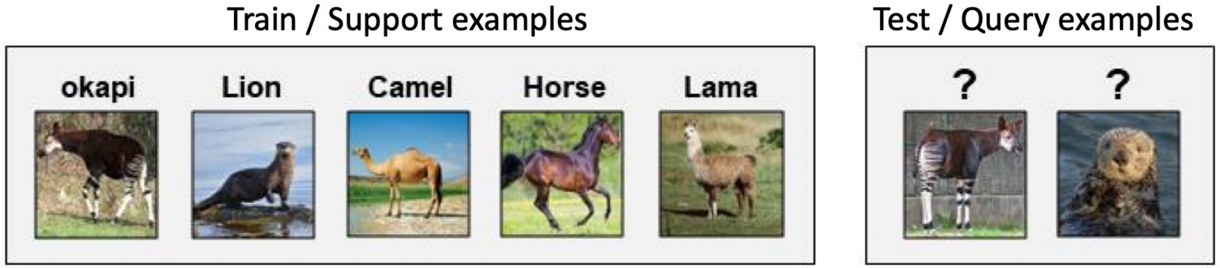
Joint Distribution Matters: Deep Brownian Distance Covariance for Few-Shot Classification

Jiangtao Xie[1,*], Fei Long[1,*], Jiaming Lv[1], Qilong Wang[2], Peihua Li[1,†]
[1]Dalian University of Technology, China [2]Tianjin University, China

CVPR2022 Oral

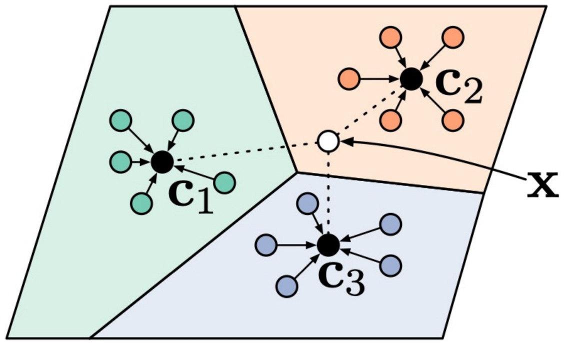
Few-Shot Classification

- Using small amounts of data to learn classifications with unseen labels
- N-way K-shot method:
  - **N =** number of classes
  - **K =** training examples per class, **as small as 1 or 5**



Example: 5-way **1**-shot classification task

- meta-learning / learning to learn：
  - model based methods
  - metric based methods √
  - optimization based methods



Method	Probability model	Dis-similarity/similarity measure	Joint distribution	Latency	Accuracy (%)	
					1-shot	5-shot
ProtoNet [33]	Mean vector	$\Vert\boldsymbol{\mu}_X - \boldsymbol{\mu}_Y\Vert^2$ or $\frac{\boldsymbol{\mu}_X^T\boldsymbol{\mu}_Y}{\Vert\boldsymbol{\mu}_X\Vert\Vert\boldsymbol{\mu}_Y\Vert}$	No	Low	49.42	68.20
CovNet [44]	Covariance matrix	$\Vert\boldsymbol{\Sigma}_X - \boldsymbol{\Sigma}_Y\Vert^2$	No	Low	49.64	69.45
ADM [20]	Gaussian distribution	$D_{\mathrm{KL}}(\mathcal{N}_{\boldsymbol{\mu}_X,\boldsymbol{\Sigma}_X}\Vert\mathcal{N}_{\boldsymbol{\mu}_Y,\boldsymbol{\Sigma}_Y})$	No	Low	53.10	69.73
DeepEMD [47]	Discrete distribution	$\min_{f_{\mathbf{x}_j,\mathbf{y}_l}\geq 0}\sum_j\sum_l f_{\mathbf{x}_j,\mathbf{y}_l}c_{\mathbf{x}_j,\mathbf{y}_l}$ s.t. $\sum_l f_{\mathbf{x}_j,\mathbf{y}_l}=f_{\mathbf{x}_j}, \sum_j f_{\mathbf{x}_j,\mathbf{y}_l}=f_{\mathbf{y}_l}$ for $\forall j,l$	Yes	High	65.91	82.41
DeepBDC (ours)	Characteristic function	$\int_{\mathbb{R}^p}\int_{\mathbb{R}^q}\frac{\lvert\phi_{XY}(\mathbf{t},\mathbf{s})-\phi_X(\mathbf{t})\phi_Y(\mathbf{s})\rvert^2}{c_p c_q\Vert\mathbf{t}\Vert^{1+p}\Vert\mathbf{s}\Vert^{1+q}}d\mathbf{t}d\mathbf{s}$	Yes	Low	**67.34**	**84.46**

