Wavelet Transform Assisted Neural Networks for Human Activity Recognition

Roshwin Sengupta and Ilia Polian
Institute of Computer Architecture and Computer Engineering
University of Stuttgart
Stuttgart, Germany
{roshwin.sengupta, ilia.polian}@iti.uni-stuttgart.de

John P. Hayes
Computer Engineering Laboratory
University of Michigan
Ann Arbor, Michigan, USA
jhayes@umich.edu

***Abstract*—Human activity recognition (HAR) is a challenging area of research with many applications in human-computer interaction. With advances in artificial neural networks (ANNs), methods of HAR feature extraction from wearable sensor data have greatly improved and have increased interest in their classification using ANNs. Most prior work has only investigated the software implementations of ANN-based HAR. Here, we investigate, for the first time, two novel hardware implementations for use in resource-constrained edge devices. Through architecture exploration, we identify first a hybrid ANN we call DCLSTM incorporating the convolutional and long-short-term memory techniques. The second is a much more compact implementation WCLSTM that uses wavelet transforms (WTs) to enhance feature extraction; it can achieve even better accuracy while being smaller and simpler; it is therefore the better choice for resource-constrained applications. We present hardware implementations of these ANNs and evaluate their performance and resource utilization on the UCI HAR and WISDM datasets. Synthesis results on an FPGA platform show the superiority of the WT-assisted version in accuracy and size. Moreover, our networks achieve a better accuracy than earlier published works.**

Keywords— Artificial neural networks, wavelet transform, human activity recognition, hardware implementation.

I. Introduction

Human activity recognition (HAR) has been receiving significant amount of attention in the recent years due to advances in sensor technology and artificial intelligence [1]. HAR has widespread applications in the domains of health care [2], home behavior analysis [3], gesture recognition [4] and surveillance systems [5]. HAR systems can be vision- [6] or sensor-based [7], using wearable sensors [8] or smartphones [9]. They collect data from, e.g., accelerometers, gyroscopes and magnetometers and analyze them to detect and recognize simple or complex activities. Various machine learning algorithms help overcome difficulties due to ambiguities in sensor data and the dynamic nature of human activities [10]. Recent advancements in artificial neural networks (ANNs) like convolutional neural networks (CNNs), and recurrent neural networks (RNNs) like long short term memory (LSTM) have paved the path for their use in HAR applications [11]-[15].

The traditional software implementation of ANNs like LSTMs and CNNs on CPUs or GPUs has some drawbacks that support the use of dedicated hardware realizations especially in resource-constrained edge devices. At the same time, resource restrictions are essential in applications such as the Internet of Things or autonomous driving, where the impact of ANNs is constantly growing. Due to the recurrent nature of the LSTMs, it is difficult to parallelize their operations on either CPUs or GPUs. CPUs and especially GPUs incur very high power consumption when processing computationally expensive CNNs. This paper focuses on designing dedicated ANN hardware optimized for HAR. For evaluation, we use field programmable gate arrays (FPGAs), which provide the necessary parallelism, energy efficiency, fixed-point arithmetic and flexibility through reconfigurability.

To address the challenging nature of HAR tasks, we employ hybrid networks that combine LSTMs with CNNs. The inherently recurrent nature of LSTMs makes them suitable for capturing temporal information from the time series nature of HAR sensor signals, whereas the spatially deep CNNs are better suited for learning complex human activities [13]. In order to build a suitable network for HAR, we designed a hybrid ANN that we call DCLSTM via architecture exploration that tries to capture the advantages of both CNNs and LSTMs.

Even though DCLSTM outperforms other published networks, it is notably large and hence is less suitable for implementation on smaller hardware devices. We therefore propose WCLSTM, a much smaller version of DCLSTM that is assisted by wavelet transforms (WTs) and is better suited to HAR applications. The accuracy of the ANN improves strikingly due to the addition of WT, as WT filters out noise and enables better feature extraction by providing information about the input data both in the time and frequency domains. We determine the performance of our networks by applying them on two widely used datasets, UCI HAR [16] and WISDM [17], and compare their accuracy with previously published work. Our WCLSTM network outperforms all other networks. We also examine the hardware resource utilization of our networks and demonstrate that FPGA implementations achieve significant speedup when compared to CPU or GPU implementations.

