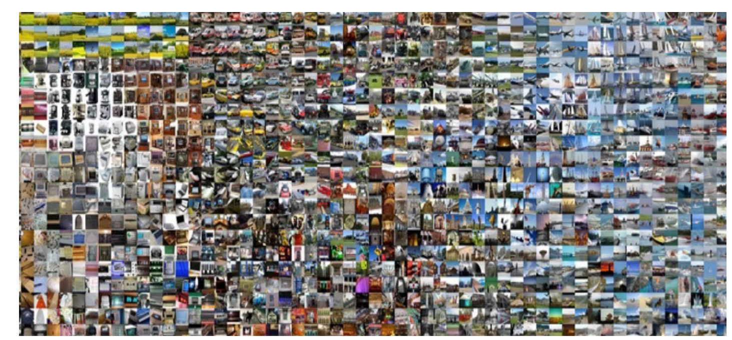
How SGD Can Succeed Despite Non-Convexity and Over-Parameterization

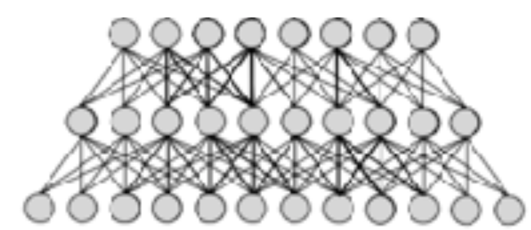
Amir Globerson, Tel Aviv University and Google

Deep Learning

- Highly expressive non-linear models.
- Standard usage protocol:
  - Collect large training sets



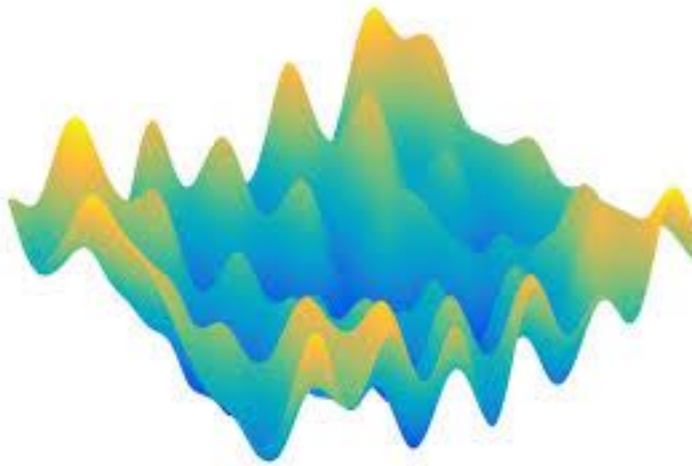
  - Design large models



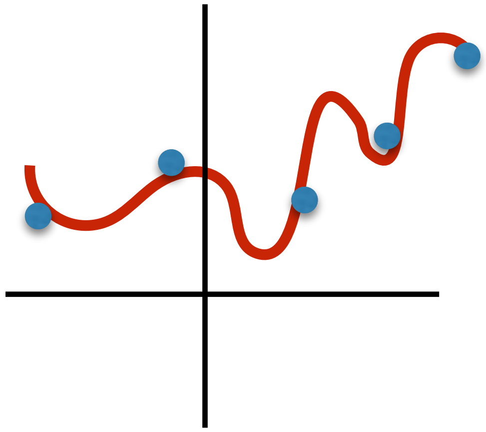
  - Train using SGD and GPUs

Key Problems

- **Non convexity**: high dimensional non-convex optimization. NP hard.



- **Overfitting**: Typically more parameters than data points. Overfits in the worst case (VC bounds).



- **Design**: Which architecture should we use for a given problem?

Outline

- A case where optimization is global (ICML 17)
- A case where large models generalize (ICLR 18)
- Design principles for architectures (ICML 18)

Globally Optimal ConvNets with Gaussian Inputs

ICML 17, with Alon Brutzkus

The Optimization Challenge

- Show a setting where GD/SGD successfully optimizes a **non-linear** **neural-net**

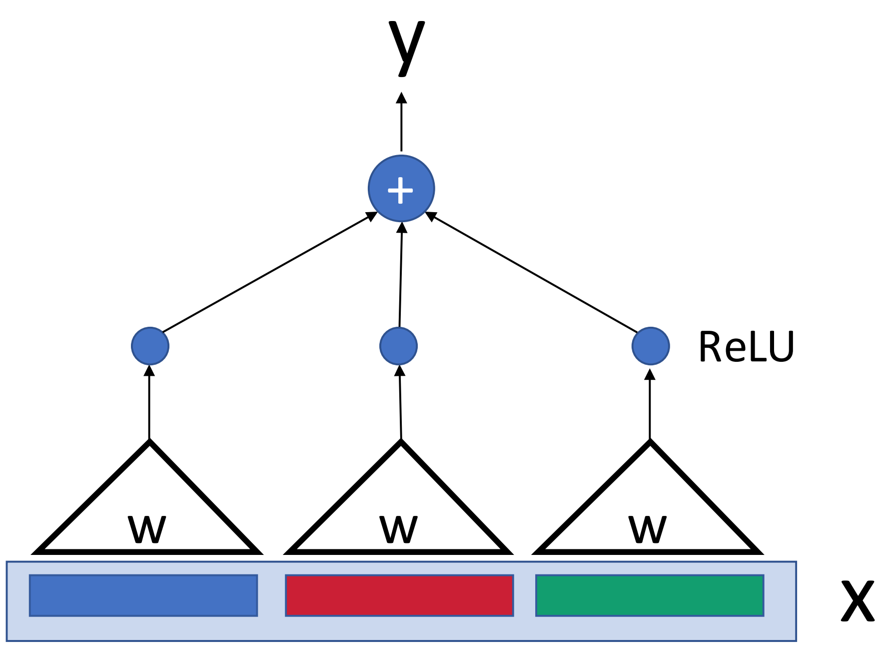
- Much progress on “linear” neural nets (Ma, Kawaguchi, Srebro and others)

- Nice early example: **Baum’s algorithm** for intersection of half spaces works for *symmetric distributions*.

Our Main Results

- A simple model of a convolutional layer, where:
  - Learning with arbitrary inputs is hard
  - Learning with Gaussian inputs is tractable using gradient descent.

The non overlap model



- Formally: $$f(\boldsymbol{x};\boldsymbol{w}) = \frac{1}{k}\sum_i \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}[i]\right)$$
- k is the number of hidden units.

The Learning Problem

- Consider the "realizable" case
- Input features **x** generated by some distribution ***D.***
- Output y produced using "true" weights $\boldsymbol{w}^*$
- Goal is to minimize squared loss:

$$\ell(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \left[\left(f(\boldsymbol{x}; \boldsymbol{w}) - f(\boldsymbol{x}; \boldsymbol{w}^*) \right)^2 \right]$$

Distributional Dependent Tractability

- Optimizing non-overlap models is worst case hard.
- Need to make assumptions on data generating distribution.
- Here we study the case where X_i correspond to independent standard normal variables.
- Denote this by G.

