Deep Learning

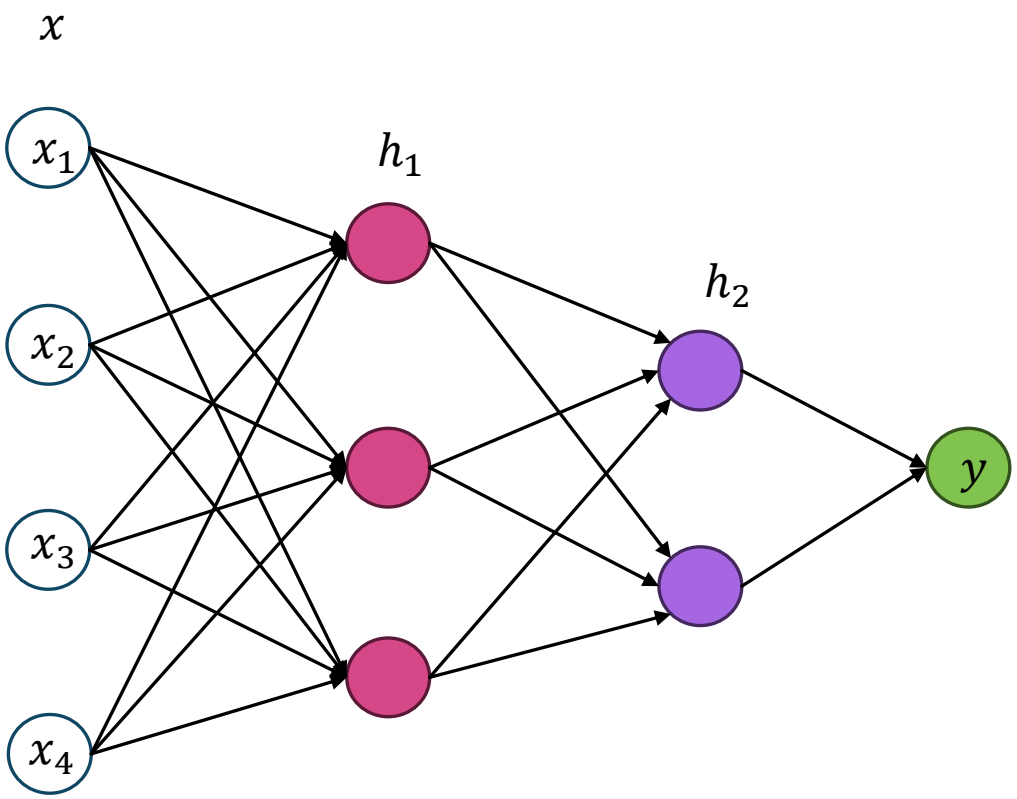
Angshuman Paul

Assistant Professor

Department of Computer Science & Engineering

Representation Learning

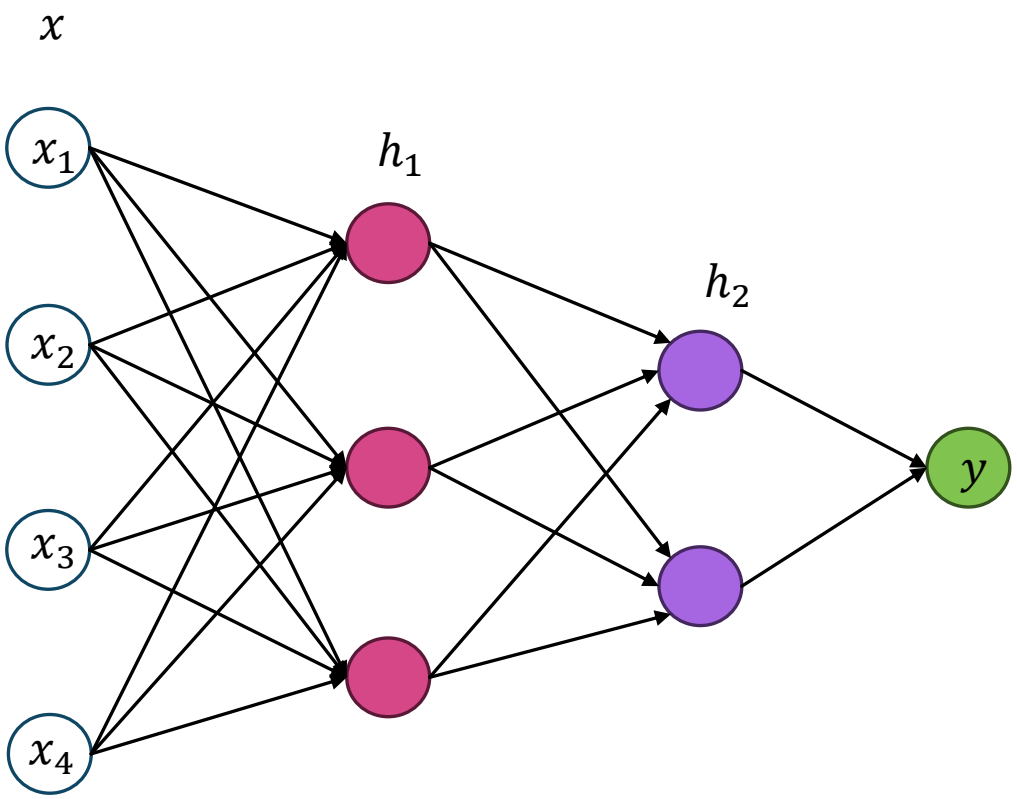
Neural Networks Recap



- Backpropagation was made popular by Rumelhart in 1986
- But until almost 2006, it was very difficult to train a deep model with backprop
  - Even after large number of epochs, training did not converge
- Then something changed in 2006
- What is the change?

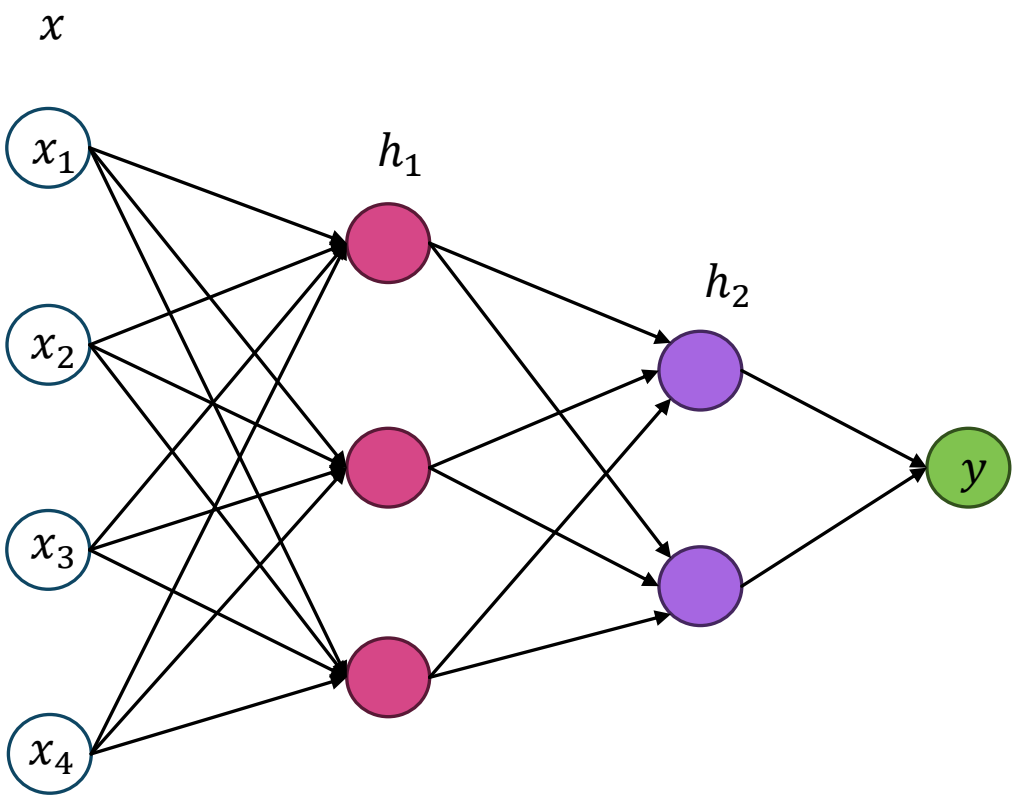
Unsupervised Pre-training

Unsupervised Pre-training



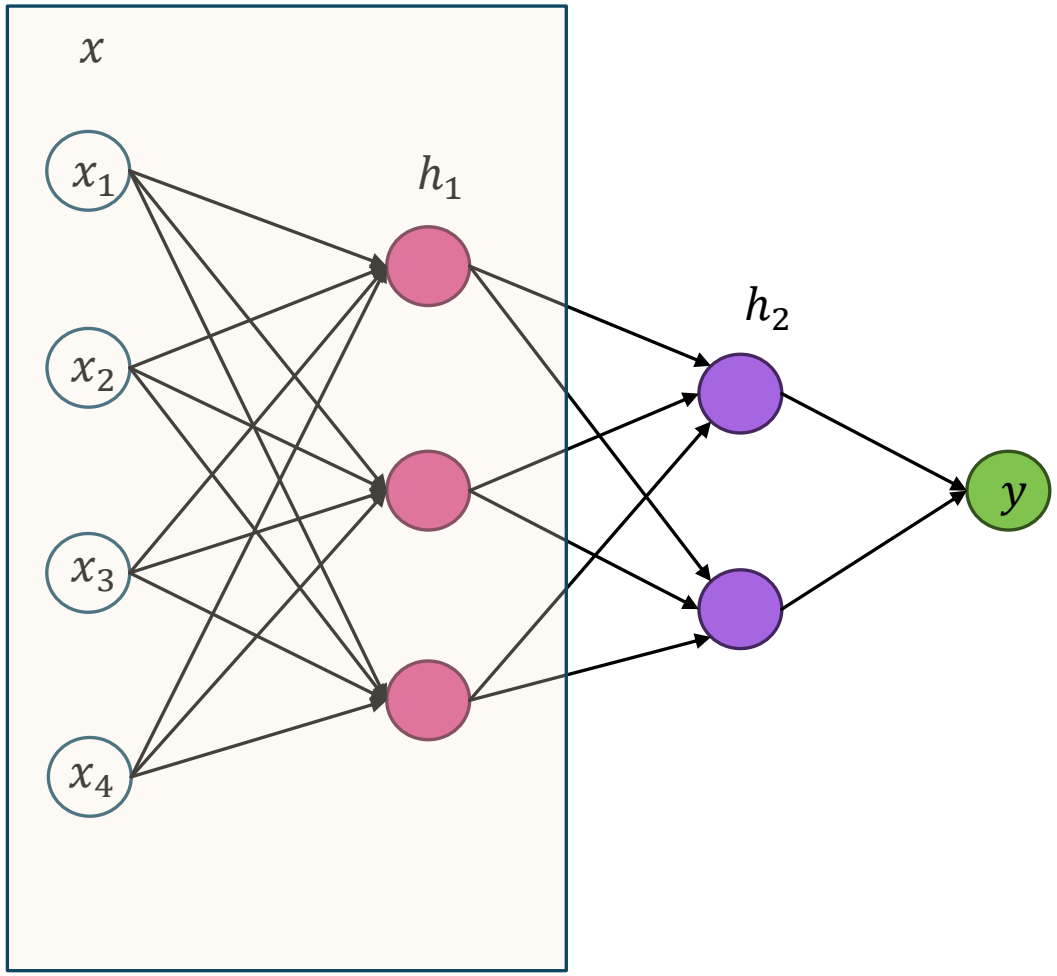
- In the original paper, unsupervised pre-training was proposed in the context of RBM
- However, it is equally applicable in the context of autoencoder

Unsupervised Pre-training: How to Do?



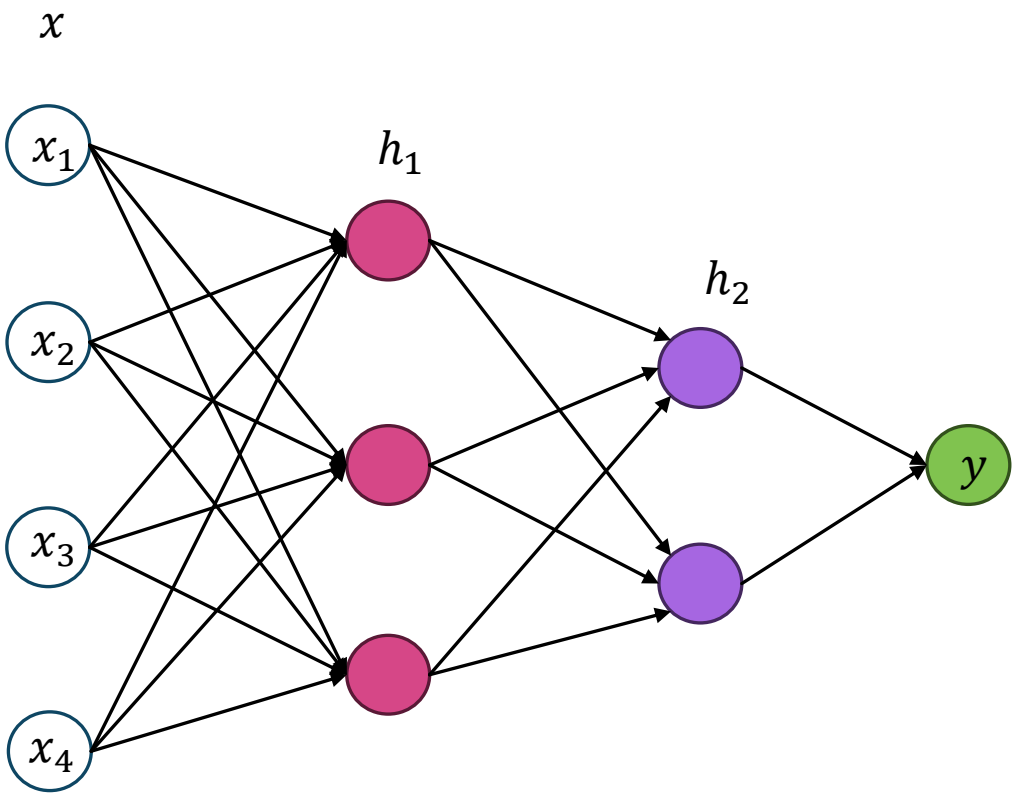
- Initially, forget about the original task that you were interested in
- Focus on the first two layers x and h_1

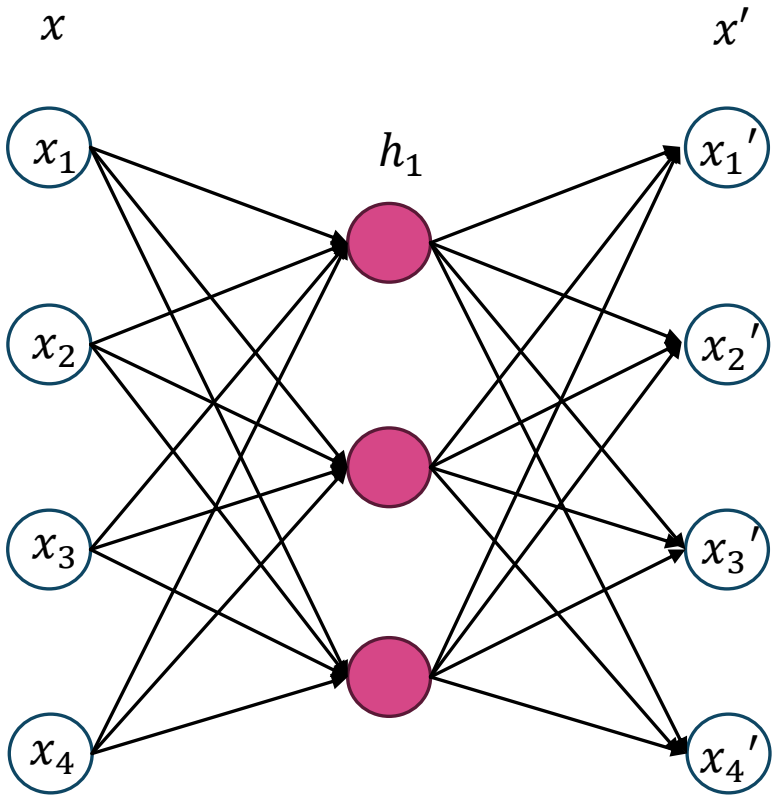
Unsupervised Pre-training: How to Do?



