The Universe of Possibilities Just Widened with Machine Learning

Using Machine Learning to make the world a better place and flesh out our understanding of ourselves

Please sign-in: **ucfai.org/signin**

By: John Muchovej

View this deck on Google Slides

The Universe of Possibilities Just Widened with Machine Learning

Part 1: Intro the Machine Learning (ML)

Please sign-in: **ucfai.org/signin**

Who am I and why should you trust me?

- John Muchovej
    - Comp Sci & Math-Econ major
- Machine Learning Researcher (2 labs)
    - Computer Vision
    - Natural Language Processing
- Freelance Data Scientist
- Contractor for Udacity, mentoring 300+ students in 10 “Nanodegrees” – namely Machine Learning, Deep Reinforcement Learning, and the like
- Founder of AI@UCF, sponsored by CBMM@MIT and Intel

Quick note:

A *single* deck will be used for both Part 1 and Part 2

Disclaimer:

An information dense deck lies ahead, don’t let that discourage you. I’m giving you words & resources to learn about after KH3.

At the end of the slide deck, you'll find links to helpful resources to learn more about this on your own!

Part 1: Intro to Machine Learning (ML)

Agenda

- What is Machine Learning? (ML)
- What can you use it for?
- Downloading datasets
- Exploring data
- What are Neural Networks?
- Building Neural Networks in PyTorch

What is Machine Learning?

- A toolset which avoids explicit programming by **learning from data**
- Modern machine learning is **approximates functions**
- Machine learning is much less about your algorithm of choice and more about ***the data you have access to***

> More data beats a cleverer algorithm [1]

- Pedro Domingos, 2012

What can we use ML for?

Computer Vision

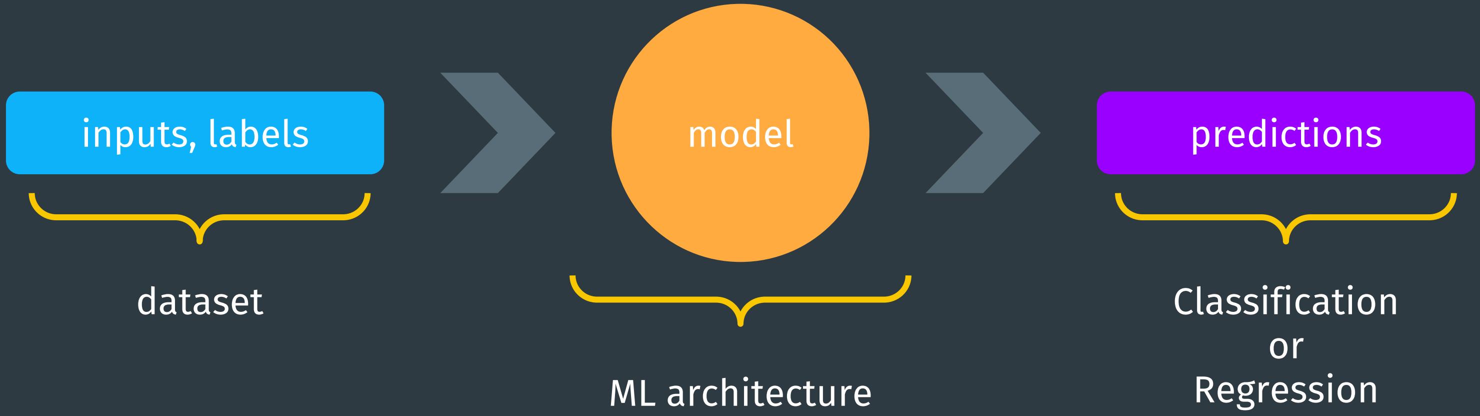
- Object detection
- Object recognition
- Object segmentation
- Object classification
- Landmark detection
- Optical Character Recognition
- ...

Natural Language Processing

- Sentiment Analysis
- Named Entity Recognition
- Syntax Analysis
- Content classification
- ...