Gaussian Inputs

- A useful integral from Cho and Saul [2009]:

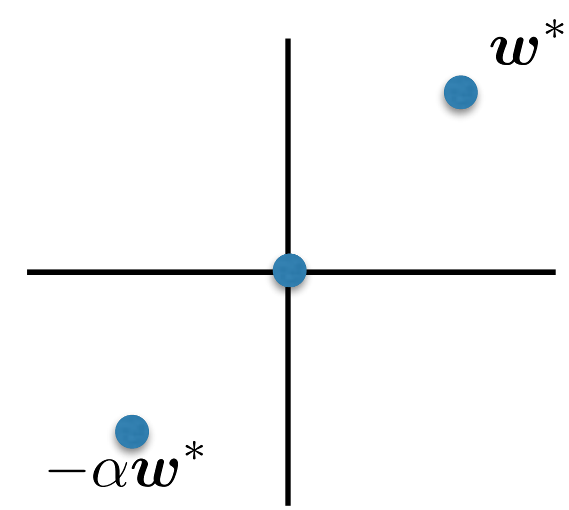
$$\mathbb{E}_{\mathcal{G}}\left[\sigma(\boldsymbol{u}\cdot\boldsymbol{x})\sigma(\boldsymbol{v}\cdot\boldsymbol{x})\right] = \frac{1}{2\pi}\left\|\boldsymbol{u}\right\|\left\|\boldsymbol{v}\right\|\left(\sin\theta_{\boldsymbol{u},\boldsymbol{v}} + \left(\pi - \theta_{\boldsymbol{u},\boldsymbol{v}}\right)\cos\theta_{\boldsymbol{u},\boldsymbol{v}}\right)$$

- Where $\theta_{\boldsymbol{u},\boldsymbol{v}}$ is the angle between the vectors.
- Denote this by g(**u**,**v**). Expected loss:

$$\begin{aligned}\ell(\boldsymbol{w}) &= \mathbb{E}_{\mathcal{G}}\left[\left(f(\boldsymbol{x};\boldsymbol{w}) - f(\boldsymbol{x};\boldsymbol{w}^*)\right)^2\right]\\ &= \frac{1}{k^2}\left[\gamma\|\boldsymbol{w}\|^2 - 2kg(\boldsymbol{w},\boldsymbol{w}^*) - 2\beta\left\|\boldsymbol{w}\right\|\left\|\boldsymbol{w}^*\right\|\right]\end{aligned}$$

Convergence of GD

- The Gaussian loss has the following critical points:
  - Non differentiable max at zero
  - Global min at $\boldsymbol{w}^*$
  - Degenerate saddle point at $-\alpha \boldsymbol{w}^*$



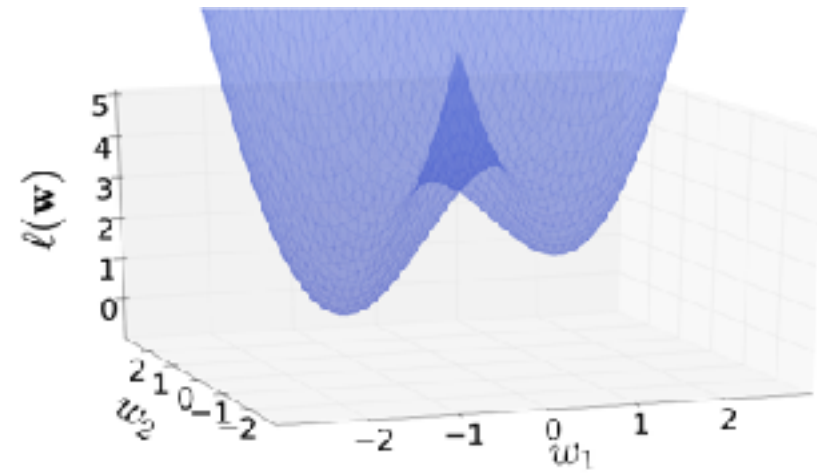
- GD will converge in $O(\epsilon^{-2})$

Networks with Overlap

- Can show that local minima emerge.

- But, random initialization seems to find global optimum with high probability

- Analysis left for future work.



Other Results for Gaussians

- Du et al., “Gradient Descent Learns One-hidden-layer CNN: Don’t be Afraid of Spurious Local Minima”, 2017. *[Training two layers]*
- Zhong et al., “Recovery Guarantees for One-hidden-layer Neural Networks”, 2017. [Fully connected]
- Ge et al., Learning One-hidden-layer Neural Networks with Landscape Design, 2017
- Safran & Shamir. “Spurious Local Minima are Common in Two-Layer ReLU Neural Networks”, 2017 [Local minima exist for fully connected]
- Goel & Klivans. “Eigenvalue Decay Implies Polynomial-Time Learnability for Neural Networks”, 2017 [Weaker distribution assumptions]
- Soltanolkotabi et al., Theoretical insights into the optimization landscape of over-parameterized shallow neural networks, 2017

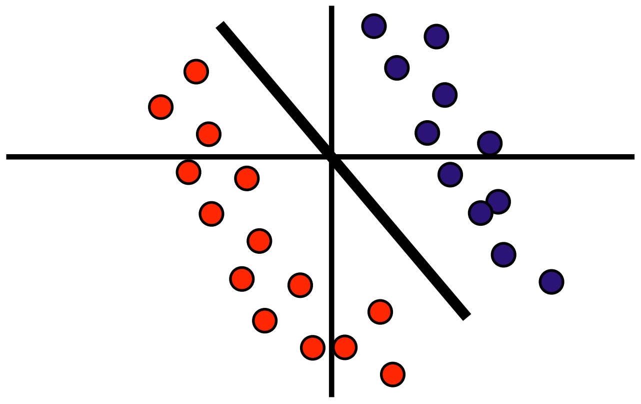
SGD Learns Over-parameterized Networks that Provably Generalize on Linearly Separable Data

