

Deep Learning Introduction

Ari Weinstein

Ellison Institute of Technology

Lecture Outline

- Introduction
- Theoretical Intuition
- Core Building Blocks
- Training Dynamics & Debugging, AdamW
- Advanced Issues

Introduction

- Goal is to ensure common core understanding:
 - What is needed for deep learning (DL) to work
 - DL building blocks that accomplish this
- Other lectures will focus on advanced topics (vision, sequence/text, reinforcement learning)
- Assume basic knowledge on feedforward networks and backprop
- This is a survey; goal is to give you some knowledge and you can dig into areas as needed later

Case Study: LeNet vs AlexNet (Introduction)

- What was needed to get deep learning to work
- There was a period of great excitement around NNs (late 80s-90s)
 - But early progress didn't continue, and approach was largely ignored

	LeNet-5	AlexNet
When	1998	2012
Classification	10 (digits)	1K (natural images)
Parameters	60K	60M
Corpus	60K b/w	1.2M rgb
Activation	tanh	relu
Regularization	none	dropout
Compute	CPU	GPU

Deep Learning in the past 10+ years (Introduction)

- Huge increase in data
- Huge advances in computational hardware for these models
 - New architectures that work well on hardware (ex/ transformers vs RNN)
- Controlling model complexity (regularization)
- Controlling activations and gradients (normalization)
- Not much in the way of meaningful science or mathematical advances

Theoretical Intuitions

- Bias/Variance Tradeoff: model complexity and fit
 - Addressed by regularization
- Exploding/Vanishing Gradients: keeping DL machinery working
 - Addressed by normalization

Bias/Variance Tradeoff (Theoretical Intuitions)

- In an ideal world, we know the form of the equation we are modelling
 - Works in Newtonian physics, not in language modelling
- $Error = Bias^2 + Variance + \epsilon$
- Simpler models: more bias, less variance (underfitting)
- Complex models: less bias, more variance (overfitting)
- Data ~makes overfitting go away
- Double-descent: overparametrized models don't always overfit

Regularization (Theoretical Intuitions)

- Because of bias/variance tradeoff, in naïve modelling there is U-shaped profile of accuracy across model complexity
- Ideally, modelling would control complexity and try to find the sweet spot by itself
- There is a whole literature on regularization, we heavily rely on
 - Dropout (especially useful in lower-data regimes)
 - Weight decay in AdamW (L2 regularization)
- Data augmentation

Vanishing, Exploding Gradients (Theoretical Intuitions)

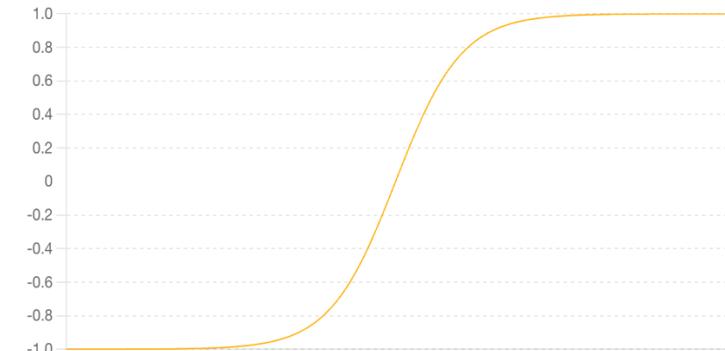
- For Loss L , weight w , layer n
- $\frac{\partial L}{\partial w_n} \propto \left(\prod_n^N \frac{\partial w_{n+1}}{\partial w_n} \right)$ -- chain rule
- Because there is a long chain of products (large N) this naturally wants to go one of two ways:
- Most grads < 1 ; gradient quickly decays to ≈ 0
 - Training stops
- Most grads > 1 ; gradient quickly explodes
 - Huge weight updates, training diverges

Core Building Blocks

- Activation Functions
 - How can we make sure activations propagate forward, gradients propagate backward effectively
- Normalization
 - How to speed and stabilize training

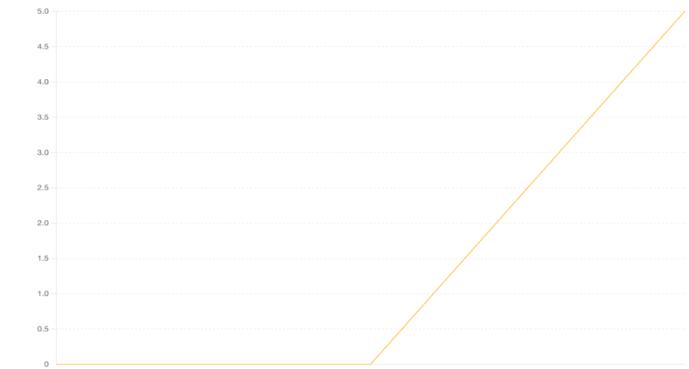
TanH (Activation Functions)

- Theoretically, any nonlinearity is fine
- $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Problem with vanishing gradients
 - Max grad of 1 at 0
 - “Saturates” with $\tanh'(x) \rightarrow 0$
- Extreme simplification helped DL progress



