Zalando Fashion-MNIST

PROJEKTARBEIT

ZU

MACHINE LEARNING

Master

"DATA SCIENCE & INTELLIGENT ANALYTICS"

FH-Kufstein Tirol

Autoren:

Magdalena Breu | 1810837687

Valentin Muhr | 1810837102

Christoph Rabensteiner | 1810837995

Jochen Hollich | 1810837475

Franz Innerbichler | 1810837297

06. Juli 2019

Inhaltsverzeichnis

1. Einleitung

Pattern Recognition und -Klassifizierung ist eine zentrale Problemstellung im Feld der Computer Vision. Mithilfe von Machine Learning (ML) Algorithmen und Künstlicher Intelligenz sollen Muster erkannt werden und das Bild einer Klasse zugeordnet werden. Im Zuge dieser Projektarbeit soll ein Klassifizierungs-Modell auf dem Zalando Fashion-MNIST Datensatz trainiert und getestet werden. Dafür wurde auf verschiedene Architekturen von Convolutional Neural Networks (CNN) zurückgegriffen.

Als Benchmark für Bilderkennungsalgorithmen gilt der MNIST Datensatz, bestehend aus 70,000 Bildern von handgeschriebenen Ziffern, an dem neue Algorithmen validiert und evaluiert werden können. Der Zalando Fashion-MNIST Datensatz wurde als alternativer Benchmark für Machine Learning (ML) Algorithmen erstellt. Als Grund dafür nennt das Zalando Research Team, dass die Problemstellung der Erkennung von handgeschriebenen Ziffern nicht mehr zeitgemäß wäre und bereits Accuracies von 99.7% erzielt werden könnten, daher besteht sehr wohl der Bedarf nach einem neuen Benchmarking Dataset.

Der Zalando Fashion-MNIST Datensatz besteht aus 70,000 Beispielbildern von Fashion Items. Es gibt 10 Klassen von Bildern, aufbereitet als gelabelte 28x28 Graustufenbilder.

Tab 1.1 Labels des Zalando Datensatzes

Label	**Description**
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

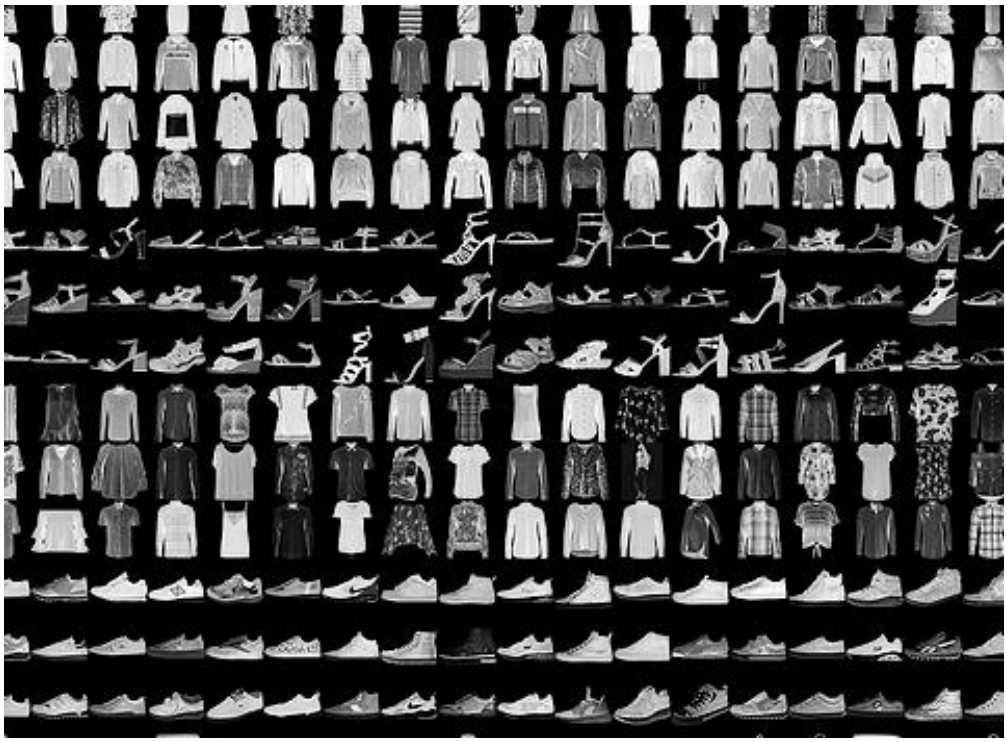


Abbildung 1 Fashion-MNIST Beispiel Bilder[1]

2. Methodik

Für die Bearbeitung der Problemstellung wurden zwei CNNs jeweils mit unterschiedlichen Hyperparametern trainiert:

- Epochen: 10, 50, 100
- Dropout: 75%, 50%, 25%, 0
  (Definiert die Wahrscheinlichkeit, dass ein Neuron im Netzwerk verbleibt)
- Learning rate: 0.01, 0.001

Epoche: Ein Dataset wird einmal vorwärts und rückwärts durch das NN geführt. Pro Epoche werden die Gewichtungen angepasst. Ein zu geringes n für Epochen führt dazu, dass die komplexten Visuellen Features der Bilddaten nicht erlernt werden können und das Netzwerk nicht and die das best-mögliche Fehlerminimum kommt.

Dropout: Technik zur Regularisierung. Kleiner Wert kann zu Underfitting führen, da zu viele Neuronen aus dem Netzwerk gestoßen werden und das Modell dadurch nicht genügen Kapazität besitzt die visuellen Features zu erlernen.

Learning rate: Definiert wie schnell das NN die Parameter aktualisiert, werden hier zu große Schritte definiert, kann es sein, dass das Modell das best-mögliche Fehlerminimum überspringt und nicht best-

[1] https://github.com/zalandoresearch/fashion-mnist

möglich performt. Wird eine zu niedrige Rate definiert, kann es sein, dass das Modell an lokalen Minima hängen bleibt und das tatsächliche Minimum niemals findet.

2.1. AlexNet

Die AlexNet-Architektur besteht aus acht Layern, wobei die ersten fünf Layer Convolutional Layer sind (Anwendung von Filterkernels) und die letzten drei Fully Connected Layer. Als Aktivierungsfunktion werden ReLU Funktionen verwendet.

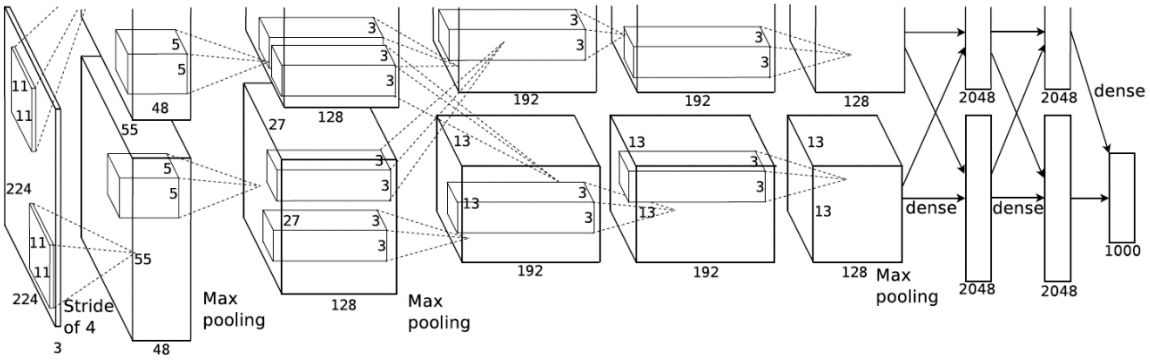


Abbildung 2 AlexNet Architektur[2]

2.2. LeNet-5

LeNet besteht aus sieben Layer, bestehend aus drei Convolutional Layer, zwei Pooling Layers, einem Fully Connceted Layer und einem Output Layer. Als Aktivierungsfunktionen werden Tangens hyperbolicus Funktionen verwendet.