- Initially, forget about the original task that you were interested in
- Focus on the first two layers x and h_1
- We will first train the parameters between these two layers using an unsupervised objective

Unsupervised Pre-training: How to Do?

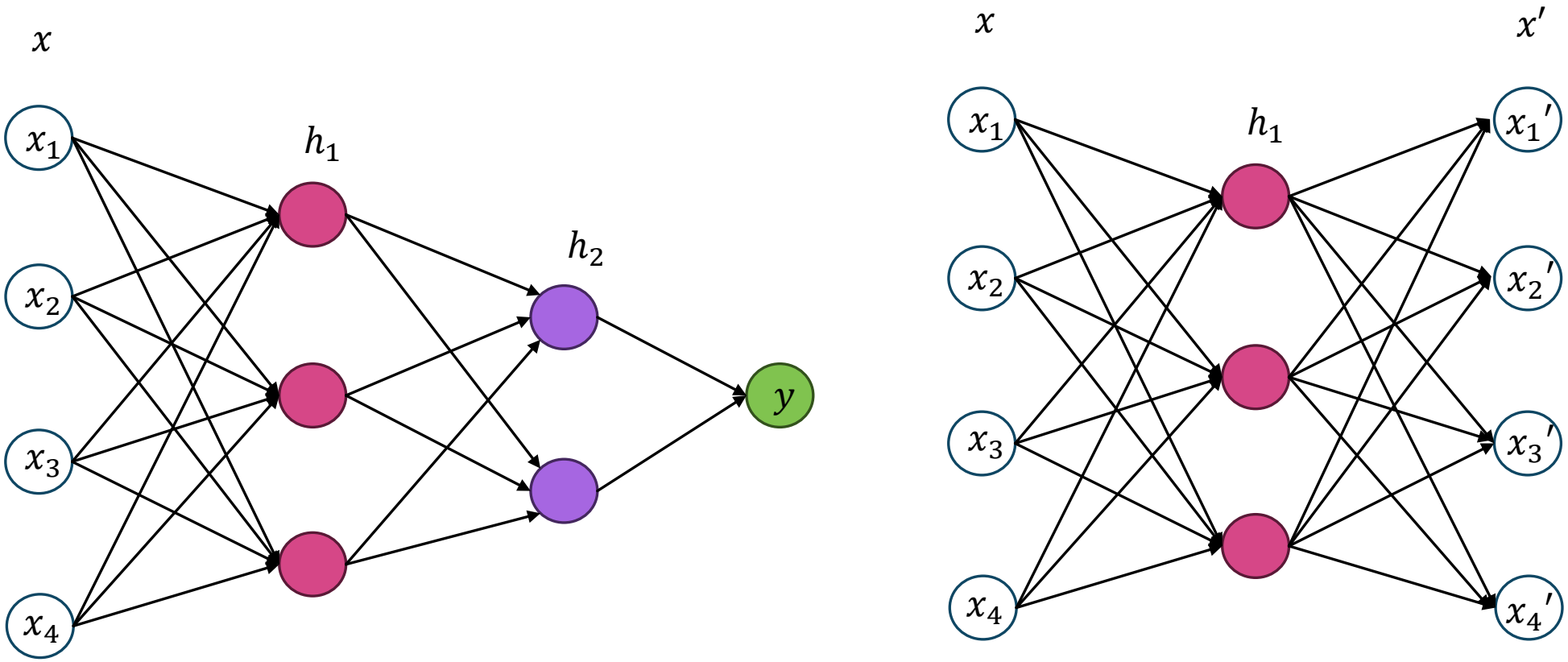




Reconstruct x as x' from h_1

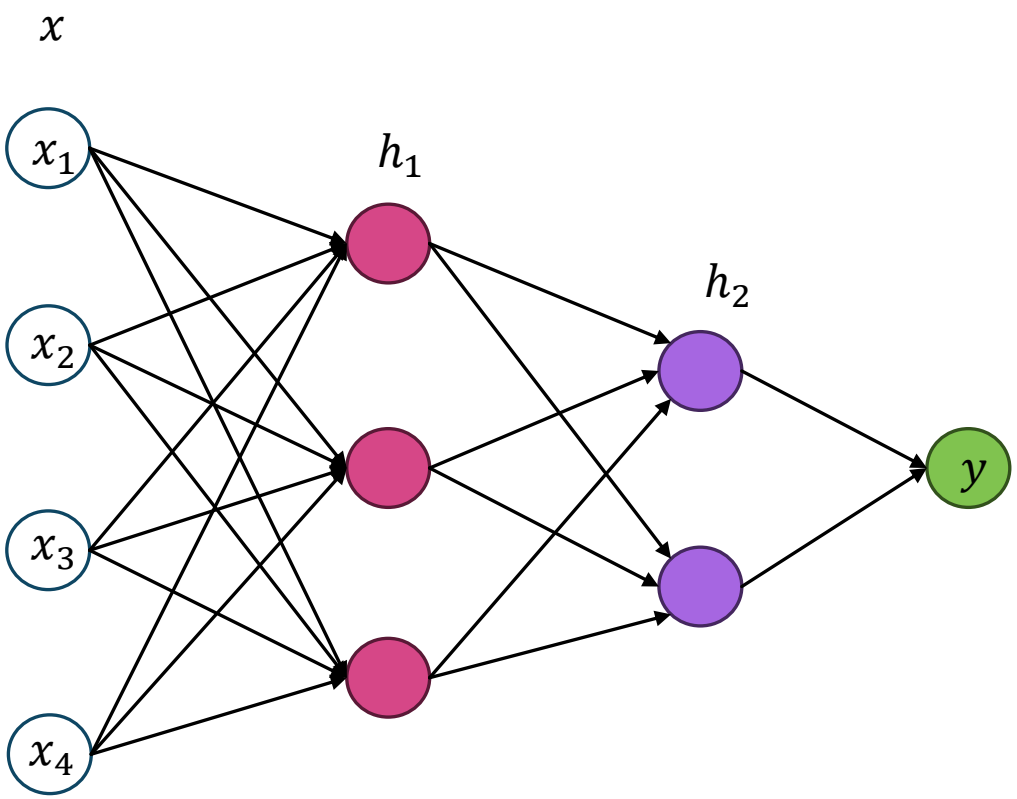
Minimize the MSE between x and x'. This is an unsupervised task

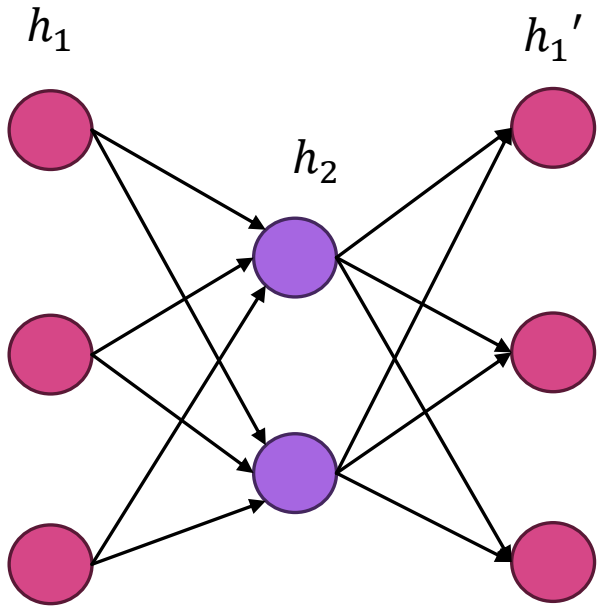
Unsupervised Pre-training: How to Do?



After this step, h_1 has learnt an abstract representation of the input and learnt the important features of the input data

Unsupervised Pre-training: Next Step

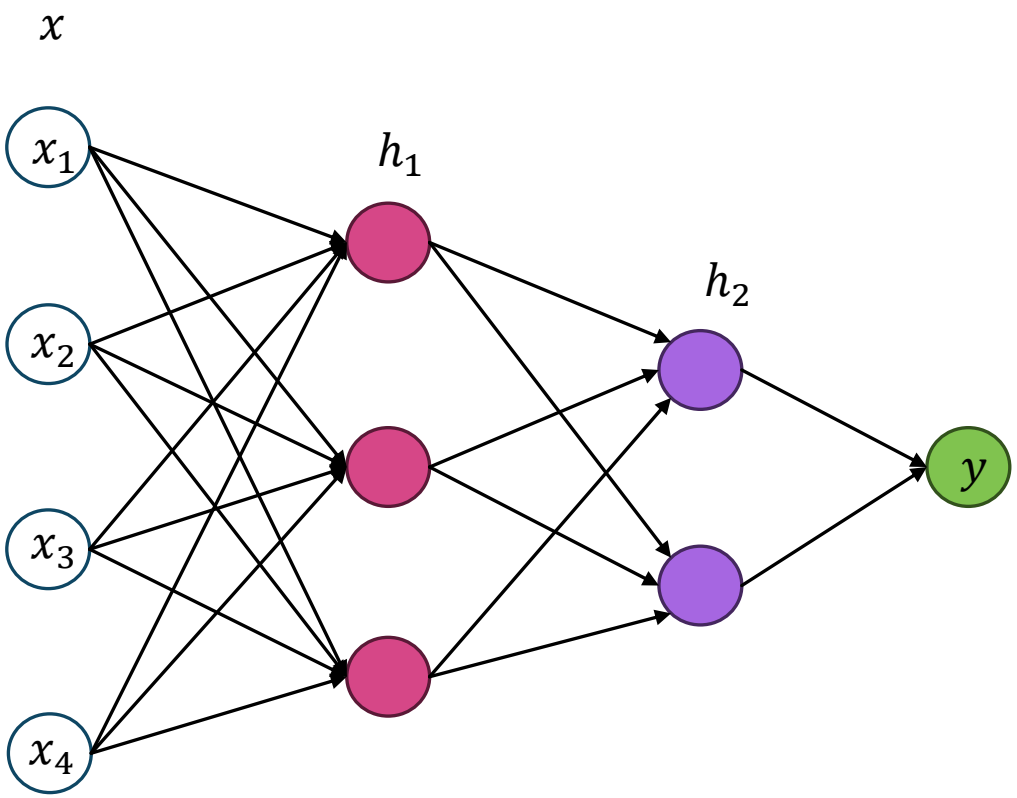


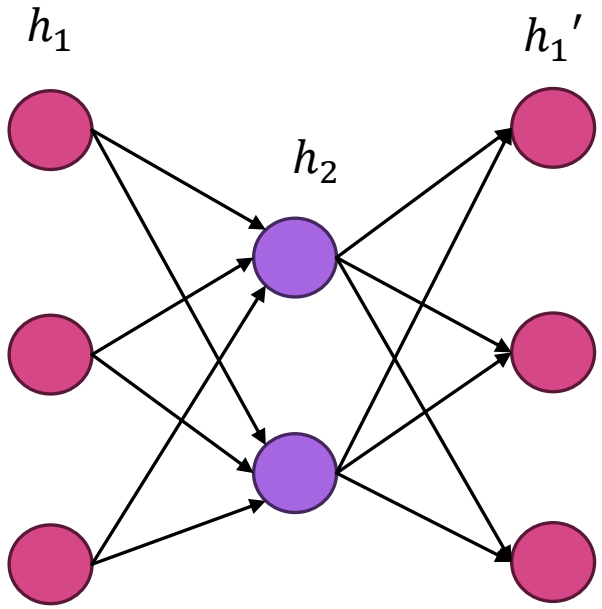


Fix the parameters between x and h_1 and learn the parameters between h_1 and h_2 through autoencoding

