Sign Language Recognition

Laure-Emilie MARTIN[1)], Agathe PLU[2)]
[1)] CentraleSupélec, Gif-sur-Yvette, laure-emilie.martin@student-cs.fr
[2)] CentraleSupélec, Gif-sur-Yvette, agathe.plu@student-cs.fr

Abstract

In this project, we undertake a thorough investigation into American Sign Language (ASL) with the overarching aim of developing a system capable of recognizing spelled words contained in a video. We follow a two steps method, the first one being a classification task over the 24 static letters of ASL, the second one being the word recognition. We delve into the exploration of various vision preprocessing methodologies, including the utilization of different color spaces, outline detection techniques, and background removal procedures. These preprocessing steps are then integrated with advanced neural network architectures for classification such as MobileNetV2 and CNN. We achieve a commendable 70% accuracy on the test set and, despite the inherent complexities, our system's performance remains robust even when applied to video data, with a respectable 42% accuracy in retrieving words.

1 Introduction

Sign language, often referred to as the visual-spatial language used primarily by the deaf-impaired community, embodies a form of communication that transcends the limitations of auditory speech. Its significance as a linguistic and cultural entity cannot be overstated, as it serves as a gateway to inclusion and understanding for millions worldwide. There exists over 300 distinct sign languages across the globe, each reflecting the cultural nuances of its users.

Sign language recognition poses a challenge as it conveys meaning through many modes at once: manual features, *i.e.* hand shape and their motion; non-manual features, *i.e.* facial expressions or body movements; and finger spelling (Fig. 1), *i.e.* when a word is spelt explicitly by the representation of alphabet letters through hand configurations. Moreover, each signer has a distinct style, akin to individual handwriting or accents, further complicating recognition efforts.

Having highlighted the multifaceted nature of sign language, with its several kinds of gestures, facial expressions, and body movements, our focus narrows specifically to manual gestures. In this paper, we delve into the study of hand movements, restricting our exploration to static finger-spelling. We will utilize computer vision and a small Neural Network to recognize each individual letter for classification purposes. Our ultimate objective is to interpret spelled words in a video by employing our system to accurately identify them.

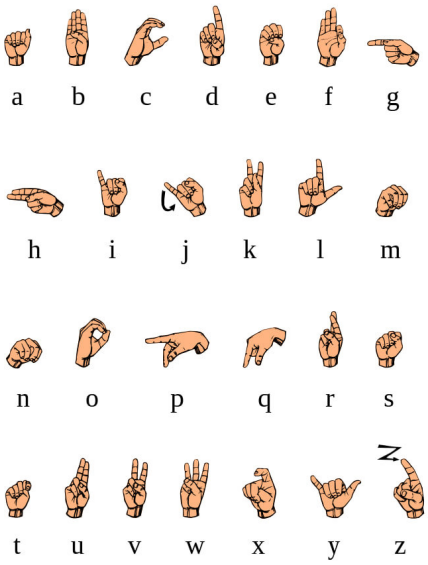


Figure 1: ASL Alphabet

2 Related works

Sign Language Recognition (SLR) has gathered significant attention due to the diverse nature of sign languages worldwide, encompassing over 300 distinct languages. Research efforts have extended beyond American Sign Language (ASL) [8] [3] to include languages such as French [4], Indian [11], Korean [3], Chinese [10], Italian [10], Arabic [2], British [3], and Argentinian [5] Sign Languages, reflecting the global interest in this field. There are two fundamental distinctions in SLR: isolated and continuous sign language recognition. Isolated SLR focuses on recognizing individual signs or gestures, while continuous SLR aims to interpret and translate

entire sign language sentences [11].

Acquisition methods for sign language gestures predominantly fall into two categories: sensors-based and vision-based approaches. Sensors-based methods involve embedded wearable sensors, as gloves and helmets [6], to capture hand movements, while vision-based methods rely on cameras to capture and analyze gestures. We focused our interest on the last method, which often involves 2D image analysis or 3D modeling for enhanced accuracy.

Vision-based approaches have demonstrated effectiveness in capturing hand shapes, with techniques like color-coded gloves aiding in segmentation and detection to overcome overlapping fingers and interference from facial features in images [3] [5]. Finger spelling recognition, requiring precise hand shape description, utilizes methods like edge detection and color segmentation [7]. Techniques like isolating body and hand features, and utilizing Long Short-Term Memory (LSTM) networks aid in overcoming these challenges [5].

