**Study of CNN, RNN, and LSTM**

Setyo Legowo

Informatics – Institut Teknologi Bandung

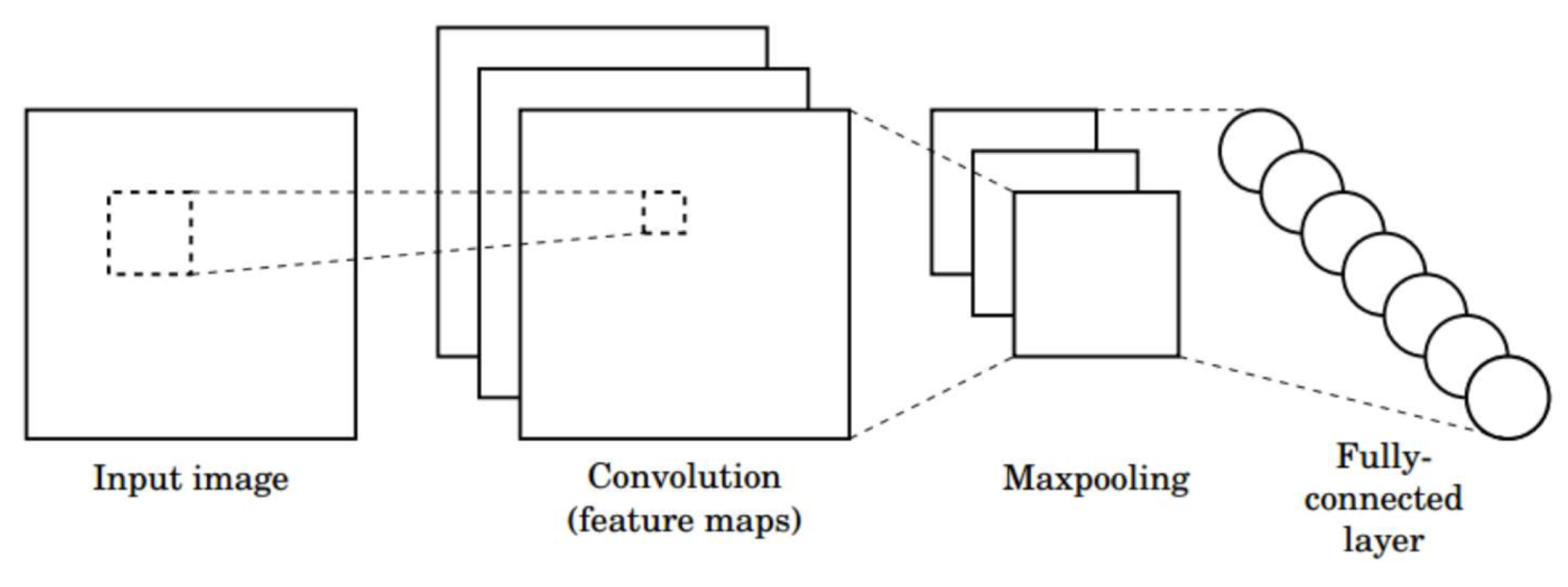
setyolegowo94@students.itb.ac.id

**1 Introduction**

In this report I will show what I have done in the last 5 weeks studying deep learning algorithm such as Convolutional Neural Network (CNN), Recurrence Neural Network (RNN), and also report about exercise using deep learning algorithm for stock market prediction and Dinosaurs name generation.

**2 Studying CNN**

In my opinion, CNN is one variation of neural network for learning instances that require multiple filters to recognize the pattern inside the instances. The algorithm usually used for image and speech recognition task. The filter itself is represented as weight in neural network however the “neuron” does not have activation function like in neural network.

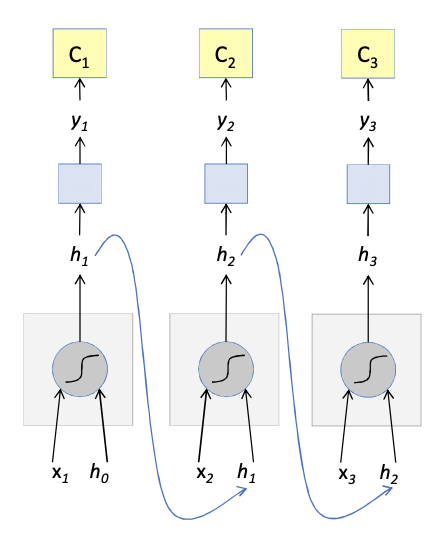


**Fig. 1.** Convolution process from an input image to feature maps

In the architecture of an CNN usually after convolution is pooling. Pooling is used to reduce complexity or down sampling operation to make the machine more robust to variance. There are 2 common types in pooling: max pooling and average pooling. In this step also do not have activation function.

