INF264 – Project 2
Digit Recognizer

Deadline: Friday October 20th, 23:59
Deliver your submission at MittUiB

General Information

Projects are a compulsory part of this course. You need to pass both projects to be eligible for the final exam. **The project can be done either alone or in pairs**. If you decide to work in pairs, add a paragraph to your report explaining the division of labor. Note that both students will get the same grade, regardless of the division of labor. Grading will mainly be based on the following three qualities:

- **Correctness.** Your answers are correct and your code works as expected.
- **Clarity of code.** Your code is easy to read and understand for someone who has not seen it before. Use meaningful variable names and comments where necessary.
- **Reporting.** Your report is well-structured and clearly written.

Especially, weight is put on model selection, evaluation procedures, and the quality of the report when grading.

Deliverables

You should deliver **exactly two files**[1]:

1. A detailed **PDF report** containing an explanation of your approach, design choices and results. The main point of the report is to show that you have understood the task and that you can explain your approach and results in a clear and concise manner. See section B.2 for more details about the report.
2. A **ZIP file** containing your code. We may want to run your code if we think it is necessary to confirm that it works the way it should. Please include a `README.txt` file in your ZIP file that briefly explains how we should run your code. Note that all the numbers that you report should be reproducible from the code that you submit. See section B.1 for more details about the code.

Programming languages. Please submit your code in Python. Notebooks, standalone scripts, or a combination of both are all acceptable. If you want to use another programming language, please ask us first.

[1]The reason we ask you **not** to put the report inside the ZIP file is that in this way, the graders can use MittUiB"s Speedgrader-functionality. So please make graders" work easier and follow the instructions.

Code of Conduct

All code and the report delivered should be your own work. In other words, you are not allowed to directly copy code from online tutorials, public code repositories etc. If you do use code from other sources, you should clearly state this in your report (and add a comment in the code). You are allowed to use machine learning libraries.[2] You are, of course, also free to use libraries such as `matplotlib`, `numpy` and `pandas` for tasks such as data analysis, data manipulation and visualisation. If you are unsure whether something is allowed or not, please ask us first. Note that this year's project differs from last year's project both in terms of the task and the dataset. So submitting last year's project is not an option.

Late Submission Policy

All late submissions will get a deduction of 2 points. In addition, there is a 2-point deduction for every starting 12-hour period. That is, a project submitted at 00:01 on October 21st will get a 4-point deduction and a project submitted at 12:01 on the same day will get a 6-point deduction (and so on). All projects submitted on October 23rd or later are automatically failed. Executive summary: Submit your project on time. The late penalties are designed to be harsh enough so that nobody should benefit by returning their work late. Also note that late penalties can easily drag a good project under the acceptance threshold. There will be no possibility to resubmit failed projects.

1 Tasks

Learning goals: Learn a proper process for selecting and evaluating classifiers, and learn how dimensionality reduction can be used in conjunction with supervised methods.

Figure 1: Santa's workshop. (Image by Freepik.com)

Scenario: As December draws near, Santa is concerned about meeting the gift preparation deadline for all the children. The Chief Elf Officer (CEO) of Santa's Workshop has reached out to you, a

[2]Using `sklearn` should be sufficient to complete this project. If you are ambitious about deep learning, you can consider to use libraries such as, for example, `PyTorch`.

prominent machine learning expert, to enhance the efficiency of the sorting process. Your mission is to create an advanced automated gift recognition system that could potentially save Christmas this year.

The production department is responsible for crafting 16 distinct types of gifts, each marked with a handwritten hexadecimal digit (a number from 0 to 9 or a letter from A to F) indicating its type. These gifts are transported to the sorting department via a conveyor belt, where they need to be sorted based on their markings. Engineers have installed a camera to capture images of each gift as it arrives on the conveyor belt. See section A for more details about the dataset(s) you will be working with.

Problem 1: Digit Recognizer

Your main goal is to **design and build a reliable classifier** that can take input images from the camera and correctly identify the corresponding label for each gift. It is important to note that sometimes the diligent elves in the production department may forget to label certain gifts. In such cases, the camera will capture empty images. Your system must be capable of detecting these empty labels, allowing the CEO of the production department to be notified and take necessary actions.

