Motion-Based Handwriting Recognition

Junshen Kevin Chen, Wanze Xie, Yutong He | {jkc1,wanzexie,kellyyhe}[at]stanford.edu

Overview

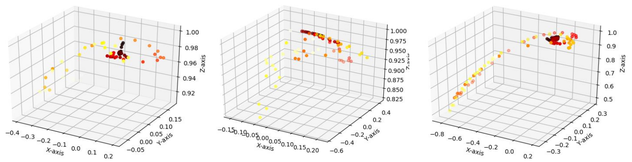
This project attempts **motion-based handwriting recognition** to explore an alternative to a vision-based approach, for various use cases where there is no convenient surface to write on (e.g. VR).

For this project, we build the collection **hardware** with Arduino and motion sensor, collect our own **original dataset**, then use various techniques to process data and perform **classification.**

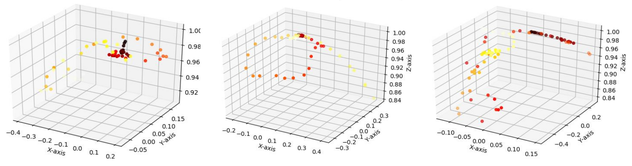
The inputs of all of our models are the yaw, pitch roll data for each time stamp, and the outputs are the predictions of the corresponding letter for each writing event.

Data

We have collected 10,000+ data sequences from 20+ subjects, and preprocessed the gyroscope and accelerometer data.

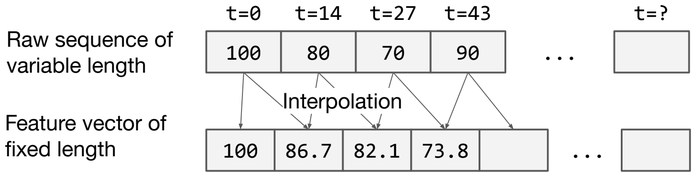


Letter 'a' written by 3 subjects

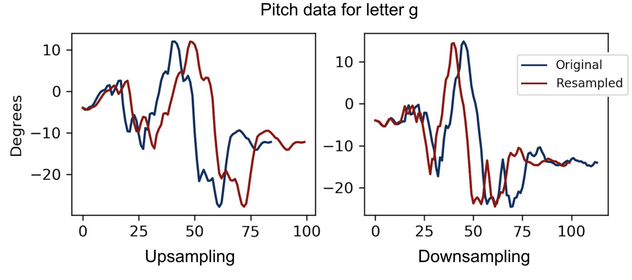


Letter 'a', 'o', 'u' written by the same subject

Feature Interpolation & Re-sampling

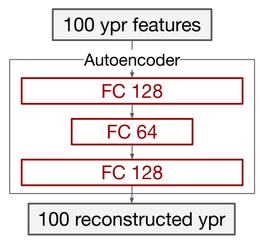


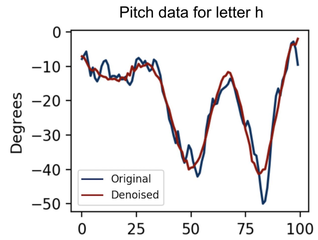
Since the sensor samples at a **variable rate**, and people write in **different speed**, we interpolate each sequence based on time delta, to a fixed length for models that require a fixed-sized input.



Denoising Autoencoder

We explore **autoencoder** to remove noise from the raw data. After efficiently learned the latent representation of the data, the model is used to preprocess each input for the classifiers before training.





Data Augmentation

Due to the difficulty to obtain a large dataset, we augment our dataset by: 1. **stretching** by a scalar for all timestamps; 2. adding a Gaussian **noise** to each timestamp; 3. **rotating** by a small quaternion, to generate more training samples for our models.

