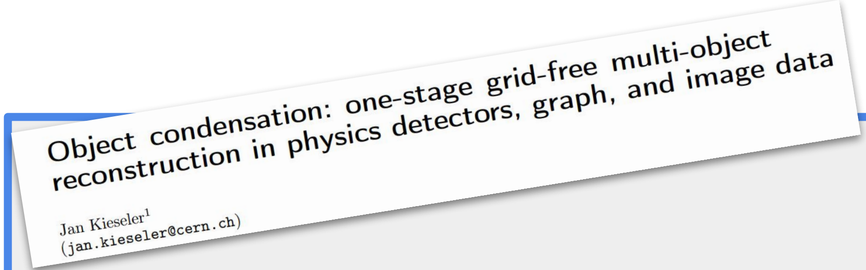
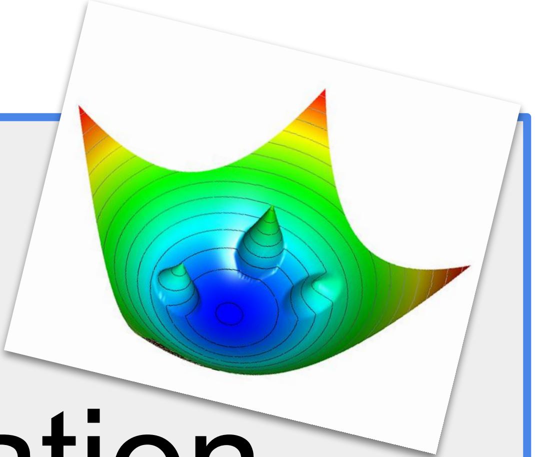

Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data

Jan Kieseler[1]
(jan.kieseler@cern.ch)



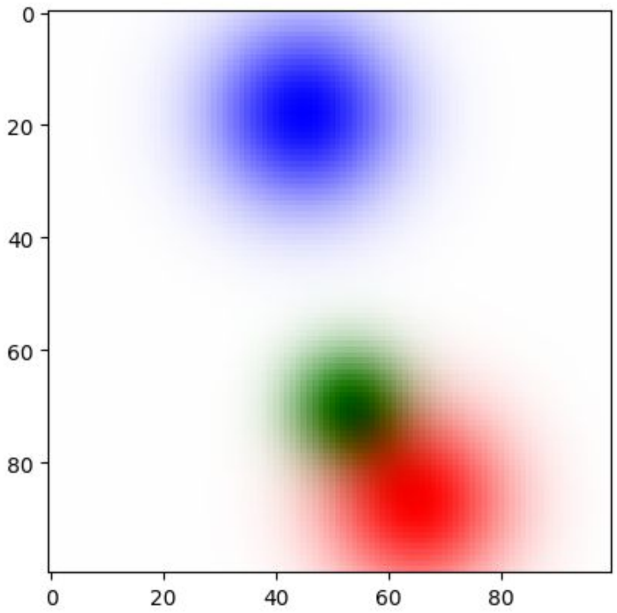
Object Condensation

GSS 1-26-2024

Hypothetical **(A)**

Given… A Pixel grid (100x100x3)
…write code from scratch that…

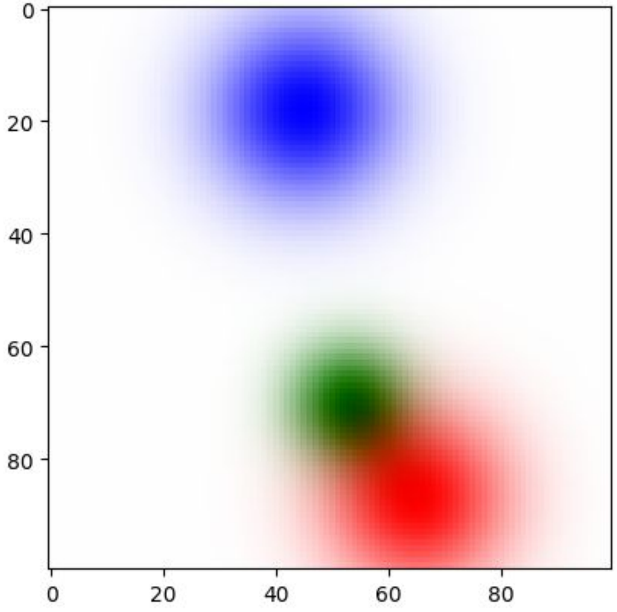
Predicts… The number of unique objects and their color



Hypothetical **(A)**

Given… A Pixel grid (100x100x3)
…write code from scratch that…

Predicts… The <u>number</u> of unique objects and their <u>color</u>

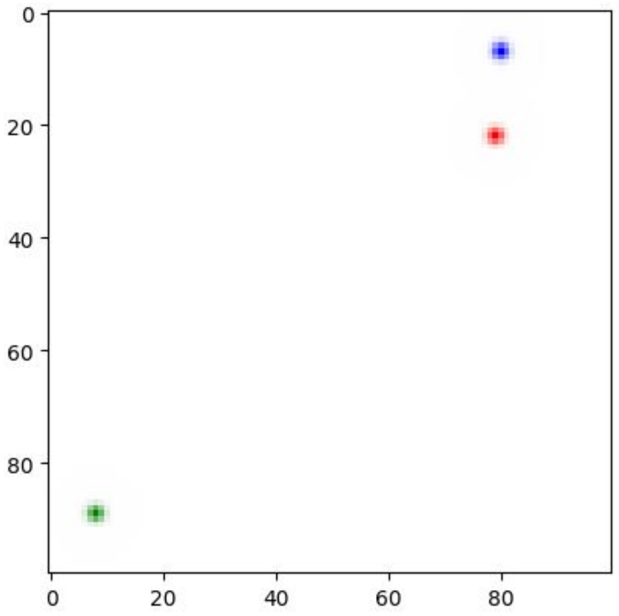


Challenges:

- Object may be only partially visible within the volume
- Program must work for an arbitrary # of visible objects
- How to/Should we deal with empty cells?
- Regions of overlap
- Noise?

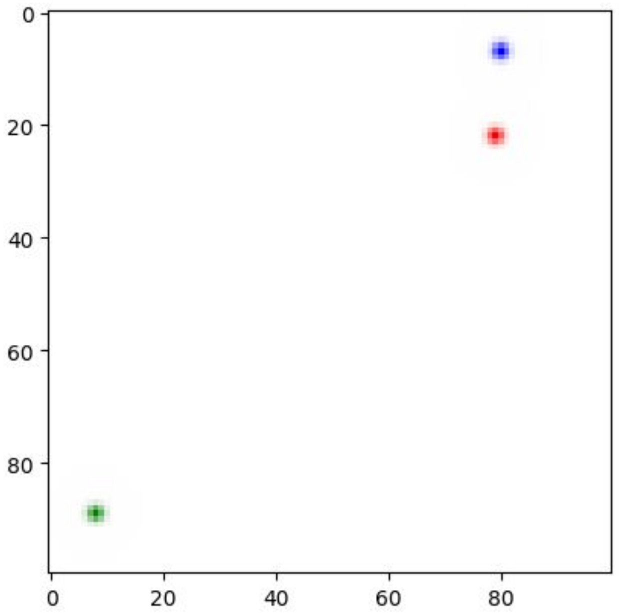
Hypothetical **(B)**

Given… A Pixel grid (100x100x3)
Guarantee… No overlap, only 1 “bright” pixel per object
Predicts… The number of unique objects and their color



Hypothetical **(B)**

Given… A Pixel grid (100x100x3)
Guarantee… No overlap, only 1 “bright” pixel per object
Predicts… The <u>number</u> of unique objects and their <u>color</u>



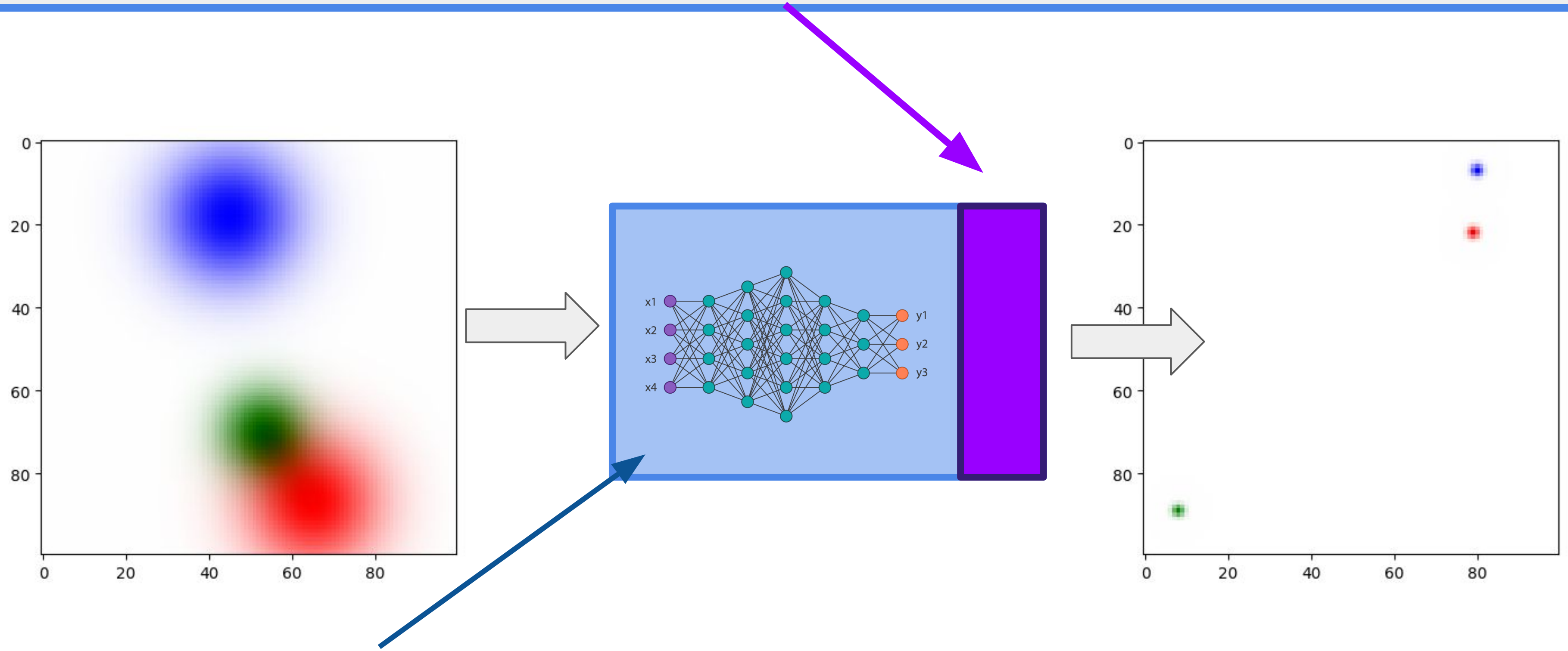
<u>Solution becomes much simpler to picture…</u>

… threshold away dim pixels ($\beta < 0.5$) …

… count number of remaining bright pixels …

… read off their colors …

★Object Condensation★ Foreshadowing



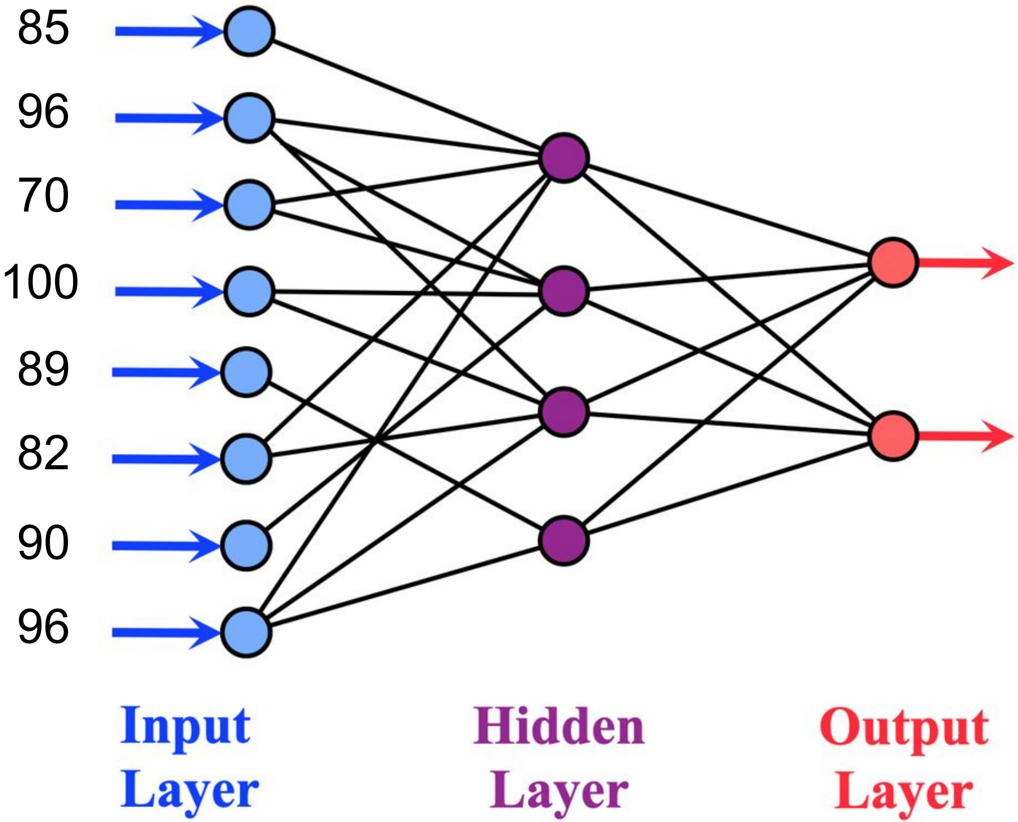
Neural Network, CNN, GNN, etc…

Supervised Machine Learning

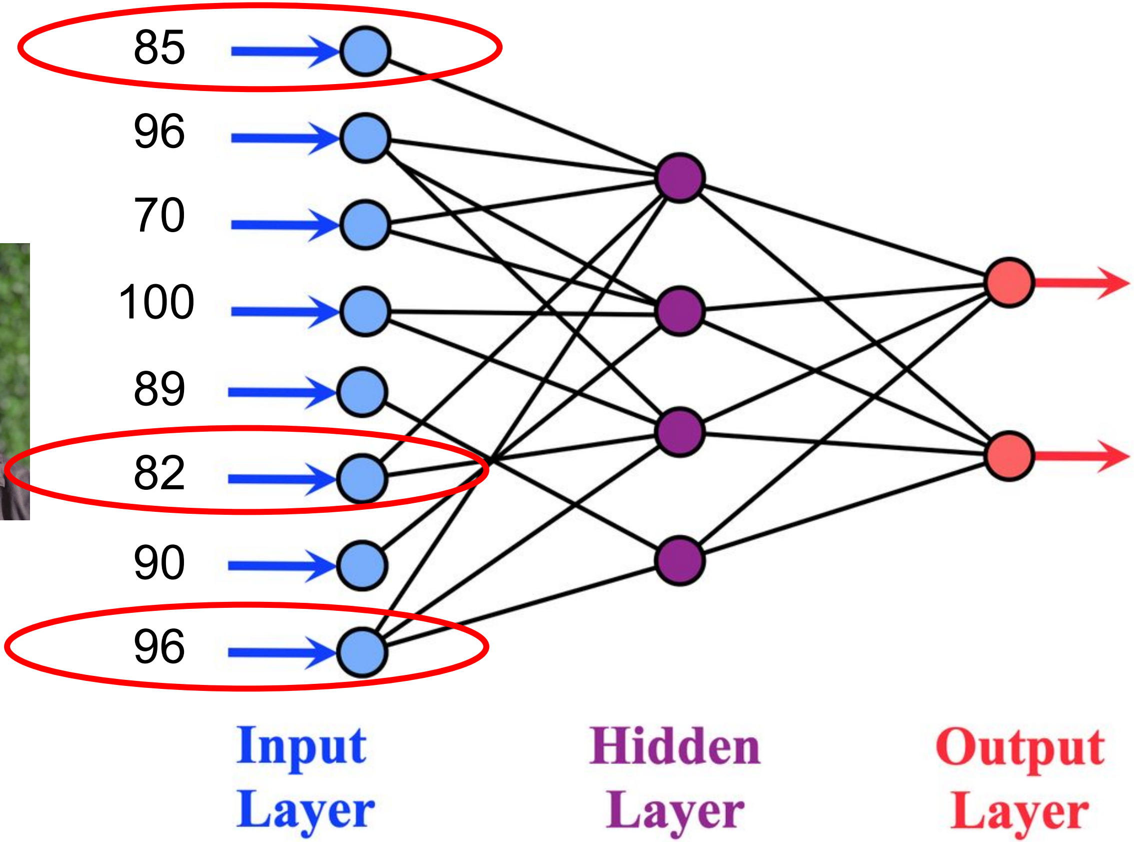
Given
8 quiz grades

Predict
1. Midterm grade
2. Physics Major?

