# Dokumentation

# Künstliche Intelligenz: Ziffernerkennung

| | |
|---|---|
| Aufgabenstellung: | Zahlenerkennungsprogramm |
| Betriebssystem: | DryOS, VxWorks |
| Programmiersprache: | C |
| Verwendeter Editor: | Notepad++/Notepad2 |
| Verwendete Software: | JavaNNS |
| Datum: | 28.06.2009 |

Bearbeitet von:
Christian Kettmann (Kettmeister)
Andreas Kahlenbach (CHDKLover)

## Inhaltsverzeichnis

1 Aufgabenstellung ..... 3
2 Technische Rahmenbedingungen ..... 3
3 Benutzerdokumentation ..... 3
3.1 Vorbereitung ..... 3
3.2 Programmstart ..... 3
3.3 mögliche Fehler, die auftreten können ..... 4
4 Entwicklerdokumentation ..... 4
4.1 Aufbau des Neuronalen Netzes ..... 5
4.2 Erstellen der Inputdaten ..... 6
4.3 Sammeln von Trainingsdaten ..... 7
4.4 Lernvorgang mit Hilfe des JavaNNS ..... 7
5 Auswertung ..... 7
5.1 Verbesserungen ..... 7
5.2 Bewertung ..... 8

# 1 Aufgabenstellung

Realisierung eines Systems, welches auf der Basis eines neuronalen Netzes handgeschriebene Ziffern erkennt.

# 2 Technische Rahmenbedingungen

Es wird eine Canon Kompakt Kamera (IXUS oder Powershot) benötig, für die CHDK (Canon Hacker's Development Kit) portiert ist. Hier wird davon ausgegangen, dass dies der fall ist und wird deswegen nicht näher erläutert. (Informationen zur Installation: www.wirklemms.de/chdk/forum/viewtopic.php?t=3)
Zur Entwicklung des CHDK- Buildes, wurde Notepad++ und Notepad2 verwendet, zur Kompilierung ein Gnucompiler für Windows (gcc) und zum Trainieren des Netzes JavaNNS (http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/)

# 3 Benutzerdokumentation


1: Löschen-Taste
2: Print-Taste
3: Disp.-Taste
4: Menu-Taste
5: Set/Funk.-Taste

## *3.1 Vorbereitung*

Es wird davon ausgegangen, dass die Kamera mit dem benötigten CHDK-Build startet und richtig konfiguriert ist.
Zusätzlich muss das trainierte Netz in Form einer nr.net Datei auf die Speicherkarte in den Ordner */CHDK/Data/* kopiert werden.

## *3.2 Programmstart*

Gestartet wird die Kamera im Aufnahmemodus. Um das Zahlenerkennungsprogramm starten zu können, muss das Alternativmenü aufgerufen werden. Das macht man, in dem man zuerst die „Print-Taste“ drückt und anschließend die „Menu-Taste“. Unter Folgenden Menupunkt befindet sich das Zahlenerkennungsprogramm.

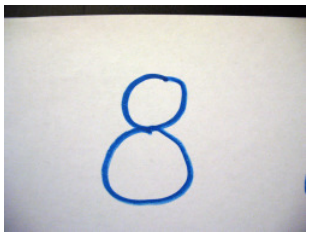
```
Verschiedene Einstellungen
    → Spiele
        → NrDetection
```

Mit den Coursertasten kann sich im Menu bewegt und mit „SET“ ein Programm gestartet werden.

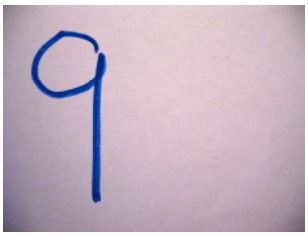
Jetzt muss die Kamera über die geschrieben Zahl gehalten und auf „SET“ gedrückt werden. Die Kamera Zeigt jetzt die Zahl an, die sie geschrieben haben und die Wahrscheinlichkeit, wie sich die Kamera bei Ihrer Entscheidung sicher ist.
Mit der „Print-Taste“ kann das Programm wieder verlassen werden.

