INTRODUCTION TO MACHINE LEARNING: DEEP LEARNING

MICHELLE KUCHERA

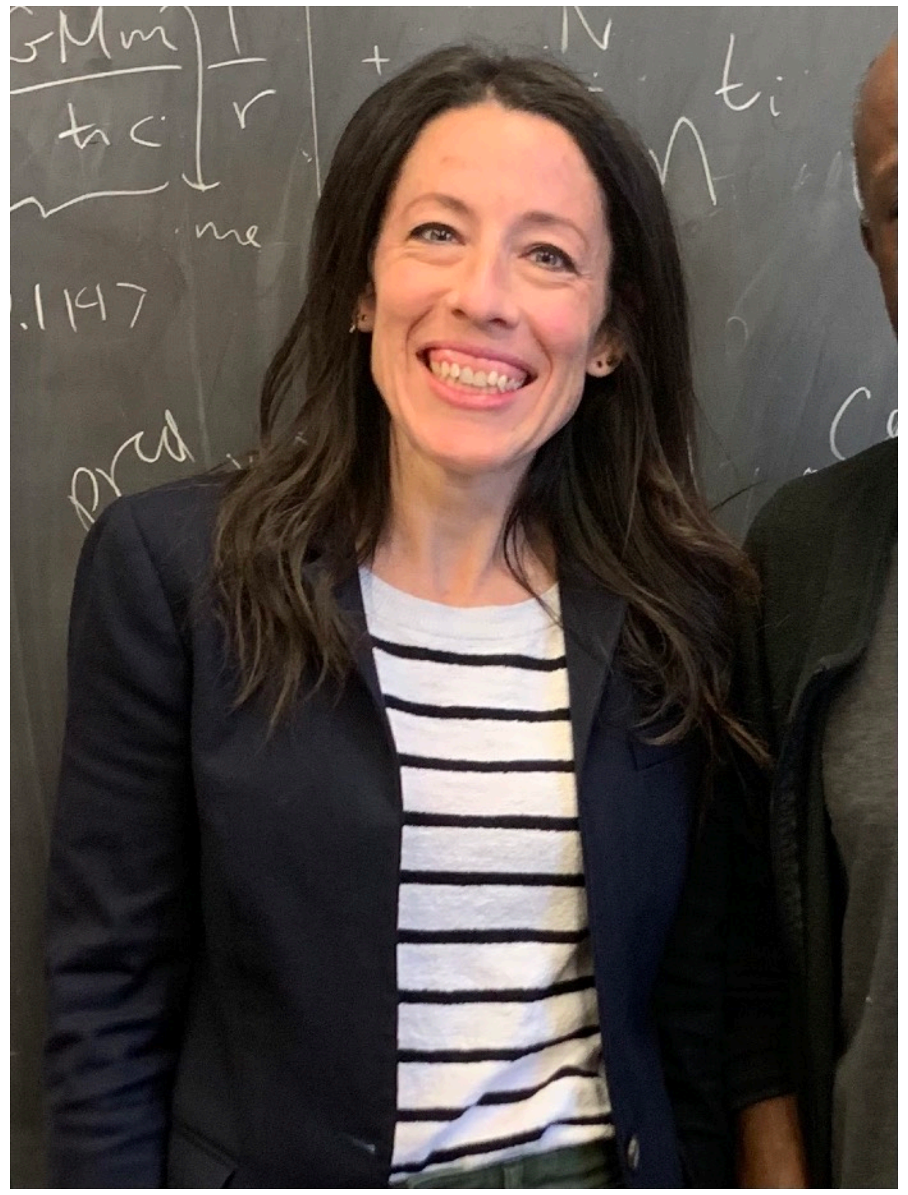
JOINT ICTP-IAEA ADVANCED SCHOOL/WORKSHOP ON MACHINE LEARNING IN CITIZEN SCIENCE

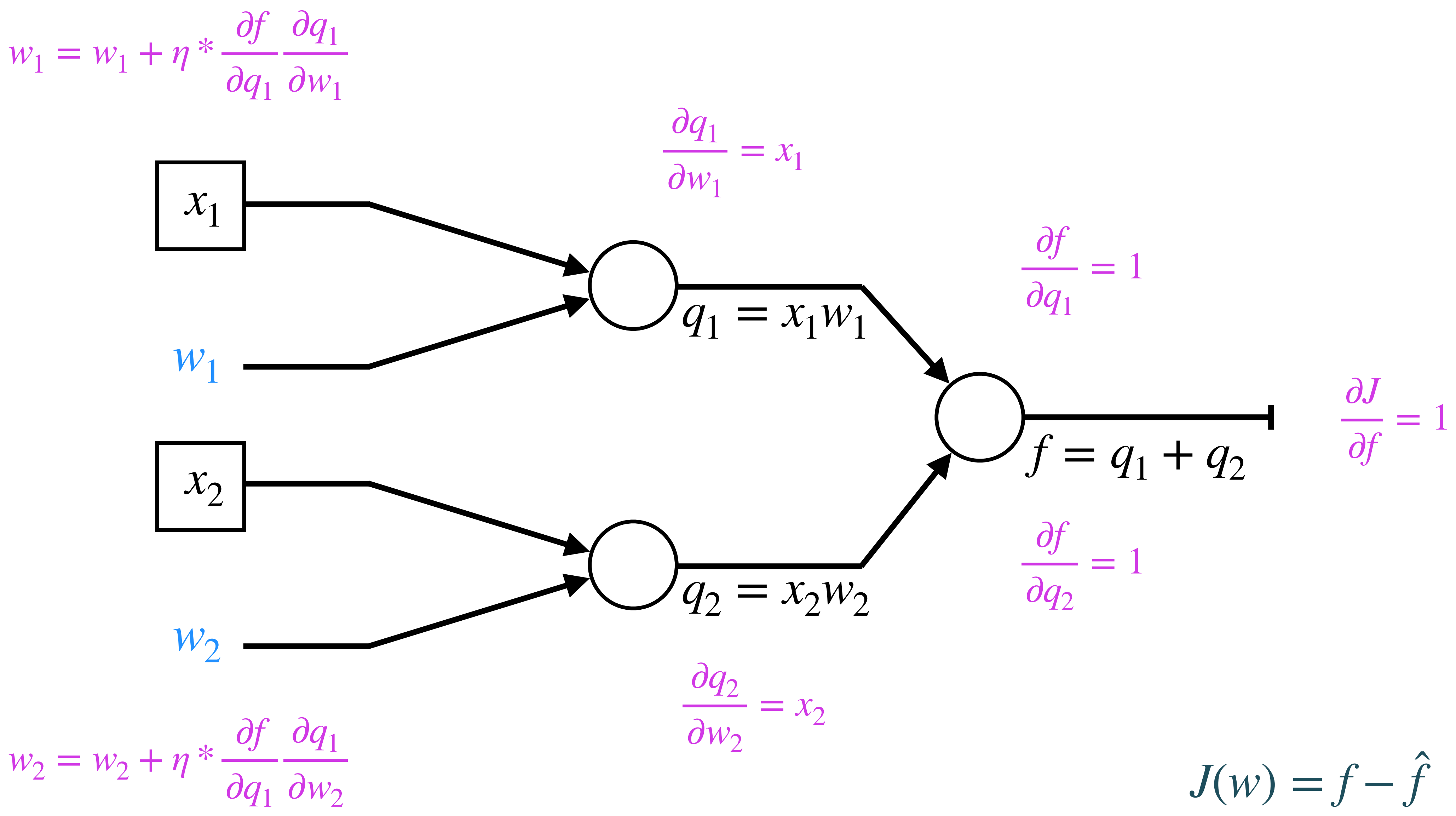
28 FEBRUARY 2023

ALPhA
DAVIDSON COLLEGE
DAVIDSON

MICHELLE KUCHERA

B.S., M.S. PHYSICS
M.S., PH.D. COMPUTATIONAL SCIENCE





ALPhA DAVIDSON COLLEGE

CERN

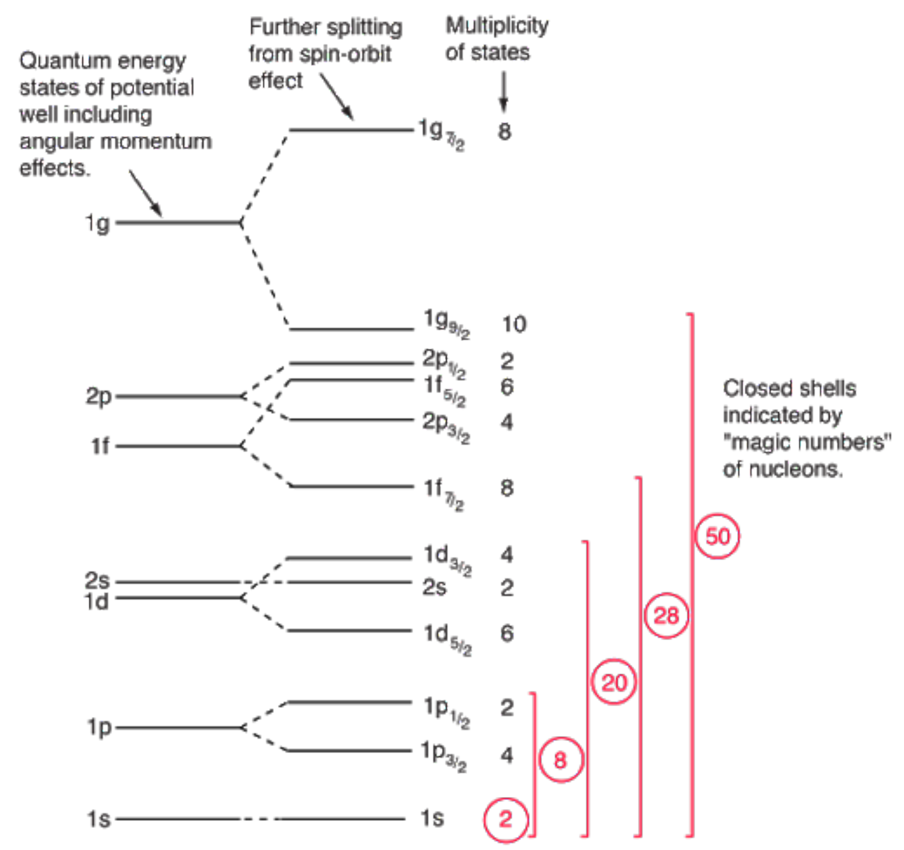
CMS Compact Muon Solenoid

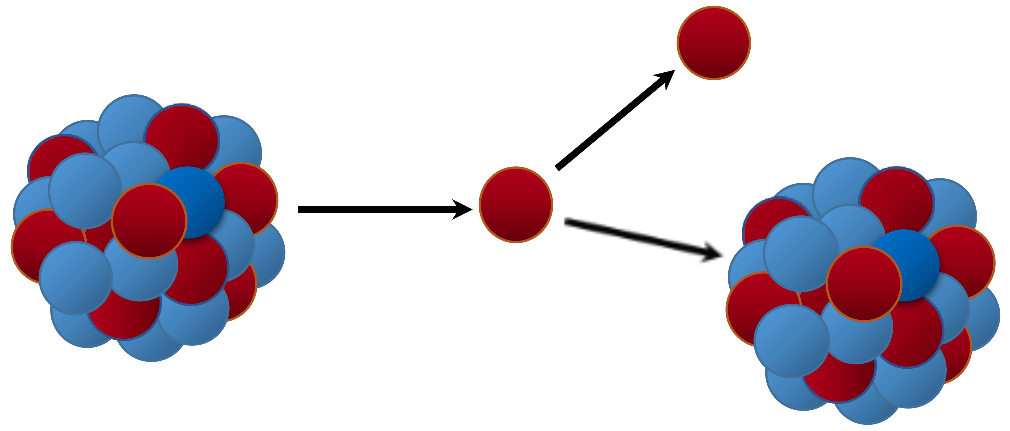
Jefferson Lab

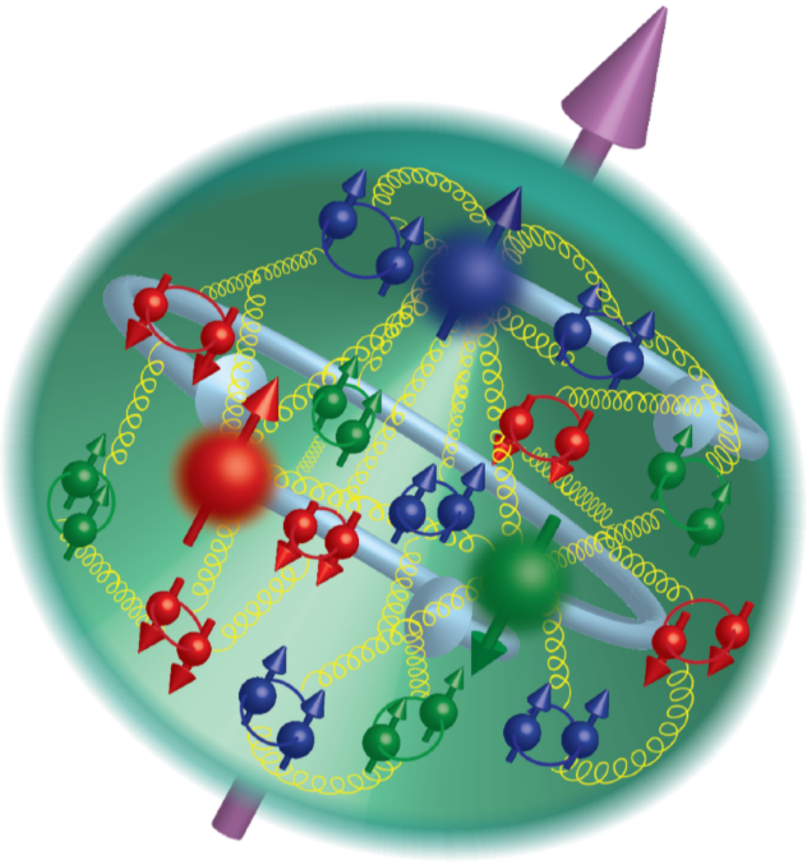
FRIB

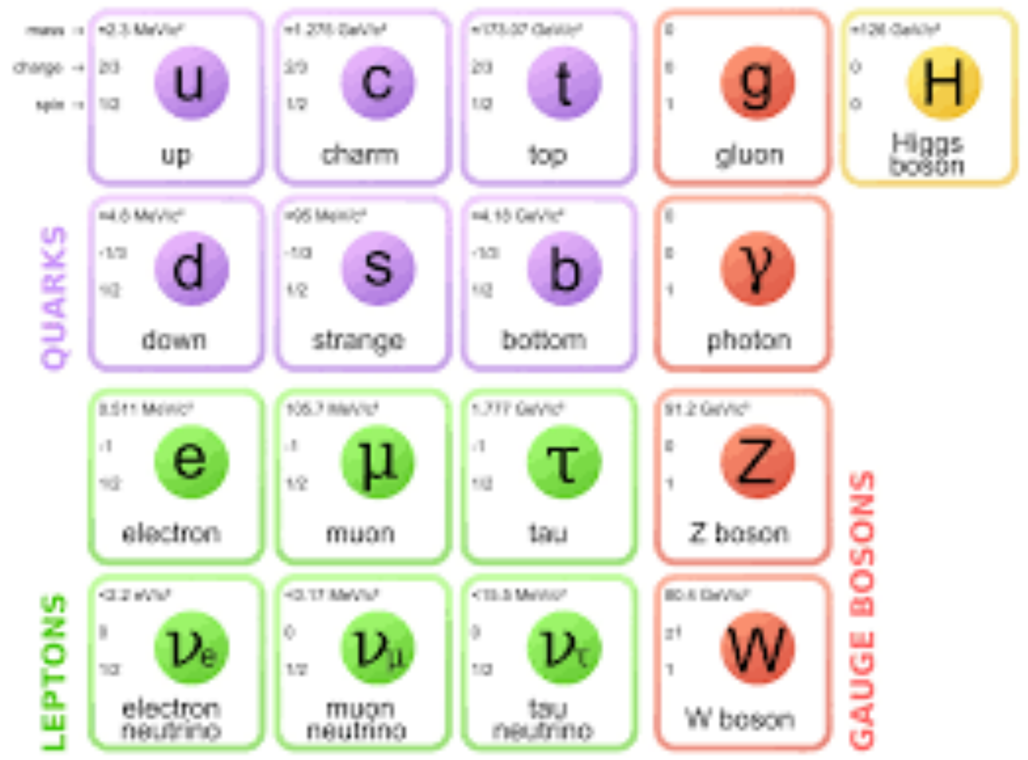
DAVIDSON

NSCL







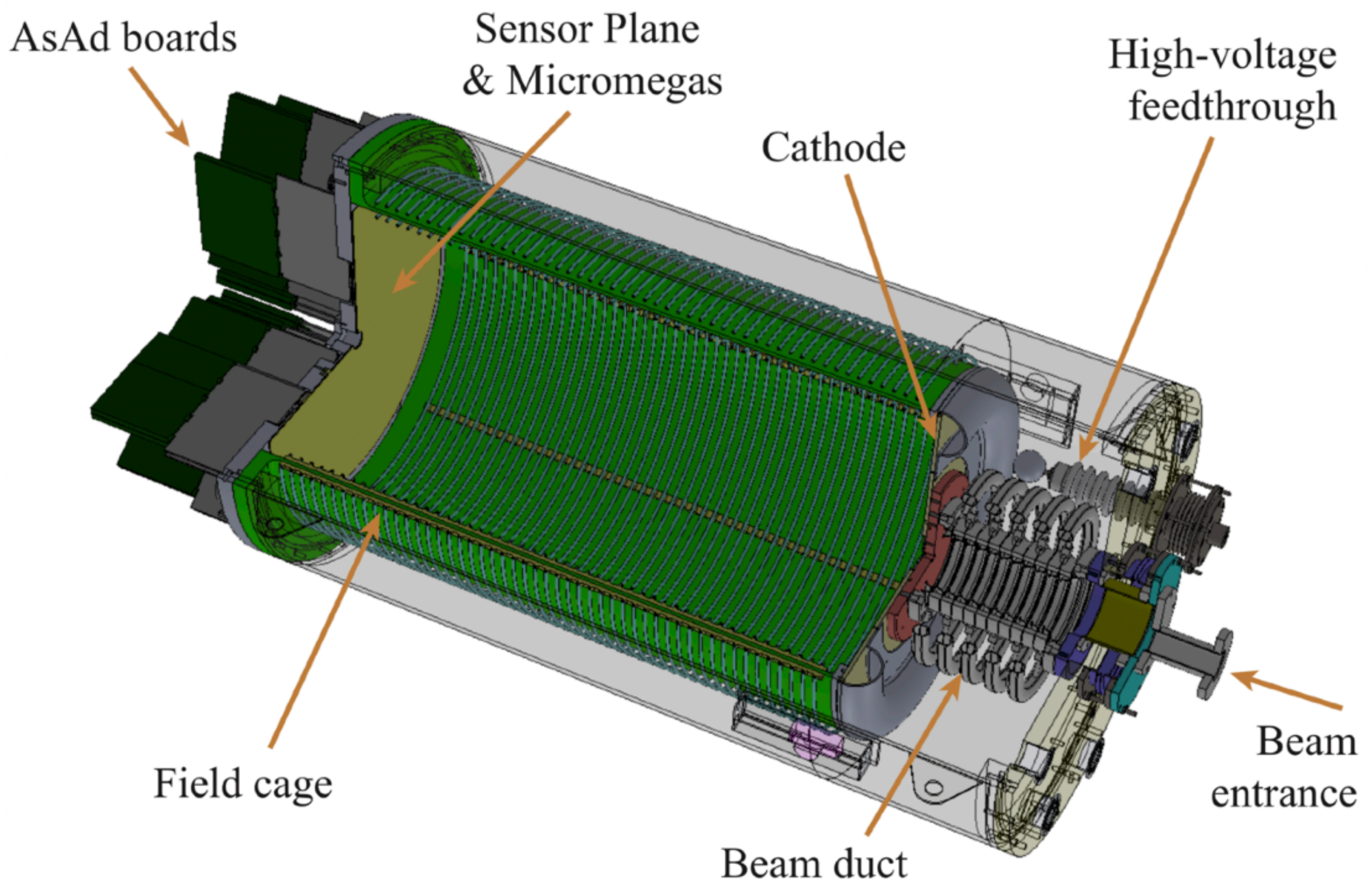


FRIB

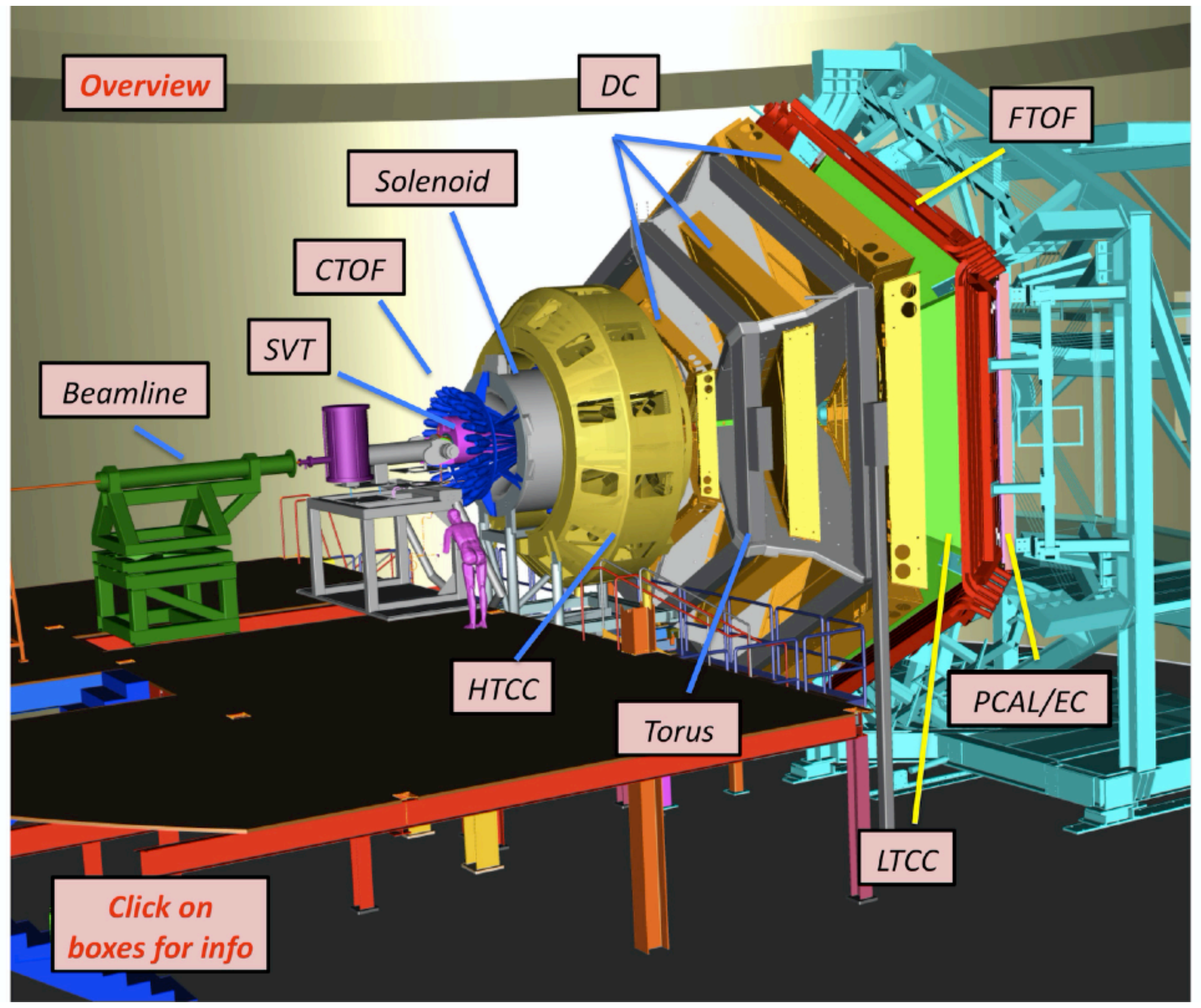
Jefferson Lab

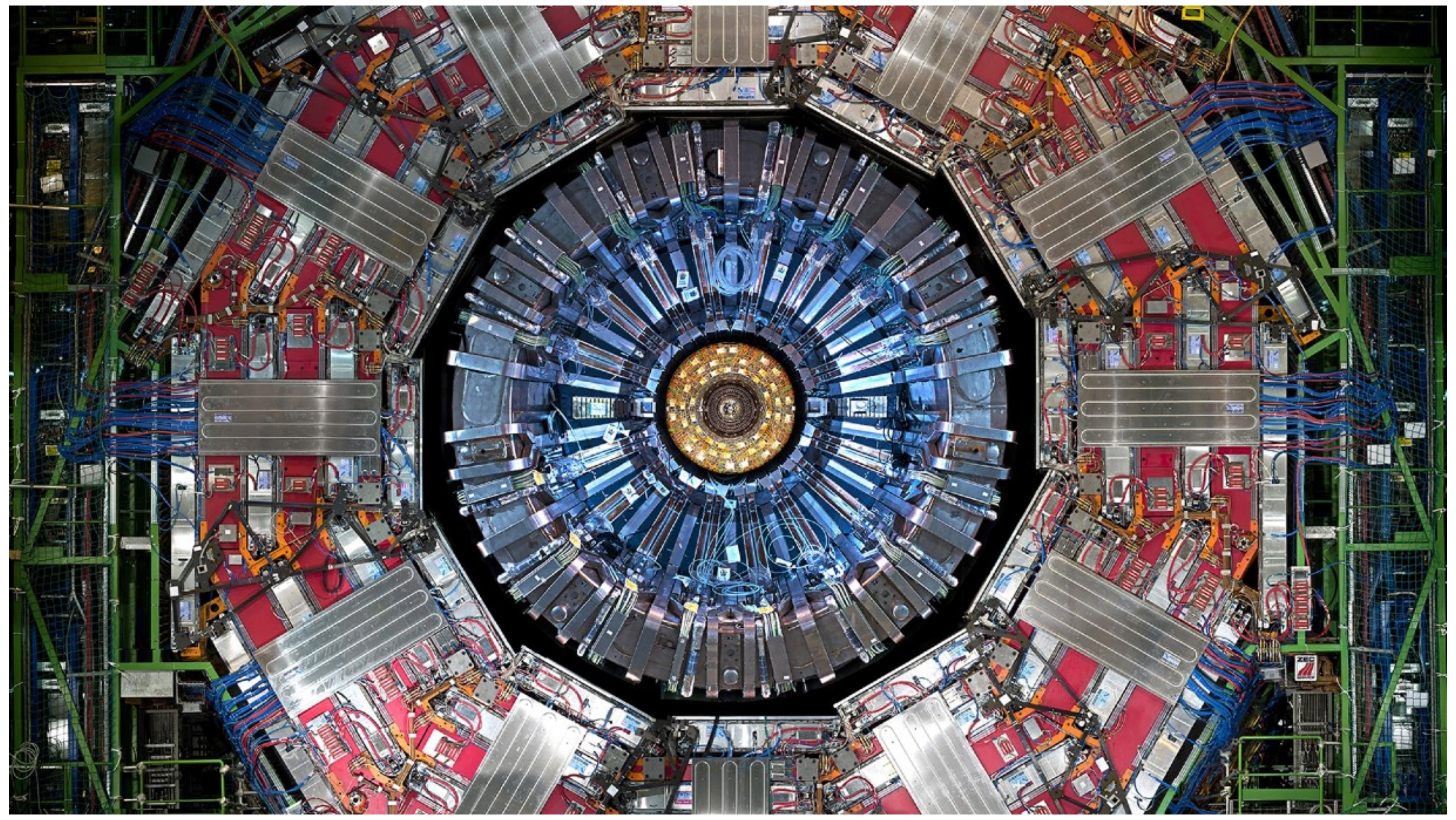
CERN

EXPERIMENTAL DATA



J. BRADT ET. AL., *NUCLEAR INSTRUMENTS AND METHODS*, 2017.



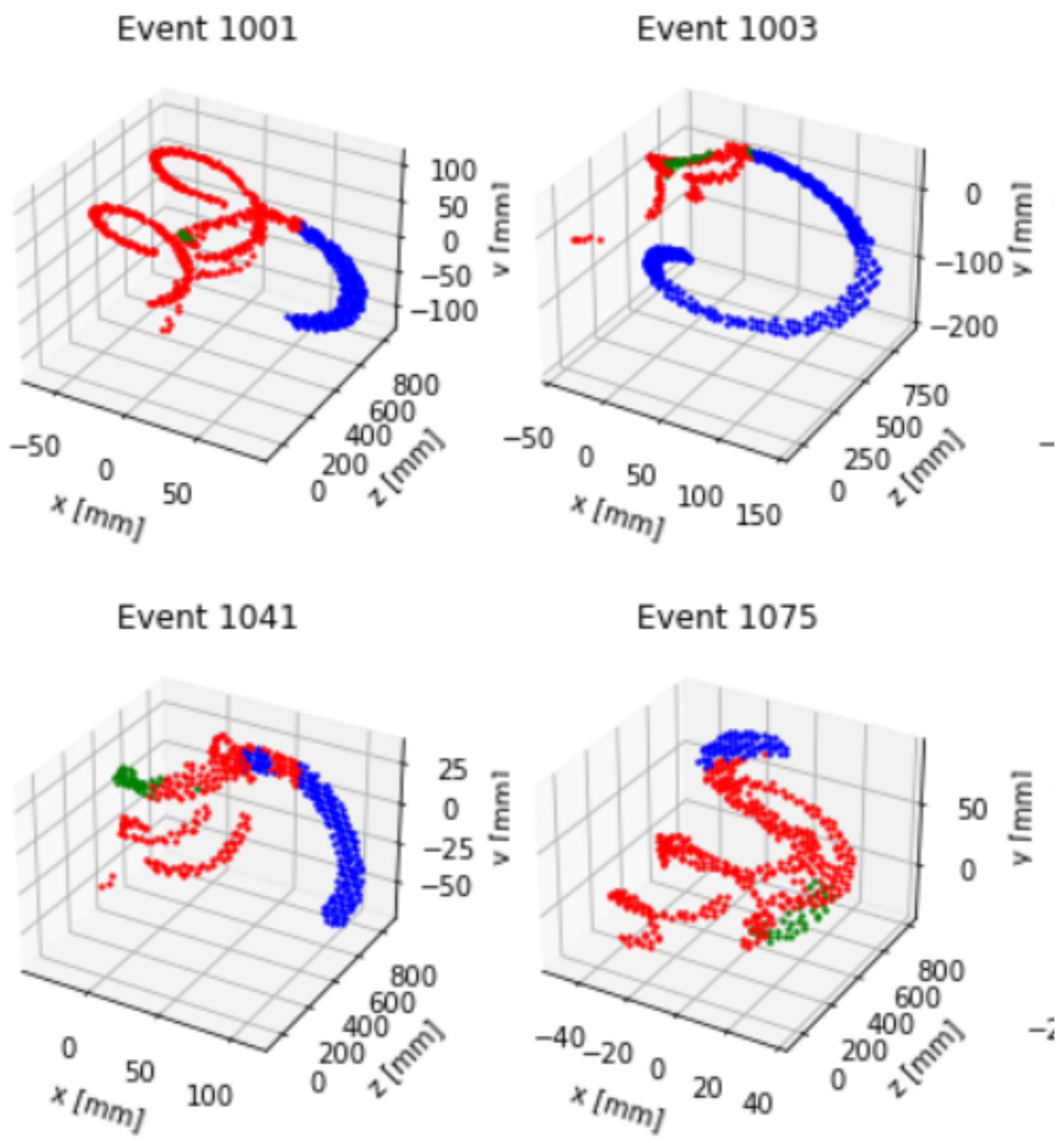


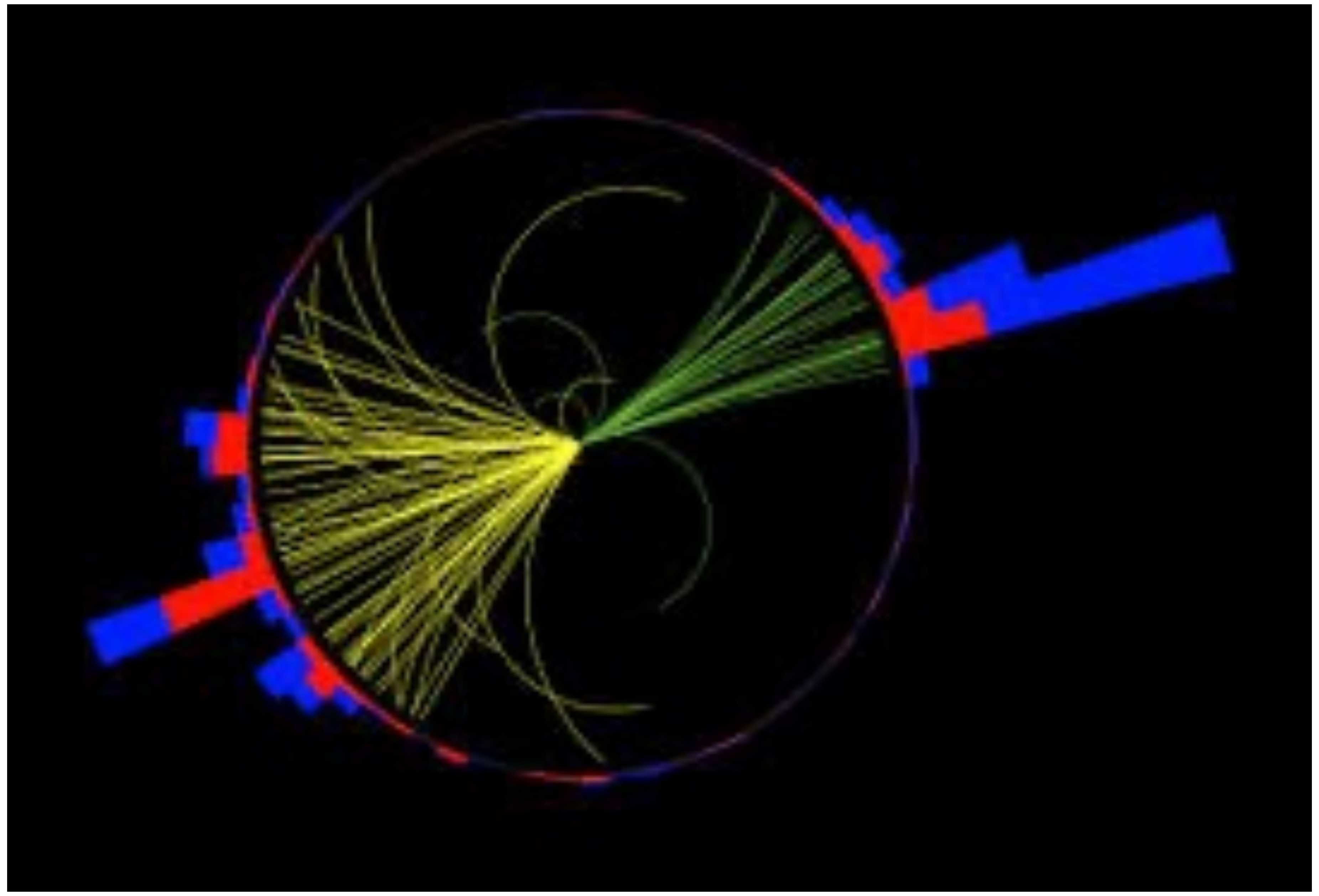
AT-TPC

CLAS 12

CMS

