Final Presentation Group DAW

09.03.2023

Daniel Dieterle
Anton Huber
Willi Schiebelbein

Agenda

1. Motivation & Problem
2. Dataset
3. Preprocessing & Feature Extraction
4. Deep Learning Solution
5. Experiments
6. Results
7. Conclusion

Motivation & Problem

Motivation & Problem

- Find a way to monitor activity levels with wearable sensors
- Recognize and categorize what kind of activity was done
- Draw meaningful conclusions about whether the user has an active lifestyle
- Provide direct feedback to the user about their activity levels to encourage a more active and healthier lifestyle

Dataset

Dataset

- Activity recognition with healthy older people using a batteryless wearable sensor[1]
- 14 Healthy older people (66 – 86 years old) wearing 3 batteryless sensors on top of their clothing
- 75128 instances total
- Activities performed:

  Sit on bed, Sit on chair, Lying, Ambulating

Sensor enabled wearable RFID technology for mitigating the risk of falls near beds [1]

Dataset attributes

Attribute	Description
Time in seconds	Time the signal was measured
Acceleration	Acceleration in three different axes (frontal, vertical, lateral)
Id of antenna	The id of the antenna reading the signal
RSSI	Received Signal Strength Indicator
Phase	Phase of the signal received
Frequency	Frequency of the signal received
Activity	The label of the activity performed in that moment

Preprocessing & Feature Extraction

Preprocessing & Feature Extraction

- Normalization: Bring values in a specified range [0,1]

  → Ensures values are on a similar scale (Improves performance and training stability)
- Relabeling: Previous experiments limited to active and inactive, now use all 4 classes
- Feature Removal: Drop time in seconds since measurements were not evenly spaced

Preprocessing & Feature Extraction

- Interpolation: Pad dataset with artifical samples to have equal sampling times across all measurements

  → Smallest sampling interval is 0.025 seconds

  → Pads samples so that entire dataset has sampling interval of 0.025 seconds

  → Evenly spaced samples make more sense in the context of a RNN

Preprocessing & Feature Extraction

- Minority Oversampling (SMOTE): Useful for simple classification, not feasible for time-series

→ Classification is time dependent and smote would infringe on that by adding random steps

Preprocessing & Feature Extraction

Additional features:

- Trunk rotational motion (θ)
- Coronal rotational motion (α)
- Transverse rotational motion (β)

$$\theta \approx \tan^{-1}(a_f / \sqrt{a_v^2 + a_l^2})$$

$$\alpha \approx \tan^{-1}(a_l / a_v)$$

$$\beta \approx \tan^{-1}(a_l / a_f).$$

Sequence Learning with Passive RFID Sensors for Real-Time Bed-Egress Recognition in Older People [2]

Preprocessing & Feature Extraction

Additional features:

- Energy Frontal Axis (Squared Sum, Min, Max, Mean)
- Energy Vertical Axis (Squared Sum, Min, Max, Mean)
- Energy Lateral Axis (Squared Sum, Min, Max, Mean)

→ **Common Goal:** Add more context for classification!

Deep Learning Solution

Deep Learning Solution

- Previous experiments limited to simple classification
- Dataset has a time-context that we want to include

  → **Use a recurrent neural network (RNN)**

Deep Learning Solution

Sliding Window Principle:

- For every person we predict an activity every **n** time instances
- Use the previous **i** instances for context
- Drop the last instances that don’t fit into the window

Deep Learning Solution

Model:

- RNN/LSTM/GRU-Layer with two layers with 10 hidden features each
- Fully connected Linear Classifier to get Class predictions
- Apply Argmax to get most likely prediction

Experiments

Experiments

Sliding Window Principle:

- Try different window and hop sizes to see how much context is needed for a good prediction
- Window sizes tested: 0.5 sec, 1 sec, 2 sec
- Hop sizes tested: one tenth and one fifth of window size

Experiments

Additional Features:

- Try every combination of adding/not adding the acceleration and frequency domain features
- Check which features are most beneficial

Experiments

Models:

- Try different types of RNNs as the backbone of our model to see which performs best
- Models tested: RNN, GRU, LSTM

