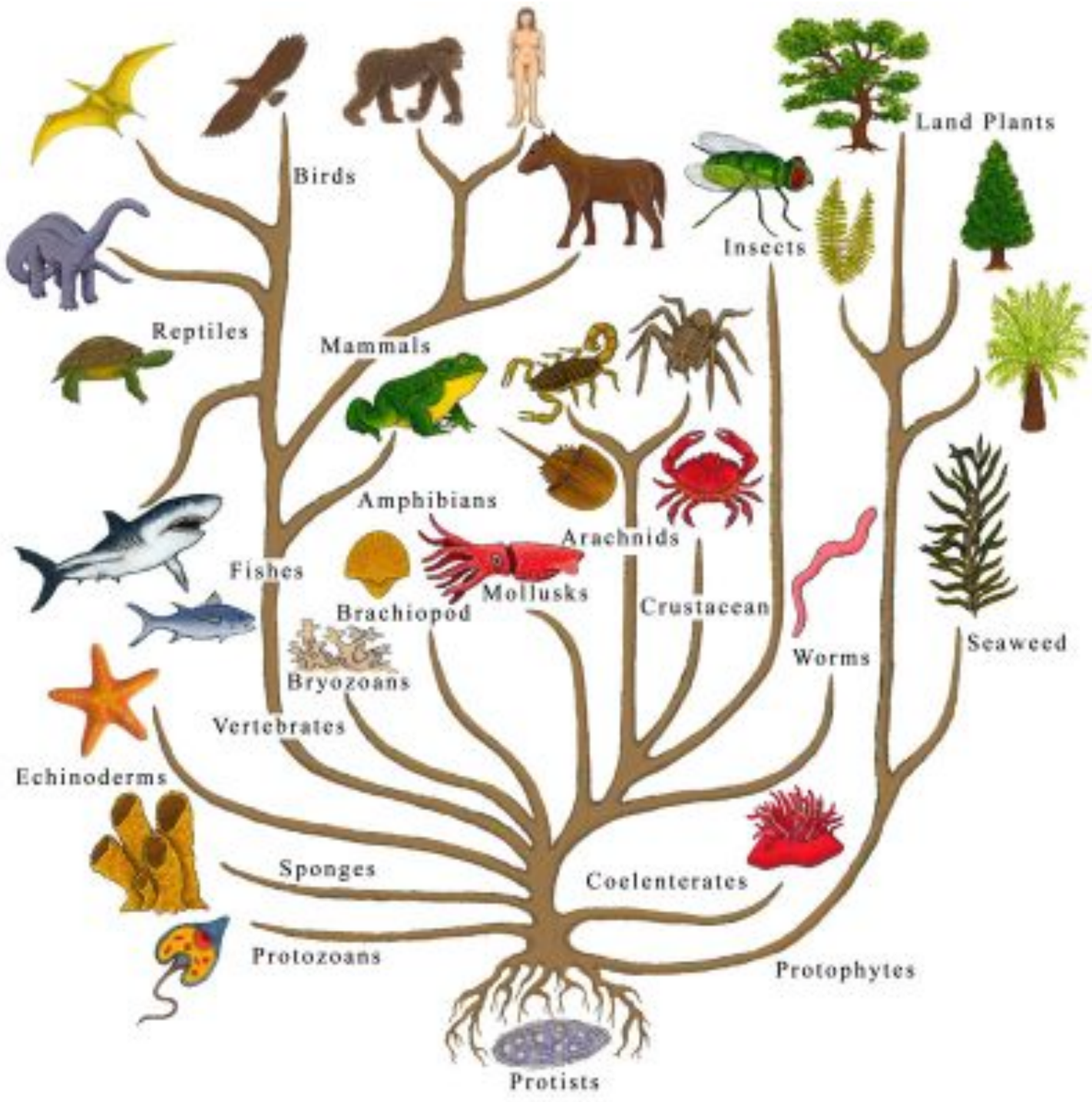
# ELEC/COMP 576: Species of Convnets

**Ankit B. Patel**
*Baylor College of Medicine (Neuroscience Dept.)*
*Rice University (ECE Dept.)*

# Species of Convnets

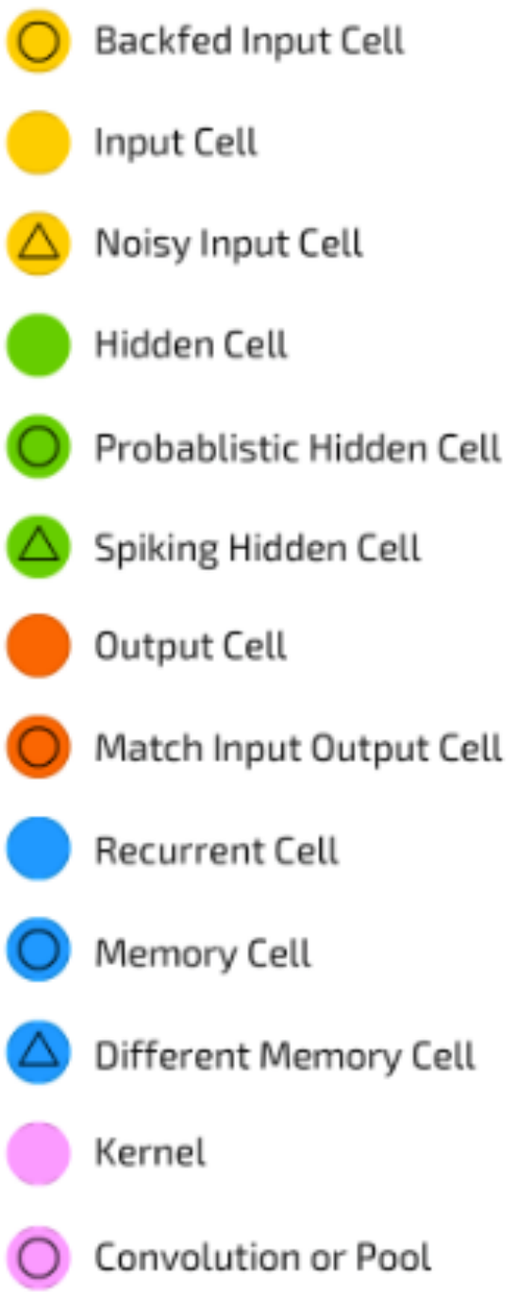
# Evolutionary Biology

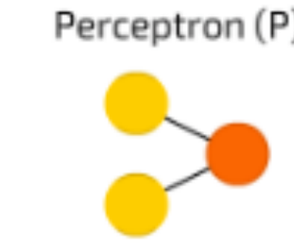


*A mostly complete chart of*

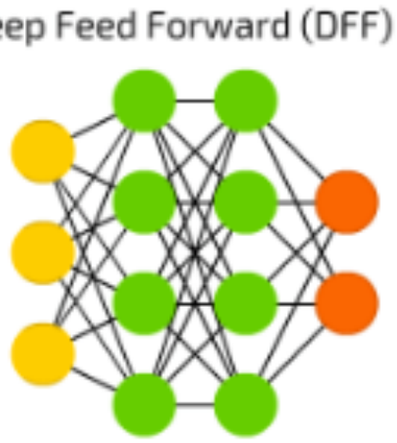
# Neural Networks

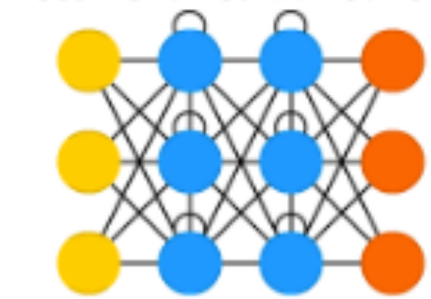


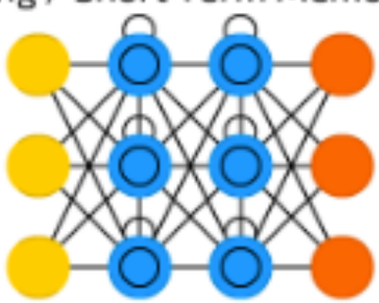


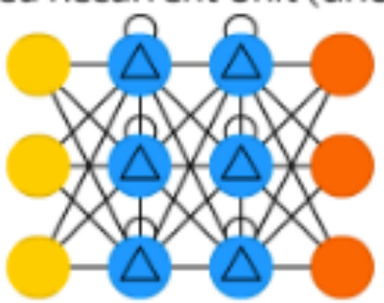





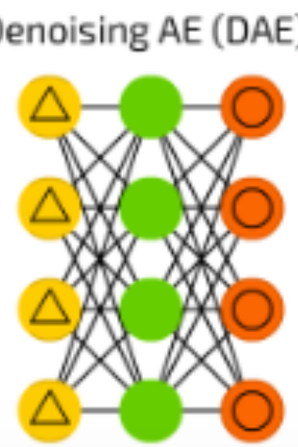





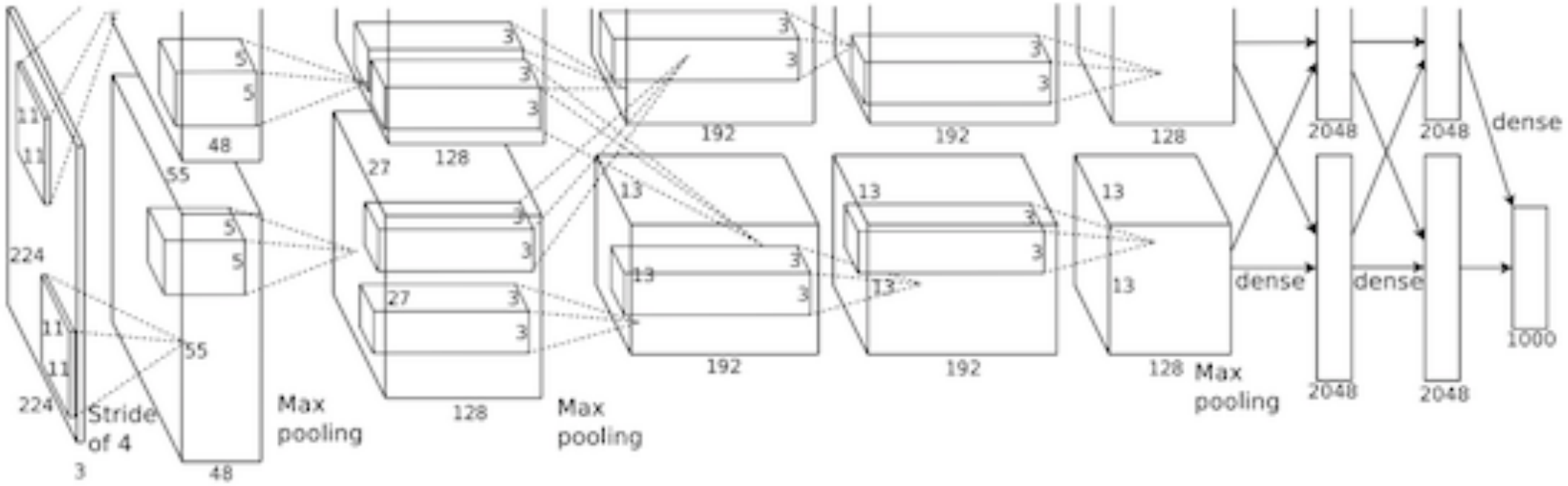





http://www.asimovinstitute.org/neural-network-zoo/

# Object Recognition

# Alex Net



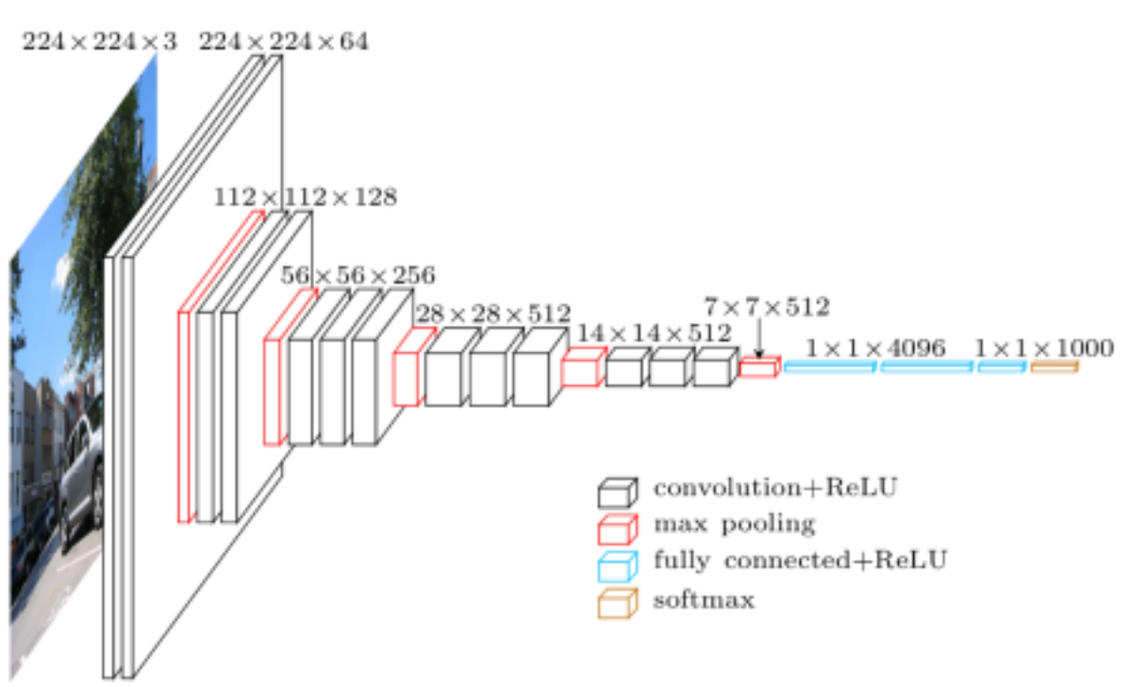

**Innovations**:

- Go Deep
- Train on Multiple GPUs
- ReLU
- DropOut
- Data Augmentation
- Response Normalization

# VGG Net

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

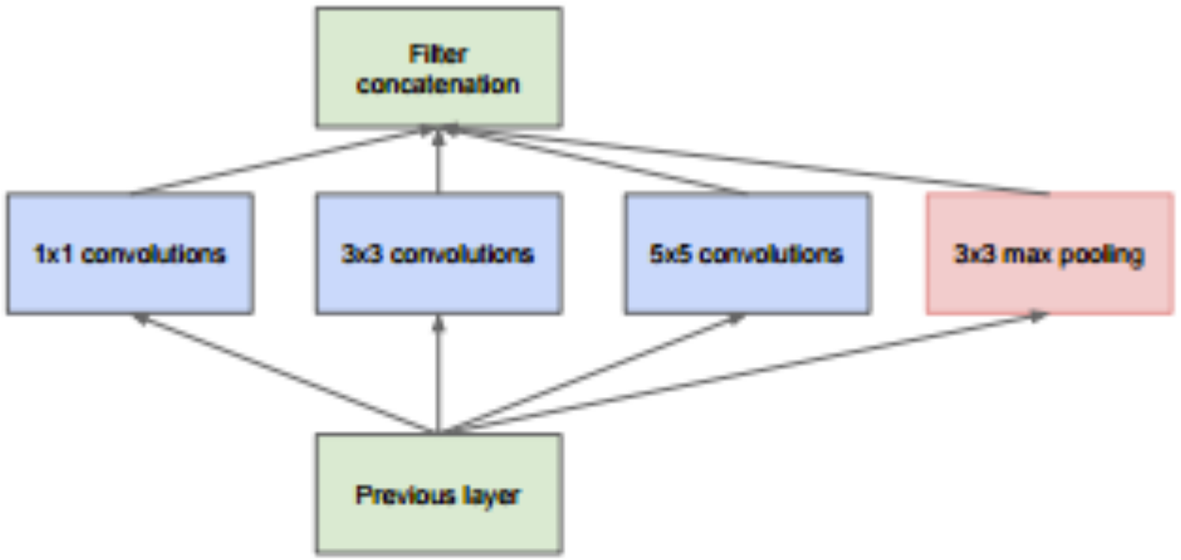
| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64 **LRN** | conv3-64 **conv3-64** | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128 **conv3-128** | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 |
| maxpool | | | | | |
| conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 **conv1-256** | conv3-256 conv3-256 **conv3-256** | conv3-256 conv3-256 conv3-256 **conv3-256** |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 **conv1-512** | conv3-512 conv3-512 **conv3-512** | conv3-512 conv3-512 conv3-512 **conv3-512** |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 **conv1-512** | conv3-512 conv3-512 **conv3-512** | conv3-512 conv3-512 conv3-512 **conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |



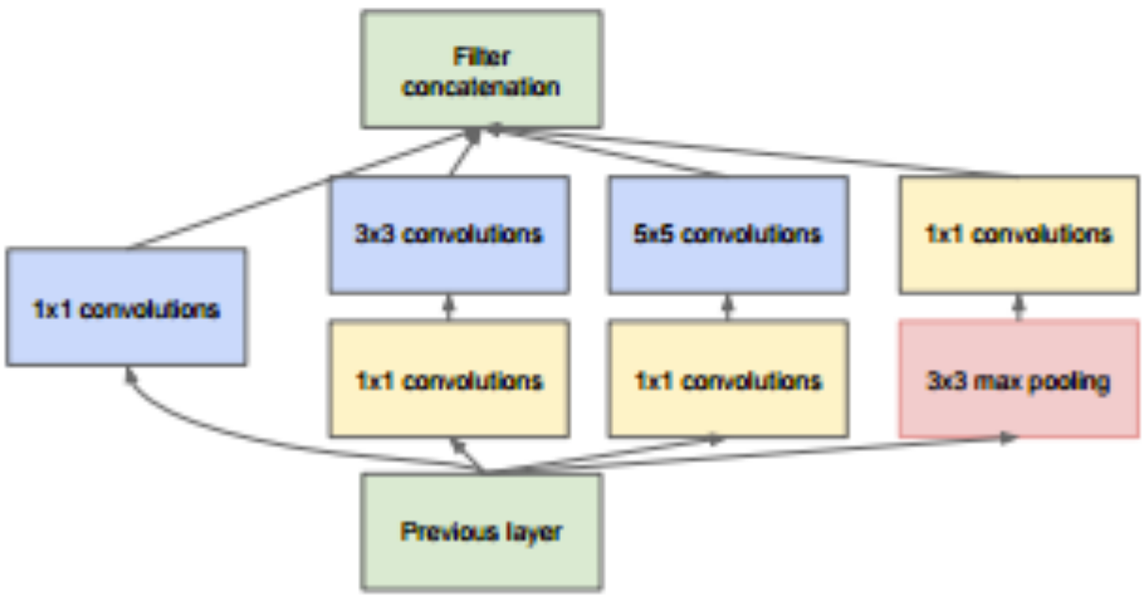

**Innovations**:

- Replace one big filters by multiple smaller filters
- Growing NNs

# GoogLenet



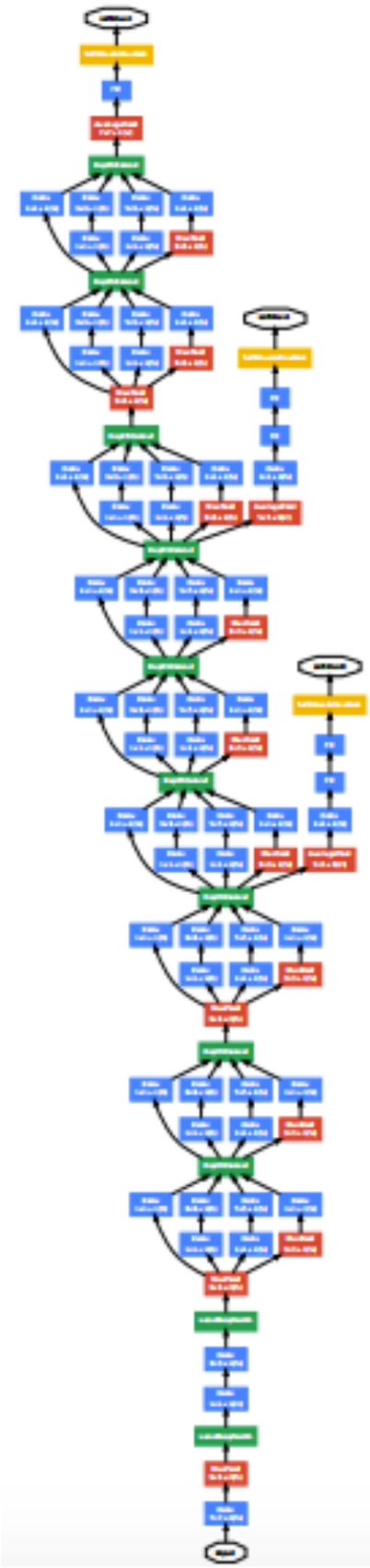

(a) Inception module, naïve version



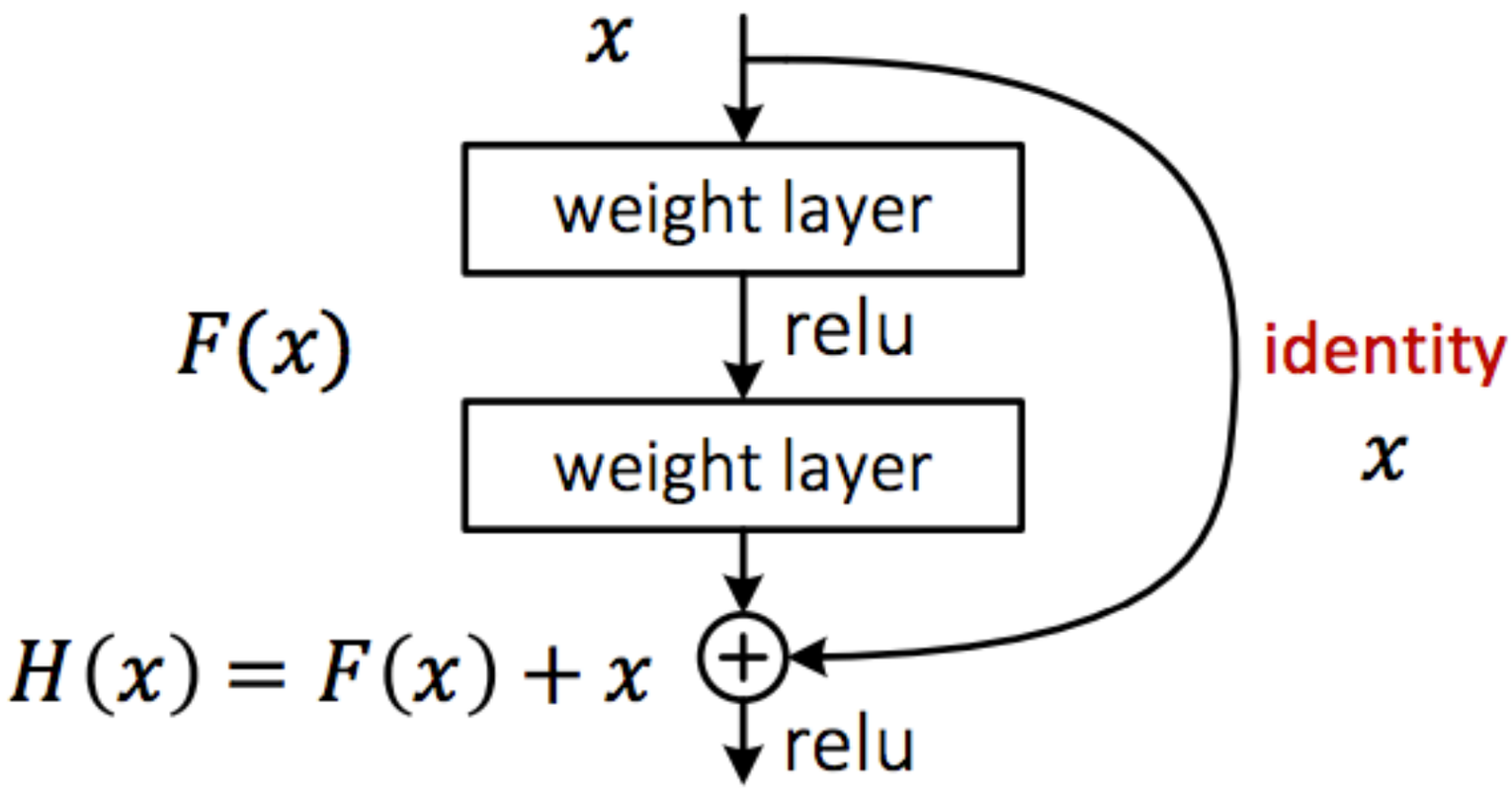

(b) Inception module with dimensionality reduction

**Innovations**:

- Scale-Pooling
- 1x1 Conv (Dim. Reduction)



