Language detection from images using CNNs

Domain Background

This project comes under **Computer Vision**. Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action.

This project particularly focuses on Image Classification. The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity.

Problem Statement

Motivation for this project actually came from a project at work. We have an application which lets the school going kids click an image of their homework question, and we provide them with a detailed solution. Now as you can imagine, this relies heavily on OCRs as we depend on the text it returns to search for answers in our database. But since the OCRs work really well on English but fail to return text for other regional languages, many of the questions asked by the students go unanswered.

Predominantly the images contain English, Hindi and Math questions. Google Vision is the OCR we use. For hindi images, the API worked better whenever I passed Hindi as a language hint. Now I know that Google Vision is not great with math, but it was just an extension of the idea to detect math images as well.

In order to get better results for Hindi images, I'll have to somehow detect the language before sending the images to Google Vision. That is when I thought of using CNNs to detect language.

As anyone starting out with an idea would do, I started googling stuff about language detection. That is when I found this really interesting paper
https://hal.archives-ouvertes.fr/hal-01282930/document

The method used in this paper involves three subtasks: writing type identification (printed/handwritten), script identification and language identification. The methods for the writing type recognition and the script discrimination are based on the analysis of the connected components while the language identification approach relies on a statistical text analysis, which requires a recognition engine.

Personally, I'd want to start with something much simpler and then build on it.

Dataset and Inputs

Input images are obtained from the images searched by the students. Dataset consists of around 200 images for each english, hindi and math categories. Test set consists of around 20 images for every category. They are put into different folders and the folders are used as their class labels.

The CNN requires all the images to be of the same size before being fed into them. Using PIL library of Python, I will reshape all the images into a common shape. Then every image will be converted to matrix using the imread function of PIL. All of these matrices are put into an numpy array. Then it is normalised by subtracting every value from the mean and dividing by standard deviation.

CNN is trained on these images and then it is tested against the 3 test folders.

Solution Statement

Solution would be to use Neural Nets to classify images into one of the categories. For image classification tasks, Convolutional Neural Networks perform really well. CNNs can directly work on volumes of data, thus preserving spatial information present in the images. We train the CNN for 150-200 epochs, after which it will be ready to make accurate predictions, I hope.

Benchmark Model

Since I couldn't find work related to language detection using CNNs (Most of the work follows the approach of writing type, script and language detection steps), I'd like to use a

simple Fully Connected Neural Network as my benchmark model. This should establish some sort of a lower bound for this language detection approach.

Evaluation Metrics

We will be monitoring loss and accuracy of the model. The model keeps updating its hyperparameters after every epoch. Ideally loss should decrease and accuracy should increase with every epoch.

Project Design

Reshaping Images:

Before the images are passed on to the CNN layers, it should be preprocessed. The input images that i've chosen for this task are of different shapes. That would require that images are reshaped to a common shape before being fed to the model.

Library Used:

I will be using Keras for this project.

The Design:

The design that I have thought for this project would consist of CNN with **4 sets** of convolutional layers. Each set consists of a Convolutional Layer, a Max Pooling layer and a Dropout regularization layer. These are followed by 2 Fully Connected layers. The last layer is responsible for predicting the output.

First Set consists of:

- A Convolutional Layer with 32 filters of size (3,3). Chosen activation function for this layer is 'relu'.
- A Max Pooling layer with pool size of (3,3) and with padding 'same'.
- Dropout regularization layer with a dropout rate of 0.25.

Second Set consists of:

- A Convolutional Layer with 32 filters of size (3,3). Chosen activation function for this layer is ‘relu’.
- A Max Pooling layer with pool size of (3,3) and with padding ‘same’.
- Dropout regularization layer with a dropout rate of 0.25.

Third Set consists of:

- A Convolutional Layer with 64 filters of size (3,3). Chosen activation function for this layer is ‘relu’.
- A Max Pooling layer with pool size of (3,3) and with padding ‘same’.
- Dropout regularization layer with a dropout rate of 0.25.

Fourth Set consists of:

- A Convolutional Layer with 128 filters of size (3,3). Chosen activation function for this layer is ‘relu’.
- A Max Pooling layer with pool size of (3,3) and with padding ‘same’.
- Dropout regularization layer with a dropout rate of 0.25.

These four sets are followed by a **Flatten** layer, because the output from the 4th set will be provided as the input to a Fully Connected layer.

Fully Connected Layer 1:

- This consists of 128 nodes and uses a ‘relu’ activation function.
- This is followed by Dropout regularization layer with a dropout rate of 0.5

Fully Connected Layer 2:

- This layer consists as many nodes as the number of output labels available, so in this case it consists of 3 nodes. This layer uses a ‘Softmax’ activation function. This layer is responsible for classifying the image into one of the three output labels.