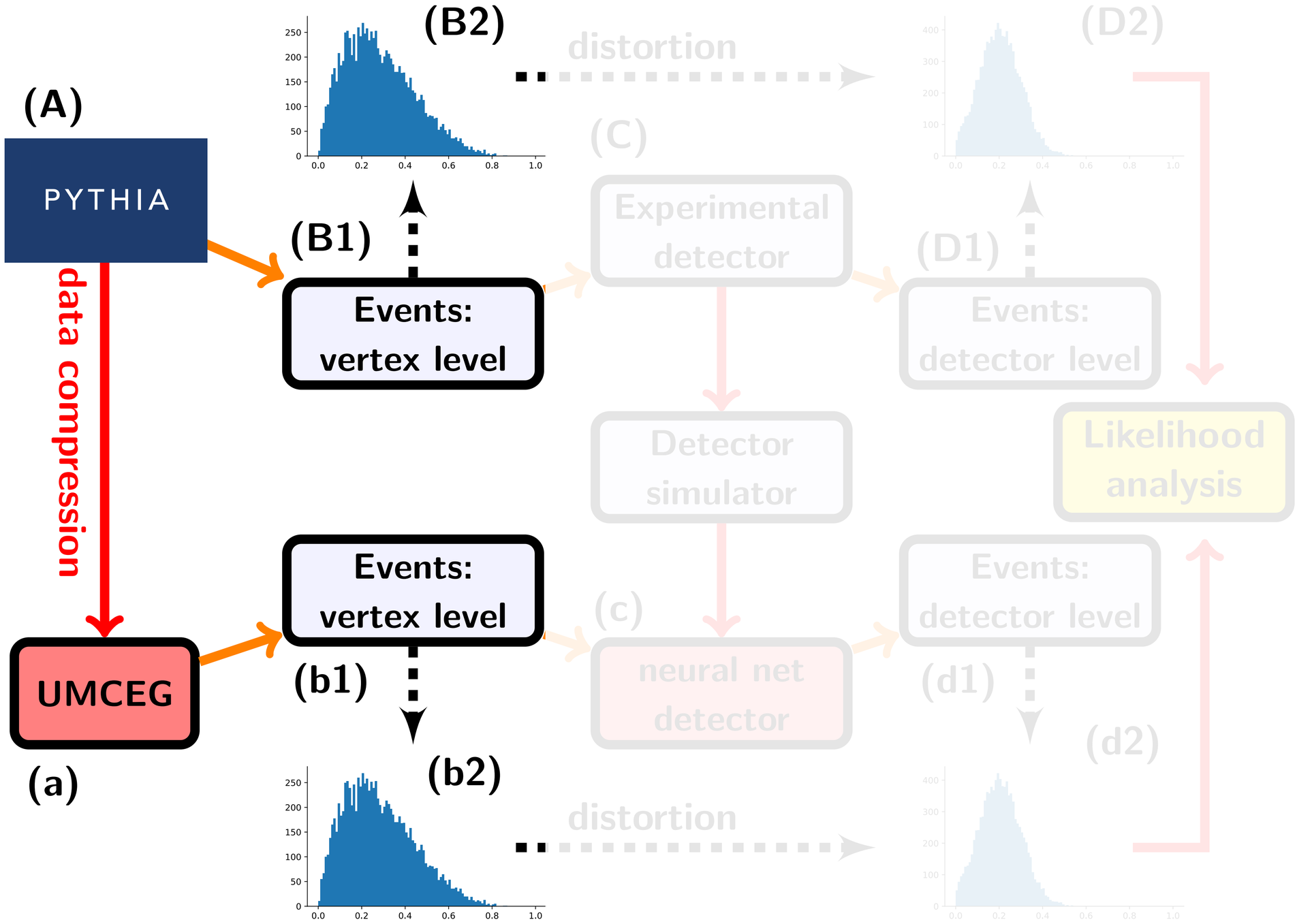
MACHINE LEARNING FOR EVENT SIMULATION AND CLASSIFICATION IN THE ACTIVE-TARGET TIME PROJECTION CHAMBER

MICHELLE KUCHERA
DAVIDSON COLLEGE
JEFFERSON LAB SEMINAR
8 JANUARY 19

JLAB THEORY TALK???

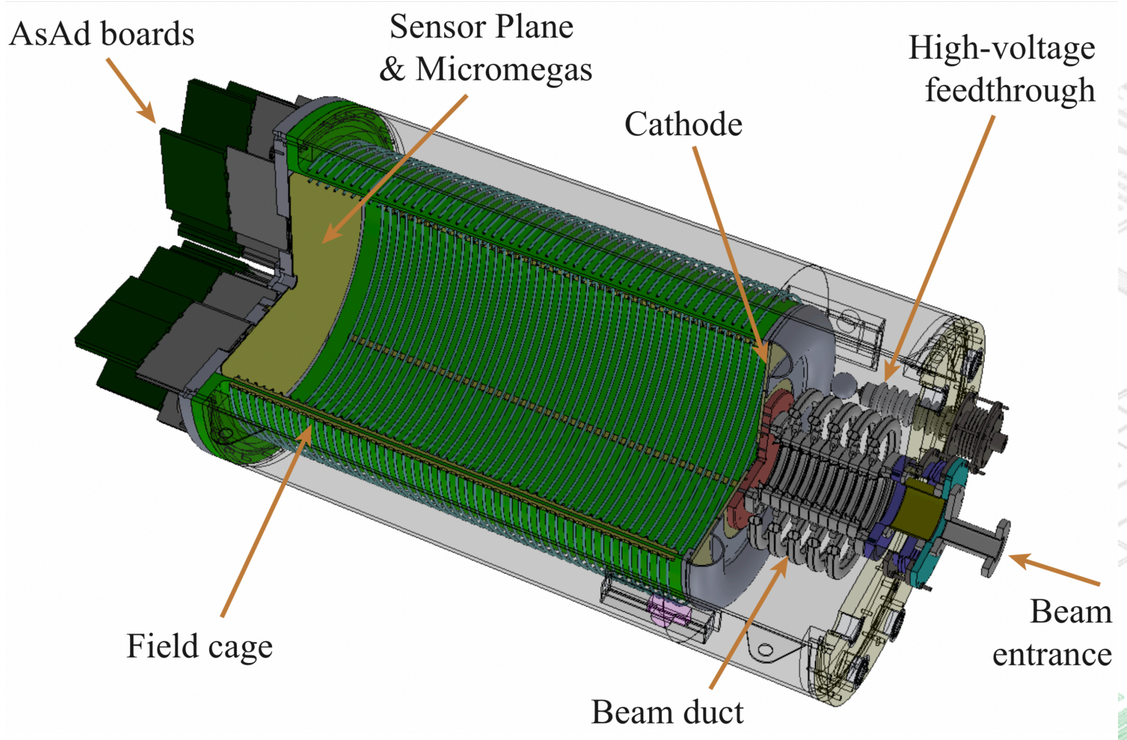
UMCEG
EIC


(A)
PYTHIA
data compression
UMCEG
(a)
(B1)
Events:
vertex level
(B2)
(b1)
Events:
vertex level
(b2)
distortion
distortion
(C)
Experimental
detector
Detector
simulator
(c)
neural net
detector
(D1)
Events:
detector level
(D2)
(d1)
Events:
detector level
(d2)
Likelihood
analysis

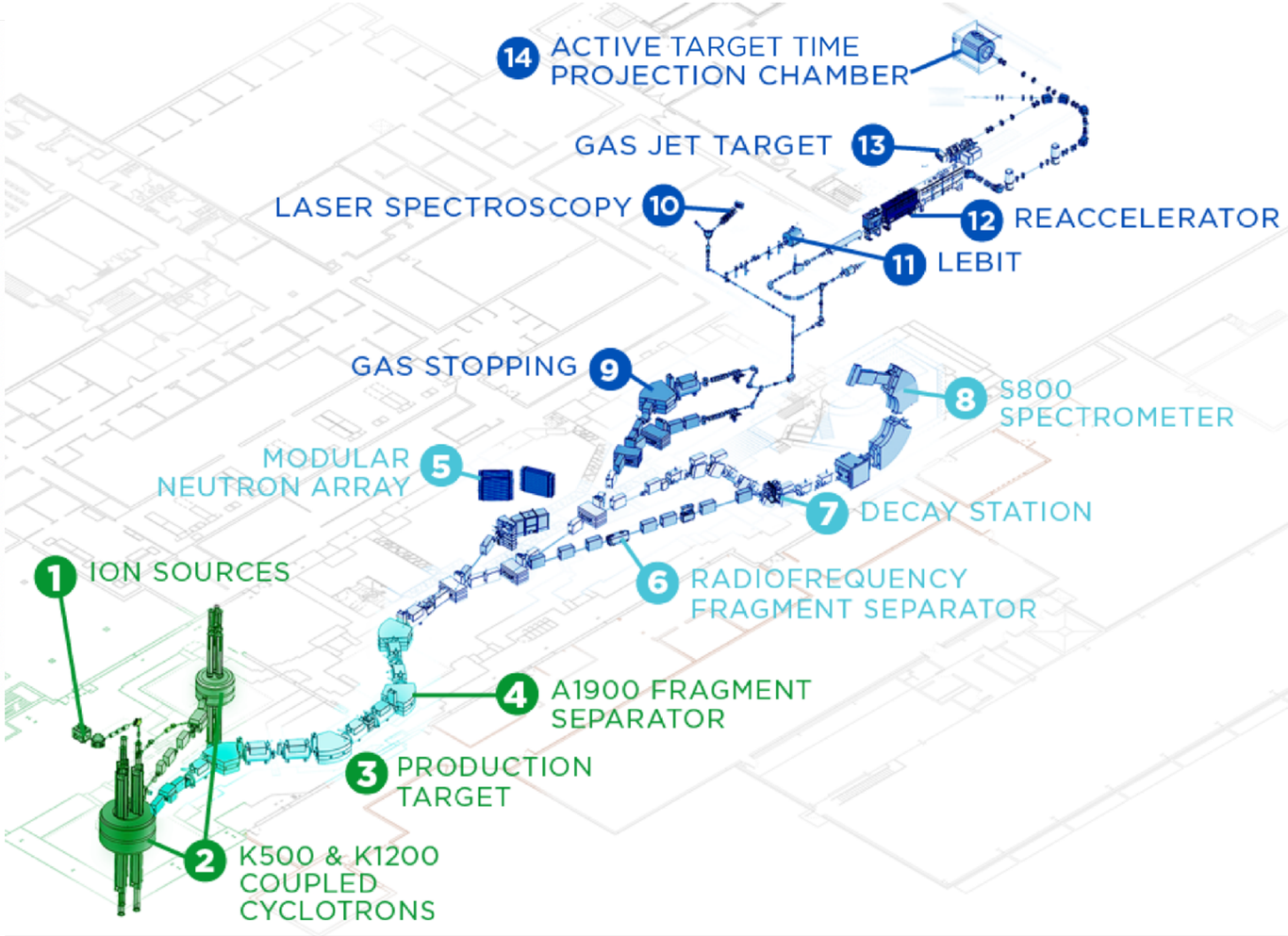
CHALLENGES IN DATA ANALYSIS

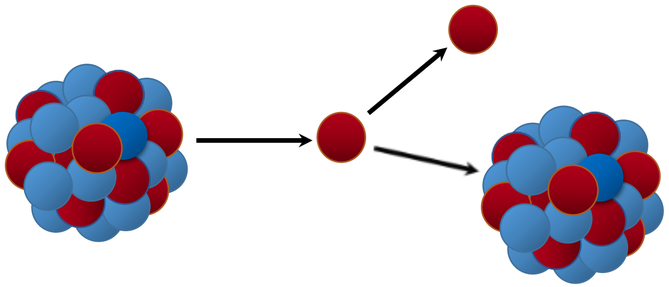
WITH THE ACTIVE-TARGET TIME PROJECTION CHAMBER (AT-TPC)

ACTIVE TARGET - TIME PROJECTION CHAMBER (AT-TPC)



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

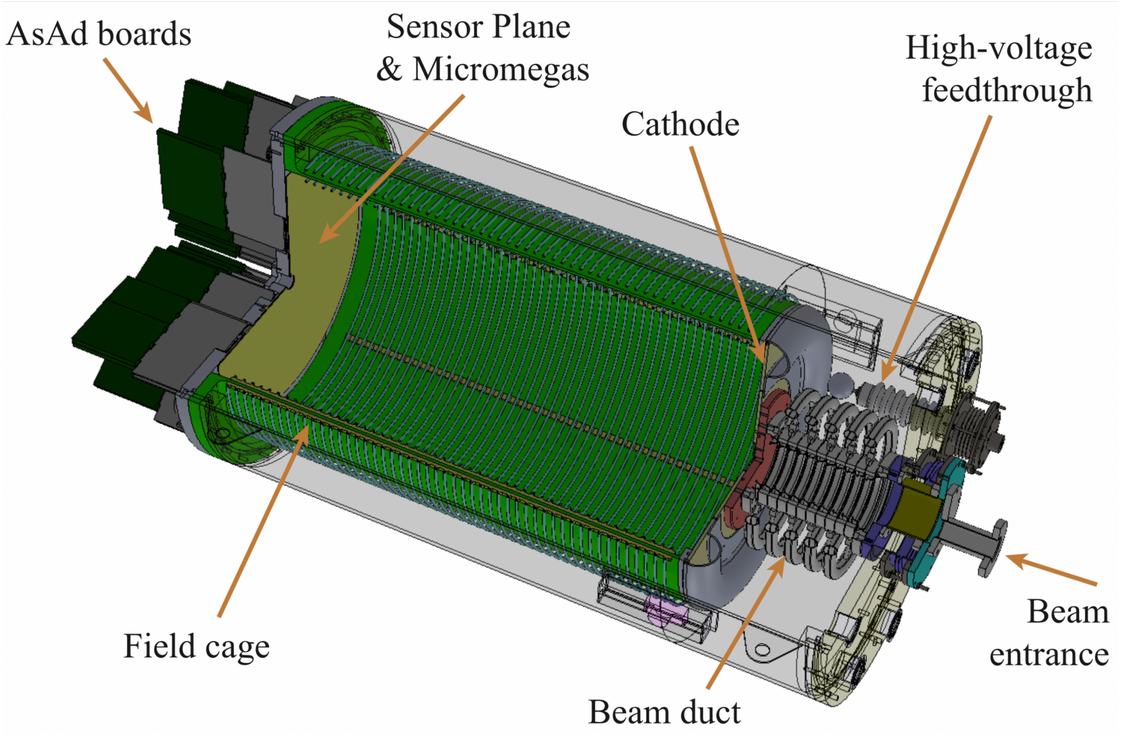




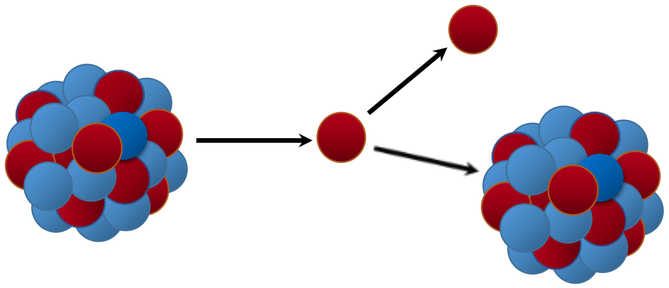
J. Z. TAYLOR, HONOR'S THESIS, DAVIDSON COLLEGE