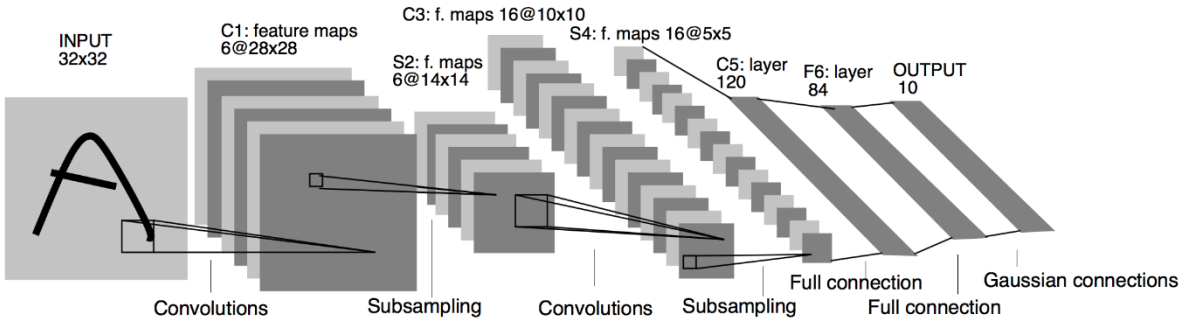


Abbildung 3 LeNet Architektur[3]

[2] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[3] https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4

Max Pechyonkin zitiert das Original-Paper von LeCun (https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4) und schreibt über Lene-5), dass es verschiedene Architektonische Kniffe im Lenet-5 gibt, die aber nicht mehr modern sind. Erstens verwenden die individual convolutional kernels im layer C3 nicht alle Features aus dem Layer C2 um es einerseits weniger rechenintensiv zu machen, was nach heutigem Standard unüblich ist. Andererseits macht es Sinn convolutional kernels unterschiedliche Muster lernen zu lassen: unterschiedliche Inputs erlaubt das Lernen unterschiedlicher Pattern.

Als 2. Grund nennt Max Peychonkin, die interne Architektur, die auf den layer F6 und die 10 Neurons der output layers zugeschnitten sind.

3 Training

Die Modelle AlexNet und Lenet-5 wurden nativ unter Tensorflow implementiert und auf GPU-Basis trainiert. Für dieses Szenario eignen sich GPU-basierte Implementierungen besonders, da diese von der höheren Rechenkapazität und schnelleren Berechnungsoperationen profitieren können. Die Trainings der Modelle wurde unter Beachtung der oben definierten Hyperparameter durchgeführt, sodass wir jedes AlexNet und jedes LeNet-5 insgesamt 24-mal trainierten. Dabei wurden vom Trainingsdatensatz 10% für die interne Validierung beim Training abgespalten. Die genaue Verteilung liegt bei (60%, 10%, 20%) für Training, Validierung und Testing.

Die gewählten Hyperparameter verblieben beim Training statisch und passten sich nicht dynamisch an. Obwohl es diese Funktionalität durchaus gibt, speziell für die dynamische Anpassung der Learning Rate, haben wir uns bewusst dagegen entschieden mit dem Ziel die Unterschiede in den verschiedenen Hyperparameter zu erkennen und welche Auswirkung sie auf das Training von Modellen haben.

Das Training der Modelle beanspruchte im Durchschnitt 3 Minuten für das Training mit 10 Epochen und durchschnittlich 30-60 Minuten für das Training mit 100 Epochen.

Alle Ergebnisse der Auswertungen durch Validierung und die daraus gewonnen Werte für Accuracy und Loss wurden in einem Logfile abgespeichert, um später in Tensorboard zu untersuchen und zu interpretieren.

4 Ergebnisse

Die Ergebnisse der Modellierung können ausgezeichnet in TensorBoard (Teil von Tensorflow) visualisiert werden (siehe Abbildung 4 Tensorboard Dashboard). Tensorboard zeigt die Accuracy und die Loss Veränderungen des Trainings über die Zeit des Trainings hinweg unter Verwendung eines Validierungsdatensatz. Die Ergebnisse der Test-Bilder sind in Confusion matrices wiedergegeben.

Abbildung 4 Tensorboard Dashboard

Abbildung 4 zeigt gefittete Modelle mit einer hohen Smoothing rate, um die Performanz besser anzeigen zu können. Manche Modelle schmieren ab, wie z.B. Modell alexnet_E50_D0.5_R0.01, manche Modelle konnten kein Learning erreichen (Modelle, die bei 0.1 Accuracy stehen bleiben).

Im Folgenden sind die Ergebnisse pro Epochen, Drop-out rate, Learning Rate und NN dargestellt.

4.1 Ergebnisse pro Epochen, Dropout Rate, Learning Rate und NN

Nachfolgend sind die Trainingsverläufe der Modelle aus dem Tensorboard dargestellt. Die Abbildungen zeigen die Übersichtsgraphen für die accuracy- (groß) und die lossfunktion (klein) in einem relevanten Ausschnitt. Die Modelle wurden nach dem jeweiligen Parameter gefiltert, z.b. wenn E10 angegeben ist, dann sind alle Modelle dargestellt, die mit 10 Epochen gefittet worden sind.

E10: (siehe Abbildung 5 Modelle mit E10)

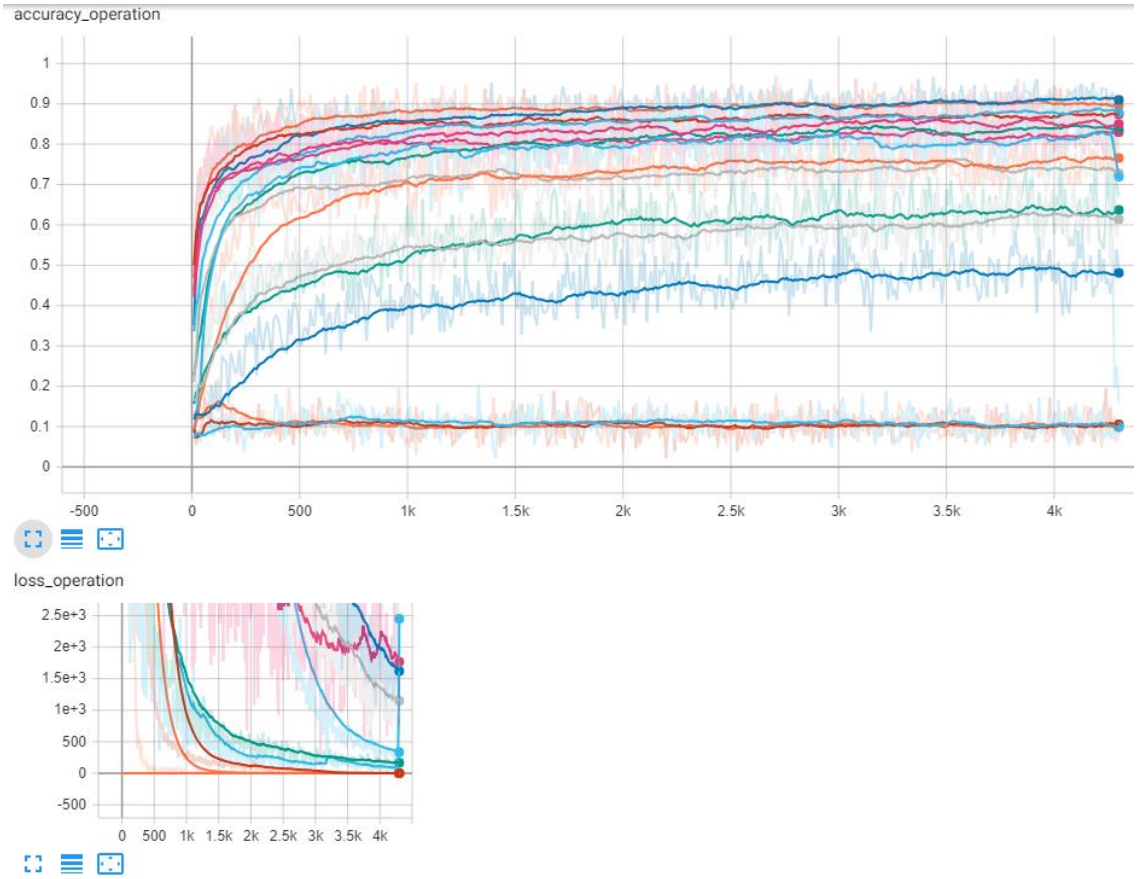


Abbildung 5 Modelle mit E10

E50: (siehe Abb 5)

