EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Machine Learning in Graphics and Vision

- Latent Variable Models-

SoSe 2018

Hendrik Lensch

Principal Component Analysis

Principal Component Analysis (PCA)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Simplifying (statistical / observed) data sets

- Linear transformation, also called
  - Karhune-Loeve transform (KLT in image processing)
  - Hotelling transform
  - Hauptkomponentenanalyse
  - Empirical orthogonal functions (EOF)

- Transform data set to new coordinate system
  - Basis vectors depend on data set
    - other than e.g. discrete cosine transform
  - Greatest variance by any projection determine first axis
    - „first principal component“
  - Then second orthogonal axis with second largest variance, etc.

- Optimal in L^2 sense, i.e. reconstruction error
  - Computational cost

Principal Component Analysis (PCA)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Pseudo-Algorithm (you cannot compute it this way ;-))
- Assume zero mean data set x
  - subtract average from data set
  - senter of gravity becomes origin
  - Type equation here.
- Find principal component

$$u_1 = \text{argmax}_{\|u\|=1} E(u \cdot x)^2$$ direction of largest variance

- For other components
  - Remove first $k-1$ principal compontents from data

$$x'_{k-1} = x - \sum_{i=1}^{k-1} (u_i \cdot x) u_i$$

  - Find k-th principal component

$$u_k = \text{argmax}_{\|u\|=1} E(u \cdot x'_{k-1})^2$$

Reminder statistics of data matrix X

- Empirical mean, i.e. average

$$\bar{x} = Ex = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Covariance

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$var\ X = cov(X,X)$$

  - variance

$$\sigma = \sqrt{var\ X}$$

  - standard deviation

- Covariance matrix of a 3-dimensional data set

$$C := \begin{pmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) \end{pmatrix}$$

Principal Component Analysis (PCA)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Equivalent to singular value decomposition (SVD)
- Singular value decompositoin of data matrix X

$$X = USV^T$$

- Looking at the covariance matrix $C \coloneqq XX^T = US^2U^T$ reveals:
  - SVD yields eigenvalue decomposition of covariance matrix
  - diagonal matrix S^2 contains eigenvalues
  - eigenvectors with largest eigenvalues correspond to dimension with strongest correlation

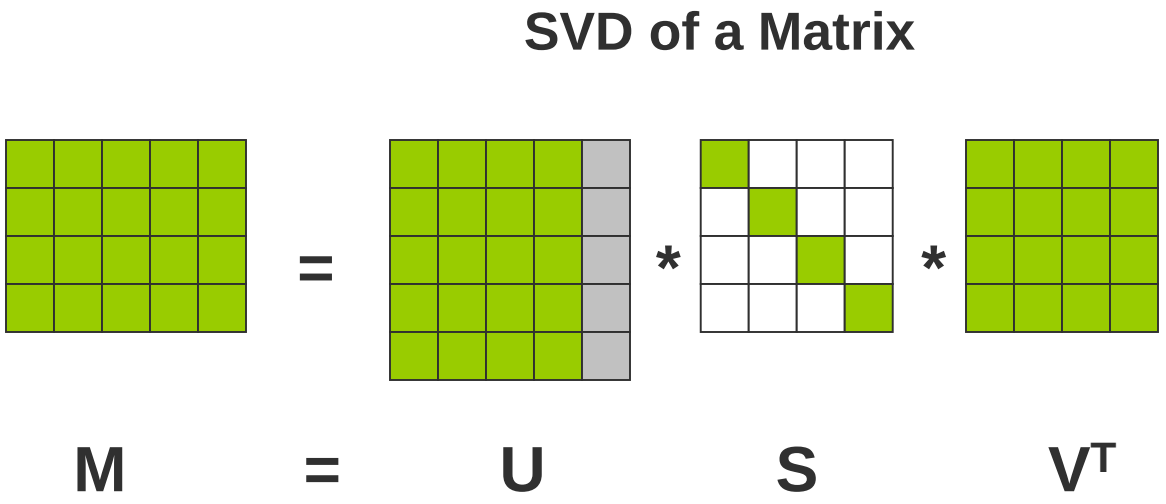
- Reduced-space data matrix

$$Y = U_n^T X = S_n V_n^T$$

by projection on the first n singular vectors U_n

Singular Value Decomposition (SVD)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



U and V are orthogonal matrices, and S is a diagonal matrix consisting of singular values.

Singular Value Decomposition (SVD)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

SVD of a Matrix: observations

$$M = U S V^T$$

Multiply both sides by M^T

Multiplying on the left

$$M^TM = (U S V^T)^T U S V^T$$

$$M^TM = (V S U^T) U S V^T$$

$$U^TU = I$$

$$M^TM = V S^2 V^T$$

Multiplying on the right

$$MM^T = U S V^T (U S V^T)^T$$

$$MM^T = U S V^T (V S U^T)$$

$$V^TV = I$$

$$MM^T = U S^2 U^T$$

Singular Value Decomposition (SVD)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

SVD of a Matrix: observations

$M^TM = V S^2 V^T$ ⟵ diagonalizations ⟶ $MM^T = U S^2 U^T$

Diagonalization of a Matrix: (finding eigenvalues)

$A = W \Lambda W^T$

where:
A is a square, symmetric matrix
Columns of W are eigenvectors of A
Λ is a diagonal matrix containing the eigenvalues

Therefore, if we know U (or V) and S, we basically have found out the eigenvectors and eigenvalues of MM^T (or M^TM) !

Principal Component Analysis (PCA)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Analysis of n-dimensional data
- Observes correspondence between different dimensions
- Determines principal dimensions along which the variance of the data is high

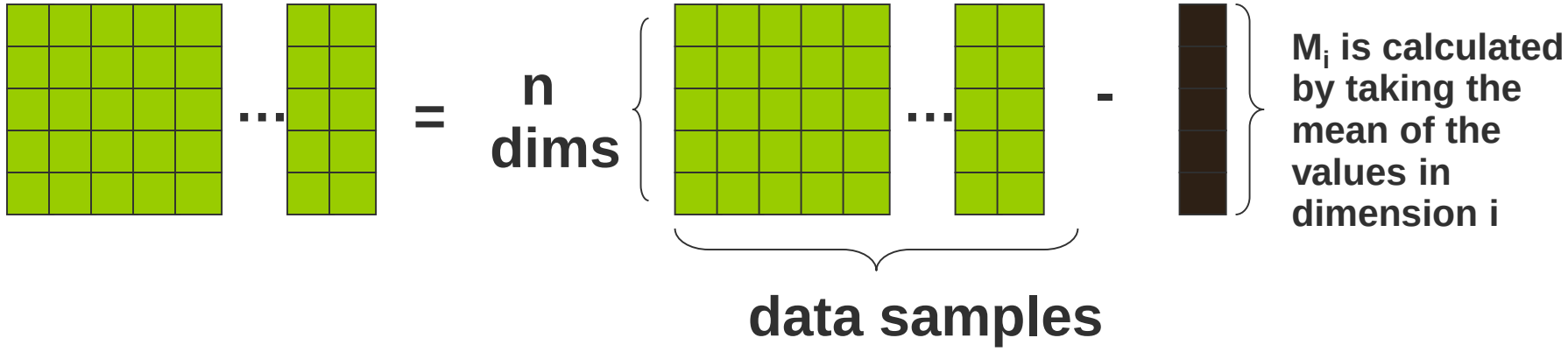
Why PCA?

- Determines a (lower dimensional) basis to represent the data
- Useful compression mechanism
- Useful for decreasing dimensionality of high dimensional data

Principal Component Analysis (PCA)

Steps in PCA: #1 Calculate Adjusted Data Set

Adjusted Data Set: A **Data Set: D** **Mean values: M**

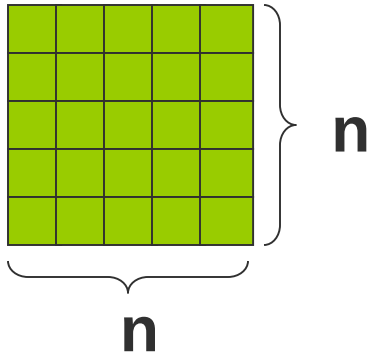


M_i is calculated by taking the mean of the values in dimension i

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Steps in PCA: #2 Calculate Co-variance matrix, C, from Adjusted Data Set, A

Co-variance Matrix: C



$C_{ij} = cov(i,j)$

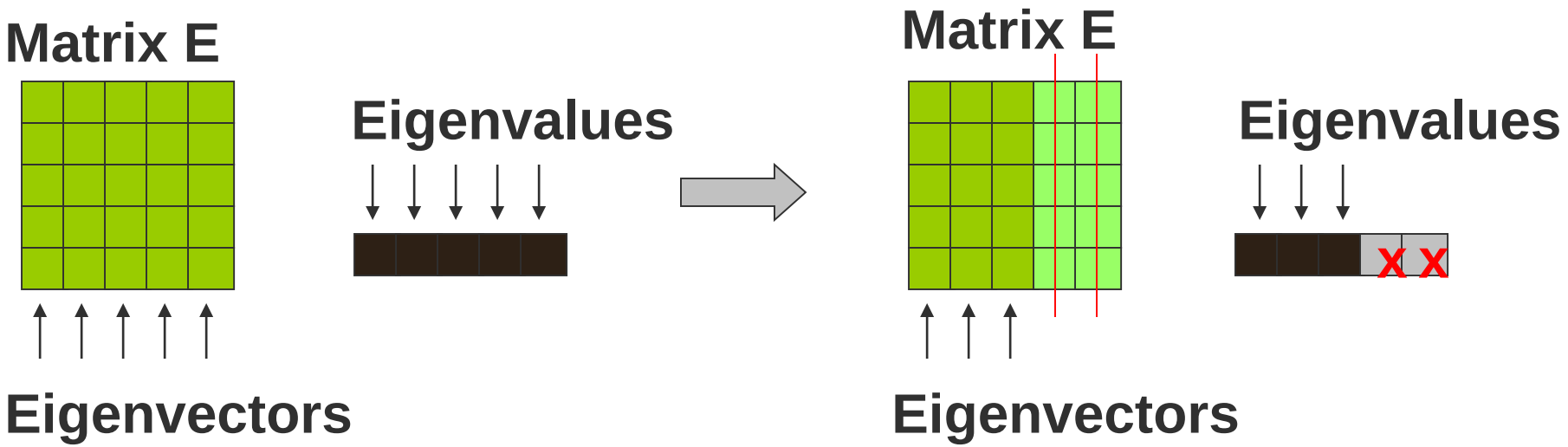
$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

Note: Since the means of the dimensions in the adjusted data set A are 0, the covariance matrix can simply be written as:

$$C = (A\,A^T) / (n-1)$$

Principal Component Analysis (PCA)

Steps in PCA: #3 Calculate eigenvectors and eigenvalues of C



If some eigenvalues are 0 or very small, we can essentially discard those eigenvalues and the corresponding eigenvectors, hence reducing the dimensionality of the new basis.

Steps in PCA: #4 Transforming data set to the new basis

$$F = E^T A$$

where:
F is the transformed data set
E^T is the transpose of the E matrix containing the eigenvectors
A is the adjusted data set

Note that the dimensions of the new dataset, F, are less than the data set A

To recover A from F:

$(E^T)^{-1}F = (E^T)^{-1}E^T A$

$(E^T)^T F = A$

$EF = A$

* E is orthogonal, therefore $E^{-1} = E^T$

PCA using SVD

Recall: In PCA we basically try to find eigenvalues and eigenvectors of the covariance matrix, C. We showed that $C = (AA^T) / (n-1)$, and thus finding the eigenvalues and eigenvectors of C is the same as finding the eigenvalues and eigenvectors of AA^T

Recall: In SVD, we decomposed a matrix A as follows:

$$A = U S V^T$$

and we showed that:

$$AA^T = U S^2 U^T$$

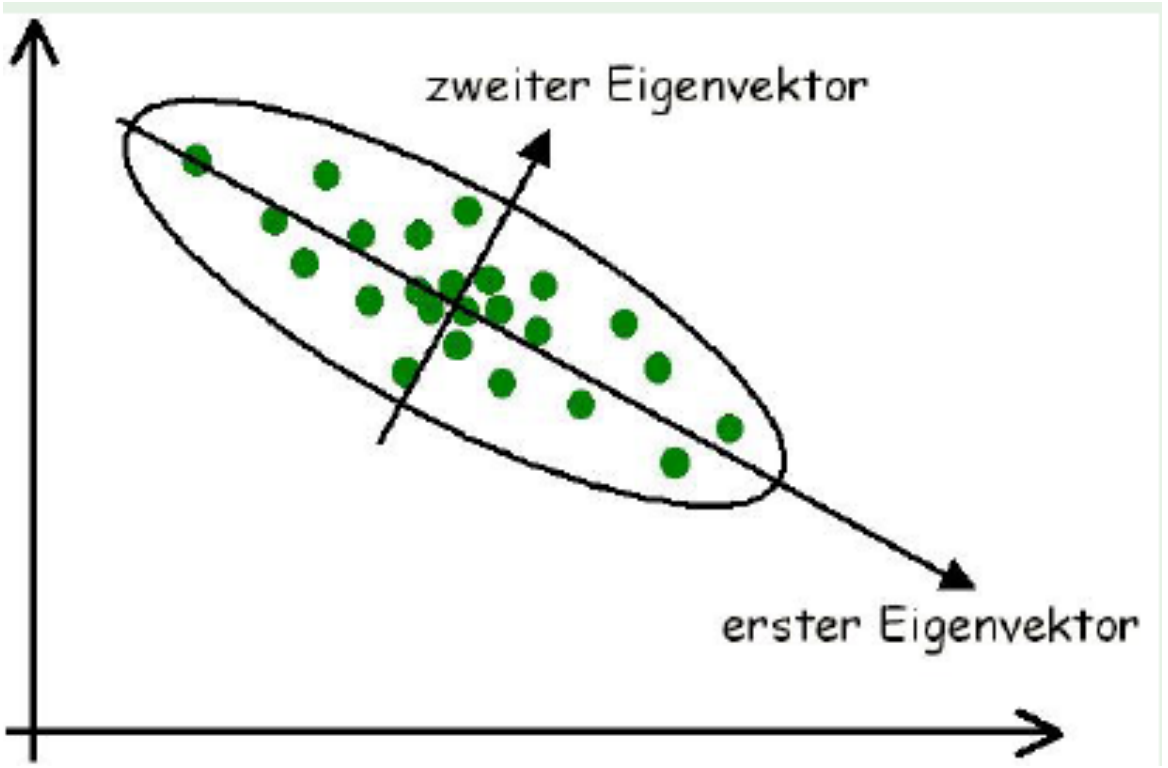
where the columns of U contain the eigenvectors of AA^T and the eigenvalues of AA^T are the squares of the singular values in S

Thus SVD gives us the eigenvectors and eigenvalues that we need for PCA

PCA - examples

- Can be used for dimensionality reduction
  - Keep most important aspects of data
  - Take the first n axes and project

- Examples:
  - inertia tensor
  - bounding box orientation for point clouds
  - BRDFs
  - faces
  - …



Eigenfaces – PCA on 2D face images

- Training images

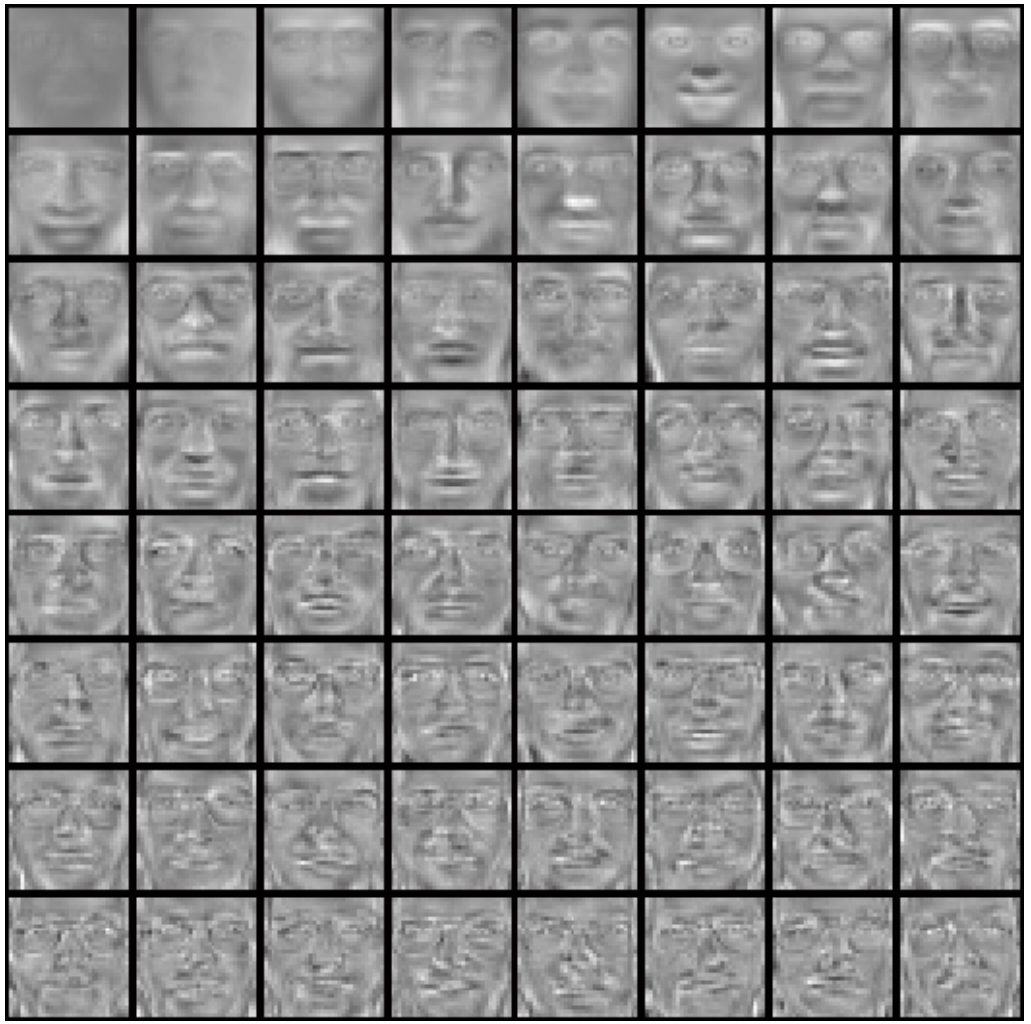
$$x_1, \dots, x_N$$

Eigenfaces – PCA on 2D face images

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mean face μ

Top eigenvectors $u_1, \dots, u_k$



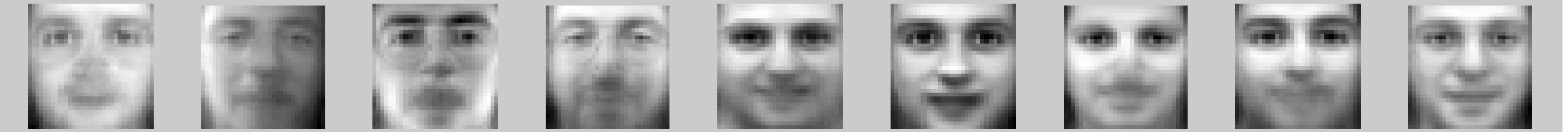
Eigenfaces – PCA on 2D face images

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Principal component (eigenvector) u_k

$\mu + 3\sigma_k u_k$

$\mu - 3\sigma_k u_k$

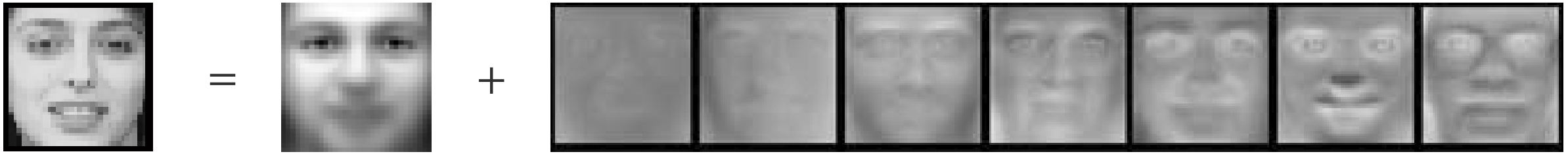


Eigenface example

- Face x in eigenspace / „face space“ coordinates:

$$x \rightarrow \left[u_1^T(x-\mu), \ldots, u_k^T(x-\mu)\right] = [\mathrm{w}_1, \ldots, \mathrm{w}_\mathrm{k}]$$

