



# ISP AUFGABE 4

Gruppe 3, Team 5

06.06.2017

Dimitri Meier, Saeed Shanidar, Andreas Berks

dimitri.meier@haw-hamburg.de,  
saeed.shanidar@haw-hamburg.de,  
andreas.berks@haw-hamburg.de

## Inhaltsverzeichnis

2.	Feedforward Neural Network .....	2
2.1.	Einfaches Netz mit Softmax-Funktion .....	2
2.2.	Mehrschichtiges Netz .....	2
2.3.	ReLU als Aktivierungsfunktion.....	3
2.4.	Adam-Optimizer und adaptive Lernrate .....	4
2.5.	Dropout .....	5
3.	Convolutional neural Network (CNN).....	7

## 2. Feedforward Neural Network

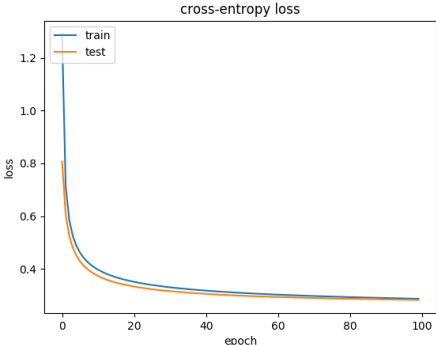
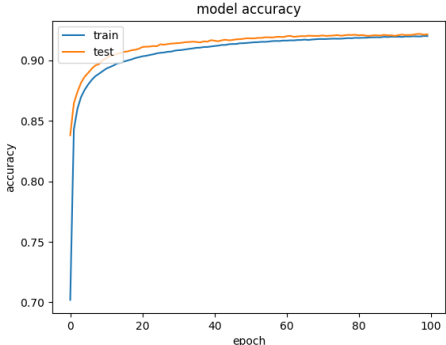
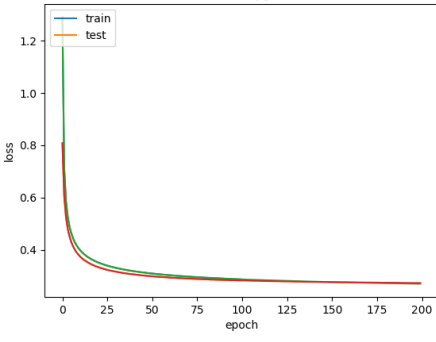
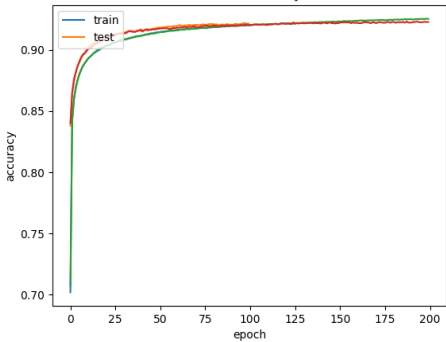
### 2.1. Einfaches Netz mit Softmax-Funktion

#### 2.1.1. Beschreibung

Es soll ein einfaches Netz konzipiert werden, welches direkt aus der Eingabe-Schicht die Wahrscheinlichkeiten zur Vorhersage der Klassen berechnet.

Dies kann in Form einer Softmax-Schicht erfolgen.

#### 2.1.2. Plots

Einschichtiges Netz mit Softmax activation			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
100			Test loss: <b>0.282576597619</b>  Test accuracy: <b>0.9214</b>
200			Test loss: <b>0.272844834143</b>  Test accuracy: <b>0.9226</b>