Other classification algorithms such as Hidden Markov Models (HMM) [7] [10] [3] [6], Neural Networks (NN), and k-Nearest Neighbors (KNN) [3] are commonly employed in SLR. HMMs, in particular, have been prominent since the 1990s due to their ability to handle the temporal aspect of sign language [3]. Real-time SLR systems leverage technologies like Kinect and Leap Motion Controller for hand tracking and gesture recognition [8]. Recent strides in the field have seen the integration of deep learning methodologies, particularly Convolutional Neural Networks (CNN) and MobileNetV2 [8]. These models, trained on large datasets, exhibit promising results, achieving high accuracy rates even in real-time scenarios. However, the study in real-time is complicated by the lack of large dataset [11].

Preprocessing steps, such as color conversion, noise reduction, sharpening, and extracting key points from hand gestures to enhance classification accuracy. Integration with technologies like OpenCV facilitates real-time gesture recognition, bridging communication gaps for individuals with hearing impairments [6] [2]. In particular, feature extraction plays a crucial role in SLR, with both tracking-based and non-tracking-based methods employed. Tracking-based techniques utilize motion history images (MHIs), Histogram of Oriented Gradients (HOG), wavelets, and Kalman filters [3], while non-tracking-based methods focus on global detection, shape descriptors and geodesics (*i.e.* generalisation of straight lines) [4]. Hybrid approaches integrating both tracking and non-tracking techniques have shown promising results with high accuracy rates [3].

In conclusion, SLR encompasses a diverse range of languages and employs various acquisition and recognition methods. Advancements in deep learning and computer vision continue to enhance the accuracy and efficiency of SLR systems, contributing to improved communication accessibility for the deaf and hard of hearing communities worldwide.

3 Approach

Our primary objective is to utilize an American Sign Language (ASL) dataset to train a compact neural network capable of classifying each letter, with the exception of 'j' and 'z' as they involve motion. To enhance our predictions, we will explore various computer vision techniques to refine our preprocessing steps.

Our approach comprises several key steps. First, we will collect and create our own training, testing and validating datasets by consolidating existing resources. Next, we will establish a robust preprocessing pipeline to enhance the quality of our dataset. Finally, we will train our neural network on the refined dataset to achieve accurate classification results.

We will ultimately employ this neural network training pipeline to interpret 300 spelled words in ASL from a video.

3.1 Dataset

We downloaded our first dataset from [9]. We removed to this dataset the letters j, z, space, delete and nothing, as the two first ones are in movement, whereas the three others are not interesting for our study case. We only took a portion of this dataset as it is very huge and comprised of very similar images. Our training on this dataset provided very poor results, therefore we used a second one [1] to add diversity and we merged both to obtain our newest train, test and validation sets.

We finally incorporated a third dataset to enhance the accuracy of video interpretation. While the initial two datasets consisted of images captured from a first-person perspective, the video depicts a signer from an external viewpoint. This supplementary dataset comprises photographs captured from an external perspective. We will refer this dataset as **Extended Dataset** and the previous one as **Classic Dataset**.

3.2 Preprocessing

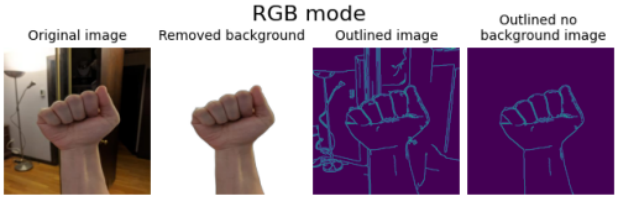
We introduced in our training pipeline several preprocessing techniques and compared their effects

on the performance (Fig. 2).

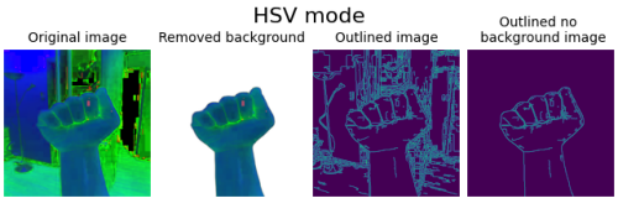
We first compared different color representations spaces. The default one was RGB, and we also used HSV, XYZ and LAB spaces. HSV represents colors based on their hue, saturation, and value, making it intuitive for color selection tasks. XYZ is a color space encompassing all colors visible to the human eye, whereas LAB describes perceptual color differences.

