Normalization Matters in Zero-Shot Learning

Ivan Skorokhodov
KAUST, MIPT
Thuwal, Saudi Arabia, 23955
iskorokhodov@gmail.com

Mohamed Elhoseiny
KAUST
Thuwal, Saudi Arabia, 23955
mohamed.elhoseiny@kaust.edu.sa

Abstract

An ability to grasp new concepts from their descriptions is one of the key features of human intelligence, and zero-shot learning (ZSL) aims to incorporate this property into machine learning models. In this paper, we theoretically investigate two very popular tricks used in ZSL: "normalize+scale" trick and attributes normalization and show how they help to preserve a signal's variance in a typical model during a forward pass. Next, we demonstrate that these two tricks are not enough to normalize a deep ZSL network. We derive a new initialization scheme, which allows us to demonstrate strong state-of-the-art results on 4 out of 5 commonly used ZSL datasets: SUN, CUB, AwA1, and AwA2 while being on average 2 orders faster than the closest runner-up. Finally, we generalize ZSL to a broader problem — Continual Zero-Shot Learning (CZSL) and test our ideas in this new setup. The source code to reproduce all the results is available at https://github.com/universome/czsl.

1 Introduction

A good intelligent system must be able to capture new concepts just by having their descriptions. For example, when a parent describes to their child how a monster from a fairy tale looks like, the child does not need to see this monster several times before it will be able to differentiate between monsters and non-monsters. Unfortunately, modern neural networks still struggle to have such property and zero-shot learning (ZSL) is a research field that tries to remedy this problem.

In this work, we focus on zero-shot image classification which is a dominant direction in ZSL research and investigate two ubiquitous tricks that are employed by practitioners: "normalize + scale" trick and attributes normalization trick. These two tricks are typically motivated by intuition [9] and, to the best of our knowledge, there is no rigorous understanding of the nature of the benefit they bring.

In this work, we provide a theoretical justification that the real reason why they aid training is the normalization of a signal during a forward pass by controlling its variance not to vanish or blow up. Next, we show that just using these two tricks are not enough for the variance control in a deep architecture. Thus we derive a new initialization scheme that fixes this problem.

We test our approach on 5 commonly used ZSL datasets and obtain state-of-the-art results on 4 of them. Apart from that, due to its simplicity, our approach is also orders of magnitudes faster to train than SotA methods.

Apart from this theoretical analysis and the new initialization scheme, we put our hand on generalizing zero-shot learning to a more broader setup: continual zero-shot learning (CZSL). An ability to acquire new knowledge without forgetting it afterwards is an essential property for any intelligent system and it motivates us to consider this property in ZSL models as well. We develop ideas of [11] in this direction and formulate a new scenario for training ZSL agents. We also generalize existing ZSL metrics to assess a model's behaviour for the new setup and test our approach with the corresponding baselines in this playground.

Preprint. Under review.

2 Related work

Zero-shot learning (ZSL) has a long-standing history [41] and nowadays people employ zero-shot models not only for image classification, but also for object detection, text classification, named entity recognition, semantic segmentation and many other tasks [52, 4, 5].

In the image classification domain, people tend to build their models not in a raw pixel space, but in some feature space, where features are computed either using classical computer vision methods [30, 43] or extracted from a pretrained encoder [49]. ZSL methods for image classification can be roughly divided into two groups: generative-based and embedding-based. The main goal for generative-based approaches is to build a conditional generative model that would synthesize images conditioned on class descriptors [50, 55, 13, 23, 22, 35]. At test time, one then generates a synthetic classification dataset using unseen class attributes and uses it to train a traditional classifier or to perform kNN-classification for the test images. Embedding-based approaches try to find such a mapping that embeds attributes in the feature space in such a way that the images have small distance with attribute embeddings of the corresponding class [43, 18, 1, 54].

There are two almost ubiquitous tricks that are used in the embedding-based ZSL: "normalize+scale" and attributes normalization. The first one is about using cosine similarity with the additional large scaling instead of the dot product while computing the logits [38, 39, 53]. The second trick is the division of attributes by their L_2 norm instead of converting them to zero-mean and unit variance [8, 38, 53, 55] as is usually done for input data. These two tricks are crucial for good performance and Table 1 ablates their importance.

Our work continues the development of the embedding-based ZSL and we build upon a deep attribute embedder, similar to [38]. But in our case we do not use the episodic training and optimize our models with a traditional optimization procedure. We theoretically investigate how the above two tricks control a signal's variance inside a model and show that they are not enough to normalize a deep attribute embedder. To alleviate this issue we propose a proper initialization scheme which is based on a different initialization variance and a dynamic standardization layer.

One of the most influential works on initialization is Xavier's init [20] where authors showed how to preserve the variance during a forward pass. [25] applied similar analysis but taking ReLU activations into account. Apart from these two dominating methods, there are some less frequently used ones. For example, there is a growing interest in two-step [31], data-dependent [34] and orthogonal initialization schemes [26]. However, the importance of a good initialization for attribute embedding functions in zero-shot learning is not well understood and our work aims to fill this gap.

In [6], authors developed a proper initialization scheme for hypernetworks [24, 29] and it was their analysis that motivated our work. Hypernetworks seem similar in nature to attribute embedding models since both of them generate a classification matrix, but they are very different under the hood. Our scenario differs from [6] in several principal ways: a) we consider transformation $\boldsymbol{y} = \boldsymbol{x}^\top W H(A)$ instead of $\boldsymbol{y} = \boldsymbol{x}^\top \mathbb{W} H(\boldsymbol{a})$ (here $\mathbb{W}$ is a 3D matrix); b) we do not have an independence assumption for attribute vectors; c) we do not have a zero-mean assumption for attribute vectors; and d) we do not consider the covariance matrix of the form $\sigma^2 I$. Without these four conditions it would be unrealistic to apply the analysis of [6] for ZSL due to the nature of the models and training procedures used in ZSL. We elaborate on this in Section 3 and Appendix G.

A closely related branch of research is the development of normalization layers for deep neural networks [27, 3, 47] since they also influence a signal's variance. When one applies BatchNorm before a non-linearity or when batchnorm is used with piecewise-linear activations (like ReLU or LeakyReLU [21, 51]) then it provides invariance to initialization scale since this scaling constant vanishes during BatchNorm transformation. However, the proposed standardization layer (see Section 3) is perceived not as a traditional normalization operation, but as a theoretically required part of the initialization scheme.

A large part of our work is devoted to continual zero-shot learning: a new problem for ZSL agents that is inspired by continual learning ideas [32]. In this way, it is a development of the scenario proposed in [11], but authors there cared only about ZSL performance one task ahead. In [28, 40] authors motivate the use of task descriptors for zero-shot knowledge transfer, but in our work we consider class descriptors instead. We defined CZSL as a generalized version of ZSL which allows us to naturally employ all the existing ZSL metrics to our new setup.

3 Normalization in Zero-Shot Learning

3.1 Notation

A usual zero-shot learning setup considers an access to the following datasets: dataset of seen images with their labels $D^{\text{s}} = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{N_s}$; dataset of unseen images with their labels $D^{\text{u}} = \{\boldsymbol{x}_i^u, y_i^u\}_{i=1}^{N_u}$; set of class descriptors for seen classes $A^s = \{\boldsymbol{a}_i\}_{i=1}^{C_s}$; set of class descriptors for unseen classes $A^u = \{\boldsymbol{a}_i\}_{i=1}^{C_u}$. Here N_s, N_u, C_s, C_u are amount of seen images, amount of unseen images, amount of seen classes and amount of unseen classes respectively. Dataset D^s is split into two non-intersecting parts: D^s_{tr} and D^s_{ts} — training seen and testing seen datasets. During training we have an access only to D^s_{tr} and A^s, and after it is done we evaluate the model on D^s_{ts} and D^u with an additional access to A^u. So the main goal for zero-shot learning is, given a dataset of seen classes, to build such a model that would have a good performance both on seen and unseen ones [49].

A typical embedding-based approach includes *attribute embedder* $F_{\boldsymbol{\theta}} : \boldsymbol{a}_c \to \boldsymbol{f}_c \in \mathcal{Z}$ where $\boldsymbol{a}_c$ is a vector of attributes for class c and $\mathcal{Z}$ is the feature space for images, which is usually obtained by a fixed encoder $E : \mathcal{X} \to \mathcal{Z}$ [49]. The goal of embedding-based methods is to optimize a cross-entropy loss between the true classes and the predicted ones, where the logit $\hat{y}_c$ for class c is computed as:

$$\hat{y}_c = \frac{\boldsymbol{f}_c^\top \boldsymbol{z}}{\|\boldsymbol{f}_c\| \|\boldsymbol{z}\|}. \tag{1}$$

Transformation F_θ is usually fairly simple [38] and in many cases even linear [43, 14] and it is the training procedure and different regularization schemes that carry the main load.

The goal of a good initialization scheme is to hold the signal inside a model from severe fluctuations. A zero-shot learning classifier can be seen as a "normal" one, but with a fixed body and a dynamic head, i.e. parameters for the output projection matrix are computed from class attributes by $F_{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{y}} = W_{\boldsymbol{\theta}} E(\boldsymbol{x}), \tag{2}$$

where, with a slight abuse of notation, $W_{\boldsymbol{\theta}} = F_{\boldsymbol{\theta}}(A)$ for matrix of class attributes A. This parameter matrix $W_{\boldsymbol{\theta}}$ changes on each iteration which makes it especially important to make the signal not to fluctuate too much.

