INSTITUTO TECNOLÓGICO DE IZTAPALAPA

ING. SISTEMAS COMPUTACIONALES

ONNX-MLIR (OPEN NEURAL NETWORK EXCHANGE - MULTI-LEVEL INTERMEDIATE REPRESENTATION)

PRESENTAN:

CORTES ÁNGELES RICARDO DANIEL 171080104 %25
DELGADO RODRIGUEZ DANIEL RAUL 191080091 %25
GOMEZ RODRIGUEZ JUAN MANUEL 161080275 %25
PAREDES PEGUEROS JUAN ANTONIO 171080125 %25

EQUIPO: X-FORS

ASESOR:

PARRA HERNANDEZ ABIEL TOMAS

CIUDAD DE MÉXICO **JUNIO / 2021**

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx **www.tecnm.mx | www.iztapalapa.tecnm.mx**

INDICE

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Introducción

Que es MLIR es una descripción de la infraestructura del software de compilador que permite reducir su objetivo de construir sus compiladores para una conexión. Sirve para diseñado para ser un IR híbrido que puede soportar múltiples requisitos diferentes en una infraestructura unificada. Por ejemplo, esto incluye:

- Capacidad para albergar optimizaciones de bucle de estilo informático de alto rendimiento en los núcleos (fusión, intercambio de bucle, ordenamiento en teselas, etc.) y para transformar diseños de memoria de datos.
- Capacidad para representar operaciones específicas de un objetivo, por ejemplo, operaciones de alto nivel específicas de un acelerador.
- Cuantización y otras transformaciones de gráficos realizadas en un gráfico de aprendizaje profundo.

ONNX define un conjunto común de operadores (los componentes básicos de los modelos de aprendizaje automático y aprendizaje profundo) y un formato de archivo común para permitir que los desarrolladores de IA usen modelos con una variedad de marcos, herramientas, tiempos de ejecución y compiladores.

Los operadores se implementan de forma externa al gráfico, pero el conjunto de operadores integrados es portátil a través de marcos. Cada marco que soporte ONNX proporcionará cumplimentaciones de estos operadores en los tipos de datos aplicables.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Resumen

Comprender la clave de su funcionamiento tanto la área de hardware y software de las compañías de electrónica, informática, entre otras, ya que las comunicaciones entre los datos existente en las operaciones integradas son la base de una red de comunicación que da la traficación de un ejemplo de un código abierto que implementa la inteligencia artificial (AI) que es dedica a aprender y dar solución a los problemas por medio de un **intercambio de redes neuronales abierto** o ONNX sus siglas

Palabras clave

Calcular, redes, ecosistema, compartir, ideas, modelos

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx **www.tecnm.mx | www.iztapalapa.tecnm.mx**

Objetivos

OBJETIVO GENERAL

Es identificar el concepto de la **ONNX-MLIR** y aprender su desarrollo, aplicaciones y donde se ve, ya que su nombre lo dice que es un intercambio de redes neuronales abiertas, que quiere decir que son un conjunto de códigos que aprende de uno de otro para adaptarse y aprender nuevas funciones ya que esto se aplica en el área de la IA

