Practical Machine Learning - Project 1

- Smartphone User Identification Kaggle Competition, 2022-2023 -

Ionescu Andrei, 407
andreei.ionescu@gmail.com

Abstract

The Smartphone User Identification Kaggle Competition (Radu, 2022) consists in a classification challenge to discriminate between 20 users, on a given dataset of mobile signal recordings. The works presented in this paper highlight the full end-to-end pipeline of: data analysis and preprocessing, augmentation techniques, hyperparameter tuning and showcasing multiple approaches using both classical models such as SVM and KNN as well as deep neural networks variants: Temporal Convolutional Neural Network (TCNN), Attention-based Temporal Convolutional Neural Network (ATCNN) and Hybrid AutoEncoder-based SVM Classifer (HAESVM).

1 Dataset

1.1 Dataset Description

The task at hand is to classify the source (user) of mobile signal recordings that are taken at 100 Hz frequency, using an accelerometer, for an approximate duration of 1.5 *seconds*. For each user a fixed amount of 450 *recordings* are given, totaling up to 9.000 *samples* that have associated labels and can be used for training. On the other hand, the evaluation is done on a test set which has 5.000 *unlabeled samples*. Each time step in one recording has coordinates (x, y, z), meaning that the sizes of the subsets are as follows:

- $(N_{train}, T, C) \mapsto (9.000, 150, 3)$ *labeled* training samples.
- $(N_{test}, T, C) \mapsto (5.000, 150, 3)$ *unlabeled* testing samples.

Where N represents the number of samples, S represents the temporal/sequence dimension and C reflects the number of coordinates for one time step.

1.2 Dataset Analysis

As mentioned in the Dataset Description, there are 20 users or classes where each have 450 recordings, making the training **dataset balanced** *class-wise* from the beginning. As a first step in the analysis I inspected the global characteristics of the given data coordinates in Table 1, and observed that the computed *mean* and *stardard deviation* differ by a margin for each coordinate. Moreover, according to the *standard deviation* it seems that the values are a bit spread out which requires data scaling[1.3]. Also in regards to this, it can be an indication of residing outliers in the data which will be investigated further.

[1] / [2]	x	y	z
count	1.348955e+06	1.348955e+06	1.348955e+06
mean	-2.151555e-01	4.731727e+00	8.017973e+00
std	8.352723e-01	1.205409e+00	1.060832e+00
min	-7.778163e+00	-6.853400e-01	1.212663e+00
25%	-7.314290e-01	3.876213e+00	7.340024e+00
50%	-2.184710e-01	4.532822e+00	8.043920e+00
75%	3.351880e-01	5.391742e+00	8.720282e+00
max	7.284957e+00	1.104026e+01	1.581849e+01

Table 1: Global Training Dataset Characteristics

1.2.1 Outliers Detection and Removal

Outliers present in the training data can be quite problematic as they can *destabilize* the training process by triggering big update steps in the wrong direction, leading the model parameters to a worse local minima. Considering this fact and the *std*, *min*, *max* values shown earlier, I displayed boxplots for each coordinate (x, y, z) for the **entire training dataset** and for **each class**, in order to view if this is really the case. These plots can be seen in the following figure:

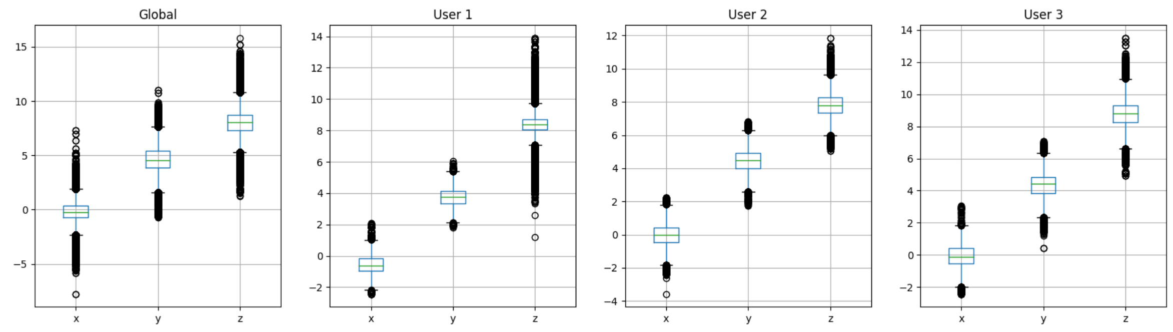

Figure 1: Training Data Outliers
Each boxplot has its whiskers set to the *Q1* or *Q3* shifted by $\pm 1.5 * \mid Q3 - Q1 \mid$. The first subplot illustrates the values for the **entire training dataset** and the rest of them are for the **first three users**. Similar values are present for other users as well but aren't included for brevity.

