Convnets from First Principles:

Generative Models, Dynamic Programming & EM

Ankit B. Patel

Baylor College of Medicine (Neuroscience Dept.)

Rice University (ECE Dept.)

Stats 385, Stanford University 10-18-2017

Deep Learning:
What is it good for?

Why do we need Deep Learning?

[Girshick et al., CVPR 2014]

Key Challenge: Object recognition (and sensory perception in general) is plagued by large amounts of **nuisance variation**.

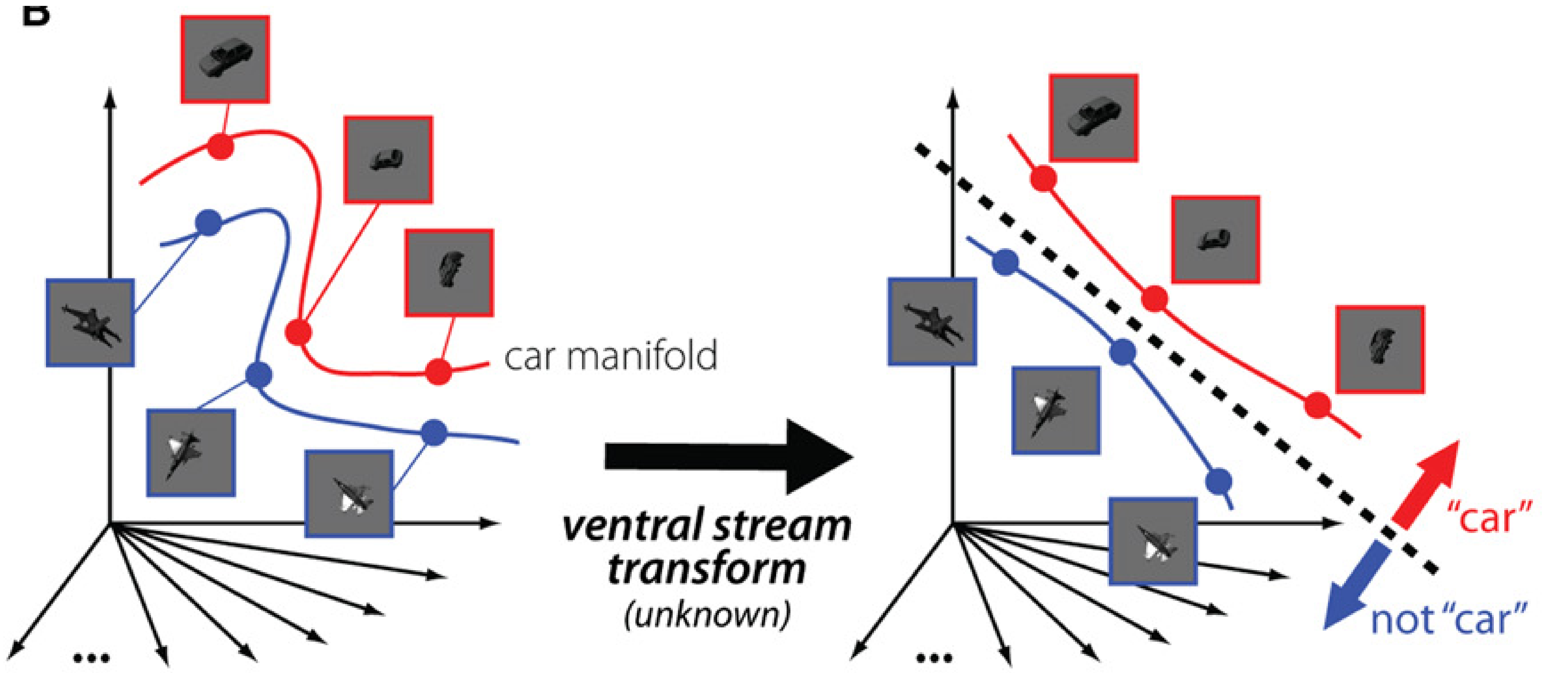
- **Nuisance Variation**: affects sensory input (image) but not the task target (object class)
- Ex: Object Recognition, Nuisances = changes in location, pose, viewpoint, lighting, expression, . . .
- Ex: Speech Recognition, Nuisances = changes in pitch, volume, pace, accent, . . .
- Nuisance variables are task-dependent and can be implicit

The Trouble with Nuisances

Problem: *How to deal with nuisance variation in the input?*

Solution: Build representations that are

- **Selective**: Sensitive to task-relevant (target) features
- **Invariant:** Robust to task-irrelevant (nuisance) features
- **Multi-task:** Useful for many different tasks



DiCarlo, J. J. et al. How does the brain solve visual object recognition? Neuron (2012).

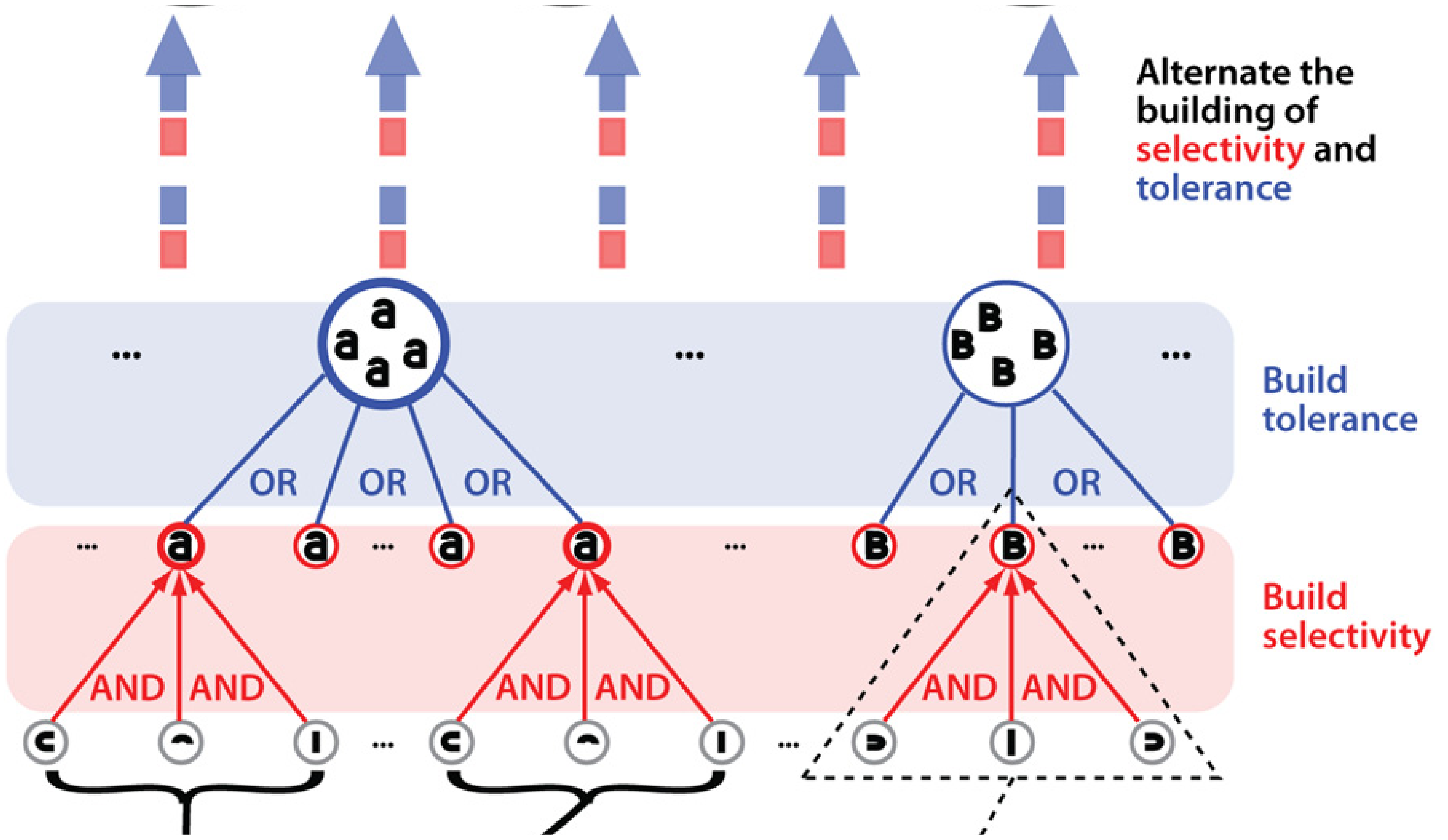
The Holy Grail of Machine Learning

Learn a **disentangled representation**:
one that factors out variation in the sensory input
into meaningful intrinsic degrees of freedom.

A Potential Solution: Deep Processing

Potential Solution: Look to the Brain for guidance.

- Hubel and Wiesel's discovery of simple/complex cells and their special properties of *selectivity* and *tolerance/invariance*



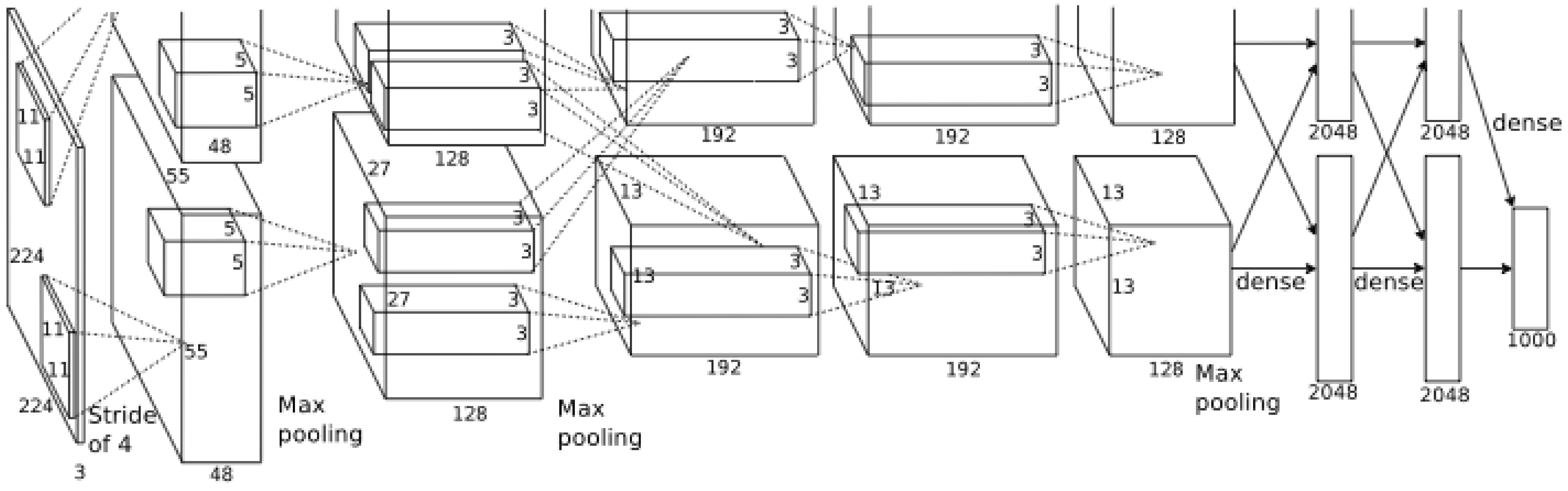
DiCarlo, J. J. et al. How does the brain solve visual object recognition? Neuron (2012).

Key Inspiration from Neuroscience

Build up feature selectivity and tolerance over multiple layers in a hierarchy $\Rightarrow$ ML architectures: Neocognitron, HMAX, SIFT, and modern **Deep Convnets**

Deep Learning:
Current Successes & Failures

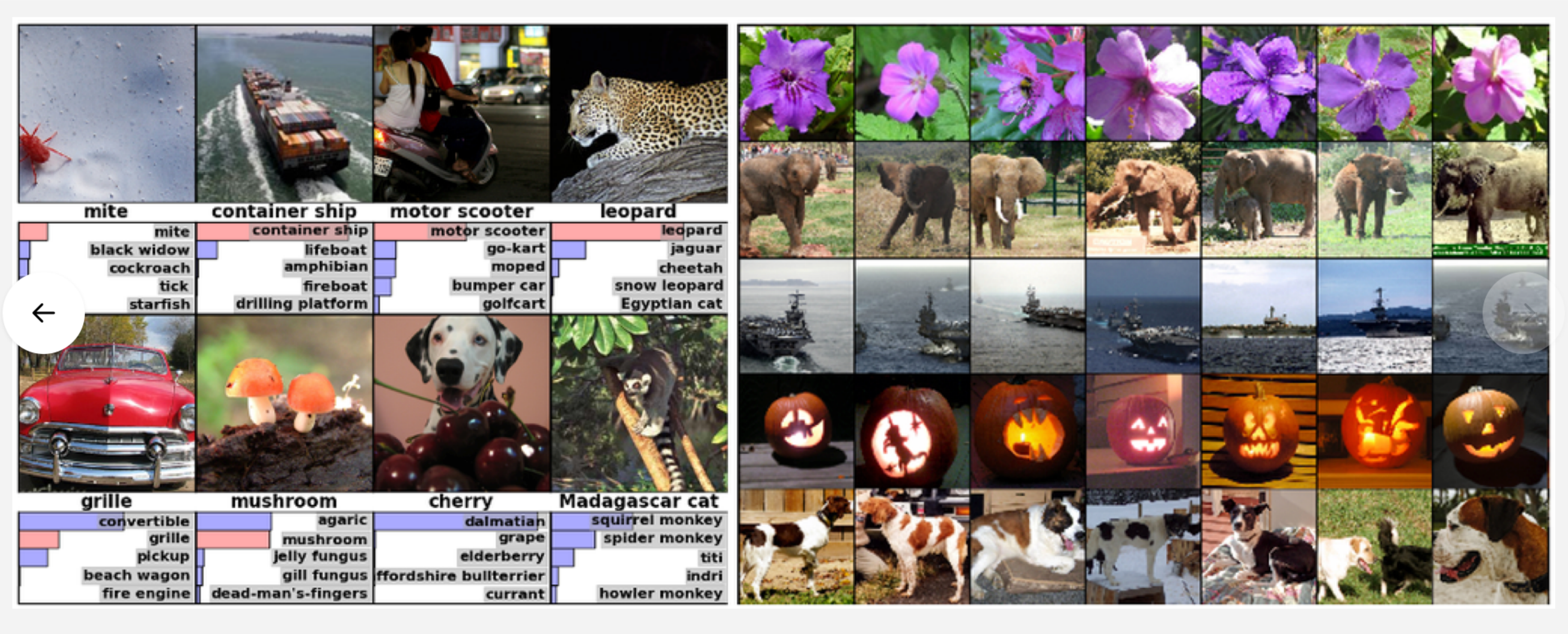
Object Recognition with Convnets



A. Krizhevsky et al. ImageNet classification with deep convolutional neural networks (NIPS 2012)

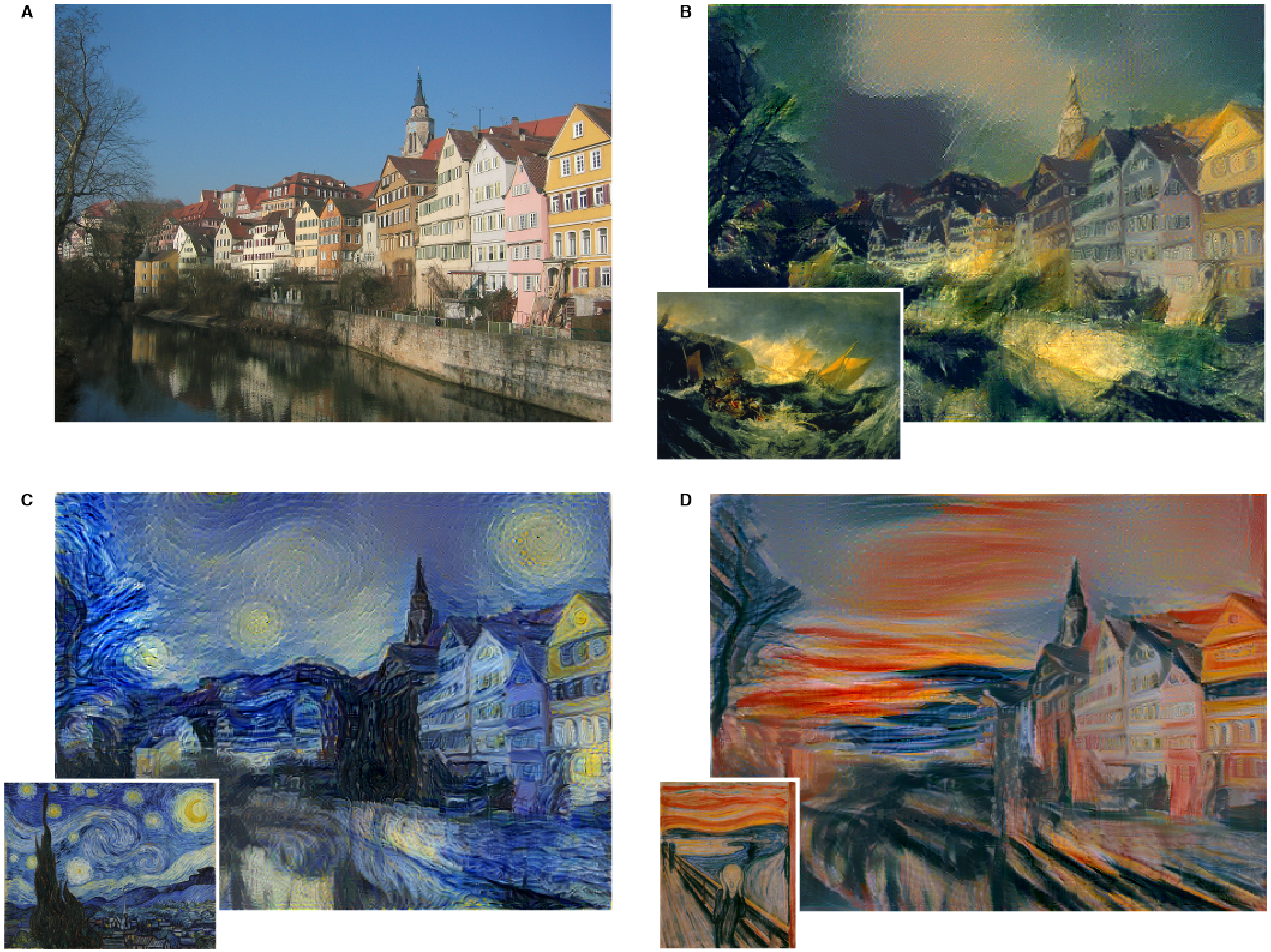
- **Deep Convnets**
  - 2012: Krizhevsky et al advanced state-of-the-art in object recognition in the ImageNet Challenge (1.2 million labeled images of objects)
  - Subsequently benchmarks in **many other vision tasks** were pushed forward many years ⇒ **Transfer Learning**
  - Recently, Google's and MSR's latest DCNs have achieved **95% accuracy**, with **superhuman** performance in most categories
  - Deployed commercially in Google and Baidu Personal Image Search

Object Recognition with Convnets



Transferring the Style from one Image to another

Content Image



Generative Models for Natural Images

(a) Varying c_1 on InfoGAN (Digit type)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

Some Cold Water: Tesla Autopilot Misclassifies Truck as Billboard

Problem: Why? How can you trust a blackbox?

Self-Driving Systems have Difficulty with Different kinds of Weather

BUSINESS INSIDER FINANCE

6 scenarios self-driving cars still can't handle

Danielle Muoio

Aug. 25, 2016, 9:02 AM 81,621

Intelligent Machines

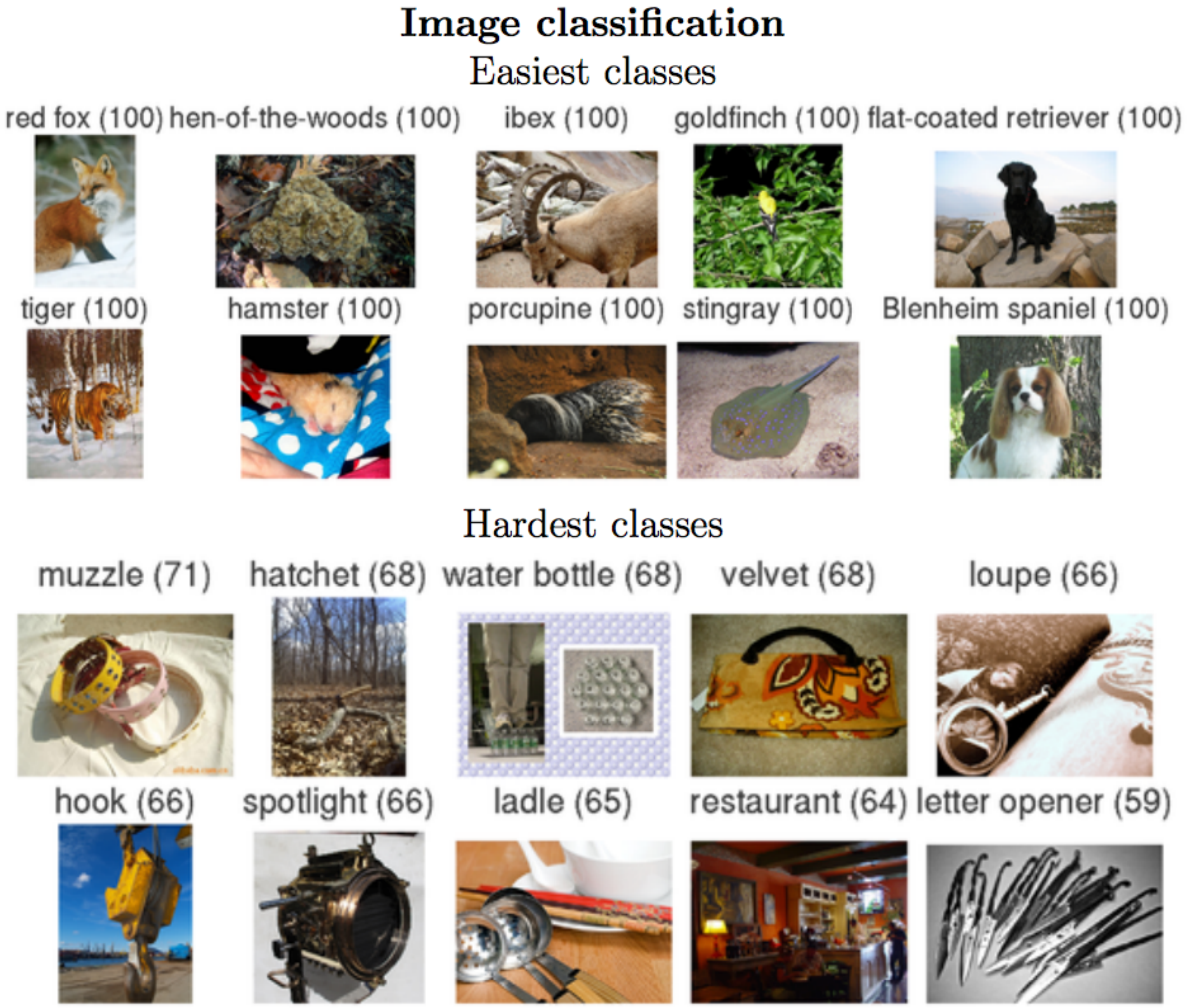
Hidden Obstacles for Google's Self-Driving Cars

Impressive progress hides major limitations of Google's quest for automated driving.

by Lee Gomes August 28, 2014

Problem: Self-driving cars have difficulty "seeing" in rain or heavy snow or when it gets cloudy.

“Kryptonite” Categories: Why do Convnets have difficulty with certain classes of objects?



Russakovsy et al. 2014

Convnets worse than humans on:

- small or thin objects,
- transparent objects,
- image filters,
- abstract representations (e.g. rendered, paintings, sketches, statues, plush toys),
- extreme closeups,
- unconventional viewpoints,
- heavy occlusion.

Problem: Why are Convnets so great at certain categories while struggling with others?

The Need for a Theory

Key Open Questions about Deep Learning Systems

- How and why do they work? Can we derive their structure from first principles? Can we compress/explain the myriad empirical observations/best practices about deep nets?
- Can we shed new light on the hidden representations of objects? Can we generate new theories and testable predictions for both artificial/real neuroscience?
- Why do they fail? How to improve them? How to alleviate their intrinsic limitations?
- Can we help guide the search for better architectures/algorithms/performance in applied DL?