EXPERIMENTAL DATA

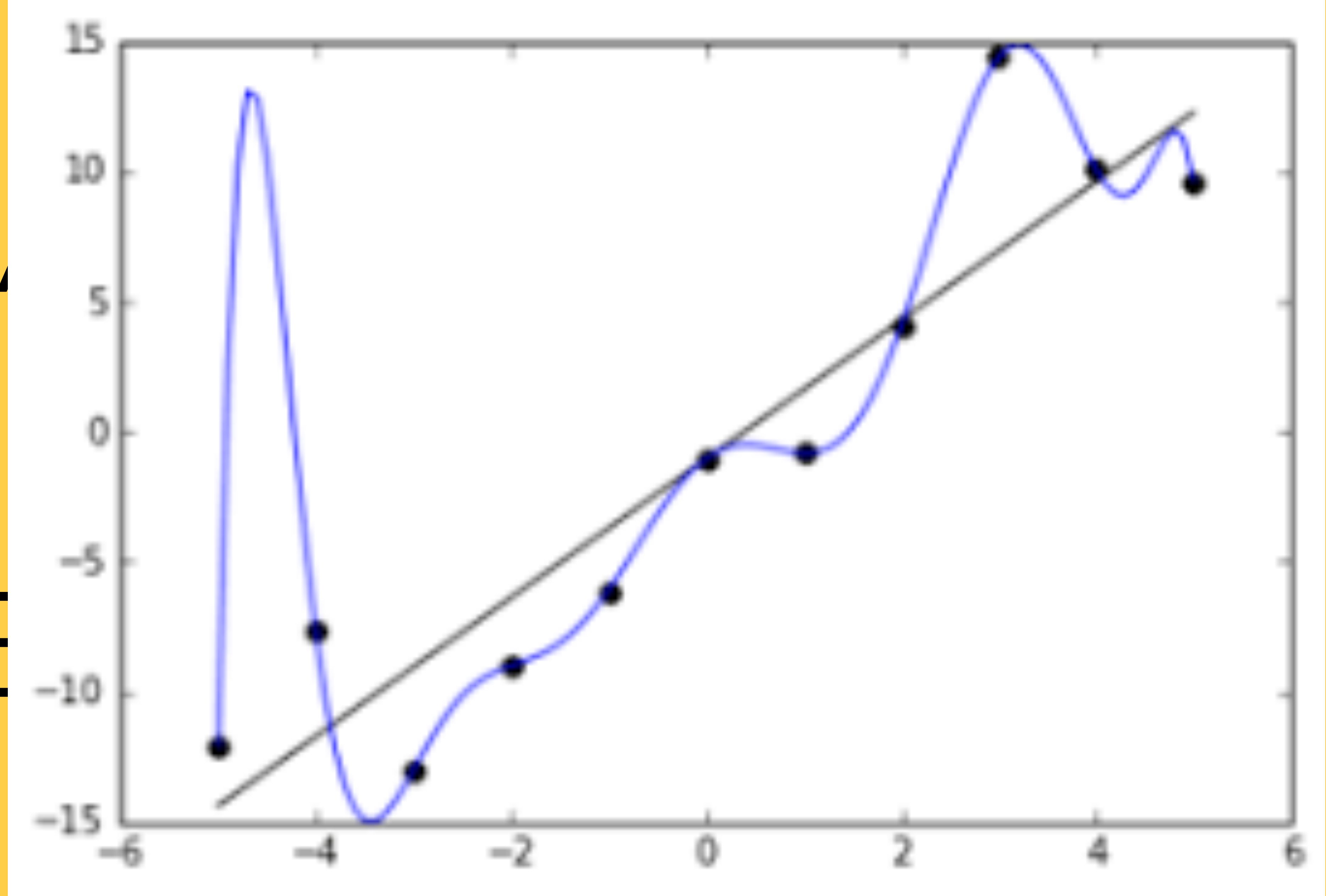




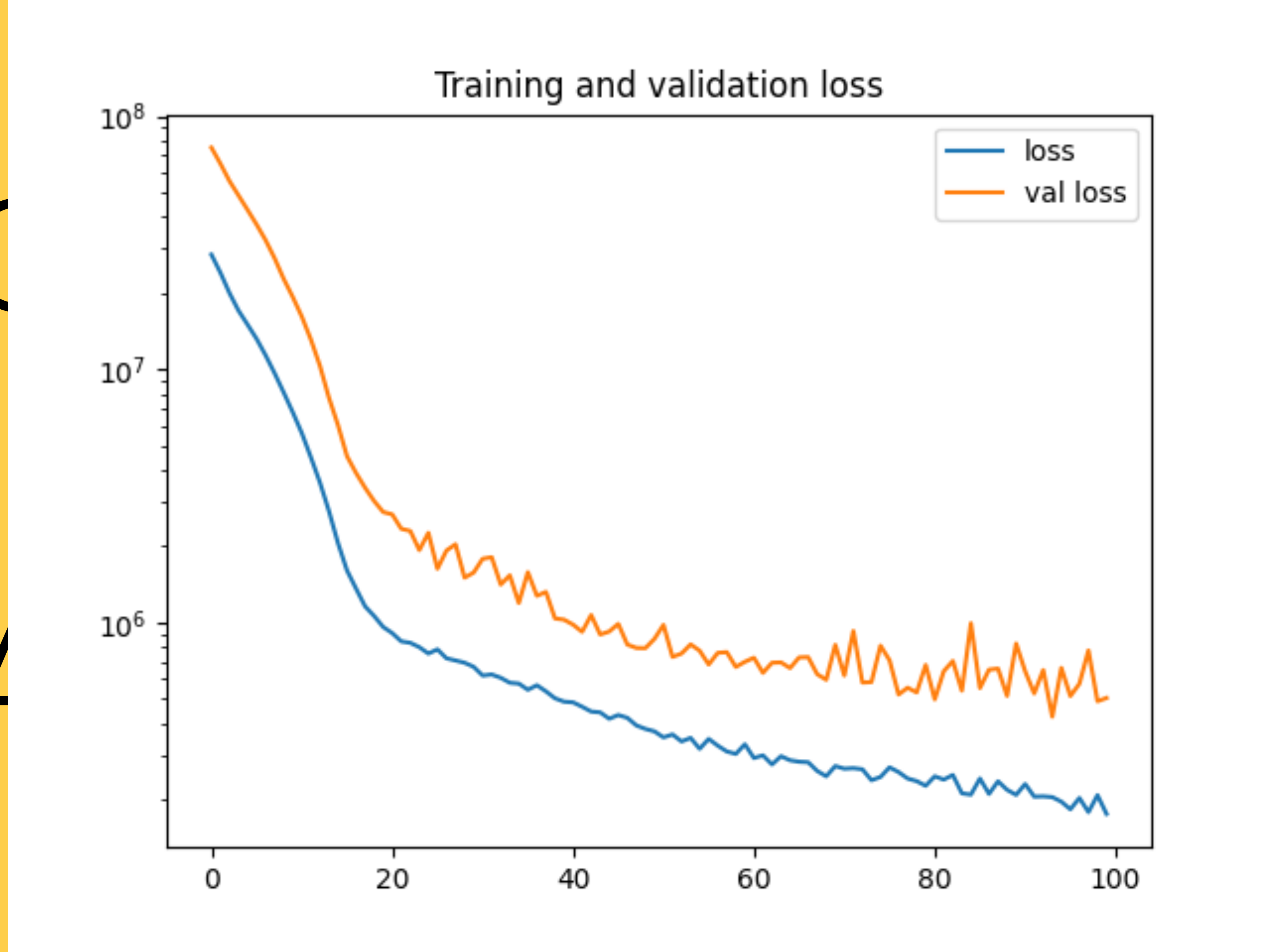
AT-TPC

CLAS 12

CMS

MA 
15
10
5
0
−5
−10
−15
−6
−4
−2
0
2
4
6
ING

LE DA


Training and validation loss
loss
val loss
10^8
10^7
10^6
0
20
40
60
80
100

GENERATIVE MODELING

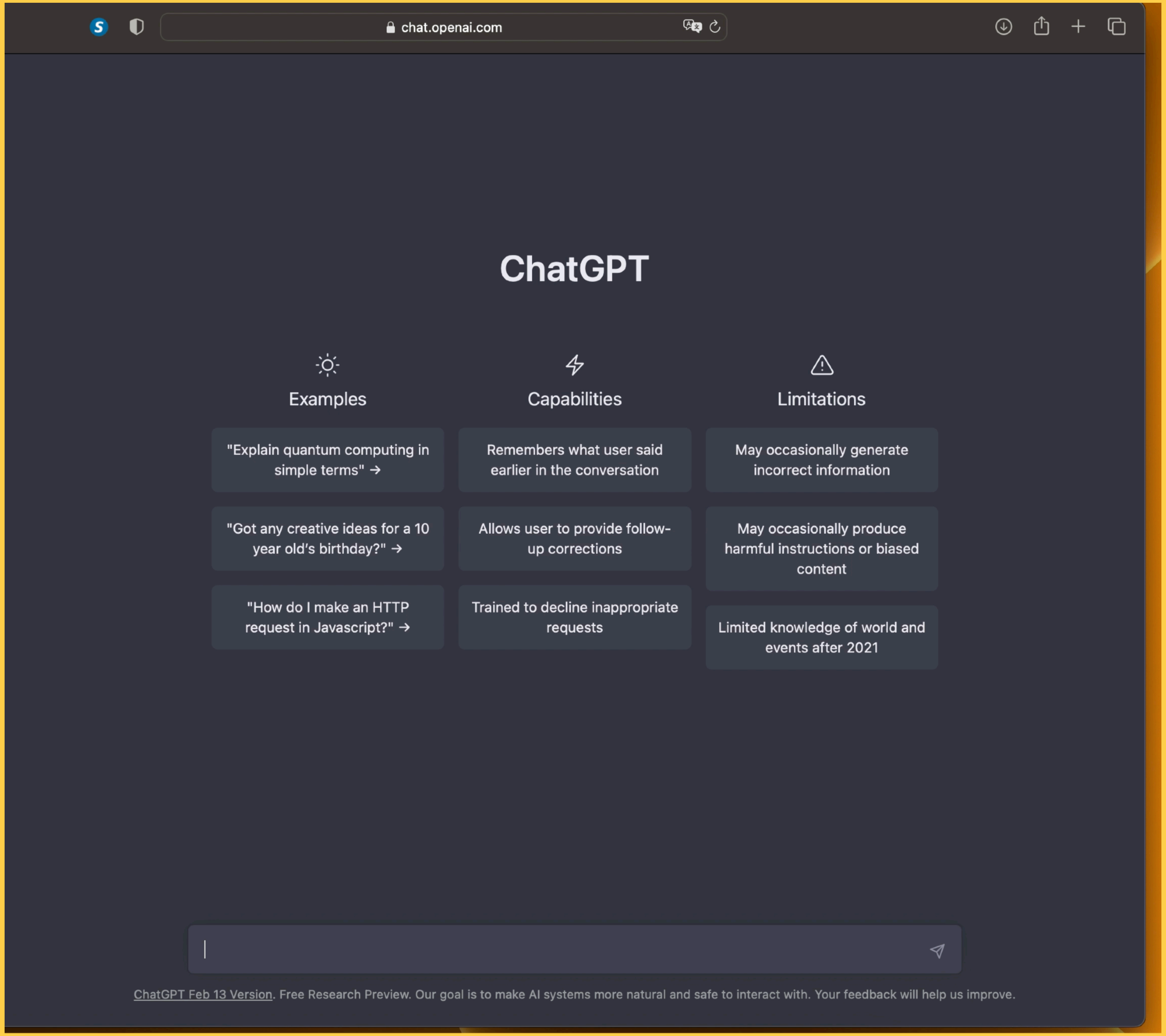
REGRESSION

CLASSIFICATION

GENERATIVE MODELING

REGRESSION

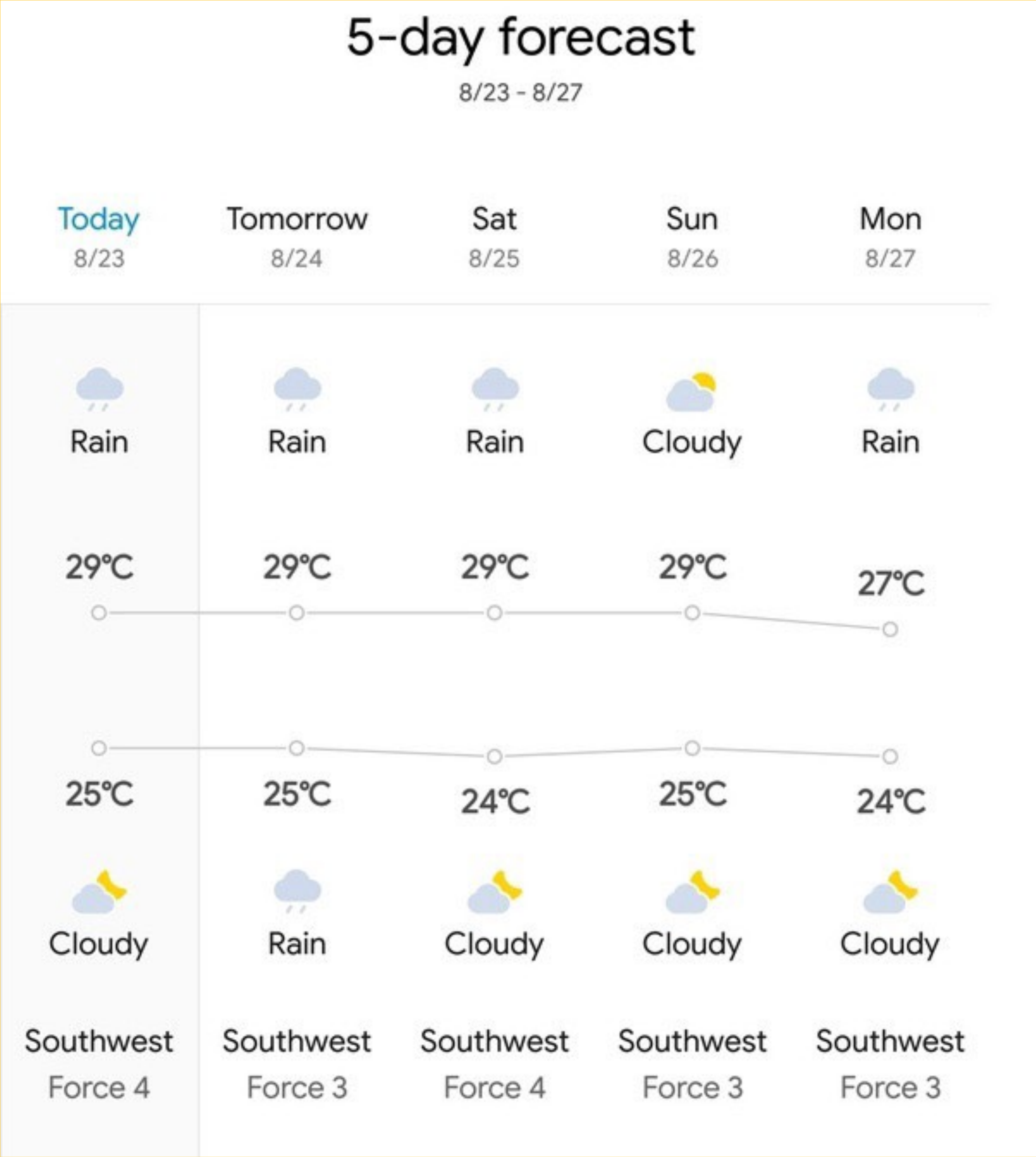
CLASSIFICATION



GENERATIVE MODELING

REGRESSION

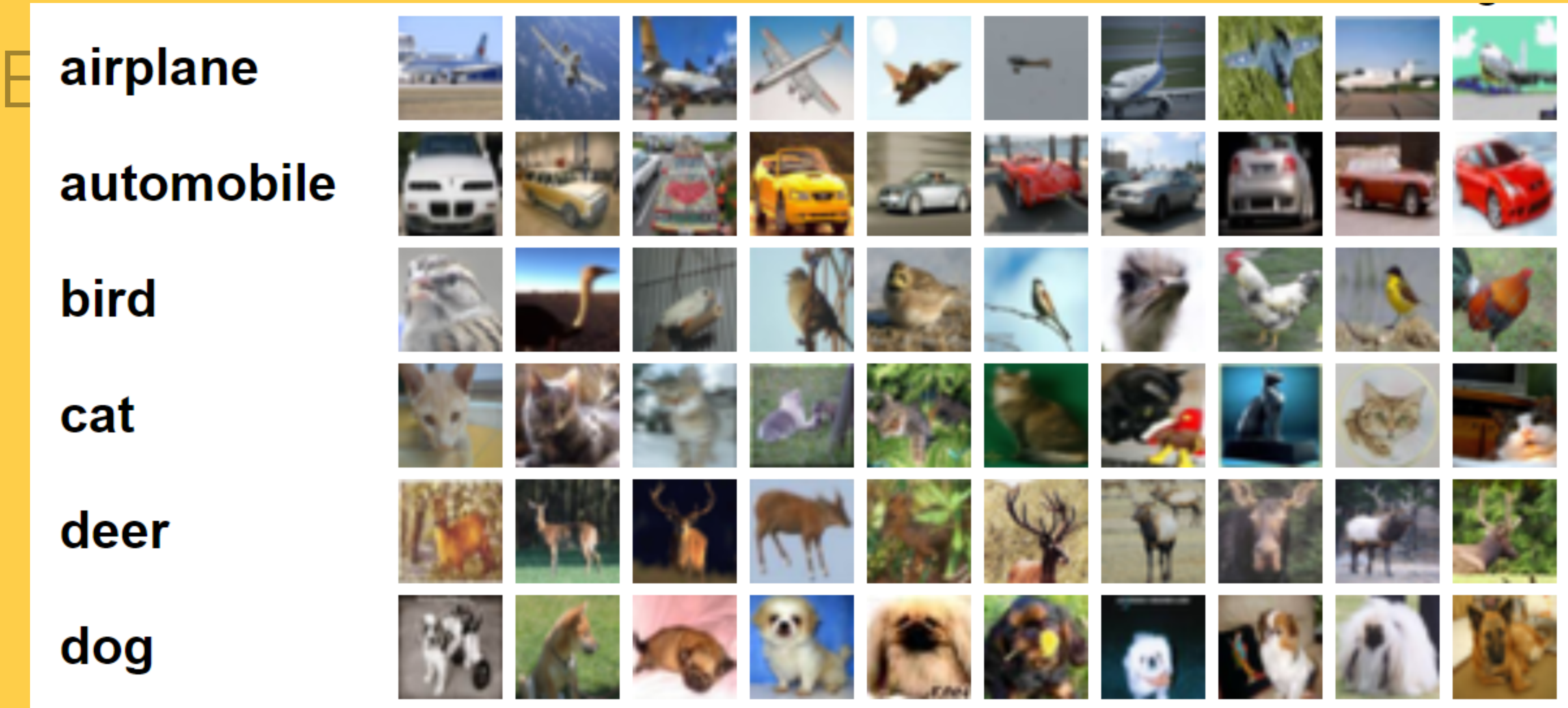
CLASSIFICATION



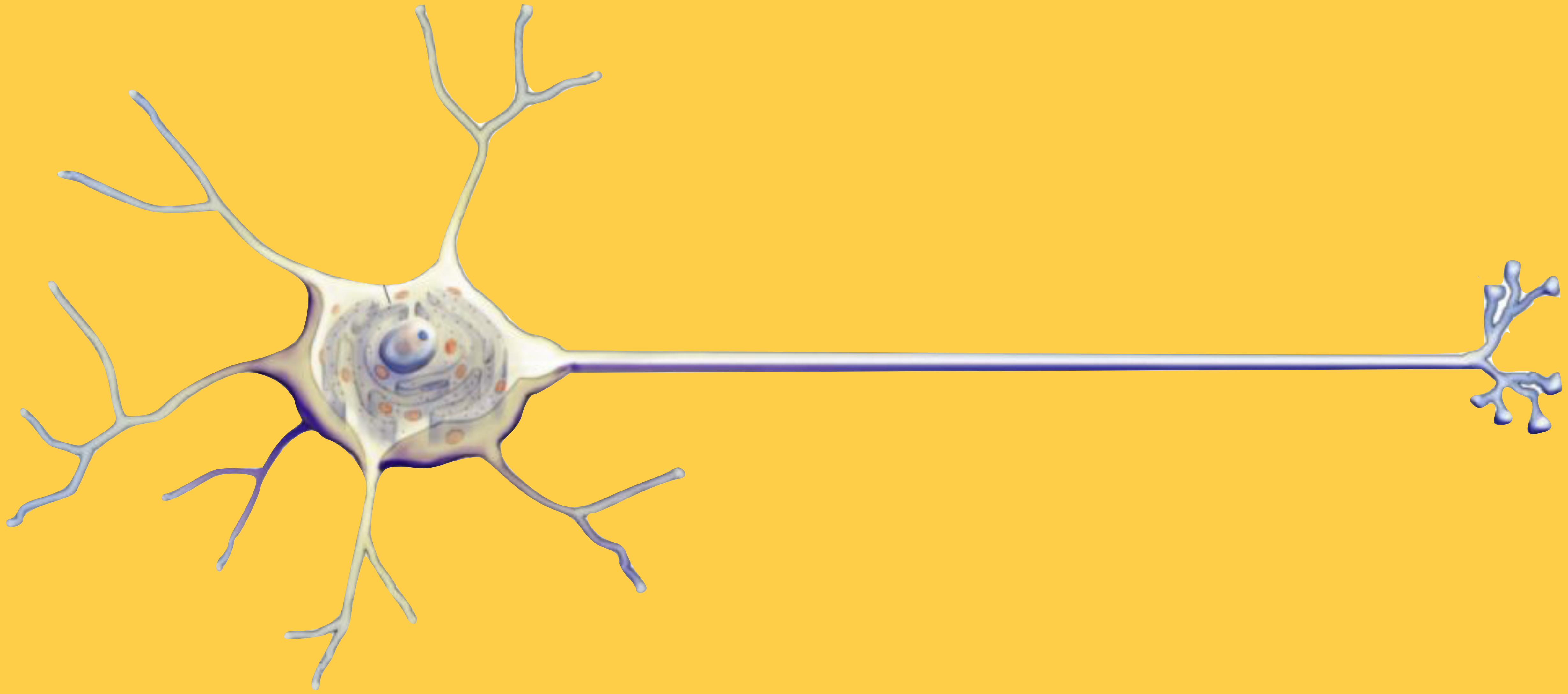
GENERATIVE MODE

REGRESSION

CLASSIFICATION



NEURON



MATHEMATICS

ELSEVIER

Neural Networks
Volume 4, Issue 2, 1991, Pages 251-257

Approximation capabilities of multilayer feedforward networks

Kurt Hornik

Show more

Share Cite

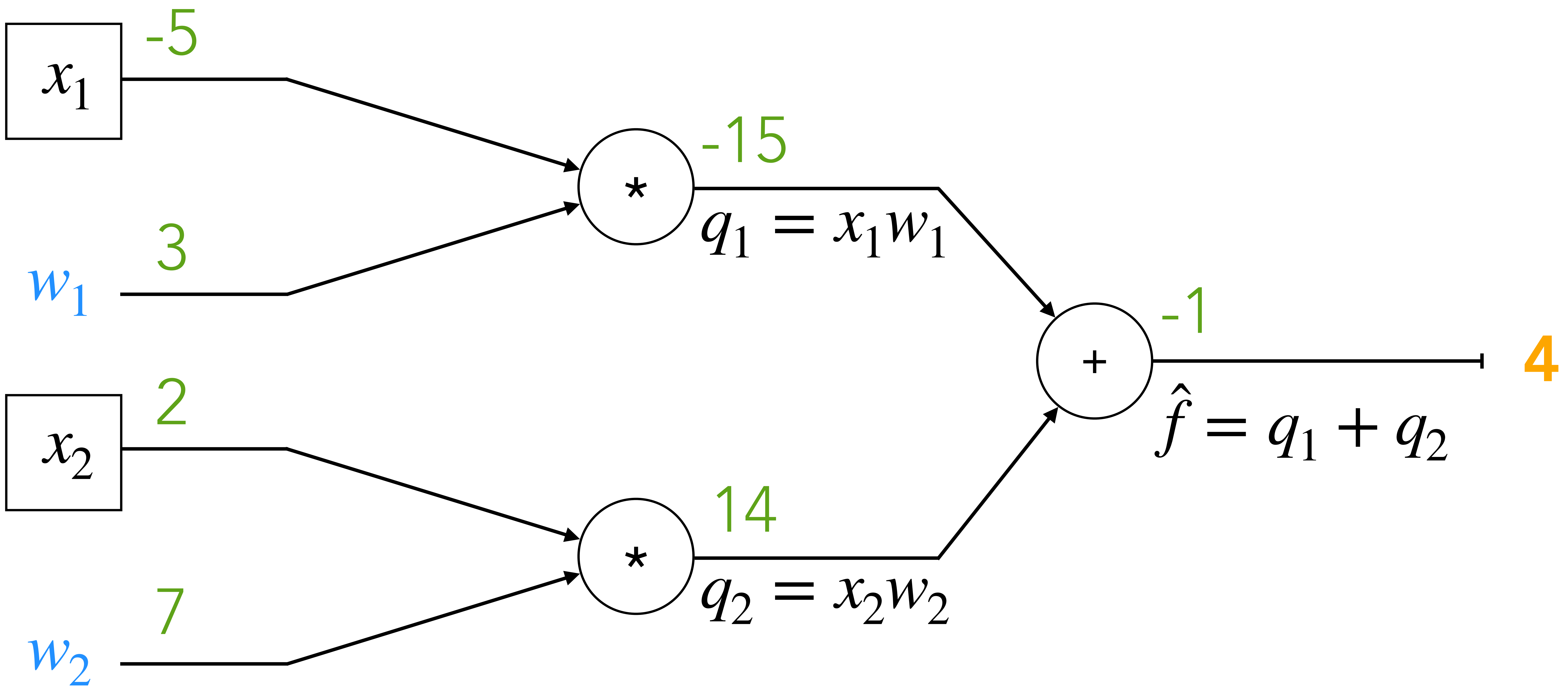
https://doi.org/10.1016/0893-6080(91)90009-T

Abstract

We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.

