The Hitchhiker's Guide to TensorFlow:

Beyond Recurrent Neural Networks

(sort of)

Keith Davis

@keithdavisiii
iamthevastidledhitchhiker.github.io

Topics

- Kohonen/Self-Organizing Maps
- LSTMs in TensorFlow
- GRU vs LSTM
- Performance

Topics

- **Kohonen/Self-Organizing Maps**
- LSTMs in TensorFlow
- GRU vs LSTM
- Performance

Kohonen Maps: Overview

- **Background:**
  - More commonly known as Self-Organizing Maps.
  - Technically a neural network, but has similarities to clustering algorithms like K-Means.
  - Offers a way to visualize high-dimensional data in a 2D or 3D space.

Kohonen Maps: Overview

- **Implementations (Python 2 & 3)**
  - PyMVPA
  - py-kohonen
  - sevamoo/SOMPY
  - JustGlowing/minisom

Kohonen Maps: Overview

- **Simplified Steps:**

1. Initialize random weights for each node in an m X n grid

2. Select random training vector and identify a node with weights that are closest to the training vector. (This node is known as the best matching unit, or BMU)

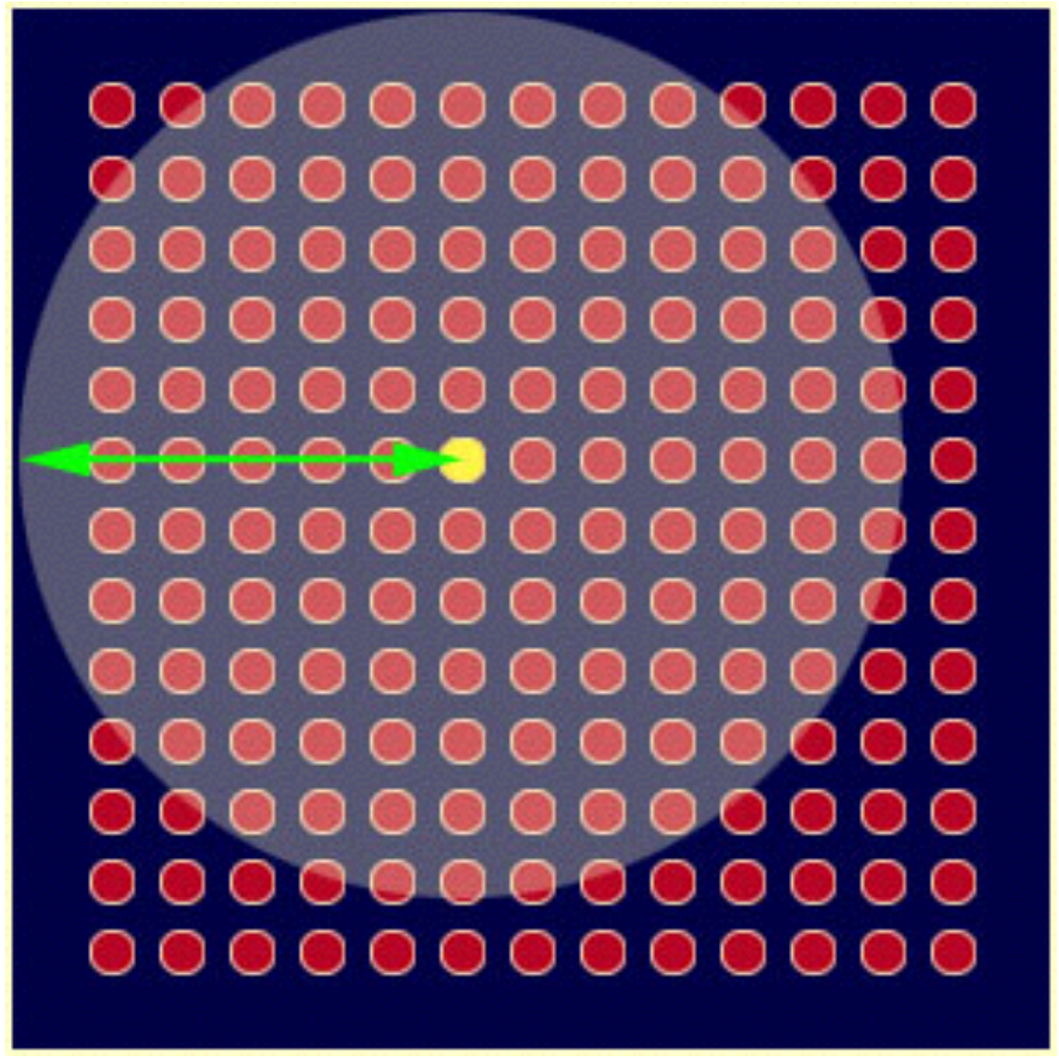
3. Find all nodes within neighborhood of BMU and adjust their weights to make them more similar to the BMU.