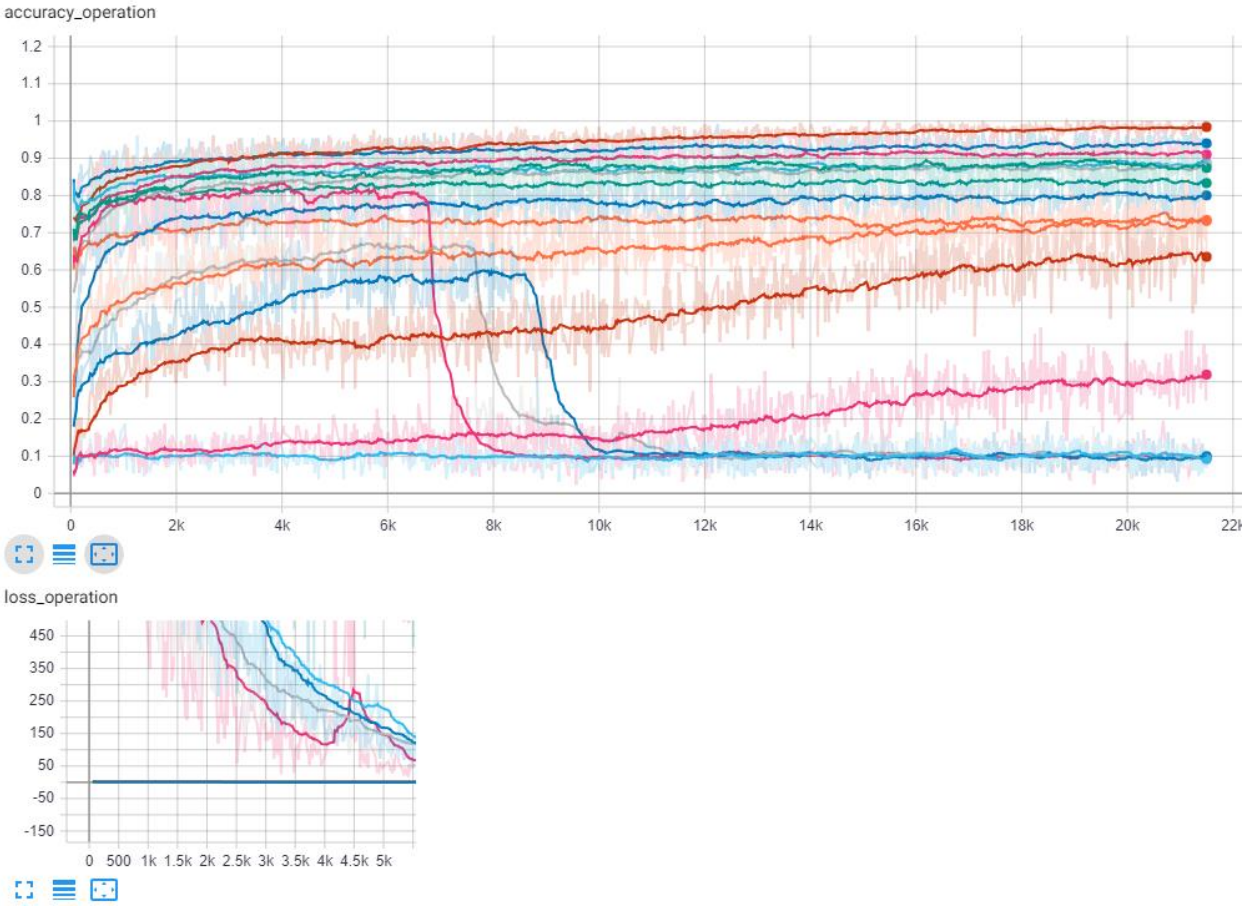


Abbildung 6 Modelle mit E50

E100: (siehe Abbildung 7 Modelle mit E100)

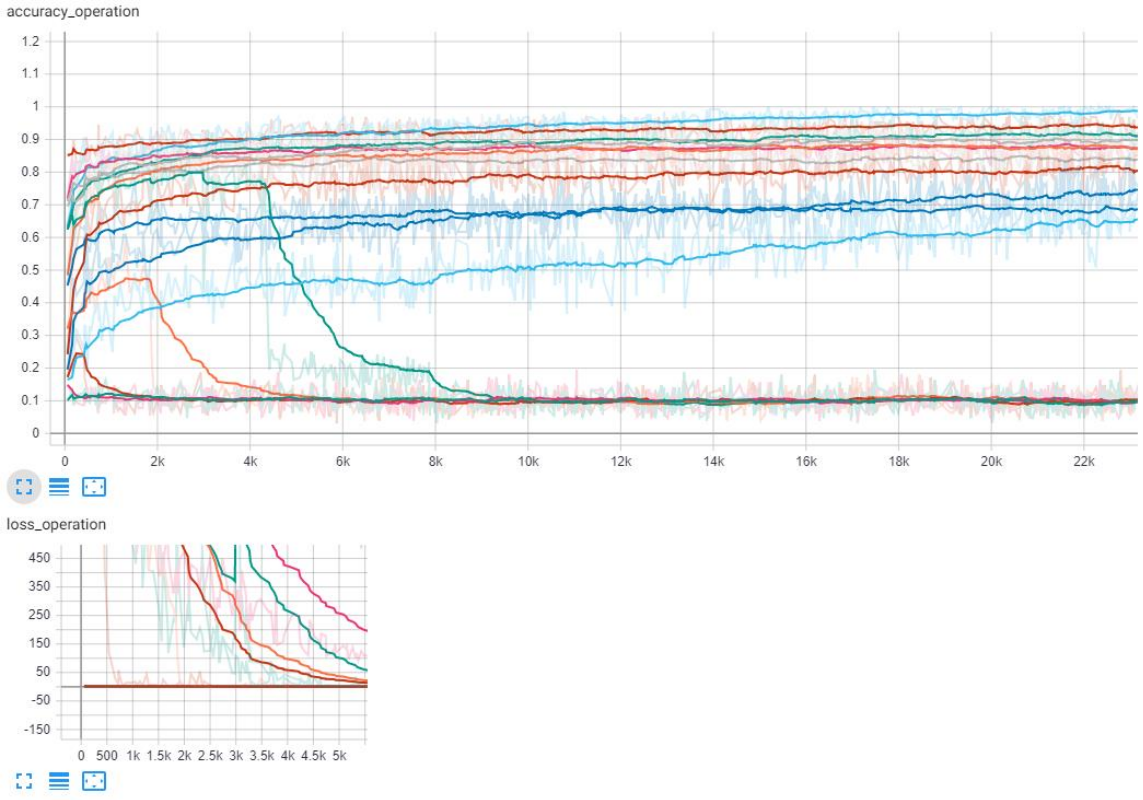


Abbildung 7 Modelle mit E100

D1.0: Dropout ist 0%, d.h. alle Hidden Nodes bleiben im Modell (Gefahr von Overfitting): Lenet performt besser als AlexNet (siehe Abbildung 8 Modelle mit D 1.0 8)