As it can be observed there are a large number of values residing outside the established $\pm 1.5 * IQR$ range. In order to eliminate them two approaches can be applied and the results can be seen in :

- Compute the *Q1* and *Q3* values for each (x, y, z) coordinate for the **entire dataset** and exclude those coordinates, which fall outside the whiskers as mentioned in Figure 1, from their respective recording while keeping the rest of the sampled values.
- Similarly to the previous approach, compute the *Q1* and *Q3* values for each *class* and for each (x, y, z) coordinate inside that class and proceed the same way during elimination.

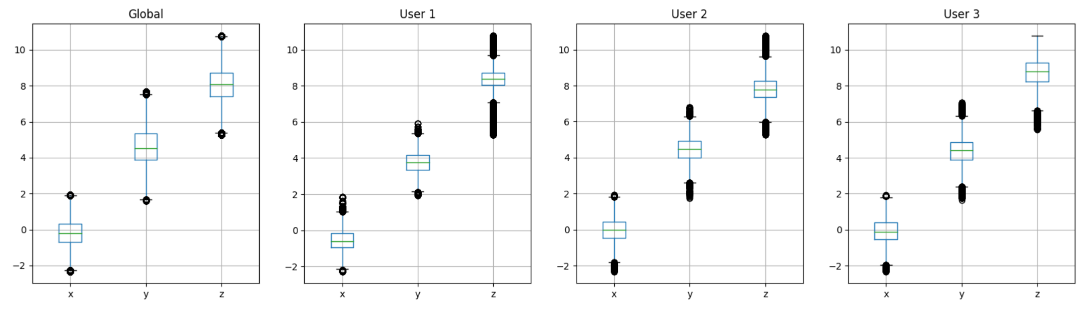

Figure 2: Global Removal of Outliers in Training Data

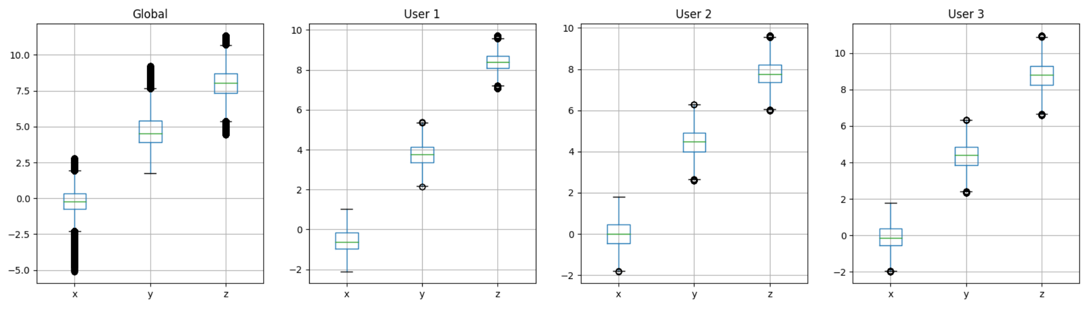

Figure 3: Intraclass Removal of Outliers in Training Data

The *intraclass* approach can only be applied on the training and validation sets and not on the test set as it does not have labeled samples. Even though the two approaches try to eliminate outliers it can be seen that when applying one method or the other there still remain some values either in the *global-scope* or the *class-scope*. Both methods could be applied simultaneously but this might result in recordings that have very few time points and the behavior of the user might not be preserved when trying to impute missing values.

During manual fine-tuning experiments, the removal methods tend to show small improvements during K-Fold Cross Validation when they are applied on both sets and slightly worse performance on the Kaggle public subset. A intuition for this would be that the task is easier to learn using the new representation but tends to overfit because the outliers might actually indicate individual spikes correlated to the user behavior.

1.2.2 Filling Missing Values

Both the training and the testing subsets contain user recordings that either **surpass** or **lack** values from the pre-established 150 sequence size. This can be an issue as many classical and deeplearning models expect a fixed input size in order to be trained and predict new values.

To overcome the first issue some entries need to be eliminated from the recordings in order to adjust an upper-bound of length 150. The values can be removed either from the start, end or inside the sequence. Because the sequence length also represents a temporal dimension which reflects the user's behavior, removing random values or intervals from inside the sequence might remove essential information. In the implementation I considered only the removal of values that are present at the **end of the temporal sequence**.

