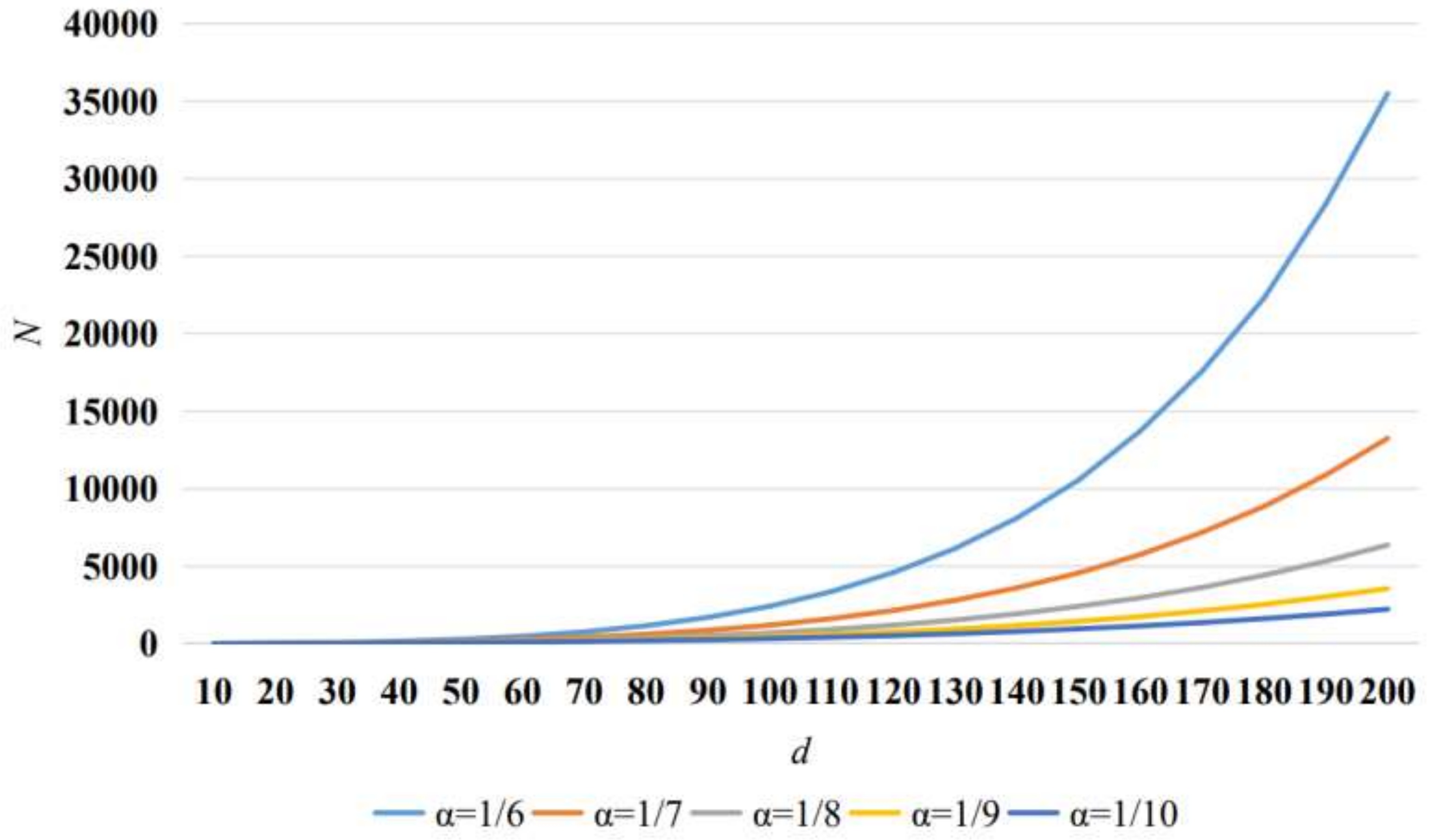


Figure 1. Relations between α, d and N.