- DeepBDC: a fundamental but largely overlooked dependency modeling method
- formulate DeepBDC as a highly modular and efficient layer

Brownian Distance Covariance

- random vectors $X \in \mathbb{R}^p, Y \in \mathbb{R}^q$
- joint characteristic function

$$\phi_{XY}(\mathbf{t},\mathbf{s})=\int_{\mathbb{R}^p}\int_{\mathbb{R}^q}\exp(i(\mathbf{t}^T\mathbf{x}+\mathbf{s}^T\mathbf{y}))f_{XY}(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y}$$

- marginal distribution

$$\phi_X(\mathbf{t})=\phi_{XY}(\mathbf{t},\mathbf{0}) \quad \phi_Y(\mathbf{s})=\phi_{XY}(\mathbf{0},\mathbf{s})$$

- BDC metric

$$\rho(X,Y)=\int_{\mathbb{R}^p}\int_{\mathbb{R}^q}\frac{|\phi_{XY}(\mathbf{t},\mathbf{s})-\phi_X(\mathbf{t})\phi_Y(\mathbf{s})|^2}{c_p c_q\|\mathbf{t}\|^{1+p}\|\mathbf{s}\|^{1+q}}d\mathbf{t}d\mathbf{s}$$

- empirical characteristic functions

$$\phi_{XY}(\mathbf{t},\mathbf{s}) = \frac{1}{m}\sum_{k=1}^{m}\exp(i(\mathbf{t}^T\mathbf{x}_k+\mathbf{s}^T\mathbf{y}_k))$$

Discrete BDC

- For the set of m observations $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$
- Using Euclidean distance

$$\widehat{\mathbf{A}} = (\widehat{a}_{kl}) \in \mathbb{R}^{m \times m} \text{ where } \widehat{a}_{kl} = \|\mathbf{x}_k - \mathbf{x}_l\|$$
$$\widehat{\mathbf{B}} = (\widehat{b}_{kl}) \in \mathbb{R}^{m \times m} \text{ where } \widehat{b}_{kl} = \|\mathbf{y}_k - \mathbf{y}_l\|$$

- BDC metrix

$$\mathbf{A} = (a_{kl}) \quad a_{kl} = \widehat{a}_{kl} - \frac{1}{m}\sum_{k=1}^{m} \widehat{a}_{kl} - \frac{1}{m}\sum_{l=1}^{m} \widehat{a}_{kl} - \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l=1}^{m} \widehat{a}_{kl}$$

- BDC metric

$$\rho(X, Y) = \operatorname{tr}\left(\mathbf{A}^T \mathbf{B}\right)$$
$$\rho(X, Y) = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T \mathbf{b}$$

-> **BDC metric has a closed form expression for discrete observations**

Székely G J, Rizzo M L. Brownian distance covariance[J]. The annals of applied statistics, 2009, 3(4): 1236-1265.

Deep BDC

- take for example $\boldsymbol{x}_k$ as a random observation (the k-th column of X)
- squared Euclidean distance matrix $\widetilde{\mathbf{A}} = (\tilde{a}_{kl})$
- Euclidean distance matrix $\widehat{\mathbf{A}} = (\sqrt{\widetilde{a}_{kl}})$
- BDC matrix $\mathbf{A}$

$$\widetilde{\mathbf{A}} = 2\big(\mathbf{1}(\mathbf{X}^T\mathbf{X} \circ \mathbf{I})\big)_{\mathrm{sym}} - 2\mathbf{X}^T\mathbf{X}$$

$$\widehat{\mathbf{A}} = \big(\sqrt{\widetilde{a}_{kl}}\big)$$

$$\mathbf{A} = \widehat{\mathbf{A}} - \frac{2}{d}\big(\mathbf{1}\widehat{\mathbf{A}}\big)_{\mathrm{sym}} + \frac{1}{d^2}\mathbf{1}\widehat{\mathbf{A}}\mathbf{1}$$

