Effective 2D Stroke-based Gesture Augmentation for RNNs

Mykola Maslych
maslychm@knights.ucf.edu
University of Central Florida
Orlando, Florida, USA

Eugene M. Taranta II
etaranta@gmail.com
University of Central Florida
Orlando, Florida, USA

Mostafa Aldilati
mad51@knights.ucf.edu
University of Central Florida
Orlando, Florida, USA

Joseph J. LaViola Jr.
jjl@cs.ucf.edu
University of Central Florida
Orlando, Florida, USA

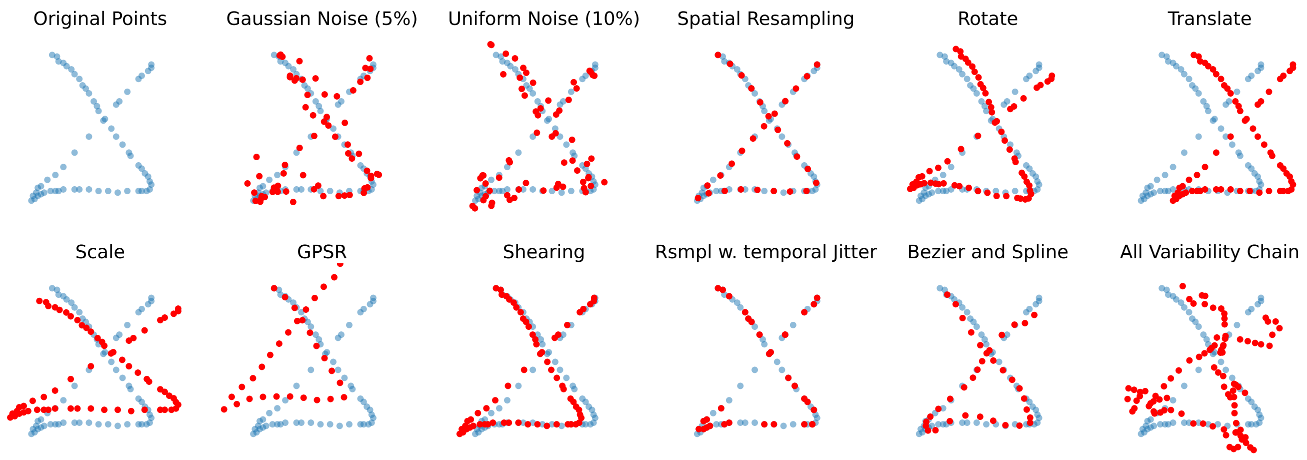

Figure 1: Examples of gesture augmentation technique applied to the "delete" symbol from $\$1 - GDS$ dataset

ABSTRACT

Recurrent neural networks (RNN) require large training datasets from which they learn new class models. This limitation prohibits their use in custom gesture applications where only one or two end user samples are given per gesture class. One common way to enhance sparse datasets is to use data augmentation to synthesize new samples. Although there are numerous known techniques, they are often treated as standalone approaches when in reality they are often complementary. We show that by intelligently chaining augmentation techniques together that simulate different gesture production variability types, such as those affecting the temporal and spatial qualities of a gesture, we can significantly increase RNN accuracy without sacrificing training time. Through experimentation on four public stroke-based 2D gesture datasets, we show that RNNs trained with our data augmentation chaining technique achieves state-of-the-art recognition accuracy in both writer-dependent and writer-independent test scenarios.

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**; **User interface programming**; • **Theory of computation** → **Pattern matching**.

KEYWORDS

datasets, neural networks, gesture recognition and customization, data augmentation

ACM Reference Format:
Mykola Maslych, Eugene M. Taranta II, Mostafa Aldilati, and Joseph J. LaViola Jr.. 2023. Effective 2D Stroke-based Gesture Augmentation for RNNs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3544548.3581358

1 INTRODUCTION

Gestural interfaces present an intuitive way to interact with software, such as through swipes on mobile devices and application-specific shortcuts. However, for general acceptance, they require high classification accuracy. User tolerance for errors towards gestural interfaces in the presence of a reliable alternative input method was found to be approximately 40% [25], and it was indicated that gesture recognizer accuracy should be much greater than 90% to "appear indistinguishable from a perfect system" [3]. Such accuracy is possible with predefined gestures using robust recognizers trained with lots of data, but more difficult with custom gestures where only a few samples per gesture are given. One common approach that overcomes this limitation are k-NN based pattern matching algorithms [48, 58, 62, 73]. However, the draw back of k-NN recognizers is that they rely on manual feature extraction or resample each input trajectory to a constant number of points, which limits their robustness to variability.

Approaches based on recurrent neural networks (RNNs) are more robust to gesture variability, allow for variable-length inputs, and require no hand-crafted feature extraction [28, 35]. However, RNNs have two shortcomings which prevent their adoption as the go-to for customizable gesture recognizers: (1) they require a large amount of data to outperform alternative approaches, and (2) with a large number of samples, they take long to train. Data augmentation techniques [22, 24, 40, 56, 75] offer a solution to the lack of training samples problem. However, solutions to the significant training time for RNNs have not been previously explored. Because RNN training time depends primarily on the number of training samples, the two issues are related. The more data is used for training, the higher the resulting recognition accuracy will be, but at the cost of a longer training time.

In this work we evaluate the effectiveness of various data augmentation approaches on training highly accurate RNN gesture recognition systems. We show that highly accurate custom gesture recognition is possible with just one or two original training samples per class, given that appropriate augmentation techniques are used to generate more samples. We tested each technique applied separately and in combinations with others in writer-dependent and writer-independent experiments which simulate real-world use-cases. We also introduce multiple augmentation *chaining* methods which outperform the alternative approaches in the literature. These chaining methods apply a sequence of augmentation methods from different categories to the data, which works better than applying a single or only a few categories of variability at a time. Lastly, we show that it is possible to train an RNN gesture recognizer with no original validation data, by generating a synthetic validation dataset which helps with selecting a good training stopping point before the model is overfit.

2 RELATED WORK

2.1 Gesture recognition and customization

Early stroke gesture classification algorithms relying on hand-crafted features go back to Rubine's 1991 [48] recognizer, with its original feature set later being expanded [5] and adapted for 3D gestures [51]. The features are extracted from training samples (templates), and candidate gestures are classified using a linear classifier [48], support vector machines (SVM) [63], linear discriminant classifiers (LDA) or other classical machine learning approaches. This allowed the end user to specify gestures by example and to associate them with software functions, thereby enabling *gesture customization*. User preference for customizable gesture interfaces was also explored. For example, Nacenta et al. found that user-designed gestures are more memorable than pre-designed gesture sets [41], which can be helpful when gestures are used as shortcuts for opening applications or speed dialing [10].

Another popular class of gesture recognizers is the $-family [1, 2, 32, 66, 73], and those inspired by them [58, 62, 64]. These approaches focus on ease-of-implementation and rapid prototyping for gesture customization. They utilize local cost functions, such as euclidean distance (ED) for points or the inner product (IP) for vectors to measure the degree of dissimilarity between a new input and each of the templates. Some perform pairwise comparisons [1, 58, 73], others use dynamic time warping (DTW) [62], and some are articulation-invariant, representing gestures as point clouds [65, 66, 68]. These approaches are direct competition for RNNs, and when it comes to custom gesture recognition, they are considered state-of-the-art; much more work was done in the space than with other approaches. For a survey on 2D stroke gesture recognition we direct the reader to a recent survey by Magrofuoco et al. [37].

Approaches which process a single point at a time and support variable-length and continuous inputs without resampling include Hidden Markov Models (HMMs) [26, 29, 34], RNNs [6, 7, 28, 35], and continuous dynamic programming (CDP) [49, 57, 61]. These are relevant because they allow starting the classification process while the input is still being produced by the user. The segmentation problem lies in identifying the gesture starting and ending points within the continuous data sequences containing them. RNN, CDP and other case-specific approaches have been successful in solving this issue [6, 8, 29, 34, 49, 57, 61].

In this work we are evaluating the usefulness of augmentation techniques for position-based time-series RNN approaches. Relevant work in the space of RNN approaches includes DeepGRU [35], which consists of an encoder network, attention module, and a classification module. It was shown to achieve and outperform state-of-the-art approaches across many datasets. Similarly, multiple sizes of single-task and multi-task LSTM-based networks were evaluated by Ledda and Spano [28]. They were trained on synthetic versions of the single-stroke $1 [73] and multi-stroke $N [1] datasets generated with Gestures a Go Go [30], and tested on the original dataset's gestures.

These approaches work well for cases where customization is not a concern, but related work in this domain reported long training times [28, 35, 36], which may not be suitable for customization. Another concern with these existing approaches is the amount of original data required for achieving high accuracy. Data sets of such large sizes are not available in real-world customization scenarios, so our solution with chaining the augmentation approaches is distinct in that it works well even with a very limited amount of original training data.

