Introduction

Arthur J. Redfern
arthur.redfern@utdallas.edu

Outline

- Welcome to the course
- Motivation
- Why Now
- Course preview
- Themes
- Logistics
- Expectations
- Opportunities
- Questions

Welcome To The Course

Thank You

- You're taking an important step in your academic and professional careers
- I appreciate you taking it with me
- I'm looking forward to a fun semester

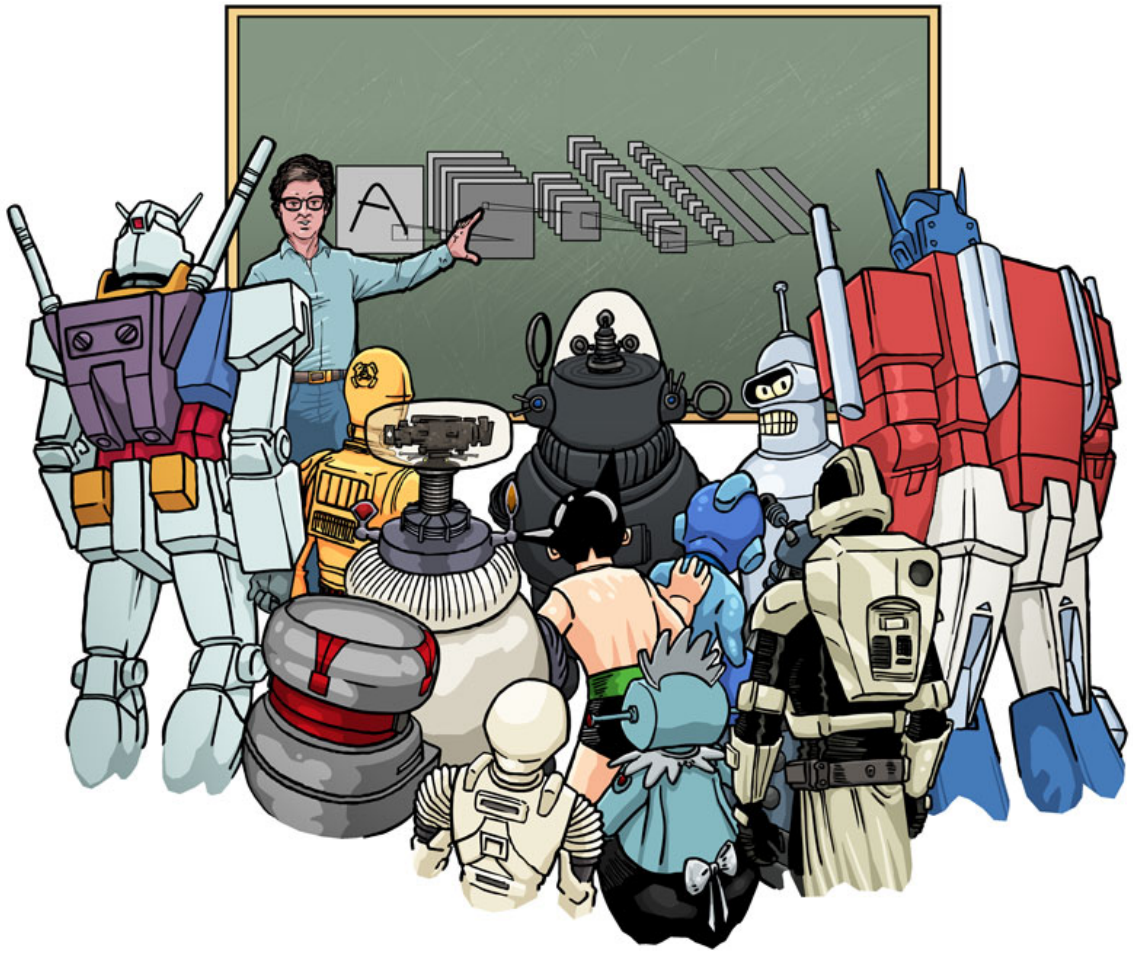
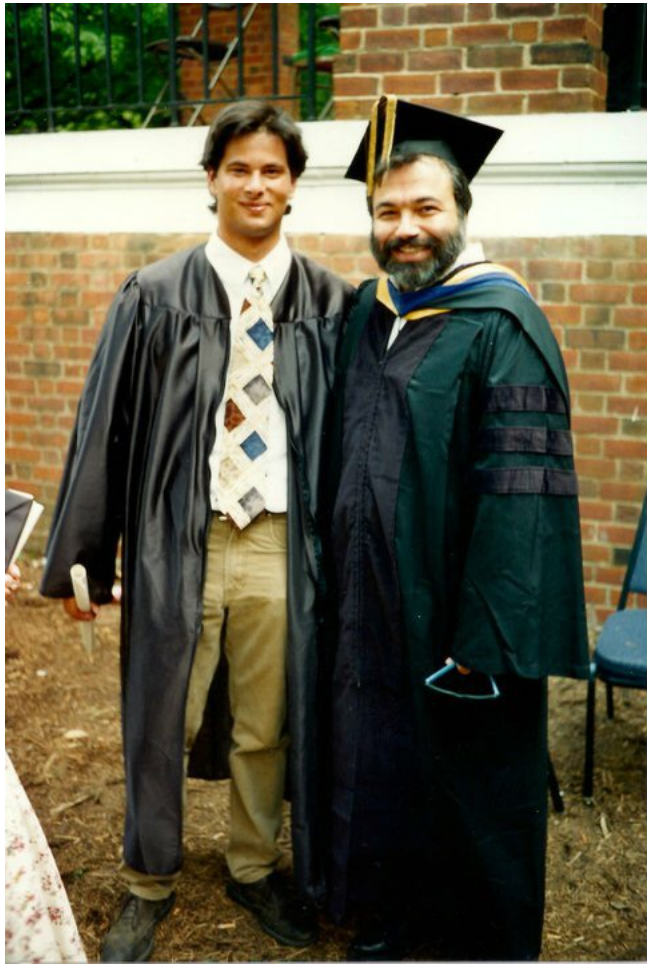


Figure from https://www.reddit.com/r/MachineLearning/comments/4tmpl9/tribute_an_art_made_by_andré_pinto_anthill_comics/

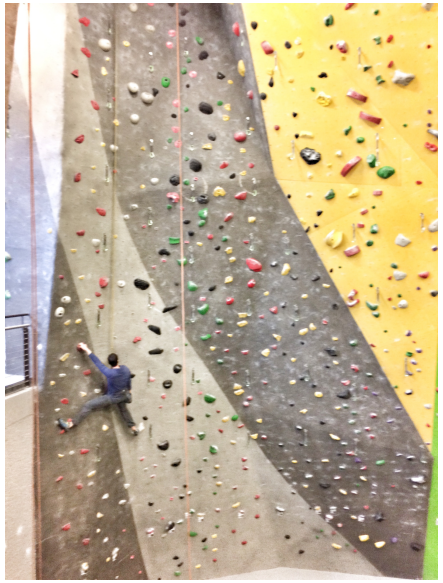
About Me

- Grew up a little south of Richmond, Virginia
- BS in EE from University of Virginia in 1995
  - Started in chemical engineering
  - Moved to electrical engineering
  - Specialized in signal processing
- PhD in ECE from Georgia Tech in 1999
  - My thesis was on Volterra systems (a nonlinear generalization of convolution)
- I'm ok with you calling me any of {Arthur, Dr. Redfern, Professor Redfern}



About Me

- Moved to Dallas to work at Texas Instruments in 2000
  - Physical layer communication system design
  - Signal processing for analog systems
  - Machine learning
- Currently I manage a machine learning lab in the TI Embedded Processors organization
  - Algorithms, software and hardware for different applications
  - This class will cover much of the same (that's not an accident)
- Live in Dallas, Texas with my wife and son
- I like to add a new hobby every few years
  - Cars, guitar, poker, golf, running, biking, yoga, climbing, ...



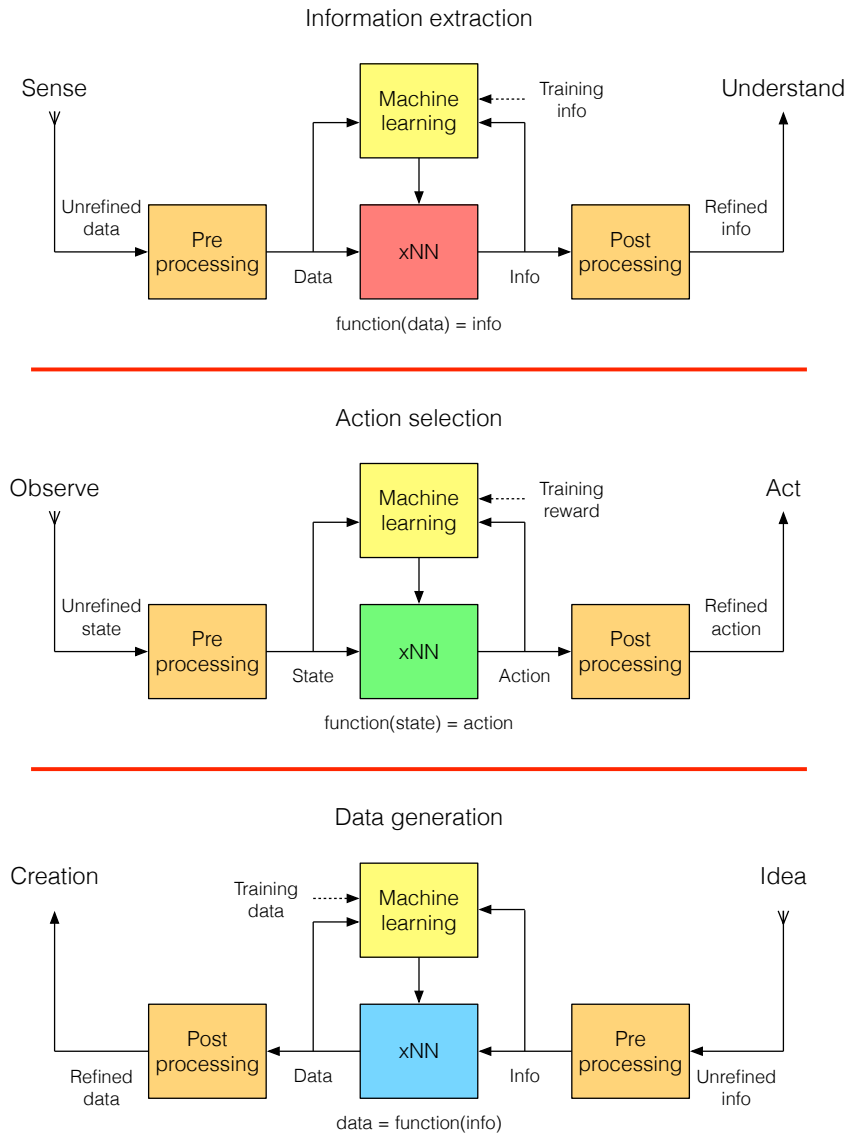
Motivation

Why xNNs (NNs, CNNs, RNNs, Self Attn, …)?

- They work better than anything else for many applications
  - Supported by theory
  - Realized in practice

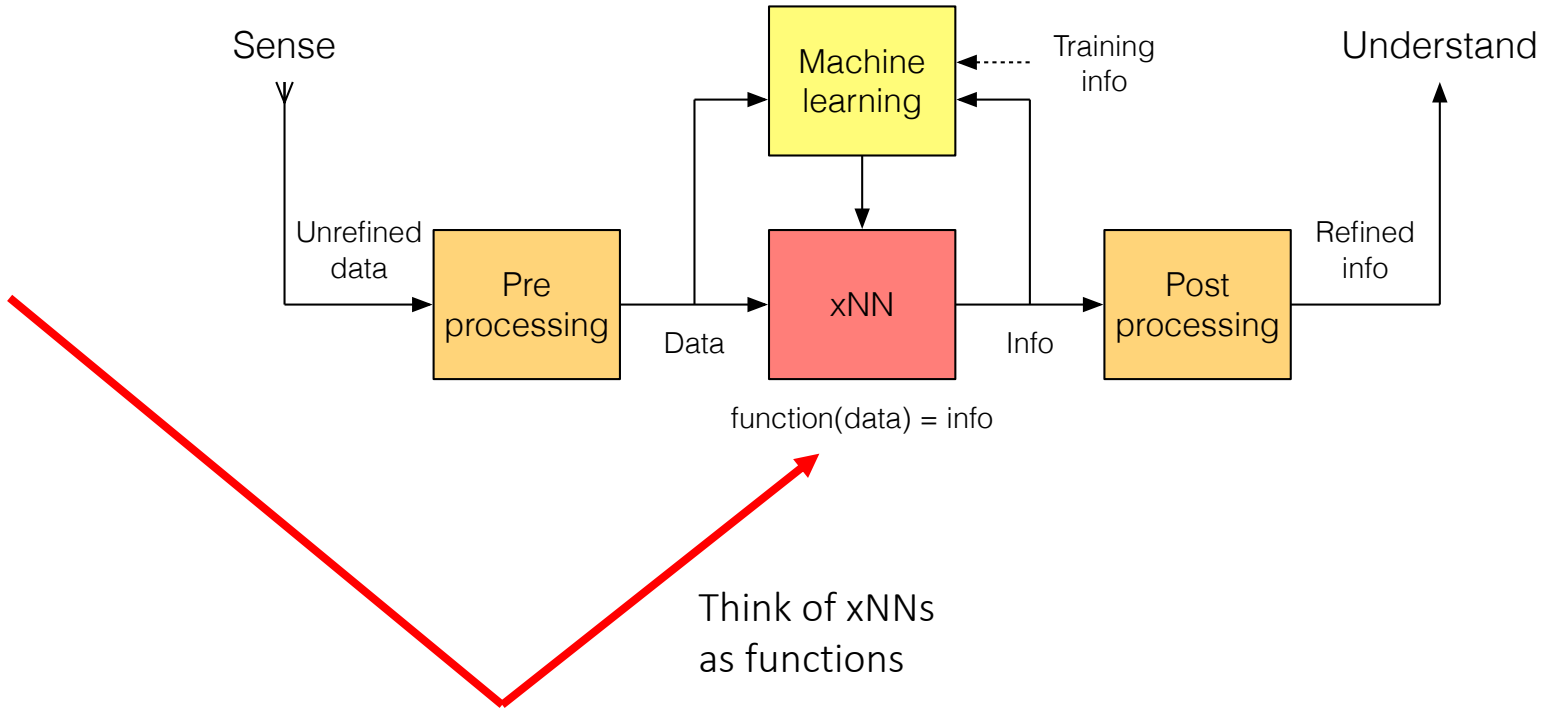
Definitions (not Webster quality)

- Intelligence is the ability to acquire and apply knowledge
  - Artificial intelligence is intelligence exhibited by algorithms
- Learning is the acquisition of knowledge from experience
  - Machine learning is learning from data (experience) applied to an algorithm such that it exhibits artificial intelligence
  - Deep learning is machine learning applied to a deep structure
- xNNs are deep structures trained using deep learning to exhibit artificial intelligence



