In-depth analysis of generative ZSL methods' generalizability

1st Bedetta Alessanro
UniVPM
Ancona, Itay
alessandrobedetta941@gmail.com

2nd Pistagnesi Laura
UniVPM
Ancona, Italy
laura.pistagnesi@gmail.com

***Abstract*—Zero Shot Learning methods are a particular kind of neural network models which aim at carrying out classification tasks on the fly. This means that the goal of the learning phase is the classification of data on which the net was never directly trained.**
In particular, this article is going to be focused on the generative methods, i.e the ones in which we train a portion of the net to generate feature maps based on particular attribute vectors given as input. This attributes can be defined as sets of characteristics capable of identifying each and every class. The very final goal of this paper is to change the main parameters of the data-set as much as possible, and then test whether the results produced by every method remain in a relatively narrow range or not, maintaining the considered approaches comparable under different circumstances.
Eventually, if our tests produce the desired results, at that point we could say that methods are relatively stable to parameters changes and also that given two different ZSL methods A and B, these will be able to be compared without taking under too much consideration the conditions chosen for the training and test phases. (circa 185 parole)

I. Introduction

The development of the Zero Shot Learning methods represented a big step forward in the resolution of classification tasks. This is because they allow us to rely on information which is not directly derived from data-sets, but rather, semantic information known to all of us. We could also say that the whole principle, on which all of these approaches are based on, is quite intuitive; e.g Instead of teaching our nets how to recognize a polar bear, by showing them images representing the latter, we could teach them the concept of bear and the concept of white fur; at that point when the net will see that the polar bear class is identified by the two concepts, described above, put together, it will be able to classify correctly an image containing a polar bear without ever seeing one directly. Now that we have provided a general overview on these approaches let's see in more detail what this article covers. Our work was aimed Our work has been aimed at determining the variance of the results of the ZSL methods when making changes to the data-sets used during training and testing. The observation of behavior and the variability of these approaches will give us information on the methods' stability and also will tell us if a direct comparison of methods makes sense. It means that if the results show that varying the conformation of the used data-sets also vary the results obtained with each method, then we can say that conditions matter while choosing a particular ZSL method and that it makes no sense to say that an A approach is in general better than a B approach.

II. Related Works

- **Generalizability Analysis of Attribute-Based Zero-Shot Learning Methods by Luca Rossi, Roberto Pierdicca, Primo Zingaretti, and Marina Paolanti [4]**:
This paper represented our main starting point for this study. After giving a general overview on the state-of-the-art, by deriving a general taxonomy of the ZSL research field, the article continues with a theoretical formalization of the generalizability problem in ZSL, particularly focused on the attribute-based inductive setting. The second section describes some methods to define the class splits and semantic spaces used to test the methods generalizability. This second section in particular, is the one that represents a preface to our work.
- **Zero-Shot Learning - The Good, the Bad and the Ugly by Yongqin Xian, Bernt Schiele, Zeynep Akata [1]** :
The importance of research on ZSL has meant that countless approaches to this problem have been proposed in recent years. This article deals with taking a step back and looking at the state of the art from a more general point of view, as well as proposing a benchmark for the common evaluation of the various approaches. Using the latter the authors have also concentrated on analyzing a significant number of the state-of-the-art methods in depth, both in the classic zero-shot setting but also in the more realistic generalized zero-shot setting.
- **Feature Generating Networks for Zero-Shot Learning by Yongqin Xian, Tobias Lorenz, Bernt Schiele, Zeynep Akata [2]** :
This paper is focused on a particular generative approach to GZSL and classic ZSL. We know that due to the remarkable training data imbalance between seen and unseen classes, most of existing state-of-the-art approaches fail to achieve the desired results on the GZSL task. To avoid the need for labeled examples of unseen classes, they proposed a novel generative adversarial network (GAN) that synthesizes CNN features conditioned on class-level semantic information, offering a shortcut directly from a semantic descriptor of a class to a class-

conditional feature distribution. The proposed approach, pairing a Wasserstein GAN with a classification loss, is able to generate sufficiently discriminative CNN features to train softmax classifiers or any multimodal embedding method. The experimental results demonstrate a significant boost in accuracy over the state of the art on five challenging datasets – CUB, FLO, SUN, AWA and ImageNet – in both the zero-shot learning and generalized zero-shot learning settings.

- **Latent Embedding Feedback and Discriminative Features for Zero-Shot Classification by Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees G. M. Snoek, Ling Shao [3]** :
ZSL main goal is to classify unseen categories for which no data is available during training. In GZSL the test samples can further belong to seen or unseen categories. The state-of-the-art relies on Generative Adversarial Networks that synthesize unseen class features by leveraging class-specific semantic embeddings. During training, they generate semantically consistent features, but discard this constraint during feature synthesis and classification. This paper proposes to enforce semantic consistency at all stages of (generalized) zero-shot learning: training, feature synthesis and classification. The main additional component is the feedback loop, from a semantic embedding decoder, that iteratively refines the generated features during both the training and feature synthesis stages. The synthesized features together with their corresponding latent embeddings from the decoder are then transformed into discriminative features and utilized during classification to reduce ambiguities among categories. Their experiments on (generalized) zero-shot object have revealed the benefit of semantic consistency and iterative feedback, outperforming existing methods on six zero-shot learning benchmarks.

