ERHVERVS
AKADEMI
AARHUS

Artificial Intelligence & Automation F2022.

Sila. May 25th, 2022.

Mandatory Exam Assignment 2. Spring 2022.

Complete as many of the following exercises as you are able to.
Credit will be given for the efficiency and effectiveness of your solutions and for clear written explanations of the solutions and (important) your own thinking in making the solutions, including the decisions, limitations and interpretations you have made in order to come up with your solution. You must include screen dumps and/or illustrations in your report. So, that it is possible to understand the intended functionality, even if it is (for some reason) difficult to run the handed in scripts. It is your responsibility to find a reasonable level for these illustrations/screen dumps. You must hand-in individuel reports, group hand-ins are not allowed in this exam. Also remember that this is an exam assignment, and it is therefore, as standard for exam assignments, not allowed to make your solution publicly available on the internet, neither before or after the handin Your hand-in can be written in danish or in english. You can do the exercises in any order you want – they are not dependent on each other.
Your report must be handed in on ***Wiseflow on Sunday the 5th of June 2022 at 20.00 O'clock*** *at the latest.*
If the report consists of multiple files (report, scripts) then it must be zipped into one file.

Exercise 1.

In this exercise you should make wordclouds based on 2 different texts (you decide which texts you will use).

The texts can be entered into your program as a string in the Python code (using triple quotes in Python).

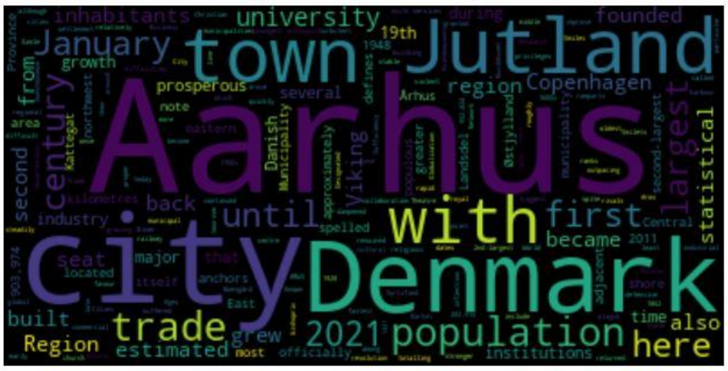
```
word_string='''This is my text...'''
```

Or, you can read the texts from a file (if you prefer that).

The Python library nltk can be used to find words in a given string:

```
words = nltk.tokenize.word_tokenize(word_string)
```

A wordcloud for the Aarhus text on Wikipedia will probably look something like this:

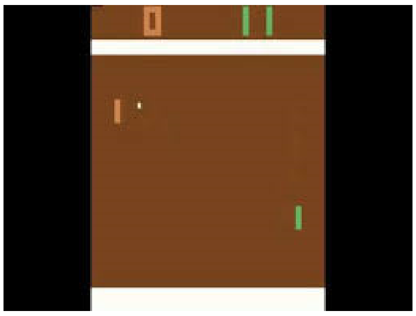


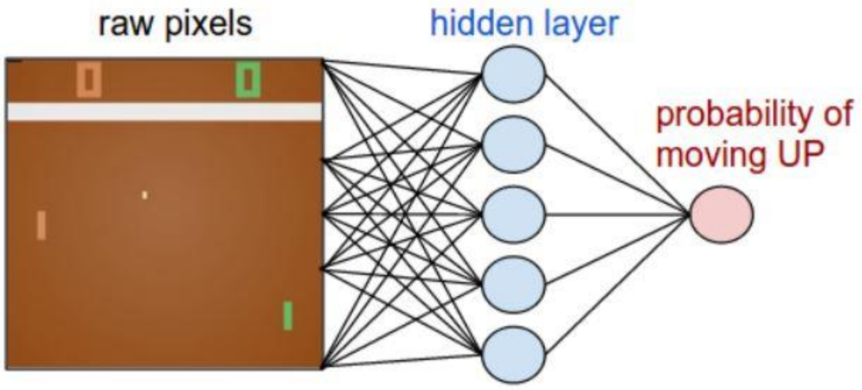
Explain what you find for the 2 texts, that you have decided to work on. And attach the Python code that you have used (Find inspiration in code, on Canvas, for week 10-11).

Exercise 2.

Earlier, in the course we have looked at the game of Pong.

Here, a neural network will take input in the form of a screen shot. And use such screen shots to calculate a best move.





credits: http://karpathy.github.io/2016/05/31/rl/

Getting started:
In order to play Pong, you should install the Gym libraries that are able to simulate playing Pong.

