Robustness of deep LSTM networks in freehand gesture recognition

Monika Schak and Alexander Gepperth

University of Applied Sciences Fulda, 36037 Fulda, Germany

Abstract. We present an analysis of the robustness of deep LSTM networks for freehand gesture recognition against temporal shifts of the performed gesture w.r.t. the "temporal receptive field". Such shifts inevitably occur when not only the gesture type but also its onset needs to be determined from sensor data, and it is imperative that recognizers be as invariant as possible to this effect which we term *gesture onset variability*. Based on a real-world hand gesture classification task, we find that LSTM networks are very sensitive to this type of variability, which we confirm by creating a synthetic sequence classification task of similar dimensionality. Lastly, we show that including gesture onset variability in the training data by a simple data augmentation strategy leads to a high robustness against all tested effects, so we conclude that LSTM networks can be considered good candidates for real-time and real-world gesture recognition.

Keywords: LSTM · hand gestures · 3D sensors

1 Introduction

This article is in the context of freehand gesture recognition using 3D sensors and Long Short-Term Memory (LSTM) networks. We present a case study on the robustness properties of such a system, notably w.r.t. delays and variations in gesture onset. Such effects play a strong role in real-world applications, such as systems that allow the user to interact using dynamic freehand gestures, for example within the scope of 3D object manipulation or augmented reality. In this context, the time at which gestures are initiated is not known and therefore the system needs to be able to not only recognize the gesture itself but also its onset. This is, in some ways, analogous to visual object detection problems where the identity as well as the location of an object needs to be estimated.
In that regard, it is very important to investigate to what extent popular sequence recognizers like LSTM networks are capable of handling onset variations of the gestures (sequences) they are trained to recognize. This article investigates, first of all, the "native" robustness of LSTM networks to such variations and then goes on to specifically train LSTM networks to be robust to such variations by including them into the training data. As in visual object recognition data augmentation is used by simply generating new training data that contain the undesirable variations from existing samples, and training new classifiers

from these transformed data. We investigate several possible types of onset variations, and show what effect training LSTM classifiers with those variations included has on classification accuracy.

2 Our Approach

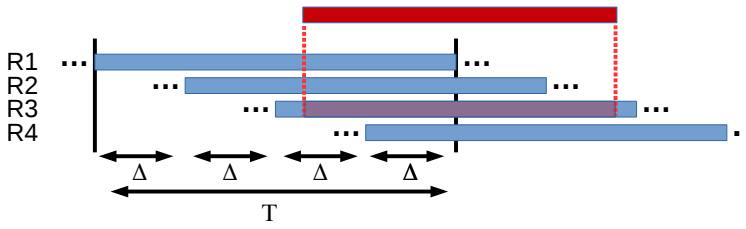

Fig. 1. Overview of the "shifted recognizer" architecture used to recognize freehand gestures. Exemplarily, we show an architecture using 4 recognizers that are shifted w.r.t. each other by a time interval of $\Delta = T/4$ frames. The temporal receptive fields (TRFs) of each recognizer are indicated by blue bars, and each recognizer is run in a continuous loop. A freehand gesture, indicated by a red bar, will usually fall into the TRF of exactly one recognizer which will then hopefully report this, thus giving both gesture type and gesture onset, the accuracy of the latter being principally determined by Δ.

For addressing the issue of recognizing gestures that can start at any given moment in time, we propose a solution analogous to object detection, where N identical gesture recognizers are run in a continuous loop in parallel. Recognizers are trained to recognize gestures of fixed length T which determines their *temporal receptive field* (TRF). Each recognizer is reset after a full TRF and each receives the same frames from a 3D sensor. However, each recognizer is delayed by $\Delta = \frac{T}{N}$ frames w.r.t. the other recognizers so if enough recognizers are run in parallel it can be hoped that a gesture of length $\tau < T$ will fall fully into the TRF of a single recognizer which will then report it. This approach, which is visualized in Fig. 1 (where the gesture falls into the TRF of recognizer R3), suffers from some obvious problems:

- In general, gesture onset will not coincide with the first frame of the recognizer's TRF.
- If gestures are performed without breaks between them, there will be appended or prepended frames to each elementary gesture, coming from other gestures in a recognizers TRF.