Minimize MSE between $h_1'$and h_1

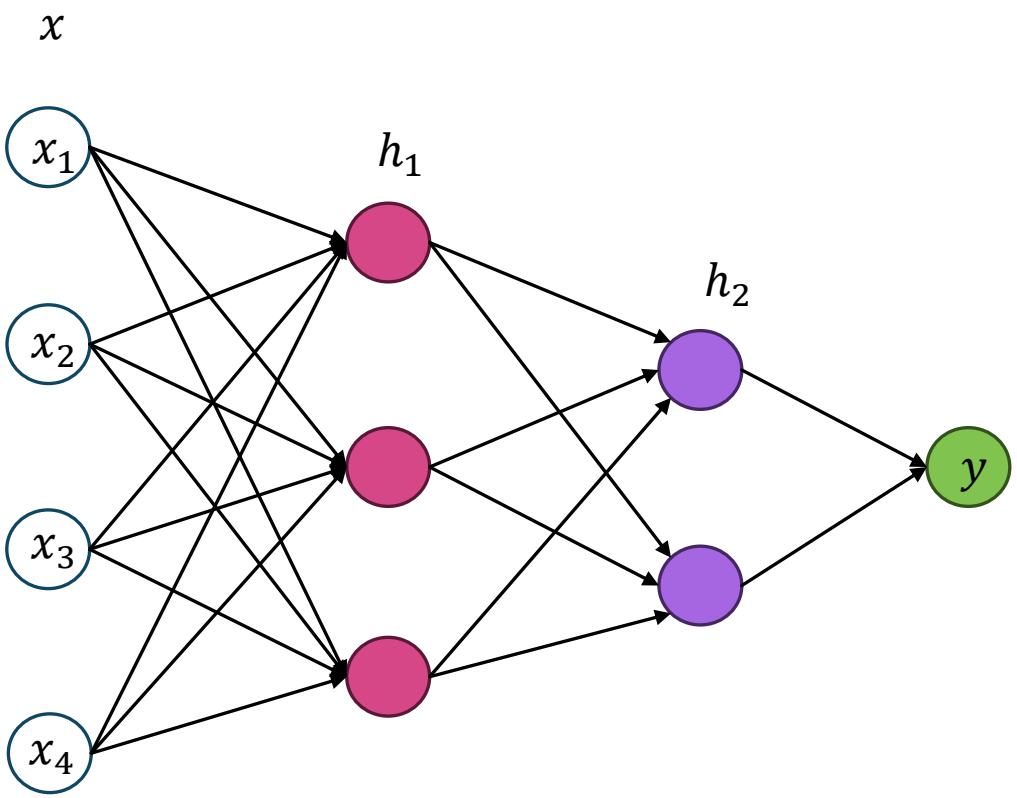
Unsupervised Pre-training: Next Step

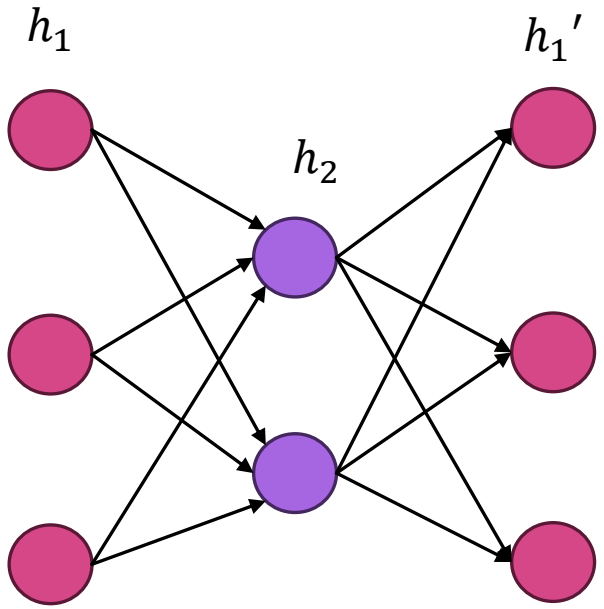




For deeper networks, repeat these steps for all the layers except the last layer

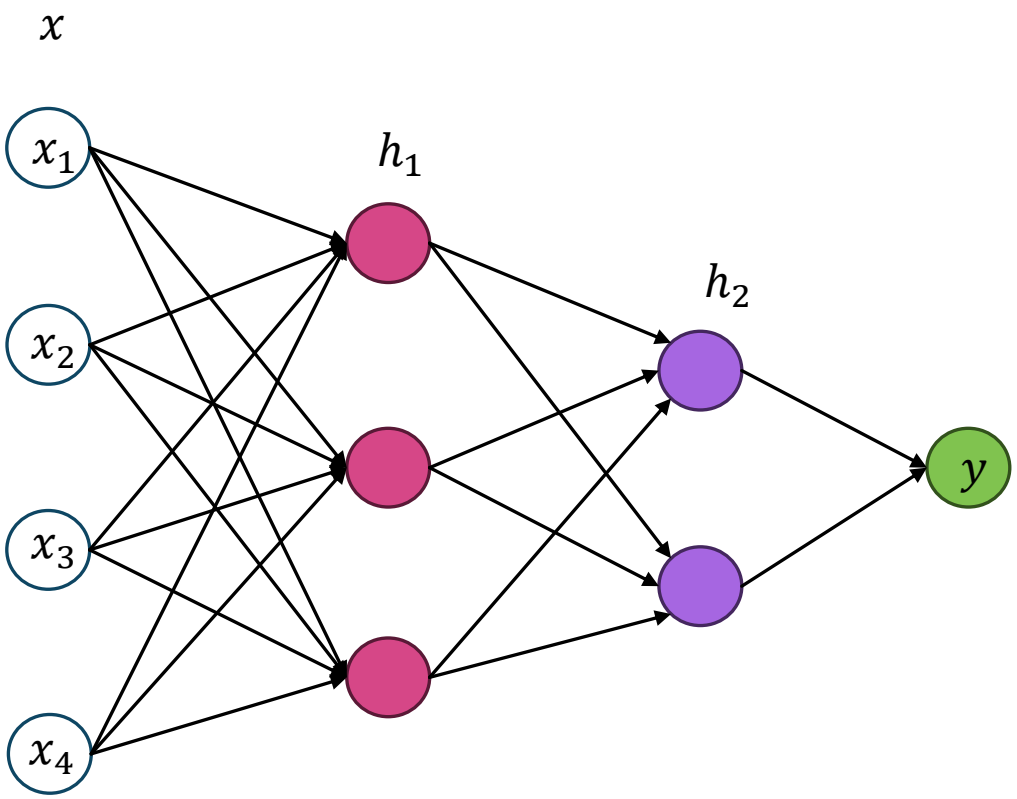
Unsupervised Pre-training: Next Step





Greedy Layer-wise unsupervised pre-training

Fine Tuning



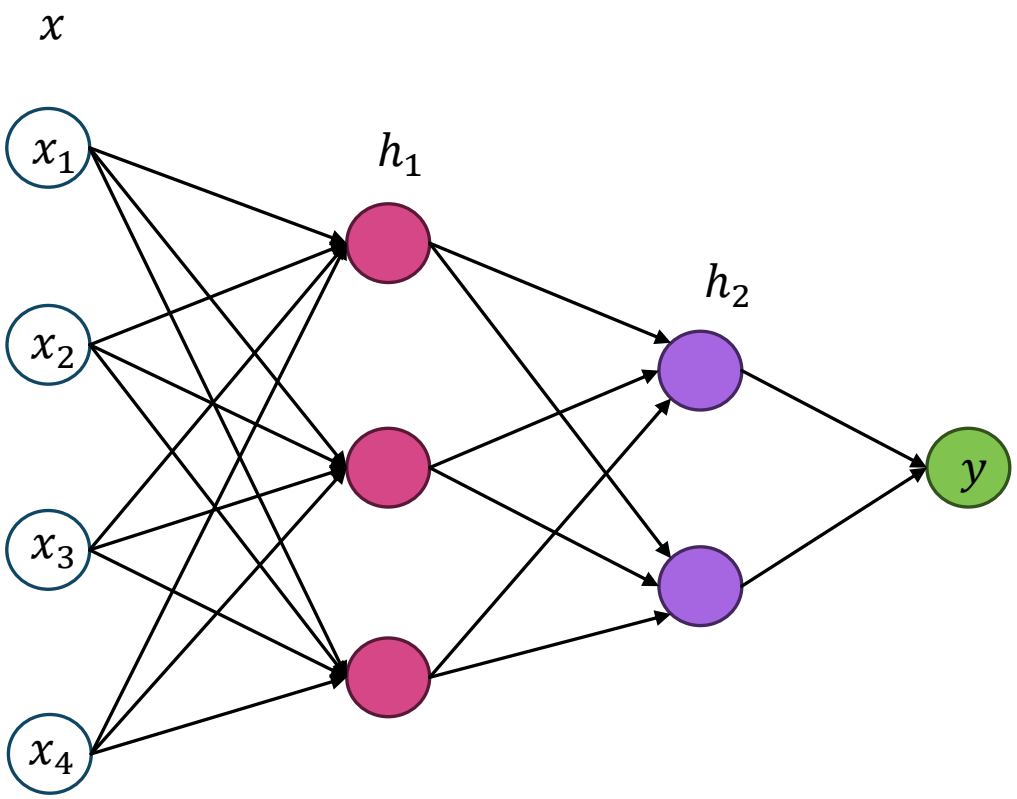
- After Greedy layer-wise pre-training is done, add the output layer
- Keeping all the learnt parameters, I will fine tune the entire neural network for the original task

Greedy Layer-wise Pre-training: Why Such a Name?

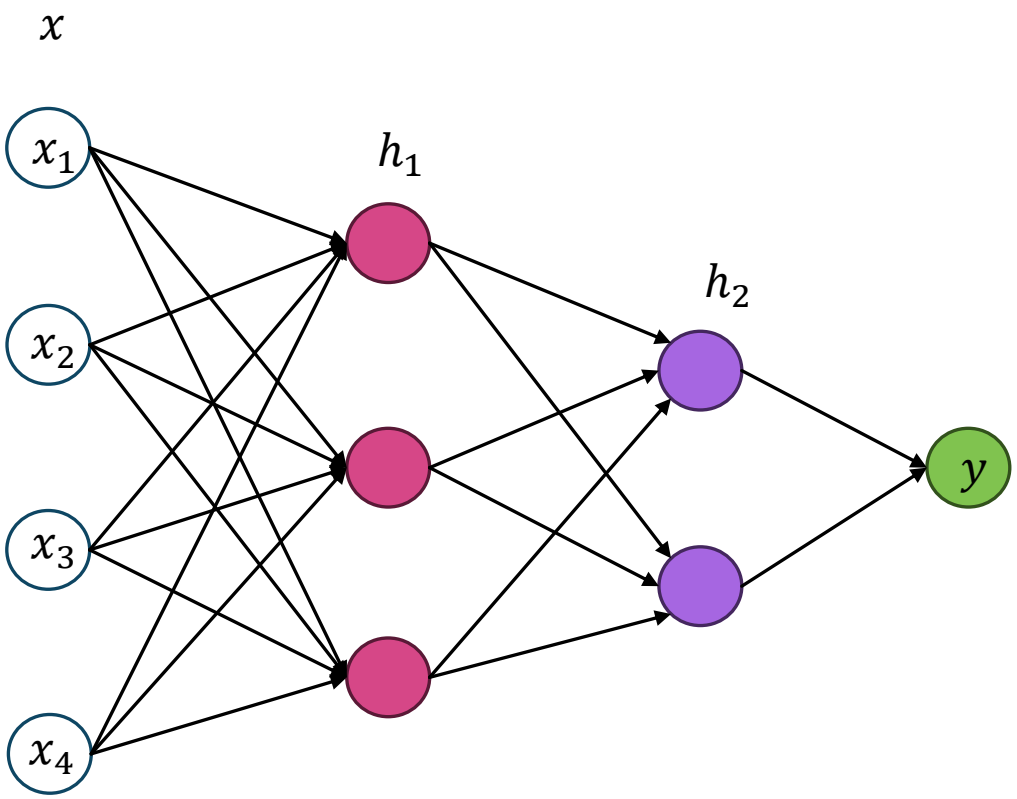
- Greedy: Optimizes each piece of the solution independently, one piece at a time, rather than jointly optimizing all pieces

- Layer-wise: Because these independent pieces are the layers of the network

Why Does this Strategy Work?

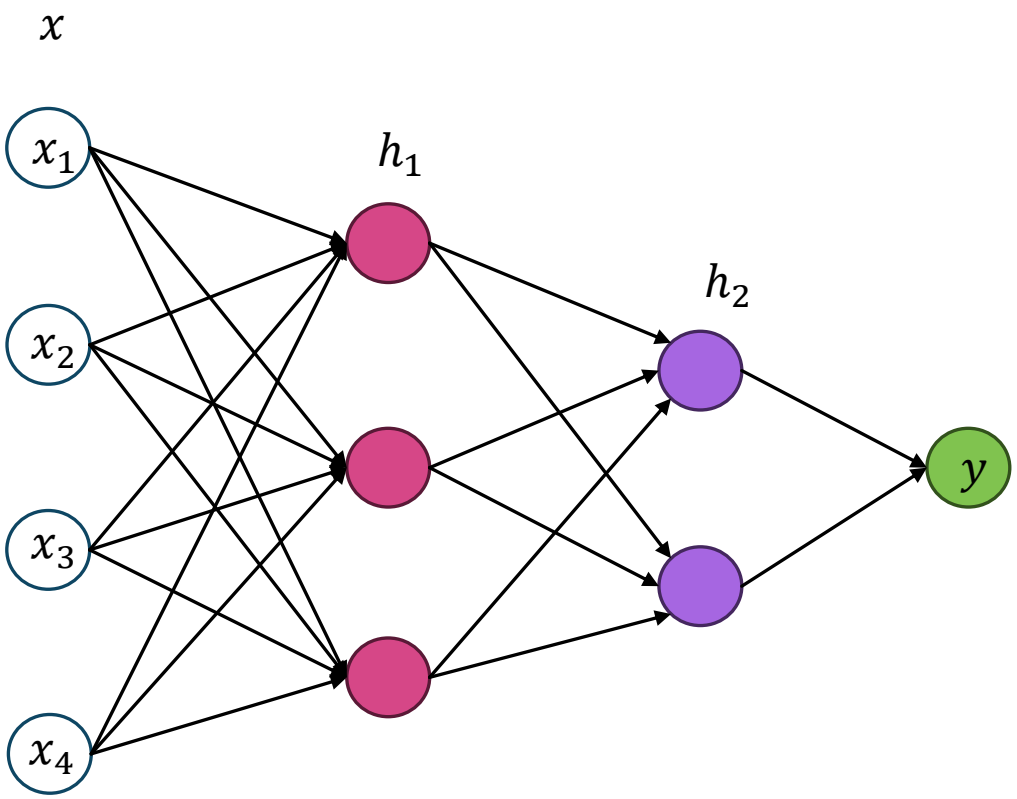


Why Does this Strategy Work?



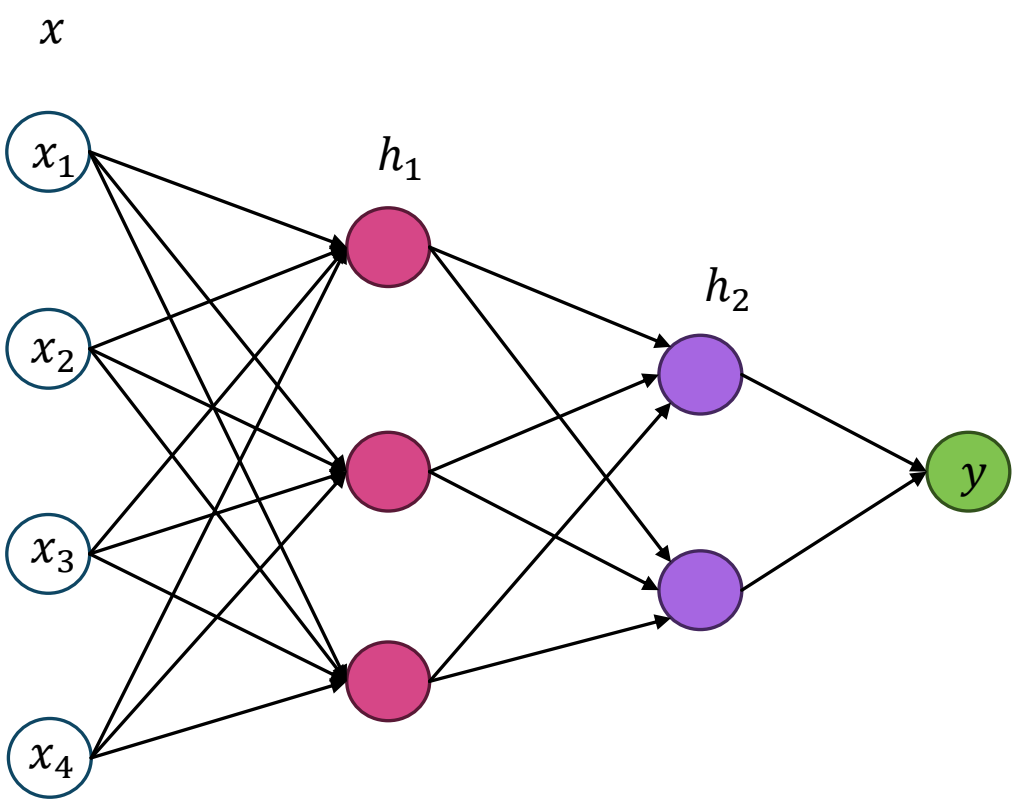
- Better initialization

Why Does this Strategy Work?



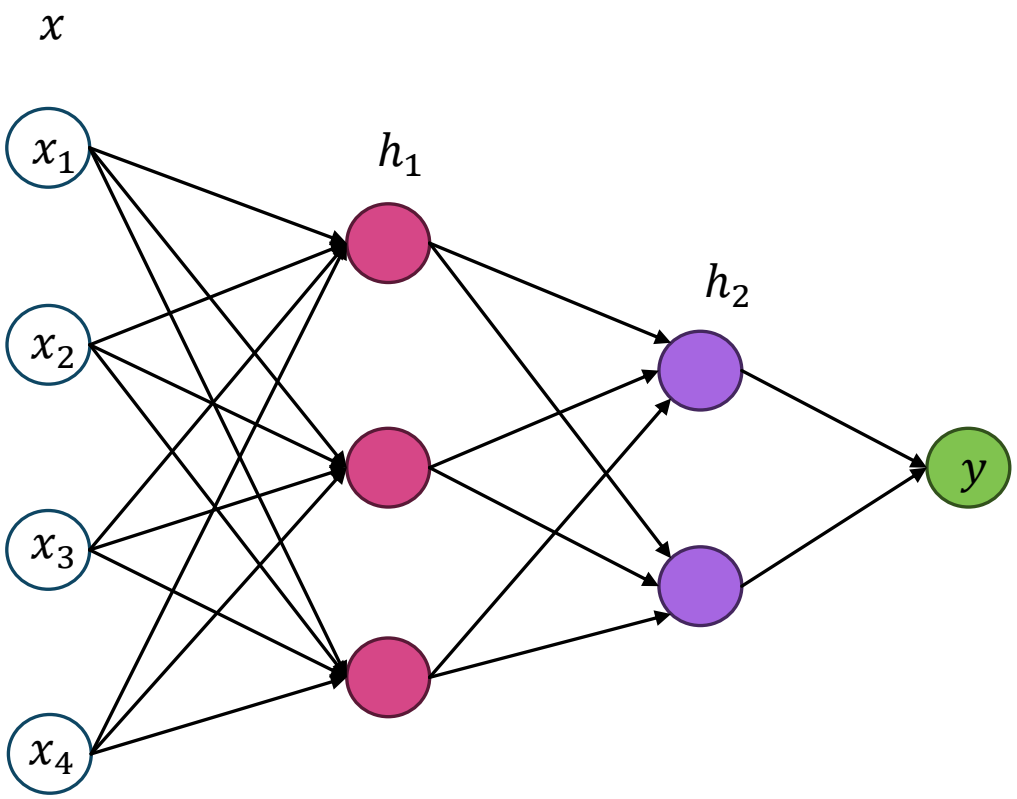
- Unsupervised pre-training may help in reducing the loss of the NN
- However, it was shown that an NN with a last layer having high capacity can easily reduce the loss
- What is high capacity?

Why Does this Strategy Work?



- Unsupervised pre-training may help in reducing the loss of the NN
- However, it was shown that an NN with a last layer having high capacity can easily reduce the loss
- What is high capacity?
  - Highly capable of representing various curves
    - High number of parameters
      - Many nodes

Why Does this Strategy Work?



- If we don't have high capacity NN (but deep NN), unsupervised pre-training may help
- These are empirical results
  - No theoretical proves

Important to Note

- Usually, unsupervised pretraining is more effective when
  - the initial representation is poor: perspective of representation learning
  - The number of training examples is small: the perspective of regularization

- **Unsupervised pretraining does not show a benefit in every problem**

- **In some problem, it even causes harm**

Present Status

- Unsupervised pretraining is **mostly abandoned**
- Major use: Natural language processing
  - Because huge unlabeled datasets may be available
- Modern trend: Deep learning with
  - Huge dataset
  - Dropout, Batchnorm etc.

