Table of Contents

J-AIL

Just-in-time Artificial Intelligence Library

Executive Summary 2

Introduction 3

Features 5

Placeholder 6

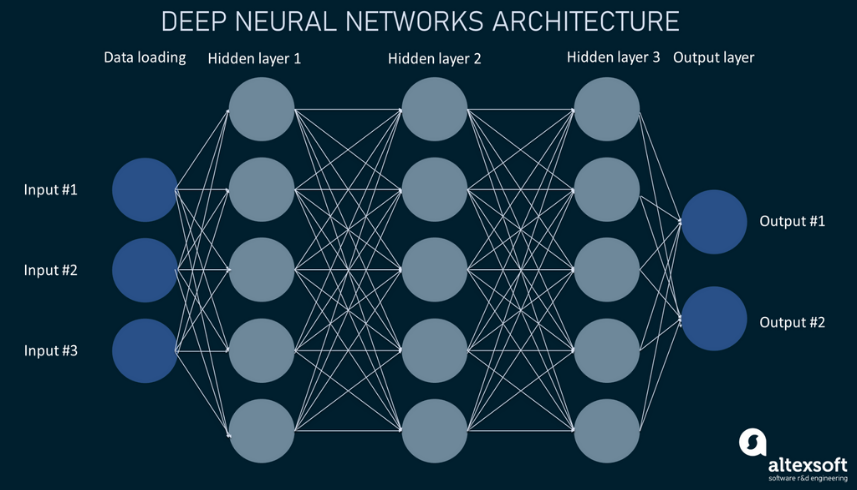
Placeholder 7

Placeholder 8

Placeholder 9

J-AIL

Executive Summary

Python is the leading programming language for AI development. However, its performance limitations hinder large-scale computations. Top end AI models make billions of computations per operation, and Python’s performance becomes a bottleneck. J-AIL aims to solve this bottle neck by leveraging the computational speed of C++.

In addition to performance, J-AIL aims to solve key issues found in existing AI development libraries. Many of them are bloated, complex and too generalized. By building a custom library, we can create a high-performance and lightweight toolkit tailored to our specific needs. With this tool our development team will be able to push the boundaries of AI and contribute to the future of technology.

Introduction

In the current age of AI, it is imperative to have a way for developers to make quick and efficient AI models, J-AIL will be the means in which they can do that. J-AIL will be an artificial intelligence framework that focuses on speed and simplicity. Because it is a programming framework that means it will only be used by the developers of the team. But in the end, developers using J-AIL will be capable of making high end AI models.

J-AIL will be similar to existing AI development libraries such as JAX and TensorFlow, the main difference will be how lightweight it is. Developing our own framework instead of using existing frameworks can be beneficial for a few reasons. First the developers won’t have to read through confusing and convoluted documentation. For example, TensorFlow, though powerful, has many pages of documentation that developers would have to sift through in order to find meaningful information (TensorFlow, 2024). Second, the lightweight framework will help in keeping code simple and maintainable by only providing necessary functionality. Both of these will assist in rapid development of AI models.

One of the main reasons for developing J-AIL is that it’s written in C++ as opposed to Python. “C++ can be up to 100 times faster than Python” (capaciteam, n.d.). This is important because of the billions of computations that AI models need in order to effectively train to be accurate. Although other frameworks like JAX are also written in C++, their Python-based application programing interface introduces an unnecessary layer of abstraction that complicates the development process (JAX, 2024).

In conclusion, J-AIL intends to fill a gap in AI development frameworks by offering a lightweight, high-performance framework. Its lightweight nature will reduce code complexity and give developers greater control over their AI models, leading to faster development cycles, improved performance, and a more efficient AI development process overall.

Features

Custom array (matrix) data structure

* **Transpose** – Swaps the columns and rows of a matrix. This is used in some neural network functions
* **Reshape** – This is used to change the shape of a matrix while keeping the data the same. This is used in some neural network functions.
* **Initialize with random values** – This is used when training AI models. It is essential for an AI development library.

Mathematics functionality

* **Matrix multiplication and addition** – This is a fundamental part of AI development. All of the math behind neural networks and transformers is matrix multiplication and addition.
* **Vector dot product** – An optimization for matrix multiplication. Vector dot products are also essential for attention blocks which are the heart of transformers.
* **Affine transformation** – Used primarily when adding the bias to a matrix.
* **Auto Derivative** – Auto derivative is not necessarily a mathematical concept, but there is a significant amount of calculus concepts behind it. Auto derivative is the application of those calculus concepts put into code.
* **Normalization functions** – The normalization functions are essential for making data more accessible to the machine learning model. These functions stop data from becoming significantly large negative numbers which don’t work well with machine learning models.
  + **ReLu**
  + **GELu**
  + **Sigmoid**
  + **Tanh**

Neural Network functionality

* **Back propagation -** Back propagation is used to adjust the values of the neural network during training. This is the backbone of neural networks, without this, neural networks would not be possible.
* **Forward propagation -** Data is run through the neural network and then output by the network. The output is either used in the loss function or it is the prediction of the network depending on if the network is training or predicting.
* **Loss -** Loss functions calculate the accuracy of the model and are used in back propagation for training.
  + **MSE / Mean Square Error**
  + **Cross Entropy**

Transformer Functionality

* **Multi-Layer Perceptron** – This piece of the transformer is essential for correlating non-linearly separable data points. It is essentially how large language models store facts.
  + **Up projection**
  + **Down projection**
* **Multi-head attention** – This is used because different words have different meanings based on the surrounding context. Each head of attention learns a different set of values so each head can associate different meanings to different words.
* **Token prediction** – This is where the AI “talks”. It is essentially the AI picking a word out of a given probability distribution and outputting it.

Data interaction functionality

* **Tokenization** – Tokenization is the process of taking a string of words and those words into smaller words so the AI can process them better.
  + **Training** – Creates the list of tokens.
  + **Tokenizing** – Splits a string or words based on the tokens that were trained.
  + **Pre-tokenizing** – Formats a string of words into a format that can be used to train a set of tokens or tokenize the string using an existing set of tokens.
* **Export Model weights** – This is essential so the AI model doesn’t have to be retrained every time it is restarted.
* **Import Model weights** – The counter side to exporting. This allows the model to import its weights from a file so it doesn’t have to retrain every time it restarts.

***References***

JAX. (2024). *Foreign Function Interface (FFI).* JAX. <https://docs.jax.dev/en/latest/ffi.html>

TensorFlow. (2024). *All Symbols in TensorFlow 2.* TensorFlow. <https://www.tensorflow.org/api_docs/python/tf/all_symbols>

Capaciteam. (n.d.) *C++ vs. Python: All You Need to Know.* Capaciteam. <https://capaciteam.com/c-plus-plus-vs-python/>