This raises the question of how well a LSTM network is able to deal with different onsets consisting of either random frames or frames containing parts of other gestures. Here is where our work comes in.

3 Related work: RNNs and dynamic hand gesture recognition

RNNs are primarily used because of their ability to remember the past activities or inputs and then produce an output that takes past inputs and states into consideration. Due to this specific property of RNNs they are used in fields which involve temporal dependencies over several time steps, such as gesture recognition, activity recognition or speech recognition. LSTM networks were introduced by [7] to specifically address the vanishing gradient problem that was complicating efficient RNN training. LSTM networks have been used for language modelling [12], sequence generation [4], speech recognition [6] and image captioning [8].

There are various approaches to freehand gesture recognition with LSTM networks (of course, this kind of recognition task can also be performed without LSTMs, e.g., [11, 21, 3, 1, 9]). Especially in visually defined problems where camera images are analyzed, several authors employ LSTM networks as the top layer of a CNN, which performs the basic image transformations that are required for sufficient invariance and robustness [18, 13, 14]. The precise way of setting up the CNN and coupling CNN and LSTM can of course vary considerably. In case of multiple input streams (e.g., camera and 3D sensors), the underlying CNN architecture can become considerably more complex and offers more design choices, yet many authors still let LSTM recognizers operate on the top layer activities of a (possibly hybrid) CNN architecture, e.g., in [19]. Explicit extensions to the basic deep LSTM model are proposed in [20] in the form of a convolutional LSTM architecture.

4 Methods and data

We use two datasets in this study: a real-world freehand gesture dataset that we recorded ourselves, and a synthetic one constructed from the well-known visual MNIST benchmark [10]. Our own freehand gesture dataset is introduced because we find that using a single ToF-sensor to capture hand gestures for interacting with a device (for example mobile devices or vehicles) allows for more freedom and increased expressiveness [17]. The synthetic dataset is introduced because the real-world gesture dataset is relatively small, and we wish to exclude that results are due to overfitting. Tab. 1 gives an overview over the properties of each dataset.

Dataset	training samples	test samples	frames	dimensions	classes	class distrib.
MNIST sequences	20.000	5.000	40	784	4	balanced
3D hand gestures	320	80	40	625	4	balanced

Table 1. Properties of the two datasets used in this study. The real-world hand gesture dataset is relatively small, which is why we decided to complement it by a large-scale sequence classification dataset constructed from the MNIST classification benchmark. Sequences in both datasets have a length of 40 frames, and the frame dimensionalities in both datasets are comparable (625 -vs- 784). Both datasets consist of equally distributed classes.

4.1 Freehand gesture dataset

Data acquisition Data is collected from a DepthSense TOF sensor at a resolution of 320×160 pixels. Depth thresholding removes most of the irrelevant background information, leaving only hand and arm voxels. Principal-Component Analysis (PCA) is utilized to crop most of the negligible arm parts. The remaining part of the point cloud carries the relevant information, i.e. the shape of the hand. Fig. 2 shows the color-coded snapshot of a hand posture. We record four different hand gesture types from ten different people for our database: close hand, open hand, pinch-in and pinch-out. The latter gestures are performed by closing/opening two fingers. For recording a single dynamic gesture, 40 consecutive snapshots are taken from the sensor and cropped by the aforementioned procedure. We recorded ten gesture samples per person and gesture type, giving a total of 400 gestures or 100 per gesture type (class). As 40 frames per gesture are present, this sums up to a total of 16,000 sensor frames (point clouds).

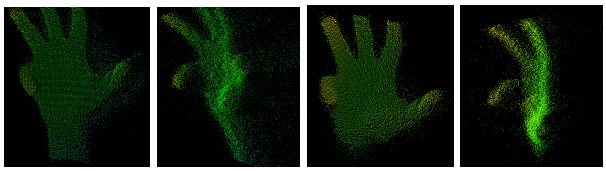

Fig. 2. Individual point clouds (frames) from the freehand gesture dataset after cropping from the front (left) and side view (right) during a grabbing motion. Point clouds such as these are further processed to form a 625-dimensional feature vector before being presented to the LSTM-based gesture recognizer.