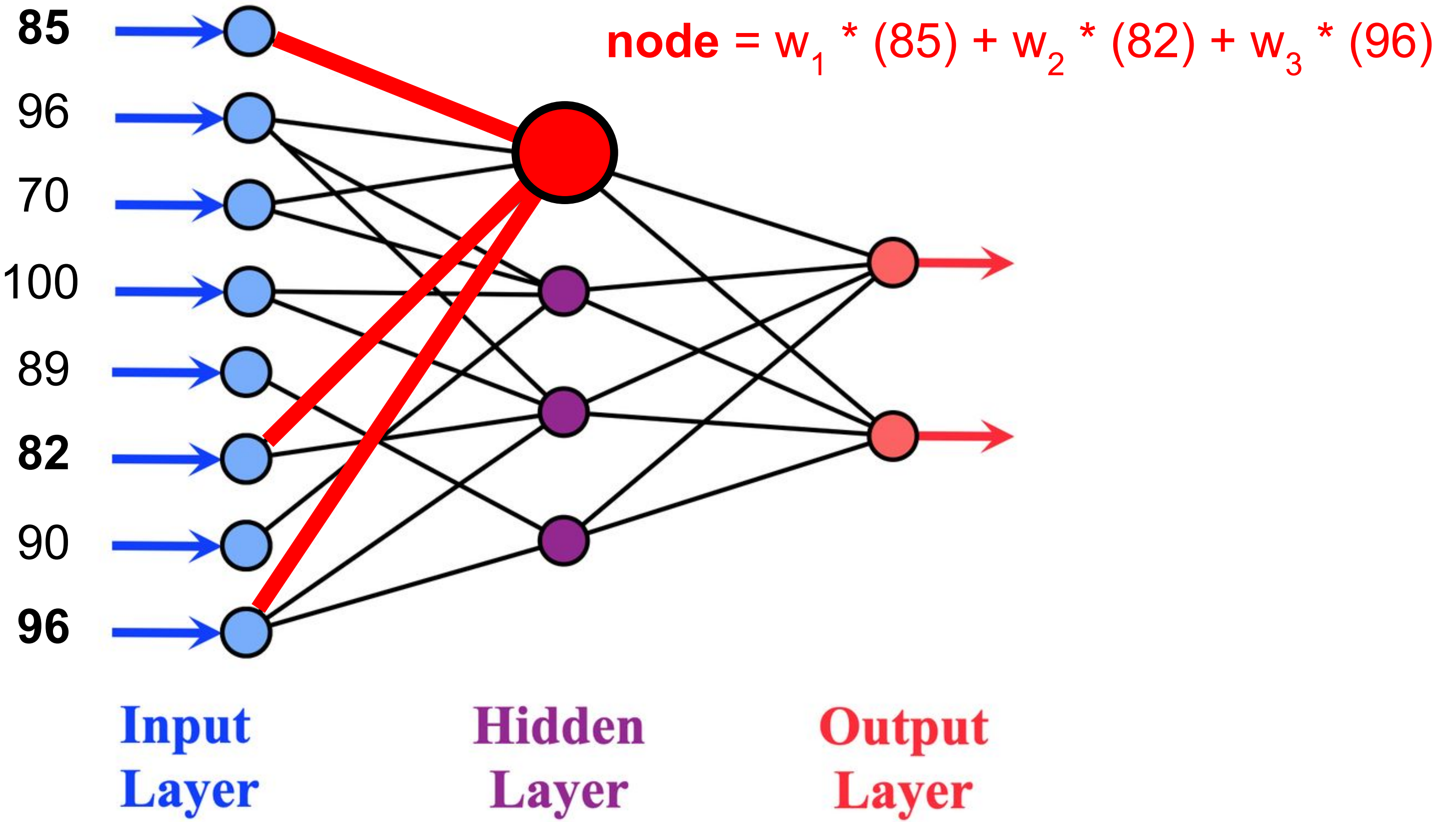
Supervised Machine Learning



Supervised Machine Learning

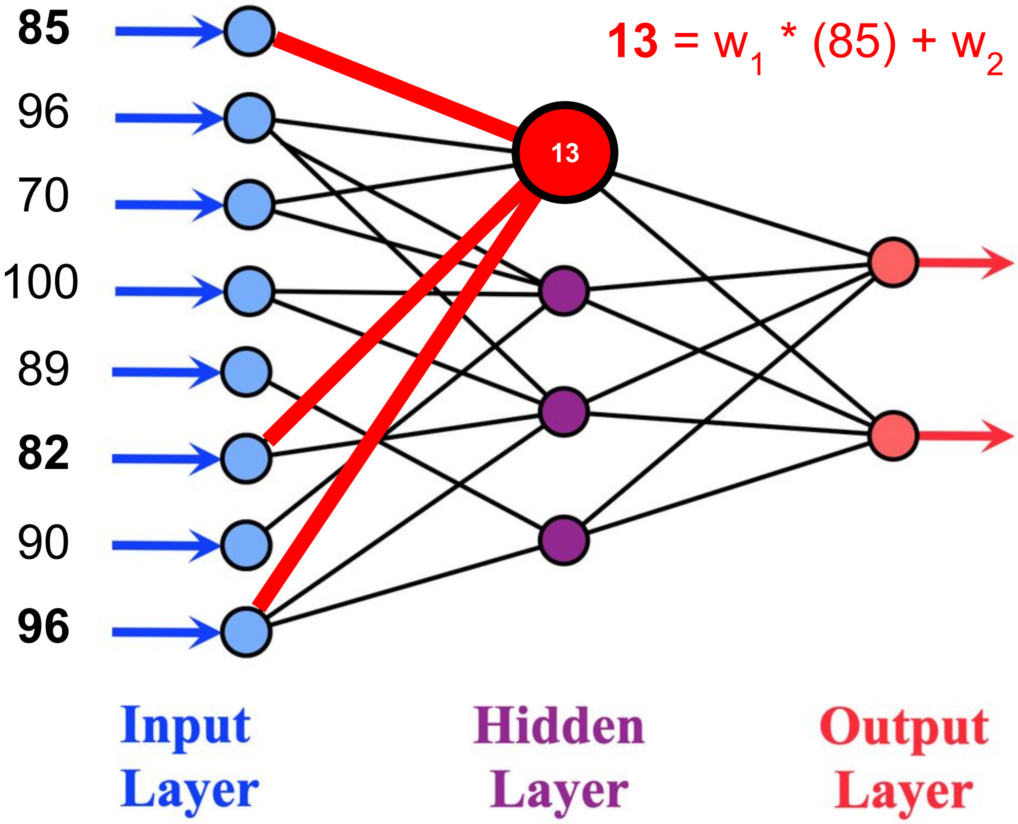


Supervised Machine Learning

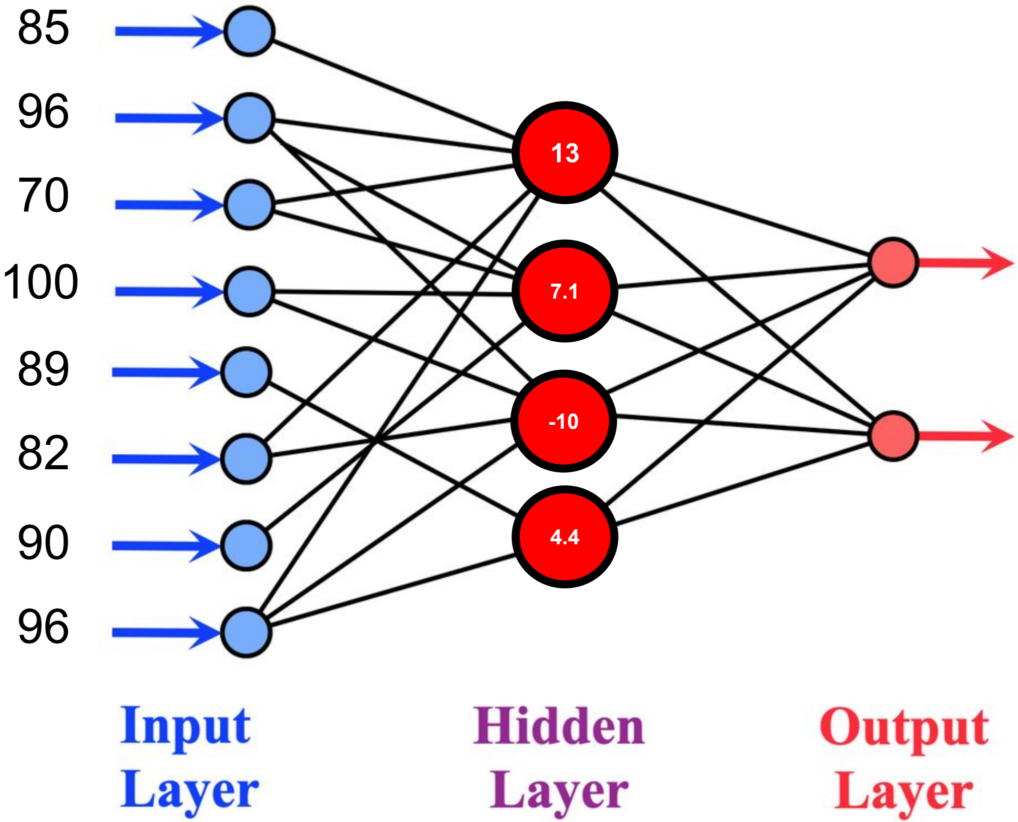


Supervised Machine Learning

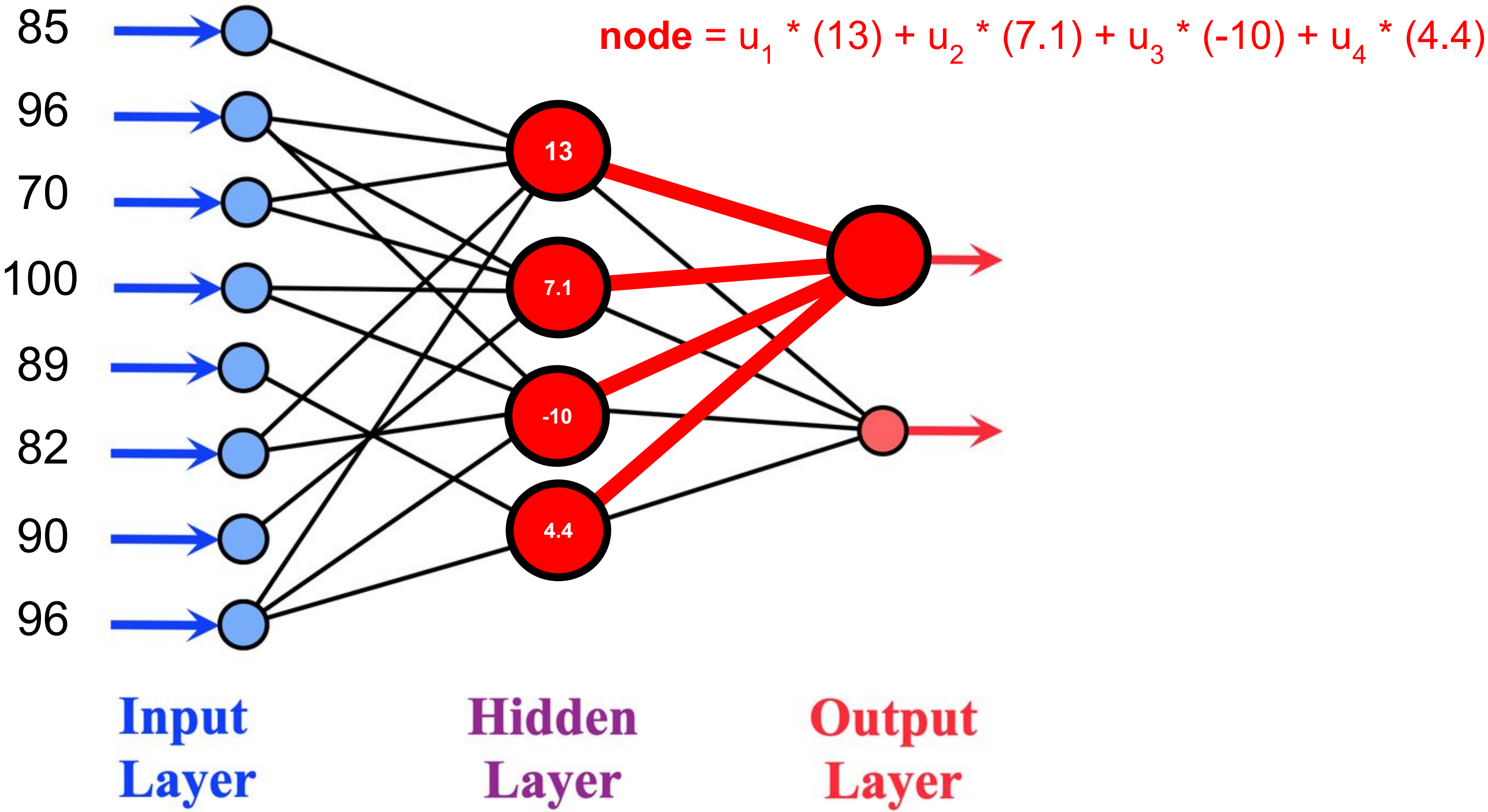
$$\mathbf{13} = w_1 * (85) + w_2 * (82) + w_3 * (96)$$



