On the Expressive Power of Deep Neural Networks

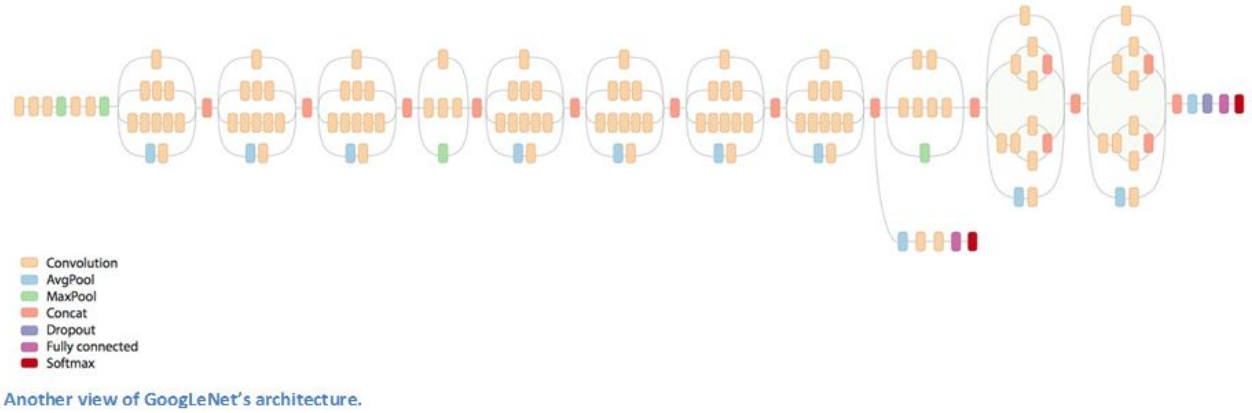
Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, Jascha Sohl Dickstein

Tom Brady

Deep Neural Networks

- Recent successes in using Deep neural networks for image classification, reinforcement learning etc.

f() = cat



Another view of GoogLeNet's architecture.

But why do they work?

- Lack of theoretical understanding of the functions a Deep Neural network is able to compute
- Some work into shallow networks
  - Universal approximation results (Hornik et al., 1989; Cybenko, 1989)
  - Expressivity comparisons to boolean circuits (Maass et al., 1994)
- Some work into deep networks
  - Establishing lower bounds on expressivity
    - E.g. Pascanu et al., 2013; Montufar et al., 2014 $\forall W \mathcal{T}(F_{A_1}([0,1];W)) < \mathcal{T}(F_{A_1}([0,1];W_0)$
  - But previous approaches use hand-coded constructions of specific network weights
  - Functions studies are unlike those learned by networks trained in real life
- Lacking:
  - Good understanding of “typical” case
  - Understanding of upper bounds
    - Do existing constructions approach the upper bound of expressive power of neural networks?

Contributions

- Measures of expressivity to capture expressive power of architecture
- Activation Patterns
  - Tight upper bounds on the number of possible activation patterns
- Trajectory length
  - Exponential growth in trajectory length as function of depth of network
  - small adjustments in parameters lower in the network can result in large changes later
  - Trajectory Regularization
    - Batch normalization works to reduce trajectory length
    - Why not directly regularize on trajectory length?

Expressivity

- Given architecture A, associated function $F_A(x; W)$
- Goal:
  - How does this function change as A changes for values of W encountered in training, across inputs x
- Difficulty:
  - High dimensional input, quantifying F over input space is intractable
- Alternative:
  - Study **one dimensional trajectories** through input space

Trajectory

Definition: Given two points, $x_0, x_1 \in \mathbb{R}^m$, we say $x(t)$ is a trajectory (between x_0 and x_1) if $x(t)$ is a curve parametrized by a scalar $t \in [0, 1]$, with $x(0) = x_0$ and $x(1) = x_1$.

Some trajectories:

- Line x(t) = tx1 + (1 - t) x0
- Circular arc x(t) = cos(πt/2)x0 + sin(πt/2)x1
- May be more complicated, and possibly not expressible in closed form

Measures of Expressivity: Neuron Transitions

- Given network with piecewise linear activations (e.g. ReLU, hard tanh), the function it computes is also piecewise linear
- Measure expressive power by counting number of linear pieces
- Change in linear region caused by a **neuron transition**
  - transitions between inputs x, x + δ if activation switches linear region between x and x + δ.
  - E.G. ReLU from off to on or vice versa
  - Hard tanh from -1 to linear middle region to saturation at 1
- For a trajectory x(t), can define $\mathcal{T}(F_A(x(t); W))$ as the **number of transitions** undergone by output neurons as we sweep the input along x(t)

