Kann SAS Ihre Handschrift lesen? Machine Learning am Beispiel von Stacked Denoising Autoencoders

Gerhard Svolba
SAS Austria
Mariahilfer Straße 116
A-1070 Wien
Sastools.by.gerhard@gmx.net

Zusammenfassung

Dieser Beitrag zeigt, wie die Aufgabe der automatischen Erkennung der handschriftlichen Ziffern 0-9 mit Hilfe von Neuronalen Netzen bewerkstelligt werden kann. Anhand eines Beispiels wird end-to-end beginnend bei den Quellendaten, über die analytischen Methoden, bis hin zur Präsentation der Ergebnisse gezeigt, wie diese Aufgabe mit neuronalen Netzwerken in SAS spezifiziert und gelöst werden kann. Als Basisdaten für die Analyse wurden frei verfügbare Daten des Mixed National Institute of Standards and Technologies (MNIST) verwendet.

Im Beitrag wird das Problem aus einer Machine Learning Perspektive betrachtet. Es wird praktisch erklärt, wie ein sog. "Stacked Denoising Autoencoder" aufgebaut ist und welche Eigenschaften dieser hat. Weiters wird illustriert, was sich hinter den Begriffen "Deep Learning", "Semi-Supervised machine learning", oder "Feature Extraktion" verbirgt. Der SAS Code für die Lösung dieser Aufgabe wird gezeigt und ein kurzer Überblick gegeben, welche Machine Learning Funktionalität in welchen SAS Werkzeugen verfügbar ist.

Schlüsselwörter: Machine Learning, Data Mining, Deep Learning, Convolutional Neural Network, Autoencoder, Hidden Layer, Pattern Recognition, Image Mining

1 Einleitung und Problemstellung

Die Erkennung von Bildern, insbesondere die Extraktion von Features und Konzepten in Bildern, ist ein immer wichtigeres Thema in der Datenanalyse. Dabei kommen Deep Learning Methoden und Convolutional Neural Networks zum Einsatz. Ziel ist die Verdichtung der Information aus den Bilddaten zu wenigen Merkmalen, welche die Bildinhalten suffizient beschreiben und für die weitere Analyse verwendet werden können.

In diesem Beispiel wird PROC NEURAL des SAS Enterprise Miner verwendet, um einen Stacked Denoising Autoencoder für die Ziffern Daten des Mixed National Institute of Standards und Technologie (MNIST) zu bauen. Dabei wird ein mehrstufiges neuronales Netzwerk gebaut. Merkmale aus der mittleren Schicht des Autoencoders werden unter Verwendung der Inputdaten und des PROC NEURAL-Score-Codes extrahiert. Die so extrahierten Features haben einen hohen Vorhersagewert. Die Qualität der extrahierten Features wird mit der 1. und 2. Hauptkomponente verglichen.

2 Fragestellung und Analysedaten

2.1 Analysedaten

Die MNIST Daten (Mixed National Institute of Standards und Technologie) sind ein berühmter Datensatz von handschriftlichen Ziffern in Form von Bilddaten. Diese werden sehr häufig in Machine Learning Forschungsprojekten für das Training korrekter Klassifikationen verwendet.