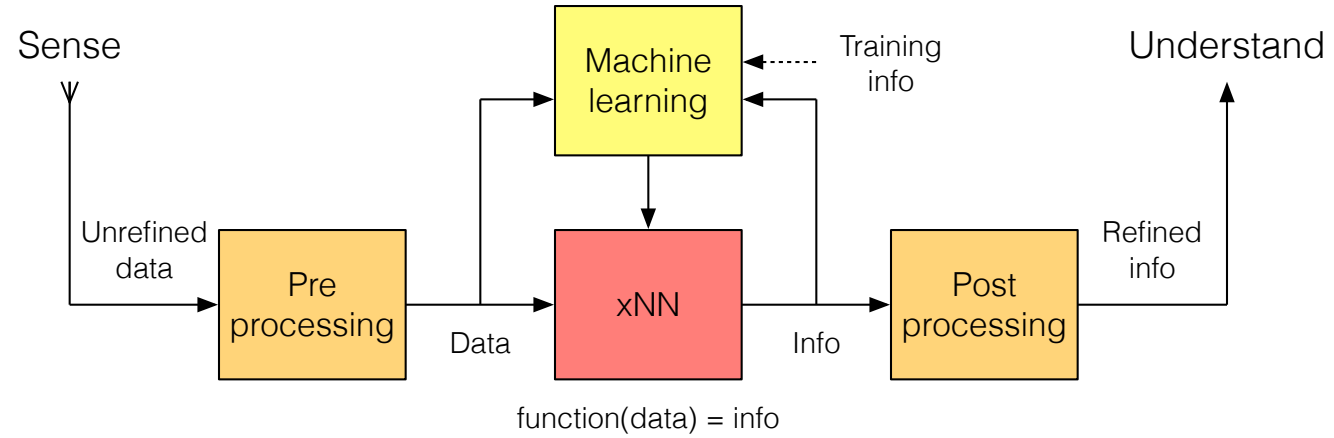
Information Extraction

- Trend: the instrumentation of everything (people, objects, spaces)
- Transformation from data space to information space
  - Vision, language, speech, ...
- Will also hear information extraction referred to as analytics
  - With the result being actionable information



Sense – Compute (xNN) – Understand

- Pre processing
  - Make feature extraction easier
  - Data cleaning, dimensionality reduction, …
  - Frequently via application specific side information
- xNN
  - Encoder feature extraction
    - Make prediction easier
  - Decoder prediction
    - Classification (discrete)
    - Regression (continuous)
- Post processing
  - Clean up predictions
  - Frequently via application specific side information



Supervised and self supervised machine learning is commonly used for end to end training of the xNN from data

Theory

- Many tasks can be cast as a classification or regression problem
  - Probability of input x belonging to class y
  - Given an input x predict a pmf P(y|x)
- Neural networks are universal function approximations (under mild conditions)
  - The view of xNNs as a function applicable to all sorts of different data inputs is very important to keep in mind
  - Typically use them to approximate the mapping from x to P(y|x)
- It's possible to design xNNs to exploit structure in the data
  - Spatial with CNNs
  - Sequential with RNNs
  - Localized with attention

Figure from https://www.mapillary.com/dataset/vistas

Practice

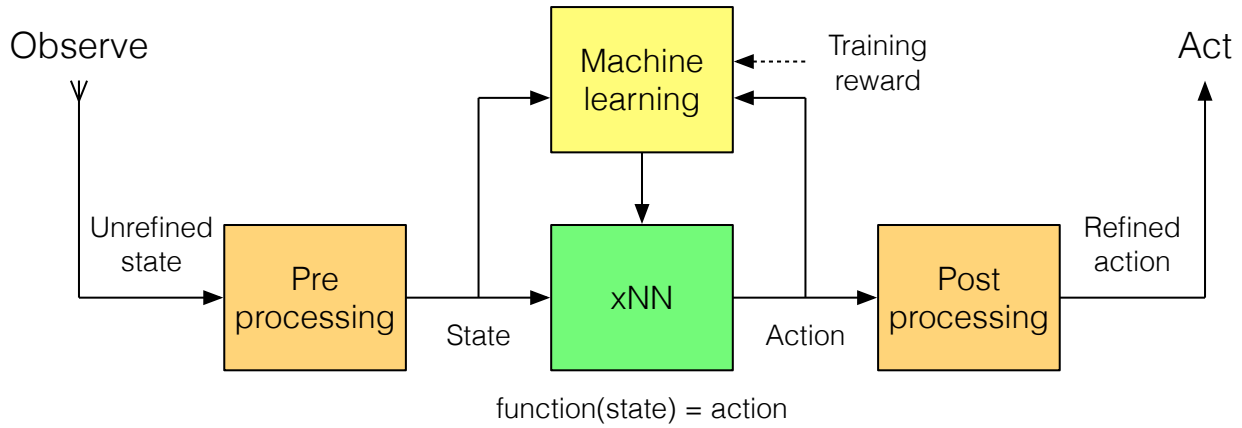
- It's possible to train xNNs from input output data
  - End to end supervised learning allows feature learning vs feature engr (though implicit in network design and training hyper params)
  - Automatic differentiation with reverse mode accumulation
  - Gradient descent variants
  - Keys are convergence and regularization for generalization
- Software and hardware exists for efficient implementations
  - High level graph specification and transformations
  - Low level graph software runtime
  - Primitives for compute and data movement
  - Lowering of tensor ops to matrix matrix mult where possible
- They provide state of the art results in many applications
  - Vision, language and speech are the biggest successes
  - Games, art, control, genomics, ...
  - A general tool with logical extensions to new apps going forward

You can't discuss top performing vision, language or speech algorithms without xNNs

Figure from https://www.tensorflow.org/tfrc/

Action Selection

- Transformation from state to action
- Methods for making decisions
  - Value based
  - Policy based
  - Model based
- Within this xNNs used for
  - Value function approximation
  - Action value function approximation
  - Policy function approximation
- Frequently to improve search and planning



Observe – Compute (xNN) – Act

- We'll still use data and labels to train xNNs
  - Just like in supervised and self supervised learning for information extraction

- But the collection of the data and labels will frequently be done during training via reinforcement learning

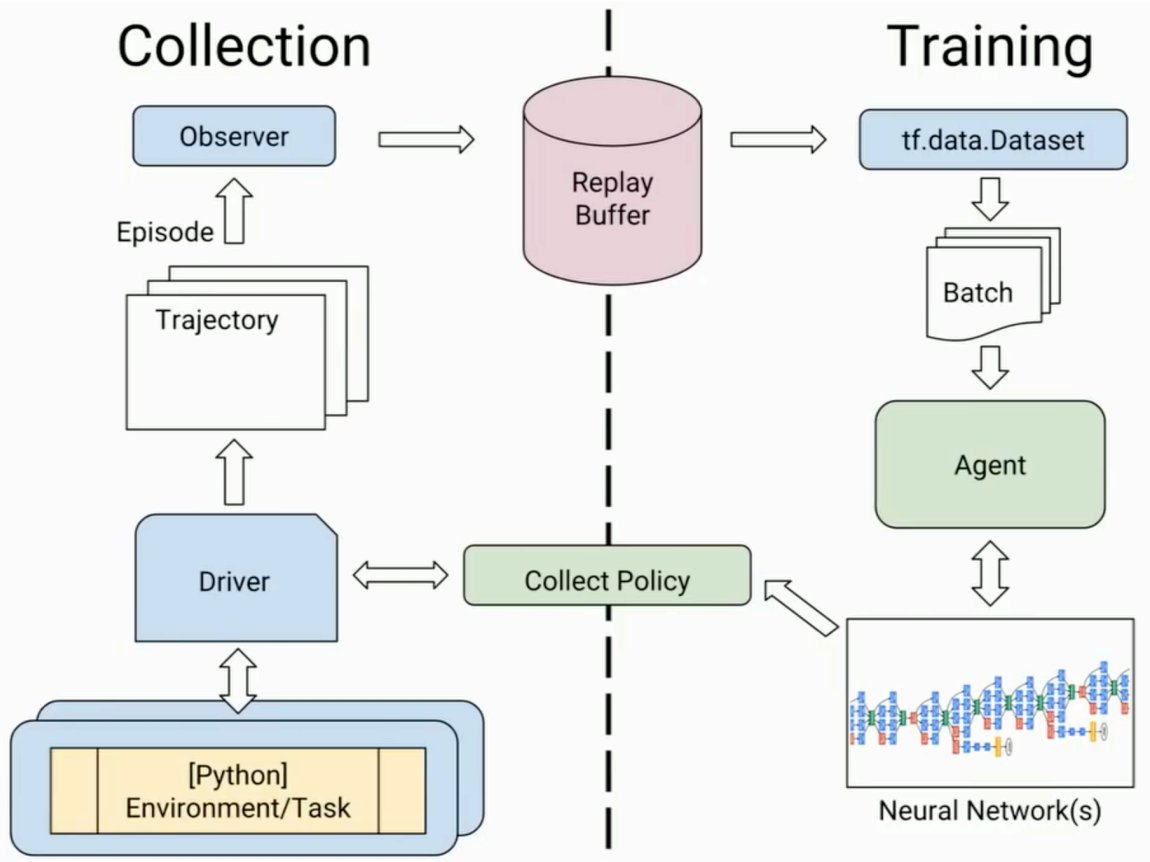
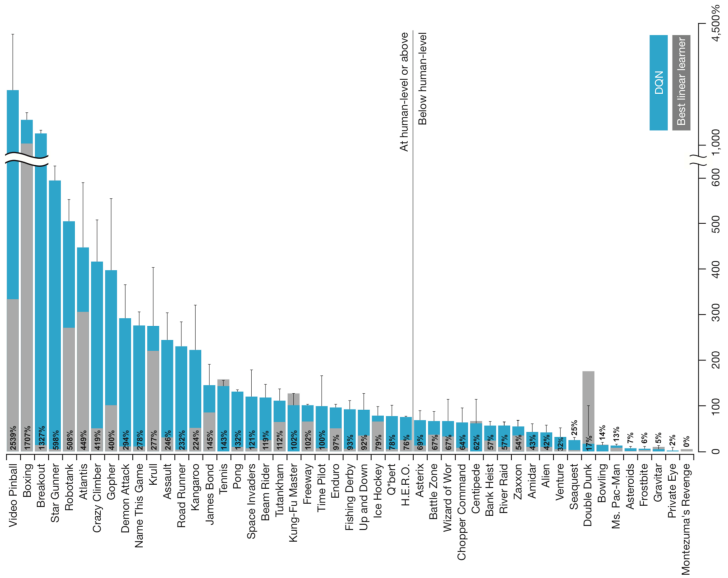


Figure from https://www.youtube.com/watch?v=-TTziY7EmUA

Action Value Function Approximation

- Many successes in simulate-able environments
- Video games
  - Atari 2600, StarCraft II, DotA 2, …



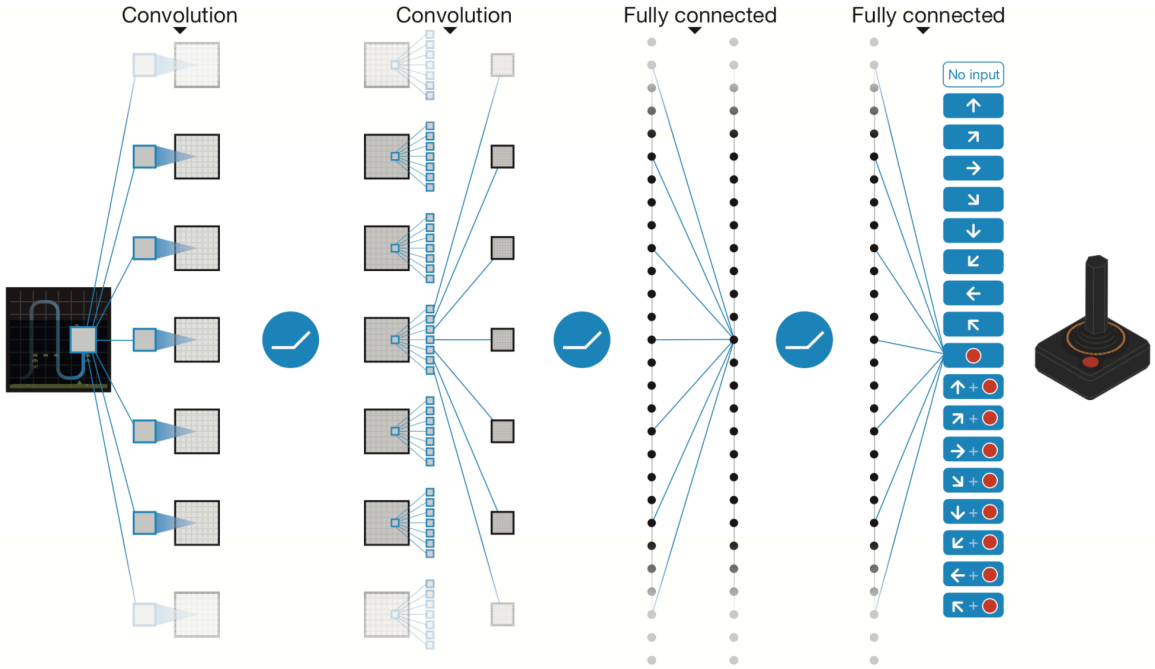
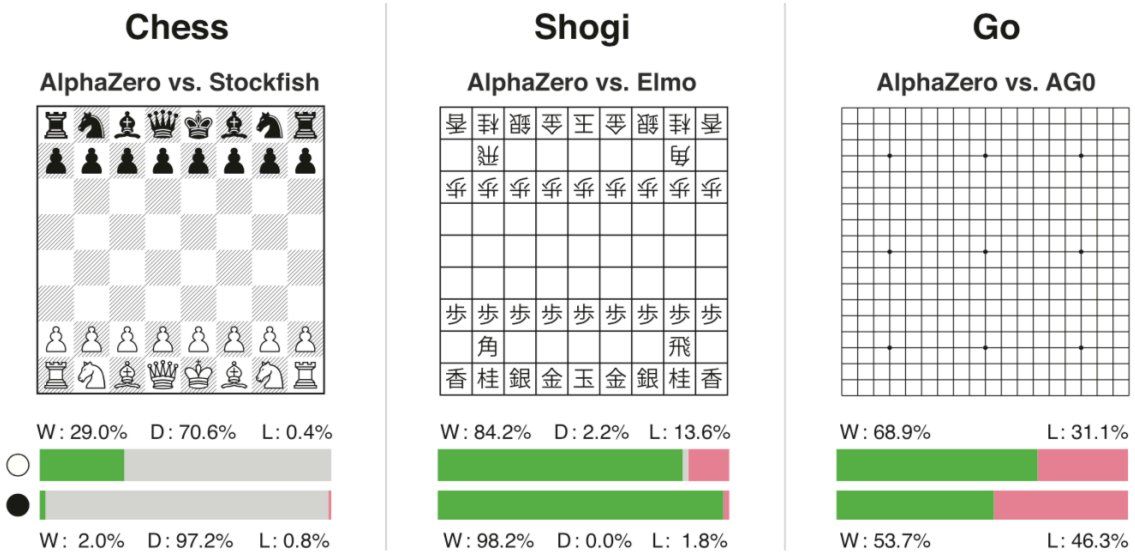
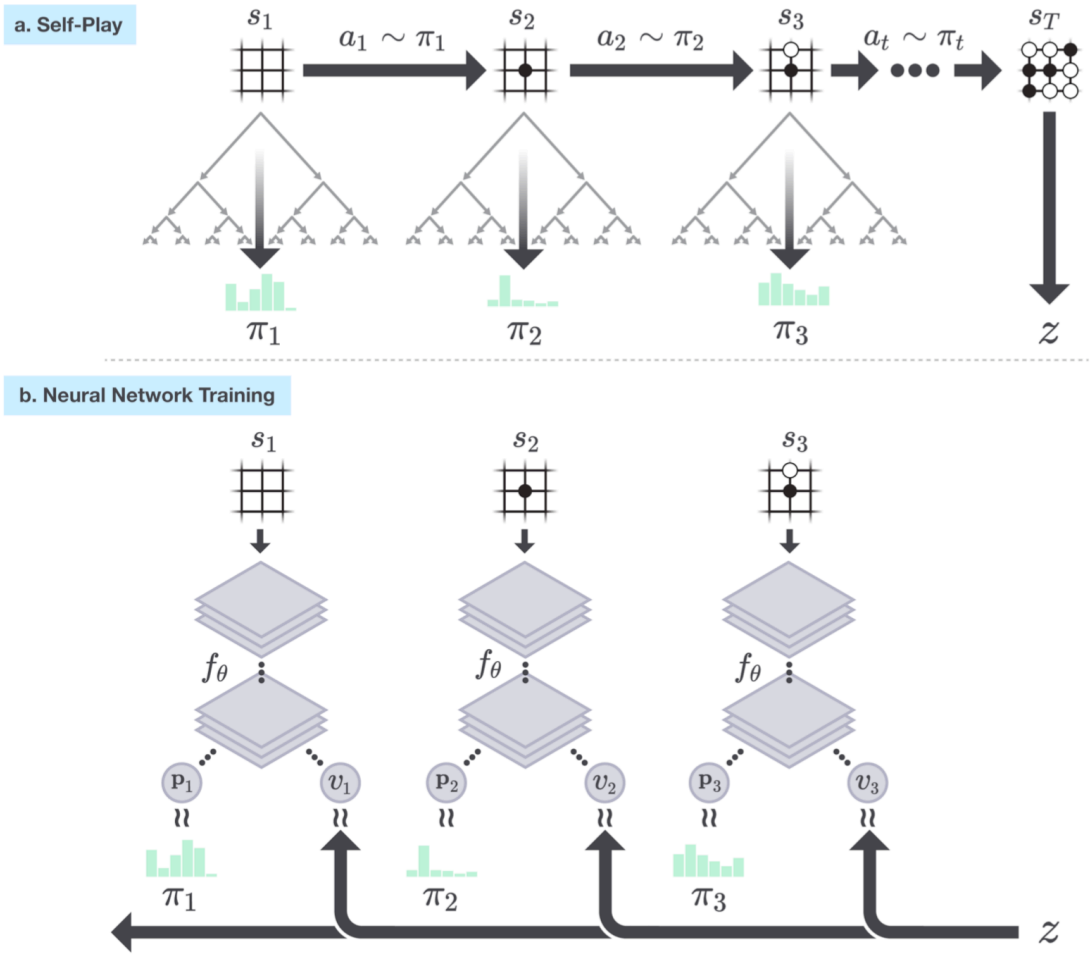