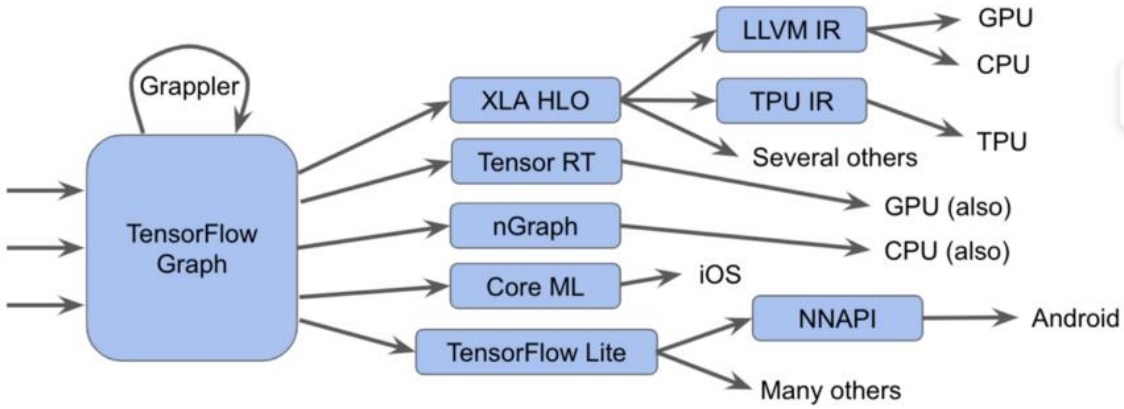
OBJETIVO ESPECIFICOS

1. **Entender su flujo grafico de datos en el ecosistema operativo**
2. **Hacer posible que los desarrolladores utilicen las combinaciones adecuadas de herramientas para su proyecto.**
3. **Aprovechar los servicios de los proveedores de la nube para crear, entrenar e inferencia sus modelos.**
4. **Dar a explicar los diagramas o códigos completos de su funciones**

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx **www.tecnm.mx | www.iztapalapa.tecnm.mx**

Justificación

En la investigación sobre el interpretación de redes neuronales abiertas o ONNX es de los datos a transmitir para una conexión trabajos unidos que han de alcanzar su objetivo y generar una estructura o red de trabajos que permiten dar el resultado, esto se ve en distintos niveles MLIR que aplica con funciones matemáticas, algoritmos, probabilidades, estadísticas y IA.



Esto nos permitira dar solucion a problemas que se generan he incluso aprender de esos error para una mayor desempeño en las actividades y se ve en la area de diseño graficos, comunicacion . Esto significa que consiste en una especificación de representación intermedia (IR) y un conjunto de herramientas para transformar esta representación. Cuando hablamos de compiladores, pasar de una vista de nivel superior a una vista de nivel inferior se denomina reducción, y utilizaremos este término en el futuro. El modelo espera que los datos de entrada y salida estén en un formato específico. ML.NET permite definir el formato de los datos mediante clases. En ocasiones, es posible que ya sepa cuál es el formato. De lo contrario, puede usar herramientas como Netron para inspeccionar el modelo de ONNX.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Marco teórico

MARCO TEÓRICO La base para la construccion de los compiladores se fundamenta en la teoría de las maquinas de estado finito y el estudio de las gramáticas libres de contexto principalmente, es por ellos que está sección presenta nociones sobre estos temas para tener una mejor aproximación a la puesta en práctica en la construcción del analizador léxico y la creación del lenguaje para el analizador sintáctico. AUTÓMATA FINITO O MAQUINAS DE ESTADO FINITO Un autómata finito (AF) o máquina de estado finito es un modelo matemático que realiza cómputos en forma automática sobre una entrada para producir una salida. Este modelo está conformado por un alfabeto, un conjunto de estados y un conjunto de transiciones entre dichos estados. Su funcionamiento se basa en una función de transición, que recibe en un estado inicial una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un estado final o de aceptación, que representa la salida. La finalidad de los autómatas finitos es la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

GRAMÁTICA LIBRES DE CONTEXTO En lingüística e informática, una gramática libre de contexto (o de contexto libre) es una gramática formal en la que cada regla de producción es de la forma: V→w Donde V es un símbolo no terminal y w es una cadena de terminales y/o no terminales. El término libre de contexto se refiere al hecho de que el no terminal V puede siempre ser sustituido por w sin tener en cuenta el contexto en el que ocurra. Por otro lado, estas gramáticas son suficientemente simples como para permitir el diseño de eficientes algoritmos de análisis sintáctico que, para una cadena de caracteres dada determinen como puede ser generada desde la gramática.

Los analizadores LL y LR tratan restringidos subconjuntos de gramáticas libres de contexto. La notación más frecuentemente utilizada para expresar gramáticas libres de contexto es la forma Backus-Naur.

Ejemplo 1 Una simple gramática libre de contexto es S → aSb | ε donde | es un o lógico y es usado para separar múltiples opciones para el mismo no terminal, ε indica una cadena vacía. Esta gramática genera el lenguaje no regular .

