Supplementary Material:
Image-free Classifier Injection for Zero-Shot Classification

In this supplementary material, we firstly in Sec. A provide additional implementation details regarding the baselines and our stopping criterion. In Sec. B, we combine applicable baselines with ICIS and report the combined performance. We report additional results for the I-GZSL task for an ImageNet pretrained classification model in Sec. C. In Sec. D, we show that the architectural additions of ICIS deal with bias towards seen classes, a common issue of GZSL, despite having no access to images. We finally provide an extended analysis of a failure case of our model in the generalized zero-shot task on CUB in Sec. E.

A. Additional implementation details

Implementation and adaptation of baselines. In this section, we detail how the baselines used for comparison with ICIS have been implemented (and potentially adapted) for the image-free ZSL task.

ConSE: We follow the original implementation of [39], using the classifier scores to perform a convex combination of the class label embeddings.

COSTA: In [36], different co-occurrence similarities are proposed. In our experiments, normalised co-occurrence similarity led to the best results.

*Sub. Reg.**: We adapt the subspace regularisers proposed in [2] originally acting on model parameters, to instead serve as an additional loss on the predicted classifiers during training. Concretely, we apply a projection matrix P_S on the predicted classifiers, and apply a loss based on the distance d between the predicted and projected weights:

$$\mathcal{L}_{Sub.Reg.} = \sum_{u \in U} d(\mathbf{w}_u, P_S^T \mathbf{w}_u). \tag{8}$$

The projection matrix P_S is calculated from the existing classifiers of the seen classes. It projects a predicted weight $\mathbf{w}_u$ onto the subspace spanned by the existing seen class classifiers. We found that using the squared error, as originally done in [2], worked the best.

*wDAE**: In order to adapt the setup from [16] to the I-ZSL setting, two primary changes were made. Firstly, since image features are not available, classifier estimates originally created by feature averaging are not accessible. Therefore, we train a weight predictor to provide the initial estimates. In the main paper, we use the MLP base model as the initial predictor, but here in the supplementary material we also report results when used in conjunction with ICIS (Table 4). Secondly, for the proposed denoising autoencoder (DAE) to be applicable in our setting, we adapt the loss function for the model. Since we do not have access to images to test downstram classification accuracy, we remove the image-related term of Eq. (6). in [16], resulting in the loss function

$$\mathcal{L}_{wDAE} = \sum_{u \in U} d(\mathbf{w}_u, \mathbf{w}_u^*). \tag{9}$$

Here, $\mathbf{w}_u$ are the initially estimated weights, and $\mathbf{w}_u^*$ are the reconstructed weights by the DAE, with d as mean squared distance leading to the best downstream results.

VGSE (WAvg and SMO*)*: We adapt the weighting schemes of [61] to directly produce new classifiers. For WAvg, we use the same hyperparameters as presented in the paper. For SMO, we experimented with both $\alpha = 0$ and $\alpha = -1$. However, $\alpha = 0$ consistenly led to better downstream results, and all reported results are computed with this value.

Stopping criterion. We take inspiration from [34], where an early stopping criterion based on statistical properties of the gradients was proposed. Here, we base our simple stopping criterion on the slope of the training loss. Concretely, we compute a running mean of the training loss over the latest 10 epochs, as well as the 10 previous epochs. We then compute the (negative) slope between these averages, and compare it to a threshold value which is set to $2 \cdot 10^{-4}$. This value was determined from the simple empirical observations that higher values were too high (*i.e.* training was terminated almost immediately across different architectures), while lower values were too low (*i.e.* the slope rarely went below the threshold, such that the training loop continued indefinitely, despite the network already having converged). When applying the mean squared error as the loss function (instead of the cosine loss), the losses are smaller by a factor of approximately 10^{-3} and the threshold value is therefore then scaled by this factor.

B. Improving baselines with ICIS

As the model components that ultimately make up ICIS are orthogonal to Sub. Reg.* and wDAE*, we include results using these baselines in combination with our ICIS framework in Table 4. For the results reported under Sub. Reg.* and wDAE*, the initial weight estimator network is an MLP network with a similar structure as the descriptor-to-weights mapping of ICIS (*i.e.* the MLP base model performance from our ablation study). For this initial weight predictor, Sub. Reg.* and wDAE* do not improve results, compared to the results of the MLP base model by itself. Indeed, the zero-shot accuracies generally drop (*e.g.* 41.3% to 37.6% on CUB for Sub. Reg.*, as can be seen in Table 1 and Table 2 of the paper), and the harmonic mean does not increase.

