DEEP ACTIVE LEARNING FOR IMAGE CLASSIFICATION

Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, Sethuraman Panchanathan

Center for Cognitive Ubiquitous Computing (CUbiC)
Arizona State University

Motivation

- Traditional deep active learning

  --- The two-step method

  I. Train a deep model

  II. Actively samples based on the model

- The proposed method

  Solve these two independent problems simultaneously.

Traditional Deep Active Learning

Deep Model TRAINING

Utilize the cross entropy to estimate the empirical classification error

$$\operatorname*{argmin}_{f \in \mathscr{F}} E(\mathscr{D}; f) = \frac{1}{n_l} \sum_{i=1}^{n_l} L(f(x_i), y_i)$$

where $L(f(x_i), y_i) = -\sum_{j=1}^{C} 1\{y_i = j\} \log f_j(x_i),\ \forall i \in [1, \ldots, n_l]$ $\qquad f_j(x_i) = e^{h_{ij}^N} / \sum_{j'} e^{h_{ij}^N}$

Uncertainty Sampling

$$H(f(x_i)) = -\sum_{j=1}^{C} f_j(x_i) \log f_j(x_i),\ \forall i \in [n_l + 1, \ldots, n]$$

Problem: treat active learning and deep model training as two independent problems

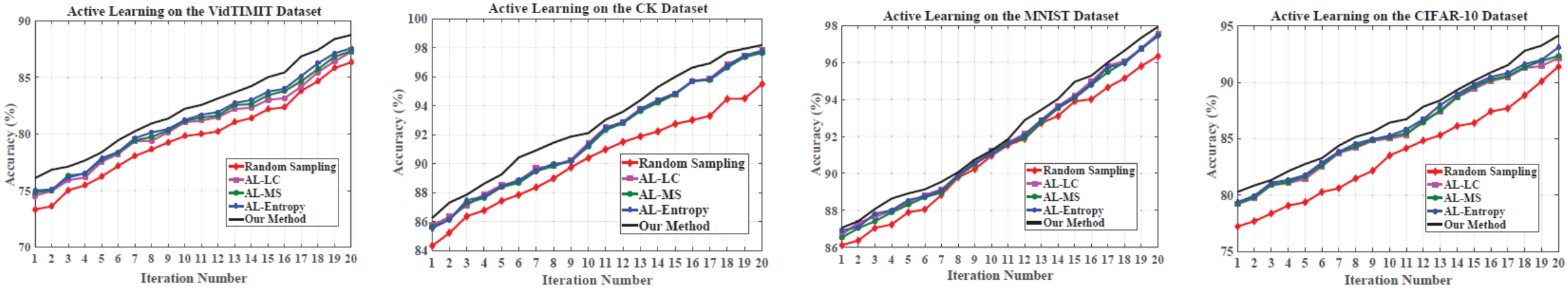
Joint Deep Active learning

It makes sure that examples with the largest entropy are the most uncertain and informative unlabeled ones

The joint objective function

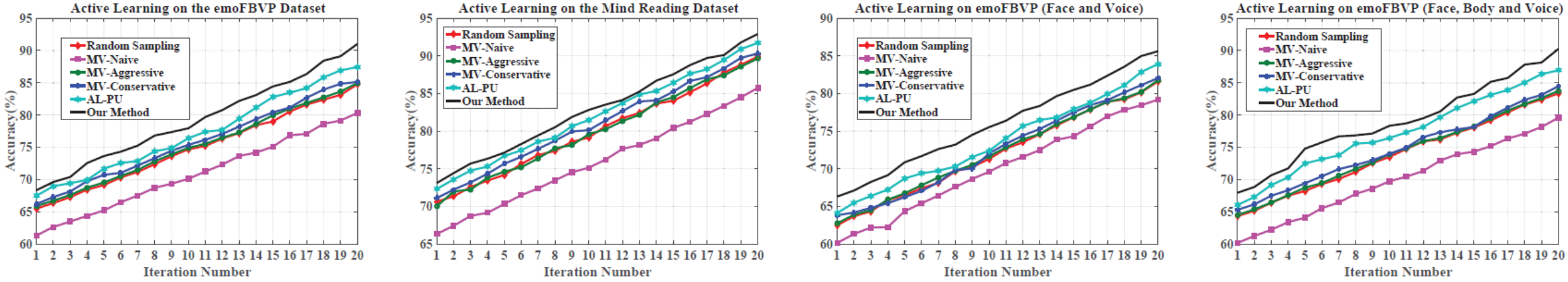
$$\underset{f \in \mathscr{F}}{\operatorname{argmin}} E(\mathscr{D}; f) = \frac{1}{n_l} \sum_{i=1}^{n_l} L(f(x_i), y_i) + \frac{\lambda}{n_u} \sum_{i=n_l+1}^{n} H(f(x_i))$$