### *3.3 mögliche Fehler, die auftreten können*

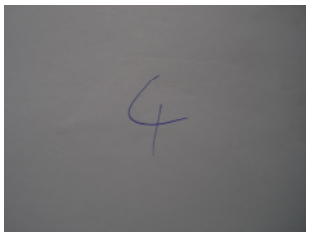
Zahl wird nicht richtig ausgeschnitten:



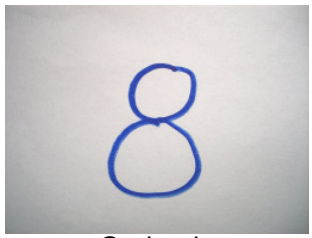
hochizontal sind störende Bildbereiche im Bild



Zahl nicht in der Mitte



schlechte Lichtverhältnisse/ Zahl zu klein/ Strich zu dünn



Optimal

Kamera stürzt nach starten des Programms ab (seltener Fehler)

Dieser Fehler tritt auf, wenn die Kamera nicht mehr genügend freien Arbeitsspeicher zur Verfügung hat. In diesem Fall muss man die Stromversorgung der Kamera unterbrechen, indem man den Deckel des Batterieschachtes öffnet.
Um Dieses Problem zu beheben ist es nötig, Teile des CHDK's nicht mit zu kompilieren. Dies ist für Laien sehr schwer, deswegen sollte man in diesem Fall sich an die Entwickler wenden.

# 4 Entwicklerdokumentation

Im Folgenden werden grundlegende Informationen gegeben, die Helfen sollen das Projekt zu verstehen oder zu erweitern.

Gut zu wissen:

- Das Farbmodell der Kamera ist wie folgt aufgebaut.
  `vy1uy2y3y4`
  Y1, Y2, Y3 und Y4 stehen jeweils für Grauwerte. V und U stehen für die Farbe. Es werden jedoch nur die Grauwerte verarbeitet.

- Die Funktion `vid_get_viewport_live_fb()` gibt ein Pointer auf den Liveview-Framebuffer (LVFB) zurück.

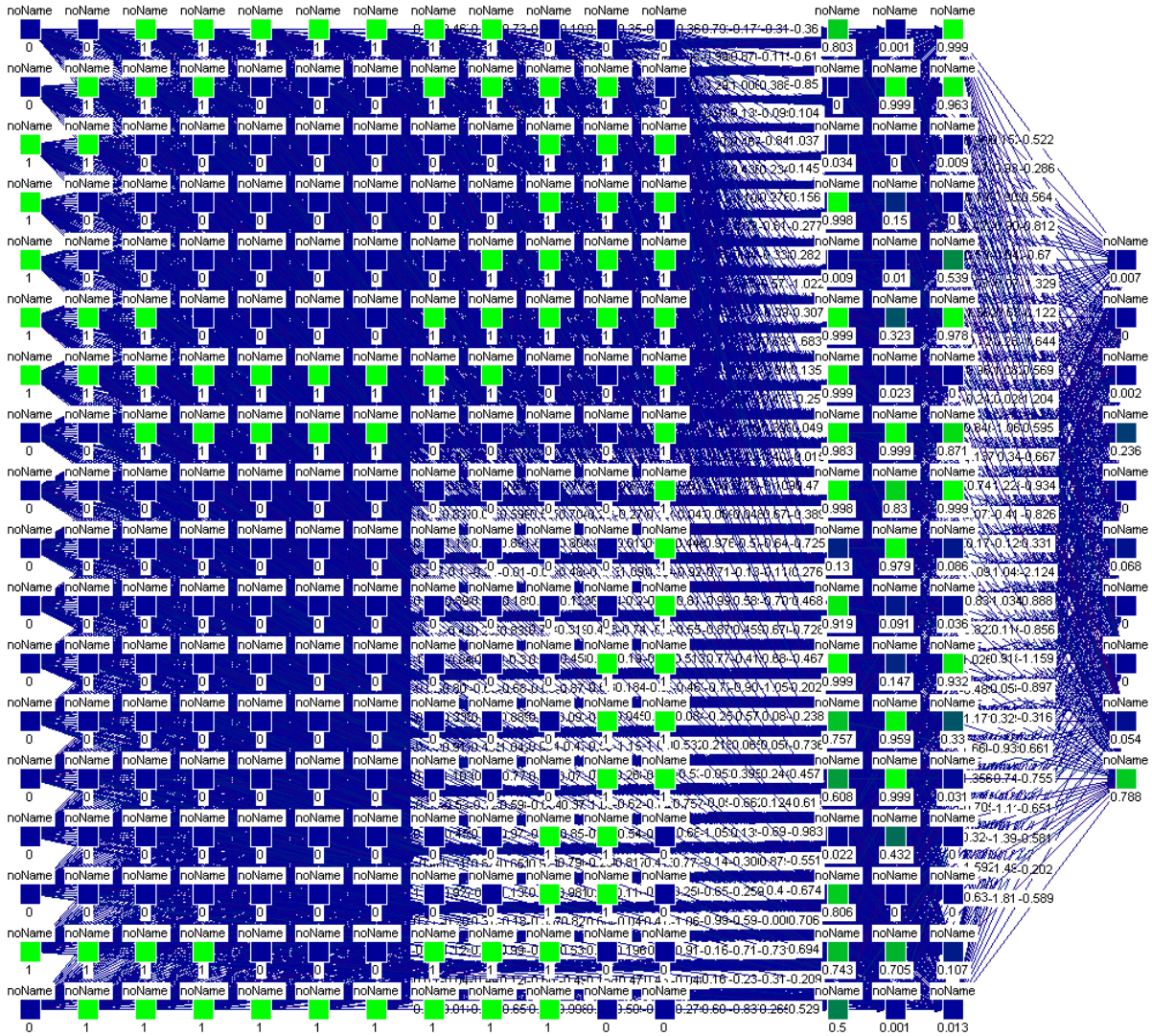
- `gui_nrdetection_kbd_process()` ist eine Funktion, die beim drücken einer Taste ausgeführt wird

