Alphabet Level Indian Sign Language Recognition

Abstract: Communication entails the exchange of information, ideas, or emotions. For effective communication, a shared language is essential. However, deaf, and mute individuals, who cannot hear or speak, rely on sign language. Despite its significance, sign language is often overlooked by the general populace, leading to communication barriers between deaf individuals and others. To address this issue, we propose leveraging deep learning and computer vision to develop a solution. By training a model to interpret Indian Sign Language and translate them into English, we can facilitate communication between deaf ones and the hearing population. Existing Indian Sign Language Recognition systems employ machine learning algorithms to recognize single and double-handed gestures but lack real time capabilities. In this paper, we introduce a novel approach to creating an Indian Sign Language dataset with the help of webcam. Subsequently, we employ transfer learning to train a TensorFlow model, enabling real time Recognition The outcome of the system relies on the confidence level, and the system's mean confidence level stands at 80%. Despite the restricted dataset, our system attains praiseworthy accuracy

Keywords: Indian Sign Language (Indian-SL), Sign Language Recognition (SLR), Computer Vision, Machine Learning, TensorFlow, Object Detection.

1. Introduction

Communication is the process of transmitting information from one location, individual, or collective to another, involving a conveyed message, a speaker, and a recipient. Its effectiveness depends on the successful reception and comprehension of the message. Communication includes informal/formal modes, oral/written forms, wordless cues, grapevine communication, visual aids, feedback mechanisms, and active listening. Formal communication follows predetermined channels and protocols, while unofficial or grapevine communication is spontaneous. Oral communication includes face-to-face interactions and distance communication via technology like voice/video calls. Written communication uses emails, letters, notices, etc. Non-verbal communication uses gestures, facial expressions, and body language. Visual communication involves sources like television or social media. Active listening is about understanding what others communicate. Deaf and mute individuals face challenges in communication, with sign language providing a non-verbal means. Technology can translate sign language into spoken languages like English. Past research used data gloves, motion capturing systems, sensors, and vision-based Sign Language Recognition (SLR) systems, including those developed with MATLAB for high accuracy but lacking real-time capabilities. This paper aims to create a real-time SLR system using the TensorFlow API and train it with a custom dataset from a webcam.

2. Related works

In essence, sign languages consist of structured hand gestures conveying specific meanings, primarily used by the hearing-impaired community for everyday communication[2]. These languages are visual, utilizing hand, facial, and bodily movements as their primary mode of expression. Despite there being over 300 distinct sign languages worldwide, the percentage of individuals proficient in any particular sign language remains low, hindering communication for those with hearing impairments[4]. Systematic Literature Review (SLR) systems offer a solution by enabling communication in sign language without requiring fluency in it; instead, they recognize gestures and translate them into commonly spoken languages such as English.

The field of SLR is extensive, with ongoing research addressing various challenges. Machine learning techniques play a crucial role in enabling electronic systems to make decisions based on data and experience. Classification algorithms typically rely on two datasets: a training dataset for providing experiences to the classifier and a testing dataset for evaluating the performance of the model. Previous research has focused on developing efficient methods for data acquisition and classification[2][6]. Approaches to data acquisition can be broadly categorized into direct methods of measurement and approaches based on vision.[2]. Direct measurement methods utilize devices such as motion-capturing systems, motion data gloves, or other sensors to extract the data of motion accurately, enabling strong SLR procedures. Vision-based approaches, on the other hand, involve extracting discriminative spatial and temporal features from TrueColor images. These methods often begin by detecting and tracking hand regions, employing techniques such as semantic segmentation, skin color detection, face detection, and background removal to isolate and track moving hand parts accurately[7].

Numerous processing methods have been applied in the development of SLR systems[11], including Hidden Markov Models (HMM), neural networks, Support Vector Machines (SVM), and wavelet-based techniques, among others. These methods yield varying accuracy results, with models like SVM and wavelet-based methods achieving high accuracy rates. However, accuracy is influenced by multiple factors such as dataset size, image clarity, and the specific processing model employed.