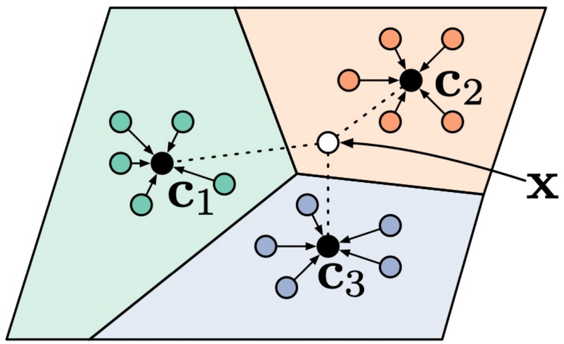
- involving standard matrix operations

-> appropriate for parallel computation on GPU

- $\boldsymbol{x}_k$: the k-th channel of the feature of an image

-> use BDC matrix as a self-similarity/encoder

Application on few-shot learning: ProtoNet



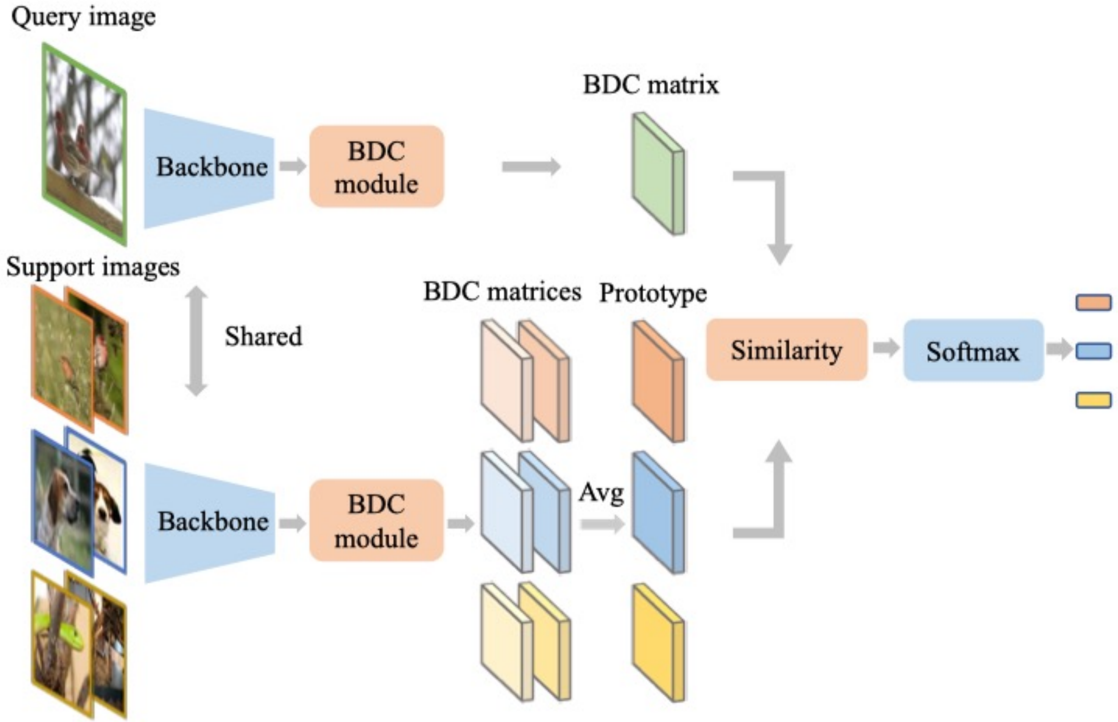
Based on ProtoNet:

- BDC matrix of an image $\mathbf{z}_j$: $\mathbf{A}_{\theta}(\mathbf{z}_j)$
- prototype of the support class k:

$$\mathbf{P}_k = \frac{1}{K} \sum_{(\mathbf{z}_j, y_j) \in \mathcal{S}_k} \mathbf{A}_{\theta}(\mathbf{z}_j)$$

