Coding AI / ML: From Prototyping to reproduceable (Data) Science

GSPDA Workshop May 2024

Steven Goldenberg & Daniel Lersch for the
JLAB Data Science Department

Friday, May 24, 2024

Jefferson Lab

U.S. DEPARTMENT OF ENERGY | Office of Science

JSA

On the Menu

- 2h Tutorial / Crash course in machine / deep learning supported analyses

- First half:
  - Brief introduction to machine / deep learning
  - Model Evaluation Metrics
  - orkflows

- Second half:
  - [Hands-On Classification Example]
  - Presentation on deep learning workflow

On the Menu

Plot taken from [Brenda Ngs talk at deep learning for science school 2019]

AI ⊃ ML ⊃ DL

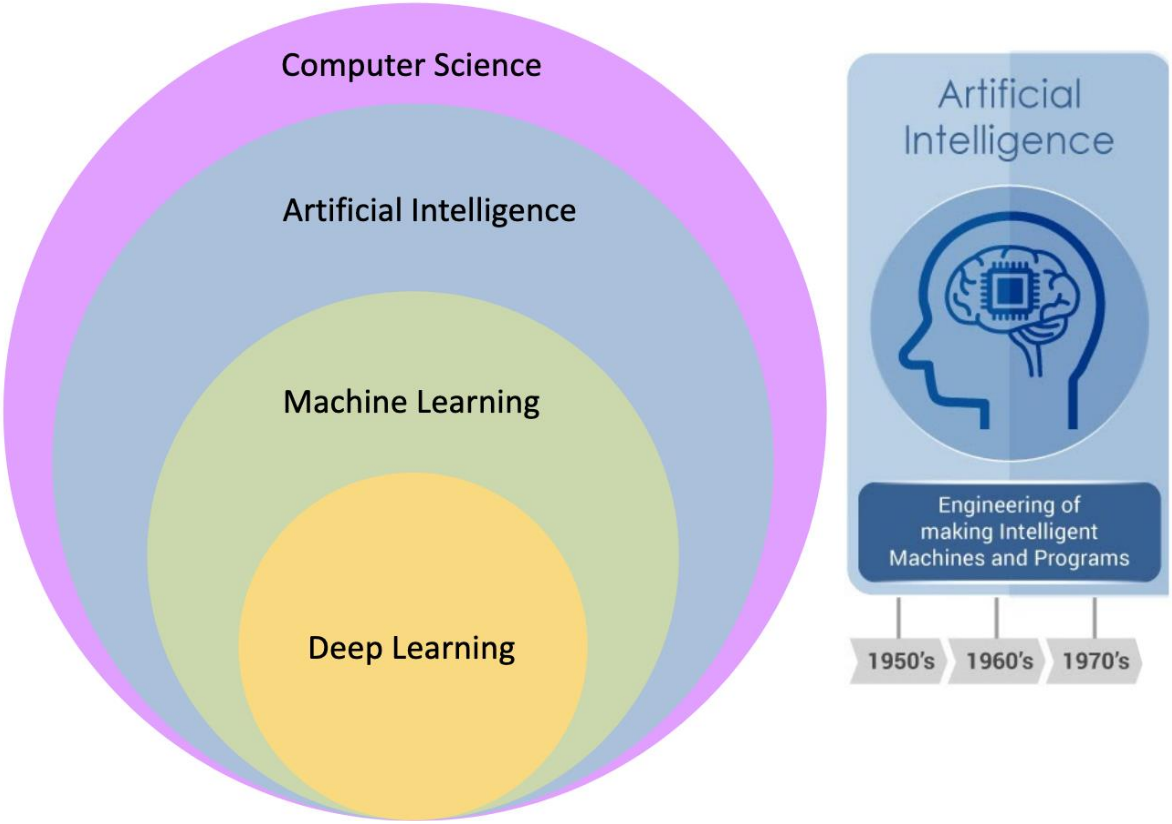
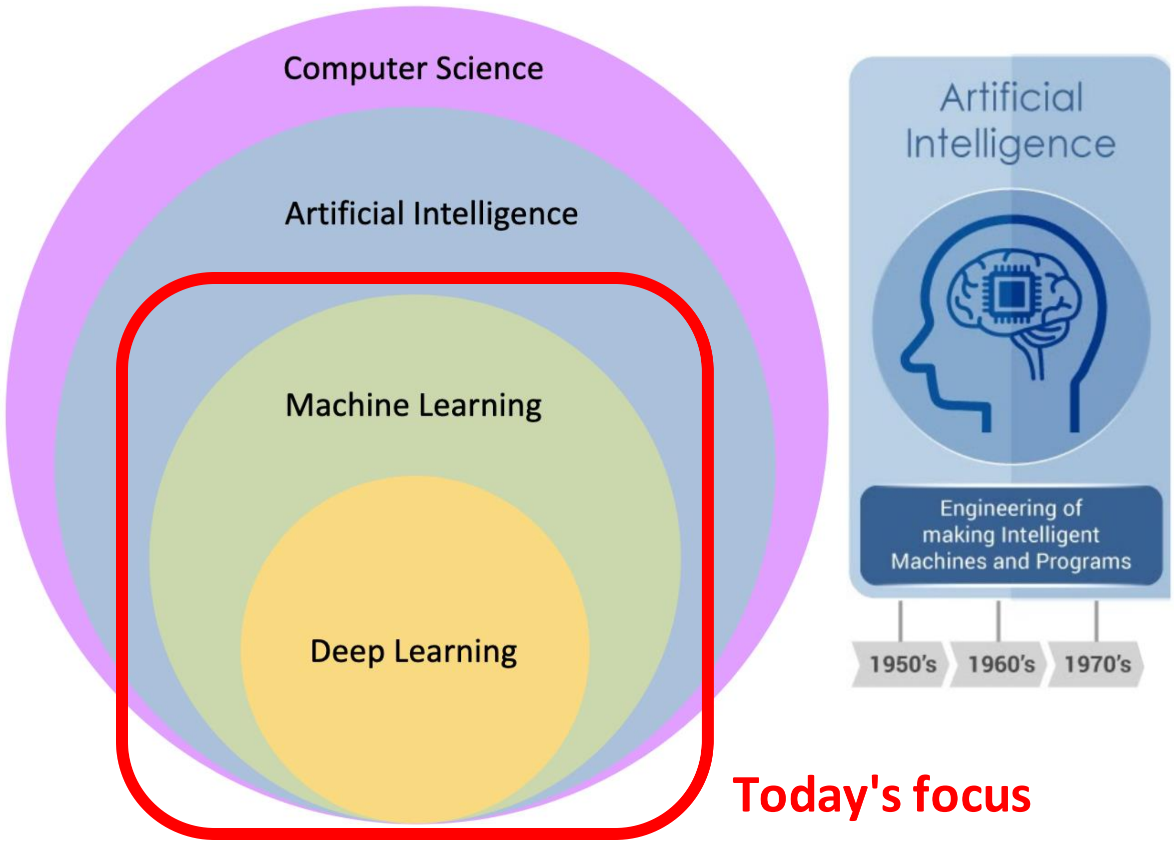


Image source: https://www.embedded-vision.com/industry-analysis/blog/artificial-intelligence-machine-learning-deep-learning-and-computer-visionwha

On the Menu

Plot taken from [Brenda Ngs talk at deep learning for science school 2019]

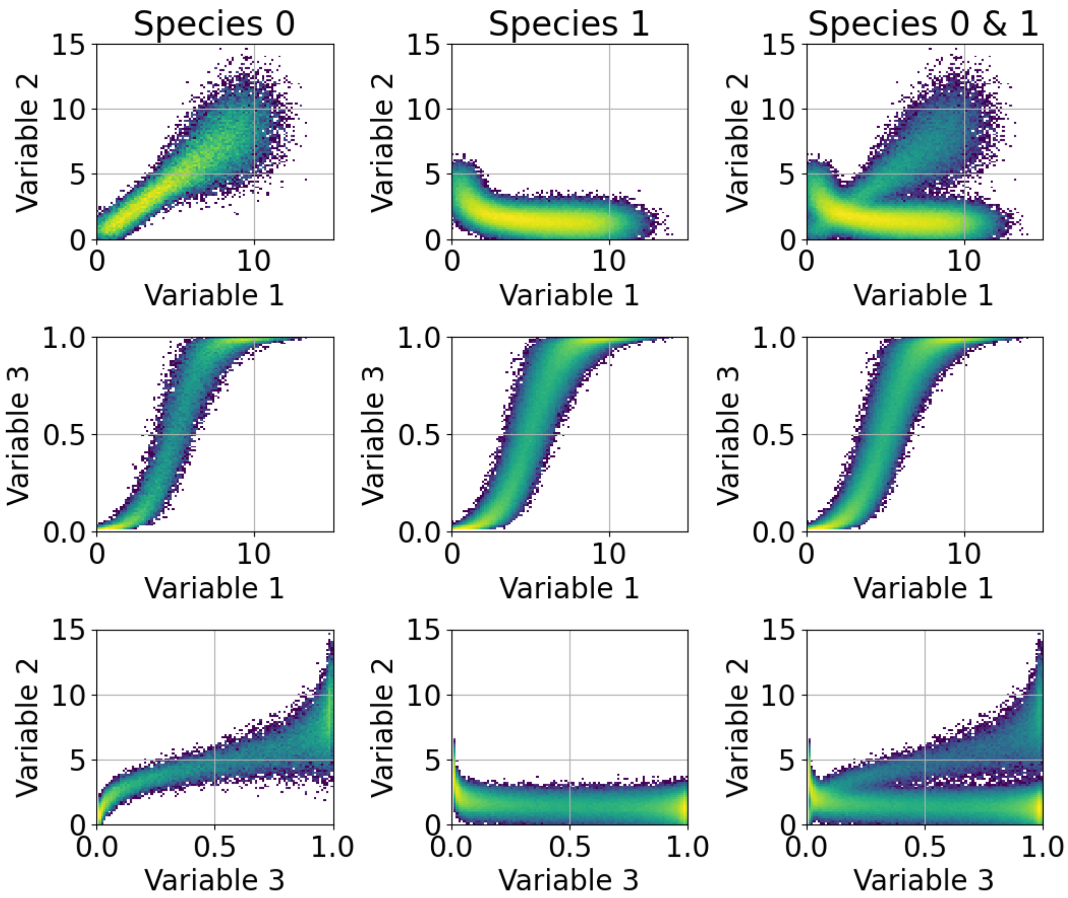
AI ⊃ ML ⊃ DL



Today's focus

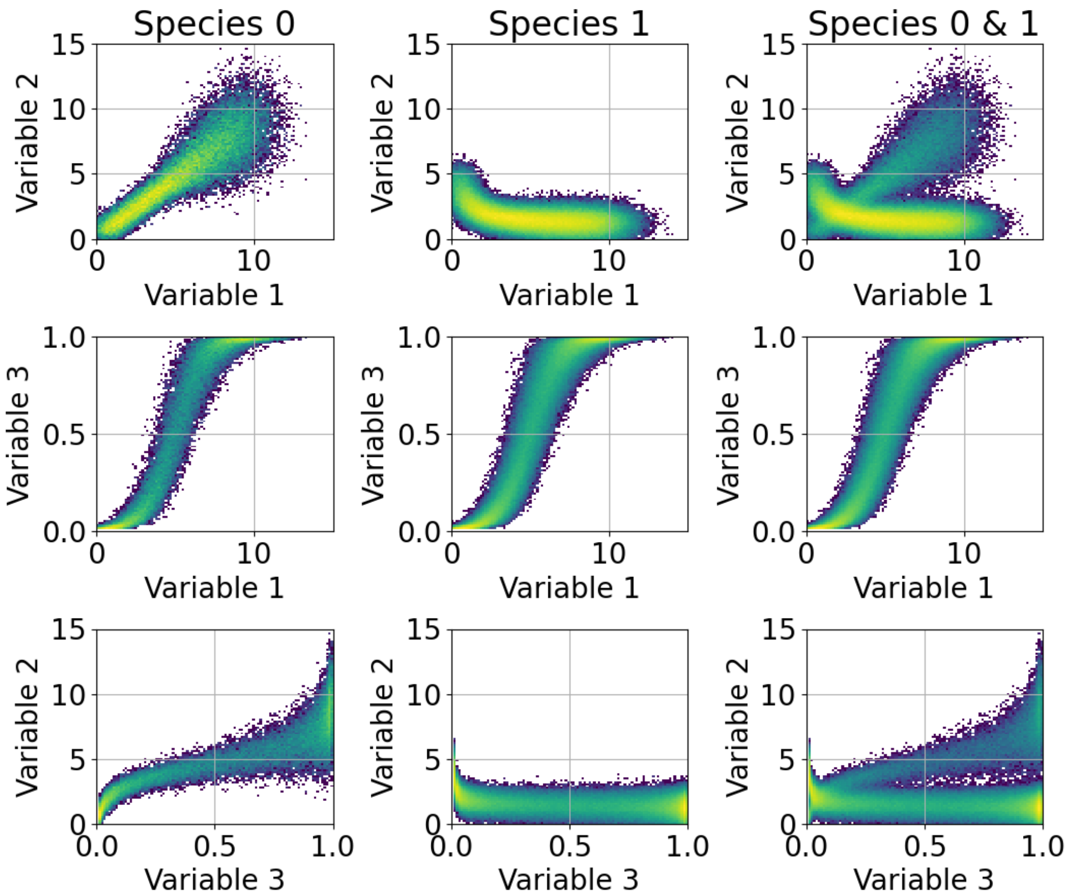
Image source: https://www.embedded-vision.com/industry-analysis/blog/artificial-intelligence-machine-learning-deep-learning-and-computer-visionwha

A Binary Classification Problem



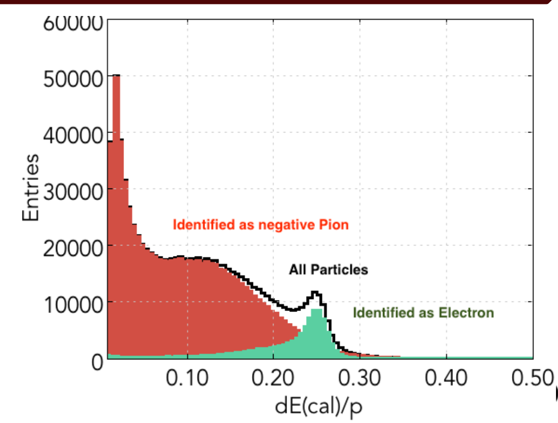
- 288k events with 2 (Particle) Species
- Each characterized by 3 variables (e.g. information from a detector)
- Species 1 is more abundant than species 0
- **Task:** Identify each species, based on the provided information

A Binary Classification Problem

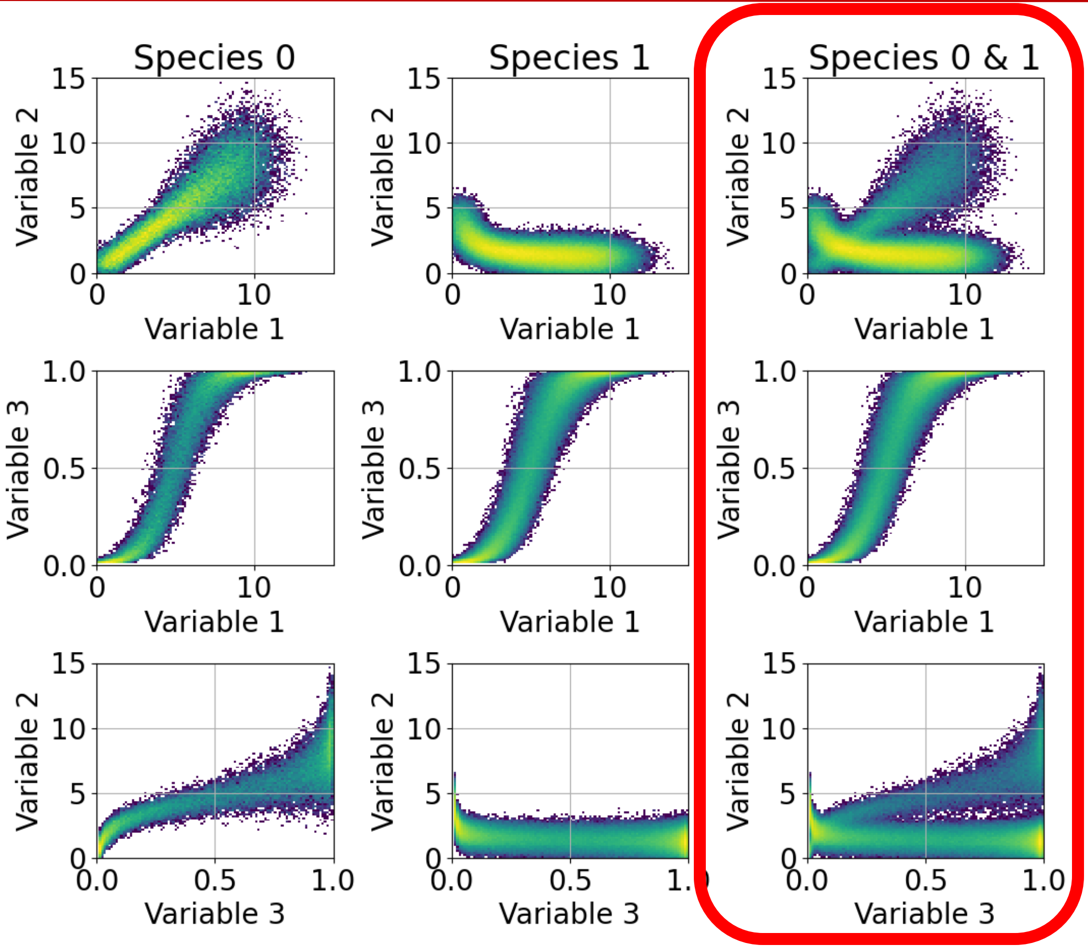


- 288k events with 2 (Particle) Species
- Each characterized by 3 variables (e.g. information from a detector)
- Species 1 is more abundant than species 0
- **Task:** Identify each species, based on the provided information

Related problem at JLab : electron / pion separation with pions being the majority



