2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems
Yogyakarta, Indonesia, November 08-10, 2017

Recurrent Neural Networks Model for WiFi-based Indoor Positioning System

Yuan Lukito
Program Studi Teknik Informatika
Universitas Kristen Duta Wacana
Yogyakarta, Indonesia
yuanlukito@ti.ukdw.ac.id

Antonius Rachmat Chrismanto
Program Studi Teknik Informatika
Universitas Kristen Duta Wacana
Yogyakarta, Indonesia
anton@ti.ukdw.ac.id

Abstract—**This research focus on the implementation of recurrent neural networks (RNN) model for indoor positioning system (IPS). Unlike global positioning system (GPS), IPS is used in closed structures such as hospitals, museums, shopping centers, office buildings, and warehouses. Positioning system is a key aspect in IPS. We propose, implemented, and evaluated an RNN model for positioning system. We used Wi-Fi-based IPS dataset from our previous research and made some comparison of RNN model performance with other methods. From the model evaluation results, we can conclude that RNN model is suitable for Wi-Fi-based IPS. It also produces generally higher accuracy compared with multi-layer perceptron model (MLP), Naïve Bayes, J48, and SVM. The RNN model training process still needs some tweaking on the parameters used in training.**

Keywords—recurrent neural networks; indoor positioning system; wi-fi.

I. Introduction

Global positioning system (GPS) is commonly used for determining our position on earth, giving navigation and orientation guidance. Widely used navigational applications such as Google Maps, Waze, TomTom, and Sygic rely heavily on GPS. GPS generally does not work in closed structures such as hospitals, museums, shopping centers, office buildings, and warehouses. Indoor Positioning System (IPS) can provides position, direction, and orientation in a closed structure [1].

A key aspect of IPS is positioning system. Every location-based service needs an accurate and reliable positioning system. There are many approaches that have been explored to build such system. Recent developments of IPS indicated that Wi-Fi-based IPS is the most widely used because of its pervasive usage everywhere. Many building managements provide Internet connection via Wi-Fi that installed in many places inside the building. This approach does not need additional hardware or special equipment.

Although Wi-Fi-based IPS is a valuable solution, it also has some major problems. For example, there are many variations of Wi-Fi hardware specifications. Another example is Wi-Fi access points are usually installed to provide Internet connection, not for IPS. Hence, it is not optimized for IPS. Wi-Fi based IPS can be viewed as a classification problem. The features commonly used for this classification problem is received signal strength (RSS). It consists of information about Wi-Fi signal strength, usually measured in dBm, gathered from surrounding Wi-Fi access points. Every room or place inside a building usually has different RSS, so it can be used to identify its position.

Neural networks is an excellent choice for classification problem. It has been used for many popular classification problems, such as handwriting recognition, self-driving car, speech recognition, and sentiment analysis. Our previous studies [2] [3] have reported that multi-layer perceptron is suitable for Wi-Fi-based IPS although it still needs some improvements.

Wi-Fi signal strength is not constant, it fluctuates from time to time depending on number of users connected to the Wi-Fi access points and interferences from other devices. In this research we explore and examine recurrent neural networks (RNN) for Wi-Fi-based IPS. Recurrent neural networks has recurrent connection between layers. It is able to learn spatial and temporal pattern because it stores current values for future references.

The aim of this paper is to build and examine the performance of RNN model for Wi-Fi-based IPS. We conducted experiments using our dataset generated from our previous research [2]. We also compared the result with our previous studies which use different methods.

II. Literature Review

A. Indoor Positioning System

Museum is a well-known example of a place where IPS implementation can be valuable. Visitors are guided by an application on their smartphone to navigate and find places inside a museum. The application gives contextual information about museum collections depends on the position of the visitors. Visitors can also search and get navigational guidance to a specific collection they are interested to. Another example of valuable IPS implementation is shopping center. Visitors can use an application installed on their smartphone to locate a specific tenant, store, or restaurant. The IPS system can also give contextual and localized advertisements to the visitors.

Determining physical location in a closed structure is the main problem of an IPS. A considerable amount of literature

This research was supported by Direktorat Riset dan Pengabdian Masyarakat, Direktorat Jenderal Penguatan Riset dan Pengembangan, Kementrian Riset, Teknologi dan Pendidikan Tinggi (Penelitian Dosen Pemula), No. 118/SP2H/LT/DRPM/IV/2017

has been published on IPS. There are many implementations of IPS using different approaches on hardware and methods such as Bluetooth [4], ultrasonic [5], visible light communication [6] Radio Frequency Identification (RFID) [7], surveillance camera [8] and Wi-Fi [9] [10] [11]. Wi-Fi signal is the most used approach because of its widespread availability of Wi-Fi access points in many buildings in the world.

