Fine-Grained Zero-Shot Learning with DNA as Side Information
Supplementary Material

Although the main text is prepared to be self-contained, we provide further details for implementation, hyperparameter tuning, mathematical derivations used in the main text and some clarifications on parts that were left for the supplementary materials due to space restrictions.

1 Overview

In this section, we touch upon a brief overview of contributions we mentioned in the main text: BOLD datasets, DNA embeddings, and BZSL model intuition.

1.1 BOLD datasets

Since BOLD is an open-access database, a manual effort is needed to further curate a clean dataset. Figure 1 displays a small subset of images deleted during cleaning process. Note that, for INSECT dataset, only cases with images and matching DNA barcodes are included whereas for CUB dataset, we did not impose this restriction as we only needed DNA information. Consequently, we retrieved all DNA barcodes from bird species (extracted from COI genes only) present in BOLD.

Figure 2 presents further details on INSECT dataset such as number of species per genera and number of samples per species. From Figure 2(a), one can observe that dataset can be considered balanced as more than 90% of species have samples between 10 and 30. Fine-grained nature of the dataset, on the other hand, can be seen from Figure 2(b). Out of 578 genera, 270 of them have at least 2 species in the dataset. In 50 genera, there are more than 4 species coming from the same genus, which makes the data challenging yet, at the same time, provides a chance to find similar seen classes for the unseen classes.

1.2 DNA Embeddings

Although traditional sequencing methods [5, 4] have shown that DNA barcodes extracted from COI gene are very powerful to uniquely identify living organisms in species level, it was to our surprise to see that DNA embeddings obtained from a simple CNN model yielded more than 99% accuracy on INSECT dataset with $1,213$ species. On BIRD species dataset with more than 1000 classes, the same CNN model achieved 95.6% accuracy which clearly shows that DNA barcodes'

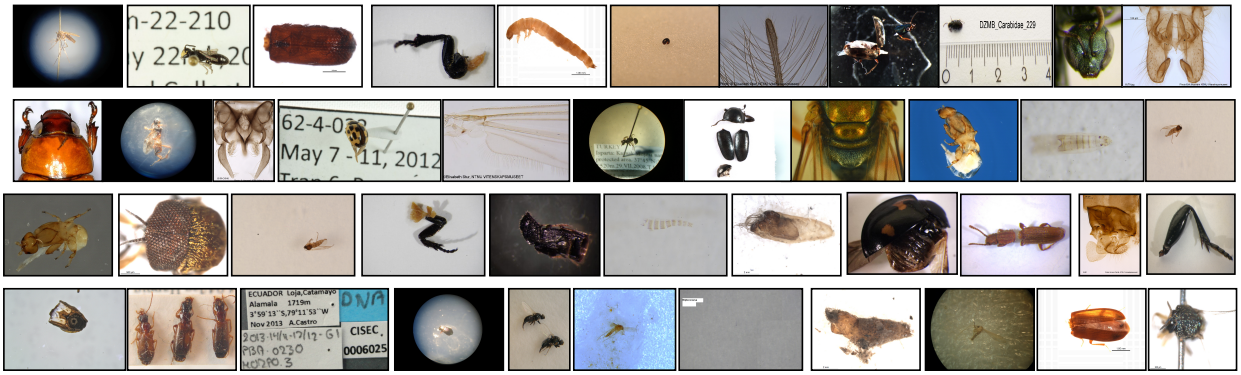


Figure 1: Small subset of sample images deleted from INSECT dataset during data cleaning. Images inside of a circle are taken from microscope camera, thus, had very low resolution. Some images display only body parts, which is enough to extract DNA information but useless for image classification. There are many images in which insects are positioned very far from camera, hence almost no morphological characteristics were visible.

utility is not bounded just by insects species. Figure 3 displays the TSNE plot of DNA embeddings from benchmark CUB dataset using randomly selected 15 classes. On one hand, species clusters are well-separated. On the other hand, species belonging to the same genus, inside the colored rectangles, grouped close to each other.

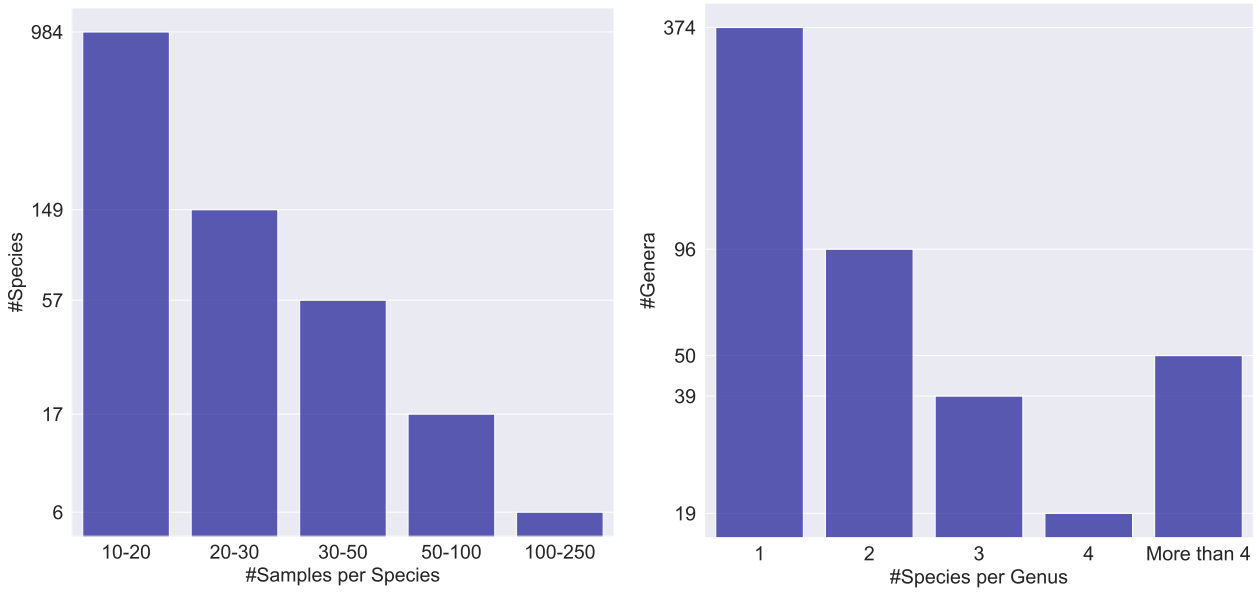
1.3 Hierarchical Bayesian Model Intuition

The Figure 4 below validates the model intuition that there is a deeper level of hierarchy among existing classes than a single global Bayesian prior can explain. Indeed, images from semantically similar classes are embedded close to each other due to their shared latent parameters.

2 Details of posterior predictive distribution (PPD) derivation

The proposed model (BZSL) assumes Gaussian data model and Normal-Inverse Wishart (NIW) prior on class parameters. Derivations for each class (seen and surrogate-unseen) are treated independently as our model imposes independence on data generation conditioned on local priors and global hyperparameters. By preserving the conjugacy, the assumption of common covariance among classes sharing the same local prior further simplified the derivations. Six Steps of the derivation are outlined in Figure 5 and pseudo-code is presented in Algorithm 1. Please refer to the Table 1 for all variables and parameters used in the calculations. Class sufficient statistics, which are only available for seen classes, are defined by $\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, S_{ji}$ and n_{ji}, which represent sample mean, scatter matrix and size of class i associated with local prior j, respectively. The notations ω_{jc} and ω_j used in Algorithm 1 represents current seen and unseen classes, whose PPD are being derived. The notation $\phi(\cdot)$ stands for attribute vector(s) of the corresponding class(es).