# ResNet



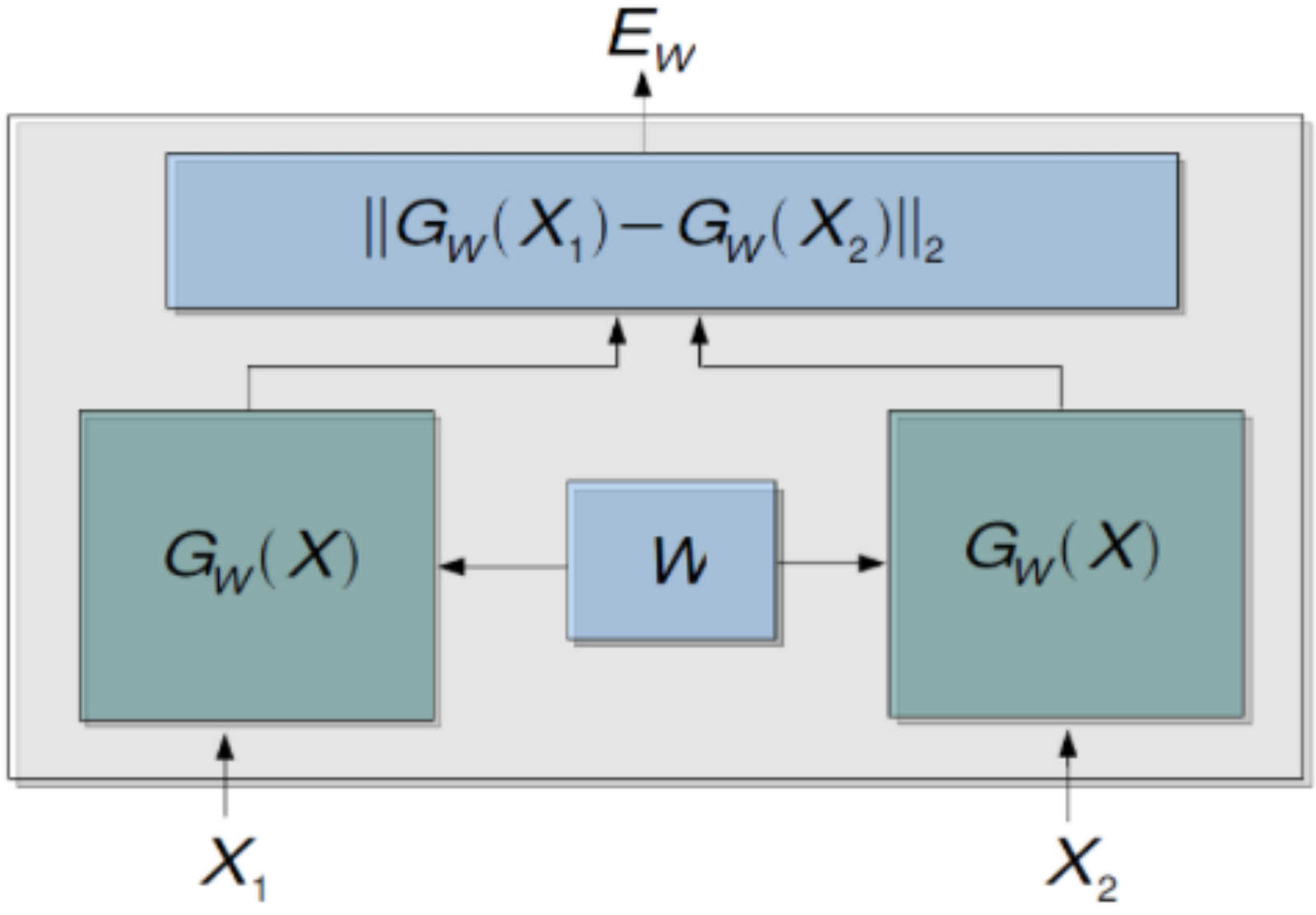


**Innovation**:

- Make it easy to express identity
- Preconditioning

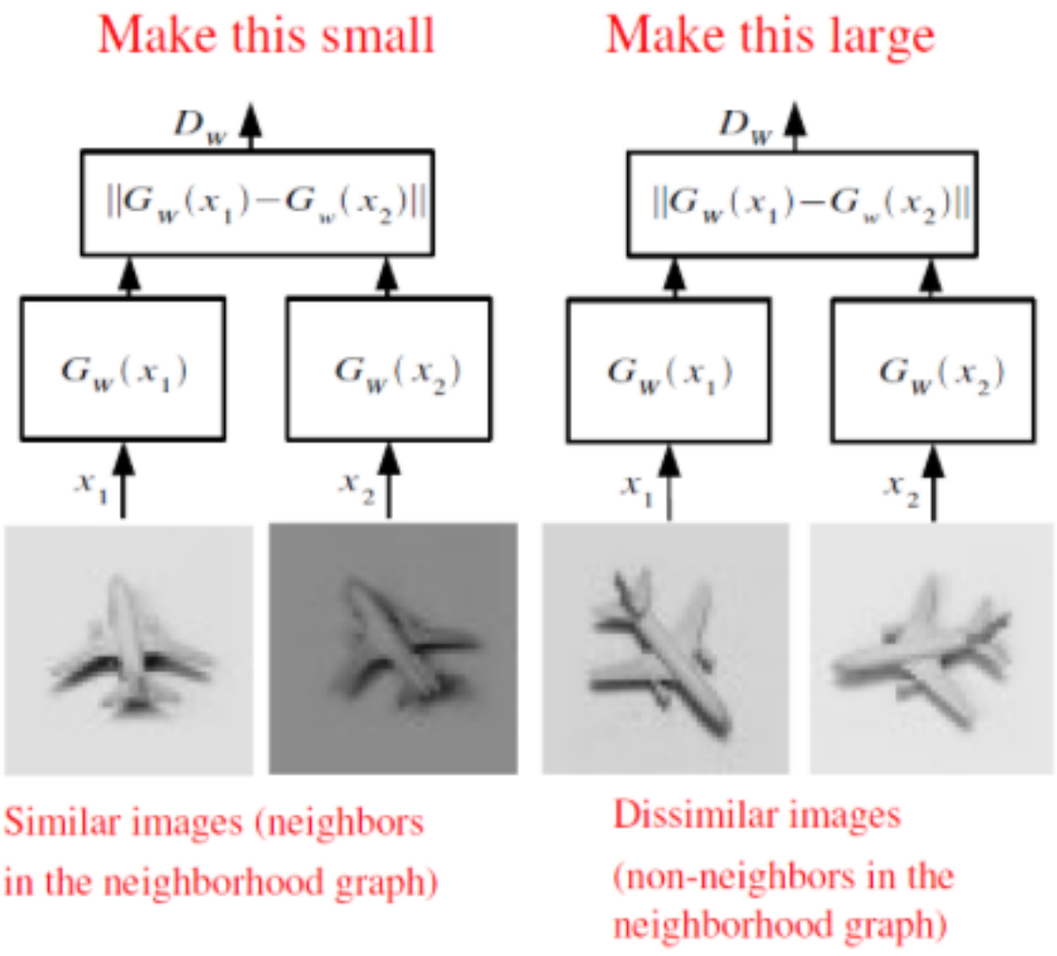
# Siamese Networks

# Siamese Networks for Similarity Discrimination/Matching

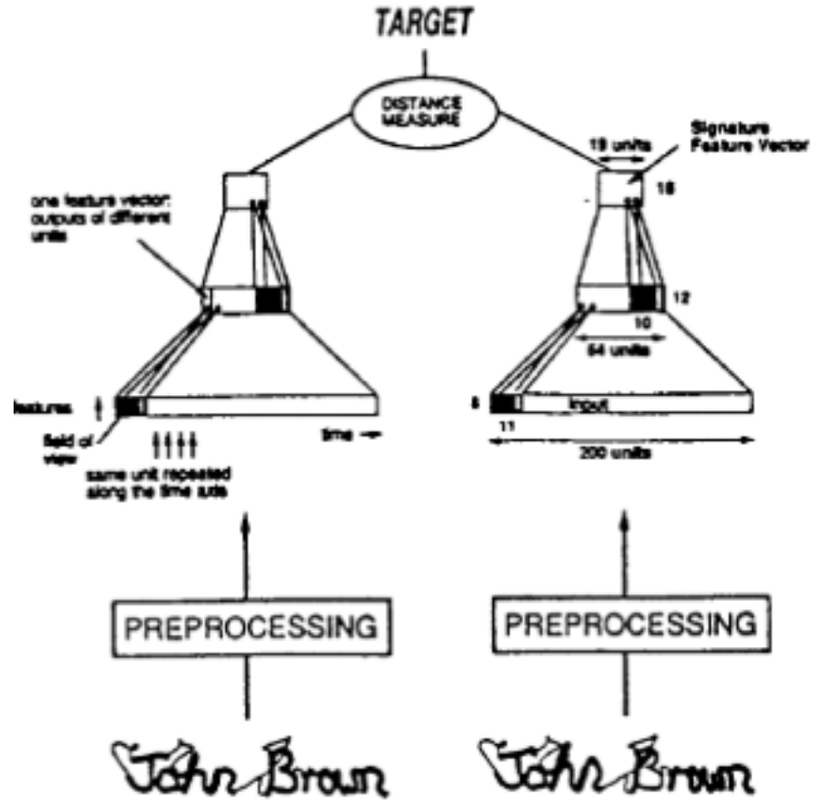



*Learning Hierarchies* of *Invariant Features*. Yann LeCun.
*helper.ipam.ucla.edu/publications/gss2012/gss2012_10739.pdf*

# Siamese Networks for Similarity Discrimination/Matching




*Learning Hierarchies of Invariant Features*. Yann LeCun.
*helper.ipam.ucla.edu/publications/gss2012/gss2012_10739.pdf*




*Bromley J, Guyon I, Lecun Y, et al. Signature Verification using a" Siamese" Time Delay Neural Network, NIPS Proc. 1994.*

# Face Recognition

# DeepFace

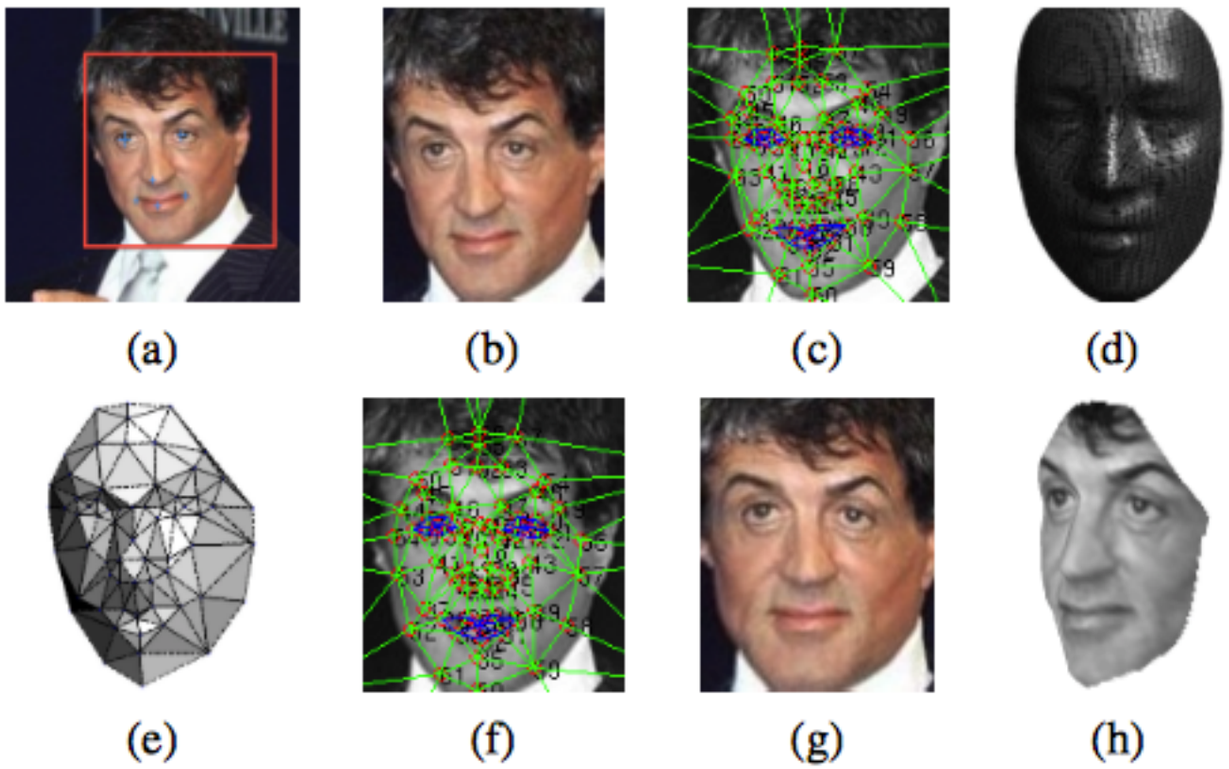



Figure 1. **Alignment pipeline.** (a) The detected face, with 6 initial fiducial points. (b) The induced 2D-aligned crop. (c) 67 fiducial points on the 2D-aligned crop with their corresponding Delaunay triangulation, we added triangles on the contour to avoid discontinuities. (d) The reference 3D shape transformed to the 2D-aligned crop image-plane. (e) Triangle visibility w.r.t. to the fitted 3D-2D camera; darker triangles are less visible. (f) The 67 fiducial points induced by the 3D model that are used to direct the piece-wise affine warpping. (g) The final frontalized crop. (h) A new view generated by the 3D model (not used in this paper).

# DeepFace

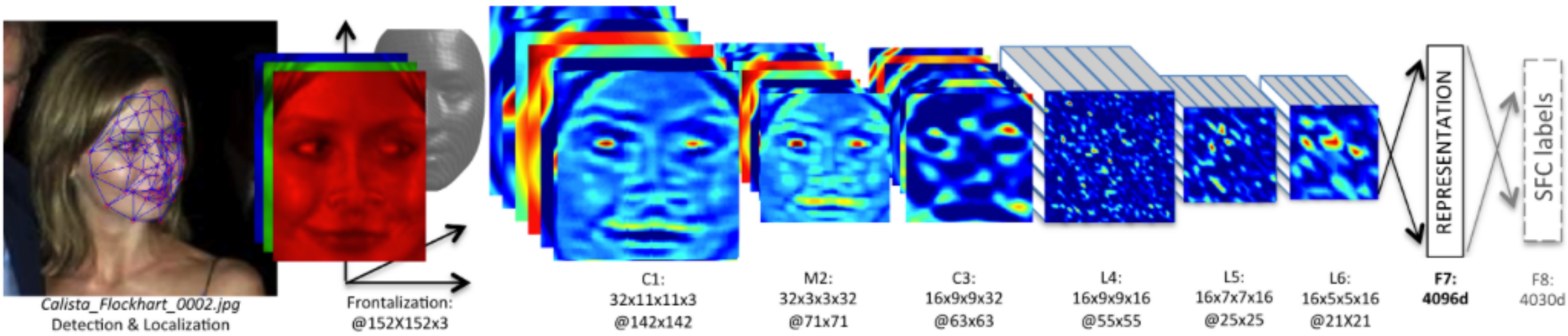



Figure 2. **Outline of the *DeepFace* architecture**. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

# DeepFace

| Method | Accuracy ± SE | Protocol |
|---|---|---|
| Joint Bayesian [6] | 0.9242 ±0.0108 | restricted |
| Tom-vs-Pete [4] | 0.9330 ±0.0128 | restricted |
| High-dim LBP [7] | 0.9517 ±0.0113 | restricted |
| TL Joint Bayesian [5] | 0.9633 ±0.0108 | restricted |
| DeepFace-single | **0.9592** ±0.0029 | unsupervised |
| DeepFace-single | **0.9700** ±0.0028 | restricted |
| DeepFace-ensemble | **0.9715** ±0.0027 | restricted |
| DeepFace-ensemble | **0.9735** ±0.0025 | unrestricted |
| Human, cropped | 0.9753 | |

Table 3. Comparison with the state-of-the-art on the *LFW* dataset.

| Method | Accuracy (%) | AUC | EER |
|---|---|---|---|
| MBGS+SVM- [31] | 78.9 ±1.9 | 86.9 | 21.2 |
| APEM+FUSION [22] | 79.1 ±1.5 | 86.6 | 21.4 |
| STFRD+PMML [9] | 79.5 ±2.5 | 88.6 | 19.9 |
| VSOF+OSS [23] | 79.7 ±1.8 | 89.4 | 20.0 |
| DeepFace-single | **91.4** ±1.1 | 96.3 | 8.6 |

Table 4. Comparison with the state-of-the-art on the *YTF* dataset.

# FaceNet


Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by $L_2$ normalization, which results in the face embedding. This is followed by the triplet loss during training.

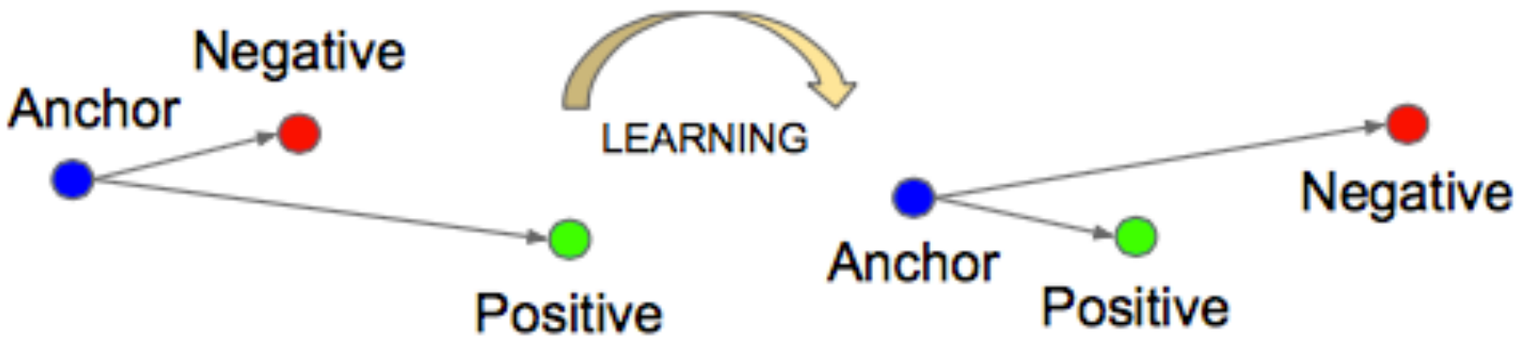



Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

# FaceNet

- Labeled Faces in the Wild:
  - No alignment: 98.87% ± 0.15
  - With alignment: 99.63% ± 0.09
- Youtube Faces DB: 95.12% ± 0.39 (state-of-the-art)


Figure 6. **LFW errors.** This shows all pairs of images that were incorrectly classified on LFW. Only eight of the 13 false rejects shown here are actual errors the other five are mislabeled in LFW.

