TÜRK-ALMAN ÜNİVERSİTESİ
TÜRKISCH-DEUTSCHE UNIVERSITÄT

Zukünftige Börsenprognose (RNN oder CNN)

Zeynep Sena Karabacak

Informatik – TDU

e160503110@stud.tau.edu.tr

Github: https://github.com/zeynepsenak/Stockprediction-CNN-RNN-
(RNN kodu çalışıyor CNN kısmı henüz tam değil)

Abstract

Die Börsenprognose ist ein bedeutendes Thema in unserem Leben. Wissenschaft und Technologie entwickeln sich, damit es einfacher sein wird, den Aktienmarkt mithilfe von maschinellem Lernen, eingehendem Lernen und verschiedenen Lernmethoden vorherzusagen. Aufgrund der steigenden Inflation in der Welt, ist es noch wichtiger.[1]

Für die Schätzungen der Kursbewegungen an der Börse werden meistens das Recurrent Neural Network (RNN) verwendet. Es gibt aber auch Versuche mit der Convolutional Neural Network.[2][3] Die Idee, Zahlen zu visualisieren und dann neue visuelle Datensätze zu erstellen, ist interessant. Aber hat CNN gut Ergebnisse für die Aktienprognose? Sollten wir den Hauptweg gehen, der RNN ist, oder CNN versuchen.

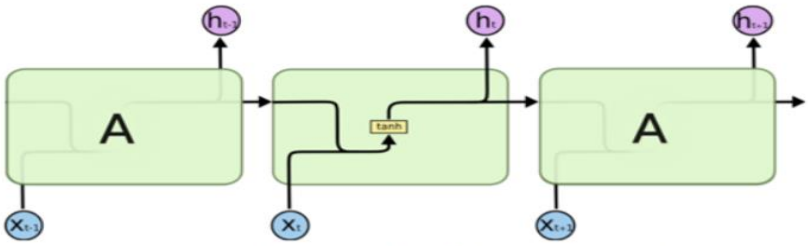
In diesem Artikel geht es um Aktienprognose mit CNN und RNN. Als Eingabe werden wir aus der Internet dem Eröffnungspreis, dem hohen Preis, dem niedrigen Preis, dem Schlusskurs und dem Aktienvolumen nehmen und es mit CNN und RNN testen. Als Ergebnis werden wir die Vor- und Nachteile der beiden untersuchen.

1. Einleitung

Wirtschaft ist wichtig für Länder, Unternehmen und Einzelpersonen. Aber Finanzindikatoren sind so kompliziert und auch die Fluktuation an den Aktienmärkten ist sehr gewalttätig. Die Vorhersage des Marktwerts ist von großer Bedeutung, um den Gewinn beim Kauf von Aktienoptionen zu maximieren und gleichzeitig das Risiko gering zu halten. Mit dem Fortschreiten des Maschinellen Lernens steigt jedoch die Möglichkeit, ein stetiges Vermögen an der Börse zu erzielen, und es hilft Experten, die informativsten Indikatoren für eine bessere Vorhersage herauszufinden.

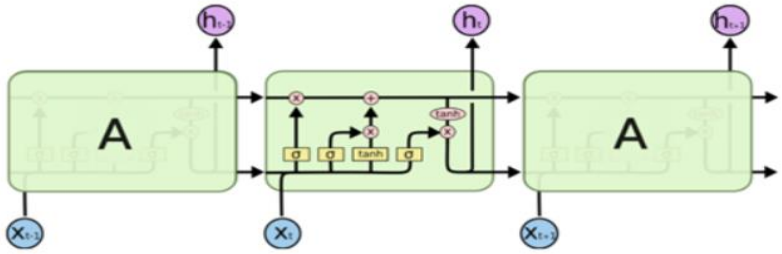
Bevor beginnen lass uns die CNN und RNN Struktur kennenlernen. Feed Forward neuralem Netz Aufbau ist eine Methode, die eine Ausgabe generiert, indem mathematische Operationen auf die