Figure from https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

MCTS Enhanced With xNN Policy And Value

- Many successes in simulate-able environments
- Perfect information 2 player zero sum board games
  - Chess, Shogi, Go, …

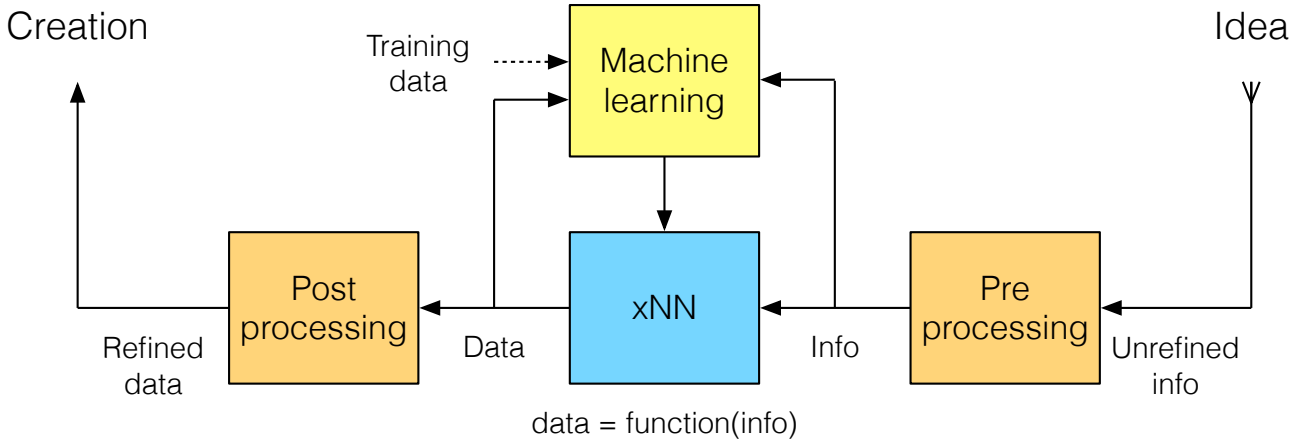


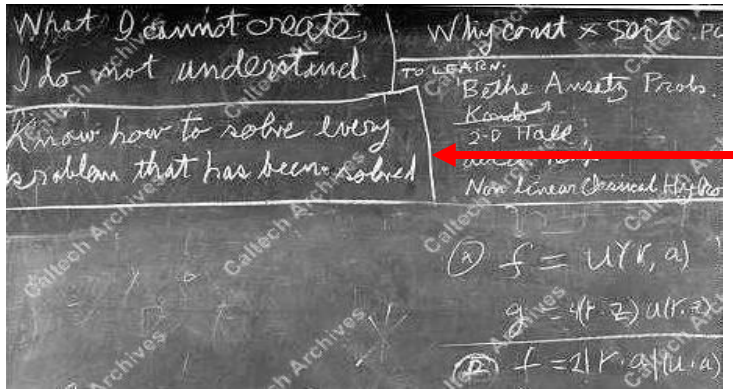


Figures from https://deepmind.com/blog/alphago-zero-learning-scratch/ and https://deepmind.com/research/publications/general-reinforcement-learning-algorithm-masters-chess-shogi-and-go-through-self-play/

Data Generation

- Richard Feynman: “What I cannot create, I do not understand.”
- Generative models are used to create data from information
  - Idea (description) to image / art
  - Idea (topic) to text
  - Idea (text) to speech
  - Idea (genre) to music
- In some cases classical (hand engineered) methods are competitive with or better than learning (xNN) methods





Also: “Know how to solve every problem that has been solved”!!!

Picture from http://archives.caltech.edu/pictures/1.10-29.jpg

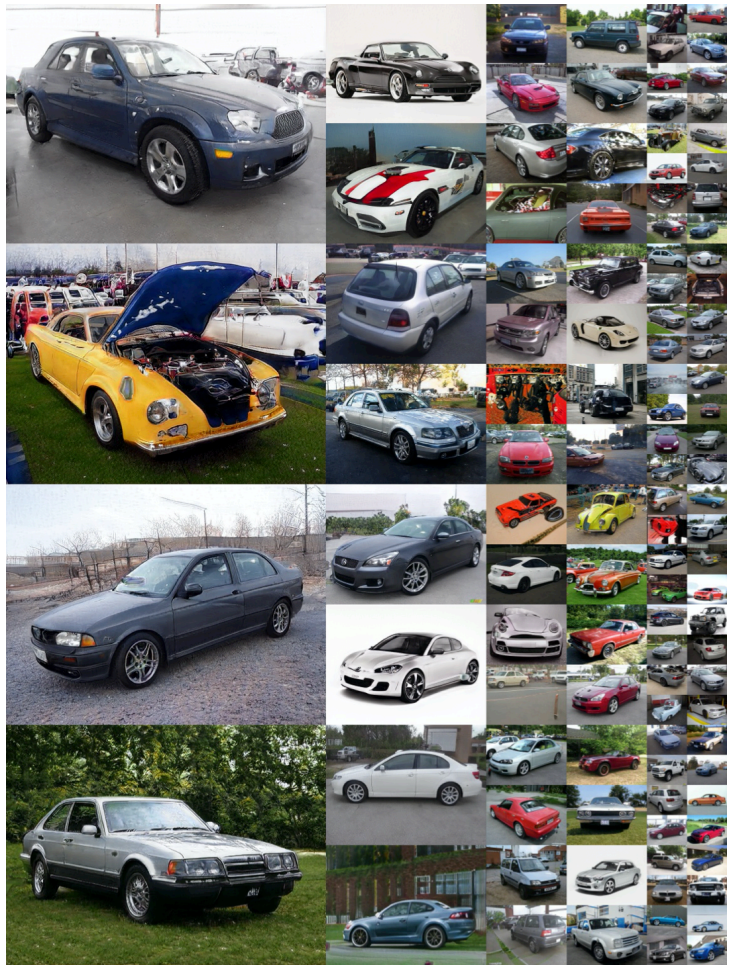
GANs Circa Jun 2014

Figure from https://arxiv.org/abs/1406.2661

GANs Circa Oct 2017

Figure from https://arxiv.org/abs/1710.10196

GANs Circa Mar 2019



Figures from https://arxiv.org/abs/1812.04948

Why Now

All The Stars Are Lining Up

- A confluence of developments
  - More data
  - More compute
  - Better network designs
  - Better training algorithms
  - More efficient community information distribution
  - Applicability to problems of interest
  - More people, companies and funding
- Successes leading to positive feedback

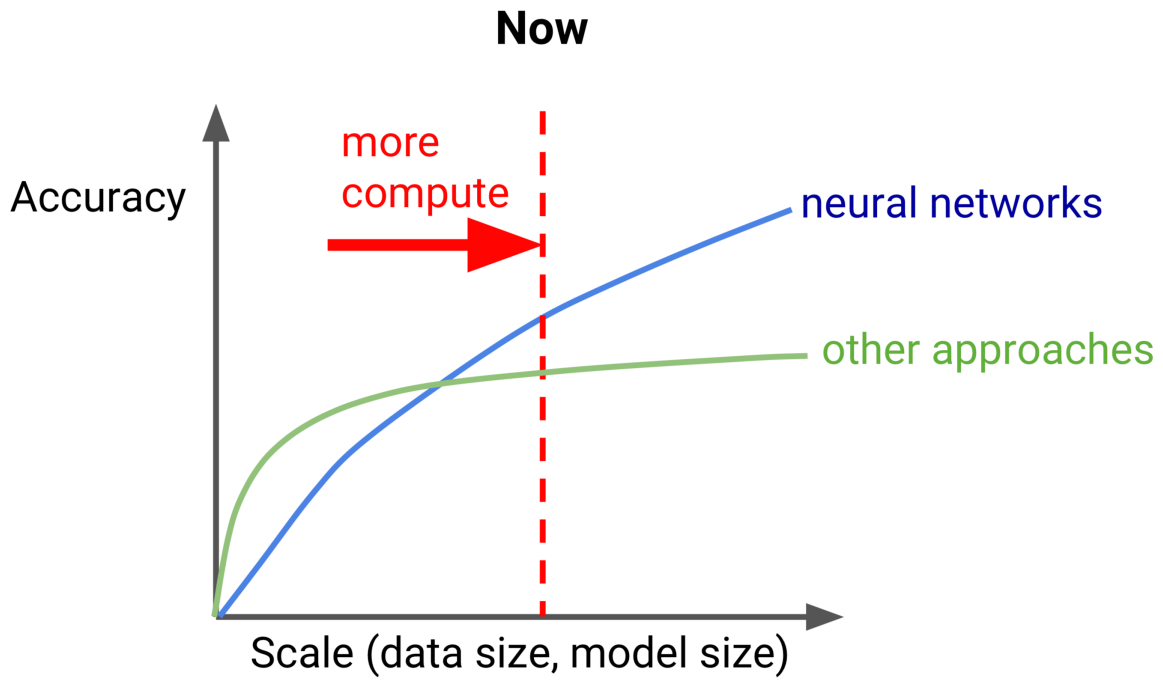
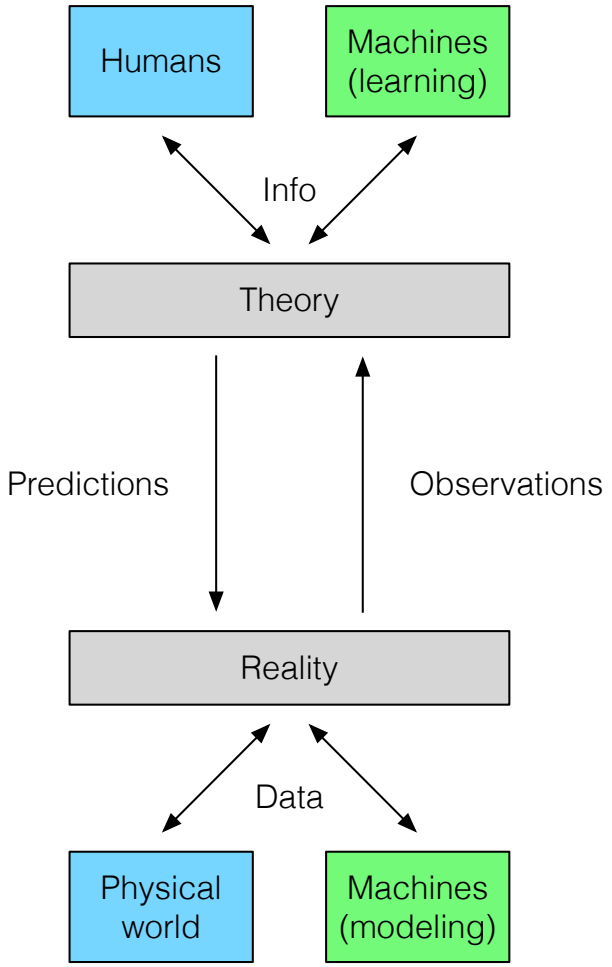


Figure from https://www.hotchips.org/wp-content/uploads/hc_archives/hc29/HC29.22-Tuesday-Pub/HC29.22.69-Key2-AI-ML-Pub/HotChips%20keynote%20Jeff%20Dean%20-%20August%202017.pdf

