

Check-In Code:

vision

# Intro to CV: PyTorch and MLP

Presented by Alan, Stephanie, and Weiji



ACM AI



E L E M E N T . A I

\$5000 Prizes + \$250 raffle + free food  
Saturday, February 18th

[ai.acmucsd.com/competitions/Element.AI](http://ai.acmucsd.com/competitions/Element.AI)

# today's agenda

1

## Overview

Computer Vision & MNIST

2

## General (Supervised) Machine Learning Problems

How do we formulate and approach a machine learning problem?

3

## Introduction to PyTorch

Tensors, Gradients, Autograd, Linear Algebra

4

## Multilayer Perceptron

Applying what we've learned



ACM AI

# Interactive Notebook

*Make a copy*

<https://acmurl.com/cv1-interactive>



ACM AI

# Overview

Introduction to Computer Vision and MNIST



ACM AI

# What is Computer Vision?

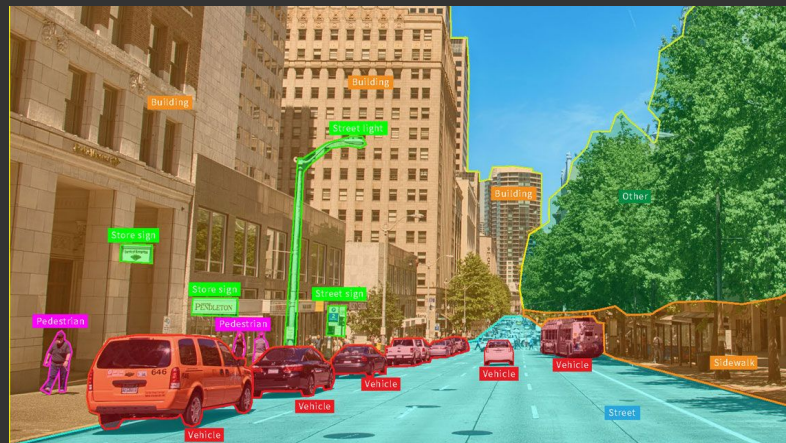
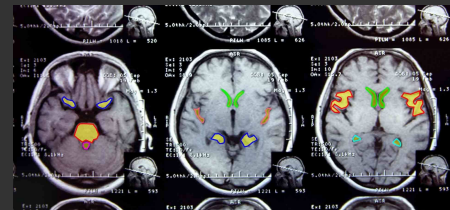
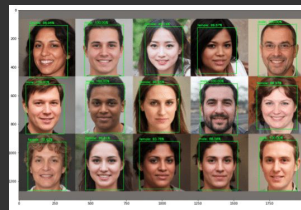
## Definition:

- Computer Vision is a field of AI that enables computers and systems to derive meaningful information from digital images, vidoes, and other visual inputs - and take actions or make recommendations based on that information.

## Applications:

- Autonomous Vehicles
- Facial Recognition
- Medical Imaging
- Virtual Reality
- Security
- Robotics

Today, we're talking about the tools (PyTorch) needed for CV and building towards Neural Networks (the foundations of CV)