MATHEMATICS

COMPUTATIONAL GRAPH

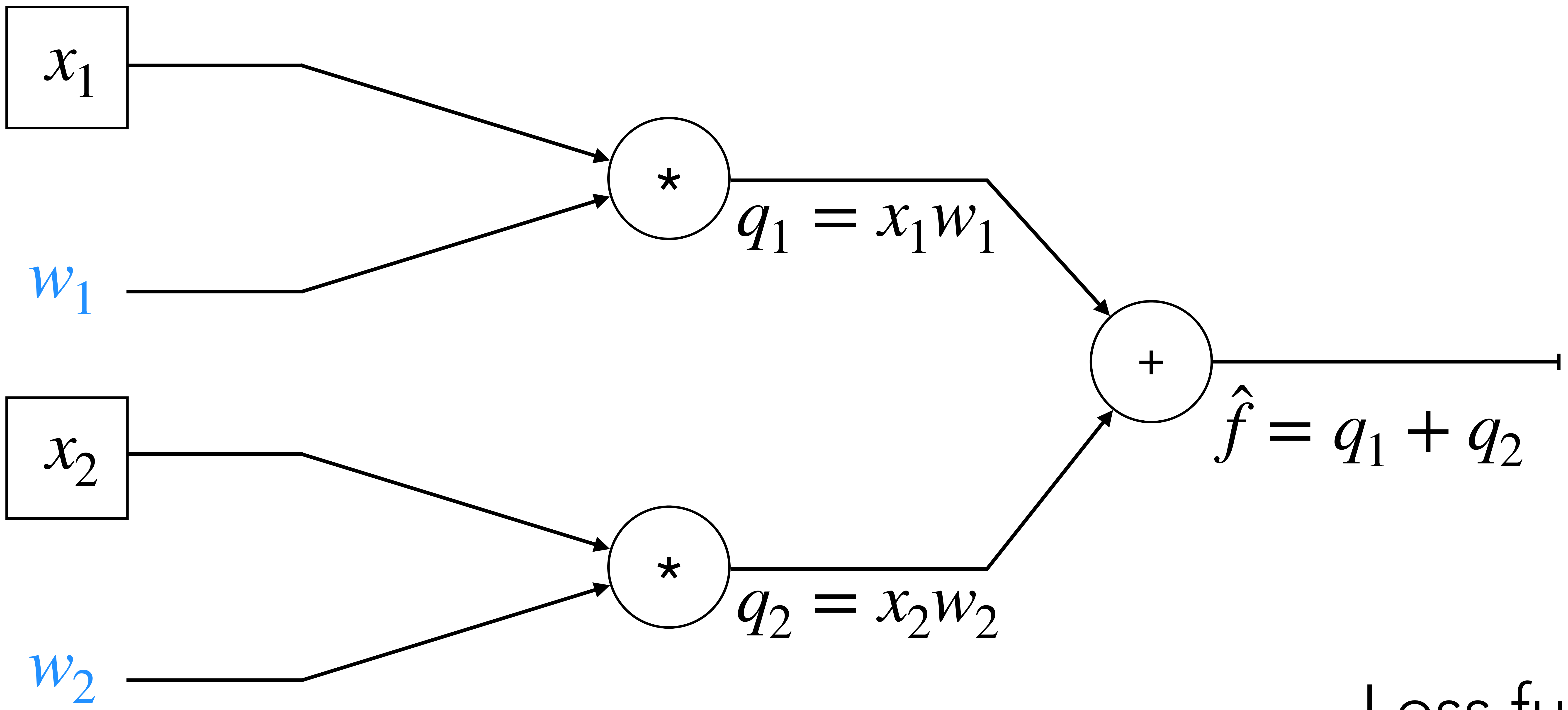


$$\hat{f} = x_1 w_1 + x_2 w_2$$

MACHINE LEARNING

SUPERVISED LEARNING

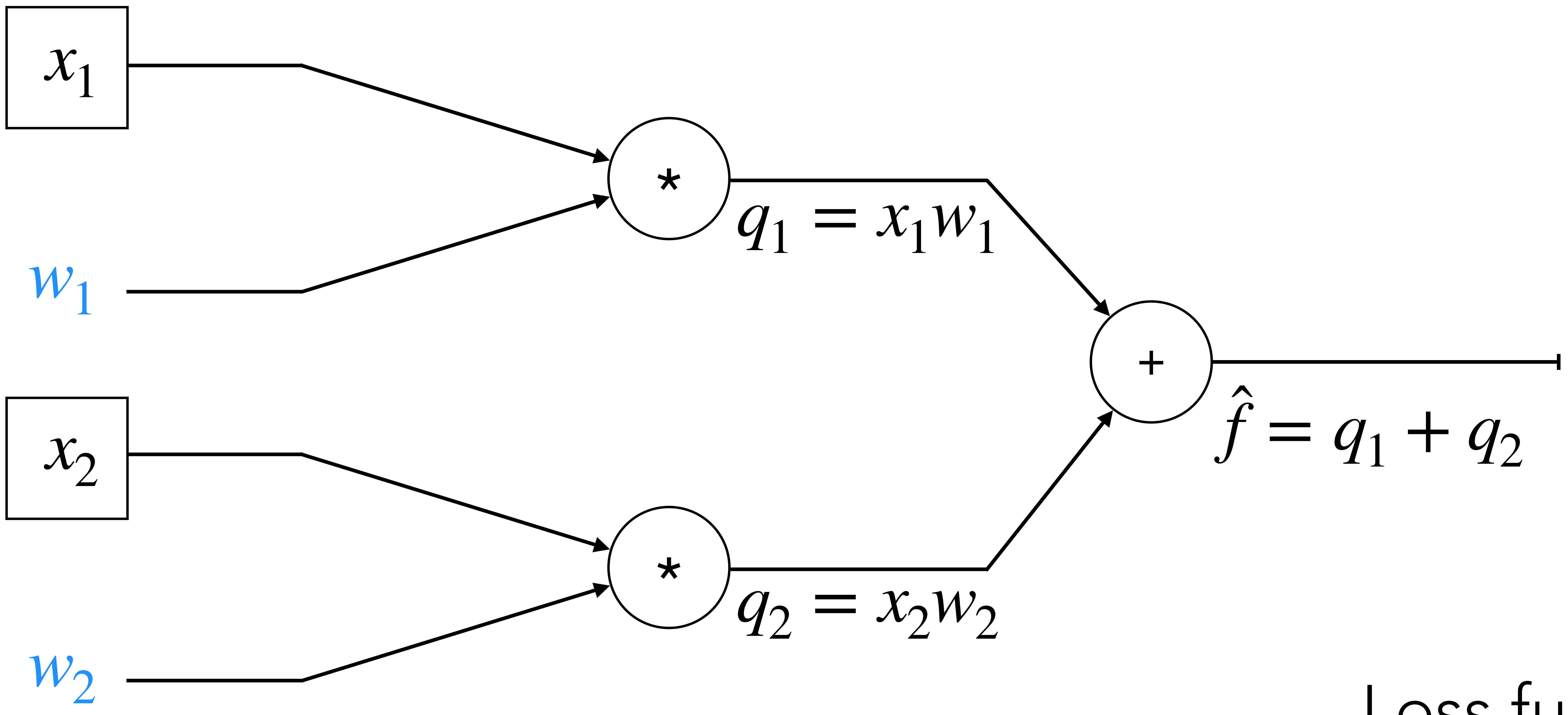
REGRESSION



Loss function

$$\hat{f} = x_1 w_1 + x_2 w_2 \qquad J(w) = \hat{f} - f$$

SUPERVISED LEARNING



Loss function

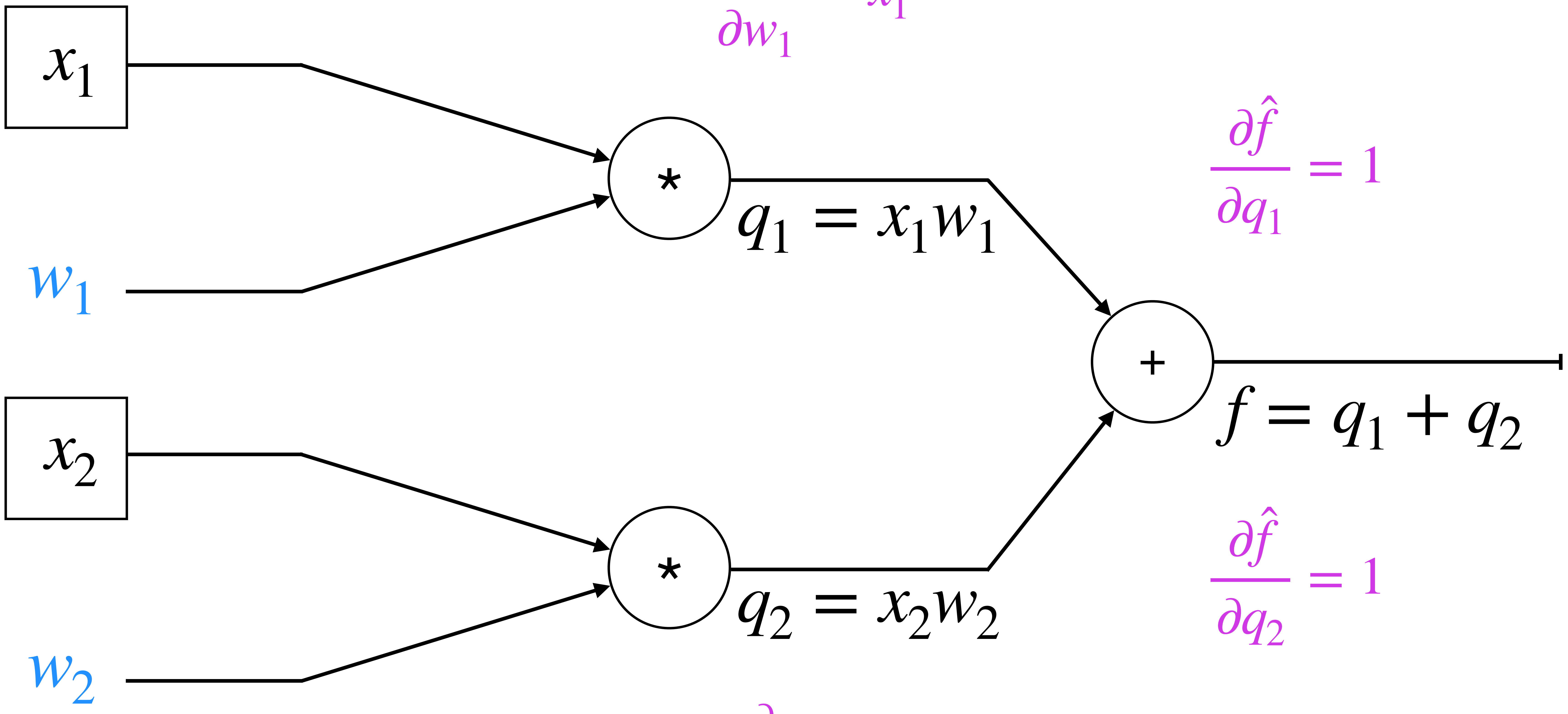
$$\hat{f} = x_1 w_1 + x_2 w_2$$

$$J(w) = \hat{f} - f$$

BACKPROPAGATION

$$w_1 = w_1 - \eta * \frac{\partial J}{\partial \hat{f}} \frac{\partial \hat{f}}{\partial q_1} \frac{\partial q_1}{\partial w_1}$$

$$\frac{\partial q_1}{\partial w_1} = x_1$$



$$\frac{\partial \hat{f}}{\partial q_1} = 1$$

$$\frac{\partial J}{\partial \hat{f}} = 1$$

$$\frac{\partial \hat{f}}{\partial q_2} = 1$$

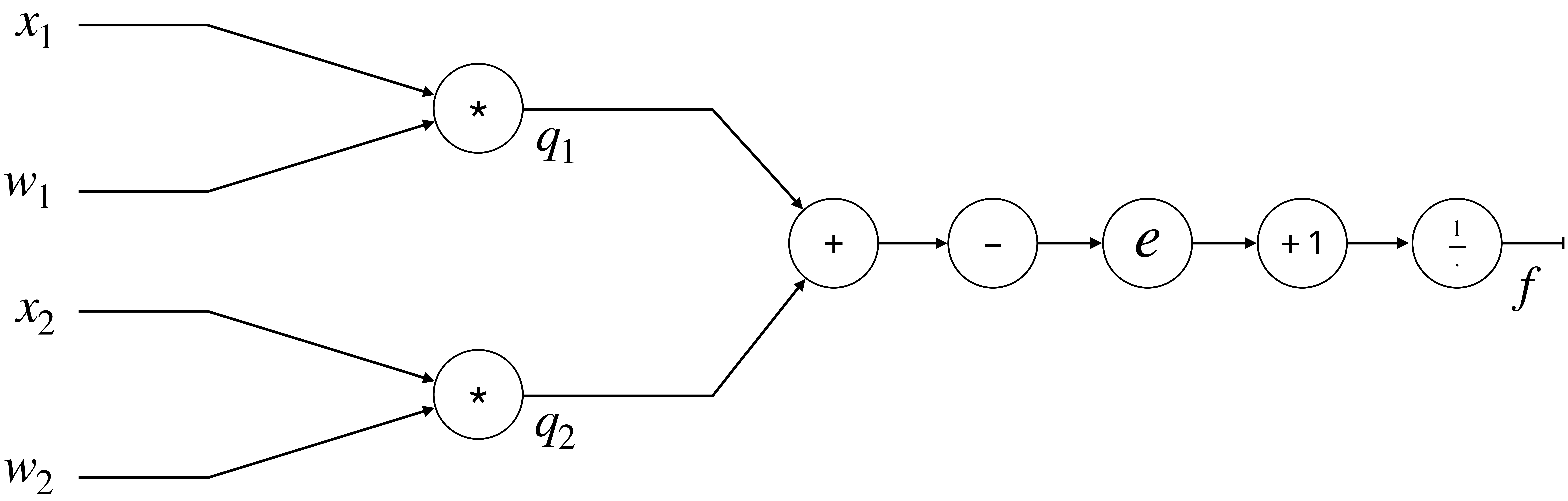
$$\frac{\partial q_2}{\partial w_2} = x_2$$

$$w_2 = w_2 - \eta * \frac{\partial J}{\partial \hat{f}} \frac{\partial \hat{f}}{\partial q_2} \frac{\partial q_2}{\partial w_2}$$

Loss function

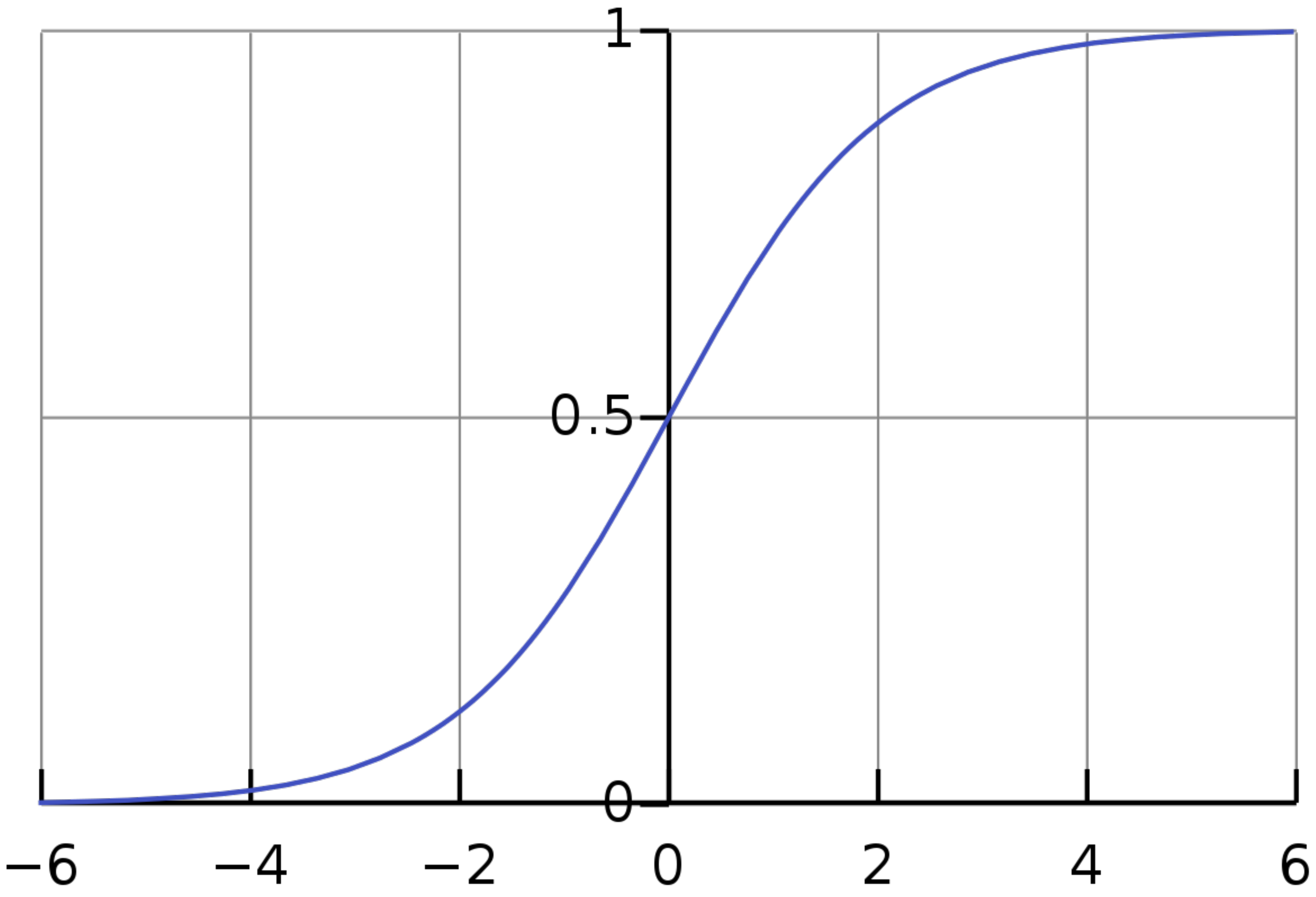
$$J(w) = \hat{f} - f$$

LOGISTIC REGRESSION

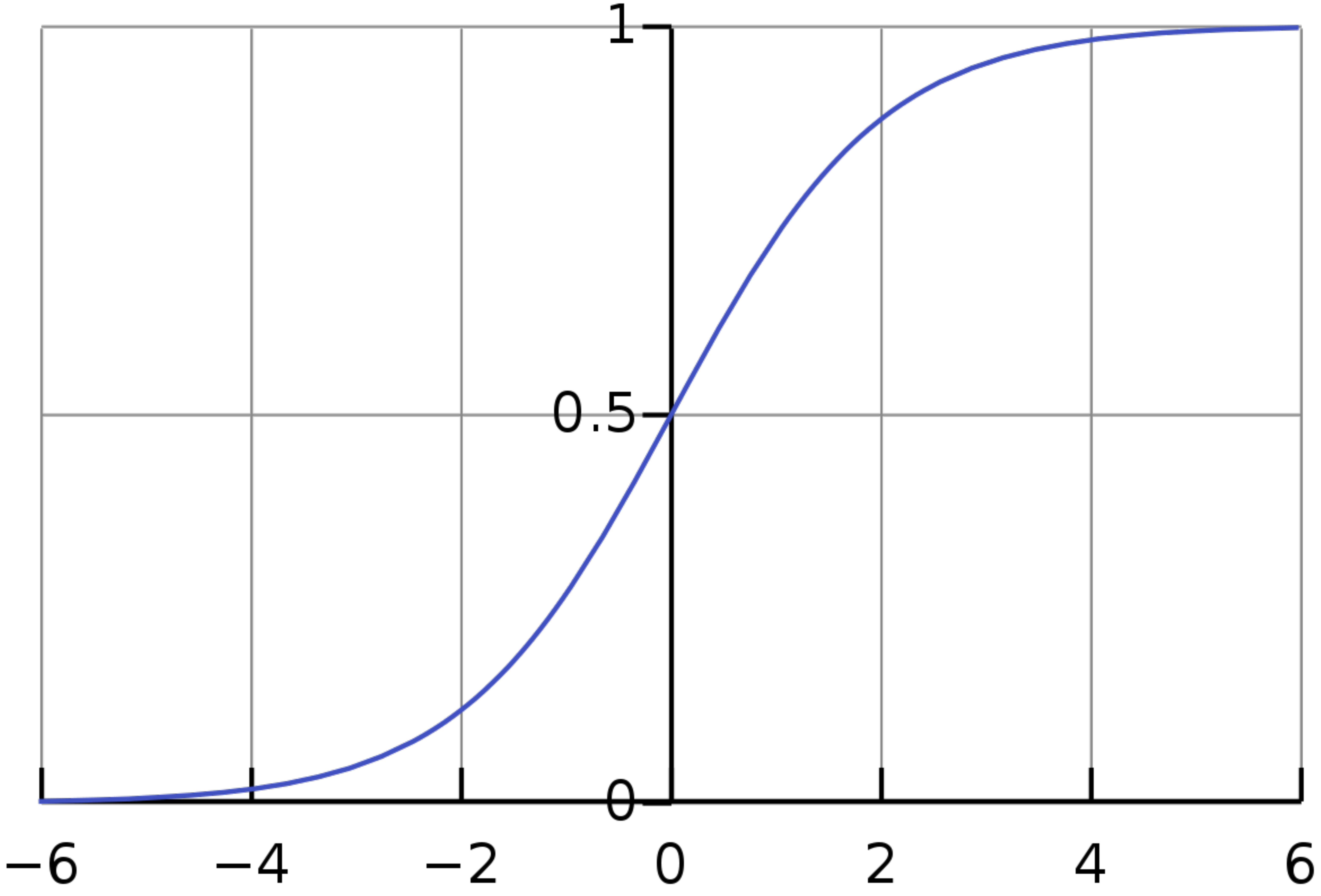


$$f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}}$$

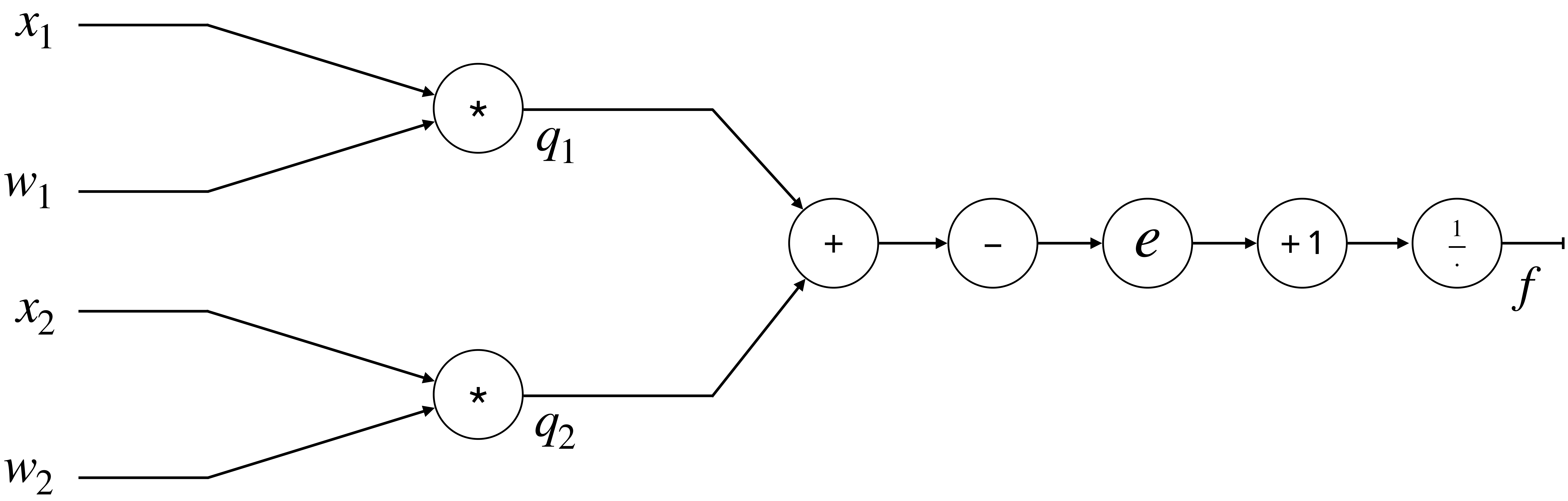
LOGISTIC REGRESSION



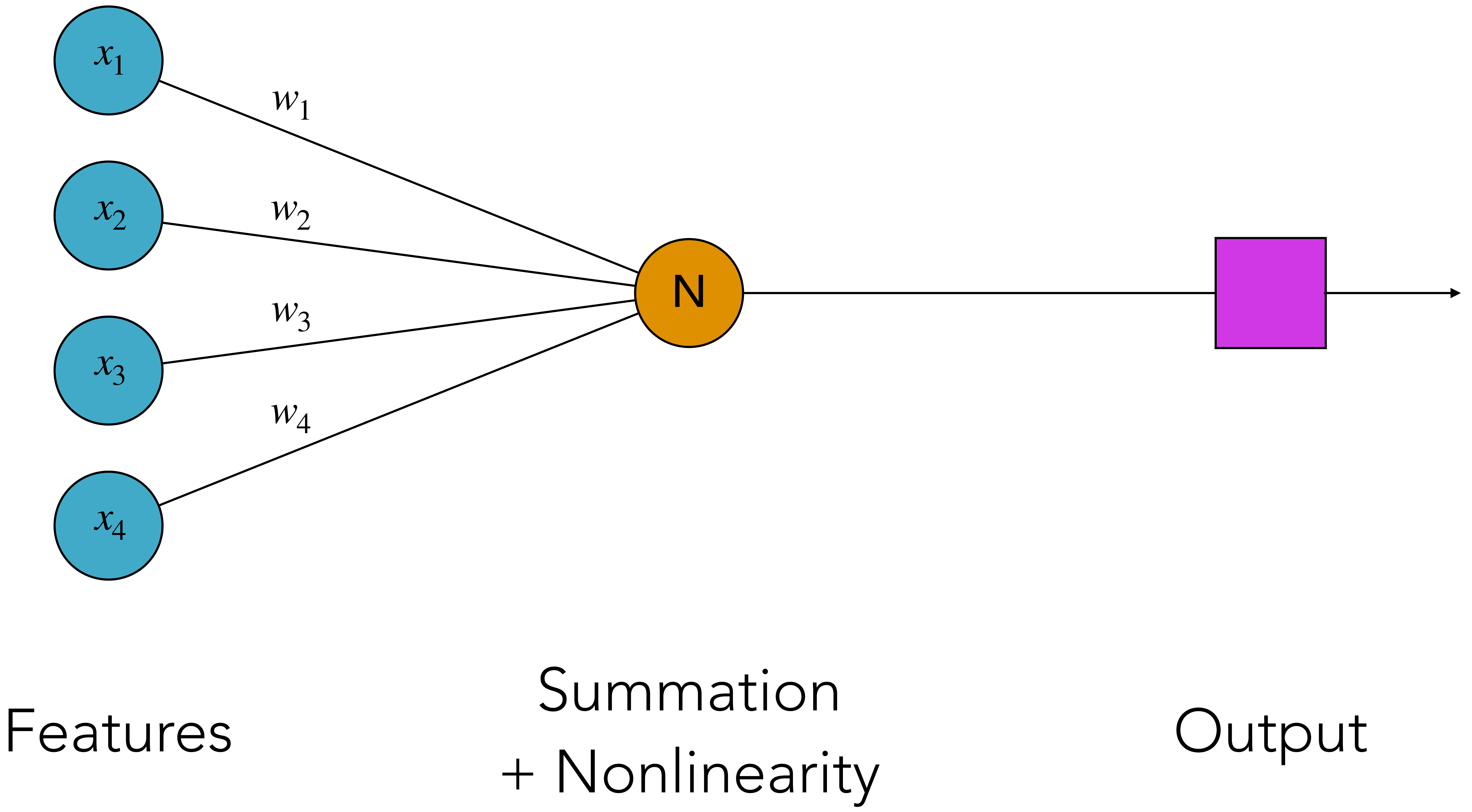
BINARY CLASSIFICATION

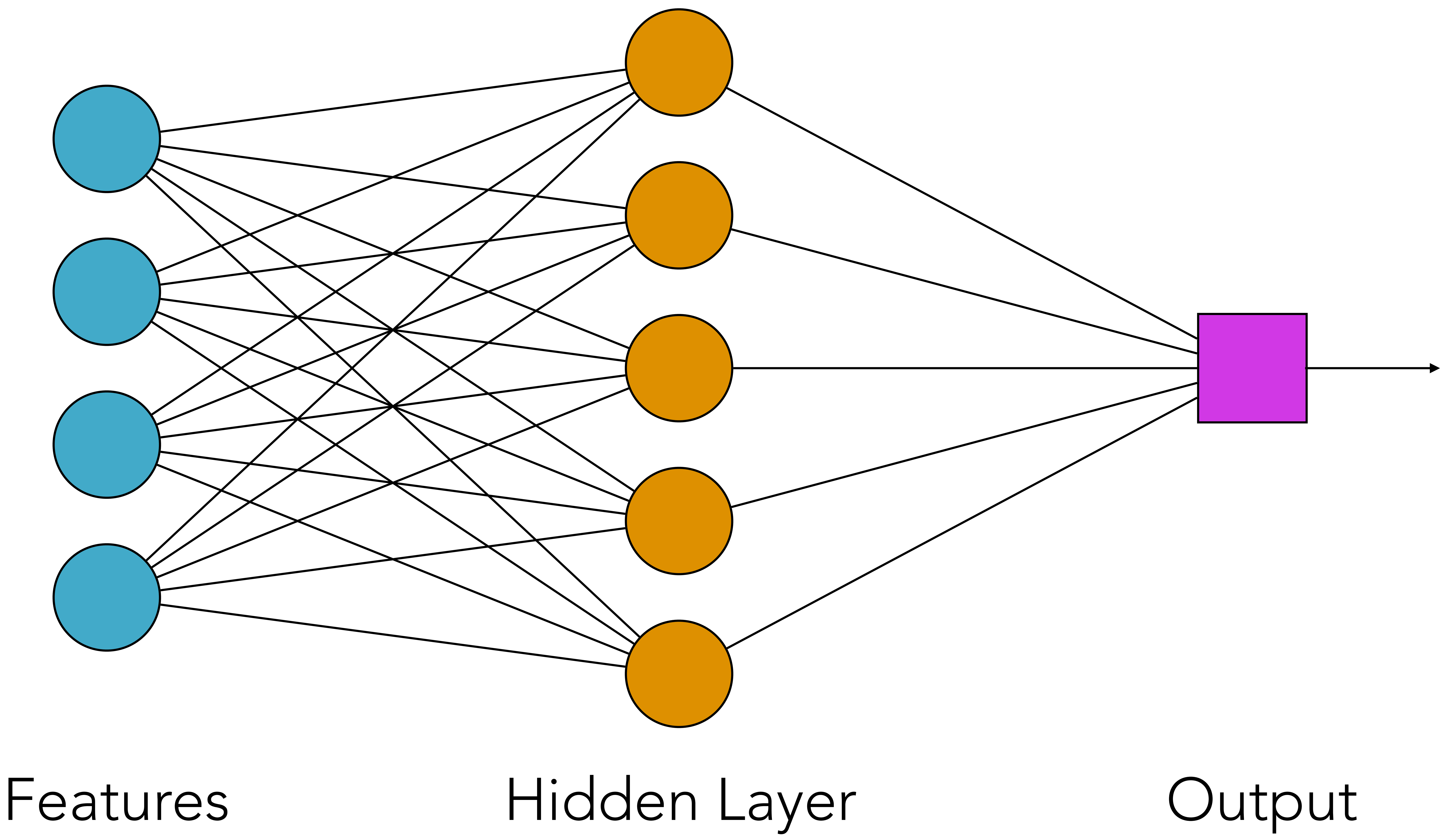


LOGISTIC REGRESSION



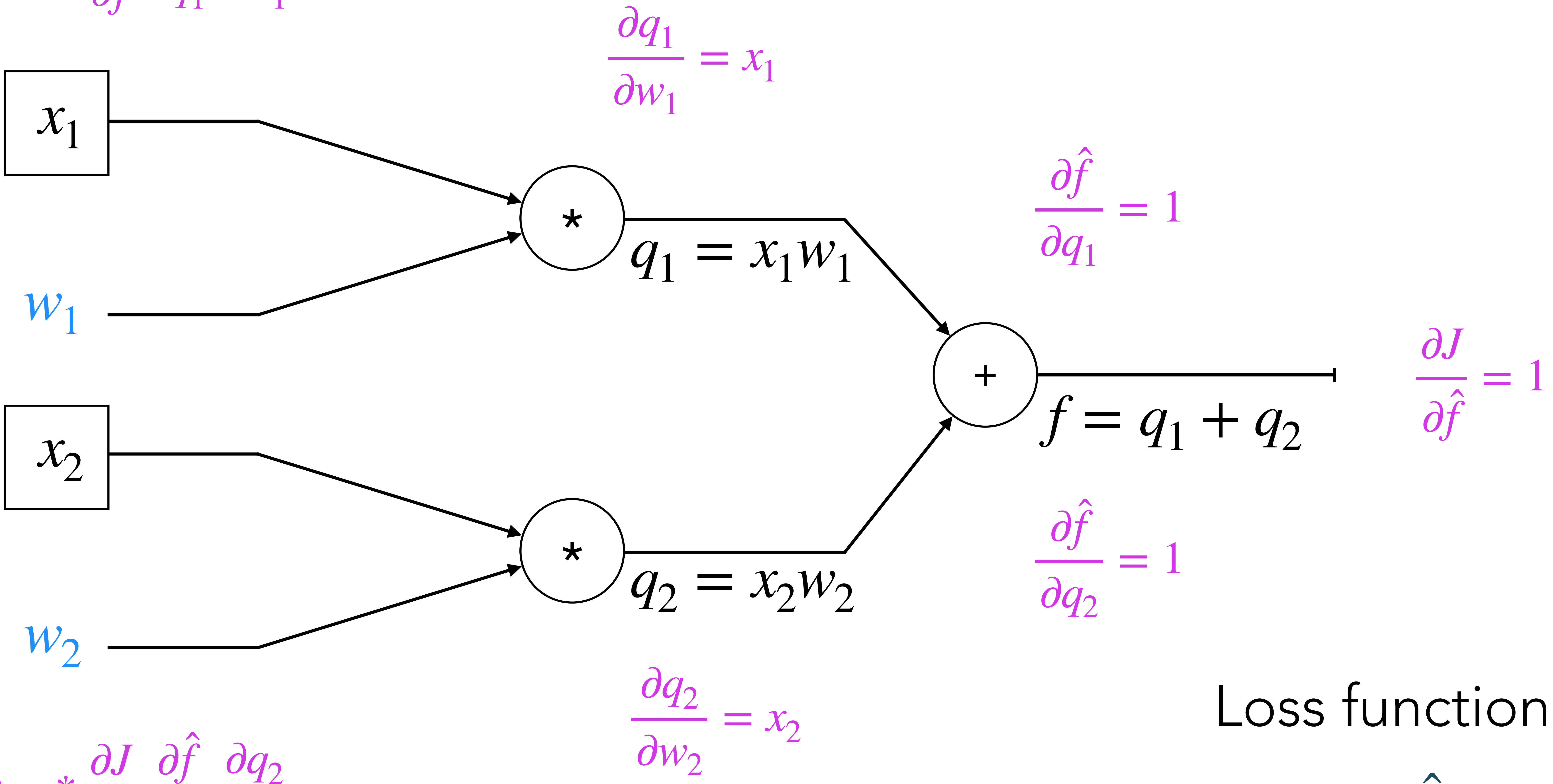
$$f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}}$$


x_1
x_2
x_3
x_4
w_1
w_2
w_3
w_4
N
Features
Summation
+ Nonlinearity
Output


Features
Hidden Layer
Output

BACKPROPAGATION

$$w_1 = w_1 + \eta * \frac{\partial J}{\partial \hat{f}} \frac{\partial \hat{f}}{\partial q_1} \frac{\partial q_1}{\partial w_1}$$



$$w_2 = w_2 + \eta * \frac{\partial J}{\partial \hat{f}} \frac{\partial \hat{f}}{\partial q_2} \frac{\partial q_2}{\partial w_2}$$