For exact instructions, Google"how-to". Probably:
pip install gym[atari,accept-rom-license]==0.21.0
or similar, will be sufficient.

a)
Based on what you have seen earlier in this course, pick a Python program that can learn how to play Pong using a neural net, or design your own neural net that can learn how to play Pong.

Explain the design of the neural net used in the program. How many hidden layers. What is the input and what is the output? Specifically, explain what the neural is looking at (the data input) in order to learn the game of Pong.

If you have time you might want to try out different designs, and compare the different solutions.

b)
Explain how rewards are used to update the weights of the neural net (in the solution that you have picked, or designed yourself). I.e. show relevant code lines, and explain in a headline form how the update works (That is: 15 - 30 lines of description will be sufficient).

c)
In the context of reinforcement learning, explain what "experience replay" and DQN "target networks" are, and why these techniques (sometimes) are helpful. Again, your answer should be max. 15 -30 lines long.

Extra, if you have time:

Based on a pretrained net (for python program you have picked), or a net you have trained yourself, show runs from a pong playing program (That is: output with training results, from a neural net that wins at least some of the games...). Explain what you see in the output.

Further: It is also possible to make gifs or video that show how your program is playing. So, animated gif's or video (attached to your report) will, obviously, also be a good illustration of how the program is doing.

Where pretrained weights might be stored in a file (say)

"`my_model_weights.h5`"

And where the weights could then be loaded into your model with the command:

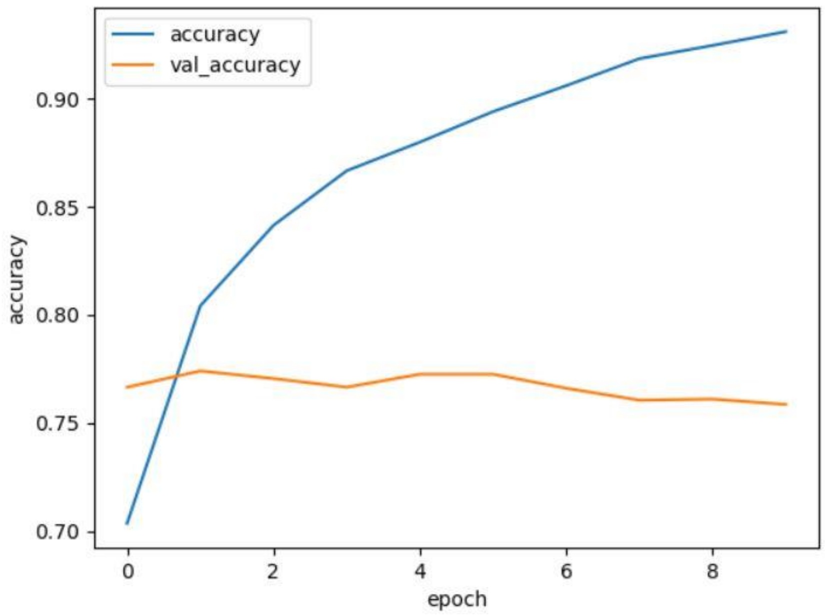
```
model.load_weights('my_model_weights.h5')
```

Indeed, pretrained weights could very well be helpful, as training a neural net to play pong can be rather slow. Runs for 30 hours of training and 10,000 episodes are sometimes needed, before you are on to something that actually plays well.

Exercise 3.

In natural language processing we are often interested in doing sentiment analysis.

In this exercise we will make a sentiment analysis on the twitter samples included in the NLTK package, Using a neural net, a straight-forward approach gives the following results on the training and test set:



The code is given in the file Tweet_Sentiment_Keras_Exam.py (See Canvas. This version replaces Tweet_Sentiment_Keras.py, week 12 also on Canvas).

a) Can you improve the results for the validation set? By training for more epochs? By making changes to the design of neural net? What neural net design changes will you suggest?
b) Other than looking at the neural net, there are probably many other changes to the program that could give us a performance improvement? Without coding it (if you want you can code it of course, but it is not assumed here) do you have any suggestions for what we could do next in in order to further improve the results (Give 1-3 suggestions).
c) In the program, you have been given, it rates two movie reviews (!) as being either positive or negative (look in the code for this).
What do you find with your own test examples (3-5)? Do you agree with what the program finds?

Exercise 4.

Earlier in this course we looked at Imdb reviews, and used a neural net to extract sentiment information from these reviews. In this exercise we will try to improve our results on the Imdb dataset.

The IMDB sentiment dataset can be found here:
http://ai.stanford.edu/~amaas/data/sentiment
(A version, aclImdb, is also available on Canvas).

