General Explanation of System and its Goals

For the second part of this coursework, the chatbots theme has been adapted to cover movies as well due to the difficulty of using a TV show related dataset. The classification of the images are movie posters and their respective genres. The aims for this section are to be able to create an image dataset, and then use it to train a convolutional neural network. This can then be saved and exported to the chatbot, for it to be used to predict the classification for new posters. The chatbot should be able to pick a random image from a folder of sample images, or allow the user themselves to pick an image, either of which can be used in predicting its genre.

System Requirements

* An image dataset needs to be created or sourced in order to train the CNN model.
* CNN model should be produced and saved in the .h5 format externally from the chatbot program to be imported in.
* Should be able to load in the model that can then be used for image classification.
* Program should be able to pick a random movie poster from the sample image on command, or similarly allow the user to select a movie poster.
* Chatbot should be able to recognise the new commands for image selection and classification using AIML.

Employed AI Techniques

**Deep Image Classification:** This involves using a Convolutional Neural Network trained using a dataset (in this case with images). Before being used for training, the dataset needs to be pre-processed. This involves resizing, assigning a class, and splitting the dataset into training and validation. This is to help analyse the accuracy of the model once it has been trained. Additionally, the dataset can be autotuned to help with performance. Next, the dataset must be normalised to as the values need to be between 0 and 1. Finally, the dataset needs to be augmented to counteract overfitting. This adds additional images created from pre-existing ones through rotations, flips and zooms. Then, the model can be created that is sequential and uses convolution blocks with max pooling layers. The model can then be compiled, and then can optionally return a summary of the model. The model is then trained using the fit function, that passes the training data, validation data, and the number of epochs. Lastly, the results of the training can be plotted on a graph to help determine how effective it was.

Program Explanation

CNN\_MoviePosters

Program starts by defining some variables including batch size, image height and width along with the path for the dataset images.

Then the dataset is pre-processed using keras. The data is split into training and validation at 80%-20% ratio. The data is also resized here to ensure there is a standard size of image. Batch size is also defined. The class names are also obtained from the training dataset.

The dataset can be autotuned to improve performance. This is done via precaching. The dataset is also normalised to convert the RGB values to be between 0 and 1, as the model cannot process 0 to 255. The data can optionally be augmented to counteract overfitting. This creates more images from others already in the dataset through rotating, flipping and zooming.

The model is then created using the sequential function. The model features multiple convolution blocks and max pooling layers, along with dropout, flatten and dense layers. The model is then compiled and summarised. The model can then finally be trained using the validation data, training data and the number of epochs, which in this case is 30.

I have also included code to check the accuracy and loss of the model through training to determine how effective the training has been and to check for overfitting.

myTVBot

displayImage(file) was added to the program as multiple points in the program require images to be displayed. The function passes the file as a parameter and outputs using the PIL module. This was found to be more reliable than using Matplotlib for this purpose.

Additionally, indexes added in the main function for the extra functionality of the program. The first, at index 3, is used to get a random image from the sample images folder and display it for the user, while also setting that as the “currentPoster”.

Index 4 uses tkinter to open a file dialog box for the user to select a file themselves. This file then is set as the “currentPoster”.

Index 5 adds the image classification component to the program. It defines the class names, before loading the model, pre-processing the “currentPoster” and converting it to an array. The image class is then predicted using the model, along with the percentage confidence of the prediction.