- loss function:

$$\arg\min_{\theta} \; - \sum_{(\mathbf{z}_j, y_j) \in \mathcal{D}^{\mathrm{que}}} \log \frac{\exp(\tau \mathrm{tr}(\mathbf{A}_{\theta}(\mathbf{z}_j)^T \mathbf{P}_{y_j}))}{\sum_k \exp(\tau \mathrm{tr}(\mathbf{A}_{\theta}(\mathbf{z}_j)^T \mathbf{P}_k))}$$



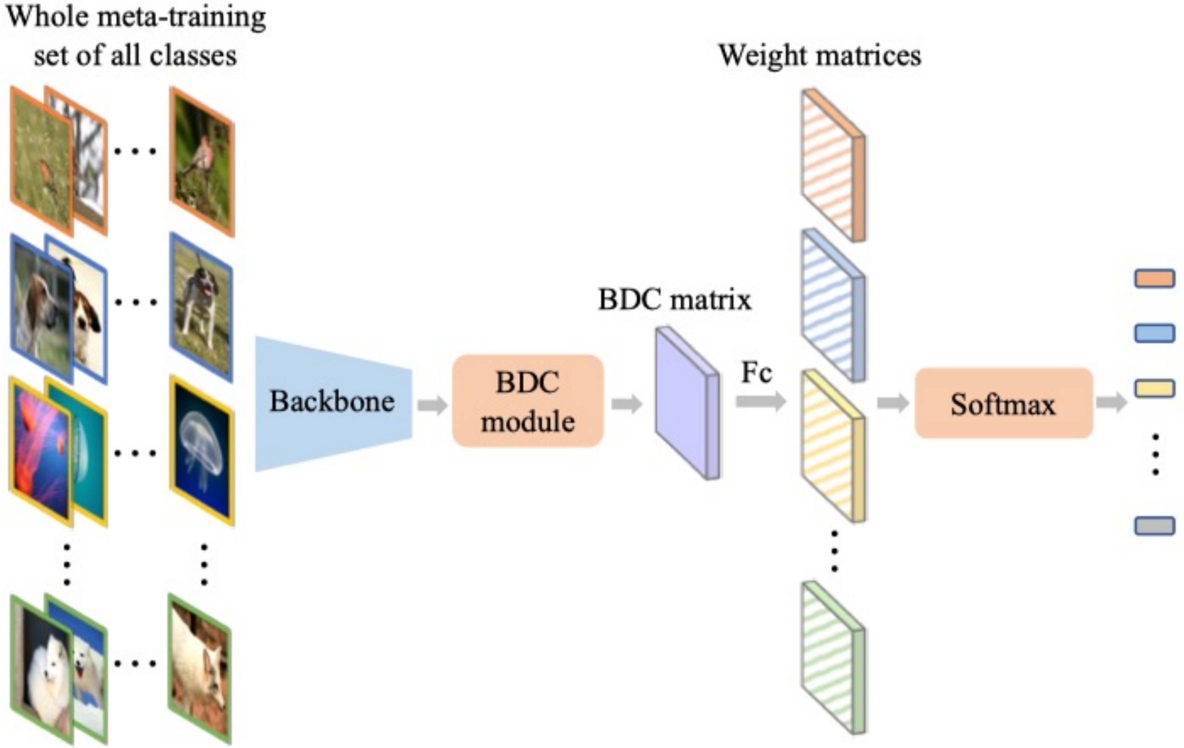
(a) Meta DeepBDC–Instantiation with ProtoNet [33] as a blueprint.

Application on few-shot learning: STL

Based on simple transfer learning (STL):

- Use the idea of clustering
- k-th weight matrix: $\mathbf{W}_k \in \mathbb{R}^{d\times d}$
- loss function：

$$\arg\min_{\boldsymbol{\theta},\mathbf{W}_k} -\sum_{(\mathbf{z}_j,y_j)\in\mathcal{C}^{\mathrm{train}}} \log \frac{\exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T\mathbf{W}_{y_j}))}{\sum_k \exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T\mathbf{W}_k))}$$



(b) STL DeepBDC–Instantiation based on Good-Embed [37] relying on non-episodic training.

