Restoration and preservation of sharada scripture

Transliteration System

● A transliteration pipeline was implemented in python.
● The pipeline allows transliteration from Sharada to Devanagari , IAST & ITRANS
● A dictionary with character mappings from Sharada -> Devanagari , Devanagari -> IAST + ITRANS , Devanagari -> Sharada , IAST -> Devanagari were created.
● A transliteration algorithm was written taking into consideration the presence of composite / compound characters in sanskrit.

Creation of a annotated dataset

● Multiple scriptures were annotated manually using LabelMe
● All the characters starting with each of the 34 consonants क, ख, ग, घ , ... श, ष, स, ह, ळ were added into into 34 different groups.
● Group 1 contains क, का, कि, की, ..., कं, कः, and all compound letters that start with क are included in this group. क्क, क्खी, क्या, क्ले, क्लीं, क्षी, etc. all come in this group.
● All Vowel letters - अ, आ, इ, ई, ... अं, अः, ॐ are placed in Group 34.

● All punctuation marks are placed in Group 35.

● All digits 0, 1, 2, ...9 and other numbers are placed in Group 36.

● Group 37 is reserved for all the letters which do not belong to the previous groups.

● The JSON files were then used to extract the character image based on whether it was a rectangle or a polygon. Multiple scriptures have letters which are found at a tilted angle or have obstructions with the adjacent characters and hence there was a requirement of marking the characters with the help of a polygon. PIL was mainly used to perform the extraction of characters and in the end we were left with **10333** characters.
● The dataset is maintain in two formats

1. 7z archive : It consists of multiple subdirectories where each folder is provided the name of the label and images are stored inside the directory. The names of the images are in the format {label_name}_{rnd_string} -> the rnd_string consists of 15 characters with the label name before the underscore. This allows us to identify distinct characters.
2. Pickle file : The first column of the pickle file consists of the group id. The second column of the pickle file consists of a numpy array. Each element in the array represents a pixel value in an image. The array is composed of a sequence of

RGB (Red, Green, Blue) values, where each RGB value represents the intensity of the corresponding color channel for a particular pixel. The array has a shape of (height, width, 3), where height represents the number of rows or pixels vertically, width represents the number of columns or pixels horizontally, and 3 represents the three color channels (red, green, and blue) for each pixel. The third column consists of the label itself , where the image label is stored in IAST. If the pickle file is employed, it is typically used in conjunction with the transliterator in order to convert the IAST values to devanagari / sharada.

OCR

● Open CV was used to extract the characters while performing the OCR
● Skew correction : Multiple images which are available in the archive.org repository have varying skew angles. A function was written to correct the skew angles.
● Noise Removal and Skeletonization of the input image
● Otsu thresholding is applied to the input image and then the contours are used to perform the line segmentation
● Once the line segmentation is successfully performed the images are then sent to the character segmentation function. The same methodology as line segmentation is applied as line segmentation but the histograms are drawn using the vertical scale and then characters are extracted and sent to the deep-learning model

● The code utilizes the ResNet50 model, which is a pre-trained CNN, to perform OCR (Optical Character Recognition) classification.
● The ResNet50 model is used as a feature extractor to extract relevant features from the input images, and additional layers are added on top of the extracted features to perform the classification task
● An accuracy of 84% was achieved when the classification was performed.
● In multiple instances the model stopped after the 13th epoch due to the early stopping callback which was initialized before. Early stopping allows the training to stop if the validation loss does not improve after a certain number of epochs. The model weights were restored to the best performing state based on the validation loss, and the training process ended earlier than the specified 25 epochs.
● It can be difficult to determine how many epochs to cycle through to train a neural network. Overfitting will occur if you train the neural network for too many epochs, and the neural network will not perform well on new data, despite attaining a good accuracy on the training set. Overfitting occurs when a neural network is trained to the point that it begins to memorize rather than generalize.
● The initial model made use of 206 classes / labels and more labels will be added as the database is improved.

● Multiple compound characters are rarely seen in the scriptures. This results in an imbalance in the label which is also one of the reasons for the decreased accuracy and the increased chances of overfitting.

Complete pipeline

- The scanned image is sent to the main python script which then forwards it to the segmentation module where the line and character segmentation takes place.
- The characters which are segmented by OpenCV are placed in folders according to their line numbers. The characters are also given the name as per their position in the line.
- Then the characters are sent individually to the OCR classifier where they are classified and the group_id is then sent back to the main program. Here the program links the group_id with the original image while maintaining its position.
- Once all the ocr is performed on all the images , the main python programs reads the images iteratively while appending them to the main list.
- Characters which are recognizes are provided as output.