Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS7GV1 Computer vision

PCA

Dr. Martin Alain

Introduction

We have seen that images can be reconstructed efficiently with wavelets.

We introduce Principal Component Analysis (PCA) as a way to learn a basis of functions (~ learnt DCT wavelets!).

PCA is also associated with the whitening operation

https://en.wikipedia.org/wiki/Whitening_transformation

and "batch normalisation" in CNN

Mean and Covariance of a set of vectors I

Consider that we have a set of vectors $\{\mathbf{x}_i\}_{i=1\cdots N}$ in $\mathbb{R}^d$. We can define

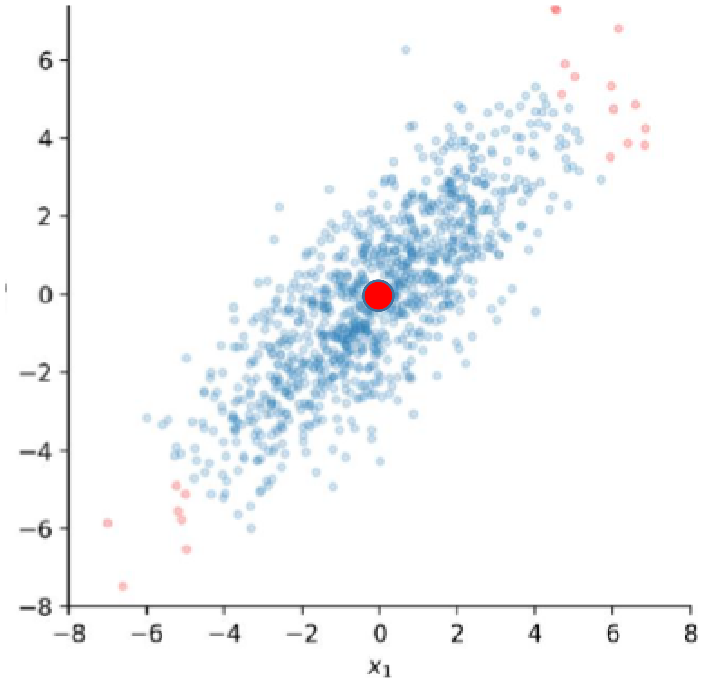
- the mean $\overline{\mathbf{x}}$ such that

$$\overline{\mathbf{x}} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N}$$

  spatially the mean can be understood as the center of gravity of the clouds of points $\{\mathbf{x}_i\}_{i=1\cdots N}$.

- the covariance matrix:

$$\mathrm{C} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$



https://en.wikipedia.org/wiki/Multivariate_normal_distribution

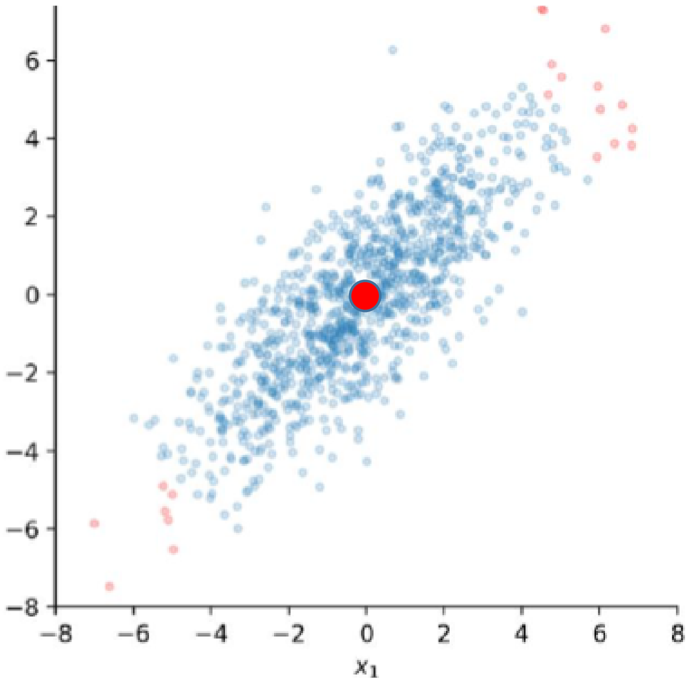
Mean and Covariance of a set of vectors II

with defining $\tilde{\mathbf{x}}_i = \mathbf{x} - \overline{\mathbf{x}}, \; \forall i$ then

$$\mathrm{C} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$$

or

$$\mathrm{C} = \frac{1}{N} \begin{bmatrix} \sum_{i=1}^{N} \tilde{x}_{1i}^2 & \sum_{i=1}^{N} \tilde{x}_{1i}\tilde{x}_{2i} & \cdots & \sum_{i=1}^{N} \tilde{x}_{1i}\tilde{x}_{di} \\ \sum_{i=1}^{N} \tilde{x}_{1i}\tilde{x}_{2i} & \sum_{i=1}^{N} \tilde{x}_{2i}^2 & & \\ & & \ddots & \\ & & \ddots & \\ & & & \sum_{i=1}^{N} \tilde{x}_{di}^2 \end{bmatrix}$$



The Lagrangian I

Definition

We consider the optimization problem:

$$\begin{array}{lll} \text{minimize} & f_0(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) = 0 & i = 1, \cdots, m \\ & h_j(\mathbf{x}) \leq 0 & j = 1, \cdots, p \end{array}$$

with $\mathbf{x} \in \mathbf{R}^d$.

The Lagrangian $\mathscr{L} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p$ associated with the problem is defined as:

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{j=1}^{p} \nu_j h_j(\mathbf{x})$$

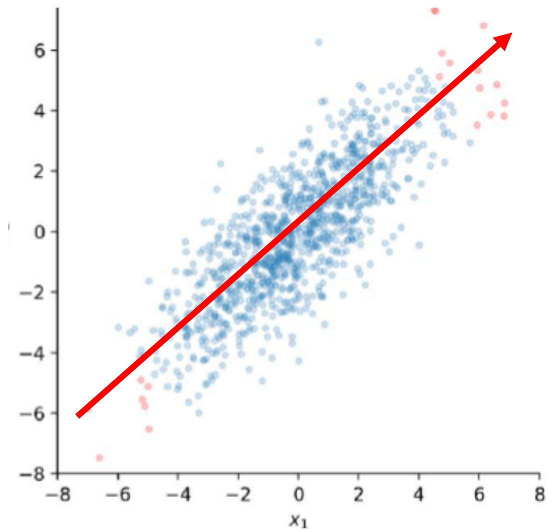
The vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are called the Lagrange multiplier vectors.