Ejemplo 2 Aquí hay una gramática libre de contexto para expresiones enteras algebraicas sintácticamente correctas sobre las variables x, y y z: S → x | y | z | S + S | S – S | S *S | S/S | (S) Generaría, por ejemplo, la cadena (x + y) *x – z *y / (x + x)

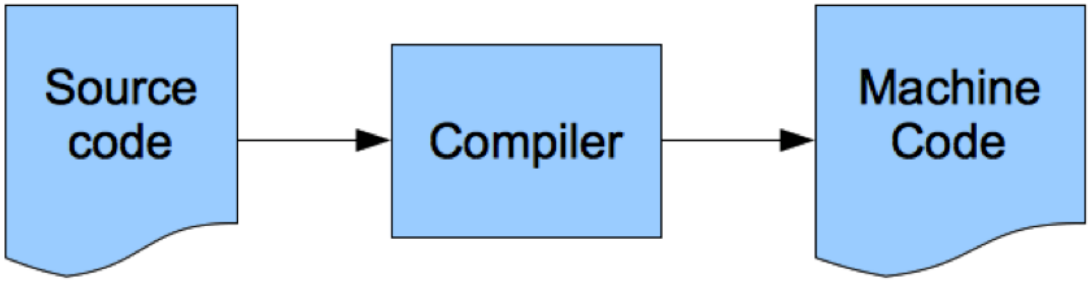
Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

OBJETIVOS : Implementar las etapas de análisis léxico y sintáctico de un compilador aplicando la teoría de autómatas finitos y lenguajes formales para crear la gramática del lenguaje.

a) **COPILADOR**

Es un Software que traduce un programa escrito en un lenguaje de programación de alto nivel (C / C ++, COBOL, etc.) en lenguaje de máquina. Un compilador generalmente genera lenguaje ensamblador primero y luego traduce el lenguaje ensamblador al lenguaje máquina. Una utilidad conocida como «enlazador» combina todos los módulos de lenguaje de máquina necesarios en un programa ejecutable que se puede ejecutar en la computadora.

El compilador de tiempo de ejecución puede optimizar ramales estáticos no utilizados de un programa durante la ejecución, y por tanto es útil para la evaluación parcial en los casos en los que un programa tiene muchas opciones, muchas de las cuales pueden ser determinadas como innecesarias en un entorno específico. Esta característica es usada en la pipeline de OpenGL en el Mac OS X Leopard (v10.5) para proporcionar soporte a características ausentes del hardware. El código de gráficos dentro de la pila de OpenGL se dejó en forma intermedia, y después compilado cuando se ejecuta en la máquina de destino. En sistemas con GPUs de gama alta, el código resultante era bastante escaso y redirigía las instrucciones a la GPU con cambios mínimos. En sistemas con GPUs de gama baja, LLVM compilaba procedimientos opcionales para que se ejecutaran en la unidad central de procesamiento (CPU) local que emularan las instrucciones que la GPU no podía ejecutar internamente. LLVM mejoró el rendimiento de las máquinas de gama baja que usaban conjuntos de chips Intel GMA. Un sistema similar fue desarrollado bajo la Gallium3D LLVMpipe e incorporado en la shell de GNOME para permitir que se ejecute sin una GPU.



Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

b) **LLVM**

El proyecto LLVM comenzó en 2000 en la Universidad de Illinois en Urbana-Champaign, bajo la dirección de Vikram Adve y Chris Lattner. LLVM fue desarrollado inicialmente bajo la Licencia de código abierto de la Universidad de Illinois, una licencia de tipo BSD. En 2005, Apple Inc. Contrató a Lattner y formó un equipo para trabajar en el sistema de LLVM para varios usos dentro de los sistemas de desarrollo de Apple.LLVM es parte integrante de las últimas herramientas de desarrollo de Apple para Mac OS X e iOS.

El nombre “LLVM” era en principio las iniciales de “Low Level Virtual Machine”, pero esta denominación causó una confusión ampliamente difundida, puesto que las máquinas virtuales son solo una de las muchas cosas que se pueden construir con LLVM. Cuando la extensión del proyecto se amplió incluso más, LLVM se convirtió en un proyecto paraguas que incluye una multiplicidad de otros compiladores y tecnologías de bajo nivel, haciendo el nombre aún menos adecuado. Por tanto, el proyecto abandon las iniciales. Actualmente, LLVM es una “marca” que se aplica al proyecto paraguas, la representación intermedia LLVM, el depurador LLVM, la biblioteca estándar de C++ definida por LLVM, etc. LLVM también puede generar código máquina relocalizable en el momento de compilación o de enlazado, o incluso código máquina binario en el momento de ejecución.

