Evaluierung von Transferlernen mit Deep Direct Cascade Networks

Simon Tarras

June 18, 2025

Contents

Chapter 1

Einführung

1.1 Setup

Alle Test wurden auf einem Erazer Gaming Notebook P15601 unter Windows 10 durchgeführt. Die Neuronalen Netze laufen dabei ausschließlich auf der CPU und wurden nur trainiert, wenn der Rechner am Stromnetz angeschlossen war. Dieser Rechner hat einen intel Core i5 der neunten Generation mit 4 Kernen auf 8 logischen Prozessoren. Die Betriebsgeschwindigkeit liegt bei 2,4-5,1 GHz und die RAM-Größe liegt bei 15,8 GB bei einer Geschwindigkeit von 2667 MHz.

Es wurde mit PyCharm und der library Keras programmiert. Die Texte sind mit BibTex erstellt worden und die Plots mit der MatPlotLib library.

Die genutzten Datensätze sind Modified National Institute of Standards and Technology Dataset (MNIST), Streetview House Numbers (SVHN), Boston Housing Prices (Boston) und California Hausing Prices (California).

Dabei sind MNIST und Boston die Sourcedaten und SVHN und California die Targetdaten. Jeder Targetdatensatz wird händisch verkleinert, da es darum geht, nicht genügend Daten für sie allein zu haben und eine andere Methode genutzt werden muss.

1.2 Durchführungen

Es wurden sowohl für Klassifikation als auch für Regression drei verschiedene Ansätze der Kaskadierung genutzt. Ebenso wurde Direct TF mit Domainwechsel durchgeführt.

Die drei Ansätze sind Deep Cascade, Direct Cascade und eine Kaskadierung von einem Netz im Netz mit mehreren Inputs.

Bei Deep Cascade wird ein Netz Layer für Layer aufgebaut und jedes Layer einzelnd trainiert und gefreezt. Bei Direct Cascade werden ganze Netze trainiert und dessen Prediction als zusätzlichen Input für das nächste Netz zu nutzen. Bei der dritten Variante wird ein Netz trainiert, dann auf einem Teilnetz davon die Prediction gemacht, um mit dieser das ganze Netz außer das vorher erwähnte

Teilnetz zu trainieren. Nur Deep Cascade wird genauer betrachtet, denn die beiden anderen Ansätze sind nur zum Vergleichen da.

Es wurde bei jedem gleichbleibende Epochenanzahlen, zufällige und von einer Metrik abhängige durchgeführt.

Bei allen Neuronalen Netzen wurden die dafür benötigten Daten in ein Trainings-, Validation- und Testdatensatz aufgeteilt. Ebenfalls wurde MNIST auf 32x32 erweitert, sowie SVHN in graue Bilder mit einem Channel verändert. Es wurde erweitert, da keine Daten unnötig verloren werden sollten. Die Reduzierung von SVHN liegt daran, dass MNIST nur Schwarz-Weiß-Bilder sind und es nicht möglich ist, dies in bunte Bilder zu verändern. Die Veränderungen der Datensätze kommt daher, dass sie technisch gleich aussehen müssen, da sie sonst nicht als Input desselben Netzes genutzt werden können.

