# 2AMM10 Assignment 1 : Character Recognition in Different Scripts

## 1 Introduction

Dr. Mado Vlenkovski is a world-leading paleographer, who specializes in translating and interpreting pieces of ancient text in varying scripts. Such texts are often found in archaeological sites. Although he is already quite old, dr. Vlenkovski is always keen on using the newest technologies to improve the quality of his work. For example, whereas many of his colleagues still use pen and paper, dr. Vlenkovski has built a software package called Pal-O-Graph™, which can be used to annotate texts. After consulting ChatGPT – the Chat Group for Paleographers and Translators – dr. Vlenkovski has learned that there is one particular bottleneck in Pal-O-Graph™, namely the character annotation task.

Character annotation is a key part in the text annotation and translation process, where each written character in the relevant alphabet is mapped to the right character class. This is a rather repetitive and error-prone task. As such, dr. Vlenkovski would like to include a module in Pal-O-Graph™ that can streamline this process. You, as AI and machine learning experts, have been asked by dr. Vlenkovski to build such a tool.

After an initial consultation with dr. Vlenkovski, you have learned about the following requirements and characteristics of the problem:

- Many paleographers specialize in specific alphabets, for example those found in a particular geographic region. Consequently, paleographers would like to be able to provide a set of annotated images of characters from a new alphabet *at inference time*. This means that the model should generalize to alphabets not seen during training, given a single set of example annotated images from that alphabet.
- Ideally, the model would classify every character correctly in a single guess. Practically speaking, the community would also benefit from a model that suggests a set of $k$ character labels. By incorporating such a model in a clever user interface, this could speed up the annotation process. As such, we would like to know the top-$k$ accuracy for the model, for the following values of $k$: $\{1, 2, 4, 8\}$.

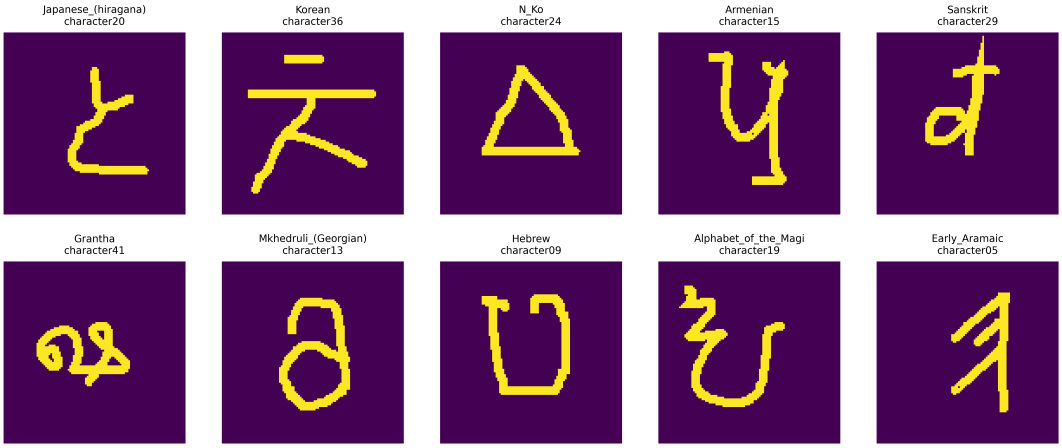


Figure 1: Some example images of characters.

### 1.1 Task 1: Character recognition

Based on the above problem description, the goal of task 1 is to build a model and train it using the provided training set $T$. At inference time, a set $A$ of annotated images from a new alphabet is provided, where each $a \in A$ has a provided ground-truth label $l(a)$. Note that the new alphabet was not in the training set. The task of the model is then to classify images in a set of unseen images $U$ from the same new alphabet. As such, the model should produce a set $M(u)$ of $k$ predicted labels for each image $u \in U$, with the goal that the true label $l(u)$ is in $M(u)$.

As resources for this task, you have a training dataset consisting of images of characters from varying alphabets and their labels. To evaluate your model, a test set has been constructed, which consists of multiple sets of images $A$ and $U$, corresponding to the annotated and to-be-labeled images, for 20 new alphabets that were not in the training set. For this task, we define accuracy as the total number of images from all sets $U$ combined that have the correct label in their respective set of $k$ suggested labels, divided by the total number of images in all sets $U$:

$$\text{top-}k\text{-accuracy(model)} = \frac{\sum_U \sum_{u \in U} \mathbf{1}_{l(u) \in M(u)}}{\sum_U |U|}.$$

In the above equation, $\mathbf{1}_{l(u) \in M(u)}$ is the indicator function which equals 1 if $l(u) \in M(u)$, and 0 otherwise.