Experiment



(a) VidTIMIT (b) CK (c) MNIST (d) CIFAR-10

Fig. 2. Active Learning on the Uni-modal Datasets. Best viewed in color.



(a) EmoFBVP (b) Mind Reading (c) EmoFBVP: Face and Voice (d) EmoFBVP: Face, Body and Voice

Fig. 3. (a) and (b), Active Learning on the Multi-modal Datasets emoFBVP and MindReading. (c) and (d) Active Learning on Subsets of Modalities of the emoFBVP Multi-modal Dataset. Best viewed in color.

Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds

Jordan T. Ash
Princeton University

Chicheng Zhang
Microsoft Research NYC

Akshay Krishnamurthy
Microsoft Research NYC

John Langford
Microsoft Research NYC

Alekh Agarwal
Microsoft Research Redmond

Motivation

- Classical active learning criteria
  - Uncertainty → Every datum in the batch is nearly identical
  - Diversity → Provide little useful information to the model

Suffers some trouble in batch mode !

- Combine two criteria in term of gradient
  I. Gradient Embedding
  II. Sampling

The Gradient Embedding

The cross-entropy loss

Note: $\ell_{\mathrm{CE}}(p, y) = \sum_{i=1}^{K} I(y = i) \ln \frac{1}{p_i}$

The weight of the last layer

$$\ell_{\mathrm{CE}}(f(x;\theta), y) = \ln\left(\sum_{j=1}^{K} e^{W_j \cdot g(x;V)}\right) - W_y \cdot g(x;V)$$

The weight of the previous layers

$$f(x;\theta) = \sigma(W \cdot g(x;V))$$

Assume that the model's current prediction is the true label

The i-th block of g_x can be computed as

$$(g_x)_i = \frac{\partial}{\partial W_i} \ell_{\mathrm{CE}}(f(x;\theta), \hat{y}) = (p_i - I(\hat{y} = i)) g(x;V)$$

$$\hat{y} = \mathrm{argmax}_{i \in [K]}\, p_i$$

Proposition: the norm of g_x is a low bound on the norm of the loss gradient induced by example with true label y

The Sampling Step

k-MEANS++ seeding

Algorithm 2 The k-MEANS++ seeding algorithm [4]

Require: Ground set $G \subset \mathbb{R}^d$, size k.
Ensure: Center set C of size k.
$C_1 \leftarrow \{ c_1 \}$, where c_1 is sampled uniformly at random from G.
for $t = 2, \ldots, k$: **do**
Define $D_t(x) := \min_{c \in C_{t-1}} \|x - c\|_2$.
$c_t \leftarrow$ Sample x from G with probability $\frac{D_t(x)^2}{\sum_{x \in G} D_t(x)^2}$.
$C_t \leftarrow C_{t-1} \cup \{ c_t \}$.
end for
return C_k.

BADGE Algorithm

Algorithm 1 BADGE: Batch Active learning by Diverse Gradient Embeddings

Require: Neural network $f(x;\theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of examples in a batch B.

Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with queried labels.

Train an initial model θ_1 on S by minimizing $\mathbb{E}_S[\ell(f(x;\theta), y)]$.

for $t = 1, 2, \ldots, T$: **do**

  For all examples x in $U \setminus S$:

  1. Compute its hypothetical label $\hat{y}(x) = h_{\theta_t}(x)$.
  2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{\text{out}}} \ell(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) layer.

  Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm, on $\{ g_x : x \in U \setminus S \}$, and query for their labels.

  $S \leftarrow S \cup S_t$.

  Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\text{CE}}(f(x;\theta), y)]$.

end for

return Final model θ_{T+1}.

Gradient embedding

Sampling

Experiment

Observations:

- As we show in our experiments, baseline methods that exploit just uncertainty or diversity do not consistently work well across model architectures, batch sizes, or datasets.
- BADGE is robust to batch size, architecture, and dataset, generally performing as well as or better than the best baseline across our experiments, which vary all of the aforementioned aspects of the setup.

Configurations:

- Coreset, Confidence sampling, Margin sampling, Entropy sampling, ALBL (Active Learning by Learning) and Random
- SVHN, CIFAR10, MINIST, and OpenML

Experiment

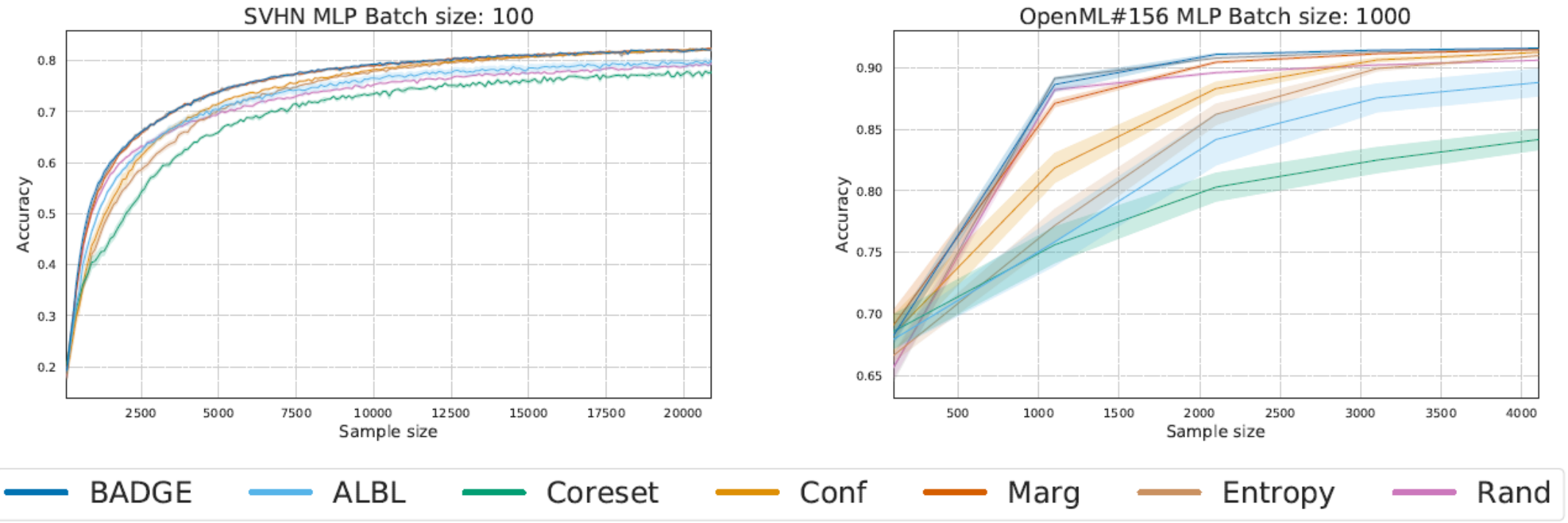


Figure 3: Test accuracy as a function of the number of total labeled samples for a range of active learning algorithms. CORESET, a diversity-based algorithm, does poorly because when data are complex and the model architecture has no useful prior, the last-layer embedding is not meaningful. **Left**: an MLP trained on SVHN with a batch size of 100. **Right**: An MLP trained on dataset #156 from OpenML with a batch size of 1000.

