Documentation Du Projet

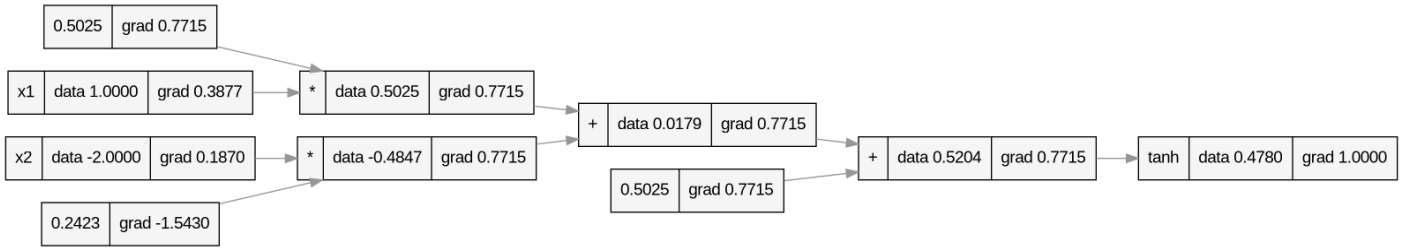
Ce projet est une réimplémentation en C++ moderne du célèbre moteur d'autodifférence *Micrograd*, inspiré par le travail d'Andreï Karpathy. Il construit un graphe de calcul dynamique pour le calcul des dérivées via la rétropropagation (backpropagation), posant les fondations nécessaires à la construction de réseaux de neurones de base.

Structure du Projet :

Fichier	Description
engine.h/engine.cpp	Définit la classe Value, le conteneur fondamental pour les données (data) et le gradient (grad), et implémente les fonctions de rétropropagation (_backward) et l'algorithme de parcours topologique (backward).
nn.h/nn.cpp	Implémentation des primitives de réseaux de neurones : Neuron, Layer, et MLP. Utilise la classe Value pour gérer les poids et les biais.
test.cpp	Tests Unitaires. Vérifie l'exactitude des gradients pour des expressions complexes via test_sanity_check et test_more_ops.
graph.h/graph.cpp	Pour la visualisation : Contient la fonction generate_dot qui crée une représentation Graphviz du graphe de calcul à partir d'un nœud racine.
testneuron.cpp	Test complet d'un seul Neuron pour démontrer le forward pass, le backward pass et la génération du fichier neuron_graph.dot.

Exemple de Graphe de Calcul (neuron_graph.png) :

Le graphe généré par testneuron.cpp illustre le processus complet de l'autodifférence pour un neurone à deux entrées.



La structure de chaque boîte est op (si présente) | data (Valeur actuelle) | grad (Dérivée par rapport à la perte finale).

-Forward Pass (Gauche à Droite) : Il représente la combinaison linéaire (Σwixi+b) suivie de la fonction d'activation tanh.

-Backward Pass (Visualisé par grad) : Le gradient se propage de la droite (grad: 1.0000 au niveau de tanh) vers la gauche pour calculer la contribution des poids (w) et du biais (b) à la perte finale.