
+

•

Workshop 2:

○

Erklärung von Netzen + Training mittels Tensorflow-Playground
(playground.tensorflow.org) –

Welche Netze sind besonders effektiv - konvergieren schnell? –

Welche können sich trotz hoher Komplexität nicht anpassen?

Franck de Préaumont Aliénor - 2010837068

Kaya Gökan – 2010837313


Training eines einfachen
neuronalen Netze mit "default
setting" von Tensor Flow
Playground:
Konvergieren die Modelle gut?
Sonst welche Lösungen haben
wir?

Aktivierungsfunktion

Deep learning

Feature engineering

Regularisierung und overfitting



Neuronale Netze mit Tensorflow Playground

- Die Elemente von Tensorflow playground:
 - **Lernrate:** Bestimmung der Lerngeschwindigkeit
 - **Aktivierungsfunktion: Softout-Funktionen tanh (Hyperbolic Tangent)** [In einem neuronalen Netzwerk ist die Aktivierungsfunktion dafür zuständig, die summierte gewichtete Eingabe vom Knoten in die Aktivierung des Knotens oder der Ausgabe für diese Eingabe umzuwandeln]
 - **Regularisierung:** Um eine Überanpassung zu verhindern -> Durch Regularisierung wird das Gewicht starker/schwacher Verbindungen langsam erhöht/verringert, um die Pattern-Klassifizierung schärfer zu machen.
 - **L1:** - ist effektiv in spärlichen Feature-Spaces, in denen einige unter vielen ausgewählt werden müssen
 - trifft eine Auswahl und weist große Gewichtswerte zu und macht die Gewichte der nicht ausgewählten sehr klein (oder null)
 - **L2:** - ist wirksam bei Eingaben, die korreliert sind
 - steuert den Gewichtswert entsprechend dem Korrelationsgrund
 - **Blue line zeigt ein positives Gewicht und die Orange line weist ein negatives Gewicht zu**
- => Die Konzentration liegt auf die "Klassifikation"

I- Training: Einfaches Model gemäß Standardeinstellungen

1



Wir trainieren die Classification Problemen mit Standardeinstellungen von Tensorflow Playground:

Die ersten drei Netze können mit den Standardeinstellungen gelöst werden. Der vierte kann jedoch nicht.

Der erste und einfachste Netz erreicht ein Total Loss und ein training Loss von 0,001 mit 101 Epoch.

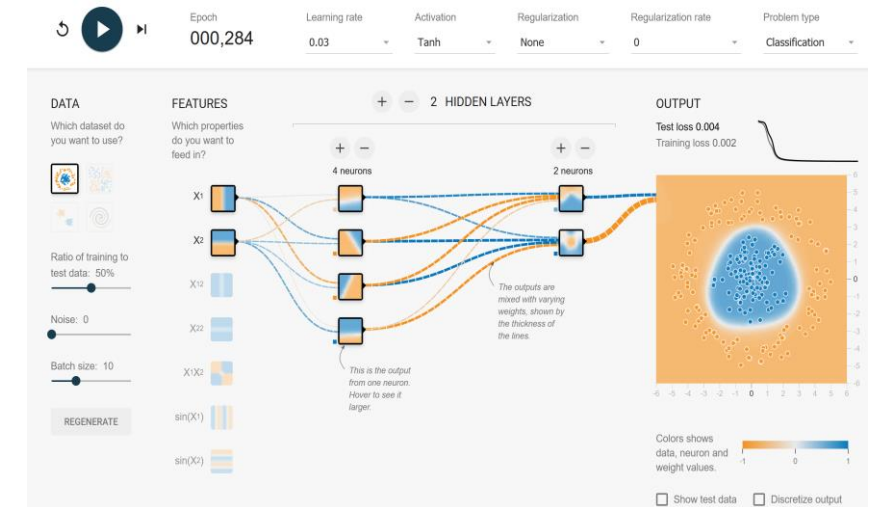
Das zweite Netz konvergiert nicht so schnell wie das erste: ein Training loss von 0,002 und ein Test Loss von 0,004 ist erreicht mit 284 Epoch.

Das dritte Netz konvergiert noch langsamer. Es braucht 543 Epoch um 0,002 Test Loss und 0,002 Training Loss zu erreichen

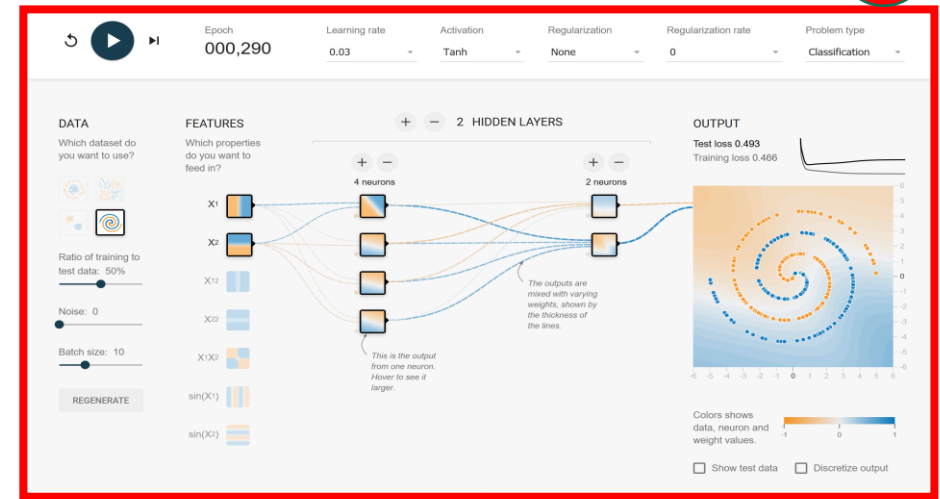
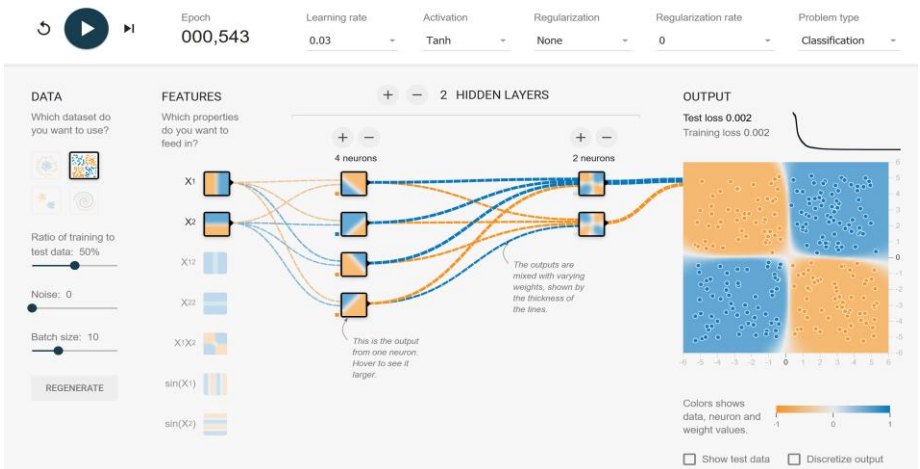
Die vierte Modellierung erreicht mit 290 ein Test Loss von 0,493 und ein Training loss von 0,466. Das vierte Netz konvergiert nicht richtig.