Although we haven't utilized local priors for seen class, please note that the following derivations are based on a more general setup, where seen classes PPD can also incorporate local priors. At



(a) Samples per Species

(b) Species per Genera

Figure 2: INSECT Data statistics

the end of the step 6, we will provide the exact PPD formulation that we used in our experiments in which local priors are omitted in Seen class PPD.

Parameter	**Description**
D	image feature space dimension
d	superscript that refers to the d^{th} component of each parameter
j	local prior index
i	actual class index
k	image index
c	index of current class
t_i	local prior indicator for class i
t_k	class indicator for data point k
K	# of neighbors of current class in surrogate-class
$\boldsymbol{\mu_j}$	mean of local prior j
Σ_j	covariance of local prior j
$\boldsymbol{\mu_{ji}}$	mean of class i of local prior j
$\bar{\boldsymbol{x}}_{\boldsymbol{ji}}$	sample mean of class i of local prior j
S_{ji}	scatter matrix of class i of local prior j
n_{ji}	size of class i of local prior j
$\boldsymbol{x_{jik}}$	data point k from class i of local prior j

Table 1: The notation used in the derivation of PPD.

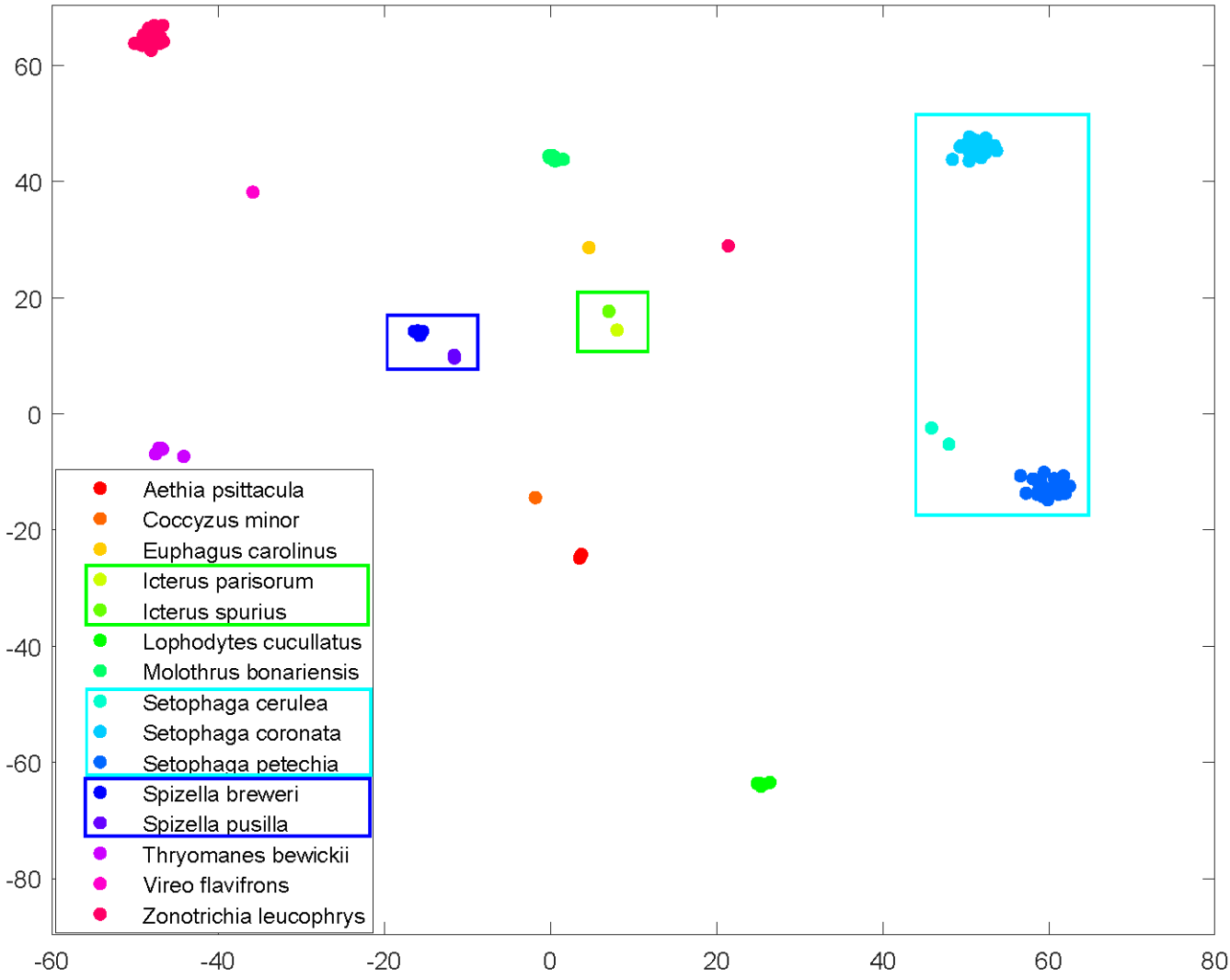
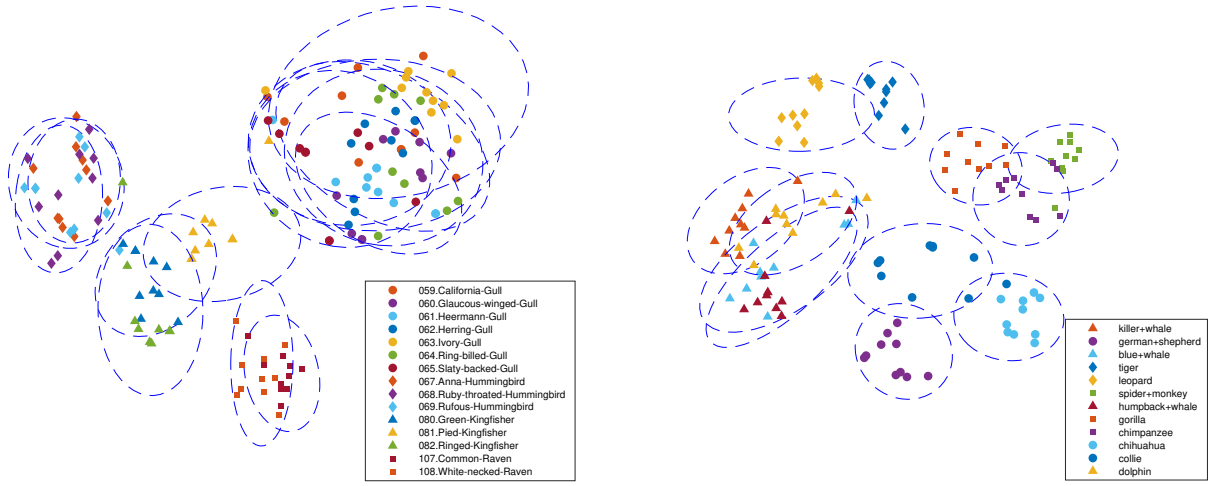
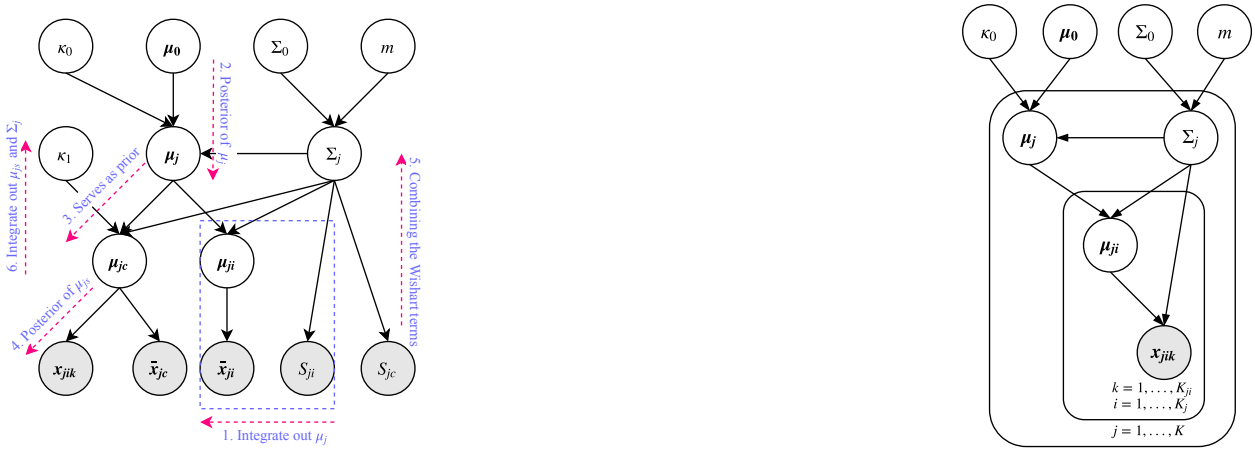