Informationen angewendet werden, die von der Eingabe in Neuronen auf Schichten kommen. Im Gegensatz zu anderen neuronalen Netzen wertet die RNN-Struktur auch frühere Eingaben aus. Das heißt, die Ausgabe erfolgt durch Kombinieren der vorherigen und aktuellen Eingaben. Es wird dann berechnet, um die Fehlerrate mit Backpropagation (Gradient-Descent) zu reduzieren. Gradient ist der Wert, der zum Anpassen des Gewichts von Neuronen verwendet wird. In langen miteinander verbundenen Netzwerken wird jedoch die Auswirkung des Fehlers verringert, und der Gradient beginnt zu verschwinden. Dies kann es unmöglich machen, das richtige Ergebnis zu erzielen. Die Verwendung von RELU anstelle von Sigmoid oder Tanh als Aktivierungsfunktion kann als Lösung für dieses Problem gezeigt werden. Eine andere Lösung für dieses Problem ist LTSM (Long Term Short Memory). Es wurde entwickelt, um dieses Problem zu lösen.



The repeating module in a standard RNN contains a single layer.

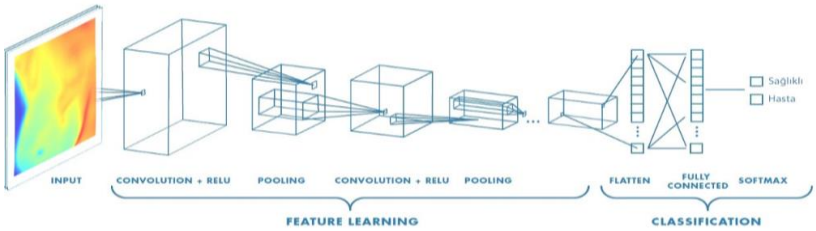
LTSM hat vier speziell gebundene Schichten anstelle der neuronalen Schicht. Diese Schichten werden Türen genannt. Bei diesen Türen entscheidet die Zelle, was gespeichert, wann gelöscht, gelesen und geschrieben werden soll. Diese Gates haben eine Netzwerkstruktur und eine Aktivierungsfunktion. Genau wie bei Neuronen wird die eingehende Eingabe nach Gewicht weitergeleitet oder gestoppt.



The repeating module in an LSTM contains four interacting layers.

Andererseits lernt und verwendet CNN die einzigartigen Eigenschaften von Objekten, um die visuellen Eingaben zu unterscheiden, die es empfängt. Dies geschieht ähnlich in unserem Gehirn. Wir nennen es ein Flugzeug, indem wir die Merkmale des Bildes betrachten, das wir betrachten, wie die zwei Flügel, den Motor und die Fenster.

Cnn macht zuvor erkennen sie Eigenschaften auf niedrigerer Ebene wie Kurven und Kanten und erstellen sie für abstraktere Konzepte. Um diese Funktionalität zu erreichen, verarbeitet Cnn das Bild mit verschiedenen Ebenen. Diese; Convulational Layer, Non-Linearity Layer, Pooling Layer, Flattening Layer und Fully Connected Layer.



Für CNN ist die Eingabe jedoch visuell aber was wir an der Börse haben, sind viele Zahlenwerte. Wir können diese Zahlenwerte jedoch grafisch darstellen. Wir können diese Grafiken auch als visuelle Eingabe in CNN platzieren. Es gibt Studien, die sich mit dieser Idee befassen. Es ist jedoch nicht so bekannt wie RNN für die Börsenanalyse. Hier stellt sich die Frage: Kann die Visualisierung der Zahlen bessere Vorhersagen liefern?