- Reconstruction:

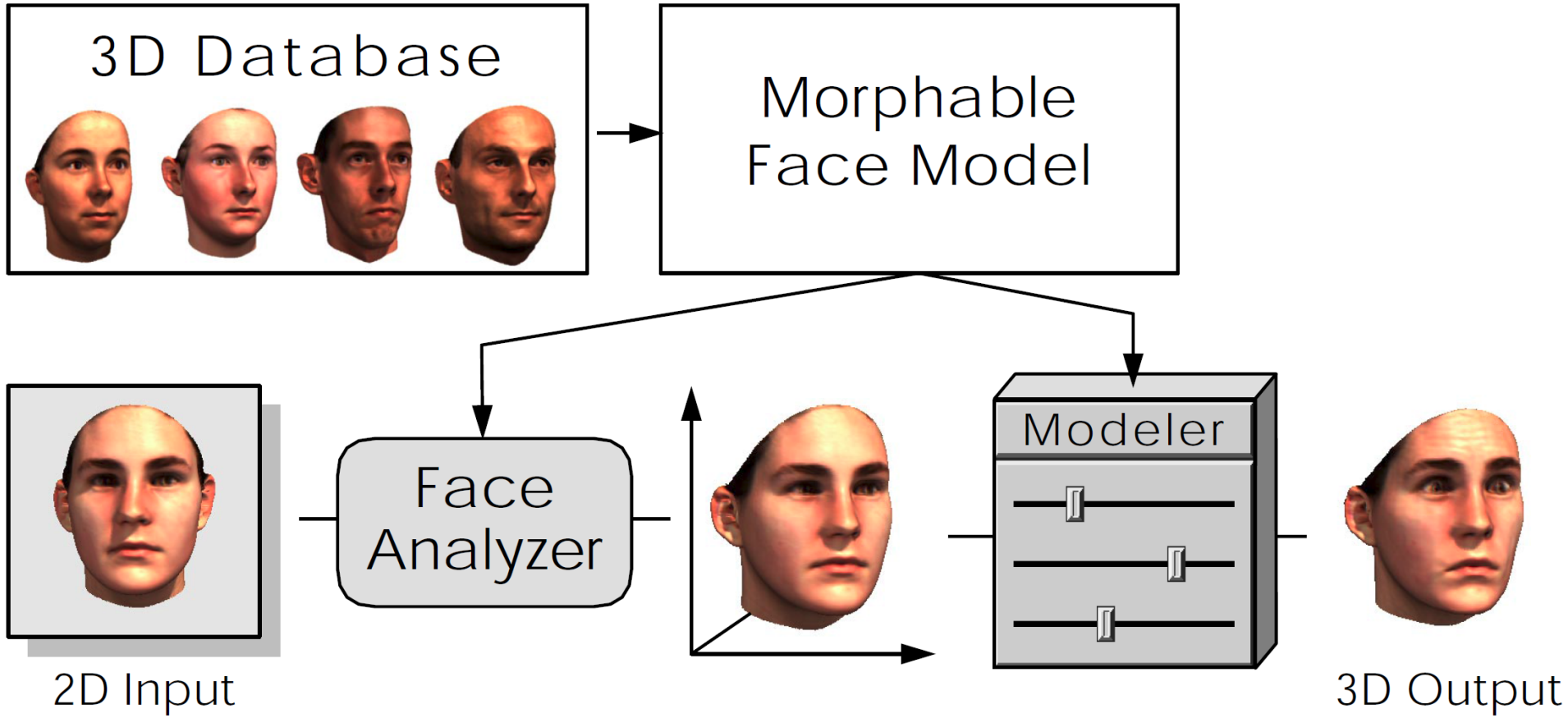


$$\hat{x} \quad = \quad \mu \quad + \quad w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + \ldots \quad + w_ku_k$$

PCA-Example: Morphable Face Model

- [Blanz&Vetter, 1999, A Morphable Model for the Synthesis of 3D Faces]

- Simple way to synthesize faces

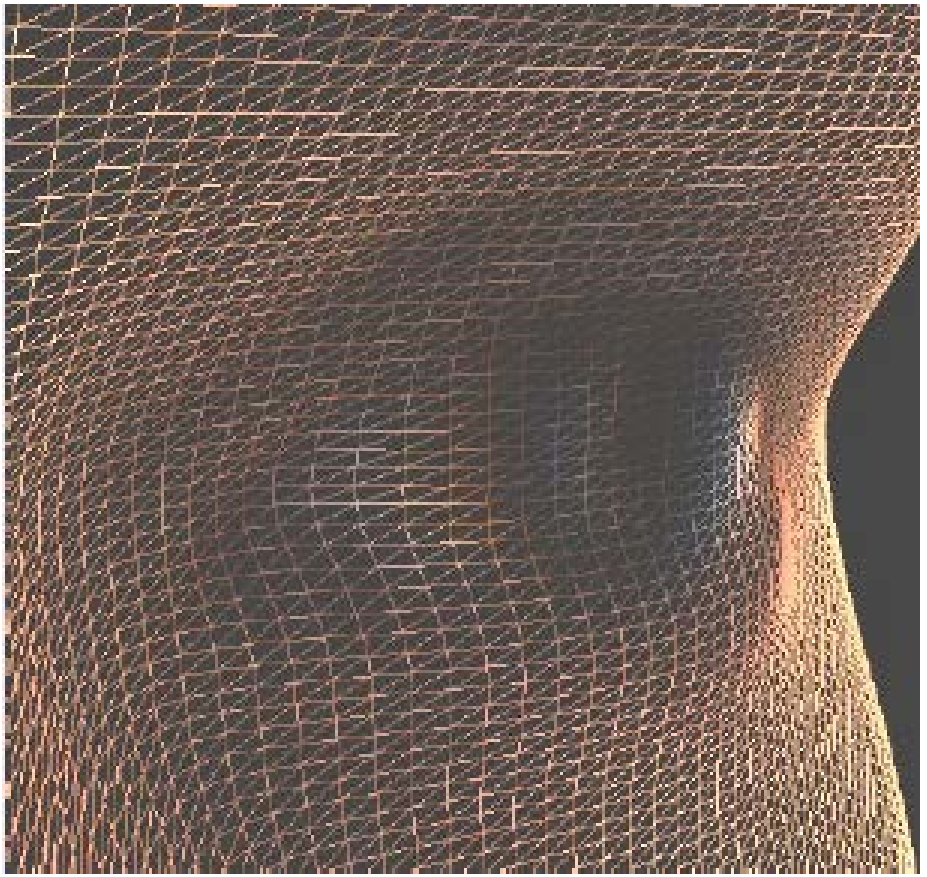


Capturing a Face Database

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Cyberware scan of 100 females and 100 males
- Representation $I(h,\phi) = \big(R(h,\phi), G(h,\phi), B(h,\phi), d(h,\phi)\big)$ at 512×512

- Automatic processing: align scale and position, cut, etc.
  - Shape vector $S_i = \big(x_{i,1}\ y_{i,1}\ z_{i,1}\ \dots x_{i,70000}\ y_{i,70000}\ z_{i,70000}\big)^T$
  - texture vector $T_i = \big(r_{i,1}\ g_{i,1}\ b_{i,1}\ \dots r_{i,70000}\ g_{i,70000}\ b_{i,70000}\big)^T$



Establishing Full Correspondence Between All Faces

- Morphing requires correspondence, i.e. cross-dissolve insufficient due to different shape

- Compute optical flow $\left(\Delta h(h,\phi), \Delta\phi(h,\phi)\right)$ by minimizing

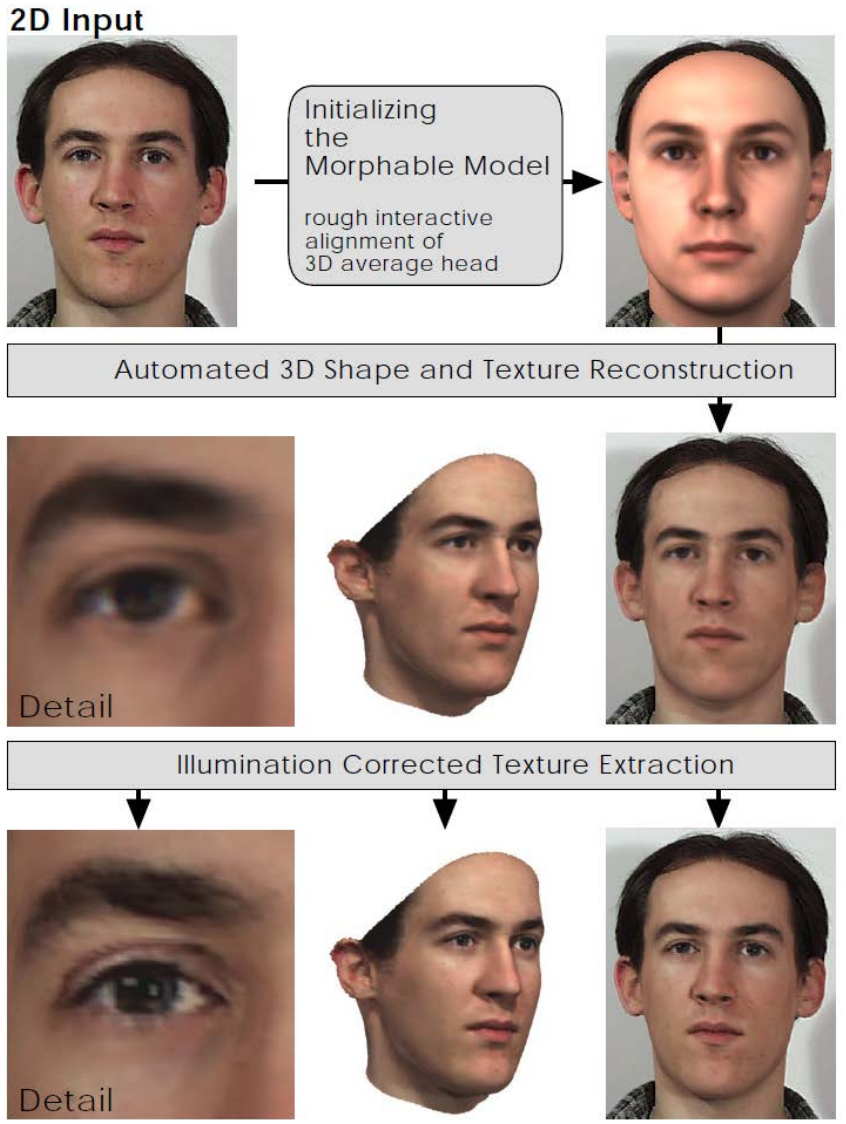
$$\left\| I_1(h,\phi) - I_2\left(h + \Delta h(h,\phi), \phi + \Delta\phi(h,\phi)\right)\right\|_2$$

- For two faces $I_1(h,\phi)$ and $I_2(h,\phi)$
  - Problem on forehead and cheek due to lack of detail
    - fill these regions by interpolation
  - Can add further features like mean curvature etc.

- Morphing I_2 to I_1 by flow field interpolation

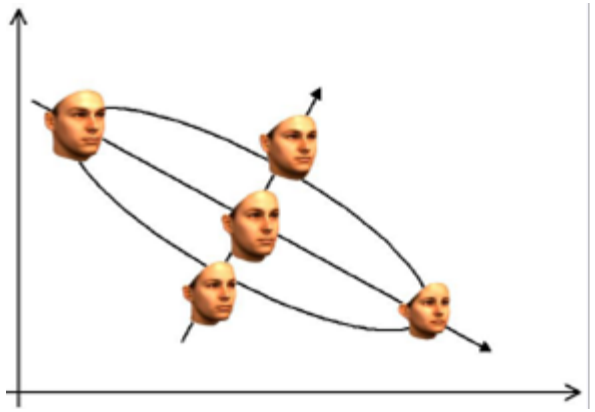
$$I_2\left(h + \alpha\Delta h(h,\phi), \phi + \alpha\Delta\phi(h,\phi)\right) \qquad \alpha \in [0,1]$$

Complete Matching Pipeline

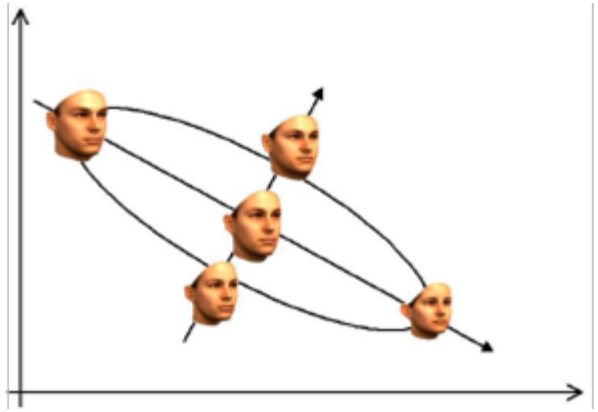


Latent Face Model

- Find facial characteristics by PCA
- Compute covariance matrices C_S and C_T over



shape differences $\Delta S_i = S_i - \bar{S}$



texture differences $\Delta T_i = T_i - \bar{T}$

- Yielding the model of face space

$$S_m = \bar{S} + \sum_{i=1}^{m-1} \alpha_i s_i \quad \text{and} \quad T_m = \bar{T} + \sum_{i=1}^{m-1} \beta_i t_i$$

with eigenvectors $s_\mathrm{i}, \mathrm{t_i}$ in descening order according to their eigenvalues

- Generative model: modify average face

Generative Model

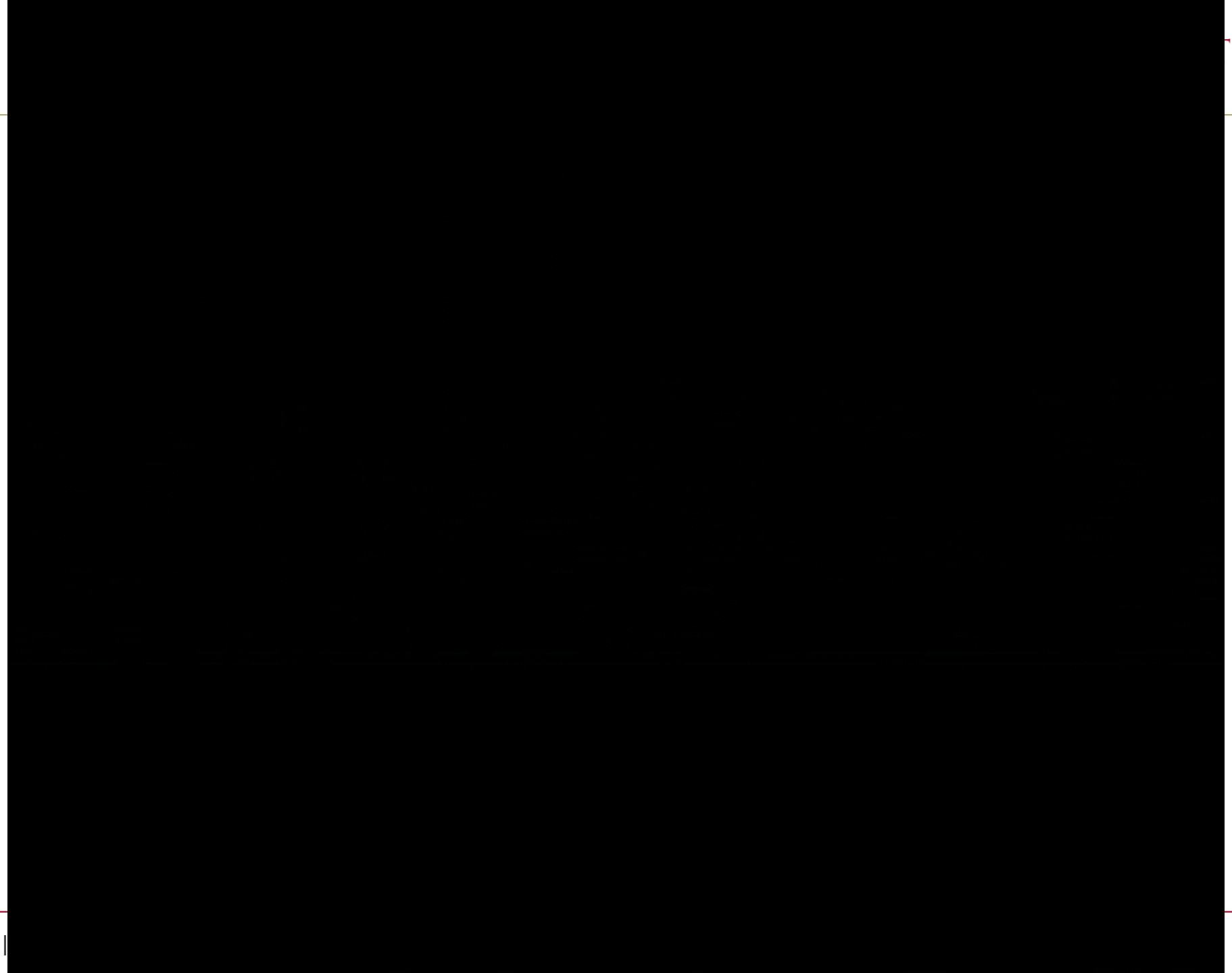
- Produce probable faces
- Assume normal distributed faces in the data base yield pdf

$$p(\alpha) \sim e^{-\frac{1}{2}\sum_{i=1}^{m-1}\left(\frac{\alpha_i}{\sigma_{S,i}}\right)^2}$$

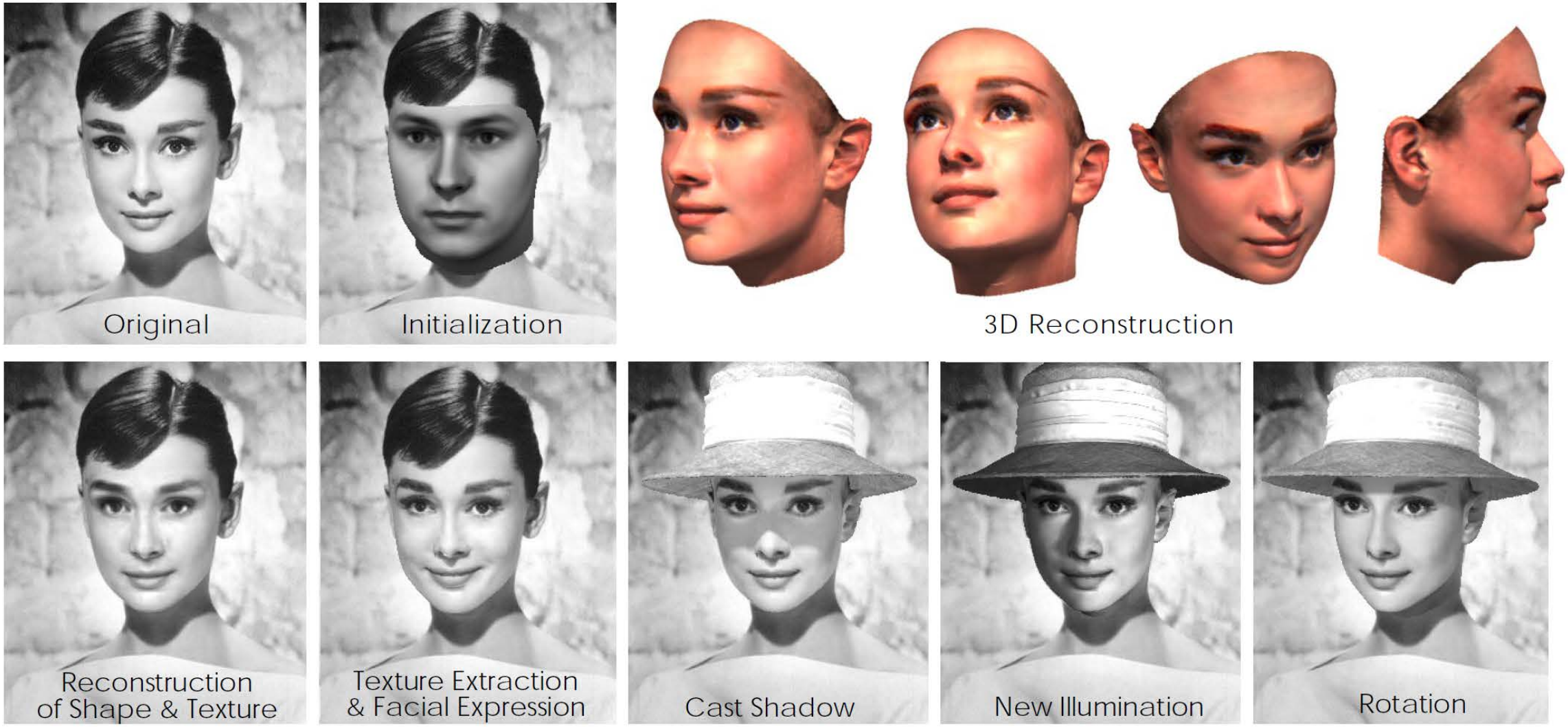
$$p(\beta) \sim e^{-\frac{1}{2}\sum_{i=1}^{m-1}\left(\frac{\beta_i}{\sigma_{T,i}}\right)^2}$$