2.2 Gestural data augmentation

To improve performance of Neural-Network-based approaches, in many fields practitioners and researchers turn to data augmentation, which refers to the practice of synthetically generating samples used for training the models. A slew of techniques have been used for images [11, 39, 53], voice [14, 50, 52] and video sequences [38, 81]. Vision-based optical character recognition (OCR) and handwriting recognition methods similarly rely on data augmentation [13, 19, 74]. While the mentioned approaches were not originally designed for gestural data, some of them work with sequential data, so can be applied to gestures.

For gestures, there have been methods to improve the classification accuracies through synthesizing new samples, however, they are heavily focused on gesture realism. Leiva et al. introduced a web application called *Gestures à Go Go* [30] to produce synthetic samples from real data based on kinematic theory of rapid human movement [44] and a Sigma-Lognormal ($\Sigma\Lambda$) model [4, 45]. Taranta et al. released *Gesture Path Stochastic Resampling* (GPSR) [59], which performs resampling, random point removal, and between-point vector normalization to synthesize new samples. *Gestures à Go Go* and *DeepNAG* [36] experimentally found that their methods synthesize gesture which participants take for real ones. However, in data augmentation, gesture realism is not the main goal, so here we focus on the techniques that will result in the biggest accuracy boost, not the beautification of the synthesized gestures.

DeepGRU [35] accuracy results were reported with utilizing random scaling, translations, rotations, and GPSR [59]. Further, LSTM-based architectures applied local resampling to make the distance between each pair of points constant, and subsequently applied random scaling and translations [28]. In comparison, authors of *DeepNAG* [36] compared their GAN (generative adversarial network) and NAG (non-adversarial generation) approaches with random Gaussian noise and GPSR, and found that their methods improved model accuracy more than the alternatives. Although the above mentioned RNN-based approaches have specified the details of augmentation they applied [28, 35, 36], it remains unclear whether the best combinations of methods are being utilized. In this work we aim to fill this gap by conducting a series of experiments targeted at studying specifically the various gestural augmentation methods and how they can be combined.

3 METHODS

3.1 System overview

All experiments that we conduct in this paper use one of the two pipelines shown in Figure 2. The pipeline on the left (a, Regular pipeline) represents a usual RNN training pipeline, where an original dataset is split into three sets: training, validation and testing, with no overlap between them. The training set is augmented using augmentation techniques under evaluation. The pipeline on the right (b, Synthetic validation pipeline) differs in that it simulates a scenario where the testing set is extremely limited; this pipeline generates a synthetic validation dataset in place of a real dataset. To optimize the RNN architecture and select parameters for the evaluated augmentation techniques, we use the Regular pipeline. This covers Experiment 1 (Section 4.2) and Experiment 2 (Section 4.3). To measure the performance of our combined augmentation techniques in a scenario which approximates a real-world use-case, we use the Synthetic validation pipeline. We use it in our Experiment 3 (Section 4.4), Experiment 4 (Section 4.5), Experiment 5 (Section 4.6), and in the experiment that we present as part of the discussion (Section 5.1).

3.2 Data Preprocessing

In this work we are focusing on enabling real-time (mid-gesture) classification, which is achieved by re-using and updating the "hidden state" of the RNN from one point to the next. We use the last hidden state for classification, and due to the limited pre-processing steps that we apply, practitioners can pass a single point at a time to the model and still get a classification result. We ran empirical tests on subsets of the \$1-dataset [73] and found that only translation of the first trajectory point to the origin resulted in improved performance when combined with data augmentation. This computation does not require parameter extraction and can be easily applied in real-time, by subtracting the value of the first point of the trajectory input from each subsequent point. As a result, the model becomes translation-invariant. However, the scale, rotation, and point count invariances are handled through the data augmentation process. All results are reported with the point representation of trajectories with translation invariance as just described.

Our approach is different from most literature on gestures, where some commonly removed variability categories include translation, rotation, scale, and point count variabilities. In those cases, an entire gesture's trajectory is available at classification time, and to achieve partial invariance, the trajectory can be resampled to a fixed number of points, rotated by an indicative angle [32], its bounding box scaled to a fixed size, and its centroid translated to the origin. Instead of scaling and translating, trajectories can also be z-normalized. Alternatively, after the uniform resampling step, the trajectories can be represented by normalized direction vectors, which helps deal with scaling and translation. Rotation around the first vector [64] can also be applied to help with rotation invariance. Since our goal is mid-gesture classification, where the full trajectory is not available at classification time, we apply only limited preprocessing and show that our model is still able to learn other variability categories through the process of augmentation.

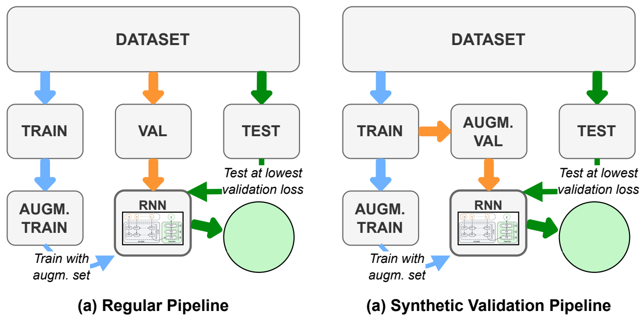

Figure 2: Overview of the evaluation pipeline. Left (a): Regular pipeline, used in Experiment 1 (Section 4.2) and Experiment 2 (Section 4.3); right (b): Synthetic validation pipeline used in Experiment 3 (Section 4.4), Experiment 4 (Section 4.5), Experiment 5 (Section 4.6), and in the test presented in Discussion (Section 5.1). Pipeline (b) generates a synthetic validation dataset for cases where original data set is extremely limited.

3.3 Data Augmentation Techniques

We are interested in data augmentation from the perspective of model classification accuracy, not beautification of synthesized gestures. Even the simplest changes to the original gesture trajectory, such as applying rotations, translations or scaling are valid augmentation techniques — as long as the new data improves the model performance.

In Section 2.2 we provided an overview of the data augmentation approaches used in research and practice. In this section, we summarize the approaches selected for evaluation. To find augmentation techniques which could potentially be used for gestures, we searched the literature for specifically gesture-related data augmentation and synthetic data generation approaches, as well as for those approaches which were not originally intended for gestures, but could be applied to sequential data (for example, techniques originally designed for non-gesture time-series data such as video or voice). Table 1 provides a summary of the identified approaches with citations to their respective sources[1].

For our main evaluation in Experiments 1, 2 and 3, we only focus on augmentation methods that can be described by simple transformations and require a single original sample per class. In Experiment 4, we also compare our best combined augmentation method to two alternative gesture generation methods: GPSR [59] and Sigma-Lognormal ($\Sigma\Lambda$) [4, 30, 45] methods. We considered including DeepNAG in Experiment 4, but training a deep generative model requires thousands of original samples in the first place, and this amount of data is not available in a customization scenario, therefore we only evaluated approaches that can generate new samples without pre-training a model. Fusion and averaging methods [15, 77] were also excluded because they require more original samples per class than afforded in our experiments. Lastly, augmentation using *random translations* [28, 35] was also not used because our pre-processing steps make our model translation-invariant.

Table 1: Gesture augmentation techniques selected for evaluation.

Technique Name	**Reference**
(1) Noise (Uniform/Gaussian/Perlin)	[12, 36]
(2) Scaling	[28, 35]
(3) Rotation	[11, 28, 35, 53, 73]
(4) Slanting (shearing)	[11, 17, 46]
(5) Camera direction change (perspective)	[47, 71, 78]
(6) Spatial resampling	[28, 66, 73]
(7) Temporal resampling	[33, 55, 76]
(8) Temporal jitter	[18, 81]
(9) Time stretching (point duplication)	[24, 54, 72]
(10) Frame skipping	[20]
(11) Bezier and spline deformation	[16, 31, 69, 77, 79]
(12) Random erasing (replacing with 0s)	[53, 80]

The following list is a detailed description of each of the techniques in the order they are presented in the table.

[1]This table does not identify the *first* mention or use of each technique, but instead lists the most relevant (subjectively) references for each technique's usage.

