JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY
NOIDA

Compiler Design Lab

PROJECT REPORT

(Neural network compiler)

SUBMITTED TO:
Dr. Dharmveer Singh Rajpoot

SUBMITTED BY:
Vaibhav Sharma (15103311)
Jitesh Pabla (15103297)
Sajal Subodh (15103332)

INTRODUCTION

Title Of the Project : Neural network compiler

Objective :
The aim of this project is to create a small lightweight and easily written language for basic and powerful neural networks. This will help for fast prototyping and for the educational purposes of beginners as well as intermediate learners. This is to developed for use by the hobbyist, educational sector or any non-professional developer. To ensure that it meets the needs of these personal developers, the compiler will be easy to comprehend, extend, modify and with a concise set of tokens.

Scope
The scope of the development is a lightweight easily adaptable and easily maintainable neural network compiler:

- It will be designed for the linux OS, which can be used in any OS which will run python.
- It will be freely available as an Open Source project for anyone to use and extend as they wish.
- It will be written in the python programming language because of its low memory footprint and simplicity, and high usage in the machine learning community.
- As well as providing the core neural network functionality, the structure of the compiler will be open ended to allow easy extension and customisation by the intended user.

Background Research Topics

Artificial neural networks (ANNs)
Artificial neural networks (ANNs) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any a priori knowledge about cats, e.g., that they have fur, tails, whiskers and cat-like faces. Instead, they evolve their own set of relevant characteristics from the learning material that they process.

An ANN is based on a collection of connected units or nodes called artificial neurons (a simplified version of biological neurons in an animal brain). Each connection (a simplified version of a synapse) between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

Python Lex and Yacc (PLY)
PLY is a 100% Python implementation of the common parsing tools lex and yacc. It provides very extensive error reporting and diagnostic information to assist in parser construction. The original implementation was developed for instructional purposes. As a result, the system tries to identify the most common types of errors made by novice users. It uses Python introspection features to build lexers and parsers.This greatly simplifies the task of parser construction since it reduces the number of files and eliminates the need to run a separate lex/yacc tool before running your program.

PLY can be used to build parsers for "real" programming languages. Although it is not ultra-fast due to its Python implementation, PLY can be used to parse grammars consisting of several hundred rules (as might be found for a language like C). The lexer and LR parser are also reasonably efficient when parsing typically sized programs. People have used PLY to build parsers for C, C++, ADA, and other real programming languages.

Keras
Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

PROJECT DESCRIPTION

Overview
The neural network compiler uses the following set of tokens:

'int',
'float',
'load',
'split',
'filepath',
'input',
'output',
'train',
'use',
'sequential',
'test',
'add_layer',
'compile',
'fit',
'evaluate',
'relu',
'sigmoid',
'adam',
'tanh',
'linear',
'adagrad',
'rmsprop',
'sgd'

Grammar -

start : LoadFile Split Use Addl Addl2 Compile Fit Evaluate

LoadFile : load filepath

Split : split splitinput splitoutput

splitinput : input int int

splitoutput : output int

Use : use TON

TON : sequential

Addl : add_layer int int Act

Addl2 : add_layer int Act Addl2

Addl2 :

Act : relu

| sigmoid

| tanh

| linear

Compile : compile Optimizer

Optimizer : adam

| adagrad

| sgd

| rmsprop

Fit : fit epoch BatchSize

epoch : int

BatchSize : int

Evaluate : evaluate

A user will use these set of tokens to define a load a dataset, define a neural network, and then run the dataset on the defined neural network to compute the accuracy it achieves.

A sample program is given below:

```
load diabetes.csv
split input 0 8 output 8
use sequential
add_layer 8 12 relu
add_layer 1 sigmoid
add_layer 8 relu
compile adam
fit 50 10
evaluate
end
```

1- Loades a dataset using pandas into the compiler.

2- splits the dataset into input and output values.

3- defines which neural network to use for this given dataset.

4, 5 and 6- Provides further tweaking of the chosen neural network for the user. Adds a layer into the neural network, for line 4 the two

integers denote the number of input dimensions and number of neurons, while in 5 and 6 the integer denotes the number of neurons. The last word denotes the activation function for each layer.

7- defines the optimiser for training the neural network.

8- defines the number of epochs and batch size.

9- run the training of the neural network and displays accuracy.

10- defines the end of the program.

CONCLUSION

The initial aim of this project was to develop a small easily extensible, easy and stand alone neural network compiler implementation that is suited to the hobbyist, educational sector or any non-professional developer.
The program itself has been designed and implemented; the resulting implementation was then evaluated according to distinct criteria set out in and this entire process is documented in this project report.

Future Enhancements:

- Add more neural network models for eg. simple perceptron, recurrent neural networks, Adaline etc.
- Remove dependency on keras, for added educational benefits.
- Write proper documentation for the compiler.
- Make a command line tool to run the program directly (just like any programming language like Python, Lua etc)

References:

- https://github.com/dabeaz/ply
- https://keras.io/
- https://docs.scipy.org/doc/
- youtube.com
- wikipedia.org1