Overcoming the lack of entries is not as trivial as zero-padding or randomly filling in some statistics somewhere in the recording because this would disrupt the expected flow of information and would confuse the models between samples which needed filling and those which did not. A better approach is to consider this aspect and, in turn, interpolate new values randomly between existing ones:

1. Create an empty recording and set it's head and tail to match original recording's.
2. Insert the in-between values from the original recording into the new one, at random positions while maintaining their original ordering.
3. Fill in the gaps in the new recording by interpolating between existing non-gap values.

One thing that I considered during the proposed approach for *lack of entries* is to drop recordings that do not pass a lower-bound margin of sequence length because there would be too many interpolated values and the samples would mostly be generated and might not offer enough distinctive information to the model during the training phase. During experiments I tried to use different lower-bounds such as: 50, 75, 100. Only the 50 lower-bound value showed marginally better performance in comparison to keeping those recordings, for certain models. The results of the transformations described in this subsection can be visualized in Figure 4 and 5.

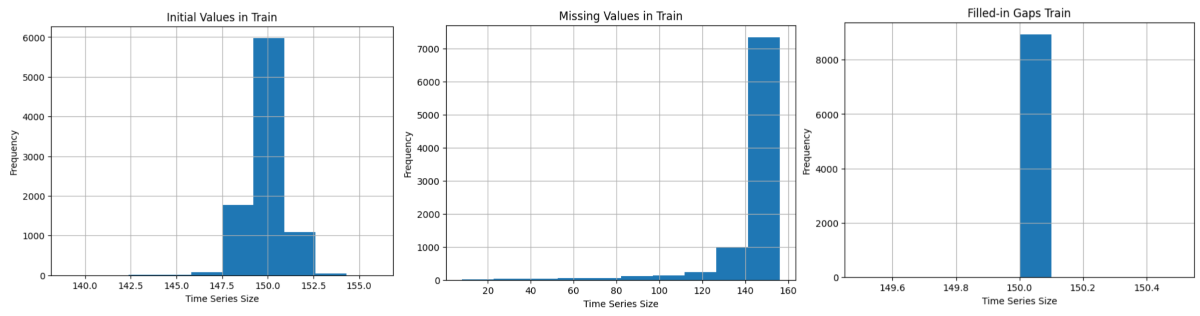

Figure 4: Before and After Histograms of Recording Sizes
The data is first capped at 150, global outliers are removed, sequences smaller than 50 are eliminated and new values are randomly in-between interpolated.

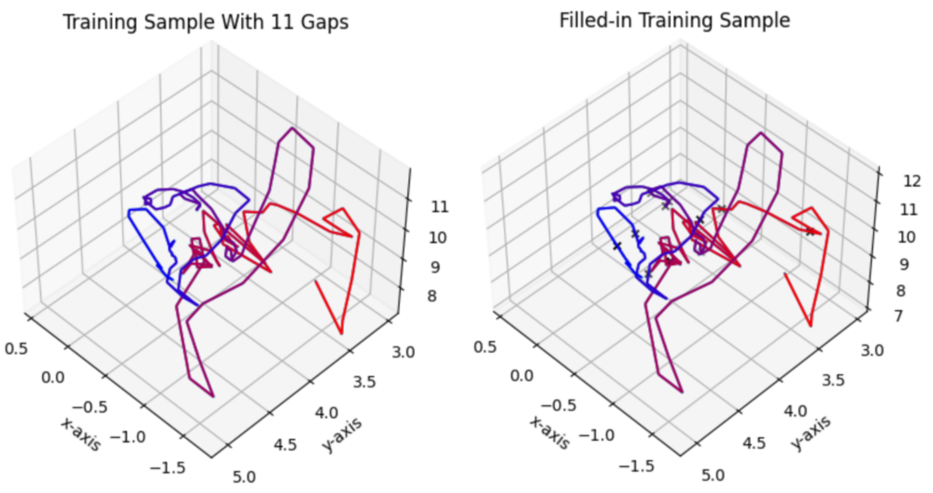

Figure 5: Gaps Filling Algorithm Applied on a Training Sample
The read-to-blue gradient indicates the start and end of the sequence. The filled-in gaps are marked by the 'x' symbol. These visualizations are generated prior to outlier elimination.