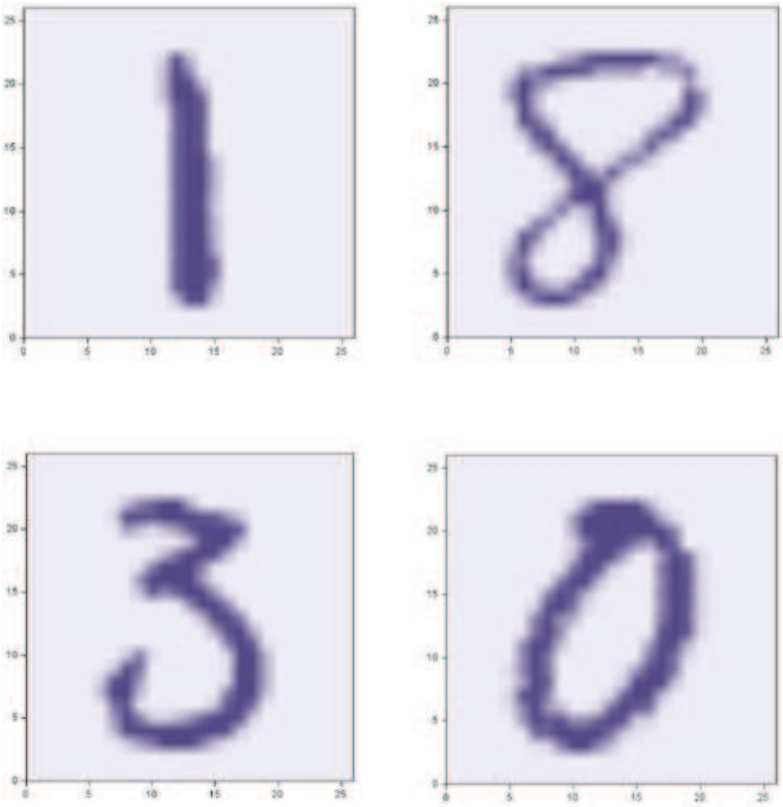

Abbildung 1: Darstellung der Daten in einem 28x28 Grid

Die Daten wurden für die Analyse vorab zentriert und Pixels auf den äußeren Rändern, die alle blank sind, wurden entfernt. Die verbleibenden Variablen wurden für die Analyse verwendet.

2.2 Darstellung der Daten in einer Datenmatrix

In den Daten sind etwa 60.000 Bilder enthalten, zu denen auch die Zielvariable „Ziffernwert“ verfügbar ist. Die Datenstruktur sieht somit wie folgt aus.

- 28 mal 28 (= 784) Inputvariablen mit den Werten 0-255. D.h. jedes Pixel in der 28x28 Matrix wird durch seine Farbtiefe dargestellt.
- Eine Ziel-Variable mit dem Ziffernwert.
- Eine ID Variable für das Bild.

Siehe dazu auch Abbildung 2.

digit	pix1	pix2	...	pix784	TARGET (LABEL)
1	0	8	...	0	4
2	0	3	...	0	3
3	244	1	...	0	2
4	78	3	...	3	7
5	0	0	...	4	8
...	...	...	...	...	...
...	...	...	...	...	...
42000	3	0	...	...	9

Abbildung 2: Darstellung der Daten in einer Datentabelle

3 Konstruktion eines Autoencoders

3.1 Stacked Denoising Autoencoder

Die Analyse der Daten erfolgt nun nicht im klassischen Sinn, indem man ein Predictive Model mit 784 Input Variablen für die Zielvariable „Ziffernwert“ 0-9 erstellt. Es wird ein mehrstufiges Neuronales Netzwerk trainiert, welches sowohl die 784 Variablen sowohl als Input als auch als Output Daten beinhält.

Dieses Konstrukt wird auch als „Autoencoder“ bezeichnet. Genauer gesagt ein stacked Autoencoder, weil das neuronale Netzwerk mehrstufig ist. Die Inputvariablen werden somit durch mehrere Stufen verwendet, um die gleiche Anzahl von Output Variablen wieder vorherzusagen.

Zusätzlich werden die Inputdaten in zufälliger Weise „verunreinigt“, d.h. manche Werte der 784 Input Variablen werden zufällig durch andere Werte ersetzt. Der Autoencoder hat somit den Auftrag, diese Verunreinigung wieder zu bereinigen (Denoising). Dieser Schritt wird durchgeführt, um die Stabilität des Algorithmus mit anderen bzw. verunreinigten Daten umzugehen, zu erhöhen.

Diese Struktur ist in Abbildung 3 dargestellt. Es wird hier ein neuronales Netzwerk mit 5 Hidden Layers verwendet. Die unterschiedlichen Layer beziehen die Input-Information und extrahieren und beschreiben unterschiedliche Inhalte und Konzepte und geben diese Informationen wieder den nächsten Layer weiter.

