2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

Evaluation of Recurrent Neural Network Models for Parkinson's Disease Classification Using Drawing Data

Arjun Shenoy A V, Michael A. Lones, Stephen L. Smith, Marta Vallejo

Abstract— **Parkinson's disease is a disorder that affects the neurons in the human brain. The various symptoms include slowness of motor functions (bradykinesia), motor instability, speech impairment and in some cases, psychiatric effects such as hallucinations. Most of these, however, are also common side effects of natural aging. This makes an accurate diagnosis of Parkinson's disease a challenging task. Some breakthroughs have been made in recent years with the help of deep learning. This work aims at considering figure drawing data as a time series of coordinates, angles and pressure readings to train recurrent neural network models. In addition, the work compares two recurrent network models, Long Short-Term Memory and Echo State Networks, to explore the advantages and disadvantages of both architectures.**

I. Introduction

Parkinson's disease (PD) is a neurodegenerative syndrome characterized by the loss of dopaminergic neurons in the substantia nigra pars compacta region of the midbrain and the formation of filamentous inclusions (Lewy bodies) [1]. PD is the second most common neurodegenerative disease after Alzheimer's disease and is particularly common in industrialized nations, where it can be found in approximately 0.3% of the population. Their symptoms can be divided into motor and non-motor. Motor symptoms include slowness of movement (bradykinesia), stiffness and inflexibility in muscles (rigidity) and tremor [2]. Non-motor symptoms include depression, insomnia, dementia, anosmia, pain, and constipation, among others [3]. Various studies have been conducted to determine the aetiology of the disease, but its exact cause is still unknown [4].

Currently, there is no cure or treatment to stop the neurodegeneration process [5]. Instead, treatments, such as levodopa, focus on improving the quality of life of the patients [6]. Proper diagnosis of the disease is, however, a challenging task. The non-motor symptoms are often poorly recognized and inadequately treated [3]. Motor-based symptoms can be indicators of other neurological conditions. Furthermore, the same symptoms can be interpreted differently depending on the examiner's prior experience and training [7].

With the advent of machine learning, more systematic diagnosis techniques have been proposed. Vallejo *et al.*[8] applied evolutionary algorithms on drawing data to diagnose motor and cognitive deficiencies in PD. Oh *et al.*[9] used Convolutional Neural Networks (CNNs) to analyze electroencephalogram signals for PD abnormalities. [10] detailed the use of CNNs on movement data collected from PD subjects to detect freezing of gait. [11], [12] are examples of using Recurrent Neural Networks (RNNs) for the diagnosis of PD based on motor symptoms and drawing data.

In this paper, we evaluate the effectiveness of RNN architectures to distinguish PD patients from healthy subjects using drawing data collected from tablets. Specifically, we:

- analyze figure drawing data as a time series instead of considering them as image data.
- evaluate two RNN architectures, namely Long-Short Term Memory (LSTM) and Echo State Networks (ESN), on how effective they perform the classification.
- compare performance metrics with those gathered from an image-based CNN approach on the same dataset[13].

II. Theoretical Background

A. Recurrent neural networks

RNNs are models in which there is a path from a given node back to itself or to other nodes in the previous or the same layer [14]. Recurrent connections imply that the output at any given timestep not only depends on the input but also on previous output(s). RNN's ability to refer back to previous outputs enables the learning of temporal relationships.

RNNs have a cardinal pitfall when it comes to setting the weights of the recurrent connections. Traditional training algorithms such as backpropagation or real-time recurrent learning tend to be too slow compared to the time gap between the input sequence entries. Furthermore, the error signals that are propagated backwards, as a part of the training, tend to either explode (exploding gradient) or become extremely small (vanishing gradient) The former situation leads to a situation of oscillating weights, while if the latter situation occurs, the network could take an indefinite amount of time to train the weights, or in some cases, fail to train properly [15]. As a solution to the vanishing gradient problem, Hochreiter *et al.*[15] proposed the LSTM architecture, which was designed to remember long sequences. An individual LSTM maintains a memory cell with multiple gates inside of it to remember appropriate information from previous inputs. The content inside the memory cell is updated only when necessary [16].