Wi-Fi fingerprint is a set of RSS values that represents Wi-Fi signal characteristics at a particular location [11]. Although Wi-Fi fingerprinting technique is able to achieve high positioning accuracy, it requires a lot of laborious and time-consuming works to collect RSS values at all known locations. RSS can be collected in four different ways: (1) point-by-point calibration, (2) walking survey, (3) sensor-based labelling and (4) semi-supervised learning [12].

B. *Recurrent Neural Networks*

RNN is similar in many ways to a feedforward neural networks. The main difference is RNN does not always have forward connection. A recurrent connection connects a neuron from the current layer into a neuron in the previous layer or into the neuron itself. The recurrent connection allows the networks to make use of past context [13]. The example of a recurrent connection is shown in Fig. 1.

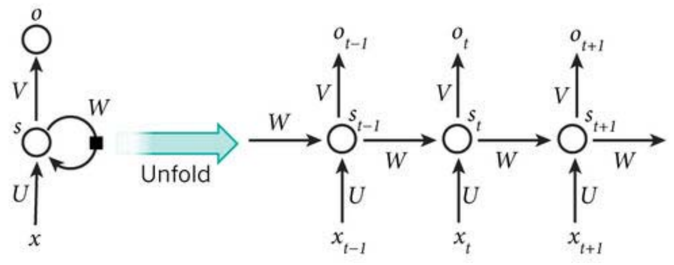

Fig. 1. Recurrent Connection and the Unfolding In Time [14].

RNN is used for many classification problems that involve sequential data, such as speech recognition [14], generating image caption [14], generating image [15], text to speech [16] and generating natural image [17].

C. *Tensorflow and TFlearn*

Tensorflow is an open source software library for numerical computation using data flow graphs [18]. The name “Tensorflow” is derived from the flow of tensors, which are multidimensional data arrays. It is widely used for machine learning and machine intelligence. Tensorflow is written in C++ and CUDA programming languages. It provides an interface to many other programming languages such as Python, Java, C++ and Go.

TFlearn is a wrapper library in Python programming language for Tensorflow designed to provide higher-level API to Tensorflow [19]. TFlearn can simplify code required to build a neural networks while still maintaining compatibility with Tensorflow.

III. METHODOLOGY

This research was conducted in the following steps illustrated in Fig. 2.

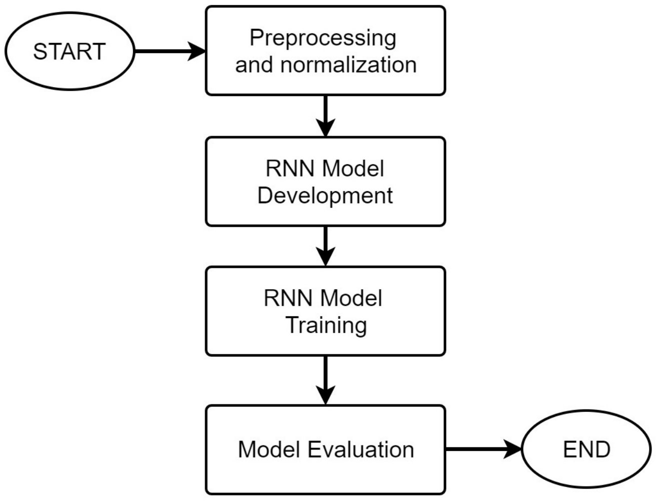

Fig. 2. Research Methodology.

In this research we used dataset from our previous research and added normalization step into the dataset. After the dataset has normalized, we developed the RNN model using Python and Tensorflow library to speed up the development process. We continued with training the RNN model and evaluated the model.

A. *Preprocessing and Normalization*

In this research we used Wi-Fi IPS dataset from our previous research for training and evaluation purpose. Table 1 shows some examples of our Wi-Fi dataset.

TABLE I. EXAMPLES OF DATASET USED IN THIS RESEARCH.

Place	**AP-1**	**AP-2**	**AP-3**	**…**	**AP-177**
Rektorat	-89	-100	-92	…	-84
Perpustakaan	-78	-80	-100	…	-100
…	…	…	…	…	…
Atrium D	-82	-100	-100	…	-89

There are 42 public places and 177 Wi-Fi access points inside UKDW campus recorded in our dataset. Every single record consists of place (label) and RSS values (177 features). RSS was measured in dBm. There are 11658 records on the dataset.