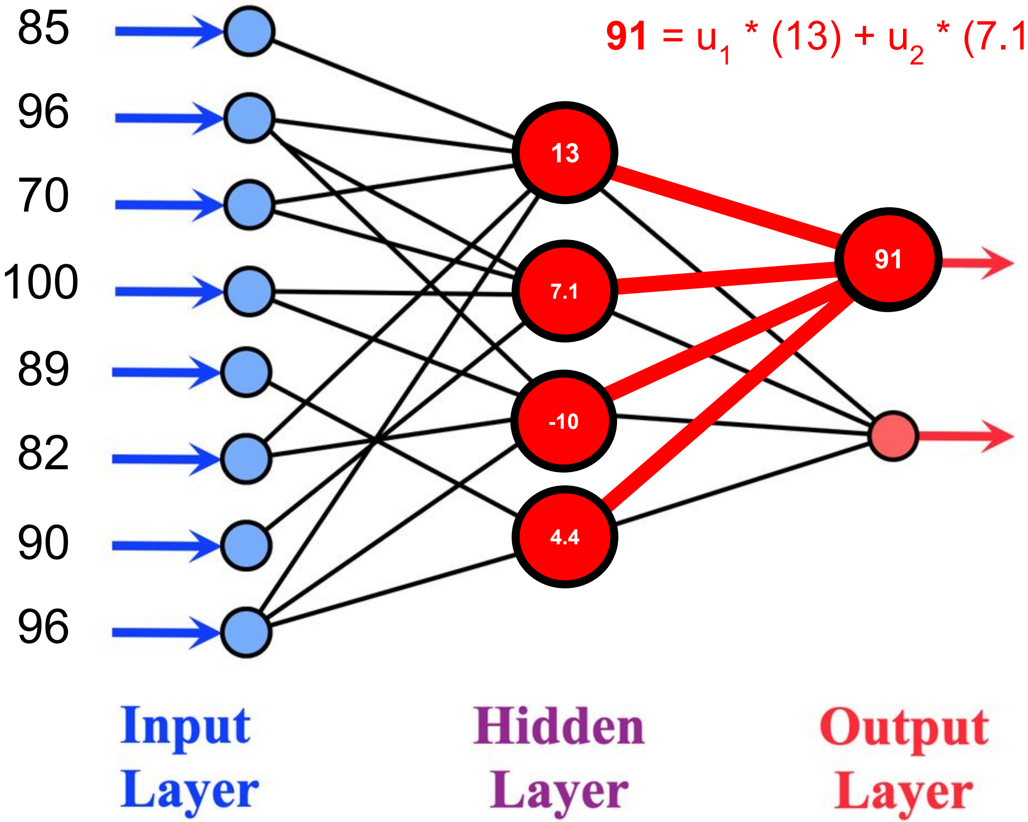
Supervised Machine Learning



Supervised Machine Learning

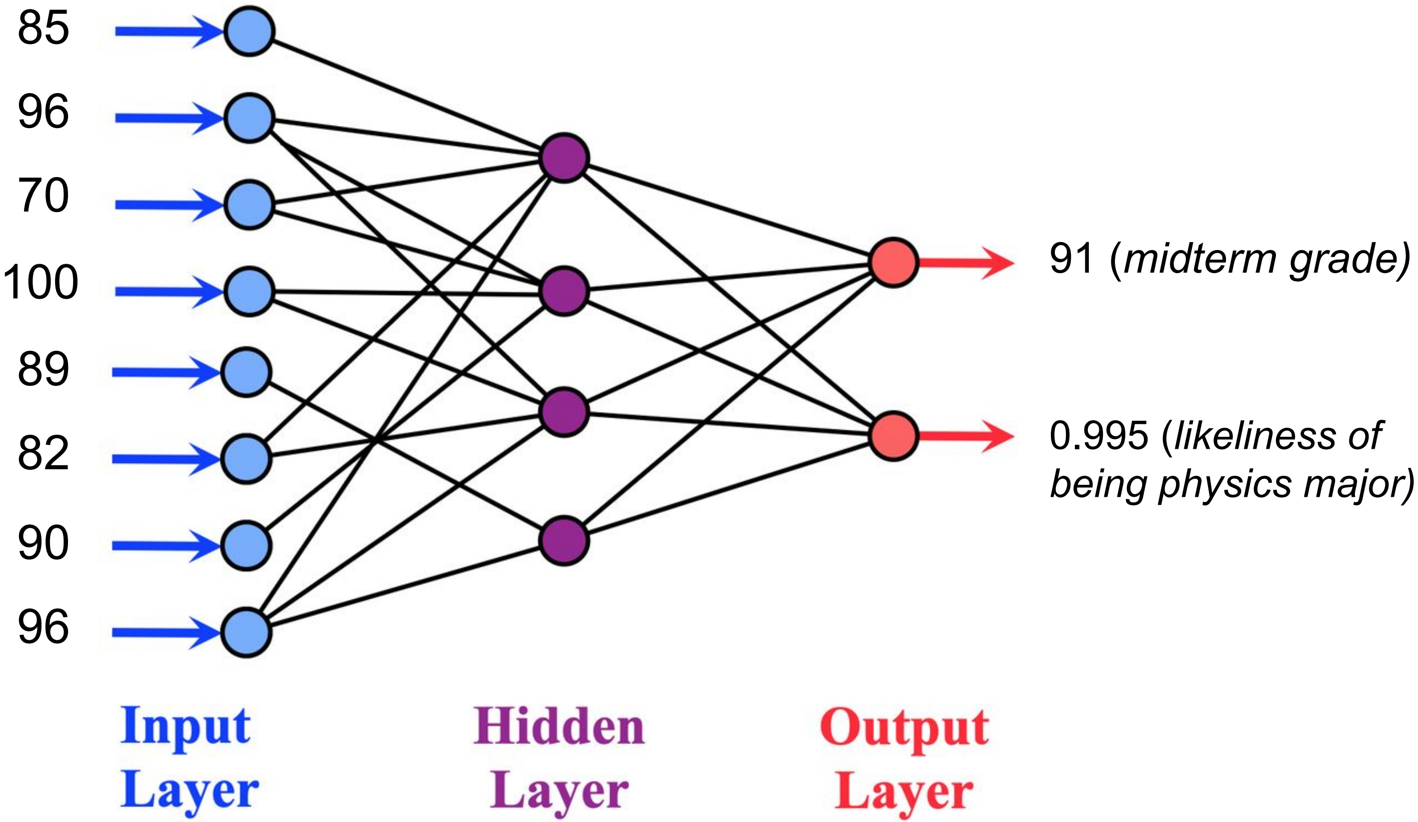


Supervised Machine Learning

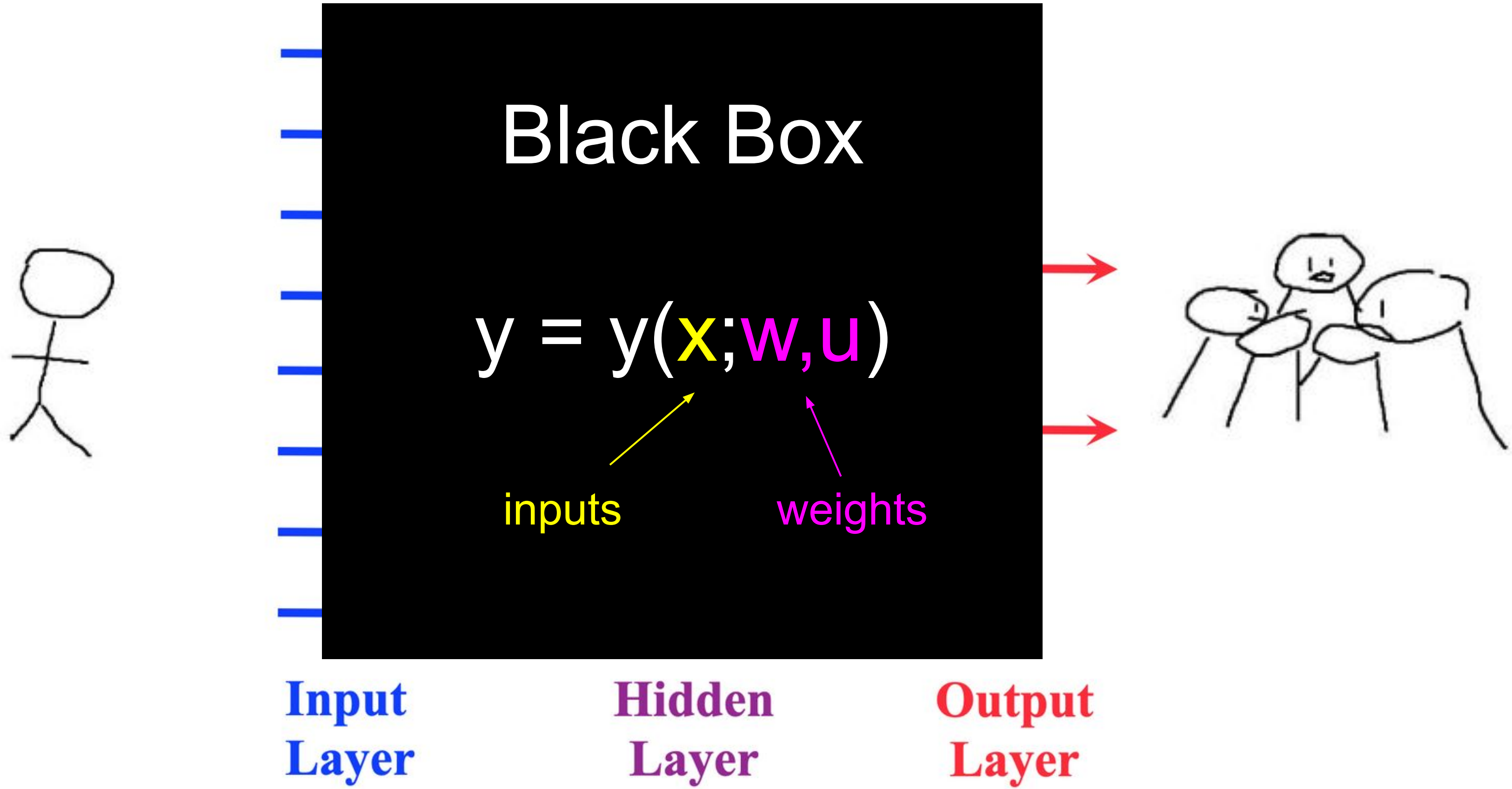


$$91 = u_1 * (13) + u_2 * (7.1) + u_3 * (-10) + u_4 * (4.4)$$

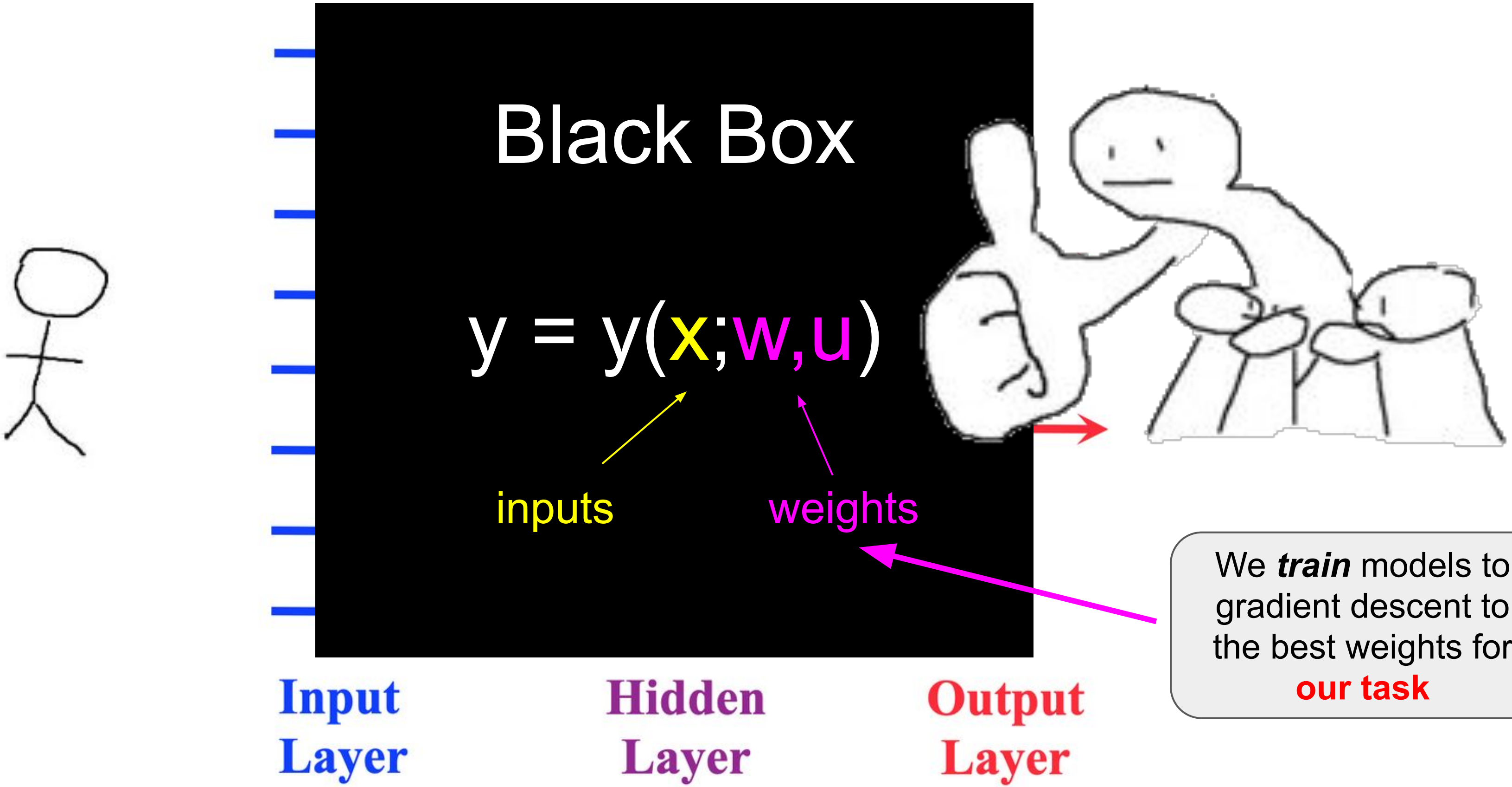
Supervised Machine Learning



Supervised Machine Learning

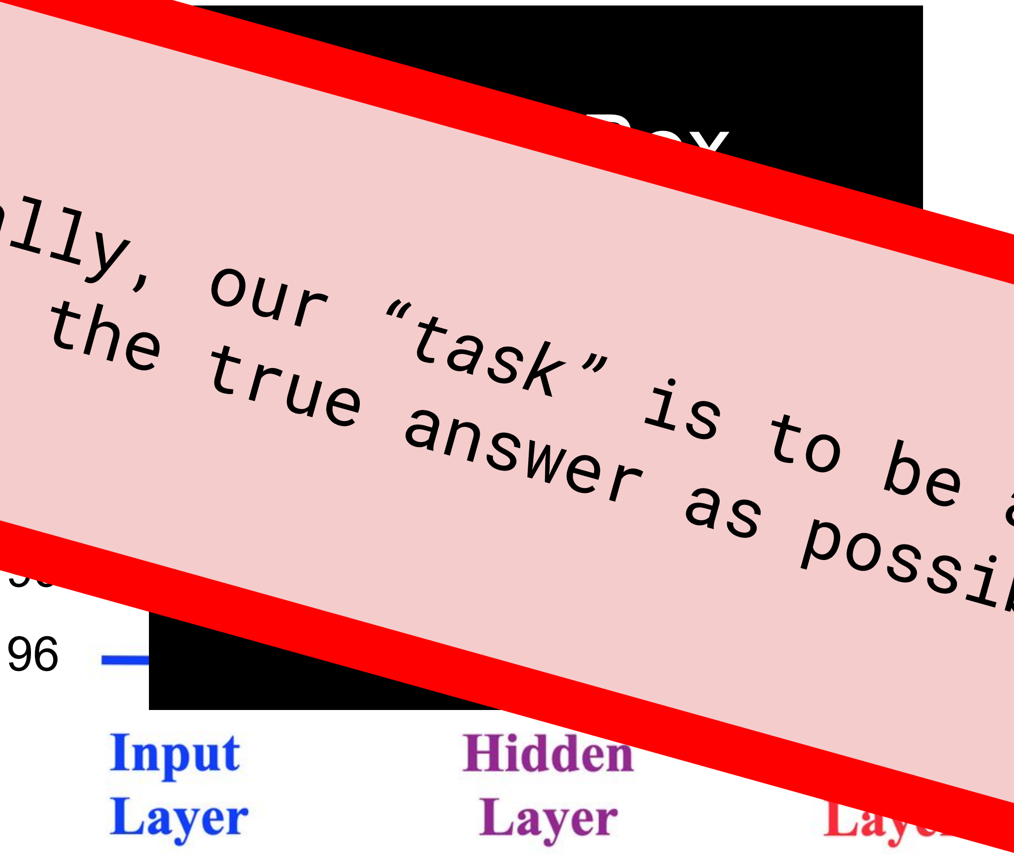


Supervised Machine Learning



We ***train*** models to gradient descent to the best weights for **our task**

Supervised Machine Learning



Typically, our "task" is to be as close to the true answer as possible

Supervised Machine Learning

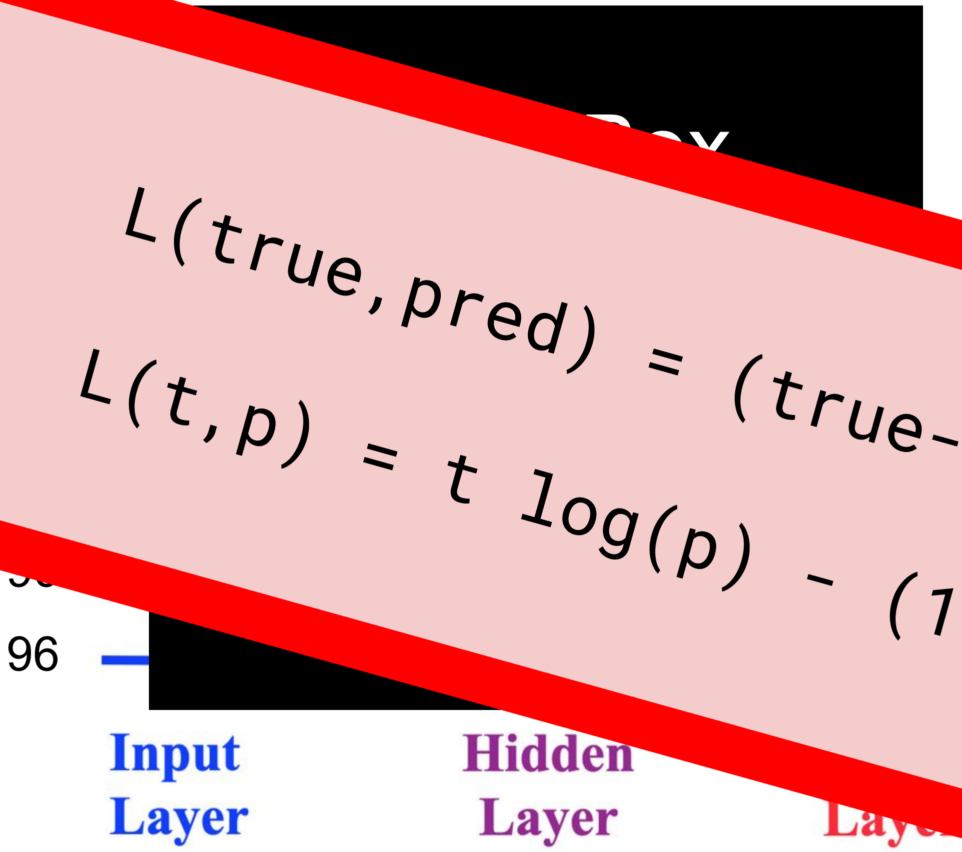
We define a **Loss Function**, a metric calculated during training that tells the model how bad its choice of **weights** is

96

Input Layer

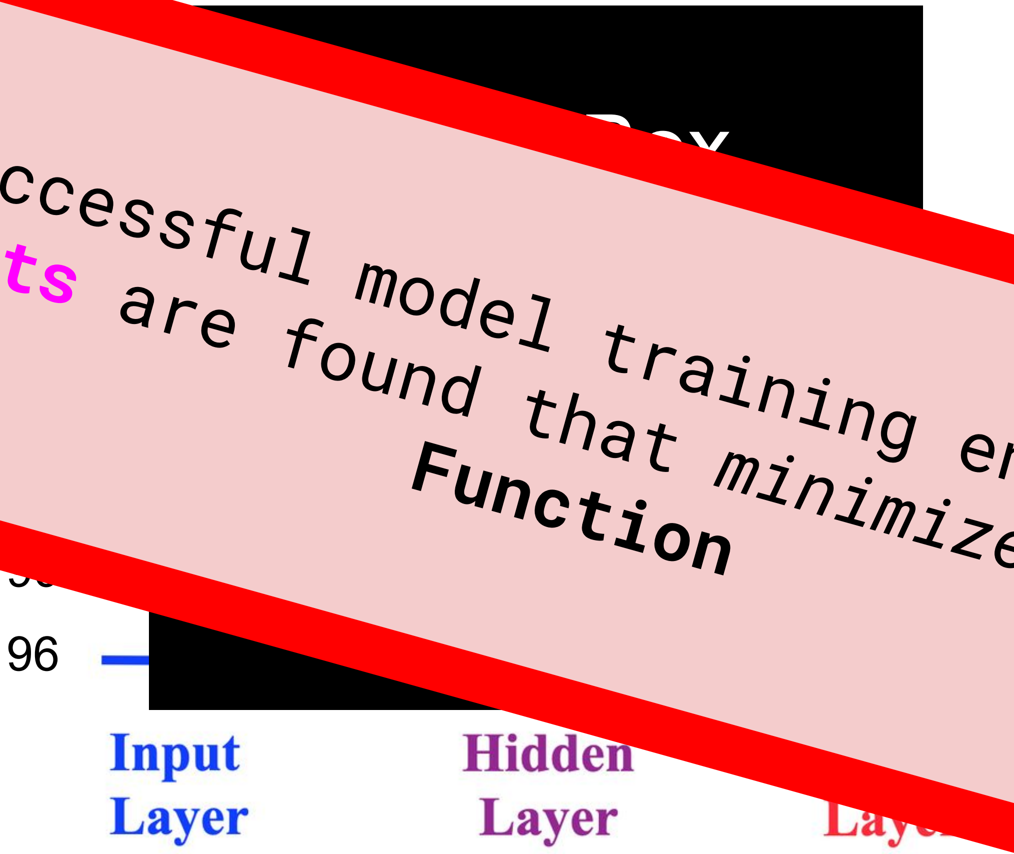
Hidden Layer

Supervised Machine Learning



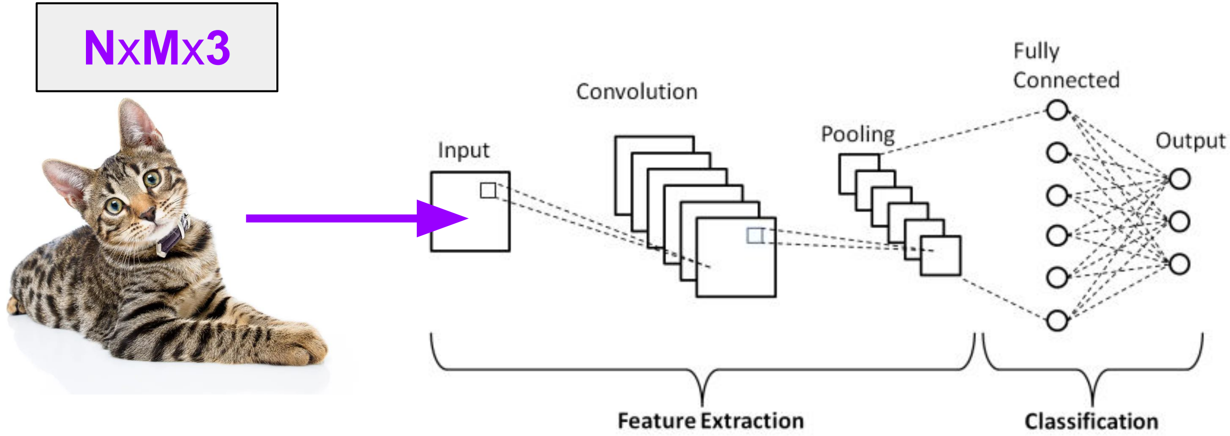

```
Ex A:    L(true,pred) = (true-pred)^2
Ex B:    L(t,p) = t log(p) - (1-t) log(1-p)
```

Supervised Machine Learning



Successful model training ends when **weights** are found that minimize the **Loss Function**

Task: Image Classification



dog	(0.025)
car	(0.004)
boat	(0.001)
cat	**(0.950)**
house	(0.001)
flute	(0.003)
grass	(0.000)
…	…

Given… An isolated ‘grid’ of inputs
Output… A list of prediction scores for each trained category

★Training★ is straightforward. ImageNet has ~14 million labeled images with more than 22,000 categories.

Image *within Image* Classification

NxMx3

???