B. Reservoir Computing

Reservoir computing is another set of computational architectures derived from RNN theory. A reservoir computing

[1]Arjun Shenoy is with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK aa507@hw.ac.uk
[2]Michael A. Lones is with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK m.lones@hw.ac.uk
[2]Stephen L. Smith is with the Department of Electronic Engineering, University of York, UK stephen.smith@york.ac.uk
[3]Marta Vallejo is with the School of Engineering and Physical Science, Heriot-Watt University, Edinburgh, UK m.vallejo@hw.ac.uk

model consists of a set of nodes (reservoir), which maps a given input into a high dimensional space and a set of output nodes (readout nodes), which read this high dimensional representation and map it into the expected output [17]. The nodes in the reservoir are sparsely interconnected, and the weights between them are fixed before training. The only training involved is that of the readout nodes, which is usually done by a computationally light method such as linear or ridge regression [17]. This computational efficiency is one of the key benefits that have attracted research in this field. Two popular implementations of reservoir computing are the ESN [18] and the Liquid State Machine (LSM). The ESN typically uses sigmoid nodes in the reservoir, while LSMs usually use spiking neurons. The ESN primarily consists of three sets of nodes:

1) Input nodes: They take the input signal and transfer it to the nodes in the reservoir. The weights of the connection between the input nodes and the reservoir (W_{in}) are fixed during initialization and are not trainable.
2) Reservoir: A set of sparsely interconnected nodes that operates as a typical RNN. The inter-connectivity of nodes is typically kept at 1%, which allows the division into a set of loosely connected sub-networks [19]. The number of nodes in the reservoir depends upon the task at hand. The weights of the synapses that connect the nodes in the reservoir (W) are not trainable but set during initialization and scaled by a constant known as the "spectral radius". This ensures that the reservoir satisfies the Echo State Property (ESP), which means that when provided with an input signal, it becomes excited and develops a lingering internal response signal that is perceived as a high dimensional non-linear transformation of the input [20].
3) Output nodes: They map the high-dimensional non-linear response from the reservoir to the desired output. The weights of the connections between the reservoir and these nodes (W_{out}) are randomly initialized and are modified during training such that a linear combination of the high-dimensional reservoir response maps to the expected output values.

C. Data augmentation

Data augmentation consists of a collection of techniques used to enhance the size and quality of data available to train a model. This increase results in greater performance of the model. Shorten and Khoshgoftaar [21] described various augmentation methods that can be applied on image data. In time series, however, the data is an ordered sequence of entries, and operations such as rearranging parts of the input would not guarantee the validity of the resulting samples. We list some transformations that can be applied to time series data:

1) Translation: Since in an image dataset, adding a constant to the coordinates only displaces the drawing without causing any label altering effect, adding a constant to the time series data can be considered as a new valid sample.
2) Scaling: Changing the magnitude of the data by multiplying it by a random scalar value.
3) Jittering: Adding minor fluctuations in smooth intervals to mimic the presence of noise.

III. Methodology

A. Data preparation

The data was obtained from the Leeds Teaching Hospitals NHS Trust[1] [13] from 87 subjects: 58 patients and 29 healthy controls. The controls were friends and relatives of the patients of similar ages, and were only included if they had no neurological disorder. The subjects were asked to draw an Archimedean spiral pentagon (see Fig. 1), using an inking stylus on a pressure-sensitive tablet.

Fig. 1 Pentagon drawing

Each subject performed the drawing four times, two times with each hand (dominant and non-dominant hand). We refer to each successful attempt at drawing the figure with a given hand as an *iteration* of drawing the figure. We worked with the first iteration of the pentagon figure drawn using the dominant hand, as this was the primary subset used in [13], and our aim was to use the results from this earlier work as a benchmark to evaluate the performance of our models. We refer to this particular subset of data as the ***first set of figure drawing data***. In certain scenarios, we also used the second iteration of pentagon drawings using the dominant hand, called the ***second set of figure drawing data***.