Transfer Learning & Domain Adaptation

Transfer Learning

- Training in one setting
  - Data distribution P_1
  - Class labels $y_1(1), y_1(2), \ldots, y_1(m)$
    - e.g., cat, dog

- Generalization in another setting
  - Data distribution P_2
  - Class labels $y_2(1), y_2(2), \ldots, y_2(n)$
    - e.g., pigeon, parrot, sparrow

Transfer Learning

- Training in one setting
  - Data distribution P_1
  - Class labels $y_1(1), y_1(2), \ldots, y_1(m)$
    - e.g., cat, dog
- Generalization in another setting
  - Data distribution P_2
  - Class labels $y_2(1), y_2(2), \ldots, y_2(n)$
    - e.g., pigeon, parrot, sparrow

In classification problems, number of classes in the two settings can be different

Transfer Learning

- Training in one setting
  - Data distribution P_1
  - Class labels $y_1(1), y_1(2), \ldots, y_1(m)$
    - e.g., cat, dog
- Generalization in another setting
  - Data distribution P_2
  - Class labels $y_2(1), y_2(2), \ldots, y_2(n)$
    - e.g., pigeon, parrot, sparrow

The underlying assumption is that the data in the two settings share some common characteristics

e.g., different animals and birds can have similar lower level details, such as, edges and textures

Transfer Learning

- Training in one setting
  - Data distribution P_1
  - Class labels $y_1(1), y_1(2), \ldots, y_1(m)$
    - e.g., cat, dog

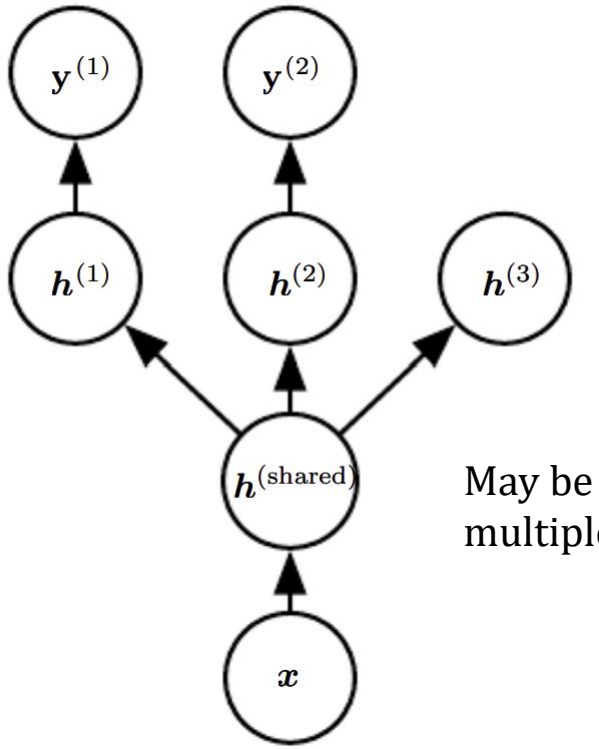
- Generalization in another setting
  - Data distribution P_2
  - Class labels $y_2(1), y_2(2), \ldots, y_2(n)$
    - e.g., pigeon, parrot, sparrow

The parameters learnt in the lower layers for the first task may be helpful in extracting meaningful lower-level representation for the second task also

Transfer Learning

- Training in one setting
  - Data distribution P_1
  - Class labels $y_1(1), y_1(2), \ldots, y_1(m)$
    - e.g., cat, dog
- Generalization in another setting
  - Data distribution P_2
  - Class labels $y_2(1), y_2(2), \ldots, y_2(n)$
    - e.g., pigeon, parrot, sparrow

So, lower layers may be common in such situations



May be multiple layers

Transfer Learning

- Usually we expect the inputs across various tasks have some shared characteristics

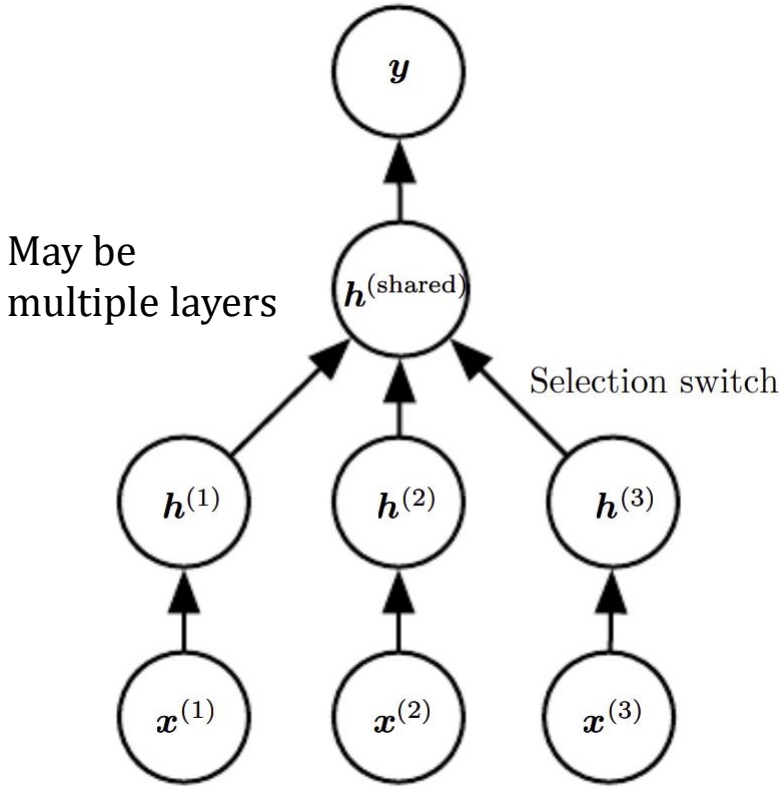
- However, in some cases, the output semantics may be common across tasks
  - e.g., a video captioning system that take various videos as input and produce captions
    - In task 1: it may be trained with videos of football
    - In task 2: it may see videos of astronauts

Transfer Learning

- However, in some cases, the output semantics may be common across tasks
  - e.g., a video captioning system that take various videos as put and produce captions
    - In task 1: it may be trained with videos of football
    - In task 2: it may see videos of astronauts
- In this case, the characteristics of the layers near the output layer should be similar
  - Upper layers for the two tasks may be common

Transfer Learning

- In this case, the characteristics of the layers near the output layer should be similar
  - Upper layers for the two tasks may be common



Domain Adaptation

- Task remains the same in two (or more) settings
- The distribution of the data in two settings change
- For example
  - Identification of buildings from
    - Drone images
    - Satellite images

Domain Adaptation

- Task remains the same in two (or more) settings
- The distribution of the data in two settings change
- For example
  - Identification of buildings from
    - Drone images
    - Satellite images

https://www.istockphoto.com/photos/drone-building
https://www.researchgate.net/figure/Satellite-view-of-the-mixed-use-building_fig121_265787339

Domain Adaptation

- There may be an underlying function in the NN that identifies buildings
- Often, unsupervised pre-training helps

Concept Drift

- Gradual Changes in data distribution over time
- The goal here also is to extract meaningful information from the data of the first task to use it in the second task
- Example: a model for rainfall prediction

Extreme Forms of Transfer Learning

- One-shot learning
  - Only one annotated example (per class for classification problem) of task 2
- Zero-shot learning
  - No annotated example of task 2
  - Auxiliary information task 2

Zero-shot Learning

- Seen classes: during training, we have
  - Data X
  - Class label Y
  - Auxiliary information A
- Unseen classes: during training, we have
  - Only auxiliary information A

Zero-shot Learning

- At test time, we have data from both the seen and the unseen classes
- Task: classify the incoming data
- Goal during training: Learning the common characteristics from the seen classes
  - Then do transfer learning to the unseen classes

Have You Seen An Elephant Shrew/ Sengi

- A mammal
- Body is similar to that of a rat
- Head is a miniature version of that of anteater

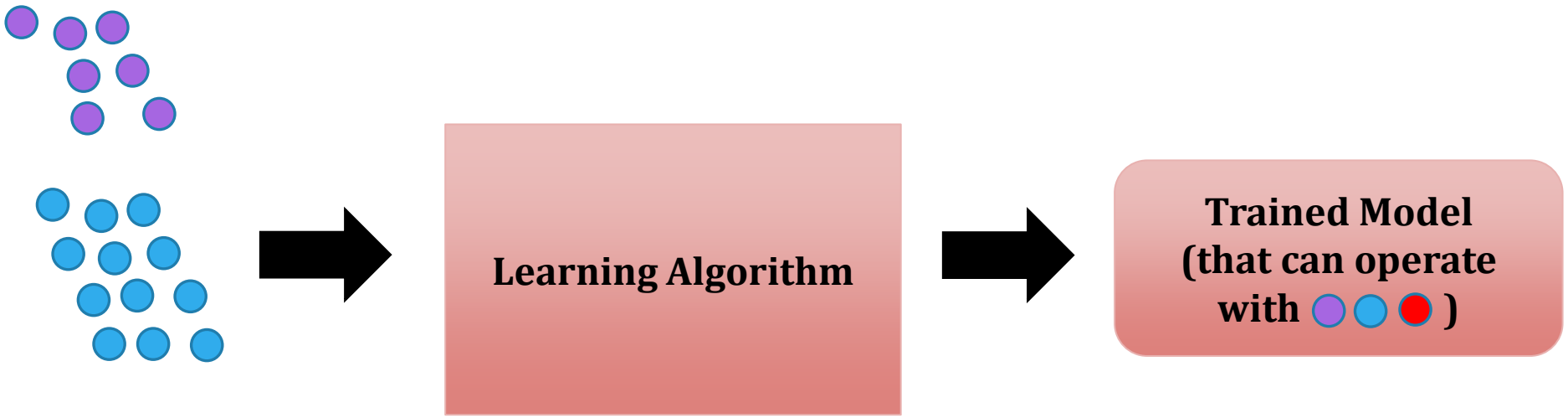
A Rat

An Anteater

Sengi !

