Einführung ins Deep Learning

deep space computing AG

6. Dezember 2019

deep-space.ch

Mission/Rechnerverbund

Deep Space Computing

- Mission: Die Erstellung von Berechnungssystemen und Anwendungen
- Dienstleistungen: Software-Beratung und Entwicklung, Aufsetzen von Clusters

Die Firma betreibt einen Rechencluster mit folgenden Spezifikationen:

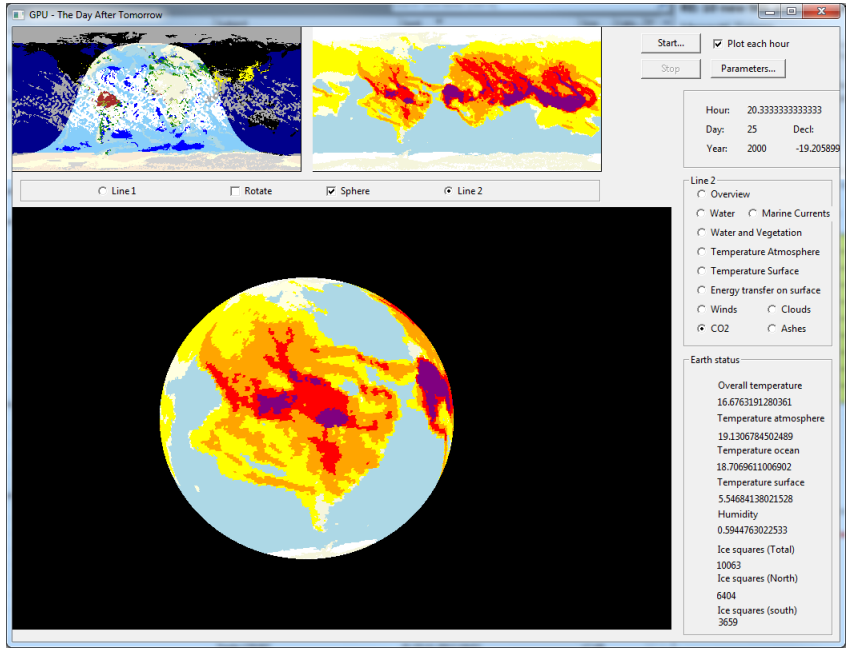
PCs	33
Mini PCs	11
CPUs	328
Grafikkarten	44
CUDA-Kerne	47'000
Tensor-Kerne	288
Betriebssysetm	Ubuntu Linux 18.04 / Windows 10
Entropiequellen	eine aus thermischer Halbleiterverbindung und eine aus Rauschen der Ionosphäre
Andere Hardware	eine Xilinx FPGA
Kühlung	3 Lüfter, 1 Dyson
Umwelt Monitoring	Homematic System: unter anderem Erkennung von Rauch, Blitze und Monitoring des Verbrauches
In Betrieb seit	01.07.2017
Stromverbrauch	7 kW

Etwa die Hälfte der Rechner im Rechnerverbund sind CUDA Superrechner .

~100 Teraflops x 2000 Summit (USA) 200 Petaflops

Prof. Dan Jacobson, 2 Hexaflops using Tensor Cores in Genomics Deep Learning Code -> Gordon Bell Award 2018

Inhalt



- Was ist Deep Learning?
- Arten von Optimierungsprobleme
- Evolution der Hardware
- Arten von neuronalen Netze
- Anwendungen von Deep Learning im Energiebereich
  - Beispiel: Voraussage der Stromnachfrage
  - Beispiel: TSP auf Quantenrechner (z.B. Techniker, die Stromzähler der Stadt auslesen und mit Smart-Meter ersetzen)

Was ist Deep Learning?

Was ist Deep Learning

- Deep Learning ist die moderne Ausführung des maschinellen Lernes und vereint Methoden wie zum Beispiel neuronale Netze, Support Vector Machines und Entscheidungsbäume, um Klassifikation- und Regressionsaufgaben zu bewältigen
- Neuronale Netze waren bereits in den 1960-er Jahre im Einsatz, die Evolution der Hardware und der Trainingsmethoden macht ab 2006 möglich, neuronale Netze mit viel mehr Schichten zu trainieren.
- Die heutigen Erfolge der Künstlichen Intelligenz im Bereich Spracherkennung (z.B. Alexa) und Bilderkennung (z.B. selbstfahrende Auto, wiederverwendbare Raketen, Sprachübersetzer) sind dem Deep Learning zu verdanken.

Arten von Optimierungsprobleme

Optimierungsproblem	**Linear**	**Konvex**	**Extrem nicht linear**
Beispiele	Kraftwerksoptimierung, Inventory Management, Scheduling	Spracherkennung und Sprachübersetzung, Maschinelles Lernen, Künstliche Intelligenz	Pharmacodynamics, Differentialgleichungsysteme
Löser	IBM CPLEX, glpk	Keras, Tensorflow	(mu, lambda)-CCMA-ES, Particle Swarm Optimization
Bereich	Lineare Programmierung	Konvexe Programmierung	Nichtlineare Programmierung
Zielfunktion	Linear	Konvex	Nicht linear
Algorithmus	Simplex	Gradient Descent	Sample based

Deep Learning

Slide aus PASC19-Konferenz, Daniel Wälchli, ETH

Evolution der Hardware

- Die physische CPU *(AMD Ryzen, 32)*
- Die logische CPU *(Hyperthreading, +50%)*
- Die Single Instruction Multiple Data Architektur *(Intel SSE)*
- Die Grafikkarte *(Render Units auf Vektor-Projektion spezialisiert)*
- Field Programmable Gate Arrays (FPGA), *(Inferenz von Neuronalen Netzen)*
- Die GPGPU (Global Purpose Graphics Processing Unit) *ab 2006*
- CUDA Cores, SIMD auf Steroiden *(Nvidia GTX, bis zu 4000)*
- Tensor Cores, atomare Matrixoperationen *(Nvidia RTX, bis zu 500)*
- Quantum Computing (Dwave 2000Q)

Arten von neuronalen Netze

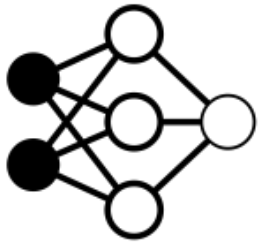
- Feed Forward *(Multilayer Perzeptron)*
- Recurrent Network *(Long Short Term Memory, Sprachübersetzung)*
- Convolutional Network *(Bilderkennung, Spracherkennung)*
- Autoencoder *(Input=Output, Anomalieerkennung)*
- Encoder – Decoder *(n Encoder, n Decoder, nxn Sprachübersetzungen)*
- Selbstbildende neuronale Netze *(NEAT, CPPN)*
- Neuronale Turing Machine

