Learning Similarity Metrics

Yann LeCun

Computational and Biological Learning Lab

The Courant Institute of Mathematical Sciences

New York University

http://yann.lecun.com

http://www.cs.nyu.edu/~yann

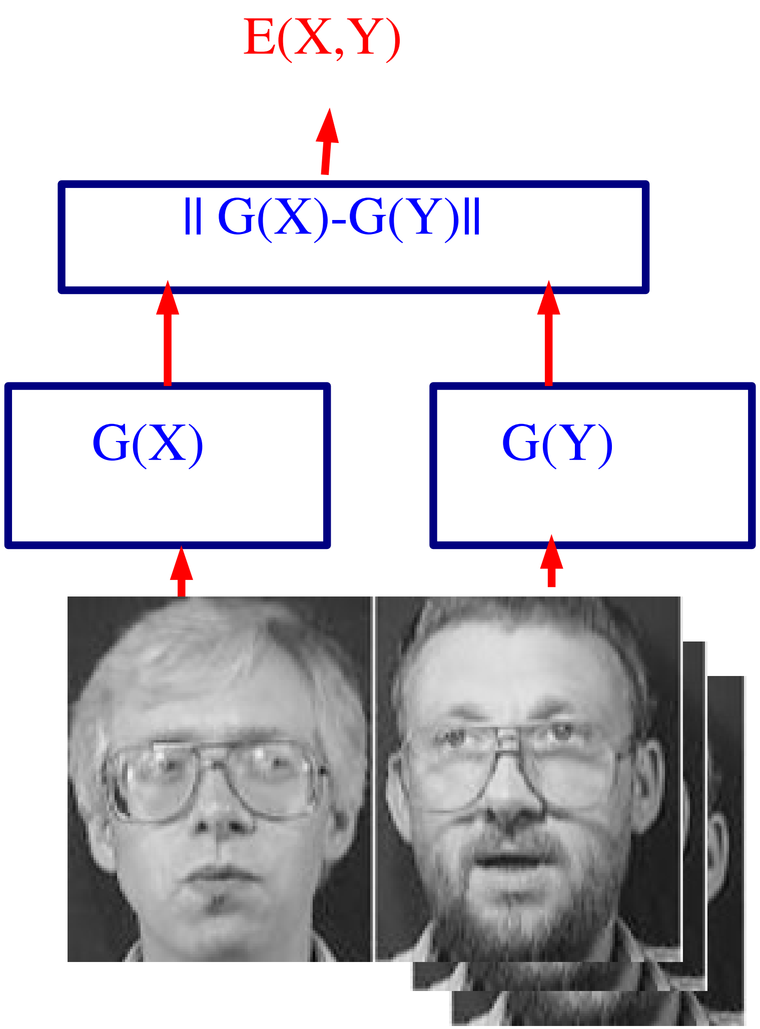
Learning a Similarity Measure

- **Many methods for classification, clustering, and dimensionality reduction rely on a similarity measure.**
- **Question: how do we learn a mapping G(X) such that the Euclidean distance in the transformed space ||G(X1)-G(X2)|| corresponds to the "semantic distance" between X1 and X2 in the input space?**
- **The idea goes back to Fisher's Linear Discriminant Analysis (LDA):**
  - find a projection such that, in the projected space, the ratio of inter-class variance to intra-class variance is maximized.
  - while the idea is appealing, its performance for classification is abysmal (better off using logistic regression, which is much simpler, faster, and better).
- **There has been a regain of interest in new kinds of metric learning over the last few years which use local discrimination criteria and non-linear mappings**

Metric Learning is not Embedding

- **There are lots of methods to embed points into a low dimensional space: Multi-Dimensional Scaling, Isomap, LLE, Laplacian Eigenmaps,.....**
- **These methods do not produce a full mapping from the input space to the low dimensional space.**
  - They merely map the training samples
  - They cannot be applied to new samples without some additional hack.
- **In Metric Learning, we want to learn a mapping G(X) that can be applied to any new X (not just the training samples).**

Example: Face Recognition with Nearest Neighbor Classification

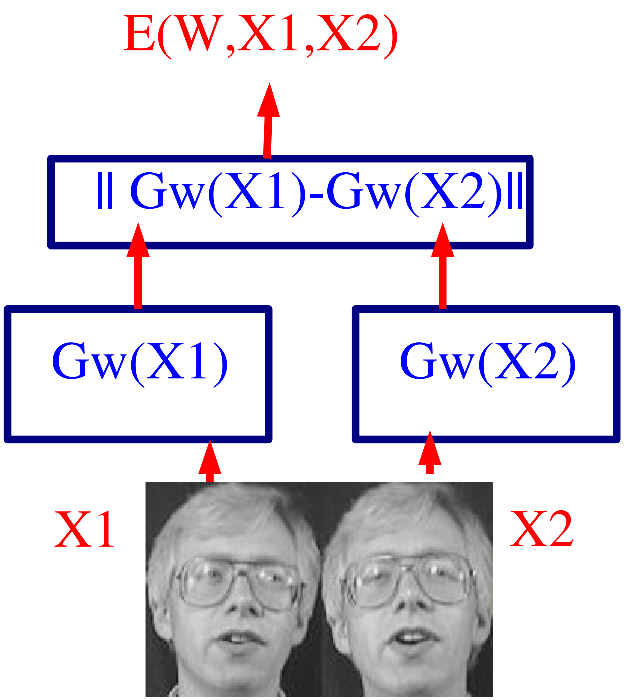


- **X and Y are images**
- **Y is a discrete variable with many possible values**
  - All the people in our gallery
- **Example of architecture:**
  - A function G(X) maps input images into a low-dimensional space in which the Euclidean distance measures dissemblance.
- **Inference:**
  - Find the Y in the gallery that minimizes E(X,Y) (find the Y that is most similar to X)
  - Minimization through exhaustive search.

Basic Idea of Metric Learning

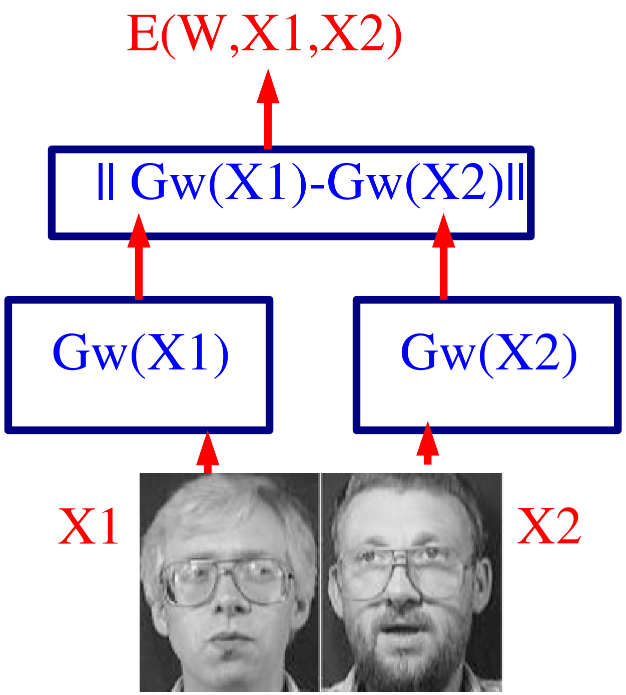
- **Pick a family of transformation {Gw(X), w in $\mathcal{W}$}**
- **Use a “Siamese Architecture”, and learn a parameter W that will:**

Make this small



Semantically similar samples
(e.g. same label)

Make this large



Semantically different samples
(e.g. different labels)

Methods

- **Specific methods differ in how they pick:**
  - the loss function
  - the architecture E(W,X1,X2) (linear or non-linear)
  - the optimization algorithm (gradient descent, SDP, ...)
  - how they approximate the loss function and its gradient: the loss has sums with a quadratic number of terms in the number of training samples.
- **Cosine-based Siamese networks (non-linear G(X))**
- **Neighborhood Component Analysis (linear and non-linear versions)**
- **Contrastive Loss Function Methods (margin-like loss)**
- **Invariant Manifold Learning (DrLIM)**
- **Non-linear NCA with unsupervised pre-training**

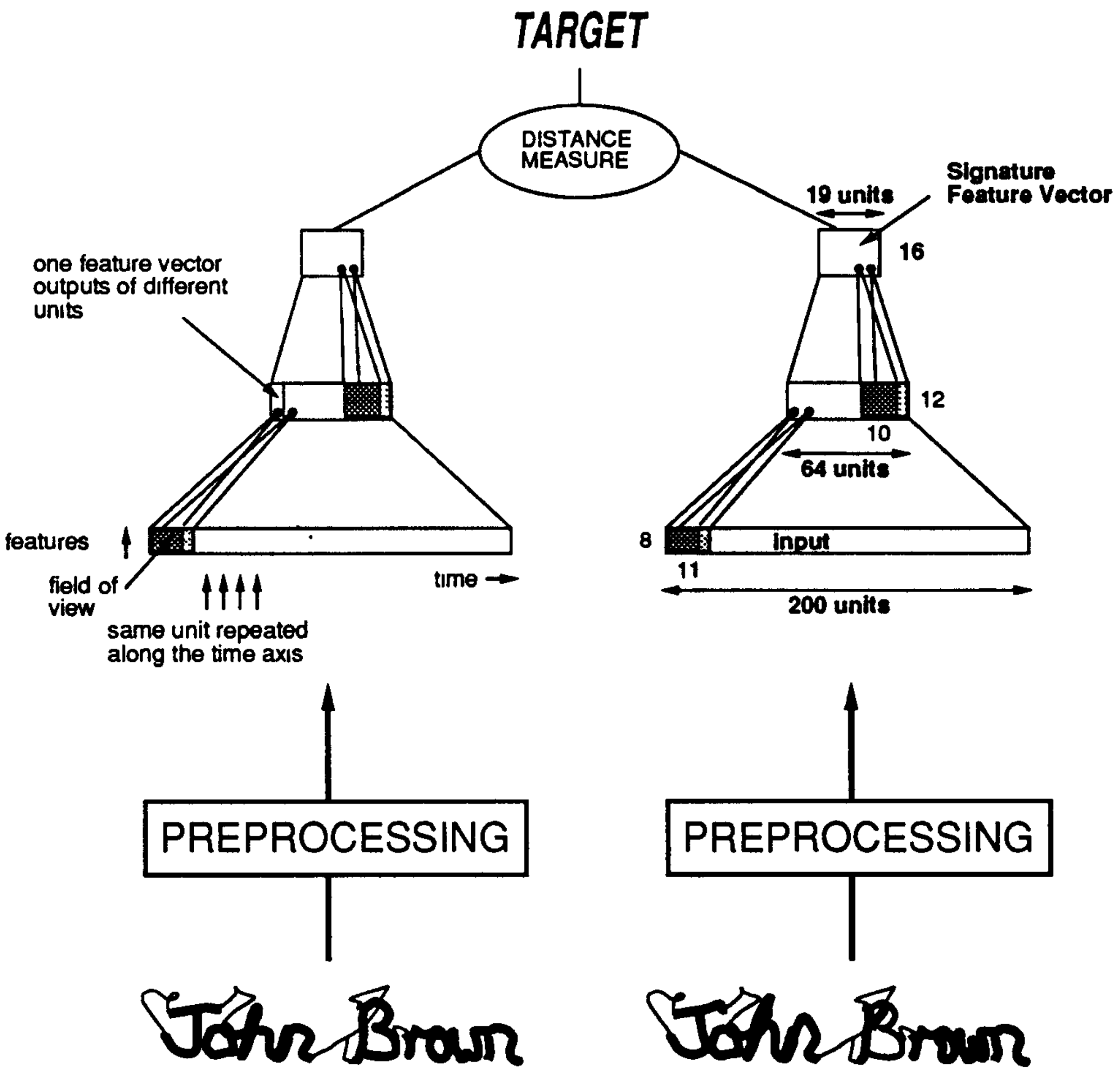
Trainable Metric vs Other Dimensionality Reduction Methods

- PCA-based dimensionality reduction methods
  - Linear projection trained non-discriminatively to maximize variance.
  - Disadvantages: linear; no discrimination.
- LDA-based dimensionality reduction methods
  - Linear projection trained discriminatively to maximize inter-class variance and minimize intra-class variance.
  - Disadvantage: linear
- Kernel – PCA and Kernel – LDA
  - Non-linear extensions of the above.
  - Disadvantage: no invariance unless it's built into the kernel.
- LLE and MDS
  - Maps each training sample into low-dim Euclidean space that preserve distances or angles.
  - Disadvantages: no direct mapping, no parameterized invariance, no simple way to use the "semantic" distance between training samples.
- Advantages of trainable metrics:
  - The non-linear parameterization of the mapping allows to learn dissimilarity metrics that are **invariant to irrelevant transformations** of the inputs.

Trainable Metrics vs hand-crafted invariances

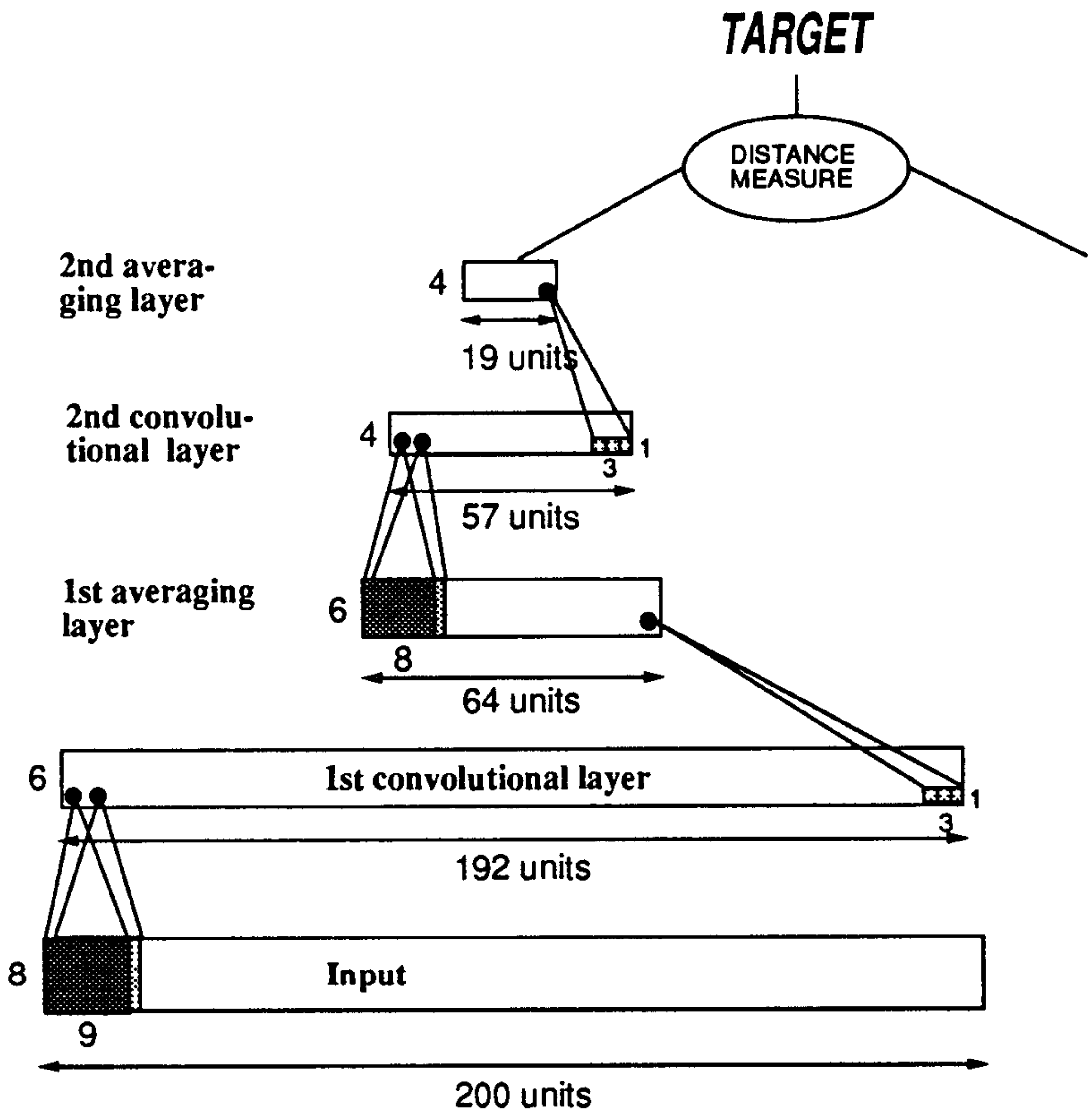
- Dissimilarity metrics with hand-crafted invariances
  - Tangent distance methods.
  - Elastic matching.
  - Warping-based normalization algorithms.
- Disadvantages
  - Cannot learn invariance to transformations that are hidden in the data (e.g. Glasses or no glasses for face recognition).

Siamese Architecture for Comparing Time-Series Data



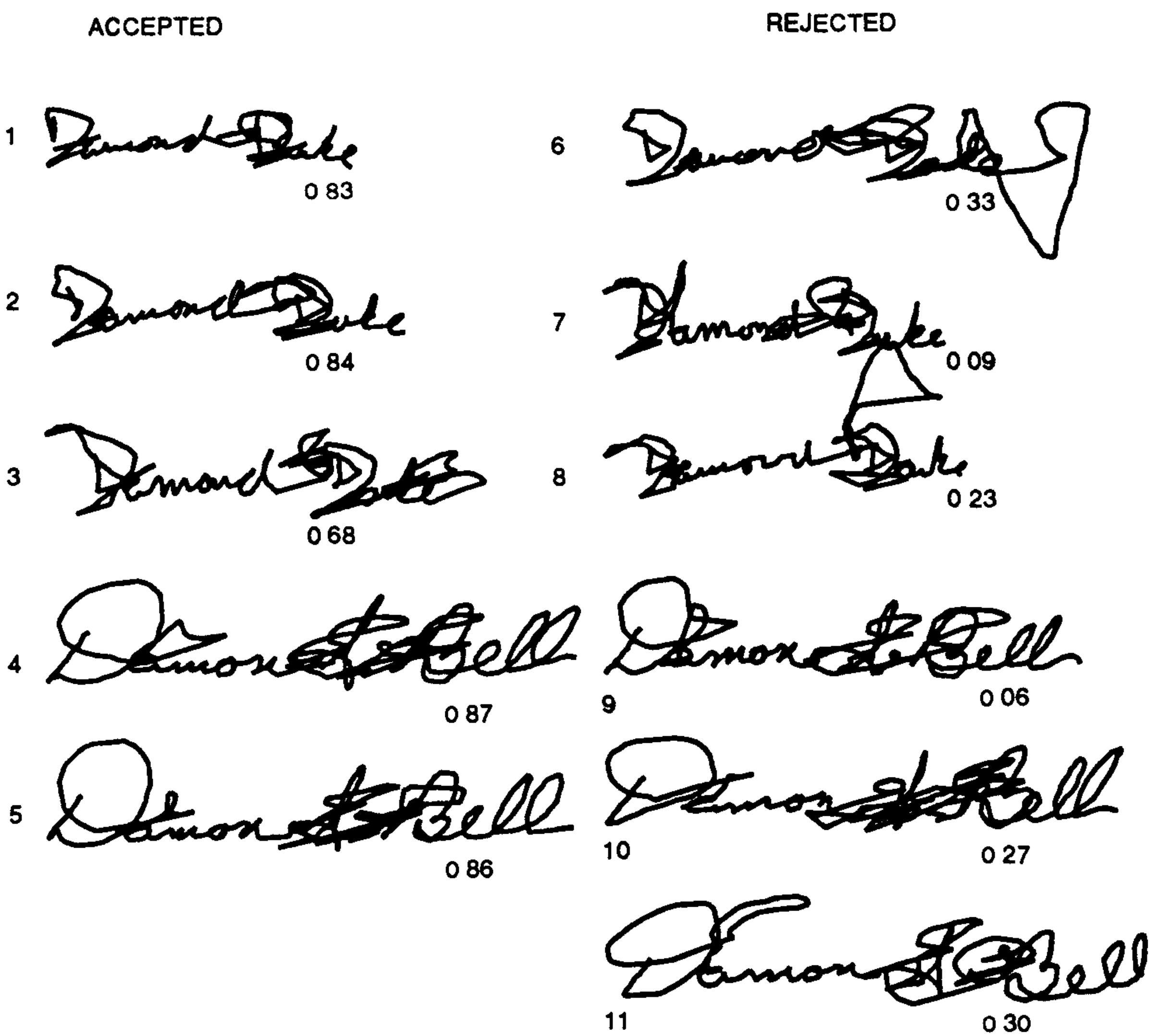
- **Signature Verification** (Bromley, Guyon, LeCun, Sackinger, Shah NIPS 1994)
- The signatures are represented by the XY trajectory of the pen

1D Convolutional Net (TDNN)



Examples

- Loss function:
  - maximize cosine of output vectors for genuine pair
  - make it close to zero (or -1) for forged pair



- 80% of forgeries detected for 97% genuine signatures accepted
- The "code" for a signature only has 80 dimensions.