(1) **Noise:** Different kinds of random noise can be added to sequence data, including uniform, Gaussian [36] and Perlin [12] noises. Adding random noise to training data allows the model to learn a more robust representation.
(2) **Scaling:** Scaling each feature (dimension) by a percentage of its own bounding box generates synthetic data which helps the trained model perform better with test samples of varying scales. This is valuable for models which are not scale-invariant. There are examples of scaling being successfully applied in training trajectory-based gesture RNNs [28, 35].
(3) **Rotation:** Rotating 2D trajectories around their centroids allows the trained model to account for when users input gestures at different angles. This approach has been successfully applied in both nearest-neighbor [73] and RNN approaches [28, 35].
(4) **Shearing:** Shearing is a transformation that results in "stretching" of the original trajectory along a line in the trajectories' coordinate system. Points which lie farther from the origin are affected more. To the best of our knowledge, this augmentation technique has not been tested with RNN-based gesture recognizers, however it has been extensively used in the field of handwriting recognition [17, 46].
(5) **Perspective change:** Changing the perspective on the gesture trajectory can be thought of as pointing a camera at the trajectory from a different angle. This technique is often used with 3D skeletal data to achieve view-invariance [47, 71, 78]. When working with 2D gestures drawn on an XY coordinate plane, perspective change refers to rotating the trajectory around the X and Y axes. In practice this is done by adding a third dimension to the 2D data and multiplying the points of an original trajectory by the X and Y rotation matrices.
(6) **Spatial resampling:** Spatially resampling the trajectory refers to generating a list of distance intervals and using such a list to sample points along the original trajectory. The resulting points form a trajectory which looks similar to the original, but with modified distances between subsequent points. Uniform spatial resampling fixes the number of points to describe a trajectory to a constant, which forces the between-point intervals along the original path to be equal for a given gesture. Such uniform approach has been extensively used in k-nearest-neighbor classifiers to account for variability in gesture completion time and point counts [32, 58, 73]. An alternative approach to uniform gesture resampling was used with LSTMs [28], where the intervals' length was fixed to a constant value.
(7) **Temporal resampling:** Temporal resampling is similar to spatial resampling in a sense that a list of intervals is generated, except the intervals are in the time domain. By assuming that the sampling rate in the original data is constant, it is possible to extract the velocity information of the trajectory without the actual timestamps. Sampling at the generated interval lengths along such a velocity profile produces points along the original gesture path. Data augmentation methods similar in concept have been used with noticeable performance gains in LSTM-based full-body skeleton human action recognition tasks [33, 55, 76].

(8) **Temporal jitter:** Resampling with temporal jitter (frame jitter) was applied to gestural video data [18, 81]. The idea is to sample frames at indices $i \pm r$ position, where r is random offset for each new index (for example, when $i = 5$ and $r = 2$ keep each $5_{th} \pm rand_i(-2, 2)$ point). This approach can be though of as a special case of temporal resampling with varying intervals.

(9) **Time stretching:** Through duplicating a random subset of points in the trajectory we can simulate a scenario where software samples faster than hardware, resulting in repeated sampling of the same value. Point duplication method has been used for generating synthetic datasets for DTW algorithms [24, 54, 72], and it could potentially be useful for training RNNs to have robust performance under noisy sampling. This technique can also be thought of as a special case of temporal resampling with allowing for zero-length intervals.

(10) **Frame skipping:** The opposite of point duplication is the complete removal of some of the points from the trajectory. In this way, frame skipping has the opposite effect from point duplication, each point has a non-zero chance of being deleted. This approach was mentioned in a survey on human action recognition [20] and could be useful for improving robustness to noise in gesture classification RNNs.

(11) **Bezier and Spline deformation:** Many approaches which use Bezier curves and splines were proposed for synthetic data generation for drawings [16, 77, 79] and gestures [31, 69]. The generation steps consist of finding reasonable control points (knots), then perturbing them, and finally fitting the new points to a spline curve. To learn more about the implementation details of such approaches, we direct the reader to the cited literature in the *Bezier and spline deformation* row of Table 1.

(12) **Random Erasing:** Sometimes a system can lose tracking of where the user pointer/pen tip is, resulting in some trajectory points being recorded as 0s. This can be dealt with by filtering out such values, but an alternative way is to train a model to handle this kind of input. Loosely inspired by an augmentation method for CNNs, where blocks of image pixels are "blacked-out" [53, 80], we replace some of the trajectory with values of 0 in hopes for the model to learn a potentially lossy trajectory representations.

4 EVALUATION

4.1 RNN model and implementation details

We tried to make our RNN architecture as generic as possible, while keeping in mind a few requirements. The model has to be (1) as small as possible with (2) no sacrifice in performance[2], and (3) easily modifiable in size (add/remove layers, neurons per layer). The first and second requirements ensure that the model fits the customization requirement: the user will benefit from high classification accuracy and short model training times. The third requirement allows us to quickly scale up the model if that is required. We started our implementation by creating a model similar to ST-S [28], and then

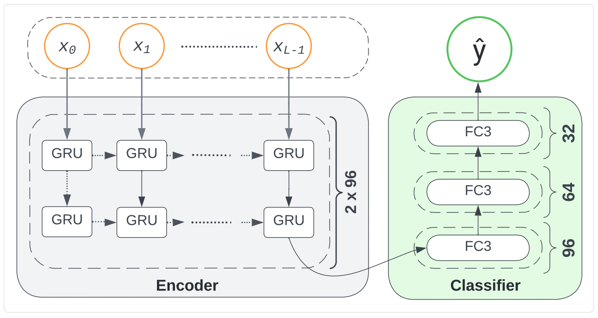

Figure 3: RNN Architecture used in the evaluation of augmentation techniques. *Encoder* consists of two gated recurrent unit (GRU) [9] layers and the *classifier* consists of three stacked feed-forward linear layers with batch normalization [23] and dropout [21] between them. The input is a sequence of 2D vectors of arbitrary length $\mathbf{x} = (x_0, x_1, ..., x_{L-1})$ and the output $\hat{y}$ is the predicted class label.

progressively decreased its size during the implementation of augmentation techniques, until further reduction in size was hurting performance. During this phase, we utilized only a subset of the 1-GDS [73] dataset, which is one of the limitations of our work, however classification results in our Evaluation Section 4.4 show that the model works well for multiple datasets. As part of this process, we ran an informal test to find a trade-off between training time (which depended on the number of synthesized samples) and accuracy, and found that 300 synthetic samples per class worked well for all augmentation techniques. At first, we had ReLU [42] layers in the *classifier* (see Figure 3), but replacing them with batch normalization [23] and aggressive dropout [21] values yielded better results than using non-linear layers. Our resulting model is modest in size and consists of two components: a double GRU layer, connected to a stack of three Linear layers.

We implemented our model using PyTorch Lightning[3], a wrapper around the PyTorch framework [43]. The model takes raw unprocessed (no normalization, no standardization, no filtering) input trajectories from the evaluated datasets. We utilized the Adam optimizer [27] with learning rate set to 10^{-3}, batch size of 512, and Cross Entropy Loss criterion. During the development and for timing the trial runs we used a machine running Windows 10 with a single NVIDIA GeForce GTX 1080 GPU, Intel Core-i7 7700K processor and 16 GB RAM. We ran the repeated experiments in parallel (8 at a time) on a GPU cluster (single GPU per experiment). Reference implementation of the model and augmentation techniques will be available with the camera-ready version.

Earlier in Section 3.3 we listed all individual augmentation techniques that we implemented. To find optimal parameters for each one of them, we utilized a subset of the $1-GDS dataset and ran a series of iterative searches over the ranges of possible parameters. To reduce the number of variables in our evaluation, we used these parameters across all our further experiments and datasets. For

[2]As compared to the larger model that we started with

[3]https://www.pytorchlightning.ai/

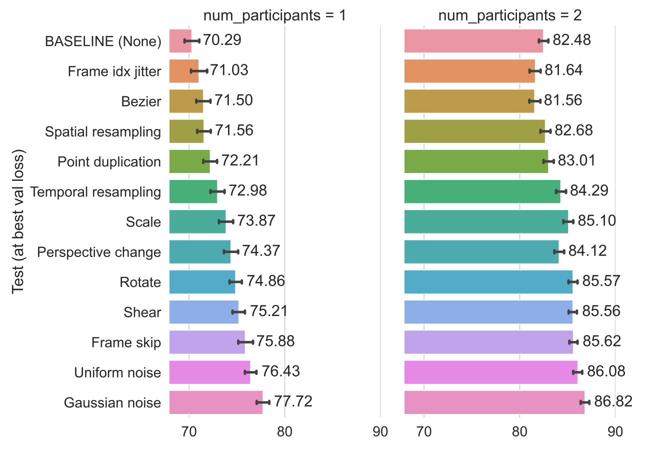

Figure 4: Experiment 1: accuracy results for a writer-independent experiment (train with one participant, test with another) over varying augmentation technique and the number of training participants. A single original sample per participant per class used as a seed samples for generating synthetic samples. Bars indicate a 95% confidence interval.

exact parameters, please refer to the supplementary material and to the reference implementation[4].

4.2 Experiment 1: Performance of individual techniques

4.2.1 Experiment 1 Setup: Part one of our evaluation is dedicated to testing the effect of individual augmentation techniques applied to the data. The question it answers is to what extent synthetic data generated by each augmentation techniques represents the real data. This experiment is similar to a standard writer-independent experiment protocol used widely in the gesture customization literature [58, 62, 66, 73], where a very small number of original training samples are used to train a recognizer. We used the $1 dataset for this experiment. The variables we varied were the number of training participants and the augmentation technique. In this writer-independent experiment, a single sample per gesture class was sampled from p random participants and these gestures were used as seed samples to create an augmented training set (300 synthetic samples per class). The validation set consisted of all available data of two random participants different from the training participants, and the testing set consisted of two more randomly selected participants, different from both the training and validation sets. The sizes of the resulting sets (train:val:test) were 4800+X:960:960, where X denotes the number of original train seed samples. The resulting augmented training dataset was used to train the RNN model and test accuracy is saved at the point of lowest validation loss. We ran this experiment 300 times varying p of 1 and 2, then averaged the results.