For a traditional classifier, the output matrix is initialized in such a way that the variance is preserved either for a forward pass (fan-in mode) or for a backward pass (fan-out mode). And it will be good for us to have such properties for ZSL models as well.

3.2 Understanding "normalize + scale" trick

One of the most popular "tricks" among zero-shot learning practitioners is the normalization of vectors $\boldsymbol{z}$ and $\boldsymbol{f}_c$ to a unit norm followed by additional scaling afterwards [38, 53]. In other words, a logit for class c is computed as:

$$\hat{y}_c = \left(\gamma \cdot \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}\right)^\top \left(\gamma \cdot \frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right), \tag{3}$$

where γ is a hyperparameter that is usually set in the range $[5, 10]$ and sometimes optimized with the rest of the weights [38]. It is not hard to see that this is equivalent to using the previously mentioned cosine similarity followed by a large scaling by γ^2:

$$\hat{y}_c = \gamma^2 \frac{\boldsymbol{z}^\top \boldsymbol{f}_c}{\|\boldsymbol{z}\| \|\boldsymbol{f}_c\|} \tag{4}$$

But why do we need it?

Statement 1 (informal). *"Normalize+scale" trick forces the variance for $\hat{y}_c$ to be approximately equal to*:

$$\text{Var}\left[\hat{y}_c\right] \approx \gamma^4 \frac{d_z}{(d_z - 2)^2}, \tag{5}$$

where d_z is the dimensionality of the feature space (see Appendix A for the assumptions, derivation and the empirical study). This formula demonstrates 2 things:

1. If one uses cosine similarity without scaling, then the variance for $\hat{y}_c$ will be extremely low (especially for large d_z); our model will always output almost uniform distribution and the training would stale due to vanishing gradients out of the start. So the community came up with this "scaling" trick not being rigorously sure of why it works.
2. We cannot influence the variance of $\hat{y}_c$ by changing the variance of weights $W_{\boldsymbol{\theta}}$ since it just does not depend on it.

In our experiments we found that the results are very sensitive to the value of γ and usually a practitioner has to perform an extensive search to find the optimal value. But our formula suggests another strategy of picking it: one can obtain any desired variance $\nu = \text{Var}\left[\hat{y}_c\right]$ by setting γ to:

$$\gamma = \left(\frac{\nu \cdot (d_z - 2)^2}{d_z}\right)^{\frac{1}{4}} \tag{6}$$

For example, for $\text{Var}\left[\hat{y}_c\right] = \nu = 1$ and $d_z = 2048$ (dimensionality of ResNet-101 feature space) we obtain $\gamma \approx 6.78$, which falls right in the middle of $[5, 10]$ — a usual search region in ZSL implementations for the optimal value for γ. The above consideration not only gives a theoretical understanding of the trick, which we believe is important on its own right, but also allows to speed up the search by either picking the predicted "optimal" value for γ or by searching in its vicinity.

3.3 Understanding attributes normalization trick

In the previous subsection it was shown that "normalize+scale" trick makes the variances of $\hat{y}_c$ be independent from variance of weights, features and attributes. This can create an impression that it does not matter how we initialize the weights — normalization would undo any fluctuations. However it is not true, because it is still important how the signal flows "under the hood", i.e. for an unnormalized and unscaled logit value $\tilde{y}_c = \boldsymbol{z}^\top \boldsymbol{f}_c$. Another very common trick among ZSL researchers is the normalization of attribute vectors to a unit norm, i.e. replacing $\boldsymbol{a}$ with $\boldsymbol{a}/\|\boldsymbol{a}\|$ [8]. In this subsection we provide some theoretical underpinnings of its importance.

Let's first consider a linear case for F_θ, i.e. $F_\theta = V \in \mathbb{R}^{d_z \times d_a}$ and $W_\theta = VA$ for matrix of class attributes $A \in \mathbb{R}^{d_a \times K}$, where d_a is the attributes dimensionality and K is the number of classes. Usually, to derive an initialization scheme people use 3 strong assumptions for any random vector that is encountered [20, 25, 6]: 1) coordinates have zero-mean 2) coordinates are independent from each other; and 3) the covariance matrix is of the form $\sigma^2 I$.

But in ZSL world, these assumptions are safe to assume only for feature vector $\boldsymbol{z}$ but not for attribute vector $\boldsymbol{a}$, because for the commonly used datasets an attribute vector does not have zero mean, does not have the same variance across dimensions and, of course, different dimensions are not independent from each other (see Appendix B). That motivates us to derive the variance for $\hat{y}_c$ without relying on them.

Statement 2 (informal). *Attributes normalization trick forces the variance of pre-logit value $\tilde{y}_c$ to have the variance* (see Appendix B for the formal statement and the proof):

$$\text{Var}\left[\tilde{y}_c\right] = d_z \cdot \text{Var}\left[z_i\right] \cdot \text{Var}\left[V_{ij}\right] \cdot \mathop{\mathbb{E}}_{\boldsymbol{a}}\left[\|\boldsymbol{a}\|_2^2\right] \tag{7}$$

So now if one wants to preserve the variance from $\boldsymbol{z}$ to $\tilde{\boldsymbol{y}}$, they obtain the following formula for the required variance of V_{ij}:

$$\text{Var}\left[\tilde{y}_c\right] = \text{Var}\left[z_i\right] \Longrightarrow \text{Var}\left[V_{ij}\right] = \frac{1}{d_z \cdot \mathop{\mathbb{E}}_{\boldsymbol{a}}\left[\|\boldsymbol{a}\|_2^2\right]} \tag{8}$$

This formula for variance is identical to Xavier fan-out initialization (we consider a linear transformation for F_θ here), but with the additional scaling by $1/\mathop{\mathbb{E}}_{a}\left[\|\boldsymbol{a}\|_2^2\right]$. And when one uses "attributes normalization" trick, they effectively remove this additional requirement for initialization, since after applying this trick we have $\mathop{\mathbb{E}}_{a}\left[\|\boldsymbol{a}\|_2^2\right] = 1$.

3.4 Normalization in deeper models

What happens when $F_{\boldsymbol{\theta}}$ is not linear? Imagine that $F_{\boldsymbol{\theta}} = V \circ H_{\boldsymbol{\varphi}}$, i.e. a transformation $H_{\boldsymbol{\varphi}}$ followed by a linear operator V. Let $\boldsymbol{h} = H_{\boldsymbol{\varphi}}(\boldsymbol{a})$ be the output of $H_{\boldsymbol{\varphi}}$.

The analysis of this case is equivalent to the analysis of a linear one but now we use $\boldsymbol{h}$ everywhere instead of $\boldsymbol{a}$. In a word, we would like to initialize the matrix V s.t.:

$$\text{Var}\left[V_{ij}\right] = \frac{1}{d_z \cdot \mathop{\mathbb{E}}_{\boldsymbol{h}}\left[\|\boldsymbol{h}\|_2^2\right]} \tag{9}$$

This initialization is now dependent on the magnitude of $\boldsymbol{h}$, so normalizing attributes to a unit norm will not help us anymore to preserve the variance. To initialize the weights of V using this formula one would need to use a two-step initializition: first initializing $H_{\boldsymbol{\varphi}}$, then computing $\mathop{\mathbb{E}}_{\boldsymbol{h}}\left[\|\boldsymbol{h}\|_2^2\right]$ and then initializing V. This is tiresome and not reliable since $\mathop{\mathbb{E}}_{\boldsymbol{h}}\left[\|\boldsymbol{h}\|_2^2\right]$ changes on each iteration, so we propose a more elegant solution: to perform a dynamic standardization procedure for $\boldsymbol{h}$. In other words, on each iteration we normalize a batch of hidden states $\boldsymbol{h}_1, ..., \boldsymbol{h}_K$ by subtracting its mean and dividing by its standard deviation. This is equivalent to batch normalization layer [27], but without any training parameters. In other words, we insert aforementioned standardization layer S between V and $H_{\boldsymbol{\varphi}}$:

$$F_{\boldsymbol{\theta}} = V \circ S \circ H_{\boldsymbol{\varphi}} \tag{10}$$

This does not add any additional parameters and has imperceptible computational overhead. At test time, we use statistics accumulated during training, just like batch norm does.

Layer S forces $\boldsymbol{h}$ to have zero mean and unit standard variance, which allows us to have a constant value for the expectation of its squared norm (see details in Appendix C):

$$\mathop{\mathbb{E}}_{\boldsymbol{h}}\left[\|\boldsymbol{h}\|_2^2\right] = d_h, \tag{11}$$

where d_h is the dimensionality of $\boldsymbol{h}$. This produces a more concise formula for initializing V:

$$\text{Var}\left[V_{ij}\right] = \frac{1}{d_z d_h}. \tag{12}$$

To summarize, our proposed method incorporates both these ideas: standardization procedure (10) and initialization of the output matrix using the variance formula (12).

Statement 3 (informal). *If one uses proper variance value for initialization* (12) *together with the standardization layer* (10) *then the variance between $\boldsymbol{z}$ and $\tilde{y}$ for deep attribute embedder $F_{\boldsymbol{\theta}}$ is preserved.*

4 Continual Zero-Shot Learning

In continual learning (CL), a model is being trained on a sequence of tasks that arrive one by one. Each task is defined by a dataset $D^t = \{\boldsymbol{x}_i^t, y_i^t\}_{i=1}^{N_t}$ of size N_t. The goal of the model is to master all the tasks sequentially in such a way that at each task t it has good performance both on the current task and all the previously observed ones. In this section we develop the ideas of [11] and formulate a Continual Zero-Shot Learning (CZSL) problem.