2.Die Methodik für RNN

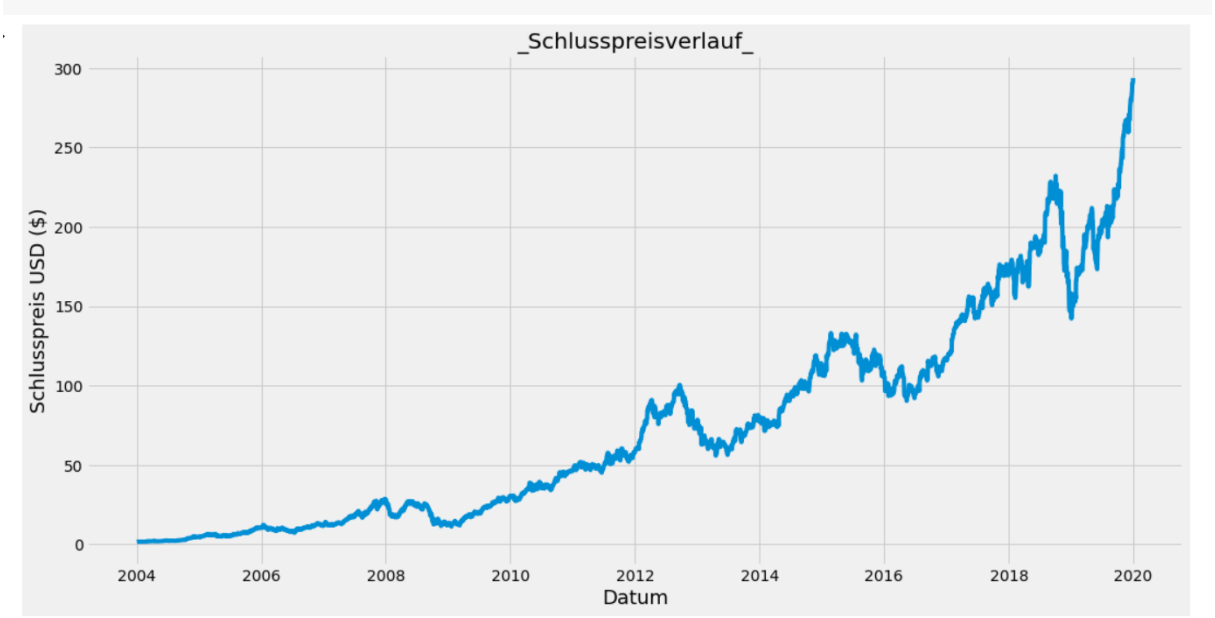
Hier wird das Recurrent Neuronale Netz (RNN) und das Long Short Term Memory (LSTM) berücksichtigt. Metodologie:

In der ersten Phase werden Daten von Apple übernommen. [6].

Date	High	Low	Open	Close	Volume	Adj Close
2004-01-02	1.553571	1.512857	1.539286	1.520000	36160600.0	1.315860
2004-01-05	1.599286	1.530000	1.530000	1.583571	98754600.0	1.370894
2004-01-06	1.601429	1.550714	1.589286	1.577857	127337000.0	1.365947
2004-01-07	1.630714	1.566429	1.578571	1.613571	146718600.0	1.396865
2004-01-08	1.695000	1.617857	1.631429	1.668571	115075800.0	1.444478
...	...	...	...	...	...	...
2019-12-24	284.890015	282.920013	284.690002	284.269989	12119700.0	282.831299
2019-12-26	289.980011	284.700012	284.820007	289.910004	23280300.0	288.442780
2019-12-27	293.970001	288.119995	291.119995	289.799988	36566500.0	288.333313
2019-12-30	292.690002	285.220001	289.459991	291.519989	36028600.0	290.044617
2019-12-31	293.679993	289.519989	289.929993	293.649994	25201400.0	292.163818

4027 rows × 6 columns

Apple’s Schlusspreisverlauf zieht von 01-01-2004 bis 01-01-2020 wieso aus:



Für zweiten Schritt habe Ich die Trainingsdaten als die ersten 0,8 aller Daten festgelegt. Dann habe ich die Daten im Bereich 0-1 skaliert.

Und für die nächste Stufe werden die Daten in x_train und y_train aufgeteilt. Für alle 60 haben wir Eingaben in x_train und für die Ausgaben 61st in y_train gemacht. Und dann werden sie in ein numpy Array konventiert. Und das erste 61-teilige Datensatz sieht wieso aus (Erste 60 in x_train und 61st in y_train) :

```
[array([0.        , 0.00021761, 0.00019805, 0.00032031, 0.00050858,
       0.00042056, 0.00059905, 0.00069441, 0.00071397, 0.00038388,
       0.00035209, 0.00035454, 0.0003252 , 0.00022006, 0.00031297,
       0.000423  , 0.00043767, 0.00030319, 0.00034231, 0.00031297,
       0.00025429, 0.00023962, 0.0001247 , 0.00027874, 0.00034965,
       0.00033987, 0.00041567, 0.00061616, 0.00059905, 0.00042056,
       0.00045968, 0.00048413, 0.00029097, 0.00027385, 0.0002225 ,
       0.00026407, 0.0003741 , 0.00043034, 0.00064551, 0.00066996,
       0.00061861, 0.00064551, 0.0009487 , 0.00133502, 0.00115408,
       0.00142305, 0.00156486, 0.00143527, 0.00153552, 0.00126411,
       0.00111007, 0.00120054, 0.0010734 , 0.00111985, 0.00111985,
       0.00098048, 0.00103183, 0.00136681, 0.00140837, 0.0016211 ])]
[0.0016235433938904783]
```

Dann habe ich Modell gebaut. Es hat 2 Schichten mit 50 Neuronen und 2 Dense Schichte. Erste Dense Schichten hat 25 Neuronen und zweite hat 1 Neuron.

