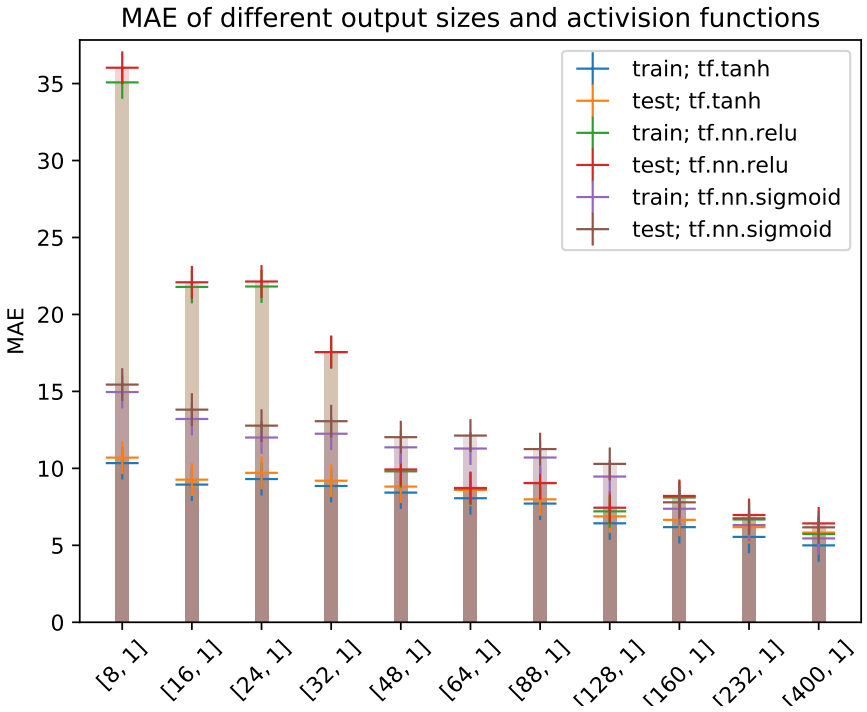
Kleine Zusammenfassung (08.07.19)

Fuer jedes Training wurden die folgenden Hyperparameter gewaehlt:

Batch Size	25
Learning Rate	0.001
Iterationsschritte	400.000

Es wurde der QM7 Datensatz als lokalisierte Coubombmatrix kodiert und mit einem sonnet Netz trainiert. Dabei wurden immer regularizers verwendet. Zunaechst mit nur einem hidden layer mit unterschiedlichen activation functions und unterschiedlicher Anzahl nodes im layer. Das Ergebnis sah wie folgt aus:



Man sieht, dass der tanh immer am besten abgeschnitten hat. Also wurde ab jetzt nurnoch der tanh als activation function genutzt. Anschliessnd wurde an der Architektur des Netzes und der Anzahl der hidden layer experimentiert. Die Ergebnisse davon sind in Tabelle 1 zu sehen.

NN size	MAE test	MAE train	diff	time [min]
[4, 1]	12.292	12.192	0.1	2.65
[8, 1]	10.849	10.341	0.508	2.35
[16, 1]	8.935	8.499	0.436	2.35
[32, 1]	9.118	8.657	0.461	3.53
[64, 1]	8.962	8.233	0.729	4.96
[128, 1]	7.741	6.95	0.791	6.01
[4, 4, 1]	12.603	12.39	0.213	3.14
[8, 4, 1]	11.064	10.758	0.306	2.96
[8, 8, 1]	10.395	10.085	0.309	2.94
[16, 4, 1]	9.18	8.558	0.622	3.07
[16, 8, 1]	10.177	9.5	0.678	3.04
[16, 16, 1]	7.872	7.13	0.741	3.13
[32, 4, 1]	7.038	6.509	0.529	4.28
[32, 8, 1]	6.884	5.839	1.045	4.25
[32, 16, 1]	6.411	5.508	0.903	4.16
[32, 32, 1]	6.337	5.484	0.853	4.89
[64, 4, 1]	179.143	178.974	0.169	5.75
[64, 8, 1]	5.779	4.94	0.839	5.9
[64, 16, 1]	5.621	4.716	0.906	6.02
[64, 32, 1]	5.219	4.024	1.194	6.33
[64, 64, 1]	4.989	3.334	1.654	7.54
[4, 4, 4, 1]	14.047	14.231	0.183	3.66
[8, 4, 4, 1]	12.356	12.148	0.208	3.3
[8, 8, 4, 1]	10.668	10.191	0.477	3.37
[8, 8, 8, 1]	10.927	10.405	0.522	3.64
[16, 4, 4, 1]	9.136	8.545	0.591	3.51
[16, 8, 4, 1]	8.218	7.4	0.817	3.84
[16, 8, 8, 1]	7.208	6.658	0.55	3.76
[16, 16, 4, 1]	8.117	7.467	0.65	3.89
[16, 16, 8, 1]	7.853	7.221	0.631	4.06
[16, 16, 16, 1]	7.459	6.676	0.783	3.95
[32, 4, 4, 1]	7.288	6.576	0.712	4.63
[32, 8, 4, 1]	7.634	7.004	0.629	4.86
[32, 8, 8, 1]	6.091	5.233	0.859	4.91
[32, 16, 4, 1]	6.18	5.319	0.861	5.05
[32, 16, 8, 1]	5.828	4.804	1.024	5.09
[32, 16, 16, 1]	5.574	4.583	0.992	5.17
[32, 32, 4, 1]	179.145	178.986	0.159	5.56
[32, 32, 8, 1]	5.586	4.427	1.159	5.57
[32, 32, 16, 1]	5.369	4.522	0.848	5.76
[32, 32, 32, 1]	5.013	3.709	1.304	6.23
[4, 4, 4, 4, 1]	12.391	12.52	0.129	4
[8, 4, 4, 4, 1]	11.488	11.097	0.39	3.79
[8, 8, 4, 4, 1]	11.058	10.906	0.153	3.9
[8, 8, 8, 4, 1]	11.007	10.378	0.629	3.99
[8, 8, 8, 8, 1]	9.902	9.439	0.463	4.12
[16, 4, 4, 4, 1]	9.059	8.634	0.425	4
[16, 8, 4, 4, 1]	7.704	7.095	0.609	4.05
[16, 8, 8, 4, 1]	8.939	8.293	0.646	4.27
[16, 8, 8, 8, 1]	7.722	7.099	0.623	4.34
[16, 16, 4, 4, 1]	9.023	8.214	0.809	4.12
[16, 16, 8, 4, 1]	7.143	6.266	0.877	4.62
[16, 16, 8, 8, 1]	6.767	6.101	0.666	4.58
[16, 16, 16, 4, 1]	6.569	5.758	0.811	4.59
[16, 16, 16, 8, 1]	8.278	7.315	0.963	4.67
[16, 16, 16, 16, 1]	6.853	5.906	0.947	4.68

