Project I. - Gait Recognition

Abishek Ranjan Singh
arsingh3@ncsu.edu

Shubham Miglani
smiglan@ncsu.edu

Rajan Anbazhagan
ranbazh@ncsu.edu

I. Generative Model

A. Pre-Processing

- **Sliding window** with 50% overlap approach was followed to utilize the given data and convert it into a supervised machine learning problem.The approach followed is to reshape the information we have by fixed windows that will give the model the most complete information possible at a given time point from the recent past, in order to achieve an accurate prediction. In addition, giving previous values of the response itself as an independent variable affects the models.
- **Standardize** the features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as

$$z = (x - u)/s$$

where u is the ean of the training samples and u is the standard deviation of the training samples. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Standardization of a dataset is necessary as the model might behave badly if the individual features do not more or less look like standard normally distributed data .For instance some features with a higher variance may dominate over the model and may not let it learn characteristics from other features.

B. Data Exploration

- First we tried to segregate and visualize the features (Fig 1.) for each class label to explore how evenly the data is distributed. On visualization we were able to find there was a data imbalance related to certain class labels which werer later handled.

- Next we try to find a reasonable comparison between labelling the data while implementing sliding window technique using with either the mode of the labels or the mid value of the labels (Fig 2). From the comparison we find that both the values are almost similar and hence we decide to proceed with mode for labellilng data.

C. Model

For the model we go with a Variational Auto Encoders. We will use a simple VAE architecture similar to the one described in the Keras blog.
The Encoder model has the following structure

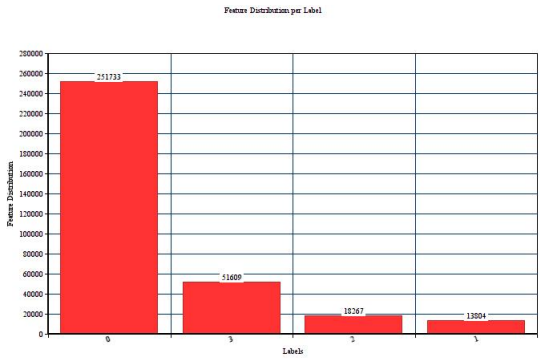

Fig. 1. Feature distribution per Label

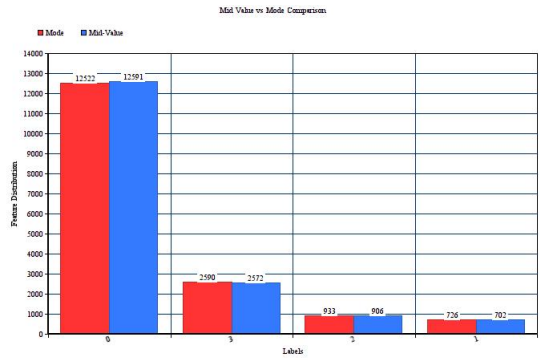

Fig. 2. Mode vs Mid- Value feature distribution comparison

Model: Encoder			
Layer (Type)	Output Shape	Param	Connected to
input-2 (InputLayer)	[(None, 960)]	0	
dense-11 (Dense)	(None, 512)	492032	input-2[0][0]
dense-12 (Dense)	(None, 256)	131328	dense-11[0][0]
dense-13 (Dense)	(None, 64)	16448	dense-12[0][0]
dense-15 (Dense)	(None 2)	130	dense-13[0][0]
dense-16 (Dense)	(None, 2)	130	dense-13[0][0]
lambda-1 (Lambda)	(None, 2)	0	dense-15[0][0] dense-16[0][0]

The decoder model has the following architecture

Model: Encoder			
Layer (Type)	Output Shape	Param	Connected to
input-2 (InputLayer)	[(None, 960)]	0	
dense-11 (Dense)	(None, 512)	492032	input-2[0][0]
dense-12 (Dense)	(None, 256)	131328	dense-11[0][0]
dense-13 (Dense)	(None, 64)	16448	dense-12[0][0]
dense-15 (Dense)	(None 2)	130	dense-13[0][0]
dense-16 (Dense)	(None, 2)	130	dense-13[0][0]

D. Hyper parameter Tuning

For hyper parameter tuning we experimented by changing the learning rate and batch size. We found that a batch size of 32 with a learning rate of 0.001 gave the best results. Also we tried modifying the simple VAE and tried using Deep VAE and Deep VAE with 1d Convoutional layers. Where we found the Deep VAE gave better results.

