ImageNet Classification with Deep Convolutional Neural Networks

Michael Schatz
Oct 2, 2024

JHU EN.601.449/EN.601.649: Applied Comparative Genomics

Assignment 3 Due Monday September 30

Files

main

Go to file

- assignments
  - assignment1
  - assignment2
  - assignment3
    - input_files
    - README.md
- lectures
- policies
- LICENSE
- README.md

Preview | Code | Blame 163 lines (101 loc) · 8.33 KB

Raw

Assignment 3: BWT and Variant Calling

Assignment Date: Monday, September 23, 2024
Due Date: Monday, September 30, 2023 @ 11:59pm

Assignment Overview

In this assignment you will implement the BWT and explore the requirements for variant calling. The programming exercises can be computed in any programming language, although we recommend python (or C++, Java, or Rust). R is generally inefficient at string processing unless you take great care. See the resources at the bottom of the page for tips for the variant calling exercises.

As a reminder, any questions about the assignment should be posted to Piazza.

Question 1. BWT Encoding [20 pts]

In the language of your choice, implement a BWT encoder and encode the string below. Faster (Linear time) methods exist for computing the BWT, although for this assignment you can use the simple method based on standard sorting techniques. Your solution does *not* need to be an optimal algorithm and can use O(n^2) space and O(n^2 lg n) time.

Here is the recommended pseudo code (make sure to submit your code as well as the encoded string):

```
computeBWT(string s)
  ## add the magic end-of-string character
  s = s + "$"

  ## build up the BWM from the cyclic permutations
  ## note the ith cyclic permutation is just "s[i..n] + s[0..i]"
  rows = []
  for (i = 0; i < length(s); i++)
```

Recap

“AlexNet”

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

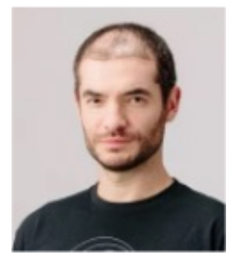
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

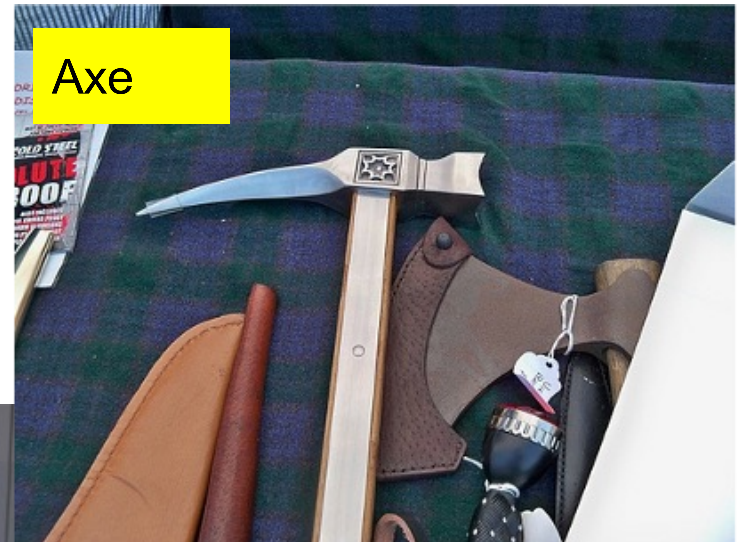
Alex Krizhevsky
U. Toronto/Google



Ilya Sutskever
U. Toronto/OpenAI/
Safe Superintelligence Inc

Geoffrey Hinton
U. Toronto/Vector Institute

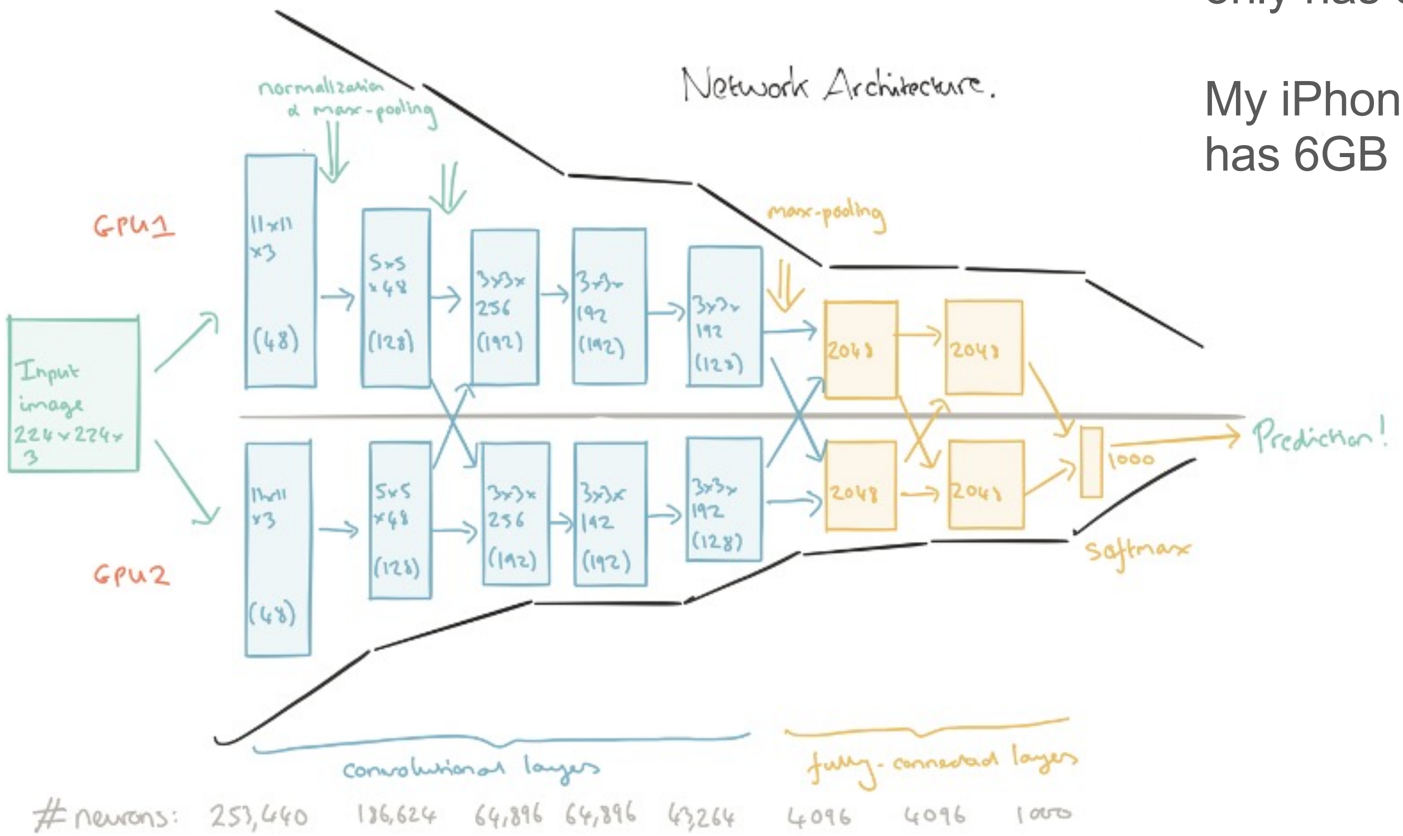
What is this?