1 person
5 sheep
1 dog

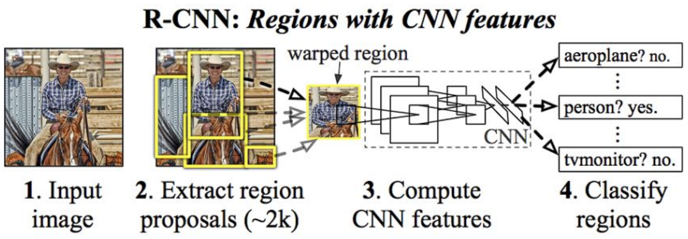
Given… An isolated ‘grid’ of inputs
Output… A potentially arbitrary number of objects, each classified

★**Training** is more difficult!★

- Cannot easily train for datasets with all possible category combinations
- How would one deal with situations where objects *overlap*?
- The ★**Approach**★ must be changed (can’t do simple CNN)

Some Object Detection Approaches

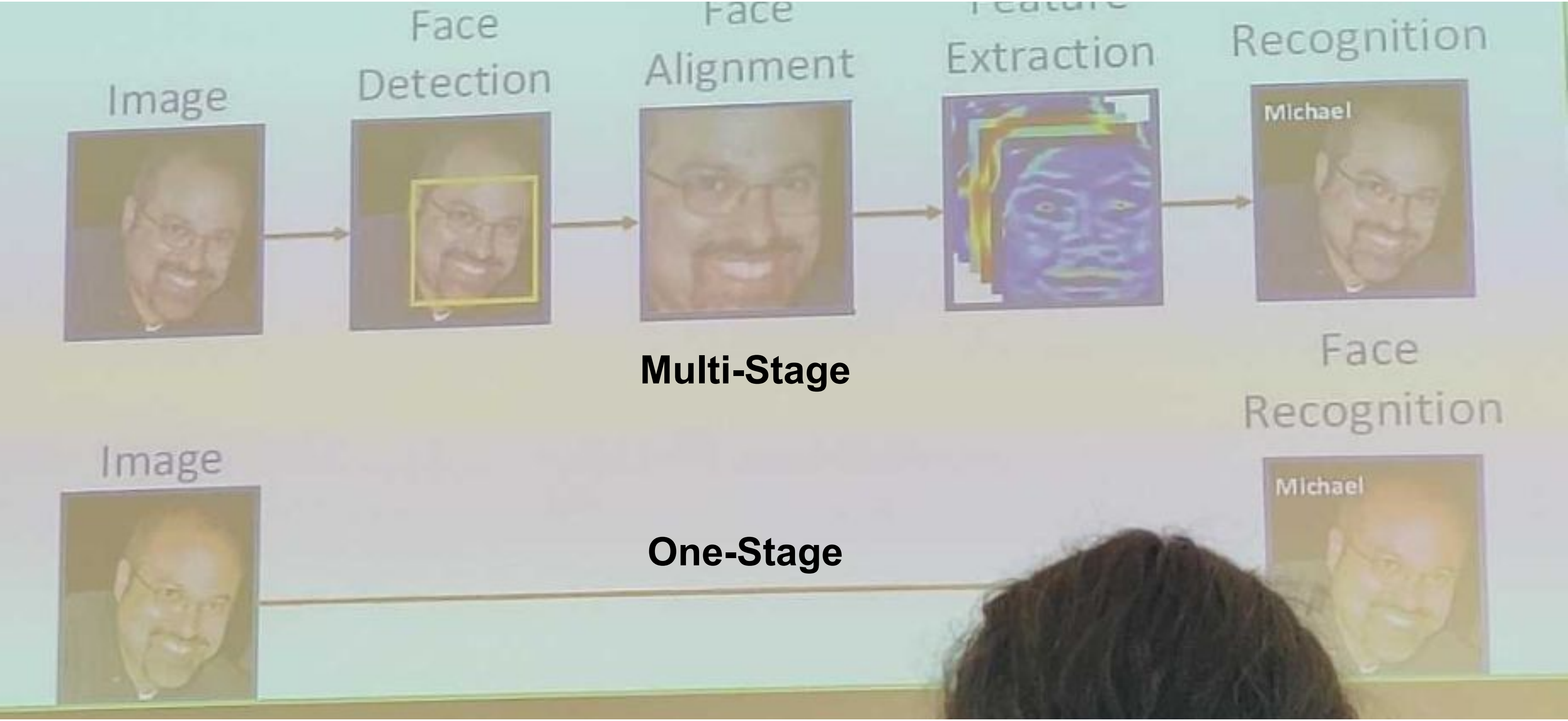
- Deformable Parts Models (DPM): https://ieeexplore.ieee.org/document/5255236
  - Scan a trained single image classifier over intervals within the main image (*slow and inefficient)*
- Region Based CNN (R-CNN): https://arxiv.org/abs/1311.2524



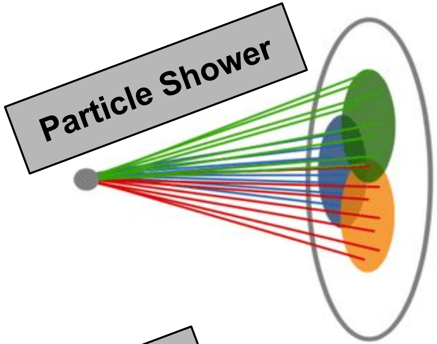
1. Propose bounding boxes
2. Classify each bounding box
3. Postprocess to eliminate duplicates & refine boxes → reclassify to see improvements

- You Only Look Once (YOLO): https://arxiv.org/pdf/1506.02640.pdf
  - Reduce the object detection into a fast, biologically similar **one-stage** approach
  - **Entire image is used when training** → full context allows for reduction of backgrounds
  - The "*Probability that we should have a bounding box here"* is weighted by the bounding box class

Obligatory Funny Slide at AI4EIC



The N-to-K Problem



Raw Inputs

N “sparse” hits in a detector
- x
- y
- z
- timing
- edep

Object Clustering

Grouping of hits into more general objects (ex: jets)

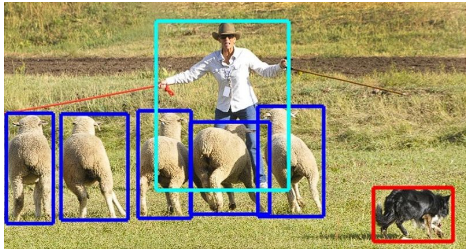
Physics

Process K clusters info to yield
- Centroid
- Total energy
- Spread/width
- Particle ID

Raw Inputs

Grid of N pixels
- x
- y
- RGB

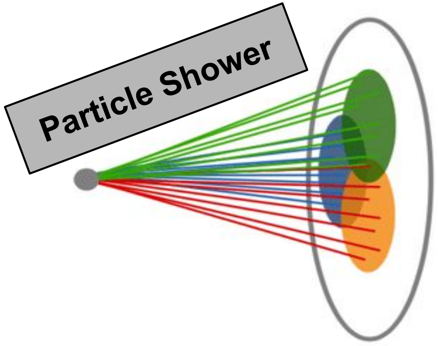
Object Clustering



Classify

1 person
5 sheep
1 dog

The N-to-K Problem



Raw Inputs

N "sparse" hits in a detector
- x
- y
- z
- timing
- edep

Object Clustering

Grouping of hits into more general objects (jets)

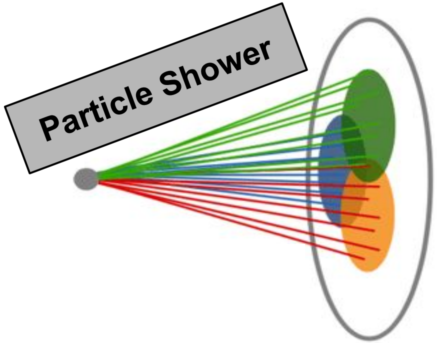
Physics

Process **K** clusters info to yield
- Centroid
- Total energy
- Spread/width
- Particle ID

Because… The structure of particle physics data is ** *non-trivial (sparse)* **

Then… The *Object Clustering* and *Physics Predictions* stages are separate analyses (MLs)

The N-to-K Problem



Raw Inputs

N "sparse" hits in a detector
- x
- y
- z
- timing
- edep

Object Clustering

Grouping of hits into more general objects (jets)

Physics

Process K clusters info to yield
- Centroid
- Total energy
- Spread/width
- Particle ID

Because… The structure of particle physics data is ** *non-trivial (sparse)* **

Then… The *Object Clustering* and *Physics Predictions* stages are separate analyses (MLs)

★Object Condensation★ is an approach to train a machine learning model to both *cluster* and *predict physical properties* of a sparse data set

Object Condensation Basics

Input Space → Point Cloud

- A point cloud is a discrete set of *(N)* data points with (*d*) components
- ex: (x , y , z , t , E)

Output Space → Another Point Cloud

- End result will be a discrete set of (*N*) data points with (*d' + 2 + 1)* components
- d' → Number of properties to predict (ex: $\mathbf{p_x}$ **,** $\mathbf{p_y}$ **,** $\mathbf{p_z}$ **, pid ,** …)
- +2 → Latent space coordinates (*explained later*)
- +1 → Confidence beta value (*explained later*)

In essence, **object condensation** lays the groundwork for how the **+2** and **+1** variables are to be 'predicted' such that clustering is automatically performed

Object Condensation Basics

Input Space → Point Cloud

- A point cloud is a discrete set of *(N)* data points with (*d*) components
- ex: (x , y , z , t , E)

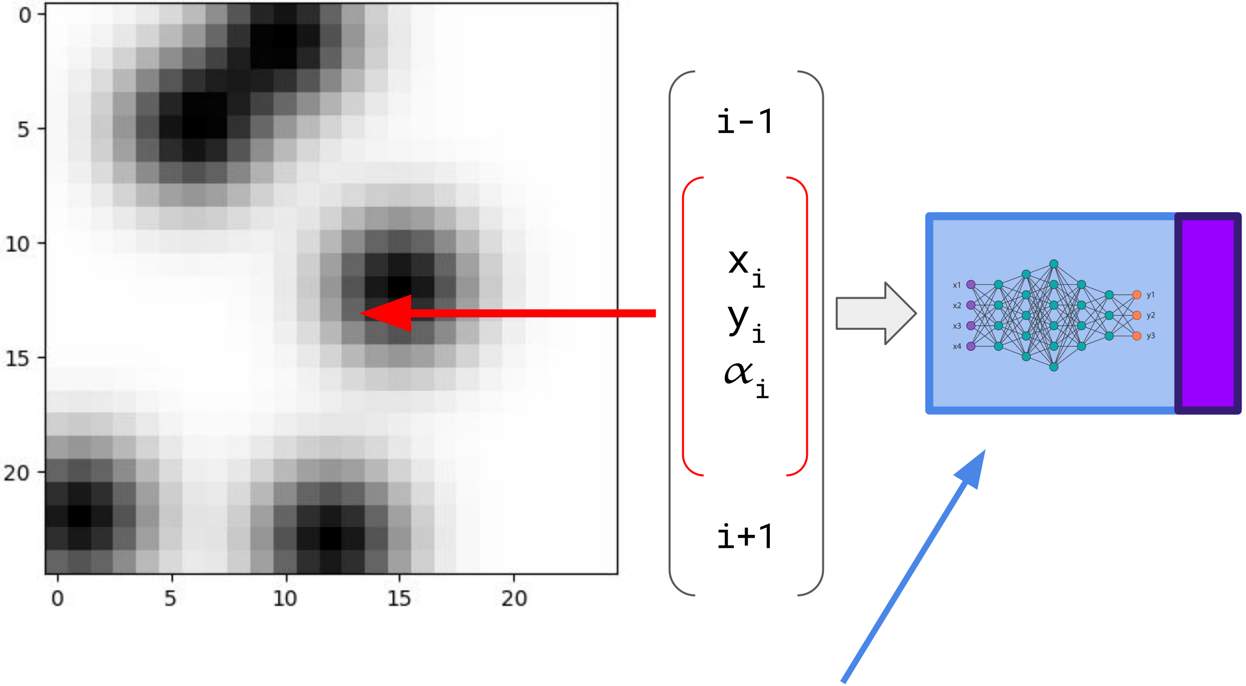
Output Space → Another Point Cloud

- End result will be a discrete set of (*N*) data points with (*d' + 2 + 1)* components
- d' → Number of properties to predict (ex: $\mathbf{p_x}$ **,** $\mathbf{p_y}$ **,** $\mathbf{p_z}$ **, pid ,** …)
- +2 → Latent space coordinates (*explained later*)
- +1 → Confidence beta value (*explained later*)

In essence, **object condensation** lays the groundwork for how the **+2** and **+1** variables are to be 'predicted' such that clustering is automatically performed

Much more complicated "task" to formulate

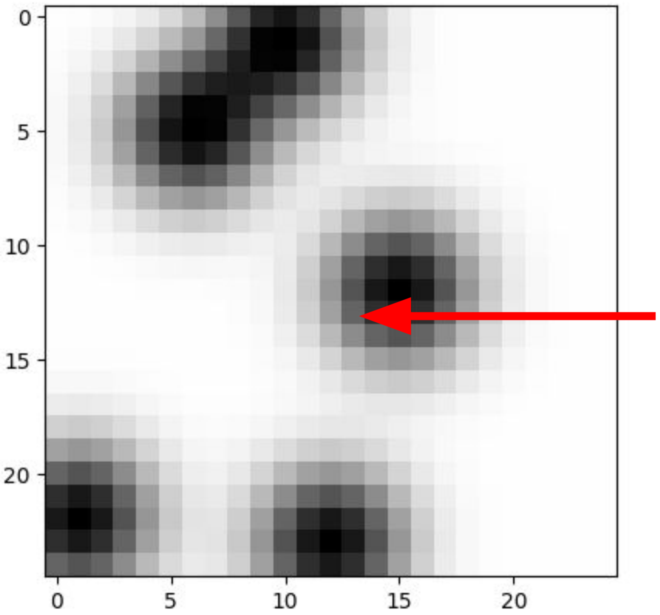
Object Condensation Basics

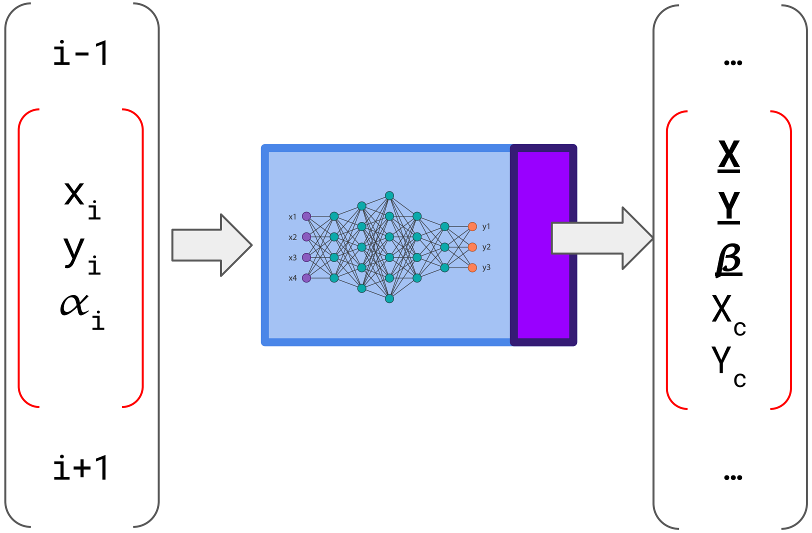


I want my ML model to tell me how many clusters, and their centroids (x_c , y_c)

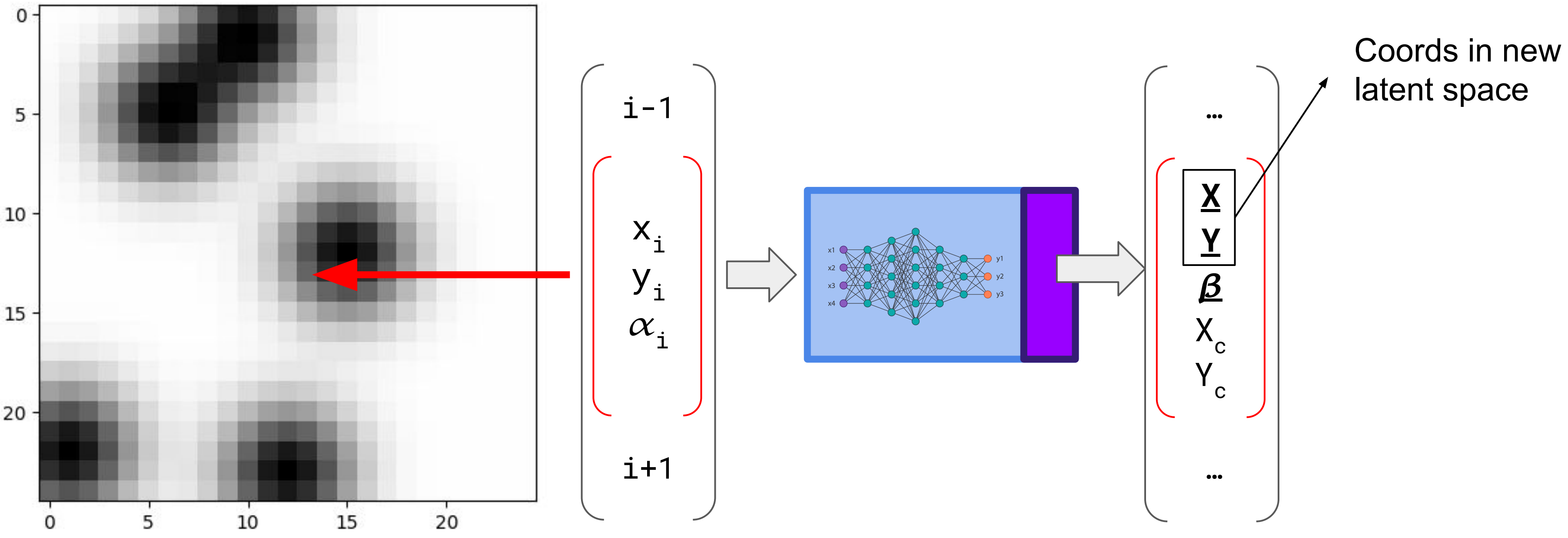
*Lets see what a **well-trained** model does, then discuss how we even **train** it to perform the task at hand (clustering!)*