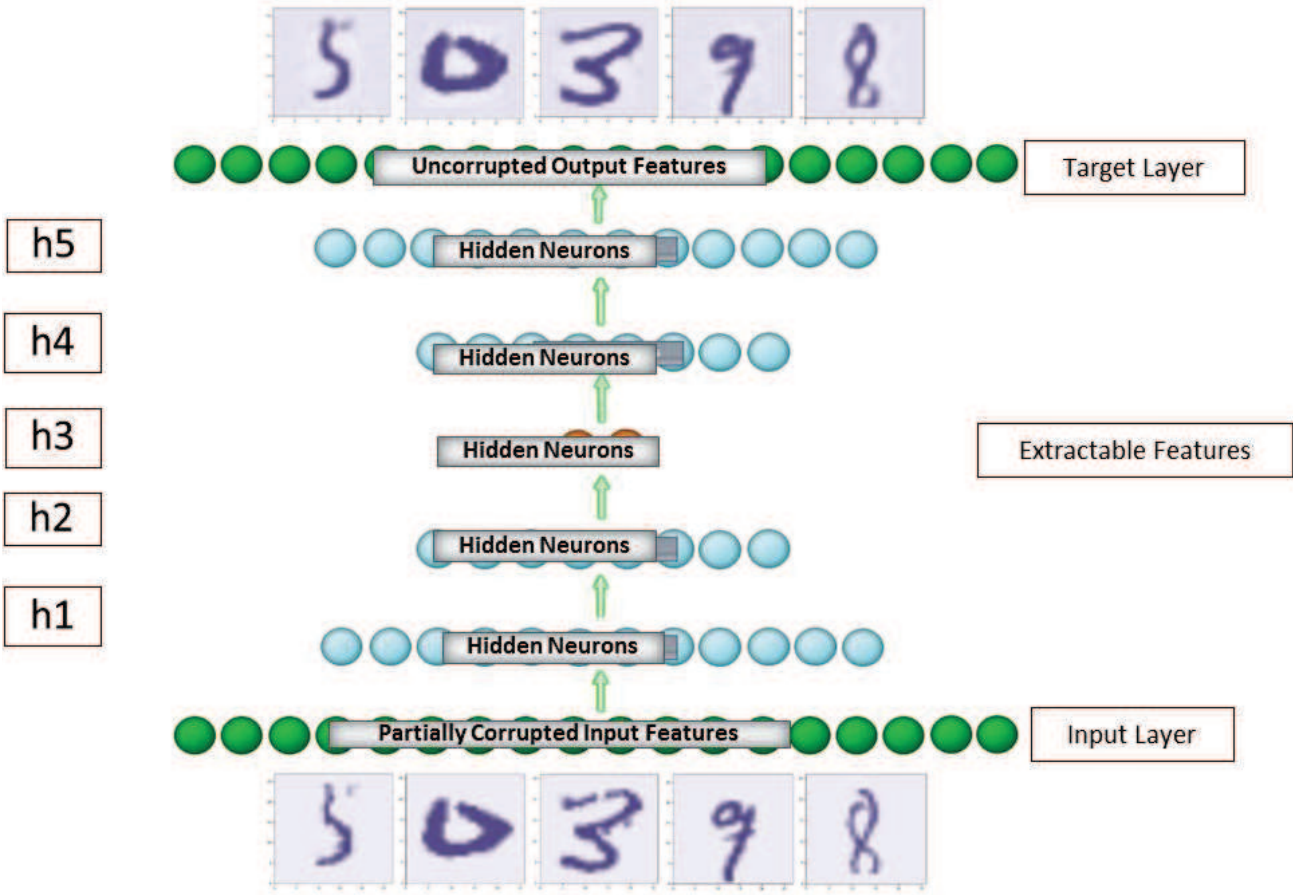

Abbildung 3: Architektur eines 5-stufigen Neural Networks für einen Autoencoders

Für die gegenständliche Analyse ist insbesondere der 3. Hidden Layer von Interesse. Dieser reduziert die Anzahl der Neuronen auf 2 und zwingt so das neuronale Netzwerk die Input-Information auf 2 Neuronen zu reduzieren um diese in weiteren Ebenen wieder zu extrahieren und die 784 Zielvariablen korrekt abzubilden.

3.2 Umsetzung mit SAS Code

In Abbildung 4 ist der Stacked Denoising Autoencoder nochmals horizontal dargestellt.