2. Generate EBVs

而后我们给出 EBVs 的生成方式。我们随机初始化一个矩阵 $\mathbf{W} \in \mathbb{R}^{d\times N}$ 用以代表等夹角向量基集合 $\mathcal{W}$，其中 d 代表每个基向量的维度， N 代表需要的基向量的个数。而后对 W 中每一个 d 维的基向量进行归一化，这样， W 中任意两个基向量 $\mathbf{w}_i$ 与 $\mathbf{w}_j$ 可以表示为 $\hat{\mathbf{w}}_i$ 与 $\hat{\mathbf{w}}_j$, $i, j \in \{1, 2, \ldots, N\}$ 且 $i \neq j$ ， $\hat{\mathbf{w}}_i = \frac{w_i}{||w_i||}$ 。如此， $\hat{\mathbf{w}}_i$ 与 $\hat{\mathbf{w}}_j$ 的球面距离就可以用余弦相似度替代，表示为 $\arccos(\hat{\mathbf{w}}_i \cdot \hat{\mathbf{w}}_j)$ 。在随机梯度下降过程中，通过梯度裁剪的方式截断满足 $-\alpha \leq \mathbf{w}_i \cdot \mathbf{w}_j \leq \alpha$ 的任意基向量对的梯度，同时优化剩余的基向量对，整体优化函数可以表示为：

$$\underset{W}{\operatorname{argmin}} \sum_{i=1}^{N-1} \sum_{j>i}^{N} \max\left(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{w}}_j - \alpha, 0\right), \cdots\cdots (3)$$

即若 $\hat{\mathbf{w}}_i \cdot \hat{\mathbf{w}}_j - \alpha < 0$,则截断对应的梯度，不再进行优化。

Algorithm 1 Generation of EBVs in a PyTorch-like style

```
# $d$: Dim for each coordinate vectors
# $N$: The number of coordinate vectors
# $\alpha$: Threshold of $\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{w}}_j, i \neq j$
# $\boldsymbol{W}$: The EBVs matrix $\boldsymbol{W} \in \mathbb{R}^{d \times N}$
# slice: In the case of $N \gg d$, optimize $\boldsymbol{W}$ by slicing
# $\lambda$: Learning rate.

Initialize $\boldsymbol{W}$ randomly;
while True:
    #Normalize each row in $\boldsymbol{W}$
    $\boldsymbol{W}$ = Normalize($\boldsymbol{W}$)
    for i in $\lceil N/slice \rceil$:
        start = i * slice
        end = $\min(N$, (i + 1) * slice)
        E = F.onehot(arange(start, end), $N$)
        C = (W[start:end]@W.T).abs()-E
        loss = $\mathrm{ReLU}$(C $- \alpha$).sum() # Cutout the
        gradient of vector pairs which satisfies
        $-\alpha \leq \hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{w}}_j \leq \alpha$
        loss.backward()
    if $\max(\alpha$, C.$\max()) < \alpha + o(\alpha)$:
        Save($\boldsymbol{W}$)
        break
    $\boldsymbol{W}$ = $\boldsymbol{W}$ $- \lambda$ * $\boldsymbol{W}$.grad # Update $\boldsymbol{W}$
```

Table 1. Comparisons on the ImageNet-1K validation set. "FC" denotes models ending with a 1000-way fully connected layer with `softmax`. The test size for each image is set as "224^2" if there is one result while it is set as "$224^2/256^2$" if there exist two results.

Backbone	Method	Optimizer	LR	Epoch	Setting	Params.	GFLOPs	Test size	#Forward pass	Top-1 Acc (%)
ResNet-50	FC	SGD	0.1	90	A0	25.6M	4.1	224^2	600k	77.15
ResNet-50	FC EBVs	SGD	0.5	5/100	A1	25.6M	4.1	$224^2/256^2$	131k	76.88/77.92 **77.55/78.73**
ResNet-50	FC EBVs	SGD	0.5	5/100	A2	25.6M	4.1	$224^2/256^2$	131k	77.71/78.59 **78.14/78.99**
ResNet-50*	FC EBVs	SGD	0.5	5/600	A2	25.6M	4.1	$224^2/256^2$	755k	79.51/ – **79.73/80.45**
ResNet-50	FC FC EBVs	AdamW	0.001 0.01 0.01	5/100	A1	25.6M	4.1	$224^2/256^2$	131k	72.57/73.79 72.51/74.03 **75.62/77.14**
ResNet-50	FC FC EBVs	AdamW	0.001 0.01 0.01	5/100	A2	25.6M	4.1	$224^2/256^2$	131k	75.42/76.48 NaN **76.46/77.52**
Swin-T	FC EBVs	AdamW	0.001	5/100	A1	28.3M	4.5	224^2	131k	75.64 **78.37**
Swin-T	FC EBVs	AdamW	0.001	5/100	A2	28.3M	4.5	224^2	131k	79.12 **79.34**

* represents the result of the model with a general fully connected layer with `softmax` is provided by TorchVision [61].

Table 4. Ablation studies of the performance of stacked incremental improvements on top of baseline of our proposed EBVs. w/o EBVs denote models ending with a general fully connected classifier. ResNet-50 baseline is under Setting A0 in the paper but with only 1-crop testing. "Top-1 Acc" denotes Top-1 accuracy while "Abs. Diff." denotes absolute difference. The test size for each image is set as 224^2 except "FixRes Mitigations".

