**Recurrent Neural Network**

Muhammad Haris Maulana

Institute Teknologi Bandung, Master Program in Informatics, Bandung, Indonesia

23519008@std.stei.itb.ac.id

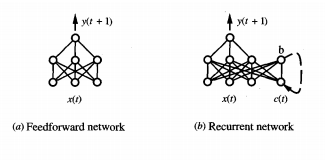
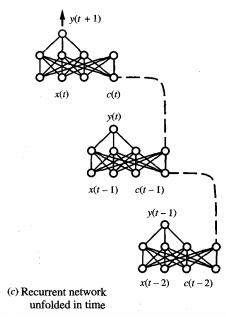
**Abstract.** Recurrent Neural Network atau RNN adalah sebuah *artificial neural network* khusus untuk dataset bertipe *time series* yang menggunakan hasil output dari jaringan sebagai input untuk input untuk perhitungan berikutnya. Dalam laporan ini akan dibahas dasar dari RNN beserta bagimana implementasi RNN dalam bentuk prediksi *stock market* dan *character level language model.*

**1 Pendahuluan**

*Recurrent Neural Networks* (RNN) adalah sebuah *artificial neural networks* (ANN) yang menggunakan dataset yang berurutan (*time series data)* dan menggunakan hasil output dari jaringan sebagai input untuk perhitungan berikutnya [1]. Input tersebut dapat dianalogikan sebagai ‘memori’ dan dapat berguna dalam memproses data yang dependen terhadap rantaian data atau kejadian yang dapat mempengaruhi sebuah keluaran seperti *Natural Language Processing* atau sistem rekomendasi[2].

Terdapat 4 tugas pembelajaran RNN yaitu:

1. Dasar Recurrent Neural Networks
2. Character Level Language Model – Dinosaurus Island
3. Prediksi Stock berbasis LSTM RNN menggunakan dataset Tata
4. Prediksi Stock berbasis LSTM RNN menggunakan dataset mandiri

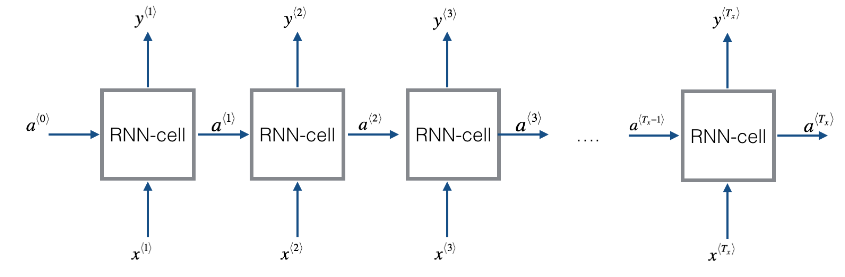
 

**Fig. 1.** Ilustrasi Feedforward Network dan Recurrent network

**2 Pengerjaan**

**2.1 Dasar Recurrent Neural Network**

RNN yang akan dipelajari memiliki struktur yang dapat dilihar pada Fig. 2, dimana RNN terdiri atas urutan RNN Cell yang hasil dari setiap Cell menjadi input untuk Cell berikutnya dan seterusnya. Dalam pengerjaan ini, kalkulasi akan dilakukan untuk setiap individu Cell dan mengiterasi kalkulasi sesuai dengan jumlah input satu persatu.



**Fig. 2.** Ilustrasi RNN Sederhana

**2.1.1 Cell Forward**

Sebuah RNN dapat dianalogikan sebagai pengulangan sebuah sel tunggal. Pertama diimplementasikan perhitungan untuk satu langkah waktu. Gambar pertama pada Fig. 3 menjelaskan operasi untuk satu langkah waktu sel RNN.

Langkah dalam kalkulasi sebuah Cell adalah sebagai berikut

1. Kalkulasi hasil *hidden state* menggunakan fungsi aktivasi tanh

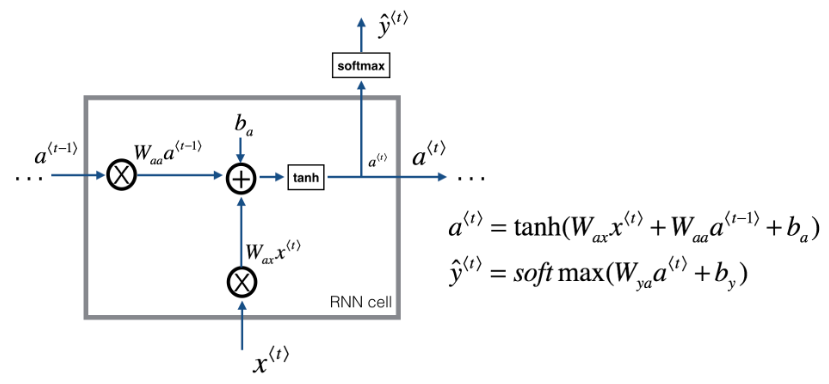


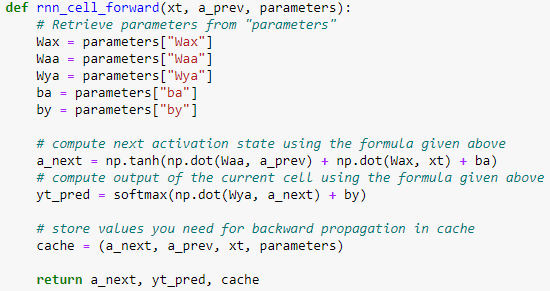
1. Menggunakan hasil kalkulasi *hidden state* sebelumnya, kalkulasi prediksi:



1. Simpan hasil kalkulasi kedalam *cache*





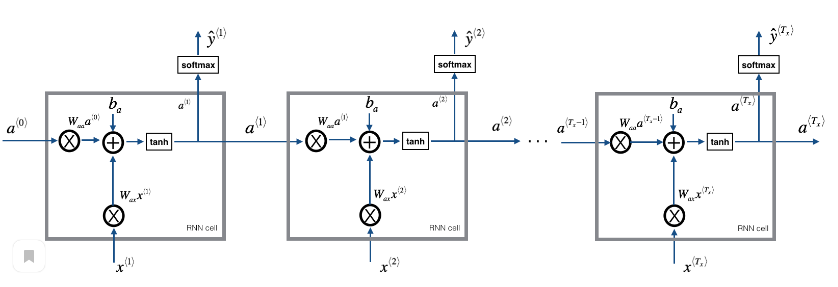


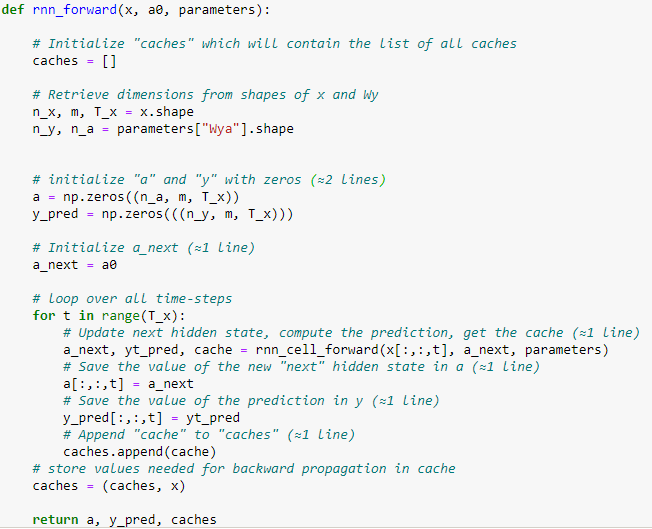
**Fig. 3.** RNN Cell Forward

**2.1.2 RNN Forward Pass**

Langkah-langkah pada RNN Forward Pass meliputi:

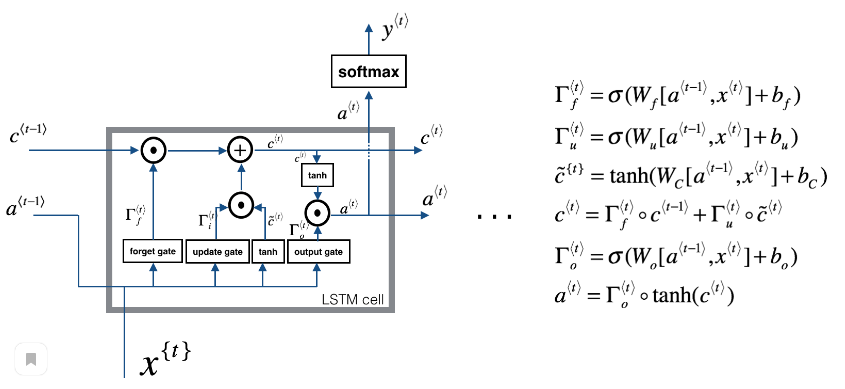
1. Membuat vektor untuk menyimpan hasil kalkulasi *hidden states* RNN dalam bentuk vektor bernilai 0
2. Tentukan nilai *hidden state* awal
3. Mulai iterasi untuk setiap RNN Cell ke-i dengan:
   1. Menjalankan Kalkulasi RNN Cell Forward untuk mendapatkan nilai *hidden state*
   2. Simpan kalkulasi pada posisi ke-i
   3. Masukkan *cache* kedalam list khusus *cache*





**Fig. 4.** RNN Forward

**2.1.3 LSTM**



**Fig. 5.** LSTM

LSTM yang dibentuk merupakan RNN dengan 3 Gate khusus yang mempengaruhi hasil output, gates tersebut adalah:

1. Forget Gate

Bertujuan untuk ‘melupakan’ atau mengabaikan memori yang sebelumnya tersimpan dalam kasus tertentu, misalkan dalam NLP bahasa Inggris menentukan apakah objek merupakan *singular* atau *prural* dan dalam prediksi stock jika data yang berumur terlalu tua dianggap tidak relevan terhadap prediksi. Berikut adalah rumus fungsi *forget* yang dapat bernilai 1 atau 0



