Stroke Classification from Wearable Motion Sensors

Yixue Wendy Feng

March 2020

1 Introduction

1.1 Background & Related Work

Mobile sensing technology has motion sensors that can collect motion data at a low cost and has shown promising prospect in healthcare applications. For patients who have experienced stroke or are in rehabilitation, mobile sensing can be used for activity recognition and identifying physiological responses and movement patterns to help prevent recurrent strokes.

Related work has been done in identifying post-stroke motor movement and are mostly conducted in controlled environment. Past studies suggested that there's a large inter-individual variability and no 2 strokes are alike [1]. Semrau et al. (1995) [2] conducted the study on 116 patients 1, 6, 12 and 26 weeks post-stroke to study their stroke recovery. They had participants sit in a robotic exoskeleton and performing proprioceptive and motor function tasks, and found that those with mild to moderate stroke may have longer recovery time that expected. Lowrey et al. (2014) found bi-manual upper extremity assessment help identify impairment in post-stroke patients [3], and is useful since many daily activities requires both hands.

There have also been studies on using wearable motion sensor for post-stroke activity recognition. Cheng et al. (2016) [4] used wearable inertial measurement unit (IMU) devices on home-cared post-stroke patients. They employed a convolutional equalizer preprocessor, polynomial interpolation equalizer preprocessor and ANN postprocessor as their platform for activity recognition, so that extracted features are equal in length to better support the classification task. Another study by Chen et al. (2015) [5] collected motion sensor data from 50 stroke patients by having them perform the Trail Making Test (TMT), a widely-used method for stroke detection, and another body sensing game-based Trail Making Test (BSG-TMT). They achieved 91% accuracy on binary classification by using only 4 selected motor pattern features, and almost 100% accuracy when medical history features are included (one of which indicates whether the patient has previously had a stroke). This work is limited given their limited sample size and data is collected in a very controlled environment. However they provide an approach for feature engineering using mutual information-based feature selection method and k-fold cross validation that I hope to explore in this project.

Most past studies are done in a very controlled environment where patients perform a specific task or activity (walking, sitting, standing, etc), and some used specialized robotic equipment which can be very expensive and hard to obtain. These studies focus on post stroke recovery or study motor patterns in known stroke patients, indicating that patients with stroke history has unique bi-manual motor patterns that can be identified from data collected from wearable motion sensors. In this study, our goal is to be able to identify those patients with stroke history by conducting stroke classification on data collected from portable and cheaper wrist-worn sensors in a less controlled environment. This project also serves as a exploratory study that could potentially be expanded to study a larger pool of patients and identify those experiencing strokes in real-time.

1.2 Data Description

The dataset consists of motion sensor data collected from wrist-worn devices on 62 post-op patients, 21 of which have previously had a stroke and 41 have not. The collection period varies by patient, ranging from 38 to 198 minutes and the data is collected in uncontrolled environment, where there's no data recorded on what activities the patient was engaged at each timestamp. The varying lengths distribution among the dataset can be seen in Figure 1.

The original sampling rate is about 10Hz, and the data is preprocessed by taking the average for each second. Table 1 below shows the 3 features in the dataset (each are tri-axial, with measurements on the x, y and z axis), their units and the percentage of missing data amongst all patients. Percentage of missing data varies among each individual, and a higher percentage of left hand data is missing. Linear acceleration is normalized to $9.8m/s^2$. For both left, right hand, 3 axis and 3 features, there are a total of 18 features in the dataset.

Features (tri-axial)	Unit	Left	Right
Linear Acceleration (Acc)	m/s^2		
Angular Acceleration (Ang)	rad/s^2	6.288%	8.682%
Angular Velocity (Vel)	rad/s		