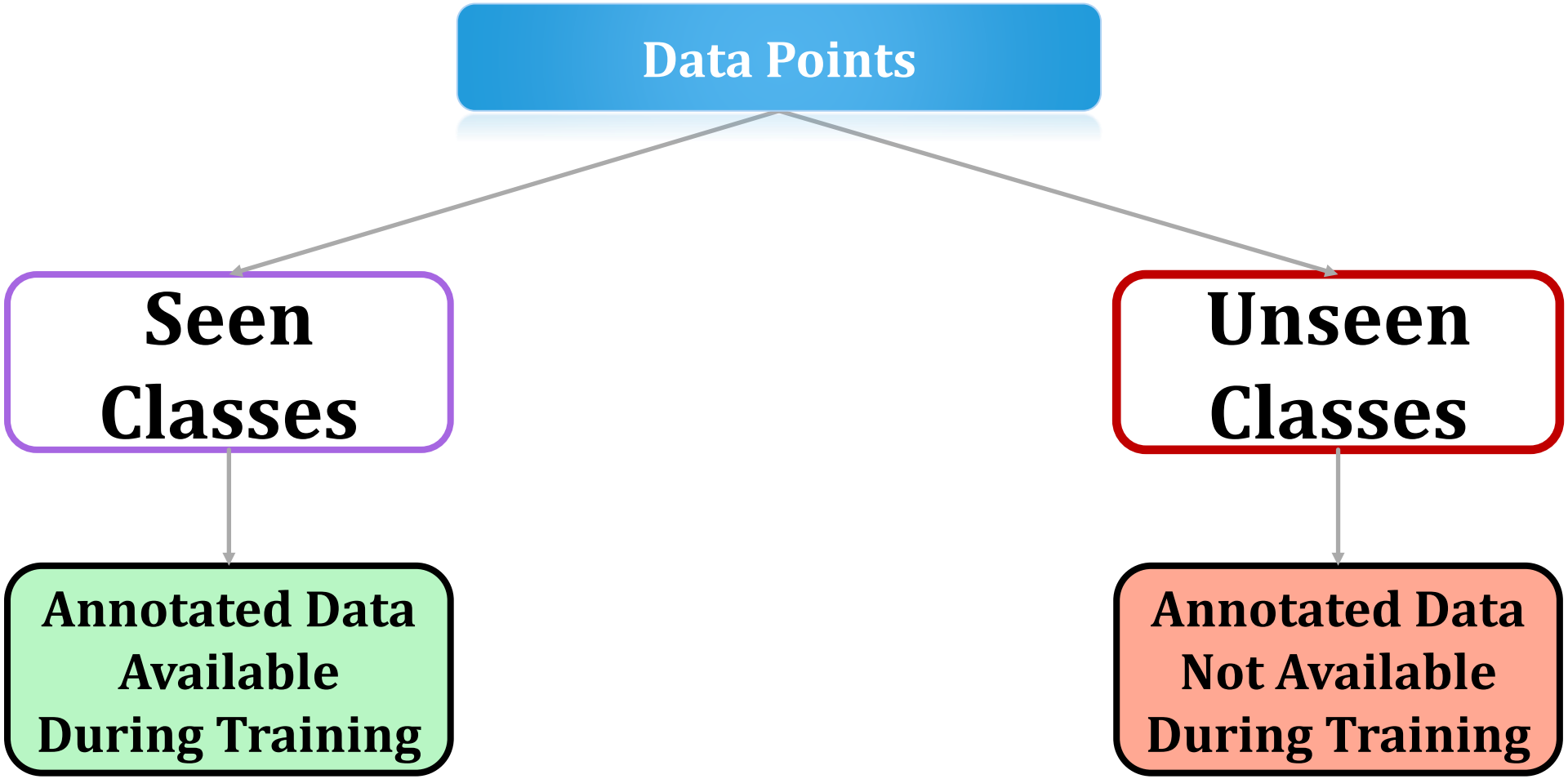
Zero-shot Learning

Zero shot learning: Training

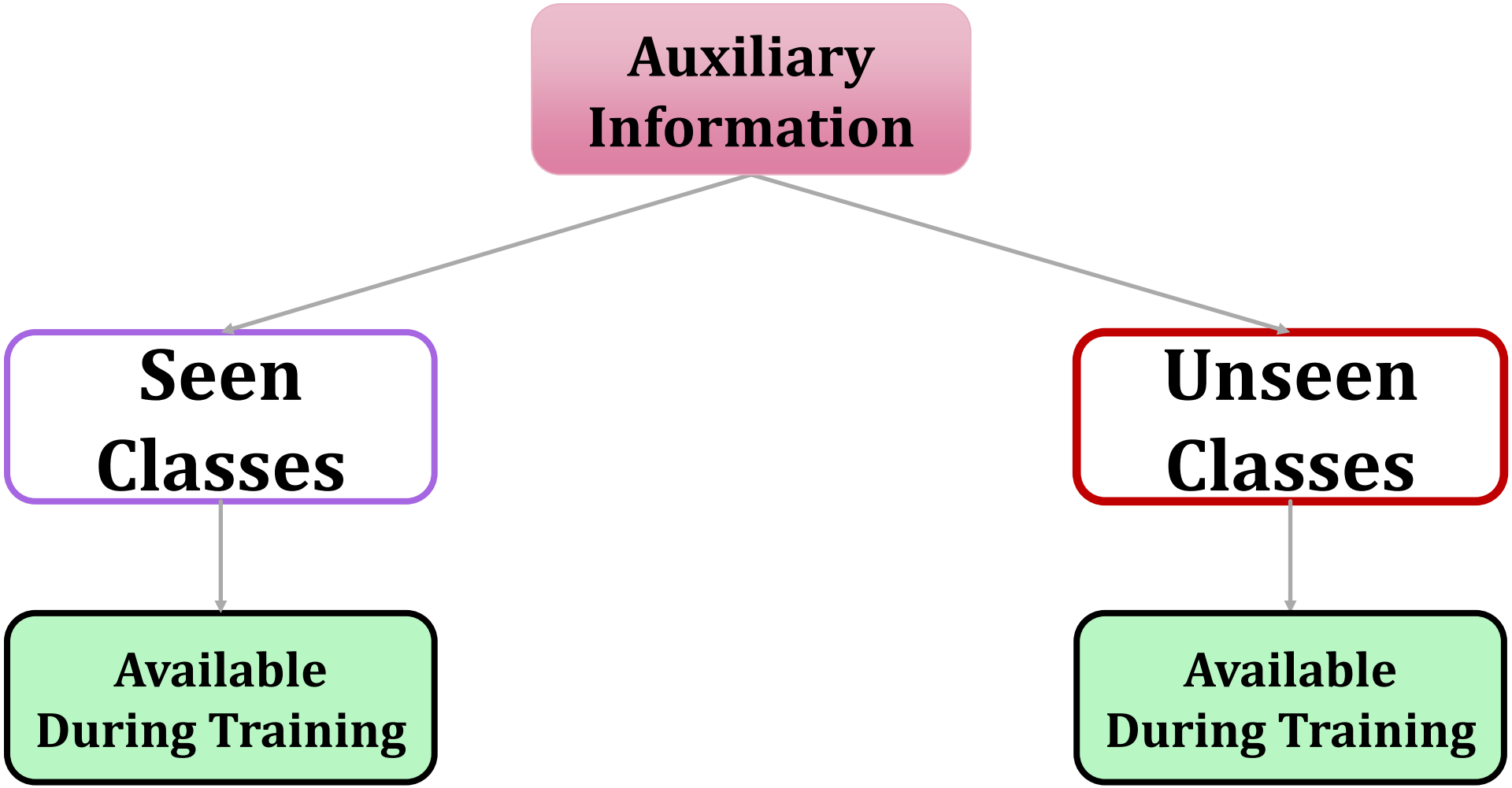


Zero shot learning: Testing

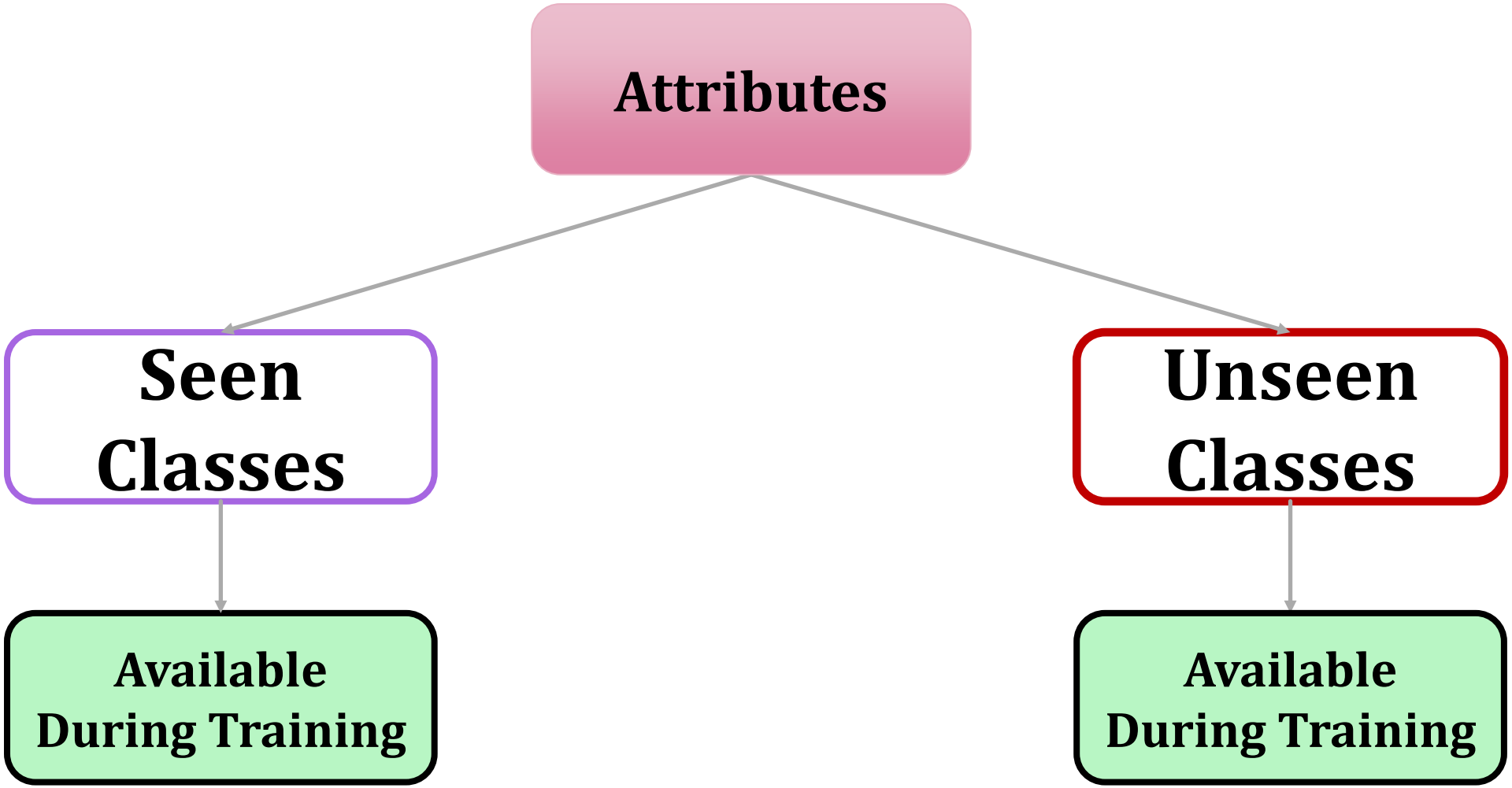
Zero-shot Learning



Zero-shot Learning



Zero-shot Learning



Attributes

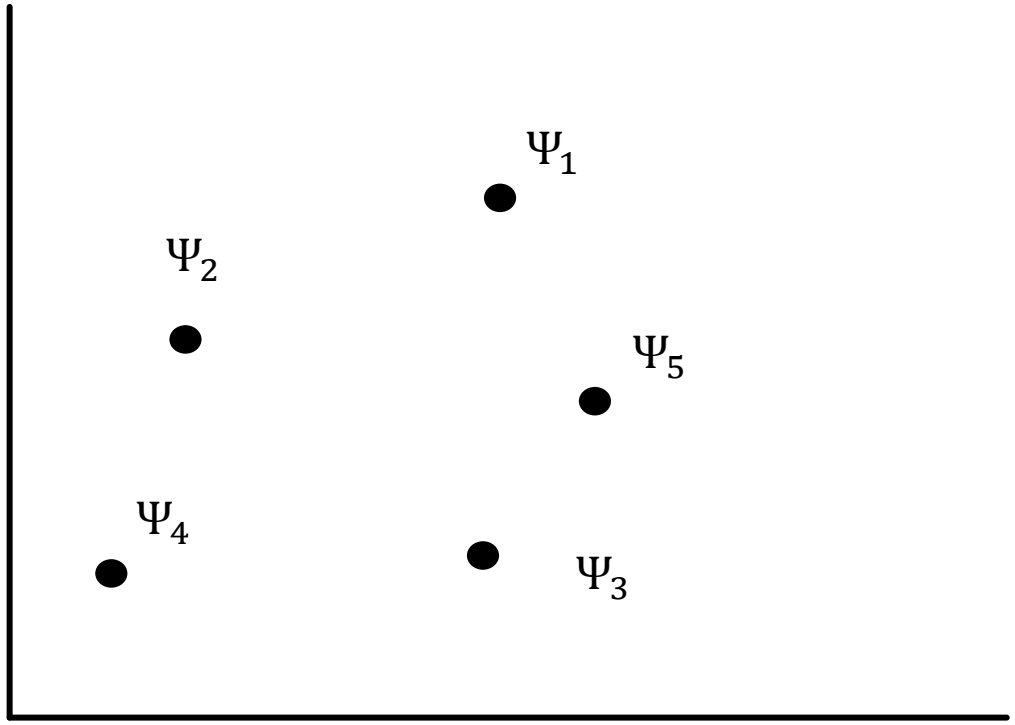
- Higher level information of an object
  - Color of an object: red/green etc
  - Shape: round/ square
  - Texture, etc.
- According to [1], we call a property of an object an attribute, if a human has the ability to decide whether the property is present or not for a certain object
- Attributes can provide auxiliary information

[1] “Attribute-Based Classification for Zero-Shot Visual Object Categorization”, Lampert *et al.,* TPAMI, 2014

