Asynchronous Active Learning with Distributed Label Querying

Requirements

To install requirements:

```
pip install -r requirements.txt
```

Training

To train the model(s) in the paper, you should configure the parameters located in the configuration code segment in the following files:

main.py

```
server_num
worker_num
WorkerQueryNum          # Number of queries per worker
ServerTrainNum          # server training times
ServerTrainThreshold    # The server updates only when the worker queries the
increment
Assignment_Type         # Worker query strategy allocation method: RA or DA
Query_Strategy          # The given query strategy under DA
Split_Strategy          # The method of dividing the data set into workers: RS
or DS
args_data_type          # cifar-10, cifar-100, or miniimagenet
args_batch_size         # Number of queries per worker
args_initial_size
args_val_size
```

server_train.py

```
args_data_type
args_experiment_folder
```

work_query.py

```
args_data_type
query_batch           # Number of worker's single query
```

In addition, due to the long time required for full model training, two parameters (epoch and batch size) could be adjusted according to the actual situation in the models.py file.

After configuration, run this command:

```
python main.py
```

Evaluation

To evaluate the generated model, run:

```
python eval.py --data_type dataset_name --path model_path
```

Results

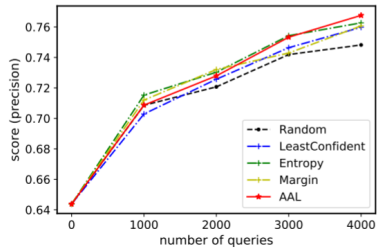
[Single Server]

Parameter configuration

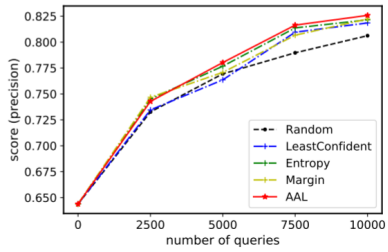
```
## Common parameter setting
server_num = 1
worker_num = 10
ServerTrainNum = 6
WorkerQueryNum = 6
Split_Strategy = RS
args_val_size = 5000

## For the AAL method
Assignment_Type = RA
Query_Strategy = NULL

## For other methods
Assignment_Type = DA
Query_Strategy = "UncertaintyEntropy" or'Uncertainty' or "UncertaintyMargin" or
"Random"

## For images from left to right
ServerTrainThreshold = 1000 or 2500 or 5000
args_batch_size = 100 or 250 or 500
query_batch = 100 or 250 or 500

## For Cifar-10
args_data_type = "cifar-10"
args_initial_size = 5000

## For Cifar-100
args_data_type = "cifar-100"
args_initial_size = 10000

## For miniimagenet
args_data_type = "miniimagenet"
args_initial_size = 10000
```

Performance comparison of different methods with a single server. The performance curves
are plotted as the querying number increases with different batch sizes.

