Reliability of Zero-Shot Learning

Presenter: Ashraf Mahgoub

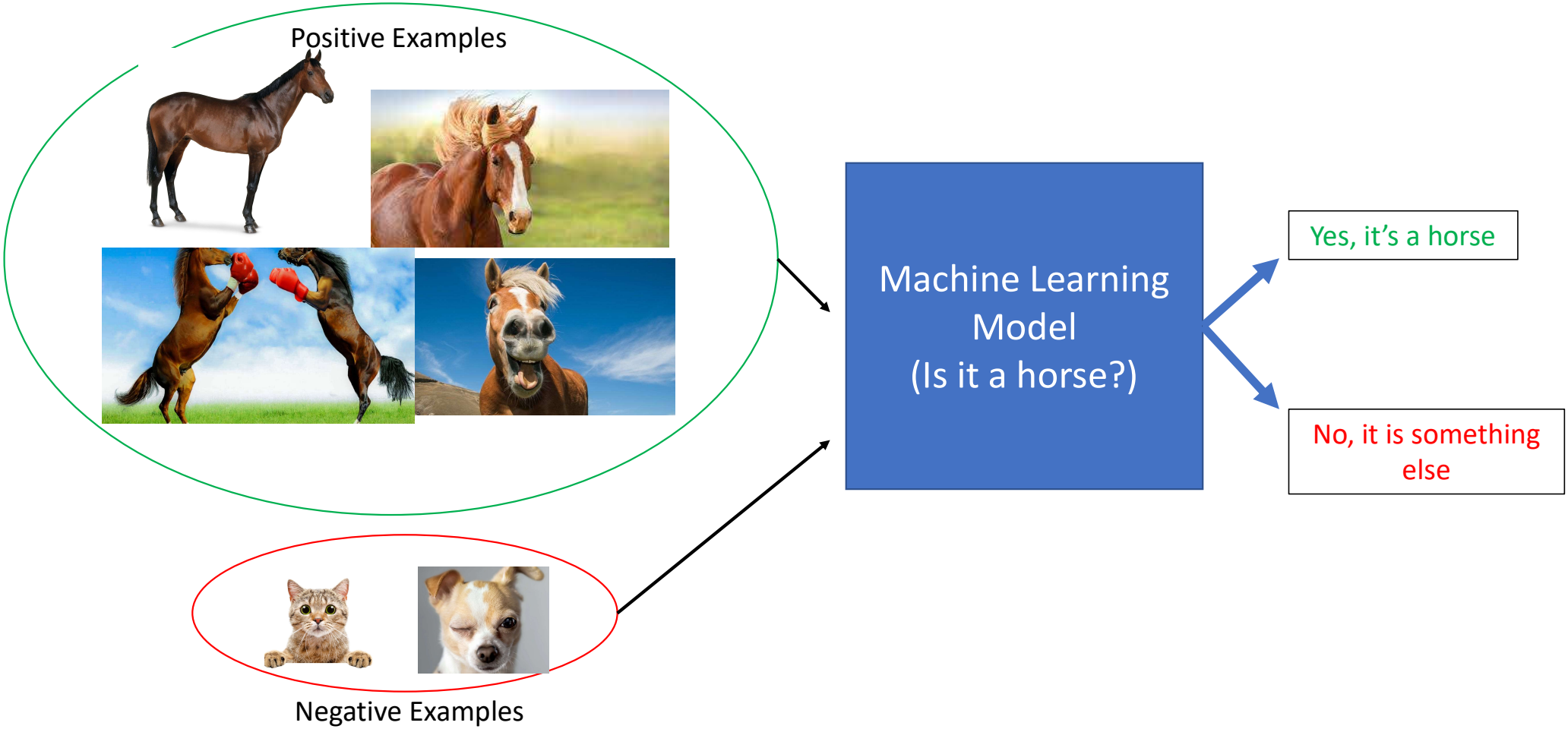
Introduction: Zero-Shot Learning

- Assume we want to build a machine learning model to classify these 8 classes of animals
- In the best case, we need at least one example for each animal
- Typically we will need much more than one example per class

Introduction: Zero-Shot Learning

- Now assume we want to build a machine learning model to classify *all* animals
- There's 1,899,587 described species in the world, so we will need a dataset with roughly 2 million different classes
- Questions:
  - Do we need data for *each class*?
  - What if a new species is discovered over time?
  - How can we include this *new unseen classes* in the model?

Motivation: Zero-Shot Learning



Motivation: Zero-Shot Learning

- Now assume we want the model to recognize if it is a horse, a zebra, or something else?
- Can we do this without collecting data for zebras?

First Classification Step

(Does it look like a horse?)

Yes

No

Second Classification Step

(Does it have black and white stripes?)

Yes

No

Zebra

Horse

Something else

Long slender legs?
Long thick necks?
Deep torso build?
Large elongated heads?