#### 2.1.3. Beobachtungen

TODO

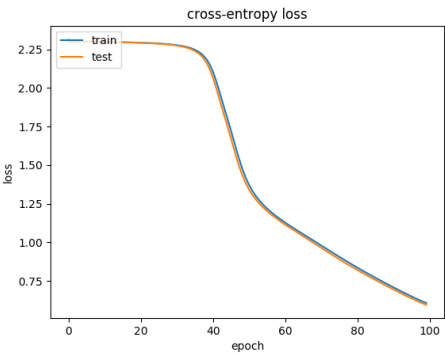
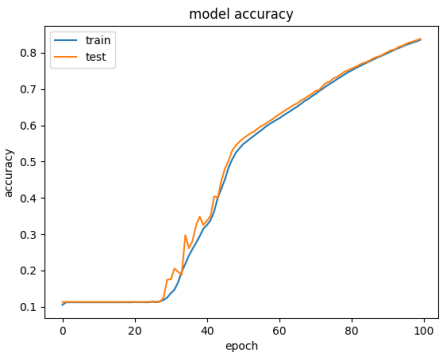
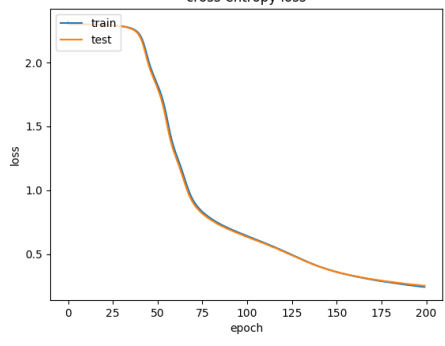
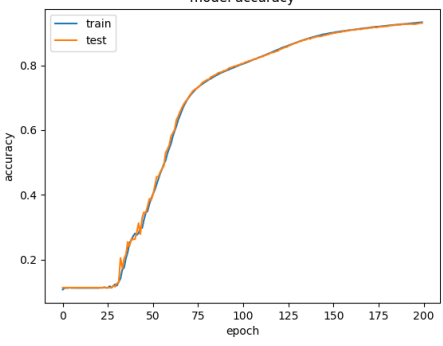
### 2.2. Mehrschichtiges Netz

#### 2.2.1. Beschreibung

Bauen Sie nun ein mehrschichtiges neuronales Netz. Das Netz soll aus aufeinanderfolgenden Schichten mit 200,100,60 und 30 Neuronen bestehen.

Diese sollen jeweils die Sigmoid-Funktion zur Aktivierung verwenden. Die darauffolgende wie zuvor mit der Softmax-Funktion.

### 2.2.2.Plots

Mehrschichtiges Netz mit Softmax und Sigmoid			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
100			Test loss: <b>0.595168864346</b> Test accuracy: <b>0.838</b>
200			Test loss: <b>0.254255178577</b> Test accuracy: <b>0.9303</b>

### 2.2.3.Beobachtungen

TODO

## 2.3. ReLU als Aktivierungsfunktion

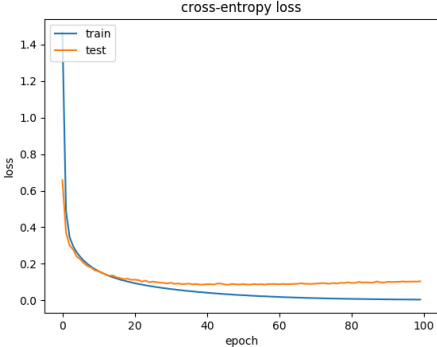
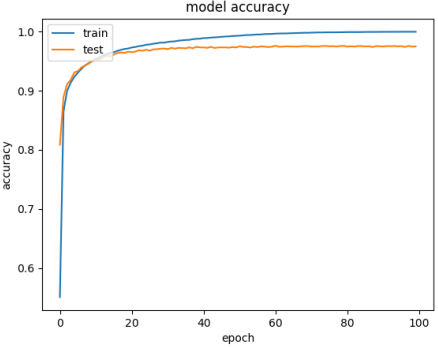
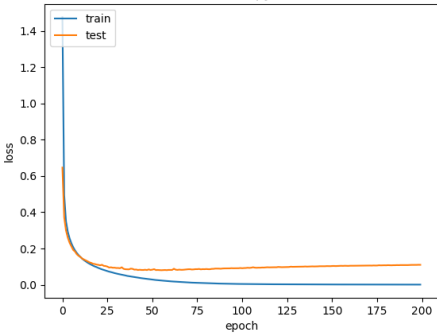
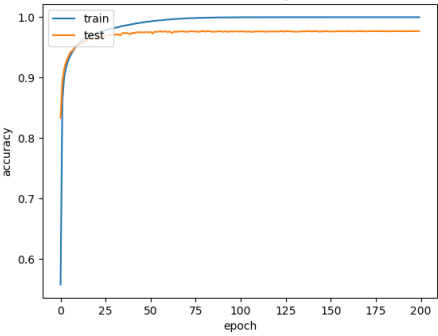
### 2.3.1.Beschreibung

Ersetzen Sie nun die Sigmoid- durch die ReLU-Funktion zu Aktivierung.

