Deep Learning vs. ML: Real-World Use, Evaluation, and Tools

Course:
INFO-6146 Tensorflow & Keras with Python

Developed by:
Mohammad Noorchenarboo

May 27, 2025

Current Section

What Is the Core Difference?

Traditional Machine Learning (ML) and **Deep Learning (DL)** both fall under the umbrella of AI, but differ in how they learn patterns and handle data.

High-Level Difference

ML requires structured data and manual feature extraction. DL uses multi-layered networks to learn patterns automatically from raw data.

Both use optimization to reduce error, but the depth and data types they handle are very different.

Data Requirements and Representation

Traditional ML:

- Works best on structured/tabular data
- Requires manual preprocessing and feature engineering

Deep Learning:

- Excels on unstructured data (images, audio, text)
- Learns features during training from raw input

Example

A logistic regression model predicting diabetes needs engineered features (e.g., BMI, age). A CNN can learn from raw image scans of the retina.

Architectural Complexity

ML Models:

- Shallow structures (e.g., decision trees, linear models)
- Few parameters

DL Models:

- Deep layered networks (ANNs, CNNs, RNNs, Transformers)
- Millions to billions of parameters

Informative

Deeper networks allow DL models to learn complex, hierarchical abstractions – something ML models struggle with.

Compute Needs and Scalability

Traditional ML:

- Trains quickly on CPUs
- Requires little memory

Deep Learning:

- Requires GPUs/TPUs for efficient training
- Scales better with large datasets and distributed training

Warning

Training deep models without proper hardware may be impractical for large datasets.

Comparison Table: ML vs DL

Aspect	**ML vs DL**
Data Type	ML: Structured/tabular DL: Unstructured (image, text, audio)
Feature Engineering	ML: Manual DL: Learns automatically
Model Complexity	ML: Shallow models DL: Deep, multi-layered networks
Compute Requirements	ML: CPU-friendly DL: Needs GPUs/TPUs
Interpretability	ML: Transparent DL: Often black-box

Current Section

Why Is Deep Learning So Powerful?

Deep Learning (DL) has redefined what machines can do – especially in vision, language, and generative tasks.

Key Strengths

- Learns features directly from raw data (end-to-end)
- Handles unstructured data (e.g., text, images, audio)
- Performance scales with data and model size

DL models are flexible function approximators capable of capturing complex patterns that classical ML struggles with.

End-to-End Learning: No Manual Pipelines

DL learns to optimize directly from input to output. No need to manually design and feed in features.

Example

In traditional NLP: text $\rightarrow$ tokens $\rightarrow$ TF-IDF* $\rightarrow$ ML classifier. With DL (e.g., BERT), raw text $\rightarrow$ embeddings $\rightarrow$ classification in one model.
* TF-IDF (Term Frequency-Inverse Document Frequency)

Impact

Simplifies development and reduces domain-specific preprocessing overhead.

Handling Unstructured Data

Unstructured data includes:

- Images (pixels)
- Text (sentences, paragraphs)
- Audio (waveforms, spectrograms)
- Videos (frames + time)

Example

CNNs process raw pixel arrays for facial recognition. Transformers like Whisper convert raw speech to text.

Traditional ML struggles with this data unless it's converted to tabular form.

Scalability: More Data, Better Performance

Deep learning models tend to improve as:

- You add more labeled data
- You increase the number of layers/parameters
- You provide faster compute (e.g., GPUs, TPUs)

Real-World Proof

GPT-4, AlphaFold, and DALLE all benefit from massive datasets and deep architectures that grow with scale.

Limitations of Deep Learning

Despite its power, DL has several practical drawbacks.

Core Limitations

- Requires large labeled datasets
- Expensive to train and maintain
- Difficult to interpret (black-box behavior)
- Prone to adversarial attacks and spurious correlations

Warning

DL models may fail catastrophically if training data is biased or incomplete.

Comparison Summary: DL Strengths vs. Limitations

Aspect	Deep Learning Characteristics
Learning Process	Learns features automatically (end-to-end)
Data Flexibility	Handles unstructured data directly
Performance	Improves with scale (data, model size, compute)
Resource Demand	Requires significant GPU/TPU resources
Transparency	Low interpretability; hard to debug

Current Section

Why Model Selection Matters

Choosing between ML and DL depends on the problem, data, goals, and available resources.

Key Idea

The "best" model is not the most complex one – it's the one that solves your task reliably, efficiently, and with acceptable cost.

This section will help you match your data and constraints to the right algorithm family.

