ATLANTIS: A Framework for Automated Targeted Language-guided Augmentation Training for Robust Image Search *Supplementary materials*

Singh *et al*. https://github.com/intherejeet/ATLANTIS

In this supplementary material, we provide additional details, ablations, and results related to our paper. In section A, we describe our main evaluation metric, Recall@K. In section B, we introduce zero-shot and full-shot learning tasks in image retrieval. Section C details the functions used in ATLANTIS's Data Insight Generator (DIG) and Augmentation Protocol Selector (APS) components. Section D presents the detailed experimental settings for the Stanford Online Products (SOP) dataset [6]. Section E verifies the effectiveness of ATLANTIS's pipeline against conventional approaches and highlights the gains in synthetic data augmentation efficiency. Section F provides experiments for the training feedback component in ATLANTIS. In section G, we present detailed results and analysis for different values of the diversity factor Δ used in synthetic data filtering. Section H offers detailed adversarial robustness evaluations, including results for adversarial attacks with larger noise sizes. In section I, we present results for the case when novel classes were augmented by ATLANTIS through synthetic data. Finally, in section J, we present processing speed and computational complexity of ATLANTIS.

A Evaluation Metric: Recall@K

We utilise the standard evaluation metrics in DML: Recall@K (R@K) [2] with $k = \{1,2,4,8\}$ for the CUB-200-2011 [8] and Cars196 [4] data, and $k = \{1,10,100,\ 1000\}$ for the SOP data [6]. An increase in $R@K$ indicates improved image retrieval performance of the trained model.

A.1 Definition

For a given DML function f, let $\mathcal{F}_q^k$ denote the set of the first k nearest neighbours of a sample $x_q \in \mathcal{X}_{\text{test}}$, defined as

$$\mathcal{F}_q^k = \underset{\mathcal{F}\subset\mathcal{X}_{\text{test}},|\mathcal{F}|=k}{\arg\min} \sum_{x_n\in\mathcal{F}} d\left(f(x_q), f(x_n)\right) \tag{1}$$

Recall@K is then calculated as

$$R@K = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x_q\in\mathcal{X}_{\text{test}}} \begin{cases} 1 & \text{if } \exists\, x_i \in \mathcal{F}_q^k \text{ such that } y_i = y_q \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This implies that Recall@K measures the proportion of cases where, for a given query x_q, at least one sample among its top k nearest neighbours x_i belongs to the same class, i.e., $y_i = y_q$.

B Zero-Shot and Full-Shot Learning Settings

DML is designed to position semantically similar instances closer in the learned space than dissimilar ones, facilitating its application to Zero-Shot Learning and Full-Shot Learning settings. These settings enable models to generalise to unseen classes or to exploit the entirety of the available training data. The following is an overview of Zero-Shot and Full-Shot Learning within the DML context.

B.1 Zero-Shot Learning (ZSL)

ZSL enables the recognition of objects unseen during training by utilising semantic relationships between categories, thereby bridging the semantic gap between seen and unseen classes through visual-semantic embeddings and adaptive metric learning [3, 9]. In all our experiments, we adhered to the standard benchmark 50:50 train-test split used in prior works [1].

B.2 Full-Shot Learning (FSL)

Contrary to ZSL, FSL utilises the same set of classes for both the training and test phases, thus exploiting all labelled training data across known classes. This ensures comprehensive class representation for detailed feature extraction and class distinction. In this context, we adhered to the standard 50:50 train-test split, performing an intra-class, sample-wise split to maintain a balanced class distribution across the training and testing sets with non-overlapping samples.

C Details of the Functions Used in Different ATLANTIS Components

C.1 Data Insight Generator (DIG) Functions

The functions `FrequencyEval`, `ContextMaker`, `DomainInfer`, `ClassInfer`, `NewClassSearch`, and `InsightExtract` each utilises a predefined objective to guide the input large language model (LLM) in executing their respective tasks. The description of each function is as follows:

C.1.1 `FrequencyEval` Function.

`FrequencyEval` analyses informative tokens, focusing on nouns and verbs after stop-word removal, utilising heuristics and POS tagging to highlight key tokens. For example, in the CUB-200-2011 data [8], it may identify bird species names, "perching," "swimming," and "flying" as among the top informative terms, offering insights into the subjects of the data and their behaviours. The flexibility in the number of terms extracted, such as the top 50, allows for customisation based on the data's specificity, underscoring `FrequencyEval`'s adaptability to diverse data types.

C.1.2 `ContextMaker` Function.

`ContextMaker` utilises outputs from `FrequencyEval` and metadata, employing an LLM with a master prompt tailored to generate a nuanced data context. This integration forms a cohesive overview, spotlighting domain-specific features for subsequent analysis. An example master objective for the LLM looks like as follows:

```
"You are a sophisticated AI model...trained to discern
context and patterns...Generate a comprehensive
data context while identifying and including
information important for data retrieval tasks, domain
identification...quantify occurrences..."
```

C.1.3 `DomainInfer` Function.

Infers the domain distribution across the data using an LLM by leveraging the contextual information generated by `ContextMaker`, identifying primary domains that encapsulate the data's semantic diversity. The LLM is defined with an objective like as follows:

```
"You are a sophisticated AI model...trained to discern
context and patterns...From the provided data context
information...Deduce key semantic domains from
informative tokens...list 3 key domains based on the
identified domain criteria..."
```

C.1.4 `ClassInfer` Function.

In supervised training with metadata labels, class distribution and imbalance analysis are straightforward, using heuristics for class-wise frequency. Without explicit labels, this function utilizes $\mathcal{C}$ and an LLM to identify key classes and highlight those with few samples, addressing class imbalances.

C.1.5 `NewClassSearch` Function.