4. Adjust the learning rate and neighborhood size

5. Repeat steps 2 through 4 for a predetermined # of iterations.

Kohonen Maps: Overview

Neighborhood Function



source: http://www.ai-junkie.com/ann/som/som3.html

SOMs in TensorFlow: Prep

- **First we need data...**
  - Step 1: Find a data set. (WorldBank Data)
  - Step 2: Prepare the data.
    - Eliminate redundant or poorly formatted metrics.
    - NAs: To impute or not to impute?
    - Group similar features together
- Step 3: Output data

SOMs in TensorFlow: Code

- **Inputs**
  - m X n array of training data
  - training iterations
  - Dimensions (columns of data)
  - Sigma (radius of neighborhood for BMU)
  - Alpha (learning rate)
  - Dimensions of node grid
- **Setup**
  - Generate node grid
  - Assign random weights to each node

SOMs in TensorFlow: Code

- **Training**
  - For each training vector:
    - For each node in grid:
      - Compare weight distance
    - Select node w/smallest distance as BMU
    - Update neighbor nodes weights to be more similar to BMU.
  - Save grid and vector mappings to a list.
  - Repeat for n iterations.
- **Outputs**
  - List of grids w/weights and vector mappings for training vectors.

SOMs in TensorFlow: Code

- **Visualizing Results**
  - Iterate through list of grids and vector mappings.
  - For each dimension in the training data, plot vector mappings to each grid and map the topography.

Demo TensorFlow

- This slide intentionally left blank.

Kohonen Maps: Uses

- Data projection and visualization: SOMs preserve topology.
- Help identify clustering tendency
- Visualization of dimensionality reduction. Can be used to visualize the relative mutual relationships among the data.

