BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Learning from Images

Training DL Best Practice
Autoencoder

Master DataScience
Winter term 2019/20

Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

Syllabus

Date	Lecture topic	Assignment due date
10.10.19	Introduction / Quiz	
17.10.19	A shortcut through image processing	
24.10.19	Features and Feature Matching	
31.10.19	Image retrieval	**Assignment 1**
07.11.19	Image classification Features + SVM	
14.11.19	Neural Network Recap	**Assignment 2**
21.11.19	ConvNets, ConvNet Architectures	
28.11.19	Object recognition (YOLO, MobileNet)	
05.12.19	Auto-encoder and embeddings	**Assignment 3**
12.12.19	Generative Adversarial Networks	
20.12.19	**Project idea presentation**	
09.01.20	Generative Model Architectures	**Assignment 4**
16.01.20	Reinforcement learning on images	
23.01.20	Recent research development in CV	
31.01.20	**Project presentation**	
12.02.20	**Documentation due**	

Image Basics / Image features

Supervised Image Classification

Unsupervised & Generative Methods

Reinforcement Learning

Best Practice

Based on http://karpathy.github.io/2019/04/25/recipe/

Step-by-Step

- **Problem:**
  - **Coding errors give direct feedback**
  - NN fail in many many silent ways
  - **A “fast and furious” approach to training neural networks does not work!**

- **Step 1:** Understand your data!
  - Look through it: Find duplicates, corrupted images, understand their distribution
  - Questions:
    - Are very local features enough / do we need global context?
    - How much variation is there and what form does it take?
    - What variation is wrong could/should be taken out?
    - Does spatial position matter or do we want to average pool it out?
    - How much does detail matter and how far could we afford to downsample the images?
    - How noisy are the labels?

Step 2: Setup very basic training/evaluation pipeline

Now suffering begins

- Implement very simple model, e.g. linear classifier, tiny ConvNet
- Now train, visualize losses, accuracy, model predictions
- **Use fix random seed** – guarantees same outcome

- **Simplify**
  - **no fanciness, no regularization (e.g. dropout, data augmentation, regularization terms)**

- **Evaluate over complete test set not just batch**
  - **helps ensuring bug free**

- **Verify loss and init**
  - **E.g. softmax loss should be** -log(1/n_classes)

Xavier / He initialization

- He et al. (2015) proposed activation aware initialization of weights (for ReLu and leaky ReLU)

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{2/(n^{l-1})}$$

- Xavier initialization is standard heuristic method (tanh)

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{1/(n^{l-1})}$$

- Other commonly used initialization:

$$W^l = randn(n^l, n^{l-1}) \cdot \sqrt{2/(n^{l-1} + n^l)}$$

Step 2: Setup very basic training/evaluation pipeline

- Be your human baseline

- **Input-indepent baseline, e.g. zero valued input should perform worse then when we use data**

- **Overfit one batch**
  - increase the capacity of our model (e.g. add layers or filters)
  - verify to reach lowest achievable loss (e.g. zero)

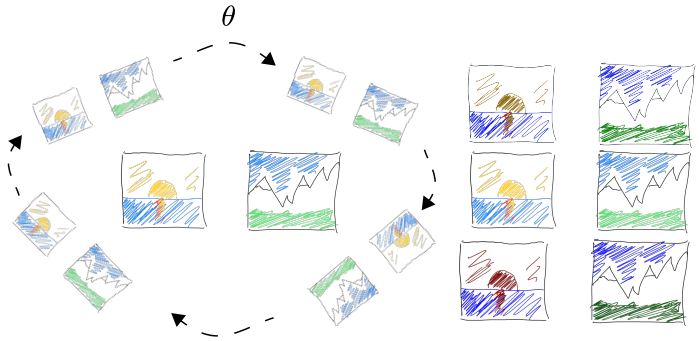
- **verify decreasing training loss**
  - **Now we should underfit the data – toy model**
  - **Increase capacity and loss should go down**

Step 3: Overfit

- Train a model large enough that it can overfit (i.e. focus on training loss)
- Regularize to improve test loss
- find the most related paper and copy paste their simplest architecture that achieves good performance
- Use Adam as optimizer with standard learning rate
- **complexify only one at a time e.g. use smaller images in the beginning**

Step 4: Regularize

- Time to regularize and gain some validation accuracy by giving up some of the training accuracy
- Use pretrained model
- **Get more data**, e.g. data augmentation



- **Decrease the batch size** that will result in stronger regularization
- **Use dropout**
- **Early stopping**
  - Stop training based on your measured validation loss to catch your model just as it's about to overfit.