The `NewClassSearch` function searches for novel classes $\mathcal{N}$. Utilising an LLM and providing it with $\mathcal{I}_{\text{class}}$ and $\mathcal{R}$ as inputs, it generates new, practical classes that are not present in the data but should be included based on data characteristics and the training objective. An example LLM objective for the CUB-200-2011 data is structured as follows:

```
"You are an advanced Language Model with a
specialisation in ornithology, specifically in
the augmentation of bird species...with extensive
experience in large-scale image datas...From the
provided data context and class information...Generate
10 new, practical, yet previously not included
classes...These new species should be entirely
distinct from the input species in terms of visual
characteristics and features..."
```

The new classes for the CUB-200-2011 data were generated as follows:

```
{
      "data_overview": "The dataset consists of images of two bird classes with captions. The
            Baltimore   Oriole is represented in three different perching scenes, whereas the Laysan
            Albatross is depicted in a single flying scene against a cloudy backdrop.",
      "class_imbalance_recommendations": "To maintain equity between classes, augment the dataset with
            additional Laysan Albatross images showing various activities and settings. Consider
            synthesizing Baltimore Oriole images in equally diverse scenarios to match.",
      "domain_imbalance_recommendations": {
            "Baltimore Oriole": "Increase domain variety by introducing images depicting varied
                  activities such as feeding or in flight, and set against a wider range of
                  backgrounds including urban settings and different weather conditions.",
            "Laysan Albatross": "Augment the single representation with images covering a spectrum of
                  behaviors and environments, like different flight phases, resting, varied times of
                  day, and interactions with the sea and other marine life."
            },
      "class_names": [
            "Baltimore Oriole",
            "Laysan Albatross“
            ]
}
```

Figure 1: An example output from DIG for a subset of CUB-200-2011 dataset.

`"[African Grey Parrot, Atlantic Puffin, Bald Eagle, Emperor Penguin, Golden Eagle, Harpy Eagle, Peacock, Peregrine Falcon, Scarlet Macaw, Snowy Owl]"`

Though the generation of new classes significantly diversifies the data augmentation, it introduces complications in the ORDC filtering due to incompatibility, hence they are required to be included without filtering, with the exception.

C.1.6 `InsightExtract` Function.

Compiles and formats all the information extracted in prior steps to make it compatible with subsequent components' inputs.

$\{D_{\text{insight}} : \{\mathcal{M} : \{$`Metadata Info.`$\},$
$\mathcal{I}_{\text{domain}} : \{$`Domain Imabalance Info.`$\},$
$\mathcal{I}_{\text{class}} : \{$`Class Imabalance Info.`$\},$
$\mathcal{N} : \{$`List of New Classes`$\}\}\}$

Figure 1 illustrates an example output from the DIG component for a subset of the CUB-200-2011 dataset. The output includes a comprehensive data overview D_{insight}, recommendations for addressing class imbalance $\mathcal{I}_{\text{class}}$ by augmenting images of the Laysan Albatross and Baltimore Oriole, and domain imbalance recommendations $\mathcal{I}_{\text{domain}}$ to increase variety in depicted scenes and activities. The class names "Baltimore Oriole" and "Laysan Albatross" are identified, with specific augmentation suggestions aimed at enhancing dataset diversity and addressing identified weaknesses.

C.2 Augmentation Protocol Selector (APS) Functions

C.2.1 `AugmentationObjective`.

This function leverages an LLM with a defined objective to develop a data augmentation goal $\mathcal{O}_{\text{aug}}$ for a downstream LLM responsible for generating text descriptions of the target

Algorithm 1 Filtering Feedback and APS's Protection from Infinite Loop

```
Require: Domain imbalance threshold $\tau_{\text{domain}}$, Class imbalance threshold $\tau_{\text{class}}$, Predefined
    Exceptions $\mathcal{D}_{\text{exempt}}$
Ensure: Alert list $\mathcal{A}$ for APS
 1: Initialise $\mathcal{A}$ to an empty list
 2: Initialise feedback_iteration to 1
 3: if domain imbalance (filtered out fraction from a domain $> \tau_{\text{domain}}$) then
 4:     Add domain imbalance alert (with domain info and imbalance amount) to $\mathcal{A}$
 5: if class imbalance (filtered out fraction from a class $> \tau_{\text{class}}$) due to filtering then
 6:     Add class imbalance alert (with class info and imbalance amount) to $\mathcal{A}$
 7: for each alert $a$ in $\mathcal{A}$ do
 8:     if $a$ is in $\mathcal{D}_{\text{exempt}}$ then
 9:         Remove $a$ from $\mathcal{A}$
10: if $\mathcal{A}$ is empty then
11:     Send no alert to APS
12: else
13:     Send alert list $\mathcal{A}$ to APS
```

synthetic data. The defined objective is as follows:

```
"While serving as a selector for the image
retrieval synthetic data augmentation protocol, the
system...Defines an objective prompt for another
LLM tasked with generating text descriptions of
target synthetic images...based on the input insights
$D_{\text{insight}}$..."
```

C.2.2 `ReferenceDescription`.

This function generates the reference description $\mathcal{D}_{\text{ref}}$ of the target data, upon which the augmentation objective $\mathcal{O}_{\text{aug}}$ is executed by an LLM. An example is as follows:

```
"The bird species is Bard Eagle.  This class has no
image in our data hence the generated images should
be represent diverse domain scenarios as per the
characteristics of this bird...."
```

C.2.3 `IncorporateFeedback`.

This function incorporates feedback from data filtering and training performance during the later stages of DML model training or fine-tuning. While training feedback is elaborated in the main paper, the filtering feedback mechanism is described in algorithm 1.

