CPSC 340:
Machine Learning and Data Mining

Deep Learning

Last Time: Neural Networks

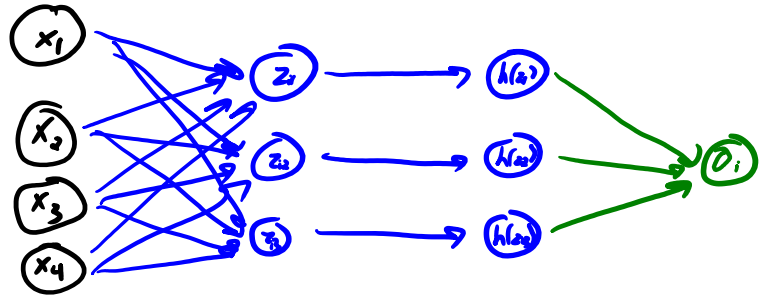
- Neural networks with one hidden layer:
  - Learn features and a classifier at the same time.
  - Output is two linear transformations (W, v), separated by non-linearity (h):

$$o_i = v^T h(W x_i)$$

Linear combination of "activations"

Non-linear transformation of each z_{ic}

"z_i": linear combination of input



  - Linear classification/regression using non-linearly transformed latent features z_i.
  - Optimize logistic/softmax loss (classification) or squared error loss (regression) using SGD:

$$\frac{1}{2}\sum_{i=1}^{n}(\underbrace{v^T h(Wx_i)}_{o_i} - y_i)^2$$

(regression)

$$\sum_{i=1}^{n}\log(1+\exp(-y_i\underbrace{v^T h(Wx_i)}_{o_i}))$$

(binary classification)

Is Training Neural Networks Scary?

- Learning:
  - For binary classification, the NLL under the sigmoid likelihood is:

$$f(W, v) = \sum_{i=1}^{n} \log(1 + \exp(-y_i v^T h(W x_i)))$$

f_i = loss function on example i

    - With 'W' fixed this is convex, but with both 'W' and 'v' as variables it is non-convex.
    - And finding the global optimum is NP-hard in general.
  - Nearly-always trained with variations on stochastic gradient descent (SGD).

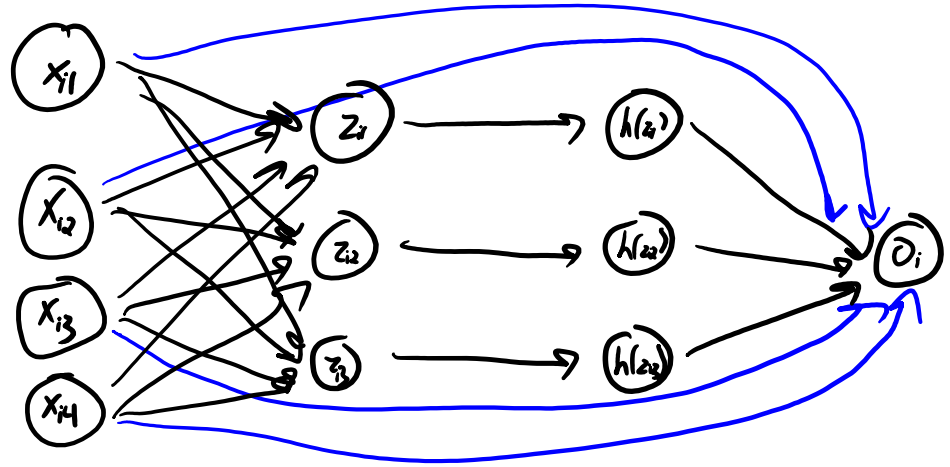
$$W^{k+1} = W^k - \alpha^k \nabla_W f_{i_k}(W^k, v^k)$$
$$v^{k+1} = v^k - \alpha^k \nabla_v f_{i_k}(W^k, v^k)$$

i_k is a training example chosen uniformly at random

    - Many variations exist (with "momentum", AdaGrad, **Adam (173k cites)**, AdamW, and so on).
    - But SGD is not guaranteed to reach a global minimum for non-convex problems.
- Is non-convexity a big drawback compared to logistic regression?
  - And if 'k' is large, is this likely to overfit?

Neural Networks ≥ Logistic Regression

- Consider a neural network with one hidden layer and connections from input to output layer.
  - The extra connections are called "skip" connections.



$$o_i = w^T x_i + v^T h(W x_i)$$

linear model

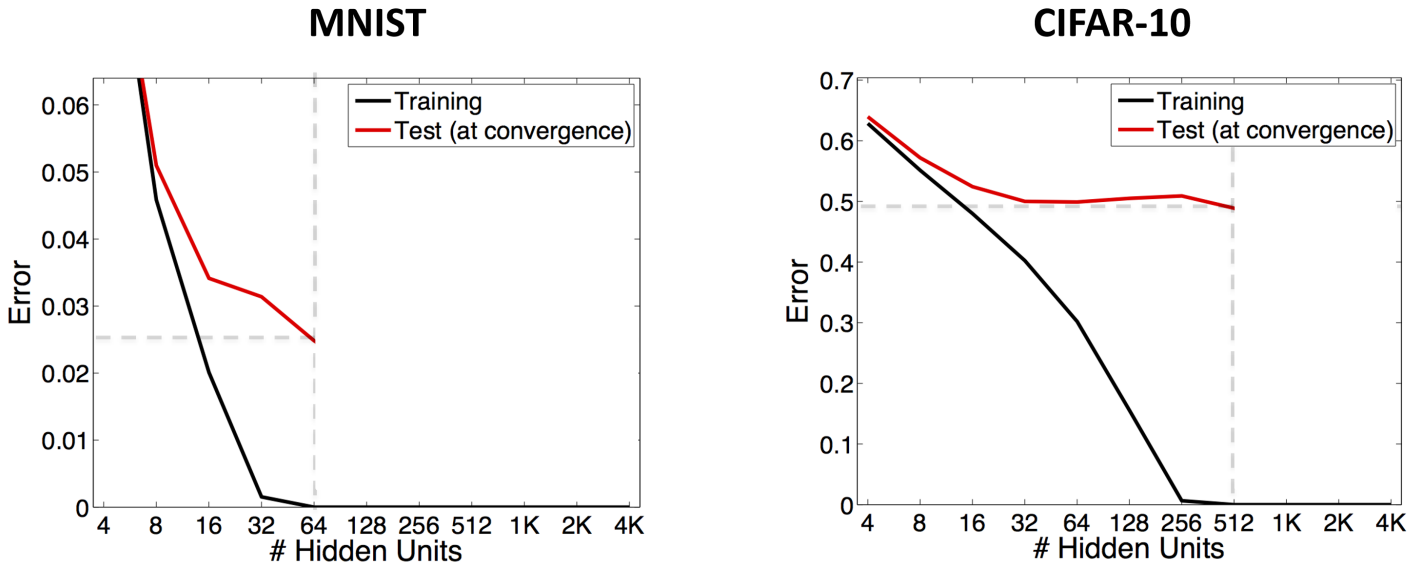
neural network

- You could first set v=0, then optimize 'w' using logistic regression.
  - This is a convex optimization problem that gives you the logistic regression model.
- You could then set 'W' and 'v' to small random values, and start SGD from the logistic regression model.
  - And if you are worried about overfitting, you could use early stopping based on validation set.
  - Even though this is non-convex, the neural network can only improve on logistic regression.

- In practice, we typically optimize everything at once (which usually works better than the above).

“Hidden” Regularization in Neural Networks

- Fitting neural network with one hidden layer (SGD, no regularization):

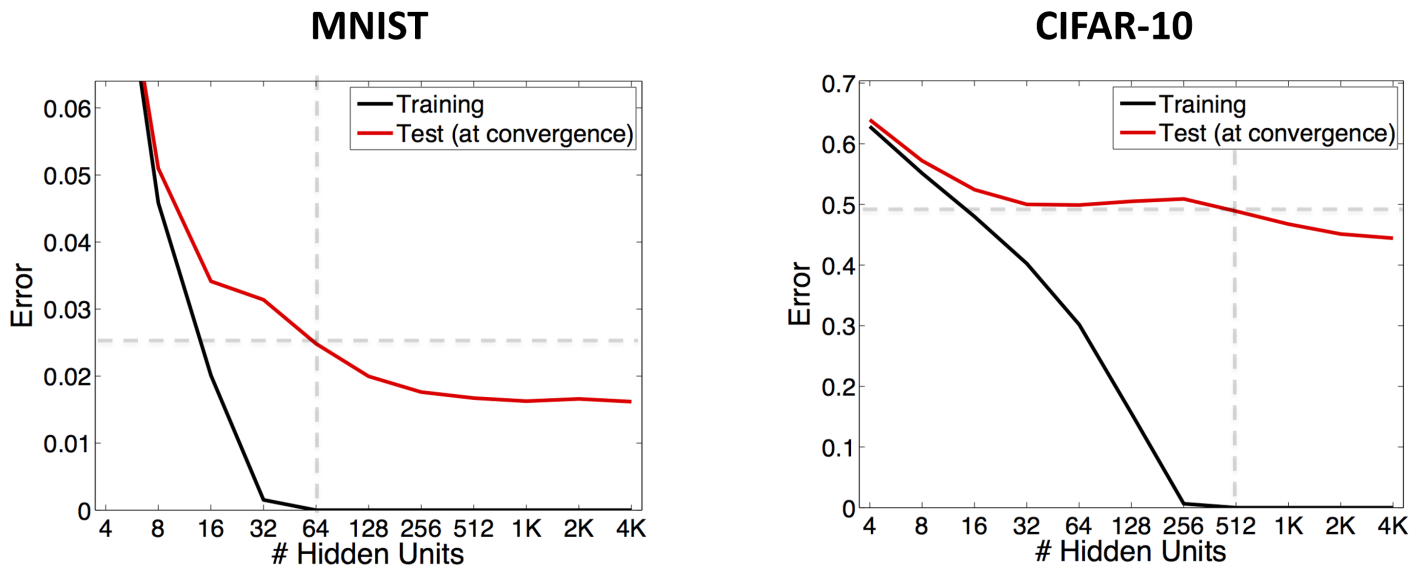


- On each step of the x-axis, the network is re-trained from scratch.
- Training error goes to 0 with enough units: we’re finding a global min.
- What should happen to test error as we increase size of hidden layer?

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

“Hidden” Regularization in Neural Networks

- Fitting neural network with one hidden layer (SGD, no regularization):

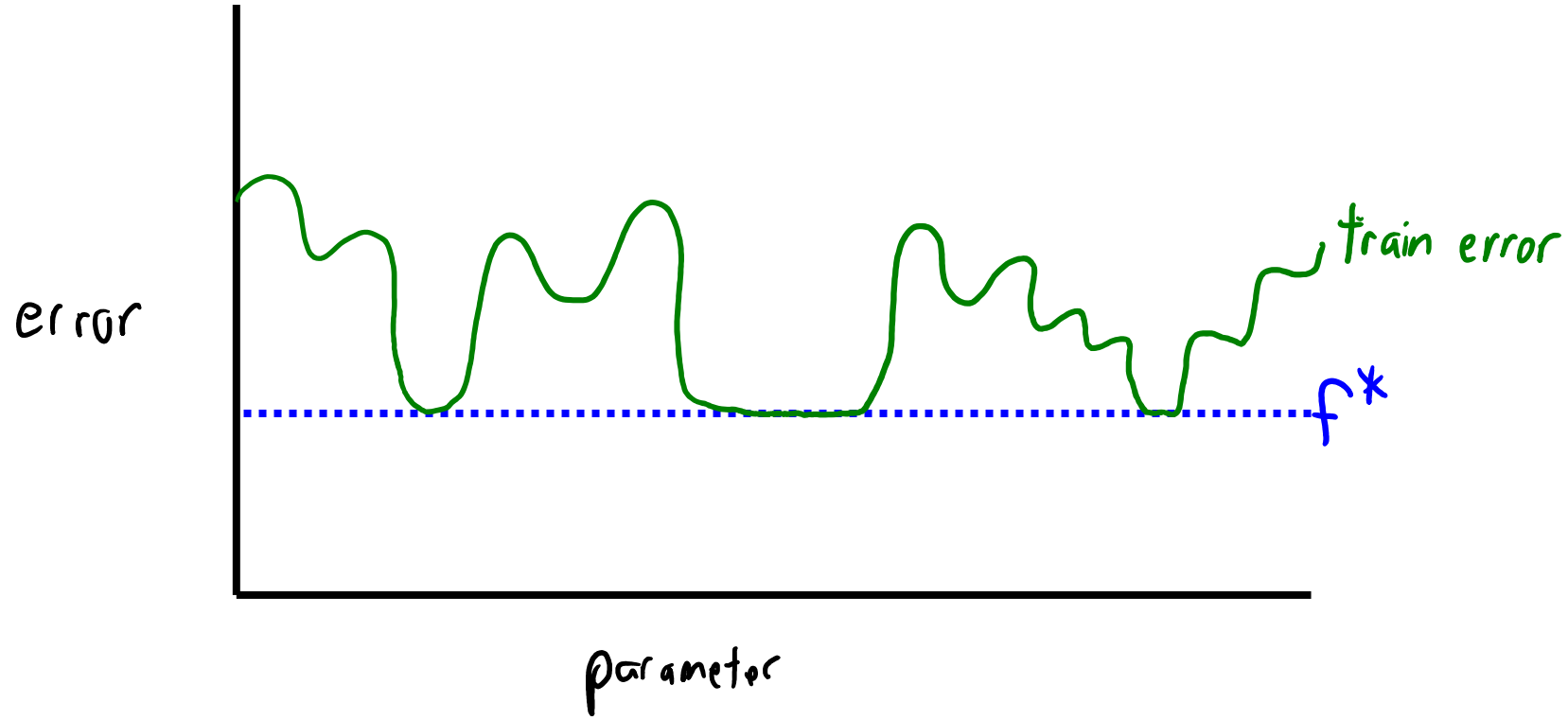


- Test error continues to go down!?! Where is fundamental trade-off??
  - It is still fundamental, but trade-off focuses on the “worst” global minimum.
- There do exist global mins with large #hidden units have test error = 1.
  - But among the global minima, SGD is somehow converging to “good” ones.

