Alesi Mattia
Proietti Marco

Zero-Shot Learning

Progetto CV&DL 2022/2023

Zero-Shot Learning

Consists of teaching model to recognize “unseen classes” never seen during training (as opposed to “seen classes”).

Notations

Seen Classes

Classes for which we have labeled images during training.

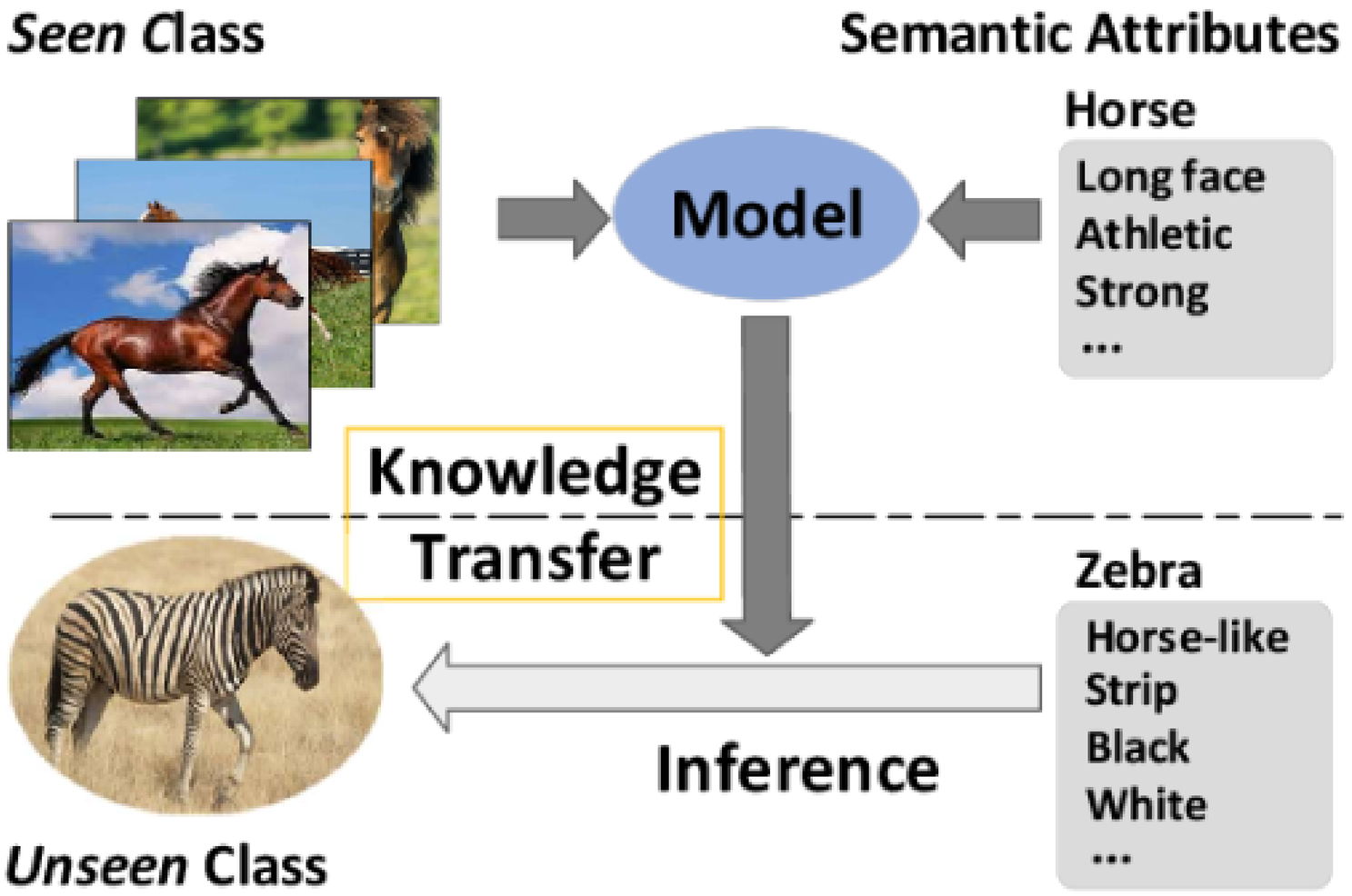
Unseen Classes

Classes for which labeled images are not present during the training phase

Auxiliary informations

Descriptions/semantic attributes/word embeddings for both seen and unseen classes.
This information acts as a bridge between seen and unseen classes.
Each class (seen or unseen) can be identified by its attribute vector