Figure 2 illustrates an example of APS-guided generated text descriptions of the target synthetic images in ATLANTIS for a subset of the CUB-200-2011 dataset. The text descriptions are generated for two bird classes: Baltimore Oriole and Laysan Albatross. Each description captures diverse activities and environments, such as a Baltimore Oriole plucking insects from a city park oak tree and a Laysan Albatross performing a banking manoeuvre under the setting sun. These targeted text descriptions are used to create synthetic images

```
{
      "Baltimore Oriole":[
            0: "A Baltimore Oriole skillfully plucking insects from the bark of a city park oak tree,
            cityscape in the blurred background."
            1: "A sleek Baltimore Oriole braving a rainstorm, droplets cascading off its vibrant
            orange plumage as it perches on a suburban telephone wire."
      ]
      "Laysan Albatross":[
            0: "A Laysan Albatross in a dynamic banking maneuver under the radiant hues of a setting
            sun over the Pacific Ocean."
            1: "Solitary Laysan Albatross captured at the moment of touchdown on the choppy sea
            surface, wings widespread to brace the impact."
            2: "A vigilant Laysan Albatross perched on a rocky shore, surveying the coastline under
            the soft glow of dawn."
            3: "An inquisitive Laysan Albatross gracefully skimming over coral reefs, its reflection
            mirrored on the tranquil turquoise waters, interacting with curious tropical fish."
      ]
}
```

Figure 2: An example of APS-guided generated text descriptions of the target synthetic images in ATLANTIS for a subset of CUB-200-2011 dataset.

that address identified weaknesses in the original dataset, thereby enhancing data diversity and model robustness.

D Experiments for SOP Dataset

The SOP data encompasses 22,634 classes across 12 product categories, totalling 120,053 eBay product images. It is partitioned into two subsets: the first 11,318 classes, comprising 59,551 images for training, and the subsequent 11,316 classes, with 60,502 images, designated for testing.

Given the significant training complexity and computational overhead associated with synthetic data generation due to the extensive number of classes and samples, we have constrained our data augmentation through ATLANTIS to assess its efficacy. Although LLMs for generating consistent product image descriptions—whether pertaining to real or fictitious products—is relatively straightforward, we observed that current text-to-image models lack the requisite precision to reliably produce product images across a broad range of scenarios. Consequently, our reliance on well-know products to insure current text-to-image models can correctly generate their images, was necessitated.

To address this, we exploited the DIG component within ATLANTIS to introduce eight novel product categories: `{Office Supplies, Cookware, Sunglasses, Foot wear, Sports Equipment, Musical Instruments, Furni-ture, Televisions}`. For each category, we generated ten classes, and within each class, we created 20 captions, subsequently synthesising images for each. As a result, the original data was augmented with 80 new classes and 1600 synthetic images, representing a modest increment relative to the SOP data's original size. Still, we found ATLANTIS yielding performance improvements surpassing the current state-of-the-art.

E Verifying Effectiveness of ATLANTIS Pipeline

Though, ATLANTIS is the first to provide an automated and efficient solution for improving image search models through targeted synthetic data augmentation, the framework components like DIG can also be used by end users for understanding and analysing their data. To confirm the superiority of ATLANTIS's pipeline in performing targeted synthetic data augmentation, we still compare our results with existing untargeted augmentation approaches to justify the presence and effectiveness of the *in-series* components: DIG, APS, and ORDC. We compare ATLANTIS against the naive-untargeted augmentation in input data with domain scarcity and class-imbalance scenarios.

E.1 Experimental Setting

We keep the experimental settings for the baseline (DINO$_H$) and ATLANTIS (**A**-DINO$_H$) consistent with those described in Tables 1 and 2 of the main draft, focusing on input data with domain scarcity and class imbalance scenarios, respectively. For the naive-untargeted augmentation (**NA**-DINO$_H$) case, we perform random augmentation of synthetic images generated using the SDXL-1.5 32-bit model [7] (the same text-to-image model used in all our other experiments) without considering domain scarcity and class imbalance in the original data. We maintain the number of synthetic augmented images similar to that in ATLANTIS's augmentation for both domain-scarce and class-imbalance settings to ensure a fair comparison.

E.2 Ablation Results

We conducted a comprehensive ablation study to verify the effectiveness of the ATLANTIS pipeline. Our experiments focused on evaluating the performance under domain scarcity and class imbalance scenarios.

Domain Scarcity: Table 1 compares the performance of the baseline (DINO$_H$ without any augmentation), naive-untargeted synthetic augmentation (**NA**-DINO$_H$), and ATLANTIS augmented (**A**-DINO$_H$) models on CUB-200-2011 and Cars196 datasets. In scenarios with domain scarcity, **A**-DINO$_H$ significantly outperforms both DINO$_H$ and **NA**-DINO$_H$. For instance, on the CUB-200-2011 dataset with the Φ domain, **A**-DINO$_H$ achieves R@1 of 68.4%, compared to 64.2% for DINO$_H$ and 65.1% for **NA**-DINO$_H$. Similarly, in the Cars196 dataset, **A**-DINO$_H$ reaches R@1 of 75.4%, outperforming DINO$_H$ (70.3%) and **NA**-DINO$_H$ (67.1%). The similar patterns were observed for the remaining cases with different domain scarcities. These results demonstrate that while naive-untargeted augmentation (**NA**-DINO$_H$) *can* provides some improvement over the baseline, it is not sufficient to address domain scarcity effectively. The targeted approach of ATLANTIS augmentation proves more effective in resolving domain-specific deficiencies.

