Elektronik für Informatiker

Eine Einführung in Analoge und Digitale Systeme für Informatiker mit Elektronikgrundlagen und Signalverarbeitung

Prof. Dr. Stefan Bosse

Universität Koblenz - Praktische Informatik

Softwareentwurf von Analogrechnern

Bisher waren die Schaltungen gegeben oder wurden manuell aus mathematischen Modellen und Funktionen abgeleitet. Das Ziel ist aber der automatisierte Entwurf von Analogrechnern.

Softwareentwurf von Analogrechnern

Bisher waren die Schaltungen gegeben oder wurden manuell aus mathematischen Modellen und Funktionen abgeleitet. Das Ziel ist aber der automatisierte Entwurf von Analogrechnern.

Simulation ist ein Hilfsmittel das Verhalten von Analogschaltungen zu untersuchen und zu analysieren.

Softwareentwurf von Analogrechnern

Bisher waren die Schaltungen gegeben oder wurden manuell aus mathematischen Modellen und Funktionen abgeleitet. Das Ziel ist aber der automatisierte Entwurf von Analogrechnern.

Simulation ist ein Hilfsmittel das Verhalten von Analogschaltungen zu untersuchen und zu analysieren.

Aber wie kann man automatisiert eine Analogschaltung aus mathematischen Funktionen und Daten ableiten? Wir wollen dies am Beispiel des Analogen Künstlichen Neuronalen Netzwerks begreifbar machen.

Künstliche Neuronale Netzwerke

[Bosse, 2025, SysInt Conference]

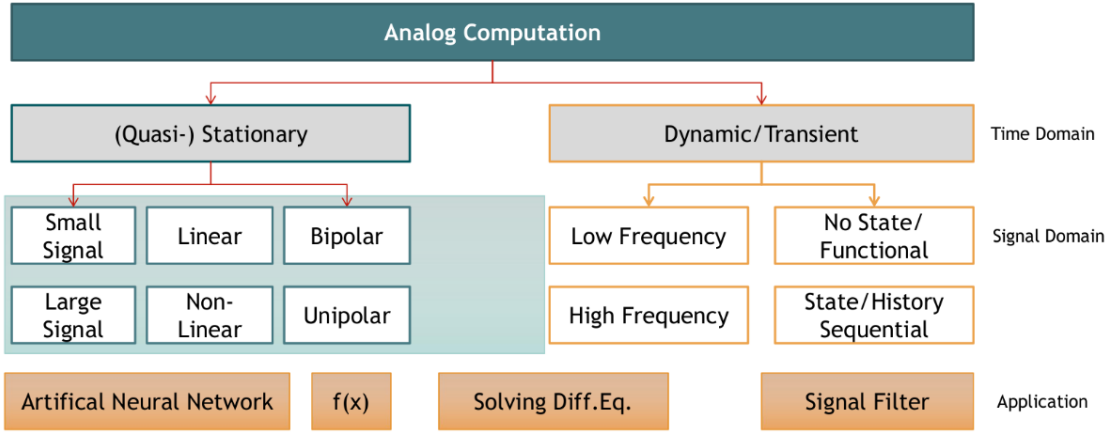


Abb. 1. Bei der Lösung von DGL hatten wir dynamische (transitorische) Systeme. Bei der Berechnung von KNN Modellen sind wird wirder im stationären (funktionalen) Bereich.

Künstliche Neuronale Netzwerke

[Bosse, 2025, SysInt Conference]