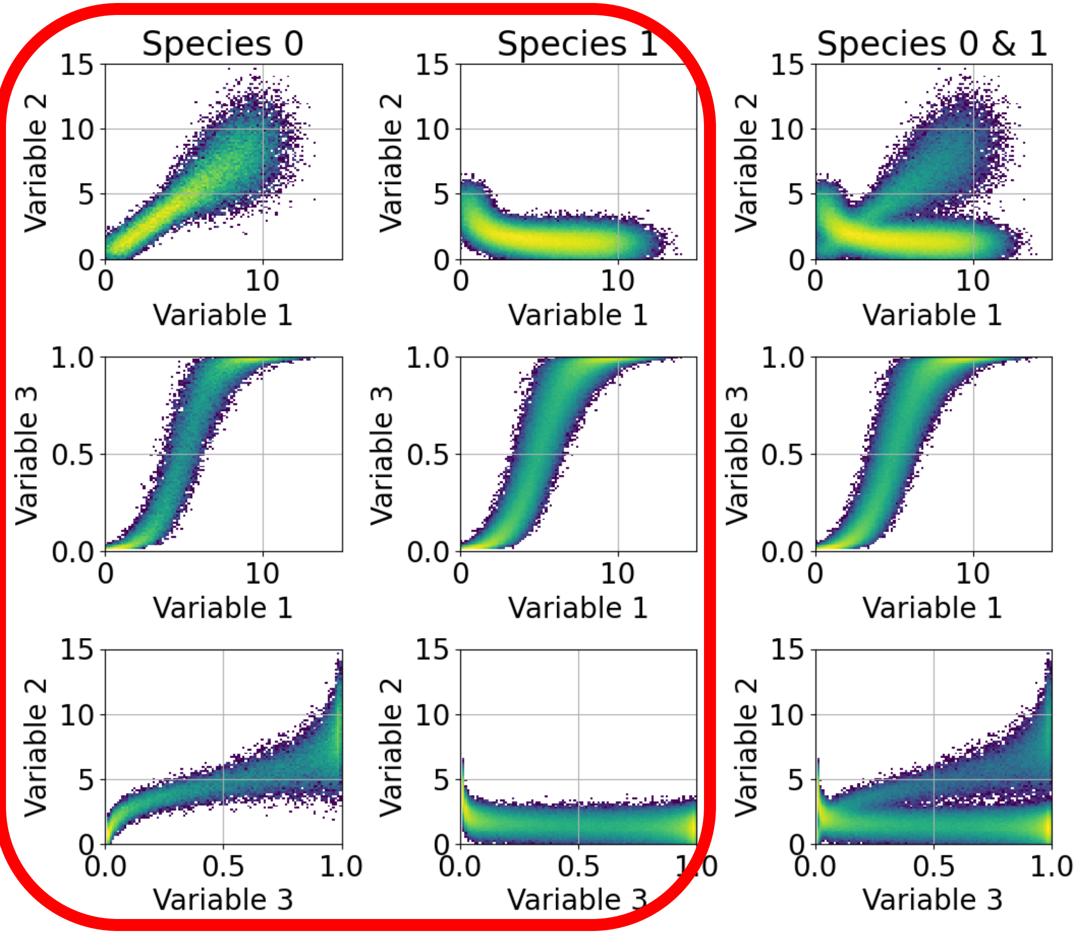
A Binary Classification Problem



- 288k events with 2 (Particle) Species
- Each characterized by 3 variables (e.g. information from a detector)
- Species 1 is more abundant than species 0
- **Task:** Identify each species, based on the provided information

This is what we see in our data

A Binary Classification Problem

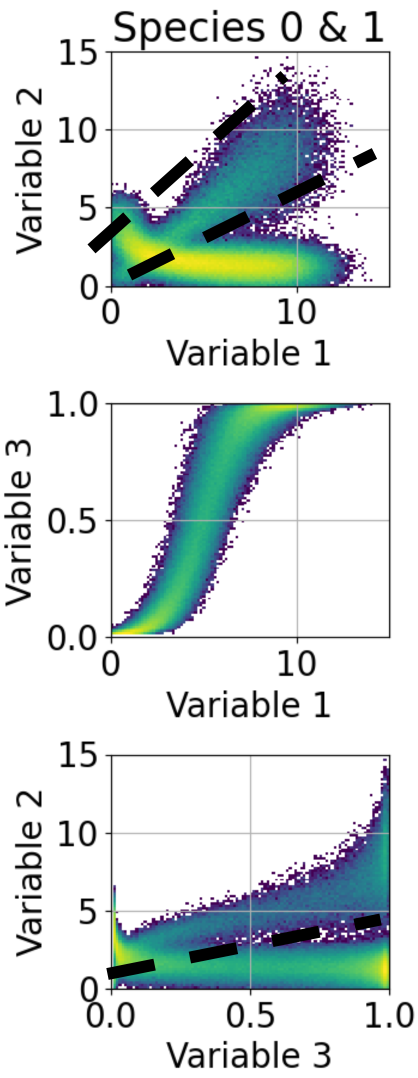


- 288k events with 2 (Particle) Species
- Each characterized by 3 variables (e.g. information from a detector)
- Species 1 is more abundant than species 0
- **Task:** Identify each species, based on the provided information

This is what we would like to see after identification

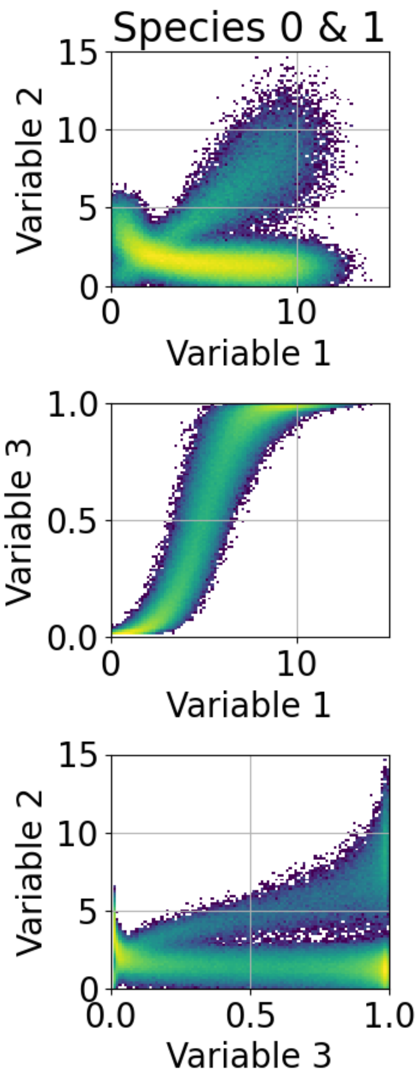
Jefferson Lab

What are we looking for?



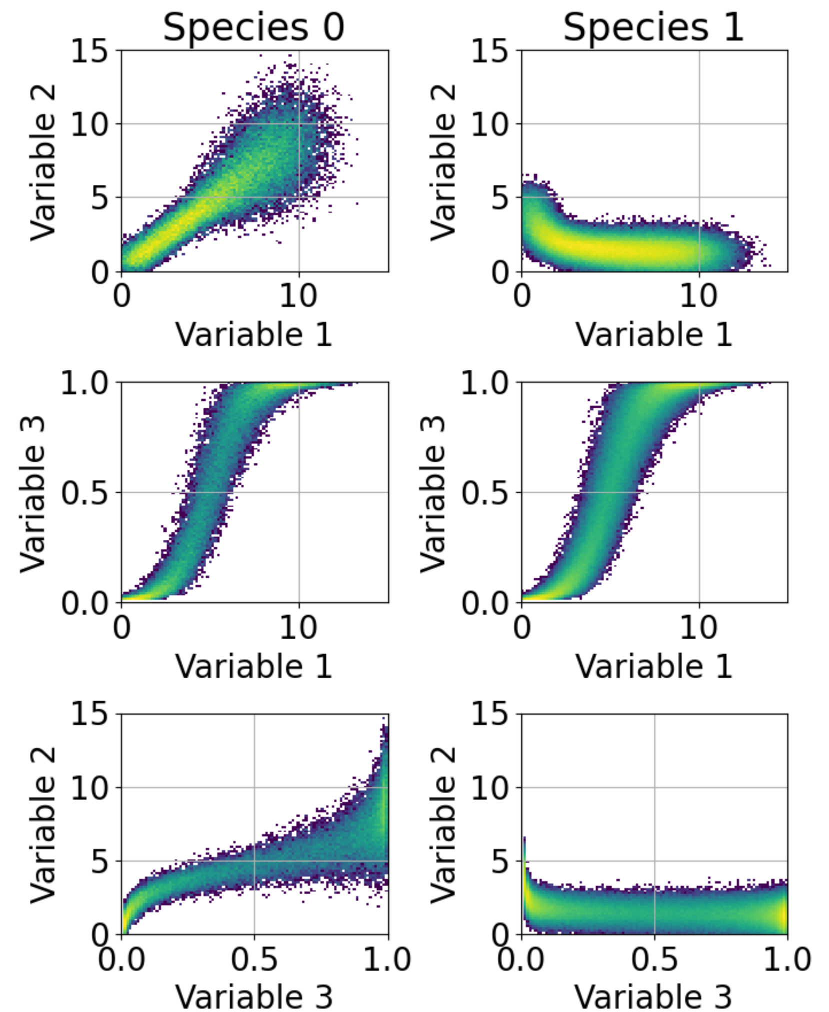
- We could try to solve this "by hand"
- Use linear cuts to separate species (nothing wrong with this approach)
- Only drawbacks:
  - Overlapping regions cause misidentification
  - Do not fully utilize (unknown) variable correlations --> Linear cut is too simple
- Spend more time on tuning the cuts --> Use a more complex function ?
- What is the underlying function that helps us to separate the two species ?

What are we looking for?



Find a model that mimics the underlying function

Mysterious Model



What we expect from our Model

1. Predictive Power
   - Extract all available information withing the given data
   - Utilize correlations, even the hidden ones
   - Provide smallest prediction error possible
2. Generalizability
   - Applicable to future data sets that we are unaware of
   - Avoid overfitting (do not want a model that is tailored to one specific data set)
3. Explainability
   - This is a tricky one and a can of worms...
   - Need to understand model performance on given data
   - How do certain features impact the prediction ?
   - This is an entire research field on its own

The Model

Input Data **Model** **Response**

$$X \rightarrow \boxed{f_\theta} \rightarrow \hat{Y}$$

- Model has internal parameters θ
- Response depends on input data and internal parameters: $\hat{Y} = f_\theta(X)$
- f_θ is, not necessarily, continuous and differentiable

The Model

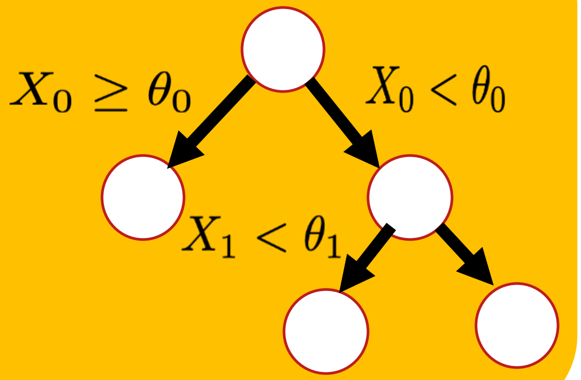
Input Data | **Model** | **Response**

$$X \rightarrow f_\theta \rightarrow \hat{Y}$$

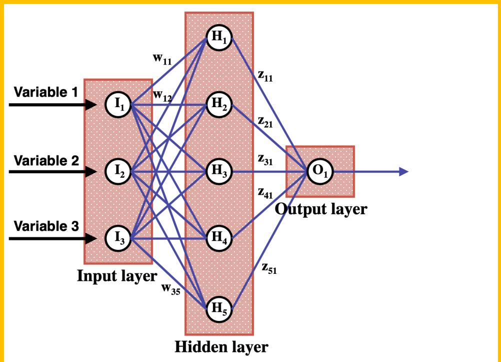
Linear Function

$$\hat{Y} = \theta_1 \cdot X + \theta_0$$

Decision Tree



Neural Network



and many more…

Jefferson Lab

The Model

Input Data **Model** **Response**

$$X \rightarrow f_\theta \rightarrow \hat{Y}$$

How do we set these ?

Jefferson Lab

Model Training / Fitting

Input Data | **Model** | **Response**

$$X \longrightarrow \boxed{f_\theta} \longrightarrow \hat{Y}$$

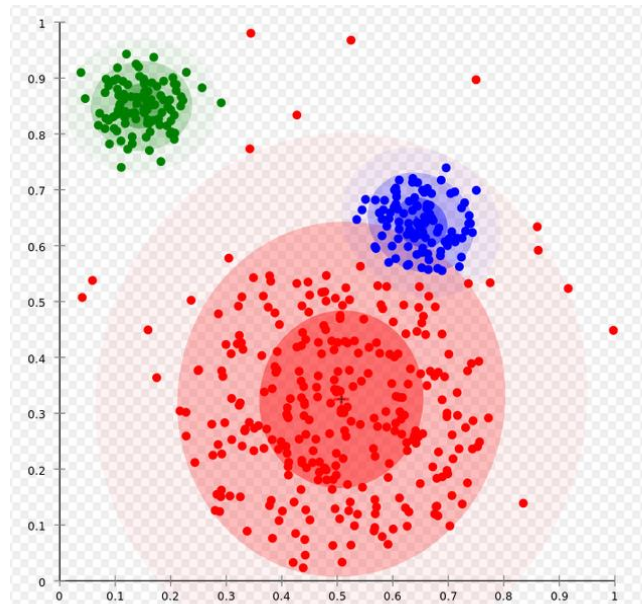
- Find θ that minimize / maximize objective F: $\frac{dF(\hat{Y})}{d\theta} = \frac{dF(f_\theta(X))}{d\theta} = 0$
- Objective is defined by underlying problem that you are trying to solve
- **Supervised Learning:**
  - $F(\hat{Y}) = F(\hat{Y}, Y)$
  - Targets Y are known (e.g. labels)
- **Unsupervised Learning:**
  - No specific targets
  - Clustering algorithms: $F(\hat{Y}) \propto$ Distance
  - Autoencoders: $F(\hat{Y}) = F(\hat{Y}, X)$

Model Training / Fitting

Input Data — **Model** — **Response**

$$X \rightarrow f_\theta \rightarrow \hat{Y}$$

- Find θ that minimize / maximize objective F: $\frac{dF(\hat{Y})}{d\theta} = \frac{dF(f_\theta(X))}{d\theta} = 0$
- Objective is defined by underlying problem that you are trying to solve
- **Supervised Learning:**
  - $F(\hat{Y}) = F(\hat{Y}, Y)$
  - Targets Y are known (e.g. labels)
- **Unsupervised Learning:**
  - No specific targets
  - Clustering algorithms: $F(\hat{Y}) \propto$ Distance
  - Autoencoders: $F(\hat{Y}) = F(\hat{Y}, X)$



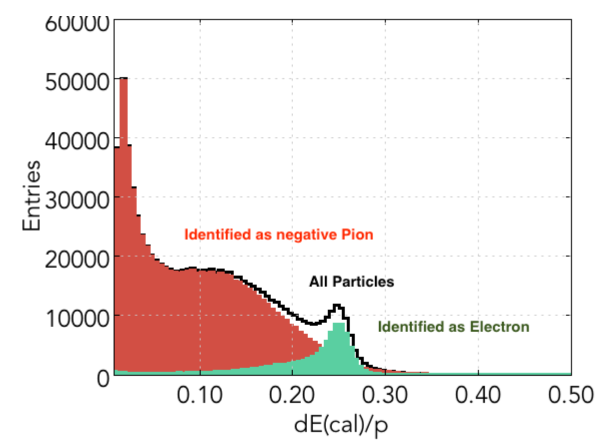
Model Training / Fitting

Input Data | **Model** | **Response**

$$X \rightarrow \boxed{f_\theta} \rightarrow \hat{Y}$$

- Find θ that minimize / maximize objective F: $\frac{dF(\hat{Y})}{d\theta} = \frac{dF(f_\theta(X))}{d\theta} = 0$
- Objective is defined by underlying problem that you are trying to solve
- **Supervised Learning:**
  - $F(\hat{Y}) = F(\hat{Y}, Y)$
  - Targets Y are known (e.g. labels)

Today's focus



- **Unsupervised Learning:**
  - No specific targets
  - Clustering algorithms: $F(\hat{Y}) \propto$ Distance
  - Autoencoders: $F(\hat{Y}) = F(\hat{Y}, X)$

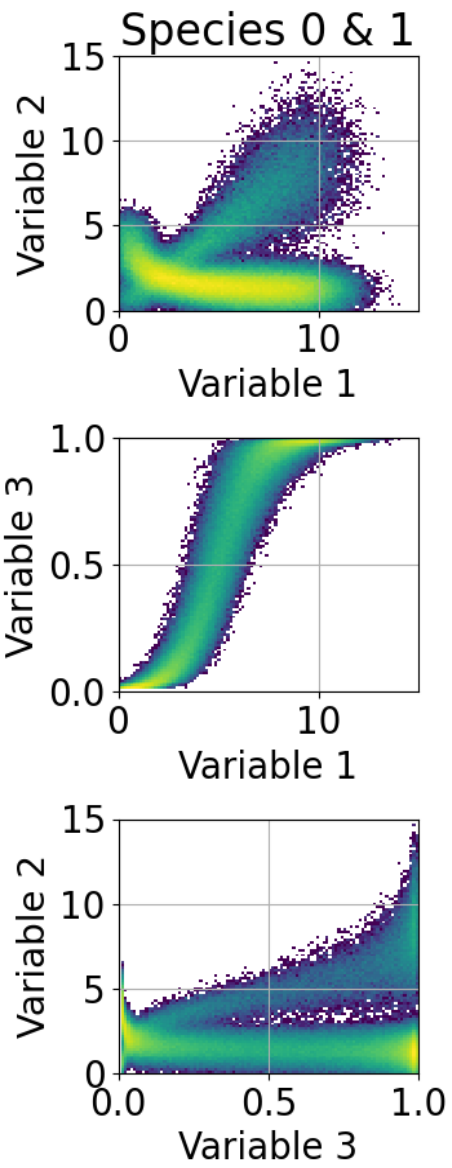
Optimization Techniques

$$\frac{dF(\hat{Y})}{d\theta} = \frac{dF(f_\theta(X))}{d\theta} = 0$$

- Various optimization techniques on the market
- Simulated Annealing (SA), Genetic Algorithm (GA), Particle Swarm, Backpropagation,…
- Some models work better with certain optimization techniques than others

Model	Preferred Optimization Method
Linear Model	Chi-Square Minimization, SA, GA
Decision Tree	Iterative Dichotomiser
Neural Networks	Backpropagation

Training Strategy for our Classification Problem



- Variables are summarized in 3D feature vector

$$X = (\text{Variable 1, Variable 2, Variable 3})$$

- Our data is labeled

$$\text{Label } \ell = \begin{cases} 1, & \text{if X is species 1,} \\ 0, & \text{if X is species 0} \end{cases}$$

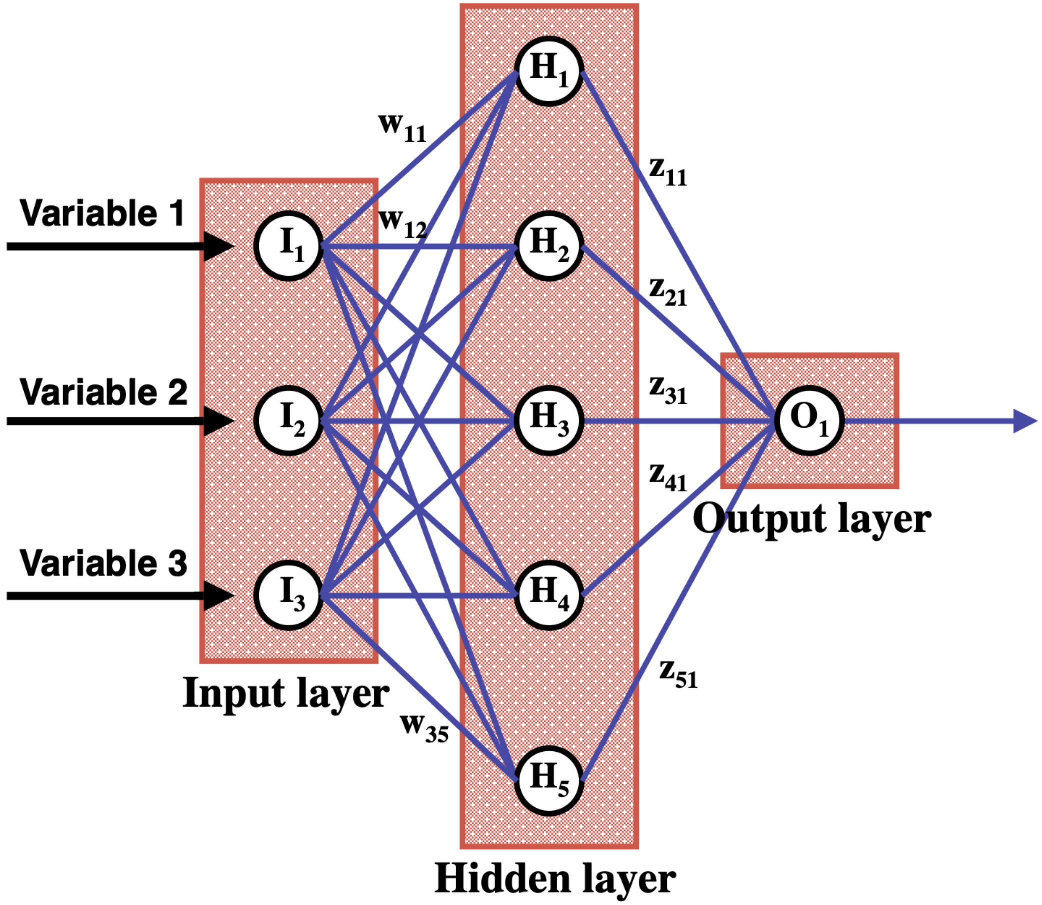
- Use supervised learning to train a model
  - Model learns labels
  - Use only 75% of data for training (explain later what happens to the remaining 25%)

- Use trained model to separate species

$$\text{model}(X) \approx \begin{cases} 0, & \text{identify as species 0,} \\ 1, & \text{identify as species 1} \end{cases}$$

- What kind of model do we want to use ?

