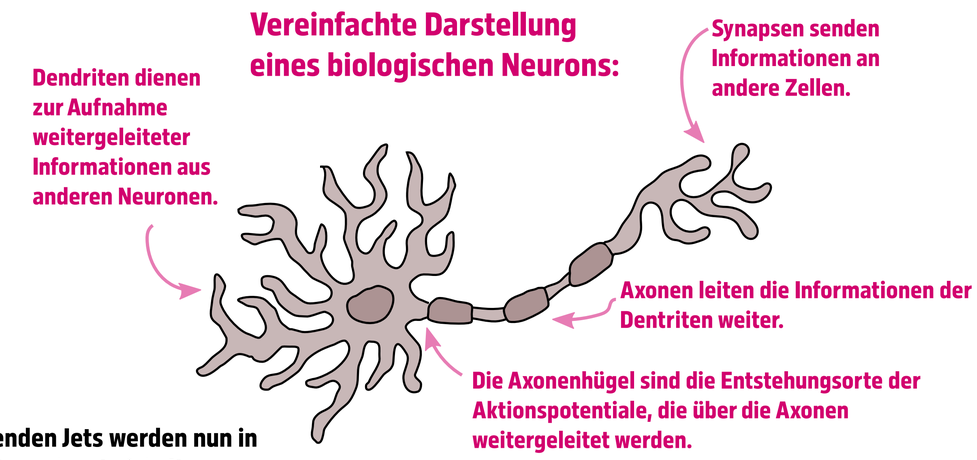
KÜNSTLICHE NEURONALE NETZWERKE und NEURONEN

Künstliche neuronale Netzwerke sind fester Bestandteil vieler Anwendungen des maschinellen Lernens. Haben Sie schon einmal ChatGPT, MidJourney, DeepL oder Grammerly benutzt? Dies sind nur einige Beispiele dafür, wie Ihnen das maschinelle Lernen im Alltag begegnet.

Aber wie funktioniert das eigentlich? Jede Anwendung hat so seine Eigenheiten. Einen Grundbaustein haben sie aber alle gemeinsam, das künstliche neuronale Netzwerk. Da liegt es natürlich in unserer Natur als Teilchenphysiker*innen uns das etwas genauer anzusehen.

Künstliche neuronale Netzwerke bestehen aus Neuronen, inspiriert von den Neuronen wie sie z.B. im menschlichen Gehirn zu finden sind. Sind sie also eine Kopie des menschlichen Gehirns? Nein, das wäre ein bisschen übertrieben, denn so kompliziert wie das menschliche Gehirn ist kein künstliches neuronales Netzwerk. Jedoch sind sie seit 1943 nicht mehr wegzudenken. Richtig populär wurden sie aber erst in den letzen Jahrzehnten mit dem Aufschwung in der Rechenleistung von modernen Computern.

Vereinfachte Darstellung eines biologischen Neurons:



Die in einem Kollisionsexperiment entstehenden Jets werden nun in Zahlen gegossen. Im Anschluss wird ein Netzwerk entwickelt, welches diese Zahlen entgegen nimmt und uns auf deren Grundlage ausgibt zu welcher Art von Jet sie gehören.

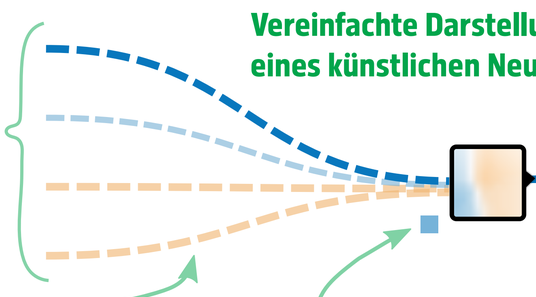
Nachdem die am Anfang in das Netzwerk eingegebenen Zahlen von hinreichend vielen Neuronen bearbeitet wurden, kann im Prinzip jede zugrundeliegende Funktion vom Netzwerk angenähert werden. Funktion meint hier eine mathematische Abbildung die Messdaten entgegen nimmt und uns zurückgibt zu welcher Teilchenart sie gehören.