Basic Rule of Thumb

Use Traditional ML when:

- Data is structured (tables, spreadsheets)
- Dataset is small (hundreds to low thousands of samples)
- Interpretability is a priority
- You need fast training and deployment

Use Deep Learning when:

- Data is unstructured (images, text, audio)
- You have large datasets (thousands to millions of samples)
- You can train on GPU or TPU
- Performance is prioritized over explainability

Applied Examples by Domain

Task Type	Suggested Model Approach
Predict hospital readmission from EHRs	Traditional ML (structured, explainable, small-medium data)
Classify sentiment from tweets	DL (Transformers handle sequential text and context)
Detect fraud in financial transactions	Start with XGBoost/LightGBM, escalate to DL if scale demands
Object detection in traffic camera feed	DL (CNN-based object detectors or vision transformers)
Student grade prediction from CSV	ML (e.g., logistic regression, tree-based models)

Python Workflow: Traditional ML Baseline First

Try ML first with scikit-learn

```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
score = model.score(X_test, y_test)
```

Good Practice

Always start with a fast baseline (e.g., decision tree, logistic regression). Then escalate to DL only if needed.

When and How to Switch to Deep Learning

Consider DL if:

- Your ML model underfits or saturates on performance
- Feature engineering becomes a bottleneck
- You need to process raw images, audio, or sequences
- You want to reuse pretrained models (e.g., Hugging Face Transformers)

Start simple with Keras MLP

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

model = Sequential([
    Dense(64, activation='relu', input_shape=(X.shape[1],)),
    Dense(1, activation='sigmoid')
])
```

Current Section

Where Is Deep Learning Used Today?

Deep learning powers many of the most advanced systems in AI today.

Industries Transformed by DL

- Healthcare
- Finance
- Autonomous vehicles
- Education and personalization
- Creative industries (music, art, design)

Its ability to learn from raw, high-dimensional data makes it ideal for complex, noisy environments.

Use Case: Computer Vision (CNNs)

Example

A convolutional neural network (CNN) classifies skin lesions as benign or malignant using raw medical images.

Other applications:

- Face recognition (e.g., FaceID)
- Autonomous driving (object detection in traffic)
- Industrial defect detection

Loading pretrained CNN from Keras

```python
from tensorflow.keras.applications import ResNet50

model = ResNet50(weights='imagenet')
```

Use Case: Natural Language Processing (Transformers)

Example

Use a transformer model (e.g., BERT) to classify customer support tickets by urgency level.

Other NLP tasks:

- Translation (Google Translate)
- Text summarization
- Chatbots and virtual assistants

Text classification using Hugging Face

```python
from transformers import pipeline
classifier = pipeline("text-classification")
classifier("This course is amazing!")
```

Use Case: Healthcare (Multimodal DL)

DL in healthcare:

- Predicting disease from EHR + imaging
- Medical chatbots (e.g., Med-PaLM)
- Genome pattern recognition

Example

Train a CNN on chest X-rays and combine it with a tabular model of patient vitals using a hybrid architecture.

Benefit: Captures complex nonlinear patterns missed by rule-based systems.

Use Case: Autonomous Systems

Self-driving cars use DL to:

- Detect lane markings, pedestrians, and vehicles
- Fuse input from multiple sensors (camera, LiDAR, radar)

Example

Tesla's Dojo supercomputer trains neural nets for driving from millions of hours of video data.

DL enables systems to learn directly from driving behavior and vision, rather than using programmed rules.

Use Case Summary Table

Domain	DL Application Example
Healthcare	Disease prediction from X-ray + EHR
Vision	Object detection, facial recognition
NLP	Chatbots, text classification, summarization
Autonomous Vehicles	Driving policy learning from video
Finance	Fraud detection, risk modeling, algorithmic trading

Current Section

Why Talk About Failure?

Even the most advanced deep learning models can fail under real-world conditions.

DL is not bulletproof

- Biased training data $\rightarrow$ biased predictions
- Domain mismatch $\rightarrow$ poor generalization
- Noise, outliers, or adversarial examples $\rightarrow$ model instability

Critical Thinking

Understanding when DL can fail helps you design safer, more robust systems.

Failure: Biased or Incomplete Data

DL learns what it sees. If training data is flawed, so is the model.

Example

A facial recognition system trained mostly on lighter skin tones may fail to recognize darker-skinned individuals.

Consequence

Bias can lead to unfair decisions – in hiring, justice, finance, or healthcare.

Failure: Domain Mismatch

Training on one domain and deploying in another often fails.

Example

A sentiment classifier trained on movie reviews performs poorly on legal documents due to language mismatch.

Warning

Always validate model performance on the target domain before deployment.

Failure: Adversarial Attacks

Adversarial attacks are small, often invisible changes to input data that cause DL models to fail.

Example

Adding noise to a stop sign image can fool a CNN into predicting “speed limit” instead of “stop”.

Impact

This is a critical vulnerability for DL in safety-sensitive domains like self-driving cars or medical imaging.

Failure: Overconfidence and Lack of Uncertainty

DL models tend to be overconfident in their predictions.

Example

A model might assign 99% confidence to a completely irrelevant or noisy input.

Caution

Confidence scores from DL models do not always reflect true reliability.

Tip

Use uncertainty estimation methods like Monte Carlo Dropout or ensemble averaging in high-stakes systems.

Summary: Common DL Failure Scenarios

Failure Type	Description
Bias in Training Data	Model inherits systemic bias (e.g., race, gender)
Domain Shift	Trained on one distribution, deployed on another
Adversarial Attacks	Small perturbations fool the model
Overconfidence	High certainty on incorrect predictions
Label Noise or Poor Preprocessing	DL learns from noise or bad inputs

Current Section

Why So Many Libraries?

The deep learning ecosystem is broad. Different libraries are optimized for different needs – from fast prototyping to large-scale deployment.

What You'll Learn

How to choose the right library or toolkit for:

- Training models (ML and DL)
- Working with pretrained models
- Building custom architectures
- Exporting and deploying

scikit-learn: Traditional ML Foundation

scikit-learn (sklearn) is the go-to library for classical ML in Python.

Highlights

- Easy to use, consistent API
- Includes preprocessing, model selection, pipelines
- Models: SVM, Random Forest, Logistic Regression, etc.

Example: Random Forest

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Keras and TensorFlow: Easy to Build, Scale to Production

Keras (running on TensorFlow backend) is ideal for building DL models quickly and intuitively.

Strengths

- High-level API with great documentation
- Supports both Sequential and Functional APIs
- Easy access to prebuilt models (e.g., ResNet, MobileNet)
- Deployable with TensorFlow Serving, TF Lite

Simple MLP with Keras