LLVM permite un conjunto de instrucciones y sistema de tipos independientes del lenguaje. Cada instrucción está en una forma estática de asignación única (SSA, en inglés, static single assignment), es decir, que cada variable (llamado un registro tipado) es asignada una sola vez y congelado. LLVM permite que el código sea compilado estáticamente, al igual que lo es bajo el sistema GCC tradicional, o por el contrario que se deje para una compilación tardía desde la IF a código máquina en una compilación en tiempo de ejecución de manera similar a como lo hace Java.

El sistema de tipos consiste de tipos básicos como entero, números de coma flotante y cinco tipos de datos compuestos (en inglés): punteros, vectores, matrices, tuplas y funciones. Un constructo tipado en un lenguaje concreto puede ser representado combinando estos tipos básicos en LLVM. Por ejemplo, una clase en C++ puede ser representada por una combinación de estructuras, funciones y matrices de punteros a funciones.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

El amplio interés que ha recibido LLVM ha llevado a una serie de tentativas para desarrollar frontales totalmente nuevos para una variedad de lenguajes. El que ha recibido la mayor atención es Clang, un nuevo compilador que soporta C, Objective-C y C++. Apoyado principalmente por Apple, Clang aspira a reemplazar al compilador de C y Objective-C en el sistema GCC con un sistema más moderno que sea más fácil de integrar con entornos de desarrollo integrado (IDEs), y que tenga un soporte más amplio para multihilo. El desarrollo de Objective-C bajo GCC estaba estancado y los cambios de Apple en el lenguaje eran soportados en una rama mantenida por separado. Crear su propio compilador les permitió abordar muchos de los mismos problemas que LLVM abordó para la integración con IDEs y otras características modernas, a la vez que hacer la rama principal de desarrollo la rama de implementación de Objective-C.

c) **MLIR**

MLIR está diseñado para ser un IR híbrido que puede soportar múltiples requisitos diferentes en una infraestructura unificada. La capacidad de representar gráficos de flujo de datos (como en TensorFlow), incluidas las formas dinámicas, el ecosistema operativo extensible por el usuario, las variables de TensorFlow, etc. Las optimizaciones y transformaciones se realizan normalmente en tales gráficos (por ejemplo, en Grappler). Capacidad para albergar optimizaciones de bucle de estilo informático de alto rendimiento en los núcleos (fusión, intercambio de bucle, ordenamiento en teselas, etc.) y para transformar diseños de datos en la memoria. Transformaciones de “reducción” de generación de código como inserción DMA, gestión explícita de caché, ordenamiento en teselas de memoria y vectorización para arquitecturas de registro 1D y 2D.

Capacidad para representar operaciones específicas de un objetivo, por ejemplo, operaciones de alto nivel específicas de un acelerador. Cuantización y otras transformaciones de gráficos realizadas en un gráfico de aprendizaje profundo.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Metodología de trabajo

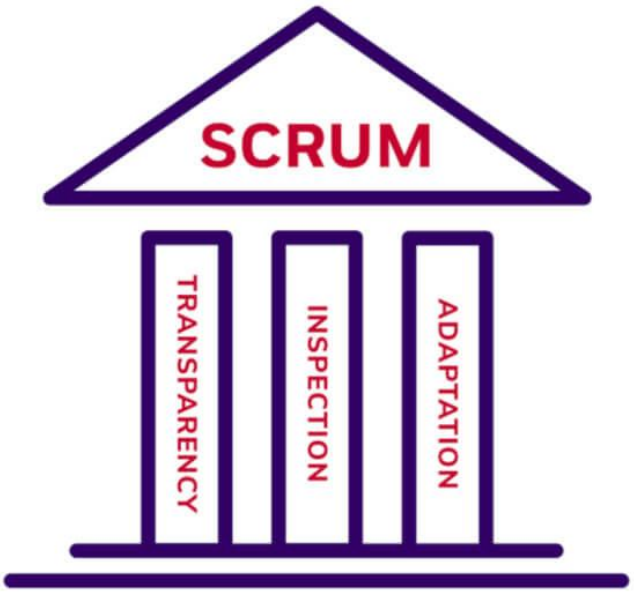
Como equipo estuvimos investigando que metodología nos conviene utilizar, tuvimos varias opciones, pero decimos utilizar la metodología ágil (SCRUM) ya que se adapta a nuestras necesidades de investigación.

