DUL: Autoencoders

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

June 23, 2025

Table of Contents

Table of Contents (cont.)

Motivation

- Real-world data is often **large**, noisy, and difficult to interpret.
- We aim to **compress** data efficiently while preserving essential information.
- It is also valuable for machines to **generate** new, realistic data samples.
- **Autoencoders** are a type of **unsupervised learning** model that address both compression and generation tasks.
- They **learn** to represent data in a **lower-dimensional space**, capturing its most important features.

⇒ Learning Outcomes

By the end of this session, you will be able to:

- Explain the basic concept and purpose of autoencoders
- Identify and describe the main components: **encoder**, **bottleneck**, and **decoder**
- Understand how autoencoders can be used to generate new data
- Recognize practical applications and limitations of autoencoders

Autoencoders

Autoencoders are a type of artificial neural network used for learning efficient codings of input data in an unsupervised manner. The goal is to learn a compressed (encoded) representation of the input and then reconstruct the original input from this compressed version.

- **Developed originally in the 1980s**, autoencoders have gained renewed popularity due to advances in deep learning and hardware.
- Often used in **dimensionality reduction, denoising, anomaly detection, and generative modeling**.

Autoencoders (cont.)

- Autoencoders are a family of neural networks designed to learn efficient representations of data, typically for dimensionality reduction or feature learning.
- They work by compressing the input into a latent-space representation (encoding), and then reconstructing the original input from this representation (decoding).
- The main goal is to project the input into a lower-dimensional latent space and then reconstruct the input from that latent representation.

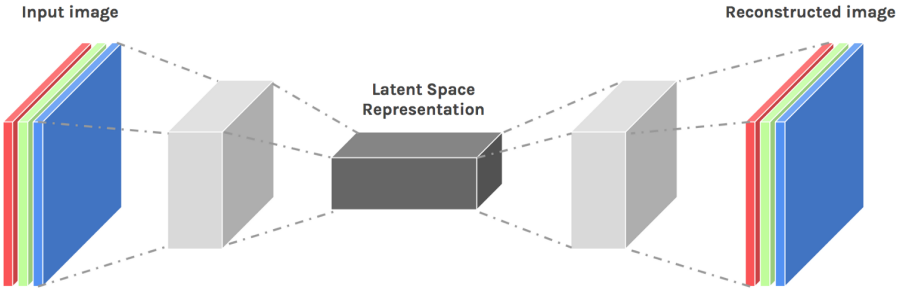
Autoencoders (cont.)

Autoencoders consist of two main components:

- **Encoder**: Maps the input data to a latent space Z, compressing the input into a lower-dimensional representation.
- **Decoder**: Reconstructs the input data from the encoded latent representation.

Variants of autoencoders may use altered or corrupted versions of the input as the target output, which can help the model learn more robust features (e.g., denoising autoencoders).

Autoencoders (cont.)



Autoencoder architecture

Autoencoders (cont.)

- **Bottleneck (Latent Space)**
  - The bottleneck is a small hidden layer in the middle of the network.
  - It forces the network to squeeze the input into a shorter, simpler code.
  - This helps the network find and keep only the most important information.
  - A small bottleneck stops the autoencoder from just memorizing the input.
- **Loss Function**
  - Tells us how close the output is to the original input.
  - The most common loss is **Mean Squared Error (MSE)**—it measures the average difference between input and output.
  - For special cases, we can use other losses, like binary cross-entropy for yes/no data, or KL divergence for special types of autoencoders.
  - The loss function guides the network to learn better and more useful features.

Design Considerations:

- **How deep?** More layers can learn more, but also make training harder and risk overfitting.
- **How wide?** More neurons per layer can help, but too many may just copy the input instead of learning.
- **Symmetry:** Encoder and decoder can look the same, but don't have to—sometimes different shapes work better.
- **Regularization:** Tricks like dropout or weight penalties help prevent the model from just memorizing.
- **Latent size:** The "bottleneck" size is key—too small loses info, too big doesn't compress.
- **Activation:** Functions like ReLU or sigmoid help the network learn complex patterns.

Autoencoders: Architecture (cont.)

Encoder:

- Takes input data (e.g., image, text, audio)
- Uses neural layers to reduce it to a smaller dimension
- Learns to keep only the important features
- *Example*: 784-pixel image to 32 numbers

Autoencoders: Architecture (cont.)