1. Update Gate

Bertujuan untuk mengimplementasikan hasil keputusan pada *forget gate*. Fungsi ini juga dapat bernilai antara 1 atau 0



Untuk menghitung hasil perubahan vektor digunakan:

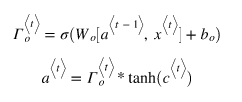


Rumus diatas dan fungsi update digunakan untuk menentukan hasil perubahan dalam *hidden state*



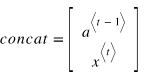
1. Output Gate

Untuk menentukan hasil output berdasarkan gate forget dan update, digunakan formula:

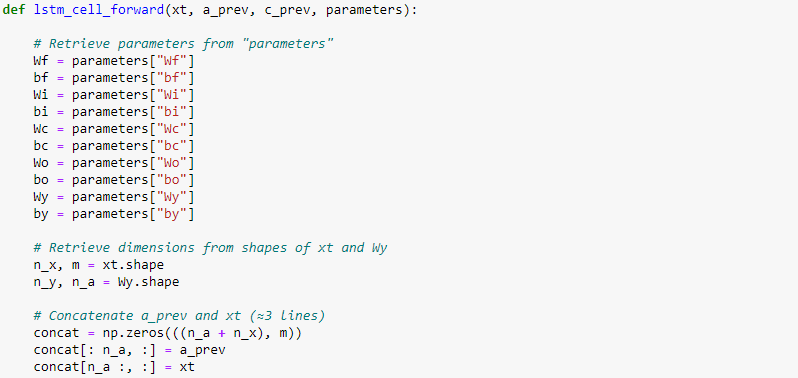


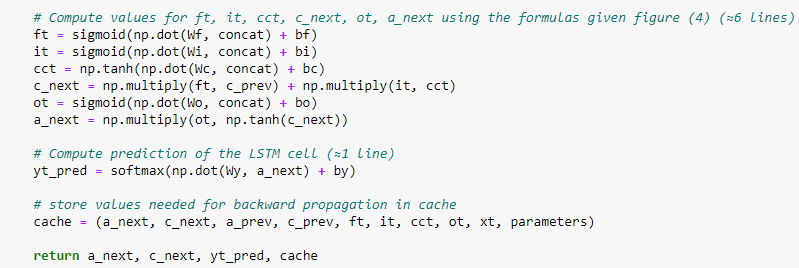
**2.1.4 LSTM Cell Forward**

Kalkulasi setiap Cell seperti 2.1.1 namun ditambah perhitungan khusus untuk setiap *gate* dalam LSTM Cell yaitu *forget, update,* dan output. Pertama-tama gabung matriks *hidden state* (a) dan *input* (x) sehingga:



Lalu hitung setiap *weight* dan *bias* dari *gate* dan input menggunakan fungsi sigmoid dan hyperbolic tangent yang diilustrasikan dalam fig. 6. Lalu gunakan fungsi *softmax* untuk menentukan hasil outout cell.