Loss function

$$J(w) = \hat{f} - f$$

AUTOMATIC DIFFERENTIATION

TensorFlow

Keras

PyTorch

SESSION 2 TOPICS

- Convolutional Layers
- Classification
- Pre-trained models
- Unsupervised Methods
- Best practices
- Hot topics in ML research

MICHELLE KUCHERA
DAVIDSON COLLEGE

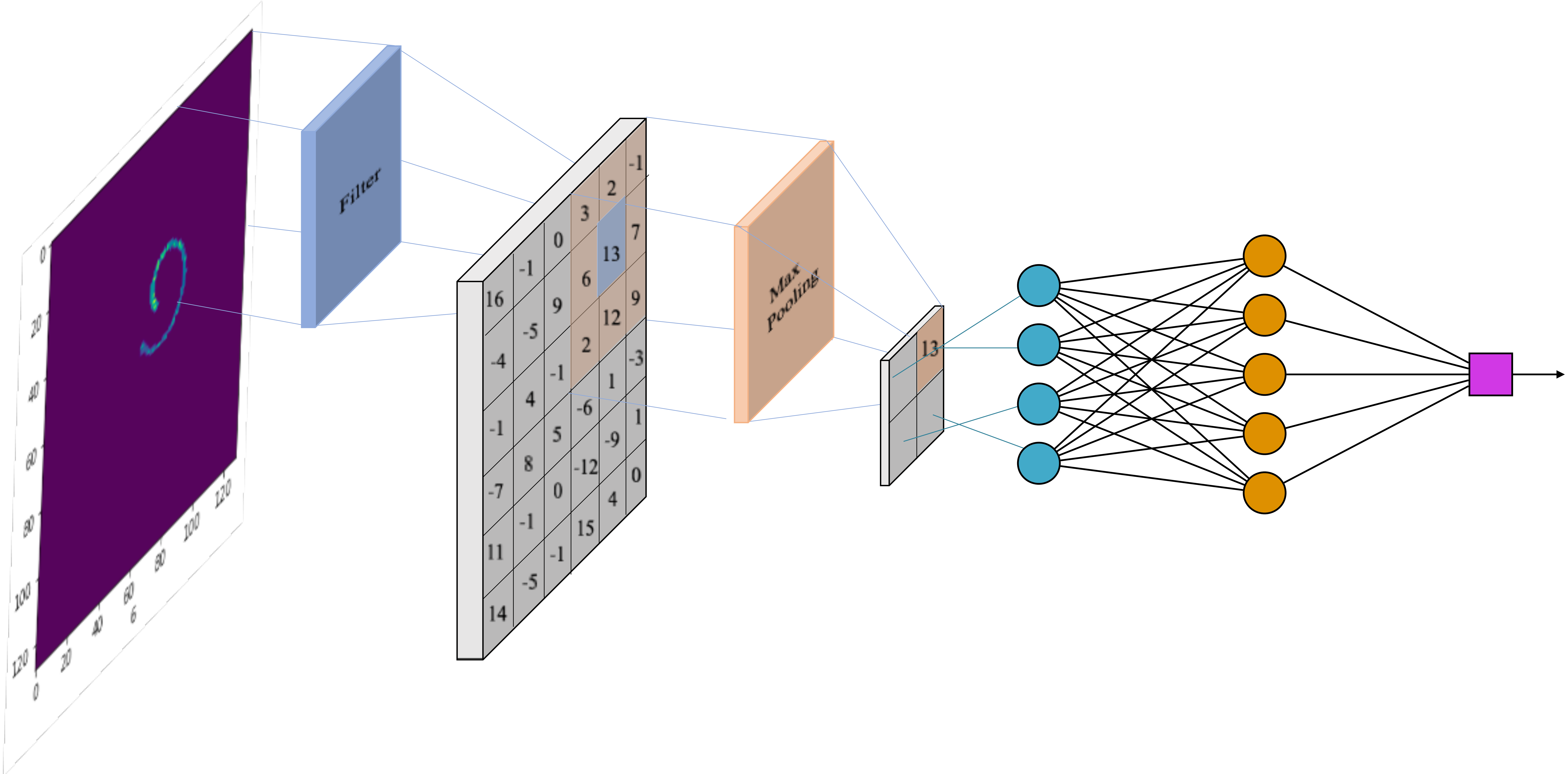
JOINT ICTP-IAEA WORKSHOP ADVANCED SCHOOL ON COMPUTATIONAL NUCLEAR SCIENCE

23 MAY 2022

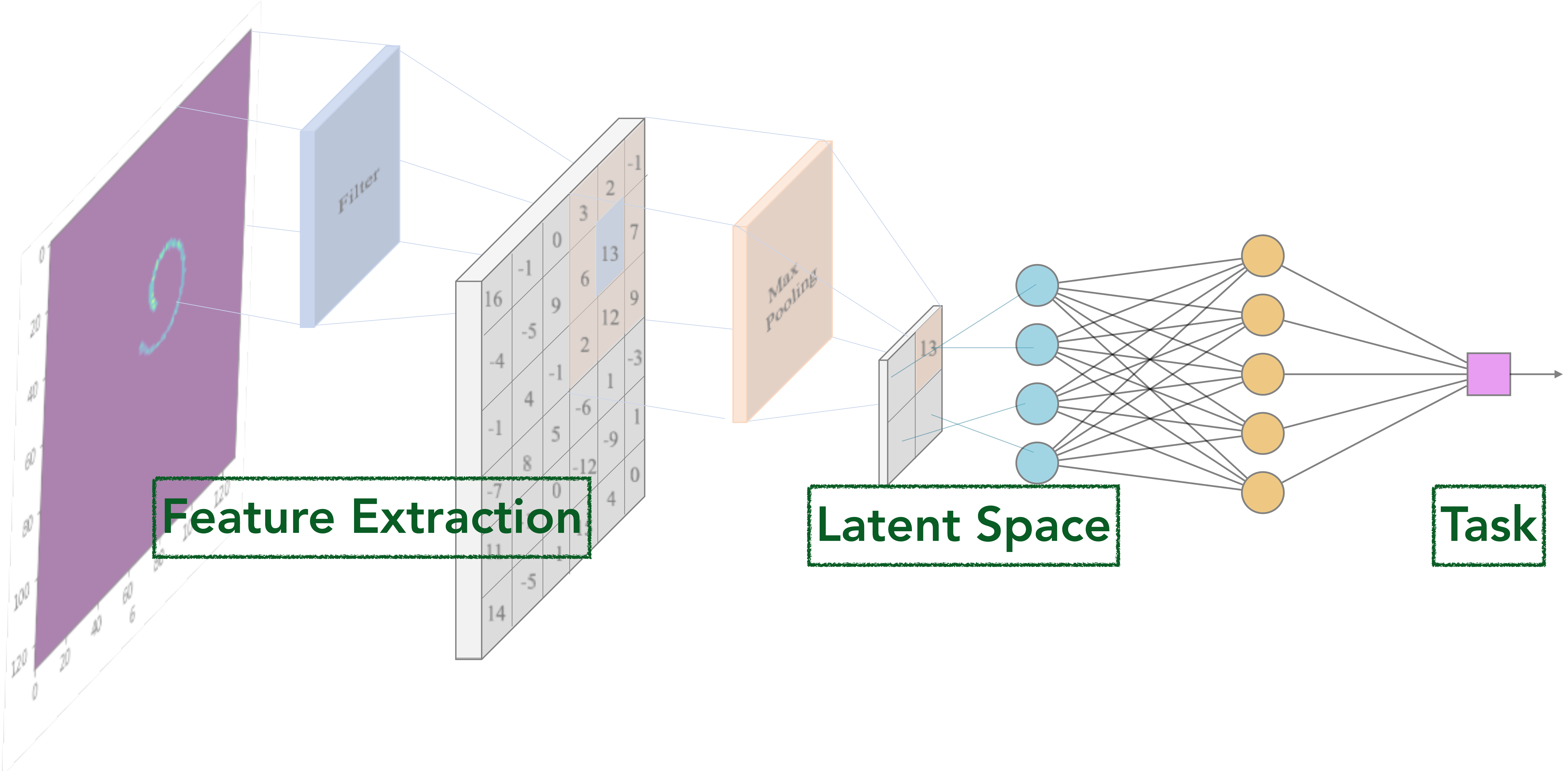
CONVOLUTIONAL NEURAL NETWORKS

CLASSIFICATION

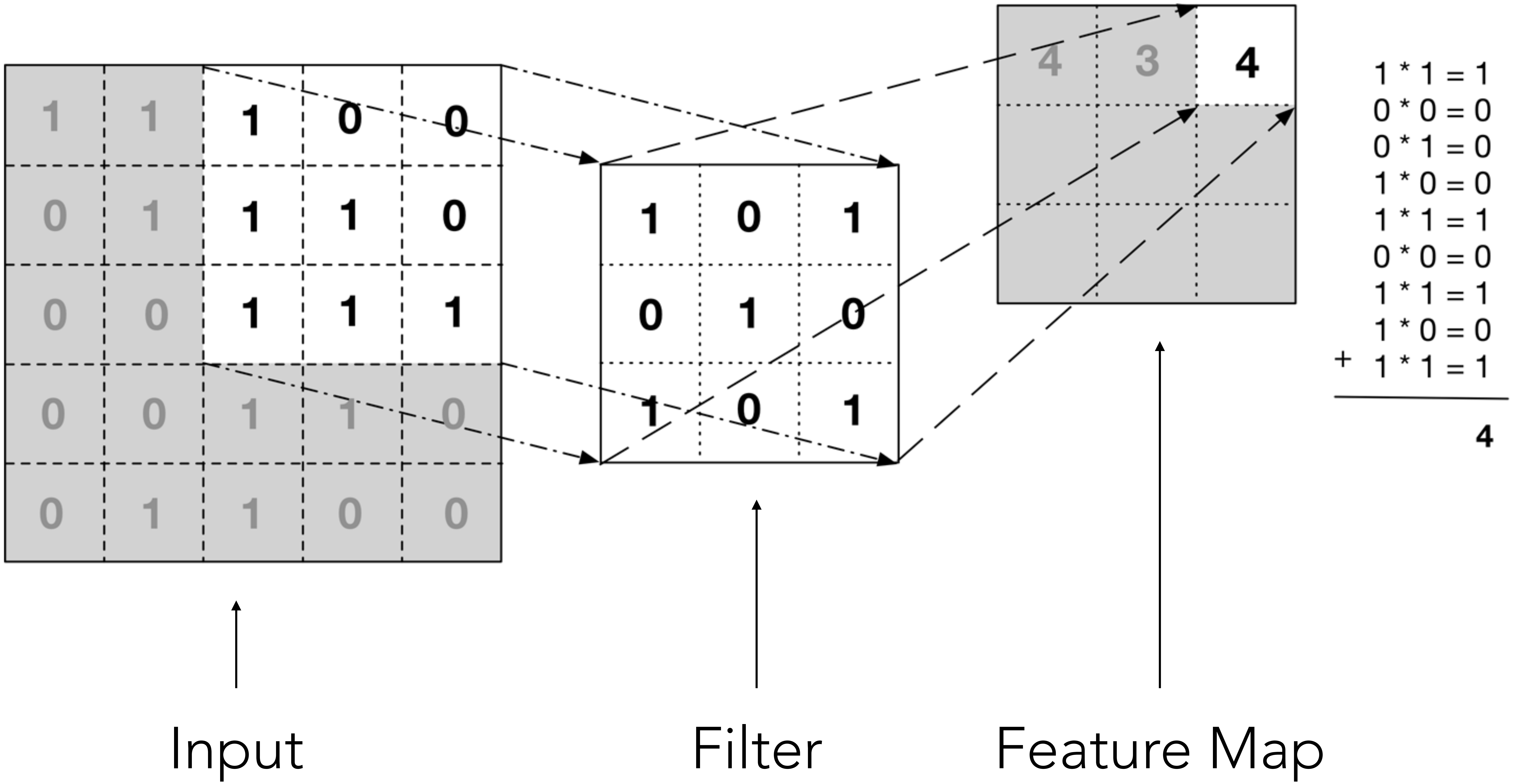
CONVOLUTIONAL NEURAL NETWORKS



CONVOLUTIONAL NEURAL NETWORKS

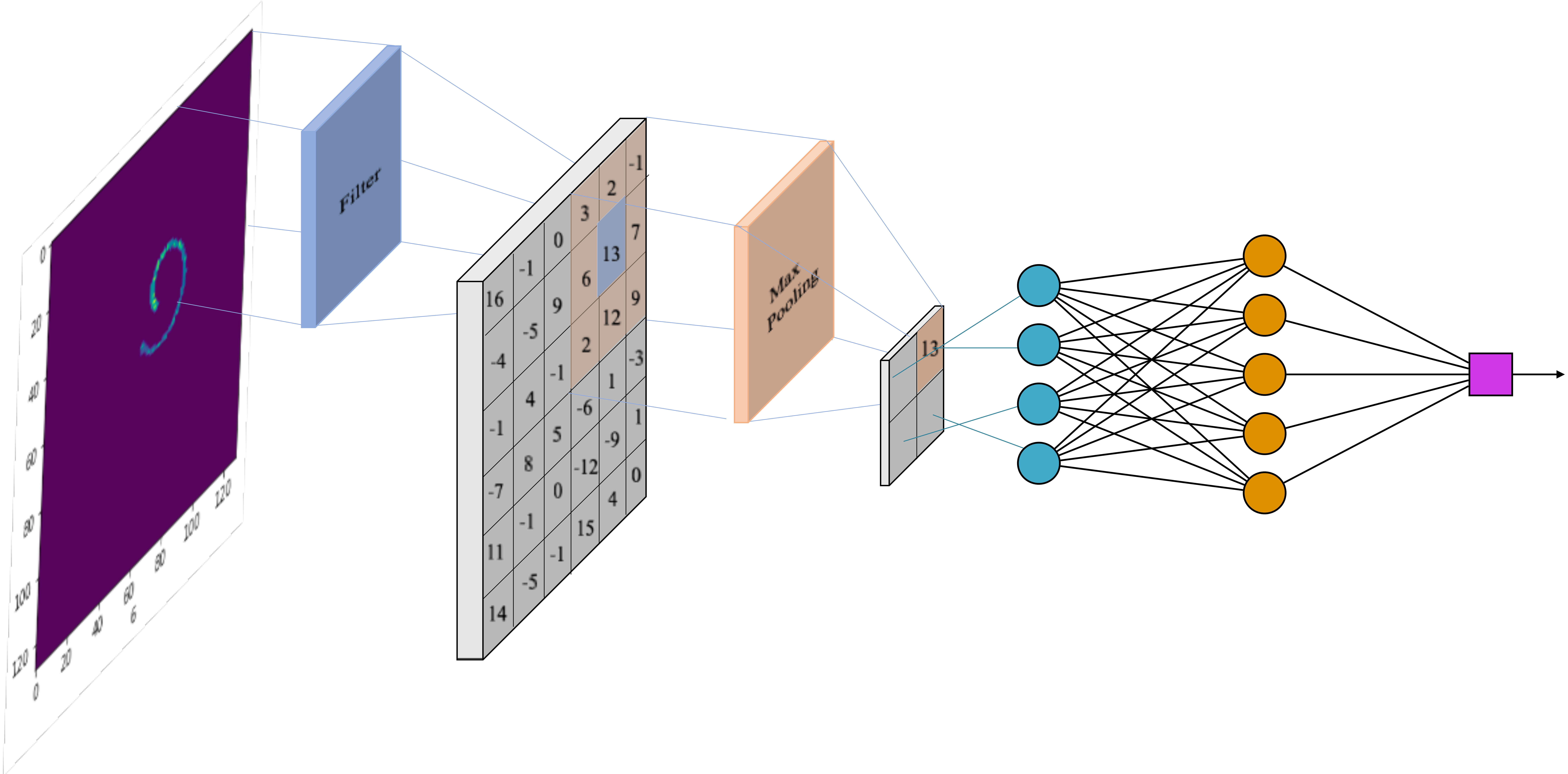


DISCRETE CONVOLUTION



ADAPTED FROM *DEEP LEARNING*, ADAM GIBSON & JOSH PATTERSON

CONVOLUTIONAL NEURAL NETWORKS

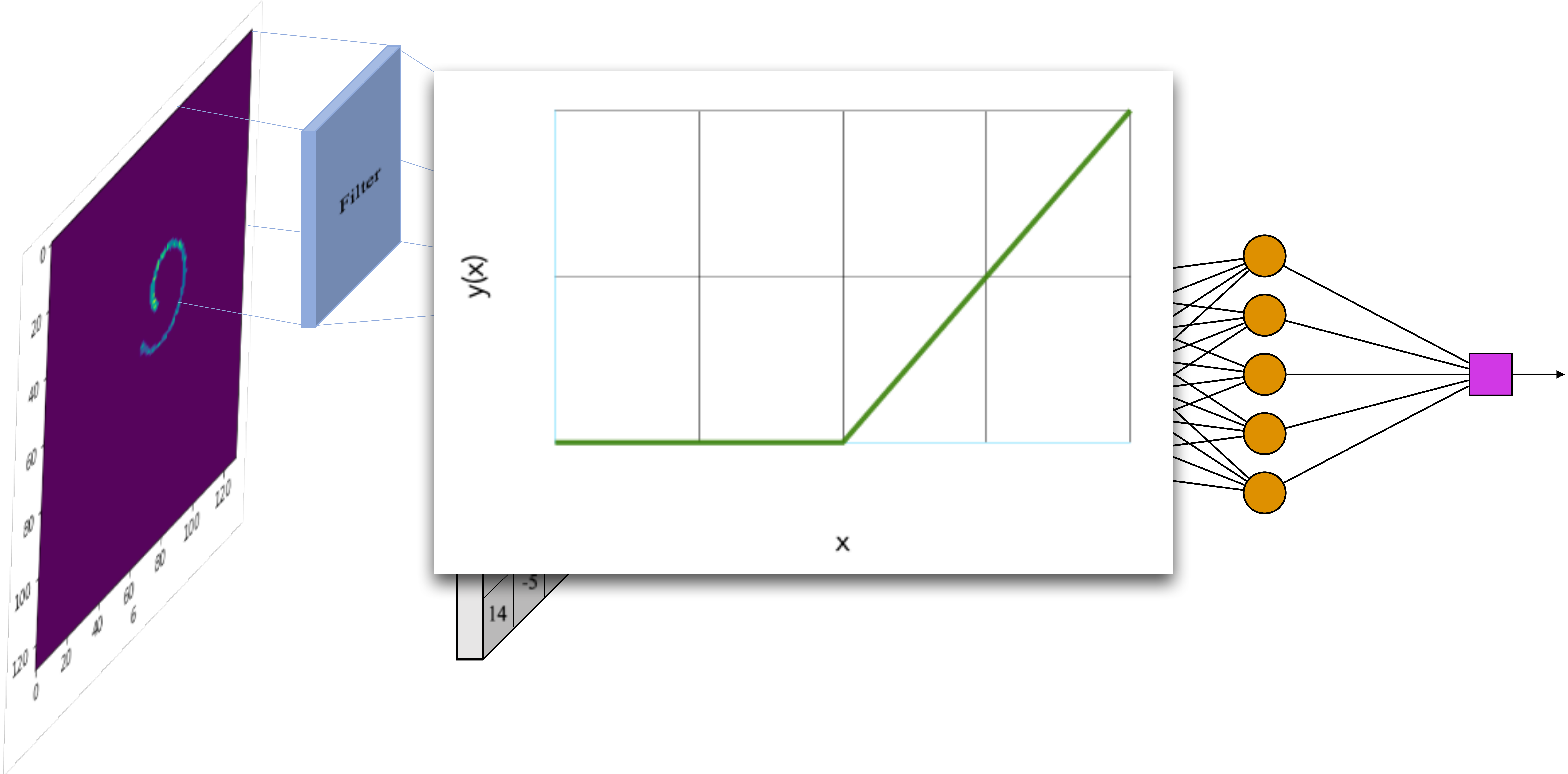



1156

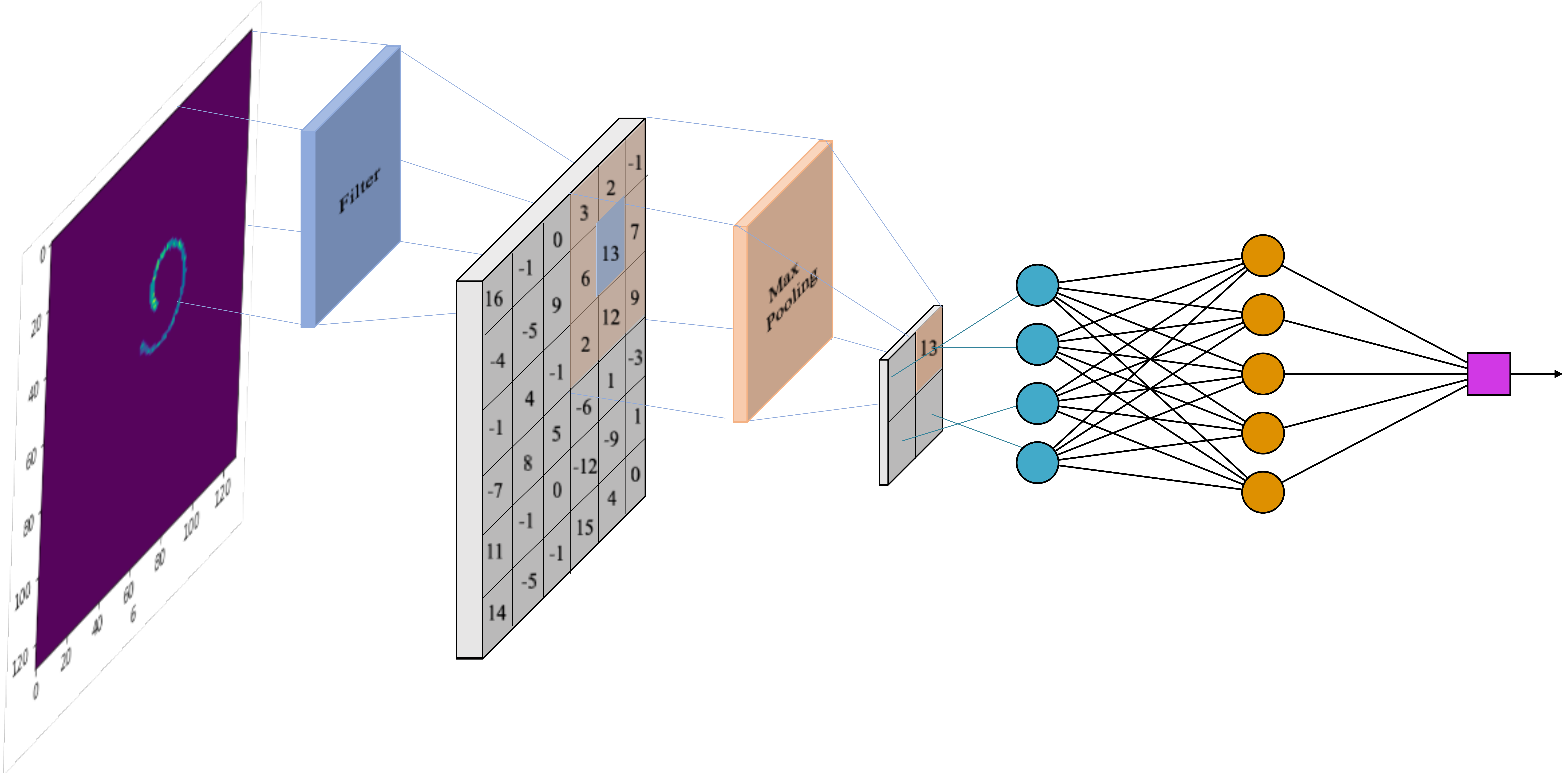
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
5	5	5	5	5
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

-1	-1	5	-1	-1
-1	-1	5	-1	-1
-1	-1	5	-1	-1
-1	-1	5	-1	-1
-1	-1	5	-1	-1

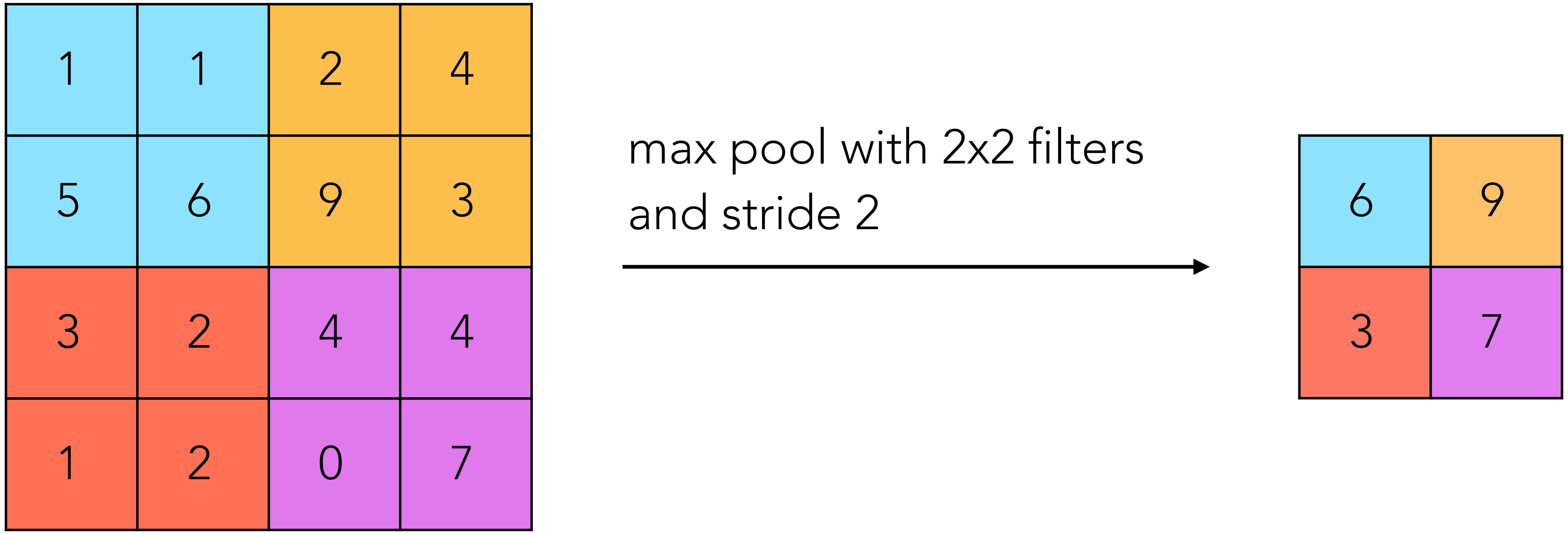
CONVOLUTIONAL NEURAL NETWORKS



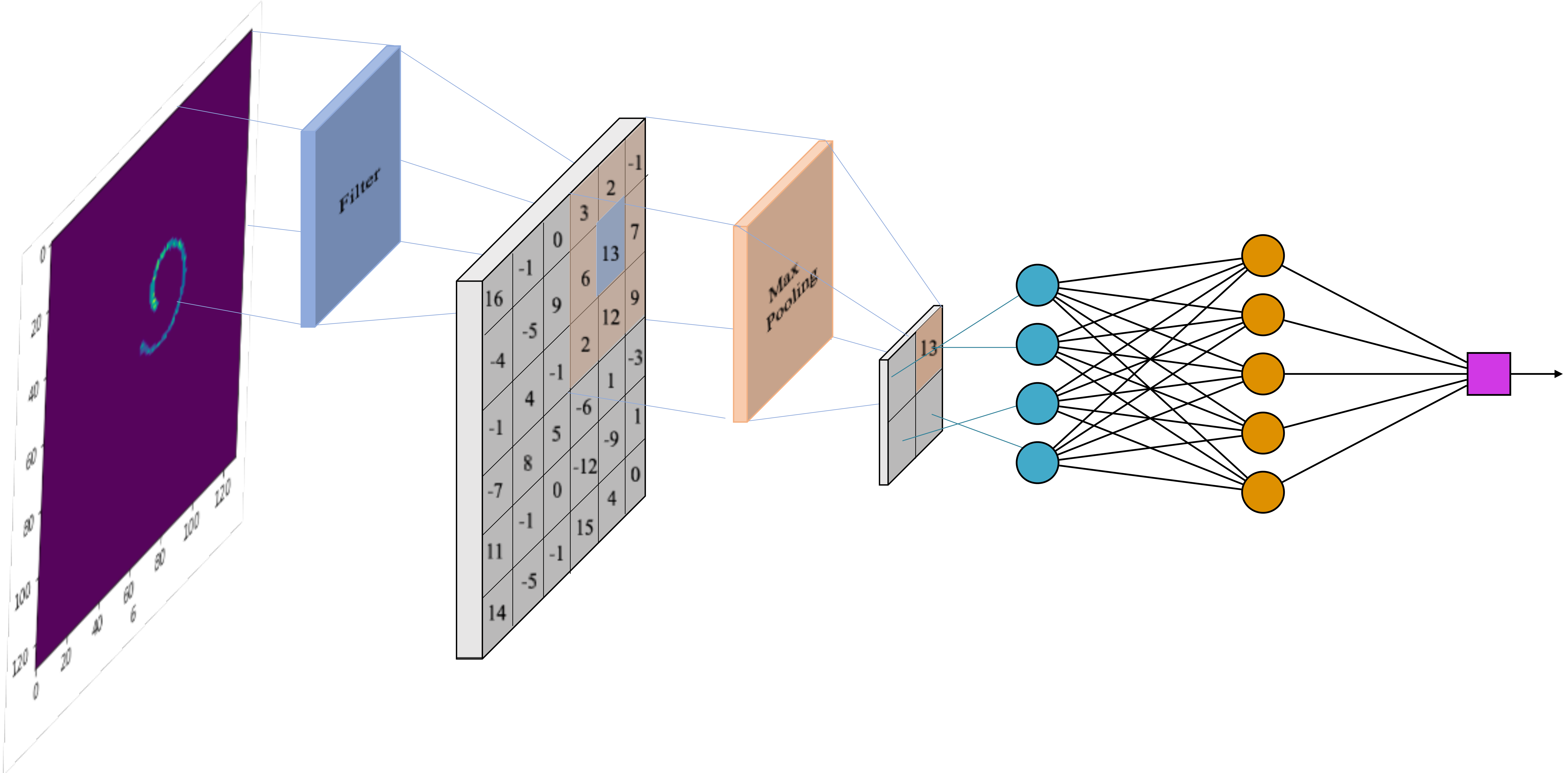
CONVOLUTIONAL NEURAL NETWORKS



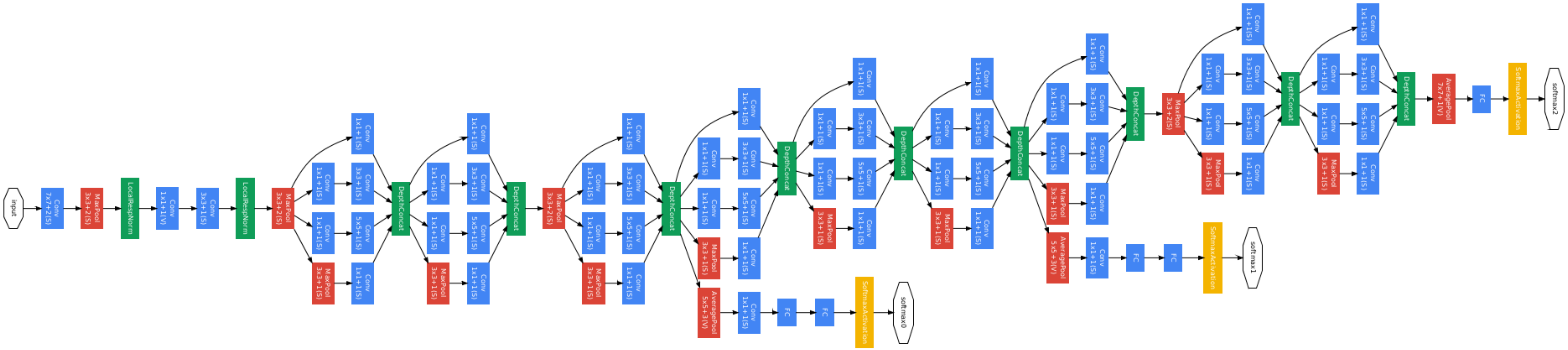
MAX POOLING


1 1 2 4
5 6 9 3
3 2 4 4
1 2 0 7
max pool with 2x2 filters
and stride 2
6 9
3 7

CONVOLUTIONAL NEURAL NETWORKS



"GoogLeNet network with all the bells and whistles"



CHRISTIAN SZEGEDY ET. AL. GOING DEEPER WITH CONVOLUTIONS.