Zero-Shot Learning - Outline



Types of Zero-Shot Learning

Conventional ZSL

•The images to be recognized at test time belong **only** to **unseen classes**.
•This setting is practically less useful as, in realistic scenarios, the assumption that the images at test time will come only from unseen classes is difficult to guarantee.

Generalised ZSL

•The images to be recognized at test time may belong to **seen or unseen** classes.
•This setting is practically **more useful**/realistic and much more challenging than the conventional setting.
•The model has been trained only on seen class images and therefore its predictions are **biased** towards seen classes: many unseen class images can be wrongly classified into seen classes at test time.

Attribute-based ZSL

A semantic encoding is a **mapping** between the attribute space and the classes.
It's defined by the human operator.

The auxiliary information is stored as a one hot encoded vector consisting of 5 elements — e.g. tail, beak, feather, whiskers, furry.
Each element of the vector is an **attribute**, its value is 1 if the attributes describe that class, and 0 elsewhere.

tail	1
beak	0
feather	0
whiskers	1
furry	1

Cat attribute vector

Models

Theese models represents the state-of-the-art and they are all comprensive of the same modules.

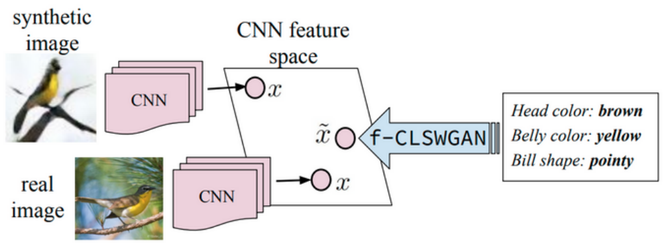
In the first step, we use a generative method (e.g. GAN), conditioned on an attribute vector, to generate a synthetic dataset with the unseen classes.
In the second step, we train a normal classifier on the complete (original + synthetic) dataset.

It can be possibile to work on every aspect of these model to run experiments but in our study we focused on the classification network architecture and on the manipulation of attributes.

Models in details

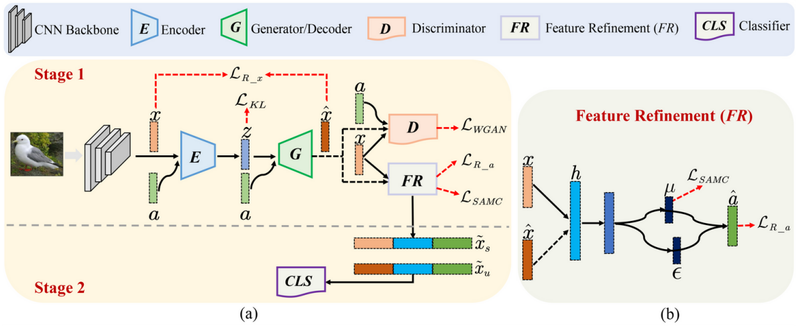
CLSWGAN

It's 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.



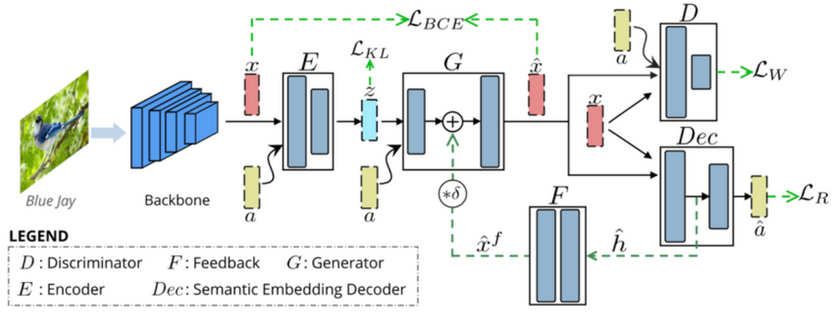
FREE

FREE employs a Feature Refinement (FR) module that incorporates **semantic to visual** mapping into a unified generative model to refine the visual features of seen and unseen class samples.



