Machine Learning – Lecture 21

Wrapping Up

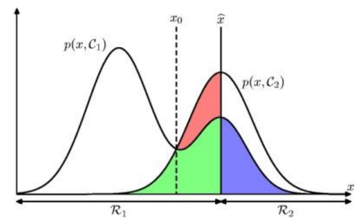
25.01.2018

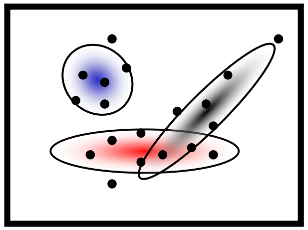
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

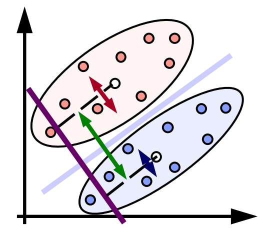
leibe@vision.rwth-aachen.de

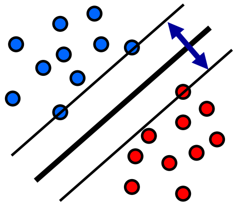
Course Outline

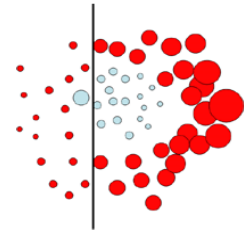
- Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
- Classification Approaches
  - Linear Discriminants
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Random Forests
- Deep Learning
  - Foundations
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Current Research Directions

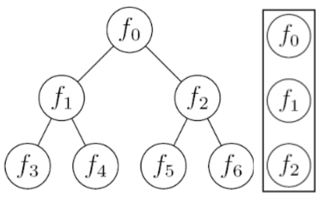


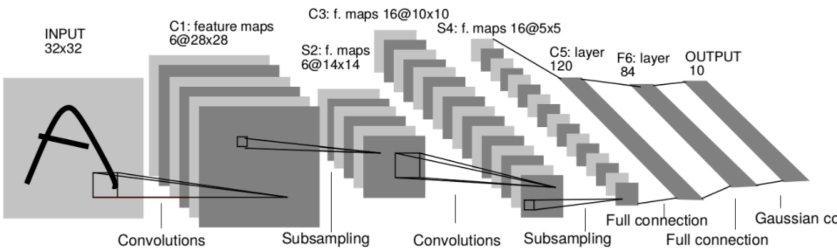




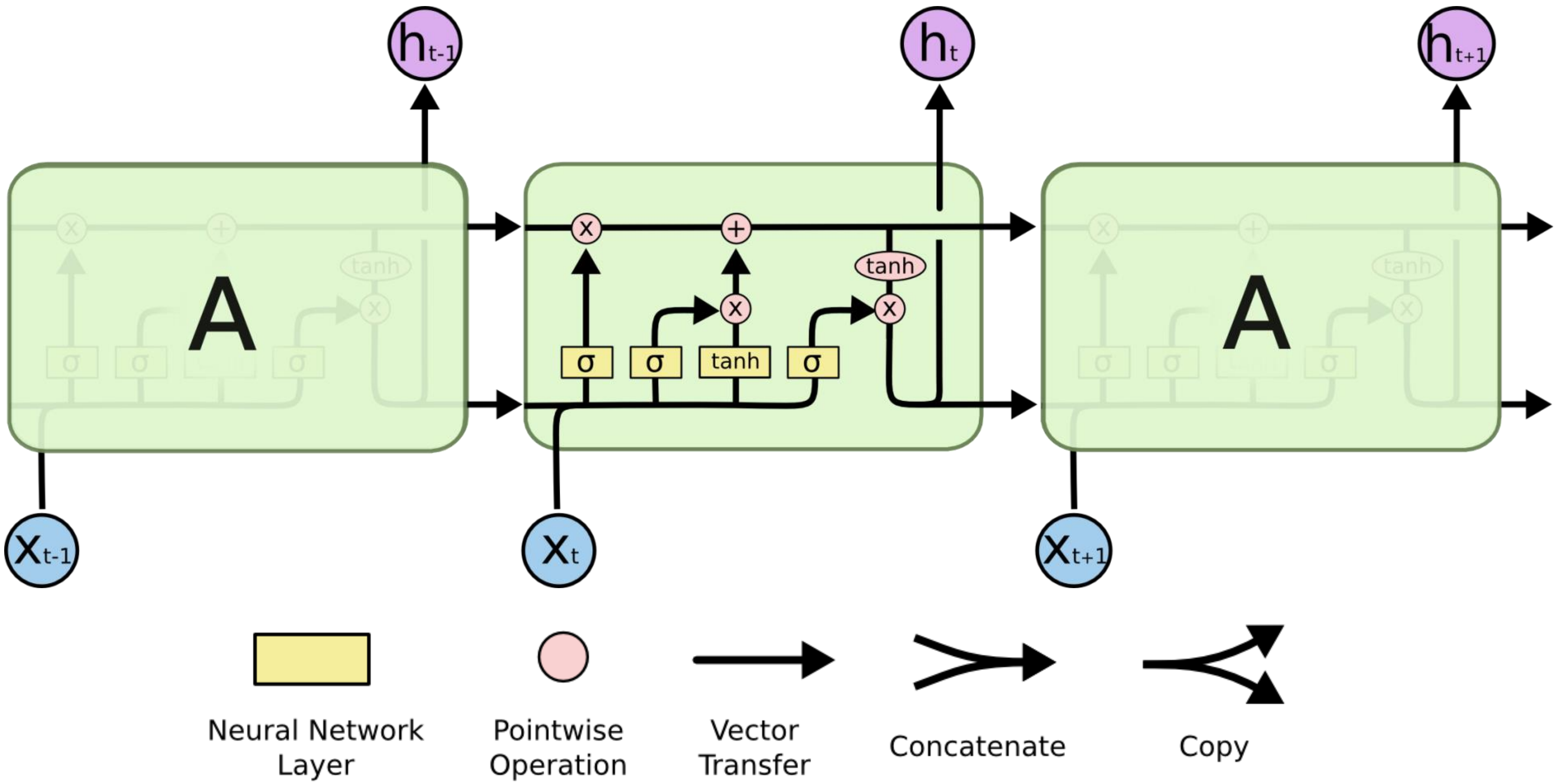








Recap: Long Short-Term Memory

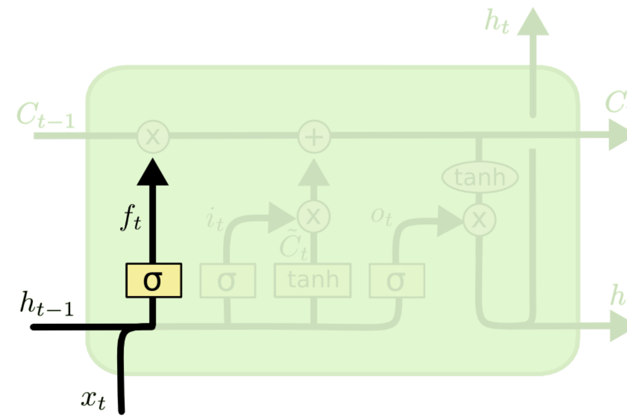


- LSTMs
  - Inspired by the design of memory cells
  - Each module has 4 layers, interacting in a special way.