Table 1: Feature description

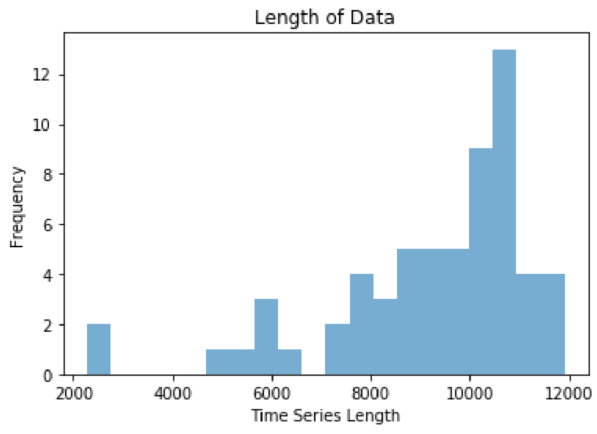


Figure 1: Histogram of Time Series Lengths Among Patients

2 Methods

The goal of this project is to classify whether a patient has had a stroke before based on motion sensor data collected from wrist-worn devices on both hands. The problem can be formulated as a multivariate time series (MTS) binary classification task. The two biggest challenges in this project is preprocessing multivariate varying length times series data and building a robust enough classifier given limited samples ($N = 62$) and unbalanced class distribution. Similar to the pipeline proposed by Tan et al. [6], I used different combinations of data preprocessors and classifiers to compare performances in this time series classification task.

2.1 Data Preprocessing

For data per feature ($P = 18$) per patient ($N = 62$), each times series is scaled using the standard scaler (implementation from scikit-learn), to zero mean and unit variance. The methods for length processing can be a bit tricky. Most classifiers work for only equal length data, so the most straight forward way is to truncate the longer time series. But we also want to keep as much data as possible given the nature of our dataset. The shortest data collection period is only 2278 seconds, where the longest is 11903 seconds. Three methods for data preprocessing was adopted in this project, summarized in Table 2.

	Preprocessor	Shape
Scaling	Standard Scaler	$62 \times 18 \times T^{*}$
Length Processing	Equal Length	$62 \times 18 \times 6000$
	Unequal Length	$62 \times 18 \times T$
	Data Augmentations	$2585 \times 18 \times 600$

* T is a variable, referring to the length of the data collection period, which varies by patient.

Table 2: Data Preprocessors

The easiest way to process time series data into equal length is truncating longer time series and zero padding shorter times series. For the equal length processing, I chose a length of 6000 seconds (100 minutes). While it's on the shorter side, we want the classifier to perform well but also run at a reasonable speed. The first 6000 seconds of the long time series are kept, and the shorter time series are zero padded on both side. Another preprocessor is to keep the varying length data as is, since in real life application, the data collection period can vary due to many reasons, i.e. the patients was taken for scans, or any emergency disruptions that can occur in a hospital setting.

One other limitation I had to work with is the limited dataset of 62 patients. I decided to augment the dataset by sampling 600 seconds (10 minutes) intervals from the data. Another reason is that for long time series, classifiers can take very long to train, making it hard to scale to more larger dataset. The number of samples generated for each patient is length of time series $\times 3/600$, to ensure that there are some overlap between the intervals generated from the same patient, and that those patients who had more data collected have more representation in the augmented dataset. The samples with more than 75% zero values are discarded. Each sample contains all 18 features (the size of one sample is 18×600), and the label for each sample is the label of the patient data it was sampled from.

2.2 Classifier

The classifiers I used in this project are distanced-based machine learning methods and some deep learning methods, summarized in Table 3 (Experiment number is marked in the last column).

Classifier		Distance	Data	Exp
Distance Based	1NN	Euclidean	Equal Length	1
		DTW	Equal Length (Accelerometer Magnitude)	2
		DTW	Equal Length	3
	MDS + SVM	DTW	Unequal Length	4
Deep Learning	LSTM	/	Augmented Data	5
	SepConv	/	Augmented Data	6

Table 3: Model Description

2.3 Distance-Based Machine Learning Classifiers

Euclidean distance based classifiers can only be used for equal length data, where dynamic time warping (DTW) distance can be applied to varying length data. I used implementations of 1NN classifiers from scikit-learn package, and DTW distance from the pyts package.