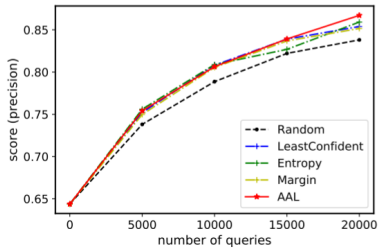
Cifar-10



(a) Cifar-10, batch_size=1000

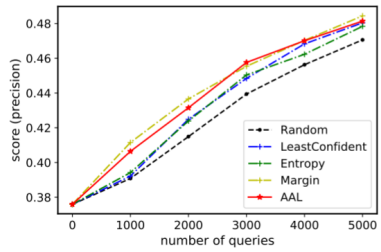


(b) Cifar-10, batch_size=2500

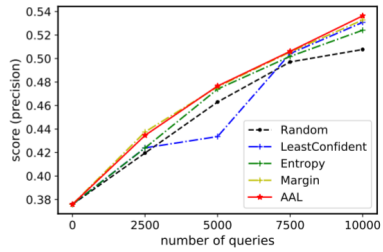


(c) Cifar-10, batch_size=5000

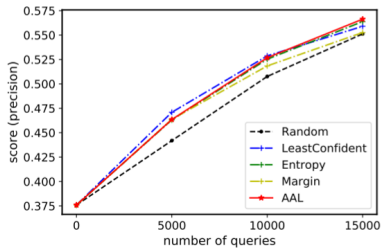
Cifar-100



(d) Cifar-100, batch_size=1000

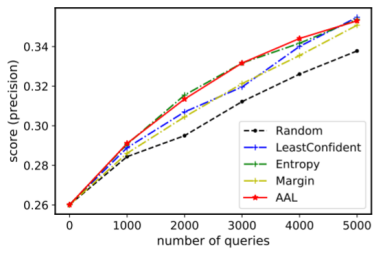


(e) Cifar-100, batch_size=2500

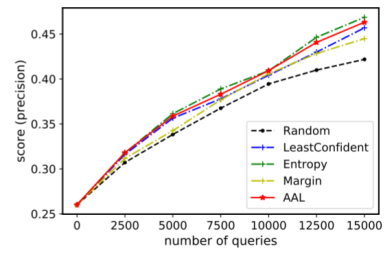


(f) Cifar-100, batch_size=5000

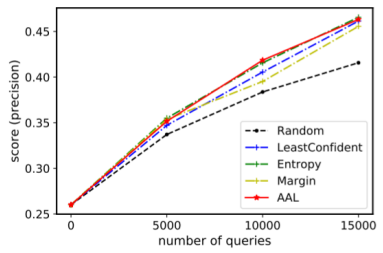
miniImagenet



(g) m-ImageNet, batch_size=1000

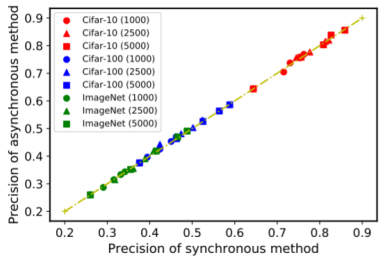


(h) m-ImageNet, batch_size=2500

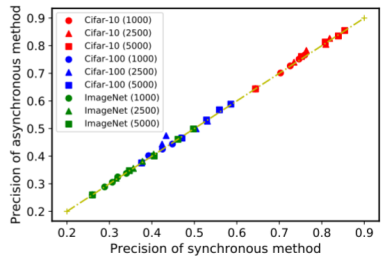


(i) m-ImageNet, batch_size=5000

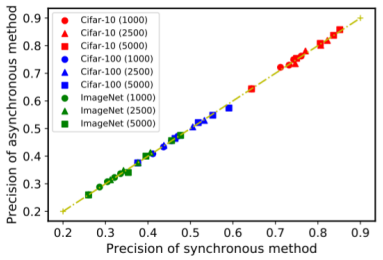
Performance comparison between synchronous and asynchronous mechanisms with different sampling strategies.



(a) Entropy based sampling

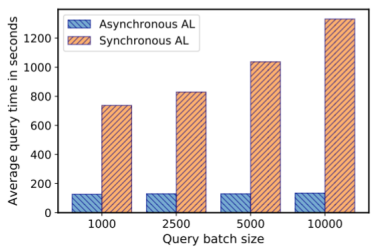


(b) Least confidence sampling

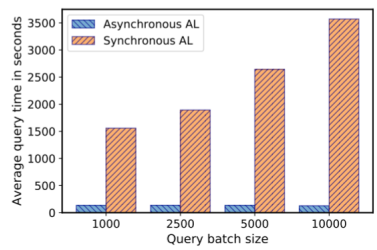


(c) Margin based sampling

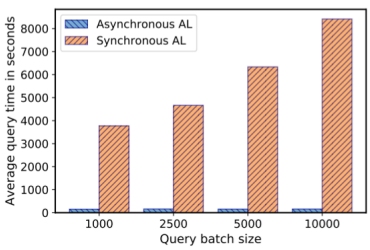
Comparison of average query time between synchronous and asynchronous methods on different datasets.



(a) Cifar-10



(b) Cifar-100



(c) Mini-ImageNet

[Multi-Server]

Parameter configuration

```
##Those common to the above will not be repeated

## Common parameter setting
worker_num = 10
WorkerQueryNum = 40
args_batch_size = 50
query_batch = 50
Split_Strategy = RS

# server_num = 1
server_num = 1
ServerTrainNum = 2
ServerTrainThreshold = 10000

# server_num = 2
server_num = 2
ServerTrainNum = 4
ServerTrainThreshold = 5000

# server_num = 4
server_num = 4
ServerTrainNum = 8
ServerTrainThreshold = 2500
```

The average performance of different methods with different numbers of servers.

Method	Entropy			Least Confident			Margin			AAL		
#Servers	1	2	4	1	2	4	1	2	4	1	2	4
Cifar-10	0.819	0.832	0.841	0.817	0.833	0.845	0.818	0.834	0.844	0.818	0.833	0.843
Cifar-100	0.353	0.362	0.369	0.356	0.370	0.383	0.349	0.357	0.365	0.357	0.365	0.371
ImageNet	0.386	0.399	0.408	0.388	0.397	0.406	0.379	0.389	0.399	0.394	0.403	0.410