Experiment

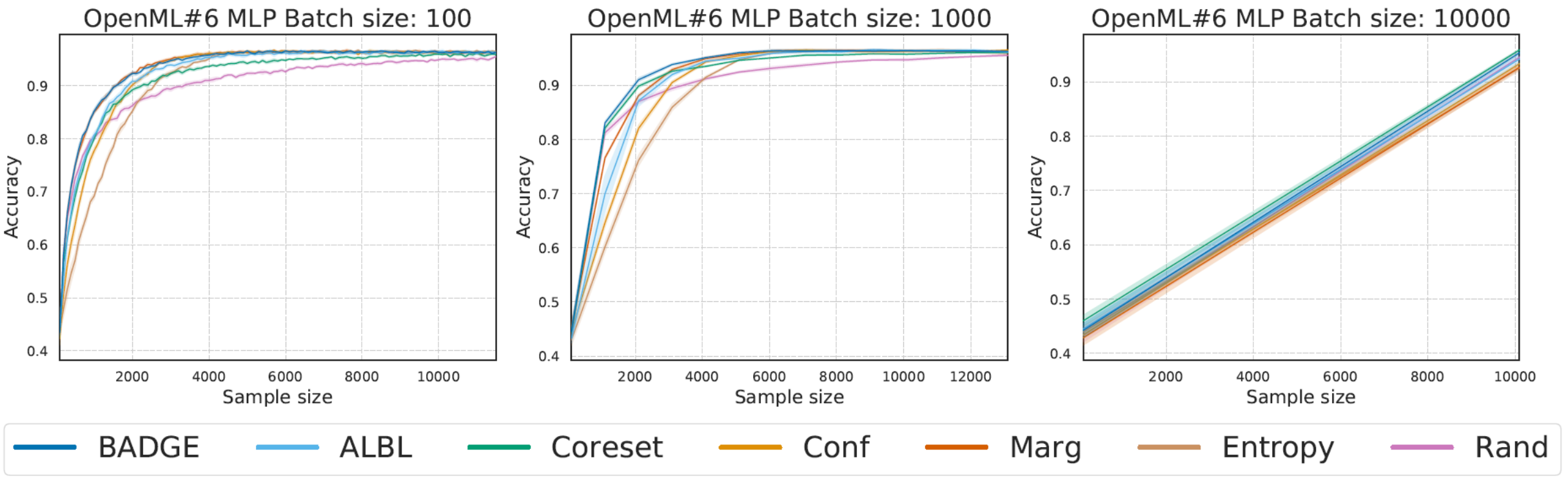


Figure 6: Learning curves for OpenML #6 with MLP.

