Advancements in Deep Learning: Exploring RNNs, LSTMs, Attention Mechanisms, Vision Transformers, GANs, and Autoencoders

Kerem Zengin

January 2, 2024

Contents

Abstract

In recent years, deep learning has emerged as a transformative force in artificial intelligence, propelling advancements across various domains. This paper delves into key deep learning architectures, providing a comprehensive overview of Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), Attention Mechanisms, Vision Transformers (ViT), and Convolutional Neural Networks (CNNs). Each architecture is examined for its unique contributions to processing sequential data, image recognition, and feature extraction. Additionally, we explore the realm of Generative Adversarial Networks (GANs) and Autoencoders, highlighting their growing significance in generating realistic and diverse data, along with practical implementations and analyses. This synthesis elucidates the intricate workings, strengths, and potential applications of these groundbreaking deep learning architectures, paving the way for future innovations and research in the field.

1 Autoencoders: Implementation and Analysis

1.1 Introduction

Autoencoders are a type of artificial neural network used to learn efficient encodings of data, typically for the purpose of dimensionality reduction or feature extraction. Unlike many other types of neural networks, autoencoders are unsupervised learning models that aim to reconstruct the input data as closely as possible, often compressing the data into a lower-dimensional space in the process.

1.2 Types of Autoencoders

Autoencoders can be categorized into several types based on their architecture and the specific task they are designed for. Some of the common types include:

- **Basic Autoencoders**: These consist of a simple feed-forward network with a bottleneck layer at the center, compressing the input into a lower-dimensional code.
- **Convolutional Autoencoders**: Designed for image data, these autoencoders use convolutional layers to exploit the spatial structure of the data.
- **Denoising Autoencoders**: These are trained to remove noise from corrupted inputs, essentially learning to reconstruct the original, uncorrupted data.
- **Variational Autoencoders**: A more advanced type, variational autoencoders not only learn to compress the data but also to generate data similar to the inputs, being part of generative models.

1.3 Implementation and Experiments

The objective of this project was to implement a two-layer fully connected autoencoder using only NumPy, applied on the MNIST handwritten digits dataset. The implementation involved several key steps outlined below:

1. **Data Loading and Preprocessing**: The MNIST dataset was loaded, and the images were flattened from their original 28x28 pixel format into 784-dimensional vectors. The pixel values were then normalized to fall within the range of 0 to 1.

2. **Parameter Initialization**: The network consisted of two layers. The input layer size was set to 784 (corresponding to the flattened images), and a hidden layer size of 64 was chosen. The weights for both layers were initialized with a zero mean and a standard deviation of 0.01.

3. **Activation Functions and Loss Calculation**: The ReLU activation function was implemented for the hidden layer, and the sigmoid activation function for the output layer. The mean squared error was used as the loss function to measure the reconstruction error of the autoencoder.

4. **Backward Mode Implementation**: The backward mode for both ReLU and sigmoid activation functions was implemented to facilitate the backpropagation process during training.

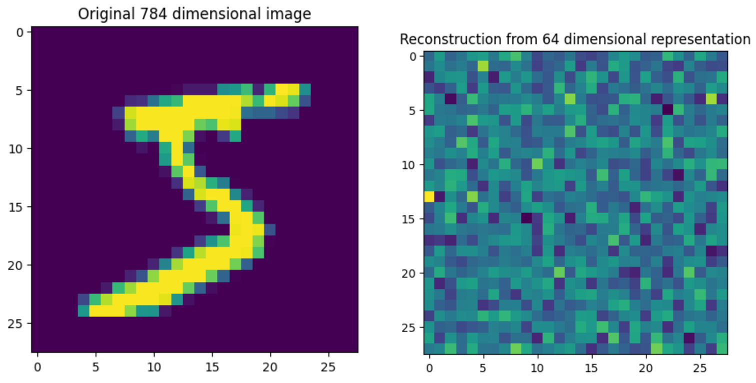
5. **Training Loop**: The autoencoder was trained over 10 epochs with a batch size of 1024 and a learning rate of 0.0001. Each epoch involved forward propagation through the network, computation of loss, and backpropagation to update the weights.