Welcher Effekt lässt sich nun durch die Plots beobachten?

Wie genau ist das Modell nun im Vergleich zu den anderen Modellen?

### 2.3.2.Plots

Mehrschichtiges Netz mit Softmax und ReLU			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
100			Test loss: <b>0.104041506718</b>  Test accuracy: <b>0.975</b>
200			Test loss: <b>0.109916656613</b>  Test accuracy: <b>0.977</b>

### 2.3.3.Beobachtungen

TODO

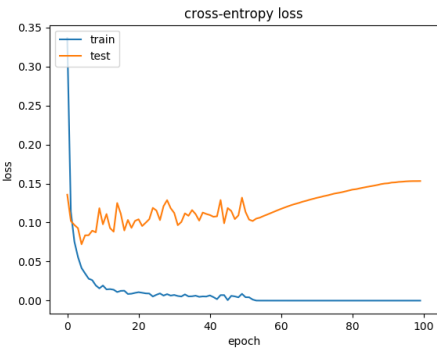
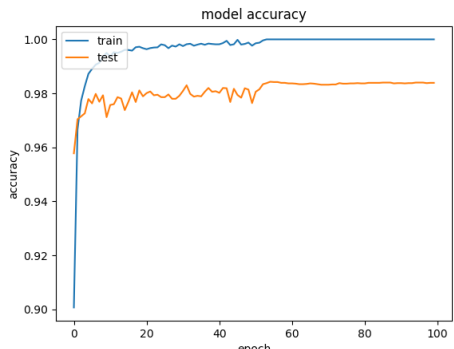
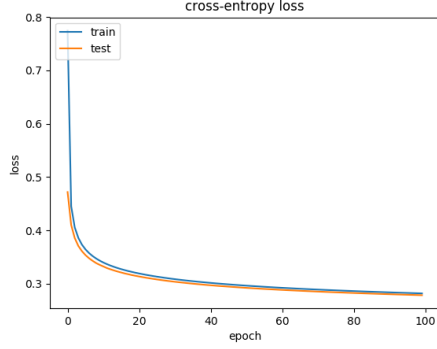
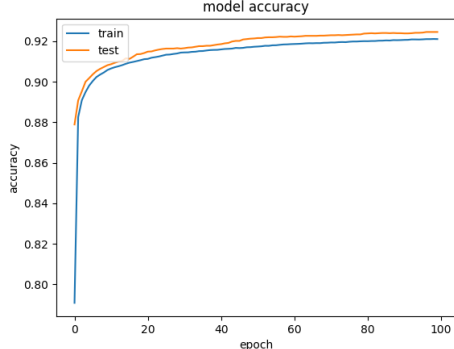
## 2.4. Adam-Optimizer und adaptive Lernrate

### 2.4.1.Beschreibung

Verwenden Sie nun Adan zum Training des Netzes.

Der Parameter decay steuert dabei die Verringerung der Lernrate nach jedem Lernschritt. Führen Sie nun für **EPOCH=100** Iterationen zwei Versuche durch, wobei Sie zunächst **decay=0.0** verwenden und im Anschluss **decay=0.01** einstellen. Was beobachten Sie nun im Plot der Loss-Funktion?