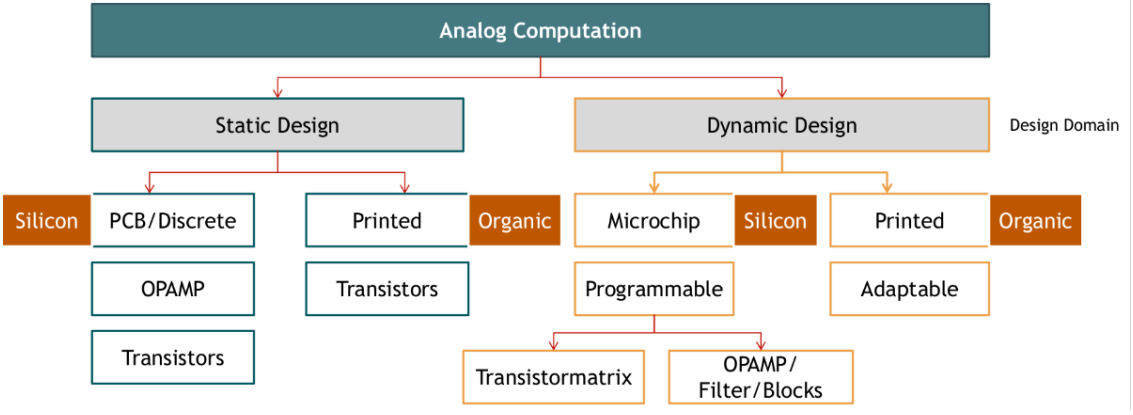


Abb. 2. Bei den Technologien unterscheiden wir statische und dynamische Schaltungen. Die KNN Modelle sind i.A. statisch aber parametrisierbar.

Künstliche Neuronale Netzwerke

[Bosse, 2025, SysInt Conference]

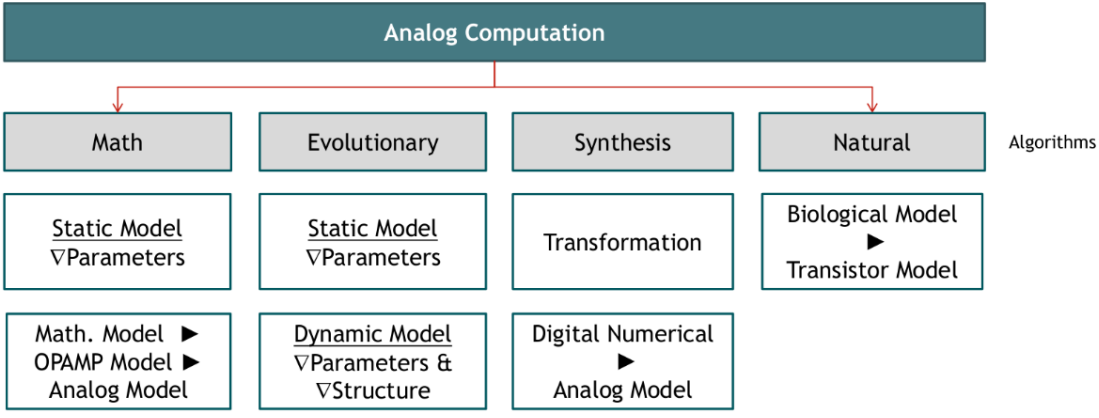


Abb. 3. Algorithmisch können wir verschiedene Verfahren verwenden um Analogschaltungen und Analogrechner zu entwerfen.

Künstliche Neuronale Netzwerke

Im Prinzip ein universeller nichtlinearer Funktionsapproximator aus Daten.

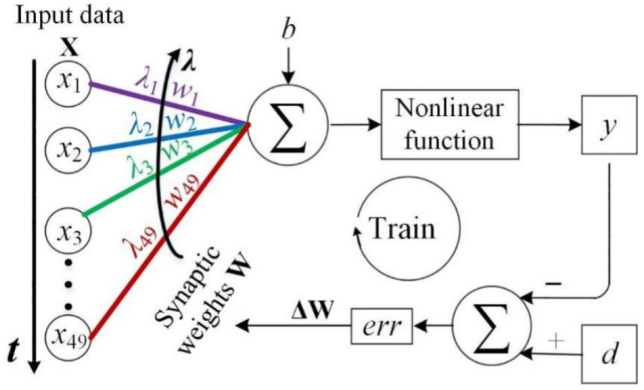
$$\mathrm{KNN}(\vec{x}, D) : \vec{x} \rightarrow \vec{y} \approx f(\vec{x}) : \vec{x} \rightarrow \vec{y}$$

$$\mathrm{KNN}\left(\hat{k}, \vec{b}\right) = g_1(g_2(\ldots(g_n(\vec{x}))))$$

$$g_i\left(\vec{u}_i, \vec{k}_i, b_i\right) = a\left(\sum_{j=1}^{|u|} u_{i,j} k_{i,j} + b_i\right)$$

Künstliche Neuronale Netzwerke

Der Funktionsapproximator - das KNN Modell - hat eine feste Struktur (d.h. auch die Analogschaltung ist fix in ihrem Aufbau), ist aber parametrisierbar. Die Parameter werden bestimmt (iterativ angepasst) indem der Fehler des Modells minimiert wird: Maschinelles Lernen aus Daten. Die Daten können experimentell sein oder durch eine anzupassende Funktion gegeben sein.