The dataset consists of four attributes, namely the xy co-ordinates, the angles made by the pen, the pressure readings, and the timestamp for each reading. We implemented a data cleaning and manipulation step in order to prepare the data before passing it to the model. The sampling rate was fixed at 200 Hz, and xy coordinates were rescaled back to the tablet's true dimensions $(20.3 \times 32.5cm)$. The initial dataset was prepared using the *first set of figure drawing data* as described above and split into training and testing such that the data pertaining to 20% of the subjects was reserved solely for testing. The rest was used for training. An equal split of patients and controls was preserved in each set. The training set was then broken down again, such that a fragment of it would be used for validation (75% training, 25% validation).

Since the number of controls was significantly lower than the patients, two approaches boosted the number of control samples. The first was to append the training dataset with a sample of control data from the *second set of figure drawing data*. The second approach was to use data augmentation

[1]This study received UK National Regional Ethics Service approval and local Research and Development approval from Leeds Teaching Hospitals NHS Trust. All participants had provided informed consent.

techniques (scaling, jittering and translation) on the control samples of the training dataset and then append the newly obtained instances back to the training set. The dataset obtained from both approaches was passed to the model.

B. LSTM configuration

The LSTM classifier [22] (Fig. 2), selected by an empirical process, consisted of a layer of two LSTMs stacked consecutively, followed by a dropout layer (0.5 probability), a flatten layer (20 nodes) and, finally, a sigmoid activation function. The weights of the network were trained using backpropagation with Adam optimizer using binary cross-entropy [23] as the loss function. The learning rate was kept as 0.005. The validation was done after every three rounds of training. The process of training and validation used the full training and validation datasets, and the set of values reporting a minimum validation loss was chosen for testing.

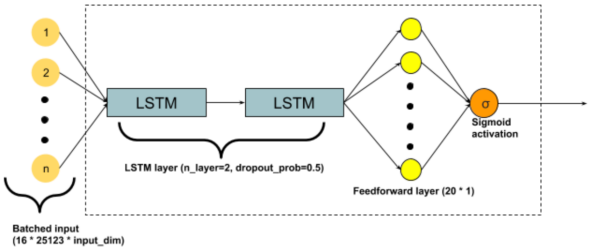


Fig. 2 Architecture of the LSTM based classifier

During the configuration stage, we analyzed the following hyperparameters:

- Number of LSTM units in the layer.
- Dimensionality of the output from the LSTM layer, number of nodes in the dropout and feedforward layers.
- Number of items passed to the model in a single epoch.
- Number of iterations for training and validation.

We ran multiple experiments to determine the ideal values for each of these. A single experiment involved setting particular values to the hyperparameters, running the training and testing modules and recording the performance metrics. We arbitrarily decided to pass the time series of only the xy coordinates to the model during configuration. Table I describes the best set of hyperparameters.

Table I: LSTM Hyperparameters	Value
Number of layers	2
Number of nodes	20
Batch size	16

The hidden and cell states were set as tensors, whose shape depended on the number of layers, nodes, and batch size. We initialized them randomly from a normal distribution. Each time the LSTMs were initialized with different hidden and cell state values, and the overall performance of the model varied significantly. We re-ran the experiments multiple times to allow the tracking of the optimum combination of configurations using performance metrics. After multiple iterations, we were able to find a reasonable set of values for the initial hidden and cell states.

C. ESN configuration

The ESN classifier is based on [24] and [25]. It consists of a set of input nodes, which pass the input in batches to the reservoir. The activation of RNN cells is configurable, but we kept the default *tanh* activation function. The reservoir is followed by a layer of output nodes, and the output from this layer is finally passed through a sigmoid activation function to limit the output values to the range [0,1]. The weights between the input nodes and the reservoir and within the reservoir were fixed at the time of initialization. Training the ESN involves passing batches of inputs to the model and then fitting the weights between the reservoir and the output nodes. The algorithm used for training the output weights and the related parameters is also configurable. Validation was done after every three rounds of training.