Image source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

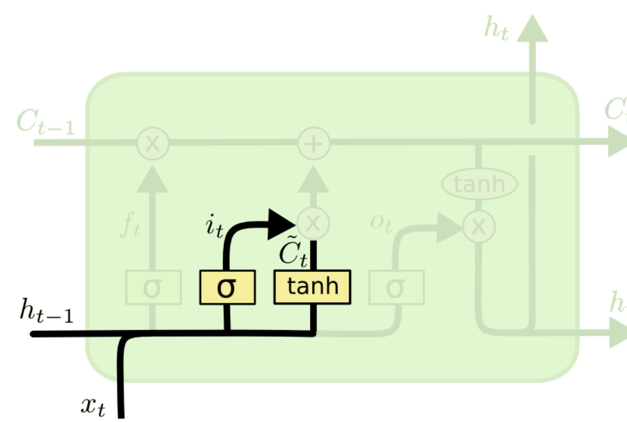
Recap: Elements of LSTMs

- Forget gate layer
  - Look at $\mathbf{h}_{t\text{-}1}$ and $\mathbf{x}_t$ and output a number between 0 and 1 for each dimension in the cell state $\mathbf{C}_{t\text{-}1}$.
    - 0: completely delete this,
    - 1: completely keep this.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- Update gate layer
  - Decide what information to store in the cell state.
  - Sigmoid network (input gate layer) decides which values are updated.
  - tanh layer creates a vector of new candidate values that could be added to the state.



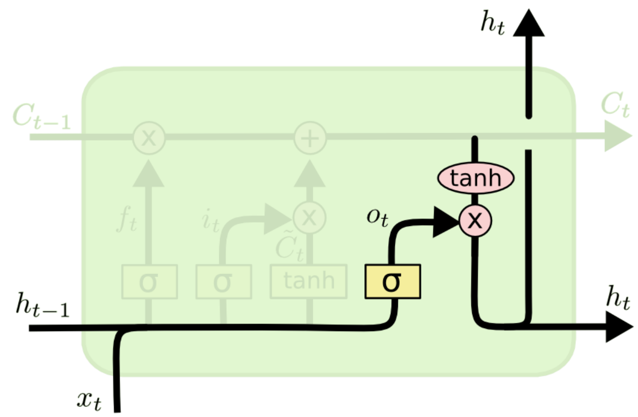
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recap: Elements of LSTMs

- Output gate layer
  - Output is a filtered version of our gate state.
  - First, apply sigmoid layer to decide what parts of the cell state to output.
  - Then, pass the cell state through a tanh (to push the values to be between -1 and 1) and multiply it with the output of the sigmoid gate.

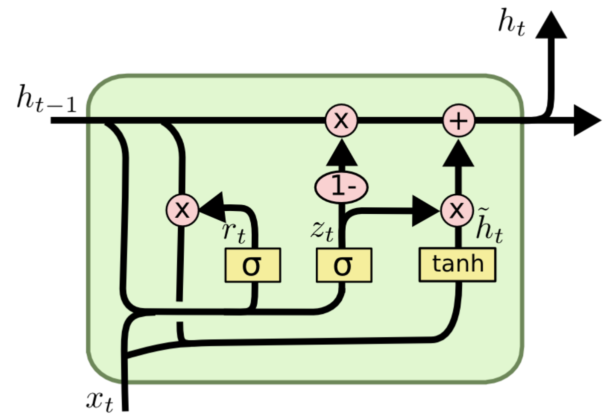


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Source: Christopher **Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/**

Recap: Gated Recurrent Units (GRU)

- Simpler model than LSTM
  - Combines the forget and input gates into a single update gate z_t.
  - Similar definition for a reset gate r_t, but with different weights.
  - In both cases, merge the cell state and hidden state.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

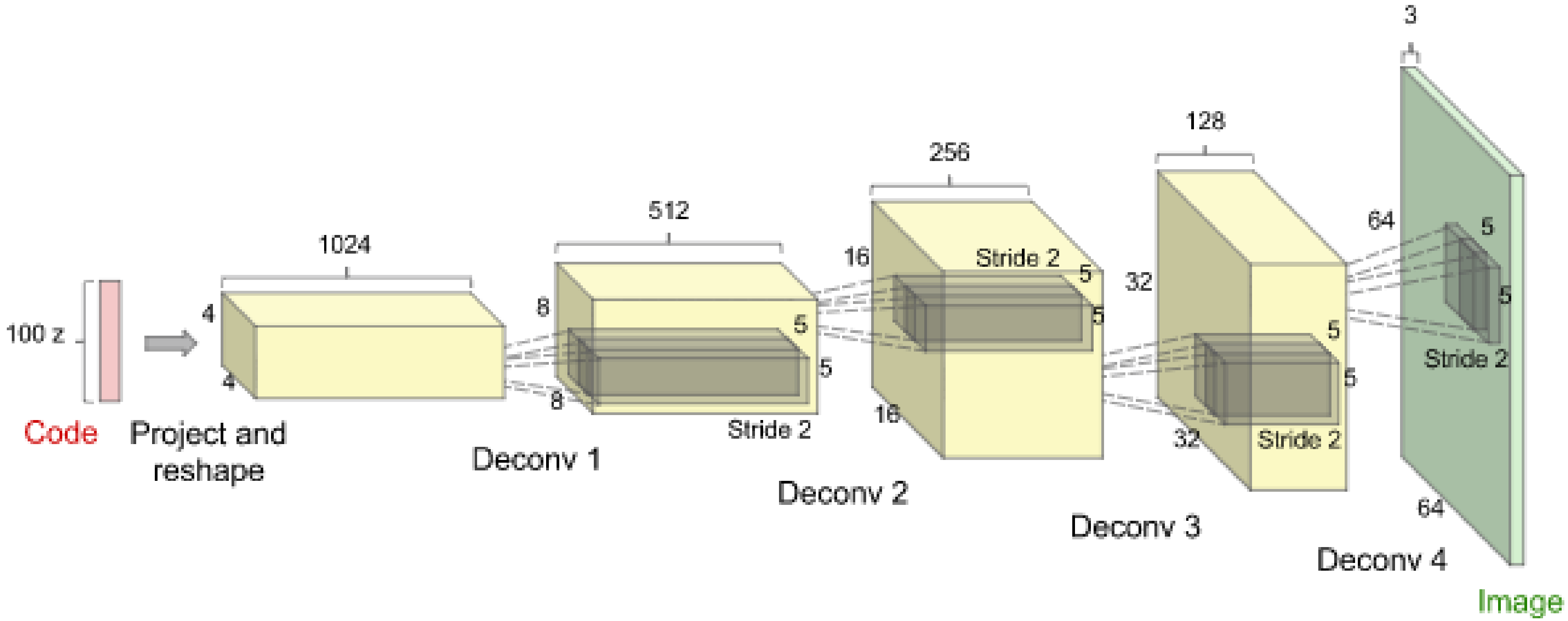
- Empirical results
  - Both LSTM and GRU can learn much longer-term dependencies than regular RNNs
  - GRU performance similar to LSTM (no clear winner yet), but fewer parameters.

