Classifying Sign Languages and Fingerspellings with Convolutional Neural Networks

Brandon Vu
Computer Science
vubt@stanford.edu

Richard Chen
Computer Science
richchen@stanford.edu

Nyle Wong
Computer Science
nylewong@stanford.edu

Abstract

Sign language was developed by the deaf community and is a language where handshapes and gestures are the main modes of communication. We will be working with fingerspelling datasets which provide handshapes for the letters of a language's alphabet. We trained two models. One model for detecting the bounding box location of a person's hand and another model for classifying the language and letter of an image. We show that we can get 90% accuracy on the test set with the classifier and 96% Average Precision (AP) with the hand detector. We show that we can crop the image using the hand detector and pass that image to the classifier. We show that using the crop and classify method performs 4 times as accurate than without cropping on a new dataset it has never seen before.

1. Introduction

There are many sign languages across the world which some are related and some have been developed independently. American Sign Language (ASL) is one of the most studied sign languages, and it has many existing datasets and models for classifying it. Other languages such as Japanese Sign Language (JSL) or Irish Sign Language (ISL) have much less resources and datasets. We want to build models to address this gap.

We want to facilitate the interaction of deaf individuals across the globe by developing models to recognize the language and letters that a person is signing. In addition, we want to be able to recognize all sign languages in a realistic setting (with background and a person in the image), even for low-resource languages that do not have extensive datasets of signs in realistic settings. The hope of this research is to include more under-resourced sign languages in sign-detection and to show we can apply computer vision techniques to build a robust system for detecting signs.

We first trained a classifier using a convolution neural network (CNN) on a conglomerate dataset of images of fingerspelling from many different languages. The classifier predicts both the language and the letter of the sign in the image. Then, we used a pretrained Faster-RCNN model trained on the COCO dataset and finetuned it on a hand dataset which allows us to predict the bounding box location of hands in an image. We use the hand detector to detect hands in a realistic image with background and crop and center the hand which we then pass to our language and letter classifier. We obtain high performance for both the hand detector and the classifier with the hand detector having 96% AP and the classifier having 90% accuracy on the test set. The end system we develop is one that we can pass in a realistic image and "ignore" the background by cropping the hand out and classifying it.

2. Related Work

There are many works that produce datasets of sign language fingerspellings. For example, Arabic Sign Language (ArASL), Japanese Sign Language (JSL), and Irish Sign Language (ISL) to name a few [6, 4, 13]. Previous attempts have been made at classifying sign language, even before techniques involving deep neural networks have been developed. One work in 2011 used Histogram of Orientation Gradient (HOG), Histogram of Boundary Description (HBD), feature extraction, and a linear SVM classifier to achieve 98.1% accuracy on the dual-handed Indian Sign Language [8]. With the rise of Neural Networks (NNs), we can create more robust models that can extract features automatically. One work used synthetic images created from 3D hand models and real images from students to generate an 8000 image dataset of Japanese Sign Langauge (JSL). In their approach, they used Convolutional Neural Networks (CNNs) and achieved 98% accuracy [5]. The dataset contained handshapes with black backgrounds, which does not resemble realistic contexts for signs where there could be non-black backgrounds. In contrast, our work builds on this dataset, and incorporates others that have a variety of backgrounds.

Another recent work used a dataset of ASL videos and a 3D-CNN based neural network to achieve 96% precision [16]. Although using 3D data captures more information, we did not want to rely on specialized hardware to capture 3D data because it is less accessible, which is why we stick with 2D CNNs and 2D image data. Fast recognition of signs is important for real-world use, so some researchers have tried to train light-weight models for classifying ASL [14]. We take this into account when choosing our model. Our work differs from the normal approach of sign detection models because we are building one model that can classify both the language and the letter instead of just letters of a single language [11].

We are using a hand dataset with images from EgoHand to train our hand object detector. The original EgoHand paper implemented several region proposal methods which included sampling methods and Selective Search. Our work will utilize Faster-RCNN which includes a Region Proposal Network (RPN) which reduces the overhead of region proposal and improves computation time [2].

