# Intro to Deep Learning

## Instructions:

- Answer all questions **clearly and concisely.**
- Support answers with diagrams, equations, or examples where necessary.
- Cite sources (especially book chapters mentioned) where applicable.
- Write in your **own words** — avoid copying from the internet.
- Submit your completed assignment in **Github Repo Link (PDF+Code Notebook)**

---

## Section A: Fundamentals of Deep Learning

### Q1. What is Deep Learning?

Deep learning is a subset of machine learning that uses artificial neural networks (ANNs) with multiple hidden layers ("deep" architectures) to learn hierarchical representations of data. Unlike traditional machine learning—which relies on manual feature engineering—deep learning automatically extracts features from raw data (e.g., pixels in images, words in text) through its layered structure. The term "deep" refers to the number of hidden layers, which enables the model to capture complex patterns by building on simpler ones.

### Q2. Key Components of Deep Learning

The main components of a deep learning model include:

**Neural Networks/Deep Neural Networks (DNNs):** Composed of interconnected "neurons" organized into layers (input, hidden, output).

<u>**Layers:**</u>

**1. Input Layer:** Receives raw data (e.g., image pixels).
**2. Hidden Layers:** Process data through weighted connections and activation functions (learn hierarchical features).
**3. Output Layer:** Produces final predictions (e.g., class probabilities).
**4. Activation Functions (e.g., ReLU, Sigmoid):** Introduce non-linearity, allowing the model to learn complex relationships.
**5. Loss Functions (e.g., Cross-Entropy, Mean Squared Error):** Quantify the error between predictions and true labels, guiding optimization.
**6. Optimizers (e.g., SGD, Adam):** Adjust model parameters (weights/biases) to minimize loss during training.

## Q3. Understanding Neural Networks, Neurons, and the Perceptron

**Neuron:** A basic unit in a neural network. It takes inputs (**$x_i$**), multiplies them by weights (**$w_i$**), sums them (with a bias **b**), and applies an activation function (**f**) to produce an output:

$$y = f(\sum wixi + b)$$

**Perceptron Model:** A single-layer neural network for binary classification. It computes a weighted sum of inputs, adds a bias, and passes the result through a step function (e.g., Heavisine) to output 0 or 1.

**Multiple Perceptrons Forming a Neural Network:** Connecting multiple perceptrons into layers (input → hidden → output) creates a neural network. Hidden layers combine simple perceptron outputs to learn complex patterns (e.g., edges → shapes → objects in images).

## Q4. Hierarchical Representations

Hierarchical representations in deep learning refer to how models learn features at different levels of abstraction. Low-level features (e.g., edges, lines in images) are detected in early layers. These are combined in subsequent layers to form mid-level features (e.g., shapes, textures), and finally high-level features (e.g., objects, faces) in deeper layers. This structure mimics human visual processing, where simple elements build into complex concepts.

## Q5. Fitting Parameters using Backpropagation

Backpropagation is an algorithm to train neural networks by computing gradients of the loss function with respect to model parameters (weights/biases).

**Steps:**
**Forward Pass:** Compute predictions and loss.
**Backward Pass:** Use the chain rule to calculate gradients of the loss w.r.t. each parameter, starting from the output layer and moving backward.
**Parameter Update:** Adjust weights using an optimizer (e.g., SGD) to reduce loss.

## Q6. Non-Convex Functions

Non-convex functions have multiple local minima, saddle points, and flat regions. In deep learning, the loss landscape is non-convex due to the complexity of neural networks. Optimization is challenging because gradient-based methods may get stuck in local minima (suboptimal solutions) instead of finding the global minimum, leading to inconsistent model performance.

## Q7. Training and Model Optimization

### Training Process:

**Forward Pass:** Input data propagates through the network to generate predictions.
**Loss Calculation:** Compare predictions with true labels using a loss function.
**Backward Pass:** Compute gradients via backpropagation.
**Parameter Update:** Adjust weights using an optimizer.

### Optimization Techniques:

**Dropout:** Randomly deactivates neurons during training to prevent overfitting.
**SGD (Stochastic Gradient Descent):** Updates parameters using mini-batches for efficiency.
**Learning Rate Scheduling:** Gradually reduces the learning rate to stabilize training.
**Batch Normalization:** Normalizes layer inputs to reduce "internal covariate shift," speeding up training.

## Q8. Challenges and Requirements

### Challenges:

**Data Requirements:** Need large, labeled datasets for training.
**Interpretability:** Models act as "black boxes," making it hard to understand decisions.
**Model Complexity:** High computational cost and risk of overfitting.
**Computational Cost:** Requires significant GPU resources for training large models.

### Essential Requirements:

**Quality Data:** Large, diverse, and well-labeled datasets.
**Computational Resources:** GPUs/TPUs for efficient training.
**Expertise:** Knowledge of neural network design, optimization, and evaluation.
**Regularization:** Techniques (e.g., dropout, weight decay) to prevent overfitting.