Für die Regressionsnetze müssen alle Spalten weggenommen werden, die kein Gegenüber im anderen Datensatz besitzen. Somit fielen die Spalten: Verbrechensrate, Anteil der Wohngebiete über 25000 Fuß, Nicht-Einzelhandelanteil der Gewerbeflächen, Flussgrundstück, Stickoxidkonzentration, Entfernung zu Arbeitsvermittlungszentren, Erreichbarkeit von Autobahnen, Vollwertsteuersatz, Schüler-Lehrer-Verhältnis und die Anzahl von Schwarzen im Bosten weg, während im California die folgenden Spalten wegfielen: Längengrad, Breitengrad, Schlafzimmer und Bevölkerung. Aus der Gesamtanzahl der Räume und der Haushalte wird die durchschnittliche Anzahl an Räumen pro Wohnung errechnet. Diese Spalten haben alle keinen Gegenüber im anderen Datensatz und eine Spalte ist aus ethnischen Gründen nicht nutzbar, was daran liegt, dass der Datensatz aus den Siebzigern stammt. Übrig blieben von Boston nur noch die durchschnittliche Anzahl der Räume pro Wohnung, die Menge der Häuser, die vor 1940 erbaut worden sind und der prozentuale Anteil der Bevölkerung mit niedrigem Status. Bei California blieben das Errechnete und das Hausalter, sowie das durchschnittliche Einkommen. Die Anzahl der Räume pro Wohnung passen offensichtlich zueinander, während der prozentuale Anteil der Bevölkerung mit niedrigem Status antiproportional zu dem durchschnittlichen Einkommen ist. Dies wird vorher zu einer Proportionalität umgewandelt. Als etwas komplizierter erweist sich das Alter. Mit Prozentrechnung kann man aber das ungefäre Alter der Häuser aus dem Boston Datensatz abschätzen. Da immer eine Häuseranzahl von einhundert betrachtet wird, ist dies die Gesamtmenge und folgende Formel löst das Problem:

$$\frac{Hausanzahl * Hausalter}{Gesamtmenge} \tag{1.1}$$

Die Hausanzahl ist hier die Menge der Häuser, die vor 1940 erbaut worden sind. Das Hausalter bezieht sich auf das Alter der eben erwähnten Häuser und ist auf Heute angedacht; sind also 85 Jahre.

Die Hypothese war, dass man mit TF bei zu wenig Daten eine verhältnismäßig gute Performanz der Netze erwarten kann, sowie, dass durch einen Kaskadierungsansatz das Training der Netze sehr kurz ist.

Generell wird zuerst eine Weile auf dem Sourcedatensatz trainiert und dann auf den Targetdatensatz gewechselt ohne die bisherigen Netze zu verändern. Bis auf den Direct Cascade Ansatz werden auch die Inputs während des ganzen

Prozesses nicht verändert. Bei Direct Cascade werden die Inputs immer größer, denn die Prediction des vorherigen Netzes wird zum Input hinzugefügt.

1.3 Metrik

Es wurden drei Metriken erstellt. Die Accuracy- (ACCM), Loss- (LM) und MAE-Metrik (MAEM). MAE heißt dabei Mean absolute Error. Alle drei Metriken sind für Early Stopping und entscheiden, wieviele Epochen genutzt werden. Die Accuracy-Metrik bricht immer dann ab, wenn die Validation-Accuracy mindestens um 10% schlechter ist als die Trainingsaccuracy, da dann in dem Netzwerk Overfitting herrscht.

Die Loss- und die MAE-Metrik brechen beide dann ab, wenn der Validation-Wert der aktuellen Epoche schlechter ist als in der Epoche davor. Dies hat zur Folge, dass die Netze in lokale Minima hineinlaufen und nicht wieder herauskommen. Dabei unterliegt die Anzahl der Netze für das Direct Cascade keiner Metrik.

Chapter 2

Klassifikation

2.1 Filternetze

Die Klassifikation über Filternetze im Direct Cascade Ansatz. Es wird zudem zwischen eindimensionalen und zweidimensionalen Bilddaten unterschieden, um zu zeigen, dass die Dimensionalität für die Accuracy irrelevant ist und es im zweidimensionalen nur länger dauert.

Der zweidimensionale Fall hat folgende Updateregel: Die Prediction wird ausgelesen und dessen Wert wird in ein Array mit demselben Shape hineingeschrieben und dies wird dann mit den Trainingsdaten auf der Channelachse konkateniert. Dies ergibt den Augmented Vector, der als Input für das nächste Netz genutzt wird.