CHOOSING AN ARCHITECTURE

HOW MANY LAYERS?

HOW MANY NODES PER LAYER?

LEARNING RATE

DROPOUT?

WHAT ACTIVATION FUNCTION(S)?

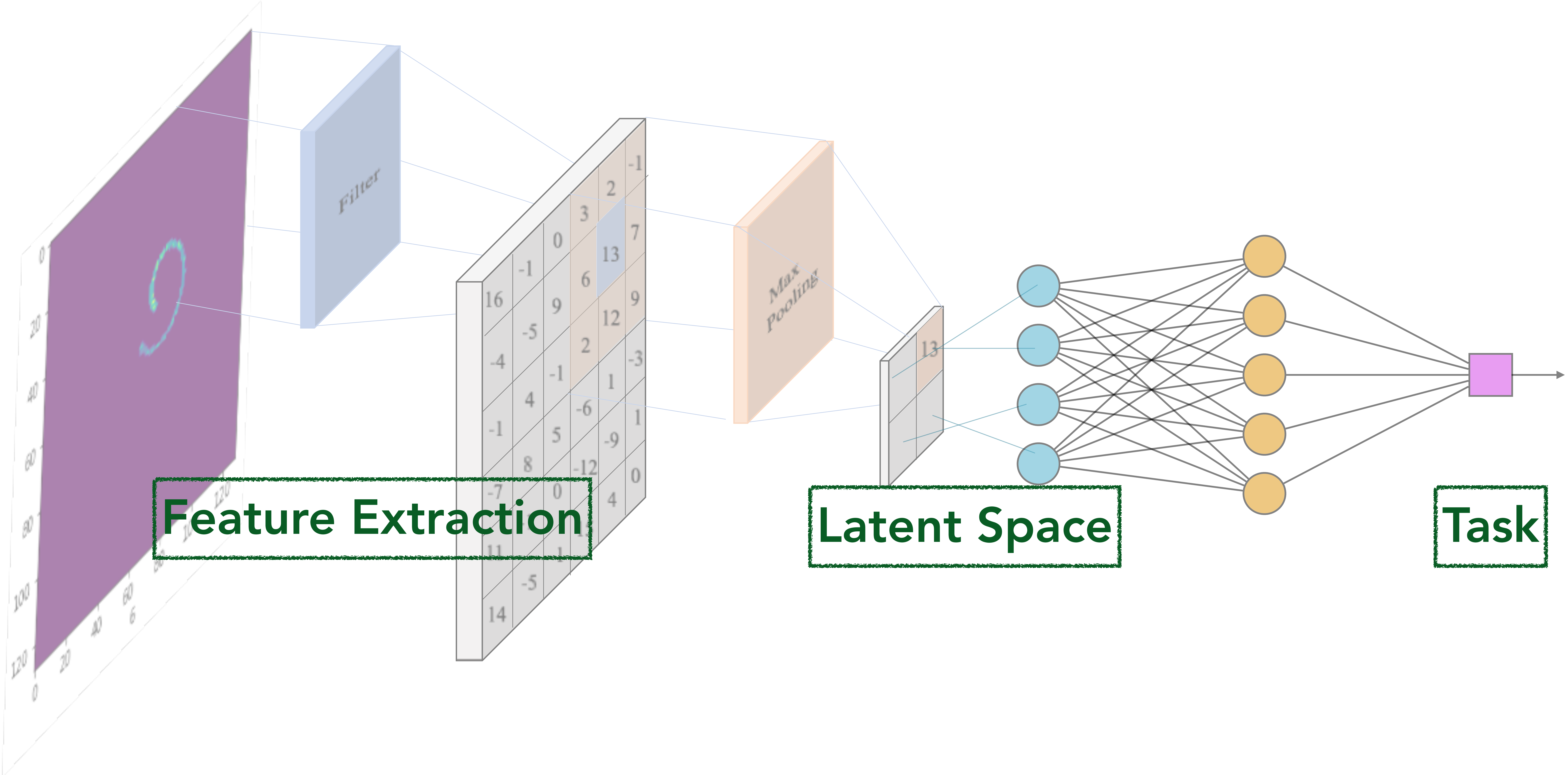
HOW MANY CONVOLUTION LAYERS?

FILTER SIZE?

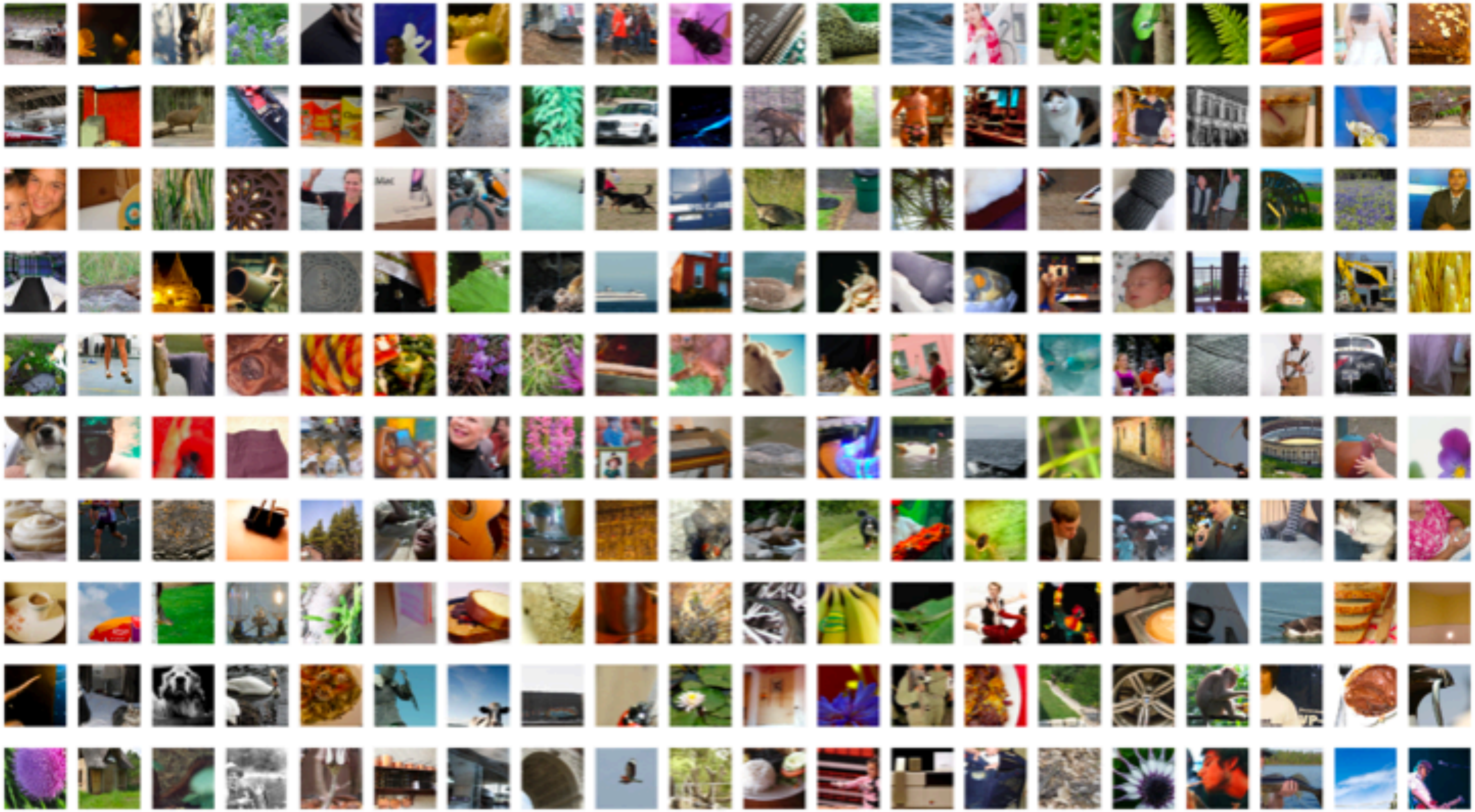
STRIDE?

POOLING?

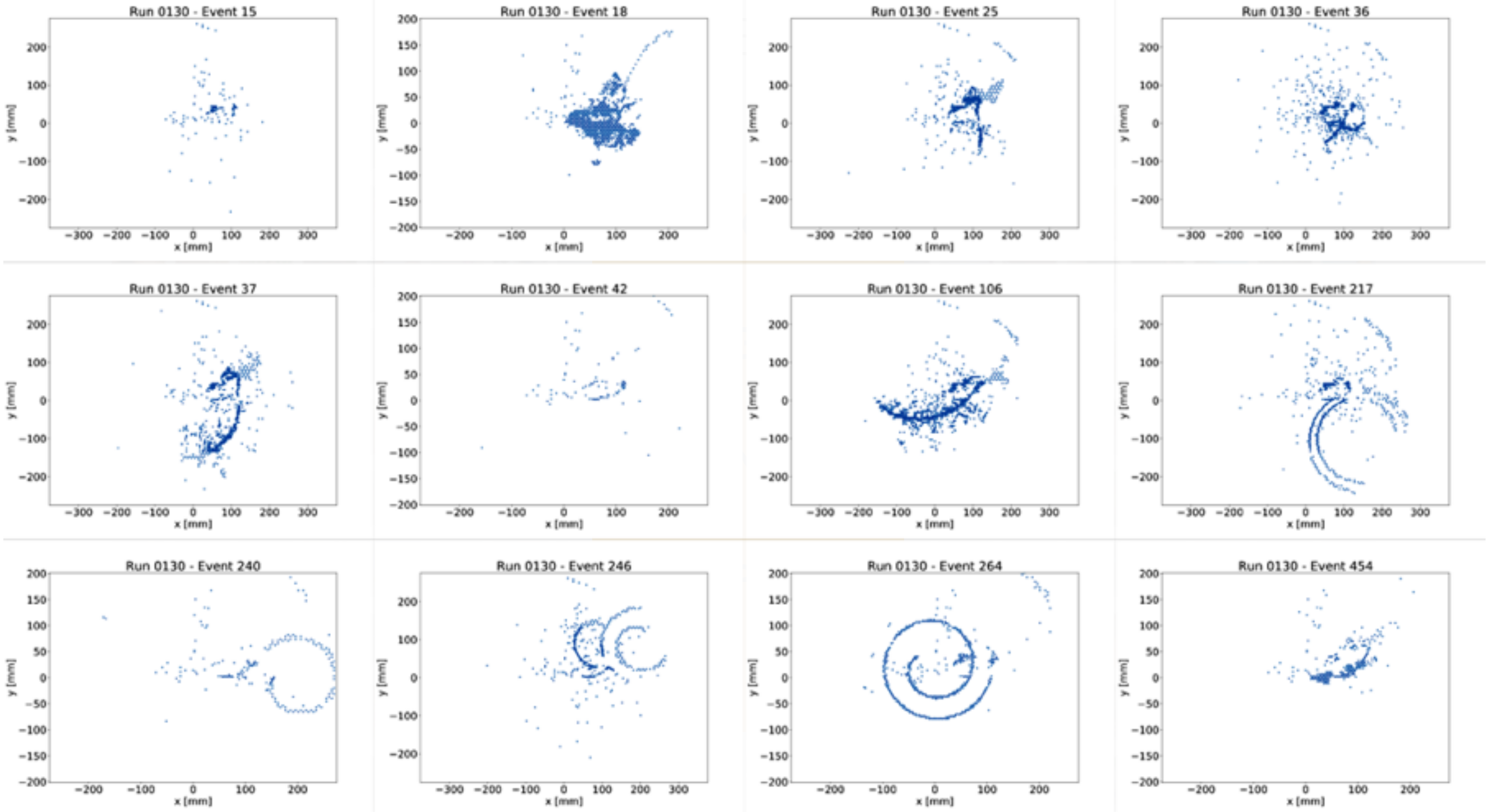
PRE-TRAINED MODELS



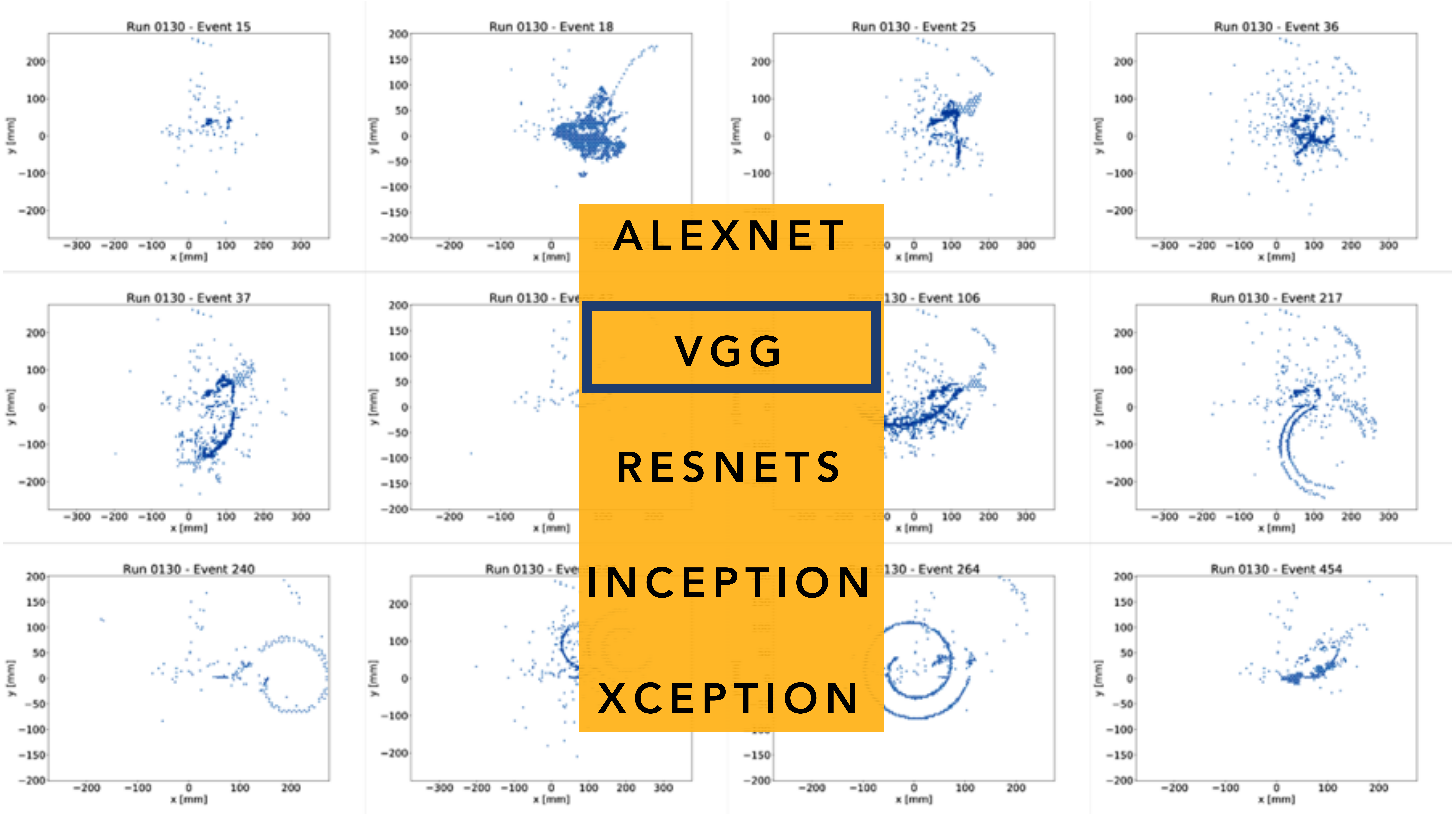
PRE-TRAINED MODELS



PRETRAINED MODELS



Run 0130 - Event 15
Run 0130 - Event 18
Run 0130 - Event 25
Run 0130 - Event 36
Run 0130 - Event 37
Run 0130 - Event 42
Run 0130 - Event 106
Run 0130 - Event 217
Run 0130 - Event 240
Run 0130 - Event 246
Run 0130 - Event 264
Run 0130 - Event 454
x [mm]
y [mm]



ALEXNET

VGG

RESNETS

INCEPTION

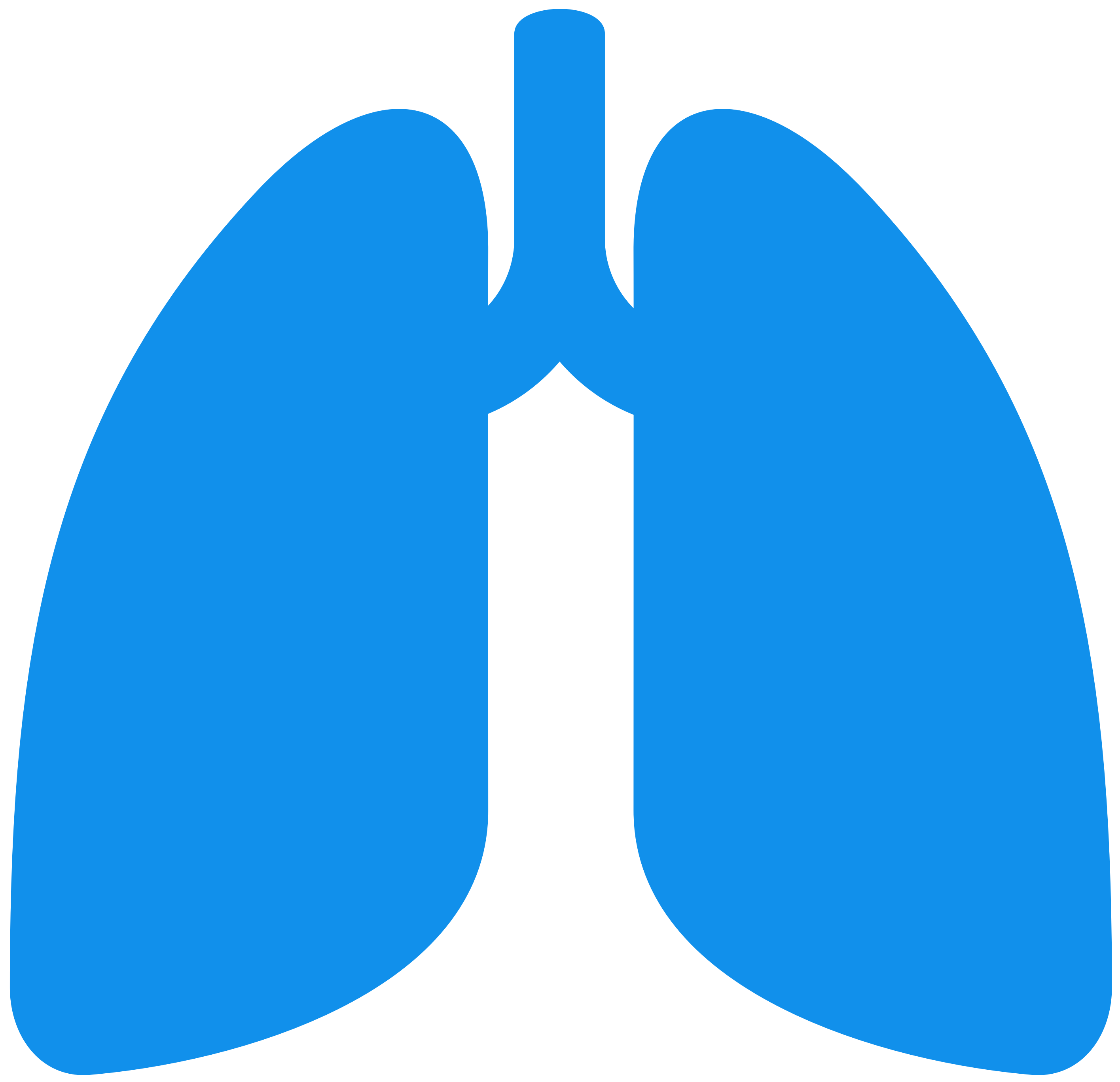
XCEPTION

J. Z. TAYLOR, HONOR'S THESIS, DAVIDSON COLLEGE

M.P. KUCHERA, ET. AL. NUCLEAR INSTRUMENTATION AND METHODS A (2019)

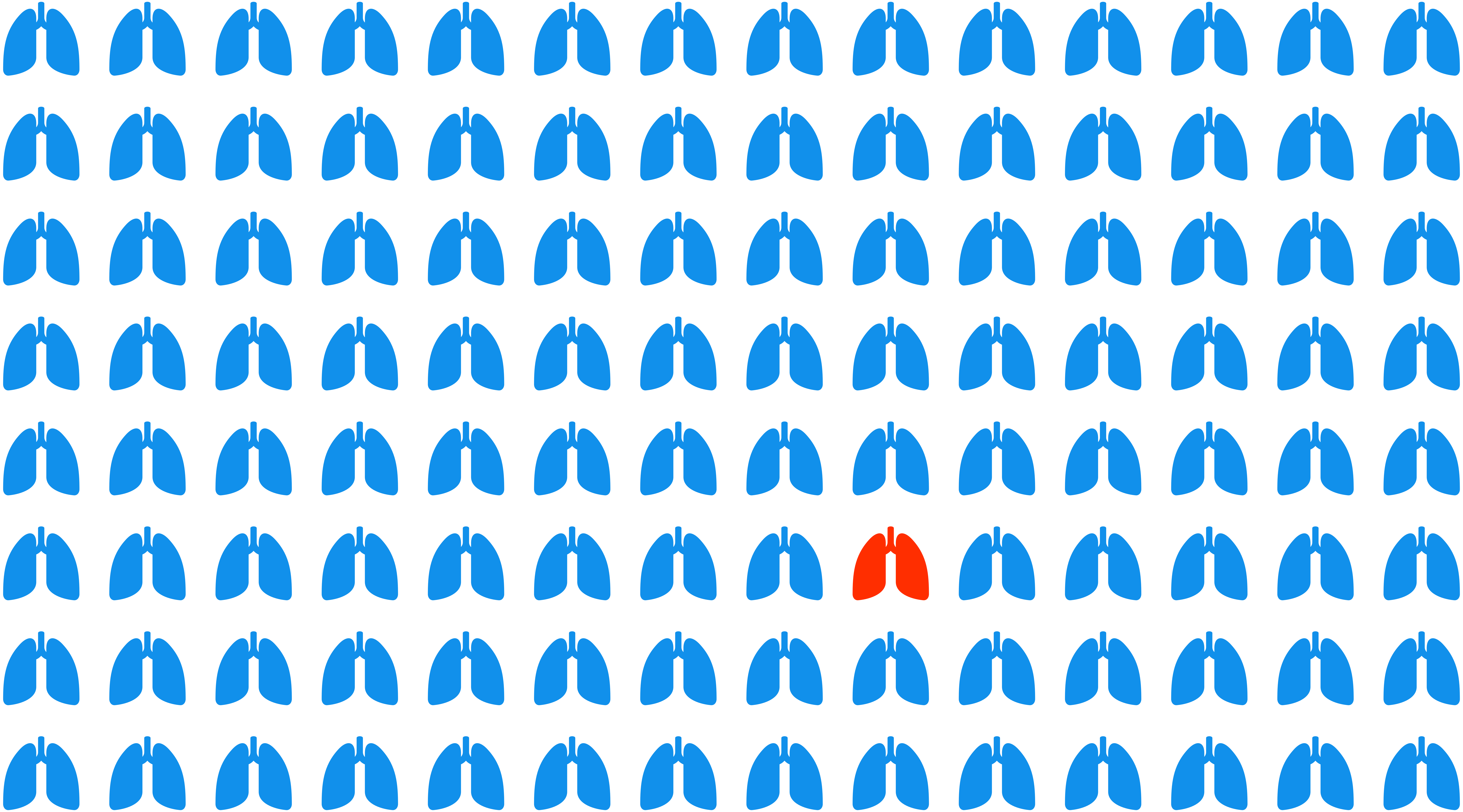
Application 2: Can we use machine learning to **accurately** classify events in detectors?

Metrics



Detect Lung Cancer

99% Accuracy



		PREDICTED	
		Proton	Not Proton
TRUE	Proton	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
	Not Proton	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

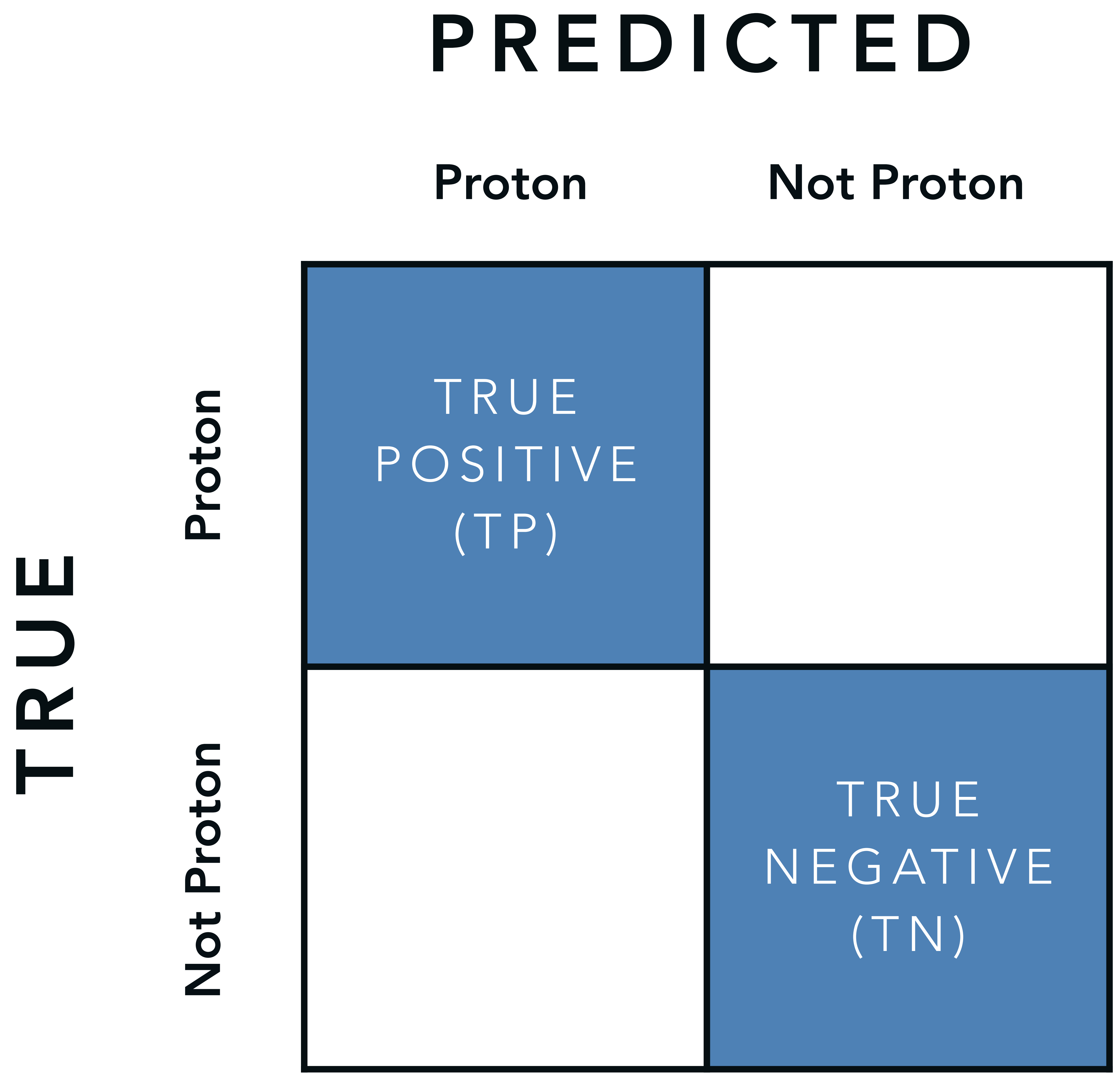
		PREDICTED	
		Proton	Not Proton
TRUE	Proton	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
	Not Proton	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

$$\text{accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

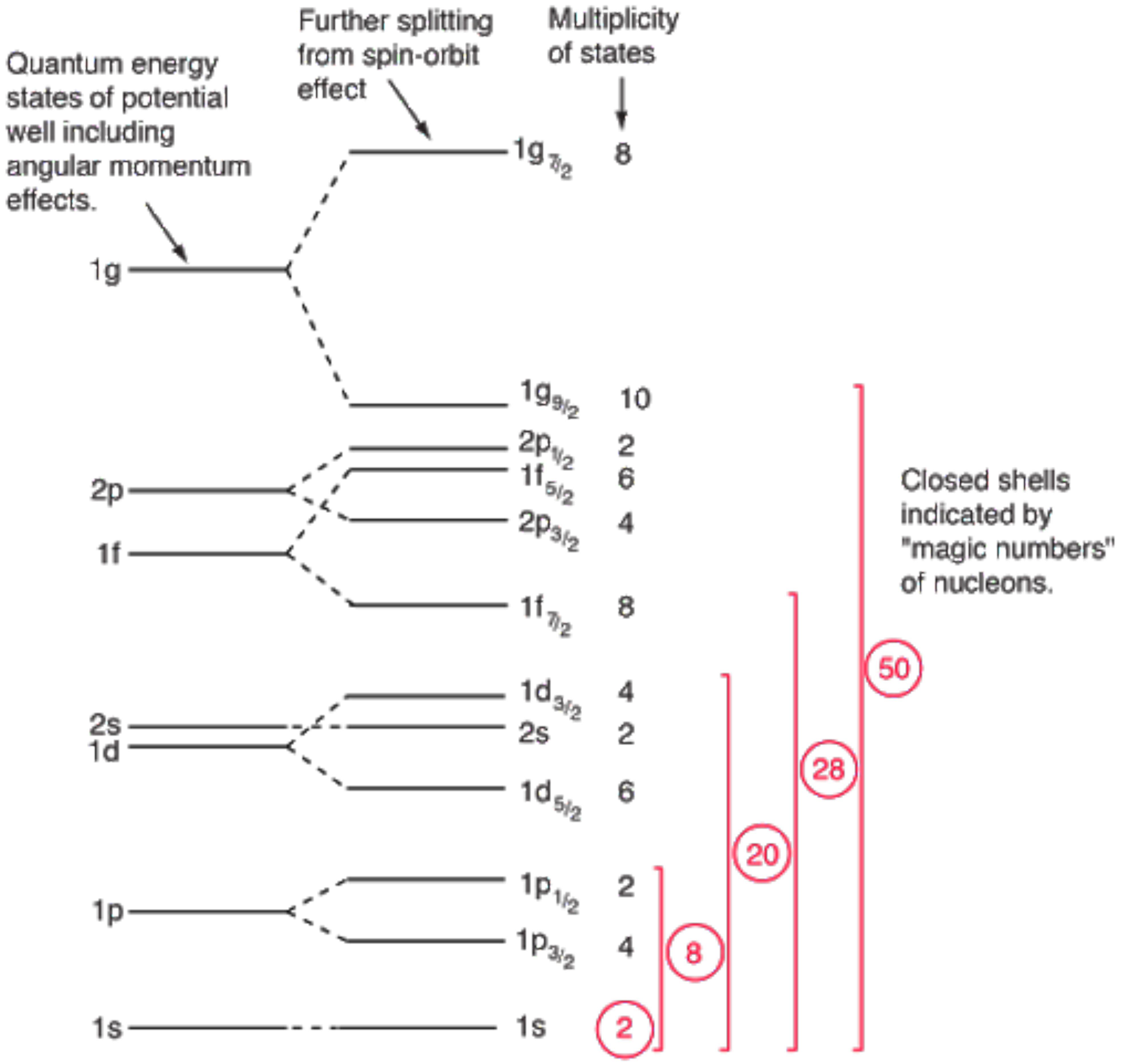
$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

