Contents

WEEK 5: AUTOENCODERS & EMBEDDINGS

Introduction to Autoencoders

Autoencoders are a class of neural networks designed to learn efficient representations of data in an unsupervised manner. The fundamental architecture consists of two main components: an encoder that compresses the input into a lower-dimensional latent representation, and a decoder that reconstructs the original input from this compressed representation. The objective of an autoencoder is to minimize the reconstruction error, which measures the difference between the input and its reconstruction. This simple yet powerful concept has led to numerous applications in dimensionality reduction, feature learning, and generative modeling.

The encoder function f maps the input x to a latent representation z: $z = f(x)$, while the decoder function g maps the latent representation back to the reconstruction $\hat{x}$: $\hat{x} = g(z) = g(f(x))$. The network is trained to minimize the reconstruction loss, typically measured using mean squared error (MSE) for continuous data or binary cross-entropy for binary data:

$$L(x, \hat{x}) = ||x - \hat{x}||^2 = ||x - g(f(x))||^2$$

The bottleneck layer, which represents the latent space z, forces the network to learn a compressed representation of the input. The dimensionality of this layer is a critical hyperparameter that determines the capacity of the autoencoder. A smaller bottleneck forces more aggressive compression and can lead to loss of information, while a larger bottleneck may allow the network to simply memorize the input without learning meaningful features.

Autoencoder Architectures

The basic autoencoder architecture can be extended in various ways to suit different applications and data types. Undercomplete autoencoders have a bottleneck layer with dimensionality lower than the input, forcing the network to capture the most salient features of the data. These are particularly useful for dimensionality reduction and feature extraction. When the latent dimension is chosen appropriately, undercomplete autoencoders can learn representations similar to principal component analysis (PCA), but with the added flexibility of non-linear transformations.

Overcomplete autoencoders, in contrast, have a latent dimension larger than or equal to the input dimension. Without additional constraints, these networks could simply learn the identity function. However, when combined with regularization techniques such as sparsity constraints or denoising objectives, overcomplete autoencoders can learn useful representations. Sparse autoencoders add a sparsity penalty to the loss function, encouraging the network to learn representations where only a small fraction of neurons are active for any given input:

$$L_{sparse}(x, \hat{x}) = ||x - \hat{x}||^2 + \lambda \sum_j KL(\rho||\hat{\rho}_j)$$

where ρ is the target sparsity parameter, $\hat{\rho}_j$ is the average activation of neuron j, and KL denotes the Kullback-Leibler divergence.

Convolutional autoencoders are specifically designed for image data, using convolutional layers in the encoder and transposed convolutions (or upsampling followed by convolutions) in the decoder. This architecture preserves spatial structure and is particularly effective for image-related tasks such as denoising, super-resolution, and compression. The convolutional encoder progressively reduces spatial dimensions while increasing the number of channels, capturing hierarchical features from low-level edges to high-level semantic information.

Recurrent autoencoders are used for sequential data such as time series or text. The encoder is typically a recurrent neural network (RNN) that processes the input sequence and produces a fixed-size latent representation, while the decoder is another RNN that generates the output sequence from this representation. Variations include using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells to better capture long-range dependencies in the data.

Denoising Autoencoders

Denoising autoencoders (DAEs) are a powerful variant that learns robust representations by training the network to recover clean data from corrupted inputs. The key idea is to add noise to the input data during training, while keeping the target output as the original, uncorrupted data. This forces the network to learn features that are robust to noise and capture the underlying structure of the data distribution rather than merely memorizing the input.

The training process involves two steps: first, the input x is corrupted using a stochastic corruption process $C(x|\tilde{x})$ to produce a noisy version $\tilde{x}$. Common corruption methods include additive Gaussian noise, salt-and-pepper noise, or random masking of input features. Second, the autoencoder is trained to minimize the reconstruction error between the original clean input x and the reconstruction from the corrupted input $\hat{x} = g(f(\tilde{x}))$:

$$L_{DAE}(x) = ||x - g(f(\tilde{x}))||^2$$

Denoising autoencoders have been shown to learn representations that correspond to the underlying data manifold, effectively capturing the structure of the data distribution. The corruption process can be viewed as a regularization technique that prevents the network from learning trivial identity mappings and encourages the discovery of more robust features. Vincent et al. (2008) demonstrated that denoising autoencoders are implicitly learning the score function (gradient of the log probability) of the data distribution.

The choice of corruption level is an important hyperparameter. Too little corruption may not provide sufficient regularization, while too much corruption may make the reconstruction task impossible, preventing the network from learning useful features. In practice, the corruption level is often determined through validation set performance or based on domain knowledge about the expected noise characteristics in the application.

Variational Autoencoders

Variational Autoencoders (VAEs) represent a significant advancement in generative modeling, combining ideas from autoencoders with probabilistic inference. Unlike standard autoencoders that learn a deterministic mapping to the latent space, VAEs learn a probability distribution over the latent space. This probabilistic formulation enables VAEs to generate new samples by sampling from the learned latent distribution and passing these samples through the decoder.