There has been work that has explored the benefits of object-based classification over image-based classification. One work studied the classification of drain blockage through images and removed edges around regions of interest in order to focus the feature extraction on the object itself [12]. They showed that this improves the classification accuracy. In a different study that used UAV images, they compared the method of segmenting objects in the image first versus not segmenting at all, and they found that segmenting the image improved the classification accuracy [17]. In another work that used satellite imaging, they show that classification accuracy using object segmentation is capped by the segmentation accuracy [10]. Our work utilizes the general conclusion that isolating the object before classifying it improves the classification accuracy. Instead of using segmentation, we use a bounding box detector which accomplishes a similar goal and is much easier to train.

3. Methods

The process of classifying the sign language and letter in a general image is divided into two models. First, an object detection model is used to identify hands and localize them within the image. Second, the hand in the image is cropped according to its bounding box. Finally, the second model classifies which letter of what sign language is present.

3.1. Hand Detection Model

The object detection model is trained on both the EgoHands dataset [2] and David Lee's ASL dataset [7]. The EgoHands dataset contains images with bounding boxes of multiple hands in a variety of backgrounds and environments. David Lee's ASL dataset contains images with bounding boxes for single and often contorted hands (from signing ASL) also in a variety of backgrounds. The datasets were merged and the model was trained on the merged set to expose it to not only regular hands in random environments, but also signing hands, which may take on unconventional shapes. The annotations for the merged dataset were modified to only classify hands since the constituent datasets were originally given more specific annotations such as the type of hand in EgoHands (left or right hand, my or your hand) or the type of letter in the ASL set (A, B, ...).

The goal of training this hand detection model is to localize the hand before feeding into a secondary model for the actual language and letter classification. This helps minimize noise in the background that might confuse the model, and allows this method to be more generalizable for low-resource languages without robust training data.

The creation of the object detection model was based on the Detectron2 library [18] created by Facebook AI Research, which provides a platform to train and evaluate object detection models. We selected one of the object detection baselines trained on the COCO dataset from their model zoo, namely the Faster-RCNN-R50-FPN-3x model. We chose this model because it provides a strong trade-off between minimizing inference time and maximizing average precision.

This model uses the Faster R-CNN framework for object detection initially introduced by Ren et. al [15]. Similar to its predecessor Fast R-CNN [3], the framework runs the image through a backbone CNN before proposing regions of interest. These regions are then cropped and resized before being forwarded through per region CNNs and then categorized by object type. The primary contribution of Faster R-CNN over Fast R-CNN is solving the computational bottleneck of regional proposals by introducing Region Proposal Networks (RPNs), which is a fully convolutional network that makes predictions for both object bounds and scores at those positions. The RPN functions by having anchor boxes at each point in the feature map and then performing binary classification as to whether the box contains an object. By utilizing the RPN, Faster R-CNN features region proposals with significantly less computational cost.

The model also uses a ResNet-FPN backbone of 50 layers and a 3x learning rate scheduler. The feature pyramid network (FPN) was created by Lin et al. in 2016 [9]. This method takes in the image and outputs several proportionally sized feature maps in a fully convolutional fashion. The 3x learning rate training schedule ensures that the model is not undertrained and sets the number of iterations as well as the learning rate drops accordingly in order to maximize the resulting precision.

3.2. Language and Letter Classifier

After the hand has been identified and localized by the hand detector model, the cropped images are fed into a sec-

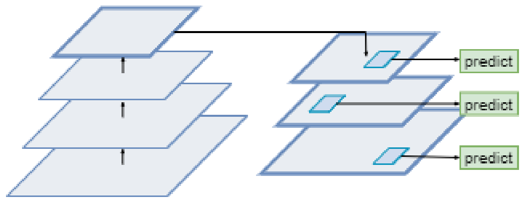

Figure 1: Feature Pyramid Networks

ond model. This second model serves the purpose of classifying which sign language and which letter is being signed by the hand in the image. This model is trained on multiple fingerspelling datasets currently over four different languages. The architecture of this model is visualized in Figure 2, utilizing three convolutional layers with 3x3 kernels, with a 2x2 max pooling after each. This then feeds into two fully connected layers separated by ReLU activation functions before finally a dropout layer is applied with a probability of 0.5.