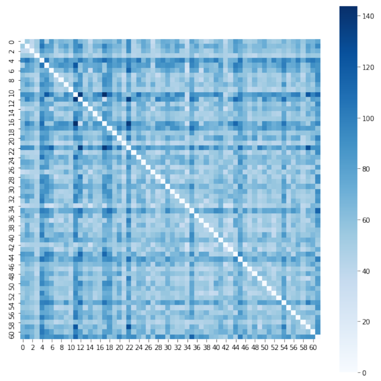
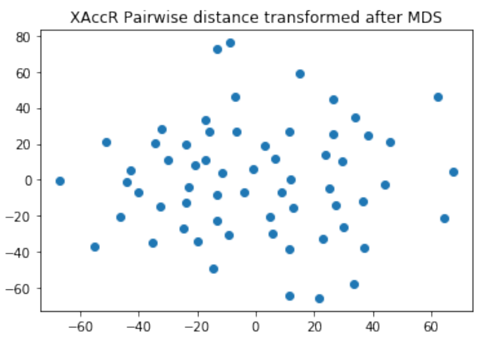
Experiment 1 with Euclidean and equal length preprocessed data is the baseline for other experiments. To compare, I also computed the accelerometer magnitude for each timestamp per patient (see Equation 1), and used it as the only feature in Experiment 2. Accelerometer magnitude, intuitively, describes the magnitude of motion of the motion sensors, and is a reflection of how active the patients' is at a timestep. Experiment 3 differs from experiment 1 only in that it uses DTW distance, and this comparison can gives us an idea of which distance metric performs better in the classification task.

$$\text{magnitude} = \sqrt{{acc_x}^2 + {acc_y}^2 + {acc_z}^2} \tag{1}$$

Experiment 4 initially was designed to use 1NN classifier with DTW distance computed from unequal length data. However, the algorithm didn't not terminate in almost 24 hours, because computing DTW distance is a very slow and resource intensive process. So I decided to precompute pairwise DTW distance between all patients for each feature and save them to file. While this process was time consuming as well, taking a little over 14 hours, this is more efficient than fitting a KNN classifier directly onto the data. After computing the pairwise distances, we get 18 (for each feature), a 62×62 matrix.

Multidimensional Scaling (MDS) [7] is a method that model a dissimilarity matrix (pairwise distance matrix) in geometric space. The implementation is from the scikit-learn package. Metric MDS is used where distance between two points in the similarity matrix corresponding to the exact distance in the embedding space. MDS was run on the square pairwise distance matrix, reducing the dimensionality of data from $N \times N$ to $N \times 2$ ($N = 62$). A visualization of both distance representation is shown in Figure 2.

After MDS transformation, each of the $N \times 2$ matrix for each feature is stacked, creating a $N \times 2P$ matrix where P is the number of features ($P = 18$). Then support vector machine classifier using the

(a) Pairwise Distance Matrix ($N \times N$)(b) MDS Transformed Pairwise Distances ($N \times 2$)

Figure 2: Pairwise Distance Representation Before and After MDS for Right Hand x-Axis Accelerometer Data

radial basis function (RBF) kernel is fitted on the new matrix, adjusting weight for class imbalance.

2.4 Deep Learning Classifiers

After running Experiment 1-4, experiment 4 produced the best result, however computing pairwise distance took more than 14 hours for 62 patients, making this method hard to scale up for larger dataset. I decided to use deep learning, given its ability to learn nonlinear transformation and flexibility to varying types of data. The augmented data as described in previous section was used for both deep learning classifiers. The augmented data (2585 samples) are split in to training, validation and test set (60/20/20).

A single layer LSTM (with hidden state of size 128), followed by two linear layers with ReLU activation was used to establish a deep learning baseline. The learning rate is $8e-4$ and the model was trained for 25 epochs. While it was able to produce satisfiable results, because of its slow training speed, I decided to try the separable convolution neural network as well.