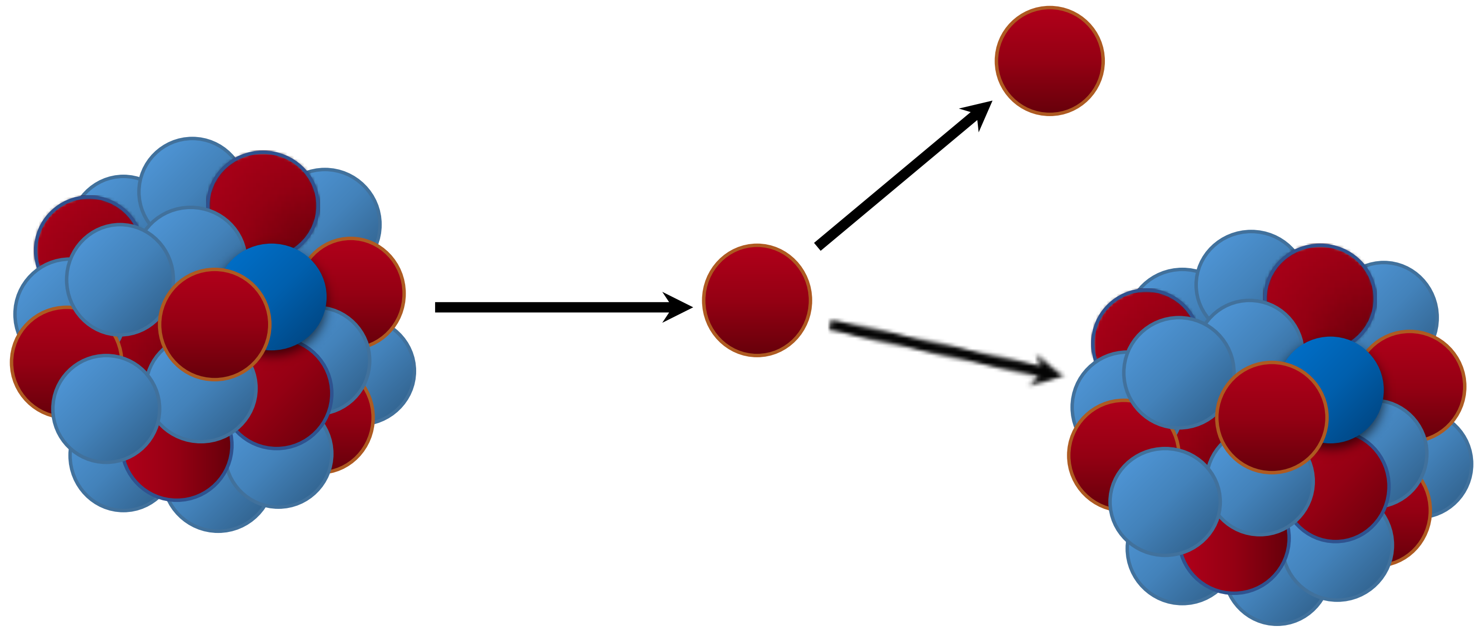


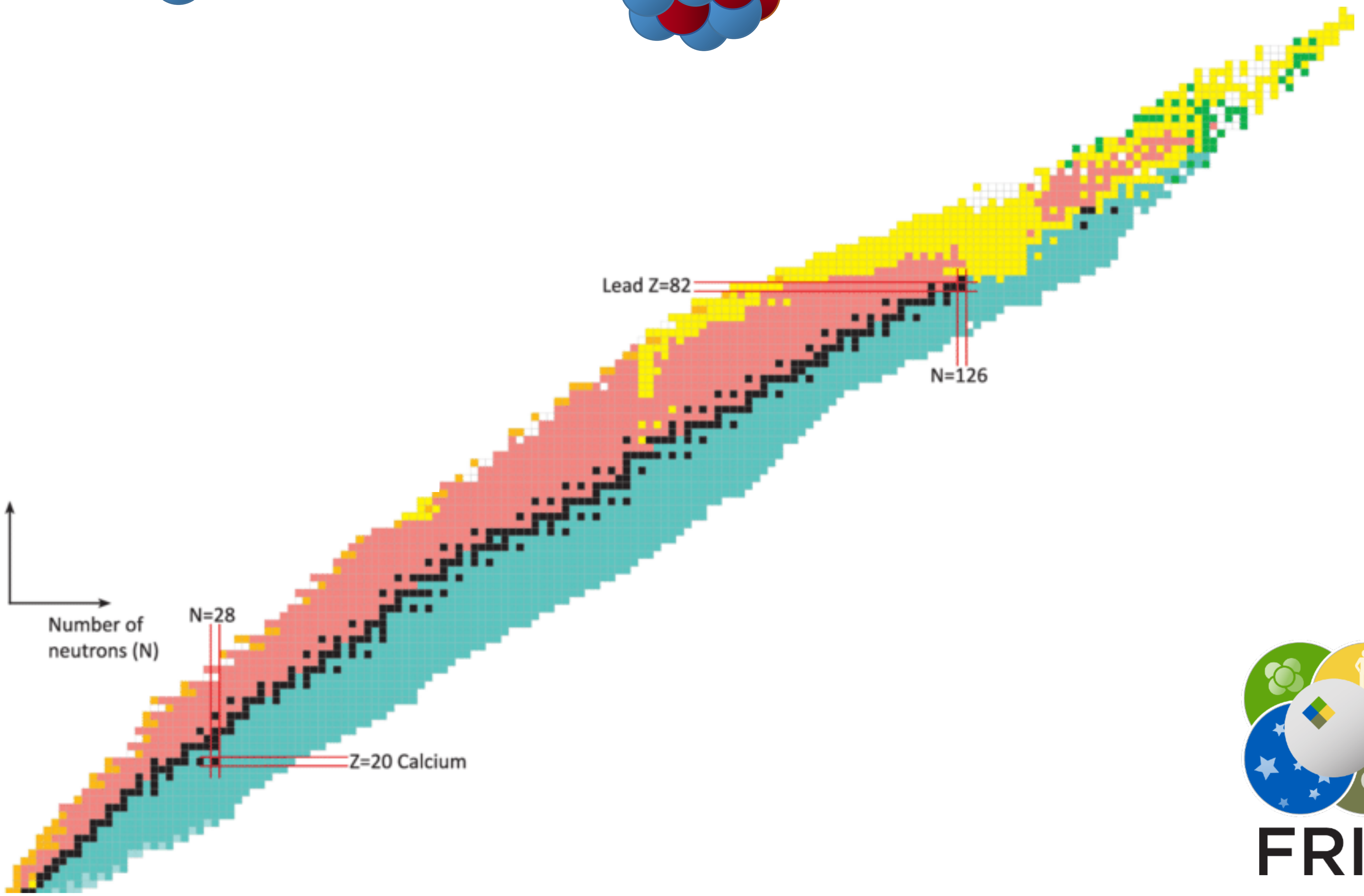
PERFECT MODEL

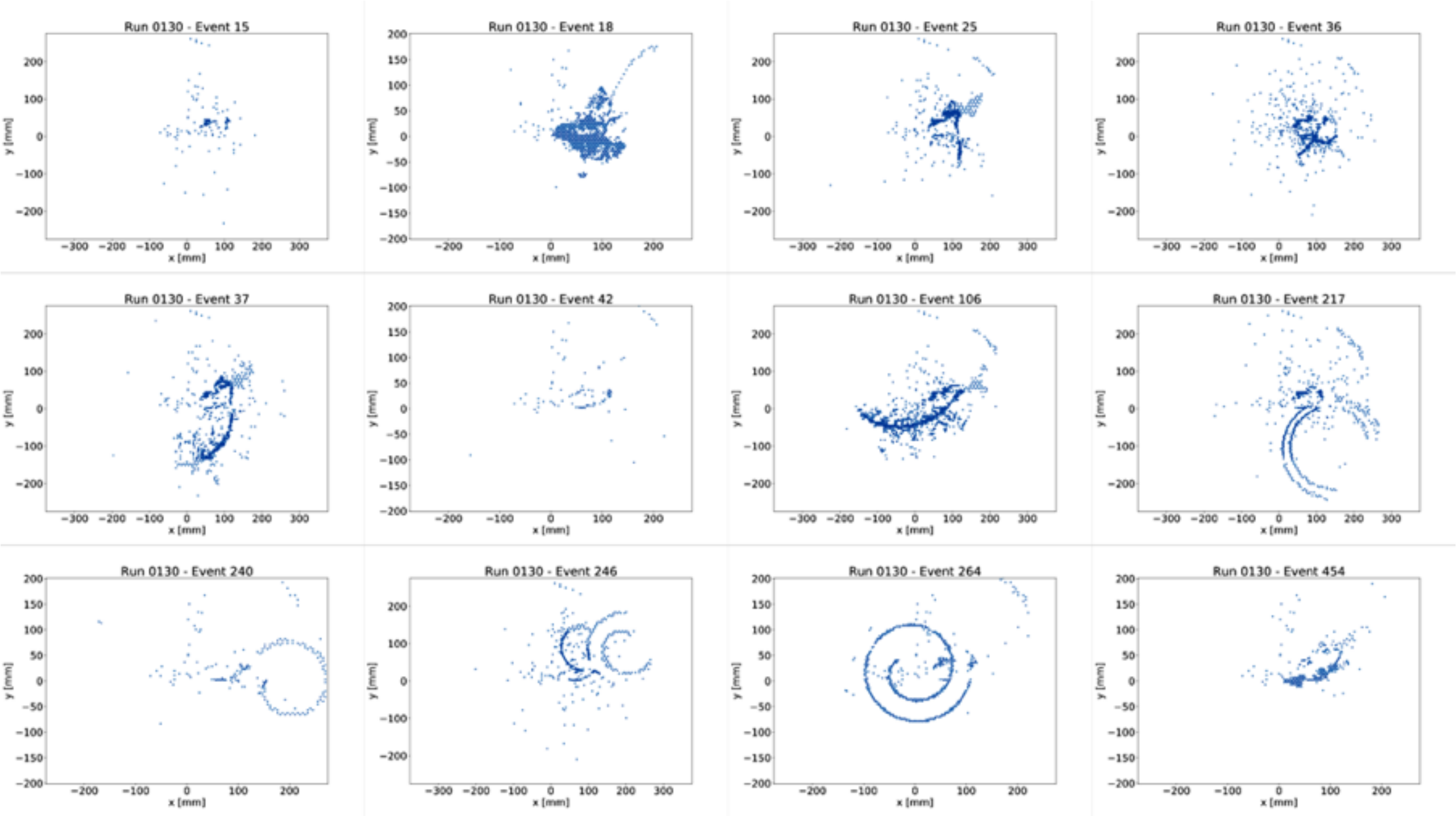
Application: Can we use machine learning to **accurately** classify events in detectors?

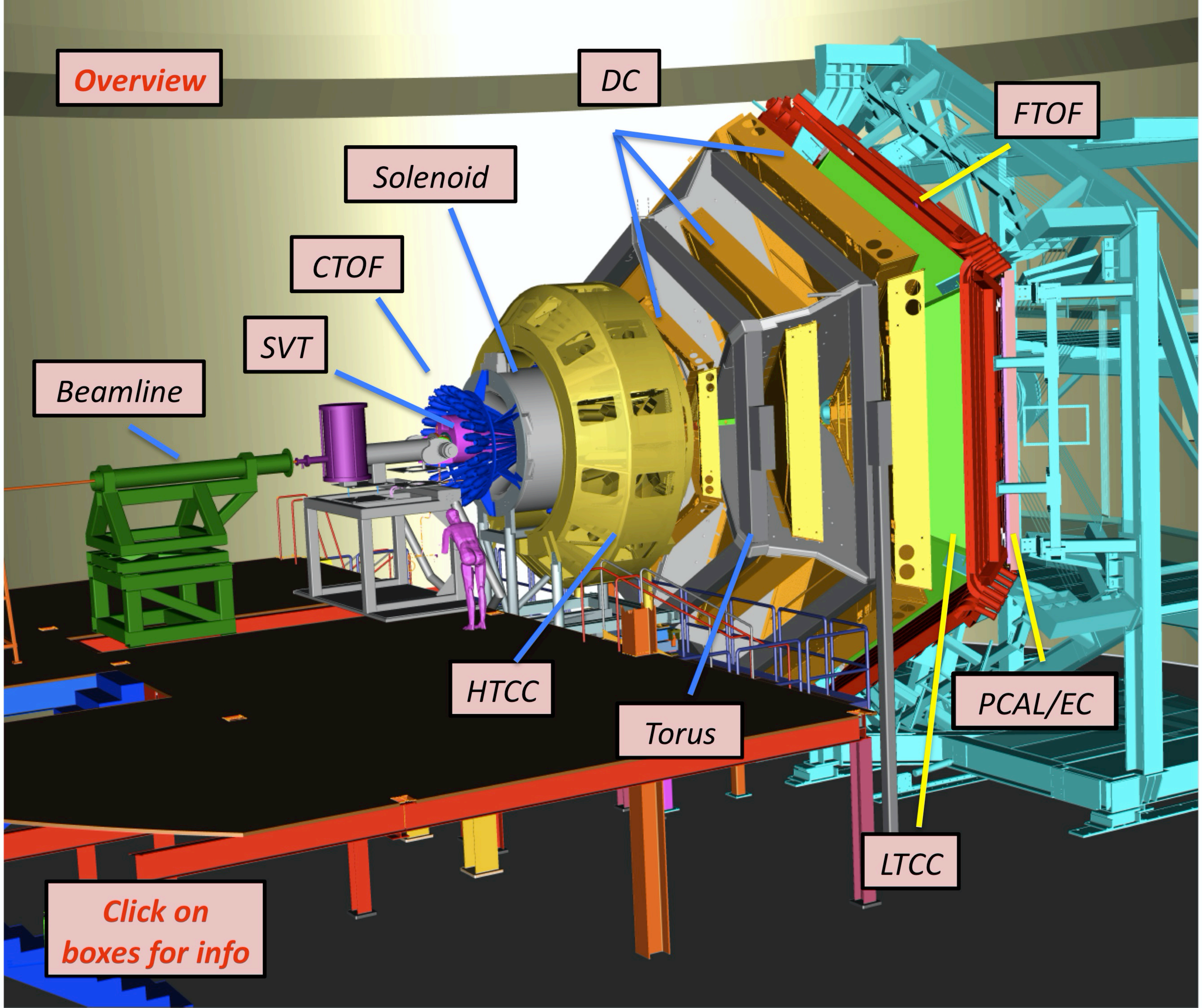
ACTIVE-TARGET TIME PROJECTION CHAMBER (AT-TPC)








Run 0130 - Event 15
Run 0130 - Event 18
Run 0130 - Event 25
Run 0130 - Event 36
Run 0130 - Event 37
Run 0130 - Event 42
Run 0130 - Event 106
Run 0130 - Event 217
Run 0130 - Event 240
Run 0130 - Event 246
Run 0130 - Event 264
Run 0130 - Event 454
x [mm]
y [mm]



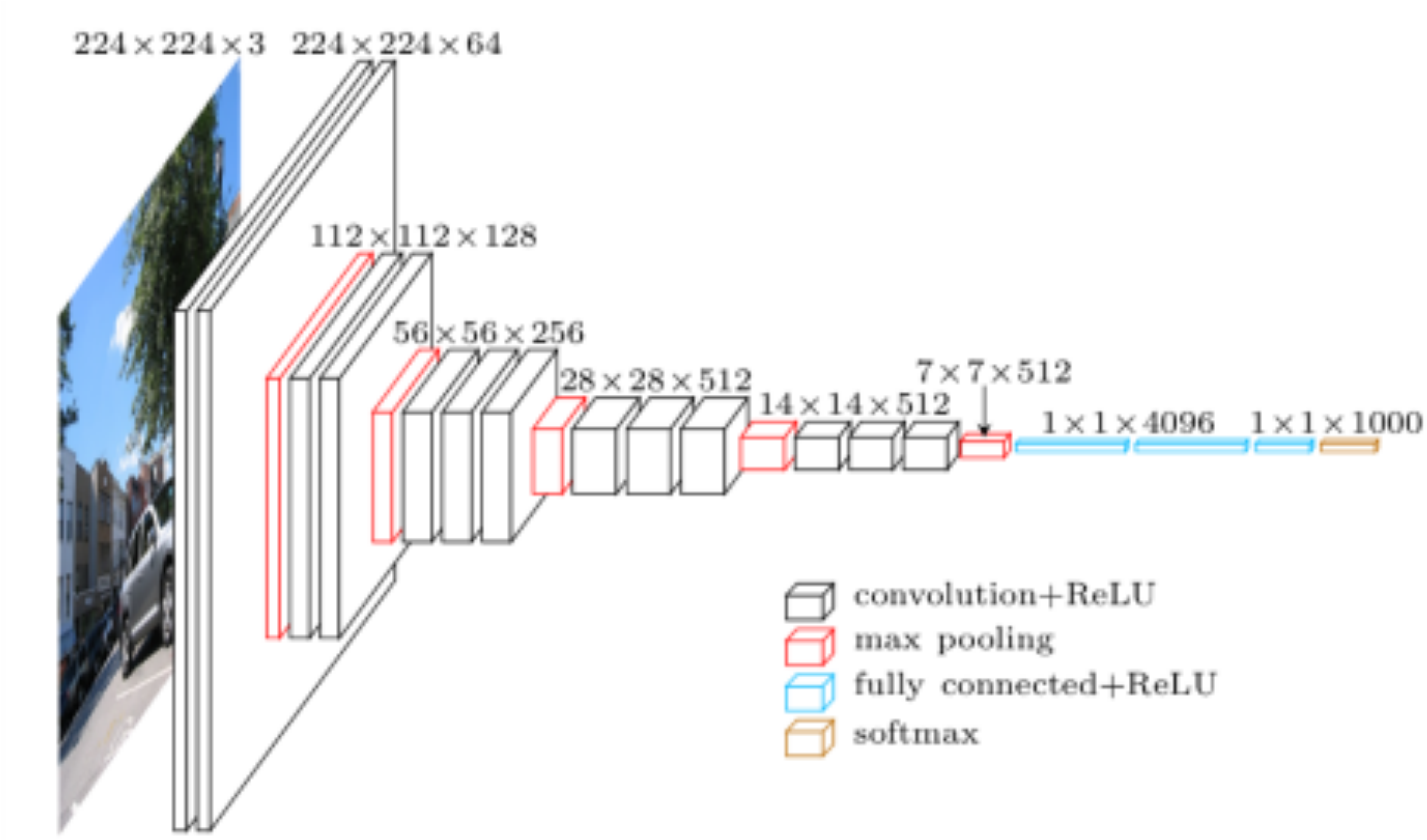
Overview
DC
FTOF
Solenoid
CTOF
SVT
Beamline
HTCC
Torus
PCAL/EC
LTCC
Click on boxes for info

EXPERIMENTAL DATA

AT-TPC

HALL B

VGG16 ARCHITECTURE



PRE-TRAINED ON IMAGENET DATA!

AT-TPC

Experiment	Precision	Recall	F1
Experimental → Experimental	0.96	0.90	0.93
Simulated → Simulated	1.00	1.00	1.00
Simulated → Experimental	0.90	0.60	0.72

HALL B

Precision	Recall	F1
0.97	0.93	0.95

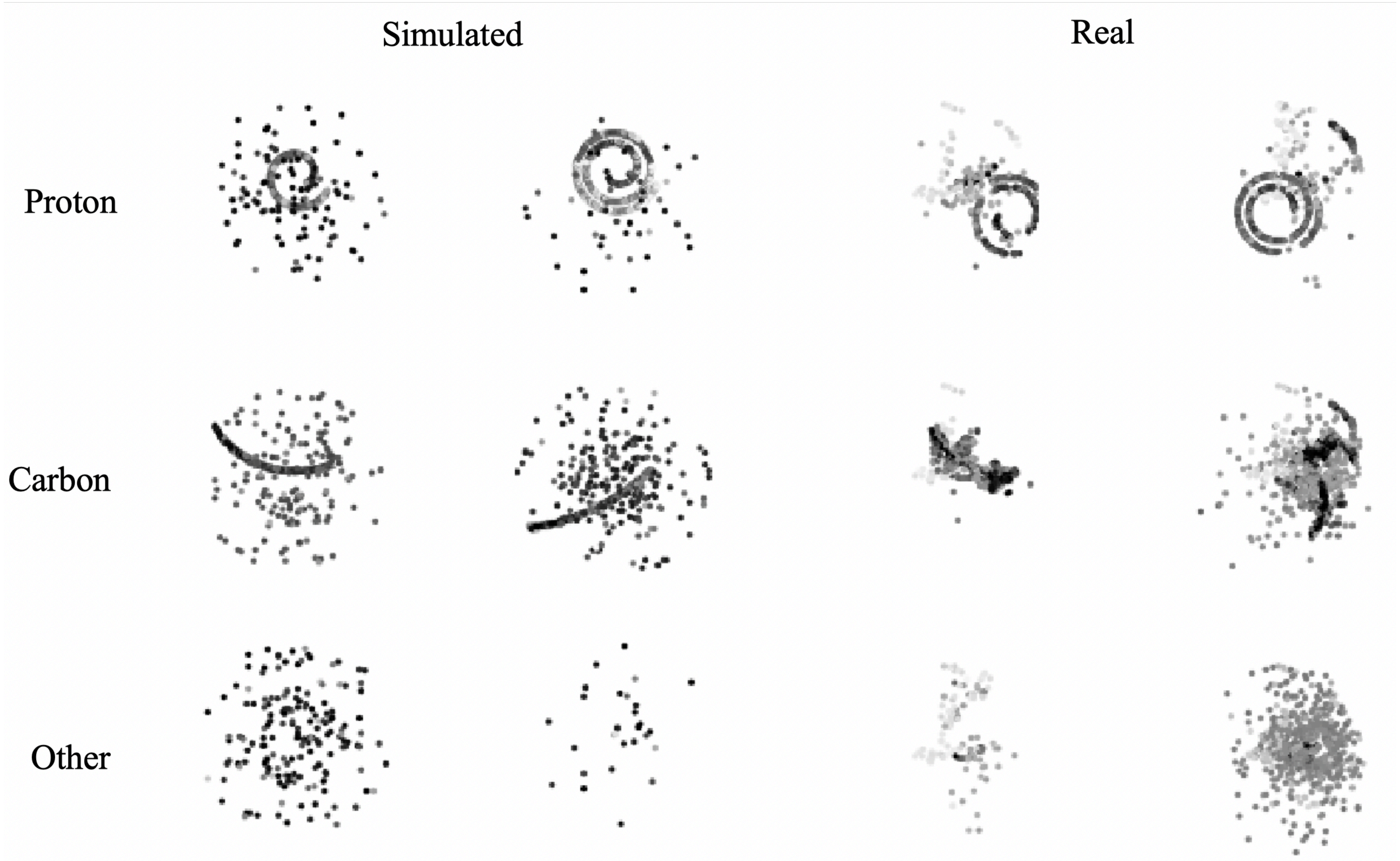
AT-TPC

Experiment	Precision	Recall	F1
Experimental → Experimental	0.96	0.90	0.93
Simulated → Simulated	1.00	1.00	1.00
Simulated → Experimental	0.90	0.60	0.72

HALL B

Precision	Recall	F1
0.97	0.93	0.95

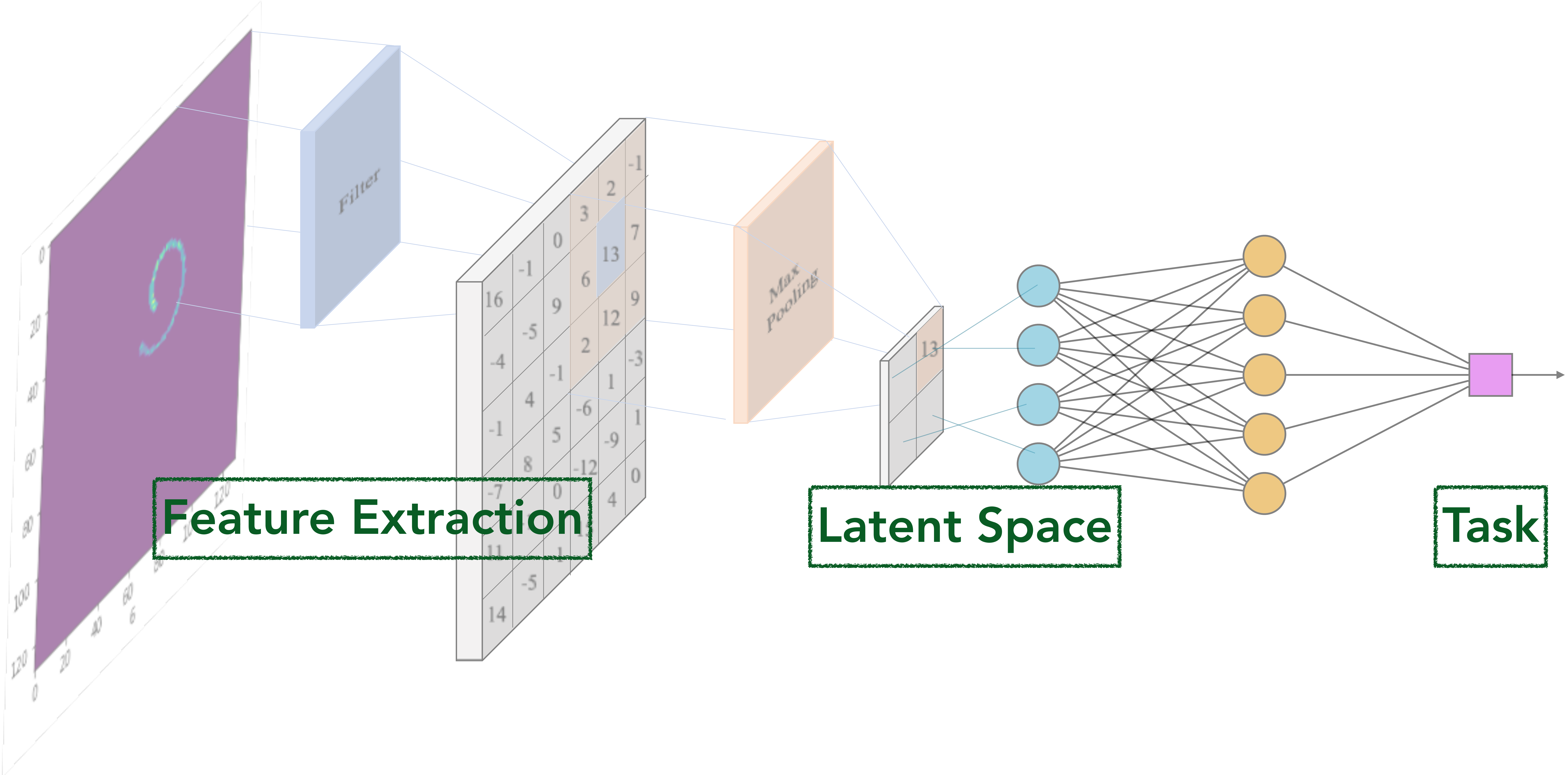
6x faster!


