CIS 550 - Advanced Machine Learning

(Spring 2024)

**PROJECT Report (Summary)**

**Bridging Linguistic Barrier: American Sign Language (ASL) to English Alphabet Conversion Using CNNs**

**PROFESSOR**

|  |  |
| --- | --- |
| NAME | DR. Ashokkumar Patel |
| OFFICE | DION 302B |
| EMAIL | [apatel38@umassd.edu](mailto:apatel38@umassd.edu) |

**GROUP MEMBERS:**

**Group-32**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Name** | **Student ID** |
| **29** | **Supreeth Mohan** | **02036259** |
| **18** | **Venkata Sai Sandeep Kasiraju** | **02084046** |
| **06** | **Aryan Bhalla** | **02107402** |

About the Project

This project is centered on tackling the communication barriers that exist between users of American Sign Language (ASL) and English speakers. ASL, primarily employed by the deaf and hard-of-hearing community, relies on a rich array of gestures, facial expressions, and body movements to convey meaning. However, its grammar and structure diverge significantly from those of spoken languages like English.

The primary goal of this endeavor is to develop a sophisticated system capable of translating ASL gestures representing individual letters of the English alphabet into written text. To achieve this, Convolutional Neural Networks (CNNs) are employed. CNNs are a type of deep learning model particularly adept at analyzing visual data, making them an ideal choice for interpreting intricate hand inherent gestures in ASL.

Beyond its practical utility, ASL carries profound cultural significance as a form of expression and identity within the Deaf community, both locally and globally. By bridging the linguistic and cultural gaps between ASL users and English speakers, this project aspires to foster efficient communication and mutual kindness.

The application of CNNs for ASL-to-English alphabet conversion holds immense transformative potential. By deciphering the nuanced visual cues of ASL, this technology stands to significantly enhance inclusion and accessibility for the deaf and hard-of-hearing population. Acting as a real-time translation tool between ASL gestures and English letters, the project aims to facilitate seamless cross-lingual communication, empowering individuals and contributing to the creation of a more inclusive society.

What, Why, and How?

**What:**

The project aims to create a machine learning solution that transforms American Sign Language (ASL) alphabets A to Z into their corresponding English characters. This endeavor entails developing a system that can identify ASL hand gestures and convert them into written English text.

**• Importance of Sign Language:** Sign language holds immense importance in the lives of individuals within the deaf and hard-of-hearing community, serving as a unique mode of expression and facilitating meaningful connections.

**• Harnessing Machine Learning:** By leveraging machine learning algorithms, specifically those tailored for image recognition, the project seeks to forge a connection between American Sign Language (ASL) and written English communication

**• Advancing Accessibility:** The primary objective of the project is to elevate accessibility and inclusivity by developing a trustworthy tool for ASL-to-English translation, thereby dismantling language barriers.

**• Humanitarian Goals:** Central to the project's mission is its humanitarian aim of empowering individuals with hearing impairments. It aims to provide them with a reliable avenue for communication and self-expression, ultimately striving to improve their overall quality of life.

**Why:**

The project is driven by multiple motivating factors:

**• Overcoming Communication Challenges:** This point addresses the need to bridge the communication gap between ASL users and English speakers, a divide that significantly hinders interaction. Solutions involve developing tools or systems that can translate ASL into spoken or written English, facilitating clear and effective communication between the deaf community and those who do not know sign language.

**• Embracing Diversity:** The focus here is on promoting inclusivity by providing equal communication opportunities for people with hearing disabilities. This means creating environments and technologies that accommodate the unique needs of the deaf community, ensuring they can participate fully in social, educational, and professional settings.

**• Harnessing Technological Advancements:** This involves utilizing cutting-edge machine learning techniques to improve image recognition capabilities, which can be particularly transformative in interpreting ASL. Advancements in this area are not only pivotal for communication tools but also hold promise for enhancing accessibility in a broad range of applications, from automated interpretation services to educational platforms.

**How:**

The project will advance in the following manner:

**• Data Compilation:** The initial phase involves compiling a comprehensive dataset comprising images of American Sign Language hand gestures corresponding to each letter of the English alphabet, from A to Z.