Table 1: Ergebnisse des Trainings

Die Ergebnisse sind alle nicht im Bereich von 1 kcal. Ich vermute es liegt daran, dass bei der Uebergabe der Coulombmatrix in das Netz nicht wirklich convolution passiert, da man direkt die gesamte 23*23 Matrix uebergibt und somit keine lokalisierten Eigenschaften.

Moegliche Verbesserungen:

- Nicht auf der ganzen Matrix sondern den einzelnen Zeilen trainieren, oder besser (naechster Punkt)
- Kodierung als Graph mit der Graph Nets Library um zusaetzlich praezise updatefunctions zu waehlen
- Optimierte Wahl der Hyperparameter. Das gilt auch fuer die regularizers.
- Besseres Abbruchkriterium fuer das Training. Hier hab ich nur nach Auganmass entschieden, dass ab ca. 400.000 sich nicht mehr viel aendert. Siehe Abblidung 1 (auf der lezten Seite). Hier wurden 800.000 Schritte gemacht

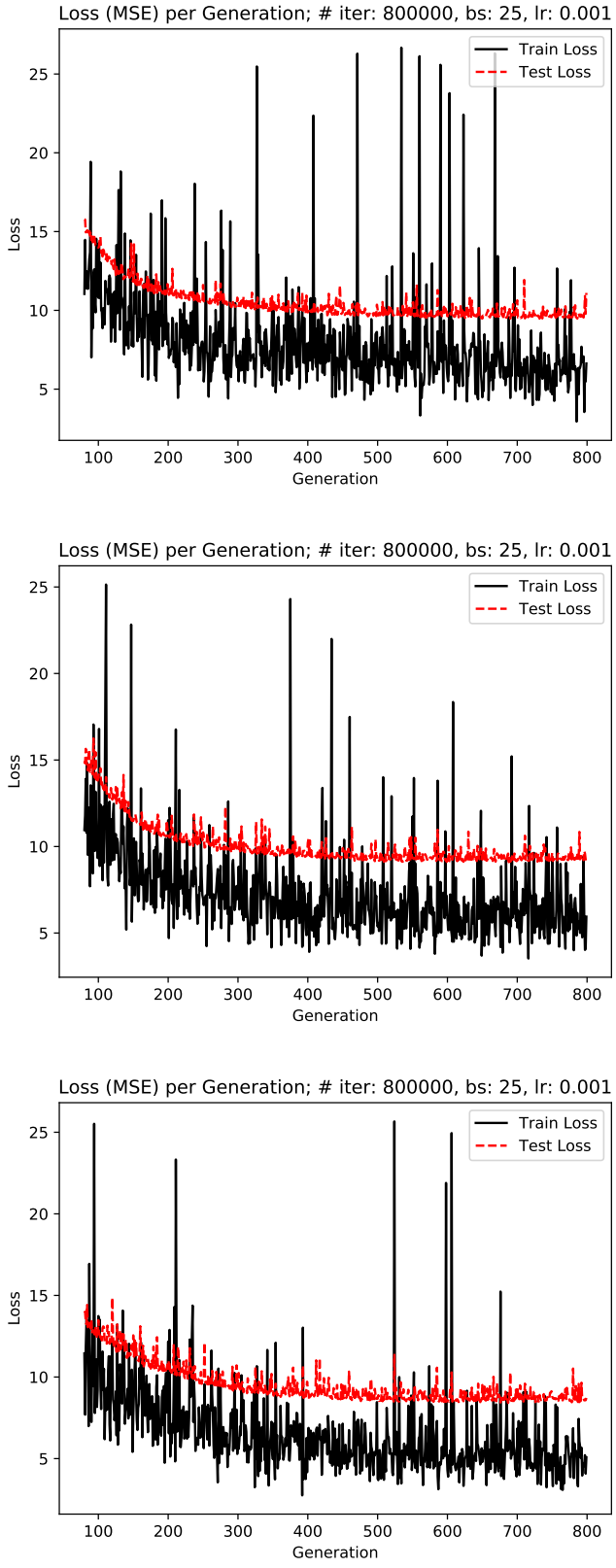


Figure 1: Loss gegen Generation. Eine Generation entsprechen hier 1000 Schritte