Step 5: Tune / Step 6: Squeeze out the juice

- Use random / grid search over hyperparameters
- Hyper-parameter optimization with bayesian methods

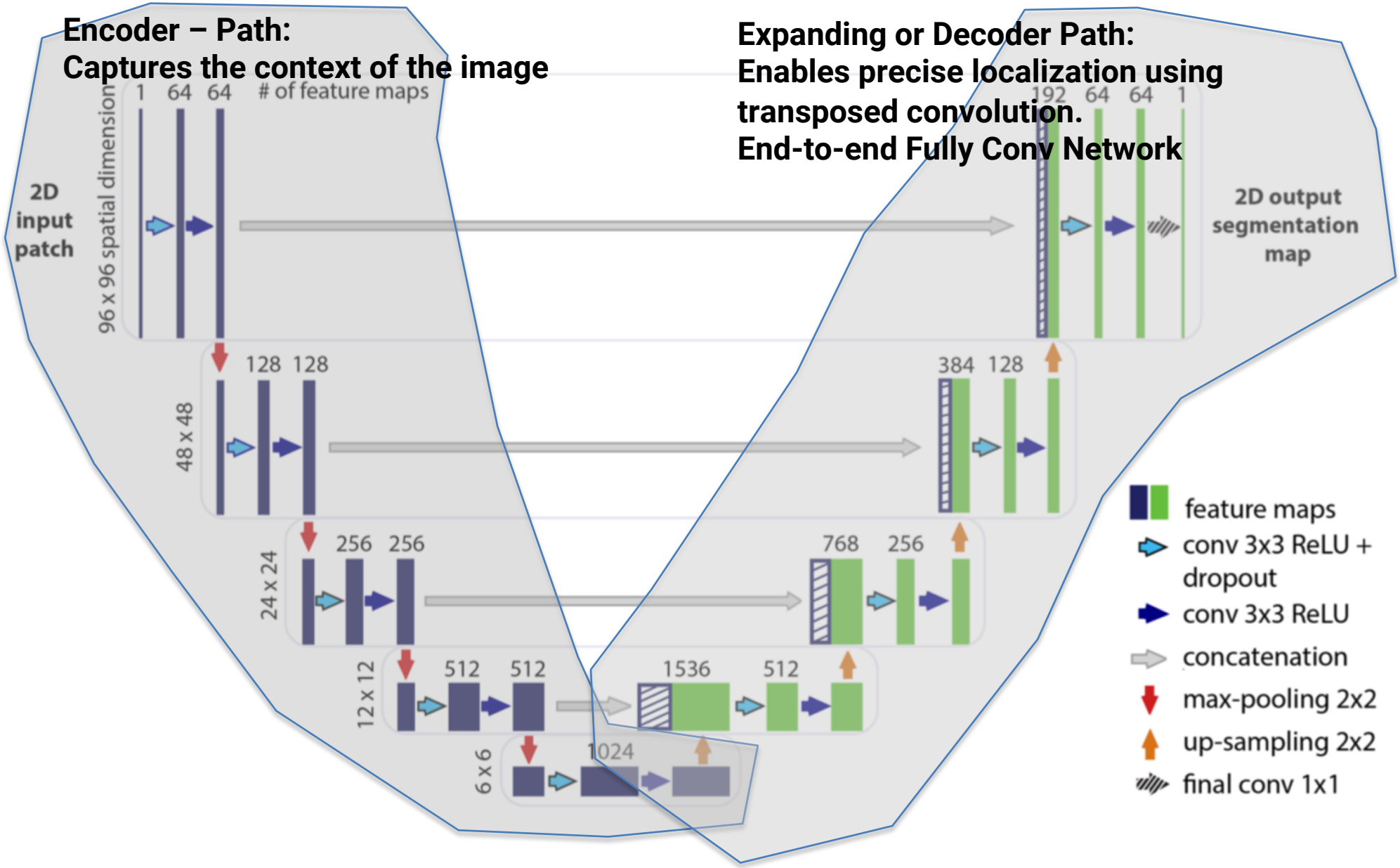
https://botorch.org/

- Use model ensembles
- Leave it training

What is an Autoencoder?

