CS 181AI
Lecture 2

Intro to ML Models

Arthi Padmanabhan

Jan 23, 2023

Logistics

- Assignment 1 released this afternoon, due next **Monday, Jan 30** (one week turnaround because 90% of it will be done after today's demo)
- Code from class will be available this afternoon as well
- Reading group assignments will be sent by tomorrow morning, and first four papers will be on course webpage by then as well

Setup for Today

- Navigate to colab.research.google.com and create a new notebook

Today’s Plan

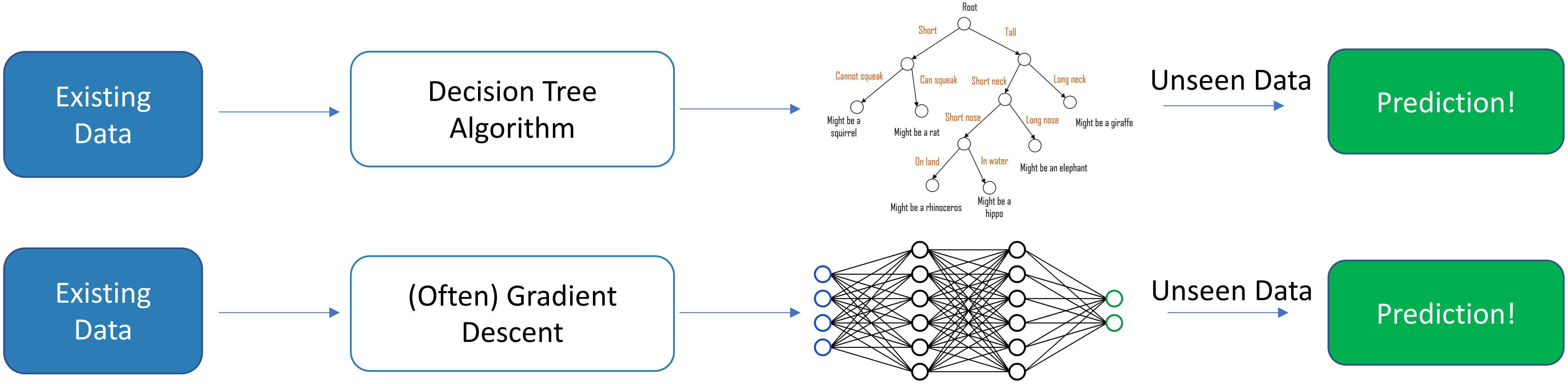
- Background on machine learning models
- Creating a neural network vs. using it – today’s focus is on using it
- Get set up with downloading an existing vision model (that someone else created) and using it on our images

Algorithms and Models and Neural Networks (oh my!)

- **Algorithm** is the pattern-recognition procedure that’s run on data to create a **model**
- **Neural network** is just one type of machine learning model

Algorithm

Model



Existing Data

Decision Tree Algorithm

Unseen Data

Prediction!

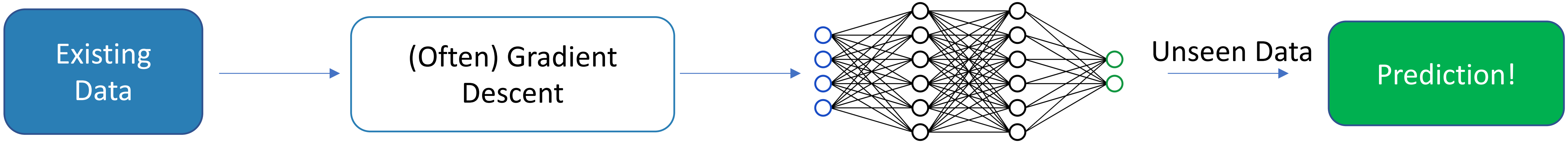
Existing Data

(Often) Gradient Descent

Unseen Data

Prediction!

Neural Networks



Existing Data

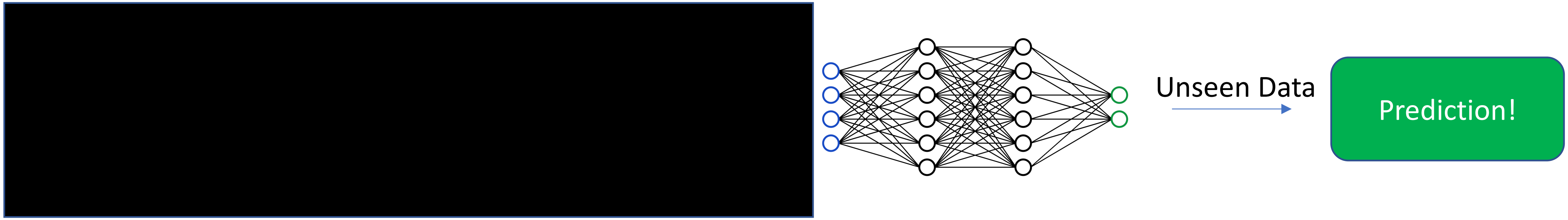
(Often) Gradient Descent

Unseen Data

Prediction!