The Lagrangian III

We are mainly interested in minimizing functions with equality constraints.
Differentiating the Lagrangian $\mathscr{L}(\mathbf{x}, \boldsymbol{\lambda})$ in $\mathbf{x}$ gives d equations and differentiating the Lagrangian $\mathscr{L}(\mathbf{x}, \boldsymbol{\lambda})$ in $\boldsymbol{\lambda}$ gives m equations.
Solving the optimization problem then become solving

$$\begin{cases} \frac{\partial \mathscr{L}}{\partial \mathbf{x}} = 0 \\ \frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}} = 0 \end{cases}$$

Determining Principal components I



Consider that we have a set of vectors $\{\mathbf{x}_i\}_{i=1 \cdots N}$ in $\mathbb{R}^d$ arranged in a matrix X:

$$\mathrm{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ \vdots & & & \vdots \\ x_{d,1} & x_{d,2} & \cdots & x_{d,N} \end{bmatrix}$$

we are looking in a direction or vector $\mathbf{v} \in \mathbb{R}^d$ such that the projections of $\{\mathbf{x}_i\}_{i=1 \cdots N}$ on $\mathbf{v}$ leads to the scatter of N points with the highest dispersion.

Determining Principal components II

v: unit vector

- The projection of $\mathbf{x}_i$ on to $\mathbf{v}$ is $\mathbf{v}\mathbf{v}^T\mathbf{x}_i$.

- The distance between two projections is

$$\begin{aligned}\|\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{v}\mathbf{v}^T\mathbf{x}_j\|^2 &= (\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{v}\mathbf{v}^T\mathbf{x}_j)^T(\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{v}\mathbf{v}^T\mathbf{x}_j) \\ &= \mathbf{v}^T(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{v}\end{aligned}$$

- Considering all the vectors $\{\mathbf{x}_i\}_{i=1\cdots N}$, the criterion to be maximized is:

$$\begin{aligned}\mathscr{J}(\mathbf{v}) &= \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{v}^T(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{v} \\ &= \mathbf{v}^T\mathrm{C}\mathbf{v}\end{aligned}$$

Determining Principal components III

so the problem can be summarized as finding $\mathbf{v}$ such that:

$$\begin{cases} \max_{\mathbf{v}} \mathscr{J}(\mathbf{v}) \\ \text{subject to } \ \mathbf{v}^T\mathbf{v} = 1 \end{cases}$$

Using the Lagrange multipliers, the equivalent problem is:

$$\max \mathscr{J}(\mathbf{v}) - \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

Determining Principal components IV

The solution is found by solving $\frac{\partial \mathscr{J}(\mathbf{v})}{\partial \mathbf{v}} = 0$ and $\frac{\partial \mathscr{J}(\mathbf{v})}{\partial \lambda} = 0$:

$$\begin{cases} \mathrm{C}\mathbf{v} - \lambda \mathbf{v} = 0 \\ \mathbf{v}^T \mathbf{v} = 1 \end{cases}$$

So $\mathbf{v}$ is an eigenvector of C, and $\mathscr{J}(\mathbf{v}) = \lambda$. Then to get the biggest dispersion we should choose the eigenvector associated with the highest eigenvalue. Hence the name of this method principal component analysis.

This result can be generalized: the k principal directions are the k eigendirections of the highest eigenvalues.

Determining Principal components V

Theorem (Principal Component Analysis)

From a set of vectors $\{\mathbf{x}_i\}$

1. *Compute the mean* $\overline{\mathbf{x}}$
2. *Center each observations* $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \overline{\mathbf{x}}$
3. *Compute the covariance matrix*

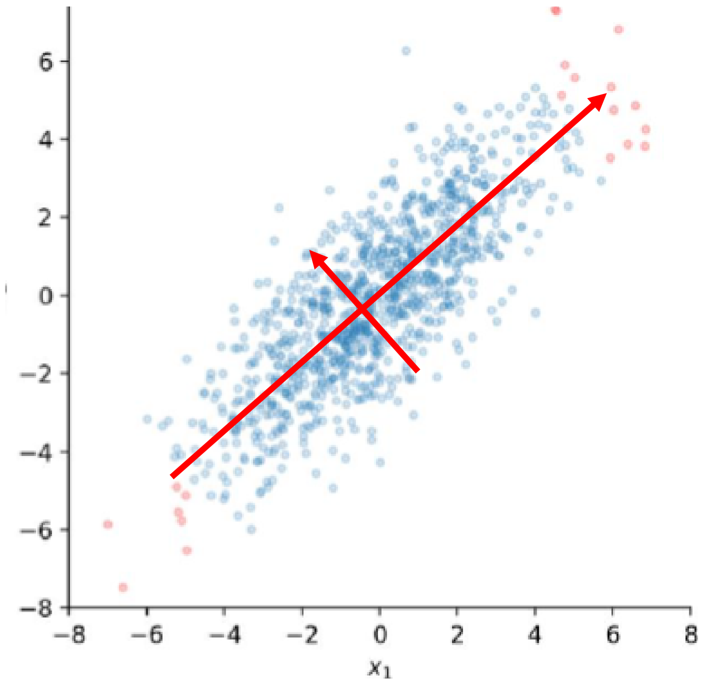
$$\mathrm{C} = \frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$$

4. *Compute the eigenvectors of* C *and sort them from the one associated with the highest eigenvalue , to the one associated with the lowest eigenvalue.*