Danach Modell wird mit dem Optimierer 'adam' trainiert. Adam ist eine Kombination aus RMSprop und stochastischem Gradientenabstieg mit Impuls betrachtet werden. Es verwendet die quadratischen Gradienten, um die Lernrate wie RMSprop zu skalieren, und nutzt den Impuls, indem es den gleitenden Durchschnitt des Gradienten anstelle des Gradienten selbst wie SGD mit Impuls verwendet. Und Verlust wird mit RMSE (Root Mean Square Error) berechnet. Der RMSE-Wert wird als Wurzel des Durchschnitts der Quadrate des bei jedem Schritt gemachten Fehlers bezeichnet. Auf diese Weise berechnen wir, wie genau wir erreicht haben. Wenn es 0 ist, dann ist das Modell perfekt.

Unser RNN-Modell sieht wieso aus:

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 60, 50)            10400
_________________________________________________________________
lstm_8 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_5 (Dense)              (None, 25)                1275
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

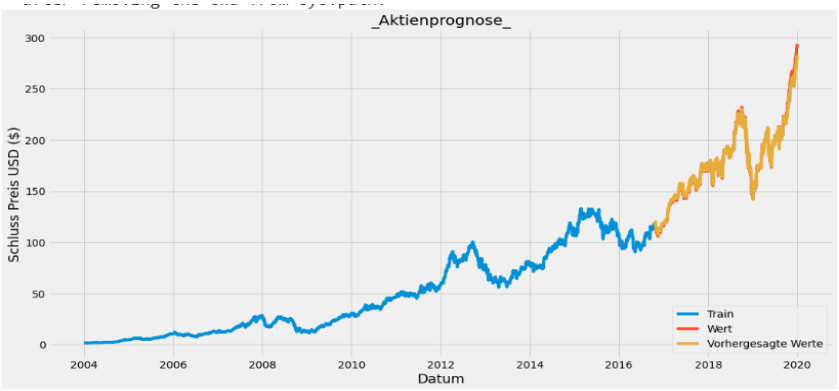
Um den 61. Tag vorherzusagen, haben wir die Daten der letzten 60 Tage angegeben. Dies liegt daran, dass der mögliche Wert eines neuen Tages nicht nur vom vorherigen Tag abhängt. Input ist x_train[i], das also [1,60] ist. Unit ist die Größe der Ausgabe in LSTM Schicht. Hier haben wir den Wert 50, also ist die Ausgabe [1,50] für erste und zweite LSTM's Schiten. Wir reduzieren unsere Input 60 bis 50. Mit diesen LSTM-Ebenen lernt das Modell, was zu speicherm und was zu löschen ist. Dann reduzieren wir unsere Ausgabedimensionalität, um das Ergebnis mit der Dense Schicht (25 Units) zu linearisieren. Und mit der zweiten Dense Schicht erhalten wir eine lineare Ausgabe. Wir müssen für jede Eingabe eine Ausgabe erhalten, und deshalb verwenden wir diese Dense-Layers.

Dann wird die test-Daten geprüft und dann habe ich als Ergebnis für unterschiedliche epoches und batch-sizes diese Grafiken.

Für batch size: 4, epoch: 7

RMSE-Wert : 3.590120342682842

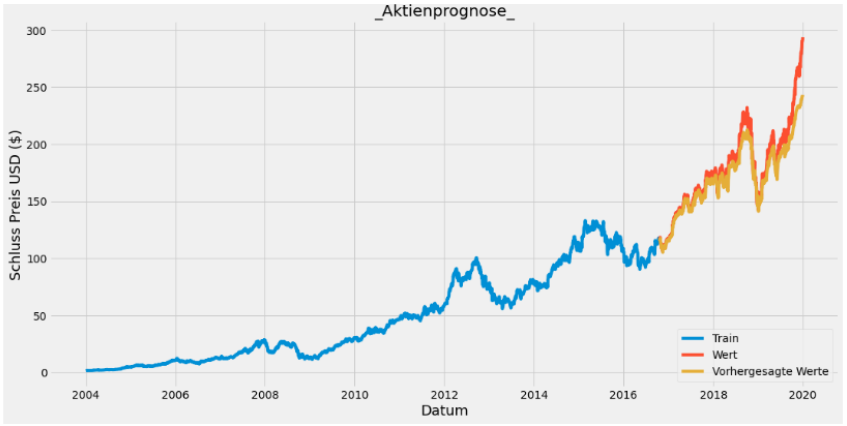
2019-12-30	291.519989	279.247284
2019-12-31	293.649994	281.705811