ERIN O'DONNELL, NSCL

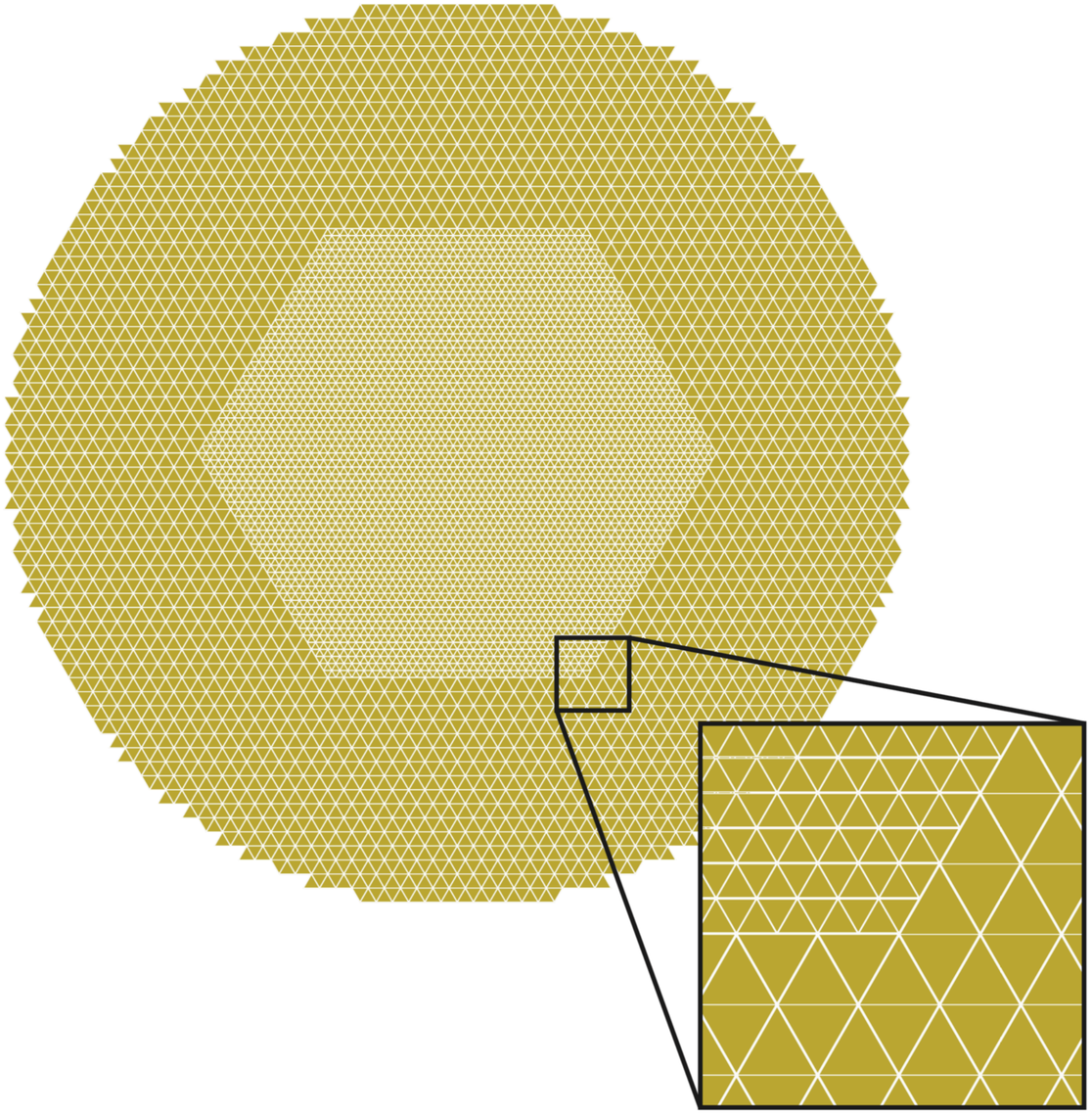
ACTIVE TARGET - TIME PROJECTION CHAMBER (AT-TPC)



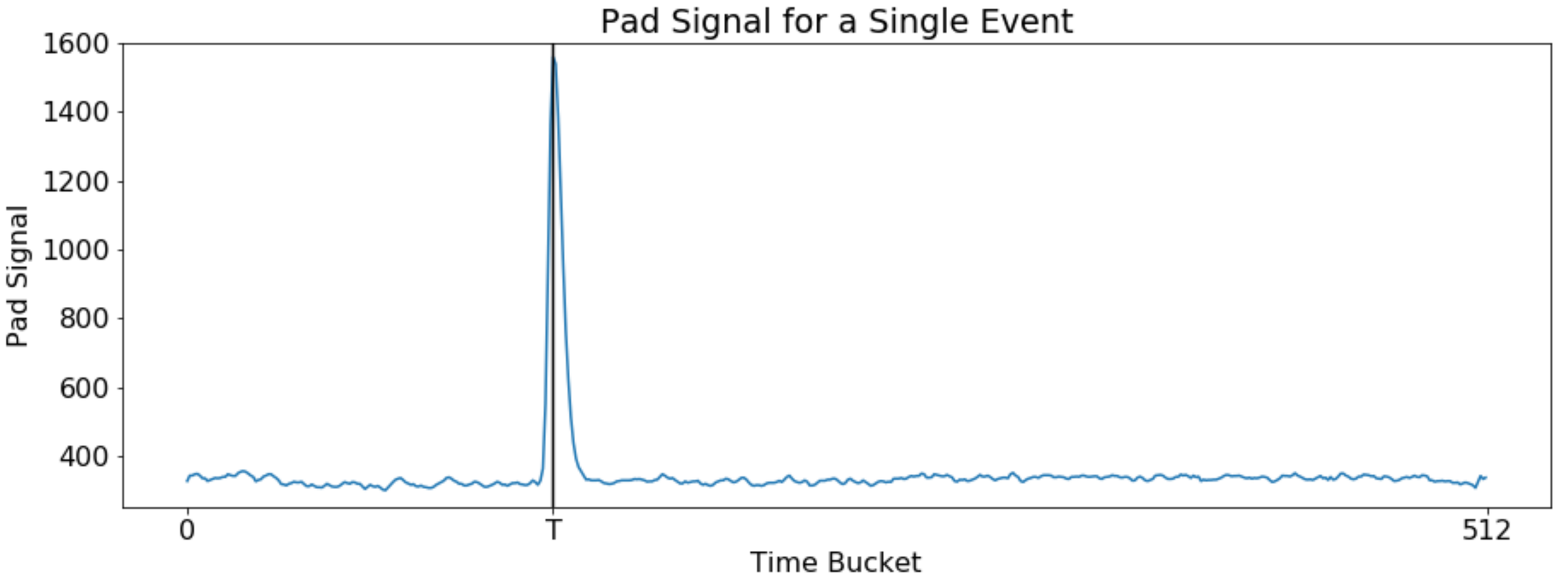
J. BRADT ET. AL., *NUCLEAR INSTRUMENTS AND METHODS*, 2017.

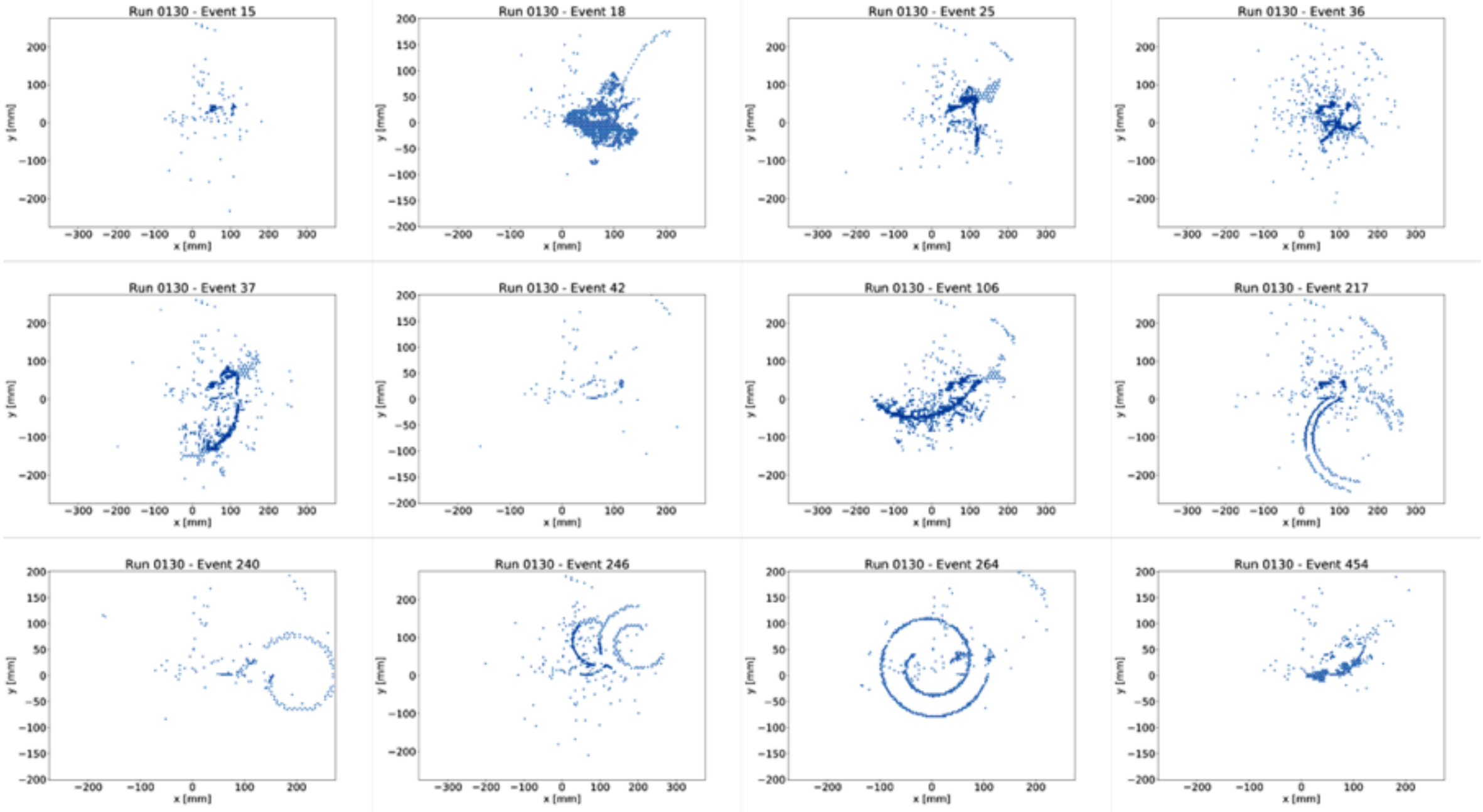


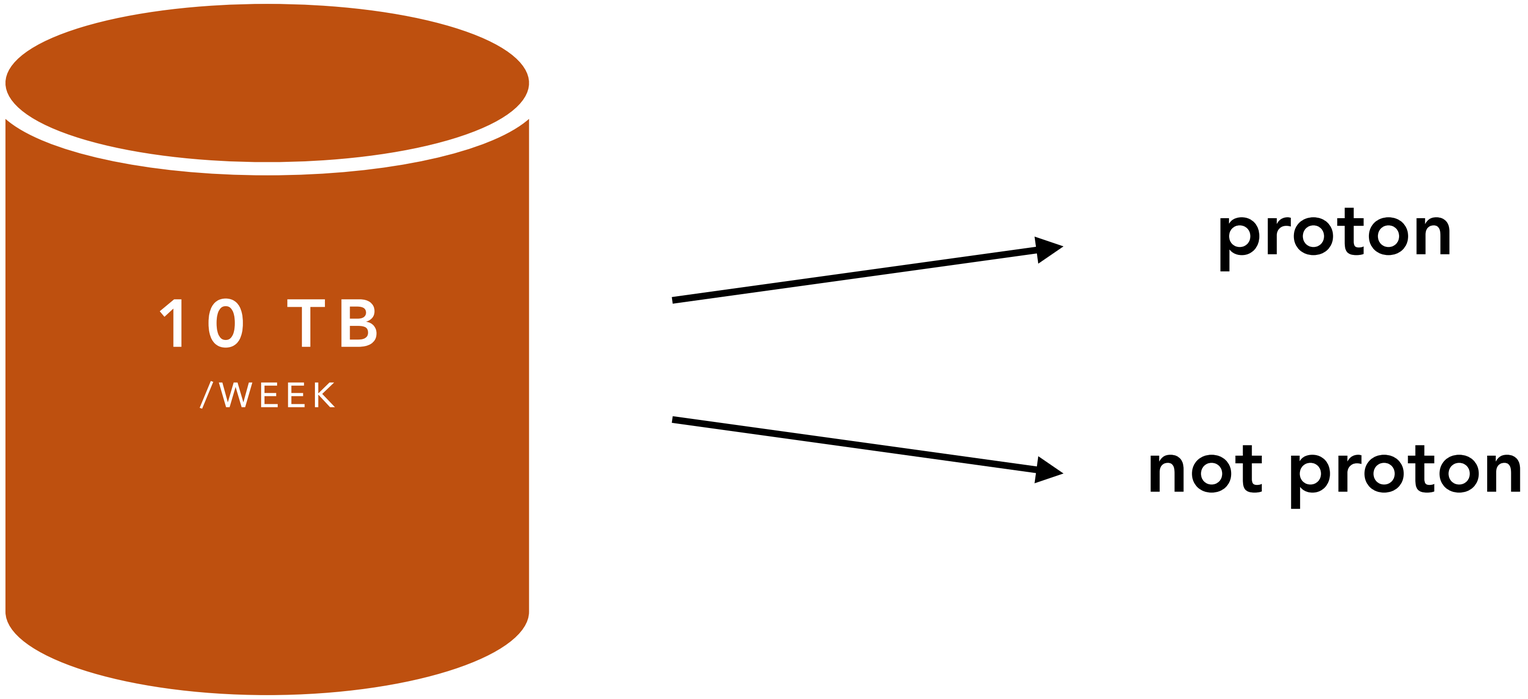
J. Z. TAYLOR, HONOR'S THESIS, DAVIDSON COLLEGE



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

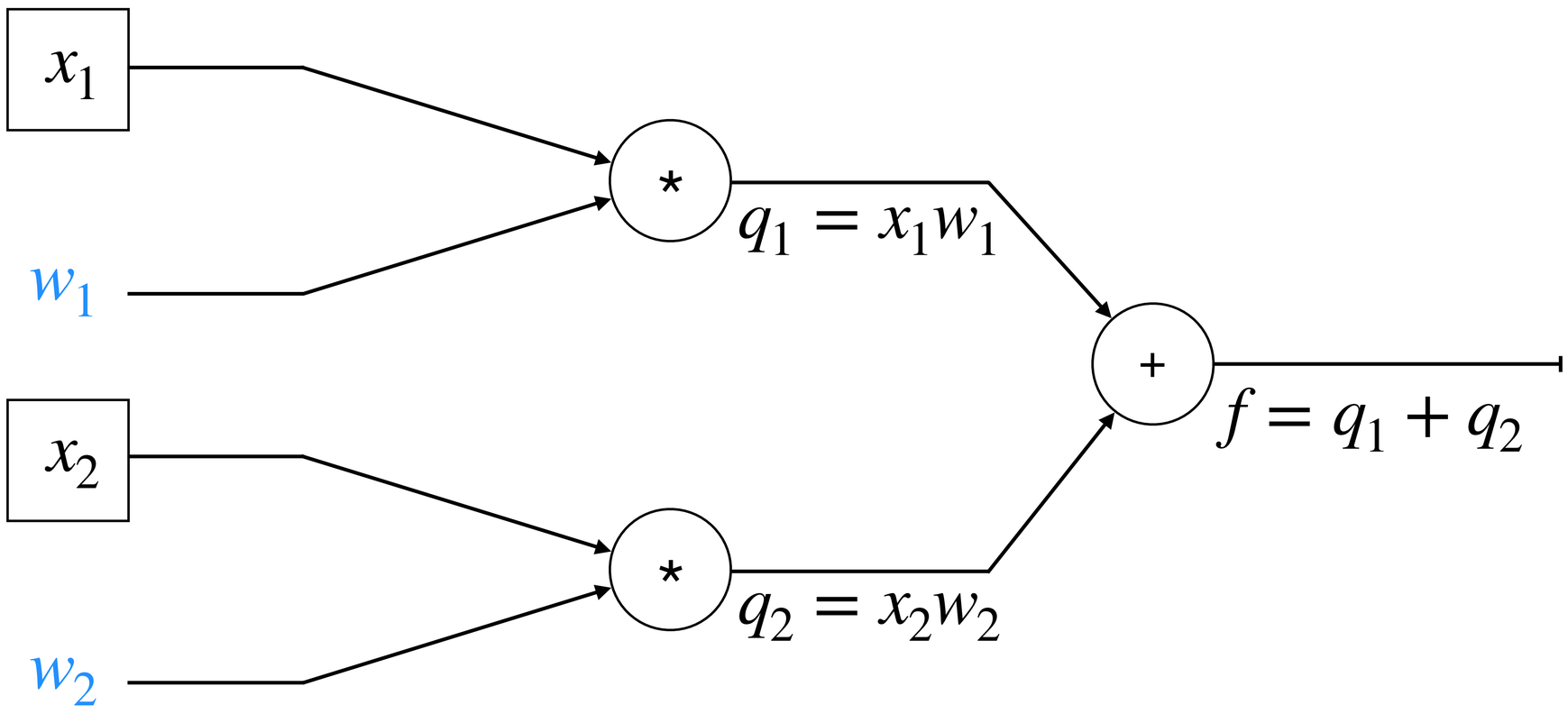

Pad Signal for a Single Event
1600
1400
1200
1000
800
600
400
Pad Signal
0
T
512
Time Bucket


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]