- Simulating this density yields random „probable faces“

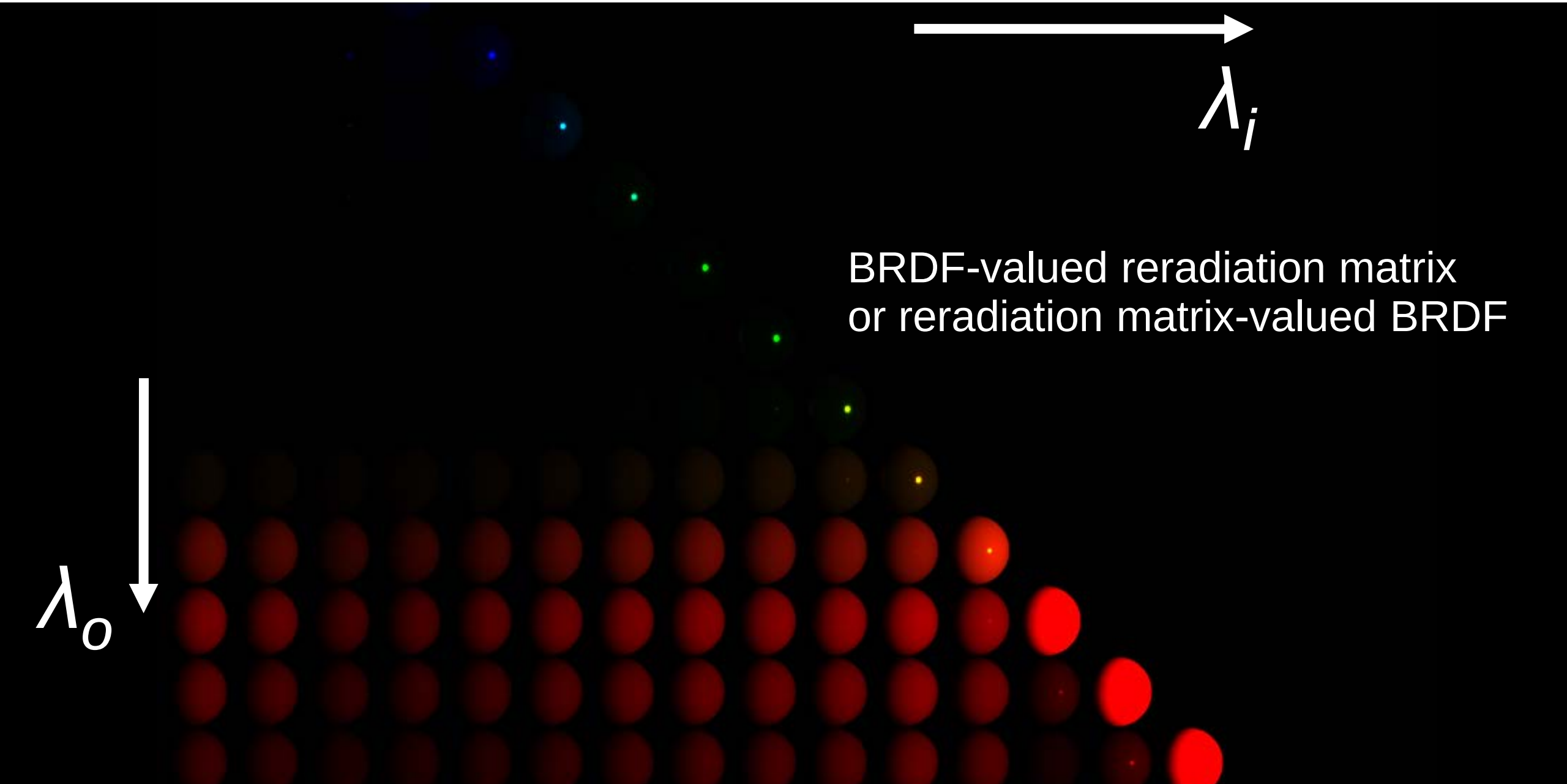
Video



Results


Original
Initialization
3D Reconstruction
Reconstruction of Shape & Texture
Texture Extraction & Facial Expression
Cast Shadow
New Illumination
Rotation

Example: bispectral BRRDF



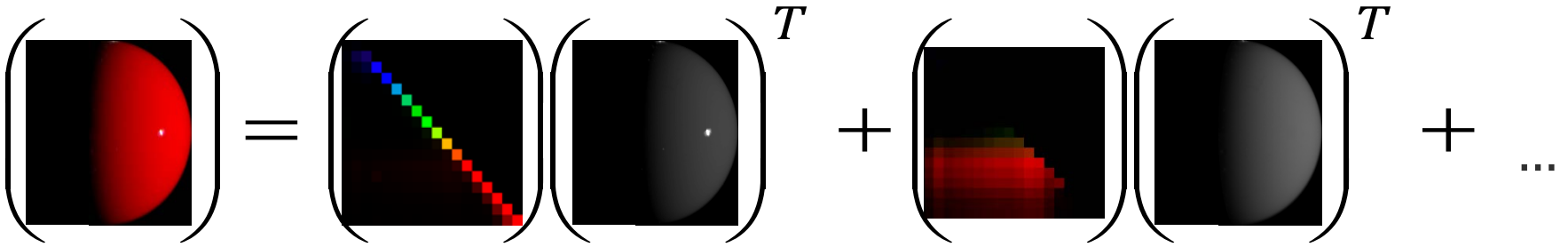
BRDF-valued reradiation matrix
or reradiation matrix-valued BRDF

Example: PCA on Bispectral BRRDFs

- Representing the bispectral BRRDF as a series of separable functions using SVD

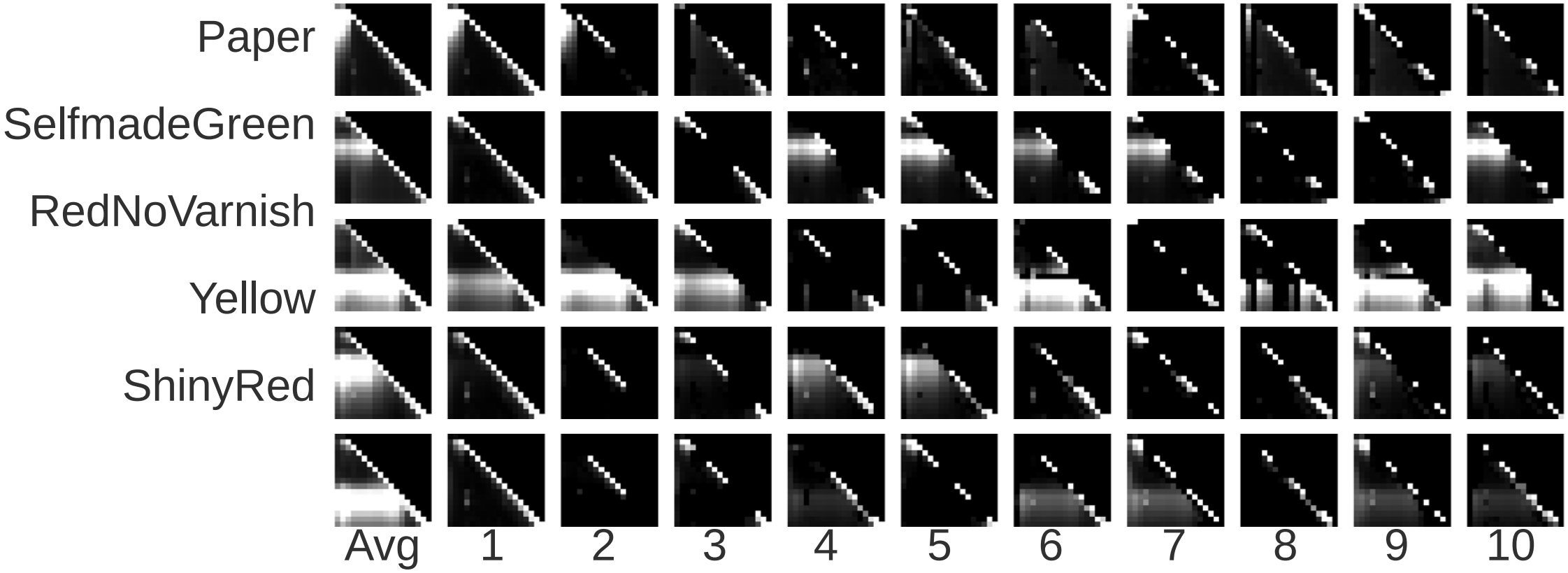
$$f_r(\omega_o, \lambda_o, \omega_i, \lambda_i) = \sum_k f_k^{\lambda}(\lambda_o, \lambda_i) f_k^{\omega}(\omega_o, \omega_i)$$

- Fluorescence carries some weak angular depencency,
  i.e. a low rank decomposition is possible.



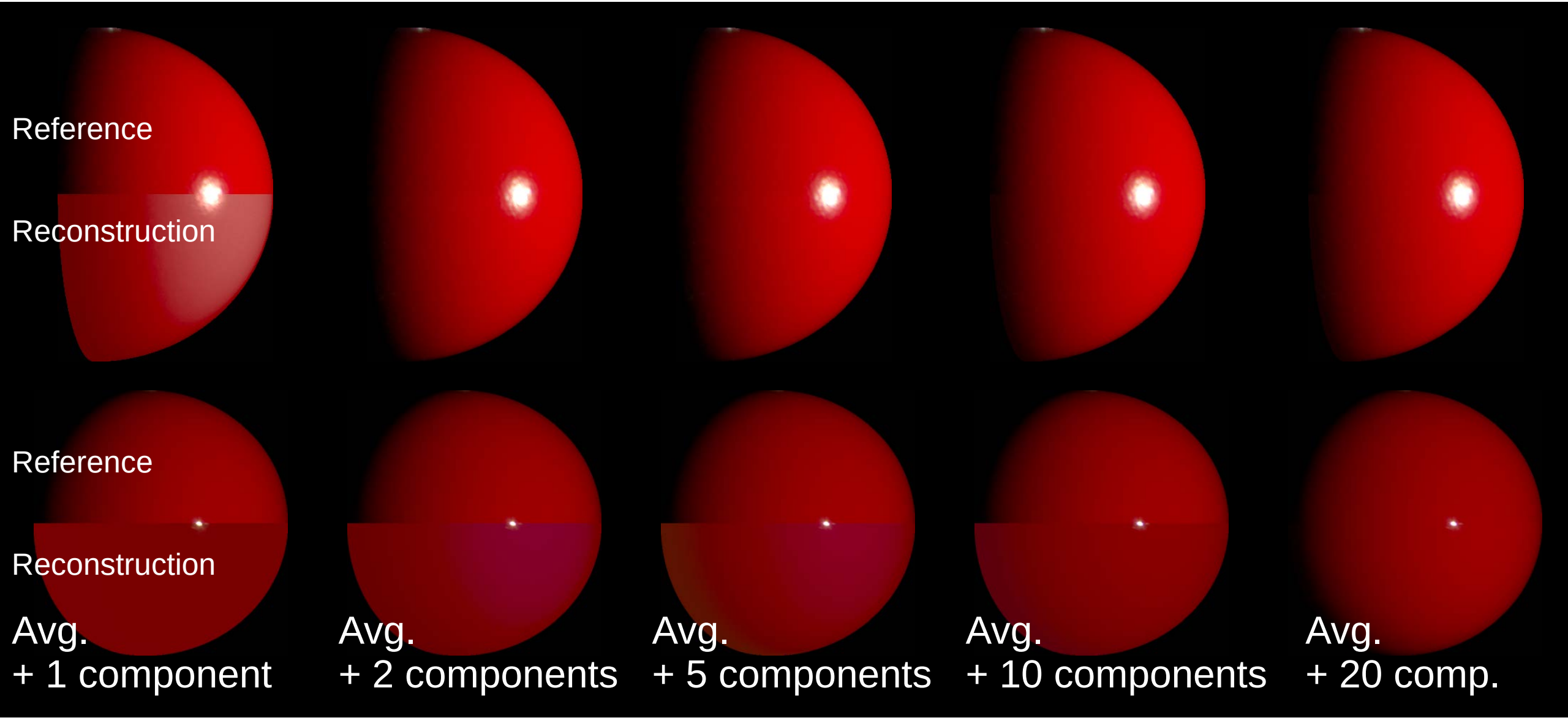
Bispectral basis vectors

- Common bispectral basis B for all angular samples



Reconstruction from principal components

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN



Vector Quantization

Vector Quantization

- Input: set of vectors

$$Y = \left\{ y_i \right\}_{i=1}^{N}$$

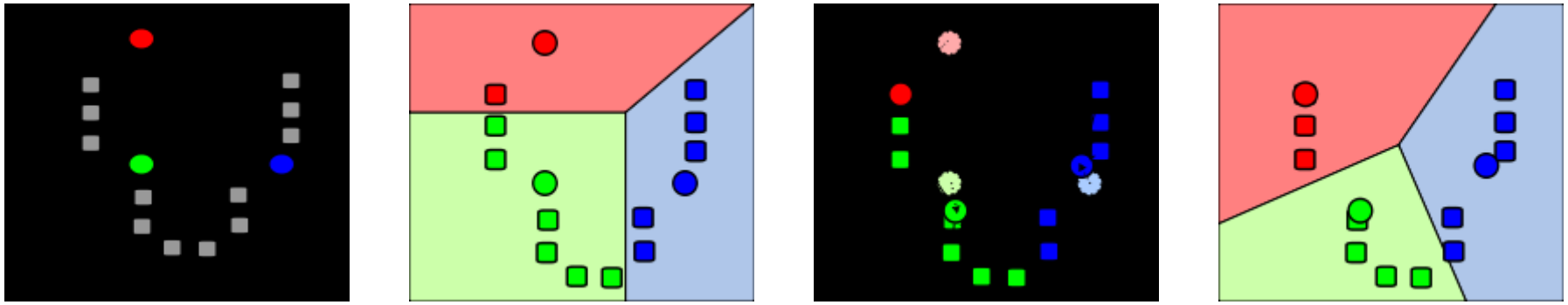
- Output : Codebook of K<<N vectors
- Each input vector is represented by its nearest neighbor

- Find code book by Lloyd iteration / K-Means clustering

$$\min_{\mathbf{C},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{C}\mathbf{X}\|_F^2 \right\} \quad \text{s.t.} \quad \forall i,\ \mathbf{x}_i = \mathbf{e}_k \text{ for some } k.$$

Lloyd Iteration / K-Means-Clustering

[from Wikipedia]



1. Distribute K random samples
2. Assign samples to nearest cluster representative (corresponding to inclusion in Voronoi cell)
3. Update cluster center by computing the mean of the cluster samples
4. Repeat until convergence

K-SVD

- Similar to Vector Quantization but allow for linear combinations of at max T_0 code vectors ($K > T_0$)

- much better approximation of original samples
  - Small (T_0) number of coefficients per sample compared to PCA
  - Optimized code book for different sample regions

- Objective function:

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \leq T_0.$$

- Sparse initialization of X by any pursuit method

Orthogonal Matching Pursuit (OMP)

… to obtain a good sparse projection

Extend reconstructed basis I one by one

1. Select best coefficient j with j-th column a_j of A

$$\arg\max_j |< r, a_j >| \qquad r = \tilde{x} - Sx = \tilde{x} - A\hat{x}$$

i.e. j with the largest effect on residual.

2. Assemble matrix A_I with the so-far selected columns I
3. Reconstruct

$$\hat{x} = A_I^* \tilde{x}$$

4. Repeat until

$$r = 0$$

K-SVD Algorithm

Initialization : Set the random normalized dictionary matrix $\mathbf{D}^{(0)} \in \mathbb{R}^{n \times K}$. Set $J = 1$.
Repeat until convergence,
Sparse Coding Stage: Use any pursuit algorithm to compute $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$

$$\min_{\mathbf{x}} \left\{ \|\mathbf{y}_i - \mathbf{D}\mathbf{x}\|_2^2 \right\} \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq T_0.$$

Codebook Update Stage: For $k = 1, 2, \ldots, K$

- Define the group of examples that use $\mathbf{d}_k$, $\omega_k = \{i | \, 1 \leq i \leq N, \, \mathbf{x}_i(k) \neq 0\}$.
- Compute

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}^j,$$

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to those elements that initially used $\mathbf{d}_k$ in their representation, and obtain $\mathbf{E}_k^R$.
- Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$. Update: $\mathbf{d}_k = \mathbf{u}_1$, $\mathbf{x}_R^k = \mathbf{\Delta}(1,1) \cdot \mathbf{v}_1$

Set $J = J + 1$.