SLR systems can be categorized into isolated and continuous SLR. Isolated SLR focuses on recognizing single gestures, typically associated with alphabets, digits, or specific gestures. In contrast, continuous SLR aims to recognize and translate entire sentences, requiring more complex processing techniques. Challenges in SLR research include the laborious labeling process for isolated SLR, the complexity of temporal segmentation in continuous SLR, the costliness of data acquisition devices, and the scarcity of large datasets.

Furthermore, vision-based methodologies face challenges such as inaccuracies due to overlapping hand and finger movements. Additionally, misconceptions about sign language persist, such as the belief that sign language is uniform across different regions, whereas it often reflects the spoken language of the respective region. For instance, Indian Sign Language utilizes both single-hand and double-hand gestures to convey meaning[3][5].

In this, a dataset was created using python and opencv with the webcam for real time detection in the development of an SLR system. This approach aims to address some of the challenges in SLR research, particularly in terms of data acquisition and real time processing capabilities.

3. Data Acquisition

A real time system for detecting Indian-SL gestures is under development, utilizing a webcam and Python with OpenCV for data acquisition. OpenCV, a powerful library for computer vision, offers functions tailored for real time image processing, accelerating the integration of machine perception into various applications. With over 2500 efficient algorithms, OpenCV facilitates tasks such as face detection, object recognition, human action classification, and object tracking[19].

The dataset comprises images representing the alphabet and numeric values from 0 to 9 in Indian Sign Language, with each alphabet represented by 50-60

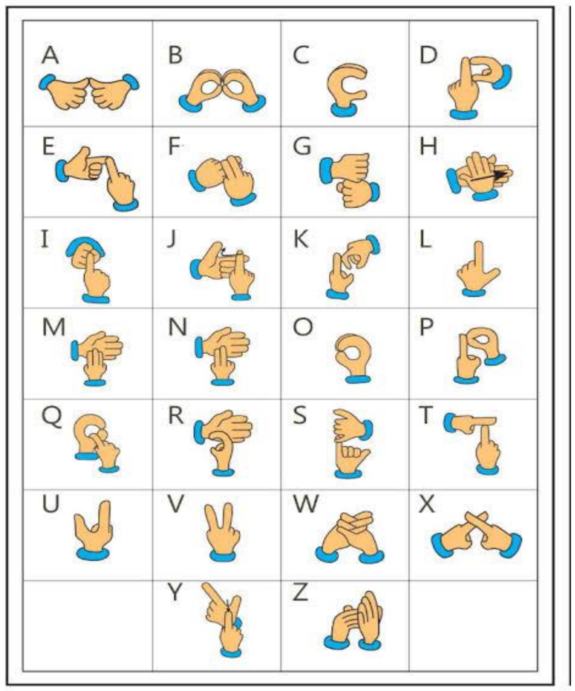

Figure 1 Alphabets of Indian Sign Language[22].

images[22]. Images are captured every 1 seconds, allowing for slight variations in gesture recording, while a 5-second interval between individual signs allows for transitioning between different alphabets. Gathered(captured) images are stored in their respective folders for further processing and analysis.

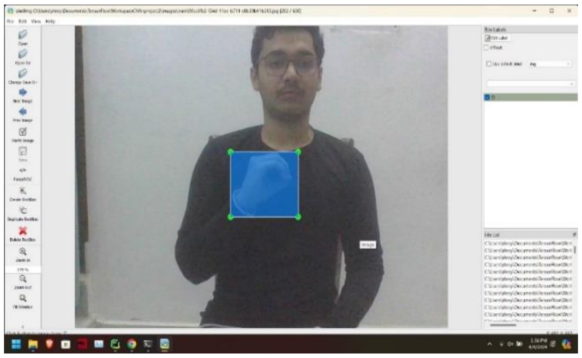

Figure 2 Assigning labels to a portion of the image.