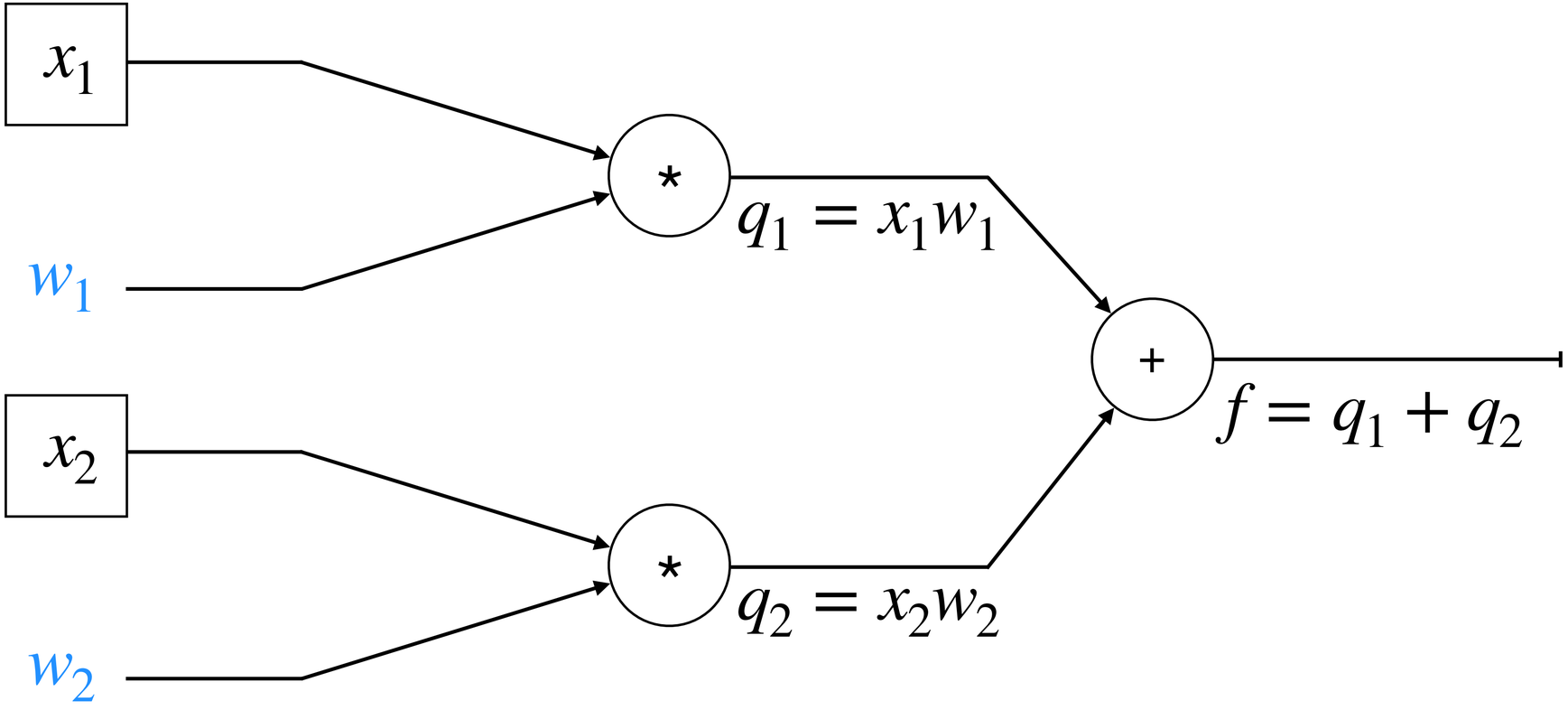

10 TB
/WEEK
proton
not proton

BACKGROUND OF NEURAL NETWORK METHODS

NETWORK GRAPH



SUPERVISED LEARNING



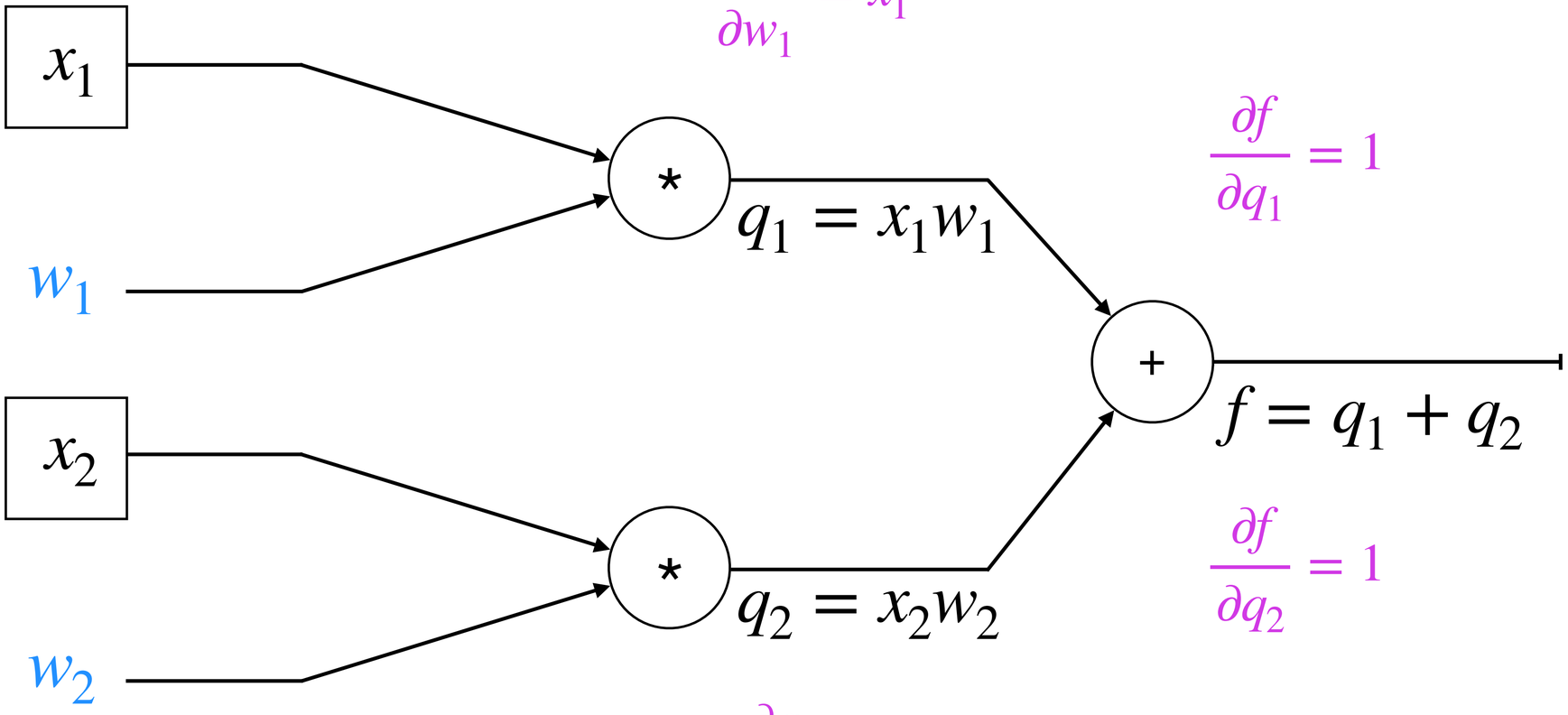
Loss function

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

BACKPROPAGATION

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

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



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

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

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

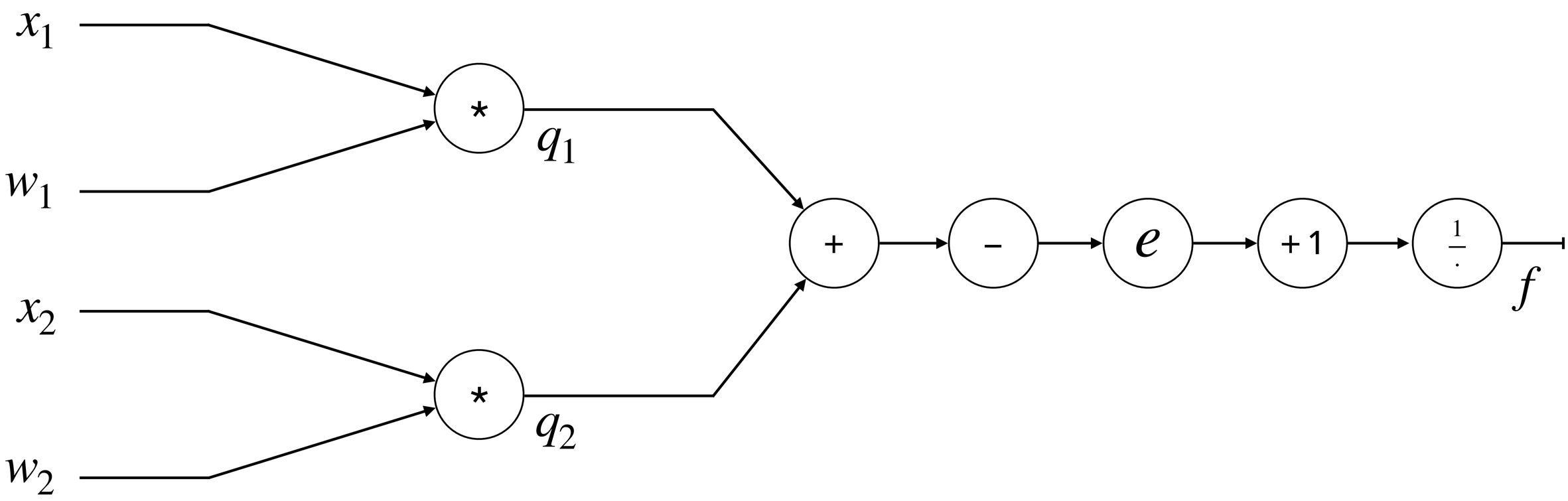
$$\frac{\partial q_2}{\partial w_2} = x_2$$

$$w_2 = w_2 + \eta * \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial w_2}$$

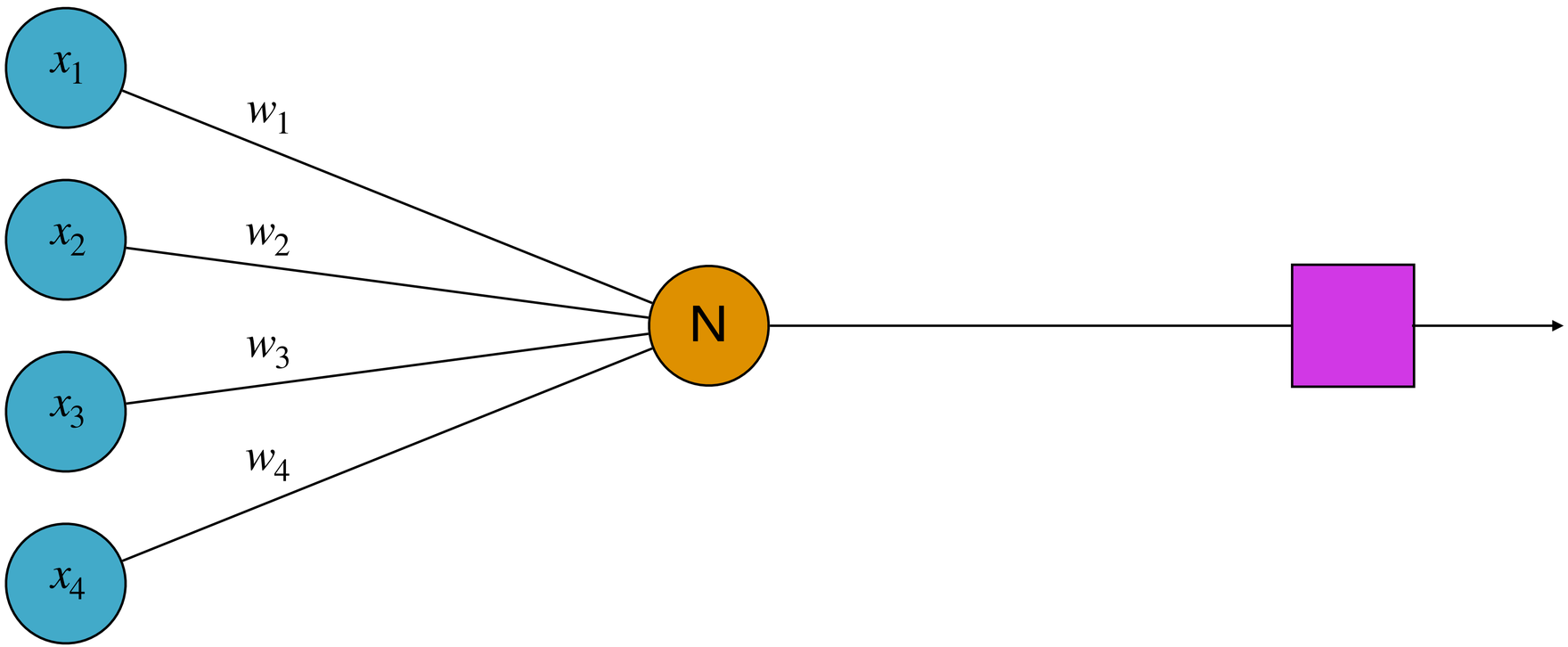
Loss function

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

LOGISTIC REGRESSION



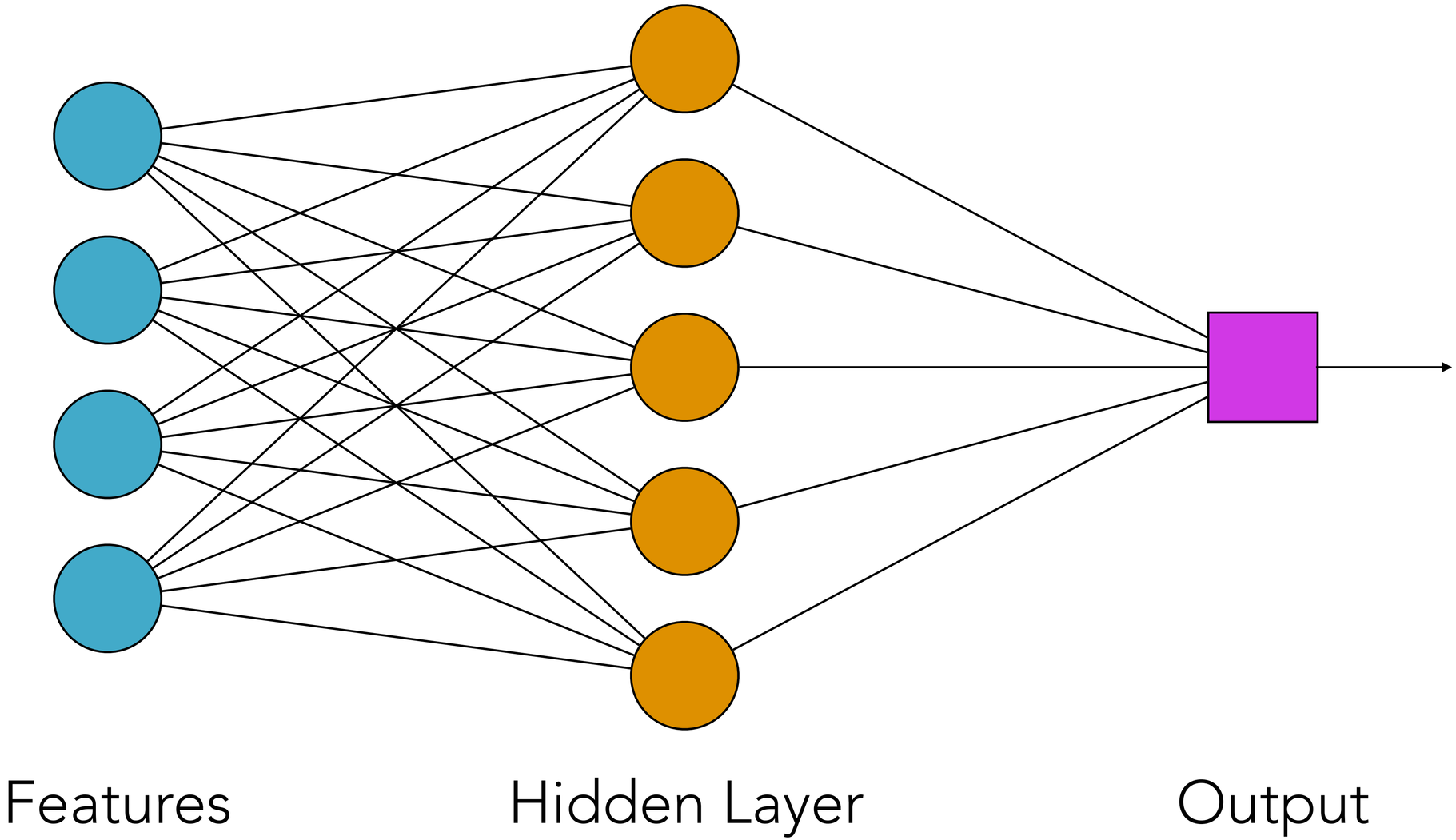
$$f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}}$$