III. Some basic terminology

- DS : DataSet
- DL : Deep Learning
- ZSL : Zero Shot Learning
- GZSL : Generalized Zero Shot Learning
- GAN : Generative Adversarial Network
- WGAN : Wasserstein Generative Adversarial Network

IV. Evaluated Methods

For our experiments, we considered two ZSL methods (CLSWGAN and TF-VAEGAN) and five splits methods (GCS, CCS, MAS, MCS, PCA). We now briefly describe this architectures and methods.

A. ZSL methods

The basis of both models is the GAN. GAN consists of a Generative network G and a Discriminative network D that compete in a two player minimax game. G tries to fool D producing realistic-looking samples; D tries to figure out whether an image came from the training set (real) or the Generator network (fake), assigning a probability to it. The GAN aims to generate outputs that could be taken by the real distribution of provided training data but are different from the provided data itself.

1) ***CLSWGAN:*** Xian et al. [1] devised a conditional WGAN with a classification loss to synthesize features for the unseen classes. They improved the WGAN integrating the semantic feature into both generator and discriminator. In addition, to improve the performance of the generator so to generate discriminative features, the negative log-likelihood is used to minimize the classification loss over the generated features. The objective function is:

$$\min_G \max_D \mathcal{L}_{WGAN} + \beta \mathcal{L}_{CLS}$$

where $\mathcal{L}_{WGAN}$ is the WGAN loss, $\mathcal{L}_{CLS}$ is the classification loss and β is the weighting factor.

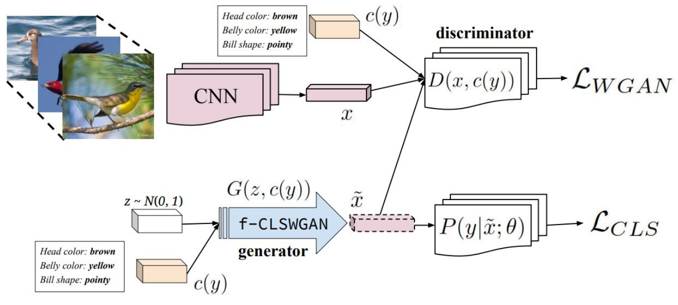
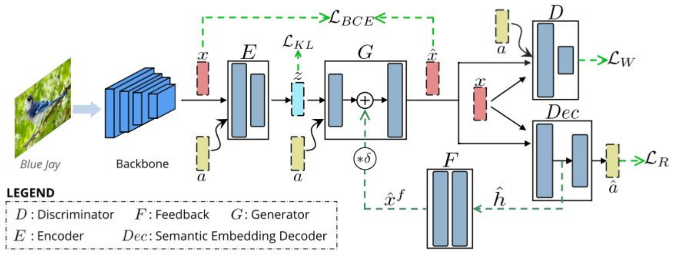


Fig. 1. CLSWGAN architectures

2) ***TF-VAEGAN:*** Naratan et al. [3] developed a TF-VAEGAN ZSL framework consisting of a variational autoencoder (VAE), a generative adversarial network (GAN), a semantic embedding decoder *Dec* and a feedback module F. Visual features *x* of a seen class image, extracted from the backbone network, and the corresponding semantic embeddings *a* are input to the encoder E. The encoder E generates a latent code *z*, which is then input, together with semantic embedding *a*, to the generator G to synthesize features $\widehat{x}$. E and G constitute the VAE. Then the discriminator D learns to distinguish between real features *x* and synthesized features $\widehat{x}$. G and D constitute the GAN. The semantic embedding decoder *Dec* reconstruct the semantic embedding *a* from the generated features $\widehat{x}$. The latent embedding $\widehat{h}$ from *Dec* is input to feedback module F whose output is added to the latent representation of G to refine features. The loss for training TF-VAEGAN is given by:

$$\mathcal{L}_{total} = \mathcal{L}_{vaegan} + \beta \mathcal{L}_R$$

where $\mathcal{L}_{vaegan} = \mathcal{L}_V + \alpha \mathcal{L}_W$ is the sum between VAE loss and WGAN loss, $\mathcal{L}_R$ is the cycle-consistency loss of *Dec* and β and α are hyper-parameter for weighting factors.



TF-VAEGAN architectures

B. *Split methods*

We considered both different classes splits and different attributes splits. For GCS, CCS and MAS we considered also the inverse methods that means seen classes become unseen classes and vice versa.

*1) **Greedy Class Split (GCS):*** The GCS aims to keep as much semantic information as possible among the seen classes so as to ensure that, for example, if we define a zebra as a horse with stripes, we have stripes among the seen classes. So, for each class, this method sums the values of its signature vector and it sorts the classes by these sums in descending order. Finally it selects the first n_S classes as seen classes, and the other n_U as unseen classes.