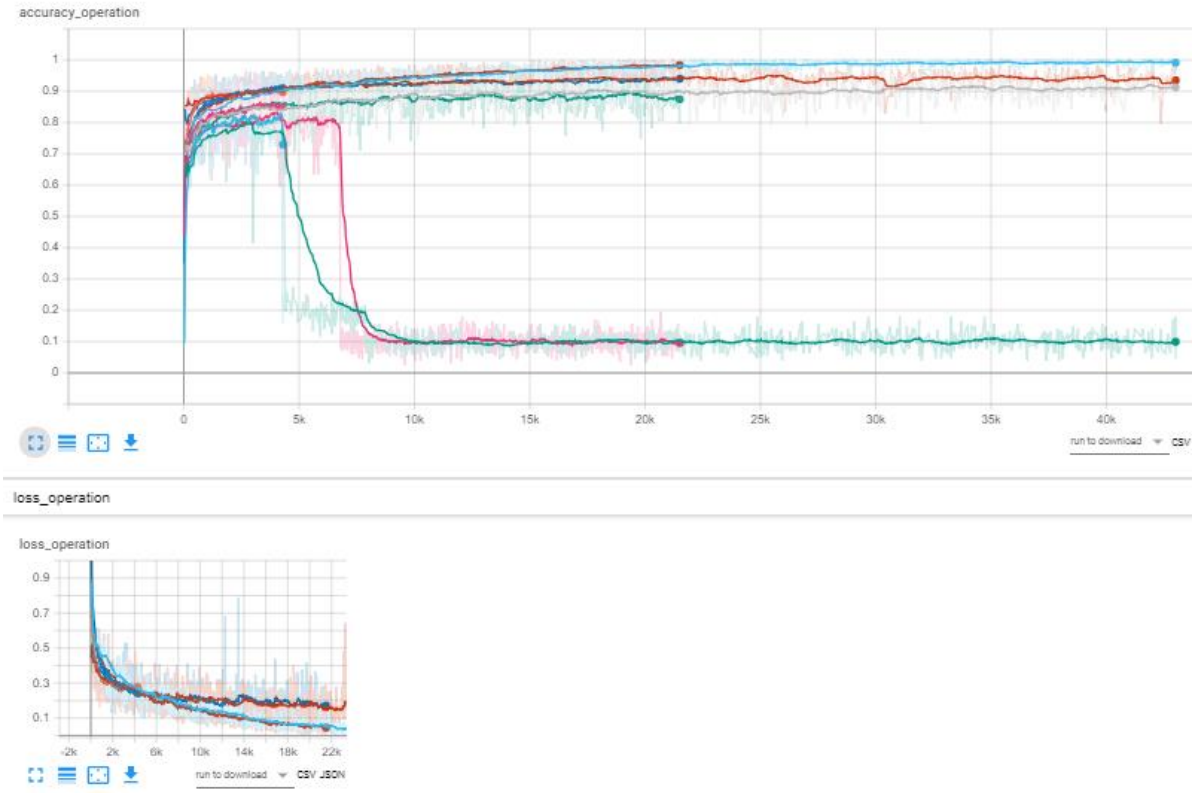


Abbildung 8 Modelle mit D 1.0

D0.75: Dropout ist 25%; Lenet performt viel besser als AlexNet (siehe Abb 9)

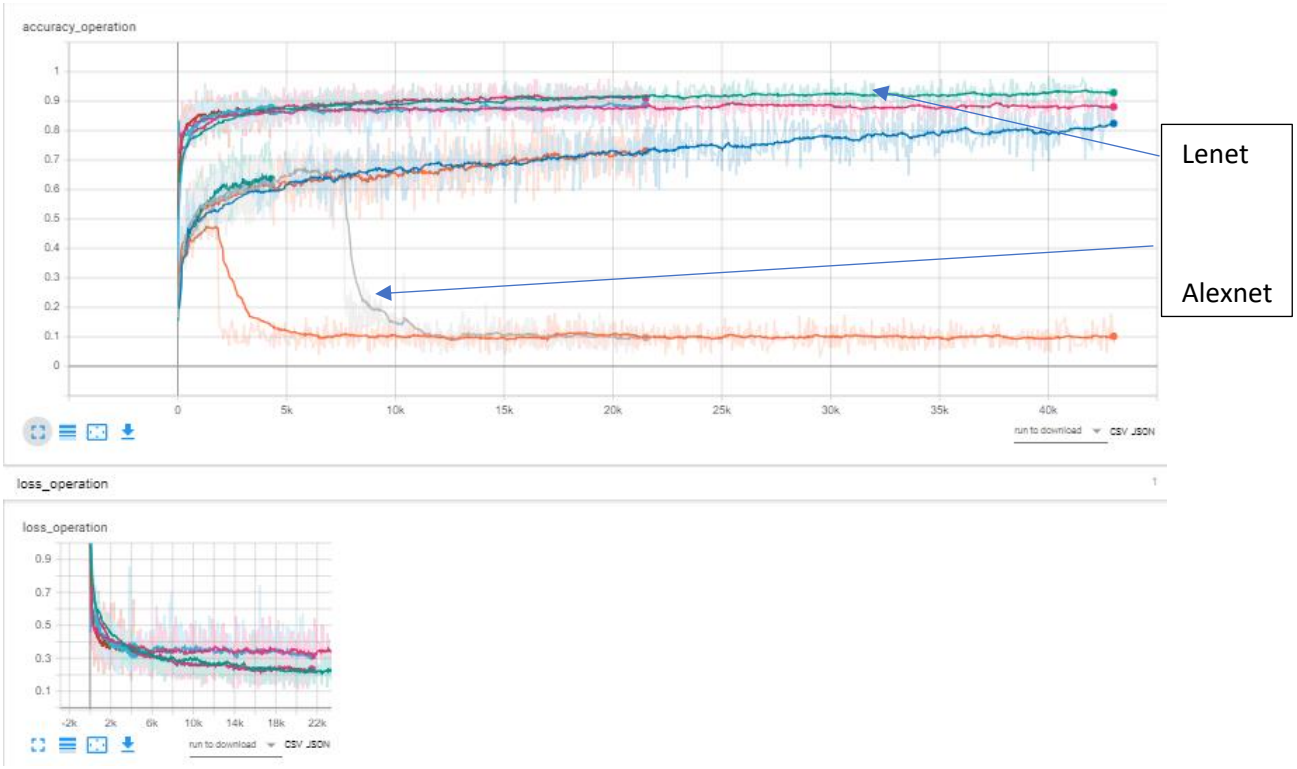


Abbildung 9 Modelle mit D 0.75

D0.5: Dropout von 50% der Hidden Nodes: Lenet besser als AlexNet (siehe Abb 10)