Data pre-processing From each frame as represented by a point cloud, we extract a descriptor of fixed size (regardless of the size of the point cloud) that can be fed to the deep LSTM recognizers [16, 17]. Such descriptors need to describe the phenomenology of hand, palm and fingers in a precise manner, while remaining computationally feasible. In this contribution, we decided on a rep-

resentation that is largely based on surface normals. First, these normals are computed for all points. Then, we repeatedly (5000 times) select two random points and compute their point feature histogram (PFH) descriptor [15, 2] which yields four numbers based on the distance and the relative orientation of the surface normals. Each of the four numbers is coarsely quantized into five bins, giving a total of 625 discrete possibilities, and a histogram over all 5000 realizations is taken. These 625-dimensional histograms, each describing a single point cloud, form the individual frames of the sequences we use for training and testing deep LSTM recognizers.

Dataset creation The set of 400 recorded gesture samples can be described by a three-dimensional tensor G, indexed by sequence index $0 \leq i < 400$, frame index $0 \leq j < 40$ and histogram index $0 \leq h < 625$. A single histogram entry accesses as G_{ijh}. The available gesture samples are split into training and test samples for the LSTM recognizers, the train to test ratio being 80/20.

4.2 Synthetic dataset

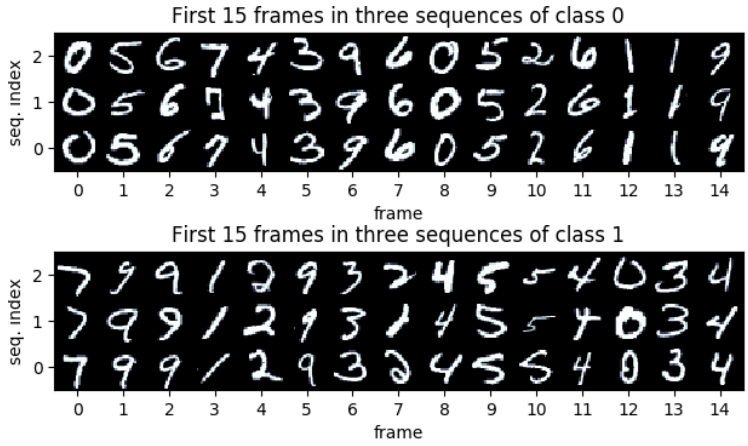

Fig. 3. Six sample sequences from the synthetic MNIST-based dataset. Three sequences (above) come from class 0 and three sequences (below) come from class 1. Classes 3 and 4 are not shown. Please note that sequences from the synthetic dataset have a length of 40 frames, but only the first 15 frames are shown due to space constraints.

The synthetic dataset is created to be largely analogous in structure to the freehand gesture dataset, having the same number of frames per sequence and a similar dimensionality per frame. However, due to the synthetic nature of this dataset, we can generate a larger amount of samples with very little effort. The construction is straightforward: we generate four 40-element sequence templates (as shown in Fig. 3). The sequences equate to concatenated frames from a video.

Each sequence template is described by the vectors $\boldsymbol{s}_0$ to $\boldsymbol{s}_3$, where each position j in all vectors contains integers that indicate a randomly drawn MNIST class: $s_{ij} \in [0, 9]$. For each sequence template $\boldsymbol{s}_k$ we generate $25,000$ sequences of $0 \leq j < 40$ frames each, each frame j being represented by a (flattened) randomly drawn MNIST sample of class s_{kj}. Thus, the synthetic data can be represented by a three-dimensional tensor accessed by sequence index $0 \leq i < 25,000$, frame index $0 \leq j < 40$ and image index $0 \leq h < 28 \times 28$. Again, a train/test split of 80/20 is performed subsequently.