Neighborhood Component Analysis (NCA)

[Golberger, Roweis, Hinton, Salakhutdinov, NIPS 2004]

- **Linear version:** $d(x,y) = (x-y)^\top Q(x-y) = (Ax-Ay)^\top(Ax-Ay).$
- **Probability that Xi picks Xj as neighbor:**

$$p_{ij} = \frac{\exp(-\|Ax_i - Ax_j\|^2)}{\sum_{k\neq i}\exp(-\|Ax_i - Ax_k\|^2)} \quad , \qquad p_{ii} = 0$$

- **Loss function:**

$$f(A) = \sum_i \sum_{j\in C_i} p_{ij} = \sum_i p_i$$

- **Gradient:**

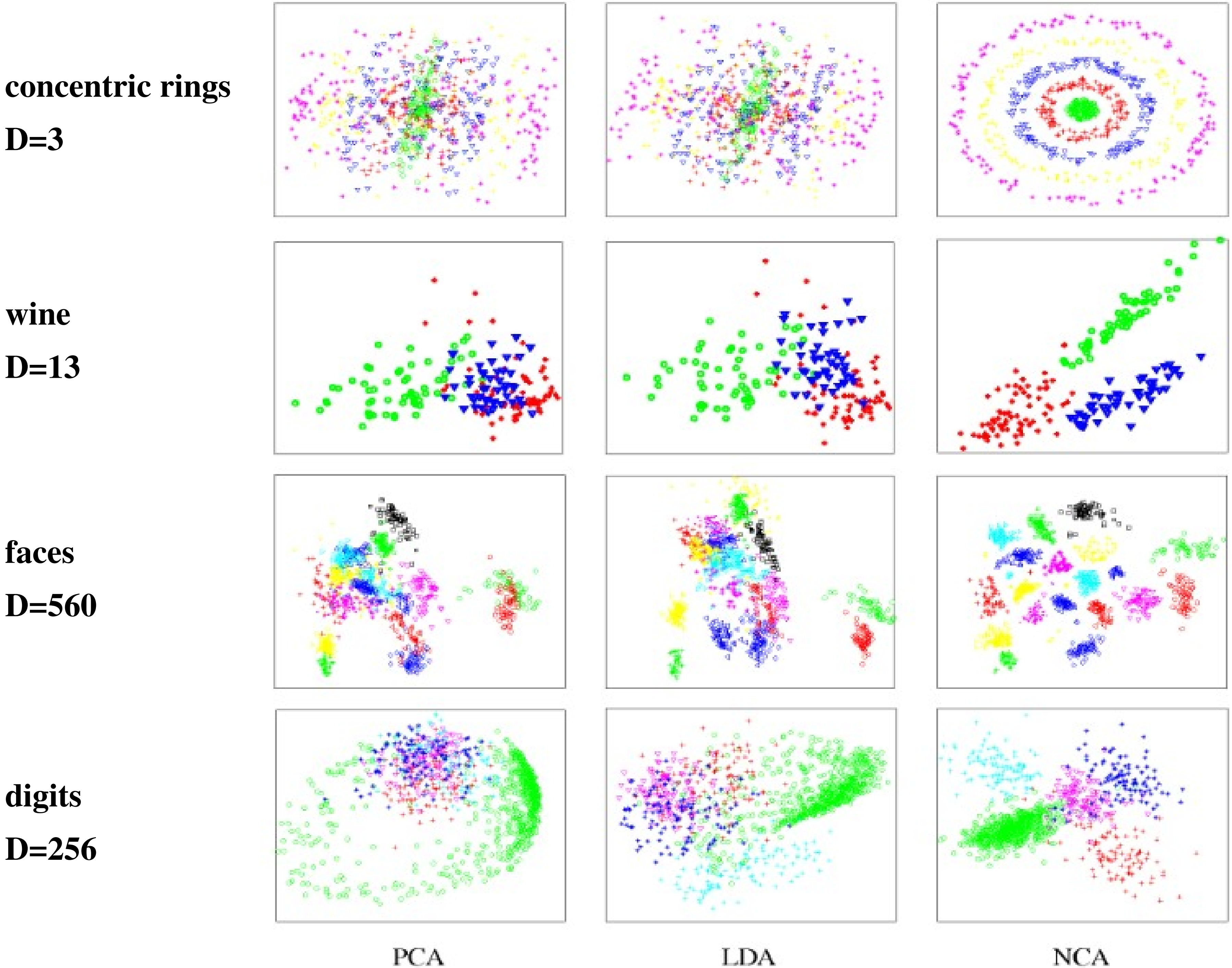
$$\frac{\partial f}{\partial A} = 2A\sum_i\left(p_i\sum_k p_{ik}x_{ik}x_{ik}^\top - \sum_{j\in C_i} p_{ij}x_{ij}x_{ij}^\top\right)$$

Neighborhood Component Analysis (NCA)

[Golberger, Roweis, Hinton, Salakhutdinov, NIPS 2004]

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_k p_{ik} x_{ik} x_{ik}^\top - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^\top \right)$$

- **Problem: the first term has a lot of terms in it (as many as there are training samples) ==> quadratic**
- **Solution: thresholding and random sampling**
  - the Pik values fall off very quickly. Most of them can dafely be ignored
  - it suffices to take a random subset of the samples.

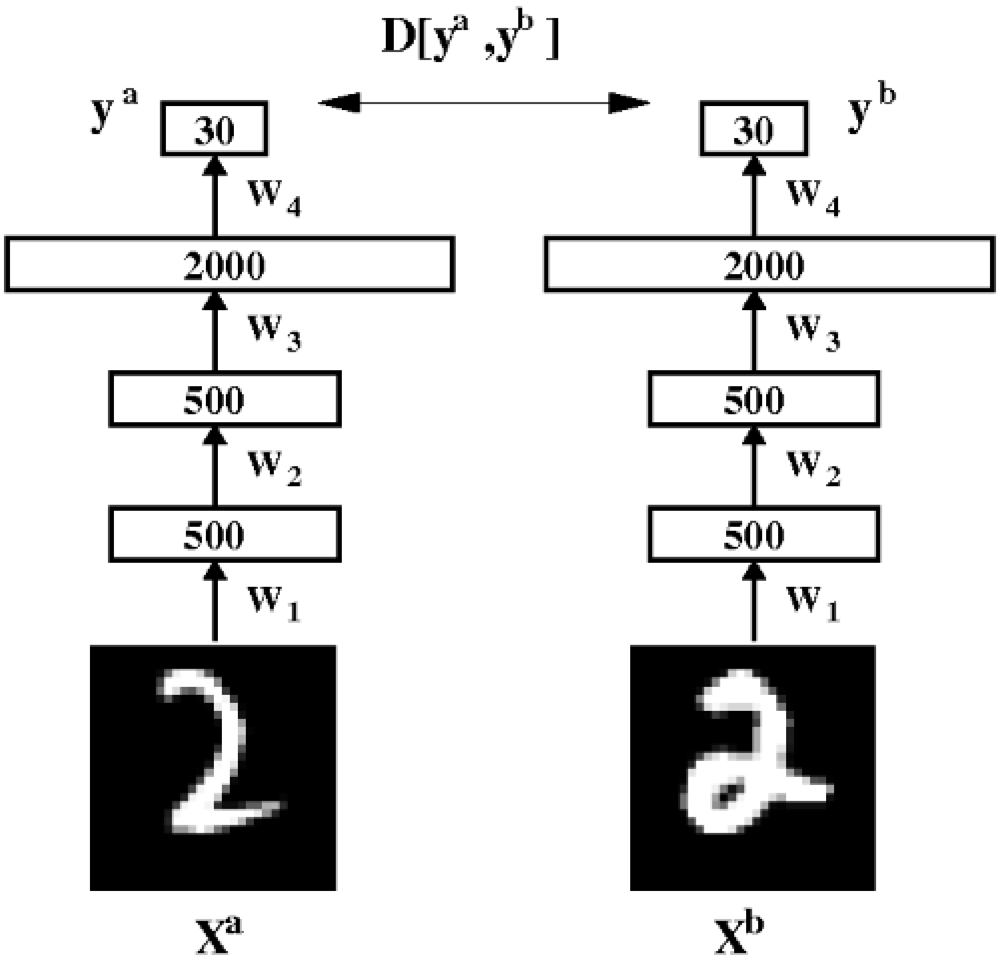


from [Golberger, Roweis, Hinton, Salakhutdinov, NIPS 2004]

Non-Linear NCA with Unsupervised Pre-Training

[Salakhutdinov and Hinton "Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure" AISTATS*07]

- **Basic Idea:** **use NCA with a very "deep" neural net, capable of producing highly non-linear mappings.**
- **Problem:** **these networks are difficult to train with gradient descent**
- **Solution:**
  - 1. pre-train the network layer by layer using an unsupervised method
  - 2. refine the pre-trained network with non-linear NCA



Non-Linear NCA with Unsupervised Pre-Training

[Salakhutdinov and Hinton "Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure" AISTATS*07]

- **Same method as regular NCA:**

We are given a set of N labeled training cases $(\mathbf{x}^a, c^a)$, $a = 1, 2, ..., N$, where $\mathbf{x}^a \in R^d$, and $c^a \in \{1, 2, ..., C\}$. For each training vector $\mathbf{x}^a$, define the probability that point a selects one of its neighbours b (as in [9, 13]) in the transformed feature space as:

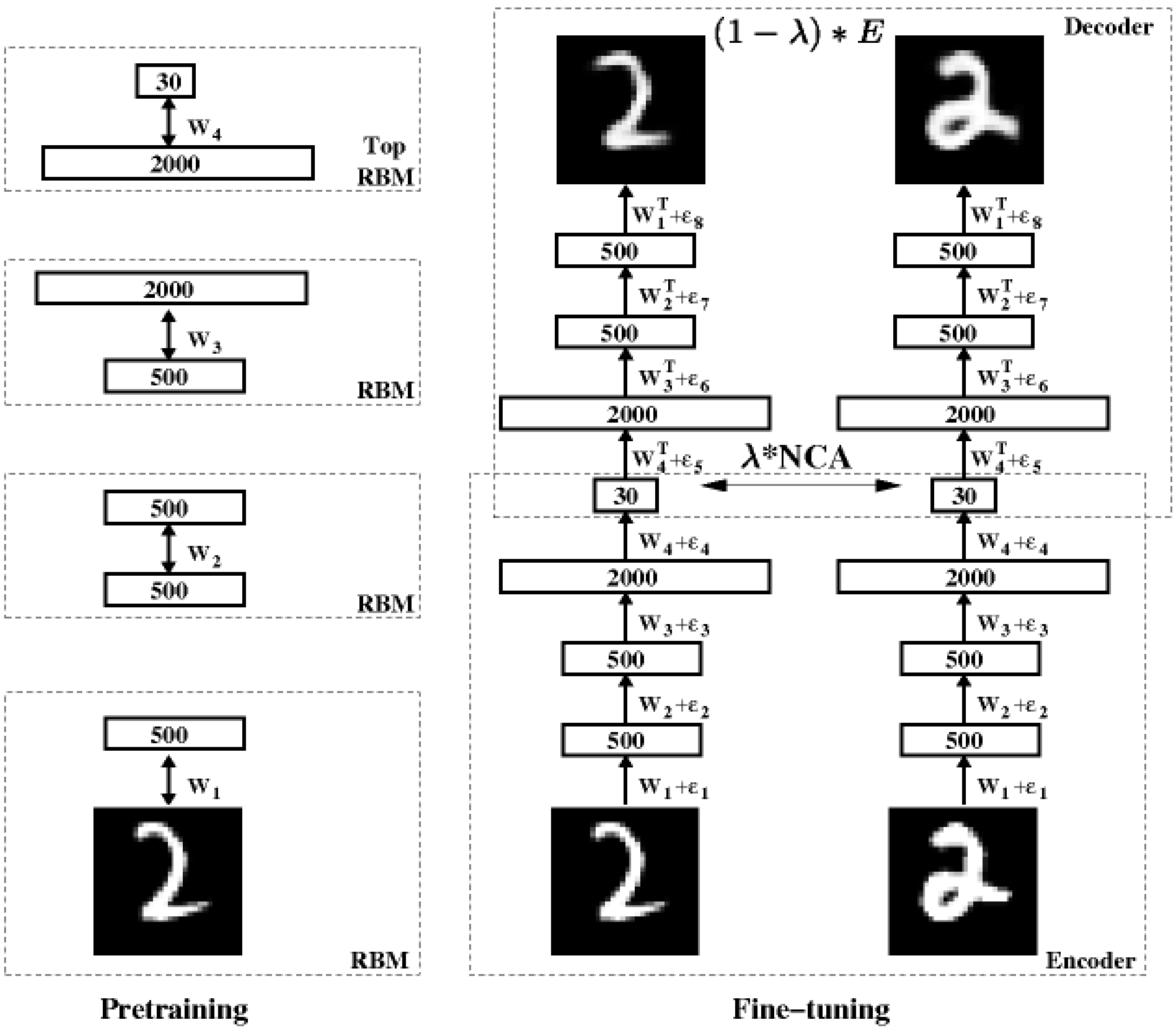
$$p_{ab} = \frac{\exp(-d_{ab})}{\sum_{z \neq a} \exp(-d_{az})}, \qquad p_{aa} = 0 \qquad (3)$$

We focus on the Euclidean distance metric:

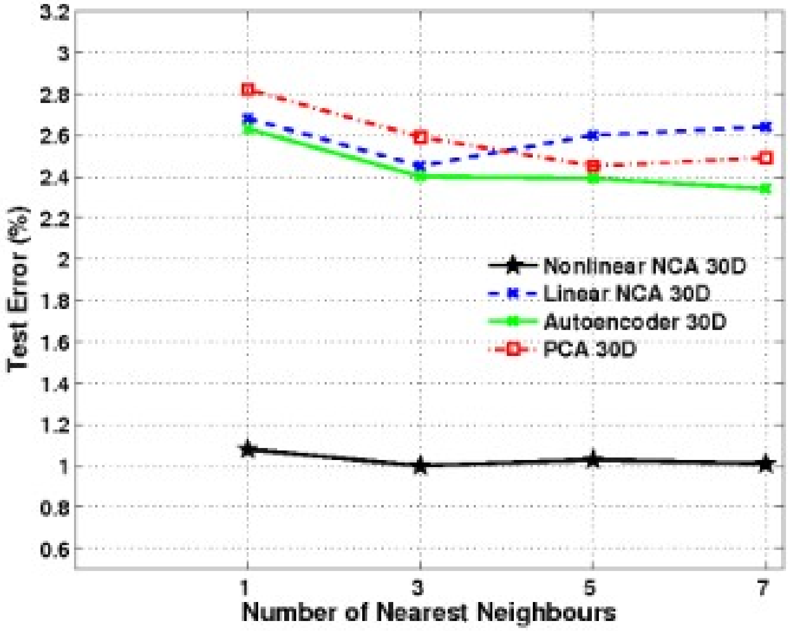
$$d_{ab} = \| f(\mathbf{x}^a | W) - f(\mathbf{x}^b | W) \|^2$$

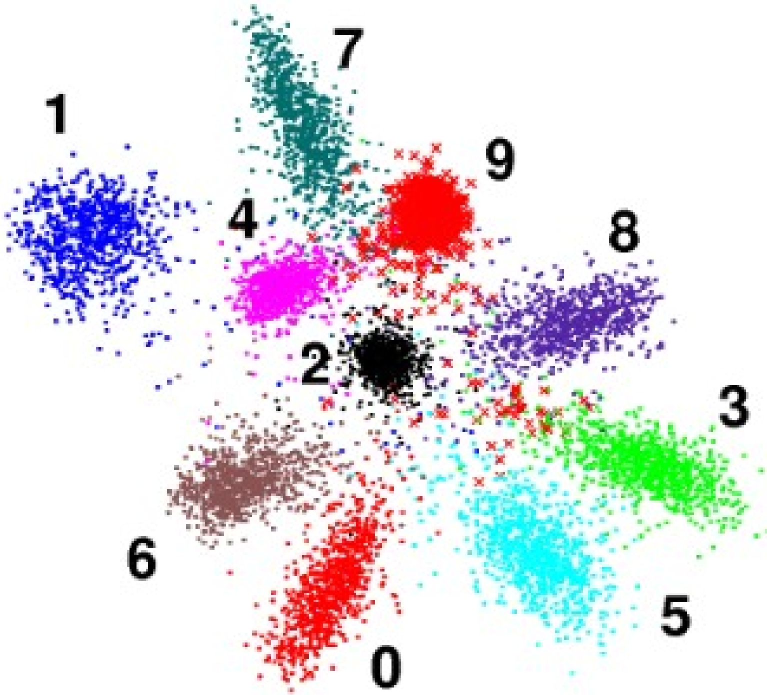
Non-Linear NCA with Unsupervised Pre-Training

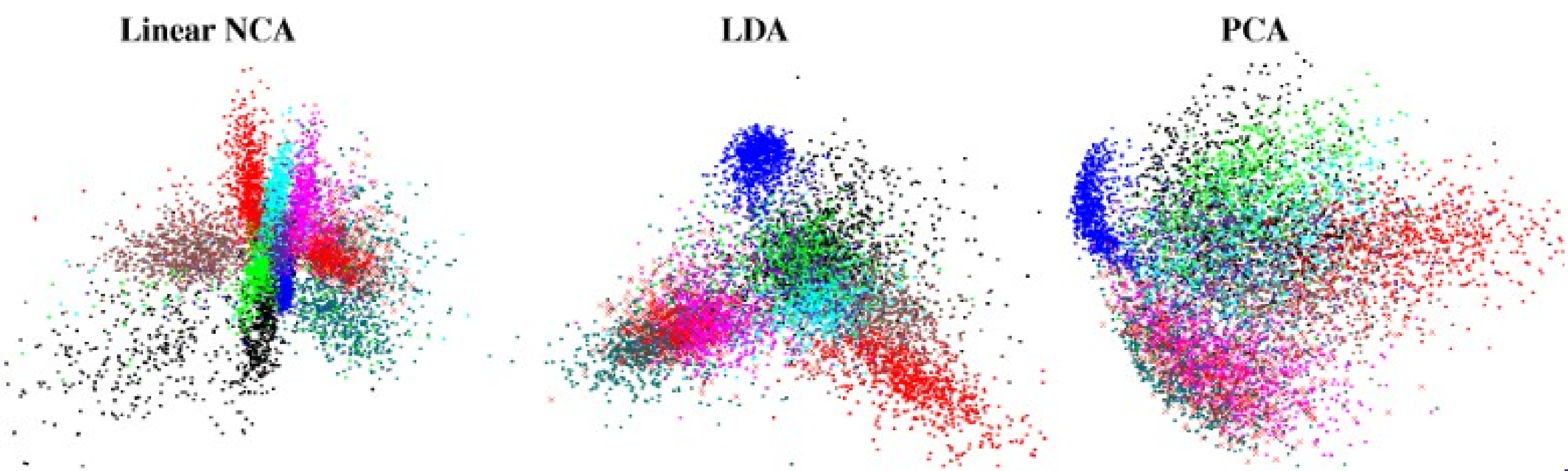
- Each layer is trained with the Restricted Boltzmann Machine algorithm
- The fine-tuning minimizes a linear combination of the NCA loss and the reconstruction error