4.2.2 Experiment 1 Results: A plot for visual comparisons and exact values of the Experiment 1 results can be found in Figure 4. With a single training participant and a single training sample per class, the baseline (no augmentation applied) for all other scores is 70.29%. All augmentation techniques boosted the model accuracy, except for *erasing*, which decreased the accuracy to 65.67% and was therefore excluded from further evaluation. *Gaussian* augmentation gave the largest improvement to the RNN, resulting in accuracy of 77.72%. With a single training sample from two participants, the *baseline* increased to 82.48%, and *Gaussian* scored at 86.82%. Performance gains from the rest of the techniques were smaller, but *Frame skipping*, *Shearing*, *Rotations* and *Perspective change* all increased accuracy by more than 4%. Interestingly, with two training participants, *Frame index jitter* and *Bezier and Spline deformation* techniques decreased the model accuracy from 82.48% down to 81.64% and 81.56% respectively. It is possible that these techniques do not add enough variability to the data so that the presence of additional original training samples in itself boosts accuracy more.

This experiment paints a high-level picture of what accuracies can be expected when using individual augmentation techniques. However, we want to improve the classification scores further by combining the effect of multiple techniques to synthesize new gesture trajectories. This leads us to our next experiment, where to a gesture trajectory we apply multiple augmentation techniques one after another.

4.3 Experiment 2: Chaining multiple techniques

4.3.1 Setup. In the first part of the evaluation we studied the effect of the applying individual augmentation techniques on classification performance. The next question is whether applying multiple augmentation techniques (from now on we will refer to this process as *chaining*) to the same synthetic sample yields a classification accuracy boost. As a specific example: is applying a chain of *Rotate*→*Scale*→*Perspective*→*Gaussian* transformations better than applying only one of them? Further, if the order in which techniques are chained has impact on how the chain performs, all permutations of all combinations of 11 different techniques make up $T = \sum_{i=1}^{11} 11!/(11-i)! = 108,505,111$ possible chains. Even further, optimal parameters for techniques may be chain-dependent, which together with the fact that computing a good accuracy average for a chain requires 300 model training runs[5], means that an iterative search for the best augmentation chain is not practical. To partially deal with this issue, we fix the parameters for individual techniques, making it possible to directly compare two chains to one another. Using this method we can show, for example, that a chain under the name "*rotate scale perspective gaussian*" in Figure 5 outperforms each of the individual techniques that we compared in this experiment. More similar examples are presented in the same figure.

4.3.2 Categories of variability introduced by augmentation. To reduce the search space of possible chains, we decided to chain together those techniques which through their transformations added variability from different categories to the trajectories. In the nearest-neighbor gesture recognition literature, pre-processing steps reduce and remove specific variability categories [32, 58, 64, 66, 68, 73]. For example, *Noise* can be removed by filtering; *orientation* variability, by rotating by an indicative angle [32]; *scale*,

[4]https://github.com/maslychm/gesture_augmentation

[5]Note that an individual run randomly selects 1 out of 30 samples per class across 10 participants.

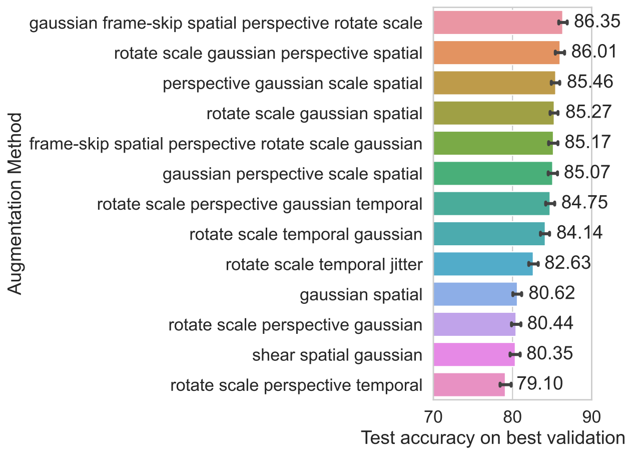

Figure 5: Experiment 2: Accuracies for selected chains in a writer-independent experiment on the GDS dataset. Bars indicate a 95% confidence interval.

by re-scaling the bounding box; and *point count*, by resampling trajectories to be represented by a fixed number of points. We also included *shape* variability, that we define as "change in trajectory shape, noticeable even after scaling, rotating and perfectly overlapping the synthetic trajectory on top of the original". For example, GPSR, noise, and Bezier-based augmentation modifies the gesture shape in such a way. On the other hand, resampling and point duplication - does not (unless extremely low resample count is used). Table 2 summarizes the categories of variability for the techniques which performed above the baseline in Experiment 1 (Section 4.2).

Table 2: Categories of variability for individual techniques which outperformed the baseline. Check-marks indicates that the given augmentation technique adds variability from a given category.

Technique	Noise	Scale	Pt. Cnt	Shape	Orient.
Gaussian	✓			✓	
Uniform	✓			✓	
Scaling		✓			
Spat. rsmpl.			✓		
Temp. rsmpl.			✓		
Frame skip			✓		
Point dupl.			✓		
Frame jitter			✓	✓	
Bezier deform.			✓	✓	
Rotate					✓
Shear		✓		✓	✓
Persp. change				✓	✓

4.3.3 Results for selected chains. Given that the full chaining space is large and can not be tested easily, we formed some simple chains which combined techniques from multiple categories, and then modified them as we observed patterns in model accuracies. We used some guiding principles based on our early experimentation. We observed that those chains that added transformations from the same category multiple times generally performed worse than those that did it once. This means that we generally want to apply a single variability category once. We also noticed that presence of all five of the categories generally gave better performance than when fewer than five categories were present. We also observed that *Gaussian noise* applied before *spatial resampling* gave better performance than when it was applied after it, and similarly for *frame skipping*. This means that the order of applied techniques is important and we should maximize the accuracy be preserving orders that worked well. Guided by these observations, we used trial-and-error and arrived at a chain "*gaussian frame-skip spatial perspective rotate scale*" (All Variability Chain or AVC) which in our testing outperformed other chains that we tested. Results of this experiment can be found in Figure 5. We decided to use AVC in the next experiment, where we pit it against alternative data augmentation techniques in a real-world-like writer-dependent scenario, over multiple datasets.

4.4 Experiment 3: Customization scenario

4.4.1 Experiment setup. A writer-dependent test with a very limited number of original training samples simulates a real-world scenario where a user inputs only a few gestures per class as examples. Validation data, used to gauge if at a particular training epoch the model performs as expected, is also not available in such cases. To deal with this issue, we generate a synthetic validation dataset using the same augmentation process as for training, but double the number of synthetic samples. Without this step the models performed worse due to overfitting.

Our writer-dependent experiment had the following protocol: select k original samples per class from a single participant, and use them as seed samples to generate an augmented training set with 300 synthetic samples per class; use the same original samples to generate a synthetic validation set with 600 synthetic samples per class; use the remaining original samples which were not used for training and form a test dataset (no overlap between train, validation and test sets); train the model using the training and report the test accuracy from the model which had the lowest validation loss. We repeated this experiment for each participant in the dataset, 30 times per participant, for each evaluated augmentation technique.

To evaluate performance of AVC, we used *None* augmentation, and also the best-performing individual augmentation technique from Experiment 1 (Section 4.2): *Gaussian*; *Simple* (*Rotate* $\rightarrow$ *scale* $\rightarrow$ *gaussian*) chain is common in RNN gesture literature [28, 35, 36]; AVC applies *gaussian* $\rightarrow$ *frame-skip* $\rightarrow$ *spatial* $\rightarrow$ *perspective* $\rightarrow$ *rotate* $\rightarrow$ *scale.*

4.4.2 Experiment Results. We ran this writer-dependent test protocol on four single-stroke 2D gestural datasets commonly used as benchmarks in the literature: $1-GDS [73], EDS1 [70], EDS2 [70], and Lemarchand [60]. Figure 6 and Figure 7 show the results for the writer-dependent experiment with synthetic validation data over 1 and 2 original training samples, respectively. For every dataset and the number of original training samples per class, *AVC* chain

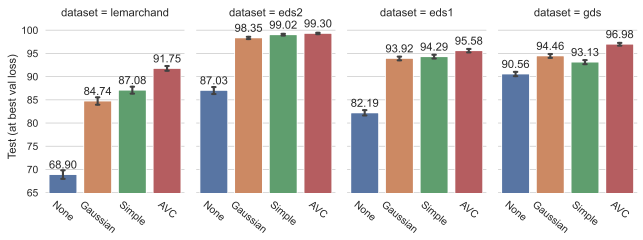

Figure 6: Experiment 3: Writer-dependent experiment with a single training sample per class across GDS, EDS1, EDS2 and Lemarchand datasets. *None* is no augmentation; *Simple* chain is (*rotate* → *scale* → *gaussian*); *Gaussian* is random Gaussian Noise; AVC chain is *gaussian* → *frame-skip* → *spatial* → *perspective* → *rotate* → *scale*. Confidence interval bars are 95%.