Das Analoge Künstliche Neuron

- Das mathematische Modell soll auf eine analoge Schaltung abgebildet werden.
- Wir benötigen:
  - Einen Verstärker mit Produktsummenfunktion (also invertierender Verstärker / Addierer)
  - Eine nichtlineare Aktivierungsfunktion (typisch die auf der Exponentialfunktion basierende sigmoid Funktion)

Die Neuronfunktion kann mit positiven und negativen Gewichten (Parametern) besetzt sein. Bei einem invertierenden Addierer sind die Gewichte aber immer positiv (bzw. durch die Negierung am Ausgang äquivalent negativ).

Das Analoge Künstliche Neuron

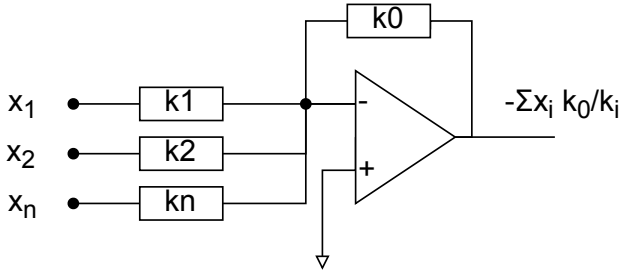


Abb. 5. Die Produktsummenberechnung mit invertierenden Addierer. Die Gewichte sind jeweils Verstärkungsfaktoren $k_i=R_0/R_{ki}$

Das Analoge Künstliche Neuron

- Wir müssen also einen Differenzverstärker (Subtrahierer) verwenden um sowohl positive wir auch negative Gewichte nutzbar zu machen.
- Für die negativen Gewichte kann man die Widerstände unabhängig berechnen
- Für die positiven Gewichte wird es komplizierter und mit Seiteneffekt auf die negativen (Gesamtverstärkungsfaktor ändert sich)

Das Analoge Künstliche Neuron

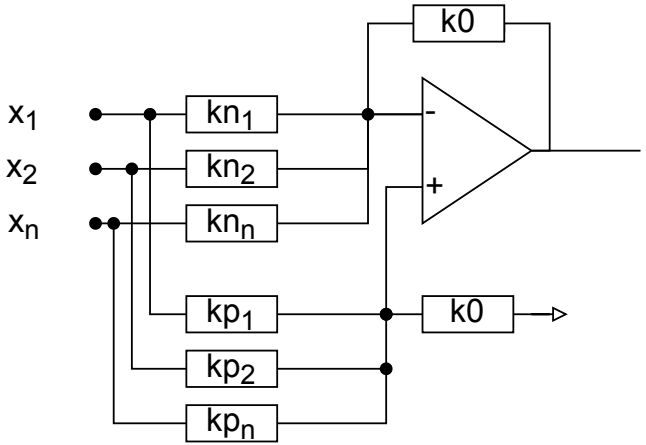


Abb. 6. Subtrahierer für Produktsummenberechnung

Das Analoge Künstliche Neuron

Daher müssen wir auf den Instrumentenverstärker zurückgreifen der aus zwei invertierenden Addierern und einem (festen) Subtrahierer besteht. Hier können wir die positiven und negativen Gewichte unabhängig einstellen. Wichtig: Ein Gewicht *k* ist entweder an den negativen oder den positiven Zweig anzuschliessen!!