*2) **Clustered Class Split (CCS):*** The CCS defines the set of seen classes and the set of unseen classes as two clusters. So the CCS aims to minimize the intra-cluster distances and maximize the inter-cluster distances. To define clusters, this method sorts the classes by the sum of their row (or column) values in descending order. The first n_S classes are those with the lowest distances overall, meaning that they form a cluster in the semantic space. Those classes will be the seen classes. The other n_U are far from this cluster in the semantic space, so they will form another cluster and they will be the unseen classes.

*3) **Minimal Attribute Split (MAS):*** The MAS aims to remove unnecessary attributes, i.e., highly correlated attributes. This methods computes correlation between two attributes in a class as the ratio of co-occurrencies of the union over the intersection of this attributes. So it gets rid of the most correlated attributes.

*4) **Minimal Correlation Split (MCS):*** The MCS aims to find a split with low correlated attributes. So this method generates a series of random splits until it gets one with correlation less than a value K.

*5) **Principal Component Analysis (PCA):*** The PCA aims to select the most important features that capture maximum information about a dataset. This methods transforms a set of features in a dataset into a smaller number of features, called principal components, selecting them based on variance.

V. Experimental protocol

During the experiments, all the tests were carried out with the aim of trying to understand which were the factors that contributed more on the classification error. As written in [4], given a dataset *d*, a set of reasonable conditions *k* and a ZSL method *m*, we can divide the overall loss in **method loss** and **semantic loss**.
The first one is the loss due to a possible method underfitting while the second is the error attributable to the conditions considered during the definition of the semantic space. Recent works are, for the most part, focused on trying to make the method loss better; this is because the semantic loss is believed to not be dependent on the chosen method m.
The main goal of our experiments was to test this assumption, which can be represented as the following equation, while also trying to sort out which are the best possible conditions (attribute and class splits) for each method.

$$|SL(k, d) - SL(m, k, d)| < \epsilon$$

A. ***Data-sets***

For our experiments we chose to use a group which included both fine and coarse grained datsets. In particular we trained and tested on the following sets :

- **Animals with attributes (AWA)** : It is a coarse grained set which consists of 30475 images of 50 animals classes with six pre-extracted feature representations for each image. The animals classes are aligned with Osherson's classical class/attribute matrix, thereby providing 85 numeric attribute values for each class.
- **Caltech-UCSD Birds (CUB)** : It is the most widely-used dataset for fine-grained visual categorization task. It contains 11,788 images of 200 subcategories belonging to birds, 5,994 for training and 5,794 for testing. Each class is bonded to a set of 312 binary attributes.
- **Flower Dataset (FLO)** : The FLO contains 102 flower categories. The flowers chosen to be flower commonly occuring in the United Kingdom. Each class consists of between 40 and 258 images and each class is identified by a set of 1024 attributes.

For all three datasets we had a file named res101.mat containing three fields: features, labels and image files. The name came from the fact that the features were extracted using a ResNet101, pre-trained on the ImageNet DS.
Batch-size was set to 64 for all tests.

B. ***Parameters***

For all tests the number of epochs was set to 100 and only the GZSL scenario was considered; for both methods the learning rate was set to 0.00001 and the latent space vector dimension was set to match the attribute vector dimension. In the clswgan net the number of hidden units in the fully connected layers was set to 4096 in the generator and 1024 in the discriminator, while in the tf-vaegan net the two values were both set to 4096.

The different conditions were obtained by changing the following parameters :

- **Manual Seed** : A seed is used to ensure that results are reproducible. In other words, using this parameter makes sure that anyone who re-runs your code will get the exact same outputs. In our case seeds ensure that the data is divided (training/validation/test) the same way every time the code is run
- **Attribute Split** : We can consider the attributes as the axes of our attribute space, capable of identifying every class in our datasets. An attribute split is nothing but a criterion according to which we define these axes. Changing it has a great impact on the final results of the method we are taking under consideration.
- **Class Split** : Gan methods are based on the generation a synthetic data-set which is then used to train the final classifier. Seen classes are the ones used during the GAN training for teaching the generator to translate attribute vectors into feature vectors. Unseen classes, on the other hand, are the labels that are going to be contained in the generated dataset. Given the classes' attribute vectors, choosing which ones are to be considered in each of the two groups is going to have a big influence on the final accuracy.

C. *Performance Metrics*

The performance metric used in every test was the classification accuracy calculated on the final classifier output. As said before we only considered the GZSL setting, that is why the output consists of two values: accuracy on the seen classes, and accuracy on the unseen classes (synthetic dataset).

$$Accuracy\ seen = T_s/N_{tot}$$

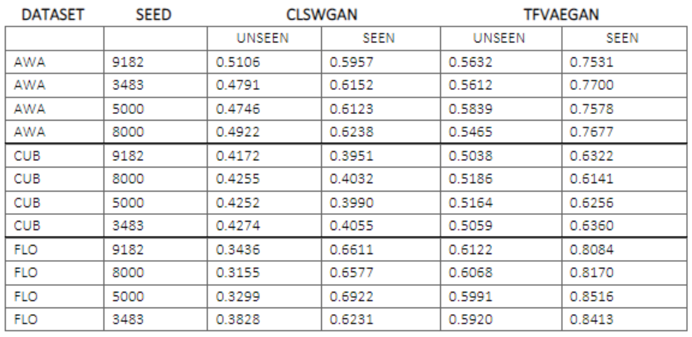
$$Accuracy\ unseen = T_u/N_{tot}$$

T_s *: Number of seen classes samples correctly classified*
T_s *: Number of unseen classes samples correctly classified*
N_{tot} *: Number of samples*

D. *Environment and Tools*

All the code was written in Python and all the implementation of methods, splits and other components was done using the **Pytorch** deep learning library.
After trying to run some tests locally we quickly realized that we needed a lot more processing power, as even with 10 epochs the experiments were taking hours to complete. That's the reason behind our decision to use Google Colab as working environment, which made the test process a lot less painful.