However, if we apply ICIS to wDAE* and Sub. Reg* together, we see consistent performance gains across all datasets and setting. For instance, the zero-shot accuracy

Method	+ ICIS	Zero-Shot Accuracy			Generalised Zero-Shot Accuracy								
		CUB	AWA2	SUN	CUB			AWA2			SUN		
		Acc	Acc	Acc	u	s	H	u	s	H	u	s	H
Sub. Reg.* [2]	✗	37.6	37.5	48.3	0.0	87.6	0.0	0.0	96.1	0.0	0.0	50.1	0.0
	✓	**60.5**	**65.8**	**51.8**	45.8	73.6	**56.5**	35.7	93.3	**51.6**	45.6	25.1	**32.3**
wDAE* [16]	✗	38.2	37.0	49.9	0.0	87.3	0.0	0.1	96.0	0.3	0.0	49.3	0.0
	✓	**60.4**	**65.1**	**51.9**	46.1	73.1	**56.6**	36.0	93.3	**52.0**	45.1	25.5	**32.6**

Table 4: Improving downstream classification performance on ZSL benchmarks in the I-ZSL setting when combining existing methods in the literature with our proposed ICIS framework. We measure the results as unseen accuracy (Acc) for the zero-shot task, unseen (u) and seen (s) accuracy and their harmonic mean (H) for the generalised zero-shot setting. Methods marked with * are adapted to the image-free setting.

increases of more than 20% on both AWA2 and CUB for both methods, while on SUN the performance gain is at least of 2%. The gap is even clearer in the generalised setting, where Sub. Reg.* and wDAE* are able to correctly recognise unseen classes only when coupled with ICIS, going from 0 harmonic mean to over 50 for CUB and AWA2, while of more than 32 for SUN. These results provide further evidence of how ICIS reduces the bias on seen classes in image-free generalised zero-shot learning. Furthermore, they demonstrate the flexibility of ICIS, being a simple approach that can benefit the performance of other methods based on classifier weights prediction.

C. Additional results

Here, we show results demonstrating the performance of ICIS for different pre-training setups of the backbone, and for the I-GZSL task on CUB for an ImageNet pretrained model with injected CUB classifiers.

Firstly, we analyse the impact of varying the ResNet-101 backbone. In Table 1, the backbone is fine-tuned on the seen classes of the ZSL datasets, as commonly done in the literature [58]. Alternatively, the classification head can be trained while keeping the feature extractor frozen. We report results for this option in the left column of Table 5 on CUB. We observe that also when using an ImageNet-pretrained feature extractor, ICIS still perform significantly better than competing methods in terms of both zero-shot accuracy (Acc) and on the generalized I-ZSL task in terms of resulting harmonic mean (H). Overall, the trend of the results do not change between using a pre-trained and fine-tuned feature extractor. Curiously, the performance of some methods increase slightly when using the ImageNet-pretrained classifier (e.g. wDAE* [16] with Acc and H increasing from 38.2% and 0.0 to 40.0% and 4.1, respectively.

Secondly, we report I-GZSL results corresponding to Table 3, i.e. the generalized zero-shot classification task for an ImageNet pretrained model for which we inject classifiers for classification on CUB in *addition* to classifying

Model	CUB to CUB		ImageNet to CUB		
	Acc	H	s	u	H
ConSE [39]	35.9	0.9	77.0	0.0	0.0
COSTA [36]	21.8	0.0	75.2	6.7	12.3
Sub.Reg.* [2]	40.3	4.7	77.4	0.0	0.0
wDAE* [16]	40.0	4.1	77.3	7.4	13.5
WAvg* [61]	2.0	1.8	77.4	0.0	0.0
SMO* [61]	39.3	32.2	70.0	2.5	4.9
ICIS (Ours)	**50.8**	**41.2**	77.3	11.6	**20.2**

Table 5: Left: CUB I-ZSL (Acc) and I-GZSL (H) results using a frozen ResNet feature extractor pre-trained on ImageNet. Right: I-GZSL results from ImageNet to CUB using CLIP class label text embeddings. Methods marked with * are adapted to I-ZSL.

ImageNet. The results are reported in the right column of Table 5. We can see that the results from learning to inject classifiers from a classifier trained on a semantically unrelated dataset (in this case ImageNet) leads to lower results than when learning from a semantically related one (e.g. CUB seen classes). However, ICIS is still able to infer and inject classifiers for the generalized task that lead to non-trivial performance on this difficult task (unseen accuracy 11.6%) while maintaining an accuracy of 77.3% on the seen classes (with the pre-trained ResNet model achieving a 77.4% seen class accuracy before injecting additional classifiers).