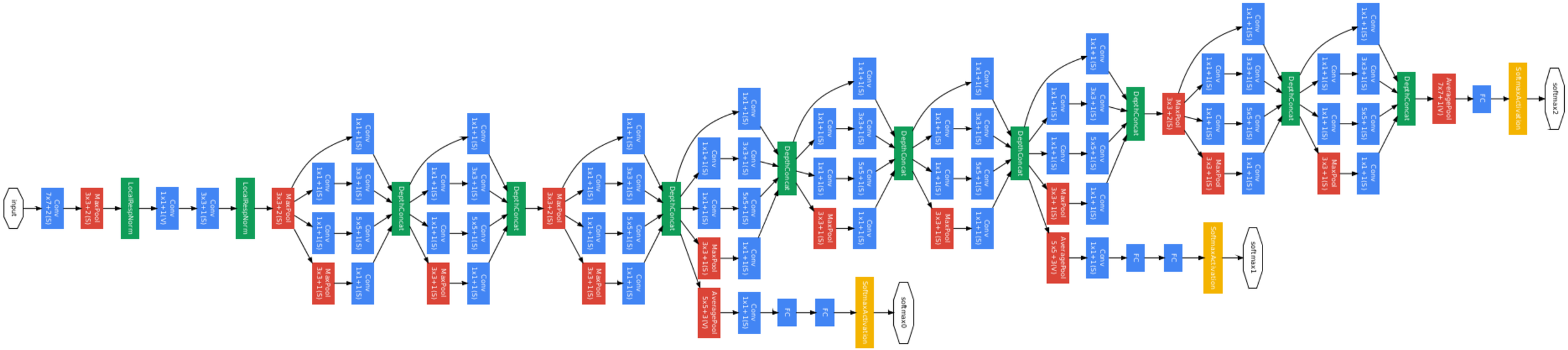
Features

Summation
+ Nonlinearity

Output


Features
Hidden Layer
Output

“GoogLeNet network with all the bells and whistles”



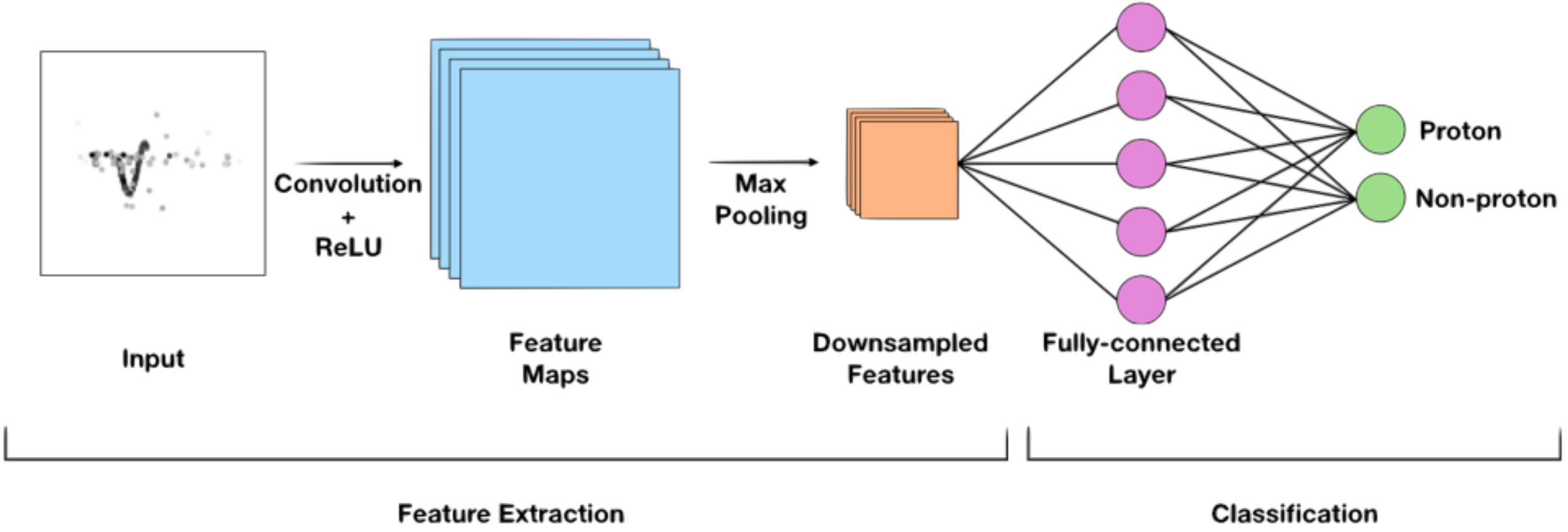
CHRISTIAN SZEGEDY ET. AL. GOING DEEPER WITH CONVOLUTIONS.

APPLYING DEEP LEARNING TO SOLVE AT-TPC CHALLENGES

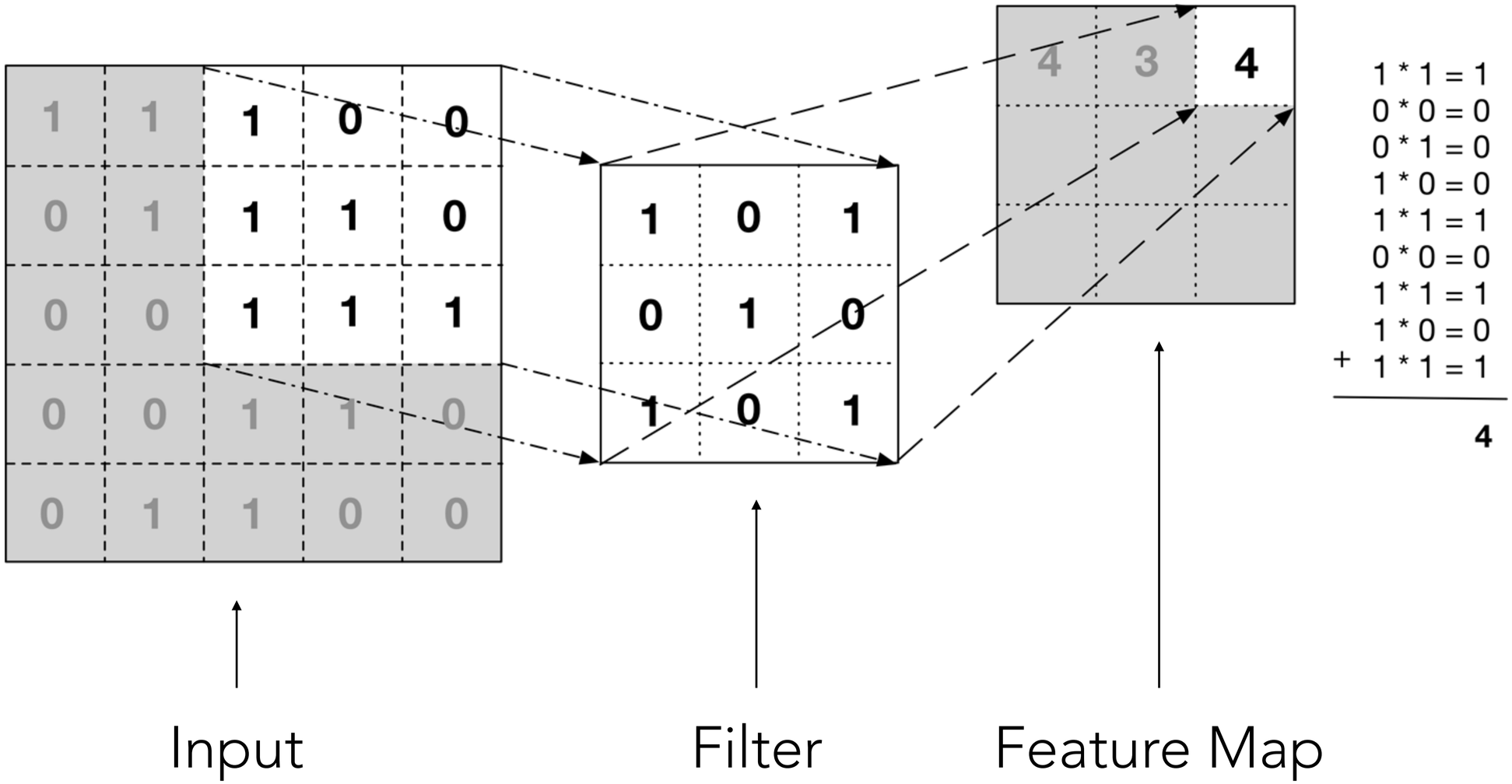
CLASSIFICATION
SIMULATION

CONVOLUTIONAL NEURAL NETWORKS

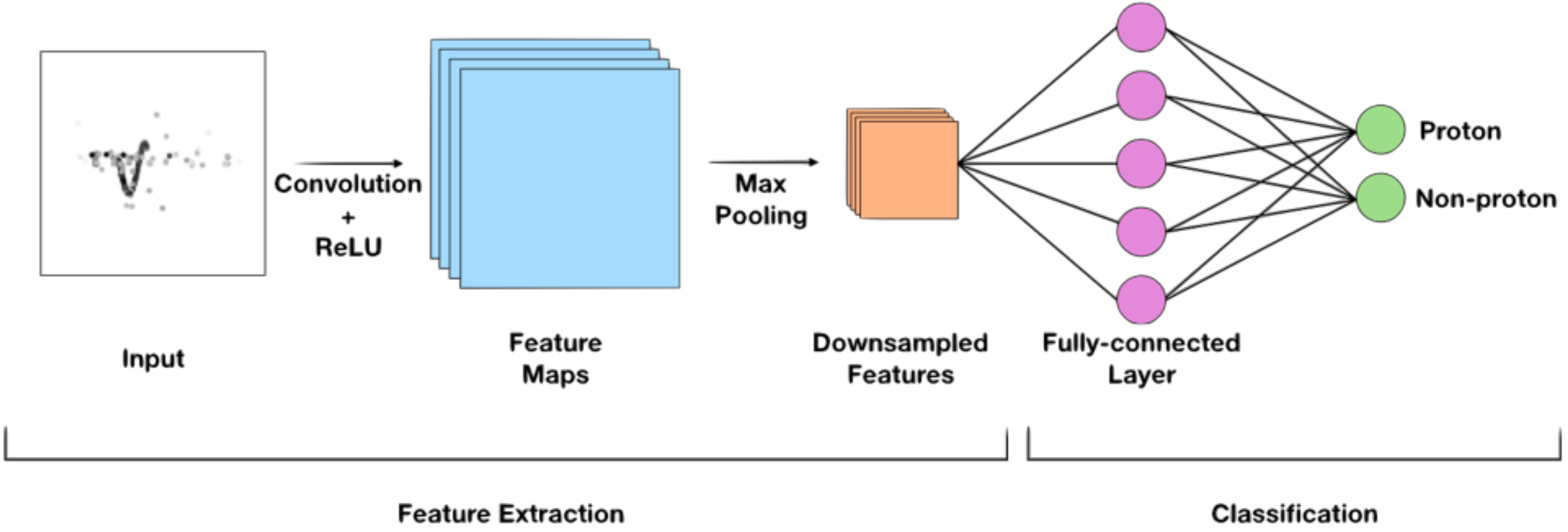
CLASSIFICATION


Convolution
+
ReLU
Max
Pooling
Proton
Non-proton
Input
Feature
Maps
Downsampled
Features
Fully-connected
Layer
Feature Extraction
Classification

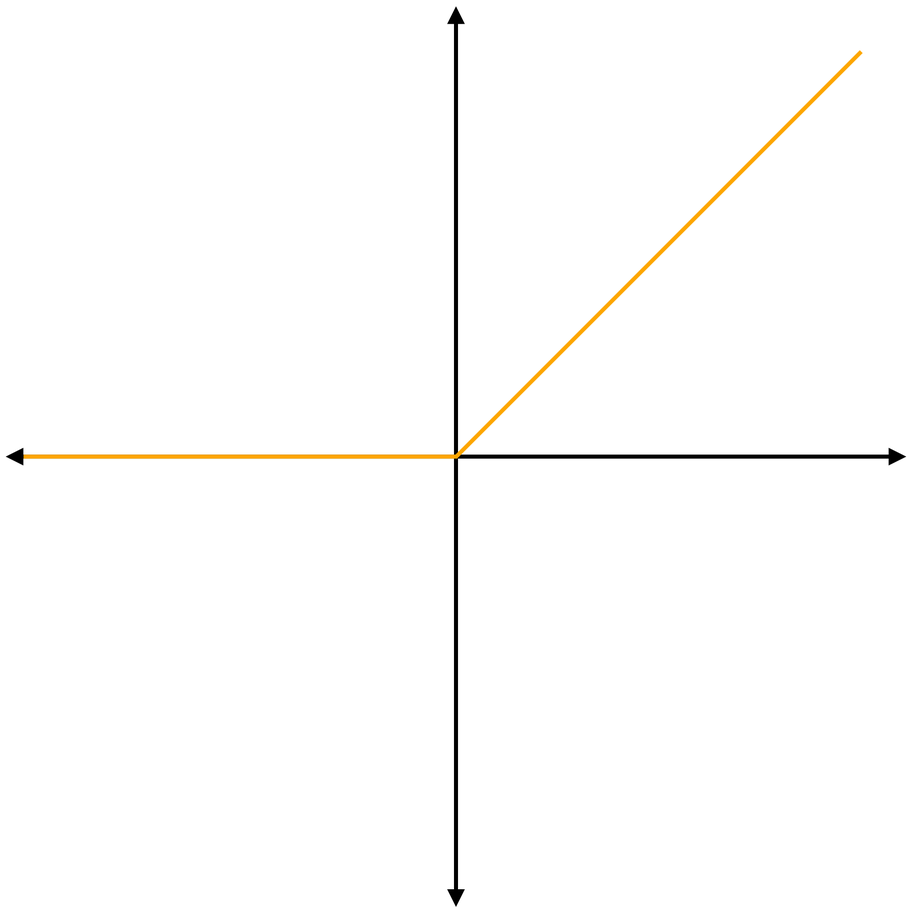
DISCRETE CONVOLUTION



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


Convolution
+
ReLU
Max
Pooling
Proton
Non-proton
Input
Feature
Maps
Downsampled
Features
Fully-connected
Layer
Feature Extraction
Classification

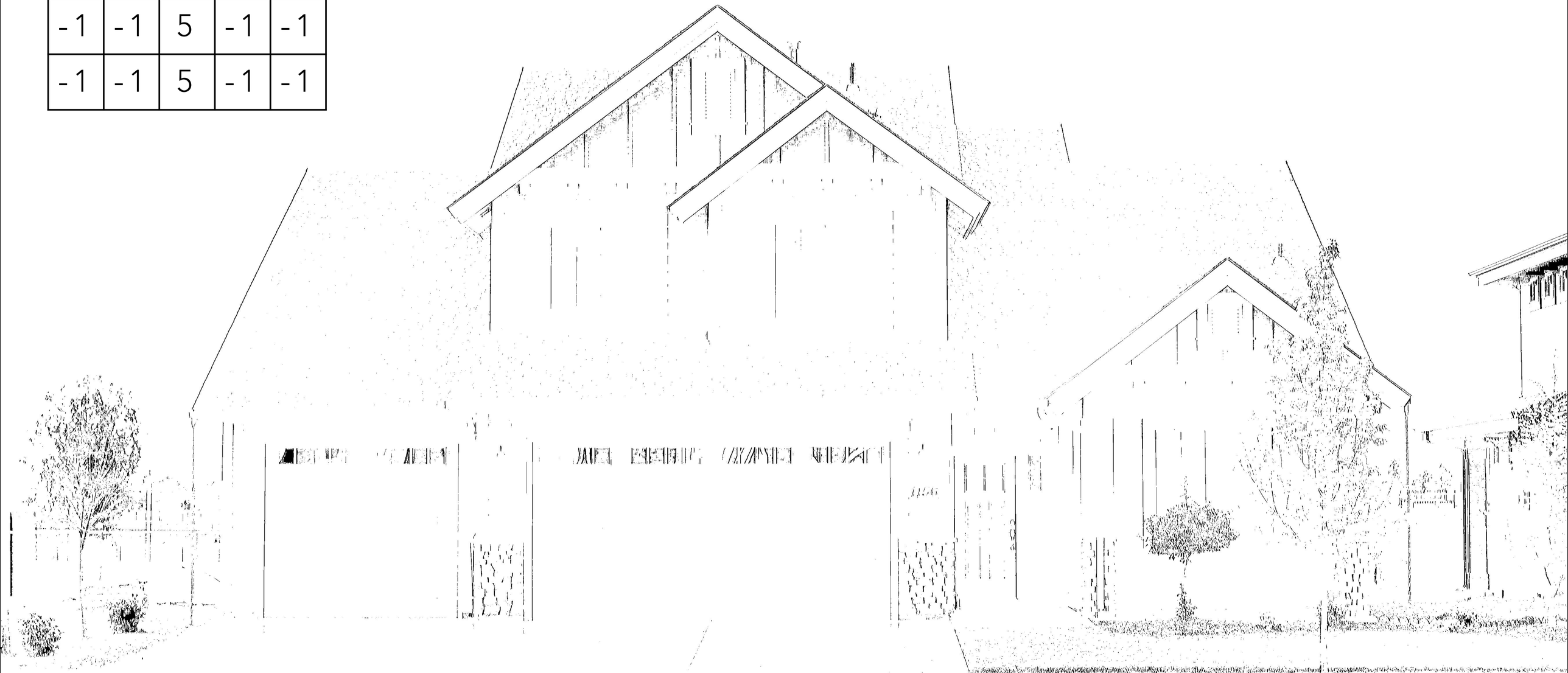
RECTIFIED LINEAR UNIT (ReLU)