There's significantly more which you can do, beyond what's enumerated here.
As long as you have data, you can probably use ML for it! :D

The Process (Supervised Learning)



inputs, labels

model

predictions

dataset

ML architecture

Classification
or
Regression

Downloading Datasets

- Finding data can be somewhat difficult
- Two great resources, though:
  a. Kaggle (kaggle.com/datasets)
  b. Awesome Public Datasets, GitHub (awesomedata/awesome-public-datasets)
- Generally, data can come in all kinds of forms
  a. For a hackathon, though, try to snag tabular or image data
  b. Audio can be pretty cumbersome if this is your first exposure to ML

Let’s pop over to Jupyter Notebooks for “Exploring data”

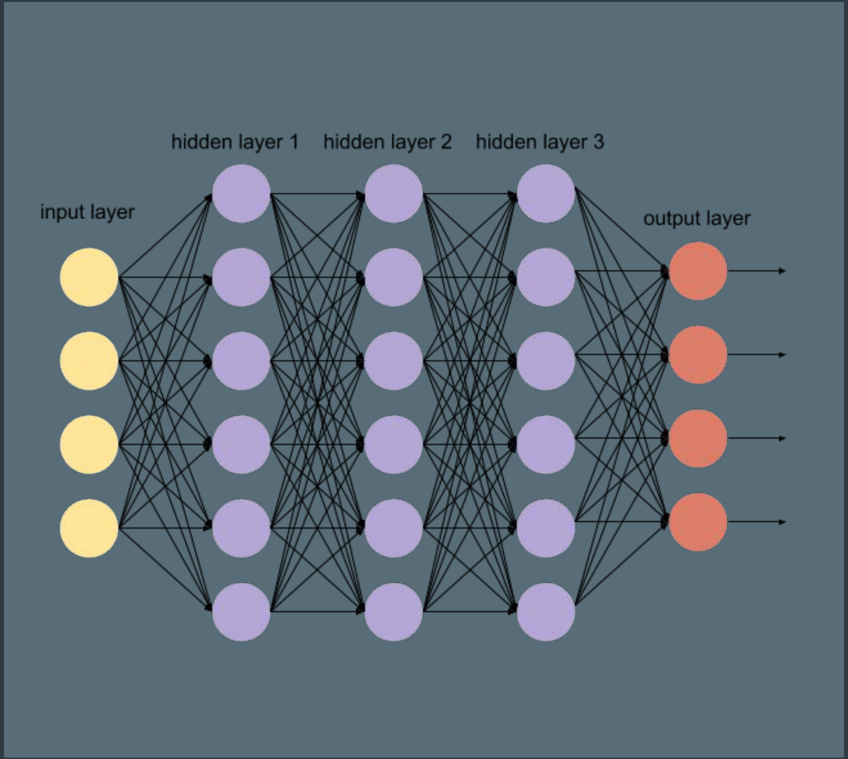
What are Neural Networks? (NNs)

- They've turned ML from your standard research field into the benchmarks-land-on-the-front-page-field we have today
- They have incredible representation power
  - More on this in a bit
- They can be increasingly sped by using GPUs
- One of many, many ML, "Learning Algorithms"
  - Support Vector Machines
  - k-Nearest Neighbors

Suffice it to say, NNs are just a fraction of what *is* considered "ML."

More technically, Neural Networks (NNs / FCNs)