Our second preprocessing step involved background removal using the *u2net* architecture. This step helped isolate the hand by eliminating the background from the image.

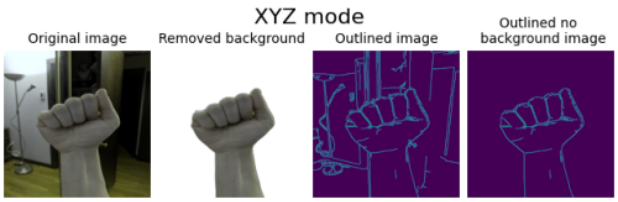
Finally, we used the Canny edge detector from the *OpenCV* library to extract the hand outlines.



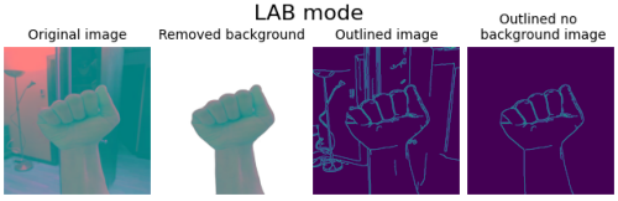
(a) RGB images



(b) HSV images



(c) XYZ images



(d) LAB images

Figure 2: Preprocessing transformation

3.3 Neural network

For the 24-class classification task, we tested two models: a Convolutional Neural Network (CNN), commonly utilized in literature for similar tasks, and the pre-trained MobileNetV2 due to its well-established performance in the classification domain. To adapt MobileNetV2 to our task, we added a 10% dropout layer at the beginning and end, along with a dense output layer with 24 outputs; these parameters were the only ones trained during the process. The dropout layers were included to prevent overfitting by randomly deactivating a fraction of the nodes, while the dense layer ensured the model could output predictions for all 24 classes. Moreover, due to MobileNetV2's pre-training on the dataset *ImageNet*, which consists of images sized 224x224, we included a resizing step in our preprocessing pipeline to ensure compatibility with the model's input requirements.

Both models were optimized using the Adam optimizer, a popular choice for its adaptive learning rate capabilities, which can lead to faster convergence. Additionally, we utilized categorical cross-entropy loss as the loss function, which is suitable for multi-class classification tasks as it measures the difference between predicted and actual class distributions.

We conducted model testing across several epochs, ranging from 3 to 15, which proved sufficient for achieving convergence without encountering overfitting issues.

3.4 Video

The primary objective of this project was to accurately recognize spelled words in a real video using models that had shown the best performance in previous benchmarks. To achieve this, we selected a specific video [12] containing 300 spelled words, each consisting of five letters.

To process the video, we developed a script to segment it into individual words. This segmentation was based on a threshold determined by the ratio of red to blue pixels in each frame. When the hand disappeared from the screen, this ratio exhibited a significant change, allowing us to detect word boundaries. We set the threshold for this ratio to 0.99, a value determined through analysis of a graph plotted at the beginning of the video. This graph depicted the ratio over the course of an extract of the video Fig. 3.

Subsequently, we isolated each letter within the words using a script that extracted five frames from the video, timed according to the video's rhythm. While this method generally aligned with the pace of letter spelling, occasional errors occurred due to

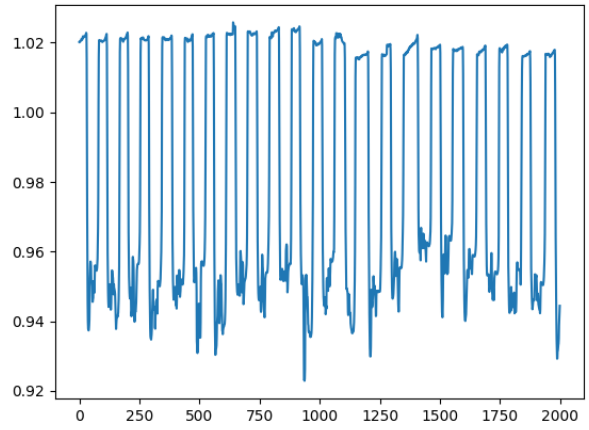


Figure 3: Red to blue pixels ratio, the x-axis represents the number of frames and the y-axis represents the ratio.

variations in rhythm during the letter tracing process.

