# Supplementary: Efficient Unsupervised Visual Representation Learning with Explicit Cluster Balancing

Ioannis Maniadis Metaxas[1]*, Georgios Tzimiropoulos[1], and Ioannis Patras[1]

Queen Mary University of London
{i.maniadismetaxas,g.tzimiropoulos,i.patras}@qmul.ac.uk

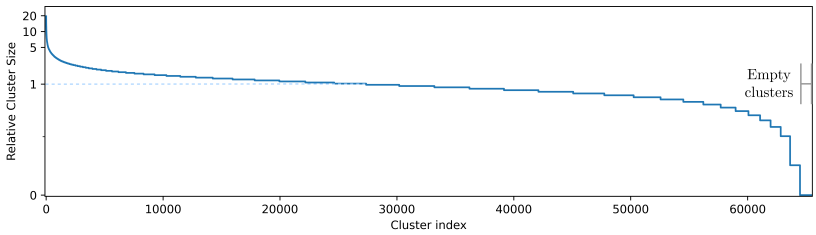
The structure of the supplementary is as follows: In Sec. 1, we study the effectiveness of ExCB's cluster balancing module, by examining the relative cluster sizes *after* training and comparing it with a similar work, DINO [1]. In Sec. 2 the balancing operator $\mathcal{B}$ is analyzed in depth, including the intuition behind our design choices and ablations regarding their effectiveness. Subsequently, in Sec. 3, we present a visualization of ExCB's learned feature representations, demonstrating that classes are well separated despite the training process being unsupervised. Finally, to facilitate understanding of ExCB, we provide pseudo-code for a training step in Pytorch style in Sec. 4.

## 1 Cluster Balancing Effectiveness

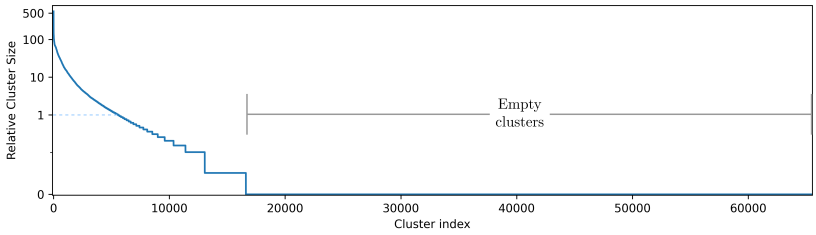
In this section, we expand on the effectiveness of ExCB's cluster balancing method. We present in Fig. 1 the distribution of samples over the clusters for ExCB and contrast it with DINO [1], a landmark work in the area of clustering-based self-supervised frameworks. Both ExCB and DINO pretrain with $K = 65,536$ clusters, which facilitates a fair comparison. Differently from the results presented in the main paper's Sec. 4, where cluster measurements were made during training (i.e. over an epoch with the balancing module active), the measurements presented in Fig. 1 were made via simple inference, using each model's final weights. The assignments are calculated on ImageNet's train set. For DINO we use the publicly available weights provided for a ViT-S/16 backbone trained for 800 epochs with multi-crop.

As seen in Fig. 1, for ExCB only 1.6% of clusters are empty, contrasted with 74.7% for DINO. Additionally, the samples among non-empty clusters are much more evenly distributed, as seen by contrasting Figs. 1a and 1b. These findings complement the ones presented in the main paper's Sec. 4, and reinforce our claims regarding the effectiveness of ExCB's cluster balancing approach, which we consider critical for the state-of-the-art performance and remarkable training efficiency of ExCB. Finally, we believe these findings highlight the effectiveness of our choice to balance clusters based on explicitly measuring their relative size, as opposed to relying on proxy metrics, such as, in the case of DINO, the "centerness" of sample-cluster similarities.

* Corresponding author.




**(a)** Sample distribution for ExCB. 1,056 clusters (1.6%) are empty. The largest cluster has a relative size of 19.5 (i.e. is assigned 382 samples).




**(b)** Sample distribution for DINO. 48,925 clusters (74.7%) are empty. The largest cluster has a relative size of 555.9 (i.e. is assigned 10,946 samples).