4.3 Deep LSTM for recognition

In our model for dealing with the video frames sequentially, we use a deep RNN with LSTM model neurons. The LSTM term for neuron is "memory cell" and the term for hidden layer is "memory block". A single memory cell at time t has a dynamic state or activity c_t, whereas the vector of all activities in a memory block is denoted $\boldsymbol{c}_t$.
Any change to the cell state is done with the help of gates: input gate i_t and forget gate f_t, each gate having a value between 0 and 1. The input gate determines how much of the input is forwarded, and the forget gate determines how much of the cell's previous state to retain. Output to subsequent layers, denoted $\boldsymbol{h}_t$, is generated via a simple transfer function (termed output gate).
The LSTM model equations for computing activations $\boldsymbol{h}_t$ of a single LSTM layer read as follows:

$$\boldsymbol{i}_t = \sigma\left(W_{xi}\boldsymbol{x}_t + W_{hi}\boldsymbol{h}_{t-1} + W_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) \tag{1}$$

$$\boldsymbol{f}_t = \sigma\left(W_{xf}\boldsymbol{x}_t + W_{hf}\boldsymbol{h}_{t-1} + W_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) \tag{2}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t \tanh\left(W_{xc}\boldsymbol{x}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right)) \tag{3}$$

$$\boldsymbol{o}_t = \sigma\left(W_{xo}\boldsymbol{x}_t + W_{ho}\boldsymbol{h}_{t-1} + W_{co}\boldsymbol{c}_t + \boldsymbol{b}_o\right) \tag{4}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \tanh(\boldsymbol{c}_t) \tag{5}$$

Equation 1 describes the calculation of the input gate. The output of the input gate is a value between 0 and 1. Equation 2 refers to the calculation of the forget gate. The output of the forget gate also is a value between 0 and 1. Equation 3 describes the calculation of the new cell state, replacing the old one. Equation 4 refers to the calculation of the output gate. The output of the output gate is a value between 0 and 1. Equation 5 refers to the calculation of the hidden state or the output of the memory block, which is the input for the next memory block. Due to the tanh function it can output a value between -1 and 1.

We use a standard deep LSTM architecture with linear softmax readout layer and cross-entropy loss function as outlined in [5]. Number and size of memory blocks, which are all set to have the same number of memory cells, are denoted (L, S). The final output of the LSTM network is produced by applying a linear regression readout layer that transforms the states $\boldsymbol{h}_t$ of the last hidden layer into class membership estimates, using the standard softmax non-linearity leading to positive, normalized class membership estimates.

4.4 TensorFlow implementation

We use the TensorFlow (v1.11) implementation of a Recurrent Neural Network with multiple LSTM cells under Python (v3.6) - mainly the classes *MultiRNN-Cell* and *LSTMCell*, which allow to create networks of stacked LSTM cells. We always use the Adam optimizer included in TensorFlow for performing gradient descent as well as a linear softmax readout layer and cross-entropy loss function.

5 Experiments

For our experiments we use a deep LSTM architecture consisting of three hidden layers with ten LSTM cells each $(L, S) = (3, 10)$, a fixed learning rate of $\epsilon = 0.001$ and a training to test data ratio of 80/20. During a test phase we varied the learning rate in the range of $\epsilon \in [0.0001, 0.1]$ and the amount of layers and cells in the range of $L \in [1, 10]$ and $S \in [5, 50]$ and used the learning rate, amount of hidden layers and LSTM cells for our architecture that performed best on the test sets. Our architecture has a TRF of $T = 60$ frames. Since the gestures in our database have a length of 40 frames, they can be offset by 20 frames at most. Parameters like learning rate, number of hidden layers and amount of LSTM cells per hidden layer need to be chosen depending on the test and training data since in general the quality of the results can depend on the parameters. At first we describe two preliminary experiments to test our network and databases, then we continue with four experiments about the robustness of LSTM networks.

5.1 Preliminary experiment: ahead-of-time classification

Recurrent neural networks are able to estimate a class before the end of the sequence they are currently classifying. For the use of LSTM networks in real-world scenarios, this ability can be very helpful because the network does not have to wait until the end of the sequence to classify it. Therefore, we test the classification accuracy for our 40-frame sequences after $0 < n \leq 40$ frames.