Concrete Theoretical Questions

Key Open Questions about Deep Learning Systems

- What are the implicit modeling assumptions?
- What is the inference task and algorithm?
- What is the learning algorithm?
- Can we generate new testable predictions for artificial/real nets?
- What modeling assumptions are being violated in failures? How can we improve the models, tasks and algortihms?

Related Work

- Theories of Deep Learning:
  - **Anselmi, Poggio et al., i-Theory** — **Early influence: Notion of marginalizing over group transformations**
  - Soatto et al., Nuisance in vision — **Notion of nuisances in vision inference tasks (indep. developed)**
  - Darrell, Malik et al., Deformable parts models
  - Carin et al., Generative models
  - Arora et al., Reversible networks
  - Mallat, Bolcskei et al., Scattering Nets
  - **Papyan, Elad et al., Convolutional Sparse Coding model** — **Related Results, (indep. developed)**
  - Lecun et al., Local Minima are close to Global Minima
  - Bengio et al., origami folding theory

Outline

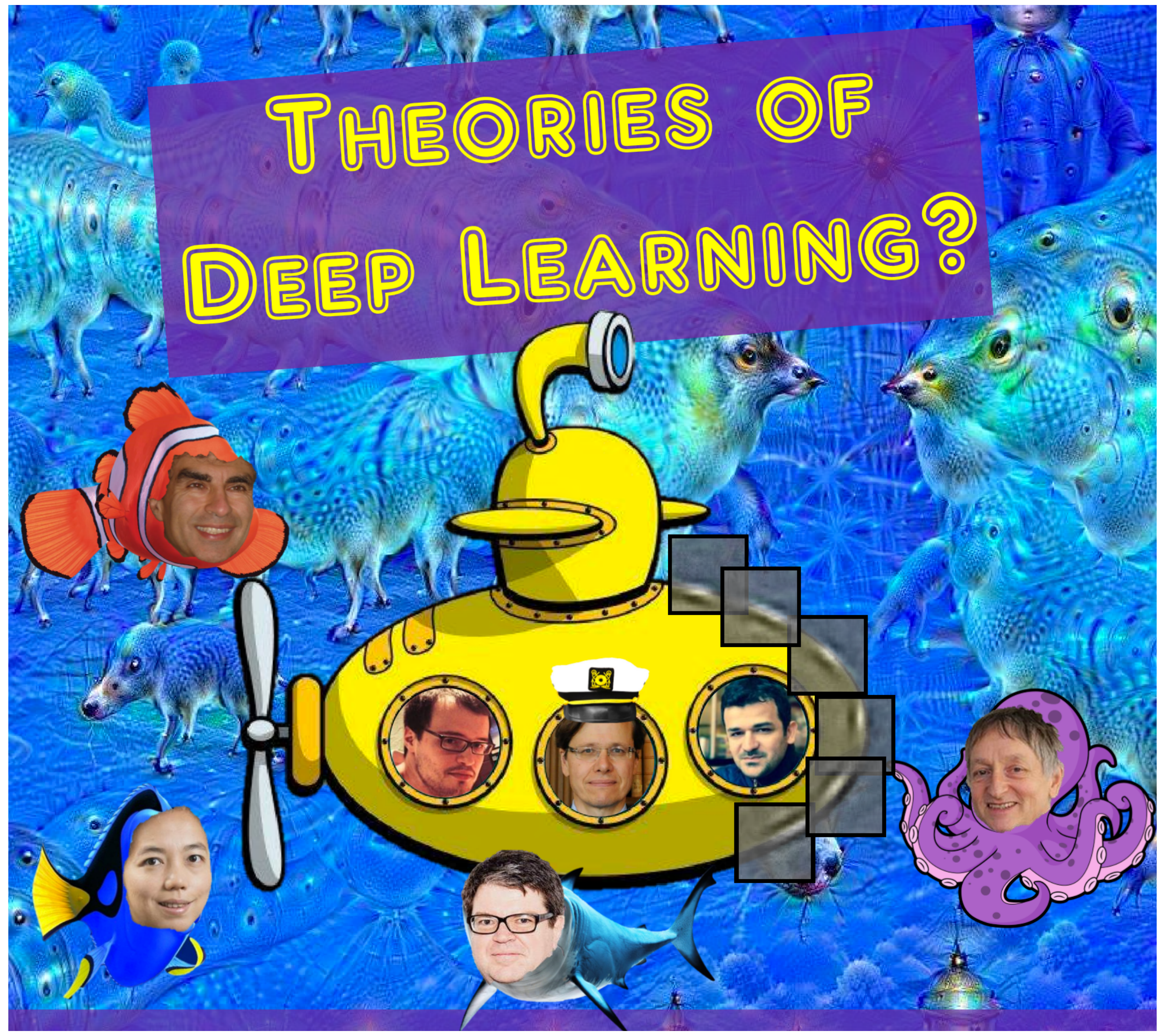
- Generative Model underlying Convnets
- Inference: The Dynamic Programming Interpretation of Convnets
- Learning: Hard EM Algorithm Interpretation
- New Explanations & Insights
- Limitations & Challenges & The Way Forward

Deep Convnets from First Principles: A Generative Modeling Approach

Many Species of Convnets…
But only a few Key Operations

There are many architectures, but just a few key operations and objectives:

- 2D (De)Convolution, Spatial max-(un)pooling, ReLu, Skip-connections
- Batch Normalization
- DropOut, Noise Corruption
- Data Augmentation
- Objectives: XEnt, NLL, Reconstruction Error, Mutual Information



Strategy: Focus on properties conserved across all species of Convnets

Strategy: Let's find a generative model

- Define a generative model that captures **nuisance variation**
- Recast feedforward propagation in a DCN as **MAP inference of the full latent configuration** (target + nuisance variables) $\rightarrow$ generative classifier
- Apply a **discriminative relaxation** $\Rightarrow_d$ discriminative classifier
- Learn the parameters via **Batch Hard EG Algorithm** $\rightarrow$ SGD-Backprop Training of a DCN
- Use new generative model to **address limitations of DCNs**: top-down inference, learning from unlabeled data, hyperparameter optimization,

Strategy: Let's find a generative model

If we succeed, some **great benefits**:

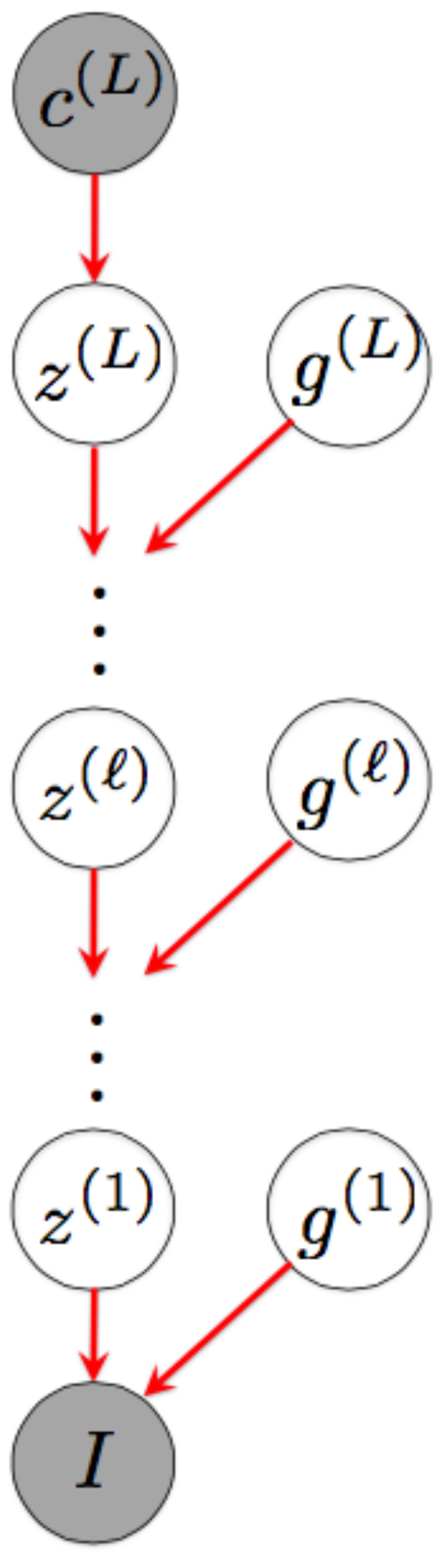
- Make clear the prior knowledge that's implicit
- Learn from unlabeled and labeled data
- Principled top-down inference for fine-scale tasks
- A systematic way to design new kinds of networks, improving performance and addressing weaknesses
- Model Selection for learning structure/architectural parameters
- etc. etc.... **all the good stuff that comes with generative model**

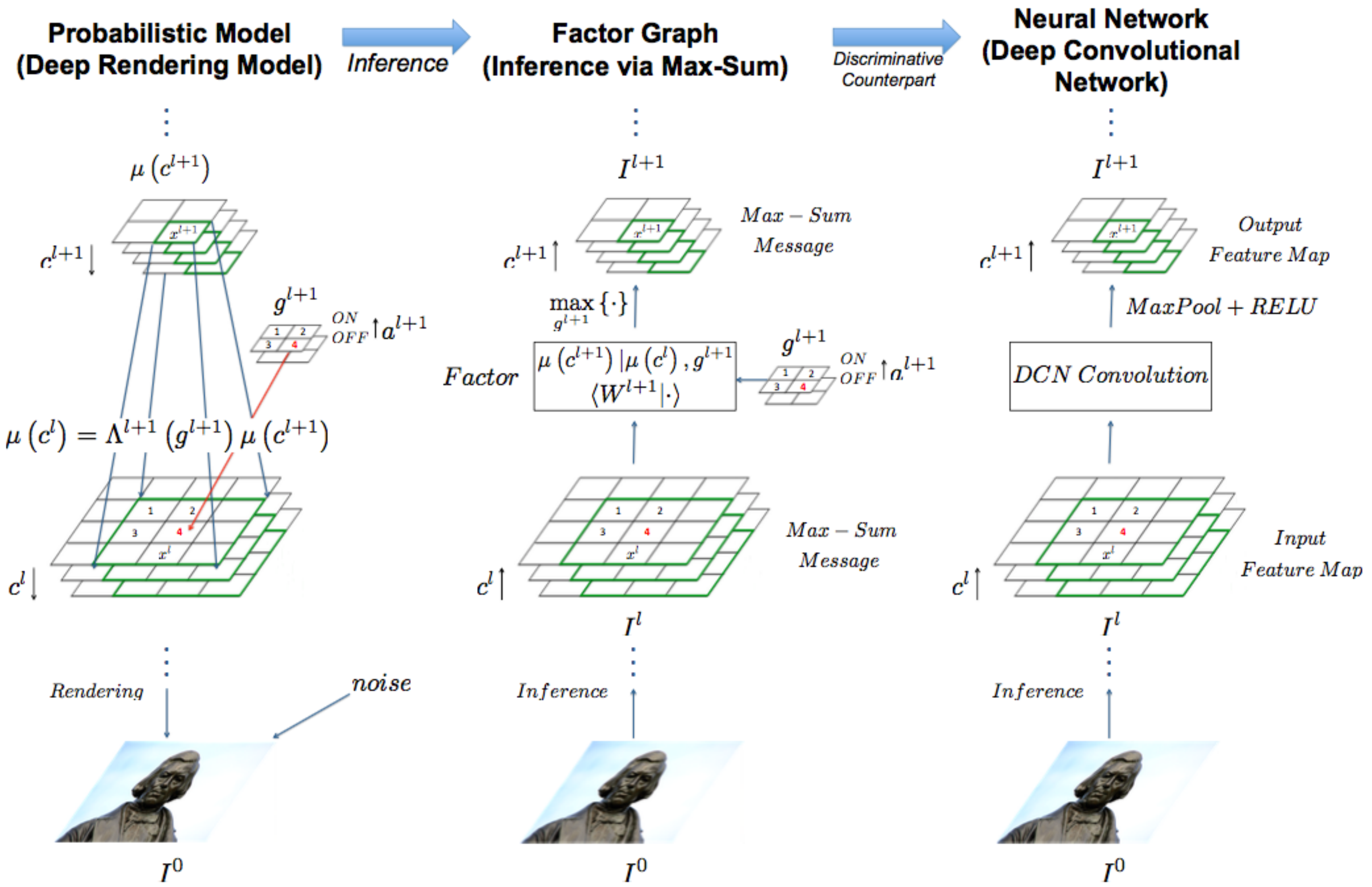
Main Strategy

If we can find a generative model underlying deep vision architectures,
we can go beyond convnets by addressing their limitations in a principled way.

Overview of Strategy: Reinterpret Convnets as Inference in Generative Model

Deep Rendering Mixture Model

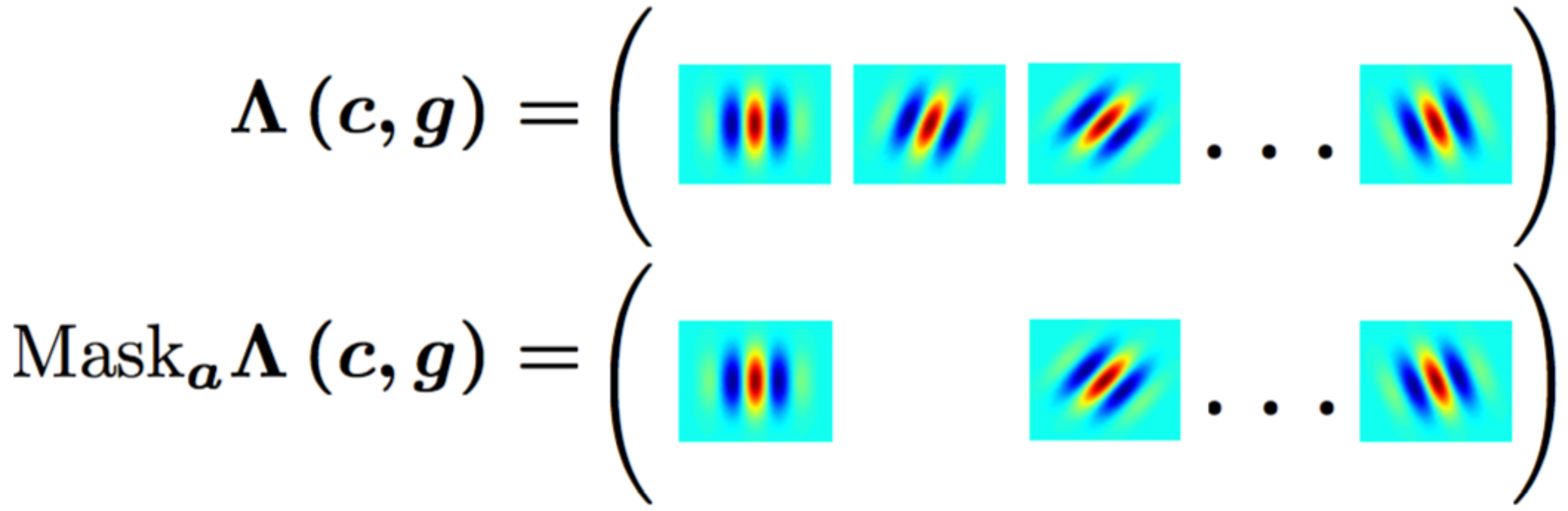




The Shallow Rendering Mixture Model: An Analogy with Sparse Coding

- **Rendering a sample from RMM:**
    1. Decide which elements of dictionary will be used (mask ***a***)
    2. Decide how much to weigh each element (factor **z**)
    3. Decide where to render (fine-scale position ***g***)
    4. Render image patch ***I***

$$a = [\ 1 \quad 0 \quad 1 \ \ldots \ 1\ ]$$



$a =$ switching vector, $z =$ factor loadings

$$I = \Lambda(c,g)(a \odot z) + \text{ noise} = \mathrm{Mask}_a \Lambda(c,g) z + \text{ noise}$$

Theorem: Inference in RMM yields One Layer of Convnet

Theoretical Result:

$$\hat{c}(I) = \arg\max_{c\in\mathscr{C}} \max_{g\in\mathscr{G}} \max_{a\in\mathscr{A}} \left\langle \frac{1}{\sigma^2} a\mu_{cg} \middle| I \right\rangle - \frac{1}{2\sigma^2}(\|a\mu_{cg}\|_2^2 + \|I\|_2^2) + \ln \pi_c \pi_g \pi_a$$

$$\equiv \text{Choose}(\text{MaxPool}(\text{ReLU}(\text{Conv}(\text{I}))))$$

Each Convnet operation has a Probabilistic Meaning

- Max-Sum Inference in the Shallow Rendering Model $\Rightarrow_d$ Feedforward propagation in a single Convnet layer:
- Translational invariance $\Rightarrow_d$ Convolutional layer
- Max-marginalizing over a and g $\Rightarrow_d$ ReLU, Max-pooling

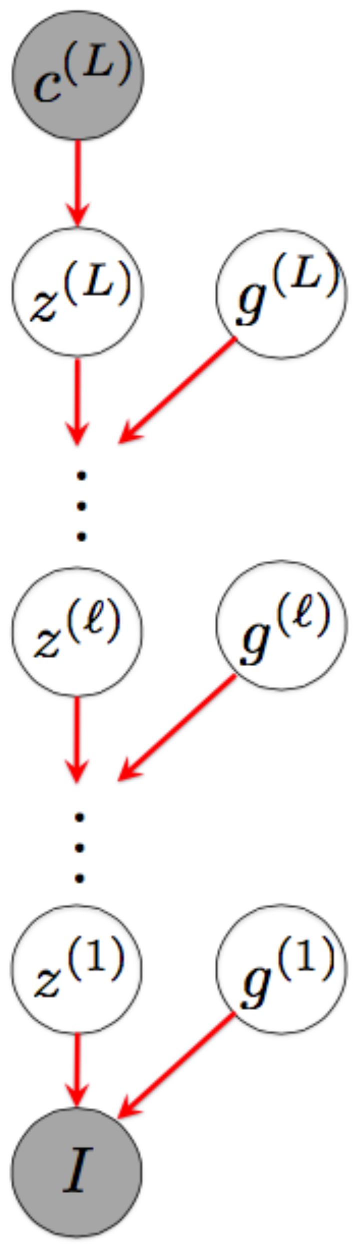
Deep Rendering Mixture Model (DRMM)

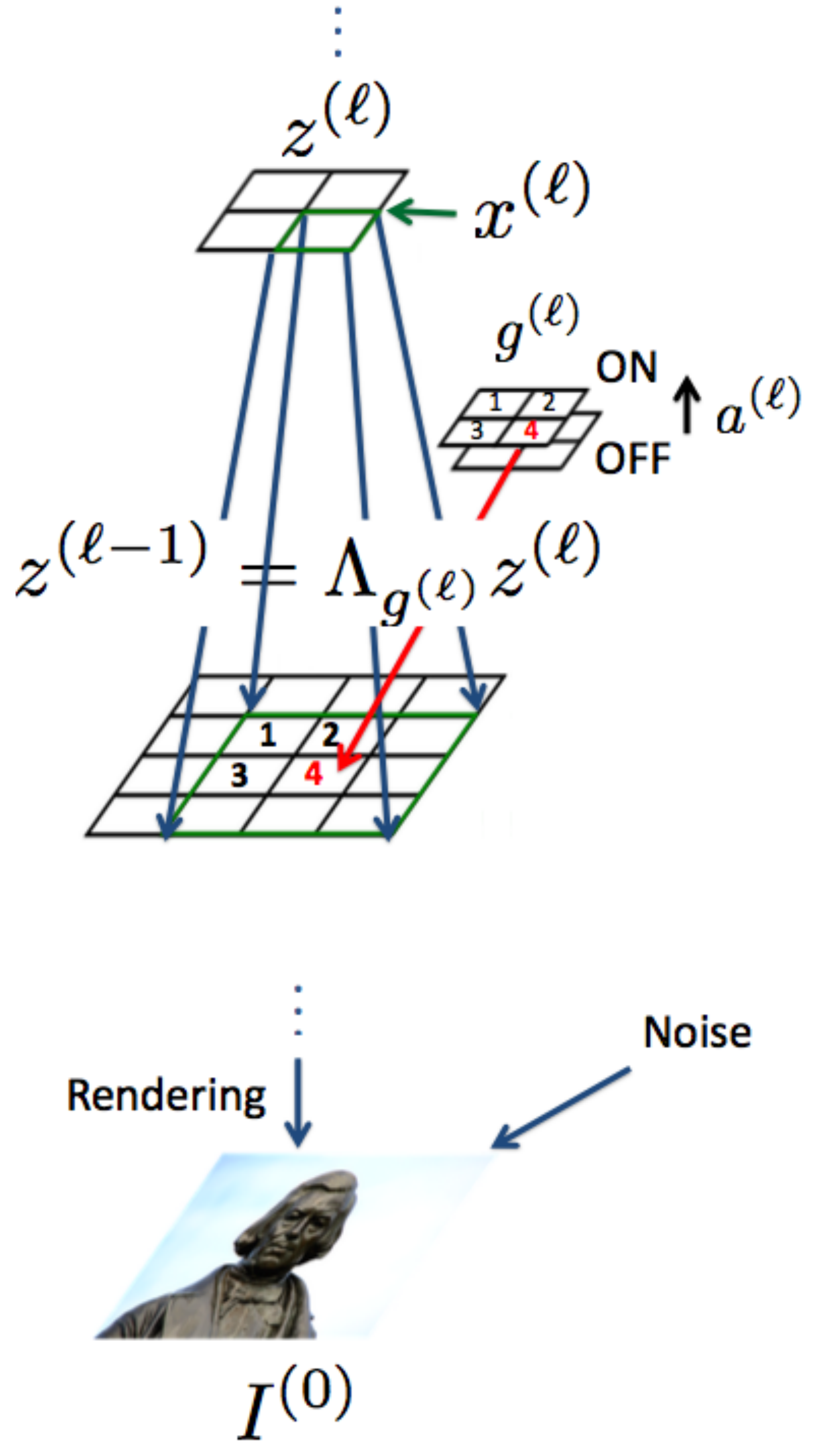
$$c^{(L)} \sim \mathrm{Cat}(\{\pi_{c^{(L)}}\}), \quad g^{(\ell)} \sim \mathrm{Cat}(\{\pi_{g^{(\ell)}}\})$$

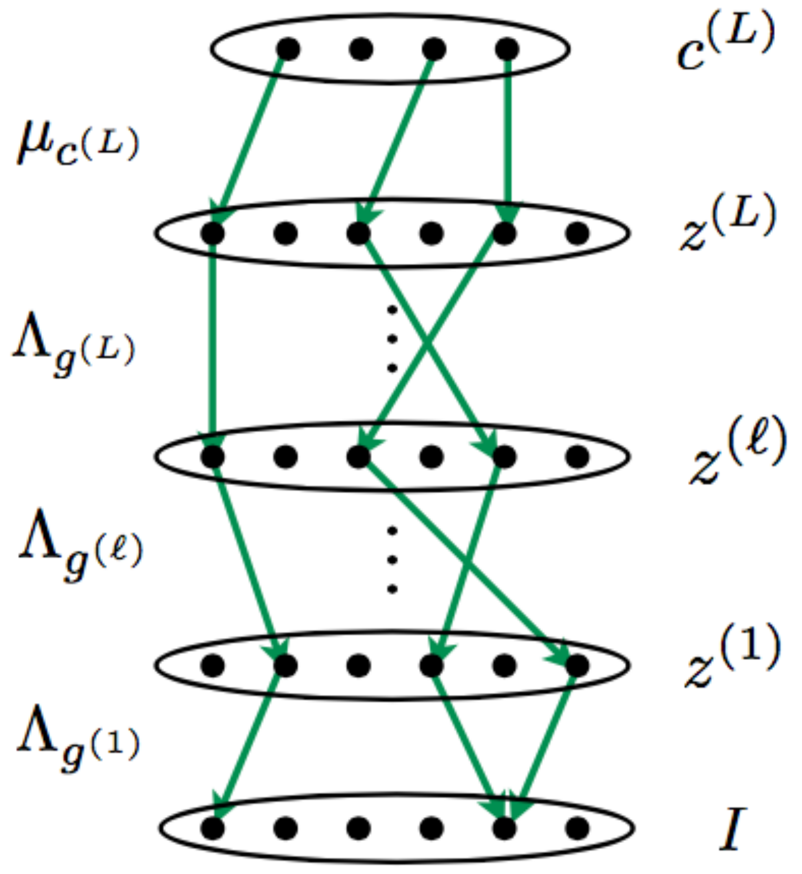
$$\mu_{cg} \equiv \Lambda_g \mu_{c^{(L)}} \equiv \Lambda^{(1)}_{g^{(1)}} \Lambda^{(2)}_{g^{(2)}} \cdots \Lambda^{(L-1)}_{g^{(L-1)}} \Lambda^{(L)}_{g^{(L)}} \mu_{c^{(L)}}$$

$$I \sim \mathcal{N}(\mu_{cg}, \sigma^2 \mathbf{1}_{D^{(0)}}),$$

- Latent **Nuisance** variables control correlations at multiple length scales
- **Inference:** turns out to be Convnet (bottom-up pass)
- **Learning**: Can use EM algorithm
- **Upshot:** Unifies supervised, unsupervised, and semi-supervised learning for Convnets