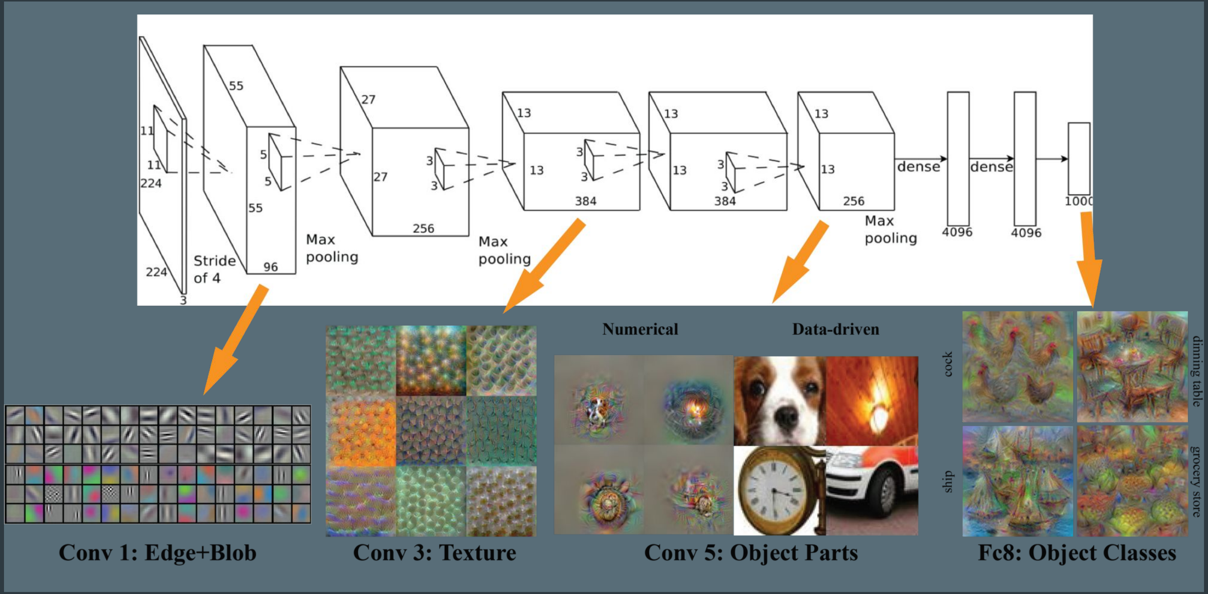
- NNs take in data, as an array (or vector)
- They multiply and combine the data you give it to produce new “features”
- The weights are “learned” through optimization
  - This flavor of optimization is called “Gradient Descent”



Convolutional Neural Networks (CNNs)

- NNs remove spatial data by flattening their input
- CNNs overcome that
- Typically called “feature extractors”

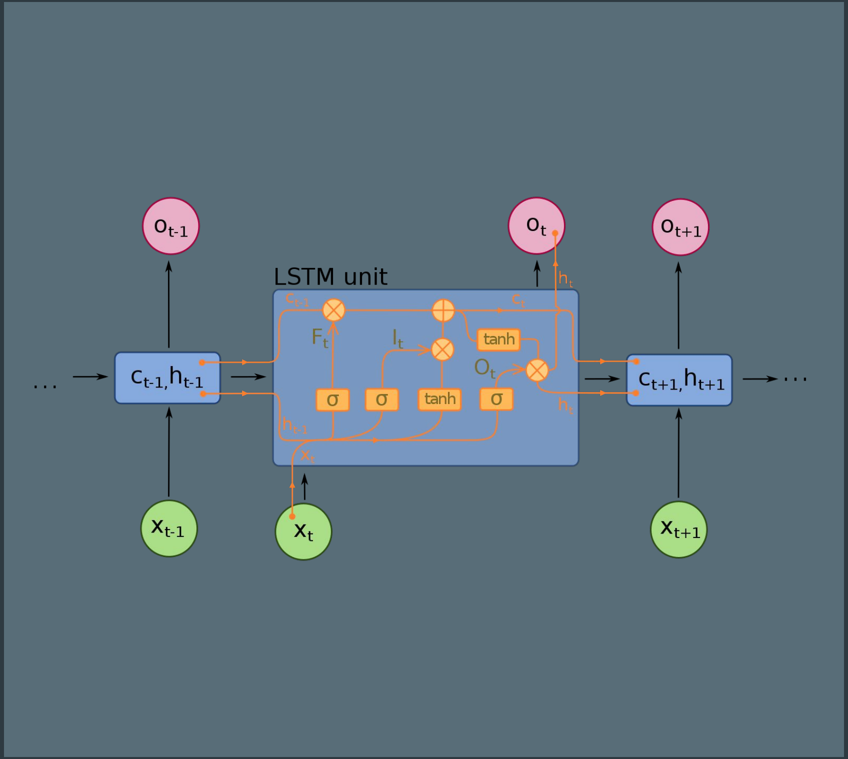
- Goal of CNNs is to extract high-level objects to be used by the FCNs