Wie können wir das Problem am bestens lösen?

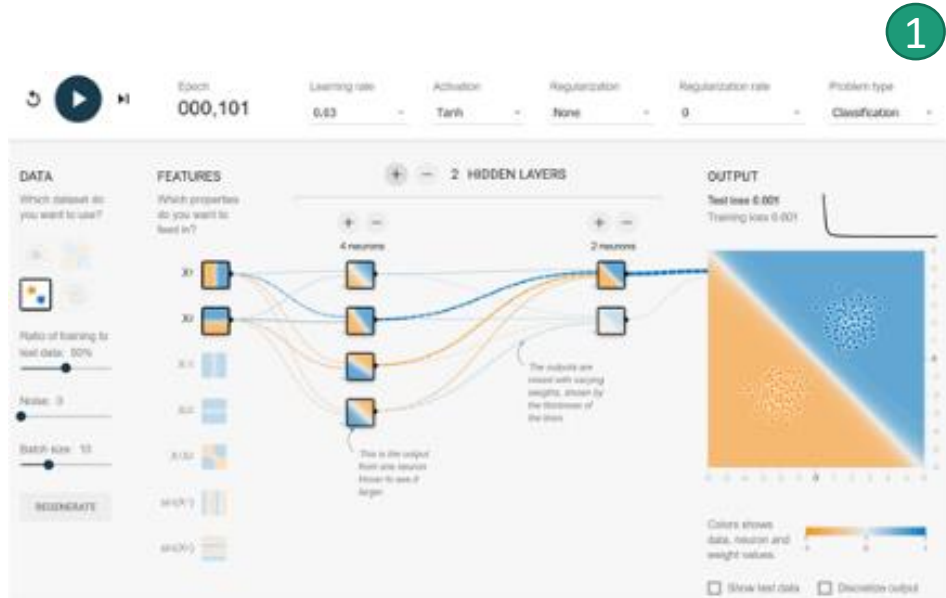
2



4

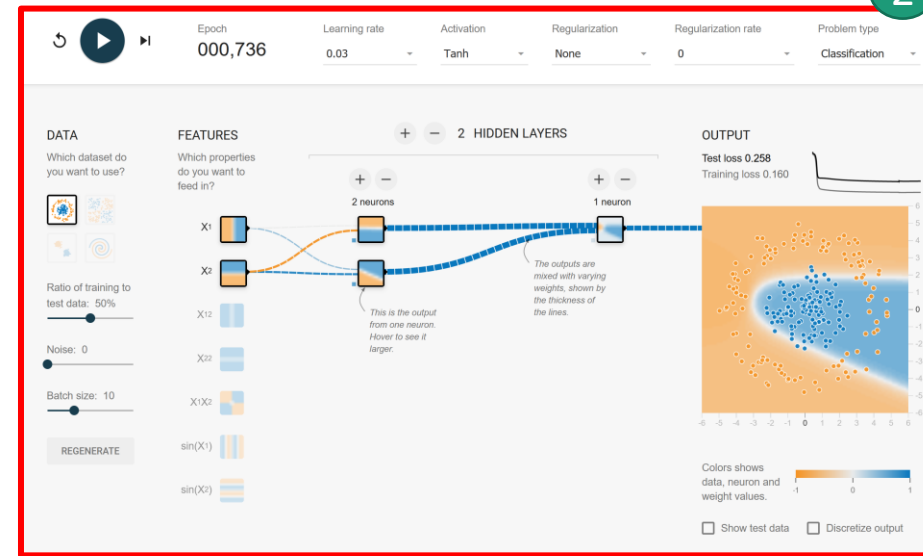


Einfluss von der Anzahl den Neuronen: Warum Neuronen sich im Verborgenen erhöhen ?

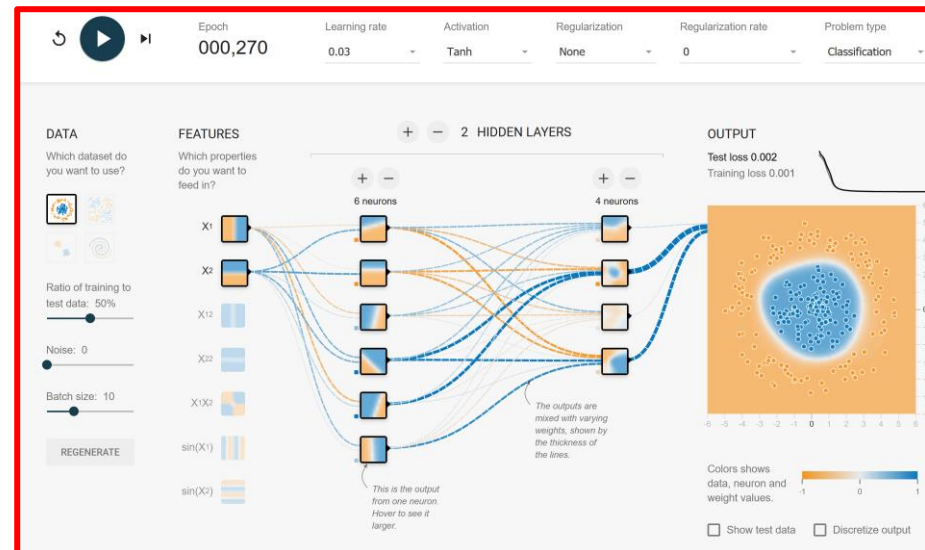


101 Epoche - mit 2 Hidden layers und 6 Neuronen

=> Das Hinzufügen von Neuronen in der verborgenen Schicht bietet Flexibilität, um unterschiedliche Gewichte und parallele Berechnungen zuzuweisen. Das Hinzufügen von Neuronen nach einem gewissen Grad ist jedoch mit geringem Nutzen rechenintensiv.



736 Epoch – mit 2 Hidden layers und 3 Neuronen



270 Epoch – mit 2 Hidden layers und 10 Neuronen

Im ersten Training (mit Standard Settings) erreicht man ein Test loss und und ein Training loss von 0,001 in 101 Epoch mit 2 Hidden Layers und insgesamt 6 Neuronen.

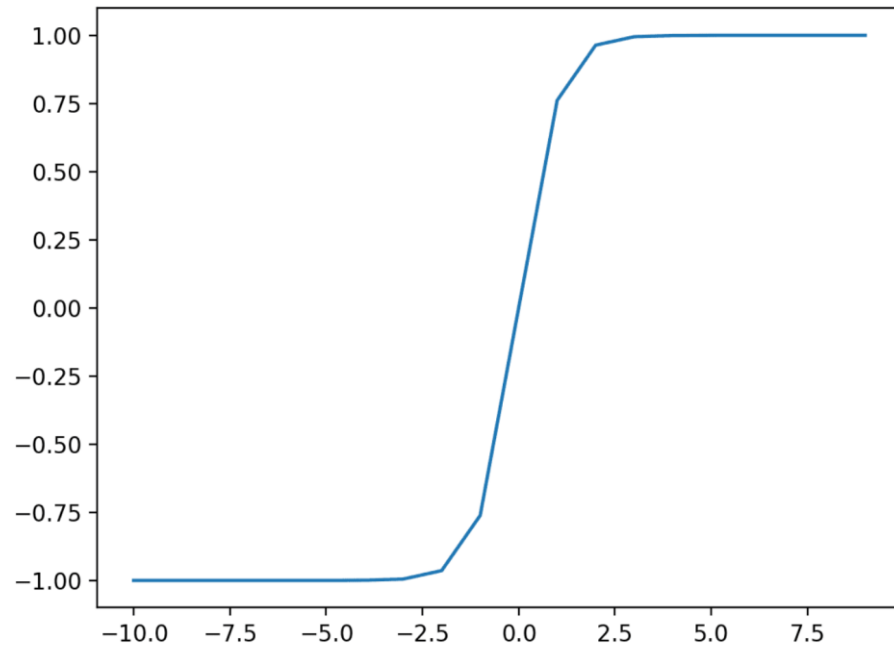
Beim zweiten Training ist der Anzahl von Neuronen gesunken, die Performance ist da schlechter geworden. Mit 736 Epoch erreicht man ein Test Loss von 0,258 und ein training loss von 0,160.