https://www.neyshabur.net/papers/inductive_bias_poster.pdf

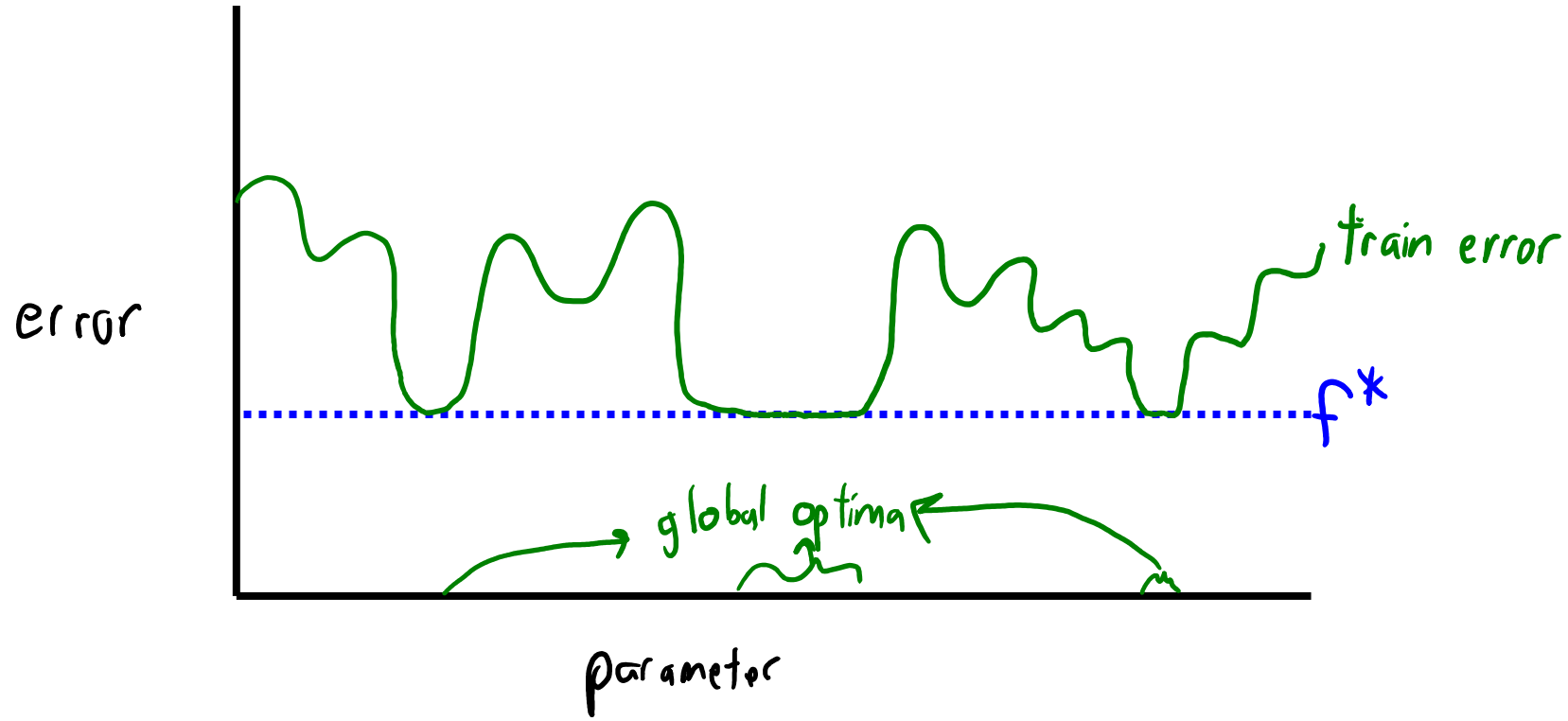
Multiple Global Minima?

- For standard objectives, there is a global min function value f*:



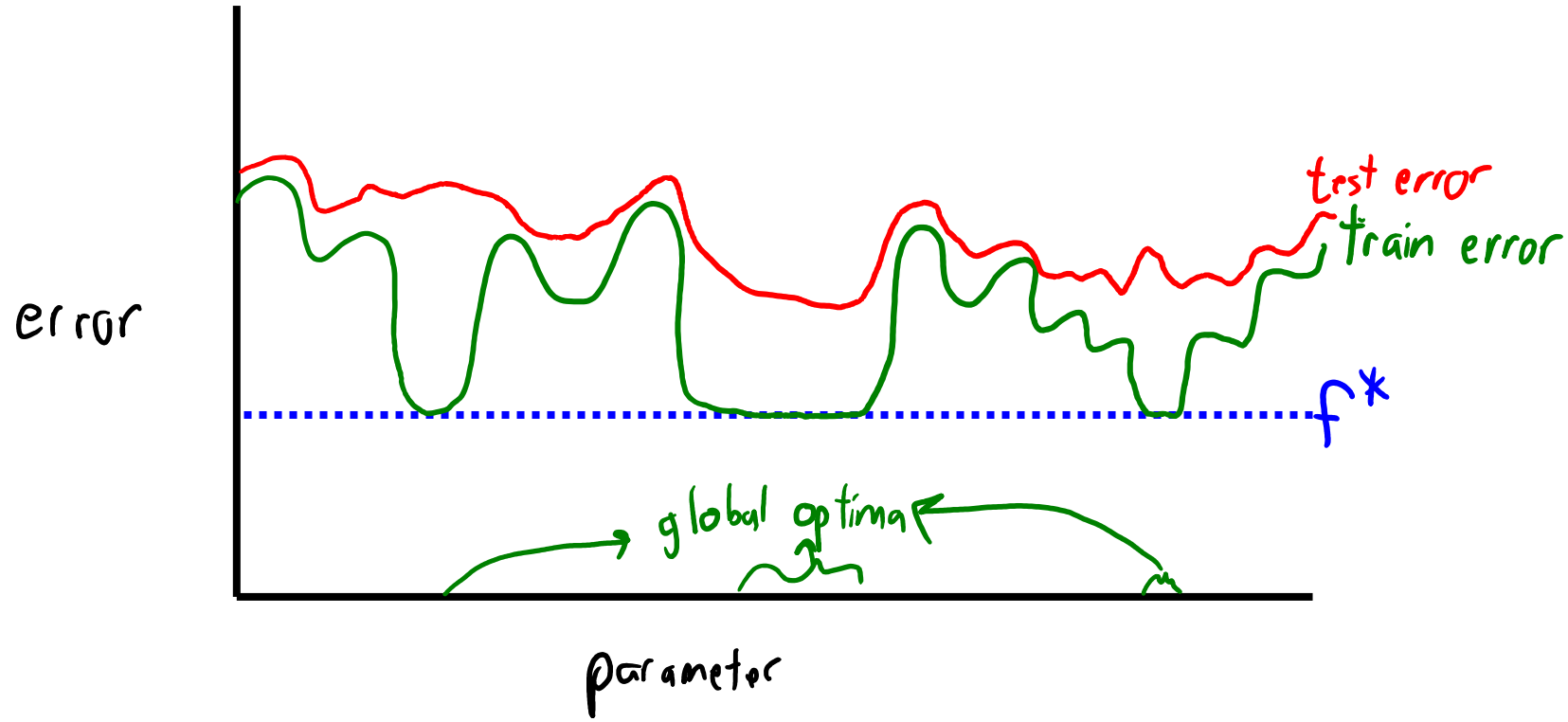
Multiple Global Minima?

- For standard objectives, there is a global min function value f*:



- But this may be achieved by many different parameter values.

Multiple Global Minima?



- These training error "global minima" may have very-different test errors.
- Some of these global minima may be more "regularized" than others.

Implicit Regularization of SGD

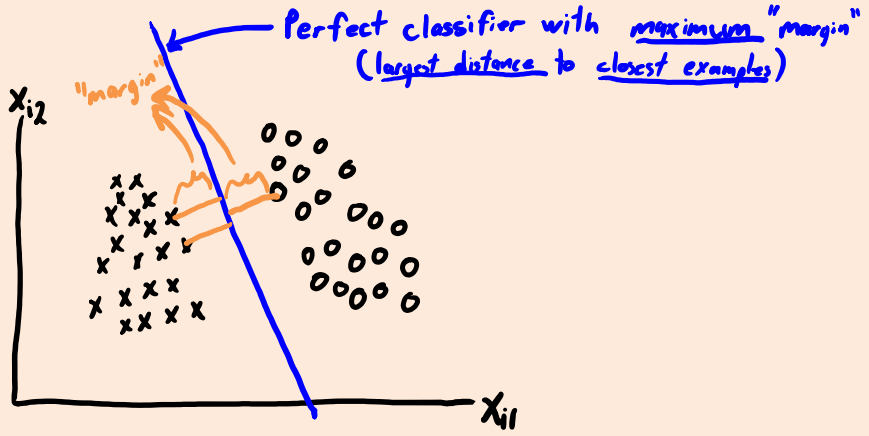
- There is empirical evidence SGD finds regularized parameters.
  - We call this the "implicit regularization" of the optimization algorithm, a new concept/phenomenon observed in the deep learning era.

- Beyond empirical evidence, we know this happens in simpler cases.

- An example of provable implicit regularization:
  - Consider a least squares problem where there exists a 'w' where Xw=y.
    - Residuals are all zero and we fit the data exactly for some 'w'.
  - You run gradient descent or SGD starting from w=0.
  - Converges to solution Xw=y that has the minimum L2-norm.
    - So using SGD is like using L2-regularization, but regularization is "implicit".

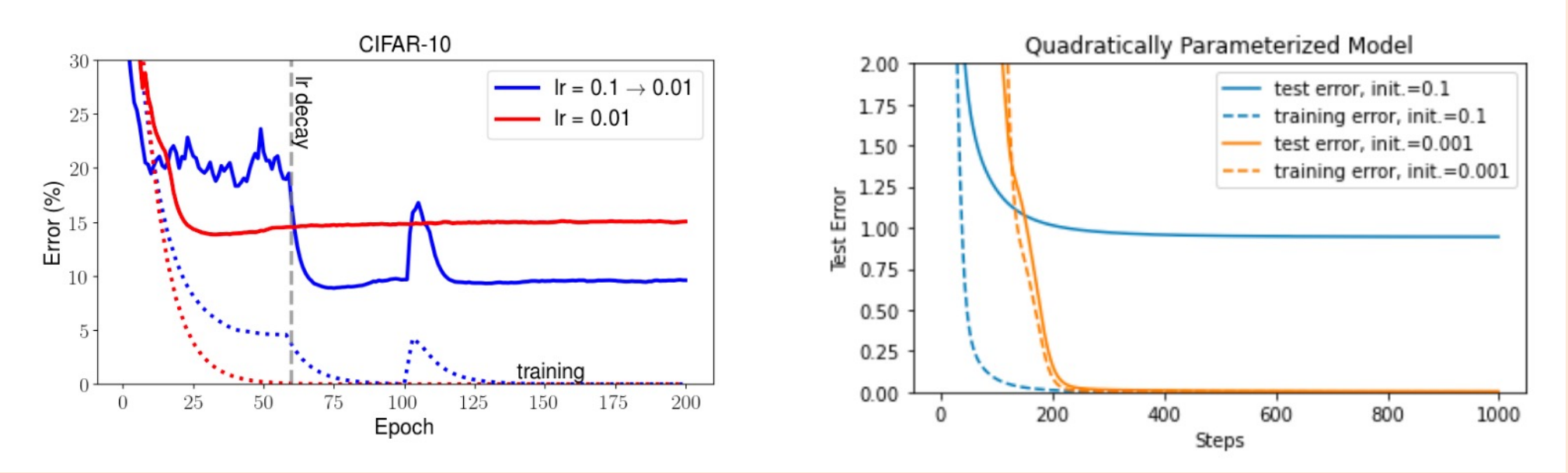
Implicit Regularization of SGD

- Another example of provable implicit regularization:
  - Consider a logistic regression problem where data is linearly separable.
    - A linear model can perfectly separate the data.
  - You run gradient descent from any starting point.
  - Converges to max-margin solution of the problem (minimum L2-norm solution).
    - So using gradient descent is equivalent to encouraging large margin.