Methods for Zero-Shot Classification

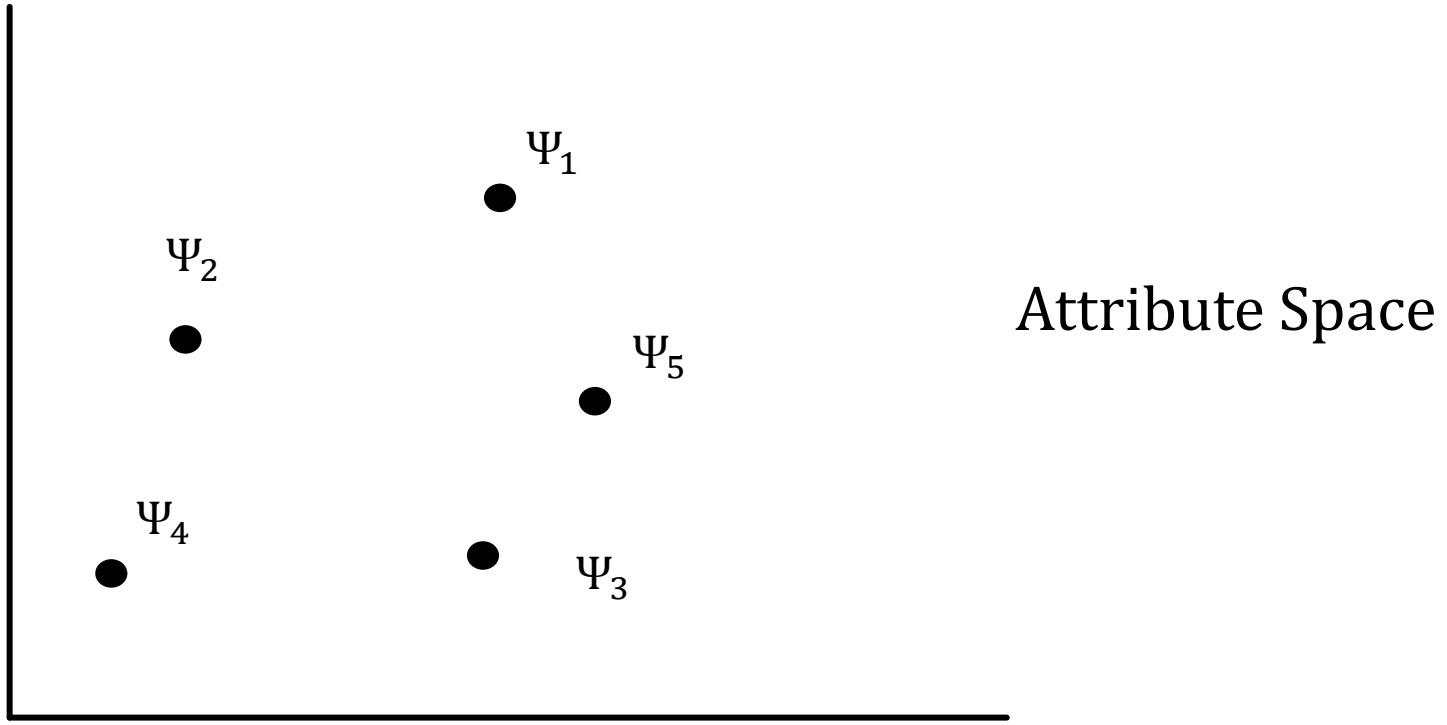
Two-Stage Approaches

- Get the attribute signatures for seen and unseen classes



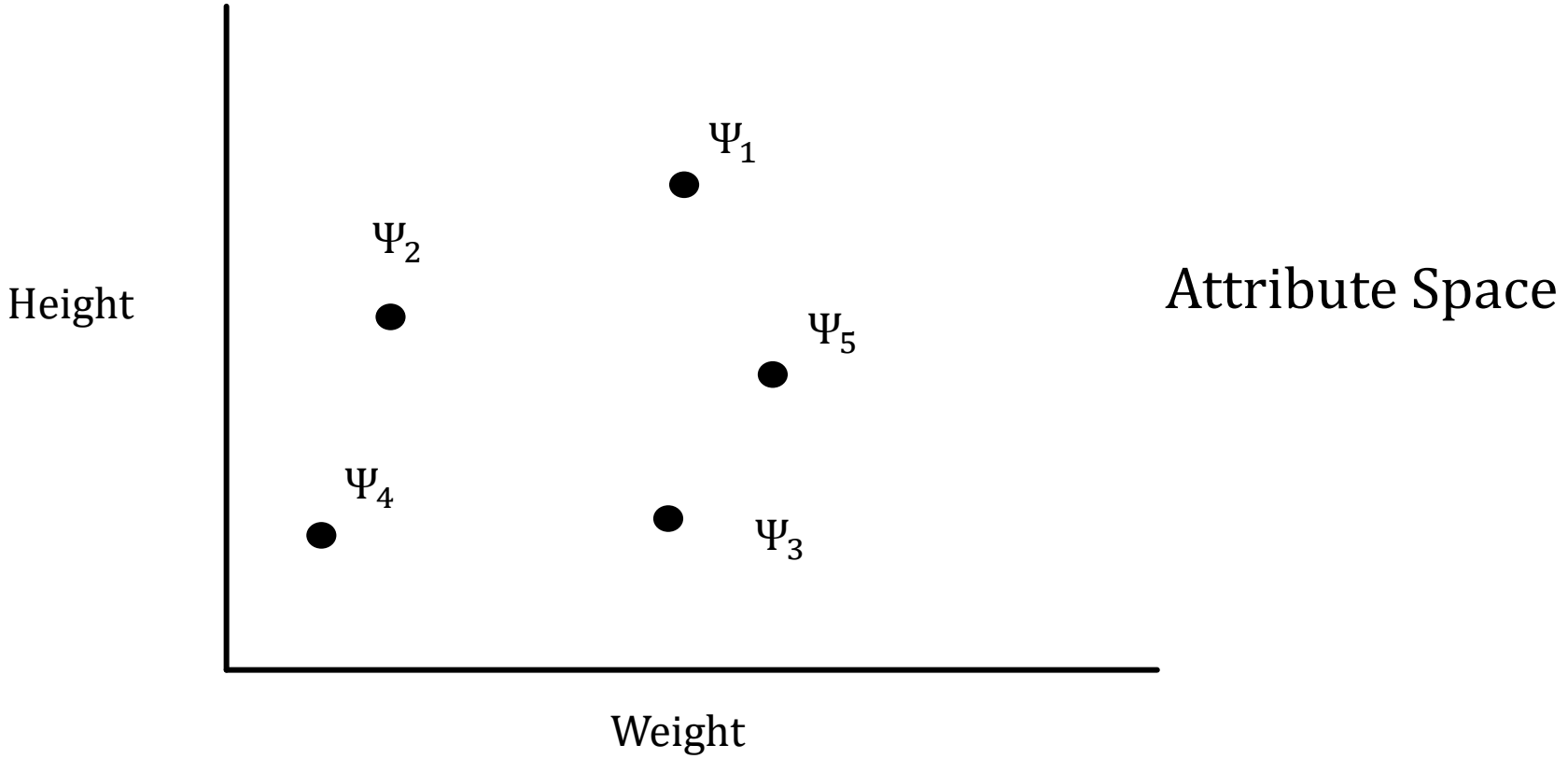
Two-Stage Approaches

- Get the attribute signatures for seen and unseen classes



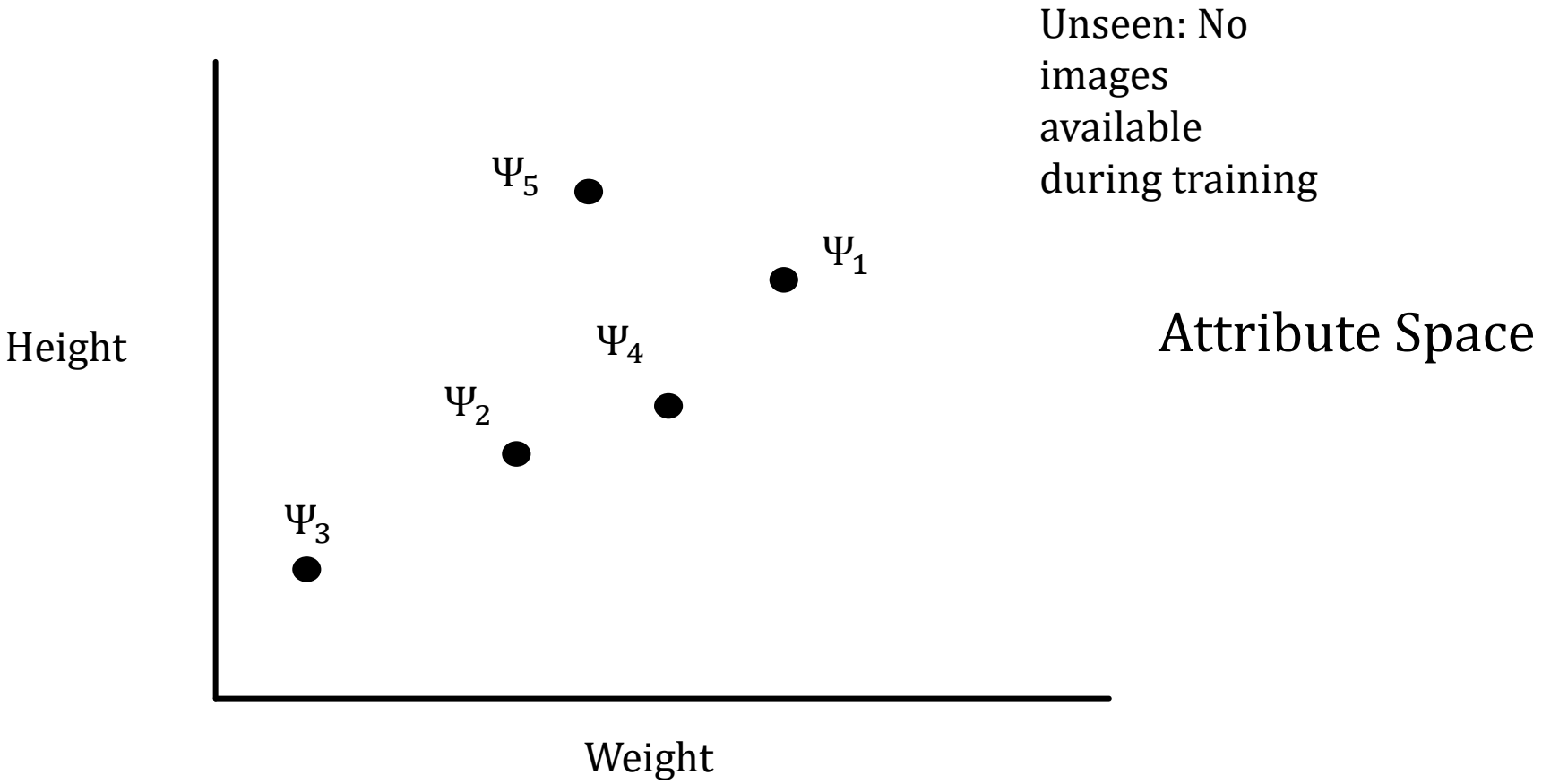
Two-Stage Approaches

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



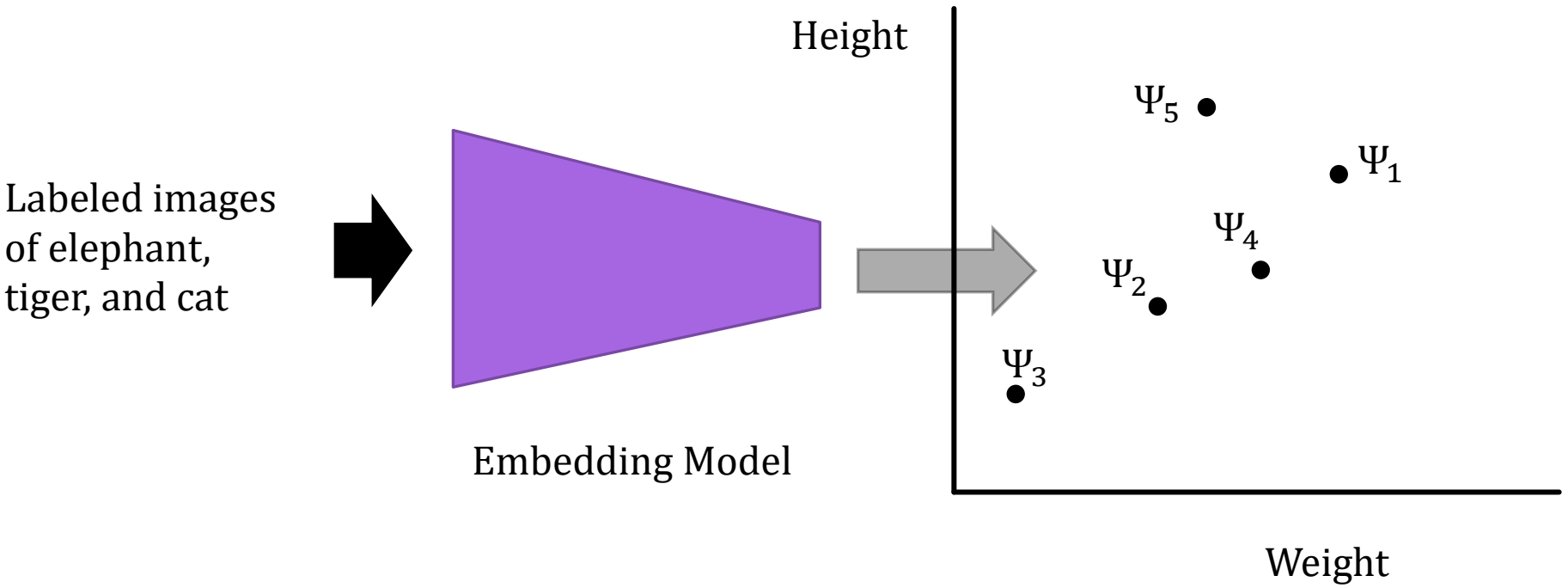
Two-Stage Approaches

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



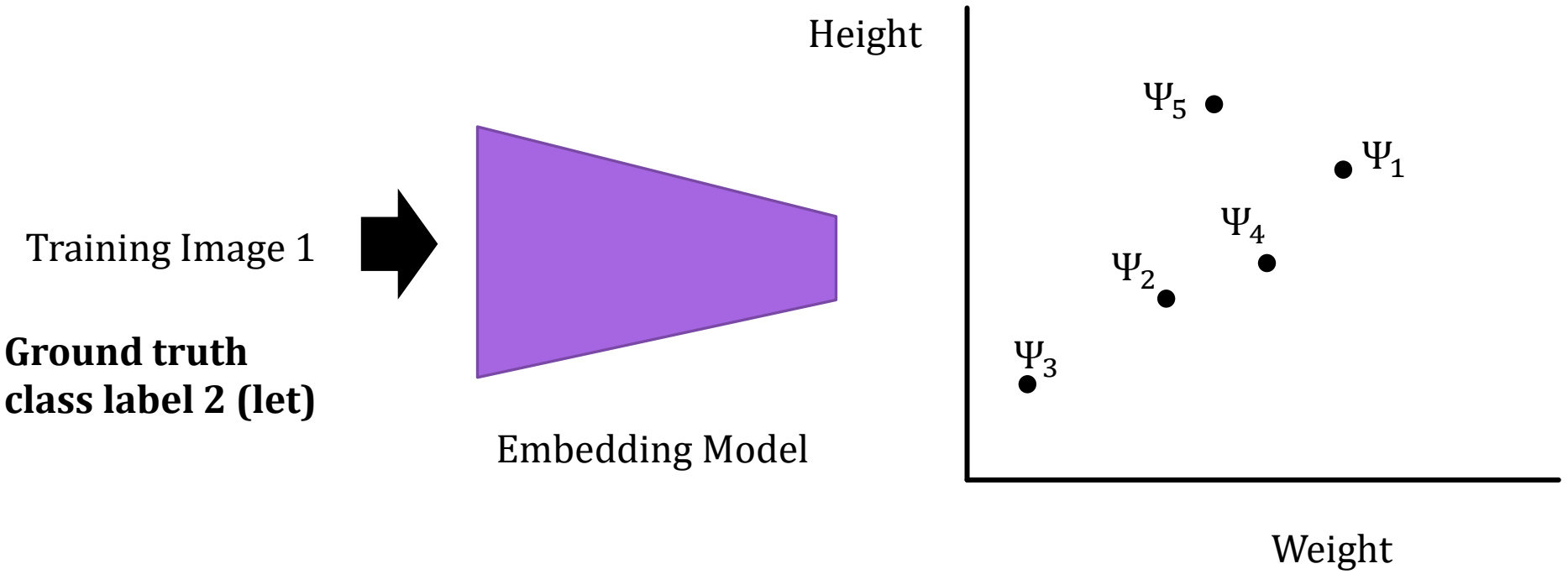
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



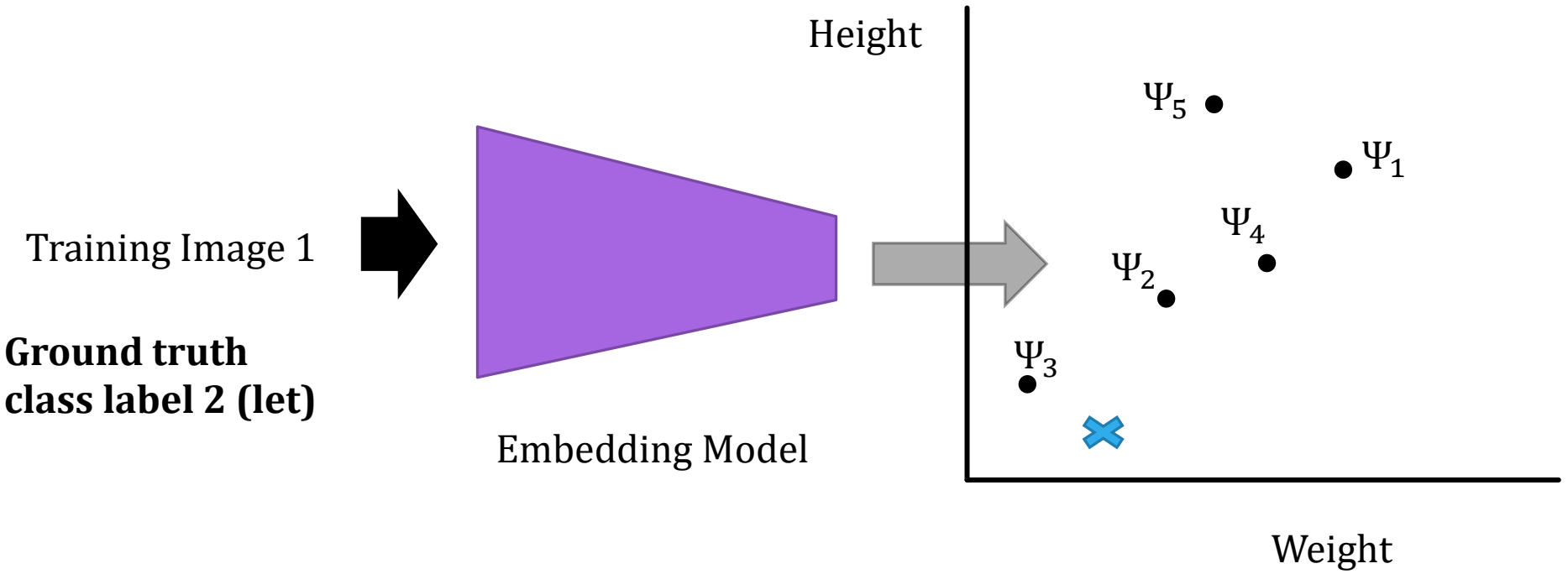
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



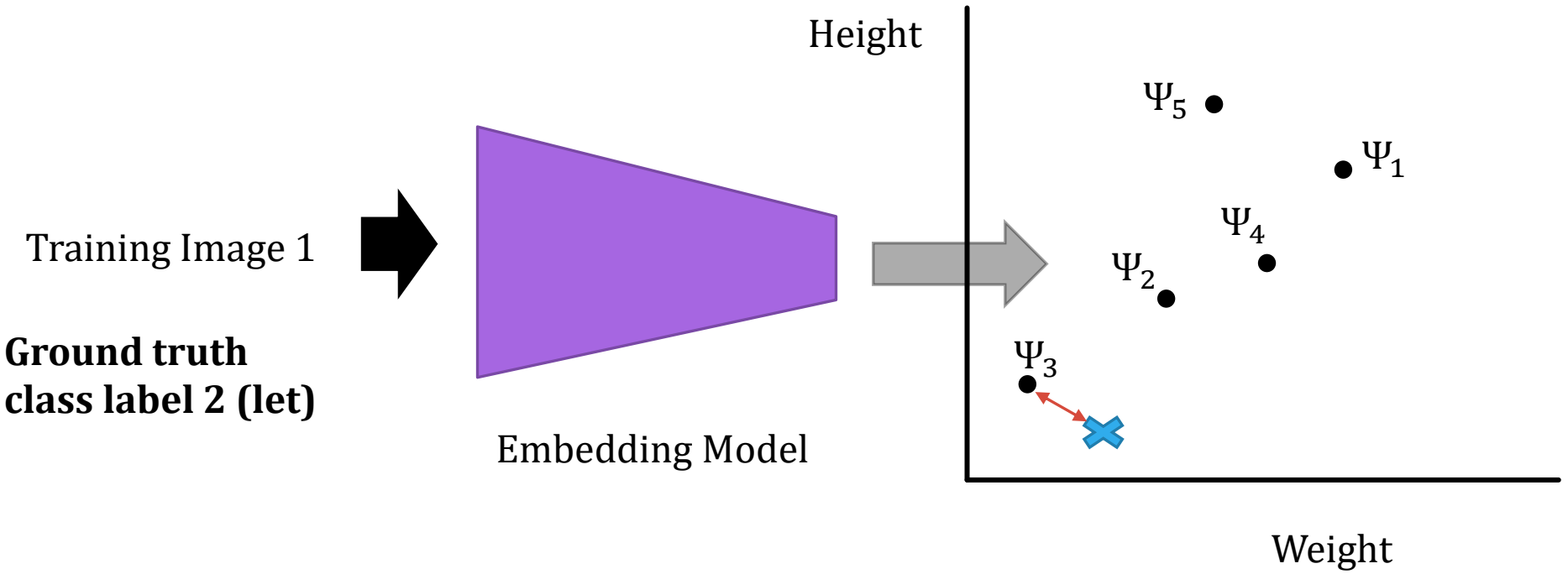
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



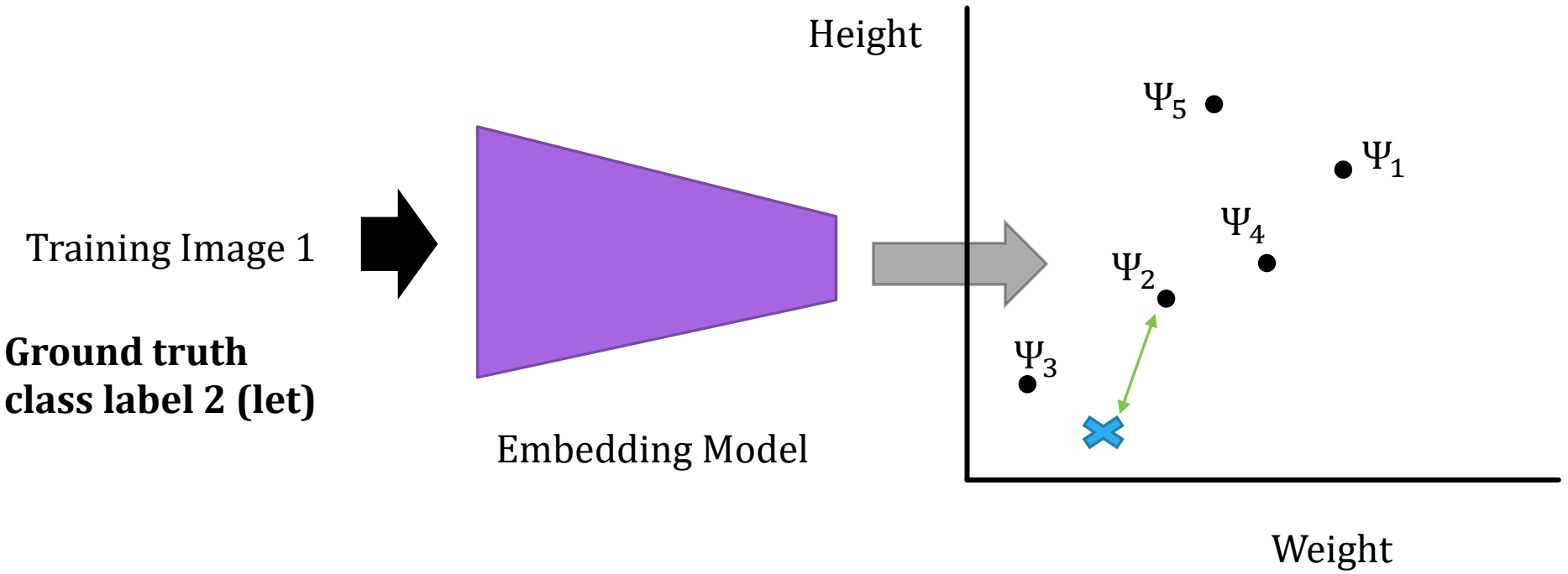
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