Measures of Expressivity: Activation Pattern

Activation pattern $\mathcal{AP}(F_A(x; W))$

- A String of length number of neurons from set
  - {0, 1} for ReLUs
  - {−1, 0, 1} for hard tanh
- Encodes the linear region of the activation function of every neuron, for an input x and weights W

Can also define $\mathcal{A}(F_A(x(t); W))$ the number of distinct activation patterns as we sweep x along x(t)

- Measures how much more expressive A is over a simple linear mapping

Upper Bound for Number of Activation Patterns

Theorem 1. (Tight) Upper Bound for Number of Activation Patterns *Let* $A_{(n,k)}$ *denote a fully connected network with* n *hidden layers of width* k*, and inputs in* $\mathbb{R}^m$*. Then the number of activation patterns* $\mathcal{A}(F_{A_{n,k}}(\mathbb{R}^m; W)$ *is upper bounded by* $O(k^{mn})$ *for ReLU activations, and* $O((2k)^{mn})$ *for hard tanh.*

Trajectory transformation exponential with depth

- Trajectory increasing with the depth of a network

Definition: Given a trajectory, $x(t)$, we define its length, $l(x(t))$, to be the standard *arc length*:

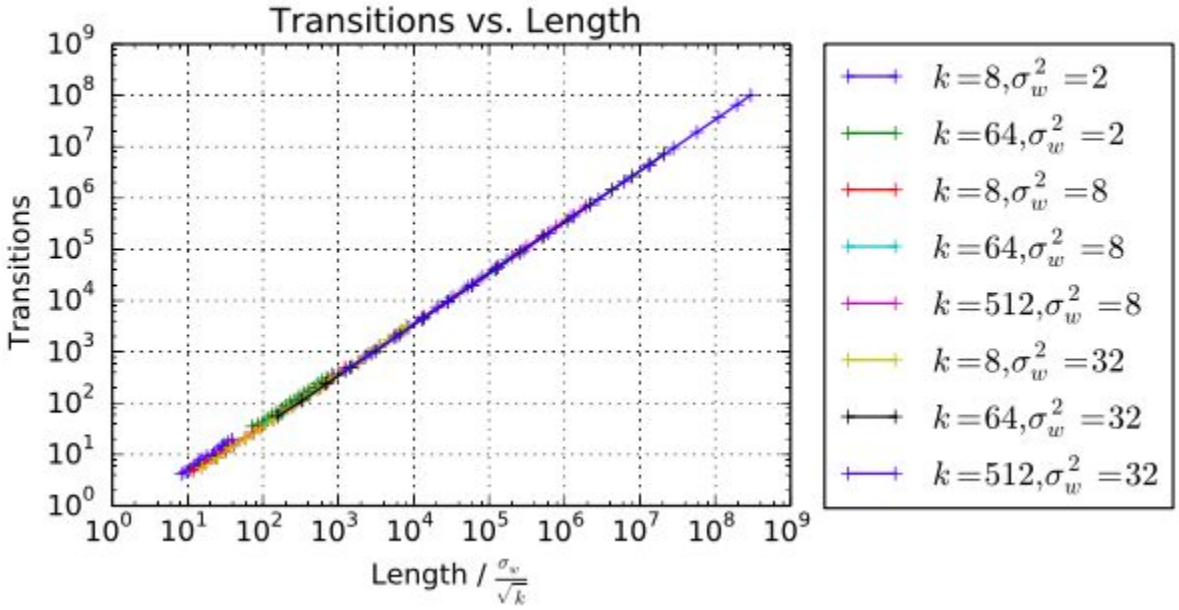
$$l(x(t)) = \int_t \left\| \frac{dx(t)}{dt} \right\| dt$$

- Image of the trajectory in layer d of the network $z^{(d)}(x(t)) = z^{(d)}(t)$
- Proved that For a fully connected work with
  - n hidden layers each of width k
  - Weights ~N(0, σw2/k)
  - Biases ~N(0, σb2)