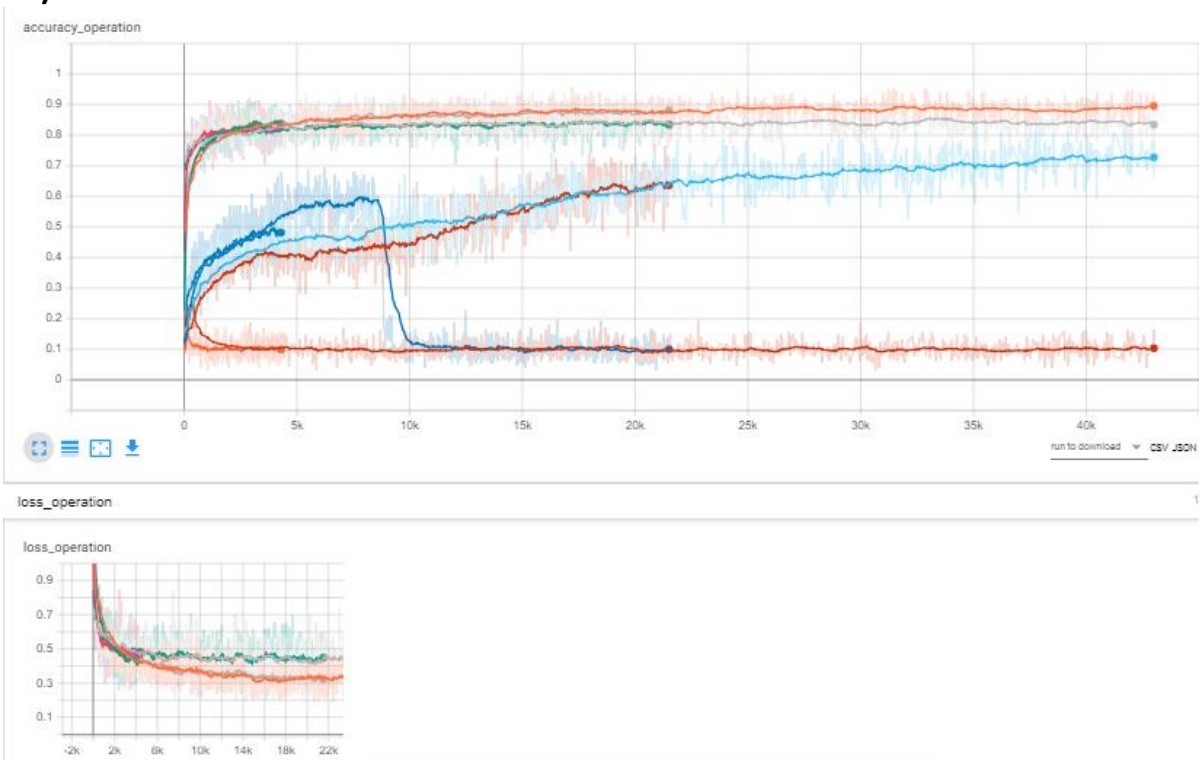


Abbildung 10 Modell mit D 0.5

D0.25: Dropout von 75% der Hidden Nodes; AlexNet Modelle versagen (siehe Abb 11)

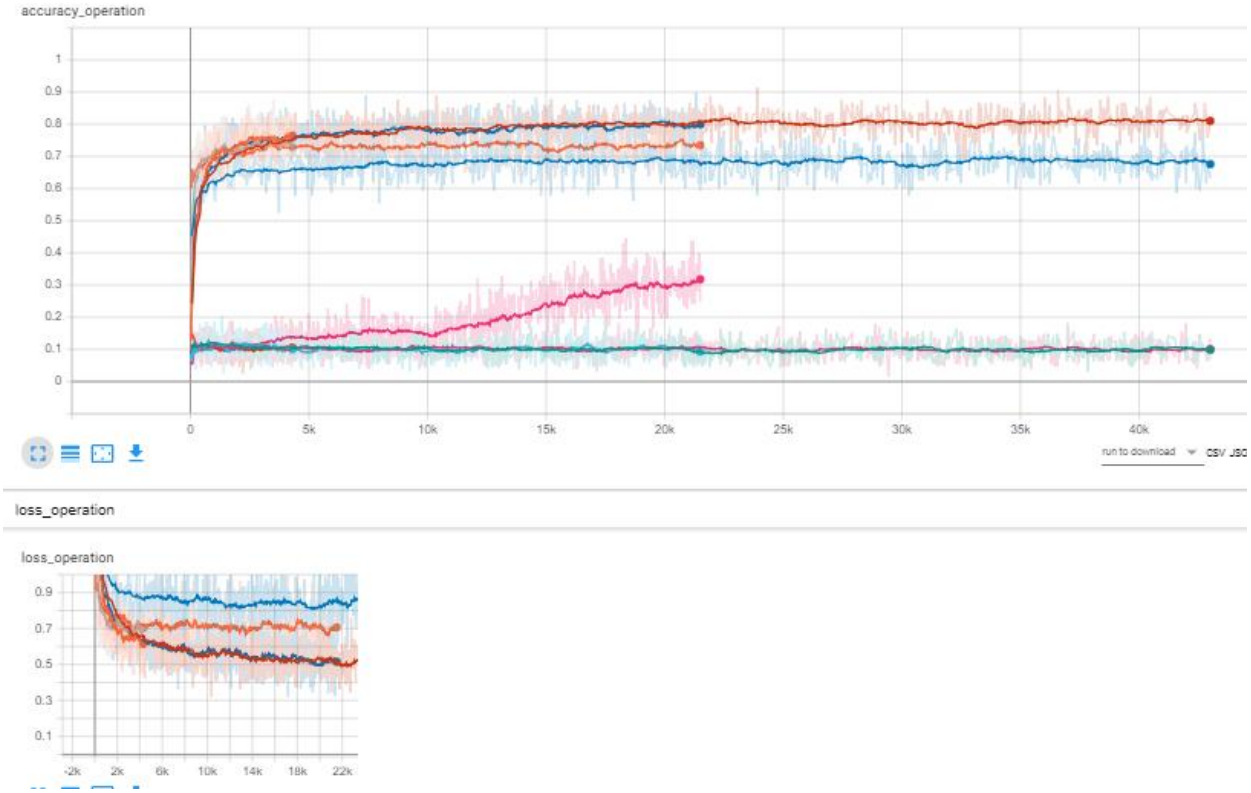


Abbildung 11 Modelle mit D 0.25

R0.01: wieder ist Lenet besser als AlexNet (siehe Abb 12)

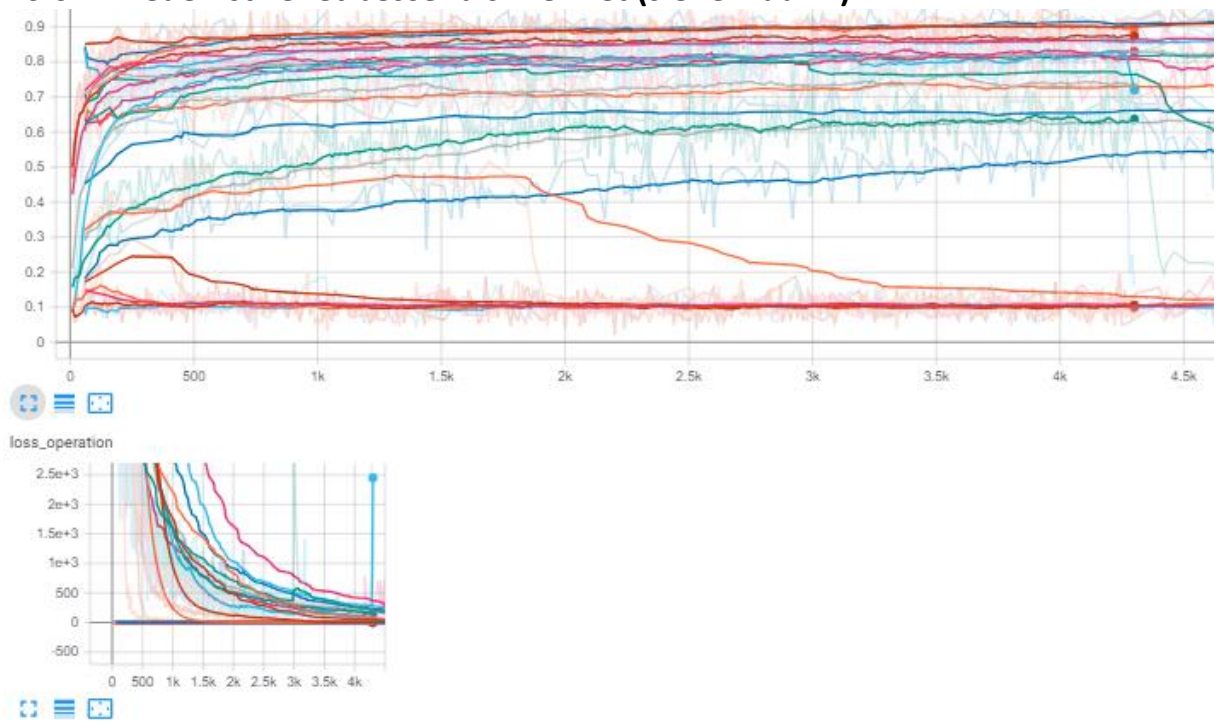


Abbildung 12 Modell mit R 0.01

R0.001 loss_operation, wieder Lenet besser (siehe Abb 13)