They Help With Both Theory And Reality

- Loop
  - Theory allows the prediction of reality
  - Reality provides observations that lead to theory
- Reality (the brawn)
  - For most of history was provided by physical systems
  - Computers / algorithms are now commonly used for modeling physical systems and generating observations
    - At 1 point in time computational science was semi controversial
    - Now it's common place and not given a 2nd thought
- Theory (the brain)
  - For most of history was provided by humans
  - Computers / algorithms are starting to become more common for the extraction of information from observations and have the potential to lead to new theory
  - Especially helpful in cases with large amounts of data and complex relationships



Companies Find Internal And External Apps

- Typically think about external applications
  - Ex: build a vision system to put in a product and sell to consumers
- But there's a less visible (no pun intended) trend of more and more internal uses for companies
  - Basically anywhere there's a heuristic being used people are looking to determine if it can be replaced with an xNN trained from input output data
  - Ex: Google Datacenter cooling
  - Ex: compiler heuristics

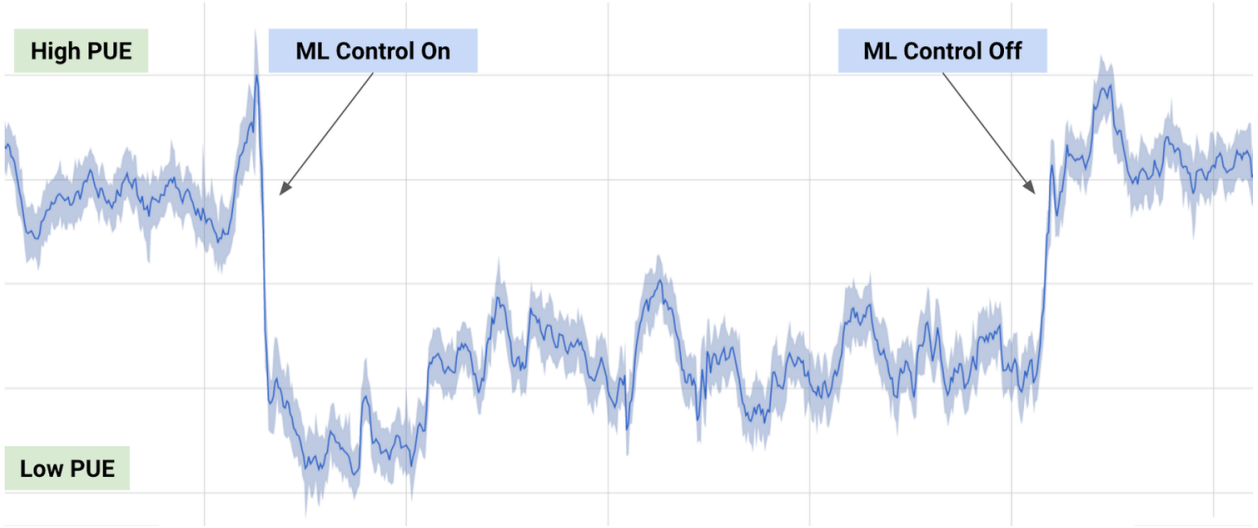


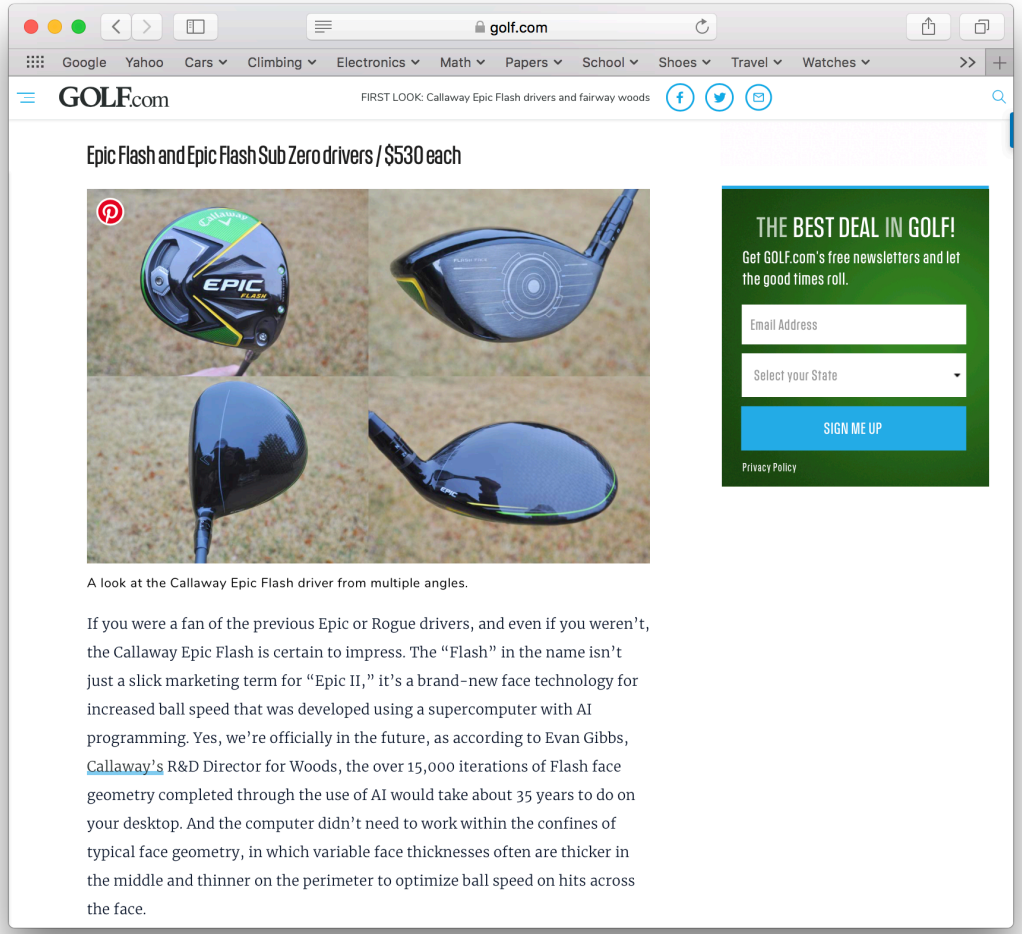
Figure from https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/

They Capture The Imagination

Figure from https://grail.cs.washington.edu/projects/soccer/

They're Hard To Escape

- It feels like AI / ML / xNNs are everywhere …
- Ex: Callaway Epic Flash Driver
  - "It's a brand-new face technology for increased ball speed that was developed using a supercomputer with AI programming."
  - https://www.golf.com/gear/2019/01/04/callaway-epic-flash-drivers-fairway-woods/
- Side note: unfortunately for me the weak link in my golf game is the human and not the club :(



What's In A Name?

- Some people are attracted to neural networks because they sound cool
  - Woo-hoo! We're building robot brains!
- Some people are turned off by neural networks because they sound like fiction
  - I'm not a sucker, I've seen this movie before
- How would you think about neural networks if they were instead called a composition of matrix multiplication, convolution, piecewise linear and transcendental functions?
- A suggestion: don't get hung up on names too much either way as you don't want someone else's words having control over you, instead look at what a technology can do

Mostly useless fact: I'm a big fan of the name Aquaman's parents gave him in the movie

Figure from https://www.vanityfair.com/hollywood/2019/01/aquaman-box-office-biggest-dc-movie

Course Preview

Cooking Vs Baking

- Cooking is ~ an art
  - "In general, cooking is considered an “art” because you are free to change the measurements or ingredients based on your preference”
- Baking is ~ a science
  - "Considered a “science” because it generally calls for accurate and precise measurements of the ingredients"
- Deep learning has room for both
  - But even when we’re cooking **we will still be guided by theory** (similar to how the best chefs cook)

Quotes from https://difference.guru/difference-between-cooking-and-baking/ and figure from http://whiteapronblog.com/2017/06/02/north-carolina-thirteen-layer-cake/

3 Parts To The Course

Math

- Linear algebra
- Algorithms (just a little)
- Probability
- Calculus
- Analysis (just a little)

Networks

- Design
- Training
- Implementation

Applications

- Vision
- Language
- Speech
- Games
- Art

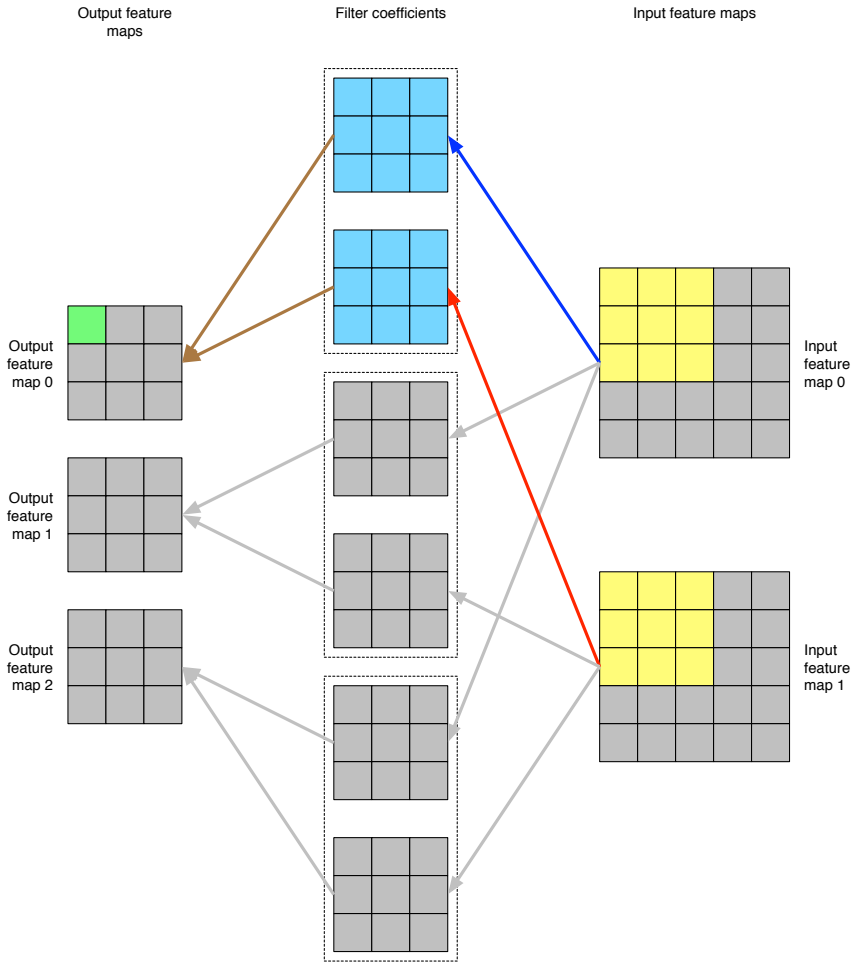
Math: Linear Algebra

- The start of the 1st spiral presentation of material

- The key component of the most important network layers are linear transformations
  - NN (dense or fully connected) layers
  - CNN layers
  - RNN layers
  - Self attention layers
  - Average pooling

- We'll spend our time thinking about how layers combine inputs in different ways to create outputs
  - Data to weak features
  - Weak features to strong features
  - Strong features to classes or values

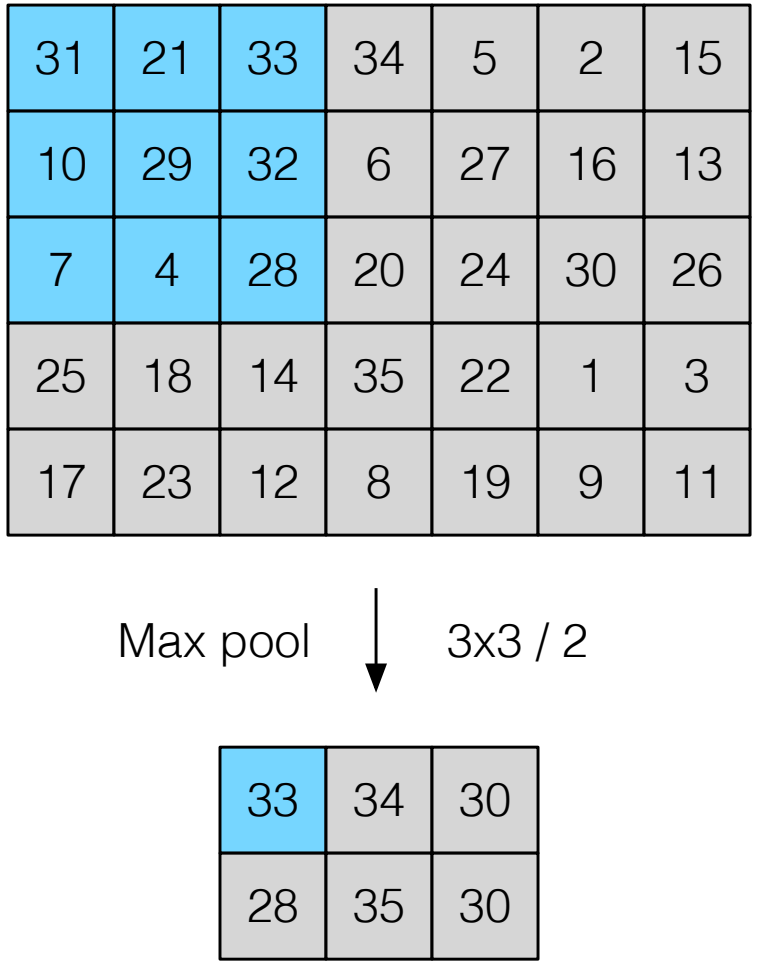
- A laundry list of topics
  - Sets, fields, vectors, matrices, tensors, functions, vector spaces, norms, inner products, matrix multiplication, CNN style 2D convolution, RNNs, attention, average pooling, DFTs and PCA



Math: Algorithms

- Pooling methods, sometimes built on max operations, are used in network design to efficiently increase the receptive field size (how much the network can see)

- A laundry list of topics
  - Comparison sorts, sequential merge sort, parallel merge sort, pooling, median and rank order filtering, max pooling, spatial pyramid pooling, RoI pooling, non maximal suppression



Math: Probability

- Probability is used to modify network designs and create loss functions for training (among many other things)
  - Normalization at the start of the network
  - Batch norm after linear mappings for convergence
  - Dropout after large NN layers for generalization
  - Cross entropy to compare pmfs to determine a loss for optimization
  - Beam search to combine network predictions with external predictions
- Networks can frequently be thought of as a function that maps input to P(output | input)
  - Input = images, sounds, words in language 1, game state, ...
  - Output = classes, graphemes, words in language 2, move, ...
- A laundry list of topics
  - Probability spaces, events, random variables, expected value, normalization, law of large numbers, central limit theorem, random processes, stationarity, time averages, ergodicity, entropy, mutual information, Kullback Leibler divergence, data processing inequality, compression, Huffman and arithmetic coding