Object Condensation Basics

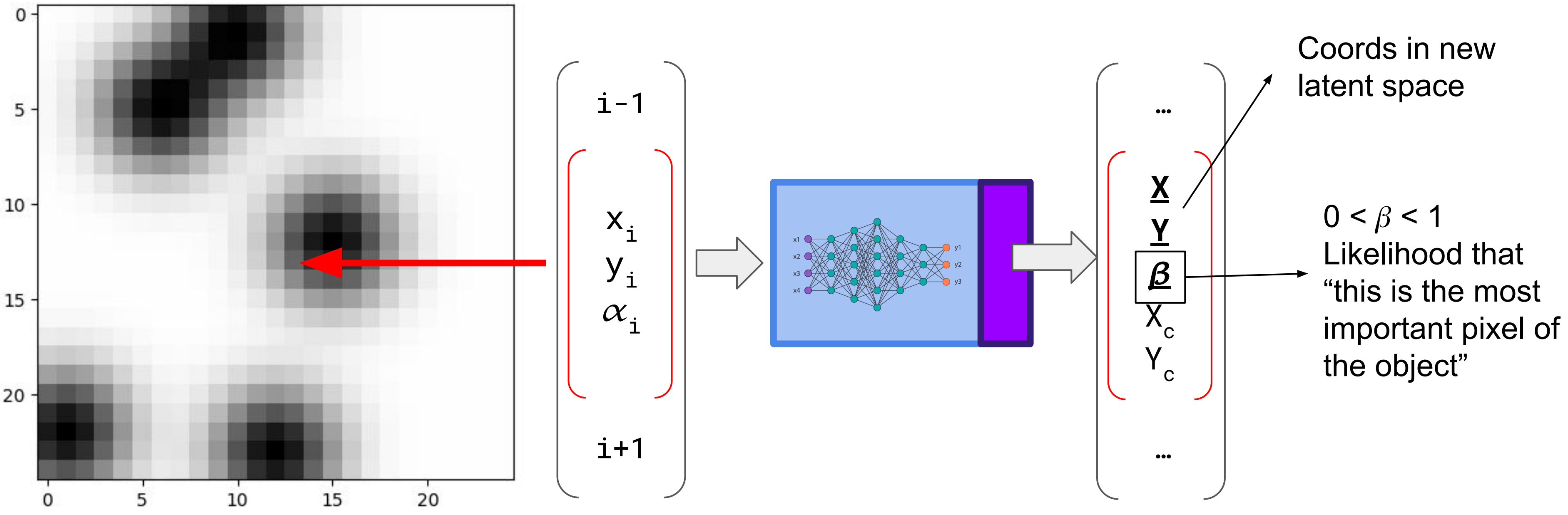




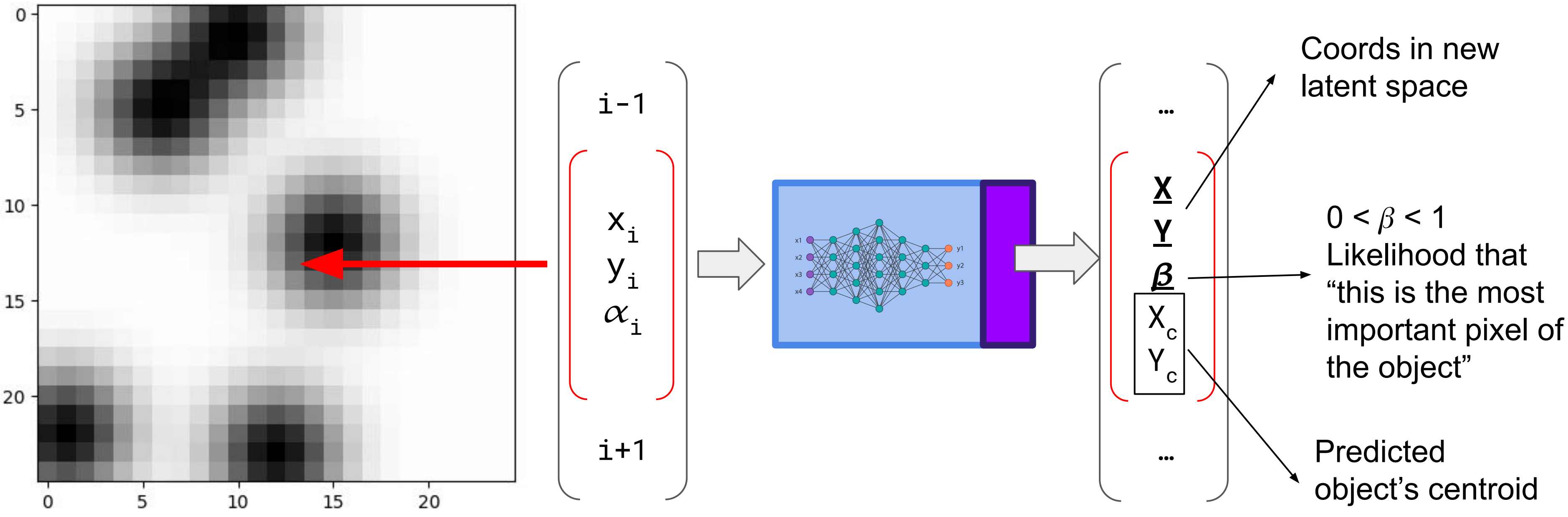
Object Condensation Basics



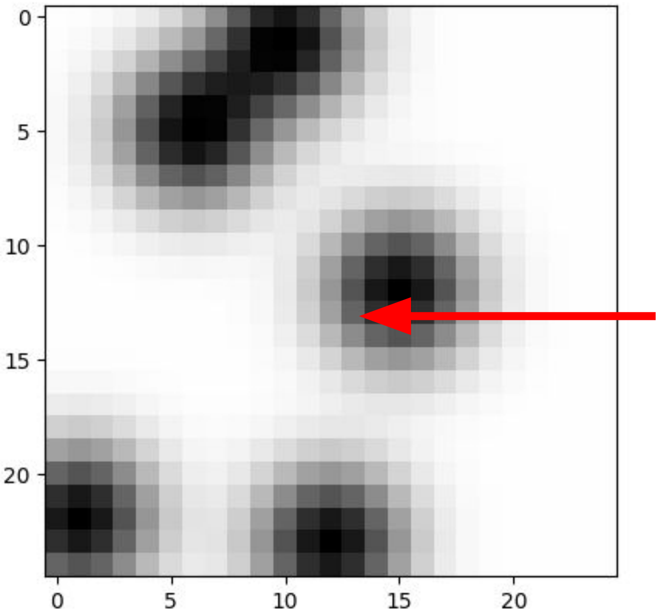
Object Condensation Basics

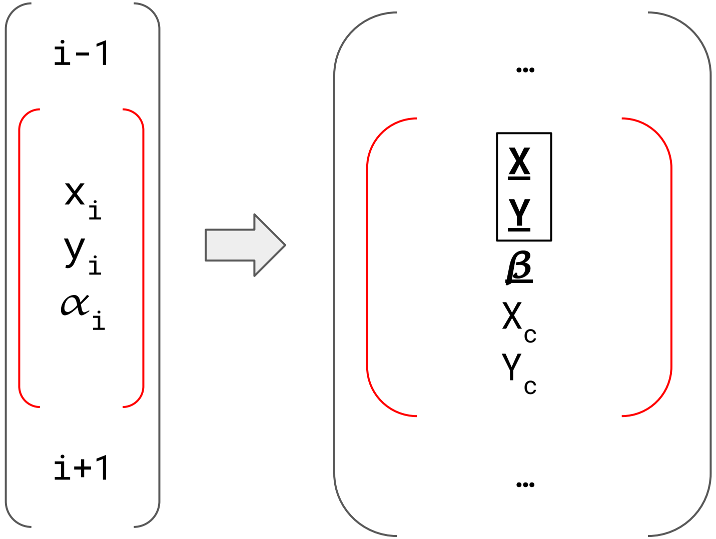


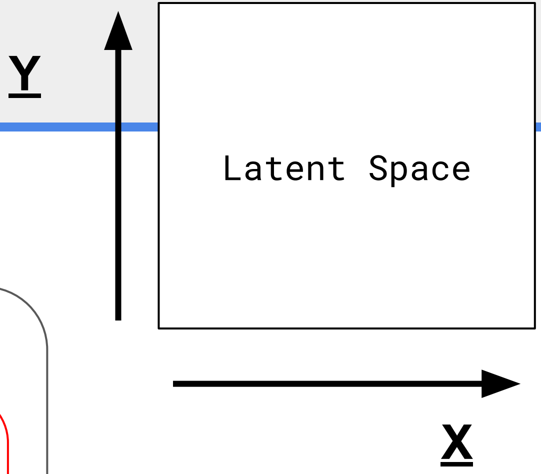
Object Condensation Basics



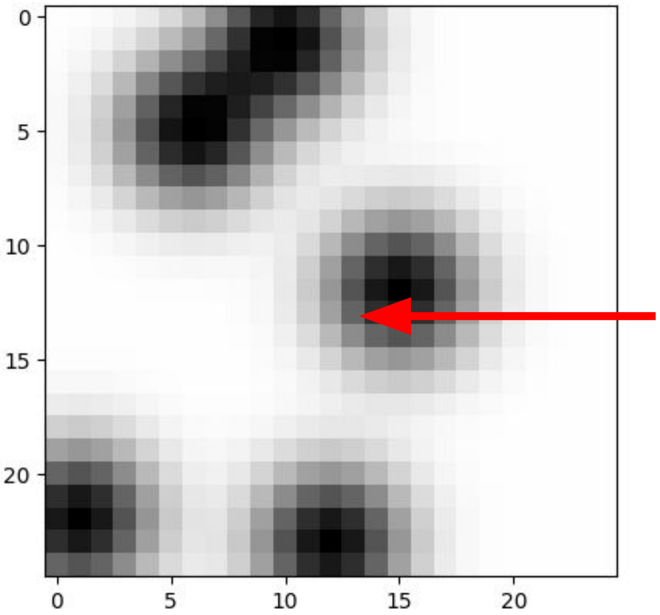
Object Condensation Basics







Object Condensation Basics



i-1

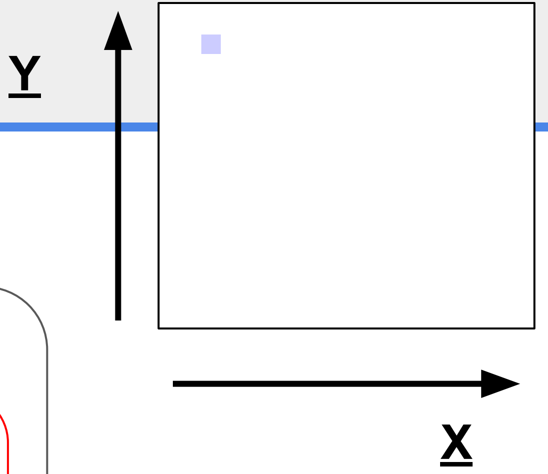
x_i
y_i
α_i

i+1

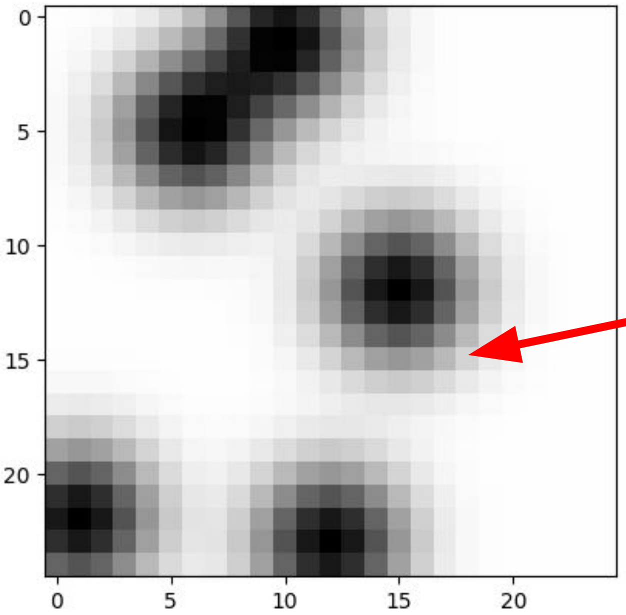
…

X = 12
Y = 55
β = 0.002
X_c = ...
Y_c = ...

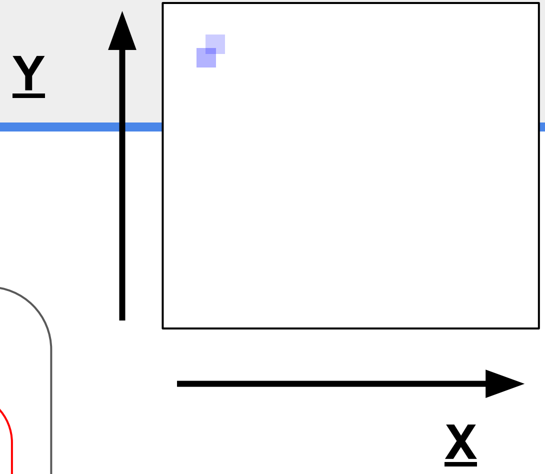
…



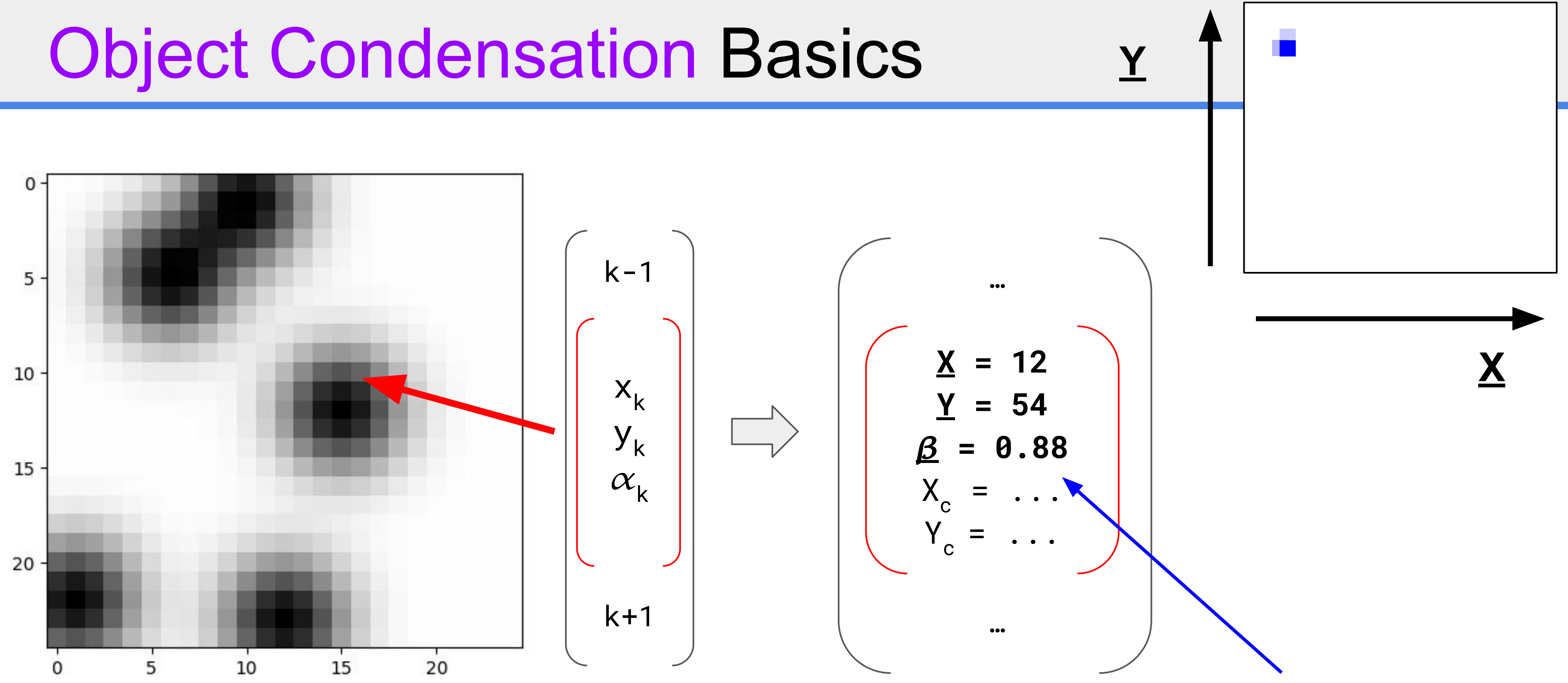
Object Condensation Basics



$$\begin{pmatrix} j-1 \\ \begin{pmatrix} x_j \\ y_j \\ \alpha_j \end{pmatrix} \\ j+1 \end{pmatrix} \Rightarrow \begin{pmatrix} \dots \\ \begin{pmatrix} \underline{X} = 11 \\ \underline{Y} = 54 \\ \underline{\beta} = 0.01 \\ X_c = \dots \\ Y_c = \dots \end{pmatrix} \\ \dots \end{pmatrix}$$

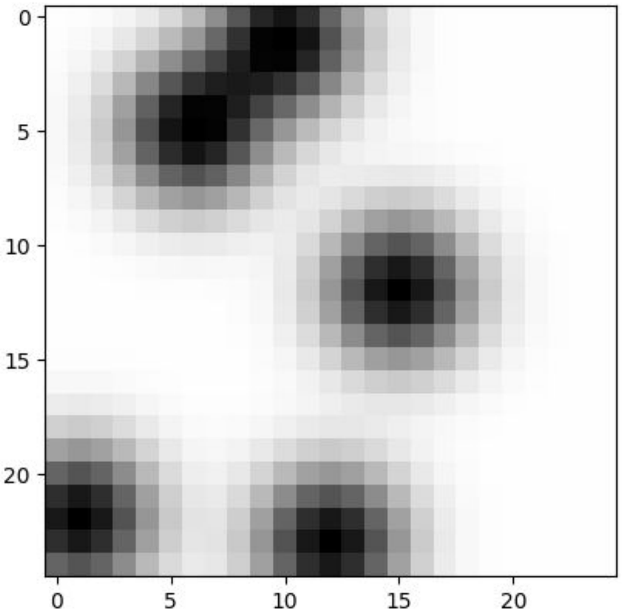


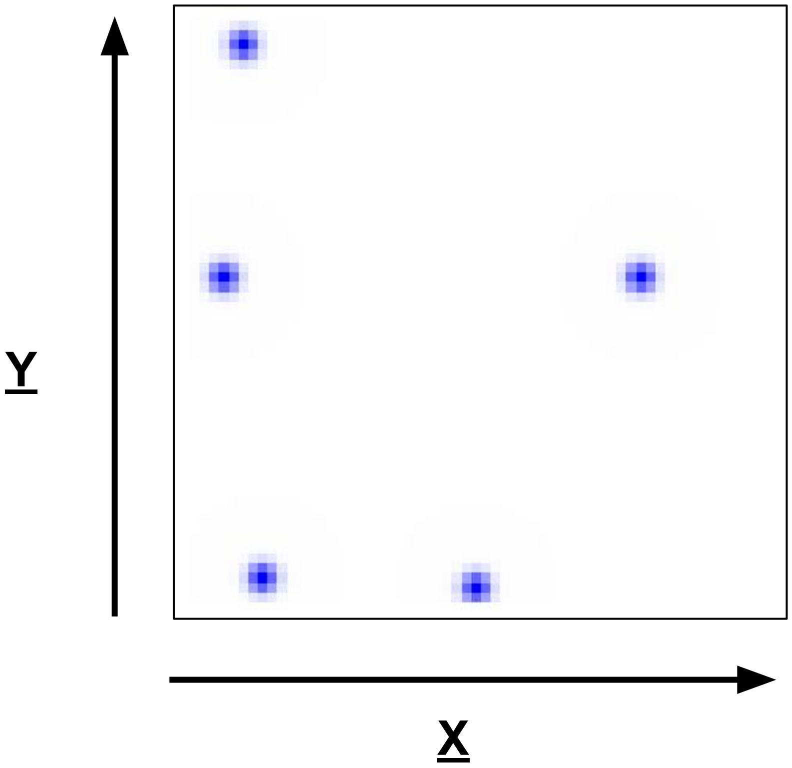
Object Condensation Basics



High β implies the model thinks this point is very important!

Object Condensation Basics

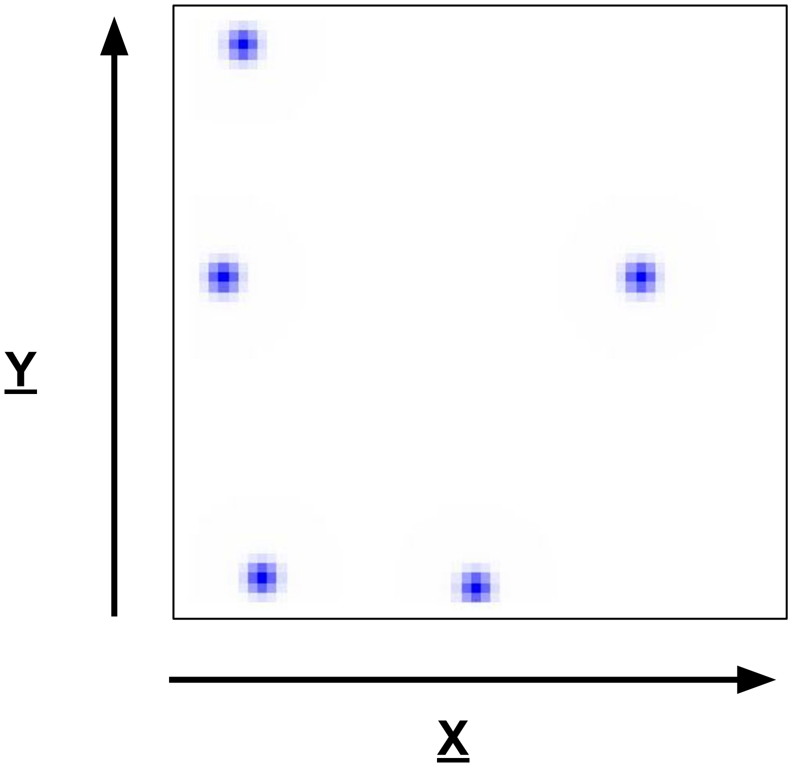




Input: Set of $[x,y,\alpha]$ (625x3)
Output: Set of $[\underline{\mathbf{X}},\underline{\mathbf{Y}},\beta,x_c,y_c]$ (625x5)

Model is trained to make 1 bright β per object

Object Condensation Basics



Solution becomes much simpler to picture…

… threshold away dim pixels (β < 0.8) …

… count the # pixels remaining …

… read off their predicted x_c and y_c …

★Object Condensation★

What is going on under the hood?