Like in CL, CZSL also assumes a sequence of tasks, but now each task is a zero-shot learning problem. This means that apart from D^t we also receive a set of the corresponding class descriptions A^t for each task t. In this way, traditional zero-shot learning can be seen as a special case of CZSL with just two tasks. In [11], authors evaluate their zero-shot models on each task individually, without considering the joint classification space. For example, to measure a model's performance on unseen, they look only one step ahead, which may gives a limited picture of the model's quality. Instead, we borrow ideas from Generalized ZSL [10, 49], and propose to measure the performance on all the seen and all the unseen data for each task. More formally, we have the following datasets:

$$D^{\leq t} = \bigcup_{r=1}^{t} D^r \qquad D^{>t} = \bigcup_{r=t+1}^{T} D^r \qquad A^{\leq t} = \bigcup_{r=1}^{t} A^r \qquad A^{>t} = \bigcup_{r=t+1}^{T} A^r \tag{13}$$

which are the dataset of all seen data, dataset of all unseen data, dataset of seen class attributes and the dataset of unseen class attributes respectively. We will use subscripts "tr"/"ts" to differentiate between train/test data like we did in Section 3.

Our metrics are based on *generalized accuracy* (GA) [10, 49, 15] for ZSL. "Traditional" seen (unseen) accuracy computation discards unseen (seen) classes from the prediction space, thus making the task easier, since the model has fewer classes to be distracted with. For generalized accuracy, we always consider the joint space of both seen and unseen. This gives three metrics: GZSL-S (generalized accuracy on seen), GZSL-U (generalized accuracy on unseen) and GZSL-H (a harmonic mean between GZSL-S and GZSL-U).

Our metrics for CZSL use ZSL metrics under the hood. Namely, we propose the following evaluation measures:

- *Mean Seen Accuracy (mSA).* We compute GZSL-S after tasks $t = 1, .., T$ and take the average:

$$\text{mSA}(F) = \frac{1}{T} \sum_{t=1}^{T} \text{GZSL-S}(F, D_{\text{ts}}^{\leq t}, A^{\leq t}) \tag{14}$$

- *Mean Unseen Accuracy (mUA).* We compute GZSL-U after tasks $t = 1, ..., T-1$ (we do not compute it after task T since $D^{>T} = \varnothing$) and take the average:

$$\text{mUA}(F) = \frac{1}{T-1} \sum_{t=1}^{T-1} \text{GZSL-U}(F, D_{\text{ts}}^{>t}, A^{>t}) \tag{15}$$

- *Mean Harmonic Seen/Unseen Accuracy (mH).* We compute GZSL-H after tasks $t = 1, ..., T-1$ and take the average:

$$\text{mH}(F) = \frac{1}{T-1} \sum_{t=1}^{T-1} \text{GZSL-H}(F, D_{\text{ts}}^{\leq t}, D_{\text{ts}}^{>t}, A) \tag{16}$$

- *Mean Area Under Seen/Unseen Curve (mAUC).* We compute AUSUC [10] after tasks $t = 1, ..., T-1$ and take the average:

$$\text{mAUC}(F) = \frac{1}{T-1} \sum_{t=1}^{T-1} \text{AUSUC}(F, D_{\text{ts}}^{\leq t}, D_{\text{ts}}^{>t}, A) \tag{17}$$

- *Mean Joint Accuracy (mJA).* On each task t we compute the generalized accuracy on all the test data we have for the entire problem.

$$\text{mJA}(F) = \frac{1}{T} \sum_{t=1}^{T} \text{ACC}(F, D_{\text{ts}}, A) \tag{18}$$

  This metric allows us to understand how far behind a model is from the traditional supervised classifiers. A perfect model would be able to generalize on all the unseen classes from the very first task and maintain the performance on par with normal classifiers.

These metrics are a natural extension of the corresponding ZSL metrics and help to evaluate ZSL capabilities of a model. Apart from them, we also use a popular forgetting measure [40, 11] to see if the model is capable of preserving its previously acquired knowledge.

5 Experiments

5.1 Model

In all the experiments we use the same model architecture for our attribute embedder $F_{\boldsymbol{\theta}}$ as in [38]. For ZSL experiments, we used an ImageNet-pretrained ResNet-101 features provided by [49] to be comparable with the other methods. For CZSL experiments, we used ResNet-18 model as an image encoder $E_{\boldsymbol{\varphi}}$ which we train jointly with $F_{\boldsymbol{\theta}}$. Specifically, it is a fully-connected neural network with a single hidden layer of dimensionality d_h. In [38] authors used $d_h = 1600$ for all the datasets, but in our experiments we found it beneficial to decrease the number of hidden units for smaller-scale datasets like CUB [46] or aPY [16]. For the proposed initialization scheme, we also insert the standardization layer as described in Section 3. We compare to 3 baseline initializations: Kaiming fan-in and fan-out inits and Xavier init [20]. Hyperparameter details for each dataset are described in Appendix D and Appendix E.

Table 1: GZSL Seen/Unseen Harmonic mean for ZSL experiments. "AN" stands for "attributes normalization", "NS" — for "normalize+scale" trick.

	SUN	CUB	AwA1	AwA2	aPY
DEM [54]	25.6	29.2	47.3	45.1	19.4
LATEM [48]	19.5	24.0	13.3	20.0	0.2
ALE [36]	26.3	34.4	27.5	23.9	8.7
DEVISE [18]	20.9	32.8	22.4	27.8	9.2
SJE [37]	19.8	33.6	19.6	14.4	6.9
ESZSL [43]	15.8	21.0	12.1	11.0	4.6
SYNC [7]	13.4	19.8	16.2	18.0	13.3
SAE [33]	11.8	13.6	3.5	2.2	0.9
GFZSL [45]	0.0	0.0	3.5	4.8	0.0
RelatNet [44]	-	47.0	46.7	45.3	-
SP-AEN [12]	30.3	46.6	-	37.1	22.6
PSR [2]	26.7	33.9	-	32.3	21.4
GAZSL [55]	26.7	-	-	15.4	24.0
cycle-(U)WGAN [17]	24.4	-	-	19.2	23.6
DCN [39]	30.2	38.7	-	39.1	23.9
f-CLSWGAN [50]	39.4	**49.7**	59.6	17.6	21.4
CIZSL [13]	27.8	-	-	24.6	26.2
CVC-ZSL [38]	39.3	47.5	69.1	66.7	**39.0**
Xavier	33.0 (± 0.6)	46.4 (± 0.9)	57.1 (± 1.2)	51.2 (± 2.3)	16.5 (± 2.1)
Kaiming fan-in	31.5 (± 0.8)	46.7 (± 0.8)	54.0 (± 2.2)	52.1 (± 1.8)	17.3 (± 2.3)
Kaiming fan-out	32.2 (± 0.8)	46.6 (± 0.8)	54.3 (± 1.4)	52.7 (± 2.0)	18.3 (± 1.5)
Ours	**41.7 (± 0.6)**	**49.7 (± 0.7)**	**69.3 (± 0.9)**	**68.0 (± 1.2)**	21.4 (± 2.3)
w/o NS	14.6 (± 0.5)	21.0 (± 2.3)	27.4 (± 14.7)	38.6 (± 4.2)	4.7 (±3.2)
w/o AN	41.2 (± 0.3)	47.3 (± 0.7)	68.9 (± 1.7)	63.0 (± 1.7)	16.2 (± 8.9)
w/o NS, w/o AN	17.3 (± 1.1)	12.7 (± 0.8)	33.5 (± 4.8)	32.1 (± 4.0)	0.9 (± 1.7)

5.2 ZSL experiments

For our ZSL experiments we use five standard datasets: SUN [42], CUB [46], AwA1, AwA2 [49] and aPY [16]. We use the extracted ResNet-101-features and the proposed splits provided by [49] and refer a reader to [49], Table 1 for detailed dataset statistics. To perform cross-validation we first allocate 10% of seen classes for a *validation unseen* data (for AwA1 and AwA2 we allocated 15% since there are only 40 seen classes). Then we allocate 10% out of the remaining 85% of the data for *validation seen* data. This means that in total we allocate $\sim$ 30% of all the seen data to perform validation. Since our experiments are very fast to run we perform cross-validation, allocating validation data randomly for different runs.

We found it beneficial to train the model for different number of epochs and different learning rates for different datasets. All the training details can be found in Appendix D or the provided source code. For small datasets like CUB, SUN and aPY we use label smoothing to regularize the model (we also apply it for the baseline methods as well).

We evaluate the model on the corresponding test datasets using 5 metrics: ZSL-accuracy (i.e. accuracy on unseen), GZSL seen accuracy (GZSL-S), GZSL unseen accuracy (GZSL-U), GZSL-S/GZSL-U harmonic mean (GZSL-H) and AUSUC [10]. The main metric usually considered by practitioners for evaluation is GZSL-H so report it for our method, our baseline initializations and other reference baselines in Table 1. For ZSL accuracy, GZSL-S, GZSL-U and GZSL-AUSUC we refer a reader to Appendix D.

What is especially attractive about our method is its training speed. Since it is just a 2-layer MLP optimized without any bells and whistles on a relatively small datasets, the training is much faster compared to the modern sophisticated ZSL approaches. We use official open source implementations of [38] and [13] to measure their corresponding optimization duration for different datasets. We picked these two methods because they demonstrate state-of-the-art results and belong to two different families of ZSL methods: [38] is an embedding-based and [13] is generative-based. We benchmark

Table 2: Training time for the selected methods.