Finally, we applied our best-performing models to recognize each letter, predicting the five most probable letters for each case. Utilizing these probabilities assigned to each predicted letter, we employed an algorithm to compute the score for every combination of letters. This technique is inspired by common AI hybrid techniques using the probability of the model-predicted result to infer the final results. We then compared the resulting words against an English dictionary to keep only the correct words.

4 Results

4.1 Models accuracy for different preprocessing steps

We started by comparing our two models with our first conditions: RGB images, with outlines detection and with background removal on three epochs. The CNN shows poor performances compared to MobileNetV2 (see respectively Fig. 6 and Fig. 7) and was taking much longer to train. Hence, we only focused afterwards on MobileNetV2.

We conducted several tests with all our preprocessing combination, mainly on three and eight epochs training (see Fig. 8). This was done with on the **Classic Dataset** presented above (Sec. 3.1). This training period allowed us to highlight our best preprocessing configuration, achieved with RGB or XYZ colors, outline detection and background removal, on eight epochs. We achieved an accuracy of 70% on the first predicted letter, and respectively 88% and 87% on the three first predicted letters (see Fig. 10 and Fig. 11).

The same tests were conducted on the **Extended Dataset** with the aims of upgrading the final performance on the video (see Fig. 9). The two most effective configurations with this dataset matched those from the previous tests, achieving accuracies of 67% and 69% respectively, over eight epochs. With this new dataset, 15 epochs on MobileNetV2 were needed to achieve full convergence. Therefore, we trained these two configurations up to this point and achieved an accuracy of 70% for both on the first letter. For the RGB case, we achieved an accuracy of 89%, compared to 88% for the XYZ case on the three first predicted letters (see Fig. 12 and Fig. 13).

4.2 Spelled words detection on the video

The first challenge was to separate each letter of a word, which we realized based on the rhythm. We analyzed ourselves the results of this splitting and calculated the number of correctly split letters. Overall, on 1475 split letters 243 were not correctly isolated (thus 16%), for instance the letters from the word shown in 4.

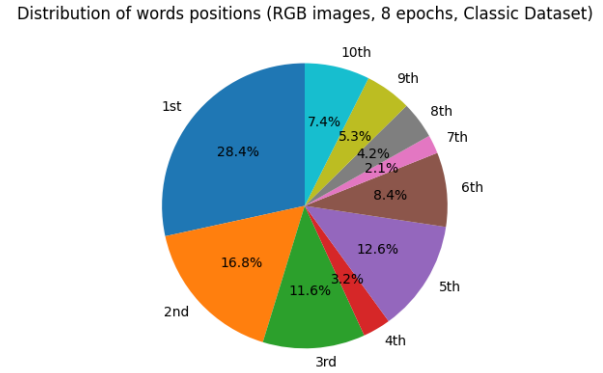
(a) *THERE*, correctly split

(b) *BRING*, not correctly split

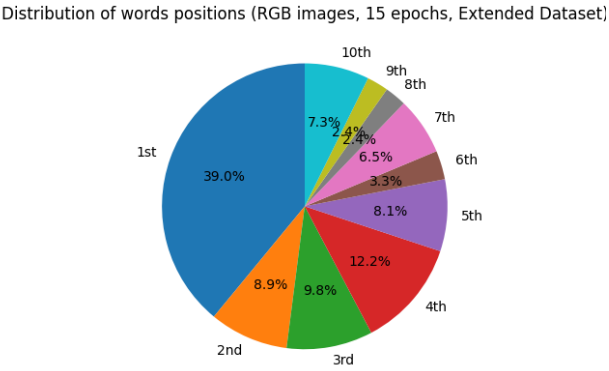
Figure 4: Examples of split letters

On all words, we predicted the five most probable letters and used our script to find the ten most probable existing words. We then checked if the actual word was among this set of words and computed a custom accuracy based on this. The best result we achieved was with MobileNetV2 trained on the **Extended Dataset** for 15 epochs with RGB images, outline detection and background removal. Out of 295 words, 123 were correctly identified (*i.e.*, they were part of the ten most probable words) compared to 120 for the same configuration with XYZ images. For the configuration with RGB images, outline detection and background removal on the **Classic Dataset** trained for eight epochs, we obtained 95 correctly predicted words. This difference in results demonstrates the robustness of our model with the dataset extension we

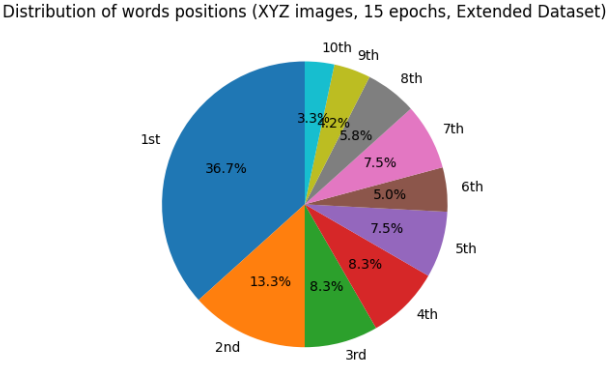
realised.