Im dritten Training erhöhen wir der Anzahl an Neuronen (10). das Netz konvergiert aber nicht so gut wie bei der 1. Training. Ein Test Loss von 0,002 und ein training loss wird erreicht in 270 Epochs.

Standardeinstellungen von Tensorflow Playground

Standardeinstellungen von Tensorflow Playground sind:

- Eingabefunktionen (x_1 , x_2) + 1 Ausgabeetikett,
- Lernrate = 0,03
- Regularisierung: keine
- Aktivierungsfunktion: Tanh



Plot of Inputs vs. Outputs for the Tanh Activation Function.

The hyperbolische Tangentenaktivierung (hyperbolic tangent activation):

Wie Sie sehen können, liegt der Wertebereich zwischen -1 und 1. Abgesehen davon alle anderen Eigenschaften von tanh Funktion sind die gleichen wie die der Sigmoidfunktion. Ähnlich wie bei Sigmoid ist die Tanh-Funktion an allen Punkten kontinuierlich und differenzierbar. Die Tanh funktion bringt eine [bessere Performanz](#) als die Sigmoid funktion.

Normalerweise wird tanh gegenüber der Sigmoid-Funktion bevorzugt, da sie nullpunktzentriert (zero centered) ist und die Gradienten nicht auf eine bestimmte Richtung beschränkt sind.

Die Aktivierungsfunktion, die in versteckten Schichten verwendet wird, wird typischerweise basierend auf dem Typ der Architektur des neuronalen Netzwerks gewählt.

Wir verstehen, dass die Verwendung einer Aktivierungsfunktion einen zusätzlichen Schritt auf jeder Schicht während der Vorwärtspropagation einführt. Die Frage ist nun - wenn die Aktivierungsfunktion die Komplexität so sehr erhöht, können wir dann auf eine Aktivierungsfunktion verzichten? Stellen Sie sich ein neuronales Netzwerk ohne die Aktivierungsfunktionen vor. In diesem Fall würde jedes Neuron nur eine lineare Transformation der Eingaben mithilfe der Gewichte und Vorspannungen durchführen. Obwohl lineare Transformationen das neuronale Netzwerk einfacher machen, wäre dieses Netzwerk weniger leistungsfähig und nicht in der Lage, die komplexen Muster aus den Daten zu lernen.

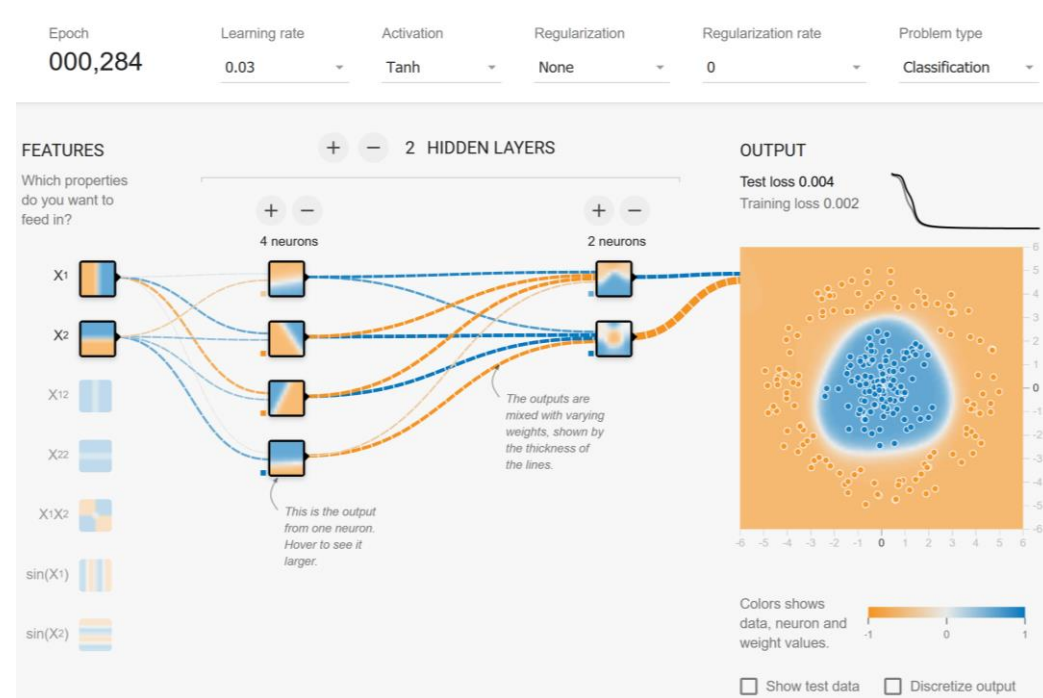
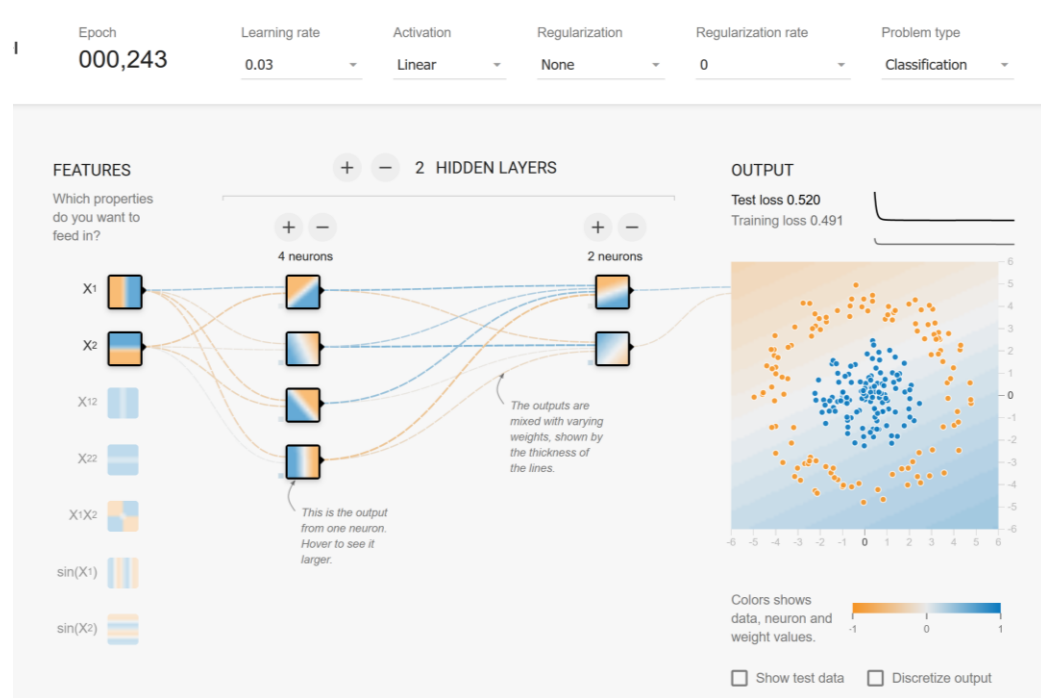
Anwendung Linearer Funktion

Warum nutzen wir keine Lineare Funktion?

