Training Day-90 Report:

Recurrent Neural Networks (RNNs), Restricted Boltzmann Machines (RBMs), and Autoencoders using Keras:

Recurrent Neural Networks (RNNs)
Definition:
RNNs are a class of neural networks designed for sequential data. They use loops to retain memory of previous computations, making them suitable for time-series data, text, and speech.

Key Concepts in RNNs

1. **Sequential Memory:**
    - RNNs maintain a "hidden state" to store information from prior time steps.
2. **Unfolding:**
    - Input data is unfolded over time steps, allowing the network to process sequential information.
3. **Challenges with RNNs:**
    - **Vanishing/Exploding Gradients:** Issues during backpropagation for long sequences.
    - Addressed by advanced architectures like **LSTMs** and **GRUs**.

Implementing RNN with Keras

```
import tensorflow as tf

# Define the RNN model
model = tf.keras.Sequential([
   tf.keras.layers.SimpleRNN(128, activation='relu', input_shape=(10, 50)),
   tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
          loss='binary_crossentropy',
          metrics=['accuracy'])

# Summary
model.summary()
```

Restricted Boltzmann Machines (RBMs)

Definition:
RBMs are energy-based probabilistic models that learn a joint probability distribution over visible and hidden variables. They are often used for dimensionality reduction, feature learning, and pretraining deep networks.

Structure of an RBM

1. **Visible Layer:**
    - Represents input data.
2. **Hidden Layer:**
    - Learns abstract features from the visible layer.
3. **Energy Function:**
    - Defines relationships between visible and hidden units to measure configuration quality.

Applications of RBMs

- Collaborative filtering (e.g., recommendation systems).
- Dimensionality reduction.
- Pretraining layers of deep networks.

RBMs in TensorFlow
While TensorFlow does not provide native RBM support, RBMs can be implemented using TensorFlow or specialized libraries like pytorch-boltzmann.

Autoencoders with Keras
Definition:
Autoencoders are unsupervised learning models that encode input data into a smaller latent representation and reconstruct the input from this representation. They are widely used for dimensionality reduction, anomaly detection, and generative modeling.

Structure of an Autoencoder

1. **Encoder:**
    - Compresses input data into a latent representation.
    - Example:
    - tf.keras.layers.Dense(128, activation='relu')
2. **Latent Space:**
    - A bottleneck layer that forces the network to learn compressed features.
3. **Decoder:**
    - Reconstructs the input data from the latent representation.

Types of Autoencoders

1. **Basic Autoencoders:**
    - Learn to reconstruct data.

2. **Denoising Autoencoders:**
    - Learn to reconstruct data from noisy input.
3. **Variational Autoencoders (VAEs):**
    - Generate new data samples similar to the training data by introducing probabilistic latent spaces.

Implementing an Autoencoder in Keras

```
import tensorflow as tf

# Encoder
encoder = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(784,)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu')
])

# Decoder
decoder = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(64,)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(784, activation='sigmoid')
])

# Full autoencoder
autoencoder = tf.keras.Model(inputs=encoder.input, outputs=decoder(encoder.output))

# Compile the autoencoder
autoencoder.compile(optimizer='adam', loss='mse')

# Summary
autoencoder.summary()
```

Comparison

Aspect	RNNs	RBMs	Autoencoders
Purpose	Sequential data processing	Feature learning, pretraining	Dimensionality reduction
Architecture	Recurrent loops	Energy-based probabilistic	Encoder-decoder structure
Common Use Cases	NLP, time-series prediction	Recommendation systems	Anomaly detection, compression

Applications

1. **RNNs:**
    - Text generation, speech recognition, language translation.
2. **RBMs:**
    - Collaborative filtering (e.g., Netflix recommendation engine).
3. **Autoencoders:**
    - Denoising images, feature extraction, anomaly detection.