Anwendungen vom Deep Learning im Energiebereich

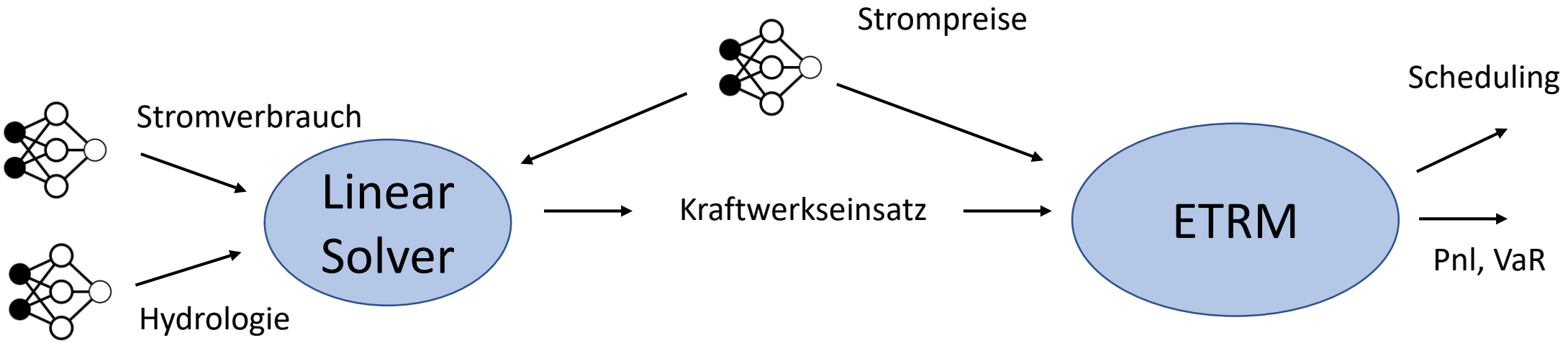
Deep Space Computing

Kurzfristige Voraussage:

- des Stromverbrauches
- der HPFC Preise für die nächste Woche
- der Hydrologie



Verwenden der obigen Inputs in einem Kraftwerksplanoptimierer und in der Portfolioevaluation