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^d l(x(t))$$

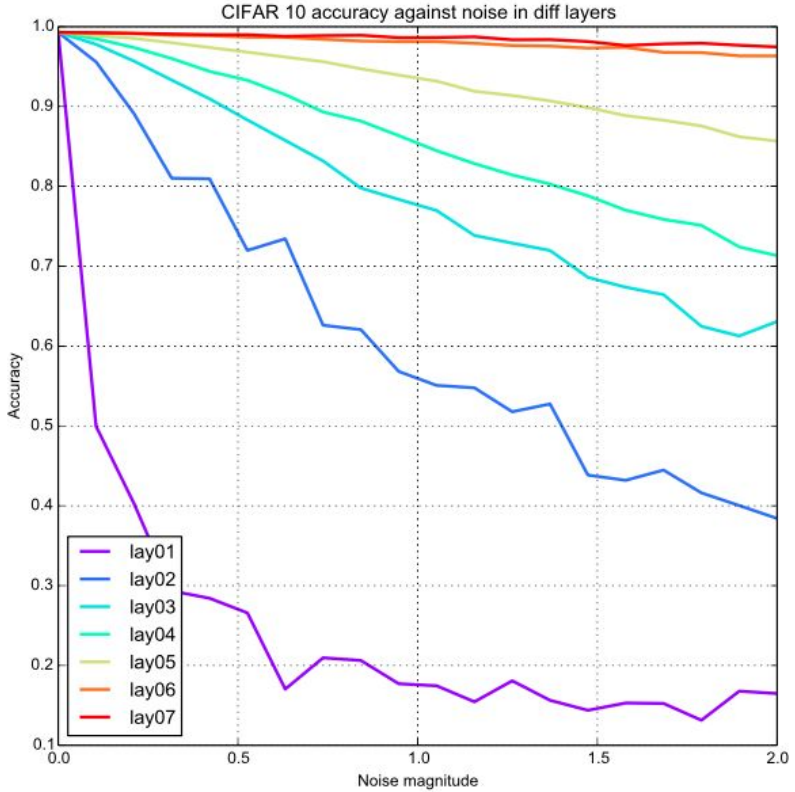
for hard tanh

Number of transitions is linear in trajectory length

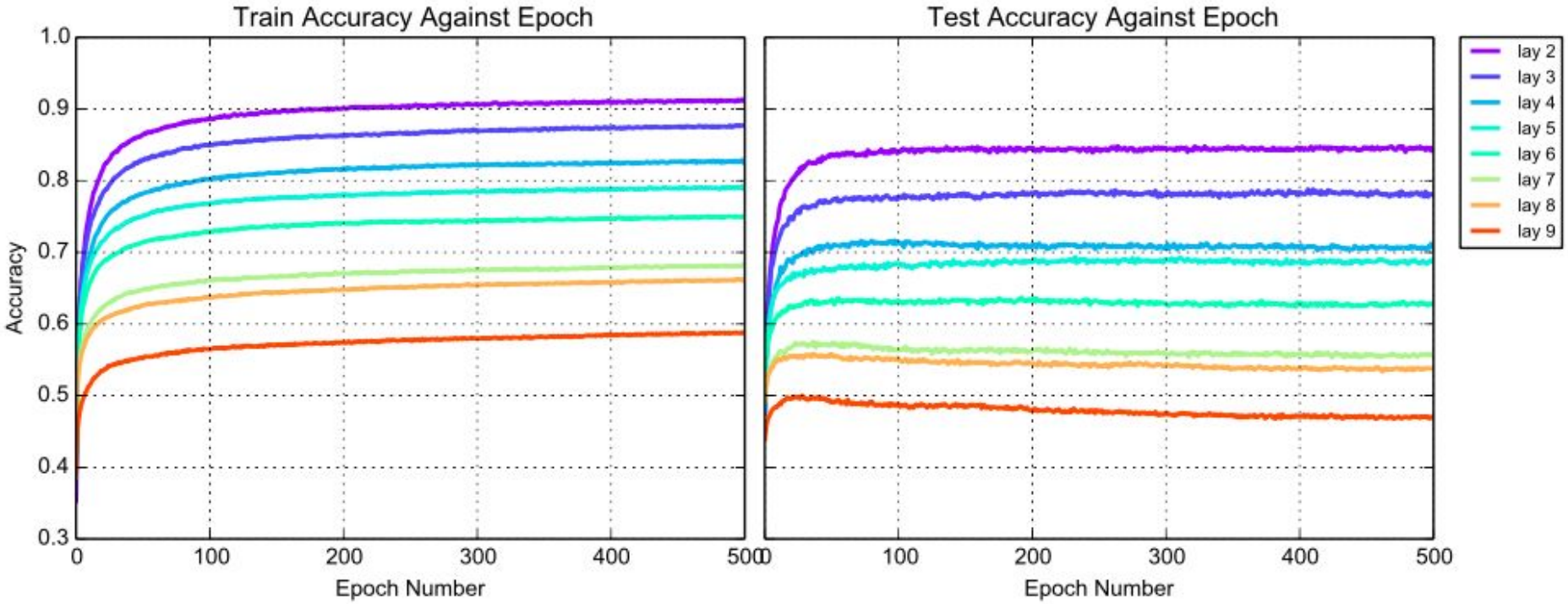


Early layers most susceptible to noise

A perturbation at a layer grows exponentially in the *remaining depth* after that layer.