Network Architecture (annotated)

GTX 580 GPU
only has 3GB!

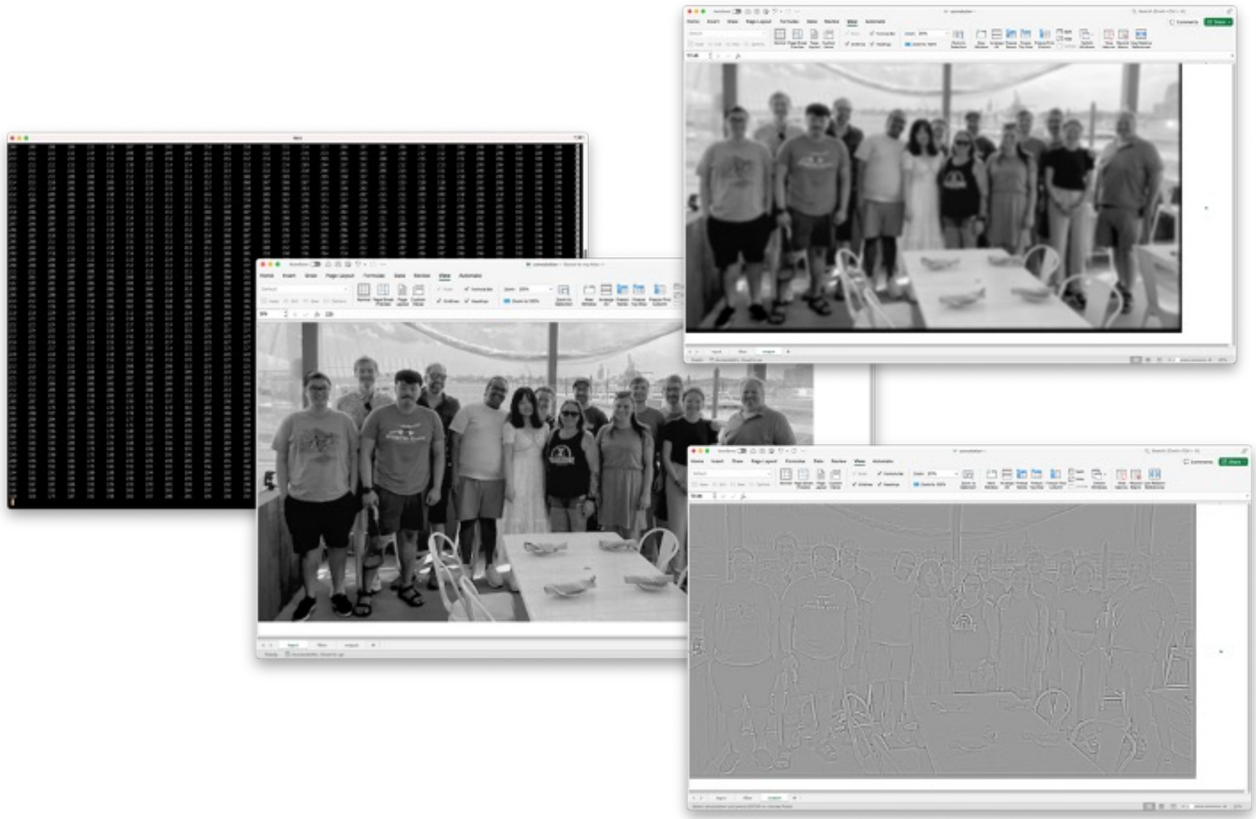
My iPhone's GPU
has 6GB



https://blog.acolyer.org/2016/04/20/imagenet-classification-with-deep-convolutional-neural-networks/

Kernels

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	



= input!A1 * filter!$A$1 + input!A2 * filter!$A$2 + input!A3 * filter!$A$3 + input!A4 * filter!$A$4 + input!A5 * filter!$A$5 +
input!B1 * filter!$B$1 + input!B2 * filter!$B$2 + input!B3 * filter!$B$3 + input!B4 * filter!$B$4 + input!B5 * filter!$B$5 +
input!C1 * filter!$C$1 + input!C2 * filter!$C$2 + input!C3 * filter!$C$3 + input!C4 * filter!$C$4 + input!C5 * filter!$C$5 +
input!D1 * filter!$D$1 + input!D2 * filter!$D$2 + input!D3 * filter!$D$3 + input!D4 * filter!$D$4 + input!D5 * filter!$D$5 +
input!E1 * filter!$E$1 + input!E2 * filter!$E$2 + input!E3 * filter!$E$3 + input!E4 * filter!$E$4 + input!E5 * filter!$E$5