- Related implicit regularization results are known for boosting, matrix factorization, and linear neural networks.

Implicit Regularization of SGD Examples



- Different global minima have vastly different test errors

https://cs229.stanford.edu/notes2022fall/main_notes.pdf

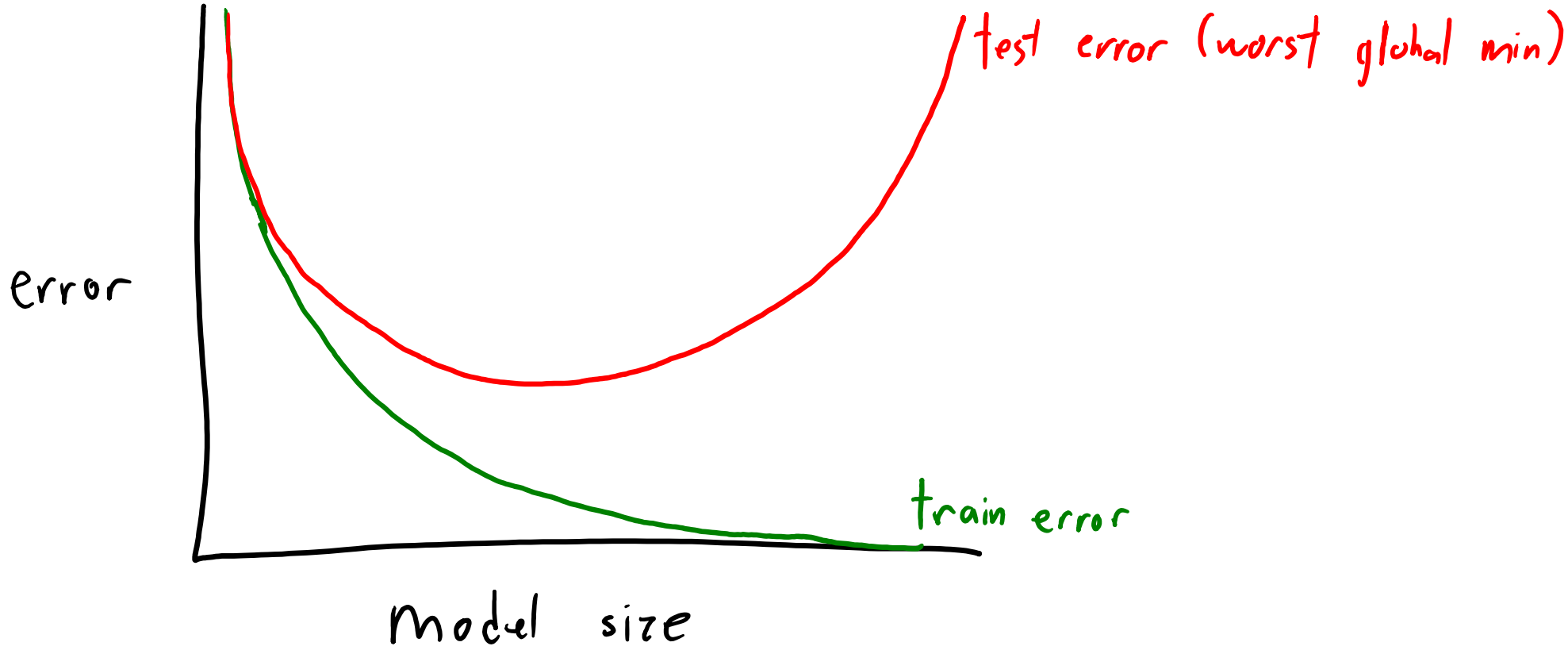
Next Topic: Double Descent Phenomenon

Double Descent Curves

- What is going on???

https://openai.com/blog/deep-double-descent/

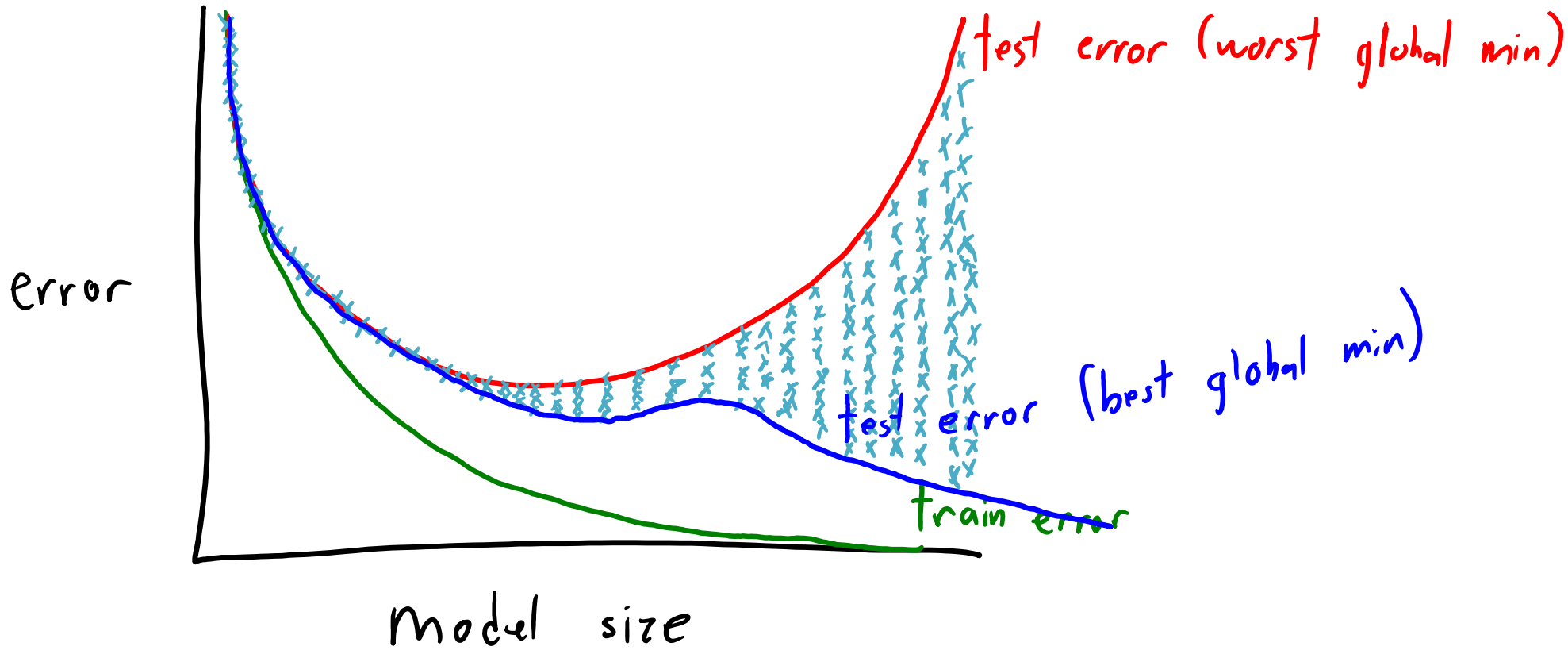
Worst vs. Best “Global Minimum”



Worst vs. Best "Global Minimum"

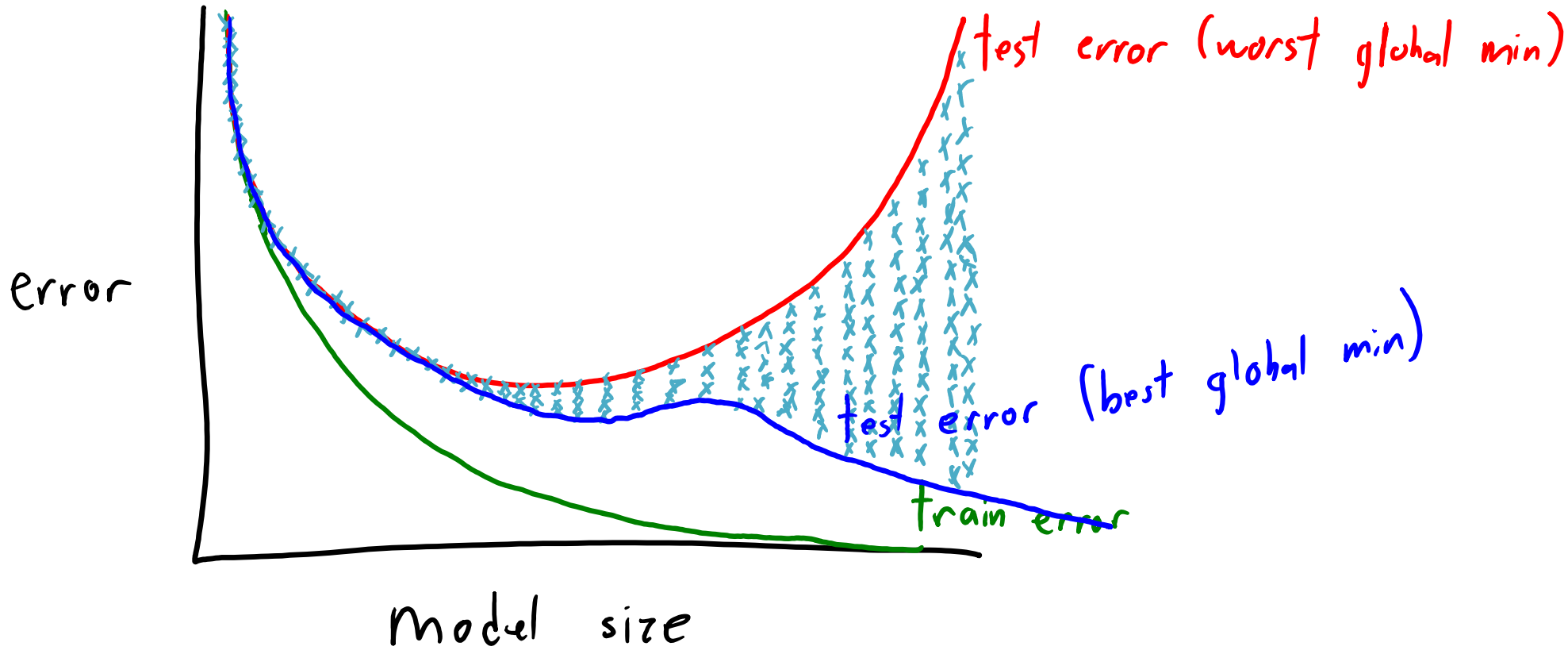
- Learning theory (trade-off) results analyze global min with worst test error.
  - Actual test error for different global minima will be better than worst case bound.
  - Theory is correct, but maybe "worst overfitting possible" is too pessimistic?

Worst vs. Best "Global Minimum"



- Consider instead the global min with best test error.
  - With small models, "minimize training error" leads to unique (or similar) global mins.
  - With larger models, there is a lot of flexibility in the space of global mins (gap between best/worst).
- Gap between "worst" and "best" global min can grow with model complexity.

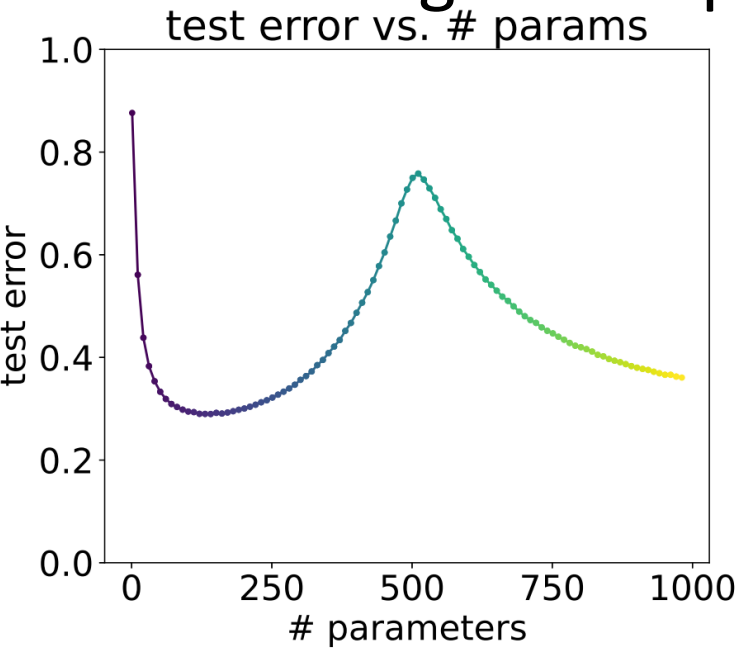
Worst vs. Best "Global Minimum"



- Can get "double descent" curve in practice if parameters roughly track "best" global min shape.
  - One way to do this: increase regularization as you increase model size.
- Maybe "neural network trained with SGD" has "more implicit regularization for bigger models"?
  - But "double descent" is not specific to implicit regularization of SGD and not specific to neural networks.