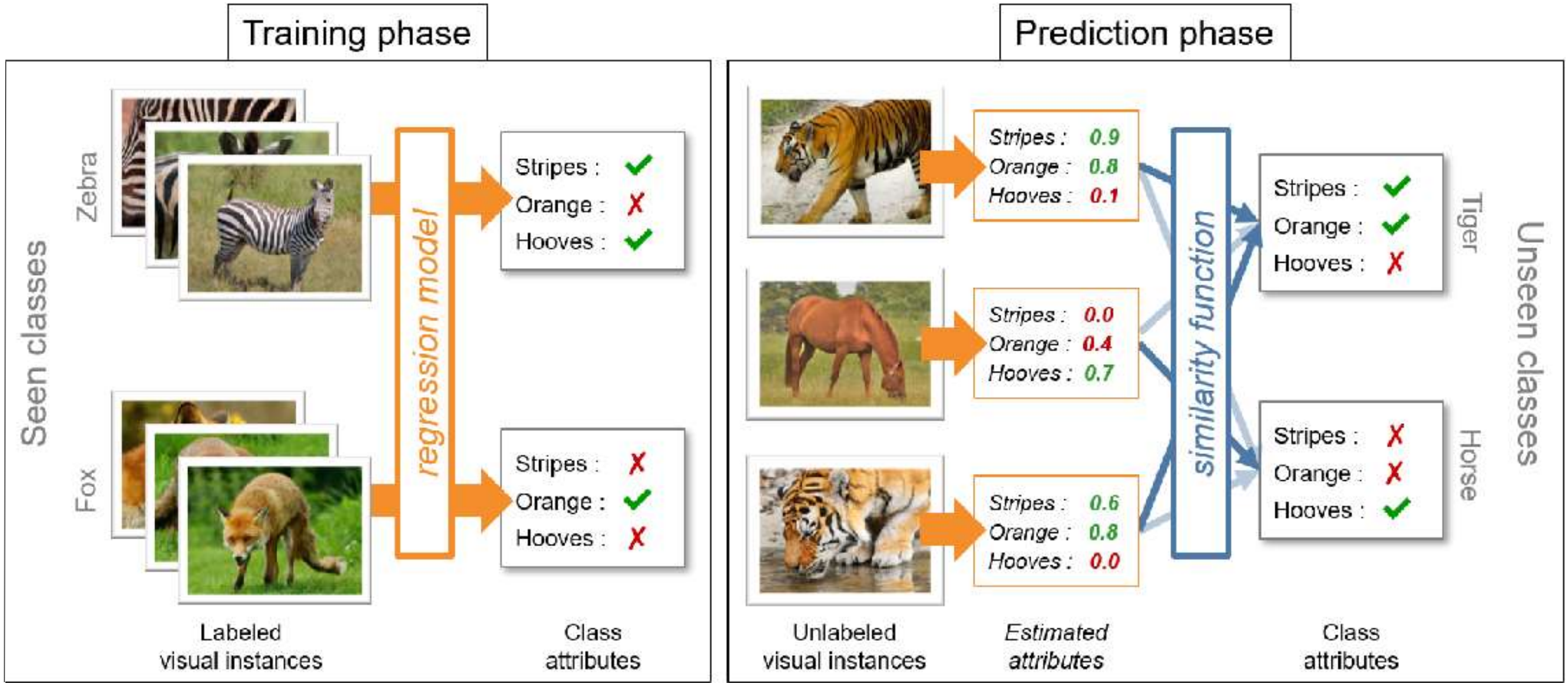
Attributes Vector

Definition: Zero-Shot Learning

- Traditional Learning
  - Input:
    - Data set $X = \{X_0, X_1, X_2, \ldots. X_d\}$
    - Labels for each data point $L = \{L_0, L_1, L_2, \ldots. L_m\}$
    - Typically we have $m << L$
  - Output:
    - A model that maps new data point X_{new} → a label $L_i \in \{L_0, L_1, L_2, \ldots. L_m\}$

- Zero-Shot Learning:
  - Input:
    - Data set $X = \{X_0, X_1, X_2, \ldots. X_d\}$
    - Labels for each data point $L = \{L_0, L_1, L_2, \ldots. L_m\}$
    - Typically we have $m << L$
  - Output:
    - A model that maps new data point X_{new} → a set of attributes (semantic features) $F \in \{F_0, F_1, F_2, \ldots. F_m\}$
    - A second model that estimates the "similarity" between the set of attributes and the attribute profiles
    - New attribute profiles can be added dynamically ("on-the-fly")

Training vs Prediction



Assumption: Semantic attributes are inclusive

Black Bulbul

Grosbeak

Semantic Representation

Bird?
Head Color?
Body Color?

Eagle

We need other attributes:

1- Face color?
2- Beak color?

We need a dynamic representation for features!

A Generative Adversarial Approach for Zero-Shot Learning from Noisy Texts (CVPR, 2018)

- Main idea:
  - Can we "imagine" what a new class will look like "*Visually*" based on its "textual" description?
  - If so we can do the following:
    1. Given a "textual" description of the new class, generate images that "Visually" meets the description
    2. Use these images to train a regular supervised model

- Is this still a "Zero-Shot" learning problem?
  - Form ML perspective: No
  - From Systems perspective: Yes

A Generative Adversarial Approach for Zero-Shot Learning from Noisy Texts (CVPR, 2018)

- Main idea:
  - Can we "imagine" what a new class will look like "*Visually*" based on its "textual" description?
  - If so we can do the following:
    1. Given a "textual" description of the new class, generate images that "Visually" meets the description
    2. Use these images to train a regular supervised model

- But how can we map "textual" descriptions to "Images" (text-to-photo)??

How to “imagine” classes from textual descriptions?

- GAN: Generative Adversarial Networks
  - Assume we have a model that maps images → features