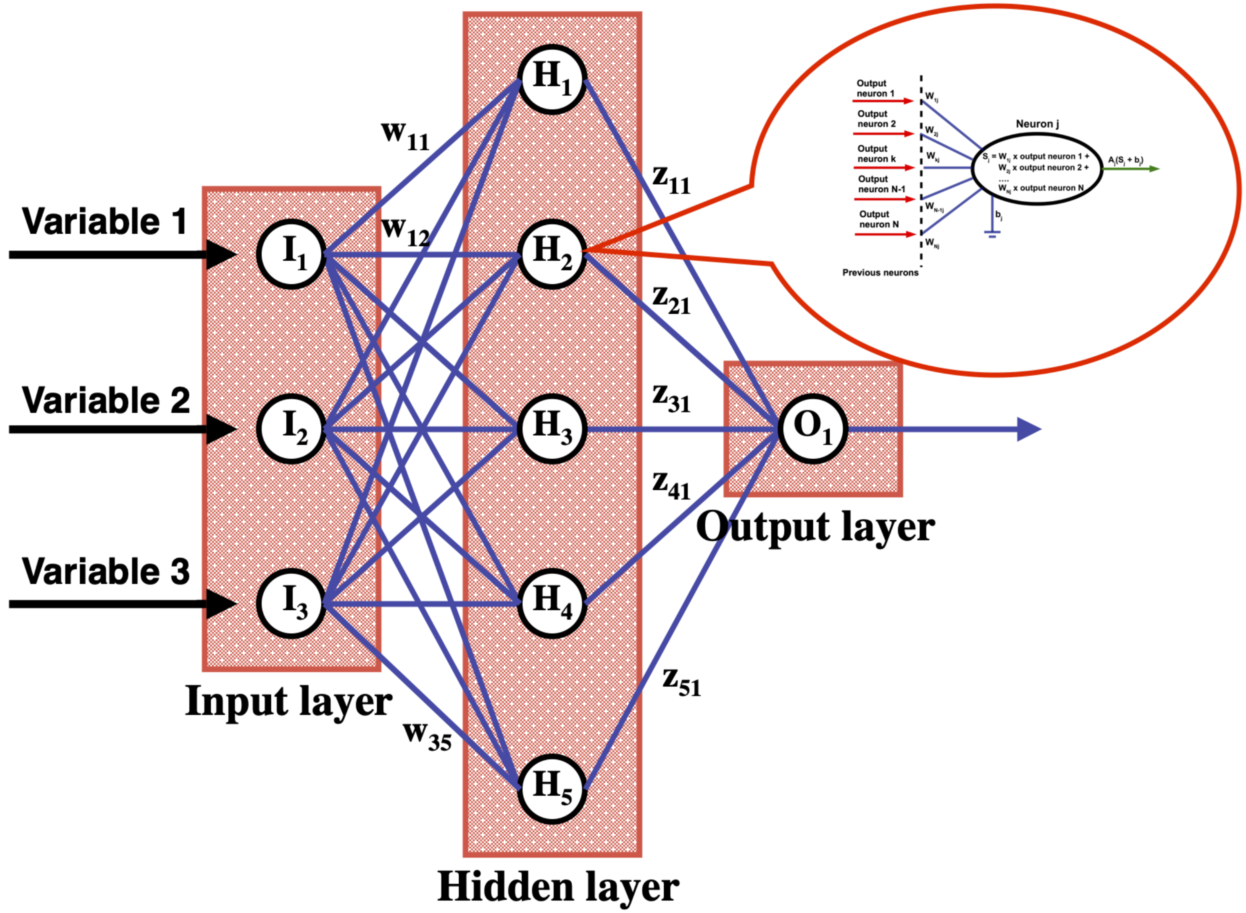
Neural Networks



- **Multilayer Perceptron (dense neural network)**
- **Network Architecture:** Hidden layers + Neurons
- **Learnable Parameters:** Weights and Biases

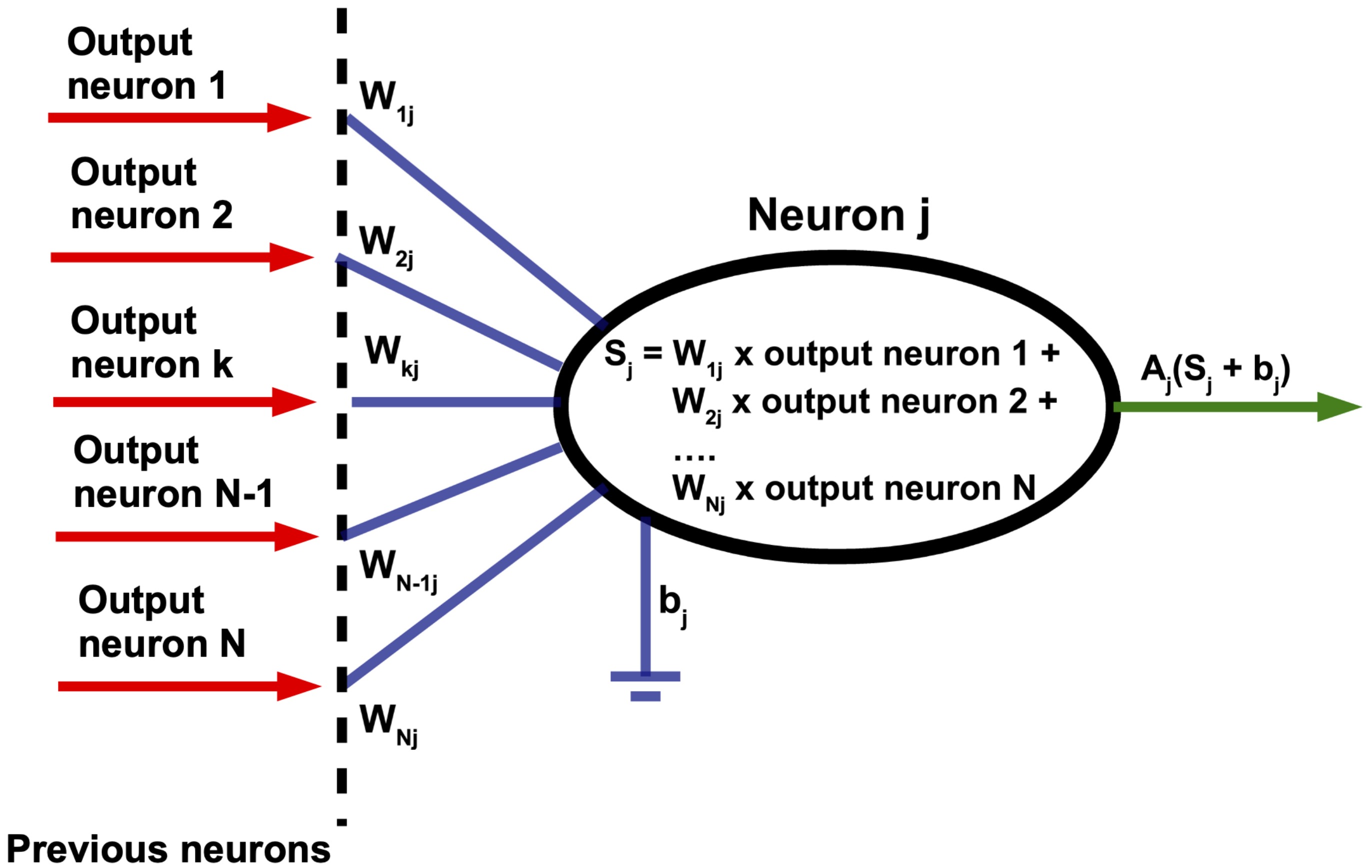
Jefferson Lab

Neural Networks



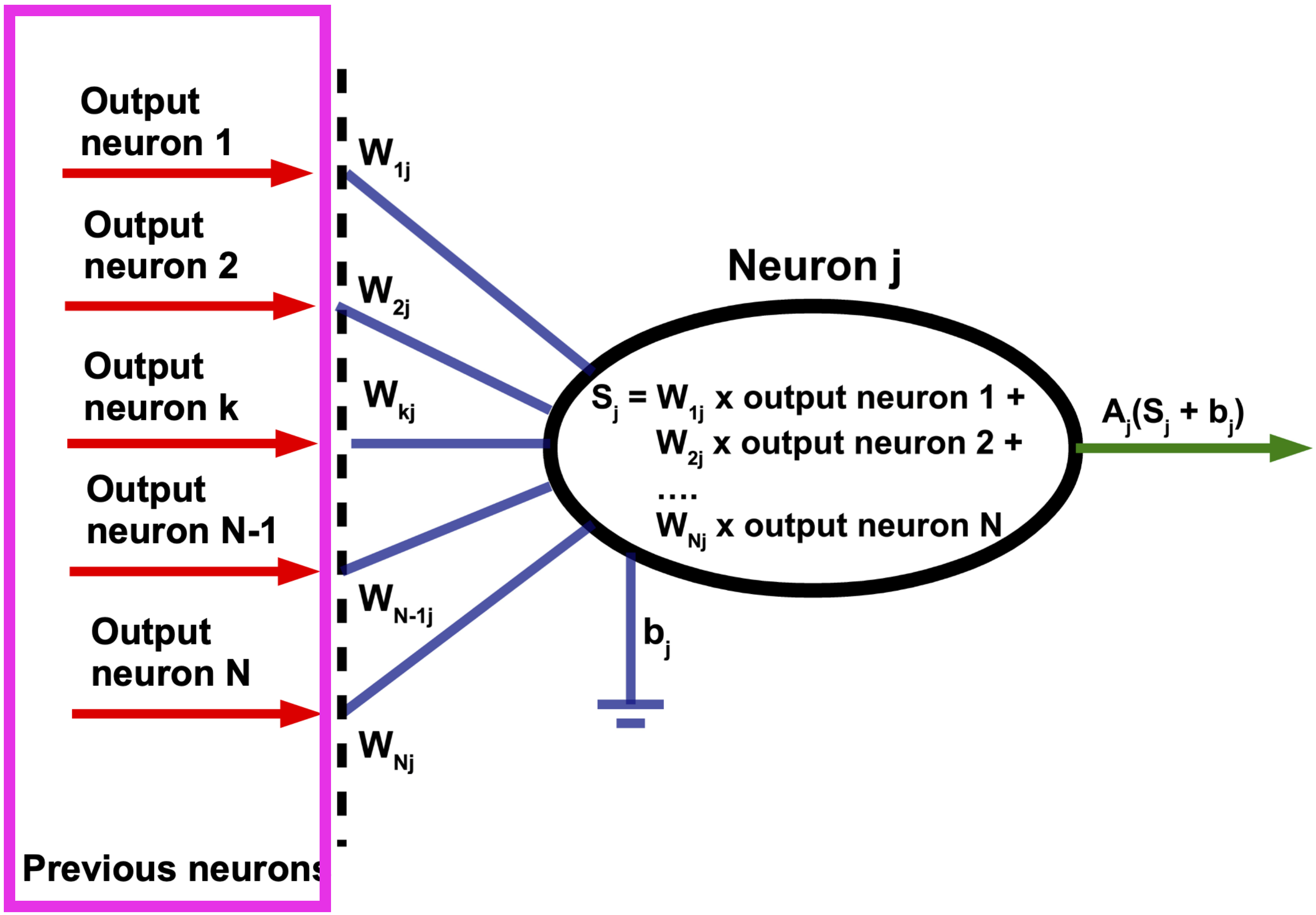
- **Multilayer Perceptron (dense neural network)**
- **Network Architecture:** Hidden layers + Neurons
- **Learnable Parameters:** Weights and Biases