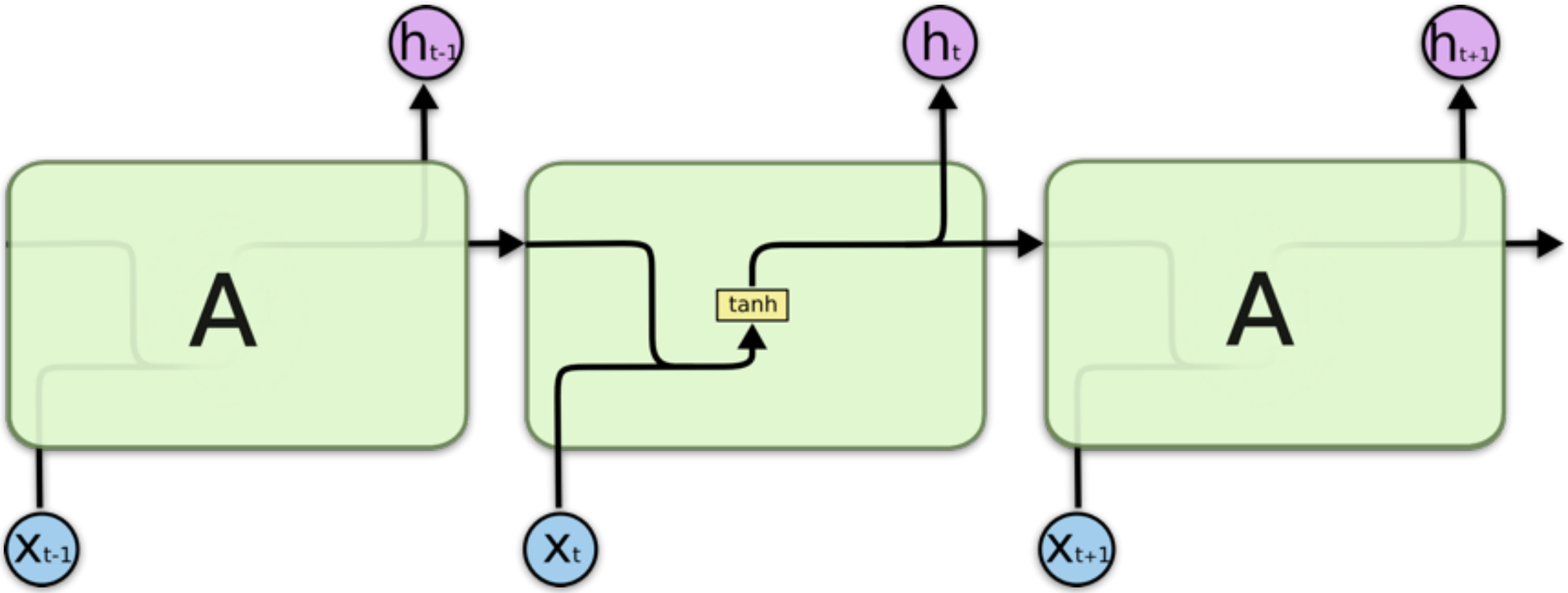
Kohonen Maps: Common Issues

- Interpretability
  - What does “closeness” even mean?
  - Still easier to understand than other ANNs.
- Heavily influenced by metrics used.
  - Tends to be more useful as an interim step, although “big picture” visualizations are still fairly useful. (Initially used for speech recognition)

Topics

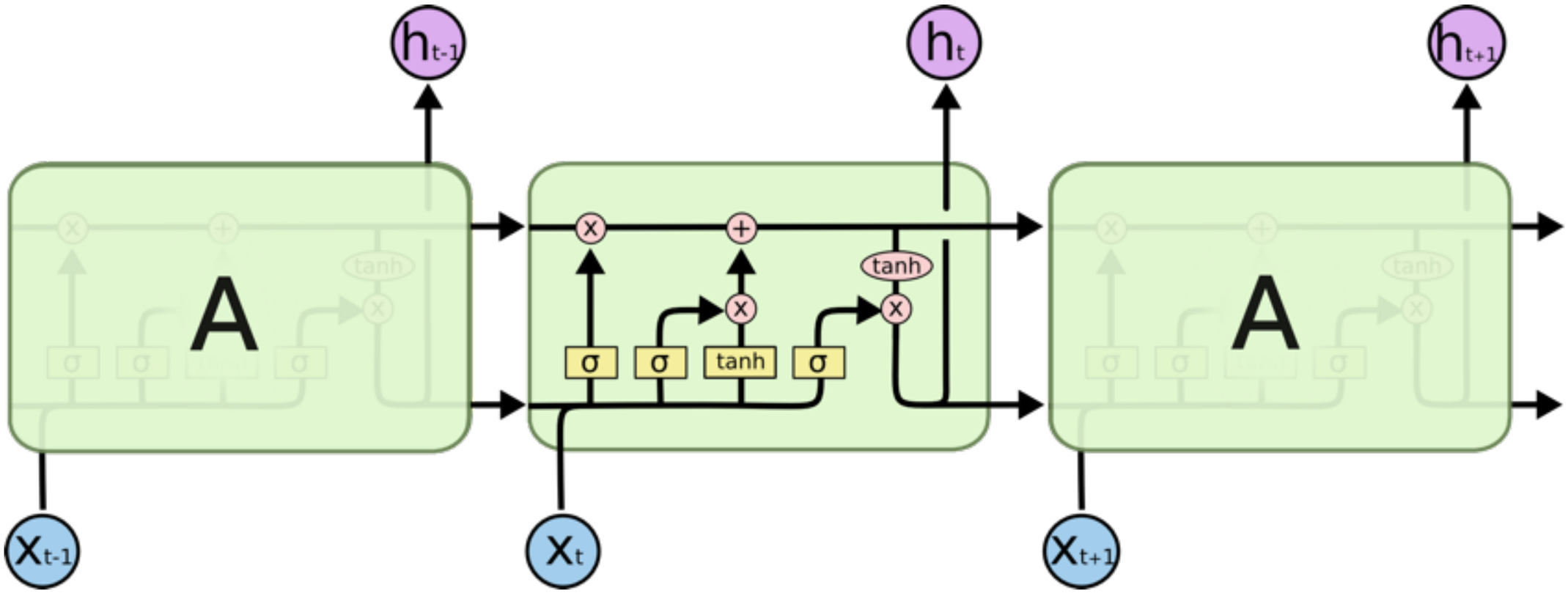
- Kohonen/Self-Organizing Maps
- **LSTMs in TensorFlow**
  - **GRU vs LSTM**
  - **Andrej Karpathy's Char-RNN**
  - **Performance**