A. B. Patel, **T. Nguyen**, R. G. Baraniuk.

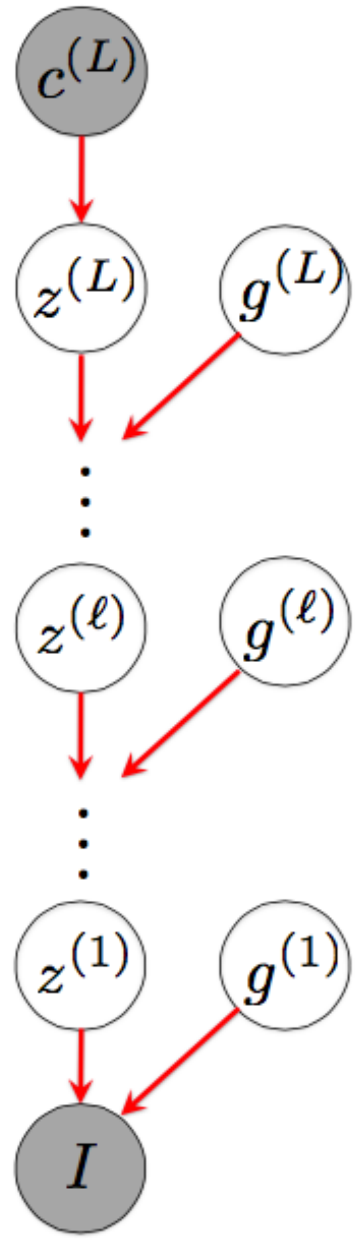
A Probabilistic Framework for Deep Learning. NIPS 2016

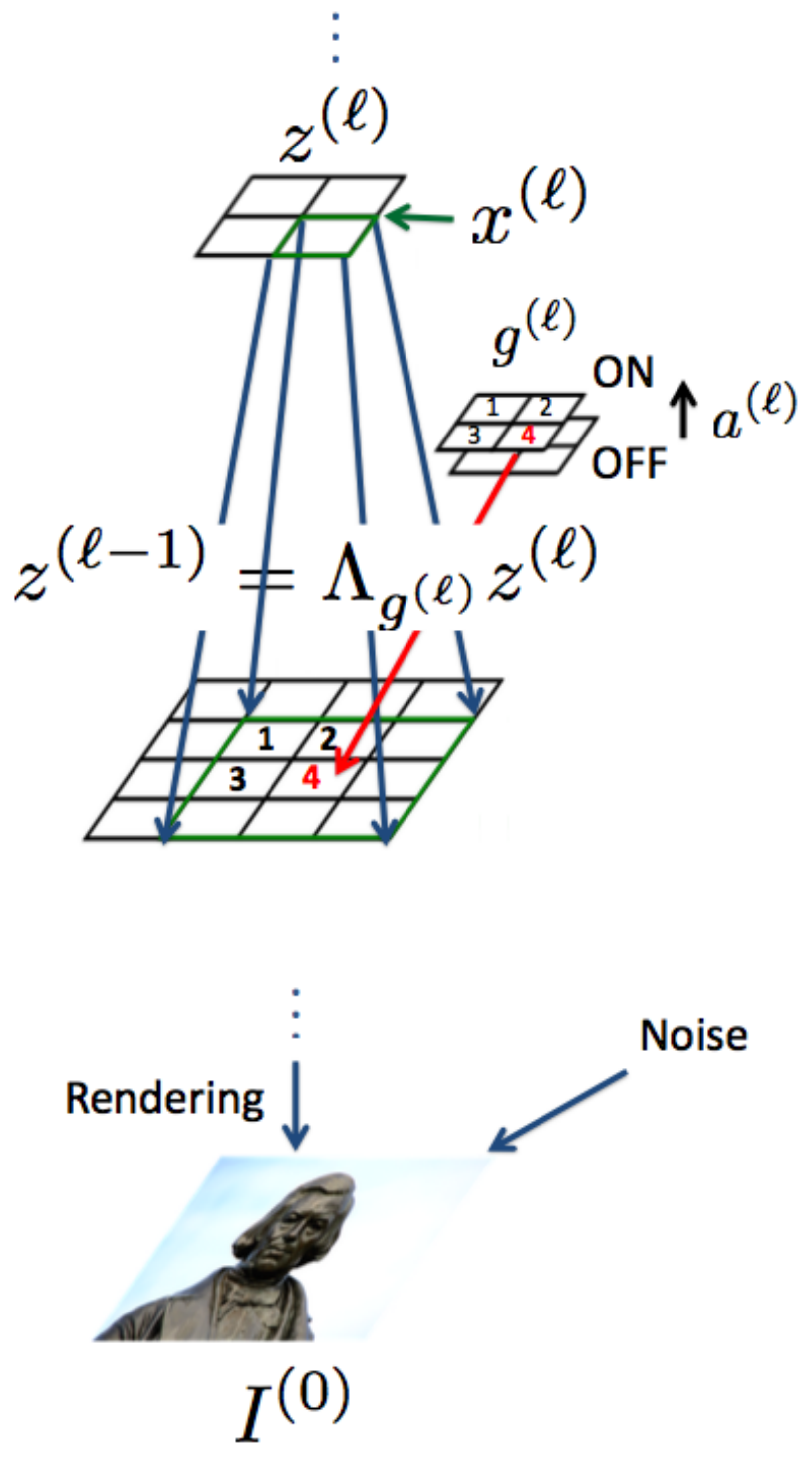
Each Layer of the DRMM is a Sparse Coding Model

Rendering Process: (1) Choose fine-scale location t, (2) choose words a from dictionary Γ, and then (3) render:

- $g_x^\ell \equiv (t_x^\ell, a_x^\ell) \in \{UL, UR, LL, LR\} \times \{ON, OFF\}$
- $z_n^{(\ell)} = \Lambda_{g_n^{(\ell+1)}} z_n^{(\ell+1)}$ where $\Lambda_{g^\ell} \equiv \Gamma^{(\ell)} M_{a^{(\ell)}} \mathscr{T}_{t^{(\ell)}}$ and $M_a \equiv \mathrm{diag}(a)$.

Related Work: Similar to the Conv. Sparse Coding Model, developed independently by Papyan-Romano-Elad (2016)





A. B. Patel, **T. Nguyen**, R. G. Baraniuk.

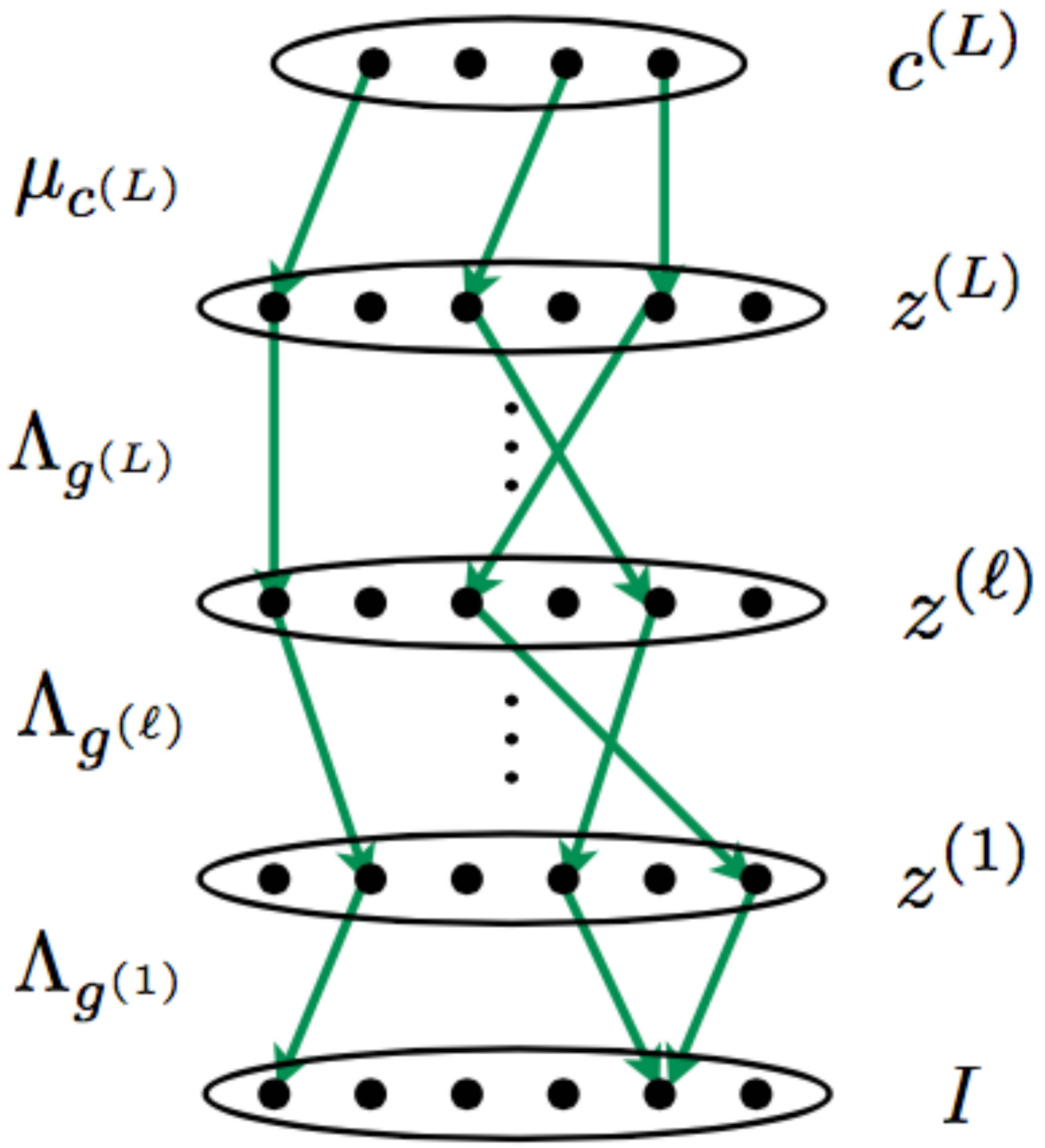
A Probabilistic Framework for Deep Learning. NIPS 2016

Deep Rendering Mixture Model: The Sum-over-Paths Formulation

Rendering Process: (1) Choose fine-scale location t, (2) choose words a from dictionary Γ, and then (3) render:

- $g_x^\ell \equiv (t_x^\ell, a_x^\ell) \in \{UL, UR, LL, LR\} \times \{ON, OFF\}$
- $z_n^{(\ell)} = \Lambda_{g_n^{(\ell+1)}} z_n^{(\ell+1)}$ where $\Lambda_{g^\ell} \equiv \Gamma^{(\ell)} M_{a^{(\ell)}} \mathcal{T}_{t^{(\ell)}}$ and $M_a \equiv \text{diag}(a)$.
- $I_x = \sum_{p:c \to x} \prod_\ell t_p^{(\ell)} a_p^{(\ell)} \gamma_p^{(\ell)} = \sum_p t_p a_p \gamma_p$. **[Defn. of Matrix Mult.]**

Intuition: Generalization of Sparse Coding model to a **Sparse Path Coding Model** with a dictionary of (exponentially many) paths from category c^L to input pixel I_x, but of which only a **sparse** subset of **active paths** (defined by $\{t_x^{(\ell)}, a_x^{(\ell)}\}$) are used to explain/render a single image (patch).



Related Work: Sum-over-Paths inspired by Feynmann's formulation of QM and Choromanska et al. (2014)

A. B. Patel, **T. Nguyen**, R. G. Baraniuk.

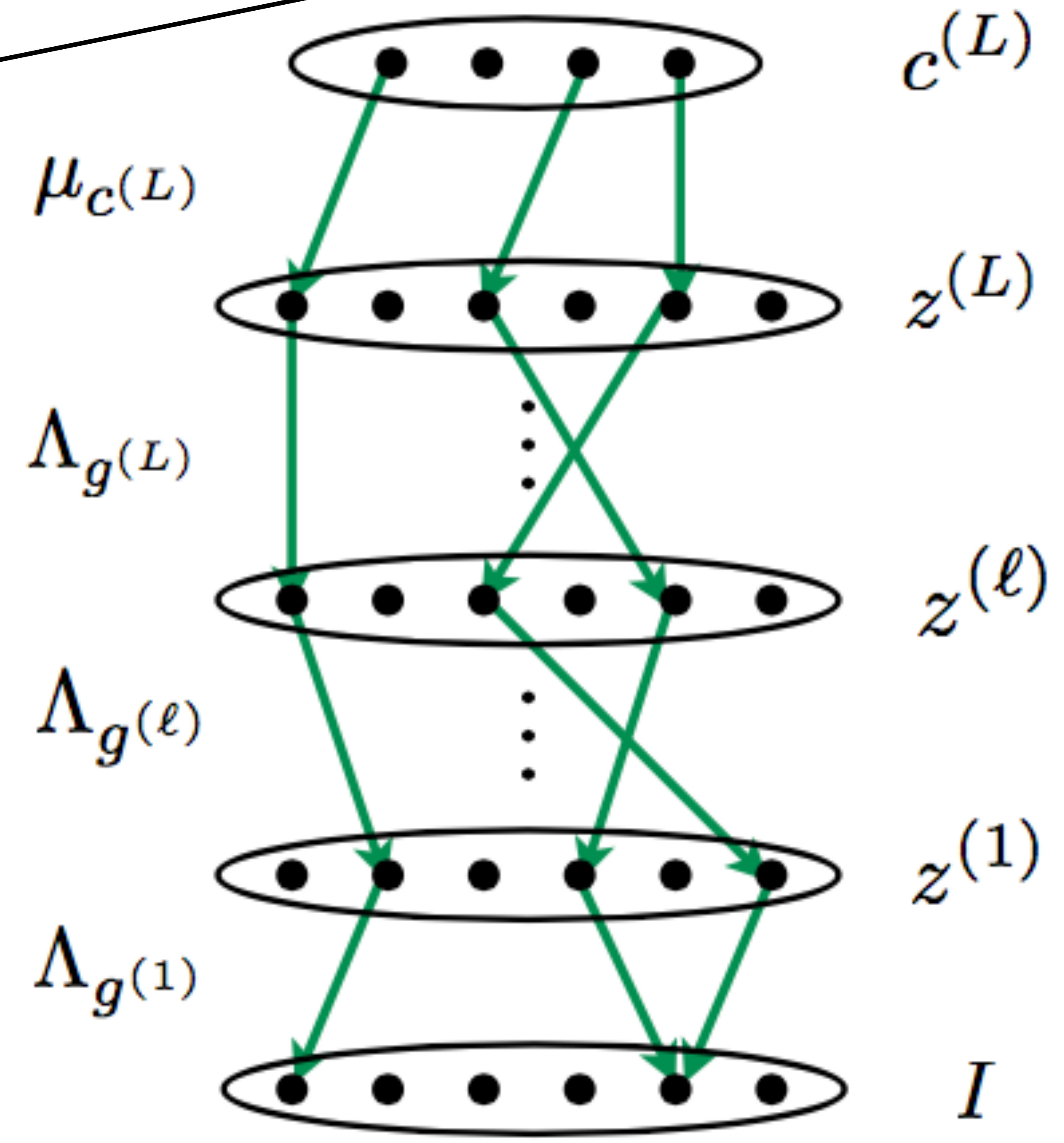
A Probabilistic Framework for Deep Learning. NIPS 2016

What do the Active Paths mean?

Active Paths Encode/Lead to Task-Relevant Pixels

- DRMM Generation equivalent to sum over **active** paths from top to bottom
  - A path *p* is **active** if all switching variables on that path are active.
  - Each active path's contribution = product of weights.
  - Since only a few neurons are ON, very few of all possible paths will be active.
- **Interpretation:** Active paths encode/lead to *Task-Relevant (aka salient) Pixels*

$$I_x = \sum_{p:c\to x} \prod_\ell t_p^{(\ell)} a_p^{(\ell)} \gamma_p^{(\ell)} = \sum_p t_p a_p \gamma_p$$



Inference

Question: What is the inference task performed by a Convnet? (1 min)

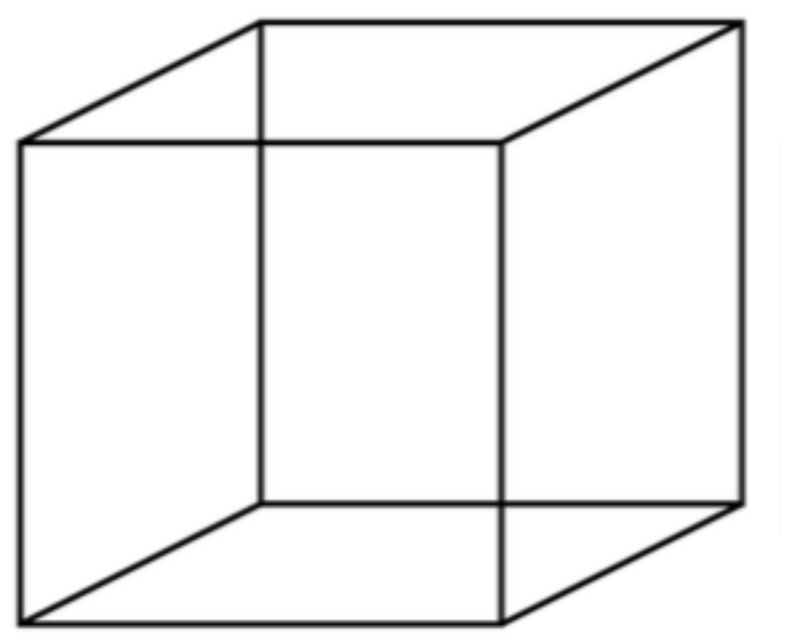
Ans: *JMAP* Inference of the *entire* configuration of Latents in the DRMM yields Deep Convnets

- **What is the Inference Task?** *Joint MAP* inference of category c *and* latent nuisance variables *g = (a, t)*

$$\hat{c}_{\mathrm{MS}}(I) = \arg\max_{c \in \mathscr{C}} \max_{g \in \mathscr{G}} p(I|c,g)p(c)p(g)$$

- Must infer a **single, globally consistent** configuration, not just the overall category.

- **Intuition:** Necker Cube, Face-Vase Illusion. If two global interpretations are equally likely, pick one but not both.



Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!

- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM

- Recovers structure of Deep Convnet *exactly*.

- **Proof:** "Pushing the max to the right." It is a bit involved but see Supplement of our latest papers for details

$$\hat{c}_n = \underset{c}{\operatorname{argmax}} \max_{g} \mu_{cg}^T I_n^{(0)} \qquad (17)$$

[JMAP Inference Task]

Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!

- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM

- Recovers structure of Deep Convnet *exactly*.

- **Proof:** “Pushing the max to the right.” It is a bit involved but see Supplement of our latest papers for details

$$\begin{aligned}
\hat{c}_n &= \underset{c}{\operatorname{argmax}} \max_{g} \mu_{cg}^T I_n^{(0)} && (17)\\
&= \underset{c}{\operatorname{argmax}} \max_{g} (\Lambda_g \mu_c)^T I_n^{(0)} && (18)\\
&= \underset{c}{\operatorname{argmax}} \max_{g(L:1)} \mu_c^T \Lambda_{g(L)}^T \cdots \Lambda_{g(2)}^T \Lambda_{g(1)}^T I_n^{(0)} && (19)
\end{aligned}$$

[DRMM Defn.]

Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!
- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM
- Recovers structure of Deep Convnet *exactly*.
- **Proof:** “Pushing the max to the right.” It is a bit involved but see Supplement of our latest papers for details

$$
\begin{aligned}
\hat{c}_n &= \operatorname*{argmax}_{c} \max_{g} \mu_{cg}^T I_n^{(0)} && (17)\\
&= \operatorname*{argmax}_{c} \max_{g} (\Lambda_g \mu_c)^T I_n^{(0)} && (18)\\
&= \operatorname*{argmax}_{c} \max_{g^{(L:1)}} \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T \Lambda_{g^{(1)}}^T I_n^{(0)} && (19)\\
&= \operatorname*{argmax}_{c} \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}} \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T (M_{a^{(1)}} \Lambda_{t^{(1)}}^T) I_n^{(0)} && (20)
\end{aligned}
$$

[DRMM Layer Defn.]

Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!
- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM
- Recovers structure of Deep Convnet *exactly*.
- **Proof:** "Pushing the max to the right." It is a bit involved but see Supplement of our latest papers for details

$$\hat{c}_n = \underset{c}{\operatorname{argmax}} \max_{g} \mu_{cg}^T I_n^{(0)} \tag{17}$$

$$= \underset{c}{\operatorname{argmax}} \max_{g} (\Lambda_g \mu_c)^T I_n^{(0)} \tag{18}$$

$$= \underset{c}{\operatorname{argmax}} \max_{g(L:1)} \mu_c^T \Lambda_{g(L)}^T \cdots \Lambda_{g(2)}^T \Lambda_{g(1)}^T I_n^{(0)} \tag{19}$$

$$= \underset{c}{\operatorname{argmax}} \max_{g(L:2)} \max_{t(1)} \max_{a(1)} \mu_c^T \Lambda_{g(L)}^T \cdots \Lambda_{g(2)}^T (M_{a(1)} \Lambda_{t(1)}^T) I_n^{(0)} \tag{20}$$

$$= \underset{c}{\operatorname{argmax}} \max_{g(L:2)} \max_{t(1)} \max_{a(1)} \underbrace{\left(\mu_c^T \Lambda_{g(L)}^T \cdots \Lambda_{g(2)}^T\right)}_{\equiv z^{(1)\downarrow T}} M_{a(1)} \underbrace{\left(\Lambda_{t(1)}^T I_n^{(0)}\right)}_{\equiv u_n^{(1)\uparrow}(t^{(1)})} \tag{21}$$

$$= \underset{c}{\operatorname{argmax}} \max_{g(L:2)} \max_{t(1)} \max_{a(1)} z^{(1)\downarrow T} M_{a(1)} u_n^{(1)\uparrow}(t^{(1)}) \tag{22}$$

[Assoc. of Matrix Multiplication]

Exercise: Solve this optimization (2 min)

- **Exercise:** To get a feel for how the DP algorithm works, try solving this optimization in closed form. Note that *z, a, u* are all *D*-dim vectors.

- **Hint:** “Push the max to the right.”

$$\max_{a \in \{0,1\}^D} z^T M_a u \qquad M_a \equiv \mathrm{diag}(a)$$

Exercise: Solve this optimization (2 min)

- **Exercise:** Try *(a)* solving this optimization in closed form. Note that *z, a, u* are all *D*-dim vectors. *(b)* What if *z* is nonnegative?

$$\max_{a\in\{0,1\}^D} z^T M_a u \qquad M_a \equiv \text{diag}(a)$$

- **Hint:** "Push the max to the right."