Recurrent Neural Networks (RNNs)

- Neither CNNs nor FCNs can handle sequences
- RNNs overcome that by maintaining “state”

- The most commonly used RNNs:
  - LSTM (Long Short-Term Memory)
  - GRU (Gated Recurrent Unit)



The nitty-gritty (i.e. hyperparameters)

- Typical optimization algorithm is Adam [2]
- Typical activation function for non-output nodes is ReLU [3]
- Loss functions: (PyTorch documentation)
    - Loss functions are used to actually train your model
    - When doing classification, consider using...
        - *Cross-Entropy*
    - When doing regression, consider using...
        - *Mean-Squared Error (MSE)*
        - *L1*
        - *SmoothL1*
            - From personal experience, this one seems to do best

Let’s familiarize ourselves with the PyTorch Framework

Let us know what you thought!
ucfai.org/feedback

The Universe of Possibilities Just Widened with Machine Learning

Part 2: DevOps for Machine Learning

Please sign-in: **ucfai.org/signin**

Who am I and why should you trust me?

- John Muchovej
  - Comp Sci & Math-Econ major
- Machine Learning Researcher (2 labs)
  - Computer Vision
  - Natural Language Processing
- Freelance Data Scientist
- Contractor for Udacity, mentoring 300+ students in 10 “Nanodegrees” – namely Machine Learning, Deep Reinforcement Learning, and the like
- Founder of AI@UCF, sponsored by CBMM@MIT and Intel

Part 2: DevOps for ML

Quick note:
This is Part 2, if you want to learn ML, checkout Part 1

Agenda

- DevOps is the bottleneck of ML
- Setting up a Google Cloud Platform account (GCP)
- Selecting resources for GCE (Compute Engine)
- Using GPUs in PyTorch
- High-level exploration of the different GCP APIs

DevOps is the bottleneck of ML

- Getting your models up and running can be a pain on the cloud
- Ensuring your development and production environment are as similar as possible is challenging
- Check-out open-source “cutting edge research” – try reproducing their results and running their code
  - Actually, don’t. It’s a pain, liable to cause hemorrhaging

Setting up a Google Cloud Platform account (GCP)

- Those Anaconda YAML files from earlier should be enough to get you started on GCP!
  - I’ve included a setup script which will also get you running with PyTorch, CUDA, and the like. :D

- Launch the “Compute Engine”
- Go to your Quotas and request...
  - “Global GPU limit” be raised to 3-4
  - “us-*#-(a/b/...)” zones raised to 2-3 (take a look at GPU locations)

Selecting resources for GCE (Compute Engine)

- Start out with 4-cores, 16GB RAM (n1-standard-4)
- Attach a single NVIDIA T4 / P4 (this will run you $0.79-1.14 / hour)
- **Focus on deploying multiple experiments with different "hyperparameters"**

Google Cloud Compute Engine pricing / etc.

- GPU pricing
- CPU / RAM pricing

Using GPUs in PyTorch

- The PyTorch documentation on this is great
- General rules of thumb, though
  - Put your model on the GPU
  - Process your data on CPU, then transfer it to the GPU
  - In a time-crunch like this, stick to 1 GPU

High-level exploration of the different GCP APIs

If you’re doing a project, like an Alexa skill, odds are the GCP ML APIs should do enough!

- AutoML is a really interesting tool
    - It allows you to perform “transfer learning” by uploading your own data to optimize Google’s pre-existing models
    - AutoML Vision (Beta)
    - AutoML Natural Language (Beta)
    - AutoML Translation (Beta)
- As a researcher, I haven’t really used these, however I’ve read/heard that the Python library is okay and than raw REST requests might be more useful

Questions?

Let us know what you thought!

ucfai.org/feedback