(a) RGB images, 8 epochs, **Classic Dataset**



(b) RGB images, 15 epochs, **Extended Dataset**



(c) XYZ images, 15 epochs, **Extended Dataset**

Figure 5: Distribution of words positions for different configurations

We traced the distribution of words positions for correctly interpreted words for these three configurations (Fig. 5). We noticed that roughly 35% of correctly predicted words appeared in first position (*i.e.* they are the most probable words). With our best model, we obtain that more than 75% correctly predicted words were in the top 4.

5 Discussion

Overall, we achieved satisfying classification results, significantly enhanced by preprocessing steps, particularly outline detection and background removal. We achieved a 70% accuracy rate in correctly predicting letters. However, our performance diminished when applied to video data, yielding a 42% accuracy rate. This outcome is not unexpected given the inherent complexity of the task, compounded by the scarcity of images in the datasets captured from the optimal viewpoint. Indeed, images are typically taken in selfie mode, whereas videos are recorded from an external perspective.

We implemented a letter splitting technique based on rhythm, which yielded satisfactory results, with only 16% of images improperly segmented. Nevertheless, this approach has inherent limitations. To address this issue, exploring real-time recognition techniques on video data could have potentially improved segmentation accuracy. Additionally, to enhance our results, we could have considered combining the outputs of models trained on RGB and XYZ data, as we observed discrepancies in their predictions of correct words. This could have been achieved by aggregating word scores and selecting the highest-ranking predictions.

References

[1] American sign language letters dataset v1, October 2020.

[2] Shaikhah Almana and Alauddin Al-Omary. Real-time arabic sign language recognition using cnn and opencv. In *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 32–36, 2022.

[3] Helen Cooper, Brian Holt, and Richard Bowden. *Sign Language Recognition*, pages 539–562. 01 2011.

[4] Abdesslam Benzinou Kamal Nasreddine. Reconnaissance automatique de gestes manuels en langue. 06 2016.

[5] Dimitrios Konstantinidis, Kosmas Dimitropoulos, and Petros Daras. Sign language recognition based on hand and body skeletal data. 05 2018.

[6] Ashish S. Nikam and Aarti G. Ambekar. Sign language recognition using image based hand gesture recognition techniques. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, 2016.

[7] Aneesh Pradeep, Mukhammadkhon Asrorov, and Muxlisa Quronboyeva. Advancement of sign language recognition through technology using python and opencv. In *2023 7th International Multi-Topic ICT Conference (IMTIC)*, pages 1–7, 2023.

[8] Suraiya Yasmin et al Refat Khan Pathan, Munmun Biswas. Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. 2023.

[9] Debashish Sau. Asl(american sign language) alphabet dataset, 2021.

[10] Veronica Juliana Schmalz. Real-time italian sign language recognition with deep learning. In *CEUR Workshop Proceedings*, volume 3078, pages 45–57. CEUR Workshop Proceedings, 2022.

[11] Sharvani Srivastava, Amisha Gangwar, Richa Mishra, and Sudhakar Singh. Sign language recognition system using tensorflow object detection API. *CoRR*, abs/2201.01486, 2022.

[12] Bill Vicars. Fingerspelling (5-letter words) american sign language (asl), 2019.