- **Proof:**

$$
\begin{aligned}
v^\star &\equiv \max_{a\in\{0,1\}^D} \mathcal{M}_a(z^T)u \\
&= \max_{a\in\{0,1\}^D} z^T \operatorname{diag}(a) u \\
&= \max_{a\in\{0,1\}^D} \sum_{i\in[D]} z_i a_i u_i \\
&= \sum_{i\in[D]} \max_{a_i\in\{0,1\}} a_i(z_i u_i)
\end{aligned}
$$

max-sum

$$
\begin{aligned}
&\equiv \sum_{i\in[D]} \hat{a}_i(z_i u_i) \qquad \text{solution } \hat{a} \text{ is given by } \hat{a} = [z \odot u > 0]. \\
&= \sum_{i\in[D]} [z_i u_i > 0] \cdot z_i u_i \\
&= \sum_{i\in[D]} \operatorname{ReLu}(z_i u_i) \\
&= \mathbf{1}_D^T \operatorname{ReLu}(z \odot u). \\
&= \operatorname{sgn}(z) \odot \operatorname{ReLu}\left(\operatorname{sgn}(z) \odot u\right)
\end{aligned}
$$

Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!
- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM
- Recovers structure of Deep Convnet *exactly*.
- **Proof:** “Pushing the max to the right.” It is a bit involved but see Supplement of our latest papers for details

$$
\begin{aligned}
\hat{c}_n &= \underset{c}{\operatorname{argmax}}\ \max_{g}\ \mu_{cg}^T I_n^{(0)} && (17)\\
&= \underset{c}{\operatorname{argmax}}\ \max_{g} (\Lambda_g \mu_c)^T I_n^{(0)} && (18)\\
&= \underset{c}{\operatorname{argmax}}\ \max_{g^{(L:1)}}\ \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T \Lambda_{g^{(1)}}^T I_n^{(0)} && (19)\\
&= \underset{c}{\operatorname{argmax}}\ \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}}\ \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T (M_{a^{(1)}} \Lambda_{t^{(1)}}^T) I_n^{(0)} && (20)\\
&= \underset{c}{\operatorname{argmax}}\ \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}}\ \underbrace{\left(\mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T\right)}_{\equiv z^{(1)\downarrow T}} M_{a^{(1)}} \underbrace{\left(\Lambda_{t^{(1)}}^T I_n^{(0)}\right)}_{\equiv u_n^{(1)\uparrow}(t^{(1)})} && (21)\\
&= \underset{c}{\operatorname{argmax}}\ \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}}\ z^{(1)\downarrow T} M_{a^{(1)}} u_n^{(1)\uparrow}(t^{(1)}) && (22)\\
&\overset{(a)}{=} \underset{c}{\operatorname{argmax}}\ \max_{g^{(L:2)}} \max_{t^{(1)}}\ z^{(1)\downarrow T} M_{\hat{a}_n^{(1)}} u_n^{(1)\uparrow}(t^{(1)}) && (23)
\end{aligned}
$$

[Exercise]

Inference in the DRMM yields Deep Convnets

- **Surprise:** Joint MAP inference of latent configuration can be done *exactly* in NN-DRMM!

- Use of max-product Dynamic Programming algorithm that exploits **recursive substructure** in DRMM

- Recovers structure of Deep Convnet *exactly*.

- **Proof:** "Pushing the max to the right." It is a bit involved but see Supplement of our latest papers for details

$$
\begin{aligned}
\hat{c}_n &= \underset{c}{\operatorname{argmax}} \max_{g} \mu_{cg}^T I_n^{(0)} && (17)\\
&= \underset{c}{\operatorname{argmax}} \max_{g} (\Lambda_g \mu_c)^T I_n^{(0)} && (18)\\
&= \underset{c}{\operatorname{argmax}} \max_{g^{(L:1)}} \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T \Lambda_{g^{(1)}}^T I_n^{(0)} && (19)\\
&= \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}} \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T (M_{a^{(1)}} \Lambda_{t^{(1)}}^T) I_n^{(0)} && (20)\\
&= \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}} \underbrace{\left(\mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T\right)}_{\equiv z^{(1)\downarrow T}} M_{a^{(1)}} \underbrace{\left(\Lambda_{t^{(1)}}^T I_n^{(0)}\right)}_{\equiv u_n^{(1)\uparrow}(t^{(1)})} && (21)\\
&= \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} \max_{t^{(1)}} \max_{a^{(1)}} z^{(1)\downarrow T} M_{a^{(1)}} u_n^{(1)\uparrow}(t^{(1)}) && (22)\\
&\overset{(a)}{=} \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} \max_{t^{(1)}} z^{(1)\downarrow T} M_{\hat{a}_n^{(1)}} u_n^{(1)\uparrow}(t^{(1)}) && (23)\\
&\overset{(b)}{=} \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} z^{(1)\downarrow T} \left(s^{(1)\downarrow} \odot \max_{t^{(1)}} s^{(1)\downarrow} \odot \left(M_{\hat{a}_n^{(1)}} u_n^{(1)\uparrow}(t^{(1)})\right)\right) && (24)\\
&\overset{(c)}{=} \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} z^{(1)\downarrow T} \left(s^{(1)\downarrow} \odot \max_{t^{(1)}} s^{(1)\downarrow} \odot \left(s^{(1)\downarrow} \odot \operatorname{ReLu}\left(s^{(1)\downarrow} \odot u_n^{(1)\uparrow}(t^{(1)})\right)\right)\right) && (25)\\
&\overset{(d)}{=} \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} z^{(1)\downarrow T} \underbrace{\left(s^{(1)\downarrow} \odot \operatorname{MaxPool}\left(\operatorname{ReLu}\left(\operatorname{diag}(s^{(1)\downarrow}) u_n^{(1)\uparrow}(\mathcal{T})\right)\right)\right)}_{\equiv I_n^{(1)}(s^{(1)\downarrow})} && (26)\\
&= \underset{c}{\operatorname{argmax}} \max_{g^{(L:2)}} \mu_c^T \Lambda_{g^{(L)}}^T \cdots \Lambda_{g^{(2)}}^T I_n^{(1)} && (27)
\end{aligned}
$$

Deriving/Explaining Other Architectures and Learning Algorithms in DRMM PoV

- Batch Normalization
- ResNets
- DropOut
- Weight Normalization
- Orthonormal Init, Glorot-Bengio Init

Missing-at-Random Inference in the DRMM yields DropOut

- **Neural Nets:** Prevent co-adaptation of feature detectors by randomly dropping out unit activations
- **Ensembles:** exponentially many models with lots of parameter sharing
- **DRM:** Equivalent to *Missing-at-Random Input Data* EM algorithm

A Unified Explanation of DropOut

- Input features I^{ℓ}_{nx} assumed missing at random: latent variables $s^{\ell}_{nx} \sim \mathrm{Bern}(p = \frac{1}{2})$
- Expected complete-data NLL has expectation over s^{ℓ}_{nx} which is approximated by sampling in the E-step $\Rightarrow$ randomly mask unit activations $\Rightarrow$ DropOut:

$$\max_{\theta} \mathbb{E}_S[Q(\theta)] \approx \max_{\theta} \sum_{\{s^{\ell}_{nx}\}} Q(\theta; \{s^{\ell}_{nx}\})$$

Inference in the *Pre-Conditioned* DRMM yields Deep ResNets

- **Neural Nets:** Make it easy to express/learn identity transformations
- **Optimization:** Hierarchical basis pre-conditioning
- **Ensembles:** not an ultra-deep net; instead ensemble of exponentially many shallower nets [Veit et al 2016]
- **DRM:** Coarse-to-fine generation: upsample coarse-grained image + add in fine-scale details via residuals (similar to Inverse Wavelet Transform) iff $a^\ell = 1 \Rightarrow$ exponentially many paths from coarser levels to finer levels in DRM

A Unified Explanation of ResNets in the DRMM

- Coarse-to-fine Generation: $\Lambda_{g^\ell} = 1 + \Delta_{g^\ell} = 1 + a^\ell \odot \Delta_{t^\ell} = 1$ iff $a^\ell = 0$
- Imposing this structure in the bottom-up pass:

$$\max_g \Lambda_{g^\ell}^T I_n = \max_g (1 + \Delta_{g^\ell}^T) I_n = I_n + \max_{t,a} a^\ell \odot \mathrm{W}_{t^\ell} I_n = I_n + \mathscr{H}(I_n; \theta_{\mathrm{Res}})$$

Inference in the *Evolutionary* DRMM yields Decision Trees

- **Surprise:** Variant of the DRMM with a hierarchy of categories (e.g. tree of life) yields JMAP inference DP algorithm that is decision tree

- **Evolutionary DRMM** generation process:

$$
\begin{aligned}
\mu_{c^{(L)}g} &= \Lambda_g \mu_{c^{(L)}} \equiv \Lambda_{g^{(1)}} \cdots \Lambda_{g^{(L)}} \cdot \mu_{c^{(L)}} \\
&\equiv \mu_{c^{(L)}} + \alpha_{g^{(L)}} + \cdots + \alpha_{g^{(1)}}, \quad g = \{g^{(\ell)}\}_{\ell=1}^{L} \\
I &\sim \mathcal{N}(\mu_{c^{(L)}g}, \sigma^2 1_D) \in \mathbb{R}^D.
\end{aligned}
$$

$$
\mu_{c^{(\ell)}} = \Lambda_{g^{(\ell+1)}} \cdot \mu_{c^{(\ell+1)}} = \mu_{c^{(\ell+1)}} + \alpha_{g^{(\ell+1)}}
$$

- **Proof:** push max to the right through the sum of per-species templates

$$
\begin{aligned}
\hat{c}^{(L)}(I) &= \underset{c^{(L)} \in \mathcal{C}^L}{\operatorname{argmax}} \max_{g \in \mathcal{G}} \langle \mu_{c^{(L)}} + \alpha_{g^{(L)}} + \cdots + \alpha_{g^{(1)}} | I \rangle \\
&= \underset{c^{(L)} \in \mathcal{C}^L}{\operatorname{argmax}} \max_{g^{(1)} \in \mathcal{G}^1} \cdots \max_{g^{(L-1)} \in \mathcal{G}^{L-1}} \langle \underbrace{\mu_{c^{(L)}} + \alpha_{g^{(L)*}}}_{\equiv \mu_{c^{(L-1)}}} + \cdots + \alpha_{g^{(1)}} | I \rangle \\
&\cdots \\
&\equiv \underset{c^{(L)} \in \mathcal{C}^L}{\operatorname{argmax}} \langle \mu_{c^{(L)}g^*} | I \rangle.
\end{aligned}
$$

$$
\begin{aligned}
g^*_{c^{(\ell+1)}} &\equiv \underset{g^{(\ell+1)} \in \mathcal{G}^{\ell+1}}{\operatorname{argmax}} \langle \underbrace{\mu_{c^{(\ell+1)}g^{(\ell+1)}}}_{\equiv W^{(\ell+1)}} | I \rangle \\
&\equiv \text{ChooseChild}(\text{Filter}(I)).
\end{aligned}
$$

The ***Dynamic Programming*** Algorithm Interpretation of Convnets

Deep Convnets: A Dynamic Programming (DP) Interpretation

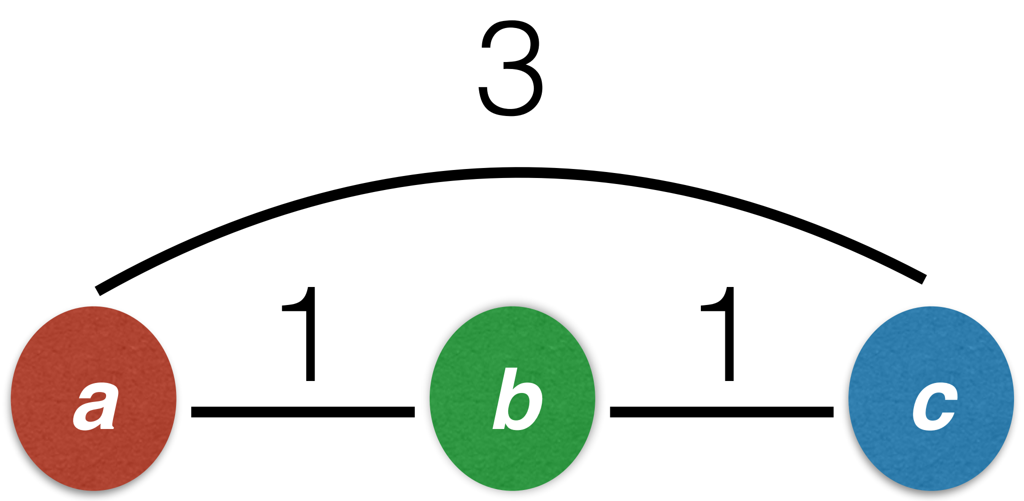
- **New Interpretation:** Convnets are a DP algorithm for finding the memory of maximum similarity (min. distance) to the input.
- Mathematically Equivalent, Psychologically Inequivalent
- What implications does this new perspective have for understanding Convnets at a *mechanistic* level?

Review of DP with an Example: The Shortest Path Problem

- **Shortest Path problem:** Find the shortest path from a source to destination node in a directed graph.
- **Problem:** Exponentially many paths to check
- **Insight:** Exploit self-similarity of the optimal path to design algorithm that optimally re-uses past computation
- **Question:** Can you skip all the recursion steps of the DP?

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <= ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Intuition Behind DP: Shortest Path Problem



Initialize:

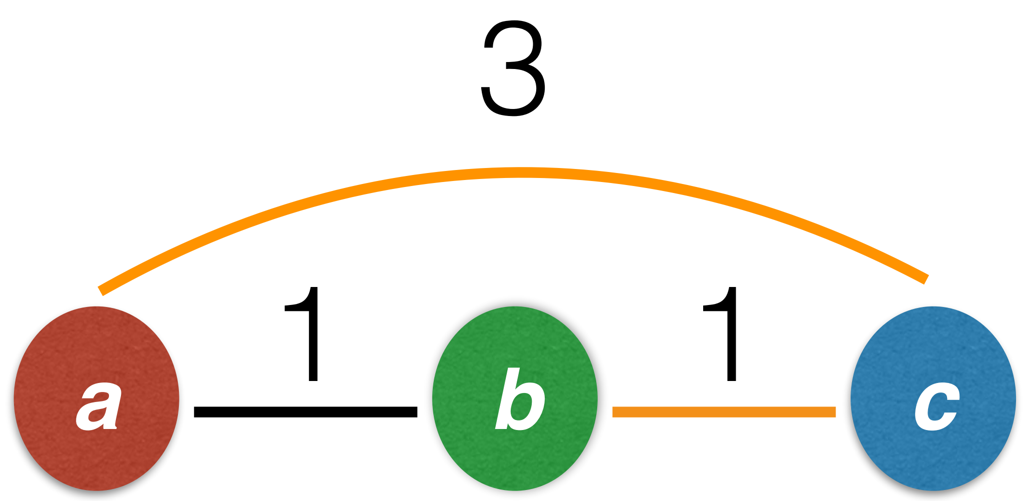
Node *a*		
len	dist	next
0	Inf	—
1		
2		

Node *b*		
len	dist	next
0	Inf	—
1		
2		

Node *c*		
len	dist	next
0	0	—
1		
2		

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Intuition Behind DP: Shortest Path Problem



DP Update 1: Send dist. info and decide best **active** paths

Node *a*		
len	**dist**	**next**
0	Inf	—
1	**3**	**c**
2		

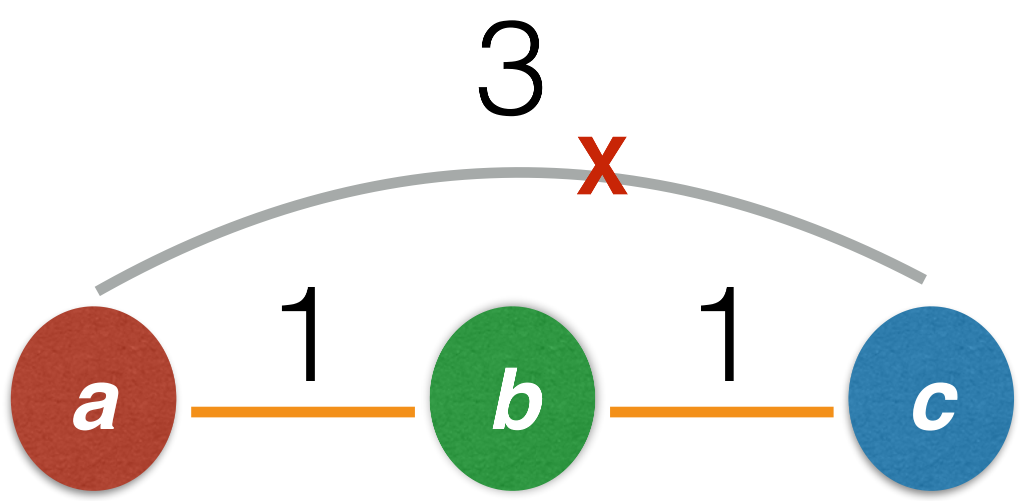
Node *b*		
len	**dist**	**next**
0	Inf	—
1	**1**	**c**
2		

Node *c*		
len	**dist**	**next**
0	0	—
1	0	—
2		

- **Observations about DP Algo (that generalize to all DPs):**
- Each hypothesis claims "This is the best (sub)path I've found thus far. But I'm not sure that it's a part of the global optimal path."

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Intuition Behind DP: Shortest Path Problem



DP Update 2: Send dist. info and decide best **active** paths

Node *a*		
len	**dist**	**next**
0	Inf	—
1	3	c
2	**1**	**b**

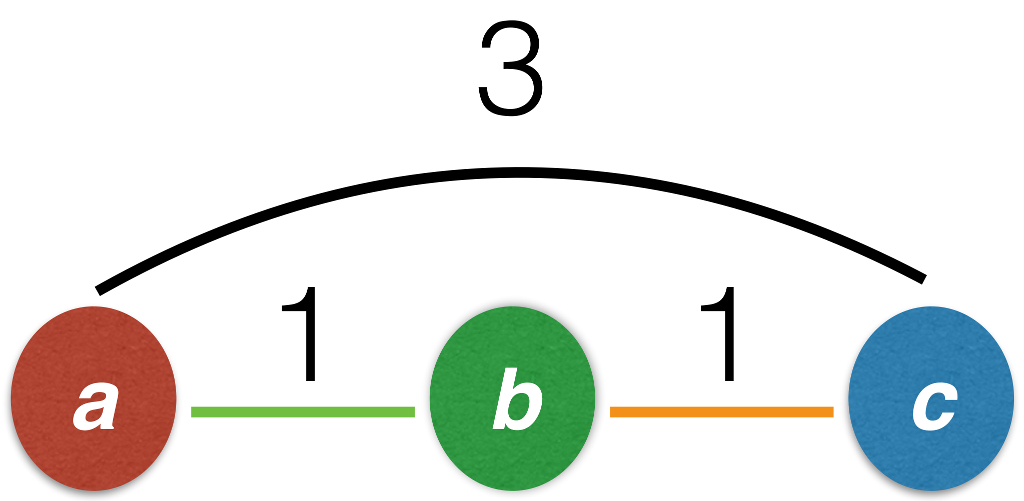
Node *b*		
len	**dist**	**next**
0	Inf	—
1	1	c
2	**1**	**c**

Node *c*		
len	**dist**	**next**
0	0	—
1	0	—
2	0	—

- **Observations about DP Algo (that generalize to all DPs):**
- Each hypothesis claims "This is the best (sub)path I've found thus far. But I'm not sure that it's a part of the global optimal path."
- Early hypotheses can be superseded by others much later on. When a hypothesis is superseded, its never used again. Sub-optimal paths can thus be inactivated much later on than when they were first constructed.

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Intuition Behind DP: Shortest Path Problem



Backtrace 1:

Node *a*		
len	**dist**	**next**
0	Inf	—
1	3	c
2	1	b

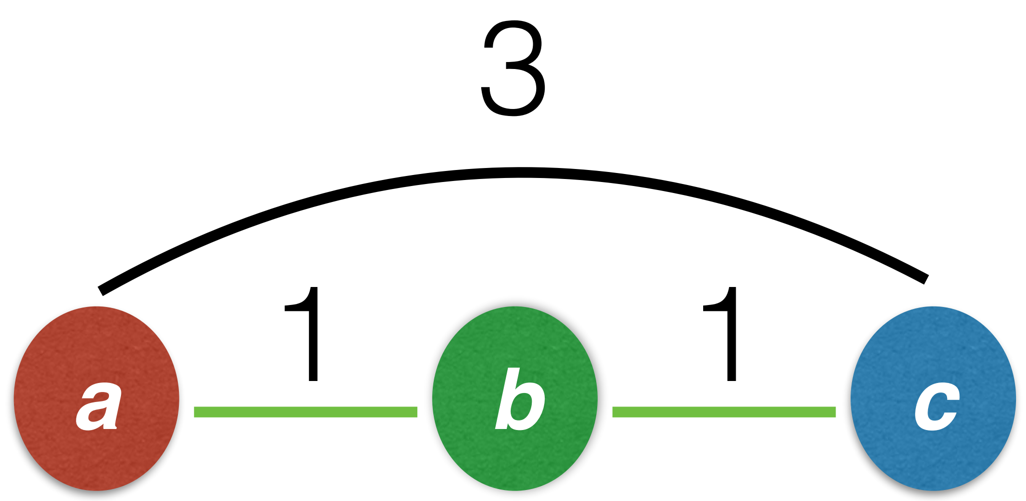
Node *b*		
len	**dist**	**next**
0	Inf	—
1	1	c
2	1	c

Node *c*		
len	**dist**	**next**
0	0	—
1	0	—
2	0	—

- **Observations about DP Algo (that generalize to all DPs):**
- Each hypothesis claims "This is the best (sub)path I've found thus far. But I'm not sure that it's a part of the global optimal path."
- Early hypotheses can be superseded by others much later on. When a hypothesis is superseded, its never used again. Sub-optimal paths can thus be inactivated much later on than when they were first constructed.

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Intuition Behind DP: Shortest Path Problem



Backtrace 2:

Node *a*		
len	**dist**	**next**
0	Inf	—
1	3	c
2	1	b

Node *b*		
len	**dist**	**next**
0	Inf	—
1	1	c
2	1	c

Node *c*		
len	**dist**	**next**
0	0	—
1	0	—
2	0	—

- **Observations about DP Algo (that generalize to all DPs):**
- Each hypothesis claims "This is the best (sub)path I've found thus far. But I'm not sure that it's a part of the global optimal path."
- Early hypotheses can be superseded by others much later on. When a hypothesis is superseded, its never used again. Sub-optimal paths can thus be inactivated much later on than when they were first constructed.
- When global optima is found, we can reconstruct the optimal hypothesis via backtracing.