- `gui_nrdetection_init()` wird beim Start des Programms ausgeführt (ist also eine Art Konstruktor).

- Die Teile der C-Standartbibliothek können verwendet werden.

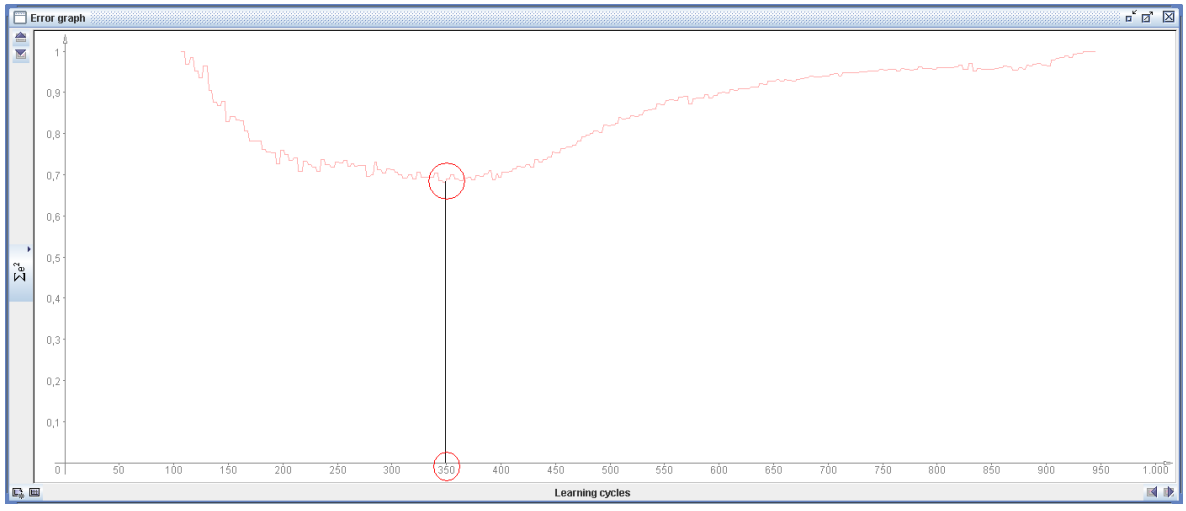
- Die Speicherplatz sparenden Datentypen wurden auf Grund sehr beschränkten Speicherplatzes verwendet