TFVAEGAN

It combines a GAN with a **variatonal autoencoder**.
The innovative part is that there is the utilization of a semantic embedding decoder at **all stages**: training, feature synthesis and classification



Dataset

AWA

Contains 30'475 images of animals

- 40 seen classes
- 10 unseen classes

FLO

Contains 10'320 images of flowers

- 82 seen classes
- 20 unseen classes

CUB

Contains 11'788 images of birds

- 150 seen classes
- 50 unseen classes

SUN

Contains 14'340 images of scenes

- 645 seen classes
- 72 unseen classes

Experiments

Parameters Tuning

- Reduce the size of the hidden layer
- Increase batch size
- Test new pooling options (mean, max, first)

Conditioning choise of classes

- Test by only taking the "second most similar" class

Attribute Manipulation

- Exaggerate the attributes from the most similar class, e.g. by multiplying them by a constant
- Find "salient" attributes, i.e. rare attributes that particularly distinguish a class from the others, especially with regard to similar classes

Results

What we achieved?

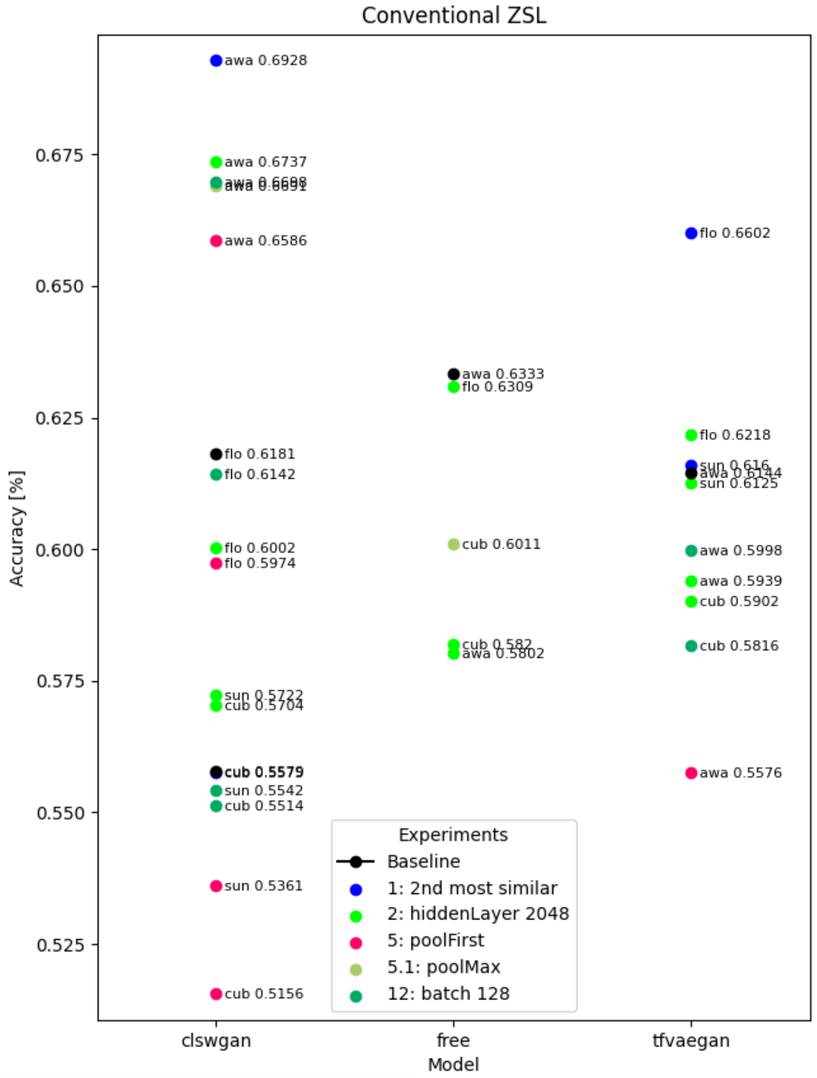
Tabular data of all the accuracy results for each experiment ran