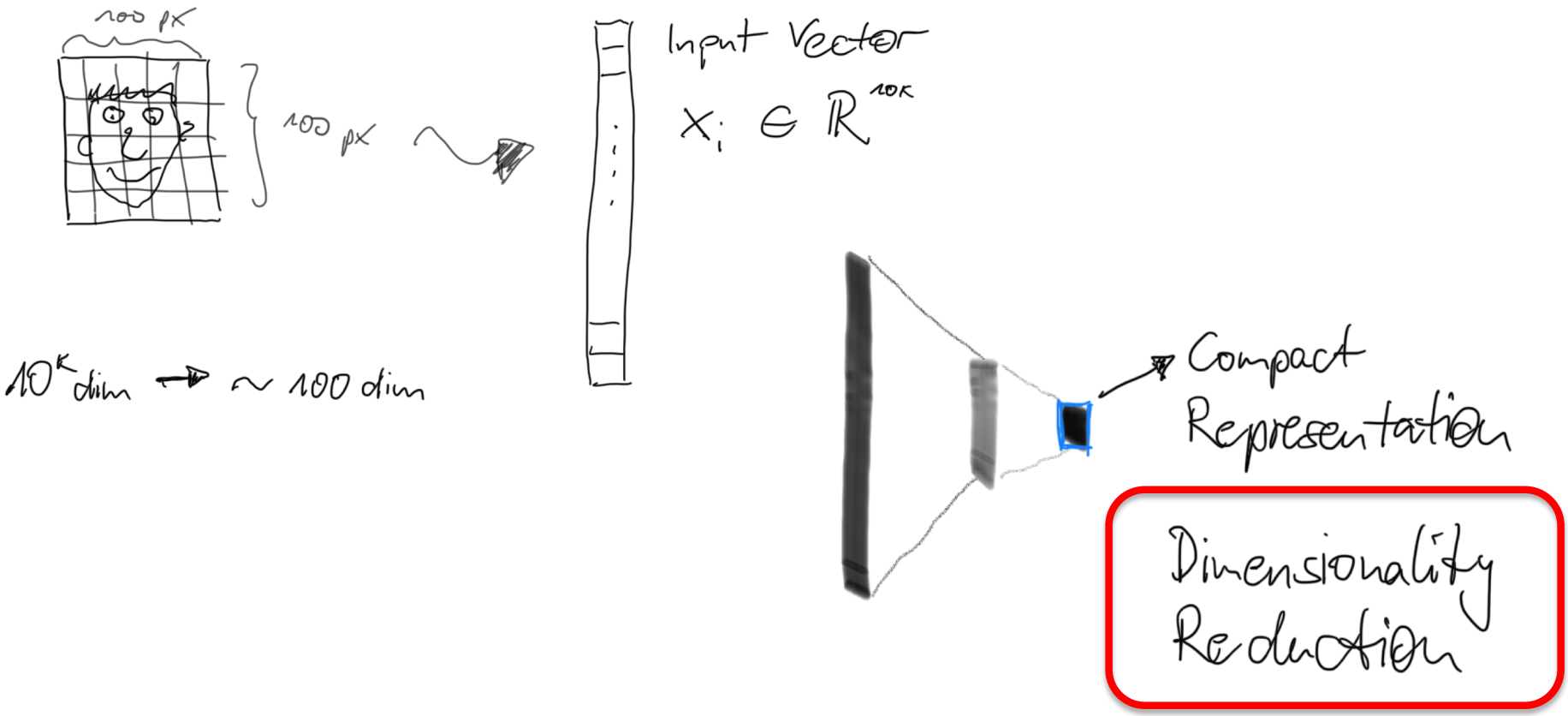
http://localhost:8888/notebooks/Beuth/Teaching/Courses/lfi/Notebooks/IRIS-AE_vs_PCA.ipynb