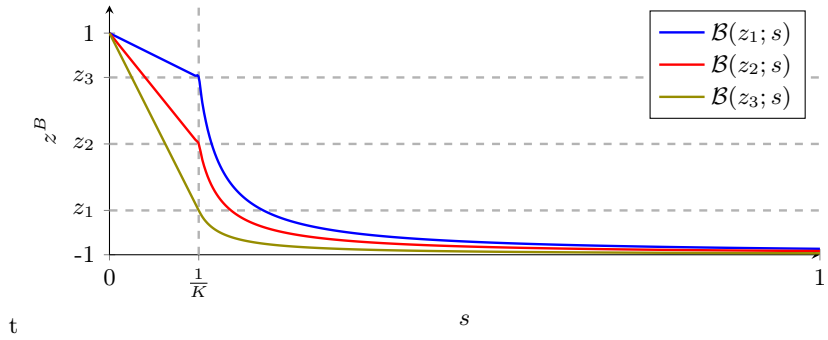
**Fig. 1:** Sample distribution over the clusters for ExCB and DINO. Clusters are sorted according to their relative size, defined as $\frac{N_c K}{N}$, where $N_c$ is the number of samples assigned to that cluster, N=1,281,167 is the number of total samples and K=65,536 is the number of clusters. In each plot, we highlight the optimal relative cluster size of 1 (for $N_c = \frac{N}{K}$) and the empty clusters ($N_c = 0$).

## 2 Balancing Operator $\mathcal{B}$

The balancing operator $\mathcal{B}$ adjusts sample-cluster cosine similarities according to the relative cluster size vector $\boldsymbol{s}$, as shown in the main paper's Eq. 10. Specifically, as the original sample-cluster similarity $z \in [-1, 1]$, in Eq. 10 we first shift it to a positive range of $[0, 2]$, adjust it depending on whether $s > \frac{1}{K}$ or $s < \frac{1}{K}$, and then shift it back to its original range of $[-1, 1]$ to obtain the final similarity value $z^B$.

The impact of $\mathcal{B}$ is illustrated in Fig. 2 for different initial values of $z$ and varying relative cluster sizes $s$. As seen there, for a cluster with optimal size $\frac{1}{K}$, $z^B = \mathcal{B}(z; s) = z$, while for $s < \frac{1}{K}$, $z^B > z$ up to a maximum value of $z^B = 1$ for $s = 0$, and for $s > \frac{1}{K}$, $z^B < z$ down to a minimum value of $z^B = -1$ for $s = 1$. The result of this operation, as explained and demonstrated in the main paper's Sec. 3 and 4 respectively, is that clusters remain approximately balanced throughout training.

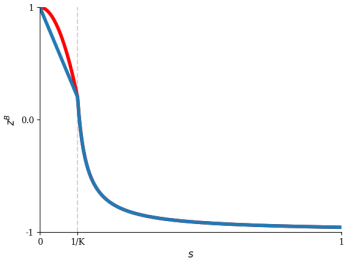
We note here that, as seen in the main paper's Eq. 10 and Fig. 2, $\mathcal{B}$ is linear for $s > \frac{1}{K}$. This choice was made because, for $K = 65,536$, the range of $[0, \frac{1}{K}]$ is very small relative to $[\frac{1}{K}, 1]$. We therefore opted for a less steep function in the expectation that it would lead to more stable training. We validate this




t

**Fig. 2:** The values of $z^B = \mathcal{B}(z; s))$ for indicative sample-cluster similarity values $z_1$, $z_2$ and $z_3$, and for varying relative cluster sizes $s$. Note that $s = 0$ means the cluster has not been assigned any samples (empty cluster), $s = 1$ means the cluster has been assigned all samples (collapse), and $s = \frac{1}{K}$ is the optimal, balanced case.

**Table 1:** Linear evaluation accuracy with ResNet-50, pretrained with multi-crop for 100 epochs, with different functions $\mathcal{B}$ for $s < \frac{1}{K}$. The plot demonstrates the corresponding curves of $\mathcal{B}$ for a given $z$ and for varying values of $s$.

| $\mathbf{B}(z; s)$ if $s < \frac{1}{K}$ | Linear Acc. | Plot |
|---|---|---|
| $1 - [1 - z]\,(sK)^2$ | 74.3 |  |
| $1 - [1 - z]\,(sK)$ | **74.5** | |



intuitive choice experimentally in Tab. 1, where we see that a linear function indeed performs better than an exponential one, although we emphasize that both are stable.

## 3 Embedding Visualization

To complement the results presented in the main paper's Sec. 4, we include a visualization of the features learned by ExCB. Specifically we extract features $(h_t \circ f_t)(x)$ from ImageNet's train set for the 10 classes used in the ImageNet-10 subset [2], and use the t-SNE [3] algorithm to project them in 2 dimensions. We present the outcome in Fig. 3, where samples are colour-coded according to their ground truth class label. We observe that classes are relatively compact to a

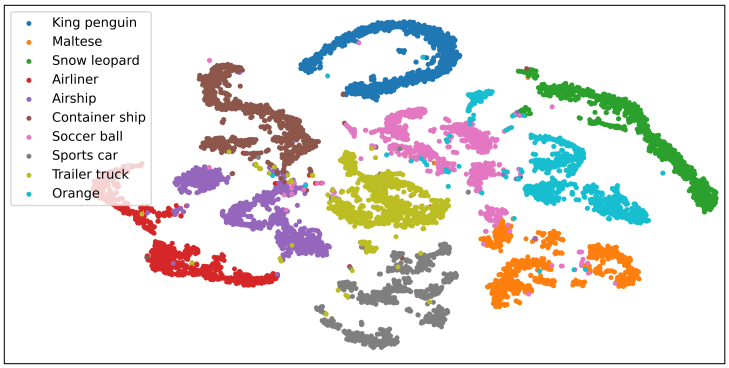



**Fig. 3:** Embedding visualization for 10 classes in ImageNet's train set.

substantial degree, which is notable given that these embeddings are the result of unsupervised pretraining.