Figure 3: TSNE plots of DNA embeddings from CUB dataset. Class names are represented by birds' scientific names.



(a) CUB (b) AWA

Figure 4: t-SNE visualization from AWA and CUB datasets shows that classes similar in the attribute space indeed cluster closer in the feature space as well. For example, white-necked raven clusters next to common raven instead of Gull, Hummingbird and Kingfisher species. The same phenomenon also appears in coarse grained datasets (AWA) as different kinds of dogs, monkeys, cetaceans and carnivorous cats cluster together with their other kinds. Nevertheless, note that groups are not as intermingled as in fine grained CUB dataset.



(a) Graphical model (b) Plate diagram

Figure 5: PPD derivation outline

Sufficient Statistics

As Gaussian distribution requires only mean and covariance to be uniquely identified, hence data in classes can be summarized by their sufficient statistics; sample means and covariances.

$$P(\boldsymbol{x_{jik}}|\boldsymbol{\mu_{ji}}, \Sigma_j) \sim N(\boldsymbol{x_{jik}}|\boldsymbol{\mu_{ji}}, \Sigma_j) \tag{1}$$

$$\bar{\boldsymbol{x}}_{\boldsymbol{ji}} = \frac{1}{n_{ji}} \sum_{k:t_k=i} \boldsymbol{x_{jik}} \tag{2}$$

$$\bar{\boldsymbol{x}}_{\boldsymbol{ji}} \overset{5}{\sim} N(\boldsymbol{\mu_{ji}}, \Sigma_j n_{ji}^{-1}) \tag{3}$$

Algorithm 1 Posterior Predictive Distributions in BZSL

Input: Training data, $\phi(seen)$, $\phi(unseen)$

Output: PPD parameters for each seen class ($\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}}, \bar{v}_{jc}, \bar{\Sigma}_{jc}$) and unseen class ($\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{v}_j, \bar{\Sigma}_j$)

Set hyper-parameters: $\kappa_0, \kappa_1, m, s, K$
Compute $\boldsymbol{\mu_0}$ (mean of class means) and Σ_0 (mean of class covariances scaled by s)
for each seen class ω_{jc} **do** ▷ Images available
    Calculate current class params: $\bar{\boldsymbol{x}}_{\boldsymbol{jc}}, n_{jc}, S_{jc}$
    Calculate $S_{\boldsymbol{\mu}}$ (Eq 34)
    Calculate PPD by combining *global prior* and *data driven likelihood*: $\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}}, \bar{v}_{jc}, \bar{\Sigma}_{jc}$ (Eq 37)
end for
for each unseen class ω_j **do** ▷ No image available
    Find K most similar seen classes:
    $\mathcal{L}^2(\phi(\omega_j), \phi(seen))$
    for each selected seen class ω_{ji} **do**
        Calculate class params: $\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, n_{ji}, S_{ji}$
    end for
    Calculate $\tilde{\kappa}_j$ (Eq 30)
    Calculate PPD parameters using *local* and *global priors*: $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{v}_j, \bar{\Sigma}_j$ (Eq 38)
end for

$$S_{ji} = \frac{1}{n_{ji}} \sum_{k:t_k=i} (\boldsymbol{x_{jik}} - \bar{\boldsymbol{x}}_{\boldsymbol{ji}})(\boldsymbol{x_{jik}} - \bar{\boldsymbol{x}}_{\boldsymbol{ji}})^T \tag{4}$$

$$(n_{ji} - 1)S_{ji} \sim W(\Sigma_j, n_{ji} - 1) \tag{5}$$

(3) follows from eq. (2) and independence assumption given local prior parameters. (5) is a very definition of Wishart distribution as S_{ji} is scatter matrix of class i from local prior j.

Step 1: Marginal Likelihood

The class sample means $\bar{\boldsymbol{x}}_{\boldsymbol{ji}}{}'s$ are connected to their local prior (j) by integrating out the intermediate class parameter $\boldsymbol{\mu_{ji}}$. Note that all three parameters ($\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, \boldsymbol{\mu_{ji}}, \boldsymbol{\mu_j}$) are Normally distributed and terms depend on local prior covariance (Σ_j) are treated constant for this derivation.

$$P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1) = \int P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_{ji}}, n_{ji}, \Sigma_j) P(\boldsymbol{\mu_{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1) d\boldsymbol{\mu_{ji}} \tag{6}$$

$$= \int N(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_{ji}}, \Sigma_j n_{ji}^{-1}) N(\boldsymbol{\mu_{ji}}|\boldsymbol{\mu_j}, \Sigma_j \kappa_1^{-1}) \tag{7}$$

$$= \int (2\pi)^{-\frac{d}{2}} |\Sigma_j/n_{ji}|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_{ji}})^T (\Sigma_j/n_{ji})^{-1} (\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_{ji}})) \tag{8}$$

$$* (2\pi)^{-\frac{d}{2}} |\Sigma_j/\kappa_1|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\boldsymbol{\mu_{ji}} - \boldsymbol{\mu_j})^T (\Sigma_j/\kappa_1)^{-1} (\boldsymbol{\mu_{ji}} - \boldsymbol{\mu_j})) d\boldsymbol{\mu_{ji}}$$

$$= \int C_1 C_2 exp(-\frac{1}{2}(\boldsymbol{\mu_{ji}} - \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}})^T (\frac{\Sigma_j}{n_{ji} + \kappa_1})^{-1} (\boldsymbol{\mu_{ji}} - \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}}) + C_3) d\boldsymbol{\mu_{ji}} \tag{9}$$

$$= C_1 C_2 \exp(C_3) \int exp(-\frac{1}{2}(\boldsymbol{\mu_{ji}} - \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}})^T (\frac{\Sigma_j}{n_{ji} + \kappa_1})^{-1} (\boldsymbol{\mu_{ji}} - \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}}) d\boldsymbol{\mu_{ji}} \tag{10}$$

$$P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1) = C_1 C_2 \exp(C_3) (2\pi)^{\frac{d}{2}} |\frac{\Sigma_j}{\kappa_1 + n_{ji}}|^{\frac{1}{2}} \tag{11}$$