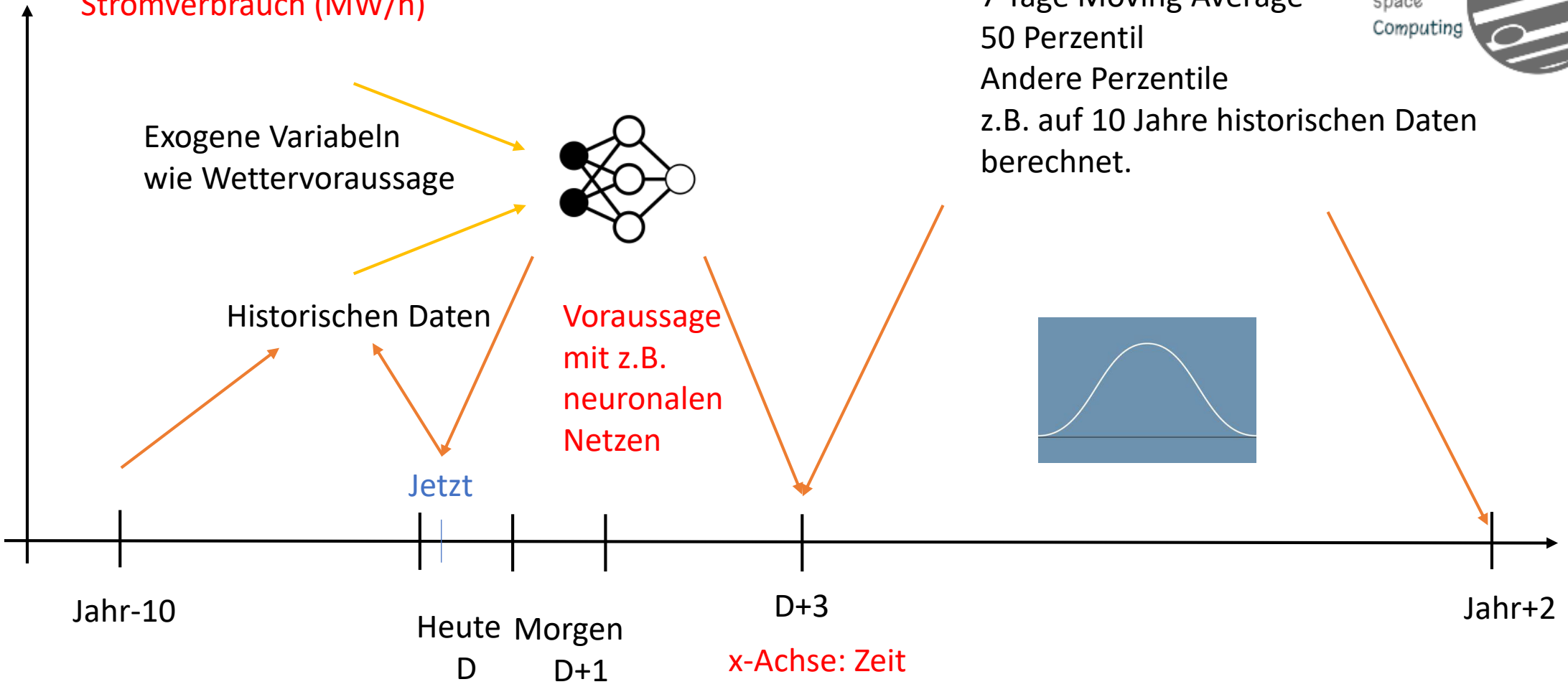


Voraussage und Statistik zusammenführen

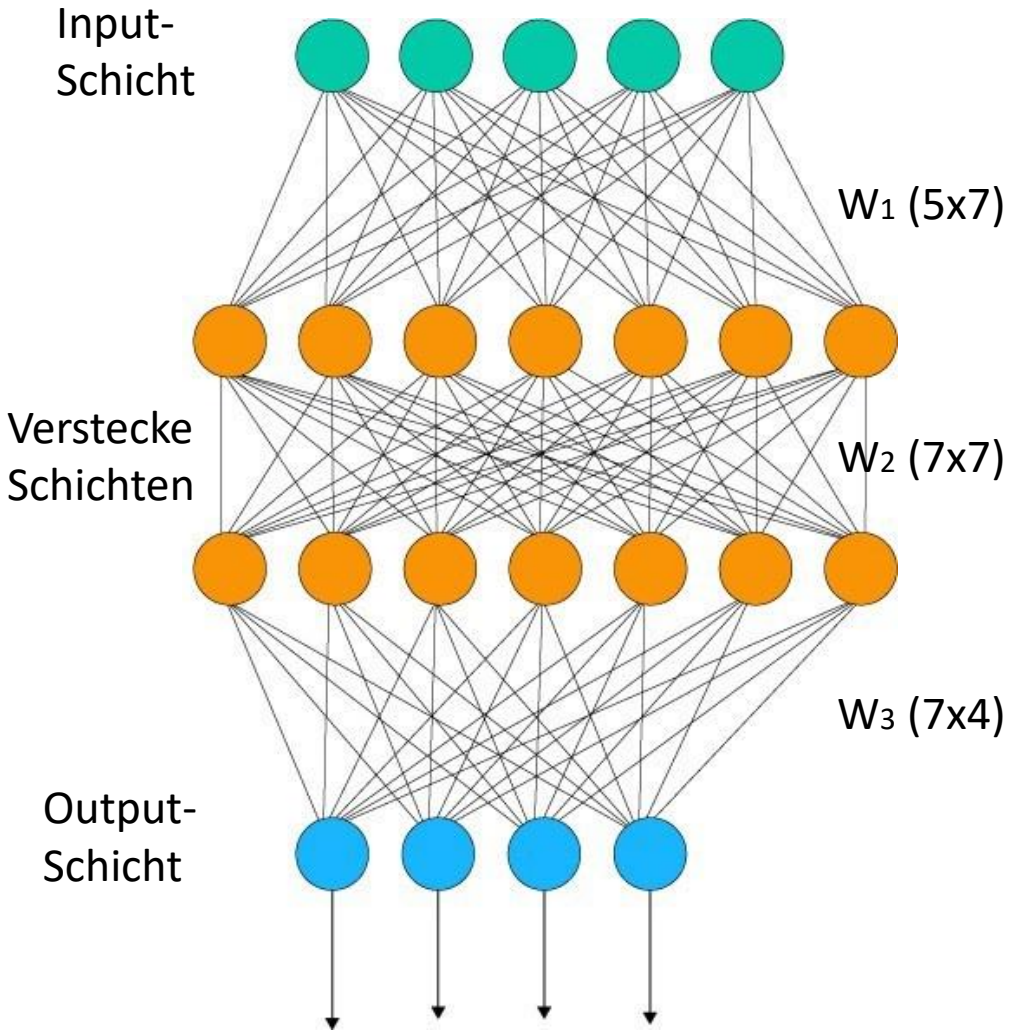
y-Achse:
Hydrologie (m^3/s)
Stromverbrauch (MW/h)

Statistische Daten:
7 Tage Moving Average
50 Perzentil
Andere Perzentile
z.B. auf 10 Jahre historischen Daten berechnet.

Deep Space Computing



Das neuronale Netz



- W_i ist eine Gewichtungsmatrix. In jeder Spalte sind die Gewichte eines einzelnen Neurons. Am Anfang vor dem Training sind die Gewichte zufällig
- alle Vektoren waagrecht, Verzicht auf ^{T} im folgenden
- x ist der Inputvektor (1x5),
- y* ist der gewünschte Outputvektor (1x4)
- φ ist die nichtlineare Aktivierungsfunktion

Das neuronale Netz ist eine nichtlineare Abbildung zwischen Input x und Output y:

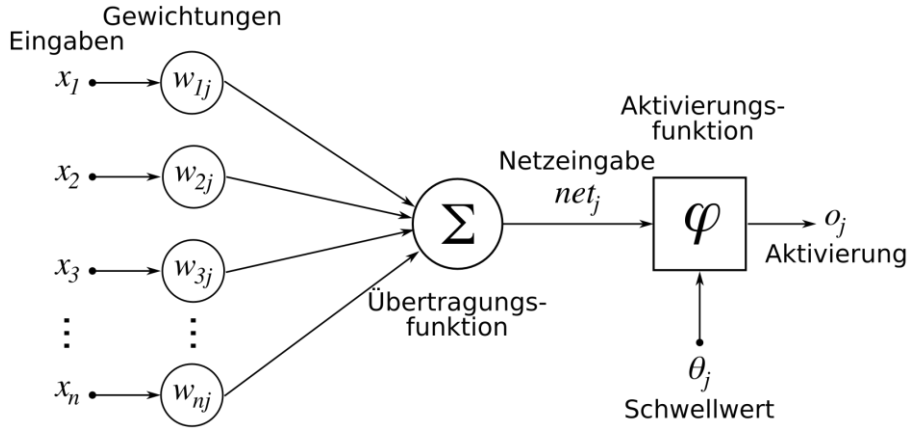
$y = \varphi(\varphi(\varphi (x W_1) W_2) W_3)$

- X ist eine Sammlung von n Inputvektoren (nx5)
- Y* ist der gewünschte Output für X (nx4)

$Y = \varphi(\varphi(\varphi (X W_1) W_2) W_3)$

Beim Trainieren versucht man Y möglichst nahe an Y* zu bringen, durchs Verändern der W_i.

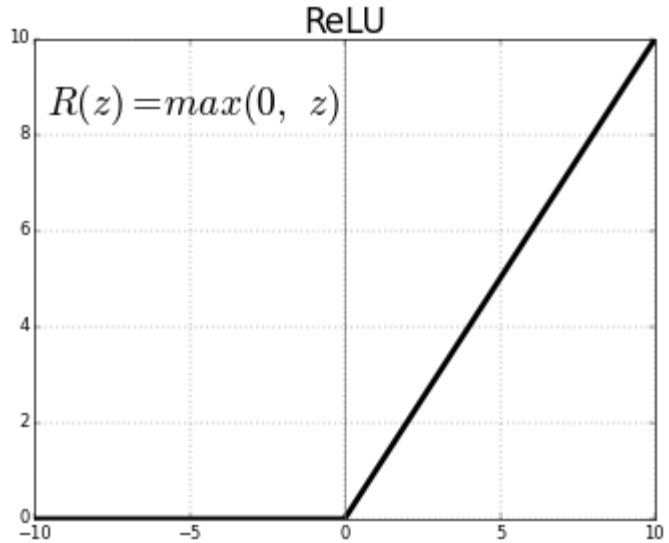
Aktivierungsfunktion φ



- Die Aktivierungsfunktion wird auf alle Neuronen nach der Matrixmultiplikation angewendet.
- Ohne Aktivierungsfunktion könnte man die Matrizen mit den Gewichten in einer einzigen Matrix durch Multiplikation kombinieren. Aus mehreren Schichten wird nur eine.

