**BUILDING A COMPILER FOR DOMAIN SPECIFIC LANGUAGES(DSL) USING DEEP LEARNING**

A CAPSTONE PROJECT REPORT

# (compiler design for deep learning models-CSA 1484)

***Submitted to***

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING**

***By***

**T.Rechan reddy (192211812),**

**Sk.Sohail(192210074),**

**N.Vamsi Chowdari(192210317).**

***Course Faculty***

**Dr Michael.G**



**SAVEETHA SCHOOL OF ENGINEERING, SIMATS, CHENNAI - 602105**

**JULY-2024**

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUE OF MEDICAL AND TECHNICAL SCIENCES,CHENNAI - 602105**

**BONAFIDE CERTIFICATE**

Certified that this project report **“BUILDING A COMPILER FOR DOMAIN SPECIFIC LANGUAGES(DSL) USING DEEP LEARNING”**is the Bonafide work of **“R.Raj mohan, M.Kesavan”** who carried out the project work under my supervision.

**Submitted to**

**MICHAEL**

(Course Faculty)

Department of Deep Learning, Saveetha School of Engineering, SIMATS

## SIGNATURE of Course Faculty

**ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N. M. Veeraiyan**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. Deepak Nallaswamy,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. Ramya Deepak,** Saveetha School of Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B. Ramesh,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Murali**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

**STUDENT NAME’s**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **CHAPTER**  **NO** | **TITLE** |
| **1** | **Introduction** |
| **2** | **Project Description and Goals** |
| **3** | **Technical Specifications:** |
| **4** | **Design Approach and Details** |
| **5** | **Schedule, Tasks, and Milestones:** |
| **6** | **Project Demonstration:** |
| **7** | **Cost Analysis:** |
| **8** | **Result:** |
| **9** | **Discussion:** |
| **10** | **Conclusion:** |

**Introduction:**

Building a compiler for Domain-Specific Languages (DSLs) using deep learning represents a cutting-edge approach that leverages neural networks to enhance the efficiency and accuracy of code translation and optimization. Traditional compiler construction relies heavily on manually defined grammar rules and syntactic structures, which can be limiting and labor-intensive, particularly for specialized languages. By integrating deep learning techniques, such as neural networks, into the compiler design, we can automate and refine the process of parsing, semantic analysis, and code generation. This approach enables the compiler to learn from large datasets of code examples, adapt to complex and evolving DSLs, and potentially uncover patterns and optimizations that are not easily captured by conventional methods. Ultimately, this fusion of deep learning and compiler technology aims to streamline the development of DSLs, making it more accessible and efficient for specialized applications in various fields.

**Project Description and Goals:**

The project aims to design and implement a compiler for a Domain-Specific Language (DSL) leveraging deep learning techniques. This involves developing a deep learning model capable of parsing, understanding, and transforming DSL code into executable output or another intermediate representation. The goal is to streamline the development and optimization of DSLs by utilizing advanced AI techniques to automate and enhance various stages of the compilation process.

DSL Specification and Design:

Define the Domain: Identify and articulate the domain or specific problem area for which the DSL is being created. Examples include finance, bioinformatics, or web development.

Design Language Syntax: Create the syntax and semantics of the DSL. This includes defining the grammar, constructs, and language features tailored to the domain.

**Data Collection and Preprocessing:**

**Gather DSL Code Samples:**

Collect a diverse set of code examples written in the DSL to serve as training data. This may include both synthetic examples and real-world code if available.

**Preprocess Data:**

Clean and preprocess the code samples to make them suitable for training deep learning models. This may involve tokenization, normalization, and encoding.

**Parsing and Analysis:**

Implement a parsing module that uses the trained deep learning model to analyze and interpret DSL code.

**Intermediate Representation:**

Develop an intermediate representation of the DSL code that can be used for optimization and code generation.

**Code Generation:**

Create a code generation component that transforms the intermediate representation into executable code or another desired output format.

**Optimization:**

Incorporate optimization techniques to enhance the performance of the generated code. This could include applying AI-driven optimization strategies based on deep learning insights.

**Achievements and Leaderboards:**

Implement a system for tracking achievements and high scores to encourage replayability and competition.

**Technical Specifications:**

Brick n Ball Smash is developed using Unity, leveraging its robust engine for seamless 2D physics and fluid animations. The game is optimized for both mobile and desktop platforms, supporting iOS, Android, Windows, and macOS, with responsive touch and keyboard controls. It features high-definition graphics and a dynamic sound system for an immersive experience. The game utilizes a modular level design, allowing easy updates and expansions. It employs an efficient collision detection system for precise gameplay and integrates cloud-saving features for cross-device progress synchronization. Additionally, Brick n Ball Smash supports social media integration for sharing achievements and leaderboards for competitive play.

**Platform Compatibility:**

"Brick n Ball Smash" is available on multiple platforms, including mobile devices and desktops. Enjoy seamless gameplay across iOS, Android, and Windows for endless fun.

**Design Approach and Details:**

"Brick n Ball Smash" features a classic brick-breaking design with intuitive paddle controls and vibrant visuals. The game emphasizes engaging gameplay through varied levels, strategic power-ups, and dynamic obstacles.