**• Data Preparation:** After data compilation, meticulous cleaning and preprocessing of the dataset will be conducted for training purposes. This process encompasses normalization and augmentation techniques to ensure optimal performance.

**• Model Creation:** The next step involves developing a machine learning model, potentially leveraging Convolutional Neural Networks, tailored to accurately recognize American Sign Language hand gestures.

**• Model Training:** Train the developed model using the meticulously prepared dataset, allowing it to establish correlations between American Sign Language (ASL) gestures and English letters.

**• Performance Assessment:** Assess the model's efficacy and precision by evaluating its performance on testing data, analyzing accuracy and effectiveness to determine its overall performance.

**• Implementation:** If the model exhibits satisfactory performance, proceed to deploy it as a user-friendly tool capable of seamlessly translating American Sign Language gestures into English letters in real time.

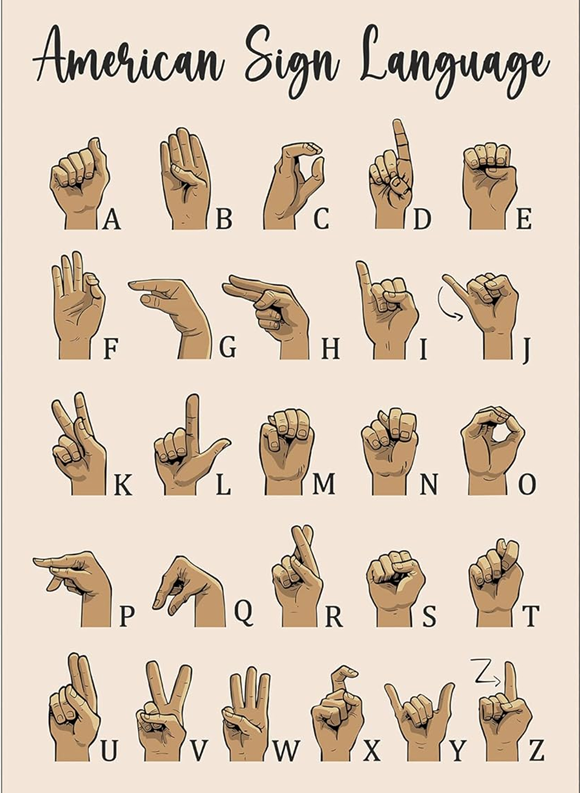
By amalgamating expertise in machine learning with the profound significance of sign language, the project strives to create a tangible impact on communication accessibility and the overall well-being of individuals impacted by hearing impairments.

Dataset Details

* **Dataset Source**
  + Kaggle is acknowledged as the origin of our dataset, renowned for its extensive collection of data science and machine learning resources. It served as a central hub where we accessed invaluable datasets essential for conducting advanced analytics and predictive modeling.
* **Dataset Link**
  + The dataset was obtained through a specific link on Kaggle, granting us access to the ASL Alphabet dataset essential for our project. This facilitated the download and utilization of data crucial for tasks related to sign language recognition:

<https://www.kaggle.com/datasets/grassknoted/asl-alphabet>

* **Relevance**
  + The dataset played a pivotal role, serving as the cornerstone of our project's machine-learning initiative. It constituted the essential element upon which the learning algorithms were trained, guaranteeing that our solution was constructed on a sturdy and dependable data foundation.
* **Dataset Contents**
  + The dataset included a comprehensive set of images depicting the American Sign Language alphabet from A to Z, accompanied by their corresponding labels. This extensive content played a crucial role in the supervised learning phase of our model development.
* **Data Diversity**
  + The diversity within the dataset was crucial, providing a wide array of images depicting the ASL alphabet. This diversity enriched both the training and validation processes of our model, ultimately contributing to a comprehensive and effective learning experience.

Project Summary with screenshots

**Imports:** The necessary Python libraries are imported for the task. This includes `numpy` for numerical operations, `cv2` (OpenCV) for image processing, `matplotlib` for plotting, `PIL` (Python Imaging Library) for image handling, `os` for operating system interactions, and `random` for generating random numbers. Additionally, specific modules from `tensorflow.keras` are imported for deep learning tasks.