Many

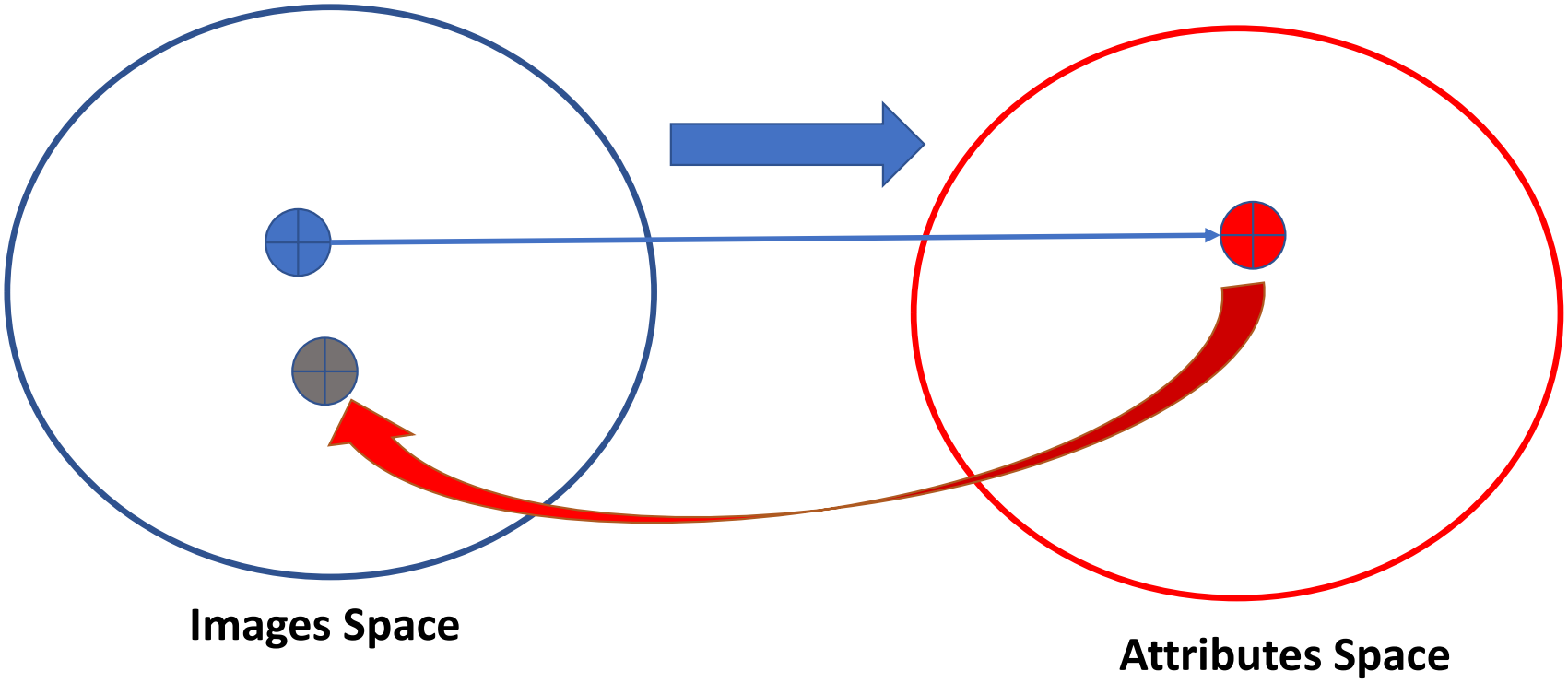
Model

One

Long slender legs? Yes
Long thick necks? Yes
Deep torso build? Yes
Large elongated heads? Yes

Can we map features → images?

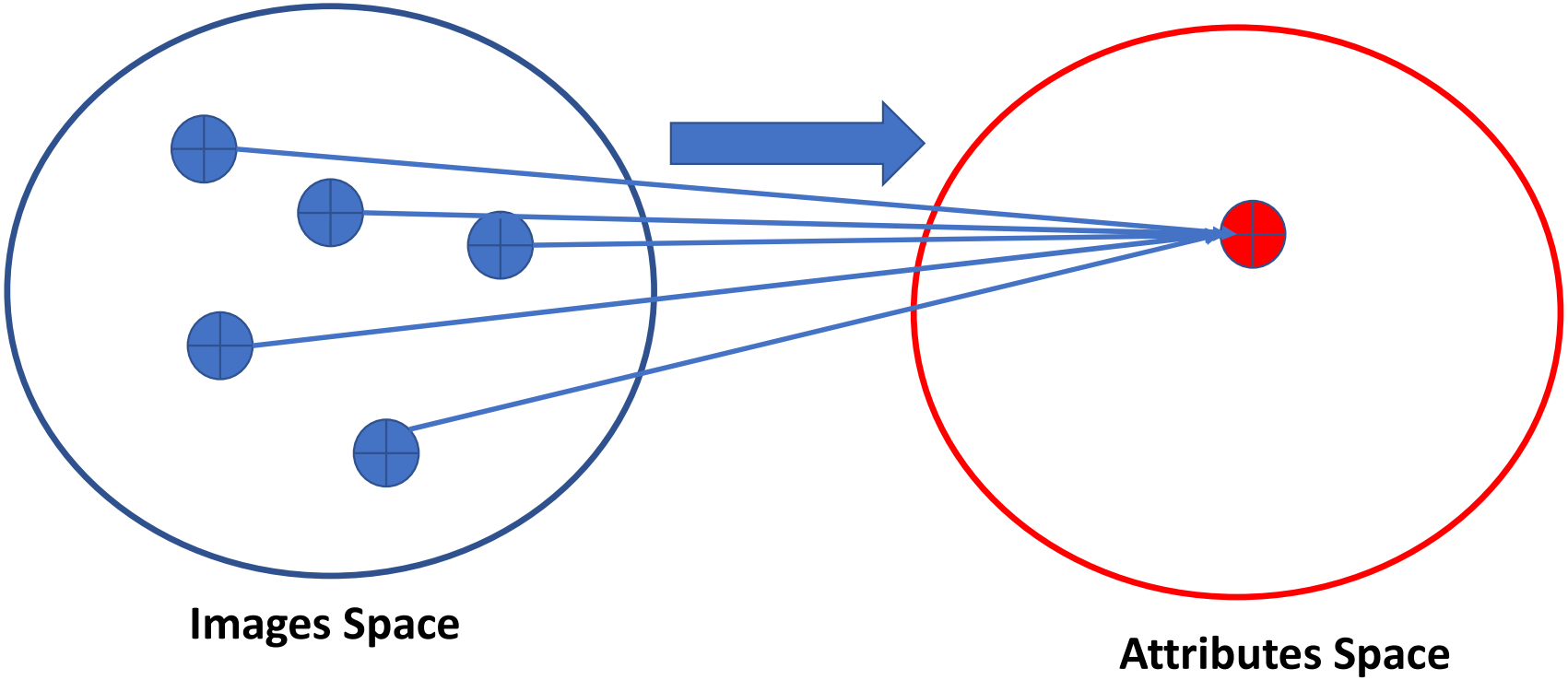
Challenges of GAN: generate “Real” images