RNNs



source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs



source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

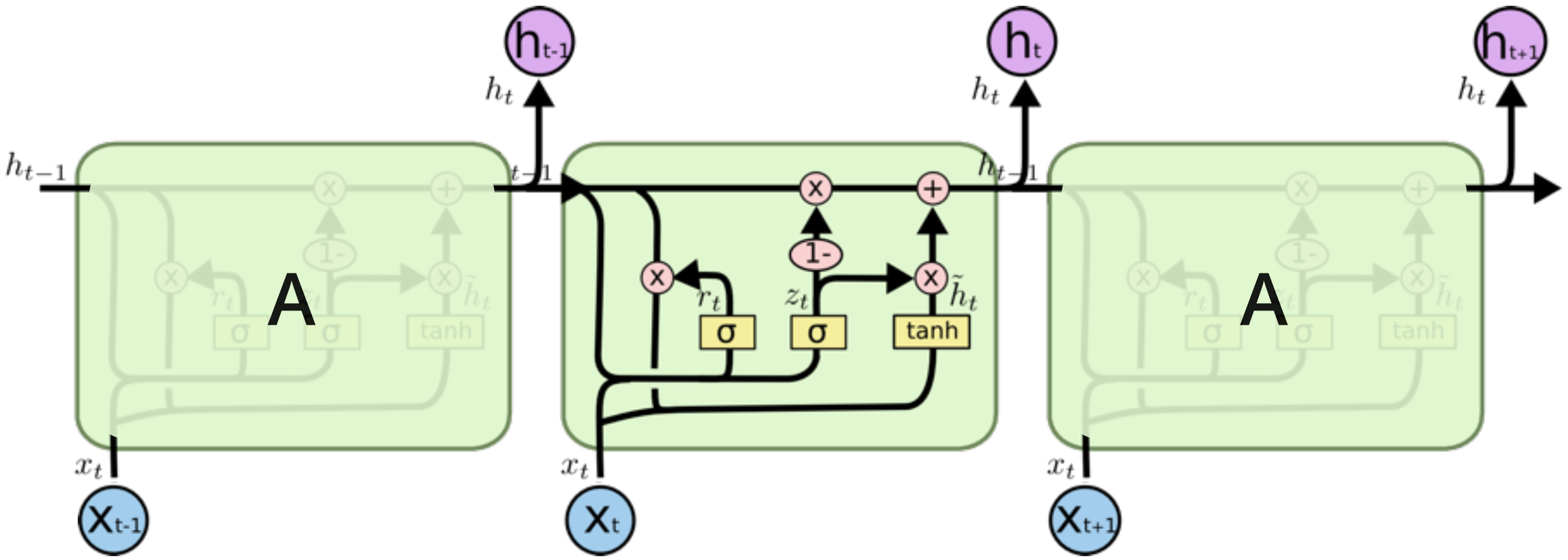
LSTMs

- Long Short Term Memory
- Special kind of RNN
- Better at understanding long-term dependencies than typical RNNs.

LSTMs : Simplified Explanation

- Decide what to keep from cell state (forget gate)
    - Decision made by Sigmoid layer
- Decide what to new information to store in cell state
    - Sigmoid layer (selects values to update)
    - Tanh layer (selects new values to replace old values)
- Multiply old state by forget gate (remove values)
- Add new information (add new values)
- Filter output based on cell state
    - Put cell state through another tanh layer and multiply by another sigmoid layer

GRU



source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

- Gated Recurrent Unit
- Variation of LSTM
- Combines forget and input gates; merges cell state and hidden state.
- Model is simpler than LSTM.

Demo TensorFlow

- This slide intentionally left blank.

TensorFlow vs. Torch

- LSTM Performance (CPU)
  - TensorFlow: ~0.30 sec/batch
  - Torch: ~0.21 sec/batch
  - 12000 batch LSTM trains 18 minutes faster in Torch

- GRU Performance (CPU)
  - TensorFlow: ~0.38 sec/batch
  - Torch: ~0.19 sec/batch
  - 12000 batch GRU trains 38 minutes faster in Torch

TensorFlow vs. Torch

- TensorFlow Pros:
  - Implementations for many different RNNs built-in.
  - Faster deployment
- TensorFlow Cons:
  - implementations for RNNs built-in (IDKWTFIAD)
  - Still needs OpenCL support

Questions?

Keith Davis
@keithdavisiii
IAmTheVastIdledHitchhiker.github.io