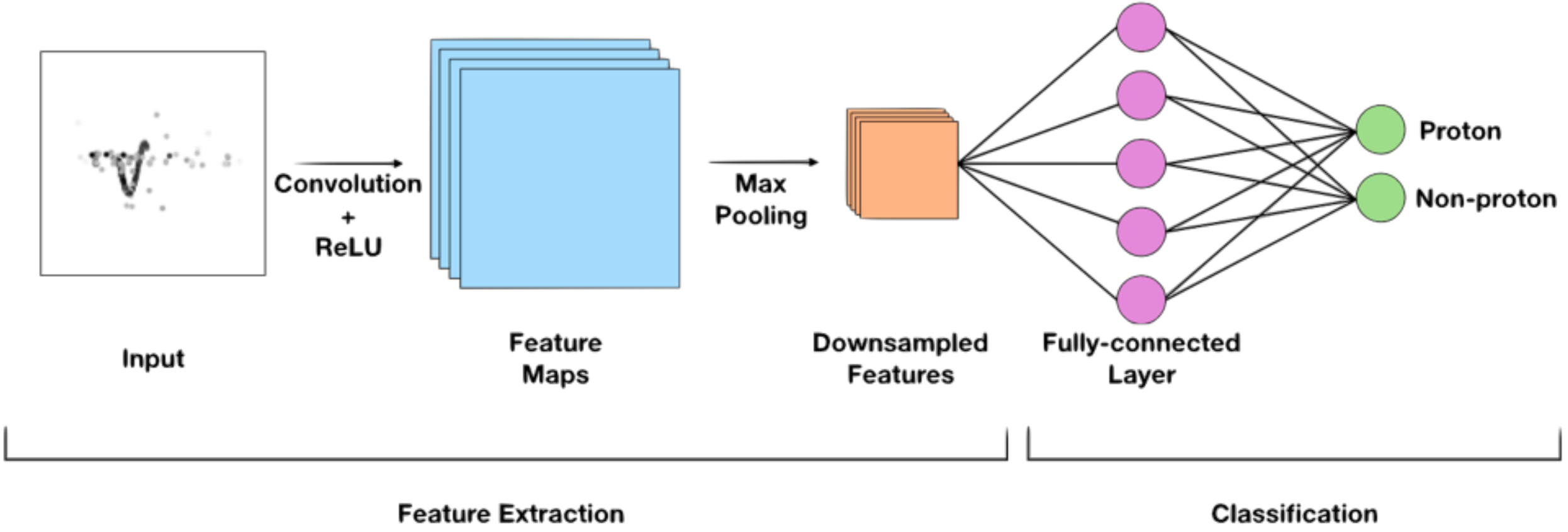



1156

-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



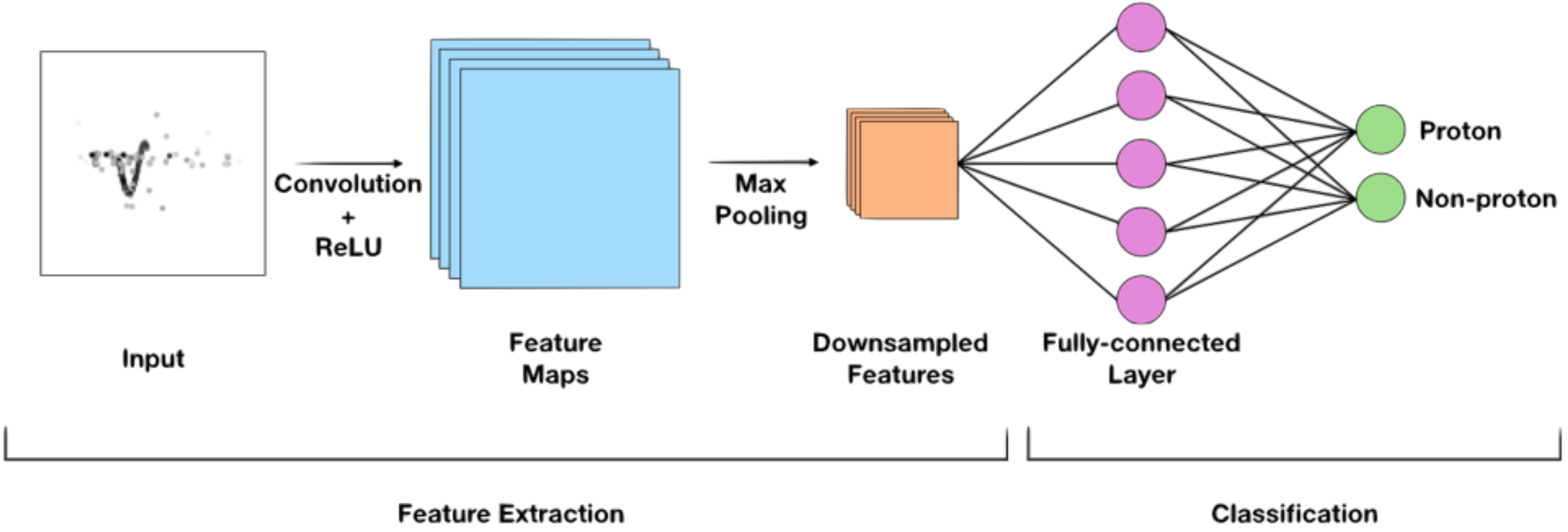

Convolution
+
ReLU
Max
Pooling
Proton
Non-proton
Input
Feature
Maps
Downsampled
Features
Fully-connected
Layer
Feature Extraction
Classification

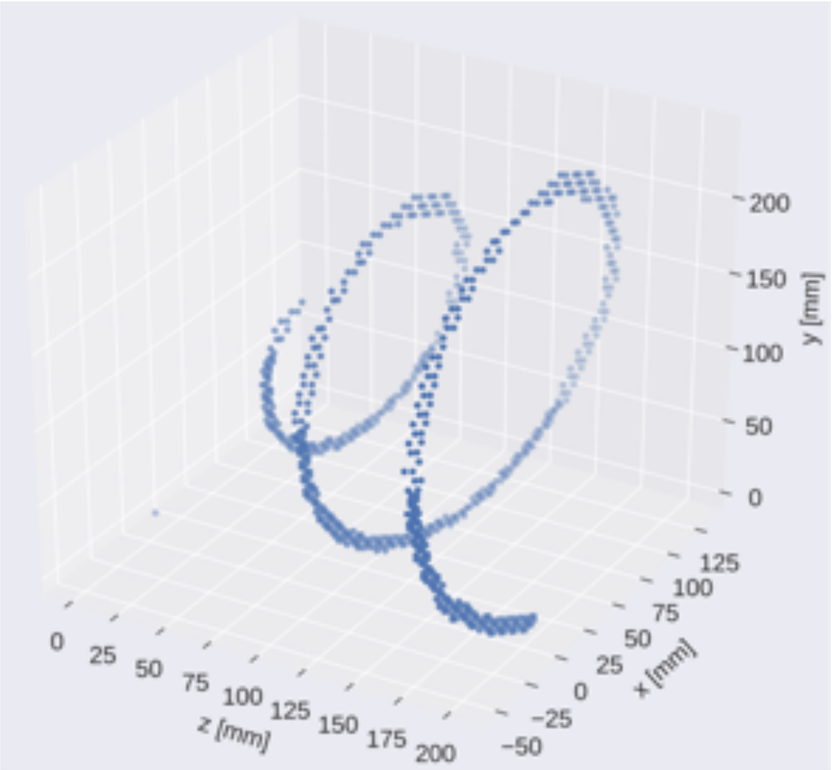
MAX POOLING

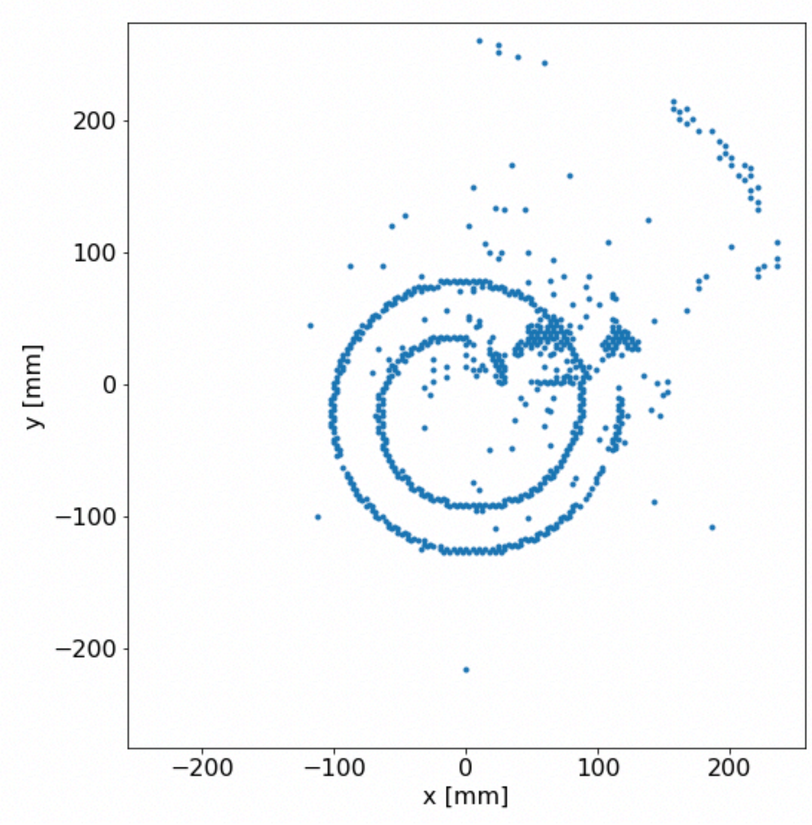
1	1	2	4
5	6	9	3
3	2	4	4
1	2	0	7

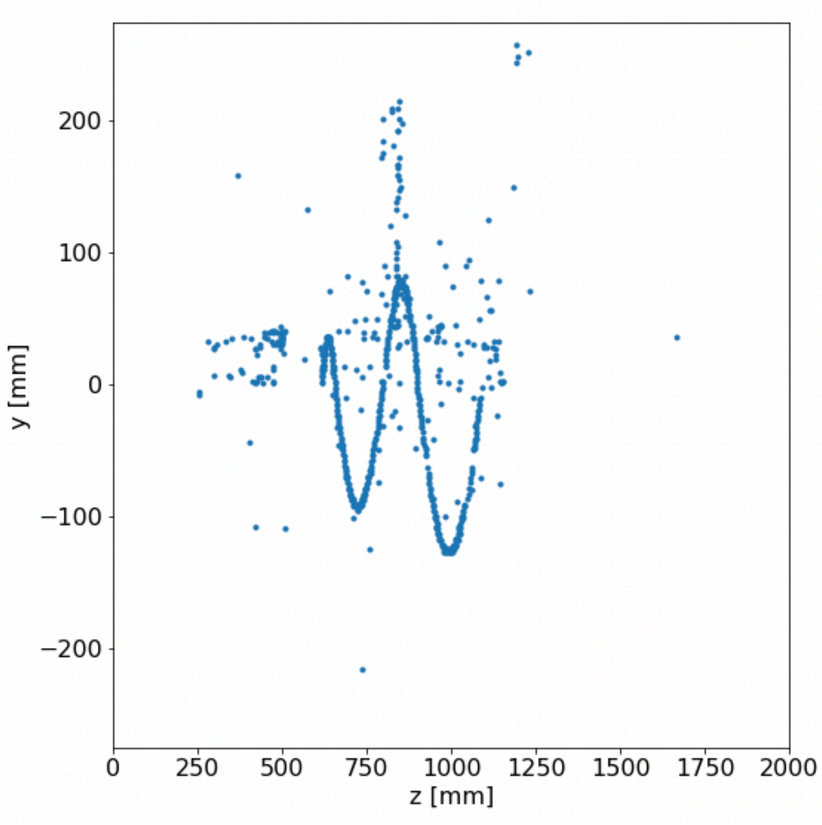
max pool with 2x2 filters
and stride 2

6	9
3	7


Convolution
+
ReLU
Max
Pooling
Proton
Non-proton
Input
Feature
Maps
Downsampled
Features
Fully-connected
Layer
Feature Extraction
Classification



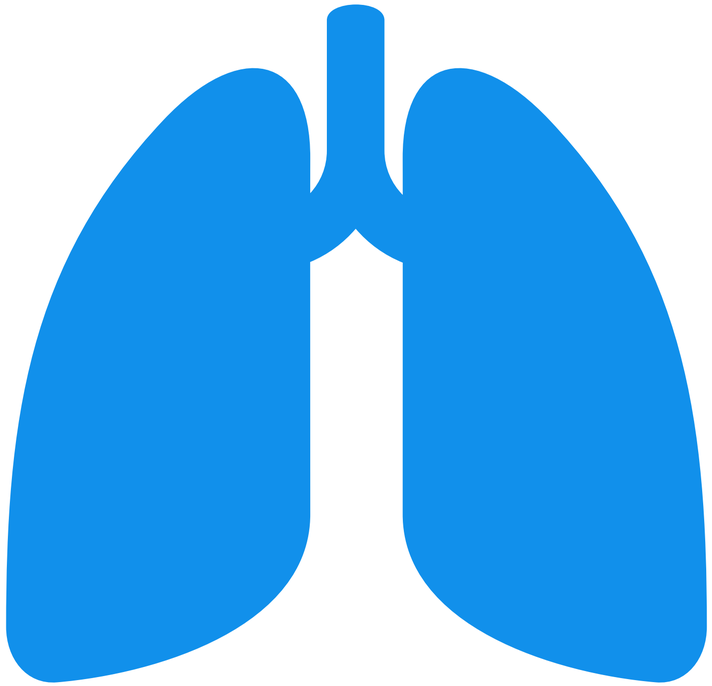




J. TAYLOR, *ML METHODS FOR EVENT CLASSIFICATION*, 2018.

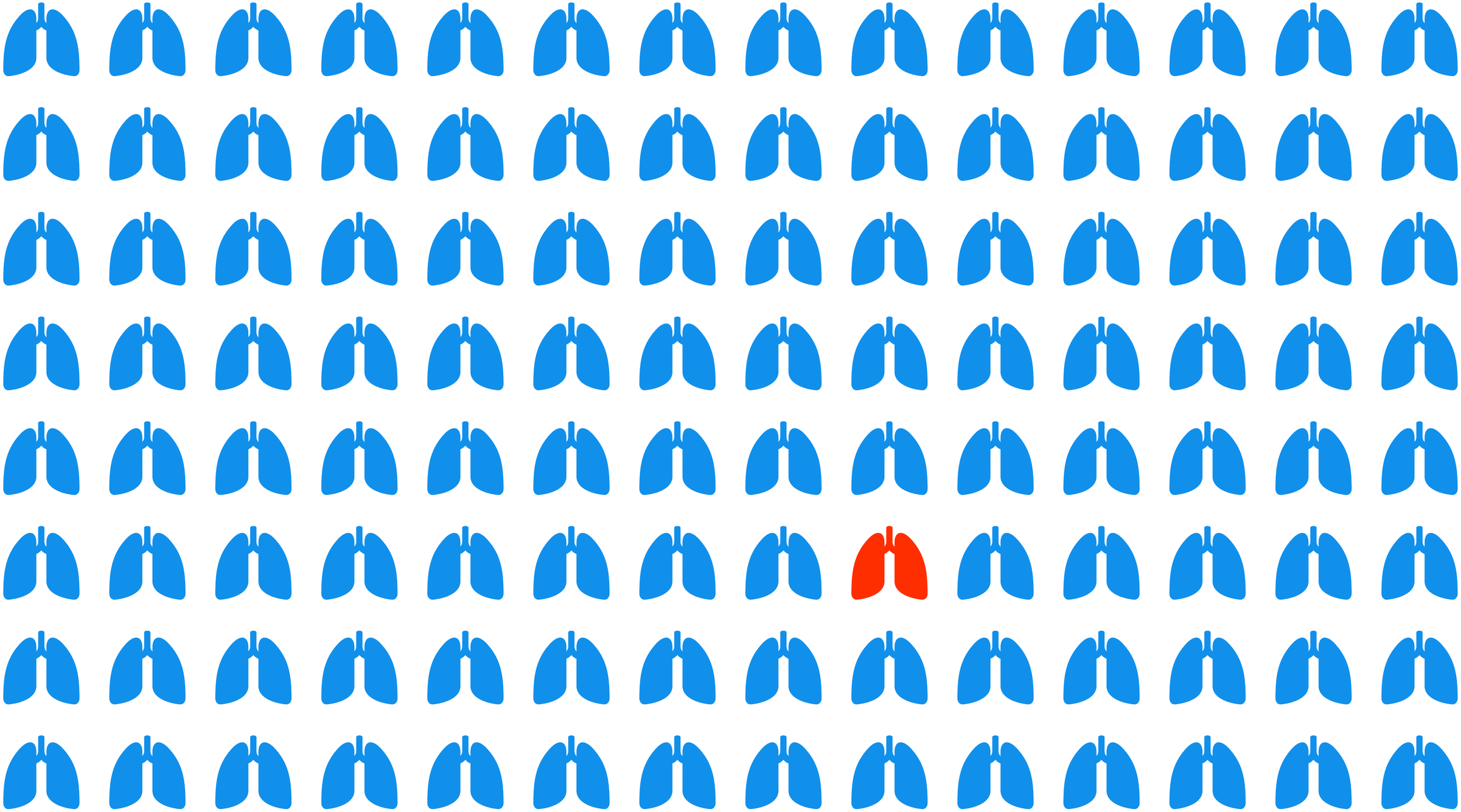
Can we use machine learning to **accurately** classify proton events from the AT-TPC?