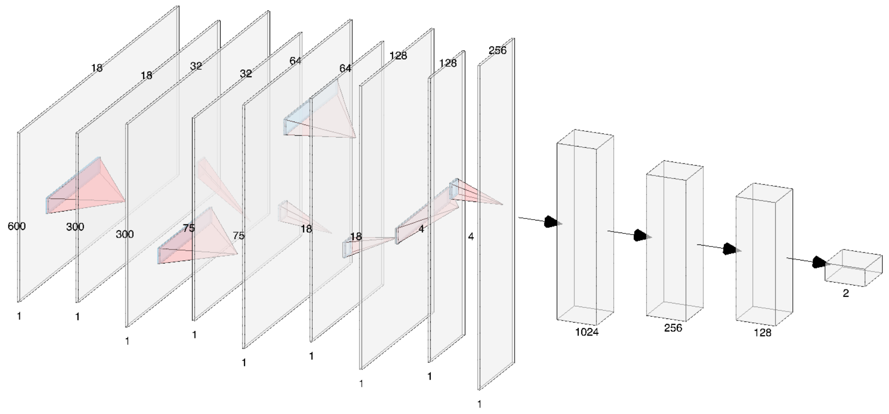


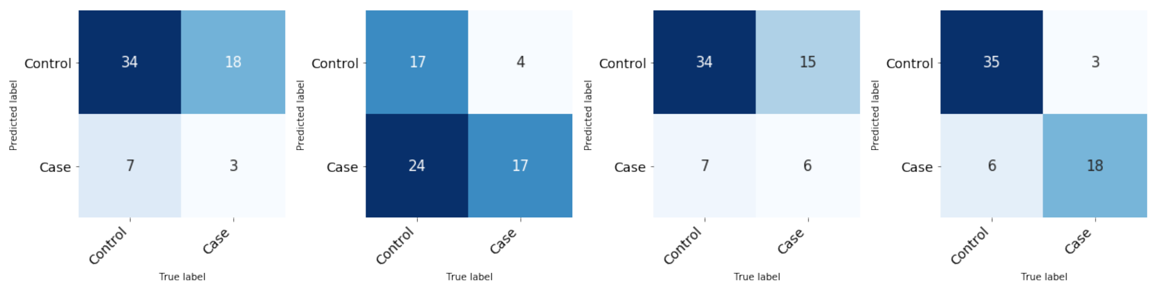
Figure 3: SepConv Architecture

The separable convolution neural network (SepConv) uses convolutional layers similar to convolutional neural networks (CNN), but differs in that it performs depth-wise convolution, unlike the most common square kernel convolution. This architecture was first proposed in MobileNet [8], a lightweight deep neural network, to reduce the number of parameters and the number of times we transform the input compared to normal convolution, effectively saving computational power. Depth-wise and point-wise convolution, changes the dimension of the input separately, length/width wide and channel wise respectively, and is often used in image convolution tasks. This however, can also be applied on time series data, where applying non-linear transformation using separable convolution layers can help extract higher level features, and achieve a more compact representation of the time series data for classification.

The model architecture I used is shown in Figure 3. The input, output size for each layer are also marked. The input to the neural network is of size batch $\times$ 18 $\times$ 600. There are four separable convolution layers, each including a depth-wise convolution with a kernel of size 8×1, and a point-wise convolution with a kernel of size 1×1. While the name can be a bit confusing, here depth-wise convolution reduces input length (length of time series), and point-wise convolution increases input width (input feature space). The batch size is 32, learning rate is $3e-4$ and the model was trained for 45 epochs. The implementation was adapted from one on Kaggle [1]

3 Evaluation

Accuracy, sensitivity and specificity is calculated, and confusion matrices are plotted for each model. The running time is also recorded. All evaluation results can be found in Table 4. For all distance based machine learning models (Exp 1-4 in Table 3), given the limited samples of 62 patients, all experiments are evaluated using 4-fold cross validation. The confusion matrices for Exp 1-4 are shown in Figure 4.



(a) Model 1 (b) Model 2 (c) Model 3 (d) Model 4

Figure 4: Confusion Matrices for Distance Based Machine Learning Models

For the two deep learning models (Exp 5-6 in Table 3), the models are trained on the training set ($n = 1551$), and hyperparameters are tuned based on both performance on the training and validation set ($n = 517$). Both accuracy and loss for each epoch is plotted, shown in Figure 6. After tuning hyperparameters, the models are evaluated using the test set ($n = 517$). Confusion matrices are plotted for all training, validation and testing set for both deep learning models, shown in Figure 5. An ROC curve is plotted for the both models on the test set, shown in 7.