Die Besonderheit des zweidimensionalen Netzes ist es, dass es in der Updateregel sehr viel Arbeitsspeicher benötigt wird. Deshalb wird nicht nur der Targetdatensatz auf 1% verkleinert, sondern auch der Sourcedatensatz.

Es wurde einmal ohne Metrik, einmal mit der Accuracy-Metrik und einmal mit der Loss-Letrik ausgetestet.

Die Figure 2.1 zeigt die Plots der Tests zuerst ohne eine Metrik, dann mit der ACCM und zum Schluss mit der LM. Da der Sourcedatensatz verringert wurde, ist die Accuracy im ersten Bereich geringer als es erwartbar ist. Das Early Stopping ist hier nicht erkennbar, da die Berechnungszeit des Augmented Vector hier für die Zeit entscheidend ist.

Im eindimensionalen Fall gibt es folgende Updateregel: Die Bilddaten werden zuerst in eindimensionale Bilder verändert mit Channels. Dies ist der Input des Netzes. Die Prediction und der Netzinput, dessen Channelachse vorübergehend entfernt wird, werden direkt konkateniert und das Ergebnis, um die Channelachse erweitert.

Hier wird nur der Targetdatensatz verkleinert und einem ohne Metrik, mit der ACCM und der LM ausgetestet. Dies zeigt die Figure 2.2.

Hier werden die Metriken eindeutig gesehen, denn die Trainingszeit ist sehr verschieden. Es wird auch klar, dass der eindimensionale Fall schlechter ist als

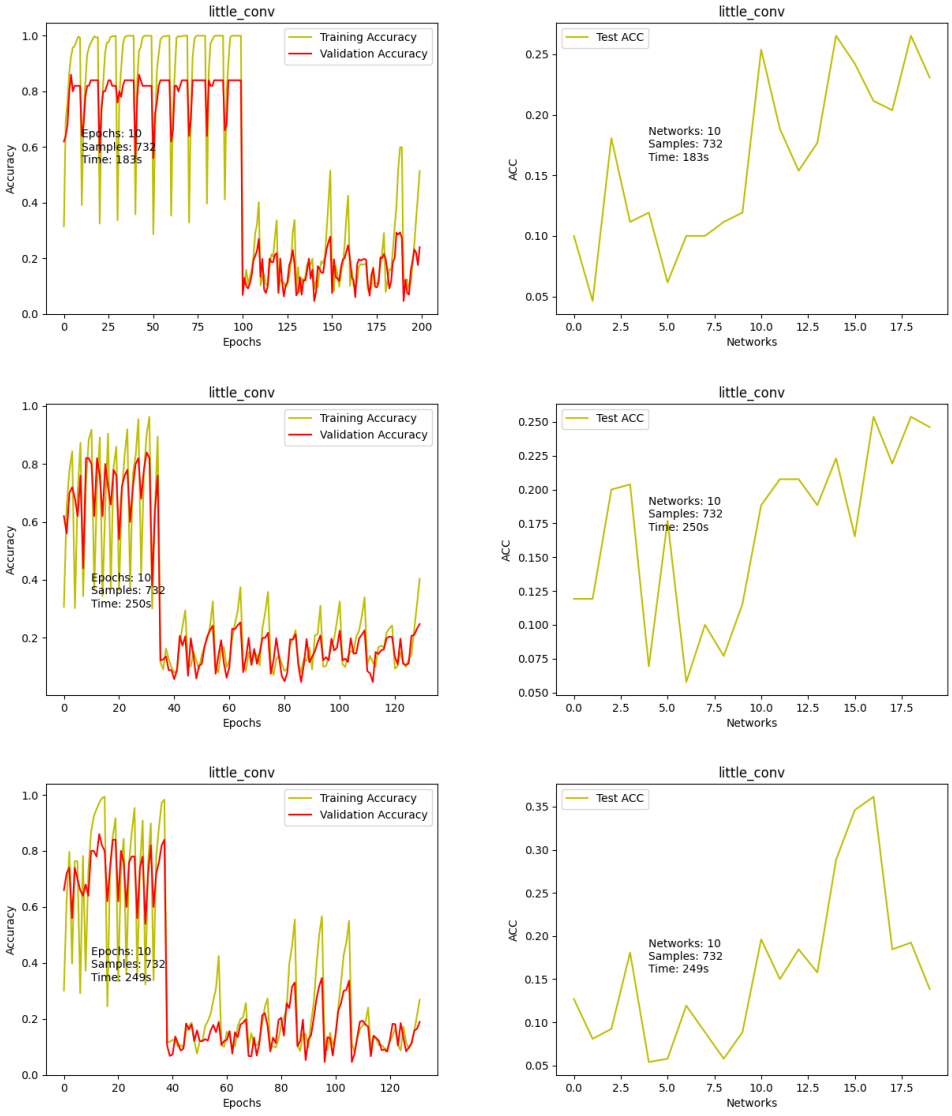
little_conv
Training Accuracy
Validation Accuracy
Epochs: 10
Samples: 732
Time: 183s
Accuracy
Epochs
little_conv
Test ACC
Networks: 10
Samples: 732
Time: 183s
ACC
Networks
little_conv
Training Accuracy
Validation Accuracy
Epochs: 10
Samples: 732
Time: 250s
Accuracy
Epochs
little_conv
Test ACC
Networks: 10
Samples: 732
Time: 250s
ACC
Networks
little_conv
Training Accuracy
Validation Accuracy
Epochs: 10
Samples: 732
Time: 249s
Accuracy
Epochs
little_conv
Test ACC
Networks: 10
Samples: 732
Time: 249s
ACC
Networks