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

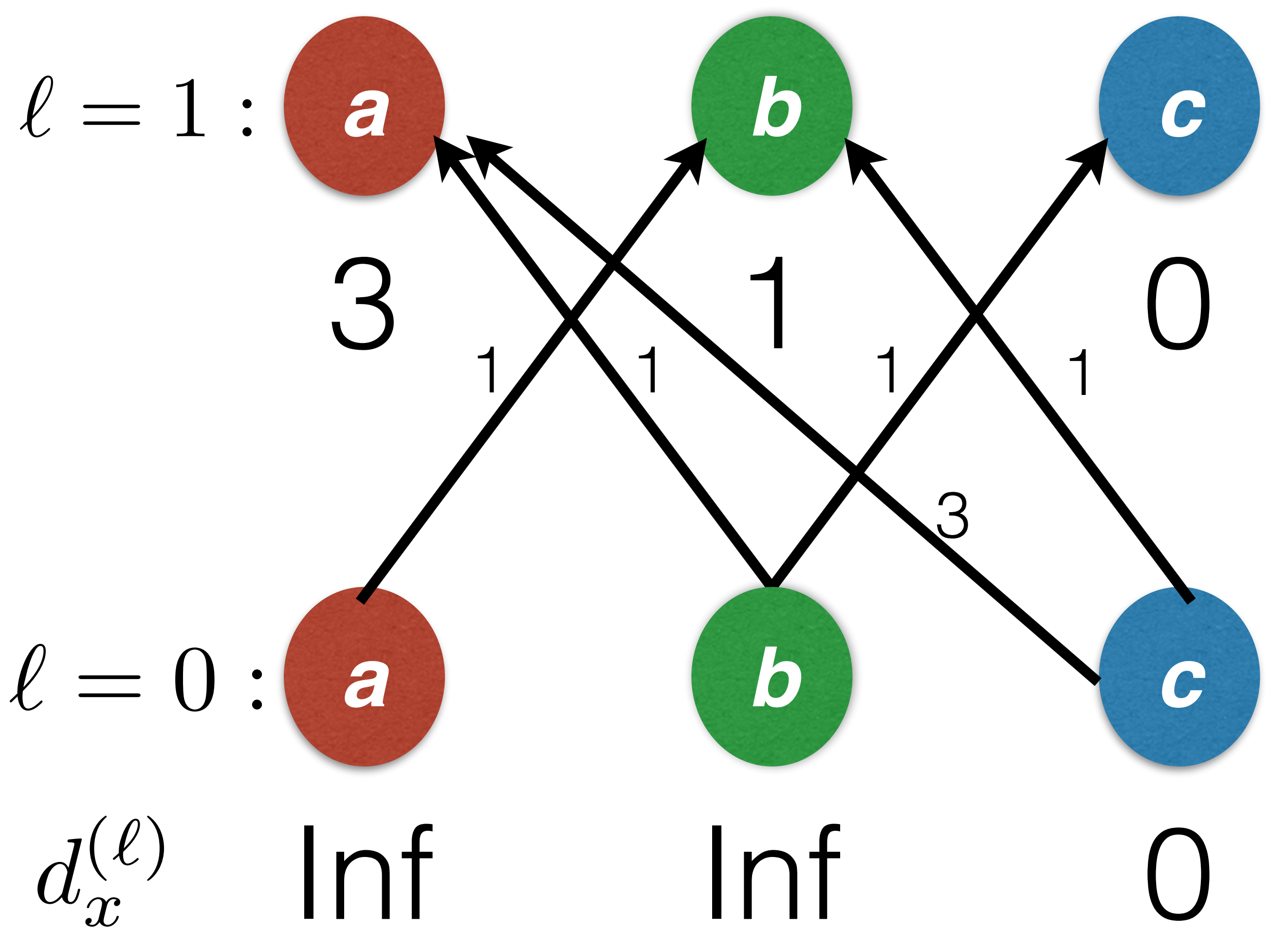
DP Initialize:

$\ell = 0:$	a	b	c
$d_x^{(\ell)}$	Inf	Inf	0

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y \in \mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

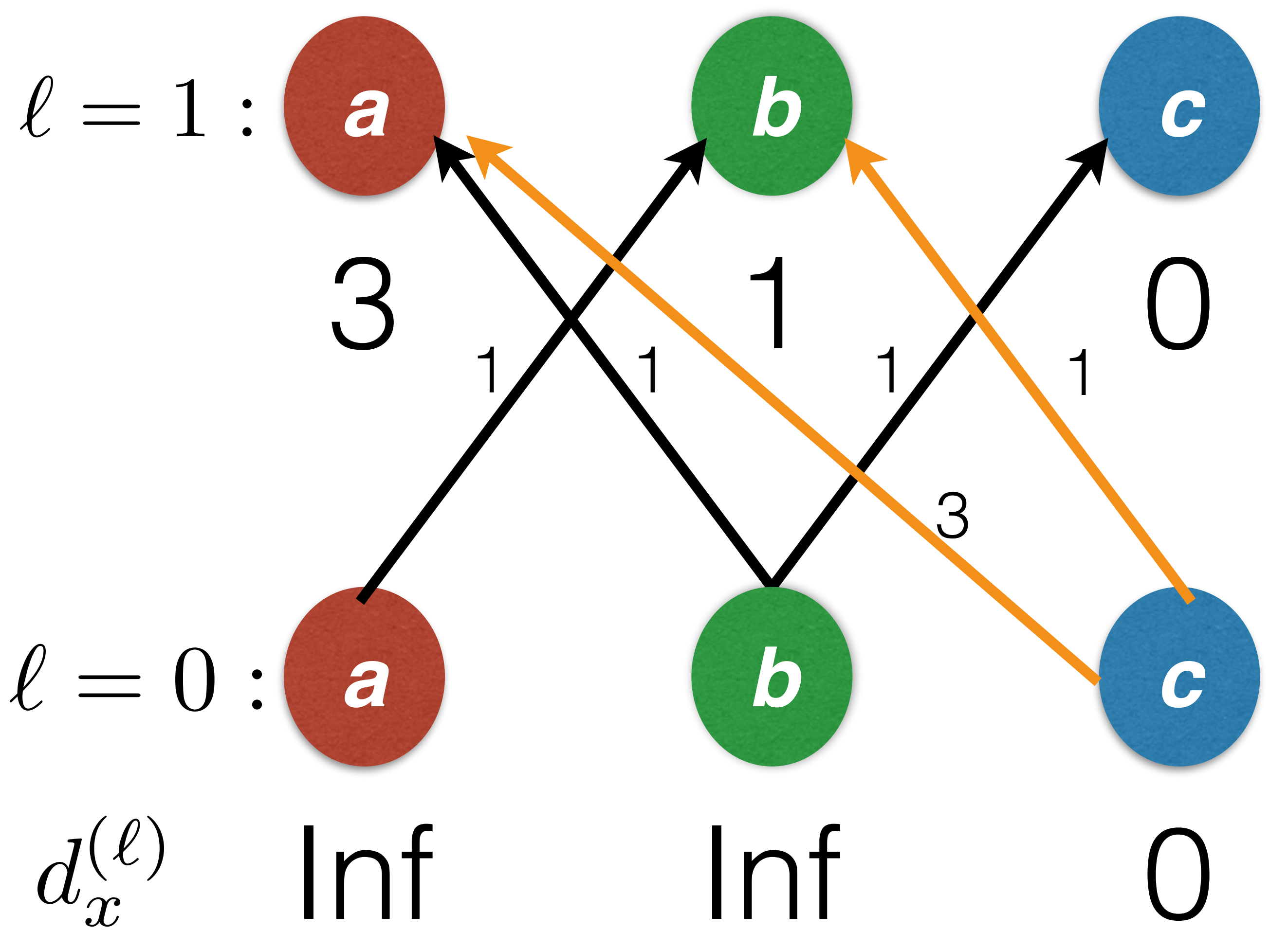
DP Update 1: Propagate distance info



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y \in \mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

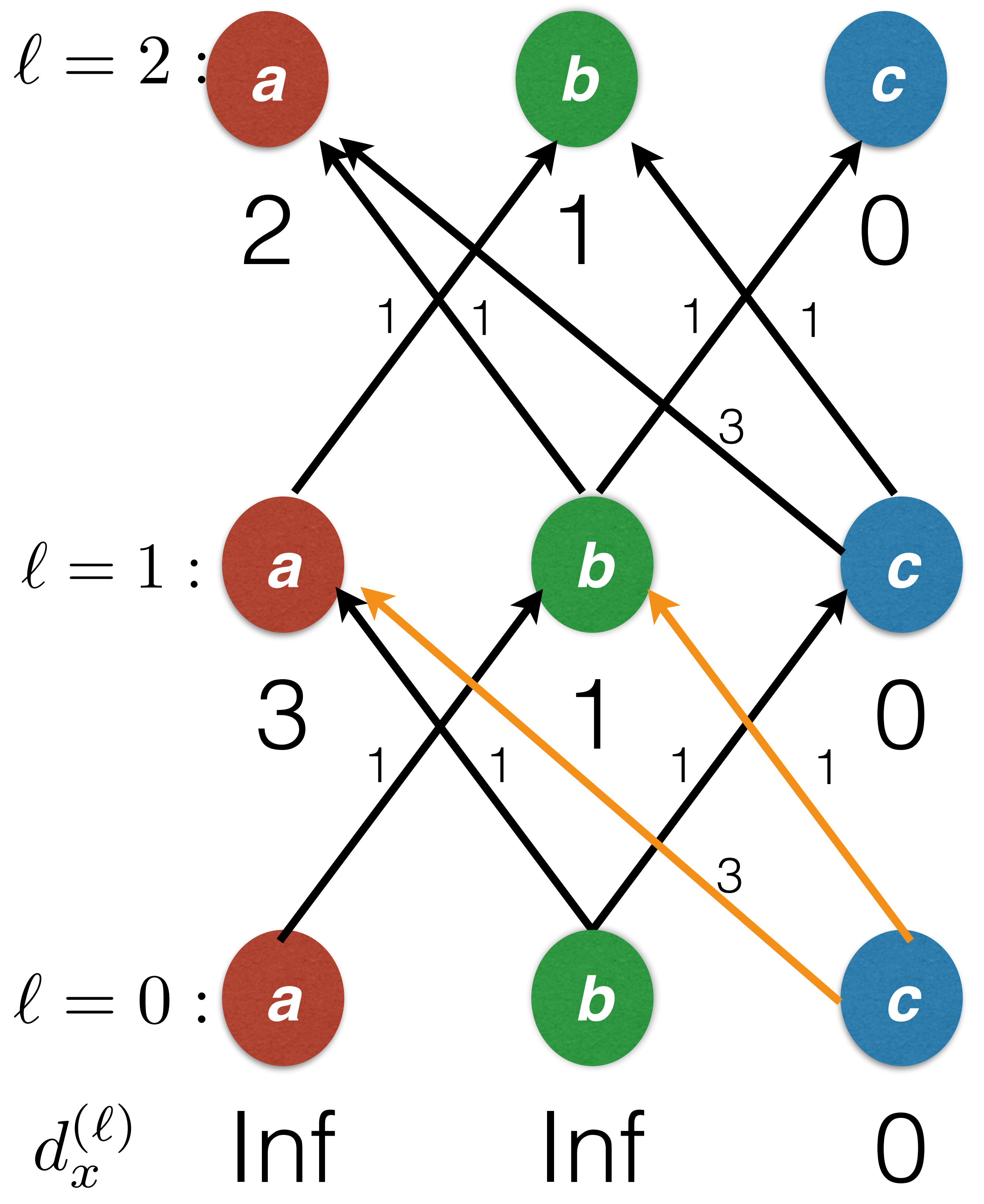
DP Update 1: Decide best (**active**) paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

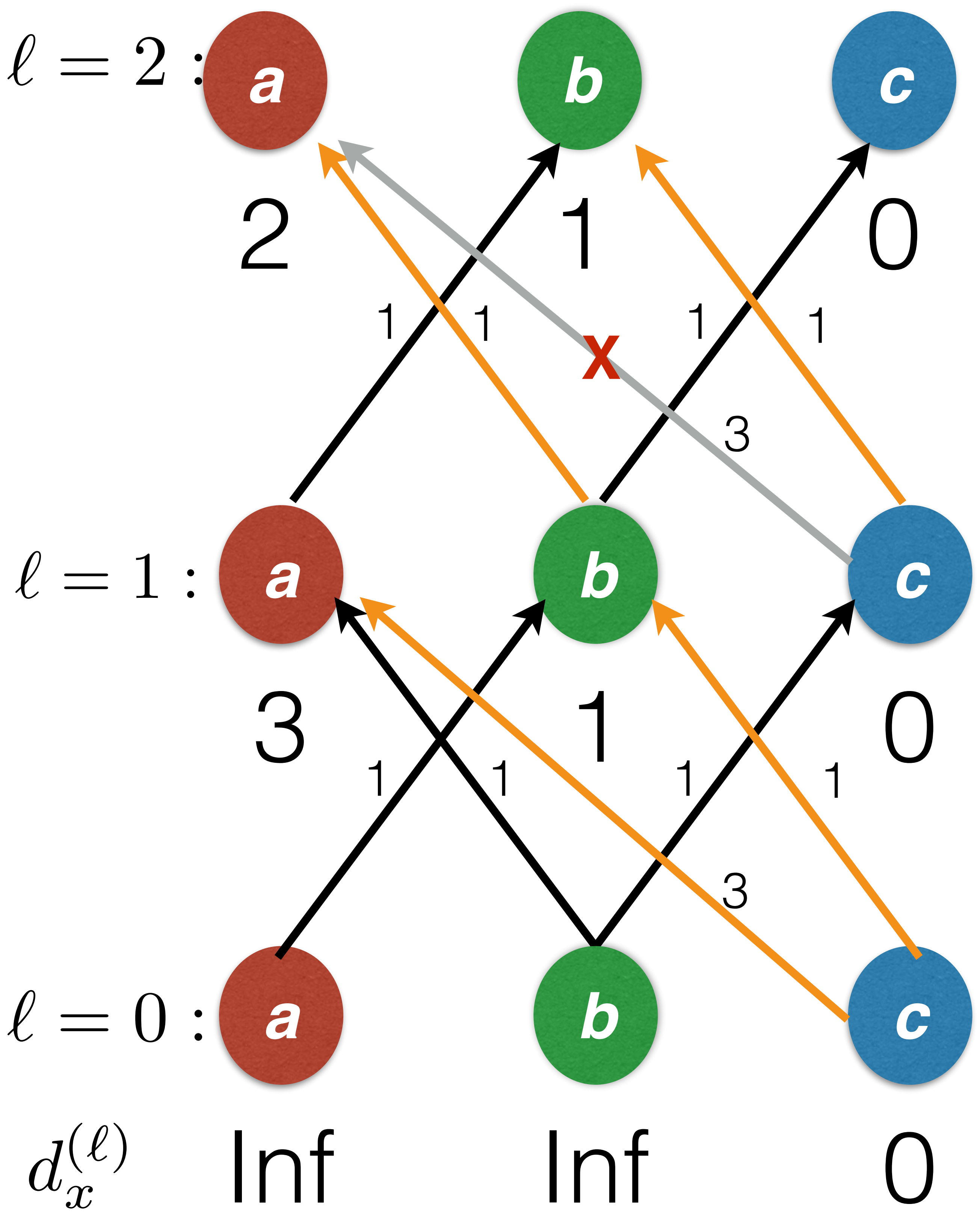
DP Update 2: Propagate distance info



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y \in \mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

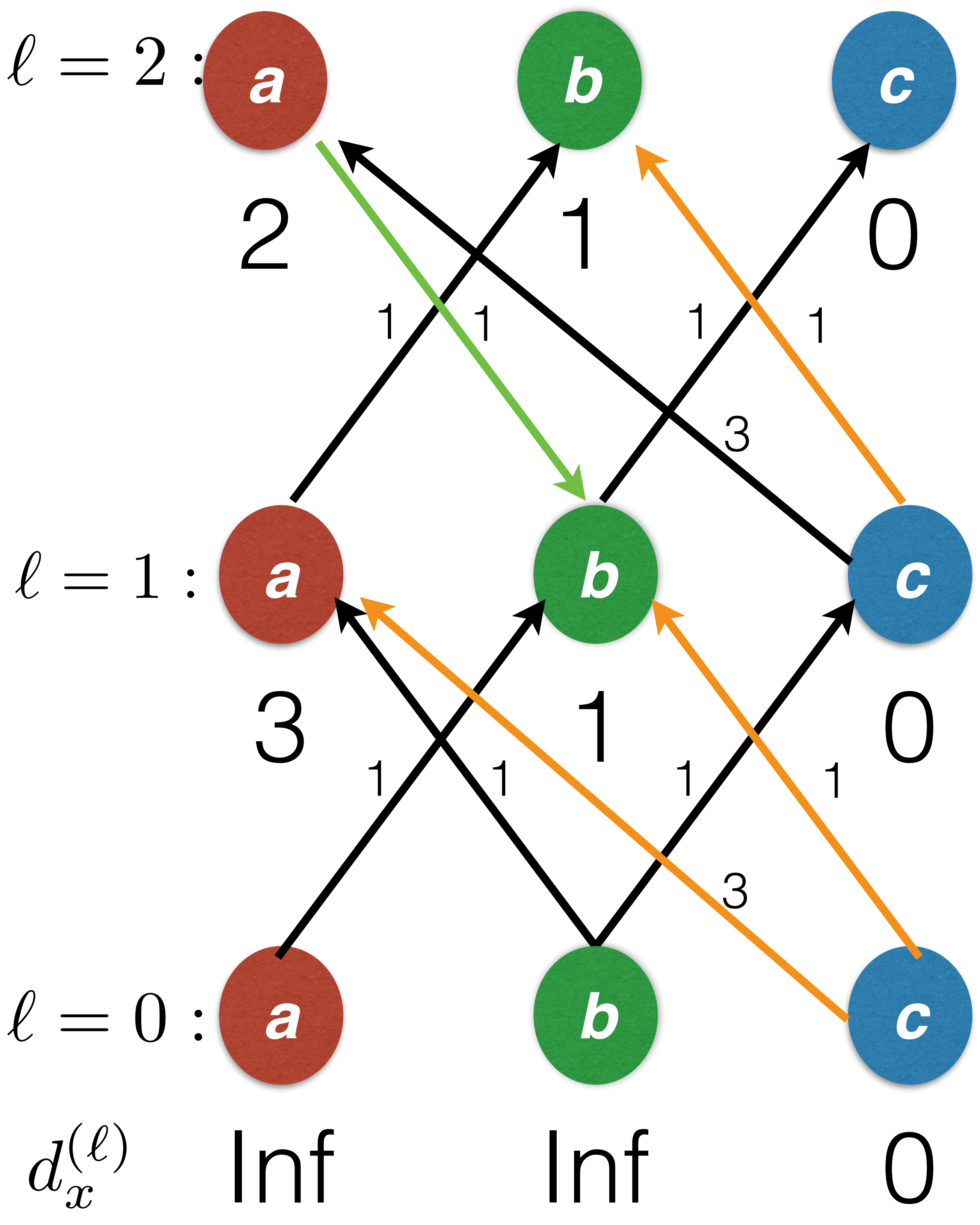
DP Update 2: Update best (**active**) paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

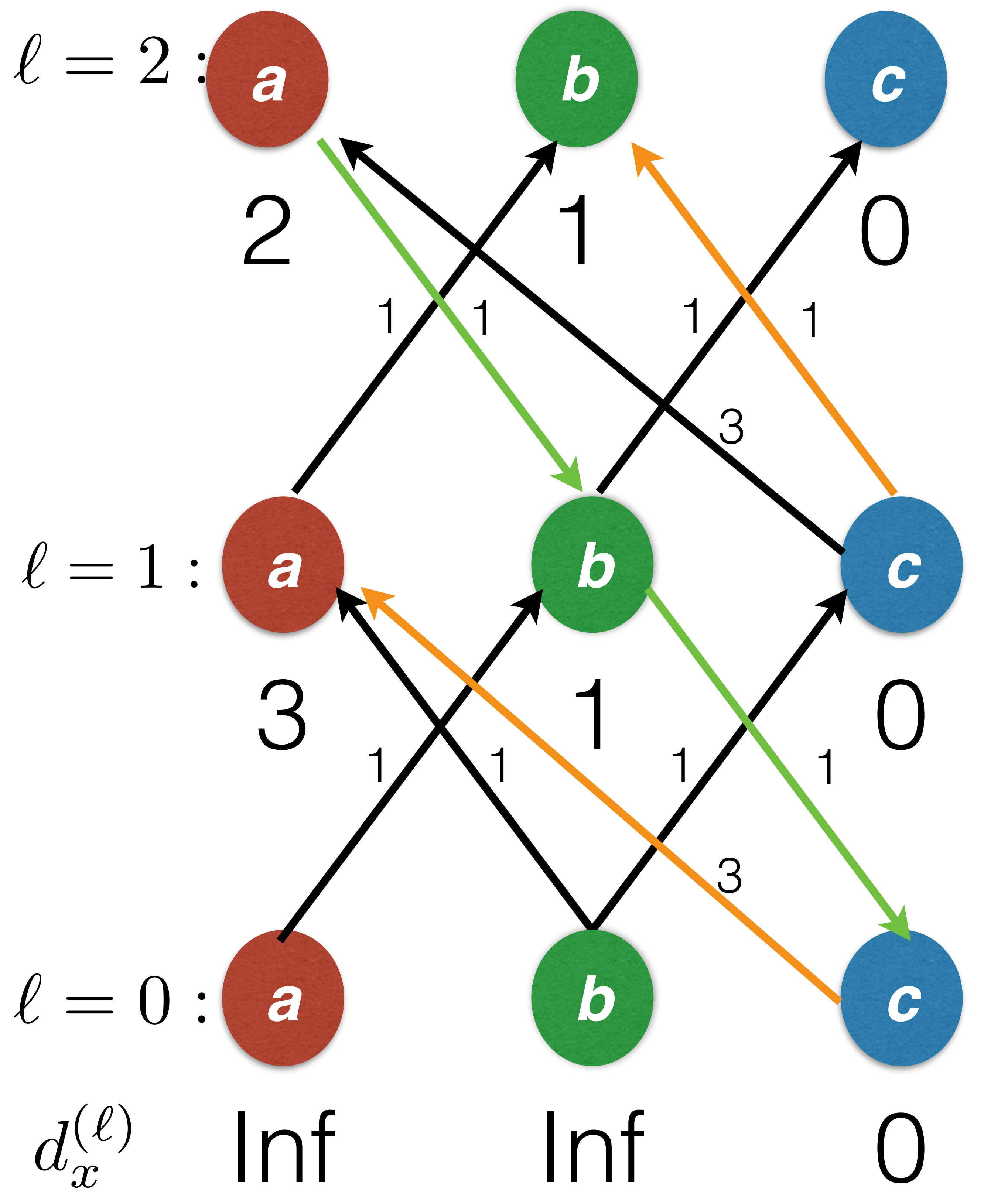
Backtrace: Reconstruct **best** (**active**) paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Unrolling the DP in Time

Backtrace: Reconstruct **best** (**active**) paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y \in \mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

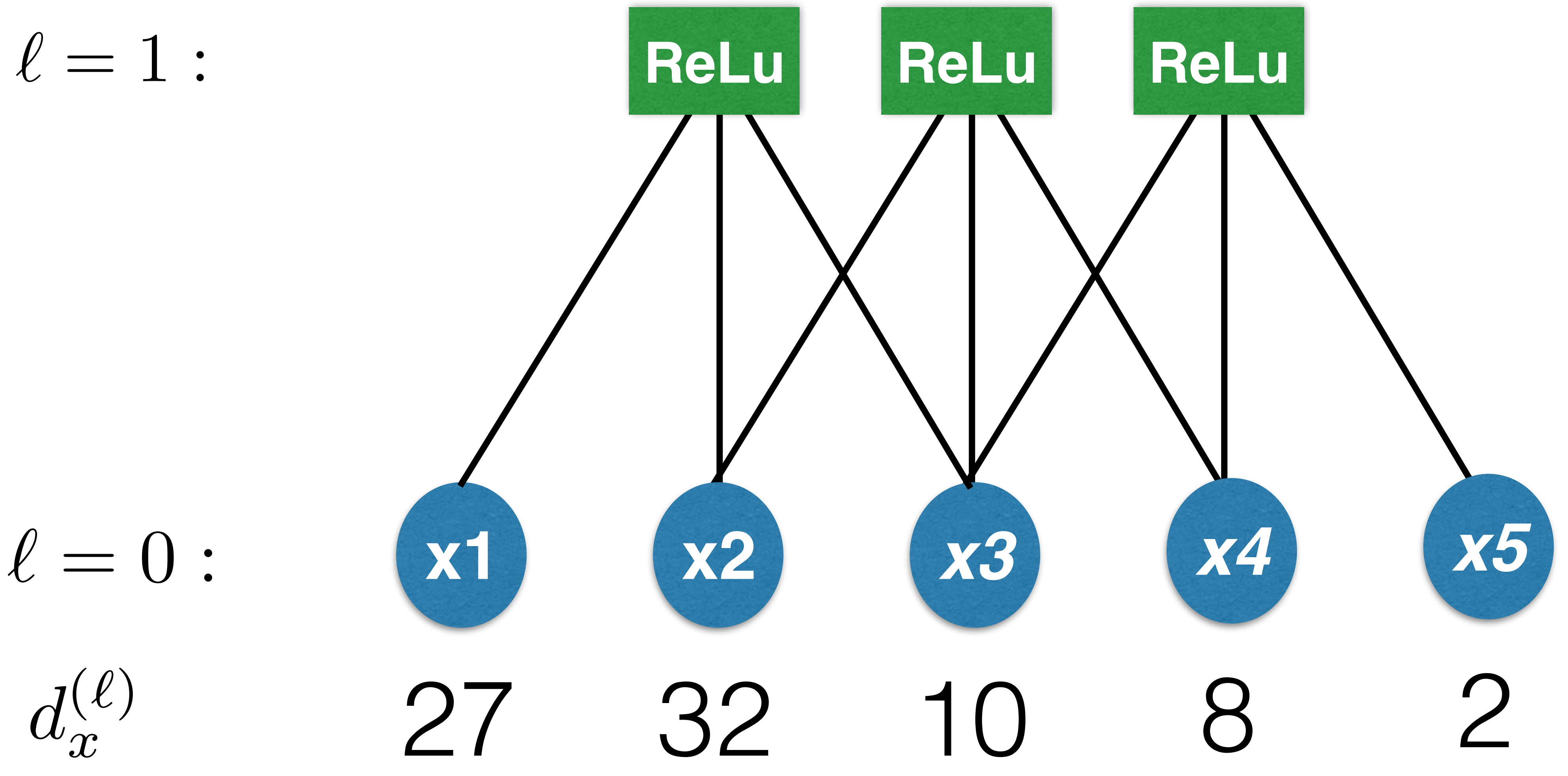
Initialize: Setup all input pixels

$\ell = 0:$	x1	x2	x3	x4	x5
$d_x^{(\ell)}$	27	32	10	8	2

- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <= ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

