ChessAIThon: Integrating Chess, Coding & AI

A comprehensive educational initiative combining chess-based learning with coding principles and AI concepts to develop critical thinking, problem-solving, and creativity in VET curricula.

Project Objectives

Curriculum Integration

Provide tools to integrate chess-based learning for developing critical thinking, problem-solving, and creativity, including teaching coding principles, AI concepts, and version control through chess scenarios.

Interactive Learning Platform

Develop an online platform with a chessboard interface using Chess.js library for displaying legal moves, validating user moves, and storing valid moves in a database.

Data Management & AI Training

Implement robust data handling capabilities and provide methodologies for students to train AI models, culminating in a chess and AI competition.

Implementation Plan

1

Phase 1: Methodology Development (Months 1–6)

Develop curriculum for coding through chess, transversal skills, data structures, machine learning concepts, and version control. Create lesson plans and prepare teacher training.

2

Phase 2: Platform Development (Months 4–12)

Create database, develop chessboard interface, implement move validation, and integrate AI training modules and version control.

3

Phase 3: Student Engagement (Months 9–18)

Guide students in using the platform, training AI tools, and preparing for competition.

4

Phase 4: Competition & Dissemination (Months 15–24)

Host transnational chess and AI competition, document results, and share resources.

Curriculum Development (Phase 1)

Chapter 1: Coding Fundamentals through Chess

Teaching logic, functions, and implementation using chess problem-solving strategies

Chapter 2: Transversal Skills Development

Fostering cognitive skills, creativity, lateral thinking, problem-solving, and planning

Chapter 3: Chess Data Structures

Explaining representation in PGN, JSON, CSV, FEN, UCI, and SAN formats

Chapter 4: Machine Learning for Chess

Covering theoretical concepts and procedures relevant to chess-based AI

Chapter 5: Version Control

Outlining use of Git and GitHub for storing and sharing chess datasets

Lesson Plan Creation

Developing step-by-step guidance for chess problem-based scenarios

Platform Development (Phase 2)

Database Design

Store chess scenarios (FEN format) and moves (SAN, UCI, or resulting FEN)

Frontend Development

Create intuitive chessboard interface using Chess.js library

Move Validation

Integrate logic to check move legality and store valid moves

AI Integration

Prepare for AI training modules and version control

AI Architecture

The project's AI approach is inspired by AlphaZero but adapted for educational contexts with limited hardware resources.

ChessMarro vs AlphaZero: Key Differences

- AlphaZero learns by playing against itself repeatedly
- ChessMarro trains using human (student) games as examples
- Both use Convolutional Neural Networks (CNNs) for move prediction
- ChessMarro optimized for educational hardware constraints

While traditional algorithms like Stockfish are powerful through brute-force calculation, AI offers different advantages in learning, adaptation, and discovery in complex environments.

AI Implementation: Datasets

The team sourced extensive datasets from Lichess via Kaggle, including 30,000 pre-mate games and many complete games that required transformation into the project's format.

Data Sources

Kaggle datasets from Lichess: chess-games-in-fen-and-best-move

Storage Format

JSON (compressing each of 77 board positions into int64) or Parquet (better native compression)

Conversion Process

Transformation algorithms: convert-to-chessintion-format

AI Implementation: Training & Deployment

Training

- CNN model trained on Kaggle using Parquet files with generic games
- Mate-specific version achieved 90% precision
- Fine-tuned CNN balances precision and performance

Training code: cnn-pytorch-chess-generic

Deployment

- Model deployed on Kaggle with MCTS
- Interactive demo available
- Model shared via Hugging Face
- CPU deployment for accessibility

Demo: play-with-chessmarro

Development Workflow

Model Improvement

Use Kaggle or Colab to improve the CNN model

Version Control

Upload new versions to GitHub and Hugging Face

Deployment

Deploy on local GPU for competition and web app via CI/CD

Testing

Test model via Colab, Kaggle, and Hugging Face deployment

The centralized infrastructure includes GitHub for code and documentation, Kaggle for data transformation and training, and Hugging Face for model sharing and deployment.

Next Steps

Centralize Resources

Consolidate AI model, libraries, documentation, and source code in official GitHub repository

Finalize Training Pipeline

Complete dataset transformation tools and CNN training workflows

Prepare for Competition

Set up local GPU deployment for the transnational chess and AI competition