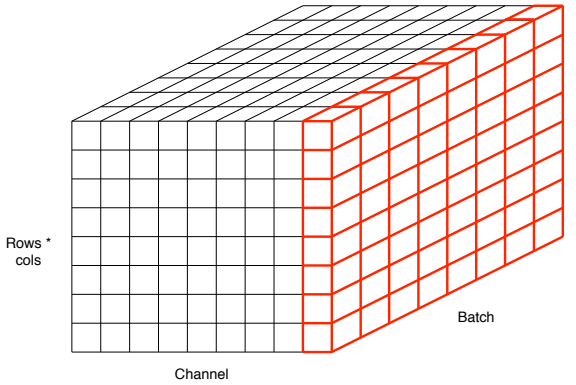
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

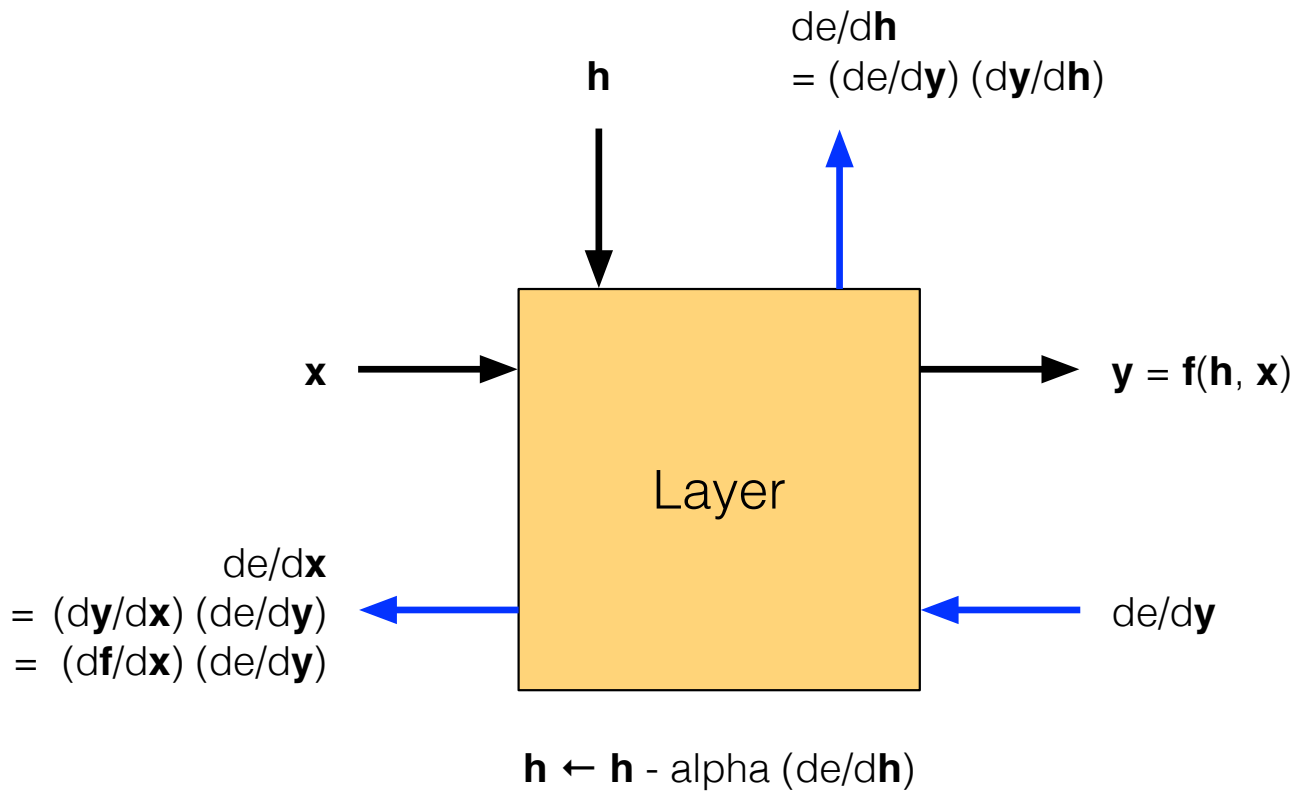
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



Procedure from original paper

Math: Calculus

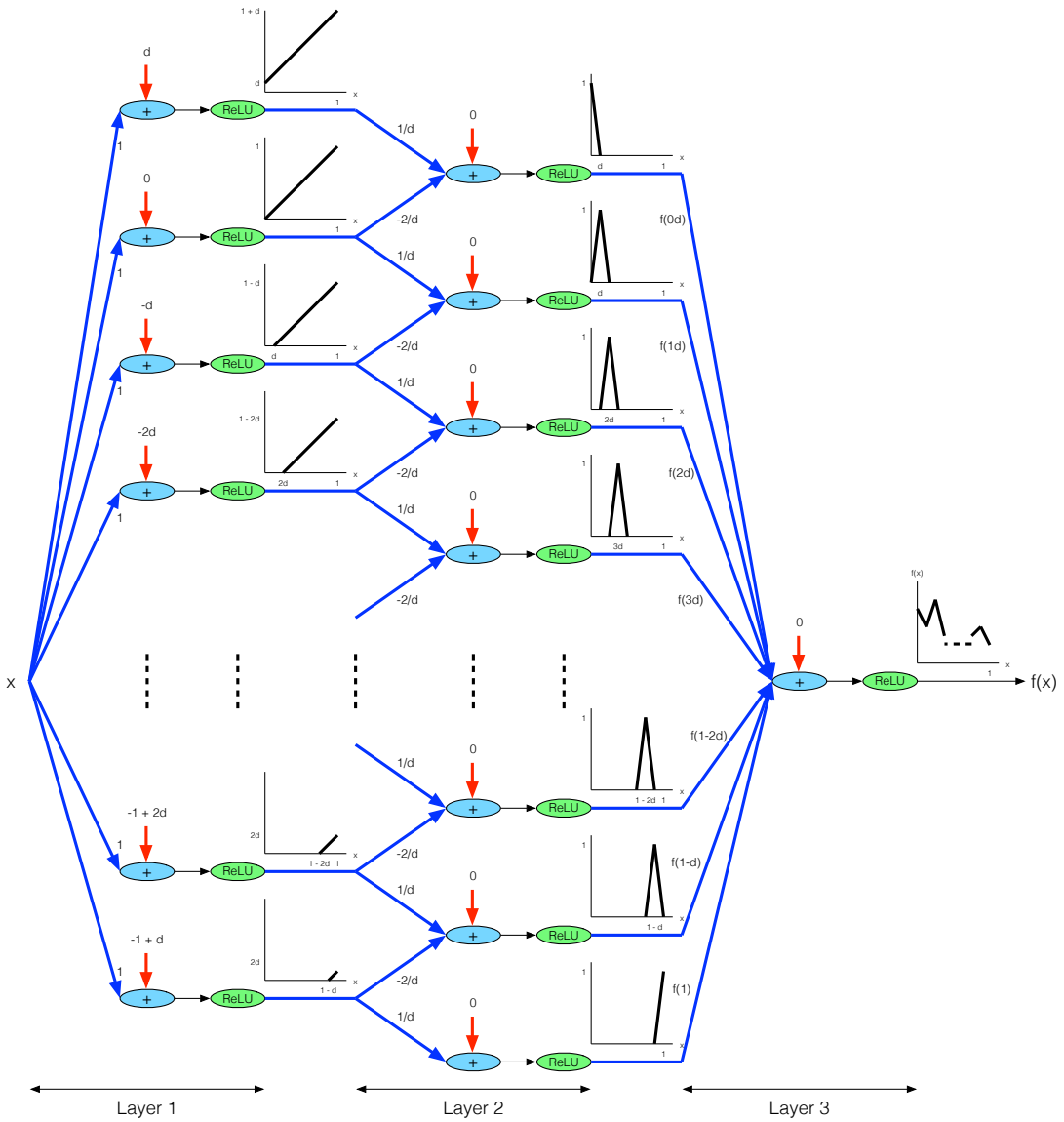
- Calculus is used for training xNNs
  - Automatic differentiation with reverse mode accumulation to back propagate error gradients is an application of the chain rule
  - Gradient descent variants to update learnable parameters
- A laundry list of topics
  - Derivatives, sub derivatives, partial derivatives, gradients, Jacobians, chain rule, critical points, gradient descent, automatic differentiation with reverse more accumulation



Math: Analysis

- Analysis style methods are used to show that under some mild conditions xNNs are universal function approximators

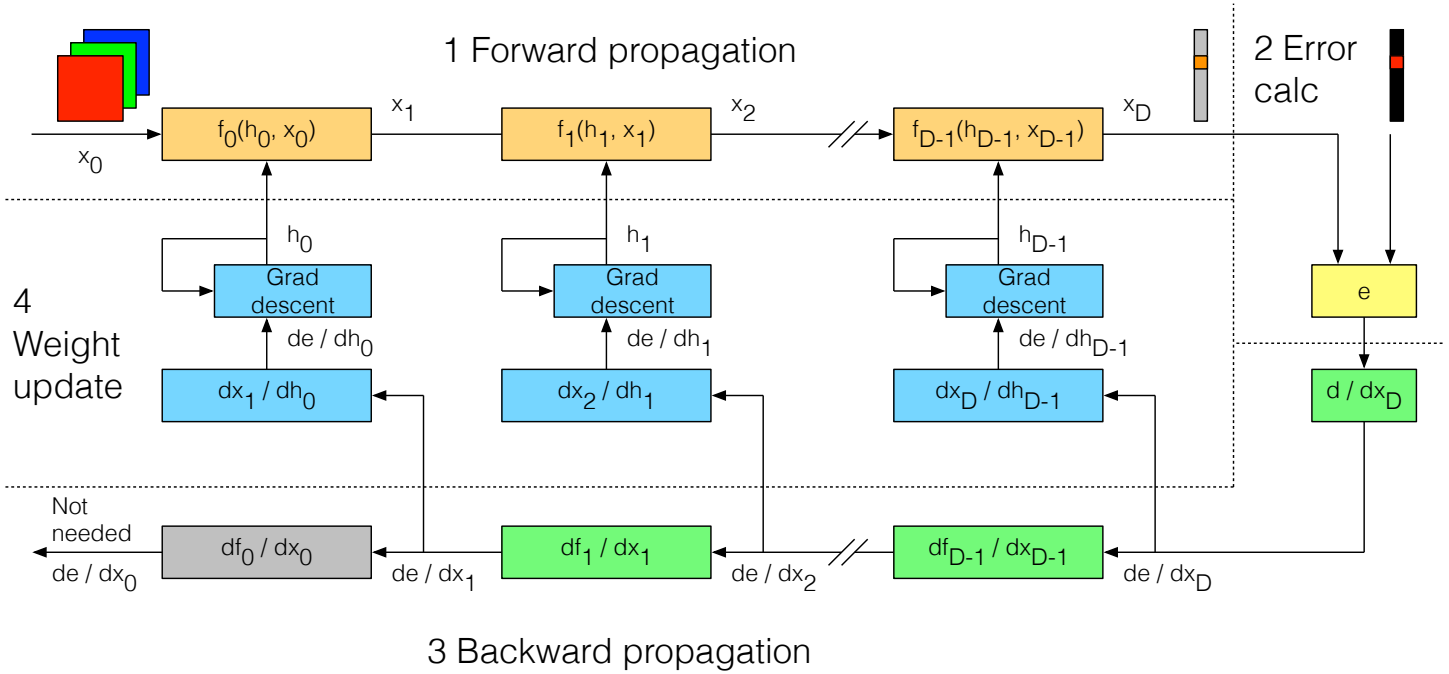
- Thoughts on this
  - It doesn't say a function exists, how to find it if it does exist, how to most efficiently approximate it and if it's the best method to approximate it
  - But it does imply that xNNs are a widely applicable tool for addressing many solvable problems
  - And in practice we'll see that this is true
  - This is important



Math: Big Picture

- Taking stock of the math classes
  - Linear algebra gave us trainable network layers
  - Algorithms gave us max pooling layers to improve the receptive field
  - Probability gave us layers that improve training convergence and generalization
  - Probability also gave us a way to create a loss function
  - Calculus gives us a way of propagating that loss to all layers
  - Calculus gives us a way of updating the weights in the linear transformations
  - Analysis told us this is a reasonable strategy for a lot of problems

- Congratulations, you now have all of the necessary pieces to design and train networks



Networks: Design

- How to design a network to achieve a goal
  - How to select and combine the layers described in the linear algebra, algorithm and probability lectures
  - Encoder decoder style architectures
  - Emphasis on different **encoder** strategies

- A laundry list of topics
  - Goals, size considerations of the network, feature maps and filter coefficients, problem complexity, graph specification, layer types, tail body head decomposition, tail designs, head designs, body designs including chain, parallel, dense and residual structures, optimized architecture search and visualization

- The start of the 2nd spiral presentation of material
  - The math lectures are a 1st look at everything needed to build and train a network
  - The network design, training and implementation lectures look at everything in more detail

