**Introduction**

Introduction: Road safety is a significant concern for transportation authorities worldwide. One crucial aspect of road safety is accurate and timely identification of traffic signs. Traffic signs provide essential information to drivers regarding speed limits, road conditions, and potential hazards. However, identification of traffic signs can be challenging, especially in poor lighting conditions or for drivers who are not familiar with the signs. Misinterpretation of driving rules due to inaccurate identification of traffic signs can lead to an increased risk of accidents.

To address this problem, we developed a machine learning model that can accurately classify traffic signs in images and display the predictions in an interactive web app. Our goal is to develop an efficient and reliable system for traffic sign recognition and classification that can reduce the number of road accidents.

Our approach is to use a combination of feature engineering and deep learning techniques, including Convolutional Neural Networks (CNNs). The dataset used for this project is the Road Signs Dataset (RSD) containing annotated images of four different types of traffic signs: traffic lights, speed limits, crosswalks, and stop signs. We preprocessed the data and divided it into training and testing sets.

We trained our model using the training set and evaluated its accuracy using the testing set. We also optimized the model's hyperparameters to achieve the best results. Finally, we deployed the model on Streamlit, allowing users to upload images and see the model's predictions in real-time.

This section of the report is critical because it provides an overview of the problem, our goals, and our approach to developing the traffic sign detection machine learning model. Additionally, it highlights the significance of the project's potential impact on transportation authorities, law enforcement agencies, and drivers who rely on accurate traffic sign identification. Finally, it provides an overview of the dataset used, our preprocessing steps, and our model's architecture, hyperparameters, and evaluation metrics.

**Task Definition:**

The project implements a machine learning model for road sign detection using a convolutional neural network (CNN) in Keras. The model is designed to detect four types of road signs: traffic lights, speed limits, crosswalks, and stop signs. The model is trained on a dataset of annotated images, where each image is labeled with the class of the road sign it contains. The dataset is loaded from a directory of XML annotation files, which are parsed using the BeautifulSoup library in Python.

The input to the model is an image of size 224x224 pixels, which is preprocessed by resizing and converting to RGB format. The preprocessed image is then converted to a NumPy array and added to the training data.

The output of the model is a vector of probabilities for each of the four classes, computed using a softmax activation function. The model is trained using categorical cross-entropy loss and the Adam optimizer. The model is compiled and fitted on the training data using the fit() method in Keras. The batch size is set to 64, and the model is trained for 10 epochs.

The model architecture consists of multiple layers of convolutional, max-pooling, and dropout layers, followed by fully connected layers with ReLU activation functions. The model includes a final softmax layer with 4 output units, one for each road sign class. The CNN layers learn to extract features from the input image, while the fully connected layers learn to classify the image based on these features.

The model is evaluated on a test set, which is a randomly selected subset of the annotated images. The performance of the model is measured using the accuracy metric, which is the proportion of correctly classified images in the test set. The model is also validated during training using a separate validation set, which helps to prevent overfitting.

**Algorithm Definition:**

The machine learning model deployed on Streamlit is a road sign detection model that uses a convolutional neural network (CNN) architecture. The CNN model is trained on a dataset of annotated road sign images, where the annotations are provided in XML files. The model is designed to classify four types of road signs: traffic lights, speed limits, crosswalks, and stop signs.

The dataset is loaded using the BeautifulSoup library, which is used to parse the XML files and extract the relevant information about each image, such as the class name, file name, width, height, and depth. The data is then preprocessed and transformed into a format suitable for training the CNN model. The images are resized to 224x224 pixels, converted to RGB format, and stored as arrays of pixel values.

The CNN model consists of multiple layers of convolutional and pooling layers, followed by fully connected layers. The model architecture includes four sets of convolutional layers, each with two 2D convolutional layers and one max pooling layer. The dropout layer is used after each set of convolutional layers to avoid overfitting. After the convolutional layers, the model includes three fully connected layers with 256, 128, and 64 units, respectively. A final dropout layer is added before the output layer, which uses the softmax activation function to produce the class probabilities.

The model is trained using the categorical cross-entropy loss function and the Adam optimizer. The training is done for 10 epochs with a batch size of 64. The model accuracy is evaluated using the accuracy metric. The training and testing data are split into 80:20 ratios using the train\_test\_split function from the sklearn library.