Das Analoge Künstliche Neuron

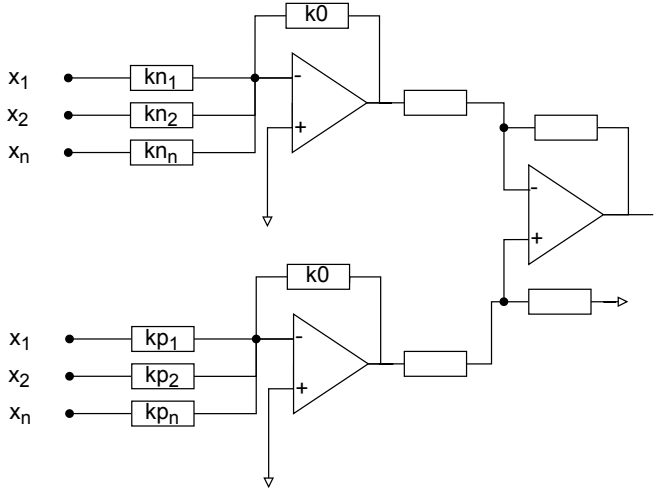


Abb. 7. Instrumentenverstärker für Produktsummen Berechnung

Das Analoge Künstliche Neuron

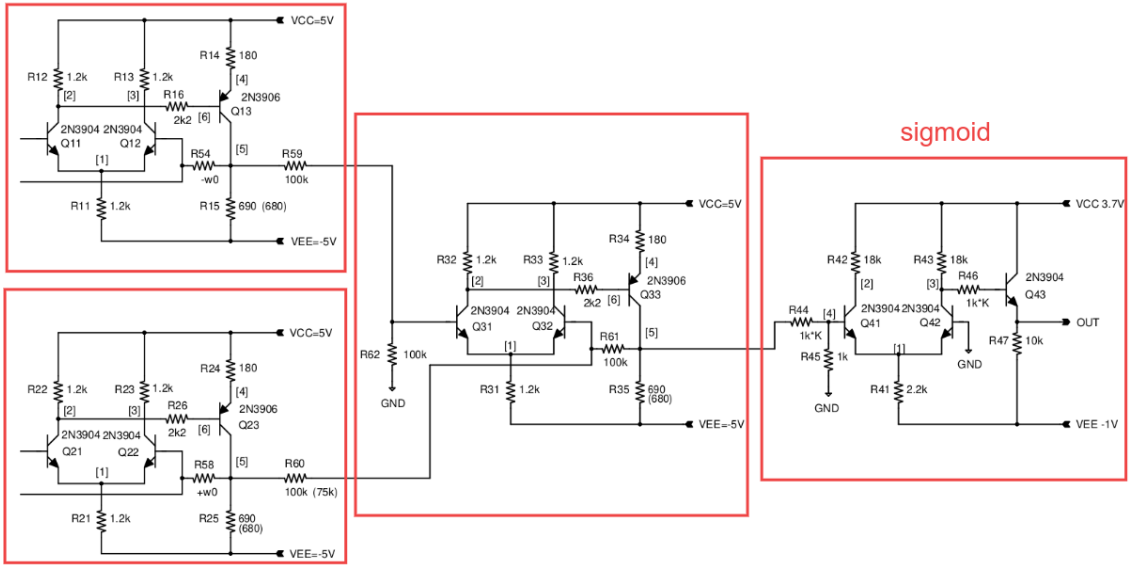


Abb. 8. Die analoge Implementierung eines Neurons eines AKNN mit drei OpAmp3 Stufen und einer weitern Schaltung für die sigmoid Aktivierungsfunktion.

Numerisches Training eines KNN

Ausgangslage:

1. Eine Datentabelle mit Beispielen in der Form $\boldsymbol{D}=(\mathbf{x},\mathbf{y})$
   - Beispiel: IRIS Klassifikationsdatensatz mit vier numerischen Eingangangsvariablen und drei Klassen (drei Ausgangswerte für drei Klassen)
2. Ein KNN mit 3 + 3 Neuronen, als rein numerisches funktionales Modell implementiert.
3. Training durch Rückpropagation des Fehlergradienten

Aber halt: Wir erhalten hiermit z.B. Gewichtungsparameter mit Werten über 100. Unser OpAmp3 hat aber nur einer Leerlaufverstärkung von ungefähr 100. Und numerische Werte als Ergebnis einer Produktsummeneberechnung können größer 10 sein! Die Kennlinie unseres OpAmp3 hat nur einen Aussteurungsbereich (deutlich) kleiner 10 V. Und die sigmoid Schaltung hat den Ausgangsspannungsbereich [0,3V]!

