

# INF264

## Project 2: Digit recognizer

**Deadline:** Friday October 20th, 23:59

**Deliver at:** <https://mitt.uib.no/courses/42639/assignments/77036>

Projects are a compulsory part of the course. This project contributes a total of 25 points to the final grade. You need to upload your answer to MittUiB before 23:59 on the 20th of October.

**The project can be done either individually or in pairs.** If you decide to work as a pair, add a paragraph to your report explaining the division of labour. Note that both students will get the same grade regardless of the division of labour.

Grading will be based on the following qualities:

- **Correctness:** your answers and code are correct and clear.
- **Clarity of code:** documentation, naming of variables, logical formatting.
- **Reporting:** thoroughness and clarity of the report.

Especially, weight is put to correct use of model selection and evaluation procedures.

**Deliverables:** You should deliver **exactly two files**:

1. a **PDF report** containing an explanation of your approach and design choices to help us understand how your particular implementation works. You can include snippets of code in the PDF to elaborate any point you are trying to make, and
2. a **zip file** of 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 explains how we should run your code. If you have multiple files in your code directory, you must mention in the `README.txt` file which file is the main file that we need to run to execute your entire algorithm. Note that all the numbers that you report should be reproducible from the code that you return. Please **do not include the data files** but refer to them in your code using a relative path.

**Programming languages:** You are allowed to submit your implementation in the following languages: Python, C++, Java. Teaching assistants support Python. If you choose to implement the project in some other programming language, you are on your own.

**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. You are allowed to use machine learning libraries.<sup>1</sup> 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.

**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.) There will be no possibility to resubmit failed projects, so start working early.

---

<sup>1</sup>Using `sklearn` should be sufficient to complete this project. If you are ambitious about deep learning, you can consider to use libraries such as `PyTorch` or `keras`.

# 1 Task: Digit Recognizer

**Learning goal:** Learn a proper process for selecting and evaluating classifiers.



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 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.

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 (see section 1.2 for more details about the task). Santa also expects you to provide a **detailed report** explaining your work, including results and the reasons behind your design choices (see section 1.3 for more details about the report).

## 1.1 Data

You can download the two data files<sup>2</sup> `emnist_hex_labels.npy` and `emnist_hex_images.npy` from the following URL: <https://filesender.sikt.no/?s=download&token=01f980ab-6d18-4a21-9d81-fc2ce591123c>. The file `emnist_hex_images.npy` contains 111399 images each of size  $20 \times 20$ , and the file `emnist_hex_labels.npy` contains the corresponding labels. To load the data files, you can simply use NumPy's `load()` function like this:

```
X = np.load("emnist_hex_images.npy")
y = np.load("emnist_hex_labels.npy")
```

The NumPy array `X` has shape `(111399, 400)`, and `y` has shape `(111399,)`. A single image `X[k]` (where  $0 \leq k < 111399$ ) is then 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 its use.

---

<sup>2</sup>The dataset is derived from the EMNIST (Extended MNIST) dataset from [Coh+17].

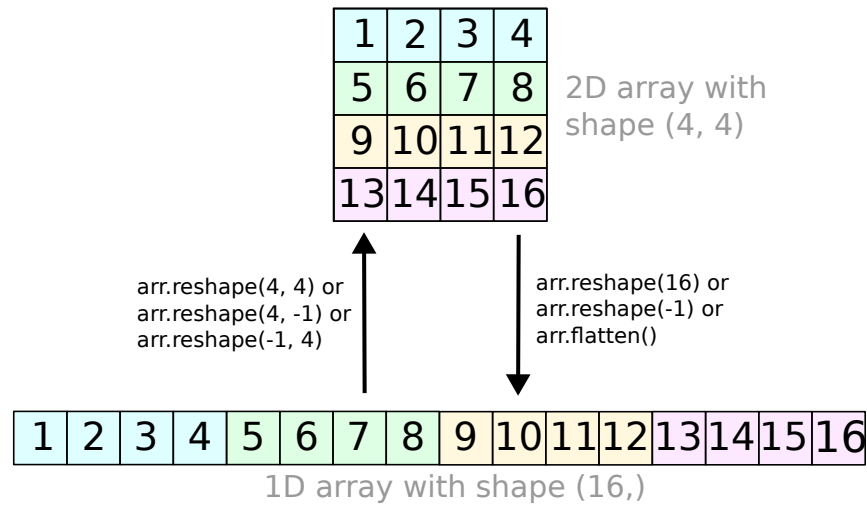


Figure 2: Illustration showing how NumPy array's `reshape()` method works. Note that `flatten()` returns a copy of the array, whereas `reshape()` tries to return a new view if possible.

If you want to visualise the  $k$ th image in  $X$  you can import `matplotlib` by `import matplotlib.pyplot as plt` and use `reshape()` together with `plt.imshow()` like this:

```
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 which encodes the class each image belongs to. The different labels are explained in table 1.


















Class	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 image	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 the file `emnist_hex_labels.npy`. The last column shows five examples images for each class from the file `emnist_hex_images.npy`.

## 1.2 Code

The goal is to produce a classifier that predicts the labels of the handwritten hexadecimal digits as well as possible. The following checklist is not necessarily complete, but can help you get started on the implementation:

## Implementation checklist

- ☐ **Data exploration:** Spend some time on getting to know the data you are working with. 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.
- ☐ **Model selection:** You should try **at least 3 different types**<sup>3</sup> 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.
- ☐ **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<sup>4</sup>.
- ☐ **Visualisation:** Can you produce other insightful plots or other types of visualisations? Can you manually inspect some of the images that your classifier struggles to classify correctly?
- ☐ **Reproducibility:** Your results should be reproducible. That is, the CEO of Santa's Workshop should be able to verify your claims. Meaning that 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.

---

<sup>3</sup>For our purposes, two classifiers are of different type if they were covered in different lectures.

<sup>4</sup>See, for example, `confusion_matrix()` and `classification_report()` from `sklearn.metrics`.

## 1.3 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.

### Technical report

The technical report should tell what you have actually done and why. It should contain detailed information on your design choices and experimental design. The technical report should contain at least the following information:

- ☐ 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 unseen data). Justify your estimate.
- ☐ Measures taken to avoid overfitting.
- ☐ Given more resources (time or computing resources), how would you improve your solution?



Remember, **our main goal is learning**, so it is perfectly fine to report failed experiments. Especially, it is appreciated 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.

## References

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