ICLR 18, with Alon Brutzkus, Eran Malach and Shai Shalev-Shwartz

Optimization/Generalization Challenge

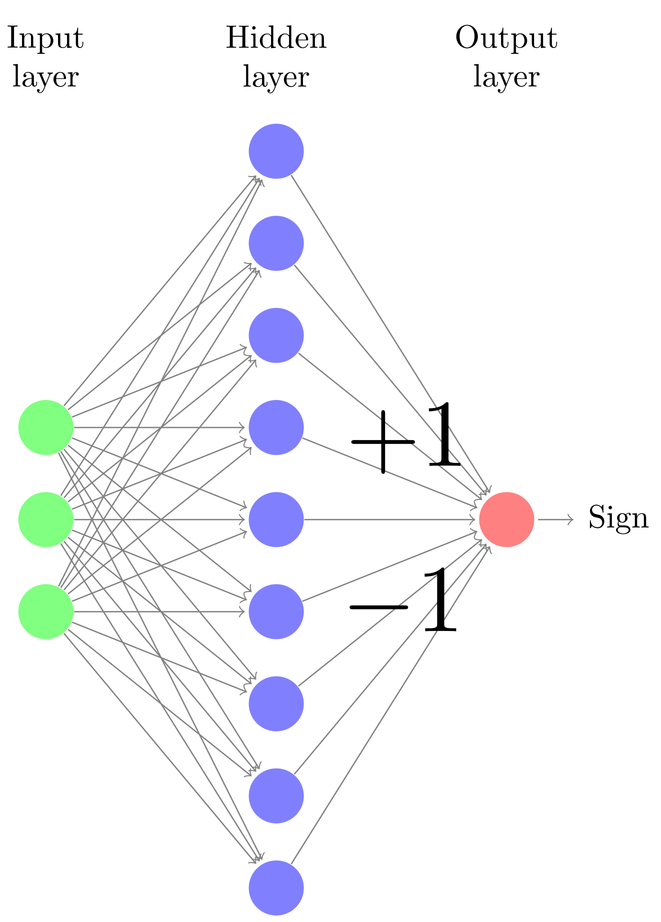
- Show case where SGD applied to a **large** neural net:
  - Optimizes successfully
  - Achieves low generalization error
- Several recent works show optimization for large networks (e.g., Soltanolkotabi, Soudry).
- But generalization not guaranteed.

The Linear Case



- Assume data is generated by some linear classification rule.

- No other distributional assumptions.



- Model is a one hidden-layer network, with **leaky ReLU** activation, and 2k hidden neurons.

- Assume only first layer is trained, and second layer is fixed to {-1,1} weights.

Understanding the Linear Case

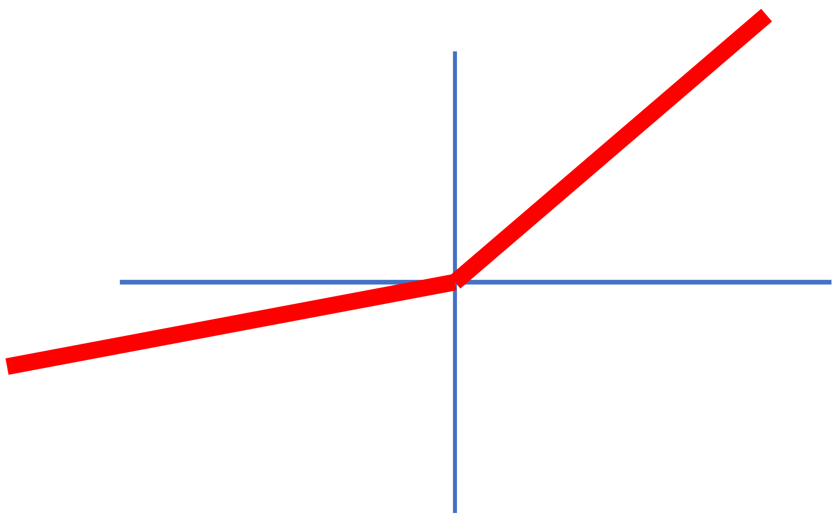
- **Perceptron** solves it!
- But can SGD on neural net find a solution?
- If SGD finds a good solution, will it generalize well?
- Prior work analyzes optimization landscape, but does not derive such results (*e.g., Auer et al. 96, Gori & Tesi 92, Frasconi 97, Nguyen & Hein 17*).

Our Results

- **Optimization**: SGD finds a zero error solution.
- **Generalization**: Guaranteed independent of k.
- **ReLU**: Can fail.

Leaky ReLU

- Activation: $\sigma(\boldsymbol{x}) = \max(\boldsymbol{x}, \alpha\boldsymbol{x})$
- Avoids the “Dead ReLU” problem
- Needed in our analysis.



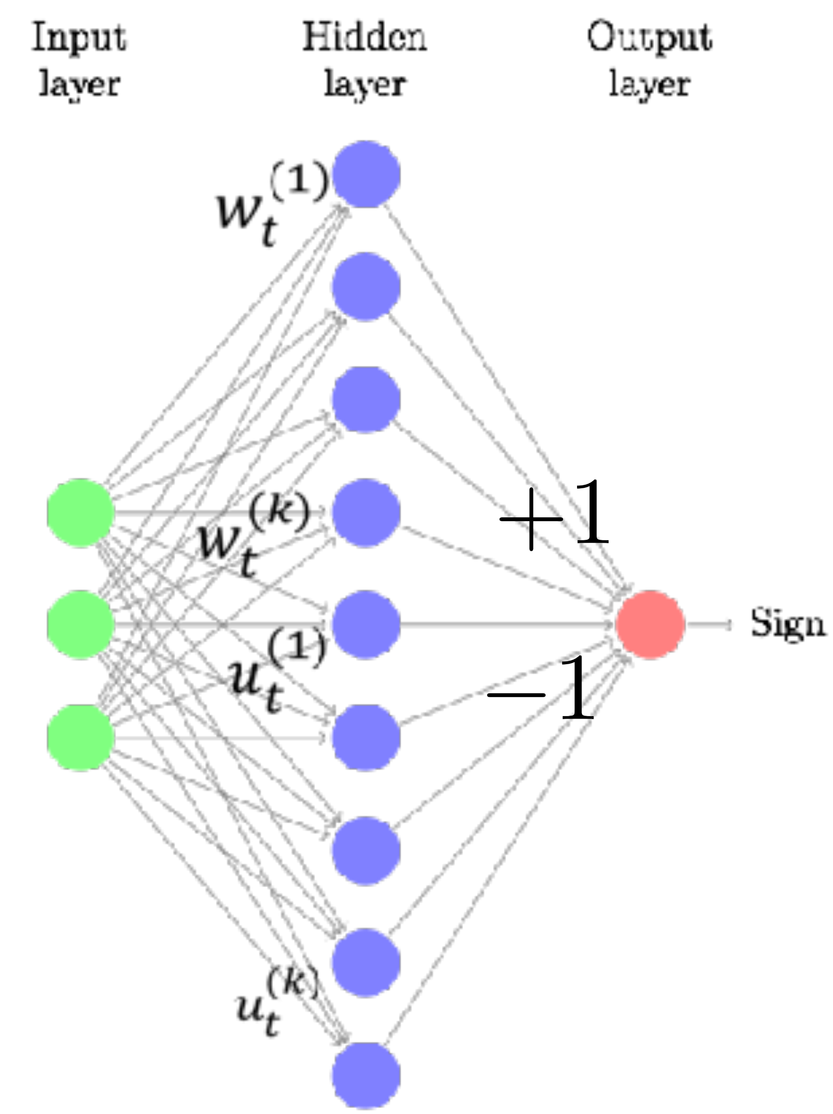
Notation

- Weights at time t:

$$W_t = [w_t^{(1)}, \ldots, w_t^{(k)}, u_t^{(1)}, \ldots, u_t^{(k)}]$$

- Network output:

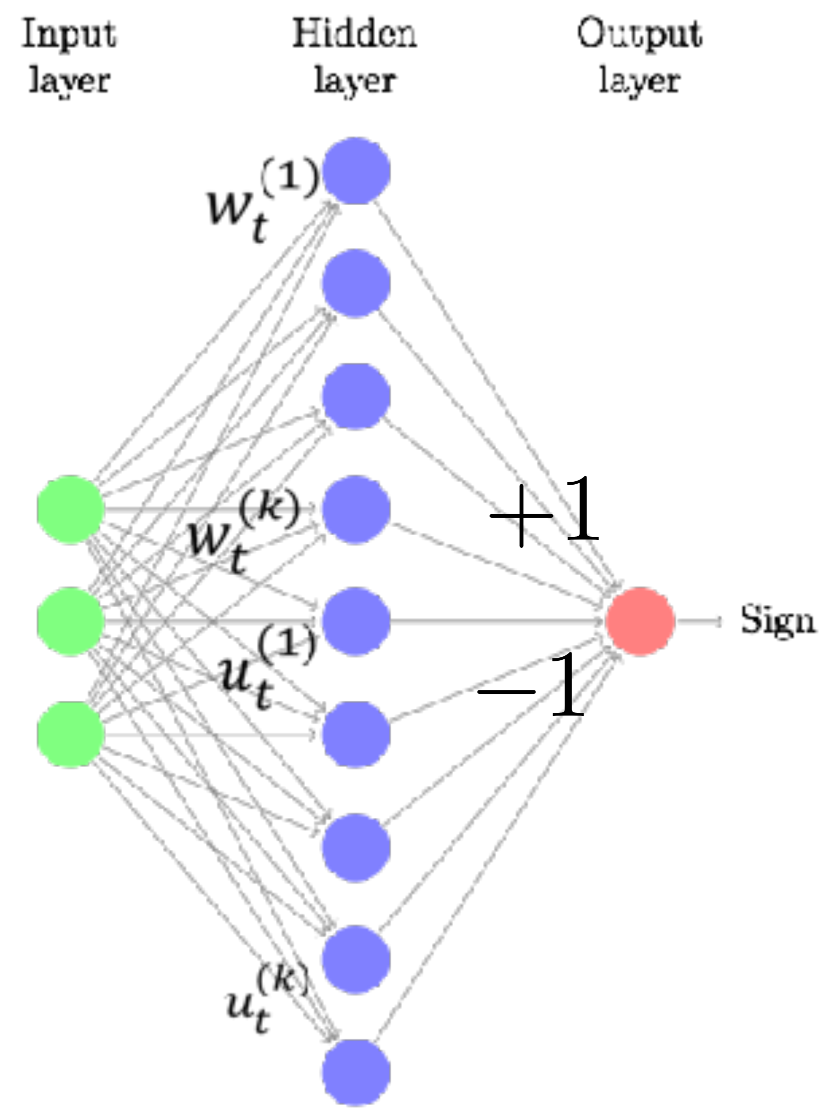
$$f(\boldsymbol{x}; W) = \sum_{i=1}^{k} \sigma(\boldsymbol{w}^{(i)} \cdot \boldsymbol{x}) - \sum_{i=1}^{k} \sigma(\boldsymbol{u}^{(i)} \cdot \boldsymbol{x})$$



Separability Assumption

- Assume $(\boldsymbol{x}, y) \sim \mathcal{D}$ where: $y\boldsymbol{w}^* \cdot \boldsymbol{x} \geq 1$
- So one ERM solution is (up to scale):

$$W^* = [\boldsymbol{w}^*, \ldots, \boldsymbol{w}^*, -\boldsymbol{w}^*, \ldots, -\boldsymbol{w}^*]$$



Algorithm

- Use simple SGD on hinge loss:

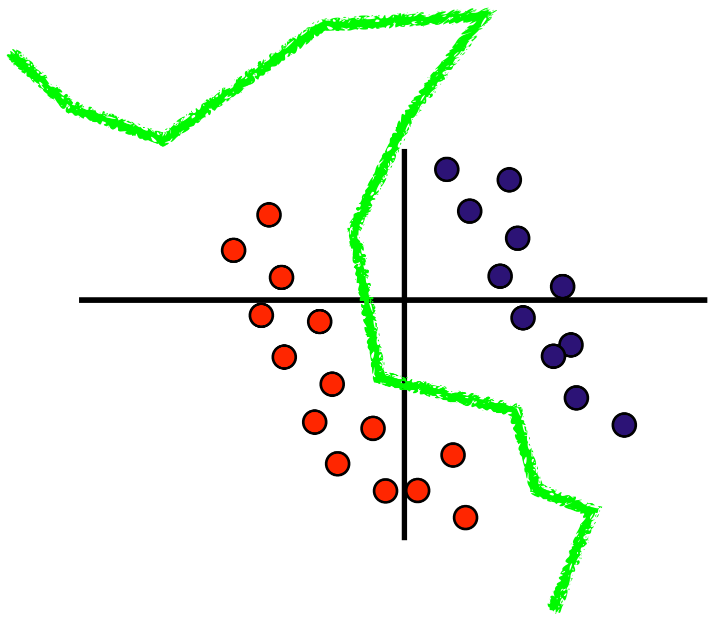
$$\ell(W) = \frac{1}{n}\sum_{i=1}^{n}[1 - y_i f(\boldsymbol{x}_i; W)]_+$$

- Fixed step size η

- Arbitrary initialization

Expressivity

- Consider kd>n case.
- Zero training error “almost” always possible (e.g., Soudry et al. 17).
- Hence overfitting is a real problem
- So perfect optimization does not guarantee good test error.



Optimization

- SGD converges to a global minimum after the following number of non-zero updates:

$$O\left(\frac{\|\boldsymbol{w}^*\|_2^2}{\alpha^2}\right)$$

Independent of k.

- Proof similar to perceptron:
  - Show $W_t \cdot W^*$ increases
  - Show $\|W_t\|_2^2$ does not increase by much.
  - Use Cauchy Schwartz to get bound on update, independent of number of hidden units.