A single Neuron

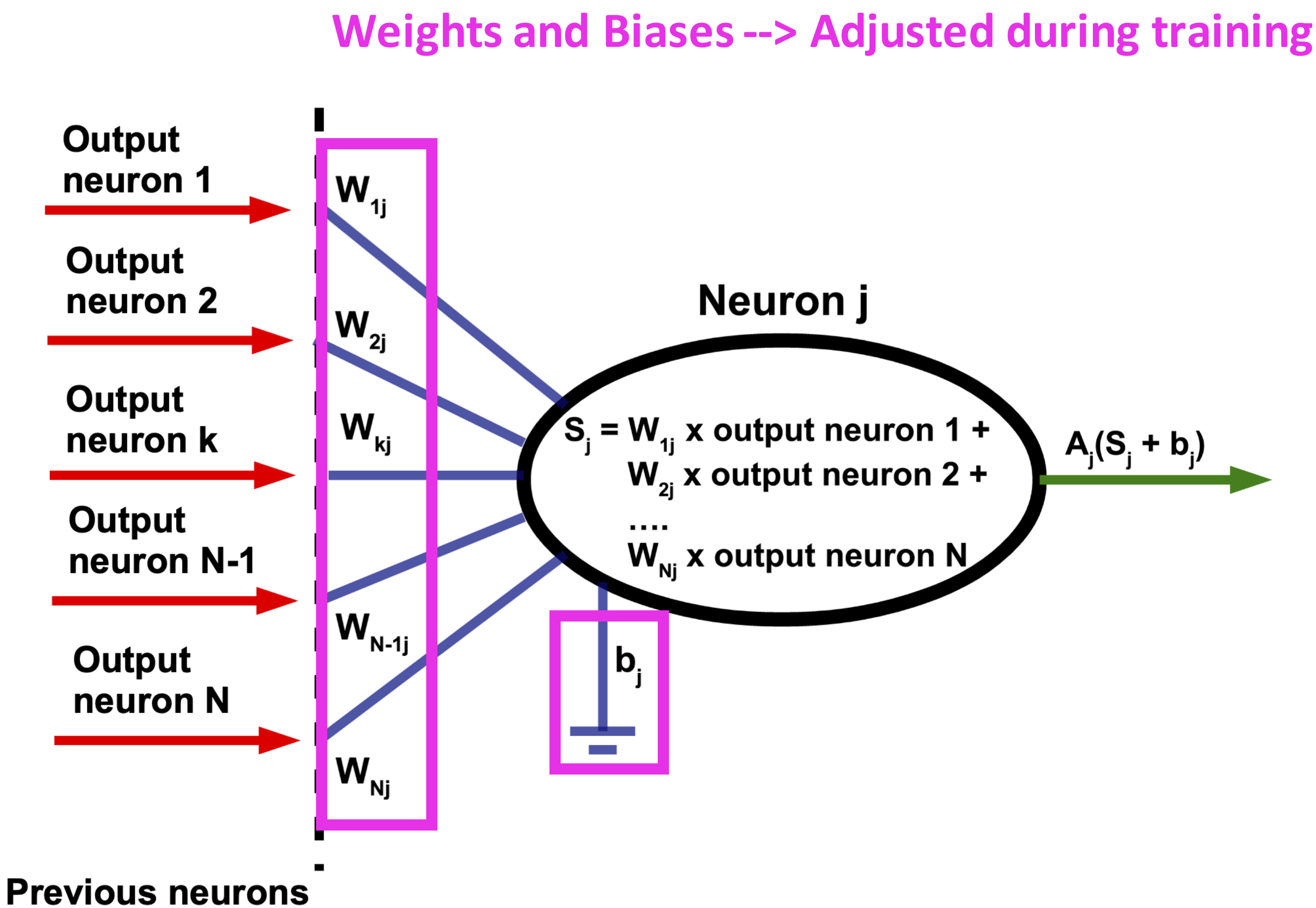


A single Neuron

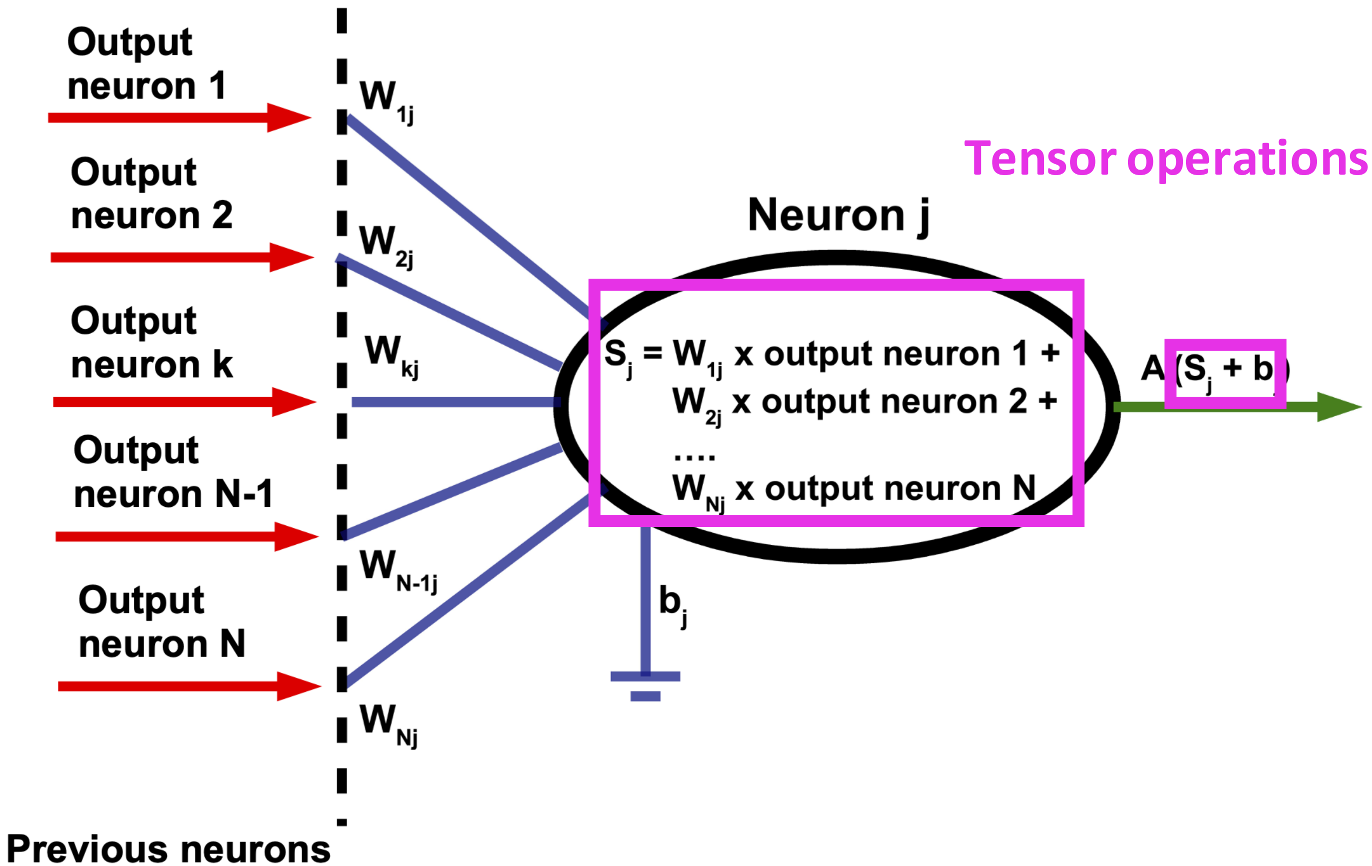
Information from previous Neurons



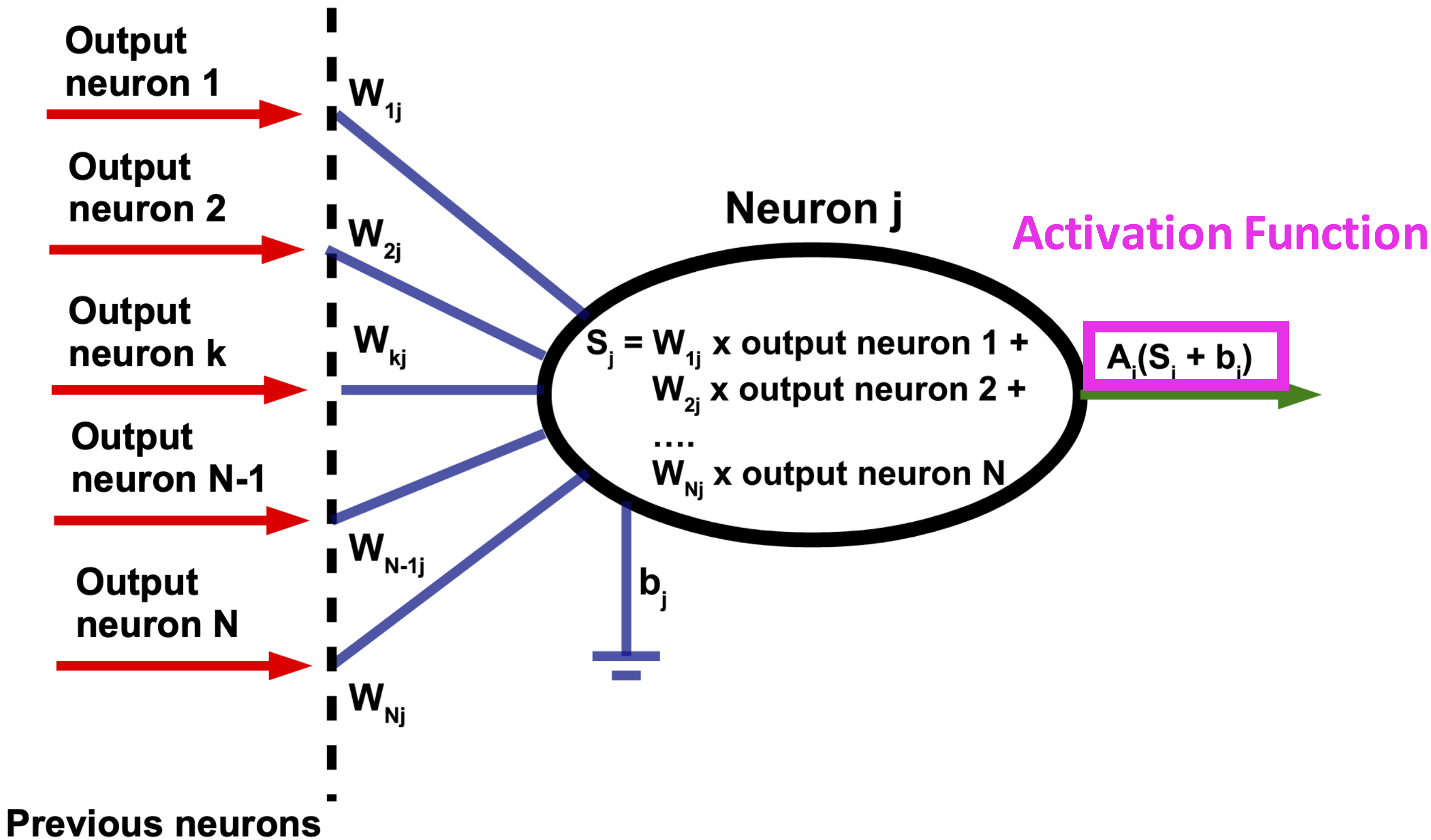
A single Neuron



A single Neuron



A single Neuron



Tensors

Rank	Shape	Dimension number	Example	
0	[]	0-D	A 0-D tensor. A scalar.	**a single number**
1	[D0]	1-D	A 1-D tensor with shape [5].	**a 5-dim vector**
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].	**a 3x4 matrix**
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].	**a cube**
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].	**I am lost…**

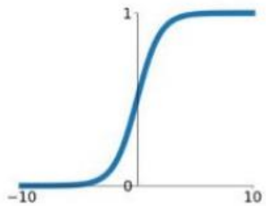
Table from tensorflow

- Forward / backward pass of data through network is expressed via tensor operations
- Weight matrix W connecting layers h and h+1
- Bias vector $\vec{b}_{h+1}$ from layer h+1
- Response from previous layer h: $\vec{S}_h$
- Get response in adjacent layer: $\vec{S}_{h+1} = W \cdot \vec{S}_h + \vec{b}_{h+1}$

Activation Functions

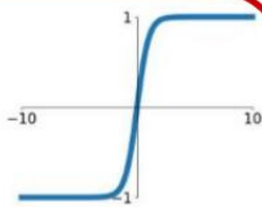
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



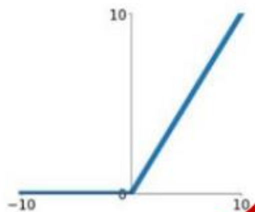
tanh

$\tanh(x)$



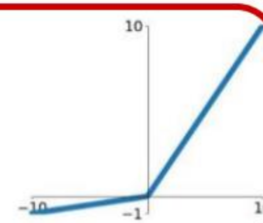
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

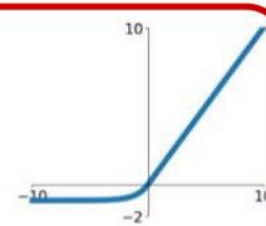


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



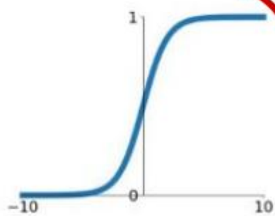
Most commonly used in modern networks as hidden layer activations

Plots taken from [Mustafa Mustafas talk at deep learning for science school 2019]

Activation Functions

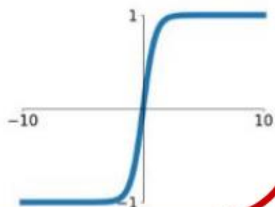
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



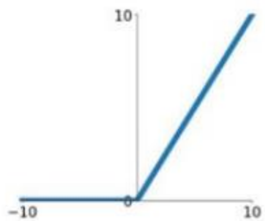
tanh

$\tanh(x)$



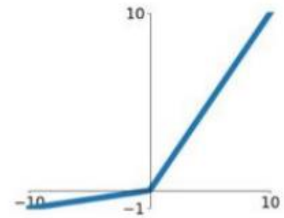
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

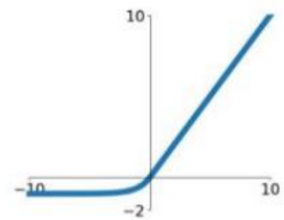


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



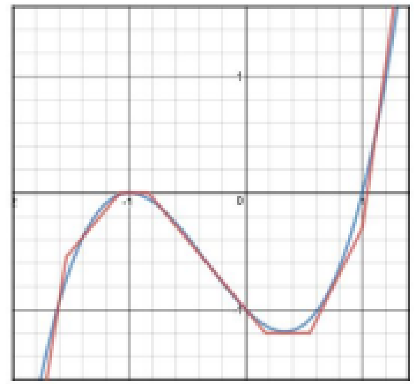
Often used for output layers

Plots taken from [Mustafa Mustafas talk at deep learning for science school 2019]

The Universal Approximation Theorem

"a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units" -- Hornik, 1991, http://zmjones.com/static/statistical-learning/hornik-nn-1991.pdf

This, of course, does not imply that we have an optimization algorithm that can find such a function. The layer could also be too large to be practical.



$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

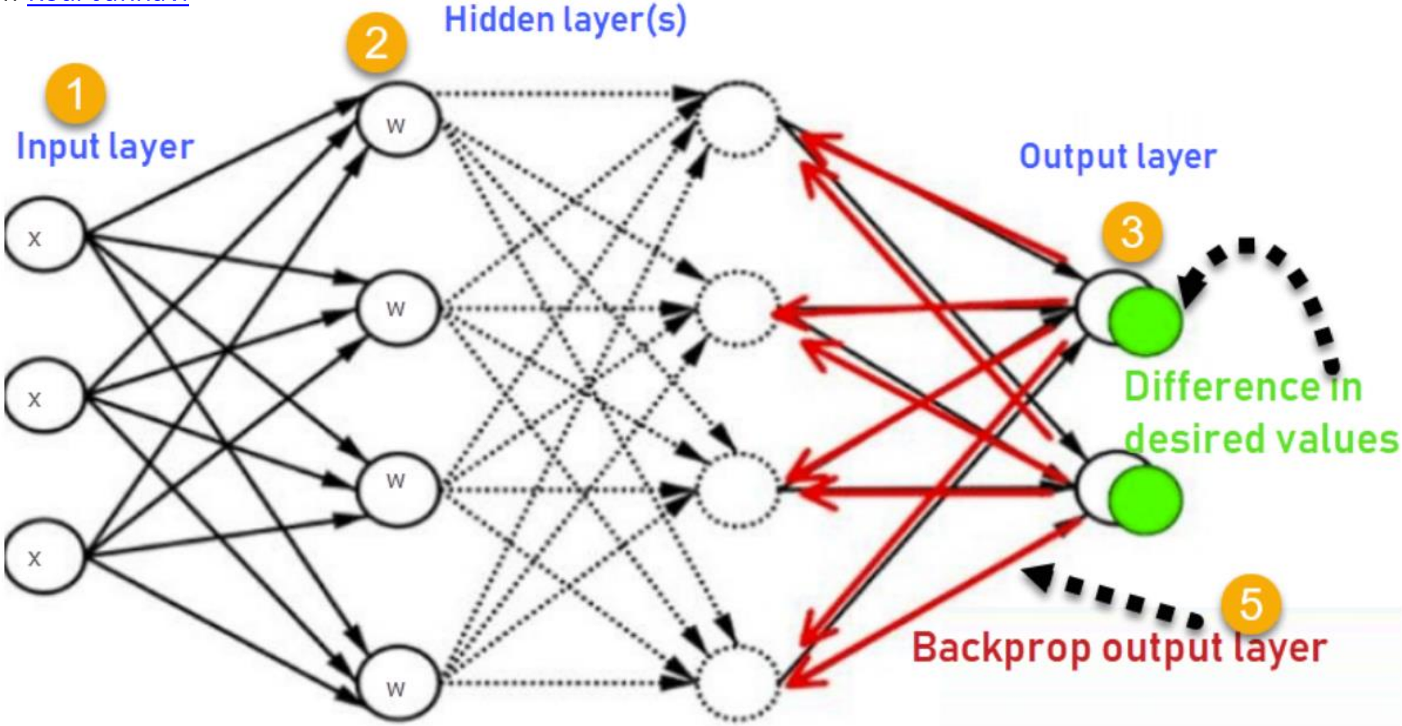
$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

Fig. credit towardsdatascience.com/can-neural-networks-really-learn-any-function-65e106617fc6

Similarly formulated by the Stone-Weierstrass-Theorem (1990): "[...] there are no nemesis functions that can not be modeled by neural networks"

Backpropagation for Neural Networks

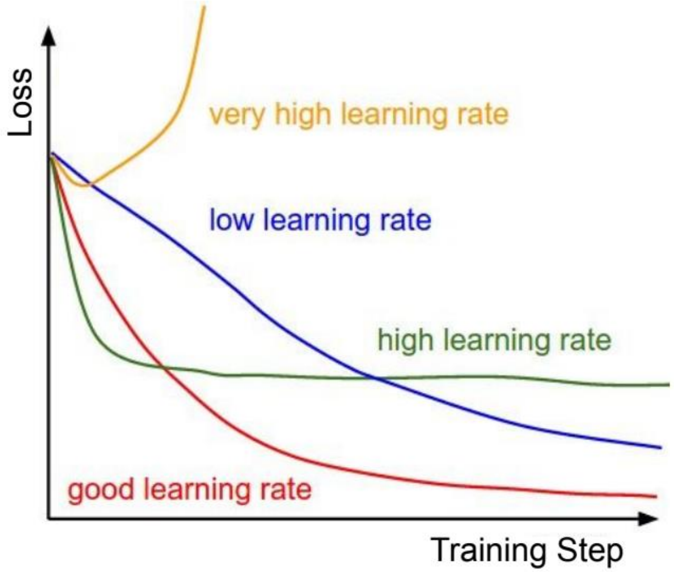
Image taken from [Kodi Jahnavi]



How Backpropagation Works

- **Forward Pass:** Pass data through network
- **Compute error**
- **Backward Pass:** Use error to update weights and biases

Parameter Updates and Loss Function



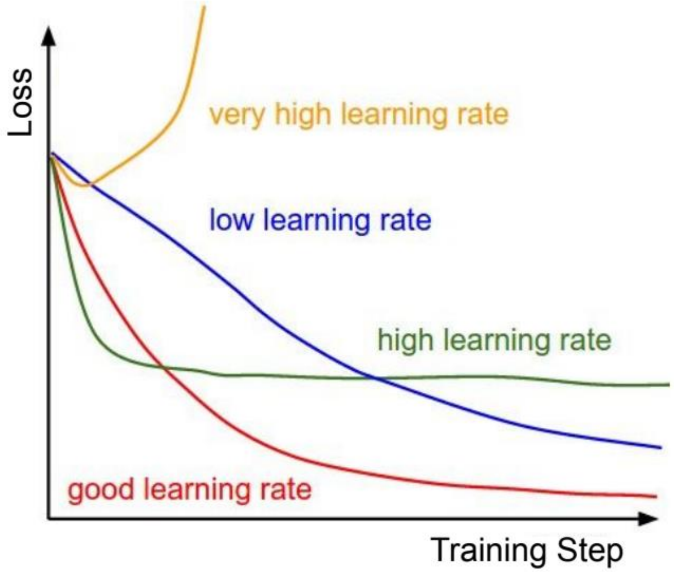
- $0 = \frac{dF(\hat{Y},Y)}{d\theta}$
- $\text{Loss} = F(\hat{Y},Y)$
- $w_{k+1} = w_k - \eta \cdot \frac{1}{m} \sum_{h=1}^{m} \nabla \text{Loss}(X_h, w_k)$
- Learning rate η, batch size m, training step k

Plots taken from [Mustafa Mustafas talk at deep learning for science school 2019]

Loss	Computation
Mean Squared Error	$\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$
Mean Absolute Error	$\frac{1}{N}\sum_{i=1}^{N}\lvert(\hat{Y}_i - Y_i)\rvert$
Binary Cross Entropy	$-\frac{1}{N}\sum_{i=1}^{N}\hat{Y}_i \log(Y_i) + (1-\hat{Y}_i)\log(1-Y_i)$

and many more…

Parameter Updates and Loss Function



- $0 = \frac{dF(\hat{Y},Y)}{d\theta}$ **Gradients are your friends!**
- $\text{Loss} = F(\hat{Y}, Y)$
- $\boxed{w_{k+1} = w_k - \eta \cdot \frac{1}{m} \sum_{h=1}^{m} \nabla \text{Loss}(X_h, w_k)}$
- Learning rate η, batch size m, training step k

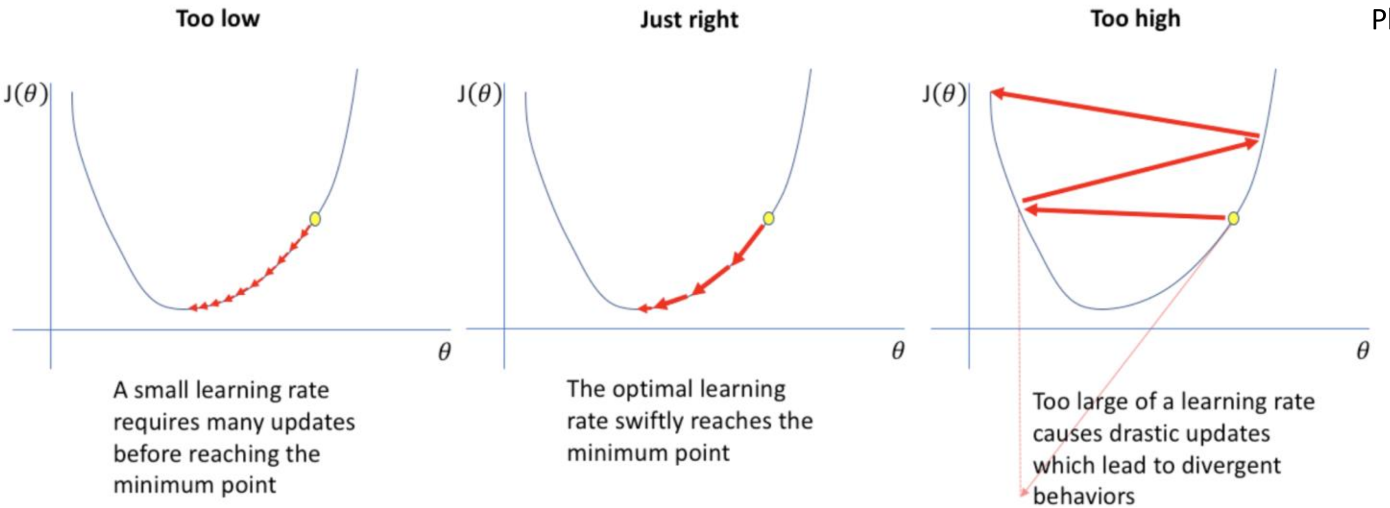
Plots taken from [Mustafa Mustafas talk at deep learning for science school 2019]

Loss	Computation
Mean Squared Error	$\frac{1}{N} \sum_{i=1}^{N} (\hat{Y}_i - Y_i)^2$
Mean Absolute Error	$\frac{1}{N} \sum_{i=1}^{N} \lvert(\hat{Y}_i - Y_i)\rvert$
Binary Cross Entropy	$-\frac{1}{N} \sum_{i=1}^{N} \hat{Y}_i \log(Y_i) + (1 - \hat{Y}_i) \log(1 - Y_i)$

and many more...