The remainder of the paper is as follows: Section 0 provides some necessary background for the basic building blocks of our hybrid networks, LSTM, CNN, and the two datasets used for HAR applications. In Section III we introduce the proposed architectures. Section IV presents cost and performance results, and Section V concludes the paper.

II. Background

A. Long Short Term Memory (LSTM) Networks

An LSTM network is a type of RNN that can predict output sequences from input sequences [18]. It identifies and remembers hidden states that cannot usually be observed directly, enabling the network to learn useful long-term dependencies.

This work was supported in part by Deutsche Forschungsgemeinschaft (DFG), project number PO 1220/12-1. John P. Hayes was supported by the U.S. National Science Foundation under Grant CCF-2006704.

An LSTM network's memory is embodied in LSTM cells. These are modules with a sequential behavior that depends on three parameters: the input x_t, the cell's previous state c_{t-1}, and a hidden state h_{t-1}, which is the cell's output in the previous time step. The states of all LSTM cells serve as the network's memory.

An LSTM cell includes a forget gate that determines what past information is relevant enough to keep, an input gate that decides what new information is relevant to add, and an output gate that calculates the current hidden state, as depicted in Fig. 1. The sigmoid activation function is used to transform the values between 0 ("not important") and 1 ("important"). This determines how much impact the old and/or the new information has on the cell output. The tanh activation function generates values between –1 and 1, and is used to normalize the data values. The mathematical operations of LSTM are shown in (1)-(6).

$$f_t = \sigma\,(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma\,(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$C'_t = \tanh\,(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

$$c_t = f_t \cdot c_{t-1} + i_t * C'_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \cdot \tanh(c_t) \quad (6)$$

B. Convolutional Neural Networks (CNNs)

CNNs are primarily used for applications like video recognition [19], image classification [10] and activity recognition [20]. As Fig. 2 shows, the first layer of a CNN is one of the possibly several convolution layers. This layer extracts features from the input. It reads an input (image or other signal) in small segments at a time and steps across the entire input, applying convolutions on it and on the filter or kernel. The output of the convolution is called a feature map. Across the entire previous layer, a filter is imposed, which is moved one pixel at a time; its output is collected in the feature map. The distance by which the filter moves across the input is called the stride.

A pooling layer, which follows the convolutional layer of a CNN, reduces the number of parameters of the feature map. It helps to retain important information while down sampling the feature map. For example, a layer with pooling size 2x2 pixels with stride of 2 applied to a feature map of size 8x8 will result in a feature map of size 4x4. Most popular pooling layers use max pooling and average pooling, where the largest or the average value of a 2x2 section of the feature map is selected, respectively. The last layer of CNN is the fully connected layer. It has a non-linear activation function or a softmax function that is responsible for the final classification of the input signal.

C. Datasets

We use two benchmark datasets UCI HAR [16] and WISDM [17] to measure the performance of our networks and compare their accuracy with previously published work. To create the UCI-HAR dataset, 30 subjects of 19-48 years age wore a Samsung Galaxy SII smartphone with embedded accelerometer and gyroscope sensors around the waist and performed six activities: standing, sitting, laying, walking, walking downstairs, and upstairs. A total of 748,406 triaxial acceleration and angular velocity data samples were collected at a 50 Hz sampling rate. The WISDM dataset was built with 36 subjects and has 1,098,209 samples. The subjects performed six activities of daily life: standing, sitting, walking, upstairs, downstairs, and jogging with an Android phone in their front leg pocket. The built-in accelerometer sensor of the smartphone was used to collect the accelerometer data at 20 Hz frequency.

III. Architecture

In this section, we present the architectures of our two ANNs for the HAR applications.