Natürlich gibt es weitaus kompliziertere Probleme als die binäre Klassifikation. Einige erfordern andere Netzwerkarchitekturen, andere brauchen mehr Rechenleistung oder mehr Eingabedaten um daraus lernen zu können. Jedoch immer entscheidend ist, wonach man das Netzwerk fragt. Algorithmen des maschinellen Lernens können wie Flaschengeister sein. Wenn man sie nach einer Antwort auf ein Problem fragt, dann geben sie häufig eine präzise Antwort auf die Frage, verstehen jedoch nicht den Sinn hinter der Frage. Vom eigenständigen Denken sind sie daher oft noch weit entfernt.

Die ersten Neuronen eines künstlichen neuronalen Netzwerks werden mit den Eingabedaten, also Zahlen, gefüttert. Bis zur finalen Ergebnisausgabe kommunizieren ab dann nur noch Neuronen miteinander, ohne weitere Interaktionen mit uns Menschen.

Zum Schluss gibt jedes Neuron eine Zahl aus. Diese wird entweder als Teil der Eingabe für folgene Neuronen in der nächsten verborgenen Ebene genutzt oder, sollte sich das ausgebende Neuron in der letzten Netzwerk Ebene befinden, als Ausgabe an den Menschen zurückgegeben.

Vereinfachte Darstellung eines künstlichen Neurons:



Jede in das Neuron hineinkommende Zahl wird mit einem Gewicht w, welches auch negativ sein kann, multipliziert. So wird pro Neuron eine gewichtete Summe aller einkommenden Zahlen gebildet. Ein künstliches neuronales Netzwerk hat viele solcher lernbarer Gewichte. Im Lernprozess passt das Netzwerk dann die Gewichte an und ändert so seine Vorhersagen.

An die gewichtete Summe der Zahlen wird jeweils ein 'Bias' (nicht im statistischen Sinne), ein weiterer lernbarer Parameter, addiert. Auch dieser wird im Training modifiziert.

$$y_{\text{aus}} = f\left(\left(\sum_{\text{eingang}=i} w_i x_i\right) + b\right)$$

Die Summe ist linear in jeder in das Neuron einfallenden Zahl x. Man kann sie also als y = wx+b schreiben, wenn man nur eine einkommende Zahl x hätte. Damit das Netzwerk komplexere Strukturen in den Daten lernen kann, wird die gewichtete Summe durch eine nicht lineare Aktivierungsfunktion modifiziert. Dies verhindert das zusammenfallen des Netzwerks zu einer einzigen Ebene.

DAS ZIEL

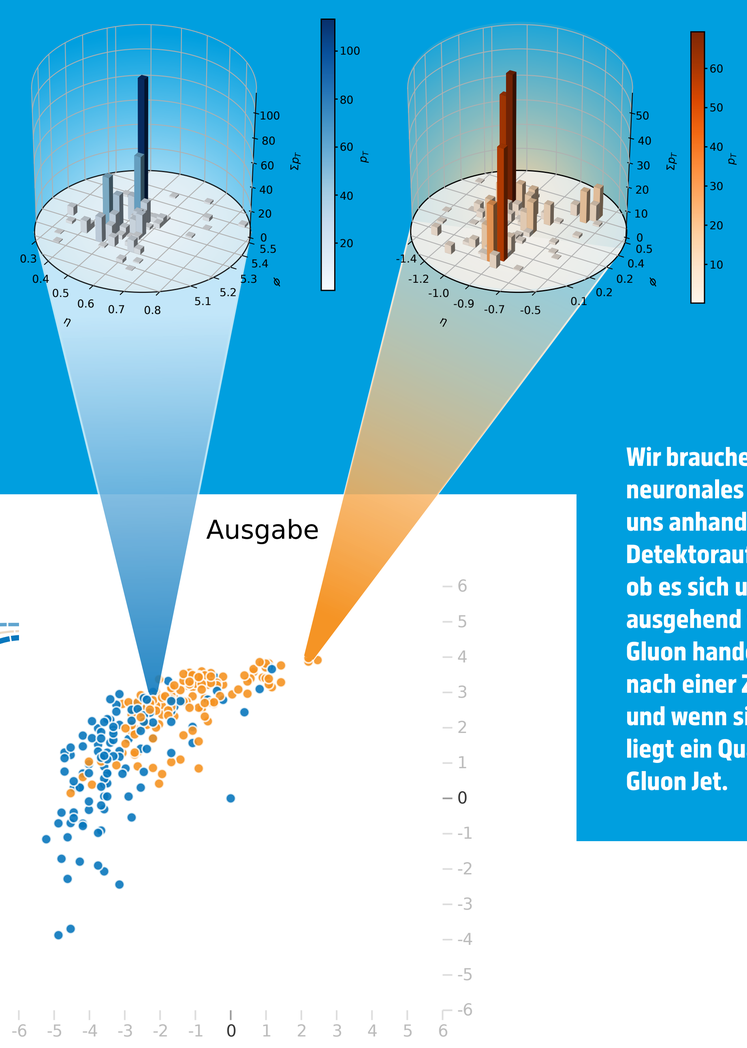
Bei uns kommt ein künstliches neuronales Netzwerk zur binären Klassifikation von Teilchenarten zum Einsatz. Als Analogie kann man sich hier ein Netzwerk vorstellen, welches Bilder von Hunden und Katzen unterscheiden kann. Die Klassifikation von Bildern ist für den Menschen häufig eine leichte Aufgabe, leider sind die von unseren Detektoren aufgenommenen Daten nur schwierig per Auge zu unterscheiden.

