Convolutional Neural Networks for Sign Language Classification

Haley So, haleyso@stanford.edu

January 31, 2021

Abstract

In this study, I will be diving into neural networks for American Sign Language classification, exploring image recognition techniques including data augmentation to improve our CNN.

1 Introduction

Now that we have settled into this new way of living, specifically living with masks and being socially distanced, people who are deaf or those that are hard of hearing have an increased difficulty understanding people. For example, no longer can they read lips since we all have masks on, and even for people who are not hard of hearing, many find themselves shouting to be heard by a socially distanced person. The communication between those hard of hearing and those who are hearing could always have been improved, even before the pandemic. The best case would be for everyone to learn sign language, but this is simply not feasible. As someone who really values communication, I wanted to research machine learning for American Sign Language (ASL) translation to help everyone understand each other better. While the original intent was to use machine learning to translate full dynamic signs to words, I was limited by space and time since training on videos can take months and months, which is significantly longer than the training time on images. For this project, I will be simplifying the to static signs to train my convolutional neural network (CNN). Once the CNN is trained, we can simply input an image and try to classify the sign to translate from sign language to spoken English.

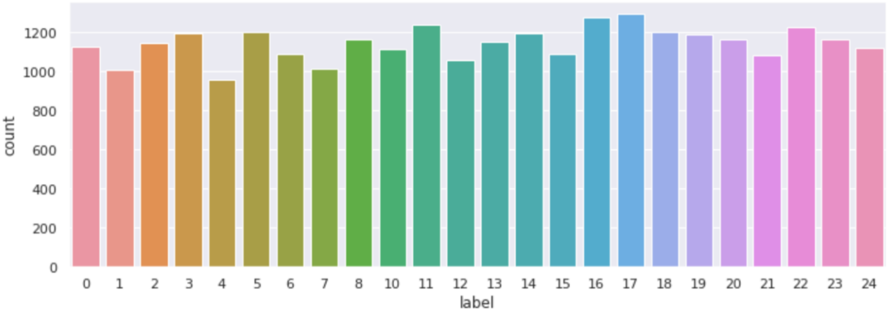
2 Related Work

In past work, researchers have used multiple kinds of algorithms to classify ASL and images in general. Linear classifiers [1, 2] are great places to start but require a lot of preprocessing of data and are limited in finding hidden relationships. In more recent work, neural networks [3, 4, 5] have been used to combat this and are able to extract relationships and information that we might never have thought about. And then moving to convolutional neural networks [6, 7], we see the success especially in these image classification problems. There have even been works for classifying dynamic signs from video based on CNNs and RNNs [8, 9]. Some take into account facial cues or mouthing as well. There's also been important work on creating these datasets [10, 11]. Also very exciting is using the kinect or sensors and hardware for data, as opposed to looking at images [12]. In this study, I focused on CNN for image classification.

3 Dataset and Features

For this project, I used the Sign Language MNIST dataset, a dataset similar to MNIST's handwritten digits image dataset, except the images are isolated signs of the ASL alphabet. Each image is a 28x28 pixel image with grayscale values between 0 and 255, so essentially, each image had 784 pixels. The dataset included all letters except for j and z, since those required motions, and each image also had their labels of course. Here are a few samples of what the images look like.

In the dataset overall, there were 34,627 images where I took about 20% to be my test set, yielding a size of around 7000 images. The remaining 80% of the data, I took a 70/30 split for training and validation datasets at a few random states. To get a better look into the data, we can see if we have any underlying distribution of the data. We can see it's is pretty uniformly distributed.



It's important to note that signers do not spell out every single word, but full words tend to be captured through motions, which we are not able to cover using static images. As a result, the number of datasets I could use was limited since most of the datasets available were for videos. I did download videos from the American Sign Language Lexicon Video Dataset (ASLLVD) and tried to extract a single image in the middle of each video to represent the particular sign, but knowing where to take the photo was difficult so I will continue this approach or use video in the future.

4 Methods

After the literature review, it became pretty clear that the state of the art algorithm for image classification is with a neural network, specifically with a convolutional neural network. One might recall from class how the regular neural networks work, but sometimes in cases like these where the spatial relation between pixels is important, the CNN is introduced since it allows us to maintain spatial information. A CNN can be broken down into a few key features. The convolution part is basically sending through little filters across the whole image that pick out the salient features. They preserve the relationships between pixels and produces a filtered image or a feature map where high scores are given to good matches. Here are a few examples of filters from my project, though I used 3x3 filters so it's not evident what they are picking out.