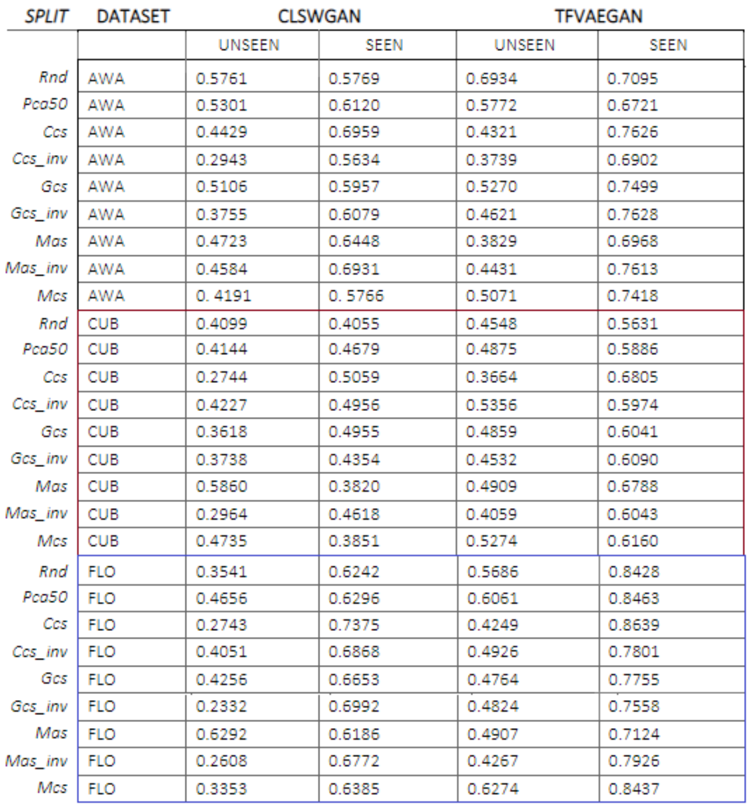
VI. Results



DATASET	SEED	CLSWGAN		TFVAEGAN	
		UNSEEN	SEEN	UNSEEN	SEEN
AWA	9182	0.5106	0.5957	0.5632	0.7531
AWA	3483	0.4791	0.6152	0.5612	0.7700
AWA	5000	0.4746	0.6123	0.5839	0.7578
AWA	8000	0.4922	0.6238	0.5465	0.7677
CUB	9182	0.4172	0.3951	0.5038	0.6322
CUB	8000	0.4255	0.4032	0.5186	0.6141
CUB	5000	0.4252	0.3990	0.5164	0.6256
CUB	3483	0.4274	0.4055	0.5059	0.6360
FLO	9182	0.3436	0.6611	0.6122	0.8084
FLO	8000	0.3155	0.6577	0.6068	0.8170
FLO	5000	0.3299	0.6922	0.5991	0.8516
FLO	3483	0.3828	0.6231	0.5920	0.8413

Test 1 : Original splits, Different seeds

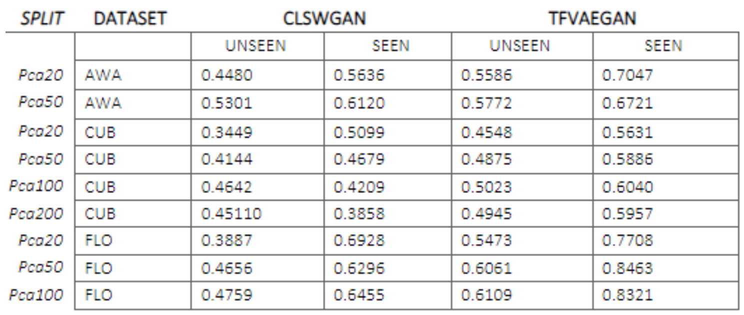
The first tests were aimed to give us a quick overview of the results when no splits were applied to the three data-sets. So what we did was just run the two methods on all data-sets and the only thing we changed during this process was the data-set seed.