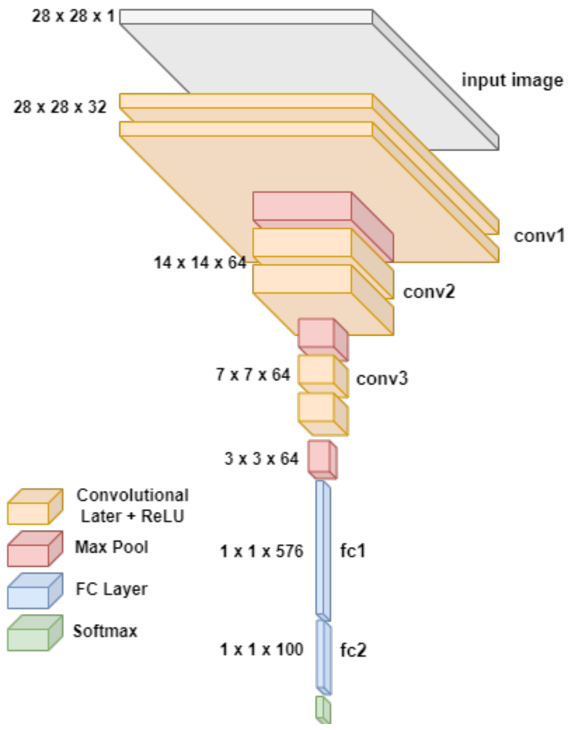

Figure 2: CNN Architecture

3.3. Initial Approach

We originally had two separate models for classifying the language and the letter. We trained one model to classify the language, and then trained another model for each language to detect the language's fingerspelling alphabet. This approach failed for several reasons. First, even though we were able to get high validation and test accuracy, by visualizing the saliency map and testing the model on images out of the dataset, we were able to tell that the model was overfitting to the training dataset. We attributed this to

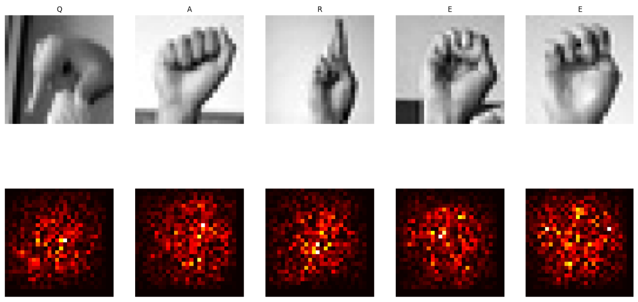

Figure 3: Bad Model Saliency Visualization

the lack of diversity of a single language's dataset. Specifically, all of the backgrounds were the same for a single dataset, so the model was not learning how to distinguish the handshape from the background. To solve this issue, we decided to combine the classification of the letter and language together. We did this by combining the categories in the following way:

Language	Category
ASL [1]	0-25
Japanese [4]	26-66
Irish [13]	67-92
Arabic [6]	93-124

This allowed us to predict the language and the letter of a fingerspelling simultaneously. In addition to making the data more diverse with a variety of backgrounds, this also saved training time because we did not need to train individual models for every language.