In einem neuronalen Netzwerk verwenden wir für alle Klassifizierungsprobleme nur nichtlineare Aktivierungsfunktionen, da unser Ausgabeetikett zwischen 0 und 1 liegt, während lineare Aktivierungsfunktionen eine beliebige Zahl zwischen -Infinity und Infinity liefern können. Infolgedessen wird die Ausgabe zu keinem Zeitpunkt konvergiert.

Wir sehen da, dass die Linear Funktion eine schlechtere Performanz als die Tanh Funktion bringt. Bei der Anwendung einer Linear Funktion konvergiert das Netz langsam. 243 Epichs werden gebraucht um ein Test Loss von 0,520 zu erreichen und ein Training loss von 0,491 zu erreichen.

Im gegensatz braucht das Netz mit der Tanh Aktivierungsfunktion 284 Epochs um 0,004 test loss und 0,002 zu erreichen.



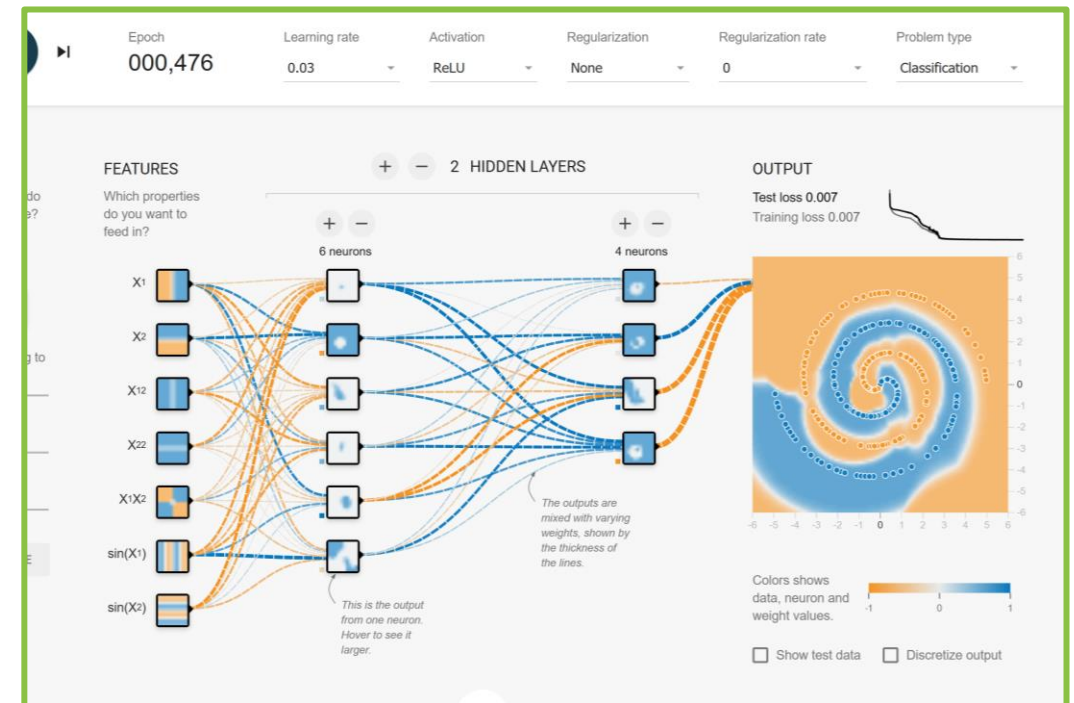
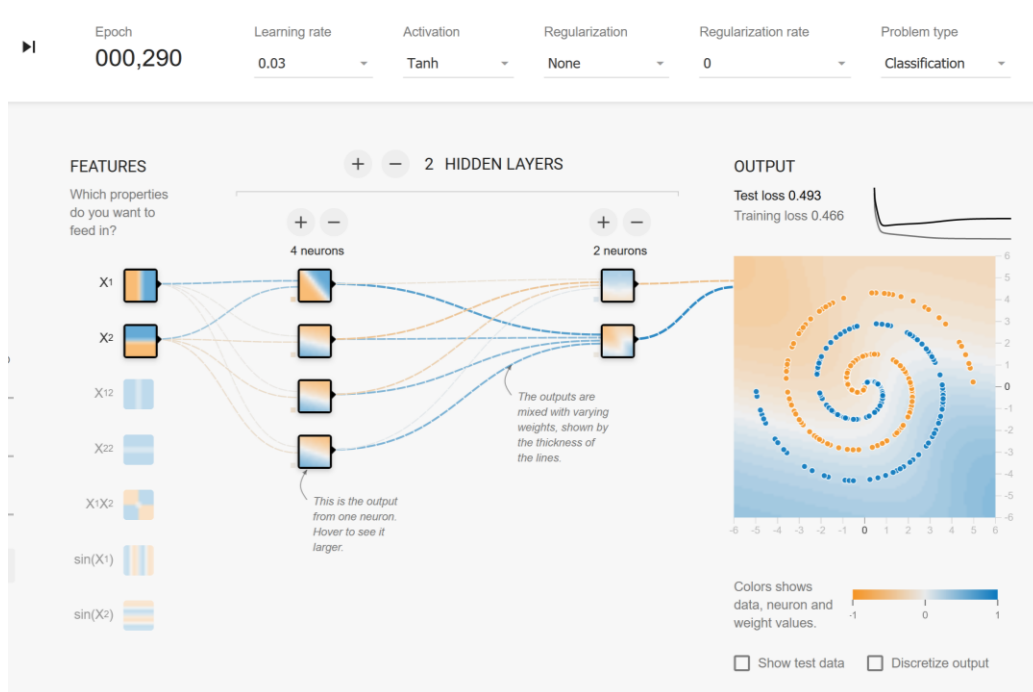
II. Lösung Feature Engineering

Der spirale Datensatz kann nicht gut konvergieren mit Standard Einstellungen von Tensorflow. Eine Lösung um die Konvergenz zu verbessern ist.

Neue Eingabefunktionen hinzufügen:

Sie nehmen die Eingabemerkmale und quadrieren sie, multiplizieren sie miteinander, nehmen sin und cos und speisen sie in ein flaches neuronales Netzwerk ein. Dies steht für klassisches maschinelles Lernen und Feature Engineering.