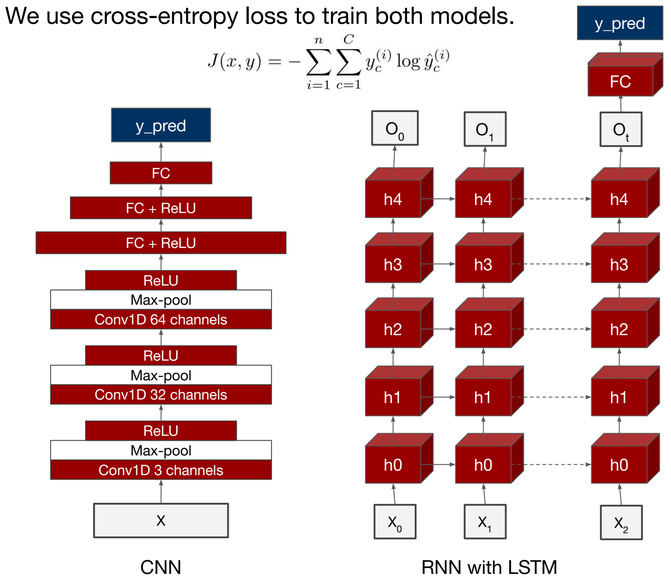
Neural Networks

We implement two neural network based models for this task.

- **CNN:** We flatten the data to a fixed length input for CNN model. We consider each feature component as an individual channel, and construct layers of 1-D convolution, with activation and pooling to construct a network.
- **RNN with LSTM:** RNN has the inherent structure for sequential data. We use a 5 layer LSTM RNN to process the inputs and a fully-connected layer applied on the last LSTM output for classification.

We use cross-entropy loss to train both models.

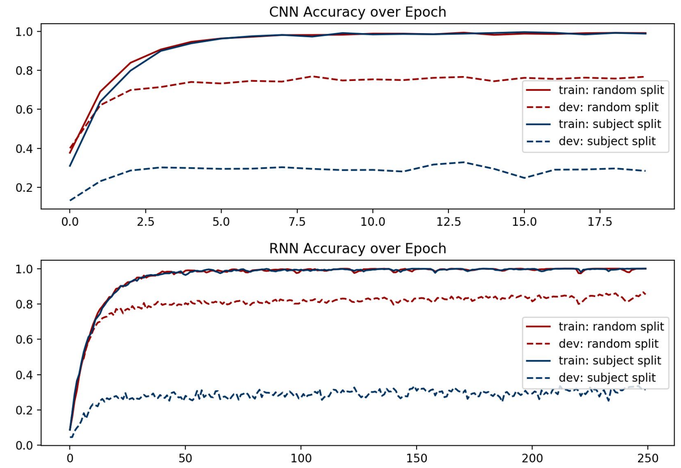
$$J(x, y) = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_c^{(i)} \log \hat{y}_c^{(i)}$$



Experiment Results

We use a K nearest neighbors classifier and a one-vs-all SVM as baseline to evaluate the performance of our various techniques.

	Random Split		Subject Split	
	Train	Test	Train	Test
KNN (K = 4)	-	0.718	-	0.128
SVM	0.753	0.591	0.999	0.156
CNN	0.985	0.710	0.977	0.358
CNN (w/ aug)	0.919	0.774	0.988	0.396
CNN (w/ aug+AE)	0.996	0.784	0.996	0.402
RNN	**1.000**	0.766	**1.000**	0.406
RNN (w/ aug)	0.999	0.843	**1.000**	0.510
RNN (w/ aug+AE)	0.999	**0.866**	0.999	**0.536**



Conclusion and Discussion

- **RNN** models achieve the **best performance** among all the algorithms we have experimented with.
- Data augmentation strategy is able to enhance the performance of all deep models.
- Denoising autoencoder is effective for handle paritialy destructed inputs.
- All models suffer from overfitting problem due to the lack of generality of the data set.

Future Work

- Collect more data from different subjects and include more diverse demographic groups.
- Incorporate unused attributes collected.
- Develop a framework for handwriting word prediction based on the current models.