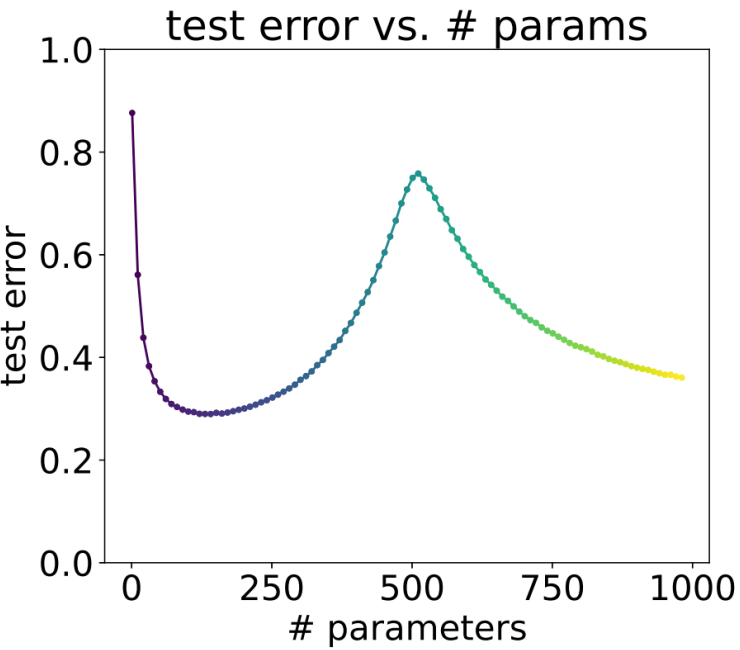
Double Descent on a Linear Least Squares Problem

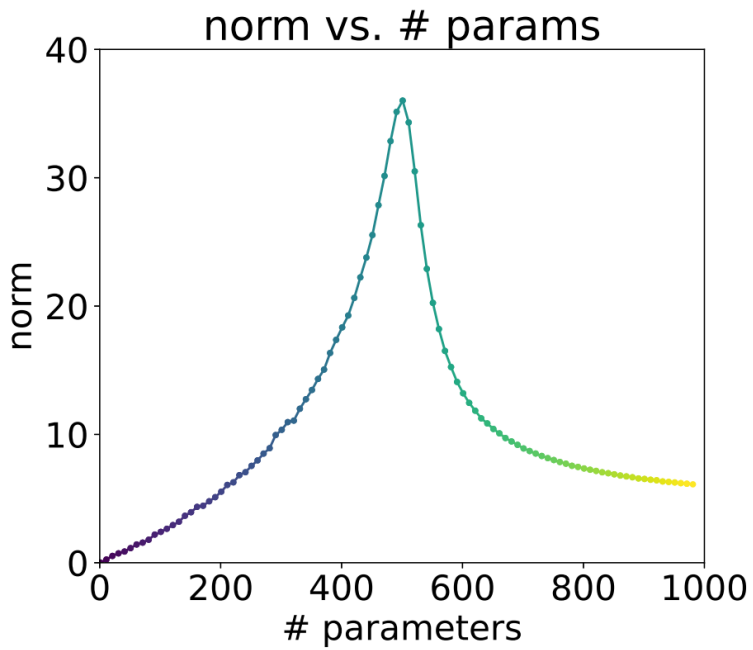
- Fitting least squares with gradient descent (n=500):



https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf

Double Descent on a Linear Least Squares Problem

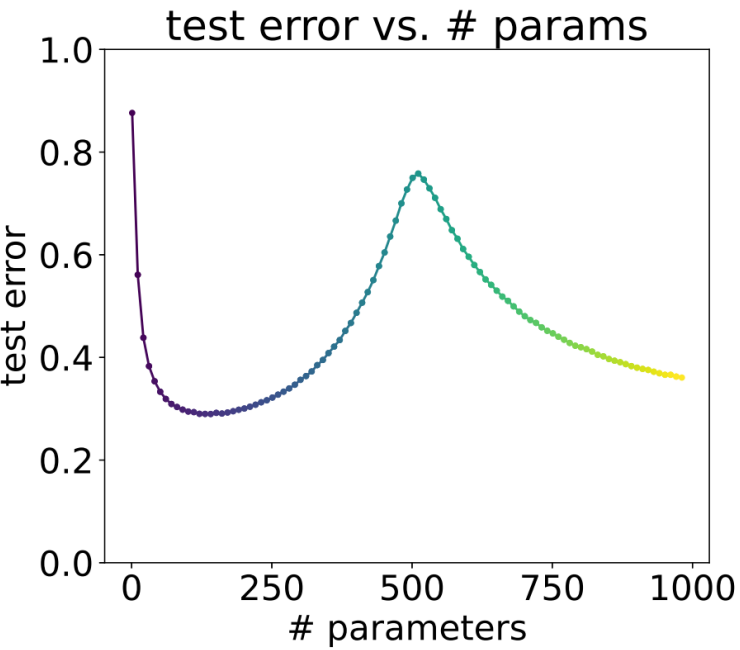


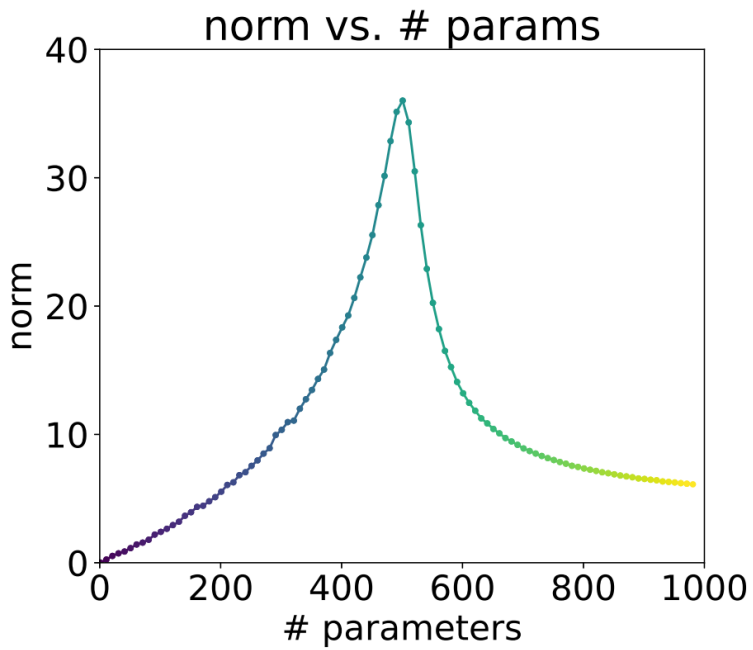


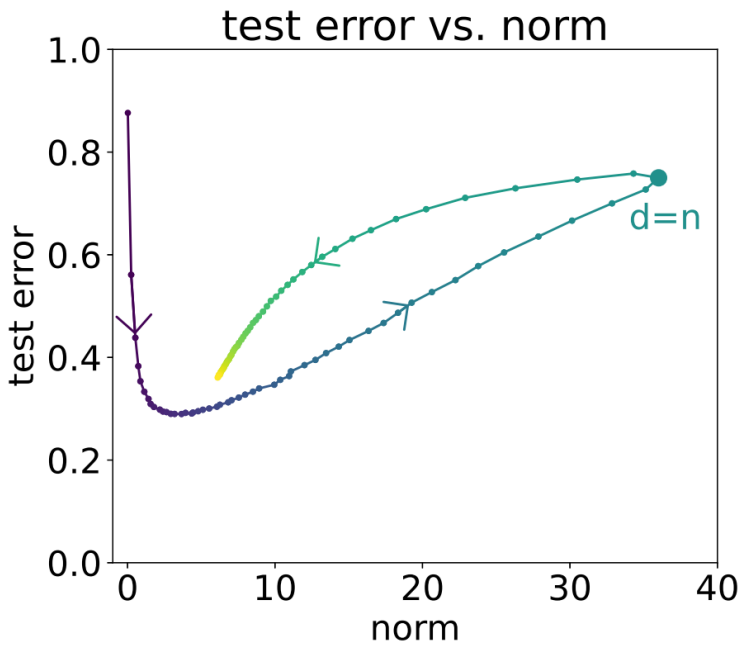
- ||w|| increases until you fit data exactly (only one ‘w’ fits exactly).
- Then norm of parameters starts decreasing (many ‘w’ can fit exactly).
  - So implicit regularization of gradient descent gives lower norm ‘w’ values.

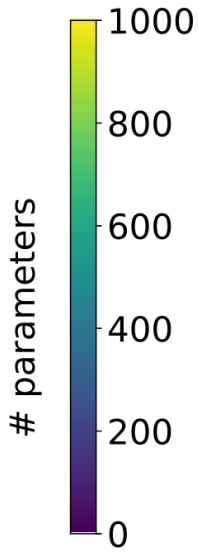
https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf

Double Descent on a Linear Least Squares Problem







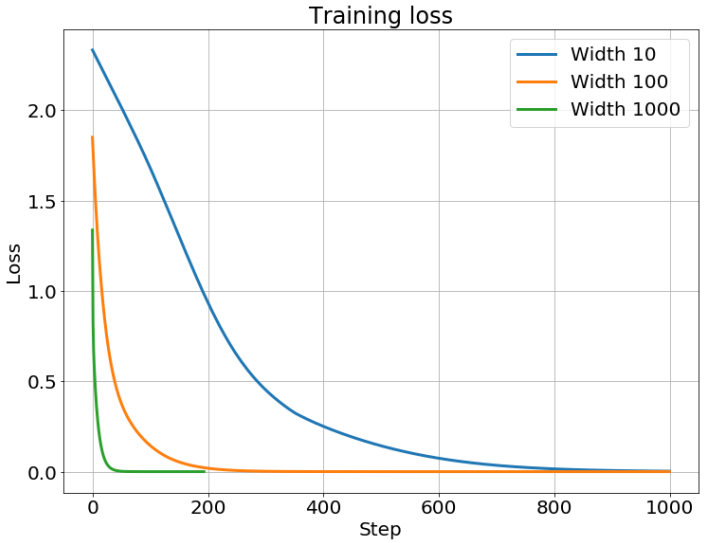


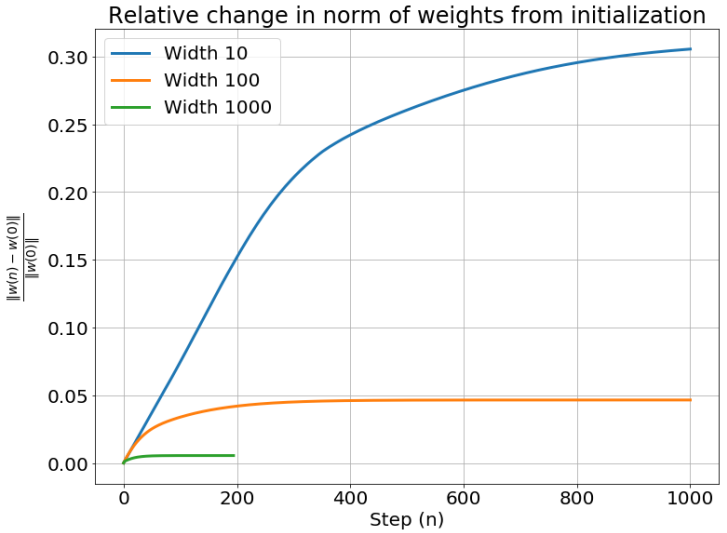
- We see fundamental trade-off if we plot error vs. norm.
  - After we have fit data exactly, models are less "complicated" as we add more parameters.
- Can also make double descent curves by increasing explicit regularization.
- Under right conditions, can see double descent in other models like random forests.

https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf

Implicit Regularization of SGD for Neural Networks

- For neural networks, why would SGD implicit regularization increase with number of hidden units?
  - Similar to least squares, maybe SGD finds low-norm solutions?
    - In higher-dimensions, there is flexibility in global mins to have a low norm?
  - Maybe SGD stays closer to starting point as we increase dimension?
    - This would be more like a regularizer of the form $||w - w^0||$.