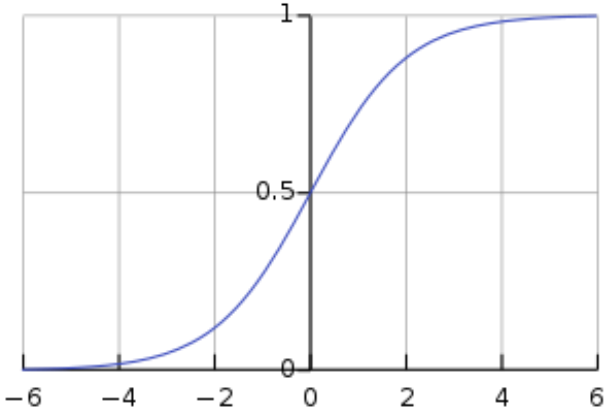
$$y = x\ W_1\ W_2\ W_3 = x\ W_{123}$$

- Die Aktivierungsfunktion ist nicht linear, verhindert so die Aggregation mehrerer Schichten und macht das neuronale Netz zum nicht linearen Gebilde.
- Rectified Lineare Unit (RELU) ist neu die meistverwendete Aktivierungsfunktion und hat tanh ersetzt. Ohne RELU kein Deep Learning. RELU verlangt Inputs im Bereich 0..1 (nicht Z-normiert)
- RELU(x) = max(0,x)
- dRELU(X)/dx = 0 ∀ x<0 und 1 ∀ x>0 x=0: nicht definiert
- Sehr schnell und einfach in Hardware zu implementieren



Andere Aktivierungsfunktionen

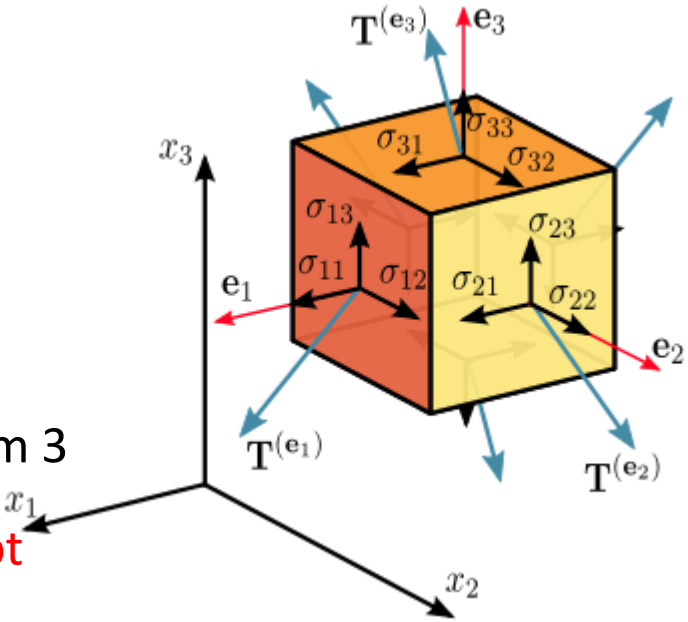
- RELU: einfach, schnell, meistverwendet, ergibt die besseren Resultate beim Training, auch wegen der Performance in der Ableitung
- Tanh: vor RELU, auf innere Schichten, Output zwischen -1 und 1
- Sigmoid (Logistic): auf innere Schichten, Output zwischen 0 und 1
- Softmax: bei Endschicht: Auswahl in Klassifizierungsprobleme
- …



Sigmoid-Funktion

Tensorbegriff

Deep Space Computing



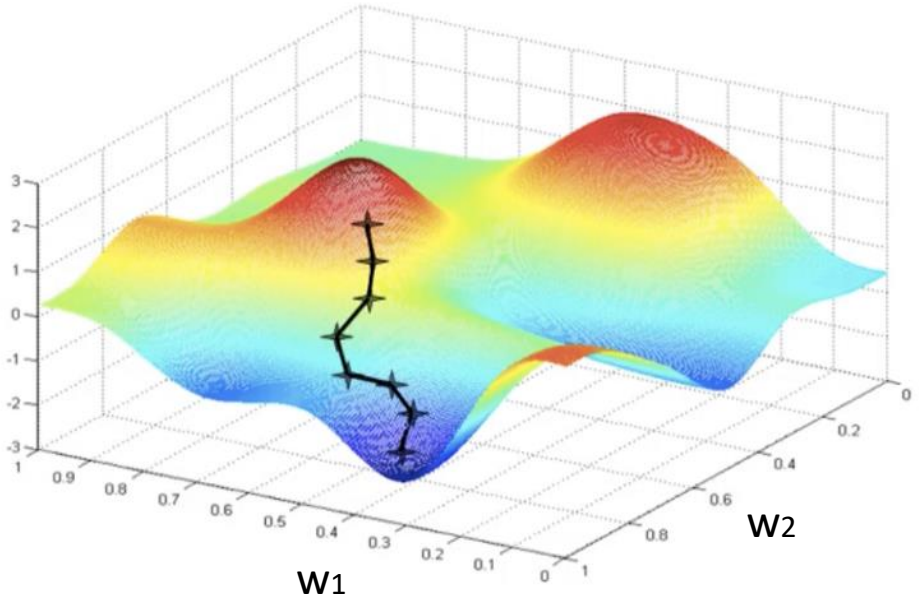
- Definition Multidimensionaler Array: 0D: skalar, 1D: Vektor, 2D: Matrix, 3D: Würfel von Werte, 4D: Zeitreihe von Würfel, nD:?
- Definition: Tensor in der Physik: multidimensionaler Array mit besonderen Eigenschaften. Cauchy-Stresstensor gibt zum Beispiel für jeden Punkt im Raum 3 Vektoren mit den Kräften, die aufs Material wirken (symmetrisch und entgegengesetzt auf die andere Seite vom Würfel). Pro Punkt im 3D Raum gibt es eine 3x3 Matrix (Tensorfeld). Vergleiche mit Temperatur (Skalarfeld) oder elektrisches Feld (Vektorfeld).
- Tensor im Deep Learning: multidimensionaler Array ohne besondere Eigenschaften
- Tensorflow: Google Framework furs Deep Learning. Ein neuronales Netz wird mit einer Gewichtungsmatriz pro Schicht dargestellt (nxm n: Input der Schicht, m: Output der Schicht). Zu jedem n-Inputvektor gibt es ein optimales Set von Gewichtungsmatrizen, die den Fehler minimiert (Tensorfeld auf n dim.-Raum).
- Neuronale Netze werden durch mehere Schichten definiert, so ist Tensorflow der Fluss der Matrizen durchs neuronale Netz.