**3 Studying RNN**

In my opinion, RNN is another variation of neural network for learning instances that behave like sequence or position of the object is more important. The algorithm usually used for prediction, classification, generator, and translation. This algorithm is better than ANN because this more robust to dependency between instance or part of the instance.



**Fig. 2.** Vanilla recurrence neural network

In RNN architecture, each part of the input and hidden state will be the input of an RNN cell. The RNN cell will output a hidden input for the next RNN cell.

**3.1 Studying LSTM**

One variance of RNN is LSTM (Long-Short Term Memory). The different is LSTM cell will track and update the state at every time-step.

**4 Exercising**

The task for exercise using deep learning algorithm mostly doing hands on to learn the real work of CNN, RNN, and LSTM in Python code. The given code is easy to copy but need more time to understand.

**3.1 Hands on of CNN**

An image is represented as 3D matrix with each dimension presented as height, width, and channels of image. There are some functions that needed for forwarding: (1) zero\_pad for add padding to the image; (2) conv\_single\_step is multiplying each matrix cell of 2 matrix, and then sum it and plus add bias number; (3) conv\_forward basically iterate the batch of image for convolution; and (4) pool\_forward basically iterate feature maps from the result of conv\_forward and doing pooling.

Backward learning in CNN is basically similar with ANN which calculate gradient to update weight/filter of convolution steps and also pooling. The only lack in 2 functions for update the calculate the gradient (conv\_backward and pool\_backward) is not using learning rate.

**3.2 Hands on of RNN and LSTM**

In this RNN cell is receive 2-dimension matrix represented as batch of inputs at a particular time and 2-dimensions matrix of previous hidden state. There are 2 weight matrices for input, 1 for input (Wax) and other for previous hidden state (Waa). Those inputs and weights then combined and add some bias. Then the result is used for activation function parameter than return hidden state for next cell. Another output from this cell which is the real output used output hidden state as parameter for SoftMax function before coming out from the cell. Therefore, each cell takes as input the hidden state from previous cell and current time step’s input data.

LSTM cell for this also receive 2 kinds of input like RNN and add another input that tracking the “cell state” or memory at particular time. There are 3 kinds of gate that use to retain/remove memory and update the outputs: forget gate, update gate and output gate. LSTM also have 5 types of weight of matrix: forget gate, update gate, first “tanh”, output gate and relating the hidden state to the output.

**3.3 Character level modeling - Dinosaurs**

We would like to build a name builder for Dinosaurs. To do that we should model it in character level. Basically, this modeling using vanilla model of RNN. Each hidden output of previous cell will be mapped with index of character as the character of the new name at particular position. In this task, the generated Dinosaurs names are limited with just one word only.

The training and generating process basically is just ordinary, such as preprocess to build character map to index and vice versa, repeat the training process in limited iteration, and look up how model generated the new word at every N iteration. The fun part from this task is seeing the result of the model because they are very funny like “knacalosaurus”, “plutosaurus” and “ndaairus”.

**3.4 Stock Market Prediction**

The final task is to create model for stock market prediction using Deep Learning, particularly LSTM. Data for stock market is abundant in Internet and some services provide API to retrieve the detail and also history of public-owned company.   
Table 1 is the example snapshot of stock market data from Yahoo! Finance.

**Table 1.** Example snapshot of stock market data Tata Global Beverages Limited (TATAGLOBAL.NS) from Yahoo! Finance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Close | Adj. Close | Volume |
| Dec 07, 2018 | 208.55 | 209.50 | 203.30 | 206.55 | 204.31 | 1,422,582 |
| Dec 06, 2018 | 210.50 | 211.65 | 207.00 | 208.15 | 205.89 | 1,270,790 |
| Dec 05, 2018 | 209.00 | 212.50 | 205.40 | 211.70 | 209.40 | 3,055,629 |
| Dec 04, 2018 | 214.75 | 216.75 | 208.80 | 209.50 | 207.23 | 1,908,684 |
| Dec 03, 2018 | 218.50 | 219.20 | 214.30 | 214.75 | 212.42 | 1,099,033 |

Stock market data basically have 6 main columns: (a) Date, (b) Open is price on beginning of the day, (c) High is the highest price on the day, (d) Low is the lowest price on the day, (e) Close is price at the end of market is closed on the day, and (f) Volume is how much stock is traded/splitted/etc. Normally stock market is open 5 days in a week or lesser which usually because of holiday.

Stock market data have sequence properties thus RNN algorithms is suitable for this case because the model would be smaller although more time would be needed. The model should be able to predict the price of the next day thus sequence classification architecture should be enough. In this case the predicted price on the next day will use Close price.

Preprocessing step in stock market modeling is create sample or data with timesteps to present sequence properties from the data. For example, using data in Table 1, for timesteps 3, each instance would be like in Table 2. Thus, the data is represented as 3D matrix of index of instance, timesteps, and stock market columns data.