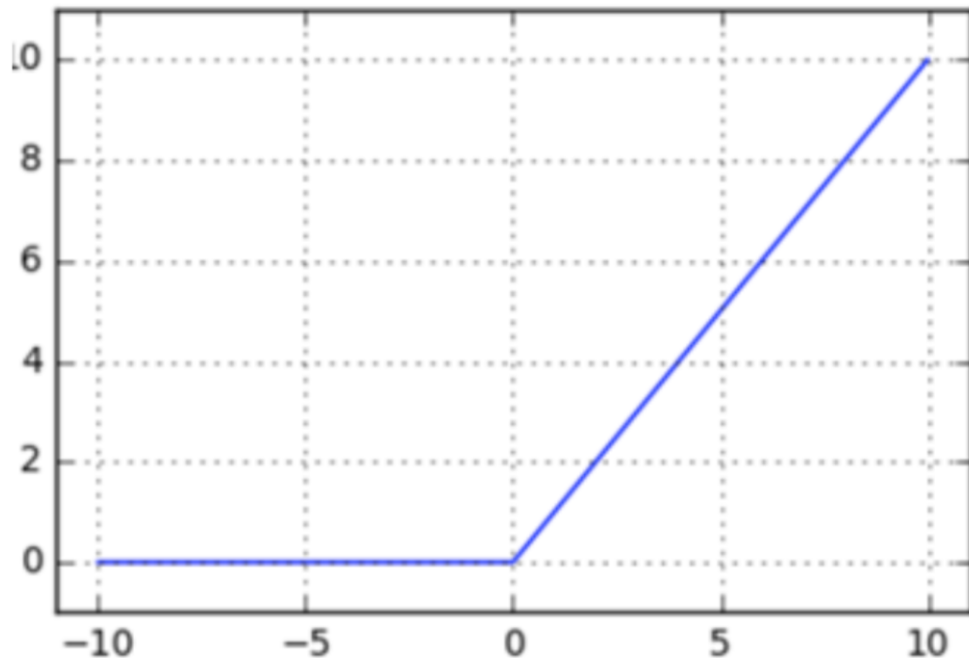
Unten haben wir neue “Features” hinzugefügt, die Anzahl an Neuronen erhöht und eine Relu Funktion benutzt (anstatt die Tanh Funktion). Mit Standard Einstellungen von Tensorflow Playground ist den Test Loss 0,493 und den training loss 0,466 nach 290 Epoch. Mit den neuen Einstellungen kriegen wir ein Test Loss und ein Training loss von 0,007, nach 476 epochs.



Die Anwendung der Relu Funktion

Limit der tanh-Funktion: die Tanh funktion hat ein « [Vanishing Gradient Problem](#) » wie eine sigmoid Funktion. refuse to learn and become slow. Der Wert für Y reagiert sehr wenig oder gar nicht auf die Änderungen des Wertes von X. Das heißt, eine große Änderung der Eingabe der Sigmoidf-Funktion wird eine kleine Änderung der Ausgabe verursachen, das Netzwerk wird definitiv

ReLU function



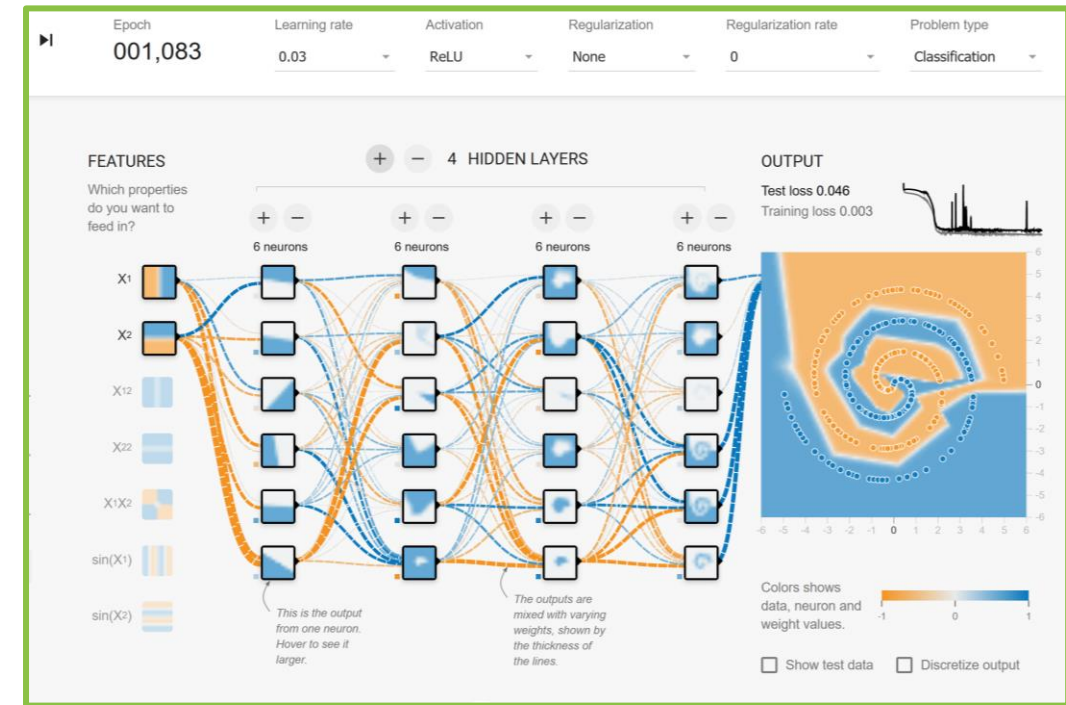
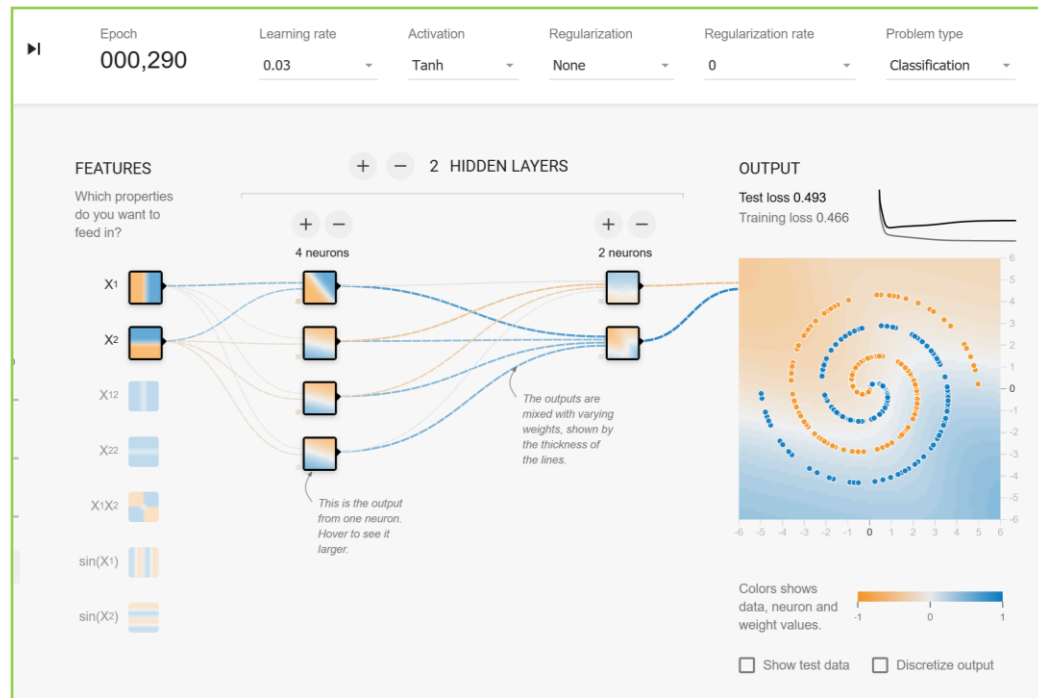
Die gleichgerichtete lineare Aktivierungsfunktion oder kurz ReLU ist eine stückweise lineare Funktion, die den Eingang direkt ausgibt, wenn er positiv ist, andernfalls gibt sie Null aus. Es ist zur Standardaktivierungsfunktion für viele Arten von neuronalen Netzen geworden, da ein Modell, das es verwendet, einfacher zu trainieren ist und häufig eine bessere Leistung erzielt. **Die Relu Funktion hat kein « Vanishing Gradient Problem »**