We normalized the RSS values using (1) to simplify and speed up training process. After this normalization process, every RSS value is in 0–1 range.

$$RSSn_i = \frac{(RSS_i + 100)}{60} \qquad (1)$$

The dataset was formatted in CSV file format with semicolon as separator. The first row indicates the columns name and the rest of the rows are the data. The place used for label is encoded as number in 0–41 range. Table 2 shows some example of the normalization results.

TABLE II. EXAMPLES OF NORMALIZATION RESULTS.

Place	AP-1	AP-2	AP-3	…	AP-177
0	0.183	0	0.133	…	0.267
1	0.367	0.333	0	…	0
…	…	…	…	…	…
41	0.300	0	0	…	0.183

The dataset was randomly divided into 10 groups to accommodate k-fold validation for model evaluation purpose. We encapsulated the dataset loading and normalization process into a class called WifiDataset shown in Fig. 3. The implementation was written in Python.

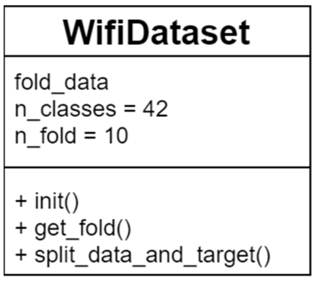

Fig. 3. Class Diagram of WifiDataset.

WifiDataset class provide three methods: *init(), get_fold()* and *split_data_and_target()*. The first method will be used for initializing the object, load dataset from CSV file into the memory and creating folds of data. Method get_fold() will be used for getting a group of data from a particular fold. The last method, *split_data_and_target()* is used for splitting a fold into data (features) and target (label).

B. Recurrent Neural Networks Model Development

The RNN model was developed in Python and encapsulated into a RNN class shown in Fig. 4.

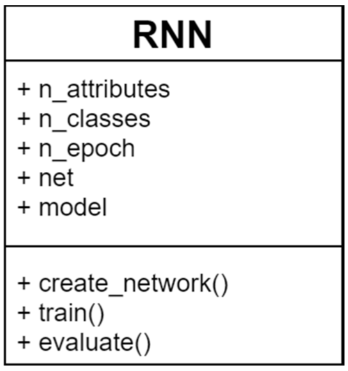

Fig. 4. Class Diagram of RNN.

RNN class provides three methods: *create_network()*, *train()* and *evaluate()*. The first method is used for initializing the model and creating the model. This method also initializing all the layers needed for the RNN model. The second method will be used for starting the training process and the last method is used for evaluation. Fig. 5 shows the steps of the RNN model implementation and evaluation.

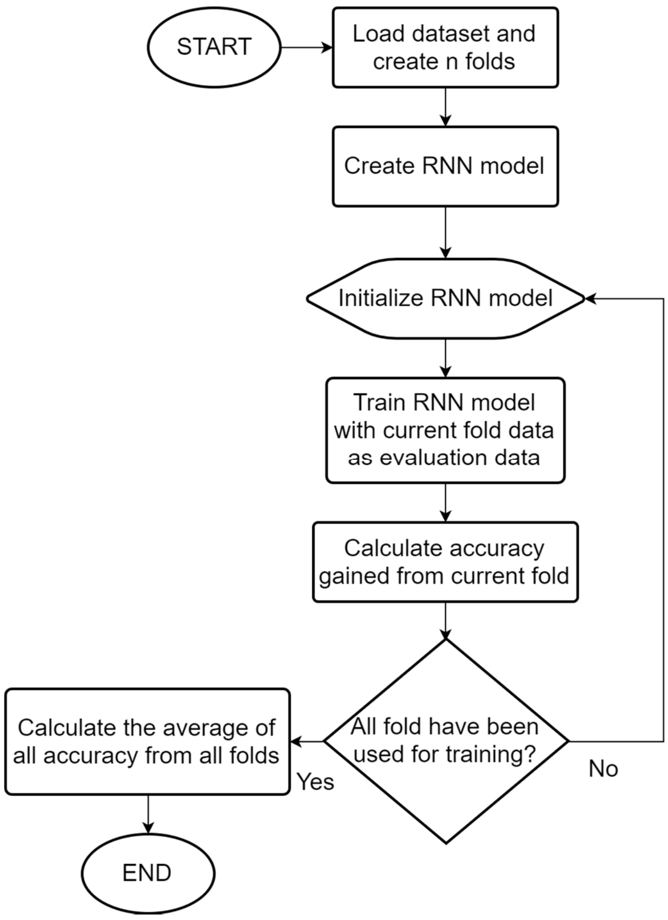
START
Load dataset and create n folds
Create RNN model
Initialize RNN model
Train RNN model with current fold data as evaluation data
Calculate accuracy gained from current fold
All fold have been used for training?
No
Yes
Calculate the average of all accuracy from all folds
END

Fig. 5. RNN Model Implementation

C. Model Training and Evaluation

The parameters used for RNN model training are shown in Table 3.