1.3 Data Preprocessing

As mentioned in the beginning, the recordings are given as a set of coordinates, $x \in \mathbb{R}^3$, that change along the temporal axis. According to the values observed in Table 1, the scaling differs from one coordinate to another as well as other characteristics. This can have negative impacts during model training in various ways such as:

- the training process is destabilized and convergence can take more time because of odd updates;
- models can learn biases towards features that have a greater scale, but which do not contribute much at inference time;

To alleviate these issues there are multiple ways of preprocessing the data such that the features are transformed to similar scales and have equal importance when given to the model. During experimentation I tried to apply *standardization*, *min-max scaling* and *L2 normalization* across different dimensions:

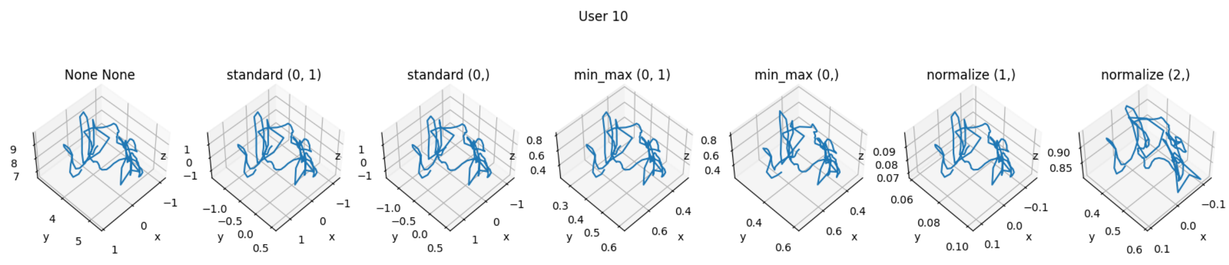

Figure 6: Data Scaling Approaches Visualized for a Training Sample
Reads from left to right. The first approach does not apply any scaling, following are the operations mentioned above. $(0, 1)$ indicates that the dataset and temporal dimensions are collapsed, while $(0,)$ reflects that only the dataset axis is collapsed. The normalization is applied per-instance across the temporal or spatial axis.

In order to investigate whether any of these known techniques improve either the training process or the accuracy, I ran each model for a small amount of epochs on the same 5-Fold Cross Validation, and recorded the mean accuracy values for each type of scaling that was applied. The data was only filled to size 150 and no recordings/outliers were removed. The results can be seen in Table ..., and showcase that *min-max scaling* and *L2 normalization* both disrupt the training process and lead to bad validation results. On the other hand, the *standardization* computed by either collapsing or not the temporal axis, shows **slighly better results** than training the models directly on the raw data. With these empirical observations in mind, the *standard* $(0,)$ approach is the one kept for further experiments.

2 Models and Tried Approaches

2.1 Support Vector Machines (SVM)

The first model that I have considered is Support Vector Machines (SVM), with the LIBSVM implementation (Chang & Lin, 2011), as the task at hand is a classification problem, with numeric data given as input features. On one hand, the model is well known for its robustness against overfitting on tasks, because of its objective of maximizing the boundary in binary classification. In order to adapt the algorithm to the current problem, which implies multiclass prediction, two approaches can be taken:

- One versus all - to train $N_{models} = 20$ by keeping one class as a positive label $(+1)$ and the other ones as negative predictions (-1);
- One versus one - by using $N_{models} = \frac{N_{class} * (N_{class} - 1)}{2} = 190$, where each one is trained on two different classes;

Futhermore, the prediction is given by a **majority vote** between the learnt models. Across the mentioned procedures, the *one versus all* approach showed better results, on average, using 5-Fold Cross Validation, while requiring significantly less time to train. Moreover, using a LinearSVM on the dataset has not shown great results compared to using the *Radial Basis Function Kernel* (Shashua, 2009) as a similarity measure, with the features being mapped in a higher dimensional space. In turn, this gives a strong indication that the data may not be linearly separable. The results of the hyperparameter fine-tuning can be seen in Table 2.

Overall, the fine-tuned classical model offers a good starting point with *RBF* kernel and the tweaked *C* regularization, which allows a softer margin to generalize better on the validation data. However, one weakness of the SVM model applied on this dataset is the fact that it **ignores the temporal dimension**. In other words, shuffling alongside the temporal axis gives the same results as each space-time point has an associated weight that receives no indication of any positional value. In order to overcome this issue, one approach would be to add *positional encoding* (Vaswani et al., 2017) in the form of:

$$X_p = \{\, x_{ij} + p_j \mid p_j = \frac{j}{T},\ j \in [1, T] \cap \mathbb{Z},\ i \in [1, N] \cap \mathbb{Z},\ x_{ij} \in X \subset \mathbb{R}^{N \times T \times C} \}$$

Where X is the dataset, N is the dataset axis, T is the temporal sequence and C represents the (x, y, z) coordinates.

