# Introduction to Deep Learning
# **Assignment 2:**
# Recurrent neural networks

November 2023

The goal of this task is to learn how to use encoder-decoder recurrent models. Specifically, we will be dealing with a sequence-to-sequence problem and trying to develop a neural network that can learn the principles behind simple arithmetic operations.

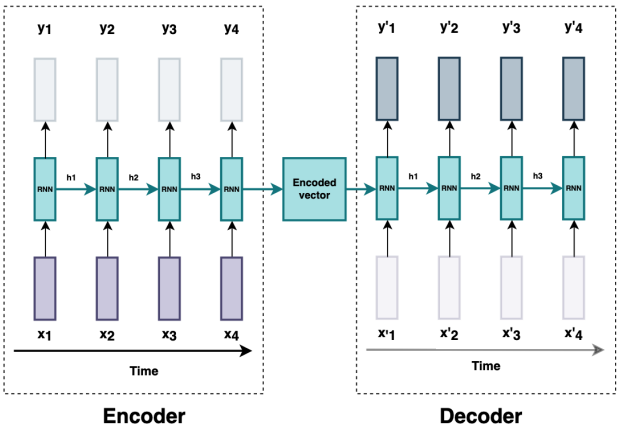


Figure 1: *An illustration of an encoder-decoder recurrent network model.*

**Description:** Let us suppose that we want to develop a neural network that learns how to add or subtract two integers that are at most two digits long. For example, given input strings of 5 characters: **'81+24'** or **'41-89'** that consist of 2 two-digit long integers and an operand between them, the network should return a sequence of 3 characters: **'105 '** or **'-48 '** that represent the result of their respective queries. Additionally, we want to build a model that generalizes well - if the network can extract the underlying principles behind the '+' and '-' operands and associated operations, it should not need too many training examples to generate valid answers to unseen queries. To represent such queries we need 13 unique characters: 10 for digits (0-9), 2 for the '+' and '-' operands and one for whitespaces ' ' used as padding.

The example above describes a text-to-text sequence mapping scenario. However, we can also use different modalities of data to represent our queries or answers. For that purpose, the MNIST handwritten digit dataset is going to be used again, however in a slightly different format. We can create an alternative image representation of our text queries described in the first paragraph by stacking multiple MNIST samples as follows:

Figure 2: *Converting text query '89+56' and its answer '145' to sequences of images using randomly selected samples from the MNIST dataset. The images are stacked along the time axis, resulting in two tensors of size [5, 28, 28] that are then sequentially fed into the recurrent models.*

We can then use these text and image queries/answers as our training data/labels and create different

variations of mappings between text and images - e.g. using text input to predict image output or vice versa. You are provided with a Jupyter notebook that contains functions for creating the image and text datasets along with sample *Keras* code for building a text-to-text RNN model.

## Part 1: Addition and subtraction

Your tasks are as follows:

1. Analyze the code for generating numerical and image queries and their respective answers from MNIST data. Inspect the provided text-to-text RNN model and try to understand the dimensionality of the inputs and output tensors as well as how they are encoded/decoded (one-hot format).

2. In the text-to-text scenario, we are not interested in "memorizing" the whole addition/subtraction table (about 20,000 strings in total); instead, we want the network to learn some general principles behind these arithmetic operations. Therefore you should try multiple different splits for your training and test sets (50% train - 50% test; 25% train - 75% test, 10% train, 90% test etc.) and evaluate the accuracy and generalization capability of your models. Compare the outputs of your trained networks to the true labels, and find out what kind of mistakes your models make on the misclassified samples. Visualize these differences and try to explain them.

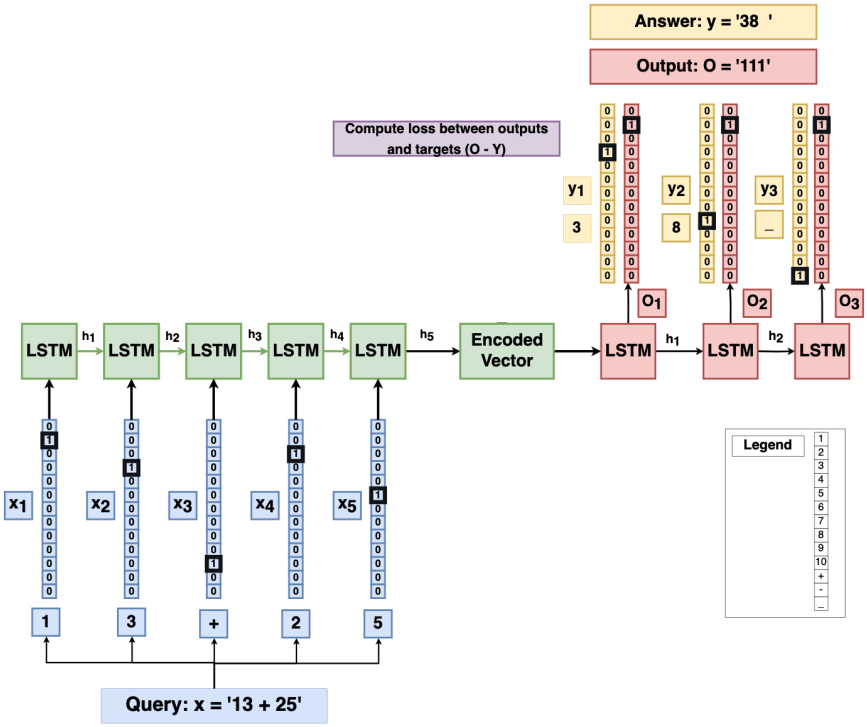


Figure 3: *LSTM Model visualisation using the arithmetic queries as an example for inputs/outputs.*

3. Create an image-to-text RNN model: given a sequence of MNIST images that represent a query of an arithmetic operation, your model should return the answer in text format. Once you have trained your model, evaluate its accuracy and compare it to the text-to-text model.

4. Build a text-to-image RNN model: given a text query, your network should generate a sequence of images that represent the correct answer. In this case, it is harder to evaluate the performance of your model qualitatively. However, you should provide examples of the output generated by your model in the report. What can you say about the appearance of these generated images?

    - *Optional: train a separate supervised model for evaluating the generated images of your text-to-image model.*

5. Try adding additional LSTM layers to your encoder networks and see how the performance of your models changes. Try to explain these performance differences in the context of the mistakes that your network was making before. *Tip: you should add a flag* ***"return_sequences=True"*** *to the first recurrent layer of your network.*

## Part 2: Multiplication

Addition and subtraction are relatively simple arithmetic operations but we can make this problem a bit more difficult by attempting to make our networks do multiplication instead.

1. Now try building models around the multiplication dataset (last cell of the notebook) - one that contains all combinations of two-digit integer multiplications (10,000 samples).

2. Create text-to-text and image-to-text models for the multiplication problem. How do the generalization capabilities change compared to **Task 1**?

3. Apply what you learned in the previous tasks and try to tune the architecture/hyperparameters of your models further. What is the highest accuracy on the test set that you can achieve and what kind of train/test sample ratio can you use?

***Deliverables****: Your submission should consist of a report in pdf format (at most 10 pages) and neatly organized code (Jupyter notebooks) so that we can reproduce your results. You also need to submit .py files of all the notebooks (by converting .ipynb to .py). You can find more information about report writing on the Brightspace → Assignments section.*

**Deadline: 08/12/23, 23:59.**