Further examining Fig. 3, we note that visually and semantically distinct classes, such as "snow leopard" and "king penguin", are well separated from the others. We observe less clear separation between classes with semantic and visual similarities, such as "container ship", "airship", "airliner" and "trailer truck", which is to be expected, as ExCB is entirely unsupervised. Interestingly, there is also some overlap between the classes "orange" and "soccer ball", which we attribute to the fact that, due to the colour augmentations used during pretraining, self-supervised models (including ExCB) learn representations that are strongly colour-invariant and rely to a larger extent on shapes to infer semantic information.

## 4 Algorithm

In order to facilitate understanding of ExCB, as described in the main paper's Sec. 3, we provide pseudo-code for ExCB's training steps in Pytorch style in Algorithm 1. Distinctly, to illustrate the detach operation applied to centroids for local crops, we present the forward function of the centroid layer $q_s/q_t$ in Algorithm 2.

**Algorithm 1** A training step of ExCB during pretraining with multi-crop.

```
# x: A batch of N images
# fs, hs, gs, qs: The student model's components
# gt, ht, qt: The teacher model's components
# s: The relative cluster size measuring vector
# m, m_s: The momentum parameters for the model and the balancing operator
# t_t, t_s: The temperature parameters for the teacher and student
# G, L: The number of global and local views

x_G, x_L = augment_G(x), augment_L(x) # Global/Local views of samples x

# Sample-cluster similarities zh for the projector hs
z_hG = qs(hs(fs(x_G)), detach=False)
z_hL = qs(hs(fs(x_L)), detach=True)

# Sample-cluster similarities zg for the predictor gs
z_gG = qs(gs(hs(fs(x_G))), detach=True)
z_gL = qs(gs(hs(fs(x_L))), detach=True)

# Balanced sample-cluster similarities zb for the teacher
with torch.no_grad():
     z_t = qt(ht(ft(x_G)))
     z_b = balancing(z_t)

# Cluster probability assignments for the teacher and student
p_t = softmax(z_b/t_t) # G×N×K
p_h = softmax(cat(z_hL, z_hG)/t_s) # G×N×K
p_g = softmax(cat(z_gL, z_gG)/t_s) # L×N×K

# Calculation of the loss
loss_g, loss_h = 0, 0
for v in range(G):
     for vg in range(G):
         if v!=v_g
             loss_h += mean((p_t[v]*log(p_h[vg])).sum(dim=1))
         loss_g += mean((p_t[v]*log(p_g[vg])).sum(dim=1))
     for vl in range(L):
         loss_h += mean((p_t[v]*log(p_h[vl])).sum(dim=1))
         loss_g += mean((p_t[v]*log(p_g[vl])).sum(dim=1))
loss = 0.5 * loss_h / (G+L-1) + 0.5 * loss_g / (G+L)

# Student and teacher updates
loss.backward()
update(fs, hs, gs, qs)
for ms, mt in zip([fs, hs, qs], [ft, ht, qt]):
     mt = mt * m + ms * (1-m)


def balancing(z_t):
     batch_labels = z_t.argmax(dim=-1)
     s_b = normalize(one_hot(batch_labels), p=1)
     s = s * m_s + s_b * (1-m_s)
     z_b = 1+(1-z_t)*(1-relu(1-sK)) # Efficient implementation of ℬ for s<1/K
     z_b = (1+z_b)*(1-relu(1-1/sK))-1 # Efficient implementation of ℬ for s>1/K
     return z_b
```

**Algorithm 2** The forward function for centroid layers $q$.

```
# K, D, B: The number of clusters, input vector size and batch size
# C: The cluster centroids' K × D matrix
# x: The centroid layer's B × D input

def forward(x, detach=False):
     xn = F.normalize(x, p=2, dim=-1)
     if detach:
         Cn = normalize(C.detach(), p=2, dim=-1)
     else:
         Cn = normalize(C, p=2, dim=-1)
     return xn@Cn.T
```

## References

1. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
2. Chang, J., Wang, L., Meng, G., Xiang, S., Pan, C.: Deep adaptive image clustering. In: Proceedings of the IEEE international conference on computer vision. pp. 5879–5887 (2017)
3. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)