Learned Convolutional Kernels

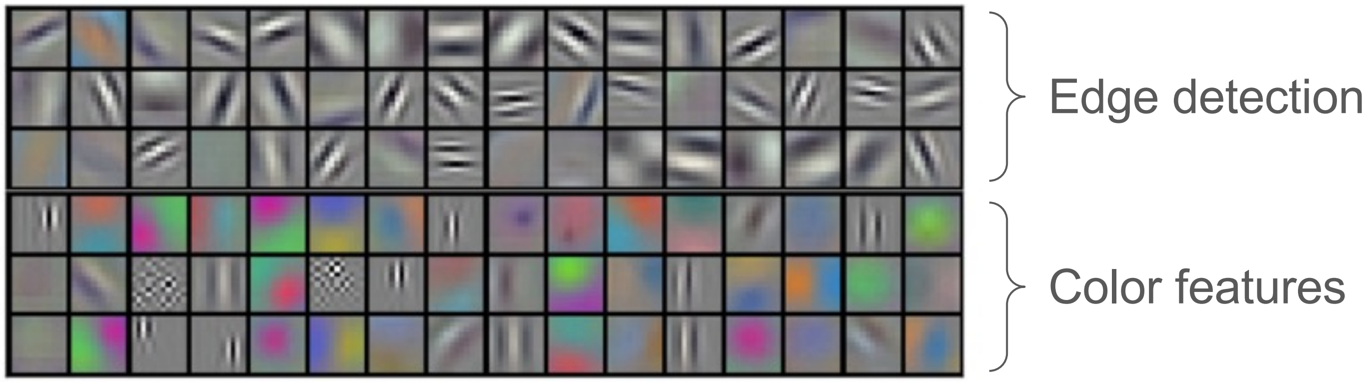
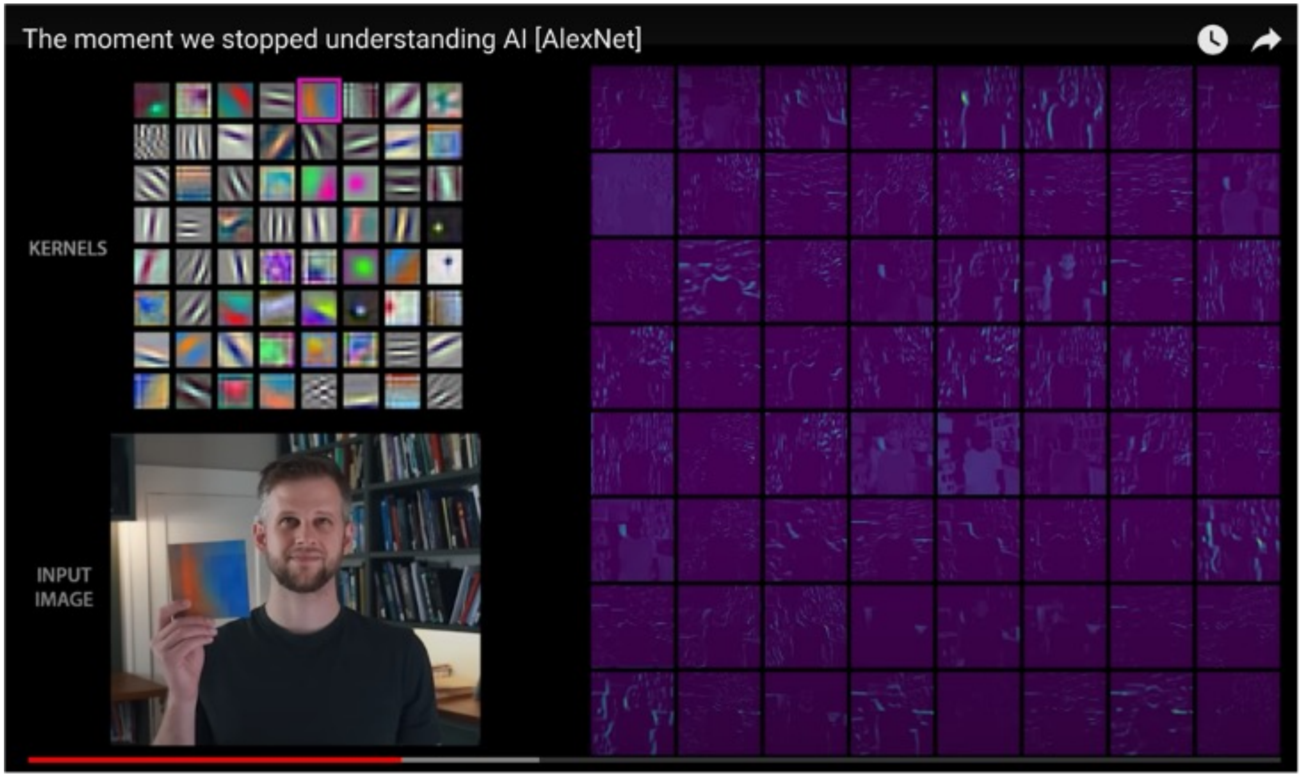


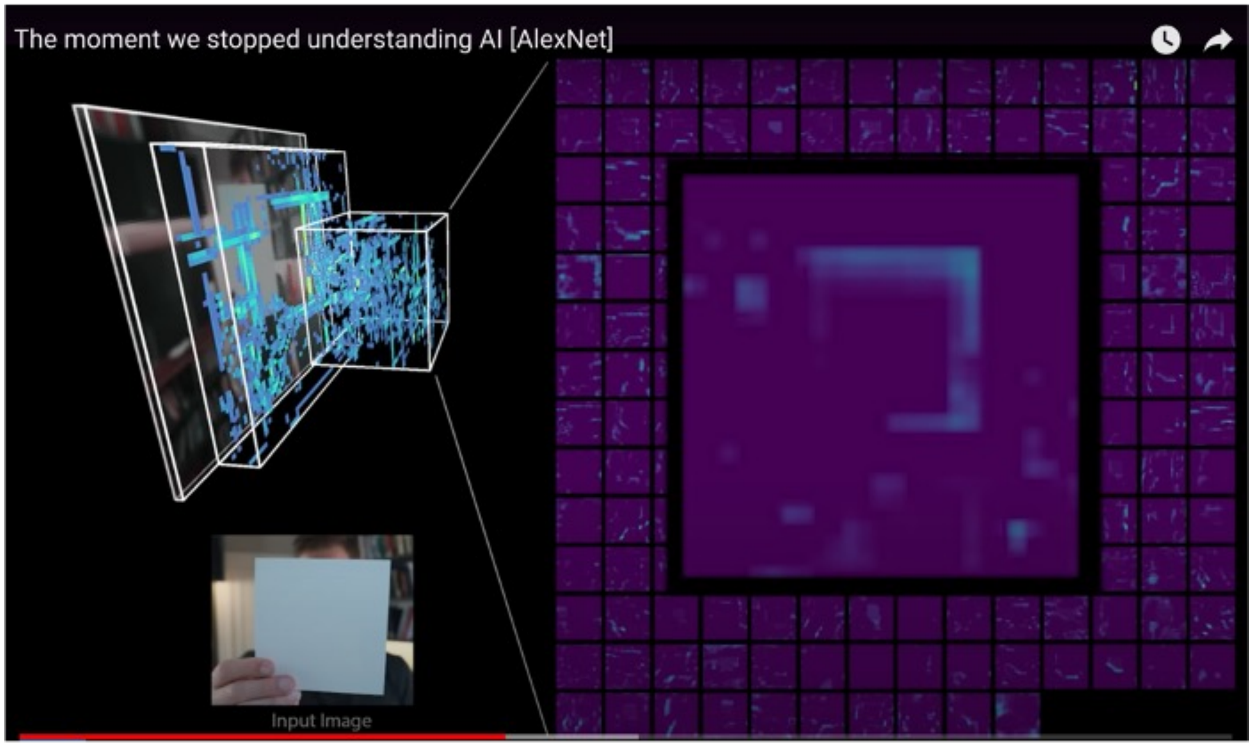
Figure 3: 96 convolutional kernels of size $11\times11\times3$ learned by the first convolutional layer on the $224\times224\times3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