[1] https://www.kaggle.com/purplejester/pytorch-deep-time-series-classification

Exp	Accuracy	Sensitivity	Specificity	Time(s)	# Parameters[#]
1	59.677%	14.286%	82.927%	1.15	/
2	54.849%	80.953%	41.463%	1717.03	/
3	64.516%	28.571%	82.927%	7.19	/
4	85.484%	85.714%	85.366%	0.01[*]	/
5[+]	80.851%	67.729%	93.233%	2242.73	92546
6[+]	**92.070%**	**89.305%**	**93.636%**	**161.18**	**349924**

[#] The number of trainable parameters in the neural network
[*] Pairwise distance for features took 14 hours to compute.
[+] Evaluation metrics reported in this table for model 5 and 6 are computed on the test set ($n = 517$).

Table 4: Model Results

3.1 Distance-Based Machine Learning Classifiers

Among the non deep learning methods, from Exp 1 through 4, Exp 4 using MDS on pairwise distance and SVM classifier is the best model performance wise.

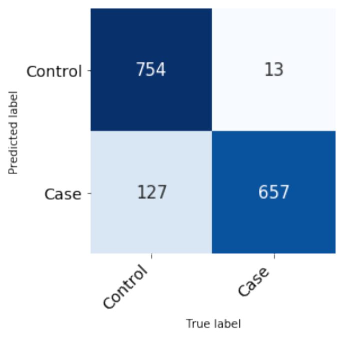
Exp 1-3, while they have similar accuracy, fail to consistently classify for both case and control groups. From the confusion matrices (a) through (c) in Figure 4, and sensitivity and specificity results on Table 4, we can see that Exp 1-3 performs only good on either case or control group, but not both. Exp 1-3 all uses a non-parametric 1NN model, where classification results depend largely on the dataset and is especially limiting when dealing with smaller sample size, which is our case here. 1NN also fails to take into consideration the class imbalance issue. Exp 1 and 3, where all 18 features are used, predict more samples as the control group, which is the majority class. Exp 2, which uses accelerometer magnitude as its only feature, interestingly has a much high sensitivity score than specificity, meaning that its performance is better for the case group than control group. This shows that the accelerometer magnitude could potentially be a good indicator of stroke history in patients.

Exp 3 has a higher sensitivity score compared to Exp 2, showing that DTW distance can improve the model performance for the case group, which led to Exp 4. Exp 4, similar to Exp 3, also uses DTW distance as a metric, but instead of relying on the closest neighbor, MDS reduces the dimensionality of the pairwise distances, and reduces the number of features fed into the classifier. SVM is better at handling outliers than KNN, can learn nonlinear solutions using the radial basis function (RBF) kernel and has better performance when we have fewer training data. In addition, class imbalance was also adjusted for in the SVM classifier. Despite its high accuracy score, Exp 4 requires all pairwise DTW distances be precomputed, which is extremely slow to compute, making this pipeline hard to scale up for larger datasets.

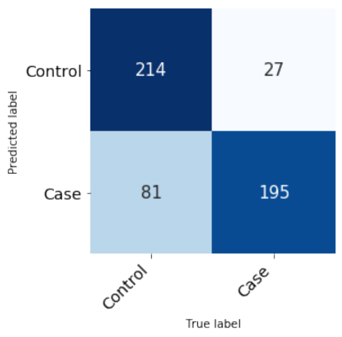
3.2 Deep Learning Classifiers

Both 2 deep learning models performs better than Exp 4, considering that each sample in the deep learning model is only 600 seconds long, compared to the original length of the data. From the confusion matrices in Figure 5, we can see that both models are more robust, performing well on all train, validation and test set. All accuracy, sensitivity and specificity for train, validation and test set are recorded in Table 5.

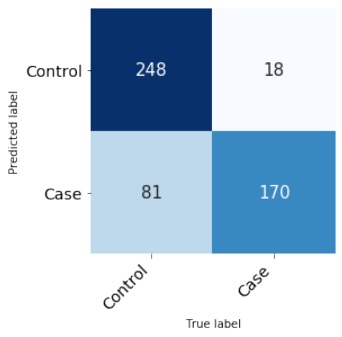
The LSTM model has fewer parameters, but it takes a long time to train. It has better sensitivity