# Object Tracking

# Visual Examples

# MDNet: Convnet for Object Tracking

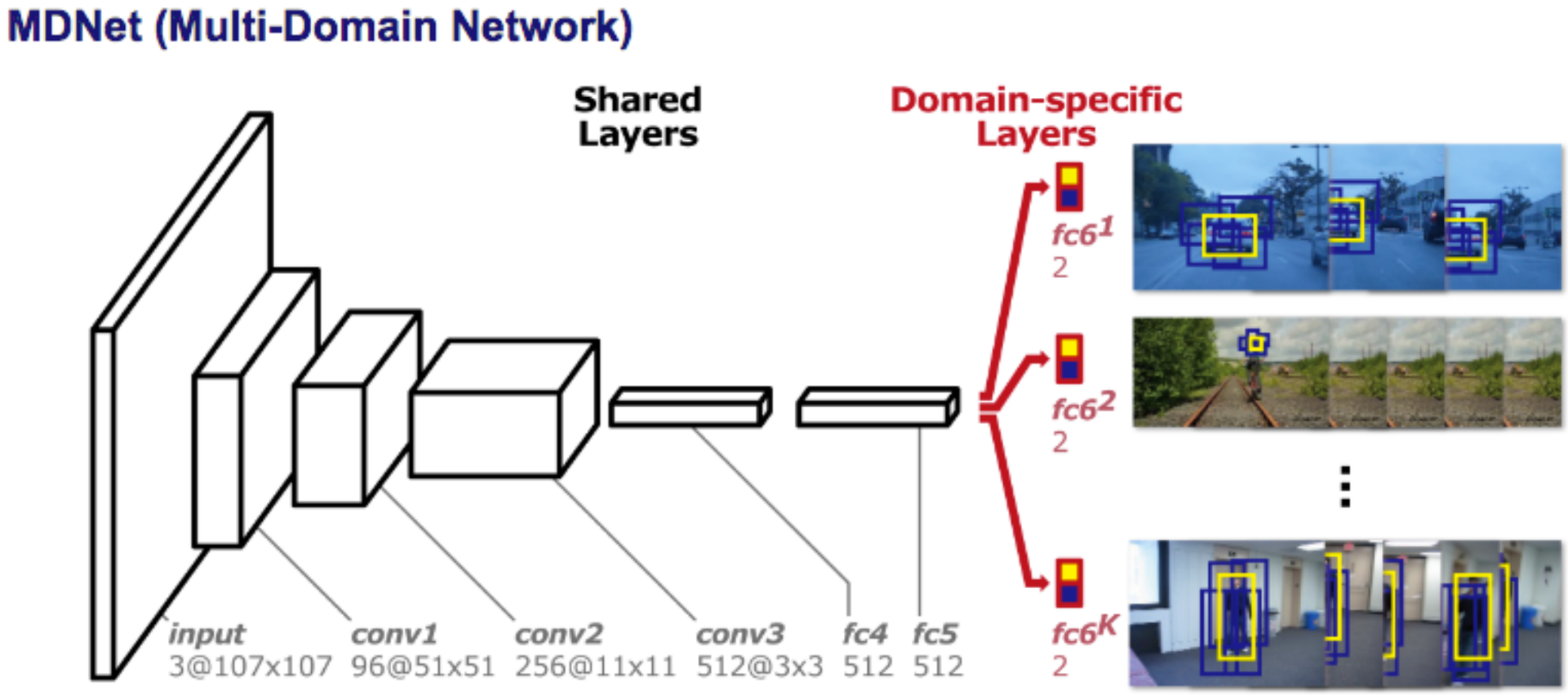
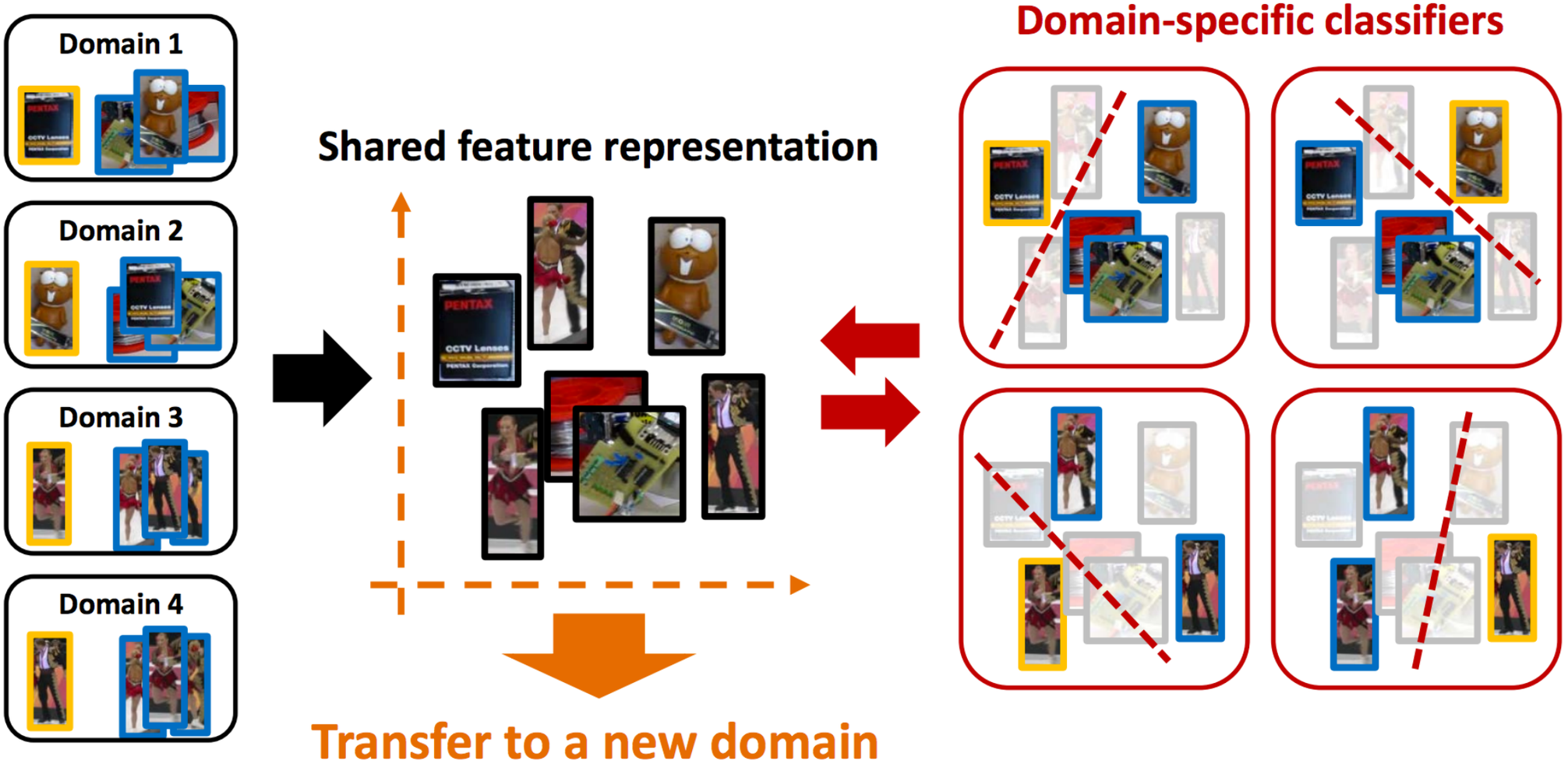


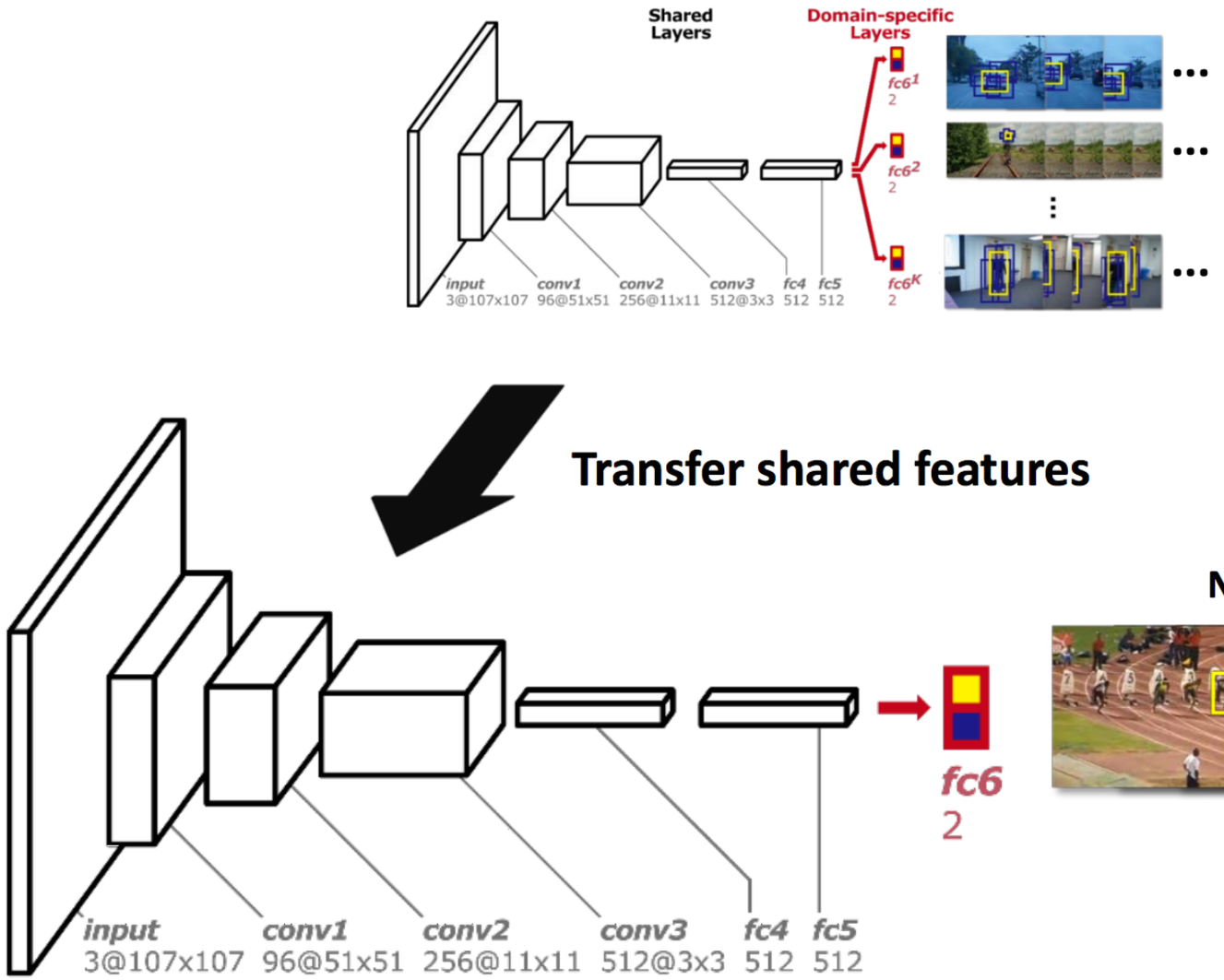

**Figure 1**. The architecture of our Multi-Domain Network (MDNet), which consists of shared layers and multiple branches of domain-specific layers. Yellow and blue bounding boxes denote the positive and negative training samples in each domain, respectively.

# Which Object is the Target?



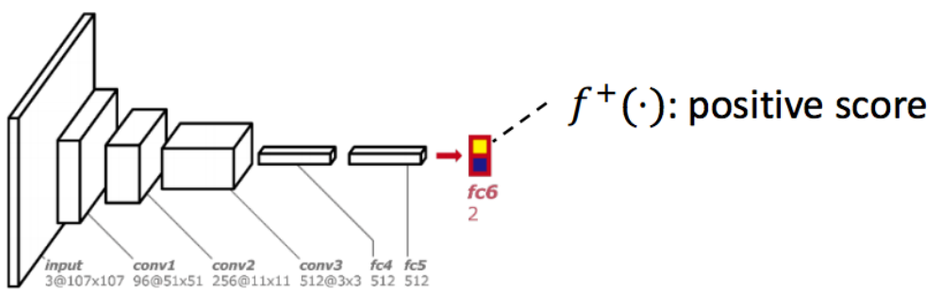

Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." arXiv preprint arXiv:1510.07945 (2015).

# Test Time: Transferring the Shared Features to Standard Convnet

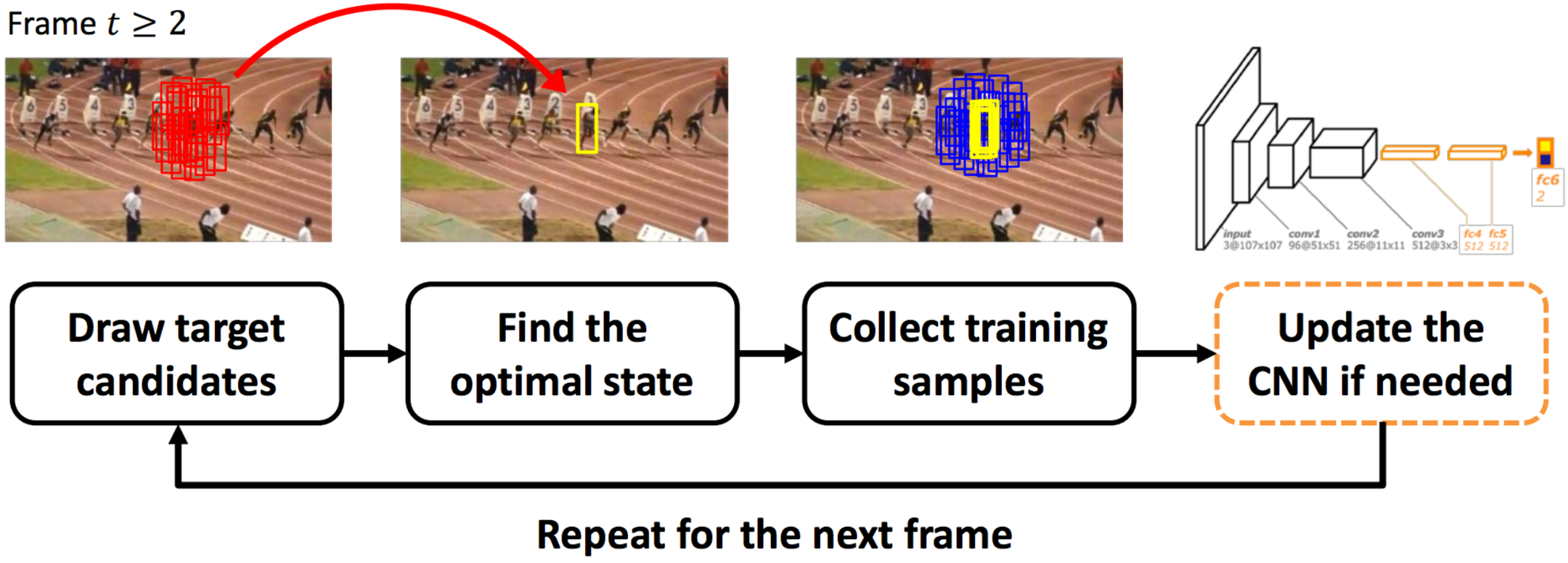



Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." arXiv preprint arXiv:1510.07945 (2015).

# Test Time: Transferring the Shared Features to Standard Convnet

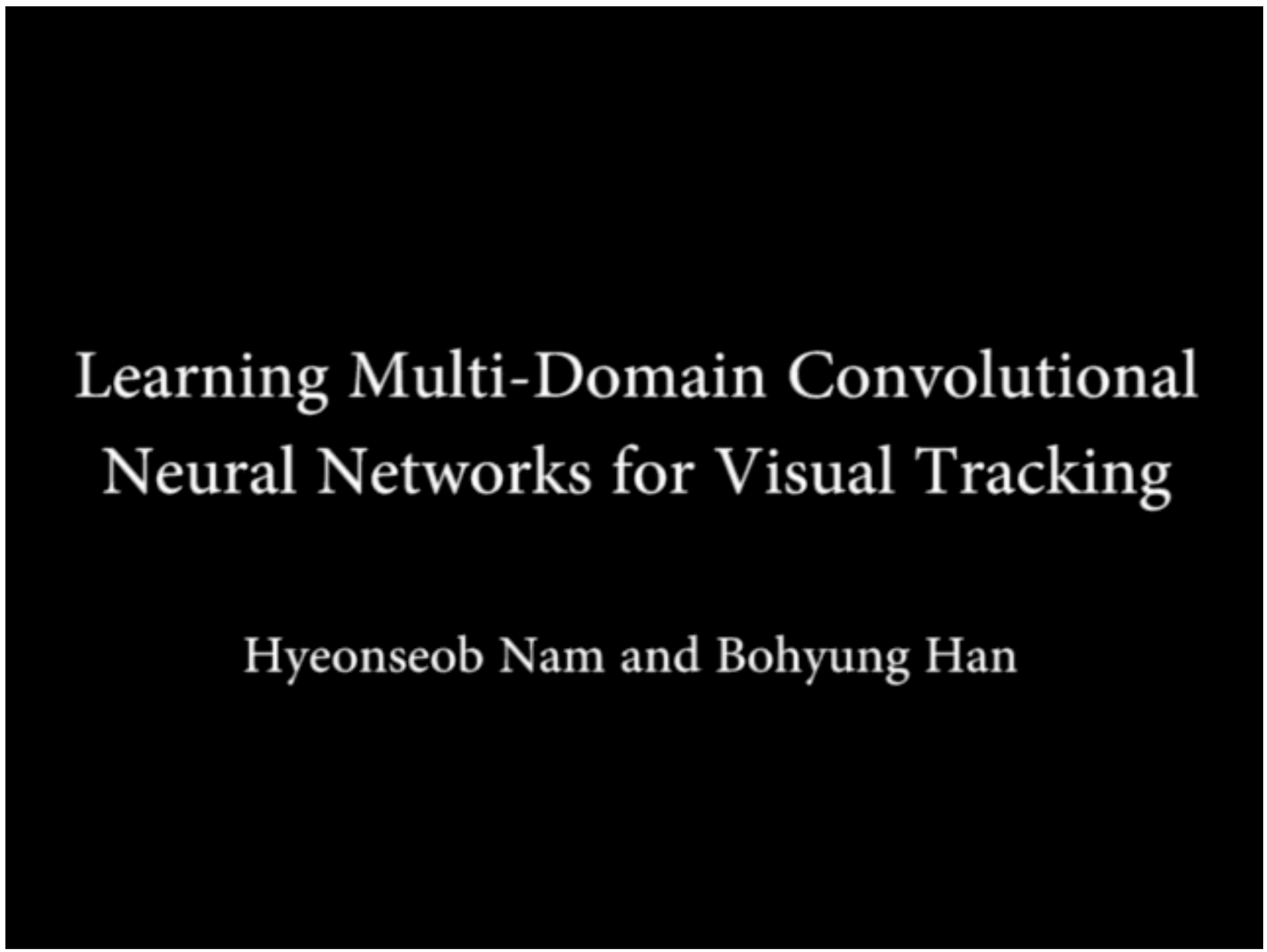



$$\mathbf{x}^* = \underset{i}{\operatorname{argmax}} f^+(\mathbf{x}^i)$$




Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." arXiv preprint arXiv:1510.07945 (2015).

# MDNet Video Results




Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." arXiv preprint arXiv:1510.07945 (2015).

# Image Segmentation

# Image Segmentation

# Fully Convolutional Networks for Semantic Segmentation

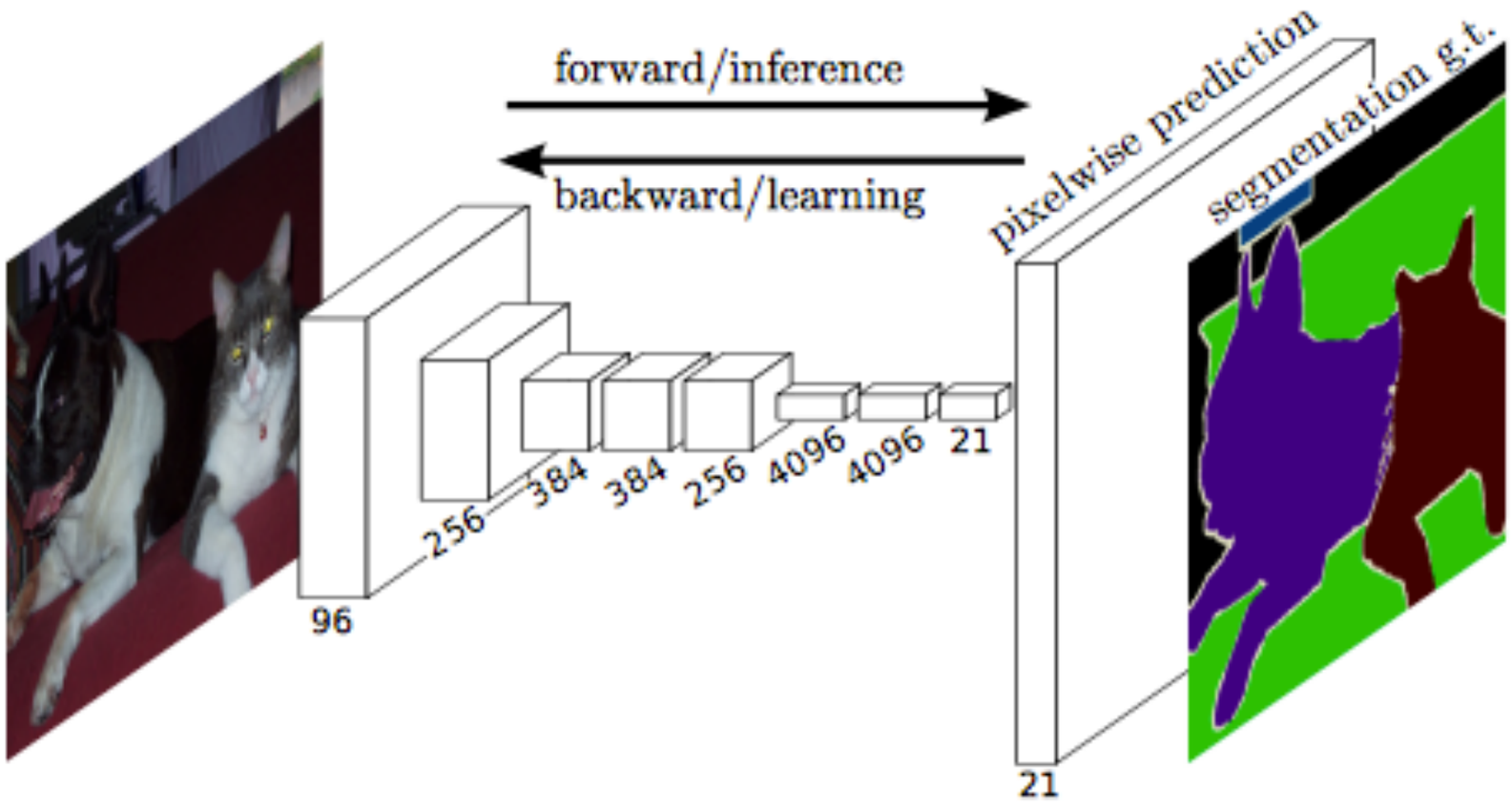



Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

[Jonathan Long, Evan Shelhamer, Yann LeCun]

# Fully Convolutional Networks for Semantic Segmentation

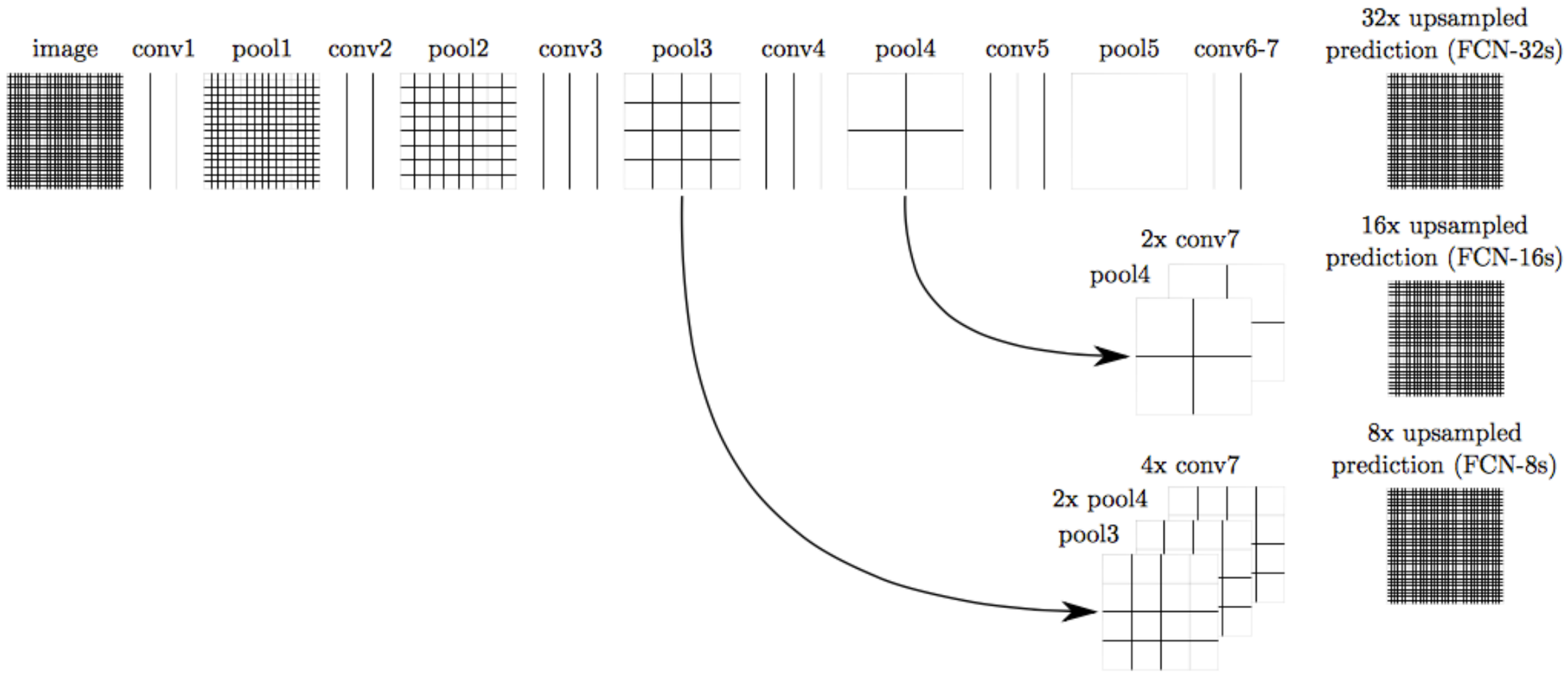



Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

[Jonathan Long, Evan Shelhamer, Yann LeCun]