# General ML Problems

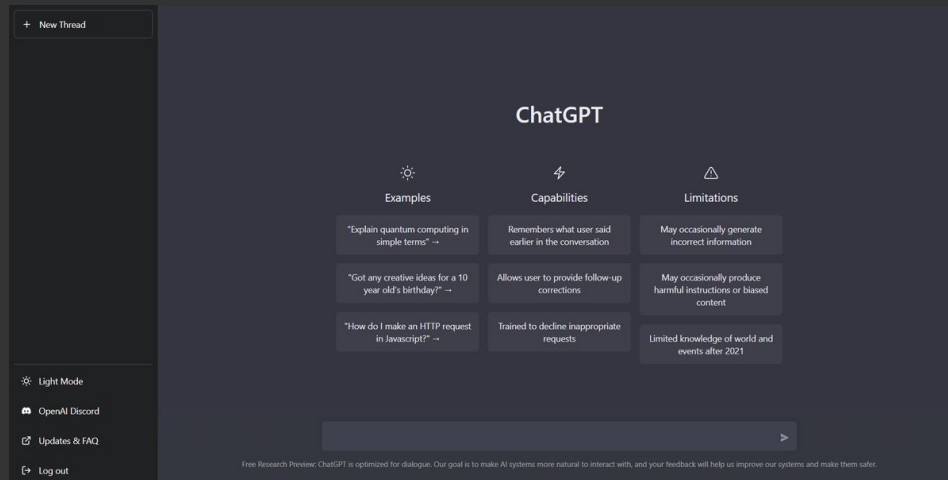
Formulating a (supervised) ML problem



ACM AI

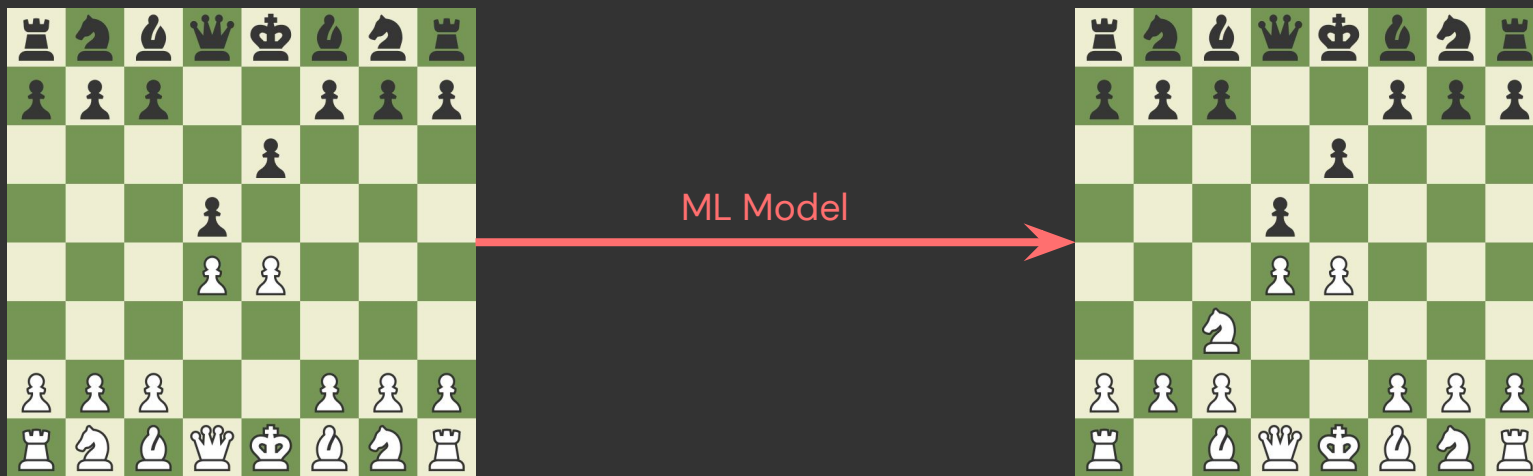
# General Machine Learning Problems

- In general, ML can be applied to many different situations, and their formulations are broad and varied
- From decision-making (such as in playing chess) to holding conversations (chatbots)





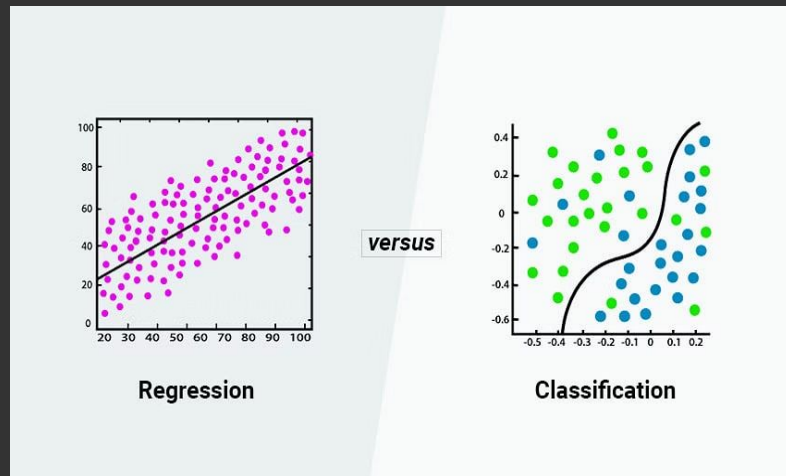
At the core it's about designing a function mapping input to a desired output.