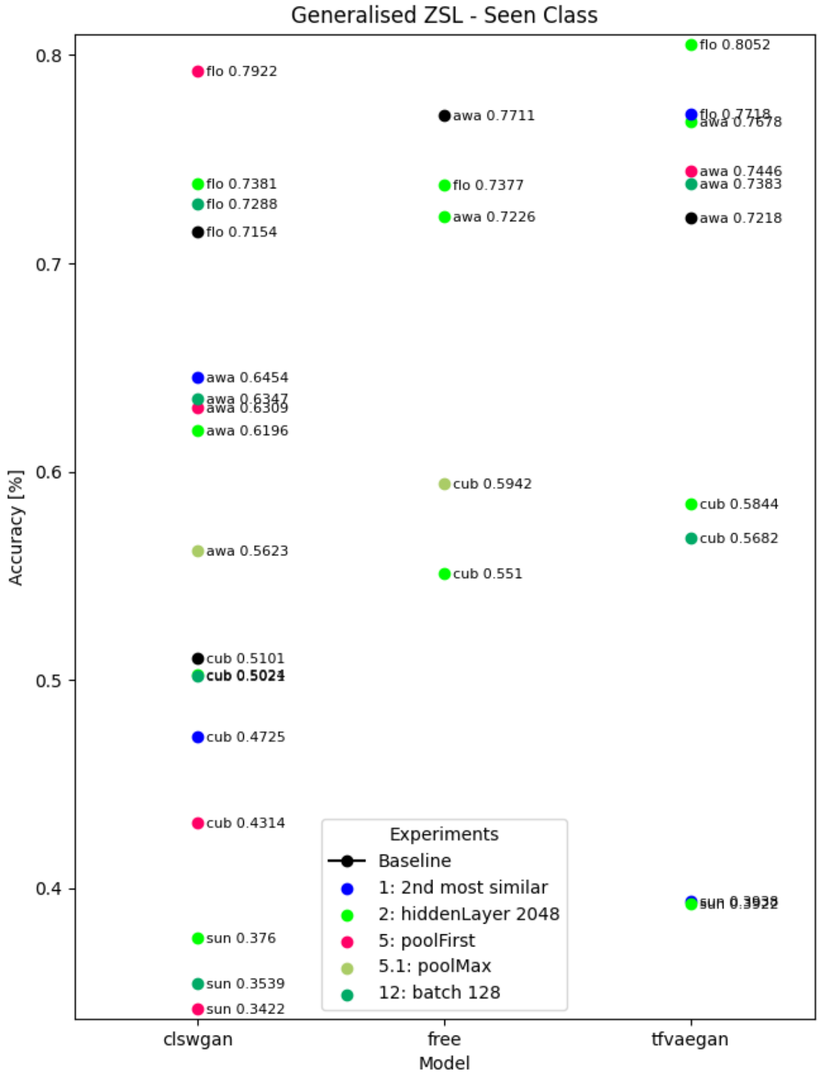
Model	Experiment	AWA2			CUB			FLO			SUN		
		cZSL	Seen	Unseen	cZSL	Seen	Unseen	cZSL	Seen	Unseen	cZSL	Seen	Unseen
clswgan	Baseline				0.5579	0.5101	0.4077	0.6181	0.7154	0.5088			
	n.1	0.6928	0.6454	0.5003	0.5575	0.4725	0.4317						
	n.2	0.6737	0.6196	0.536	0.5704	0.5024	0.4565	0.6002	0.7381	0.4958	0.5722	0.376	0.4076
	n.3	0.6586	0.6309	0.4345	0.5156	0.4314	0.4052	0.5974	0.7922	0.4559	0.5361	0.3422	0.3868
	n.4	0.6691	0.5623	0.4723									
	n.5	0.6698	0.6347	0.51	0.5514	0.5021	0.4135	0.6142	0.7288	0.5265	0.5542	0.3539	0.4125
free	Baseline	0.6333	0.7711	0.5037									
	n.1												
	n.2	0.5802	0.7226	0.4682	0.582	0.551	0.4864	0.6309	0.7377	0.5937			
	n.3												
	n.4				0.6011	0.5942	0.4727						
	n.5												
tfvaegan	Baseline	0.6144	0.7218	0.4565									
	n.1							0.6602	0.7718	0.5904	0.616	0.3938	0.4472
	n.2	0.5939	0.7678	0.4256	0.5902	0.5844	0.4897	0.6218	0.8052	0.5432	0.6125	0.3922	0.441
	n.3	0.5576	0.7446	0.3934									
	n.4												
	n.5	0.5998	0.7383	0.3596	0.5816	0.5682	0.4765						