Latent Space (Bottleneck):

- This is the core representation of the data
- It's where the model is forced to be efficient
- If too big: model memorizes; too small: model forgets
- Sweet spot = good generalization

Autoencoders: Architecture (cont.)

Decoder:

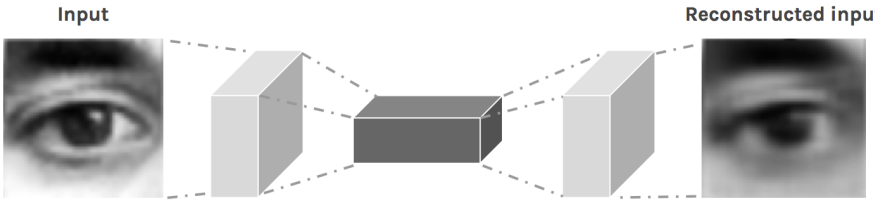
- Takes the bottleneck output
- Expands it back to the original input size
- Tries to make it as close as possible to the real input
- Learns to generate data from compressed features
- *Example*: Converts 32 numbers back to a 784-pixel image

Autoencoders: Architecture (cont.)

Component	Description
Encoder	Transforms the input data into a compressed latent representation using layers such as Dense, Convolutional, or Recurrent layers.
Latent Space (Bottleneck)	The central compact representation of the input. Forces the model to learn the most essential features.
Decoder	Reconstructs the input from the latent space. Typically mirrors the encoder's structure in reverse.
Loss Function	Measures the difference between input and output (e.g., Mean Squared Error, Binary Cross-Entropy). Drives learning during training.
Training	Involves minimizing reconstruction error using gradient-based optimization methods such as SGD or Adam.

Architecture Summary Table

Autoencoders: Architecture (cont.)



Sample Autoencoder

Autoencoders: Interactive Demo

https://douglasduhaime.com/posts/visualizing-latent-spaces.html

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*
- **Does not work**. Autoencoders just learn a function that maps input to output. The learned latent space is too discontinous to work as a generative model.

Autoencoders as Generative Models

- Autoencoders project data into a latent space Z.
- The latent space is not necessarily continuous or structured.
- *What if we sample a new embedding vector from Z and then have the decoder reconstruct the image from it?*
- **Does not work**. Autoencoders just learn a function that maps input to output. The learned latent space is too discontinous to work as a generative model.
- Sampling randomly from the latent space may result in invalid outputs.
- They do not maximize the likelihood of the data.

Autoencoders as Generative Models (cont.)

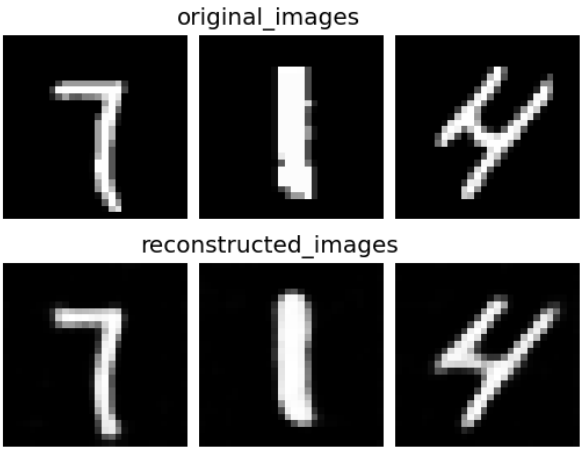


Image reconstruction with autoencoder trained on MNIST digits

generated_images

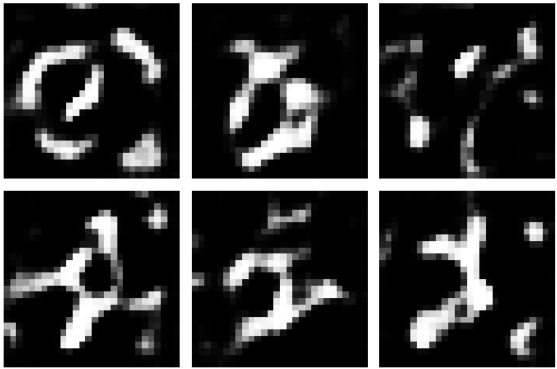


Image generation with autoencoder trained on MNIST digits. Encoding vector sampled from latent space Z and the passed to decoder.

