universität wien

Fakultät für Informatik

# Breaking the Reclustering Barrier in Centroid-based Deep Clustering

13th International Conference on Learning Representations

April 24th - 28th 2025, Singapore

Authors: Lukas Miklautz[1,*], Timo Klein[1,2,*], Kevin Sidak[1,2,*], Collin Leiber[3,4], Thomas Lang[1,2], Andrii Shkabrii[1,2], Sebastian Tschiatschek[‡ ,1,5], Claudia Plant[‡ ,1,5]

[1] Faculty of Computer Science, University of Vienna, Vienna, Austria

[2] UniVie Doctoral School Computer Science, University of Vienna, Vienna, Austria

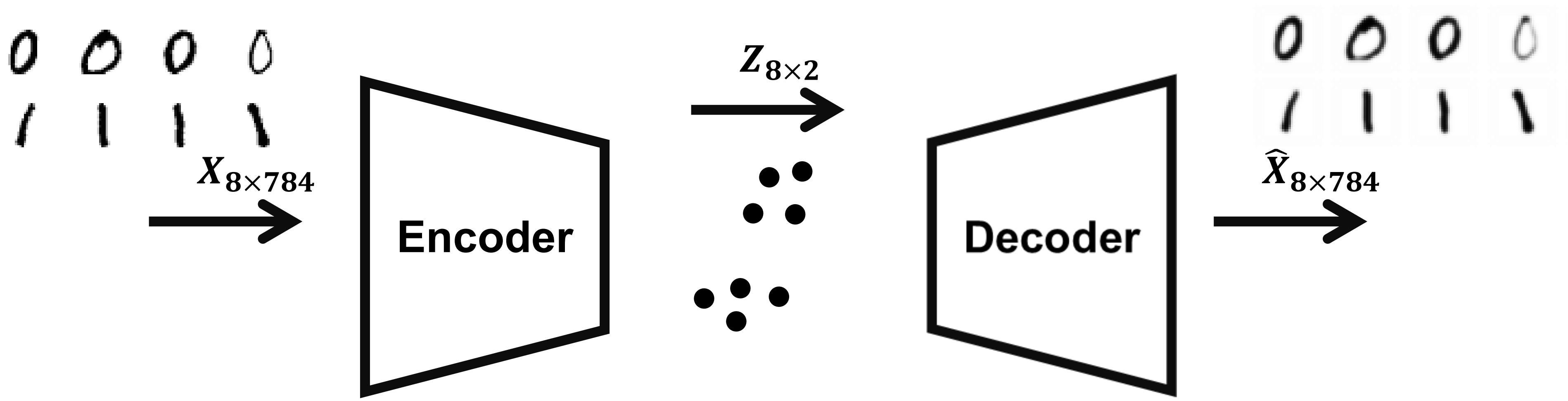
[3] Department of Computer Science, Aalto University, Espoo, Finland

