

Vom neuronalen Netzwerk zu GPT

YouTube-Videos

Andrej Karpathy - Lasst uns GPT bauen: von Grund auf, in Code, Schritt für Schritt erklärt.

Umar Jamil - Attention is all you need (Transformer) - Modellerklärung (inklusive Mathematik), Inferenz und Training

StatQuest mit Josh Starmer - Transformer Neural Networks, die Grundlage von ChatGPT, einfach erklärt!!!

Pascal Poupart - CS480/680 Vorlesung 19: Aufmerksamkeit und Transformer-Netzwerke

Der A.I. Hacker - Michael Phi - Illustrierter Leitfaden zu Transformers Neural Network: Eine Schritt-für-Schritt-Erklärung

Wie ich lerne

Nachdem ich die Hälfte des Buches "Neural Networks and Deep Learning" gelesen hatte, begann ich, das Beispiel eines neuronalen Netzwerks zur Erkennung handgeschriebener Ziffern nachzubauen. Ich habe ein Repository auf GitHub erstellt: <https://github.com/lzwjava/neural-networks-and-zhiwei-learning>.

Das ist der wirklich schwierige Teil. Wenn man es von Grund auf schreiben kann, ohne Code zu kopieren, versteht man es sehr gut.

Mein Replikationscode enthält noch keine Implementierung von `update_mini_batch` und `backprop`. Durch das genaue Beobachten der Variablen in den Phasen des Ladens der Daten, des Feed-Forward-Prozesses und der Auswertung habe ich jedoch ein viel besseres Verständnis für die Vektoren, Dimensionen, Matrizen und die Form der Objekte erlangt.

Und ich begann, die Implementierung von GPT und Transformer zu lernen. Durch Word Embedding und Positionskodierung wird der Text in Zahlen umgewandelt. Im Wesentlichen gibt es dann keinen Unterschied mehr zu einem einfachen neuronalen Netzwerk, das handgeschriebene Ziffern erkennt.

Andrej Karpathys Vortrag "Let's build GPT" ist sehr gut. Er erklärt die Dinge verständlich.

Der erste Grund ist, dass es wirklich von Grund auf beginnt. Wir sehen zunächst, wie der Text generiert wird. Es ist irgendwie verschwommen und zufällig. Der zweite Grund ist, dass Andrej die Dinge sehr intuitiv erklären konnte. Andrej hat das Projekt nanoGPT über mehrere Monate hinweg durchgeführt.

Ich hatte gerade eine neue Idee, um die Qualität der Vorlesung zu beurteilen. Kann der Autor diese Codes wirklich schreiben? Warum verstehe ich sie nicht und welches Thema hat der Autor ausgelassen? Abgesehen von diesen eleganten Diagrammen und Animationen, was sind ihre Schwächen und Mängel?

Zurück zum Thema des maschinellen Lernens selbst. Wie Andrej erwähnt, das Dropout, die Residual Connection, die Self-Attention, die Multi-Head Attention, die Masked Attention.

Indem ich mir mehr der oben genannten Videos ansah, begann ich, ein wenig zu verstehen.

Durch die Positionskodierung mit Sinus- und Cosinus-Funktionen erhalten wir einige Gewichte. Durch das Word Embedding wandeln wir die Wörter in Zahlen um.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Die Pizza kam aus dem Ofen und schmeckte gut.

In diesem Satz, wie weiß der Algorithmus, ob es sich auf Pizza oder Ofen bezieht? Wie berechnen wir die Ähnlichkeiten für jedes Wort im Satz?

Wir möchten eine Reihe von Gewichten haben. Wenn wir das Transformer-Netzwerk verwenden, um die Aufgabe der Übersetzung zu erledigen, kann es jedes Mal, wenn wir einen Satz eingeben, den entsprechenden Satz in einer anderen Sprache ausgeben.

Über das Skalarprodukt hier. Ein Grund, warum wir hier das Skalarprodukt verwenden, ist, dass das Skalarprodukt jede Zahl im Vektor berücksichtigt. Was wäre, wenn wir das quadrierte Skalarprodukt verwenden würden? Wir würden zuerst die Quadrate der Zahlen berechnen und sie dann das Skalarprodukt bilden lassen. Was wäre, wenn wir ein umgekehrtes Skalarprodukt durchführen würden?

Über die Maskierung hier ändern wir die Zahlen der Hälfte der Matrix in negative Unendlichkeit. Dann verwenden wir Softmax, um die Werte auf den Bereich von 0 bis 1 zu bringen. Wie wäre es, wenn wir die Zahlen in der linken unteren Ecke in negative Unendlichkeit ändern?

Plan

Weiterhin Code und Papers lesen sowie Videos ansehen. Einfach Spaß haben und meiner Neugier folgen.

<https://github.com/karpathy/nanoGPT>

<https://github.com/jadore801120/attention-is-all-you-need-pytorch>