Experiment

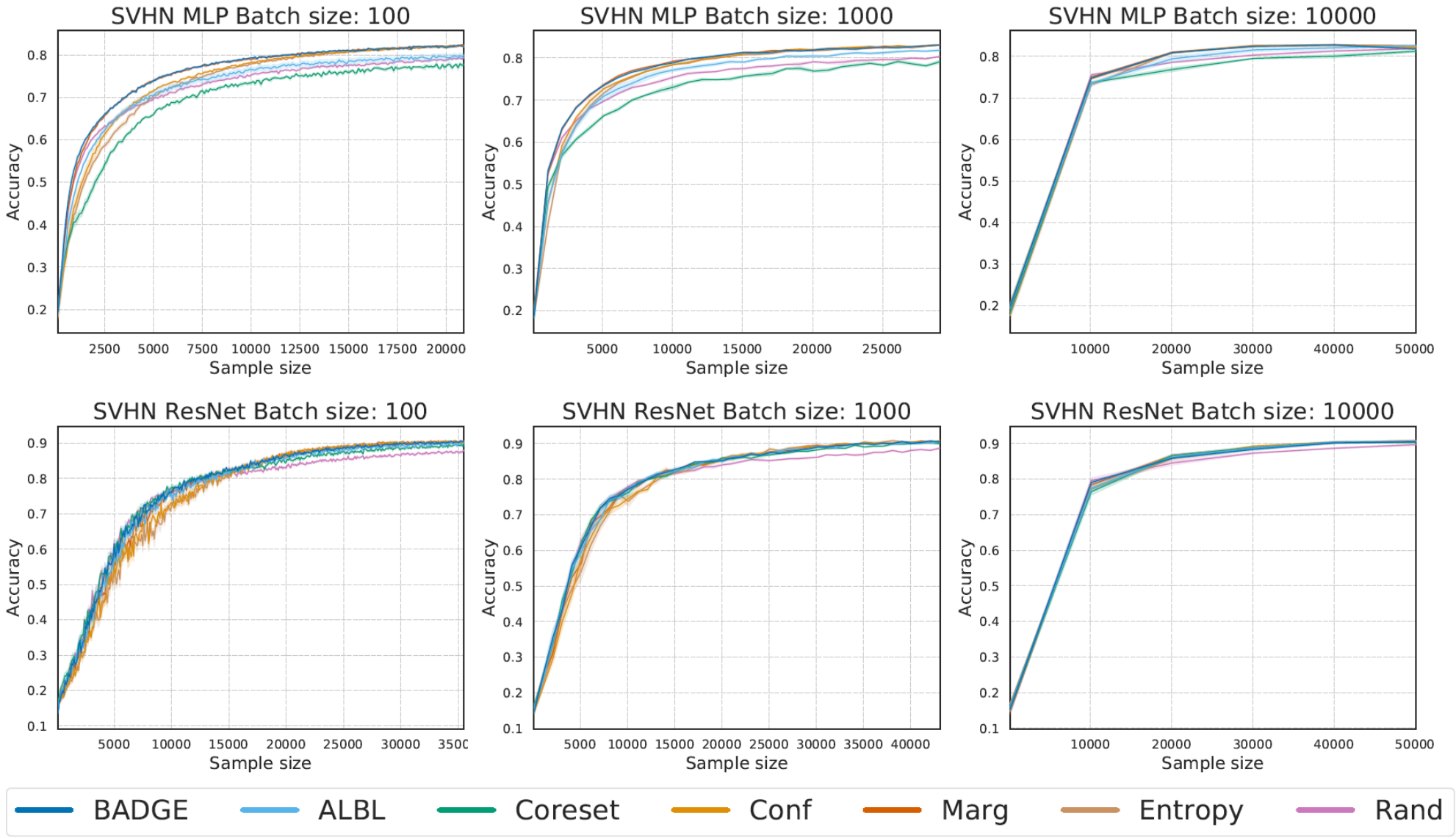


Figure 10: Learning curves for SVHN with MLP and ResNet.

