Intro to Neural Networks + Overview of AI Architectures

Overview of Deep Learning Architectures

Lesson Objectives

By the end of this lesson, students will be able to:

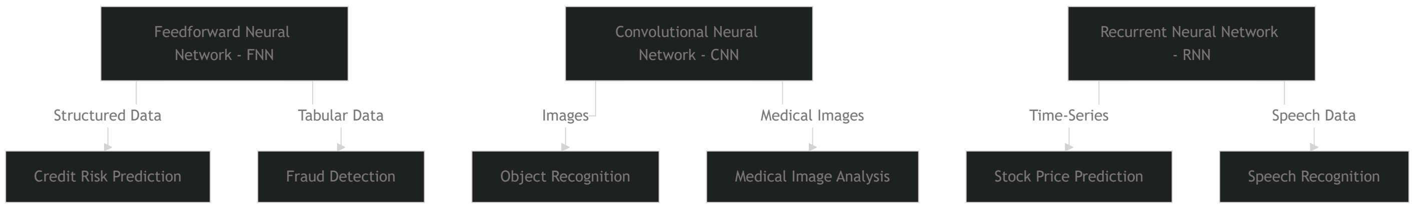
- Identify key deep learning architectures and their applications.
- Understand the differences between FNNs, CNNs, RNNs, and Transformers.
- Implement a basic CNN model for image classification.
- Load and fine-tune a Transformer model for text classification.

What is a Deep Learning Architecture?

Deep learning architectures define **how neural networks process and learn from data**. Different architectures are optimized for different types of input data, such as structured numbers, images, or text.

> **Key Takeaway:** The choice of architecture impacts model performance, efficiency, and suitability for specific AI tasks.

Traditional Deep Learning Architectures



Feedforward Neural Networks (FNNs)

Used primarily for structured/tabular data. These networks process inputs in a single direction, from input to output.

Convolutional Neural Networks (CNNs)

Designed for **image processing**, CNNs extract spatial patterns using convolutional layers.

Recurrent Neural Networks (RNNs)

Specialized for **sequential data**, where order matters.

Coding Walkthrough: Implementing a Simple CNN

In this walkthrough we are going to achieve two things:

1. **Train** a basic CNN to classify images from the **MNIST dataset**.
2. **Understand** how convolutional layers extract spatial features.

About the MNIST Dataset:

> The MNIST dataset is a collection of **70,000 grayscale images of handwritten digits** (0-9), commonly used for training image classification models. Each image is 28x28 pixels and labeled with its corresponding digit. MNIST serves as a benchmark dataset for testing

> different machine learning and deep learning techniques, making it an ideal starting point for working with Convolutional Neural Networks (CNNs).

Python Code:

Copy

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize pixel values

# Define a simple CNN model
model = keras.Sequential([
    keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),
    keras.layers.MaxPooling2D((2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# Compile and train the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=[
model.fit(x_train[..., None], y_train, epochs=3, validation_data=(x_test[..., Non
```

Discussion Questions:

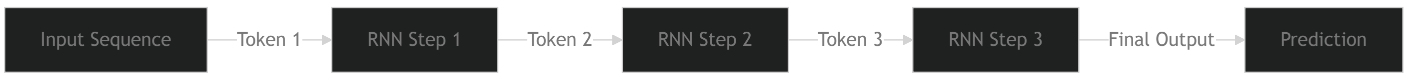
- What does each CNN layer do in this model?
- How does this architecture differ from a simple feedforward network?

The Rise of Transformers

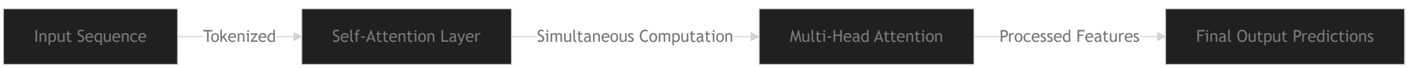
Why Transformers?

- Overcame the limitations of RNNs in processing sequential data.
- Introduced the concept of **self-attention**, which allows models to focus on important words or pixels.

Sequential Proccessing in RNNs



Parallel Processing in Transformers



> **Key Takeaway**: Transformers are faster than RNNs because they process all input tokens simultaneously using self-attention, whereas RNNs process tokens sequentially, making them slower and less efficient for long sequences.

Example Applications:

- **Text Processing:** GPT, BERT, T5
- **Image Analysis:** Vision Transformers (ViTs)

Coding Walkthrough: Fine-Tuning a Transformer for Text Classification

Objective:

- Load a pre-trained Transformer model (**DistilBERT**) and fine-tune it for sentiment classification.

Python Code:

Copy

```
from transformers import pipeline

# Load a pre-trained Transformer model for text classification
classifier = pipeline("text-classification", model="distilbert-base-uncased-finet
```

```
# Test with sample sentences
print(classifier("This movie was amazing!"))
print(classifier("I did not enjoy this product at all."))
```

Discussion Questions:

- What advantages do Transformers have over RNNs?
- Why are pre-trained models important in deep learning?

Summary & Key Takeaways

- Different deep learning architectures specialize in different data types.
- CNNs excel at **image processing**, while Transformers dominate **text and multimodal tasks**.
- **Choosing the right architecture is essential for model performance and efficiency**.
- Pre-trained models can **reduce training time and improve results**.

< Previous

Next >