III. Deep learning Lösung

Eine andere Lösung um die Konvergenz von dem Spiralen Datensatz zu verbessern ist den Anzahl an Hidden Layers zu erhöhen.

Ein anderer Ansatz ist das Deep Learning. Sie manipulieren die Eingabefunktionen nicht geschickt (ein Prozess, der bei weniger trivialen Beispielen viel schwieriger wird). Sie verwenden einfach nur die Roheingaben X_1 und X_2 . [Sie fügen jedoch weitere Ebenen hinzu.](#)

Zum Beispiel hier unten haben wir der Anzahl an Hidden layers erhöht, eine Relu Funktion benutzt und die Anzahl an Neuronen erhöht. das Netz konvergiert aber langsam. Der Test loss ist 0,046 und der training loss ist 0,003 nach 1083 Epochs.



Wie wird die Konvergenz von der Regularisierung beeinflusst?

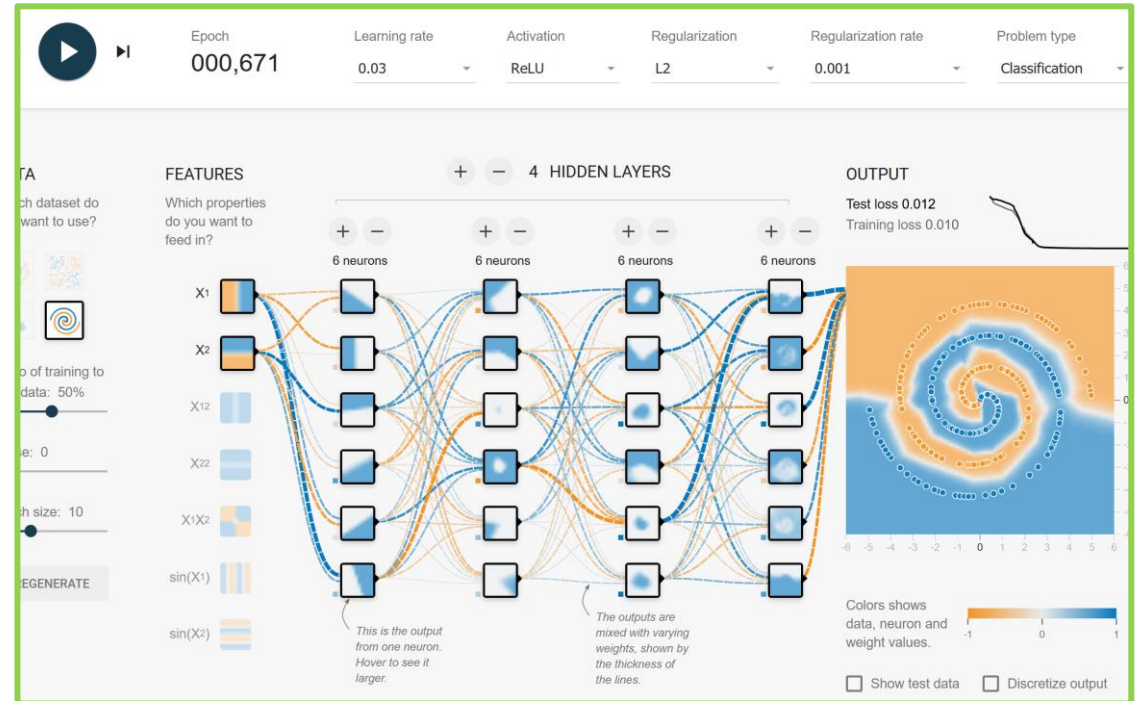
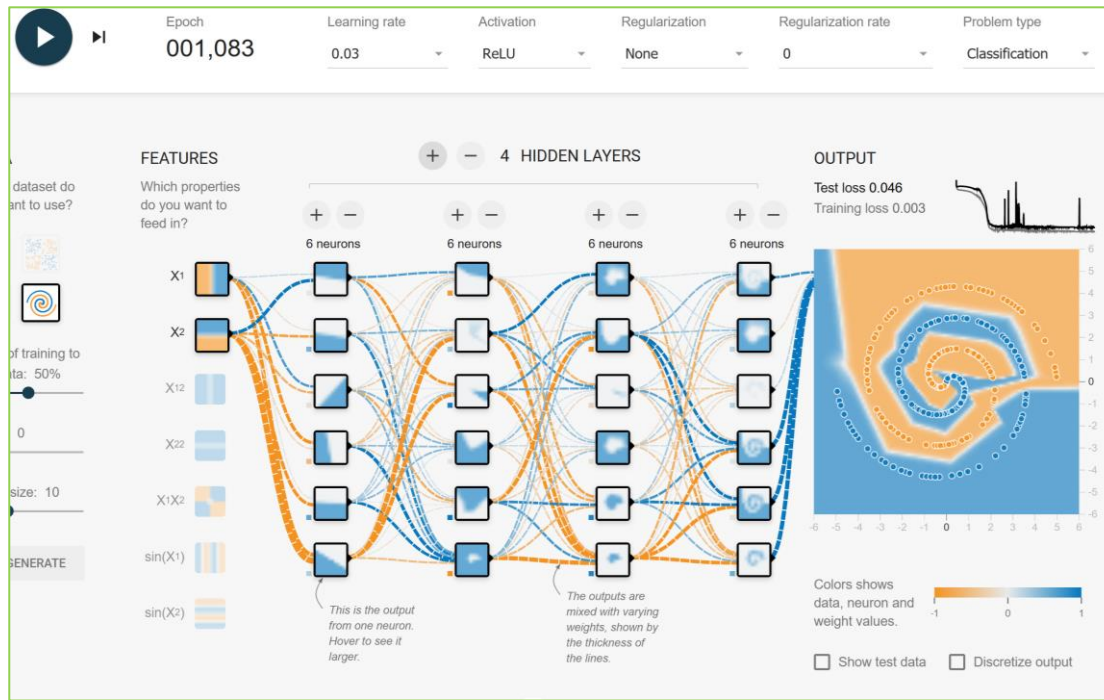
L2 Regularisierung

Wir können auch die Konvergenz der spiralen Netz mit der Regularisierung beeinflussen.