### 2.4.2.Plots

Mehrschichtiges Netz mit Softmax und ReLU (Adam-Optimizer)			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
100			Test loss: <b>0.153217251623</b> Test accuracy: <b>0.9839</b>  <b>decay=0.0</b>
100			Test loss: <b>0.278563123053</b> Test accuracy: <b>0.9246</b>  <b>decay=0.1</b>

### 2.4.3.Beobachtungen

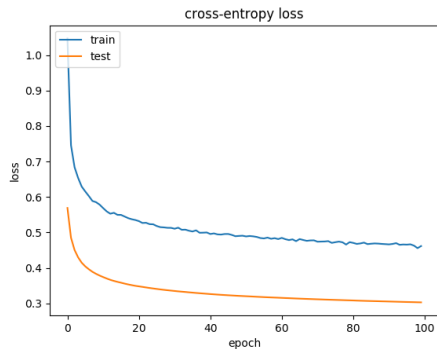
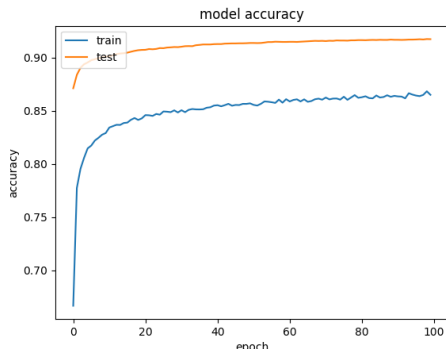
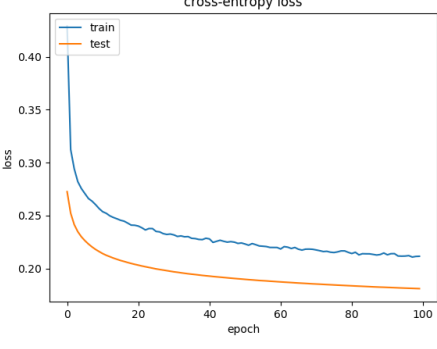
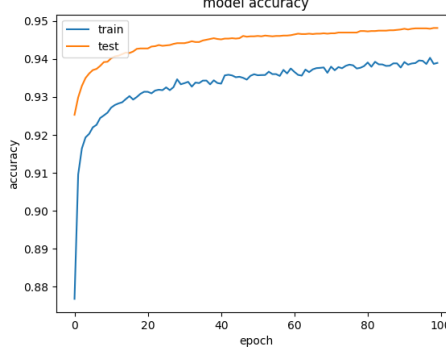
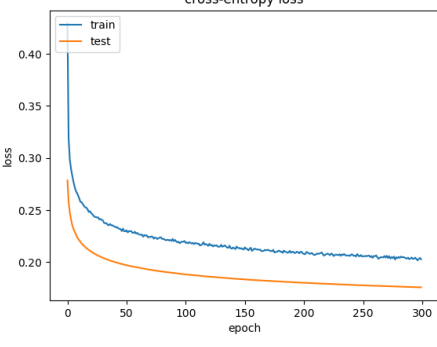
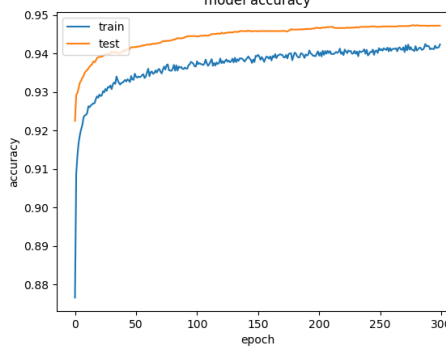
## 2.5. Dropout

### 2.5.1.Beschreibung

Fügen Sie nun dem zuvor verwendeten Modell Dropout hinzu, indem Sie nach jeder Schicht (ausgenommen die Softmax-Schicht) ein Dropout hinzufügen.

Nun können Sie verschiedene Werte für Dropout ausprobieren und den Effekt auf das Training des Netzes beobachten. Auch können Sie verschiedene Architekturen ausprobieren. Gute Ergebnisse sollten ins Protokoll aufgenommen werden.