Applications of Autoencoders

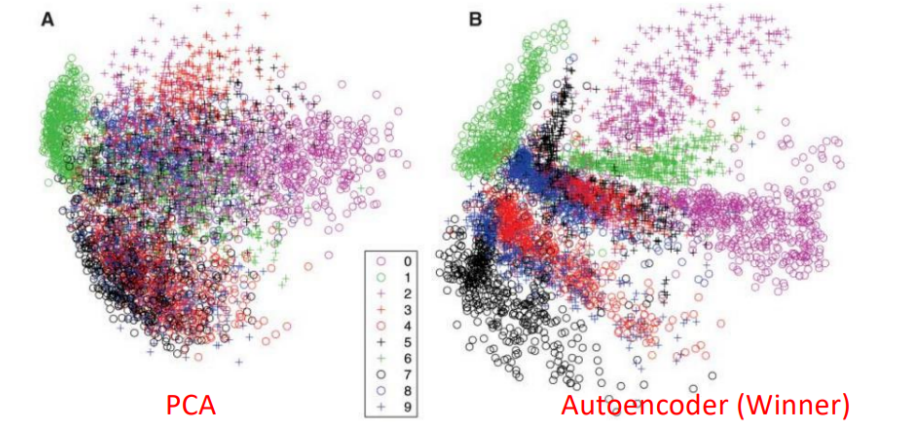
- Dimensionality Reduction
- Super-Resolution
- Colorization
- Anomaly Detection
- Denoising

Autoencoders help us reduce the number of features in our data, similar to PCA, but in a more flexible way.

- The encoder takes big, complex data and squeezes it into a smaller, simpler form.
- This smaller version keeps the important information.
- Great for making it easier to see patterns or to prepare data for other machine learning tasks.

Example: We can use autoencoders to turn images (like MNIST digits) into just 2 or 3 numbers, so we can plot and explore them easily.

Dimensionality Reduction (cont.)



t-SNE visualization on MNIST digits dataset. PCA vs. Autoencoders. The image vector is projected into $\mathbb{R}^2$.

Super-Resolution

Autoencoders can be used to reconstruct high-resolution images from their low-resolution counterparts.

- Input: Low-resolution image.
- Output: High-resolution image.
- Often implemented with convolutional layers for spatial pattern learning.

Use Case: Enhancing medical images, satellite images, or upscaling low-resolution photos.

Super-Resolution (cont.)

Image super-resolution using Autoencoders

Image Colorization

Autoencoders can learn to predict color information for grayscale images.

- Input: Grayscale image (1 channel).
- Output: Colorized image (3 channels — RGB).
- Requires learning semantic and contextual relationships to apply realistic colors.

Use Case: Restoring historical black-and-white photos.

Image Colorization (cont.)