Detektoren an Kollisionsexperimenten sind zu verstehen wie eine spezielle Art von Kamera. Das Problem ist aber, dass wir die Teilchen an denen wir interessiert sind nicht mit unseren Detektoren fotografieren können. Wie unterscheiden wir sie dann? Glücklicherweise wandeln sie sich sofort nach ihrer Entstehung in einen Haufen anderer messbarer Teilchen um, wir messen sie als sogenannte Jets. Die im Jet vorhandenen Teilchen heißen Fragmente. Die beiden Teilchenarten an denen wir interessiert sind erzeugen unterschiedlich aussehende Jets.

Die beiden Teilchenarten um die es bei uns geht heißen Quarks und Gluonen. Wissen Sie wo Gluonen entdeckt wurden? Fragen Sie nach!

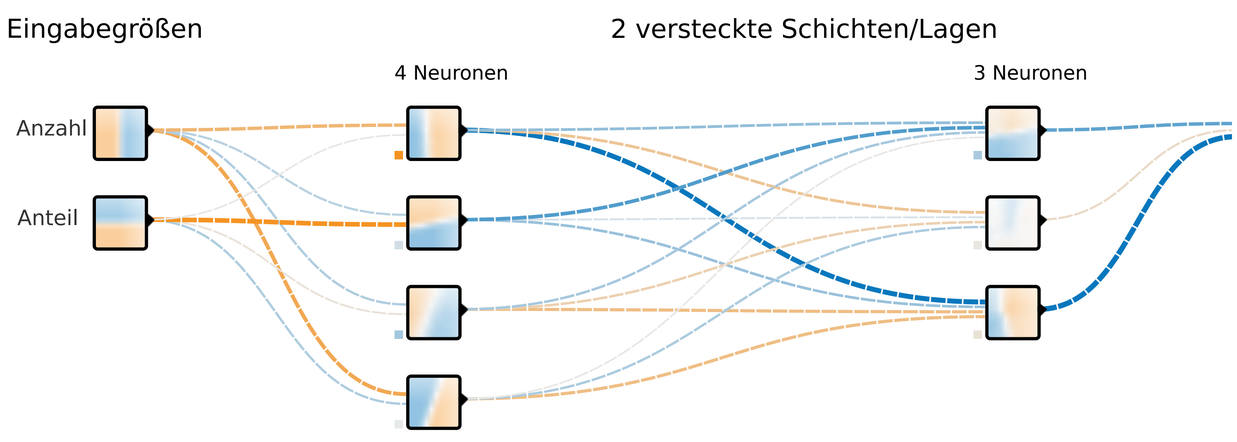
Wir haben hier zwei mit dem Detektor fotografierte Teilchen Jets. In blau der eines Gluons, in orange der eines Quarks. Sehen Sie gravierende Unterschiede?

Jeder Würfel ist ein Pixel unserer Detektor Kamera. Die Würfelhöhe und Farbe deuten an welcher transversale Impuls "pT" im jeweiligen Pixel gemessen wurde.



η (ausgesprochen als "eta") und φ (ausgesprochen als "phi") geben die Pixelkoordinaten unserer Detektorkamera an.

Wir brauchen ein künstliches neuronales Netzwerk welches uns anhand der Detektoraufnahme sagen kann ob es sich um einen Jet ausgehend von einem Quark oder Gluon handelt. Wir fragen also nach einer Zahl zwischen 0 und 1 und wenn sie größer als 0.5 ist liegt ein Quark Jet vor, sonst ein Gluon Jet.