Class Imbalance: Table 2 illustrates the performance comparison under class imbalance conditions. The **A**-DINO$_H$ consistently surpasses DINO$_H$ and **NA**-DINO$_H$ across various metrics. For example, with $\kappa = 50$ classes in the CUB-200-2011 dataset, **A**-DINO$_H$ attains an R@1 of 76.5%, whereas DINO$_H$ and **NA**-DINO$_H$ record 71.0% and 74.9% respectively. For $\kappa = 75$ classes of the CUB-200-2011 data, **A**-DINO$_H$ achieves an R@1 of 76.0%, significantly higher than 64.3% for DINO$_H$ and 73.3% for **NA**-DINO$_H$. The simialar pattern was also observed for the Cars196 data. These findings confirm that naive-untargeted augmentation (**NA**-DINO$_H$) fails to adequately address class imbalance, whereas the targeted augmentation strategy of ATLANTIS effectively resolves these issues and leads to superior performance.

Overall, the results clearly indicate that ATLANTIS provides substantial improvements in both domain scarcity and class imbalance scenarios compared to the naive-untargeted synthetic data augmentation, even with state-of-art diffusion models. This demonstrates the necessity and effectiveness of our targeted synthetic data augmentation strategy, which

Method	PD	AD	Δ^*	CUB-200-2011 Data				Cars196 Data			
				R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
DINO$_H$	Φ	-	-	64.2	75.6	84.4	91.0	70.3	79.2	86.8	91.7
NA-DINO$_H$	Φ	$\star$	∞	*65.1*	*76.3*	*84.3*	*90.7*	*67.1*	*77.4*	*85.2*	*91.3*
A-DINO$_H$	Φ	Ψ,Ω	1	**68.4**	**78.4**	**86.3**	**91.9**	**75.4**	**84.7**	**91.0**	**94.4**
DINO$_H$	Ψ	-	1	69.2	78.1	85.5	90.6	66.9	76.1	83.9	89.8
NA-DINO$_H$	Ψ	$\star$	∞	*69.6*	*79.4*	*87.1*	*92.5*	*67.8*	*78.0*	*85.3*	*90.8*
A-DINO$_H$	Ψ	Ω,Φ	1	**72.8**	**81.6**	**88.3**	**93.4**	**75.8**	**84.2**	**90.4**	**94.5**
DINO$_H$	Ω	-	1	63.1	74.3	83.0	89.9	56.9	68.6	78.2	85.6
NA-DINO$_H$	Ω	$\star$	∞	*66.2*	*76.6*	*84.9*	*91.3*	*66.4*	*74.9*	*82.3*	*88.2*
A-DINO$_H$	Ω	Φ,Ψ	1	**66.9**	**77.6**	**85.8**	**91.8**	**75.2**	**84.5**	**90.9**	**94.5**

Table 1: Comparison of results in the baseline (DINO$_H$), naive-untargeted synthetic augmentation (**NA**-DINO$_H$), and ATLANTIS augmentation (**A**-DINO$_H$) settings under *domain scarcity*. PD: domains in original training data, AD: super-set of augmented domains based on class-specific characteristics, Δ^*: filtering hyperparameter for synthetic data before augmentation (with ∞ indicating no ORDC filtering). Φ, Ψ, Ω: three activity-based domains (`'Flying'`, `'Sitting'`, `'Swimming'`) in the CUB-200-2011 [8]; three vehicle body-type based domains (`'Sedan'`, `'SUV-Crossover'`, `'Performance Sport or Convertible'`) in the Cars196 data [4].

identifies and addresses weaknesses in the training data using *in-series* components: DIG, APS, and ORDC, that naive augmentation methods overlook.

E.3 Efficiency Gains Against Conventional Augmentation Techniques

In this subsection, we compare the efficiency gains of our ATLANTIS framework against the conventional naive-untargeted augmentation approaches.

The naive-untargeted augmentation methods introduces synthetic data without specific objectives, thereby adding substantial noise and redundancy to the training dataset. Let $\mathcal{X}_n$ represent the naive synthetic dataset, which is generated by augmenting the original dataset $\mathcal{X}_o$ with α synthetic instances per real instance. The total size of the augmented dataset $\mathcal{X}_n$ can be expressed as:

$$|\mathcal{X}_n| = (1+\alpha)|\mathcal{X}_o| \tag{3}$$

Given that naive augmentation does not target specific weaknesses, it requires a large number of synthetic generations to achieve comprehensive coverage of the data space. Assuming p is the probability of generating a useful instance that addresses a specific weakness, the expected number of synthetic instances $E[\mathcal{X}_{useful}]$ needed to effectively cover the weaknesses is given by:

$$E[\mathcal{X}_{useful}] = \frac{1}{p} \tag{4}$$

In contrast, our ATLANTIS framework systematically identifies and targets weaknesses using the DIG and APS. By directly addressing these weaknesses, ATLANTIS avoids unnecessary data generation, thus significantly reducing the required synthetic data. Let $\mathcal{X}_t$ represent the targeted synthetic dataset. The total size of the targeted augmented dataset $\mathcal{X}_t$ can be expressed as:

Method	κ	ZSL	Δ^*	CUB-200-2011 Data				Cars196 Data			
				R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
DINO$_H$	50	✓	-	71.0	81.5	88.5	93.6	75.0	83.5	90.0	94.4
NA-DINO$_H$	50	✓	∞	*74.9*	*83.9*	*90.1*	*94.5*	*76.5*	*85.3*	*90.8*	*94.5*
A-DINO$_H$	50	✓	1	**76.5**	**84.8**	**90.5**	**94.7**	**82.6**	**89.2**	**93.8**	**96.5**
DINO$_H$	75	✓	-	64.3	75.7	84.7	90.9	61.8	73.0	81.6	88.4
NA-DINO$_H$	75	✓	∞	*73.3*	*82.7*	*89.4*	*94.0*	*71.6*	*81.0*	*87.7*	*92.7*
A-DINO$_H$	75	✓	1	**76.0**	**84.3**	**90.3**	**94.1**	**80.6**	**88.1**	**92.9**	**95.7**