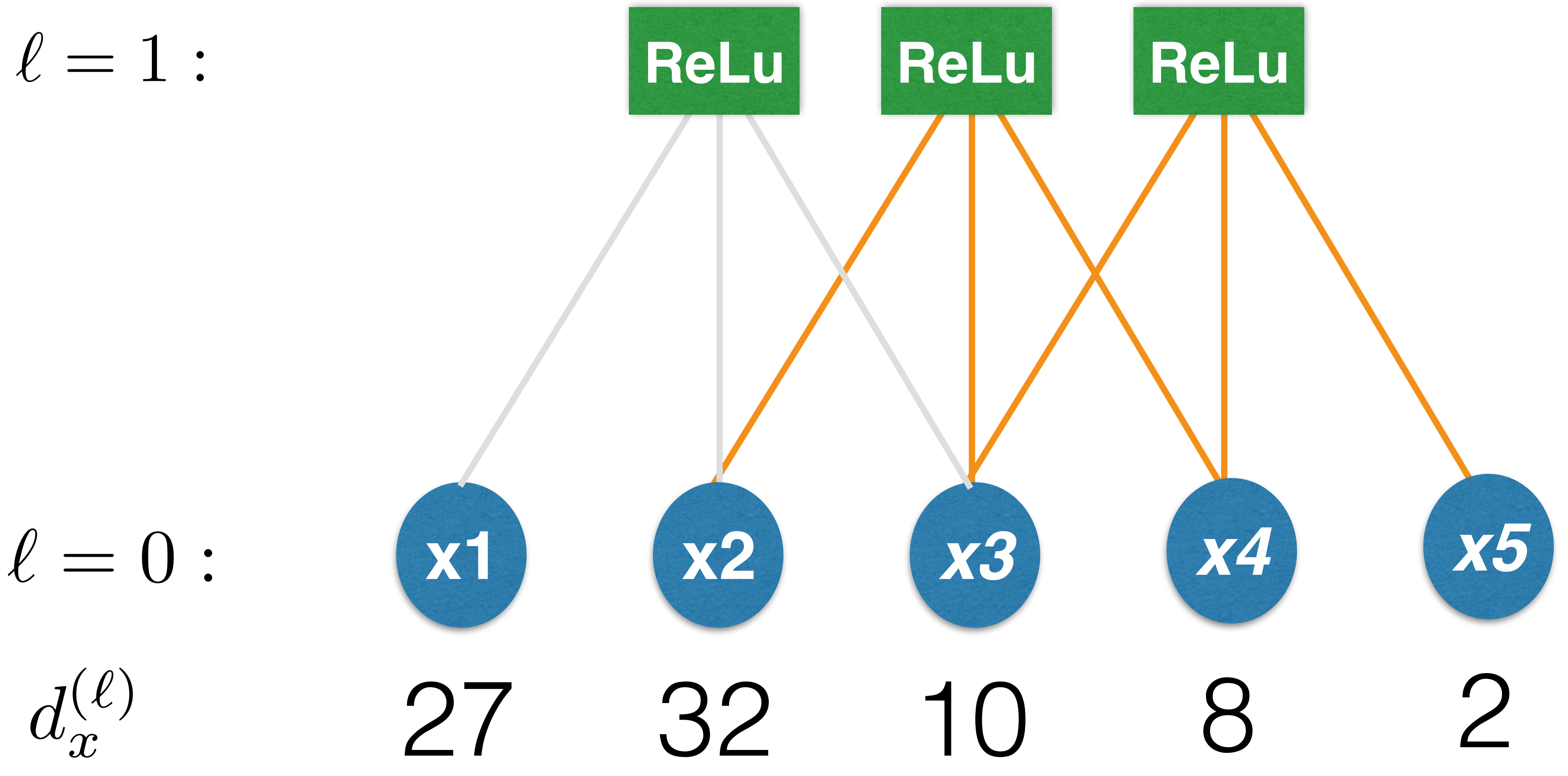
DP Update 1: Send distance info forward



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

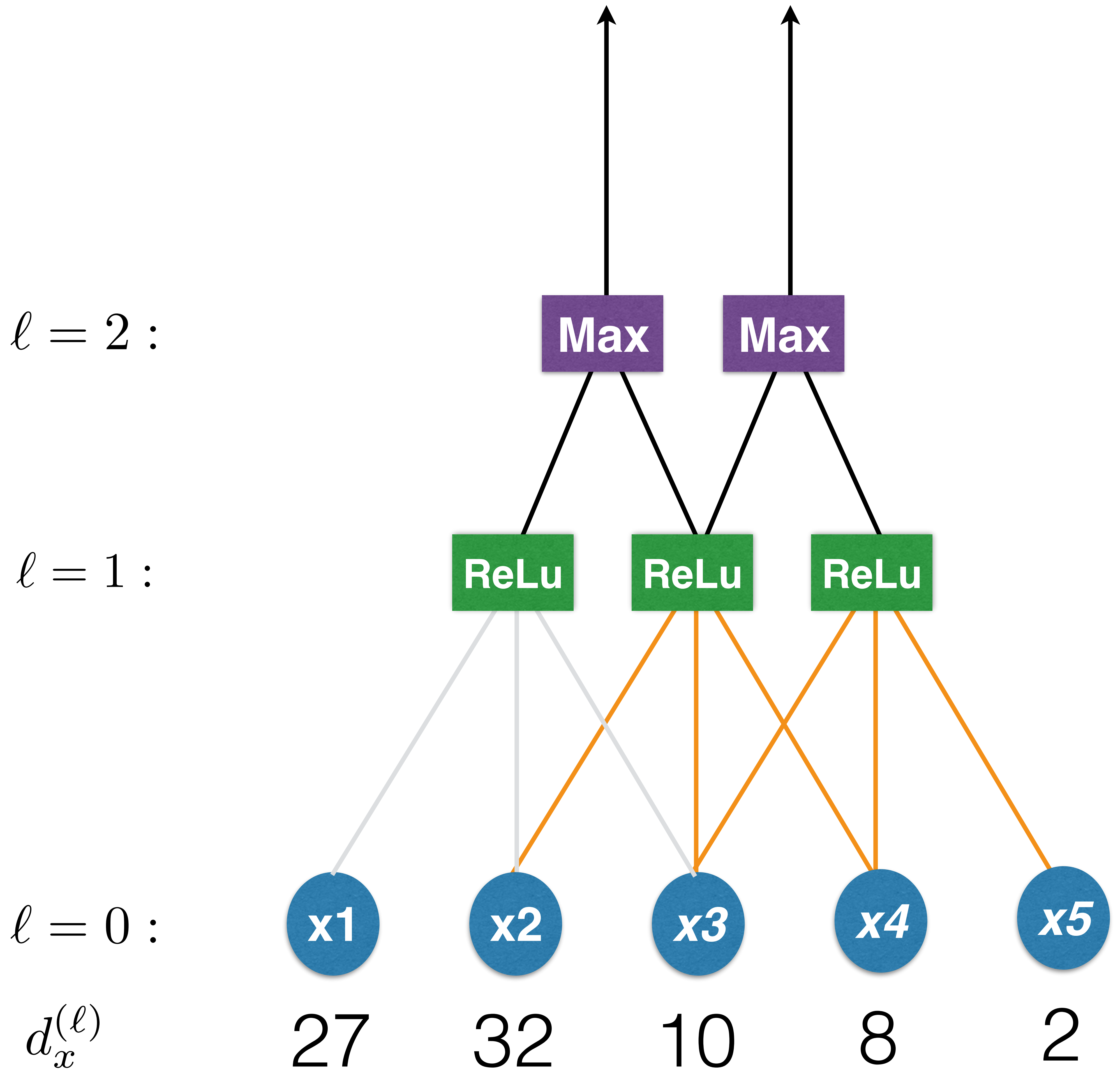
DP Update 1: Update **active** paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

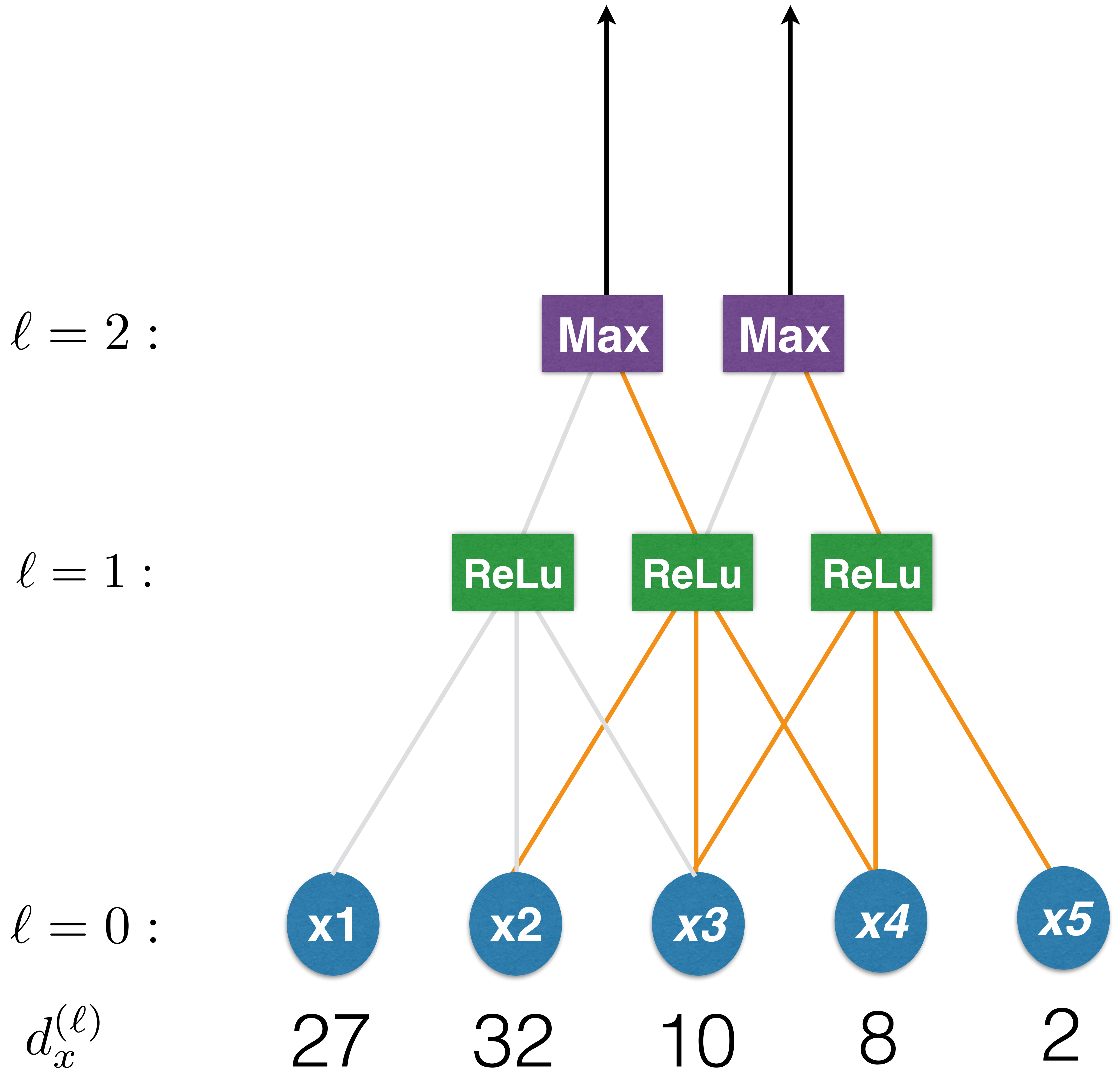
DP Update 2: Send distance info forward



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

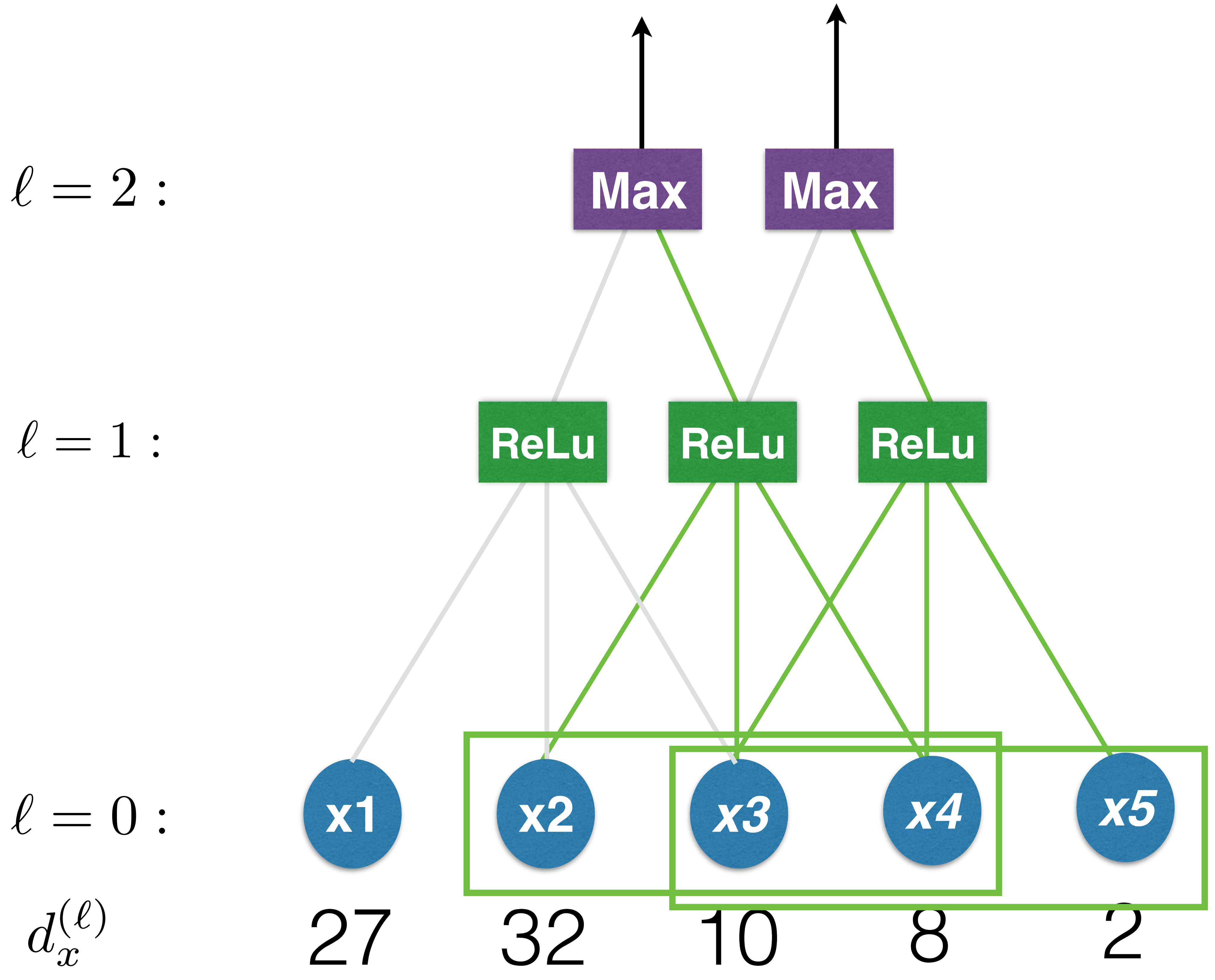
DP Update 2: Update best **active** paths



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Connection between Unrolled DP and Deep Convnets

Backtrace: reconstruct **best active** paths
i.e. the ***task-relevant patches***



- **Solution:** Every DP problem has the same basic ingredients:
  - **Cost:** minimize dist $\min_{\pi} \sum_{(u,v)\in\pi} d_{uv}$
  - **Recursion Variable:** path length ℓ
  - **Local Cost:** min distance from node to dest with <=ℓ edges, $d_x^{(\ell)}$
  - **Recursion Relation:** $d_{xz}^{(\ell)} = \min_{y\in\mathcal{N}_x} \{d_{xy}^{(1)} + d_{yz}^{(\ell-1)}\}$
  - **Local-to-Global:** iterate RR until converges
  - **Reconstruct Minimizer:** In DP tables, keep track of which node is next on minimal path (so far). Then **Backtrace**.

Implications of the DP Interpretation

- **Non-convex but Tractable hard E-step:** Convnets are an efficient DP algorithm (unrolled in time) for the JMAP inference task. The objective is *non-convex and yet still tractable* due to the DP.
- **Receptive fields:** RFs are "derived" as necessary recursion variables for a DP algorithm
- **New Interpretation of Firing Neurons:** Firing means *"I believe there are task-relevant pixels in my RF"*
- **Top-Down Inference/Reconstruction:** Under NN conditions, no need to do a top-down pass for inference. Just trace active paths back to input pixels!
- **Only Active Paths Matter:** Only the sparse set of optimal active paths matter for the final decision of the Convnet ==> *Deep Sparse Path Coding*

DP Interpretation: New Explanations & Testable Predictions

Using the DP Interpretation, we can explain/predict many empirical observations about Convnets:

- **Lots of False Positives in Early Layers:** Neurons in early layers have small RFs so not sure which features will ultimately be task-relevant —> many will fire —> lots of false positives —> should see high saliency for task-irrelevant pixels e.g. in image background.

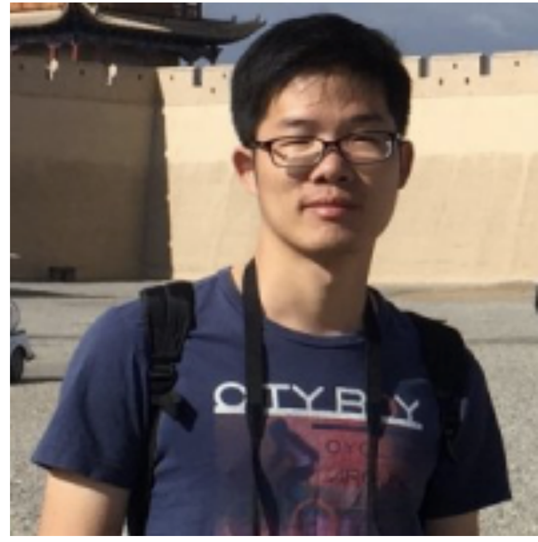
- **Role of each Layer:** Since recursion variable = RF size, each subsequent layer will effectively examine larger regions of input, and according to DP, will try to keep true positives (**selectivity**) and filter out more false positives (**invariance**)

  - *Corollary:* Layer-by-Layer Saliency maps should become increasingly invariant to task-irrelevant pixels e.g. background. At each layer *L*, trained vs random weights will show the value-add of the *L*-th layer in terms of filtering out false positives.

- **Depth is Necessary:** We have a qualitatively new reason for depth — its not directly about expressive power (e.g. No-Flattening Theorems). Instead its about the recursion variable in the DP algorithm i.e. its about filtering out some fraction of the false positives at each layer.

Saliency Maps show Selectivity and Invariance are Built up over Layers

Yang Zhang



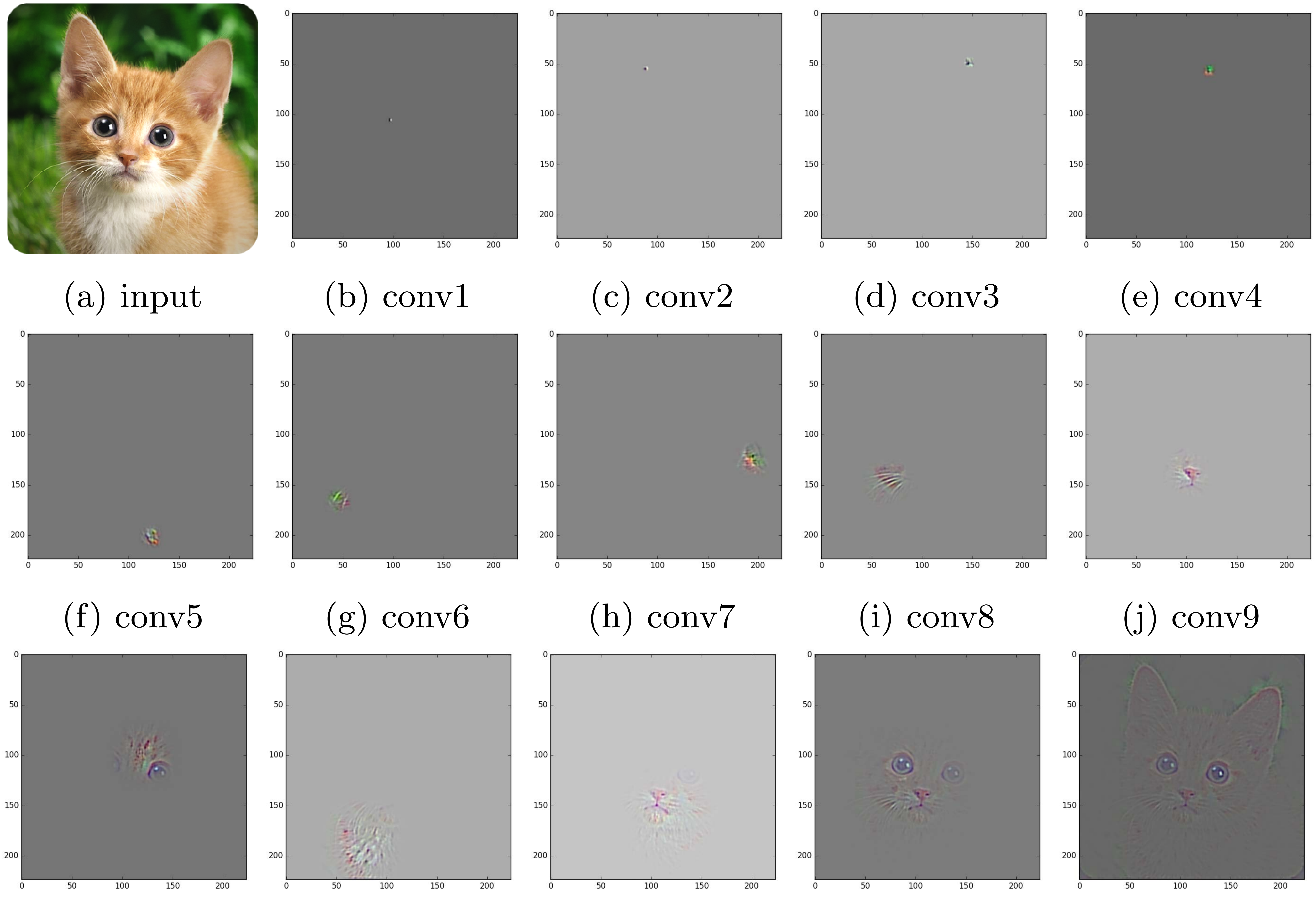
Weili Nie

Question: How do Convnets build up invariance to background?

Experiment: Visualize saliency maps for active neurons at each layer.

Observations:

- Neurons in early layers are <u>*selective for all*</u> detectable features in input, including background.
- Neurons in deeper layers are <u>*selective only*</u> for small subset of input pixels (those useful for discriminating class)
- Neurons in deep layers are <u>*invariant*</u> to (almost all) irrelevant pixels e.g. background and below the neck.