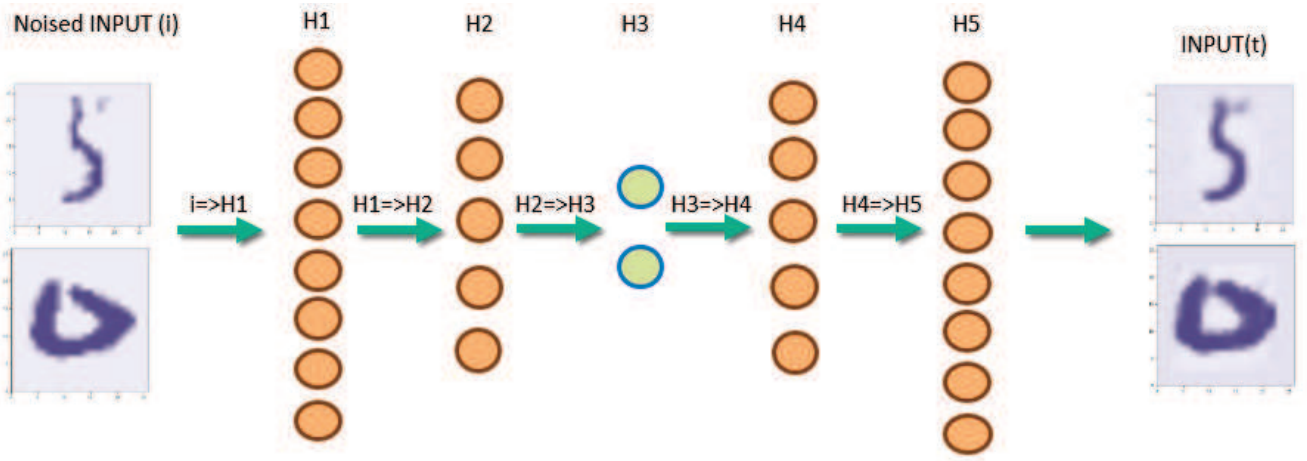

Abbildung 4: Autoencoder mit 5 Hidden Layers (Horizontale Darstellung)

Dieses neuronale Netzwerk wurde mit folgenden SAS Code mit der NEURAL Procedur des SAS Enterprise Miner dargestellt.

```
proc neural  data   = autoencoderTraining
             dmdbcat= work.autoencoderTrainingCat
             performance compile details cpucount= 12 threads= yes;

archi MLP hidden= 5;
hidden 300 / id= h1;
hidden 100 / id= h2;
hidden 2     / id= h3 act= linear;
hidden 100 / id= h4;
hidden 300 / id= h5;

input corruptedPixel1-corruptedPixel784/id= i level= int std= std;
target pixel1-pixel784/ act= identity id= t level= int std= std;

initial random= 123;  prelim 10 preiter= 10;
freeze h1->h2; freeze h2->h3;  freeze h3->h4; freeze h4->h5;
train technique= congra maxtime= 129600 maxiter= 1000;
freeze i->h1; thaw h1->h2;
train technique= congra maxtime= 129600 maxiter= 1000;
freeze h1->h2;  thaw h2->h3;
train technique= congra maxtime= 129600 maxiter= 1000;
freeze h2->h3;  thaw h3->h4;
train technique= congra maxtime= 129600 maxiter= 1000;
freeze h3->h4;  thaw h4->h5;
train technique= congra maxtime= 129600 maxiter= 1000;
thaw i->h1;  thaw h1->h2; thaw h2->h3; thaw h3->h4;
train technique= congra maxtime= 129600 maxiter= 1000;

code file= 'C:\Path\to\code.sas';

run;
```

- Zu Beginn wird die Architektur des Neuronalen Netzwerks definiert.
- Im nächsten Schritt werden die Input und Output Variablen definiert.
- Dann werden die Optimierungsoptionen für die einzelnen Hidden Layers definiert.

Für die weitere Analyse ist insbesondere der Scorecode (siehe CODE Statement) interessant. In diesem Code sind die Gewichte und Parametrisierung des Neuronalen Netzwerks enthalten, welche von den Input Daten zu den Werten in den einzelnen Hidden Layers und letztendlich zu den Zielvariablen führen. Dieser Code kann auf neue Daten angewandt und die jeweiligen Werte der Neuronen auf unterschiedlichen Hidden Layers extrahiert werden.

Für die Datenverdichtung ist die Extraktion der beiden Neuronen auf Hidden Layer 3 wichtig:

- Diese können entweder, wie oben beschrieben, durch Anwendung des Score Codes auf die Trainingsdaten oder neue Daten erhalten werden.
- Oder als Output Tabelle nach der Durchführung der NEURAL Prozedur für die Trainingsdaten erhalten werden.

3.3 Automatische Erkennung von "Konzepten" in den Daten

Eine Aufgabe von Deep Learning Neural Networks besteht darin, die Konzepte in den Daten (oder Bilddaten) zu extrahieren. Abbildung 5 zeigt die Verbindungen zwischen den Neuronen im Hidden Layer 5 und dem Output Layer.