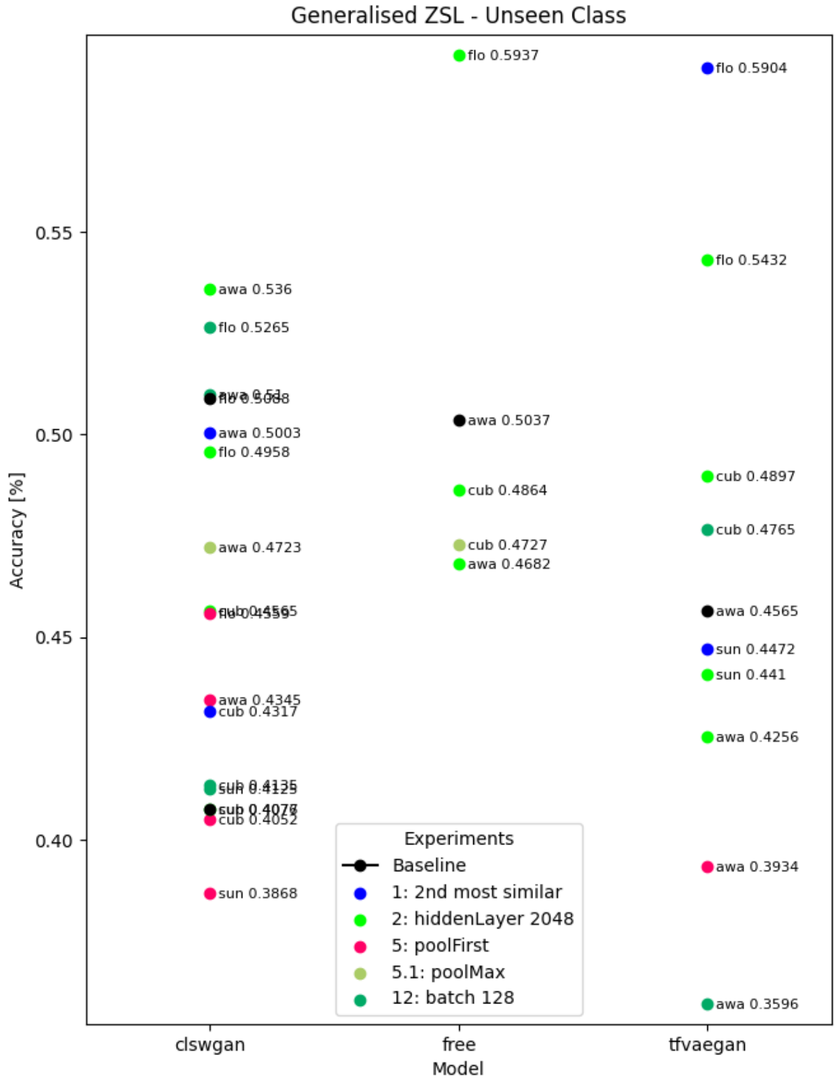
Results

What we achieved?