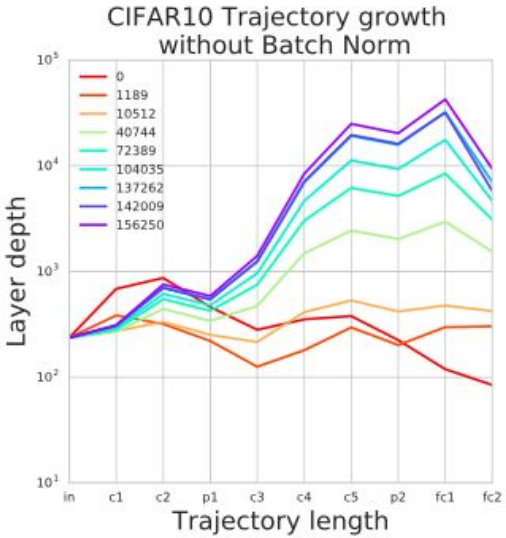
Early layers most important in training

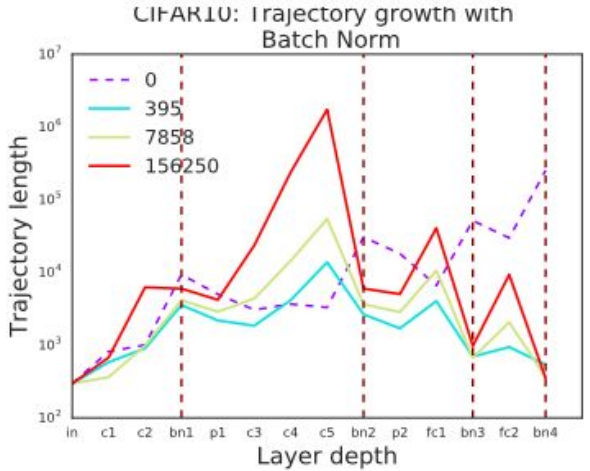


Trajectory Regularization

- Higher trajectory, higher expressive ability
- But also more unstable
- Regularization seems to be controlling trajectory length

Wrong axis labels →

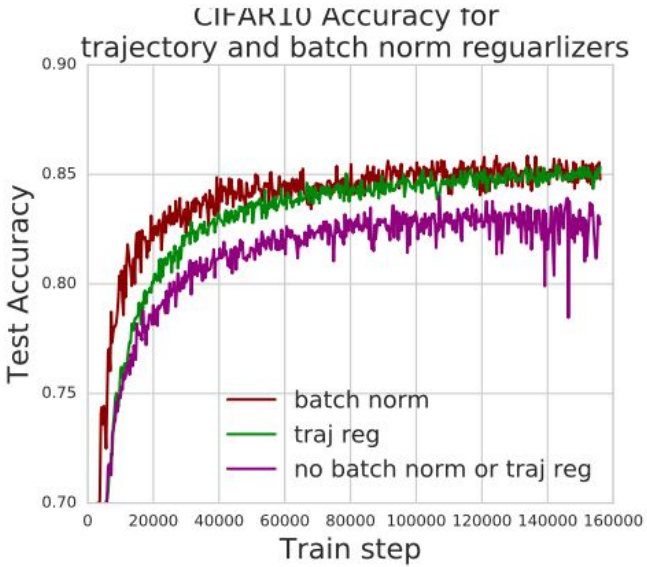




Trajectory Regularization

- add to the loss λ(current length/orig length)

- Replaced each batch norm layer of the CIFAR10 conv net with a trajectory regularization layer



Contributions

- Measures of expressivity to capture expressive power of architecture
- Activation Patterns
  - Tight upper bounds on the number of possible activation patterns
- Trajectory length
  - Exponential growth in trajectory length as function of depth of network
  - small adjustments in parameters lower in the network can result in large changes later
  - Trajectory Regularization
    - Batch normalization works to reduce trajectory length
    - Why not directly regularize on trajectory length?

Conclusions

- This paper equips us with more formal tools for analyzing the expressive power of networks
- Better understanding of importance of early layers: how and why
- Trajectory regularization is an effective technique, grounded in notion of expressivity
- Further work needed investigating trajectory regularization
- Trajectory has possible implications for understanding adversarial examples