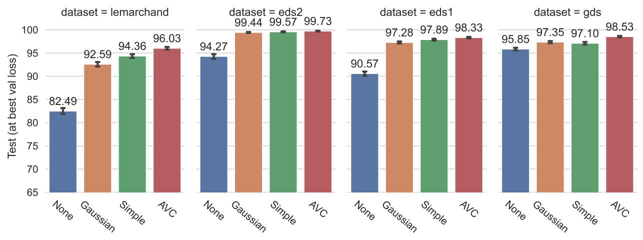

Figure 7: Experiment 3: Writer-dependent experiment with two training sample per class across GDS, EDS1, EDS2 and Lemarchand datasets. *None* is no augmentation; *Simple* chain is (*rotate* → *scale* → *gaussian*); *Gaussian* is random Gaussian Noise; AVC chain is *gaussian* → *frame-skip* → *spatial* → *perspective* → *rotate* → *scale*. Confidence interval bars are 95%.

boosts the performance over *Gaussian* augmentation the most. *Simple* chain also improves recognizer accuracies and in most cases outperforms augmentaiton using *Gaussian* noise.

4.5 Experiment 4: Alternative approaches in customization scenario

4.5.1 Experiment Setup. We are interested in how existing alternative gesture synthesis methods compare to our *AVC* chain in a customization scenario. As introduced in Section 2.2, more complex methods for generating synthetic gestures are available, namely Gestures à Go Go [30], GPSR [59] and DeepNAG [36]. Gestures à Go Go extracts per-trajectory reconstruction parameters and passes them to a Sigma-Lognormal ($\Sigma\Lambda$) model where these parameters are used to synthesize a new trajectory. GPSR first spatially resamples the original trajectory using intervals of random lengths, then removes a small number of points, and lastly normalizes the distances between the remaining points to synthesize new samples. These two methods require only a single original sample per class to produce new trajectories so they are applicable in a customization scenario. DeepNAG, on the contrary, needs thousands of original samples to generate new samples, and its generative model has to be trained anew for each dataset. The amount of data it requires is not available in the customization context, so we only included Gestures à Go Go and GPSR in this experiment.

Both Gestures à Go Go and GPSR take user-defined parameters that influence how variable the synthesized trajectories will look like. To generate synthetic samples with Gestures à Go Go, we used a public API[6] that was released together with the original paper, and set the parameters to those recommended in the method's publication: *shape variability* to 1, *length variability* to 1, *same timestamps* to false. To generate synthetic samples with GPSR, we implemented the algorithm locally and set the *variance* parameter to 0.25 according to the method's publication. We set the *resample count* parameter to a random number between half of the original trajectory length and double the original length, and *remove count* to a random number between 2 and 6. We used the respective parameters for Gestures à Go Go and GPSR to generate 300 synthetic trajectories per original sample in the \$1-GDS dataset, and ran a writer-dependent experiment with synthetic validation pipeline.

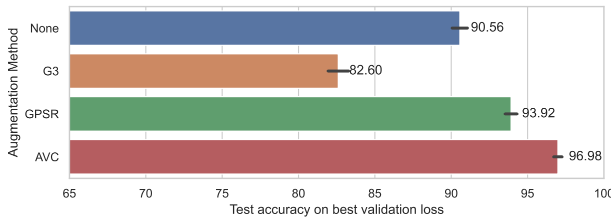

Figure 8: Experiment 4: Writer-dependent experiment with a single training sample per class. \$1-GDS dataset, synthetic validation pipeline. Bars represent a 95% confidence interval.

4.5.2 Experiment Results. Figure 8 shows the results for the writer-dependent experiment where we compared two alternative gesture synthesis approaches to our *AVC* chain. *AVC* chain outperformed both GPSR and Gestures à Go Go methods, with a notable difference. From no augmentation, GPSR improved the accuracy up to 93.92% (by approximately 3.5%). Interestingly, augmenting the dataset using Gestures à Go actually decreased the RNN accuracy by approximately 8%, reducing the average accuracy to 82.6%. A possible explanation for this method performing poorly is that it does not produce enough variability to cover the gesture variability present in the test set, and the RNN overfits as a result.

4.6 Experiment 5: Synthetic training set size VS training time VS accuracy

Recognizer training time is an important consideration in customization. Training times should be as short as possible since that will eliminate the waiting on the user's end. Unfortunately, RNNs are slow to train, and the training times increase when the training set size is increased. Prior research reported training times ranging from 10 minutes to several hours [28, 35, 36], and since the reduction of training time is one of the focal points of our work, we conducted an experiment where we varied the training set size.

[6]https://g3.prhlt.upv.es/

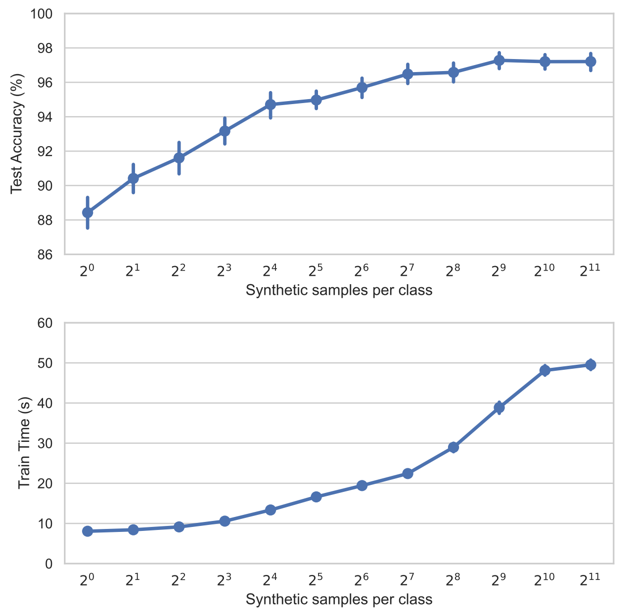

Figure 9: Writer-dependent experiment with varying the number of synthetic samples per class with $1-GDS dataset (using synthetic validation pipeline). Top: training accuracy given varying synthetic training set size, bottom: training time (in seconds) given varying synthetic training set size.

We used the *AVC* chain to generate a variable number of synthetic samples for a writer-dependent experiment with Synthetic Validation pipeline and ran this experiment on Pop!_OS[7][8] with an Nvidia GTX1080 graphics card.

Our results show that with a single synthetic sample per class (total train set size of 32), training time is 9 seconds on average, with 64 synthetic samples per class (total train set size of 1040), the training time increases to 20 seconds (see Figure 9). We tested up to 2048 synthetic samples per class, which brought the total training set size up to 32874, and found that the model took 50 seconds to train with this train set size. The model accuracy reached 97% and stopped improving with values larger than 512 synthetic samples per class, which is close to 300 - a hyper-parameter value for the number of synthetic samples per class that we chose for our previous experiments.

5 DISCUSSION AND FUTURE WORK

The goal of this work is to improve the user experience with gestural interfaces through improving the performance of RNNs used for recognition. With RNNs, the two primary concerns are recognition accuracy and training time. Training time depends on the architecture and training set size. In Experiment 1 (Section 4.2), we

[7]https://pop.system76.com/

[8]At the time of this writing, the Windows implementation of PyTorch DataLoader did not support multiple workers for batch loading, which caused the training time graph to spike at 32 synthetic samples per class, so we switched to a linux-based OS for this experiment.

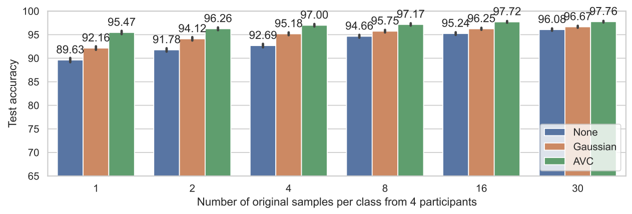

Figure 10: Writer-independent experiment with 4 training participants over a varying count of original training samples per participant on the $1-GDS dataset. *None* is no augmentation; *Gaussian* is random Gaussian Noise; AVC chain is *gaussian* → *frame-skip* → *spatial* → *perspective* → *rotate* → *scale*. Confidence intervals are 95%.

saw many individual augmentation methods which are not mutually exclusive. Because each of them adds variability from different categories, they can be combined into *chains* of augmentation. We used this and were able to boost the RNN model performance while keeping the synthetic dataset limited in size.

5.1 Performance due to augmentation chaining

Through the process we followed in our Experiment 2 (Section 4.3), we arrived at our *AVC* chain, which outperformed the tested alternatives across four 2D gesture datasets in a writer-dependent experiment (Section 4.4) and also alternative methods designed specifically for gesture synthesis (Section 4.5). This confirms that chaining multiple augmentation methods based on variability categories can considerably boost the RNN model performance when compared to other augmentation methods.