# Fully Convolutional Networks for Semantic Segmentation

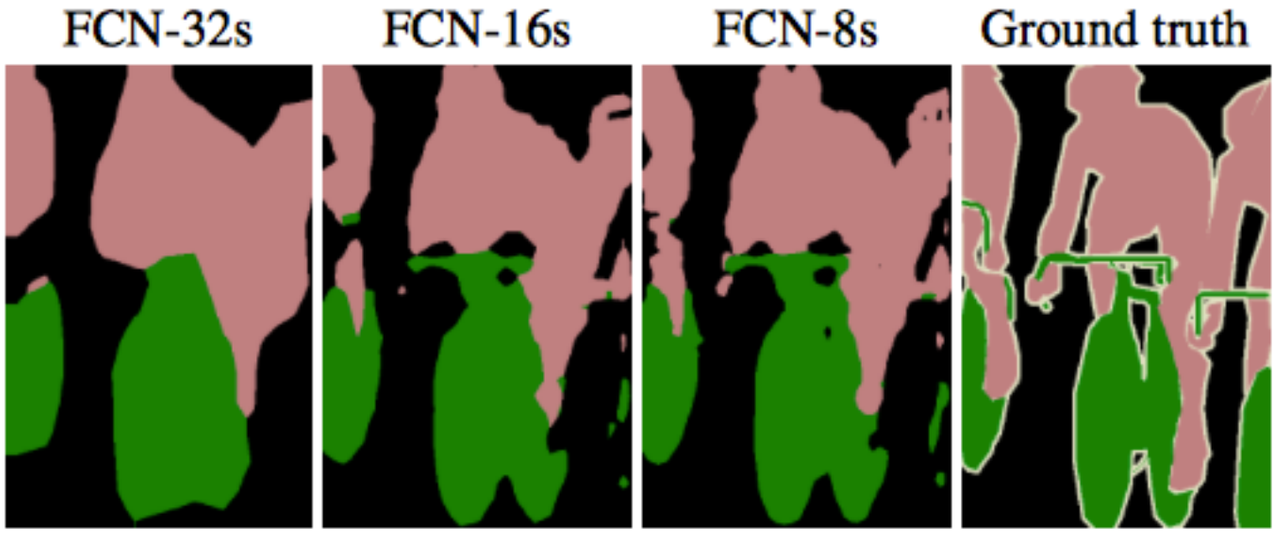



Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

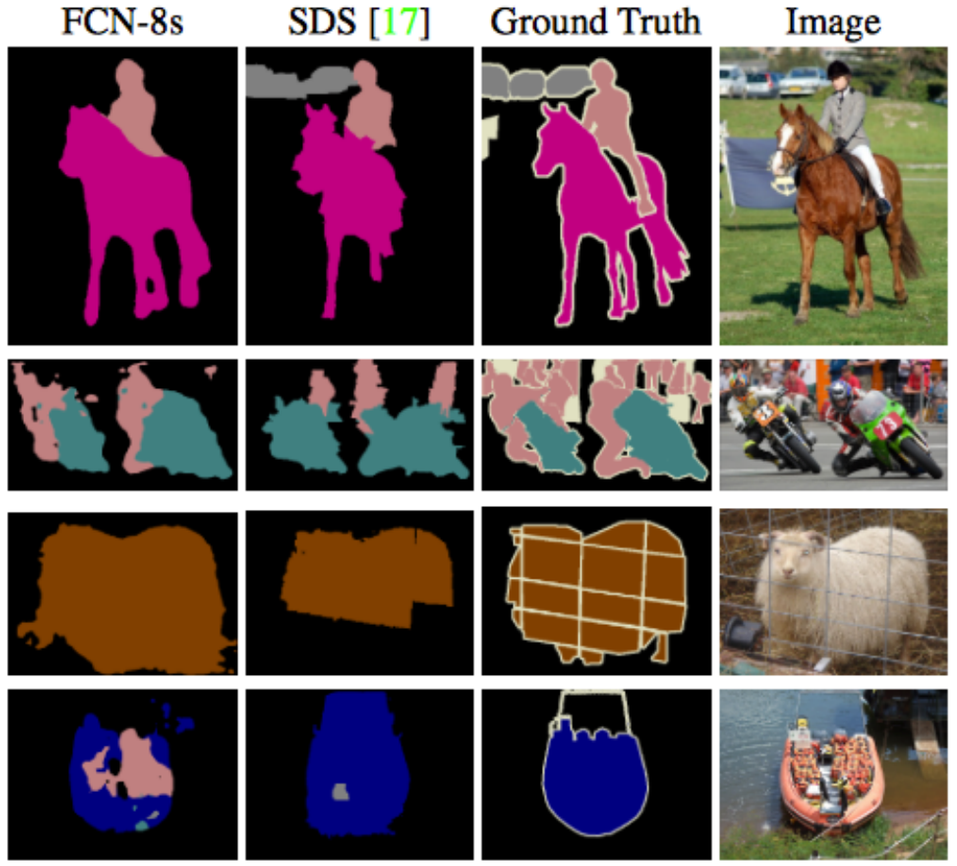



Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system by Hariharan *et al*. [17]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fourth row shows a failure case: the net sees lifejackets in a boat as people.

[Jonathan Long, Evan Shelhamer, Yann LeCun]

# U-Net: Convnet for Segmentation of Neuronal Structures in Electron Microscopic Stacks (Won the ISBI Cell Tracking Challenge 2015)

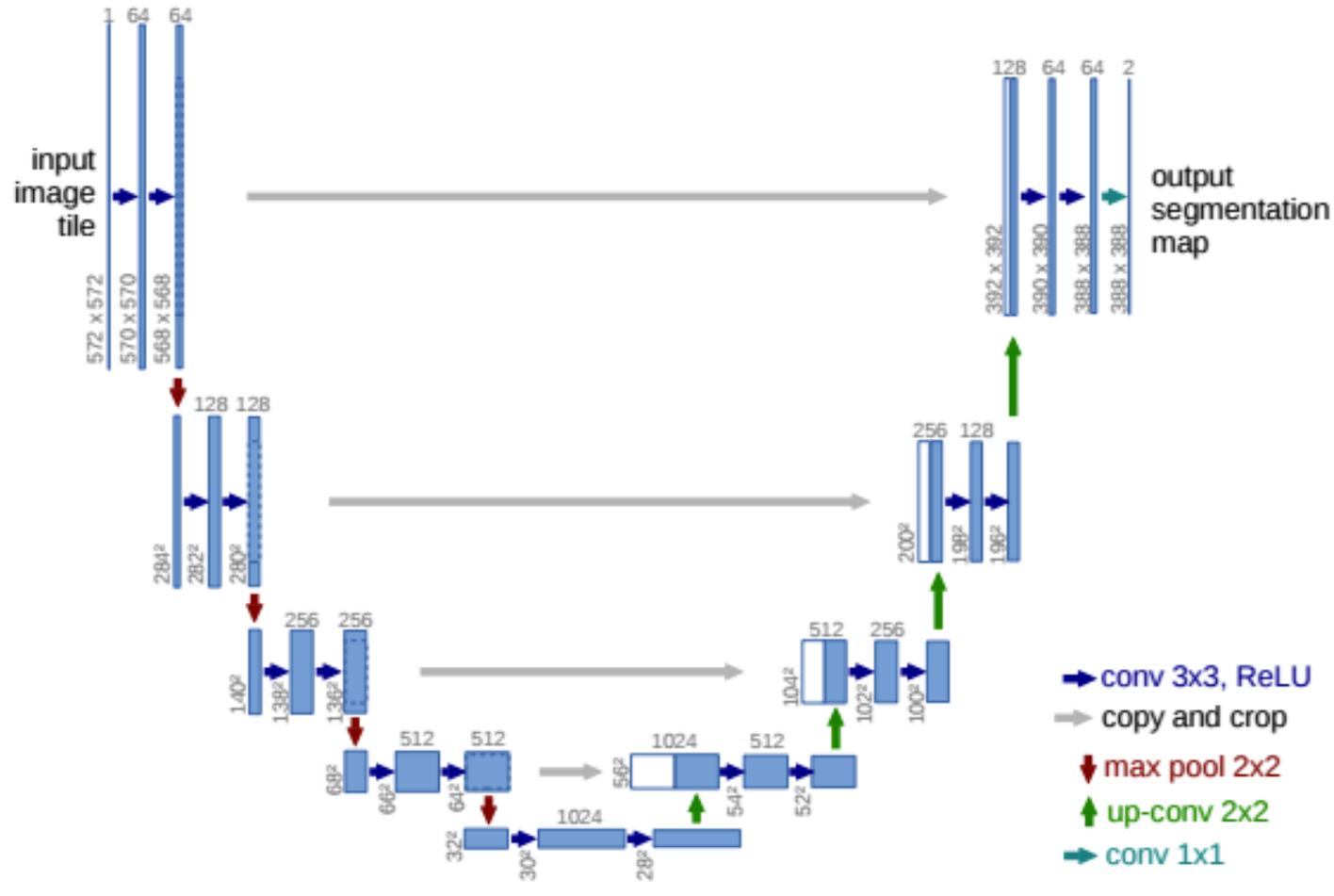



**Fig. 1.** U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

# Stereo Matching

# Stereo Convnets

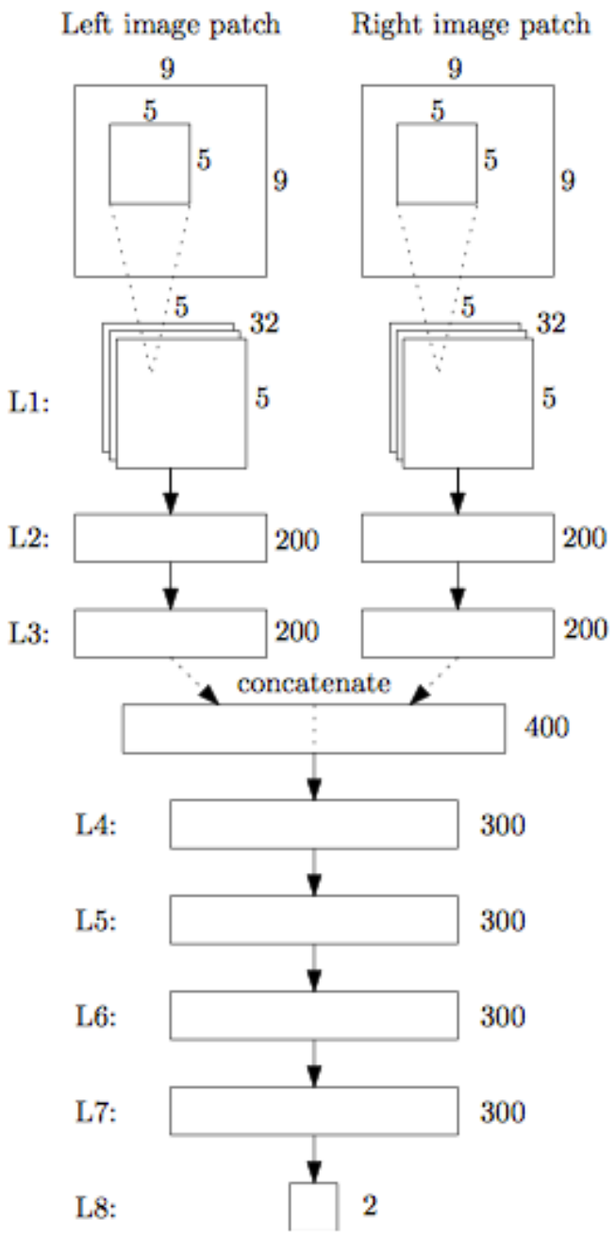



Figure 2. The architecture of our convolutional neural network.

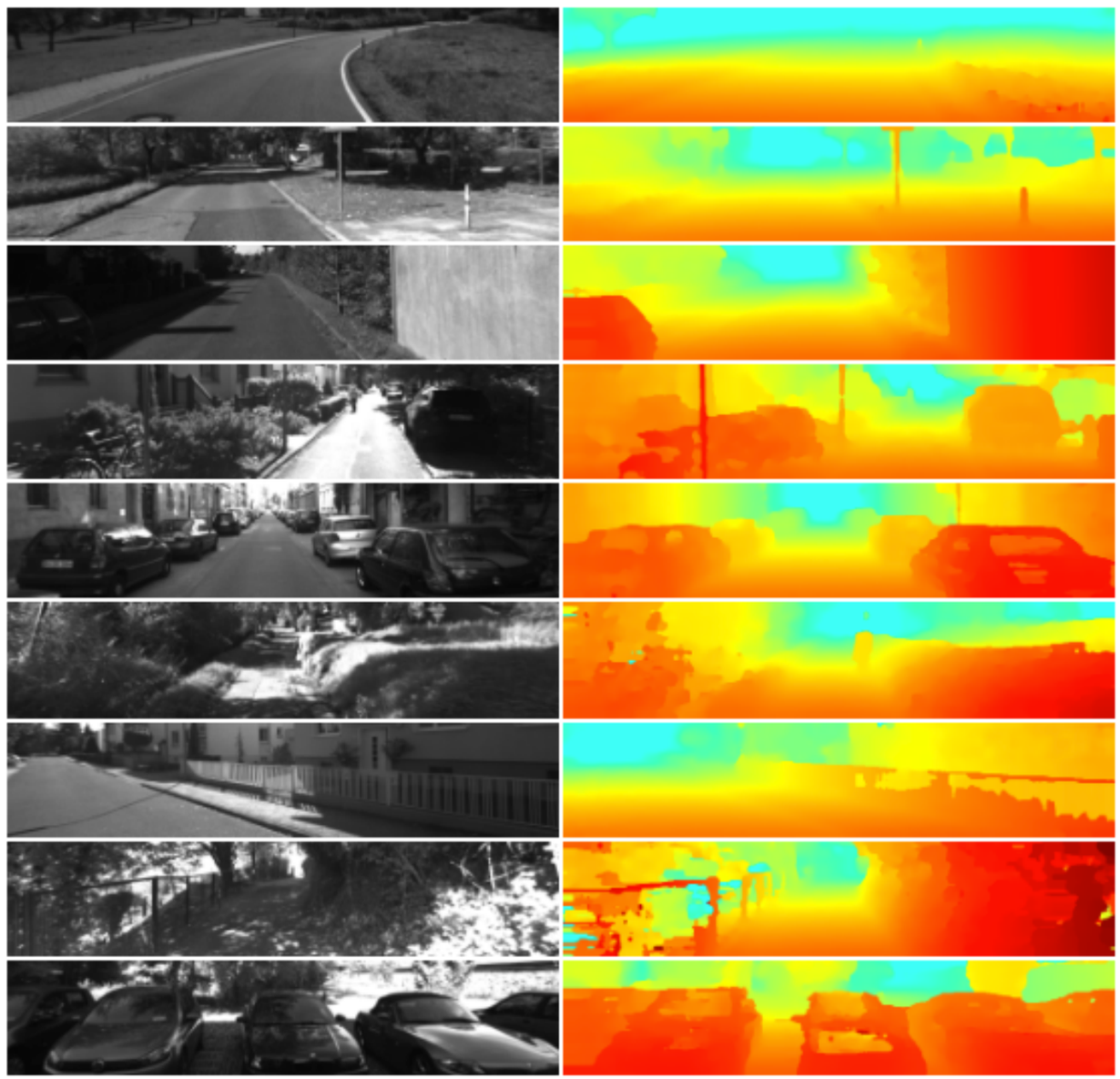


Figure 5. The left column displays the left input image, while the right column displays the output of our stereo method. Examples are sorted by difficulty, with easy examples appearing at the top. Some of the difficulties include reflective surfaces, occlusions, as well as regions with many jumps in disparity, *e.g.* fences and shrubbery. The examples towards the bottom were selected to highlight the flaws in our method and to demonstrate the inherent difficulties of stereo matching on real-world images.

[Jure Zbontar, Yann LeCun]

# Speech Synthesis

# WaveNet

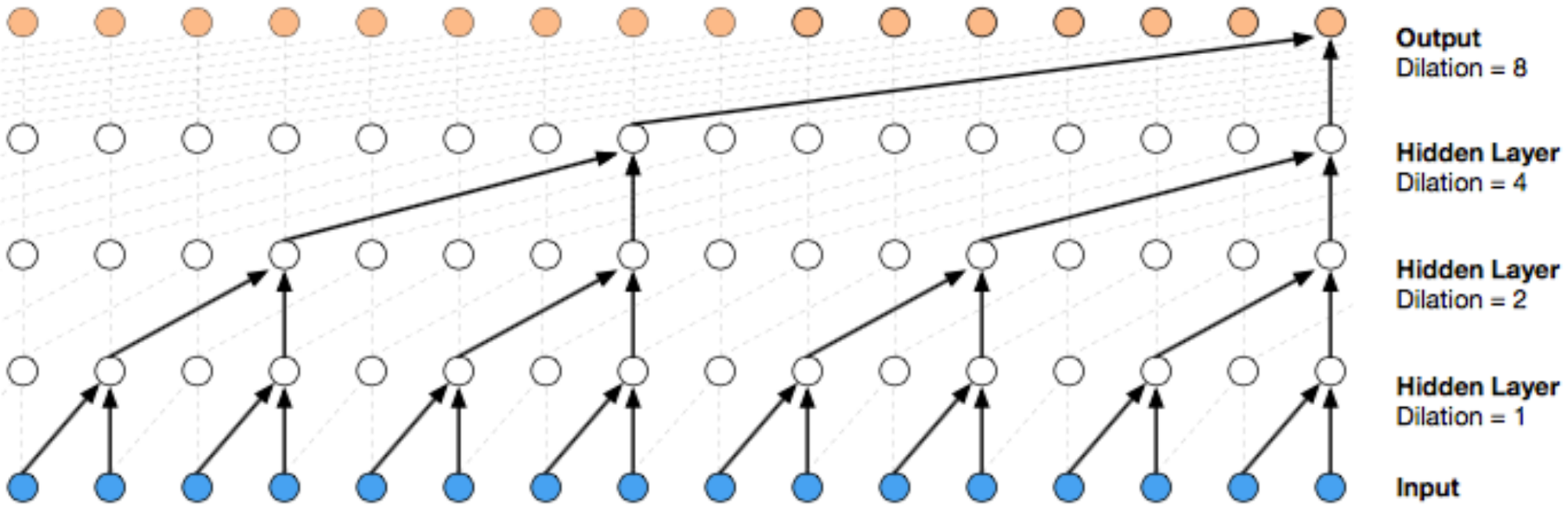
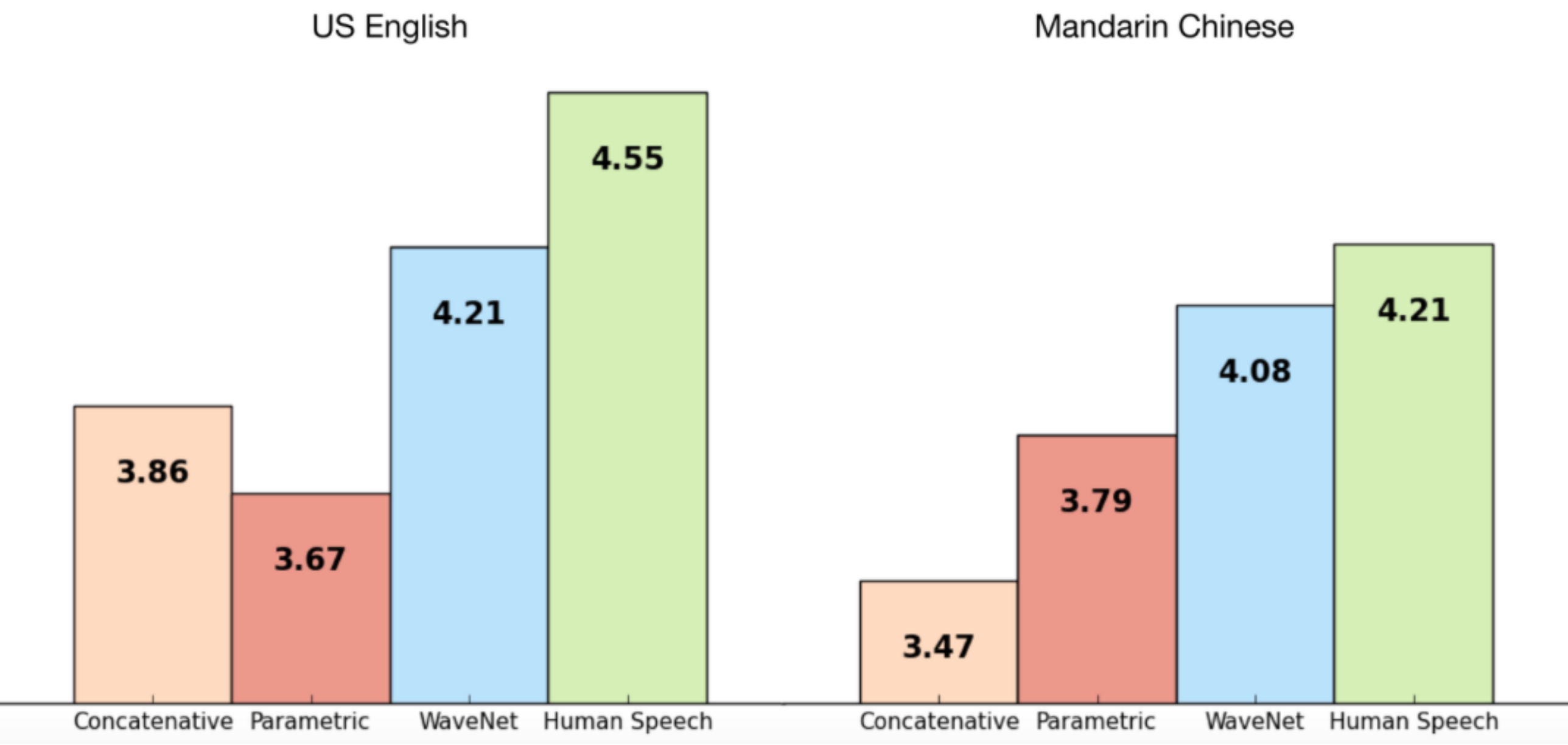



Figure 3: Visualization of a stack of *dilated* causal convolutional layers.