Results

Results – Window sizes

Window size	0.5 seconds (200 samples)		1 second (400 samples)		2 seconds (800 samples)	
Hop size	0.05 sec (20 s.)	0.1 sec (40 s.)	0.1 sec (40 s.)	0.2 sec (80 s.)	0.2 sec (80 s.)	0.4 sec (160 s.)
Accuracy	0.868	0.861	0.841	0.837	0.827	0.798
Balanced Accuracy	0.766	0.753	0.743	0.726	0.720	0.708
F1-score (macro)	0.775	0.769	0.752	0.734	0.730	0.708
F1-score (weighted)	0.865	0.857	0.838	0.833	0.820	0.796

Results – Window sizes

- The best window size to use is 0.5 sec with a hop size of 0.05 sec
- The trend shows that a bigger window size generally performs worse

Results - Features

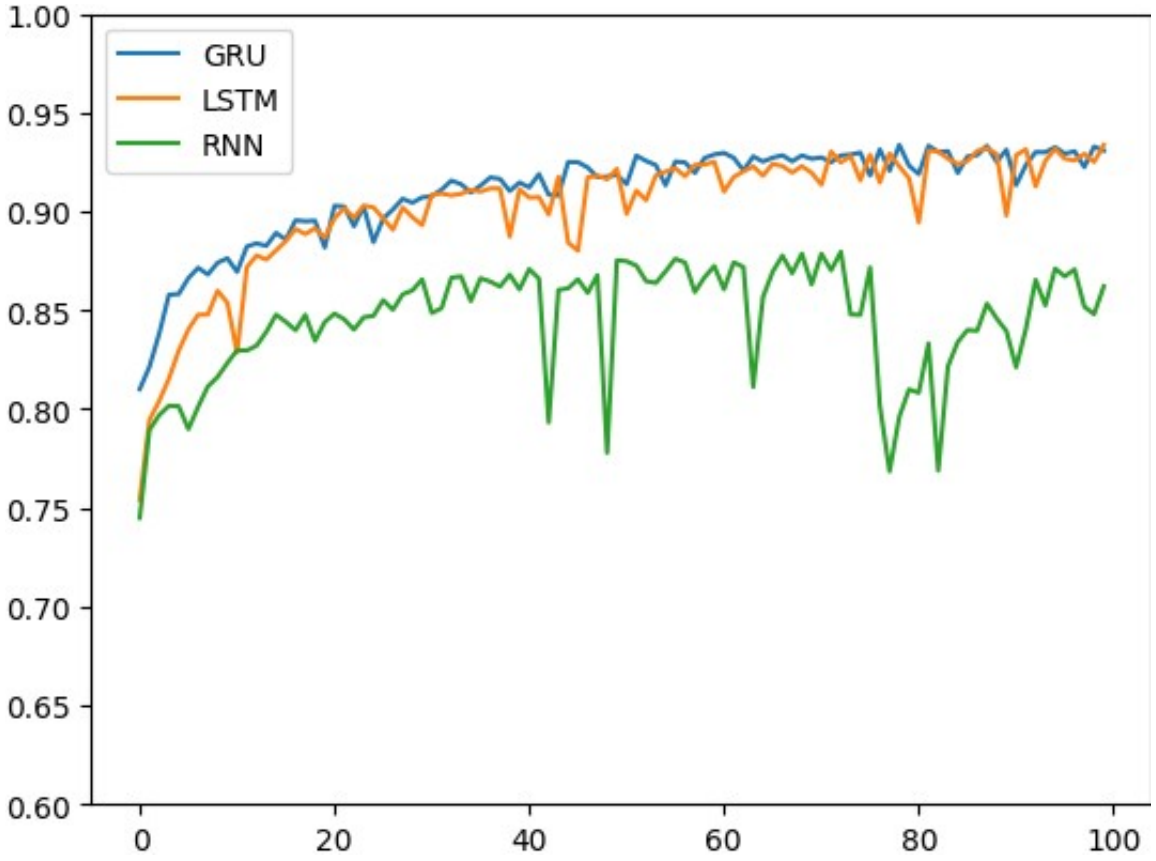
Features used	Default	Acc.	Freq.	Acc. + Freq.
Accuracy	0.872	0.863	0.698	0.691
Bal. Accuracy	0.759	0.771	0.526	0.506
F1-score (macro)	0.77	0.774	0.522	0.49
F1-score (weighted)	0.865	0.861	0.69	0.682