As can be seen in Fig. 4, the classification accuracy is only slightly below its maximum value even for small n. This shows that our network is able to obtain a high accuracy very early in the process of classifying a sequence. Hence, an action can be performed according to the "guessed" classification output before the whole gesture is processed. This allows for faster response times and offers a significant advantage in comparison to having to wait for the whole sequence to be processed. As expected, the results for the artificial dataset are as high as all previous work using MNIST. This is a well-known fact, since MNIST is a very easy-to-solve benchmark even when considering only image classification instead of sequences as we do here.

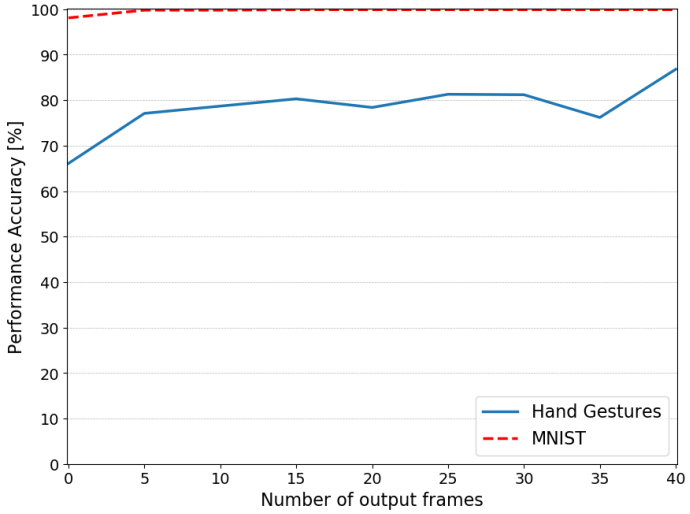

Fig. 4. Classification accuracy depending on the output frame used to generate the gesture classification.

5.2 Preliminary experiment: feasibility of the database

The reason for this experiment is to test if our database of hand gestures is big enough. We use two different strategies to split our data into train and test sets: The first strategy randomly chooses 20% of samples from the whole database as a test set and uses the rest for training. This strategy is only used for our experiment to show that overfitting is a problem here. As required by machine learning theory, data used for training and testing need to be independent and identically distributed.
The second strategy chooses all the gestures from eight people for training and all gestures from the remaining two people for testing, achieving the same 80/20 split as before. As Fig. 5 shows, there is a significant difference in accuracy between the two strategies. Since different people perform the same gesture quite differently, the differences between the training and the test set of the second strategy are too big to achieve better results. A bigger database with gestures from substantially more people would be required to decrease this overfitting behavior and train the network well enough to generalize the classification for gestures from unfamiliar people. Comparing the results from our own database to the artificial image dataset proves this point: The experiments with MNIST data shows overall better results.
In a real-life scenario the system usually would be trained with gestures from different people than the ones using the trained system later on. Therefore, more training data is needed for real-world applications.

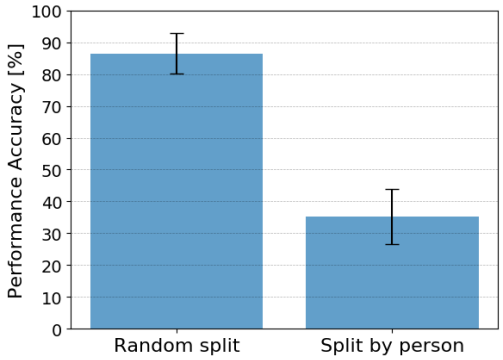

Fig. 5. Classification accuracy depending on the strategy used to split the database into test and training sets.

5.3 Dependency of LSTM networks on previous states and results

To asses whether our network needs to be reset after a TRF traversal, we concatenate two randomly picked gestures and feed them to our trained network. Accuracy is measured for the classification of the second gesture after 80 frames. For this experiment, the TRF of our network is extended to 80 frames instead of the usual 60 frames.

Fig. 6 shows that the network depends on a clean state for well recognizing a gesture, since the memory of the first gesture strongly affects the classification of the second gesture and therefore leads to a very low classification accuracy. Thus, we can conclude that the LSTM network has to be reset at the start of every TRF to ensure a correct classification.