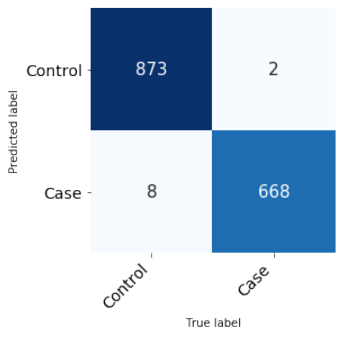
(a) LSTM Train



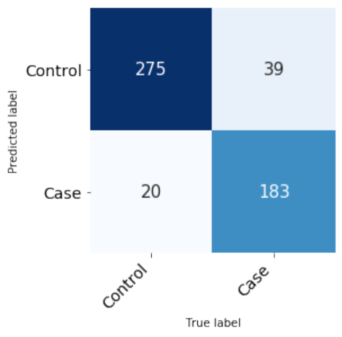
(b) LSTM Validation



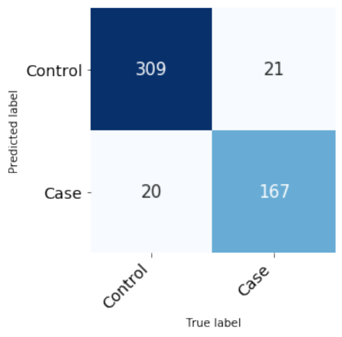
(c) LSTM Test



(d) SepConv Train



(e) SepConv Validation



(f) SepConv Test

Figure 5: Confusion Matrices for Deep Learning Methods

than specificity score on all train, validation and test set, whereas SepConv has higher specificity score than sensitivity score. Looking at numerical values, SepConv is achieves a more balanced result on both case and control group, and higher accuracy on all sets of data. While SepConv have more parameters, it is much faster, making it the best performing model among all 6 models.

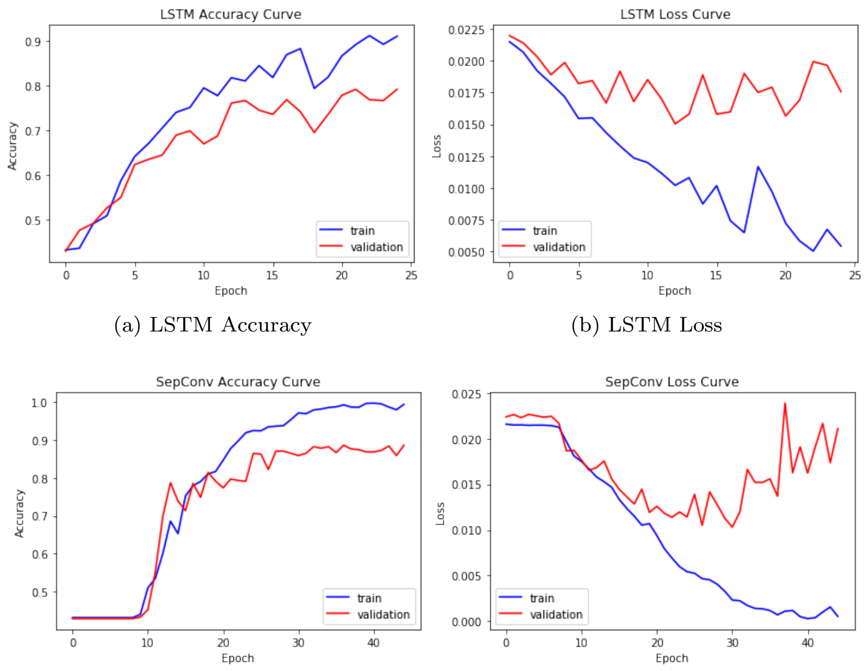
One thing to note is that, SepConv has a suspiciously high training accuracy despite that the model still performs well on validation and test set. The final model chosen whose results is shown in this report is one where the training accuracy and loss have converged. But if we look closely at the loss curve, see Figure 6, the validation loss is unstable, and increases significantly after epoch 30 although it's accuracy seems to have converged. I've tried tuning the model with a very small learning rate, and early stops the mode training before validation loss deviate from the training loss, and this produces a relatively poor performance, leading to $< 70\%$ specificity and $> 90\%$ sensitivity on the validation set. The model instability could be due to that the data comes from a small distribution, or the lack of penalty and regularization. More experiments with different learning rates, weight decay and dropout layer needs to be done to further improve model stability.