Results - Features

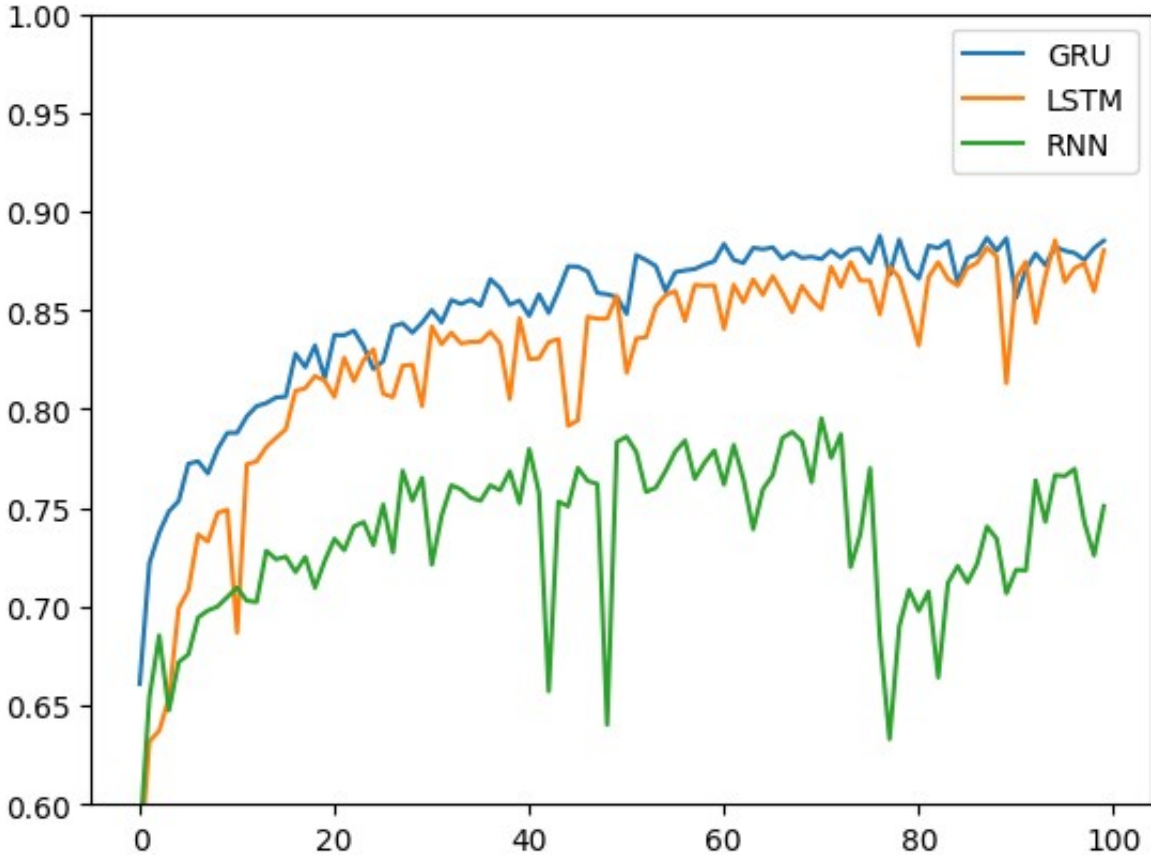
- Only adding the acceleration features scored the best balanced accuracy and macro F1-score
- Balanced accuracy and macro F1-Score are the most important metrics