TensorFlow

Training durch Backpropagation

Zielfunktion: $\min(\varepsilon(y^*, y(W))$
bei Regressionsaufgaben ist ε gleich wie MSE (Mean Squared Error), MAE (Mean Absolute Error), …
Training ist die Suche der Gewichte w_{ij}, die den Fehler ε minimieren.
Stochastisch Gradient Descent (SGD):

$\varepsilon(y^*, y(W))$



Strategie: verändern der Gewichte w_i in entgegengesetzter Richtung vom Gradient:

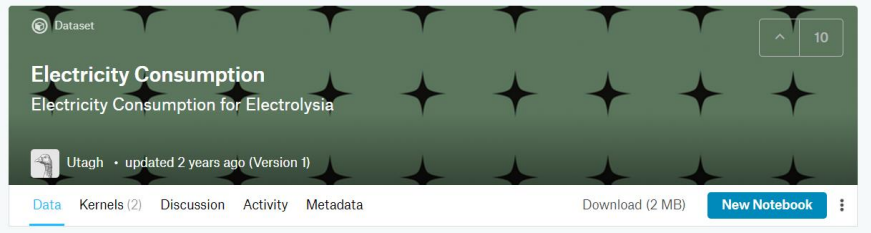
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla \varepsilon$$

$$\nabla \varepsilon = (d\varepsilon/dw_1, d\varepsilon/dw_2)$$

Momentan ist Adam (Adaptive Momentum), eine verbesserte Variante von SGD, der bestbekannte Optimizer für Deep Learning Aufgaben.

Beispiel: Voraussage der Stromnachfrage

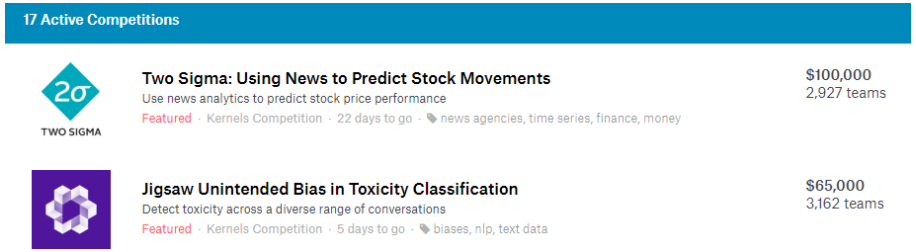
- Datensatz aus kaggle , eine Online Community fur Datenwissenschaftler und furs Deep Learning



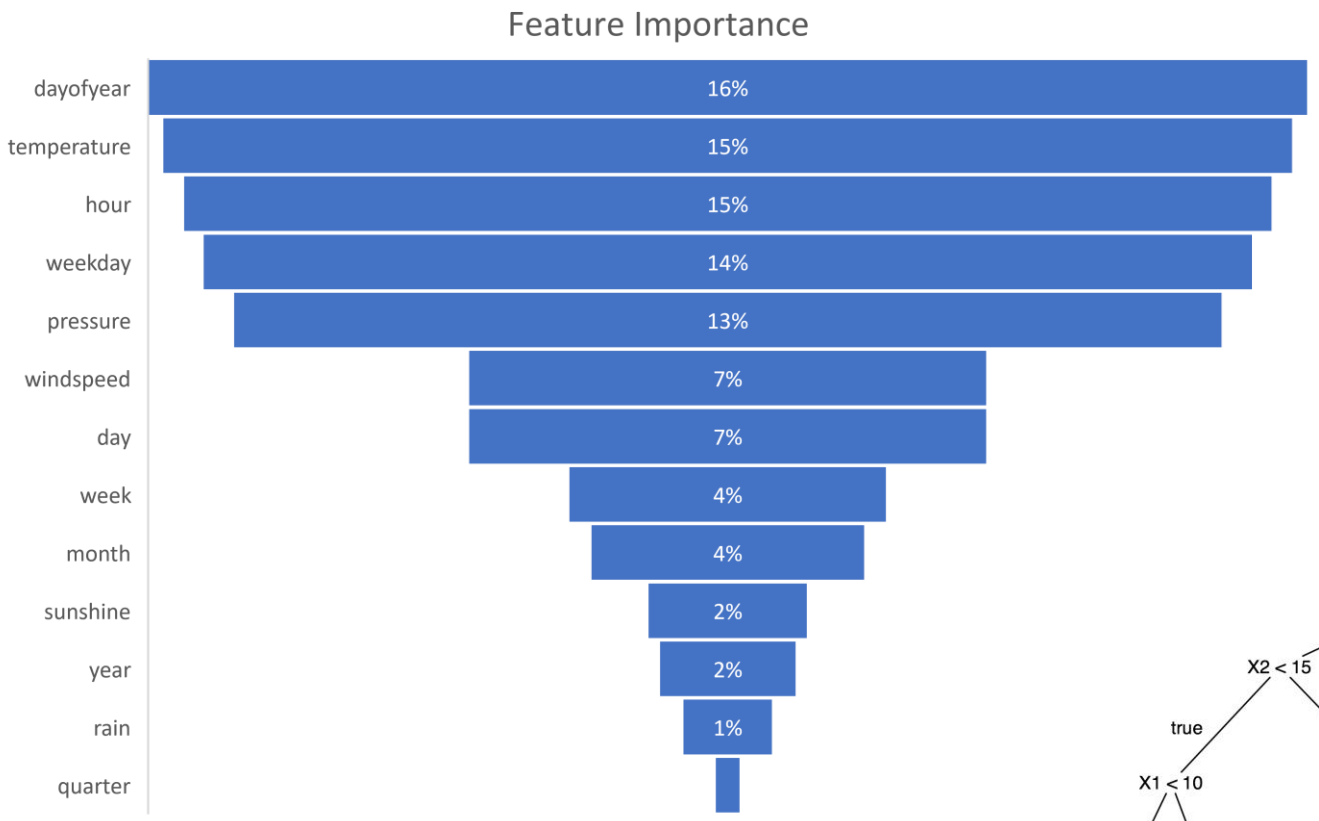
Company of Electrolysia supplies electricity to the city. It is looking to optimise its electricity production based on the historical electricity consumption of the people of Electrovania.