SPLIT	DATASET	CLSWGAN		TFVAEGAN	
		UNSEEN	SEEN	UNSEEN	SEEN
Rnd	AWA	0.5761	0.5769	0.6934	0.7095
Pca50	AWA	0.5301	0.6120	0.5772	0.6721
Ccs	AWA	0.4429	0.6959	0.4321	0.7626
Ccs_inv	AWA	0.2943	0.5634	0.3739	0.6902
Gcs	AWA	0.5106	0.5957	0.5270	0.7499
Gcs_inv	AWA	0.3755	0.6079	0.4621	0.7628
Mas	AWA	0.4723	0.6448	0.3829	0.6968
Mas_inv	AWA	0.4584	0.6931	0.4431	0.7613
Mcs	AWA	0. 4191	0. 5766	0.5071	0.7418
Rnd	CUB	0.4099	0.4055	0.4548	0.5631
Pca50	CUB	0.4144	0.4679	0.4875	0.5886
Ccs	CUB	0.2744	0.5059	0.3664	0.6805
Ccs_inv	CUB	0.4227	0.4956	0.5356	0.5974
Gcs	CUB	0.3618	0.4955	0.4859	0.6041
Gcs_inv	CUB	0.3738	0.4354	0.4532	0.6090
Mas	CUB	0.5860	0.3820	0.4909	0.6788
Mas_inv	CUB	0.2964	0.4618	0.4059	0.6043
Mcs	CUB	0.4735	0.3851	0.5274	0.6160
Rnd	FLO	0.3541	0.6242	0.5686	0.8428
Pca50	FLO	0.4656	0.6296	0.6061	0.8463
Ccs	FLO	0.2743	0.7375	0.4249	0.8639
Ccs_inv	FLO	0.4051	0.6868	0.4926	0.7801
Gcs	FLO	0.4256	0.6653	0.4764	0.7755
Gcs_inv	FLO	0.2332	0.6992	0.4824	0.7558
Mas	FLO	0.6292	0.6186	0.4907	0.7124
Mas_inv	FLO	0.2608	0.6772	0.4267	0.7926
Mcs	FLO	0.3353	0.6385	0.6274	0.8437

Test 2 : Different splits, same seed (9182), 100 epochs

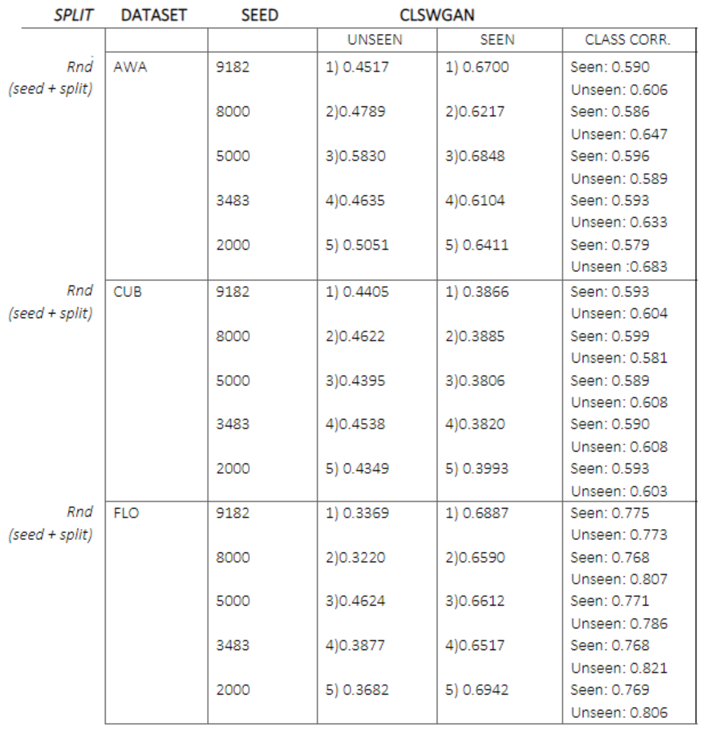
We then proceeded to run the split-generating code on the three data-sets to obtain different attribute spaces. Test 2 illustrates the results of each split; We were able to replicate all the splits for all three data-sets except for the Principal Component Analysis. This is because we wanted to use different numbers of components to see the effect on the accuracy.



SPLIT	DATASET	CLSWGAN		TFVAEGAN	
		UNSEEN	SEEN	UNSEEN	SEEN
Pca20	AWA	0.4480	0.5636	0.5586	0.7047
Pca50	AWA	0.5301	0.6120	0.5772	0.6721
Pca20	CUB	0.3449	0.5099	0.4548	0.5631
Pca50	CUB	0.4144	0.4679	0.4875	0.5886
Pca100	CUB	0.4642	0.4209	0.5023	0.6040
Pca200	CUB	0.45110	0.3858	0.4945	0.5957
Pca20	FLO	0.3887	0.6928	0.5473	0.7708
Pca50	FLO	0.4656	0.6296	0.6061	0.8463
Pca100	FLO	0.4759	0.6455	0.6109	0.8321