Gradient Descent and Optimizers

Plots taken from [Jeremy Jordans blog]



$$\boxed{w_{k+1} = w_k - \eta \cdot \frac{1}{m} \sum_{h=1}^{m} \nabla \mathrm{Loss}(X_h, w_k)}$$

- Gradient descent needed for convergence
- Learning rate is a crucial hyper parameter
- Variety of gradient (descent) based optimizers on the market

SGD(H_t, η_t)

$$\theta_{t+1} = \theta_t - \eta_t \nabla \ell(\theta_t)$$

MOMENTUM(H_t, η_t, γ)

$$v_0 = 0$$
$$v_{t+1} = \gamma v_t + \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta_t v_{t+1}$$

NESTEROV(H_t, η_t, γ)

$$v_0 = 0$$
$$v_{t+1} = \gamma v_t + \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - \eta_t \left(\gamma v_{t+1} + \nabla \ell(\theta_t)\right)$$

RMSPROP$(H_t, \eta_t, \gamma, \rho, \epsilon)$

$$v_0 = 1, m_0 = 0$$
$$v_{t+1} = \rho v_t + (1-\rho) \nabla \ell(\theta_t)^2$$
$$m_{t+1} = \gamma m_t + \frac{\eta_t}{\sqrt{v_{t+1} + \epsilon}} \nabla \ell(\theta_t)$$
$$\theta_{t+1} = \theta_t - m_{t+1}$$

ADAM$(H_t, \alpha_t, \beta_1, \beta_2, \epsilon)$

$$m_0 = 0, v_0 = 0$$
$$m_{t+1} = \beta_1 m_t + (1-\beta_1) \nabla \ell(\theta_t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2) \nabla \ell(\theta_t)^2$$
$$b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}$$
$$\theta_{t+1} = \theta_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon} b_{t+1}$$

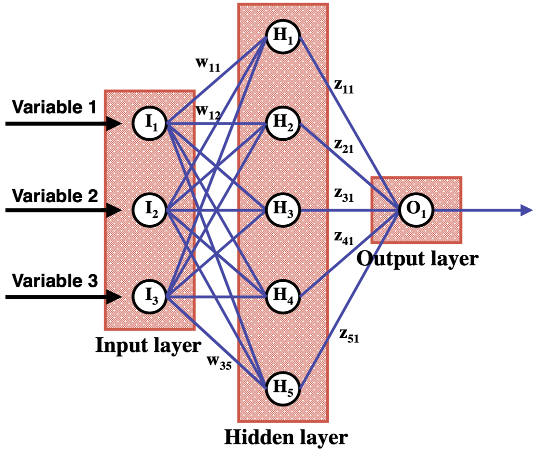
NADAM$(H_t, \alpha_t, \beta_1, \beta_2, \epsilon)$

$$m_0 = 0, v_0 = 0$$
$$m_{t+1} = \beta_1 m_t + (1-\beta_1) \nabla \ell(\theta_t)$$
$$v_{t+1} = \beta_2 v_t + (1-\beta_2) \nabla \ell(\theta_t)^2$$
$$b_{t+1} = \frac{\sqrt{1-\beta_2^{t+1}}}{1-\beta_1^{t+1}}$$
$$\theta_{t+1} = \theta_t - \alpha_t \frac{\beta_1 m_{t+1} + (1-\beta_1) \nabla \ell(\theta_t)}{\sqrt{v_{t+1}} + \epsilon} b_{t+1}$$

Taken from [On Empirical Comparisons of Optimizers for Deep Learning]

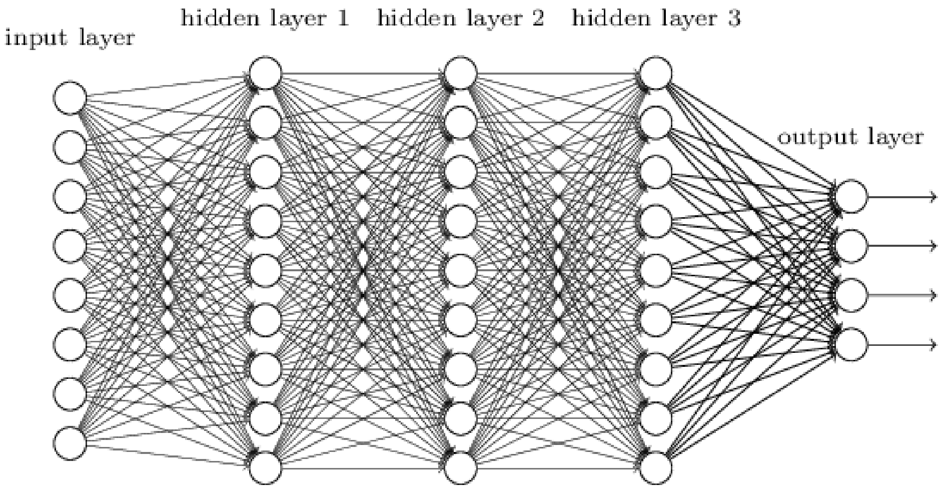
Now what is Deep Learning ?

Machine Learning



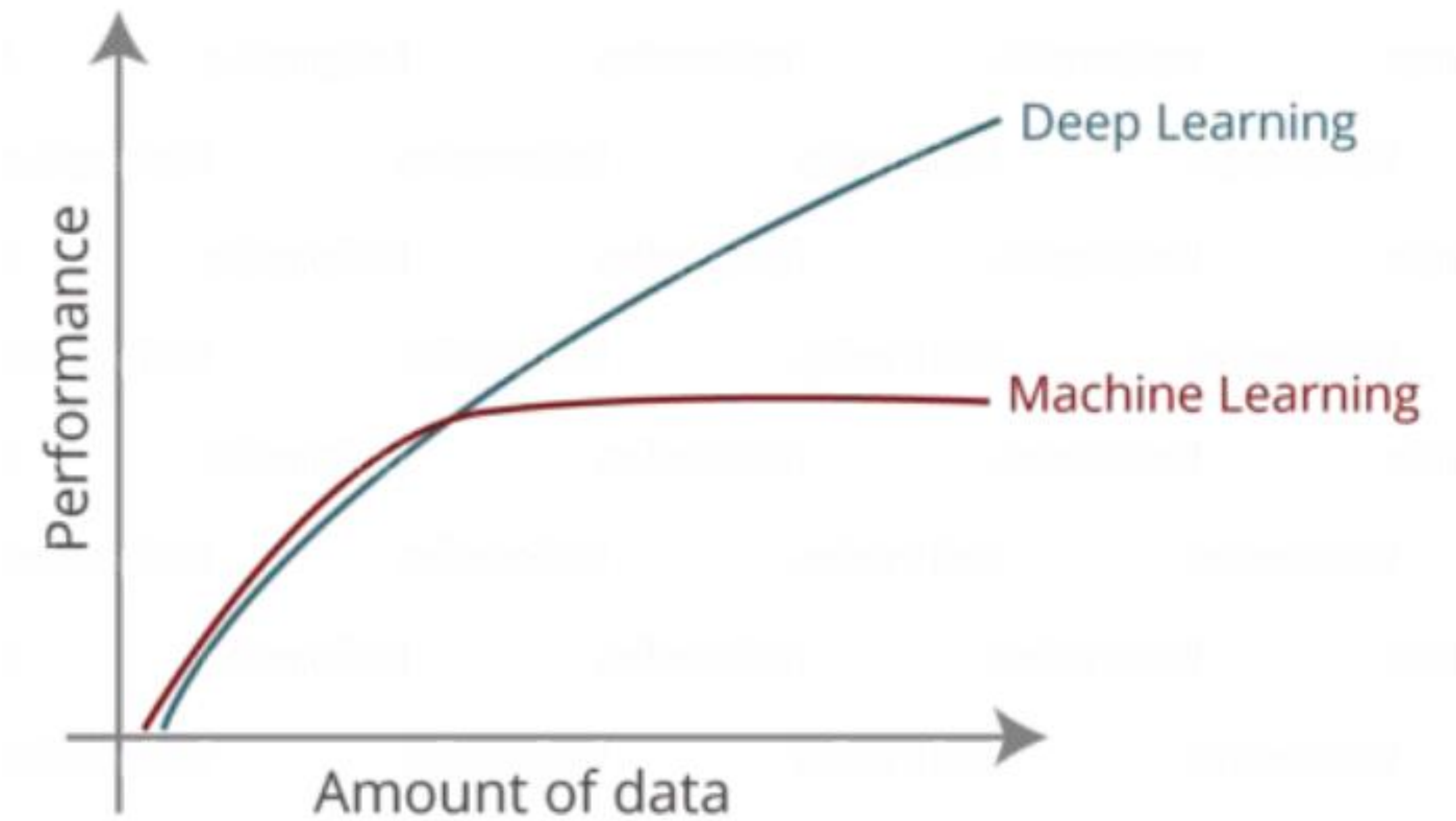
- Variety of algorithms
- Multilayer perceptrons < 3 hidden layers
- Decision trees
- Linear classifier
- ...

Deep Learning



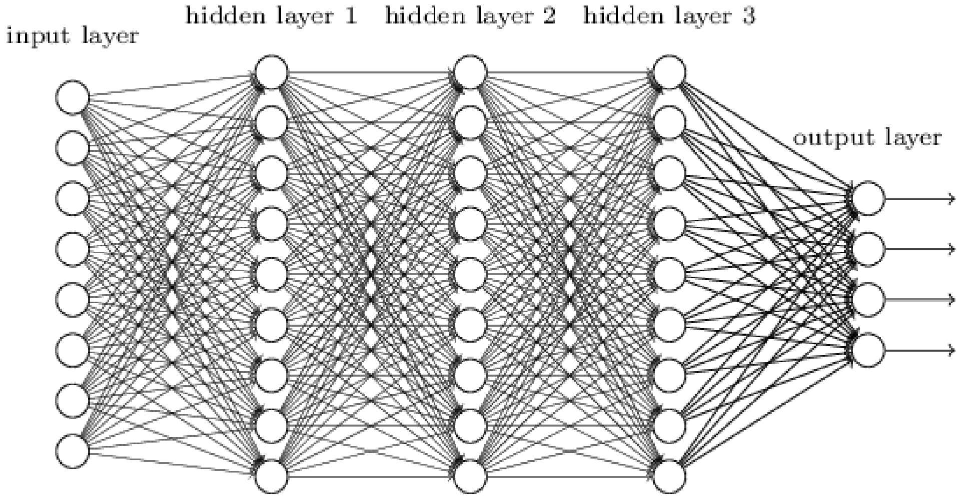
- Large neural networks
- Multilayer perceptrons >= 3 hidden layers
- Convolutional neural networks (computer vision)
- Graph neural networks
- Language models (Chat GPT)
- ....

Why Deep Learning ?



Plot taken from [Mustafa Mustafas talk at deep learning for science school 2019]

Challenges in Deep Learning



Need gradients for weight updates

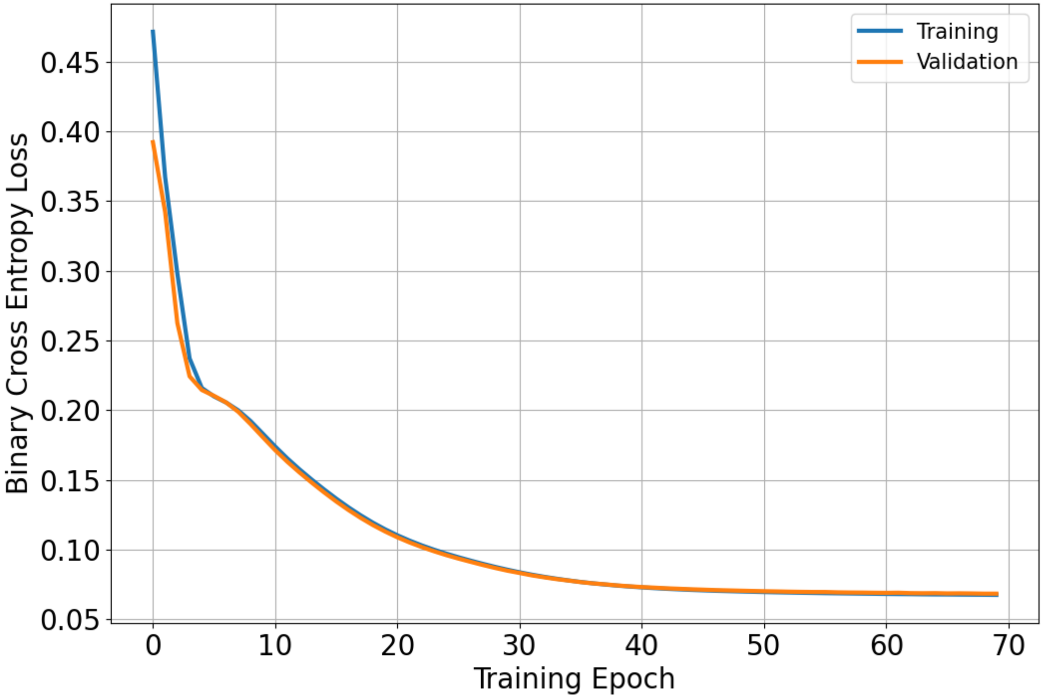
$$w_{k+1} = w_k - \eta \cdot \frac{1}{m} \sum_{h=1}^{m} \nabla \mathrm{Loss}(X_h, w_k)$$

No gradients, no updates

$$\nabla \mathrm{Loss} = 0 \Rightarrow w_{k+1} = w_k$$

- Computationally intensive --> Many algebraic operations --> **Utilize GPUs**
- Vanishing gradient problem --> Zero gradients --> No weight updates
- Overfitting --> So many parameters
- Larger models (e.g. Chat GPT) require distributed training across multiple GPUs

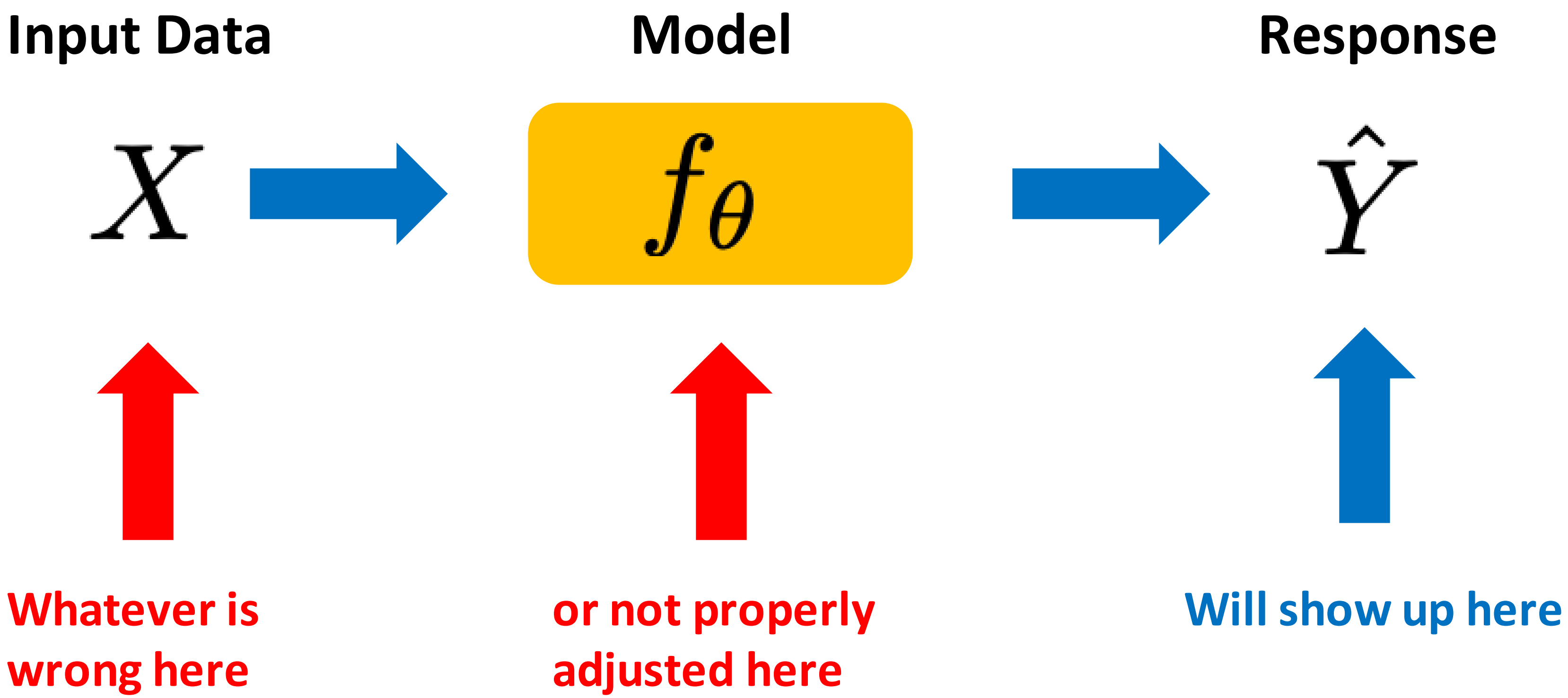
Training a Neural Network for our Classification Problem



- 75% of data used for training
- 25% of data used for validation
- Trained for 70 epochs
- Loss converged to some value --> Is this good or bad?