[4] Department of Computer Science, University of Helsinki, Helsinki, Finland

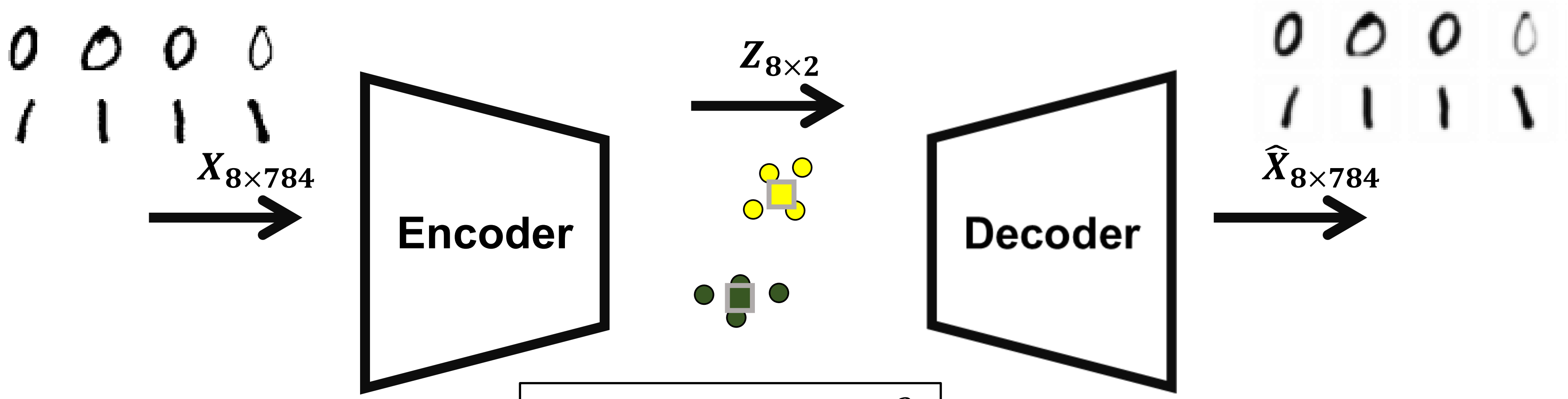
[5] ds:UniVie, Vienna, Austria

[†] Joint first authors, [‡] Joint last authors

# Centroid-based Deep Clustering

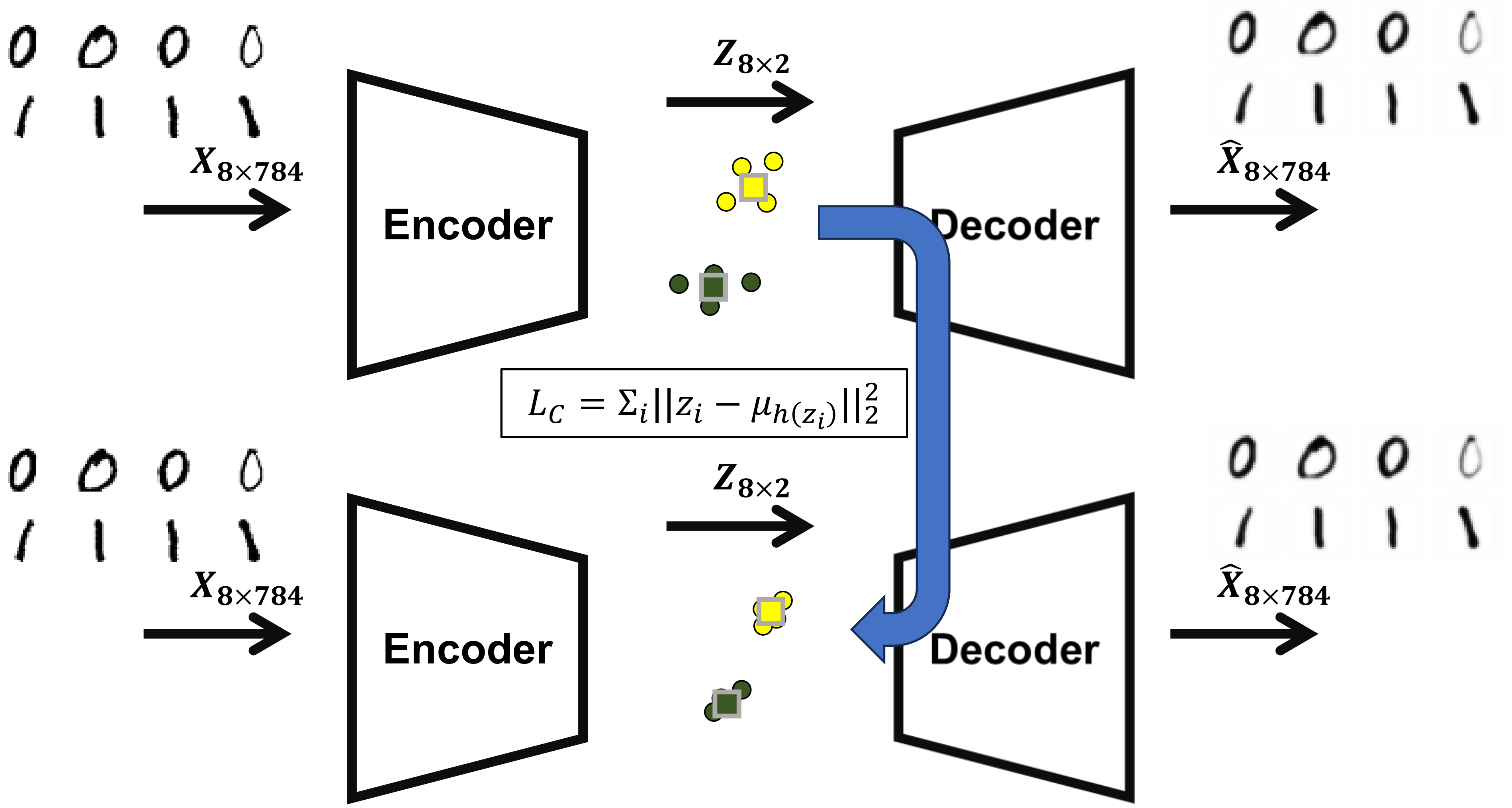


# Centroid-based Deep Clustering

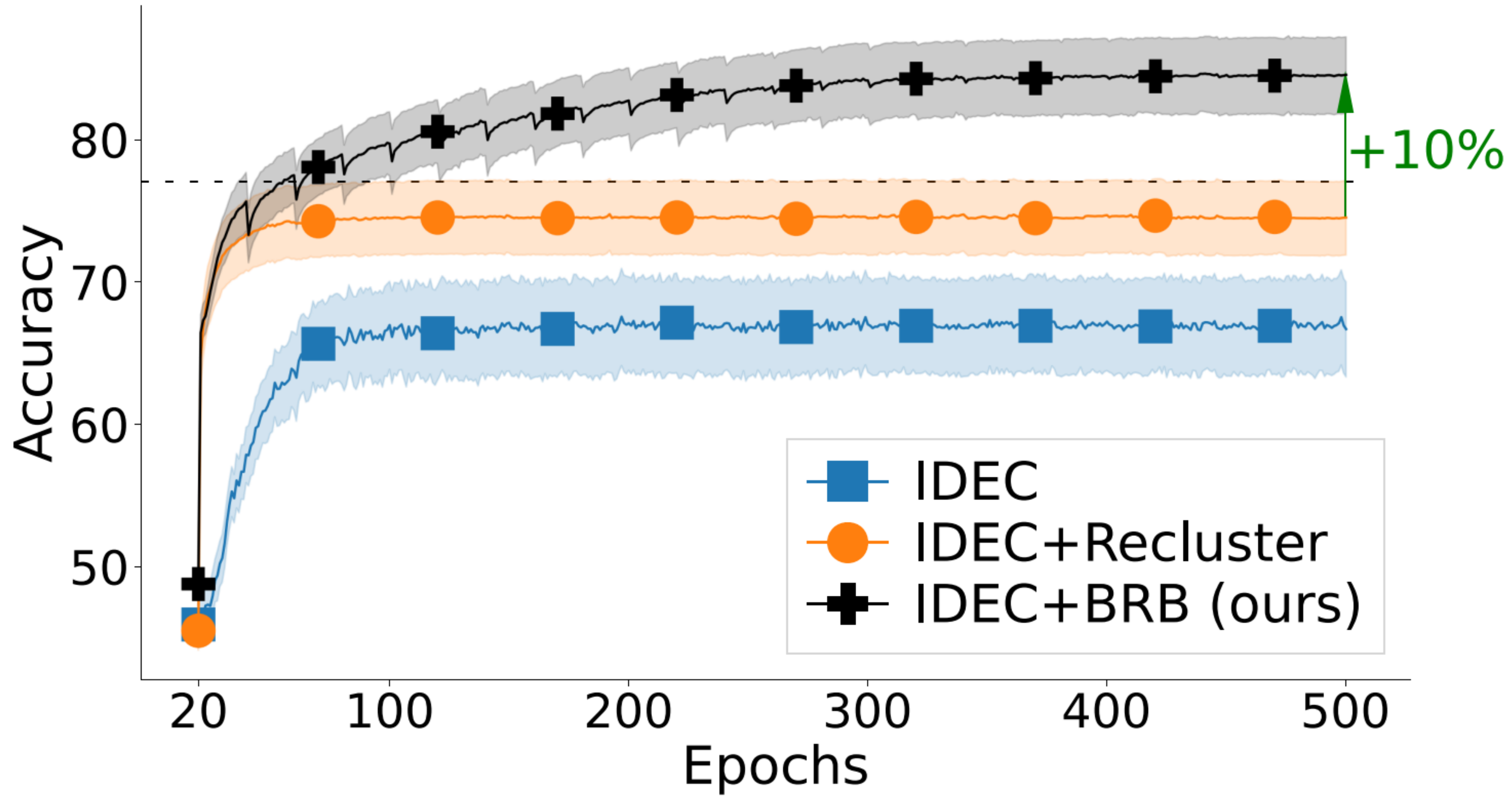


$$L_C = \Sigma_i ||z_i - \boldsymbol{\mu}_{\boldsymbol{h}(\boldsymbol{z}_i)}||_2^2$$