From 1 image to 96 images: Activation Maps



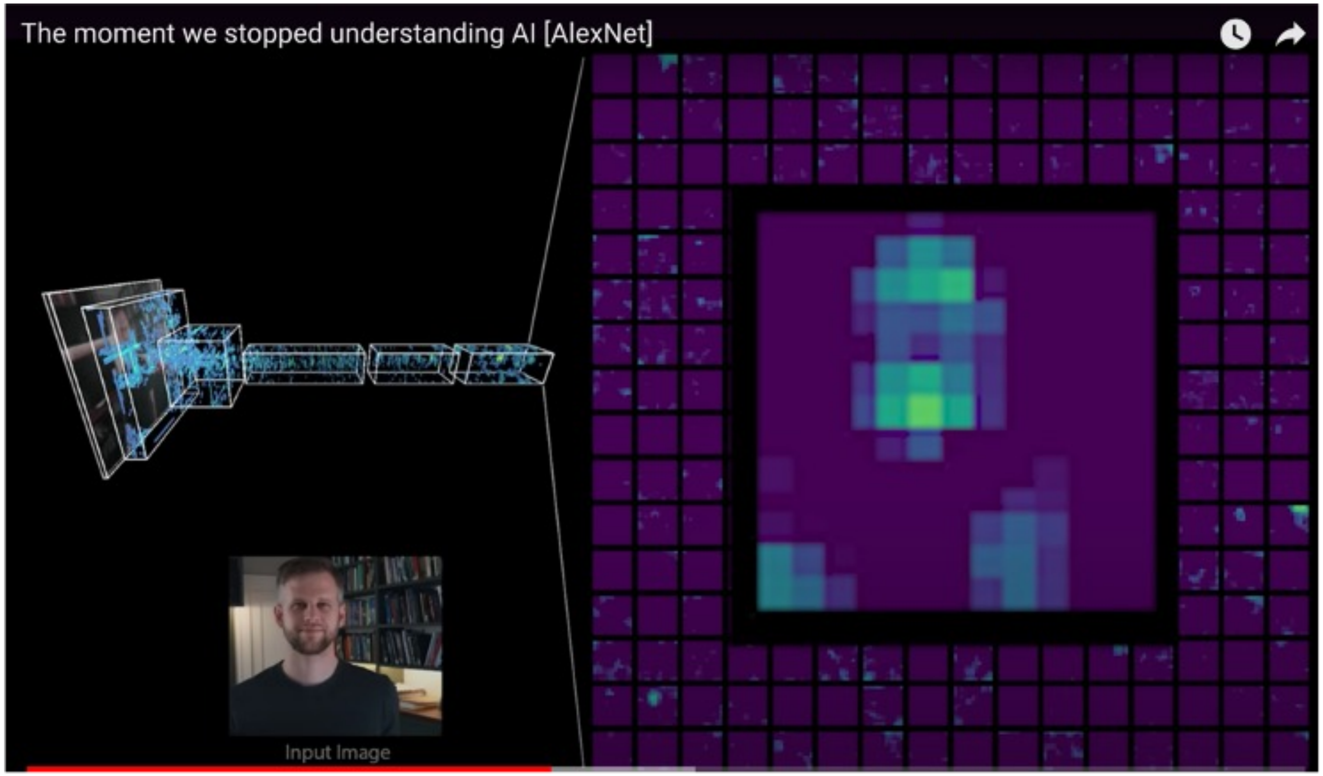
https://www.youtube.com/watch?v=UZDiGooFs54

Testing images shows the deeper layers are learning more complicated features



https://www.youtube.com/watch?v=UZDiGooFs54

Deeper layers recognize very complex features: faces!

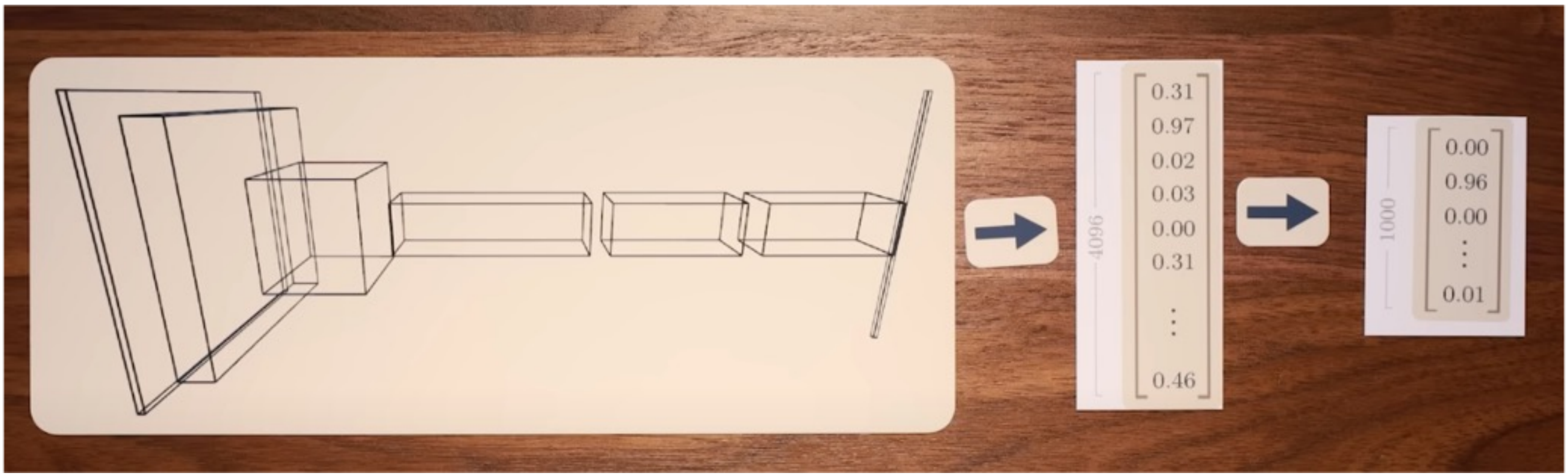


https://www.youtube.com/watch?v=UZDiGooFs54

Final two layers are special:

Last layer [1x1000]: Probability of the image in class i

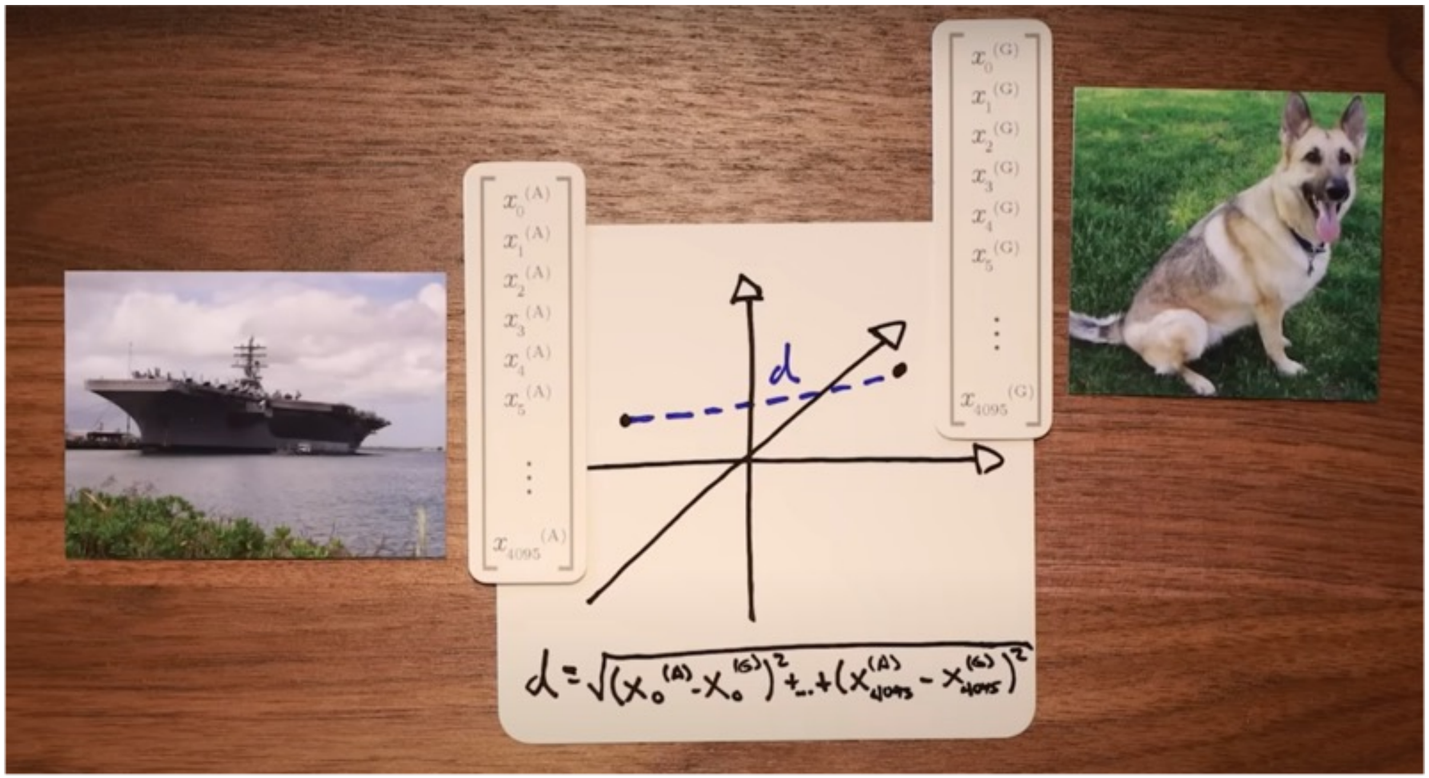
Second last [1x4096]: Projection of image into 4096-dim space