$$C_1 = (2\pi)^{-\frac{d}{2}} |\Sigma_j/n_{ji}|^{-\frac{1}{2}} \tag{12}$$

$$C_2 = (2\pi)^{-\frac{d}{2}} |\Sigma_j/\kappa_1|^{-\frac{1}{2}} \tag{13}$$

$$C_3 = -\frac{1}{2}(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}{}^T (\Sigma_j/n_{ji})^{-1} \bar{\boldsymbol{x}}_{\boldsymbol{ji}} + \boldsymbol{\mu_j}{}^T (\Sigma_j/\kappa_1)^{-1} \boldsymbol{\mu_j} - \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}}^T (\frac{\Sigma_j}{n_{ji} + \kappa_1})^{-1} \frac{\kappa_1 \boldsymbol{\mu_j} + n_{ji} \bar{\boldsymbol{x}}_{\boldsymbol{ji}}}{\kappa_1 + n_{ji}}) \tag{14}$$

$$C_3 = -\frac{1}{2}((\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_j})^T \frac{n_{ji} \kappa_1 \Sigma_j^{-1}}{(n_{ji} + \kappa_1)} (\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_j})) \tag{15}$$

$$P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1) = (2\pi)^{-\frac{d}{2}} |\frac{\Sigma_j(\kappa_1 + n_{ji})}{n_{ji} \kappa_1}|^{-\frac{1}{2}} \exp(-\frac{1}{2}(\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_j})^T \frac{n_{ji} \kappa_1 \Sigma_j^{-1}}{(n_{ji} + \kappa_1)} (\bar{\boldsymbol{x}}_{\boldsymbol{ji}} - \boldsymbol{\mu_j})) \tag{16}$$

$$P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1, n_{ji}) = N(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j (\frac{1}{n_{ji}} + \frac{1}{\kappa_1})) \tag{17}$$

(6), (7) and (8) follow the model assumption and definition of Normal distribution. (9) is derived by completing the (8) into normal distribution and combining extra elements into constant C_3. New term in the (11) comes from evaluation of integral from (10) as the exponential term is in the Gaussian form. Combining the similar terms in (14), we get (15), hence marginal likelihood (16). Hereon, it is trivial to observe that the likelihood is in Gaussian form with mean and covarinace as in (17).

Step 2: Posterior of $\boldsymbol{\mu_k}$

We combined the sufficient statistics (means) of classes sharing the same local prior in the posterior distribution of surrogate-class mean $\boldsymbol{\mu_j}$.

$$P(\boldsymbol{\mu_j}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) \propto P(\boldsymbol{\mu_j}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0) \prod_{i:t_i=j} P(\bar{\boldsymbol{x}}_{\boldsymbol{ji}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1) \tag{18}$$

$$P(\boldsymbol{\mu_j}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) \propto exp(-\frac{1}{2}(\boldsymbol{\mu_j} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})^T(\sum_{i:t_i=j} \frac{n_{ji}\kappa_1}{(n_{ji}+\kappa_1)} + \kappa_0)\Sigma_j^{-1}(\boldsymbol{\mu_j} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})) \tag{19}$$

$$P(\boldsymbol{\mu_j}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) = N(\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\kappa}_j^{-1}\Sigma_j) \tag{20}$$

$$\bar{\boldsymbol{\mu}}_{\boldsymbol{j}} = \frac{\sum_{i:t_i=j} \frac{n_{ji}\kappa_1}{(n_{ji}+\kappa_1)}\bar{\boldsymbol{x}}_{\boldsymbol{ji}} + \kappa_0\boldsymbol{\mu_0}}{\sum_{i:t_i=j} \frac{n_{ji}\kappa_1}{(n_{ji}+\kappa_1)} + \kappa_0} \tag{21}$$

$$\bar{\kappa}_j = (\sum_{i:t_i=j} \frac{n_{ji}\kappa_1}{(n_{ji}+\kappa_1)} + \kappa_0) \tag{22}$$

Applying Bayes rule, posterior can be proportioned as in (18). As local prior mean and sample mean are Normal distributed (from step 1), we get (19). Completing square procedure used in previous step would give the exact normalization, hence, posterior can be written in a closed form of Gaussian as in (20). The last part can also be verified by observing all exponential terms are quadratic, indeed Gaussian.

Step 3: Updated prior of $\boldsymbol{\mu_{jc}}$

As new information is available from classes sharing the same local prior, current class mean ($\boldsymbol{\mu_{jc}}$) can leverage this information by updating its prior. Marginalizing out local prior mean $\boldsymbol{\mu_j}$ would render this information propagation as below,

$$P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) = \int P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_j}, \Sigma_j, \kappa_1)P(\boldsymbol{\mu_j}|\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\Sigma}_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j})d\boldsymbol{\mu_j} \tag{23}$$

$$P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) = \int N(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_j}, \Sigma_j\kappa_1^{-1})N(\boldsymbol{\mu_j}|\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\Sigma}_j)d\boldsymbol{\mu_j} \tag{24}$$

$$P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) = N(\boldsymbol{\mu_{jc}}|\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\Sigma}_j + \Sigma_j\kappa_1^{-1}) \tag{25}$$

Step 4: Posterior on $\boldsymbol{\mu_{jc}}$

Combining the prior from step 3 with current class sample mean $\bar{\boldsymbol{x}}_{\boldsymbol{jc}}$ from step 1, we derive the posterior for current class mean $\boldsymbol{\mu_{jc}}$. Analogously, applying Bayes rule and observing that both

distributions are Normal, we obtain an another Gaussian.

$$P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}) \tag{26}$$

$$\propto P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}) P(\bar{\boldsymbol{x}}_{\boldsymbol{jc}}|\boldsymbol{\mu_{jc}}, \Sigma_j n_{jc}^{-1}) \tag{27}$$

$$\propto N(\boldsymbol{\mu_{jc}}|\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\Sigma}_j + \Sigma_j \kappa_1^{-1}) N(\bar{\boldsymbol{x}}_{\boldsymbol{jc}}|\boldsymbol{\mu_{jc}}, \Sigma_j n_{jc}^{-1}) \tag{28}$$

$$P(\boldsymbol{\mu_{jc}}|\boldsymbol{\mu_0}, \Sigma_j, \kappa_0, \kappa_1, \{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}) = N(\frac{n_{jc}\bar{\boldsymbol{x}}_{\boldsymbol{jc}} + \tilde{\kappa}_j \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}}{n_{jc} + \tilde{\kappa}_j}, \Sigma_j(\tilde{\kappa}_j^{-1} + n_{jc}^{-1})) \tag{29}$$

$$\tilde{\kappa}_j^{-1} = \bar{\kappa}_j^{-1} + \kappa_1^{-1} \tag{30}$$

Note that in order to simplify the notation, κ terms are collected in $\tilde{\kappa}_j^{-1}$.

Before stepping into the covariance related derivations, we would like to draw your attention to an expression similar to C_3 from equation (15) that appears with $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ in the remaining terms while deriving posterior.

$$(\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})^T \frac{n_{jc}\tilde{\kappa}_j}{(n_{jc} + \tilde{\kappa}_j)} \Sigma_j^{-1} (\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}) = \frac{n_{jc}\tilde{\kappa}_j}{n_{jc} + \tilde{\kappa}_j} tr((\Sigma_j^{-1})(\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})(\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})^T)) \tag{31}$$

Note that $\bar{\boldsymbol{x}}_{\boldsymbol{jc}}$ and $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ are observed values, and the equation depends only on Σ_j. This formula stays in the exponent creating a factor that contributes to Wishart distribution and is denoted as $P(S_\mu|\Sigma_j)$.