	SUN	CUB	AwA1	AwA2	aPY
CIZSL [13]	3 hours	2 hours	1.5 hours	1.5 hours	1 hour
CVC-ZSL [38]	4 hours	4 hours	1.5 hours	1.5 hours	1.5 hours
Ours	**1 minute**	**30 seconds**	**40 seconds**	**1.5 min**	**30 seconds**

Table 3: Continual Zero-Shot Learning results

	SUN				CUB			
	mAUC	mH	mJA	Forgetting	mAUC	mH	mJA	Forgetting
Kaiming fan-in	1.8 ± 0.2	8.9 ± 0.7	9.4 ± 0.5	0.04 ± 0.01	3.3 ± 0.8	12.9 ± 1.8	10.5 ± 1.9	0.11 ± 0.03
Kaiming fan-out	1.6 ± 0.3	8.6 ± 1.2	9.0 ± 0.9	$\mathbf{0.03} \pm \mathbf{0.01}$	3.5 ± 0.7	13.5 ± 1.4	11.3 ± 1.5	0.11 ± 0.03
Xavier	1.7 ± 0.3	8.9 ± 1.3	9.3 ± 0.8	0.04 ± 0.01	3.1 ± 0.4	12.4 ± 1.0	10.8 ± 1.3	0.11 ± 0.03
Ours	$\mathbf{2.4} \pm \mathbf{0.3}$	$\mathbf{10.5} \pm \mathbf{0.9}$	$\mathbf{11.1} \pm \mathbf{0.7}$	$\mathbf{0.03} \pm \mathbf{0.01}$	$\mathbf{3.6} \pm \mathbf{0.8}$	$\mathbf{13.2} \pm \mathbf{1.5}$	$\mathbf{11.8} \pm \mathbf{1.9}$	$\mathbf{0.02} \pm \mathbf{0.02}$

our approach and include the comparison in Table 2 on NVidia GeForce RTX 2080 Ti. As one can clearly see from the results, our method outperforms existing models in terms of optimization speed by orders of magnitude.

5.3 CZSL experiments

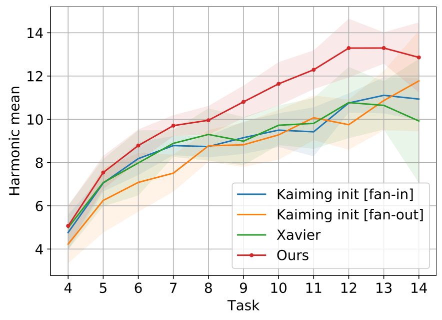
We test our approach in CZSL scenario on two datasets: CUB [46] and SUN [42]. CUB dataset contains 200 classes which we randomly split into 20 tasks, 10 classes per task. SUN dataset contains 717 classes which we randomly split into 15 tasks, the first 3 tasks have 47 classes and the rest of them have 48 classes each (717 classes are difficult to separate evenly). We use official train/test splits for training and testing.

We follow the proposed cross-validation procedure from [11]. Namely, for each run we allocate the first 3 tasks for hyperparameter search, validating on the test data. After that we reinitialize the model from scratch and train on the rest of the tasks. This reduces the effective number of tasks by 3, but provides a more fair way to perform cross-validation [11]. For cross-validation, we use identical search pools of hyperparameter values for all the presented models. We use ResNet-18 model for both datasets as an image encoder E_φ, pretrained on ImageNet. An important difference compared to ZSL setup is that we do not keep image encoders fixed and train them jointly with the model. All the other hyperparameter details and additional evaluation studies can be found in Appendix E or the accompanying source code.

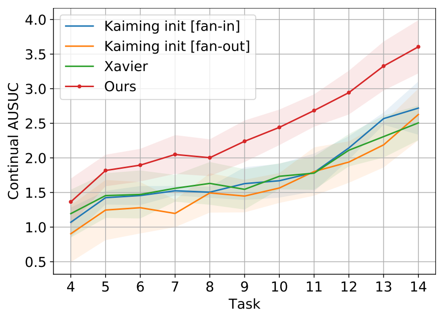
As one can see from Table 3, our proposed initialization scheme significantly outperforms commonly used ones that were supposed to control the variance of a signal. Besides, our approach also enjoys lesser forgetting which is an important property for continually trained models. However, we are still far behind traditional supervised classifiers as one can infer from mJA metric. For example, some state-of-the-art approaches on CUB surpass 90% accuracy [19] which is drastically larger compared to what the considered approaches achieve. Figures 1a and 1b visualizes the learning dynamic of our approach vs baselines. They start approximately equally and the gap increases the more tasks the model learns.

6 Conclusion

In this work, we provided theoretical groundings for two popular tricks used in zero-shot learning: "normalize+scale" trick and attributes normalization. We demonstrated that they aid training by controlling the variance during a forward pass in a linear model and for a deeper one, they are not enough to keep a signal from fluctuations. This motivates us to develop a new initialization scheme that fixes this problem and allows to obtain strong state-of-the-art results on 4 out of 5 commonly used ZSL datasets both in terms of the qualitative performance and training speed. Next, we generalize zero-shot learning into a broader setting of continual zero-shot learning. We propose several metrics



(a) Harmonic Mean for SUN dataset after each task



(b) AUSUC for SUN dataset after each task

Figure 1: CZSL results for SUN dataset

for it and test our ideas in this new scenario. We believe that our work will spur the development of stronger zero-shot systems and motivate their deployment in real-world applications.

Broader Impact

Our methods is a natural step in pushing zero-shot learning further and we believe that zero-shot learning on its own is no more harmful than any other machine learning research direction. It is a humble part of the general development of artificial intelligence that can lead to ethical issues or unfavorable societal consequences only when used improperly.

Our ideas should be helpful to other ZSL practitioners since they shed light on several not-well-understood heuristics and provide ways to build ZSL models more rigorously. Our proposal of continual zero-shot learning should motivate researchers to consider more complex real-world scenarios of training their systems. The proposed method can potentially leverage some biases in a dataset if they would help it to achieve higher performance.

References

[1] Zeynep Akata, Mateusz Malinowski, Mario Fritz, and Bernt Schiele. Multi-cue zero-shot learning with strong supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 59–68, 2016.

[2] Yashas Annadani and Soma Biswas. Preserving semantic relations for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[5] Maxime Bucher, Tuan-Hung VU, Matthieu Cord, and Patrick Pérez. Zero-shot semantic segmentation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 468–479. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8338-zero-shot-semantic-segmentation.pdf.

[6] Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lma24tPB.

[7] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[9] Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[10] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV (2)*, pages 52–68, 2016. URL https://doi.org/10.1007/978-3-319-46475-6_4.

[11] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

[12] Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] Mohamed Elhoseiny and Mohamed Elfeki. Creativity inspired zero-shot learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[14] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[15] Mohamed Elhoseiny, Francesca Babiloni, Rahaf Aljundi, Marcus Rohrbach, Manohar Paluri, and Tinne Tuytelaars. Exploring the challenges towards lifelong fact learning. *CoRR*, abs/1812.10524, 2018. URL http://arxiv.org/abs/1812.10524.

[16] Ali Farhadi, Ian Endres, Derek Hoiem, and David A. Forsyth. Describing objects by their attributes. In *CVPR*, pages 1778–1785. IEEE Computer Society, 2009. ISBN 978-1-4244-3992-8. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2009.html#FarhadiEHF09.

[17] Rafael Felix, Vijay B. G. Kumar, Ian Reid, and Gustavo Carneiro. Multi-modal cycle-consistent generalized zero-shot learning. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[18] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding-model.pdf.

[19] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

[21] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[22] Y. Guo, G. Ding, J. Han, and Y. Gao. Zero-shot learning with transferred samples. *IEEE Transactions on Image Processing*, 26(7):3277–3290, 2017.

[23] Yuchen Guo, Guiguang Ding, Jungong Han, and Yue Gao. Synthesizing samples for zero-shot learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1774–1780, 2017. doi: 10.24963/ijcai.2017/246. URL https://doi.org/10.24963/ijcai.2017/246.

[24] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[26] Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgqN1SYvr.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/ioffe15.html`.

[28] David Isele, Mohammad Rostami, and Eric Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In *International Joint Conferences on Artificial Intelligence*, 2016.

[29] Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rylnK6VtDH`.

[30] Dinesh Jayaraman and Kristen Grauman. Zero-shot recognition with unreliable attributes. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3464–3472. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5290-zero-shot-recognition-with-unreliable-attributes.pdf`.

[31] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[32] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[33] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[34] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[35] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4281–4289, 2018.

[36] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[37] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Kai Li, Martin Renqiang Min, and Yun Fu. Rethinking zero-shot learning: A conditional visual classification perspective. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[39] Shichen Liu, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Generalized zero-shot learning with deep calibration network. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2005–2015. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7471-generalized-zero-shot-learning-with-deep-calibration-network.pdf`.

[40] David Lopez-Paz and Marc Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6467–6476. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7225-gradient-episodic-memory-for-continual-learning.pdf`.

[41] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3650-zero-shot-learning-with-semantic-output-codes.pdf`.

[42] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014.

[43] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2152–2161, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/romera-paredes15.html`.

[44] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[45] Vinay Kumar Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 792–808, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71246-8.

[46] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[47] Yuxin Wu and Kaiming He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[48] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[49] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning - the good, the bad and the ugly. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[50] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[51] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[52] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3914–3923, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1404. URL `https://www.aclweb.org/anthology/D19-1404`.