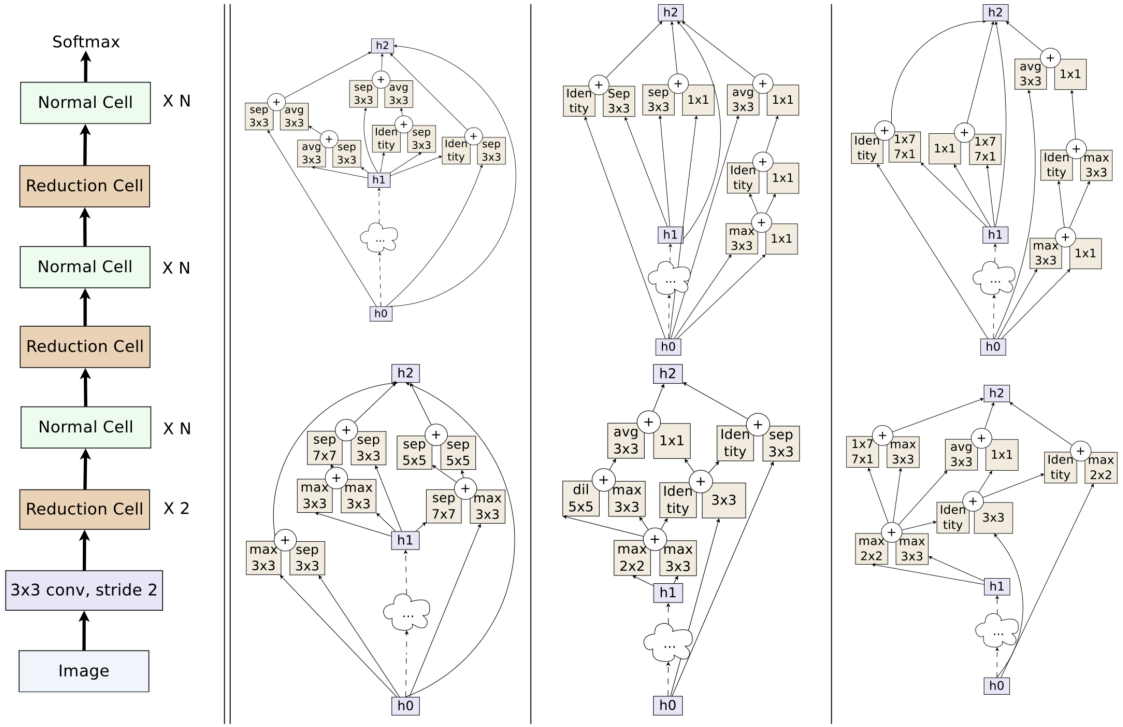
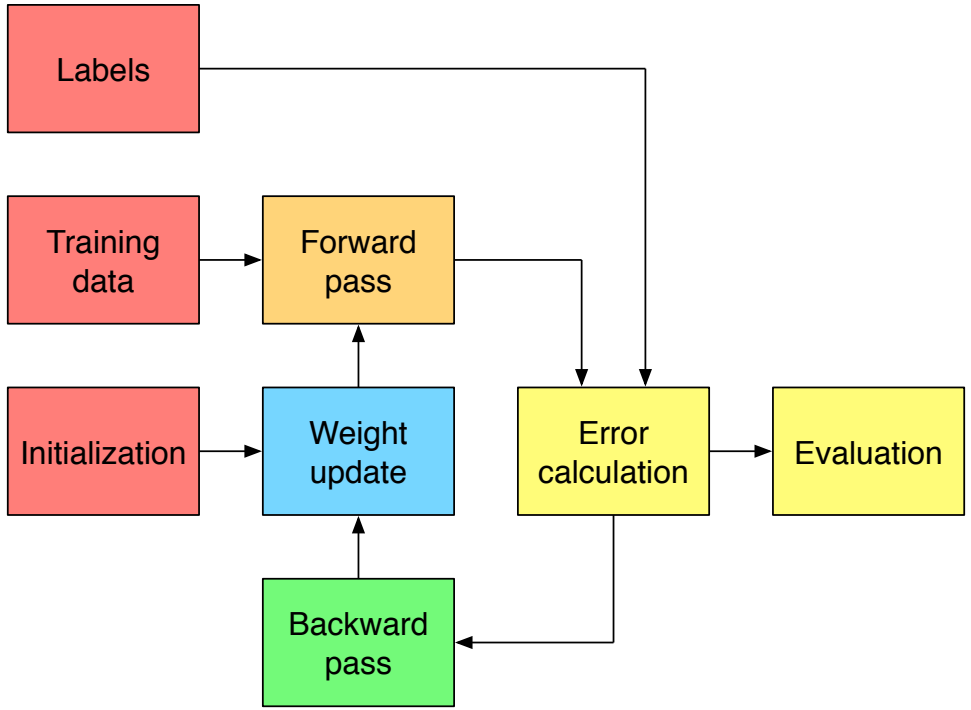


Figure from https://arxiv.org/abs/1802.01548

Networks: Training

- Estimating the weights that control the xNN mapping
  - **Convergence** speed and accuracy of result
  - Regularization to improve **generalization**
  - Builds on the material in the calculus and probability lectures

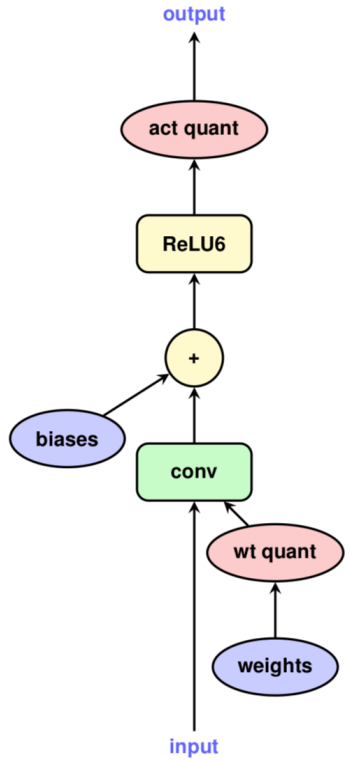
- A laundry list of topics
  - Supervised learning, differences with function optimization, convergence, overfitting, regularization and generalization, the curse of dimensionality, training validation testing data splits, natural data, labeling, cleaning, synthetic data, hand engineered generation, learned generation, data augmentation, random initialization, transfer learning, curriculum learning, batch normalization, group normalization, stochastic width and depth, classification and regression losses, loss surface shapes, unequal class weighting, aux network heads, multiple network heads, check pointing, reversible architectures, batch size, weight update methods including SGD, momentum, AdaGrad, RMSProb and Adam, learning a solver, weight decay, gradient noise, gradient clipping, synchronous stochastic gradient descent and hyper parameter selection



Networks: Implementation

- Bigger networks enable higher accuracy but require more data movement and compute
  - Network modifications to improve implementations, software to connect algorithms to hardware and hardware design

- A laundry list of topics
  - **Networks**: theoretical complexity, precision, hardware size vs model size, training vs testing, data formats, quantization, network sizing and network simplification
  - Software: high level graph specification, high level graph transformations, static vs dynamic graphs, sessions, graph compilers, low level graphs, runtime initialization, runtime execution, software design examples, predicting performance and benchmarking
  - Hardware: Moore's law, Dennard scaling, dark silicon and dark memory, power, roofline models, SoC architectures, domain specific architectures, control, memory, data movement, compression, Amdahl's law, computational basis, matrix multiplication primitive, inner, outer, Strassen style, input power of 2, sparse and analog matrix multiplication, Winograd style convolution, sort primitive, tree configurations, torus configurations and hardware design examples



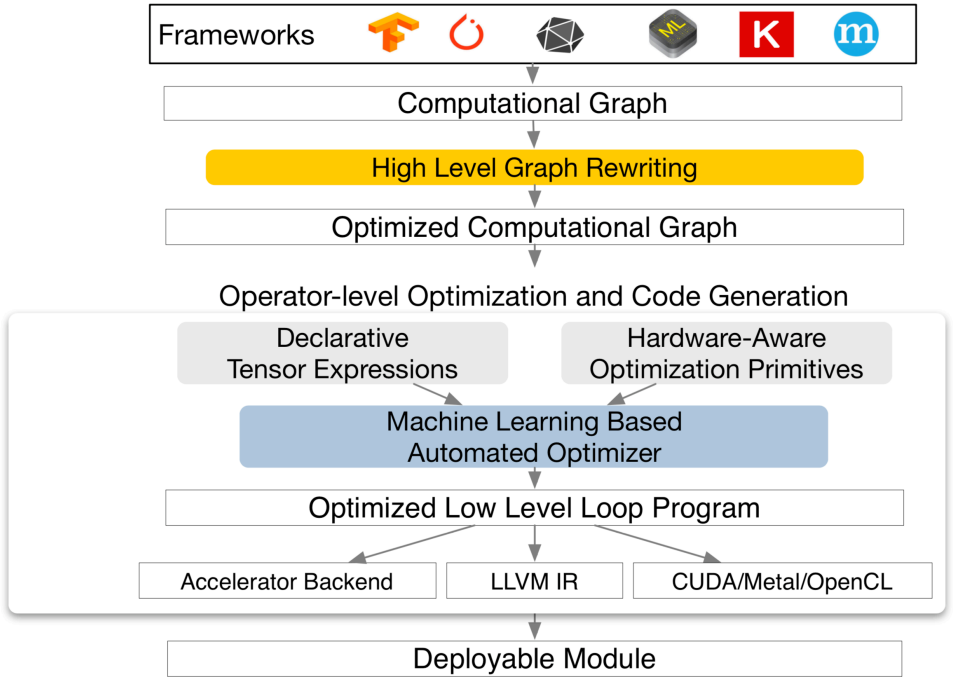
Networks

Figure from https://arxiv.org/abs/1712.05877

Networks: Implementation

- Bigger networks enable higher accuracy but require more data movement and compute
  - Network modifications to improve implementations, software to connect algorithms to hardware and hardware design

- A laundry list of topics
  - Networks: theoretical complexity, precision, hardware size vs model size, training vs testing, data formats, quantization, network sizing and network simplification
  - **Software**: high level graph specification, high level graph transformations, static vs dynamic graphs, sessions, graph compilers, low level graphs, runtime initialization, runtime execution, software design examples, predicting performance and benchmarking
  - Hardware: Moore's law, Dennard scaling, dark silicon and dark memory, power, roofline models, SoC architectures, domain specific architectures, control, memory, data movement, compression, Amdahl's law, computational basis, matrix multiplication primitive, inner, outer, Strassen style, input power of 2, sparse and analog matrix multiplication, Winograd style convolution, sort primitive, tree configurations, torus configurations and hardware design examples



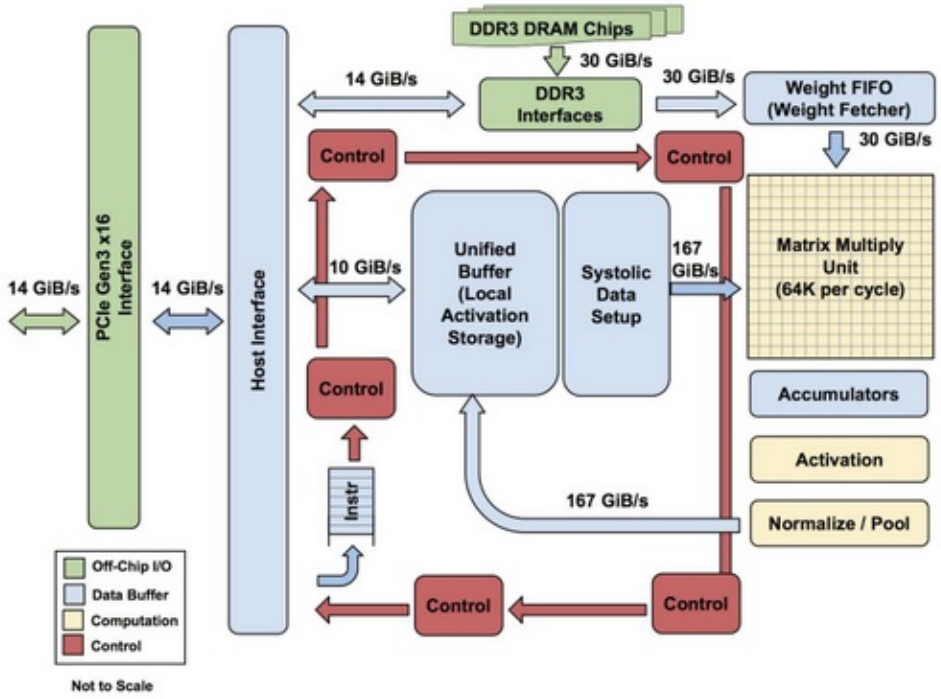
Software

Figure from https://arxiv.org/abs/1802.04799

Networks: Implementation

- Bigger networks enable higher accuracy but require more data movement and compute
  - Network modifications to improve implementations, software to connect algorithms to hardware and hardware design

- A laundry list of topics
  - Networks: theoretical complexity, precision, hardware size vs model size, training vs testing, data formats, quantization, network sizing and network simplification
  - Software: high level graph specification, high level graph transformations, static vs dynamic graphs, sessions, graph compilers, low level graphs, runtime initialization, runtime execution, software design examples, predicting performance and benchmarking
  - **Hardware**: Moore's law, Dennard scaling, dark silicon and dark memory, power, roofline models, SoC architectures, domain specific architectures, control, memory, data movement, compression, Amdahl's law, computational basis, matrix multiplication primitive, inner, outer, Strassen style, input power of 2, sparse and analog matrix multiplication, Winograd style convolution, sort primitive, tree configurations, torus configurations and hardware design examples



Hardware

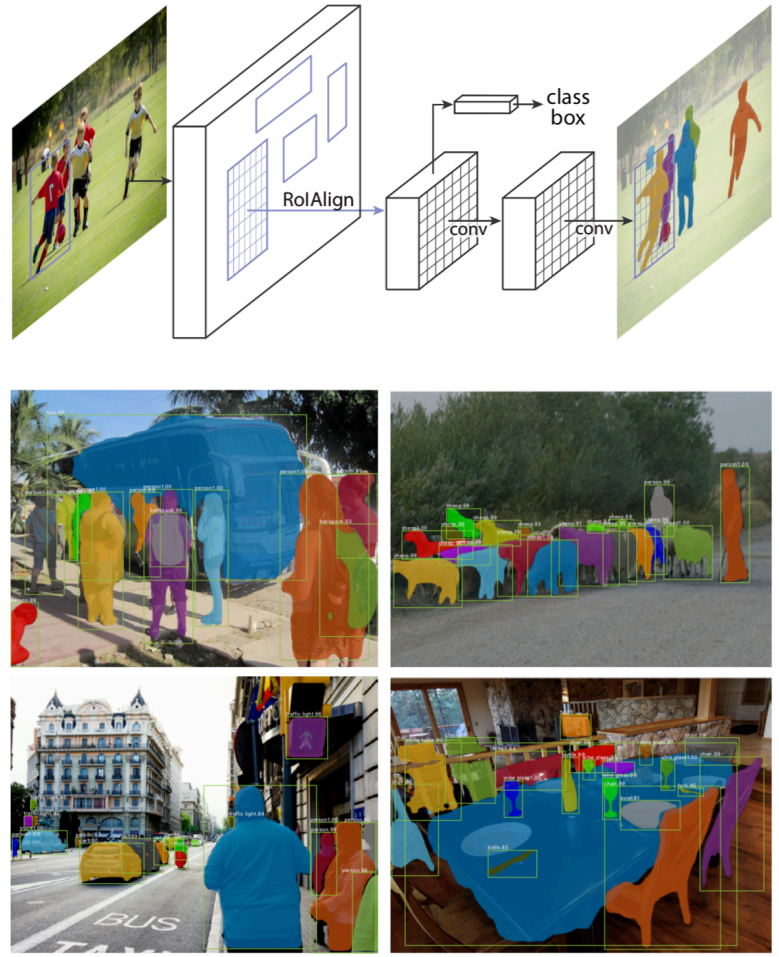
Figure from https://www.nextplatform.com/2017/04/05/first-depth-look-googles-tpu-architecture/

Applications: Vision

- The start of the 3rd spiral presentation of material

- The vision lectures focus on CNNs
  - Exploit spatial structure
  - Transform all sorts of vision applications into classification problems
  - Use CNNs as a function that maps from images to information
  - Encoder **decoder** architectures with special attention paid to the spatial localization of information
  - Image classification, pixel classification, multiple object detection and multiple object segmentation
  - Depth estimation and motion estimation

- A laundry list of topics
  - Image capture and processing, hardware design examples, up sampling, encoder decoder with skip connections, Atrous convolution, spatial pyramid pooling, feature pyramids, anchor boxes, spatial pyramid pooling, RoI pooling, region proposal networks, iterative methods, non maximal suppression, confidence threshold, intersection over union, precision, recall, precision recall curve, 2 and 3 stage approaches to object based segmentation, RoI align, stereo fundamentals and motion fundamentals