4 Conclusion & Discussion

In this project, I experimented mainly with 6 different models, including 4 distance based machine learning and 2 deep learning models, paired with a specific data preprocessor. Applying multidi-

Models		# Samples	Accuracy	Sensitivity	Specificity
LSTM	Train	1551	90.974%	98.060%	85.585%
	Validation	517	79.110%	87.838%	72.542%
	Test	517	80.851%	90.426%	75.380%
SepConv	Train	1551	99.355%	99.701%	99.092%
	Validation	517	88.588%	82.432%	93.220%
	Test	517	92.070%	88.830%	93.921%

Table 5: Evaluation Results for Deep Learning Models



(a) LSTM Accuracy (b) LSTM Loss

(c) SepConv Accuracy (d) SepConv Loss

Figure 6: Learning Curves for Deep Learning Methods

mensional scaling (MDS) on pairwise DTW distances, and using an support vector machine (SVM) classifier yields the best results among the non deep learning models. However, its slow running time points to the need to use deep neural network to efficiently extract high level information for classification. Overall, the Separable Convolution model on augmented data performs the best, in terms of accuracy and running time. Depth-wise convolution provides a efficient and effective method to transform multivariate time series and extract a more compact feature representation for classification. Its ability to accurately classify patient based on 10 minutes of motion sensor data can potentially be used in online learning with streaming data. Validation has yet to be completed on a larger dataset to test the robustness and stability of this model.

Some other future steps are listed below:

1. Extract one label per participant for deep learning models, instead of per sample (potentially use a voting classifier);

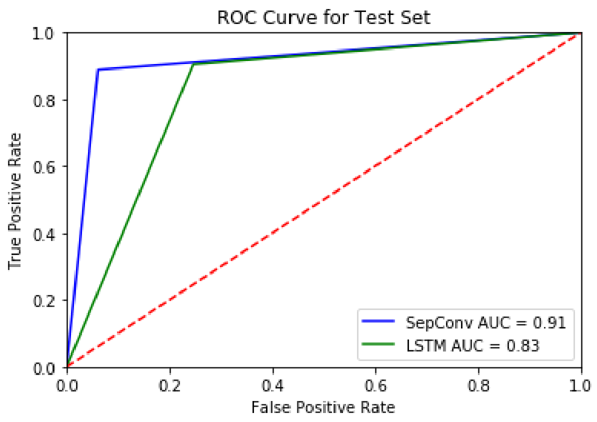


Figure 7: ROC Curve for Test Set

2. Validate the SepConv model on more data;
3. Calibrate motion sensor data before preprocessing to account for sensor inaccuracies;
4. Extract intervals with anomaly instead of random sampling.

Acknowledgement

This project is advised by Dr. Insup Lee, Dr. James Weimer, Dr. Ivan Ruchkin and Dr. Steven R. Messe.

References

[1] E. Zarahn, L. Alon, S. Ryan, R. Lazar, M.-S. Vry, C. Weiller, R. Marshall, and J. Krakauer, "Prediction of motor recovery using initial impairment and fmri 48 h poststroke," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 21, pp. 2712–21, 04 2011.

[2] J. A. Semrau, T. M. Herter, S. H. Scott, and S. P. Dukelow, "Examining differences in patterns of sensory and motor recovery after stroke with robotics," *Stroke*, vol. 46, no. 12, pp. 3459–3469, 2015.

[3] C. Lowrey, C. Jackson, S. Bagg, S. Dukelow, and S. Scott, "A novel robotic task for assessing impairments in bimanual coordination post-stroke," *International Journal of Physical Medicine and Rehabilitation*, vol. S3:002, 02 2014.

[4] Y.-W. Cheng, P.-Y. Chen, C.-Y. Yang, and H. Samani, "Imu based activity detection for post mini-stroke healthcare," in *2016 International Conference on System Science and Engineering (ICSSE)*, 2016 International Conference on System Science and Engineering (ICSSE), IEEE.

[5] Y. Chen, C. Miao, X. Yang, B. Chen, C. Leung, and H. Yu, "Using motor patterns for stroke detection," 2015.

[6] C. W. Tan, F. Petitjean, E. Keogh, and G. Webb, "Time series classification for varying length series," 10 2019.

[7] W. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, pp. 401–419, 02 1952.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.