[53] Ji Zhang, Yannis Kalantidis, Marcus Rohrbach, Manohar Paluri, Ahmed Elgammal, and Mohamed Elhoseiny. Large-scale visual relationship understanding. In *AAAI*, 2019.

[54] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[55] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

A “Normalize + scale” trick

As being discussed in Section 3.2, “normalize+scale” trick changes the logits computation from a usual dot product to the scaled cosine similarity:

$$\hat{y}_c = \langle \boldsymbol{z}, \boldsymbol{f}_c \rangle \Longrightarrow \hat{y}_c = \left\langle \gamma \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma \frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|} \right\rangle, \tag{19}$$

where $\hat{y}_c$ is the logit value for class c; $\boldsymbol{z}$ is an image feature vector; $\boldsymbol{f}_c$ is the attribute embedding for class c:

$$\boldsymbol{f}_c = F_{\boldsymbol{\theta}}(\boldsymbol{a}_c) = V H_{\boldsymbol{\varphi}(\boldsymbol{a}_c)} \tag{20}$$

and γ is the scaling hyperparameter. Let’s denote a penultimate hidden representation of $F_{\boldsymbol{\theta}}$ as $\boldsymbol{h}_c = H_{\boldsymbol{\varphi}}(\boldsymbol{a}_c)$. We note that in case of linear $F_{\boldsymbol{\theta}}$, we have $\boldsymbol{h}_c = \boldsymbol{a}_c$. Let’s also denote the dimensionalities of $\boldsymbol{z}$ and $\boldsymbol{h}_c$ by d_z and d_h.

A.1 Assumptions

To derive the approximate variance formula for $\hat{y}_c$ we will use the following assumptions and approximate identities:

(i) All weights in matrix V:

- are independent from each other and from z_k and $h_{c,i}$ (for all k, i);
- $\mathbb{E}[V_{ij}] = 0$ for all i, j;
- $\text{Var}[V_{ij}] = s_v$ for all i, j.

(ii) There exists $\epsilon > 0$ s.t. $(2+\epsilon)$-th central moment exists for each of $h_{c,1}, ..., h_{c,d_h}$. We require this technical condition to be able apply the central limit theorem for variables with non-equal variances.

(iii) All $h_{c,i}, h_{c,j}$ are independent from each other for $i \neq j$. This is the least realistic assumption from the list, because in case of linear $F_{\boldsymbol{\theta}}$ it would be equivalent to independence of coordinates in attribute vector $\boldsymbol{a}_c$. We are not going to use it in other statements. As we show in Appendix A.3 it works well in practice.

(iv) All $f_{c,i}, f_{c,j}$ are independent between each other. This is also a nasty assumption, but more safe to assume in practice (for example, it is easy to demonstrate that $\text{Cov}[f_{c,i}, f_{c,j}] = 0$ for $i \neq j$). We are going to use it only in normalize+scale approximate variance formula derivation.

(v) $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, s^z I)$. This property is safe to assume since $\boldsymbol{z}$ is usually a hidden representation of a deep neural network and each coordinate is computed as a vector-vector product between independent vectors which results in the normal distribution (see the proof below for $\boldsymbol{f}_c \rightsquigarrow \mathcal{N}(\boldsymbol{0}, s^f I)$).

(vi) For $\boldsymbol{\xi} \in \{\boldsymbol{z}, \boldsymbol{f}_c\}$ we will use the approximations:

$$\mathbb{E}\left[\xi_i \cdot \frac{1}{\|\boldsymbol{\xi}\|_2}\right] \approx \mathbb{E}[\xi_i] \cdot \mathbb{E}\left[\frac{1}{\|\boldsymbol{\xi}\|_2}\right] \quad \text{and} \quad \mathbb{E}\left[\xi_i \xi_j \cdot \frac{1}{\|\boldsymbol{\xi}\|_2^2}\right] \approx \mathbb{E}[\xi_i \xi_j] \cdot \mathbb{E}\left[\frac{1}{\|\boldsymbol{\xi}\|_2^2}\right] \tag{21}$$

This approximation is safe to use if the dimensionality of $\boldsymbol{\xi}$ is large enough (for neural networks it is definitely the case) because the contribution of each individual ξ_i in the norm $\|\boldsymbol{\xi}\|_2$ becomes negligible.

A.2 Formal statement and the proof

Statement 1 (Normalize+scale trick)**.** *If conditions (i)-(vi) hold, then:*

$$\text{Var}[\hat{y}_c] = \text{Var}\left[\left\langle \gamma \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma \frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|} \right\rangle\right] \approx \frac{\gamma^4 d_z}{(d_z - 2)^2} \tag{22}$$

Proof. First of all, we need to show that $f_{c,i} \rightsquigarrow \mathcal{N}(0, s^f)$ for some constant s^f. Since

$$f_{c,i} = \sum_{j=1}^{d_h} V_{i,j} h_{c,j} \tag{23}$$

from assumption (i) we can easily compute its mean:

$$\mathbb{E}\left[f_{c,i}\right] = \mathbb{E}\left[\sum_{j=1}^{d_h} V_{i,j} h_{c,j}\right] \tag{24}$$
$$= \sum_{j=1}^{d_h} \mathbb{E}\left[V_{i,j} h_{c,j}\right] \tag{25}$$
$$= \sum_{j=1}^{d_h} \mathbb{E}\left[V_{i,j}\right] \cdot \mathbb{E}\left[h_{c,j}\right] \tag{26}$$
$$= \sum_{j=1}^{d_h} 0 \cdot \mathbb{E}\left[h_{c,j}\right] \tag{27}$$
$$= 0. \tag{28}$$

and the variance:

$$\text{Var}\left[f_{c,i}\right] = \mathbb{E}\left[f_{c,i}^2\right] - \left(\mathbb{E}\left[f_{c,i}\right]\right)^2 \tag{29}$$
$$= \mathbb{E}\left[f_{c,i}^2\right] \tag{30}$$
$$= \mathbb{E}\left[\left(\sum_{j=1}^{d_h} V_{i,j} h_{c,j}\right)^2\right] \tag{31}$$
$$= \mathbb{E}\left[\sum_{j,k=1}^{d_h} V_{i,j} V_{i,k} h_{c,j} h_{c,k}\right] \tag{32}$$

Using $\mathbb{E}\left[V_{i,j} V_{i,k}\right] = 0$ for $k \neq j$, we have:

$$= \mathbb{E}\left[\sum_{j}^{d_h} V_{i,j}^2 h_{c,j}^2\right] \tag{33}$$
$$= \sum_{j}^{d_h} \mathbb{E}\left[V_{i,j}^2\right] \mathbb{E}\left[h_{c,j}^2\right] \tag{34}$$

Since $s_v = \text{Var}\left[V_{i,j}\right] = \mathbb{E}\left[V_{i,j}^2\right] - \mathbb{E}\left[V_{i,j}\right]^2 = \mathbb{E}\left[V_{i,j}^2\right]$, we have:

$$= \sum_{j}^{d_h} s_v \mathbb{E}\left[h_{c,j}^2\right] \tag{35}$$
$$= s_c \mathbb{E}\left[\sum_{j}^{d_h} h_{c,j}^2\right] \tag{36}$$
$$= s_v \mathbb{E}\left[\|\boldsymbol{h}_c\|_2^2\right] \tag{37}$$
$$= s^f \tag{38}$$
$$\tag{39}$$

Now, from the assumptions (ii) and (iii) we can apply Lyapunov's Central Limit theorem to $f_{c,i}$, which gives us:

$$\frac{1}{\sqrt{s^f}} f_{c,i} \rightsquigarrow \mathcal{N}(0,1) \tag{40}$$

For finite d_h, this allows us say that:

$$f_{c,i} \sim \mathcal{N}(0, s^f) \tag{41}$$

Now note that from (vi) we have:

$$\begin{aligned}
\mathbb{E}\left[\hat{y}_c\right] &= \mathbb{E}\left[\left\langle \gamma\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right\rangle\right] && (42)\\
&= \gamma^2\left\langle \mathbb{E}\left[\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}\right], \mathbb{E}\left[\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right]\right\rangle && (43)\\
&\approx \gamma^2\mathbb{E}\left[\frac{1}{\|\boldsymbol{z}\|}\right]\cdot\mathbb{E}\left[\frac{1}{\|\boldsymbol{f}_c\|}\right]\cdot\left\langle\mathbb{E}\left[\boldsymbol{z}\right],\mathbb{E}\left[\boldsymbol{f}_c\right]\right\rangle && (44)\\
&= \gamma^2\mathbb{E}\left[\frac{1}{\|\boldsymbol{z}\|}\right]\cdot\mathbb{E}\left[\frac{1}{\|\boldsymbol{f}_c\|}\right]\cdot\left\langle\mathbf{0},\mathbf{0}\right\rangle && (45)\\
&= 0 && (46)
\end{aligned}$$

Since $\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0}, s^{\xi})$ for $\boldsymbol{\xi}\in\{\boldsymbol{z},\boldsymbol{f}_c\}$, $\frac{d_\xi}{\|\boldsymbol{\xi}\|_2^2}$ follows scaled inverse chi-squared distribution with inverse variance $\tau = 1/s^{\xi}$, which has a known expression for expectation:

$$\mathbb{E}\left[\frac{d_\xi}{\|\boldsymbol{\xi}\|_2^2}\right] = \frac{\tau d_\xi}{d_\xi - 2} = \frac{d_\xi}{s^{\xi}(d_\xi-2)} \tag{47}$$