Für batch size: 1, epoch: 300

RMSE-Wert : 11.687409314722181

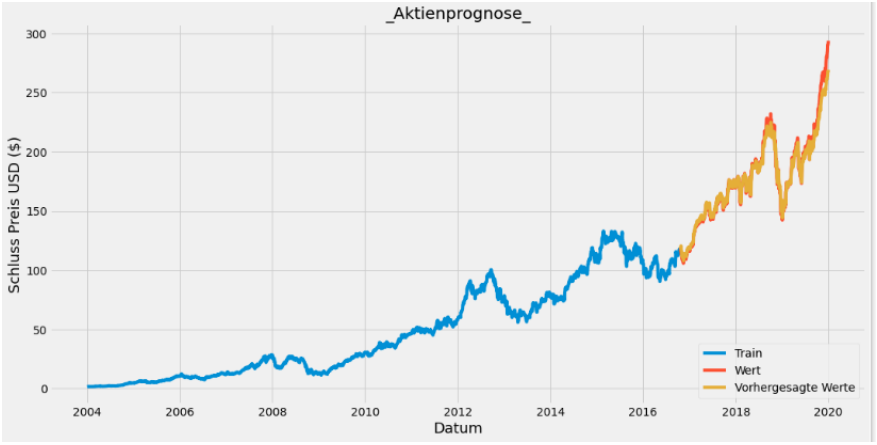
2019-12-30	291.519989	242.379822
2019-12-31	293.649994	243.164108



Für batch size: 32, epoch: 100

RMSE-Wert : 7.025421235846431

2019-12-30	291.519989	267.852936
2019-12-31	293.649994	269.168365

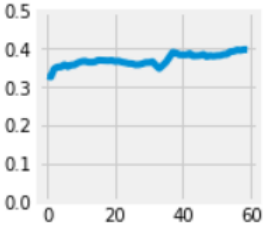


Das beste Ergebnis war Epoche: 7 und Batch-Size: 1. RMSE für diese Werte ist 3.59. Das ist nicht perfekt aber auch nicht schlecht.

3.Die Methodik für CNN

Ich habe die gleichen Apple Daten auch in CNN verwendet und Börsenprognose für CNN habe ich zwei Wege probiert.

Die erste Methode, mit der ich versucht habe, die Zahlen zu visualisieren, besteht darin, die Abschlussdaten der letzten 60 Tage aufzuzeichnen und das Diagramm zu speichern. Es gibt Diagramme mit blauen Linien, und für jedes Bild wurden 120 Tage zuvor Daten geschlossen. Und es gibt die Ausgabe von Tag 61, die jeder visuellen Eingabe entspricht.