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Currently Hot Research Directions

- Generative Models
  - Networks for image generation
  - Generative Adversarial Networks (GAN)
- Towards General Models of Computation
  - Memory Networks
  - Neural Turing Machines
- Deep Reinforcement Learning

Generative Networks

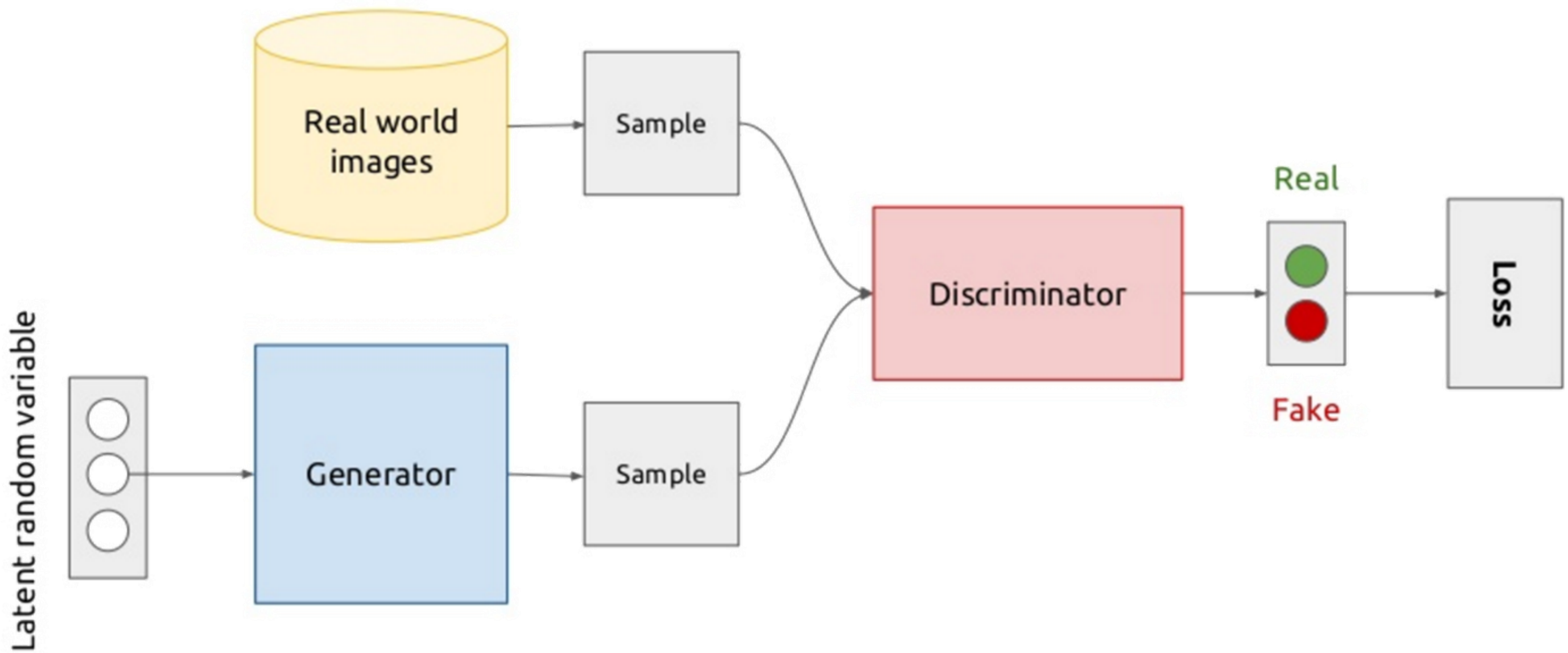


- Using a network to generate images
  - Sampling from noise distribution
  - Sequence of upsampling layers to generate an output image
  - *How can we train such a model to produce the desired output?*

Image from https://blog.openai.com/generative-models/

Generative Adversarial Networks (GAN)

- Conceptual view



- Main idea
  - Simultaneously train an image generator and a discriminator.
  - Interpreted as a two-player game
  - Very tricky to train…

Currently Hot Research Directions

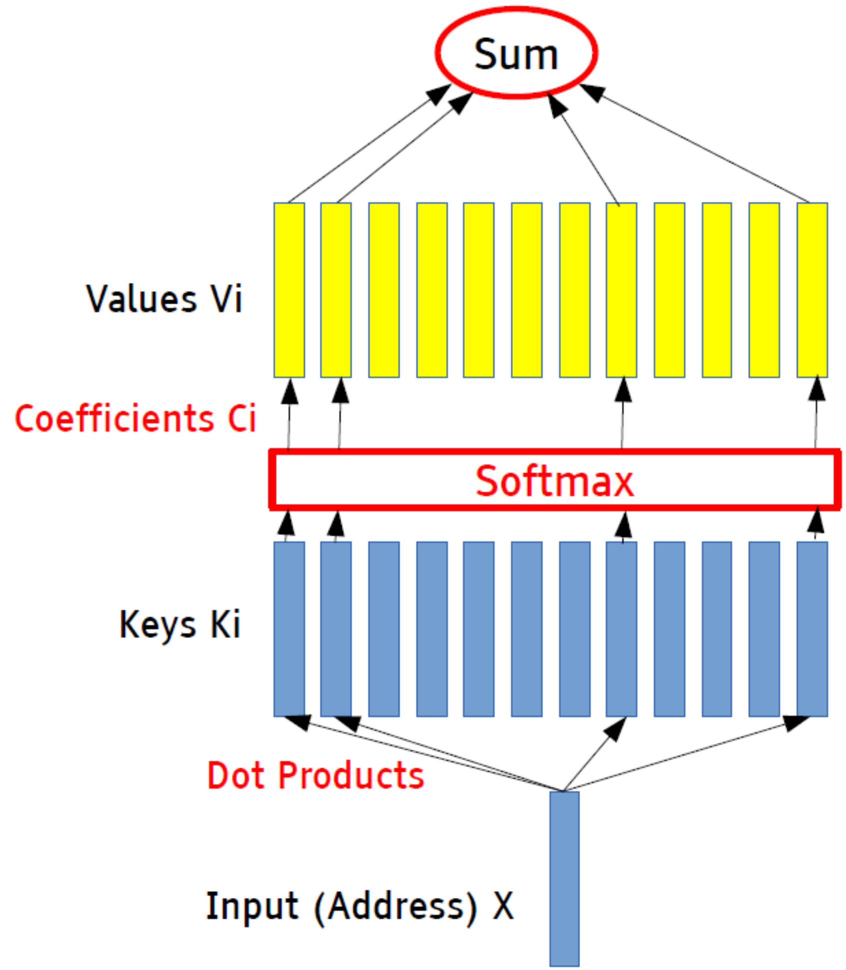
- Generative Models
  - Networks for image generation
  - Generative Adversarial Networks (GAN)
- Towards General Models of Computation
  - Memory Networks
  - Neural Turing Machines
- Deep Reinforcement Learning

Memory Networks

- Soft, differentiable memory
  - Stores <key, value> pairs
  - Input is matched to the stored keys
  - Output is the average over all values that correspond to the matched keys

- Key Idea
  - Make all steps differentiable.

  ⇒ Then all parameters (including access keys, stored values, etc.) can be learned with end-to-end supervised learning.