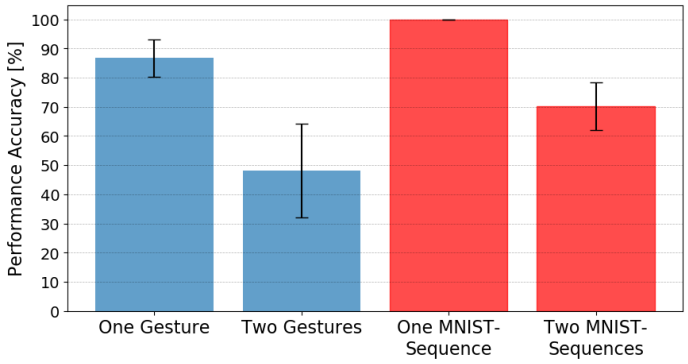

Fig. 6. Classification accuracy whether only one gesture (left) is classified or the second of two concatenated gestures is classified (right).

5.4 Fixed noisy or clean gesture onsets

In this experiment, we investigate the robustness of our LSTM network to gesture shifts (onset variability) of a fixed number of frames. For this, we create new test samples from our dataset that consist of 60 frames and contain the actual 40-frame gesture starting at frame n. Hence, the gesture being fed to the trained network consists of three parts which total 60 frames: $x_1x_2x_3$ with $0 \leq |x_1| \leq 20$, x_2 being the actual gesture with $|x_2| = 40$ and $|x_3| = 60 - |x_1| - |x_2|$. Both parts x_1 and x_3 either are initialized to zero-frames or random frames picked from the database.

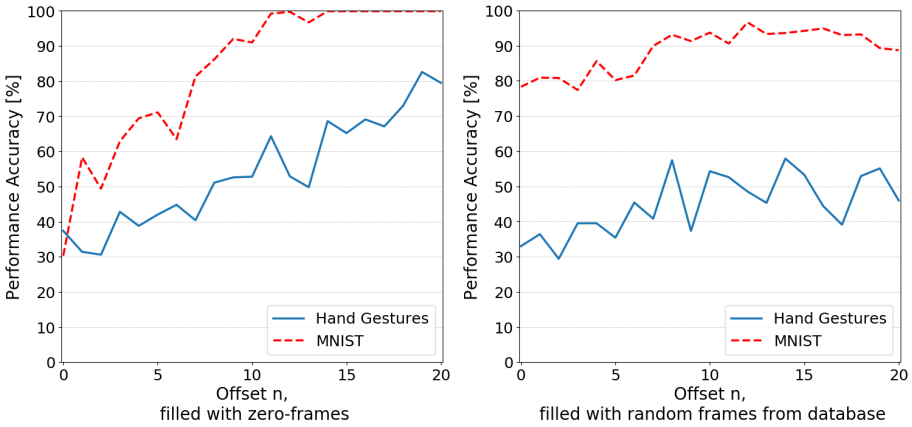

Fig. 7. Classification accuracy for a fixed clean (left) or noisy (right) gesture onset of n frames.

The results in Fig. 7 show that our network can hardly handle any amount of random frames prior or after an actual hand gesture. In contrast to this, it is possible for our network to work decently with fixed-time gesture onsets filled with zero-frames but zero-frames after the actual gesture considerably influence the accuracy.
For the MNIST dataset, the results are similar. Zero-frames after the actual sequence strongly influence the accuracy while zero-frames before the sequence hardly do. The effect of random frames prior or after an MNIST sequence is lower than for hand gestures but still visible.

5.5 Random noisy or clean gesture onsets

Unlike the last experiment where we used a fixed gesture onset, we now embed each test sample into a new 60-frame sample at a random point in time. The samples are initialized either to zero or to random frames from the database. According to the last experiment, each gesture being fed to the trained network

also consists of three parts which total 60 frames: $x_1x_2x_3$ with $0 \leq |x_1| \leq 20$ with a random length $|x_1|$ for each gesture, x_2 being the actual gesture with $|x_2| = 40$ and $|x_3| = 60 - |x_1| - |x_2|$.