For data gathering, several dependencies have been imported, including cv2 (OpenCV), uuid, os and time. This os module is utilized for managing file paths, as it offers functions for interacting with the operating system. It is a standard utility module in Python. The time module serves various purposes, such as representing time in different formats like objects, numbers, and strings. Moreover, it can be utilized for assessing code reliability or inserting pauses during code execution. In this scenario, it's employed to establish intervals between captured images, providing enough time for accurate hand signs. This uuid library aids in the naming image files by generating unique identifiers or time-based names. These identifiers are generated based on time and computer hardware, ensuring uniqueness for each file. After capturing all the images, they are labeled using the Labeling package, a free open-source tool designed for graphical image labeling. Each image's hand gesture portion is annotated to denote the corresponding sign it represents. Upon saving, the labeled images generate XML files containing comprehensive details, including the labeled portions. Subsequently, these XML files are utilized to generate TensorFlow (TF) records. The dataset, comprising both images and XML files, is then partitioned into training and validation sets following an 90:10 ratio. Specifically, 90% of the images are designated for training, while the remaining 10% are set aside for validation. This process is systematically repeated for all 26 alphabets, ensuring a well-annotated and balanced dataset for subsequent model training and evaluation.

4. Methodology & Architectures

4.1. EfficientDet D0

The EfficientDet D0 architecture represents a single-stage object detection approach that balances speed and accuracy. Unlike the two-stage Faster R-CNN, EfficientDet D0 predicts object class probabilities and bounding boxes in a single pass, leading to significantly faster inference. However, achieving this speed without compromising accuracy necessitates careful design considerations. EfficientDet D0 utilizes a compound scaling method that increases the model's capacity (resolution, depth, and width) in a balanced way. This ensures the model can learn complex features from data while maintaining real time inference on a single GPU. The EfficientDet family, introduced by Zhong et al. (2020)[23], showcases D0's effectiveness on various object detection applications.

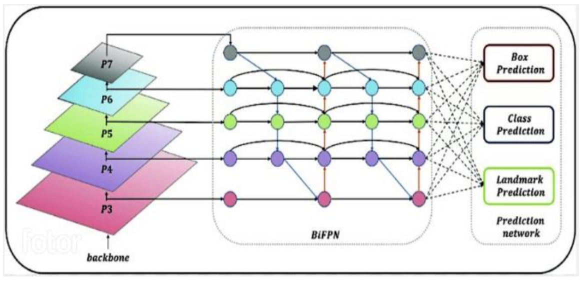

Figure 3 The Structure of EfficientDet D0[24].

4.2. SSD ResNet101 V1 FPN (RetinaNet101)

The SSD ResNet101 V1 FPN 640x640 (RetinaNet101) architecture offers a balance between accuracy and speed for object detection. It leverages a single-stage approach similar to EfficientDet D1, making predictions in one pass. This model employs the SSD (Single-Shot MultiBox Detector) framework, which utilizes (CNN) convolutional neural network to predict class probabilities and bounding boxes for objects at various scales within an image. The ResNet-101 backbone extracts rich features from the input image, while the Feature Pyramid Network (FPN) ensures these features are tailored to different object sizes. This combination allows the model to handle objects of diverse scales effectively. RetinaNet, a loss function specifically designed for SSD, is incorporated to improve the model's ability to differentiate between foreground objects (objects of interest) and background clutter. This architecture is well-suited for tasks requiring good accuracy on a range of object sizes, while maintaining faster inference speeds compared to two-stage detectors like Faster R-CNN.[25]

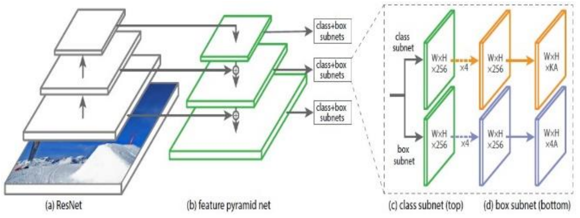

Figure 4 Architecture of RetinaNet[26].

4.3. SSD MobileNet V2 FPNLite

The SSD MobileNet V2 FPNLite 320x320 architecture prioritizes real time object detection on resource-constrained devices like mobile phones. This model combines the strengths of SSD, MobileNet V2, and FPNLite to achieve this goal. Similar to SSD ResNet, it employs the single-stage SSD framework for fast predictions. However, it utilizes the MobileNet V2 backbone, a lightweight CNN designed for mobile and embedded applications. MobileNet V2 achieves impressive accuracy while maintaining low computational cost by utilizing depthwise separable convolutions. The FPNLite network, a smaller version of the FPN, further reduces processing requirements without sacrificing too much accuracy on object detection across different scales. This architecture is ideal for real time object detection tasks on mobile devices where processing power and battery life are crucial considerations[27].