Using PCA

Each vector in the set can be written as a linear combination of the mean and the eigenvectors:

$$\mathbf{x} = \overline{\mathbf{x}} + \sum_j \alpha_j \mathbf{v}_j$$

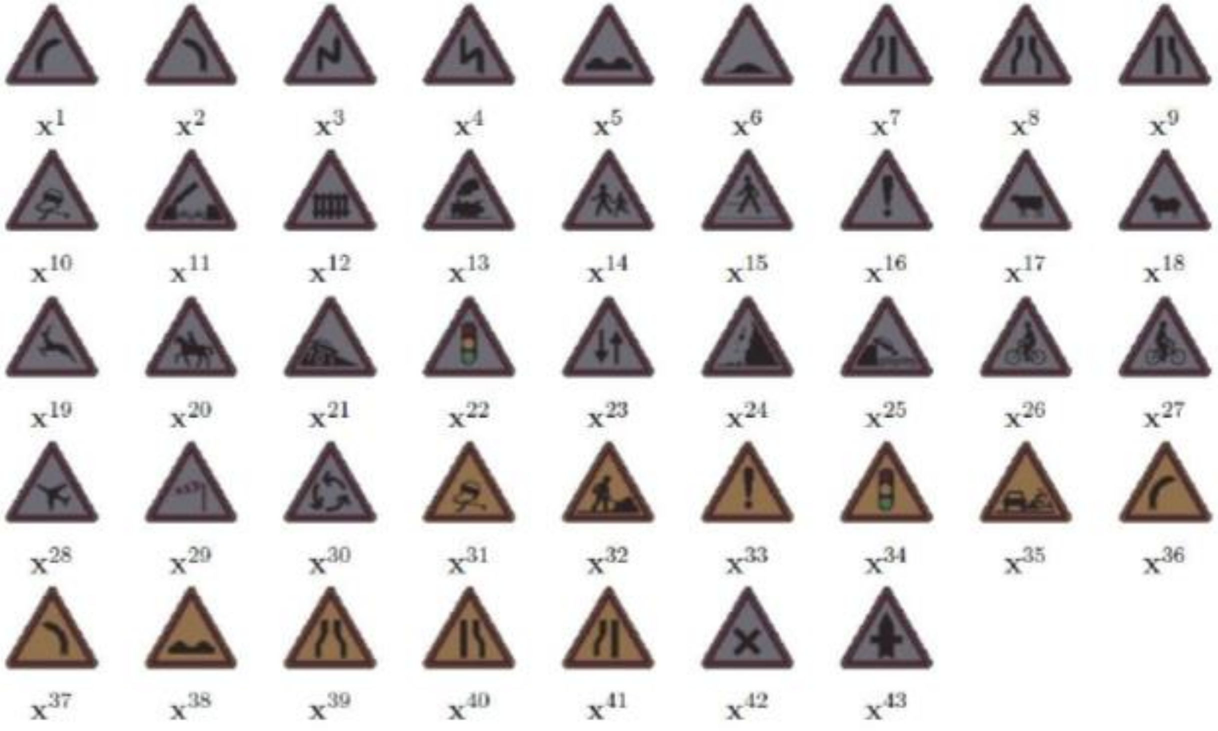


How many eigenvectors really needed?

Applications:

- Compression / Dimensionality reduction
- Visualisation of data distribution
- etc.

Training Dataset



vector embedding of image data

Pixels $p_1\ p_2 \ldots p_N$

$$\mathbf{x} = \begin{pmatrix} r_1 \\ g_1 \\ b_1 \\ \vdots \\ r_N \\ g_N \\ b_N \end{pmatrix}$$

Robust Visual Recognition of Colour Images
R. Dahyot at al, IEEE conference on Computer Vision and Pattern Recognition (CVPR'00), DOI:10.1109/CVPR.2000.855886