Hyper Parameter	Setting
Architecture	2 hidden layers with 20 neurons each
Activation Functions	tanh for hidden layers and sigmoid for output layer
Learning Rate	1e-4
Batch Size	128

Model Evaluation / Analysis (1)



==> Need to evaluate model AFTER training

Model Evaluation / Analysis (2)

Input Data **Model** **Response**

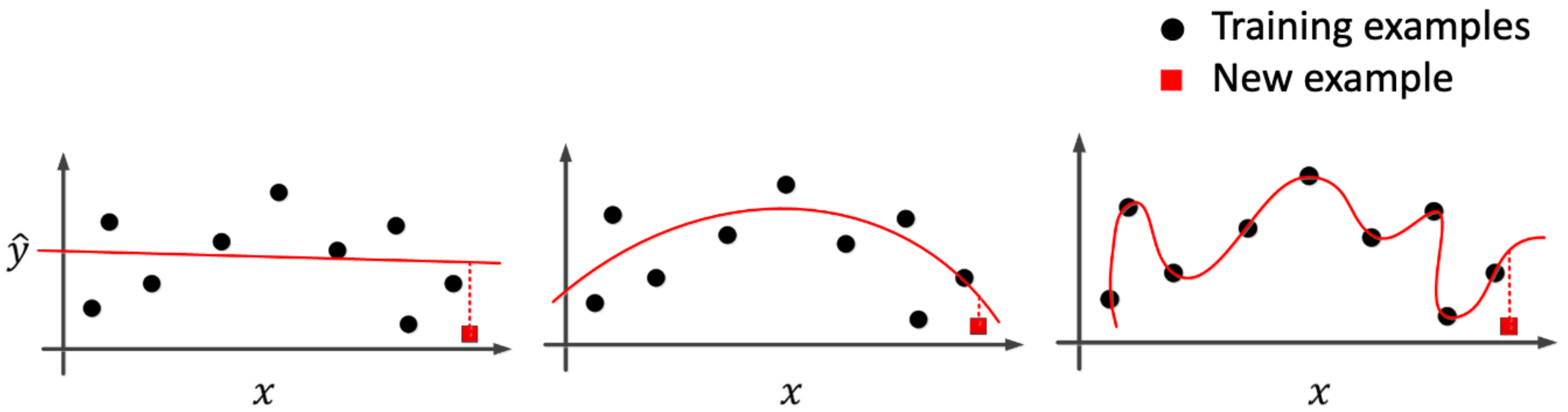
$$X \longrightarrow \boxed{f_\theta} \longrightarrow \hat{Y}$$

- **Core idea:** Compare model response to known truth Y
- Could use the loss function
  - Single value only
  - Helps to understand training progress
  - Does not tell how well model generalizes
- Perform model evaluation on separate (validation) data set
  - Data NOT used for training
  - Check how model performs on "unknown" data set --> Generalizability

Model Evaluation / Analysis (3)

Input Data **Model** **Response**

$$X \rightarrow \boxed{f_\theta} \rightarrow \hat{Y}$$



Underfitting Good fit! Overfitting

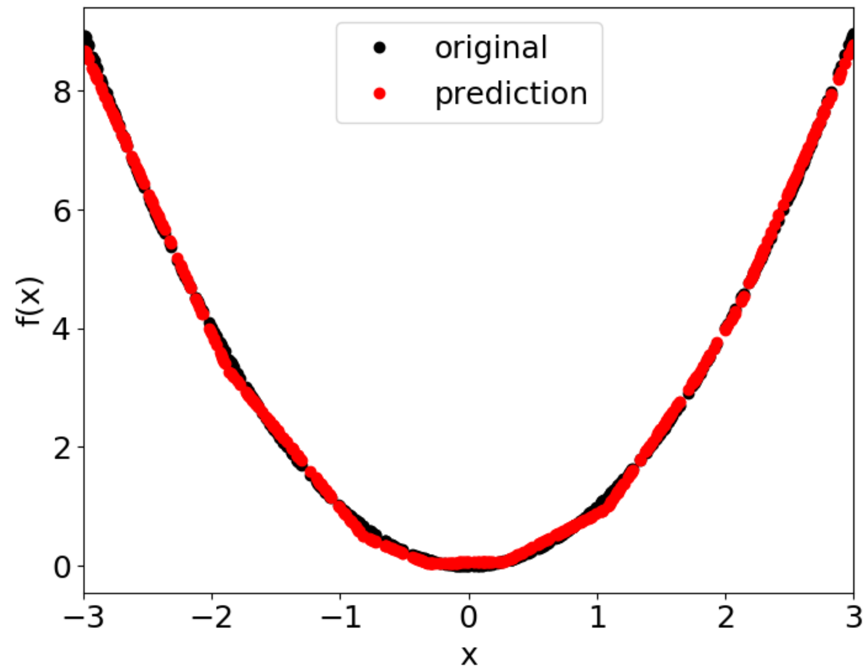
Plot taken from [Brenda Ngs talk at deep learning for science school 2019]

Evaluation Metrics

- Depend on the underlying problem that you are trying to solve (regression vs. classification)
- **Regression:**
  - Chi-Square
  - Mean Squared Error
  - Likelihood
  - Model response
  - ...
- **Classification:**
  - Confusion matrix
  - ROC-Curve
  - Accuracy
  - Model response
  - ...

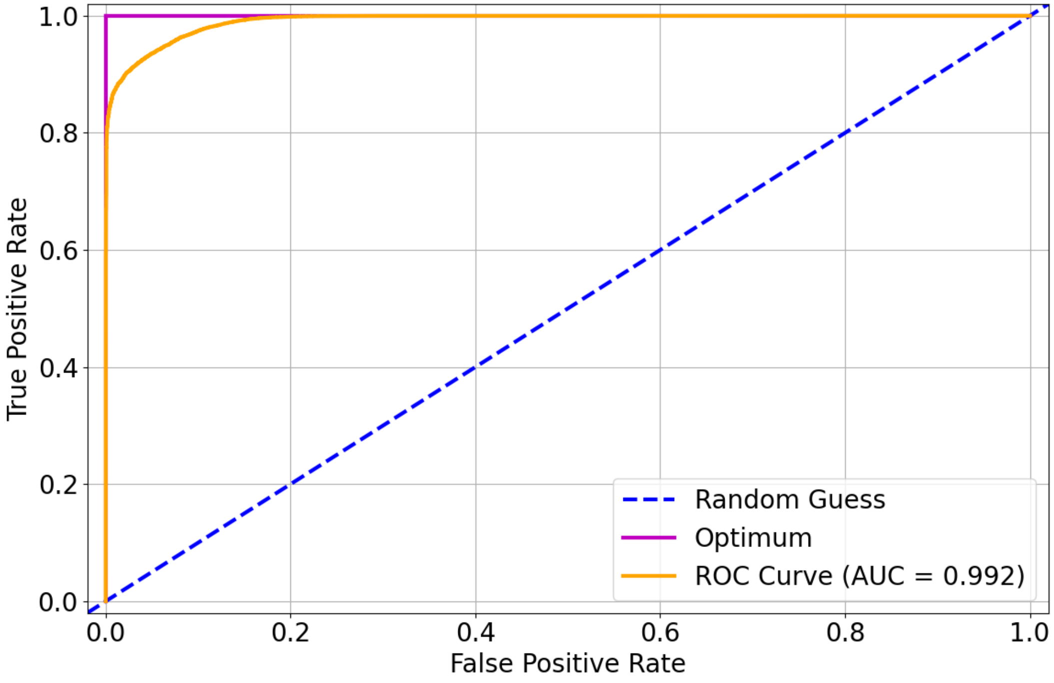
Evaluation Metrics

- Depend on the underlying problem that you are trying to solve (regression vs. Classification)
- **Regression:**
  - Chi-Square
  - Mean Squared Error
  - Likelihood
  - Model response
  - ...
- **Classification:**
  - Confusion matrix
  - ROC-Curve
  - Accuracy
  - Model response
  - ...



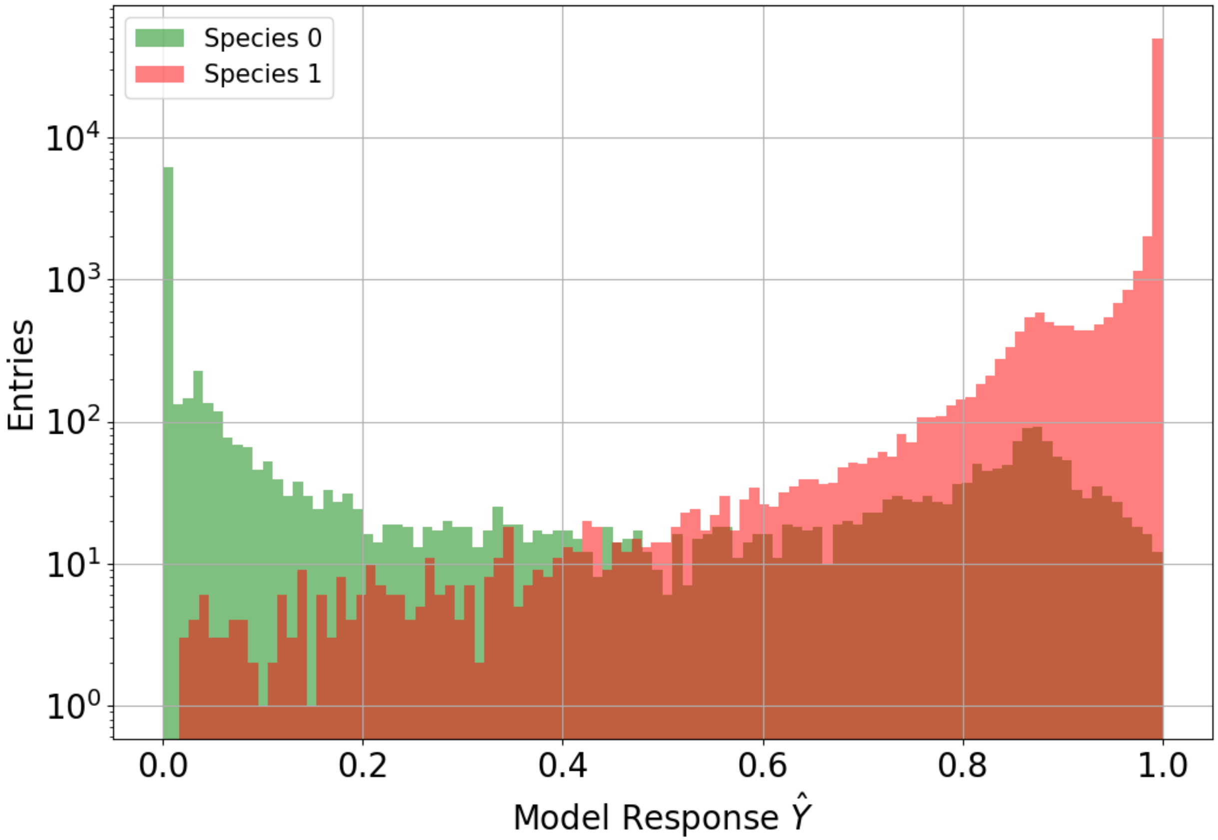
Evaluation Metrics

- Depend on the underlying problem that you are trying to solve (regression vs. Classification)
- **Regression:**
  - Chi-Square
  - Mean Squared Error
  - Likelihood
  - Model response
  - ...
- **Classification:**
  - Confusion matrix
  - ROC-Curve
  - Accuracy
  - Model response
  - ...



Evaluating a Binary Classifier: Model Response

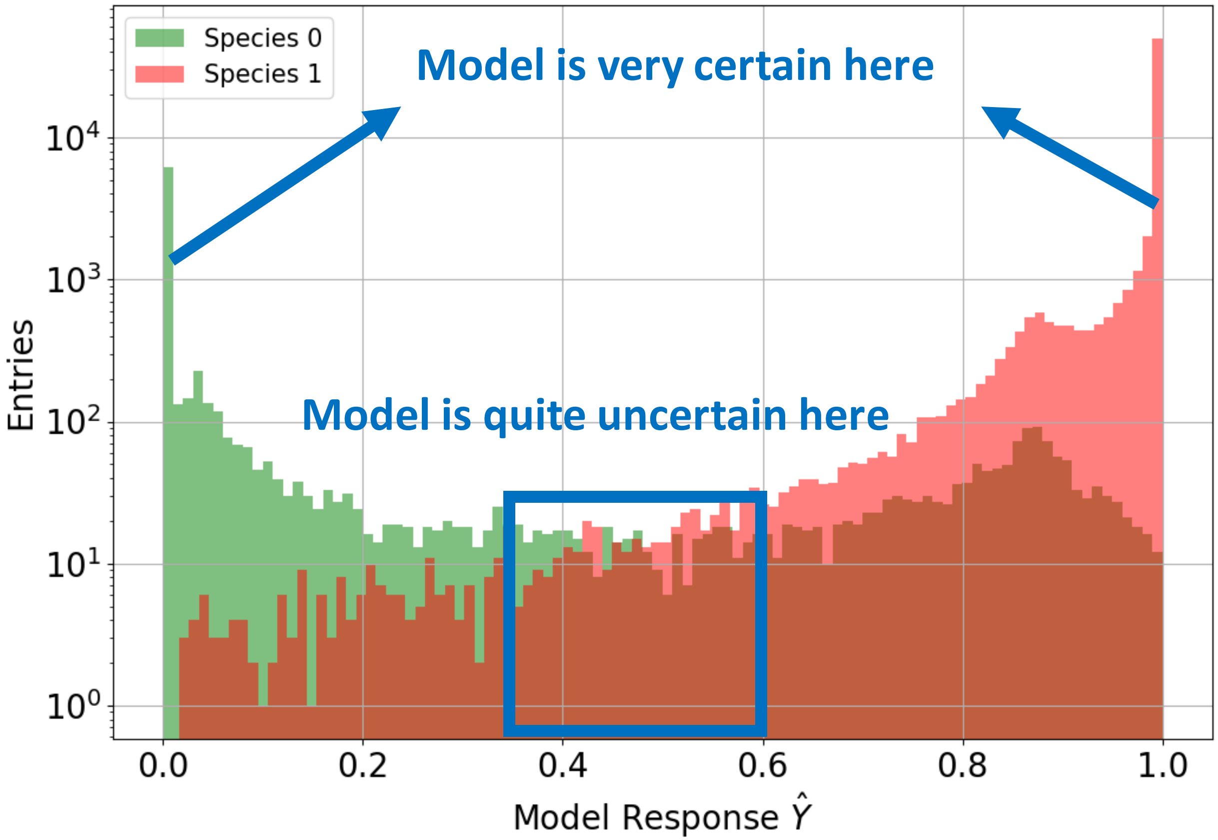
- Check model response on validation data
- Response is continuous, but Y is discrete
- Need a function to discretize continuous values
- Model response plots are the first things to check !



Jefferson Lab

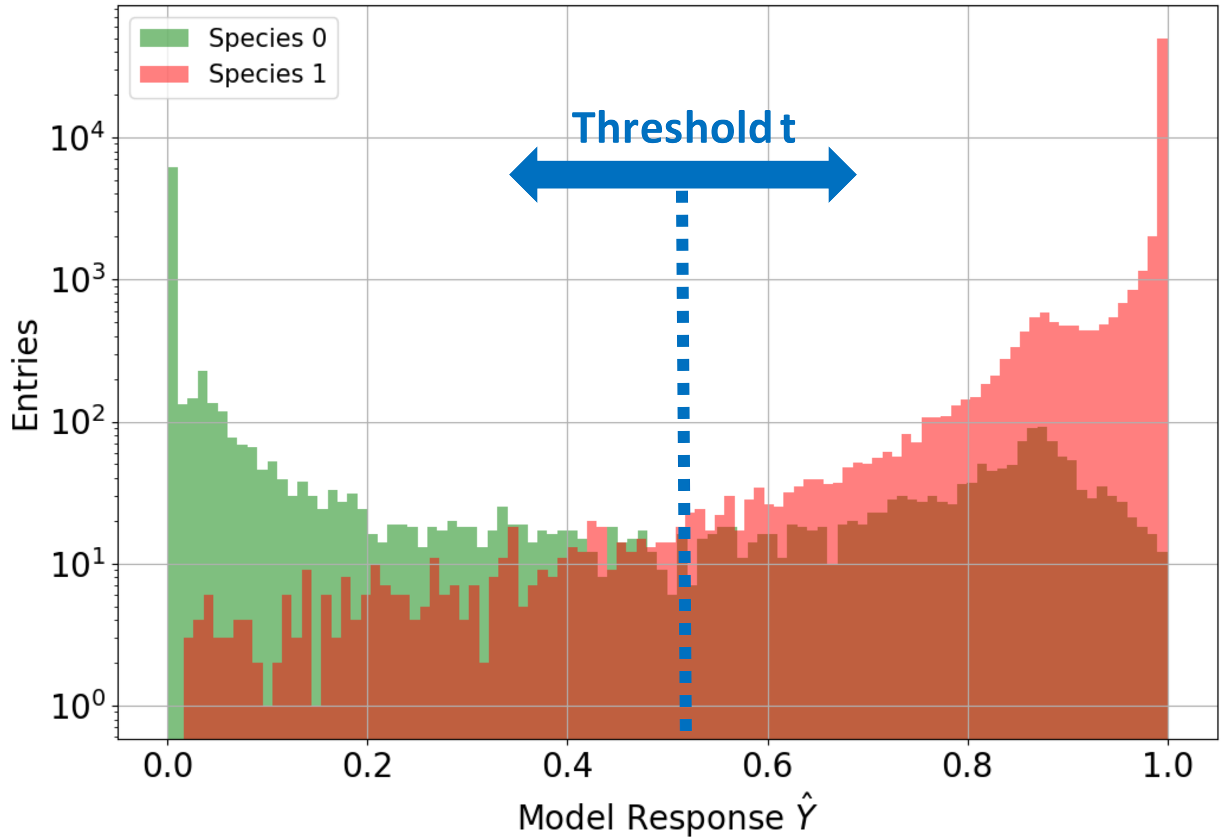
Evaluating a Binary Classifier: Model Response

- Check model response on validation data
- Response is continuous, but Y is discrete
- Need a function to discretize continuous values
- Model response plots are the first things to check !



Evaluating a Binary Classifier: Model Response

- Check model response on validation data
- Response is continuous, but Y is discrete
- Need a function to discretize continuous values
- Model response plots are the first things to check !



$$\text{Predicted Label } \hat{\ell} = \Theta(\hat{Y}, t) = \begin{cases} 1, \hat{Y} \geq t, \\ 0, \hat{Y} < t \end{cases}$$

Evaluating a Binary Classifier: Counting

$$\text{True Positives TP} = \text{All events correctly identified as species 1}$$

$$\text{False Positives FP} = \text{All events falselyidentified as species 1}$$

$$\text{True Negatives TN} = \text{All events correctly identified as species 0}$$

$$\text{False Negatives FN} = \text{All events falsely identified as species 0}$$

Nearly all evaluation metrics for binary classification are derived from these quantities!