★Object Condensation★

What we understand so far:

- Object Condensation requires a frontend architecture (ex: CNN) to predict *for each* point a $\boldsymbol{\beta}$, a set of **latent space coords**, and the **object's features**

What we demand that frontend to produce:

1. Points corresponding to the same object *group together* in the **latent space**
2. Only a single point per object has a large $\boldsymbol{\beta}$
3. During clustering, the points with larger β possess the most accurate predictions of the **object's features**

★Object Condensation★

What we understand so far:

- Object Condensation requires a frontend architecture (ex: CNN) to predict *for each* point a $\boldsymbol{\beta}$, a set of **latent space coords**, and the **object's features**

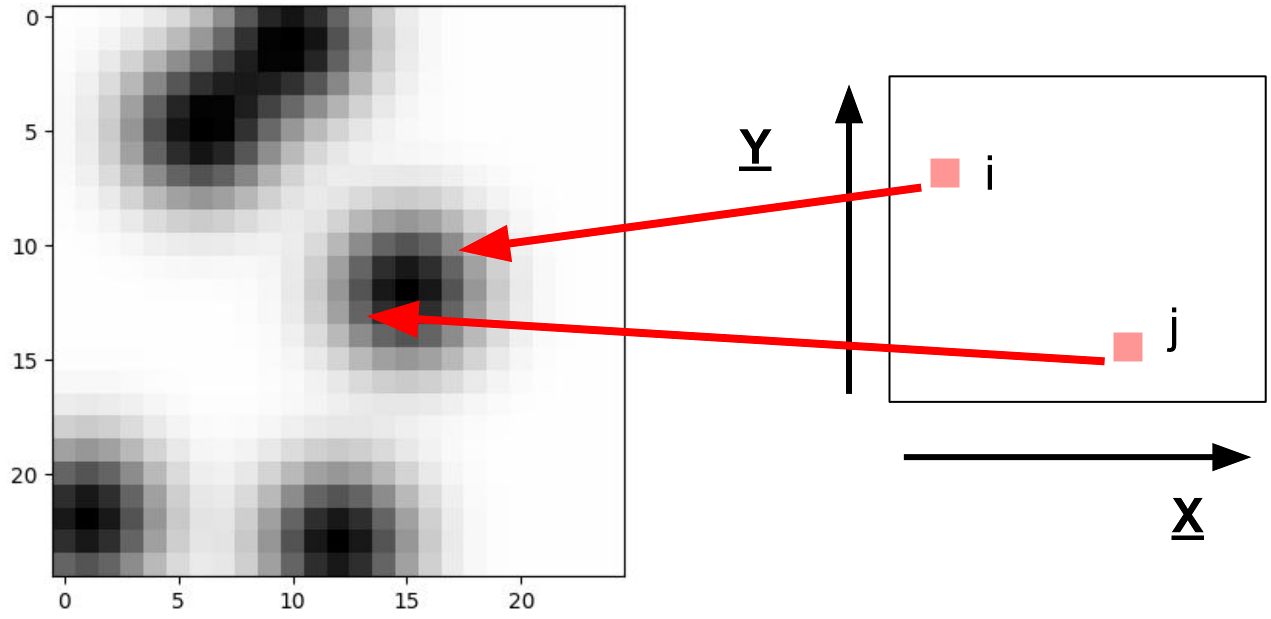
What we demand that frontend to produce:

1. Points corresponding to the same object *group together* in the **latent space**
2. Only a single point per object has a large $\boldsymbol{\beta}$
3. During clustering, the points with larger β possess the most accurate predictions of the **object's features**

Each of the three demands can be written as their own *Loss Function* which penalizes the frontend for not doing its job

These unique training requirements create a learning environment that gives us what we want (clear clustering of objects w/ feature predictions)

Demand A: Points group together

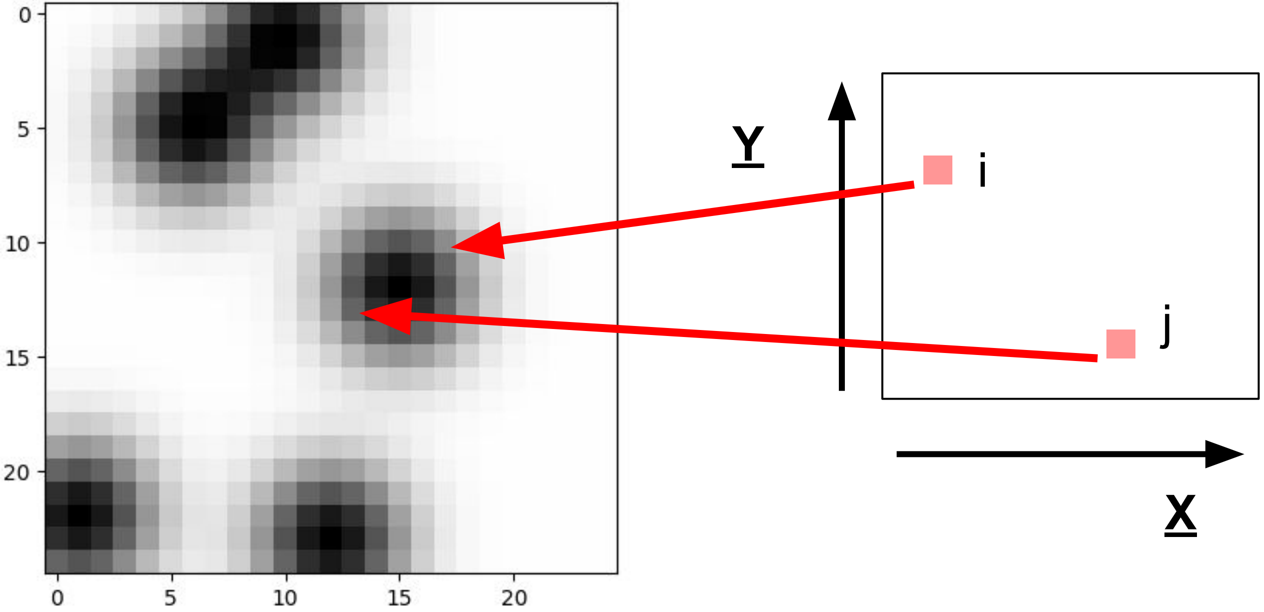


During training, we know these points (i,j) come from the same object…

…we want them to attract to one another in the latent space…

Before training, **X** & **Y** are random

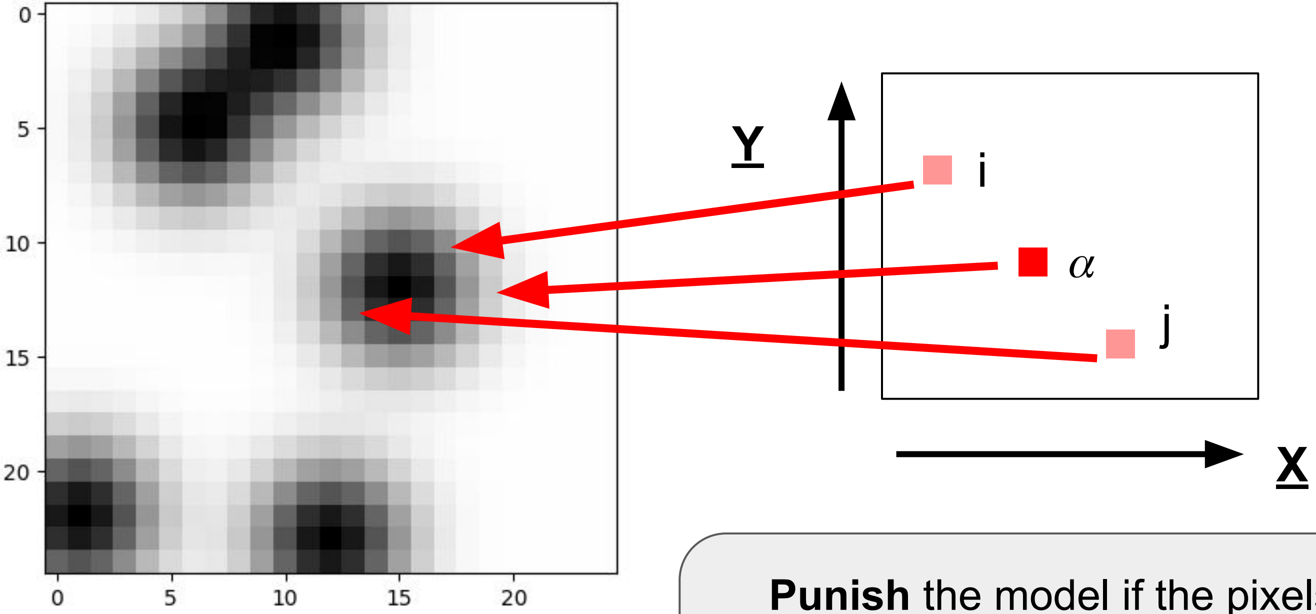
Demand A: Points group together



If we want to encourage our frontend model to make these **X,Y** closer, have it minimize some **attractive potential**

How to define?

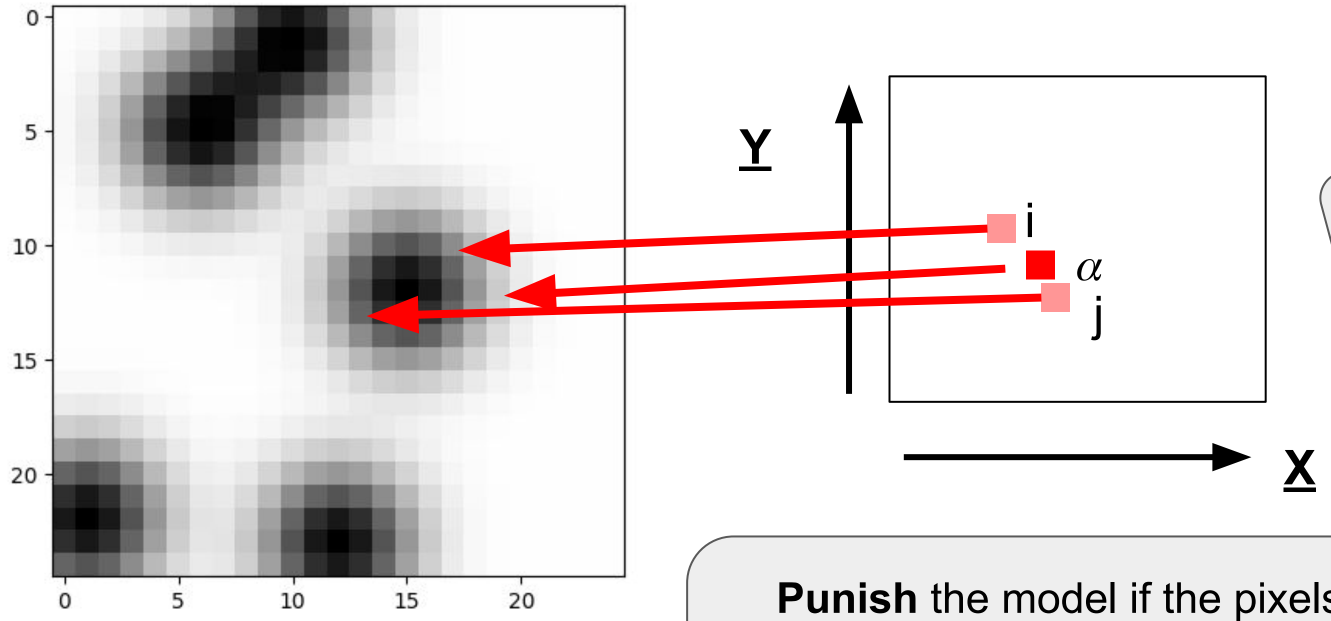
Demand A: Points group together



Punish the model if the pixels are “far away” from the *brightest* (highest β) pixel

Label that brightest pixel as $\boldsymbol{\alpha}$

Demand A: Points group together

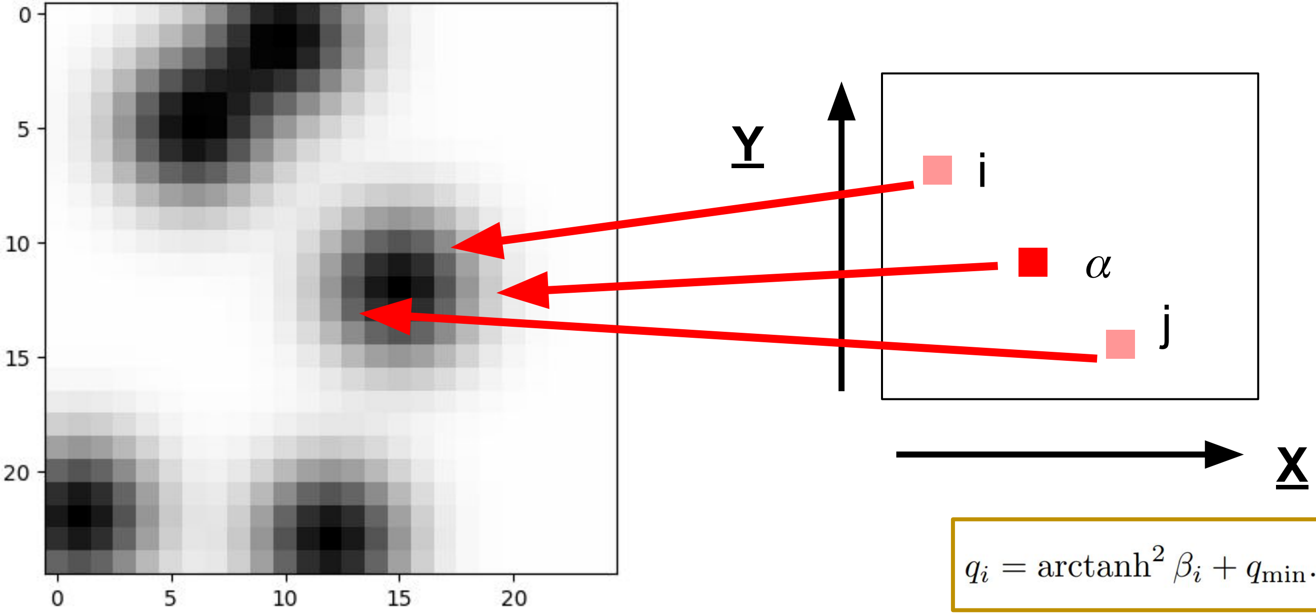


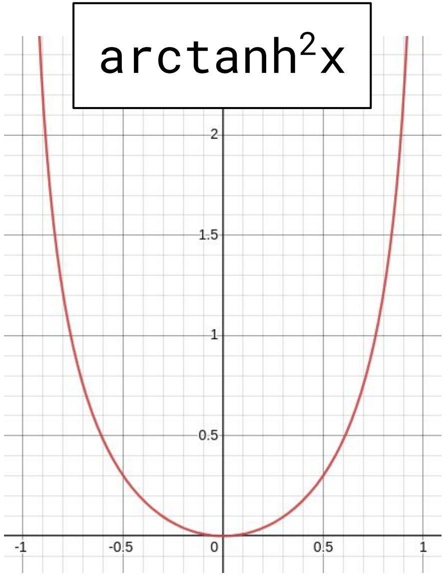
Less punishment !!!

Punish the model if the pixels are "far away" from the *brightest* (highest β) pixel

Label that brightest pixel as $\boldsymbol{\alpha}$

Demand A: Points group together





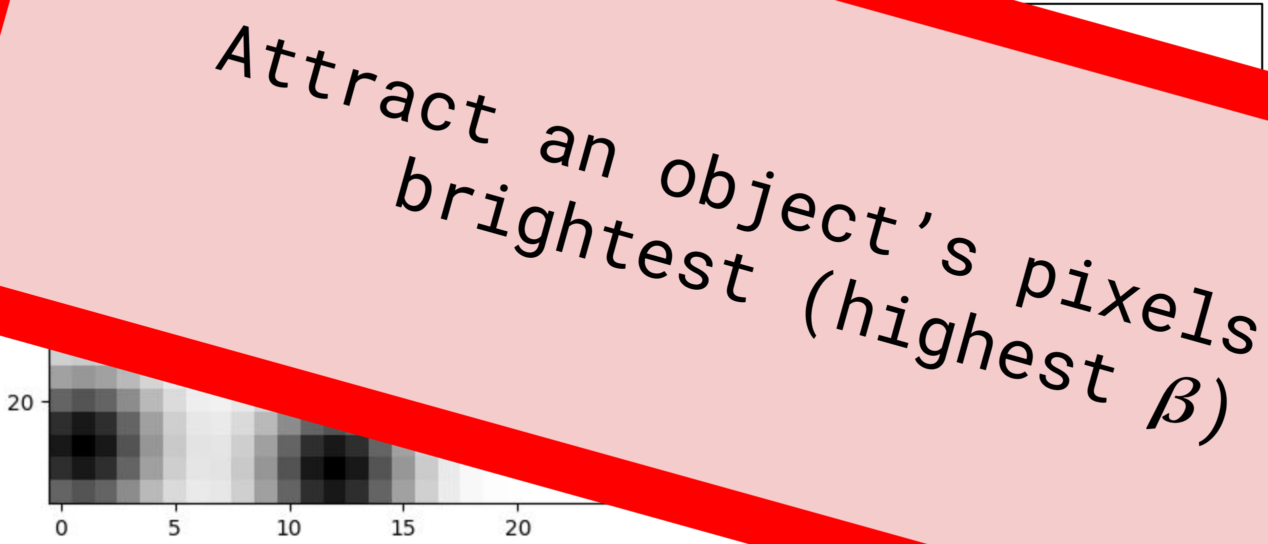
$$q_i = \operatorname{arctanh}^2 \beta_i + q_{\min}.$$

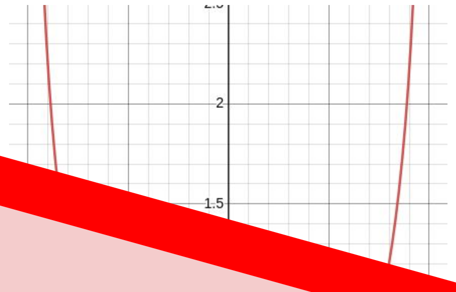
$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k},$$

i → unique id for each pixel [0,1,...,624]
α→per object k, id of pixel with highest charge
k→unique id of the object [0,1,2,3,4]
x → latent space coordinates