Training set with Data Augmentation

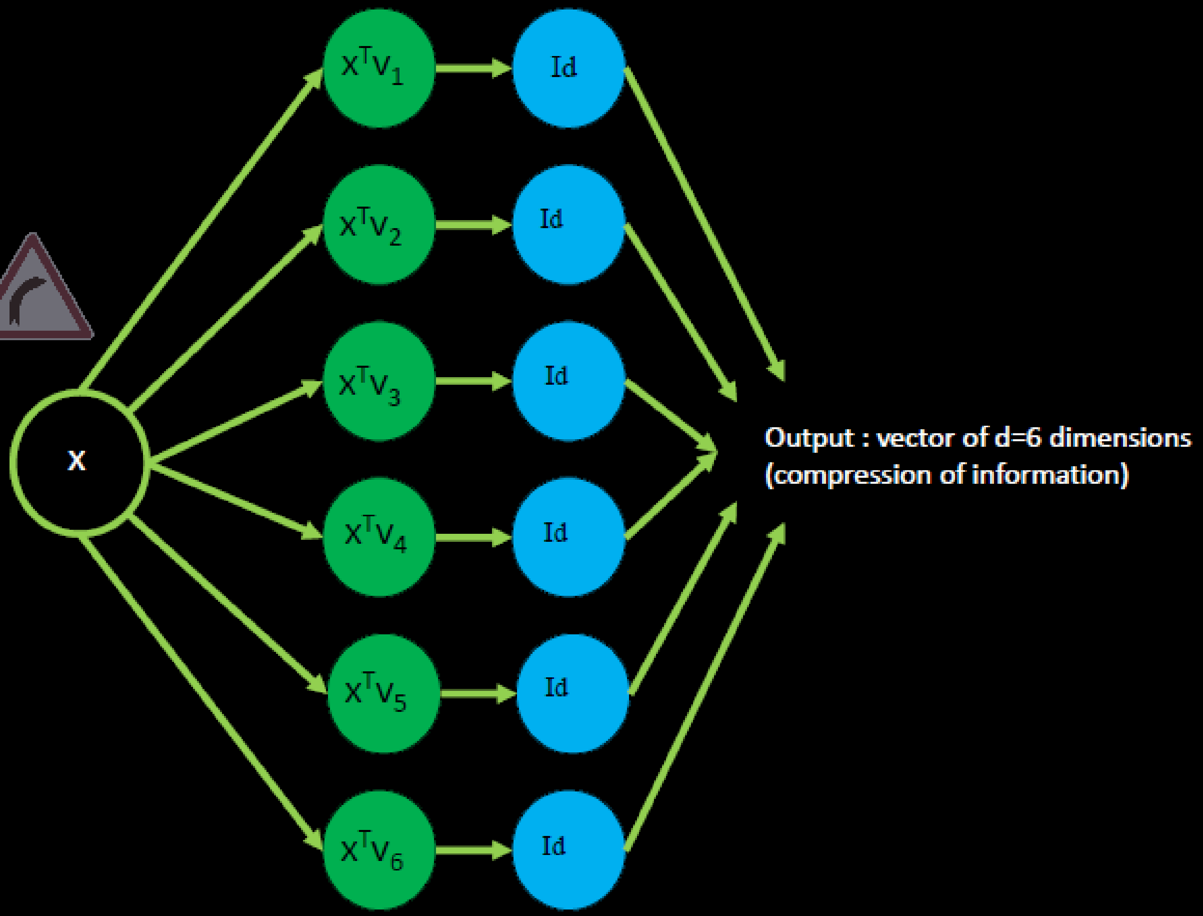
Example data augmentation with rotation

[Robust Visual Recognition of Colour Images](#)
R. Dahyot at al, IEEE conference on Computer Vision and Pattern Recognition (CVPR'00), DOI:10.1109/CVPR.2000.855886

Training dataset can be augmented by generated imaged, e.g. using geometric transformation

https://www.tensorflow.org/tutorials/images/data_augmentation
https://en.wikipedia.org/wiki/Data_augmentation

Learning the Principal Components

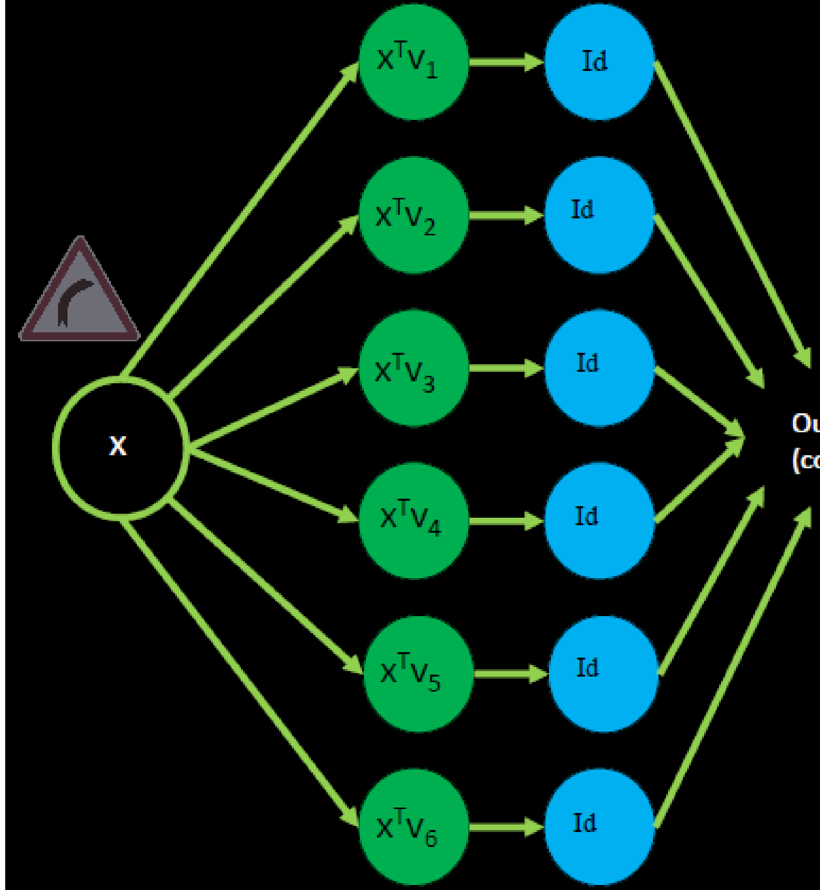


$$\begin{aligned}\mathscr{J}(\mathbf{v}) &= \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{v}^T(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{v}\\ &= \mathbf{v}^T C\mathbf{v}\end{aligned}$$

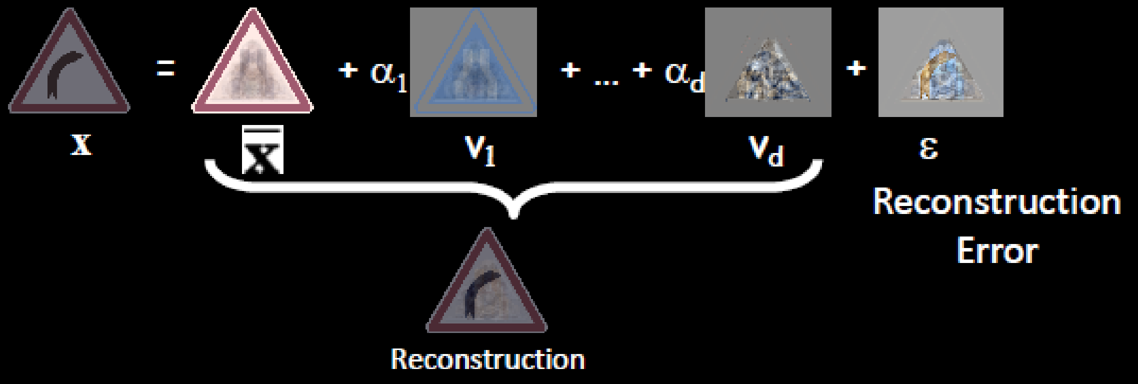
$$\begin{cases} \max_{\mathbf{v}} \mathscr{J}(\mathbf{v}) \\ \text{subject to} \quad \mathbf{v}^T\mathbf{v} = 1 \end{cases}$$

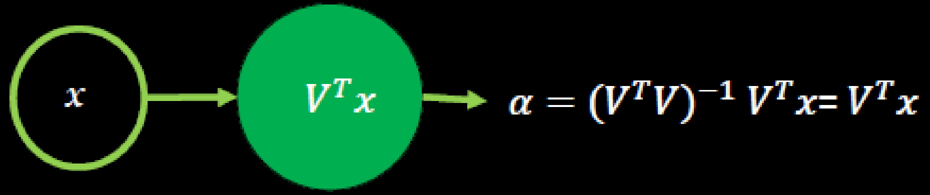
- **Projection** with Basis of learnt vectors $\{v_j\}$. Learning is performed as a solution for minimising a cost function *J* defined the training dataset (with some constraints)
- **Activation** function is chosen as the identity function.