Experiment

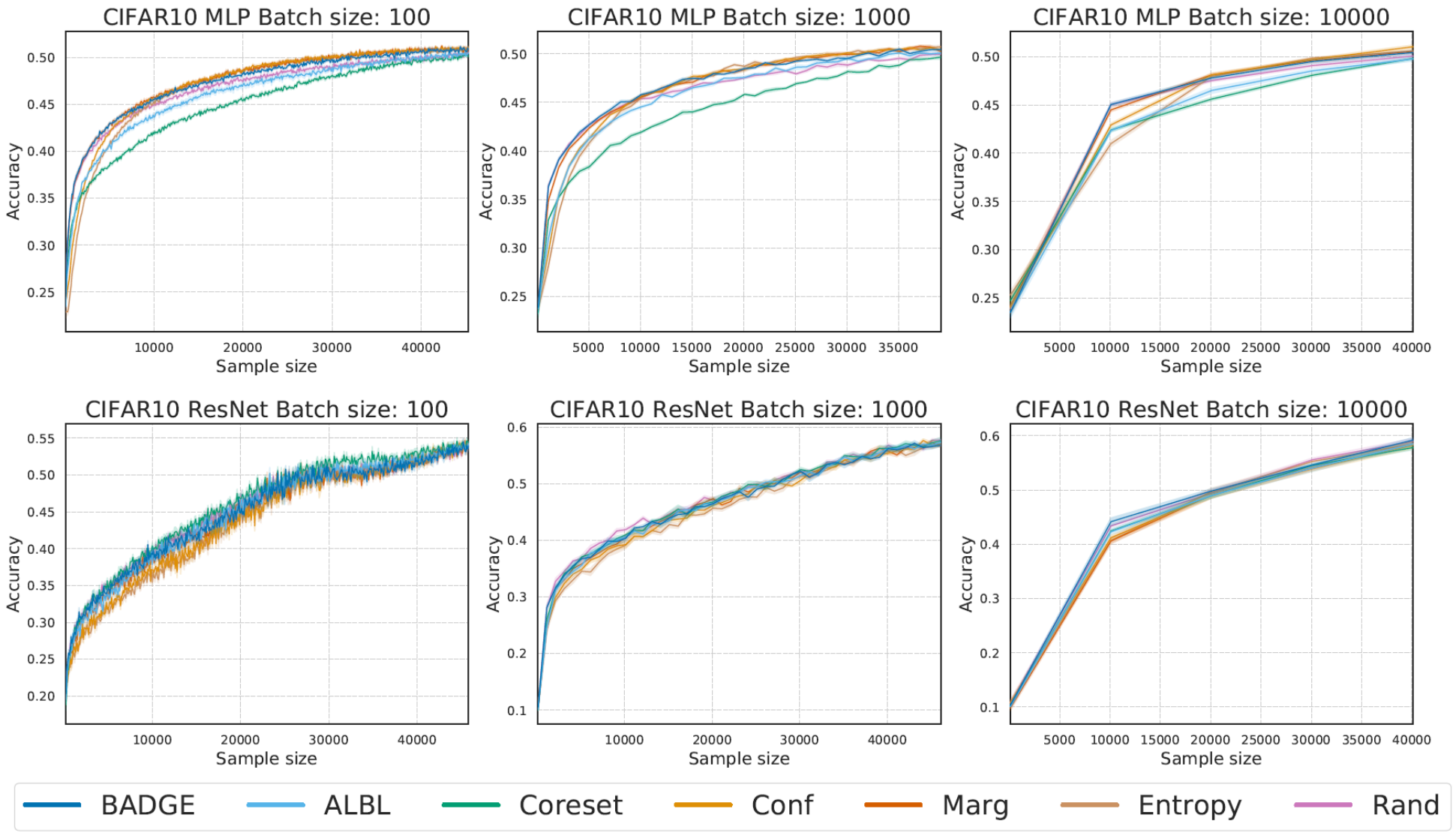


Figure 12: Learning curves for CIFAR10 with MLP (top) and ResNet (bottom).