Das Funktionsnetzwerk

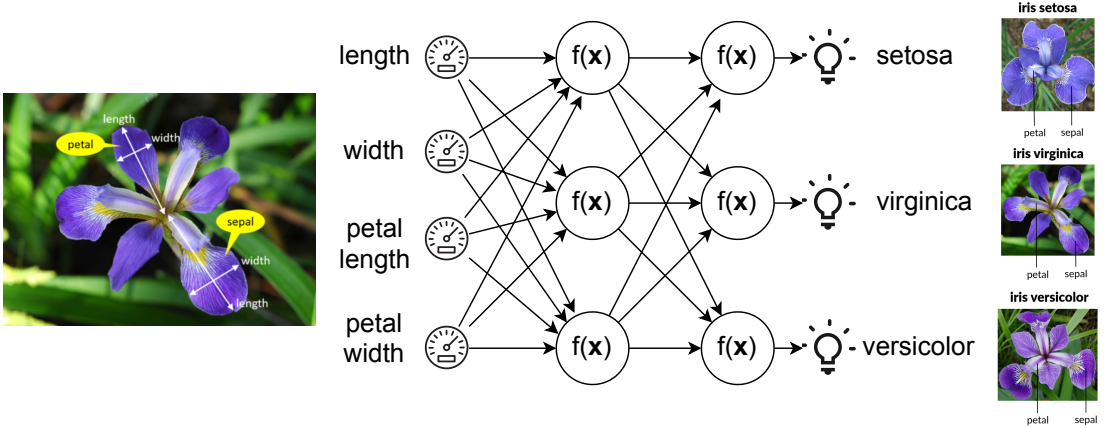
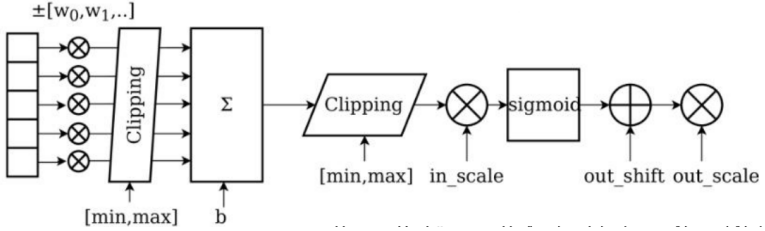


Abb. 9. Das Metamodell eines AANN (gerichteter Funktionsgraph) zur Klassifikation von Pflanzen anhand metrischer Parameter

Numerisches Training eines KNN

Wir müssen das numerische Modell verbessern und wenigstens auf die Eigenschaften des idealen Operationsverstärkers begrenzen:

1. Begrenzung der Werte der Produktsummenberechnung auf [-10,+10] (oder kleiner).
2. Begrenzung der möglichen Gewichte auf [0,10].
3. Skalierung der sigmoid Funktion auf Ausgangswerte von [0,3] und eine Skalierung der Eingangswerte mit einem konstanten Faktor.



Übertragung auf AKNN

- Die Parameterwerte aus dem numerischen Training mit dem modifizierten KNN Modell müssen in Widerstandswerte umgerechnet werden.
  - Theoretisch mit dem mathematischen OpAmp Modell, aber durch Nichtlinearität der OpAmp3 Schaltung mit Korrekturfunktionen.
- Eingangsvariablen sind jetzt skalierte Spannungen im Bereich [0,1V].
- Ausgangsspannung ist pro Klassifikationsneuron [0,3V].
- Test des AKNN durch digitales Schnittstelle mit AD und DA Wandlern vom Computer aus (oder exemplarisch unabhängig mit einstellbaren Spannungsquellen).
  - Funktioniert schon! Klassifikationsfehler < 20% (für alle 151 Beispielsinstanzen)

Übertragung auf AKNN

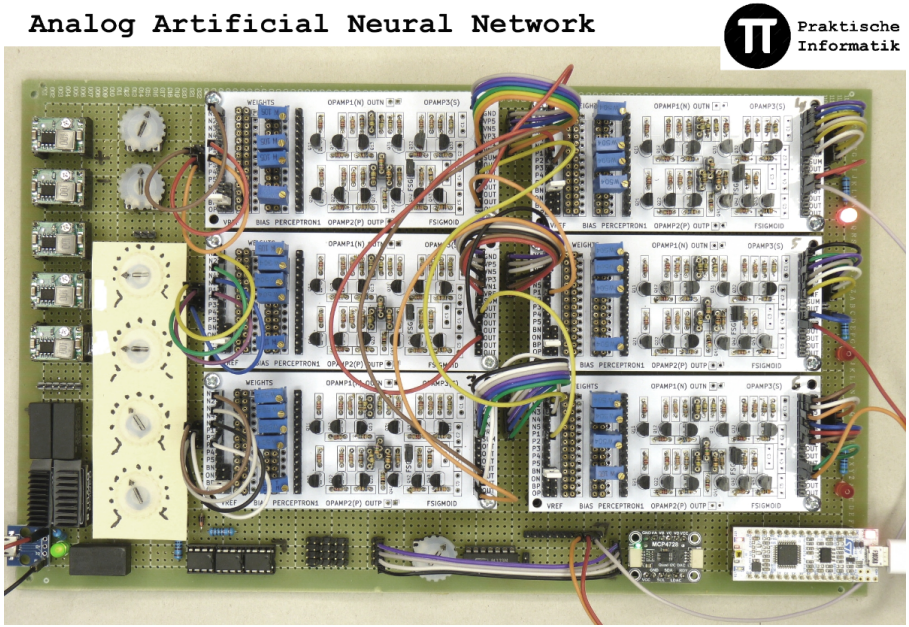


Abb. 10. Einstellung der Parameter mit einstellbaren Widerständen. Sehr mühsam, nur proof of concept!