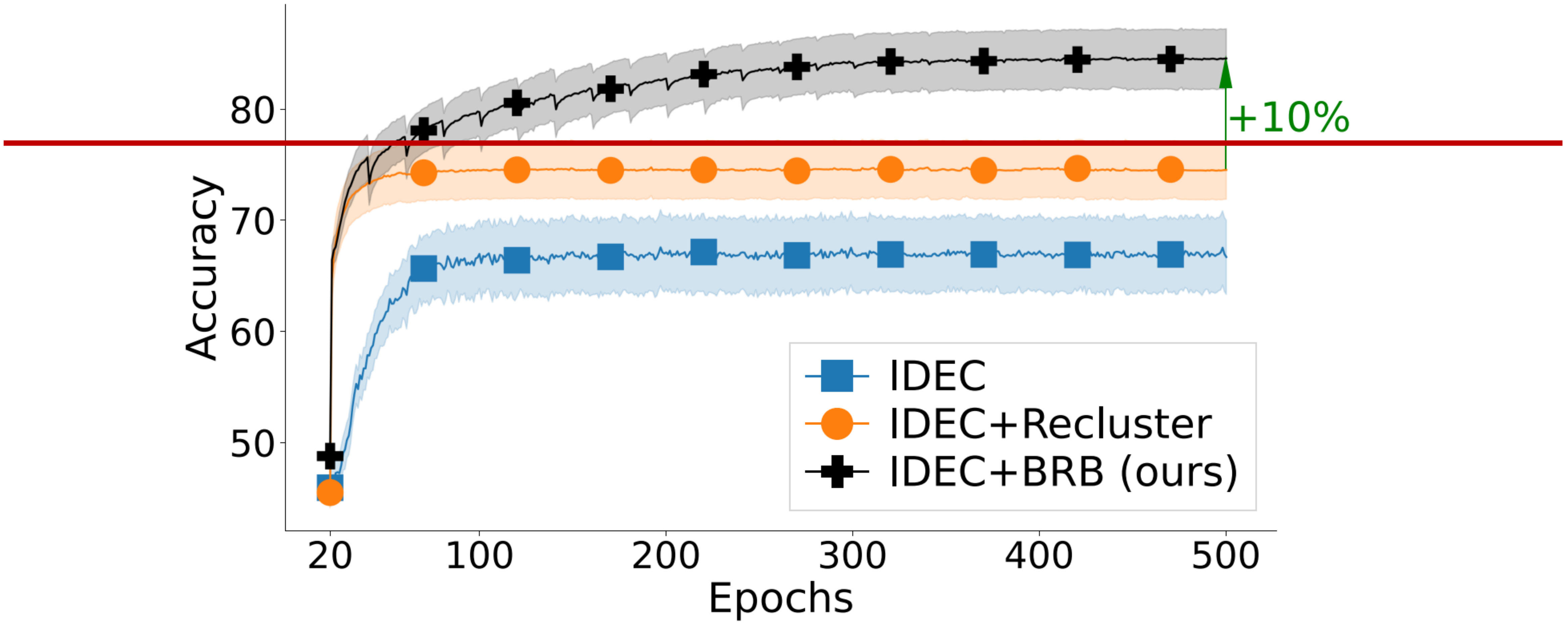
# Centroid-based Deep Clustering



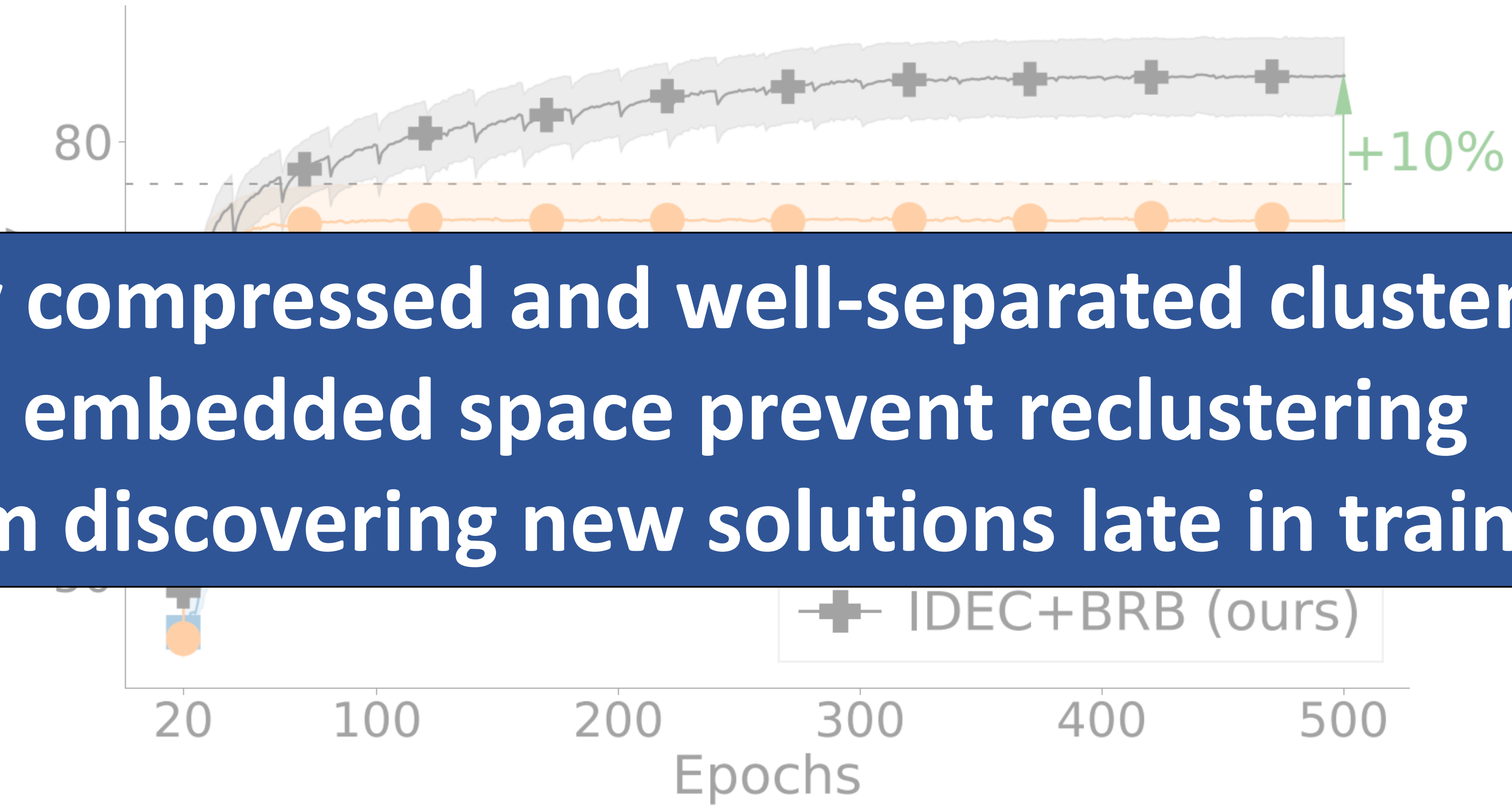
# Reclustering Barrier



## Reclustering Barrier

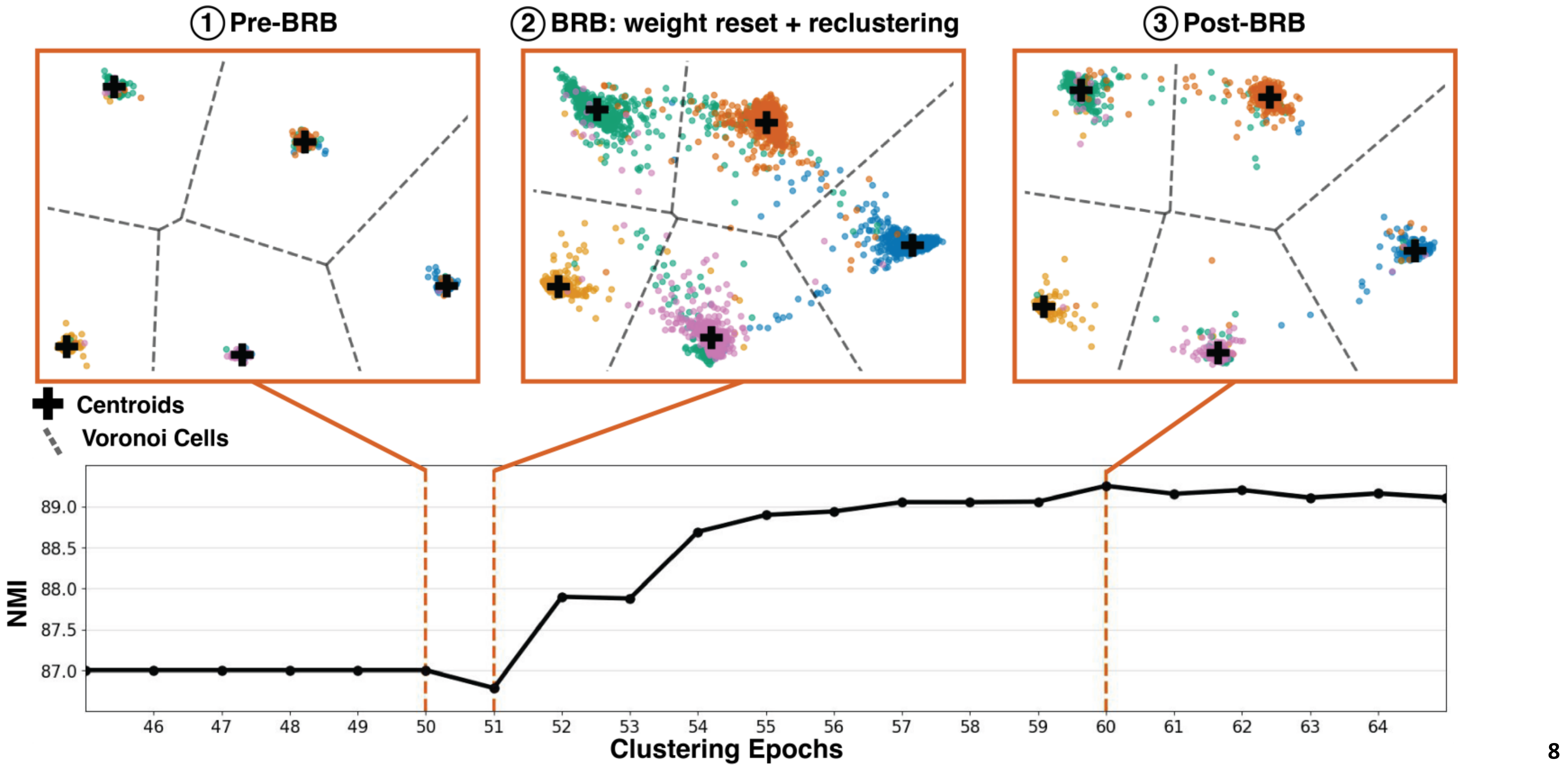


## Reclustering Barrier



**Strongly compressed and well-separated clusters in the embedded space prevent reclustering from discovering new solutions late in training**