SCRUM se trata de un proceso en el que se aplican un conjunto de buenas prácticas de manera regular. Se trabaja colaborativamente, en equipo y se van realizando entregas parciales y regulares del producto final.

La metodología Scrum al estar enmarcada dentro de las metodologías agile, Scrum se basa en aspectos como:

- La flexibilidad en la adopción de cambios y nuevos requisitos durante un proyecto complejo.
- La colaboración e interacción con el cliente.
- El desarrollo iterativo como forma de asegurar buenos resultados.

Los pilares o características de la metodología Scrum más importantes son:



Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

- Transparencia

Con el método Scrum todos los implicados tienen conocimiento de qué ocurre en el proyecto y cómo ocurre. Esto hace que haya un entendimiento “común” del proyecto, una visión global.

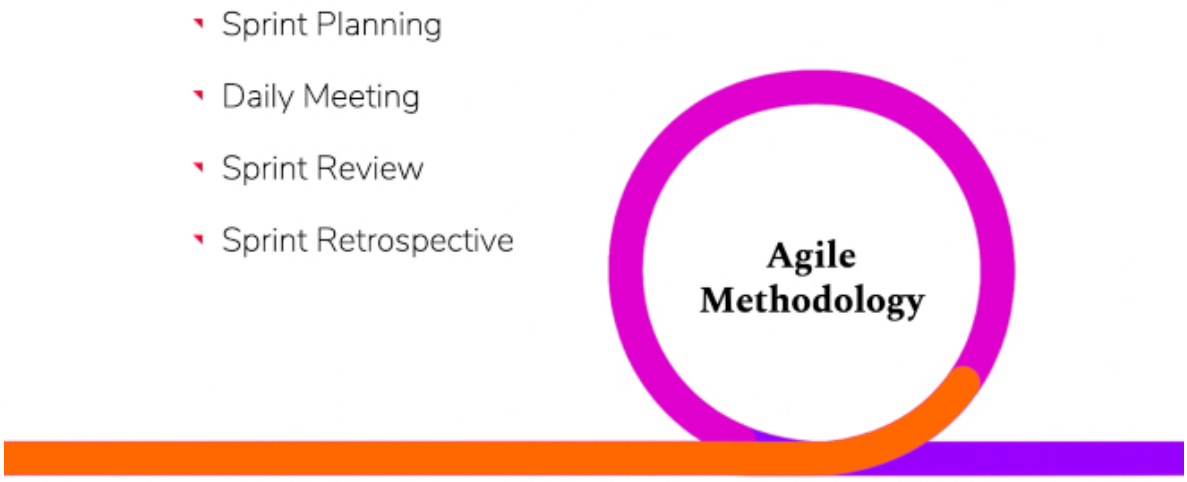
- Inspección

Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera autoorganizada.

- Adaptación

Cuando hay algo que cambiar, el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir el éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

Los hitos de la Metodología de trabajo Scrum



Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

- Sprint

El sprint es el corazón de Scrum, es el contenedor de los demás hitos del proceso.

- Sprint planning

En esta reunión todo el equipo Scrum define qué tareas se van a abordar y cuál será el objetivo del sprint.

- Daily meeting

Es una reunión diaria dentro del sprint que tiene como máximo 15 minutos de duración.

- Sprint review

La review del valor que vamos a entregar al cliente se hace en esta reunión, al final de cada sprint.

- Sprint retrospective

La retrospectiva es el último evento de Scrum, tiene una duración de 3 horas para Sprints de un mes, y es la reunión del equipo en la que se hace una evaluación de cómo se ha implementado la metodología Scrum en el último sprint.