You should try (at least) **two different types of classifiers**[3] and perform model selection including hyperparameter tuning. You should also compute an estimate of the expected performance on new data for your final classifier. How you choose to measure the performance of your classifier is up to you, but you should justify your choices.

Problem 2: Dimensionality Reduction

Investigate how PCA (Principal Component Analysis), or another dimensionality reduction technique, can be used to reduce the dimensionality of the data. That is, apply PCA to the dataset to reduce the number of features from 400 to k (where $k < 400$ is a hyperparameter). Train your classifier on the reduced dataset and evaluate how this affects the performance and computational efficiency of final classifier from the previous task.

You can use `sklearn.decomposition.PCA` to perform PCA. Remember to fit the PCA model on only the training data to avoid data leakage. You should also consider how to choose the number of components k for PCA.

Problem 3: Detecting Out-of-Distribution Images

Since the elves in the production department have been a bit overworked lately, they have been a bit careless and forgotten to wrap all the gifts. This means that the camera has captured some images of the gift contents instead of the labels. You have been given a separate dataset `corrupt_dataset.npz` containing these images (with 85 out-of-distribution images).

The CEO of the production department is not happy about this, and has asked you to help identify these corrupt (out-of-distribution) images so that they can be removed from the dataset. Your task is to come up with an approach to identify these corrupt images. Maybe you can repurpose your classifier to help you with this task? You are free to be creative in your approach.

Of course, Santa also expects you to provide a **detailed report** explaining your work, including results and the reasons behind your design choices (see section B.2 for more details about the report).

[3]For our purposes, two classifiers are of different type if they were covered in different lectures.

A Dataset Description

Download the dataset[4] `dataset.npz` from MittUiB. You can load the dataset using NumPy as follows:

```
dataset = np.load("dataset.npz")
X, y = dataset["X"], dataset["y"]
```

The NumPy array `X` has shape `(n, 400)` consisting of n grayscale images of size 20×20 pixels, and `y` has shape `(n,)`. A single image `X[k]` is a NumPy array with shape `(400,)` taking values in the range $0 - 255$ where 0 is black and 255 is white. Note that the images are flattened into a 1-dimensional array. If you need to convert a flattened image `X[k]` into a 2-dimensional array, you can call `X[k].reshape(20,20)`. See fig. 2 for an illustration of reshaping a NumPy array.

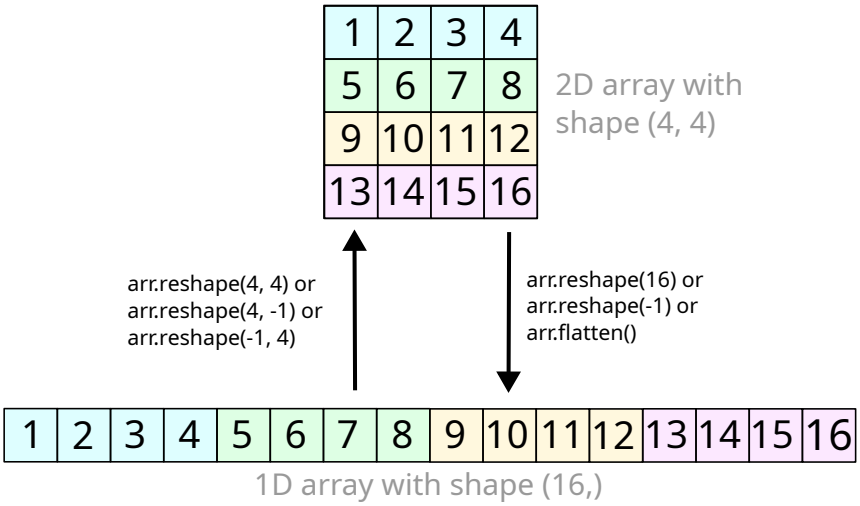

Figure 2: Illustration showing how NumPy array's `reshape()` method works. Note that `flatten()` returns a copy of the array, whereas `reshape()` tries to return a new view if possible.