Beispiele Eingabe ist:  und die Ausgabe: 293.649994

Modell, das ich benutzt habe, sieht wieso aus:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 142, 142, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 140, 140, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 70, 70, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 70, 70, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 313600)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               40140928
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 40,160,449
Trainable params: 40,160,449
Non-trainable params: 0
```

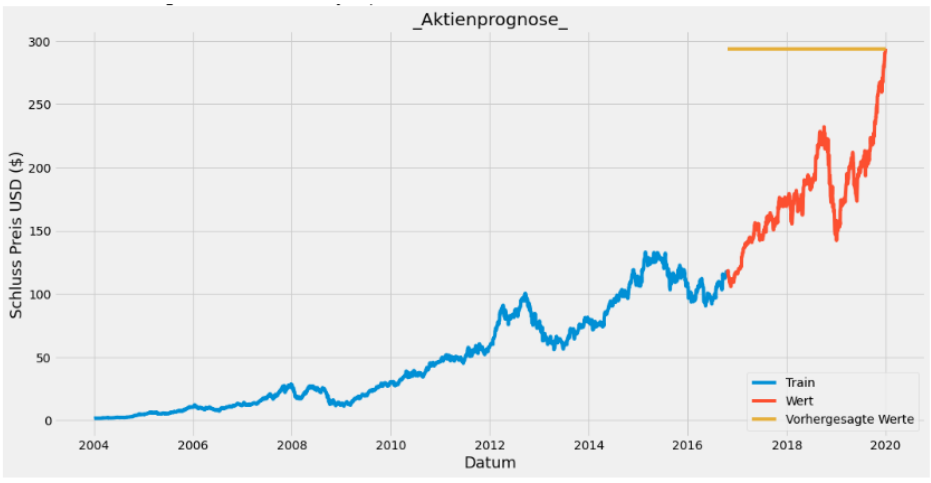
Die Conv2D-Layers ist zum Transformation das Eingabebild in eine sehr abstrakte Darstellung. In diesen Ebenen lernt das Modell durch Visualisierung des Bildes. Pooling ist die Schicht, die zur Reduzierung der Rechenkomplexität verwendet wird. Das Modell lernt hier nicht. Und dann habe ich dropout, flatten, dense Schichten verwendet, um zu linearisieren.

Aber es nie funktioniert hat. Ich habe das Modell nicht vertieft, weil ich dachte, dass die Methode zum Konvertieren der Grafik in ein Bild nicht sehr effektiv sein könnte. Es gibt nicht das geringste Anzeichen von Lernen.

```
[ ] #Train das Modell
    model.fit(x_train, y_train, batch_size=1, epochs=10)
```

```
Epoch 1/10
3162/3162 [==============================] - 1912s 605ms/step - loss: 0.7208
Epoch 2/10
3162/3162 [==============================] - 1910s 604ms/step - loss: 0.7208
Epoch 3/10
3162/3162 [==============================] - 1910s 604ms/step - loss: 0.7208
Epoch 4/10
3162/3162 [==============================] - 1909s 604ms/step - loss: 0.7208
Epoch 5/10
```

Loss ändert sich nicht und RMSE: 121.3227335936156



Die zweite Methode, die ich versuchte, bestand darin, die Daten der letzten 120 Tage in ein Numpy-Array zu konvertieren und das resultierende Array in ein visuelles zu konvertieren. Das resultierende Bild ist unten.

Und die Ausgabe für das ist: 293.64994

Modell, das ich diesmal benutzt habe, sieht wieso aus:

```
Model: "sequential_37"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_197 (Conv2D)          (None, 12, 10, 32)        896
_________________________________________________________________
conv2d_198 (Conv2D)          (None, 10, 8, 32)         9248
_________________________________________________________________
max_pooling2d_104 (MaxPoolin (None, 5, 4, 32)          0
_________________________________________________________________
dropout_95 (Dropout)         (None, 5, 4, 32)          0
_________________________________________________________________
conv2d_199 (Conv2D)          (None, 5, 4, 64)          18496
_________________________________________________________________
max_pooling2d_105 (MaxPoolin (None, 2, 2, 64)          0
_________________________________________________________________
dropout_96 (Dropout)         (None, 2, 2, 64)          0
_________________________________________________________________
conv2d_200 (Conv2D)          (None, 2, 2, 128)         73856
_________________________________________________________________
conv2d_201 (Conv2D)          (None, 2, 2, 128)         16512
_________________________________________________________________
max_pooling2d_106 (MaxPoolin (None, 1, 1, 128)         0
_________________________________________________________________
dropout_97 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
conv2d_202 (Conv2D)          (None, 1, 1, 128)         16512
_________________________________________________________________
max_pooling2d_107 (MaxPoolin (None, 1, 1, 128)         0
_________________________________________________________________
dropout_98 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
conv2d_203 (Conv2D)          (None, 1, 1, 64)          8256
_________________________________________________________________
conv2d_204 (Conv2D)          (None, 1, 1, 64)          4160
_________________________________________________________________
max_pooling2d_108 (MaxPoolin (None, 1, 1, 64)          0
_________________________________________________________________
dropout_99 (Dropout)         (None, 1, 1, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               33280
_________________________________________________________________
dropout_100 (Dropout)        (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 513
=================================================================
Total params: 181,729
Trainable params: 181,729
Non-trainable params: 0
```

Wie ich bereits gesagt habe, ist die Conv2D-Layers zum Transformation das Eingabebild in eine sehr abstrakte Darstellung. In diesen Ebenen lernt das Modell durch Visualisierung des Bildes. Hier ist ein viel tieferes Modell als das, das ich zuvor verwendet habe. Pooling ist die Schicht, die zur Reduzierung der Rechenkomplexität verwendet wird. Das Modell lernt hier nicht. Und dann habe ich dropout, flatten, dense Schichten verwendet, um zu linearisieren.

Aber diese Methode auch funktioniert nicht.

Im Gegensatz zur vorherigen nimmt die Fehlerfunktion zunächst ab, bleibt dann aber konstant und lernt nicht mehr.