# Breaking the Reclustering Barrier (BRB)



## BRB – Weight Reset

Convex combination of network's prior parameters $\theta_t^i$ at time step $t$ and freshly sampled weights $\phi^i$ from initial distribution per layer $i$:

$$\widetilde{\theta_t^i} = \alpha\, \theta_t^i + (1 - \alpha)\phi^i$$

$\alpha \in (0,1)$ is a hyperparameter specifying the reset strength

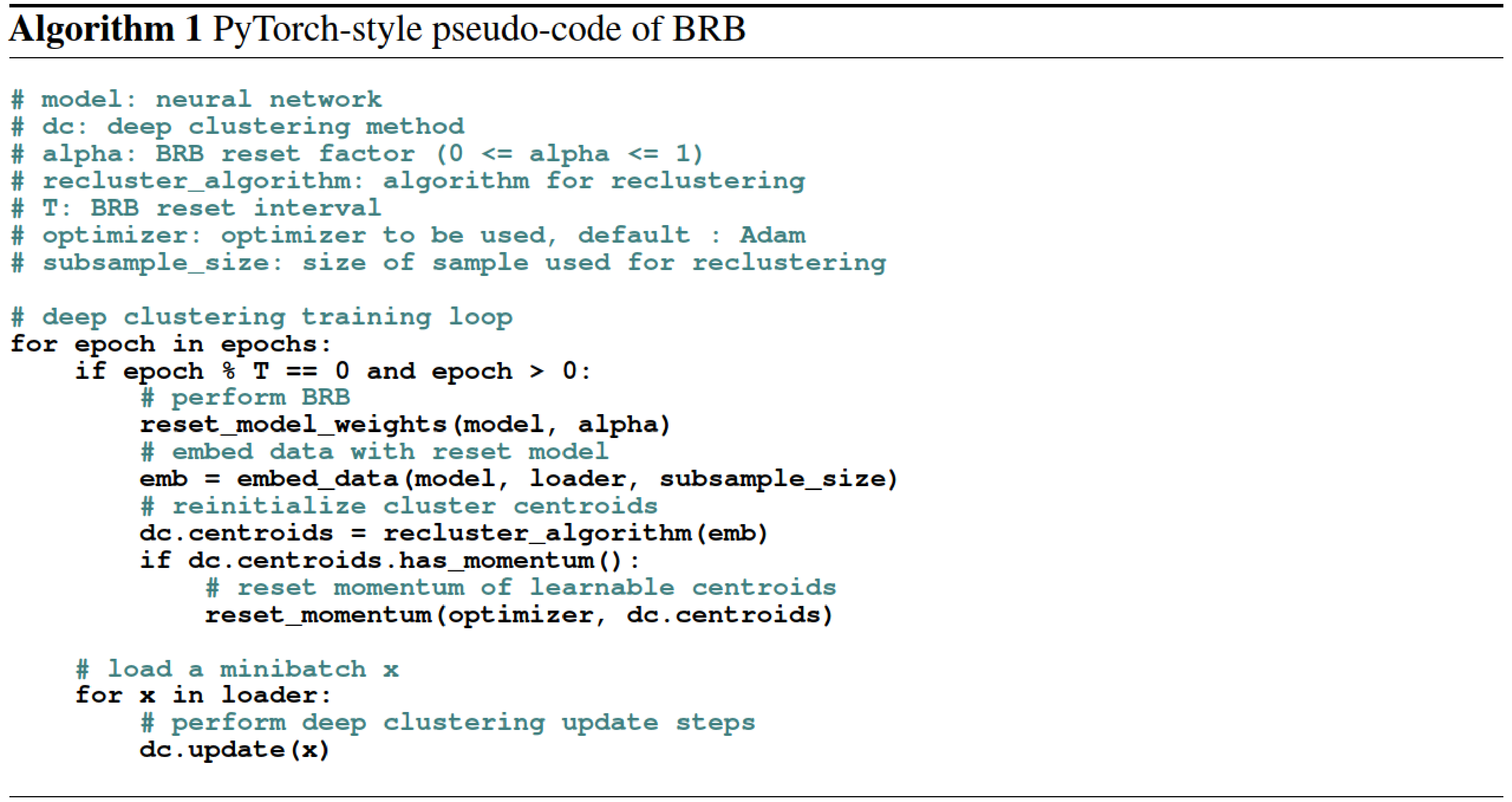
# BRB – Reclustering

After the weight reset cluster centers and assignments are not correct anymore

→Recluster the embedding given by the network with reset weights $f_{\widetilde{\theta}}$ to get new centers and assignments for the deep clustering algorithm