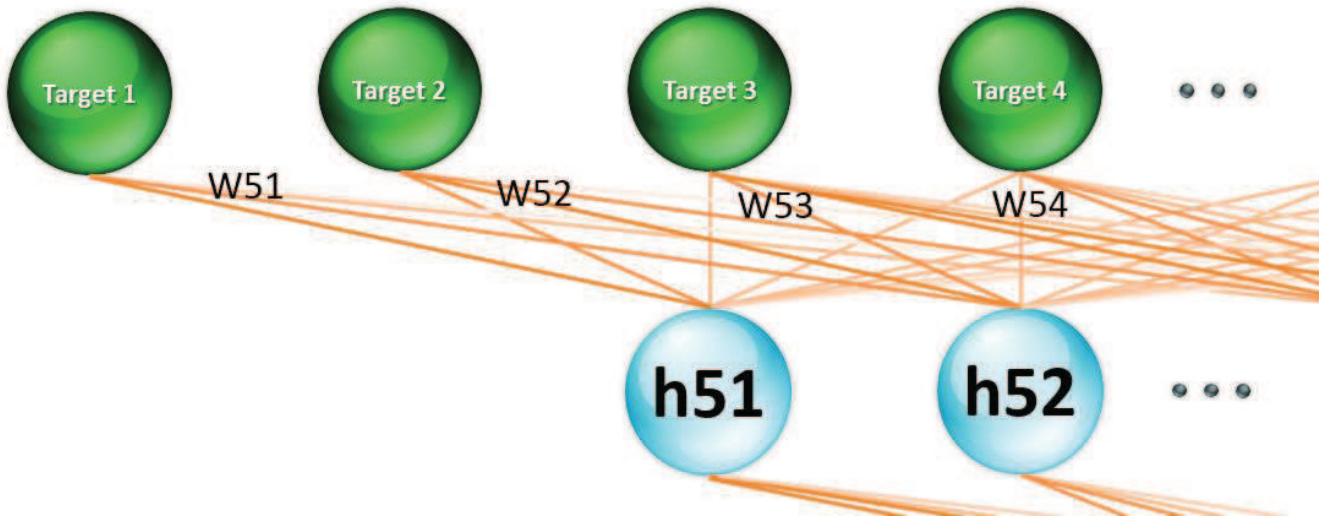

Abbildung 5: Darstellung einer ausgewählten Sektion des Autoencoders

Betrachtet man ausgewählte Scatterplots dieser Weight-Variablen W51, W52, … wie in Abbildung 6 dargestellt, sieht man, das in den einzelnen Bildern unterschiedliche Konzepte und Aspekte der ursprünglichen Bilder extrahiert wurden.

D.h. die einzelnen Weight-Variablen enthalten Informationen und Aspekte der zu beschreibenden Bilder, ohne dass deren Ziffernwert in der Analyse überhaupt verwendet wurde.

- Das 2. Bild in der ersten Zeile lässt Formen der Ziffer 5 erkennen.
- Das 4. Bild in der zweiten Zeile lässt Formen der Ziffer 6 erkennen.
- Die anderen Bilder enthalten unterschiedliche Aspekte der zu modellierenden Ziffernbilder.

Es ist dabei schön zu erkennen, dass Deep Learning Neural Networks in der Lage sind, Konzepte und Aspekte der zu modellierenden Daten zu extrahieren, ohne dass Eigenschaft, Gruppe oder das Zielbild selbst in der Analyse bekannt sind.