Compression with PCA



Output : vector of d=6 dimensions $\alpha_1, \ldots, \alpha_d$
(compression of information)





<u>No cost function to optimise</u> at <u>inference time</u>

Eigenvalues of principal components

Fig. 10. The first eight eigenfaces.

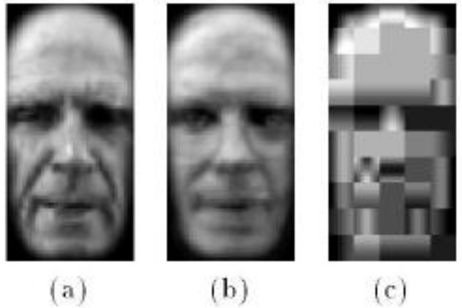
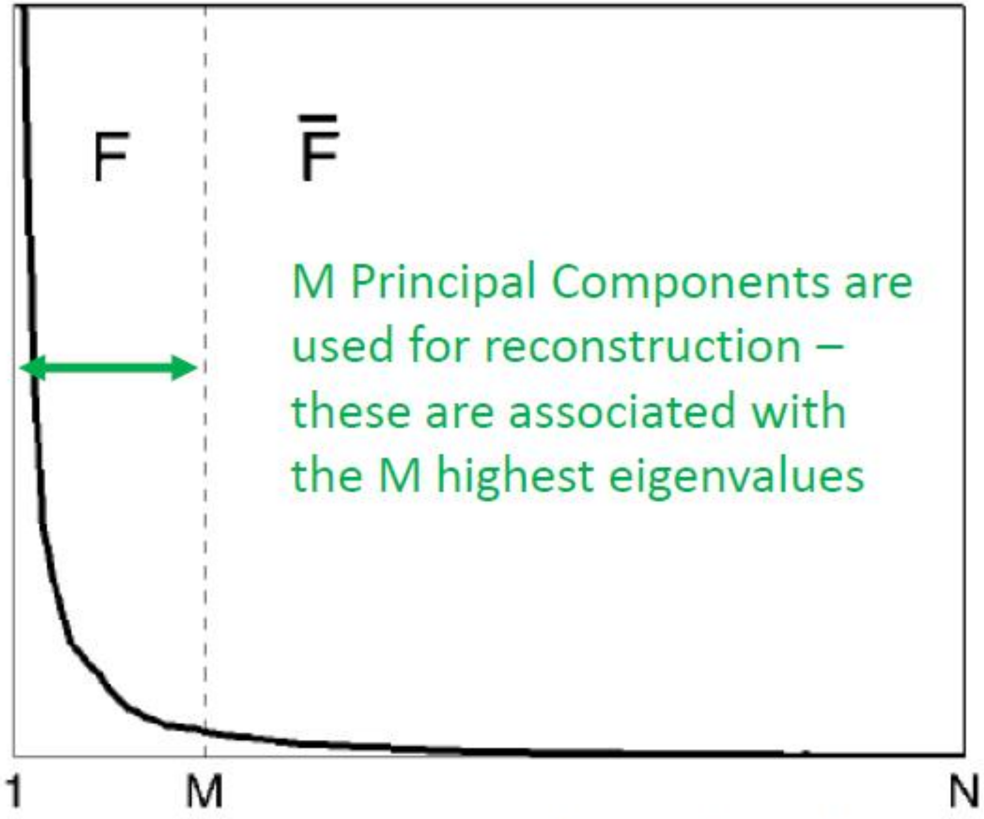


Figure 9: *(a) aligned face, (b) eigenspace reconstruction (85 bytes) (c) JPEG reconstruction (530 bytes).*

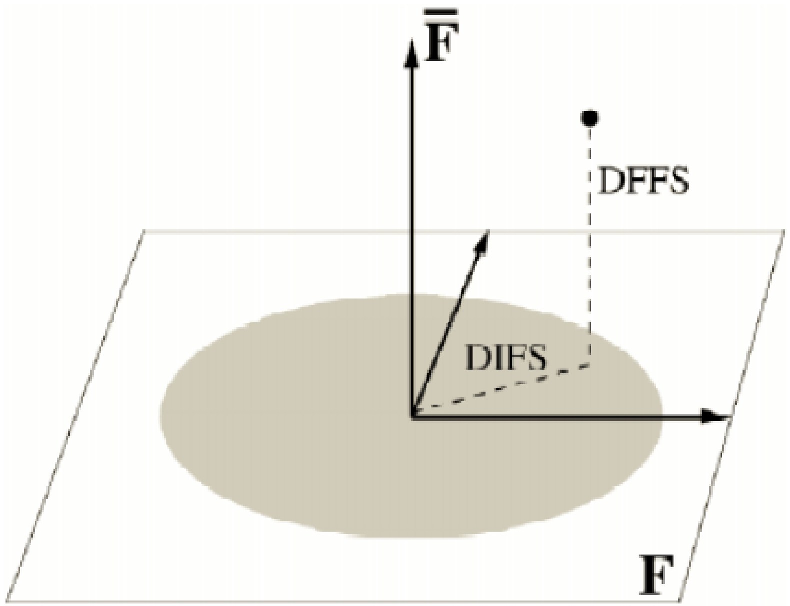
Eigenvalue spectrum obtained by PCA



Notation: N=dim(x)