Once the model is trained, it is deployed on the Streamlit platform, where the user can upload a new road sign image to be classified. The image is first preprocessed in the same way as the training data, and then fed into the model for classification. The predicted class label and the corresponding class probability are displayed to the user.

**Machine Learning Approach:**

The machine learning approach used for this project was Convolutional Neural Networks (CNNs). CNNs are a type of deep learning algorithm that are commonly used in image recognition tasks. They are effective at recognizing patterns and features within images by using filters to extract information from the image. In this project, we used a CNN to recognize and classify road signs in images.

The first step in our methodology was to preprocess and explore the data. We used the annotations provided with the dataset to extract the relevant information about the images, including the image path, width, height, depth, and class name. We then mapped the class names to numerical labels so that we could use them as target variables in our machine learning model. Next, we loaded the images and resized them to 224x224 pixels using the Pillow library. We also converted the images to RGB format and flattened them into a 3D tensor to feed into our CNN.

To explore the data, we visualized some sample images and examined the distribution of the target variables. We found that the dataset was relatively balanced, with each class having a similar number of images. Exploratory Data Analysis (EDA) is a crucial step in any data science project. It helps in understanding the dataset, identifying patterns and trends, detecting anomalies, and selecting appropriate machine learning models for the task at hand. In this project, we performed EDA on the image dataset. To begin with, we visualized some sample images to get an idea of what the images looked like. This helped us to understand the complexity of the dataset, identify any anomalies or outliers, and make decisions about the data preprocessing steps required. Next, we examined the distribution of the target variables, which in this case were the different classes of images. We found that the dataset was relatively balanced, with each class having a similar number of images. A balanced dataset is important for machine learning models because it ensures that each class is equally represented in the training set, preventing the model from being biased towards one class.

Chart

Description automatically generated

Additionally, we analyzed the pixel values of the images to identify any patterns or trends. This allowed us to gain insights into the brightness, contrast, and color of the images. We also examined the correlations between the pixel values and the target variables to see if there were any relationships between the two. Overall, EDA helped us to gain a better understanding of the image dataset and make informed decisions about the data preprocessing steps and machine learning models to use.

Chart, box and whisker chart

Description automatically generated

The model architecture is an essential aspect of building any deep learning model. In the given example, the model is a Convolutional Neural Network (CNN) that is designed to classify images into four different classes. To optimize the performance of our model, we found the best combination of hyperparameters. We varied the number of filters in each convolutional layer, the kernel size of each filter, and the learning rate of the optimizer. We also experimented with different activation functions, including relu and sigmoid. The model consists of three convolutional layers, each with an increasing number of filters (32, 64, and 128) and kernel size of (3,3), which means that the model will extract 32, 64, and 128 feature maps, respectively, from the input image. Each convolutional layer is followed by a MaxPooling layer with a pool size of (2,2), which reduces the spatial size of the output from the convolutional layer and helps in extracting more prominent features. After the three convolutional and MaxPooling layers, the Flatten layer is used to convert the 3D output from the previous layers into a 1D array. The 1D array is then passed through two dense layers, with the first having 128 neurons and using relu activation, which helps in learning non-linear features from the output of the previous layers. The second dense layer uses a dropout rate of 0.25, which helps in reducing overfitting by randomly dropping 25% of the connections between the neurons. Finally, the output layer has four neurons with softmax activation, which will output the probability for each of the four classes. The softmax function ensures that the sum of the probabilities of all classes is equal to one, which makes it easier to interpret the output of the model. Overall, the model architecture is designed to learn increasingly complex and abstract features from the input images, which allows it to classify the images accurately into one of the four classes.

