Activity 6: Predicting Languages from Names

<https://pytorch.org/tutorials/intermediate/chor_rnn_classification_tutorial.html>

Example: What are in the data/names directory?

<https://pytorch.org/tutorials/intermediate/chor_rnn_classification_tutorial.html>
<https://download.pytorch.org/tutorial/data.zip>

The data includes 18 text files named as "[Language].txt". Each file contains a bunch of names in the form of Unicode (we need to convert from Unicode to ASCII that represents 52 English alphabets--26 lower-case and 26 capital letters)

We will have a dictionary of lists of names per language.
The 18 languages (n_categories = 18) are:

Arabic (2000), Chinese (268), Czech (519), Dutch (297), English (3668), French (277), German (724), Greek (203), Irish (232), Italian (709), Japanese (991), Korean (94), Polish (139), Portuguese (74), Russian (9408), Scottish (100), Spanish (298), Vietnamese (73) (The numbers in the parentheses are the numbers of names)

There are a total of 20074 names. All of them will be used for training
First 5 names of Italian: ['Abandonato', 'Abatangelo', 'Abatantuono', 'Abate', 'Abategiovanni']

Example: Objective and Process

Objective: We wish to design a classifier that outputs a "language" for a given input "name"

Process:

Step 1: Convert each name into a "n_letters x 57" matrix where n-letters is the number of letters of the name and each row is a one-hot vector (1 X 57) that can represent one of English alphabets (52) and 5 marks including "space (blank)", "period", "comma", "semi-colon", and "apostrophe".

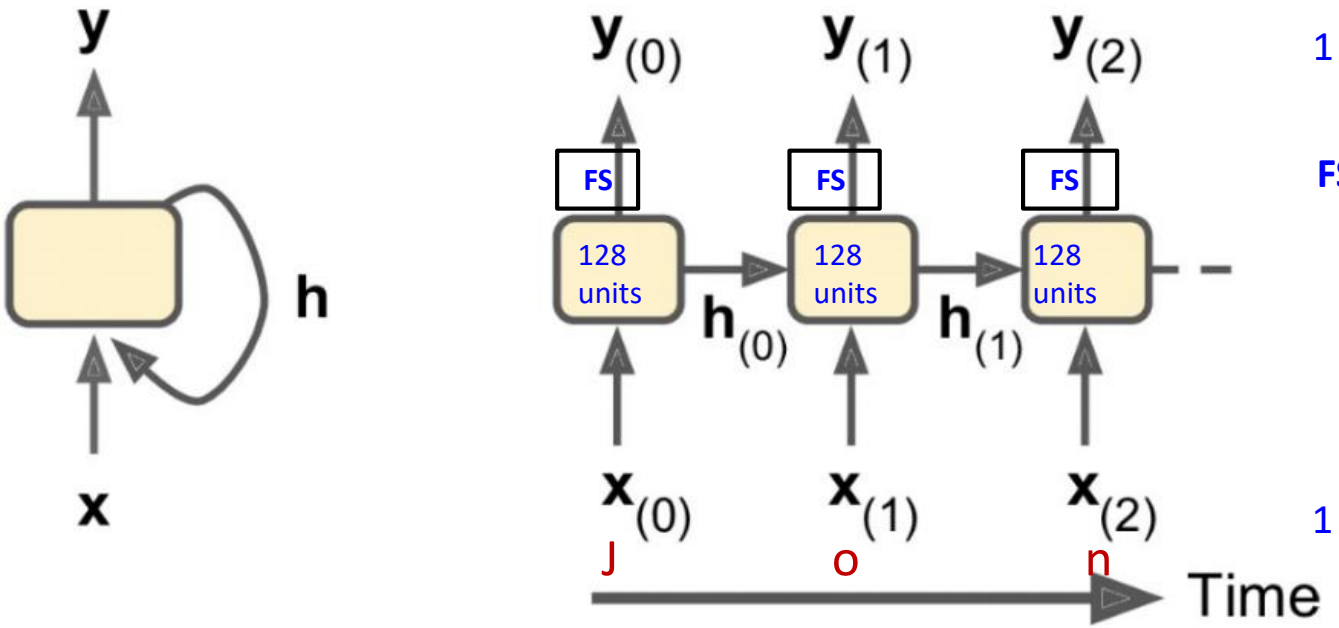
Example: The name "Jones" results in 5 x 57 matrix whose 1st row is given by

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,   Lower-case letters
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,   Capital letters
0, 0, 0, 0, 0]   Marks

Step 2: Using the input matrices representing names and the corresponding indices of the languages (categories), train an RNN with one hidden layer with 128 units

KAIST

Example: RNN for Predicting Languages



1 x 18 category vectors

FS: Fully connected & Softmax

1 x 57 one-hot vectorS

Figure 14-3. A cell's hidden state and its output may be different

Exercise: The "Predicting languages from Names" problem can be solved via **DNN** that directly uses Names (Words) for the inputs instead of employing **RNN** with Letter inputs. Design DNN for this problem.