Now we are left with using approximation (vi) and plugging in the above expression into the variance formula:

$$\begin{aligned}
\operatorname{Var}\left[\hat{y}_c\right] &= \operatorname{Var}\left[\left\langle \gamma\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right\rangle\right] && (48)\\
&= \mathbb{E}\left[\left\langle \gamma\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right\rangle^2\right] - \mathbb{E}\left[\left\langle \gamma\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right\rangle\right]^2 && (49)\\
&\approx \mathbb{E}\left[\left\langle \gamma\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \gamma\frac{\boldsymbol{f}_c}{\|\boldsymbol{f}_c\|}\right\rangle^2\right] - 0 && (50)\\
&= \gamma^4\mathbb{E}\left[\frac{(\boldsymbol{z}^\top\boldsymbol{f}_c)^2}{\|\boldsymbol{z}\|^2\|\boldsymbol{f}_c\|^2}\right] && (51)\\
&\approx \gamma^4\mathbb{E}\left[(\boldsymbol{z}^\top\boldsymbol{f})^2\right]\cdot\mathbb{E}\left[\frac{1}{d_z}\cdot\frac{d_z}{\|\boldsymbol{z}\|^2}\right]\cdot\mathbb{E}\left[\frac{1}{d_z}\cdot\frac{d_z}{\|\boldsymbol{f}_c\|^2}\right] && (52)\\
&= \frac{\gamma^4}{d_z^2}\cdot\mathop{\mathbb{E}}_{\boldsymbol{f}_c}\left[\boldsymbol{f}_c^\top\mathop{\mathbb{E}}_{\boldsymbol{z}}\left[\boldsymbol{z}\boldsymbol{z}^\top\right]\boldsymbol{f}_c\right]\cdot\frac{d_z}{s^z(d_z-2)}\cdot\frac{d_z}{s^f(d_z-2)} && (53)\\
&= \gamma^4\mathop{\mathbb{E}}_{\boldsymbol{f}_c}\left[\boldsymbol{f}_c^\top s^z I_{d_z}\boldsymbol{f}_c\right]\cdot\frac{1}{s^z s^f(d_z-2)^2} && (54)\\
&= \gamma^4\mathbb{E}\left[\sum_{i=1}^{d_z} f_{c,i}^2\right]\cdot\frac{1}{s^f(d_z-2)^2} && (55)\\
&= \gamma^4 d_z\cdot s^f\cdot\frac{1}{s^f(d_z-2)^2} && (56)\\
&= \frac{\gamma^4 d_z}{(d_z-2)^2} && (57)\\
& && (58)
\end{aligned}$$

□

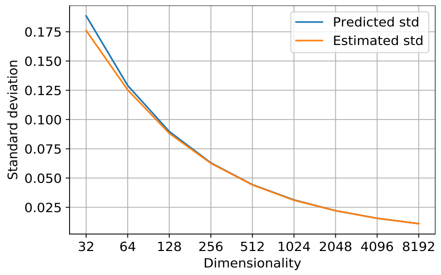
A.3 Empirical validation

In this subsection, we validate the derived approximation empirically. For this, we perform two experiments:

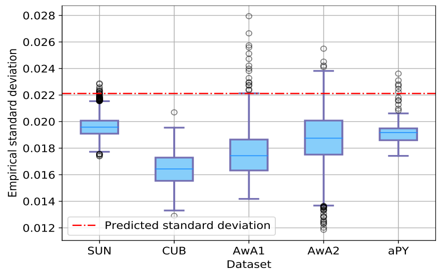
- *Synthetic data*. An experiment on a synthetic data. We sample $\boldsymbol{x}\sim\mathcal{N}(\mathbf{0}, I_d), \boldsymbol{y}\sim\mathcal{N}(\mathbf{0}, I_d)$ for different dimensionalities $d = 32, 64, 128, ..., 8192$ and compute the cosine similarity:

$$z = \left\langle\frac{\gamma\boldsymbol{x}}{\|\boldsymbol{x}\|}, \frac{\gamma\boldsymbol{y}}{\|\boldsymbol{y}\|}\right\rangle \tag{59}$$

  After that, we compute $\operatorname{Var}\left[z\right]$ and average it out across different samples. The result is presented on figure 2a.



(a) Empirical validation of variance formula (22) on synthetic data



(b) Empirical validation of variance formula (22) on real-world data and for a real-world model

Figure 2: Empirical validation of the derived approximation for variance (22)

- *Real data*. We take ImageNet-pretrained ResNet101 features and real class attributes (unnormalized) for SUN, CUB, AwA1, AwA2 and aPY datasets. Then, we initialize a random 2-layer MLP with 512 hidden units, and generate real logits (without scaling). Then we compute mean empirical variance and the corresponding standard deviation over different batches of size 4096. The resulted boxplots are presented on figure 2b.

In both experiments, we computed the logits with $\gamma = 1$. As one can see, even despite our demanding assumptions, our predicted variance formula is accurate for both synthetic and real-world data.

B Attributes normalization

We will use the same notation as in Appendix A. Attributes normalization trick normalizes attributes to the unit l_2-norm:

$$\boldsymbol{a}_c \longmapsto \frac{\boldsymbol{a}_c}{\|\boldsymbol{a}_c\|_2} \tag{60}$$

We will show that it helps to preserve the variance for pre-logit computation when attribute embedder $F_{\boldsymbol{\theta}}$ is linear:

$$\tilde{y}_c = \boldsymbol{z}^\top F(\boldsymbol{a}_c) = \boldsymbol{z}^\top V \boldsymbol{a}_c \tag{61}$$

For a non-linear attribute embedder it is not true, that's why we need the proposed initialization scheme.

B.1 Assumptions

We will need the following assumptions:

(i) Feature vector $\boldsymbol{z}$ has the properties:
  - $\mathbb{E}[\boldsymbol{z}] = 0$;
  - $\text{Var}[z_i] = s^z$ for all $i = 1, ..., d_z$.
  - All z_i are independent from each other and from all $f_{c,j}$.

(ii) Weight matrix V is initialized with Xavier fan-out mode, i.e. $\text{Var}[V_{ij}] = 1/d_z$ and are independent from each other.

Note here, that we do not have any assumptions on $\boldsymbol{a}_c$. This is the core difference from [6] and is an essential condition for ZSL (see Appendix G).

B.2 Formal statement and the proof

Statement 2 (Attributes normalization for a linear embedder)**.** *If assumptions (i)-(ii) are satisfied and* $\|\boldsymbol{a}_c\|_2 = 1$*, then:*

$$\text{Var}[\tilde{y}_c] = \text{Var}[z_i] = s^z \tag{62}$$

Proof. Now, note that:

$$\mathbb{E}[\tilde{y}_c] = \mathbb{E}\left[\boldsymbol{z}^\top V \boldsymbol{a}_c\right] = \mathop{\mathbb{E}}_{V, \boldsymbol{a}_c}\left[\mathop{\mathbb{E}}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top\right] V \boldsymbol{a}_c\right] = \mathbb{E}\left[\boldsymbol{0}^\top V \boldsymbol{a}_c\right] = 0 \tag{63}$$

Then the variance for $\tilde{y}_c$ has the following form:

$$\begin{aligned}
\mathrm{Var}\left[\tilde{y}_c\right] &= \mathbb{E}\left[\tilde{y}_c^2\right] - \mathbb{E}\left[\tilde{y}_c\right]^2 && (64)\\
&= \mathbb{E}\left[\hat{y}_c^2\right] && (65)\\
&= \mathbb{E}\left[(\boldsymbol{z}^\top V \boldsymbol{a}_c)^2\right] && (66)\\
&= \mathbb{E}\left[\boldsymbol{a}_c^\top V^\top \boldsymbol{z}\boldsymbol{z}^\top V \boldsymbol{a}_c\right] && (67)\\
&= \mathop{\mathbb{E}}_{\boldsymbol{a}_c}\left[\mathop{\mathbb{E}}_{V}\left[\mathop{\mathbb{E}}_{\boldsymbol{z}}\left[\boldsymbol{a}_c^\top V^\top \boldsymbol{z}\boldsymbol{z}^\top V \boldsymbol{a}_c\right]\right]\right] && (68)\\
&= \mathop{\mathbb{E}}_{\boldsymbol{a}_c}\left[\boldsymbol{a}_c^\top \mathop{\mathbb{E}}_{V}\left[V^\top \mathop{\mathbb{E}}_{\boldsymbol{z}}\left[\boldsymbol{z}\boldsymbol{z}^\top\right] V\right] \boldsymbol{a}_c\right] && (69)\\
&= s^z \mathop{\mathbb{E}}_{\boldsymbol{a}_c}\left[\boldsymbol{a}_c^\top \mathop{\mathbb{E}}_{V}\left[V^\top V\right] \boldsymbol{a}_c\right] && (70)\\
&= s^z s^v d_z \mathop{\mathbb{E}}_{\boldsymbol{a}_c}\left[\boldsymbol{a}_c^\top \boldsymbol{a}_c\right] && (71)
\end{aligned}$$

since $s^v = 1/d_z$, then:

$$= s^z \mathbb{E}\left[\|\boldsymbol{a}_c\|_2^2\right] \tag{72}$$

since attributes are normalized, i.e. $\|\boldsymbol{a}_c\|_2 = 1$, then:

$$= s^z \tag{73}$$

□

C Normalization for a deep attribute embedder

Using the same derivation as in B, one can show that for a deep attribute embedder:

$$F_{\boldsymbol{\theta}}(\boldsymbol{a}_c) = V \circ H_{\boldsymbol{\varphi}}(\boldsymbol{a}_c) \tag{74}$$

normalizing attributes is not enough to preserve the variance of $\mathrm{Var}\left[\tilde{y}_c\right]$, because

$$\mathrm{Var}\left[\tilde{y}_c\right] = s^z \mathbb{E}\left[\|\boldsymbol{h}_c\|_2^2\right] \tag{75}$$

and $\boldsymbol{h}_c = H_{\boldsymbol{\varphi}}(\boldsymbol{a}_c)$ is not normalized to the unit norm.