Figures from https://arxiv.org/abs/1703.06870

Applications: Language

- The language lectures focus on attention
  - Exploit localized information with attention (but also include CNNs and RNNs)
  - Use attention as a function that maps from language to information
  - Encoder **decoder** architectures with special attention paid to the localization of sequential information
  - Embeddings, language modeling and translation
- A laundry list of topics
  - The distributional hypothesis, SVD based, continuous bag of words, skip grams, visualization, word similarity and analogies, task specific optimization, N grams, neural language models, perplexity, character based, sequence to sequence, greedy and beam search decoding, structured prediction, attention, architecture exploration, self attention, transformer and BLEU

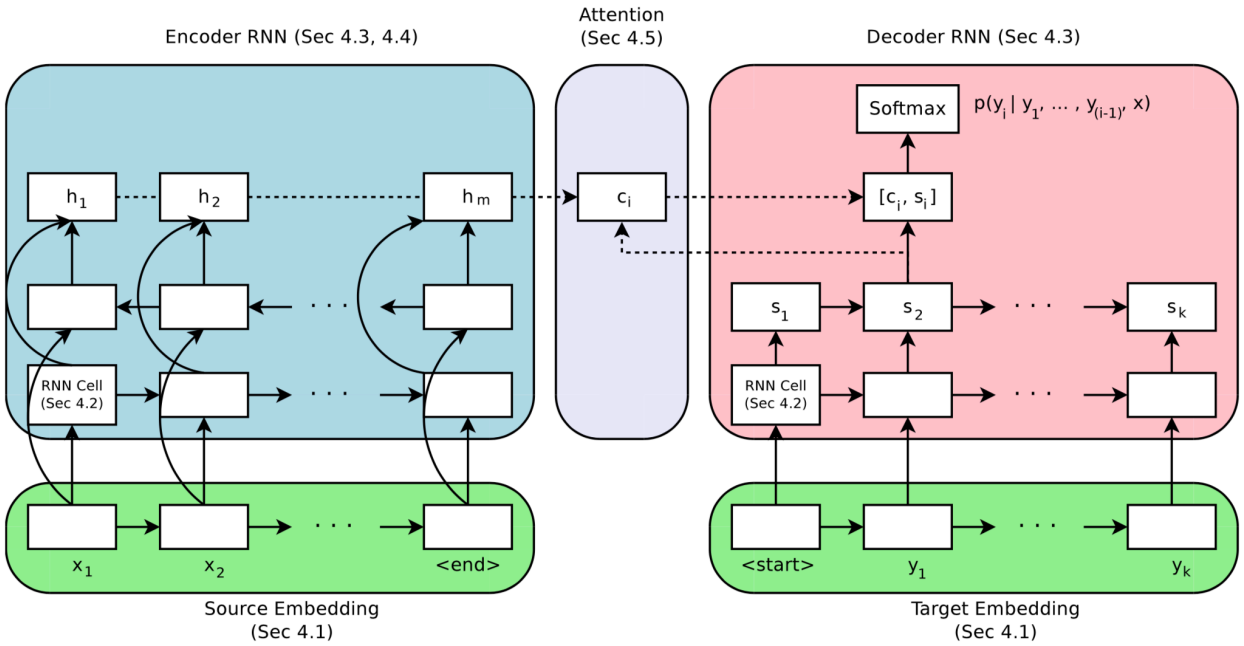


Figure from https://arxiv.org/abs/1703.03906

Applications: Speech

- The speech lectures focus on RNNs
  - Exploit sequential structure with RNNs (but also include CNNs and attention)
  - Use RNNs as a func that maps from sound to info
  - Encoder **decoder** architectures with special attention paid to the monotonic alignment of sequential information
  - Speaker identification, key word spotting, speech to text and text to speech

- A laundry list of topics
  - Speech and audio signal chain, sampling, pre emphasis, windowing and spectrograms, MFCC, RNNs, GRUs, LSTMs, bi directional, pyramidal, sources of variability, wake up, confusion matrix, sequence to sequence models, CTC, beam search, language model, auto segmentation, RNN transducer, attention, transition possibilities, alignments, intermediate representation conversion and audio signal conversion

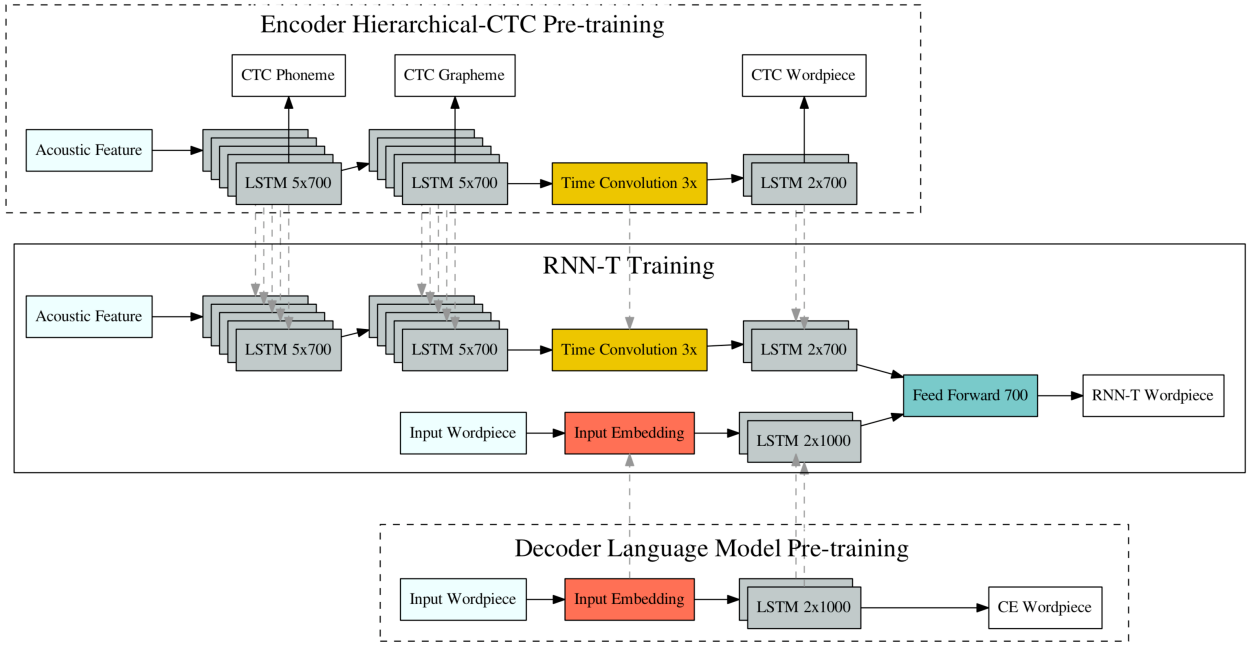


Figure from https://arxiv.org/abs/1801.00841

Applications: Games

- Reinforcement learning
  - Value based
  - Policy based
  - Model based
- Examples
  - Atari
  - Go and chess

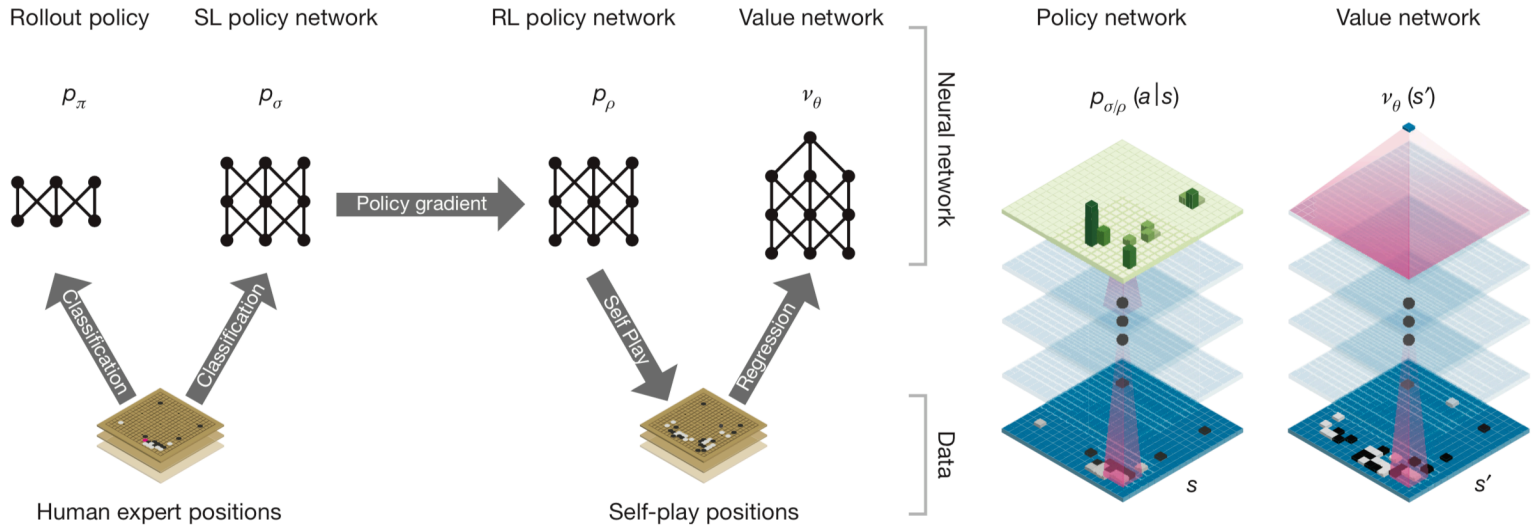


Figure from https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

Depth And Breadth

- This slide title refers to topic coverage, not networks
  - The networks we discuss will always be deep :)
- We're going to talk about a lot of topics
  - Some in depth
  - Some in passing
- Why include the briefly mentioned topics?
  - There's a lot of stuff in this field that's useful to know that it exists and have a basic idea of the why and how it works
  - But it's not necessary or practical to know every detail about everything (trust me, you won't, but that's totally ok)

Not even all of the information on all of the slides will be covered in class or on homework and tests

Then why include it in the slides? It's there if you're interested in it or need it in the future

Figure from https://explorethecanyon.com/faqs/

What's Missing?

- The amount of stuff that's not included >> the amount of stuff that's included
  - Missing sub topics within the topics that are covered
  - Missing topics altogether

- 1 role of a teacher is to be a guide
  - You'll see what I think is most important
  - But your only hope is to learn how to learn
  - The last paper is not written (~ 100 new papers in the area are uploaded to arXiv daily)

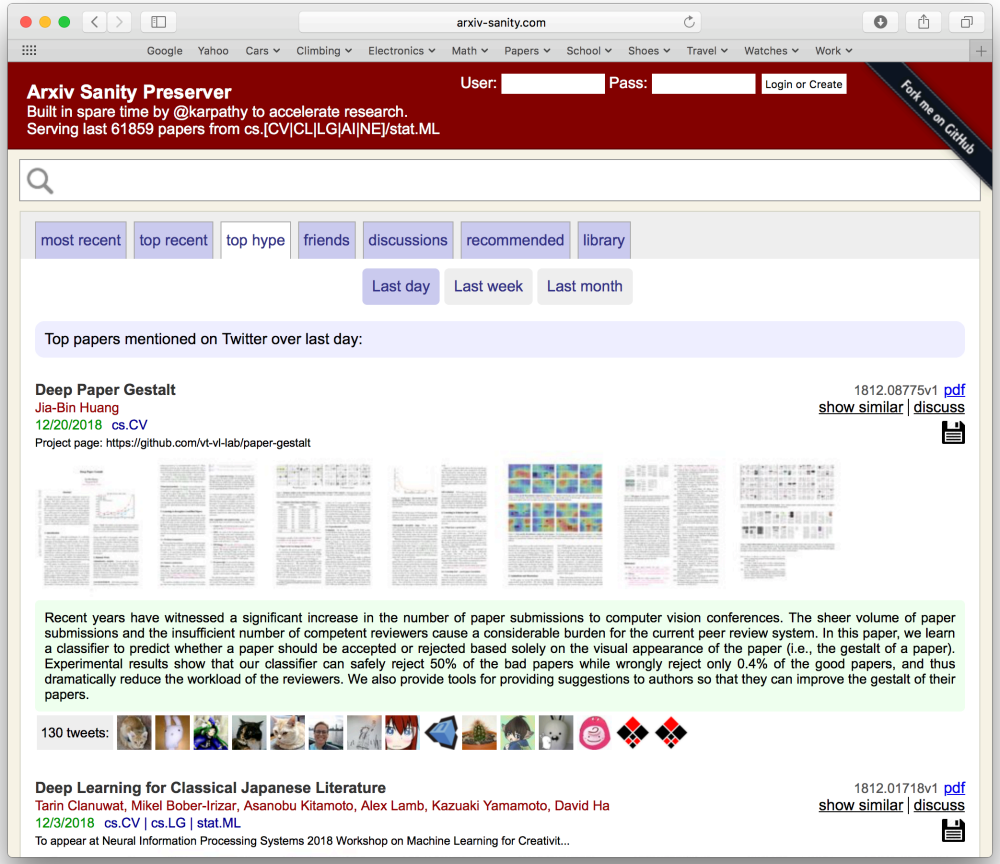
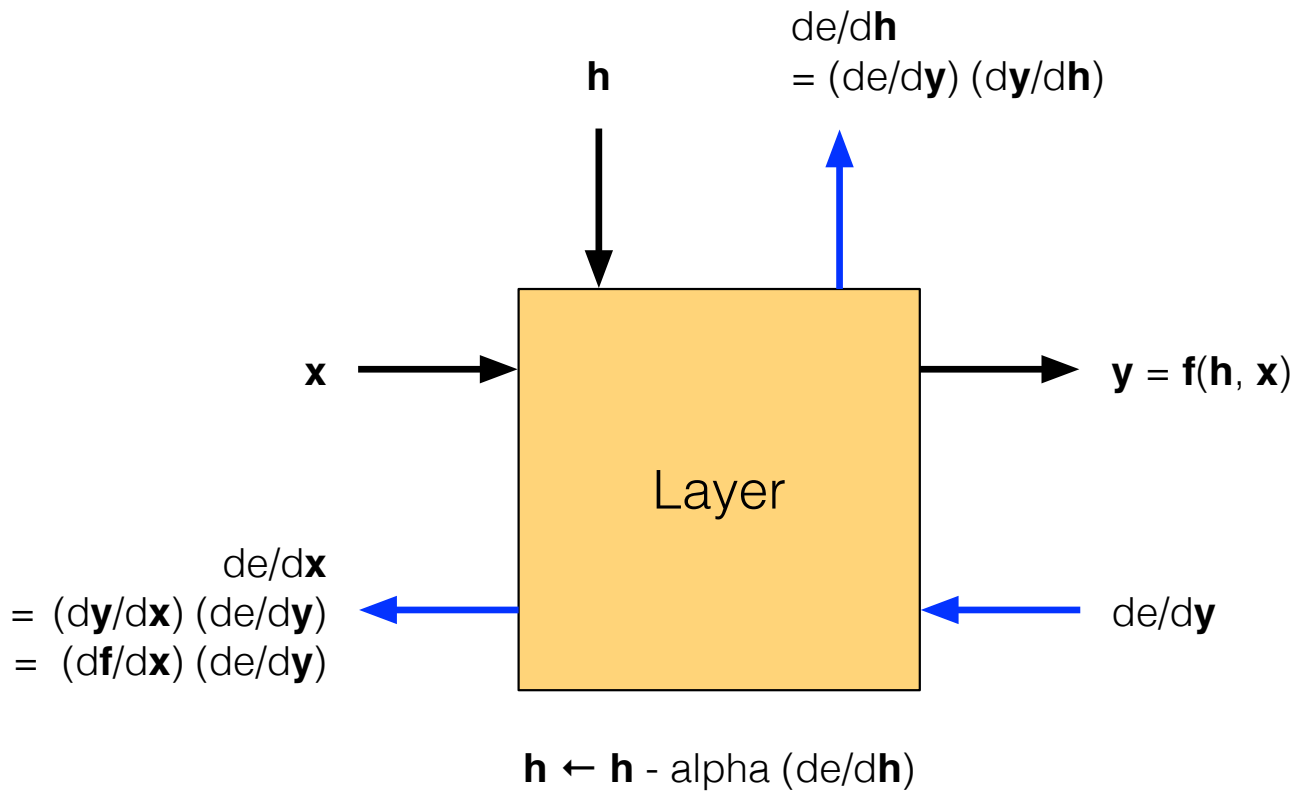