METODOLOGÍA Analizador Léxico: Para la construcción del analizador léxico del lenguaje, se ha definido los tokens y palabras reservadas para construir el autómata finito determinista que nos devolverá los tokens respectivos cuando encuentre un estado de aceptación. Palabras reservadas int float void bool double char for

If elseif else while do switch case

Default min max factorial raiz potencia escribir

Return break continue seno coseno

] ‘|&&>=^¡=

Símbolos especiales ( ) ; * % ^ , “ { } : [= ++ -= ¿ == || <

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Lista de tokens Tipos de datos : INT , FLOAT , VOID ,BOOL DOUBLE , CHAR Letras y dígitos : ID , NUMERO , DECIMAL CIENTIFICA Comparación : MENOR , MENORIGUAL , MAYOR MAYORIGUAL , IGUAL , DIFERENTE Conectores lógicos: OR , AND , ORBIT , ANDBIT Unarios : INCREMENTO , DECREMENTO

Sentencias especiales : FOR , IF , ELSEIF , ELSE , DO , WHILE SWITCH , CASE , DEFAULT Funciones especiales : MIN , MAX , FACTORIAL , RAIZ , POTENCIA , ESCRIBIR , SENO, COSENO Asignación e impresión : ASIGNACION , IMP Terminación de flujo : RETURN , BREAK , CONTINUE

Analizador Sintáctico : Para la construcción del analizador sintáctico se ha creado la gramática del lenguaje definida en notación BNF y se está usando el programa yacc para generar la implementación del analizador sintáctico. ::= ::= | ; ::= | | | | ;| ; | ; | ; | ; | ;

ID ( ){ } switch ( ) { } | case : case ‘ ID ‘ : case default : for(;;) { } for(;;) while () while ( ) { } do { } while ( ); return | break | continue return | return if ( ) if ( ) { } if ( ) { } else { } if ( ) { } else { } elseif | elseif elseif ( ) { } ID = ID = || | && | | | & < | > | = | = | ¡= ID++ | ID--; + - * % ^ / ( ) min(,)

Max(,) factorial( ) raiz( ) potencia(,) escribir(“ “) seno() coseno() ¿ ID ID | NUMERO | DECIMAL | CIENTIFICA ID

Por contraste, en los casos en que el rendimiento puro es medido como referencia, LLVM 2.9 va por detrás de GCC 4.6.1 en calidad del código, entendida como velocidad de los programas compilados, en torno al 10% de media, a la vez que compila de 20 a 30% más rápido

Existen muchos otros componentes en varios estados de desarrollo; incluyendo, sin ser exhaustivos, un frontal para bytecode de Java, un frontal para CPython la implementación de Ruby 1,9 para Mac (MacRuby), varios frontales para Standard ML, y un nuevo asignador de registros de coloración de grafos. Descripción general de la representación intermedia de varios niveles El proyecto MLIR es un enfoque novedoso para construir una infraestructura de compilador extensible y reutilizable. MLIR tiene como objetivo abordar la fragmentación del software, mejorar la compilación para hardware heterogéneo, reducir significativamente el costo de construir compiladores específicos de dominio y ayudar a conectar compiladores existentes.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Desarrollo e Implementación

El formato ONNX fue anunciado por primera vez el mes pasado por microsoft y facebook, con el fin de brindar a los usuarios más opciones en cuanto a entornos de inteligencia artificial, ya que cada proyecto de modelado tiene requerimientos especiales y se necesitan diferentes herramientas para cada etapa. Intel y otras empresas participan en el proyecto para ofrecer mayor flexibilidad a la comunidad de desarrolladores, dándoles acceso a las herramientas más adecuadas para cada proyecto único de inteligencia artificial y la habilidad de hacer fácilmente cambios entre ambientes o herramientas.

La ONNX puede ayudar a optimizar la inferencia de su modelo de aprendizaje automático. La inferencia, o puntuación del modelo, es la fase en la que el modelo implementado se usa para la predicción, más comúnmente en los datos de producción.