## Section B: Deep Learning Frameworks & Implementation

### Q9. Deep Learning Frameworks

**Three popular frameworks:**

**TensorFlow:** Developed by Google; supports production deployment, flexible architecture, and tools like TensorFlow Lite (mobile).**PyTorch:** Preferred for research due to dynamic computation graphs, ease of debugging, and strong community support.
**Keras:** High-level API running on TensorFlow; simplifies model building with user-friendly syntax.

**Why Used:** They provide pre-built layers, optimizers, and utilities, reducing boilerplate code and accelerating development.

### Q10. Building Neural Networks with Keras and TensorFlow (Book Reference: Chapter 3)

**Steps:**

**Define Layers:** Use tf.keras.layers (e.g., Dense, Conv2D) to create input, hidden, and output layers.
**Compile the Model:** Specify loss function, optimizer, and metrics (e.g., model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])).
**Train and Evaluate:** Fit the model to data (model.fit()) and evaluate performance (model.evaluate()).

### Q11. Data Preprocessing, Feature Engineering, and Feature Learning (Book Reference: Chapter 4)

**Data Preprocessing:** Cleaning and transforming raw data (e.g., normalization, handling missing values) to make it suitable for modeling.
**Feature Engineering:** Manually creating new features from existing data (e.g., polynomial features) to improve model performance.
**Feature Learning:** Automatic extraction of relevant features by the model (e.g., CNNs learning image features).
**Importance:** Preprocessing ensures data quality; feature engineering enhances model input; feature learning reduces manual effort and captures complex patterns.

---

## Section C: Image Classification Concepts

### Q12. What is Image Classification?

Image classification is the task of assigning a label (e.g., "cat," "dog") to an input image.

**Real-World Examples:**
**Medical Imaging:** Classifying X-rays as normal or abnormal.
**Retail:** Identifying products in store images for inventory management.

### Q13. Introduction to ImageNet

ImageNet is a large-scale dataset with millions of labeled images across thousands of categories. It revolutionized deep learning research by enabling breakthroughs like AlexNet (2012), which demonstrated the power of deep CNNs. The ImageNet Challenge (ILSVRC) standardized benchmarking, driving advances in computer vision.

### Q14. Classification using a Single Linear Threshold (Perceptron)

A single-layer perceptron performs binary classification by computing a weighted sum of inputs plus a bias, then applying a step function:

$$y = \{ 1 \; if \sum wixi + b \geq 0 \; otherwise \; 0$$

It decides the output based on whether the weighted sum exceeds a threshold.

### Q15. How Interpretable Are Deep Learning Features?

Deep learning models are "black boxes" because their inner workings are opaque. One interpretation method is Grad-CAM (Gradient-weighted Class Activation Mapping), which highlights important regions in an image by computing gradients of the target class score with respect to convolutional layer activations.

### Q16. Manipulating Deep Nets

Adversarial examples are inputs intentionally perturbed (e.g., adding noise to an image) to mislead the model.

**To improve robustness:**
Use adversarial training (incorporate adversarial examples into the training set).
Apply regularization techniques (e.g., dropout).

### Q17. Transfer Learning

Transfer learning reuses pre-trained models (e.g., on ImageNet) for new tasks with limited data. It leverages learned features, reducing the need for large datasets. Common Pre-Trained Models: ResNet, VGG.

## Section D: Applications of Deep Learning

### Q18. Applications in Data Science

**Speech Recognition:** Converting speech to text (e.g., virtual assistants).
**NLP:** Sentiment analysis, machine translation.
**Healthcare:** Diagnosing diseases from medical images (e.g., MRI scans).

---

### Q19. Case Study 1: Data Scientist Employee Attrition

a. **Problem Type:** Classification (predicting binary outcome: leave/stay).
b. **Model Architecture:** Feedforward neural network with input layer (features), hidden layers (ReLU activation), output layer (sigmoid for binary classification).
c. **Loss Function:** Binary cross-entropy (suitable for binary classification).
d. **HR Management Impact:** Identify at-risk employees, implement retention strategies (e.g., tailored benefits), and reduce turnover costs.

---

## Section E: Practical Project

### Q20. Project – Handwritten Digit Classification (MNIST Dataset)

**Steps:**

**Load/Preprocess:** Normalize pixel values (0–1), split into train/test sets.
**Design ANN:** Input layer (784 neurons), hidden layers (e.g., 128 neurons with ReLU), output layer (10 neurons with softmax).
**Train/Evaluate:** Compile with categorical cross-entropy, train using Adam optimizer, evaluate accuracy.
**Visualize:** Plot confusion matrix, accuracy/loss curves, sample predictions.
**Deliverables:** Model summary, architecture diagram, accuracy graphs, sample predictions, and result explanation.

---

## Section F: Reflection (Bonus)

## Q23. Your Thoughts on Deep Learning

### What Makes Deep Learning Powerful?

Ability to learn complex patterns from unstructured data (images, text), automation of feature engineering, and state-of-the-art performance in tasks like image recognition and NLP.

### Ethical/Practical Issues

Bias in training data leading to unfair outcomes, privacy concerns (e.g., facial recognition), computational environmental impact, and lack of interpretability hindering trust.

✅ ***End of Assignment***