We can map a “real” image to the attributes space

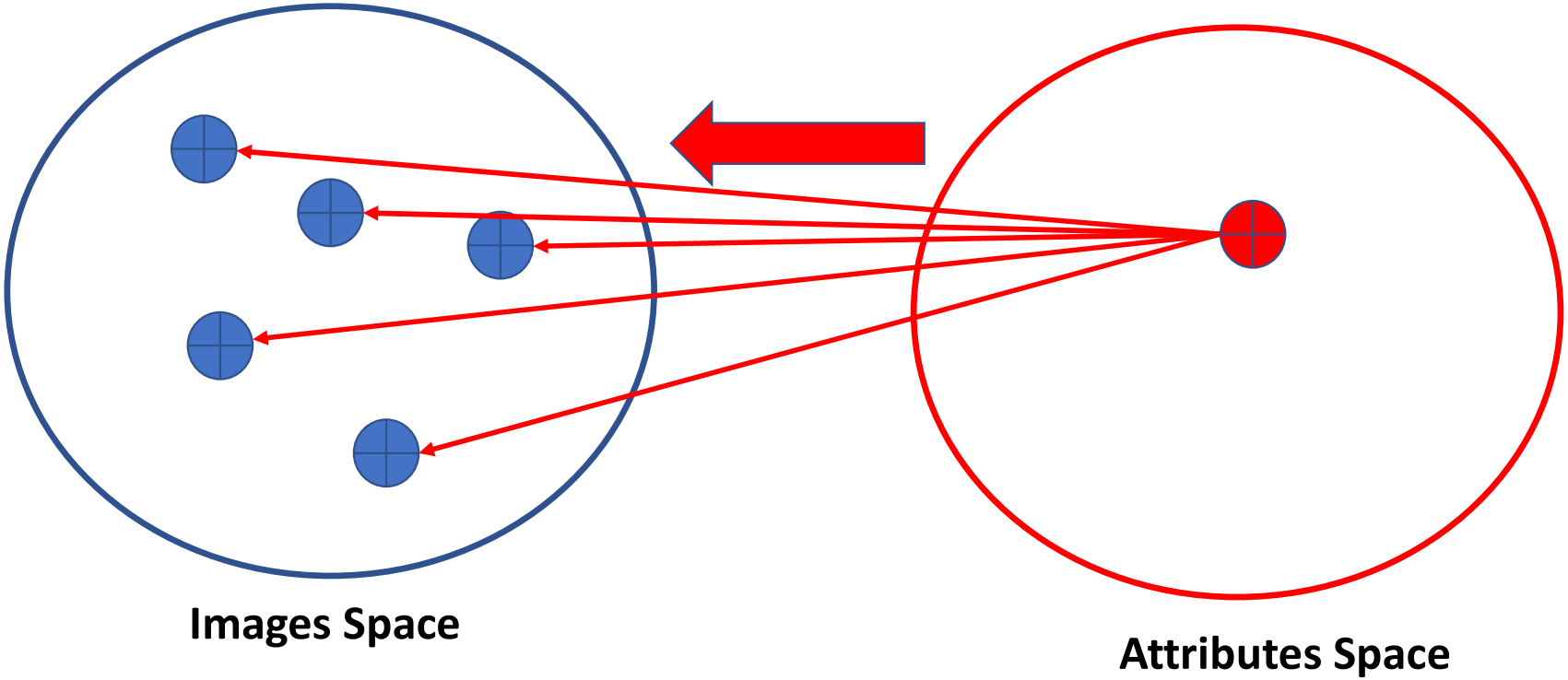
1) How can we map a single vector in the attribute space to a “real” image?

Challenges of GAN: one-to-many mapping



We can map multiple images to a single vector in the attributes space

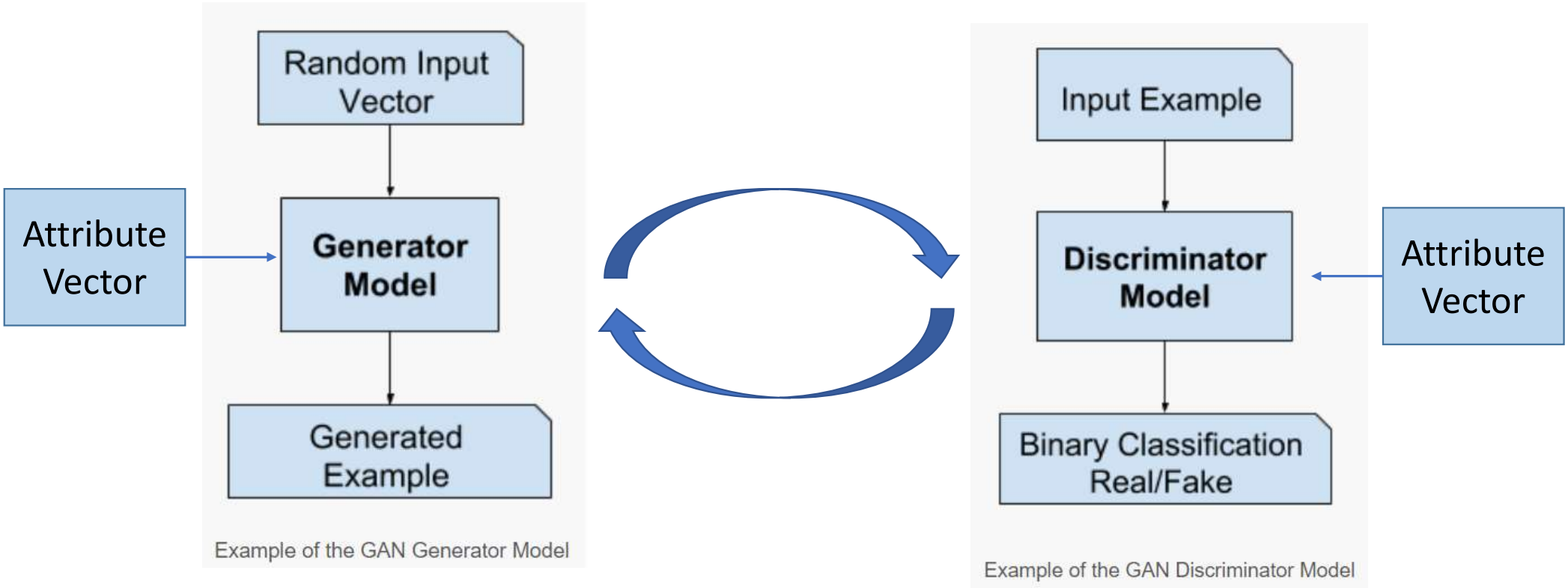
GAN: Challenges



We can map multiple images to a single vector in the attributes space

1) How can we map a single vector in the attribute space to multiple images?