A. DCLSTM

Our initial design DCLSTM embodies the advantages of both CNN and LSTM. We performed an architecture exploration (added and removed layers) until reaching around 98% accuracy so that the final network can be reliably used for detecting complex real-life human activities. We systematically tuned the hyper-parameters (number of layers, size of filter, kernel, and pooling) of the network, eventually reaching the design in Fig. 3 (a) that satisfies our criteria. The input sensor signal is given to the first 1D convolution layer, which is particularly effective in extracting identifying features from a fixed window length dataset like ours, where the window length is 128. This layer has a kernel of size 11 and a filter of size 64. It is followed by another convolutional layer with kernel size 11 and filter size 128. Both layers employ the ReLU activation function. Next, we added a max-pooling layer with pool size 2. A third convolutional layer with kernel size 11 and filter size 128 performs a deeper feature extraction, followed by another max-pooling layer with the same pool size. A flatten layer formats the matrix representation of feature maps to a vector, which is fed to the following two LSTM layers with 128 and

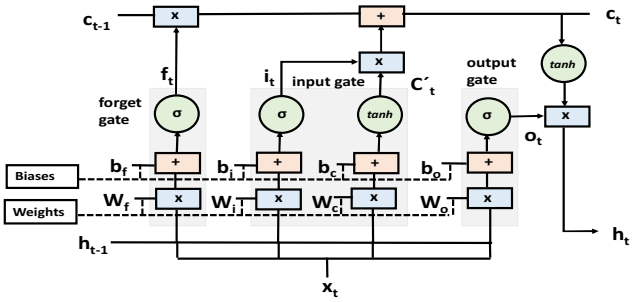

Figure 1: LSTM cell with input x_t, cell state c_{t-1}/c_t, hidden state h_{t-1}/h_t, internal signals f_t, i_t, C'_t, o_t, weights W_f, W_i, W_c, W_o, biases b_f, b_i, b_c, b_o.

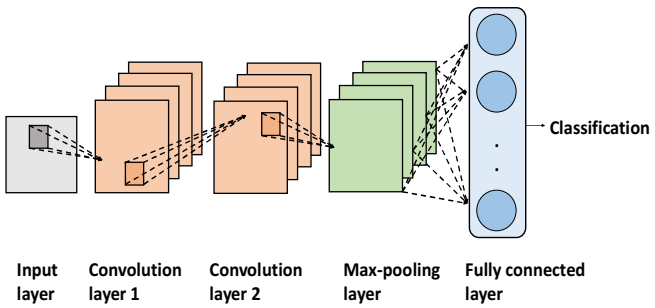

Figure 2: CNN with two convolution and one max-pooling layers.

64 hidden neurons, respectively. The fully connected output layer with softmax activation does the final classification, as shown in Fig. 3 (a). The network is trained in software using Keras and the inference stage is done using Verilog.

The three convolutional layers and the fully connected layer perform DCLSTM's overall convolution operation. We designed a convolutional kernel of size 3x3 that performs 9 multiplications in each clock cycle. This module also performs 8 addition operations to generate the final output. Reusing the same module for multiple layers would decrease the resource utilization. In our design, we also try to speed-up the convolution operation, using shift registers. The convolution operation in each step takes nine neighboring cell values from the input matrix. However, six out of these nine values were already processed in the previous step. Therefore, each step (except the first) requires only three new values. We use shift registers to shift the cell value by one in each row/column of the input matrix. This speeds up the convolution operation and increases the overall computing performance of DCLSTM.

For the LSTM layers in Fig. 3 (a), the major operations to be implemented in hardware are the time-consuming matrix vector multiplications and the non-linear activation functions (tanh and sigmoid). The matrix-vector multiplication is computed by a multiply and accumulate (MAC) unit, which takes the network's input vector and the weight matrix as input. The same input vector is multiplied with all the rows of the weight matrix and the intermediate multiplication results are accumulated in a buffer. This matrix vector multiplication is accelerated by using pipelining. Instead of waiting for all the multiplications of the first input vector with all the weight matrix rows to be done, whenever the first input vector has been multiplied with the first weight matrix row, we start multiplying the second input vector with the first weight matrix row. Data dependencies are avoided by storing all intermediate results in buffers.

For implementing the activation functions, we use piecewise linear approximation of (7). It was found to perform much better than other approximation techniques, since small errors do not affect the overall performance of the LSTM. The values of a and b are stored in block RAMs (BRAMs) and a pipelined MAC unit is used to multiply a and x and accumulate the product with b.

$$f(x) = a_i x + b_i \,,\; x \in [x_i \,,\; x_{i+1}) \tag{7}$$

We use 8-bit fixed-point numbers for all the calculations. As will be discussed in Sec. IV and shown in Table I, this network outperforms other networks, yet it is quite large and resource consuming.