https://www.youtube.com/watch?v=UZDiGooFs54

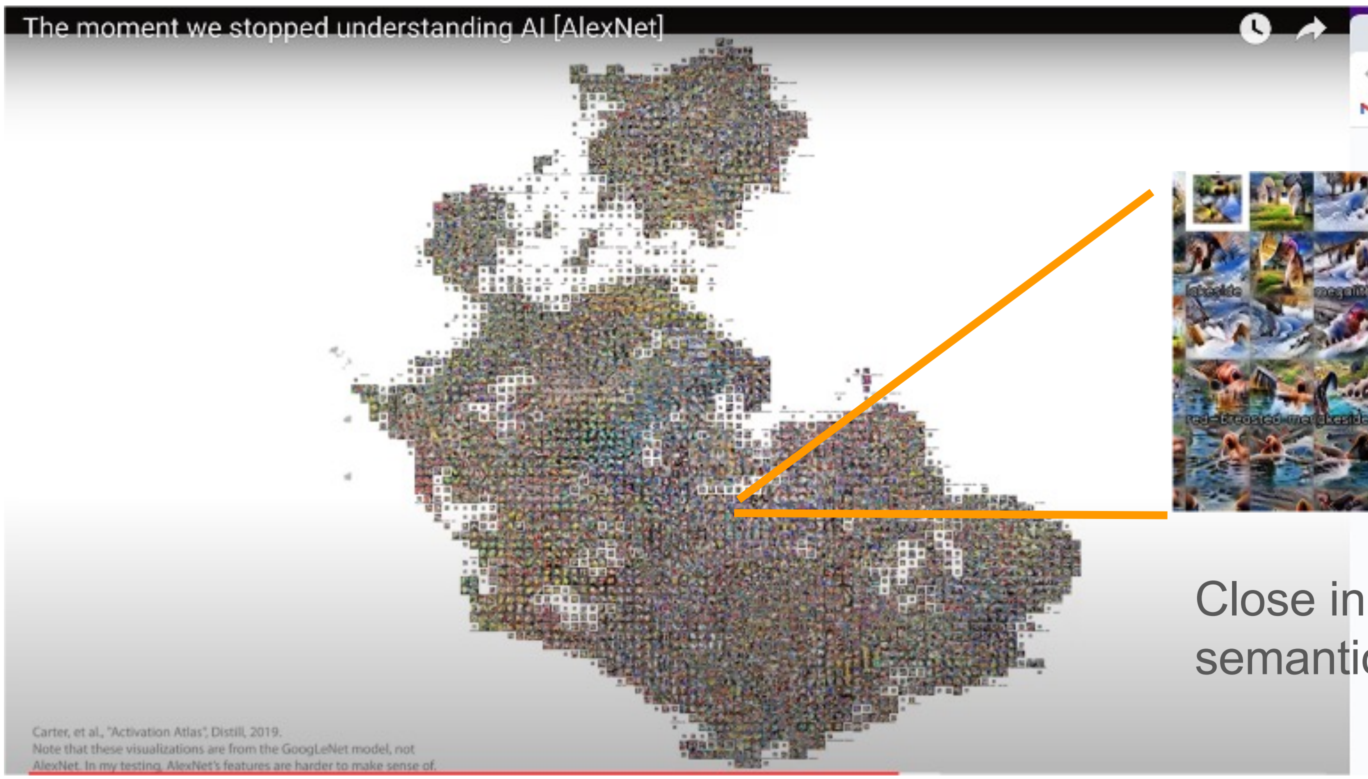
Each image creates a unique vector

Compute the distance between vectors in 4096-dim space



https://www.youtube.com/watch?v=UZDiGooFs54

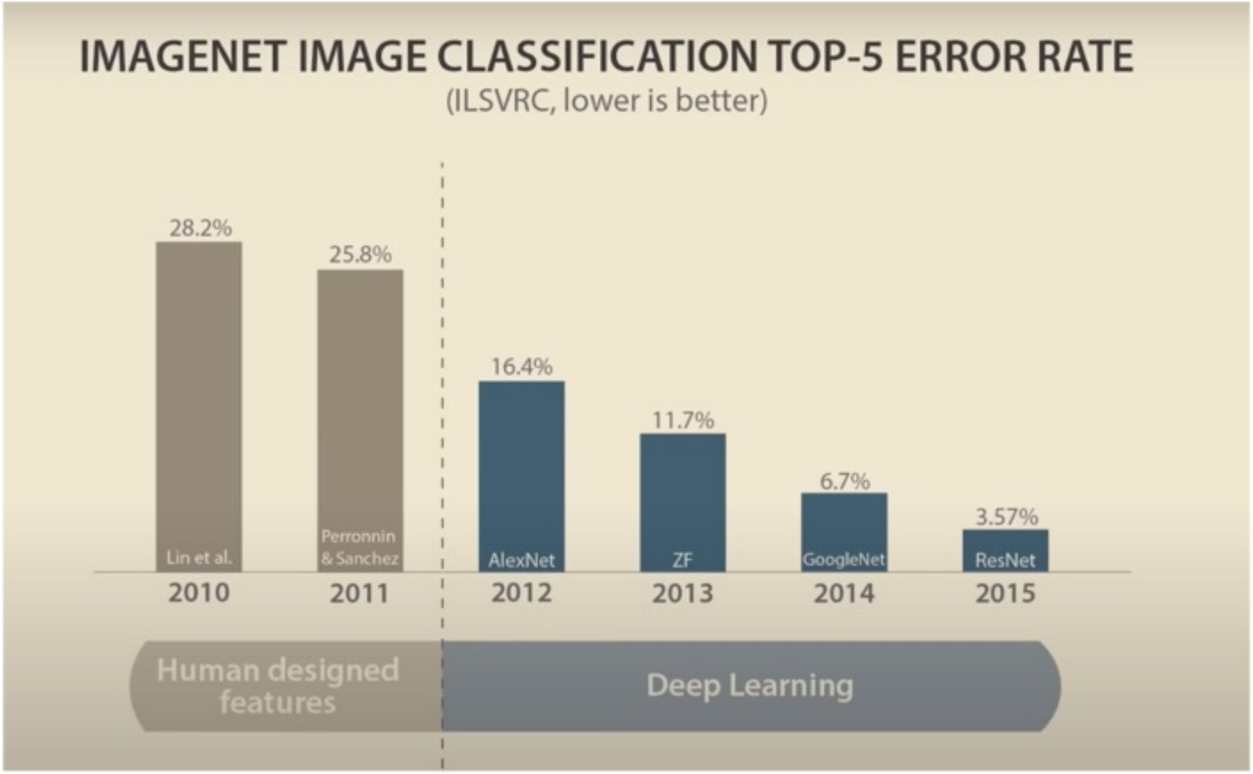
Activation Atlas: tSNE plot of images embeddings



Close in the projection are semantically related

https://www.youtube.com/watch?v=UZDiGooFs54

Deep Learning Success!

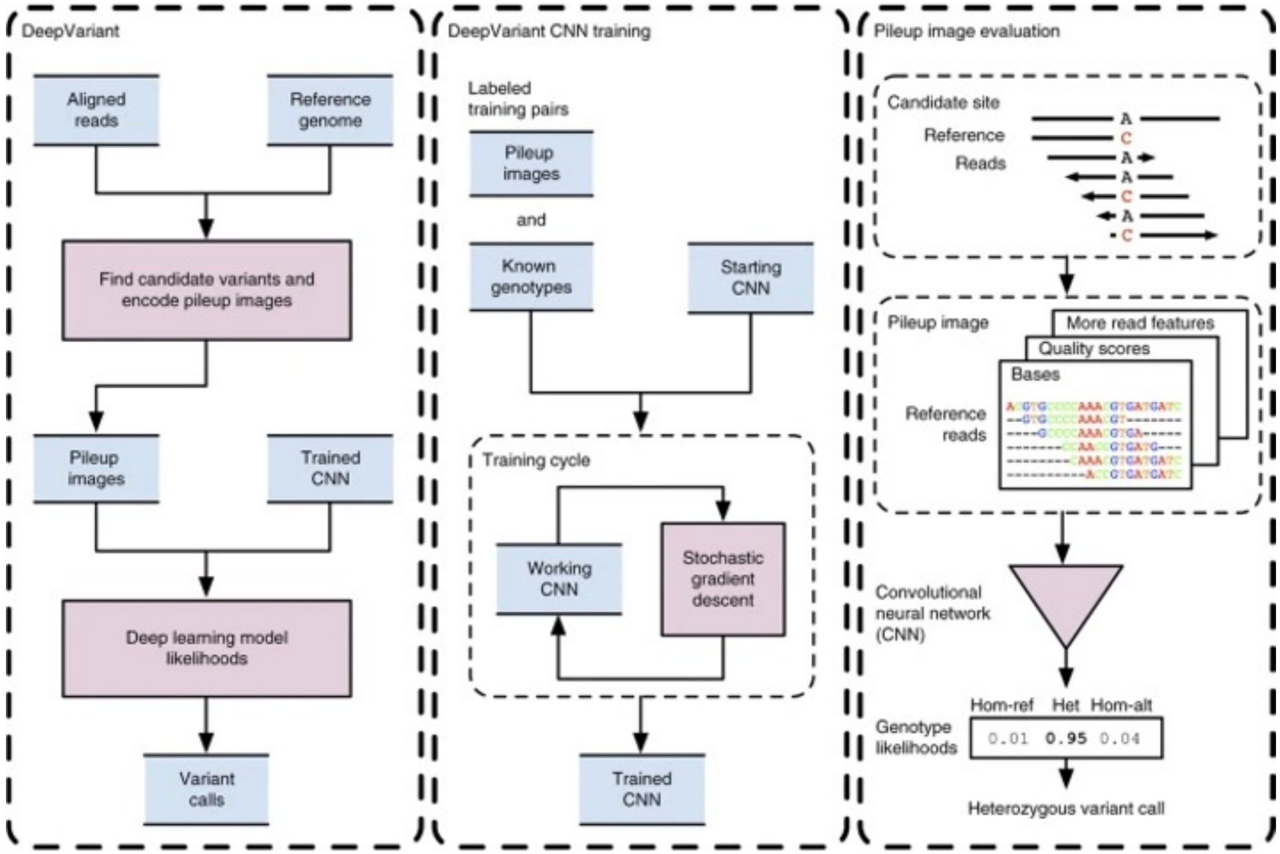


"Our results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. ***So the depth really is important for achieving our results**."*