Figure 2.1

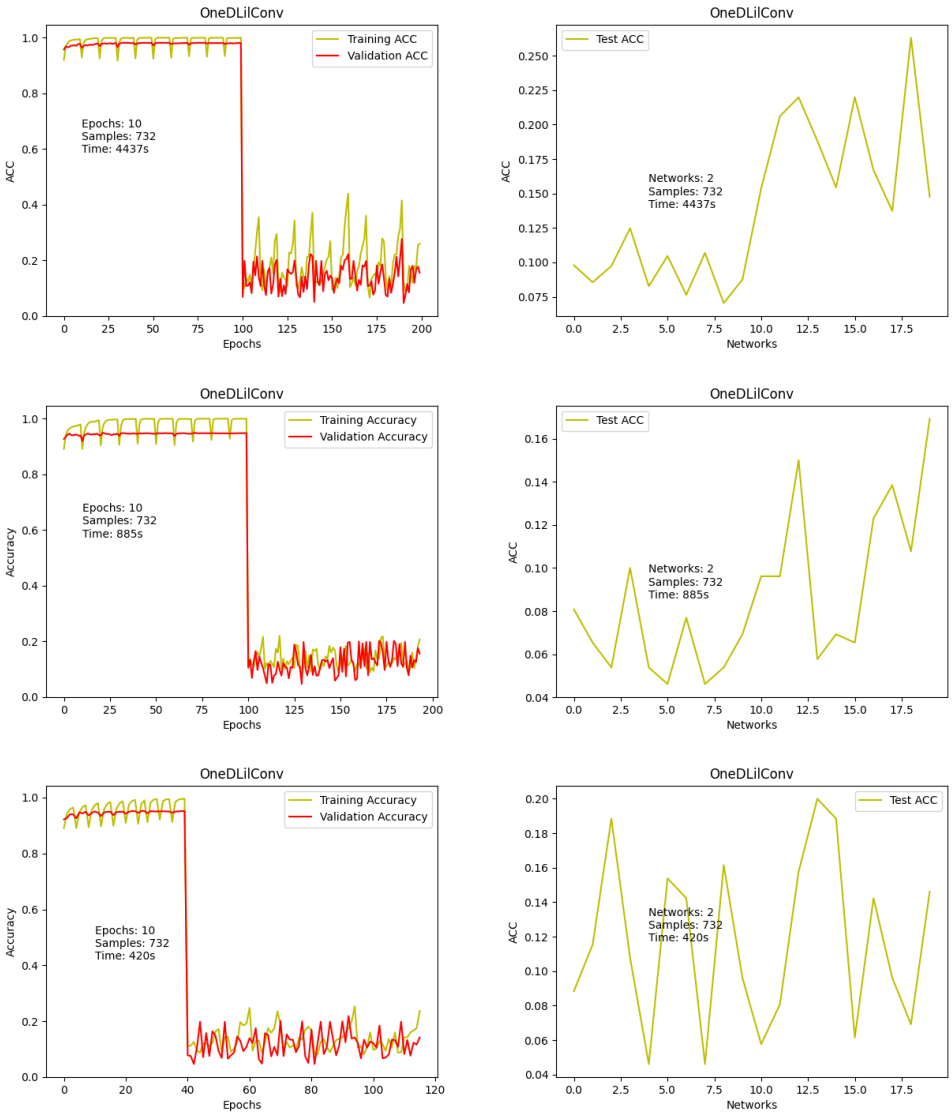
OneDLilConv
Training ACC
Validation ACC
Epochs: 10
Samples: 732
Time: 4437s
ACC
Epochs
OneDLilConv
Test ACC
Networks: 2
Samples: 732
Time: 4437s
ACC
Networks
OneDLilConv
Training Accuracy
Validation Accuracy
Epochs: 10
Samples: 732
Time: 885s
Accuracy
Epochs
OneDLilConv
Test ACC
Networks: 2
Samples: 732
Time: 885s
ACC
Networks
OneDLilConv
Training Accuracy
Validation Accuracy
Epochs: 10
Samples: 732
Time: 420s
Accuracy
Epochs
OneDLilConv
Test ACC
Networks: 2
Samples: 732
Time: 420s
ACC
Networks