The K-SVD Algorithm

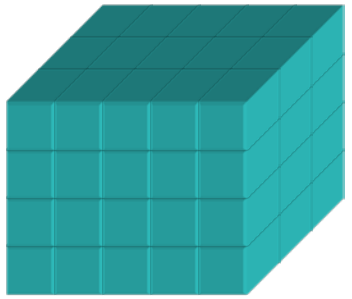
Higher-Dimensional SVD

Tensor: Generalization of an n-dimensional array

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

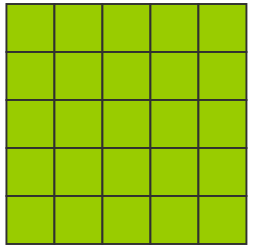
[slides from Musawir Ali]

Vector: order-1 tensor



Order-3 tensor

Matrix: order-2 tensor

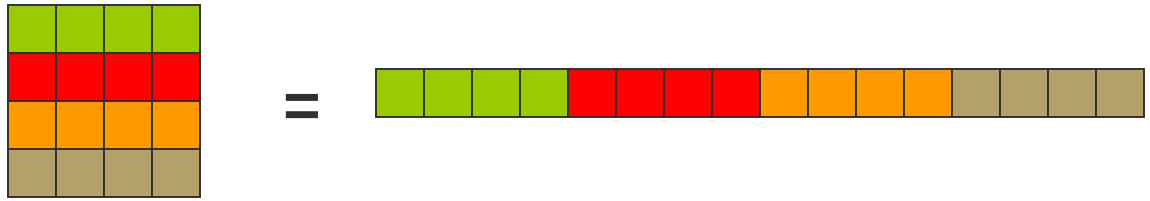


Tensor Flattening

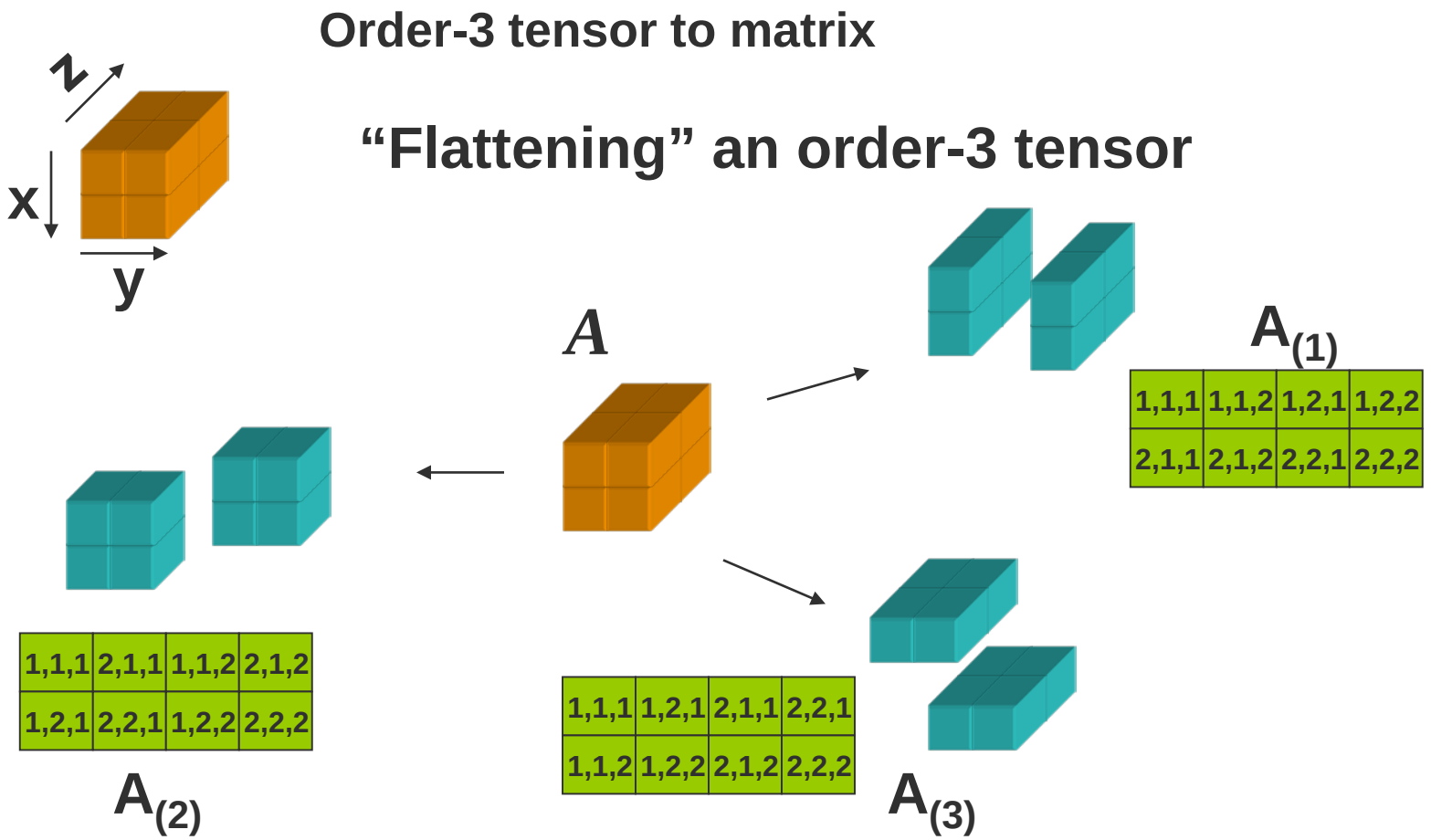
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Matrix to vector

“Vectorizing” a matrix



Reshaping Tensors



n-Mode Multiplication

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Multiplying Matrices and order-3 tensors

$$\mathcal{A} \times_n M = M A_{(n)}$$

Multiplying a tensor $\mathcal{A}$ with matrix M

$\mathcal{A}$ is flattened over dimension n (permute dimensions so that the dimension n is along the columns, and then flatten), and then regular matrix multiplication of matrix M and flattened tensor $A_{(n)}$ is performed

N-Mode SVD (HOSVD/Tucker Transform)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Generalization of SVD

Order-2 SVD: $\longrightarrow$ $D = U\ S\ V^T$

Written in terms of mode-n product:

By definition of mode-n multiplication:

$$D = S \times_1 U \times_2 V$$
$$D = U\ (\ V\ S_{(2)}\ )_{(1)}$$
$$D = U\ (\ V\ S\ )^T$$
$$D = U\ S^T\ V^T$$
$$D = U\ S\ V^T$$

Note: $S^T = S$ since S is a diagonal matrix

Note: $D = S \times_1 U \times_2 V = S \times_2 V \times_1 U$

N-Mode SVD (HOSVD/Tucker Transform)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Generalization of SVD

Order-3 SVD: $\longrightarrow$ $D = Z \times_1 U_1 \times_2 U_2 \times_3 U_3$

$$\mathcal{A} = \sum_{i=1}^{R_1} \sum_{j=1}^{R_2} \sum_{k=1}^{R_3} \sigma_{ijk} (\mathbf{u}_i \circ \mathbf{v}_j \circ \mathbf{w}_k)$$

Z, the core tensor, is the counterpart of S in Order-2 SVD

U_1, U_2, and U_3 are known as mode matrices

Mode matrix U_i is obtained as follows:

perform Order-2 SVD on $D_{(i)}$, the matrix obtained by flattening the tensor D on the i'th dimension.

U_i is the leftmost (column-space) matrix obtained from the SVD above

N-Mode SVD

1.For n = 1,…,N, compute matrix U_n in (1) by computing the SVD of the flattened matrix $D(n)$ and setting U_n to be the left matrix of the SVD

2.If it is needed, solve for the core tensor as follows:

$$\mathcal{Z} = \mathcal{D} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \ldots \times_n \mathbf{U}_n^T \ldots \times_N \mathbf{U}_N^T.$$

- Difference to 2D-SVD: tensor $\mathcal{Z}$ not necessarily diagonal (could be dense)

Candecomp-Parafac Decomposition (CP)

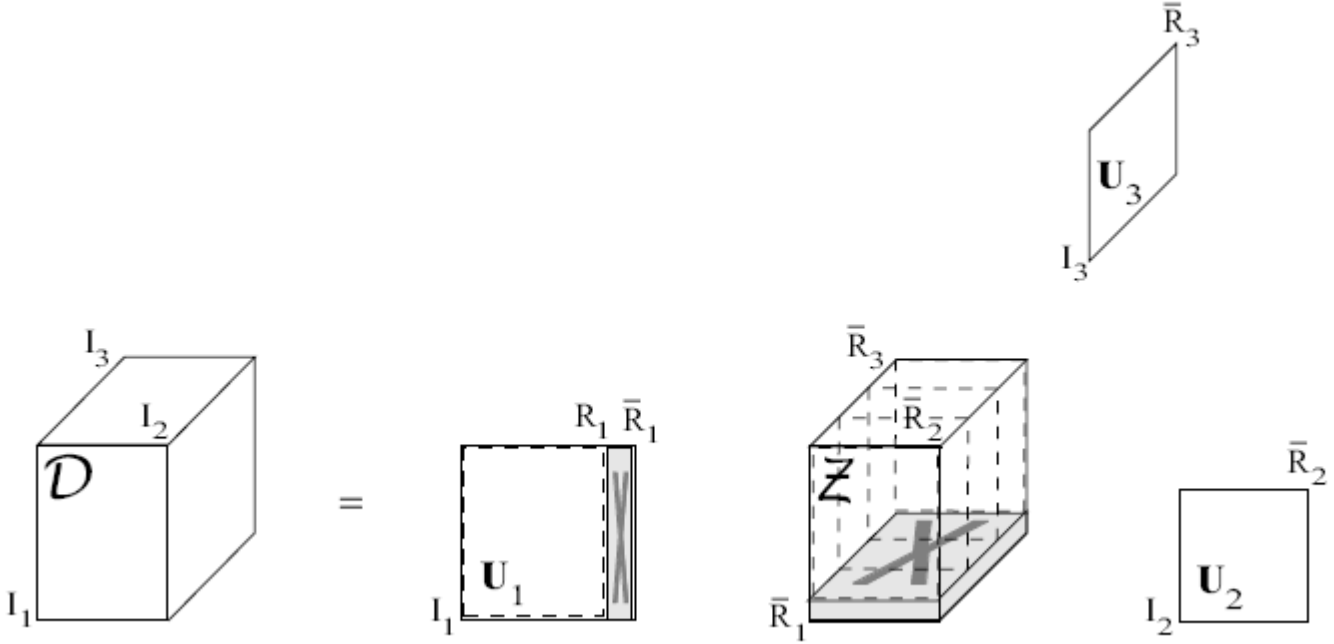
- Weighted sum of rank-1 tensors

$$\mathcal{A} = \sum_{i=1}^{R} \mathbf{u}_i \circ \mathbf{v}_i \circ \mathbf{w}_i$$

- Solved using alternating least squares (ALS)
  - Keep two components fix, solve for the other one
- Rank of the tensor if R is minimal

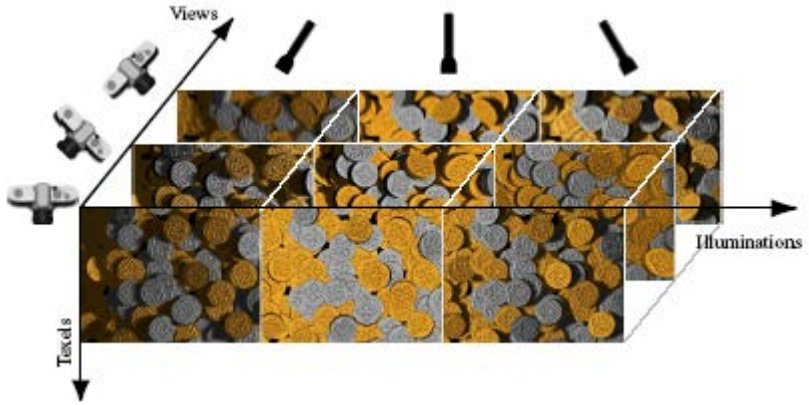
HO-PCA - Compression

- Removes some slice from Z
- Can be tailed to truncate the least important mode



PCA on Higher Order Tensors

Bidirectional Texture Function (BTF) as a Order-3 Tensor

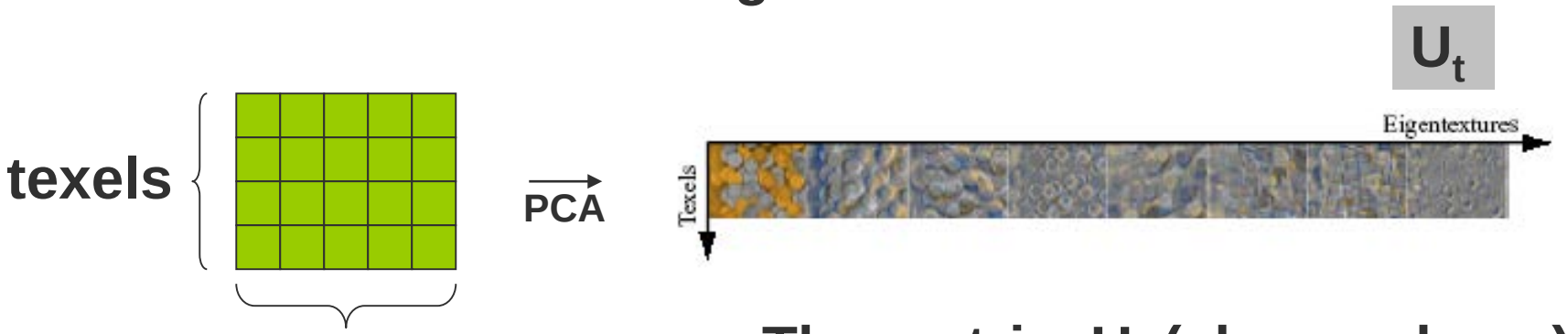


Changing illumination, changing view, and changing texels (position) along respective dimensions of the tensor

$$D = Z \times_1 U_t \times_2 U_i \times_3 U_v$$

2D-PCA on Higher Order Tensors

Performing PCA on a Order-3 Tensor



Transform data to new basis:

$F = U_t^T A$

where A is the data in original basis

The matrix, U_t (shown above), resulting from the PCA, consists of the eigenvectors (eigentextures) along its columns

Retrieving/Rendering image:

$T = Z \times_1 U_t$

Image = $T \times_1 F$

TensorTextures

A cheaper equivalent of PCA: TensorTextures

PCA: $T = Z \times_1 U_t$ ⟺ **TensorTextures:** $T = D \times_2 U_i^T \times_3 U_v^T$

Recall:

$$D = Z \times_1 U_t \times_2 U_i \times_3 U_v$$

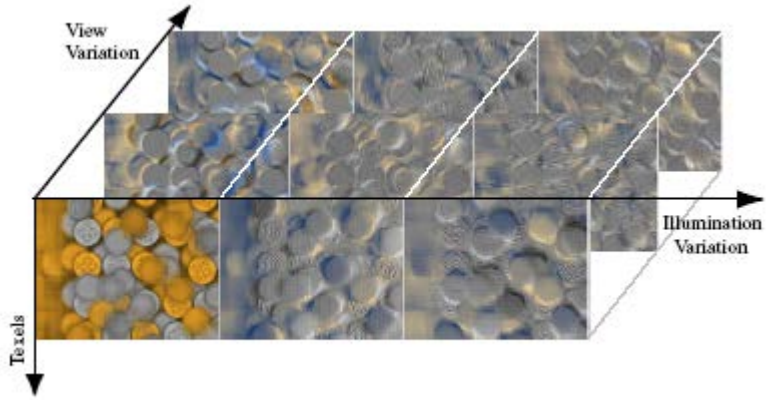
$$D = U_v \left(U_i \left(U_t Z_{(1)}\right)_{(2)}\right)_{(3)}$$

$$D = \left(U_v U_i\right) \left(U_t Z_{(1)}\right)_{(2)}$$

$$\left(U_v U_i\right)^{-1} D_{(2)} = U_t Z_{(1)}$$

$$\left(U_i^T U_v^T\right) D_{(3)} = Z \times_1 U_t$$

$$D \times_2 U_i^T \times_3 U_v^T = Z \times_1 U_t$$



Advantages of TensorTextures over PCA

PCA: $T = Z \times_1 U_t$

TensorTextures: $T = D \times_2 U_i^T \times_3 U_v^T$

Independent compression control of illumination and view in TensorTextures, whereas in PCA, the illumination and view are coupled together and thus change/compression of eigenvectors will effect both parameters.

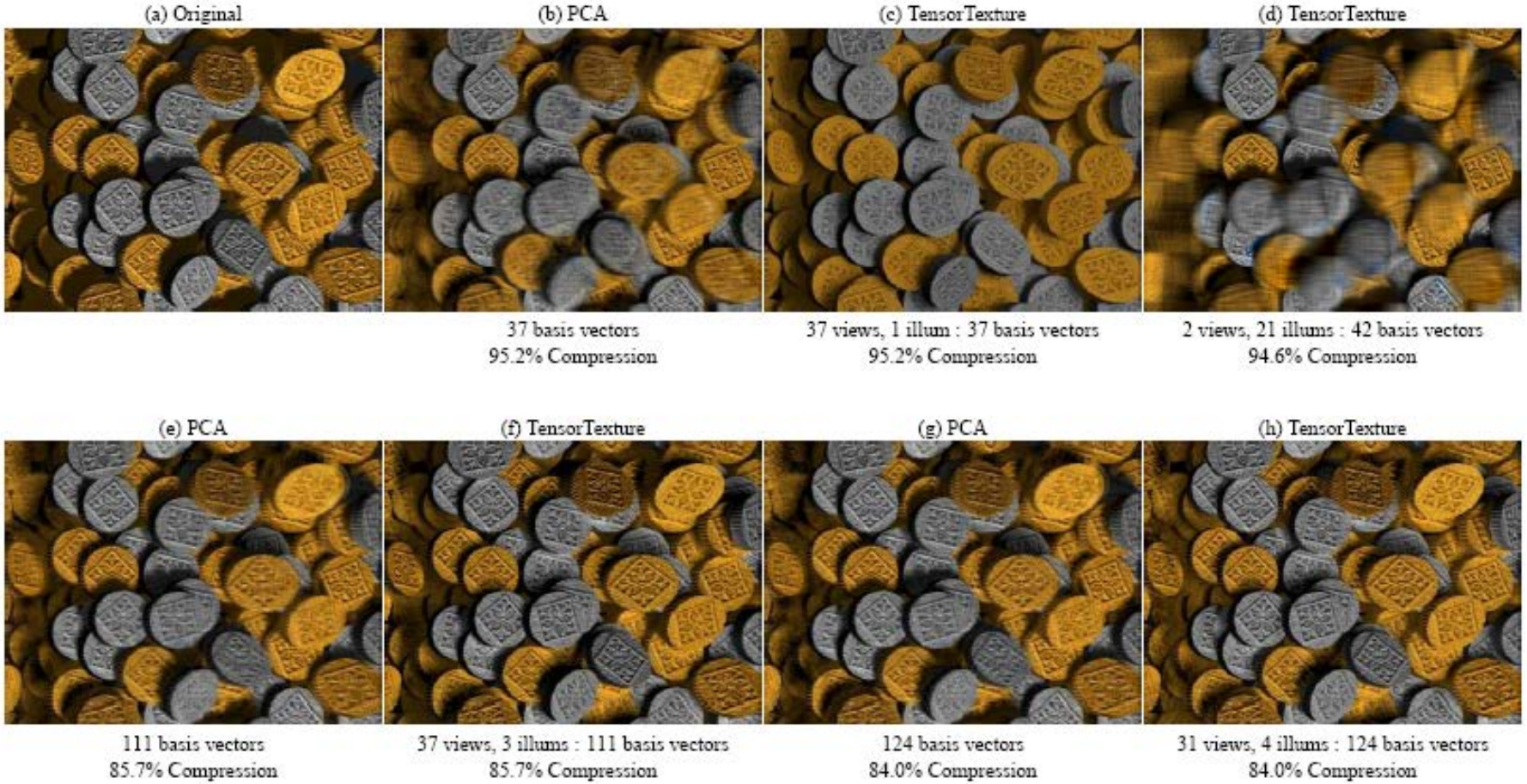
Images represented using fewer coefficients in TensorTextures.