Hyper Parameter Tuning	
Hyper Parameter	Value
Learning Rate	0.01 **0.001** 0.0001 0.00001
Batch Size	**32** 64 128 256
Models	**Deep VAE** 1-Layer VAE Deep VAE with 1d Conv

E. Training and Validation Loss

From the hyperparameter tuning, training and validation losses for the best model was plotted as given in Fig 3.

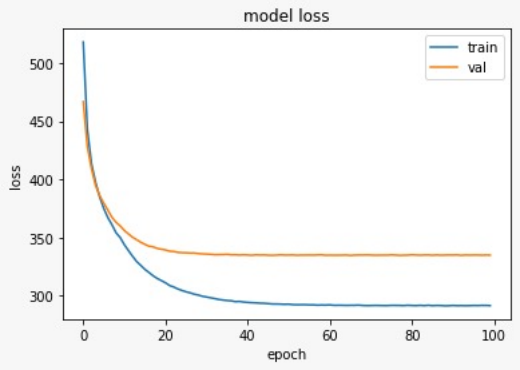

Fig. 3. Training/ Validation Loss

II. Generative Model Evaluation

A. Visualization

For 3-d latent space visualization, validation data is fed to the encoder and the output means are plotted.

The plot for latent variable visualization is displayed in Figure 4. This plot contains all the classes. Some structure can be inferred regarding differences between class 1,2 vs class 0,3 from here but for ease, more plots are generated as in the following figures.

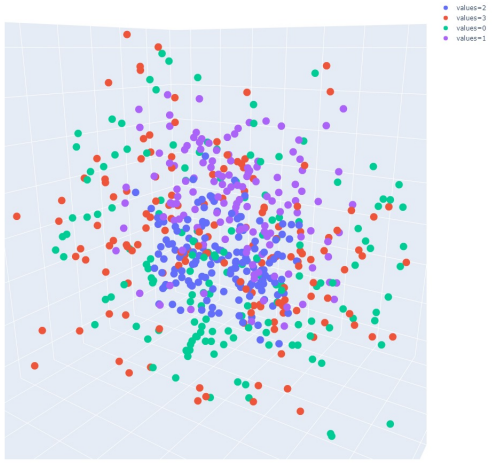

Fig. 4. Data Visualization 1

As can be seen in Figure 5, class 1 and 2 can be seen to be separating into clusters with slight overlap which will make it easier for classification. The overlap can be due to the data prepossessing or outliers present in the data. These classes represent walking upstairs and downstairs and through visualization, it's clear that they can be separated.

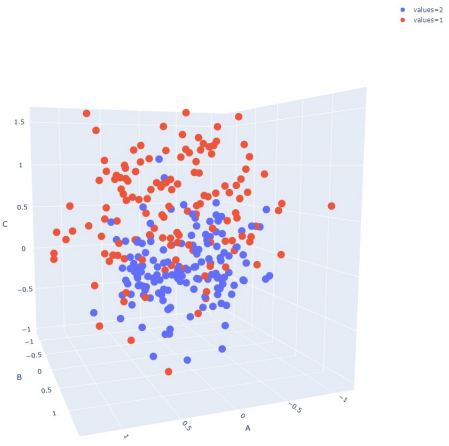

Fig. 5. Data Visualization 2

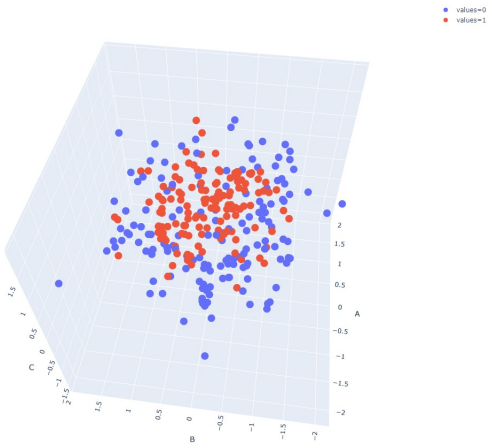

Fig. 6. Data Visualization 3

In Figure 6. class 0 and 2 are plotted, class 2 can be seen in a cluster with a smaller radius in the center where the class 0 is more spread out. Even these seem separable in 3 dimensions. Similar plot was observed for class 0 and 1 as well.