TABLE III. PARAMETERS USED FOR TRAINING AND EVALUATION

Parameter	Value
Class count	42 classes
Recurrent layer count	2 layers
Recurrent layer type	Elman Simple RNN
Neurons in the recurrent layer	128 neurons
Neurons in the input layer	177 neurons
Optimization method	Adam Optimizer
Learning rate	0.0025
Number of epochs	1000 epochs
K in k-fold validation	10 folds

We used k-fold validation for our RNN model evaluation. Our model is constructed with two fully connected recurrent layers with softmax activation function. We also used categorical cross entropy between predictions and targets as loss function for training and validation. The snippet of the code to create, initialize, train, and evaluate the RNN model is listed below.

```
# input layer
self.net = tflearn.input_data(shape=[None, 1,
self.n_attributes])

# recurrent layers
self.net = tflearn.simple_rnn(self.net, 128,
return_seq=True, bias=True)
self.net = tflearn.simple_rnn(self.net, 128)

# fully connected
self.net = tflearn.fully_connected(self.net,
self.n_classes, activation='softmax')

# categorical cross entropy and adam optimizer
self.net = tflearn.regression(self.net,
optimizer='adam',loss='categorical_crossentropy',
name="output1", learning_rate=0.0025)

# Tensorboard for logging and monitoring
self.model = tflearn.DNN(self.net, tensorboard_verbose=3)

# Start training
self.model.fit(train_data_array, train_target_array,
n_epoch=self.n_epoch, validation_set=0.1,
show_metric=False, snapshot_step=1000)

# Evaluation
accuracy = self.model.evaluate(test_data_array,
test_target_array)
```

IV. Results And Discussion

We have trained and evaluated the RNN model, the results are shown in Table 4.

Fold	Accuracy (%)	Fold	Accuracy (%)
1	82.78	6	82.25
2	82.95	7	80.94
3	83.91	8	82.78
4	82.25	9	81.99
5	82.25	10	82.53
Average		**82.47%**	

For every fold, the training process took about four hours to complete because we only used CPU. To complete the training and evaluation of all the folds, it took about 40 hours. The average of the achieved accuracy of our RNN model is 82.47%. This result is higher compared to our previous research result using multi-layer perceptron. The comparison of RNN model and other methods from our previous research is shown in Fig. 6.

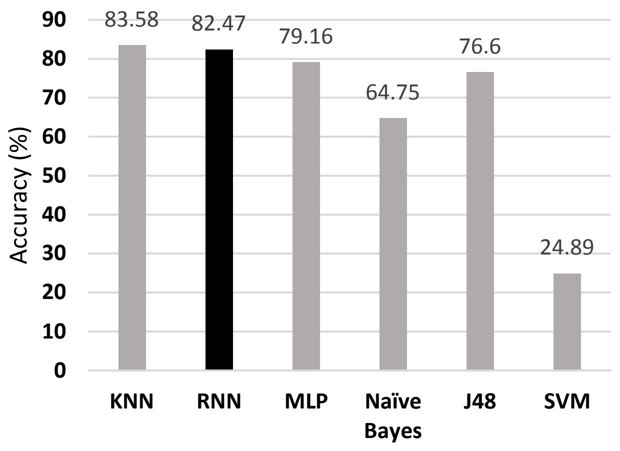

Fig. 6. Comparison of RNN Model and Our Previous Research Results

Based on our finding, RNN model produces a better accuracy compared to Multi-layer perceptron, Naïve Bayes, J48 and SVM. K-Nearest Neighbors still produces the highest accuracy. We think that with some tweaking on parameters used in training, our RNN model can surpass in term of accuracy. The training process took a long time because of loss value fluctuated as shown in Fig. 7.

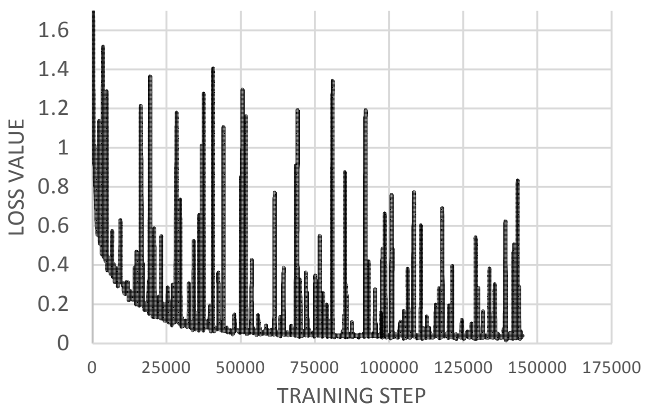

Fig. 7. Loss Value during RNN Model Training

There are some alternatives of recurrent layer such as long short term memory (LSTM) and gated recurrent unit (GRU) that have the abilities to reduce the effect of fluctuated loss value. In the near future we will implement our model with LSTM and GRU.