PCA: (v * i) basis vectors, each of size t

TensorTextures: (v + i) basis vectors, of size v and I

where v = # of view directions, i = # of illumination conditions, and t = # of texels in each image used in the creation of the tensor

TensorTextures Compression vs PCA compression



Notice that PCA has one set of basis vectors where as TensorTextures has two: one for illumination, and one for view. This enables selective compression to a desired level of tolerance of perceptual error

Sparse + Low-rank Decomposition

Compressive Sparse plus Low-rank Decomposition

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Decompose reflectance field into sparse plus low-rank
- Single view, single pixel, spectral environment map:

$$L(\omega_o, \lambda_o) = R(\omega_o, \lambda_o, \omega_i, \lambda_i) \circ L_e(\omega_i, \lambda_i)$$

$$L(\omega_o, \lambda_o) = < \overbrace{\left(R \right)}^{\lambda_i}, \overbrace{\left(L_e \right)}^{\lambda_i} >$$

Compressive Sparse plus Low Rank Decomposition

- Decompose reflectance field into sparse plus low rank
- Single view, single pixel, spectral environment map:

$$L(\omega_o, \lambda_o) = R(\omega_o, \lambda_o, \omega_i, \lambda_i) \circ L_e(\omega_i, \lambda_i)$$

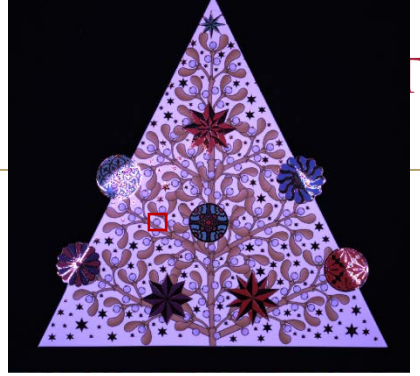
$$L(\omega_o, \lambda_o) = < \left(\int_{\omega_i} R \right), \left(\int_{\omega_i} L_e \right) >$$

Multiplexed Measurements

- Illuminate with n random hyperspectral patterns of direction and wavelength

$$\begin{pmatrix} l^0 \\ l^1 \\ \vdots \\ l^n \end{pmatrix} = \begin{pmatrix} L_e{}^0 \\ L_e{}^1 \\ \vdots \\ L_e{}^n \end{pmatrix} R \qquad C = MR$$

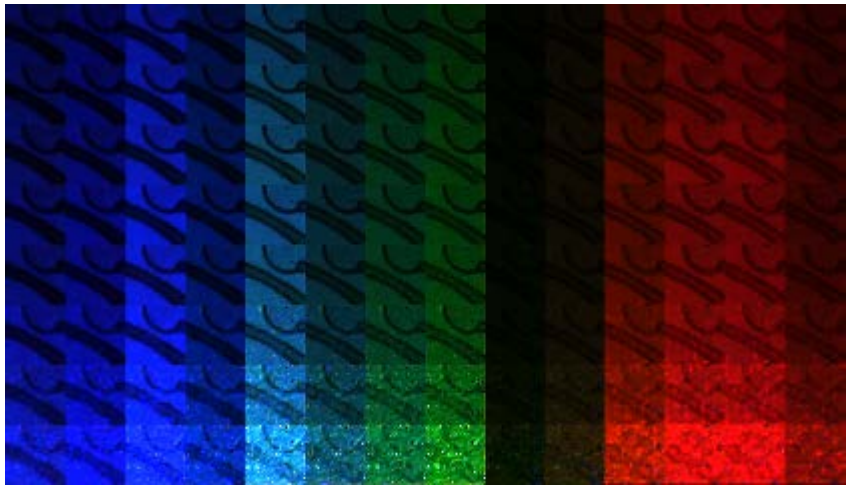
Representing the Reflectance Field



- Decompose R into a sparse and a low-rank component
  - both require less images than full-resolution measurement

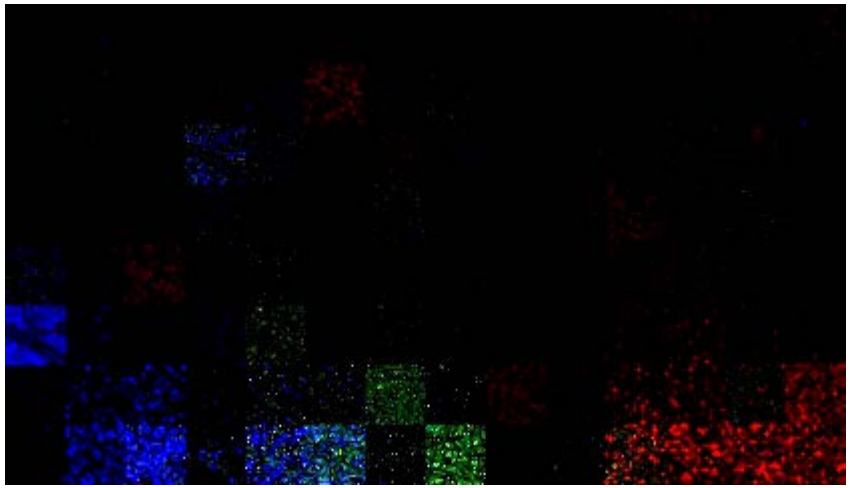
$$R = A + B$$

λ_i



ω_i

$R = A + B$



A

Simultaneous Sparse and Low-rank Decomposition

- [Yuan and Yang 2009]

$$\min_{A,B} \quad \gamma \|A\|_{l_1} + \|B\|_*$$

$$s.t. \quad A + B = C$$

- Solution using augmented Lagrangian or fixed point method

Compressive Sparse plus Low-rank Decomposition

- Slightly different problem: C is not directly observed but by few compressive samples

$$\min_{A,B} \quad \gamma \|A\|_{l_1} + \|B\|_*$$

$$s.t. \quad M(A+B) = C$$

- Solution again following the augmented Lagrangian and fixed point iteration

Compressive Sparse plus Low-rank Decomposition

- Different problem:
  C is not directly observed but by few compressive samples

$$\min_{A,B} \quad \gamma \|A\|_{l_1} + \|B\|_*$$

$$s.t. \quad M(A+B) = C$$

- Solution again following the augmented Lagrangian and fixed point iteration
- Similar approach w/ fixed rank, fixed sparsity [Walters et al. 2011]

Augmented Lagrangian

- Augmented Lagrangian formulation:

$$\min_{A,B} \quad \gamma \|A\|_{l_1} + \|B\|_*$$
$$s.t. \quad M(A+B) = C$$

- minimize

$$L(A,B,Z) = \gamma ||A||_{\ell_1} + ||B||_* - \langle Z, M(A+B) - C \rangle + \frac{\beta}{2} ||M(A+B) - C||_F^2$$

- Subgradient for iterative scheme

$$\mathbf{0} \in \gamma \partial (||A^{k+1}||_{\ell_1}) - M^T (Z^k - \beta (M(A^{k+1} + B^k) - C))$$
$$\mathbf{0} \in \partial (||B^{k+1}||_*) - M^T (Z^k - \beta (M(A^{k+1} + B^{k+1}) - C))$$
$$Z^{k+1} = Z^k - \beta (M(A^{k+1} + B^{k+1}) - C)$$

Augmented Lagrangian

- Augmented Lagrangian formulation:
  minimize

$$\min_{A,B} \quad \gamma\|A\|_{l_1} + \|B\|_*$$
$$s.t. \quad M(A+B) = C$$

$$L(A,B,Z) = \gamma||A||_{\ell_1} + ||B||_* - \langle Z, M(A+B) - C\rangle + \frac{\beta}{2}||M(A+B) - C||_F^2$$

- Iterative scheme
  - update A
  - update B
  - update Z

- Question: How to mimize for the two norms?

l_1 - Shrinkage

- hard shrinkage on each entry of A

$$\min_{A,B} \gamma\|A\|_{l_1} + \|B\|_* \quad s.t. \quad M(A+B) = C$$

$$\mathbf{0} \in \gamma\partial(||A^{k+1}||_{\ell_1}) - M^T(Z^k - \beta(M(A^{k+1} + B^k) - C))$$

$$Y_A = A^k + M^T(Z^k - \beta(M(A^k + B^k) - C))$$

$$A^{k+1} = S^{\ell_1}_{\frac{\gamma\tau}{\beta}}(Y_A)$$

- with

$$S^{\ell_1}_v(X) = sign(X_{ij})\ max\{|X_{ij}| - v, 0\}$$

Nuclear Norm Shrinkage

- hard shrinkage on singular values of B

$$\min_{A,B} \quad \gamma\|A\|_{l_1} + \|B\|_* \qquad s.t. \quad M(A+B) = C$$

$$\mathbf{0} \in \partial(||B^{k+1}||_*) - M^T(Z^k - \beta(M(A^{k+1} + B^{k+1}) - C))$$

$$Y_B = B^k + M^T(Z^k - \beta(M(A^{k+1} + B^k) - C))$$

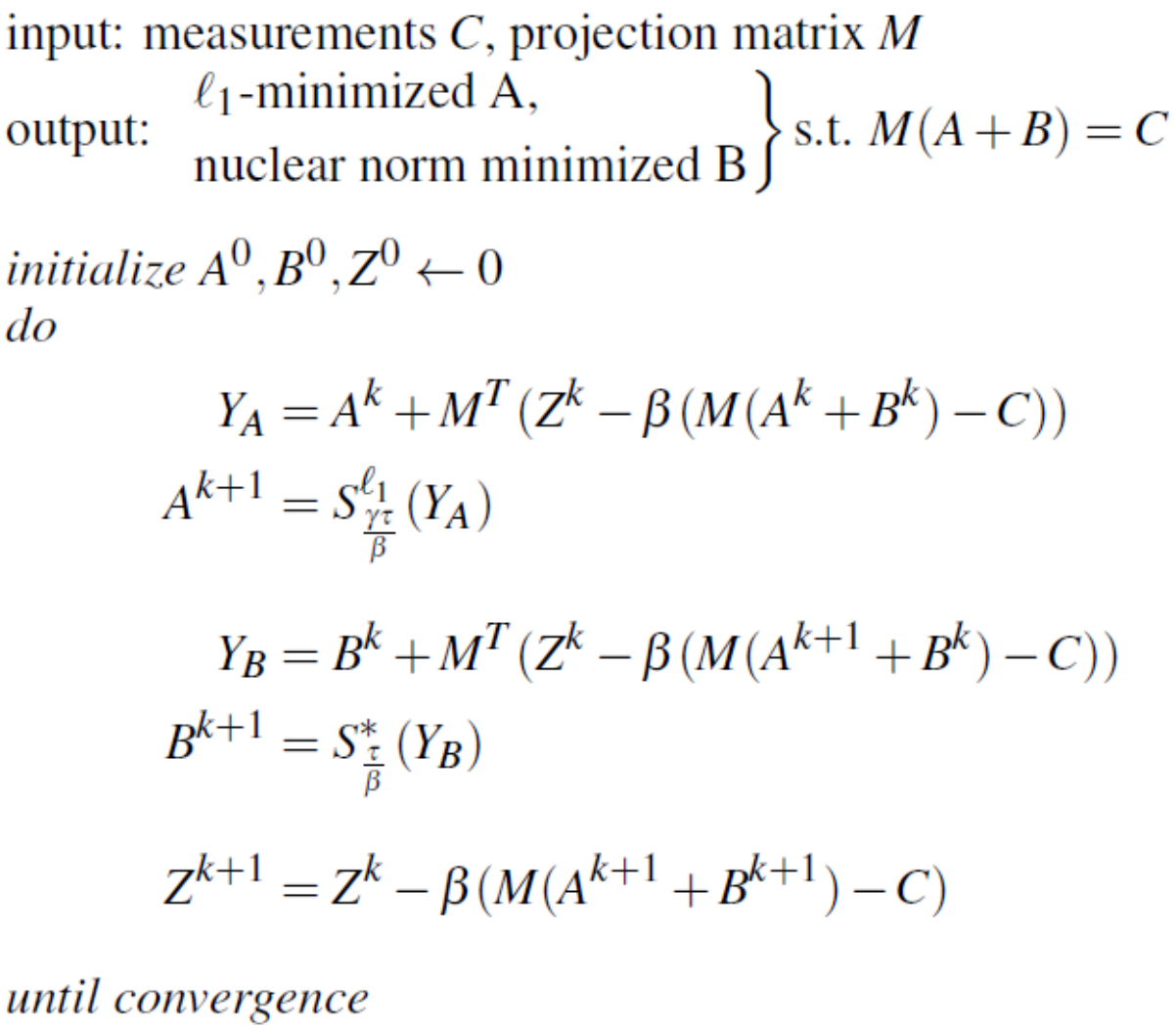
$$B^{k+1} = S^*_{\frac{\tau}{\beta}}(Y_B)$$

with

$$S_v(X) = UDiag(\overline{\sigma})V^T$$

$$\overline{\sigma_i} = \begin{cases} \sigma_i - v, & \text{if } \sigma_i - v > 0; \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm



input: measurements C, projection matrix M

output: $\left.\begin{array}{l}\ell_1\text{-minimized A,}\\ \text{nuclear norm minimized B}\end{array}\right\}$ s.t. $M(A+B)=C$

initialize $A^0, B^0, Z^0 \leftarrow 0$

do

$$Y_A = A^k + M^T\left(Z^k - \beta\left(M(A^k+B^k)-C\right)\right)$$

$$A^{k+1} = S^{\ell_1}_{\frac{\gamma\tau}{\beta}}(Y_A)$$

$$Y_B = B^k + M^T\left(Z^k - \beta\left(M(A^{k+1}+B^k)-C\right)\right)$$

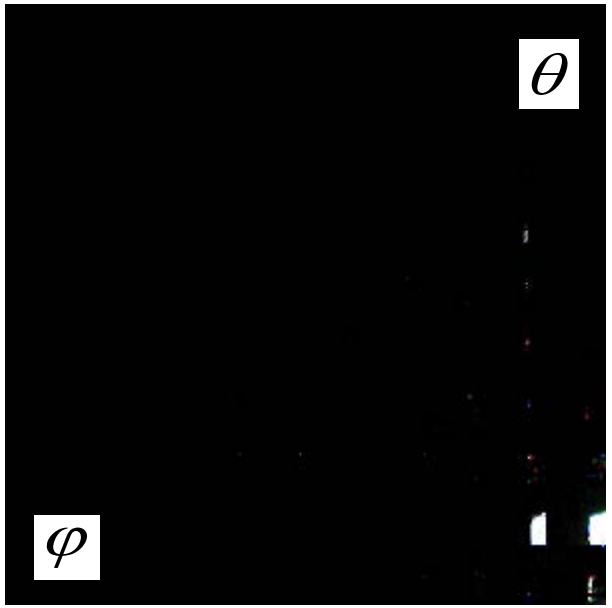
$$B^{k+1} = S^{*}_{\frac{\tau}{\beta}}(Y_B)$$

$$Z^{k+1} = Z^k - \beta\left(M(A^{k+1}+B^{k+1})-C\right)$$

until convergence

Separation Results

$A + B$



A

sparse
specular highlight

B

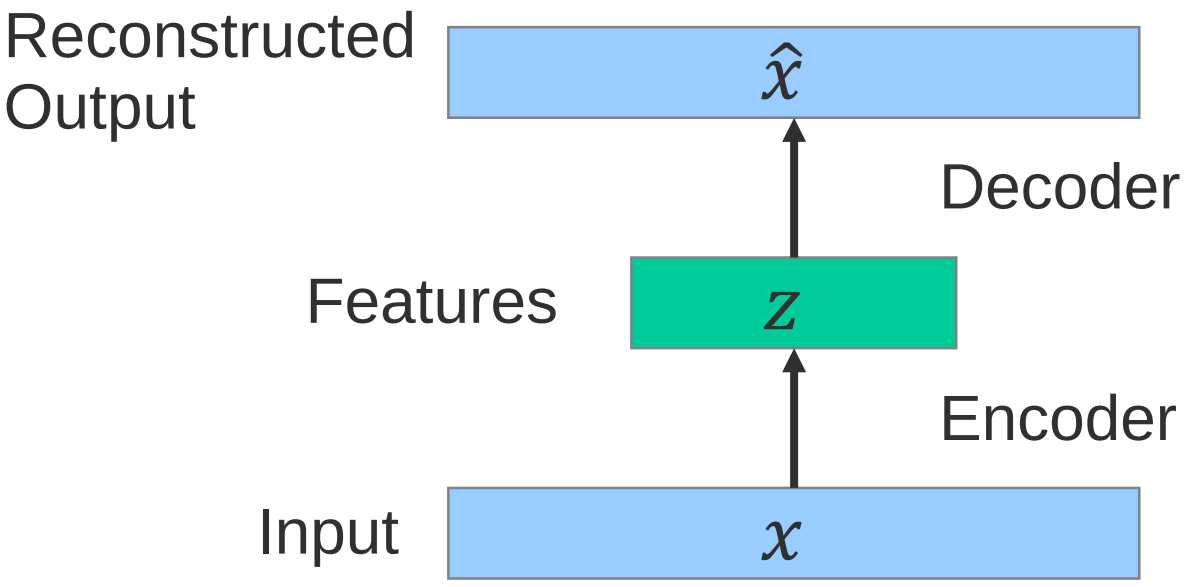
low-rank
diffuse reflection

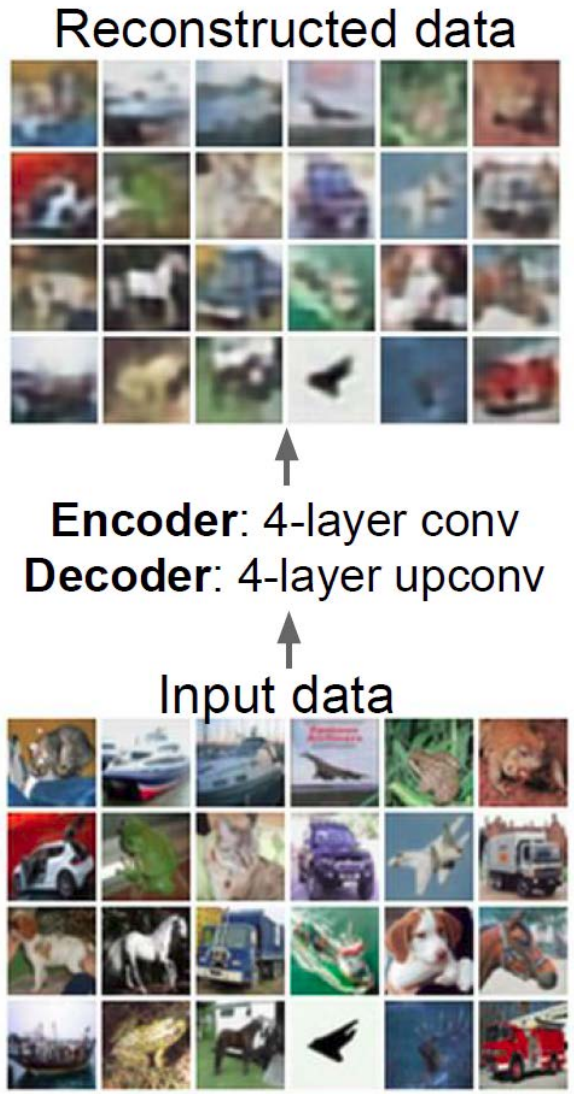
Variational Autoencoder

[slides based on FeiFei et al. **Generative Models - CS231n**]

Autoencoders

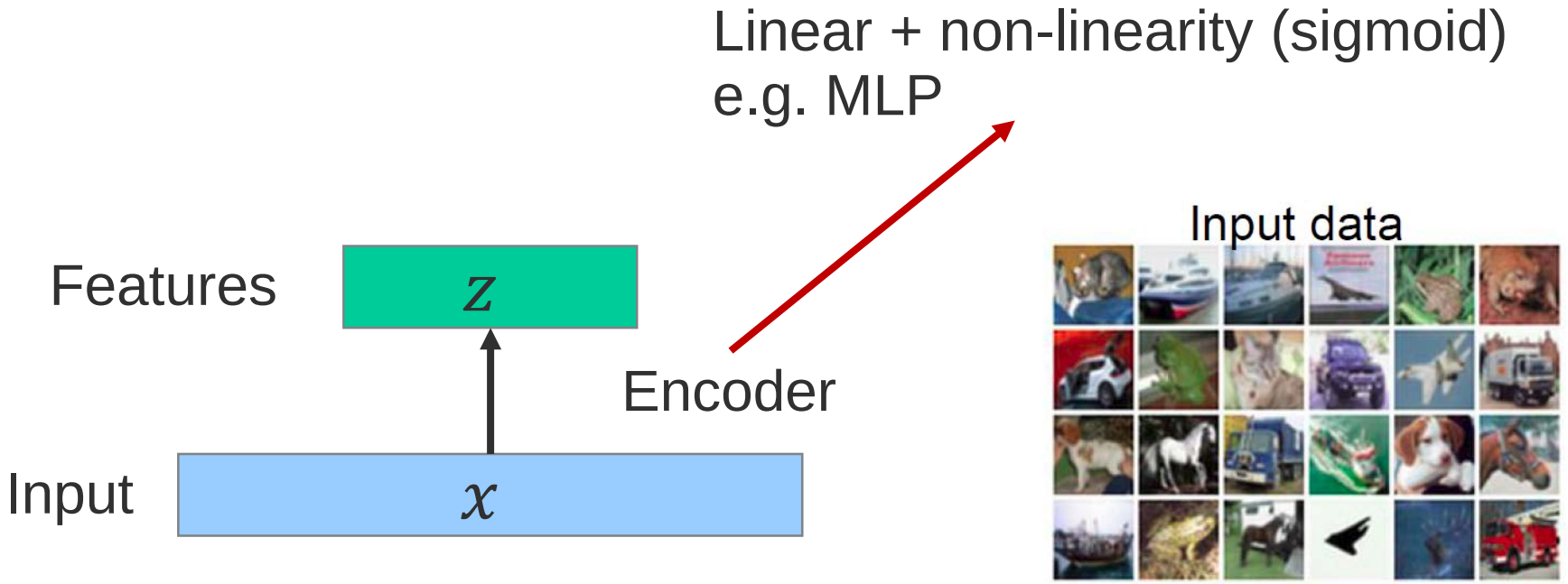
- Unsupervised appraoch for learning a lower-dimensional feature repr
  unlabeled training data