**Hints:**

- What type of architecture would be suitable for this kind of data? What motivations do you have for your choice of architecture (symmetries, pragmatic, intuition)?
- In terms of performance, your model might not achieve extremely high accuracy given that the model needs to generalize to unseen classes at test time. As a ballpark estimate, you can expect to achieve a top-$k$ accuracy of at least 30% for $k = 1$, and at least 75% for $k = 8$.

### 1.2 Task 2: Rotated Character Recognition

Task 2 is almost identical to task 1. However, in this task, we are interested in building a more robust solution. You are given the same set of training images as in task 1. However, now each image in the set $A$ of annotated images and the set $U$ of to-be-labeled images has been randomly rotated. Let us refer to these sets of rotated images as $A'$ and $U'$. The goal

of the task remains the same: first, train a model using the training set $T$. Then, the model should be able to use the annotated images in $A'$ to classify the images in $U'$ for each new alphabet. Note that it is not allowed to use the sets $A$ and $U$ from task 1 in the process of solving task 2; only the training set from task 1 can be used.

**Hints:**

- Since this task requires your approach to additionally generalize across rotations, you can expect a drop in accuracy compared to the model of task 1. However, a good solution for task 2 should at least substantially outperform the model of task 1 applied to the rotated sets $A'$ and $U'$ of task 2.
- The materials presented in the course provide the necessary background to come up with a solution for this task. However, depending on your approach, a straightforward application of these materials might not directly yield a good model. In this case, coming up with an approach that is not discussed directly in the course materials might be required to get an effective solution.
- It could help to look for inspiration online or in papers. In this case, we strongly recommend to try simple techniques and avoid using overly complex methods. Adding complexity in your method will not result in additional points over an effective simple solution.

## 1.3 Task 3: Exploiting Domain Knowledge

Until now, we have considered the task of classifying individual images of characters. However, from the problem description it is clear that the tool will often be used for classifying characters in a sequence of characters. Paleographers typically have a lot of knowledge about which character sequences are more likely to be observed than others. In this task, you will look into how to exploit such domain knowledge in combination with a data-driven machine learning model.

Specifically, we consider the following simplified setting. Assume each alphabet has two types of characters, 'type I' and 'type II'. For example, you can imagine them being consonants and vowels. For each set $A$ of annotated images of a new alphabet of task 1, the paleographer provides the type $\tau(a)$ of the characters of the images $a \in A$. Further, the paleographer provides the probabilities that a *preceding character* $p$ of type $\tau(p)$ is followed by a character $c$ of type $\tau(c)$ in the sequence, i.e. $P(\tau(c) = X|\tau(p) = Y)$, where $X$ and $Y$ are type I or type II. Finally, for each unlabeled image $u \in U$, we are given the type $\tau(p_u)$ of its preceding character $p_u$ in the sequence. Note that, although here we assume a simplified setting where each alphabet consists of two types of characters, these types do not relate to each other across alphabets.

Given this information, the goal is to somehow incorporate the given domain knowledge (in the form of the probabilities given by the paleographer) in your labeling procedure. That is, we want to find the set of labels $M'(u)$, such that the probability $P(l(u) \in M'(u)|u, \tau(p_u))$ is maximized.

**Hints:**

- There may be multiple reasonable solution approaches, and there is no need to try all of them. Focus your efforts on one approach.
- Since the domain knowledge provided is relatively weak, you should not expect shocking increases in accuracy. Although at least a few percent increase in accuracy could be expected, the focus for this task is not to get the highest possible accuracy, but rather to come up with a conceptually fitting idea for a non-standard problem and to execute that idea.

## 2 Deliverables

The submission of this assignment is done in ANS. There are two deliverables:

- Implementation of the solution
- Description and explanation of your approach, formatted as answers to the questions in ANS.

For the implementation, a skeleton Jupyter Notebook with functions that can load the provided data for each task are provided. The notebooks also contain short bits of self-explanatory code on how the data is formatted. Please submit your code in the provided skeleton Jupyter notebook. You need to upload the Jupyter Notebook code (.ipynb file extension) and a PDF version (.pdf file extension) with the execution output. The maximum upload size is 25MB. So, you may need to clean some image output from the code Notebook before uploading. If you are using Google Colab, you can generate the PDF by going to "File → Print → Save as PDF". Please generate the PDF after running all cells of your solution (with possibly images removed if necessary for file size constraints).

Please note that you should train all models from scratch for the assignments in this course. The use of pre-trained models is not allowed.