To fix the issue, we are going to use two mechanisms:

- A different initialization scheme:
$$\mathrm{Var}\left[V_{ij}\right] = \frac{1}{d_z d_h} \tag{76}$$
- Using the standardization layer before the final projection matrix:
$$S(\boldsymbol{x}) = (\boldsymbol{x} - \hat{\boldsymbol{\mu}}_x) \oslash \hat{\boldsymbol{\sigma}}_x, \tag{77}$$
$\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x$ are the sample mean and variance and $\oslash$ is the element-wise division.

C.1 Assumptions

We'll need the assumption:

(i) Feature vector $\boldsymbol{z}$ has the properties:
   - $\mathbb{E}\left[\boldsymbol{z}\right] = 0$;
   - $\mathrm{Var}\left[z_i\right] = s^z$ for all $i = 1, ..., d_z$.
   - All z_i are independent from each other and from all $f_{c,j}$.

C.2 Formal statement and the proof

Statement 3. *If the assumption (i) is satisfied, an attribute embedder has the form $F_{\boldsymbol{\theta}} = V \circ S \circ H_{\boldsymbol{\varphi}}$ and we initialize output matrix V s.t. $\mathrm{Var}\left[V_{ij}\right] = \frac{1}{d_z d_h}$, then the variance for $\tilde{y}_c$ is preserved:*

$$\mathrm{Var}\left[\tilde{y}_c\right] \approx \mathrm{Var}\left[z_i\right] = s^z \tag{78}$$

Table 4: Hyperparameters for ZSL experiments

	SUN	CUB	AwA1	AwA2	aPY
Batch size	256	256	256	128	256
Learning rate	0.0005	0.005	0.005	0.002	0.001
Number of epochs	100	100	50	100	50
Hidden dimension	1600	256	512	1600	1024
Label smoothing	0.95	0.9	-	-	-
$\mathcal{L}_{\text{ent}}$ weight	-	-	0.001	-	0.01
Gradient clipping value	10	100	10	10	100

Table 5: Additional ZSL metrics for our method

	SUN	CUB	AwA1	AwA2	aPY
GZSL-S	41.1 ($\pm$ 0.2)	51.8 ($\pm$ 1.1)	81.7 ($\pm$ 1.1)	86.4 ($\pm$ 0.5)	69.4 ($\pm$ 2.3)
GZSL-U	43.0 ($\pm$ 1.2)	46.0 ($\pm$ 1.7)	59.6 ($\pm$ 2.0)	58.2 ($\pm$ 2.1)	12.7 ($\pm$ 0.3)
ZSL-U	60.2 ($\pm$ 0.9)	53.2 ($\pm$ 2.1)	70.0 ($\pm$ 2.3)	69.6 ($\pm$ 1.7)	17.4 ($\pm$ 1.5)
AUSUC	25.6 ($\pm$ 0.5)	33.2 ($\pm$ 1.4)	61.4 ($\pm$ 2.0)	62.8 ($\pm$ 1.6)	12.3 ($\pm$ 0.8)

Proof. With some abuse of notation, let $\bar{\boldsymbol{h}}_c = S(\boldsymbol{h}_c)$ (in practice, S receives a batch of $\boldsymbol{h}_c$ instead of a single vector). This leads to:

$$\mathbb{E}\left[\boldsymbol{h}_c\right] = 0 \qquad \text{and} \qquad \text{Var}\left[\boldsymbol{h}_c\right] \approx 1 \tag{79}$$

Using the same reasoning as in Appendix B, one can show that:

$$\text{Var}\left[\tilde{y}_c\right] = d_z \cdot s^z \cdot \text{Var}\left[V_{ij}\right] \cdot \mathbb{E}\left[\|\bar{\boldsymbol{h}}_c\|_2^2\right] = \frac{s^z}{d_h} \cdot \mathbb{E}\left[\|\bar{\boldsymbol{h}}_c\|_2^2\right] \tag{80}$$

So we are left to demonstrate that $\mathbb{E}\left[\|\bar{\boldsymbol{h}}_c\|_2^2\right] = d_h$:

$$\mathbb{E}\left[\|\bar{\boldsymbol{h}}_c\|_2^2\right] = \sum_{i=1}^{d_h} \mathbb{E}\left[\bar{h}_{c,i}^2\right] = \sum_{i=1}^{d_h} \text{Var}\left[\bar{h}_{c,i}\right] \approx d_h \tag{81}$$

□

D ZSL experiments details

In this section, we cover hyperparameter and training details for our ZSL experiments. As being said, we found it beneficial to use label smoothing for several datasets to avoid overfitting on seen classes. Besides, we also found it useful to use entropy regularizer (the same one which is often used in policy gradient methods for exploration) for some datasets:

$$\mathcal{L}_{\text{ent}}(\cdot) = -H(\hat{\boldsymbol{p}}) = \sum_{c=1}^{K} \hat{p}_c \log \hat{p}_c \tag{82}$$

We train the model with Adam optimizer with default β_1 and β_2 hyperparams. The list of hyperparameters is presented in Table 4.

We additionally evaluate our models with other ZSL metrics: GZSL-S, GZSL-U, traditional ZSL accuracy and AUSUC. We report the results in Table 5.

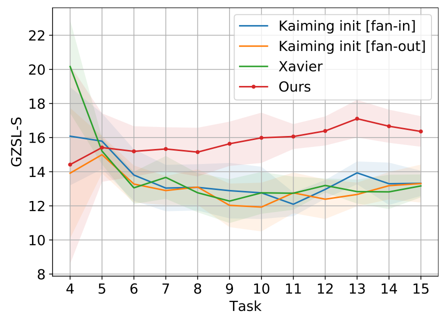
In the ablation study for experiments without attributes normalization (in Table 1) we use traditional standardization (i.e. converting the attributes to zero-mean and unit-variance), since it is a sensible procedure and otherwise they would have too huge magnitude.

E CZSL experiments details

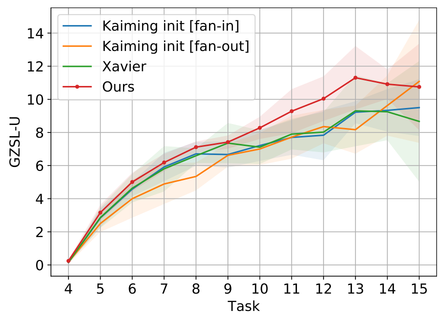
As being said, we use the validation sequence approach [11] to find the best hyperparameters for each method. We allocate the first 3 tasks to perform grid search over a fixed range. After the best hyperparameters have been

Table 6: Hyperparameters range for CZSL experiments

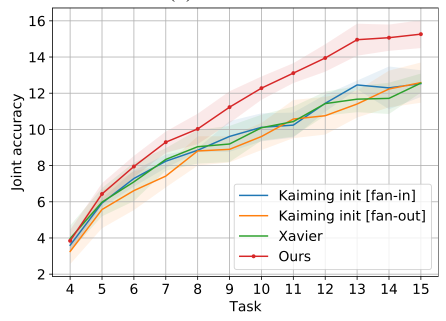
Sampling distribution	uniform, normal
Gradient clipping value	10, 100
Attribute embedder learning rate	0.001, 0.005
Attribute embedder momentum	0.9, 0.95
Image encoder learning rate	0.001, 0.005



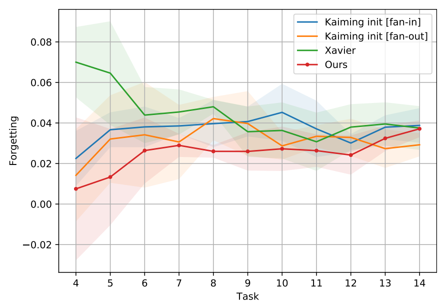
(a) GZSL-S



(b) GZSL-U



(c) Joint Accuracy



(d) Forgetting Measure

Figure 3: Additional CZSL results for SUN dataset

found, we train the model from scratch for the rest of the tasks. The hyperparameter range for CZSL experiments are presented in Table 6 (we use the same range for all the experiments).

We train the model for 5 epochs on each task with SGD optimizer. We also found it beneficial to decrease learning rate after each task by a factor of 0.9. This is equivalent to using step-wise learning rate schedule with the number of epochs equal to the number of epochs per task. As being said, for CZSL experiments we use an ImageNet-pretrained ResNet-18 model as our image encoder. In contrast with ZSL, we do not keep it fixed during training. We present additional metrics for SUN and CUB dataset on figures 3 and 4 respectively.