La optimización de los modelos de aprendizaje automático para la inferencia (o la puntuación del modelo) es difícil, ya que necesita ajustar el modelo y la biblioteca de inferencia para aprovechar al máximo las capacidades del hardware. El problema se vuelve extremadamente difícil si desea obtener un rendimiento óptimo en diferentes tipos de plataformas (nube, borde, CPU, GPU, etc) ONNX Runtime se utiliza en servicios de Microsoft de gran escala, como Bing, Office y Azure Cognitive Services

Los paquetes de Python para ONNX Runtime están disponibles en PyPi.org (CPU , GPU). Lea los requisitos del sistema antes de la instalación.
Para instalar ONNX Runtime para Python, use uno de los siguientes comandos:

```
pip install onnxruntime          # CPU build

pip install onnxruntime-gpu   # GPU build
```

Para llamar a ONNX Runtime en su secuencia de comandos de Python, use:

```
import onnxruntime

session = onnxruntime.InferenceSession("path to model")
```

La documentación que acompaña al modelo generalmente le dice las entradas y salidas para usar el modelo. También puede utilizar una herramienta de visualización como Netron para ver el modelo. ONNX Runtime también le permite consultar los metadatos, entradas y salidas del modelo:

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

session.get_modelmeta()

first_input_name = session.get_inputs()[0].name

first_output_name = session.get_outputs()[0].name

Para hacer una inferencia de su modelo, use runy pase la lista de salidas que desea que se devuelvan (déjelas en blanco si las desea todas) y un mapa de los valores de entrada. El resultado es una lista de las salidas.

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```


Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Resultados

Los modelos de redes neuronales profundas se están volviendo cada vez más populares y se han utilizado en diversas tareas como la visión por computadora, el reconocimiento de voz y el procesamiento del lenguaje natural. Los modelos de aprendizaje automático se entrenan comúnmente en un entorno rico en recursos y luego se implementan en un entorno distinto, como máquinas de alta disponibilidad o dispositivos periféricos. Para ayudar a la portabilidad de los modelos, la comunidad de código abierto ha propuesto el estándar Open Neural Network Exchange (ONNX).

Onnx-mlir se basa en el concepto de dialectos MLIR para implementar su funcionalidad. Proponemos aquí dos nuevos dialectos., Un dialecto específico de ONNX que codifica la semántica estándar de ONNX, un dialecto basado en bucle para proporcionar un punto de descenso común para todas las operaciones de dialecto de ONNX. Cada representación intermedia facilita su propio conjunto de características de optimizaciones a nivel de gráfico y basadas en bucles, respectivamente.

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail: division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx

Conclusiones

Mi conclusion sobre el tema onnx-mlir es que son herramientas de que trabajan en conjunto para realizar una accion predeterminada generando que tanto el software como el hardware sean partes una red neuronales que da a aprender los el mismo programando y genera lo que se conoce como inteligencia artificial. La IA tiene la Capacidad para albergar optimizaciones de bucle de estilo informático de alto rendimiento en todos los núcleos (fusión, intercambio de bucle, ordenamiento en teselas, etc.) y para transformar los diseños de memoria de los datos.

La conclusión de este proyecto por mi parte es que los modelos de redes neuronales profundas se están volviendo cada vez más populares y se han utilizado en diversas tareas como la visión por computadora. Para representar modelos de redes neuronales, las personas suelen usar onnx el cual es un formato estándar abierto para la interoperabilidad del aprendizaje automático. ONNX-MLIR es un compilador basado en MLIR para reescribir un modelo en ONNX en un binario independiente que se puede ejecutar en diferentes hardware de destino, como máquinas x86.

Fuentes de información

https://www.c-sharpcorner.com/blogs/onnx

https://www.tensorflow.org/mlir?hl=es-419

https://github.com/onnx/onnx-mlir

https://mlir.llvm.org/

Anexos

https://github.com/JuAntonioParedes/ProyecFinalAutomatas2.git

https://github.com/DANIELRAUL95/lenguajes-automas

https://github.com/Daniel2608hiop/AUTOMATAS-ONNX

https://github.com/juan561999/Onnx-mlir

Av. Telecomunicaciones S/N, Col. Chinampac de Juárez, C.P. 09208,
Alcaldía de Iztapalapa,Ciudad de México Tel. 5773-8210, e-mail:
division@iztapalapa.tecnm.mx www.tecnm.mx | www.iztapalapa.tecnm.mx