Table 2: Comparison of results in the baseline Hyperbolic-DINO model [1] (DINO$_H$), naive-untargeted synthetic augmentation (**NA**-DINO$_H$), and ATLANTIS augmentation (**A**-DINO$_H$) settings under **class-imbalance scenario**. κ: number of classes in the original data with samples restricted to two (by following the same methodology as in the paper), ZSL: zero-shot learning setting, Δ^*: filtering hyperparameter for synthetic data before augmentation (with ∞ indicating no ORDC filtering).

$$|\mathcal{X}_t| = (1+\beta)|\mathcal{X}_o| \tag{5}$$

where $\beta \ll \alpha$ due to the targeted nature of the augmentation. The efficiency gain η of our framework compared to the naive approach can be quantified by the ratio of the required synthetic instances:

$$\eta = \frac{E[\mathcal{X}_n]}{E[\mathcal{X}_t]} \tag{6}$$

Given that the targeted framework directly addresses weaknesses with a high probability q (where $q \approx 1$), the expected number of synthetic instances $E[\mathcal{X}_{targeted}]$ required is significantly lower:

$$E[\mathcal{X}_{targeted}] = \frac{1}{q} \approx 1 \tag{7}$$

Substituting these values, the efficiency gain η can be expressed as:

$$\eta = \frac{1/p}{1/q} = \frac{q}{p} \tag{8}$$

As $q \approx 1$ and $p \ll 1$ in naive augmentation, η becomes significantly greater than 1, demonstrating substantial efficiency gains. This efficiency is reflected in both reduced computational costs and improved model performance, as ATLANTIS generates only the necessary synthetic data, directly aligned with the augmentation objectives, thereby enhancing the robustness and generalisation of the CBIR models.

To illustrate, if $p = 0.1$ (naive method's probability of useful instance generation) and $q = 0.9$ (our framework's targeted generation probability), then:

$$\eta = \frac{0.9}{0.1} = 9 \tag{9}$$

This indicates that our ATLANTIS framework is 9 times more efficient in generating useful synthetic data compared to the naive-untargeted approach, highlighting the significant performance and efficiency benefits of our targeted augmentation strategy.

Algorithm 2 Training Feedback Generation

```
Require: Set of trained candidate DML models $\mathcal{M}$, evaluation function Eval, threshold $\tau$,
    weak-class hyper-parameter $k$.
Ensure: Best model $M^*$, Set of underperforming classes $\mathcal{C}_{\text{poor}}$
1: $M^* \leftarrow \arg\max_{M_i \in \mathcal{M}} \texttt{Eval}(M_i)$
2: Initialise $\mathcal{C}_{\text{poor}}$ to an empty set
3: for $i \leftarrow 1$ to $k$ do
4:     $c_{\text{poor}} \leftarrow \arg\min_{c \in \mathcal{C}} \texttt{Eval}(M^*, c)$
5:     if $\texttt{Eval}(M^*, c_{\text{poor}}) < \tau$ then
6:         $\mathcal{C}_{\text{poor}} \leftarrow \mathcal{C}_{\text{poor}} \cup \{c_{\text{poor}}\}$
7: Send $T_{\text{feedback}}$ containing $\mathcal{C}_{\text{poor}}$ to APS for the targeted additional data generation.
```

F Training Feedback Component

The Training Feedback module $\Theta_{tf} : (\mathcal{X}_{train}, M^*, \texttt{Eval}) \rightarrow T_{\text{feedback}}$, as described in algorithm 2, evaluates class-wise R@1 scores using $\mathcal{M}^*$ on the training set to identify a predefined number of training classes with the worst performance. In cases where multiple models $\{\mathcal{M}_0, \mathcal{M}_1, \ldots\} \in \mathcal{M}$ are being trained for data cleaned with a set of Δ values, it first identifies the best-performing model before initiating the feedback generation process. These insights are crucial for the APS to recalibrate the synthetic data generation process, targeting classes that require further augmentation.

F.1 Effectiveness of the Training Feedback Component

In most scenarios, we found that while fine-tuning CBIR models, there was no significant requirement for the training feedback loop due to the substantial pre-training of current ViT models used for fine-tuning. However, in cases where a model is trained from scratch, it becomes crucial to focus on training classes with low performance. In such instances, our training feedback loop provides valuable information to the APS to generate additional data and enhance data diversity for these struggling training classes.

Although the performance gains achieved through this training feedback loop during the fine-tuning phase were marginal in the experiments prompting us to transfer the results here, we still verified its effectiveness. Table 3 presents results for the case where the training feedback loop was used alongside other components of the ATLANTIS framework. Even with the inclusion of the training feedback loop, our framework outperformed the state-of-the-art baseline for the DINO model marginally.

For this ablation, we restricted the generations to only the classes identified by the training feedback loop, effectively isolating the impact of this component. Clearly, from the results, we can see that the feedback loop mechanism can be beneficial in further improving performance by directing the synthetic data generation process to specifically address classes with low training performance.

G Results for Filtering with Different Δ Values

Table 4 presents the outcomes of our analysis on the impact of varying Δ values in ORDC filtering across the synthetic CUB-200-2011 [8] and Cars196 [4] datasets. The datasets were with *standard balanced class and domain distributions*, hence positioning this as the most challenging scenario for ATLANTIS as the scope for improvement through synthetic

Method	$T_{feedback}$	R@1	R@2	R@4	R@8
DINO$_H$	✗	78.3	86.0	91.2	94.7
DINO$_H$	✓	**78.8**	**86.7**	**92.2**	**95.4**

Table 3: A comparison of ZSL results on the CUB-200-2011 data [8] for the DINO$_H$ [1] model with and without the training feedback loop component. $T_{feedback}$ indicates the presence (✓) or absence (✗) of the training feedback loop. The results show marginal improvements in recall scores when the training feedback loop is used, demonstrating its effectiveness in enhancing CBIR model performance by focusing on low-performing training classes. Best R@k results are highlighted in bold.