Step 5: Wishart Terms

As it is observed from plate diagram and generative model in Figure (5), local prior covariance Σ_j is shared among its all classes. Even though Wishart terms are independent of mean variables, the residual terms in the posterior calculations create additional Wishart distributions. These distributions and the ones from data scatter matrices are combined in the posterior of Σ_j as below,

$$P(\Sigma_j|\{S_{ji}, \bar{\boldsymbol{x}}_{\boldsymbol{ji}}\}_{t_i=j}, S_{jc}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}) \propto P(\Sigma_j|\Sigma_0, m) P(S_{jc}|\Sigma_j, n_{jc}) P(S_\mu|\Sigma_j) \prod_{i:t_i=j} P(S_{ji}|\Sigma_j, n_{ji}) \tag{32}$$

$$= IW(\Sigma_0 + \sum_{i:t_i=j} S_{ji} + S_{jc} + S_\mu,\ m + \sum_{i:t_i=j} (n_{ji} - 1) + n_{jc}) \tag{33}$$

$$S_\mu = \frac{n_{jc}\tilde{\kappa}_j}{\tilde{\kappa}_j + n_{jc}} (\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})(\bar{\boldsymbol{x}}_{\boldsymbol{jc}} - \bar{\boldsymbol{\mu}}_{\boldsymbol{j}})^T \tag{34}$$

Since the prior on Σ_j is Inverse-Wishart and scatter matrices are Wishart distributed, the posterior will be exactly Inverse-Wishart.

Step 6: Integration of remaining parameters

As you notice from Step 4 and 5, the posterior distribution is Normal-Inverse-Wishart and from model assumption, data is Gaussian. Hence, the integration in PPD can analytically be derived. Integration with respect to $\boldsymbol{\mu}$ will render another multivariate Normal distribution due to conjugacy,

thereafter integration w.r.t Σ completes to Inverse-Wishart. Finally, arranging the terms yields the posterior predictive distribution in the form of Student-t.

$$P(\boldsymbol{x}|\{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, S_{ji}\}_{t_i=j}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}, S_{jc}, \boldsymbol{\mu_0}, \kappa_0, \kappa_1) \tag{35}$$

$$= \int\int P(\boldsymbol{x}|\boldsymbol{\mu_{jc}}, \Sigma_j) P(\boldsymbol{\mu_{jc}}, \Sigma_j|\{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, S_{ji}\}_{t_i=j}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}, S_{jc}, \boldsymbol{\mu_0}, \kappa_0, \kappa_1) d\boldsymbol{\mu_{jc}} d\Sigma_j$$

$$= T(\boldsymbol{x}|\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}}, \bar{\Sigma}_{jc}, \bar{v}_{jc}) \tag{36}$$

$$\begin{aligned}
\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}} &= \frac{n_{jc}\bar{\boldsymbol{x}}_{\boldsymbol{jc}} + \tilde{\kappa}_j \bar{\boldsymbol{\mu}}_{\boldsymbol{j}}}{n_{jc} + \tilde{\kappa}_j} \\
\bar{v}_{jc} &= n_{jc} + \sum_{i:t_i=j} (n_{ji} - 1) + m - d + 1 \\
\bar{\Sigma}_{jc} &= \frac{n_{js} + \tilde{\kappa}_j + 1}{(n_{jc} + \tilde{\kappa}_j)\bar{v}_{jc}} (\Sigma_0 + \sum_{i:t_i=j} S_{ji} + S_{jc} + S_\mu)
\end{aligned}$$

Note that aforementioned derivation of seen class PPD includes local priors, yet in our experiments we simply dropped local priors from Seen class PPD. Hence, the final PPD for seen classes in which local priors are omitted is derived also in the form of a Student-t distribution as following,

$$\begin{aligned}
&P(\boldsymbol{x}|\{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, S_{ji}\}_{t_i=j}, \bar{\boldsymbol{x}}_{\boldsymbol{jc}}, S_{jc}, \boldsymbol{\mu_0}, \kappa_0, \kappa_1) = T(\boldsymbol{x}|\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}}, \bar{\Sigma}_{jc}, \bar{v}_{jc}) \\
&\bar{\boldsymbol{\mu}}_{\boldsymbol{jc}} = \frac{n_{jc}\bar{\boldsymbol{x}}_{\boldsymbol{jc}} + \frac{\kappa_0\kappa_1}{\kappa_0+\kappa_1}\boldsymbol{\mu_0}}{n_{jc} + \frac{\kappa_0\kappa_1}{\kappa_0+\kappa_1}}, \; \bar{v}_{jc} = n_{jc} + m - D + 1, \; \bar{\Sigma}_{jc} = \frac{(\Sigma_0 + S_{jc} + S_\mu)(n_{jc} + \frac{\kappa_0\kappa_1}{\kappa_0+\kappa_1} + 1)}{(n_{jc} + \frac{\kappa_0\kappa_1}{\kappa_0+\kappa_1})\bar{v}_{jc}}
\end{aligned} \tag{37}$$

where, S_μ is defined as in Eq (34). The index c in Equation (37) represents the current seen class, whose PPD is being derived.

Dropping the current class statistics from Eq (36), on the other hand, delivers the PPD for unseen classes as below,

$$\begin{aligned}
&P(\boldsymbol{x}|\{\bar{\boldsymbol{x}}_{\boldsymbol{ji}}, S_{ji}\}_{t_i=j}, \boldsymbol{\mu_0}, \kappa_0, \kappa_1) = T(\boldsymbol{x}|\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}, \bar{\Sigma}_j, \bar{v}_j) \\
&\bar{v}_j = \sum_{i:t_i=j} (n_{ji} - 1) + m - d + 1, \qquad \bar{\Sigma}_j = \frac{(\tilde{\kappa}_j + 1)}{\tilde{\kappa}_j \bar{v}_j} (\Sigma_0 + \sum_{i:t_i=j} S_{ji})
\end{aligned} \tag{38}$$

where $\bar{\boldsymbol{\mu}}_{\boldsymbol{j}}$ is from (21).

3 Training Details

In this section, we provide training details regarding implementation and hyperparameter tuning for CNN model, BZSL model and other state-of-the-art ZSL methods.

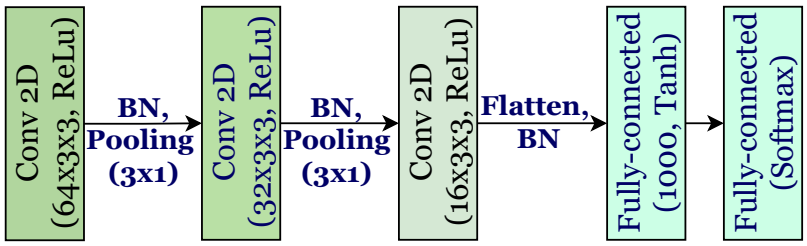


Figure 6: CNN model architecture

3.1 Implementation

Experiments are run on two machines: (1) a Dell PC with Windows 10, Intel(R) Core(TM) i9-9900 CPU @ 3.10 GHz, 64 GB RAM, and NVIDIA GeForce RTX 2080 GPU, 8GB RAM (2) School server with NVIDIA Tesla V100-PCIE GPU, 16GB RAM. LambdaLab [2] becnhmarking tests reveal that on average RTX 2080 is 56% as fast as Tesla V100 GPU for Deep Learning training. All models except FGNs are run on the PC. CADA-VAE and LsrGan are run on the school server.