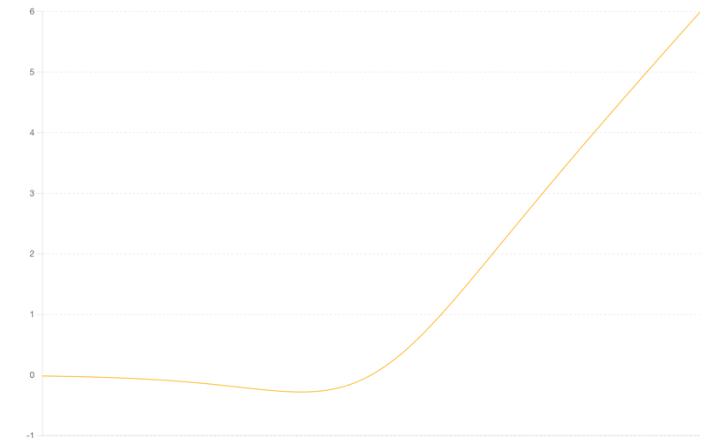
Rectified Linear Unit (Activation Functions)

- $ReLU(x) = \max(0, x)$
- $ReLU'(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$
- One of the key pieces for DL
- Gradients (on positive activation) flow perfectly
- But also kills gradient flow on negative activation (like tanh)
 - Gets stuck here, leading to “Dead ReLU”
- Yet another fix was needed



Swish (Activation Functions)

- $\text{Swish}(x) = x \sigma(\beta x)$
- For large $x^+ \approx x$, $x^- \approx 0$
- Similar to ReLU, but "softer"
- Has nonzero gradients, does not "die"
- Activations are example of engineering (iterative refinement) driving DL progress
- Other advances like SwiGLU more complex, used in transformers



Residual Connections (Core Building Blocks)

- Don't make layers relearn a whole new function, just a change
- $h_{n+1} = h_n + F(h_n)$, for activation h , layer n
- $\frac{\partial L}{\partial h_n} = \frac{\partial L}{\partial h_{n+1}} \left(\frac{\partial F(h_n)}{\partial h_n} + I \right)$
 - There is always a direct gradient path from loss to early layers
 - Even if $\frac{\partial F(h_n)}{\partial h_n}$ suffers from ~0 gradients; solves vanishing gradient problem
- Tends to be an easier function to learn, and more stable
- Allowed first networks to be trained into 1K layers

Normalization (Core Building Blocks)

- Forces activations to be of a consistent distribution
- Helps resolve vanishing/exploding gradients (activations are never overall too big or small)
 - Easier to train deep nets
- Helps early in training where dynamics can be unstable
- Reduces the need for careful initialization
- BatchNorm, LayerNorm, RMSNorm

Different forms (Normalization)

- Batch Norm: normalizes activations of a single neuron in a batch to have zero mean, unit variance
 - More common in CNNs w/large batches
- Layer Norm: normalizes the activation of a single sample across a layer
 - Common in Transformers (smaller batch sizes)
- RMS Norm: Similar to Layer Norm but simpler
 - Most modern transformers use this, as cheaper to compute

Outline: Training Dynamics and Optimizers

- Initialization: Setting your network up for success
- Optimizers (AdamW)
- Learning Rate Schedules:
- When Training Goes Wrong: Numeric and other Issues

Initialization (Dynamics, Optimizers)

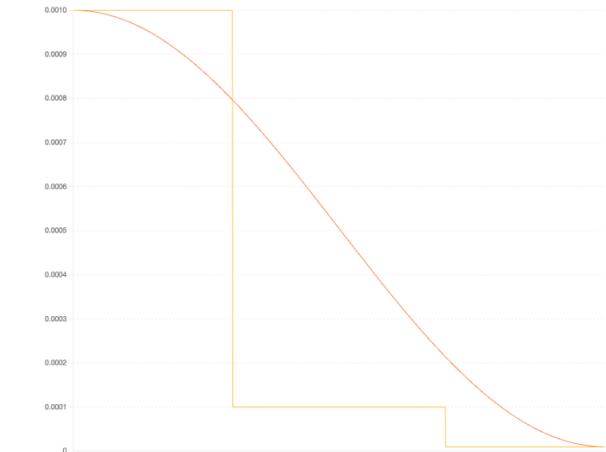
- Initialization used to be critical, poor initialization lead to vanishing/exploding activation/gradients
- Normalization largely fixes this, but init still matters
- (Glorot) Xavier initialization: $w_{ij} = \mathcal{N}\left(0, \frac{2}{n_{in}+n_{out}}\right)$
- Other common techniques are similar and tailored to activation/arch

AdamW (Dynamics, Optimizers)

- Used almost everywhere now
- $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$ (mean of gradients)
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$ (mean of squared gradients, for adaptive scaling)
- $w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{v_t} + \epsilon} - \eta \lambda w_t$
- For LR η , momentum β , weight decay λ , stability term ϵ
- Regularization built into the optimizer (λ)
- Robust to hyperparams