Autoencoders

- Unsupervised appraoch for learning a lower-dimensional feature representation from unlabeled training data

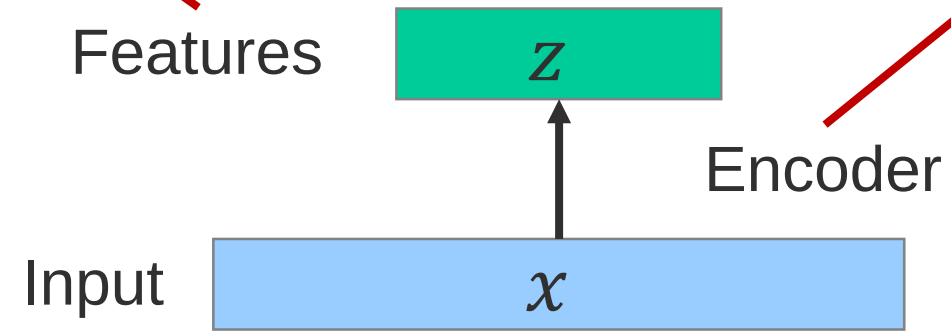


Autoencoders

- Unsupervised appraoch for learning a lower-dimensional feature representation from unlabeled training data

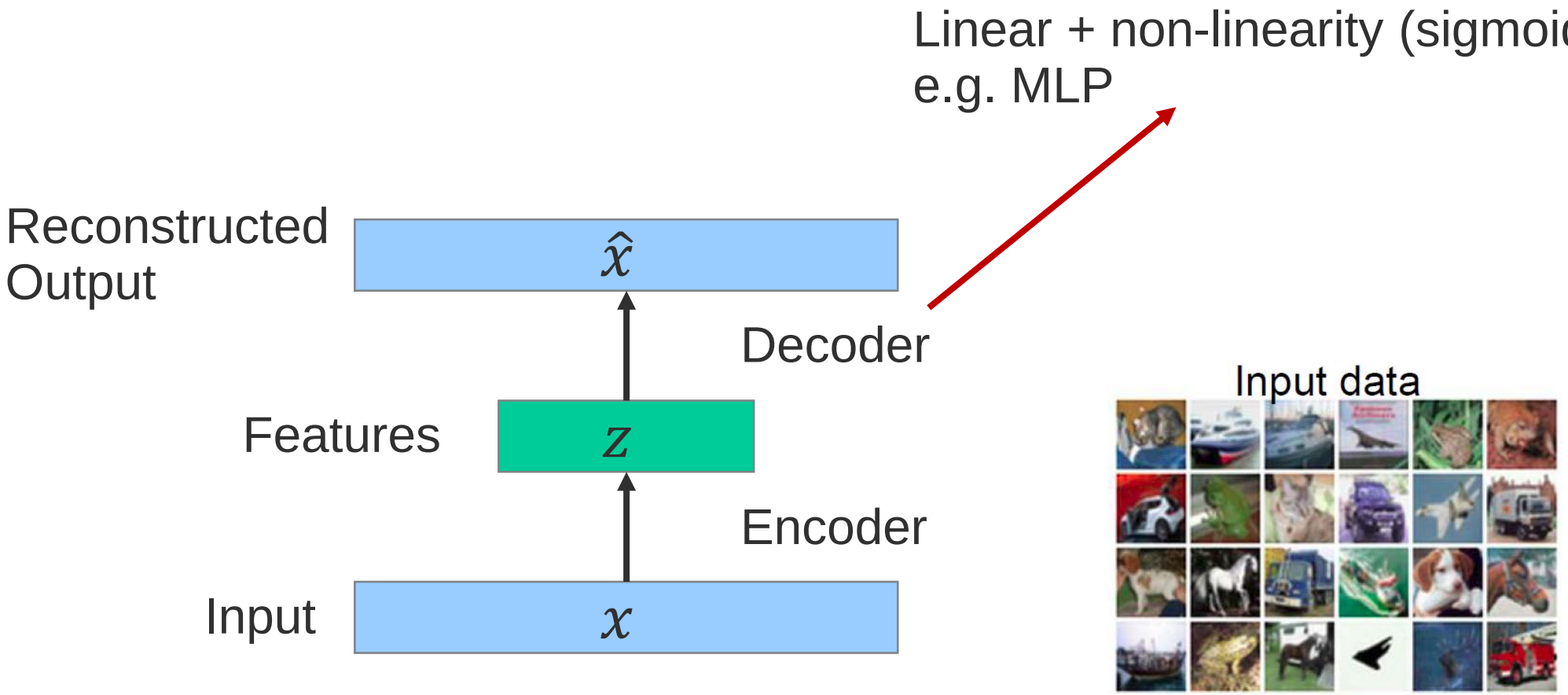
Dimensionality reduction:
z is smaller than x
Goal: features should capture meaningful factors of variation

Linear + non-linearity (sigmoid)
e.g. MLP



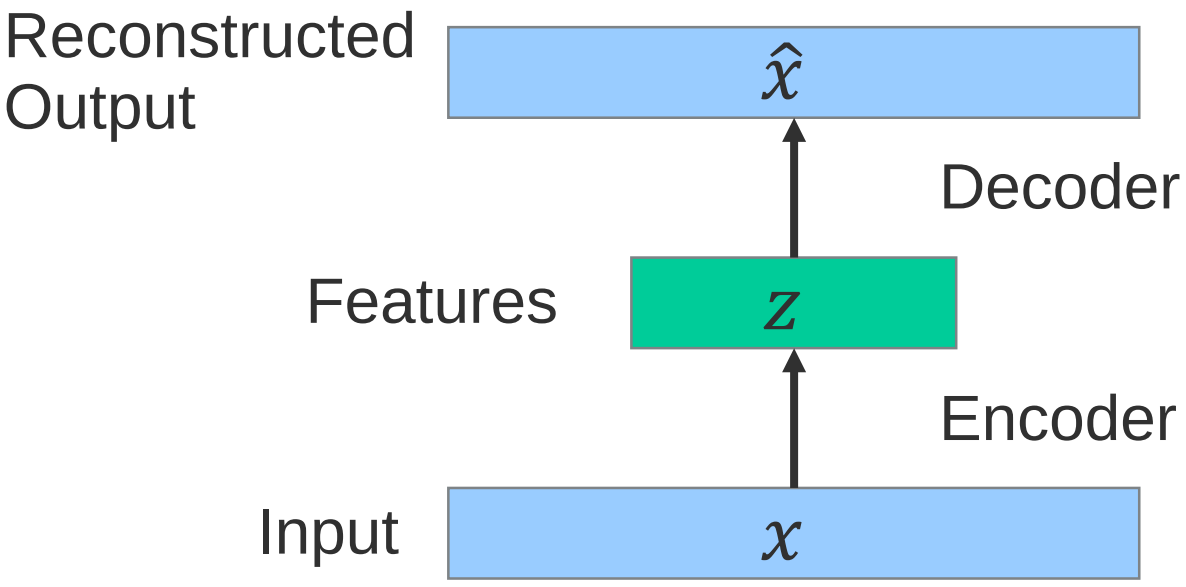
Autoencoders – How to train?

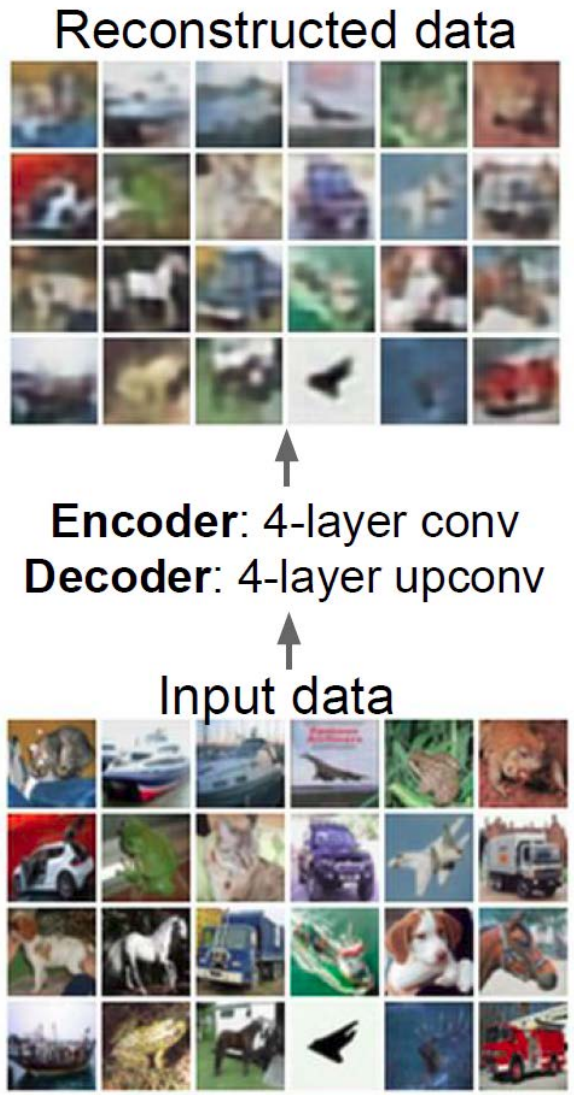
- Train such that features can be used to reconstruct original data
- “Autoencoding” - encoding itself



Autoencoders – How to train?

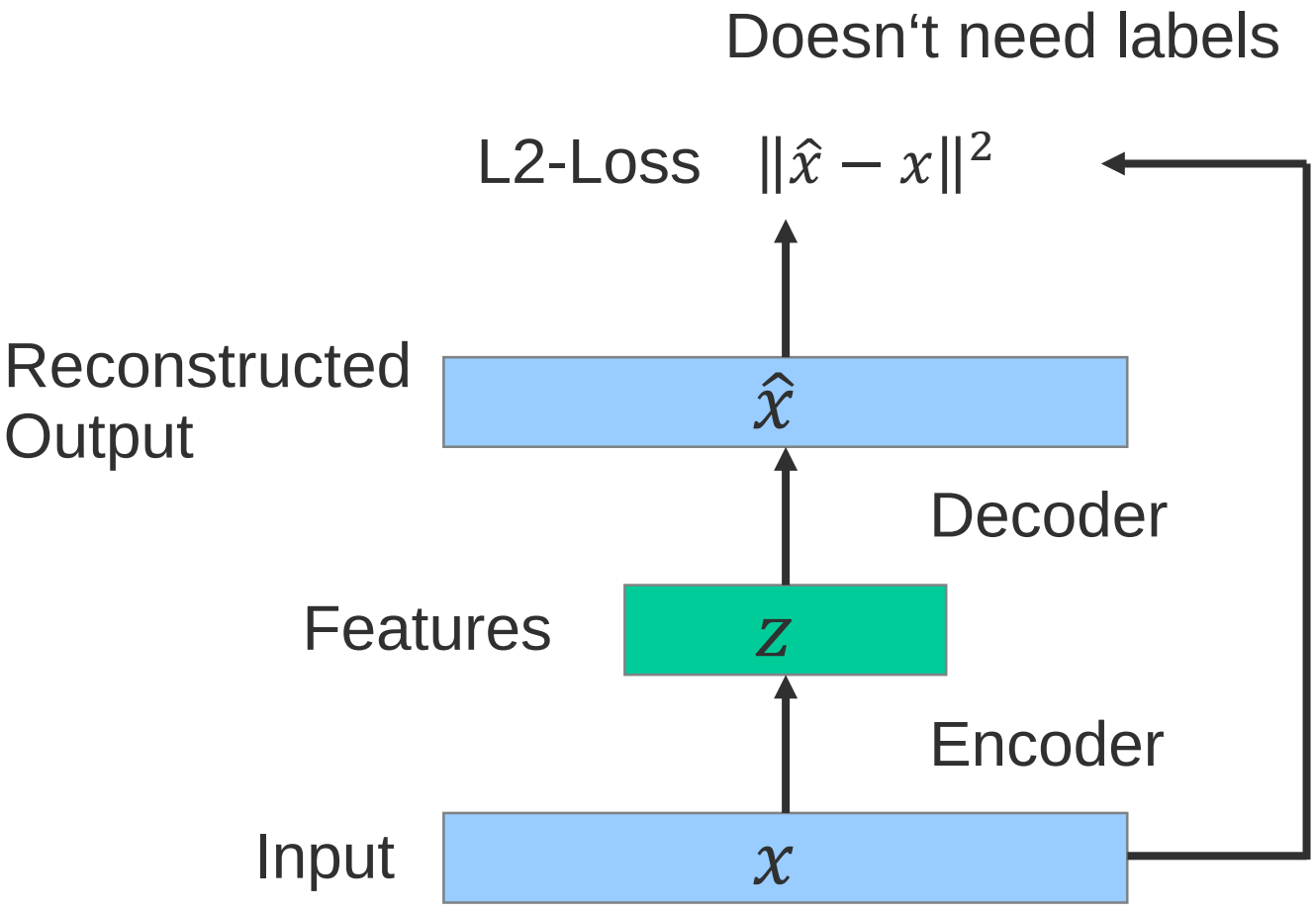
- Train such that features can be used to reconstruct original data
- “Autoencoding” - encoding itself

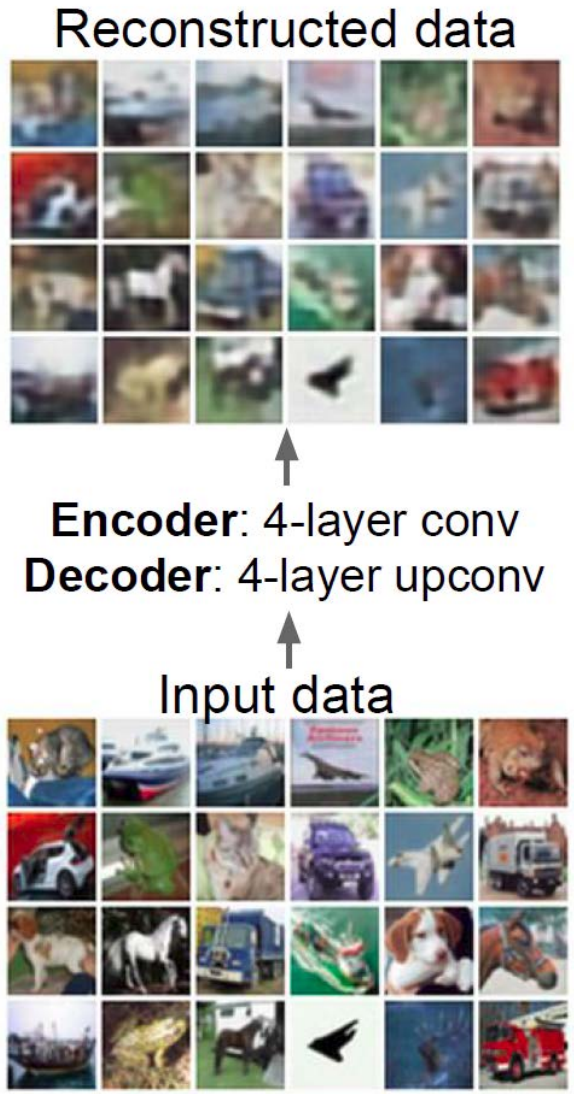




Autoencoders – How to train?

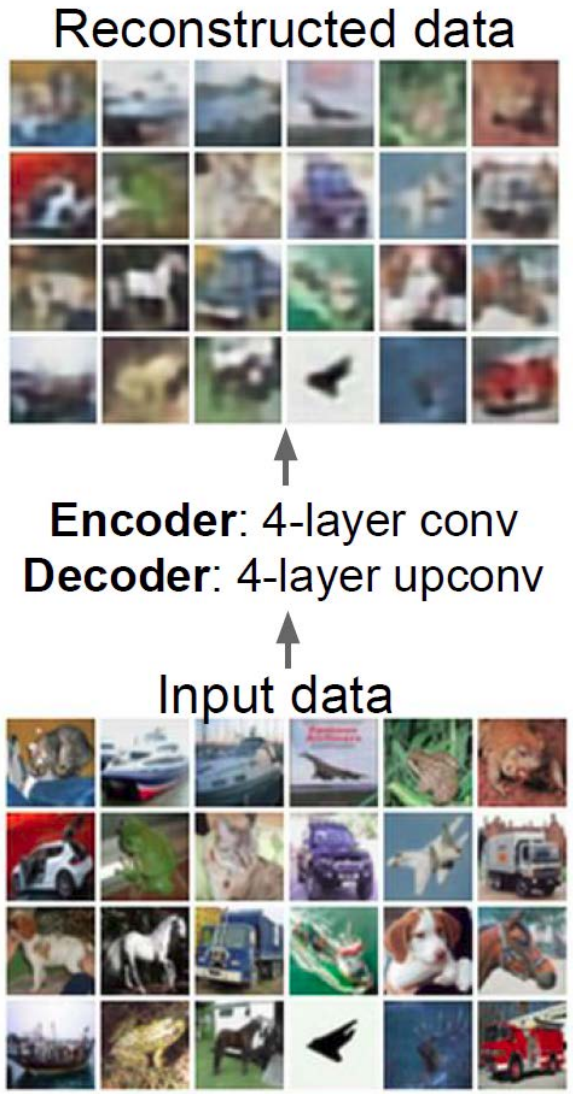
- Train such that that features can be used to reconstruct original data

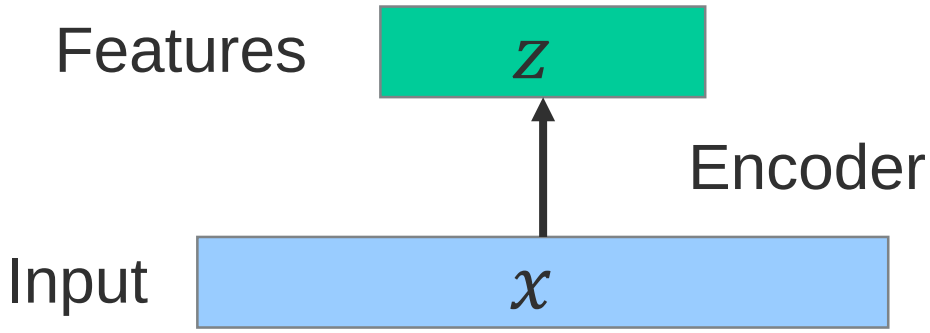




Autoencoders – How to train?

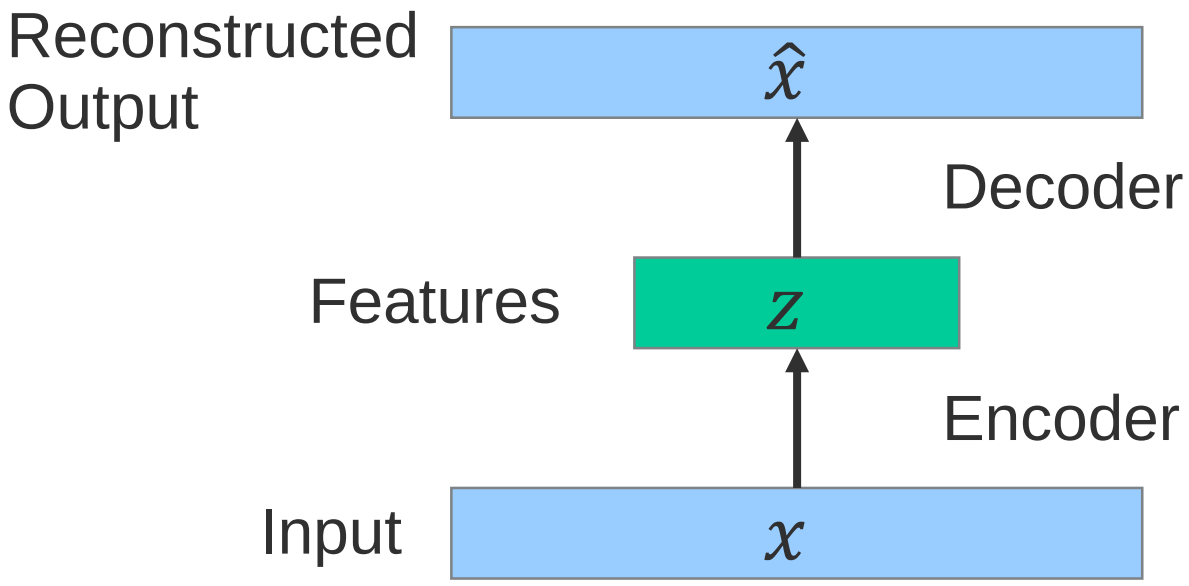
- Train such that that features can be used to reconstruct original data
- For compression: forget about the decoder after training





Autoencoders – How to train?