B. *WCLSTM*

The accuracy of ANNs can be increased by improving the feature extraction process using various signal-processing techniques like the Fourier transform (FT) and wavelet transform (WT). We leverage this finding by adding a WT block to the network, with the presumption that signals pre-processed by WTs will require less effort—and thus fewer network layers—for accurate classification. We chose WT (rather than FT) because the periodicity of the wavelets makes the WT localized in both time and frequency domains reflecting the properties of non-stationary signals from HAR applications.

The same architecture exploration with the same accuracy target as in the previous section was applied to a much smaller network with a WT block placed directly after the input layer. This process resulted in the WCLSTM network shown in Fig. 3 (b). Detailed analysis of the same small hybrid network with WT (WCLSTM) and without WT (CLSTM) will be presented in Sec. IV. In the following, we provide information on the specific wavelets used.

WTs involve two major processes, scaling and shifting. Scaling represents a signal's extension or compression in the time domain that helps to capture slow and abrupt changes in the signal by using large and small wavelet versions of the scale parameter α that stretch and shrink the wavelet, respectively. Shifting is done to arrange the features of the signal by moving the differently scaled wavelets from start to end. We use the continuous wavelet transform (CWT) which helps to extract and analyze complex spectral features of signals [21]. CWT extracts features from non-stationary 1D signals and represents them in a 2D scalogram. This is useful in detecting high power regions in the signal at a localized time and frequency and this makes the CWT best suited for our HAR applications. Apart from improving feature extraction, the CWT can also convert one-dimensional (1D) sensor signals into 2D images, which allows us to use 2D CNNs. This further increases the performance of our network significantly.

After careful study of wavelet families, we chose to use the Morlet wavelet for the analysis of the non-stationary time-series input signal [22]. The output of the CWT is fed into the first two 2D convolution layers with kernel size 11, filter size 64 and ReLU activation. To reduce data over fit and network complexity, a max-pooling layer with pool size 2 is used next. We add a flatten layer to format the output data from CNN so that it can be fed to a single LSTM layer with 128 units and tanh activation. The result from the LSTM layer is passed to a fully connected layer with a softmax activation function, which classifies the input into six different classes for both the UCI HAR and the WISDM datasets. The network was trained using Keras and synthesized on hardware using Verilog.

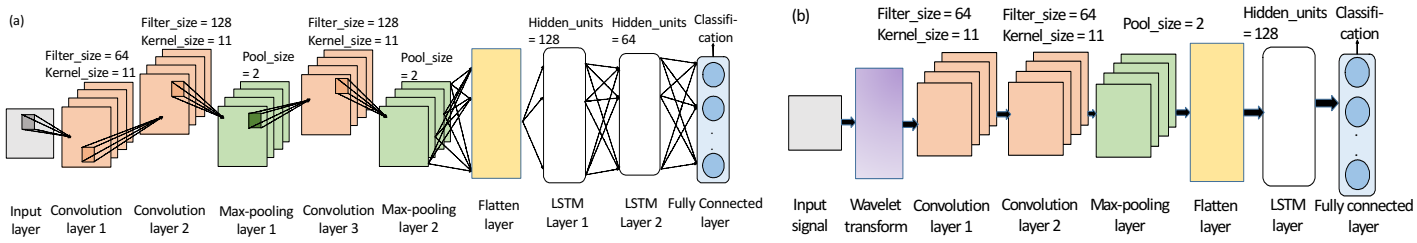

Figure 3: The DCLSTM network proposed for HAR applications (a), the WCLSTM network proposed for HAR applications (b).

To optimize the hardware realization of WT, we use the Fast Fourier Transform (FFT), which reduces complex convolution operation in the time domain to simple multiplication in the frequency domain, and vice versa. In our hardware realization, the input signal is first transformed to the frequency domain using the FFT and the result is stored in a buffer. Then the Morlet wavelet is converted to the frequency domain at different scales α (in our case 64). The resultant signal FFT and the wavelet FFT are then multiplied at all α's. The inverse FFT at each of the 64 scales produces the required CWT coefficients for that particular scale, and is repeated until the final scale is evaluated. Once the whole wavelet transform operation is complete, the output is then fed to the CNN as shown in Fig. 4. The hardware implementation of the remaining layers is the same as that of DCLSTM and only the parameters are modified to the best values found during architecture exploration.