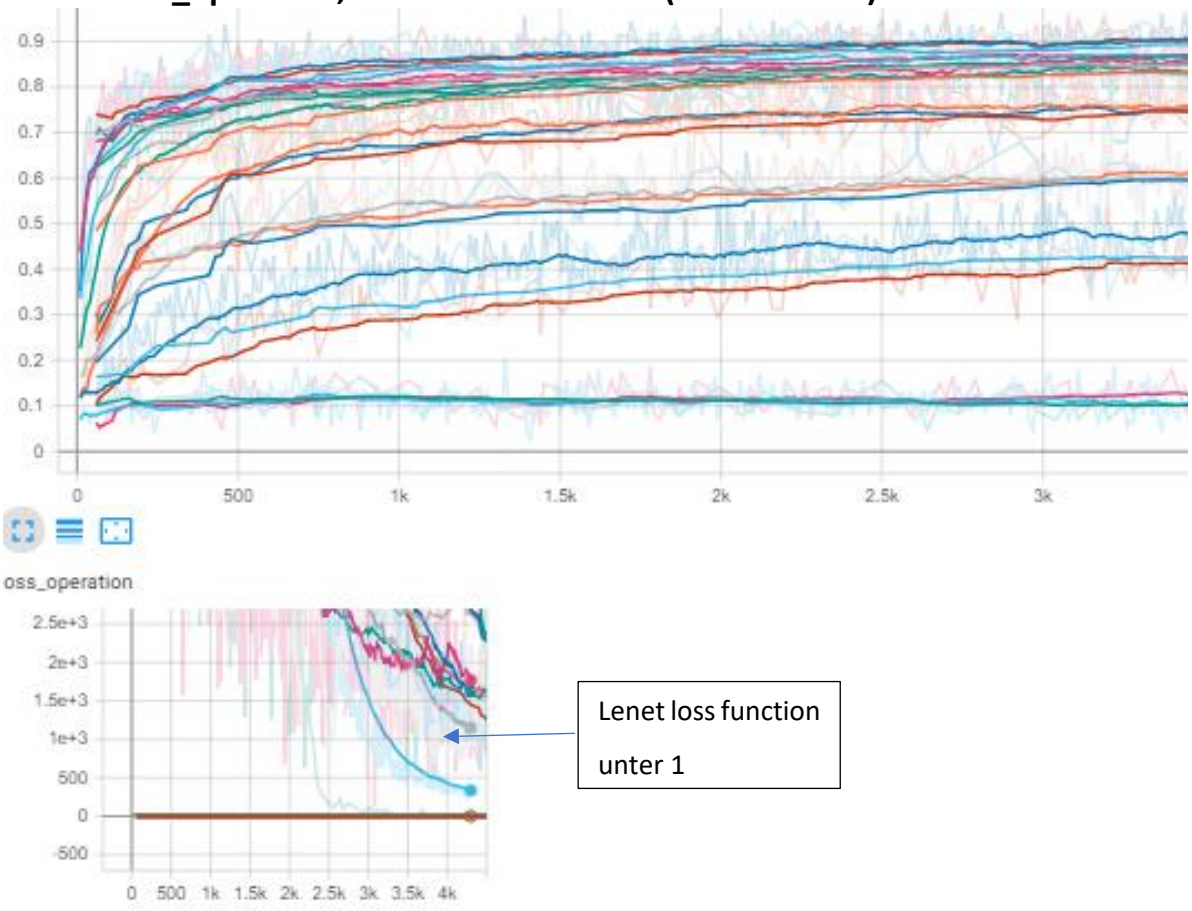


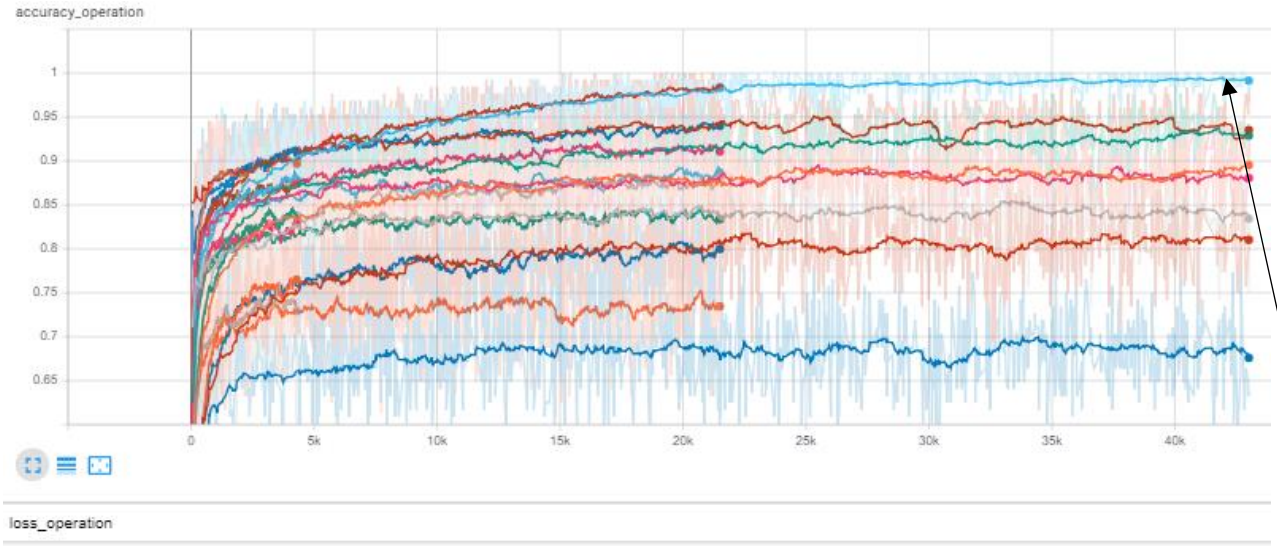
Abbildung 13 Modelle mit R 0.001

4.2 Loss-function mit schneller Konvergenz

Manche Lenet-Modelle zeigen eine schnelle Konvergenz der Loss-Funktion: bereits nach 2000 Iterationen erreichen sie einen Wert der Loss-Funktion <1. Epochen haben wenig Einfluss auf die loss-operation. Meist ist R0.001 besser als R0.01.

