TensorFlow in Action (Part 1)

Chapter 4 : Dipping Toes in Deep Learning

This chapter starts with **fully connected networks (FCNs)**, using an autoencoder on MNIST digit images as the main example. The autoencoder compresses each flattened 28×28 image into a lower-dimensional latent representation and then reconstructs it, with the goal of minimizing reconstruction error rather than predicting labels. A denoising variant is implemented in Keras: parts of the input image are randomly masked, and the network learns to recover the clean version, using ReLU in hidden layers and a tanh activation in the output to stay within $(-1,1)$. Training over several epochs shows the loss steadily decreasing (for example from around 0.15 to about 0.078), and visualization of outputs demonstrates that the model can restore corrupted digits reasonably well.

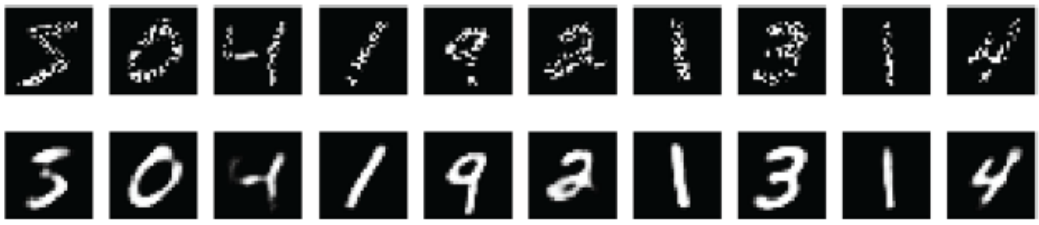

Figure 1 Images restored by the model

The chapter emphasizes that autoencoders are useful for unsupervised feature learning and dimensionality reduction, providing representations that can later help with tasks like classification or clustering.

Next, the focus shifts to **convolutional neural networks (CNNs)** for vision. Using the CIFAR-10 dataset (small 32×32 RGB images from 10 classes such as automobiles, ships, and animals), you build a CNN in Keras to detect vehicles as a proxy for traffic-congestion modeling. The author explains how convolutional layers exploit spatial structure, sharing weights across locations and using local receptive fields, followed by pooling layers to progressively reduce spatial resolution while keeping important features. You see how to stack conv–pool blocks, flatten the results, and feed them into dense layers to output class probabilities, then train and evaluate the model to check if the approach is promising for real-world deployment.

Finally, the chapter introduces **recurrent neural networks (RNNs)** for sequence data. The example here is a time-series forecasting task: predicting future atmospheric CO_2 levels from past measurements. After preparing sliding windows of historical data as input sequences, you construct and train an RNN (for example using simple RNN or more advanced units like LSTM/GRU) that processes the sequence one step at a time and outputs predictions for future

values. The chapter highlights how RNNs maintain hidden state across time steps and why they are suitable for temporal patterns like trends and seasonality, then uses the trained model to generate future CO_2 predictions and visually compare them against the ground truth.