Probabilistic visual learning for object representation, IEEE Trans. Pattern Analysis and Machine Intelligence (1997), DOI: 10.1109/34.598227

Modelling P(x) using PCA

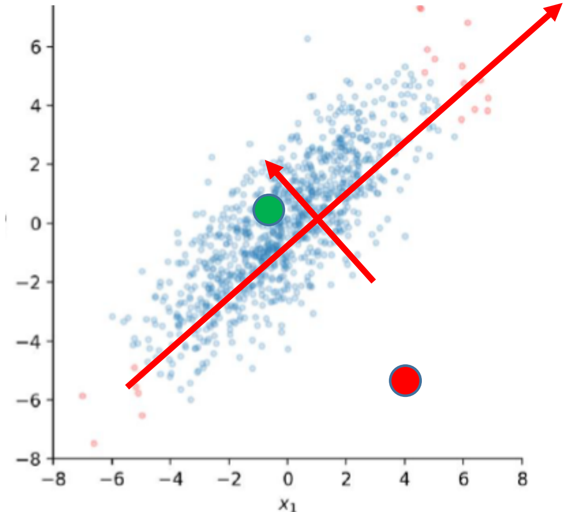


(a)

<u>Notation</u>: $y_i = x^T v_i$
projection on principal component v_i
associated with eigenvalue λ_i

$$\hat{P}(\mathbf{x}\,|\,\Omega) = \left[\frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i}\right)}{(2\pi)^{M/2}\prod_{i=1}^{M}\lambda_i^{1/2}}\right]\cdot\left[\frac{\exp\left(-\frac{\epsilon^2(\mathbf{x})}{2\rho}\right)}{(2\pi\rho)^{(N-M)/2}}\right]$$

$$= P_F(\mathbf{x}\,|\,\Omega)\,\hat{P}_{\bar{F}}(\mathbf{x}\,|\,\Omega)$$



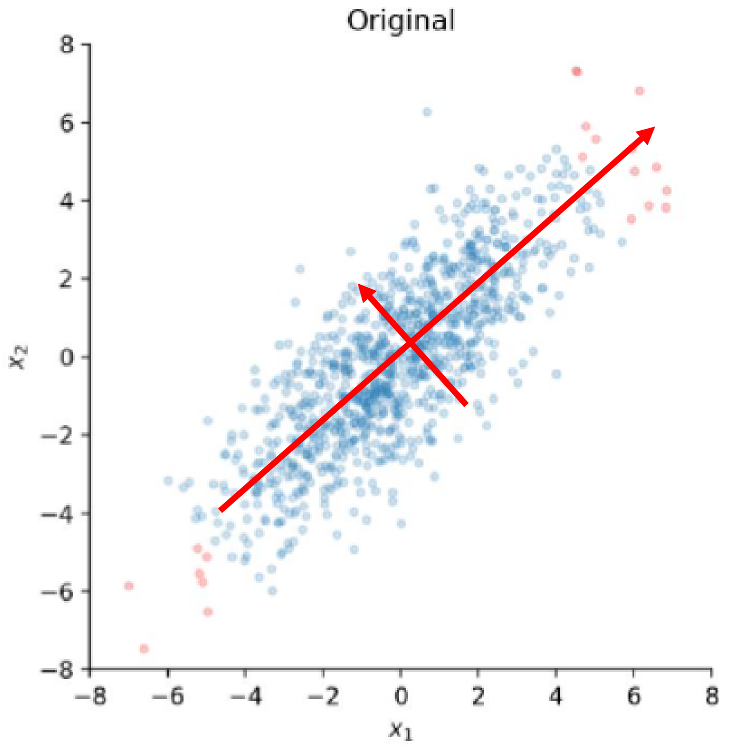
[Probabilistic visual learning for object representation](), IEEE Trans. Pattern Analysis and Machine Intelligence (1997), DOI: 10.1109/34.598227

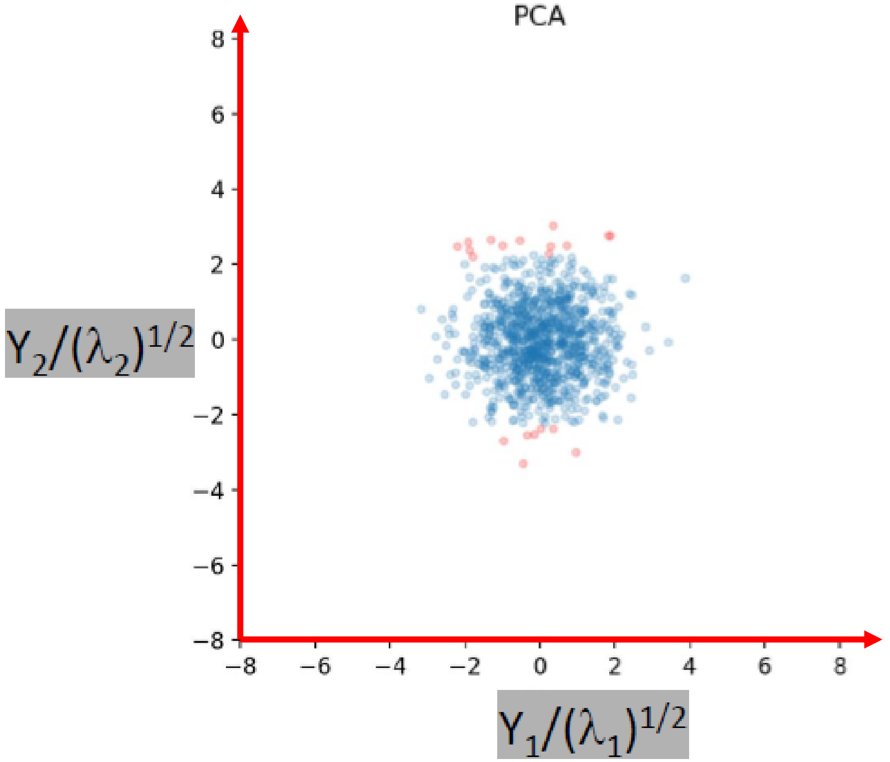
Whitening

https://cbrnr.github.io/2018/12/17/whitening-pca-zca/
https://en.wikipedia.org/wiki/Whitening_transformation