# Basic Formulations: Regression vs Classification

## Machine Learning

- Using data to train a model to make predictions
- Mathematically, this core process of predictive modeling is called function approximation
  - Approximating a mapping function ( $f$ ) from input variables ( $x$ ) to output variables ( $y$ )
  - The mapping function predicts the category/labels for a given observation.
  - The biggest difference between regression and classification is regression outputs a continuous variable while classification outputs a discrete variable



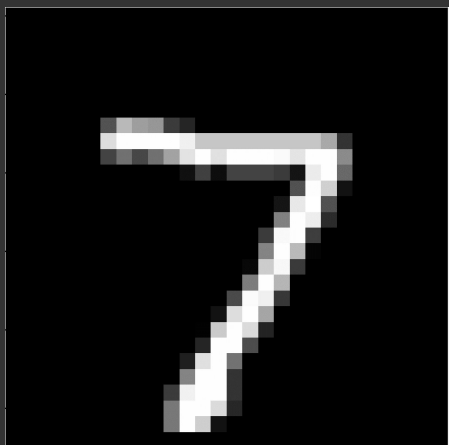
## Regression

- Ex. Predicting the price a house will set for
- Predicts a continuous quantity

## Classification

- Ex. Email being classified as 'spam' or 'not spam'
- Predicts a discrete class label





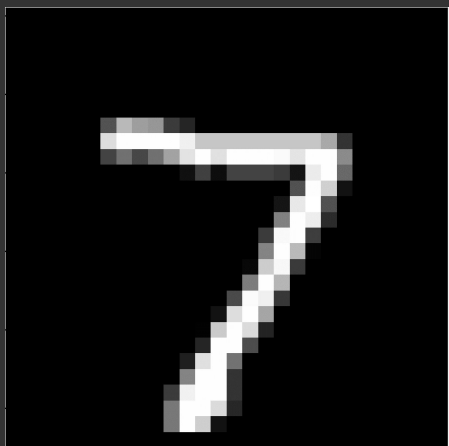
ML Model



7



ACM AI



28x28

ML Model

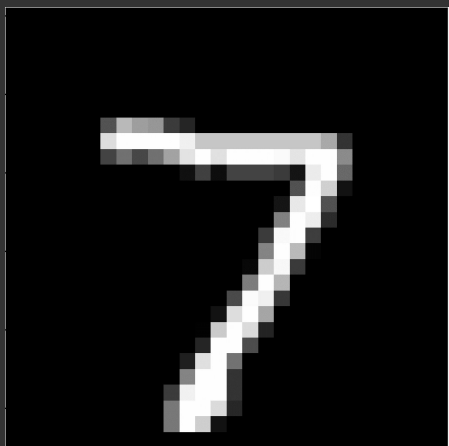


7

10 possible labels



ACM AI



784 x 1

ML Model

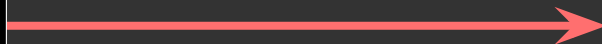
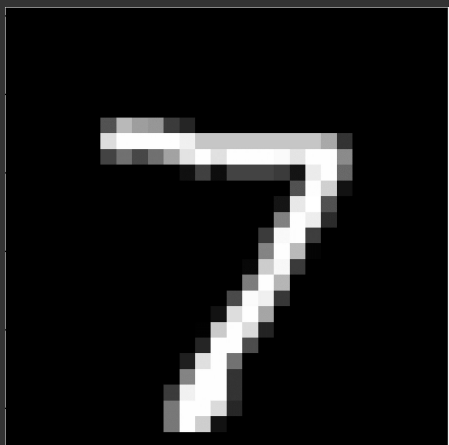


7

10 possible labels



ACM AI



7



ACM AI

# Intro to Pytorch

Tensors, Gradients, Autograd, Linear Algebra



ACM AI

# Introduction to PyTorch – Tensors

## Tensors

- A data structure (equivalent to a multidimensional array i.e. matrix) to store numeric values.
  - Default data structure for neural networks
- Tensor attributes
  - `torch.dtype` - species type of data in tensors
  - `torch.device` - specifies where tensor computations are performed, either CPU or GPU
  - `torch.layout` - species how tensors are stored in memory
- Tensor Operations
  - There's 100s of operations including arithmetic, linear algebra, matrix manipulation, etc.
  - Similar to NumPy
- CPU v. GPU
  - Default, tensors are ran on the CPU. You can switch to GPU for faster processing (but it's more taxing on the memory for larger tensors)

```
data = [[1, 2],[3, 4]]  
x_data = torch.tensor(data)
```

```
np_array = np.array(data)  
x_np = torch.from_numpy(np_array)
```

```
tensor = torch.rand(3,4)  
  
print(f"Shape of tensor: {tensor.shape}")  
print(f"Datatype of tensor: {tensor.dtype}")  
print(f"Device tensor is stored on: {tensor.device}")
```