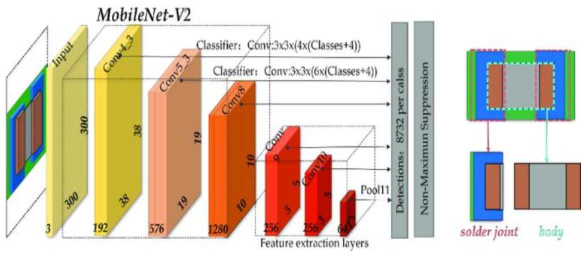
MobileNet-V2
Classifier: Conv:3x3x(4x(Classes+4))
Classifier: Conv:3x3x(6x(Classes+4))
Input
Conv4_3
Conv5_3
Conv8
Conv9
Conv10
Pool11
Detections: 8732 per calss
Non-Maximum Suppression
Feature extraction layers
solder joint
body

Figure 5 MobileNet V2-SSD structure[28].

4.4. Methodology

The intended system seeks to create a real time sign language detector utilizing the TensorFlow (Object Detection) API, trained through transfer learning on a bespoke dataset. Initially, images are captured using a webcam with OpenCV and Python, as outlined in Section 3. Following data acquisition, a labeled map is created to represent all objects within the model, assigning unique labels to each sign (alphabet) along with corresponding IDs. TF records are then generated for both training and testing data, leveraging the binary storage format of TensorFlow for improved performance and training efficiency.

The API of TensorFlow object detection simplifies development, training, and deployment of object detection models. Utilizing the TensorFlow detection model zoo, the system employs the pre-trained EfficientDet D0, SSD ResNet101 V1 FPN (RetinaNet101) and SSD MobileNet V2 FPNLite, which combines various features including the FPN-lite focal loss and feature extractor. The pipeline configuration is adjusted for transfer learning. Notably, the number of classes in the pre-trained model is updated from 90 to 36, aligning with the number of signs (alphabets & numeric) in the custom dataset, changing the batch_size, changing num_steps, total_steps and warmup_steps as to 50,000, 50,000 and 5000 respectively and lastly fine_tune_checkpoint_type classification to detection.

Training the model involves running it for 50,000 steps, with hyper-parameters configured accordingly. During training, the model experiences losses including regularization, classification, and localization losses, reflecting the iterative learning process to optimize performance. The discrepancy in localization loss occurs due to the disparity between the actual values and predicted bounding box adjustment. Following training, the model is fitted from the most recent checkpoint, rendering it prepared for real time detection. After configuring attributes, the model becomes ready for training. Upon loading the latest checkpoint generated from the trained model after training, the model is fully prepared for real time sign language detection.

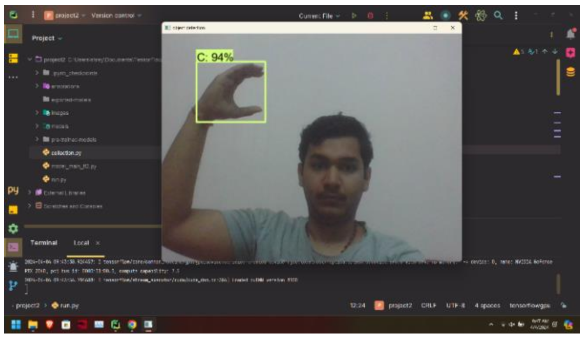

Figure 6 Instantaneous Recognition of Sign Language.

5. Evaluation Through Experiments

5.1. Experimental System and Dataset

The dataset was specifically tailored for Sign Language, with each sign representing an alphabet and numeric value of the ISL English language. The creation of the dataset followed the data acquisition method detailed in Section 3.

The experimentation phase was conducted on a system equipped with an Intel i7 10th generation processor clocked at 4.90 GHz, 16 GB of RAM, RTX 2060 GPU with 6 GB VRAM and a webcam with a resolution of 640x480 and 0.31 MP. The setup utilized the Windows 11 operating system, with 3.10 Python version, OpenCV version 4.9.0, Jupyter Notebook, and the TensorFlow Object Detection API as part of its programming environment.