$$k\text{-Means}(f_{\widetilde{\theta}_t}(X))$$

## BRB – Easy to Implement



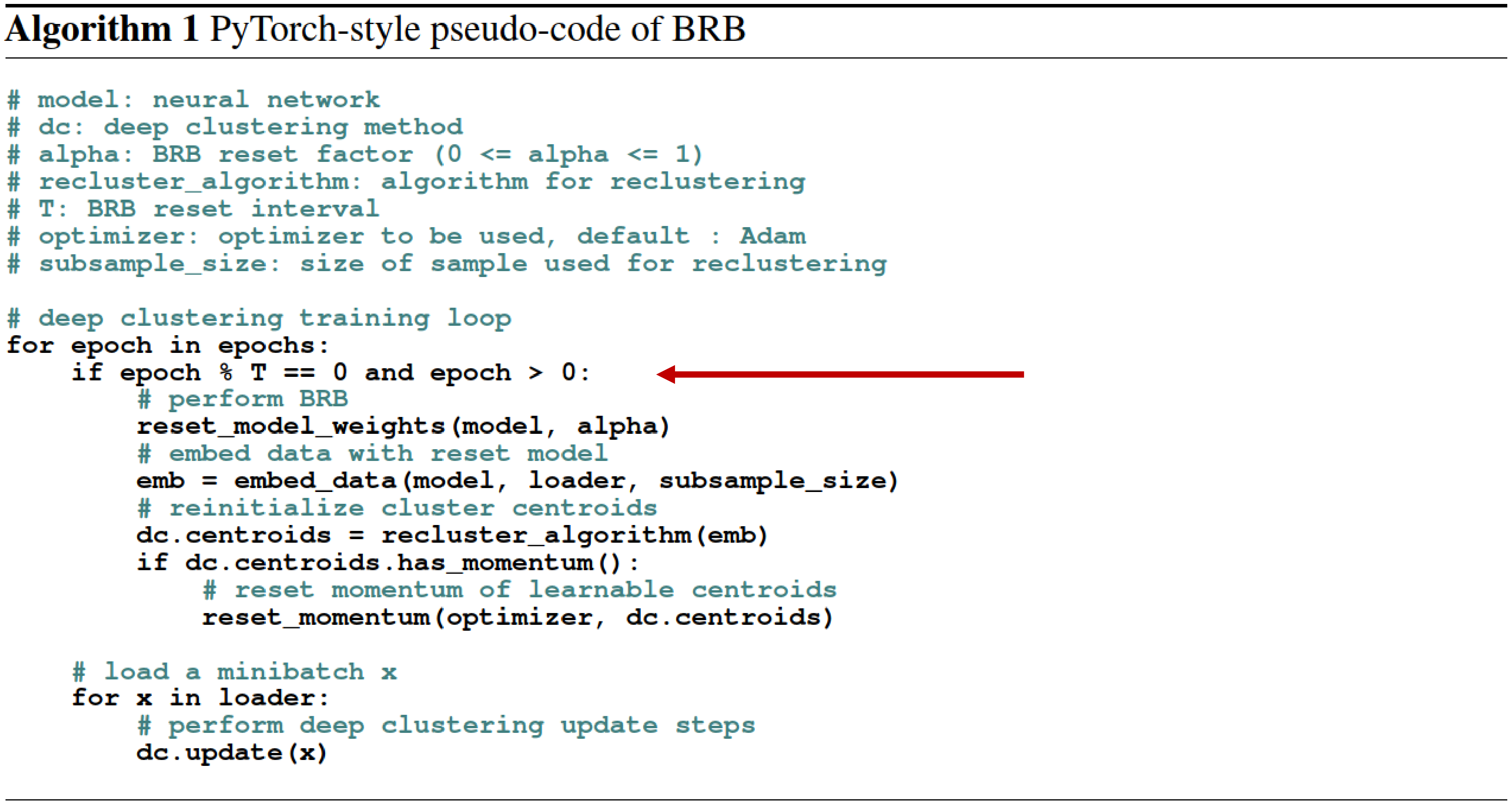
**Algorithm 1** PyTorch-style pseudo-code of BRB

```
# model: neural network
# dc: deep clustering method
# alpha: BRB reset factor (0 <= alpha <= 1)
# recluster_algorithm: algorithm for reclustering
# T: BRB reset interval
# optimizer: optimizer to be used, default : Adam
# subsample_size: size of sample used for reclustering

# deep clustering training loop
for epoch in epochs:
    if epoch % T == 0 and epoch > 0:
        # perform BRB
        reset_model_weights(model, alpha)
        # embed data with reset model
        emb = embed_data(model, loader, subsample_size)
        # reinitialize cluster centroids
        dc.centroids = recluster_algorithm(emb)
        if dc.centroids.has_momentum():
            # reset momentum of learnable centroids
            reset_momentum(optimizer, dc.centroids)

    # load a minibatch x
    for x in loader:
        # perform deep clustering update steps
        dc.update(x)
```

## BRB – Easy to Implement



**Algorithm 1** PyTorch-style pseudo-code of BRB

```
# model: neural network
# dc: deep clustering method
# alpha: BRB reset factor (0 <= alpha <= 1)
# recluster_algorithm: algorithm for reclustering
# T: BRB reset interval
# optimizer: optimizer to be used, default : Adam
# subsample_size: size of sample used for reclustering

# deep clustering training loop
for epoch in epochs:
    if epoch % T == 0 and epoch > 0:
        # perform BRB
        reset_model_weights(model, alpha)
        # embed data with reset model
        emb = embed_data(model, loader, subsample_size)
        # reinitialize cluster centroids
        dc.centroids = recluster_algorithm(emb)
        if dc.centroids.has_momentum():
            # reset momentum of learnable centroids
            reset_momentum(optimizer, dc.centroids)

    # load a minibatch x
    for x in loader:
        # perform deep clustering update steps
        dc.update(x)
```