**Schedule, Tasks, and Milestones:**

Tasks and Milestones: Complete core gameplay mechanics, implement visual assets and sound effects, and finalize by testing and polishing before release.

**Project Demonstration:**

"Brick n Ball Smash" is an engaging arcade game where players use a paddle to bounce a ball and destroy bricks. The design project demonstrates the game's intuitive mechanics and vibrant visuals through dynamic levels and interactive gameplay.

**Cost Analysis:**

Designing "Brick n Ball Smash" involves costs for concept development, art, sound, and programming, which can vary widely depending on the complexity and quality desired. A basic version may cost $10,000-$20,000, while a more polished game could exceed $50,000.

**Result:**

Brick n Ball Smash is a dynamic arcade game where players use a paddle to bounce a ball and break bricks, aiming to clear each level. With vibrant graphics and power-ups, it challenges players to strategize and achieve high scores.

**Discussion:**

"Brick n Ball Smash" is an addictive arcade game where players bounce a ball off a paddle to break bricks and clear levels. With vibrant graphics and challenging gameplay, it combines classic brick-breaking fun with exciting power-ups and increasing difficulty..

**Summary:**

"Brick n Ball Smash" is an addictive arcade game where players use a paddle to bounce a ball and break colorful bricks. Clear all the bricks across dynamic levels while collecting power-ups and aiming for high scores.

**Conclusion:**

"Brick n Ball Smash" offers addictive arcade action with vibrant graphics and dynamic levels. Players must master their reflexes and strategy to break bricks and achieve high scores.

**Code:**

**#include <SFML/Graphics.hpp>**

**#include <vector>**

**bool leftOrRightCollision(const sf::CircleShape& ball, int windowWidth);**

**bool topCollision(const sf::CircleShape& ball);**

**const int windowWidth = 800;**

**const int windowHeight = 600;**

**const float paddleWidth = 100.f, paddleHeight = 20.f, ballRadius = 10.f, brickWidth = 60.f, brickHeight = 20.f;**

**const int rowCount = 4, columnCount = 11;**

**struct Brick {**

**sf::RectangleShape shape;**

**bool destroyed{false};**

**Brick(float mX, float mY) {**

**shape.setPosition(mX, mY);**

**shape.setSize({brickWidth, brickHeight});**

**shape.setFillColor(sf::Color::Yellow);**

**shape.setOrigin(brickWidth / 2.f, brickHeight / 2.f);**

**}**

**};**

**int main() {**

**sf::RenderWindow window(sf::VideoMode(windowWidth, windowHeight), "Brick 'n Ball Game");**

**window.setFramerateLimit(60);**

**sf::RectangleShape paddle({paddleWidth, paddleHeight});**

**paddle.setFillColor(sf::Color::Red);**

**paddle.setPosition(windowWidth / 2.f, windowHeight - 50);**

**paddle.setOrigin(paddleWidth / 2.f, paddleHeight / 2.f);**

**sf::CircleShape ball(ballRadius);**

**ball.setFillColor(sf::Color::Green);**

**ball.setPosition(windowWidth / 2, windowHeight / 2);**

**sf::Vector2f ballVelocity(-5.0f, -5.0f);**

**std::vector<Brick> bricks;**

**for (int iX = 0; iX < columnCount; ++iX) {**

**for (int iY = 0; iY < rowCount; ++iY) {**

**bricks.emplace\_back((iX + 1) \* (brickWidth + 3) + 22, (iY + 2) \* (brickHeight + 3));**

**}**

**}**

**while (window.isOpen()) {**

**sf::Event event;**

**while (window.pollEvent(event)) {**

**if (event.type == sf::Event::Closed)**

**window.close();**

**}**

**if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left) && paddle.getPosition().x - paddleWidth / 2 > 0) {**

**paddle.move(-6.f, 0.f);**

**}**

**if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right) && paddle.getPosition().x + paddleWidth / 2 < windowWidth) {**

**paddle.move(6.f, 0.f);**

**}**

**ball.move(ballVelocity.x, ballVelocity.y);**

**if (leftOrRightCollision(ball, windowWidth)) ballVelocity.x \*= -1;**

**if (topCollision(ball)) ballVelocity.y \*= -1;**

**if (ball.getGlobalBounds().intersects(paddle.getGlobalBounds())) ballVelocity.y \*= -1;**

**for (auto& brick : bricks) {**

**if (!brick.destroyed && ball.getGlobalBounds().intersects(brick.shape.getGlobalBounds())) {**

**brick.destroyed = true;**

**ballVelocity.y \*= -1;**

**break;**

**}**

**}**

**window.clear();**

**for (auto& brick : bricks)**

**if (!brick.destroyed)**

**window.draw(brick.shape);**

**window.draw(paddle);**

**window.draw(ball);**

**window.display();**

**}**

**return 0;**

**}**

**bool leftOrRightCollision(const sf::CircleShape& ball, int windowWidth) {**

**return ball.getPosition().x < ballRadius || ball.getPosition().x > windowWidth - ballRadius;**

**}**

**bool topCollision(const sf::CircleShape& ball) {**

**return ball.getPosition().y < ballRadius;**

**}**

**OUTPUT:**