Appendices

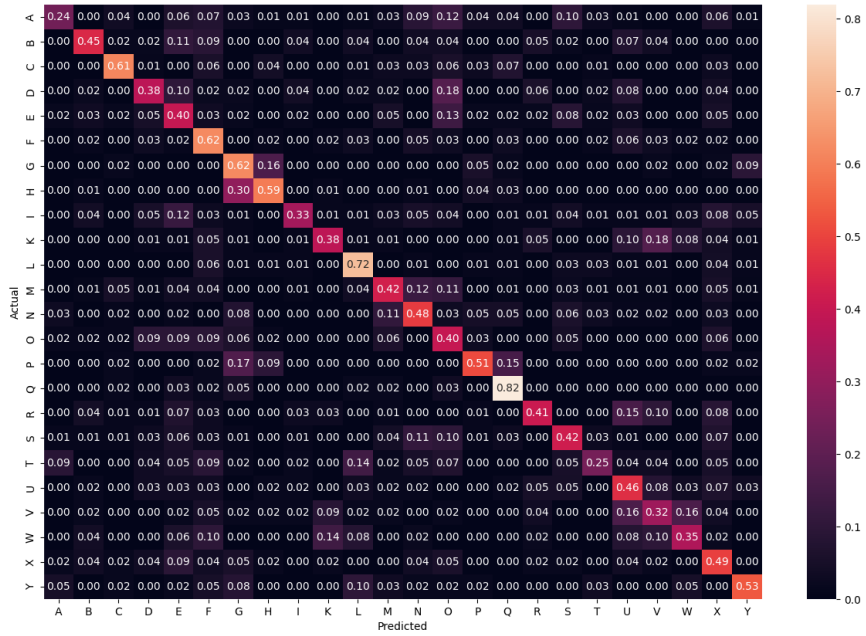


Figure 6: Confusion matrix (CNN, 3 epochs, RGB images, outline detection and background removal)

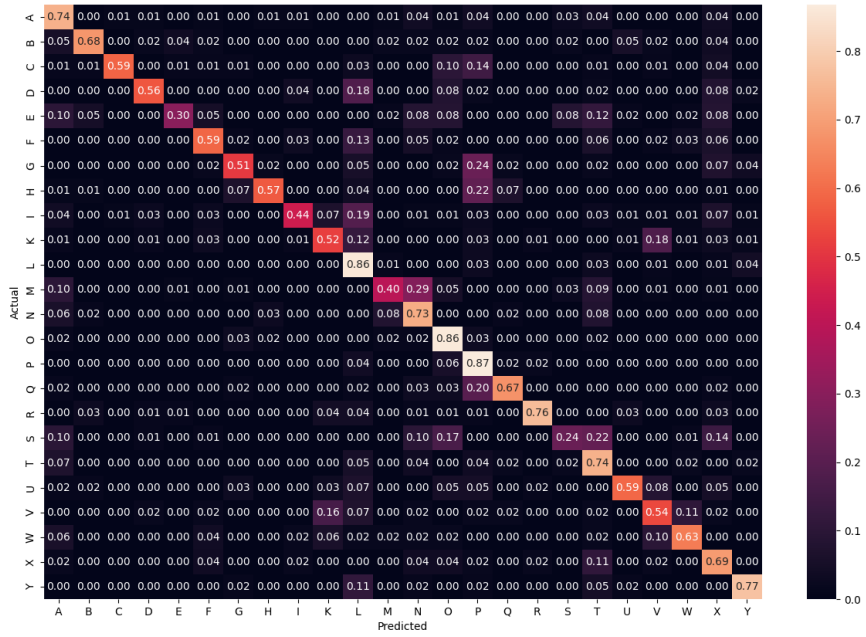


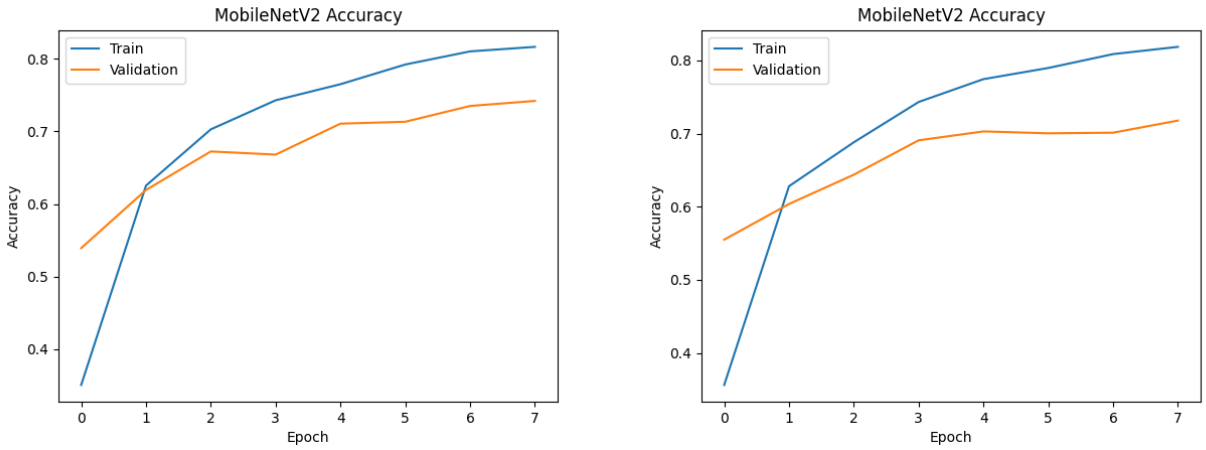
Figure 7: Confusion matrix (MobileNetV2, 3 epochs, RGB images, outline detection and background removal)