# BRB – Easy to Implement

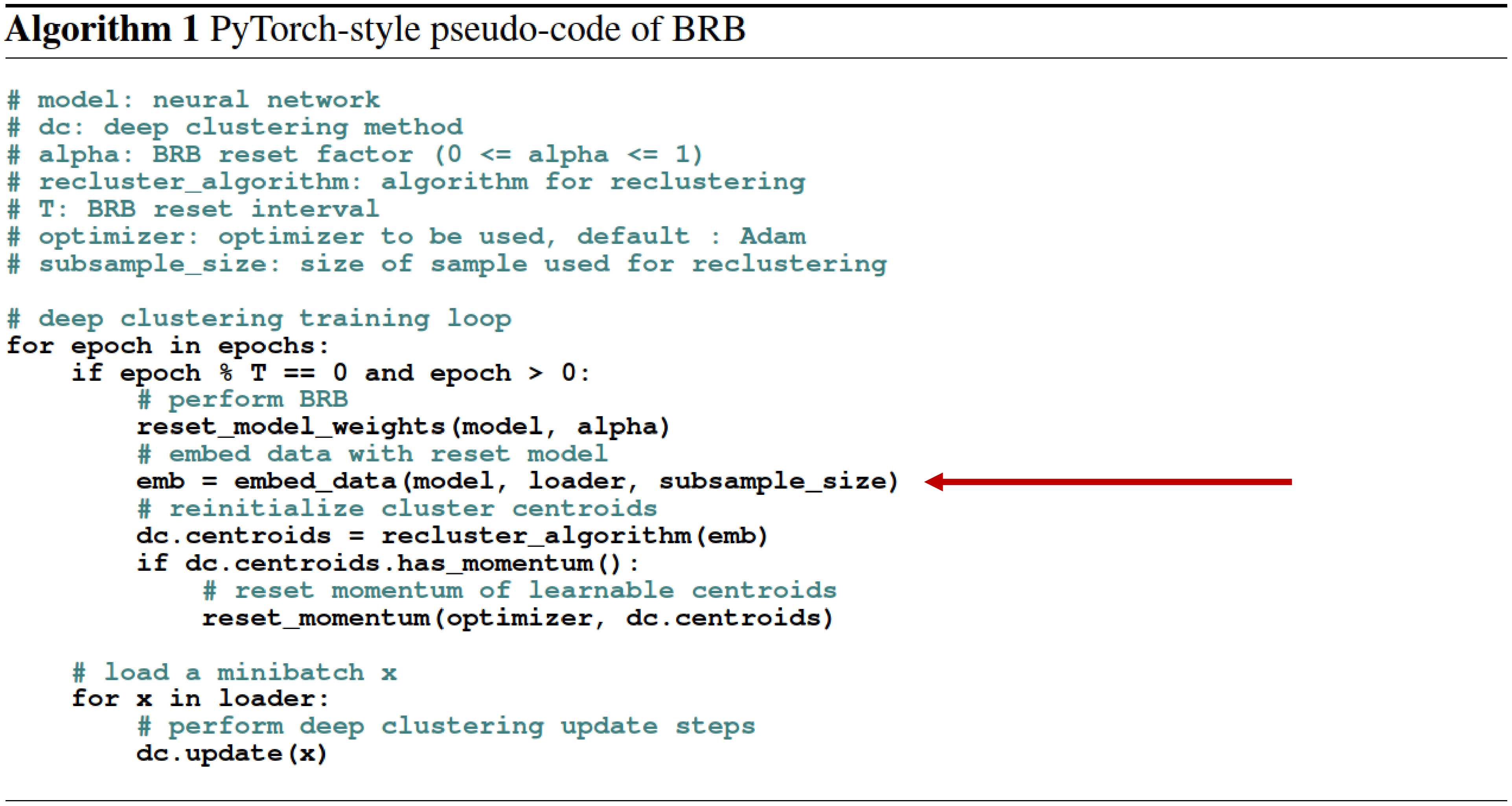
**Algorithm 1** PyTorch-style pseudo-code of BRB

```
# model: neural network
# dc: deep clustering method
# alpha: BRB reset factor (0 <= alpha <= 1)
# recluster_algorithm: algorithm for reclustering
# T: BRB reset interval
# optimizer: optimizer to be used, default : Adam
# subsample_size: size of sample used for reclustering

# deep clustering training loop
for epoch in epochs:
    if epoch % T == 0 and epoch > 0:
        # perform BRB
        reset_model_weights(model, alpha)
        # embed data with reset model
        emb = embed_data(model, loader, subsample_size)
        # reinitialize cluster centroids
        dc.centroids = recluster_algorithm(emb)
        if dc.centroids.has_momentum():
            # reset momentum of learnable centroids
            reset_momentum(optimizer, dc.centroids)

    # load a minibatch x
    for x in loader:
        # perform deep clustering update steps
        dc.update(x)
```