Fold 1		Fold 2		Fold 3		Fold 4		Fold 5		5-Fold Mean		C	Kernel	Outliers	P.E.
Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid				
0.997	0.927	0.995	0.937	0.996	0.933	0.995	0.937	0.996	0.932	0.996	**0.933**	7	RBF	Keep	Yes
0.996	0.938	0.996	0.922	0.996	0.934	0.996	0.937	0.996	0.926	0.996	0.931	7	RBF	Keep	No
0.997	0.913	0.997	0.932	0.996	0.922	0.997	0.926	0.996	0.928	0.997	0.924	7	RBF	Global	Yes
0.997	0.922	0.995	0.922	0.997	0.929	0.996	0.936	0.997	0.926	0.997	0.927	7	RBF	Global	No
0.991	0.805	0.99	0.786	0.99	0.785	0.99	0.786	0.994	0.788	0.99	0.79	2	Linear	Keep	Yes
0.997	0.758	0.996	0.758	0.997	0.764	0.997	0.761	0.996	0.747	0.997	0.758	10	Linear	Global	No

Table 2: SVM Hyperparameter Tuning
P.E. represents indicates if *positional embeddings* are added.

2.2 Hybrid AutoEncoder SVM (HAESVM)

An alternative approach to solve the temporal issue mentioned above is to reduce the features of the input data by mapping them into a **lower dimensional space**. Generally, when data is recorded it contains a quantity of noise that is seen by models and this can present itself as a barrier during optimization. Moreover, if a dataset has many dimensions this can be another issue called ***Curse of Dimensionality***, which imposes the requirement of large amounts of data in order to capture relationships in the high dimensional space which has sparse data.

According to Siddique, Sakib, and Rahman (2019), training a classifier on top of a lower representation of the given data can improve the performance for classical models such as KNN and SVM. The recordings present in this dataset may contain floating-point errors, noise because of user movement and the space-time axes, $(150, 3)$ for a single sample, might be a cause for *Curse of Dimensionality*. As an idea of improvement over the previous classifier, I modeled and trained an AutoEncoder, using convolutional layers, to map the input data to a reduced feature space of length 128. After this reconstruction stage, a SVM is trained over the bottleneck to solve the task at hand. This architecture can be seen in the following diagram:

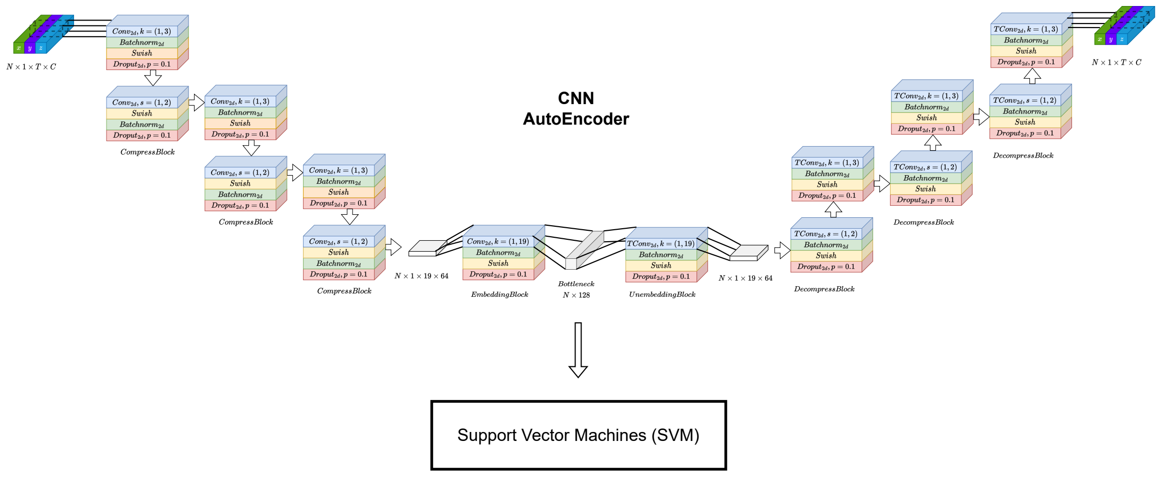

Figure 7: Hybrid AutoEncoder SVM Architecture Diagram
The AutoEncoder architecture is made up of three compression blocks, that reduce the temporal dimension using strided convolutions and double the amount of channels, followed by one embedding layer that flattens the feature maps and lastly the inverse operations which restore the temporal axis. All of the inner block operations are read in top-to-bottom order.

The model presented above includes some architectural choices which facilitate a smoother and shorter training process leading to a well represented bottleneck which can be used for further prediction using the SVM:

- The *Swish* activation function is a better alternative to *ReLU* according to the empirical performance shown by Ramachandran, Zoph, and Le (2017).
- The temporal axis is the one reduced and reconstructed, while the coordinates are considered as the channels. This should lead the model to learn the behavioral changes of the users across multiple time points.
- The pooling and unpooling operations are given by strided and transposed convolutions in order to learn richer operations as the operations are not applied on an image but on data sequences.
- 2D spatial dropout (Tompson, Goroshin, Jain, LeCun, & Bregler, 2015) with a small probability, $p = 0.1$, is used to improve generalization performance, because the channels become more correlated as the temporal axis is reduced.