The company has hired you as a Data Scientist to investigate the past consumption and the weather information to come up with a model that catches the trend as accurate as possible. You have to bear in mind that there are many factors that affect electricity consumption and not all can be measured. Electrolysia has provided you this data on hourly data spanning five years.

- Auf Kaggle werden Wettbewerbe regelmässig ausgeführt:

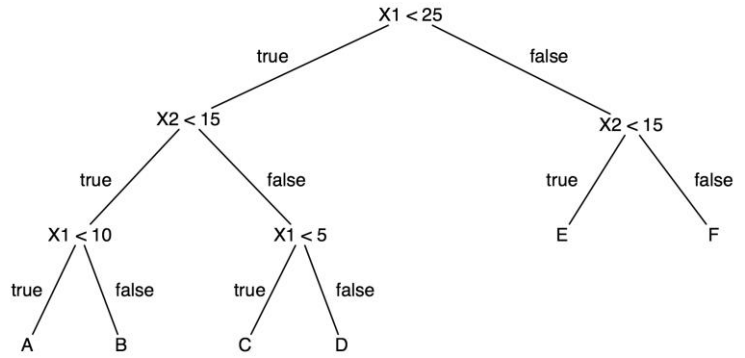


Ermitteln der wichtigen Eigenschaften



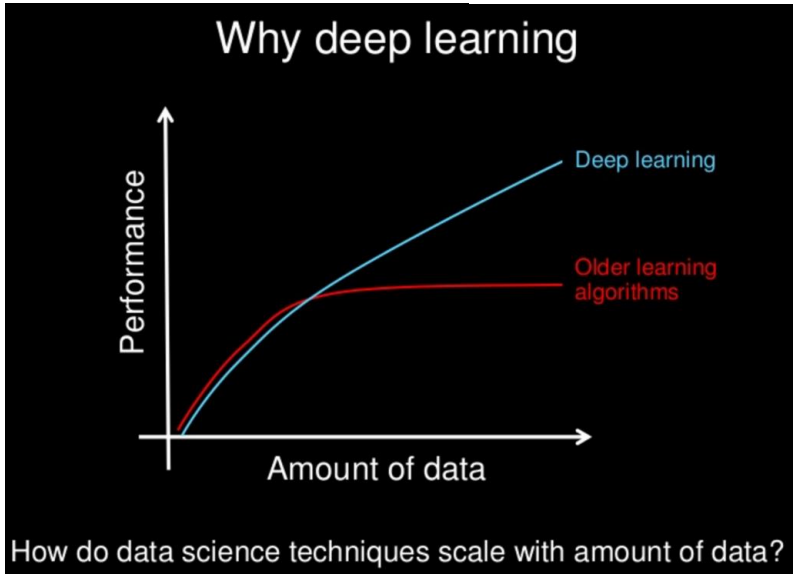

```
# create an instance for tree feature selection
tree_clf = ExtraTreesClassifier()

# fit the model
tree_clf.fit(X, y)

# Preparing variables
importances = tree_clf.feature_importances_
feature_names = df_train_features.columns.tolist()
```

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$



Trick: y mit Verbrauch ganzzahlig runden, so das n Klassen entstehen.

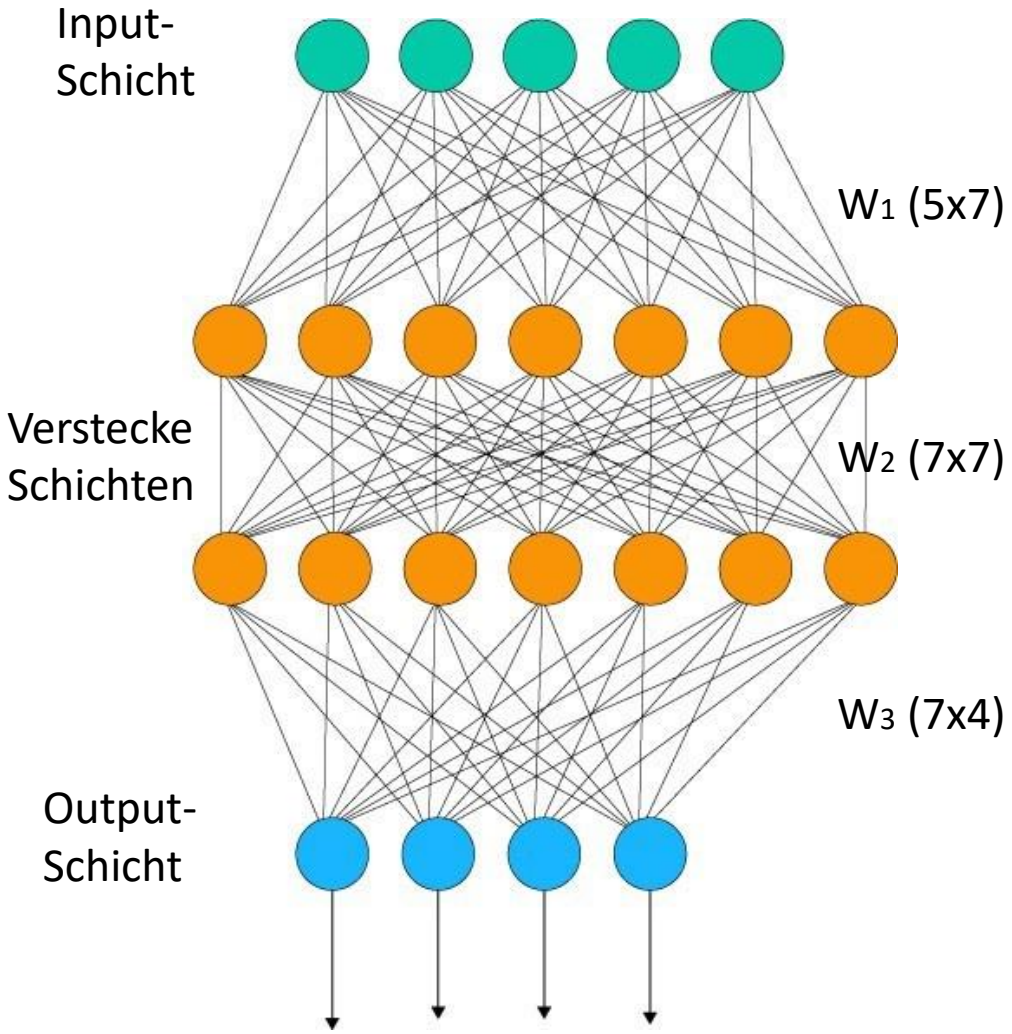
Data Augmentation



- Deep Learning funktioniert besser mit sehr vielen Daten
- Daten können aus historischen Daten generiert werden
    - Mit Jittering (Stören der Daten mit Rauschen)
    - Mit einem Autoencoder: eine besondere Art vom Neuronalen Netz in dem das Trainieren mit den Inputs zu den Outputs gleichgestellt werden
- Im Falle des Stromverbrauches:
  Man kann den Stromverbrauch verfeinert auf einzelne Regionen gekoppelt mit den regionalen Meteodaten anschauen und somit den Datenbestand deutlich erhöhen. Die finale Stromvoraussage ist die Aggregation als Summe der Stromvoraussage der einzelnen Regionen.
  Es wird also nicht eine einzelne Temperatur volumengewichtet über die Regionen aggregiert, sondern die regionalen Meteodaten zusammen mit den regionalen Verbräuchen als Input verwendet.