4.3 Performanz der beiden CNNs

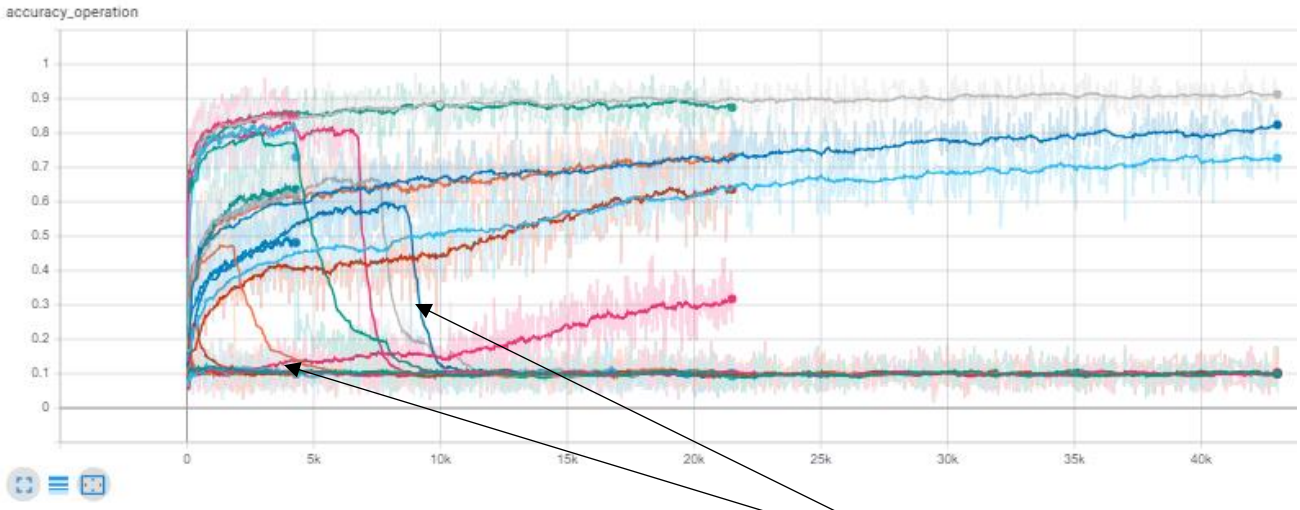
Lenet



	Name	Smoothed	Value	Step	Time	Relative
	lenet_E100_D0.25_R0.001	0.803	0.7891	21.12k	Wed Jun 12, 17:24:33	8m 4s
	lenet_E100_D0.25_R0.01	0.6887	0.6875	21.12k	Wed Jun 12, 16:55:16	8m 5s
	lenet_E100_D0.5_R0.001	0.8766	0.8984	21.12k	Wed Jun 12, 16:26:17	8m 0s
	lenet_E100_D0.5_R0.01	0.842	0.8828	21.12k	Wed Jun 12, 15:57:01	8m 7s
	lenet_E100_D0.75_R0.001	0.9119	0.8906	21.12k	Wed Jun 12, 15:27:11	8m 17s
	lenet_E100_D0.75_R0.01	0.8798	0.8828	21.12k	Wed Jun 12, 14:57:06	8m 21s
	lenet_E100_D1.0_R0.001	0.978	0.9922	21.12k	Wed Jun 12, 14:27:06	8m 5s
	lenet_E100_D1.0_R0.01	0.9374	0.9453	21.12k	Wed Jun 12, 13:58:51	7m 52s
	lenet_E10_D0.25_R0.001	0.7651	0.8295	4.3k	Wed Jun 12, 12:44:21	1m 8s
	lenet_E10_D0.25_R0.01	0.7291	0.6705	4.3k	Wed Jun 12, 12:43:11	1m 7s
	lenet_E10_D0.5_R0.001	0.8377	0.8864	4.3k	Wed Jun 12, 12:42:01	1m 7s
	lenet_E10_D0.5_R0.01	0.8291	0.875	4.3k	Wed Jun 12, 12:40:51	1m 9s
	lenet_E10_D0.75_R0.001	0.8783	0.8409	4.3k	Wed Jun 12, 12:39:41	1m 8s
	lenet_E10_D0.75_R0.01	0.8741	0.8864	4.3k	Wed Jun 12, 12:38:31	1m 7s
	lenet_E10_D1.0_R0.001	0.9098	0.8636	4.3k	Wed Jun 12, 12:37:21	1m 8s
	lenet_E10_D1.0_R0.01	0.8969	0.9205	4.3k	Wed Jun 12, 12:36:11	1m 8s
	lenet_E50_D0.25_R0.001	0.7933	0.8125	21.12k	Wed Jun 12, 13:50:39	7m 56s
	lenet_E50_D0.25_R0.01	0.7308	0.6328	21.12k	Wed Jun 12, 13:42:21	7m 56s
	lenet_E50_D0.5_R0.001	0.8704	0.8516	21.12k	Wed Jun 12, 13:34:04	7m 50s
	lenet_E50_D0.5_R0.01	0.8379	0.8281	21.12k	Wed Jun 12, 13:25:53	8m 13s
	lenet_E50_D0.75_R0.001	0.9169	0.8984	21.12k	Wed Jun 12, 13:17:18	8m 0s
	lenet_E50_D0.75_R0.01	0.8823	0.9141	21.12k	Wed Jun 12, 13:08:56	8m 4s
	lenet_E50_D1.0_R0.001	0.9801	0.9766	21.12k	Wed Jun 12, 13:00:31	8m 1s
	lenet_E50_D1.0_R0.01	0.9416	0.9375	21.12k	Wed Jun 12, 12:52:09	7m 45s

Bei diesem Lenet-Modell ist Overfitting anzunehmen: Die accuracy Funktion erreicht den Wert 1, d.h. in der Crossvalidierung kann das Modell jedes Bild vorhersagen.
(Lenet_E100_D1.0_R0.001)

AlexNet



Name	Smoothed	Value	Step	Time	Relative
alexnet_E100_D0.25_R0.001	0.1033	0.1016	19.98k	Thu Jun 13, 01:54:55	14m 6s
alexnet_E100_D0.25_R0.01	0.1014	0.04688	19.98k	Thu Jun 13, 01:11:02	14m 2s
alexnet_E100_D0.5_R0.001	0.6153	0.6406	19.98k	Thu Jun 13, 00:27:22	14m 2s
alexnet_E100_D0.5_R0.01	0.1015	0.07813	19.98k	Wed Jun 12, 23:43:56	14m 6s
alexnet_E100_D0.75_R0.001	0.7312	0.7344	19.98k	Wed Jun 12, 22:59:48	14m 9s
alexnet_E100_D0.75_R0.01	0.1083	0.1406	19.98k	Wed Jun 12, 22:15:59	14m 24s
alexnet_E100_D1.0_R0.001	0.8976	0.8672	19.98k	Wed Jun 12, 21:31:48	13m 57s
alexnet_E100_D1.0_R0.01	0.09695	0.1094	19.98k	Wed Jun 12, 20:49:30	13m 48s
alexnet_E10_D0.25_R0.001	0.1003	0.1023	4.3k	Wed Jun 12, 18:32:16	2m 31s
alexnet_E10_D0.25_R0.01	0.1053	0.07955	4.3k	Wed Jun 12, 18:29:43	2m 31s
alexnet_E10_D0.5_R0.001	0.4812	0.4886	4.3k	Wed Jun 12, 18:27:09	2m 31s
alexnet_E10_D0.5_R0.01	0.09905	0.07955	4.3k	Wed Jun 12, 18:24:35	2m 31s
alexnet_E10_D0.75_R0.001	0.6148	0.625	4.3k	Wed Jun 12, 18:22:00	2m 31s
alexnet_E10_D0.75_R0.01	0.6364	0.6705	4.3k	Wed Jun 12, 18:19:26	2m 31s
alexnet_E10_D1.0_R0.001	0.85	0.8523	4.3k	Wed Jun 12, 18:16:52	2m 31s
alexnet_E10_D1.0_R0.01	0.7298	0.1591	4.3k	Wed Jun 12, 18:14:18	2m 33s
alexnet_E50_D0.25_R0.001	0.2916	0.2188	19.96k	Wed Jun 12, 20:34:07	13m 55s
alexnet_E50_D0.25_R0.01	0.1099	0.09375	19.96k	Wed Jun 12, 20:18:38	14m 0s
alexnet_E50_D0.5_R0.001	0.617	0.5938	19.96k	Wed Jun 12, 20:02:59	13m 52s
alexnet_E50_D0.5_R0.01	0.09345	0.09375	19.96k	Wed Jun 12, 19:47:33	13m 53s
alexnet_E50_D0.75_R0.001	0.714	0.7813	19.96k	Wed Jun 12, 19:32:07	13m 51s
alexnet_E50_D0.75_R0.01	0.1021	0.125	19.96k	Wed Jun 12, 19:16:43	13m 52s
alexnet_E50_D1.0_R0.001	0.8844	0.8203	19.96k	Wed Jun 12, 19:01:19	13m 50s
alexnet_E50_D1.0_R0.01	0.1007	0.1172	19.96k	Wed Jun 12, 18:45:57	13m 36s

Relativ viele Alex-Modelle schmieren in der Accuracy mit zunehmenden Iterationen ab auf einen Wert von 0.1, der auch rein zufällig erreicht werden kann (10 Klassen). Die Modelle konnten die Bilder nicht fitten.