Model	Nb of epochs	Outlines	Background	Colors	Accuracy 1	Accuracy 3
MobileNetV2	3	Yes	Without	RGB	61%	83%
MobileNetV2	3	Yes	Without	HSV	57%	77%
MobileNetV2	3	No	Without	RGB	39%	69%
MobileNetV2	3	No	Without	HSV	48%	73%
MobileNetV2	3	No	With	HSV	16%	36%
MobileNetV2	3	Yes	With	HSV	31%	54%
MobileNetV2	3	No	With	RGB	26%	50%
MobileNetV2	3	Yes	With	RGB	41%	66%
MobileNetV2	3	Yes	Without	XYZ	64%	85%
MobileNetV2	3	Yes	Without	LAB	61%	81%
MobileNetV2	6	Yes	Without	RGB	69%	88%
MobileNetV2	8	Yes	Without	RGB	71%	88%
MobileNetV2	8	No	With	HSV	22%	42%
MobileNetV2	8	Yes	Without	HSV	64%	82%
MobileNetV2	8	No	Without	HSV	51%	78%
MobileNetV2	8	Yes	With	HSV	35%	56%
MobileNetV2	8	No	With	RGB	27%	53%
MobileNetV2	8	Yes	With	RGB	48%	72%
MobileNetV2	8	Yes	Without	XYZ	70%	87%
MobileNetV2	8	Yes	Without	LAB	66%	86%
MobileNetV2	10	Yes	Without	RGB	70%	88%
MobileNetV2	10	Yes	Without	HSV	64%	83%
MobileNetV2	10	No	Without	HSV	51%	76%
CNN	3	Yes	Without	RGB	47%	69%

Figure 8: Table results of the conducted tests on the **Classic Dataset**

Model	Nb of epochs	Outlines	Background	Colors	Accuracy 1	Accuracy 3
MobileNetV2	8	Yes	Without	RGB	67%	88%
MobileNetV2	8	Yes	Without	HSV	59%	81%
MobileNetV2	8	Yes	Without	XYZ	69%	88%
MobileNetV2	8	Yes	Without	LAB	64%	85%
MobileNetV2	8	No	Without	RGB	30%	56%
MobileNetV2	8	No	Without	HSV	44%	71%
MobileNetV2	8	No	Without	XYZ	33%	63%
MobileNetV2	8	No	Without	LAB	25%	50%
MobileNetV2	8	Yes	With	RGB	37%	61%
MobileNetV2	8	Yes	With	HSV	27%	48%
MobileNetV2	8	Yes	With	XYZ	37%	59%
MobileNetV2	8	Yes	With	LAB	37%	60%
MobileNetV2	8	No	With	RGB	20%	40%
MobileNetV2	8	No	With	HSV	17%	38%
MobileNetV2	8	No	With	XYZ	26%	48%
MobileNetV2	8	No	With	LAB	13%	28%
MobileNetV2	15	Yes	Without	RGB	70%	89%
MobileNetV2	15	Yes	Without	XYZ	70%	88%

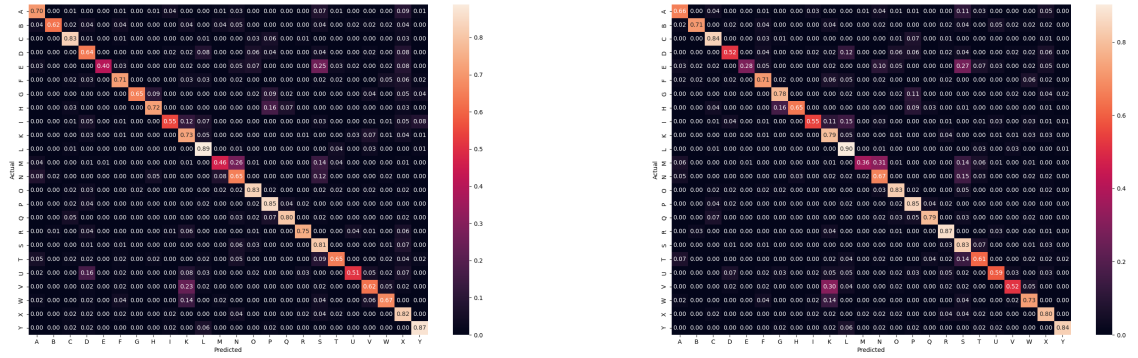
Figure 9: Table results of the conducted tests on the **Extended Dataset**