The objective of the **first stage** is to minimize the *Mean Squared Error (MSE)* or the *Mean Absolute Error (MAE)* loss functions. In order to speed up the training process and keep it stable, I used various techniques:

- Mini-batches of size 32 seems to be the *optimum* value between training time and loss convergence. Lower values led to very slow traing times and higher values showed convergence issues which required many more epochs.
- The *Adam* optimizer (Kingma & Ba, 2017) is used with the *learning rate* $= 10^{-3}$, and hits a pleateau around 50 epochs with no improvement given by reducing the learning rate, neither by lowering it initially or reducing it on the go.
- To have a better starting point during model training I used *Xavier Normal Initialization* (Glorot & Bengio, 2010) and zeroed out the bias terms.

The first stage of the training process can be observed in the following graphs, on a single fold from 5-Fold Cross Validation, and the results of hyperparameter tuning can be observed in Table ....

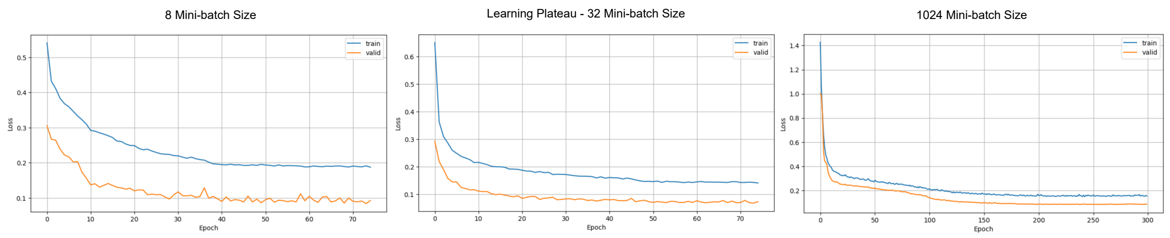

Figure 8: Training Process Across a Single Fold from 5-Fold Cross Validation
The 8 mini-batch run indicates a slow learning process with validation spikes along the epochs. The 32 mini-batch run shows a plateau that's hit around epoch 50 even though the learning rate was decreased gradually to 10^{-7}. The final 1024 mini-batch experiment showed a smooth but long learning process requiring many more epochs to hit a good enough value.

2.3 Temporal Convolutional Neural Network (TCNN)

In contrast to the previous approach, which used a learned representation and trained a classical algorithm over it, one step forward is to train a model **end-to-end** over the dataset. This would imply a restriction on the network layers to both learn a temporal representation and predict what user gave that behavior. Starting from the previous idea, I kept some elements but changed some aspects, which have proved to lead to great success in the end:

- Similarly to the AutoEncoder, the representation of the data is built using 2D convolutional layers that are being applied across the **temporal axis** while having the **coordinates in the channel dimension**. This seems to be the more intuitive approach as the convolutional operations have strong local biases and implicitly encode local changes with regards to the user behavior. By using kernels of at least size 3, the network can cover a great part of the sequence in deeper layers because of its gradually increasing *receptive field*.
- Max pooling operations are used instead of strided convolutions, which reduce some of the overhead of learning more complex operations, while exposing only the more important changes in user behavior to further layers.
- Instead of learning a separate classifier over the flattened feature maps, given by the convolutional backend, four fully-connected layers are added on top, and a final layer that predicts logits which are transformed to class probabilities using the *softmax* activation.
- 2D spatial dropout, which drops entire channels, is replaced by the usual dropout operation which zeroes out certain activations with a given probability.
- More activation functions are considered: *ReLU*, *LeakyReLU* and *Swish*.
- Stronger regularization is applied by using some of the following:
  - Dropout with $p = 0.3$ or higher;
  - Weight decay between $[2 * 10^{-4},\ 2 * 10^{-5}]$;
  - Addition of small Gaussian noise values: $\hat{X} = X + \phi,\ \phi \sim \mathcal{N}(0,\ 10^{-3^2})$;
- Similar observations regarding the optimization process are kept:
  - The Adam optimizer is used with *lr* such as $2 * 10^{-4}$ and periodic reduction of its value, starting from an $epoch = 30$ and decreasing exponentially every $step\_size = 10$ by a factor $\gamma = 8 * 10^{-1}$;
  - Mini-bacthes of size 32 offer good time / convergence tradeoff;
  - *Xavier Normal Initialization* is used;

The architecture of the **Temporal Convolutional Neural Network (TCNN)** can be inspected in the following diagram:

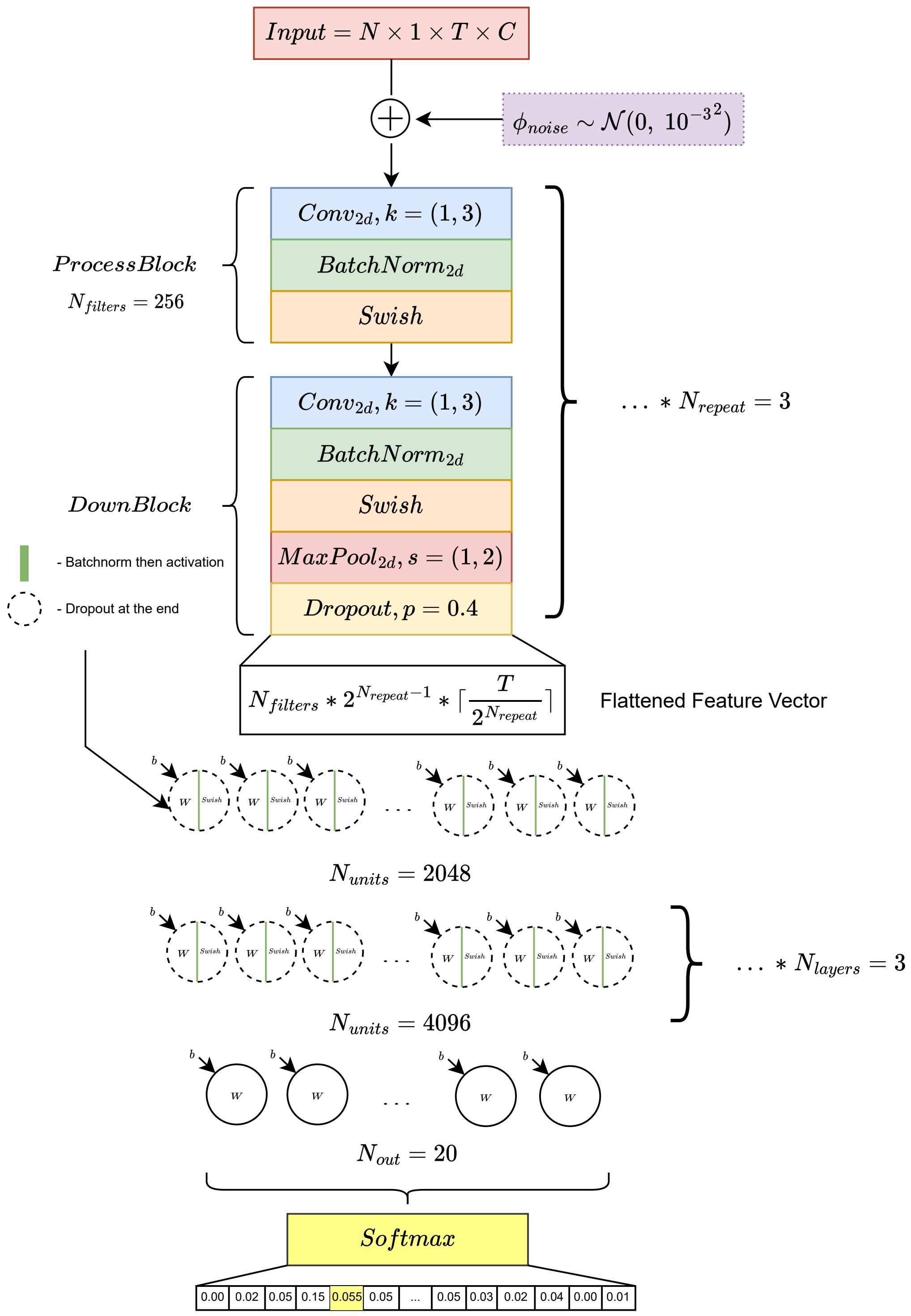

Figure 9: Caption

3 ok

This journal uses a style based on the APA system (see here).
The following are some basic citation commands in LaTeX:

`\citet` → Jenset and McGillivray (2017)
`\citet` → Australian Institute of Health and Welfare (2011)
`\citet` → Shree, Liu, Gordon, and Hobbs (2019)
`\citep` → (Fabricius-Hansen & Haug, 2012)
`\citealp` → (Eckhoff et al., 2018)
`\citealp` → (Eckhoff et al., 2018; Fabricius-Hansen & Haug, 2012; Shree et al., 2019)

3.0.1 Other simple functions

To add bullet points:

- Some point
- Another point

Or numbered points:

1. Some numbered point
2. Another numbered point

This is an example of footnote[1].