Simulated
Real
Proton
Carbon
Other

MACHINE LEARNING

UNSUPERVISED LEARNING

CONVOLUTIONAL NEURAL NETWORKS



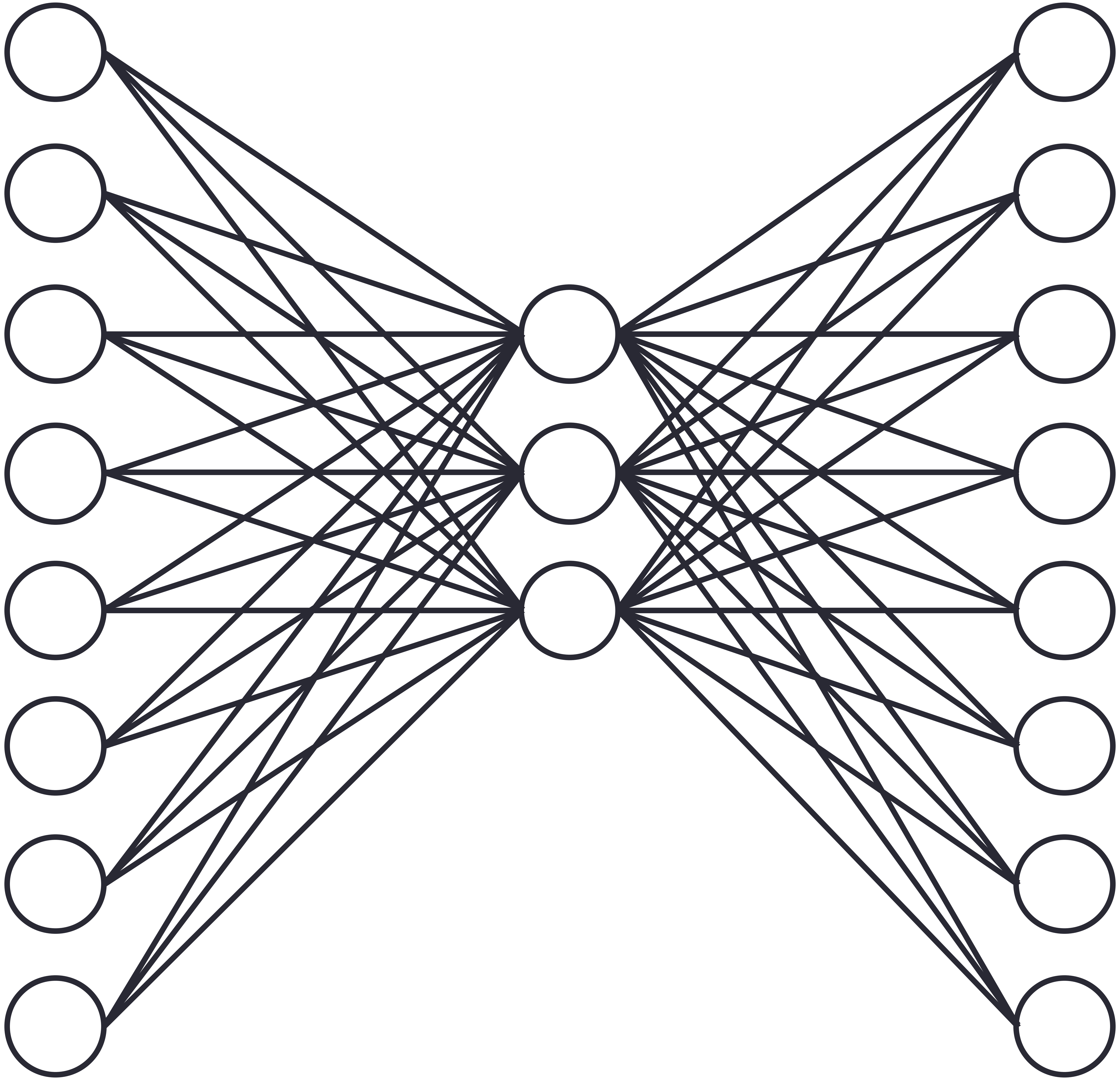
CLUSTERING — KMEANS

Goal: minimize pairwise distances between points in *same* cluster

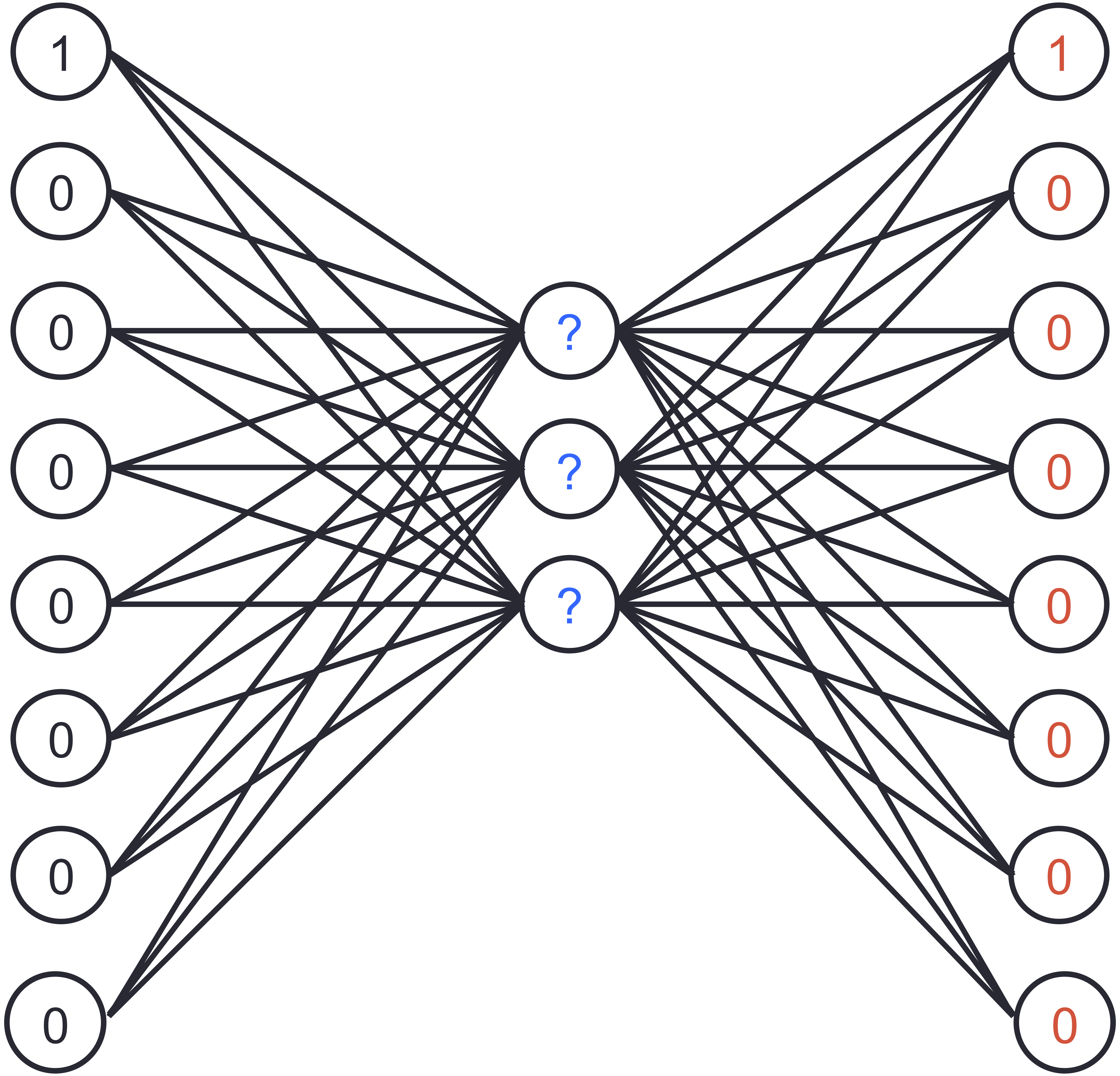
$$\min \sum_{i=1}^{k} \frac{1}{2N} \sum_{x,y,x\neq y}^{N} (\vec{x} - \vec{y})^2$$

Goal: maximize pairwise distances between points in *different* clusters

CLUSTERING — KMEANS

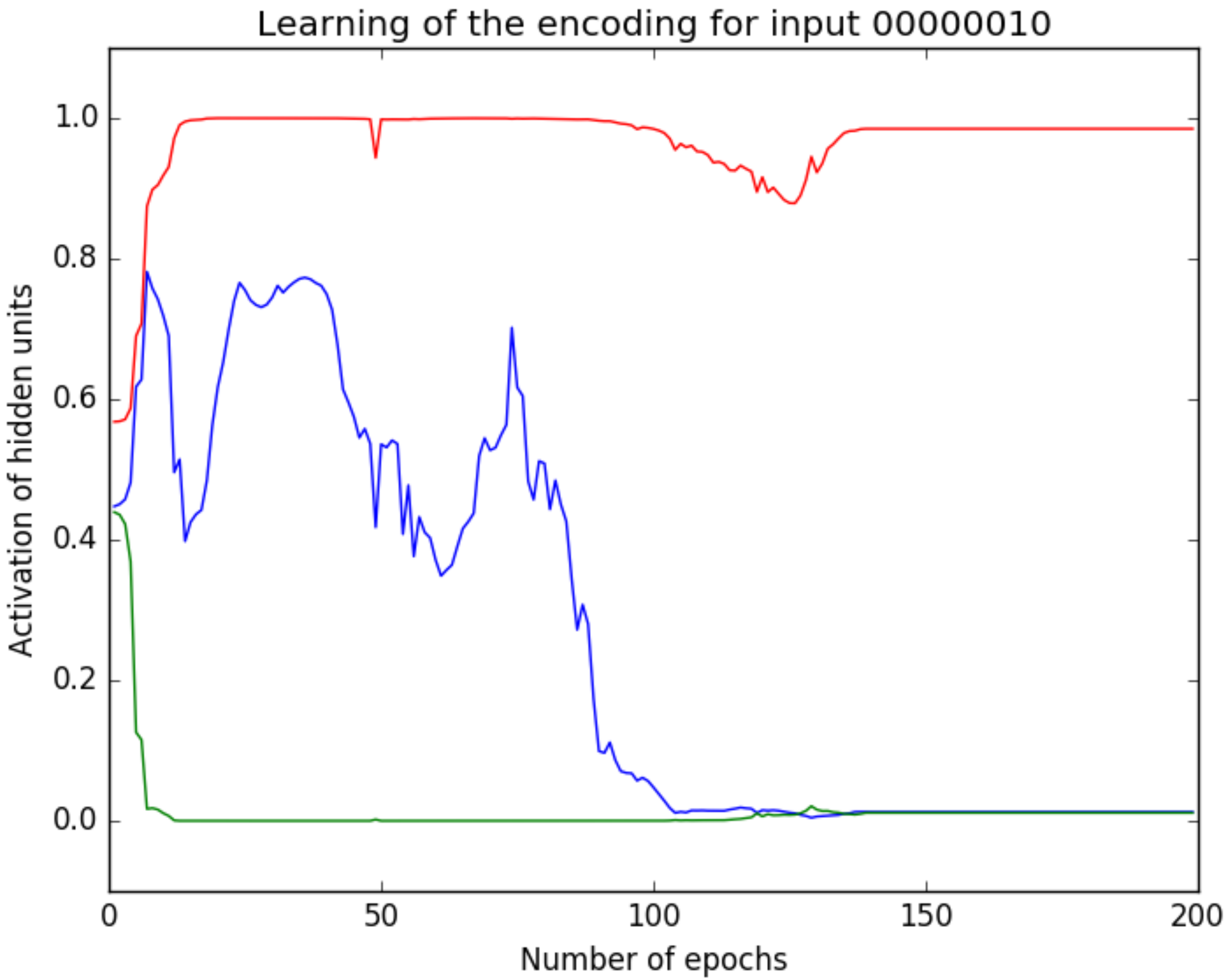


Input	Output
10000000	10000000
01000000	01000000
00100000	00100000
00010000	00010000
00001000	00001000
00000100	00000100


1
0
0
0
0
0
0
0
?
?
?
1
0
0
0
0
0
0
0

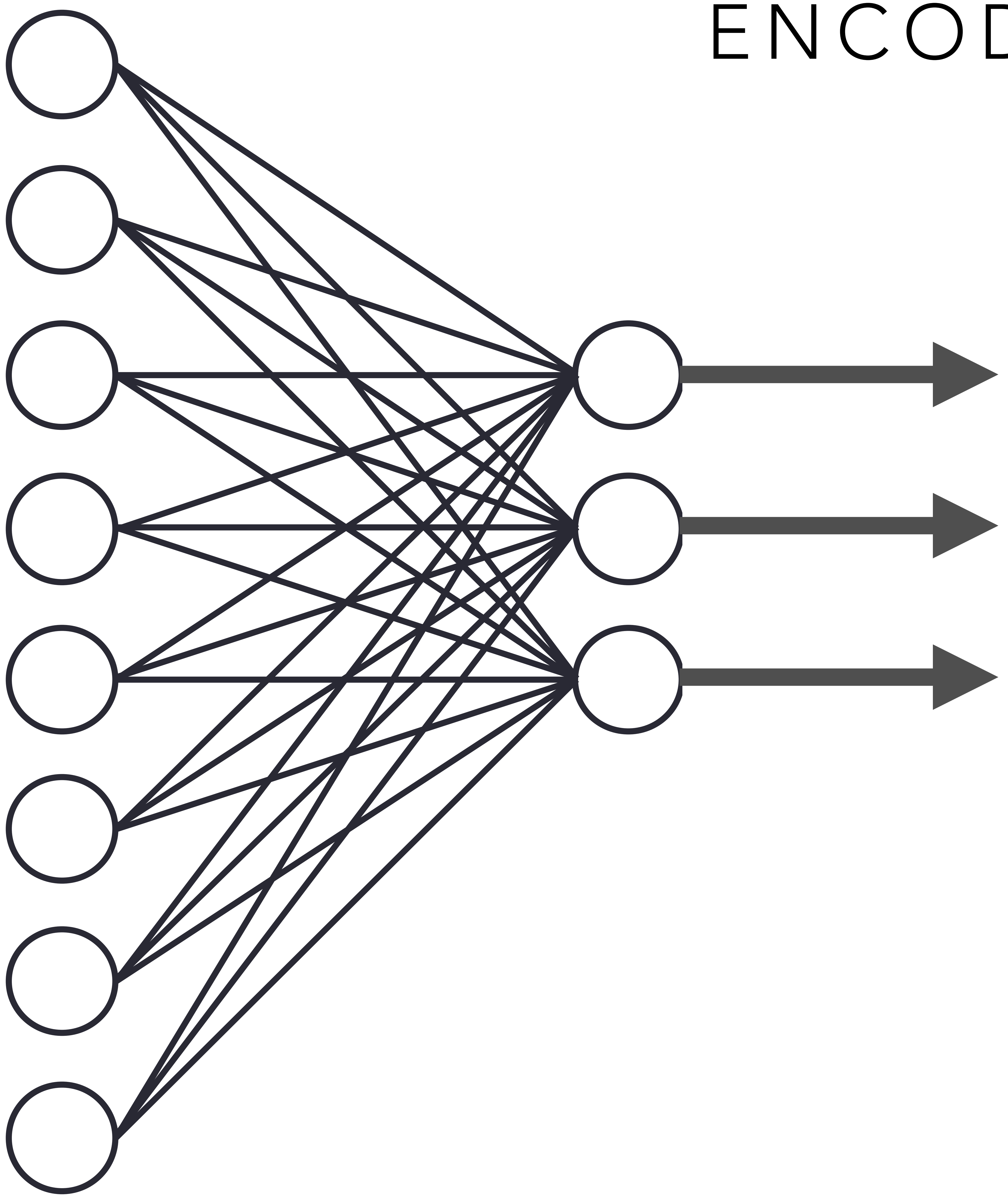
Input	Output
10000000	10000000
01000000	01000000
00100000	00100000
00010000	00010000
00001000	00001000
00000100	00000100

Input	A1	A2	A3	Output
10000000	0.9911	0.9869	0.0093	10000000
01000000	0.9892	0.0095	0.0124	01000000
00100000	0.0094	0.0283	0.0122	00100000
00010000	0.9840	0.9836	0.9900	00010000
00001000	0.0139	0.9904	0.0186	00001000
00000100	0.0128	0.9805	0.9868	00000100



Learning of the encoding for input 00000010
Activation of hidden units
Number of epochs
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200

ENCODER

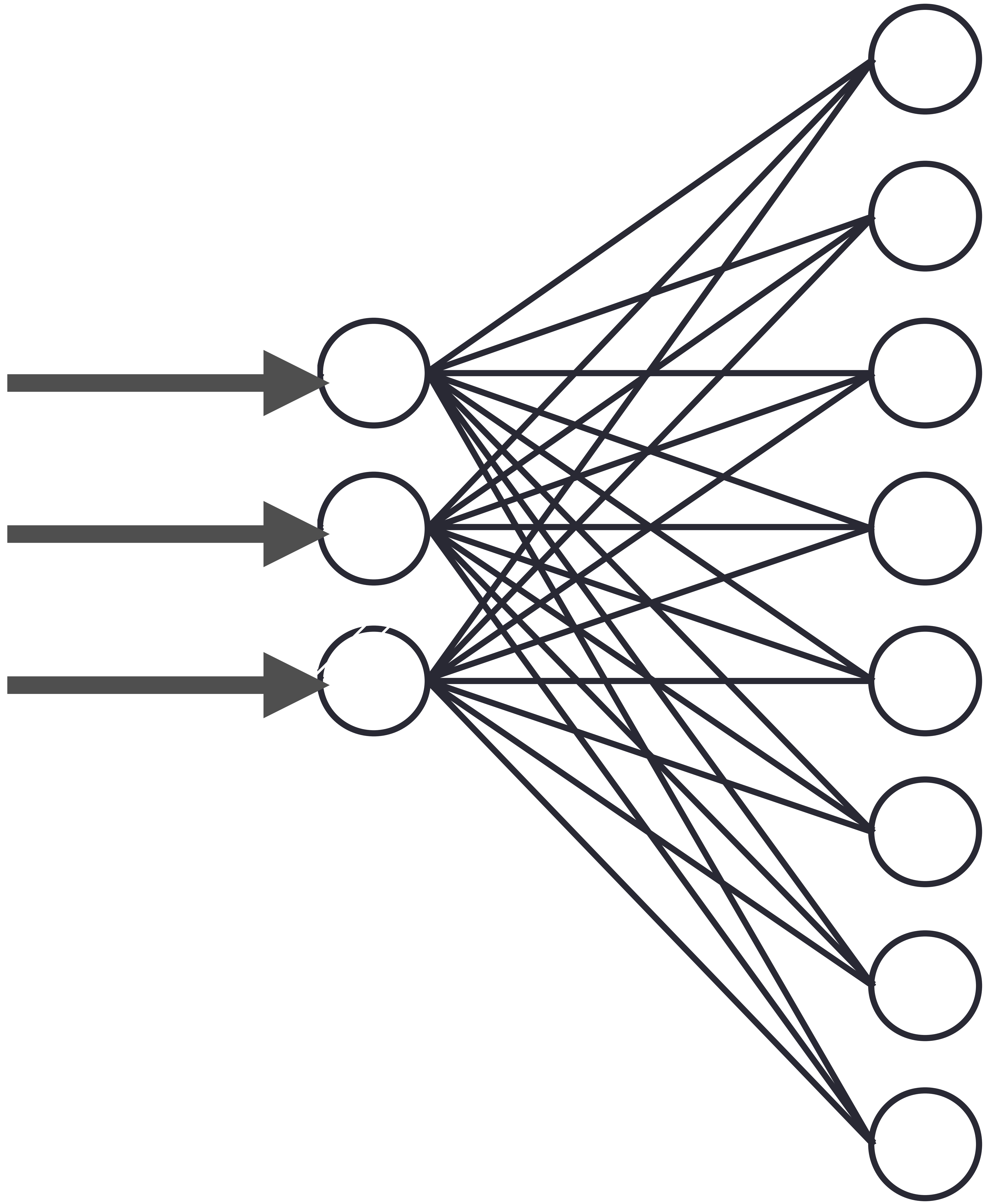


GENERATIVE MODELS

MICHELLE KUCHERA
DAVIDSON COLLEGE

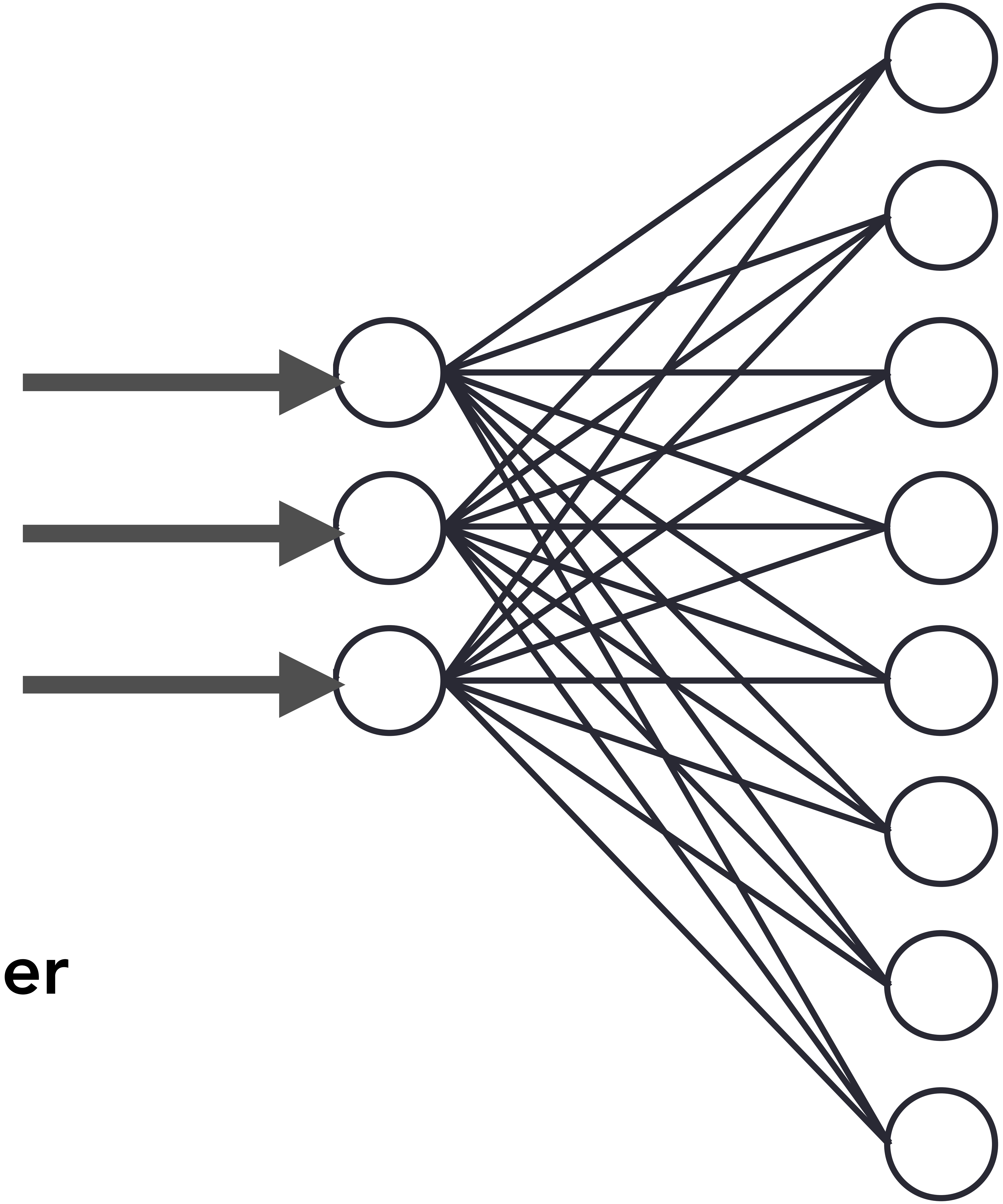
ECT* TALENT SUMMER SCHOOL
02 JULY 2020

DECODER



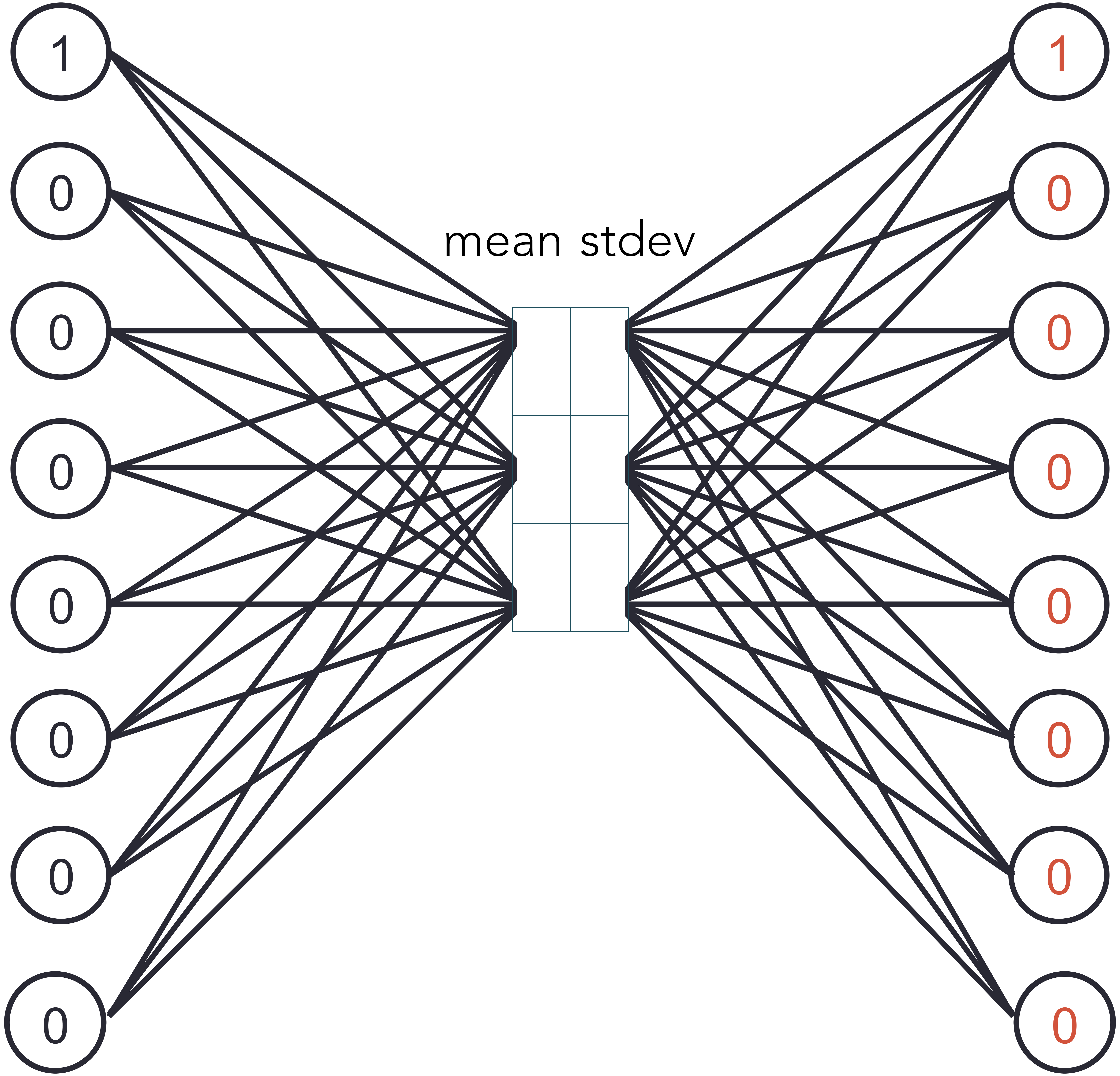
DECODER

How do we know that we are providing a latent vector that represents those seen in training?

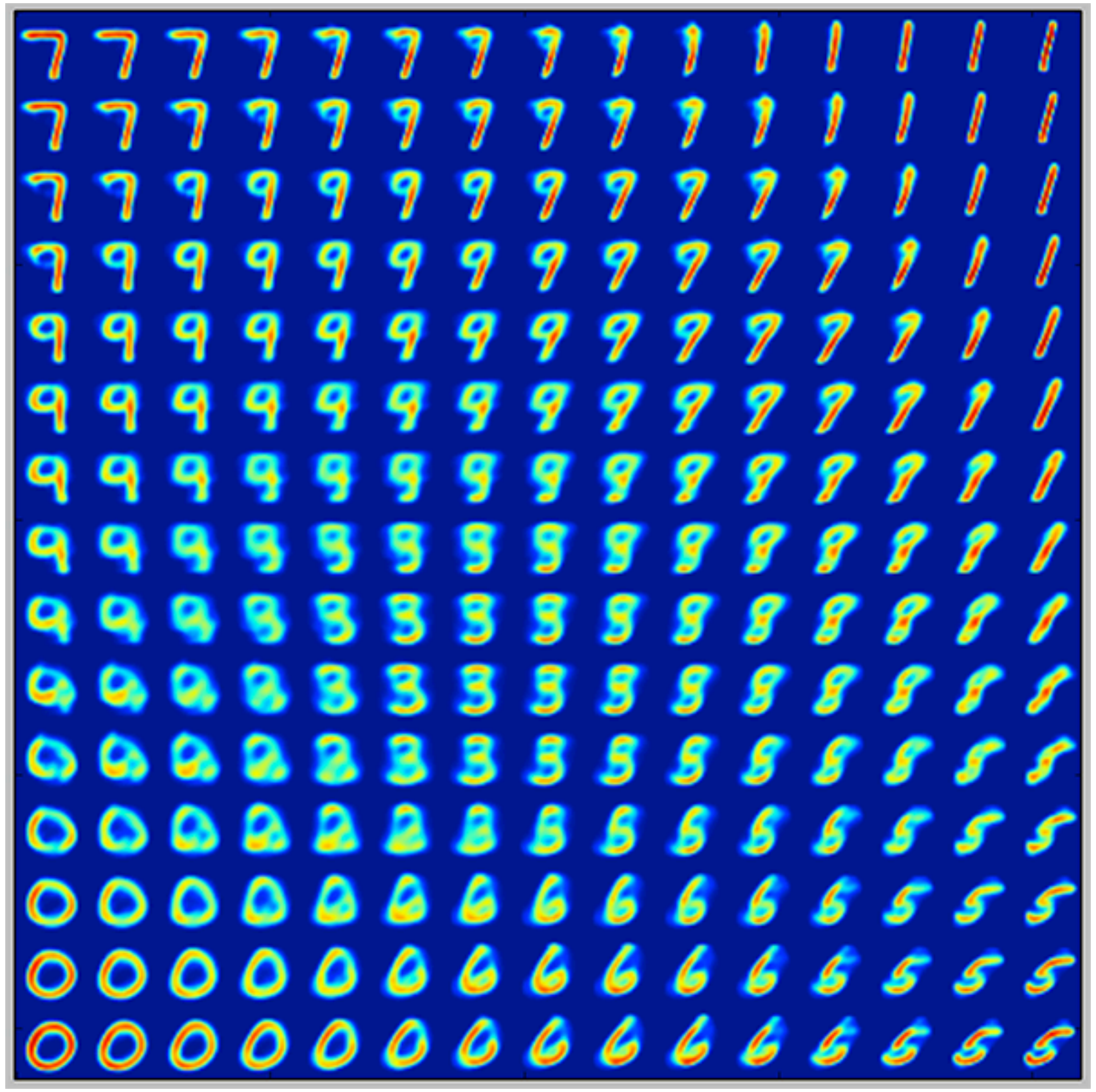


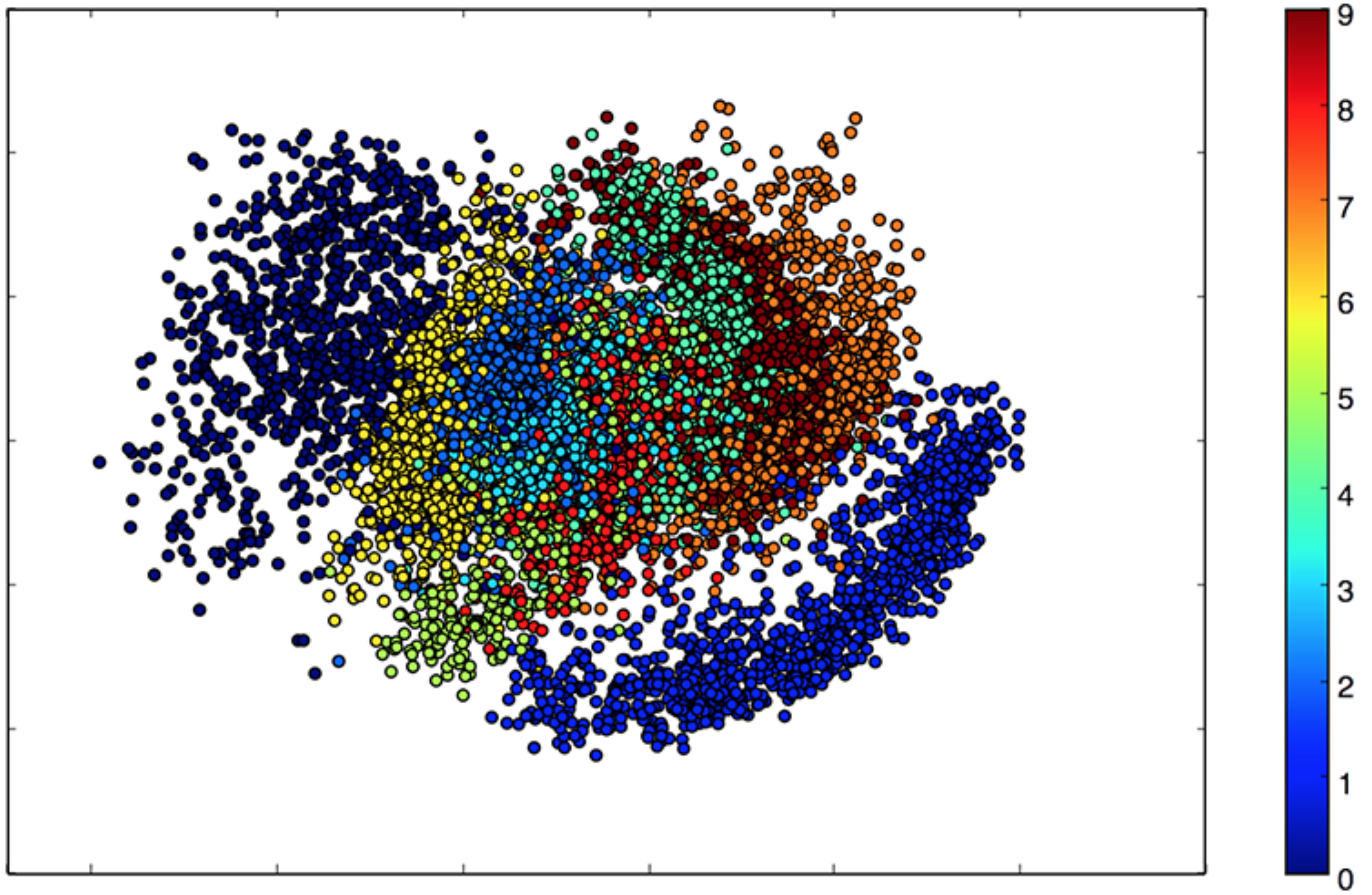
Variational Autoencoder

Encode to two outputs for each latent dimension: mean and stdev



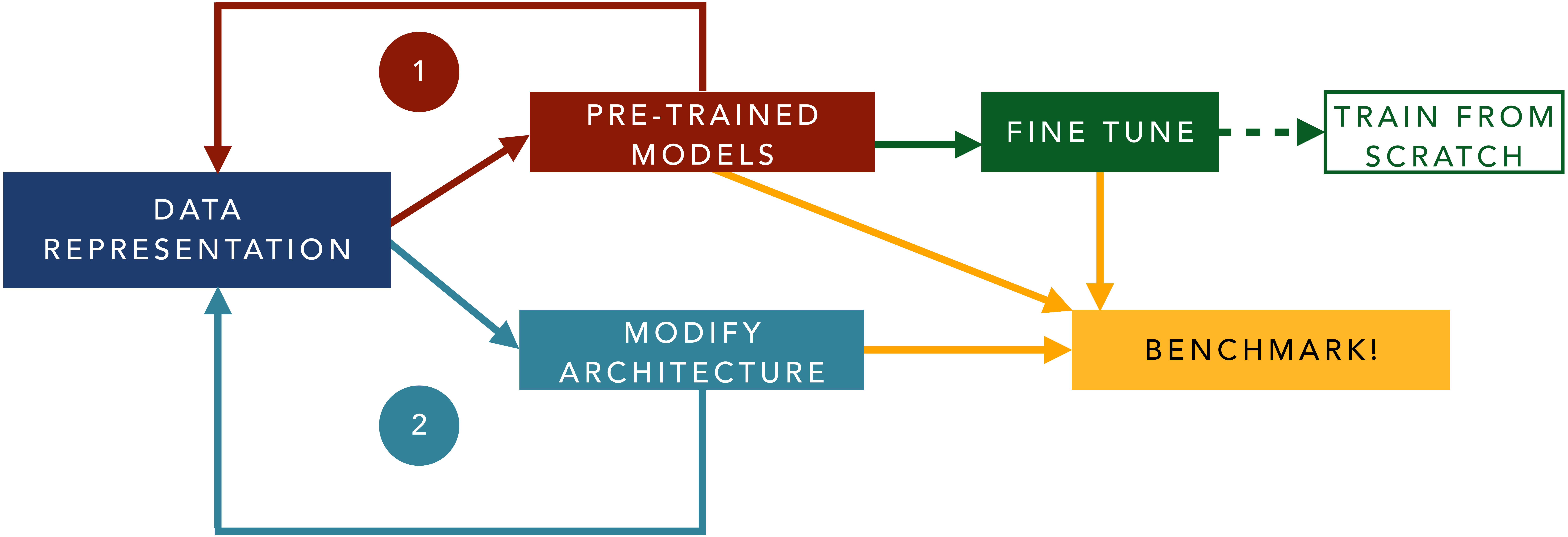
Sample similar points in latent space, decode, and compare with regularization