Generalization

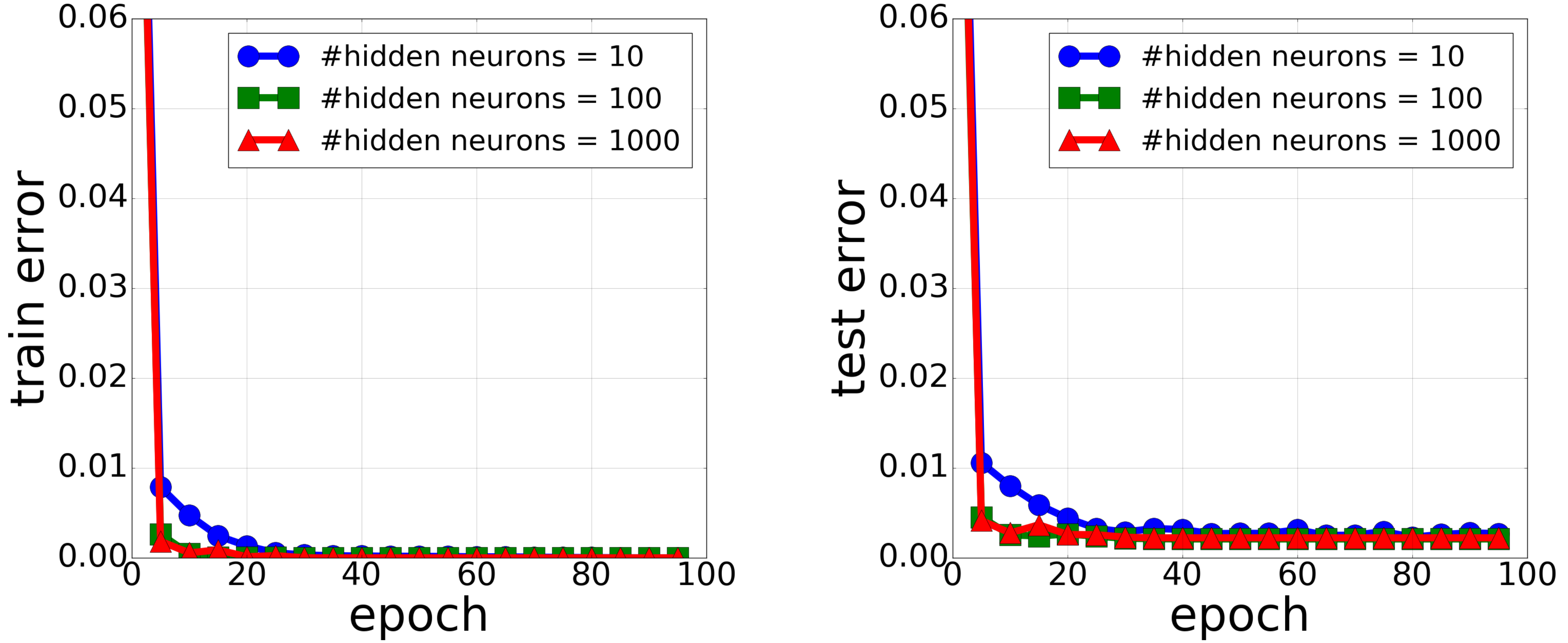
- Learned model only uses a fixed size subset of training data
- Can use a **compression bound** (Littlestone & Warmuth 86)
- Conclude that test error is: $O\left(\frac{\|\boldsymbol{w}^*\|_2^2}{n}\log\frac{n}{\delta}\right)$
- Independent of k

ReLU

- **There always exist bad local minima**
- Can construct cases where SGD fails
- **Positive:** can construct a case where for large enough network, SGD converges globally.
- **Idea:** large network overcomes dead ReLUs.

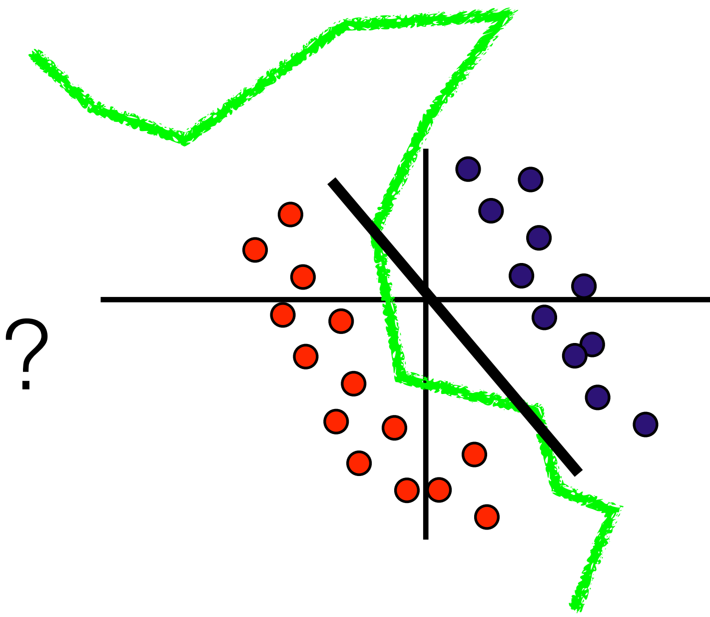
Experiments

- Binary MNIST with 4000.
- Over-parameterized regime shows no over-fitting.

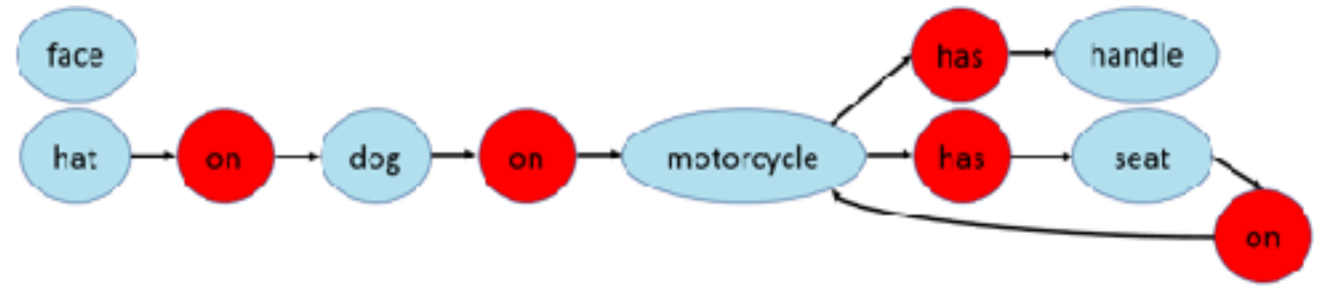


Thoughts on Inductive Bias

- Our results suggests that SGD “likes” linear nets.
- “All else being equal” SGD will converge to a linear rule (roughly)
- What is the right generalization:
    - Minimal rank/trace norm (Ma, Srebro) ?