Regularisierungsrate: Dies ist eine Technik zum Einstellen der Aktivierungsfunktion durch Hinzufügen eines zusätzlichen Strafbegriffs in der Fehlerfunktion. Der Wert der Regularisierungsrate variiert auf dem Tensorflow-Spielplatz zwischen 0 und 10.

Hinweis: L2 hilft, das Modell zu verallgemeinern und das Modell weniger vor Überanpassung zu schützen. Der Unterschied zwischen Testverlust und Trainingsverlust ist viel geringer als der ohne Verwendung der L2-Regularisierung

Regularisierung L2 wurde hinzugefügt. Infolgedessen sehen wir, dass wir bessere Ergebnisse bekommen. Der Test loss wird 0,12 und der Training loss 0,010 nach 671 Epochs.

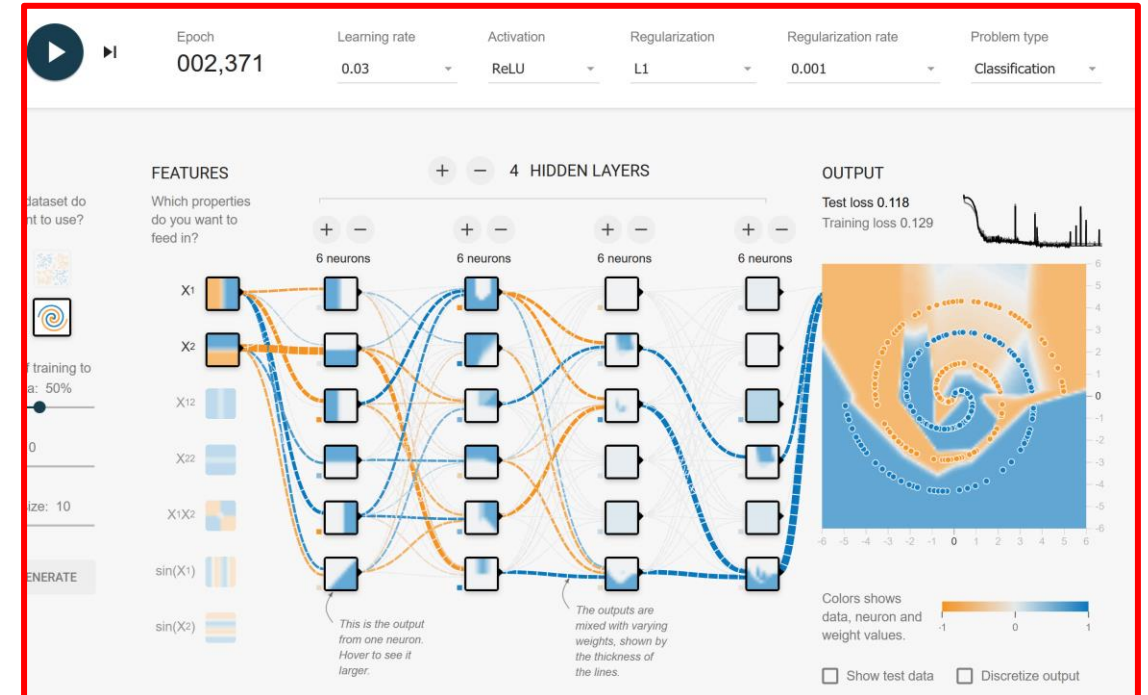
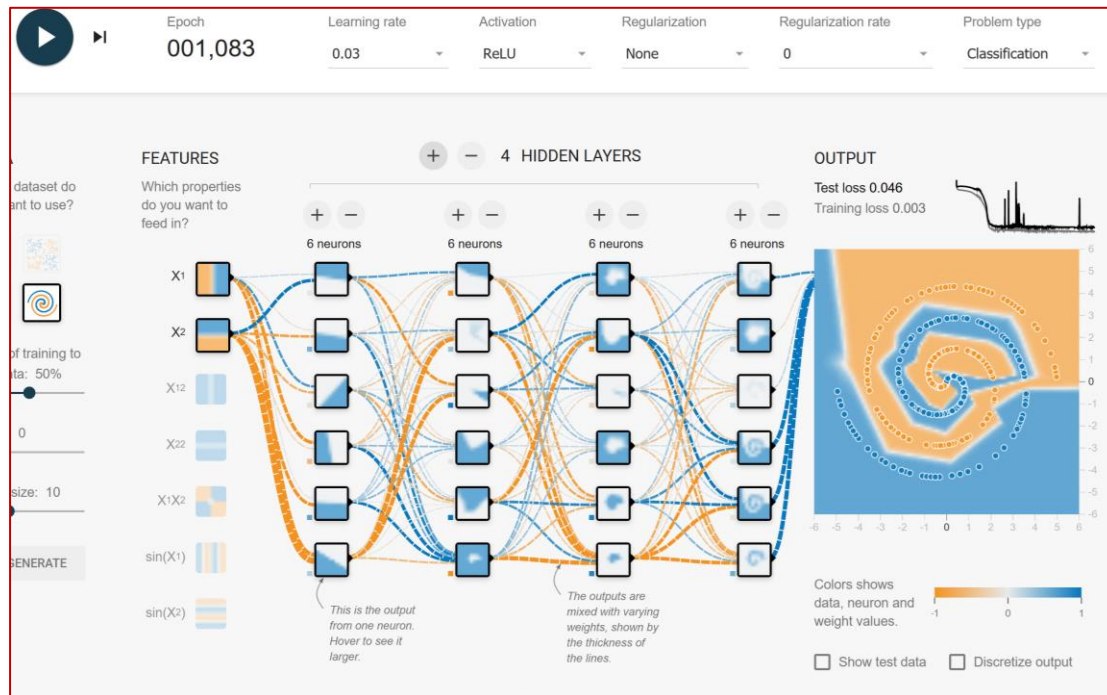