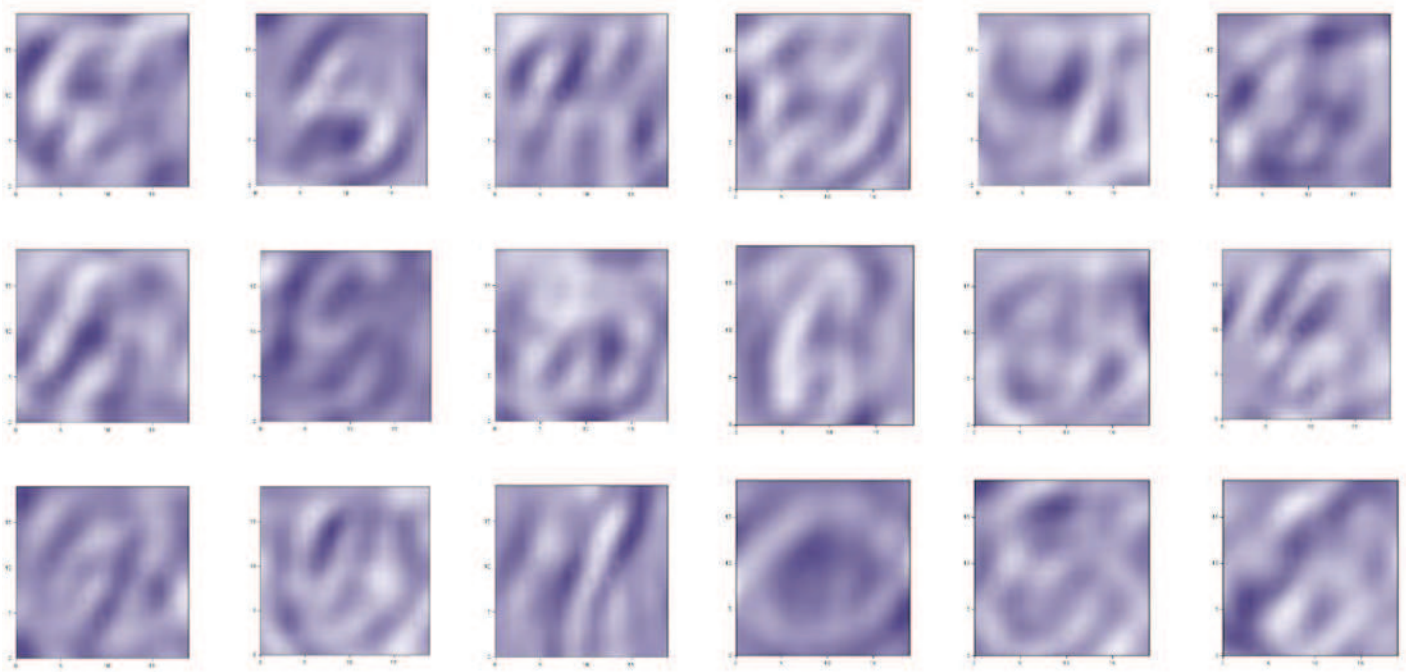

Abbildung 6: Automatisch erkannte Konzepte in den Daten

4 Darstellung der Ergebnisse

4.1 Informationskonzentration in den zwei Hidden Layers auf Ebene

Die Informationsverdichtung im Hidden Layer 3 auf 2 Neuronen kann nun in Form eines Scatterplots untersucht werden. Jedes Bild kann somit durch 2 Variablen beschrieben werden.

Abbildung 7 zeigt den Scatterplot der Werte der beiden Neuronen auf Hidden Layer 3. Zusätzlich sind die Punkte je nach tatsächlichen Ziffernbild farbkodiert. Dies ist das erste Mal in der hier beschriebenen Analyse, dass die tatsächlichen Ziffernwerte verwendet werden. Da im schwarz/weiß Druck die Farbschattierungen nicht gut differenzierbar sind, wurde zusätzlich der Legendenlabel zu ausgewählten Farbgruppen hinzugefügt.