Learning Rate Schedules (Dynamics, Optimizers)

- High learning rate dangerous with random init
- Usually start with low η and “warm up”
 - $\eta_t = \eta_{max} \frac{t}{T_W}$
- After warm up period reduce η
- Step decay: $\eta_t = \eta_0 \gamma^t$, for epoch t
 - Change often too abrupt; not common anymore
- Cosine annealing:
 - $\eta_t = \eta_{min} + 0.5(\eta_{max} - \eta_{min})(1 + \cos \pi t/T)$
- Most common is warmup followed by cos annealing (small, big, small updates)



Numerics (Generalization & Advanced Issues)

- Can be hard to track down (use tooling)
- Computers store approximations of reals, can go wrong
 - Overflow (becomes inf), underflow (becomes 0), ruins gradients
- Issues can arise at any point (early, late in training)
- Use gradient clipping, loss scaling, check for NaN and inf
- When developing, make smaller nets with large representations (float32), then move to experimenting with less/mixed precision on larger networks
- Get shallow wide nets to work, then go deeper

Numerics (Generalization & Advanced Issues)

- Warmup
- Normalization
- Gradient clipping
- Work in log-space: $\log p(x_0 \dots x_n) = \sum \log p(x_i)$
- Training:
 - Plateaus: η too small
 - Spikes: Check for NaN, Inf, zeros where they don't belong (log, divisor)
 - Divergence: η too high

Debugging Checklist

- Make sure any data can run through model and produce gradient
- If dataset is not vetted, check data cleanliness (inf, Nan, missing val, ranges)
- Overfit tiny subset of data
- Go from small to large learning rates during testing
- Examine gradient norms, especially early in training
 - If large, use normalization, gradient clipping

Outline: Advanced Topics

- Real-World Data Issues
 - When life isn't as easy as MNIST
- Representation Learning
- Fine Tuning

Real-World Data Issues (Advanced Topics)

- Increasingly important as DL systems are in use in society
- IID assumption is core in ML often doesn't hold
- Importance of diverse / representative data
- Data balance
- Error types (type 1 vs 2)
- Data pedigree
- Train/test split, data leakage
- Calibration
- Runaway feedback loops

<https://worksinprogress.co/issue/the-algorithm-will-see-you-now/>

<https://www.nature.com/articles/s41576-025-00839-w>

<https://hdsr.mitpress.mit.edu/pub/wot7mhc1/release/10>

<https://academic.oup.com/jrssig/article/13/5/14/7029190>

Representation Learning (Advanced Topics)

- Fundamental strength of deep learning is ability of models to learn meaningful representations of data
- Take potentially disparate modalities of data and produce rich and meaningful vector embeddings

Feature #34M/31164355 Golden Gate Bridge feature example

The feature activates strongly on English descriptions and associated concepts

They also activate in multiple other languages on the same concepts

And on relevant images as well

in the Presidio at the end (that's the huge park right next to the Golden Gate bridge), perfect. But not all people repainted, roughly, every dozen years." "while across the country in san francisco, the golden gate bridge was it is a suspension bridge and has similar coloring, it is often compared to the Golden Gate Bridge in San Francisco, US

ゴールデン・ゲート・ブリッジ。金門橋は、アメリカ西海岸のサンフランシスコ湾と太平洋が接続するゴールデンゲート海

골든게이트교 또는 금문교는 미국 캘리포니아주 골든게이트 해협에 위치한 현수교이다. 골든게이트교는 캘리포니아주 샌프란시스코와 알리안스를 연결하는 다리이다.

МОСТ ЗОЛОТОЕ ВОРОТА – ВИСЯЧИЙ МОСТ ЧЕРЕЗ ПРОЛИВ ЗОЛОТОЕ ВОРОТА. ОН СОЕДИНЯЕТ ГОРОД САН-ФРАНСИСКО И АЛЯНС

Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet

Representation Learning (Advanced Topics)

- Good representations allow for robust generalization
- Enables broad downstream use, and efficient fine tuning
- In modern techniques, driven primarily by unsupervised learning
 - Autoregressive prediction
 - Contrastive encoding
- What makes foundation models so flexible and powerful

Fine Tuning (Advanced Topics)

- Increasingly important, as foundation models dominate modern ML
- Sometimes you can't zero-shot your way to what you want
 - Ex/ RL loops for LLM coding
- Usually no need to tune entire model
 - LoRA & others allow fine-tuning tiny amount of params
 - Helpful when Hardware isn't same as original training
- Often only small amount of data is needed

Distillation

- Transfer capabilities from one model (teacher t) to another (student s)
 - Usually large to small model
 - But can be for other reasons; across modality
- Identify data you care about, query source model, and train target to match class probabilities
- $L = \alpha L_{CE}(y, p_s) + (1 - \alpha)T^2 L_{KL}({p_t}^T, {p_s}^T)$

Conclusion

- Discussed:
 - Fundamentals
 - Building blocks
 - Training & Numerics
 - Important considerations
 - Advanced topics
- Please feel free to reach out to me at aweinstein@eit.org!
- Please ask questions!