This visual comparison shows the accuracy results for each experiment ran



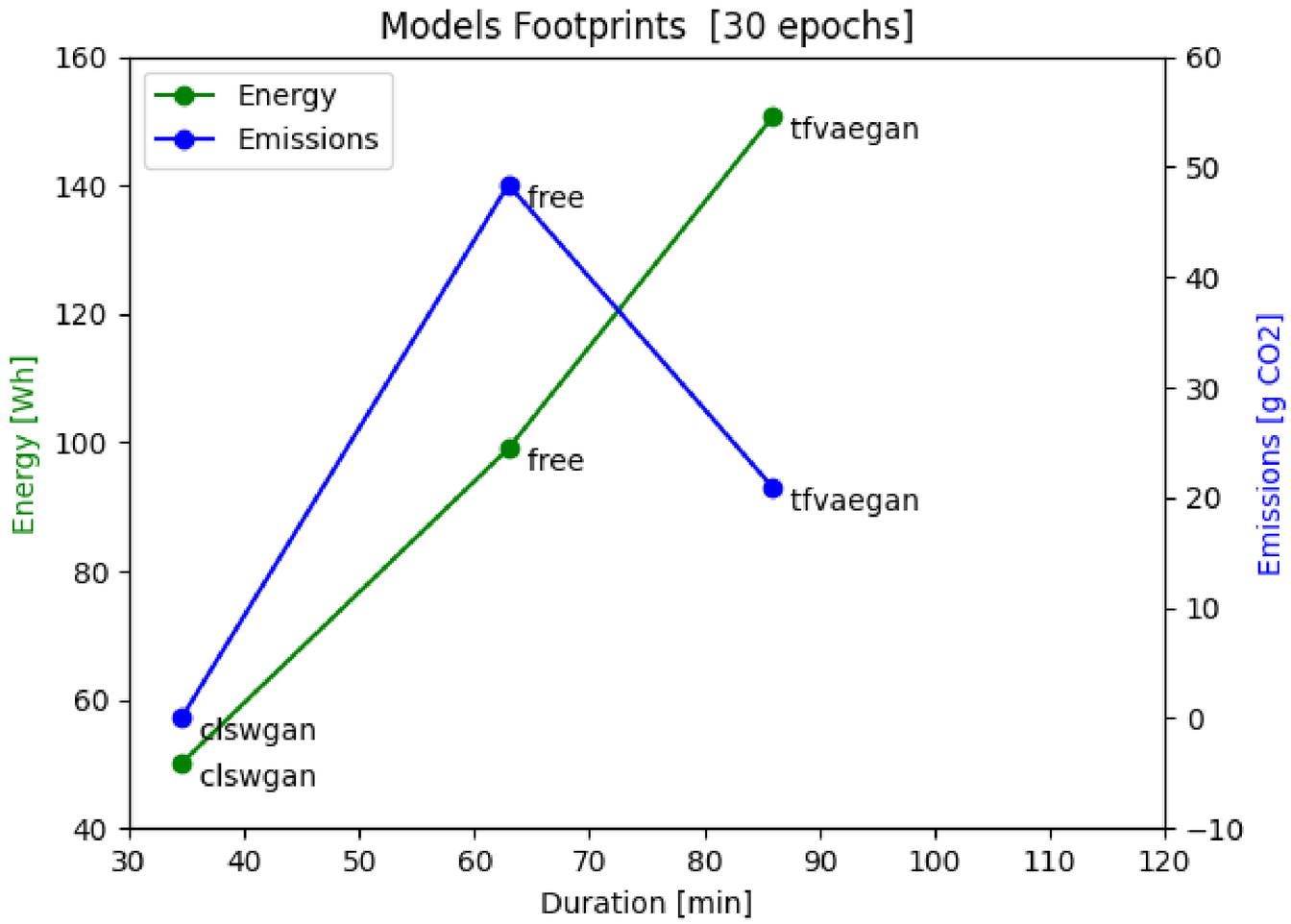




Code Carbon

Lightweight software package that estimates the amount of carbon dioxide (CO2) produced by the computing resources used to execute the code.

We used this tool for some of our experiments to measure the impact on the environment of our researches.



Future developments

Create **linear combinations** of attributes for a better conditioning of the classes

Applying a **feedback vector** to the Generator module

Find **salient attributes** which distinguishes classes, then **condition** the Generator for recognizing more unseen classes.

Thanks for your attention!