To visualise the image `X[k]` you can use the following code:

```
import matplotlib.pyplot as plt
plt.imshow(X[k].reshape(20,20), vmin=0, vmax=255, cmap="gray")
plt.show()
```

The labels `y` consist of integers in the range $0 - 16$ encoding the class each image belongs to. The different labels are listed in table 1.

[4]The datasets for this project are derived from EMNIST (Extended MNIST) from [Coh+17] and the Fashion MNIST from [XRV17].

Class Name	Label	Example images
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	
Empty	16	

Table 1: The first column lists the different classes we want to differentiate between. The middle column lists the corresponding labels as they appear in `y`. The last column shows five examples images for each class.

The second dataset `corrupt_dataset.npz` contains images of the same size as the original dataset, but without any labels `y`. These can be loaded similarly to the original dataset.

B Guidelines and Suggestions

B.1 Code and Implementation

The following list contain some points you should consider when implementing your solution:

- ☐ **Data exploration:** What can you say about the dataset, and how does it affect your design choices? What is the class distribution? Do not be afraid of actually looking at some of the images in the dataset, either.
- ☐ **Metrics:** Decide on how you want to measure the goodness of a classifier. This should be based on the task at hand, and the dataset.
- ☐ **Pre-processing:** Do you need to do any pre-processing of the data?
- ☐ **Model selection:** You should try **at least 2 different types** of classifiers. Experiment with hyperparameters. It is crucial to have correct model selection and evaluation procedures. Based on your model selection procedure, automatically select the best model. Do not touch the test set until you have selected your final model.
- ☐ **Quality assessment:** Remember to analyse your results. Perform sanity checks. How well does your selected model perform on unseen data? What are the weaknesses of your model? Consider confusion matrices and computing precision/recall for the different classes to gain insight[5].
- ☐ **Visualisation:** Can you produce other insightful plots or other types of visualisations? Can you manually inspect some of the images that your classifier misclassifies?
- ☐ **Reproducibility:** Your results should be reproducible, i.e., one should be able to easily run your code and get the same numbers that you give in your report. Thus, you should write an automated test pipeline that runs all of your tests (given enough time). That is, training and assessment of all models and hyperparameters. If you use Jupyter notebooks, make sure that your results can be reproduced after restarting the kernel and running all cells in order.

Constraints. Model selection and evaluation should not take unreasonable amounts of time. If it takes more than 30 minutes to run your code, you should consider reducing the complexity of your model, use a subset of the data for model selection (but train the final model on the full dataset) or tune fewer hyperparameters. All code should be CPU-based (no GPU acceleration).

B.2 Report

The report should consist of two parts, a **summary** and a **technical report**, both in the same PDF.

Summary

The summary should give a short, non-technical overview of your project. You should also argue, based on your results, whether the machine learning approach is appropriate for this task and what is your expectation of its performance in real-life.

[5] See, for example, `confusion_matrix()` and `classification_report()` from `sklearn.metrics`.

Technical report

The technical report should cover what you have actually done and why. It should contain detailed information on your design choices and experimental design. The list below is a guideline for what you should include in your report (but you are free to structure it as you see fit):

- ☐ Your observations about the dataset.
- ☐ Pre-processing steps (if any).
- ☐ Choice of candidate models and hyperparameters. And why were the others omitted?
- ☐ Chosen performance measure. Justify your choice.
- ☐ Model selection scheme(s) that you used. Justify your choices
- ☐ What is your final classifier, and how does it work? Justify why you think it is the best choice.
- ☐ How well it is expected to perform in production (on new data). Justify your estimate.
- ☐ How did dimensionality reduction (PCA) affect the performance and computational efficiency of your classifier?
- ☐ How did you find corrupt images in the unlabeled dataset? What was your approach and did it work as expected?
- ☐ Measures taken to avoid overfitting.
- ☐ Given more resources (time or compute), how would you improve your solution?

Remember, **our main goal is learning**, so it is perfectly fine to report failed experiments. Especially if you can explain why things did not work as you initially expected. Figures and plots are expected. Also note that whenever you report performance measures, **clearly state which data set you used** to compute them (training, validation, test).

References

[Coh+17] Gregory Cohen et al. *EMNIST: an extension of MNIST to handwritten letters*. 2017. arXiv: 1702.05373 [cs.CV].

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].