```
model = Sequential([
  Dense(64, activation='relu', input_shape=(10,)),
  Dense(1, activation='sigmoid')
])
```

PyTorch: Research Flexibility and Control

PyTorch is widely used in research and cutting-edge applications.

Strengths

- Dynamic computation graph (eager execution)
- Great for custom models and debugging
- Strong ecosystem (TorchVision, TorchText, Lightning)

Example

PyTorch is used in large-scale models like GPT-2, GPT-3, and Stable Diffusion.

Hugging Face Transformers: Pretrained Model Hub

Hugging Face provides state-of-the-art NLP and vision models with minimal code.

Strengths

- Thousands of pretrained models (BERT, RoBERTa, ViT, Whisper, etc.)
- Easy integration for inference and fine-tuning
- Large dataset hub and tokenizers

Sentiment classification with BERT

```python
from transformers import pipeline
classifier = pipeline("text-classification")
classifier("I love this course!")
```

TensorFlow Hub and ONNX: Interoperability and Reuse

TensorFlow Hub: Repository of reusable DL models. **ONNX (Open Neural Network Exchange):** Format to move models between frameworks.

Use Cases

- Deploy Keras models with TF Serving or TF Lite
- Export PyTorch models to ONNX for edge devices
- Share universal models across platforms

Summary: Library Comparison Table

Library	Best For	Key Feature
scikit-learn	Classical ML on structured data	Pipelines, metrics, preprocessing
Keras (TensorFlow)	Fast DL prototyping, production deployment	High-level API, mobile support
PyTorch	Research, custom architectures	Eager execution, modular design
Hugging Face Transformers	Pretrained models (NLP, vision, audio)	Massive model hub + APIs
TensorFlow Hub	Reuse/share Keras models	Production-ready modules
ONNX	Interoperability	Export models between frameworks

Current Section

TensorFlow Playground: Visualizing ANNs

TensorFlow Playground is a web tool to help you intuitively understand:

- Learning rate
- Activation functions
- Regularization
- Hidden layers and neurons

Try it at:

`https://playground.tensorflow.org`

Playground Example: Activation Functions

Activation functions introduce non-linearity.

Example

Try comparing ReLU, tanh, and sigmoid in the Playground to see how the network adapts to curved vs. linear boundaries.

Tip

ReLU is widely used due to efficiency and sparsity. Tanh is good for centered data.

Playground Example: Learning Rate

Learning rate controls the size of weight updates.

Example

Try setting the learning rate too high or too low in the Playground and observe model convergence or instability.

Warning

High learning rates can cause the model to oscillate or diverge entirely.

Playground Example: Regularization and Architecture Depth

- Add more hidden layers and neurons
- Apply L2 regularization
- Use noise in the dataset

Example

Observe how deeper networks with regularization create smoother, more general decision boundaries.