- Process of using data to create the model: called **training**
- For today, training process = black box

Neural Networks



- Process of using data to create the model: called training
- For today, training process = black box
- Then how do we get a usable model??

Pre-trained Models

- Other people have trained on large public datasets to create models that you can use “off the shelf”
- For vision models, they can only identify specific objects

Widely Used Datasets

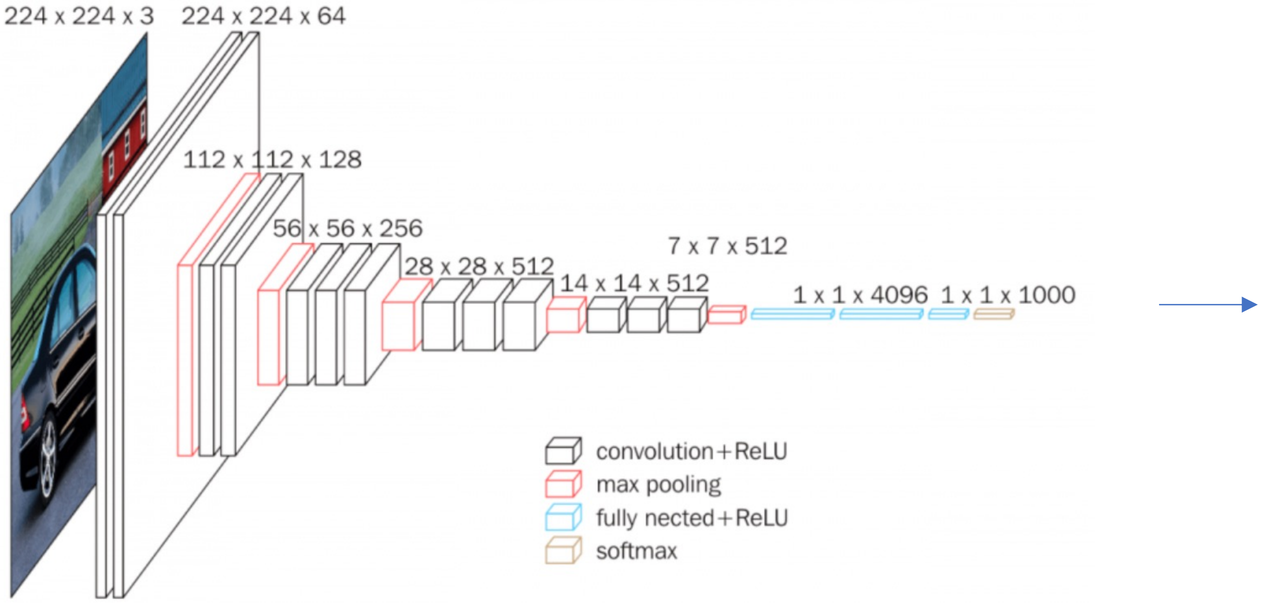
- ImageNet
- Pascal
- MNIST
- CIFAR

(a)

(b)