Non-Linear NCA on MNIST digits







Another Loss Function

- **Idea: don't push away all the points, simply push away the most offending alien points [MOAP] (the point with a different label than Xi that is closest to it)**
  - This will eventually cause a point with the same label to be closest to Xi

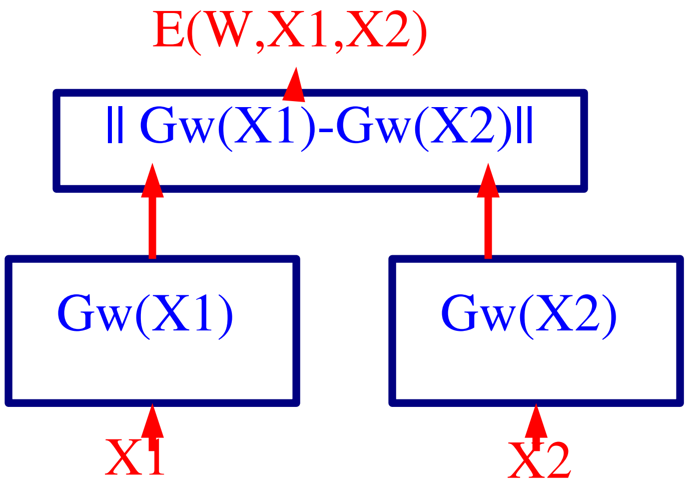
MOAP: $\bar{Y}^i = \text{argmin}_{y \neq Y^i} E(W, y, X^i)$

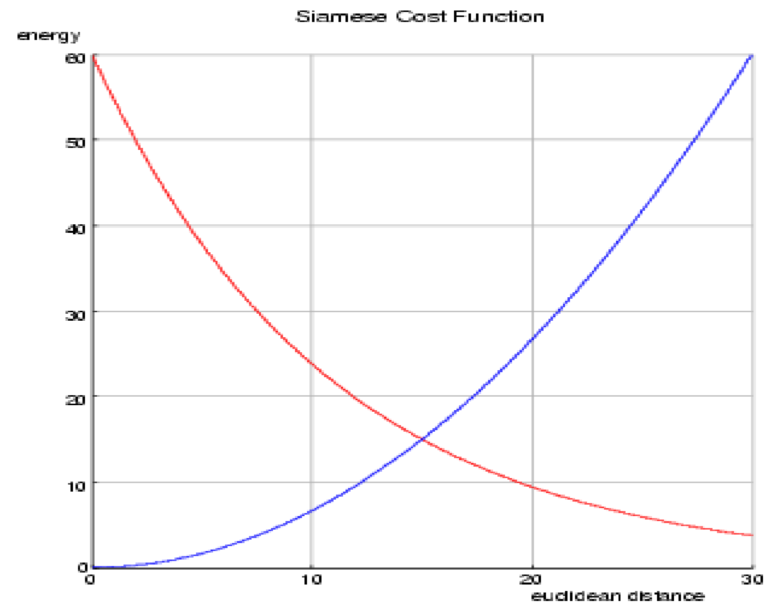
$$\mathcal{L}(W, Y^1, Y^2, \ldots, X^1, X^2, \ldots) = \sum_i L^+\left(E(W, Y^i, X^i)\right) + L^-\left(\min_{Y \neq Y^i} E(W, Y, X^i)\right)$$

Increasing function: Pushes down on the energy of the correct answers

Decreasing function: Pulls up on the energies of the most offending incorrect answer

Loss Function





- **Siamese models:** distance between the outputs of two identical copies of a model.
- **Energy function**: E(W,X1,X2) = ||Gw(X1)-Gw(X2)||
- If X1 and X2 are from the **same category (genuine pair)**, train the two copies of the model to produce **similar outputs (low energy)**
- If X1 and X2 are from **different categories (impostor pair)**, train the two copies of the model to produce **different outputs (high energy)**
- **Loss function: increasing function of genuine pair energy, decreasing function of impostor pair energy.**

Examples of Loss Functions

- **Most Offending Alien Point:**

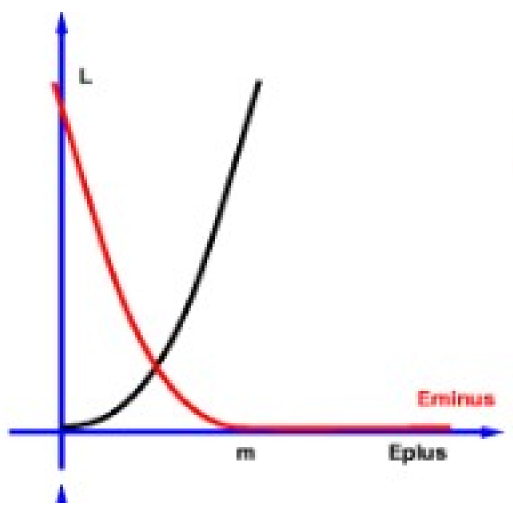
$$\bar{Y}^i = \text{argmin}_{y \neq Y^i} E(W, y, X^i)$$

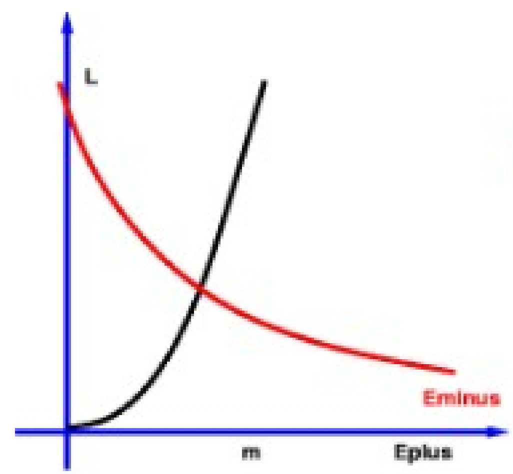
Square-Square Loss

$$\mathcal{L}(W) = \sum_i E(W, Y^i, X^i)^2 + \left(\max(0, m - \min_{Y \neq Y^i} E(W, Y, X^i))\right)^2$$

Square-Exponential Loss

$$\mathcal{L}(W) = \sum_i E(W, Y^i, X^i)^2 + K \exp\left(\min_{Y \neq Y^i} E(W, Y, X^i)\right)$$





Loss Function: Square-Exponential

- **Our Loss function for a single training pair (X1,X2)**:

$$L(W, X_1, X_2) = (1 - Y) L_G(E_{W}$$

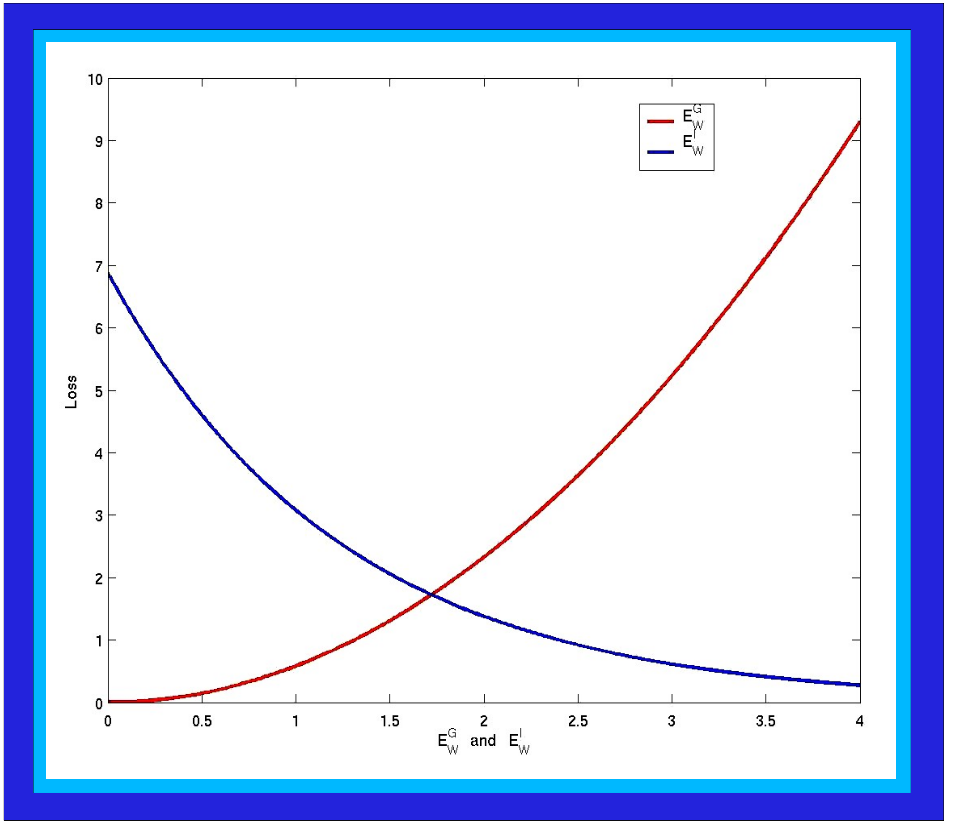
$$= (1 - Y)\frac{2}{R}(E_W(X_1, X_2$$

$$E_W(X_1, X_2) = \| G_W(X_1) - G_W($$

And R is the largest possible value of

$E_W(X_1, X_2)$

Y=0 for a genuine pair, and Y=1 for an impostor pair.



Face Verification datasets: AT&T, FERET, and AR/Purdue

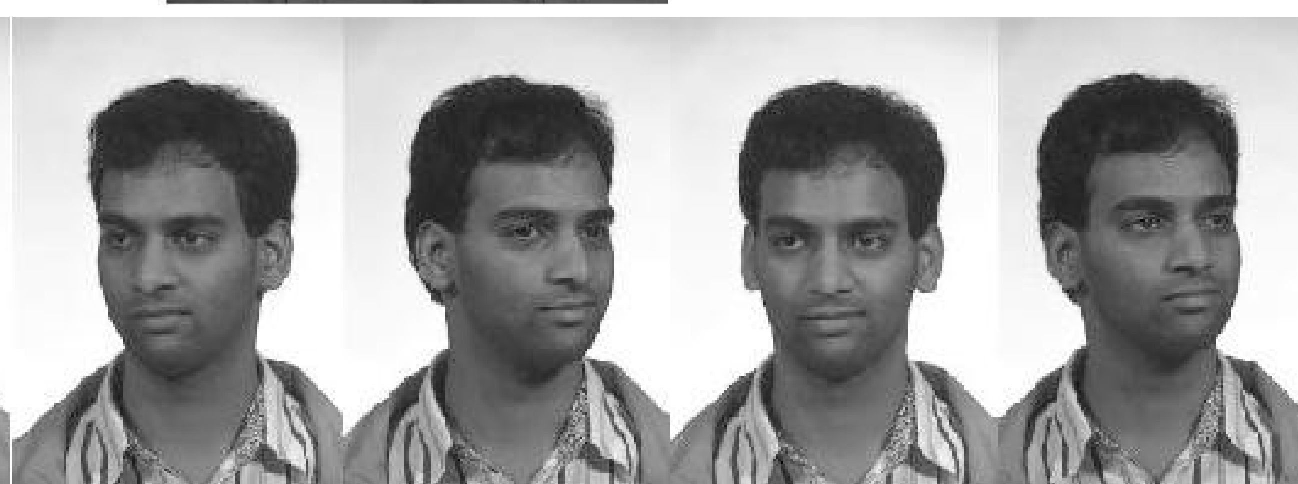
- The AT&T/ORL dataset
- Total subjects: **40**. Images per subject: **10**. Total images: **400**.
- Images had a moderate degree of variation in pose, lighting, expression and head position.
- Images from **35** subjects were used for training. Images from **5** remaining subjects for testing.
- Training set was taken from: **3500** genuine and **119000** impostor pairs.
- Test set was taken from: **500** genuine and **2000** impostor pairs.
- http://www.uk.research.att.com/facedatabase.html

AT&T/ORL Dataset

Face Verification datasets: AT&T, FERET, and AR/Purdue

- The FERET dataset. part of the dataset was used only for training.
- Total subjects: **96**. Images per subject: **6**. Total images: **1122**.
- Images had high degree of variation in pose, lighting, expression and head position.
- The images were used for training only.
- http://www.itl.nist.gov/iad/humanid/feret/

FERET Dataset



Face Verification datasets: AT&T, FERET, and AR/Purdue

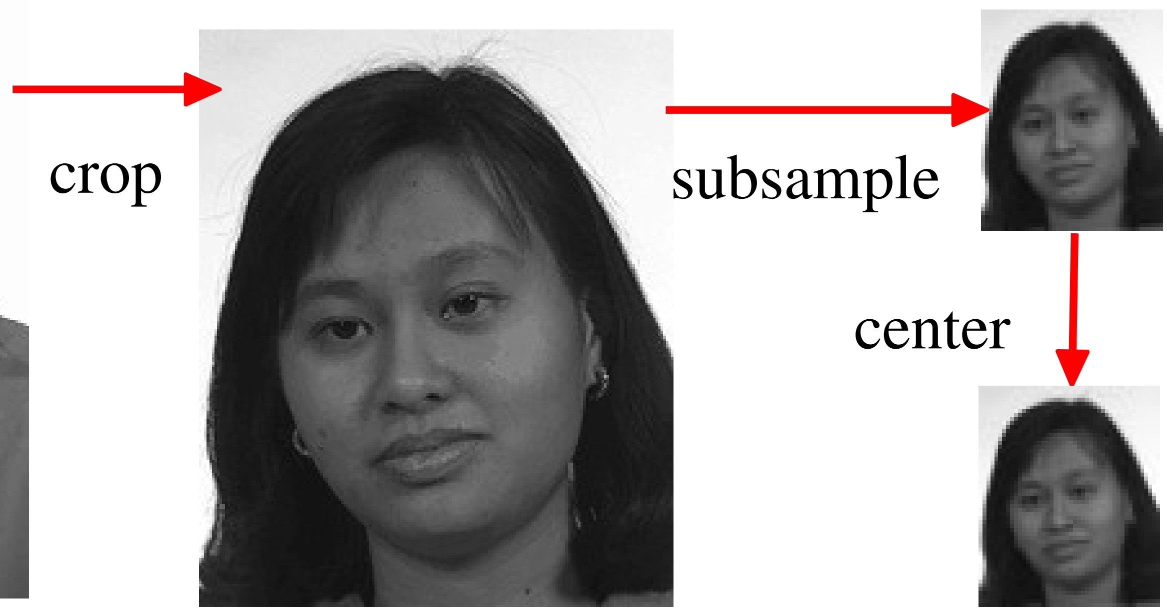
- The AR/Purdue dataset
- Total subjects: **136**. Images per subject: **26**. Total images: **3536**.
- Each subject has 2 sets of 13 images taken 14 days apart.
- Images had very high degree of variation in pose, lighting, expression and position. Within each set of 13, there are 4 images with expression variation, 3 with lighting variation, 3 with dark sun glasses and lighting variation, and 3 with face obscuring scarfs and lighting variation.
- Images from **96** subjects were used for training. The remaining **40** subjects were used for testing.
- Training set drawn from: **64896** genuine and **6165120** impostor pairs.
- Test set drawn from: **27040** genuine and **1054560** impostor pairs.
- http://rv11.ecn.purdue.edu/aleix/aleix_face_DB.html

Face Verification dataset: AR/Purdue

Preprocessing

The 3 datasets each required a small amount of preprocessing.

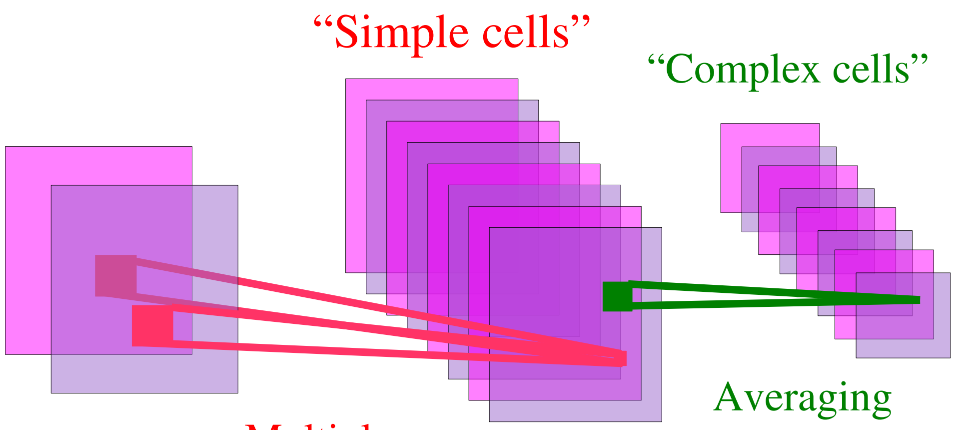
- **FERET:** Cropping, subsampling, and centering (see below)
- **AR/PURDUE**: Cropping and subsampling
- **AT&T:** Subsampling only



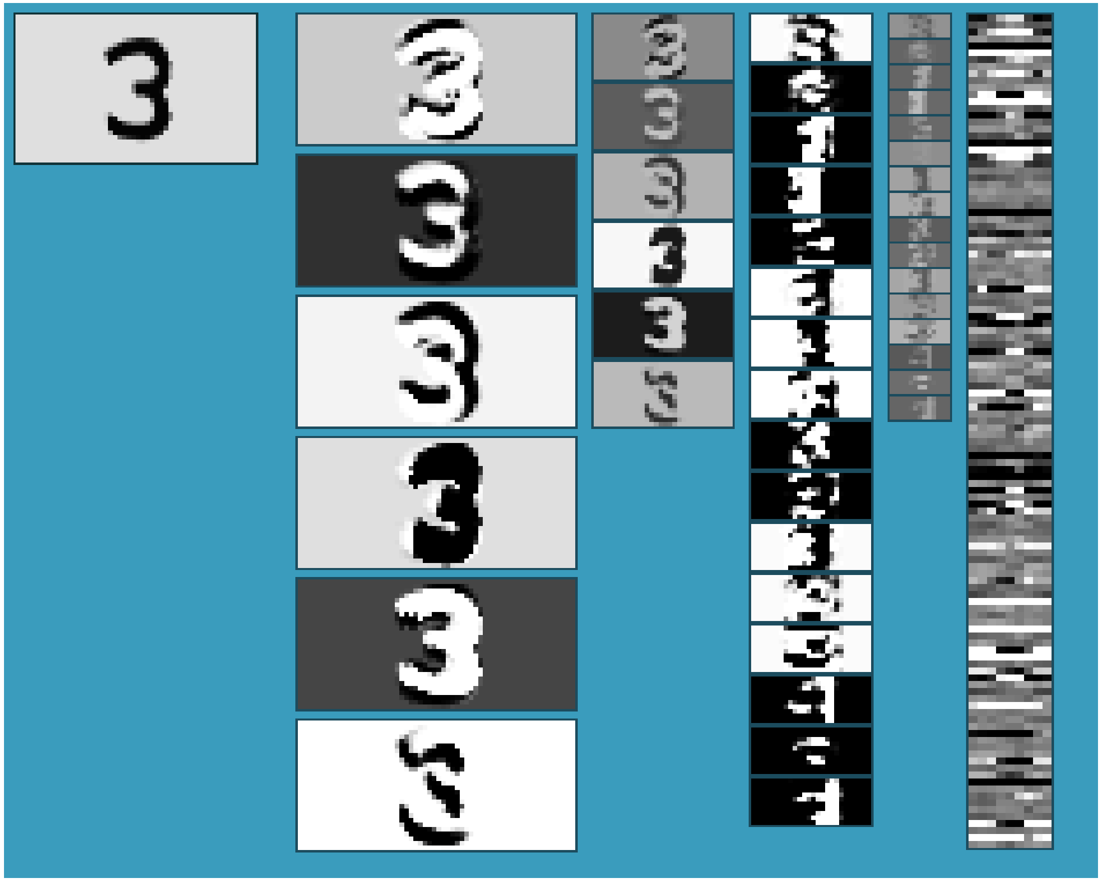
Centering with a Gaussian-blurred face template

- Coarse centering was done on the FERET database images
  1. Construct a template by blurring a well-centered face.
  2. Convolve the template with an uncentered image.
  3. Choose the 'peak' of the convolution as the center of the image.