L1 Regularisierung

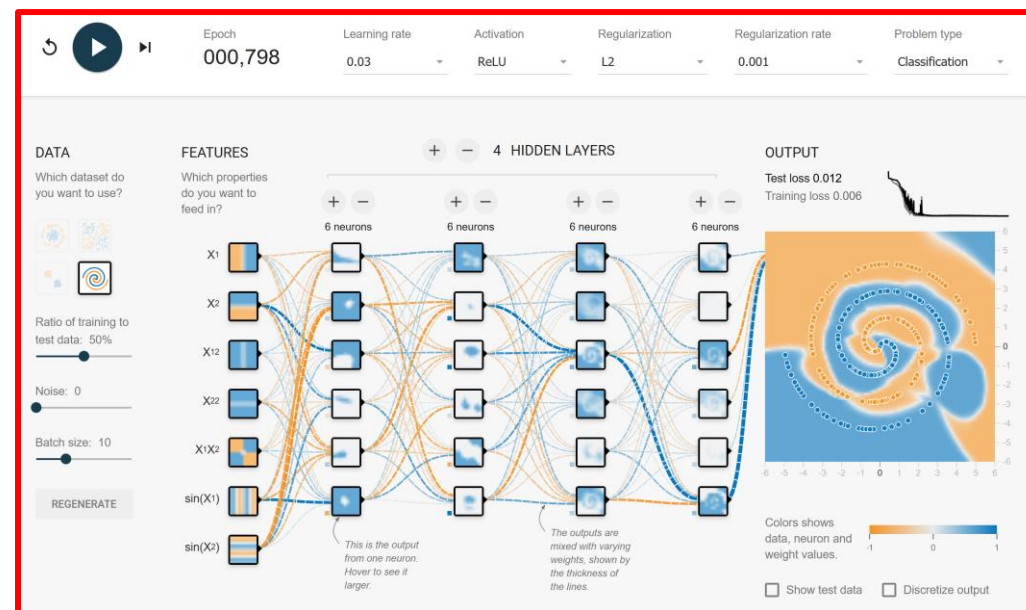
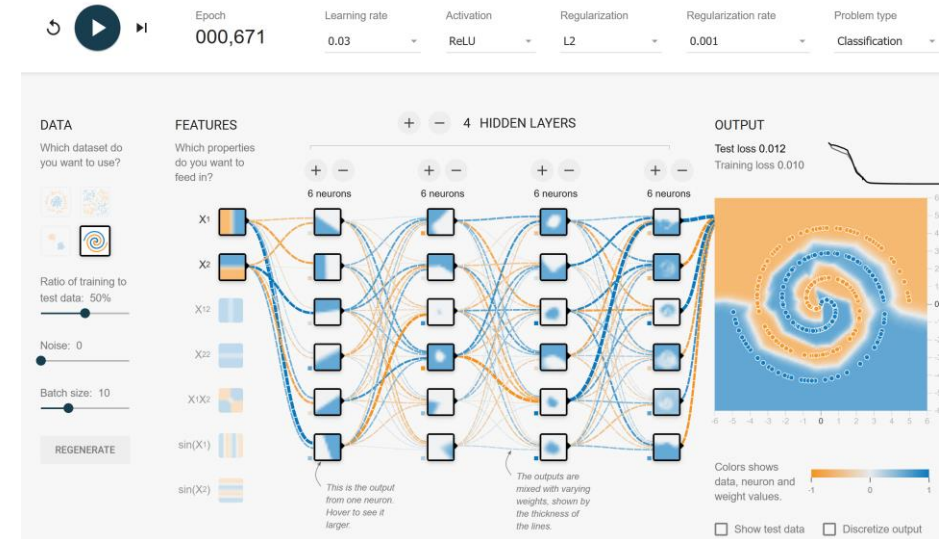
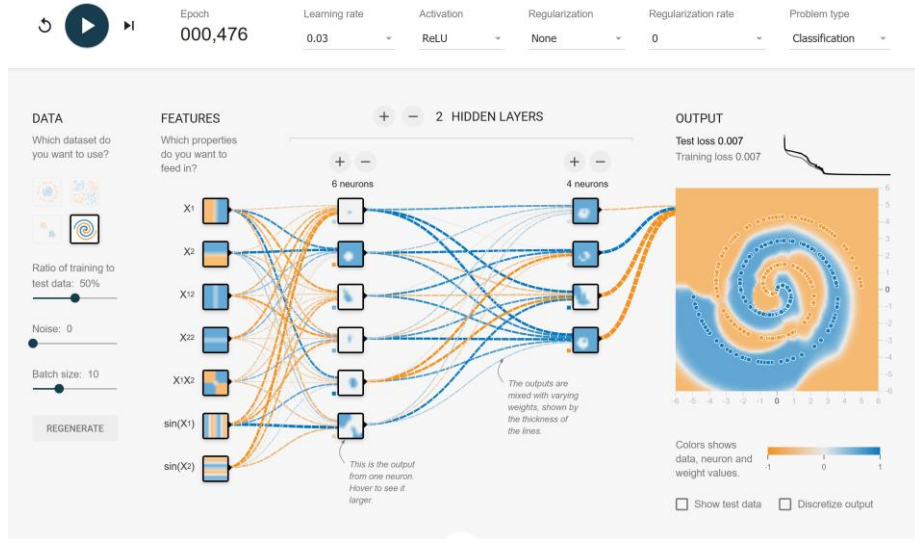
Die [L1 Regularisierung](#) kann auch die Konvergenz beeinflussen:

L1 hilft auch bei der Verallgemeinerung des Modells und ermöglicht eine stärkere Schrumpfung der Kantengewichte, wobei ein Großteil der Kantengewichte mit 0 oder nahe 0 liegen kann. Die Regularisierungsrate L1 hilft auch bei der Verallgemeinerung des Modells und ermöglicht eine stärkere Schrumpfung auf Kantengewichte, wobei ein Großteil der Kantengewichte mit 0 oder nahe 0 Werten liegen kann.

Die L1 Regularisierung wurde hinzugefügt, die Konvergenz wird aber Schwieriger, der Test Loss wird 0,118 und den Training Loss: 0,129, nach 2371 Epochs.



IV. Die Kombination: feature + deep learning



Wir kombinieren die Lösung « additional features » und « deep learning », aber das Ergebnis ist nicht wie erwartet. Im Gegensatz braucht Tensorflow Playground 798 epoch (anstatt 671 Epoch bei der deep learning lösung) um das gleiche Test loss zu bekommen



Overfitting

- Überanpassung ist ein häufiges Problem, das als die Unfähigkeit eines trainierten maschinellen Lernmodells definiert wird, gut auf unsichtbare Daten zu verallgemeinern, aber dasselbe Modell funktioniert gut mit den Daten, auf denen es trainiert wurde.
- Der Hauptzweck von Dropout besteht darin, den Effekt einer Überanpassung innerhalb eines trainierten Netzwerks zu minimieren.
- **Die Dropout-Technik reduziert zufällig die Anzahl der miteinander verbundenen Neuronen innerhalb eines neuronalen Netzwerks. Bei jedem Trainingsschritt hat jedes Neuron die Möglichkeit, aus dem gesammelten Beitrag verbundener Neuronen herausgelassen oder vielmehr ausgeschlossen zu werden.**
- Diese Technik minimiert die Überanpassung, da jedes Neuron unabhängig genug wird, in dem Sinne, dass die Neuronen innerhalb der Schichten Gewichtswerte lernen, die nicht auf der Zusammenarbeit seiner benachbarten Neuronen beruhen.

Quellen

[Understanding And Implementing Dropout In TensorFlow And Keras | by Richmond Alake | Towards Data Science](#)

<https://sigmaxi.siu.edu/Raj%20Workshop%20Exercises%20with%20Tensorflow%20Playground.pdf>

<https://www.druva.com/blog/understanding-neural-networks-through-visualization/>

<https://medium.com/@andrewt3000/understanding-tensorflow-playground-c20cdb7a250b>

[A Gentle Introduction to the Rectified Linear Unit \(ReLU\) \(machinelearningmastery.com\)](#)

[16. TanH, Sigmoid and ReLU - A clear view \(Vanishing Gradient problem\) – YouTube](#)

[Comparison of Sigmoid, Tanh and ReLU Activation Functions – AITUDE](#)

[Deep Learning with TensorFlow Playground | by Muhammad Rizwan Khan | DataDrivenInvestor](#)

[Why tanh outperforms sigmoid | Analytics Vidhya \(medium.com\)](#)

[Activation Functions | Fundamentals Of Deep Learning \(analyticsvidhya.com\)](#)