D. Dealing with bias towards seen classes

For the generalised zero-shot classification problem, bias towards seen classes is one of the primary challenges to overcome. In the GZSL literature, a common techniques to address this obstacle is to use seen and unseen class experts [9], or to introduce hyperparameters that reduce the confidence for seen classes [5].

In the image-free I-ZSL setting, these options are not applicable, since there is no access to samples on which those options can be tuned. Instead, we show in the analysis

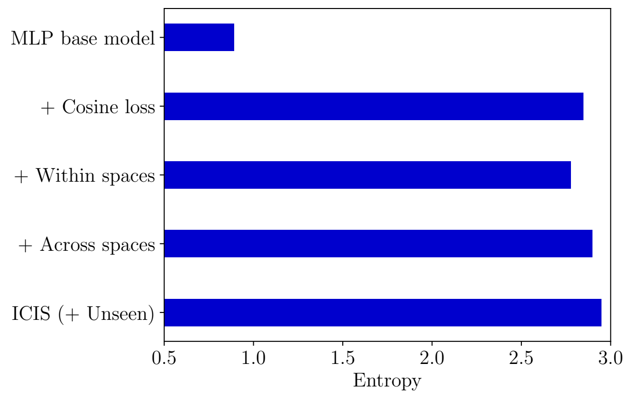

Figure 5: Inference analysis of our model ablations on CUB. We plot the mean entropy (calculated using $\log_e$) of the output distributions over the test samples from both seen and unseen classes. The cosine loss in particular has a large impact on the injected weights dealing with bias towards seen classes.

in Fig. 5 that each element of ICIS increases the average entropy of the output distribution over the seen and unseen class test samples in CUB, *i.e.* it reduces overconfidence towards seen classes.

This analysis confirms the findings of the previous one in Table 4, as the baselines (without ICIS) almost exclusively predict seen classes in the I-GZSL setting. However, the introduction of our framework's components increase the accuracy for unseen classes for the baselines notably, while having only a small impact on the seen class accuracy.

E. Study of failure case

In this section, we study some shortcomings of ICIS on a concrete example for the CUB dataset. Concretely, we consider the behaviour of the injected weights on the class for which they achieve the worst classification accuracy in the downstream I-GZSL task.

Failures of injected weights in the generalised task. For the unseen class *Tree Sparrow*, our ICIS injects classification weights which correctly classify only 5% of the corresponding samples for the generalised zero-shot classification task. In Figure 6, we plot the predictions of our model for images of this class.

For plotting the predictions, we count which classes were predicted for the *Tree Sparrow* samples, and then divide these counts by the total number of *Tree Sparrow* samples to obtain class-wise empirical (mis)classification probabilities. To get an understanding of the properties of the classes in the dataset that the *Tree Sparrow* samples were misclassified as, we rank all 200 classes in the CUB dataset based on the cosine similarity between the attributes of *Tree Sparrow*, and the attributes of each class in the dataset. For visualisation, we bin the classes in groups of 10, starting from the most similar (*i.e. Tree Sparrow* itself) to the least similar class in the dataset, summing up their empirical prediction probabilities.

Figure 6 shows how the predicted classification weights result in classifying primarily the most similar classes to *Tree Sparrow*. This suggests that the predicted weights indeed classify based on properties close to those of *Tree Sparrow*.

Figure 7 zooms in on just the classes that the *Tree Sparrow* samples were misclassified as, revealing that the predicted weights have a bias towards other unseen classes. Most misclassifications are due to predicting other unseen and similar classes, such as *White-crowned Sparrow* and *Field Sparrow*. Similarly, misclassifications among seen classes only occur on the two most similar classes to *Tree Sparrow*, namely *Chipping Sparrow* and *House Sparrow*.

For reference, we show images of birds from the listed classes in Figure 8. We include images from the two seen classes confused with Tree Sparrow (i.e. *Chipping Sparrow* and *House Sparrow*) as well as images from the two unseen classes with the largest fraction of predictions (i.e. *White-crowned Sparrow* and *Field Sparrow*). Although there are distinctive differences between classes (e.g. the black and white head of the *White-crowned Sparrow*, or the beak shape and size of the *House Sparrow*), it is clear that the predicted weights are capturing fundamental visual properties, but may not be able to capture extremely fine-grained differences among classes. Nevertheless, this indicates that ICIS has indeed learned from the visual-semantic knowledge encoded in the given classifier weights for seen classes.