Metrics



Detect Lung Cancer

99% Accuracy



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

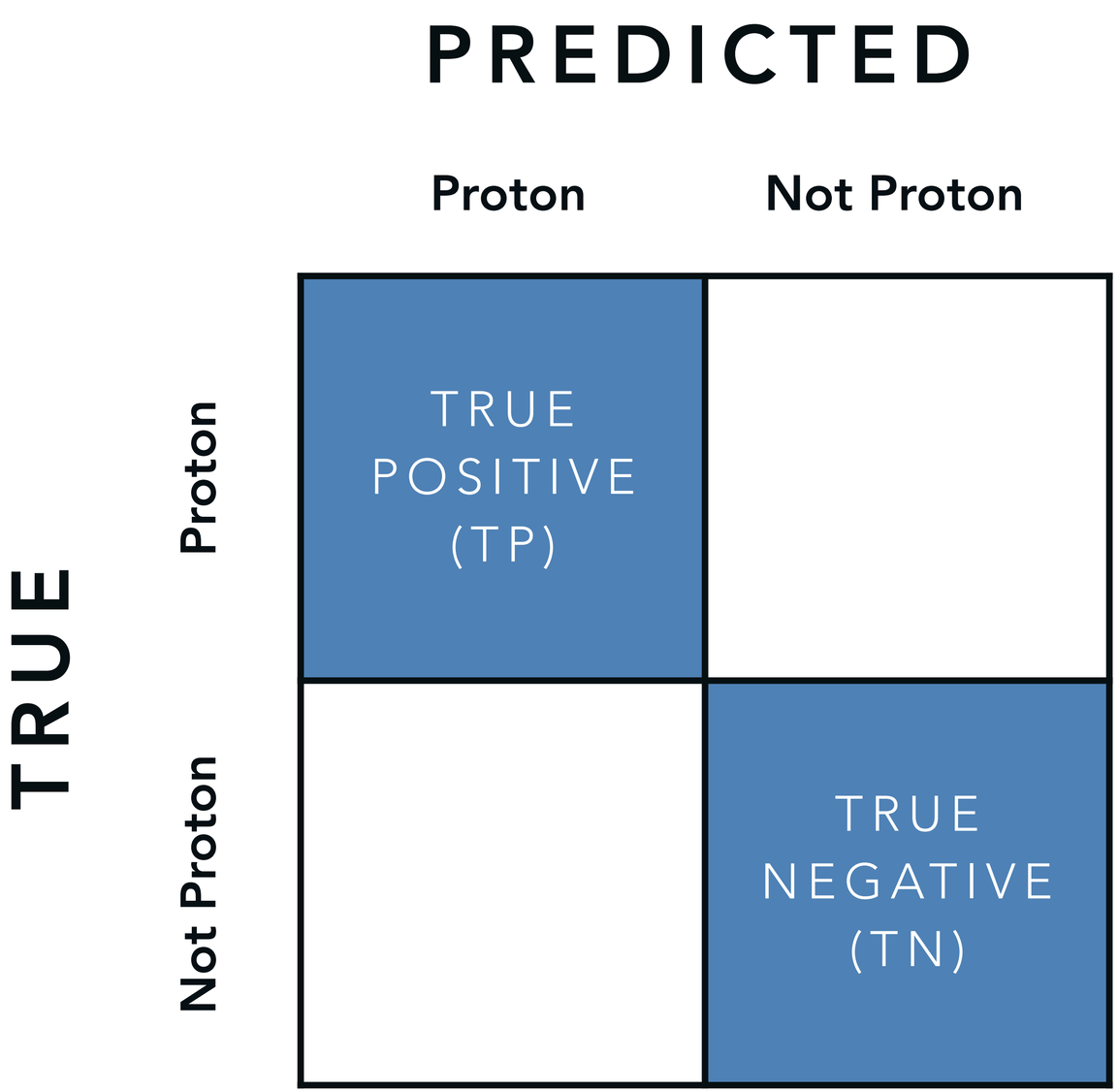
TRUE \ PREDICTED	Proton	Not Proton
Proton	TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
Not Proton	FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

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

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

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

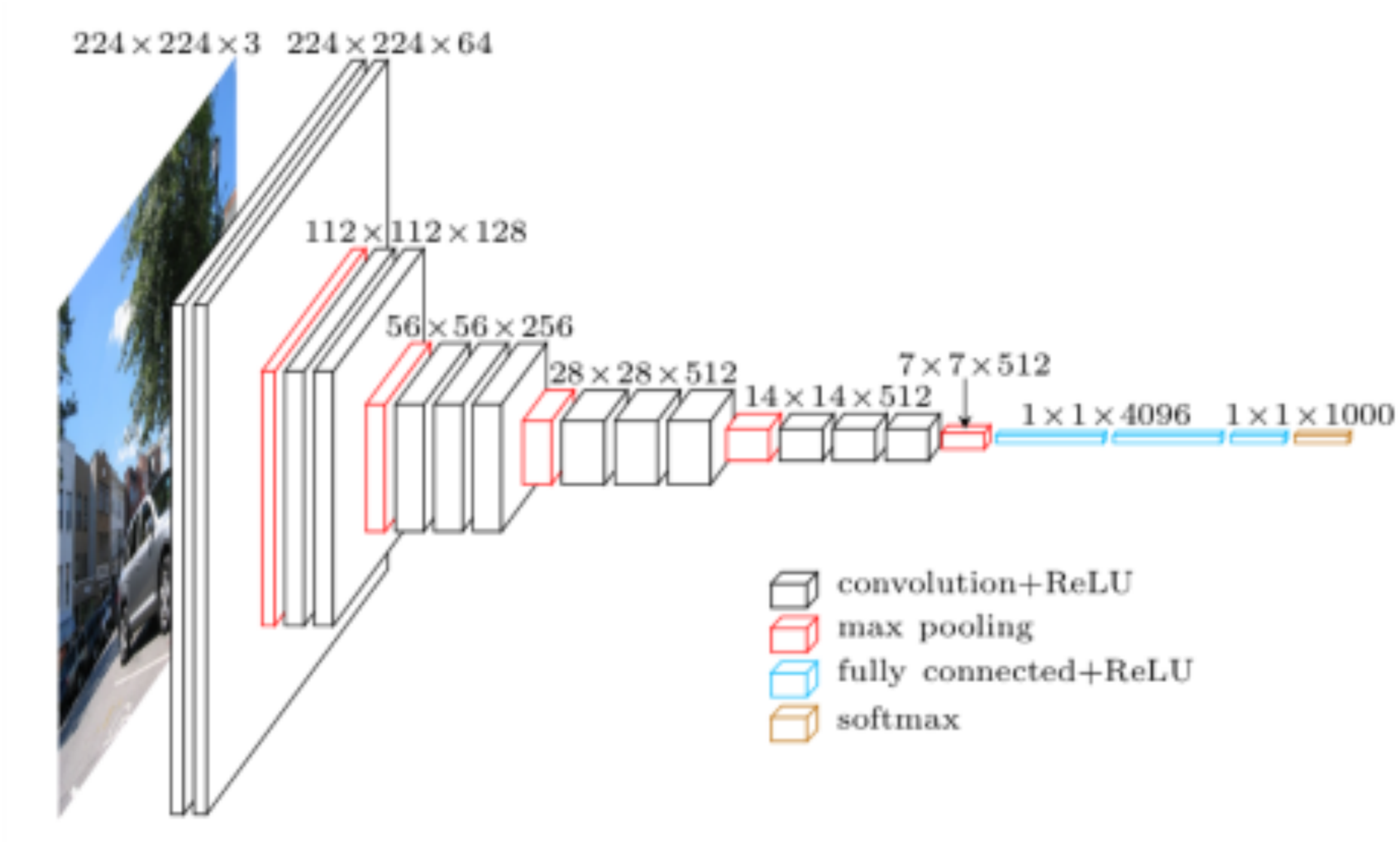
$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



PERFECT MODEL

EXPERIMENTAL DATA

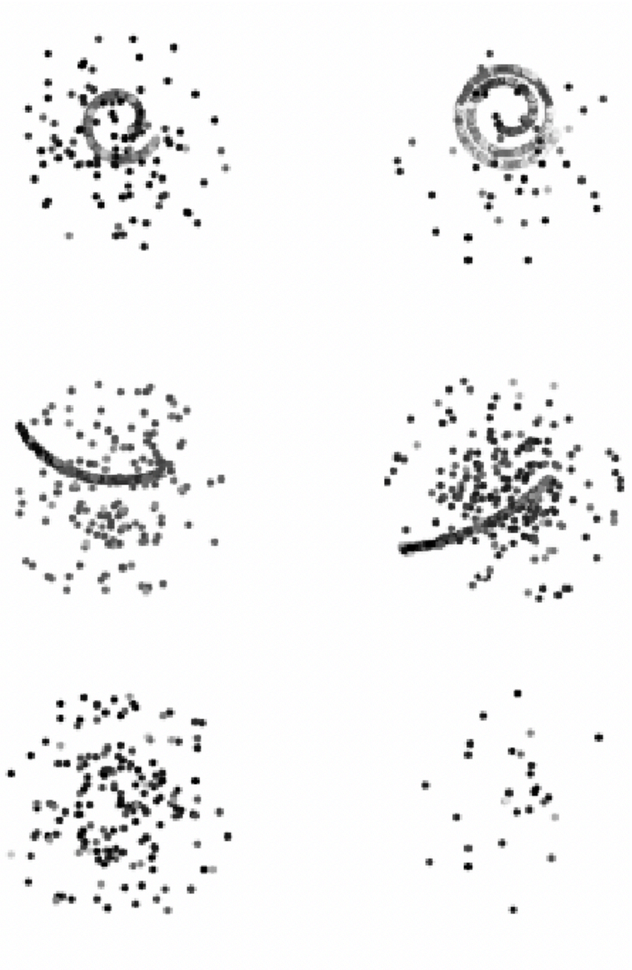
VGG16 ARCHITECTURE



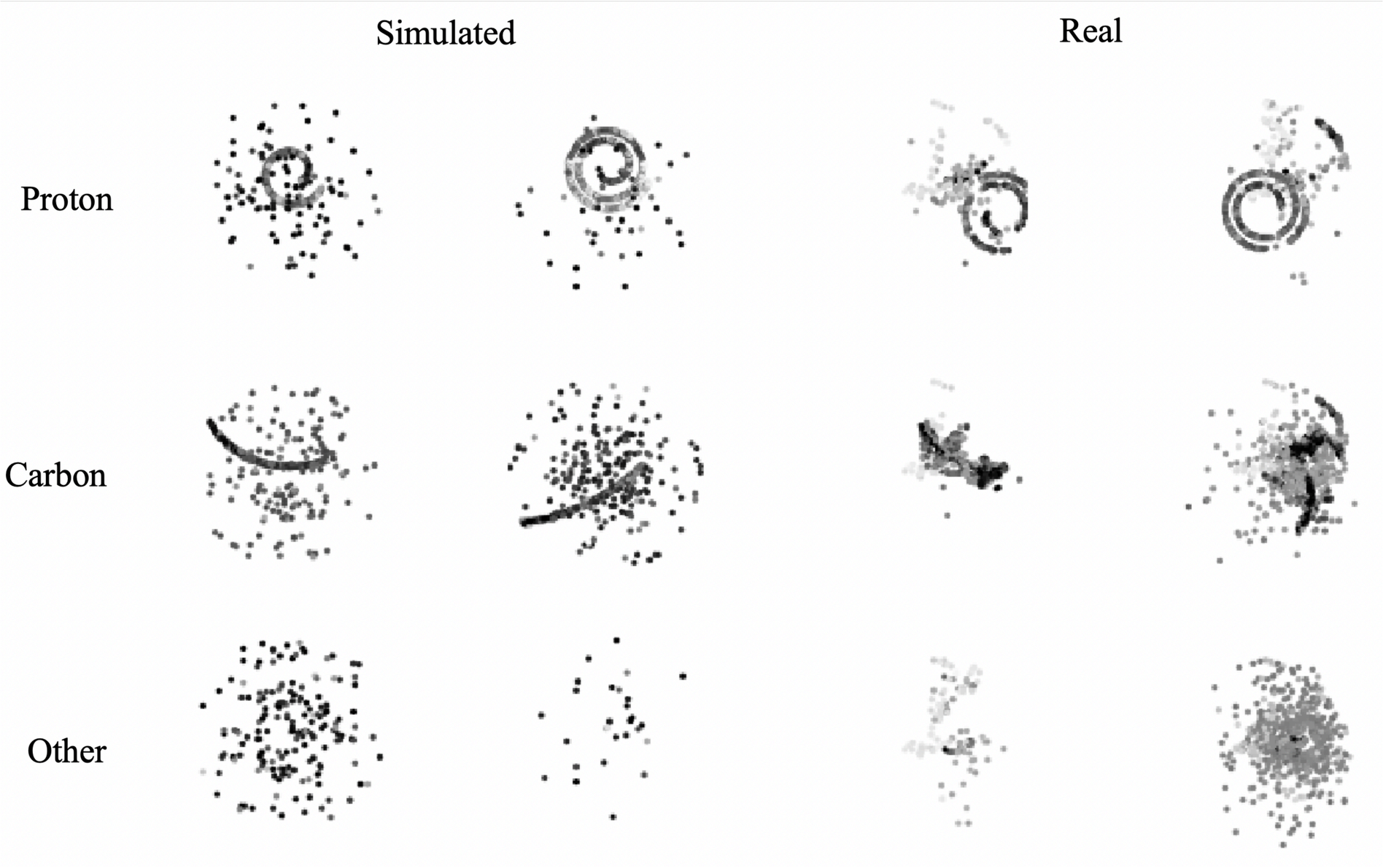
PRE-TRAINED ON IMAGENET DATA!

Experiment	Precision	Recall	F1
Experimental → Experimental	0.96	0.90	0.93

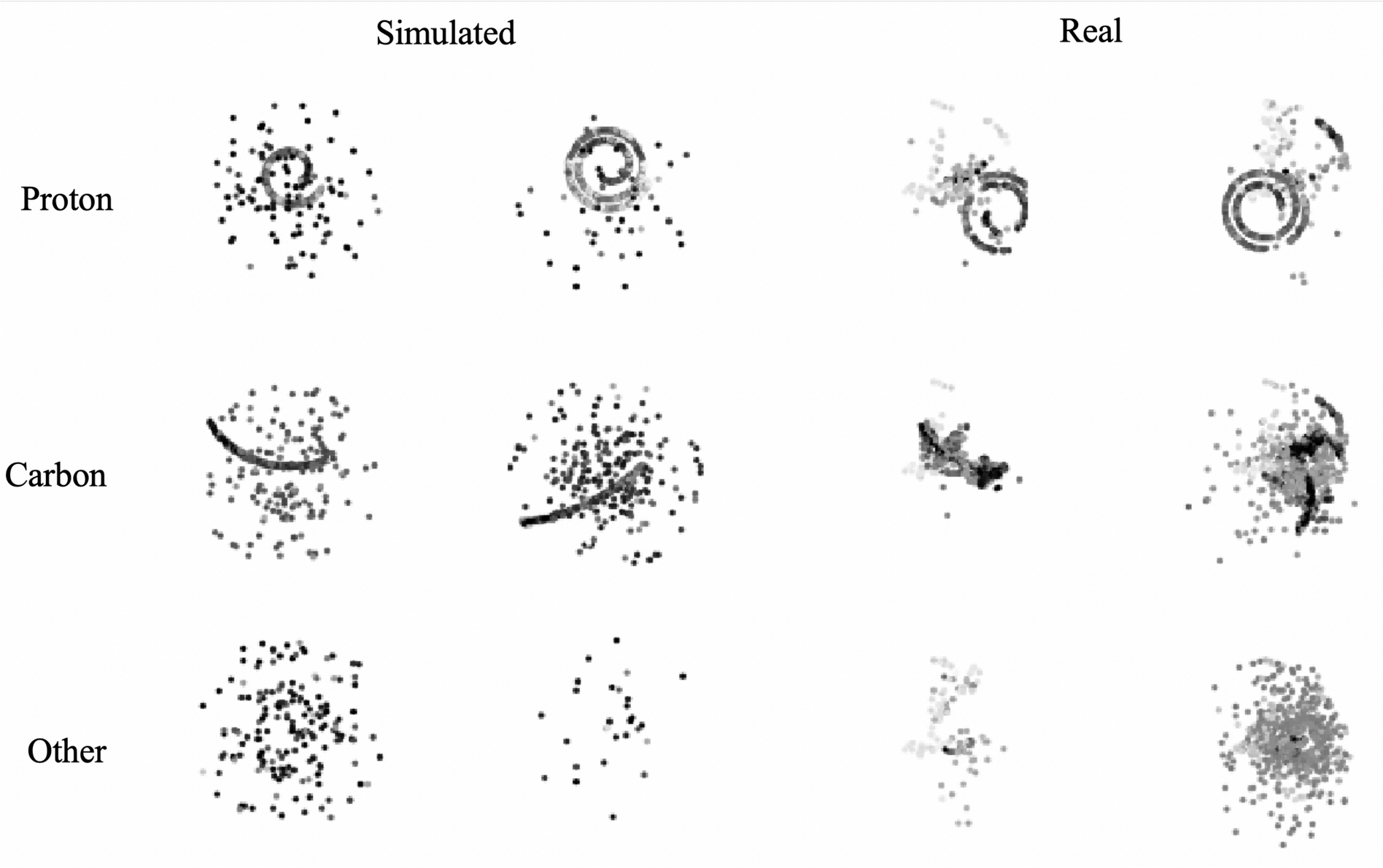
SIMULATED DATA



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


Simulated
Real
Proton
Carbon
Other

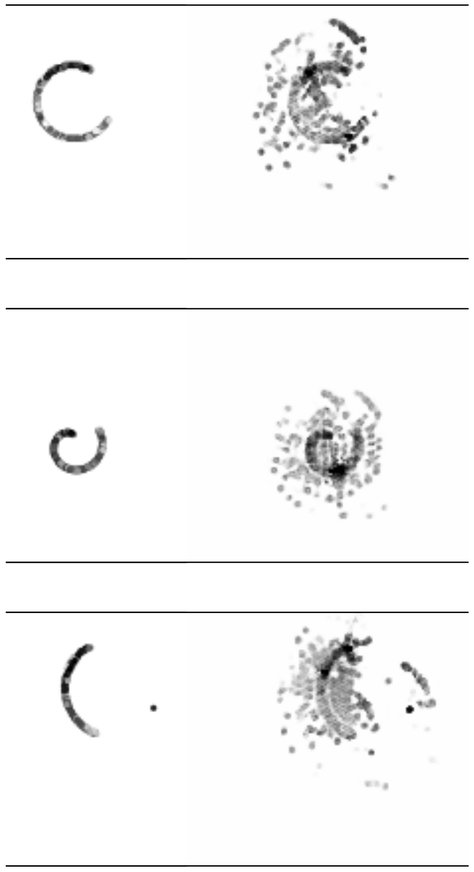
Experiment	Precision	Recall	F1
Experimental → Experimental	0.96	0.90	0.93
Simulated → Simulated	1.00	1.00	1.00
Simulated → Experimental	0.90	0.60	0.72