Notation: $y_i = x^T v_i$
projection on principal component v_i
associated with eigenvalue λ_i

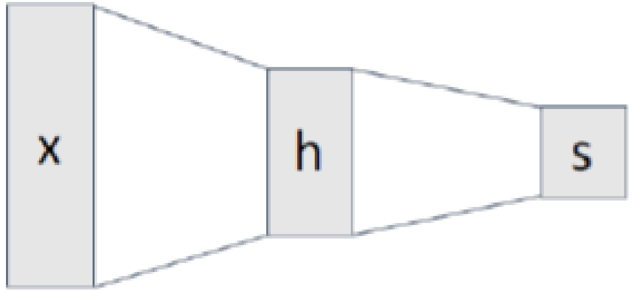
Whitening with PCA



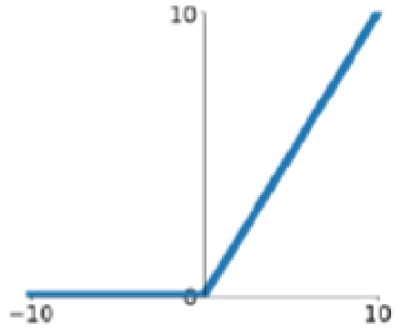


Components of a Convolutional Network

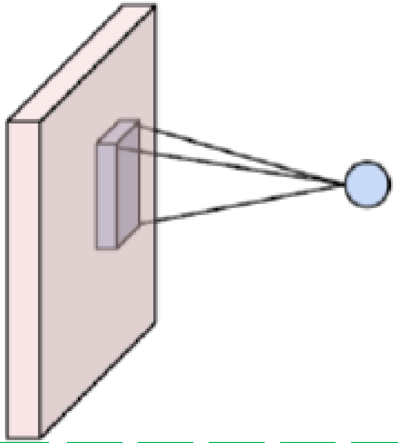
Fully-Connected Layers



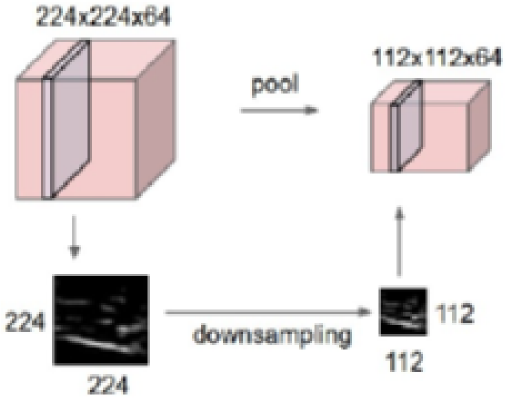
Activation Function



Convolution Layers



Pooling Layers



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Batch Normalization

Idea: "Normalize" the outputs of a layer so they have zero mean and unit variance

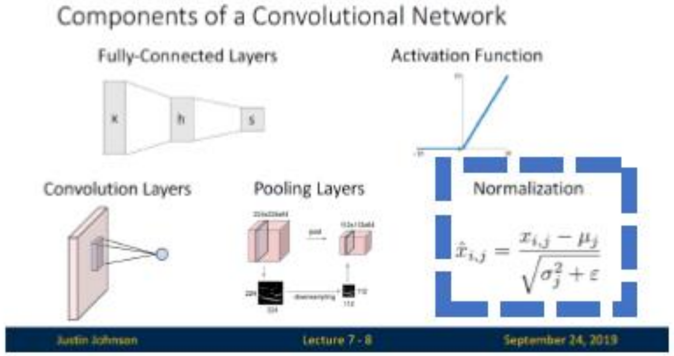
Why? Helps reduce "internal covariate shift", improves optimization

We can normalize a batch of activations like this:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

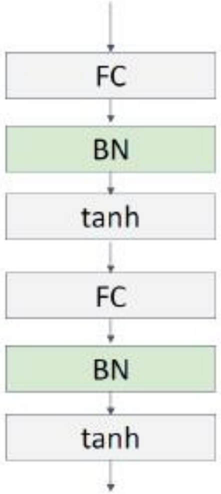
This is a **differentiable function**, so we can use it as an operator in our networks and backprop through it!

Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015



Lecture 7: Convolutional Networks (Michigan Online 2020)
https://youtu.be/ANyxBVxmdZ0

Batch Normalization



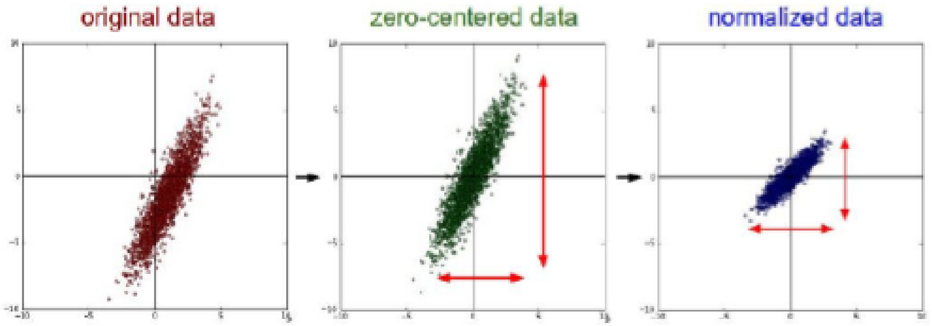
- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!
- Not well-understood theoretically (yet)
- Behaves differently during training and testing: this is a very common source of bugs!