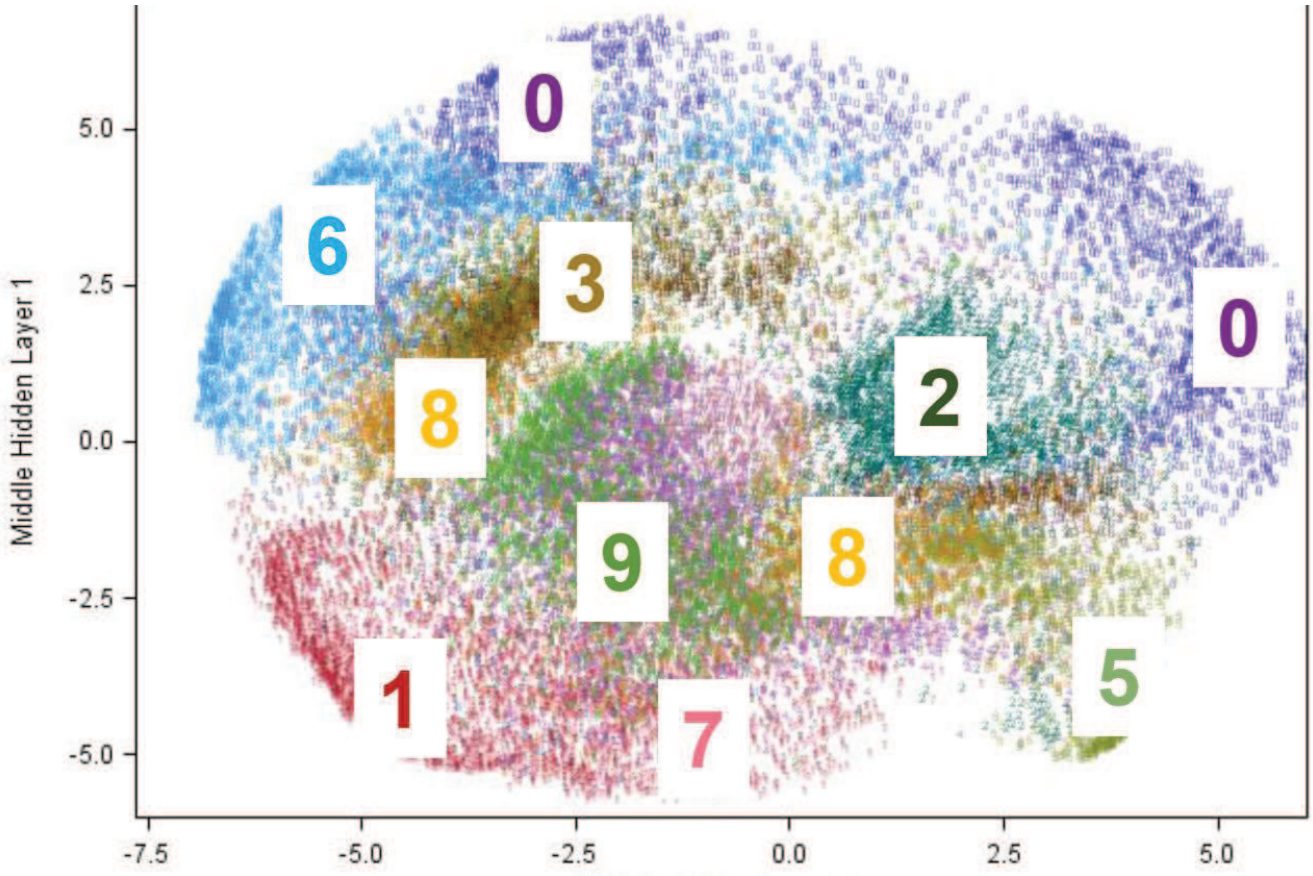

Abbildung 7: Scatterplot der beiden Hidden Layers auf Ebene 3

Es ist zu erkennen, dass die beiden Neuronen auf Ebene 3 eine sehr gute Trennung der Ziffernbilder 0-9 erlauben. Ziffern, die auch bei Betrachtung durch den Menschen als ähnlich betrachtet und ev. leicht verwechselt werden, liegen auch hier nahe beieinander: 1 und 7, 0 und 6, 3 und 8.

4.2 Ergebnisse einer Hauptkomponentenanalyse

Anstatt des der Definition und des Trainings eines Deep Learning Neuronalen Netzwerks könnte man auch versuchen, eine Hauptkomponentenanalyse auf die 784 Input Variablen durchzuführen und die ersten beiden Hauptkomponenten als beschreibende Merkmale zu verwenden.

Abbildung 8 zeigt die Ergebnisse in Form eines Scatterplots. Es ist zu sehen, dass die Trennung der einzelnen Zifferngruppen deutlich schlechter ist als beim Stacked Denoising Autoencoder. Lediglich die Ziffern 0 und 6 konnten von den anderen Ziffern abgegrenzt werden.