**Data Preparation:**

* The Paths to directories containing the training and testing datasets are established. These paths are used to access and manipulate the image data for model training and evaluation.
* From the training directory, a sorted list of folder names is extracted. These names denote the various classes present in the dataset. Sorting is implemented to ensure consistent assignment of class labels across different executions of the code.
* Two empty lists are initialized: one for storing images and another for their corresponding labels. These lists will be filled with image data and their respective labels as the images are processed.

**Image Preprocessing Function (preprocessing\_images\_in\_batch):**

* This function named `preprocessing\_images\_in\_batch` is introduced. This function is specifically crafted to load images from their corresponding class folders in batches, standardize their size to 64x64 pixels, and subsequently append the resized images and their corresponding labels to the previously initialized lists.

**Invoking the Image Preprocessing Function:** The invocation of the `preprocessing\_images\_in\_batch` function is essential for processing the images within the training dataset. This step plays a pivotal role in readying the data for machine learning model training, ensuring uniformity in image sizes and accurate labeling associations.

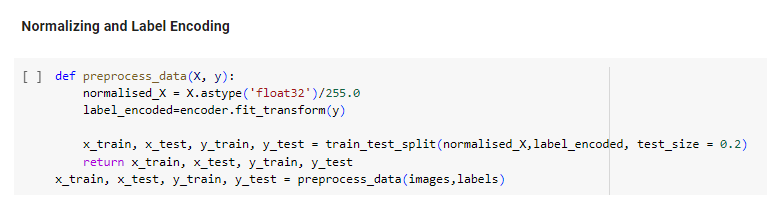


**Conversion to NumPy Arrays:**

* **Conversion:** Converting the data into NumPy arrays optimizes both the image and label data for high-performance operations and facilitates seamless integration with machine learning tools.
* Reshaping the images into a (13050, 64, 64, 3) array aligns perfectly with the input expectations of the neural network. This format indicates the dataset size, image dimensions, and color depth, ensuring compatibility with the model's requirements.

**Data Splitting and Preprocessing:**

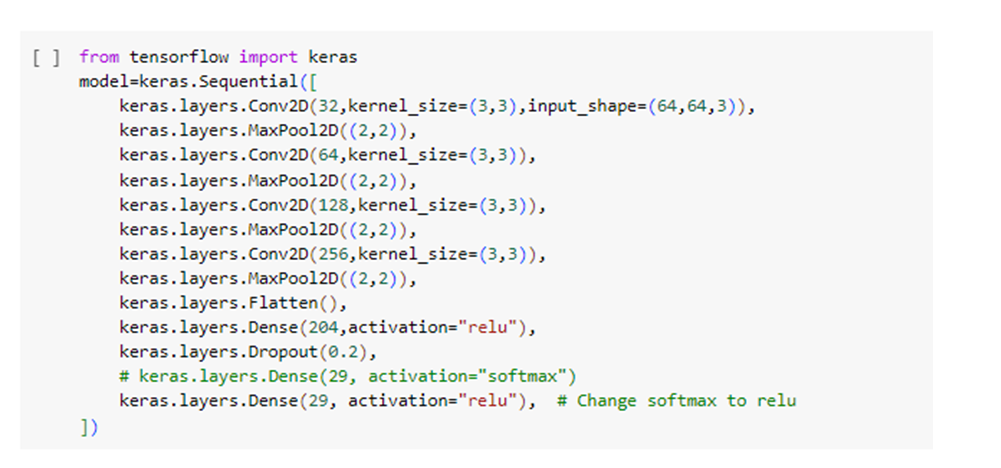
* The LabelEncoder from sklearn.preprocessing is employed to convert categorical class labels into numerical integers. This step is essential preprocessing for most machine learning classifiers, as they typically require numerical input.
* The train\_test\_split function from sklearn.model\_selection is essential for partitioning the dataset into distinct training and testing subsets. This facilitates the model's learning from one portion of the data while being validated against an unseen portion. Such division is crucial for evaluating the model's generalization ability.
* The Normalization of the pixel values to the [0, 1] range is a common practice in image processing for machine learning. This normalization aids in expediting the learning process and enhances convergence during the training of neural network models.