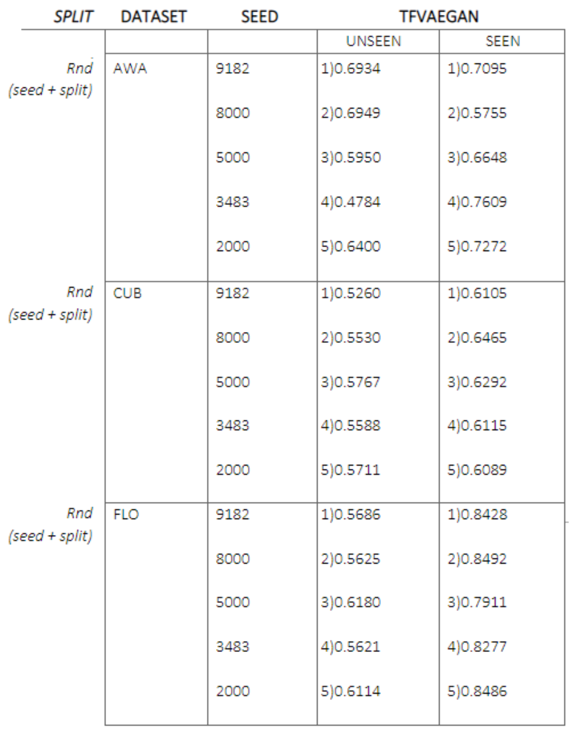
Test 3 : PCA tests with different number of components (20,50,100,200)

This problem was due to the fact that the implementation we used for the pca required to have a final attribute space with a number of dimensions less or equal to the minimum between number of classes and the initial number of attributes; so for the AWA datasets the limit was 50 and we couldn't replicate the tests featuring 100 and 200 components; same problem with the FLO data-set (102 classes) where we were not able to apply the pca 200.



SPLIT	DATASET	SEED	CLSWGAN		
			UNSEEN	SEEN	CLASS CORR.
Rnd (seed + split)	AWA	9182	1) 0.4517	1) 0.6700	Seen: 0.590 Unseen: 0.606
		8000	2)0.4789	2)0.6217	Seen: 0.586 Unseen: 0.647
		5000	3)0.5830	3)0.6848	Seen: 0.596 Unseen: 0.589
		3483	4)0.4635	4)0.6104	Seen: 0.593 Unseen: 0.633
		2000	5) 0.5051	5) 0.6411	Seen: 0.579 Unseen :0.683
Rnd (seed + split)	CUB	9182	1) 0.4405	1) 0.3866	Seen: 0.593 Unseen: 0.604
		8000	2)0.4622	2)0.3885	Seen: 0.599 Unseen: 0.581
		5000	3)0.4395	3)0.3806	Seen: 0.589 Unseen: 0.608
		3483	4)0.4538	4)0.3820	Seen: 0.590 Unseen: 0.608
		2000	5) 0.4349	5) 0.3993	Seen: 0.593 Unseen: 0.603
Rnd (seed + split)	FLO	9182	1) 0.3369	1) 0.6887	Seen: 0.775 Unseen: 0.773
		8000	2)0.3220	2)0.6590	Seen: 0.768 Unseen: 0.807
		5000	3)0.4624	3)0.6612	Seen: 0.771 Unseen: 0.786
		3483	4)0.3877	4)0.6517	Seen: 0.768 Unseen: 0.821
		2000	5) 0.3682	5) 0.6942	Seen: 0.769 Unseen: 0.806

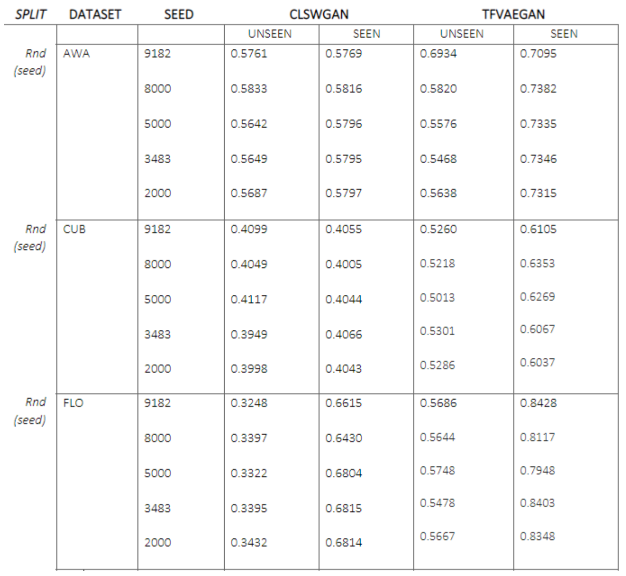
Test 4 : Seed and Random Split contribution to accuracy (CLSWGAN)



SPLIT	DATASET	SEED	TFVAEGAN	
			UNSEEN	SEEN
Rnd (seed + split)	AWA	9182	1)0.6934	1)0.7095
		8000	2)0.6949	2)0.5755
		5000	3)0.5950	3)0.6648
		3483	4)0.4784	4)0.7609
		2000	5)0.6400	5)0.7272
Rnd (seed + split)	CUB	9182	1)0.5260	1)0.6105
		8000	2)0.5530	2)0.6465
		5000	3)0.5767	3)0.6292
		3483	4)0.5588	4)0.6115
		2000	5)0.5711	5)0.6089
Rnd (seed + split)	FLO	9182	1)0.5686	1)0.8428
		8000	2)0.5625	2)0.8492
		5000	3)0.6180	3)0.7911
		3483	4)0.5621	4)0.8277
		2000	5)0.6114	5)0.8486

Test 4 : Seed and Random Split contribution to accuracy (TF-VAEGAN)