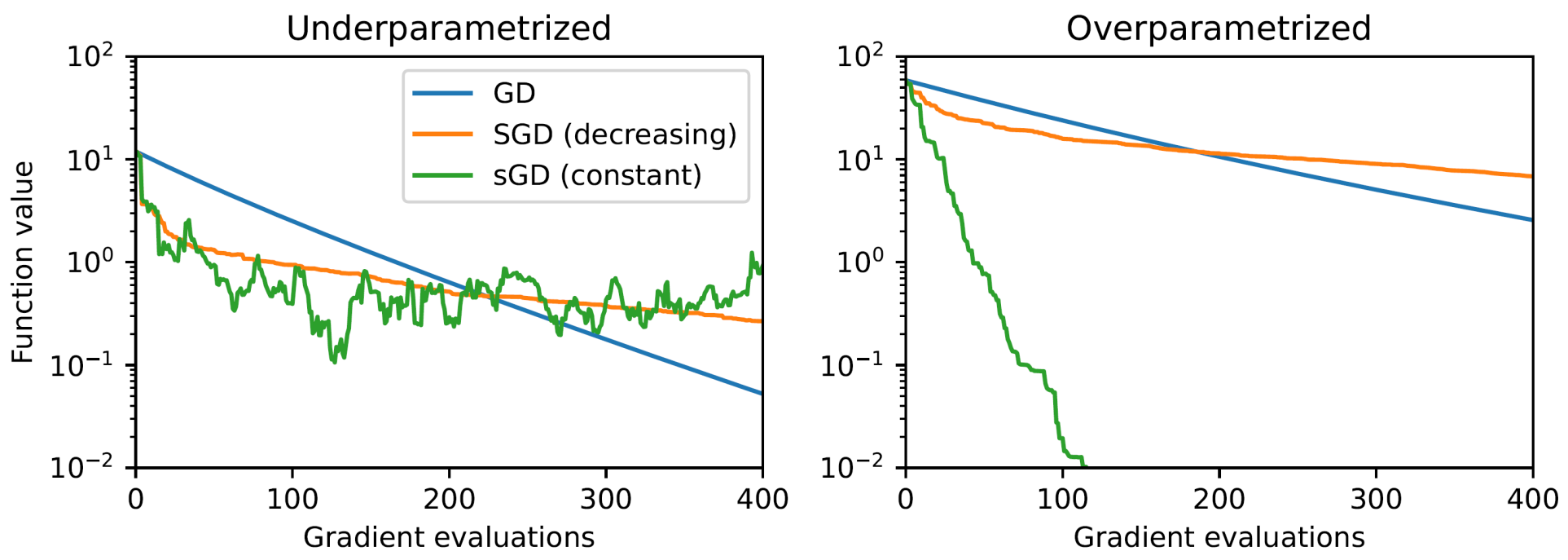
Over-Parameterization and SGD

- Over-parameterized model:
  - A model that has more parameters than needed to fit data exactly.
- Amazing properties of SGD for many over-parameterized models:
  - SGD tends to find a global minimum of training error.
  - SGD tends to have implicit regularization.
  - SGD converges with a constant step size.
    - At nearly the speed of gradient descent.

- Why can SGD converge with a constant step size?
  - Variation in gradients is 0 at solutions that fit all training examples.
    - No “region of confusion”.

Over-Parameterization and SGD

- Gradient descent vs. SGD for under/over-parameterized least squares:

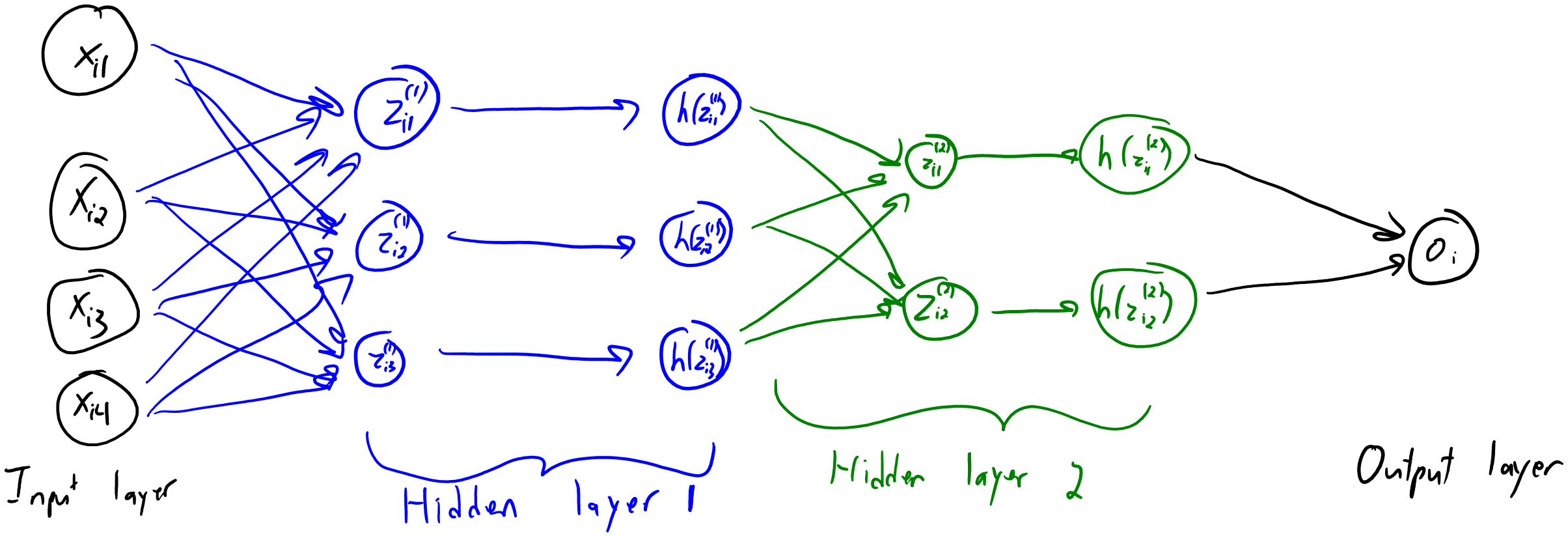


  - No need to decrease step sizes or increase batch sizes for over-parameterized.
    - And nice ways to set the step size as you go ("painless SGD", "Polyak step size").
  - Still expect good performance if you are close to being over-parameterized.

Next Topic: Deep Learning

Deep Learning (As a Picture)

- Deep learning models have more than one hidden layer:



- We apply linear transformation and activation function at each "layer".

Deep Learning (As a Function)

Linear model:

$$o_i = w^T x_i$$

Neural network with 1 hidden layer:

$$o_i = v^T h(W x_i)$$

z_i

Neural network with 2 hidden layers:

$$o_i = v^T h(W^{(2)} h(W^{(1)} x_i))$$

$z_i^{(1)}$, $z_i^{(2)}$

Neural network with 3 hidden layers

$$o_i = v^T h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i)))$$

$z_i^{(1)}$, $z_i^{(2)}$, $z_i^{(3)}$

Neural network with 4 hidden layers:

$$o_i = v^T h(W^{(4)} h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))))$$

With 'm' layers we could use:

$$o_i = v^T \left(\prod_{\ell=1}^{m} h(W^{(\ell)} x_i) \right)$$

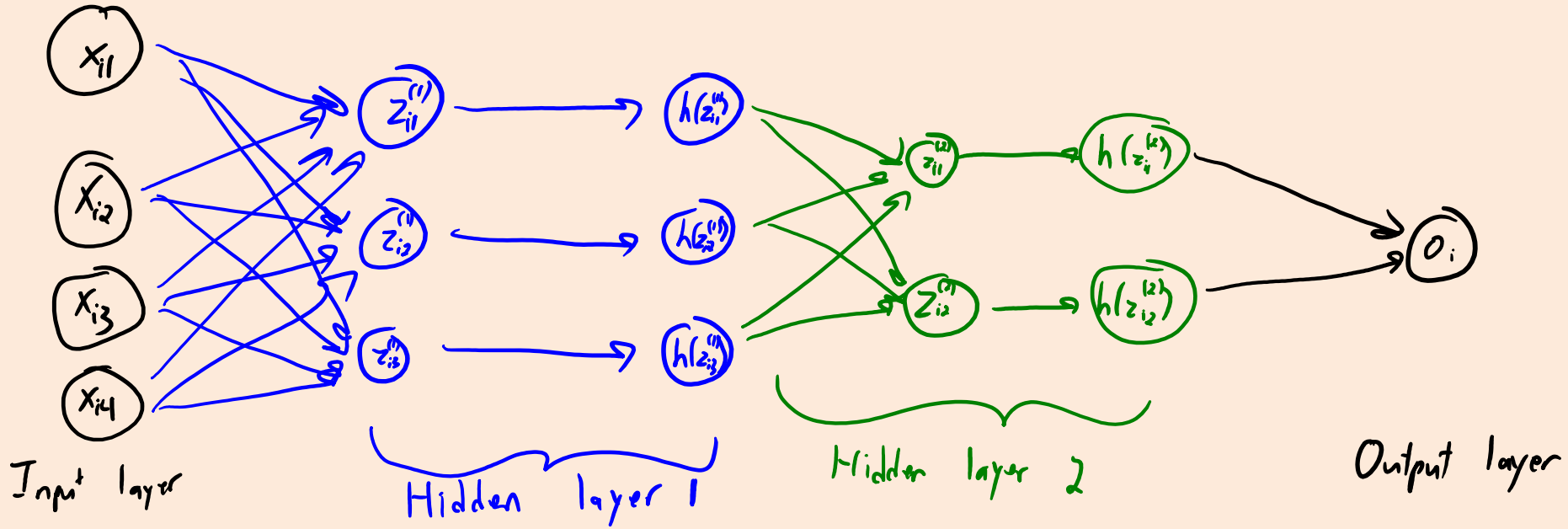
Symbol: $\prod_{k=0}^{n} f_k(t)$

Meaning: $f_n \circ f_{n-1} \circ f_{n-2} \circ \dots \circ f_2 \circ f_1 \circ f_0(t)$

https://mathwithbaddrawings.com/2016/04/27/symbols-that-math-urgently-needs-to-adopt

Notation Warning: "Number of Layers"

- In this class, we say that the network below has "2 hidden layers":
  - Number of intermediate hidden unit groups is number of "layers".



- Caution: exist other ways of counting the number of "layers".
  - Some sources would refer to the above as a 3-layer neural network.
    - They count the number of linear transformations we do.
    - So network with 1 hidden layer would be a "2-layer" network, and linear models are "1-layer networks".

Prediction with Deep Neural Networks

- The "textbook" choice for deep neural networks:
  - Alternate between doing linear transformations and non-linear transforms.

$$o_i = v^T h(W^{(4)} h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))))$$

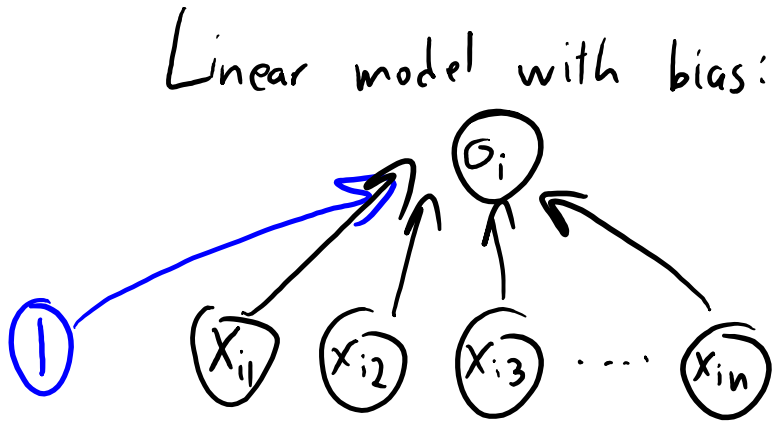
  - Each "layer" might have a different size.
    - W^1 is $k^1 \times d$.
    - W^2 is $k^2 \times k^1$.
    - W^3 is $k^3 \times k^2$.
    - W^4 is $k^4 \times k^3$.
    - v is $k^4 \times 1$.

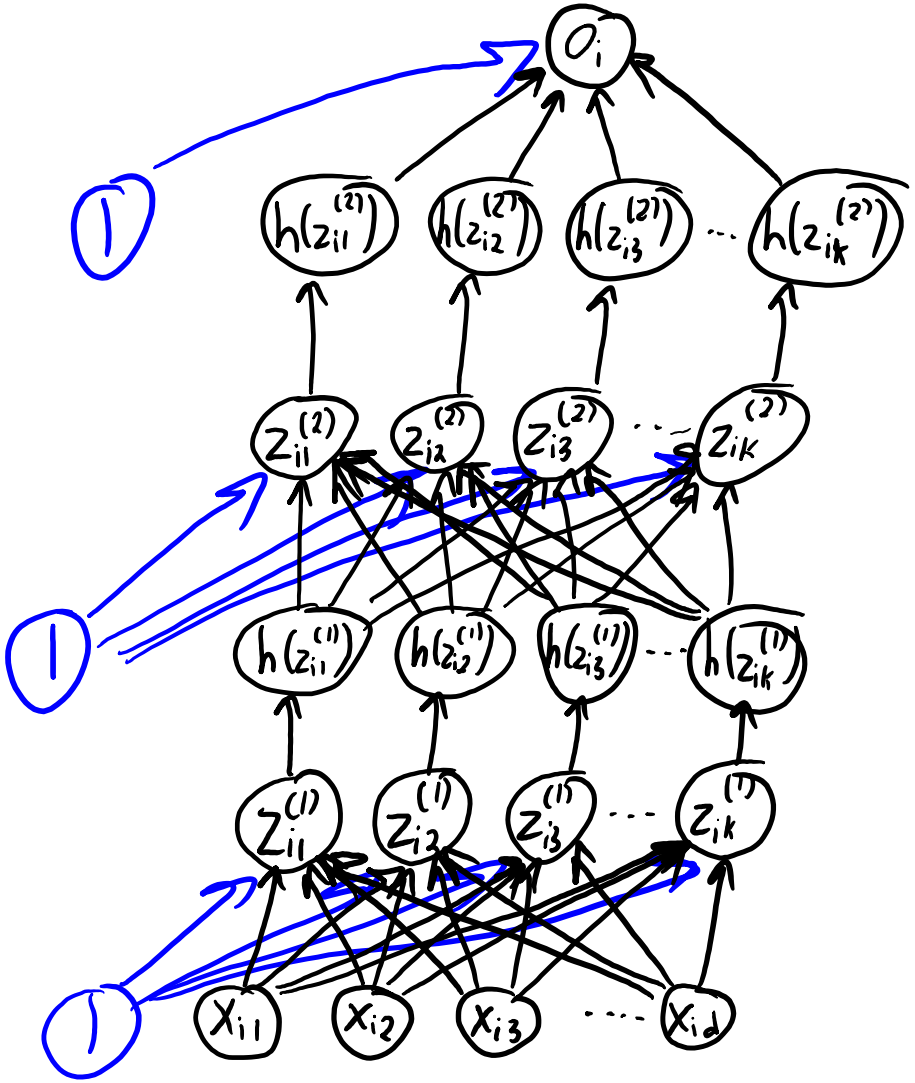
```
z[1] = W1*x
for layer in 2:nLayers
    z[layer] = Wm[layer-1]*h(z[layer-1])
end
yhat = v'*h(z[end])
```

  - We may use the same non-linear transform, such as sigmoid, at each layer.
  - Cost for prediction, which is called "forward propagation":
    - Cost of the matrix multiplies: $O(k^1 d + k^2 k^1 + k^3 k^2 + k^4 k^3)$
    - Cost of the non-linear transforms is $O(k^1 + k^2 + k^3 + k^4)$, so does not change cost.
  - Only need to change last layer based on task (like regression or classification).
    - Squared error, logistic, softmax, and so on.