- Autoencoders can reconstruct data
- Features capture factors of variation in training data.
- Can we generate new images from an autoencoder?

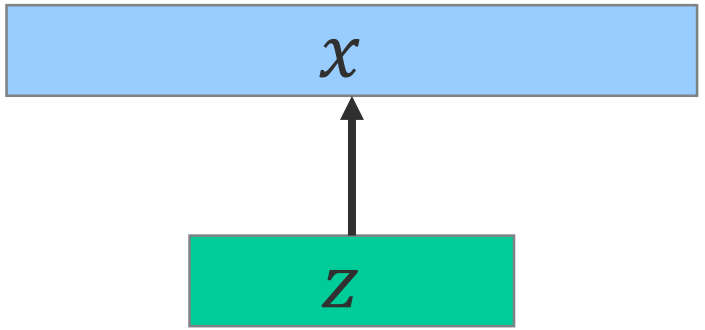


Variational Autoencoders

- Add probabilistic evaluation - will let us sample from the model to generate data!
- Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation z

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



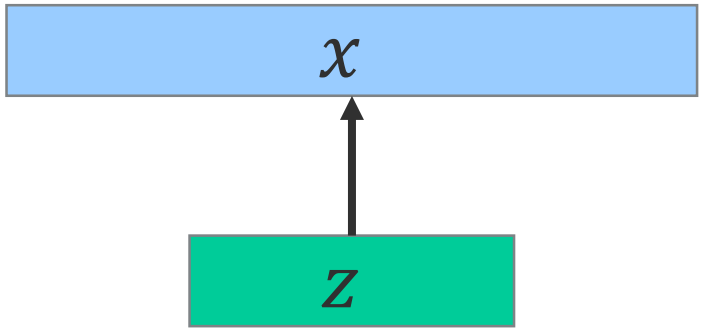
[Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014]

Variational Autoencoders

- Add probabilistic evaluation - will let us sample from the model to generate data!
- Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation z
- **Intuition**: x is an image, z is latent factors used to generate x**:** attributes, orientation, etc.
- **Goal:** We want to estimate the true parameters θ^* of this generative model.

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



[Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014]

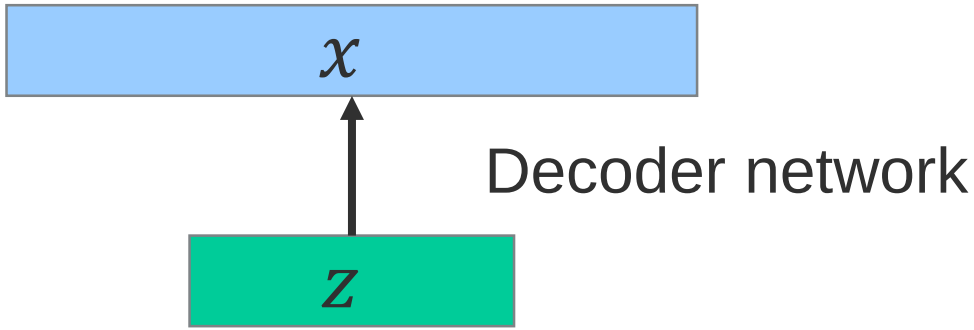
Variational Autoencoders

- **Goal:** We want to estimate the true parameters θ^* of this generative model.

- How should we represent this model?
- Choose prior p(z) to be simple, e.g. Gaussian.
- Conditional p(x|z) is complex (generates image) → represent with neural network

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$

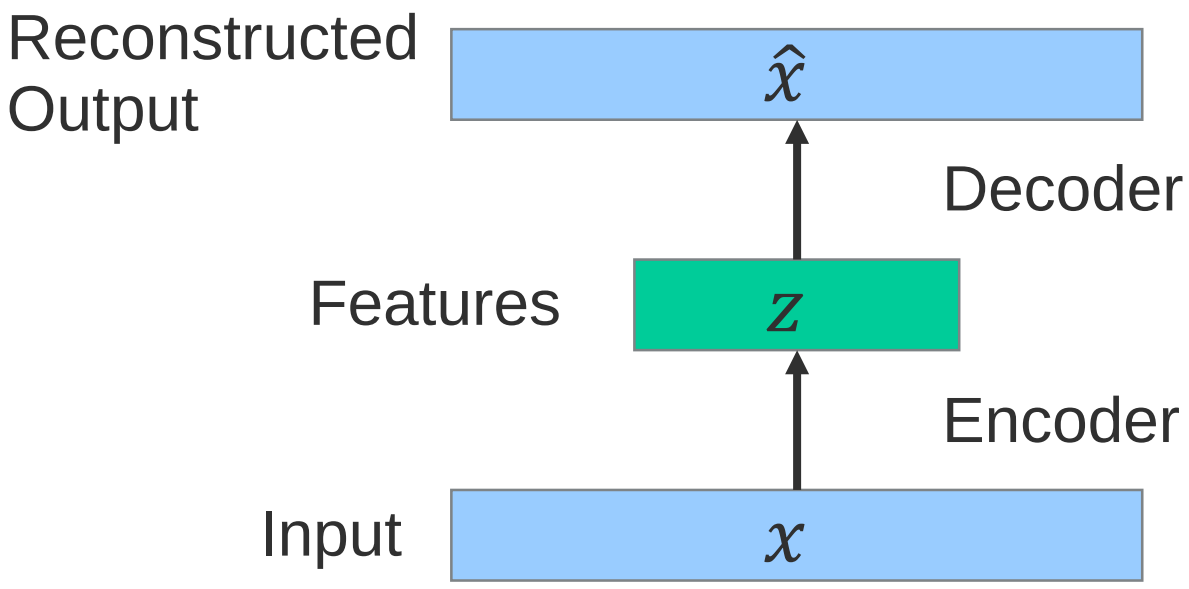


[Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014]

Training Variational Autoencoders

- In addition to decoder network modeling $p_\theta(x|z)$,
  define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$.

- Allows us to derive a lower bound on the data likelihood that we can optimize

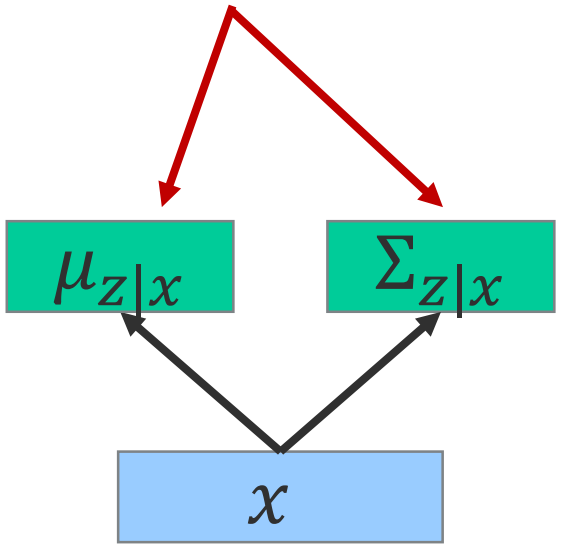


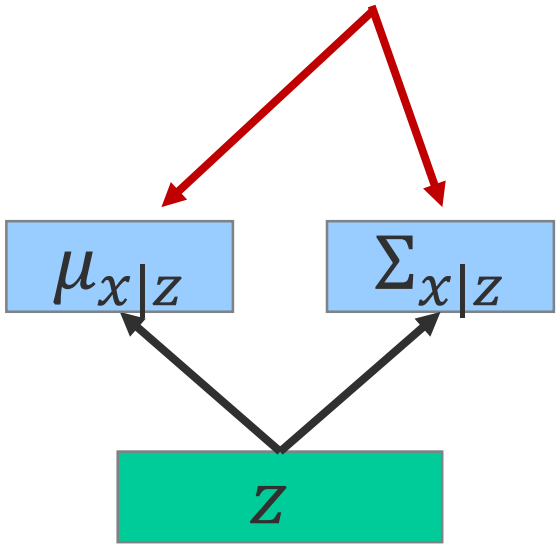
Variational Autoencoders

- Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and covariance of $(z|x)$

Mean and covariance of $(x|z)$





Encoder network
$q_\phi(z|x)_\theta$
(parameters ϕ)

Decoder network
$p_\theta(x|z)$
(parameters θ)

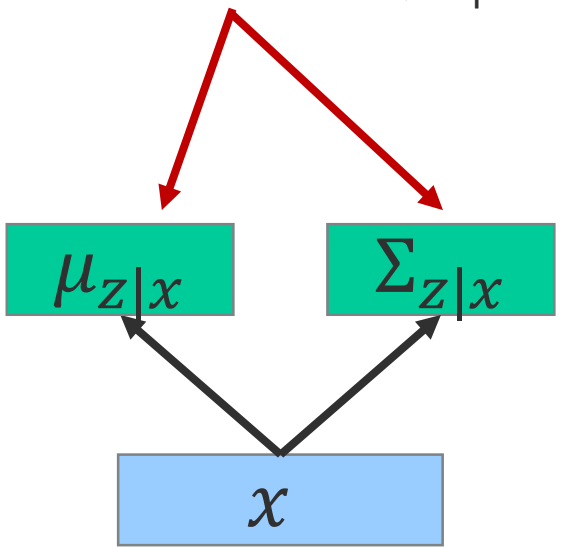
Variational Autoencoders

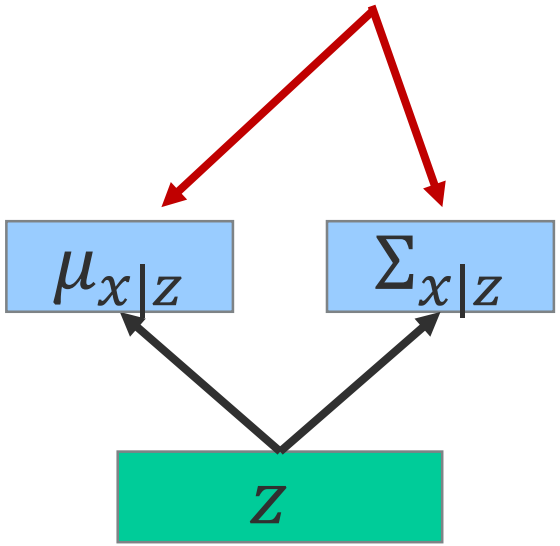
- Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic
- Encoder and decoder networks also called "inference" and "generation" networks

Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

Encoder network
$q_\phi(z|x)_\theta$
(parameters ϕ)





Decoder network
$p_\theta(x|z)$
(parameters θ)

Variational Inference

- Approximate $q(z|x)$ by p(z)

- Minimize the (Kullback-Leibler) KL-Divergence:

$$D_{KL}[p(z)||q(z|x)] = \int p(z)\log\frac{p(z)}{q(z|x)}dz$$

- Measure of deviation between two pdfs

Variational Autoencoders - Log-Likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Taking expectation wrt. z (using encoder network) will come in handy later

Variational Autoencoders - Log-Likelihood

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && (\text{Logarithms})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Variational Autoencoders - Log-Likelihood

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

The expectation wrt. z (using encoder network) let us write nice KL terms

Variational Autoencoders - Log-Likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Cannot marginalize over all possible $x^{(i)}$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders - Log-Likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders - Log-Likelihood - ELBO

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoders - Log-Likelihood - ELBO

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior, e.g. normal

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

Variational Autoencoders – Training – full cycle

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Maximize the likelihood lower bound
- Let's look at computing the bound (forward pass) for a given minibatch of input data

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

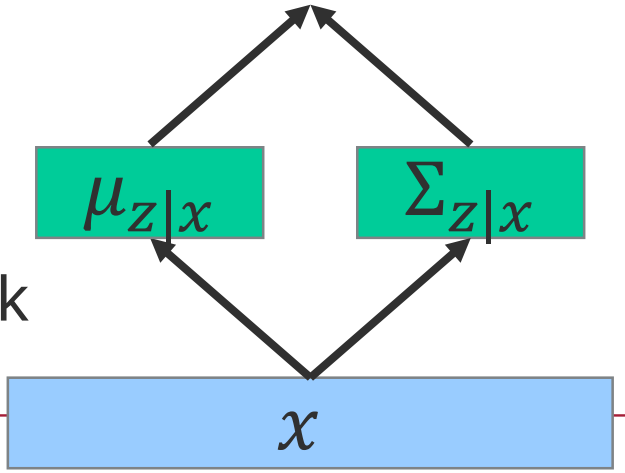
Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- Let's look at computing the bound (forward pass) for a given minibatch of input data

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior, e.g. normal

Encoder network $q_\phi(z|x)$

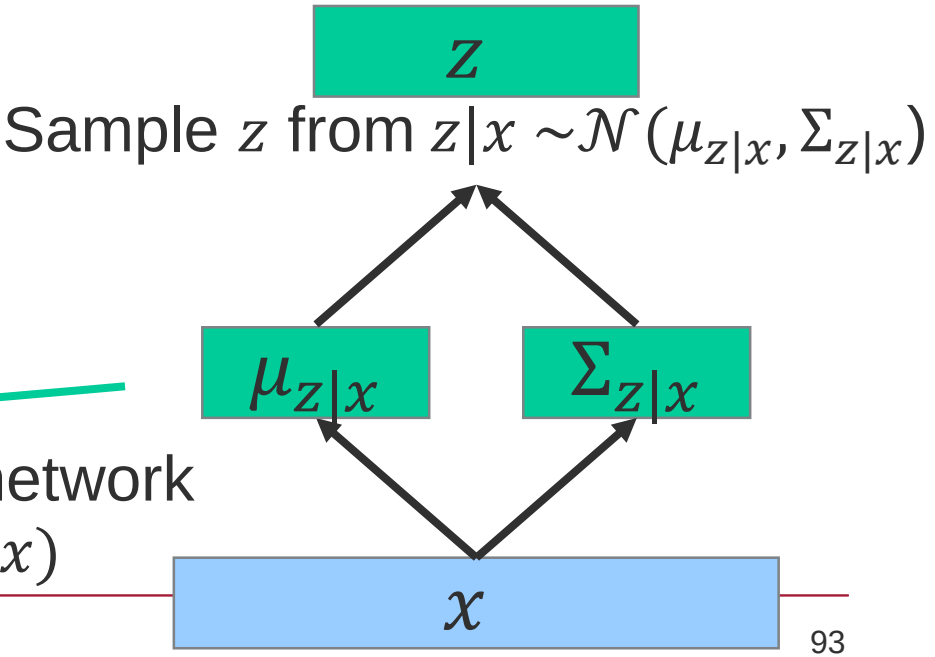


Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- Let's look at computing the bound (forward pass) for a given minibatch of input data

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior, e.g. normal

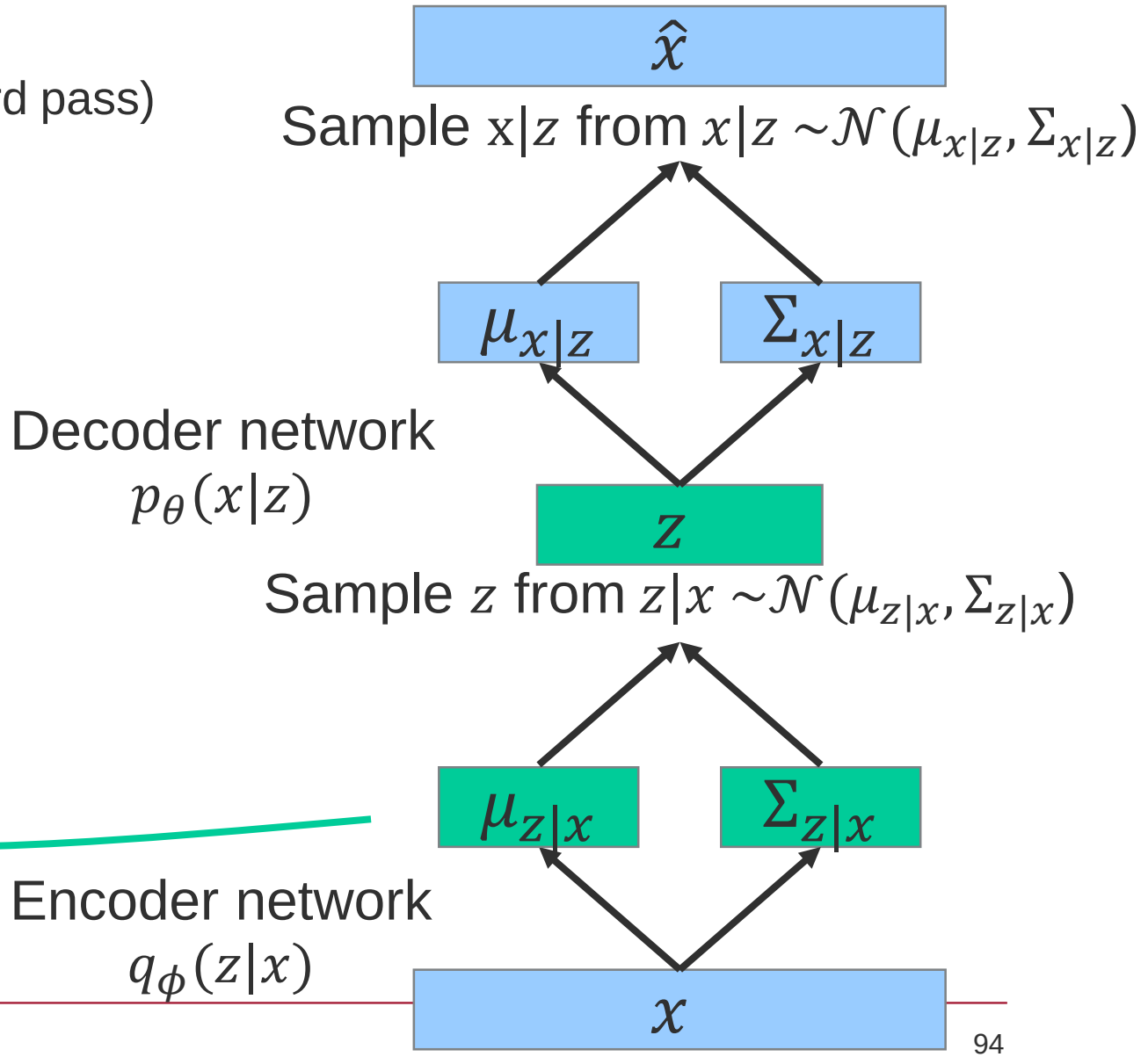


Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- Let's look at computing the bound (forward pass) for a given minibatch of input data