	Top-1 Acc (%)	Abs. Diff.
ResNet-50 Baseline	76.13	0.00
+ LR Optimizations w/o EBVs	76.49	0.36
+ TrivialAugment w/o EBVs	76.81	0.68
+ TrivialAugment	77.26	1.13
+ Random Erasing	77.55	1.42
+ Label Smoothing	77.61	1.48
+ Mixup	77.79	1.66
+ Cutmix	78.14	2.01
+ Long Training w/o EBVs	79.51	3.38
+ Long Training	79.73	3.60
+ FixRes Mitigations	**80.90**	**4.77**

Table 2. Top-1 accuracy of ResNet-32 on the long-tailed CIFAR-10 and CIFAR-100 datasets.

Dataset	Long-tailed CIFAR-10			Long-tailed CIFAR-100		
Imbalance ratio	100	50	10	100	50	10
FC	38.32	43.85	55.71	70.36	74.81	86.39
EBVs	**40.41**	**44.68**	**57.82**	**73.31**	**78.97**	**87.9**

Table 3. Comparisons of classification accuracy (%) on the iNaturalist 2018 dataset.

Method	Test size	Top-1 Acc (%)
FC*	224^2	61.7
FC	$224^2/256^2$	64.03 / 65.43
EBVs	$224^2/256^2$	**65.00 / 67.12**

* denotes the model is trained without TrivialAugment and LR optimizations.

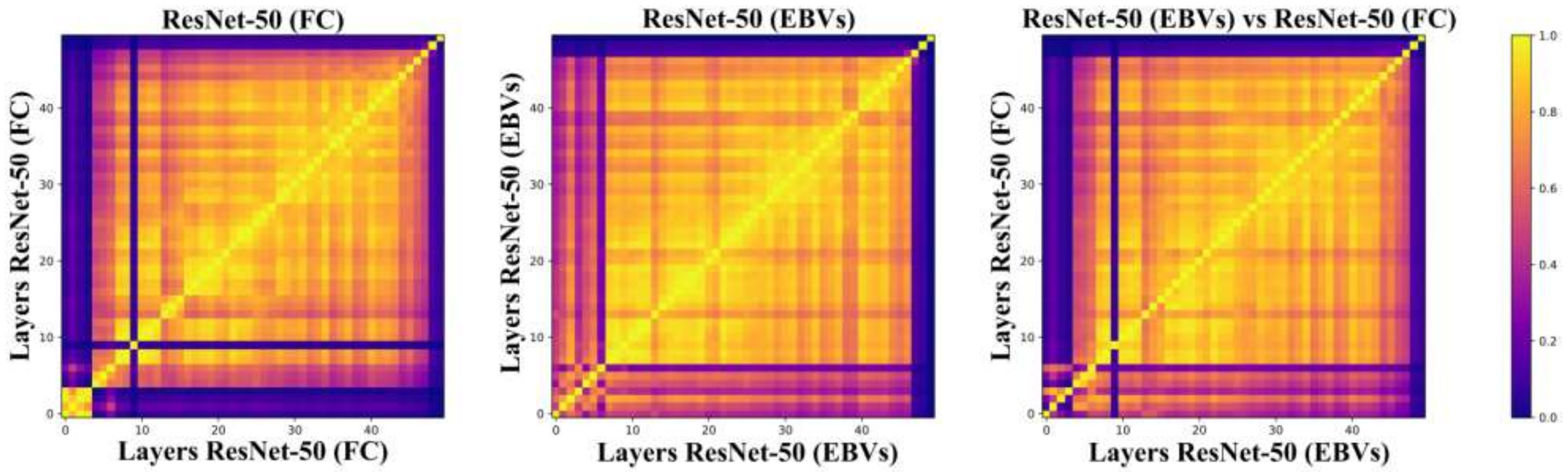


Figure 2. Representation structure of ResNet-50. **Left:** Similarity between layers within ResNet-50 ends with a general fully connected classifier (FC). It shows that the last few layers share minimal similarity with the shallow layers while a few shallow layers also share minimal similarity with all the other layers. **Middle:** Similarity between layers within ResNet-50 ends with EBVs. Only the last few layers share minimal similarity with other layers. **Right:** Similarity between layers across ResNet-50 ends with a general fully connected layer with `softmax` and our proposed EBVs. Around 10% initial layers share a little similarity while the last few layers share the least similarity.

We explore whether there are differences in the way EBVs represent and solve image tasks compared to a fully connected layer with `softmax` in this section. In order to answer this question, we have to analyze the features in the hidden layers as features are usually spread across neurons.

吉林大学

THANKS