Deep Learning Fundamentals II

Course:
INFO-6146 Tensorflow & Keras with Python

FANSHAWE

Developed by:
Mohammad Noorchenarboo

May 27, 2025

Current Section

Why Study the History of Deep Learning?

Deep learning didn't appear overnight. It evolved over decades through research breakthroughs, computational advances, and large-scale data availability.

Purpose

Understanding the history helps contextualize why certain architectures exist and what problems they were designed to solve.

Early Neural Networks (1950s–1980s)

Perceptron (1958): Introduced by Frank Rosenblatt. A simple model for binary classification.

Limitations:

- Could not solve non-linear problems (e.g., XOR)
- Training multi-layer networks was not feasible yet

Stagnation

In the 1970s, interest declined due to limited computational power and algorithmic tools.

Breakthrough: Backpropagation (1986)

Backpropagation: A method for computing gradients in multi-layer networks.

Key Paper

Rumelhart, Hinton, and Williams (1986): "Learning representations by back-propagating errors"

Impact

Enabled practical training of multi-layer neural networks, laying the foundation for modern deep learning.

LeNet (1998): First Convolutional Neural Network

Yann LeCun et al. designed LeNet-5 to recognize handwritten digits.
Key ideas:

- Local receptive fields (convolutions)
- Weight sharing
- Subsampling (pooling)

Application

Used to process checks by US banks.

AlexNet (2012): Deep Learning Revival

AlexNet (Krizhevsky et al.) won the ImageNet challenge in 2012 with a large CNN.

Why It Mattered

- Huge performance jump vs. traditional methods
- Trained on GPUs using ReLU, dropout, and data augmentation
- Sparked a wave of deep learning research and applications

Modern Architectures and Beyond

Important Milestones:

- **ResNet (2015):** Solved vanishing gradient with skip connections
- **GANs (2014):** Generated realistic images using adversarial training
- **Transformer (2017):** "Attention is All You Need" revolutionized NLP
- **GPT-3 (2020), ViT (2021), Stable Diffusion (2022):** Massive models for generative tasks

Example

Transformers are now used in text, vision, and audio tasks – including Chat-GPT.

Loading a Pretrained Model (Hugging Face)

Using BERT for text classification

```python
from transformers import pipeline

classifier = pipeline('text-classification',
                      model='bert-base-uncased')
print(classifier("I love this course!"))
```

Observation

Modern deep learning leverages pre-trained models to save time and improve accuracy.

Summary: Key Milestones in Deep Learning

Year & Milestone	Description
1958 – Perceptron	Simple binary classifier; inspired future work but limited
1986 – Backpropagation	Enabled training of multi-layer networks
1998 – LeNet-5	First successful CNN for digit recognition
2012 – AlexNet	ImageNet winner; revived DL with GPU training
2015 – ResNet	Introduced skip connections to solve vanishing gradients
2017 – Transformer	Enabled large-scale parallel NLP with attention
2020+ – GPT, ViT, Diffusion	Scaled DL models to new domains and generative AI

Current Section

When Is Deep Learning Better?

Deep learning is particularly powerful for complex, high-dimensional, and unstructured data.

Advantages

- Learns from raw data (e.g., images, text, audio)
- Scales with massive data and compute
- Enables end-to-end modeling without manual features

Key Advantages of Deep Learning

1. **Automatic Feature Extraction** Learns hierarchical features directly from data.
2. **End-to-End Learning** Minimizes need for domain-specific preprocessing.
3. **Scalability** Performance improves as data and model size grow.

Informative

Deep learning models improve with data and compute – unlike many traditional algorithms which saturate early.