3.1.1 CNN model

The details of CNN model training are mentioned in the main text, but here we present the detailed model architecture in Figure 6[1]. The same model and parameters are utilized in learning both INSECT and CUB datasets' DNA embeddings, except the sequence length. Since in INSECT dataset more than 90% of DNA barcodes have a length of 658, we transformed barcodes into 658x5 2D arrays during one-hot endcoding, whereas for bird dataset, we utilized maximum sequence length, 1500, for one-hot encoding 2D array. The reason for the latter was that bird DNA barcode lengths have a high variance and results on validation set maximized once we use maximum sequence length. For the padding of missing bases, we simply used the label *others* which represents missing and ambiguous symbols.

Jupyter notebook together with $Readme$ file for learning DNA embeddings is attached to the Supplementary material with the name *DNA_embeddings.zip*. Please refer to the $Readme$ file for setting up the *conda* virtual environment and installing required packages in order to run the code.

3.1.2 BZSL

The model is developed in MATLAB, and any version including 2016 and above should be able to run the model without any problem. The code is attached to the Supplementary material under the same *BZSL.zip*. You may find it useful to check the *Readme* file to reproduce reported results.

3.1.3 Other ZSL methods

We compared the proposed model against six state-of-the-art ZSL approaches and five of them have a publicly available codes. We got the code for ALE [1] from the authors themselves and it is

[1]In the main text, we mentioned that CNN architecture will be present in Figure 3 of Supplementary material but due to extra experiments, architecture is presented here. We are sorry for the inconvenience

developed in MATLAB. Methods and links to their codes are presented below:

- CADA-VAE [7]: https://github.com/edgarschnfld/CADA-VAE-PyTorch. The code is developed by authors in *PyTorch*. GPU enabled code.
- LsrGan [9]: https://github.com/Maunil/LsrGAN. The code is dveloped by authors in *PyTorch*. GPU enabled code.
- CRNet [10]: https://github.com/Fezaries/CRnet. The code is developed in *Python*. GPU enabled code.
- RelationNet [8]: https://github.com/lzrobots/LearningToCompare_ZSL. The code is developed by authors in *PyTorch*. GPU enabled code.
- ESZSL [6]: https://github.com/mvp18/Popular-ZSL-Algorithms. The code is developed in *Python*.

3.2 Hyperparameter Tuning

3.2.1 CNN Model

We very coarsely tuned the CNN model. After fixing the model architecture, we tuned initial learning rate, batch size and number of epochs between the ranges given below:

- Learning rate: $\{0.1, 0.01, 0.05, 0.001\}$
- Batch size: $\{32, 64\}$
- Number of epochs: $\{5, 10\}$

We also tried *SGD* optimizer but *ADAM* was superior.

3.2.2 BZSL

Referencing the main text, unconstrained model has 5 parameters to tune: $\{\kappa_0, \kappa_1, K, m, s\}$. As you may recall from the main text, hhe hyperparameter κ_0 adjusts the separation between local prior centers, on the other hand, κ_1 adjusts the dispersion between class centers inheriting the same local prior. Moreover, m controls the degree of deviation of individual Σ_j's from the $E[\Sigma_j]$ and s is the scale constant. As the expected value of Σ_j is $\frac{\Sigma_0}{m-d-1}$ where d is the dimension of the data, larger values for m assumes classes with more spherical shapes, whereas smaller values creates resilience to learn more flexible shapes.

Model hyperparameters are coarsely tuned to maximize the Harmonic mean on validation set. The only preprocessing we did was to apply PCA to reduce the dimensionality of the data from 2048 to 500. The range of each parameter and the best quintuplets associated with each dataset are depicted in the Tables 2 and 3, respectively. Running time of the model was 33 seconds per trial.

HP	Range
κ_0	$\{0.1, 1\}$
κ_1	$\{1, 10, 25\}$
K	$\{1, 2, 3\}$
m	$\{5d, 25d, 100d, 500d\}$
s	$\{1, 5, 10\}$

Table 2: Parameter ranges used in hyper-parameter tuning

Datasets	Best quintuplets
INSECT	$\{0.1, 10, 5d, 10, 3\}$
CUB (w. Att)	$\{1, 25, 500d, 10, 3\}$
CUB (w. w2v)	$\{0.1, 25, 5d, 5, 2\}$
CUB (w. DNA)	$\{0.1, 25, 25d, 5, 3\}$

Table 3: Best quintuplets from tuning with the order of $\{\kappa_0, \kappa_1, m, s, K\}$.

3.3 Other methods

Out of six ZSL methods, four of them are based on neural networks and it is not feasible, if not impossible, to tune all hyperparameters. Thus, we utilized Hyperopt [3], a distributed hyperparameter optimization technique developed in Python to tune all SotA ZSL methods, except ALE. All models are tuned to maximize the harmonic mean.

ALE was built on MATLAB and has only two parameters to tune: learning rate and margin constant, hence no need to use sophisticated software for tuning. Learning rate was tuned in the set of $\{1, 0.5, 0.25, 0.1, 0.05, 0.025, 0.01, 0.005, 0.001\}$, whereas margin constant was tuned from the set of $\{0.01, 0.1, 0.5, 1\}$. The model is run 100 epochs with early stop is set to 30 epochs.

For other methods, hyperparameters and their range are listed below (ranges are arranged by observing parameters used by authors for different benchmark datasets and then slightly extended):
CADA-VAE (100 runs):

- Classifier learning rate (LR) (*lr_cls*): *log-uniform* from range of $[log(1e-5), log(1e-2)]$
- Generative model LR (*lr_gen_model*): *log-uniform* from range of $[log(1e-5), log(1e-2)]$
- Classifier training steps (*cls_train_steps*): *random-sampling* from set of $\{4 : 1 : 40\}$
- Batch size (*batch_size*): *random-sampling* from set of $\{32, 64, 96, 128\}$
- Latent space dimensionality (*latent_size*): *random-sampling* from set of $\{32, 64, 96, 128\}$
- Regularizer loss (*loss*): *random-sampling* from set of $\{L1, L2\}$

LsrGan (25 runs due to slow training time, see Table 5):

- Class weight (*cls_weight*): *log-uniform* from range of $[log(1e-3), log(1e-1)]$
- LR (*lr*): *log-uniform* from range of $[log(1e-6), log(1e-3)]$
- Unseen class weight (*unseen_cls_weight*): *log-uniform* from range of $[log(1e-2), log(0.9)]$
- Epsilon (*epsilon*): *log-uniform* from range of $[log(1e-2), log(0.9)]$
- Upper epsilon (*upper_epsilon*): *log-uniform* from range of $[log(1e-2), log(0.9)]$

- Number of synthesized features per class (*syn_num*): *random-sampling* from set of $\{100 : 100 : 3000\}$
- Number of epochs (*nepoch*): *random-sampling* from set of $\{10 : 5 : 50\}$
- Correlation penalty (*correlation_penalty*): *random-sampling* from set of $\{5 : 10 : 50\}$
- Pretrained classifier for inference (*no_classifier*): *random-sampling* from set of $\{True, False\}$

CRNet (50 runs)

- LR (*LEARNING_RATE*): *log-uniform* from range of $[log(1e-6), log(1e-2)]$
- Weight decay (*WEIGHT_DECAY*): *log-uniform* from range of $[log(1e-20), log(1e-3)]$
- Number of clusters (*K*): *random-sampling* from set of $\{1 : 1 : 21\}$

RelationNet (50 runs)