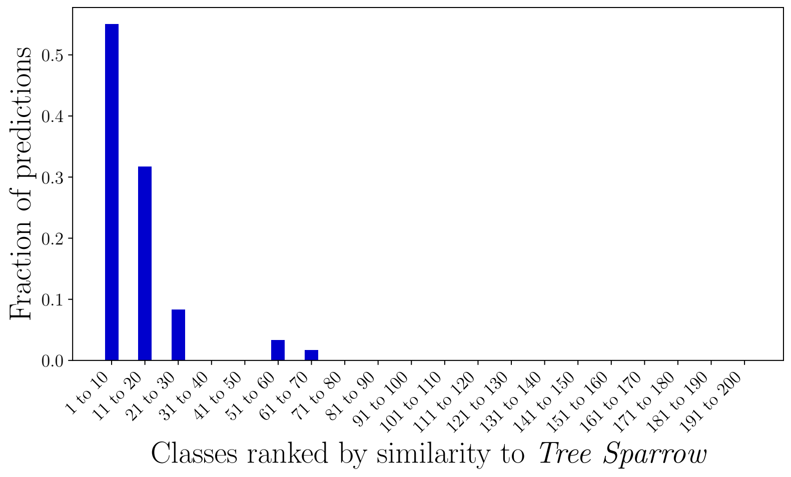

Figure 6: **Overview of the failure case *Tree Sparrow***, where the injected weights for the generalised zero-shot task on CUB only correctly classifies 5% of the samples. The x-axis indicates CUB classes ranked by the attribute similarity to *Tree Sparrow*, while the y-axis shows the number of times an image of the class *Tree Sparrow* is assigned to a group of classes. Most misclassifications are towards classes similar to the ground-truth one.

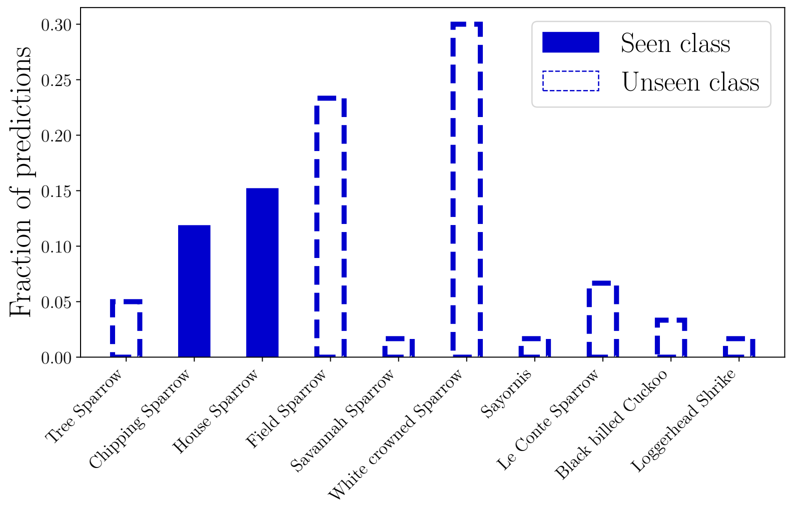

Figure 7: **Predicted classes in the failure case *Tree Sparrow***, where the x-axis indicates predicted CUB classes ranked by the attribute similarity to *Tree Sparrow*, while the y-axis shows the frequency of an image of *Tree Sparrow* being assigned to a group of classes. The misclassifications are distributed mostly between classes of other unseen (dashed) Sparrow-classes, as well as the two seen classes (solid) that are the closest to *Tree Sparrow* in terms of attributes.

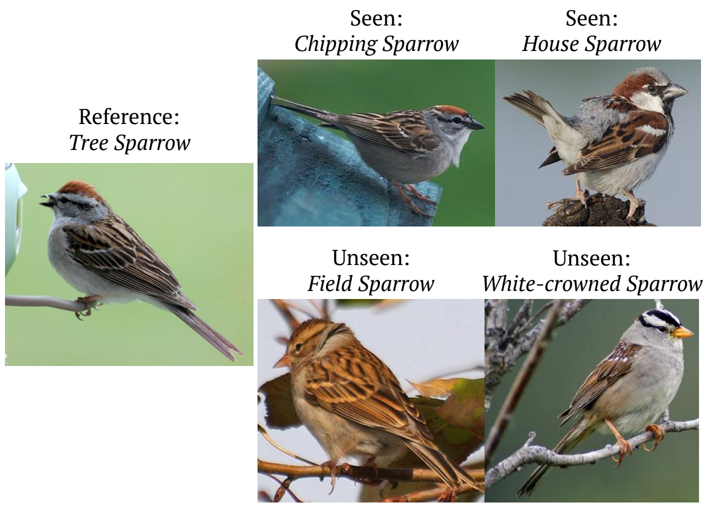

Figure 8: **Images from the seen and unseen classes most consistently confused with *Tree Sparrow*** by the classification weights predicted by our ICIS model.