## 4.1 Aufbau des Neuronalen Netzes



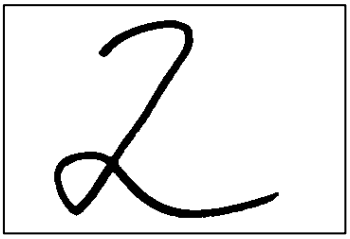
| Input: | Hidden | Output |
|---|---|---|
| 12x18 Act_Identity | 3x18 Act_Logistic | 1x10 Act_Logistic |

Verwendet wurde ein Multilayerperzeptron mit einer Eingangsmatrix von 12x18 Inputneuronen, einer Hiddenschicht von 3x18 Hiddenneuronen und einer Ausgangsschicht von 1x10 Outputneuronen. Dieses Netz wurde, in diesem Beispiel, mit reichlich 1000 Trainingsdate in 350 Zyklen trainiert, weil bei 350 Die Validierungskurve am niedrigsten ist.




Errorgraf eines Beispieltrainings (bei mehr als 350 wird das Netz übertrainiert)

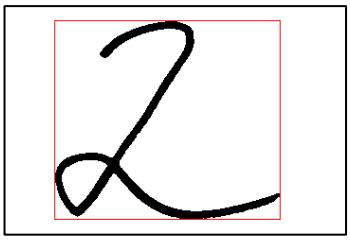
## 4.2 Erstellen der Inputdaten



Entstandenes S/W Bild

Zuerst wird der Pointer auf den LVFB in die Variable Img (vom Typ char*) gespeichert. Anschließend muss das Bild in ein S/W-Bild umgewandelt werden. Hierzu bildet man, von dem ganzen Bild, den Durchschnittsgrauwert und verwendet diesen als Schwellwerte zur Schwarz-Weis-Trennung. Nun wird der komplette LVFB durchlaufen und jeder Grauwert über dem Schwellwert wird zu Weis und unter dem Schwellwert wird zu Schwarz. Ein weiterer Ansatz, der implementiert wurde funktioniert wie folgt. Wieder wird der Puffer durchlaufen. Dieses Mal werden aber immer 4 aufeinander folgende Pixel betrachtet und von denen der Durchschnittswert gebildet. Dabei muss sich der min. und max. Wert gemerkt werden. Der Durchschnittswert des max. und min. Wertes wird dann als Schwellwert verwendet. Durch das ein- und auskommentieren der Funktion `lv2sw_andy()` und `lv2sw_christian()` kann zwischen beiden Methoden gewählt werden.

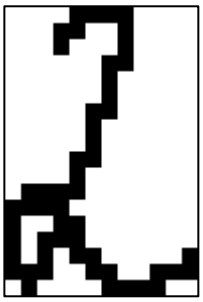
---



Ausgescnittene Zahl

Als nächster Schritt, wird die Zahl ausgeschnitten. Dazu wird in der Mitte des Bildes begonnen nach außen hochizontal und vertikal nach schwarzen Pixel zu suchen. Wird keins gefunden, werden sich die Eckpunkte des umrandenden Vierecks gemerkt.

---



Fertig skaliertes Bild

Zuletzt wird die Ausgeschnittene Zahl auf eine Größe von 12x18 Pixel skaliert

Nun kann das binäre 2Dimensionale Feld in das Neuronale Netz geschickt werden.

## 4.3 Sammeln von Trainingsdaten

```
SNNS pattern definition file V3.2
generated at Wed Aug  9 11:01:29 1995

No. of patterns : 572
No. of input units : 216
No. of output units : 10

# Input (1) A/CHDK/DATA/NR_3_0001.bmp
 0 1 1 1 1 1 1 1 1 1 0 0
 1 1 1 0 0 0 0 0 1 1 1 1 0
 1 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 1 1 1 1 0
 0 0 0 0 1 1 1 1 1 1 0 0 0
 0 0 0 0 1 1 1 1 1 1 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 1 1 1 0
 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 1 1 1 1 1 1 1 1 0 0
 0 1 1 1 1 1 1 1 0 0 0 0 0
# Output (1) A/CHDK/DATA/NR_3_0001.bmp
 0 0 0 1 0 0 0 0 0 0

...
```

Trainingsdatei für SNNS / JavaNNS (*.pat)

Unter den in 4.2 geschaffenen Vorraussetzungen, ist es recht einfach, schnell viele Trainingsdaten zu sammeln. Hat man einmal das 12x18 große binäre Array, kann man es, nach definierter Formatierung in eine pat-Datei schreiben.