Two Adversarial Nets: Zero-sum Game

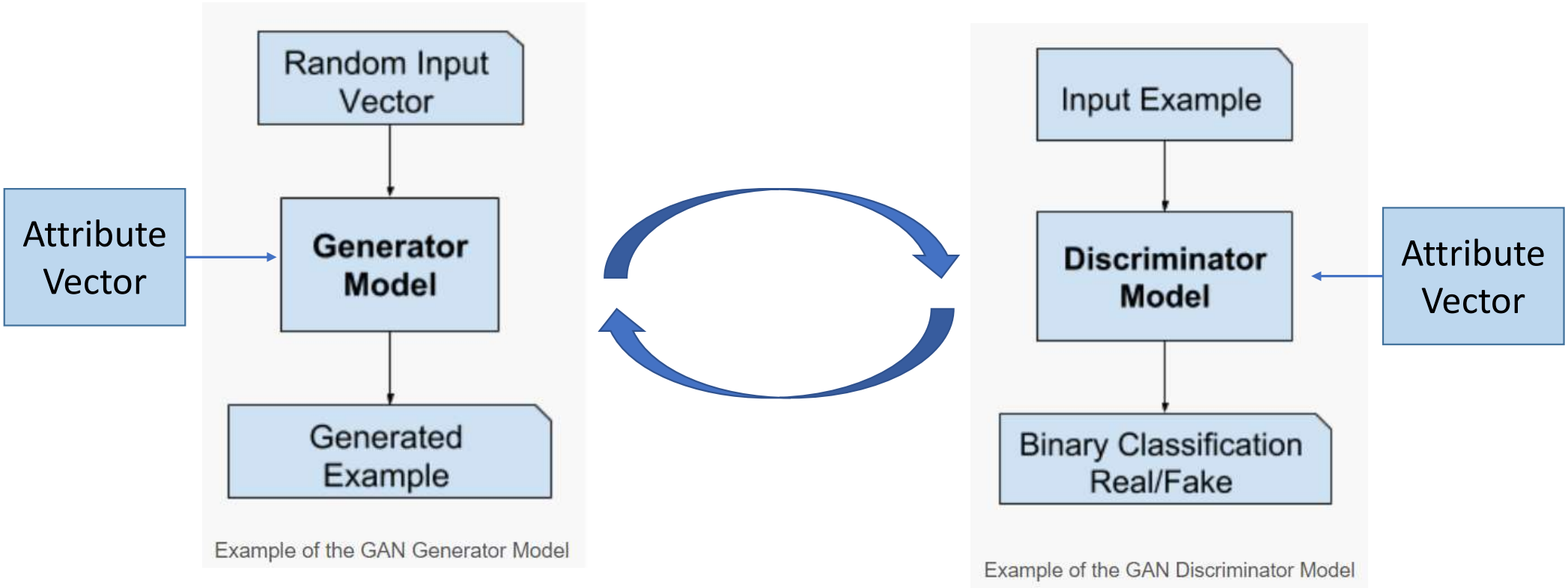


We will have two adversarial models:

(1) a model that generates an image

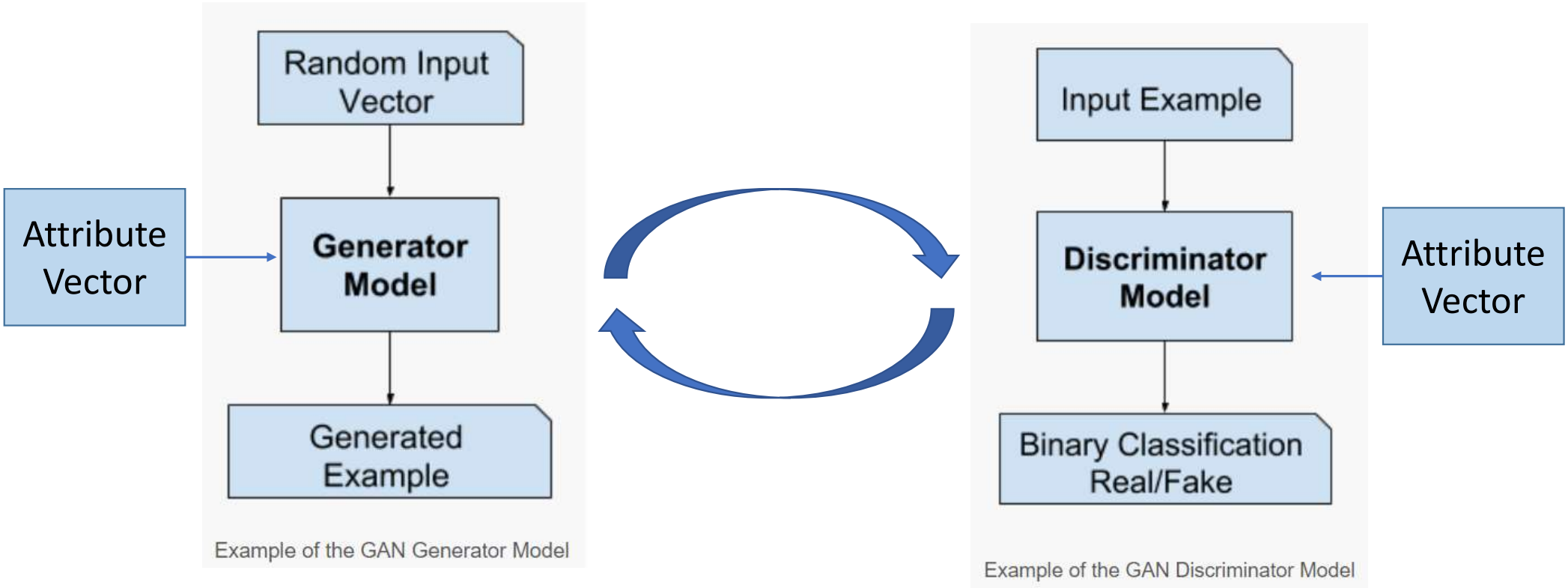
(2) a model that can classify the image as real or fake