# Introduction to PyTorch – Gradients and Autograd

## Neural Network Training

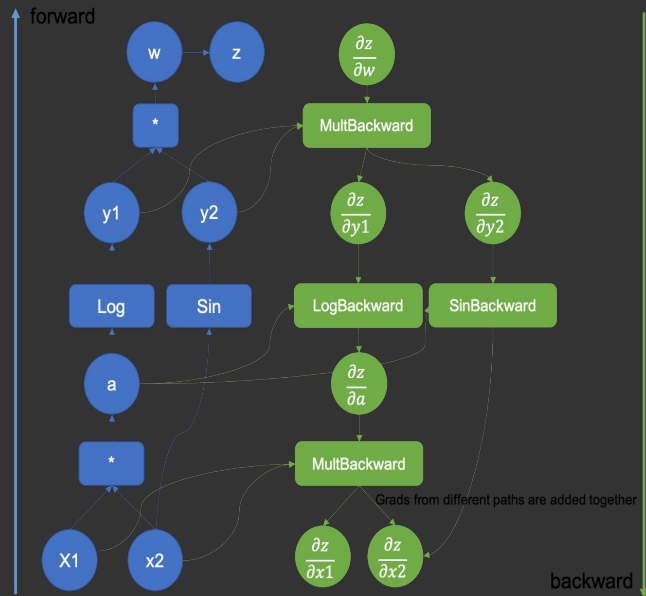
- Composed of 5 essential steps: defining architecture, forward propagation, calculating loss, **backward propagation**, and updating weights using learning rate
- Gradients and Autograd are essential to backward propagation

## Gradients

- Used to find the derivatives of the function
- Used to update the weight using a learning rate to reduce the loss and train the neural network.

## Autograd

- `torch.autograd`
  - a differentiation engine (calculates derivatives, specifically vector-Jacobian product) that allows for the computation in backpropagation
  - Simple terms, it computes partial derivatives while applying the chain rule



# Neural Networks



ACM AI

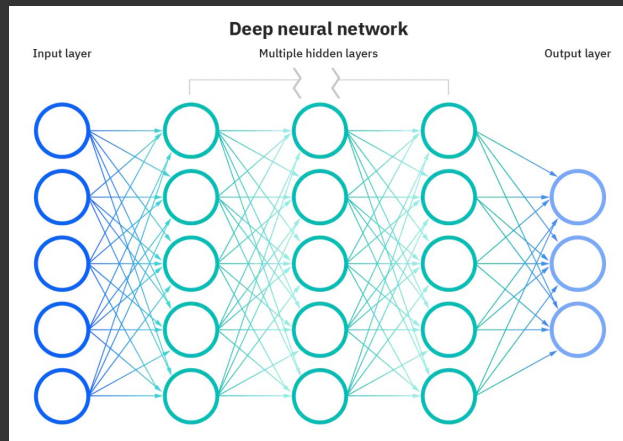
# Neural Networks: Definition

## Definition

- Mimics the brain through a set of algorithms
- A neural network is comprised of 4 main components: inputs, weights, a bias, and output
- Expressive nonlinear function approximators
  - Given an arbitrary number of “neurons”, neural networks can approximate any function
  - Introduce nonlinearities through activation functions

$$\sum_{i=1}^m w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

- Deep Learning: a subset of ML techniques based on multiple layers of neural networks