Modell aufsetzen

Deep Space Computing




```
import tensorflow as tf

model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=[5]),
        tf.keras.layers.Dense(7, activation='relu'),
        tf.keras.layers.Dense(7, activation='relu'),
        tf.keras.layers.Dense(4, activation='relu')
])


model.compile(optimizer='adam',loss='hinge_loss', metrics=['mean_absolute_error'])
model.summary()
model.fit(X, Y)
```

Trainieren des Modells auf GPUs (1/2)

Deep Space Computing

- Tensorflow verwendet komplexe Bibliotheken, die mit dem Grafiktreiber zusammenspielen müssen.
- Setup von Tensorflow ist aufwendig: mit Docker und deren Containertechnologie geht es einfacher!
- Docker ist eine leichtgewichtige Virtualisierungstechnologie: Dateisystem und Netzwerk werden virtualisiert, der Betriebssystemkernel allerdings nicht. Typischerweise: statt 20 Virtual Maschinen auf einem Host-Rechner können 400 Docker-Container parallel ausgeführt werden.

docker

TensorFlow

```
docki@phamnuwen:~$ docker images
REPOSITORY                                TAG                        IMAGE ID       CREATED          SIZE
k8s.gcr.io/kube-proxy                     v1.15.3                    232b5c793146   3 months ago     82.4MB
k8s.gcr.io/kube-apiserver                 v1.15.3                    5eb2d3fc7a44   3 months ago     207MB
k8s.gcr.io/kube-scheduler                 v1.15.3                    703f9c69a5d5   3 months ago     81.1MB
k8s.gcr.io/kube-controller-manager        v1.15.3                    e77c31de5547   3 months ago     159MB
localhost:5000/tensorflow2-tiz            latest                     4e2648055f8c   3 months ago     3.75GB
tensorflow2/tiz                           latest                     4e2648055f8c   3 months ago     3.75GB
jenkinsci/blueocean                       latest                     8167bc8f353c   3 months ago     553MB
localhost:5000/boinc-multigpu             latest                     c1ddf34006c8   4 months ago     529MB
boinc/client                              multi-gpu                  c1ddf34006c8   4 months ago     529MB
portainer/agent                           <none>                     6406ba8babae   4 months ago     13.3MB
portainer/portainer                       <none>                     2b4ddf654e1c   4 months ago     77.7MB
dockerizeddeltasql_php-apache             latest                     7369bca2783e   4 months ago     377MB
boinc/client                              nvidia                     861e4a84f387   4 months ago     259MB
localhost:5000/boinc-nvidia               latest                     861e4a84f387   4 months ago     259MB
mariadb                                   10.3                       09d25c986bab   4 months ago     343MB
alpine                                    latest                     b7b28af77ffe   4 months ago     5.58MB
nvidia/cuda                               10.1-base-ubuntu18.04      d5ce0ddf6959   4 months ago     106MB
nvidia/cuda                               9.0-base                   b8ce4f75eabc   4 months ago     137MB
tensorflow/tensorflow                     2.0.0b1-gpu-py3-jupyter    c6ac8bf1a2cc   5 months ago     3.52GB
localhost:5000/tensorflow2                latest                     c6ac8bf1a2cc   5 months ago     3.52GB
registry                                  2                          f32a97de94e1   8 months ago     25.8MB
quay.io/coreos/flannel                    v0.11.0-amd64              ff281650a721   10 months ago    52.6MB
k8s.gcr.io/coredns                        1.3.1                      eb516548c180   10 months ago    40.3MB
hello-world                               latest                     fce289e99eb9   10 months ago    1.84kB
localhost:5000/hello-world                latest                     fce289e99eb9   10 months ago    1.84kB
k8s.gcr.io/etcd                           3.3.10                     2c4adeb21b4f   12 months ago    258MB
byrnedo/alpine-curl                       latest                     026de4ba9530   15 months ago    5.93MB
nginx                                     <none>                     ebe2c7c61055   19 months ago    18MB
php                                       7.2.1-apache               f99d319c7004   22 months ago    377MB
k8s.gcr.io/pause                          3.1                        da86e6ba6ca1   23 months ago    742kB
dockerinpractice/docker-image-graph       latest                     52b010fed971   2 years ago      661MB
winking/dockerui                          latest                     d2d6a9eee080   3 years ago      10.2MB
localhost:5000/dockerui                   latest                     d2d6a9eee080   3 years ago      10.2MB
gliderlabs/resolvable                     master                     1f3be0ace382   3 years ago      255MB
php                                       5.6.1-apache               b54b0680f7f5   5 years ago      805MB
```

Trainieren des Modells auf GPUs (2/2)

- Auf jedem Rechner im Cluster wird Kubernetes installiert.
- Die Orchestrierung der Container geschieht durch Kubernetes
- Kubernetes erlaubt, die Rechner-Infrastruktur als Quelltext zu beschreiben
- Falls einen Host-Rechner ausfällt, kann Kubernetes die Docker-Containers (Pods) auf einem anderen Host-Rechner starten.
- Einfaches Verwalten einer heterogenen Hardware-Infrastruktur
- Einbinden vom externen Speicher

kubernetes

Schiffsanalogon

Deep Space Computing

- Docker war der professionelle Arbeiter, der vor Containers Schiffe bis zur letzten Ecke beladen hat.
- Kubernetes (abgekürzt k8s) = griechiesch für Rudergänger oder Steuersmann