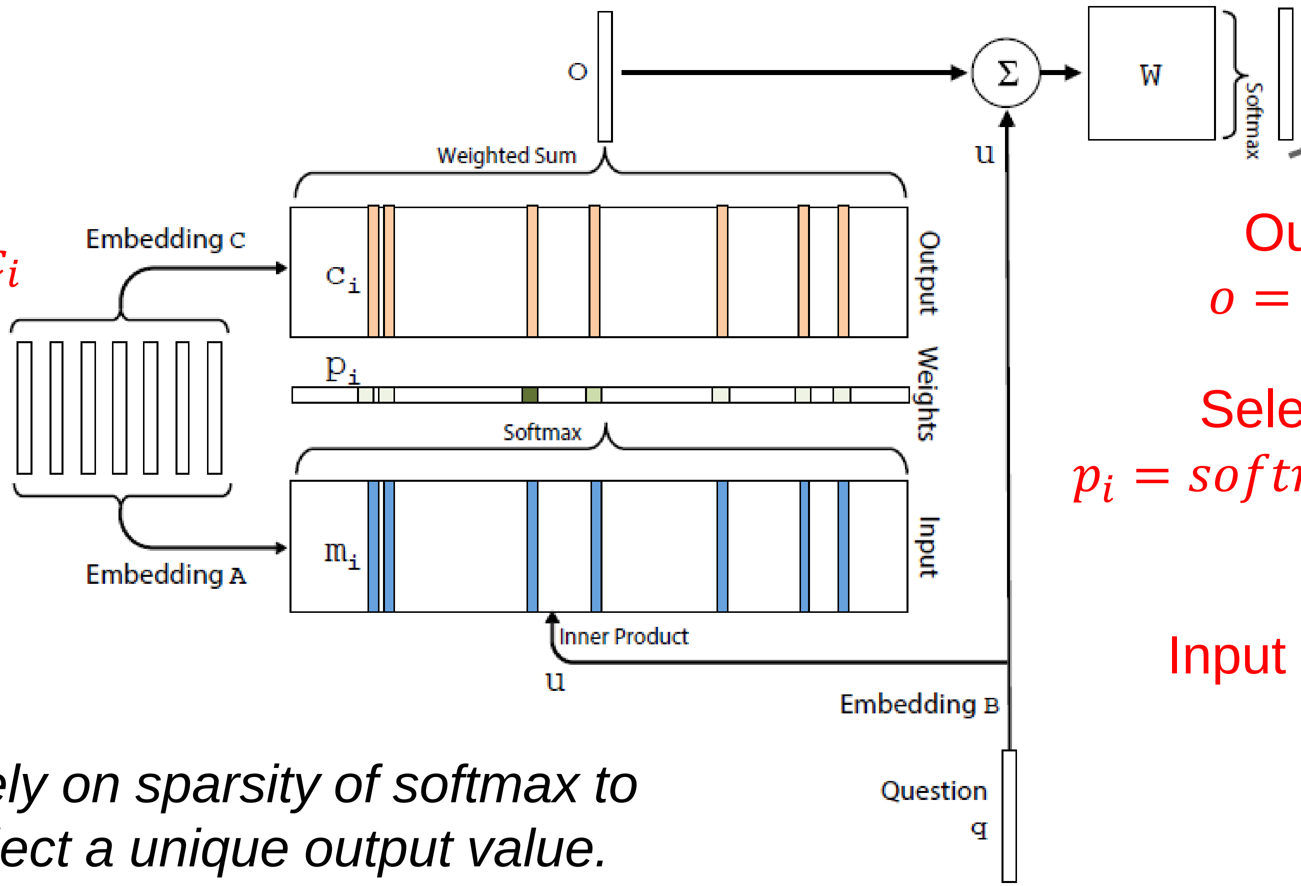


End-to-End Memory Networks

- A closer look at the memory mechanism

Values c_i

Keys m_i



Output
$o = \sum_i p_i c_i$

Selection
$p_i = softmax(u^T m_i)$

Input query u

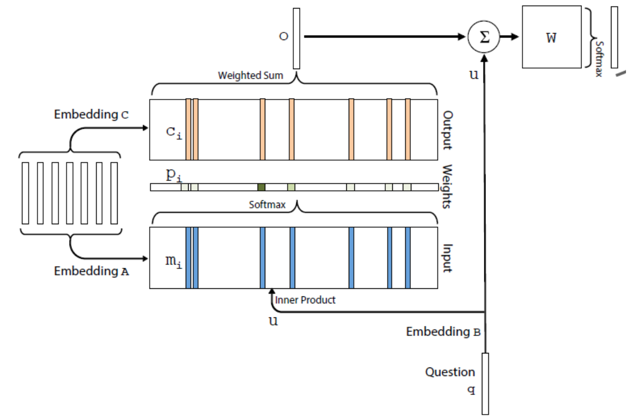
$\Rightarrow$ *Rely on sparsity of softmax to select a unique output value.*

S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, [End-to-End Memory Networks](). In NIPS 2015.

Image from [Sukhbaatar et al., 2015]

Memory Networks

- Problem with this design
  - Softmax used for the selection involves a normalization over all stored keys.
  - Memory cells that are not accessed get almost zero gradient.
  - When a backpropagation step causes the accessed memory cell to change, this massively affects the gradient flow.



Output
$o = \sum_i p_i c_i$

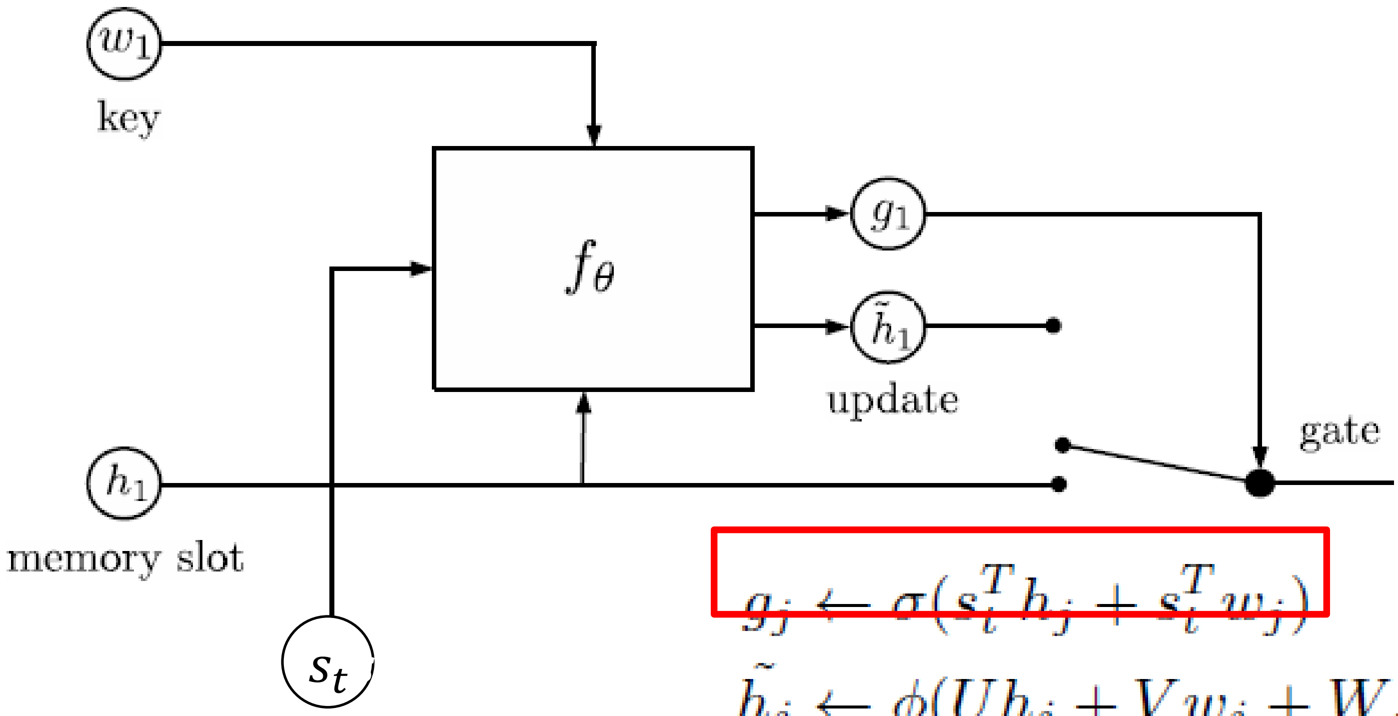
Selection
$p_i = softmax(u^T m_i)$

$\Rightarrow$ Together, this results in bad gradient propagation during learning.