Evaluating a Binary Classifier: Counting

$$\text{True Positives TP} == \sum_{i=1}^{N_1} \delta(\ell_i - 1) \cdot \delta(\hat{\ell}_i - 1)$$

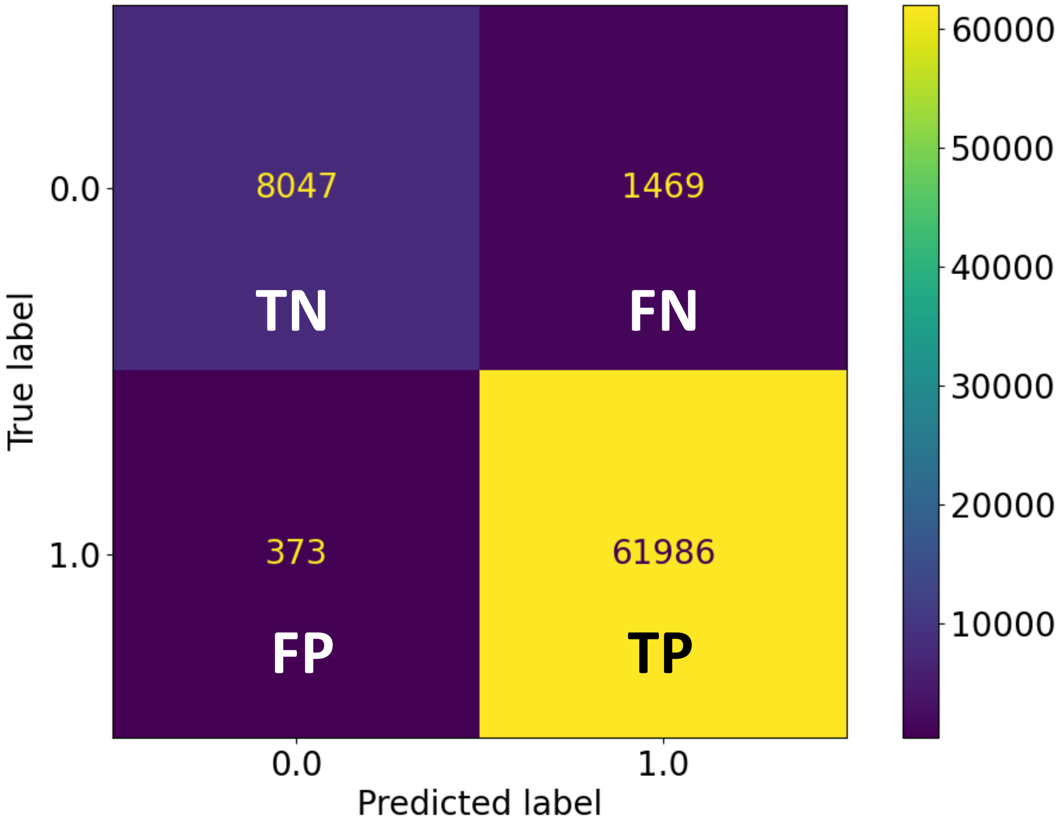
$$\text{False Positives FP} == \sum_{i=1}^{N_0} \delta(\ell_i) \cdot \delta(\hat{\ell}_i - 1)$$

$$\text{True Negatives TN} == \sum_{i=1}^{N_0} \delta(\ell_i) \cdot \delta(\hat{\ell}_i)$$

$$\text{False Negatives FN} == \sum_{i=1}^{N_1} \delta(\ell_i - 1) \cdot \delta(\hat{\ell}_i)$$

Nearly all evaluation metrics for binary classification are derived from these quantities!

Evaluating a Binary Classifier: Confusion Matrix and Balanced Accuracy



- Confusion matrix summarizes performance for given threshold t (here: t=0.5)
- Diagonal: True Identification
- Off-Diagonal: False Identification
- Ideal classifier: Diagonal ~ 1.0 and Off-Diagonal ~ 0.0
- Used balanced accuracy for imbalanced data set

$$\text{Balanced Accuracy} = \frac{1}{2} \cdot \left(\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}} \right)$$

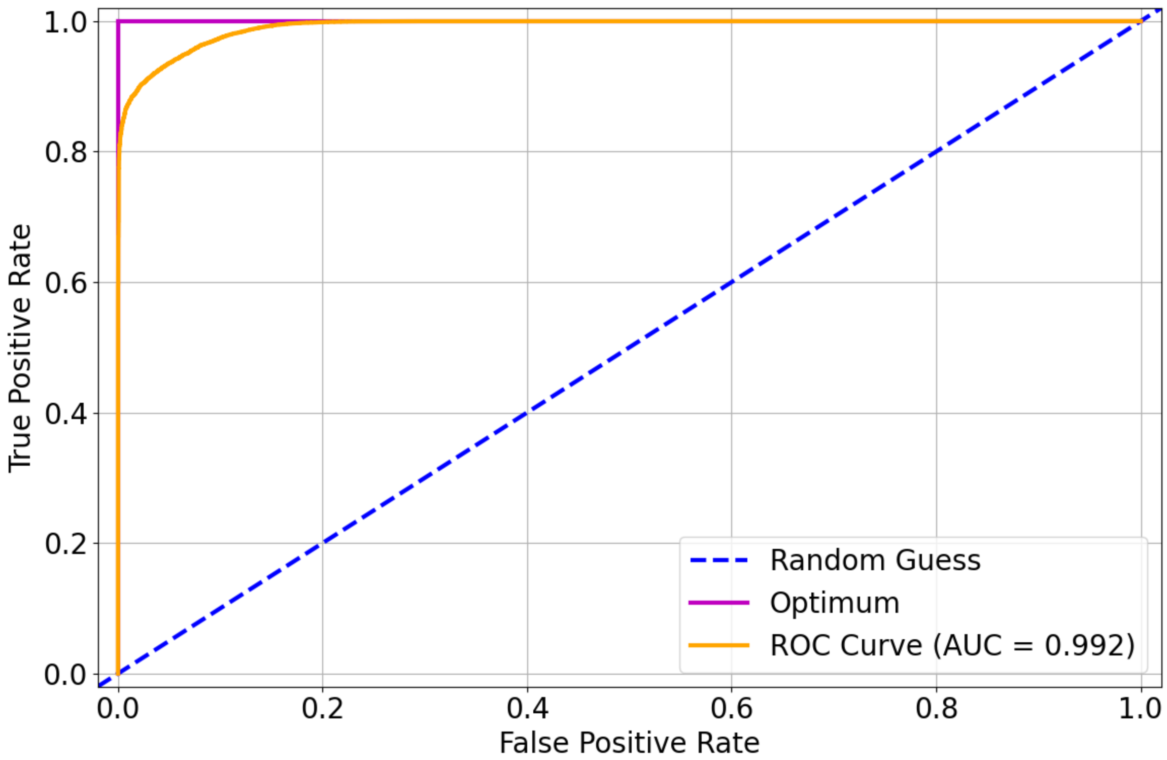
Observed for our model: Balanced Accuracy ~ 92%

Evaluating a Binary Classifier: ROC-Curve

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP+TN}}$$

- Run scan over threshold t
- Compute rates for each t
- Plot TPR vs. FPR for all scans
- AUC = Area Under Curve (Ideally = 1.0)

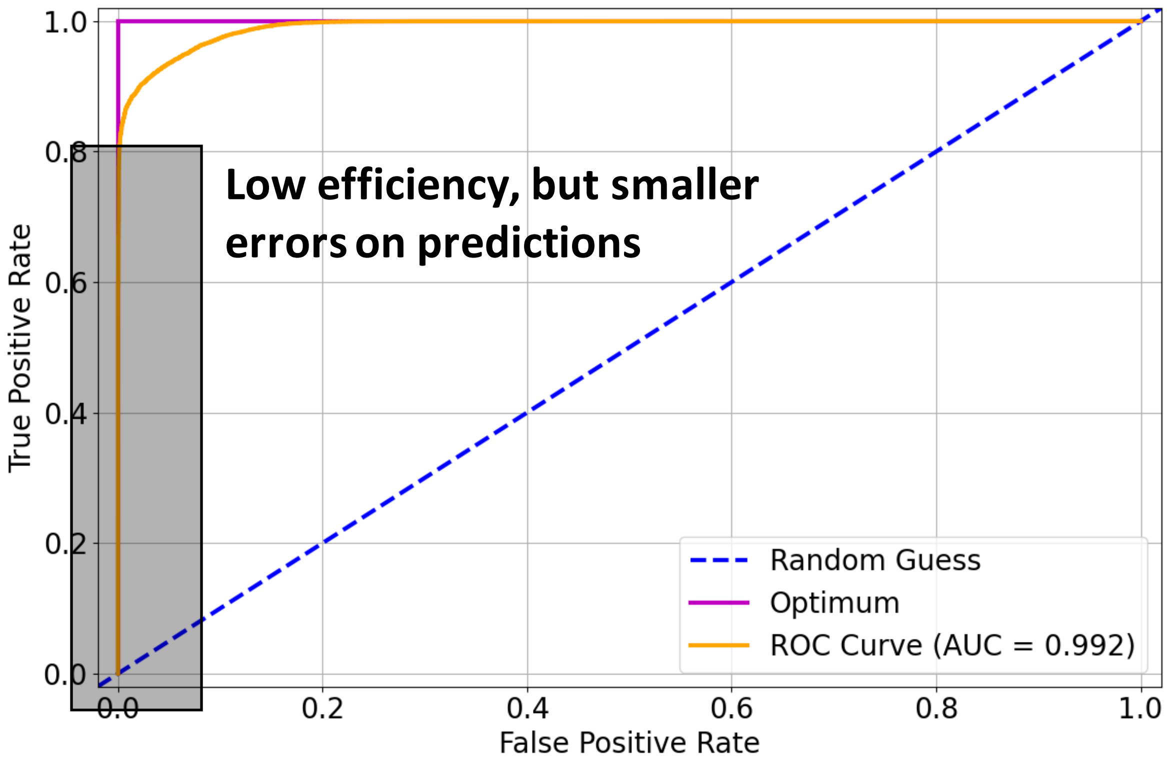


Evaluating a Binary Classifier: ROC-Curve

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP+TN}}$$

- Run scan over threshold t
- Compute rates for each t
- Plot TPR vs. FPR for all scans
- AUC = Area Under Curve
  (Ideally = 1.0)

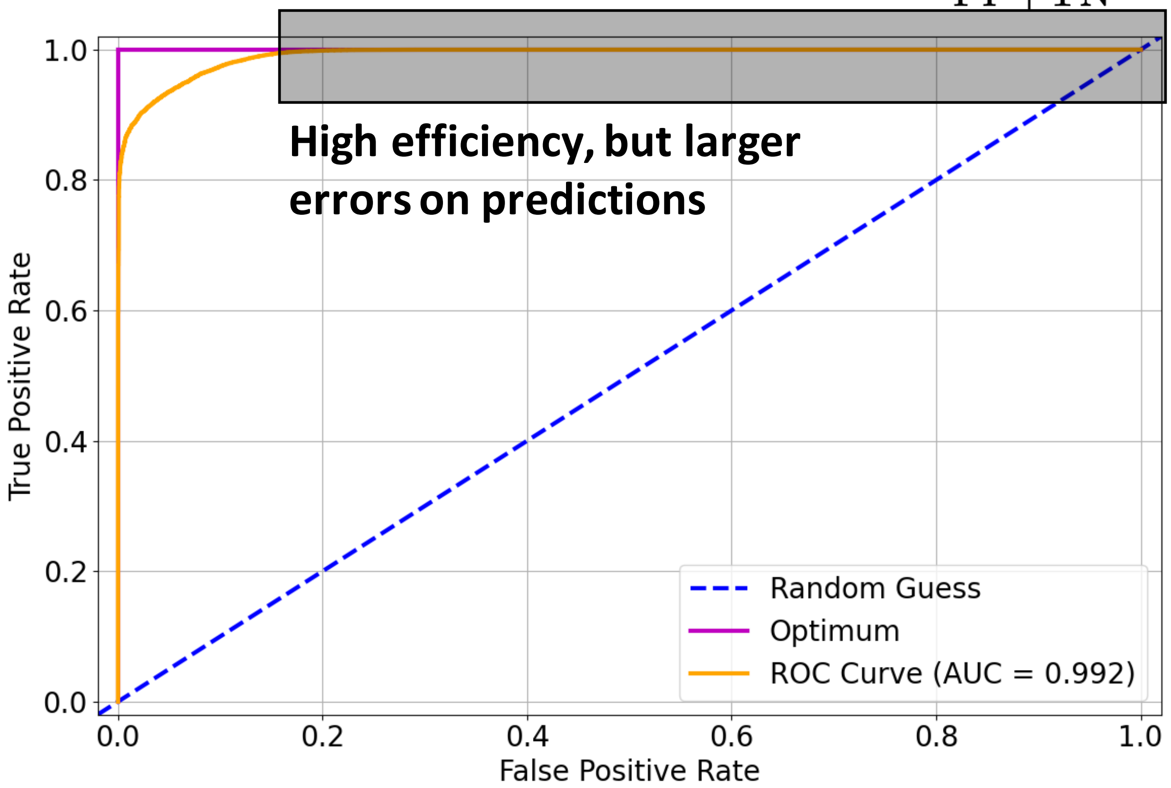


Evaluating a Binary Classifier: ROC-Curve

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP+TN}}$$

- Run scan over threshold t
- Compute rates for each t
- Plot TPR vs. FPR for all scans
- AUC = Area Under Curve (Ideally = 1.0)



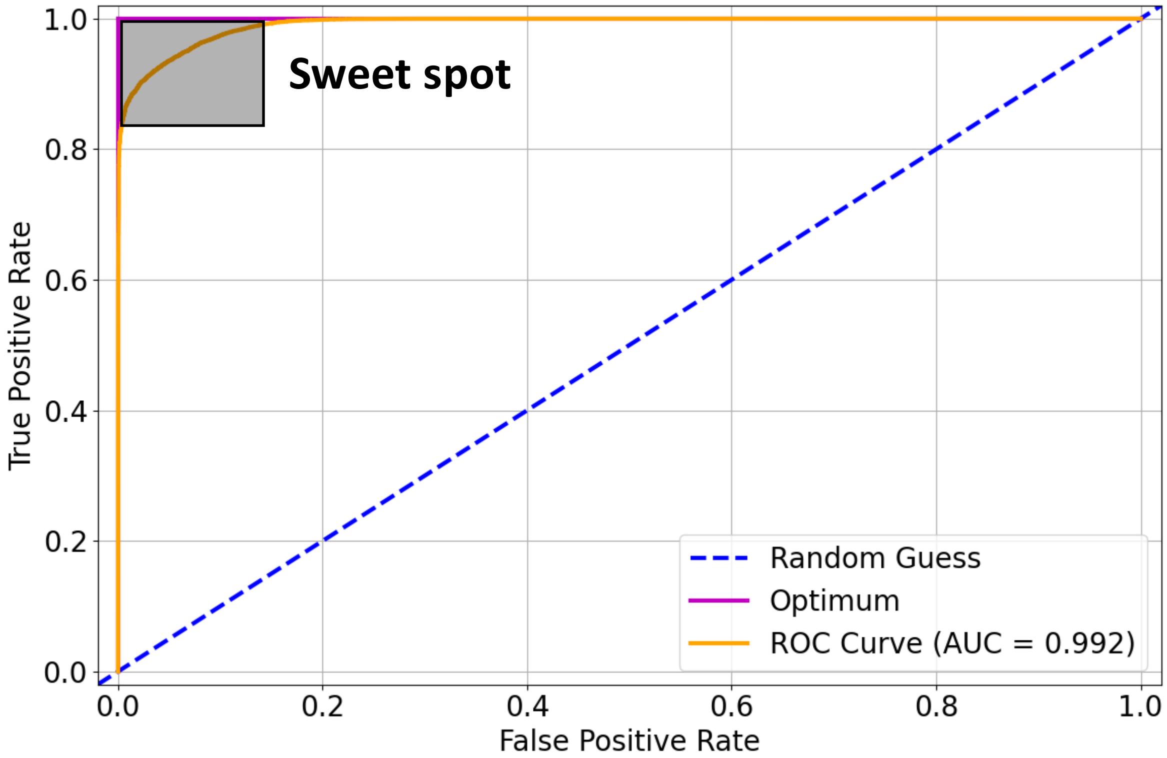
Jefferson Lab

Evaluating a Binary Classifier: ROC-Curve

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP+TN}}$$

- Run scan over threshold t
- Compute rates for each t
- Plot TPR vs. FPR for all scans
- AUC = Area Under Curve (Ideally = 1.0)

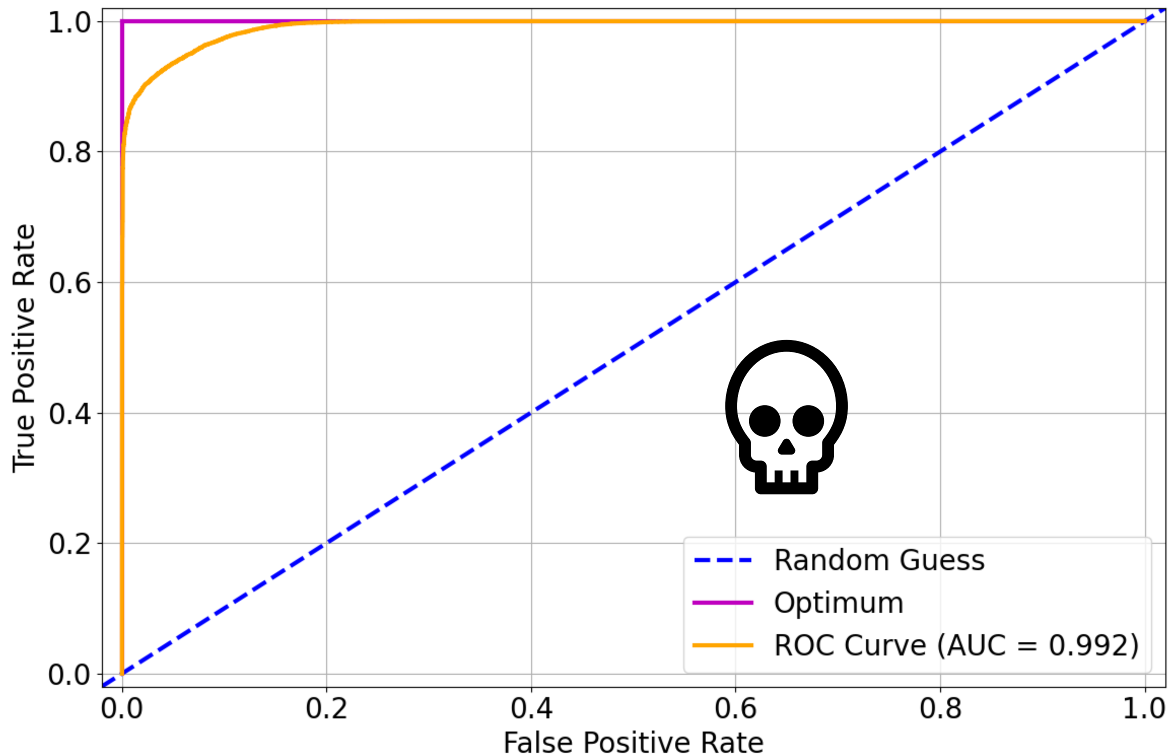


Evaluating a Binary Classifier: ROC-Curve

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

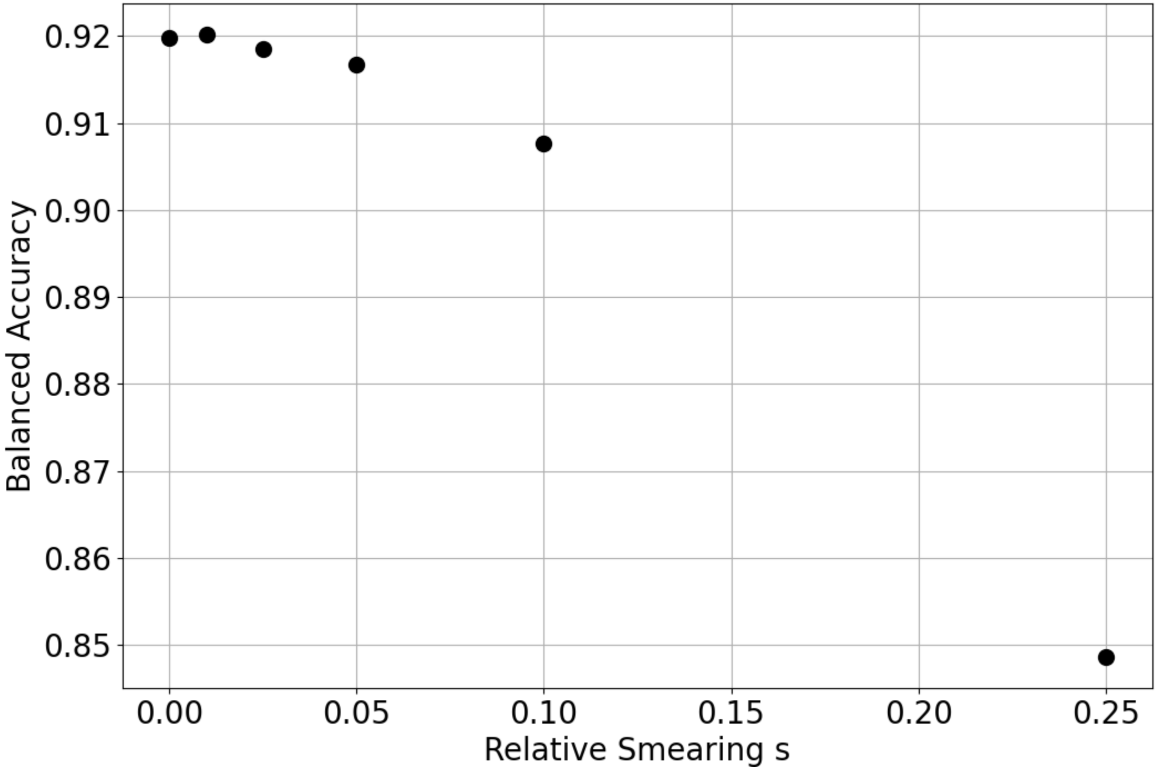
$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP+TN}}$$

