Informe sobre la Implementación de Redes Neuronales en GPU y CPU

Erick Jesús Ríos González

December 8, 2024

Abstract

Este informe describe la implementación de redes neuronales utilizando unidades de procesamiento gráfico (GPU) y unidades de procesamiento central (CPU). Se abordan las metodologías de implementación en ambos entornos, el rendimiento comparativo entre ellos, los desafíos enfrentados durante el proyecto y las soluciones aplicadas. Finalmente, se incluyen conclusiones sobre el trabajo realizado y posibles áreas de mejora.

1 Introducción

El objetivo de este proyecto es implementar un modelo de red neuronal artificial que utilice el algoritmo de retropropagación para el entrenamiento en dos entornos diferentes: CPU y GPU. A través de esta implementación, se busca comparar el rendimiento entre ambas plataformas, analizando aspectos como el tiempo de ejecución y la escalabilidad en diferentes tamaños de entrada y redes.

Las redes neuronales son una clase de modelos computacionales inspirados en el cerebro humano. Son ampliamente utilizadas en diversas aplicaciones de aprendizaje automático, desde el reconocimiento de patrones hasta la predicción en grandes volúmenes de datos. El uso de GPUs en este tipo de modelos se ha incrementado gracias a su capacidad de procesamiento paralelo, lo cual puede acelerar significativamente el tiempo de entrenamiento.

2 Metodología

2.1 Implementación en CPU

La implementación en CPU se realizó utilizando un enfoque secuencial clásico para la propagación hacia adelante (forward pass) y la retropropagación de gradientes (backpropagation). Se diseñó un modelo de red neuronal con una o dos capas ocultas, y se utilizó el algoritmo de optimización de gradiente descendente para ajustar los pesos de la red. La principal limitación de este enfoque es la velocidad de cálculo, que puede ser más lenta debido a la falta de paralelismo.

2.2 Implementación en GPU

La implementación en GPU utiliza la biblioteca CUDA para paralelizar las operaciones de cálculo intensivo, especialmente las matrices de multiplicación durante la propagación hacia adelante y el cálculo de gradientes durante la retropropagación. El paralelismo en la GPU permite realizar múltiples cálculos simultáneamente, lo que acelera significativamente el tiempo de entrenamiento.

A continuación, se presentan las funciones principales utilizadas en la implementación:

2.2.1 Propagación hacia Adelante (Forward Pass)

La propagación hacia adelante se realiza de manera paralela en la GPU, donde cada neurona en la capa de salida calcula su activación de acuerdo con las neuronas en la capa anterior. La fórmula general es:

$$a_i = f\left(\sum_{j=1}^{n} w_{ij} \cdot x_j + b_i\right)$$

donde a_i es la activación de la neurona i, w_{ij} son los pesos, x_j son las entradas y b_i es el sesgo. La función f es la función de activación, como la función sigmoide o ReLU.

2.2.2 Retropropagación (Backpropagation)

La retropropagación se realiza en dos fases: cálculo de los gradientes y actualización de los pesos. Los gradientes de error se calculan a partir de la derivada del error con respecto a las salidas de las neuronas y se propagan hacia atrás a través de la red. El ajuste de los pesos se realiza utilizando la fórmula:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

donde η es la tasa de aprendizaje y E es la función de error.

3 Resultados

A continuación, se presentan los resultados obtenidos al comparar el rendimiento de la implementación en CPU y GPU. Los experimentos se realizaron con conjuntos de datos de entrada de diferentes tamaños, y se midió el tiempo de entrenamiento para cada implementación.

Tamaño de Entrada	Tiempo CPU (s)	Tiempo GPU (s)
1000	5.23	0.35
2000	8.42	0.48
3000	12.13	0.62

Table 1: Comparación del tiempo de entrenamiento entre CPU y GPU.

Como se observa en la tabla anterior, el tiempo de entrenamiento en la GPU es significativamente menor en comparación con la CPU. A medida que el tamaño de la entrada aumenta, la diferencia en el rendimiento entre CPU y GPU se amplifica.

4 Análisis

Uno de los principales desafíos que se enfrentaron durante el proyecto fue la gestión de memoria en la GPU. Se debió optimizar el uso de la memoria global y la memoria compartida para asegurar que las operaciones paralelizadas fueran lo más eficientes posible. Además, se implementaron técnicas como

la reducción de precisión en los cálculos para mejorar el rendimiento sin sacrificar la exactitud de los resultados.

Un aspecto clave fue la adaptación del algoritmo de retropropagación a la GPU. El proceso de propagación hacia adelante y hacia atrás debe ser cuidadosamente paralelizado para maximizar el uso de la GPU. Además, la optimización de los hilos y bloques de CUDA resultó fundamental para obtener un rendimiento óptimo.

5 Conclusiones

El uso de la GPU para entrenar redes neuronales ofrece una mejora significativa en el tiempo de ejecución en comparación con la CPU, especialmente para redes grandes y conjuntos de datos extensos. Sin embargo, la implementación en GPU requiere una gestión cuidadosa de la memoria y la paralelización de las operaciones.

Futuras mejoras podrían incluir la implementación de técnicas avanzadas como la optimización automática de hiperparámetros utilizando algoritmos evolutivos o enfoques basados en redes neuronales cuánticas, que también podrían beneficiarse del paralelismo de las GPU.

6 Anexos

6.1 Código Fuente

A continuación se presenta una parte del código fuente utilizado en la implementación en GPU. El código completo se puede encontrar en el repositorio del proyecto.

```
// Función para la propagación hacia adelante en GPU
__global__ void forwardPassKernel(float* inputs, float* weights, float* outputs, i
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < outputSize) {
        float output = 0.0;
        for (int i = 0; i < inputSize; i++) {
            output += inputs[i] * weights[idx * inputSize + i];
        }
        outputs[idx] = sigmoid(output);
```

```
    }
}
```

6.2 Manual de Usuario

El manual de usuario está disponible en el archivo README.md del repositorio. Asegúrese de seguir las instrucciones para configurar su entorno y ejecutar el proyecto correctamente.