# WaveNet




Subjective preference scores (%) of speech samples

# WaveNet

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

# Computer Aided Diagnosis in Medical Imaging

# Convnet for Brain Tumor Segmentation (Top 4 in BRATS 2015)

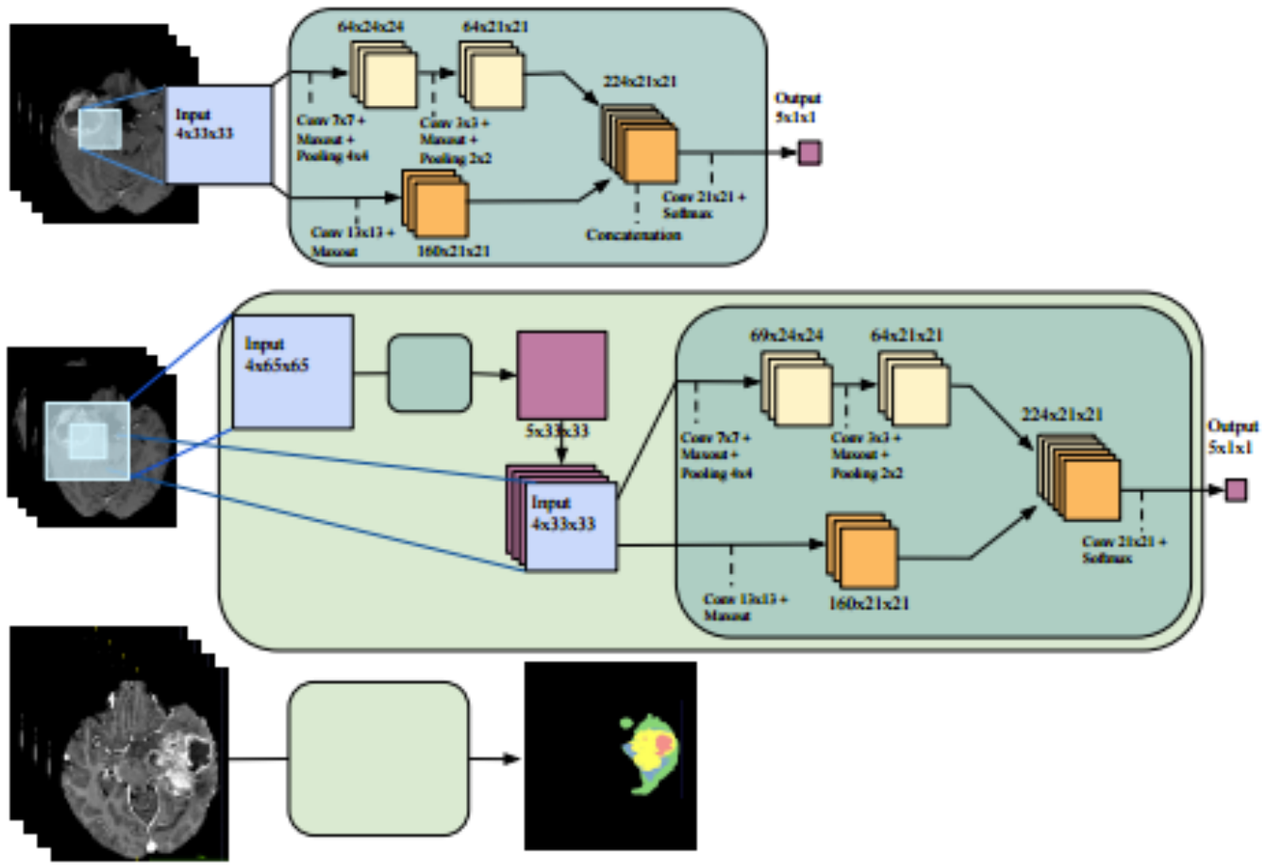



Fig. 1: The proposed architecture by Havaei et al. [35]. First row: TwoPathCNN. The input patch goes through two convolutional networks each comprising of a local and a global path. The feature maps in the local and global paths are shown in yellow and orange respectively. Second row: InputCascadeCNN. The class probabilities generated by TwoPathCNN are concatenated to the input of a second CNN model. Third row: Full image prediction using InputCascadeCNN.

# U-Net: Convnet for Segmentation of Neuronal Structures in Electron Microscopic Stacks (Won the ISBI Cell Tracking Challenge 2015)

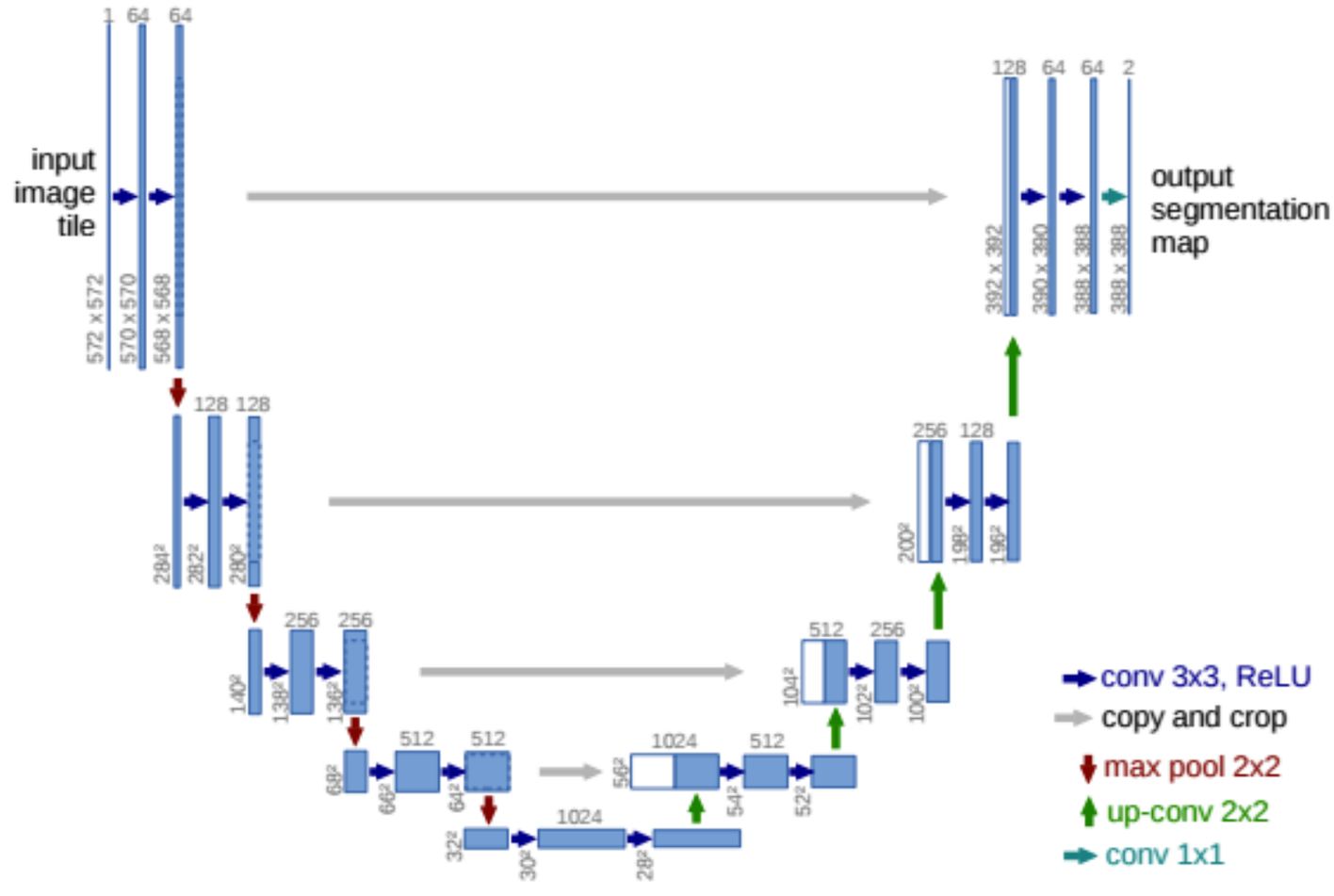
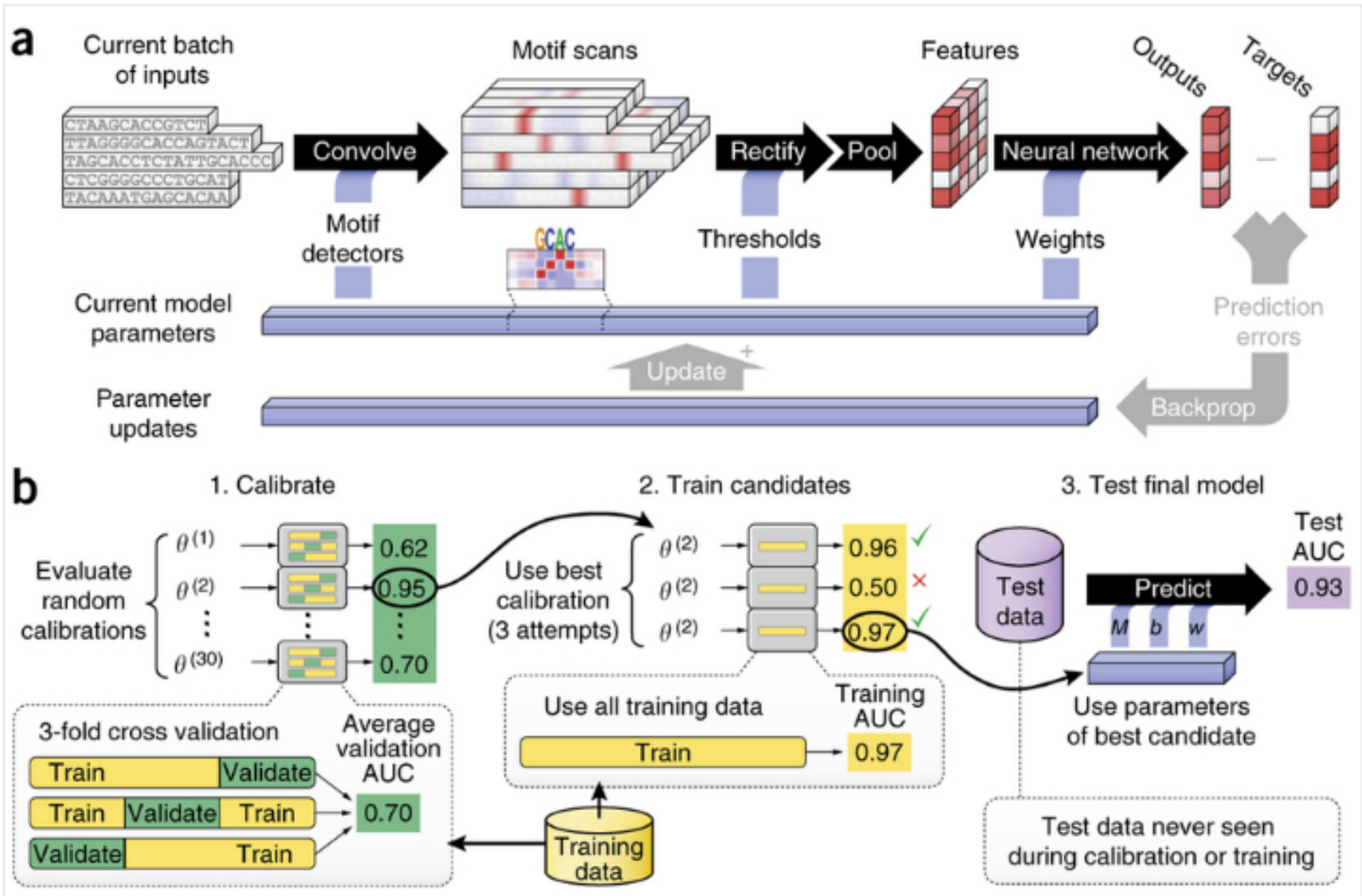



**Fig. 1.** U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

# Genetics

# DeepBind: Convnet for Predicting the Sequence Specificities of DNA- and RNA-Binding Proteins




(a) Five independent sequences being processed in parallel by a single DeepBind model. The convolve, rectify, pool and neural network stages predict a separate score for each sequence using the current model parameters (Supplementary Notes, sec. 1). During the training phase, the backprop and update stages simultaneously update all motifs, thresholds and network weights of the model to improve prediction accuracy. (b) The calibration, training and testing procedure used throughout (Supplementary Notes, sec. 2).

# Fashion

# Neuroaesthetics in Fashion: Modeling the Perception of Fashionability

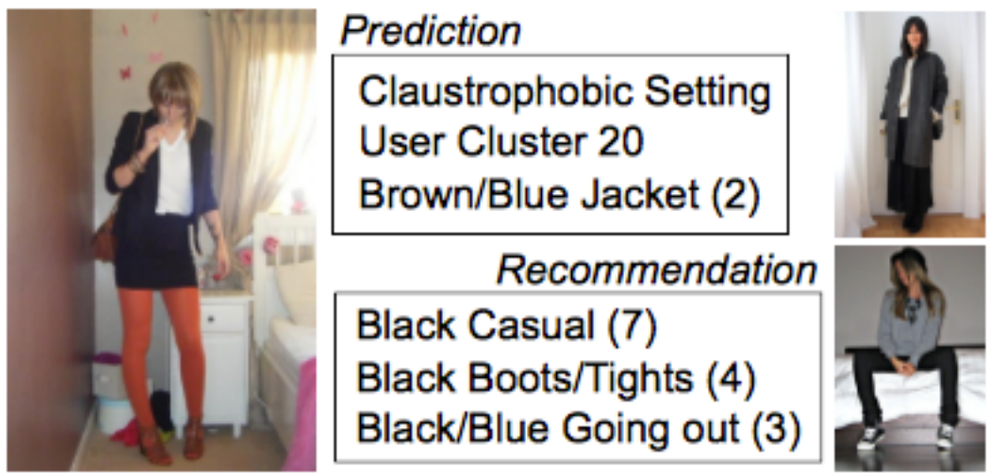



Figure 1: Example of recommendations provided by our model for the post on the left. In this case the user is wearing what we have identified as "Brown/Blue Jacket". This photograph obtains a score of 2 out of 10 in fashionability. Additionally the user is classified as belonging to cluster 20 and took a picture in the "Claustrophobic" setting. If the user were to wear a "Black Casual" outfit as seen on the right, our model predicts she would improve her fashionability to 7 out of 10. This prediction is conditioned on the user, setting and other factors allowing the recommendations to be tailored to each particular user.

[Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, Raquel Urtasun]

# Neuroaesthetics in Fashion: Modeling the Perception of Fashionability

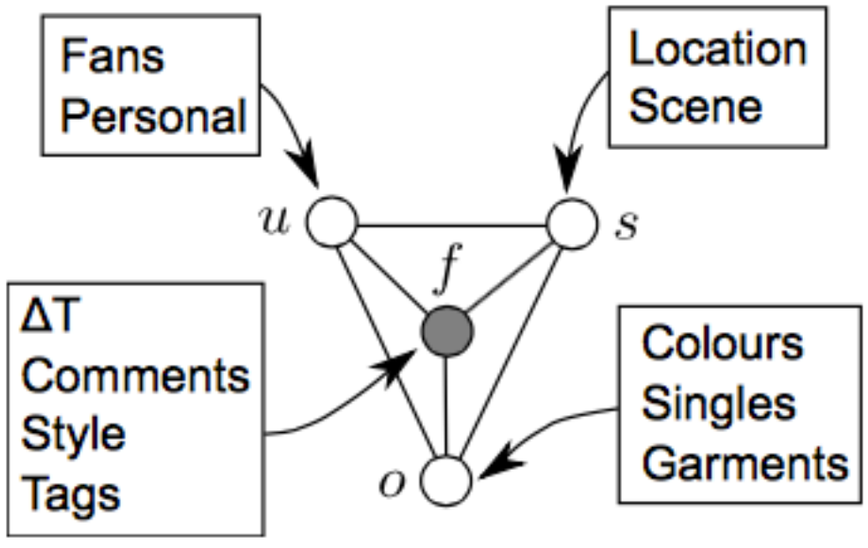



Figure 4: An overview of the CRF model and the features used by each of the nodes.

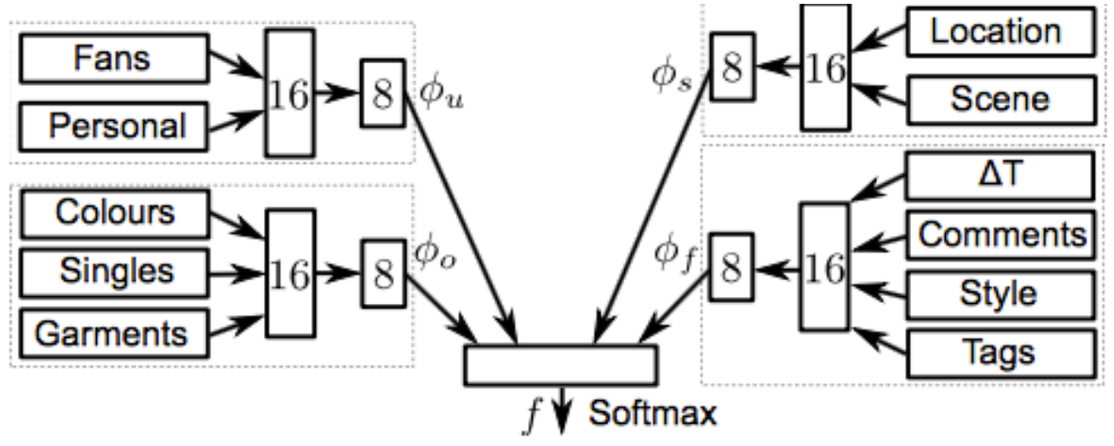



Figure 5: Illustration of the type of deep network architecture to learn features. We can see that it consists of four network joined together by a softmax layer. The output of the different networks $\phi_f$, $\phi_o$, $\phi_u$, and $\phi_s$ are then used as features for the CRF.

[Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, Raquel Urtasun]

# Unsupervised Learning

# Neuroaesthetics in Fashion: Modeling the Perception of Fashionability

Current Outfit:
Pink Outfit (3)

Recommendations:
Heels (8)
Pastel Shirts/Skirts (8)
Black/Gray Tights/Sweater (5)

Current Outfit:
Pink/Blue Shoes/Dress Shorts (3)

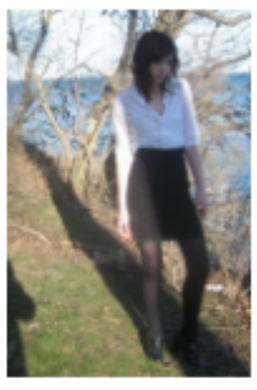
Recommendations:
Black/Gray Tights/Sweater (5)
Black Casual (5)
Black Boots/Tights (5)

Current Outfit:
Pink/Black Misc. (5)

Recommendations:
Pastel Dress (8)
Black/Blue Going out (8)
Black Casual (8)

Current Outfit:
Blue with Scarf (3)

Recommendations:
Heels (8)
Pastel Shirts/Skirts (8)
Black Casual (8)



Current Outfit:
Pink/Blue Shoes/Dress Shorts (3)

Recommendations:
Black Casual (7)
Black Heavy (3)
Navy and Bags (3)

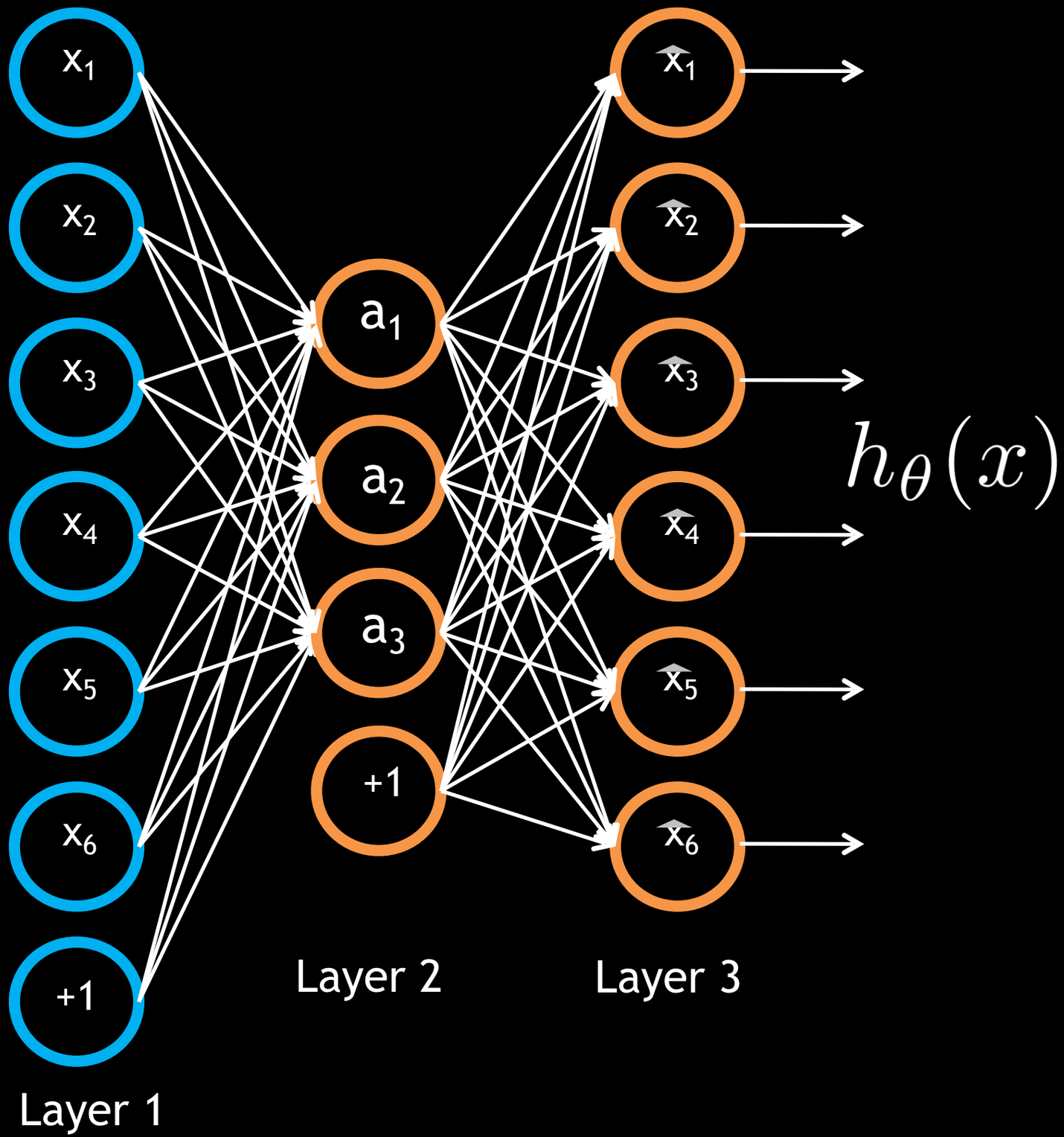
Current Outfit:
Formal Blue/Brown (5)

Recommendations:
Pastel Shirts/Skirts (9)
Black/Blue Going out (8)
Black Boots/Tights (8)

Figure 10: Example of recommendations provided by our model. In parenthesis we show the predicted fashionability.

[Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, Raquel Urtasun]

# Unsupervised feature learning with a neural network




Autoencoder.

Network is trained to output the input (learn identify function).

$$h_\theta(x) \approx x$$

Trivial solution unless:
- Constrain number of units in Layer 2 (learn compressed representation), or
- Constrain Layer 2 to be **sparse**.

# Unsupervised feature learning with a neural network

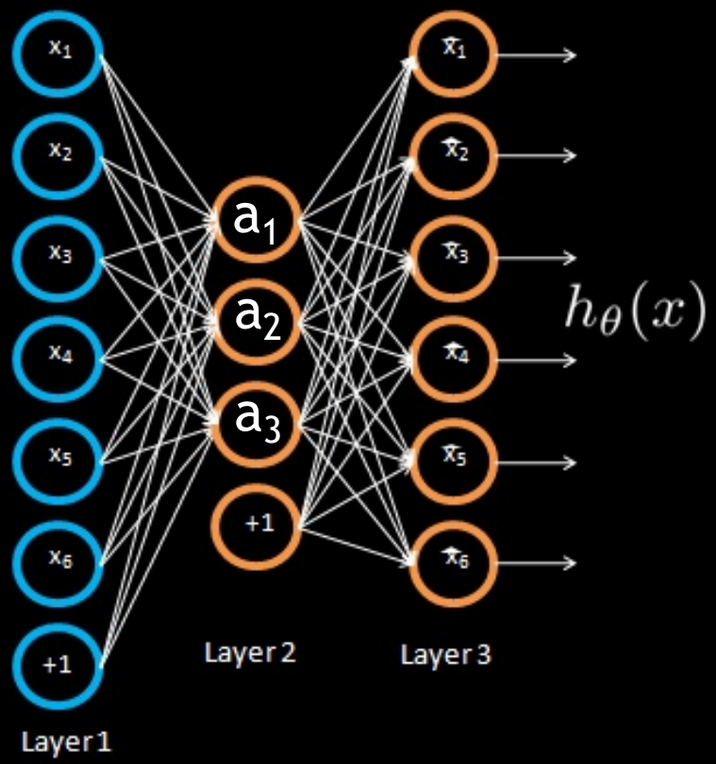
Training a sparse autoencoder.