This is a simple table:

Table 3: A caption.

1	2	3	4
a	b	c	d
e	f	g	h

Please refer to your table using: Table 3.

To add a figure, upload the figure into the `images` folder, and then embed it:

Figure 10: JOHD's logo.

To resize the figure:

Figure 11: JOHD's logo.

[1]This is a footnote

Figure 12: JOHD's logo.

Please refer to your figures as: Figure 10, Figure 11, etc.

4 Dataset description

Here you can provide, if applicable, information about the dataset(s) whose creation, collection, management, access, processing or analysis have been discussed in this paper, following this schema:

Object name Typically the name of the file or file set in the repository.

Format names and versions E.g., ASCII, CSV, Autocad, EPS, JPEG, Excel, SQL, etc.

Creation dates The start and end dates of when the data was created (YYYY-MM-DD).

Dataset creators Please list anyone who helped to create the dataset (who may or may not be an author of the data paper), including their roles (using and affiliations).

Language Languages used in the dataset (i.e., for variable names etc.).

License The open license under which the data has been deposited (e.g., CC0).

Repository name The name of the repository to which the data is uploaded. E.g., Figshare, Dataverse, etc.

Publication date If already known, the date in which the dataset was published in the repository (YYYY-MM-DD).

5 Method

Describe the methods used in the study.

6 Results and discussion

Describe and discuss the results of the study.

7 Implications/Applications

Provide information about the implications of this research and/or how it can be applied.

Acknowledgements

Please add any relevant acknowledgements to anyone else that assisted with the project in which the data was created but did not work directly on the data itself.

Funding Statement

If the research resulted from funded research please list the funder and grant number here.

Competing interests

If any of the authors have any competing interests then these must be declared. If there are no competing interests to declare then the following statement should be present: The author(s) has/have no competing interests to declare.

References

Australian Institute of Health and Welfare. (2011). *Australia's health 2004.* Retrieved from http://www.aihw.gov.au/publications/index.cfm/title/10014 (last accessed 10 September 2021)

Chang, C.-C., & Lin, C.-J. (2011). Libsvm. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27. DOI: 10.1145/1961189.1961199

Eckhoff, H. M., Bech, K., Bouma, G., Eide, K., Haug, D., Haugen, O. E., & Jøhndal, M. (2018). The PROIEL treebank family: A standard for early attestations of Indo-European languages. *Language Resources and Evaluation*, *52*(1), 29–65. DOI: 10.1007/s10579-017-9388-5

Fabricius-Hansen, C., & Haug, D. T. T. (2012). Co-eventive adjuncts: Main issues and clarifications. In C. Fabricius-Hansen & D. T. T. Haug (Eds.), *Big events, small clauses* (pp. 21–54). Berlin-Boston: De Gruyter.

Glorot, X., & Bengio, Y. (2010). *Understanding the difficulty of training deep feedforward neural networks.* Retrieved from https://www.researchgate.net/publication/215616968_Understanding_the_difficulty_of_training_deep_feedforward_neural_networks

Jenset, G., & McGillivray, B. (2017). *Quantitative historical linguistics: A corpus framework.* Oxford: Oxford University Press.

Kingma, D. P., & Ba, J. (2017, Jan). *Adam: A method for stochastic optimization.* Retrieved from https://arxiv.org/abs/1412.6980

Radu. (2022). *Smartphone user identification.* Kaggle. Retrieved from https://kaggle.com/competitions/pml-2022-smart

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *CoRR*, *abs/1710.05941*. Retrieved from http://arxiv.org/abs/1710.05941

Shashua, A. (2009, Apr). *Introduction to machine learning.* Retrieved from https://arxiv.org/abs/0904.3664v1

Shree, J., Liu, E., Gordon, A., & Hobbs, J. (2019). Deep natural language understanding of news text. In *Proceedings of the first workshop on narrative understanding* (pp. 19–27). Minneapolis, Minnesota: Association for Computational Linguistics. DOI: 10.18653/v1/W19-2403

Siddique, M. A. B., Sakib, S., & Rahman, M. A. (2019). Performance analysis of deep autoencoder and NCA dimensionality reduction techniques with knn, ENN and SVM classifiers. *CoRR*, *abs/1912.05912*. Retrieved from http://arxiv.org/abs/1912.05912

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/cvpr.2015.7298664

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from http://arxiv.org/abs/1706.03762

Supplementary Files (optional)

Any supplementary/additional files that should link to the main publication must be listed, with a corresponding number, title and option description. Ideally the supplementary files are also cited in the main text. Note: supplementary files will not be typeset so they must be provided in their final form. They will be assigned a DOI and linked to from the publication.