Experiment

- Dataset: miniImageNet(100 classes), tieredImageNet(608 classes), CUB(200 bird classes)
- Backbone: ResNet-12, ResNet-18

Method	Backbone	*mini*ImageNet		*tiered*ImageNet	
		1-shot	5-shot	1-shot	5-shot
CTM [19]	ResNet-18	64.12±0.82	80.51±0.13	68.41±0.39	84.28±1.73
S2M2 [25]	ResNet-18	64.06±0.18	80.58±0.12	–	–
TADAM [26]	ResNet-12	58.50±0.30	76.70±0.38	–	–
MetaOptNet [18]	ResNet-12	62.64±0.44	78.63±0.46	65.99±0.72	81.56±0.63
DN4 [21] †	ResNet-12	64.73±0.44	79.85±0.31	–	–
Baseline++ [4] †	ResNet-12	60.56±0.45	77.40±0.34	–	–
Good-Embed [37]	ResNet-12	64.82±0.60	82.14±0.43	71.52±0.69	86.03±0.58
FEAT [46]	ResNet-12	66.78±0.20	82.05±0.14	70.80±0.23	84.79±0.16
Meta-Baseline [5]	ResNet-12	63.17±0.23	79.26± 0.17	68.62±0.27	83.29±0.18
MELR [11]	ResNet-12	67.40±0.43	83.40±0.28	72.14±0.51	87.01±0.35
FRN [45]	ResNet-12	66.45±0.19	82.83±0.13	71.16±0.22	86.01±0.15
IEPT [50]	ResNet-12	67.05±0.44	82.90±0.30	72.24±0.50	86.73±0.34
BML [51]	ResNet-12	67.04±0.63	83.63±0.29	68.99±0.50	85.49±0.34
ProtoNet [33] †	ResNet-12	62.11±0.44	80.77±0.30	68.31±0.51	83.85±0.36
ADM [20] †	ResNet-12	65.87±0.43	82.05±0.29	70.78±0.52	85.70±0.43
CovNet [44] †	ResNet-12	64.59±0.45	82.02±0.29	69.75±0.52	84.21±0.26
DeepEMD [47]	ResNet-12	65.91±0.82	82.41±0.56	71.16±0.87	86.03±0.58
Meta DeepBDC	ResNet-12	67.34±0.43	84.46±0.28	72.34±0.49	87.31±0.32
STL DeepBDC	ResNet-12	**67.83±0.43**	**85.45±0.29**	**73.82±0.47**	**89.00±0.30**

(a) Results on general object recognition datasets.

Method	Backbone	**CUB**	
		1-shot	5-shot
ProtoNet [33]	Conv4	64.42±0.48	81.82±0.35
FEAT [46]	Conv4	68.87±0.22	82.90±0.15
MELR [11]	Conv4	70.26±0.50	85.01±0.32
MVT [27]	ResNet-10	–	85.35±0.55
MatchNet [39]	ResNet-12	71.87±0.85	85.08±0.57
Wang *et al*. LR [43]	ResNet-12	76.16	90.32
MAML [12]	ResNet-18	68.42±1.07	83.47±0.62
Δ-encoder [32]	ResNet-18	69.80	82.60
Baseline++ [4]	ResNet-18	67.02±0.90	83.58±0.54
AA [1]	ResNet-18	74.22±1.09	88.65±0.55
Neg-Cosine [23]	ResNet-18	72.66±0.85	89.40±0.43
LaplacianShot [52]	ResNet-18	80.96	88.68
FRN [45] †	ResNet-18	82.55±0.19	92.98±0.10
Good-Embed [37] †	ResNet-18	77.92±0.46	89.94±0.26
ProtoNet [33] †	ResNet-18	80.90±0.43	89.81±0.23
ADM [20] †	ResNet-18	79.31±0.43	90.69±0.21
CovNet [44] †	ResNet-18	80.76±0.42	92.05±0.20
Meta DeepBDC	ResNet-18	83.55±0.40	93.82±0.17
STL DeepBDC	ResNet-18	**84.01±0.42**	**94.02±0.24**

(b) Results on fine-grained categorization dataset.

Method	Meta-training		Meta-testing		Accuracy	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet [S-20] †	304	365	115	143	62.11	80.77
ADM [S-12] †	908	967	199	221	65.87	82.05
CovNet [S-24] †	310	374	120	144	64.59	82.02
DeepEMD [S-27]	>80K	$>10^6$	457	12,617	65.91	82.41
Meta DeepBDC	505	623	161	198	67.34	84.46
STL DeepBDC	–		184	245	67.83	85.45

Table S-5. Comparison of latency (ms) for 5-way classification on *mini*ImageNet. † Reproduced with our setting.