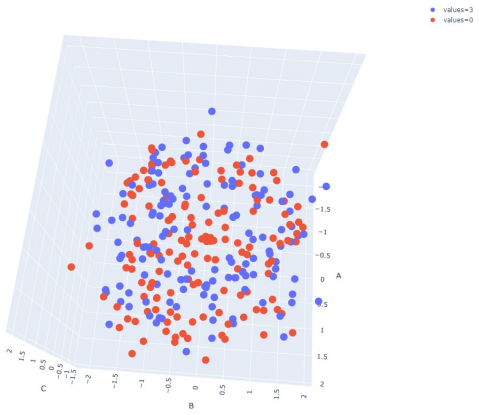

Fig. 7. Data Visualization 4

In Figure 7. class 0 and 3 are plotted. These have a lot of overlap between them and does not seem easily separable. Class 0 and 3 represents walking on solid ground and walking on grass. These activities were found to have difficulty in separating into clear clusters

B. Data Generation

The actual and generated data for all 6 sensor readings using the decoder can be seen in figure 8 and 9.

III. Predictive Model

For the predictive model, Conditional Random Fields is used. Recently, CRF's in combination with LSTM's have been

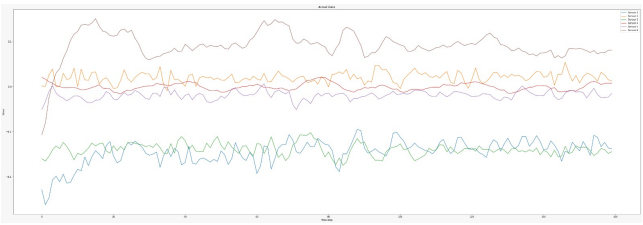

Fig. 8. Actual Data

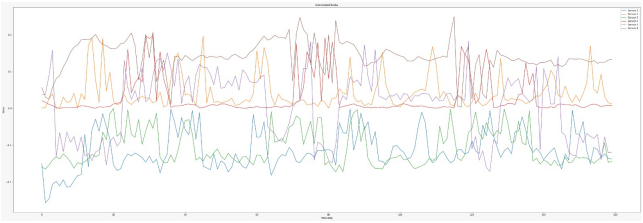

Fig. 9. Generated Data

used to improve accuracy for sequence tagging problems [4]. Similar model was used here. The output of the VAE in the previous section was taken and passed to the new model by using the encoder.

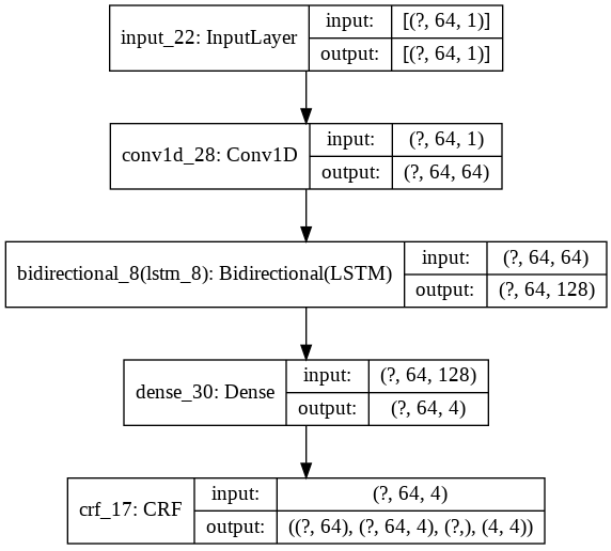

Fig. 10. Model

For supervised learning, we used a package called *tf2crf* [5] which provides a simply CRF layer to be used with keras. This package works well with the latest version of tensor-flow as well.

Latent dimension was also taken as a hyper parameter along with batch size and learning rate. The final values used are shown below.

Hyper Parameter Tuning for CRF	
Hyper Parameter	Value
Learning Rate	0.01 **0.001** 0.0001 0.00001
Batch Size	**32** 64 128 256
Latent Dimensions	16 32 **64** 128

Class weights were used because of the class imbalance problem. The ratio of each class was taken with respect to class 1 (with minimum labels in our data set) and passed as an argument to the model.fit function.