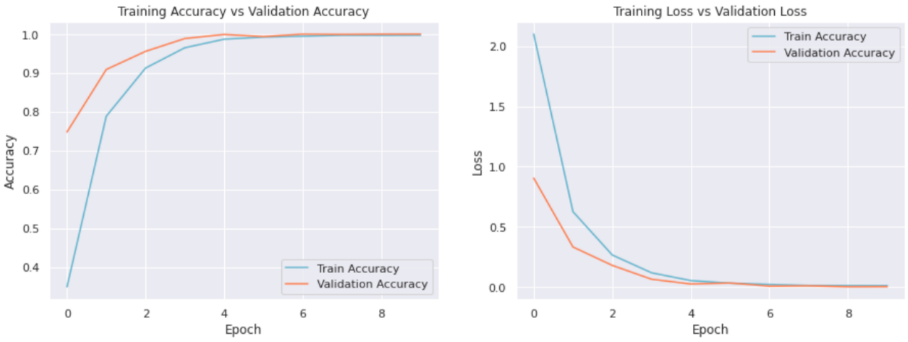
Next, there are Max Pooling layers that basically down-size the data, taking the most important parts, shrinking the image size. To avoid overfitting, dropout layers are often applied, which essentially works by setting edges to 0 at random throughout training. And of course, we have flattening layers that flatten an input to a single vector, and our regular dense layer that is a fully connected layer.

The specific architecture I ended up with was inspired by the VGG16. In my experimentation, I tried a few architectures and number of layers, filter sizes, dropout rates, but there were many parameters to tweak so I turned to papers for inspiration. The final model I ended up with featured a 2D conv layer with 64 filters of size 3x3 followed by a MaxPooling layer, another 2D conv layer of 128 filters, a dropout of 0.2, another maxpooling, a 2D conv layer of 256 filters, maxpooling, and then a fully connected layer with 512 units fully connected to a layer of the number of classes. The activation functions were all relu, except for the last layer, which was a softmax, which is just a generalization of the logistic function in multiple dimensions.

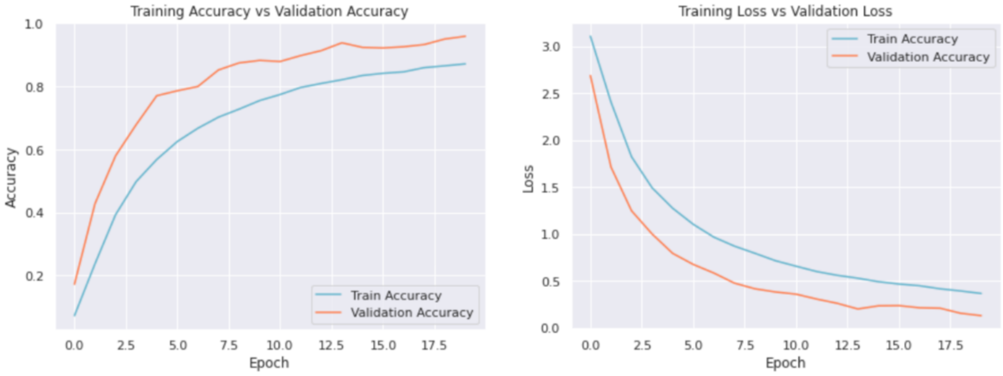
While we have a great dataset, I also explored using other techniques for image classification, including the use of data augmentation to increase the size of our dataset and to vary the images. One can rotate images, shift up/down/right/left, change the lighting, zooming in, flipping, and so many other transformations to increase our dataset and make our model more robust. Because this specific problem with sign language classification is not the standard image classification, I was careful in choosing how the images were transformed when artificially creating a larger dataset. For example, I didn't want to completely rotate 90 degrees since this rotation could produce a completely different sign. In this case, I allowed the images to be rotated up to 30 degrees, which is realistic because humans themselves don't sign at the same angle all the time. I allowed for shifting and zooming in 10% as well and horizontal flips since signers can be left handed.

5 Experiments & Results

Before I begin, the primary metric is accuracy since this is a classification problem. I used google colab and google cloud to perform my experiments. Before using inspiration from the VGG16 model, the highest accuracy I got on my test data was 83.535%, which isn't bad, but I wanted to achieve a higher accuracy. So with the VGG16 inspired architecture written out in the methods section, I used a batch size of 128 and 10 epochs to train my neural network. By the 10th epoch, the accuracy on the training data was .9970 and my validation accuracy was .9999. This seemed really promising, and with the test data, the model had an accuracy of .91118, which is much better than my original model. In the plots, we can see accuracy continuing to increase and that we are not overfitting, thanks to the dropout layers.