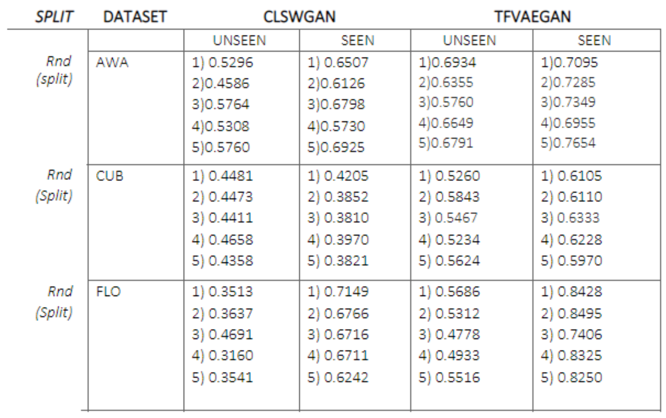
The very next group of tests we focused on was aimed at analyzing how much the different seeds and the different random splits respectively contributed to the final accuracy. So during the first group of tests, seed and random split changed each time. Here one additional information we chose to provide was a correlation measurement done on the different random splits; the correlation was calculated on the attribute vectors of seen and unseen classes. An higher correlation score indicates that the corresponding group of classes is less ethereogeneous.



SPLIT	DATASET	SEED	CLSWGAN		TFVAEGAN	
			UNSEEN	SEEN	UNSEEN	SEEN
Rnd (seed)	AWA	9182	0.5761	0.5769	0.6934	0.7095
		8000	0.5833	0.5816	0.5820	0.7382
		5000	0.5642	0.5796	0.5576	0.7335
		3483	0.5649	0.5795	0.5468	0.7346
		2000	0.5687	0.5797	0.5638	0.7315
Rnd (seed)	CUB	9182	0.4099	0.4055	0.5260	0.6105
		8000	0.4049	0.4005	0.5218	0.6353
		5000	0.4117	0.4044	0.5013	0.6269
		3483	0.3949	0.4066	0.5301	0.6067
		2000	0.3998	0.4043	0.5286	0.6037
Rnd (seed)	FLO	9182	0.3248	0.6615	0.5686	0.8428
		8000	0.3397	0.6430	0.5644	0.8117
		5000	0.3322	0.6804	0.5748	0.7948
		3483	0.3395	0.6815	0.5478	0.8403
		2000	0.3432	0.6814	0.5667	0.8348

Test 5 : Seed contribution to accuracy

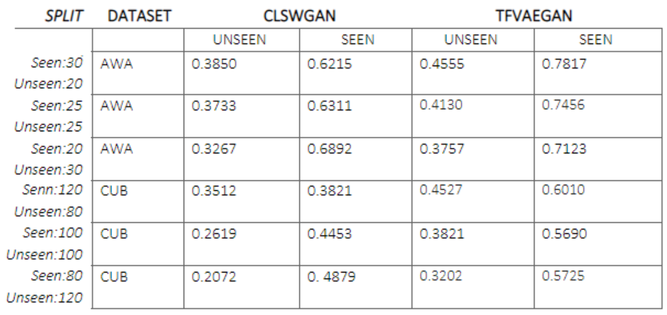
The table above contains the result of the experiments done by changing the random seed value, maintaining the same random split on the attributes.



SPLIT	DATASET	CLSWGAN		TFVAEGAN	
		UNSEEN	SEEN	UNSEEN	SEEN
Rnd (split)	AWA	1) 0.5296 2)0.4586 3)0.5764 4)0.5308 5)0.5760	1) 0.6507 2)0.6126 3)0.6798 4)0.5730 5)0.6925	1)0.6934 2)0.6355 3)0.5760 4)0.6649 5)0.6791	1)0.7095 2)0.7285 3)0.7349 4)0.6955 5)0.7654
Rnd (Split)	CUB	1) 0.4481 2) 0.4473 3) 0.4411 4) 0.4658 5) 0.4358	1) 0.4205 2) 0.3852 3) 0.3810 4) 0.3970 5) 0.3821	1) 0.5260 2) 0.5843 3) 0.5467 4) 0.5234 5) 0.5624	1) 0.6105 2) 0.6110 3) 0.6333 4) 0.6228 5) 0.5970
Rnd (Split)	FLO	1) 0.3513 2) 0.3637 3) 0.4691 4) 0.3160 5) 0.3541	1) 0.7149 2) 0.6766 3) 0.6716 4) 0.6711 5) 0.6242	1) 0.5686 2) 0.5312 3) 0.4778 4) 0.4933 5) 0.5516	1) 0.8428 2) 0.8495 3) 0.7406 4) 0.8325 5) 0.8250

Test 6 : Random Split contribution to accuracy

Test 6 reports the final accuracies obtained by keeping the same random seed (9182) and changing the random slit file each time.