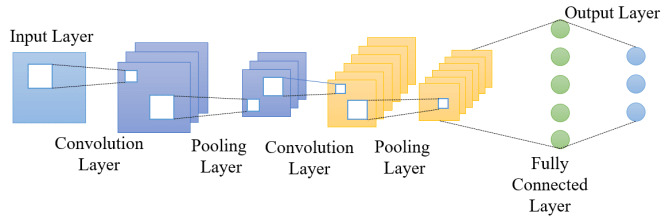
During our discussion, we emphasized the significance of the preprocess\_data function, which plays a crucial role in preparing our dataset for machine learning models. This function normalizes the image data by scaling pixel values to a range between 0 and 1, thereby enhancing model performance through consistent input scales. Furthermore, it utilizes a label encoder to transform categorical labels into numerical values, a pivotal step for the model to understand our classes. Subsequently, the data is split into training and test sets using an 80-20 ratio, furnishing separate datasets for model training and performance evaluation.

**Model Architecture:**

* A sequential CNN model is constructed using keras.Sequential, resulting in a linear stack of layers that is intuitive and easy to manipulate. In this structure, each layer possesses weights specific to that layer, enhancing modularity and ease of use.
* The architecture incorporates convolutional layers for pattern detection and pooling layers for dimensionality reduction. Subsequently, dense layers are utilized to interpret these patterns within the context of the labels.
* For the ASL alphabet dataset containing 29 distinct signs, the model's output layer consists of 29 neurons, each representing a class. This layer employs softmax activation to generate a probability distribution across the classes.



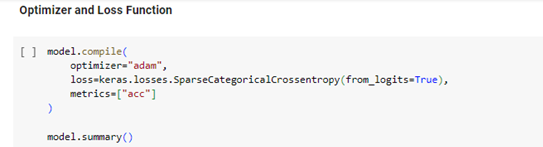
This code snippet constructs a convolutional neural network (CNN) model using TensorFlow's Keras library. The model consists of several convolutional layers with ReLU activations and max pooling layers for feature extraction. Following these layers are a flattened layer, a dropout layer for regularization, and dense layers for classification. The final dense layer utilizes softmax activation for multi-class prediction. However, a comment in the code suggests changing the activation to ReLU, which is typically not recommended for output layers in classification tasks.



This diagram illustrates the structure of a Convolutional Neural Network (CNN). The model commences with an input layer, which receives the raw data, succeeded by a sequence of convolutional layers responsible for feature extraction. Intermittently, pooling layers are inserted to decrease dimensionality and enhance feature detection robustness. The ultimate stage entails a fully connected layer, which interprets these features to generate predictions, ultimately leading to the output layer that produces the final classification outcome.

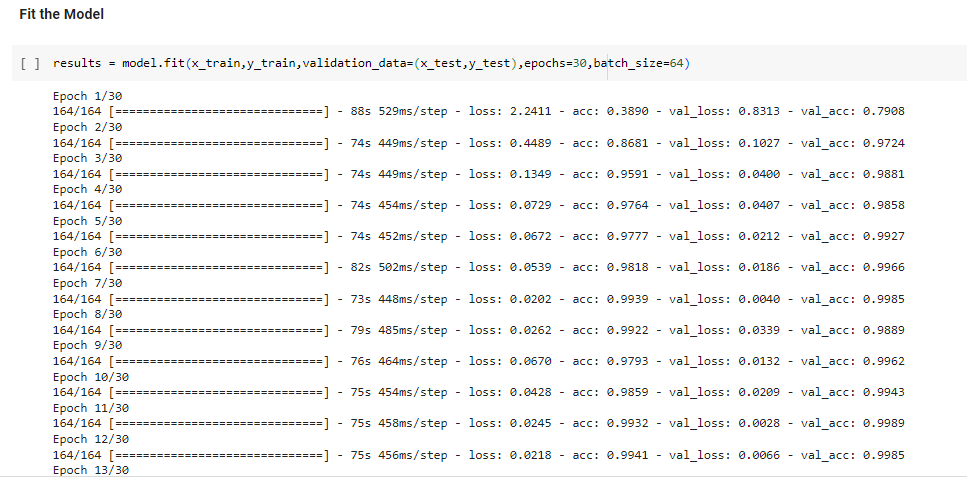
**Model Compilation:**

* The Adam optimizer is selected for its efficacy in managing sparse gradients and dynamically adjusting the learning rate, rendering it well-suited for datasets containing numerous images and classes.
* Accuracy is chosen as the metric due to its straightforward interpretation, representing the proportion of correctly classified instances out of the total predictions made.