4. Datasets

We used many datasets for this project. First, we trained our language and letter classifier using the following fingerspelling datasets: American Sign Language (ASL), Japanese Sign Language (JSL), Irish Sign Language (ISL), and Arabic Sign Language (ArASL) [1, 4, 13, 6]. Second, we trained our hand detector model using the EgoHands dataset which contained images labeling the bounding box location of the hands in the image [2]. We noticed that the EgoHands dataset did not have a diverse set of hand shapes, so we added images from another dataset that had other handshapes [7]. To obtain out-of-distribution metrics for our classifier on different backgrounds, we used a dataset that contained both bounding box labels and letter labels for ASL with a variety of backgrounds. Ideally, we would have out-of-distribution metrics for images in realistic settings of all languages, but we settled for using ASL. We used an 80-10-10 train, validation, test split for training the models.

In order to standardize the images across the different datasets for our language and letter classifier, we applied a variety of preprocessing operations. We preprocessed all images to be grayscale with dimensions of 28x28. Additionally, we noticed that there were many images in which the hand was not centered. To preprocess the data further, we initially used an OpenCV contour finding method to find a bounding box of the hand. This only worked for images where there are no other objects in the image. For the other cases, we used our hand detector model to identify the bounding box of the hand. With the bounding box defined, we cropped and shifted the image to center the hand. Finally, we used the same number of images for each sign language and also used a uniform distribution of images over the different languages. We provided an example image from our dataset of 32,220 images in Figure 4. Finally, a final detail of our dataset is that we chose to only include static signs and omit signs that involve motion. For example, the signs for *j* and *z* were excluded from the ASL dataset.

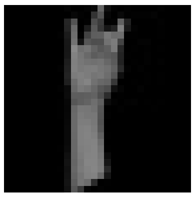
Figure 4: Irish k-sign

The following is our distribution of data from the 6 datasets.

Dataset	# images	# categories	Resolution
ASL [1]	8055	24	28x28
Japanese [4]	8055	41	28x28
Irish [13]	8055	24	28x28
Arabic [6]	8055	32	28x28
EgoHands [2]	15,053	1	720x1280
ASL [7]	1728	26	384x384

5. Experiments, Results, Discussion

We performed many experiments with models, hyperparameters, and datasets in order to arrive at our final models.

5.1. Model Design and Hyperparameters

For the hand detection model, we trained the classifier with an Adam optimizer and cross entropy loss with a learning rate of 0.0001, training for 4 epochs, and a batch size of 10. We experimented with the different number of epochs, and 4 was the optimal amount.

For the sign classifier, we experimented with a multitude of ways of combining convolutional layers, max pooling, batch normalization, and linear layers to build a strong and robust model that would not overfit to the training data. This proved to be a challenge as the datasets used in training were extremely engineered, meaning that they were more artificial and not truly representative of what we might see in the real world.

We experimented with 1-4 convolution + ReLU layers, which showed that 3 layers was optimal for this model. This then fed into 2 linear layers, where the hidden layer was set to a size of 100 to act as a transition between the convolutional layers and the output class. We also experimented with the kernel size, using both 3x3 and 5x5 kernels in the 3 convolutional layers. However, the 3x3 performed better, and the final model utilizes all 3x3 kernels, which may be due to the small nature of the 28x28 input image. To further control for overfitting, we added dropout of 0.5 and tried batch normalization, though it did not seem to improve the model by a significant amount. Finally, we added max pooling after each convolutional layer to add regularization and size down the dimensions before reaching the fully connected layer.

Besides the architecture of the model itself, we found that a learning rate of 0.0001, weight decay of 0.0001, and a batch size of 10 using an Adam optimizer and cross entropy loss was most effective. Finally, in training, we first trained the model for 2 epochs, at which point the loss and accuracy started to plateau at 80%. Then, to further increase the performance, we trained the model further for another 2 epochs with a learning rate of 0.00005, which ultimately achieved a test accuracy over 90%.

5.2. Hand Detector Experiments

The hand detector was trained with a batch size of 8 images for 1000 iterations. To evaluate the detector model, we performed qualitative and quantitative analysis of the bounding boxes on images from the test sets of both the Egohands and ASL datasets; one such example is depicted in Figure 5.