6. **Visualization**: Throughout the training process, the original and reconstructed images were visualized every 10 iterations to monitor the progress of the model and to visually assess the reconstruction quality.

1.4 Results and Discussion

The autoencoder was diligently trained on the MNIST dataset, which comprises a wide variety of handwritten digits. The training aimed to enable the autoencoder to learn a compact representation of the data, and then reconstruct the original images from this compressed form. The figure below presents a side-by-side comparison of the original images against the reconstructed images after the autoencoder's training.

The reconstructed images, when compared to the original ones, exhibit a loss of detail, which is a common characteristic in the dimensionality reduction process. Despite the reduced clarity, the general structure and form of the digits remain discernible, which suggests that the autoencoder has captured the most crucial features of the data. However, the granularity and sharpness of the original images are not fully retained, as evidenced by the somewhat blurred and smoothed features in the reconstructed images.



(a) Original Image (b) Reconstructed Image

Figure 1: Comparison of original and reconstructed images from the MNIST dataset, illustrating the effect of dimensionality reduction by the autoencoder.

1.5 Conclusion

The exploration into autoencoder architectures through this project has been insightful, highlighting both the capabilities and limitations of such models. The autoencoder demonstrated proficiency in encoding and decoding the MNIST dataset, albeit with some loss in image detail. This underscores the trade-off between data compression and fidelity in reconstruction that is intrinsic to autoencoder designs.

Further investigation may include experimenting with alternative architectures, such as convolutional autoencoders, which are better suited for image data. Additionally, variational autoencoders could be explored for their generative properties and their ability to model the probability distribution of the data.

Overall, the project solidifies the understanding of autoencoders in unsupervised learning and their application in tasks such as data denoising, dimensionality reduction, and feature extraction. The results encourage continued research and experimentation in the field, aiming for models that can achieve higher fidelity reconstructions with greater compression rates.

2 Generative Adversarial Networks

2.1 Introduction

Generative Adversarial Networks (GANs) are a powerful class of neural networks designed for generative tasks. They consist of two competing networks: a generator (G) that learns to generate plausible data, and a discriminator (D) that learns to distinguish between real and generated data. This adversarial process drives both networks to improve their performance over time, ultimately enabling the generator to produce highly realistic data.

2.2 GAN Architecture

The GAN architecture for this project includes two fully connected networks, each with a single hidden layer containing 256 neurons. The networks are constructed within a single class in PyTorch, using the Sequential model. ReLU activation functions are used for the hidden layer, and a Sigmoid activation function is employed at the output layer.

2.3 Loss Function

The Binary Cross-Entropy (BCE) Loss is used as the objective function for both the discriminator and generator. It quantifies the difference between the distributions of the real and the generated data.

2.4 GAN Variants

While the focus of this project is on a basic GAN structure, there exist several interesting variants of GANs that are worth mentioning:

- **BiGAN (Bidirectional GAN)**: Extends the GAN architecture by introducing an encoder alongside the generator and discriminator, allowing for the mapping of data to and from the latent space.
- **LSGAN (Least Squares GAN)**: Uses the least squares loss function for the discriminator to address and improve the stability of GAN training.

2.5 Model Construction

The GAN model is built with fully connected layers for both the generator and discriminator, abiding by the specifications outlined in the task. The generator takes a 100-dimensional latent vector as input and outputs a 784-dimensional vector, reshaped into a 28x28 image. The discriminator takes the flattened MNIST images as input and outputs a single scalar representing the probability of the input being real.

2.6 Training Procedure

The training process involves alternating between updating the generator and discriminator. In each epoch, the generator aims to produce images that are increasingly indistinguishable by the discriminator, while the discriminator strives to enhance its classification accuracy.

2.7 Optimizer and Hyperparameters

The Adam optimizer was selected for training both networks, with a learning rate set to 0.0002. The batch size was chosen as 16, and the networks were trained for a minimum of 10 epochs.

2.8 Visualization

The progression of the generator's capabilities was monitored by visualizing the output after each epoch. A fixed latent vector was used to generate 16 samples, which were displayed in a 4x4 grid to observe the evolution of the generated digits.