We used the Adam optimizer and categorical cross-entropy loss function to train our model. We also used a dropout rate of 0.25 to prevent overfitting. After compiling our model with the specified parameters, we trained it on the training set using the fit() method. During the training process, the model was able to learn the features and patterns in the images and make predictions on new images. We used the validation set to monitor the performance of the model and prevent overfitting. The training process involved multiple iterations (epochs) of passing the data through the model, adjusting the weights and biases to minimize the loss function, and updating the optimizer. We chose a batch size of 64, which means that the model processed 64 images at a time before updating the weights. We monitored the training process by calculating the training and validation accuracy at the end of each epoch. This helped us identify whether the model was overfitting or underfitting. If the training accuracy was much higher than the validation accuracy, it was a sign that the model was overfitting to the training data and not generalizing well to new data. In that case, we would adjust the model architecture or use regularization techniques to prevent overfitting. After training for 10 epochs, we evaluated the performance of the model on the test set, which was a completely unseen set of images. This gave us an estimate of how well the model would perform on new data in the real world. We used the evaluate() method to calculate the test accuracy and other metrics. If the test accuracy was close to the validation accuracy, it was a sign that the model was not overfitting and was able to generalize well to new data.

Our model achieved an accuracy of 74.47% on the training set and 73.30% on the testing set after 10 epochs of training. While the validation accuracy did not improve significantly beyond the second epoch, the model was able to learn some meaningful features from the images.

We visualized the results of the model by plotting some sample images with their predicted classes. We found that the model was able to correctly classify most of the images, but struggled with some of the more complex images with multiple road signs in them.

Overall, our machine learning model was able to successfully detect and classify road signs in images with a reasonable accuracy. While the model did not achieve state-of-the-art performance, it was able to learn some meaningful features from the images and could potentially be improved with more data and hyperparameter tuning.

One limitation of our approach was that we only used a single model for classification. We did not explore the use of ensembling or other advanced techniques that could improve the performance of the model.

Graphical user interface

Description automatically generated with medium confidence

**Summary:**

The aim of this project was to develop a road sign detection machine learning model and deploy it on Streamlit. The model was trained on a dataset of images containing four classes of road signs - traffic light, speed limit, crosswalk, and stop sign. The project made use of various Python libraries including NumPy, Pandas, BeautifulSoup, TensorFlow, PIL, and Keras.

The model was developed using a convolutional neural network (CNN) architecture. The CNN was designed to have multiple convolutional and pooling layers, along with dropout layers to prevent overfitting. The final output layer had four neurons representing the four classes of road signs. The model was trained on 80% of the dataset and validated on the remaining 20%. The model achieved an accuracy of 74.47% on the validation set after 10 epochs of training.

**Conclusions:**

The road sign detection model was successfully developed and deployed on Streamlit. The model achieved a reasonable level of accuracy on the validation set, indicating that it can identify the four classes of road signs in real-world images. The model can be useful for applications such as driver assistance systems, automated vehicles, and traffic analysis.

However, there were some limitations to the project. One of the main limitations was the size of the dataset. The dataset contained only a few hundred images, which may not be sufficient to develop a robust model that can generalize well to new images. Additionally, the model was trained for only 10 epochs due to hardware limitations, which may not have been sufficient for the model to converge to its optimal solution.

**Future Work:**

To overcome the limitations of this project, future work can focus on the following:

1. Increasing the size of the dataset: The model can be trained on a larger dataset containing a more diverse range of road signs to improve its ability to generalize to new images.
2. Increase Epochs: The model architecture can be trained on higher Epochs, but for this we need a high performance computational computer to run the model on.
3. Deployment on a real-time video stream: The current model was trained and tested on individual images. However, in real-world applications, road signs are typically detected in real-time video streams. Therefore, future work can focus on deploying the model on a real-time video stream and evaluating its performance.

In conclusion, we have successfully deployed a road sign detection machine learning model using the Streamlit platform. The model was trained on a dataset of annotated road sign images and achieved a validation accuracy of 73.30% after 10 epochs of training.

The model architecture consists of two Conv2D layers with ReLU activation followed by a MaxPooling2D layer and a Dropout layer. The output of the second Conv2D layer is then flattened and fed into two dense layers with ReLU activation, the last of which uses a softmax activation function to output a probability distribution over the four possible classes of road signs: traffic lights, speed limits, crosswalks, and stop signs.

The model was evaluated on a test set, achieving a test accuracy of 73.30%. Although the model's accuracy is not perfect, it demonstrates that machine learning can be an effective tool for road sign detection. With further improvements to the model architecture and training data, we believe that higher accuracy can be achieved.

In summary, the deployment of this road sign detection machine learning model represents a promising development for the field of computer vision and traffic safety.

**Reference:**https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

**Appendix:**

**https://github.com/shaikhiz/traffic-sign**