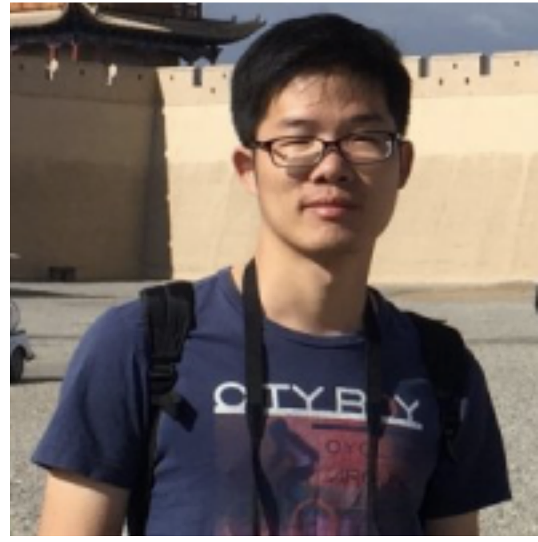
(a) input (b) conv1 (c) conv2 (d) conv3 (e) conv4

(f) conv5 (g) conv6 (h) conv7 (i) conv8 (j) conv9

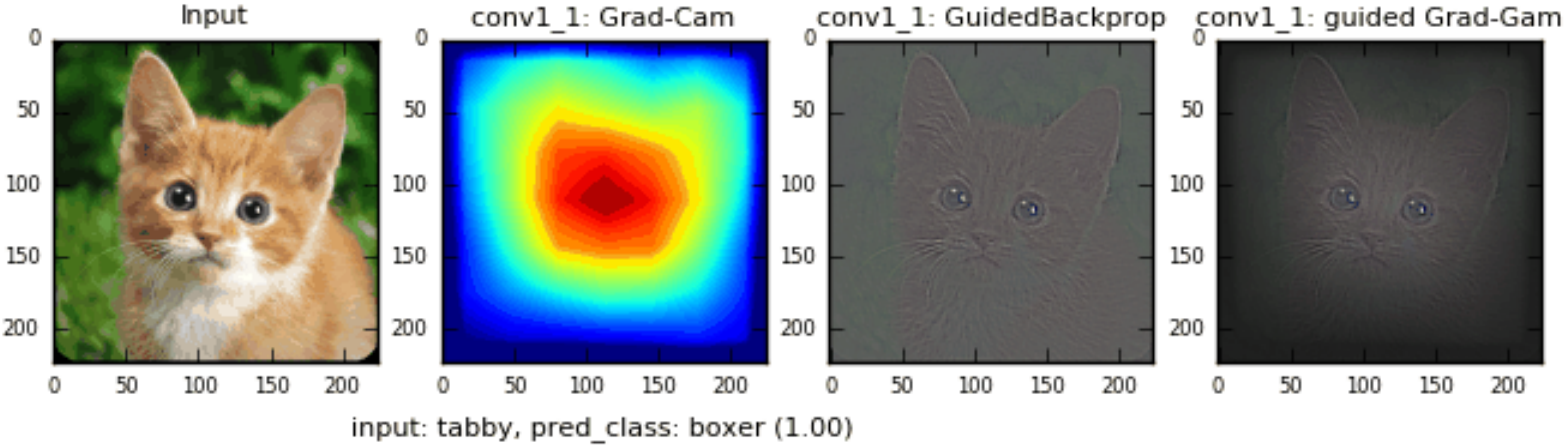
(k) conv10 (l) conv1 (m) conv1 (n) conv13 (o) fc1

Yang Zhang

Guided Backprop Saliency Maps show False Positives being Filtered Layer-by-Layer



Weili Nie



Conclusion: DP interpretation explains how neurons become increasingly invariant to task-irrelevant pixels (e.g. green background) while maintaining selectivity.

Learning

Learning via Backpropagation: A Hard EM Interpretation

Algorithm 1 Hard EM and EG Algorithms for the DRMM

E-step: $\hat{c}_n, \hat{g}_n = \underset{c,g}{\operatorname{argmax}}\ \gamma_{ncg}$

Feedforward Convnet

G-step: $\Delta\hat{\Lambda}_{g(\ell)} \propto \nabla_{\Lambda_{g(\ell)}} \ell_{DRMM}(\theta)$

Backpropagation

Implications:

- **E-step:** Non-convex yet still tractable optimization (due to DP Algorithm)
- **M/G-step:** Non-convex yet still tractable optimization (due to DP Algorithm aka Backprop). For linear NNs, *every local minima is a global minima* [Lu, Kawaguchi 2017]

Hard EM Interpretation yields New *Derivative-Free* M-step

Algorithm 1 Hard EM and EG Algorithms for the DRMM

E-step:	$\hat{c}_n, \hat{g}_n = \underset{c,g}{\operatorname{argmax}}\ \gamma_{ncg}$	**Feedforward Convnet**
M-step:	$\hat{\Lambda}_{g^{(\ell)}} = \underbrace{\mathrm{GLS}}\left(I_n^{(\ell-1)} \sim \hat{z}_n^{(\ell)} \mid g^{(\ell)} = \hat{g}_n^{(\ell)}\right) \ \forall g^{(\ell)}$	**New Derivative-Free Learning Rule**
G-step:	$\Delta\hat{\Lambda}_{g^{(\ell)}} \propto \nabla_{\Lambda_{g^{(\ell)}}} \ell_{DRMM}(\theta)$	**Backpropagation**

Hard EM Interpretation yields New *Derivative-Free* M-step

Algorithm 1 Hard EM and EG Algorithms for the DRMM

E-step: $\hat{c}_n, \hat{g}_n = \underset{c,g}{\operatorname{argmax}}\ \gamma_{ncg}$

M-step: $\hat{\Lambda}_{g^{(\ell)}} = \underbrace{\mathrm{GLS}}\left(I_n^{(\ell-1)} \sim \hat{z}_n^{(\ell)} \mid g^{(\ell)} = \hat{g}_n^{(\ell)}\right)\ \forall g^{(\ell)}$

G-step: $\Delta\hat{\Lambda}_{g^{(\ell)}} \propto \nabla_{\Lambda_{g^{(\ell)}}} \ell_{DRMM}(\theta)$

Feedforward Convnet

New Derivative-Free Learning Rule

Backpropagation

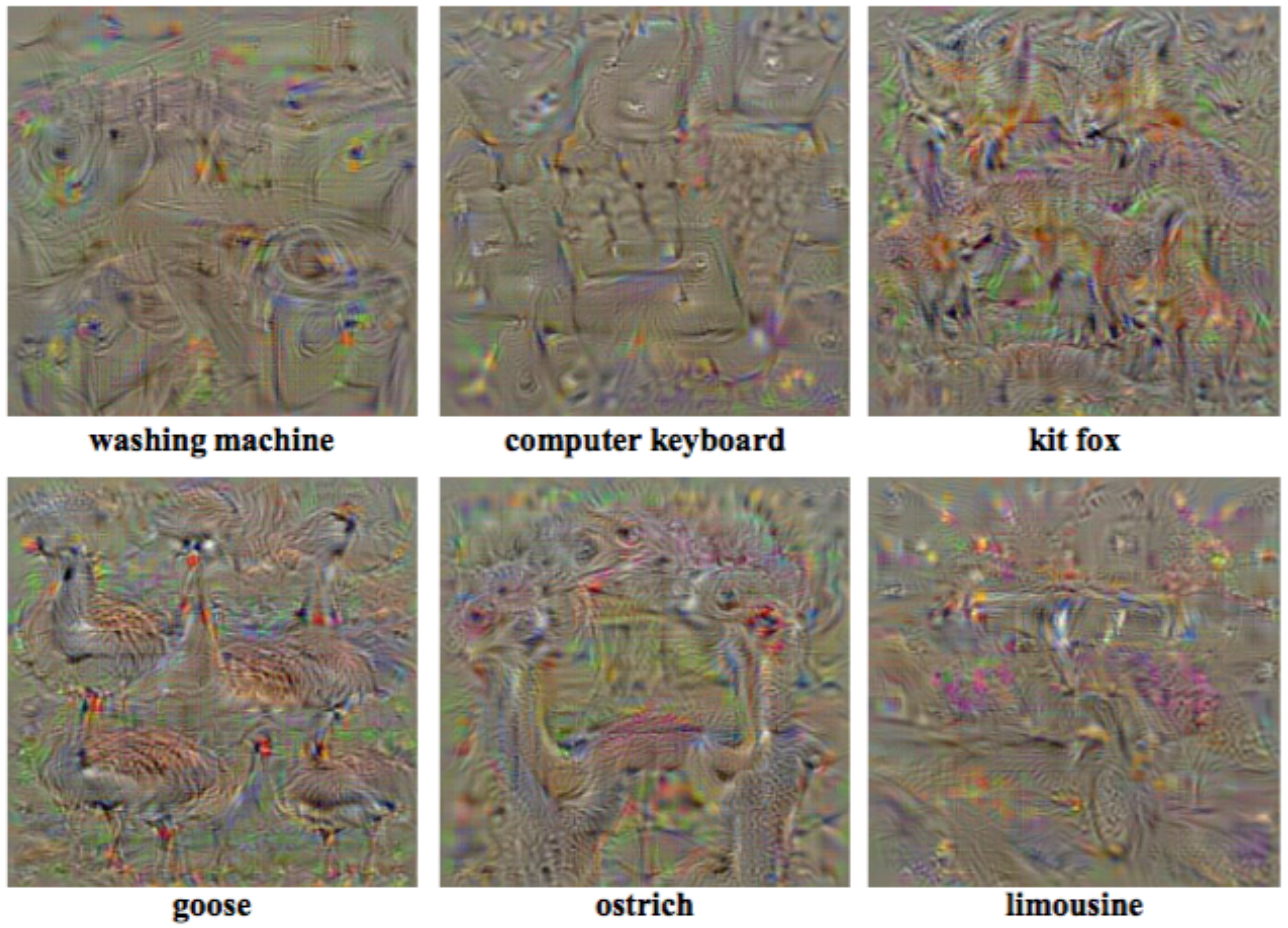
Expression for updated weights from LNN theory:
OLS solution projected onto subspace spanned by first L layers!

New Insights

How are Memories of Objects Stored in a Convnet?

- **Question:** How are appearance models of different classes of objects stored?
- **Experiment:** Probe network to find input image that **maximally excites** the (say) goose neuron (**Activity Maximization**)
- **Theoretical Result:** Closed-form expression for the activity-maximizing image I_c^*:

$$I_c^* = \sum_{\mathscr{P}_i \in \mathscr{P}} I^*_{\mathscr{P}_i}(c, g^*_{\mathscr{P}_i}) \propto \sum_{\mathscr{P}_i \in \mathscr{P}} \mu(c, g^*_{\mathscr{P}_i}). \qquad (1)$$



K. Simonyan, A. Vedaldi, and A. Zisserman.
Deep inside convolutional networks: Visualising image classification models and saliency maps (2013)

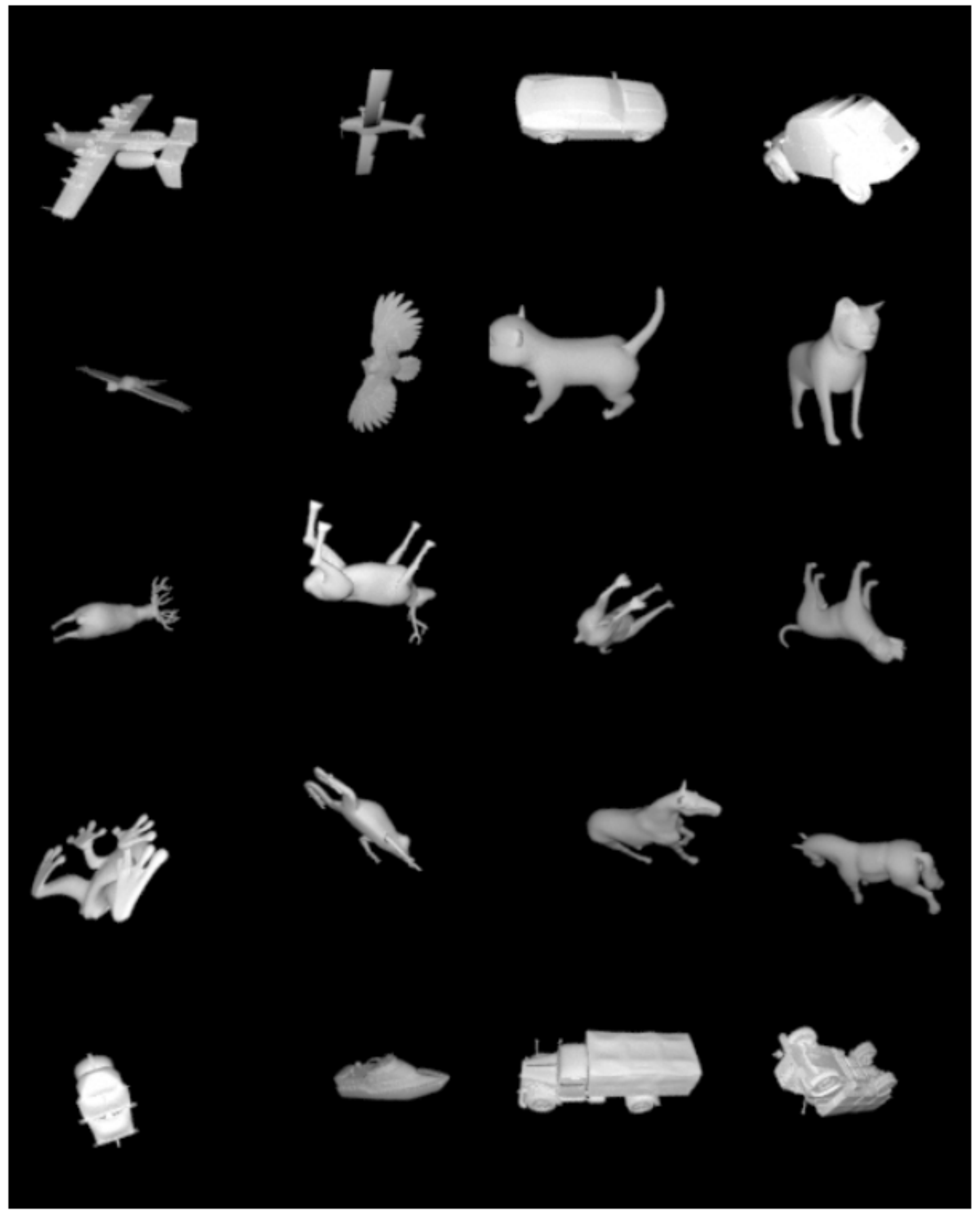
Empirical Observation

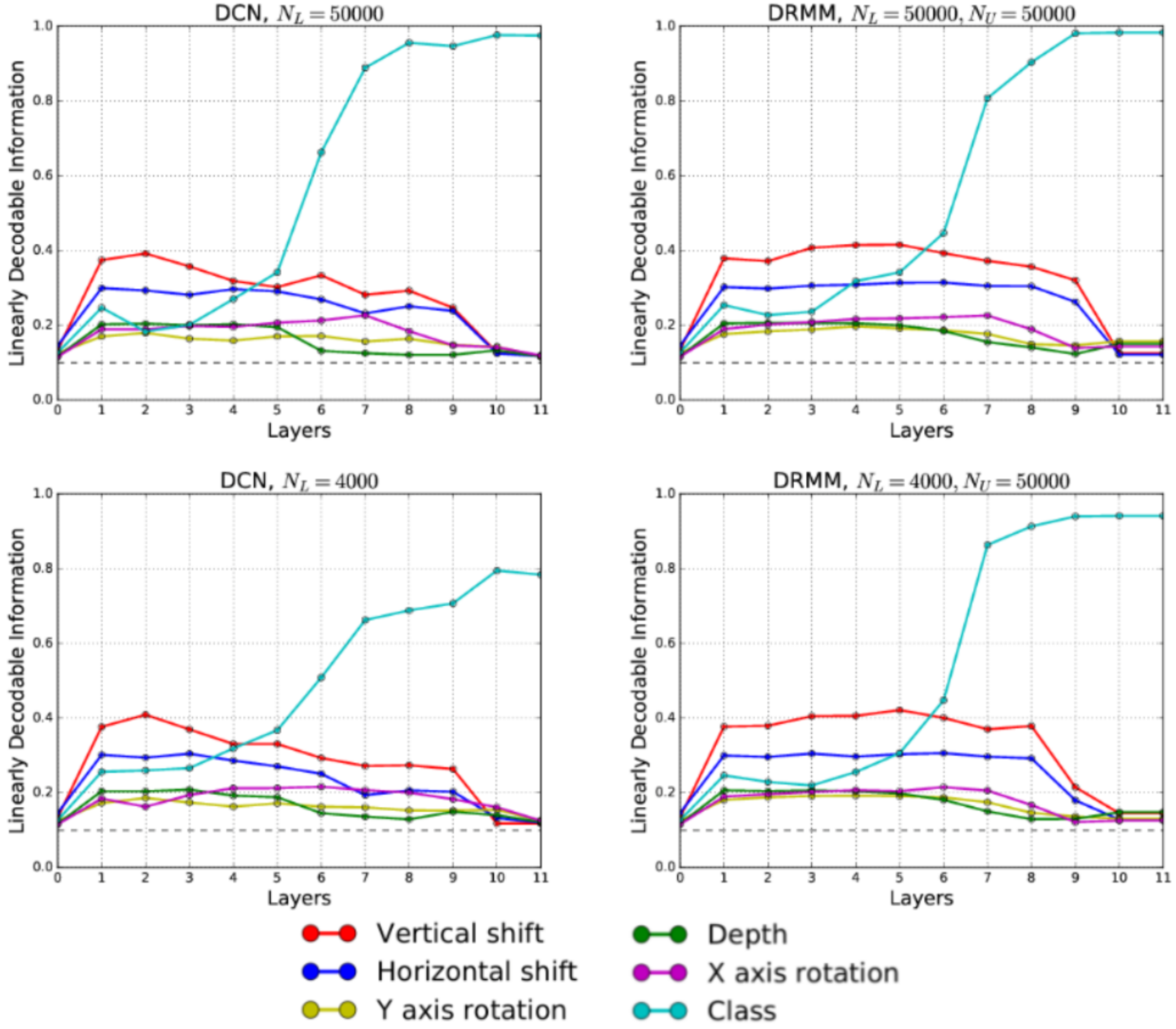
Deep convnets appear to model class appearance using a **mixture over nuisance variables**.

How much information about nuisance variables is there in a net trained for classification?

BlenderRender:

Synthetically rendered images





Unifying Neural Network and Probabilistic Perspectives

Aspect	**Neural Nets Perspective** *Deep Convnets (DCNs)*	**Probabilistic Perspective** *Deep Rendering Model (DRM)*
Model	Weights and biases of filters at a given layer	Partial Rendering at a given abstraction level/scale
	Number of Layers	Number of Abstraction Levels
	Number of Filters in a layer	Number of Clusters/Classes at a given abstraction level
	Implicit in network weights; can be computed by product of weights over all layers or by activity maximization	Category prototypes are finely detailed versions of coarser-scale super-category prototypes. Fine details are modeled with affine nuisance transformations.
Inference	Forward propagation thru DCN	Exact bottom-up inference via Max-Sum Message Passing (with Max-Product for Nuisance Factorization).
	Input and Output Feature Maps	Probabilistic Max-Sum Messages (real-valued functions of variables nodes)
	Template matching at a given layer (convolutional, locally or fully connected)	Local computation at factor node (log-likelihood of measurements)
	Max-Pooling over local pooling region	Max-Marginalization over Latent Translational Nuisance transformations
	Rectified Linear Unit (ReLU). Sparsifies output activations.	Max-Marginalization over Latent Switching state of Renderer. Low prior probability of being ON.
Learning	Stochastic Gradient Descent	Batch Discriminative EM Algorithm with Fine-to-Coarse E-step + Gradient M-step. *No coarse-to-fine pass in E-step*.
	N/A	Full EM Algorithm
	Batch-Normalized SGD	Discriminative Approximation to Full EM (assumes Diagonal Pixel Covariance)

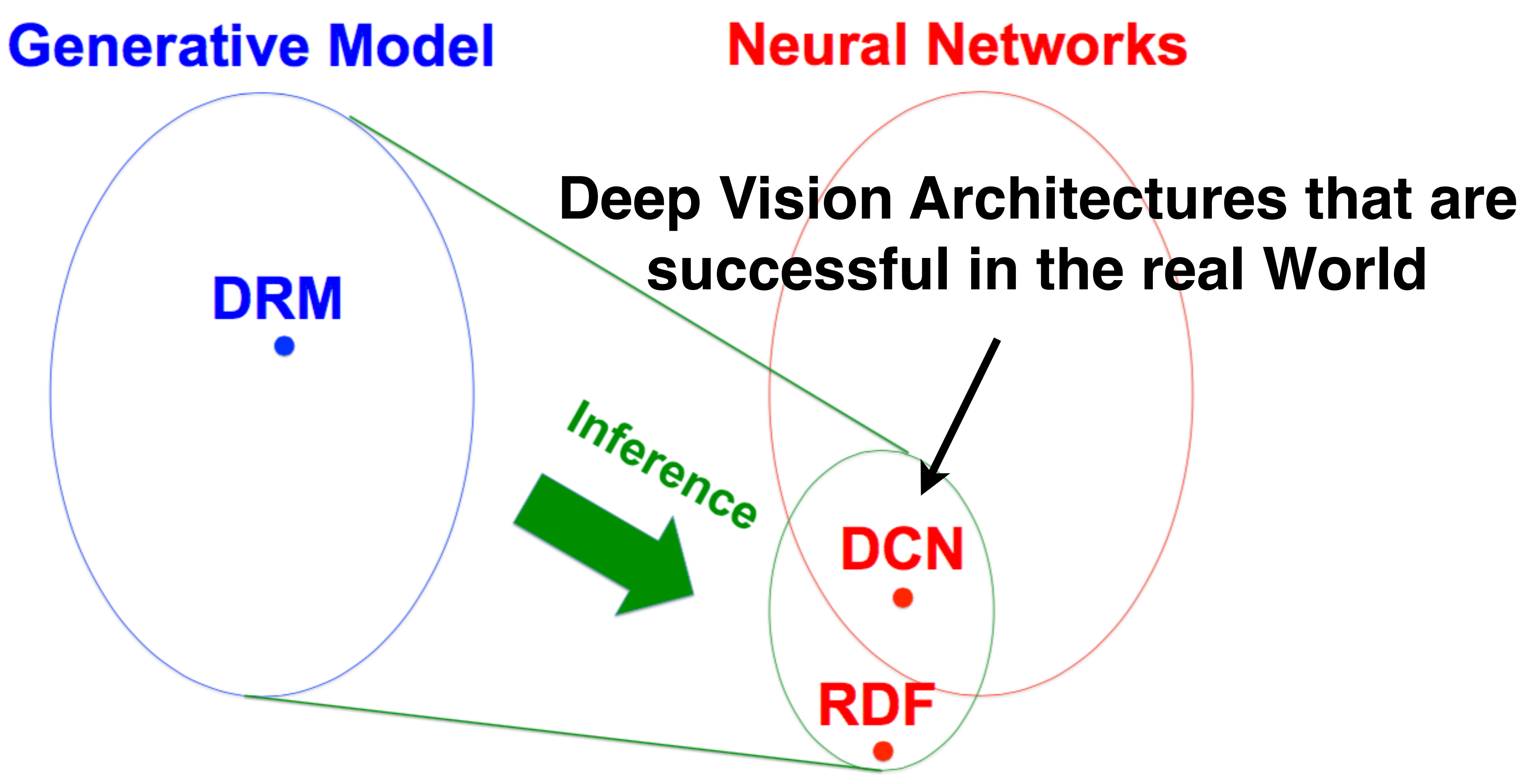
Convnets are "accidentally" Neural Nets

Question: Do all neural nets arise as inference algorithms for a generative prob. model?

Ans: To our knowledge, no. (We tried.)

Question: Then what is special about these successful real-world deep vision architectures? What property ties them all together?

Tentative Ans: Our theory suggests that the single concept (if it exists) is that they are all Efficient max-sum message passing (Dynamic Programming) algorithms for DRMM variants.