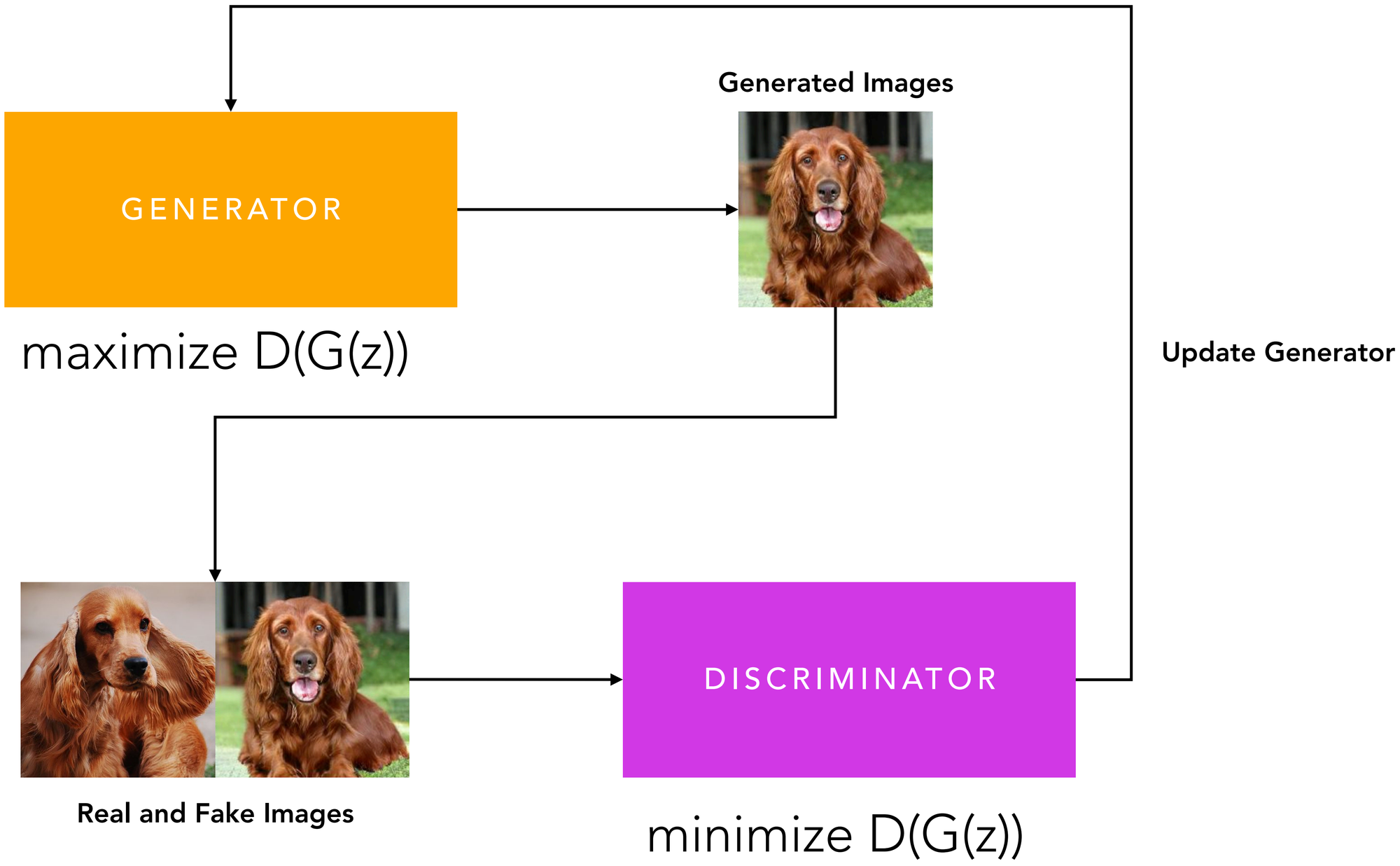

Simulated
Real
Proton
Carbon
Other

GENERATIVE ADVERSARIAL NETWORKS (GANS)

SIMULATION

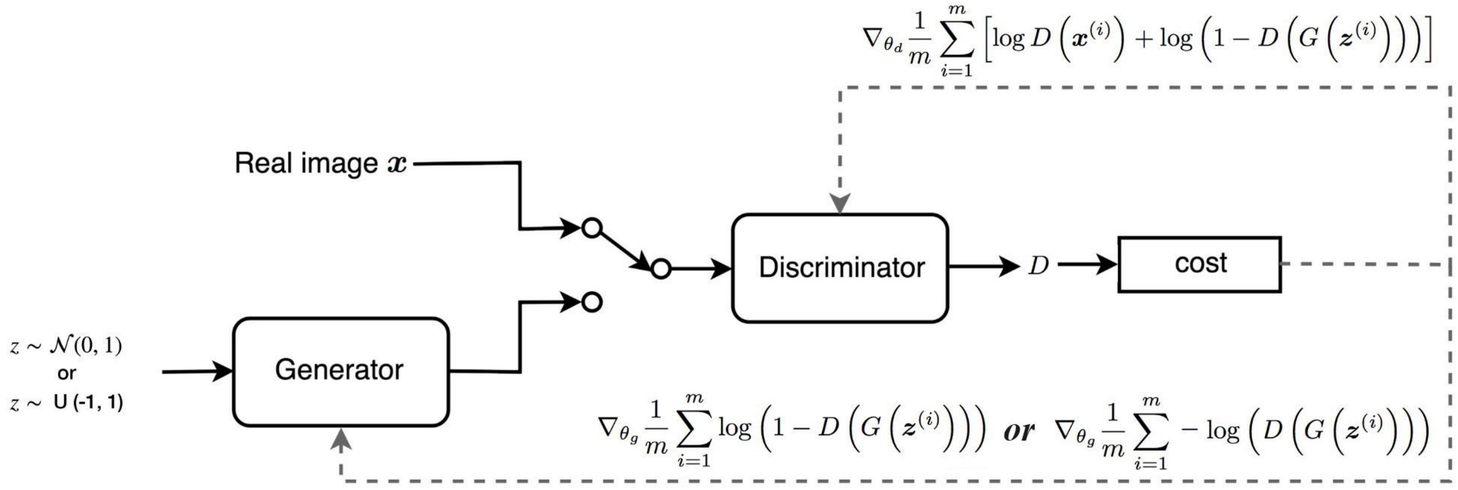
- **Goal:** learn to add realistic noise to a clean, simulated event
  - allow realistic simulation
  - transfer learn with higher accuracy



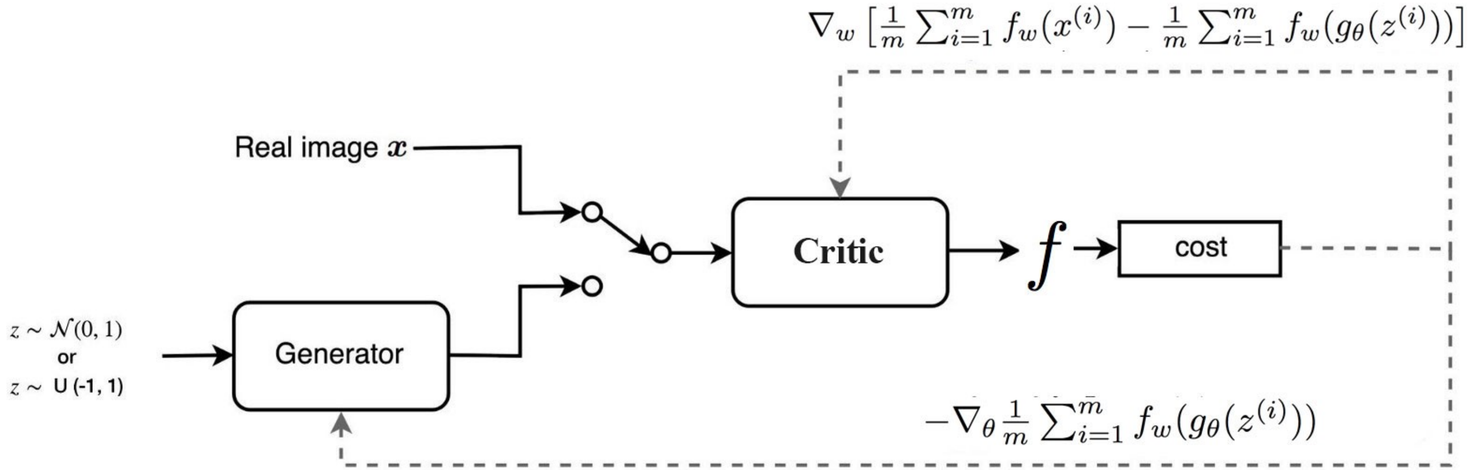


PHOTOS FROM KATRINA S; ANDREW B ET. AL., *BIGGAN*.

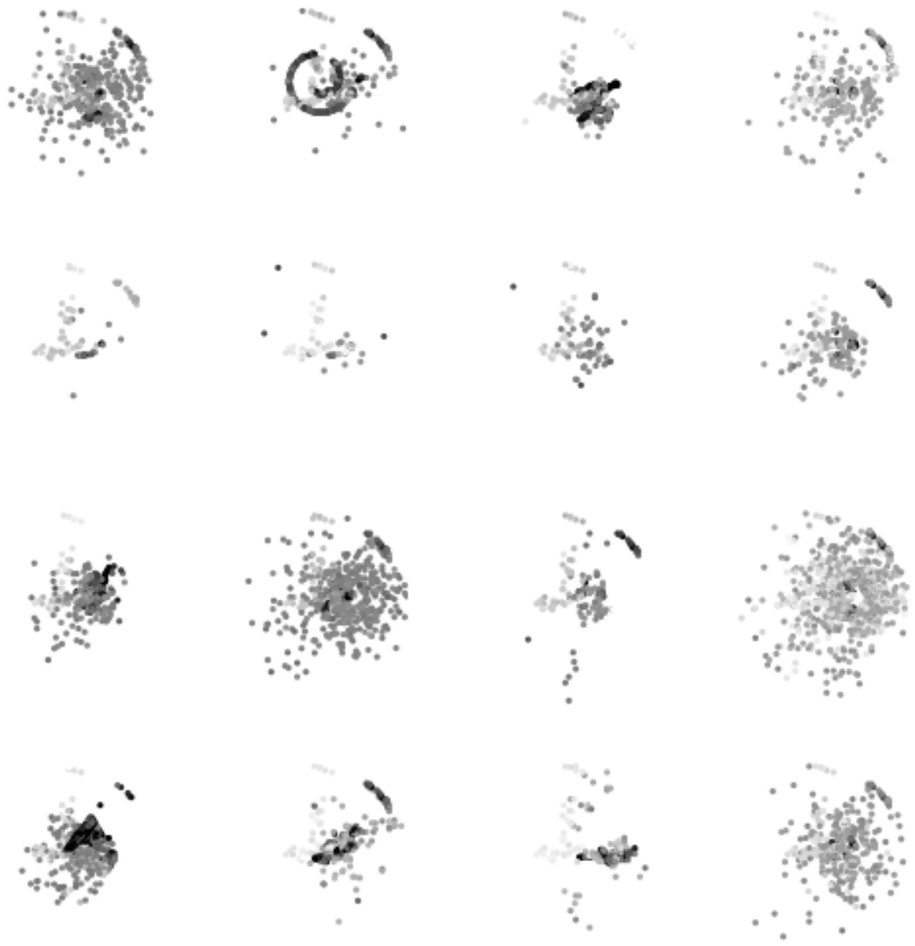
GAN (DCGAN)



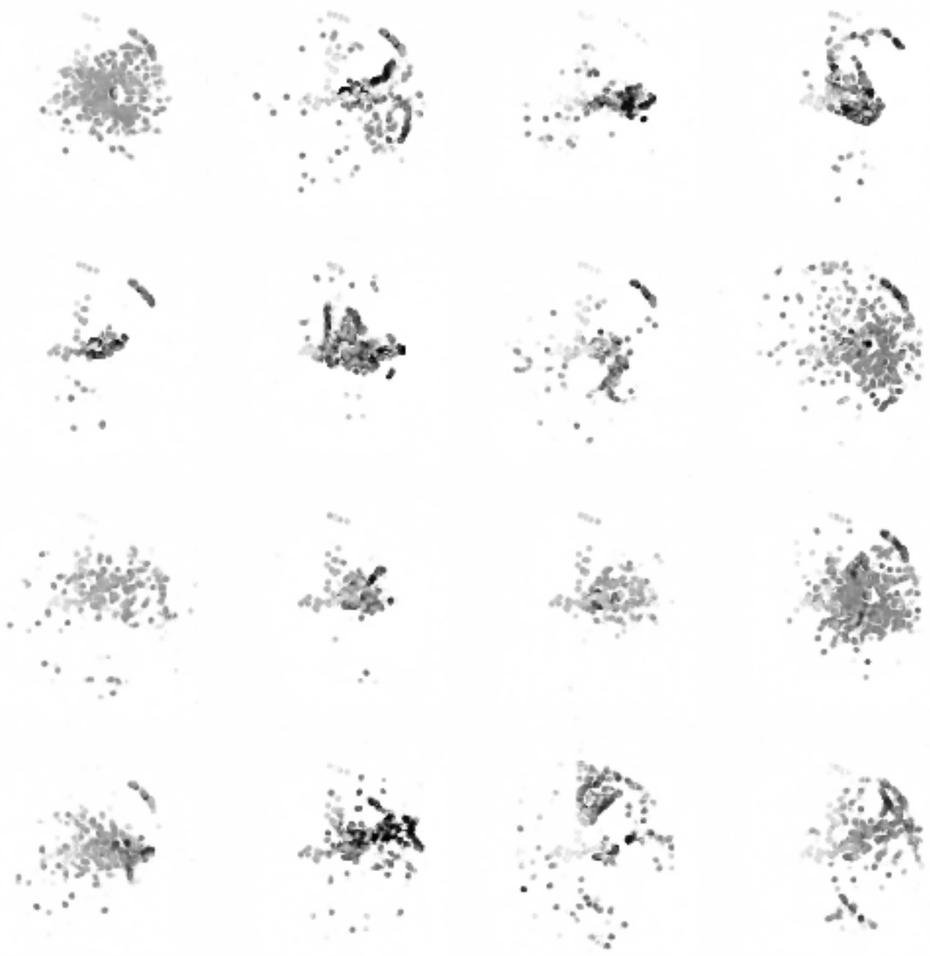
WGAN



Real



Generated



GAN Problems

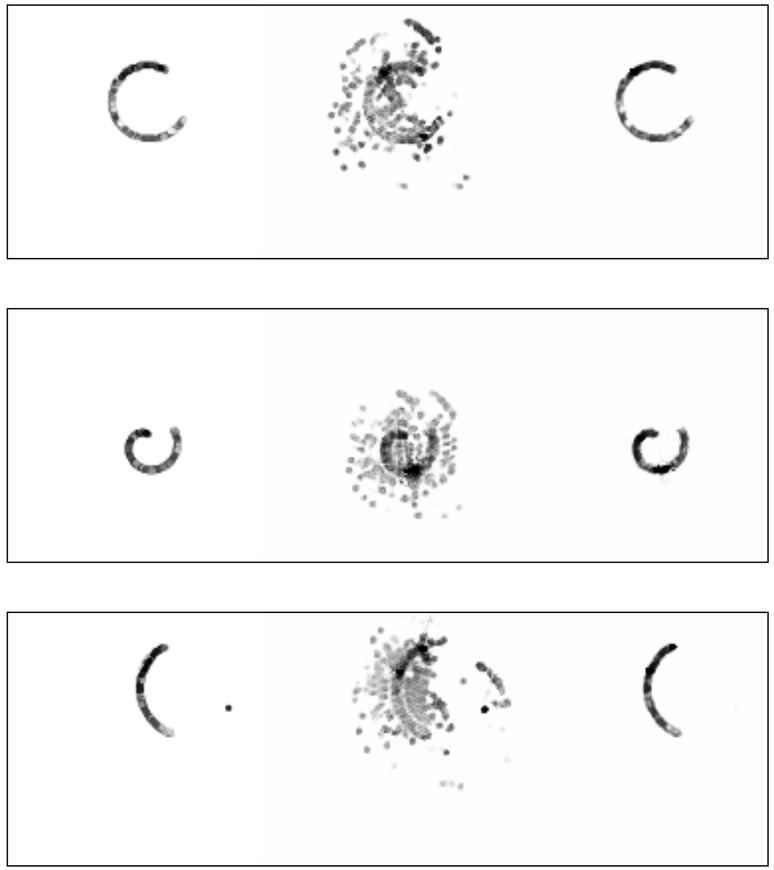
- **Vanishing gradients**
  - If the discriminator behaves badly, the generator does not have accurate feedback and the loss function cannot represent the reality.
  - If the discriminator does a great job, the gradient of the loss function drops too close to zero and the learning becomes super slow or even jammed.
- **Mode collapse**
  - During the training, the generator may collapse to a setting where it always produces same the outputs.
  - Even though the generator might be able to trick the corresponding discriminator, it fails to learn to represent the real-world data and gets stuck in a small space with extremely low variety.

