Cookie - Virtual Assistant

Authors: Aarush Pal, Deepesh Garg, Mihir Agarwal, Saatvik Maheshwari
Institute: Manipal Institute of Technology, Manipal

aarush.pal@learner.manipal.edu, deepesh.garg1@learner.manipal.edu,
mihir.agarwal@learner.manipal.edu, saatvik.maheshwari@learner.manipal.edu

Abstract --- This Project demonstrates the building of a virtual assistant which can perform the following operations - Google Search, launching applications, display weather, time and date, extract news headlines and send a WhatsApp message. The virtual assistant is accessible via Facial Verification and recognises the required action to be carried out through a Speech Recognition Mechanism and Natural Language Processing. The Facial Verification aspect has been implemented primarily using the OpenCV library and adopting the VGG - 16 architecture. The model is a Convolutional Neural Network trained using the Tensorflow and Keras frameworks. The text classification of the commands by the user is realised through Natural Language Processing by using NLTK (Natural Language Tool Kit). The model is a Bi-directional LSTM (Long Short Term Memory) trained using Transfer Learning from a model trained on a news' article dataset.

Keywords - Virtual Assistant, Facial Verification, Convolutional Neural Network, VGG-16, NLTK, Natural Language Processing, Bi-directional LSTM, Tensorflow, Keras, OpenCV.

1. INTRODUCTION

Technology in recent years has led to the automation of various mundane tasks performed by humans. One such breakthrough is the invention of Virtual Assistants. Virtual Assistants have become a significant part of our daily lives, be it Apple's Siri, Google Assistant, Microsoft's Cortana or Samsung's Bixby. "Cookie" is a versatile Virtual Assistant performing essential tasks for daily use. Have you ever wanted an easy way to send WhatsApp messages, check the daily news, get to know what the time, date and how the weather is? "Cookie" got you covered.

2. METHODOLOGY

A. Facial Recognition model

The project begins with a model using, OpenCV which takes up 100 images of the user and stores them in the system to be a part of the Facial Recognition dataset. The dataset was divided into training and testing data manually by the authors. Facial Recognition is implemented using the Haar Cascade face classifier. The CNN model was trained on the images of the four authors. “Cookie” implements transfer learning and uses the pre-trained VGG - 16 CNN architecture. The network has 16 layers which are described as follows –

→2 x convolution layer of 64 channels of 3x3 kernel and same padding
→ 1 x max pool layer of 2x2 pool size and stride 2x2
→ 2 x convolution layer of 128 channels of 3x3 kernel and same padding
→ 1 x max pool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 256 channels of 3x3 kernel and same padding
→ 1 x max pool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 512 channels of 3x3 kernel and same padding
→ 1 x max pool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 512 channels of 3x3 kernel and same padding
→ 1 x max pool layer of 2x2 pool size and stride 2x2

The model uses categorical cross entropy as loss and Adam as the optimiser. The metric used to evaluate the model is Accuracy. The model uses Data Augmentation to scale up the dataset.
The next part of the model loads the trained weights from the training model through a .h5 file. The webcam captures individual frames in a VideoCapture object and returns the cropped faces if a face is detected, else it returns the input image. After a face is detected, the model matches the detected faces with the faces in the dataset. If the probability exceeds the threshold value given, then the user gets authenticated and moves on towards the next step.

B. Text classification model:

1) Data pre-processing:

The "stopwords" were downloaded from NLTK. Stored the content/sentences in a list and removed the "stopwords" from each of the sentences. Stored the output labels in a list.
Split the dataset into train and development/validation sets in the 80:20 ratio. Fit the tokeniser on the base model dataset to get the word indexes and convert the sentences to sequences based on these indexes, padded each of the sequences to fit the input layer of the model which will be defined ahead. Performed the same preprocessing on labels to obtain the label indexes. The above mentioned data-preprocessing has been applied to both the base model dataset (News articles) and the final custom dataset.

2) Base model architecture:

The embedding layer is the input layer having a vocabulary size of 30000, embedding dimensions are 32, and the input length of the sequences is 128. The next layer is a bi-directional LSTM of 32 units. Following this is a dense or fully-connected layer consisting of 64 neurons which are fully connected to another layer having 32 neurons. The final layer is a softmax layer with 6 outputs (The output at the zeroth index is ignored), classifying the news articles into 'business', 'entertainment', 'politics', 'sport', and 'tech'. The Adam optimisation algorithm has been used to minimise the cost. The loss used is sparse categorical cross-entropy considering the fact that the classes were mutually exclusive.
All but the last layers of the base model are saved to use in the final model.

3) Final model Training:

The saved base model layers are loaded, and a 9 unit softmax layer (The output at the zeroth index is again ignored) has been added. This model is trained further on the custom dataset built specifically for the virtual assistant. The loss used here is again sparse categorical cross-entropy.
We have saved the final model as a .h5 file and loaded it in the final code for the assistant while integrating with the facial verification model.

C. Features of virtual assistant:

Google Search - The model searches the command on Google if prompted for the former as well as in the case of a fallback. We used the Google Search API to implement the feature.
Weather - Implemented the Weather function using the Google Weather API; the model asks for the location and returns the day, time and weather conditions.
Date and Time - The model returns the date and the time of Indian Standard Time(IST) using the Python DateTime library.
News - The model uses the Google News API to extract the top headlines and returns the same.
Launch App - On prompting the model to launch an app, it returns a menu with the names of the apps and asks for a speech input for the app that the user wants to launch. The launch function is developed using the Python os library.
Send Whatsapp message - The model uses the Python pywhatkit library to send a WhatsApp message. It opens up web.whatsapp.com in the default browser and gets the message to the desired phone number in Indian Country Code.