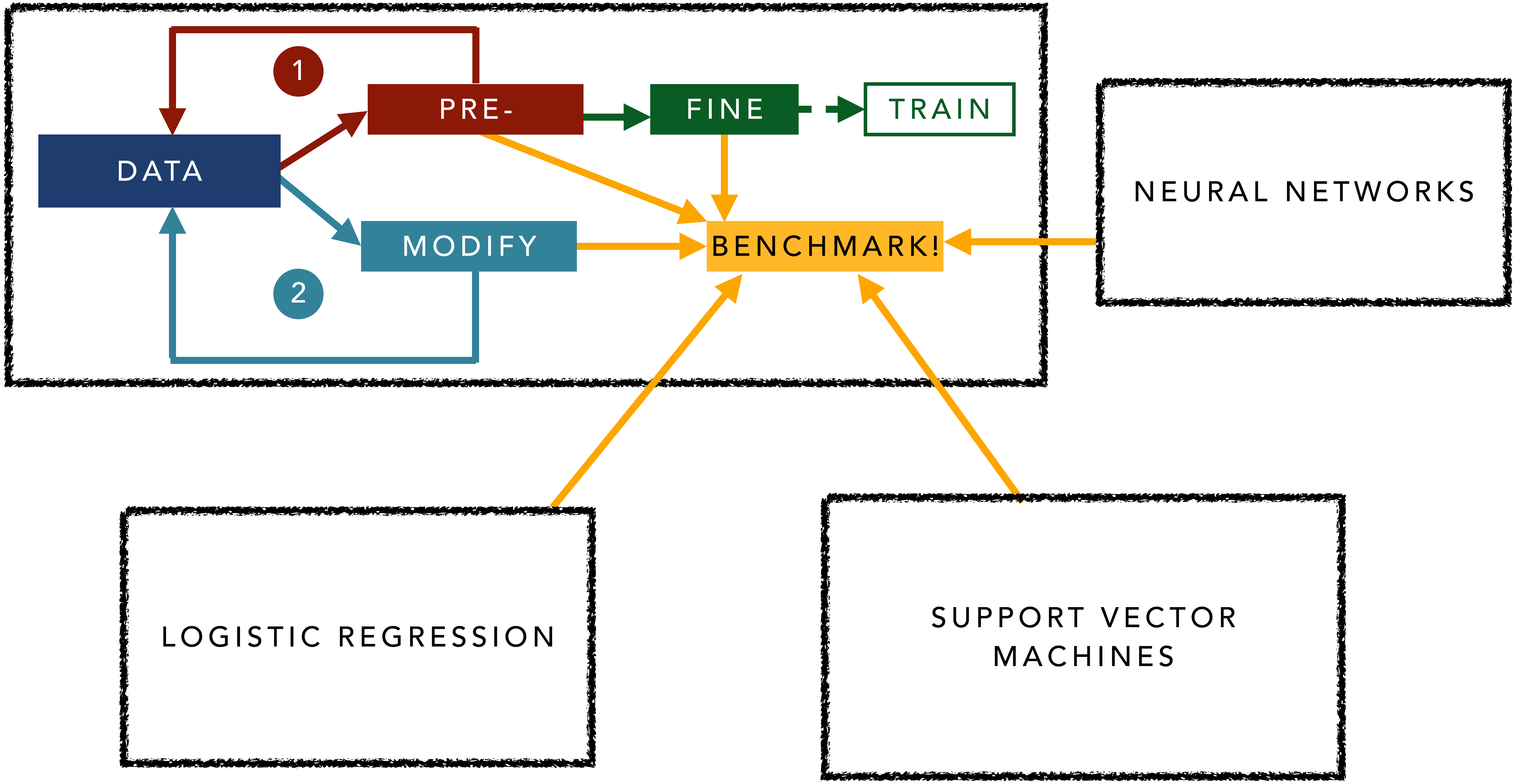


https://blog.keras.io/building-autoencoders-in-keras.html

EXAMPLE WORKFLOW



EXAMPLE WORKFLOW



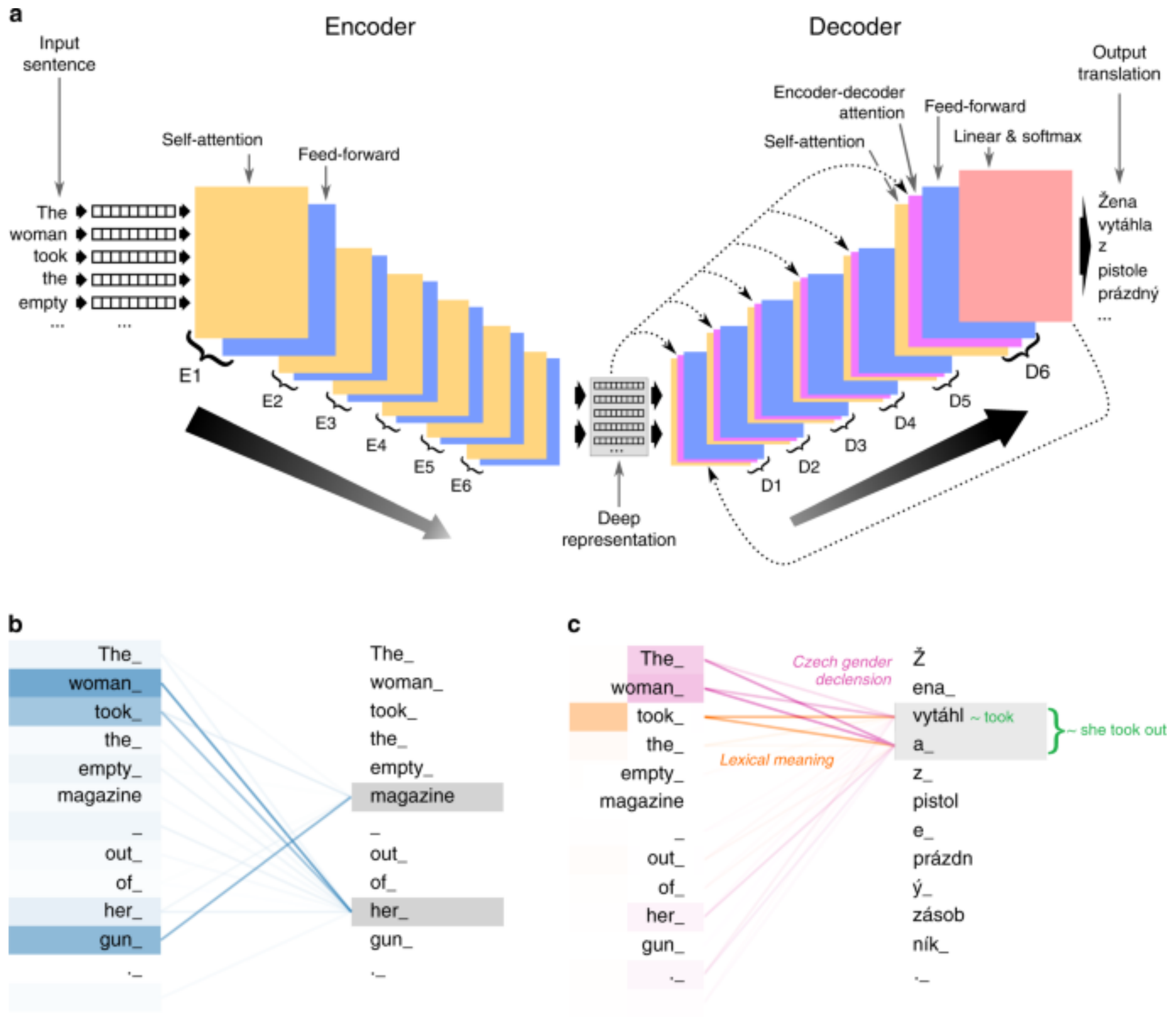
SELECT CURRENT HOT TOPICS IN AI*

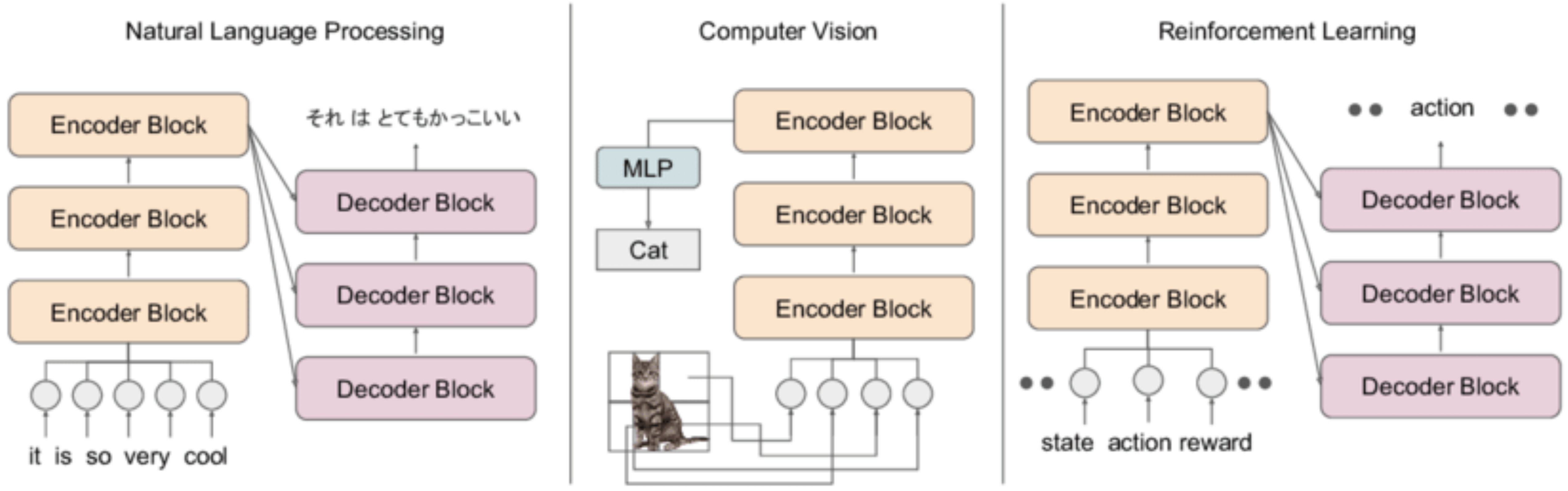
* A SOMEWHAT ANECDOTAL PERSPECTIVE

SELECT HOT TOPICS

- Transformers
- Normalizing Flows
- Zero shot learning
- Ethics

TRANSFORMERS





- large language models

- "attention"

TRANSFORMERS

Problem Description

```
# RATING: 1200
# TAGS: sortings
# LANGUAGE IS python3
# CORRECT SOLUTION
# It is the easy version of the problem. The only difference is that in this
# version n = 1.
#
# In the cinema seats can be represented as the table with n rows and m
columns.
# The rows are numbered with integers from 1 to n. The seats in each row are
# numbered with consecutive integers from left to right: in the k-th row from m
# (k - 1) + 1 to m k for all rows 1 ≤ k ≤ n.
#
# 1| 2| ⋯| m - 1| m
# ---|---|---|---|---
# m + 1| m + 2| ⋯| 2 m - 1| 2 m
# 2m + 1| 2m + 2| ⋯| 3 m - 1| 3 m
# \vdots| \vdots| \ddots| \vdots| \vdots
# m (n - 1) + 1| m (n - 1) + 2| ⋯| n m - 1| n m
# The table with seats indices
#
# There are nm people who want to go to the cinema to watch a new film. They
are
# numbered with integers from 1 to nm. You should give exactly one seat to each
# person.
#
# It is known, that in this cinema as lower seat index you have as better you
# can see everything happening on the screen. i-th person has the level of
sight
# a_i. Let's define s_i as the seat index, that will be given to i-th person.
# You want to give better places for people with lower sight levels, so for any
# two people i, j such that a_i < a_j it should be satisfied that s_i < s_j.
#
# After you will give seats to all people they will start coming to their
seats.
# In the order from 1 to nm, each person will enter the hall and sit in their
# seat. To get to their place, the person will go to their seat's row and start
# moving from the first seat in this row to theirs from left to right. While
# moving some places will be free, some will be occupied with people already
# seated. The inconvenience of the person is equal to the number of occupied
# seats he or she will go through.
#
```

Solution

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 17 20:39:24 2020

@author: Dark Soul
"""

t=int(input(''))
arr=[]
for i in range(t):
    [n,m]=list(map(int,input().split()))
    arr.append(list(map(int,input().split())))
for i in arr:
    s=0
    arr=sorted(i)
    n1=len(i)
    for j in range(n1):
        cnt=0
        for k in range(j,n1):
            if i[k]>i[j]:
                cnt=cnt+1
        s=s+cnt
    print(s)
```

AlphaCode

sentiment.ts | write_sql.go | parse_expenses.py | addresses.rb

```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of text is positive
// Use a web service
async function isPositive(t
```

Powered by

OpenAI

sentiment.ts | write_sql.go | parse_expenses.py | addresses.rb

```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

//
//
asy
```

What data has GitHub Copilot been trained on?

GitHub Copilot is powered by OpenAI Codex, a new AI system created by OpenAI. It has been trained on a selection of English language and source code from publicly available sources, including code in public repositories on GitHub.

Powered by

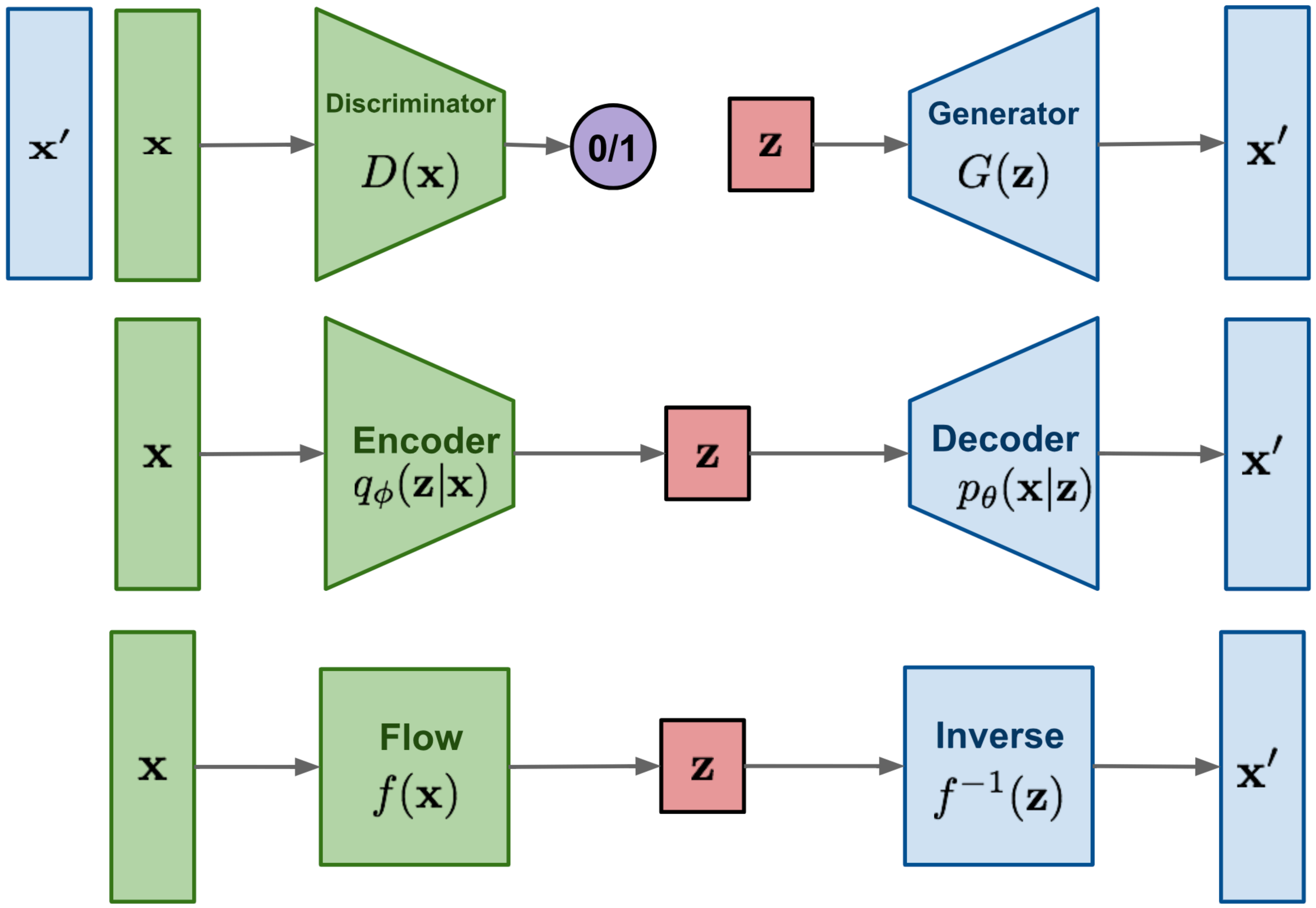
OpenAI

DISTRIBUTIONS

GAN: minimax the classification error loss.

VAE: maximize ELBO.

Flow-based generative models: minimize the negative log-likelihood



ZERO SHOT LEARNING

DALL·E 2 can create original, realistic images and art from a text description. It can combine concepts, attributes, and styles.

TEXT DESCRIPTION

An astronaut **Teddy bears** A bowl of soup

mixing sparkling chemicals as mad scientists **shopping for groceries** working on new AI research

in the style of ukiyo-e as a one-line drawing in ancient Egypt

→

DALL·E 2

ETHICS

Large, pertained models

— power (cost and authority)

— bias

image models

language models

LARGE MODELS

Largest jumps in ability currently due to extremely **large models** being trained on **our data**

Leads to ethical considerations

NEW RESEARCH

Conference papers

NeurIPS: Neural Information Processing Systems

ICML: International Conference on Machine Learning

IJCAI: International Joint Conference on Artificial Intelligence

FAccT: Fairness, Accountability, and Transparency

MACHINE LEARNING HANDS-ON WORKSHOP

MICHELLE KUCHERA

JOINT ICTP-IAEA ADVANCED SCHOOL/WORKSHOP ON MACHINE LEARNING IN CITIZEN SCIENCE

24 MAY 2022

PRACTICAL TIPS FOR TRAINING MODELS

DATA

	Feature 1	Feature 2	Feature 3	Target
Example 1				
Example 2				
Example 3				
Example 4				

NORMALIZATION

- Puts each feature on same scale
- Allows default hyperparamters to be a good starting point
  - learning rate, initialization of weights, etc.
- Options depend on data distribution
  - Standardization: mean: 0 stdev: 1
  - Min-max: [0,1]

DATA

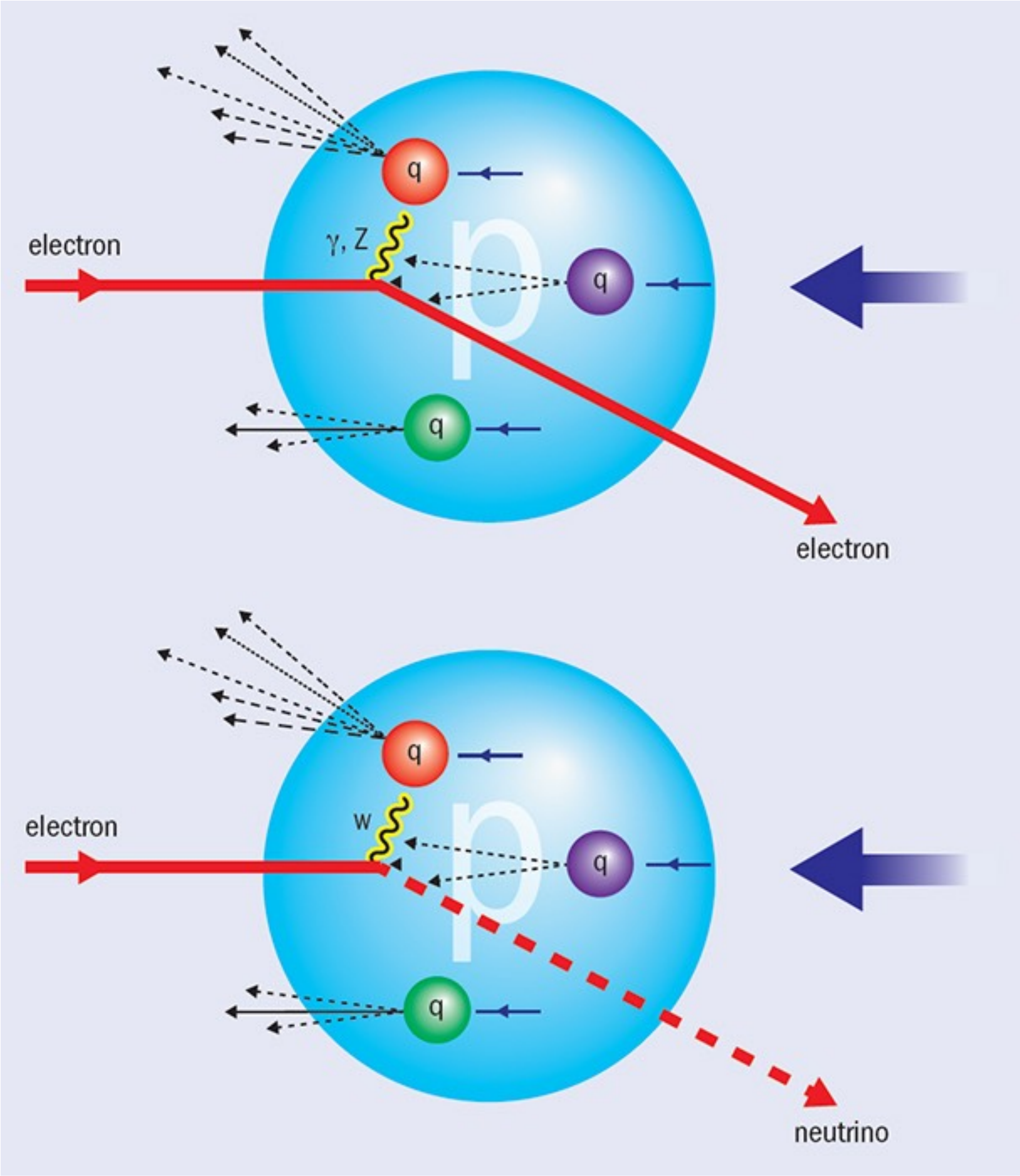
	Feature 1	Feature 2	Feature 3	Target
Example 1				
Example 2				
Example 3				
Example 4				

ENCODING

- Non-numeric data
- Class-based features:
  - One-hot encoding: 2 → [0 1]
  - When classes do not have sequential meaning: ✅ cars vs dogs vs plants ❌ months

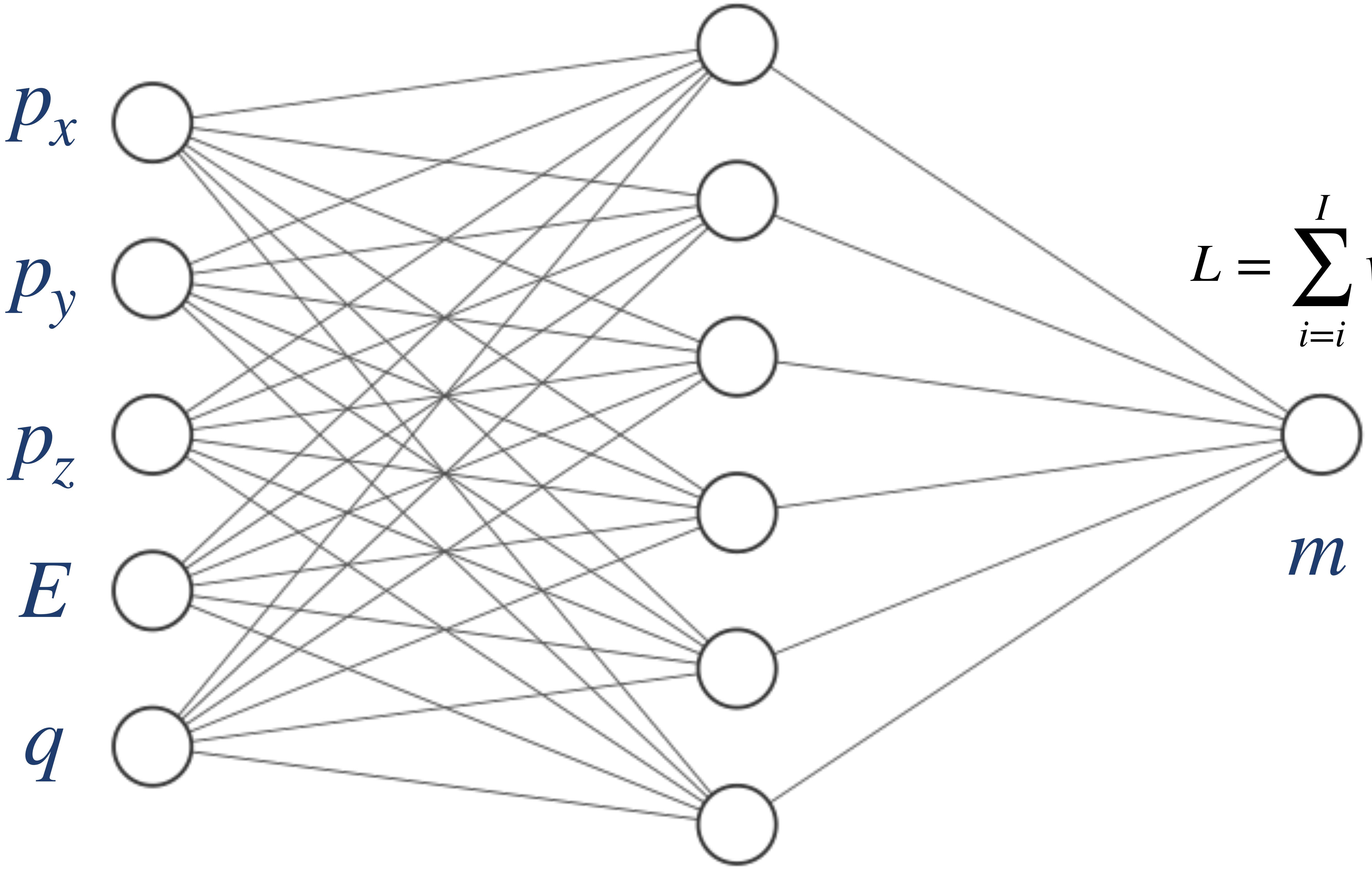
ACTIVITY DESCRIPTION

- Simulating e+p collisions
- Predicting particle-level invariant mass (regression)
- Advanced: try a generative model (e.g. autoencoders)



ACTIVITY DESCRIPTION

$$N = \frac{1}{1 + e^{-(\sum_{i=1}^{I} x_i w_i + b)}}$$



$$L = \sum_{i=i}^{I} w_i x_i + b$$

$$m^2 = E^2 - \|p\|^2$$

- Sigmoid activation for hidden layer and linear for output (regression model)
- How many “trainable parameters” in our model?

COMMUNITY

- Each of you arrived here with your own backgrounds, specialty, and path in life
- Your experience and expertise are valuable here, no matter what it is
- If the activity is within your background, help others!
- If you are totally (or a little) lost, ask for help!
- It is our shared goal to have **each** of us leave with some new skill/knowledge/understanding

GETTING STARTED

- Navigate to https://github.com/alpha-davidson/ICTP-Citizen-Science-2023
- If you have access to a google login, click "open in collar"
- Otherwise, download and open in Deepnote or download onto your personal computer (with appropriate dependencies)