Experiment

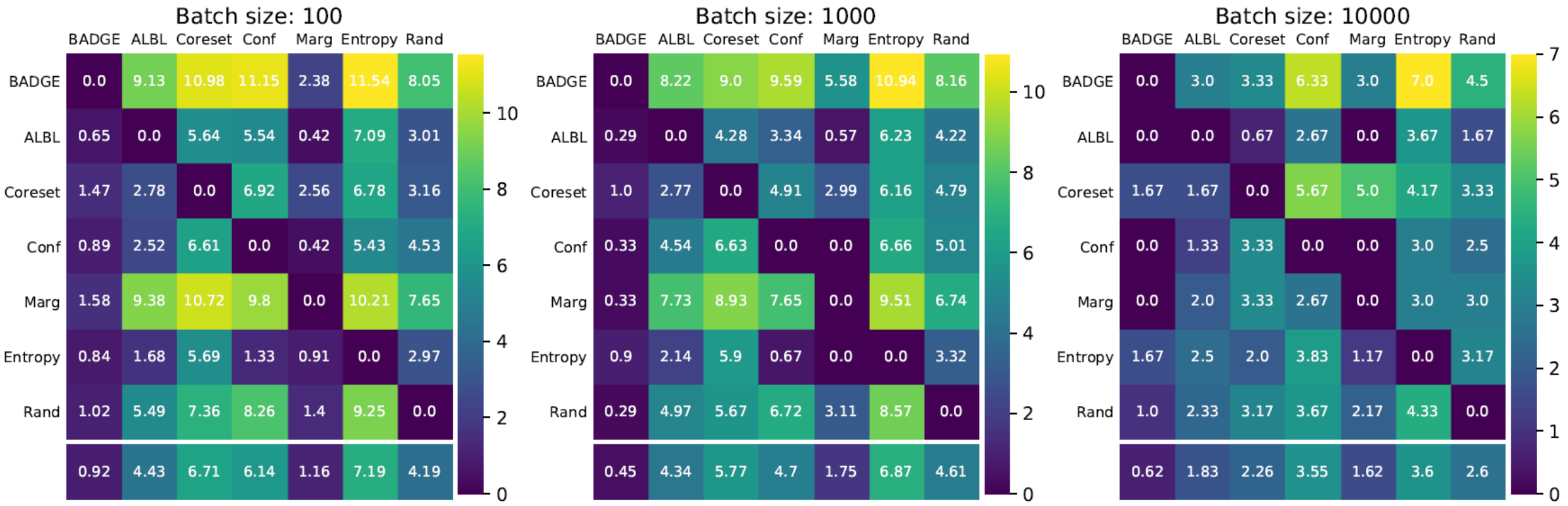


Figure 13: Pairwise penalty matrices of the algorithms, grouped by different batch sizes. Element i, j corresponds roughly to the number of times algorithm i outperforms algorithm j. Column-wise averages at the bottom show aggregate performance (lower is better). From left to right: batch size = 100, 1000, 10000.