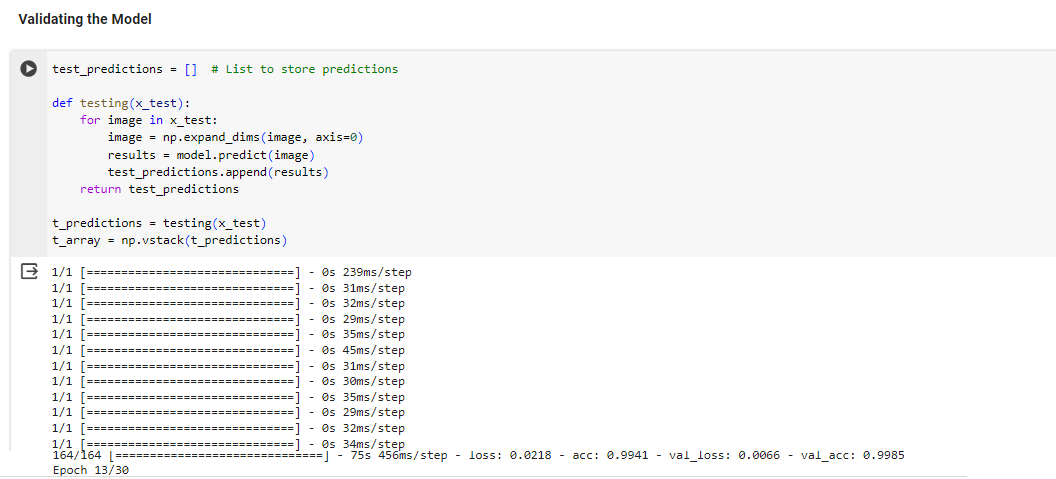


**Model Training:**

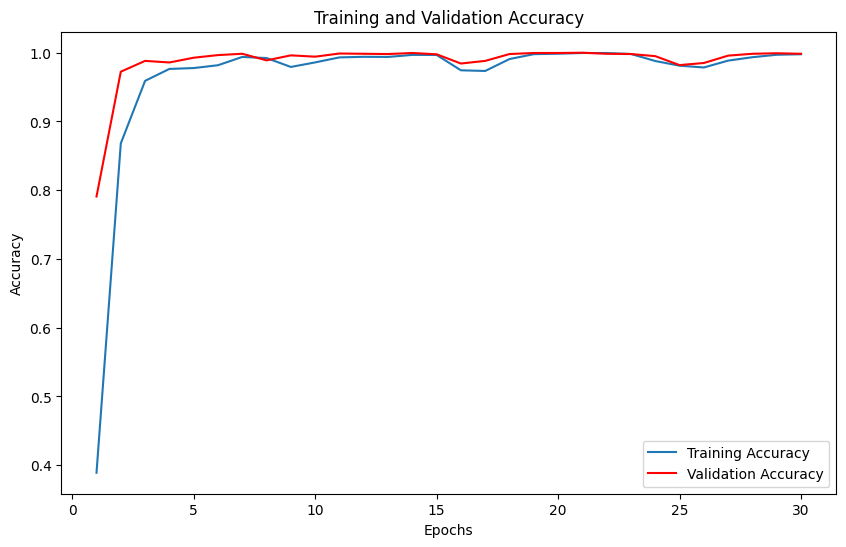
* The model undergoes training on the designated training set and evaluation on the testing set to ensure it can generalize well beyond the data it was trained on.
* Training extends over 30 epochs providing the model ample opportunities to iteratively learn from the data. Batches of 64 images each are utilized to optimize memory and computational efficiency during the training process.
* Throughout the training process, the accuracy and loss for both training and validation sets are continuously monitored and graphically represented through Matplotlib, providing a visual assessment of the model’s performance and convergence behavior.