$\Rightarrow$ Very finicky behavior...

Improved Design

- Gated memory (e.g., Recurrent Entity Network)



Gating mechanism

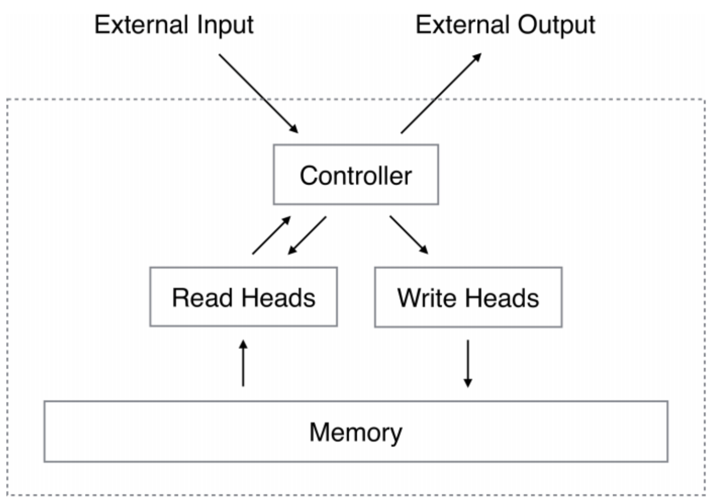
$$\boxed{g_j \leftarrow \sigma(s_t^T h_j + s_t^T w_j)}$$

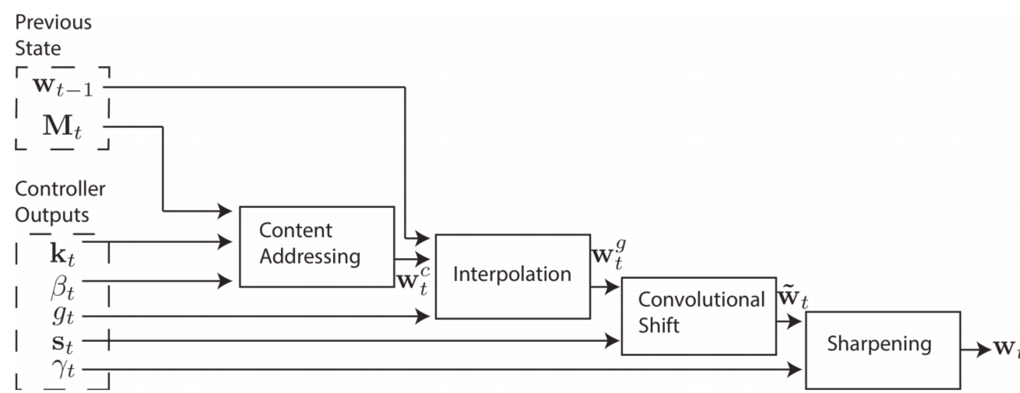
$$\tilde{h}_j \leftarrow \phi(U h_j + V w_j + W s_t)$$

$$h_j \leftarrow h_j + \boxed{g_j \odot \tilde{h}_j} \qquad h_j \leftarrow \frac{h_j}{||h_j||}$$

M. Henaff, J. Weston, A. Szlam, A. Border, Y. LeCun, Tracking the World State with Recurrent Entity Networks. arXiv 1612.03969, 2016.

Neural Turing Machines





- Goal: Enable general computation with Neural Nets
  - Again key is to make all operations differentiable.
  - Memory + Access operators + Controller
  - Learn entire algorithms from examples.