The VAE framework is based on variational inference, where the goal is to approximate the intractable posterior distribution $p(z|x)$ with a tractable approximate posterior $q(z|x)$, typically chosen to be a Gaussian distribution. The encoder network outputs the parameters of this distribution (mean μ and variance σ^2), and the latent representation is obtained by sampling from this distribution: $z \sim \mathcal{N}(\mu(x), \sigma^2(x))$.

The loss function for VAEs consists of two terms: the reconstruction loss and the Kullback-Leibler (KL) divergence between the approximate posterior and a prior distribution (typically a standard Gaussian $\mathcal{N}(0, I)$):

$$L_{VAE}(x) = \mathbb{E}_{z\sim q(z|x)}[||x - g(z)||^2] + KL(q(z|x)||p(z))$$

The reconstruction term ensures that the decoder can accurately reconstruct the input from sampled latent representations, while the KL term acts as a regularizer, encouraging the learned latent distribution to be close to the prior. This regularization is crucial for enabling generation: by ensuring the latent space follows a known distribution, we can generate new samples by sampling from the prior and decoding.

A key technical challenge in training VAEs is that sampling is a non-differentiable operation, which prevents backpropagation. This is addressed using the reparameterization trick: instead of sampling z directly from $\mathcal{N}(\mu, \sigma^2)$, we sample ϵ from a standard Gaussian $\mathcal{N}(0, I)$ and compute $z = \mu + \sigma \odot \epsilon$. This reformulation allows gradients to flow through μ and σ, enabling end-to-end training.

VAEs have been widely adopted for various generative tasks, including image generation, text generation, and molecule design. They offer several advantages over standard autoencoders: the learned latent space is continuous and well-structured, enabling smooth interpolation between data points; they provide a principled probabilistic framework for generation; and they can be used for tasks such as semi-supervised learning and representation learning.

Latent Space and Embeddings

The latent space learned by autoencoders serves as a compressed representation or embedding of the input data. Understanding the properties and structure of this latent space is crucial for leveraging autoencoders effectively in downstream tasks. A well-learned latent space should capture the essential characteristics and variations in the data while discarding noise and irrelevant details.

One important property of the latent space is its dimensionality. The choice of latent dimension involves a trade-off between compression and information preservation. Principal Component Analysis (PCA) provides a linear baseline for dimensionality reduction, where the optimal number of components can be determined by examining the explained variance. Autoencoders extend this concept to non-linear dimensionality reduction, but determining the optimal latent dimension often requires experimentation and validation on downstream tasks.

The geometry of the latent space reveals important information about the learned representation. In well-trained autoencoders, similar inputs are mapped to nearby points in the latent space, while dissimilar inputs are mapped to distant points. This property enables the latent space to be used for tasks such as clustering, visualization, and similarity search. Visualization techniques such as t-SNE or UMAP can be applied to the latent representations to create 2D or 3D visualizations that reveal the structure and organization of the data.

Interpolation in the latent space is a powerful capability that demonstrates the continuity and smoothness of the learned representation. By linearly interpolating between two latent vectors z_1 and z_2, we can generate a sequence of intermediate representations: $z_t = (1 - t)z_1 + tz_2$ for $t \in [0, 1]$. When decoded, these intermediate representations should produce smooth transitions between the corresponding inputs. For VAEs, the latent space is explicitly encouraged to be continuous through the KL regularization term, making interpolation particularly effective.

Disentanglement is another desirable property of latent representations, where different dimensions of the latent vector correspond to independent factors of variation in the data. For example, in face images, different latent dimensions might independently control attributes such as age, gender, expression, or lighting. Disentangled representations facilitate interpretability and controllable generation, but achieving disentanglement often requires specialized architectures or training procedures such as β-VAE, which emphasizes the KL term in the loss function.

Applications of Autoencoders

Autoencoders have found applications across a wide range of do domains in machine learning and data science. In dimensionality reduction, autoencoders provide a non-linear alternative to traditional methods like PCA. By learning a compressed representation that captures the most important features of high-dimensional data, autoencoders enable more efficient storage, faster computation, and improved visualization. This is particularly valuable in domains such as genomics, where datasets may have thousands or millions of features.

Anomaly detection is another important application where autoencoders excel. The idea is to train an autoencoder on normal data, and then use the reconstruction error as an anomaly score. Normal data should have low reconstruction error, while anomalous data that differs from the training distribution should have high reconstruction error. This approach has been successfully applied in fraud detection, network intrusion detection, and manufacturing quality control. Variations include using the latent space representation for outlier detection or combining reconstruction error with other statistical measures.