We initially trained the hand detector on the training set of EgoHands and evaluated it on David Lee's ASL dataset which had ASL handshapes in a variety of backgrounds. The performance was poor and there were many instances of incorrect bounding boxes and missclassifications. We attributed this to the fact that the EgoHands dataset had a limited set of handshapes since most of the hands were either open or closed but not in many other configurations. To mitigate this problem, we augmented the EgoHands dataset with images of ASL handshapes from the training set of the ASL dataset, and the performance improved signficantly

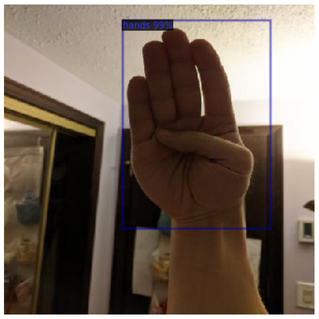

Figure 5: Bounding Box generated by Hand Detector

where we could identify almost all of the hands in the ASL test set. The small minority of images we could not detect were where the hand was in a perspective that was uncommon in the training set. Figure 6 shows an example of this where the distance and angle of the hand contributes to the missed detection.

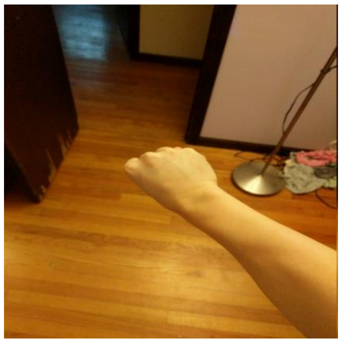

Figure 6: Missed Hand Detection

6. Results

The primary metrics we used were accuracy and Average Precision (AP). Accuracy refers to the number of correctly predicted labels of all categories divided by the total number of predicted labels. Average Precision is a metric used to commonly evaluate object detection models, and it combines precision and recall. Precision is calculated by the number of correct predictions divided the total predictions of a specific category. Recall is calculated by the number of times an image is predicted as a category divided by the ground truth amount of times the image should have been predicted as that category. Average Precision is calculated by taking the Area Under the Curve (AUC) of the precision and recall curve, and it is one metric to tell how well the object detector is performing.

For the sign classifier, results indicate strong performance for classifying the letter of the image across many languages in the dataset. The high test accuracy is good for the model, but it also indicates that the dataset may not be diverse enough.

Validation Accuracy	Test Accuracy
90.22%	90.25%

Table 1: Language and Letter Classifier Results

mAP(0.5:0.95)	AP(0.5)	AP(0.75)
66.447%	96.071%	80.632%

Table 2: Hand Detection APs on Validation Set

The AP scores are high for the Intersection of Union (IOU) threshold of 0.5, but for the higher thresholds, the average precision seems to drop. Compared to the original EgoHands paper which reports 0.826 AP, our model performs close or better for IOU thresholds 0.5 and 0.75.

mAP(0.5:0.95)	AP(0.5)	AP(0.75)
66.508%	94.661%	82.385%

Table 3: Hand Detection APs on Test Set

The test set AP scores indicate the model is doing well and not overfitting to the training dataset.

Lastly, we compare the classification model after hand detection cropping versus the classification model without cropping. On a completely new dataset which the classifier has never seen before, the model performs with 8% accuracy with cropping and 2% without. The low accuracy can be attributed to the different and diverse angles, lighting, and background of the dataset compared to the classifier's training dataset. Even though we combined many datasets into one, we hypothesize that the backgrounds and hands of the classifier's dataset were not diverse enough to classify the out-of-distribution images. Despite the low accuracy, our method still shows that cropping the hand first shows significant accuracy improvements for realistic data with backgrounds that is out of the training distribution.

To qualitatively analyze the performance of our model, we visualized both the saliency maps and the weights of the convolutional layers. We found that the saliency maps indicate that our model is learning the right features of an image. The hot spots detected on the images highlight the characteristic aspects of the signs and distinct patterns in the shape of the hand. This can be seen in Figure 8 on images from an ASL dataset.