A. Graves, G. Wayne, I. Danihelka, Neural Turing Machines. arXiv 1410.5401, 2014.

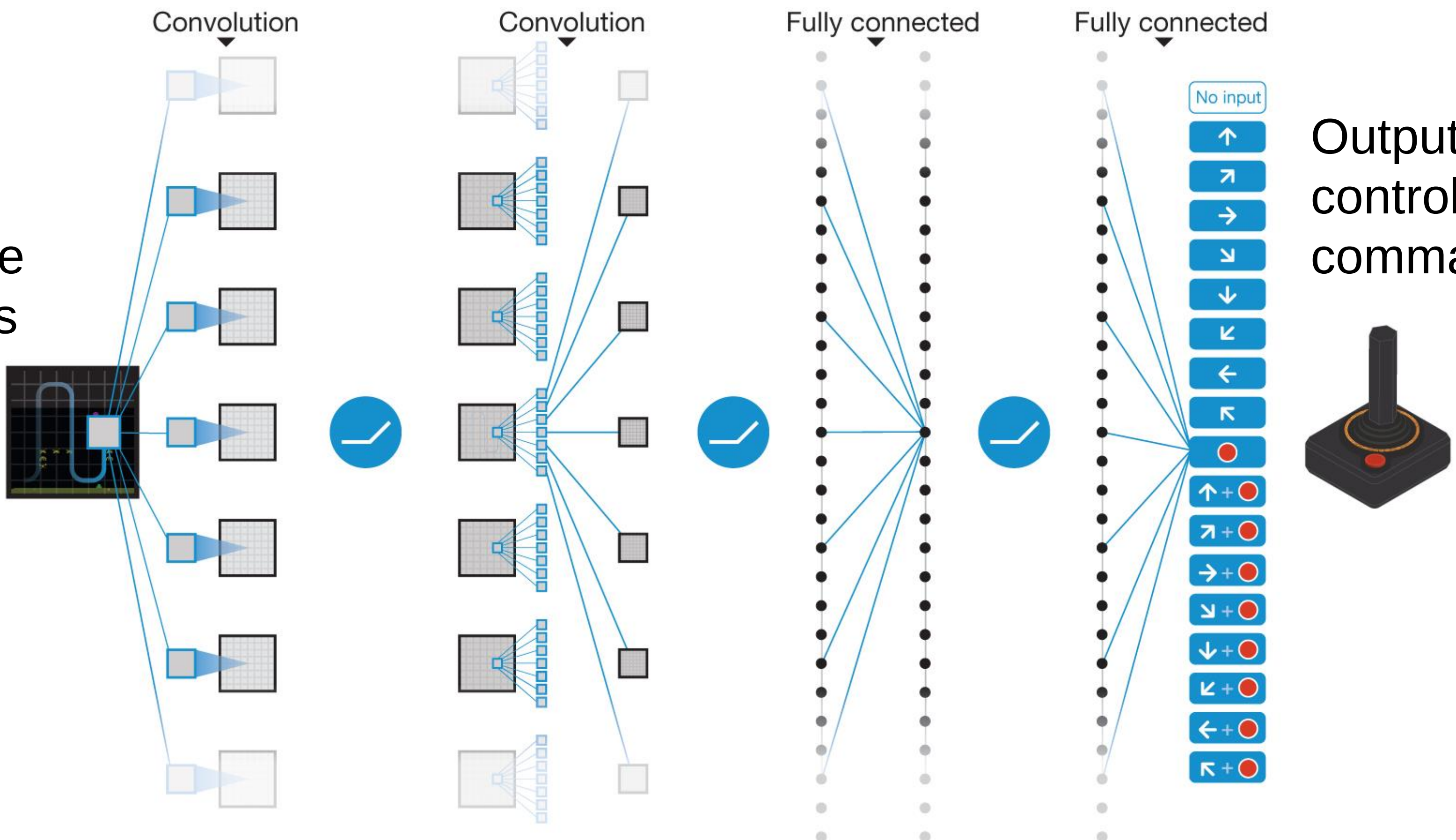
Currently Hot Research Directions

- Generative Models
  - Networks for image generation
  - Generative Adversarial Networks (GAN)
- Towards General Models of Computation
  - Memory Networks
  - Neural Turing Machines
- Deep Reinforcement Learning

Deep Reinforcement Learning

- Example application: Learning to play Atari games

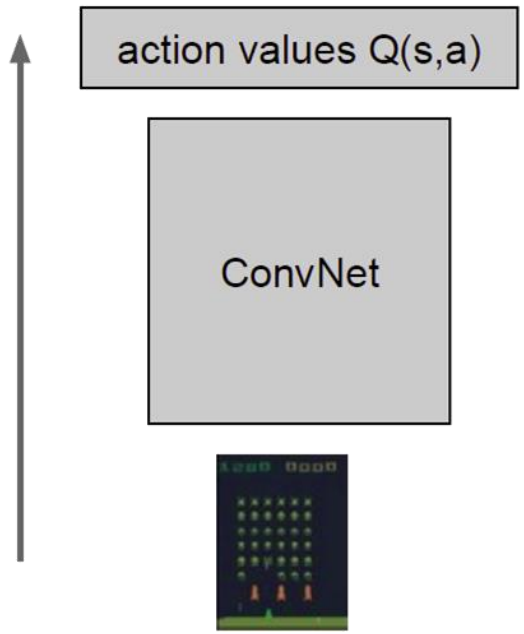
Input: pixels +game scores



Output: control commands

V. Mnih et al., [Human-level control through deep reinforcement learning](), Nature Vol. 518, pp. 529-533, 2015

Idea Behind the Model



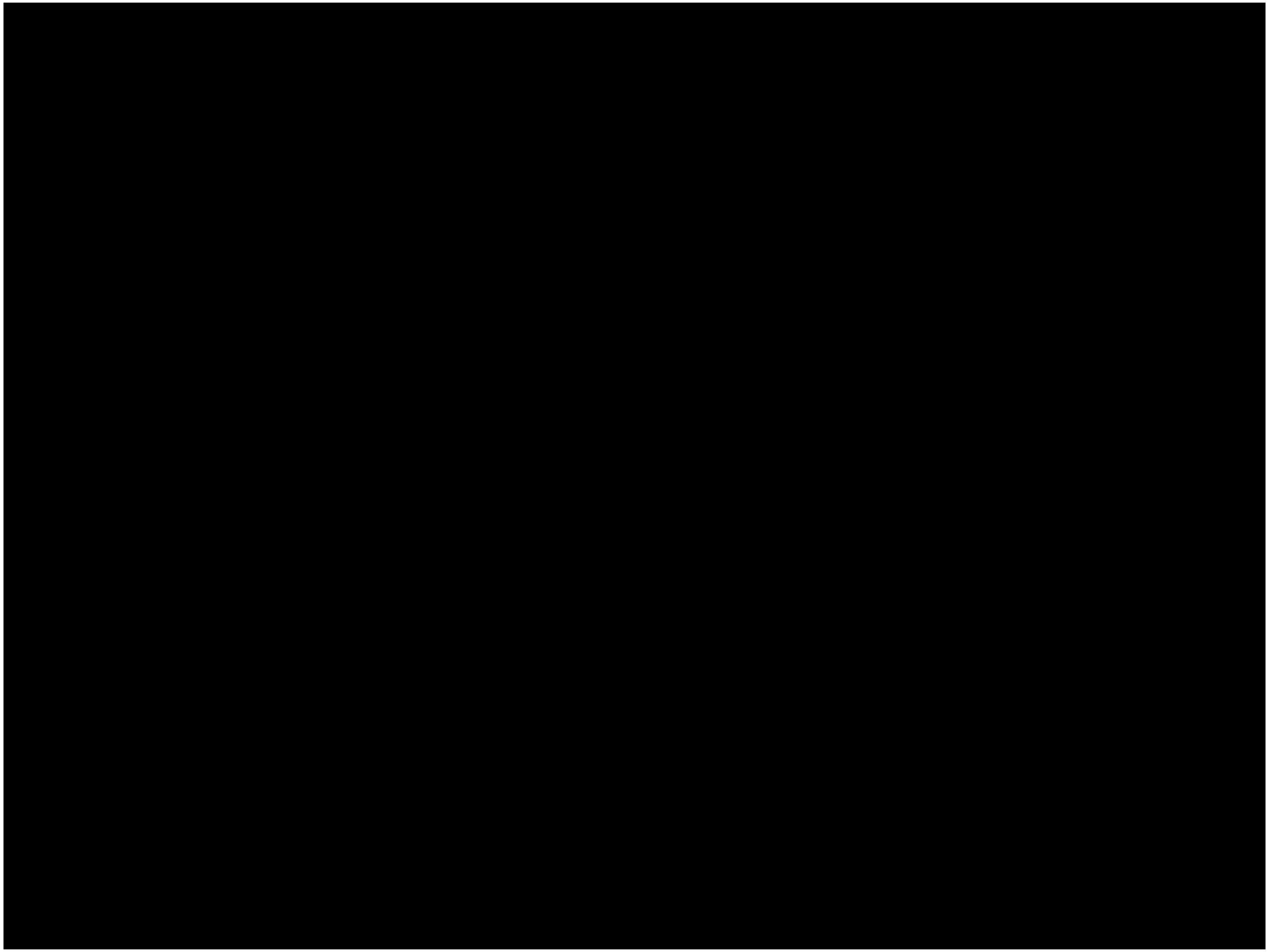
- Interpretation
  - Assume finite number of actions
  - Each number here is a real-valued quantity that represents the Q function in Reinforcement Learning
- Collect experience dataset:
  - Set of tuples {(s,a,s’,r), … }
  - (State, Action taken, New state, Reward received
- **L2 Regression Loss**