# How can we get computers to identify objects in images?

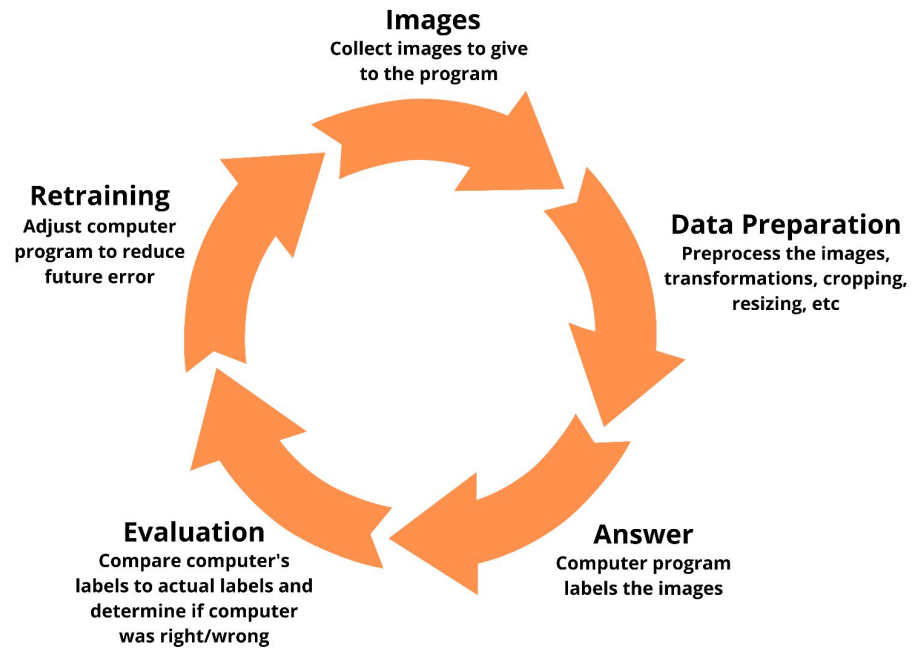


# How can we get computers to identify objects in images?



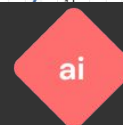
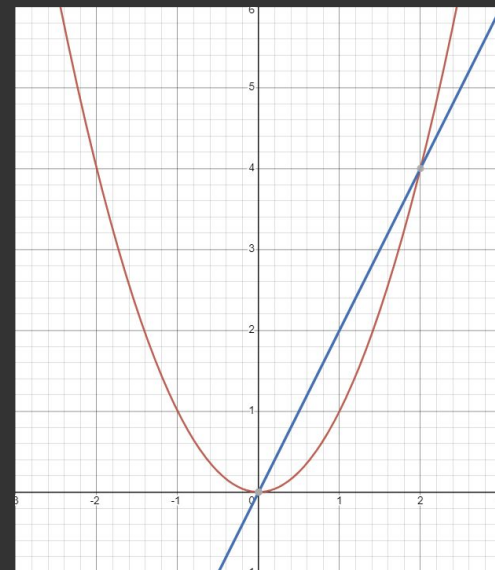
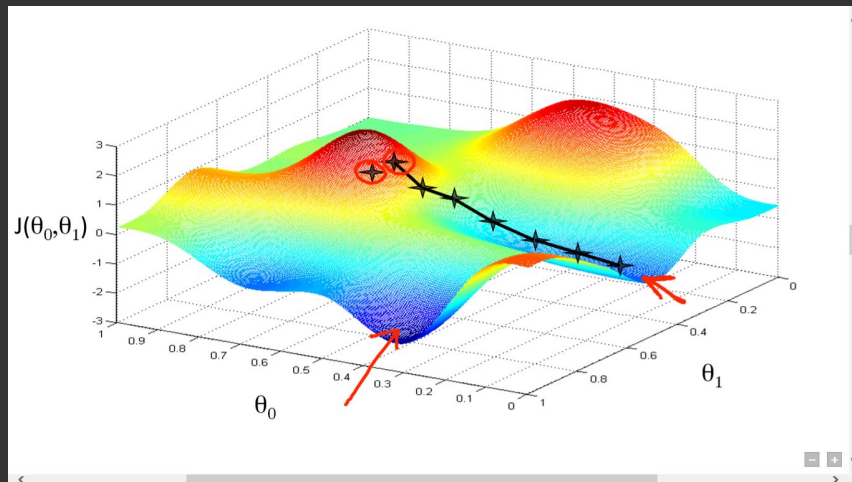
ACM AI

# The Idea



# Gradient Descent

- Model how wrong our neural network is as a differentiable function
- Gradient - multidimensional idea of derivative
  - Direction of furthest increase: negative gradient is direction of furthest decrease
- We map out the loss as a function of our neural network weights, and adjust weights to decrease error, or “loss”



# Loss Functions

- We need a method of evaluating how correct/wrong a given answer is
- We apply a softmax function to predict probabilities
- A method of evaluating how well specific algorithms models the given data
  - Large loss function = predictions deviate too far from the results
  - Using optimization functions, loss functions learns to reduce the error in prediction