Stromverbrauch-Ergebnisse

```
dm@deepspace:~$ nvidia-smi
Tue Nov 26 11:00:55 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.56       Driver Version: 418.56       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:01:00.0 Off |                  N/A |
| 67%   83C    P2   250W / 250W |    313MiB / 11177MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  Off  | 00000000:03:00.0 Off |                  N/A |
| 38%   64C    P2   160W / 250W |    967MiB / 11178MiB |     75%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     11526      C   acemd3                                       303MiB |
|    1     12638      C   ...3_x86_64-pc-linux-gnu__GW-opencl-nvidia   957MiB |
+-----------------------------------------------------------------------------+
```

```
Epoch 23/26
 1000/20000 [>.............................] - ETA: 0s - loss: 0.0046 - mean_abs
20000/20000 [==============================] - 0s 2us/sample - loss: 0.0044 - me
an_absolute_error: 0.0469
Epoch 24/26
 1000/20000 [>.............................] - ETA: 0s - loss: 0.0048 - mean_abs
20000/20000 [==============================] - 0s 2us/sample - loss: 0.0044 - me
an_absolute_error: 0.0467
Epoch 25/26
 1000/20000 [>.............................] - ETA: 0s - loss: 0.0046 - mean_abs
20000/20000 [==============================] - 0s 2us/sample - loss: 0.0044 - me
an_absolute_error: 0.0464
Epoch 26/26
 1000/20000 [>.............................] - ETA: 0s - loss: 0.0041 - mean_abs
20000/20000 [==============================] - 0s 2us/sample - loss: 0.0043 - me
an_absolute_error: 0.0463
evaluation
  32/6496 [.............................] - ETA: 5s - loss: 0.0033 - mean_absol
3072/6496 [=============>................] - ETA: 0s - loss: 0.0055 - mean_absol
6496/6496 [==============================] - 0s 20us/sample - loss: 0.0051 - mea
n_absolute_error: 0.0494
dm@phamnuwen:~/tf$
```

Speedup bei LSTM Textgenerator auf einzelnem Rechner mit einer Mid Range GPU (768 CUDA cores):

- Nur mit CPU: 8 Stunden 20 Minuten
- Mit 1x GPU GTX 1050 Ti: 20 Minuten

Speedup: etwa Faktor 25

Im Training: 4.63%, auf dem Testdatensatz: 4.94% Mean Absolute Error, auf eine einzelne GPU

Deep Learning -> Quantenrechner

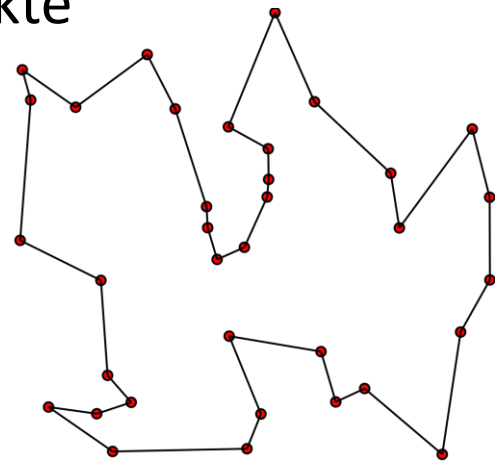
Deep Space Computing

- Es gibt erste Ansätze, um neuronale Netze auf Quantenrechner zu bringen. Der Dwave-Quantenrechner ist für jeden in der Cloud auf cloud.dwavesys.com/leap gratis verfügbar, kälter als im Weltraum (3K), 14 mK

Traveling Salesman Problem auf Quantenrechner

Deep Space Computing

- Der Rechner hat 2030 Qubits, jeder Qubit kann man mit ~5 anderen koppeln, in der sogenannten Chimera-Topology oder Chimera-Graph
- Das TSP-Problem mit N Städten muss “hot coded” sein, so N^2 Qubits sind nötig
- Die Distanz-Matrix wird über die Kopplung auf die N^2 Qubits angewendet. Es handelt sich um ein vollständiges Graph K_N^2
- Dieses K_N^2 Graph muss auf das weniger verbundene Chimera Graph gemappt werden.
- Praktisch kann man TSP auf bis zu N=7 Städte (Hybrid DWave N=16, exakte Lösung bekannt für N=85’900)
- Jede ~1.5 Jahre die Qubits verdoppeln sich, aber über 10’000 gibt es ein kryogenisches Problem zum lösen: die Kabel rauschen zu stark.



Take Aways beim Quantenrechner

Deep Space Computing

- DWave ist kein universeller Quantenrechner, aber eine dedizierte Maschine, um QUBO-Probleme zu lösen. (Quadratic Binary Optimization). Die lösbaren Probleme sind NP-Probleme, die zum Ising-Modell gemappt werden.
- DWave verwendet einen Prozess, der adiabatisches Quantum Computing heisst, das auf Quantum Annealing basiert und ähnlich mit Simulated Annealing ist.
- Simulated annealing kann TSP-Instanzen lösen.
- Mit der nächsten Dwave-Generation, Pegasus, fürs Ende 2020, das komplette "einbaubare" Graph kommt auf 180 Knoten, √180 ~ 13 Städte. So, TSP-Instanzen bis auf 13 Städte sind lösbar.

- TSP und VRP (Vehicle Routing Problem) sind lösbar auf dem DWave-Quantenrechner, da sie nur binäre Unbekannten aufweisen.
- VRPTW (VRP with Time Windows) ist nicht einfach zu mappen, da der Anfangszeitpunkt vom Auftrag eine unbekannte ist, aber nicht eine binäre.

Basic graph embedding statistics

	C16	P6	P16
Complete Graph	64 (17)	60 (7)	180 (17)
Complete Bipartite	64x64 (16)	52x52 (5)	172x172 (15)
Cubic Lattice	8x8x8 (4)	5x5x12 (2)	15x15x12 (2)
Factoring Circuit	16-bit (16)	10-bit (5)	30-bit (15)
Grid with Diagonals	16x16 (6)	15x15 (6)	45x45 (6)

(max chain length in blue)

$$H(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j \quad \text{where } x_i, x_j \in \{0,1\}$$

Links zu den Beispielen

Stromverbrauch-Beispiel:

- www.kaggle.com/utathya/electricity-consumption
- www.kaggle.com/utathya/ann-on-electricity-consumption
- www.kaggle.com/rishickesh/power-consumption

TSP auf Quantenrechner:

- cloud.dwavesys.com/leap
- github.com/BOHRTECHNOLOGY/quantum_tsp

Lesenswerter Arstechnica Beitrag auf Englisch (02.12.2019):

How neural network work – and why they’ve become a big business

Danke für die Aufmerksamkeit!

deep space computing AG

6. Dezember 2019

deep-space.ch