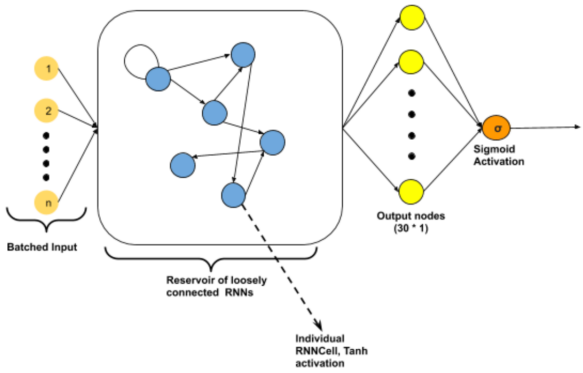


Fig. 3 Architecture of the ESN based classifier

Similar to the LSTM classifier, we ran multiple experiments to find the ideal configuration. These parameters are:

- The dimensionality of the output from the reservoir, also defined as the number of features in the hidden state.
- Number of initial timesteps of the reservoir output to be ignored. In other words, the portion of initial reservoir output which is not forwarded to the output nodes.
- Regularization parameter for ridge regression.
- Number of layers inside the reservoir.
- Spectral radius: described in Section II.
- The training algorithm to be used.
- The number of training-validation epochs.

In the case of the dimensionality of the output, we observed that the performance of the model initially improved as the size of the hidden layer approached 30. Beyond 30, it plateaued until reaching 70, beyond which it degraded. In the case of lambda, the use of smaller values had no effect on the performance of the model, while increasing it beyond a certain threshold caused the performance to degrade. We decided to set a value of 0.02 for this parameter.

We also observed that setting up the ESN with a washout of 0.5, hidden dimensionality of 30, and a regularization of 0.02 gave us a near-optimum performance (an F1-score of

96%). Similarly, decreasing the spectral radius or increasing it to values very close to 1 caused a decrease in the performance metrics. In the experiments we conducted, the model used Cholesky decomposition [26].

IV. RESULTS

A. LSTM Classifier

We initially performed the experiments with the first set of drawing data. However, the model produced an accuracy of 0.75, sensitivity of 1 and specificity of 0. In other words, regardless of the configuration, the model kept classifying every instance as a patient.

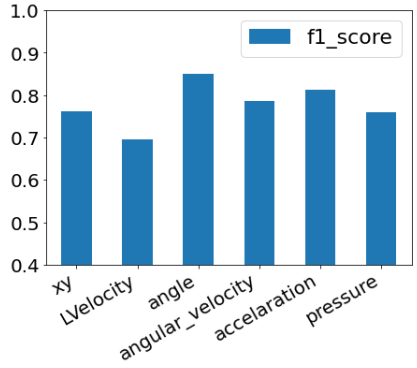


Fig. 4 Mean F1-scores using dominant hand second iteration

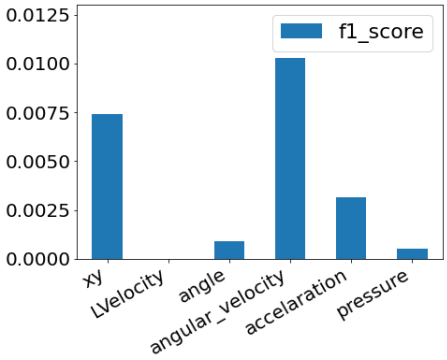


Fig. 5 F1-score variance on dominant hand second iteration

As mentioned before, we balanced the dataset in two ways, by using the second set of figure drawing data and by applying basic data augmentation techniques. The initial configuration was determined using only the xy coordinates. Once the ideal values for all the hyperparameters were determined, experiments were carried out with other parameters, including angle, pressure, velocity and acceleration.

Fig. 4 and 5 show that when trained using a time series that included the pressure and angles, the model had a high performance (mean F1-score 76%-80%) and high stability (low variance in F1-score 0.05%-0.09%). Following this observation, we searched for other combinations that resulted in better classification performance. We observed a marginally higher performance when combining pressure with velocity, angles and angular velocity. In the case of combining xy coordinates along with pressure, the variance of the performance metrics decreased sharply. Overall, the use of pressure by itself provided the best results so far, with an average F1-score of 0.945 and a variance of 0.04%.

B. ESN Classifier

In the case of ESN, we experimented solely with the pressure, as we had previously determined from the LSTM experiments that this information provided the best performance. The experiments performed and the metrics observed were outlined in Section III-C.