target value **predicted value**

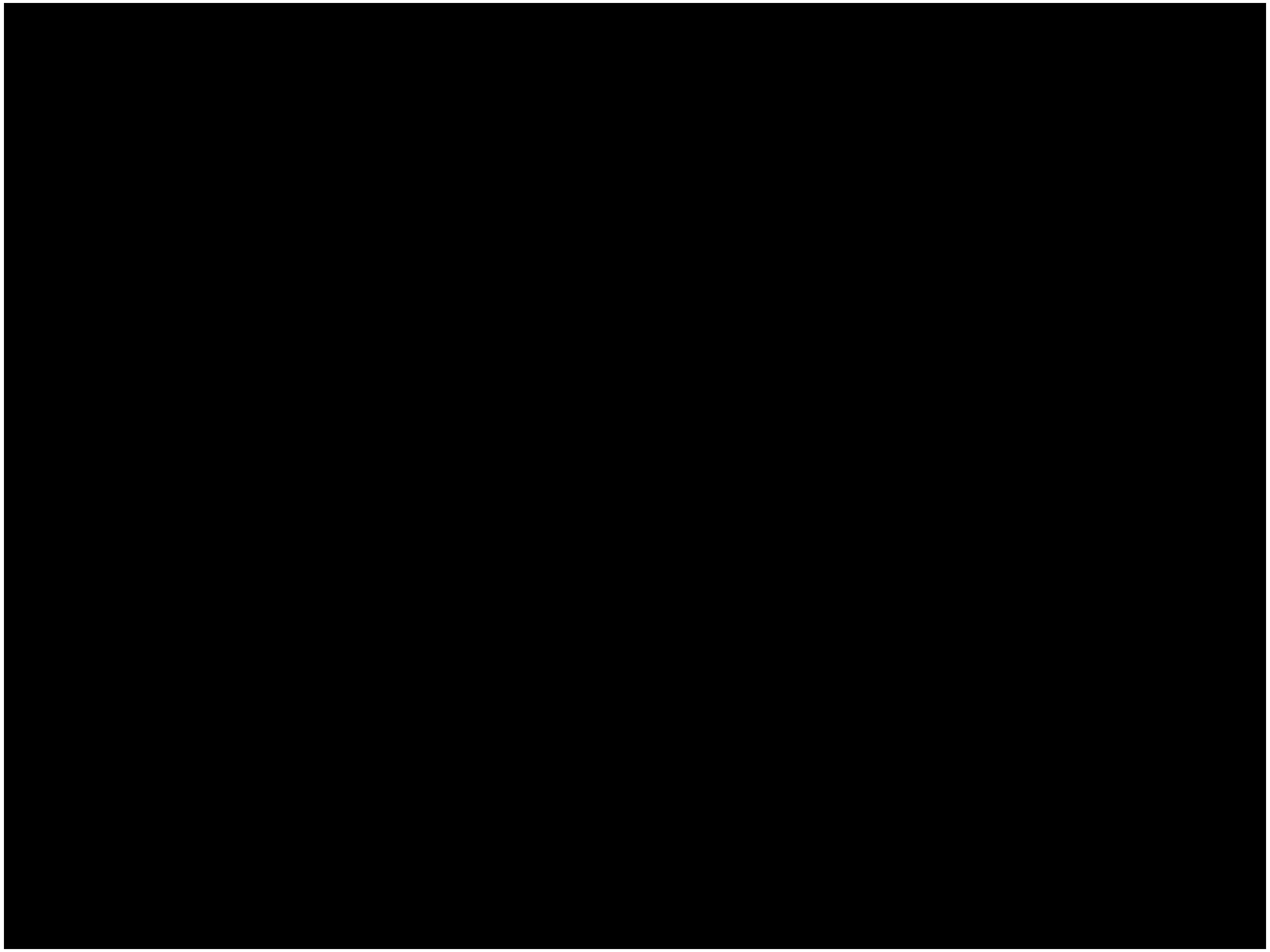
$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(\overbrace{\boxed{r + \gamma \max_{a'} Q(s',a';\theta_i^-)}}^{\text{target value}} - \overbrace{\boxed{Q(s,a;\theta_i)}}^{\text{predicted value}}\right)^2\right]$$