Data from Visual Genome dataset

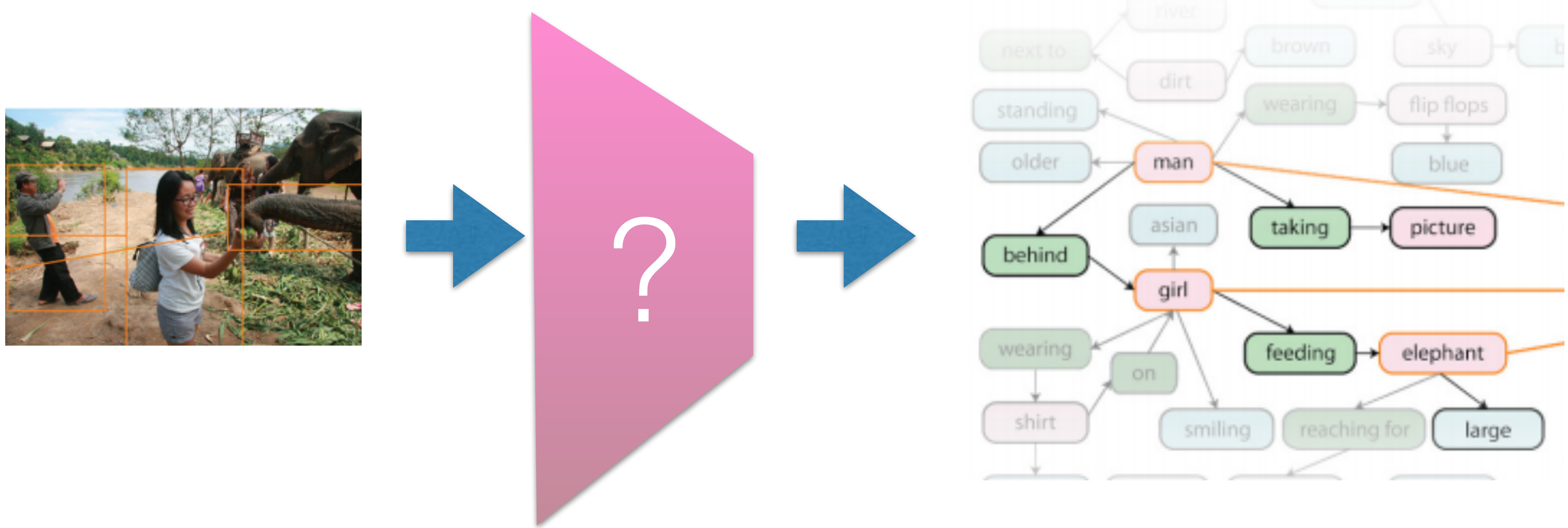


Network Design for Structured Prediction

with Roei Hertzig, Moshiko Raboh, Gal Chechik, Jonathan Berant and Nataly Brukhim

Choosing Architectures

- How can we choose a good neural architecture for a given task?



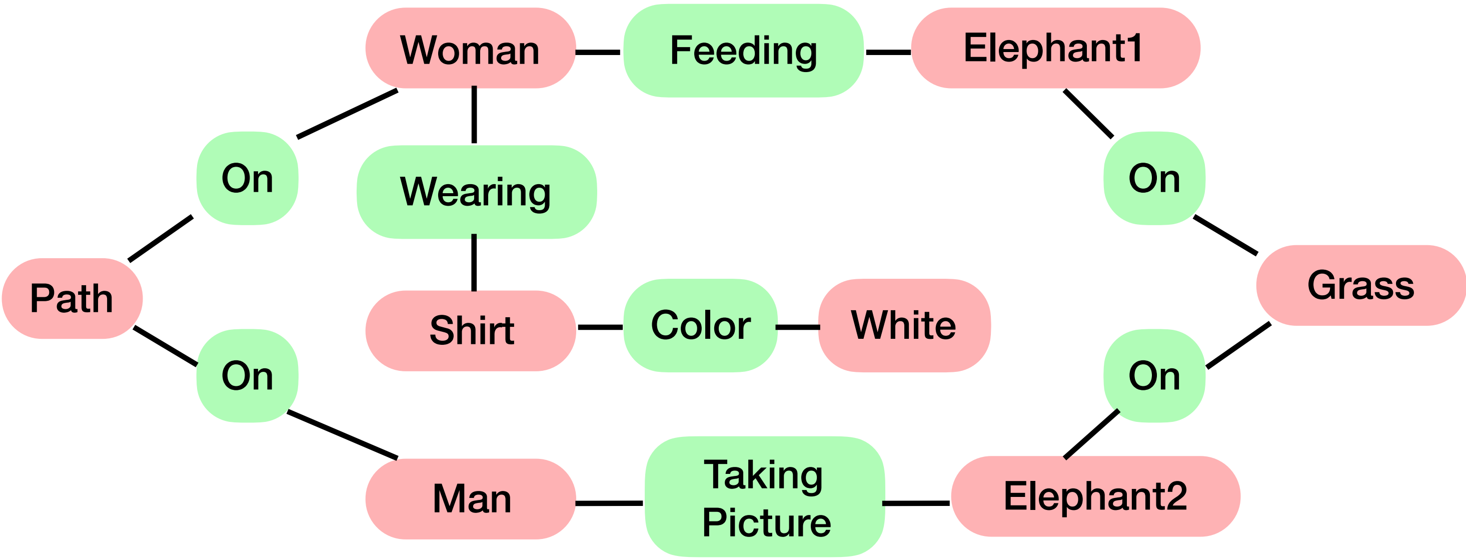
Invariance and Architectures

- Suppose you know that the true mapping has some invariances.

- Then it would be nice if the architecture has the same invariances.
- Otherwise we're just **wasting parameters**!
- Recently used in DeepSets (Zaheer et al. 17) but also discussed for graph based deep learning (Gilmer et al. 17, Dai et al. 17).
- How can this be applied to scene graphs?

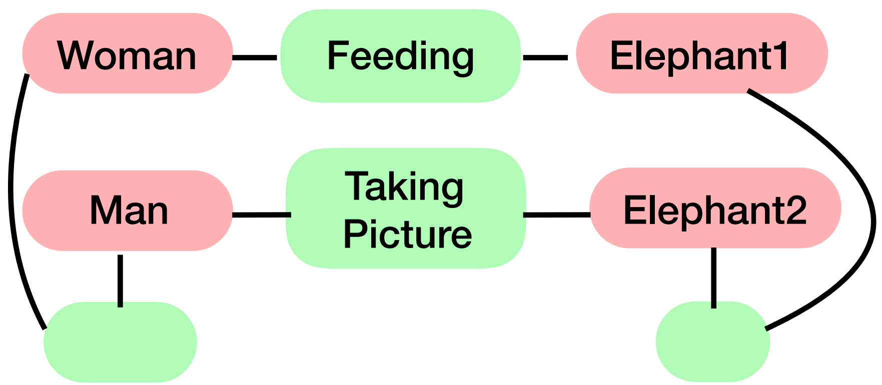
Scene Graphs Generation

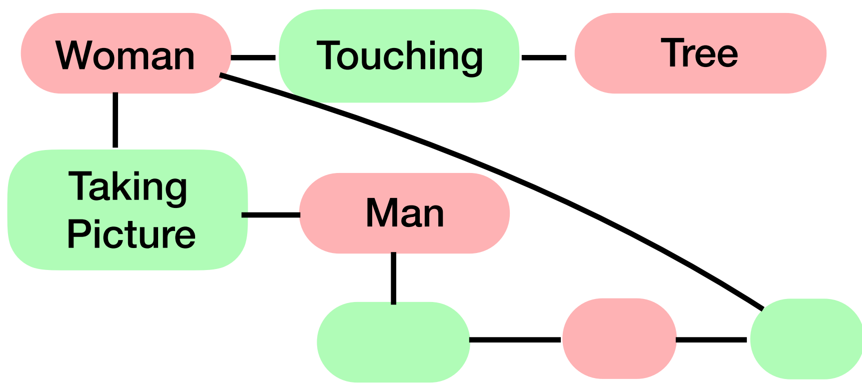
- Scene graph prediction is a “structured-prediction” problem (see Taskar 05).
- Namely, it has a complex and structured output.



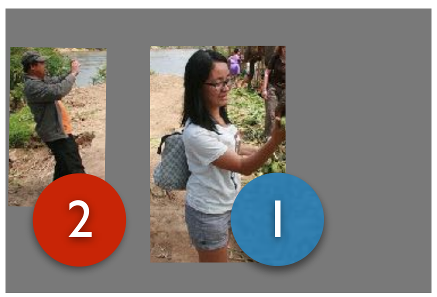
Context is Key!