5.1.1 Chaining augmentation reduces the amount of required data. To our knowledge, the exact experiment protocol we performed was only used with nearest-neighbor methods, that have been considered the state-of-the-art for gesture customization. For example, on the $1-GDS dataset, the $1 algorithm is 97.1% accurate in a writer-dependent experiment with a single training participant. In the same experiment, our *AVC* augmentation method achieves 96.98% accuracy (see Figure 6). Other work where RNNs were trained on gestural data include thousands of original samples in the training dataset. For example, thousands of original $1-GDS samples, processed using Gestures à Go were used to train a large multi-modal LSTM network and the achieved accuracy was 97.11% [28]. In another experiment, a GRU-based [36] network trained with 50% of $1-GDS and augmentation using a non-adversarial generation approach was 94% accurate. To have a more direct comparison to these testing protocols, we used the *AVC* augmentation chain with only 1.33% of all available data (a single training sample per class from 4 participants) and found that RNN achieves over 95% accuracy. Using 5.33% (four training samples per class from 4 participants), yields 97% accuracy (see Figure 10). This highlights that our approach of combining multiple categories of variability boosts recognition accuracy of RNNs beyond what was previously achieved without requiring nearly as much data as alternative appoaches.

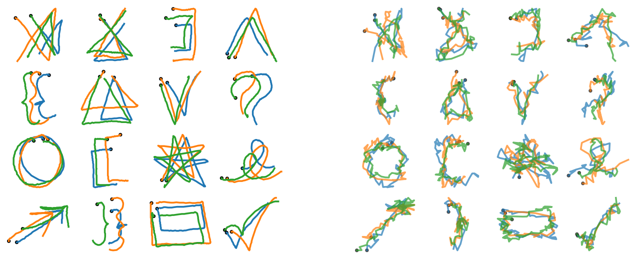

Figure 11: Left: original 2D gesture samples from $1-GDS from 3 different subjects (no transformations applied). Right: three synthetic gesture trajectories per class generated through the *AVC* chain augmentation (*gaussian* → *frame-skip* → *spatial* → *perspective* → *rotate* → *scale*).

5.1.2 Training time based on training set size. The training times we report are lower than those reported in the gesture recognition literature using RNNs [28, 35, 36]. This could be due to a combination of factors: the hardware, the choice of operating system, the deep learning framework. The GTX1080 that we used was released in 2016 and is not a fast card by modern standards, especially among hardware that is specialized for deep learning. Despite this, we were able to achieve low training times, partially thanks to early stopping[9] function that tracks validation set loss values and stops the training when loss has not decreased for a set number of epochs. Still, we believe it is possible to reduce the training time further and we plan to work on this in the future.

5.1.3 Variability in synthesized gestural data. Synthetic training data that combines multiple variability categories allows us to reduce the training time since fewer samples are needed to represent the same amount of variability in data. However, it is possible to further reduce the training time. Potential directions for future work in the area include: decreasing model size, pruning the training set to exclude samples which don't contribute to variability, and utilizing transfer learning. Further, additional training data can be collected while the system is already in use, and because RNNs have a constant inference time, user experience will not degrade from additional data. As a result, the user will benefit from a robust customizable gesture recognizer which is quick to train. In future work, we plan to explore each of these directions.

Visual inspection of the trajectories synthesized by the *AVC* chain reveals that the overall gesture shapes are recognizable, although the trajectories look very noisy (see Figure 11). The impact of Gaussian noise is most pronounced, followed by rotations. The effect of spatial resampling is not obvious from a visual observation, however given that presence of point count modification methods is the main difference between the *AVC* chain and *Simple* chain (*Rotate* → *scale* → *gaussian*), the performance gains when using the *AVC* chain must come from frame skipping and spatial resampling methods. Such methods are underexplored in the RNN gesture recognition literature, and based on our results, they deserve more attention and may reveal more performance gains if studied further.

[9] https://pytorch-lightning.readthedocs.io/en/stable/common/early_stopping.html

When it comes to gestures, there are many sources of variability that contribute towards the overall gesture trajectory. The input devices have various screen sizes and can be held at different angles. The user can be right or left-handed, have different vision or motor abilities. Environment also plays a role, since if the user is moving while using gesture shortcuts, the produced trajectories will be more noisy. Lastly, the precision of the device's sensor and the sampling rate also have an impact on trajectory. With a limited training set, the goal of augmentation is to cover the entire space of the possible variability that will be encountered during the use of the system. Thus modeling the potential sources of variability can help inform practitioners what is expected during use and therefore what augmentation parameters to use. Figure 11 shows that even though the variability in original gestures on the left side is different from the variability of trajectories synthesized using the *AVC* chain, the additional noise helps the RNN generalize and perform better on the test data. From a practical standpoint, this means that practitioners should side with as much variability as possible, as long as the test accuracy is not decreased.

5.2 Recommendations for practitioners

When generating synthetic data for training gesture recognition RNNs, it is best to combine multiple data augmentation methods by applying them one after the other to the original gestures. An augmentation chain of *Gaussian Noise* → *Frame Skipping* → *Spatial Resampling* → *Perspective Change* → *Rotation* → *Scaling* works well in practice and can be readily used for generating a synthetic dataset which will improve a 2D gesture recognizer's performance. In general, as much variability as possible should be added as long as performance on the test set improves. Supplementary material to this paper contains the exact values of the parameters used with our augmentation methods, and a reference implementation for researchers and practitioners is available at https://github.com/maslychm/gesture_augmentation.

5.3 Limitations and Future Work

The first limitation of this work is that the RNN model architecture and the individual augmentation techniques parameters were optimized on a single datset ($1-GDS). This leads us to question the generalizability of the AVC augmentaiton method we created. Through additional writer-dependent experiments on three more datsets which are widely-used in the gesture recognition space (EDS1 [70], EDS2 [70], Lemarchand [60]), we confirmed that the *AVC* chain method still outperforms the alternatives when used in combination with our RNN architecture. However, we did not explore the interaction effects between test accuracy, model size, and the number of classes in the dataset. Further improvements in the classification accuracy may be discovered by exploring these parameters. In future work we also plan to further evaluate writer-independent performance across additional datasets, including datasets with handwritten sybmols, 3D gesture datasets and other modalities.

A second limitation is that because of the huge search space in the possible chains in Experiment 2 (Section 4.3), it is possible that a better chaining strategy exists which would lead to even higher classification accuracies. As part of future work, we plan to explore various parameter space search methods that would help with automating the chaining process. It is also possible to combine GPSR and Gestures à Go with other augmentation methods. Utilizing these methods in combination with the ones we evaluated may improve the recognizer performance further, so in future work we plan to include them in our evaluation.

Lastly, as mentioned in the Discussion (Section 5.1.3), modeling the sources of variability may better inform the choice of parameters for augmentation chains. Additionally, measuring the distributions between real and augmented data may aid in understanding why certain variability improves the recognition performance. For this, previous efforts have used Euclidean [73], Inner Product [59, 62] and Cloud distances [66], as well as custom-defined accuracy variability measures [67]. All the mentioned approaches, however, resample the trajectories to a fixed length and through this process remove certain variability that we found to improve performance, such as high frequency noise and point count. Future work should find a way to aggregate multiple distance measures and define custom variability measures that will help inform augmentation methods and parameters choices. For example, using forward modeling could be a viable alternative to this approach. Additionally, such knowledge will help generate more realistic looking samples through the augmentation process, which we leave to future work.

6 CONCLUSION

In this work we implemented and evaluated a number of augmentation techniques for gestural data as applied to training Recurrent Neural Networks. We combined some of these techniques into chains on the basis of different categories of variability which techniques add to the data. These chains apply the augmentation techniques sequentially in a specific order, and we found a chain that performed better than the rest in our evaluation. We called this chain the "All Variability Chain" (AVC) and tested it on four widely-used single-stroke 2D gesture datasets, finding that it outperforms all tested alternatives. Researchers and practitioners will benefit from this new simple-to-implement data augmentation method which will boost their RNNs recognition accuracy. End users will benefit from improved RNN-based gesture recognition systems enabled by methods evaluated in this work.

ACKNOWLEDGMENTS

This work is supported in part by NSF Award IIS-1917728, Northrop Grumman., Unknot.id, and the Florida High Tech Corridor Council Industry Matching Research Program. We also thank the anonymous reviewers for their insightful feedback and the ISUE lab members for their support.

REFERENCES

[1] Lisa Anthony and Jacob O. Wobbrock. 2010. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, CAN, 245–252.

[2] Lisa Anthony and Jacob O Wobbrock. 2012. $ n-protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012 (GI '12)*. Graphics Interface Conference 2012, Toronto, Ontario, Canada, 117–120.

[3] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2014. User adaptation to a faulty unistroke-based text entry technique by switching to an alternative gesture set. In *Proceedings of Graphics Interface 2014 (GI '14)*. Canadian Information Processing Society, CAN, 183–192.

[4] Ujjwal Bhattacharya, Réjean Plamondon, Souvik Dutta Chowdhury, Pankaj Goyal, and Swapan K. Parui. 2017. A Sigma-Lognormal Model-Based Approach to Generating Large Synthetic Online Handwriting Sample Databases. *International Journal on Document Analysis and Recognition (IJDAR)* 20, 3 (Sept. 2017), 155–171. https://doi.org/10.1007/s10032-017-0287-5

[5] Rachel Blagojevic, Samuel Hsiao-Heng Chang, and Beryl Plimmer. 2010. The Power of Automatic Feature Selection: Rubine on Steroids. *SBIM* 10 (2010), 79–86.