### 2.5.2.Plots

Mehrschichtiges Netz mit Softmax und ReLU (Adam-Optimizer) + Dropout			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
100			<p>Test loss: <b>0.302877331418</b></p> <p>Test accuracy: <b>0.9174</b></p> <p><b>decay=0.1</b> <b>dropout=0.1</b></p>
100			<p>Test loss: <b>0.180879609174</b></p> <p>Test accuracy: <b>0.9482</b></p> <p><b>decay=0.1</b> <b>dropout=0.2</b></p>
300			<p>Test loss: <b>0.175914991759</b></p> <p>Test accuracy: <b>0.9472</b></p> <p><b>decay=0.1</b> <b>dropout=0.2</b></p>

### 2.5.3.Beobachtungen

TODO

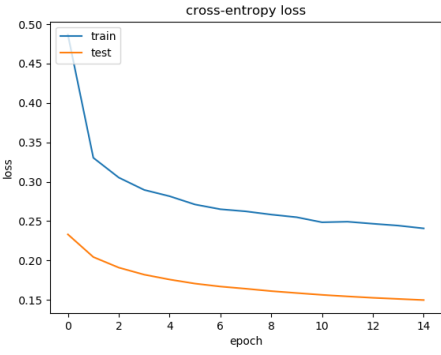
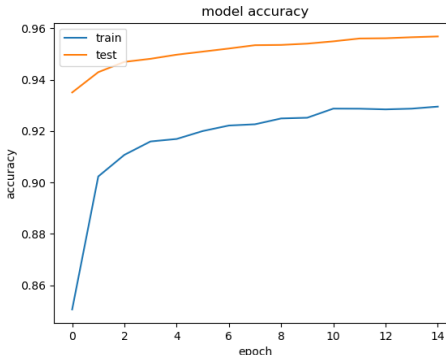
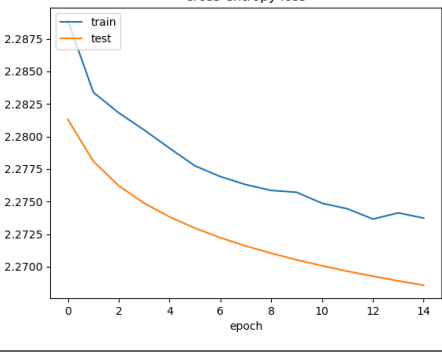
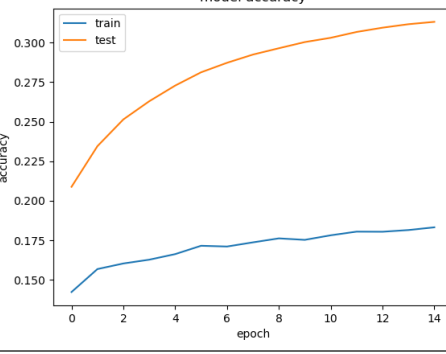
### 3. Convolutional neural Network (CNN)

#### 3.3. Beschreibung

Verändern Sie zunächst die Vorverarbeitung der Trainings- und Testdaten. Die zuvor verwendete Aneinanderreihung von Spalten der Pixel-Matrizen wird nun nicht mehr benötigt. Stattdessen muss eine Dimension hinzugefügt werden, damit die Daten mit den auszuführenden TensorFlow-Funktionen kompatibel sind.

Trainieren Sie das Modell für EPOCHS=15 Iterationen. Dabei sollten Sie ein sehr gutes Ergebnis feststellen. Sie können ebenfalls eine größere Anzahl EPOCHS ausprobieren und beobachten, wann das Training konvergiert. Des Weiteren können anstatt von Adam auch die Optimizer Adadelta oder RMSprop ausprobieren. Plotten Sie die Ergebnisse und nehmen Sie die Plots in das Protokoll auf.

#### 3.4. Plots

CNN Netz			
EPOCH	COST FUNKTION	GENAUIGKEIT	SCORE
15			Test loss: <b>0.149793775633</b>  Test accuracy: <b>0.9568</b>  <b>decay=0.1</b> <b>Optimizer=Adam</b>
15			Test loss: <b>2.26858293266</b>  Test accuracy: <b>0.3131</b>  <b>decay=0.1</b> <b>Optimizer=Adadelta</b>

#### 3.5. Beschreibung

TODO