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior, e.g. normal



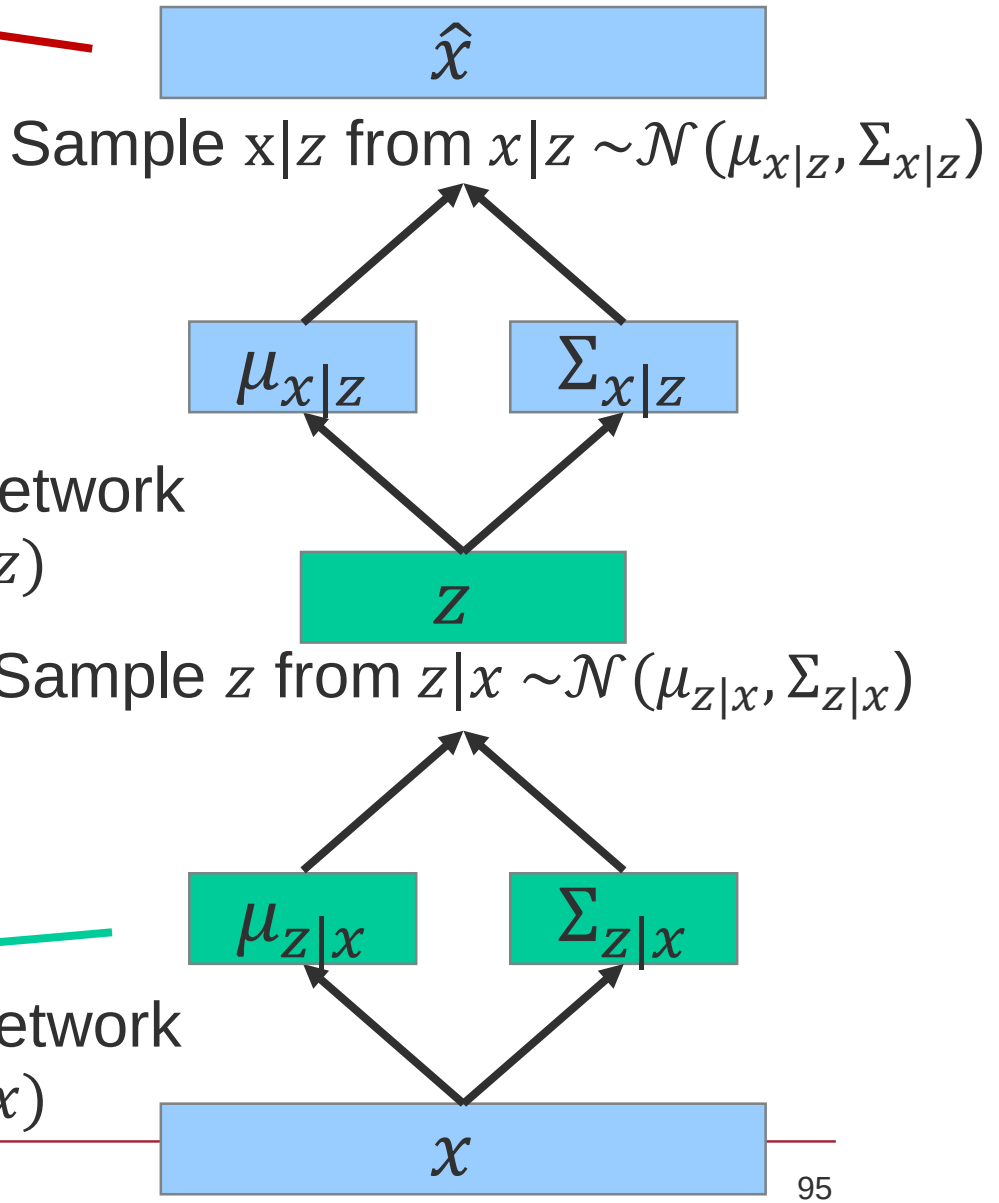
Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- Let's look at computing the bound (forward pass) for a given minibatch of input data

Maximize likelihood of original input being reconstructed

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior, e.g. normal



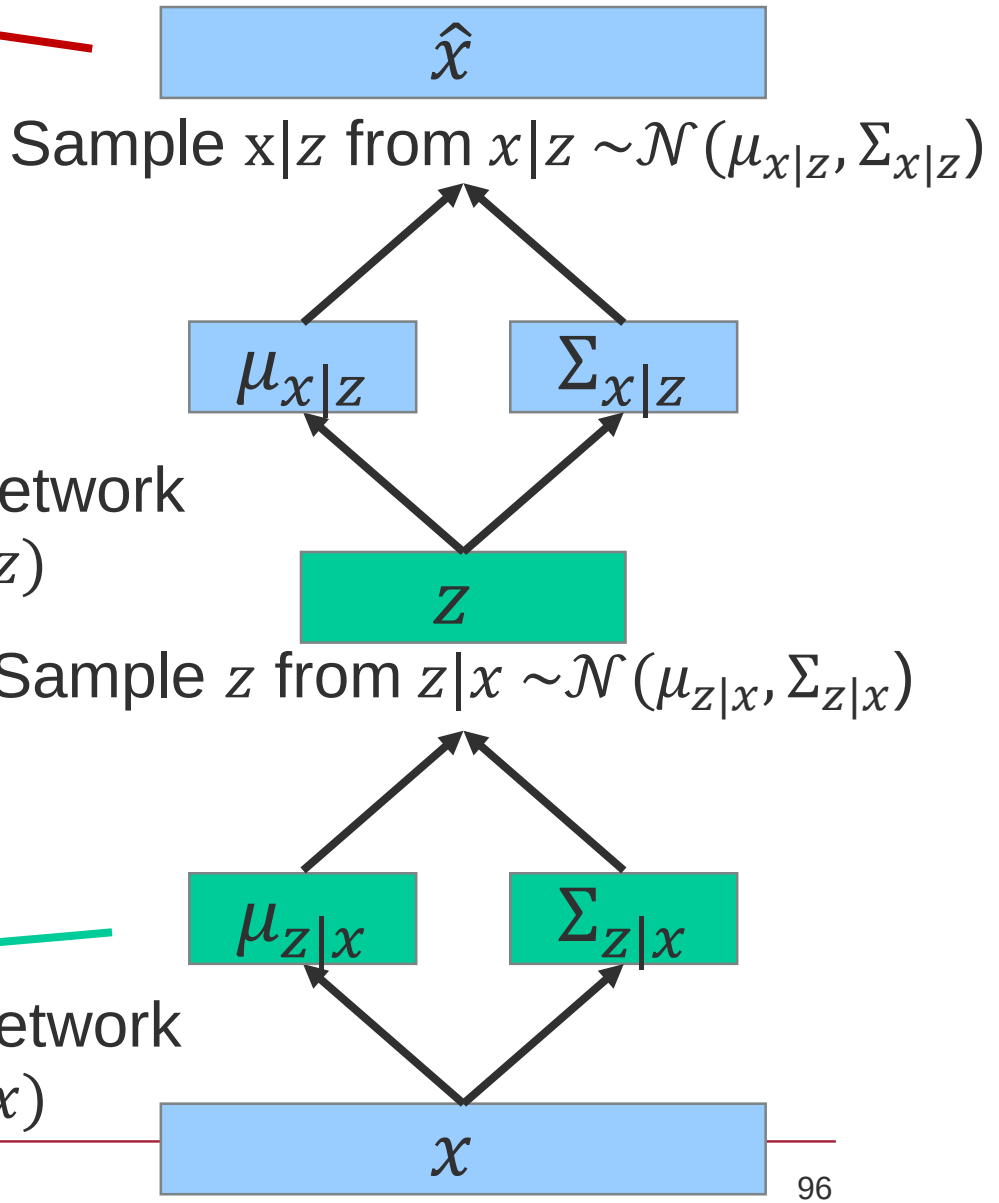
Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- **For every minibatch of input data: compute this forward pass, and then backprop!**

Maximize likelihood of original input being reconstructed

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

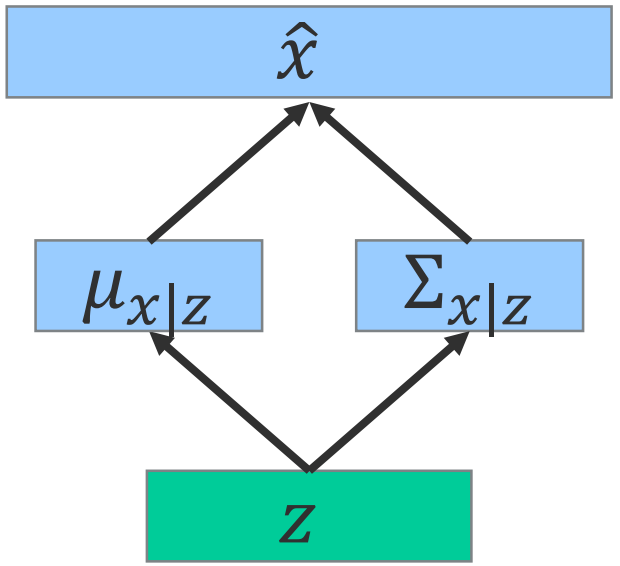
Make approximate posterior distribution close to prior, e.g. normal



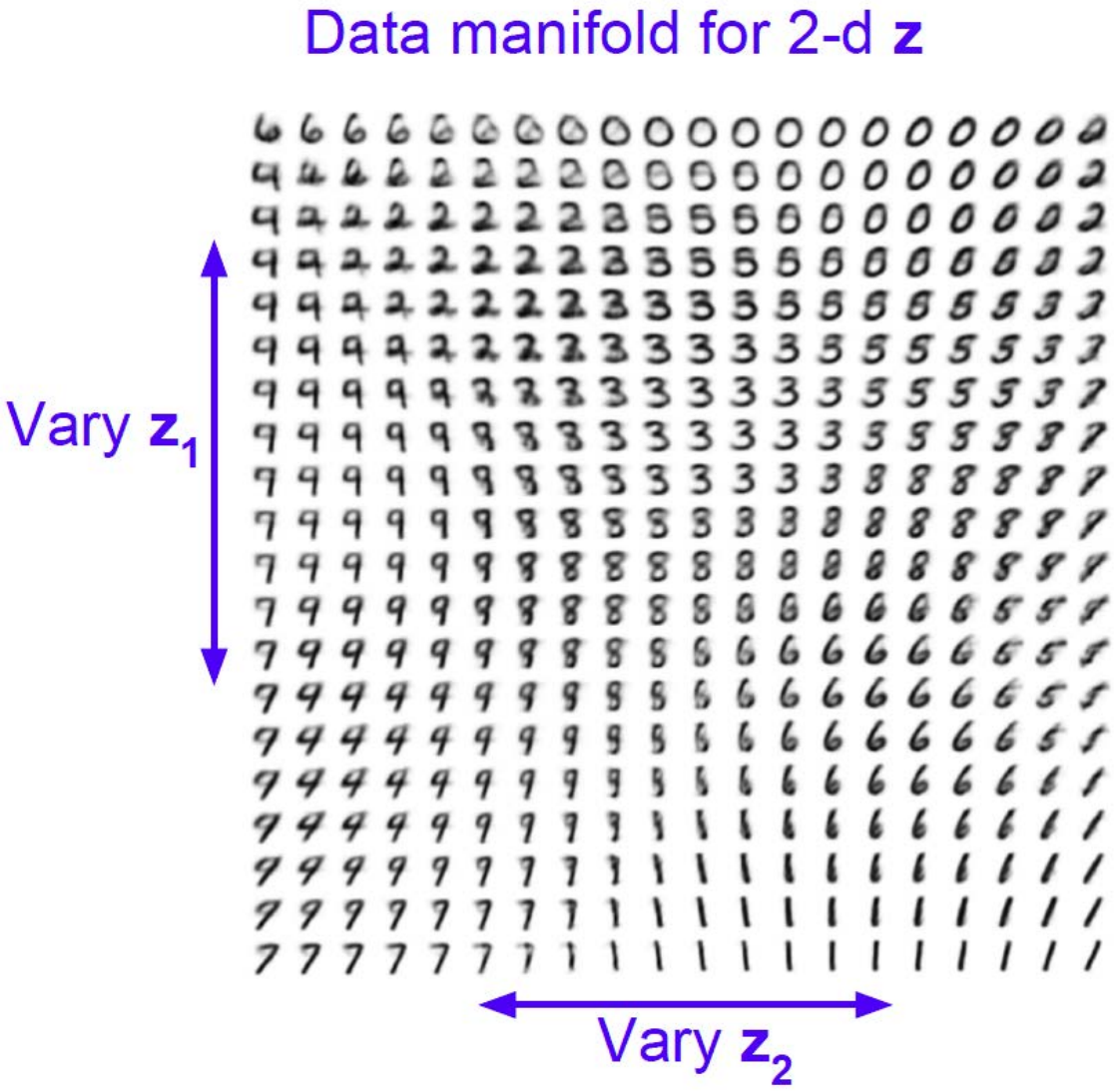
Variational Autoencoders – Generating Data

- Sample z from prior, then apply decoder

Decoder network
$p_\theta(x|z)$



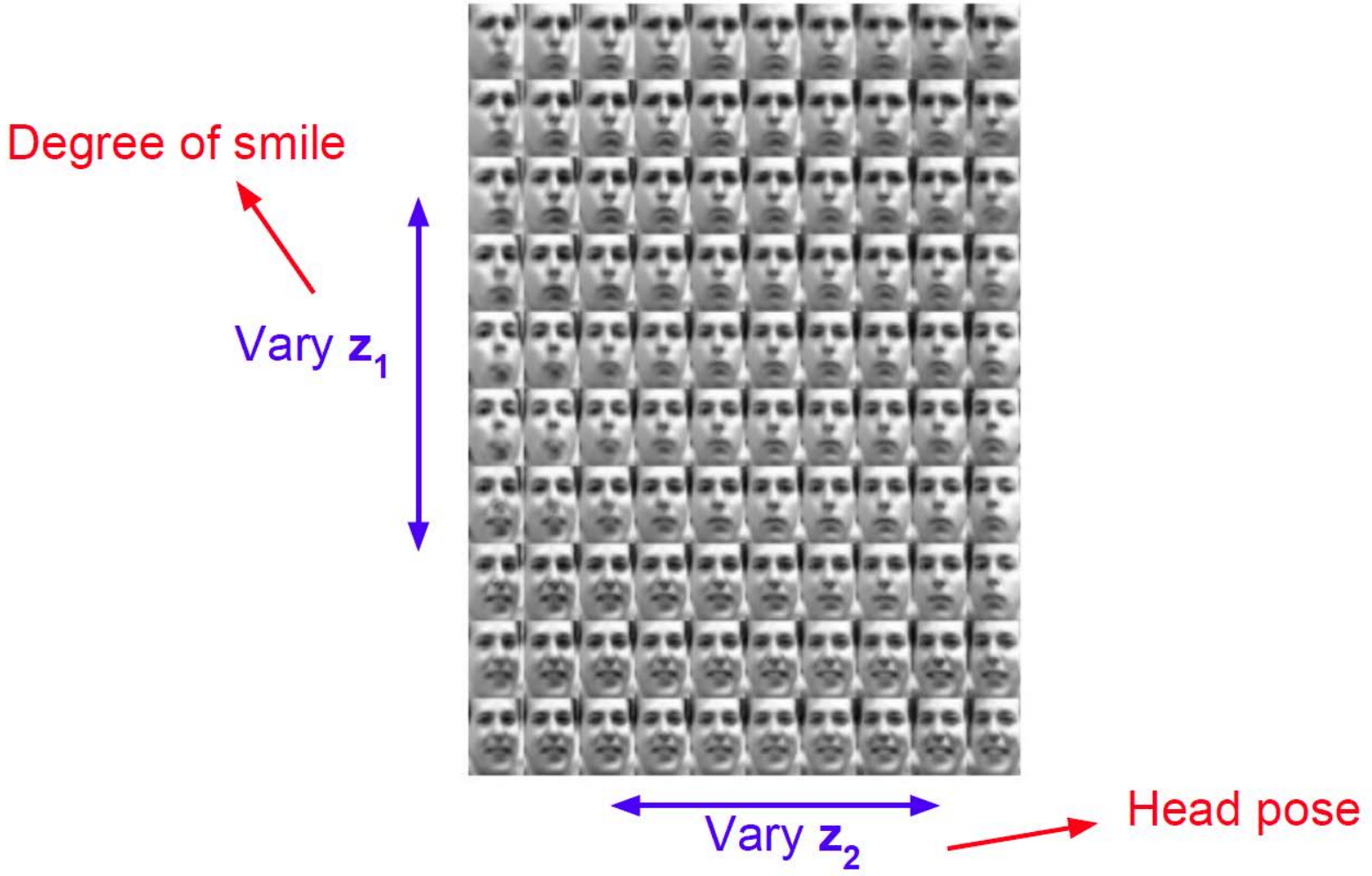
Sample z from $\mathcal{N}(0,1)$



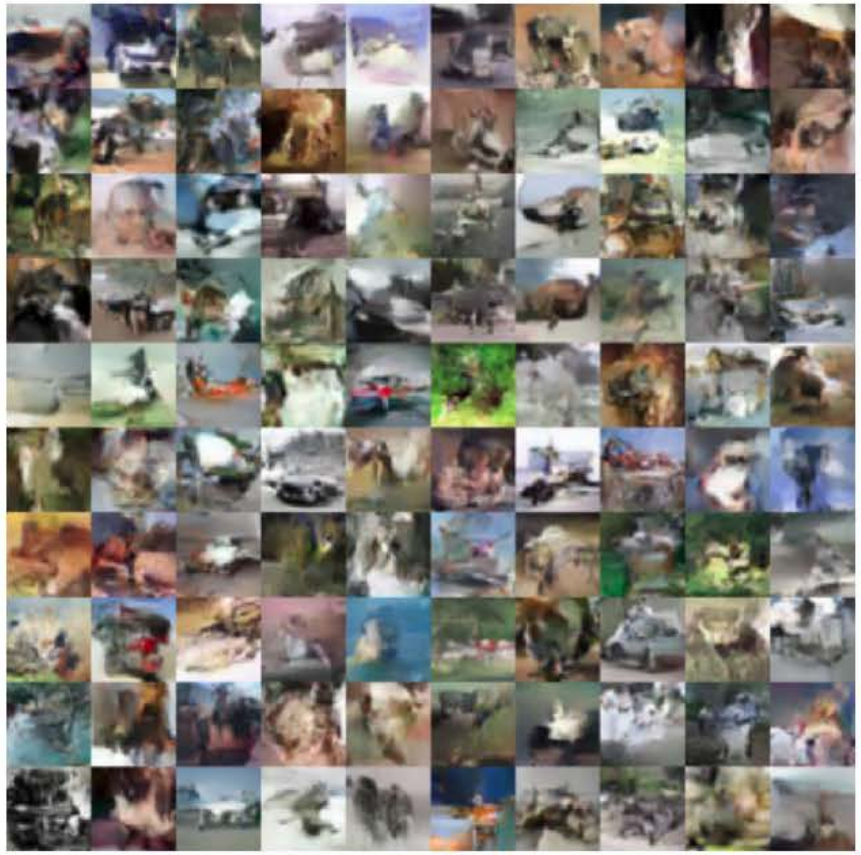
VAE - Results

- Diagonal prior on $z \rightarrow$ independent latent variables

- Different dimensions of z encode interpretable factors of variation



VAE - Results



32x32 CIFAR-10

Labeled Faces in the Wild

Discussion

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Probabilistic spin to traditional autoencoders → allows generating data
- Defines an intractable density → derive and optimize a (variational) lower bound

- **Pros:**
  - Principled approach to generative models
  - Allows inference of q(z|x), can be useful feature representation for other tasks

- **Cons:**
  - Maximizes lower bound of likelihood
  - Samples blurrier and lower quality compared to state-of-the-art (GANs)

- **Active areas of research:**
  - More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
  - Incorporating structure in latent variables

References

- M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991
- A Morphable Model for the Synthesis of 3D Faces. Volker Blanz, Thomas Vetter. ACM SIGGRAPH 1999
- K-SVD: DESIGN OF DICTIONARIES FOR SPARSE REPRESENTATION. *Michal Aharon Michael Elad Alfred Bruckstein,* Proceedings of SPARS, 2005
- TensorTextures: multilinear image-based rendering. M. Alex O. Vasilescu, Demitri Terzopoulos, ACM SIGGRAPH 2004.
- Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
- Fei-Fei Li & Justin Johnson & Serena Yeung, Generative Models - CS231n, Stanford Slides