Ioffe and Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", ICML 2015

Whitening http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture07.pdf

Decorrelated Batch Normalization

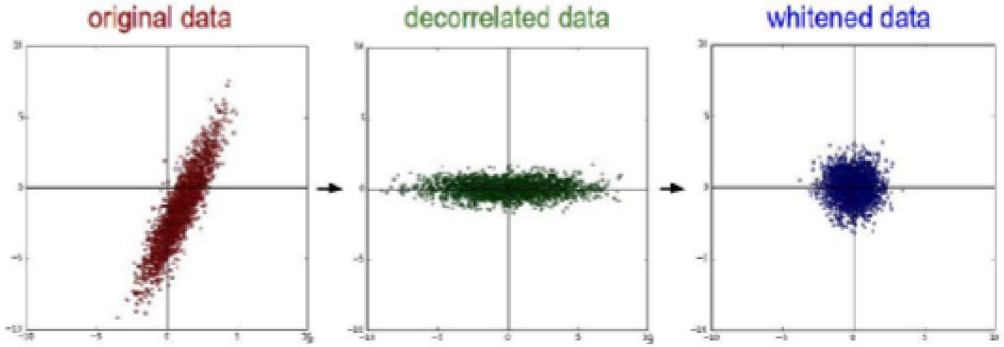
Batch Normalization



$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

BatchNorm normalizes the data, but cannot correct for correlations among the input features

Decorrelated Batch Normalization



$$\hat{x}_i = \Sigma^{-\frac{1}{2}}(x_i - \mu)$$

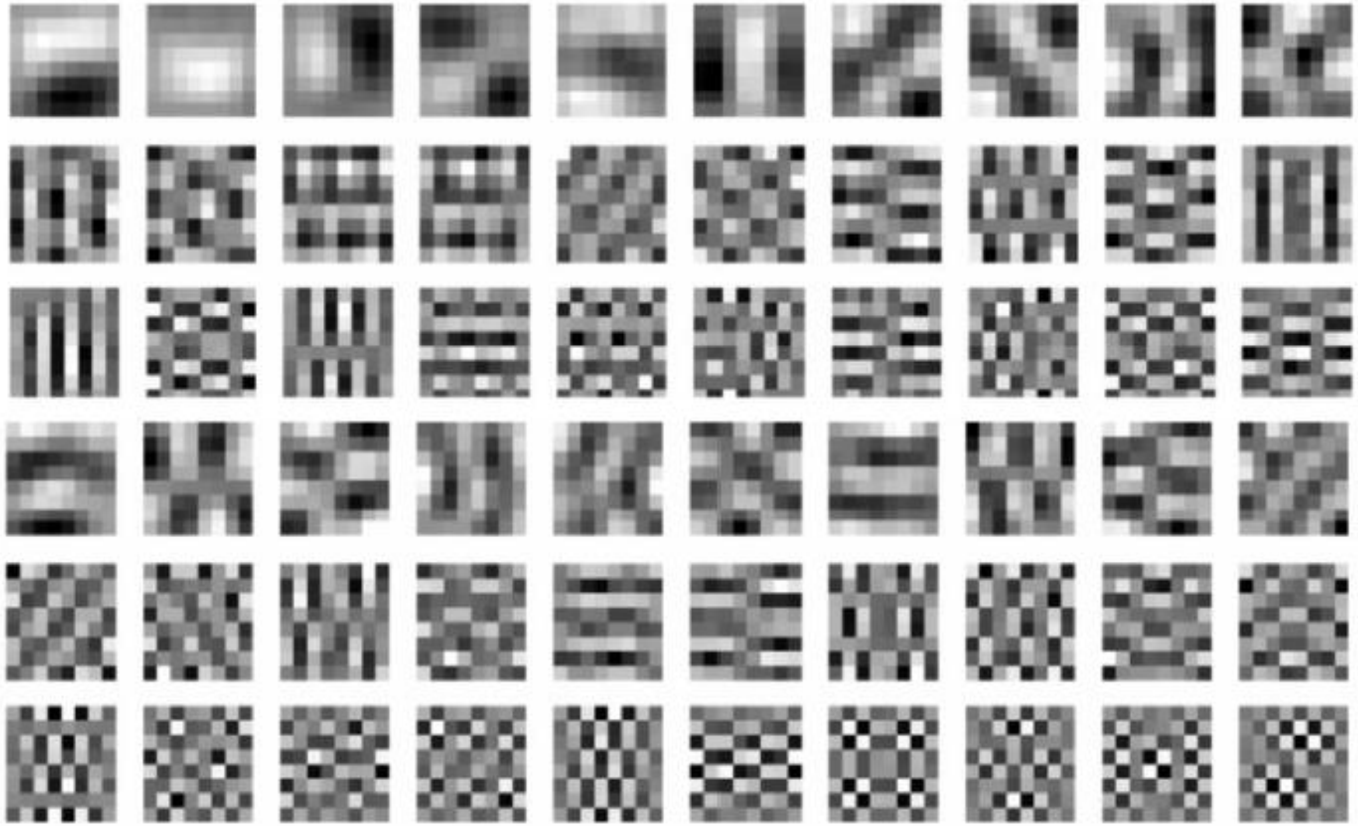
DBN **whitens** the data using the full covariance matrix of the minibatch; this corrects for correlations

Huang et al, "Decorrelated Batch Normalization", arXiv 2018 (Appeared 4/23/2018)

PCA & DCT Wavelets

https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture18-pca.pdf

60 most important eigenvectors



Looks like the discrete cosine bases of JPG!...

DCT in Neural networks:

[Harmonic Networks for Image Classification](#)
M. Ulicny, V. Krylov and R. Dahyot, British Machine Vision Conference (BMVC) 2019.

Harmonic Networks with Limited Training Samples
M. Ulicny, V. Krylov and R. Dahyot, European Signal Processing Conference (Eusipco) 2019.

Summary

- PCA allows to have a learnt orthogonal basis of vectors to represent a set of vectors (images).
- PCA basis is learnt from a training dataset – In comparison Wavelet and Fourier basis are not learnt from a dataset!
- Hence PCA basis is a fine tuned basis of vectors tailored to the training dataset
- Similarly CNN will be learning its own filters/kernel for convolutions from a training dataset
- The whitening transformation used with PCA is also a type of operation used in CNN/NN (== “batch normalisation”)