Demand A: Points group together

Attract an object's pixels towards its brightest (highest β) member

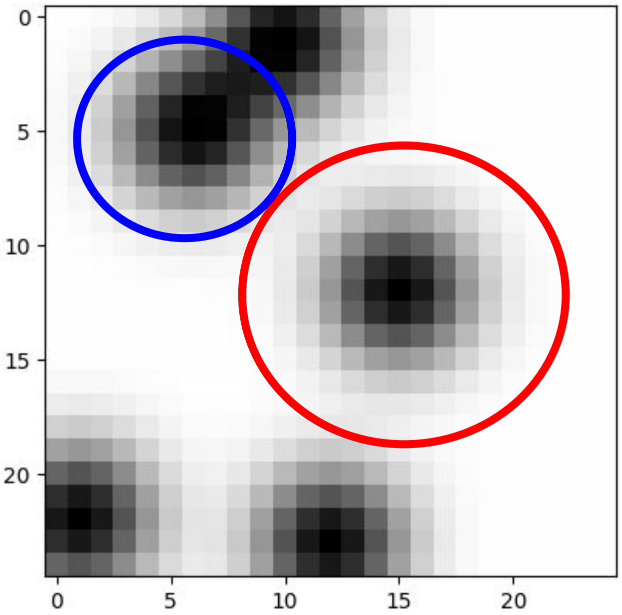


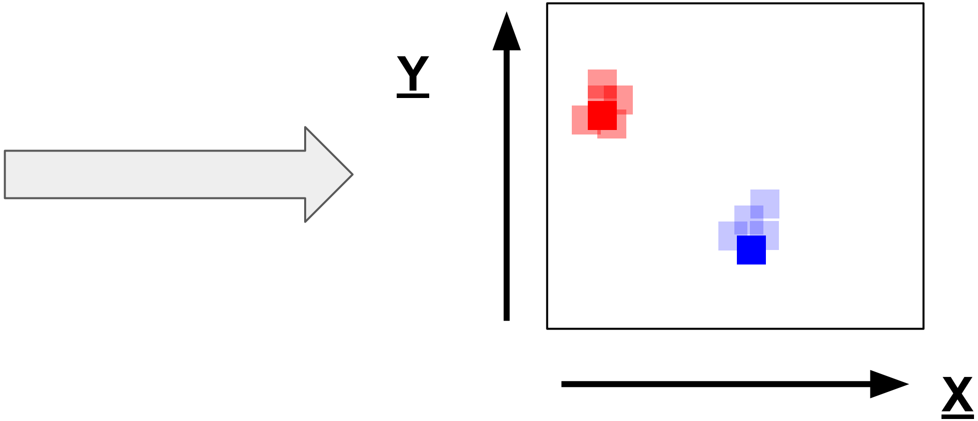


$$\breve{V}_k(x) = ||x - x_\alpha||^2 q_{\alpha k},$$

charg
k→unique id of th
x → latent space coordinat

Demand A: Points group together

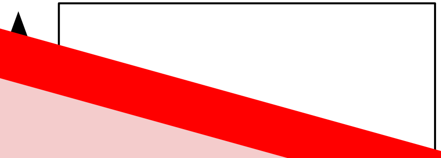




$$L_V = \frac{1}{N} \sum_{j=1}^{N} q_j \sum_{k=1}^{K} \left(M_{jk} \breve{V}_k(x_j) \right)$$

In words… for each pixel (*j*) we calculate its potential w.r.t each object (*k*) . If that pixel (*j*) is in object (*k*) , punish (increase the **Loss**) if (*j*) is far away … M_{jk} = 1 if (j) is in object (k) , else 0

Demand A: Points group together

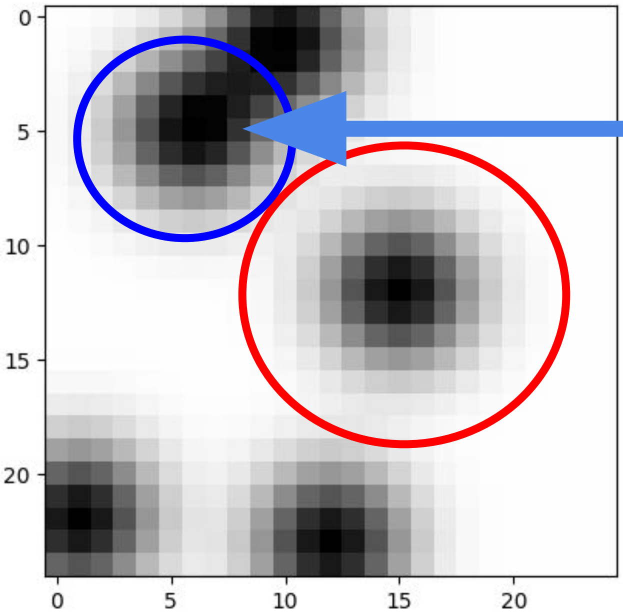


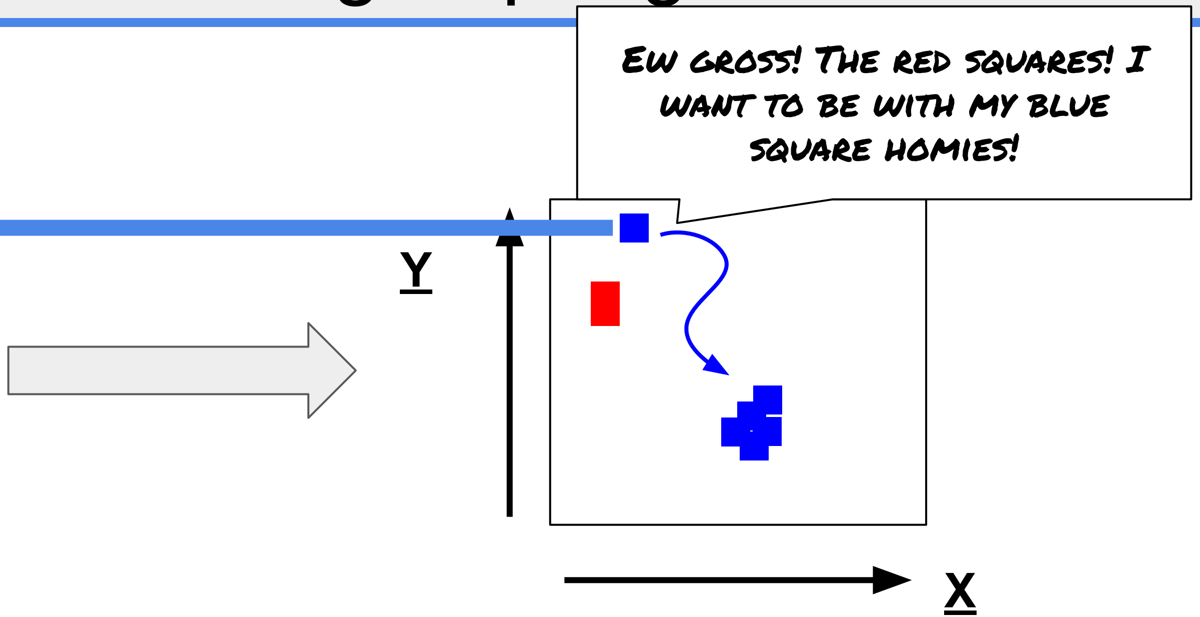
We must also "scare" away pixels from different objects so that they cluster elsewhere...

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K}\Big(M_{jk}\breve{V}_k(x_j)$$

In words...
potential w.r.t each ob
object (*k*) , punish (increase t
away … M_{jk} = 1 if (j) is in object (k) , el

Demand A: Points group together

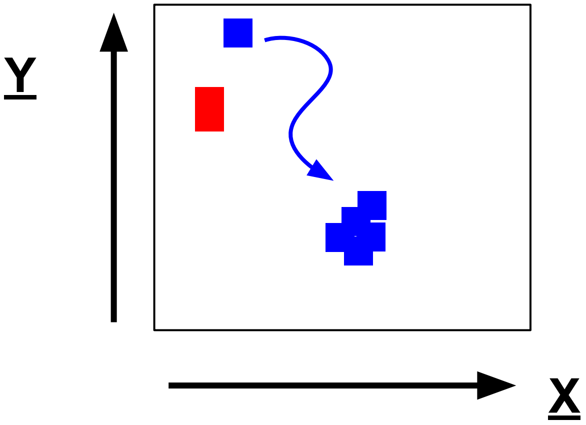




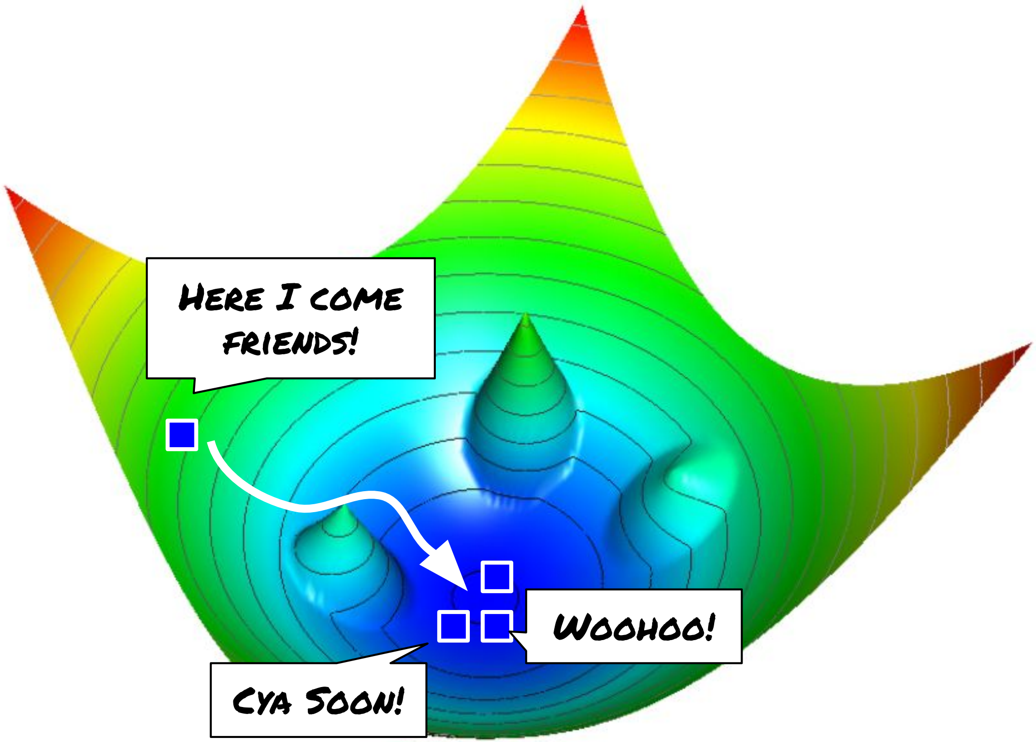
Punish the model if this repulsive term is large. Occurs when pixel (j) is near objects it is not affiliated with…

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K} \left(M_{jk}\breve{V}_k(x_j) + \boxed{(1-M_{jk})\hat{V}_k(x_j)}\right).$$

Demand A: Points group together



(Right) The total potential V experienced by the blue square as it navigates past 3 unaffiliated objects (peaked condensation points) towards its clustering home (the bottom of the well, another condensation point)



Demand B: Only one big beta per object

Recall… each pixel (j) learns a $0 < \beta_j < 1$ value
Need… Clustering to realize one pixel with a significantly larger beta than the rest
Why… Threshold the beta → get the predicted features

Demand B: Only one big beta per object

Recall… each pixel (j) learns a $0 < \beta_j < 1$ value
Need… Clustering to realize one pixel with a significantly larger beta than the rest
Why… Threshold the beta → get the predicted features

$$L_\beta = \boxed{\frac{1}{K}\sum_k (1-\beta_{\alpha k})} + \boxed{s_B \frac{1}{N_B}\sum_i^N n_i \beta_i},$$

Punish the model if the largest β per object (k) is small

Punish the model if background pixels even *think* about forming a condensation point (high mean β of the background)

n_i → 1 if point is background, else 0

Demand C: Highest β predicts features best

Recall… For each object (k) we will determine the features by reading them off of the pixel with the highest β

Demand C: Highest β predicts features best

Recall… For each object (k) we will determine the features by reading them off of the pixel with the highest β

Let…

t_i → True value for pixel (i)

p_i → Predicted value for pixel (i)

n_i → 1 if pixel (i) is background, else 0

$L(t_i , p_i)$ →★**User**★ defined custom loss function (ex: MSE for regression tasks)

Demand C: Highest β predicts features best

Recall… For each object (k) we will determine the features by reading them off of the pixel with the highest β

Let…

t_i → True value for pixel (i)

p_i → Predicted value for pixel (i)

n_i → 1 if pixel (i) is background, else 0

$L(t_i , p_i)$ →★**User**★ defined custom loss function (ex: MSE for regression tasks)

$$L_p = \frac{1}{\sum_{i=1}^{N} \xi_i} \cdot \sum_{i=1}^{N} L_i(t_i, p_i)\ \xi_i, \text{ with}$$

$$\xi_i = (1 - n_i)\, \text{arctanh}^2\, \beta_i .$$

Demand C: Highest β predicts features best

Recall… For each object (k) we will determine the features by reading them off of the pixel with the highest β

Let…

t_i → True value for pixel (i)

p_i → Predicted value for pixel (i)

n_i → 1 if pixel (i) is background, else 0

$L(t_i, p_i)$ →★**User**★ defined custom loss function (ex: MSE for regression tasks)

$$L_p = \frac{1}{\sum_{i=1}^{N} \xi_i} \cdot \sum_{i=1}^{N} \underline{L_i(t_i, p_i)}\, \xi_i, \text{ with}$$

$$\xi_i = (1 - n_i) \operatorname{arctanh}^2 \beta_i .$$

Punish the model for the loss of each non-background pixel

Demand C: Highest β predicts features best

Recall… For each object (k) we will determine the features by reading them off of the pixel with the highest β

Let…

$t_i \rightarrow$ True value for pixel (i)

$p_i \rightarrow$ Predicted value for pixel (i)

$n_i \rightarrow$ 1 if pixel (i) is background, else 0

$L(t_i , p_i) \rightarrow$ ★**User**★ defined custom loss function (ex: MSE for regression tasks)

$$L_p = \frac{1}{\sum_{i=1}^{N} \xi_i} \cdot \sum_{i=1}^{N} L_i(t_i, p_i)\ \underline{\xi_i}, \text{ with}$$

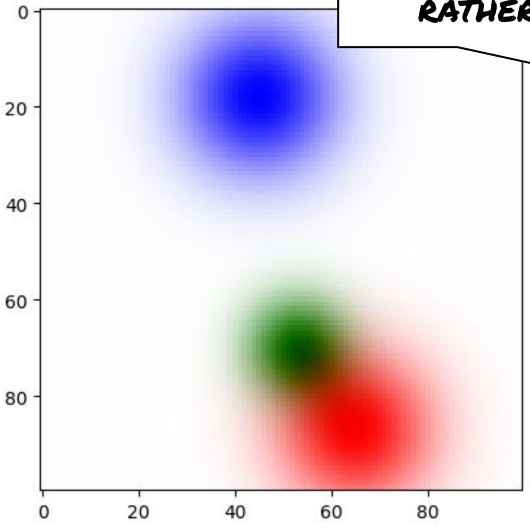
$$\xi_i = (1 - n_i) \operatorname{arctanh}^2 \beta_i .$$

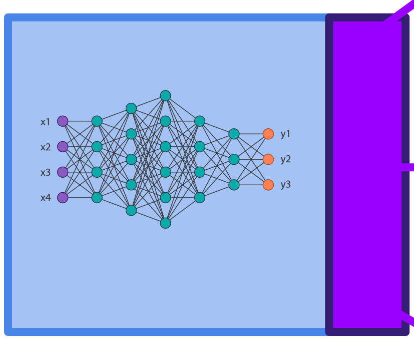
Punish the model for the loss of each non-background pixel

Punish it more when the pixel has a large β (high ξ)

★Object Condensation★

"I will learn to produce a ($\underline{X}$ $\underline{Y}$ β p) for each point based on your three, rather bizarre, requirements!"





Distinct Clusters

$$L_V = \frac{1}{N}\sum_{j=1}^{N} q_j \sum_{k=1}^{K} \left(M_{jk}\check{V}_k(x_j) + (1-M_{jk})\hat{V}_k(x_j)\right).$$

Only one representative

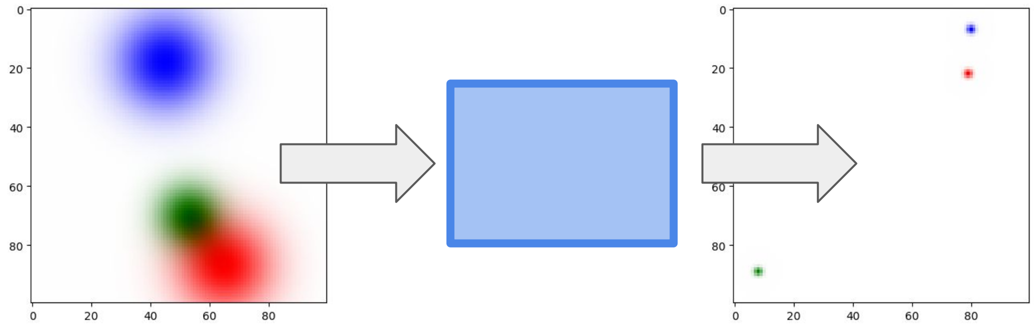
$$L_\beta = \frac{1}{K}\sum_{k}(1-\beta_{\alpha k}) + s_B \frac{1}{N_B}\sum_{i}^{N} n_i \beta_i,$$

Rep. carries features

$$L_p = \frac{1}{\sum_{i=1}^{N}\xi_i} \cdot \sum_{i=1}^{N} L_i(t_i, p_i)\,\xi_i,$$

$$\boxed{L = L_p + s_c(L_\beta + L_V).}$$

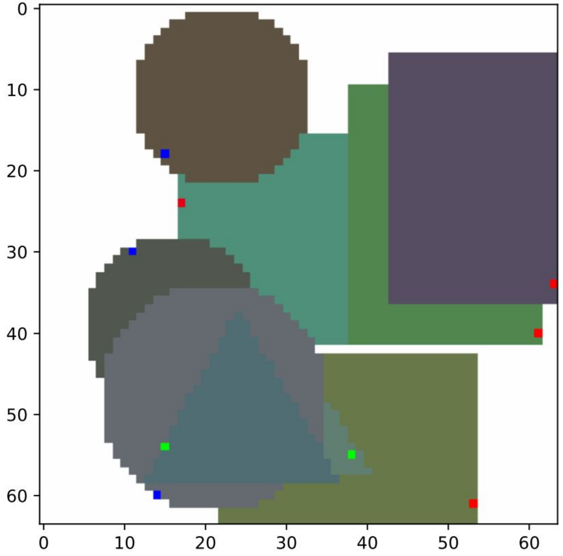
Inference



How does the paper recommend we extract the features for each object?