Final Results

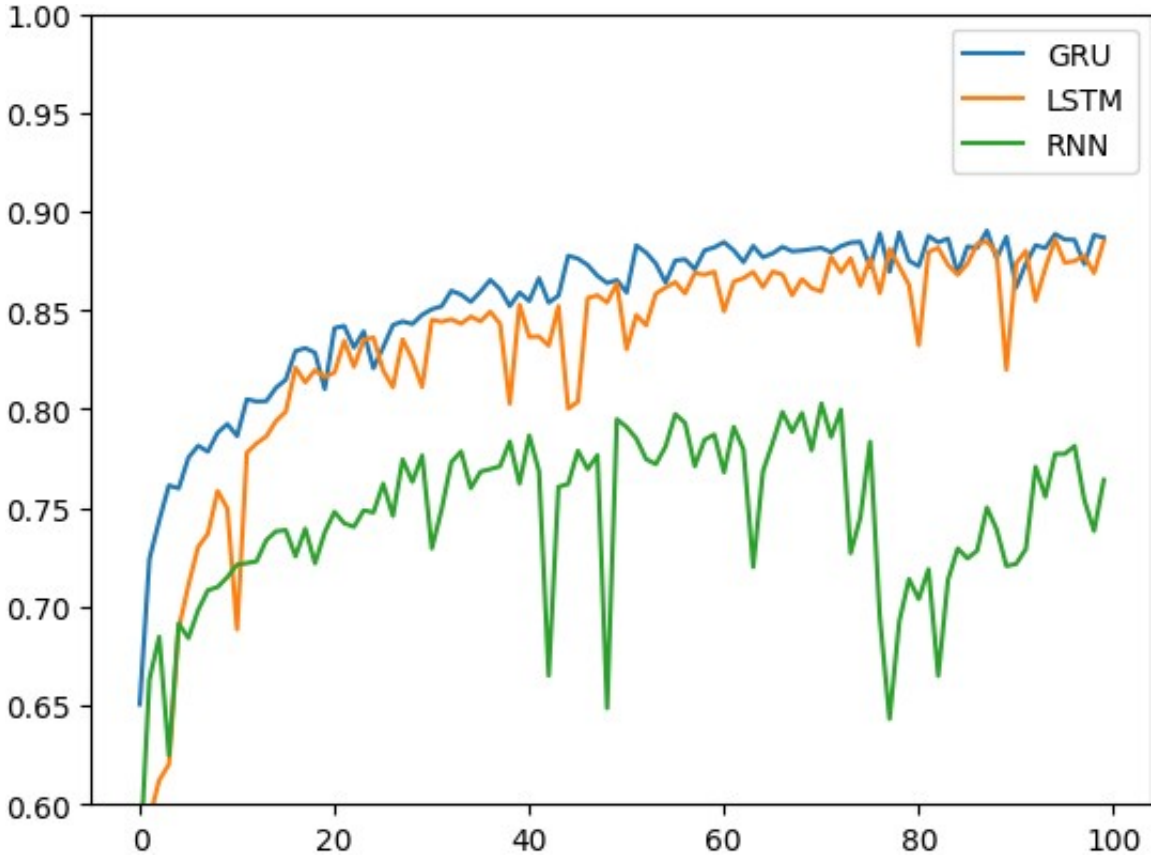
Accuracy



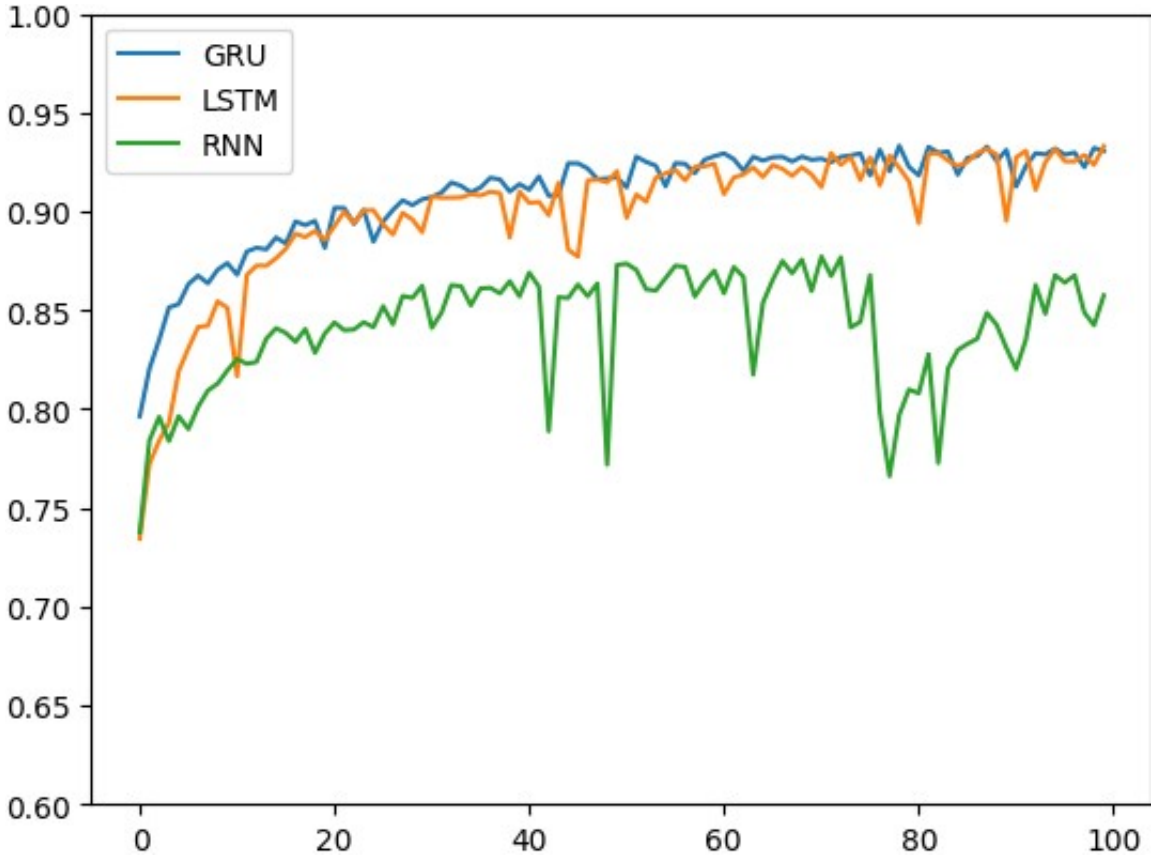
Balanced Accuracy



F1-Score (macro)



F1-Score (weighted)



Results

- LSTM and GRU outperformed RNN
- LSTM and GRU performed comparably good

  → GRU was slightly better and is more lightweight

Conclusion

Conclusion

- The Deep Learning solution performed well, but classical classifiers still outperform the neural networks
- Theory: Not enough high quality data for the network, especially interpolation might have introduced a lot of useless data
- The results were good on both approaches regardless, but could be improved by introducing more high quality data

Thank you

Appendix

Model	Accuracy	Balanced Accuracy	F1 Score
LGBMClassifier	0.985	0.925	0.985
BaggingClassifier	0.985	0.924	0.985
RandomForestClassifier	0.986	0.921	0.986
ExtraTreesClassifier	0.987	0.919	0.986
DecisionTreeClassifier	0.981	0.917	0.98
KNeighborsClassifier	0.981	0.909	0.98
ExtraTreeClassifier	0.974	0.899	0.974
SVC	0.969	0.821	0.965
QuadraticDiscriminantAnalysis	0.953	0.804	0.951
GaussianNB	0.941	0.736	0.936
AdaBoostClassifier	0.714	0.701	0.738
NearestCentroid	0.847	0.675	0.86
LogisticRegression	0.923	0.641	0.91
CalibratedClassifierCV	0.922	0.638	0.909
Perceptron	0.849	0.59	0.855
SGDClassifier	0.909	0.581	0.881
LinearSVC	0.911	0.566	0.886
LinearDiscriminantAnalysis	0.91	0.565	0.887
PassiveAggressiveClassifier	0.904	0.559	0.888
BernoulliNB	0.883	0.544	0.865
RidgeClassifier	0.899	0.492	0.86
RidgeClassifierCV	0.899	0.492	0.86
DummyClassifier	0.689	0.25	0.562