ME644 : Machine Learning for Engineers

Assingment 2

Definition

- **CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart):** CAPTCHAs are popular ways of preventing bots from attempting to log on to systems by extensively searching the password space. In its traditional form, an image is given which contains a few characters (sometimes with some obfuscation thrown in). The challenge is to identify what those characters are and in what order. □
- **Hexadecimal number:** Hexadecimal numbers are written in base 16 i.e. there are 16 "digits" instead of the usual 10. These digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The last six digits have values going from 10 to 15 □

Objective

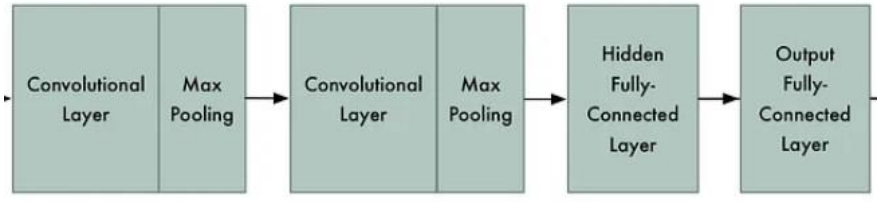
- **Aim:** To figure out whether the hexadecimal number in a given image is odd or even using Machine Learning. □
- **Given:**
  - Each CAPTCHA image in this assignment will be 500×100 pixels in size.
  - Each image will contain a code that is a 4 digit hexadecimal number.
  - The font of all these characters would be the same, as would be the font size.
  - The Latin characters A-F will always be in upper case.
  - Each character may be rotated (degree of rotation will always be either 0°, $\pm 10^{\circ}$, $\pm 20^{\circ}$, $\pm 30^{\circ}$) and each character may be rendered with a different color.
  - The background color of each image can also change. However, all background colors are light in shade.
  - Each image also has some stray lines in the background which are of varying thickness, varying color and of a shade darker than that of the background.

□

Solution

- **Model Description:**
  - **Parity (odd/even) depends on unit place:** In the base-16 number system, we can determine the parity of a number by simply looking at its unit digit. This is because when an even or odd number is multiplied by an even number, the result is always an even number. In base 16, every place value except the unit place is a multiple of 16, which means that it is always an even number. Therefore, when we add up these even numbers any number of times, the sum will always be even. Consequently, the parity of a base-16 number is solely determined by its unit digit.
  - **Training data and label:** The given CAPTCHA images contain 4 hexadecimal character and images are labeled as "EVEN" or "ODD".

    To process CAPTCHA images and extract individual hexadecimal characters, we employed the Python library called "PIL" (Python Imaging Library). Using this library, we successfully cropped the CAPTCHA images into separate character segments based on pixel range. Next, we undertook the task of manually sorting each individual character image into one of the 16 categories, based on its corresponding hexadecimal character. This meticulous sorting allowed us to create a dataset comprising single character training images, each labeled with its respective hexadecimal character.
  - **Modeling and prediction:** In order to predict the parity of CAPTCHA images, we employed a deep learning model known as a Convolutional Neural Network (CNN). The CNN architecture consisted of two convolutional layers followed by two fully-connected layers. We trained this model, which we saved as "model.h5", using the dataset comprised of single character images obtained from the given CAPTCHA images, along with their corresponding labels. When a preprocessed CAPTCHA image is fed into this model, it produces an output number ranging from 0 to 15 (both inclusive). This output number is then utilized to predict the parity of the given CAPTCHA image.
- **Training Algo Description:**
  - **Preprocessing:**The image is labeled to identify the correct hexadecimal character it represents. It is reshaped to ensure a consistent size and maintain the aspect ratio. It is also converted to grayscale, to removing the impact of varying background and font colors. By converting the image to grayscale, we can focus solely on the structural features of the characters, disregarding color information.
  - **Model architecture:**



  - (Image Credit: Google)

* **Model code:**

```
# Define the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1)))
# Input shape updated for grayscale
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(len(characters), activation='softmax'))
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

* **Input layer:** The input shape is set to (100, 100, 1), indicating a height and width of 100 pixels with a single channel (grayscale).
* **Convolutional Layers:** The first convolutional layer has 32 filters with a kernel size of (3, 3). The second convolutional layer has 64 filters with a kernel size of (3, 3). Both uses the ReLU activation function to introduce non-linearity.
* **Max Pooling Layers:** After each convolutional layer, a MaxPooling2D layer is applied. It reduces the spatial dimensions and extracts important features.
* **Flatten layer:**The Flatten layer is used to convert the output of the convolutional layers into a 1-dimensional array.
* **Fully Connected Layers:** The first dense layer consists of 64 units and employs the ReLU activation function. The final dense layer has 16 units, which uses the softmax activation function for multi-class classification.
* **Compiling model:** Configuration: Optimizer: Adam, Loss Function: Sparse Categorical Crossentropy, Metrics: Accuracy
  The Adam optimizer efficiently updates the model's weights during training. The sparse categorical crossentropy loss function is suitable for multi-class classification tasks with integer labels. The model's performance is evaluated using accuracy as the metric.

- **Hyperparameters Description:**
  - **Convolutional layers:**
    Number of filters: 32 in the first convolutional layer, 64 in the second convolutional layer.
    Kernel size: (3, 3) for both convolutional layers.
    Activation function: 'relu' (Rectified Linear Unit) for both convolutional layers.
  - **MaxPooling layers:**
    Pool size: (2, 2) for both MaxPooling layers.
  - **Flatten layer:**
    No hyperparameters.
  - **Dense layers:**
    Number of units: 64 in the first dense layer.

Activation function: 'relu' (Rectified Linear Unit) for the first dense layer.

Number of units: 16 in the output layer.

Activation function: 'softmax' for the output layer. Softmax is used for multi-class classification problems.

- **Compilation:**

  Optimizer: 'adam'. Adam is an optimization algorithm that combines the benefits of both AdaGrad and RMSProp.

  Loss function: 'sparse_categorical_crossentropy'. This loss function is suitable for multi-class classification problems with integer labels.

  Metrics: ['accuracy']. It computes the accuracy of the model during training.

- **Training:**

  Number of epochs: 10. Each epoch represents a complete pass through the entire training dataset.

  Batch size: 32. It represents the number of samples propagated through the network before the weights are updated.

- **Image preprocessing:**

  Grayscale conversion: The images are converted to grayscale using cv2.cvtColor() function.

  Image resizing: The images are resized to (100, 100) pixels using cv2.resize() function.

  Normalization: The pixel values are normalized between 0 and 1 by dividing by 255.

- **Hyperparameter tuning:** We do not used any particular method to tune the hyperparameters, instead we used **hit and trial**, motivated with online forums and other online sources. **Grid Search** is also an option for tuning the hyperparameters but that will require high computation power and time to tune all the **hyperparameters**, that's why we avoided it.

Why I used the above approach?

Initially, I used above approach without segmenting the CAPTCHA images into single character images and labeling it as "EVEN" or "ODD". It was binary class problem and the results were satisfactory, but can be improved using above approach.