However, we also found that the saliency map performs an inverse-like operations on images with dark backgrounds. Instead of highlighting the shape of the hand, Figure 9 shows that the saliency map instead outlines the inverse shape of the hand. The images seen are from the JSL dataset, where the numerical labels correspond to unique JSL signs. This shows that the datasets used can still be

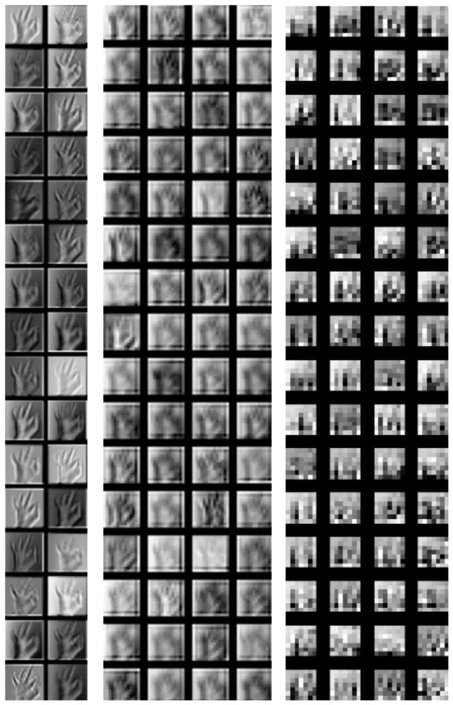

Figure 7: Convolutional Layers Visualization

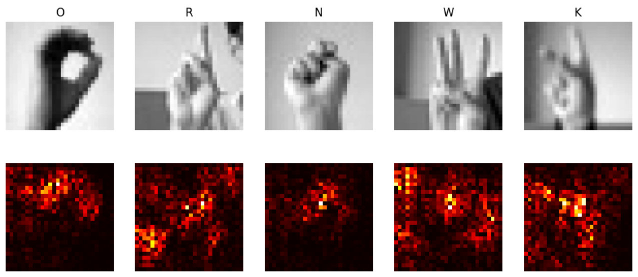

Figure 8: Saliency Map of Combo Model for ASL

more robust in order to counteract the effect of the background. While the model is able to accurately classify languages and signs, the saliency maps show that the background cannot be discounted.

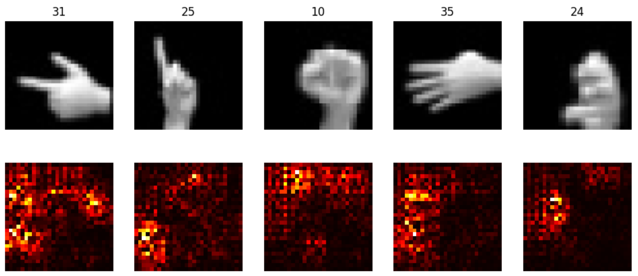

Figure 9: Saliency Map on Images with Dark Backgrounds

As another sanity check for the model, we visualize the weights of each of the three convolutional layers of the model, depicted from left to right respectively in Figure 7. The visualized weights show what we expect to see. The images show different features of the hand being highlighted, with later layers displaying less detail. We can see that the model is making general abstractions from the initial image to the final prediction.

Finally, we measured the computational runtimes for the components of our models (See Table 4). For the hyperparameters we set for the Faster R-CNN hand detector, training took 36 minutes and detection took 0.036 seconds per image. The sign language classifier model took 5 minutes to train and 0.027 seconds to predict on each image.

	Training Time	Inference Time
Hand Detector	36min	0.036s
Sign Classifier	5min	0.027s

Table 4: Time Trials for Training and Detection

7. Conclusion

We developed two high performing models: a model that can detect hands within an image and a model that can classify language and letters of fingerspellings within an image. We showed that through saliency maps and visualizing the convolution layers, our classifier model learned useful features of handshapes. We demonstrated a system to take any realistic image of fingerspellings and combine our two models to predict the language and letter of the sign.