**Training and Validation Accuracy Calculation:**



* After training, the model is utilized to make predictions on the test dataset, applying the knowledge it has acquired to new, unseen data to evaluate its predictive performance.
* The numerical indices predicted by the model are reverted to their original textual class labels using the inverse transform functionality of the LabelEncoder. This process restores the interpretable category names for better understanding and interpretation.
* The overall effectiveness of the model is assessed by calculating the accuracy, which quantifies the proportion of predictions that match the true labels. This metric offers a clear indication of the model's performance.

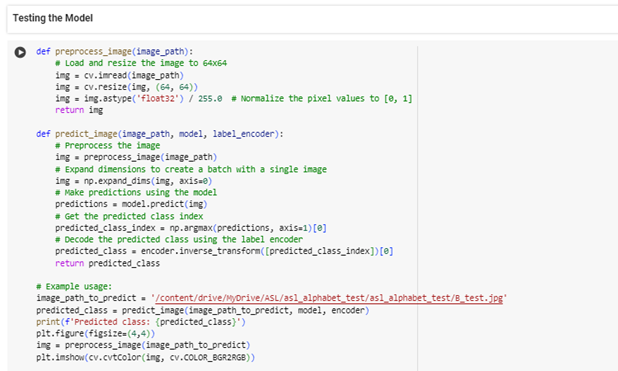


The graph illustrates that the model achieves an accuracy of 98%, signifying its adeptness in correctly predicting the gestures portrayed in the images. Throughout the training process, the accuracy begins at 0.4 and steadily increases to 0.98, indicating a progressive learning curve and gradual improvement in the model's performance over time. This demonstrates the model's capacity to learn iteratively from the training data and enhance its predictive capabilities.

**Image Prediction Functions:** Two functions are provided to predict the class of a single image:

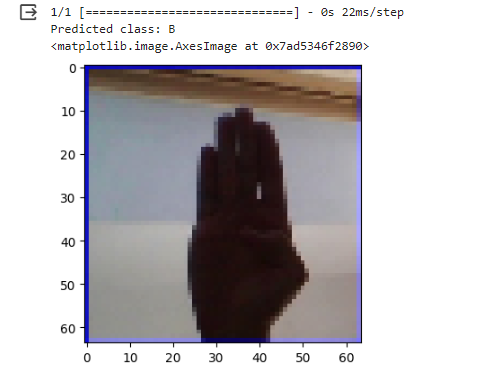
* The `preprocess\_image` function reads an image, resizes it, and normalizes the pixel values to a range appropriate for the model, ensuring that the input is uniform and optimally configured for prediction.
* The `predict\_image` function utilizes preprocess\_image to prepare the image, subsequently employing a trained neural network to predict the image's class. This process translates the model's learned patterns into a definitive classification result.

**Testing Image Predictions and Output:**



Several example images are included and processed through the predict\_image function to showcase the model's predictions for various ASL alphabet gestures. The code adheres to the typical workflow for constructing, training, evaluating, and employing a CNN for image classification duties.

The training process utilizes the ASL alphabet dataset, enabling the resultant trained model to classify novel images by predicting their respective classes.



The model has predicted class 'B' for the provided image, which appears to be a hand gesture. Following the textual output, an image of the classified hand gesture is displayed below. However, the image quality appears to be reduced, possibly for privacy reasons or to emphasize the silhouette. The entire process was completed swiftly, with each step taking only 22 milliseconds.



The image is retrieved from a designated directory, undergoes preprocessing, and is subsequently inputted into a trained model to ascertain its category. Upon analysis, the model identifies the photo as being categorized under class 'A', with this classification outcome being printed.

References/Citations:

Kaggle Notebook:

* <https://www.kaggle.com/>

Convolution Neural Networks:

* <https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/>
* <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>

American Sign Language Dataset:

* <https://www.kaggle.com/datasets/grassknoted/asl-alphabet>