Project 2: Time Series and Representation Learning

Project 2 consists of three parts. The first part explores architectures for supervised learning tasks on time series, while the second investigates transfer and representation learning on this data modality. In the third part, you will recap your findings and answer some general questions about the methods you have seen. You will explore techniques that enable learning on time series data using shallow and deep machine-learning methods. Before starting the project, be aware of the following points:

- The report has a word limit of 5000 words excluding references. There is no restriction on the number of plots.
- The report must be handed in as a PDF.
- The report has to be self-contained, i.e. no references to code.
- Underlined sections within questions specify how many points can be achieved by solving that specific subquestion.
- You will also need to hand in your code. Please include a requirements.txt or similar for your Python environment and a README.md explaining how to run your code.
- Use train/validation splits for training and tuning only. Report results on the test set. Note that the performance of the different methods can vary a lot.
- Using publicly available code is okay, but properly reference repositories when you use them. Of course, you are not allowed to use the code of other teams from the course.

This project studies two electrocardiogram datasets. We will provide you with datasets from Kaggle on Moodle and on the student cluster([your home directory]/ml4h_data/project2/project2_TS_input):

- The PTB Diagnostic ECG Database (https://physionet.org/content/ptbdb/1.0.0/)
- MIT-BIH Arrhythmia Database (https://physionet.org/physiobank/database/mitdb/)

All the samples are cropped, downsampled, and padded with zeros, if necessary, to the fixed dimension of 188. Each dataset consists of two CSV files, split into training and test data. Each of these CSV files contains a matrix, with each row representing an example in that portion of the dataset. The final element of each row denotes the class to which that example belongs.

Part 1: Supervised Learning on Time Series (20 Pts)

In this first part, you will investigate different machine learning models for supervised learning on time series. You will focus **on the PTB Diagnostic ECG Database only**, for which the task is to classify whether a given ECG time series is healthy or abnormal.

Q1: Exploratory Data Analysis (2 Pts)

Get familiar with the PTB dataset by examining different example time series and studying the distribution of labels (1 Pt). Choose one or more appropriate metric(s) for the classification task (1 Pt).

Q2: Classic Machine Learning Methods (5 Pts)

Classic ML methods, such as Logistic Regression or Random Forests, take a fixed set of features as input. These models learn linear or non-linear combinations of the features to make a prediction (regression or classification). They often have a notion of feature importance but also need careful feature design. Feature design may involve adding hand-designed features that are based on a priori knowledge about the dataset or the application (e.g. signal processing knowledge in case of time series).
(1) Choose (at least) two different classic (non-deep) ML classifiers and train them on the raw time series of the PTB dataset without adding new features. Report test set performances (1pt).
(2) Feature design and engineering is important for classic ML methods. Repeat (1) with additional features of your choice, try to improve the predictive performance of your method, and report again test set performance (3pts). See e.g. https://tsfresh.readthedocs.io/en/latest/ for signal processing-based features.
(3) Evaluate the pros and cons of the different classifiers used (1pt).

Q3: Recurrent Neural Networks (4 Pts)

Recurrent Neural Networks (RNNs) are designed to handle sequential data at variable lengths by applying the same network to every time point, preserving a notion of memory through a hidden state vector. As RNNs suffer from vanishing or exploding gradient issues, Long Short-Term Memory (LSTM) networks use gating mechanisms to control the flow of information in and out of the memory cell[1]. Implement an LSTM network, train it on the PTB dataset, and report test set performance (2 pts)

Vanilla implementations of RNNs and LSTM are unidirectional in that they process time series in a sequential manner. Why might a bidirectional model be also useful here[2] (1 pt)? Implement and train a bidirectional model (1 pt). Report test performance.

Hint: make use of existing model implementations in neural network libraries (keras, pytorch, etc).

---

[1] Hochreiter and Schmidhuber, “Long short-term memory”, 1997.
[2] Schuster and Paliwal, “Bidirectional recurrent neural networks”, 1997.

Q4: Convolutional Neural Networks (4 Pts)

Convolutional Neural Networks (CNNs) also capture dependencies within the data, this time by encouraging robustness against spatial translation. Discuss some of the relative advantages of RNNs and CNNs for time series tasks (1 pt). Implement and train a vanilla CNN and a CNN with residual blocks[3] (2 Pt). Report test performance and comment on your findings, discussing why residual layers do/don't help (1 pt).

Q5: Attention and Transformers (5 Pts)

The attention mechanism[4] can help capture longer-range dependencies in the data. Implement, train, and test a simple transformer model (2 pts). What differences and advantages do transformers show over recurrent neural networks (1 pt)?

Visualize the attention map learned by your model using input layer weights. Do you gain any insight on important data points to distinguish between normal and abnormal sequences (2 pts)?