Structure of a Model

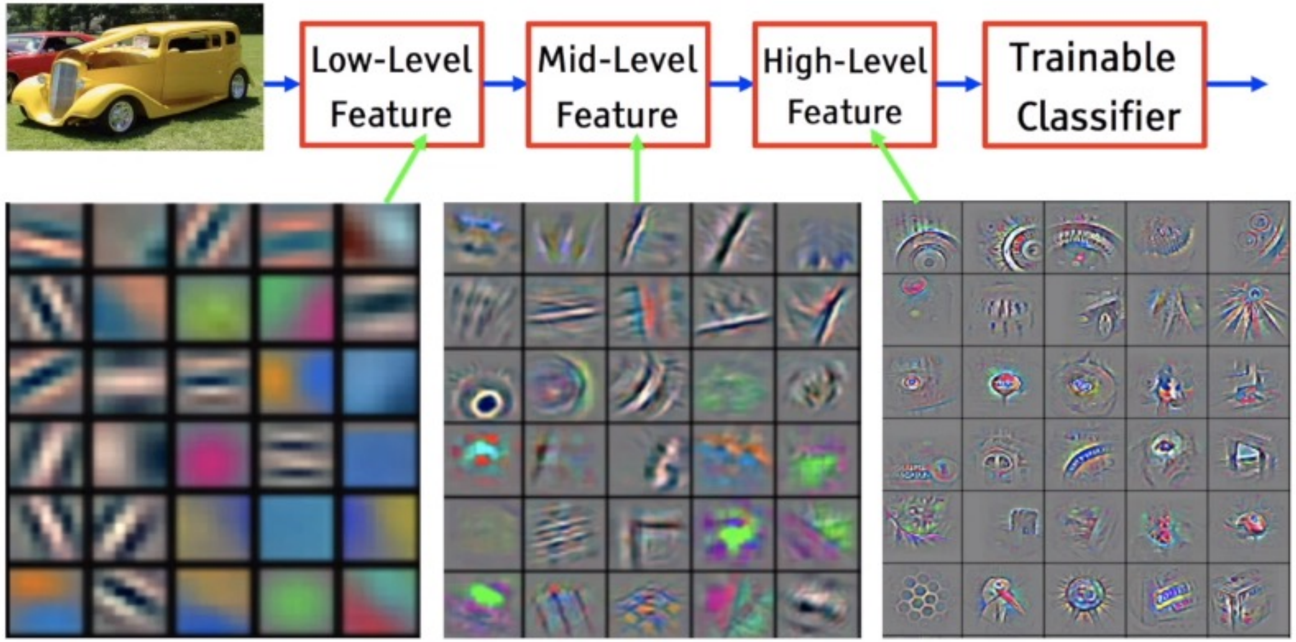
- Model consists of layers
- We’ll focus mostly on vision models, in particular convolutional neural networks (CNNs)



car

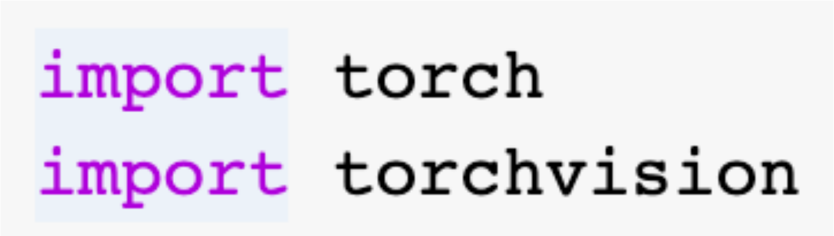
Roles of Different Layers

- Early layers **extract features**, last layers **classify image**



Today's Demo

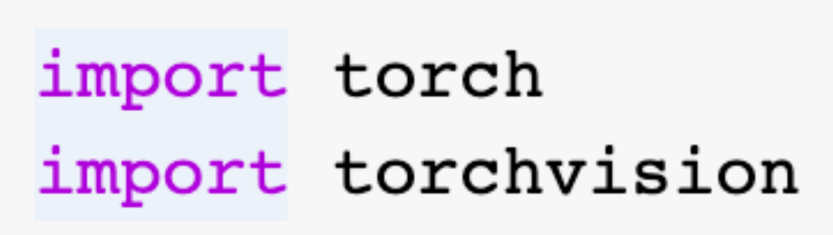
- Navigate to colab.research.google.com and create a new notebook
- Add the following lines to the top.



```
import torch
import torchvision
```

Torch & Torchvision

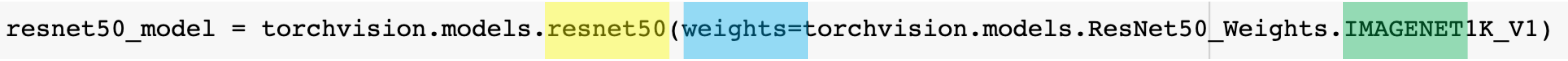
- torch: open-source ML library used for creating neural networks
- torchvision: open-source computer vision library that contains popular datasets, model architectures, etc.
- We’ll be using the “PyTorch framework”
    - PyTorch: open-source ML framework based on Python and torch



```
import torch
import torchvision
```

Today’s Demo

- Let’s download an existing model. ResNet50 is a common vision model structure. It has 50 layers and is used to classify between different objects
- Weights identify which dataset was used for training (more on this later)
- In this case, ImageNet was used



```
resnet50_model = torchvision.models.resnet50(weights=torchvision.models.ResNet50_Weights.IMAGENET1K_V1)
```

Today's Demo

- Try printing the model
- What types of layers do you see?