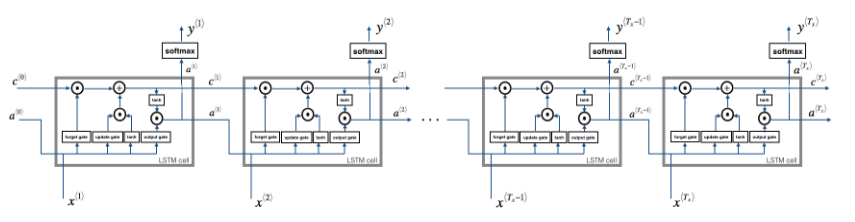


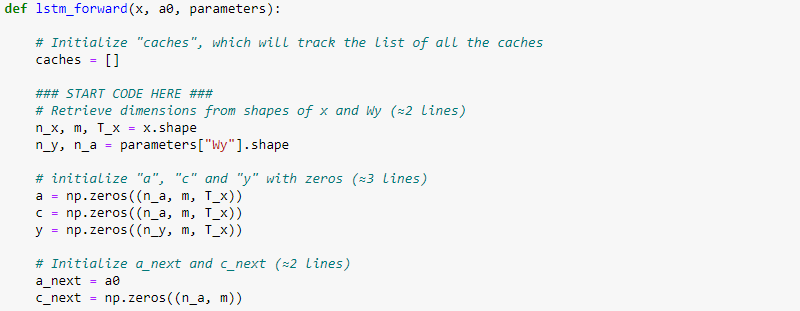


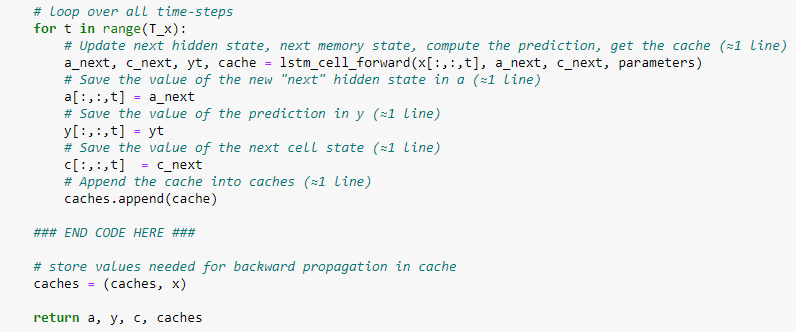
**Fig. 6.** LSTM Cell Forward

**2.1.5 LSTM Forward**

Iterasi prosedur LSTM Cell Forward untuk setiap *cell* dalam RNN





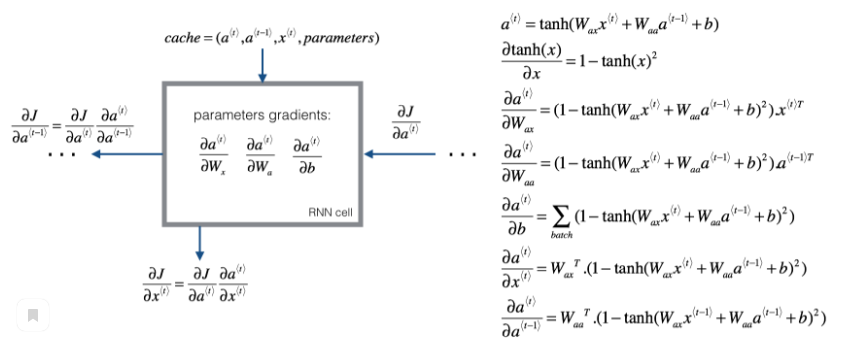


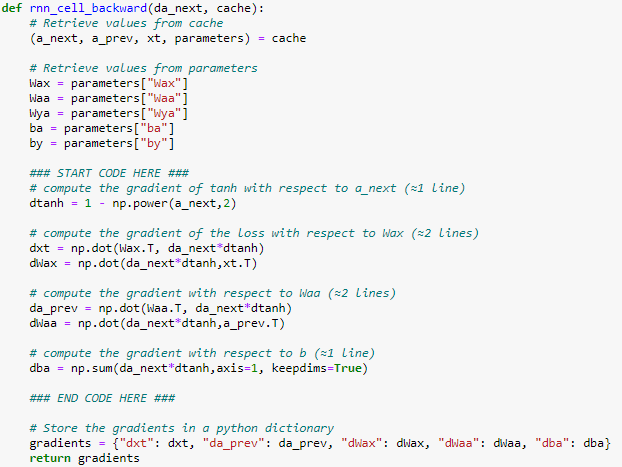
**Fig. 7.** LSTM Forward

**2.1.6 RNN Cell Backward**

Asumsikan penurunan dari fungsi tanh adalah :

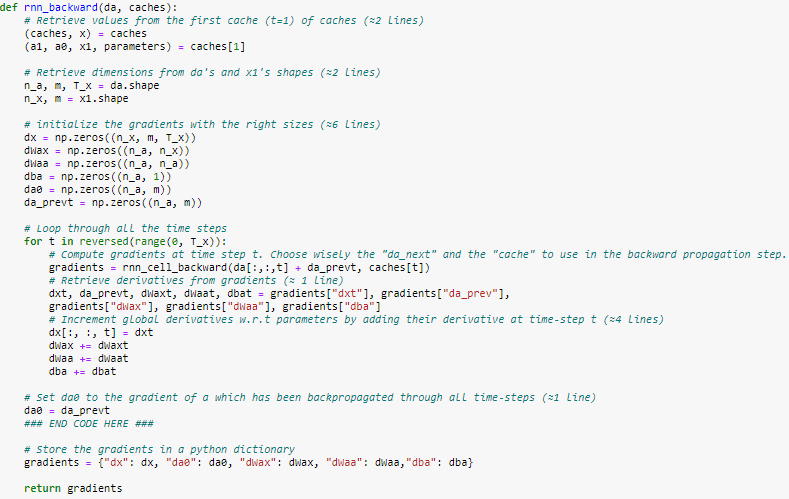






**Fig. 8.** RNN Cell Backward

**2.1.7 RNN Backward**



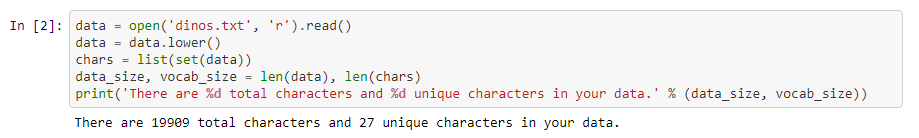
**Fig. 9.** RNN Backward

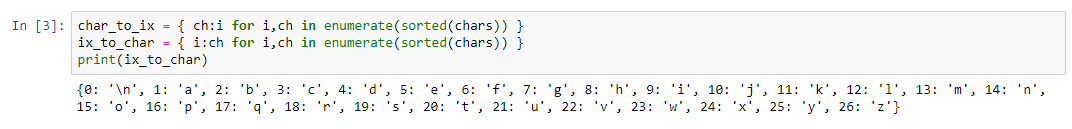
**2.2 Character Level Language Model – Dinosaurus Island**