U-Net for image segmentation

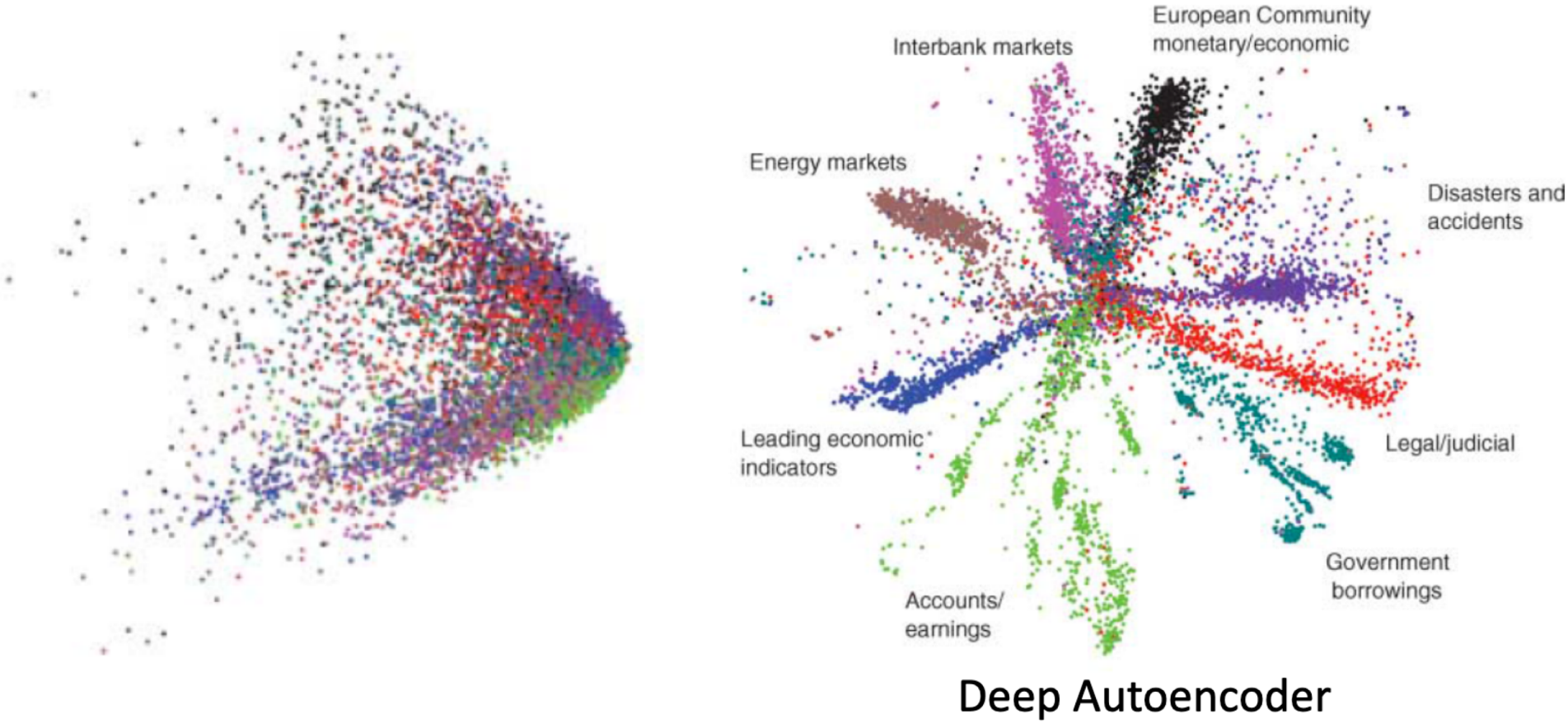


Autoencoder

- Neural network to learn an efficient and compact encoding of the underlying training data