[6] Ariel Caputo, Andrea Giachetti, Simone Soso, Deborah Pintani, Andrea D'Eusanio, Stefano Pini, Guido Borghi, Alessandro Simoni, Roberto Vezzani, Rita Cucchiara, Andrea Ranieri, Franca Giannini, Katia Lupinetti, Marina Monti, Mehran Maghoumi, Joseph J. LaViola Jr, Minh-Quan Le, Hai-Dang Nguyen, and Minh-Triet Tran. 2021. SHREC 2021: Skeleton-based Hand Gesture Recognition in the Wild. *Computers & Graphics* 99 (Oct. 2021), 201–211. https://doi.org/10.1016/j.cag.2021.07.007

[7] F. M. Caputo, S. Burato, G. Pavan, T. Voillemin, H. Wannous, J. P. Vandeborre, M. Maghoumi, E. M. Taranta II, A. Razmjoo, J. J. LaViola Jr., F. Manganaro, S. Pini, G. Borghi, R. Vezzani, R. Cucchiara, H. Nguyen, M. T. Tran, and A. Giachetti. 2019. Online Gesture Recognition. In *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, Department of Computer Science, University of Verona, Italy, 10 pages. https://doi.org/10.2312/3dor.20191067

[8] Yineng Chen, Xiaojun Su, Feng Tian, Jin Huang, Xiaolong (Luke) Zhang, Guozhong Dai, and Hongan Wang. 2016. Pactolus: A Method for Mid-Air Gesture Segmentation within EMG. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. Association for Computing Machinery, New York, NY, USA, 1760–1765. https://doi.org/10.1145/2851581.2892492

[9] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014), 15 pages. arXiv:1406.1078 http://arxiv.org/abs/1406.1078

[10] Sung-Jung Cho, Eunseok Choi, Won-Chul Bang, Jing Yang, Junil Sohn, Dong Yoon Kim, Young-Bum Lee, and Sangryong Kim. 2006. Two-stage Recognition of Raw Acceleration Signals for 3-D Gesture-Understanding Cell Phones. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, Guy Lorette (Ed.). Université de Rennes 1, Suvisoft, La Baule (France). https://hal.inria.fr/inria-00103854 http://www.suvisoft.com.

[11] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2019. RandAugment: Practical automated data augmentation with a reduced search space. https://doi.org/10.48550/ARXIV.1909.13719

[12] Kenny Davila, Stephanie Ludi, and Richard Zanibbi. 2014. Using Off-Line Features and Synthetic Data for On-Line Handwritten Math Symbol Recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*. IEEE, Hersonissos, Greece, 323–328. https://doi.org/10.1109/ICFHR.2014.61

[13] Yousef Elarian, Radwan Abdel-Aal, Irfan Ahmad, Mohammad Tanvir Parvez, and Abdelmalek Zidouri. 2014. Handwriting Synthesis: Classifications and Techniques. *International Journal on Document Analysis and Recognition (IJDAR)* 17, 4 (Dec. 2014), 455–469. https://doi.org/10.1007/s10032-014-0231-x

[14] Raul Fernandez, Andrew Rosenberg, Alexander Sorin, Bhuvana Ramabhadran, and Ron Hoory. 2017. Voice-Transformation-Based Data Augmentation for Prosodic Classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5530–5534. https://doi.org/10.1109/ICASSP.2017.7953214

[15] Germain Forestier, François Petitjean, Hoang Anh Dau, Geoffrey I. Webb, and Eamonn Keogh. 2017. Generating Synthetic Time Series to Augment Sparse Datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*. 865–870. https://doi.org/10.1109/ICDM.2017.106 ISSN: 2374-8486.

[16] Donatien Grolaux, Jean Vanderdonckt, Thanh-Diane Nguyen, and Iyad Khaddam. 2020. SketchADoodle: Touch-surface Multi-stroke Gesture Handling by Bézier Curves. *Proceedings of the ACM on Human-Computer Interaction* 4, EICS (June 2020), 1–30. https://doi.org/10.1145/3397875

[17] T.M. Ha and H. Bunke. 1997. Off-Line, Handwritten Numeral Recognition by Perturbation Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 5 (May 1997), 535–539. https://doi.org/10.1109/34.589216

[18] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality. *ACM Transactions on Graphics* 39, 4 (July 2020), 87:87:1–87:87:13. https://doi.org/10.1145/3386569.3392452

[19] Taihei Hayashi, Keiji Gyohten, Hidehiro Ohki, and Toshiya Takami. 2018. A Study of Data Augmentation for Handwritten Character Recognition Using Deep Learning. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 552–557. https://doi.org/10.1109/ICFHR-2018.2018.00102

[20] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. 2017. Going deeper into action recognition: A survey. *Image and Vision Computing* 60 (April 2017), 4–21. https://doi.org/10.1016/j.imavis.2017.01.010

[21] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012), 18 pages. arXiv:1207.0580 http://arxiv.org/abs/1207.0580

[22] Alexander Hoelzemann, Nimish Sorathiya, and Kristof Van Laerhoven. 2021. Data Augmentation Strategies for Human Activity Data Using Generative Adversarial Neural Networks. *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)* (2021). https://doi.org/10.1109/PerComWorkshops51409.2021.9431046

[23] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, In ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning. *CoRR* 37, 448–456. arXiv:1502.03167 http://arxiv.org/abs/1502.03167

[24] Brian Kenji Iwana and Seiichi Uchida. 2021. An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks. *PLoS ONE* 16, 7 (July 2021), e0254841. https://doi.org/10.1371/journal.pone.0254841

[25] Maria Karam and m. c. schraefel. 2006. Investigating user tolerance for errors in vision-enabled gesture-based interactions. In *Proceedings of the working conference on Advanced visual interfaces (AVI '06)*. Association for Computing Machinery, New York, NY, USA, 225–232. https://doi.org/10.1145/1133265.1133309

[26] Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. 2007. The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface. In *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, Boston, MA, USA, 15–22. https://doi.org/10.1109/ISWC.2007.4373770 ISSN: 2376-8541.

[27] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization, In 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. *arXiv preprint arXiv:1412.6980* 3, 1, 15 pages. http://arxiv.org/abs/1412.6980

[28] Emanuele Ledda and Lucio Davide Spano. 2021. Applying Long-Short Term Memory Recurrent Neural Networks for Real-Time Stroke Recognition. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '21)*. Association for Computing Machinery, New York, NY, USA, 50–55. https://doi.org/10.1145/3459926.3464754

[29] Hyeon-Kyu Lee and J.H. Kim. 1999. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 10 (Oct. 1999), 961–973. https://doi.org/10.1109/34.799904 Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[30] Luis A. Leiva, Daniel Martín-Albo, and Réjean Plamondon. 2016. Gestures à Go Go: Authoring Synthetic Human-Like Stroke Gestures Using the Kinematic Theory of Rapid Movements. *ACM Transactions on Intelligent Systems and Technology* 7, 2 (Jan. 2016), 1–29. https://doi.org/10.1145/2799648

[31] Jiajun Li, Jianguo Tao, Liang Ding, Haibo Gao, Zongquan Deng, Yang Luo, and Zhandong Li. 2018. A New Iterative Synthetic Data Generation Method for CNN Based Stroke Gesture Recognition. *Multimedia Tools and Applications* 77, 13 (July 2018), 17181–17205. https://doi.org/10.1007/s11042-017-5285-6

[32] Yang Li. 2010. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2169–2172. https://doi.org/10.1145/1753326.1753654

[33] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. 2016. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 816–833. https://doi.org/10.1007/978-3-319-46487-9_50 REVIEW FOR SURE.

[34] Xiao-Hui Liu and Chin-Seng Chua. 2010. Rejection of non-meaningful activities for HMM-based activity recognition system. *Image and Vision Computing* 28, 6 (June 2010), 865–871. https://doi.org/10.1016/j.imavis.2009.11.001

[35] Mehran Maghoumi and Joseph J. LaViola. 2019. DeepGRU: Deep Gesture Recognition Utility. In *Advances in Visual Computing*. Vol. 11844. Springer International Publishing, Cham, 16–31. https://doi.org/10.1007/978-3-030-33720-9_2

[36] Mehran Maghoumi, Eugene Matthew Taranta, and Joseph LaViola. 2021. DeepNAG: Deep Non-Adversarial Gesture Generation. In *26th International Conference on Intelligent User Interfaces*. ACM, College Station TX USA, 213–223. https://doi.org/10.1145/3397481.3450675

[37] Nathan Magrofuoco, Paolo Roselli, and Jean Vanderdonckt. 2022. Two-dimensional Stroke Gesture Recognition: A Survey. *Comput. Surveys* 54, 7 (Sept. 2022), 1–36. https://doi.org/10.1145/3465400

[38] Ross Messing, Chris Pal, and Henry Kautz. 2009. Activity Recognition Using the Velocity Histories of Tracked Keypoints. In *2009 IEEE 12th International Conference on Computer Vision*. 104–111. https://doi.org/10.1109/ICCV.2009.5459154

[39] Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. 117–122. https://doi.org/10.1109/IIPHDW.2018.8388338

[40] Francisco J. Moreno-Barea, José M. Jerez, and Leonardo Franco. 2020. Improving Classification Accuracy Using Data Augmentation on Small Data Sets. *Expert Systems with Applications* 161 (Dec. 2020), 113696. https://doi.org/10.1016/j.eswa.2020.113696

[41] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1099–1108. https://doi.org/10.1145/2470654.2466142

[42] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Omnipress, Haifa, Israel, 807–814. https://icml.cc/Conferences/2010/papers/432.pdf

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., Vancouver, Canada, 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[44] Réjean Plamondon. 1995. A kinematic theory of rapid human movements: Part I. Movement representation and generation. *Biological cybernetics* 72, 4 (1995), 295–307.