Given unlabeled training set $x_1, x_2, \ldots$

$$\min_{\theta} \|h_\theta(x) - x\|^2 + \lambda \sum_i |a_i|$$

Reconstruction error term

$L_1$ sparsity term



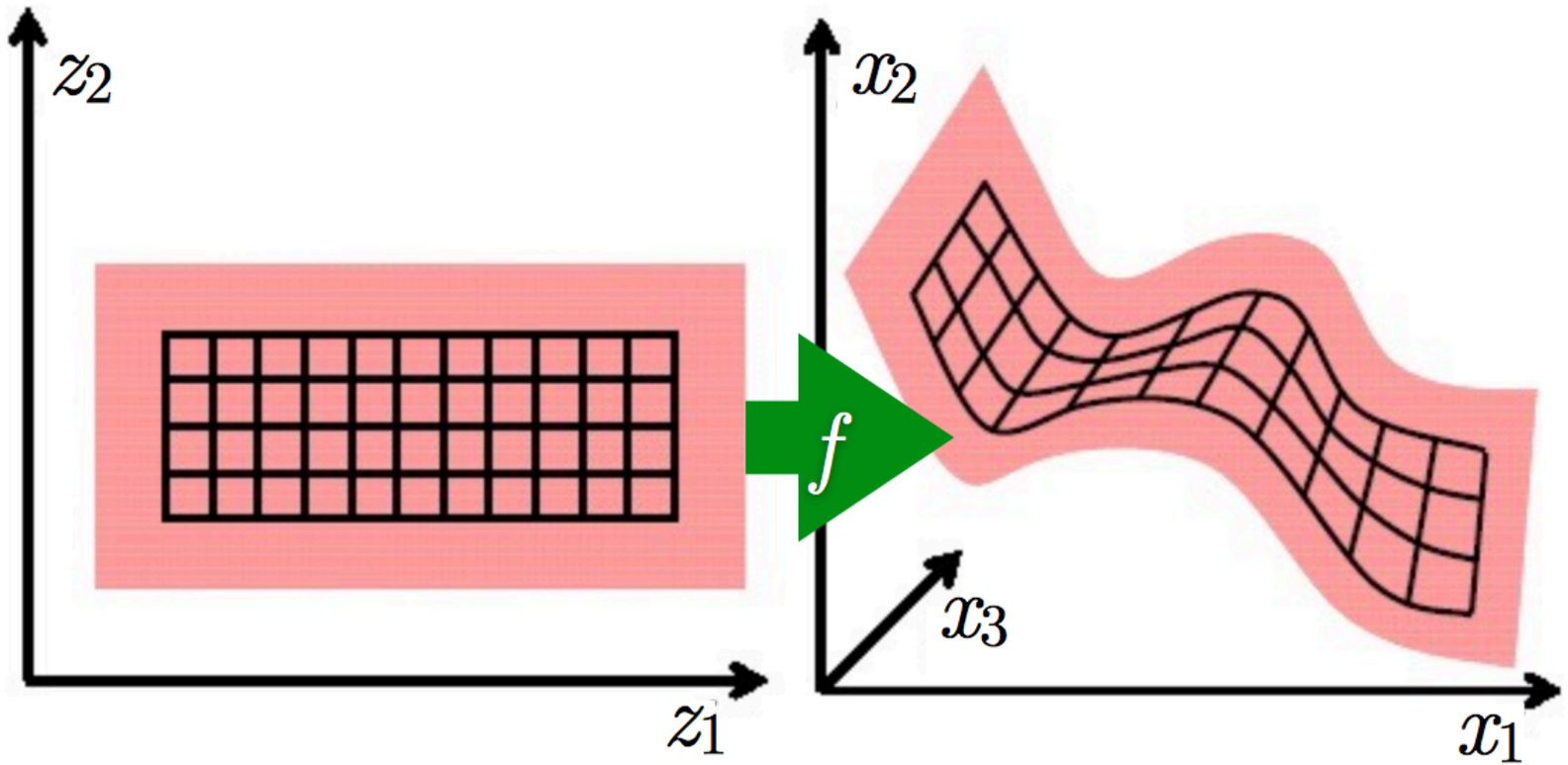
# Variational Autoencoder

- **latent variable model:** learn a mapping from some latent variable $z$ to a complicated distribution on $x$.

$$p(x) = \int p(x, z)\, dz \quad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})$$

$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = f(\boldsymbol{z})$$

- Can we learn to decouple the true **explanatory factors** underlying the data distribution? E.g. separate identity and expression in face images

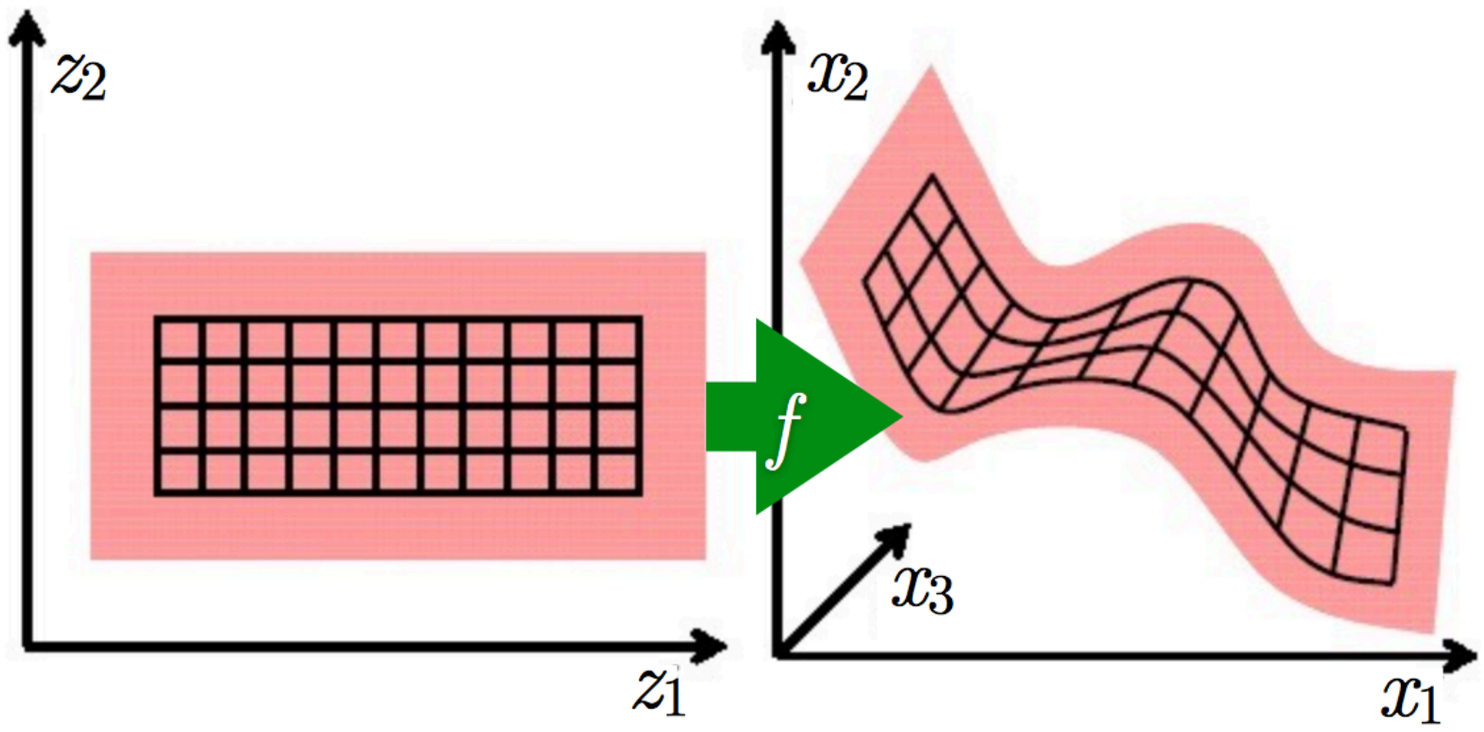


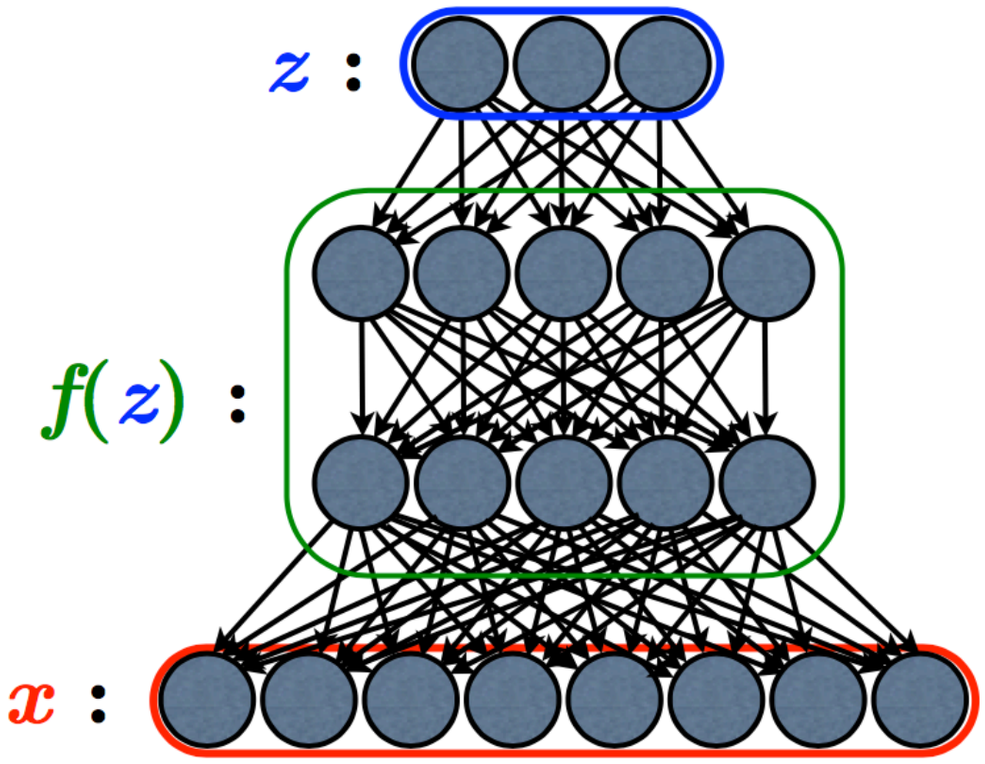
# Variational Autoencoder

- Leverage ***neural networks*** to learn a latent variable model.

$$p(x) = \int p(x, z)\ dz \quad \text{where} \quad p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$$

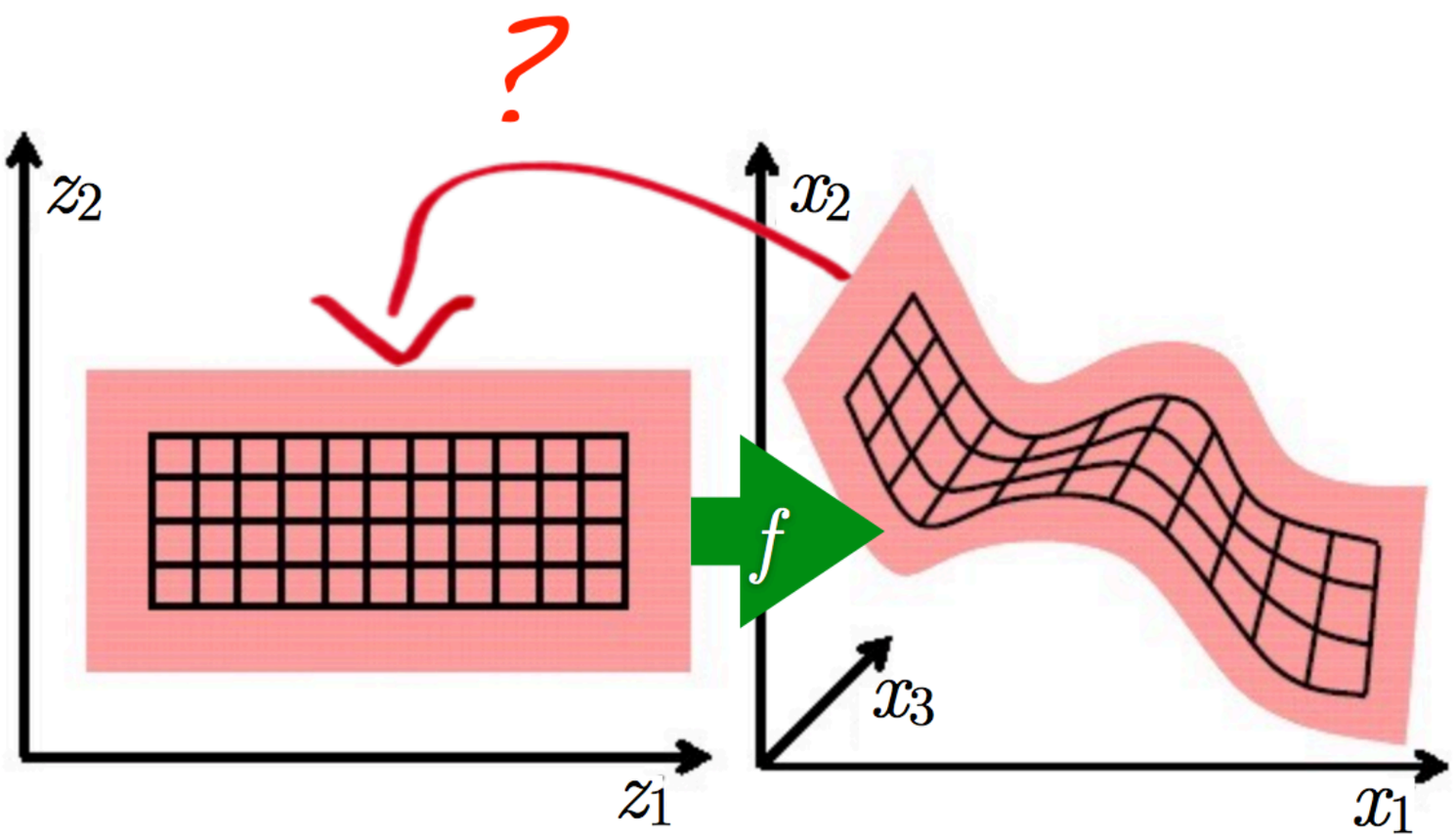
$$p(\boldsymbol{z}) = \text{something simple} \qquad p(\boldsymbol{x} \mid \boldsymbol{z}) = f(\boldsymbol{z})$$

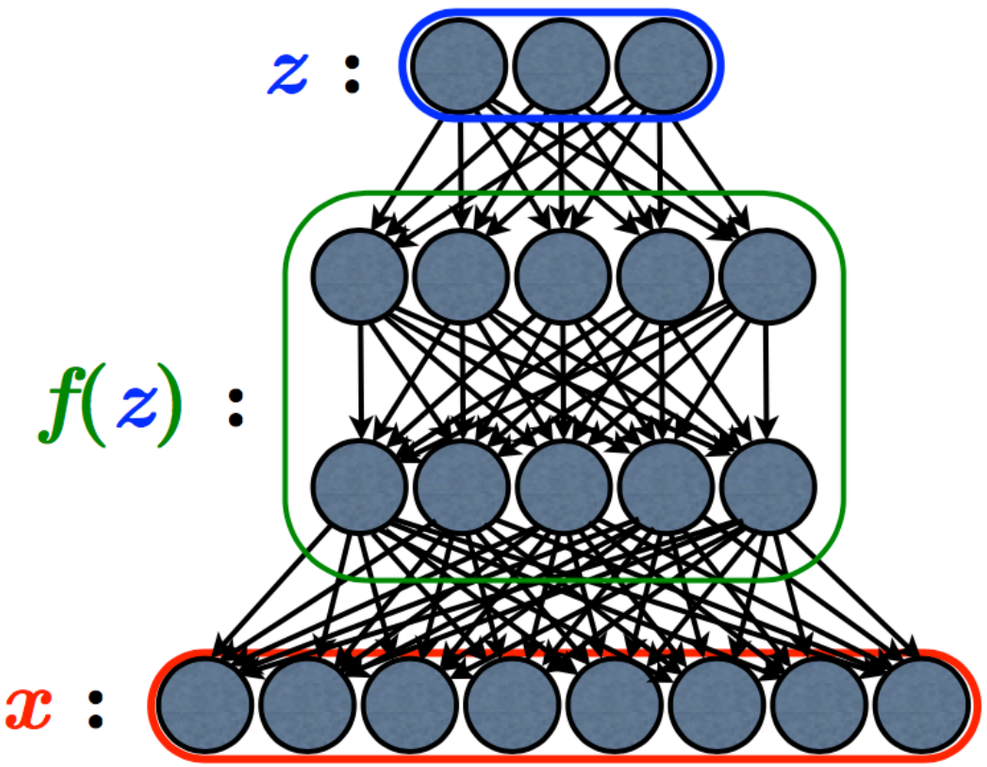


# Inference/Learning Challenge

- **Where does $z$ come from?** — The classic directed model dilemma.
- Computing the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ is intractable.
- We need it to train the directed model.



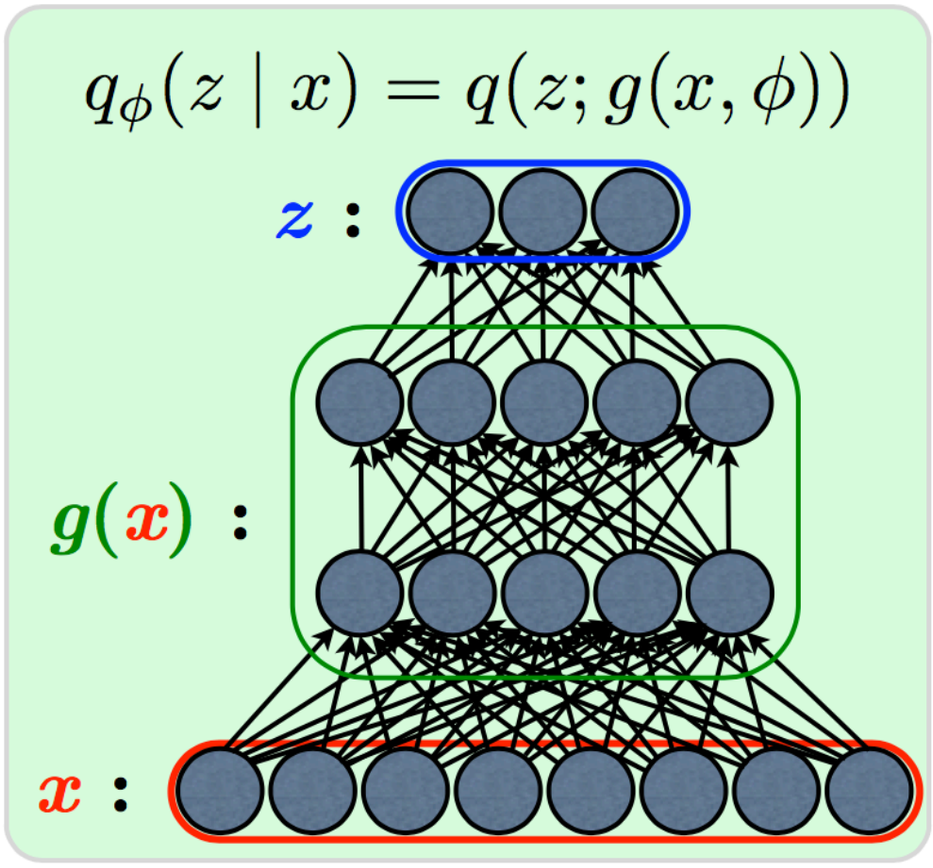



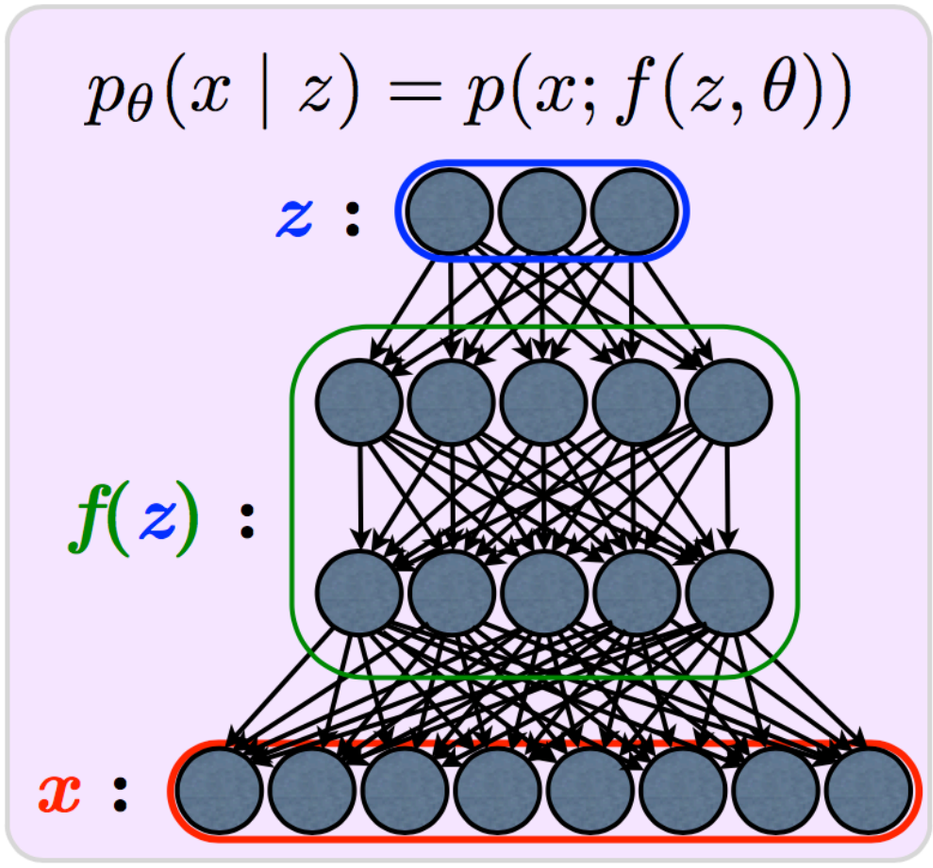
# Variational Autoencoder

- The **VAE approach**: introduce an inference model $q_\phi(z \mid x)$ that **learns** to approximates the intractable posterior $p_\theta(z \mid x)$ by optimizing the variational lower bound:

$$\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| \, p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$$