Alternated Convolutions and Subsampling

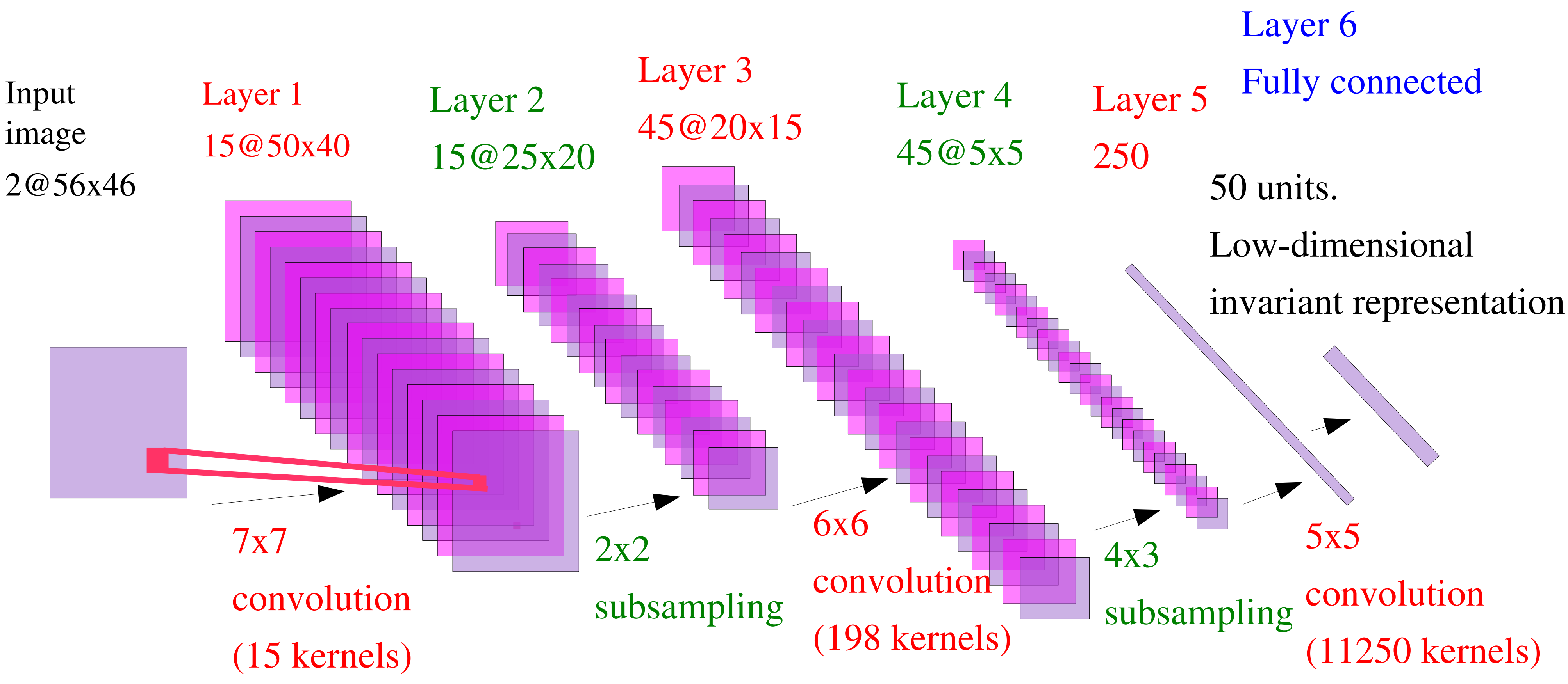


- Local features are extracted everywhere.
- averaging/subsampling layer builds robustness to variations in feature locations.
- Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....



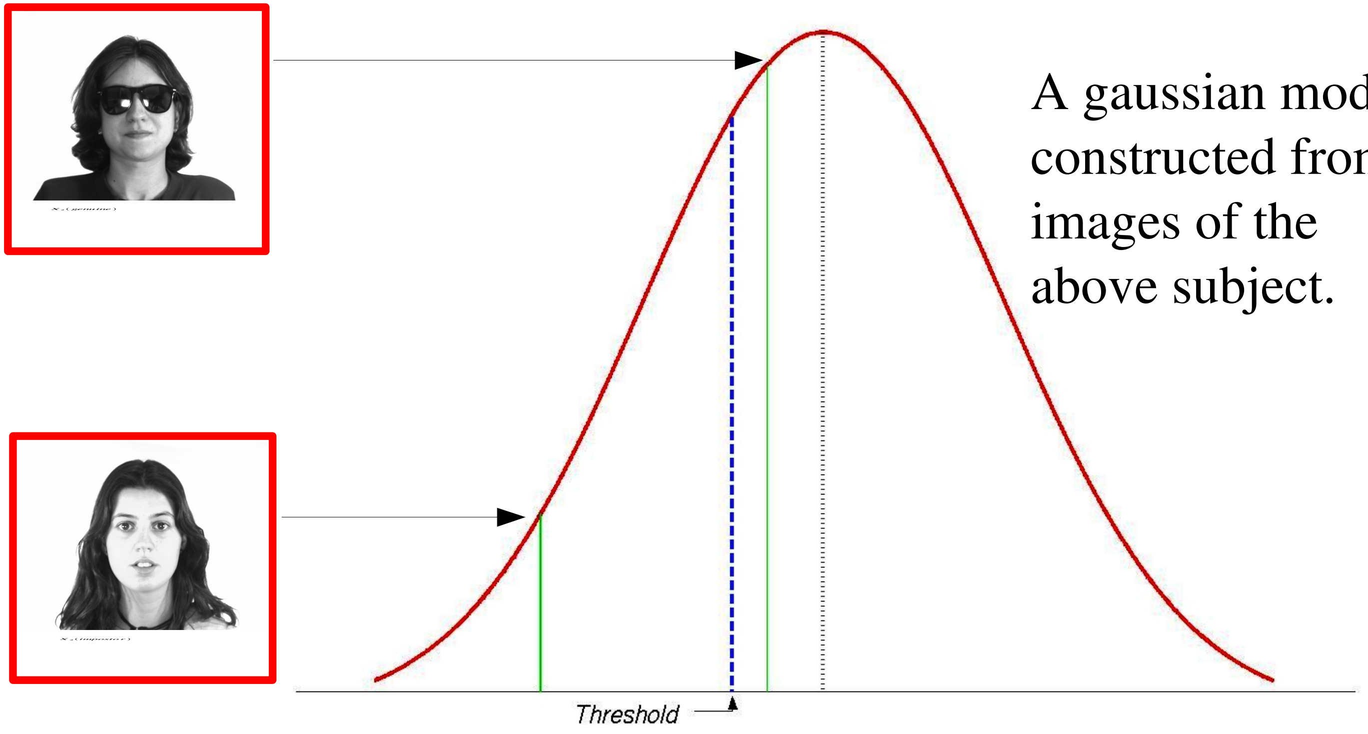
Architecture for the Mapping Function Gw(X)

Convolutional net



Internal state for genuine and impostor pairs

Gaussian Face Model in the output space



A gaussian model constructed from 5 images of the above subject.

Dataset for Verification

- tested on AT&T and AR/Purdue
- AT&T dataset

Number of subjects:	5
Images/subject:	10
Images/Model:	5
Total test size:	5000
Number of Genuine:	500
Number of Impostors:	4500

- Purdue/AR dataset

Number of subjects:	40
Images/subject:	26
Images/Model:	13
Total test size:	5000
Number of Genuine:	500
Number of Impostors:	4500

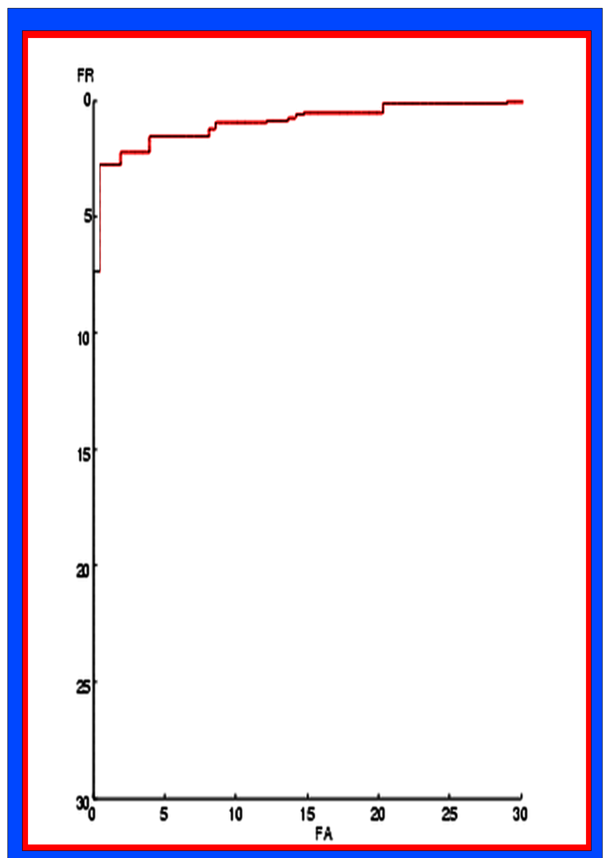
Verification Results

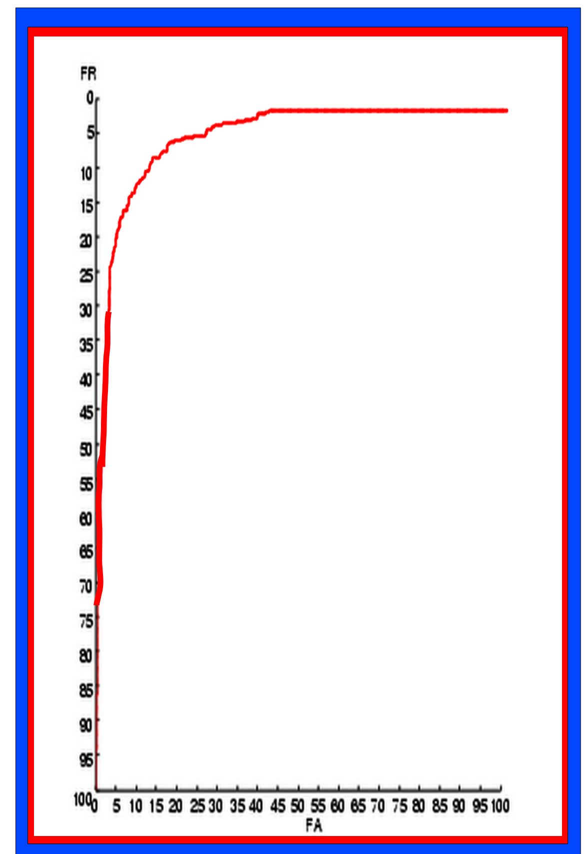
- The AT&T dataset

False Accept	False Reject
10.00%	0.00%
7.50%	1.00%
5.00%	1.00%

- The AR/Purdue dataset

False Accept	False Reject
10.00%	11.00%
7.50%	14.60%
5.00%	19.00%





Classification Examples

Example: Correctly classified genuine pairs

energy: 0.3159

energy: 0.0043

energy: 0.0046

Example: Correctly classified impostor pairs

energy: 20.1259

energy: 32.7897

energy: 5.7186

Example: Mis-classified pairs

energy: 10.3209

energy: 2.8243

Internal State

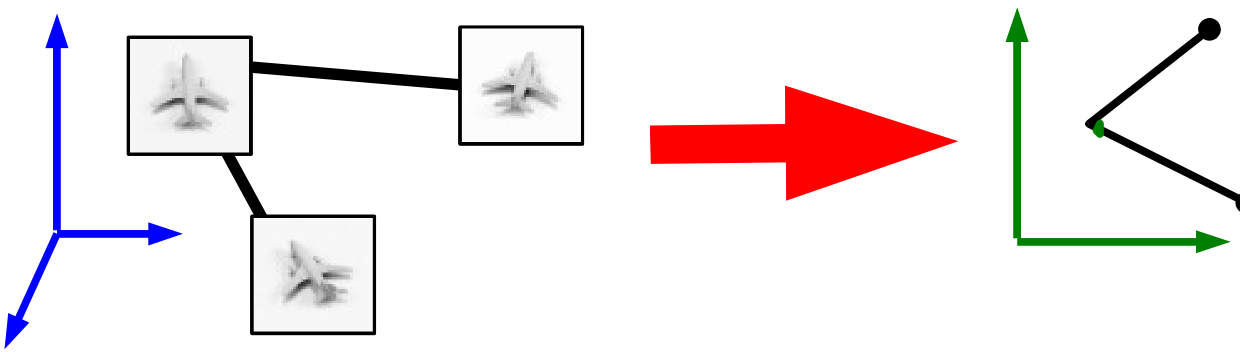
Linear Version

- **Recently, Weinberger, Blitzer and Saul [NIPS 06] proposed a version of this that uses a hinge loss, but is restricted to linear mappings.**
  - They show that semi-definite programming can be used to optimize the loss in that case.

DrLim: Dimensionality Reduction by Learning an Invariant Mapping

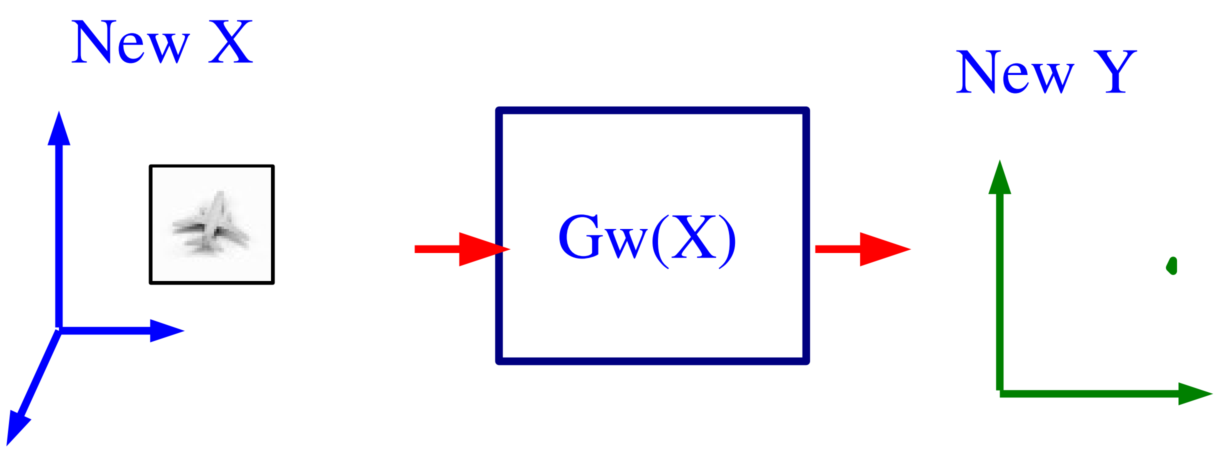
[Hadsell, Chopra, LeCun, CVPR 2006]

"Traditional" Manifold Learning



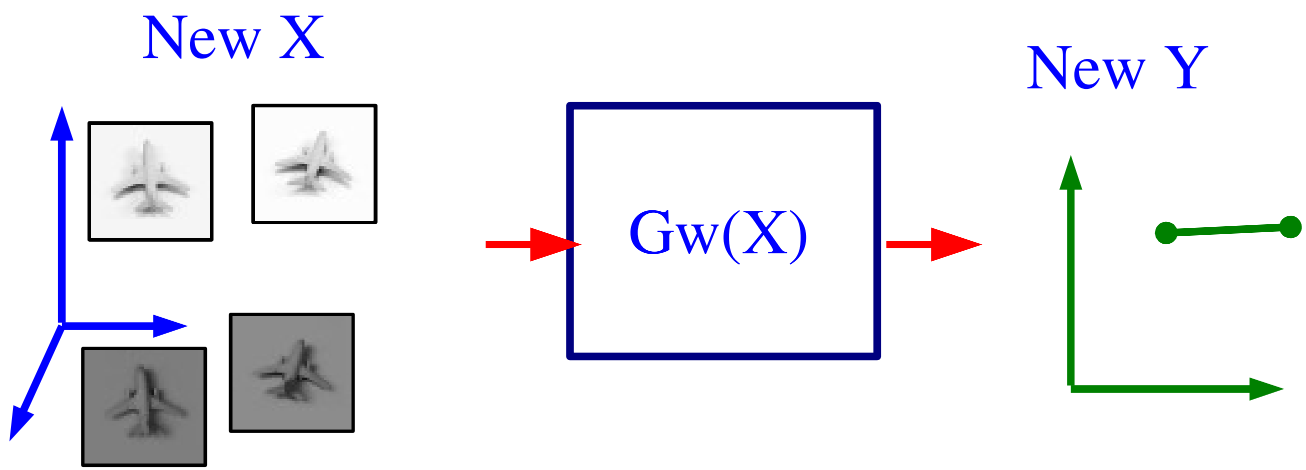
- **LLE, Laplacian Eigenmaps, and Hessian LLE: map a given set of high dimensional points to a corresponding set low-dimensional points.**
  - **All the points must be known in advance.**
  - **New points whose relationship to the original training points is not known cannot be mapped to the low-dimensional space.**
  - **There is no real "function" that maps input objects to low-dimensional output vectors.**
  - **With LLE: a "meaningful" and computable distance metric between input objects must be devised.**

Learning a FUNCTION from input to output

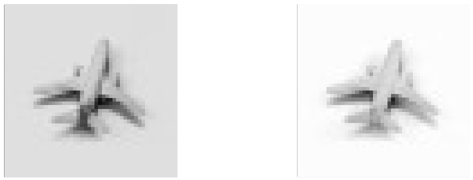


- **With a function, new points can be mapped easily**
  - We do not need to come up with a similarity metric in input space
  - We do not need to know the relationship of new points to training points
- **Questions:**
  - How do we do it? What loss function?
  - How to we determine that two samples are “similar”?

Learning an INVARIANT FUNCTION from input to output



- **We want the mapping to be invariant to irrelevant variations of the input**
  - **Example 1:** the low-dim image of an airplane should be independent of its illumination.
  - **Examples 2:** the low-dim image of a handwritten character should be independent of its position in the frame



Previous Work

- **Some methods generate a mapping, but rely on computable distance metrics in input space.**
  - Principal Component Analysis (PCA)
  - ISOMAP
  - Local Linear Embedding (LLE)
  - Multidimensional Scaling (MDS) – in Classical Sense
- **Others don't rely on distance metrics, but they do not generate a function.**
  - Laplacian EigenMaps
  - Hessian LLE
  - Kernel PCA

What do we want?