```
[ ] #Train das Modell
    model.fit(x_train, y_train, batch_size=4, epochs=25)

    Epoch 1/25
    3102/3102 [==============================] - 6s 2ms/step - loss: 0.7184
    Epoch 2/25
    3102/3102 [==============================] - 6s 2ms/step - loss: 0.7169
    Epoch 3/25
    3102/3102 [==============================] - 6s 2ms/step - loss: 0.7161
    Epoch 4/25
    3102/3102 [==============================] - 6s 2ms/step - loss: 0.7156
    Epoch 5/25
    3102/3102 [==============================] - 5s 2ms/step - loss: 0.7155
    Epoch 6/25
    3102/3102 [==============================] - 6s 2ms/step - loss: 0.7154
    Epoch 7/25
    3102/3102 [==============================] - 5s 2ms/step - loss: 0.7154
    Epoch 8/25
    3102/3102 [==============================] - 5s 2ms/step - loss: 0.7154
    Epoch 9/25
```

Aber RMSE Werte (Lernwerte) für diesen Model ist 121.3227335936156, also sehr schlecht. Die Ausgabe hier hatte Werte zwischen 0-1.

Schließlich habe ich als Klassifizierung versucht. Und ich habe die Ausgabewerte als 0 oder 1 erstellt. Wenn es 1 ist, hat sich sein Wert gegenüber dem Vortag erhöht, und wenn es 0 ist, hat sich sein Wert gegenüber dem Vortag verringert. Ich war mir sicher, dass dies viel besser funktionieren würde als zuvor. Weil CNN-Modelle im Allgemeinen verwendet werden, um festzustellen, ob Bilder ein Objekt enthalten. Das heißt, ein Bild enthält oder nicht den Object, das wir suchen. Der Ausgabewert ist also 1 oder 0.

Als ich diesmal mit demselben Cnn-Modell versuchte, wurden die Ausgabewerte in 0 oder 1 konvertiert, was ein viel schlechteres Ergebnis war. Neuer RMSE-Wert ist: 293.10631766682997

4.Abschluss

Cnn abstrahiert Bilder und legt bestimmte Eigenschaften fest und erkennt Objekte, die diese Eigenschaften haben. Wenn wir die Zahlen visualisieren, gibt es jedoch keine spezifischen Eigenschaften, die es uns ermöglichen, das Ergebnis zu verstehen. Daher konnten wir mit den von uns erstellten Cnn-Modellen kein gutes Ergebnis erzielen. Ein flaches RNN-Model machte um ein Vielfaches bessere Vorhersagen. Ein besseres Ergebnis für CNN scheint nur erzielt zu werden, wenn die Bilder, die wir aus den Zahlen erstellen, bestimmte Linien haben.

5.Related works

Stock Prediction using Convolutional Neural Network[2]

Convolutional Networks for Stock Trading[3]

Vorhersage von Aktienkursen mit LSTM[4]

Stock Market Index Prediction with artificial neural networks[5]

6.Datasets

Yahoo - Apple Stock[6]

Yahoo – Google Stock[7]

7. Verweise

1. https://www.statista.com/statistics/256598/global-inflation-rate-compared-to-previous-year/

2.https://www.researchgate.net/publication/328754813_Stock_Prediction_Using_Convolutional_Neural_Network

3. https://pdfs.semanticscholar.org/86af/cb8f6c9e182d79a25071371e6b567619b6dc.pdf

4. https://www.researchgate.net/publication/327967988_Predicting_Stock_Prices_Using_LSTM

5.http://www.acarindex.com/dosyalar/makale/acarindex-1423905602.pdf

6. https://finance.yahoo.com/quote/AAPL?p=AAPL

7. https://finance.yahoo.com/quote/GOOGL%3B/