- A scene interpretation must make sense as a whole
- Allows us to rule out explanations that are locally plausible but don't make sense globally.
- Structured prediction models (e.g., CRF, M^3N) can capture this.





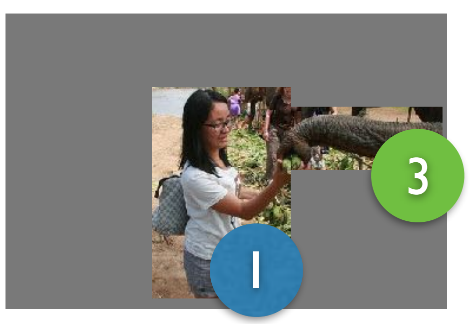
Typical Structured Prediction Architecture



2

1

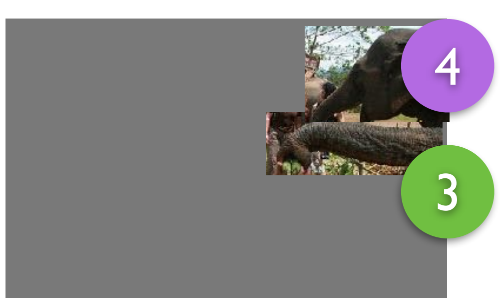
	Man	Woman	Tree	Elephant
Man	8	6		
Woman	12	5	1	
Tree	1			10
Elephant	1			3



3

1

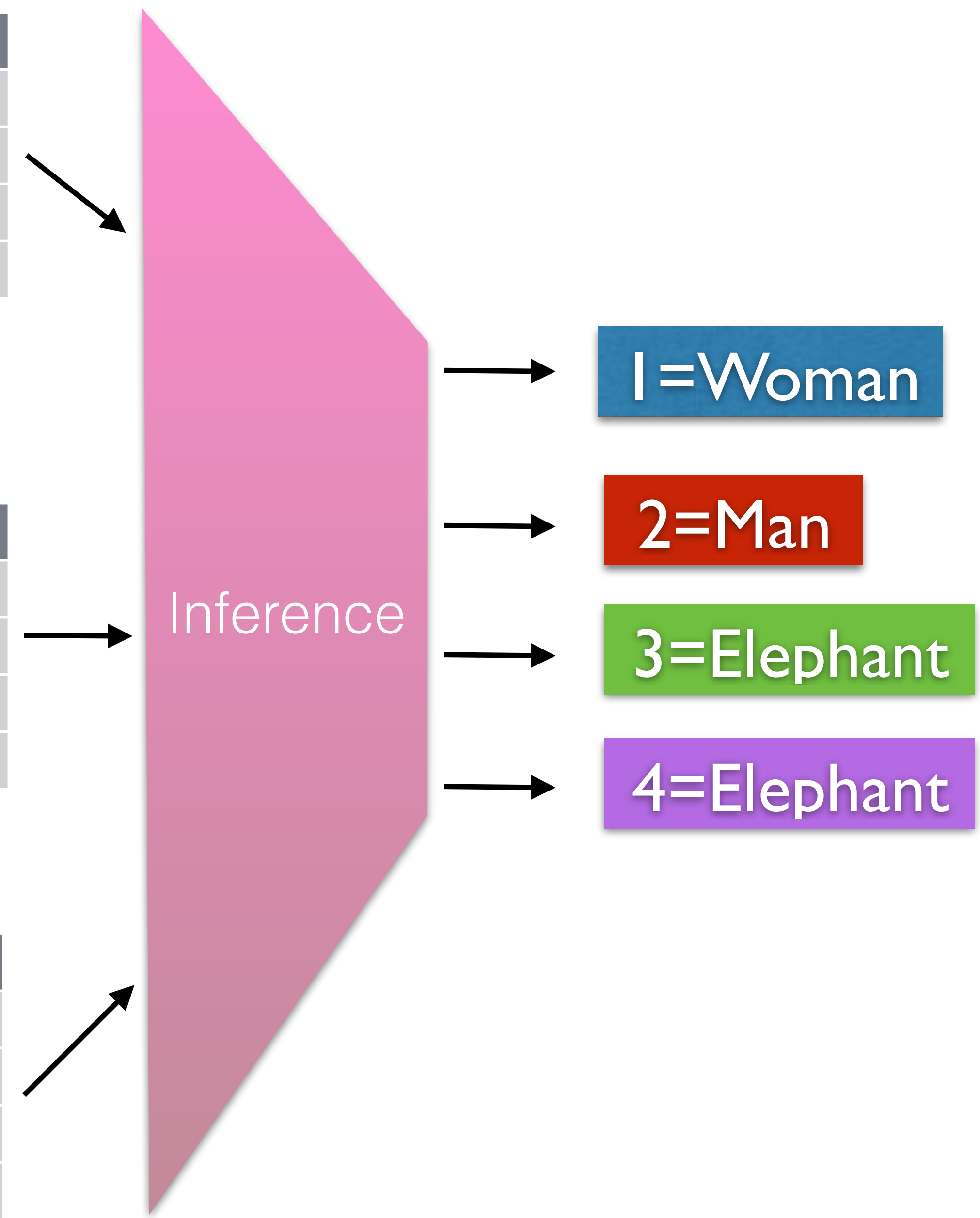
	Man	Woman	Tree	Elephant
Man	8	6		
Woman	12	5	25	18
Tree	1			6
Elephant	1		3	4