- Technically, DCNs are neural networks; but that's not the important part
- **DCNs are max-sum message passing networks** that arise from a **generative model** (DRMM)

Key Limitations & Challenges

- DP proofs rely on Non-Negativity assumption (**NN**-DRMM), whereas real trained Convnets have signed weights in general.
- Despite state-of-art performance in semi-sup learning tasks, trained DRMM generates poor quality image samples due to enormous number of iid latent variables (one per ReLu and MaxPool switch).
- Discriminative relaxation is lossy operation i.e. more than on generative model/classifier might be consistent with same discriminative classifier. [Ng & Jordan 2002, Mitchell Ch. 3]
- Currently we have little knowledge about the nature of the trained weights as function of the training data. Stay tuned here…

Summary of Theory

IARPA
BE THE FUTURE

- NN-DRMM - a hierarchical generative model which is effectively a **Deep Sparse Path Coding** model

- Convnets are an **efficient bottom-up inference algorithm** for this model (Dynamic Programming OR max-sum-product message passing)

- Provides **principled** way of alleviating limitations of Convnets: intriguingly predicts **brain-like features** (ex: feedback connections, synaptic pruning)

- First theory of deep learning that both **explains** and leads to new architecture that **performs** (state-of-the-art on several benchmarks)

- We think it will be useful in **bridging the gap** between Deep Learning and Theoretical/Computational Neuroscience

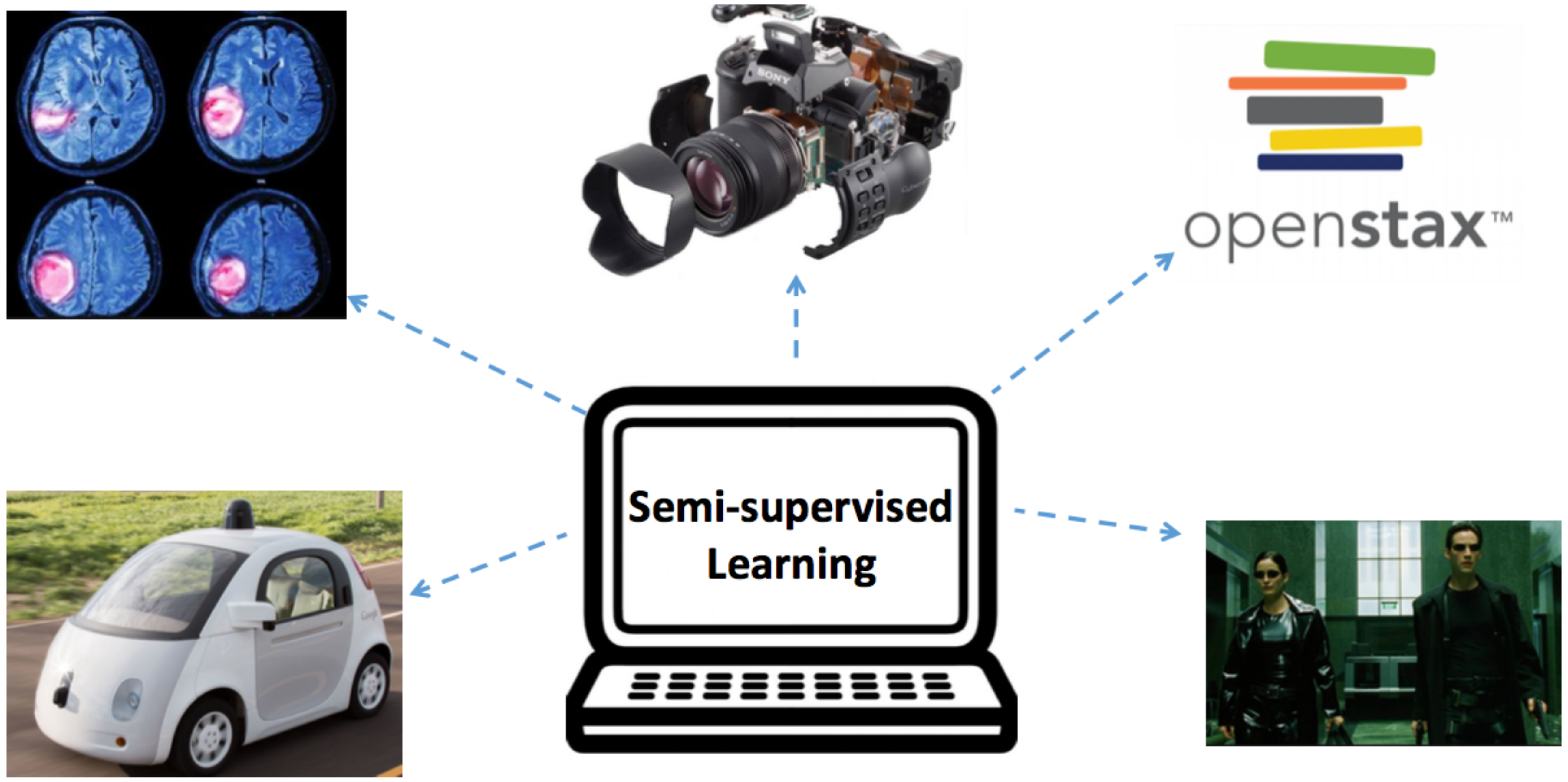
Outlook & Open Questions

- Where do we go from here?
  - More interaction between Theory and Experiment: testing predictions and experimentation with highly trained nets (“Artificial Neuroscience”)
  - Focus on problem areas for Convnets: kryptonite categories, adversarial perturbations.
  - Back to physics of image rendering: what properties of images allow them to be well-parsed by DRMM/Convnets?
  - Is there a deeper reason that DP JMAP inference algorithms (e.g. Convnets and Decision Forests) have been so successful in vision?

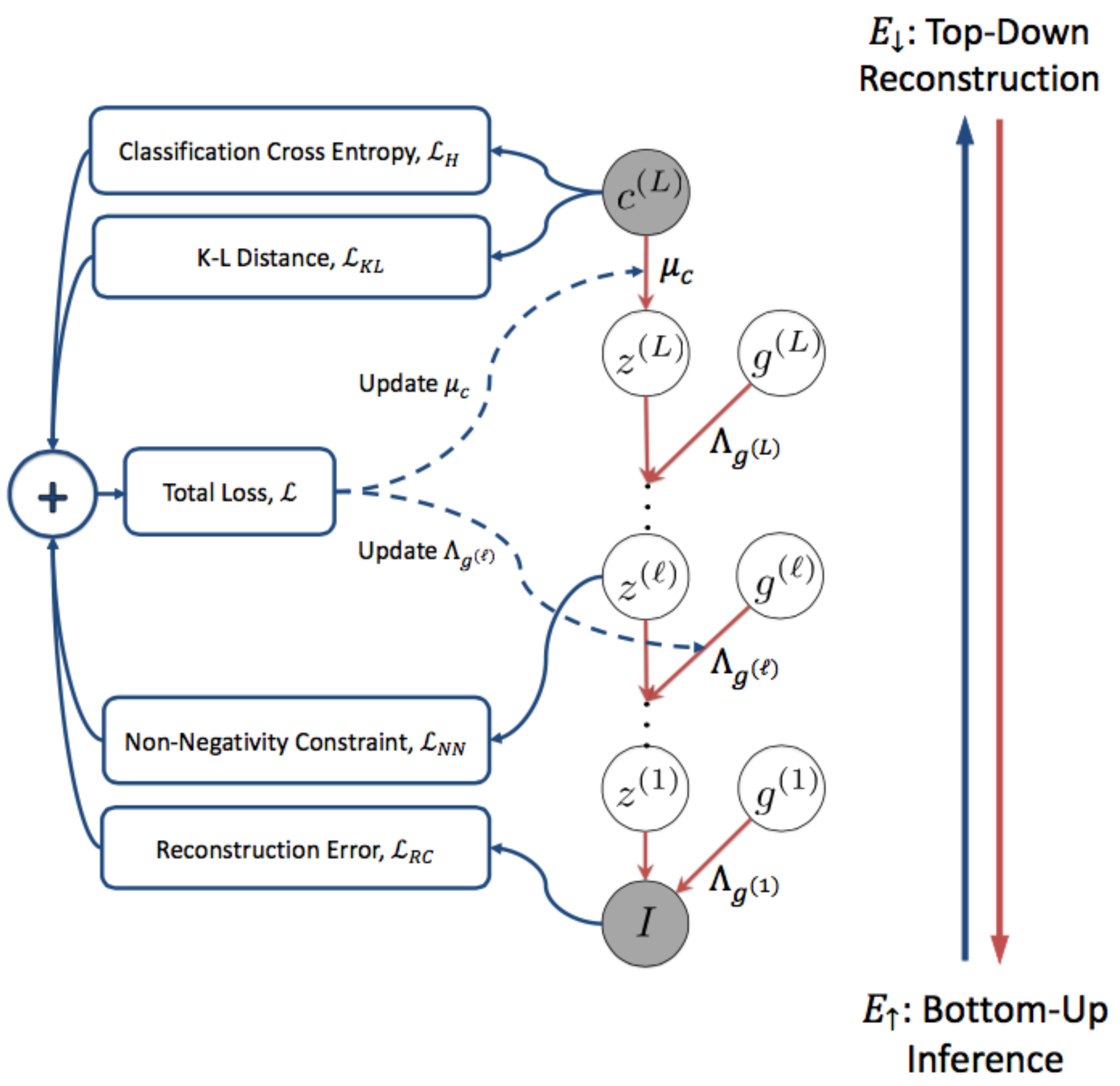
Application:
Semi-supervised Learning

Semi-supervised Learning for Visual Recognition

Semi-supervised learning makes use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data.



New DRMM Learning Algorithm with Top-Down Inference



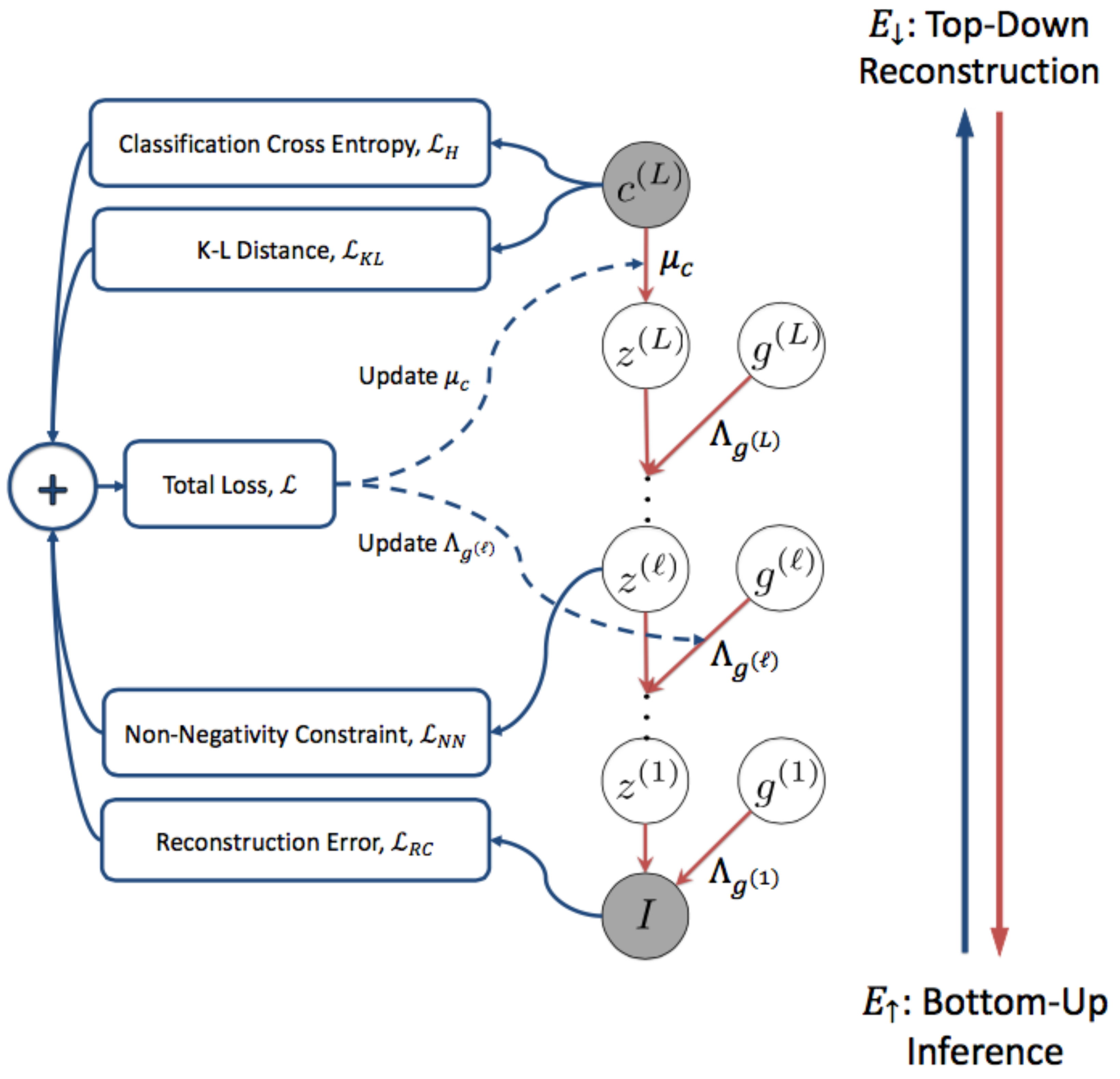
Bottom-Up E-Step ($E_\uparrow$):

$$
\begin{aligned}
u_n^{(\ell)\uparrow} &= \Lambda^T_{t^{(\ell)}} I_n^{(\ell-1)} \\
s^{(\ell)\downarrow} &= \operatorname{sgn}\left(z^{(\ell)\downarrow}\right) \\
\forall s^{(\ell)\downarrow} \in \{\pm 1\}^{D^{(\ell)}} : \hat{a}_n^{(\ell)\updownarrow}(s^{(\ell)\downarrow}) &= [s^{(\ell)\downarrow} \odot u_n^{(\ell)\uparrow} > 0] \\
\forall s^{(\ell)\downarrow} \in \{\pm 1\}^{D^{(\ell)}} : \hat{t}_n^{(\ell)\updownarrow}(s^{(\ell)\downarrow}) &= \underset{t^{(\ell)}}{\operatorname{argmax}}\ s^{(\ell)\downarrow} \odot u_n^{(\ell)\uparrow}(t^{(\ell)}) \\
I_n^{(\ell)}(s^{(\ell)\downarrow}) &= M_{\hat{a}_n^{(\ell)}}\left(\Lambda^T_{\hat{t}^{(\ell)}} I_n^{(\ell-1)}\right) \\
&= s^{(\ell)\downarrow} \odot \operatorname{MaxPool}\left(\operatorname{ReLu}\left(\operatorname{diag}(s^{(1)\downarrow}) u_n^{(1)\uparrow}(\mathcal{T})\right)\right) \\
\hat{c}_n^{(L)} &= \underset{c^{(L)}}{\operatorname{argmax}}\ \mu^T_{c^{(L)}} I_n^{(L)}
\end{aligned}
$$

Top-Down/Traceback E-Step ($E_\uparrow$):

$$
\begin{aligned}
\hat{z}_n^{(\ell)\downarrow} &= \Lambda_{\hat{g}_n^{(\ell+1)}} \cdots \Lambda_{\hat{g}_n^{(L)}} \mu_{\hat{c}_n^{(L)}} \\
&= \Lambda_{\hat{g}_n^{(\ell+1)}} \hat{z}_n^{(\ell+1)\downarrow} \\
\hat{s}_n^{(\ell)\downarrow} &= \operatorname{sgn}(\hat{z}_n^{(\ell)\downarrow}) \\
\hat{a}_n^{(\ell)\updownarrow} &= \hat{a}_n^{(\ell)\updownarrow}(\hat{s}_n^{(\ell)\downarrow}) = [\hat{s}_n^{(\ell)\downarrow} \odot u_n^{(\ell)\uparrow} > 0] \\
\hat{t}_n^{(\ell)\updownarrow} &= \hat{t}_n^{(\ell)\updownarrow}(s_n^{(\ell)\downarrow}) = \underset{t^{(\ell)}}{\operatorname{argmax}}\ s_n^{(\ell)\downarrow} \odot u_n^{(\ell)\uparrow}(t^{(\ell)})
\end{aligned}
$$

Application: Semi-supervised Learning for Visual Recognition



- For images with labels, we optimize the cross-entropy loss

$$\mathscr{L} \equiv \alpha_H \mathscr{L}_H + \alpha_{RC} \mathscr{L}_{RC} + \alpha_{KL} \mathscr{L}_{KL} + \alpha_{NN} \mathscr{L}_{NN}$$

$$\mathscr{L}_H \equiv -\frac{1}{|\mathscr{D}_L|} \sum_{n \in \mathscr{D}_L} \sum_{c \in \mathscr{C}} [\hat{c}_n = c_n] \log q(c|I_n)$$

$$\mathscr{L}_{RC} \equiv \frac{1}{N} \sum_{n=1}^{N} \left\| I_n - \hat{I}_n \right\|_2^2$$

$$\mathscr{L}_{KL} \equiv \frac{1}{N} \sum_{n=1}^{N} \sum_{c \in \mathscr{C}} q(c|I_n) \log \left(\frac{q(c|I_n)}{p(c)} \right)$$

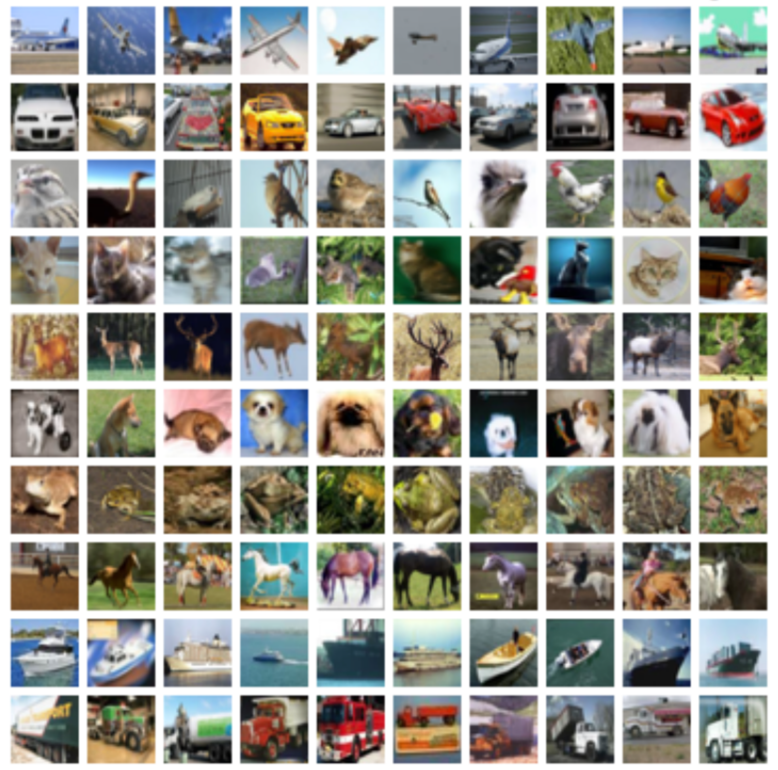
$$\mathscr{L}_{NN} \equiv \frac{1}{N} \sum_{n=1}^{N} \sum_{\ell=1}^{L} \left\| \max\left\{0, -z_n^{\ell}\right\} \right\|_2^2.$$

Tan Nguyen

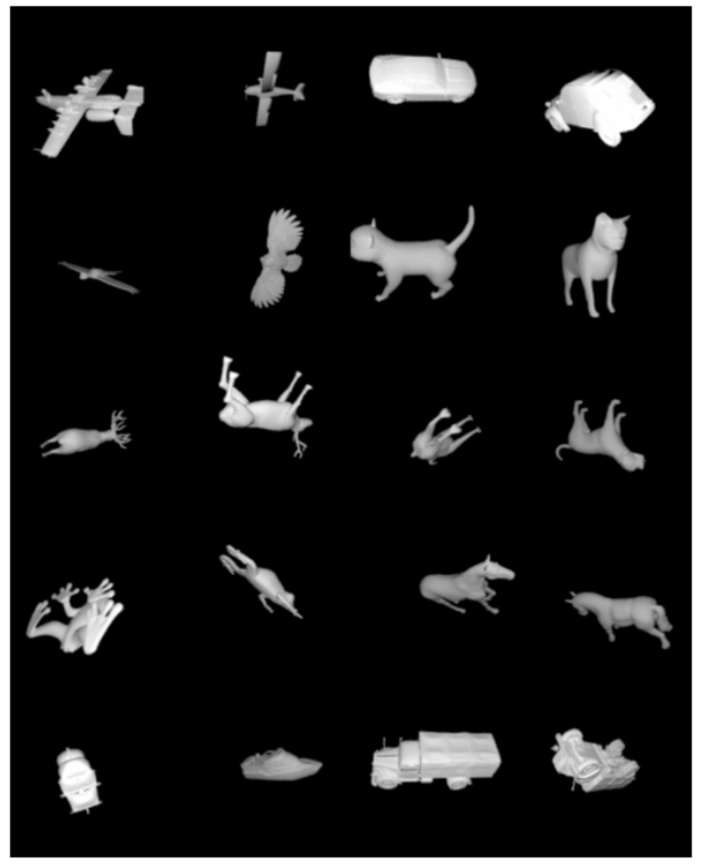
Experiments on Benchmarks

60K training examples and 10K test examples. Labels for both training and test sets are provided.

73257 training examples and 26032 test examples. Labels for both training and test sets are provided.



50K training examples and 10K test examples. Labels for both training and test sets are provided.



50K training examples and 10K test examples. Labels for both training and test sets are provided.

Experiments on MNIST: State-of-the-Art

(amongst all methods that do not use data augmentation)

Model	Test error (%) for a given number of labeled examples		
	$N_L = 50$	$N_L = 100$	$N_L = 1K$
DGN [Kingma]	-	3.33 ± 0.14	2.40 ± 0.02
catGAN [springenberg]	-	1.39 ± 0.28	-
Virtual Adversarial [Miyato]	-	2.12	-
Skip Deep Generative Model [Maaløe]	-	1.32	-
LadderNetwork [Rasmus]	-	1.06 ± 0.37	0.84 ± 0.08
Auxiliary Deep Generative Model [Maaløe]	-	0.96	-
ImprovedGAN [Salimans]	2.21 ± 1.36	0.93 ± 0.065	-
DRMM 5-layer	21.73	13.41	2.35
DRMM 5-layer + NN penalty	22.10	12.28	2.26
DRMM 5-layer + KL penalty [Patel, Nguyen]	2.46	1.36	0.71
DRMM 5-layer + KL and NN penalties [Nguyen]	**0.91**	**0.57**	**0.6**

Table: Test error for semi-supervised learning on MNIST using $N_U = 60K$ unlabeled images and $N_L \in \{100, 600, 1K\}$ labeled images.

Experiments on SVHN: State-of-the-Art

(amongst all methods that do not use data augmentation)

Model	Test error (%) for a given number of labeled examples 500	1000	2000
DGN [Kingma]		36.02 ± 0.10	
Virtual Adversarial [Miyato]		24.63	
Auxiliary Deep Generative Model [Maaløe]		22.86	
Skip Deep Generative Model [Maaløe]		16.61 ± 0.24	
ImprovedGAN [Salimans]	18.44 ± 4.8	8.11 ± 1.3	$\mathbf{6.16 \pm 0.58}$
DRMM 9-layer + KL penalty [Patel, Nguyen]	11.11	9.75	8.44
DRMM 9-layer + KL and NN penalty [Nguyen]	**9.85**	**6.78**	6.50

Table: Test error for semi-supervised learning on SVHN using $N_U = 73,257$ unlabeled images and $N_L \in \{500, 1K, 2K\}$ labeled images.

Experiments on CIFAR10

Model	Test error (%) for a given number of labeled examples	
	4000	8000
Ladder network [Rasmus]	20.40 ± 0.47	-
CatGAN [Springenberg]	19.58 ± 0.46	-
ImprovedGAN [Salimans]	$\mathbf{18.63 \pm 2.32}$	17.72 ± 1.82
DRMM 9-layer + KL penalty [Patel, Nguyen]	23.24	20.95
DRMM 9-layer + KL and NN penalty [Nguyen]	21.50	**17.16**

Table: Test error for semi-supervised learning on CIFAR10 using $N_U = 50K$ unlabeled images and $N_L \in \{4K, 8K\}$ labeled images.

Thanks!

- **Funding:** IARPA MICRONS Project

- **Contact me:** Feel free to email me: abp4@rice.edu, ankitp@bcm.edu to talk more about our theory, its potential impact in DL and neuroscience, and potential collaborations

IARPA
BE THE FUTURE

Other Current Research Projects

- Further Development of Theory [Rich B]
- **Using Theory to understand artificial and real Brains**
  - Reverse-Engineering the Visual Cortex [Pitkow, Tolias] and Conductance-based Neuron Models [Gabbani, Pfaffinger]
  - Artificial Neuroscience on RNNs that “know” C [Rich B]
  - *Qualia:* How does one get subjective experience from objective physical measurements?
- **Using Theory/real Brains to guide and develop new advances in Deep Learning**
  - Semi-supervised Learning for Object Recognition [Rich B]: NIPS 2016
  - Event-Driven RNNs for Action Recognition and Tracking [Ashok V, Rich B]
  - Infinite Training Data: Synthetically Rendering Images/Video for Active Learning [Rich B]
  - Deep Learning for Particle Physics: Finding Evidence for New Physics [Paul Padley]
  - Deep Learning for Medical Imaging and Predictive Analytics [Arvind Rao, Edward Castillo, Craig Rusin]