Heuristic Tricks

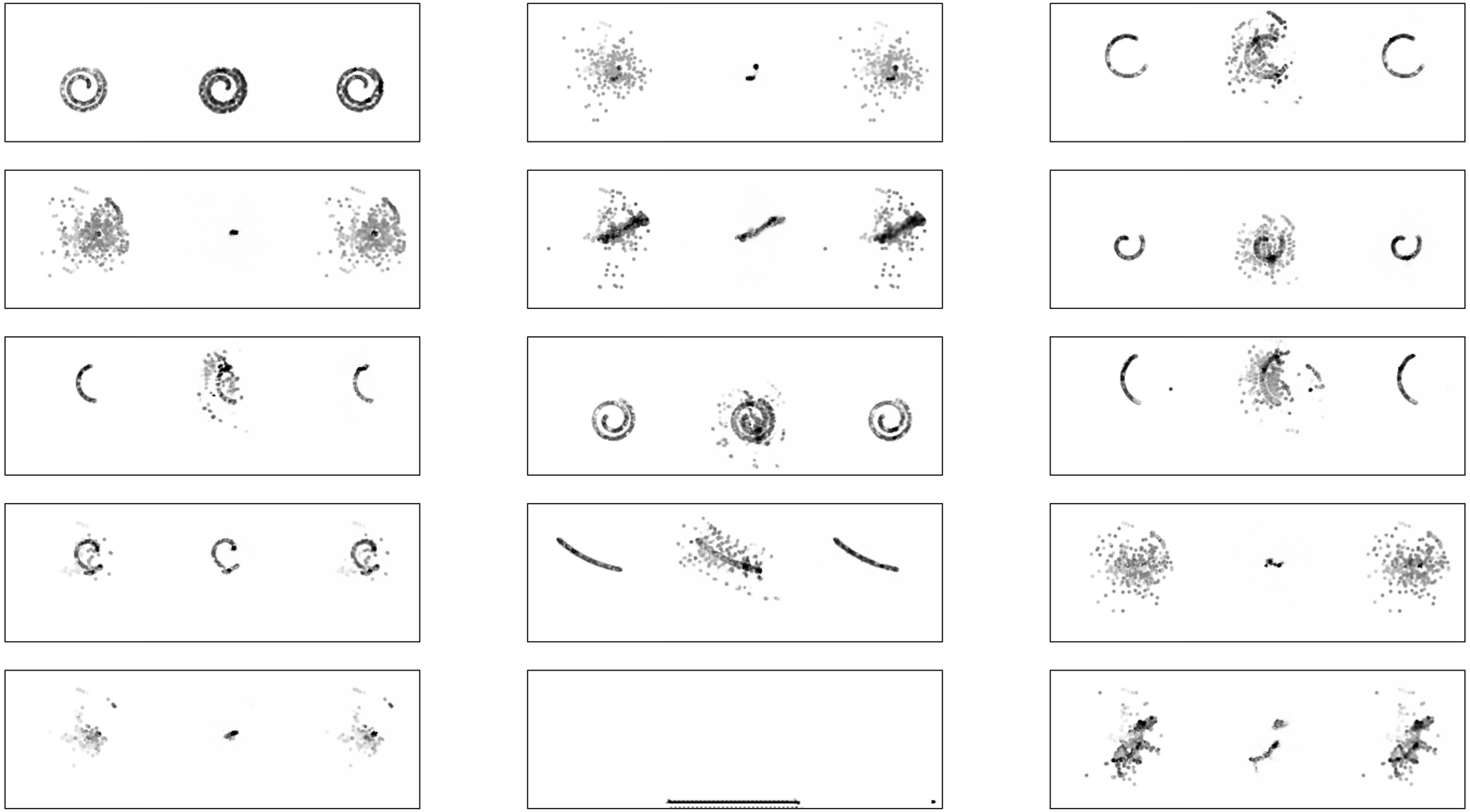
- Normalize the images between -1 and 1.
- Use `tanh` as the last activation in the generator, instead of `sigmoid`.
- Sample points from the latent space using a normal distribution, not a uniform distribution.
- Because GAN training results in a dynamic equilibrium, GANs are likely to get stuck in all sorts of ways. Introducing randomness during training helps prevent this.
  - Use dropout in the discriminator.
  - Add random noise to the labels for the discriminator.
  - Add gaussian noise to every layer of generator.
  - Use dropout in generator in both train and test phase.
- Avoid sparse gradients.
  - Instead of max pooling, use strided convolutions for downsampling.
  - Use a `LeakyReLU` layer instead of a `ReLU` activation (allows small negative values).
- Use a kernel size that's divisible by the stride size whenever using a strided `Conv2DTranspose` or `Conv2D` in both the generator and the discriminator.
- Use batch norm in both generator and discriminator.
- Remove fully-connected hidden layers for deeper architectures.
- Use SGD for discriminator and ADAM for generator.

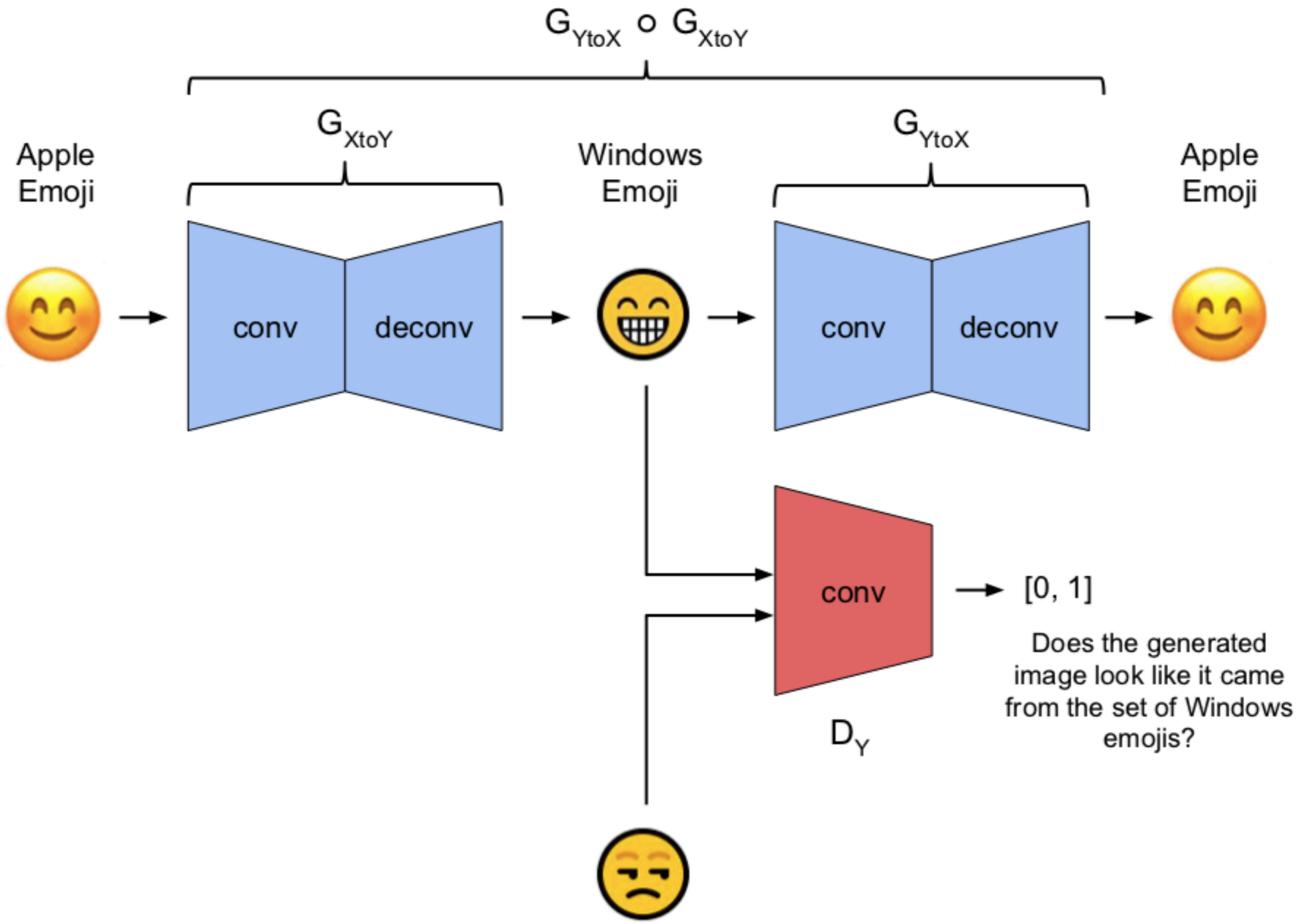
CYCLEGAN

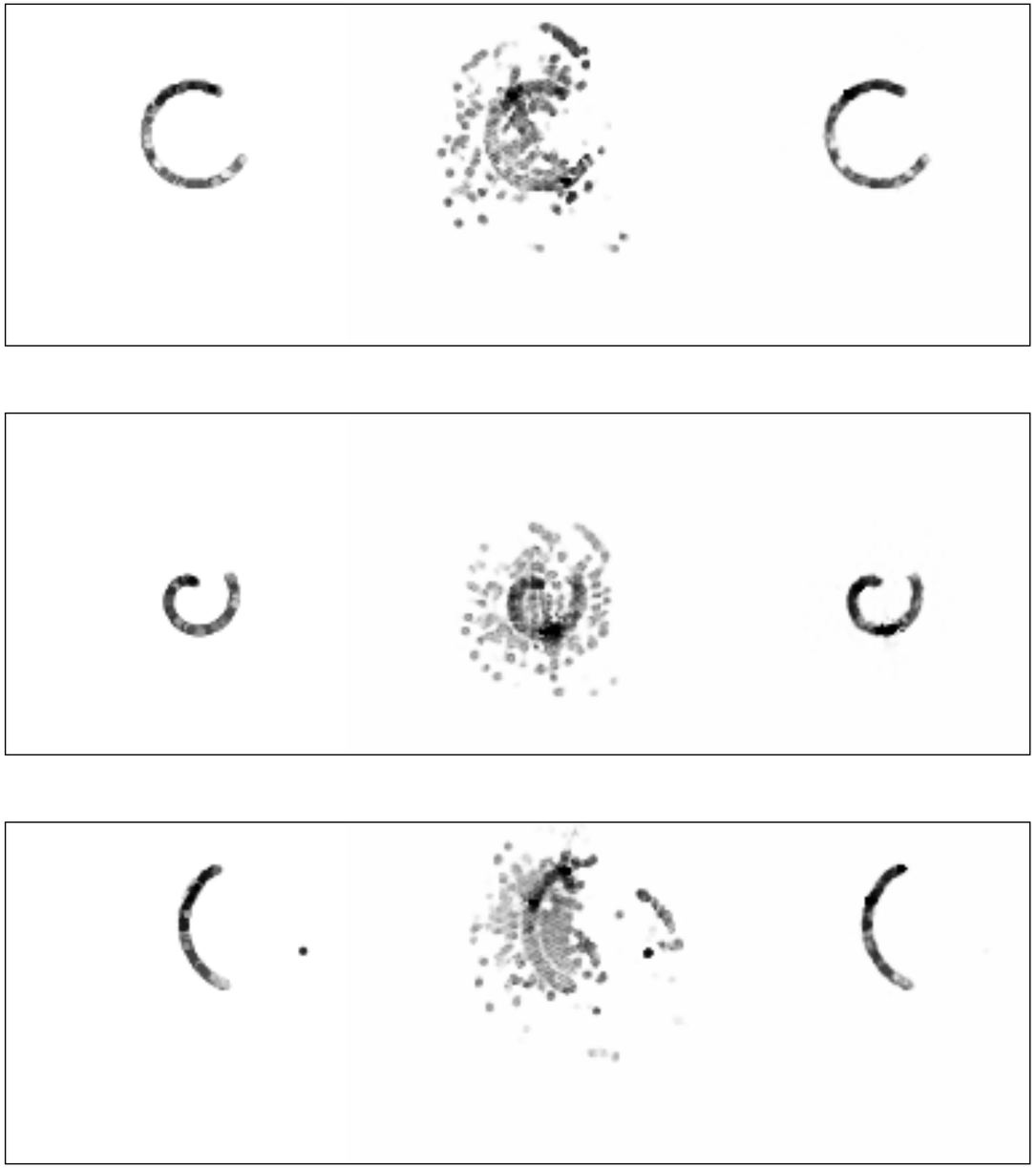
- **CycleGAN:** translate images from one domain to another
- Can both clean real data AND generate noisy data!



Original -> Translated -> Reconstructed




$G_{YtoX} \circ G_{XtoY}$
G_{XtoY}
G_{YtoX}
Apple Emoji
Windows Emoji
Apple Emoji
conv
deconv
conv
deconv
conv
[0, 1]
Does the generated image look like it came from the set of Windows emojis?
D_Y



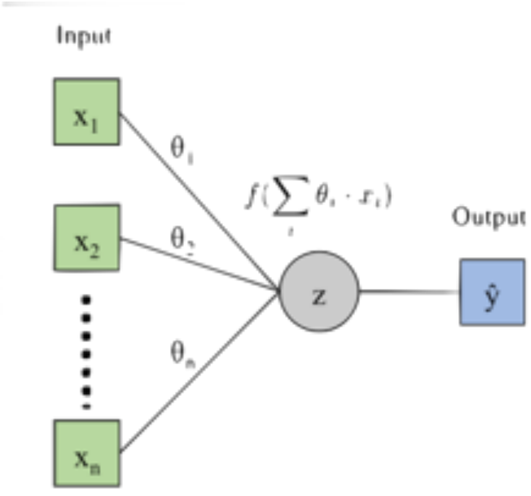
VALIDATION

- Are all generated data physical?
- Does charge distribution of generator match experimental data?

ACKNOWLEDGMENTS

- Raghu Ramanujan
- Ryan Strauss, Jack Taylor, Christina Chen
- ATTPC Group
  - Daniel Bazin, Wolfi Mittig
- NSCL/FRIB

Experimental learning task	Precision	Recall	F1
Logistic Regression	0.77	0.67	0.72
FC Neural Network	0.83	0.62	0.71



Transfer learning task	Precision	Recall	F1
Logistic Regression	0.44	0.03	0.05
FC Neural Network	0.78	0.44	0.57

CNNS

Experimental task: accuracy: 0.98	Precision	Recall	F1
Proton	0.97	0.88	0.93
Non-proton	0.96	0.99	0.98

Transfer task: accuracy: 0.95	Precision	Recall	F1
Proton	0.94	0.71	0.81
Non-proton	0.92	0.99	0.95