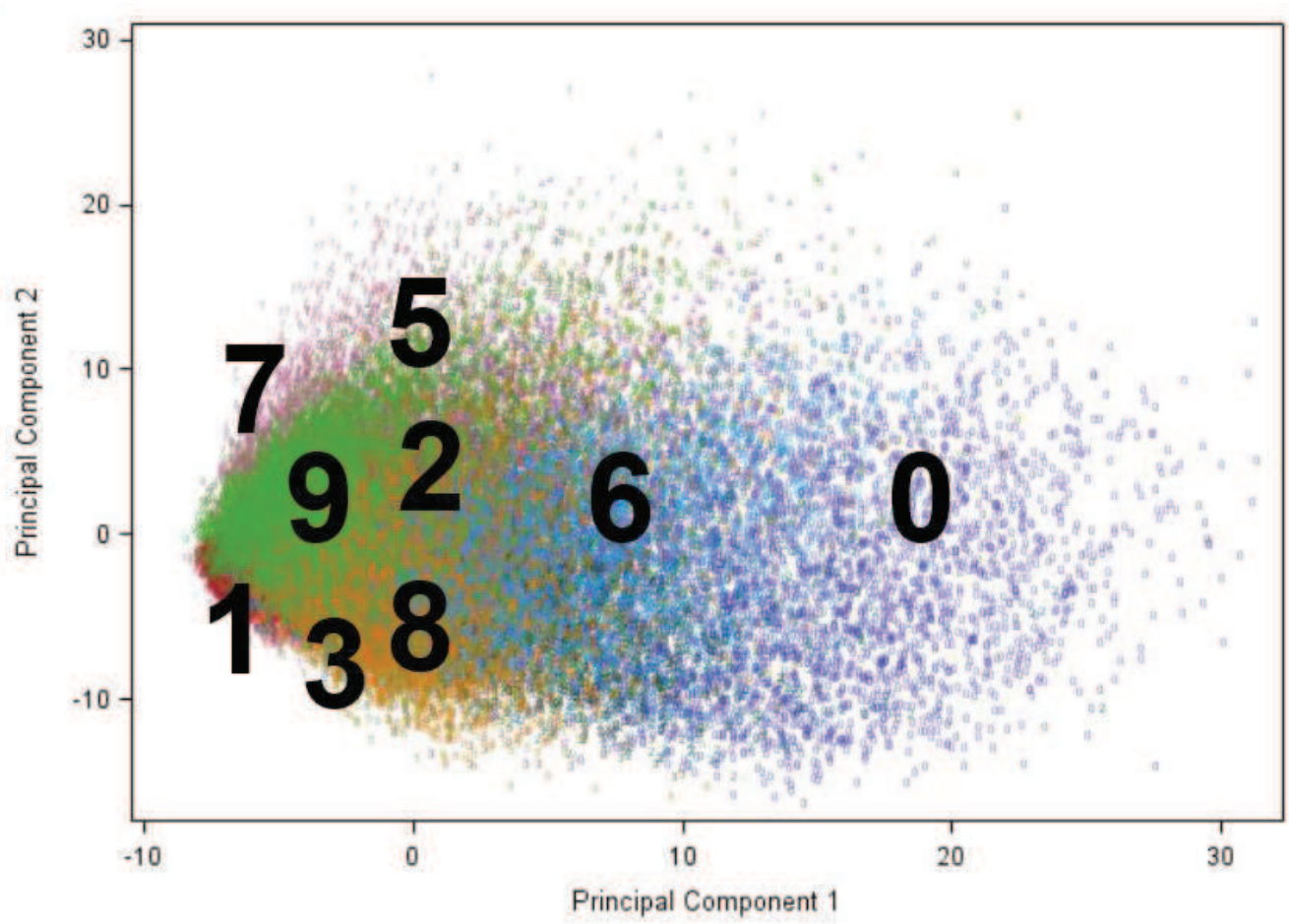

Abbildung 8: Scatterplot der beiden ersten Hauptkomponenten

5 Zusammenfassung und Schlußfolgerung

Dieses Beispiel zeigt sehr eindrucksvoll, dass Stacked Denoising Autoencoders sehr gut zur Komprimierung von Information in den Daten verwendet werden können. Diese Methoden führen nicht nur zu einer Informationsverdichtung mit hohem Informationsgehalt, sondern ermöglichen auch die Extraktion von Features in den Daten auf unterschiedlichen Ebenen des Neuronalen Netzwerks. Der Einsatz dieser Deep Learning Neural Networks führt zu sehr genauen Vorhersagen.

Deep Learning Verfahren kommen häufig bei der Mustererkennung in unstrukturierten Daten wie Text, Bilder oder Videos zum Einsatz.

Literatur

[1] P. Hall: “Overview of Machine Learning with SAS Enterprise Miner”, Proceedings des SAS Global Forums (2014)
http://support.sas.com/resources/papers/proceedings14/SAS313-2014.pdf

[2] G. Svolba: Data Quality for Analytics Using SAS: SAS Press 2012, Cary NC