## Categorical Cross Entropy

- Measures the difference between 2 probability distributions
  - In our case, we have 2: the current neural network distribution and the correct one
- Also differentiable





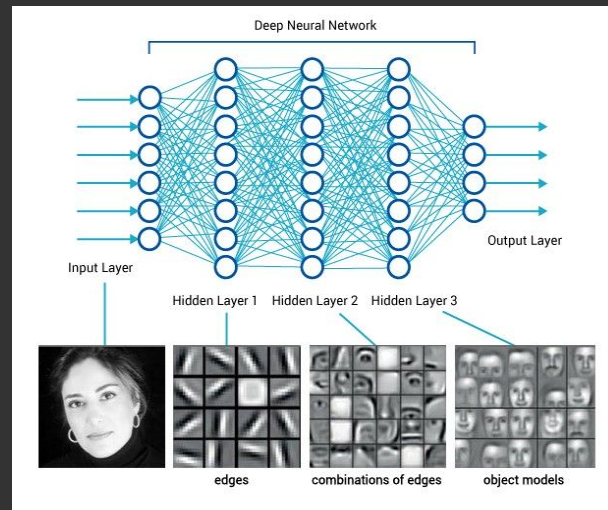
# Neural Networks: Computer Vision

## Why Neural Networks (Deep Learning) and not classic ML?

- Classic ML is more dependent on human intervention to manually determine a hierarchy of features (we need to label the dataset)
- Requires structured data
- Deep Learning w/neural networks don't! You can input in an unstructured dataset and have it give you the set of features that distinguish each object (very useful for more complex cases)

## Why Neural Networks for Computer Vision

- Traditionally, researchers had to train the computer to look for specific features in various images (top down approach).
- With a neural network approach, the deep learning algorithms trains itself to analyze the features in the image (bottom up approach)
- Thus, instead of having to tell the computer "what **should** be there," neural networks allow the computer to identify "what's there."



ACM AI

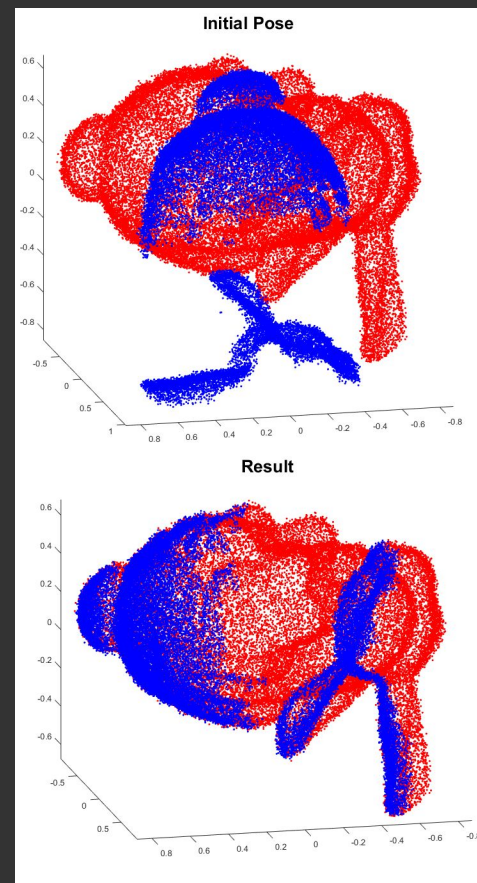
**Neural networks are not everything.**

# Neural Networks – Intuitions

- Neural networks are very popular for recognition tasks in computer vision/all the rage right now
- But they're still lacking in other areas - many 3D computer vision tasks require a mix of classical and deep ML
- Neural networks can be used as black boxes but they don't substitute for intent in approaching a problem

When are neural networks good?