Adding Bias Variables

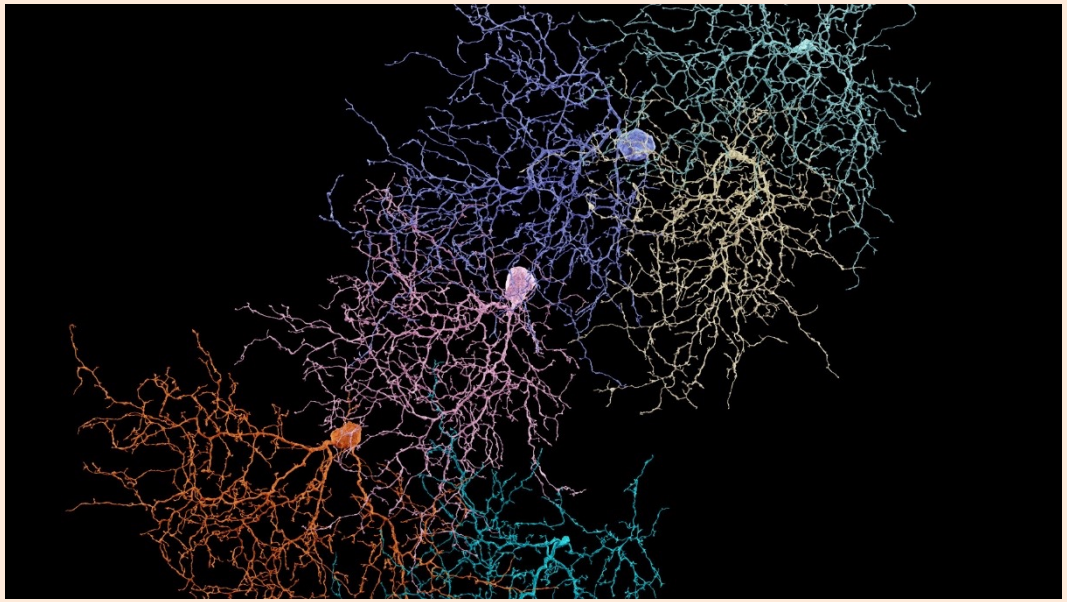
- We typically add a bias to each layer:





Why Multiple Layers?

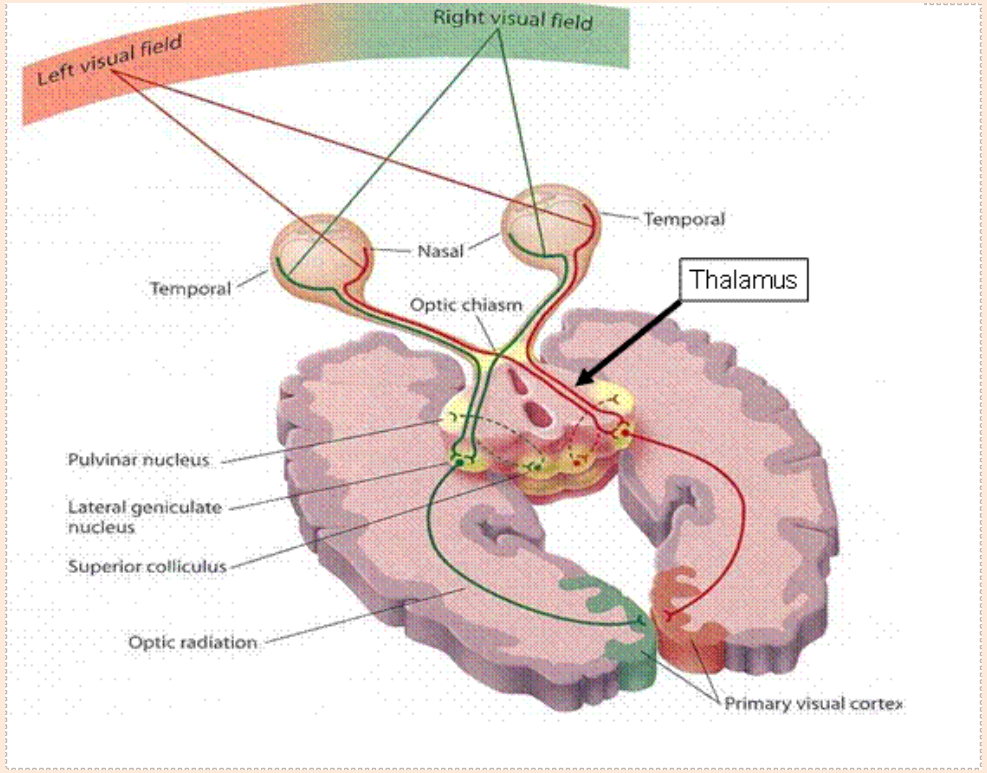
- Historically, deep learning was motivated by "connectionist" ideas:
  - Brain consists of network of highly-connected simple units.
    - Same units repeated in various places.
    - Computations are done in parallel.
    - Information is stored in distributed way.
    - Learning comes from updating of connection strengths.
    - One learning algorithm used everywhere.

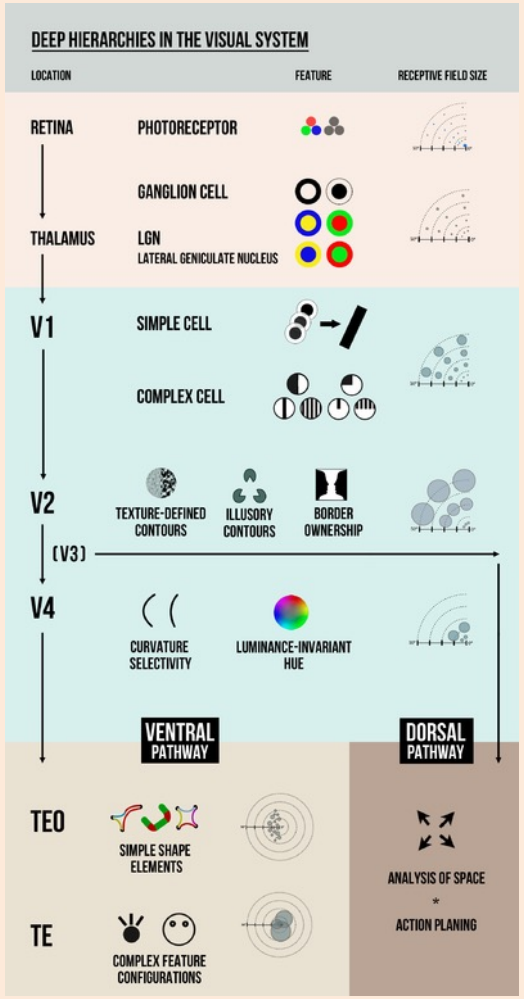


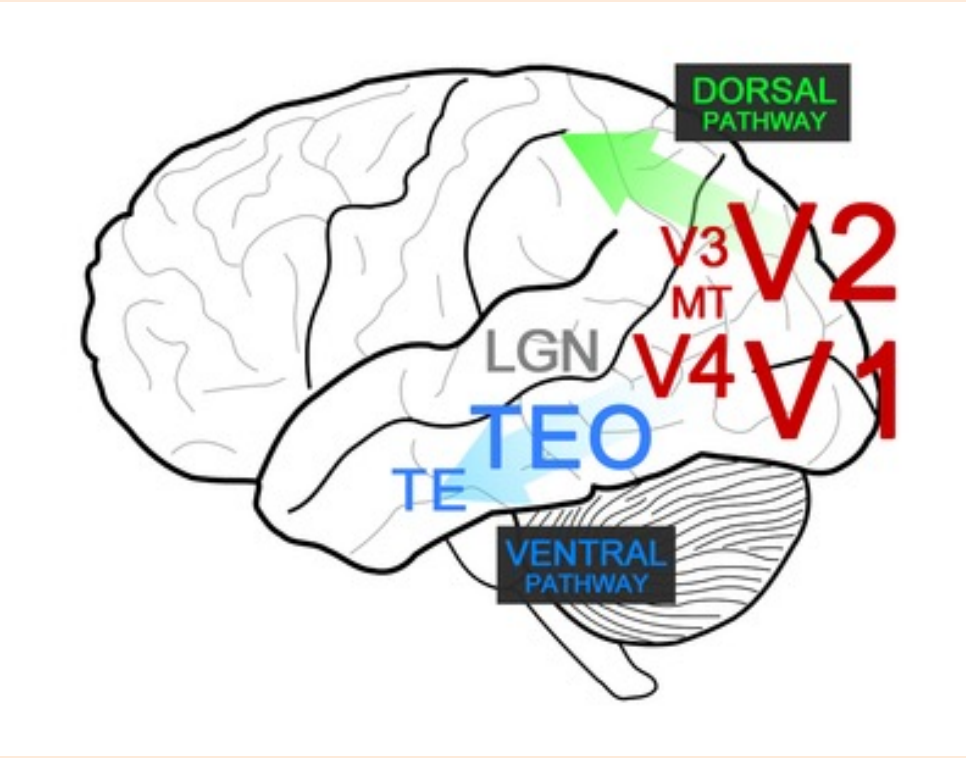
https://www.nytimes.com/2015/01/11/magazine/sebastian-seungs-quest-to-map-the-human-brain.html

Why Multiple Layers?

- And theories on the hierarchical organization of the visual system:



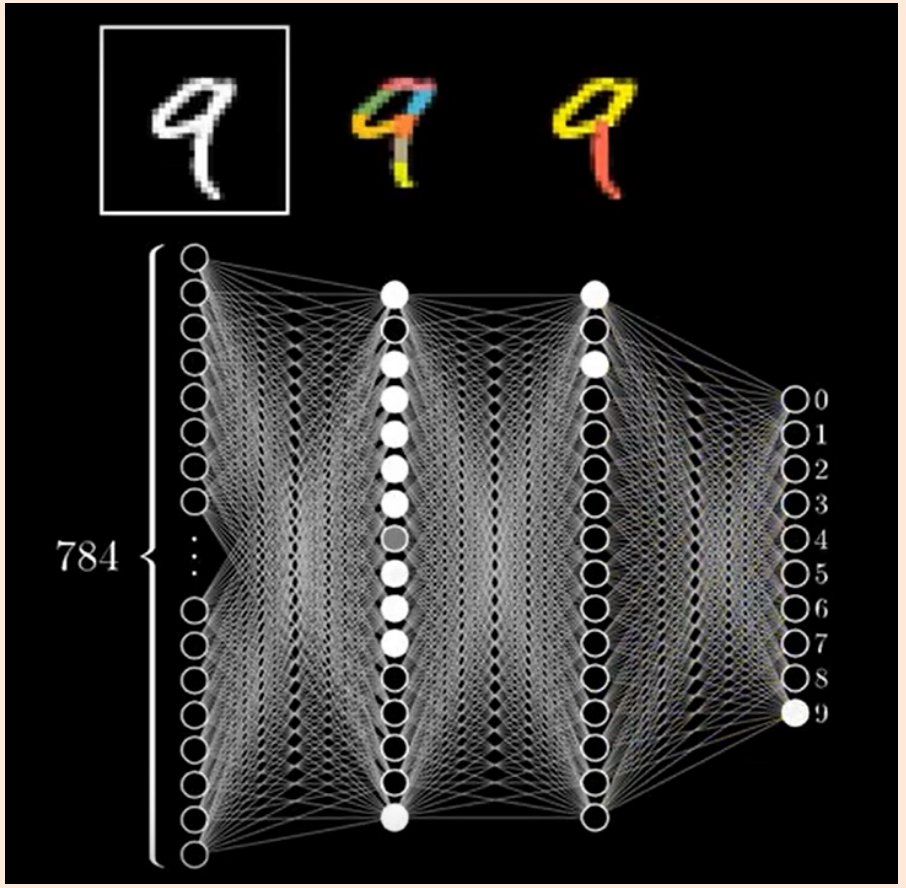




http://www.strokenetwork.org/newsletter/articles/vision.htm
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing

“Hierarchies of Parts” Intuition for Deep Learning

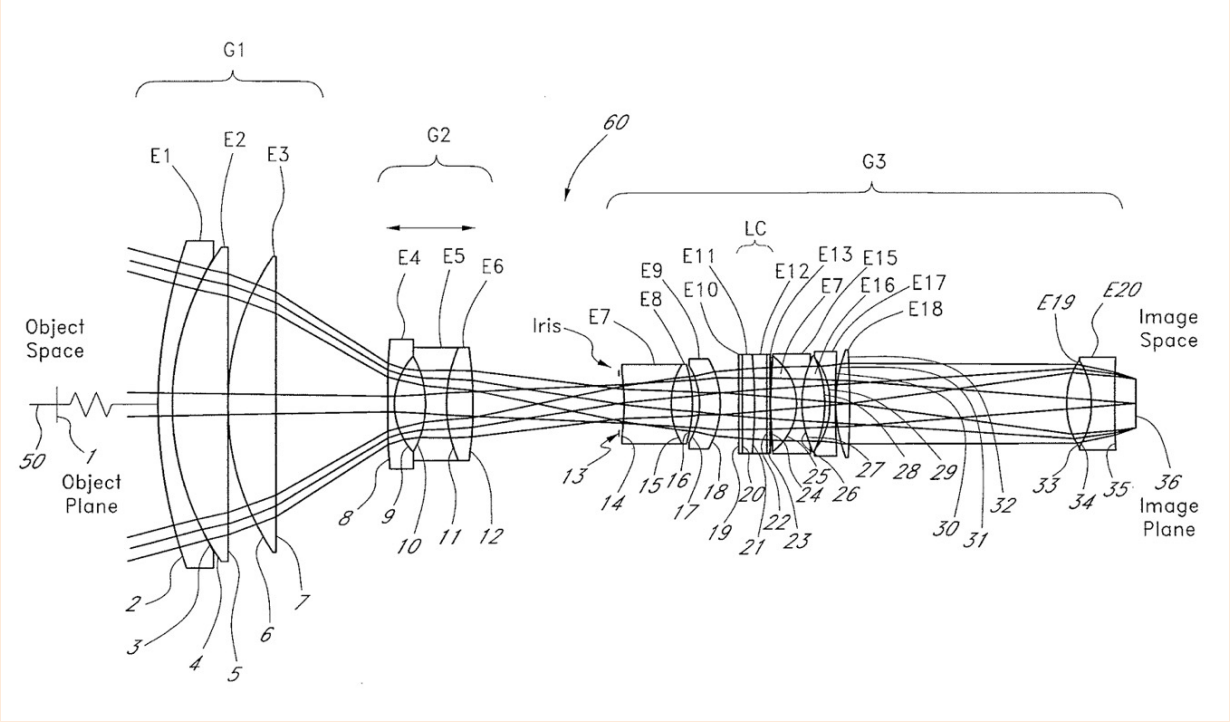
- Each “neuron” might recognize a “part” of a digit.
  - “Deeper” neurons might recognize combinations of parts.
  - Represent complex objects as combinations of simpler parts.

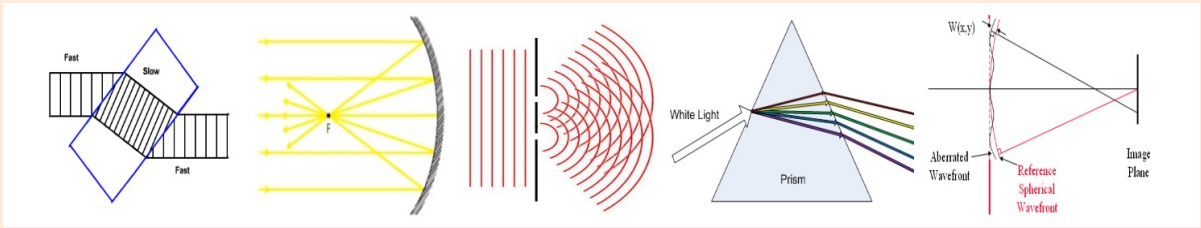


- Watch the full video here:
  - https://www.youtube.com/watch?v=aircAruvnKk

Why Multiple Layers?

- The idea of multi-layer designs appears in engineering too:
  - Deep hierarchies in camera design:





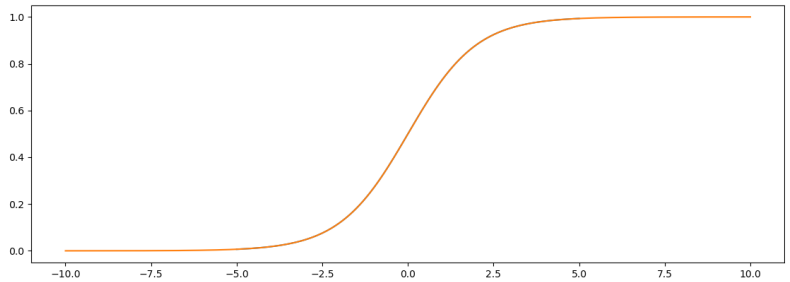
http://www.argmin.net/2018/01/25/optics/

Why Multiple Layers?

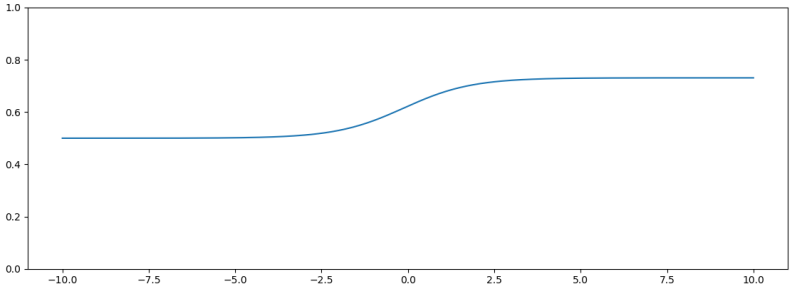
- There are also mathematical motivations for using multiple layers:
  - 1 layer gives us a universal approximator.
    - But this layer might need to be huge.
  - With deep networks:
    - Some functions can be approximated with exponentially-fewer parameters.
      - Compared to a network with 1 hidden layer.
    - So deep networks may need fewer parameters than “shallow but wide” networks.
      - And hence may need less data to train.
- Empirical motivation for using multiple layers:
  - In many domains deep networks have led to unprecedented performance.

New Issue: Vanishing Gradients

- Consider the sigmoid function:



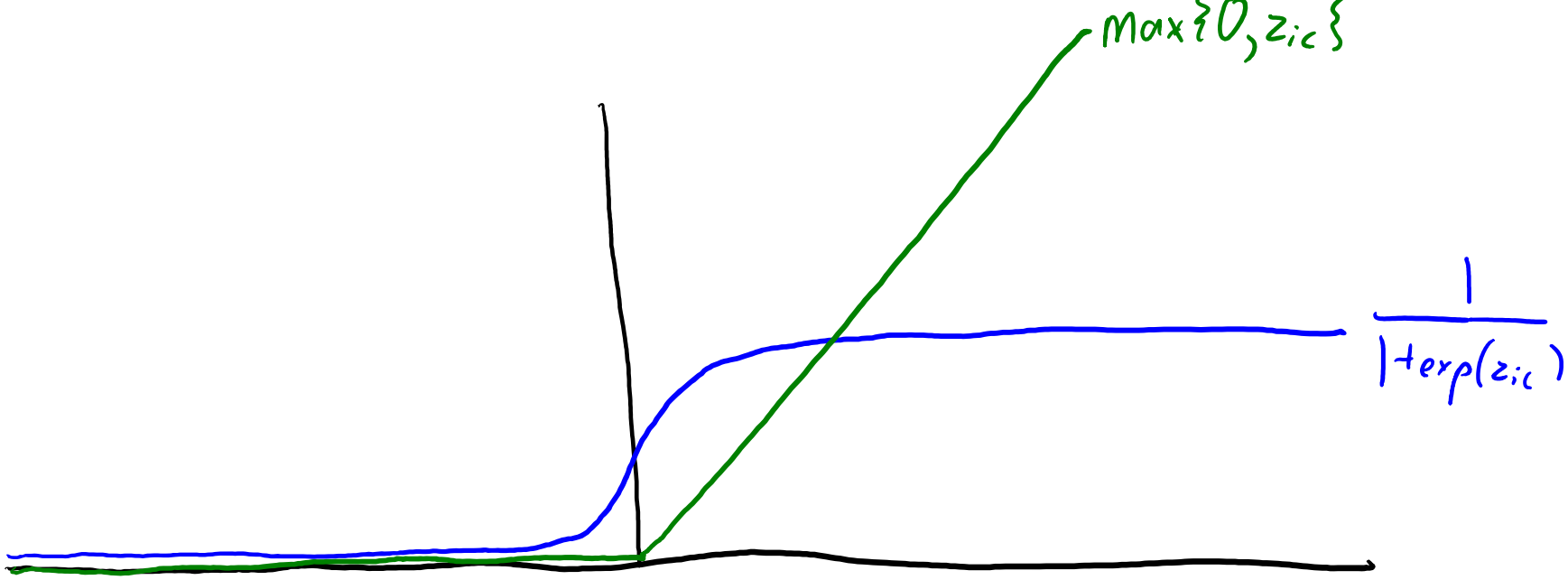
- Away from the origin, the gradient is nearly zero.
- The problem gets worse when you take the sigmoid of a sigmoid:



- In deep networks, many gradients can be nearly zero everywhere.
  - And numerically they will be set to 0, so SGD does not move.

Rectified Linear Units (ReLU)

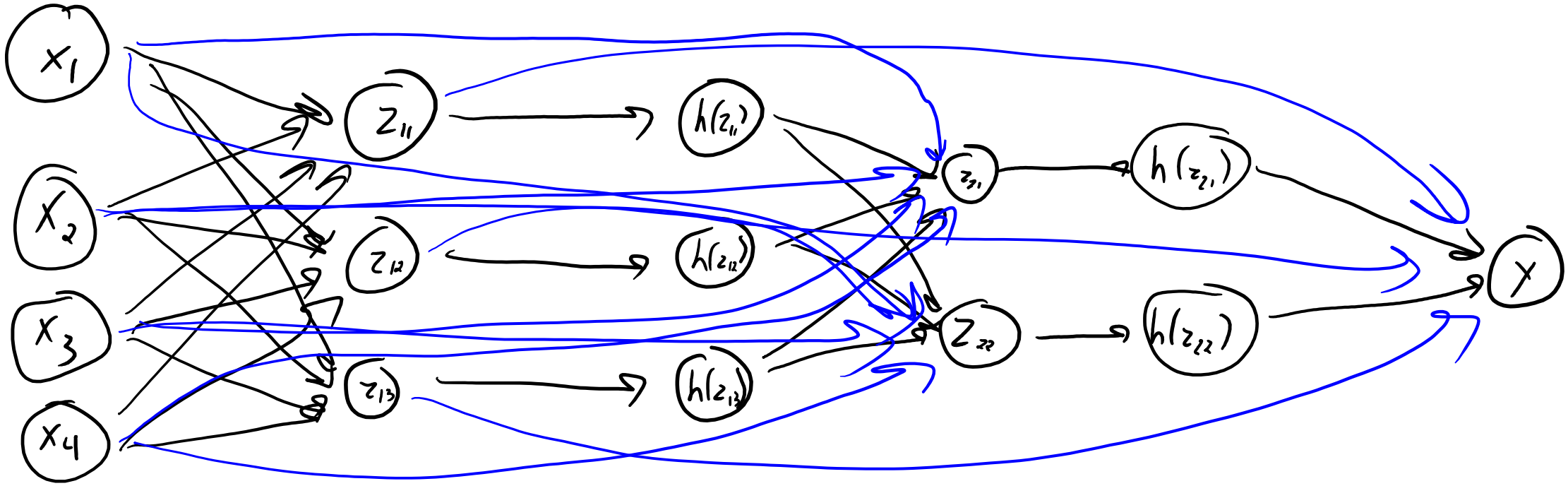
- Modern networks often replace sigmoid with perceptron loss (ReLU):



- Just sets negative values z_{ic} to zero.
  - Reduces vanishing gradient problem (positive region is never flat).
  - Gives sparser activations.
  - Still gives a universal approximator if size of hidden layers grows with 'n'.

Skip Connections Deep Learning

- Skip connections can also reduce vanishing gradient problem:



- Makes "shortcuts" between layers (so fewer transformations).
  - Many variations exist on skip connections exist.