- LR (*learning_rate*): *log-uniform* from range of $[log(1e-6), log(1e-3)]$
- Number of episodes (*episode*): *random-sampling* from set of $\{50,000 : 10,000 : 200,001\}$
- Step size of learning rate scheduler (*lr_step_size*): *random-sampling* from set of $\{10,000 : 10,000 : 200,001\}$

ESZSL (100 runs)

- Regularization constant alpha (*alpha*): *uniform* from range of $[-10, 10]$
- Regularization constant gamma (*gamma*): *uniform* from range of $[-10, 10]$

4 Additional Experiments

4.1 Ablation study

We conduct an ablation study for the proposed model to investigate the necessity and efficacy of different components of the model. Starting with the obvious one, breaking the hierarchy and removing the second layer, the model will not be able to form local priors for the surrogate classes, thus all unseen classes would be assigned to other seen classes. The utility of the Bayesian aspect of the model can be tested by comparing the proposed model against a standard Gaussian Mixture Model (GMM). In this experiment, we eliminated the global and local priors, hence, each seen class is fit a single Gaussian and each unseen class is fit a GMM with K components. Table 4 exhibits the harmonic means of the Bayesian model and GMM on INSECT and CUB (with 3 different sources of side information) datasets. Performance drop is particularly drastic on the INSECT data due to the poor unseen class accuracy. The Bayesian aspect is not only necessary for a better knowledge transfer from seen to unseen classes, but hyperparameters of the model also provide the flexibility to model datasets with various levels of granularity, which brings us to the next ablation study.

4.2 Model Runtime Analysis

Methods	INSECT	CUB (Att)	CUB (W2V)	CUB (DNA)
GMM	5.01	26.20	21.11	18.97
BZSL	**26.99**	**38.82**	**29.94**	**34.97**

Table 4: Fitting GMM instead of Bayesian hierarchy. We tuned number of components, K, and reported the best results for GMM.

Table 5 displays runtime of all ZSL methods on CUB dataset while visual attributes are used as side information. All models but LsrGan are run on the PC for this experiment. On server, average runtime per trial for LsrGan was $1,834$ seconds and using the LambdaLab research as a reference point an approximate runtime of LsrGan on PC is calculated as $1,834/0.56 = 3,274$ seconds. ESZSL and BZSL has the lowest running time per trial and BZSL is 30 times faster than the next fastest method, CADA-VAE. Both methods owe the super fast training time to their closed form solutions.

Method	**Ave. runtime per trial**
CADA-VAE	280
LsrGan	$3,274$
CRNet	$1,625$
RelationNet	882
ALE	513
ESZSL	3.6
BZSL	8.6

Table 5: Running time in seconds per trial on CUB dataset (our version in which 6 classes are not present).

4.3 Visualization of Synthesized Features from FGNs.

To better understand the effect of various side informations on FGNs' performance, we visualized generated unseen class features from CADA-VAE and LsrGan on CUB data using 3 different side information sources: visual attributes (common one), word2vecs and DNA embeddings. For each experiments we sampled 60 points for TSNE training and then randomly sampled 20 unseen classes for visualization. Figure 7 displays the TSNE plot from CUB data. Since the same randomly selected 20 unseen classes are used for all figures, we only put one legend and it is in the Figure 7. Please note that, we utilized the best setup from tuning for the model training and we only sampled 20 classes after training done for only visualization purposes. TSNE plots of synthesized features from these 2 models using visual, word2vec and DNA attributes are presented in Figures 8, 9, 10, respectively.

The first thing to notice is the bizarre distribution of synthesized features from CADA-VAE. The model to some extent is able to transfer the relative inter-class proximity from the attribute space to image feature space, and achieves competitive results yet the generated features lack quality. The superior results of LsrGan when visual attributes are available is visible by the TSNE plot in Figure 8(b). However, once the correlation between side information and image features decreases, the quality of generated features suffers. Both methods display mode collapse once word vectors are used as side information (see red circle in Figure 9). On the other hand, with DNA embeddings, features from LsrGan are scattered all around and fail to form meaningful clusters. Despite their strange shapes, features generated by CADA-VAE seem to be slightly more reasonable and it is reflected in their better performance with DNA embeddings.