Part 2: Transfer and Representation Learning (20 Pts)

In this part, we explore how to leverage information from the larger MIT-BIH database to improve learning on the smaller PTB dataset. In transfer learning, we use representations from supervised learning on a related but distinct task (here, arrhythmia classification). In representation learning, we use unsupervised learning strategies.

Q1: Supervised Model for Transfer (3 Pts)

In transfer learning, we use a model trained for another task to improve performance on a task of interest. Reuse your implementation for one of the deep learning architectures explored in Part 1 to train a model on the MIT-BIH dataset. Report test performance (1 Pt). Motivate your choice of model and metrics for this arrhythmia classification task (2 Pt).

After removing one (or more) output layer(s) of the model specific to the supervised task, note that this model can be seen as a pre-trained time series encoder. We will investigate transfer strategies using this model in Q3.

Q2: Representation Learning Model (4 Pts)

Not all datasets are labeled as collecting labels, especially for medical data, is expensive. Representation Learning tries to overcome the lack of labels by first solving surrogate tasks.

---

[3] He et al., "Deep Residual Learning for Image Recognition", 2016.
[4] Vaswani et al., "Attention is all you need", 2017.

The training results in low-dimensional representations that are transferable to similar datasets and useful for different downstream tasks.
There are different approaches to self-supervised representation learning, such as autoencoder-based approaches or contrastive learning[5][6]. Familiarize yourself with contrastive learning approaches for time series (see e.g. Awesome Self-Supervised Learning for Time Series (SSL4TS)) and, especially, the InfoNCE approach[7]. (Remark: you are free to use any other representation learning approach too.)

1. Pretrain an encoder using the objective of your choice on the training set of the MIT-BIH dataset. How do you monitor this pre-training step? (2 pts) Note that the architecture of your encoder should be as similar as possible to that trained in Q1, for a fair comparison in the next stage.
2. Evaluation of the Pretraining: train a classic ML method from Part 1 to predict the MIT-BIH labels from the learned representations of the training set and evaluate the representation learning quality on the test set of the MIT-BIH dataset (2 pts), using the same metric(s) as in Q1.

We will investigate how to use this representation learning model for our downstream task in Q3.

Important: summarize your results in a table and add that table to the report!

Q3: Visualising Learned Representations (4 Pts)

In Q1 and Q2, you have pre-trained different encoders, or representation learning models, based on the MIT-BIH dataset.

With both encoders from Q1 and Q2, obtain representations for both the MIT-BIH and PTB datasets by feeding them through the encoder. Visualize your learned representations through a dimensionality reduction technique such as t-SNE[8] or UMAP[9] (2 Pt). Are data points with different labels distributed identically? (1 pt) Are the two datasets distributed identically? (1 pt) Provide quantitative metrics to assess this.

Q4: Finetuning Strategies (9 Pts)

Our goal is now to leverage our pre-trained encoder models for prediction on the PTB dataset. For both encoders from Q1 and Q2, consider the different transfer/finetuning strategies:

1. **Classic ML method:** Similar to Q3, obtain representations for the PTB dataset by feeding the dataset through the pre-trained encoder. Use a classic ML method from Part 1 to train and test for the PTB task using these representations. (1 pt)
2. **MLP output layers:** Add output layer(s) for the PTB binary class to your encoder model. Implement the following finetuning strategies (3 pts):
    A. Train the output layer(s) only on the PTB dataset, freezing the encoder.

[5] Bengio, Courville, and Vincent, “Representation Learning: A Review and New Perspectives.”
[6] Ericsson et al., “Self-Supervised Representation Learning.”
[7] Oord, Li, and Vinyals, “Representation Learning with Contrastive Predictive Coding.”
[8] Van der Maaten and Hinton. “Visualizing data using t-SNE”, 2008.
[9] McInnes et al., “Umap: Uniform manifold approximation and projection for dimension reduction”, 2018.

B. Train the entire model on the PTB dataset (encoder + output layers).
C. First, train the output layers, then unfreeze and train the entire joint model in two separate stages.

In all cases, report performance on the PTB test dataset using the same metric(s) as in Part 1 and summarize the metrics in a single table. Comment on potential differences in performance over Part 1 results (1 pt). Which of the above fine-tuning strategies performs best and why? (2 pts) Which of the two pre-training strategies considered (transfer learning vs. representation learning) performs best and why? (2 pts)

Part 3: General Questions (7 Pts)

To conclude, we ask you to answer the following questions to recap and reason about the project.

Q1: There are many machine learning settings where classic methods are still competitive with deep learning architectures. Have you observed this in this project? Why is this (not) the case? (2 pts)

Q2: When modeling time series using deep learning architectures, when might you want a causal/unidirectional architecture? Give an example of a task. (1 pt)

Q3: Can you think of an attention-related bottleneck regarding very (very) long time series? Conceptually, which deep methods from above are more suitable for such long time series? (2 pt)

Q4: What are some challenges in using self-supervised representation learning? What difficulties have you observed in your approach? Can you think of additional ones? (2 pt)