- Essentially a nearest neighbors in the feature space
- Good for dimension reduction
- When euclidean distance represents semantic similarity

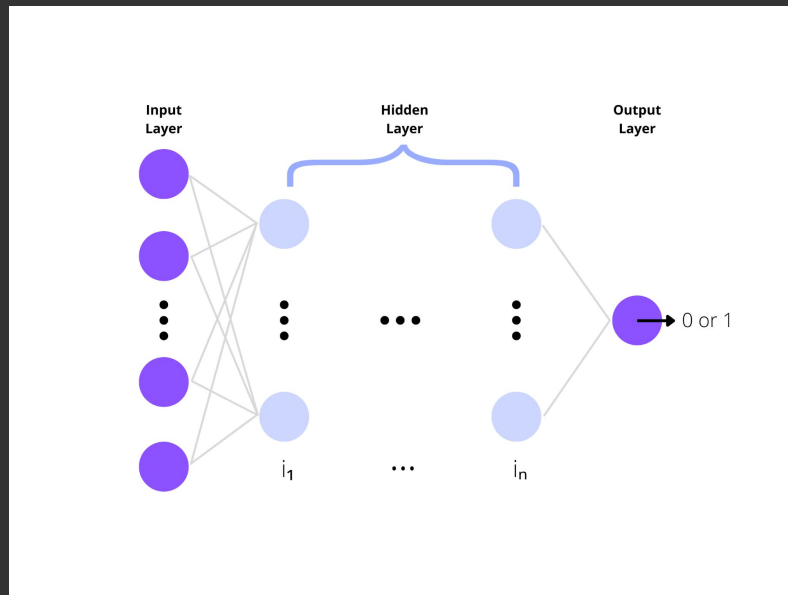


# Multilayer Perceptron

## Definition

- A fully connected class of feedforward neural networks
- Neural networks with at least three layers
- These three layers are input layer, hidden layer, and output layer

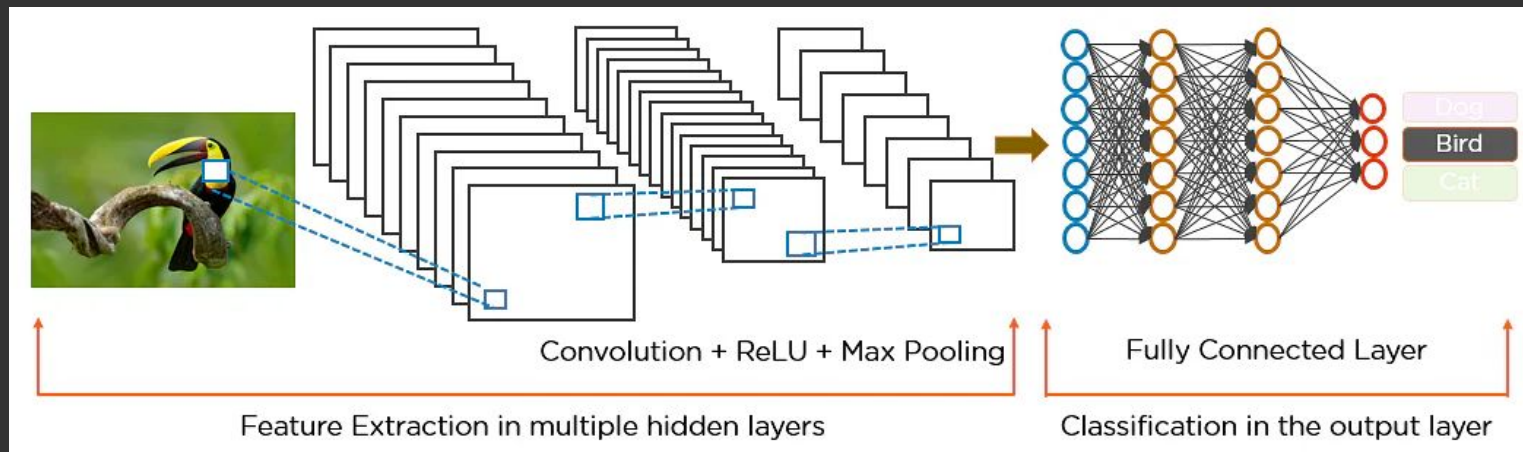
Let's learn how to make one!



# Neural Networks and CNN: Brief Overview

## Neural Network Architectures in Computer Vision

- Convolutional Neural Networks (foundation for modern CV).
- Gets an image, designates it some weightage based on different objects of the image, and then distinguishes from each other
- **More on this in Workshop 2!**



# Resources

All resources available in our GitHub repo

<https://acmurl.com/cv1-repo>

- Main Notebook (solutions)
- Interactive Notebook (questions)
- Recording (when uploaded to YouTube)
- Slides (pdf and pptx)



ACM AI

# Thanks for attending!

Make sure to check in to get membership points