5.2. Results

The system that has been created can determine Indian Sign Language letters in real time using the TensorFlow object detection API. It employs the EfficientDet D0 512x512, SSD MobileNet V2 FPNLite 320x320 SSD and ResNet101 V1 FPN 640x640 (RetinaNet101) pre-trained models from the TensorFlow model zoo. By using transfer learning on our dataset containing 2000 images (50 - 60 images for each letter), the system has been taught to precisely identify sign language hand movements. During final phase of training, the total loss incurred in above mentioned models can be seen in Table 2. The system's output is determined by its confidence rate, with an average rate of 80%. Each alphabet and numeric letter confidence rate is noted and displayed in Table 1 of the results. Increasing the dataset size can raise the system's confidence rate, thereby enhancing its recognition capabilities and improving overall performance. Despite the limited dataset, our system maintains an average confidence level of 80%.

A	B	C	D	E	F	G	H	I
80%	95%	95%	85%	90%	90%	90%	80%	40%
J	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**
65%	85%	80%	55%	50%	95%	70%	75%	40%
S	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**1**
70%	90%	90%	75%	92%	95%	75%	95%	95%
2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**
95%	90%	95%	90%	90%	95%	90%	95%	95%

Table 1 The Confidence level of each letter.

Model Name	Total Loss
EfficientDet D0 512x512	0.119
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	0.061
SSD MobileNet V2 FPNLite 320x320	0.058

Table 2 Total Loss of Each Model

5.3. Conclusion

Sign language acts as a crucial visual communication tool, particularly for person with disabilities, enabling them to express themselves and connect with others through hand, body, and facial movements. However, the general population's lack of familiarity with sign languages creates hurdles for effective communication. Automated Sign Language Recognition (SLR) systems provide a remedy by converting sign language into spoken language, thereby aiding communication. This paper presents the development and training of an SLR system with the help of TensorFlow API for object detection on the Indian-SL alphabet and numeric dataset. Real time detection of sign language gestures is achieved by capturing images via webcam with the help of python and OpenCV, minimizing costs. The system that has been created demonstrates an average confidence rate of 80%. Despite its high confidence rate, the system's performance is constrained by the small and limited dataset used for training

To enhance the system's capabilities, future efforts could focus on enlarging the dataset to encompass gestures of a wider range. Additionally, the model employed can be swapped with alternative models to explore different performance characteristics. Furthermore, the system's adaptability can be extended to accommodate various sign languages by simply modifying the dataset, thereby broadening its applicability and impact.

6. References

1. Konstantinidis, Dimitrios, Konstantinos Dimitropoulos, and Panagiotis Daras. "Sign Language Recognition Based on Hand and Body Skeletal Data." 3DTV-Conference, 2018-June (2018). https://doi.org/10.1109/3DTV.2018.8478467.
2. Kapur, R. "The Types of Communication." MIJ 6 (2020).
3. Dutta, Kousik K., and Shashank A. S. Bellary. "Machine Learning Techniques for Indian Sign Language Recognition." Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. CTCEEC 2017 333–336 (2018). https://doi.org/10.1109/CTCEEC.2017.8454988.
4. Bragg, Duncan, et al. "Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective." 21st Int. ACM SIGACCESS Conf. Comput. Access. (2019).
5. Rosero-Montalvo, Paúl David, et al. "Sign Language Recognition Based on Intelligent Glove Using Machine Learning Techniques." 2018 IEEE 3rd Ecuador Tech. Chapters Meet. ETCM 2018 (2018). https://doi.org/10.1109/ETCM.2018.8580268.
6. Zheng, Lei, Bin Liang, and Ai Jiang. "Recent Advances of Deep Learning for Sign Language Recognition." DICTA 2017 - 2017 Int. Conf. Digit. Image Comput. Tech. Appl. 2017- Decem, 1–7 (2017). https://doi.org/10.1109/DICTA.2017.8227483.
7. Zhang, Z., and F. Huang. "Hand Tracking Algorithm Based on Super-Pixels Feature." Proc. - 2013 Int. Conf. Inf. Sci. Cloud Comput. Companion, ISCC-C 2013 629–634 (2014). https://doi.org/10.1109/ISCC-C.2013.77.
8. Nikam, A. S., and A. G. Ambekar. "Sign Language Recognition Using Image Based Hand Gesture Recognition Techniques." Proc. 2016 Online Int. Conf. Green Eng. Technol. IC-GET 2016 (2017). https://doi.org/10.1109/GET.2016.7916786.
9. Enikeev, Damir G., and Sabina A. Mustafina. "Sign Language Recognition Through Leap Motion Controller and Input Prediction Algorithm." J. Phys. Conf. Ser. 1715, 012008 (2021). https://doi.org/10.1088/1742-6596/1715/1/012008.
10. Cheok, M. J., Z. Omar, and M. H. Jaward. "A Review of Hand Gesture and Sign Language Recognition Techniques." Int. J. Mach. Learn. Cybern. 2017 101. 10, 131–153 (2017). https://doi.org/10.1007/S13042-017-0705-5
11. Wadhawan, A., and P. Kumar. "Sign Language Recognition Systems: A Decade Systematic