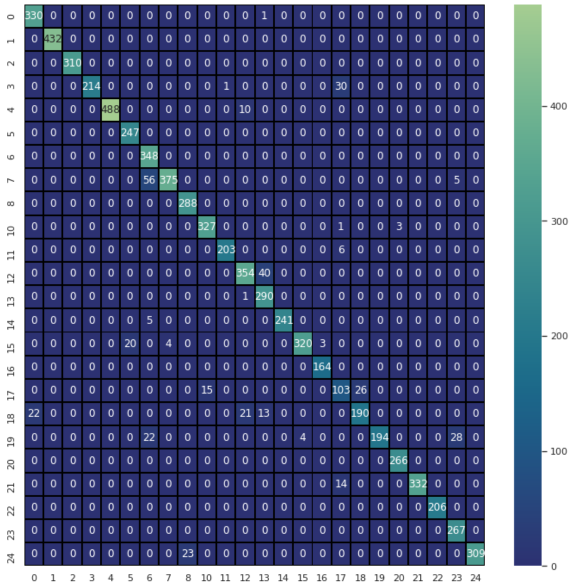


Next, to improve the model, I applied data augmentation, allowing for rotations up to 30 degrees and shifts and horizontal flips. This model I trained with more epochs since my train and validation accuracies didn't reach .99 quickly, whereas the previous model reached .99 within 5 epochs. I used 20 epochs with this new data and ended up with an accuracy of .8715 for the training data and .9591 on the validation data. We can see that since the validation data is not transformed in any way, the accuracy is higher since there aren't wonky cutoffs.



With this model on the test data, I achieved an accuracy of .94785 to .95411 on different runs. This is great and shows that augmenting a dataset will produce a more robust model. To further look into this model with the augmented data, I produced the confusion matrix between the true labels and the predicted labels for the test set. This allows us a visual of how well our neural network works.

To compare the model that accounted for slight rotations and the model that didn't, I tested a few images I took myself in colab. The sign for the letter 'Y' can be seen in the image on the next page. I purposefully tilted my hand a natural amount and checked to see what each model predicted. The model without data augmentation incorrectly predicted 'L,' whereas the other model correctly predicted a 'Y,' showing rotational robustness.



	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	330	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	214	0	0	0	0	0	0	1	0	0	0	0	0	30	0	0	0	0	0	0	0
4	0	0	0	0	488	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	348	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	56	375	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
8	0	0	0	0	0	0	0	0	288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	327	0	0	0	0	0	0	1	0	0	3	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	203	0	0	0	0	0	6	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	354	40	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	290	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	5	0	0	0	0	0	0	241	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	20	0	4	0	0	0	0	0	0	320	3	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	103	26	0	0	0	0	0	0
18	22	0	0	0	0	0	0	0	0	0	0	21	13	0	0	0	0	190	0	0	0	0	0	0
19	0	0	0	0	0	0	22	0	0	0	0	0	0	0	4	0	0	0	194	0	0	0	28	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	266	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	332	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	206	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	267	0
24	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	309

Figure 1: Confusion Matrix

For a brief look into points of failure in earlier models, I tried to use a regular neural network, but it didn't capture the spatial relation. At one point, I was also overfitting my CNN so that's how I learned about dropouts. Knowing how to form my layers was a learning process since you could start with a ton of filters and they can be large or small, but in playing around, I found smaller filters worked better than larger filters in general, which intuitively makes sense. I also noticed using odd sized filters, as opposed to even sized filters like 2x2 or 4x4, were easier to use, probably due to the fact with the even filters, we may have distortions that arise across layers.

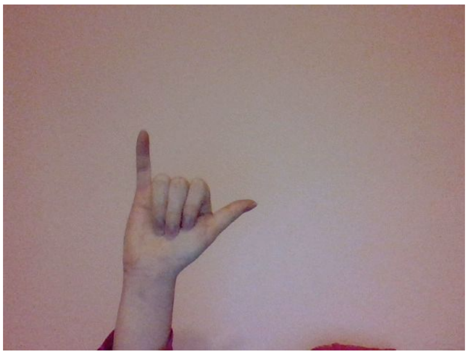


Figure 2: Tilted sign for letter Y

6 Conclusion & Future Directions

In this study, we looked into using neural networks for sign language classification. We saw the merits of using the convolutional neural network when it comes to images that allow for spatial relations between pixels to be captured. We went from a vanilla neural network, to a convolutional neural network, to one inspired by the state of the art VGG16, and we added data augmentation to produce a model that accurately predicted on 95% of the test data. And in the specific example image, we saw how the robust augmented model correctly classified a tilted sign, whereas the non-augmented model did not.