Where Traditional ML Still Wins

Limitations of Deep Learning

- Requires a large dataset
- Slower training and higher compute costs
- Less interpretable

Strengths of Traditional ML

- Works well on small, tabular datasets
- Models are easier to explain (e.g., decision trees)
- Quick to train and deploy

Code Comparison: ML vs DL

Traditional ML with scikit-learn

```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_features, y)
```

Deep Learning with Keras

```python
from tensorflow.keras import layers, models
model = models.Sequential([
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

Comparison: Deep Learning vs. Traditional ML

Aspect	**Deep Learning vs. Traditional ML**
Feature Engineering	DL learns from raw data; ML requires manual feature crafting
Performance on Unstructured Data	DL excels with text, images, and audio; ML performs poorly without preprocessing
Interpretability	ML is more explainable; DL often acts as a black box
Data Requirement	DL needs large datasets; ML performs well with smaller data
Training Time & Compute	DL needs GPUs/TPUs; ML is lightweight and fast

Current Section

What Is an Artificial Neuron?

Inspired by biological neurons, an artificial neuron computes a weighted sum of its inputs and applies an activation function.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Where:

- x_i = input features
- w_i = weights
- b = bias
- f = activation function (e.g., ReLU, sigmoid)

Example

If inputs represent exam scores, a neuron could predict pass/fail based on learned weights.

What Is a Layer?

A **layer** is a group of neurons that process data in parallel. Deep networks stack multiple layers.

Types of layers:

- **Input layer:** Receives raw data
- **Hidden layers:** Transform data through neurons
- **Output layer:** Produces final prediction

Informative

Each hidden layer transforms the data space into a more abstract representation.

Activation Functions: Bringing Non-Linearity

Activation functions help networks learn complex patterns by introducing non-linearity.

Popular choices:

- ReLU (Rectified Linear Unit)
- Sigmoid
- Tanh
- Softmax (for classification output)

Example

ReLU allows the network to learn threshold-based patterns:

$$f(x) = \max(0, x)$$

Python Example: Simple MLP with Keras

MLP with two hidden layers

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(64, activation='relu', input_shape=(20,)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')  # binary classification
])
```

Example: Sentiment Classification (Text)

Example

Given a sentence like “This course is amazing!”, an MLP can learn to map text embeddings to a sentiment score (positive/negative).

Pipeline steps:

- Tokenize and embed text (e.g., using BERT or Word2Vec)
- Feed vector into MLP
- Output a score between 0 and 1 (sentiment)

Example: Predicting Housing Prices (Tabular Data)

Example

Train an MLP using features like size, location, number of rooms to predict house price.
Input: 10 numerical features **Output:** Continuous price value (regression)

MLP for regression

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(10,)),
    Dense(1)  # no activation for regression
])
```

Activation Functions: Summary

Activation Function	Behavior and Use Case
ReLU	Fast and simple. Used in most hidden layers. Output: $\max(0, x)$
Sigmoid	Outputs between 0 and 1. Used for binary classification.
Tanh	Outputs between -1 and 1. Often used in older models.
Softmax	Converts logits into probabilities. Used in multi-class classification.

Current Section

What Are Neural Network Architectures?

Different problems require different network structures.
Three common architectures:

- **ANN (Artificial Neural Network)**: General-purpose, tabular data
- **CNN (Convolutional Neural Network)**: Images and spatial data
- **RNN (Recurrent Neural Network)**: Sequences and time series

Why This Matters

Choosing the right architecture is key to capturing the structure of your data.

ANN: Artificial Neural Networks

ANNs are feedforward networks that connect every neuron in one layer to every neuron in the next.

Used for:

- Tabular data
- Simple regression or classification

Example

Predicting diabetes using patient health metrics (numerical features).

CNN: Convolutional Neural Networks

CNNs are designed to process grid-like data, such as images.
Key Components:

- Convolutional layers (extract local features)
- Pooling layers (downsample)
- Fully connected layers (final classification)

Example

Classify handwritten digits in MNIST dataset using pixel patterns.

Simple CNN model

```
model = Sequential([
  Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
  MaxPooling2D((2,2)),
  Flatten(),
  Dense(10, activation='softmax')
])
```

RNN: Recurrent Neural Networks

RNNs are suited for sequence data – they maintain internal memory across time steps.
Used for:

- Time series forecasting
- Natural language processing
- Speech recognition

Example

Predicting the next word in a sentence or the stock price for tomorrow.

Simple RNN with Keras

```
model = Sequential([
  SimpleRNN(64, input_shape=(timesteps, features)),
  Dense(1)
])
```

Comparison: ANN vs CNN vs RNN

Architecture	Best For
ANN	General-purpose structured data (e.g., tabular features)
CNN	Image classification, object detection, spatial pattern recognition
RNN	Time series, sequences, text, speech – where order matters