- Run scan over threshold t
- Compute rates for each t
- Plot TPR vs. FPR for all scans
- AUC = Area Under Curve (Ideally = 1.0)



Don't go there: The model performs worse than a random guesser --> You are better off rolling a dice

Simple Robustness Analysis

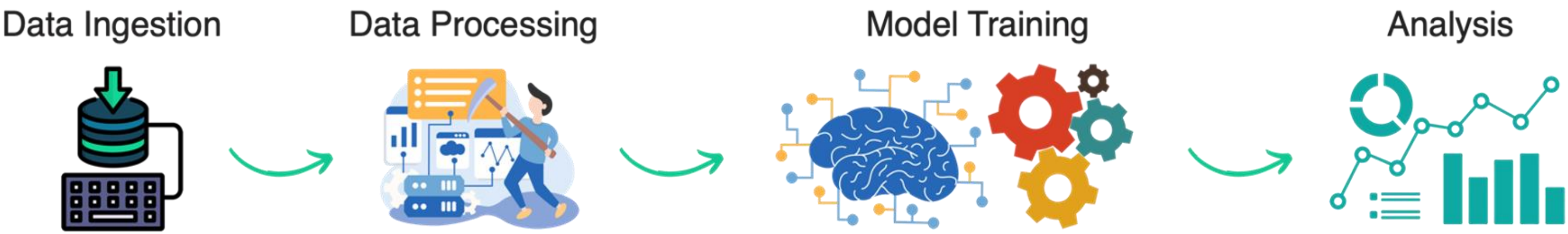


$$X_{smear} = X \cdot \mathcal{N}(1, s)$$
$$\hat{Y}_{smear} = \text{model}(X_{smear})$$
$$\hat{\ell}_{smear} = \Theta(\hat{Y}_{smear}, 0.5)$$

- What happens if trained model encounters data that is different to the training / validation data ? (e.g. different resolution effects in the detector)
- Alter validation data and feed it back into model --> Recompute performance
- When do we observe a significant drop in performance ?
- There exist better tests than this one

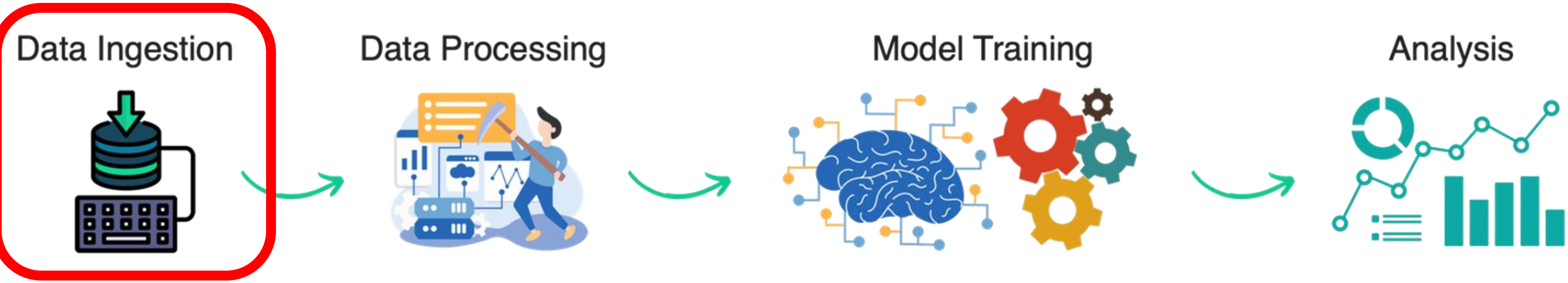
Jefferson Lab

Machine / Deep Learning Workflow

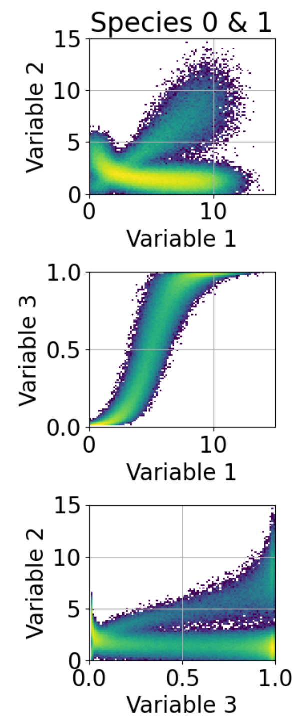


- Nearly every machine / deep learning analysis is based on these four steps
- Standardize analysis --> Enforce reproducibility and support collaborative efforts
- Dedicated generic framework developed in JLab Data Science Department

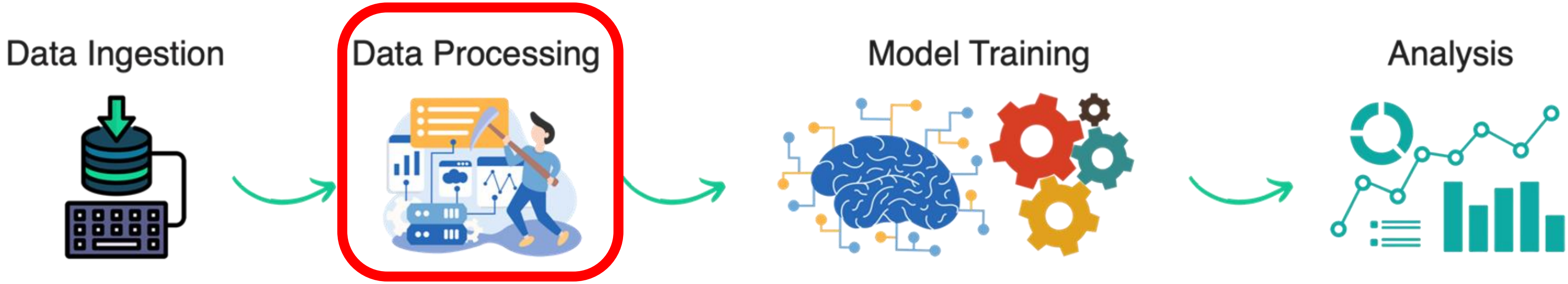
Machine / Deep Learning Workflow



- Load data (from database, numpy array, ROOT-trees,...)
- Data types
  - Digits
  - Images
  - Videos
  - Texts
- Commonly used data formats
  - .png files
  - .npy arrays (numpy)
  - .csv, .json (dataframes)



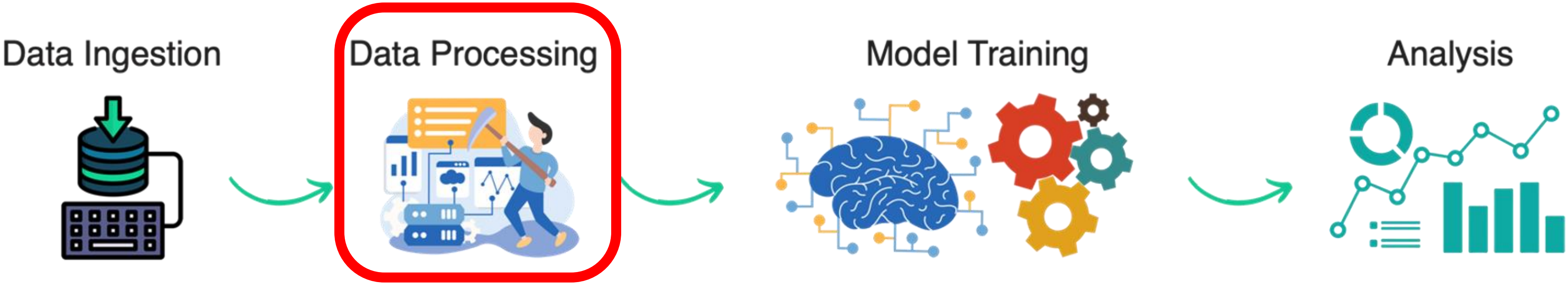
Machine / Deep Learning Workflow

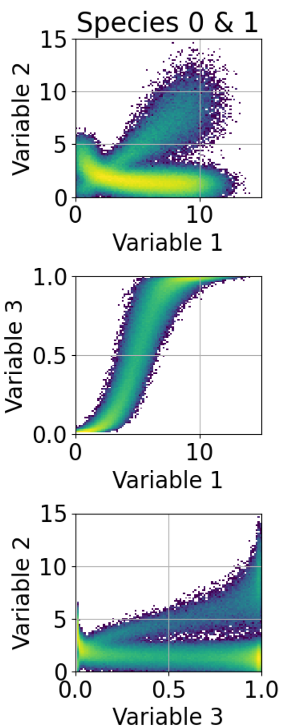


- Make sure that model can use data
- Feature engineering

Processing Method	Example	Why?
Adjust feature ranges	Do not feed vector (0.001,10000,40) into model	Model is likely to focus on large values
Exclude values	Acceptance holes in detector	Model may reconstruct false correlations
Select features	Particle energies, angles,...	Feed "useful" information to model

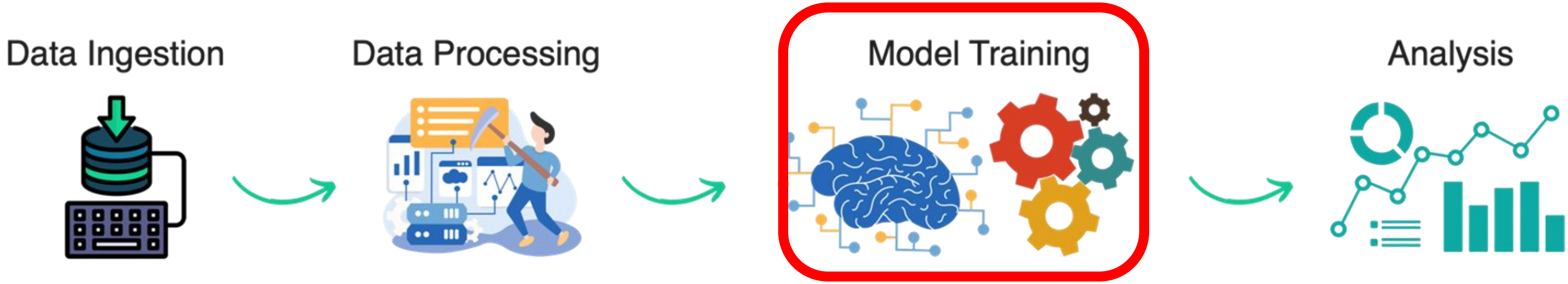
Machine / Deep Learning Workflow



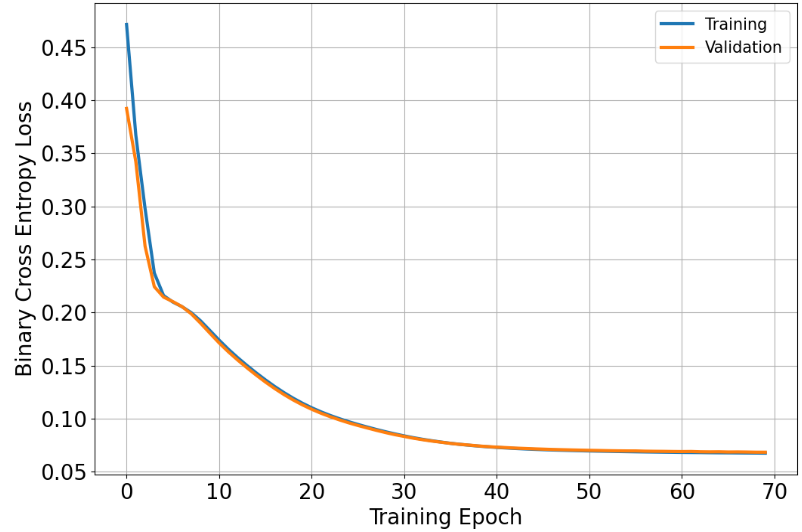


- Used a MinMax scaler in our analysis
- All features are scaled to be between 0 and 1

Machine / Deep Learning Workflow

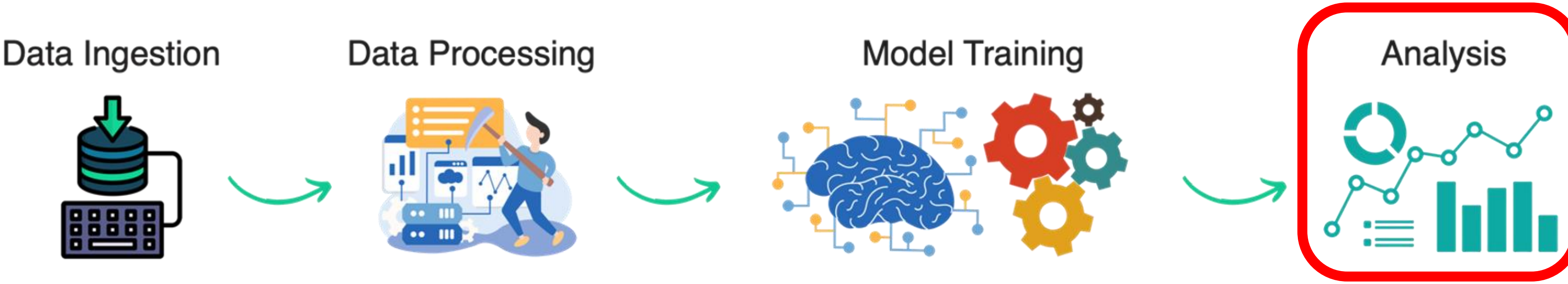


$$\frac{dF(\hat{Y})}{d\theta} = \frac{dF(f_\theta(X))}{d\theta} = 0$$

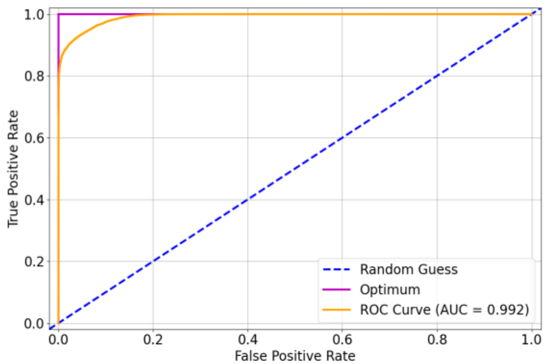


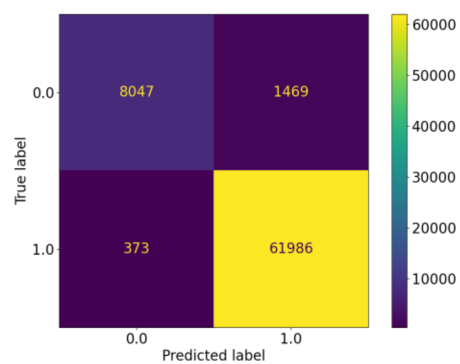
- Update internal model parameters w.r.t to given data
- Make sure that model objective (loss) converges
- Using validation data helps to better judge training

Machine / Deep Learning Workflow



- Evaluate model on separate data set --> Not used during training
- Determine model performance on validation data set --> Check for generalizability
- Compare model performance to other analyses / models
- Decide if model needs to be retrained or is ready for deployment





AUC	Balanced Accuracy
0.992	0.92

Highest possible score for both metrics: 1.0

Software and Tools

Software Package	Suited for	Language
sciki-learn	Machine learning with off the shelf models; Provides all tools to set up an entire ML workflow	python
tensorflow	Customize deep learning models; Supports variety of diagnostic tools, e.g. tensorboard	python
PyTorch	Customize deep learning models; High flexibiltiy for user to define own training / evaluation routines	python
keras	Customize deep learning models; Supports tensorflow and pytorch; HPO tools	python
ROOT TMVA	Machine learning with off the shelf models + Deep Learning with keras / PyTorch	C / C++ / python

Which one to choose? --> Depends on what you want to do and personal taste…

Summary & Outlook

- Very brief introduction to machine and deep learning
- Discussed (physics inspired) binary classification example
  - Trained neural network
  - Evaluated model on validation data
  - Observed promising performance
- A few practical tips
  - Always plot the response of your model
  - Use as many diagnostic plots as possible
  - Understand your model and avoid black boxes
  - Make your work accessible to others
  - Understand the data you are using
  - You do not need machine learning to fit a line through three data points --> Keep it simple
  - Have fun!
- Many topics not covered in this talk
  - Regression problems and multiclass classification
  - Uncertainty quantification
  - Hyper Parameter Optimization (HPO)
  - Fairness and model explainability
  - Unsupervised learning and reinforcement learning
  - Convolutional neural networks, graph neural networks, generative models ...
  - ...