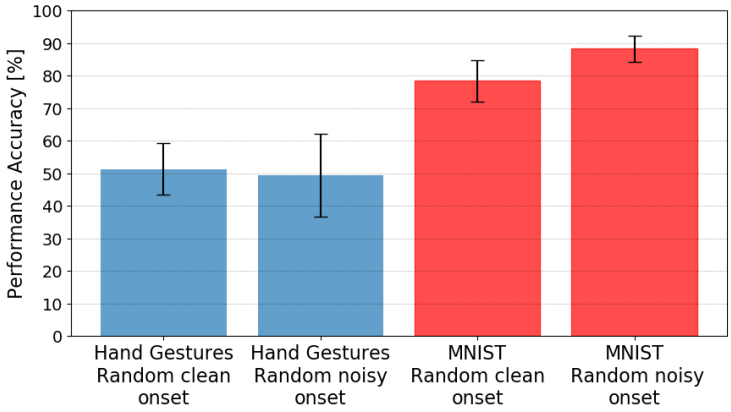

Fig. 8. Classification accuracy for a clean (left) or noisy (right) gesture onset of a random amount of frames.

As can be seen in Fig. 8, choosing a random onset for each gesture leads to a notably decreased accuracy. Since the trained network is the same as for the setting in the last experiment (cf. 5.4), the results for the random gesture onset correspond to the average results from our experiments with a fixed gesture onset.

5.6 Random clean gesture onset included in training and testing

As the last three experiments show, our network has difficulties handling gestures that do not start right at the beginning of the TRF but are padded with additional frames. A possible solution to this is to train the network in such a way that allows it to learn to ignore padding around the actual gestures. To show the effects of this, we train our network with 60 frame gestures, compound as described for the last two experiments: a random amount of zero-frames between 0 and 20 before the actual gesture and then filled up with zero-frames afterwards.

Fig. 9 combines the results from multiple experiments: The accuracy shown on the left is the result of a network trained and tested with just the gestures and without any padding (cf. 5.2). This is the baseline of how good the accuracy can be without any additional frames distracting from the gesture. The accuracy shown in the middle is the result of a network trained with just the gestures and tested on gestures with a clean random onset (cf. 5.5). And the accuracy on

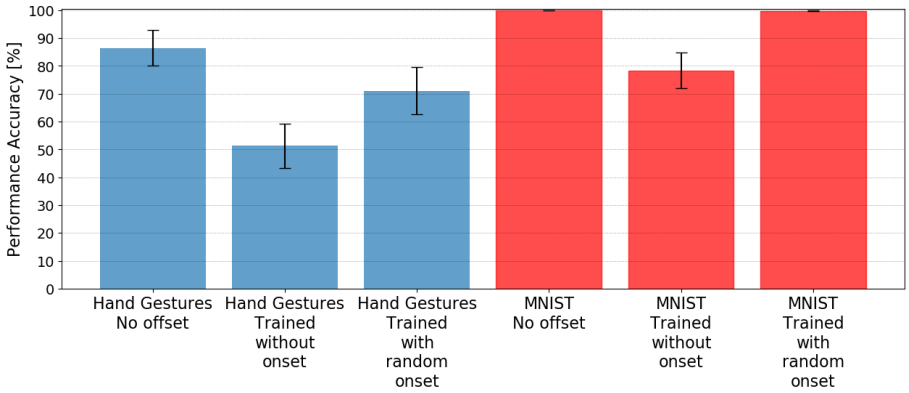

Fig. 9. Classification accuracy for gestures with no padding at all (left) and for gestures with a clean random onset - trained with just the gestures (mid) or with a clean random onset (right).

the right is the result on a network trained and tested on gestures with a clean random onset.

As can be seen, the accuracy can be considerably increased by training the network with padded gestures. Then, our network is able ignore zero-frames before and after the actual gesture to some extent. The effect can be improved by training the network with more variations of the same gesture instead of just one random padding for each gesture as done in our experiment.

6 Discussion and conclusion

This article investigates robustness properties of LSTMs in the context of sequence classification in general and hand gesture recognition in particular. Our main findings are twofold:

LSTMs are vulnerable to sequence onset variability First of all, LSTM networks are very sensitive to variability in sequence onset (of which gesture onset is a special case). This has not been empirically studied before, which is why we consider these insights highly relevant for applied settings. The generality and credibility of our results comes from using two sequence learning tasks that come from very different domains for this investigation.

