**National University of Technology**



**Computer Science Department**

Semester Spring– 2025

**Program:** Artificial intelligence

**Course:** Programming for AI

**Course Code:** CS282

**Assignment - 03**

| Submitted To: | Submitted By: |
| --- | --- |
| Umar Aftab | Muhammad Ahad Imran |
|  | F23607034 |















**How Flux.jl Simplifies Deep Learning**

Flux.jl provides a clean and intuitive way to define models using Chain and Dense layers. It supports automatic differentiation via Zygote.jl, ensuring seamless backpropagation. Additionally, GPU acceleration is built-in—just move the model and data to the GPU without modifying code.

### Key Features

#### Automatic Differentiation (AD)

**Definition:** Automatic differentiation is a computational technique for efficiently and accurately evaluating derivatives (gradients) of functions. It works by decomposing functions into elementary operations and applying the chain rule.

**Key Points in Flux.jl:**

* Utilizes Zygote.jl for AD, performing source-to-source transformation to compute gradients.
* Supports dynamic AD—works seamlessly with native Julia control flow (e.g., loops, conditionals).
* Eliminates the need to manually construct a computational graph.
* Essential for training models using gradient descent, backpropagation, and optimization algorithms.

**Benefits:**

* Reduces human error in derivative calculations.
* Enables easy experimentation with complex model architectures.
* Supports differentiation through any differentiable Julia code.

#### GPU Acceleration

**Definition:** GPU acceleration offloads compute-intensive operations (such as matrix multiplications and convolutions) to the Graphics Processing Unit (GPU), which is optimized for parallel computation.

**Key Points in Flux.jl:**

* Leverages CUDA.jl, an interface for NVIDIA’s CUDA API.
* Allows running the entire training loop (model, data, loss, gradients) on the GPU.
* Integrates seamlessly with Flux models—no extensive code modifications required.

**Benefits:**

* Significantly speeds up deep neural network training, especially for large datasets or complex models.
* Efficiently utilizes memory and computation resources on modern GPUs.
* Fully compatible with Julia, enabling high-performance computing in a streamlined environment.