Experiment

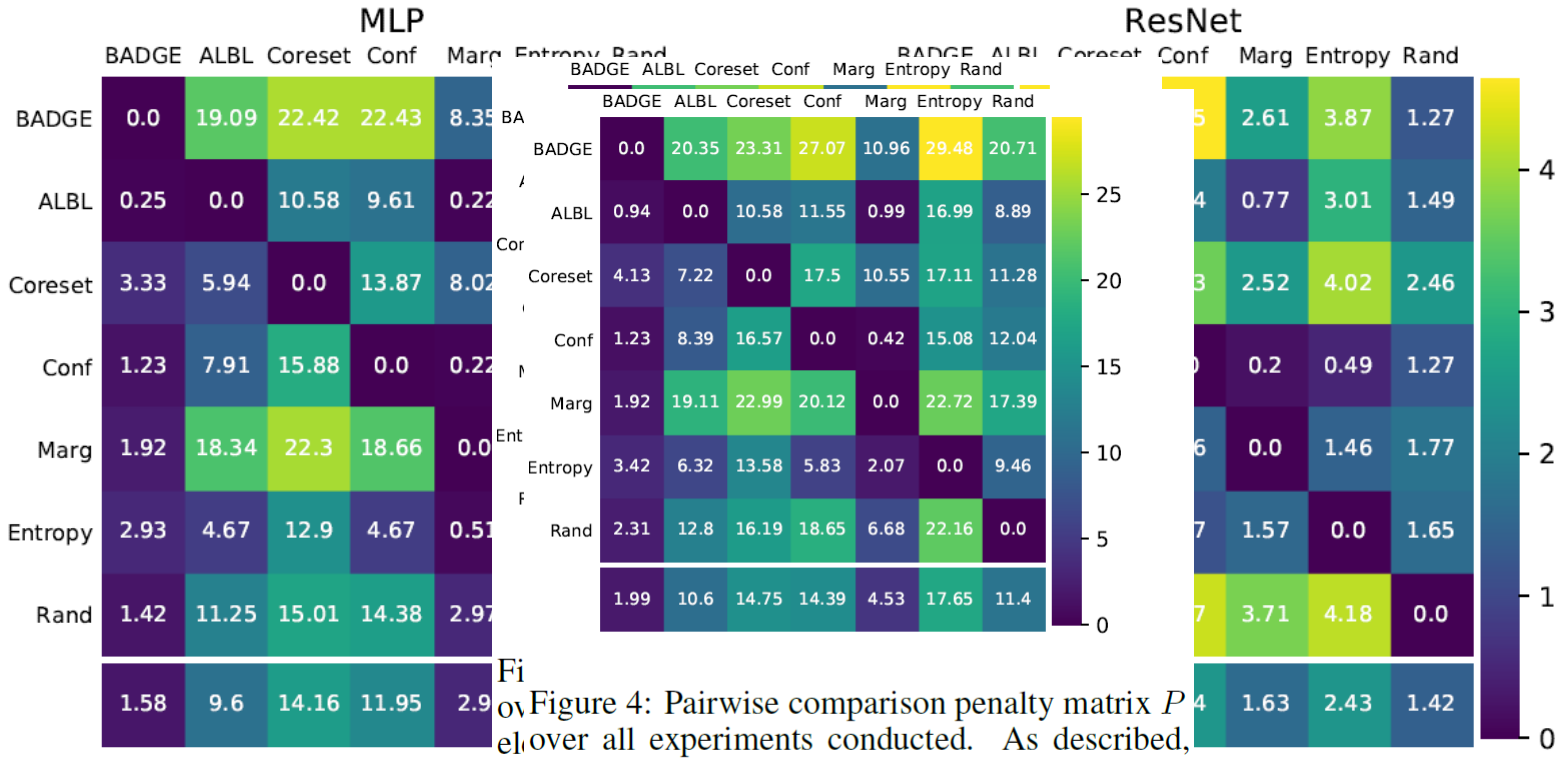


Figure 4: Pairwise comparison penalty matrix P over all experiments conducted. As described, element $P_{i,j}$ corresponds roughly to the number of times algorithm i outperforms algorithm j. Column-wise averages at the bottom show average performance (lower is better).

Figure 14: Pairwise penalty matrices of the algorithms, grouped by different neural network models. Element i, j corresponds roughly to the number of times algorithm i outperforms algorithm j. Column-wise averages at the bottom show aggregate performance (lower is better). From left to right: MLP and ResNet.

Experiment

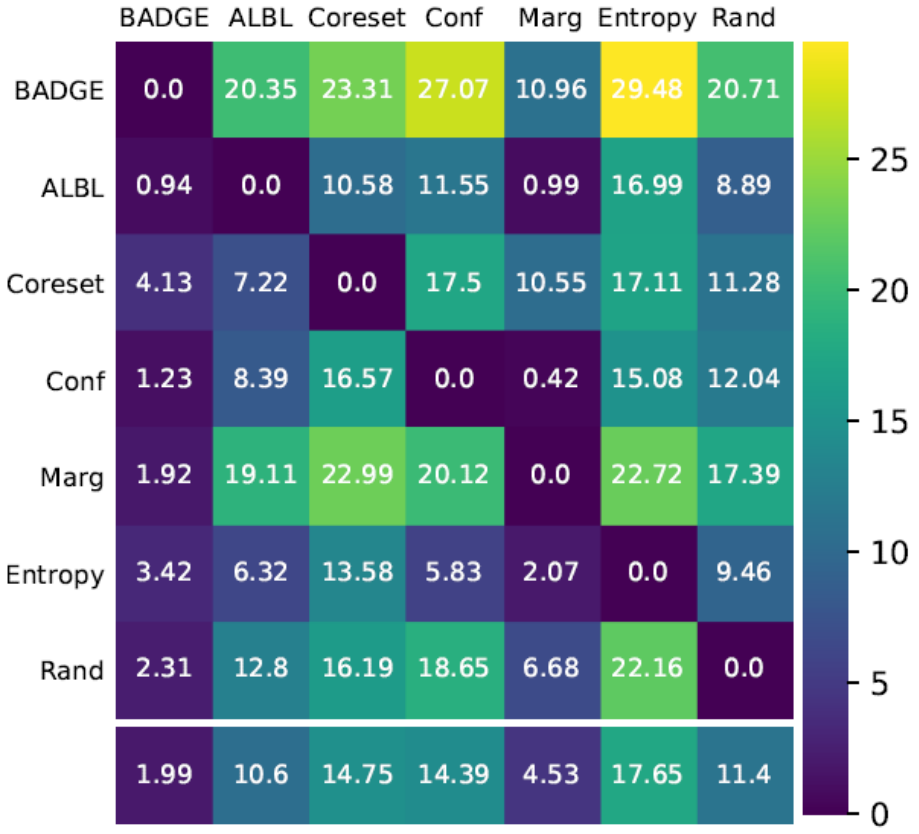


Figure 4: Pairwise comparison penalty matrix P over all experiments conducted. As described, element $P_{i,j}$ corresponds roughly to the number of times algorithm i outperforms algorithm j. Column-wise averages at the bottom show average performance (lower is better).

Thanks