Figure from http://www.arxiv-sanity.com/toptwtr

Themes

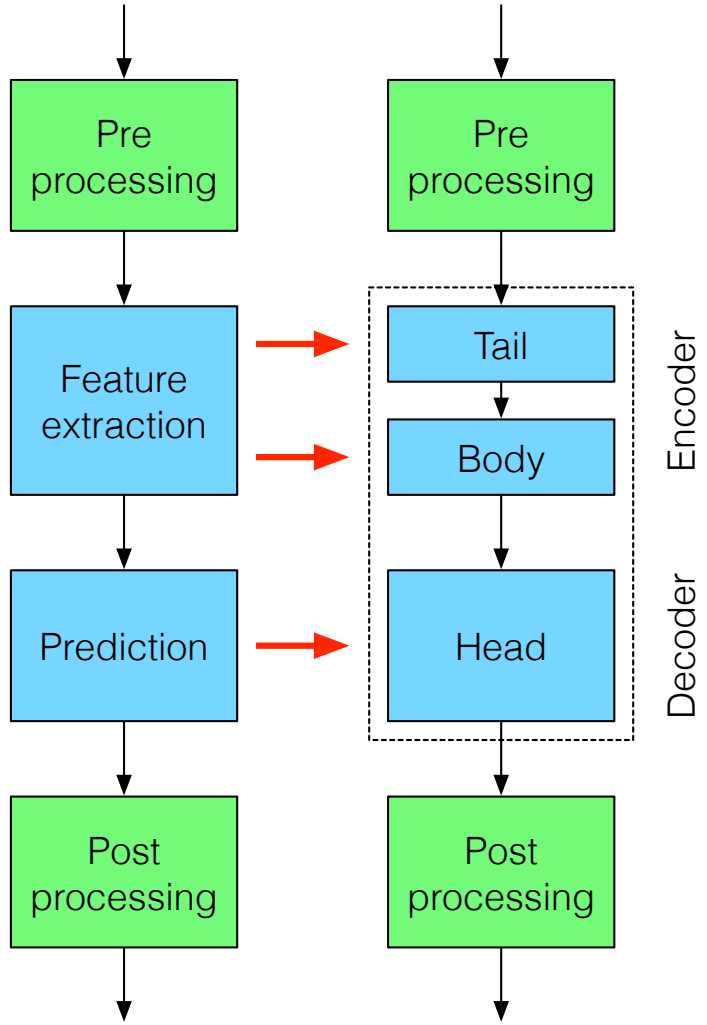
End To End Differentiability

- Every layer that we use for network design is differentiable
  - Can map inputs to outputs
  - Can also map from error gradients at the output to error gradients at the input
- Enables end to end training which seems important for achieving high levels of accuracy
- A different way of thinking about algorithm design and computation
  - Enables automation of training



Encoder Decoder Architectures

- Basic strategy for network design across a wide variety of applications
  - Encoder: data to features
  - Decoder: features to classes / values
- It's common to train an encoder / decoder on 1 problem, replace the decoder with a different decoder, then fine tune it on a different problem
- Examples
  - Vision
    - Images to features
    - Features to classes, objects, depth, …
  - Language
    - Text in language 1 to features
    - Features to text in language 2
  - Speech
    - Sound to features
    - Features to graphemes

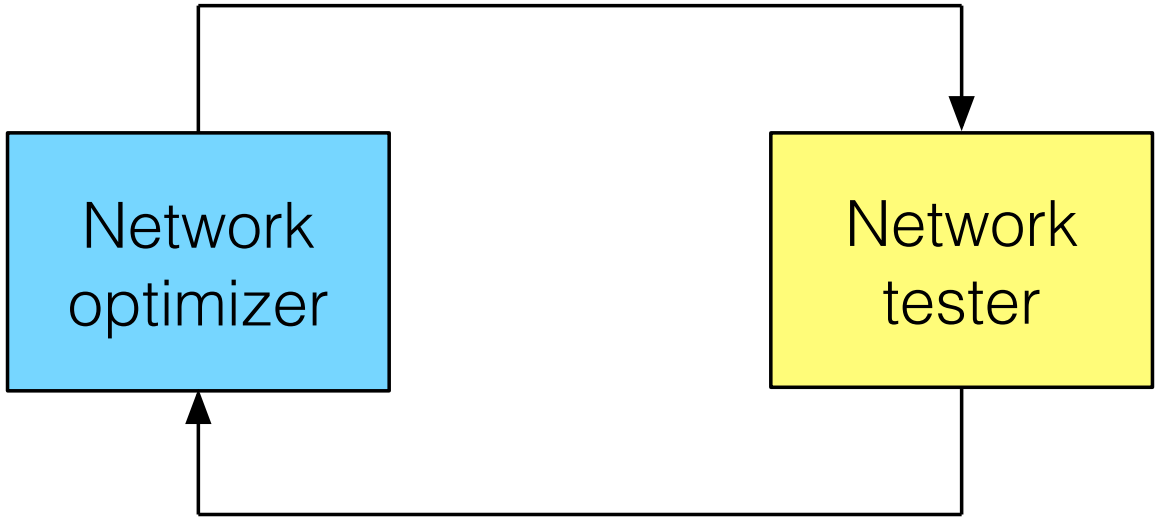


Auto ML

- Humans figure out a set of reasonable ideas, then create an iterative optimization method to refine and improve on the ideas

- Examples
  - Data augmentation
  - Network design
  - Network training
  - Network implementation
  - …

- Survey on automated machine learning
  - https://arxiv.org/abs/1904.12054



Levels Of Abstraction

- We use abstraction to manage complexity
  - Welcome to a computer science class

- Abstraction during different parts of the course
  - During the math part we'll spend most of our time at the individual layer level
  - During the network part we'll start to think in terms of building blocks composed of multiple layers
  - During the applications part we'll think in terms of large parts of a network (e.g., an encoder)

Logistics

Unofficial Course Objections

- Understand why xNNs work
- Understand how to design, train and implement xNNs
- Understand how xNNs are applied to vision, language, speech and games
- Understand how to apply xNNs to new applications
- Learn how to learn in this field

Grades

- 3 in class pencil and paper tests or projects at the end of each section
  - 25% math test (or project if class is online)
  - 25% networks test (or project if class is online)
  - 25% applications test (or project if class is online)
- Homework almost every week
  - 25% total for all assignments
- No final
- No extra credit but a curve if necessary
- eLearning for assignments and grades

This class will go fast

This class will cover a lot of topics

This class will be a lot of work (reading papers, solving problems, writing code, preparing for tests or executing projects)

Are you in a position to devote the necessary time to do your best?

Highlighting The Role Of Homework

- Homework will serve multiple purposes
  - **Reading**: extending in class knowledge and learning to learn via paper reading
  - **Theory**: evaluation and reinforcing of the concepts discussed in class
  - **Practice**: an avenue to gain familiarity with implementations

- Highlighting the implementation component of homework
  - The goal is to get everyone up and running as soon as possible with high level software libraries
  - You build skills via the relatively close reproduction of results of others
  - Which will lead to confidence in your ability to design, train, evaluate and revise networks that contain new ideas of your own

- This semester everyone will use PyTorch for the software implementation portion of their homework assignments
  - This isn't a negative reflection on other libraries (many are excellent)
  - This is a reflection on my bandwidth and need for simplification

Figure from https://pytorch.org

Communication

- UT Dallas email
  - Individual emails
- eLearning
  - Full class emails
  - Discussion board
- GitHub
  - Web site — https://github.com/arthurredfern/UT-Dallas-CS-6301-CNNs
  - Syllabus — Course syllabus with approximate schedule
  - Lectures — Relatively stable, possibly some small updates
  - Videos — Will add throughout the semester; previous semester already available
  - References — Links to many
  - Code — Will add as appropriate

Office Hours

- Will do via WebEx
- I’m also available immediately before (preferred) and for a short time after class when class is in person

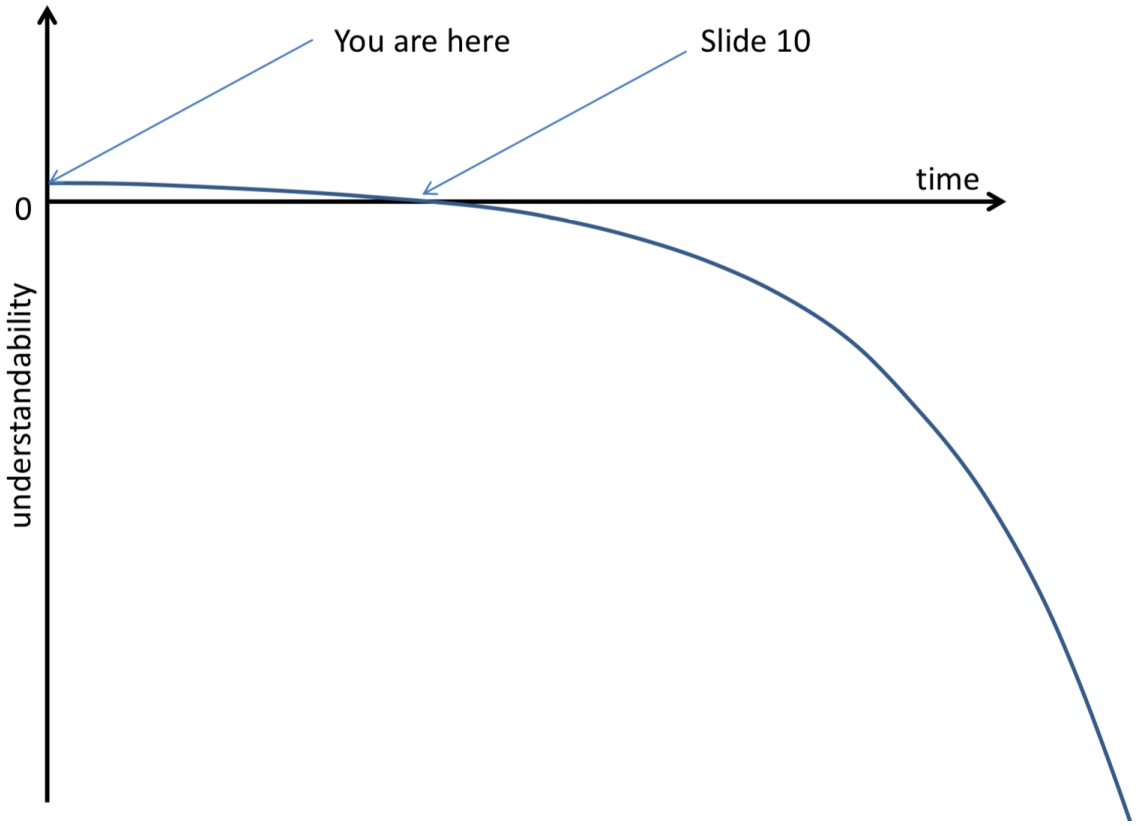
Classroom Citizenship

- Attendance is expected
  - If the class is online and you can attend during the online time, please do so; it contributes positively to a sense of community
- No computers if the class is in person
  - All slides are provided on the class web site
  - Pencil and paper for any additional notes you want to take
- Very very very limited use of mobile phones if the class is in person
  - If it becomes annoying to me I'll restrict it to 0
  - Don't let it become annoying to me

Expectations

Of Me

- A logically laid out plan for both the whole course and individual lectures
- A willingness to modify the plan as needed
- I've taught the class multiple times and have a better idea on how things will go ...
  - ... but I'm always making changes to try and improve things which re introduces some ambiguity



I'll try and avoid this style of lecture (side note: this is 1 of the funniest / best figures I've ever seen in a presentation)

Figure from http://deeplearning.cs.cmu.edu/slides/lec23_24.RL.pdf

Of Me

- My best every class
- My opinions
  - It’s a special topics class
- That I’ll speak to adults like adults
  - It's a grad class
  - I want to be precise
  - But I don't want to make things unnecessarily complicated

Of You

- Honesty
  - In your work
  - In your interactions with other students
  - In your interactions with me
- Hard work
  - Nothing meaningful in life is easy
  - **I promise you this won't be an exception**
- Preparation
  - Review the lecture on your own before class
  - Listen to the lecture in class
  - Re review the lecture after class until it makes sense

My sources say no. This course covers a lot, some of it will be easy for you, some of it will be difficult. If you find yourself stuck on something, don't stress too much, come talk to me and we'll figure it out together.

Figure from https://i.redd.it/a5fk5dg6yflz.png

Of You

- Contribute to a friendly environment
  - It's great to shine as an individual through individual accomplishments
  - It's great to shine by helping others shine
    - This is a characteristic of a leader
- Be engaged
  - **Ask questions freely**
  - Speak up if something is unclear
  - Correct me if I'm wrong
- Help me learn your name
  - Say it when you ask a question

Opportunities

A Suggestion

- Look for opportunities beyond this course
  - Your own research
  - Your own company
  - UT Dallas HackAI or similar events (side note: I'd like a team of you to win this every time)
  - …
- At the end of this course you'll have a tool that can map from all sorts of different inputs to all sorts of different outputs, a way to train it and a way to implement it
  - What can you do with something like this?
- Want to bounce ideas off of someone? Come talk to me

Questions?

References

Recommendation

- For a curated list of generally useful references that apply to all parts of this course see
  - https://github.com/arthurredfern/UT-Dallas-CS-6301-CNNs/tree/master/References

- The references at the above web site are loosely divided into math, networks and applications, though there is a lot of overlap
  - The references tend to be relatively general and many are links to related courses with overlapping information (you're encouraged to check these out, they're excellent)
  - References that are more specific / focused in nature are listed in and at the end of each set of slides as appropriate