CMME139

Introduction to Deep Learning

Assignment #4

Due Date: 14/06/2023 Wednesday (23:59)

Structure: Individual Assignment

Score Contribution: 30%, evaluated over 30 points.

This is an individual assignment.

There is no code submission for this assignment.

There will be a report submission for this assignment.

Objectives

- **To develop a deep neural network solution for a novel problem**
- **To have experience with convolutional neural networks**

Overview

In assignment #1, we have collected game icons from the 17K Mobile Games dataset that we have used in the course. **Your objective in this assignment is to select and use 400 game icons (images) to build a convolutional neural network (CNN) to answer a binary classification problem**.

In the assignment, you will start by selecting the top 200 and bottom 200 games with highest and lowest User Rating Counts and work with their icon images (You can make different selections from highest and lowest user rating segments. How you make the selection is up to you, but the report should contain that detail). The report should contain:

- A Title page
- Your data selection process, shortly (i.e., details such as whether you selected the games before or after imputing the Average User Rating Attribute)
- How you process each image
- The structure of the image as an input to your neural network (i.e., whether it is scaled or not, whether it is converted to grayscale or not). Note that, you are free to do any kind of processing on the images
- The structure of your Neural Network and its compilation parameters.
- Summary of your experiments
- Results of your experiments

In this assignment, we will adopt a different approach than a task-based structure. Here, we will describe the steps that needs to be taken in order to successfully finish the assignment along with several suggestions; and leave the rest to you.

Don't forget, you have Workshop examples and examples from the book that already dealt with images before. I strongly suggest you examine them and use them as a basis for this assignment:

- **The objective of this assignment is to examine whether there is a correlation between game iconography and user rating count**. The objective function to predict for this assignment is binary: **User Rating Count being high or low**. The problem we are aiming to solve is whether game iconography has predictable markers that can be used to make a User Rating Count prediction or not. In this regard, you will sort the dataset and select the top 200 games with Highest Rating Counts, and bottom 200 games with Lowest User Rating Counts. (**Suggestion:** you should also create an objective vector of size 400 (total number of games in our dataset for this problem), representing each games' class – low or high -).
- **Download the images, display the images, and first develop an expectation**. Can you see some identifiable feature of top 200 and bottom 200 games? (**Suggestion:** Not everything is predictable. And you cannot fine-tune a neural network to predict something that is not available. First, establish your expectations. The results may positively surprise you and that would be a finding of your study, and thereon you can fine-tune the network to get the best out of it. But it is always a good idea to develop an initial expectation so that you don't waste time on redundant fine-tuning.)
- Note that the images for game icons are of very high quality and working with such large images usually require a lot of time (And most of the time, a lower resolution image would also do relatively well in a CNN task). Hence, we strongly recommend you rescale/resize the images and work with rescaled versions. (**Suggestion:** The **EBImage** library contains an easy-to-use resize function for this purpose). You can perform other preprocessing operations on images (at the cost of completeness), such as converting the image to a grayscale variant in case building a neural network for 3-channel images become a difficult task (a working code with grayscale images is better than a not-working code with color images). (**Suggestion:** **Imager** library is a professional computer vision library that has such conversion functions)
- During your processing of the images, always be mindful of the dimensions of your image data. Keras neural networks will require you to convert your image dataset into an **Array** (or Large Array, as Rstudio sometimes refers to it, depending on the size of it.) **of size (M x N x N x 3)** assuming you will deal with color images where, M is the number of instances (images) in your data and N is the resolution of the image. As an illustrative example, **suppose we will train our neural network with 300 images and each image is a color image of 64 by 64. In this case, each image is represented as a 64 x 64 x 3 array, and a valid input for Keras for your training data is a 300 x 64 x 64 x 3 array**. (**Suggestion:** **array_reshape** and **aperm** are some functions that can be useful for you at this stage to manipulate the orders of your dimensions in an array. Read through their documentation first though: Although they look similar, their results are very different)

- Create your training and testing sets. Note that you can do this in this order, or you can do this earlier. Select whichever is more comfortable for you.
- Build a sequential neural network, just like the other examples in the course. The input shape of the neural network should be (N x N x C) where, N is the resolution of the image and C is the number of channels (3 for color, 1 for grayscale). **For building the neural network, use the following 5 layer options only**:
    - Layer_conv_2d (Convolutional layer)
    - Layer_max_pooling (Pooling Layer)
    - Layer_flatten (For image layer-to-dense-layer conversion/preparation)
    - Layer_dense (Basic Building block of all neural networks)
    - Layer_dropout (For overfitting conditions)

  Although we have mentioned that Layer_conv_3d is preferrable for color images in the lecture videos, don't use them: They add a very high complexity to the data structure, and would not benefit the model that much (**Summary:** It is not worth your effort)

- Make several experiments with your neural network and its parametrization and report your findings. In this regard, your report should:
    - Show that you have experimented with at least 3 different CNN builds (network structures)
    - Show that you have experimented with different number of filters, kernel sizes, pool sizes.
    - Rest of the parametrization is up to you, as you already know their possible effects from the previous assignments.

What to Submit

This is a report-based assignment. You should prepare a report, containing a cover page in pdf form and submit the report. There is no need for code submission.

What will be graded

- Report Quality
- Presentation/Clarity of the Results
- Presentation/Clarity of the Justifications
- Model Design
- Correctness of the Experimental Design

Some Tips and Tricks

Although you will not be sharing the code, your report should describe your workflow in this small project clearly and sufficiently. Additionally, your report should also mention the neural network structure and compilation parameters along with your findings.

Do not over-experiment on it. Not everything is predictable, and this is surely not a sanitary dataset. If you would like to see a more sanitary example, I would like to share a secondary dataset for you to try. This dataset contains dandelion (a big flower, something very very obvious) and grass images.

https://www.dropbox.com/s/bg5m5f4a7kt6367/flower_vs_grass.zip?dl=0

(Note that this dataset is not shared as a part of the assignment. It is shared so that you can experiment at your own leisure and feel more confident.)

Good Luck!

DL TEAM