Dalam tugas ini, dibutuhkan sebuah generator nama dinosaurus berdasarkan nama-nama dinosaurus yang telah ada menggunakan RNN dengan *backward propagation.*

**2.2.1 Data Preprocessing**

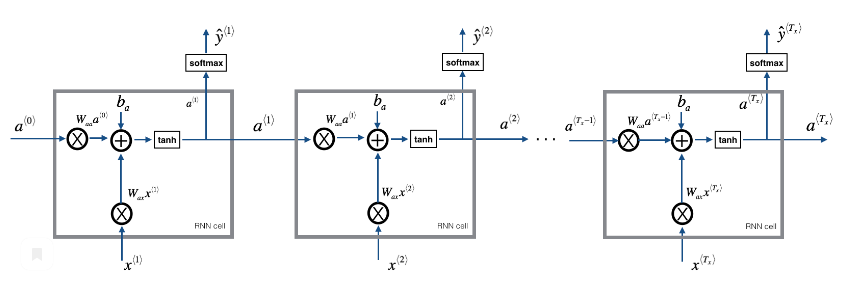
Terdapat 26 karakter "\n" atau EOS (Akhir kalimat). EOS berfungsi untuk menunjukkan akhir nama dinosaurus daripada akhir kalimat. Diperlukan sebuah kamus *python* untuk memetakan setiap karakter ke indeks dari 0-26. dan kamus *python* kedua yang memetakan setiap indeks kembali ke karakter-karakter yang sesuai. Hal ini bertujuan untuk membantu mengetahui indeks apa yang sesuai dengan karakter apa dalam output distribusi probabilitas *softmax*





**Fig. 10.** Dinos Data Preprocessing

**2.2.2 Gambaran Umum Model**



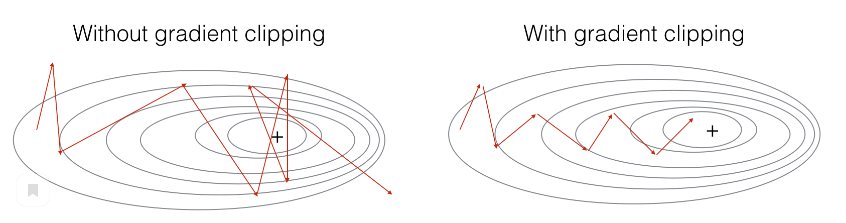
**Fig. 11.** Dinos General Structure

Berikut Struktur umum dari Model RNN yang akan dibangun

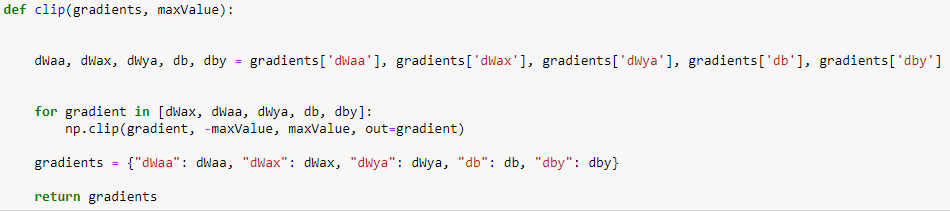
1. Inisialisasi parameter
2. Iterasi Optimisasi
   1. Forward propagation
   2. Backward propagation
   3. Clip the gradients
   4. Update parameter with the gradient descent update rule.

**2.2.3 Clipping**

*Gradient clipping* adalah teknik untuk mencegah *exploding gradien*t dalam *network* yang sangat dalam. Gradien meledak dapat terjadi ketika gradien menjadi terlalu besar dan gradien kesalahan menumpuk, menghasilkan *network* yang tidak stabil. Gradient clipping pada umumnya melakukan pengaturan nilai *threshold* dan jika gradien melebihi *threshold* tersebut maka akan dilakukan ‘pemotongan’

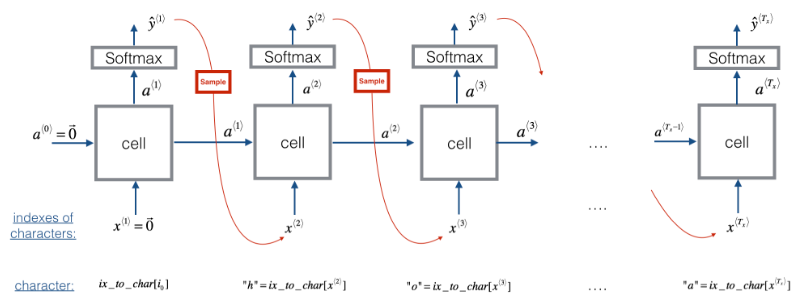


**Fig. 12.** Gradient Clipping Illustration



**Fig. 13.** Gradient Clipping Implementation

**2.2.4 Sampling**



**Fig. 14.** Sampling Illustration

*Sampling* adalah proses men-*generate* karakter-karakter yang akan digunakan untuk memberi nama dinosaurus, proses ini terbagi menjadi 4 tahap yaitu:

1. Memasukkan input *dummy* untuk men-generasi sebagai *default input*
2. Lakukan Forward Propagation untuk memperoleh output *cell* dan *hidden state* pertama dengan rumus:



1. Lakukan proses *sampling*: pilih indeks karakter berikutnya berdasarkan kemungkinan distribusi:



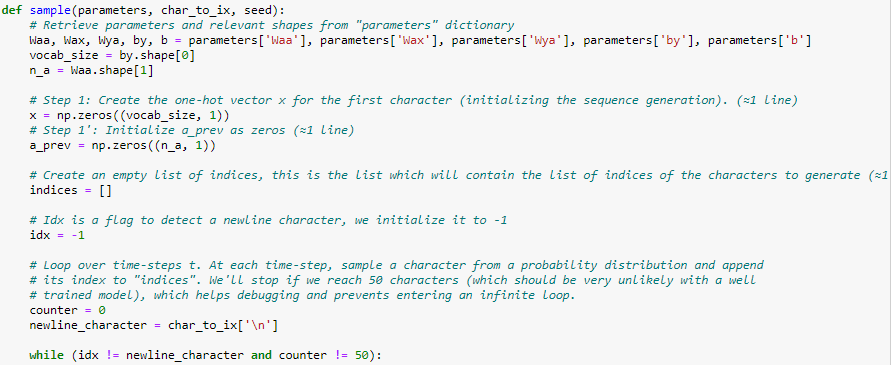
1. *Overwrite* variabel x, yang bernilai:

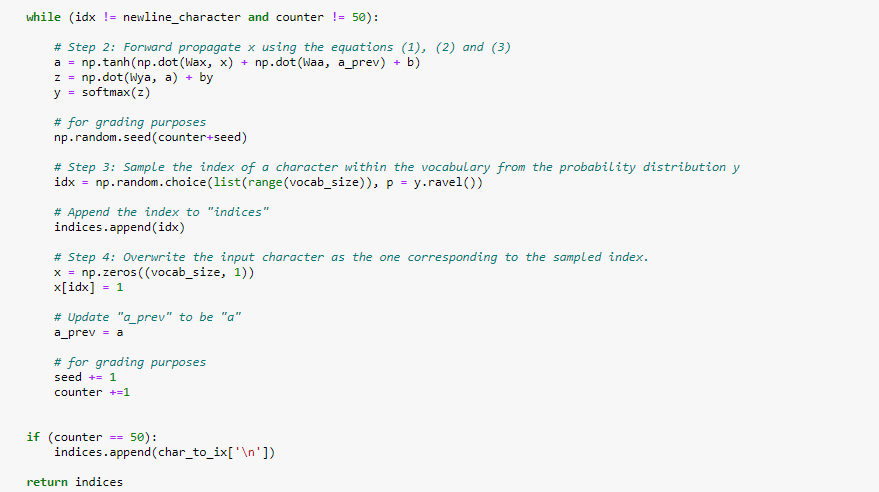


Dengan:



1. Buat *one-hot vector* berdasarkan karakter terpilih lalu lakukan *forward propagation* terhadap variabel X tersebut (kembali ke tahap 1) hingga menemukan EOS



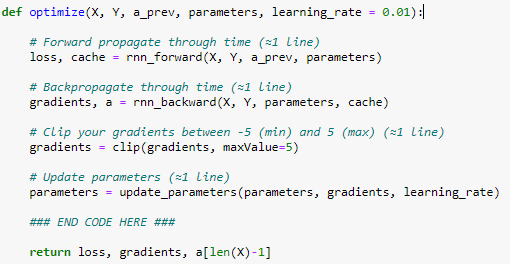


**Fig. 15.** Sampling Implementation

**2.2.5 Gradient Descent**

Lakukan *stochastic gradient descent,* yaitu proses optimisasi yang meliputi:

* Forward propagation
* Backward propagation
* Clip the gradients
* Update parameters