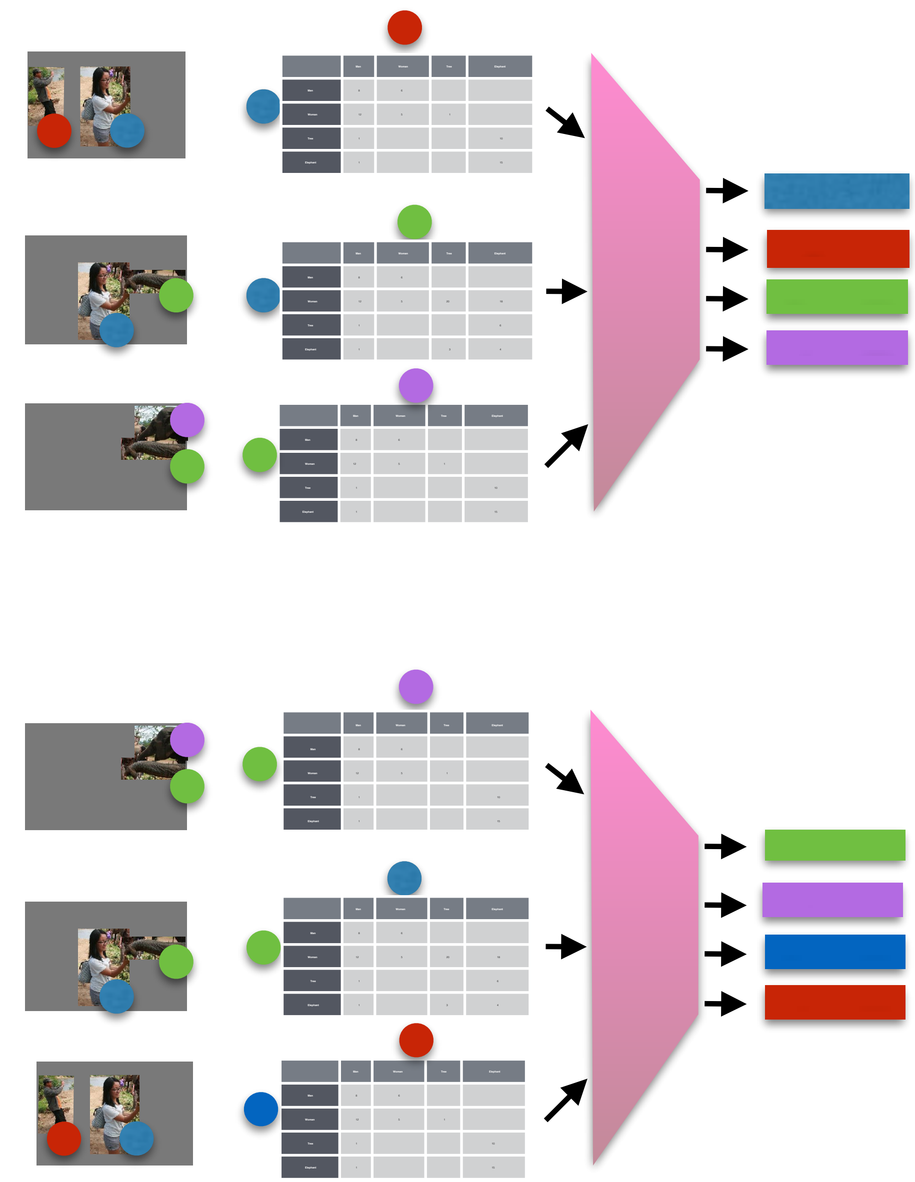
4

3

	Man	Woman	Tree	Elephant
Man	8	6		
Woman	12	5	1	
Tree	1			10
Elephant	1			40



Deep Structure Prediction Architecture



- The black box is often a message passing algorithm like belief propagation.

- But what if we want something more general?

- Which constraints should it satisfy?

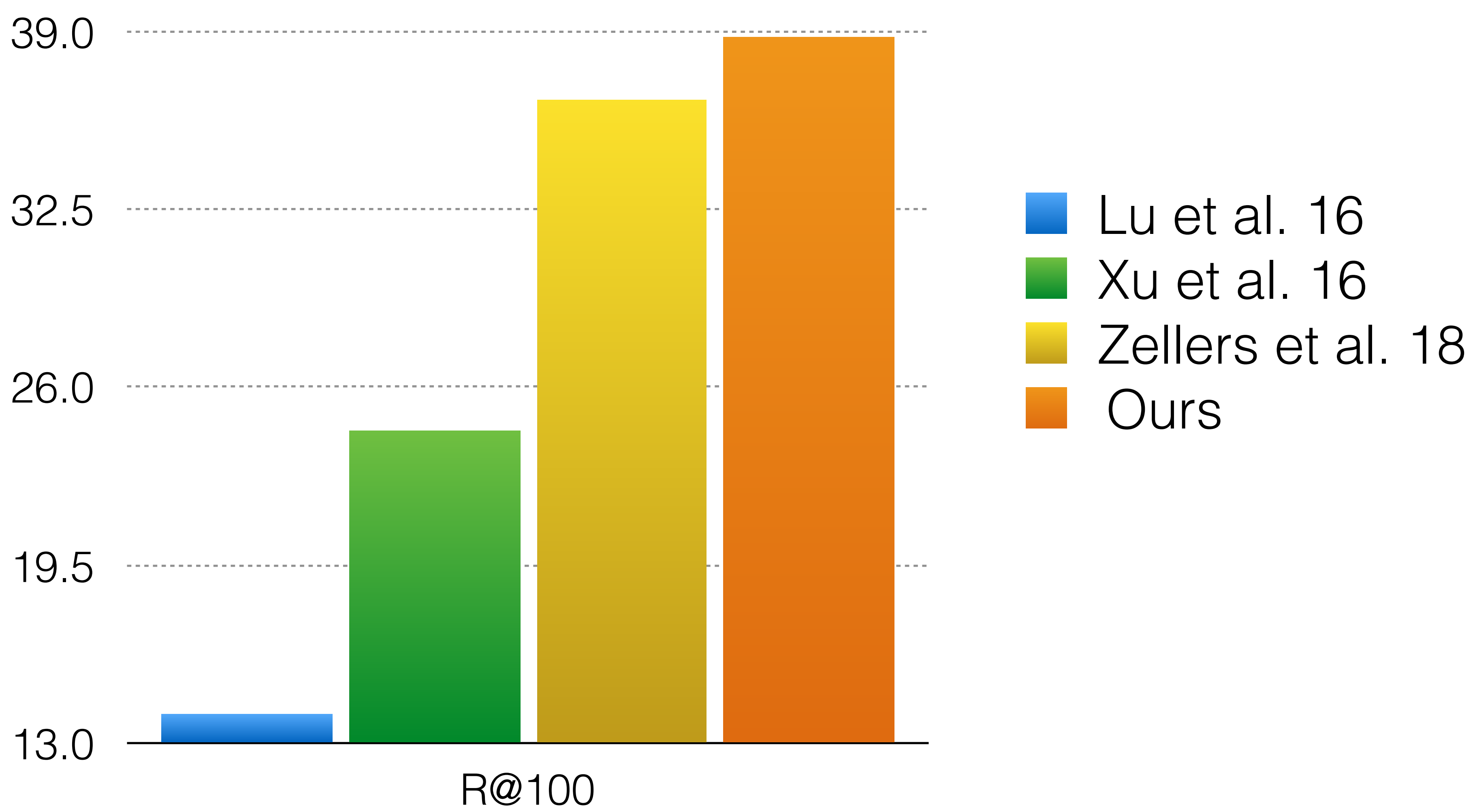
- It should be invariant to permutations of the input representation.

Invariant Structured Prediction

- We cast the desired invariance formally.
- Prove that it is satisfied **if and only if the architecture has a certain form** (specifically, it should be pooled in a certain way).
- This restricts the architecture but leaves much room for play, significantly extending message passing algorithms (e.g., with **attention**).

Results on Visual Genome

- Evaluated by returning 100 triplets and calculating recall w.r.t. ground truth.



Predict and Constrain

- Often want the solution y to satisfy property f(y)=c
- e.g., not more than 5 tables in image
- Value c may depend on input

 → f(y) = c → Constrained Opt. → y: f(y)=c

- Can be trained end-to-end (ICML 18)

Conclusions

- Optimization guarantees possible under distribution assumptions
- Generalization guarantees possible for linear case
- Design principles for complex labeling tasks
- What are optimization/generalization guarantees for these?