(a) Accuracy for RGB images

(b) Accuracy for XYZ images

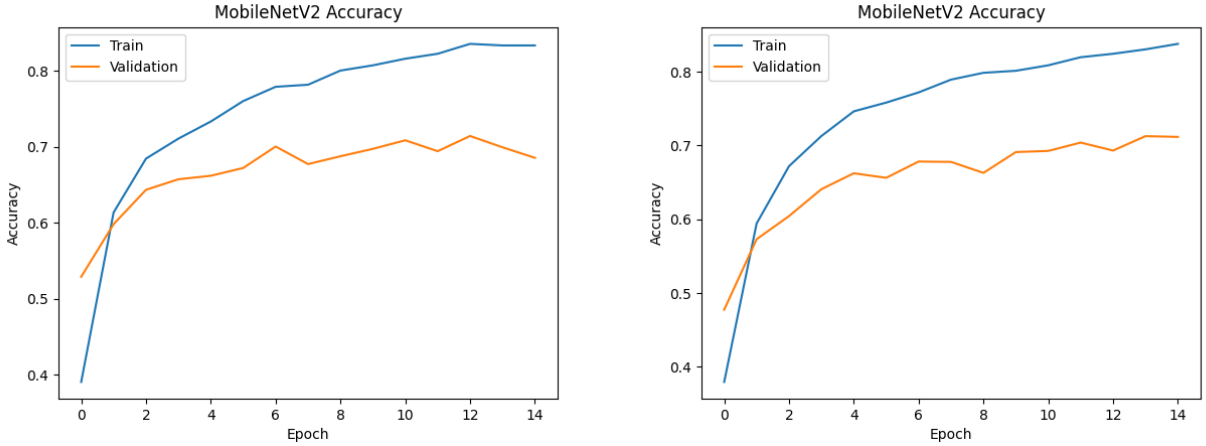
Figure 10: Accuracy comparison (8 epochs, outline detection and background removal)



(a) Confusion matrix for RGB images

(b) Confusion matrix for XYZ images

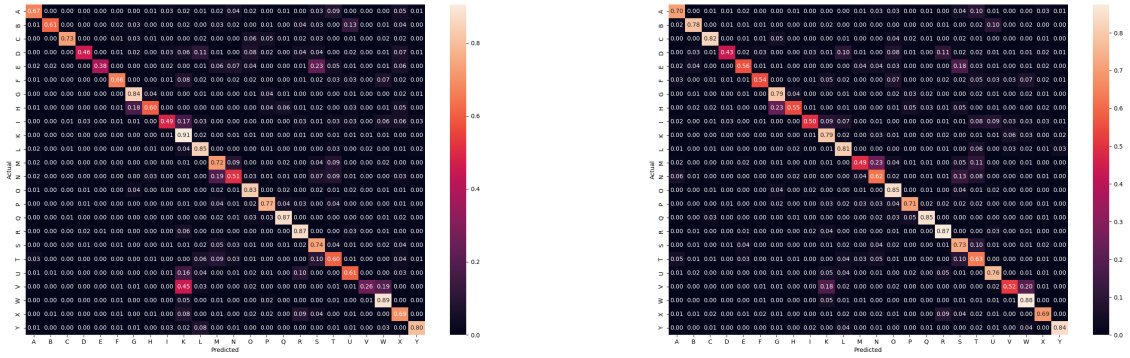
Figure 11: Confusion matrix comparison (8 epochs, outline detection and background removal)



(a) Accuracy for RGB images

(b) Accuracy for XYZ images

Figure 12: Accuracy comparison (15 epochs, outline detection and background removal)



(a) Confusion matrix for RGB images

(b) Confusion matrix for XYZ images

Figure 13: Confusion matrix comparison (15 epochs, outline detection and background removal)