Current reward + estimate of future reward, discounted by γ

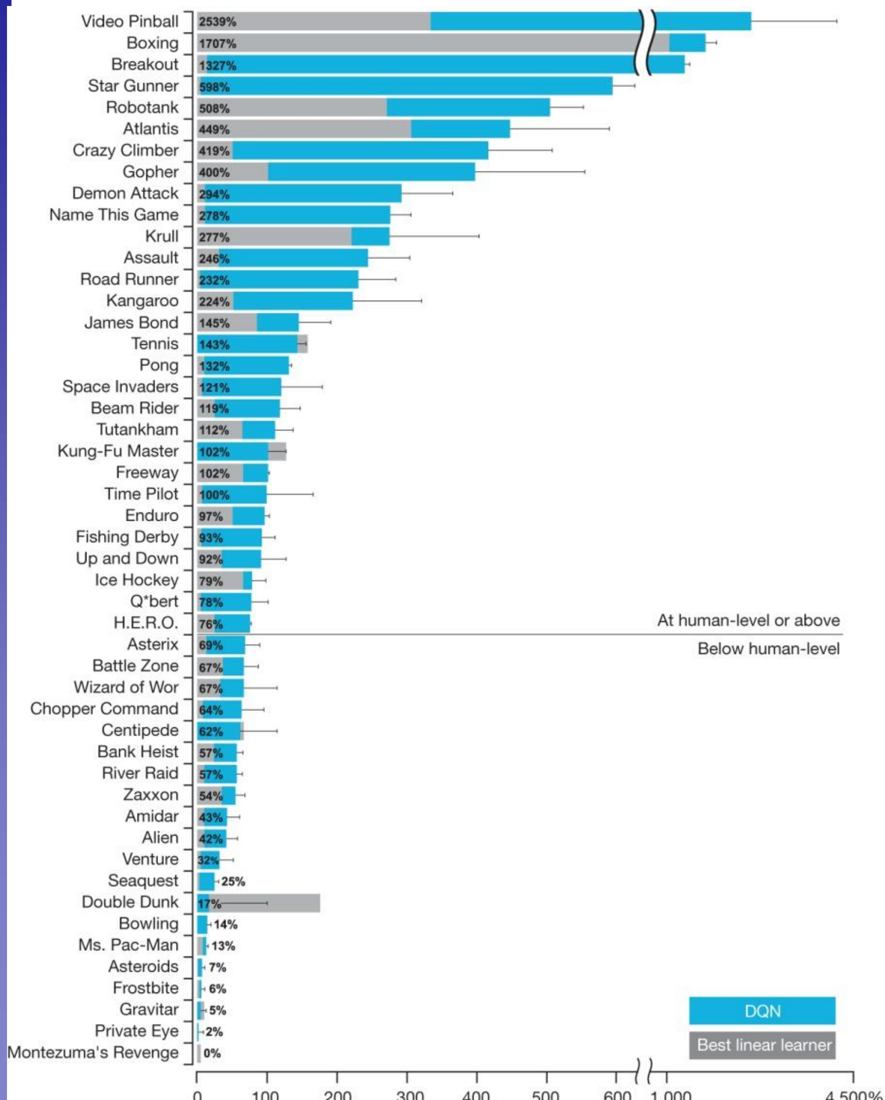
Results: Space Invaders

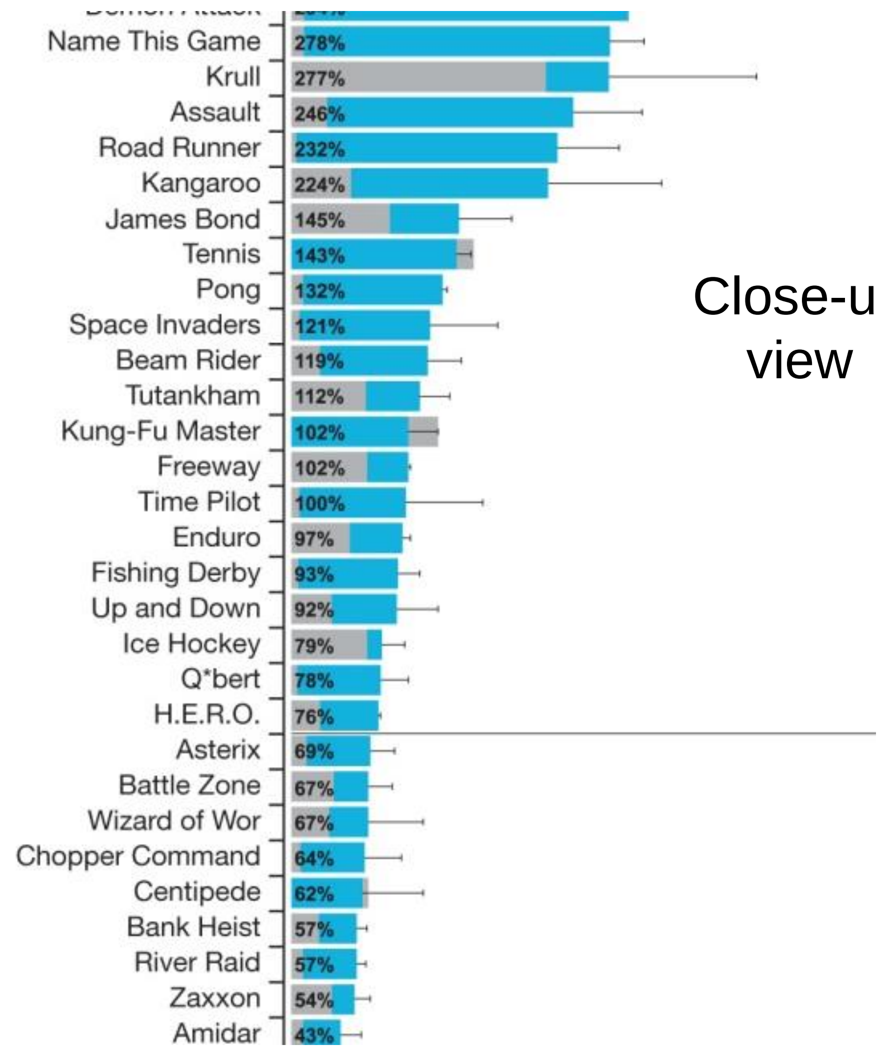


Results: Breakout



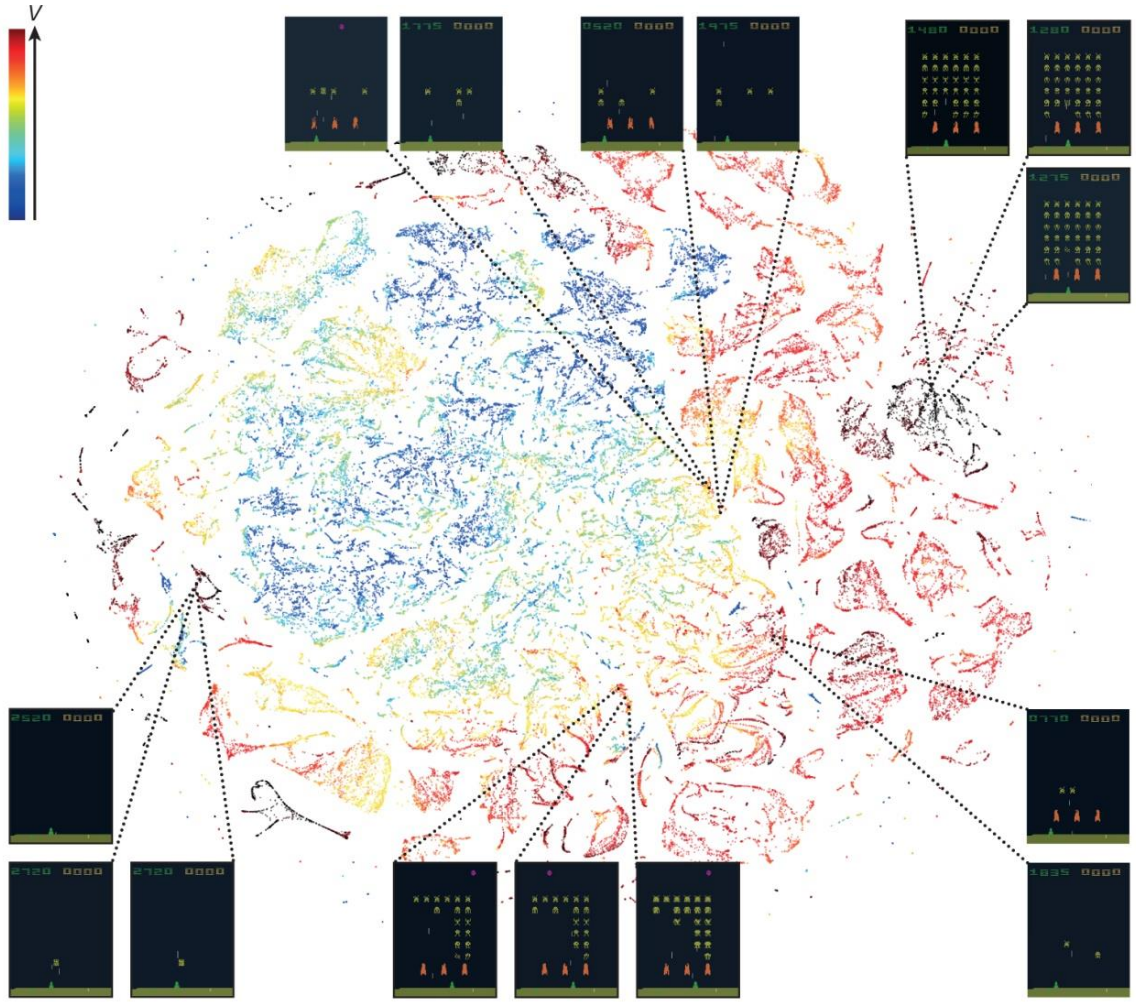
Comparison with Human Performance





Close-up view

Learned Representation



- t-SNE embedding of DQN last hidden layer (Space Inv.)

Success Story: Alpha Go

References and Further Reading

- Generative Adversarial Networks (GANs)
  - I.J. Goodfellow,J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, arXiv:1406.2661, 2014.
  - M. Arjovsky, S. Chintala, L. Boutou, Wasserstein GAN, arXiv:1701.07875, 2017.
  - L. Mescheder, P. Gehler, A. Geiger, The Numerics of GANs, arXiv:1705.10461, 2017.

References and Further Reading

- Memory Networks
  - S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, End-to-End Memory Networks. In NIPS 2015.
  - M. Henaff, J. Weston, A. Szlam, A. Border, Y. LeCun, Tracking the World State with Recurrent Entity Networks. arXiv 1612.03969, 2016.
- Neural Turing Machines
  - A. Graves, G. Wayne, I. Danihelka, Neural Turing Machines. arXiv 1410.5401, 2014.

References and Further Reading

- DQN paper
  - www.nature.com/articles/nature14236

- AlphaGo paper
  - www.nature.com/articles/nature16961