We will again investigate whether we can teach a neural net to understand what sentiments are expressed in the reviews.

In the course you were asked to understand how the “Bag Of Word” model is used in the code. Where we first see that each review is re-casted as a BOW input vector. You were asked to locate that in the code, and make sure that you understood what was going on by making print statements etc. in the code. We take this initial part, from the course, as our starting point in this exercise.

See SentimentExam.py (On Canvas).

Questions:

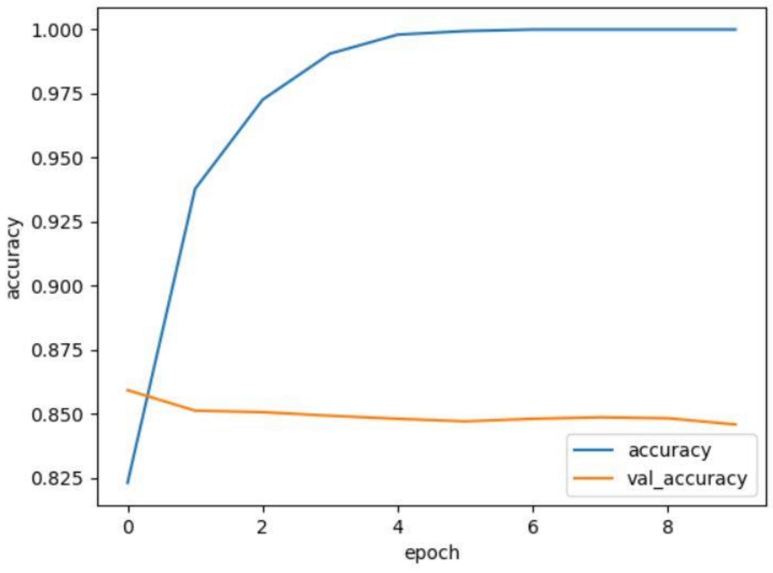
a) First, we used a pretty simple neural model, and we only trained for 10 epochs. Sticking with the simple model, does the result improve if you train for say 100 epochs? Are there other small adjustments we can try? What did you do?

   Explain, and attach plots.

b) Try to make the initial model predict a couple of new reviews, that you provide. What are your thoughts on what you see, when you give the model your own examples?

c) For such a simple initial model, the general results are certainly ok. If not super impressive. But maybe we could improve the results by introducing a hidden layer in our neural model?

   Run the next part in the given code (SentimentExam.py) with a hidden layer in the model. You should get something like this:



   What is happening here (with this introduction of a hidden layer, and the architecture as stated, in the code?) Explain in your own words (and based on your experiments).

d) Lets assume that you came to the conclusion that your model overfitted...

   What can we then do? Explain?

   What hyper-parameters would you suggest we try to change, what does your experiments show?

   In the next experiment, in the given code, we try dropout and regularization, that both “trim” the neural net by trying to make it smaller.

   Would you recommend using dropout, and/or regularization? What values for numbers of neurons, N, in your net do think gives the best results? Different activation functions, more hidden layers? What does your experiments indicate?

   Include plots along with your argumentation.

e) (In the code, SentimentExam.py) We don’t use the whole data set. But more data is usually better! Try one or two experiments with the more data (Probably best if you can try the whole dataset, but well…). Is more data the thing that will give you a few percentage more in accuracy? What do you find?

   (Btw. In week 12 we looked at sentiment analysis that used a, simple, Bert model. A similar approach might also be helpful in connection with our code here. But outside the scope of this exercise).

Exercise 5

NLP.

In this exercise we will look at a Wikipedia text about Denmark (included in the file NLPAnalyze.py on Canvas. Install models with: python -m spacy download en_core_web_sm, in cmd prompt).

a) In the given text, what are the 5 most frequently used words?

   (No coding needed, just look in the code, and do not count non-words like “.” and “.”)

b) (Looking at the “parts of speech”, POS) in the original text, how many nouns and how many adjectives do we have in the text? Here you need to code a little, but not much.

c) The library gensim (gensim_sum_ext) can be used to do to text summarization. If we use a ratio setting setting of 0.2 what does the text then look like? With ratio = 0.1?

   Use the 0.1 value, and Spacy, to give the 5 most frequent used words in the summarized text (Notice: Gensim Summarizaton is available in Gensim vers. 3.6. So, use: *pip install gensim*==3.8.3 or similar).

d) Numbers are usually a good way to describe facts. What numbers can you find in the text? Included in what sentences?

   Are all the numbers of equal importance? Probably not. So, what could we do to find the important numbers (those that best describe DK). Write down your ideas, and try to code it.