IV. Results

We evaluate the proposed networks with respect to classification accuracy on both the UCI HAR and WISDM datasets (Sec. II.C), hardware utilization, and performance in comparison to a software implementation. We refer to the network from Sec. III.A (architecture optimized without WT) as DCLSTM and to the network from Sec. III.B (WT-assisted architecture) as WCLSTM. For comparison, we also present results for the latter network with the WT block removed; we call this design CLSTM.

Table I compares the accuracy of these and previously published hybrid networks. Both proposed networks outperform all other networks on both data sets, and the WT-assisted one has a higher accuracy. In contrast, the WT-assisted network stripped of its WT block classifies rather poorly, underscoring the importance of the WT functionality.

In Table II, we present the resource utilization of the proposed networks when synthesized on a Virtex-7 FPGA board at 150 MHz using the Vivado compiler. Note that the previously proposed networks have no published FPGA implementations so we cannot compare with them in this regard. Both our designs fit on the FPGA, as the maximal resource utilization is around 40%. At the same time, DCLSTM is roughly 60% larger than WCLSTM, which can be excessive for resource-constrained applications. Thus, the WT-assisted version is both smaller and has better accuracy. The fact that DCLSTM is able to achieve a comparable (only slightly worse) accuracy suggests that the signal processing done by WT is emulated by the extra layers that are not present in WCLSTM, but at an excessive cost. Stripping the WT block from CLSTM reduces the resource consumption as expected by roughly 15%; however this reduction is likely not worth the significant deterioration in accuracy.

Finally, we demonstrate the superiority of a dedicated hardware architecture over software running on a CPU or a GPU. A comparison of software implementations running in Intel Core i5 CPU 3.20 GHz and on NVIDIA GTX GPU 980 MHz with 2048 CUDA cores is found in Table III where the runtime for classification of one test dataset is given. Despite a lower clock frequency of 150 MHz, our networks classify roughly 15× faster than the CPU, and about 8× faster than the GPU. Note that the WT-assisted network is roughly 2× faster than the larger DCLSTM.

V. Conclusion

We investigated hybrid neural network architectures for human activity recognition that provides high classification accuracy but at the same time are suitable for realization in dedicated hardware. A feature of our work is the integration of a wavelet transform block into the architecture and a study of its impact. For the first time, we provided and evaluated hardware implementations of hybrid LSTM-CNNs on an FPGA platform, demonstrating an order-of-magnitude speed-up compared to all-software solutions. We found that achieving best-in-class accuracy of around 98% is possible with and without WT assistance, yet the WT-assisted network is much more compact and faster, as fewer layers are required to achieve the accuracy target. These findings strongly suggest that WT should be part of future solutions for hardware acceleration in the HAR domain.

Table I: Comparison of accuracy for different network architectures.

Network architecture	Number of layers	Accuracy (%)	
		UCI HAR	WISDM
CNN–LSTM [13]	1 CNN layer, 1 LSTM layer, 128 units	92.0	-
LSTM–CNN [14]	2 CNN layers, 2 LSTM layers, 32 units	95.7	95.0
CNN–LSTM [15]	3 CNN layers, 1 LSTM layer, 30 units	97.0	94.0
CNN–RF [15]	3 CNN layers, Random Forest	98.0	97.0
DCLSTM	3 CNN layers, 2 LSTM layers, 128 & 64 units	98.4	97.8
WCLSTM	WT, 2 CNN layers, 1 LSTM layer, 128 units	98.9	98.5
CLSTM	2 CNN layers, 1 LSTM layer, 128 units	95.8	95.3

Table II: Resource utilization of proposed networks on Virtex-7 FPGA.

Network architecture	FF (%)	LUT (%)	BRAM (%)	DSP (%)
DCLSTM	148680 (20.4%)	147500 (40.5%)	251 (31.5%)	510 (40.4%)
WCLSTM	92650 (12.7%)	84600 (23.2%)	180 (22.6%)	340 (26.9%)
CLSTM	77330 (10.6%)	75430 (20.7%)	150 (18.8%)	290 (23.0%)

Table III: Runtimes on CPU, GPU and FPGA implementations.