The main limitation we found was the diversity of backgrounds for our classifier datasets. In future work, we would explore using a segmentation model instead of a detection model to segment the hand and crop it from the background entirely. This way, the image the classifier receives closer resembles its training distribution of images, and the resulting classification accuracy can be higher. In addition to this, in the future we would expand the diversity of data by collecting different lighting, angles, and distances of hands from the camera. Data augmentation would also be an avenue to increase our dataset by introducing noise. We also would want to work with video data since many signs in sign language are defined by a gesture instead of a static handshape.

This work illustrates a prototype for a system of sign language translation, and it brings us one step closer towards real-time detection and the development of a machine translation sign language tool.

8. Contributions & Acknowledgements

Brandon led the project data collection and paper writeup and helped with model training. Richard led the

poster design and coding development for the language and letter classification. Nyle worked on training the hand detector model and evaluation in a Colab notebook.

We used Detectron2's object detection tutorial in Colab to work off of and train our hand detector. Saliency map and Convolution Layers visualization code were inspired from Assignments 1 and 2.

References

[1] Sign language mnist: Drop-in replacement for mnist for hand gesture recognition tasks. 3, 4

[2] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2, 3, 4

[3] Ross Girshick. Fast r-cnn, 2015. 2

[4] Hana Hosoe, Shinji Sako, and Bogdan Kwolek. Recognition of jsl finger spelling using convolutional neural networks. *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 85–88, 2017. 1, 3, 4

[5] Hana Hosoe, Shinji Sako, and Bogdan Kwolek. Recognition of jsl finger spelling using convolutional neural networks. *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 85–88, 2017. 1

[6] Ghazanfar Latif, Nazeeruddin Mohammad, Jaafar Alghazo, Roaa AlKhalaf, and Rawan AlKhalaf. Arasl: Arabic alphabets sign language dataset. *Data in Brief*, 23:103777, 2019. 1, 3, 4

[7] David Lee. American sign language letters dataset. 2, 3, 4

[8] Himanshu Lilha and Devashish Shivmurthy. Analysis of pixel level features in recognition of real life dual-handed sign language data set. *2011 International Conference on Recent Trends in Information Systems*, 2011. 1

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016. 2

[10] Desheng Liu and Fan Xia. Assessing object-based classification: advantages and limitations. *Remote Sensing Letters*, 1(4):187–194, 2010. 2

[11] Abu Sayeed Md. Mehedi Hasan, Azmain Yakin Srizon and Md. Al Mehedi Hasan. Classification of american sign language by applying a transfer learned deep convolutional neural network. *23rd International Conference on Computer and Information Technology*, 2020. 2

[12] Bhupesh Mishra, Dhavalkumar Thakker, Suvodeep Mazumdar, Daniel Neagu, Marian Gheorghe, and Sydney Simpson. A novel application of deep learning with image cropping: a smart city use case for flood monitoring. *Journal of Reliable Intelligent Environments*, 6, 03 2020. 2

[13] Marlon Oliveira, Houssem Chatbri, Ylva Ferstl, Mohamed Farouk, Suzanne Little, Noel O'Connor, and A. Sutherland. A dataset for irish sign language recognition. 08 2017. 1, 3, 4

[14] Dhruv Rathi. Optimization of transfer learning for sign language recognition targeting mobile platform. 04 2018. 2

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015. 2

[16] Shikhar Sharma and Krishan Kumar. Asl-3dcnn: American sign language recognition technique using 3-d convolutional neural networks. *Multimedia Tools and Applications*, 2021. 2

[17] Hairie Ilkham Sibaruddin, Helmi Shafri, Biswajeet Pradhan, and Nuzul Haron. Comparison of pixel-based and object-based image classification techniques in extracting information from uav imagery data. *IOP Conference Series: Earth and Environmental Science*, 169:012098, 07 2018. 2

[18] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2