Types of Layers

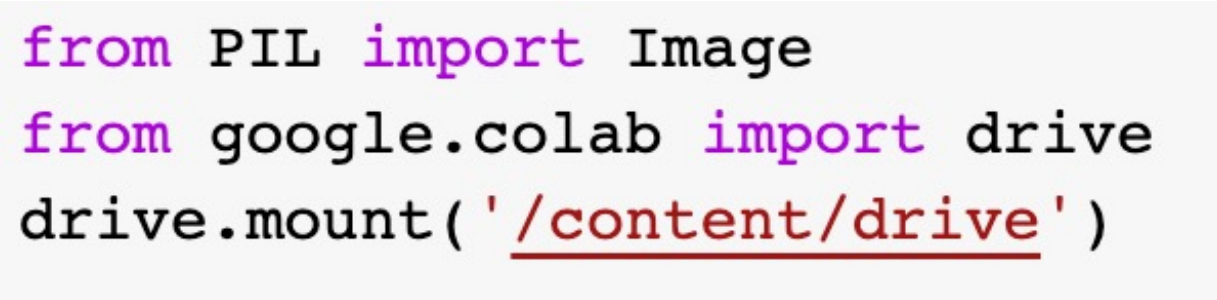
- Convolutional
- Linear
- Other items you might see when printing a model
    - Relu
    - BatchNorm

Inputs of a Model

- Tensor –> matrix
- Each step is matrix operation
- Weights are the matrices we multiply the inputs by

Today’s Demo

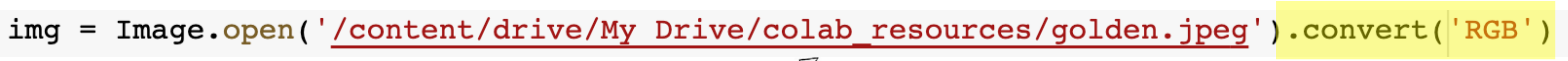
- Set up image access:
  - Put image in Google drive (use the same Google account you’re using to access colab)
  - Import Image from PIL (Python Imaging Library)
  - Mount the Google drive



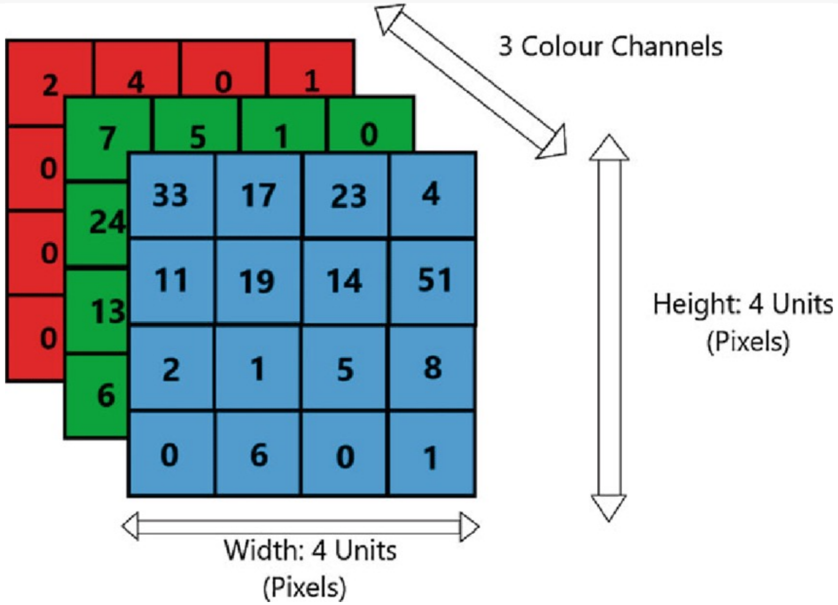
```
from PIL import Image
from google.colab import drive
drive.mount('/content/drive')
```

Today’s Demo

- Set up image access
- Open the image:
  - In this example, I put golden.jpeg in a folder called colab_resources in my Google drive and added the line below



```
img = Image.open('/content/drive/My Drive/colab_resources/golden.jpeg').convert('RGB')
```



Today’s Demo

- Set up image access
- Open the image
- Convert image to a tensor:
  - Torchvision has many types of transforms to manipulate the image before feeding it through the model. The ones that are important to us are:
    - Resize(x) -> resizes so that the aspect ratio stays the same and the height is x
    - CenterCrop -> crop to the center 224
    - ToTensor -> converts image to tensor
    - Normalize -> normalize with mean and std dev of the whole dataset

Today's Demo

- Set up image access
- Open the image
- Convert image to a tensor:
  - Torchvision has many types of transforms to manipulate the image before feeding it through the model. The ones that are important to us are:
    - Resize(x) -> resizes so that the aspect ratio stays the same and the height is x
    - CenterCrop -> crop to the center 224
    - ToTensor -> converts image to tensor
    - Normalize -> normalize with mean and std dev of the whole dataset
  - .unsqueeze(0) -> adds another dimension to the front, e.g. (3, 224, 224) becomes (1, 3, 224, 224)

Today’s Demo

- Parsing the output:
  - Each value represents 1 class from ImageNet –> the model’s prediction of how likely the image is to be that class
  - Softmax: rescale so that elements lie in the range [0,1] and sum to 1