Two Adversarial Nets: Zero-sum Game



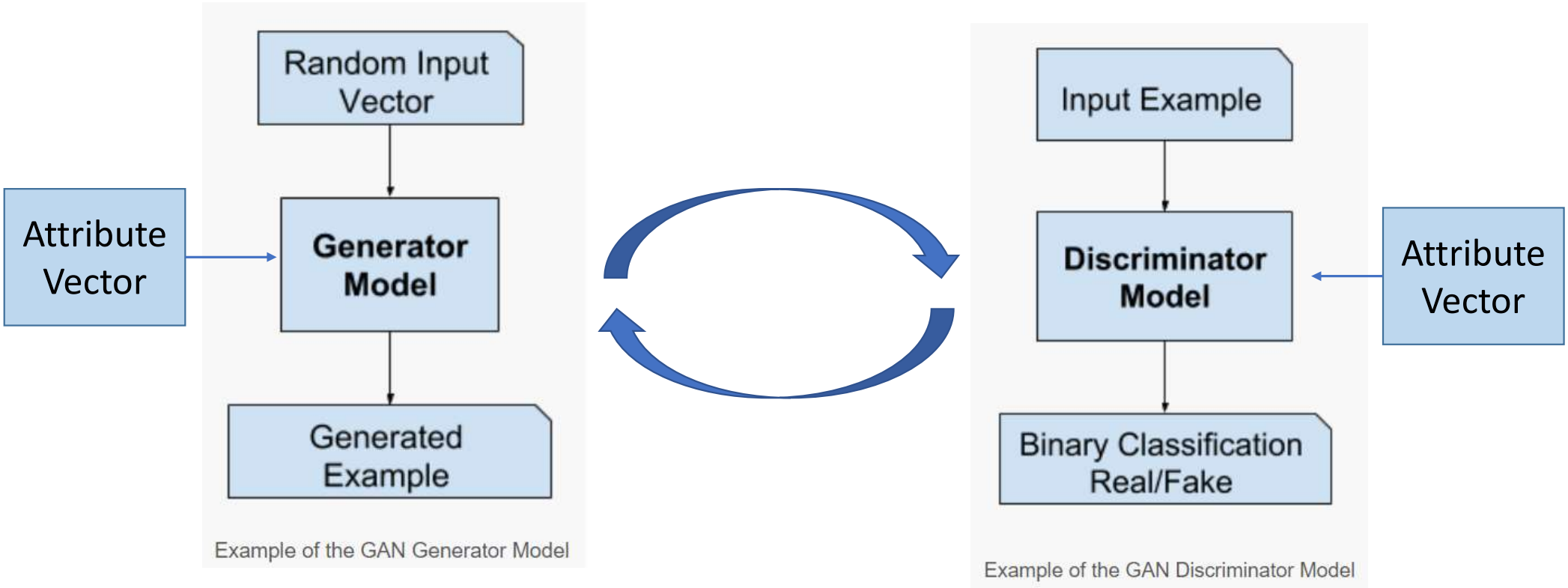
In case the Generator produces an image and the Discriminator says its fake -> no changes to discriminator + update the weights of the Generator's model

Two Adversarial Nets: Zero-sum Game



In case the Generator produces an image and the Discriminator says its Real -> no changes to Generator + update the weights of the Discriminator's model

Two Adversarial Nets: Zero-sum Game



Terminate when a "batch" of generated images are each classified as "not sure = 50% real and 50% fake"

Examples of GAN generated human faces

Example of the Progression in the Capabilities of GANs From 2014 to 2017. Taken from The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.

A Generative Adversarial Approach for Zero-Shot Learning from Noisy Texts (CVPR, 2018)

- Main steps:
  1. For a new class, get the describing “Wikipedia” article
  2. Map the “Wikipedia” article to a semantic feature vector
  3. Feed the GAN with [Semantic feature vector || random vector]
  4. The GAN generates as many “plausible” images as we need to train our classifier with the new class

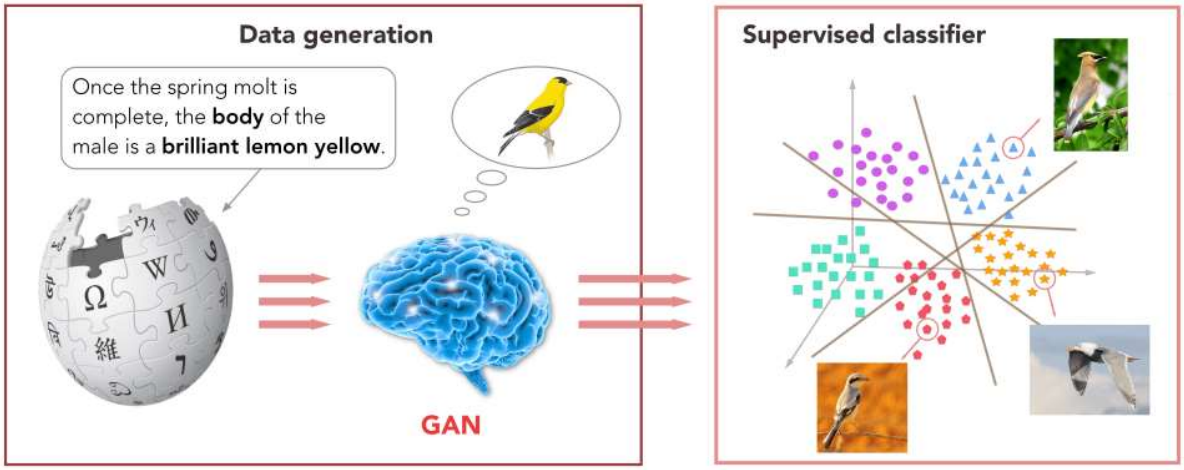


Figure 1: Illustration of our proposed approach. We leverages GANs to visually imagine the objects given noisy Wikipedia articles. With hallucinated features, a supervised classifier is trained to predict image’s label.

Evaluation

- Datasets:

1. Caltech UCSD Birds-2011 (CUB): 200 categories of bird species with a total of 11,788 images
2. North America Birds (NAB): 1011 classes and 48,562 images

- Experimentation setup:

1. Textual Representation: Raw Wikipedia articles text is tokenized into words, the stop words are removed and remaining words are stemmed. Finally Term-Frequency Inverse-Document-Frequency(TF-IDF) feature vector is extracted.
2. Visual Features: There are seven semantic parts: “head”, “back”, “belly”, “breast”, “leg”, “wing”, “tail”.

Evaluation: Testing on unseen classes

- ZeroShot Recognition:

In the scenario of SCS-split, for each unseen class, there exists one or more seen classes that belong to the same parent category.

In SCE-split, the parent categories of unseen classes are separate from those of the seen classes.