Network architecture	CPU	GPU	FPGA
DCLSTM	1.611ms	0.819ms	0.105ms
WCLSTM	0.876ms	0.436ms	0.049ms
CLSTM	0.813ms	0.402ms	0.036ms

REFERENCES

[1] C. V. San Buenaventura and N. M. C. Tiglao, “Basic human activity recognition based on sensor fusion in smartphones,” IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1182–1185.

[2] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu and P. Havinga, “Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey.”, In Proceedings of the 23rd International Conference on Architecture of Computing Systems (ARCS), 2010, pp. 1–10.

[3] P. Vepakomma, D. De, S. K. Das and S. Bhansali, "A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities," IEEE International Conference on Wearable and Implantable Body Sensor Networks (BSN), 2015, pp. 1-6.

[4] Y. Kim and B. Toomajian, "Hand gesture recognition using micro-doppler signatures with convolutional neural network," IEEE Access, vol. 4, 2016, pp. 7125-7130.

[5] M. Babiker, O. O. Khalifa, K. K. Htike, A. Hassan and M. Zaharadeen, “Automated daily human activity recognition for video surveillance using neural network,” IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA), IEEE, 2017, pp. 1–5.

[6] A. Jalal, Y.-H. Kim, Y.-J. Kim, S. Kamal and D. Kim, “Robust human activity recognition from depth video using spatiotemporal multi-fused features,” Pattern recognition, vol. 61, 2017, pp. 295–308.

[7] Y. Chen and Y. Xue, “A deep learning approach to human activity recognition based on single accelerometer,” IEEE International Conference on Systems. IEEE, 2015.

[8] A. Jordao, AC. Nazare Jr., J. Sena and WR. Schwartz, “Human activity recognition based on wearable sensor data: A standardization of the state-of-the-art,” ArXiv preprint arXiv:1806.05226, 2018.

[9] A. Mandong and U. Munir, “Smartphone based activity recognition using k-nearest neighbor algorithm,” In Proceedings of the International Conference on Engineering Technologies, Konya, 2018, pp 26–28

[10] M. I. Razzak, S. Naz and A. Zaib, “Deep learning for medical image processing: Overview, challenges and the future in classification,” BioApps. Springer, 2018, pp. 323–350.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., “Going deeper with convolutions,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[12] S. Ramasamy Ramamurthy and N. Roy, “Recent trends in machine learning for human activity recognitiona survey,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, 2018.

[13] R. Mutegeki and D. S. Han, "A CNN-LSTM approach to human activity recognition," International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020.

[14] K. Xia, J. Huang and H. Wang, "LSTM-CNN architecture for human activity recognition," IEEE Access, vol. 8, 2020, pp. 56855-56866.

[15] V. Ghate and S.C, “Hybrid deep learning approaches for smartphone sensor-based human activity recognition,” Multimedia Tools and Applications. 1-20. 10.1007/s11042-020-10478-4, 2021.

[16] J. L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra and D. Anguita, "Transition-aware human activity recognition using smartphones," Neurocomputing, vol. 171, 2016, pp. 754-767.

[17] J. R. Kwapisz. G. M. Weiss and S. A. Moore, "Activity recognition using cell phone accelerometers," ACM SigKDD Explorations Newsletter, vol. 12, no. 2, 2011, pp. 74-82.

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, 1997.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "Large-Scale video classification with convolutional neural networks," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1725-1732.

[20] O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. R. Meena, D. Tiede, and J. Aryal, “Evaluation of different machine learning methods and deep-learning convolutional neural networks for landslide detection,” Remote Sensing, vol. 11, no. 2, 2019, pp. 196.

[21] J. Sadowsky, “Investigation of signal characteristics using the continuous wavelet transform,” Johns Hopkins Apl. Technical Digest, 1996 - jhuapl.edu Neural Computation vol. 29, Issue 9, 2017 pp.2352-2449.

[22] M. X Cohen, “A better way to define and describe Morlet wavelets for time-frequency analysis”, NeuroImage, vol. 199, 2019, pp. 81-86, ISSN 1053-8119.