Table II: Results from the ESN classifier

Iteration	Accuracy	F1-Score
1	0.8125	0.8889
2	0.75	0.8333
3	0.9375	0.96
4	0.9375	0.96
5	0.75	0.8333

To evaluate the stability of the ESN, we set the configurations of the ESN (dimensionality=30, lambda=0.02, washout=0.05, epochs=10, spectral radius=0.9, number of layers=1, readout="cholesky") and ran experiments with the same datasets five times to see if the performance varied. Table III summarizes the results from the best configurations of LSTM and ESN models as well as the findings from [13].

Table III: Comparison of results with [13]

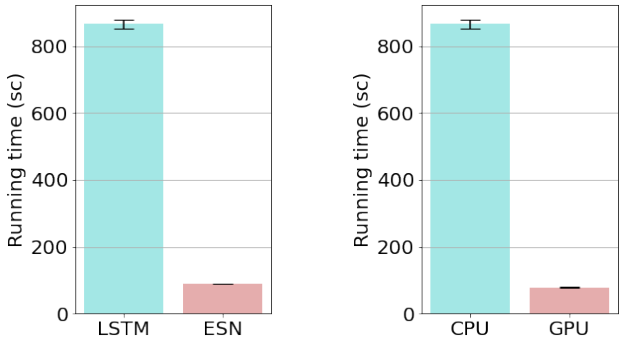
Model	Accuracy	Sensitivity	Specificity	F1-Score
CNN	95%	0.95	0.95	0.95
LSTM	91%	1.0	0.65	0.945
ESN	93.7%	1.0	0.75	0.96

C. Comparing training and testing time

In order to evaluate the training and test time of the models, we measured the time taken to run a single experiment (fixed number of epochs of training and validation followed by one round of testing). We fixed the configurations of the LSTM and ESN (described in the respective configuration sections). The data and the number of training-validation epochs were also fixed to a value of 10. The hardware was kept the same (a single CPU for each of the models). Fig. 6(a) compares the average running times of both models. ESN takes significantly less time compared to LSTM, even when run on a standard CPU machine. Further, we also found that using GPU processing greatly enhanced the performance of the LSTM model as reflected in Fig. 6(b).

V. CONCLUSIONS

One of the objectives of this work was to compare the results obtained using RNN-time series with those gathered by using CNN-image data [13]. We can see that the CNN image-based approach is more balanced and does not have any bias towards any particular label (both sensitivity and specificity are 0.95). However, we should also note that the



(a) LSTM and ESN on CPU (b) GPU-CPU for LSTM

Fig. 6 Average running time of LSTM and ESN

work in [13] also included extensive use of data augmentation techniques. The metrics obtained from LSTM and ESN classifiers are within a close range of these values. This points towards the potential of approaching figure drawing data from a time series perspective. We also observed that attributes such as pen angles and pressure had a high positive impact on the classification performance of our models. The image-based approach did not consider the pen angle, which could improve their final performance. Additionally, training a CNN is a very slow process. ESNs could significantly reduce training time whilst approaching similar accuracies.

The performance metrics collected from the LSTM classifier had, in many cases, high variance, which imply that the LSTM models were very sensitive to the hyperparameter configurations and to the order of the input data during training and validation. On the other hand, the performance of the ESN classifier showed relatively less variance, which shows that it is less sensitive to the order of the input data. In terms of operational cost, LSTM required GPU accelerated machines in order to perform at the same level of ESN on CPU. This means that ESN models are more economical compared to LSTM. Due to the fact that most of the weights in the ESN are fixed during initialization has this positive effect. Learning only occurs in the output weights, which itself is done using techniques that are computationally less intensive than backpropagation used in LSTM.

Despite its advantages, the ESN model does suffer from a major pitfall. Currently (July 2021), there is no official in-built library support in any of the main frameworks (PyTorch, TensorFlow, and Keras). On the contrary, the LSTM is a very popular RNN model and has a ready-to-use implementation and documentation in most frameworks. Due to the lack of extensive documentation, the setup of the ESN classifier was challenging. More community support and documentation might encourage researchers to use this particular architecture. Further investigation is also needed in the hypersensitivity of LSTM models and in finding proper ways to tune each of the parameters of the ESN model.

REFERENCES

[1] M. B. Feany and W. W. Bender, "A drosophila model of Parkinson's disease," *Nature*, vol. 404, pp. 394–398, 2000.