Data	Method	Δ	DINO$_H$				ViT$_H$			
			R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
$\mathcal{X}_{cub}$	Baseline	-	78.3	86.0	91.2	94.7	84.0	90.2	94.2	96.4
	Ours	∞	77.7	85.5	90.5	94.9	82.9	89.8	94.0	96.4
	Ours	3	78.0	85.7	90.7	94.9	82.9	89.8	94.0	96.2
	Ours	1.5	78.2	86.0	91.5	94.9	83.4	90.1	94.1	96.4
	Ours	1	**79.1**	**87.1**	**92.2**	**95.5**	**84.1**	**90.3**	**96.5**	**97.9**
	Ours	0.9	78.2	86.6	91.7	95.1	83.8	90.1	96.2	97.6
$\mathcal{X}_{cars}$	Baseline	-	86.0	91.9	95.2	97.2	82.7	89.7	93.9	96.2
	Ours	∞	85.2	91.1	94.8	96.8	82.6	89.3	93.0	95.8
	Ours	5	86.1	91.7	95.1	97.3	82.8	89.5	93.5	96.6
	Ours	1.5	**86.8**	**92.4**	**95.5**	**97.3**	**83.4**	**90.0**	**94.0**	**96.5**
	Ours	1.2	86.2	91.8	95.1	97.3	82.9	89.6	93.5	96.5
	Ours	1	85.7	91.5	95.1	97.2	82.7	89.4	93.1	95.9

Table 4: Results for varying Δ in ORDC filtering on the augmented data with the *standard balanced* benchmark CUB-200-2011 [8] and Cars196 [4] datasets, showing optimal diversity at $\Delta = 1$ for CUB-200-2011 and $\Delta = 1.5$ for Cars196 with ATLANTIS, DINO$_H$, and ViT$_H$ architectures.

data augmentation in the available training set was minimal. Optimal diversity and removal of outliers were achieved at $\Delta = 1$ for CUB-200-2011 and $\Delta = 1.5$ for Cars196, utilising ATLANTIS, DINO$_H$, and ViT$_H$ architectures. This underscores the strategic efficacy of ATLANTIS in enhancing dataset quality thus training, even under stringent conditions that limit the conventional techniques for the image retrieval performance gains.

Notably, our framework outperforms the state-of-the-art baseline [1] across various Δ values, especially showing superior performance at higher K values of R@K scores. This indicates that while ATLANTIS boosts recall, it slightly reduces precision if the noise is not strictly filtered in the generated data. Furthermore, too small Δ values introduce data imbalance due to stronger filtering constraints, hence lowering performance. However, the Filtering Feedback aims to counterbalance such effects resulted by filtering. Despite these measures, some imbalance may persist because of predefined stopping criteria, marginally affecting overall effectiveness. This detailed examination highlights the importance of careful parameter tuning to maximise performance, demonstrating our framework's adeptness at navigating complex data characteristics.

H Adversarial Robustness Gains Across Datasets and Attack Strengths

We evaluate ATLANTIS's adversarial robustness on CUB-200-2011 and Cars196 datasets against embedding-space white-box PGD attacks [5]. We measure performance across different attack gradients steps ($s \in 1, 10$) and noise sizes ($\varepsilon \in 0.05, 0.1, 0.5$). Table 5 expands on this by including results for larger noise sizes not discussed in the main paper, highlighting ATLANTIS's effectiveness mainly against subtle noise attacks. Comparisons between original DINO and **RR**-DINO models in ZSL and FSL settings reveal ATLANTIS's robustness improvements in standard imperceptible noise attack scenarios but show variability with larger noise sizes. This suggests a stronger defense against imperceptible noise, pointing to potential areas for future improvement.

Data	Training Mode	Steps	ε	R@1 Baseline	R@1 Gain
$\mathcal{X}_o^{cub}$	ZSL	1	0.05	62.90	1.27
$\mathcal{X}_o^{cub}$	ZSL	1	0.10	54.30	0.37
$\mathcal{X}_o^{cub}$	ZSL	1	0.50	28.60	-5.24
$\mathcal{X}_o^{cub}$	ZSL	10	0.05	63.40	0.79
$\mathcal{X}_o^{cub}$	ZSL	10	0.10	56.30	1.60
$\mathcal{X}_o^{cub}$	ZSL	10	0.50	31.60	2.85
$\mathcal{X}_s^{cub}$	ZSL	1	0.05	66.30	-0.45
$\mathcal{X}_s^{cub}$	ZSL	1	0.10	58.00	0.17
$\mathcal{X}_s^{cub}$	ZSL	1	0.50	36.00	-4.72
$\mathcal{X}_s^{cub}$	ZSL	10	0.05	66.50	1.35
$\mathcal{X}_s^{cub}$	ZSL	10	0.10	60.90	-1.31
$\mathcal{X}_s^{cub}$	ZSL	10	0.50	37.90	1.06
$\mathcal{X}_o^{cub}$	FSL	1	0.05	61.60	3.73
$\mathcal{X}_o^{cub}$	FSL	1	0.10	52.60	1.71
$\mathcal{X}_o^{cub}$	FSL	1	0.50	26.90	-6.69
$\mathcal{X}_o^{cub}$	FSL	10	0.05	63.60	1.73
$\mathcal{X}_o^{cub}$	FSL	10	0.10	55.20	3.80
$\mathcal{X}_o^{cub}$	FSL	10	0.50	28.90	4.50
$\mathcal{X}_s^{cub}$	FSL	1	0.05	79.50	0.63
$\mathcal{X}_s^{cub}$	FSL	1	0.10	73.70	0.68
$\mathcal{X}_s^{cub}$	FSL	1	0.50	54.70	-2.38
$\mathcal{X}_s^{cub}$	FSL	10	0.05	79.30	1.77
$\mathcal{X}_s^{cub}$	FSL	10	0.10	74.90	1.34
$\mathcal{X}_s^{cub}$	FSL	10	0.50	57.00	-1.58
$\mathcal{X}_o^{cars}$	ZSL	1	0.05	67.67	0.40
$\mathcal{X}_o^{cars}$	ZSL	1	0.10	55.21	-0.83
$\mathcal{X}_o^{cars}$	ZSL	1	0.50	29.87	-2.41
$\mathcal{X}_o^{cars}$	ZSL	10	0.05	68.85	-1.76
$\mathcal{X}_o^{cars}$	ZSL	10	0.10	59.70	2.24
$\mathcal{X}_o^{cars}$	ZSL	10	0.50	33.24	-4.27
$\mathcal{X}_s^{cars}$	ZSL	1	0.05	56.98	8.88