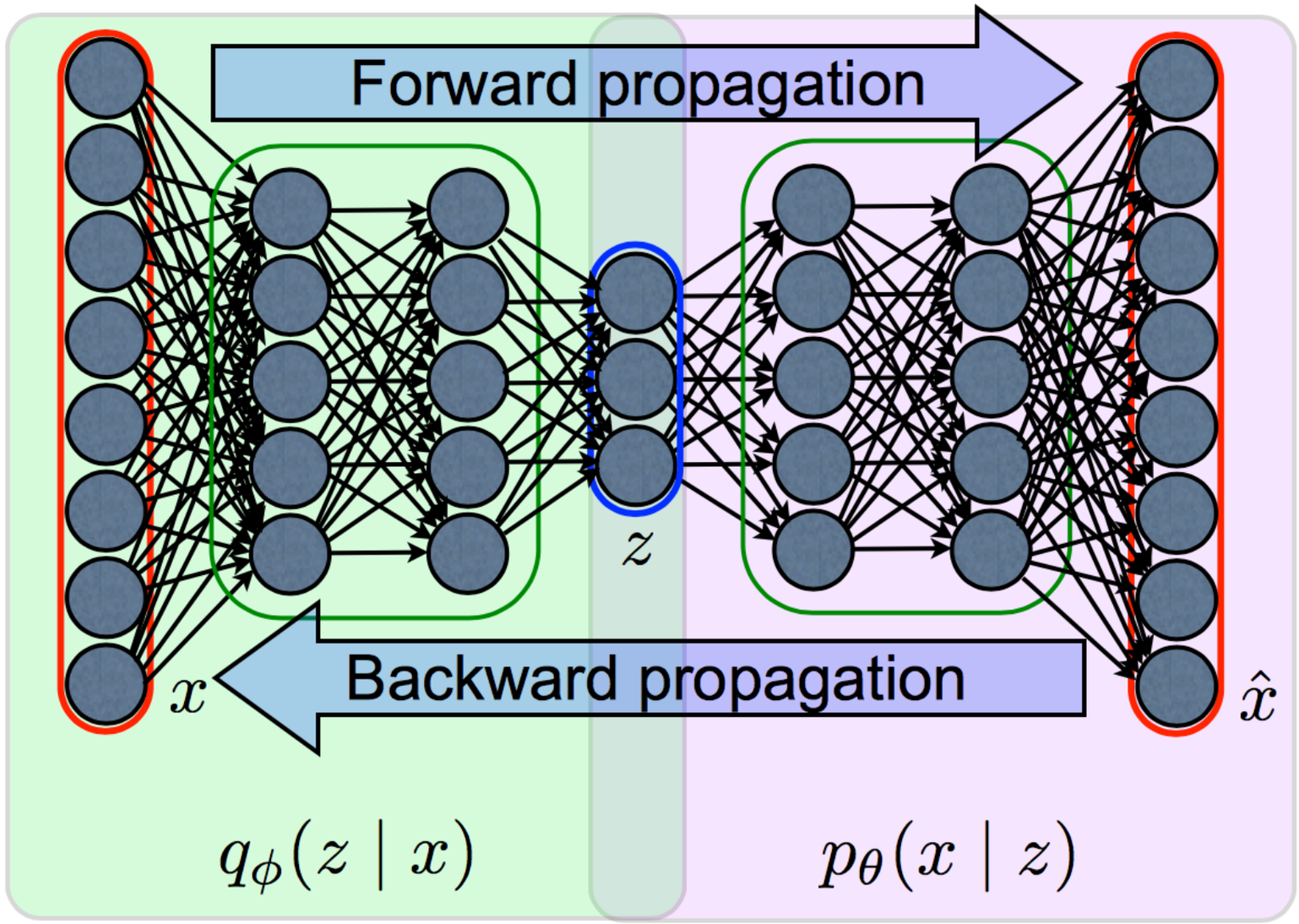
- We parameterize $q_\phi(z \mid x)$ with another neural network:



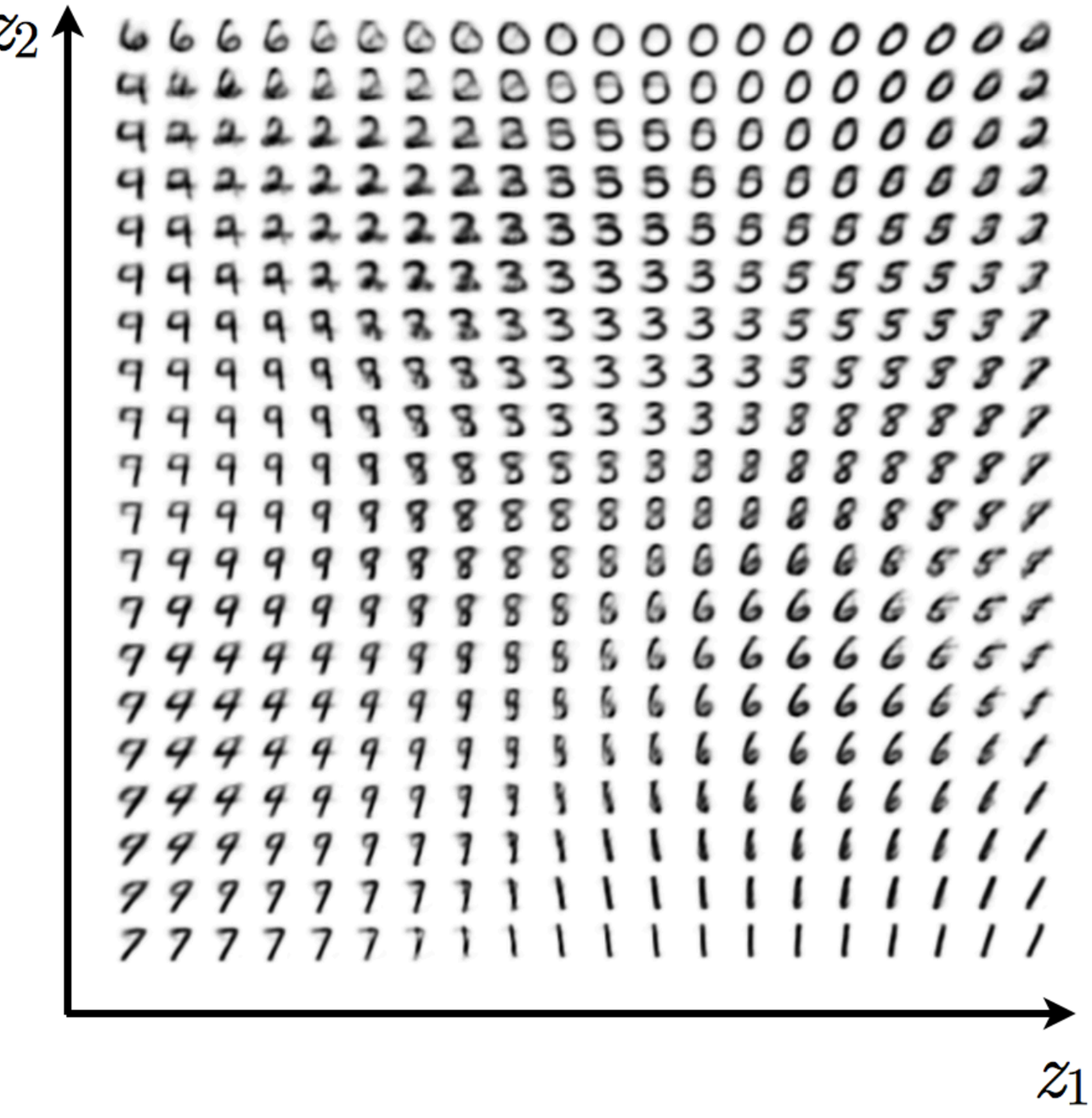
# Variational Autoencoder

Objective function: $\mathcal{L}(\theta, \phi, x) = -D_{\mathrm{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$

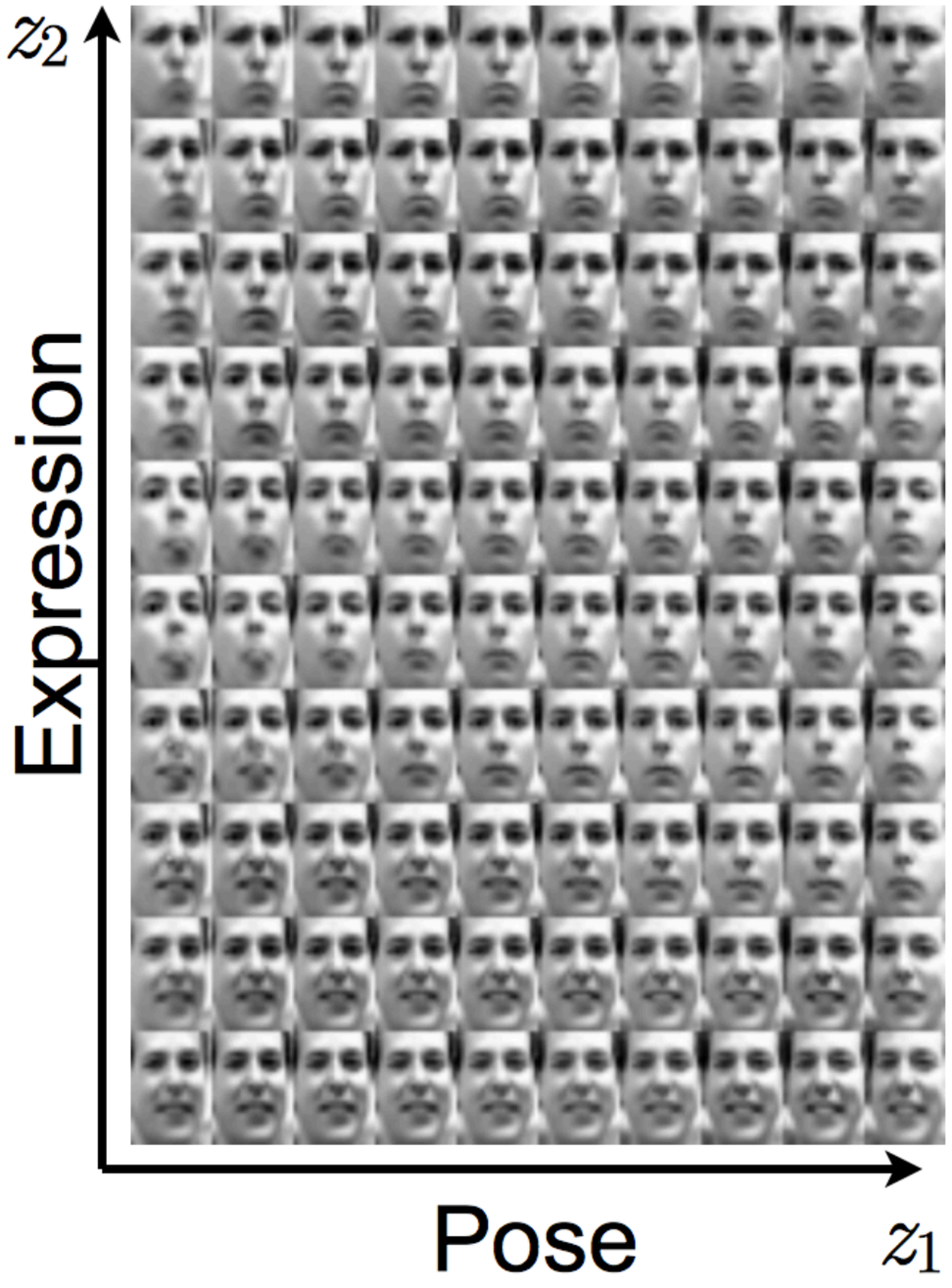


# Variational Autoencoder

MNIST:



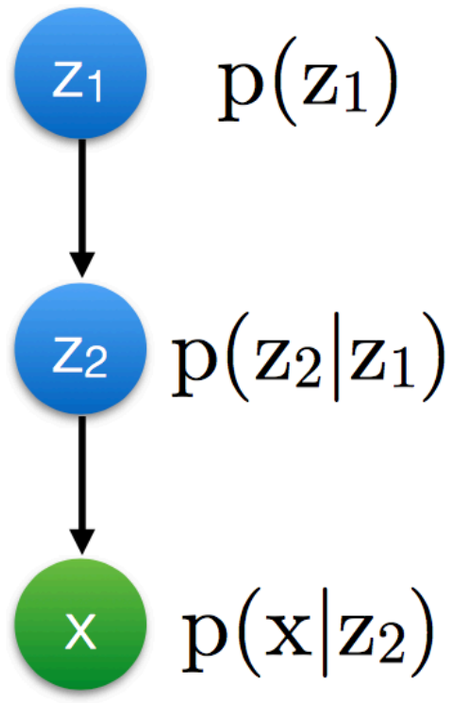

Frey Face dataset:



# Deep Latent-Variable Model

- We combining the strengths of **deep neural nets** with those of **latent-variable models**
  - **directed latent variables models**: can represent complicated **marginal distributions** over $\mathrm{x}$
  - probabilistic **deep neural nets**: can represent complicated conditional dependencies $\mathrm{p}(\mathrm{y}|\mathrm{x}) = \mathrm{f}(\mathrm{x},\mathrm{y})$
- Intractable posterior distribution $\mathrm{p}(\mathrm{z}|\mathrm{x})$

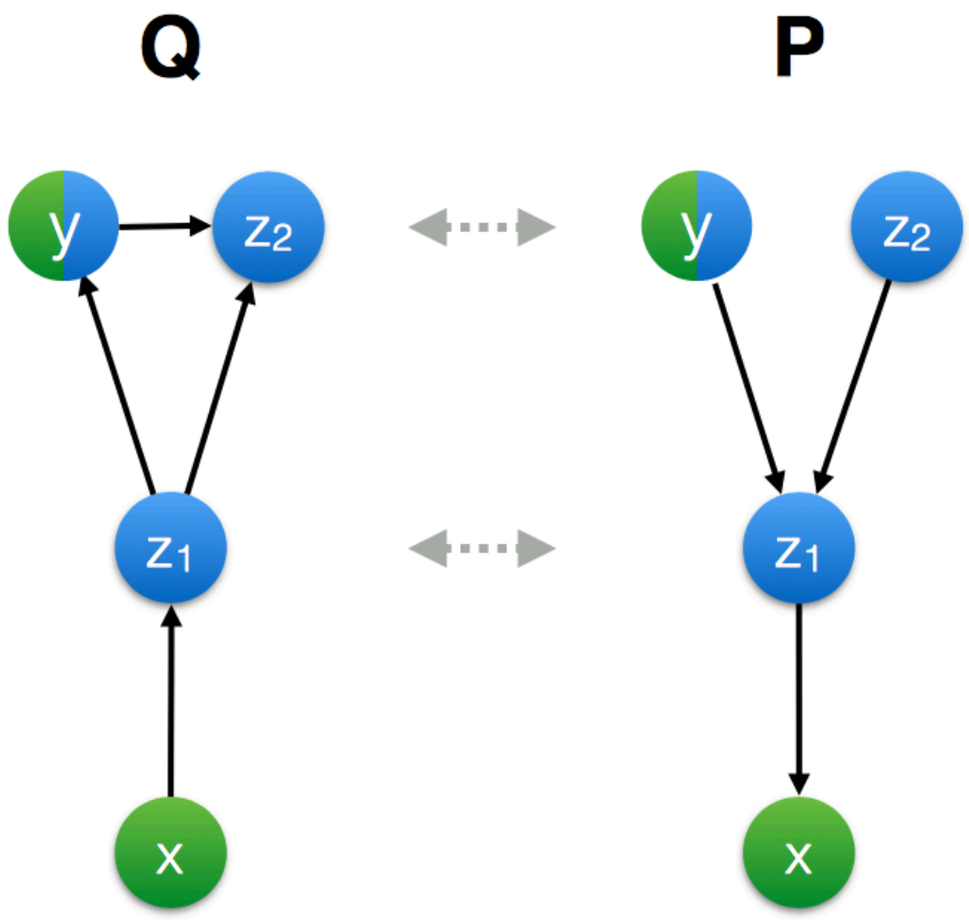
  $=>$ Approximate inference




$$\mathrm{p}(\mathrm{x},\mathrm{z}_1,\mathrm{z}_2) = \mathrm{p}(\mathrm{x}|\mathrm{z}_2)\mathrm{p}(\mathrm{z}_2|\mathrm{z}_1)\mathrm{p}(\mathrm{z}_1)$$

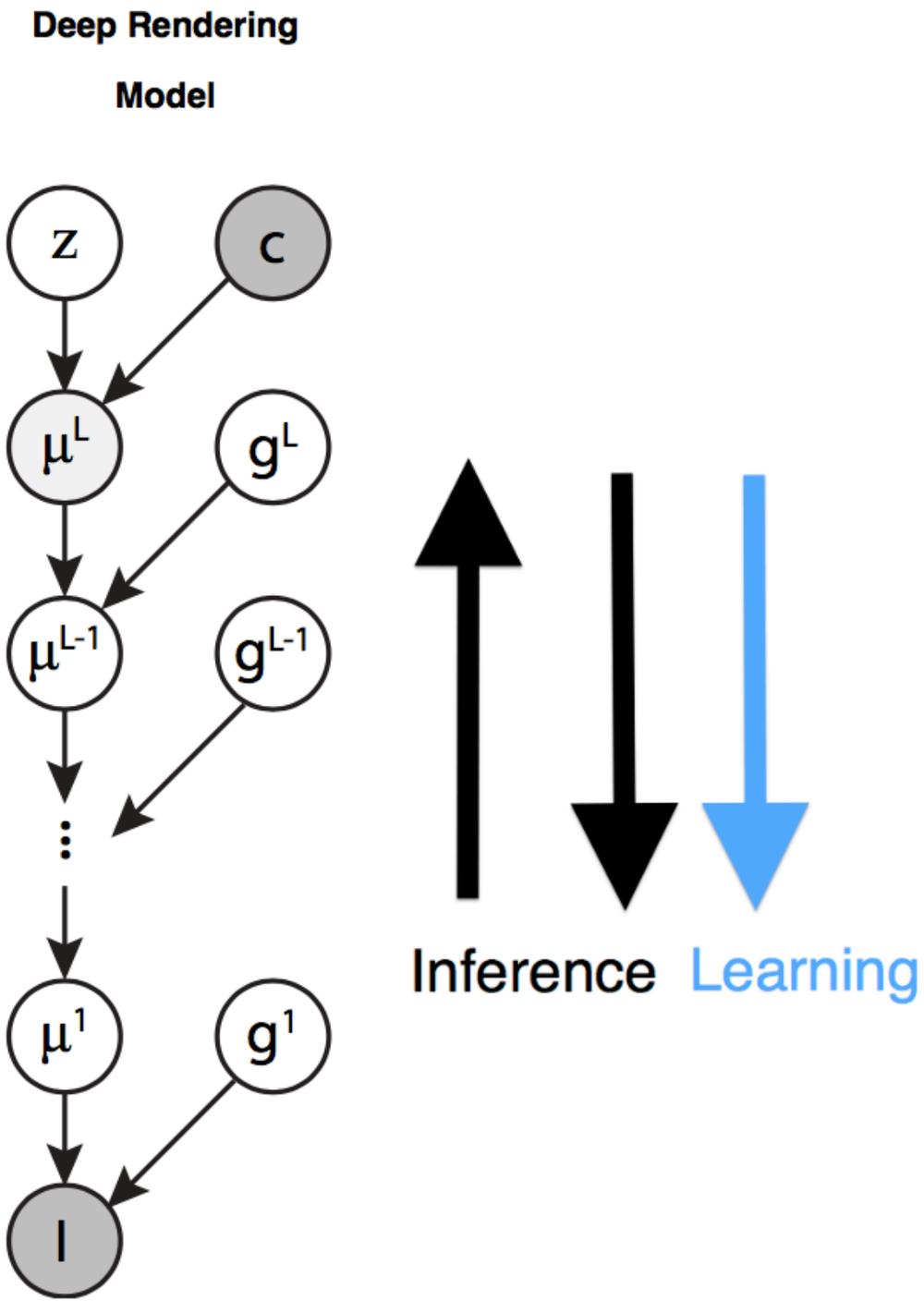
# Deep Generative Model

Stacked semi-supervised learner




Each edge is parameterised as a deep neural net

# Deep Rendering Model




**E-step:**

$$g_n^*, a_n^* = \arg\max_{g,a} \gamma_{nga}$$

$$\mathbb{E}\left[\tilde{a}_n \odot \tilde{z}_n | g_n^*, I_n; \theta\right] = \tilde{\Lambda}_{g_n^*}^{\dagger}\left[\tilde{a}_n^*\right] I_n$$

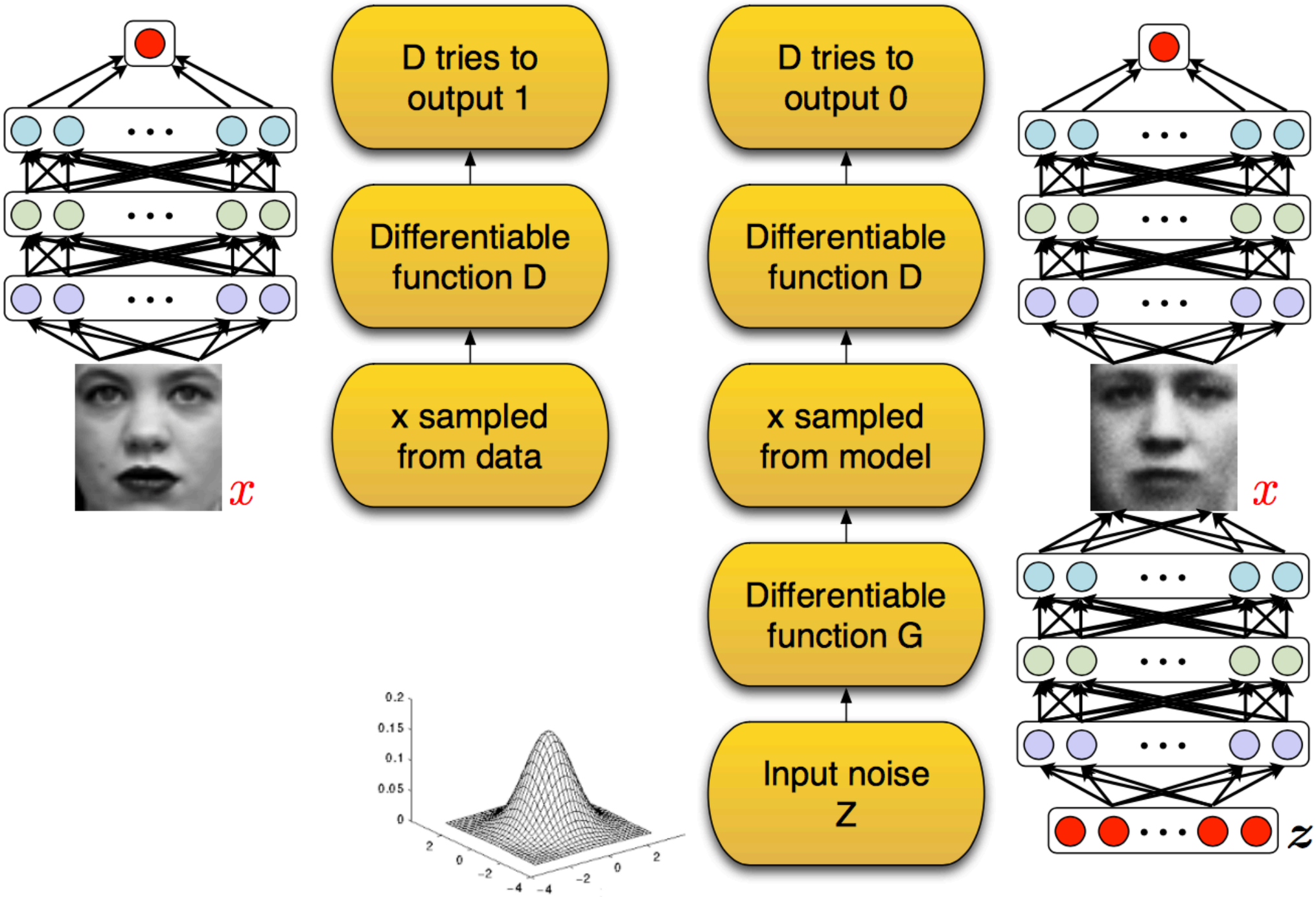
**M-step:**

$$\tilde{\Lambda}_g = \mathbf{OLS}\left(I_n \sim \mathbb{E}\left[\tilde{a}_n \odot \tilde{z}_n | g_n, I_n; \theta\right], n \in (g,a)\right)$$

**G-step:**

$$\tilde{\Lambda}_g = \eta_{\mathrm{LR}} \cdot \nabla_{\tilde{\Lambda}_g} \ell_{DRM}(\theta)$$

# Generative Adversarial Networks (GAN)



# Zero-Sum Game Objective

- Minimax objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- In practice, to estimate G we use:

$$\max_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))]$$

Why? Stronger gradient for G when D is very good.