Figure 2.2

der Zweidimensionale. Dies liegt daran, dass im eindimensionalen nur die Daten rechts und links von dem betrachteten Pixel in die Berechnung mit eingezogen werden können, während im zweidimensionalen zusätzlich auch die Daten oben, unten und die vier Ecken jenes Kreuzes betrachtet werden. Beide Netze haben eine sehr schlechte Accuracy. Dies liegt aber nicht daran, dass auf dem Sourcedatensatz Overfitting passiert ist, denn es wird nicht besser, wenn der Wechsel der Datensätze beliebig nach vorne geschoben wird.

2.2 Linearnetze

Die Klassifikation über Linearnetze mittels eines Direct Cascade Ansatzes.

Das Netz hat ein Linearlayer mit 512 Nodes und der ReLU-Activation function.

Für den augmented Vector werden alle inputdaten in eindimensionale Bilder verwandelt. Der Input wird mit der Prediction konkateniert und direkt an das nächste Netz weitergegeben.

Dieses Netz wurde ohne Metrik, mit der ACCM und LM ausgetestet.

Die Figure 2.3 zeigt, dass ein Netzwerk mit nur Linearlayers etwas schlechter ist als ein zweidimensionales Filternetz. Hier werden die Metriken auch bereits gesehen, aber sie bringen kein erhofftes Ergebnis.

Die Klassifikation funktioniert mit den Updateregeln und diesen Netzen nicht. Auch TF bringt dabei nichts. Das einzige, was halbwegs etwas bringt, ist, wenn mehr Targetdaten benutzt werden. Aber dann wird TF auch nicht mehr gebraucht.