https://www.youtube.com/watch?v=UZDiGooFs54

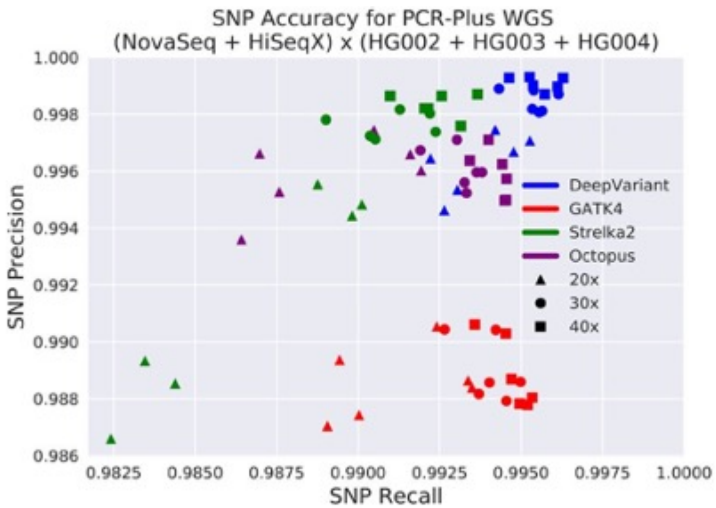
Outline

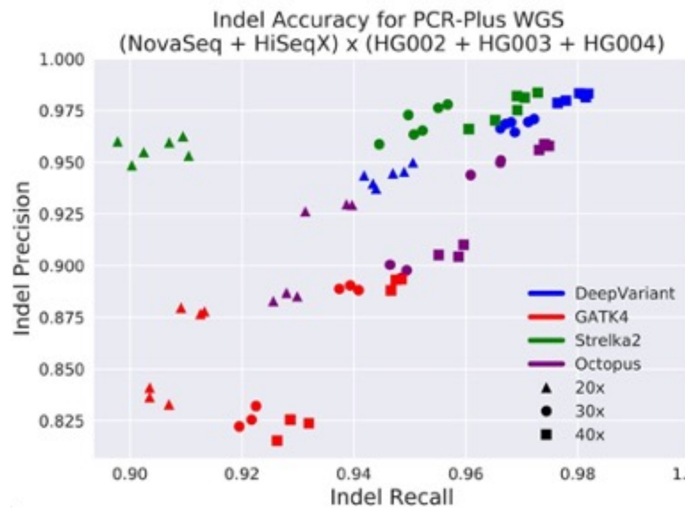
1. The Problem
2. pre-AlexNet approaches
3. How do animals & humans approach this?
4. Artificial Neural Networks
5. Convolutional Neural Networks
6. AlexNet
7. Impact



A universal SNP and small-indel variant caller using deep neural networks

Poplin et al (2018) Nature Biotechnology. https://doi.org/10.1038/nbt.4235



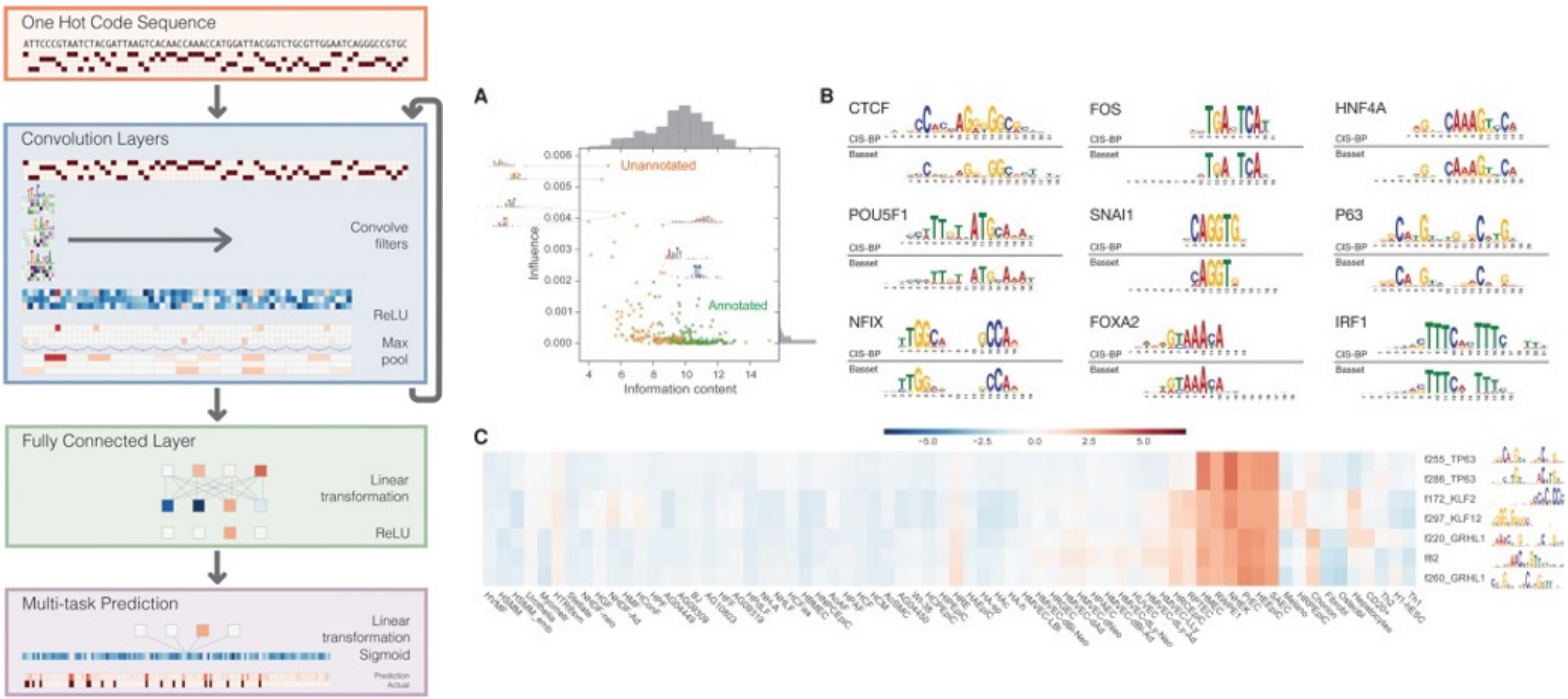


precisionFDA
Challenges

An Extensive Sequence Dataset of Gold-Standard Samples for Benchmarking and Development

Baid et al (2020) bioRxiv

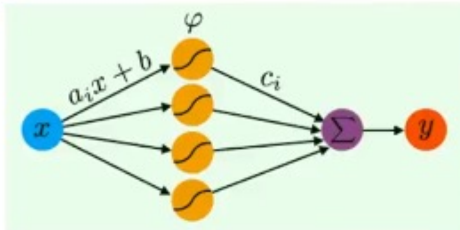
https://www.biorxiv.org/content/10.1101/2020.12.11.422022v1.full



Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks

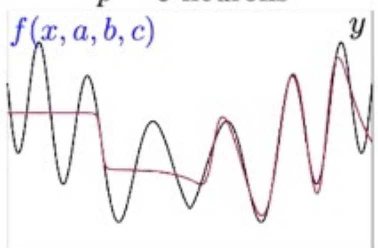
Kelly et al (2016) Genome Research. doi: 10.1101/gr.200535.115

ANNs are “Universal Approximators”

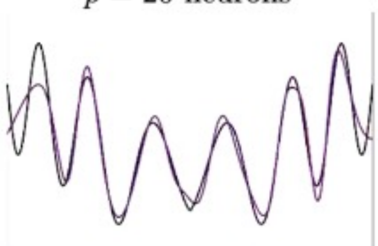


1 hidden layer perceptron:
$$y \approx f(x, a, b, c) \stackrel{\text{def.}}{=} \sum_{i=1}^{p} c_i \varphi(a_i x + b_i)$$

$p = 6$ neurons



$p = 20$ neurons



Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of *n* real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

ANNs are “Universal Approximators”

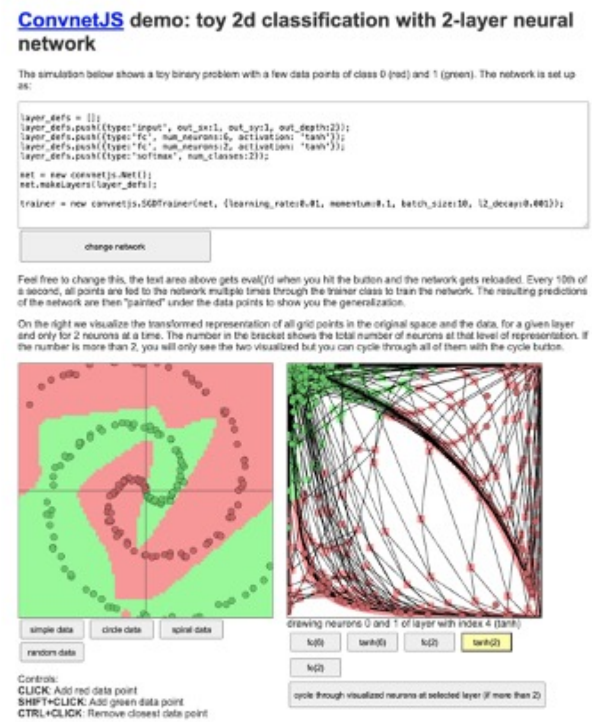
f() => 42

224*224 = 50k pixels

50k pixels * 3 bytes (RGB)
838B possible inputs

ANNs are “Universal Approximators”

f() => 42



224*224 = 50k pixels

50k pixels * 3 bytes (RGB)
838B possible inputs

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

(a) (b)

Figure 5: Adversarial examples generated for AlexNet [9].(Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Average distortion based on 64 examples is 0.006508. Plase refer to `http://goo.gl/huaGPb` for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.

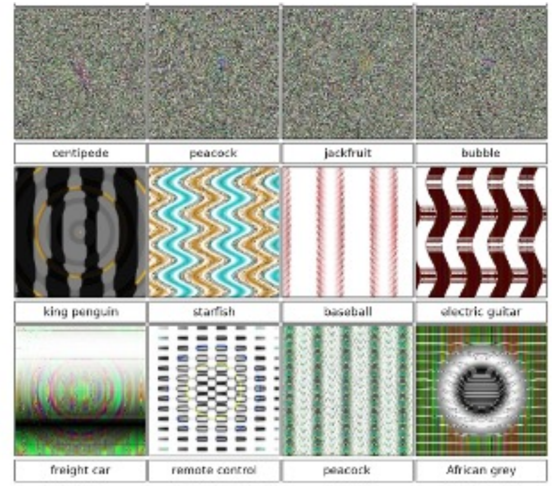


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly (*bottom*) encoded.

Intriguing properties of neural networks
Szegedy et al (2013) arXiv:1312.6199

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
Nguyen et al (2015) arXiv:1412.1897

Reflections

- **CNNs are incredibly powerful for image recognition (and similar tasks)**
    - Heavily inspired by how the human brain processes visual information from the real world
    - Start with the most basic feature kernels and then iterate into higher level concepts leveraging their spatial proximity
    - Ultimately projects images into abstract semantic embeddings

- **Certain problems are a great fit for CNNs:**
    - ImageNet: Lots of test data, nicely annotated into 1000 categories
    - DeepVariant/Clairvoyante: Lots of test data; One network architecture made it the worlds best variant caller for all types of long read data: PacBio, ONT, 10x

- **Deeper network architectures generally perform better**
    - Largely constrained by GPU RAM & Cores

“Images” in Bioinformatics

- **DNA sequence: one-hot encode**
  - Are there other encoding schemes? (modDotPlot)
  - Predict features (genes, repeat classes); Predict species (kraken)
- **Coverage tracks:**
  - Predict: SNVs (DeepVariant), CNVs, SVs (Cue)
  - RNAseq, ChIPseq (overcome error, allele specific)
- **Functional Genomics**
  - Inputs
    - Expression from multiple tissues (GTEx) from one patient
    - Multiomics: Variants + expression + methylation
  - Predict phenotype: Gender (easy), cell types (easy), age (pretty easy)
    - Phenopredict: https://academic.oup.com/nar/article/46/9/e54/4920847
  - Disease outcome: Needs the right training data, right tissues
- Multiple alignments, position weight matrix
  - Predict conserved regions, predict binding sites, etc

	LeNet-5	AlexNet	GPT-3	GPT-4
YEAR	1998	2012	2020	2023
TRAINING DATA	· MNIST Dataset · 60k training examples · 10 classes	· ImageNet Dataset (ILSVRC) · 1.2M training examples · 1000 classes	· Common Crawl, WebText, Wikipedia, others · ~500B training tokens · ~100k unique tokens	· ~13T training tokens*
TRAINING COMPUTE	· Pentium II CPU · ~0.27 GFLOPs	· Dual Nvidia GTX 580 · 3162 GFLOPs	· 10,000 Nvidia V100 GPUs · 1+ ExaFLOPs	· 25,000 Nvidia A100s GPUs* · ~4+ ExaFLOPs
ALGORITHM	· ~60k Parameters · 5 Layers · Sigmoid Activation Function	· ~60M Parameters · 8 Layers · ReLU Activation Function · Dropout	· 175B Parameters · 96 Layers · Transformers	· 1T+ Parameters* · *120 Layers · Transformers

https://www.youtube.com/watch?v=UZDiGooFs54