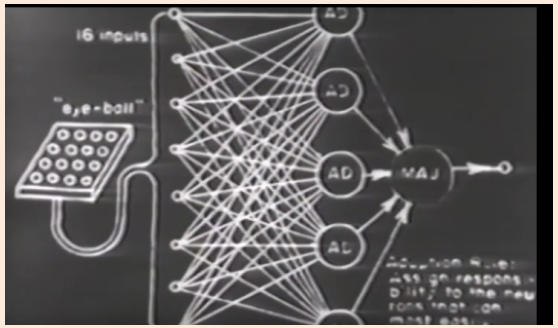
Summary

- Superiority of neural networks over linear models.
  - If we initialize with a linear model and use skip connections.
- Empirical "good news" for training neural networks with SGD:
  - With enough hidden units, SGD often finds a global minimum.
- Implicit regularization and double descent curves.
  - Possible explanations for why neural networks often generalize well.
- Over-parameterized models, that can fit data exactly.
  - SGD converges fast with a constant step size for these models.
- Deep learning:
  - Neural networks with multiple hidden layers.
  - ReLU activation function (vanishing gradient). **Adam** optimizer. Skip connections.

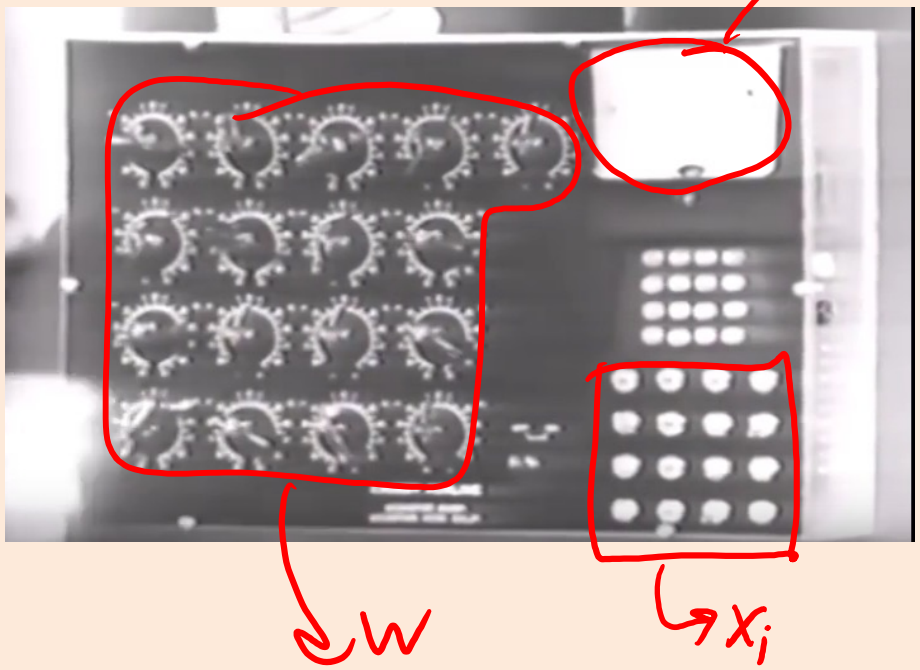
- Next time: where is my gradient?

ML and Deep Learning History

- 1950 and 1960s: Initial excitement.
  - Perceptron: linear classifier and stochastic gradient (roughly).
  - "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." New York Times (1958).
    - https://www.youtube.com/watch?v=IEFRtz68m-8
  - Object recognition assigned to students as a summer project



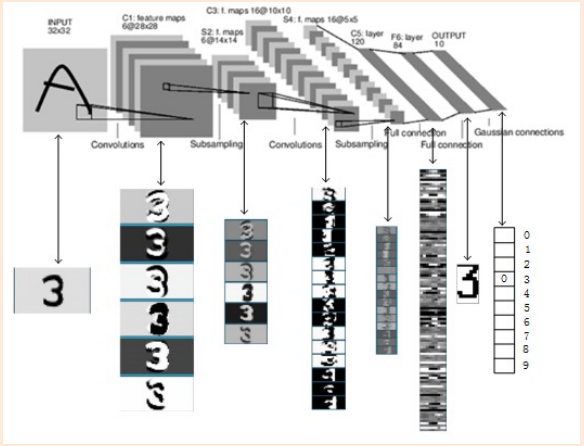
- Then drop in popularity:
  - Quickly realized limitations of linear models.

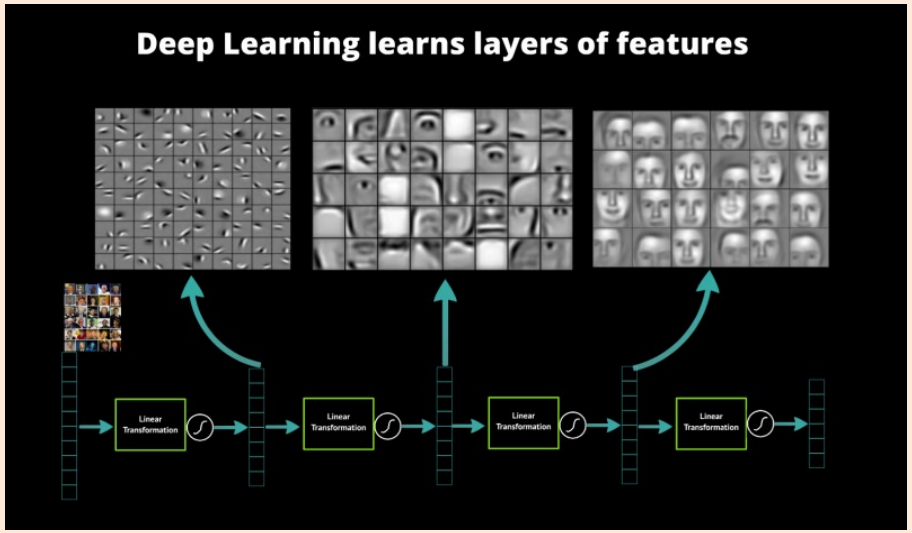


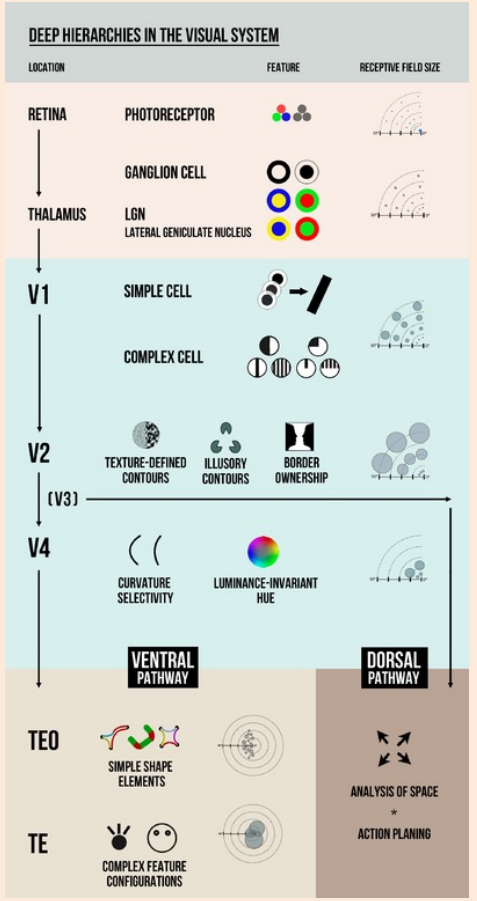
https://mitpress.mit.edu/books/perceptrons/

ML and Deep Learning History

- 1970 and 1980s: Connectionism (brain-inspired ML)
  - Want "connected networks of simple units".
    - Use parallel computation and distributed representations.
  - Adding hidden layers z_i increases expressive power.
    - With 1 layer and enough sigmoid units, a universal approximator.
  - Success in optical character recognition.







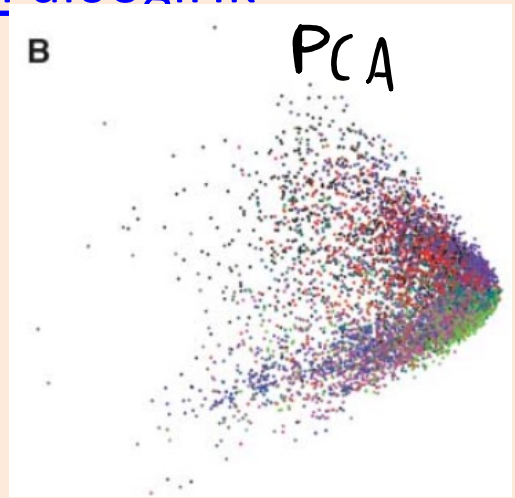
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing
http://www.datarobot.com/blog/a-primer-on-deep-learning/
http://blog.csdn.net/strint/article/details/44163869

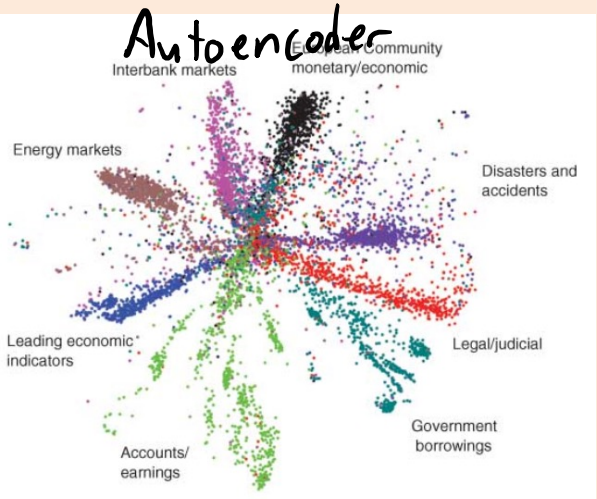
ML and Deep Learning History

- 1990s and early-2000s: drop in popularity.
  - It proved really difficult to get multi-layer models working robustly.
  - We obtained similar performance with simpler models:
    - Rise in popularity of logistic regression and SVMs with regularization and kernels.
  - Lots of internet successes (spam filtering, web search, recommendation).
  - ML moved closer to other fields like numerical optimization and statistics.

ML and Deep Learning History

- Late 2000s: push to revive connectionism as "deep learning".
  - Canadian Institute For Advanced Research (CIFAR) NCAP program:
    - "Neural Computation and Adaptive Perception".
    - Led by Geoff Hinton, Yann LeCun, and Yoshua Bengio ("Canadian mafia").
  - Unsupervised successes: "deep belief networks" and "autoencoders".
    - Could be used to initialize deep neural networks.
    - https://www.youtube.com/watch?v=KuPai0ogiHk

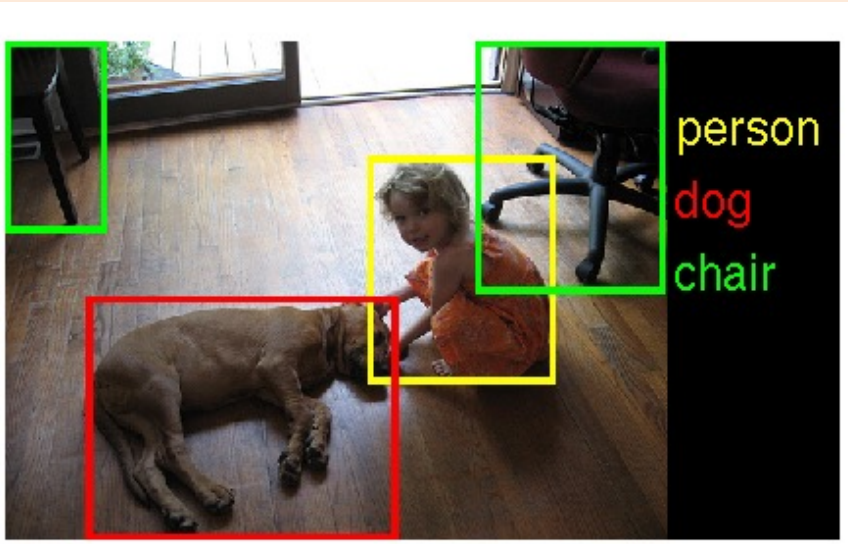




https://www.cs.toronto.edu/~hinton/science.pdf

2010s: DEEP LEARNING!!!

- Bigger datasets, bigger models, parallel computing (GPUs/clusters).
  - And some tweaks to the models from the 1980s.
- Huge improvements in automatic speech recognition (2009).
  - All phones now have deep learning.
- Huge improvements in computer vision (2012).
  - Changed computer vision field almost instantly.
  - This is now finding its way into products.



http://www.image-net.org/challenges/LSVRC/2014/

2010s: DEEP LEARNING!!!

- Media hype:
  - "How many computers to identify a cat? 16,000"
    New York Times (2012).
  - "Why Facebook is teaching its machines to think like humans"
    Wired (2013).
  - "What is 'deep learning' and why should businesses care?"
    Forbes (2013).
  - "Computer eyesight gets a lot more accurate"
    New York Times (2014).

- 2015: huge improvement in language understanding.