- Learning low-dimensional manifolds with invariance to irrelevant transformation of the inputs
- Taking advantage of prior knowledge about which sample is "semantically" similar to which other sample.
- Learning a MAPPING (an actual function) that maps inputs to the low-dimensional space, so we can apply it to new patterns whose relationship to the training samples is unknown
- Allowing complicated non-linear mapping from input to low-dimensional representations
- Relying solely on neighborhood relationships, and not requiring the existence of a computable distance metric between input patterns. So that the method can be used to any object.
- Finding a manifold in which the samples are uniformly distributed

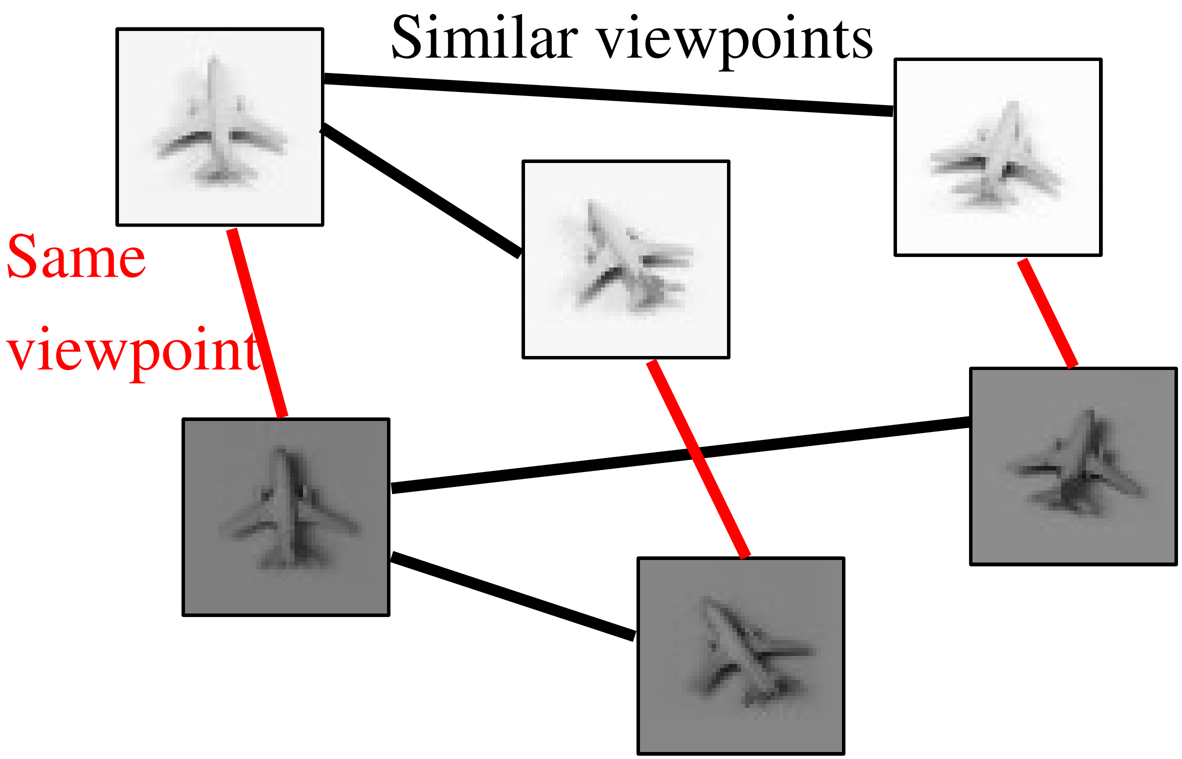
Learning Invariant Manifolds with EBMs

RECIPE

- **Build a neighborhood graph** of the training samples, possibly using prior knowledge. Two samples are neighbors if they are semantically similar.
- Pick a **parameterized family of functions** from inputs to low-dimensional output vectors (neural nets, RBF, whatever)
- Optimize the parameters of the function so as to minimize a **loss function** that make the **distance between the output vector of neighbors small**, and the **distance between output vectors of non-neighbors large**.
- **Apply the trained function to new (test) samples**

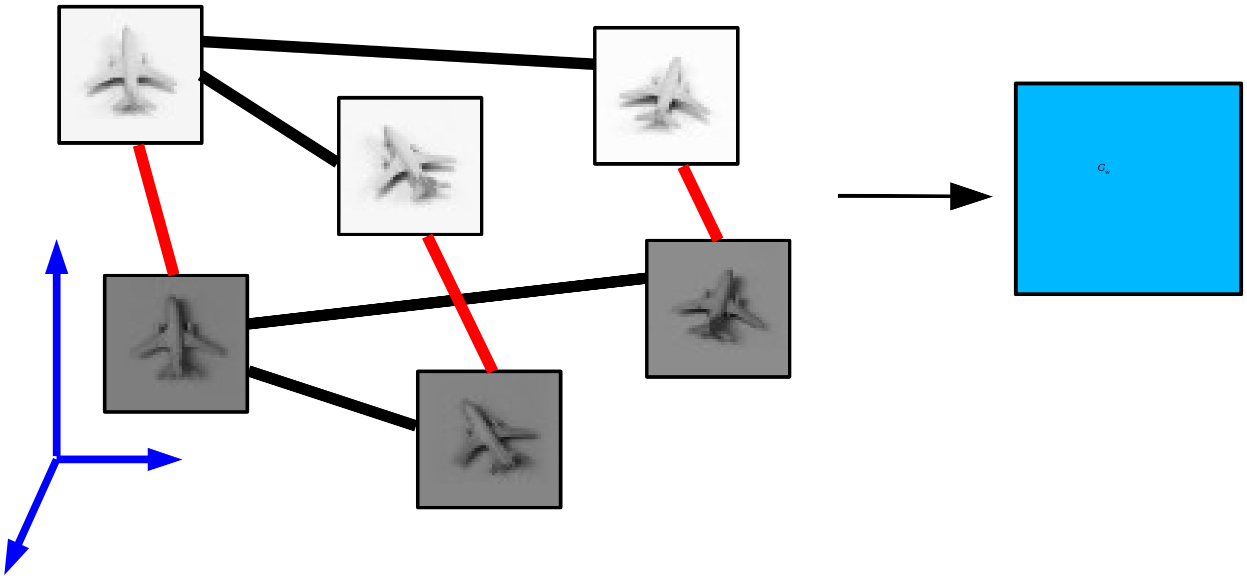
Step 1: Building a Neighborhood Graph

- **Build a graph between training samples such that:**
  - Semantically “similar” patterns have an edge between them.
  - Semantically “different” pattens don't.
- **Prior knowledge can be used to build the graph**



Step 2: Pick a Parameterized Family of Function

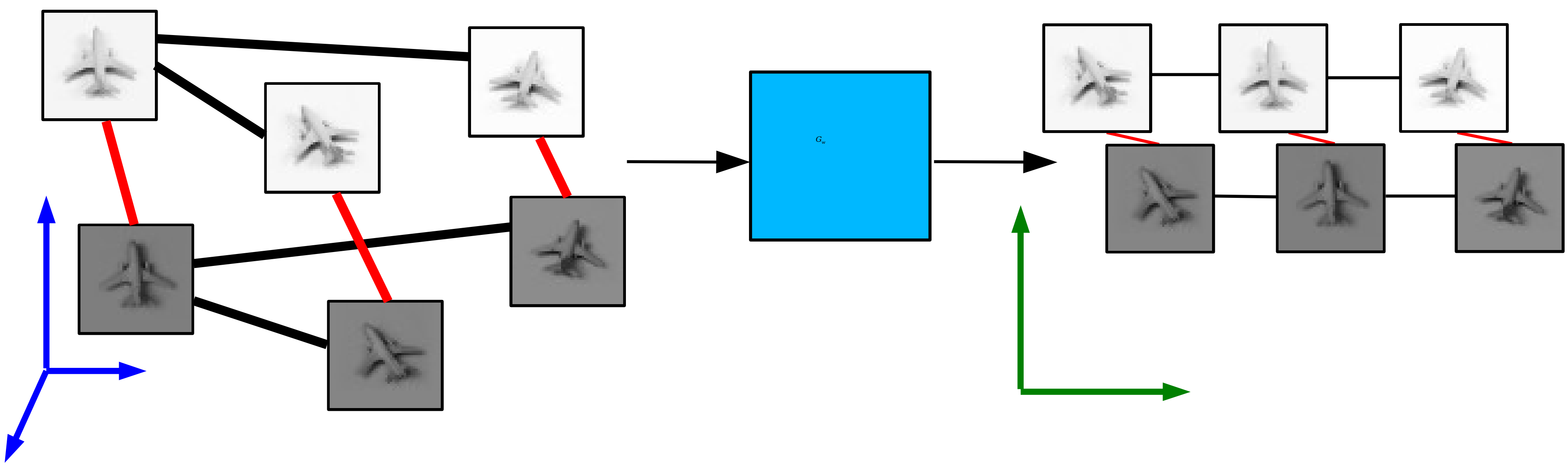
- **The function can be anything:**
  - Neural net, RBF, other non-linear families
- **There is no restriction on the form of the function family**
  - But it's better if it's smooth.
  - W: parameters vector



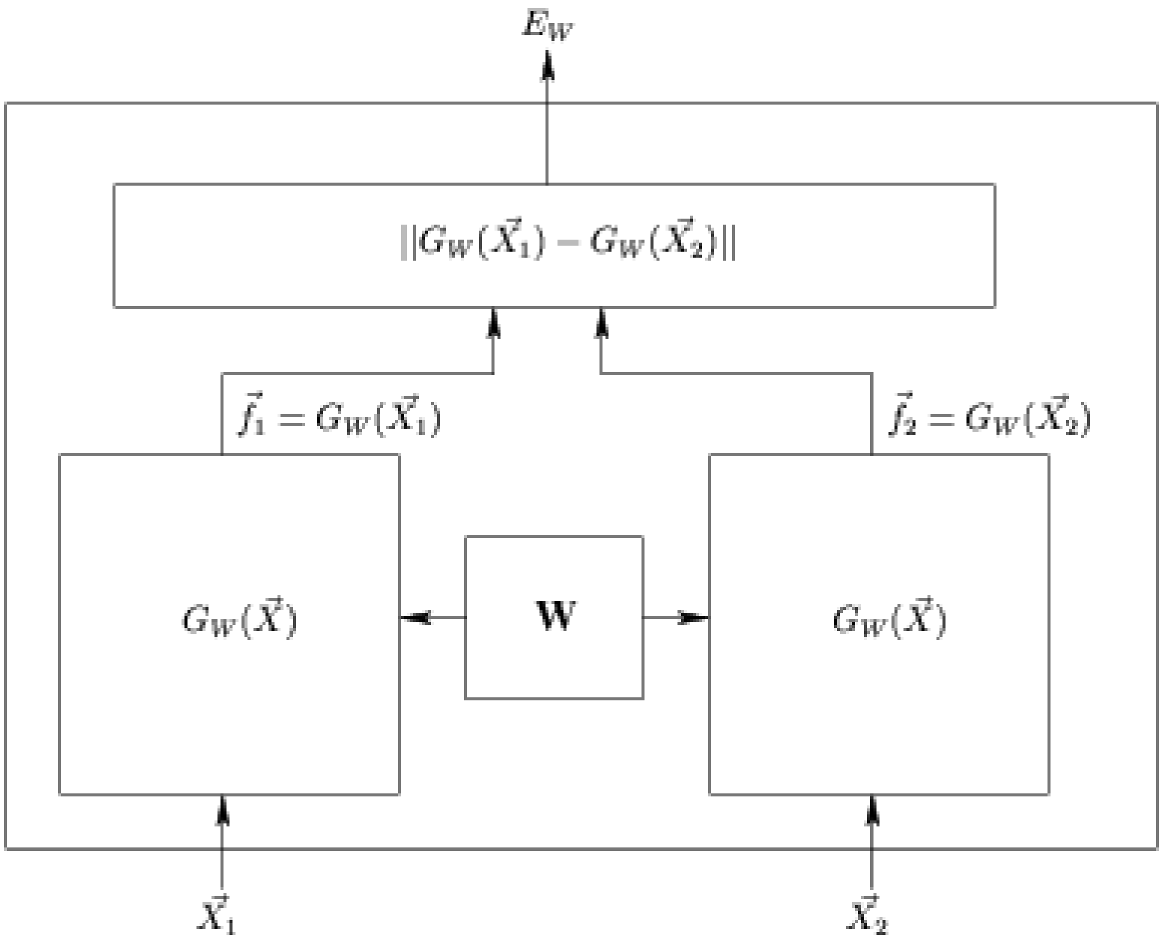
Step 3: Pick a Loss function and Minimize it w.r.t. W

Loss function:

- Outputs corresponding to input samples that are neighbors in the neigborhood graph should be nearby
- Outputs for input samples that are not neighbors should be far away from each other



Architecture



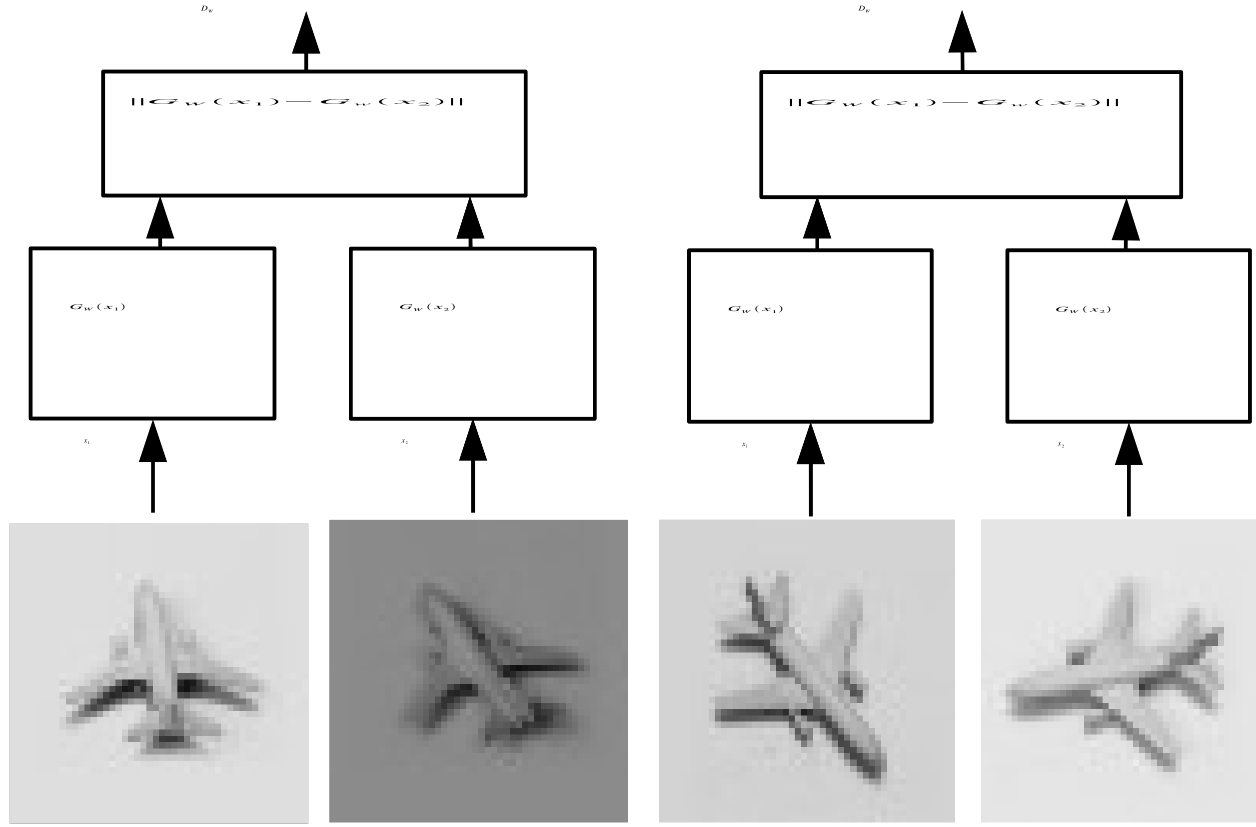
- **Siamese Architecture [Bromley, Sackinger, Shah, LeCun 1994]**

Architecture and loss function

- **Loss function:**
  - Outputs corresponding to input samples that are neighbors in the neigborhood graph should be nearby
  - Outputs for input samples that are not neighbors should be far away from each other

Make this small

Make this large



Similar images (neighbors in the neighborhood graph)

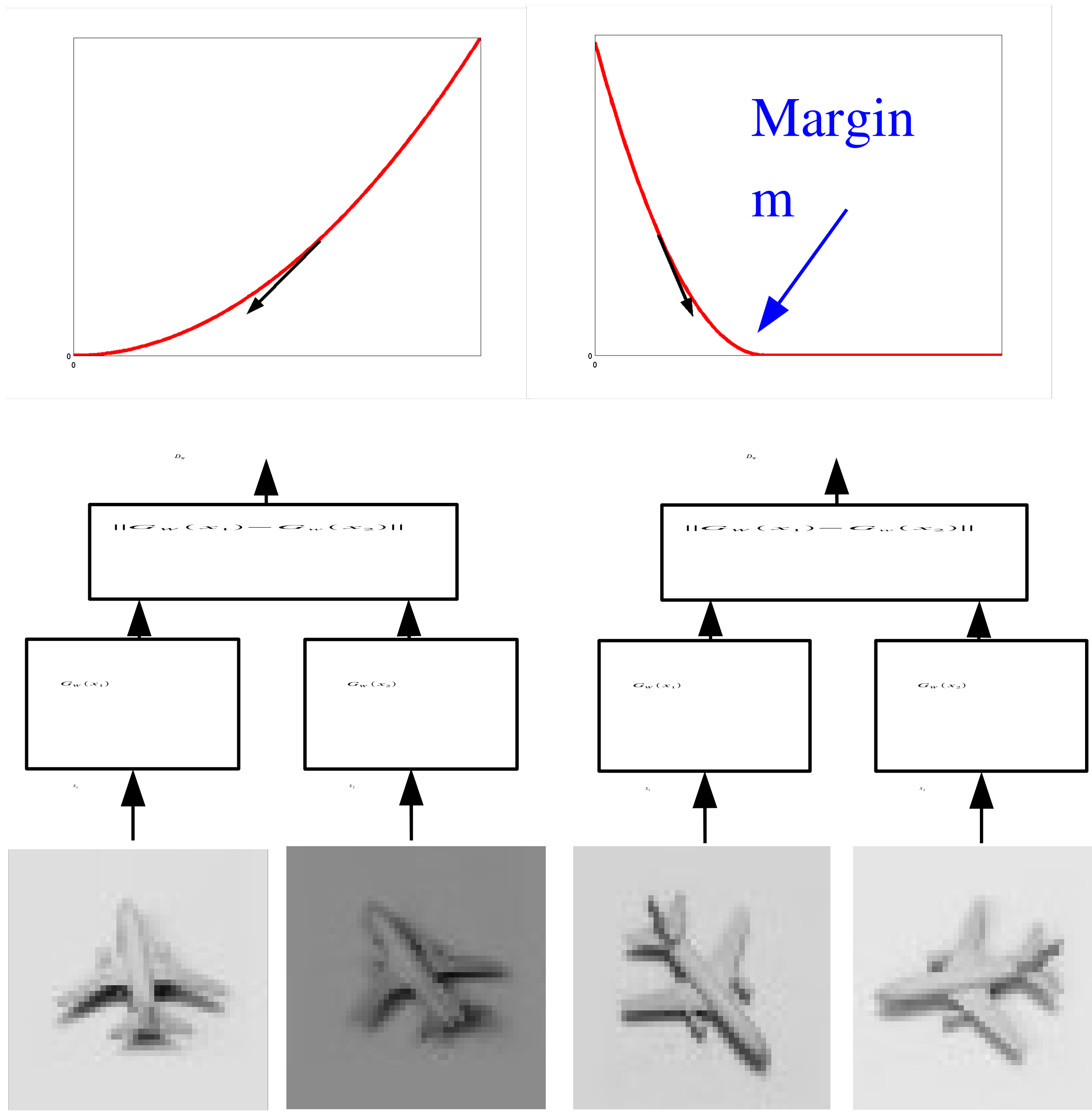
Dissimilar images (non-neighbors in the neighborhood graph)