D. Deployment:

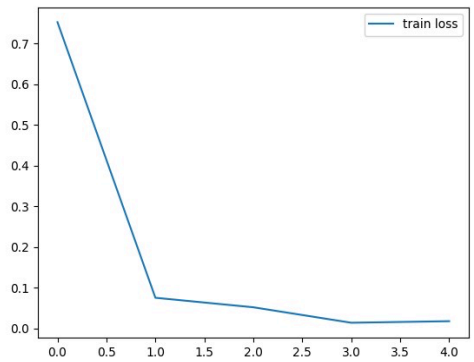
The platform used is Streamlit. A minimalistic User Interface has been built using the streamlit library functions and deployed locally.

3. CONCLUSION

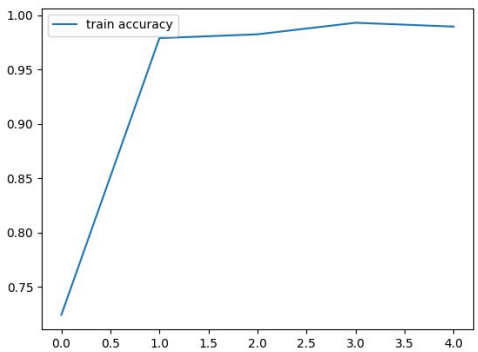
The project demonstrates the building of a virtual assistant which works by means of facial verification and can classify voice commands to a certain set of functions it can execute and use google search in case of fallback.

A. Face Verification model:

Loss:

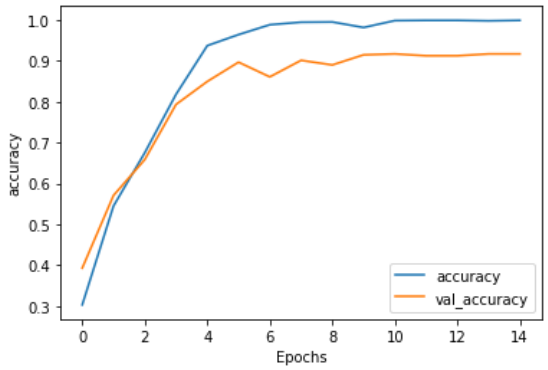


Accuracy:

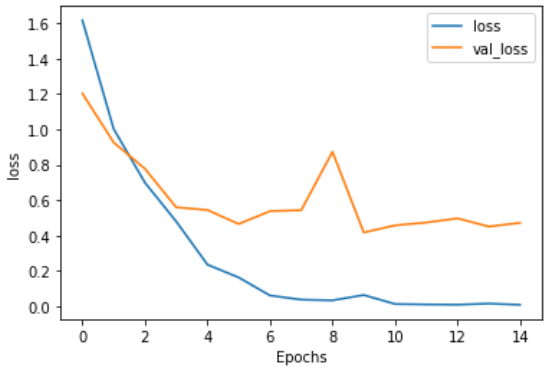


B. Base model for text classification:

Accuracy:

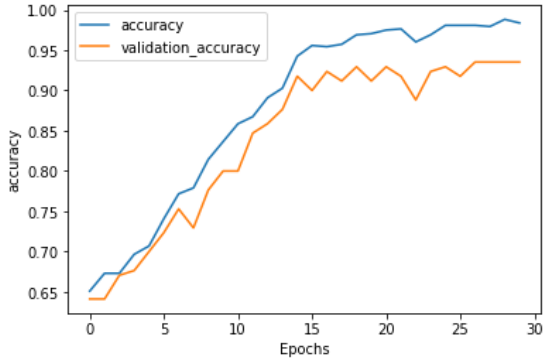


Loss:

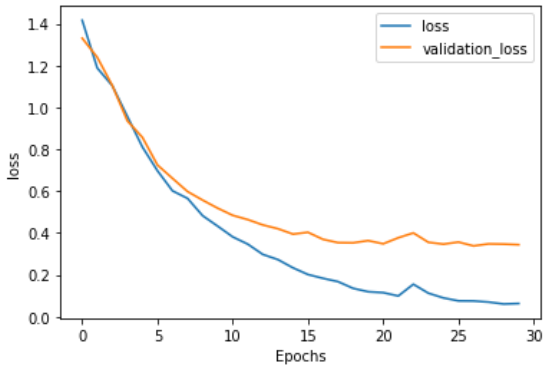


C. Final model for text classification:

Accuracy:



Loss:



Possible improvements: Addition of more number of classes for opening individual applications. Furthermore, more features could be integrated for expanding the scope of the assistant. Different neural network architectures could be tried to improve accuracy further.

4. REFERENCES

https://www.kaggle.com/hgultekin/bbcnewsarchive

https://www.kaggle.com/bouweceunen/smart-home-commands-dataset

https://realpython.com/python-speech-recognition/

https://www.google.co.in/amp/s/www.geeksforgeeks.org/performing-google-search-using-python-code/amp/

https://pypi.org/project/GoogleNews/

https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

https://www.youtube.com/user/krishnaik06

https://pypi.org/project/python-weather/

https://pypi.org/project/pywhatkit/

https://docs.python.org/3/library/datetime.html

https://opencv24-python-tutorials.readthedocs.io/en/latest/

https://www.tensorflow.org/tutorials/images/cnn

https://www.tensorflow.org/hub/tutorials/tf2_text_classification

https://arxiv.org/pdf/1409.1556.pdf

https://docs.streamlit.io/en/stable/getting_started.html#share-your-app