Continued on next page

Table 5 continued from previous page

Data	Training Mode	Steps	ε	R@1 Baseline	R@1 Gain
$\mathcal{X}_s^{cars}$	ZSL	1	0.10	46.99	7.38
$\mathcal{X}_s^{cars}$	ZSL	1	0.50	17.60	14.43
$\mathcal{X}_s^{cars}$	ZSL	10	0.05	56.61	9.43
$\mathcal{X}_s^{cars}$	ZSL	10	0.10	47.87	12.03
$\mathcal{X}_s^{cars}$	ZSL	10	0.50	26.58	6.02
$\mathcal{X}_o^{cars}$	FSL	1	0.05	60.50	0.99
$\mathcal{X}_o^{cars}$	FSL	1	0.10	47.29	1.54
$\mathcal{X}_o^{cars}$	FSL	1	0.50	18.77	2.56
$\mathcal{X}_o^{cars}$	FSL	10	0.05	61.91	0.11
$\mathcal{X}_o^{cars}$	FSL	10	0.10	51.85	2.91
$\mathcal{X}_o^{cars}$	FSL	10	0.50	22.92	-4.28
$\mathcal{X}_s^{cars}$	FSL	1	0.05	54.89	15.30
$\mathcal{X}_s^{cars}$	FSL	1	0.10	44.45	17.14
$\mathcal{X}_s^{cars}$	FSL	1	0.50	13.40	29.40
$\mathcal{X}_s^{cars}$	FSL	10	0.05	54.37	16.20
$\mathcal{X}_s^{cars}$	FSL	10	0.10	45.58	18.85
$\mathcal{X}_s^{cars}$	FSL	10	0.50	23.50	17.62

Table 5: Adversarial Robustness Gains Across Datasets and Various Attack Strengths: This table extends the results covering the original and synthetic test sets of the CUB-200-2011 ($\mathcal{X}_o^{cub}$, $\mathcal{X}_s^{cub}$) and Cars196 ($\mathcal{X}_o^{cars}$, $\mathcal{X}_s^{cars}$) datasets to include the impact of our framework against white-box embedding-space PGD attacks of varying intensities on adversarial R@1. Robustness across different numbers of attack gradient steps ($s \in \{1, 10\}$) and adversarial noise size bounds ($\varepsilon \in \{0.05, 0.1, 0.5\}$), expressed as a fraction of the input image pixel value range, is measured. Additionally, the table explores ATLANTIS's adversarial robustness against larger adversarial noise sizes ε not detailed in the paper, revealing inconsistent robustness gains for such attack strengths. This underscores ATLANTIS's efficacy primarily against imperceptible noise attacks. Results are presented for both ZSL and FSL learning settings, comparing the original DINO and our **RR**-DINO models, with further insights into the framework's behaviour under more extreme noise conditions not covered in the main paper's figures.

I Results for Novel Class Augmentation

Table 6 presents a comparison of ZSL results when new classes introduced through synthetic data augmentation by ATLANTIS had overlap with the original CUB-200-2011 training classes versus the case where completely novel classes ($\mathcal{N}_{novel}$) were introduced. These novel classes are neither part of the training nor test classes of the original CUB-200-2011 data. For a fair evaluation, we ensured that the new augmented classes (AC) did not include any test classes from the CUB-200-2011 data. $\mathcal{C}_{train}$ represents the classes in the available original training data, $\mathcal{N}_{train}$ represents novel classes introduced by ATLANTIS that are within the scope of the original CUB-200-2011 training data, and $\mathcal{N}_{novel}$ represents

Case	$\mathcal{C}_{train}$	AC		A-DINO$_H$				A-ViT$_H$			
		$\mathcal{N}_{train}$	$\mathcal{N}_{novel}$	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
A	25	75	-	73.14	82.39	89.30	93.55	83.17	90.24	93.77	96.30
A^*	25	75	10	**73.40**	**82.53**	89.30	**93.70**	**83.36**	**90.48**	**93.82**	96.24
A	25	175	-	77.48	85.99	91.27	94.62	83.29	90.07	94.04	96.40
A^*	25	175	10	**77.58**	85.82	**91.29**	94.58	**83.46**	89.99	93.92	**96.44**