Loss function

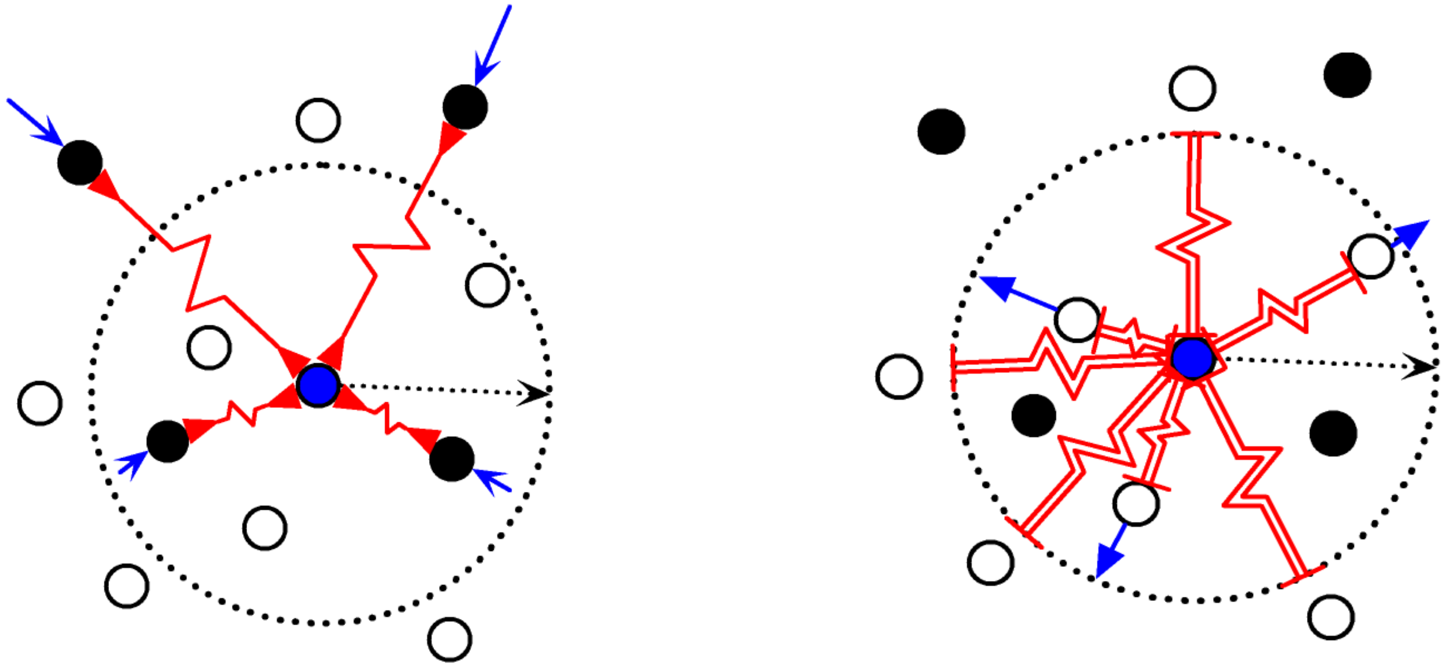
$L_{similar} = \frac{1}{2} D_w^2$

$L_{dissimilar} = \frac{1}{2} \{ max$

- **Loss function:**
  - Pay quadratically for making outputs of neighbors far apart
  - Pay quadratically for making outputs of non-neigbors smaller than a **margin** m

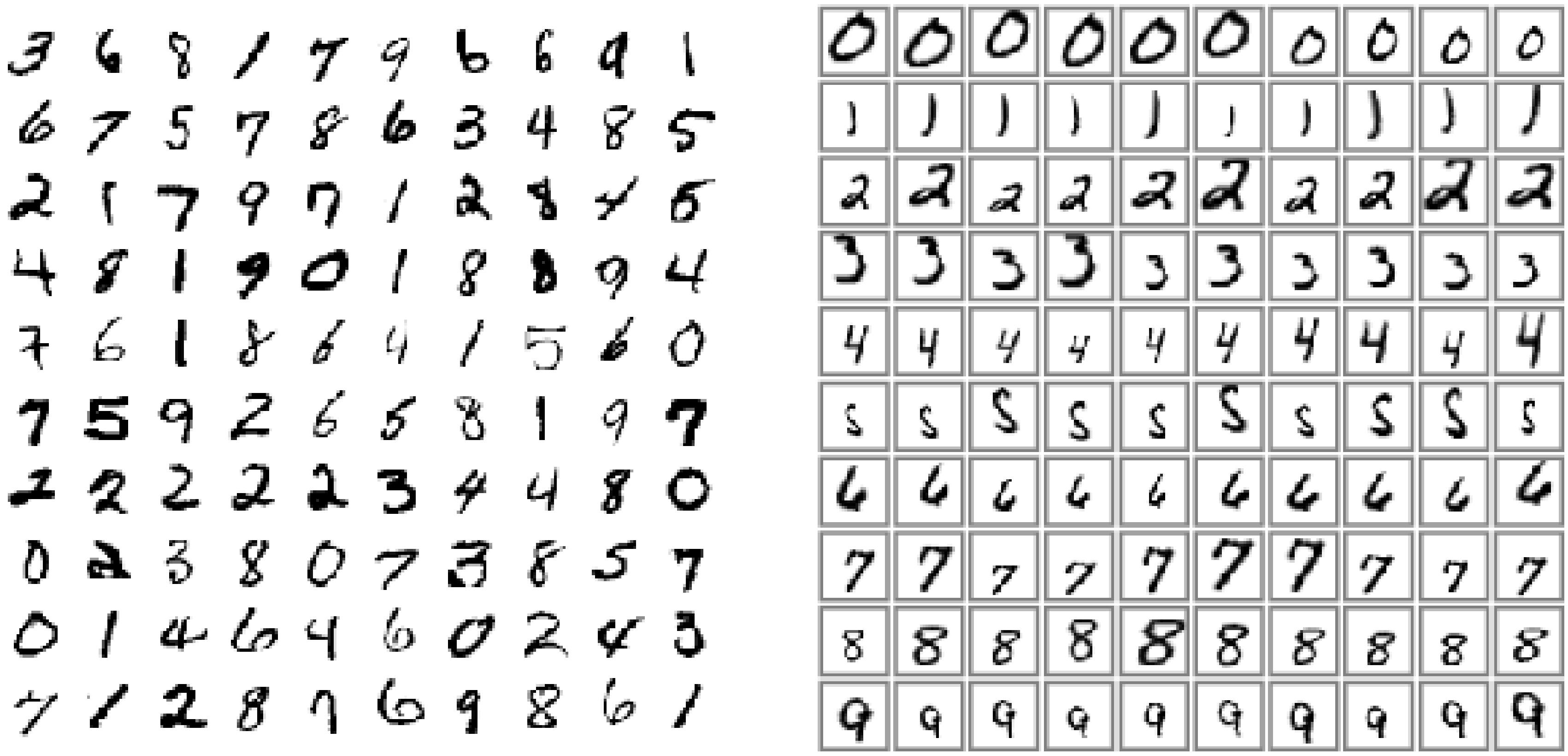


Mechanical Analogy



- **The output vectors for graphs neighbors (black points) are pulled together by a spring**
- **The output vectors of non-neighbors (white points) are repelled by a spring whose rest length is equal to the margin**
  - The value of the margin sets an arbitrary scale for the output space

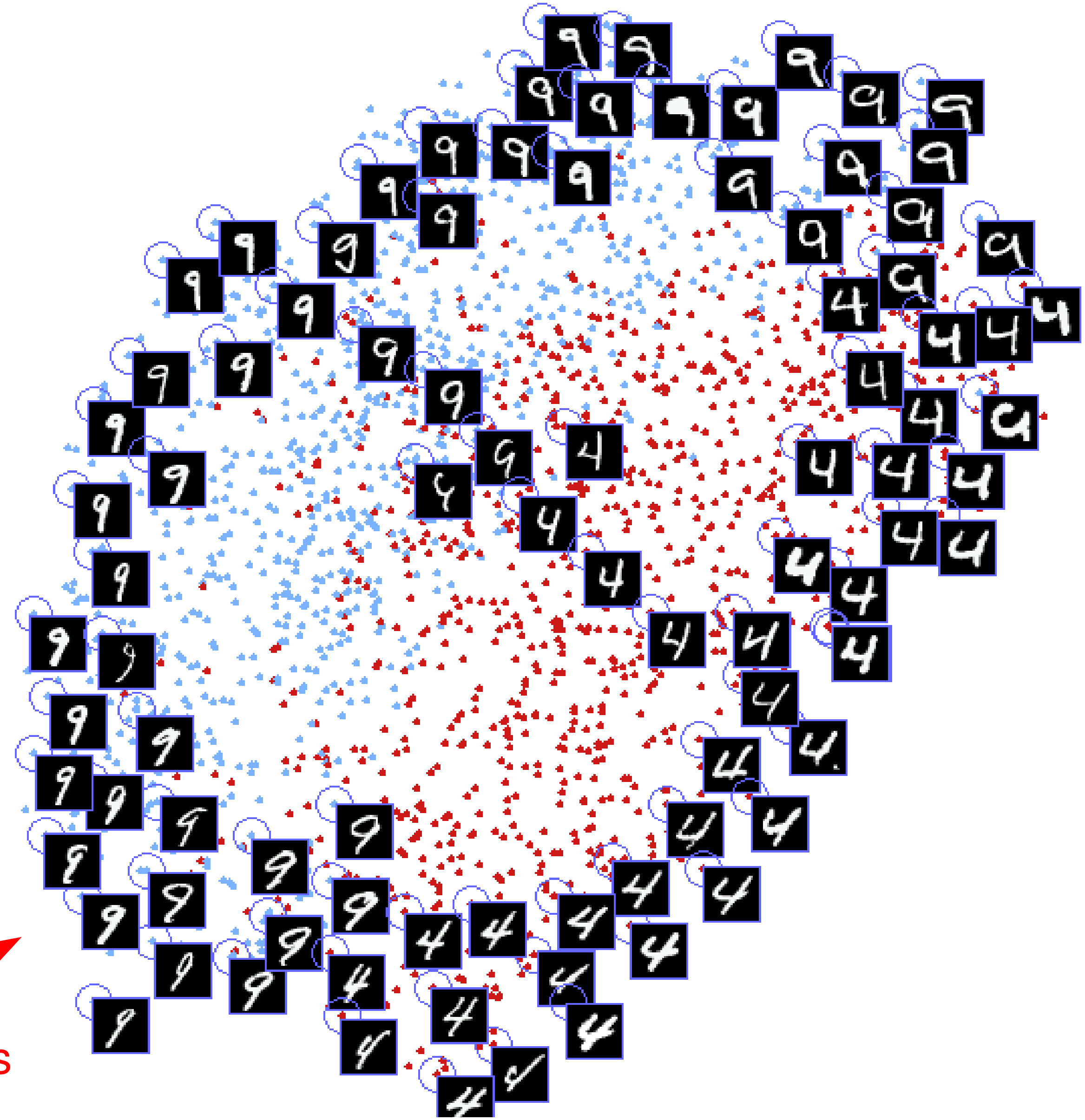
MNIST Dataset



- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

MNIST Handwritten Digits. Sanity Check

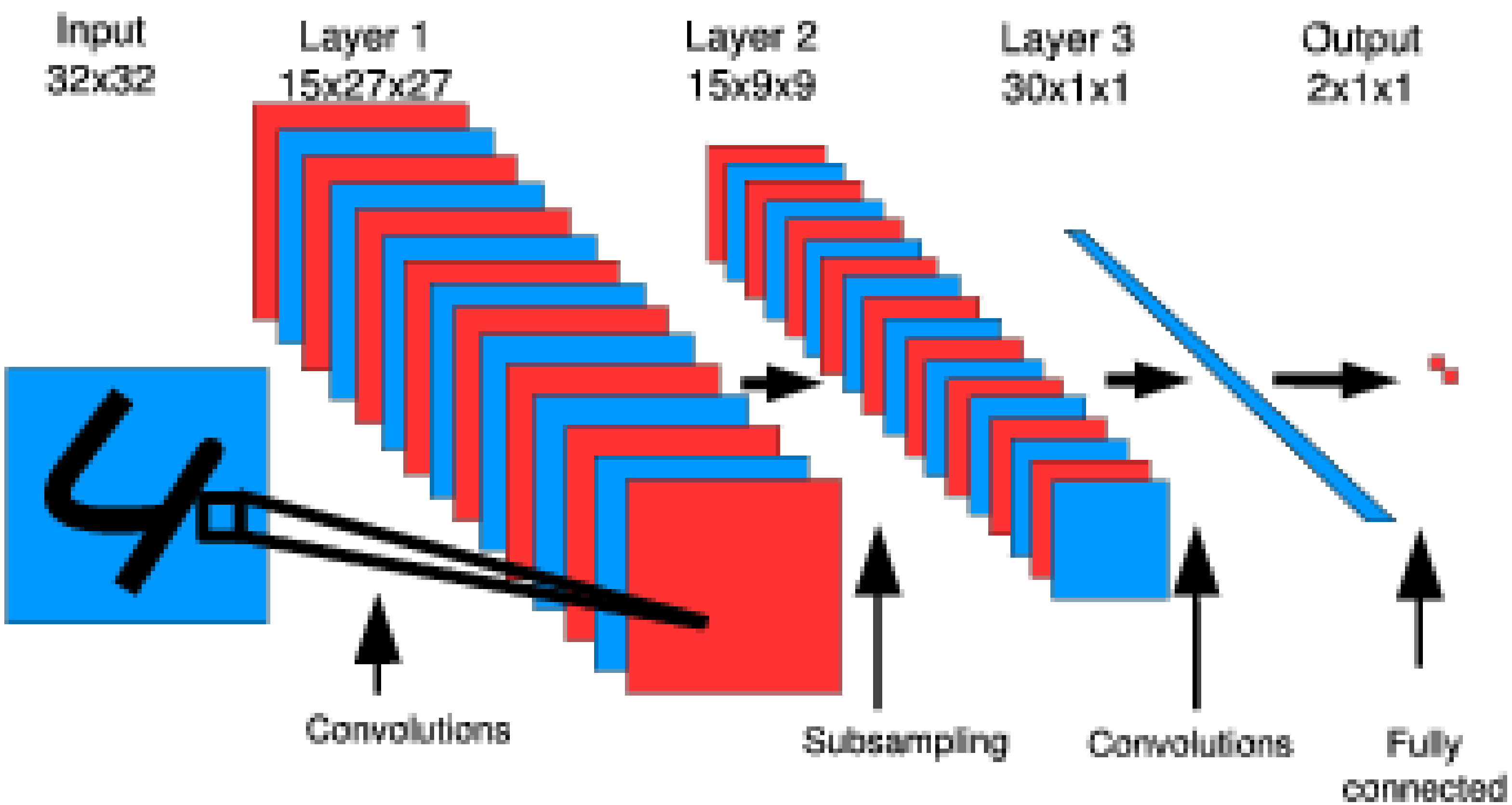
- **Objective:** Sanity check using undistorted images. No use of any prior knowledge.
- **Neighbors:** 5 nearest neighbors in euclidean space.
- **Training:** 3000 samples each of handwritten 4's and 9's.
- **Testing:** 1000 samples each of 4's and 9's.
- **Architecture:** Input dimension: 32x32. Output dimension: 2. A 4 layer Convolutional Network.



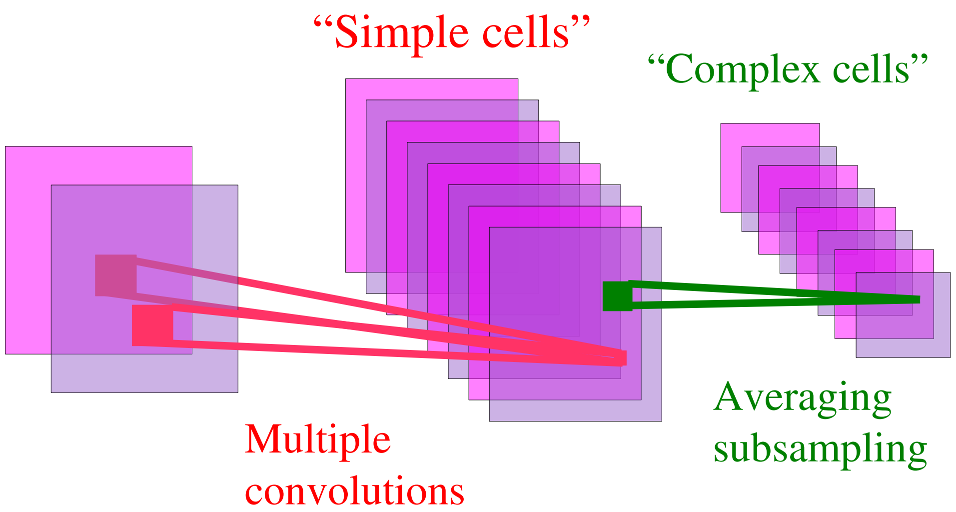
test samples

Architecture of the Gw(X) Function:

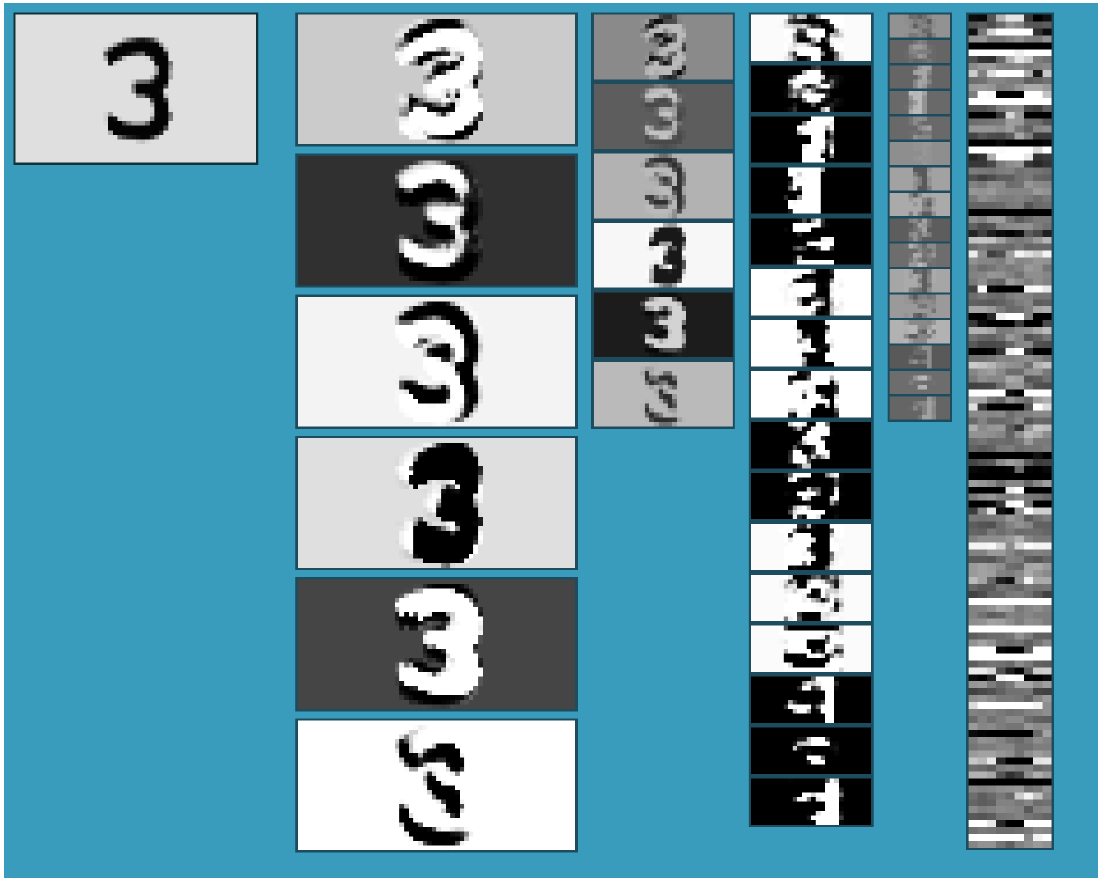
A small convolutional net



Alternated Convolutions and Subsampling

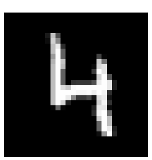


- Local features are extracted everywhere.
- averaging/subsampling layer builds robustness to variations in feature locations.
- Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....