[2] P. Marios, W. Kit, M. Sophie, B. Peter G, C. K. Ray, and P. Piccini, "Parkinson's disease symptoms: The patient's perspective," *Movement Disorders*, pp. 1646–1651, 2010.

[3] K. R. Chaudhuri, D. G. Healy, and A. H. Schapira, "Non-motor symptoms of Parkinson's disease: diagnosis and management," *The Lancet Neurology*, vol. 5, no. 3, pp. 235–245, 2006.

[4] L. M. de Lau and M. M. Breteler, "Epidemiology of Parkinson's disease," *The Lancet Neurology*, pp. 525–535, 2006.

[5] S. Fahn and D. Sulzer, "Neurodegeneration and neuroprotection in Parkinson's disease," *American Society for Experimental NeuroTherapeutics*, pp. 139–154, 2004.

[6] P. LeWitt, "Levodopa for the treatment of Parkinson's disease," *New England Journal of Medicine*, vol. 359, no. 23, pp. 2468–2476, 2008.

[7] B. Post, M. P. Merkus, R. M. de Bie, R. J. de Haan, and J. D. Speelman, "Unified Parkinson's disease rating scale motor examination: are ratings of nurses, residents in neurology, and movement disorders specialists interchangeable?," *Movement disorders: official journal of the Movement Disorder Society*, vol. 20, no. 12, pp. 1577–1584, 2005.

[8] M. Vallejo, S. Jamieson, J. Cosgrove, S. L. Smith, M. A. Lones, J. E. Alty, and D. W. Corne, "Exploring diagnostic models of Parkinson's disease with multi-objective regression," in *Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2016.

[9] S. L. Oh, Y. Hagiwara, U. Raghavendra, R. Yuvaraj, N. Arunkumar, M. Murugappan, and U. R. Acharya, "A deep learning approach for Parkinson's disease diagnosis from eeg signals," *Neural Computing and Applications*, pp. 1–7, 2018.

[10] C. R. Pereira, S. A. Weber, C. Hook, G. H. Rosa, and J. P. Papa, "Deep learning-aided Parkinson's disease diagnosis from handwritten dynamics," in *29th Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 340–346, IEEE, 2016.

[11] L. C. Ribeiro, L. C. Afonso, and J. P. Papa, "Bag of samplings for computer-assisted Parkinson's disease diagnosis based on recurrent neural networks," *Computers in biology and medicine*, vol. 115, p. 103477, 2019.

[12] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep echo state networks for diagnosis of parkinson's disease," in *26th European Symposium on Artificial Neural Networks*, pp. 397–402, 2018.

[13] M. Alissa, M. A. Lones, J. Cosgrove, J. E. Alty, S. Jamieson, S. Smith, and M. Vallejo, "Parkinson's disease diagnosis using convolutional neural networks and figure-copying tasks," *TechReport*, 2021.

[14] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Advances in Psychology*, vol. 121, pp. 471–495, Elsevier, 1997.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *25th International Joint Conference on Artificial Intelligence*, pp. 2873–2879, 2016.

[17] T. Gouhei, Y. Toshiyuki, H. Jean Benoit, N. Ryosho, K. Naoki, S. Takeda, N. Hidetoshi, N. Daiju, and H. Akira, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[18] S. E. Lacy, S. L. Smith, and M. A. Lones, "Using echo state networks for classification: A case study in Parkinson's disease diagnosis," *Artificial Intelligence in Medicine*, pp. 53–59, 2018.

[19] J. Herbert and H. Harald, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science no: 5667*, vol. 4, pp. 78–80, 2004.

[20] G. Manjunath and H. Jaeger, "Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks," *Neural computation*, vol. 25, no. 3, pp. 671–696, 2013.

[21] C. Shorten and T. Khoshgoftaar, "Survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 48, 2019.

[22] G. Loye, "Lstm: From zero to hero with Pytorch," 2020.

[23] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2019.

[24] S. Nardo, "pytorch-esn," *Github*, 2018. Accessed on February, 2021.

[25] C. Gallicchio and A. Micheli, "Deep echo state network (DeepESN): A brief survey," *arXiv preprint arXiv:1712.04323*, 2017.

[26] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," in *Signal processing: Algorithms, architectures, arrangements, and applications (SPA)*, pp. 70–72, IEEE, 2013.