Table 6: A comparison of ZSL results when new classes introduced through synthetic data augmentation by ATLANTIS had overlap with the original CUB-200-2011 training classes (indicating APS may have used prior knowledge of public data) versus the case where completely novel classes ($\mathcal{N}_{novel}$) were introduced, which are neither part of the training nor test classes of the original CUB-200-2011 data. For a fair evaluation, we ensured that the new augmented classes (AC) did not include any test classes from the CUB-200-2011 data. $\mathcal{C}_{train}$ represents the classes in the available original training data, $\mathcal{N}_{train}$ represents novel classes introduced by ATLANTIS that are not part of the available training data but are within the scope of the original CUB-200-2011 training data, and $\mathcal{N}_{novel}$ represents classes that were neither part of the available training data ($\mathcal{N}_{train}$) nor part of the original CUB-200-2011 training or test classes. The 10 novel classes introduced are African Grey Parrot, Atlantic Puffin, Bald Eagle, Emperor Penguin, Golden Eagle, Harpy Eagle, Peacock, Peregrine Falcon, Scarlet Macaw, and Snowy Owl, which were non-existent in the original CUB-200-2011 data. Case A represents training using ATLANTIS when the new augmented classes overlapped with the original training class set, and case A^* represents training with ATLANTIS where, in addition to $\mathcal{N}_{train}$, completely novel classes $\mathcal{N}_{novel}$ were also augmented. Best R@k results are highlighted in bold.

classes that were neither part of the available training data ($\mathcal{N}_{train}$) nor part of the original CUB-200-2011 training or test classes. The 10 novel classes introduced are African Grey Parrot, Atlantic Puffin, Bald Eagle, Emperor Penguin, Golden Eagle, Harpy Eagle, Peacock, Peregrine Falcon, Scarlet Macaw, and Snowy Owl, which were non-existent in the original CUB-200-2011 data.

The results in Table 6 illustrate the versatility of our framework, demonstrating its ability to introduce completely novel classes that are not even part of the original test or training data. This showcases the potential of ATLANTIS in scenarios where the available data has very few class identities, allowing it to significantly expand the diversity of class identities. This expansion benefits the user data by providing a richer and more varied dataset, ultimately leading to better-trained models.

In cases where our framework introduced novel classes, we observed slight improvements in performance metrics. For instance, introducing just 10 completely novel classes ($\mathcal{N}_{novel}$) resulted in modest gains in recall metrics across both **A**-DINO$_H$ and **A**-ViT$_H$ models. Specifically, in the case A^* where 10 novel classes were added to the 75 augmented classes, R@1 for **A**-DINO$_H$ improved from 73.14 to 73.40, and for **A**-ViT$_H$ from 83.17 to 83.36. Similarly, in the case of expanding from 175 augmented classes, adding 10 novel classes resulted in R@1 improvements from 77.48 to 77.58 for **A**-DINO$_H$ and from 83.29 to 83.46 for **A**-ViT$_H$.

These results highlight the capability of ATLANTIS to generate and incorporate novel classes, enhancing the robustness and generalisation of models trained on such enriched datasets. The framework's ability to systematically identify and augment novel class identities proves valuable in improving the overall performance of CBIR models, particularly in datasets with limited class diversity.

J Processing Speed and Computational Complexity of ATLANTIS

J.1 Processing Speed of ATLANTIS

The processing speed of the ATLANTIS framework can vary based on the complexity of the data, the specific models used, and the computational resources available. However, ATLANTIS is designed to optimize the augmentation process by leveraging LLMs and pre-trained models for efficiency. Below is a detailed estimate of the processing speed and computational complexity of generating synthetic data within ATLANTIS.

J.2 Detailed Estimate of Synthetic Data Generation Time

The framework operates with linear computational and memory complexity across its components. The approximate equation estimating the time required to generate synthetic data (excluding the final training part) is as follows:

$$T_{total}(N_o, N_s) = \gamma + (k_f + k_i) \cdot N_o + k_a + (k_d + k_s + k_o) \cdot N_s$$

where:

- N_o represents the number of original images.
- N_s represents the number of synthetic images to be generated.
- γ is the VRAM load time for all models and components of ATLANTIS, assuming model weights are already downloaded locally.
- k_f is the time required to generate a text description for one original image.
- k_i is the time required to process the text description of one original image by the DIG.
- k_a is the time required to process insights and feedbacks by the APS.
- k_d is the time required to generate a text description for one target synthetic image by an LLM while executing APS-defined objectives.
- k_s is the time required to generate one target synthetic image from a given text description.
- k_o is the time required to process a single synthetic image by the ORDC.

J.3 Parameter Values and Example Calculation

In the present setting (using an A100 80GB GPU, standard Azure SSD, and server-grade Intel Xeon CPU), the parameter values are approximately:

- $\gamma = 60$ seconds
- $k_f = 2$ seconds

- $k_i = 0.1$ seconds
- $k_a = 5$ seconds
- $k_d = 3$ seconds
- $k_s = 30$ seconds (for SDXL 32bit, 8 seconds for SDXL Turbo)
- $k_o = 2$ seconds

Using these values, the total time required to generate synthetic data for 1000 original images and 2000 synthetic images is:

$$T_{total}(1000, 2000) = 65 + 2.1 \cdot 1000 + 35 \cdot 2000$$

$$T_{total}(1000, 2000) = 65 + 2100 + 70000$$

$$T_{total}(1000, 2000) = 72165 \text{ seconds}$$

This translates to approximately 20 hours.

J.4 Training Time

For the training part, our experiments show that the convergence time for ATLANTIS scales lower than linear complexity. For example, when using the CUB200 dataset, ATLANTIS required 30-40 more iterations than the base 100 epochs to reach convergence. The exact values may vary depending on the specific case.

By considering these computational requirements and processing times, we can better plan and optimize their use of the ATLANTIS framework for synthetic data generation and model training.

References

[1] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7409–7419, 2022.

[2] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33 (1):117–128, 2010.

[3] Huajie Jiang, Ruiping Wang, Shiguang Shan, and Xilin Chen. Adaptive metric learning for zero-shot recognition. *IEEE Signal Processing Letters*, 26(9):1270–1274, 2019.

[4] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[6] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

[7] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[8] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[9] Xinyi Xu, Huanhuan Cao, Yanhua Yang, Erkun Yang, and Cheng Deng. Zero-shot metric learning. In *IJCAI*, pages 3996–4002, 2019.