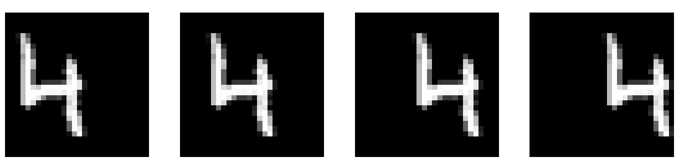


Learning a mapping that is invariant to shifts

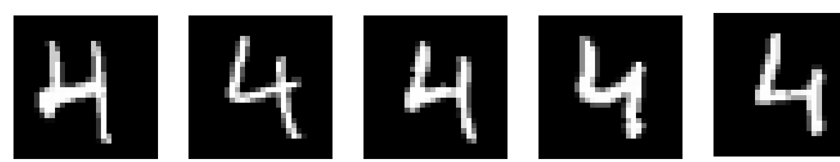
- **The position of a digit in the image frame is irrelevant**
- **Can we learn a mapping that is invariant to shifts?**
- **Dataset: Each digit is horizontally shifted by -6, -3, 0, 3, 6 pixels**
- **Neighborhood Graph: 5 (unshifted) nearest neighbors in Euclidean distance**



Original

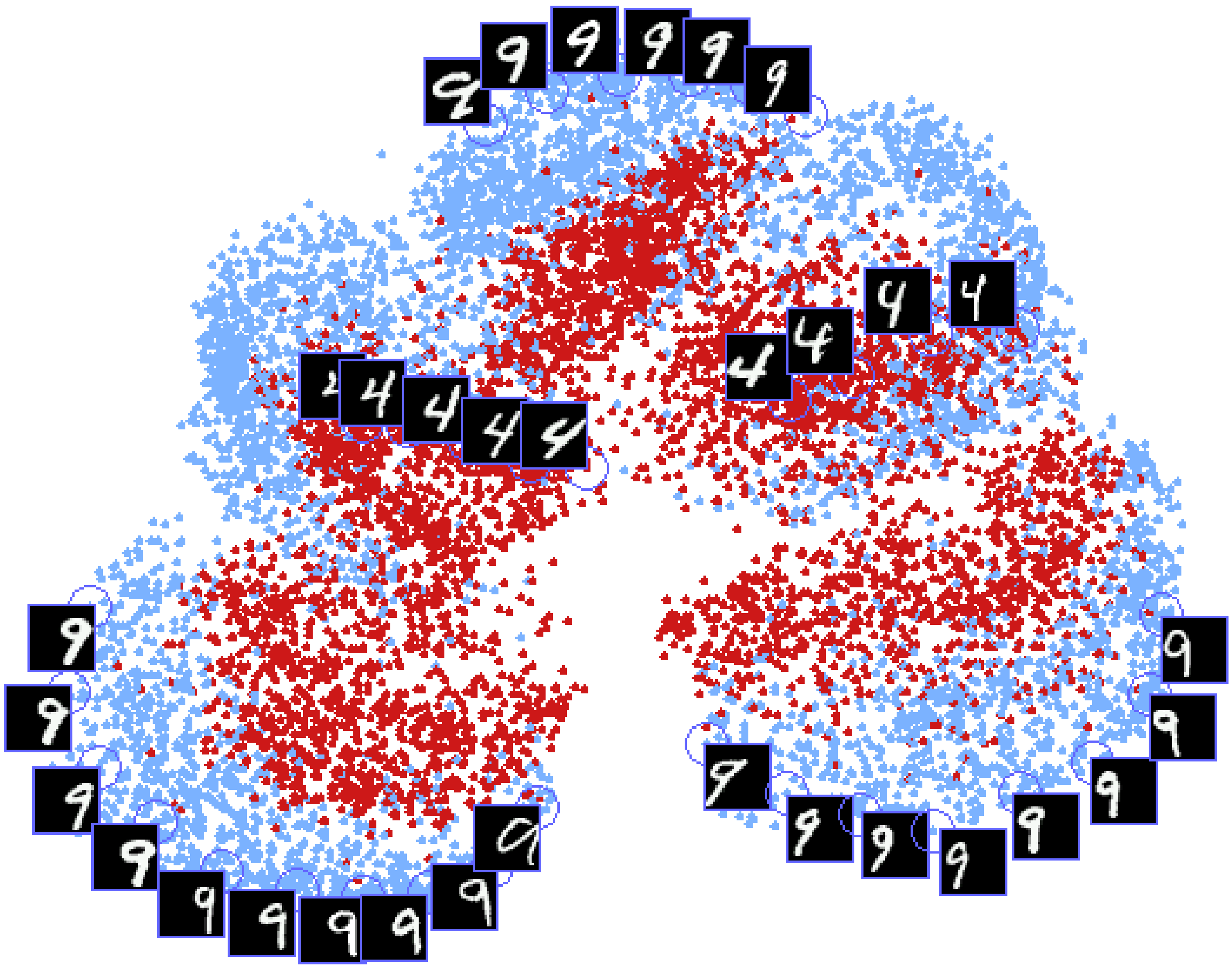


Translations of original



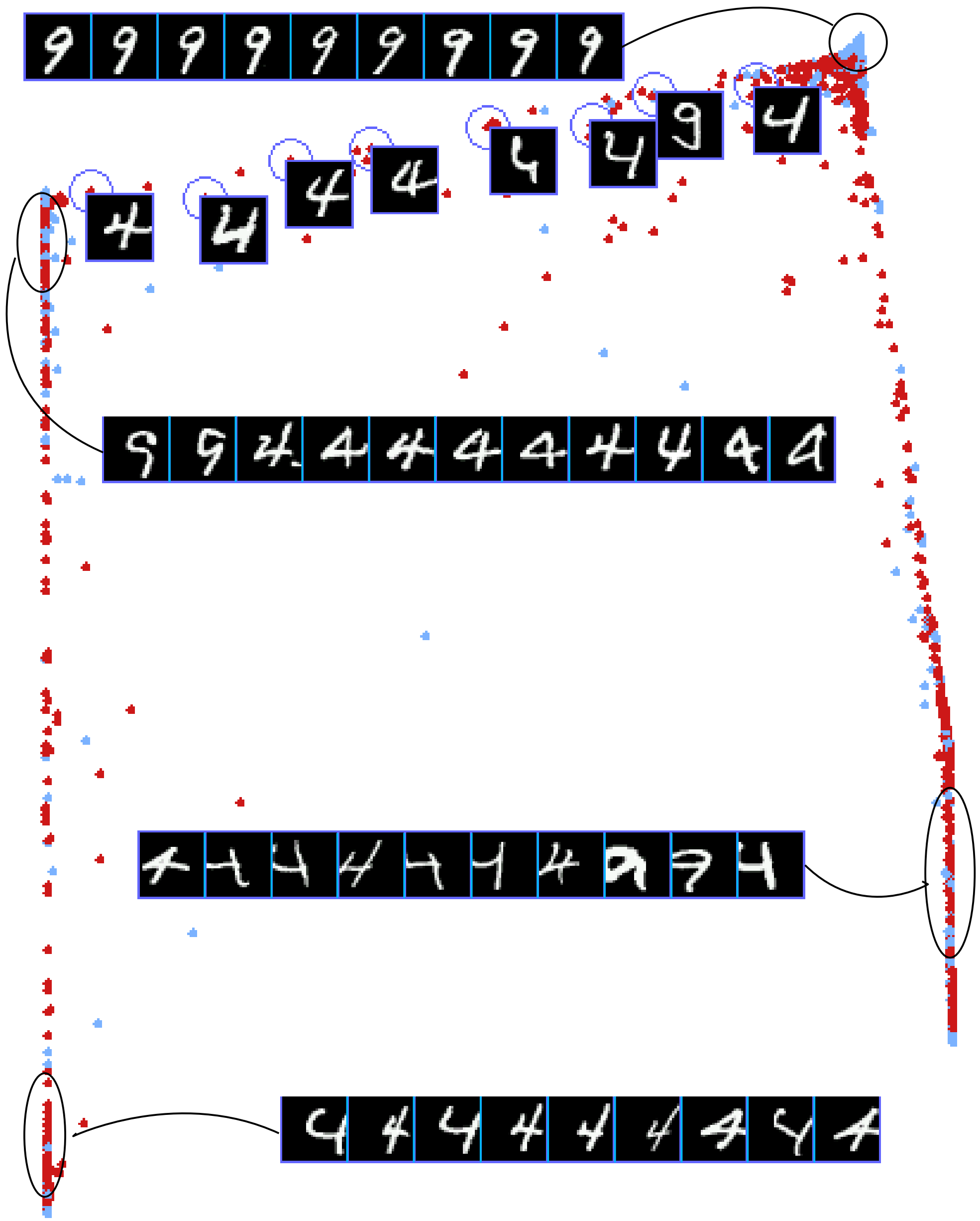
Nearest Neighbors of original

Simple Experiment with Shifted MNIST



- Training set: 3000 “4” and 3000 “9” from MNIST. **Each digit is shifted horizontally by -6, -3, 3, and 6 pixels**
- Test set (shown) 1000 “4” and 1000 “9”
- Neighborhood graph: 5 nearest neighbors in Euclidean distance.
- Output Dimension: 2

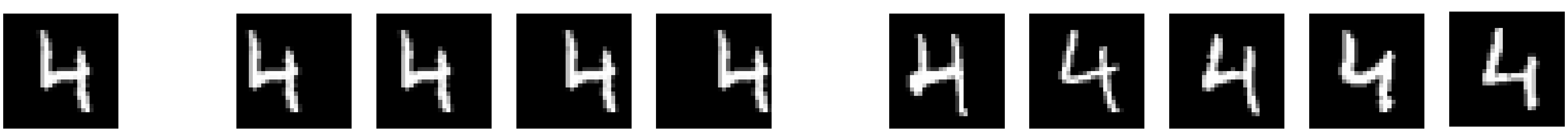
Shifted MNIST: LLE Result



- Training set: 3000 "4" and 3000 "9" from MNIST. **Each digit is shifted horizontally by -6, -3, 3, and 6 pixels**
- Neighborhood graph: 5 nearest neighbors in Euclidean distance,
- Output Dimension: 2
- Test set (shown) 1000 "4" and 1000 "9"

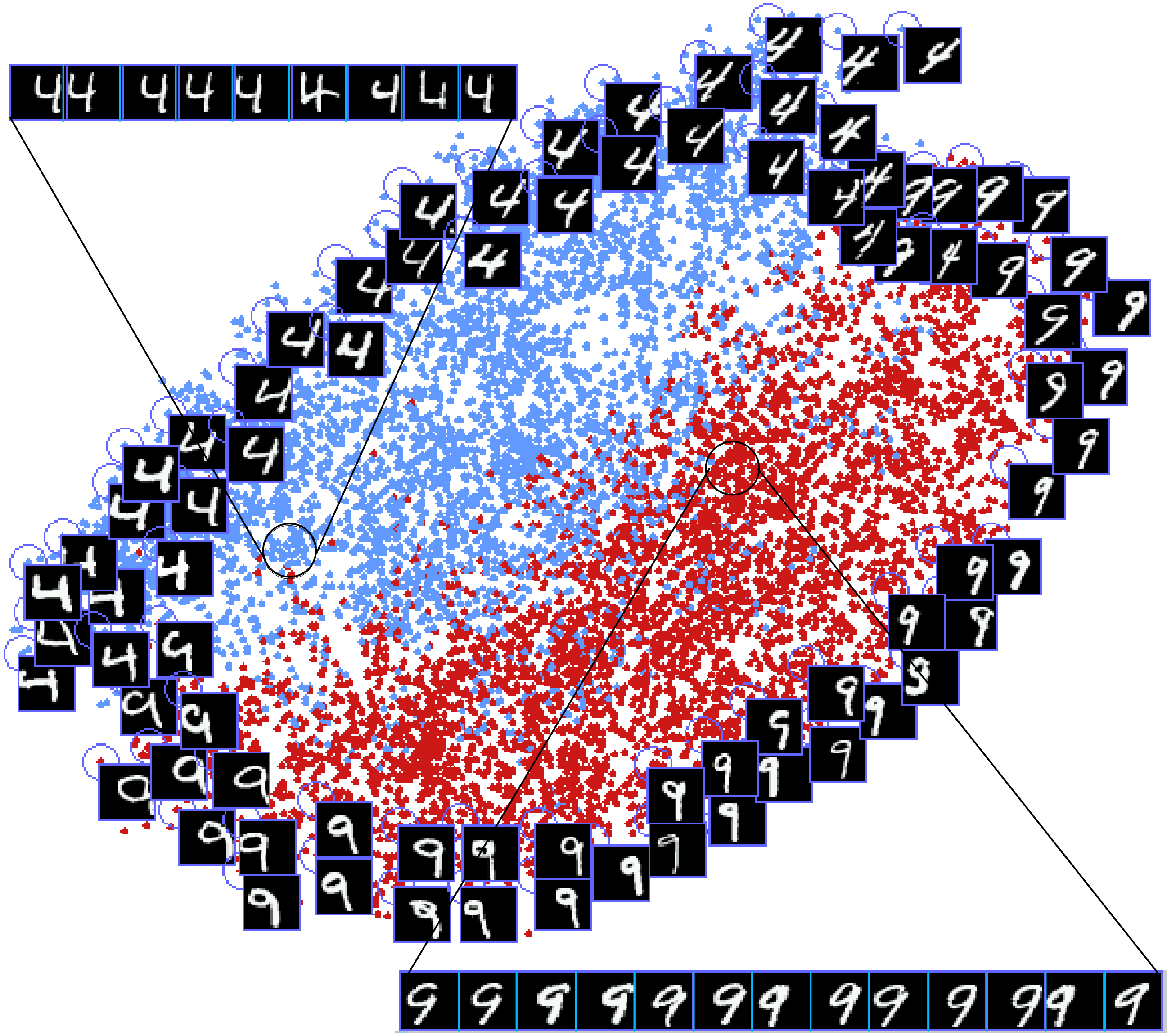
Shift-Invariant mapping: using prior knowledge

- **The position of a digit in the image frame is irrelevant**
- **Can we learn a mapping that is invariant to shifts?**
- **Dataset: Each digit is horizontally shifted by -6, -3, 0, 3, 6 pixels**
- **Neighborhood Graph: an edge is placed between each sample and**
  - Shifted versions of itself
  - Its 5 (unshifted) nearest neighbors in Euclidean distance
  - The shifted versions of its 5 Euclidean nearest neighbors



Original   Translations of original   Nearest Neighbors of original

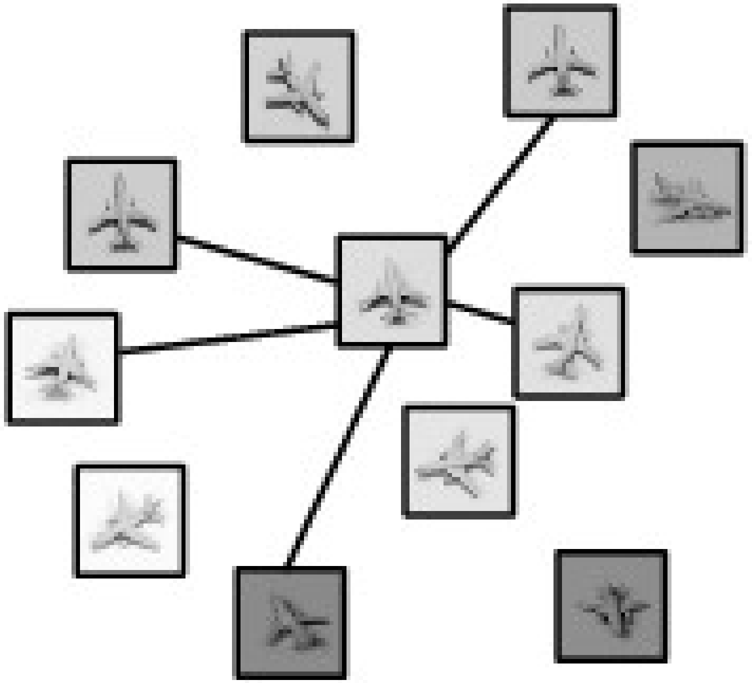
Shifted MNIST: Injecting Prior Knowledge



- Training set: 3000 “4” and 3000 “9” from MNIST. **Each digit is shifted horizontally by -6, -3, 3, and 6 pixels**
- Neighborhood graph: 5 nearest neighbors in Euclidean distance, **and shifted versions of self and nearest neighbors**
- Output Dimension: 2
- Test set (shown) 1000 “4” and 1000 “9”

Discovering the Viewpoint Manifold

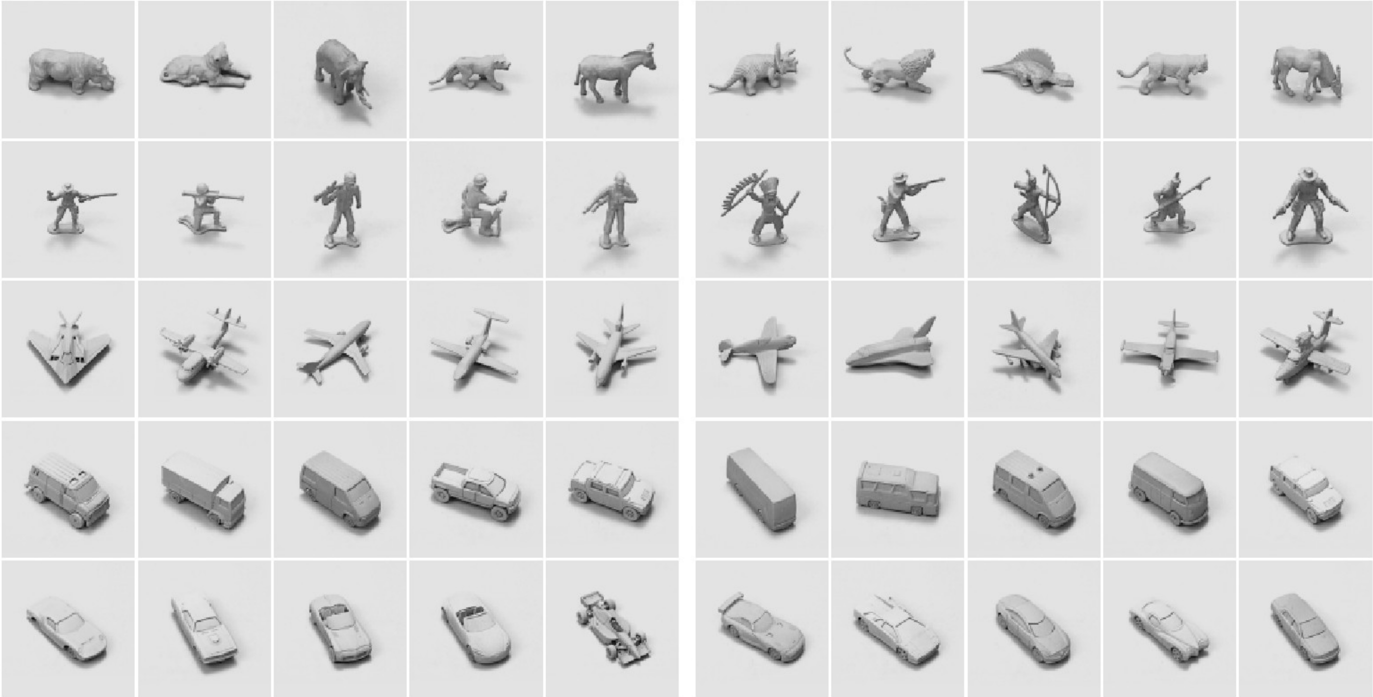
- **Data set:** 927 images of airplanes under 6 illuminations, 18 azimuth and 9 elevations
- **Resolution**: 48x48 pixels
- **Training set** :660 image
- **Test set:** 312 images
- **Architecture:** fully-connected neural net with 20 hidden units and 3 outputs
- **Neighborhood graph:** 1st and 2nd nearest neighbors in azimuth, 1st nearest neighbor in elevation, all illuminations



Generic Object Detection and Recognition with Invariance to Pose and Illumination

- **50** toys belonging to 5 categories: **animal, human figure, airplane, truck, car**
- **10** instance per category: 5 instances used for training, 5 instances for testing
- **Raw dataset: 972** stereo pair of each object instance. **48,600** image pairs total.