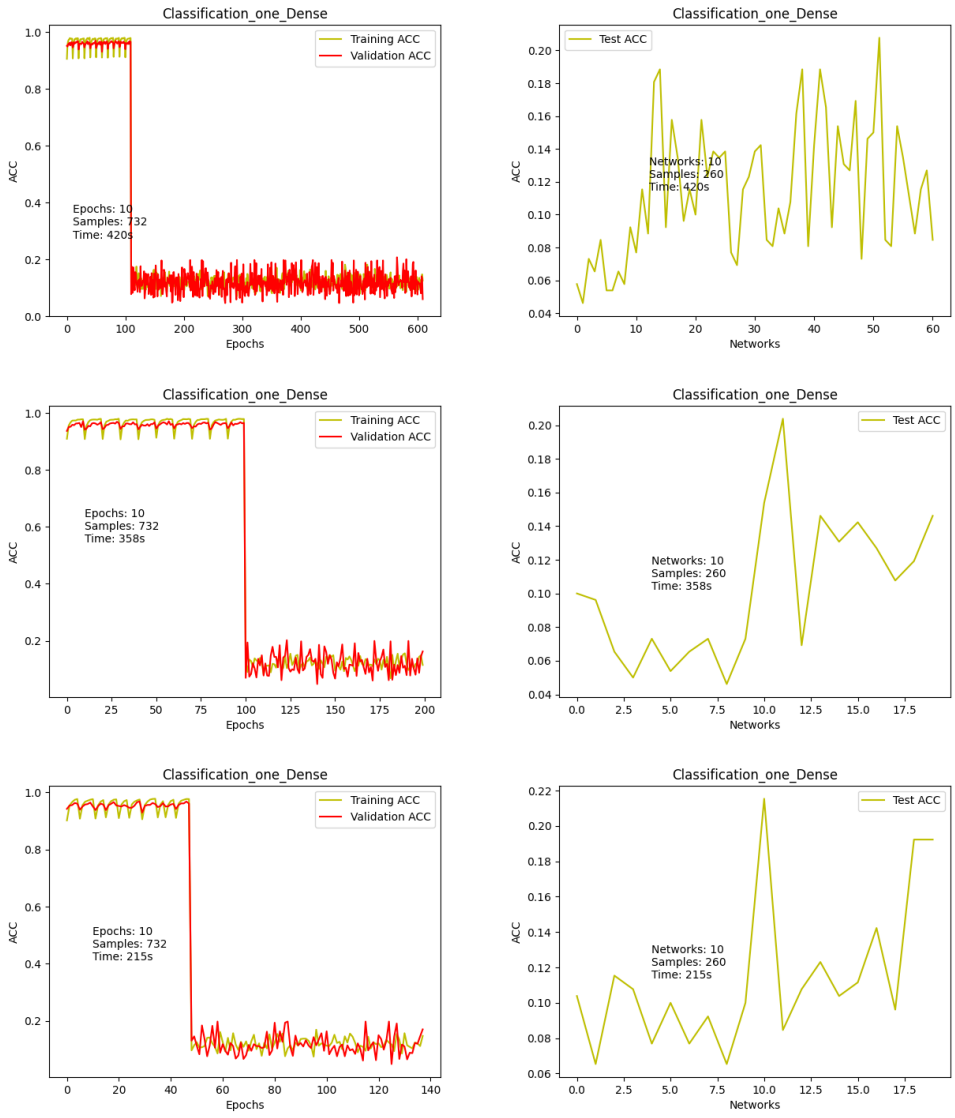
Classification_one_Dense
Training ACC
Validation ACC
Epochs: 10
Samples: 732
Time: 420s
ACC
Epochs
Classification_one_Dense
Test ACC
Networks: 10
Samples: 260
Time: 420s
ACC
Networks
Classification_one_Dense
Training ACC
Validation ACC
Epochs: 10
Samples: 732
Time: 358s
ACC
Epochs
Classification_one_Dense
Test ACC
Networks: 10
Samples: 260
Time: 358s
ACC
Networks
Classification_one_Dense
Training ACC
Validation ACC
Epochs: 10
Samples: 732
Time: 215s
ACC
Epochs
Classification_one_Dense
Test ACC
Networks: 10
Samples: 260
Time: 215s
ACC
Networks

Figure 2.3

Chapter 3

Regression