Robustness can be achieved by data augmentation Secondly, we find that LSTMs can be made very robust against sequence onset variability if this variability is incorporated into the training data by data augmentation strategies. This is particularly straightforward for sequences as the type of variability can be reduced to the maximal delay of sequence onset that a LSTM recognizer should be able to process, which mainly depends on the number of parallel recognizers

we can run, see Fig. 1. In other words, we need to first think about robustness properties and then actively include them in the training data.

Conclusion and future work We find the problem of real-world LSTM sequence recognition to be remarkably analogous to training visual sliding-window object detectors, where similar data augmentation strategies must be employed to make detectors robust against small shifts in object position. Data augmentation works for LSTM networks which have, after all, a dynamic internal structure which is potentially much more complex than that of feed-forward CNN models. Further investigations on real-world gesture and activity recognition can build on this insight in order to implement robust real-world sequence recognition. In addition, a comparison to other sequence classification models should be done to further investigate the applicability of the proposed system.

References

1. Camgoz, N.C., Hadfield, S., Koller, O., Bowden, R.: Using convolutional 3d neural networks for user-independent continuous gesture recognition. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 49–54. IEEE (2016)
2. Caron, L.C., Filliat, D., Gepperth, A.: Neural network fusion of color, depth and location for object instance recognition on a mobile robot. In: European Conference on Computer Vision. pp. 791–805. Springer (2014)
3. Duan, J., Wan, J., Zhou, S., Guo, X., Li, S.Z.: A unified framework for multimodal isolated gesture recognition. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **14**(1s), 21 (2018)
4. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
5. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: International Conference on Machine Learning. pp. 1764–1772 (2014)
6. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. pp. 6645–6649. IEEE (2013)
7. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02), 107–116 (1998)
8. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3128–3137 (2015)
9. Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 156–165 (2017)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Apllied to Document Recognition (1998)
11. Miao, Q., Li, Y., Ouyang, W., Ma, Z., Xu, X., Shi, W., Cao, X.: Multimodal gesture recognition based on the resc3d network. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3047–3055 (2017)
12. Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh annual conference of the international speech communication association (2010)

13. Nguyen, A., Kanoulas, D., Muratore, L., Caldwell, D.G., Tsagarakis, N.G.: Translating videos to commands for robotic manipulation with deep recurrent neural networks. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1–9. IEEE (2018)
14. Ordóñez, F., Roggen, D.: Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors **16**(1), 115 (2016)
15. Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M.: Aligning point cloud views using persistent feature histograms. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3384–3391. IEEE (2008)
16. Sachara, F., Kopinski, T., Gepperth, A., Handmann, U.: Free-hand gesture recognition with 3d-cnns for in-car infotainment control in real-time. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 959–964 (Oct 2017). https://doi.org/10.1109/ITSC.2017.8317684
17. Sarkar, A., Gepperth, A., Handmann, U., Kopinski, T.: Dynamic hand gesture recognition for mobile systems using deep lstm. In: Horain, P., Achard, C., Mallem, M. (eds.) Intelligent Human Computer Interaction. pp. 19–31. Springer International Publishing, Cham (2017)
18. Tsironi, E., Barros, P., Wermter, S.: Gesture recognition with a convolutional long short-term memory recurrent neural network. In: Proceedings of the European symposium on artificial neural networks computational intelligence and machine learning (ESANN). pp. 213–218 (2016)
19. Wu, J., Ishwar, P., Konrad, J.: Two-stream cnns for gesture-based verification and identification: Learning user style. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 42–50 (2016)
20. Zhu, G., Zhang, L., Mei, L., Shao, J., Song, J., Shen, P.: Large-scale isolated gesture recognition using pyramidal 3d convolutional networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 19–24. IEEE (2016)
21. Zhu, G., Zhang, L., Shen, P., Song, J.: Multimodal gesture recognition using 3-d convolution and convolutional lstm. IEEE Access **5**, 4517–4524 (2017)