Method	Backbone	5-shot
Baseline [4]	ResNet-18	65.57±0.70
Baseline++ [4]	ResNet-18	62.04±0.76
GNN+FT [38]	ResNet-12	66.98±0.68
BML [51]	ResNet-12	72.42±0.54
FRN [45]	ResNet-12	77.09±0.15
ProtoNet [33] †	ResNet-12	67.19±0.38
Good-Embed [37] †	ResNet-12	67.43±0.44
ADM [20] †	ResNet-12	70.55±0.43
CovNet [44] †	ResNet-12	76.77±0.34
Meta DeepBDC	ResNet-12	77.87±0.33
STL DeepBDC	ResNet-12	**80.16±0.38**

(a) *mini*ImageNet → CUB.

Method	Backbone	5-shot
ProtoNet [33] †	ResNet-12	55.96±0.38
ADM [20] †	ResNet-12	65.40±0.36
CovNet [44] †	ResNet-12	63.56±0.37
Baseline [4] †	ResNet-12	59.04±0.36
Baseline++ [4] †	ResNet-12	56.50±0.38
Good-Embed [37] †	ResNet-12	58.95±0.38
Meta DeepBDC	ResNet-12	68.67±0.39
STL DeepBDC	ResNet-12	**69.07±0.39**

(b) *mini*ImageNet → Aircraft.

Method	Backbone	5-shot
ProtoNet [33] †	ResNet-12	46.30±0.36
ADM [20] †	ResNet-12	53.94±0.35
CovNet [44] †	ResNet-12	52.90±0.37
Baseline [4] †	ResNet-12	50.29±0.37
Baseline++ [4] †	ResNet-12	46.44±0.37
Good-Embed [37] †	ResNet-12	50.18±0.37
Meta DeepBDC	ResNet-12	54.61±0.37
STL DeepBDC	ResNet-12	**58.09±0.36**

(c) *mini*ImageNet → Cars.

Table 5. Comparison with state-of-the-art methods for 5-way 5-shot classification in cross-domain scenarios. The best results are in **bold black** and second-best ones are in red. † Reproduced with our setting.

- size of channels: d -> size of BDC matrix: d^2

d	Parameters (M)	1-shot		5-shot	
		Acc	Latency	Acc	Latency
1280	13.25	66.36±0.43	488	83.23±0.30	614
960	13.04	66.81±0.44	280	83.68±0.28	351
640	12.84	**67.34±0.43**	161	**84.46±0.28**	198
512	12.75	67.10±0.45	134	84.23±0.28	164
256	12.59	66.90±0.43	121	84.15±0.28	148
ProtoNet [33]		62.11±0.44	115	80.77±0.30	143

Similarity function	1-shot		5-shot	
	Acc	Latency	Acc	Latency
Inner product	**67.34±0.43**	161	82.38±0.32	193
Cosine similarity	61.74±0.42	172	82.49±0.31	207
Euclidean distance	56.70±0.45	163	**84.46±0.28**	198

(a) Meta DeepBDC based on ProtoNet [33] as a blueprint.

d	Parameters (M)	1-shot		5-shot	
		Acc	Latency	Acc	Latency
512	13.41	64.92±0.43	1110	84.61±0.29	2016
256	12.75	66.15±0.43	371	85.44±0.29	587
196	12.65	66.57±0.43	285	85.36±0.29	424
128	12.55	**67.83±0.43**	184	**85.45±0.30**	245
64	12.48	66.97±0.44	137	83.18±0.30	172
Good-Embed [37]		64.82±0.44	121	82.14±0.43	155

Classifier	1-shot		5-shot	
	Acc	Latency	Acc	Latency
Logistic regression	**67.83±0.43**	184	**85.45±0.30**	245
SVM	66.29±0.44	113	84.73±0.29	144
Softmax	66.30±0.44	1250	85.20±0.29	4374

(b) STL DeepBDC based on [37] relying on non-episodic training.

Summary

- Application of Brownian distance to self-similarity
- Brownian distance is a good representation of the image
- the joint distribution is a good choice in the representation of the image

+ an easy way to implement
+ good performance on the few-shot task

- lack of the reason for the effectiveness