SPLIT	DATASET	CLSWGAN		TFVAEGAN	
		UNSEEN	SEEN	UNSEEN	SEEN
Seen:30 Unseen:20	AWA	0.3850	0.6215	0.4555	0.7817
Seen:25 Unseen:25	AWA	0.3733	0.6311	0.4130	0.7456
Seen:20 Unseen:30	AWA	0.3267	0.6892	0.3757	0.7123
Senn:120 Unseen:80	CUB	0.3512	0.3821	0.4527	0.6010
Seen:100 Unseen:100	CUB	0.2619	0.4453	0.3821	0.5690
Seen:80 Unseen:120	CUB	0.2072	0. 4879	0.3202	0.5725

Test 7 : Testing different cardinalities of seen and unseen classes

The last tests we tried to get something out of were about changing the number of seen and unseen classes to see the effect on the final accuracy. For all the previous experiments the number of seen and unseen classes were respectively : 40-10 for AWA, 150-50 for CUB and 82-20 for FLO. Given these cardinalities, we tried using three different splits for each dataset.

- **AWA (seen-unseen)** : (30-20), (25-25), (20-30)
- **CUB (seen-unseen)** : (120-80), (100-100), (80-120)
- **FLO (seen-unseen)** : (62-40), (51-51), (40-62)

VII. Discussion

Taking a look at the results yielded by our experiments we can see that we already have an answer to our most important question. So do the conditions under which a certain method is executed influence the final result to the point that the approach can not be considered generalizable?

Yes, they do.
This statement can be proven true by just taking a look at the results given by the different class/attribute splits. It is really easy to observe a quite relevant variance over the different accuracy scores.
This implies that the $|SL(k,d) - SL(m,k,d)| < \epsilon$ formula cannot be considered valid.

Now, zooming back and focusing on both methods' results, given the previous observation, we could conclude by saying that the two approaches cannot be compared, since they are not generalizable. Despite this, it's quite safe to say that, considering our tests, the TF-VAEGAN approach is often the winner between the two. We can't exclude the possible existence of a particular scenario which would result in the CLS-WGAN prevailing over the TF-VAEGAN, thus the observation cannot be considered a rule.

Now looking at the results produced by the different class and attribute splits, it's really difficult to visualize patterns in data; this is due to the lack of time that forced us to run a low number of experiments for each split.

At a first look we could say that the MAS split seems to be working good especially when paired with a fine grained dataset (CUB, FLO). This might be connected to the fact that, having an higher number of classes with a corresponding higher number of attributes, a cut to the dimensionality of the semantic space could translate to a reduction on the effect of Curse Of Dimensionality.

Another possible observation regards the PCA splits. It looks like the attribute number reduction benefits the final accuracy as long as the cut is not too extreme. For instance, the PCA cutting down the semantic space to a 20-dimensional space has always turned out to be too simplifying and thus the worst one. Another thing we can see is that when we go from 20 to 200 components in the CUB dataset, the first 2 increments (20 to 50 and 50 to 100) are the ones that have the higher accuracy increase; on the other hand going from 100 to 200 components doesn't seem to have the same effect. This could mean that there is a threshold, which we could experimentally determine, that marks the border of effectiveness for the PCA split.

Another behavior notable of attention regards the CCS split and its inversion. Is easy to observe how the two have, in all three occasions, the opposite result. When one does good the other one is bad and vice versa. Moreover, for the AWA dataset the regular CCS has good accuracy, but for the fine grained datasets the one having better results is the inverted CCS. This could be related to the fact that when dealing with fine grained datasets, having a training set of relatively similar classes could cause a lack of attribute diversity during the training phase. While when we are operating on a coarse grained dataset, we have less attributes to consider and so having seen classes close to each other, is not as much of a problem.

During the experiments described in Test 6 we also noticed that, as expected, changing the latent space seed doesn't have the same impact as changing the random split on classes.

Throughout all the experiments where we altered the number of seen and unseen classes (maintaining the total unchanged), which results are reported in Test 7, we saw something particular happen. Decreasing the number of seen classes (and increasing the number of unseen) we can see that the final accuracy on training classes gets better. This is simply caused by the fact that the final classifier has to choose between a lower number of seen classes, thus the task is easier.
On the other hand we can see a constant decrease on the unseen accuracy. As said above this can be attributed to the lower variety of attributes that the net has during training; having a lower number of labels to learn from, i.e having less training samples it's just normal that the net accuracy on the never seen classes will be worse.

VIII. Conclusions

We proved during this relatively short paper that ZSL methods are not generalizable as they yield different degrees of accuracy under different conditions. We also demonstrated that it is useless to compare two approaches without talking about the conditions under which they are applied. We are satisfied with the accuracies obtained using particular splits; some of them showed encouraging results which could be studied during future works in more depth. We really hope that we will see more future researches regarding Zero Shot Learning and that some of this methods will be featured in real life applications.

References

[1] Yongqin Xian, Bernt Schiele, Zeynep Akata "Zero-Shot Learning - The Good, the Bad and the Ugly" 2017
[2] Yongqin Xian, Tobias Lorenz, Bernt Schiele, Zeynep Akata "Feature Generating Networks for Zero-Shot Learning" 2018
[3] Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees G. M. Snoek, Ling Shao "Latent Embedding Feedback and Discriminative Features for Zero-Shot Classification" 2020
[4] Luca Rossi, Roberto Pierdicca, Primo Zingaretti, and Marina Paolanti "Generalizability Analysis of Attribute-Based Zero-Shot Learning Methods"