Image denoising and enhancement leverage the capability of denoising autoencoders to recover clean signals from corrupted inputs. Applications include removing noise from medical images, enhancing low-light photographs, and restoring damaged historical photographs. Super-resolution, where a low-resolution image is enhanced to higher resolution, can also be formulated as an autoencoder problem where the encoder processes the low-resolution input and the decoder generates the high-resolution output.

Feature learning for downstream tasks is a common use case where autoencoders serve as a pre-training step. The encoder portion of a trained autoencoder can be used as a feature extractor, with the learned latent representations serving as input to supervised learning algorithms. This is particularly useful when labeled data is scarce but abundant unlabeled data is available. The unsupervised pre-training helps the network learn useful features from the unlabeled data, which can then be fine-tuned on the smaller labeled dataset.

Data generation and synthesis using VAEs enables the creation of new, realistic samples. Applications include generating synthetic medical images for data augmentation, creating novel drug molecules, and producing realistic text or music. The probabilistic nature of VAEs allows for controlled generation by manipulating the latent variables, enabling applications such as style transfer, attribute manipulation, and conditional generation where the output is conditioned on specific attributes or labels.

Compression and efficient representation learning is another practical application where autoencoders can learn task-specific compression schemes that outperform traditional methods like JPEG for specific types of data. This is particularly relevant in scenarios where transmission bandwidth or storage capacity is limited, such as in satellite imagery or video streaming.

Challenges and Considerations

Training autoencoders effectively requires careful attention to several challenges and design choices. One fundamental challenge is balancing reconstruction quality with the complexity of the learned representation. Overfitting can occur when the autoencoder simply memorizes the training data without learning generalizable features. Regularization techniques such as weight decay, dropout, or architectural constraints help mitigate this issue. Additionally, using validation data to monitor reconstruction error and stopping training when performance plateaus can prevent overfitting.

The choice of loss function significantly impacts the learned representations. Mean squared error (MSE) is common for continuous data but may not always be the most appropriate choice. For image data, perceptual loss functions that measure similarity in feature space rather than pixel space often produce better results. For binary data, binary cross-entropy is more suitable than MSE. Domain-specific loss functions that incorporate prior knowledge about the data can further improve performance.

Mode collapse is a particular concern for VAEs and other generative models, where the model learns to generate only a limited subset of the possible outputs, failing to capture the full diversity of the data distribution. This can be addressed through techniques such as adjusting the weight of the KL term (β-VAE), using more expressive posterior distributions, or employing adversarial training.

Computational efficiency is an important practical consideration, especially for large-scale applications. Deep autoencoders with many layers can be computationally expensive to train and deploy. Techniques such as progressive training, where the network is trained incrementally with increasing complexity, or knowledge distillation, where a smaller network is trained to mimic a larger one, can help address these challenges. Additionally, architectural choices such as using separable convolutions or bottleneck layers can reduce computational requirements while maintaining performance.

Evaluation of autoencoder performance is not always straightforward. While reconstruction error provides a direct measure of how well the autoencoder can reconstruct its inputs, it may not always correlate with performance on downstream tasks. Qualitative evaluation through visualization of latent representations and reconstructions, as well as quantitative evaluation on specific tasks such as classification or clustering, provides a more comprehensive assessment of autoencoder quality.

Advanced Topics and Extensions

Recent advances in autoencoder research have led to several sophisticated extensions and variations. Adversarial Autoencoders (AAEs) combine autoencoders with generative adversarial networks (GANs) by using a discriminator to match the latent distribution to a prior distribution, rather than using KL divergence. This approach can lead to more flexible and powerful generative models while maintaining the interpretability and ease of training of autoencoders.

Hierarchical VAEs extend the basic VAE framework to learn hierarchical representations with multiple levels of latent variables. This allows the model to capture structure at different scales, from fine-grained details to high-level semantic content. Hierarchical VAEs have shown improved performance on complex datasets such as natural images and have connections to hierarchical Bayesian models.

Vector Quantized VAEs (VQ-VAE) introduce a discrete latent space by quantizing the continuous latent representations to a learned codebook of vectors. This discrete representation can be beneficial for certain applications such as text-to-speech synthesis or image generation, and can be combined with autoregressive models for powerful generative capabilities.

Transformer-based autoencoders apply the self-attention mechanism from transformers to the autoencoder framework, enabling better modeling of long-range dependencies in sequential data. These models have achieved state-of-the-art performance on tasks such as text generation, machine translation, and time series forecasting.

Semi-supervised VAEs extend VAEs to leverage both labeled and unlabeled data by incorporating a classifier in the latent space. The model learns to jointly reconstruct the input and predict the label, enabling effective use of large amounts of unlabeled data combined with a small labeled dataset. This approach has shown promising results in domains where labeling is expensive or time-consuming.

Learning Objectives

- Understand autoencoder architecture and training principles
- Implement various autoencoder variants for different tasks
- Master latent space concepts and representation learning
- Apply autoencoders to practical problems like denoising and dimensionality reduction
- Comprehend the differences between standard autoencoders and VAEs