1. Pass your input through the **frontend ML architecture**
    a. Get ($\underline{\mathbf{X}}$,$\underline{\mathbf{Y}}$,β,p) for each point in our point cloud
2. Label all points with $\beta > t_\beta$ as condensation points ($t_\beta \approx 0.1$)
3. Assign all vertices within $t_d \approx [0.1\ ,\ 1]$ in the latent space to the condensation point
4. Take the features *p* of the condensation points

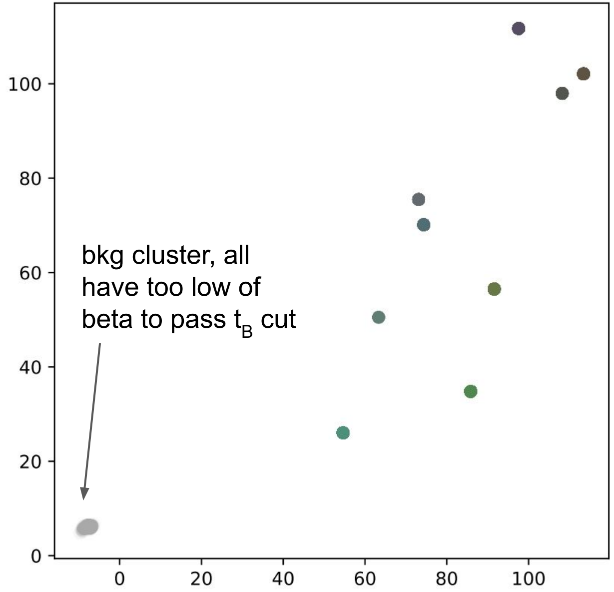
Interesting Example



9 objects

Classifier

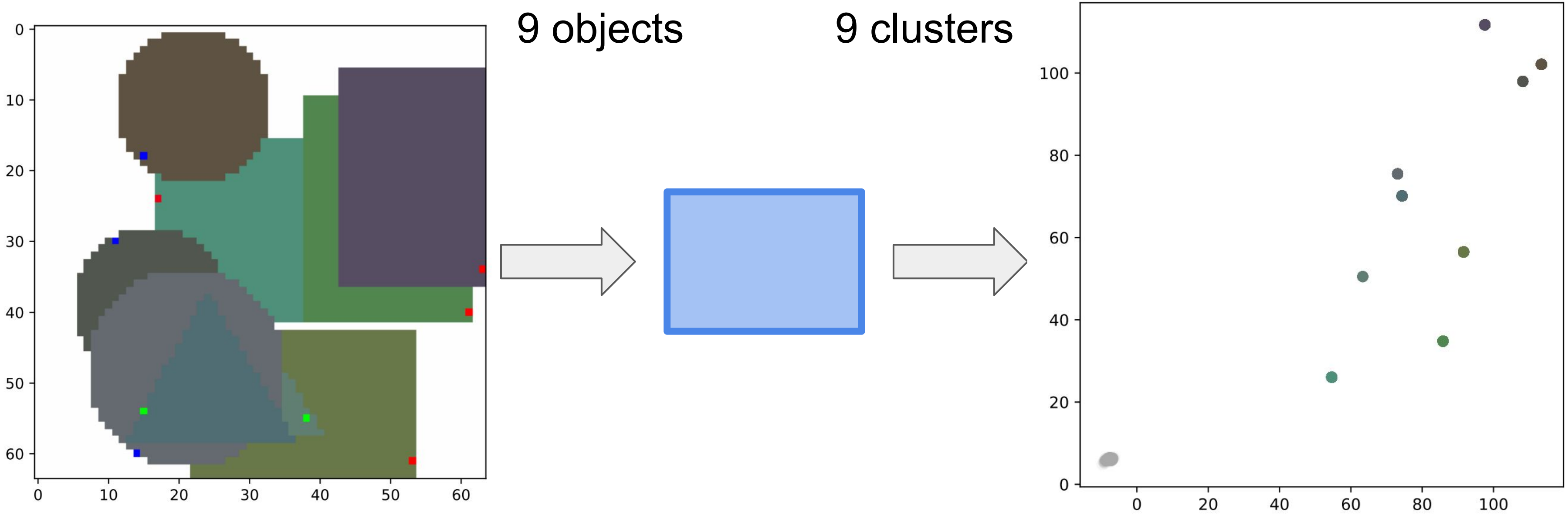
9 clusters



Input Space

Latent Space

Interesting Example



(Left Figure) The standout **red green blue** pixels are the condensation points for each object (largest β).

We can infer that the model learns that an edge, or when visible, a corner pixel of an object will carry the object's features most effectively

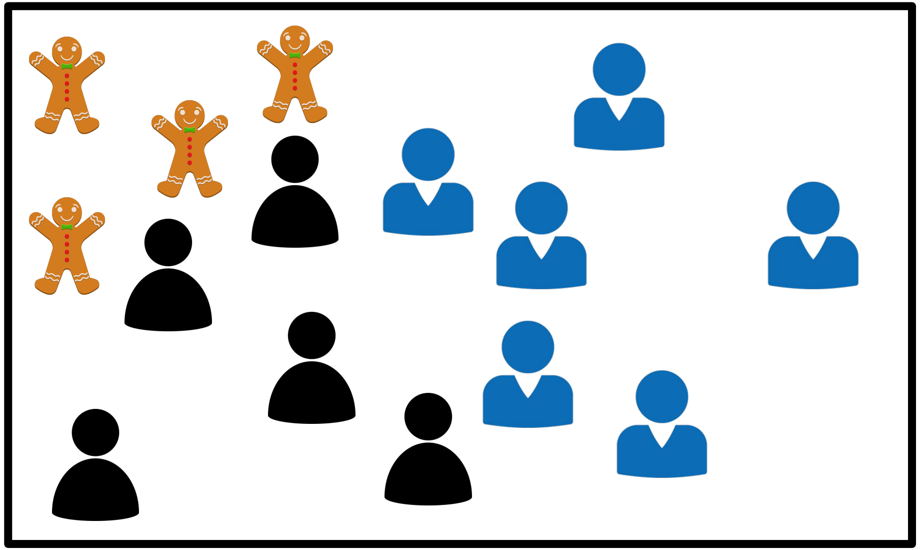
Summary

★Object Condensation★ provides a framework that, when attached to a frontend ML architecture, can assist simultaneously with <u>clustering</u> and <u>feature predicting</u> multiple objects in sparse datasets (point clouds)

The ML architecture learns to…

- Cluster points of like-objects with one another in a new latent space
- Assign one representative per object by giving it a large β value
- Focus on having points with large β give the best object feature predictions (centroid of a calorimeter cluster, momentum of a particle through a tracking system, etc.)

Greg’s General Idea



 Employee

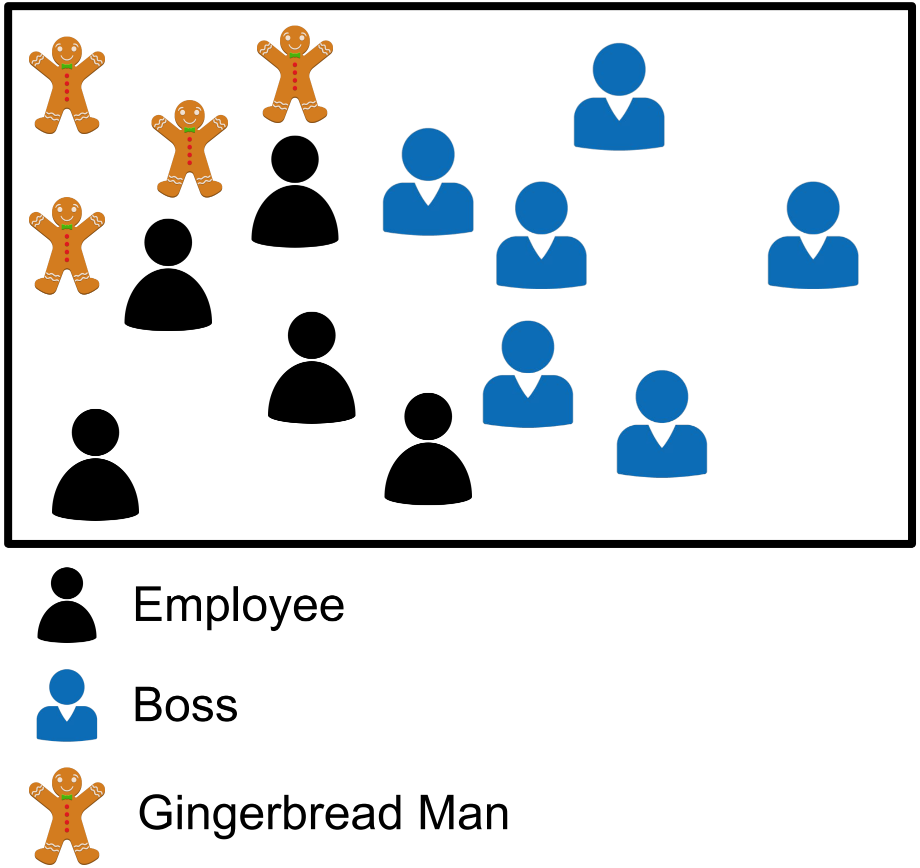
 Boss

 Gingerbread Man

Posed Problem:
We have any number of discrete clusters of people (3 currently)

We want to predict how much space each cluster is currently taking up in the room (total energy deposited in calorimeter?)

Greg’s General Idea



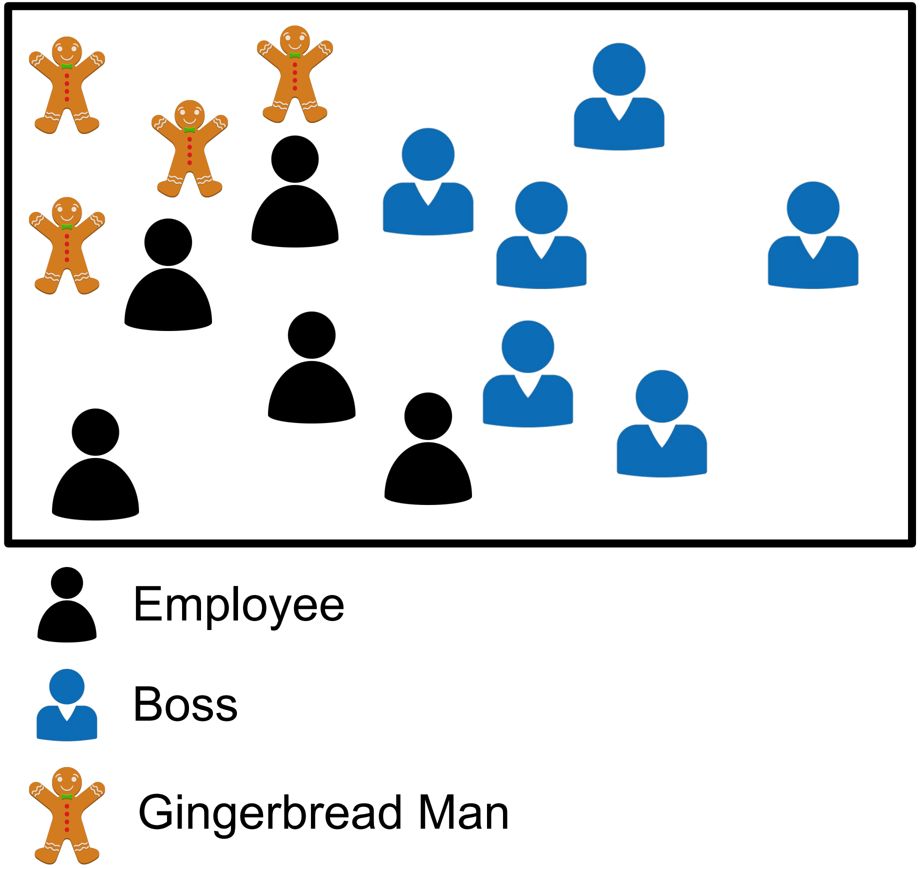
Object Condensation learns a $0 < \beta_i < 1$ value for *each* vertex

$\beta_i \rightarrow$ A measure of how likely point *i* is a condensation point

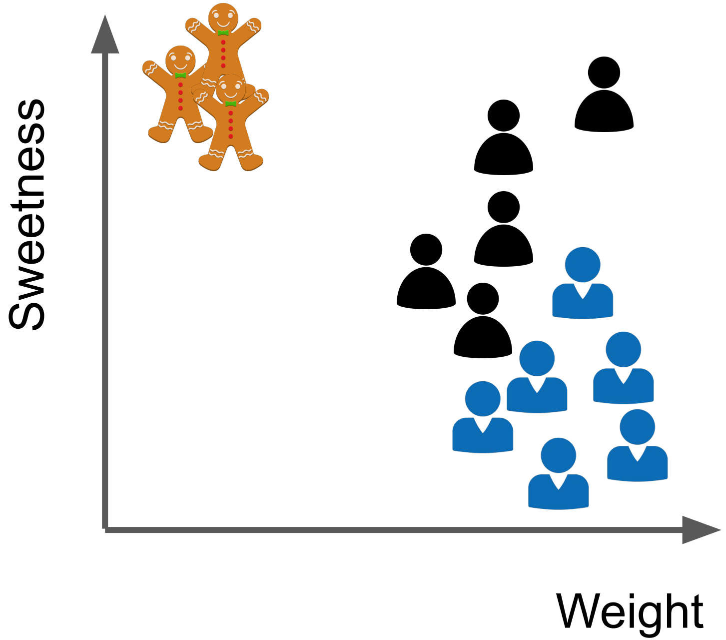
A condensation point, we can imagine, is the most archetypal representative of the distinct object

Ex: The Most Bossiest Boss in the room will have a condensation point near 1, and the other bosses will learn smaller β’s

Greg's General Idea



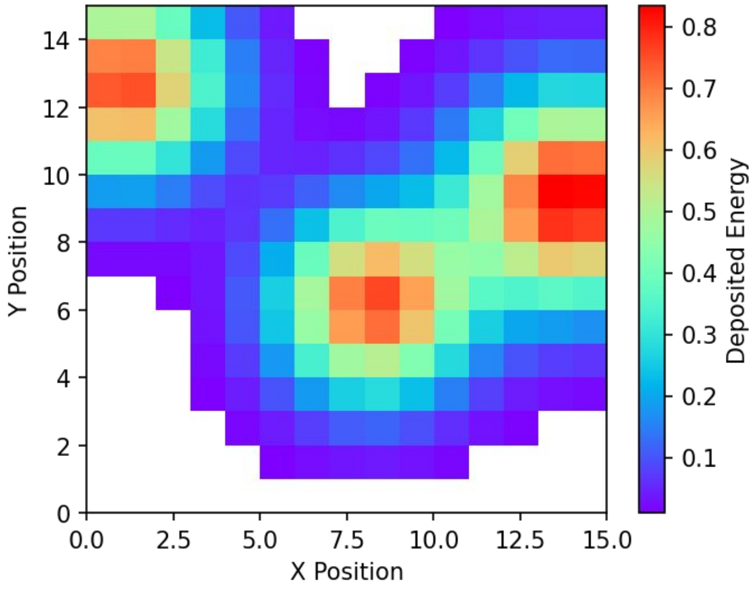
Object Condensation also maps the input features to a latent space to help distinguish between different objects



Object Condensation (Training)

Given… N_F “hot” pixels (x,y,E) and N_B bkg

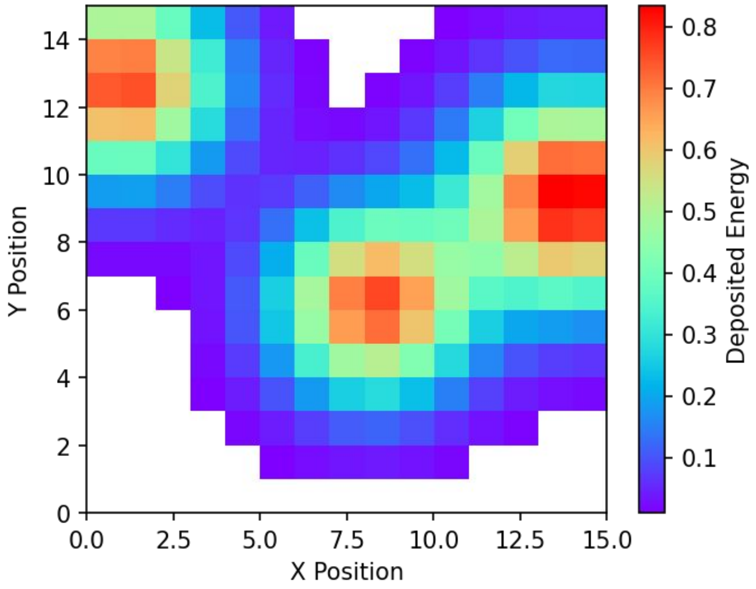
Predict… Which pixels should be clustered together and what is the total energy and centroid of the showering particle



Object Condensation (Training)

Given… N_F "hot" pixels (x,y,E) and N_B bkg

Predict… Which pixels should be clustered together and what is the total energy (E) and centroid (x,y) of the showering particle

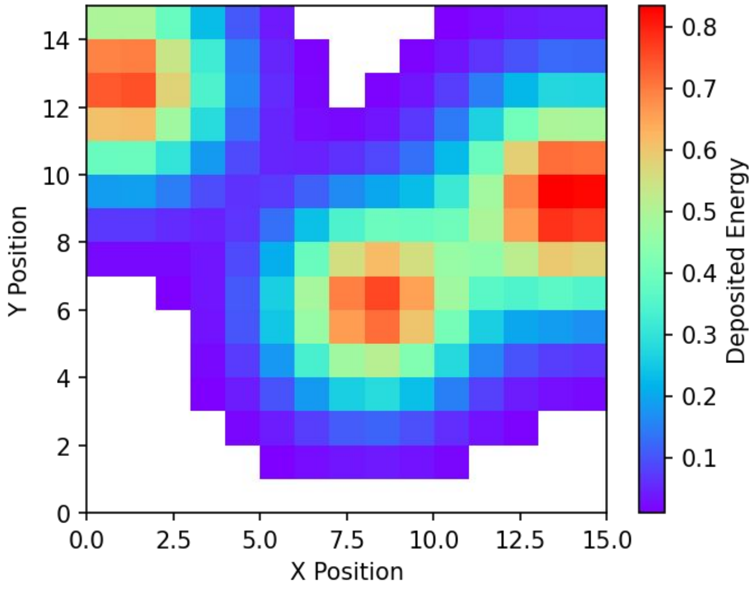


- This is a point cloud with N=225. Each vertex has an input dimension of 3.

Object Condensation (Training)

Given… N_F “hot” pixels (x,y,E) and N_B bkg

Predict… Which pixels should be clustered together and what is the total energy (E) and centroid (x,y) of the showering particle



- ➢ This is a point cloud with N=225. Each vertex has an input dimension of 3.

- ➢ The final output space will be 225 vertices. Each with dimension (d’=3) + 2 + 1 = 6