Even after using class weights, there were problems with class-3 predictions as it's F-1 score was lower as compared to other classes possibly due to various reasons such as similarity between class 0 and class 3 data or due to class 0 labels being higher in value. To solve this problem, data down-sampling was used. The final data distribution before and after down-sampling is shown as below.

Data Distribution			
W/O downsampling			
Class	Input Data	Training	Validation
0	12522	8716	3806
1	726	521	205
2	933	657	276
3	2590	1845	745
After downsampling			
Class	Input Data	Training	Validation
0	2590	1794	796
1	726	506	220
2	933	659	274
3	2590	1828	762

IV. Predictive Model Evaluation

The evaluation parameters for the test data are shown below in the form of error metrics, training loss and validation loss.

Error Metrics			
class	precision	recall	f1-score
0	0.69	0.72	0.71
1	0.92	0.72	0.81
2	0.88	0.90	0.89
3	0.73	0.75	0.74
accuracy			0.75
macro avg	0.81	0.77	0.79
weighted avg	0.76	0.75	0.75

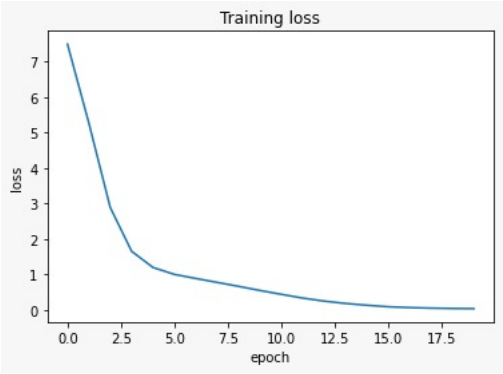

Fig. 11. Training Loss for Prediction model

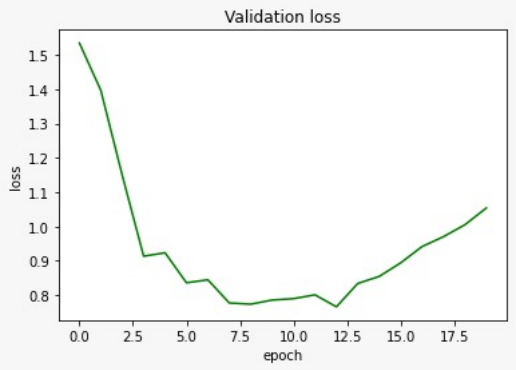

Fig. 12. Validation Loss for Prediction model

For getting the prediction results, for each individual file, the predicted value for 960 samples, we required 40 values. So, each prediction was repeated 40 times and then every alternate prediction was taken to remove the overlap effect that we done during data processing. To match the shape, the end values were appended with all zeros

For the predictions submitted on the Leaderboard, a variety of models were saved and their probabilities were combined to get the final output. As mentioned earlier in Hyper Parameter tuning for VAE, the VAE structure was varied. Each of these 3 models were trained with and without the down-sampled data. All of their probabilities were combined to get the final prediction.

In conclusion, the generative model combined with CRF did a decent job at both generating signals and prediction. The highest F1-score for our model on the leaderboad was **0.698**.

The downside is similar performance for the same data was observed using a simple one layer neural network or other algorithms such as Random Forest and XGBoost without using a generative model and directly performing classification on the dataset. The benefit of using a generative model for classification requires more analysis in the future. Also, the package *tf2crf* which was used has another model which helps to improve f1 score with unbalanced data. It could be tried as well to improve performance.

The data distribution results on the test set for different subjects are shown as below.

Results on Test Set		
Test Subject	Class	Value
Subject 9	0	6257
	2	160
	3	3080
Subject 10	0	6229
	1	680
	2	420
	3	4940
Subject 11	0	8079
	1	680
	2	840
	3	3340
Subject 12	0	8989
	1	680
	2	880
	3	780

REFERENCES

[1] https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/

[2] Alireza Abedin Varamin, Ehsan Abbasnejad, Qinfeng Shi,Hamid Rezatofighi and Damith Ranasinghe. "Deep Auto-Set: A Deep Auto-Encoder-Set Networkfor Activity Recognition Using Wearables"

[3] https://github.com/deadskull7/Human-Activity-Recognition-with-Neural-Network-using-Gyroscopic-and-Accelerometer-variables

[4] Zhiheng Huang, Wei Xu, Kai Yu. "Bidirectional LSTM-CRF Models for Sequence Tagging"

[5] https://pypi.org/project/tf2crf/