Literature Review." Arch. Comput. Methods Eng. 28 (2019): 785–813. https://doi.org/10.1007/S11831-019-09384-2.

12. Bantupalli, Krishna, and Y. Xie. "American Sign Language Recognition Using Deep Learning and Computer Vision." Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018 (2019): 4896–4831. https://doi.org/10.1109/BIGDATA.2018.8622141
13. Hore, S., et al. "Indian Sign Language Recognition Using Optimized Neural Networks." Adv. Intell. Syst. Comput. 455 (2017): 553–563. https://doi.org/10.1007/978-3-319-38771-0_54.
14. Kumar, Praveen, et al. "Independent Bayesian Classifier Combination Based Sign Language Recognition Using Facial Expression." Inf. Sci. (Ny) 428 (2018): 30–48. https://doi.org/10.1016/J.INS.2017.10.046.
15. Sharma, A., et al. "Benchmarking Deep Neural Network Approaches for Indian Sign Language Recognition." Neural Comput. Appl. 33 (2020): 6685–6696. https://doi.org/10.1007/S00521-020-05448-8.
16. Quocthang, P., N. D. Dung, and N. T. Thuy. "A Comparison of SimpSVM and RVM for Sign Language Recognition." ACM Int. Conf. Proceeding Ser. (2017): 98–104. https://doi.org/10.1145/3036290.3036322.
17. Pu, J., W. Zhou, and H. Li. "Iterative Alignment Network for Continuous Sign Language Recognition." Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (2019): 4160–4169. https://doi.org/10.1109/CVPR.2019.00429.
18. Liang, Z., S. Liao, and B. Hu. "3D Convolutional Neural Networks for Dynamic Sign Language Recognition." Comput. J. 61 (2018): 1724–1736. https://doi.org/10.1093/COMJNL/BXY049
19. About - OpenCV.
20. Albahli, Saleh, and Tahira Nazir. "AI-CenterNet CXR: An artificial intelligence (AI) enabled system for localization and classification of chest X-ray disease." *Frontiers in Medicine* 9 (2022): 955765.
21. Transfer learning and fine-tuning | TensorFlow Core
22. Poster of the Manual Alphabet in ISL | Indian Sign Language Research and Training Center
23. Zhong, Qiming, et al. "EfficientDet: Scalable and efficient object detection." arXiv preprint arXiv:1911.09511 (2020). https://openaccess.thecvf.com/content_CVPR_2020/papers/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.pdf
24. Kim, Hyo Jin, Doo Hee Lee, Asim Niaz, Chan Yong Kim, Asif Aziz Memon, and Kwang Nam Choi. "Multiple-clothing detection and fashion landmark estimation using a single-stage detector." IEEE Access 9 (2021): 11694-11704.
25. Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21-37. Springer International Publishing, 2016.
26. Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In Proceedings of the IEEE international conference on computer vision, pp. 2980-2988. 2017.
27. Sandler, Mark, et al. "MobileNetV2: Improved mobile network architecture with depthwise separable convolutions." arXiv preprint arXiv:1801.04381 (2018). https://arxiv.org/abs/1801.04381
28. Zhou, Jing, Guang Li, Ruifeng Wang, Ruiyang Chen, and Shouhua Luo. "A Novel Contrastive Self-Supervised Learning Framework for Solving Data Imbalance in Solder Joint Defect Detection." Entropy 25, no. 2 (2023): 268.