Einige AlexNet Modelle zeigen eine gute, wenn auch nicht so hervorragende Leistung, wie manche Lenet Modelle.

4.4 Accuracy Ergebnisse für den Test-Datensatz:

Tabelle 1 Testergebnisse (Accuracy) der einzelnen Modelle

Modell	test_accuracy
alexnet_E10_D1.0_R0.01	0.1688
alexnet_E10_D1.0_R0.001	0.8277
alexnet_E10_D0.75_R0.01	0.7382
alexnet_E10_D0.75_R0.001	0.7235
alexnet_E10_D0.5_R0.01	0.1
alexnet_E10_D0.5_R0.001	0.5909
alexnet_E10_D0.25_R0.01	0.1
alexnet_E10_D0.25_R0.001	0.1066
alexnet_E50_D1.0_R0.01	0.1
alexnet_E50_D1.0_R0.001	0.8107
alexnet_E50_D0.75_R0.01	0.1
alexnet_E50_D0.75_R0.001	0.6935
alexnet_E50_D0.5_R0.01	0.1
alexnet_E50_D0.5_R0.001	0.7257
alexnet_E50_D0.25_R0.01	0.1
alexnet_E50_D0.25_R0.001	0.3636
alexnet_E100_D1.0_R0.01	0.1
alexnet_E100_D1.0_R0.001	0.8634
alexnet_E100_D0.75_R0.01	0.1
alexnet_E100_D0.75_R0.001	0.8486
alexnet_E100_D0.5_R0.01	0.1
alexnet_E100_D0.5_R0.001	0.7751
alexnet_E100_D0.25_R0.01	0.1
alexnet_E100_D0.25_R0.001	0.1
lenet_E10_D1.0_R0.01	0.8885
lenet_E10_D1.0_R0.001	0.8876
lenet_E10_D0.75_R0.01	0.8808
lenet_E10_D0.75_R0.001	0.8922
lenet_E10_D0.5_R0.01	0.8436
lenet_E10_D0.5_R0.001	0.8662
lenet_E10_D0.25_R0.01	0.764
lenet_E10_D0.25_R0.001	0.8144
lenet_E50_D1.0_R0.01	0.8896
lenet_E50_D1.0_R0.001	0.9025
lenet_E50_D0.75_R0.01	0.8906
lenet_E50_D0.75_R0.001	0.9105
lenet_E50_D0.5_R0.01	0.8585
lenet_E50_D0.5_R0.001	0.8991
lenet_E50_D0.25_R0.01	0.7851
lenet_E50_D0.25_R0.001	0.8413
lenet_E100_D1.0_R0.01	0.8929
lenet_E100_D1.0_R0.001	0.8979
lenet_E100_D0.75_R0.01	0.89
lenet_E100_D0.75_R0.001	0.9107
lenet_E100_D0.5_R0.01	0.8607
lenet_E100_D0.5_R0.001	0.9006
lenet_E100_D0.25_R0.01	0.5604
lenet_E100_D0.25_R0.001	0.8505

Nach Weglassen aller Modelle, die mit dem Test-Datensatz weniger als 89% Test-accuracy erreicht haben, bleiben nur mehr Lenet Modelle übrig. Die besten Modelle haben eine Learning Rate von 0.001 und Epochen von 50 oder 100. Die Dropout Rate scheint weniger Einfluss zu haben. Ein starkes Overfitting konnte nur selten festgestellt werden. Anzeichen von Overfitting hatte lenet_E100_D0.1_R0.001: validation acc. 0.9922 (also fast 1), test acc. 0.8979.

Tabelle 2 Modelle mit Accuracy > 89%

Modell	test_accuracy	validation acc.
lenet_E100_D0.75_R0.01	0.89	0.8798
lenet_E50_D0.75_R0.01	0.8906	0.8823
lenet_E10_D0.75_R0.001	0.8922	0.8783
lenet_E100_D1.0_R0.01	0.8929	0.978
lenet_E100_D1.0_R0.001	0.8979	0.9922
lenet_E50_D0.5_R0.001	0.8991	0.8704
lenet_E100_D0.5_R0.001	0.9006	0.8766
lenet_E50_D1.0_R0.001	0.9025	0.9801
lenet_E50_D0.75_R0.001	0.9105	0.8783
lenet_E100_D0.75_R0.001	0.9107	0.9169

5 Diskussion

Alexnet gilt in der Bildvorhersage seit 2012 als guter Standard, wobei es definitiv seine Schwächen hat, die neuere und aktuellere Modelle (ImageNet) verbessern konnten. Doch sind diese Modelle viel komplexer und besitzen meist eine Vielzahl der Parameter welches Lenet oder AlexNet besitzt. Vor Alexnet war eine sinnvolle Klassifizierung von Bildern schwer möglich. Lenet wurde bereits 1998 entwickelt, um Handschriftzeichen zu klassifizieren, weshalb wir uns eine schlechtere Performanz von Lenet erwartet hatten. Ein möglicher Grund für das schlechtere Abschneiden von AlexNet könnten die vorgewichteten Filter im tiefen Netz sein. Die RGB-(red-green-blue) Input layer von AlexNet zur Verarbeitung der 3 Farbkanäle werden nicht als Ursache für das schlechte Abschneiden angesehen, weil die Graubilder zwar weniger Information als Farbbilder enthalten, dies die Input-layer aber nicht stören sollte und die Modellbildung davon (höchstwahrscheinlich) unabhängig ist.

Die besten deep Convolutional Neural Network-Modelle haben eine Test-Accuracy zur Klassifizierung der Zalando Bilder von 91%. Overfitting ist ein seltenes Problem für beide Modelle, wobei das Dropout wahrscheinlich das Overfitting unterbunden hat (siehe Lenet Modelle).

6 Analyse der Mängel der Arbeit und Ideen für die zukünftige Forschung

Mit den CNN-Architekturen Alexnet und Lenet aus dem Tensorflow-Framework haben wir die Aufgabe der Klassifizerung unseres Bilddatensatzes mit unseren Mitteln best möglich erfüllt. Dazu haben wir beide Netzwerke erstellt und die jeweiligen Ergebnisse miteinander verglichen. Was in unserer Arbeit fehlt sind die Ergebnisse weiterer Architekturen wie dem GoogleNet/Inception(2014), VGGNet (2014) und dem ResNet(2015). Wobei GoogleNet/Inception und VGGNet zu Weiterentwicklungen des klassischen CNN gelten, wohingehend Resnet auf einer grundlegend anderen Architektur aufbaut. Die Variation eines CNN-basierenden Modelle besteht auf unterschiedlicher Anzahl der Layer, Filter und dem Stapel der Convolutional Layer. Resnet als Vertreter eines Resiudal Neuronal Networks bestehen aus Layern, Batch-Normalization und haben die Möglichkeit innerhalb des Netztes Layer zu überspringen. Durch diese Vorgangsweise können. Durch diese Technik können tiefe Netzwerke erzeugt werden, dieses jedoch effizient verwendet.

Für unsere Seminarbeit wäre weiterführend der Vergleich mit der Perfromance eines RNN an unserem Datensatz interessant. Die Forschung wird sich auch weiterhin mit dem Thema beschäftigen die Performance der Modelle zu verbessern wobei zugleich die Modelle an Komplexität