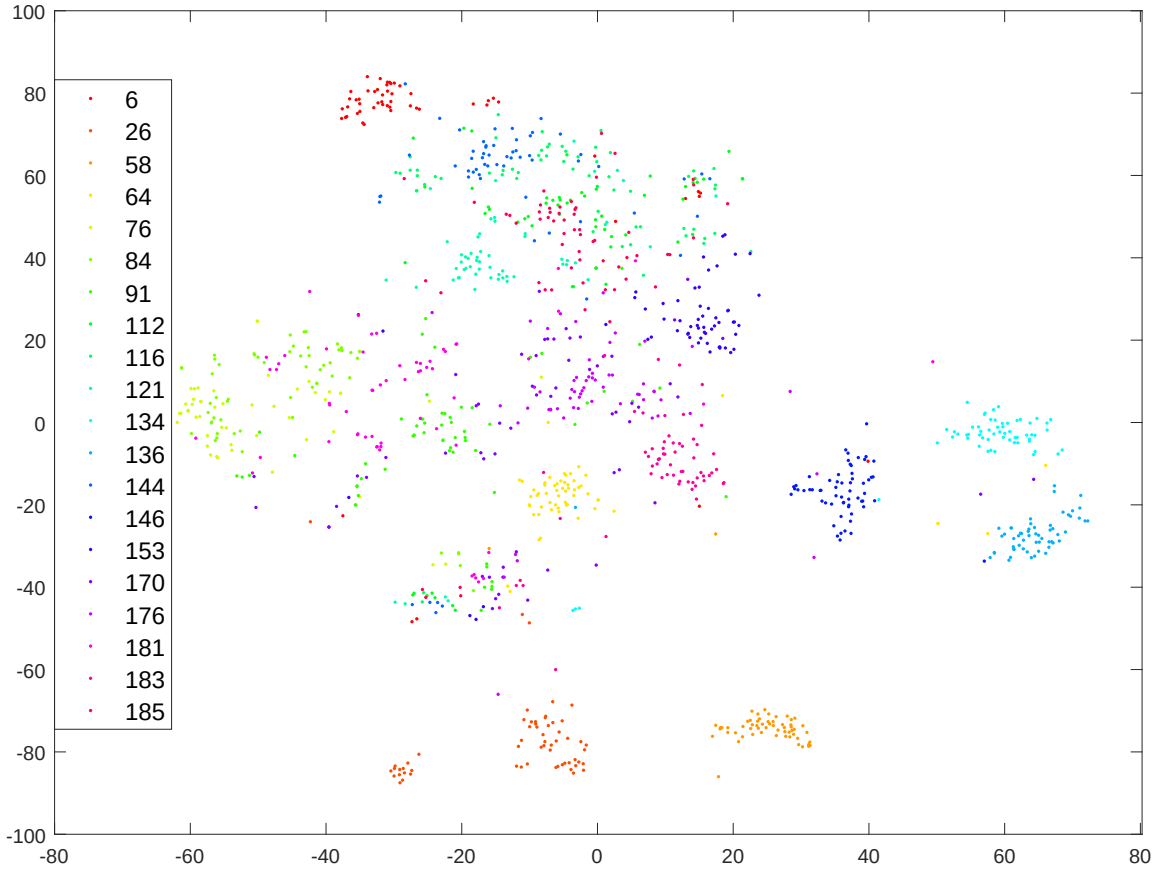
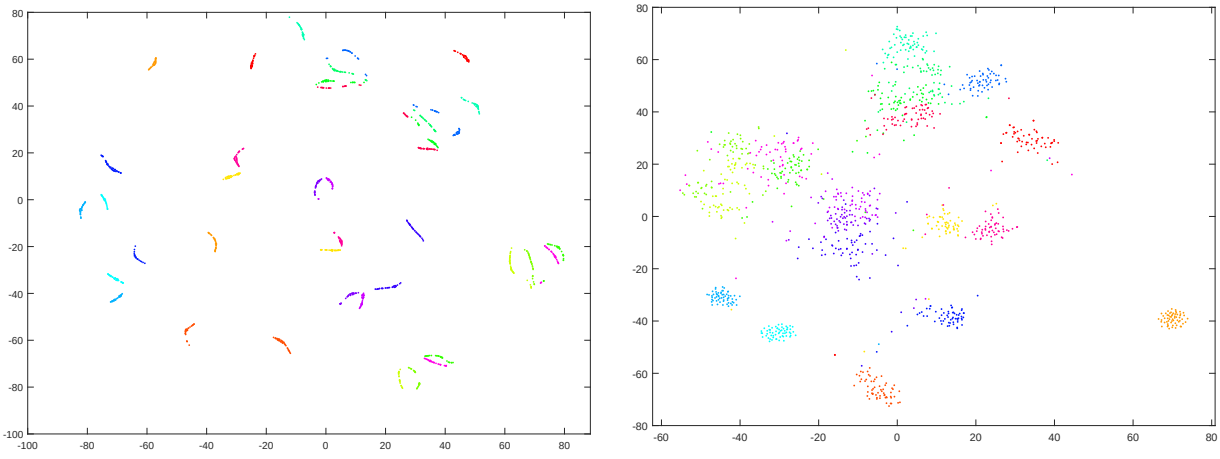
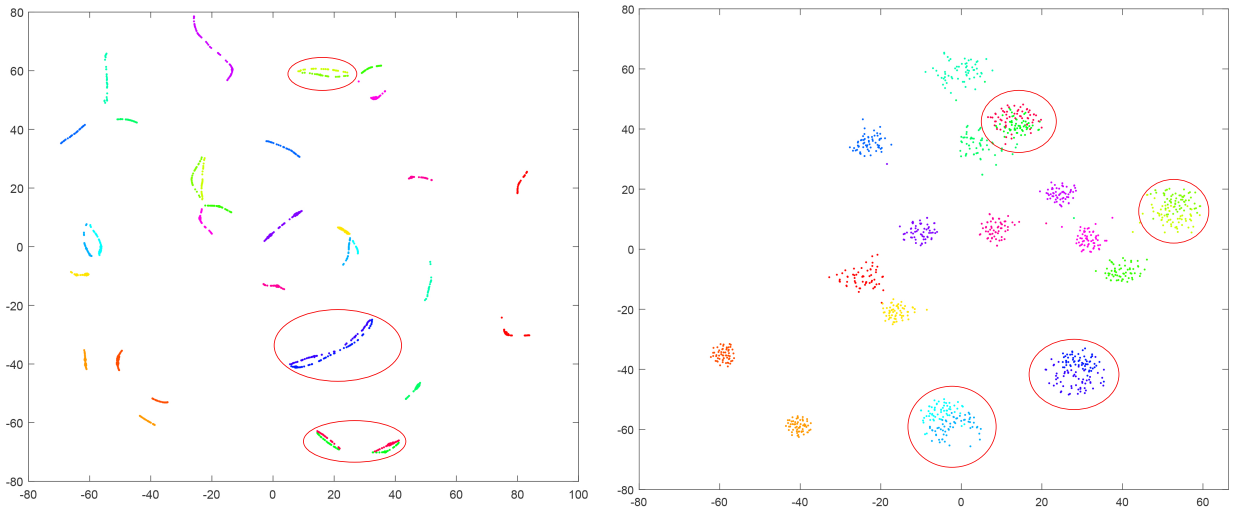


Figure 7: TSNE plot of randomly sampled 20 unseen classes from CUB data



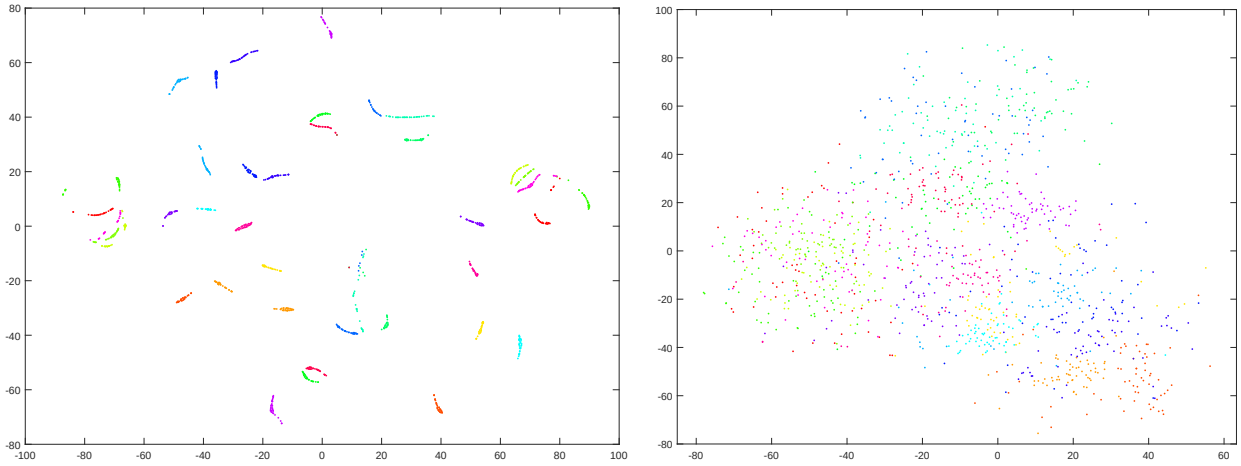
(a) Synthesized unseen class features from CADA-VAE (b) Synthesized unseen class features from LsrGan

Figure 8: TSNE plots using visual attributes as side information during model training on CUB data



(a) Synthesized unseen class features from CADA-VAE (b) Synthesized unseen class features from LsrGan

Figure 9: TSNE plots using word2vec as side information during model training on CUB data



(a) Synthesized unseen class features from CADA-VAE (b) Synthesized unseen class features from LsrGan

Figure 10: TSNE plots using DNA as side information during model training on CUB data

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *TPAMI*, 2016. 11

[2] S. Balaban, C. Li, and M. Balaban. Deep learning gpu benchmarks - tesla v100 vs rtx 2080 ti vs gtx 1080 ti vs titan v. 11

[3] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*, 2013. 13

[4] P. D. Hebert, M. Y. Stoeckle, T. S. Zemlak, and C. M. Francis. Identification of birds through dna barcodes. *PLoS biology*, 2, 2004. 1

[5] D. H. Lunt, D. X. ZHang, J. M. Szymura, and G. M. Hewitt. The insect cytochrome oxidase i gene: evolutionary patterns and conserved primers for phylogenetic studies. *PLoS biology*, 5, 1996. 1

[6] B. Romera-Paredes and P. H. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 12

[7] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrel, and Z. Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. In *CVPR*, 2019. 12

[8] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 12

[9] M. R. Vyas, H. Venkateswara, and S. Panchanathan. Leveraging seen and unseen semantic relationships for generative zero-shot learning. In *ECCV*, 2020. 12

[10] F. Zhang and G. Shi. Co-representation network for generalized zero-shot learning. In *ICML*, 2019. 12