In diesem Projekt ist eine Funktion `appentToPat` implementiert, die beim drücken der „Set-Taste“ ein Trainingsdatum an das Ende der pat-Datei schreibt. Liegt im Verzeichnis „A/CHDK/DATA/“ keine Datei mit dem Namen nr.pat so wird sie angelegt, ansonsten wird an das Ende der Datei geschrieben. Um diese Funktion, zum sammeln von Trainingsdaten verwenden zu können, muss das Projekt mit dem Parameter `#define DUMP_MODE`, der standardmäßig auskommentiert ist, kompiliert werden.

Es wurden, für das im Projekt verwendete neuronale Netz, über 1000 Trainingsdaten gesammelt, in dem Filzstifte und weiße A4 Blätter an Studenten verteilt wurden und jeder ca. 50-70 Ziffern(0 bis 9) geschrieben hat. Anschließend wurde mit 2 Kameras und der eben beschriebenen Methode, die Trainingsdatei erstellt.

## 4.4 Lernvorgang mit Hilfe des JavaNNS

Nach Erstellung des Netzwerkes mit den oben genannten Neuronenzahlen und laden der Pattern Dateien (.net), wurde das Netzwerk mit zufälligen Gewichen initialisiert. Diese wurde über den überwachten Lernprozess nach dem Backpropagation Algorithmus immer weiter auf die Eingangspattern spezialisiert. Zu einem geeignetem Zeitpunkt wurde der Lernvorgang abgebrochen und die Gewichte in eine Networkdatei (.net) exportiert (Datei → speichern unter).

# 5 Auswertung

Das Projekt zeigt nur Exemplarisch die Möglichkeiten des CHDK’s. Ähnliche Projekte, zur Erkennung von anderen klassifizierbaren Formen, sind in ähnlicher weise möglich.

## 5.1 Verbesserungen

Um das Projekt zu verbessern, gibt es ein paar Sachen, die man probieren könnte.

a) Man könnte mehr Trainingsdaten Sammeln. (Wichtig ist, dass keine fehlerhaften Trainingsdaten in der *.pat stehen)
b) Man könnte mit der Anzahl der Hiddenneuronen / Hiddenschichten variieren und dabei beobachten, bei welcher Zahl von Hiddenneuronen der Errorgraph am niedrigsten geht.
c) Man könnte die Anzahl der Eingangsneuronen erhöhen. z.B. auf 20x30 oder noch höher.

Ein weiterer Punkt, der verbesserungswürdig ist, ist das Umwandeln des LVFB in einen S/W-Puffer. Ein Algorithmus, der trotz der oben erläuterten Fehler die auftreten können, es schafft, die Zahl ordentlich auszuschneiden und zu skalieren, würde die Handhabung des Programms erleichtern.

## *5.2 Bewertung*

Nach anfänglichen Unklarheiten und fehlenden Ansätzen, hat sich die Aufgabe als nicht übermäßig schwer herausgestellt. Es war ein besonderer Anreiz, dieses Projekt auf der Kamera zu implementieren. Dadurch entstand der Vorteil, dass es leicht war, viele Trainingsdaten zu sammeln um das Netz gut zu trainieren.
Das Programm erkennt ca. 95% der Zahlen und die Performance ist akzeptabel und für Kameras mit DigiC IV-Prozessoren sogar sehr gut.
Es war für uns eine sehr lehrreiche Aufgabe, die uns Einblick in einen Teil der Künstlichen Intelligenz gewährte. Wir können uns gut vorstellen andere Aufgaben mit der KI zu lösen.