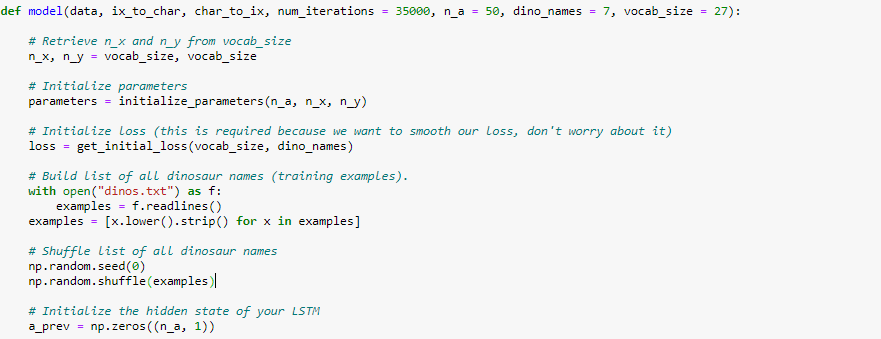


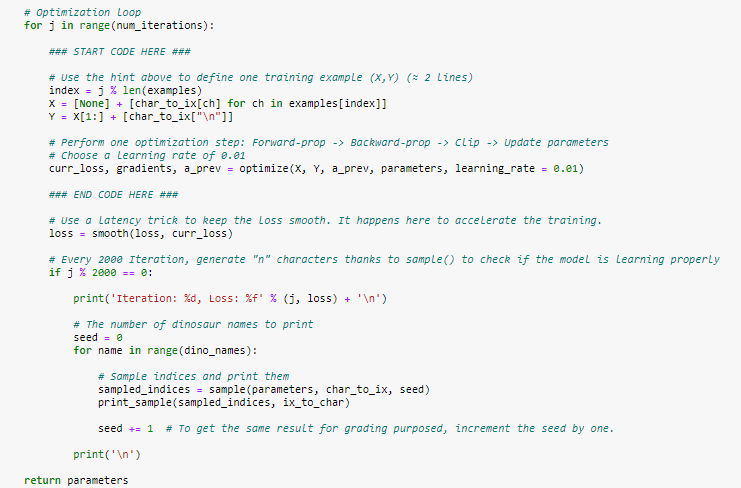
**Fig. 16.** Gradient Descent Optimization Implementation

**2.2.5 Model Train**

Untuk memodelkan generator nama dinosaurus, dilakukan tahap-tahap sebagai berikut:

1. Optimisasi Model berulang kali
2. Sampling Model dan cek performa
3. Acak Dataset untuk memastikan optimisasi mencakup distribusi data
4. Ulangi terus menerus hingga performa memenuhi target yang diinginkan



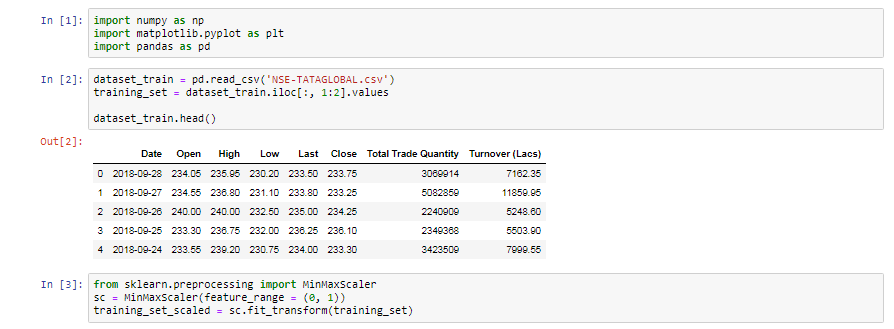


**Fig. 17.** Model Training Implementation

**2.3 Prediksi Stock Market berbasis LSTM RNN menggunakan dataset Tata**

**2.3.1 Data Preprocessing**

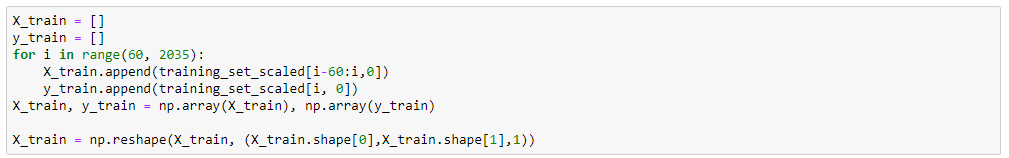
Terdapat banyak data dengan variabel angka yang cukup bervarian yang dapat memakan banyak performa sehingga diperlukan *scaling* untuk mengoptimisasi performa. Oleh karena itu digunakan *library* khusus dari Scikit- Learn yaitu *MinMaxScaler* untuk mengubah skala *dataset* menjadi numbers antara 0 dan 1



**Fig. 18.** Data Preprocessing

**2.3.2 Time Stepping**

LSTM hanya dapat bekerja dengan format spesifik seperti array 3 dimensi. Oleh karena itu dataset dikemas ulang kedalam 60 timestep dan dirubah menjadi array menggunakan NumPy. Dataset akhir akan menjadi sebuah array 3 dimensi dengan X\_train, 60 timestamps, and satu fitur per *step*.



**Fig. 18.** Time Stepping

**2.3.3 Membangun LTSM Model**

Model dibangun menggunakan Layer LTSM dan kemudian diimplementasikan *dropout* layer untuk mengurangi kompleksitas. Berikut adalah parameter pembangunan model:

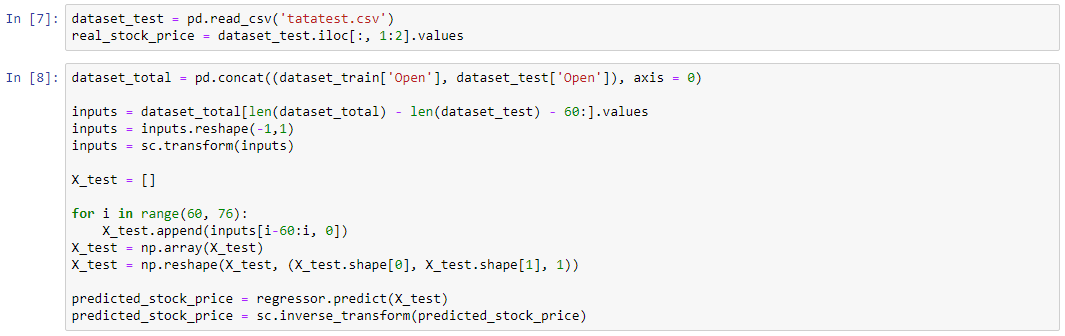
* 50 unit output untuk setiap layer
* return\_sequences=True
* input\_shape sebagai bentuk / *shape* dari *training set.*
* 20% Dropout Layers
* Gabungkan 50 unit output layer kedalam 1 output *dense layer*
* Model di *compile* menggunakan *Adam optimizer* menggunakan fungsi error *mean\_squarred\_error*.
* Lakkukan untuk 100 *epochs* dengan *batch size* = 32.



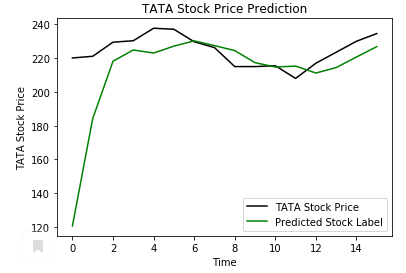
**Fig. 19.** Pembangunan Model

**2.3.4 Prediksi Stock**

Untuk melakukan prediksi atau *real-life* test, pertama *training set* dan *test set* digabung dengan dengan axis=0, lalu lakukan *timestepping* seperti sebelumnya dan *MinMaxScaler* untuk membangun *dataset* baru dan ubah menggunakan *inverse\_transform* untuk mengembalikan nilai stok dari 0..1 menjadi format yang seharusnya.



**Fig. 20.** Fungsi Prediksi Stok



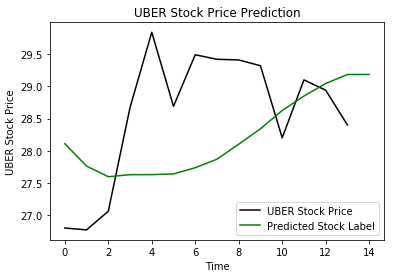
**Fig. 21.** Hasil Prediksi Stok

**2.4 Prediksi Stock berbasis LSTM RNN menggunakan dataset mandiri (UBER)**

**2.4.2 Perbedaan dengan prediksi stok Tata**

Dalam prediksi Stok UBER, diketahui terdapat 134 data dan digunakan 10 unit *timestepping. Test dataset* terdiri atas 14 data informasi stok terbaru.

**2.4.3 Hasil Prediksi**



**2.4.4 Analisis Hasil Prediksi**

Performa model prediksi stok UBER tidak sebaik dengan model prediksi TATA sebelumnya, hal ini dapat disebabkan oleh beberapa faktor antara lain:

1. Jumlah *train dataset* terlalu sedikit
2. Diperlukan optimisasi pada saat pembangunan LTSM RNN
   1. Jumlah Unit Output Space tidak optimal
   2. Untuk data yang sedikit mungkin tidak diperlukan Dropout Layer
   3. Jumlah Layer tidak optimal

**3 Kesimpulan**

RNN merupakan jenis ANN yang dapat digunakan ketika data terorganisir secara struktur waktu dan urutan data yang muncul dapat mempengaruhi jenis atau tipe data yang mungkin muncul berikutnya. Dalam pembelajaran Dasar Recurrent Neural Networks diketahui bahwa setiap RNN terdiri atas *cell-cell* yang menghasilkan output untuk dijadikan input oleh *cell* lainnya, RNN dapat memiliki sistem ‘memori’ menggunakan arsitektur LTSM sehingga dapat membantu masalah yang memerlukan konsep ‘memori’ layaknya manusia. Pada pembelajaran Character Level Language Model – Dinosaurus Island diketahui bahwa RNN cocok dalam membahas masalah mengenai NLP karena *dataset* dalam masalah NLP umunya terstruktur secara berurutan atau *time series.* Dalam pembelajaran prediksi stok berbasis LTSM RNN diketahui bahwa membangun ANN seperti RNN dapat mudah dilakukan dengan banyaknya tersedia *library* dan contoh kasus di dunia nyata dan untuk memperdalam kemampuan dalam pengetahuan *machine learning* sebaiknya mendalami bagaimana hasil model prediksi itu terbentuk dan bagaimana pendekatannya agar dapat meningkatkan performa dari model prediksi tersebut.

# Daftar Pustaka

|  |
| --- |
| [1] T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997. |
| [2] Kulbear, «Github,» 12 Juni 2018. [Çevrimiçi]. Available: https://github.com/Kulbear/deep-learning-coursera/blob/master/Sequence%20Models/Building%20a%20Recurrent%20Neural%20Network%20-%20Step%20by%20Step%20-%20v2.ipynb. [Erişildi: 8 Desember 2019]. |