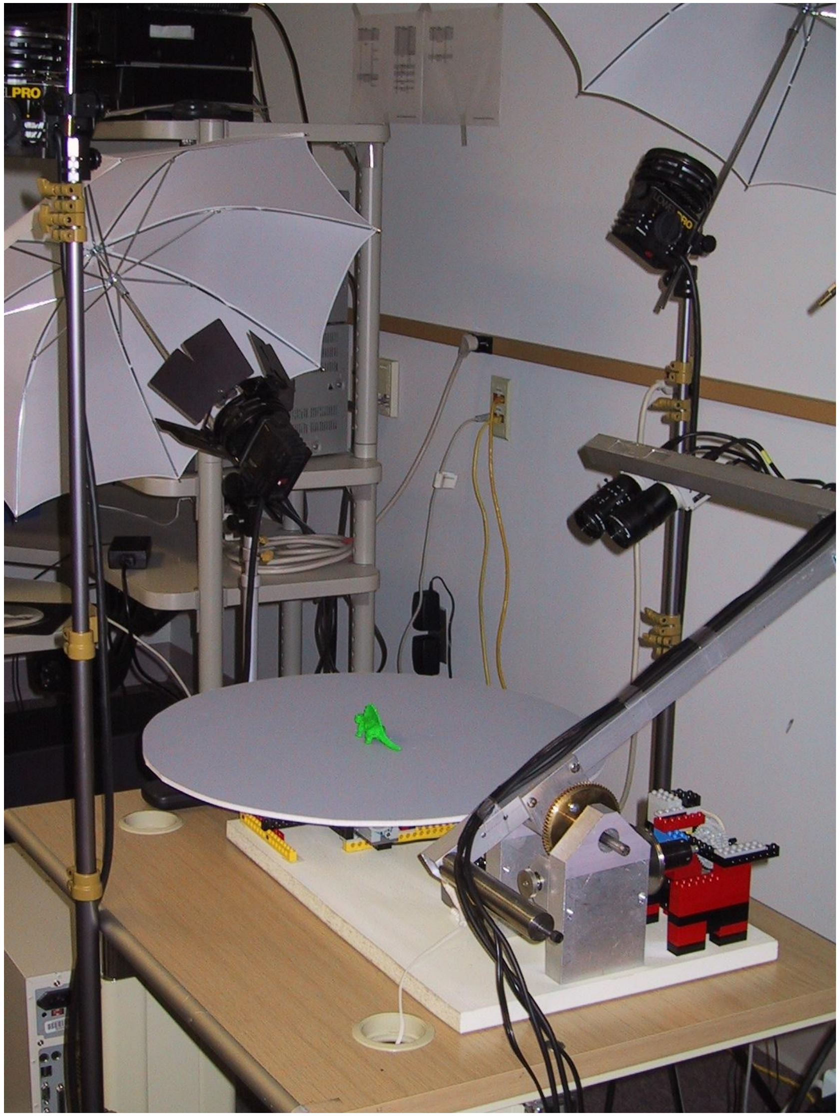
- **For each instance:**
- **18 azimuths**
  - 0 to 350 degrees every 20 degrees
- **9 elevations**
  - 30 to 70 degrees from horizontal every 5 degrees
- **6 illuminations**
  - on/off combinations of 4 lights
- **2 cameras (stereo)**
  - 7.5 cm apart
  - 40 cm from the object



Training instances **Test instances**

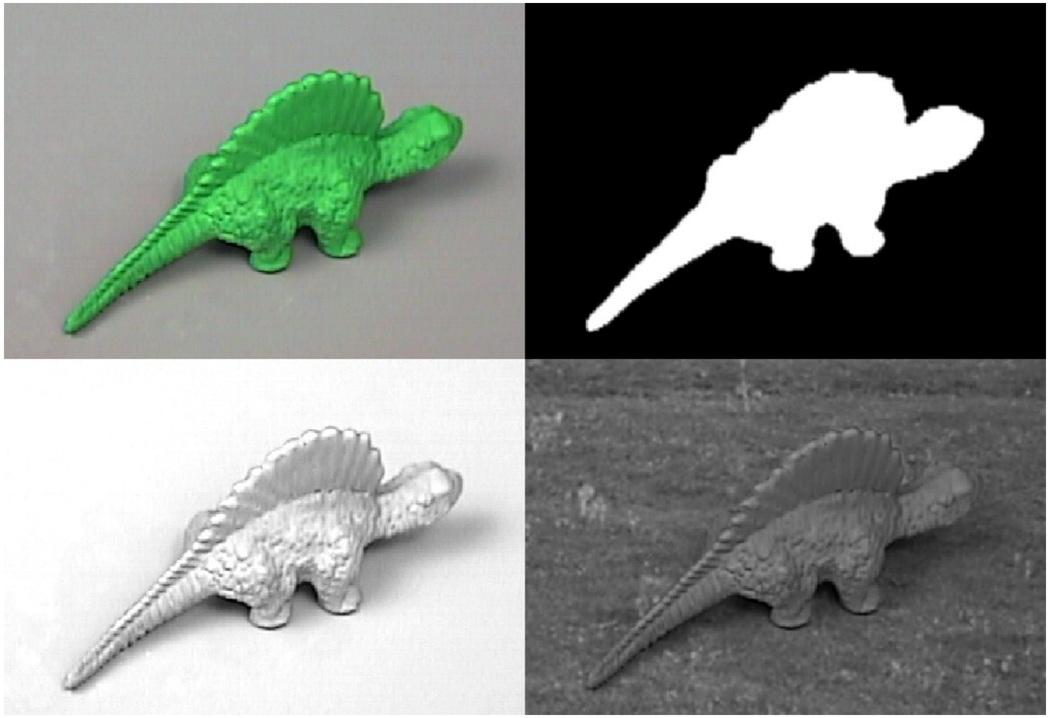
Data Collection, Sample Generation

Image capture setup

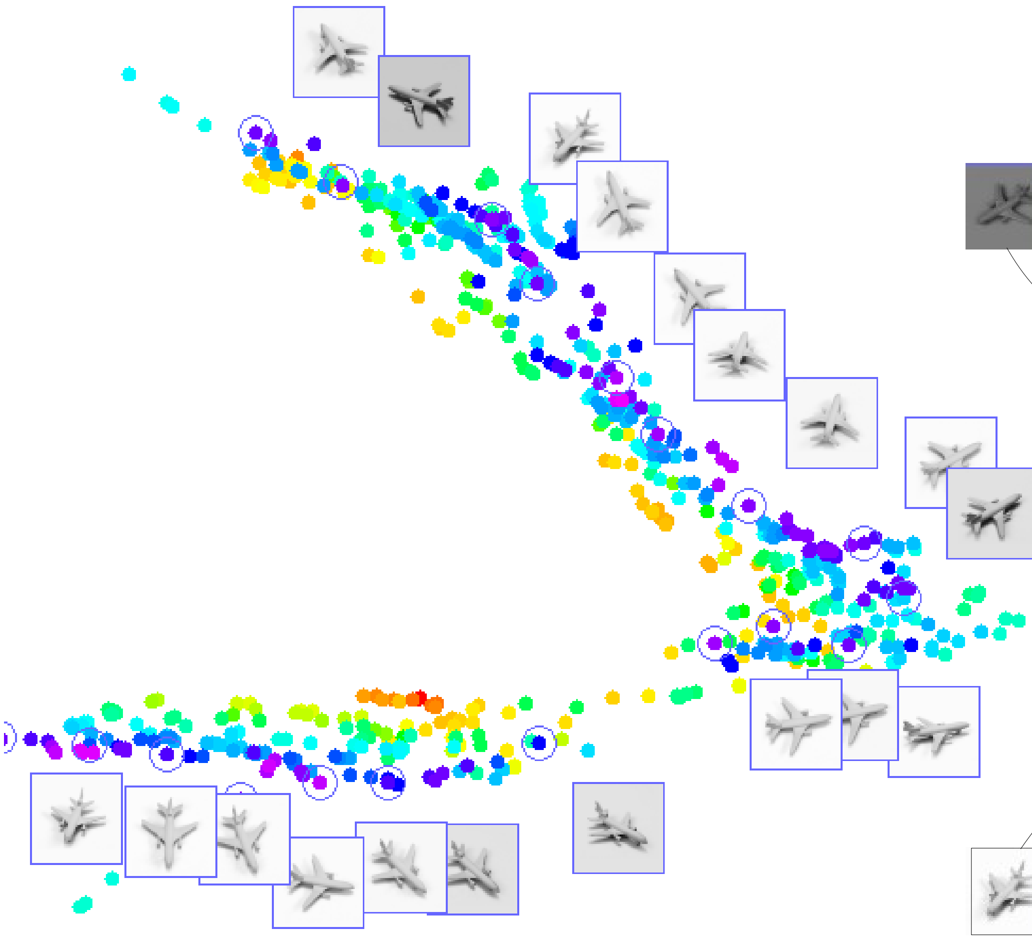


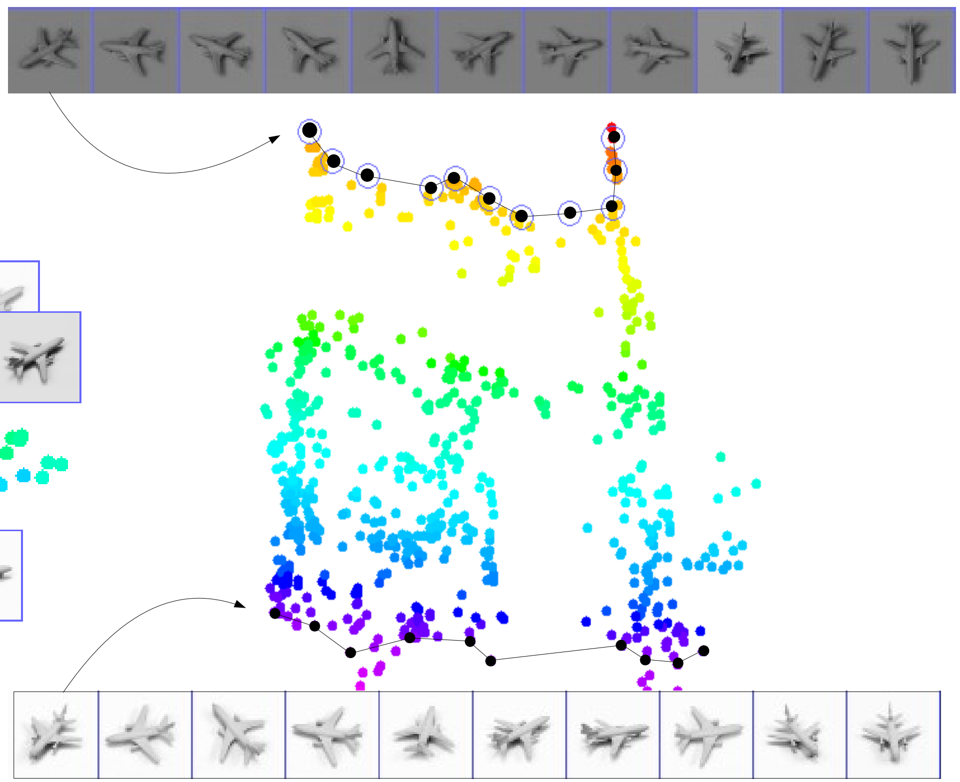
Objects are painted green so that:

- all features other than shape are removed
- objects can be segmented, transformed, and composited onto various backgrounds

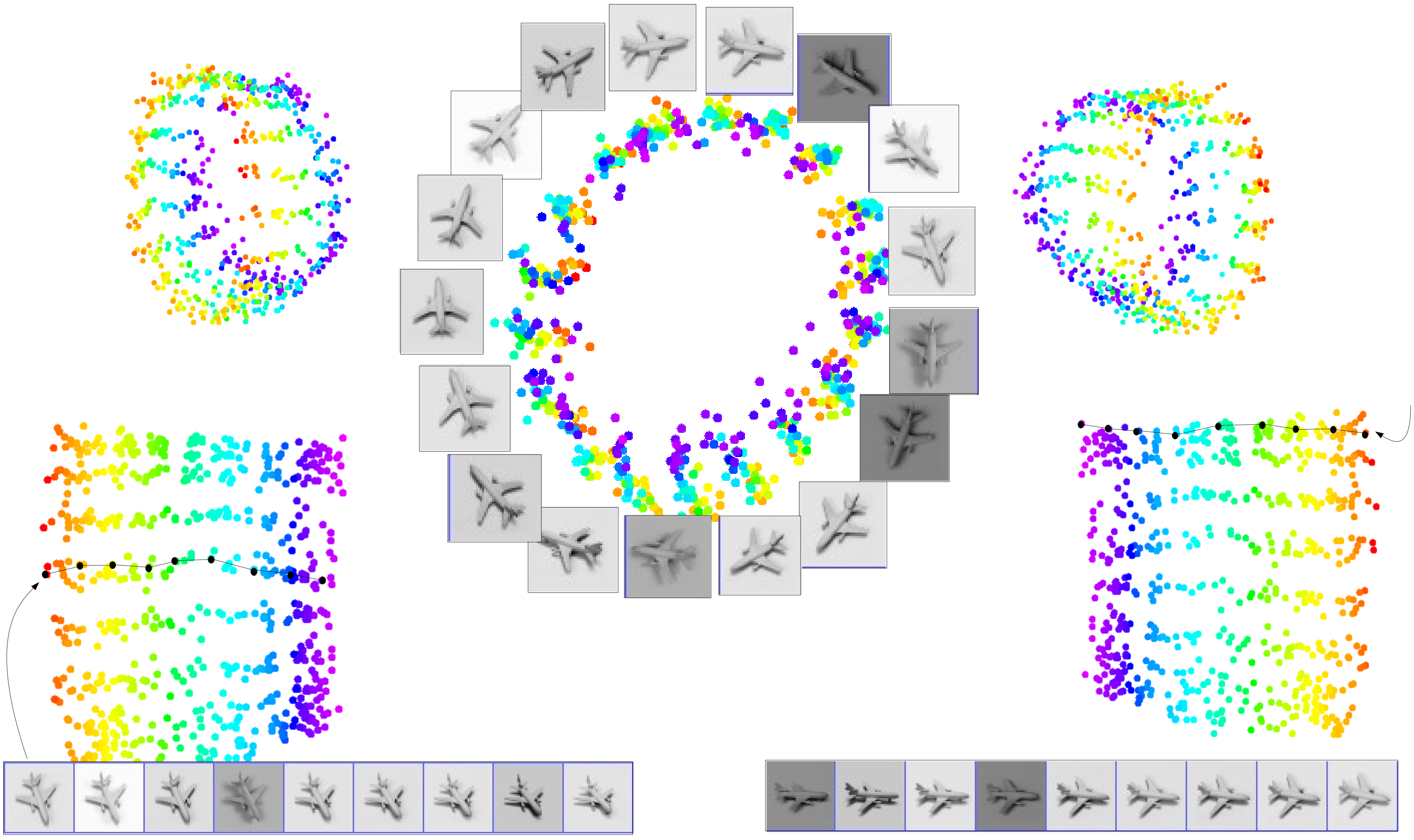


NORB Dataset: LLE

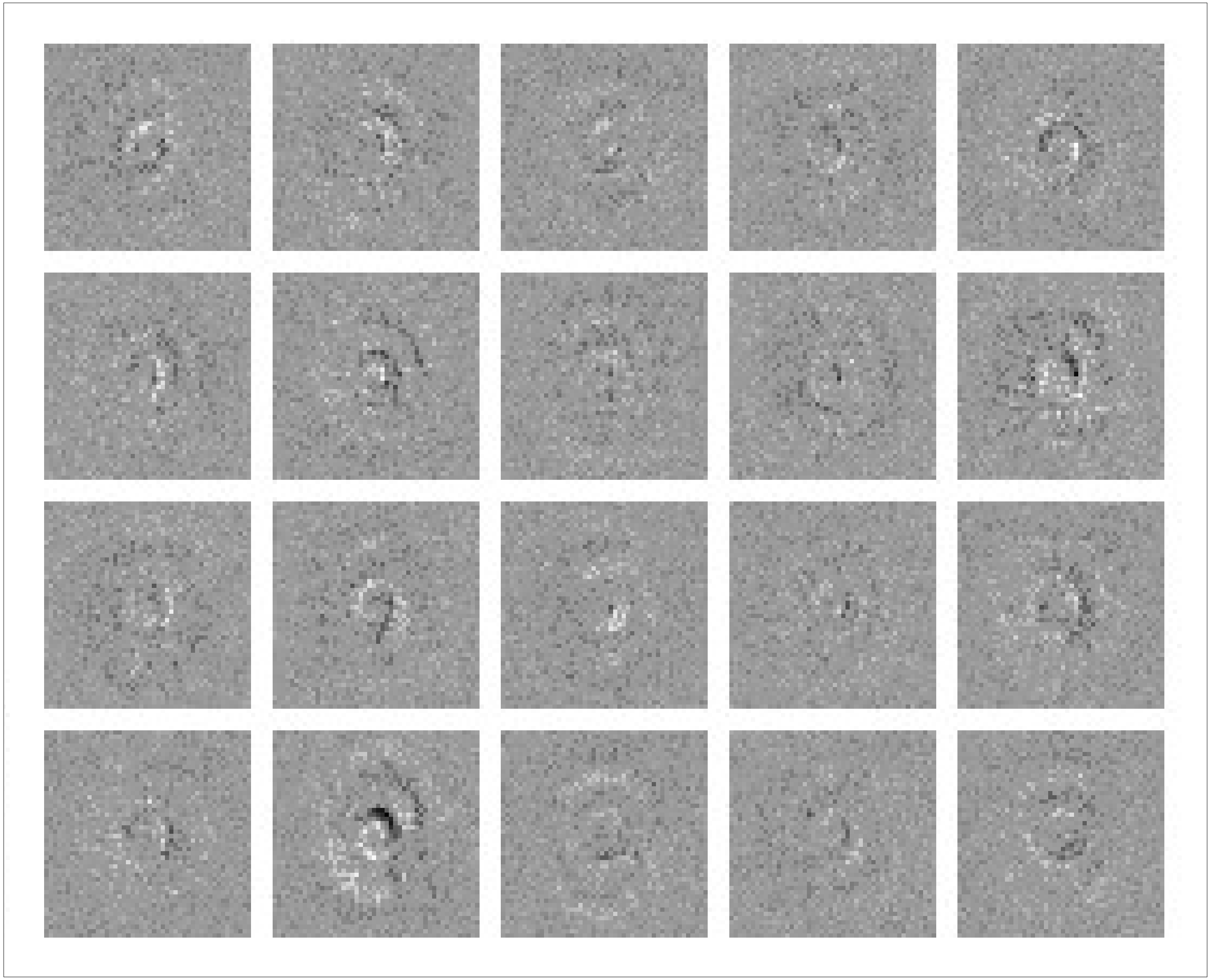




Automatic Discovery of the Viewpoint Manifold with Invariant to Illumination



NORB Dataset: Learned Hidden Units



Thank You