[45] Réjean Plamondon and Moussa Djioua. 2006. A multi-level representation paradigm for handwriting stroke generation. *Human movement science* 25, 4-5 (2006), 586–607.

[46] Robert Powalka. 1993. Experiments With Applying Slant Counteraction to Script Recognition.

[47] Hossein Rahmani and Ajmal Mian. 2016. 3D Action Recognition from Novel Viewpoints. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 1506–1515. https://doi.org/10.1109/CVPR.2016.167

[48] Dean Rubine. 1991. Specifying Gestures by Example. *ACM SIGGRAPH Computer Graphics* 25, 4 (July 1991), 329–337. https://doi.org/10.1145/127719.122753

[49] Yasushi Sakurai, Christos Faloutsos, and Masashi Yamamuro. 2007. Stream Monitoring under the Time Warping Distance. In *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, Istanbul, 1046–1055. https://doi.org/10.1109/ICDE.2007.368963

[50] Jan Schlüter and Thomas Grill. 2015. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks.. In *ISMIR*. 121–126.

[51] Jia Sheng. 2003. A study of adaboost in 3d gesture recognition. *Department of Computer Science, University of Toronto* 1 (2003), 7 pages.

[52] Arash Shilandari, H. Marvi, and H. Khosravi. 2021. Speech Emotion Recognition Using Data Augmentation Method by Cycle-Generative Adversarial Networks. https://doi.org/10.20944/PREPRINTS202104.0651.V1

[53] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6, 1 (July 2019), 60. https://doi.org/10.1186/s40537-019-0197-0

[54] Clifford K. F. So and George Baciu. 2006. Hypercube sweeping algorithm for subsequence motion matching in large motion databases. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications (VRCIA '06)*. Association for Computing Machinery, New York, NY, USA, 221–228. https://doi.org/10.1145/1128923.1128960

[55] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. 2016. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. *arXiv:1611.06067 [cs]* (Nov. 2016). http://arxiv.org/abs/1611.06067 arXiv: 1611.06067.

[56] Odongo Steven Eyobu and Dong Seog Han. 2018. Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network. *Sensors* 18, 9 (Sept. 2018), 2892. https://doi.org/10.3390/s18092892

[57] Jingren Tang, Hong Cheng, Yang Zhao, and Hongliang Guo. 2018. Structured dynamic time warping for continuous hand trajectory gesture recognition. *Pattern Recognition* 80 (Aug. 2018), 21–31. https://doi.org/10.1016/j.patcog.2018.02.011

[58] Eugene M. Taranta and Joseph J. LaViola. 2015. Penny pincher: a blazing fast, highly accurate $-family recognizer. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, CAN, 195–202.

[59] Eugene M. Taranta, Mehran Maghoumi, Corey R. Pittman, and Joseph J. LaViola. 2016. A Rapid Prototyping Approach to Synthetic Data Generation for Improved 2D Gesture Recognition. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 873–885. https://doi.org/10.1145/2984511.2984525

[60] Eugene M. Taranta, Andrés N. Vargas, and Joseph J. LaViola. 2016. Streamlined and accurate gesture recognition with Penny Pincher. *Computers & Graphics* 55 (April 2016), 130–142. https://doi.org/10.1016/j.cag.2015.10.011

[61] Eugene M. Taranta II, Corey R. Pittman, Mehran Maghoumi, Mykola Maslych, Yasmine M. Moolenaar, and Joseph J. Laviola Jr. 2021. Machete: Easy, Efficient, and Precise Continuous Custom Gesture Segmentation. *ACM Transactions on Computer-Human Interaction* 28, 1 (Jan. 2021), 5:1–5:46. https://doi.org/10.1145/3428068

[62] Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. 2017. Jackknife: A Reliable Recognizer with Few Samples and Many Modalities. In *Proceedings of the 2017 CHI*

Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA, 5850–5861. https://doi.org/10.1145/3025453.3026002

[63] Eugene M. Taranta II, Thaddeus K. Simons, Rahul Sukthankar, and Joseph J. Laviola Jr. 2015. Exploring the Benefits of Context in 3D Gesture Recognition for Game-Based Virtual Environments. *ACM Transactions on Interactive Intelligent Systems* 5, 1 (March 2015), 1:1–1:34. https://doi.org/10.1145/2656345

[64] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. 2018. !FTL, an Articulation-Invariant Stroke Gesture Recognizer with Controllable Position, Scale, and Rotation Invariances. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction (ICMI '18)*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/3242969.3243032

[65] Radu-Daniel Vatavu. 2017. Improving Gesture Recognition Accuracy on Touch Screens for Users with Low Vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4667–4679. https://doi.org/10.1145/3025453.3025941

[66] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI '12)*. Association for Computing Machinery, New York, NY, USA, 273–280. https://doi.org/10.1145/2388676.2388732

[67] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2013. Relative accuracy measures for stroke gestures. In *Proceedings of the 15th ACM on International conference on multimodal interaction (ICMI '13)*. Association for Computing Machinery, New York, NY, USA, 279–286. https://doi.org/10.1145/2522848.2522875

[68] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2018. $Q: A Super-Quick, Articulation-Invariant Stroke-Gesture Recognizer for Low-Resource Devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3229434.3229465

[69] R. D. Vatavu, S. G. Pentiuc, L. Grisoni, and C. Chaillou. 2008. Modeling Shapes for Pattern Recognition: A Simple Low-Cost Spline-based Approach. *Advances in Electrical and Computer Engineering* 8, 1 (2008), 67–71. https://doi.org/10.4316/aece.2008.01012

[70] Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, and Laurent Grisoni. 2011. Estimating the Perceived Difficulty of Pen Gestures. In *Human-Computer Interaction – INTERACT 2011 (Lecture Notes in Computer Science)*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Springer, Berlin, Heidelberg, 89–106. https://doi.org/10.1007/978-3-642-23771-3_9

[71] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. 2014. Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Columbus, OH, USA, 588–595. https://doi.org/10.1109/CVPR.2014.82

[72] Qingsong Wen, Liang Sun, Xiaomin Song, Jing Gao, Xue Wang, and Huan Xu. 2021. Time Series Data Augmentation for Deep Learning: A Survey. In *IJCAI*. https://doi.org/10.24963/ijcai.2021/631 Augmentation, Time Series.

[73] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 159–168. https://doi.org/10.1145/1294211.1294238

[74] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. 2016. Understanding data augmentation for classification: when to warp?. In *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 1–6.

[75] Xinyu Yang, Zhenguo Zhang, Xu Cui, and Rong-yi Cui. 2021. A Time Series Data Augmentation Method Based on Dynamic Time Warping. *2021 International Conference on Computer Communication and Artificial Intelligence (CCAI)* (2021). https://doi.org/10.1109/CCAI50917.2021.9447507

[76] Yong Du, Wei Wang, and Liang Wang. 2015. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, 1110–1118. https://doi.org/10.1109/CVPR.2015.7298714

[77] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. 2017. Sketch-a-Net: A Deep Neural Network that Beats Humans. *International Journal of Computer Vision* 122, 3 (May 2017), 411–425. https://doi.org/10.1007/s11263-016-0932-3

[78] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. 2017. View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition from Skeleton Data. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Venice, 2136–2145. https://doi.org/10.1109/ICCV.2017.233

[79] Ying Zheng, Hongxun Yao, Xiaoshuai Sun, Shengping Zhang, Sicheng Zhao, and Fatih Porikli. 2021. Sketch-Specific Data Augmentation for Freehand Sketch Recognition. *Neurocomputing* 456 (Oct. 2021), 528–539. https://doi.org/10.1016/j.neucom.2020.05.124

[80] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 07 (April 2020), 13001–13008. https://doi.org/10.1609/aaai.v34i07.7000 Number: 07.

[81] Guangming Zhu, Liang Zhang, Peiyi Shen, and Juan Song. 2017. Multimodal Gesture Recognition Using 3-D Convolution and Convolutional LSTM. *IEEE Access* 5 (2017), 4517–4524. https://doi.org/10.1109/ACCESS.2017.2684186