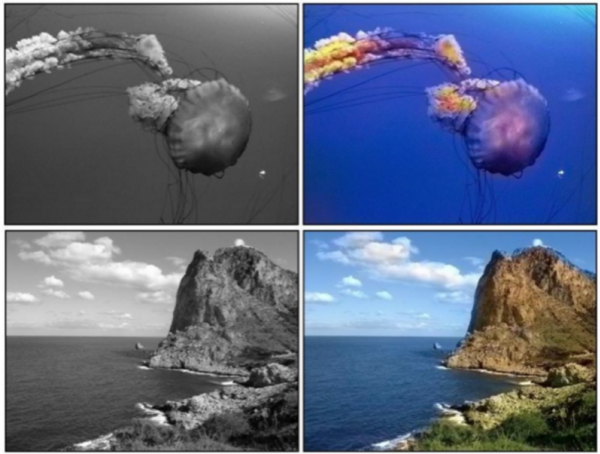


Figure 2: Image colorization using Autoencoders

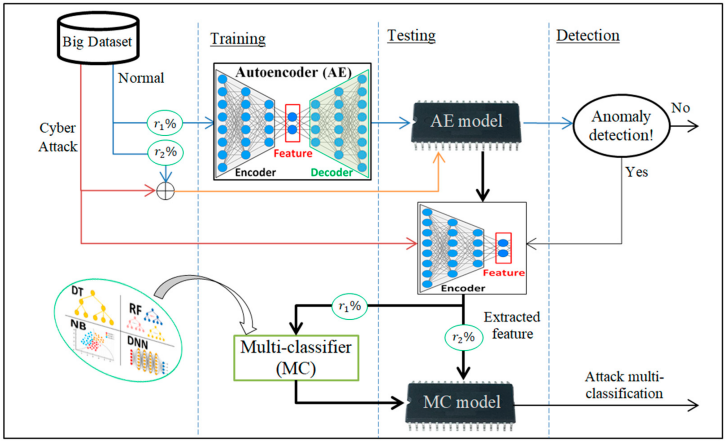
Anomaly Detection

- Autoencoders are trained to reconstruct input data.
- During training, the model learns to compress and decompress data that is similar to the training set (normal data).
- When presented with anomalous (outlier) data, the reconstruction error increases, as the autoencoder cannot effectively reconstruct unseen patterns.

Use Cases:

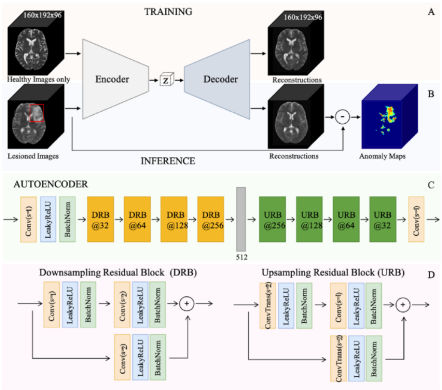
- Fraud detection in financial transactions
- Fault detection in industrial systems
- Intrusion detection in cybersecurity
- Medical anomaly detection (e.g., rare diseases in imaging)

Anomaly Detection (cont.)



Structure of the deep autoencoder (AE) for anomaly detection and feature extraction for multi-classification of cyber-attacks.

Anomaly Detection (cont.)



Autoencoder-based anomaly detection framework: A) Training the autoencoder for modeling the distribution of healthy brain anatomy. B) In the inference phase, anomaly maps reveal lesions by subtracting their reconstruction from the input image. C and D) Details of the proposed autoencoder.

Key Idea:

- Train the autoencoder to reconstruct the original, clean input from a corrupted (noisy) version.
- The encoder learns robust features that capture the underlying structure of the data, ignoring noise.

Training Process:

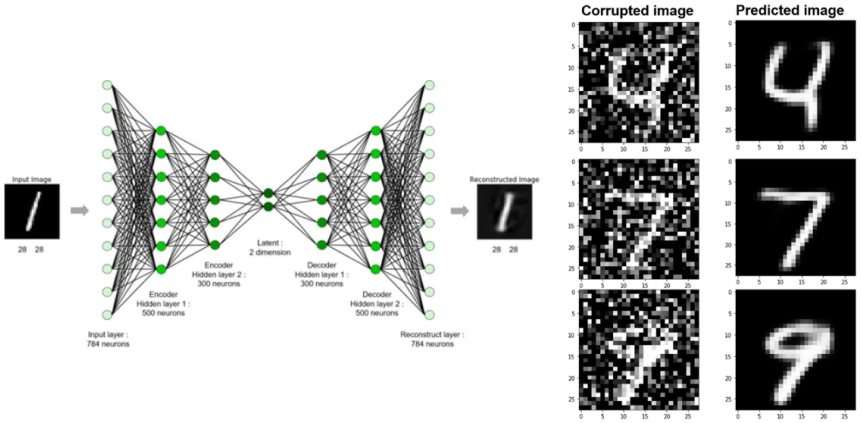
1. Add noise to the input data (e.g., Gaussian noise, salt-and-pepper noise).
2. Feed the noisy input to the autoencoder.
3. Compute the loss between the output and the original clean input.
4. Update the network weights to minimize this loss.

Denoising (cont.)

Use Cases:

- Image denoising: Removing noise from photographs or medical images.
- Speech enhancement: Improving audio quality by filtering out background noise.
- Preprocessing step: Providing cleaner data for downstream tasks such as classification or segmentation.

Denoising (cont.)



Example of denoising using an autoencoder.

Challenges

- May just copy input without real learning
- Needs careful tuning of bottleneck size
- Sometimes blurry outputs (especially basic autoencoders)
- Doesn't always generate creative results — use VAEs/GANs

References

[1] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504-507.

[2] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103).

[3] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

[4] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

References (cont.)

[5] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798-1828.

[6] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (pp. 37-49).

[7] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.