6.1 Continued Work

As an electrical engineer, I wanted to incorporate some hardware, so I actually ordered some parts and have been working on using a few kinds of sensors to make a glove that can take in values for a given sign. The goal was to create my own dataset in a way, similarly to taking photos, but now our values are the outputs of the flex sensors, the accelerometer, the gyroscope etc. After training a model on this data, I'd be able to connect the model to the glove through a microcontroller for live translation. This is still in the works though. But for a critical look at the usefulness of this project, for this to be useful and not just a clunky glove, there's a long way for it to go.

6.2 Future directions

As mentioned earlier, the original goal I had in mind was to recognize dynamic hand motions and translate them into spoken english. Due to limitations of space and time, I ended up working with static signs, and even with just static signs, there weren't really datasets available to work with, so I had to resort to the alphabet or taking frames from videos. What makes this problem interesting is that ASL actually has its own sort of "grammar" so the extracted words aren't the words we would say in spoken english. So if I had more time and experience, I would like to first do video or motion recognition on dynamic signs, and then look into language processing to translate from ASL "grammar" to spoken English.

References

[1] Singha, J. and Das, K. “Hand Gesture Recognition Based on Karhunen-Loeve Transform”, Mobile and Embedded 232 Technology International Conference (MECON), January 17-18, 2013, India. 365-371.

[2] D. Aryanie, Y. Heryadi. American Sign Language-Based Finger-spelling Recognition using k-Nearest Neighbors Classifier. 3rd International Conference on Information and Communication Technology (2015) 533-536.

[3] L. Pigou et al. Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision 6-12 September 2014

[4] P. Mekala et al. Real-time Sign Language Recognition based on Neural Network Architecture. System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium 14-16 March 2011.

[5] Y.F. Admasu, and K. Raimond, Ethiopian Sign Language Recognition Using Artificial Neural Network. 10th International Conference on Intelligent Systems Design and Applications, 2010. 995-1000.

[6] Raimundo F. Pinto Jr., Carlos D. B. Borges, Antônio M. A. Almeida, and Iális C. Paula, Jr., “Static Hand Gesture Recognition Based on Convolutional Neural Networks,” Journal of Electrical and Computer Engineering, vol. 2019, Article ID 4167890, 12 pages, 2019.

[7] Runpeng Cui, Hu Liu, and Changshui Zhang. Recurrent Convolutional Neural Networks for Continuous Sign Language Recognition by Staged Optimization. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 7361–7369, Honolulu, HI, USA, July 2017.

[8] C. Cao, Y. Zhang, Y. Wu, H. Lu and J. Cheng, “Egocentric Gesture Recognition Using Recurrent 3D Convolutional Neural Networks with Spatiotemporal Transformer Modules,” In Proceedings of IEEE International Conference On Computer Vision (ICCV), Venice, Italy, 2017.

[9] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1110–1118, 2015.

[10] Y. Zhang, C. Cao, J. Cheng and H. Lu, ”EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition,” IEEE Transactions on Multimedia (T-MM), Vol. 20, No. 5, pp. 1038-1050, 2018.

[11] Ronnie Wilbur and Avinash C. Kak. Purdue RVL-SLLL American Sign Language Database. Technical Report TR-06-12, School of Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47906, 2006.

[12] Syed Atif Mehdi and Yasir Niaz Khan. Sign language recognition using sensor gloves. In Neural Information Processing, 2002. ICONIP’02. Proceedings of the 9th International Conference on, volume 5, pages 2204–2206. IEEE, 2002

[13] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. Video-based sign language recognition without temporal segmentation. In Proc. of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2018.

[14] Oscar Koller, Jens Forster, and Hermann Ney. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. Computer Vision and Image Understanding (CVIU), 141:108–125, December 2015. ISSN 1077-3142. doi: 10.1016/j. cviu.2015.09.013.

[15] Hilde Kuehne, Hueihan Jhuang, Rainer Stiefelhagen, and Thomas Serre. Hmdb51: A large video database for human motion recognition. In High Performance Computing in Science and Engineering '12, pages 571–582. Springer, 2013.

[16] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4207–4215, 2016.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[18] M Ebrahim Al-Ahdal and Md Tahir Nooritawati. 2012. Review in sign language recognition systems. In 2012 IEEE Symposium on Computers Informatics (ISCI). IEEE, 52–57.

[19] Saba Joudaki, Dzulkifli bin Mohamad, Tanzila Saba, Amjad Rehman, Mznah Al-Rodhaan, and Abdullah Al-Dhelaan. 2014. Vision-based sign language classification: a directional review. IETE Technical Review 31, 5 (2014), 383–391.

[20] Hamid Reza Vaezi Joze and Oscar Koller. 2018. MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language. arXiv:1812.01053 [cs] (Dec. 2018).

[21] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[22]