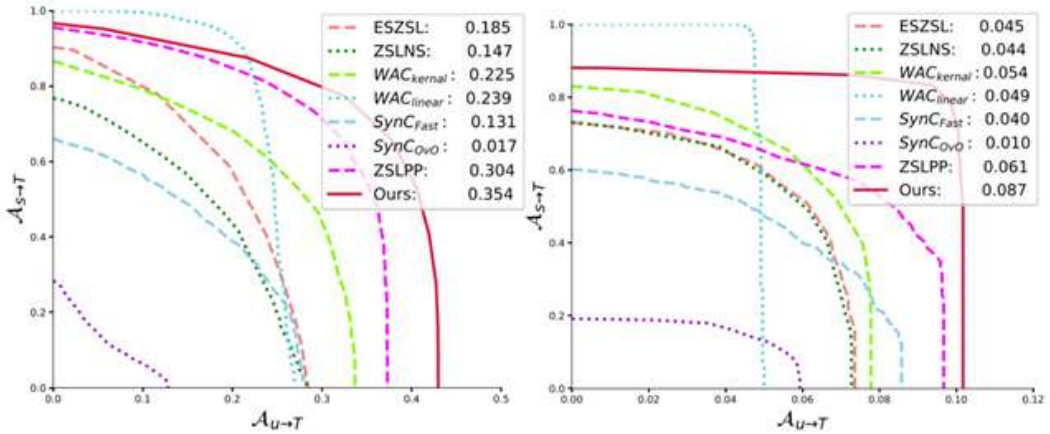
	CUB		NAB	
methods	SCS	SCE	SCS	SCE
MCZSL [1]	34.7	–	–	–
WAC-Linear [9]	27.0	5.0	–	–
WAC-Kernel [8]	33.5	7.7	11.4	6.0
ESZSL [36]	28.5	7.4	24.3	6.3
SJE [3]	29.9	–	–	–
ZSLNS [32]	29.1	7.3	24.5	6.8
SynC$_{fast}$ [5]	28.0	8.6	18.4	3.8
SynC$_{OVO}$ [5]	12.5	5.9	–	–
ZSLPP [10]	37.2	9.7	30.3	8.1
GAZSL	**43.7**	**10.3**	**35.6**	**8.6**

Table 1: Top-1 accuracy (%) on **CUB** and **NAB** datasets with two split settings.

Evaluation: Testing on both seen and unseen classes

- ZeroShot Recognition:

The conventional zero-shot recognition considers that queries come from only unseen classes.



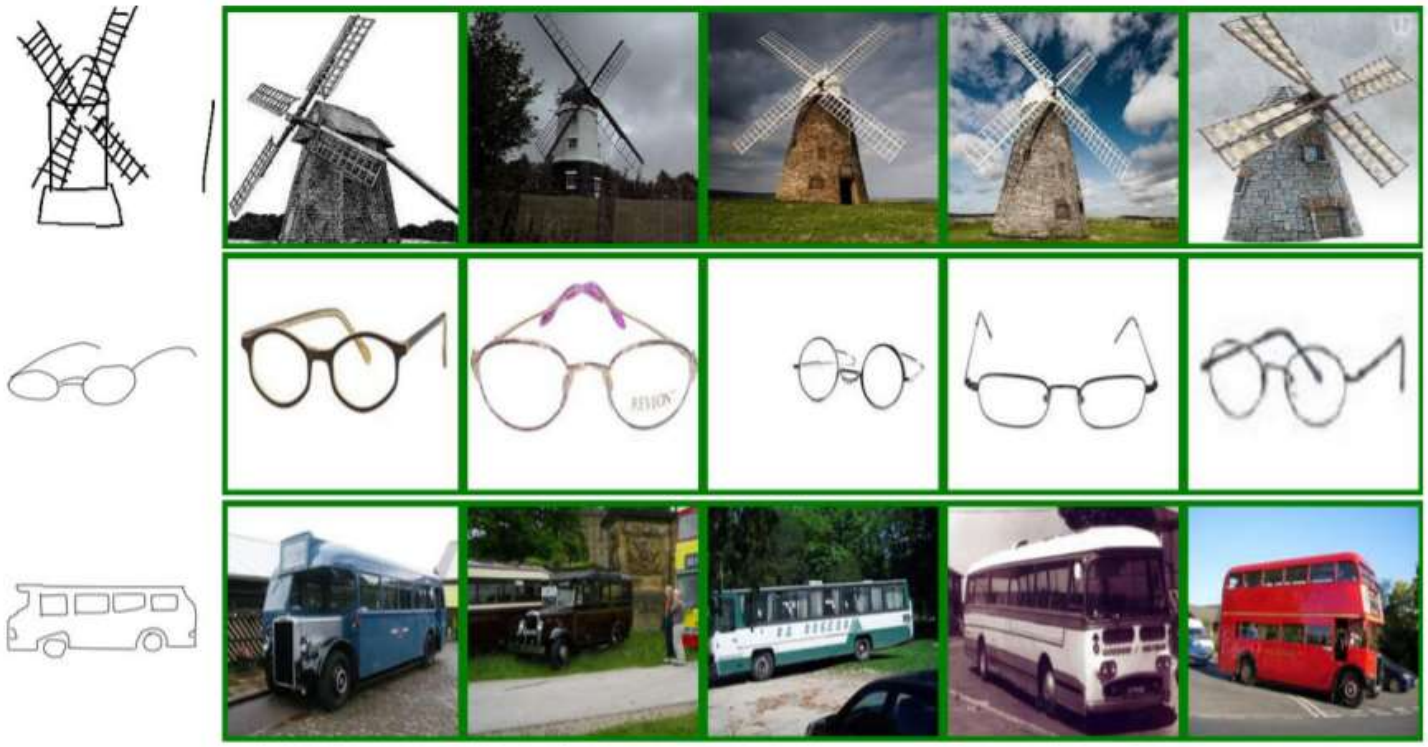
(a) CUB with SCS splitting (b) CUB with SCE splitting

Figure 4: Seen-Unseen accuracy Curve

A calibration factor α is used to balance the accuracy scores between seen and unseen classes

Evaluation: zero-shot retrieval

- Main idea:
  - Given an image as a query, find the most similar images
  - The input image can be as simple as a hand drawn sketch



Evaluation: zero-shot retrieval

	CUB			NAB		
methods	25%	50%	100%	25%	50%	100%
ESZSL [36]	27.9	27.3	22.7	28.9	27.8	20.85
ZSLNS [32]	29.2	29.5	23.9	28.78	27.27	22.13
ZSLPP [10]	42.3	42.0	36.3	36.9	35.7	**31.3**
VP-only	17.8	16.4	13.9	15.1	13.1	11.5
GAN-only	18.0	17.5	15.2	21.7	20.3	16.6
GAZSL	**49.7**	**48.3**	**40.3**	**41.6**	**37.8**	31.0

Table 3: Zero-Shot Retrieval using mean Average Precision (mAP) (%) on CUB and NAB with SCS splitting.

What's next?