Digitale Zwillinge

- Bisher wurde ein doch sehr grobes numerisches Modell der Analogschaltung eines Neurons verwendet.
  - Bei Verwendung technisch guter OpAmps (nahe dem idealen Operationsverstärker) wären schon fertig.
- Bei diesem sehr einfachen OpAmp3 sind die Abweichungen vom idealen Operationsverstärker aber zu groß.
- Die Abweichungen müssen erfasst werden durch:
  1. Reale Messungen
  2. Simulation
  3. Ein Ersatzmodell (aus Simulation oder realen Messungen) dass die Übertragungsfunktion eines Neurons approximiert (aber numerisch schnell berechenbar ist) ⇒ Wieder ein KNN!!!

Das Bipolare Modell

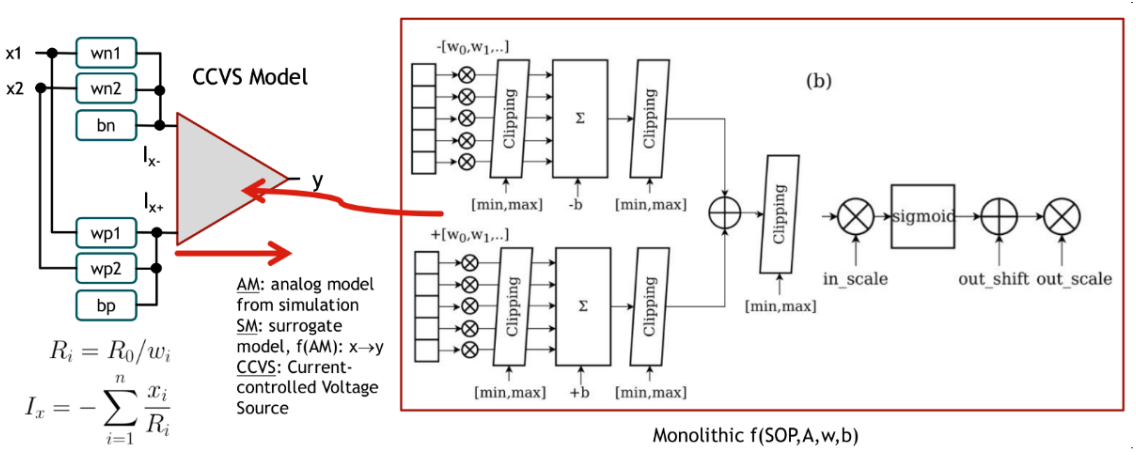


Abb. 11. Verbessertes Bipolares numerisches Modell (hier $U_a(I_x)$!!!, CCVS Modell) - Aufteilung der Eingänge, aber parallele Anschluss einer Eingangsvariable an pos. und neg. Zweig.

Training Reloaded

Bisher wurde eine analytische Funktion für die Neuronfunktion verwendet. Das Training benötigt den Gradienten dieser Funktion. Bei der analytischen Funktion einfach. Bei irgendeiner Funktion oder einem Ersatzmodell müssen wir auf numerische Differenzierung umsteigen (hey, kann analog sein!!)

$$w_i = w_i - \alpha e(f,x,y)\frac{\Delta f(x)}{\Delta w_i}, w_i = w_i - \alpha \sum_{j=1}^{batchsize} e(f,x_j,y_j)\frac{\Delta f(x_j)}{\Delta w_i}$$

w: weight parameter, e: backpropagated error, α: update rate

Abb. 12. Numerische Differenzierung des Neuronfunktion *f* und Anpassung der Gewichte über den aktuellen Fehler *e* des Modells.

Training Reloaded

Die Anpassung der Gewichte durch den Fehlergradienten funktioniert dann stabil und zielführend wenn die Funktion *f* und der Gradient der Funktion streng monoton sind. Das Ersatzmodell des digitalen Zwillings zeigt ebenso wie die analoge Schaltungen aber an den Rändern eine Umkehr der Steigung. Das führt zu hoher Instabilität des numerischen Trainings.