F Additional experiments and ablations

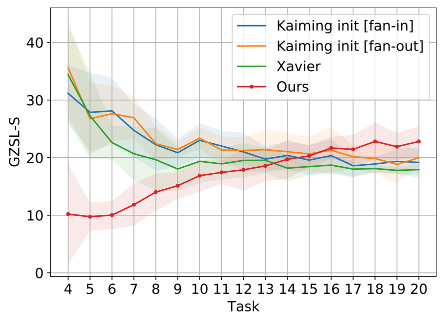
In this section we present additional experiments and ablation studies for our approach (results are presented in Table 7):

- Dynamic normalization. As one can see from formula (9), to achieve the desired variance it would be enough to initialize V s.t. $\text{Var}\left[V_{ij}\right] = 1/d_z$ (equivalent to Xavier fan-out) and use a *dynamic normalization*:

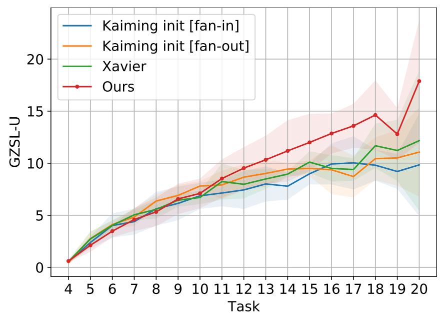
$$\text{DN}(\boldsymbol{h}) = \boldsymbol{h}/\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right] \tag{83}$$

between V and $H_{\boldsymbol{\varphi}}$, i.e. $F_{\boldsymbol{\theta}} = V \circ \text{DN} \circ H_{\boldsymbol{\varphi}}$. Expectation $\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right]$ is computed over a batch on each iteration. A downside of such an approach is that if the dimensionality is large, than a lot of dimensions will get suppressed leading to bad signal propagation.

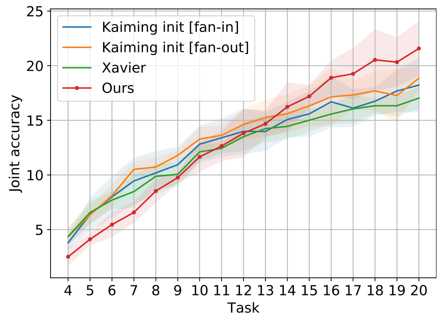
- Traditional initializations + BatchNorm. Though we place our method among initialization techniques, and batch normalization serves a different purpose, it can still amend improper initialization. That's why we test how standard initialization schemes work when equipped with batchnorm.



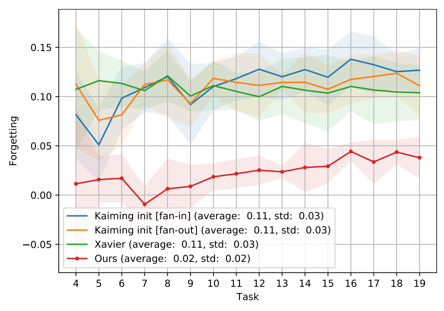
(a) GZSL-S



(b) GZSL-U



(c) Joint Accuracy



(d) Forgetting Measure

Figure 4: Additional CZSL results for CUB dataset

Table 7: GZSL Seen/Unseen Harmonic mean for additional ZSL experiments. "BN" stands for batch normalization, "S" — for standardization layer (10).

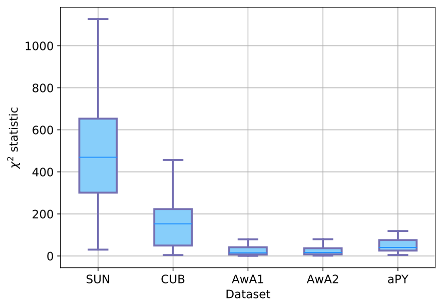
	SUN	CUB	AwA1	AwA2	aPY
Dynamic Normalization	35.0 (± 0.9)	48.0 (± 0.7)	62.93 (± 1.8)	60.0 (± 2.3)	15.7 (± 7.9)
Xavier + BN	40.1 (± 0.7)	49.3 (± 0.6)	50.7 (± 4.7)	67.3 (± 2.7)	20.2 (± 2.0)
Kaiming fan-in + BN	39.0 (± 0.3)	48.4 (± 1.8)	61.6 (± 2.7)	66.6 (± 1.5)	**21.7 (± 2.1)**
Kaiming fan-out + BN	40.1 (± 0.8)	48.9 (± 0.6)	55.3 (± 9.1)	65.9 (± 3.9)	21.2 (± 3.9)
Xavier + S	**41.8 (± 0.6)**	49.4 (± 0.2)	54.7 (± 6.3)	**69.5 (± 1.4)**	19.4 (± 1.0)
Kaiming fan-in + S	41.7 (± 0.4)	49.2 (± 1.1)	57.8 (± 11.0)	67.8 (± 1.6)	19.4 (± 0.5)
Kaiming fan-out + S	41.6 (± 0.3)	**49.7 (± 0.6)**	57.8 (± 13.3)	67.8 (± 1.1)	20.2 (± 0.7)
Ours	41.7 (± 0.6)	**49.7 (± 0.7)**	**69.3 (± 0.9)**	68.0 (± 1.2)	21.4 (± 2.3)
w/o S	35.4 (± 0.5)	47.7 (± 0.7)	61.7 (± 2.8)	54.6 (± 1.0)	18.2 (± 1.0)

- Traditional initializations + Standardization. These experiments ablate the necessity of using the corrected variance (12).
- Proper initialization without standardization layer S. This experiment ablates the necessity of the standardization (10).

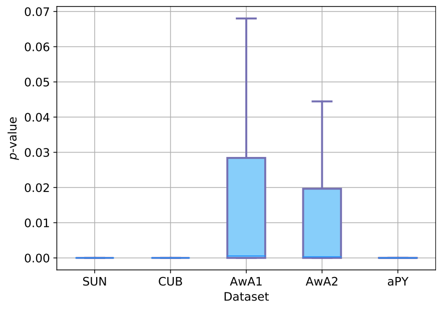
G Why cannot we have independence, zero-mean and same-variance assumptions for attributes in ZSL?

Usually, when deriving an initialization scheme, people assume that their random vectors have zero mean, the same coordinate variance and the coordinates are independent from each other. In the paper, we stated that these are unrealistic assumptions for class attributes in ZSL and in this section elaborate on it.

Attribute values for the common datasets need to be standardized to satisfy zero-mean and unit-variance (or any other same-variance) assumption. But it is not a sensible thing to do, if your data does follow normal distribution



(a) χ^2-statistics of the normality test.



(b) Corresponding p-values

Figure 5: Results of the normality test for class attributes for real-world datasets. Higher values mean that the distribution is further away from a normal one. For a dataset of truly normal random variables, these values are usually in the range $[0, 5]$.. As one can see from 5a, real-world distribution of attributes does not follow a normal one, thus requires more tackling and cannot be easily converted to it.

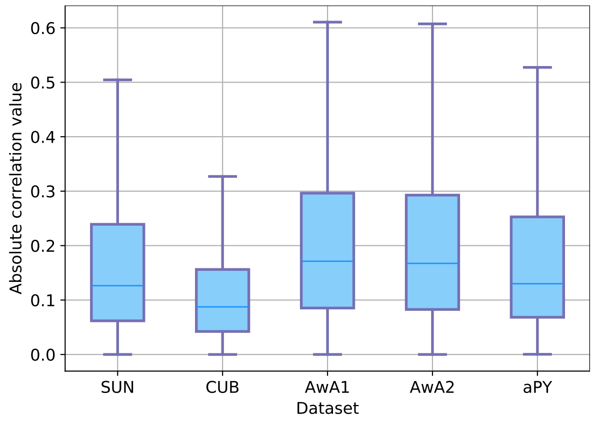


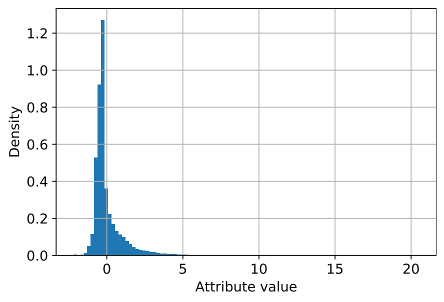
Figure 6: Distribution of mean absolute correlation values between different attribute dimensions. This figure shows that attributes are not independent that's why it would be unreasonable to use such an assumption. If attributes would be independent from each other, that would mean that, for example, that "having black stripes" is independent from "being orange", which tigers would argue not to be a natural assumption to make.

too well (an ablation in Appendix F demonstrates that when one uses standardized attributes, the performance drops terribly). And in our case, it is very far from it.

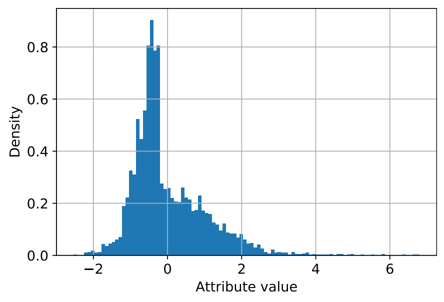
To put it more rigorously, in this section we report two statistical results:

- Results of a normality test based on D'Agostino and Pearson's tests, which comes with scipy python stats library. We run it for each attribute dimension for each dataset and report the distribution of the resulted χ^2-statistics with the corresponding p-values on Figure 5.
- Compute the distribution of absolute values of correlation coefficients between attribute dimensions.

We note, however, that attributes distribution is uni-modal and, in theory, it is possible to transform it into a normal one (by hacking it with log/inverse/sqrt/etc), but such an approach is far from being scalable.



(a) Histogram of attributes for SUN dataset



(b) Histogram of attributes for AwA2 dataset

Figure 7: Histogram of standardized attribute values for SUN and AwA2. These figures demonstrate that the distribution is typically long-tailed and skewed, so it is far from being normal.