Autoencoder



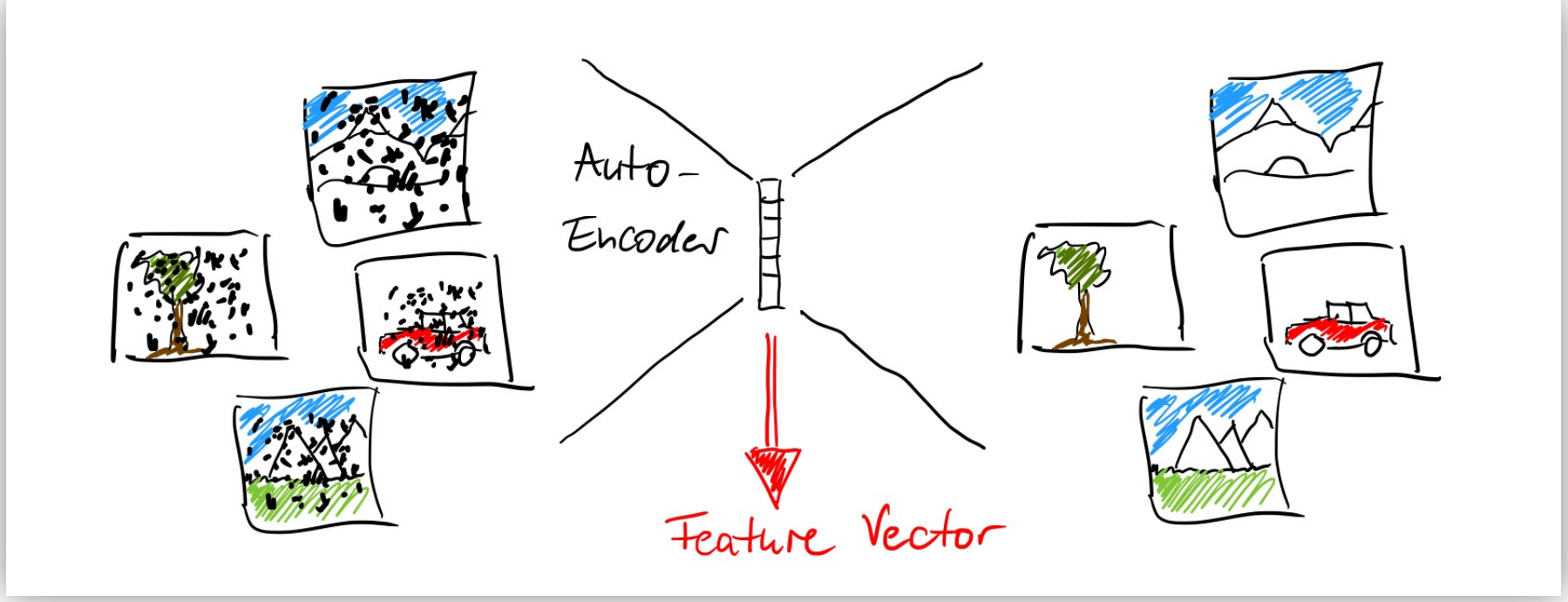
Deep Autoencoder

Source: Hinton et al., Reducing the Dimensionality of Data with Neural Networks

Lets talk about PCA and Autoencoder

Autoencoder – Architectures and Applications

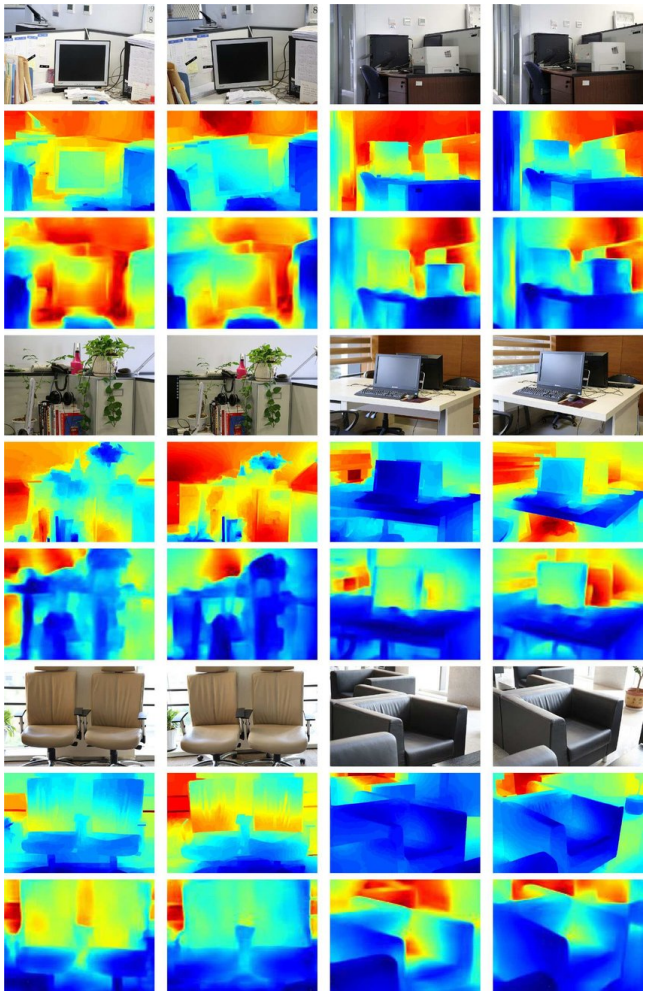
Feature Extraction / Image Retrieval etc.



Use Noise to learn stable representations

Autoencoder – Architectures and Applications

Depth Estimation



Human Pose Estimation

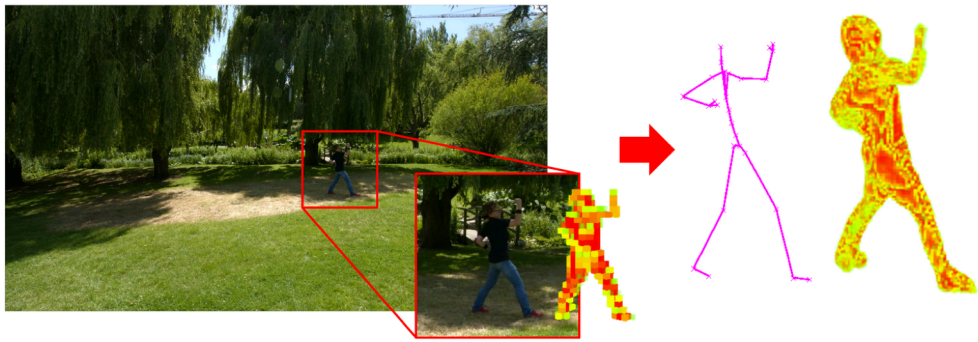


Image Completion