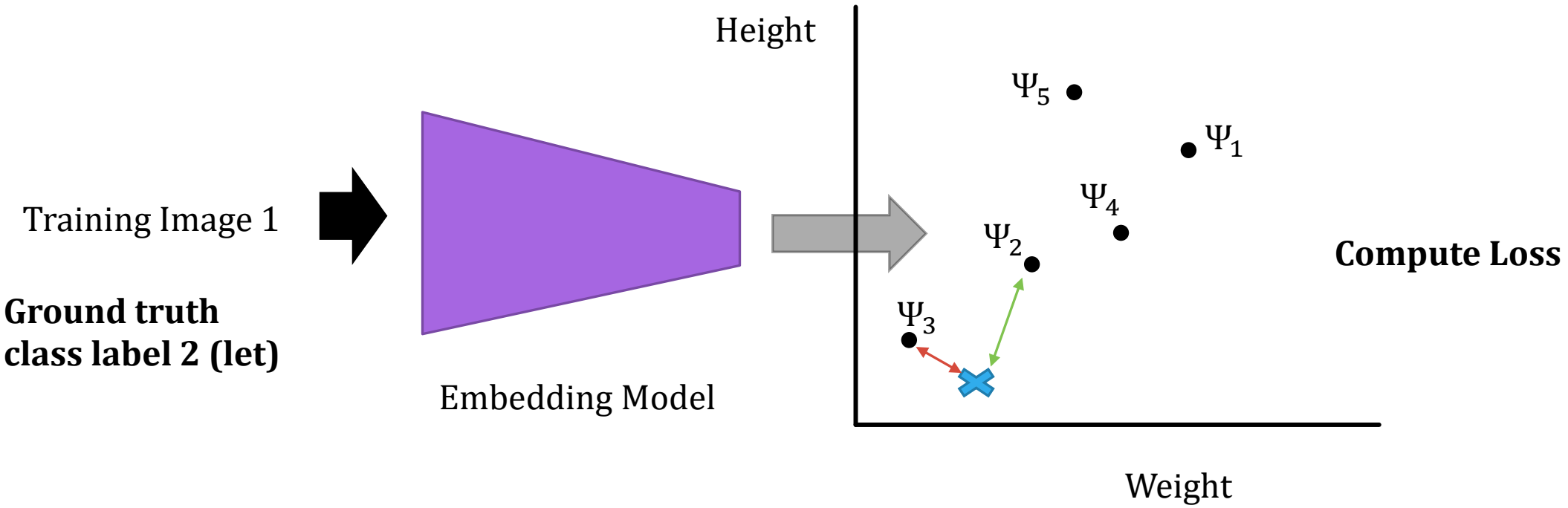
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



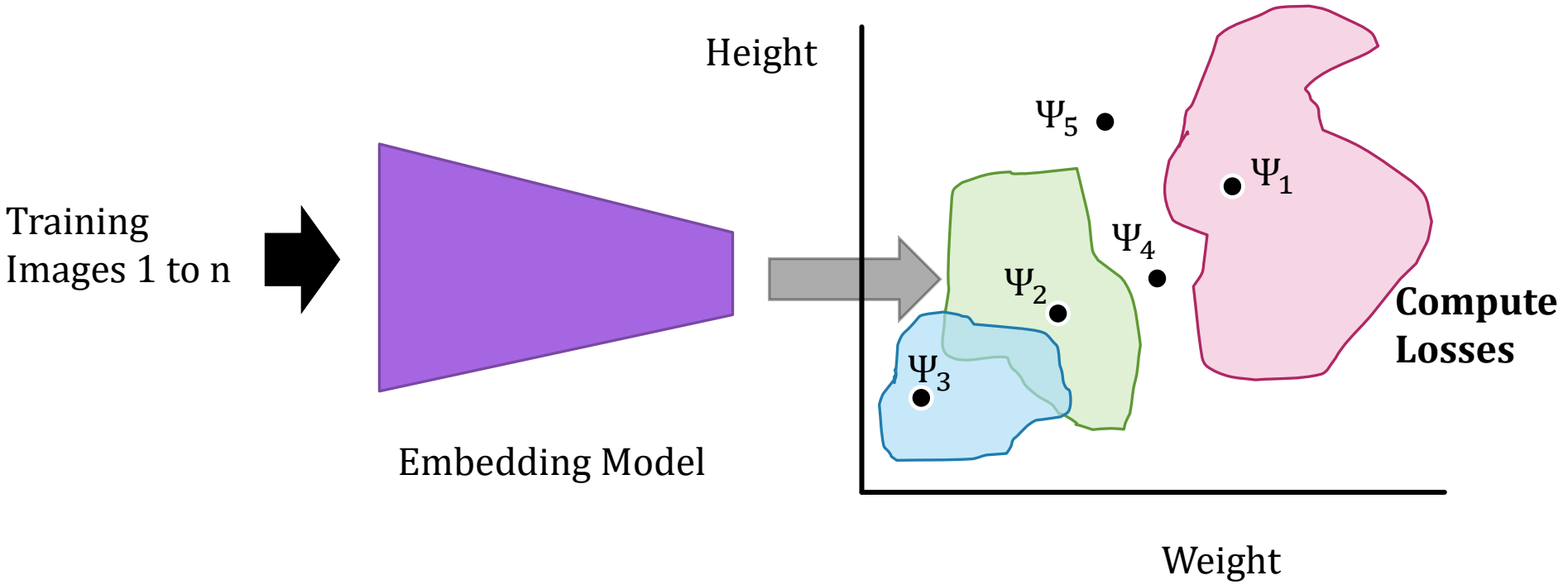
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



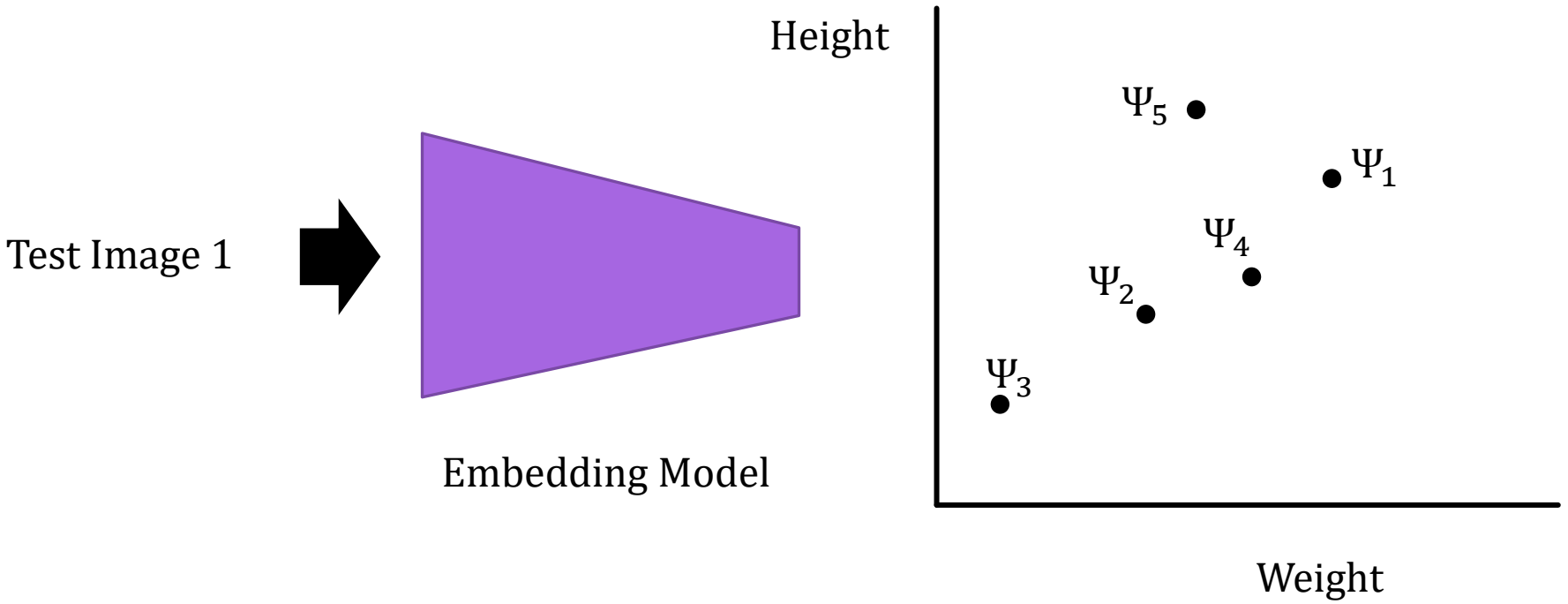
Two-Stage Approaches: Training

- Get the attribute signatures for seen and unseen classes
  - For example: classification of animals (elephant, tiger, cat, rhino, giraffe)



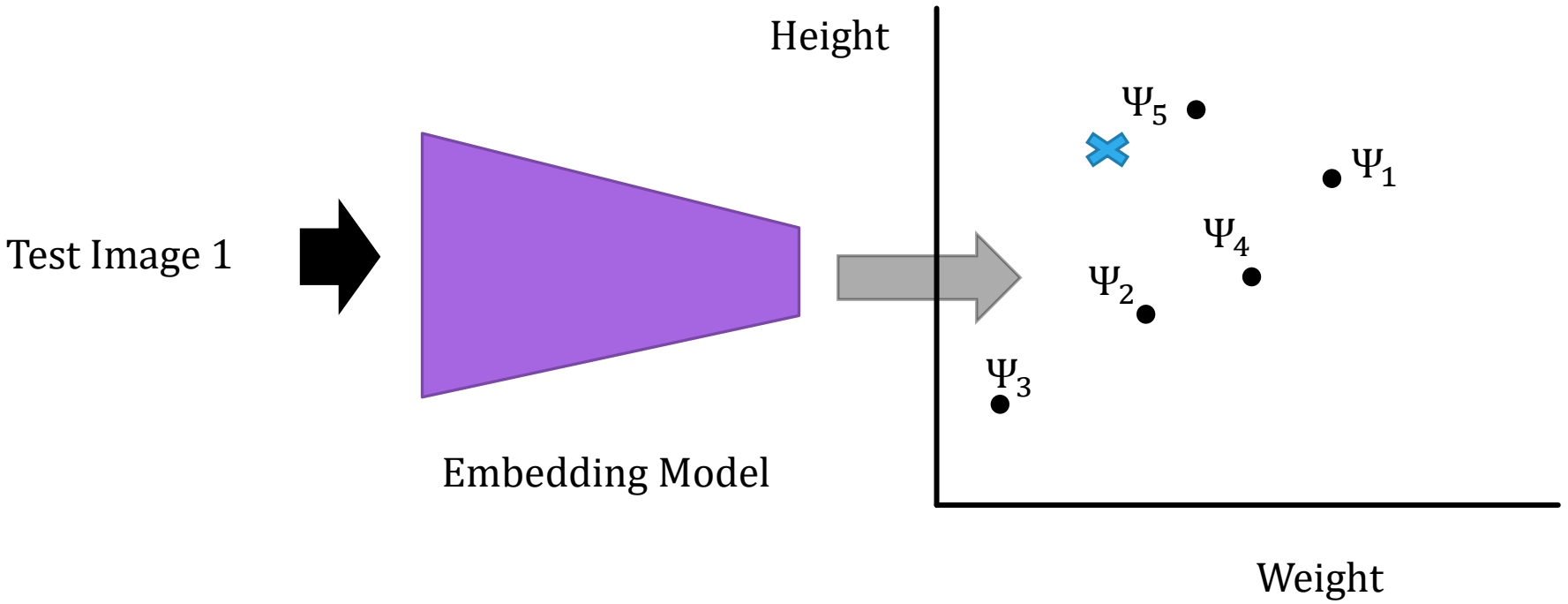
Two-Stage Approaches: Testing (Scenario 1)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to only rhino, giraffe (unseen classes)



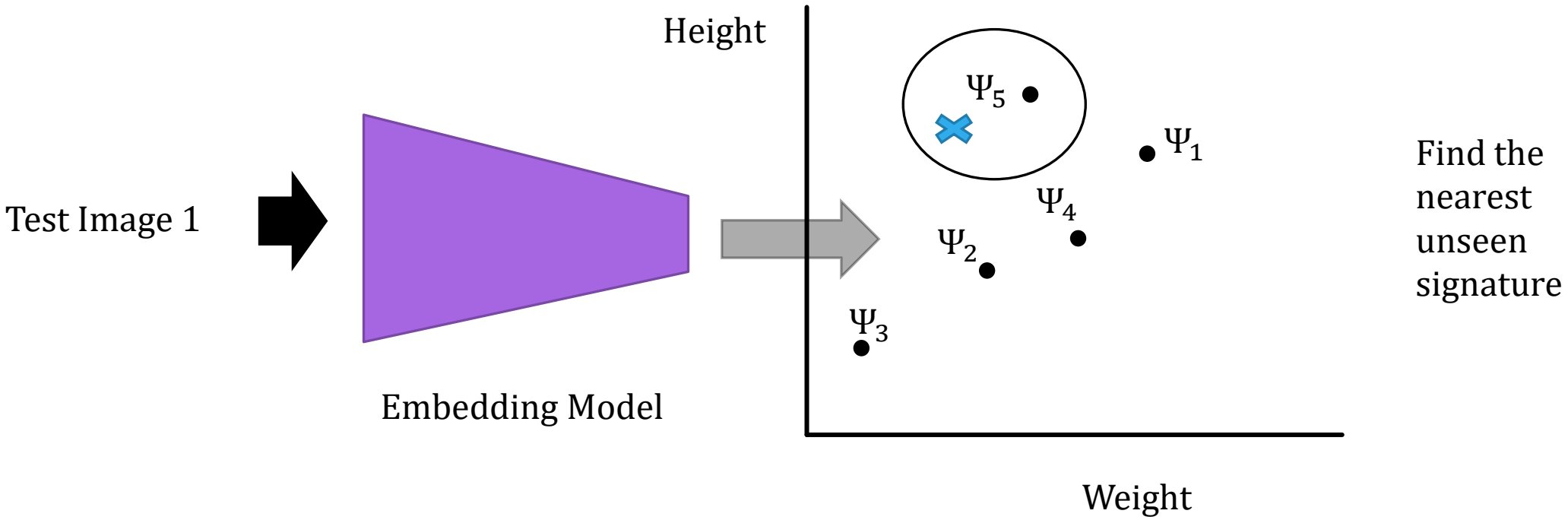
Two-Stage Approaches: Testing (Scenario 1)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to only rhino, giraffe (unseen classes)



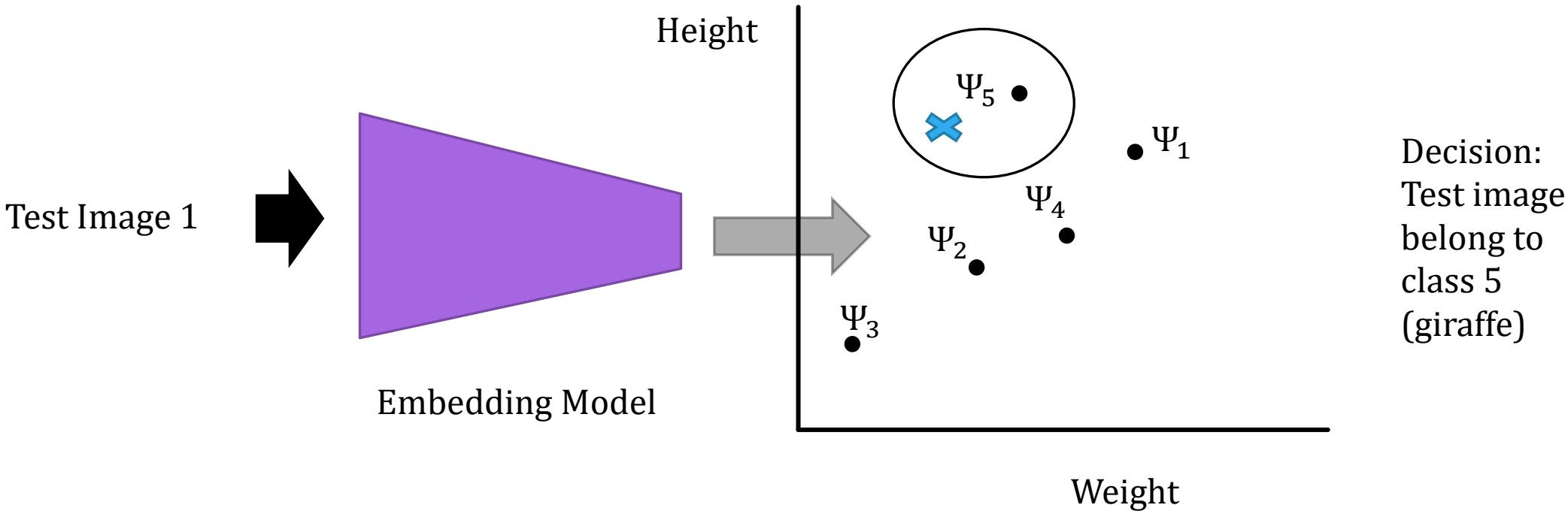
Two-Stage Approaches: Testing (Scenario 1)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to only rhino, giraffe (unseen classes)