- In this paper:
  - The highest accuracy achieved on "unseen" classes is < 50%
  - If there is no "common parent" with a "seen" class, the accuracy drops to < 15%
  - Most recent papers have achieved a slightly better performance (< 55%)

- Does it have to be zero-shot? Or can we use a few-shot learning?
  - With 5-shot, (Kai Li *et. al, CVPR 2020)* achieved 84% accuracy on the CUB dataset

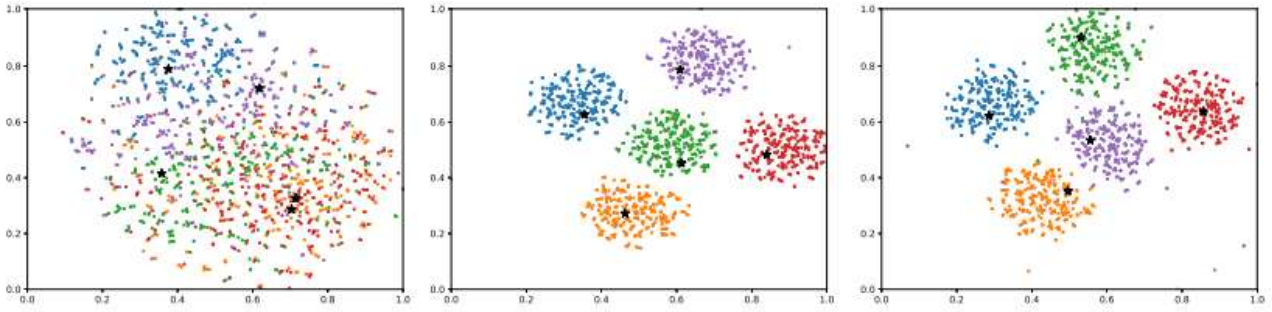


Figure 2. t-SNE [26] visualization of synthesized feature embeddings. The real features are indicated by ⋆. Different colors represent different classes.

Conclusion

- Zero-shot (or few-shot) learning is an effective technique for transfer learning in ML problems

- It can be very useful in cases of:
  - Lack of sufficient training samples for particular classes (for example, if we have many images for a horse, but none (or very few) for a zebra)
  - We can map inputs from seen and unseen classes to a set of attributes (either manually defined or automatically extracted from textual descriptions)

- The applicability of Zero-shot learning (ZSL) is beyond image classification
  - For example, suppose we want to train a performance predictor for any system
  - The performance predictor is dependent on the workload
  - Every time the workload changes, we need to "re-train" the prediction model
  - If we can map *any* workload to a set of attributes, we can use ZSL or FSL to quickly adapt to unseen workloads

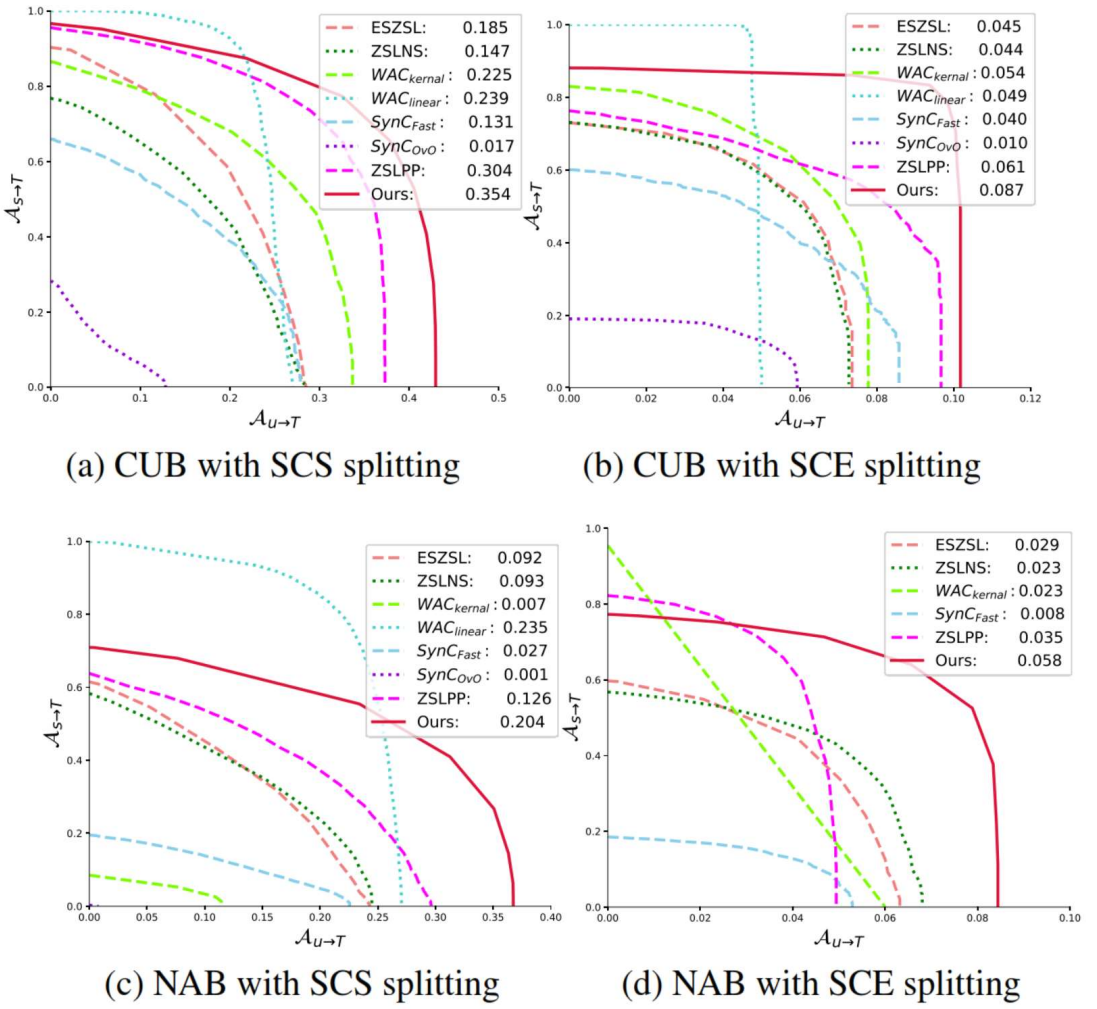
Thanks!

Evaluation:

- ZeroShot Recognition:

The conventional zero-shot recognition considers that queries come from only unseen classes.

A calibration factor α is used to balance the accuracy scores between seen and unseen classes



(a) CUB with SCS splitting (b) CUB with SCE splitting

(c) NAB with SCS splitting (d) NAB with SCE splitting

Figure 4: Seen-Unseen accuracy Curve on two benchmarks datasets with two split settings