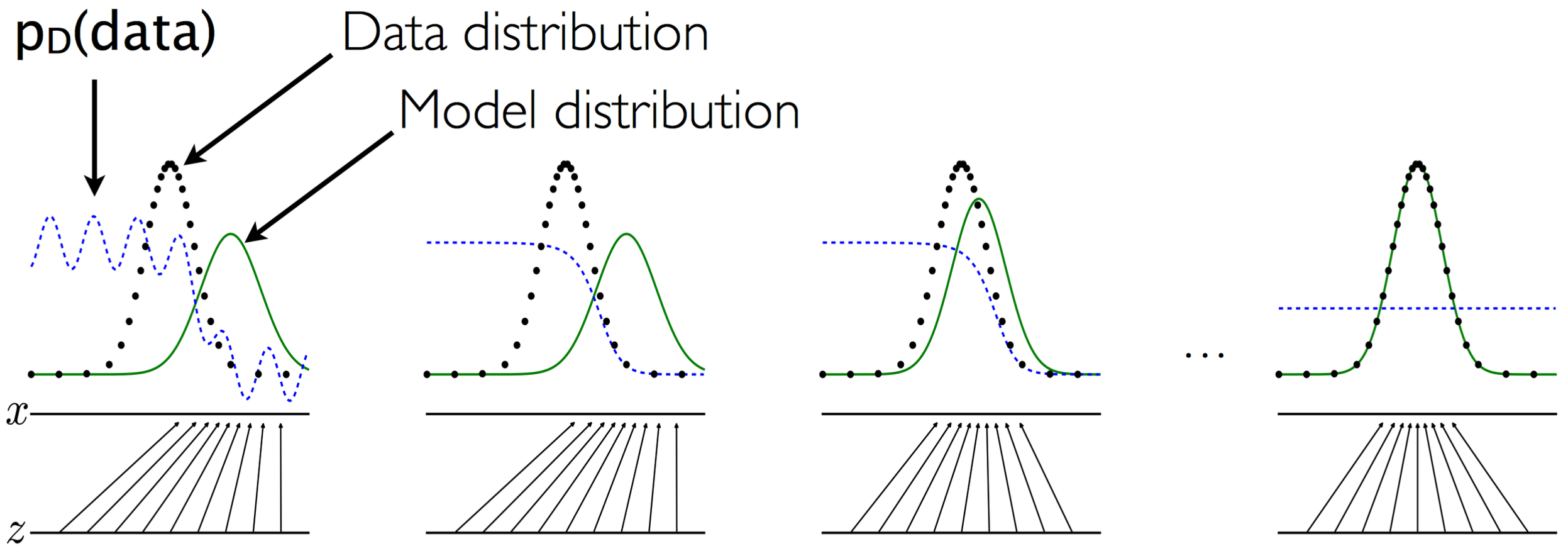
# Zero-Sum Game Objective

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- Theoretical properties (assuming infinite data, infinite model capacity, direct updating of generator's distribution):

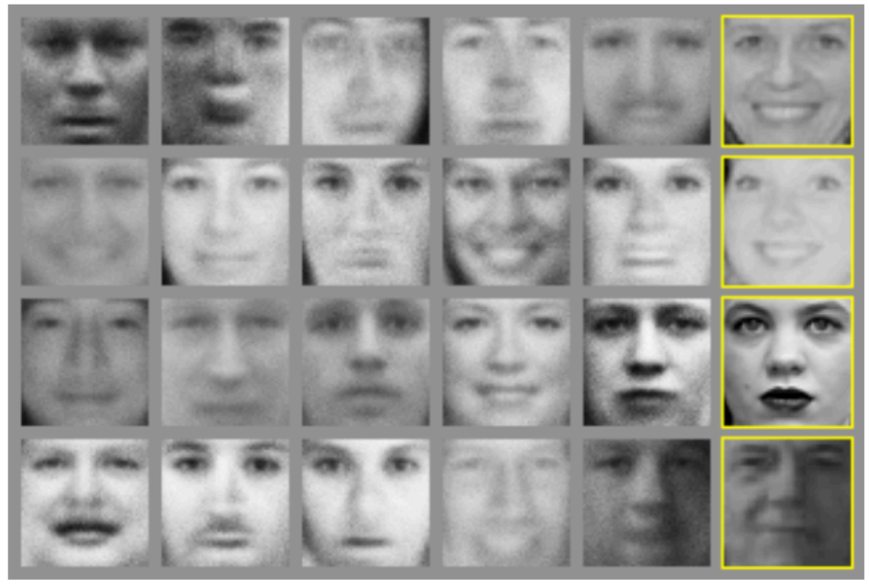
- Unique global optimum.
- Optimum corresponds to data distribution.
- Convergence to optimum guaranteed.

# Learning Process in GAN



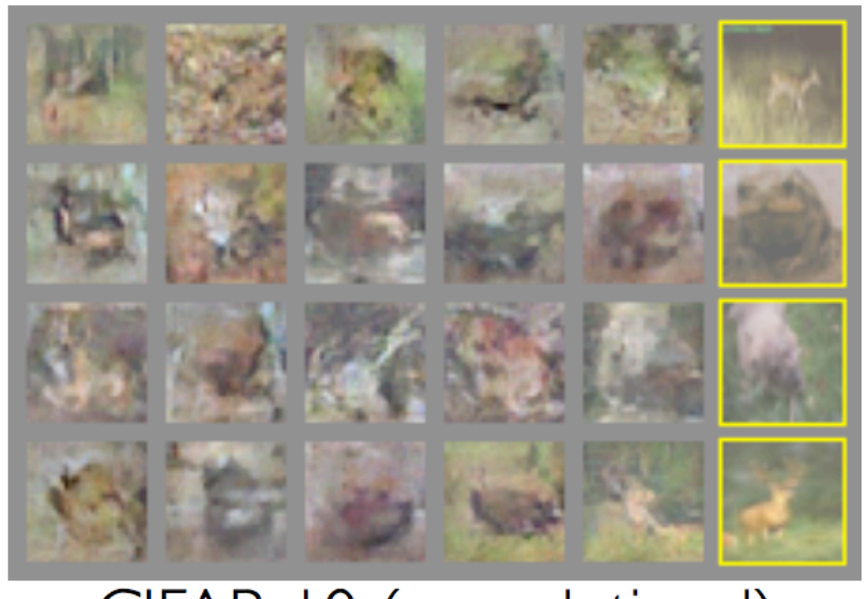
# Visualization of Model Samples

MNIST



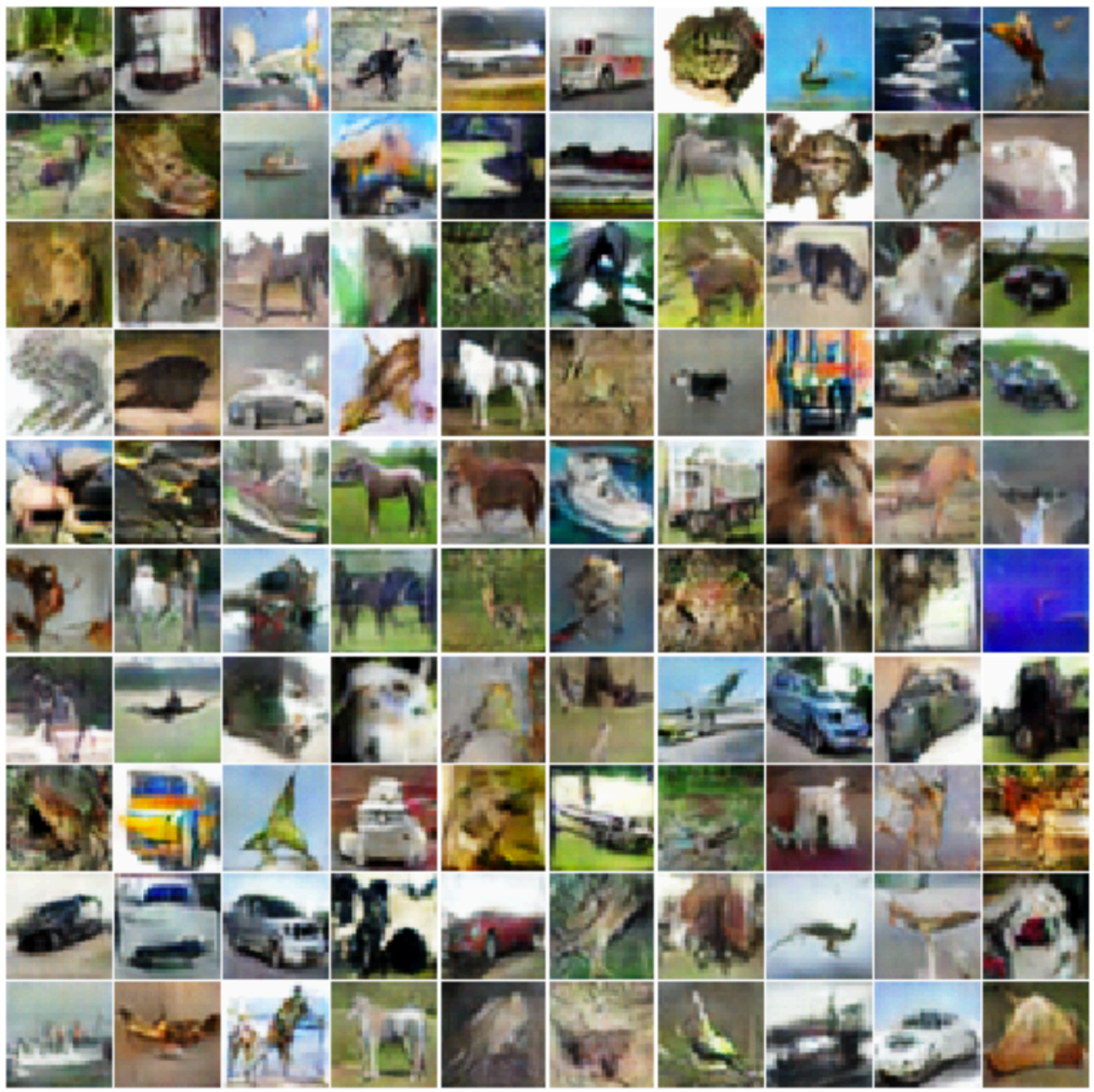
TFD

CIFAR-10 (fully connected)



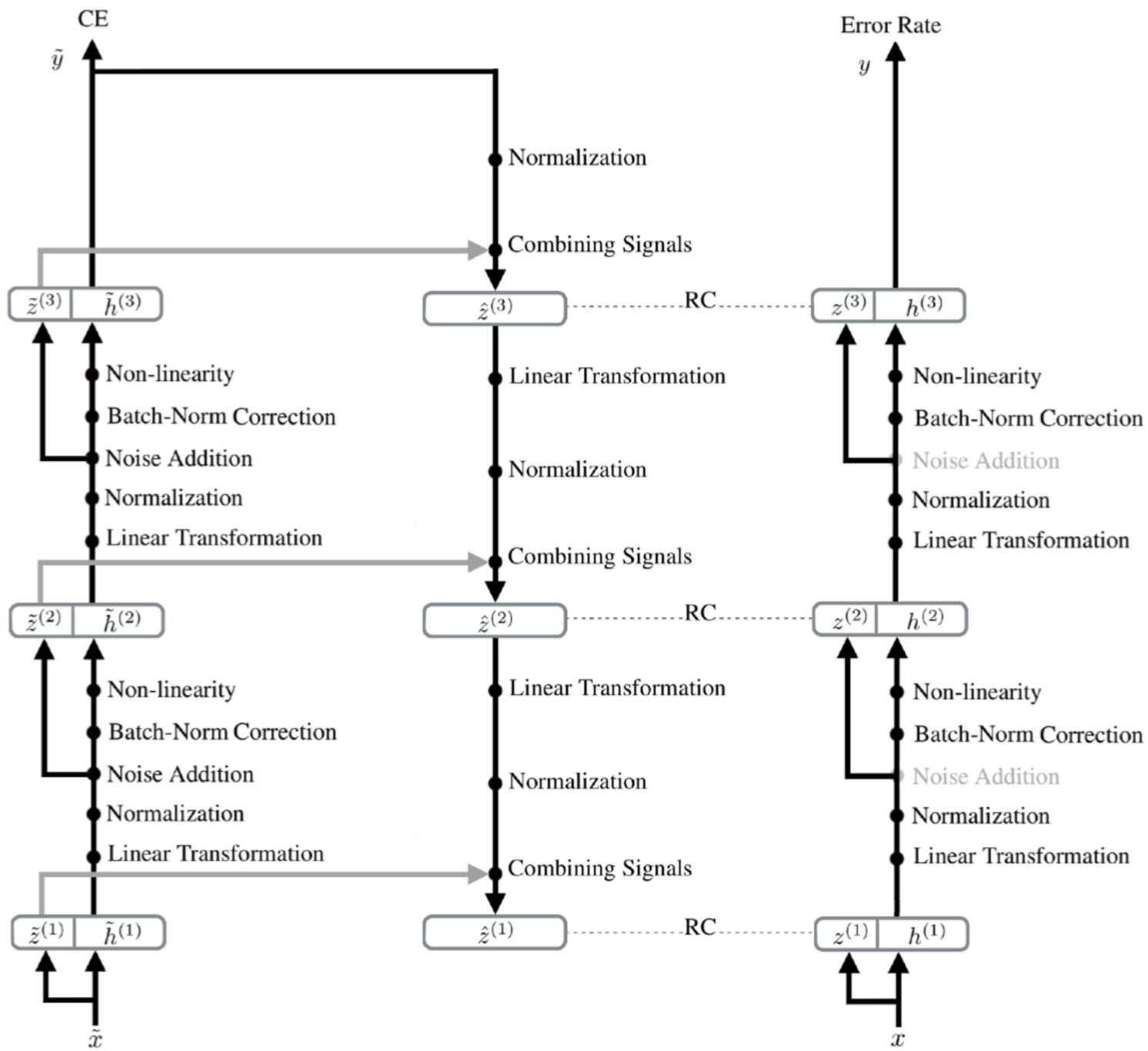
CIFAR-10 (convolutional)

# Improved GAN



# Improved GAN

# Ladder Network



# Ladder Network

**Algorithm 1** Calculation of the output $\mathbf{y}$ and cost function $C$ of the Ladder network

**Require:** $\mathbf{x}(n)$

# Corrupted encoder and classifier

$\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$

**for** l = 1 **to** L **do**

$\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{W}^{(l)}\tilde{\mathbf{h}}^{(l-1)}) + \texttt{noise}$

$\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$

**end for**

$P(\tilde{\mathbf{y}} \mid \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$

# Clean encoder (for denoising targets)

$\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$

**for** l = 1 **to** L **do**

$\mathbf{z}_{\text{pre}}^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}$

$\boldsymbol{\mu}^{(l)} \leftarrow \texttt{batchmean}(\mathbf{z}_{\text{pre}}^{(l)})$

$\boldsymbol{\sigma}^{(l)} \leftarrow \texttt{batchstd}(\mathbf{z}_{\text{pre}}^{(l)})$

$\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{z}_{\text{pre}}^{(l)})$

$\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$

**end for**

# Final classification:

$P(\mathbf{y} \mid \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$

# Decoder and denoising

**for** l = L **to** 0 **do**

**if** l = L **then**

$\mathbf{u}^{(L)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{h}}^{(L)})$

**else**

$\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{V}^{(l+1)}\hat{\mathbf{z}}^{(l+1)})$

**end if**

$\forall i : \hat{z}_i^{(l)} \leftarrow g(\tilde{z}_i^{(l)}, u_i^{(l)})$ # Eq. (2)

$\forall i : \hat{z}_{i,\text{BN}}^{(l)} \leftarrow \frac{\hat{z}_i^{(l)} - \mu_i^{(l)}}{\sigma_i^{(l)}}$

**end for**

# Cost function $C$ for training:

$\text{C} \leftarrow 0$

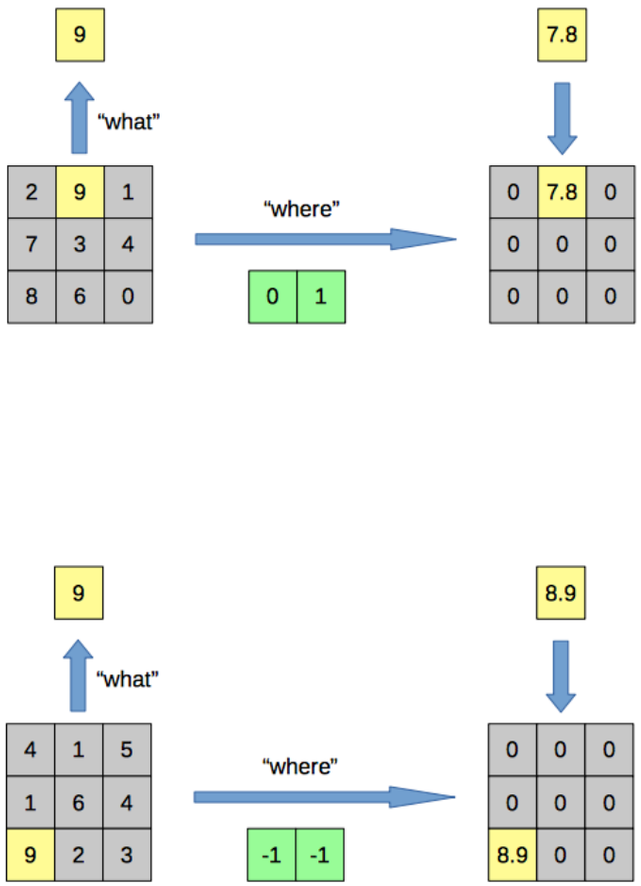
**if** $t(n)$ **then**

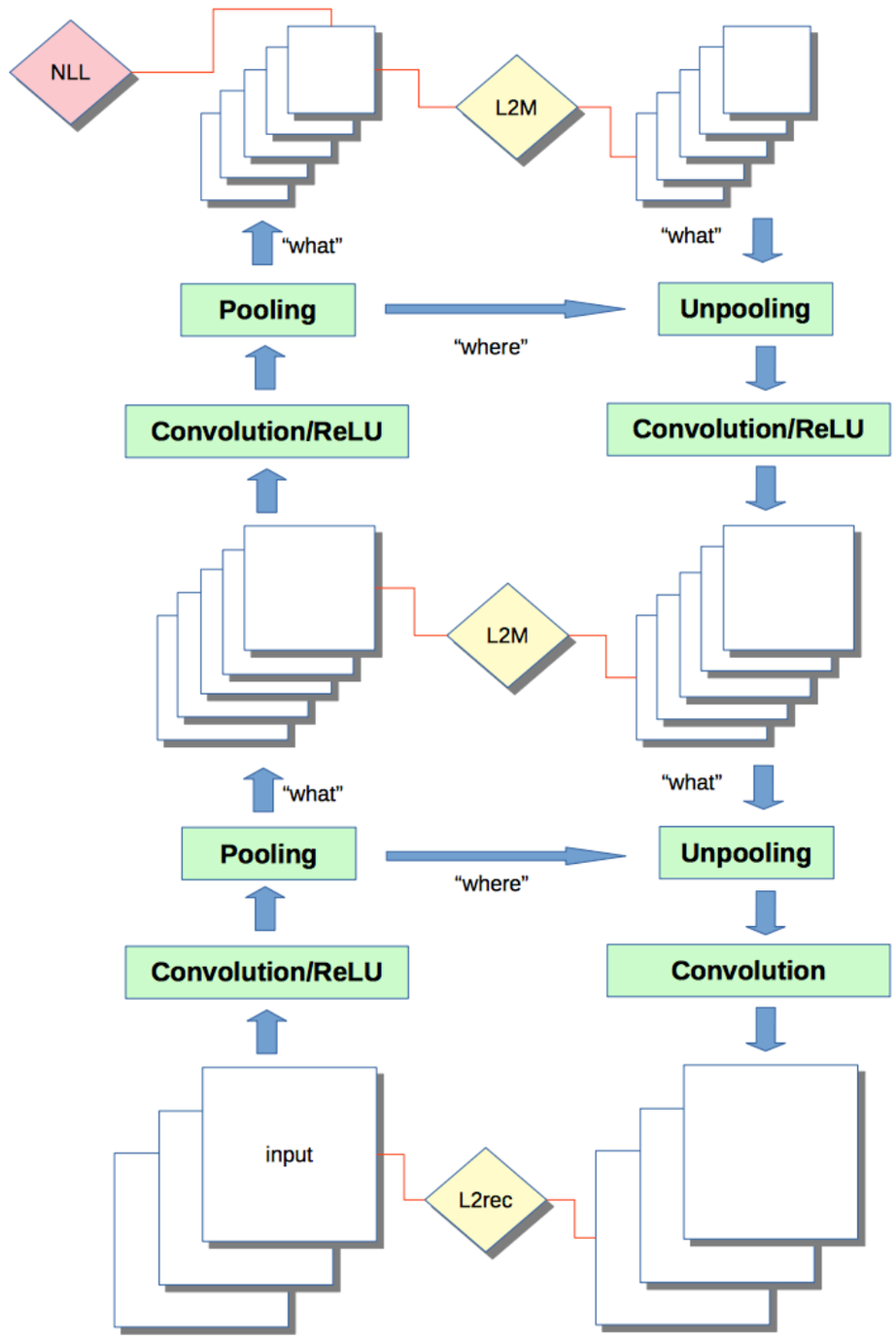
$\text{C} \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mid \mathbf{x}(n))$

**end if**

$\text{C} \leftarrow \text{C} + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}_{\text{BN}}^{(l)} \right\|^2$ # Eq. (3)

# Stacked What-Where Autoencoder



# Semi-supervised Results

| Model | Number of incorrectly predicted test examples for a given number of labeled samples | | | |
|---|---|---|---|---|
| | 20 | 50 | 100 | 200 |
| DGN [21] | | | $333 \pm 14$ | |
| Virtual Adversarial [22] | | | 212 | |
| CatGAN [14] | | | $191 \pm 10$ | |
| Skip Deep Generative Model [23] | | | $132 \pm 7$ | |
| Ladder network [24] | | | $106 \pm 37$ | |
| Auxiliary Deep Generative Model [23] | | | $96 \pm 2$ | |
| Our model | $1677 \pm 452$ | $221 \pm 136$ | $93 \pm 6.5$ | $90 \pm 4.2$ |
| Ensemble of 10 of our models | $1134 \pm 445$ | $142 \pm 96$ | $86 \pm 5.6$ | $81 \pm 4.3$ |

Table 1: Number of incorrectly classified test examples for the semi-supervised setting on permutation invariant MNIST. Results are averaged over 10 seeds.

| Model | Test error rate for a given number of labeled samples | | | |
|---|---|---|---|---|
| | 1000 | 2000 | 4000 | 8000 |
| Ladder network [24] | | | $20.40 \pm 0.47$ | |
| CatGAN [14] | | | $19.58 \pm 0.46$ | |
| Our model | $21.83 \pm 2.01$ | $19.61 \pm 2.09$ | $18.63 \pm 2.32$ | $17.72 \pm 1.82$ |
| Ensemble of 10 of our models | $19.22 \pm 0.54$ | $17.25 \pm 0.66$ | $15.59 \pm 0.47$ | $14.87 \pm 0.89$ |

Table 2: Test error on semi-supervised CIFAR-10. Results are averaged over 10 splits of data.