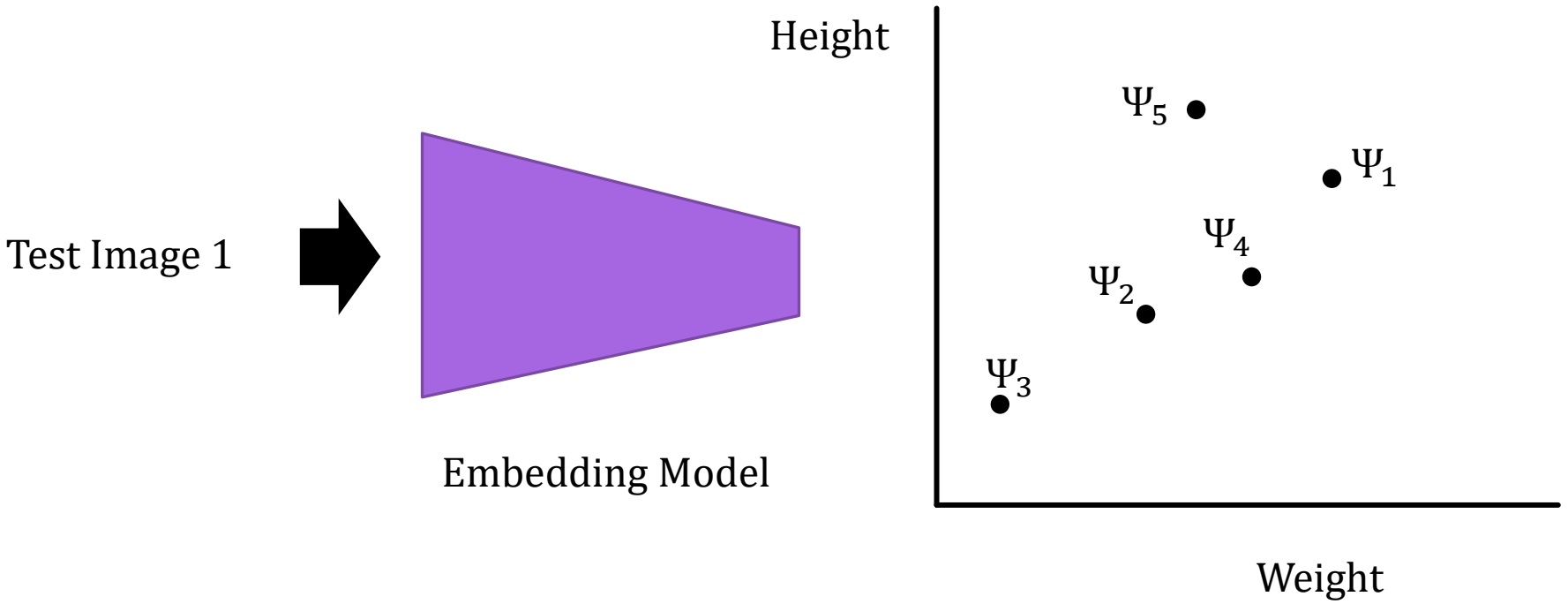
Two-Stage Approaches: Testing (Scenario 1)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to only rhino, giraffe (unseen classes)



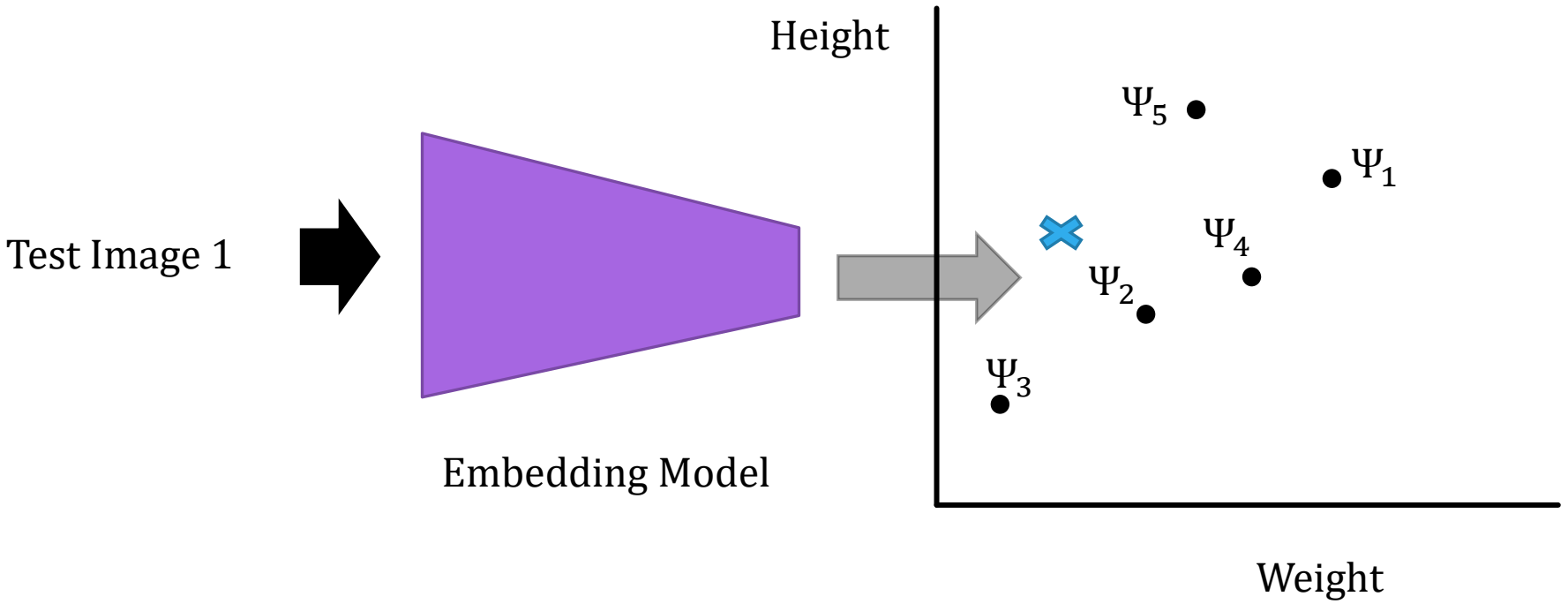
Two-Stage Approaches: Testing (Scenario 2)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to any of the 5 classes: elephant, tiger, cat, rhino, giraffe



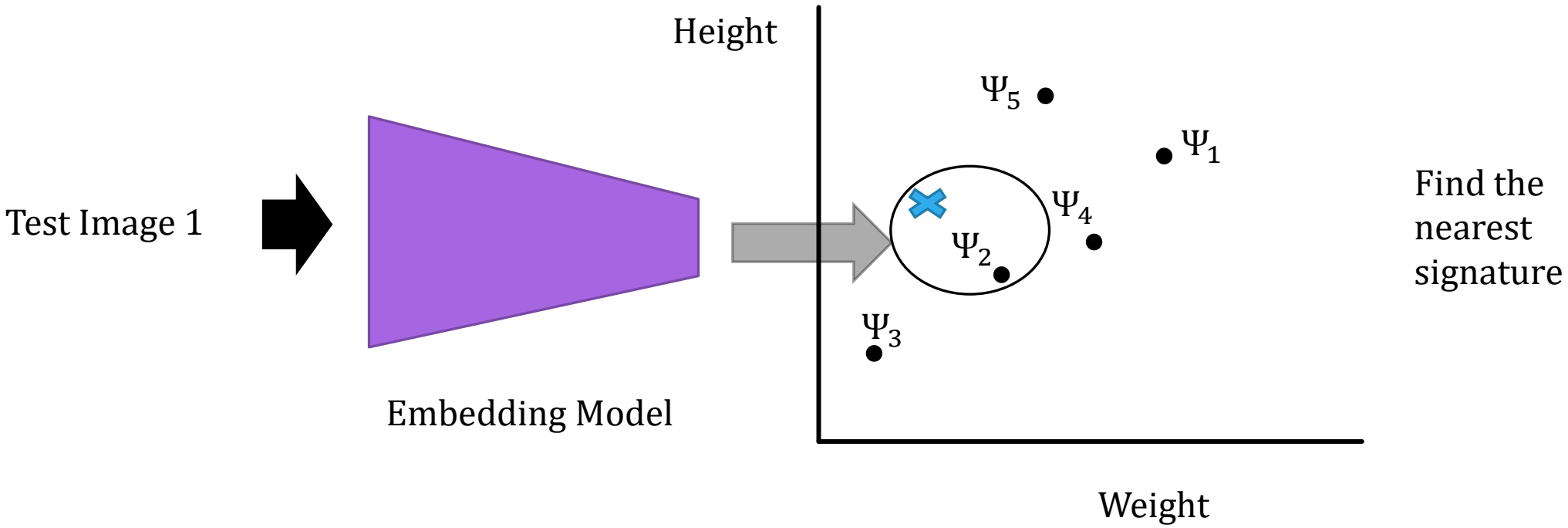
Two-Stage Approaches: Testing (Scenario 2)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to any of the 5 classes: elephant, tiger, cat, rhino, giraffe



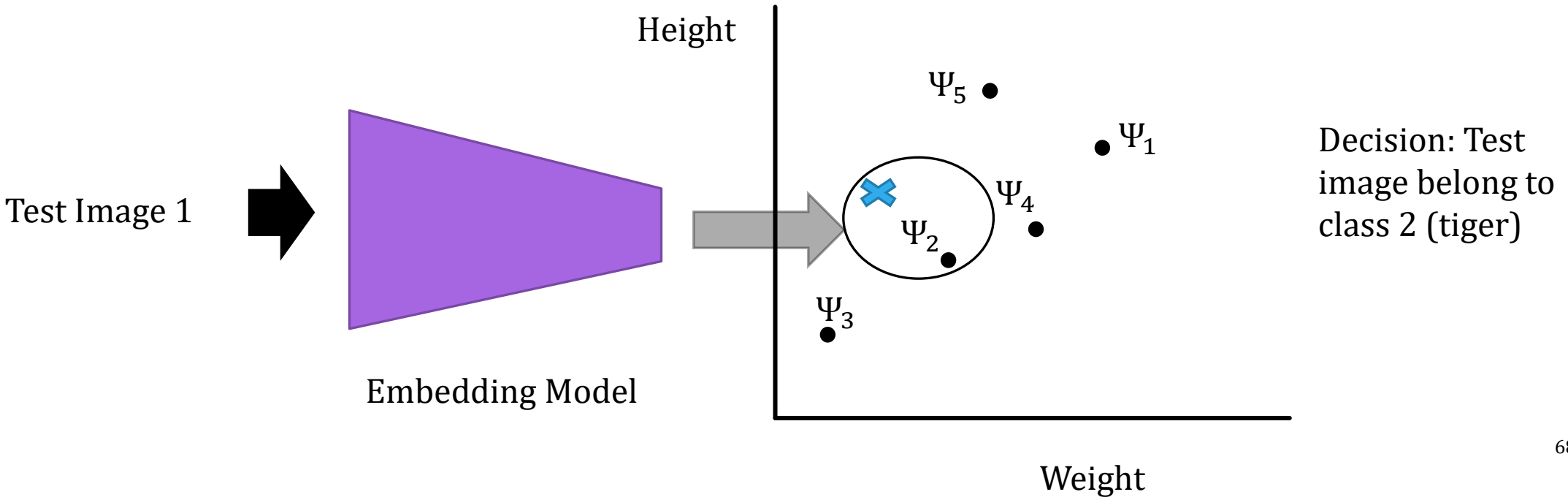
Two-Stage Approaches: Testing (Scenario 2)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to any of the 5 classes: elephant, tiger, cat, rhino, giraffe



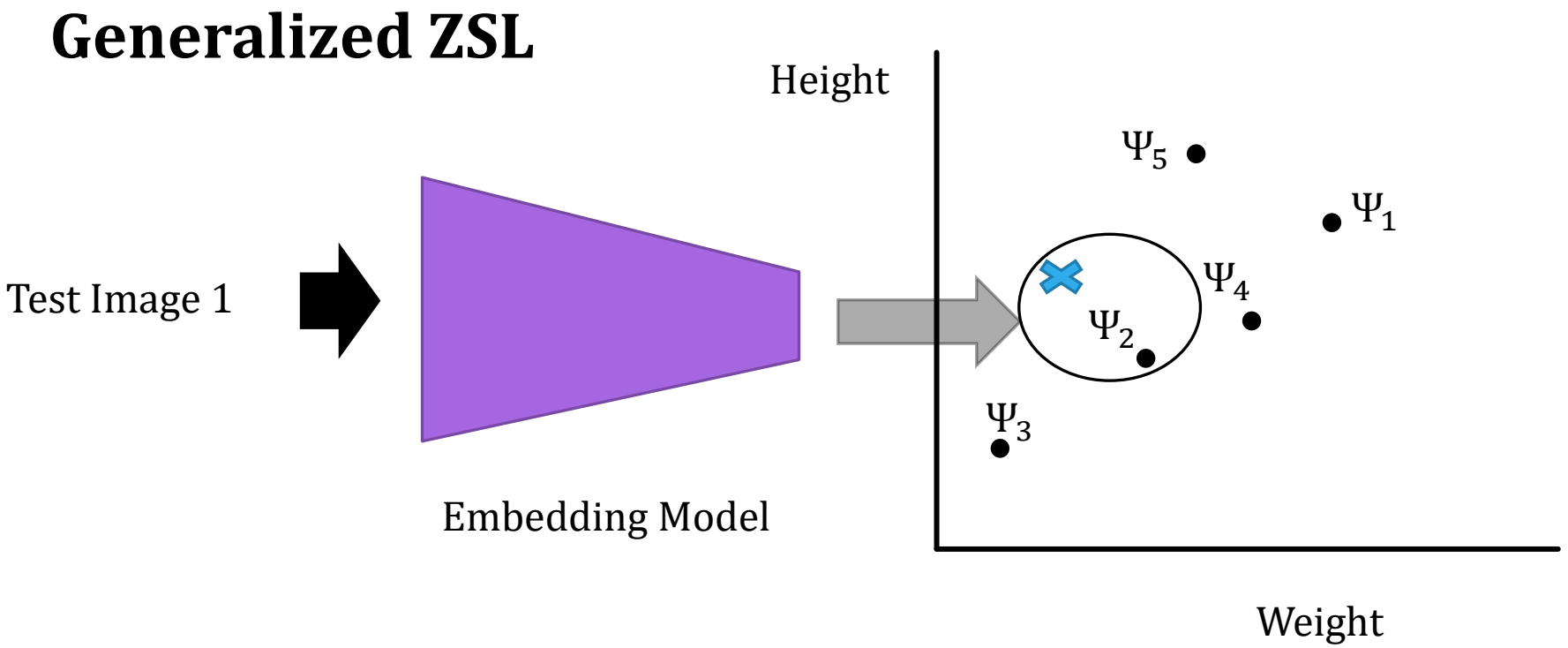
Two-Stage Approaches: Testing (Scenario 2)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to any of the 5 classes: elephant, tiger, cat, rhino, giraffe



Two-Stage Approaches: Testing (Scenario 2)

- Get the attribute signatures for seen and unseen classes
  - Test images may belong to any of the 5 classes: elephant, tiger, cat, rhino, giraffe



Zero-shot Learning