- Noch eine wichtige Randbedingung: Die Parameter des Modells werden üblicherweise mit einer stochastischen Verteilungsfunktion (also randomisiert) initialisiert um eine Startbedingugn für dasd Minimierungsproblem unabhängig von den Daten zu erzielen.
- Beobachtung bei nicht monotoner Funktion *f*: Hin und wieder konvergeiert das Training auf gute Ergebnisse, oft divergiert es.
    - Aber wir können Brute Force machen: Das Training so lange neu starten (und randomisiert die Parameter initialisieren) bis eine gute Lösung gefunden wurde (dann auch häufig mit schneller Konvergenz)

Training Reloaded

- Brute Force kann jeder. Geht es besser?
- Wir versuchten eine Pre-Training des Modells mit Simmulierter Abkühlung:
  - Der Paramersatz wird hier randomisiert verändert bis eine bessere Lösung (also kleinerer fehler) gefunden wird (iterativ)
  - Die randomosoerte Anteil wird ovn Iteration zu Iteration verringert (Stabiliserung, Abkühlung)
  - Nicht befangen durch nicht monotone Funktionen (hier vollständies Blackbox Modell)

Training Reloaded

```
function SA(data, params, num_steps=1000, noise=0.01, cooling=0.999) {
  initial_params=optimal_params=best_params=params; temp=1.0
  new_loss = best_loss = loss(data,params)
  for(i=1,num_steps) {
    temp=temp*cooling
    new_params = params.map(p => p+gaussianRandom(0, noise))
    new_loss = loss(data,new_params)
    if (new_loss < best_loss || random()*temp > best_loss/new_loss) {
      params = new_params
      if (new_loss < best_loss) { optimal_params = params; best_loss = new_loss }
    }
  }
}
```

Alg. 1. Simmulierte Abkühlung zur Lösung von Minimierungsproblemen)#

Workflow

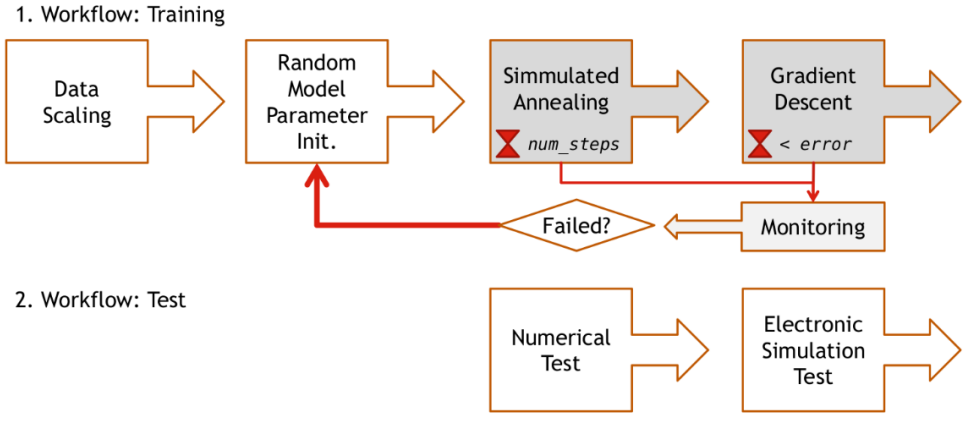


Abb. 13. Die Arbeitsabläufe als Hybrider Ansatz

Der Workflow

[Bosse, Proc. of the 7th International Conference on System-Integrated Intelligence (SysInt 2025)]