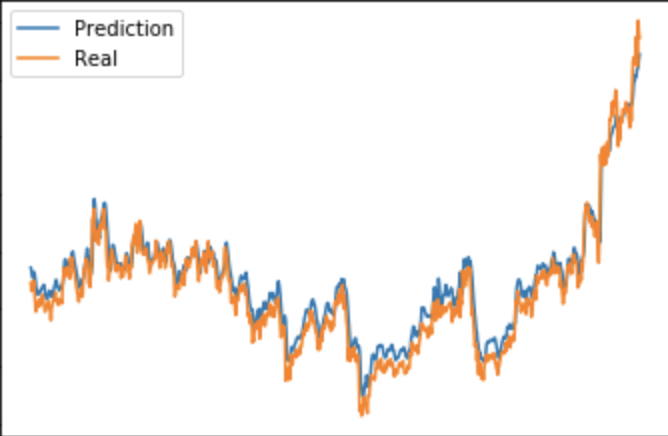
**Table 2.** An instance of stock market data in timesteps 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Close | Adj. Close | Volume |
| Dec 03, 2018 | 218.50 | 219.20 | 214.30 | 214.75 | 212.42 | 1,099,033 |
| Dec 04, 2018 | 214.75 | 216.75 | 208.80 | 209.50 | 207.23 | 1,908,684 |
| Dec 05, 2018 | 209.00 | 212.50 | 205.40 | 211.70 | 209.40 | 3,055,629 |
| “Prediction” price |  |  |  | 208.15 |  |  |

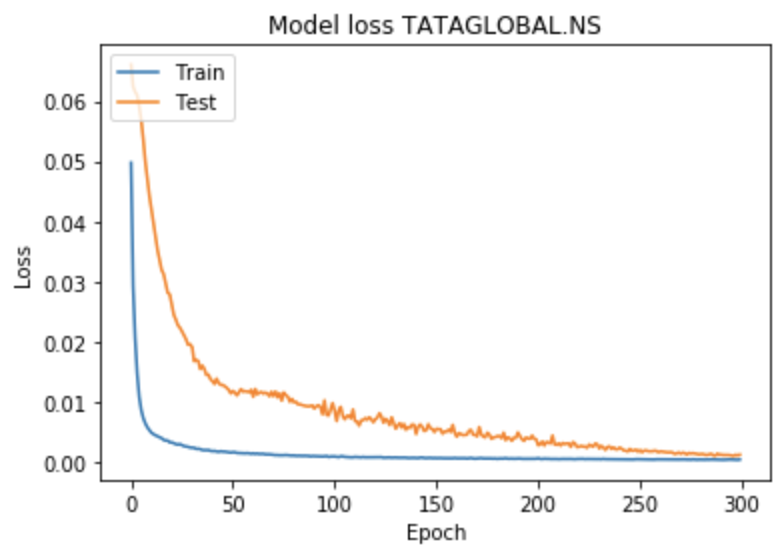
In order to build LSTM model, using Keras library, we will use several provided modules from the library. Like already mention in previous paragraph, we will use Sequential architecture or model. Beside add LSTM layer, we will add Dropout layer to reduce overfitting and Dense layer to use activation function. After building the architecture then we should choose optimization for updating the weights. Any optimizer should be worked, so we choose RMSprop as one alternative besides SGD.

To accelerate training time, we will batch the input. We will set 20 as default batch size. Then we will use 60 as default timesteps. We will use API from Yahoo! Finance to retrieve data, so we just input the stock market code that recognized by the service.

For example, to modeling prediction for Tata Global Beverages Limited stock price, we just need to provide stock market code of the company which is TATAGLOBAL.NS as input of run function. Then we automatically download the data, slicing and splitting it to timeseries data for training, validation, and test, create new model, training as much as 300 epochs, and finally test predictions against the expected result. Figure 3 is the example of final result between prediction and real stock price.

 **Fig. 3.** One example of final result between prediction and real stock price

After several time testing the model with variating the batch size, time steps and epoch, current configuration is the best. Small batch size and time steps failed to predict which closer to real price. However, the current size is big enough thus this model needs so much data, in this experiment eight years is good enough for training and testing. Small epoch size is not enough to make loss score from data validation is close to training loss score, as we can see in Figure 4. The model architecture can be different but at least have one LSTM layer and Dropout layer.

  
**Fig. 4.** Loss score between training and validation testing at each epoch.

**References**

1. Course slides from IF5181 Pattern Recognition, Institut Teknologi Bandung
2. Course slides from 6.S191 Intro to Deep Learning, Massachusetts Institute of Technology
3. Nayak, Asutosh. 2019. Predicting Stock Price with LSTM. <https://towardsdatascience.com/predicting-stock-price-with-lstm-13af86a74944>