## BRB – Easy to Implement



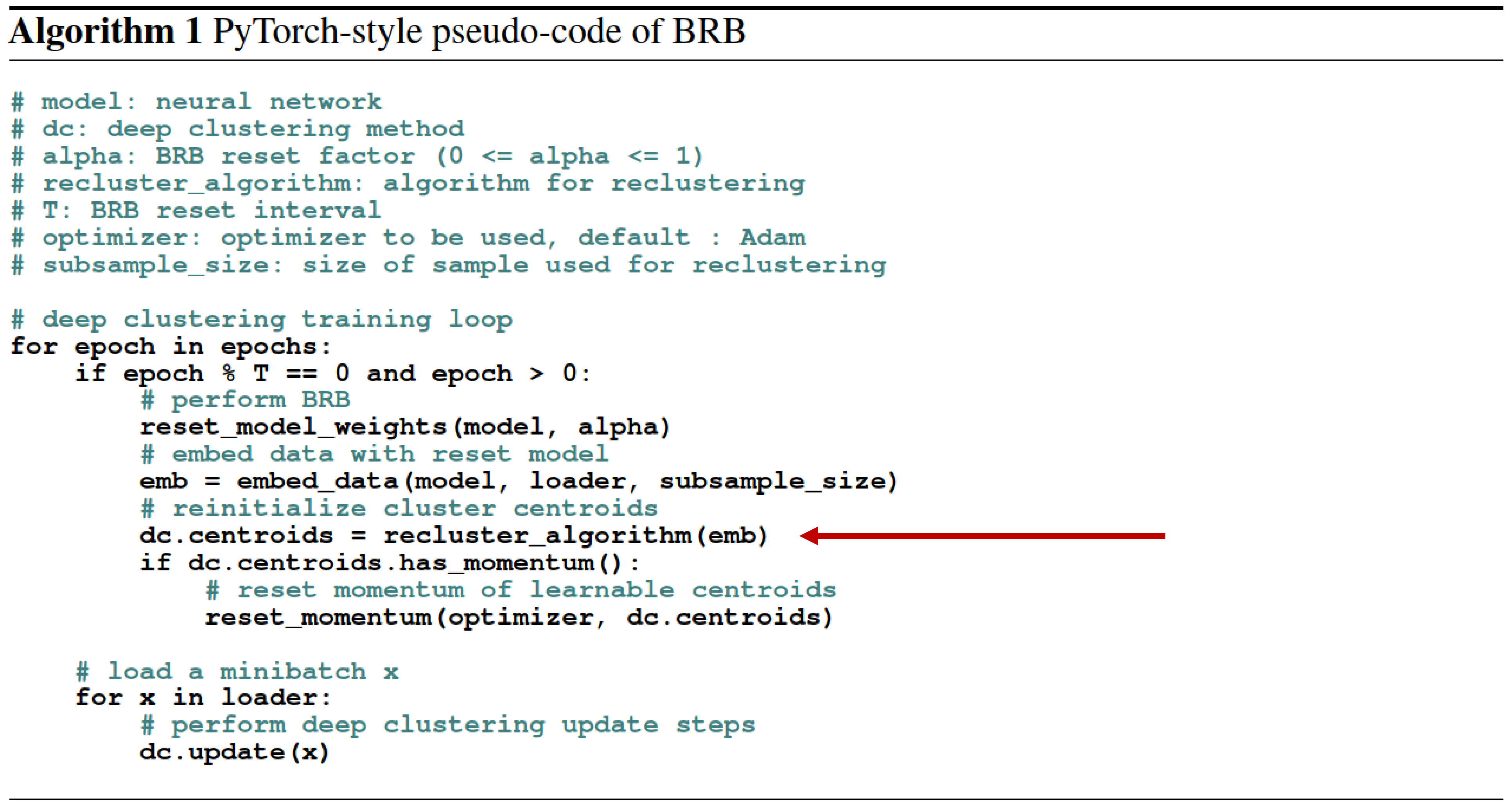
**Algorithm 1** PyTorch-style pseudo-code of BRB

```
# model: neural network
# dc: deep clustering method
# alpha: BRB reset factor (0 <= alpha <= 1)
# recluster_algorithm: algorithm for reclustering
# T: BRB reset interval
# optimizer: optimizer to be used, default : Adam
# subsample_size: size of sample used for reclustering

# deep clustering training loop
for epoch in epochs:
    if epoch % T == 0 and epoch > 0:
        # perform BRB
        reset_model_weights(model, alpha)
        # embed data with reset model
        emb = embed_data(model, loader, subsample_size)
        # reinitialize cluster centroids
        dc.centroids = recluster_algorithm(emb)
        if dc.centroids.has_momentum():
            # reset momentum of learnable centroids
            reset_momentum(optimizer, dc.centroids)

    # load a minibatch x
    for x in loader:
        # perform deep clustering update steps
        dc.update(x)
```

# BRB – Easy to Implement



**Algorithm 1** PyTorch-style pseudo-code of BRB

```
# model: neural network
# dc: deep clustering method
# alpha: BRB reset factor (0 <= alpha <= 1)
# recluster_algorithm: algorithm for reclustering
# T: BRB reset interval
# optimizer: optimizer to be used, default : Adam
# subsample_size: size of sample used for reclustering

# deep clustering training loop
for epoch in epochs:
    if epoch % T == 0 and epoch > 0:
        # perform BRB
        reset_model_weights(model, alpha)
        # embed data with reset model
        emb = embed_data(model, loader, subsample_size)
        # reinitialize cluster centroids
        dc.centroids = recluster_algorithm(emb)
        if dc.centroids.has_momentum():
            # reset momentum of learnable centroids
            reset_momentum(optimizer, dc.centroids)

    # load a minibatch x
    for x in loader:
        # perform deep clustering update steps
        dc.update(x)
```

# BRB – Results

| Methods | CIFAR10 | | | CIFAR100-20 | | |
|---|---|---|---|---|---|---|
| | ACC | NMI | ARI | ACC | NMI | ARI |
| Pretraining + $k$-Means | 68.97 | 63.98 | 40.13 | 37.22 | 42.25 | 14.86 |
| DEC | 88.29 | 80.60 | 77.23 | 50.16 | 51.66 | **35.37** |
| DEC + BRB | **90.57** | **82.57** | **81.18** | **50.46** | **51.72** | 35.05 |
| IDEC | 88.30 | 79.50 | 77.27 | 52.73 | 52.79 | 36.79 |
| IDEC + BRB | **90.72** | **83.26** | **81.81** | **55.43** | **54.81** | **38.81** |
| DCN | 88.55 | 81.02 | 78.17 | 53.27 | 52.13 | 37.30 |
| DCN + BRB | **91.23** | **83.66** | **82.42** | **56.92** | **56.76** | **41.15** |
| SCAN (Gansbeke et al., 2020) | 88.3 | 79.7 | 77.2 | 50.7 | 48.6 | 33.3 |
| GCC (Zhong et al., 2021) | 90.1 | - | - | 52.3 | - | - |
| SeCu (Qian, 2023) | 93.0 | 86.1 | 85.7 | 55.2 | 55.1 | 39.7 |

## Conclusion

**Novel algorithm**: BRB breaks through performance plateaus in centroid-based deep clustering by preserving cluster variation and exploring more diverse solutions.

**Robust Performance**: BRB improves deep clustering across various datasets and algorithms, achieving competitive results with contrastive learning and self-labeling.

**Reclustering barrier**: Our empirical analysis shows that strong initial cluster compression is a key contributor to the reclustering barrier.