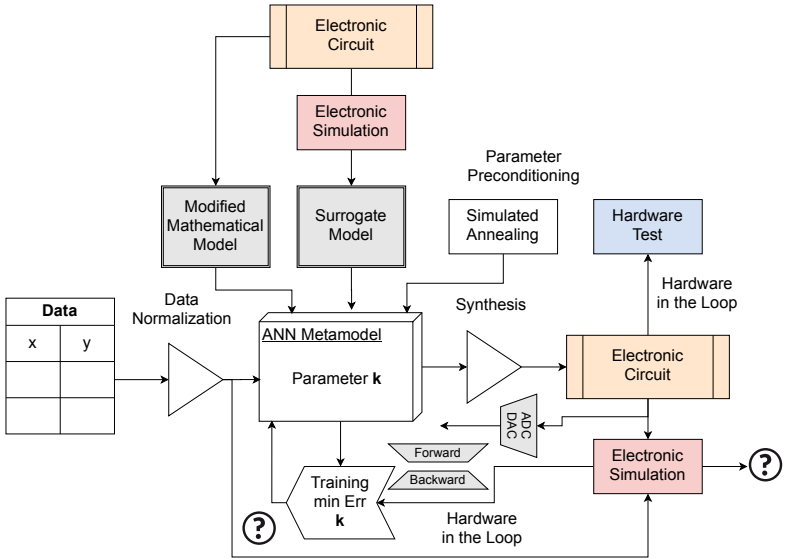
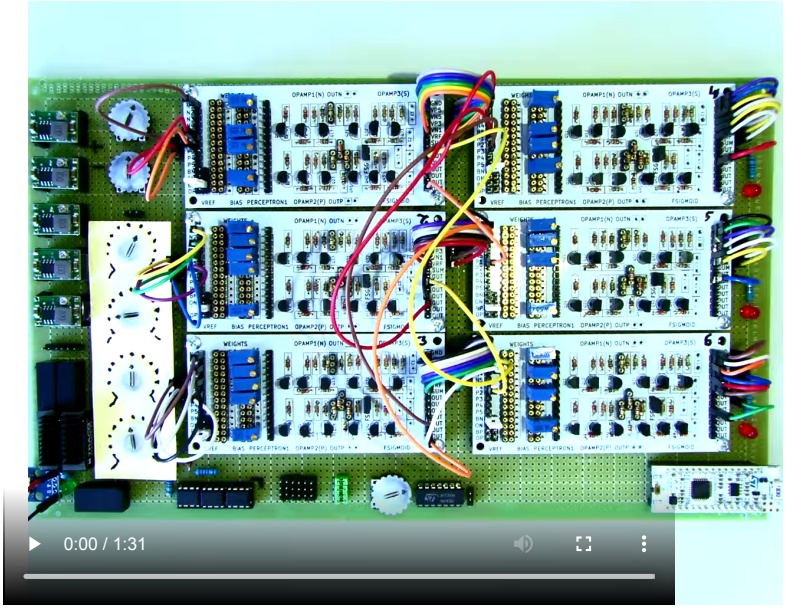


Abb. 14. Die Details der Einbindung des digitalen Zwillings (eines Teils) der Elektronikschaltung

Live Demo: Der Demonstrator



Die Ergebnisse

1. Verwendung des einfachen numerischen Modells mit Parameter- und Wertbegrenzung:
   - Schaltung funktioniert mit Korrekturfunktionen für die Berechnung der Parameterwiderstände, Klassifikationsfehler < 30%
   - Training: stabil konvergierend (obwohl Unstetigkeit durch Clipping!)
2. Verwendung des elektronischen Ersatzmodells (Stromeingang, Spannungsausgang) aus Simulation:
   - Schaltung funktioniert, keine Korrektur der Parameterwiderstände erforderlich
   - Klassifikationsfehler < 20%
   - Training: instabil. nur selten konvergierend (Nichtmonotonie des Ersatzmodells), Pre-training mit SA brachte keine Verbersserung
3. Kopplung des Analogrechners mit Computer über AD und DA Schnittstellen (12 Bit Auflösung)
   - Der Klassifikationsfehler hängt von der Reihenfolge der getesteten Dateninstanzen ab (151)
   - Grund: Thermische Drift durch unetrschiedliche Aktivierung der Elektronikkomponenten!

Die Ergebnisse

4. Die Verwendung von bipolaren NPN und PNP Transistoren (stromgesteuerte Stromquelle) führt zu hoher elektrischer Ruheleistung und Erwärmung (ca. 15 W bei 6 Neuronen, siehe auch 3.)

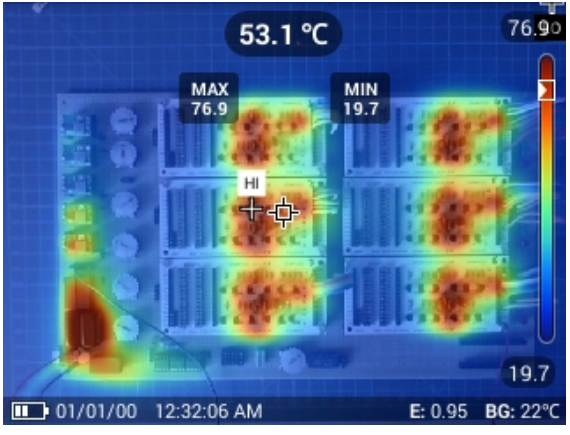


Abb. 15. Thermografiebild der Schaltung nach ca. 10 Minuten Betrieb