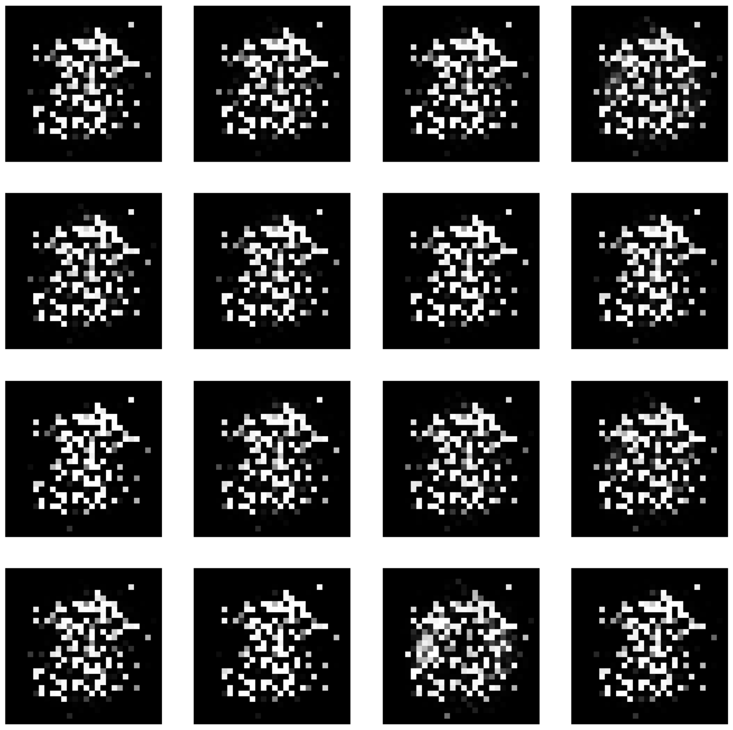


Figure 2: Generator samples after the first epoch.

2.9 Conclusion

The GAN successfully learned to generate new images that resemble the MNIST digits. Although the initial results show room for improvement, the network demonstrates the potential of GANs in generating new data that approximates the real data distribution. Further training and refinement of the model, as well as experimentation with different GAN architectures, could lead to more precise and diverse digit generation.

3 DL Architectures: RNN, LSTM, Attention Mechanisms, ViT, and CNNs

3.1 Vanilla RNNs and LSTMs

3.1.1 Vanilla RNN

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data by maintaining a hidden state that evolves over time. The vanilla RNN processes input sequences one element at a time and updates its hidden state using a simple feedforward network.

3.1.2 LSTM

Long Short-Term Memory networks (LSTMs) are an advanced variant of RNNs, equipped with mechanisms called gates that regulate the flow of information. These gates help LSTMs to remember important information over long sequences and forget the irrelevant, addressing the vanishing gradient problem often encountered in vanilla RNNs.

3.2 Recurrent vs. Self-attention Models

Recurrent models, like RNNs and LSTMs, process data sequentially, which can become a bottleneck for parallel computation. Self-attention models, on the other hand, compute the representation of each element in a sequence by considering its context in the entire sequence, allowing for parallel processing and capturing long-range dependencies more effectively.

Transformers, which are based on self-attention, have recently been outperforming RNNs in text analytics tasks because they can handle longer contexts and train more efficiently on large datasets due to their parallelizable nature.

3.3 ViT vs. CNNs

3.3.1 CNN

Convolutional Neural Networks (CNNs) are specialized for image processing with architectures designed to capture hierarchical patterns and local features in images using convolutional filters.

3.3.2 ViT

Vision Transformers (ViT) apply the principles of transformers to image recognition by treating patches of images as sequences. This allows ViTs to consider the global context of the entire image, leading to better performance on recognition tasks.

3.4 Performance Trends

Transformer-based models like ViT are outperforming CNNs in image recognition tasks because they can learn more abstract and comprehensive representations of images. The self-attention mechanism in ViTs captures global dependencies, which can be more informative than the local patterns that CNNs focus on.

3.5 Conclusion

The advancements in deep learning architectures are continuously shaping the landscape of machine learning. RNNs and LSTMs have paved the way for understanding sequential data, while self-attention and transformers have taken the stage for their ability to handle long-range dependencies and parallel computation. In image recognition, ViT presents a promising direction by leveraging the power of transformers to achieve state-of-the-art results.