V. Conclusions And Suggestions

We have successfully developed RNN model, trained and evaluated the model. Based on the experiment results, we can conclude that RNN produces better accuracy compare to other methods such as multi-layer perceptron, Naïve Bayes, J48, and SVM. Our model still needs some tweaking on training parameters to be able to surpass K-Nearest Neighbors performance. The performance of our model is affected by the fluctuation of the loss value during training. We suggest using LSTM or GRU to improve the performance of the model.

ACKNOWLEDGMENTS

This research was supported by Direktorat Riset dan Pengabdian Masyarakat Direktorat Jenderal Penguatan Riset dan Pengembangan, Kementerian Riset, Teknologi dan Pendidikan Tinggi (Penelitian Dosen Pemula), No. 118/SP2H/LT/DRPM/IV/2017. We would also like to show our gratitude to Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) and Fakultas Teknologi Informasi (FTI) Universitas Kristen Duta Wacana.

REFERENCES

[1] A. Ghose, A. Pal, A. D. Choudhury, V. Chandel and T. Chattopadhyay.United States of America Patent US9557178 B2, 2017.

[2] Y. Lukito and A. R. Chrismanto, "Perbandingan Metode-Metode Klasifikasi Untuk Indoor Positioning System," *Jurnal Teknik Informatika dan Sistem Informasi,* vol. 1, no. 2, pp. 123-131, 2015.

[3] Y. Lukito, "Model Multi Layer Perceptron Untuk Indoor Positioning System Berbasis Wi-Fi," *Jurnal Teknologi dan Sistem Komputer,* pp. 123-128, 2017.

[4] A. Ozer and E. John, "Improving the Accuracy of Bluetooth Low Energy Indoor Positioning System Using Kalman Filtering," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, USA, 2016.

[5] X. Chen and Z. Gao, "Indoor ultrasonic positioning system of mobile robot based on TDOA ranging and improved trilateral algorithm," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, Chengdu, China, 2017.

[6] H. Lv, L. Feng, A. Yang, P. Guo, H. Huang and S. Chen, "High Accuracy VLC Indoor Positioning System With Differential Detection," *IEEE Photonics Journal,* vol. 9, no. 3, 2017.

[7] W. An, Z. Shen and J. Wang, "Compact Low-Profile Dual-Band Tag Antenna for Indoor Positioning Systems," *IEEE Antennas and Wireless Propagation Letters,* pp. 400-403, 2016.

[8] L.-W. Chen, C.-R. Chen and D.-e. Chen, "VIPS: A video-based indoor positioning system with centimeter-grade accuracy for the IoT," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, USA, 2017.

[9] N. L. Dortz, F. Gain and P. Zetterberg, "Wi-Fi Fingerprint Indoor Positioning System Using Probability Distribution Comparison," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[10] O. Dousse, J. Eberle and M. Mertens, "Place Learning via Direct Wi-Fi Fingerprint Clustering," in *2012 IEEE 13th International Conference on Mobile Data Management (MDM)*, Bengaluru, India, 2012.

[11] A. Ismail, H. Kitagawa and R. Tasaki, "WiFi RSS fingerprint database construction for mobile robot indoor positioning system," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, 2016.

[12] S.-H. Jung, G. Lee and D. Han, "Methods and Tools to Construct a Global Indoor Positioning System," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. PP, no. 99, pp. 1-14, 2017.

[13] A. Graves, M. Liwicki and S. Fernandez, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 5, pp. 855-868, 2009.

[14] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[15] A. Graves, A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, Canada, 2013.

[16] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang and A. Yuille, "Deep Captioning With Multimodal Recurrent Neural Networks (M-RNN)," in *3rd International Conference on Learning Representation*, San Diego, USA, 2015.

[17] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[18] Y. Fan, Y. Qian, F. Xie and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *15th Annual Conference of the International Speech Communication Association*, Singapore, 2014.

[19] A. V. Oord, N. Kalchbrenner and K. Kavukcouglu, "Pixel Recurrent Neural Networks," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[20] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[21] TFLearn.org, "TFLearn," TFlearn.org, [Online]. Available: http://tflearn.org/. [Accessed 15 August 2017].