TRAINING des NETZWERKS

Wie lernt ein künstliches neuronales Netzwerk ein Problem zu lösen? Wie wir – es trainiert!

Zu Beginn, bevor das künstliche neuronale Netzwerk trainiert hat, entscheidet es zufällig ob ein Jet den es zu sehen bekommt ein Quark oder Gluon Jet ist. Es liefert uns also in der Hälfte aller Fälle die richtige Antwort. Leider ist das für uns nicht nützlicher als eine Münze zu werfen.

Um dem Netzwerk etwas beizubringen, simulieren wir die Jets so wie sie ein Detektor sehen würde. In unserer Simulation wissen wir natürlich genau ob ein Quark oder Gluon Jet vorliegt. Da unser Netzwerk eine Zahl zwischen 0 und 1 ausgibt, bringen wir ihm bei, eine Zahl nahe 1 auszugeben wenn es denkt die Eingabe sei ein Quark, sonst soll es eine Zahl nahe 0 ausgeben.

Wir zeigen dem Netzwerk nach und nach Quark Jets und Gluon Jets. Jedes Mal passen wir die Gewichte und Biasparameter des Netzwerks an, sodass es möglichst oft die richtige Vorhersage trifft.

Fangen wir also mit einem simulierten Quark Jet an. Wir zeigen dem Netzwerk unseren Jet und notieren den Ausgabewert des Netzwerks. Dann modifizieren wir jedes einzelne Gewicht und jeden Biasparameter ein bisschen, so dass das Netzwerk einen etwas höheren Wert als den ursprünglichen ausgibt.

Jetzt wird es Zeit für ein Gluon. Wieder zeigen wir dem Netzwerk unseren Jet und notieren den Ausgabewert des Netzwerks. Wir justieren die Netzwerkparameter jetzt so, dass das Netzwerk einen etwas kleineren Wert als zu Beginn ausgibt.

Mathematisch wird dieses Verfahren als Minimierung der Verlustfunktion beschrieben. Der Verlust ist eine Größe, die angibt wie groß der Fehler des Netzwerks bei der Klassifizierung der Jets ist. Sagt das Netzwerk zu kleine Werte für Quark Jets voraus, dann ist der Verlust groß. Ebenfalls ist der Verlust groß, wenn es zu große Werte für Gluon Jets voraus sagt. Das Minimieren der Verlustfunktion ist also eine mathematische Art und Weise dem Netzwerk mitzuteilen was wir von ihm möchten. Wir möchten, dass es Jets beider Teilchenarten möglichst korrekt klassifiziert. Nun können wir andere Algorithmen nutzen um die Netzwerkparameter zu ändern, sodass der Verlust minimal wird. Dies ist der Lernprozess des Netzwerks.

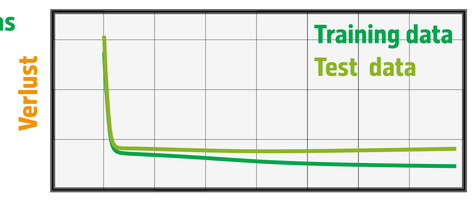
Natürlich zeigen wir dem Netzwerk nicht nur einen Quark und einen Gluon Jet. Wir simulieren tausende Jets. Jeden davon zeigen wir dem Netzwerk mehrmals. Daher ist große Rechenleistung für das Trainieren und Testen von Netzwerken nötig.

Wir trainieren und testen das Netzwerk auf unabhängigen Datensätzen. Es kann nur auf dem Trainingsdatensatz lernen. Wir nutzen den Testdatensatz um zu überprüfen, ob das Netzwerk tatsächlich dazu lernt und verbieten hier die Anpassung der Netzwerkparameter.

Hier ein Beispiel zur Minimierung des Verlusts, engl. "loss":

Das Netzwerk hat die einfachen Dinge schnell gelernt. Nun braucht es länger um die Details zu verstehen.

Zu Beginn lernt das Netzwerk schnell. Der Verlust sinkt rapide.



Nach langem Training wird es plötzlich schlechter auf dem Testdatensatz. Wie kann das sein? Es hat angefangen die Trainingsdaten auswendig zu lernen und seine Parameter dahingehend anzupassen.

Probieren Sie es selbst einmal an unserem Computer aus!

Henry Day-Hall
and Konrad Helms