Sign Language Classification using Deep Learning

Group 24: Minting Chen, Tianyi Gao, Xiaoyang Pan

Background

Sign language is a useful tools for many people with listening/ speaking difficulties; however, it is not commonly understood by people. Sign language can help building communication as people can spell out each individual letter of words, which can be easily understood.

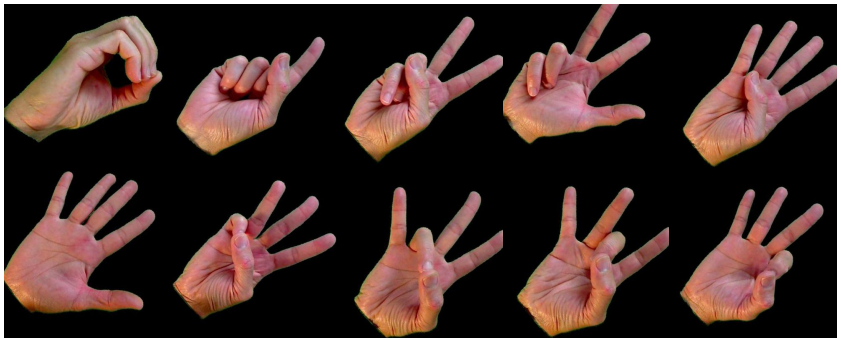


Figure 1. American Sign Language 0-9

Background

Although the accuracy of recognize the sign language is high by wearing sensors for recognition, the equipment is still expensive and not easy to carry.

Therefore, it is important to develop a sign language identification through machine learning to provide better communication.

Literatures

- ASL classification has been studied for over two decades
- Three main classifiers that used to solve the problem:
  - Bayesian Classifier - Starner used Hidden Markov Model [3]
  - Linear Classifier - Sharma et al. used SVM + KNN [2]
  - Neural Networks - Brandon used CNN with transfer learning[4]

Literatures

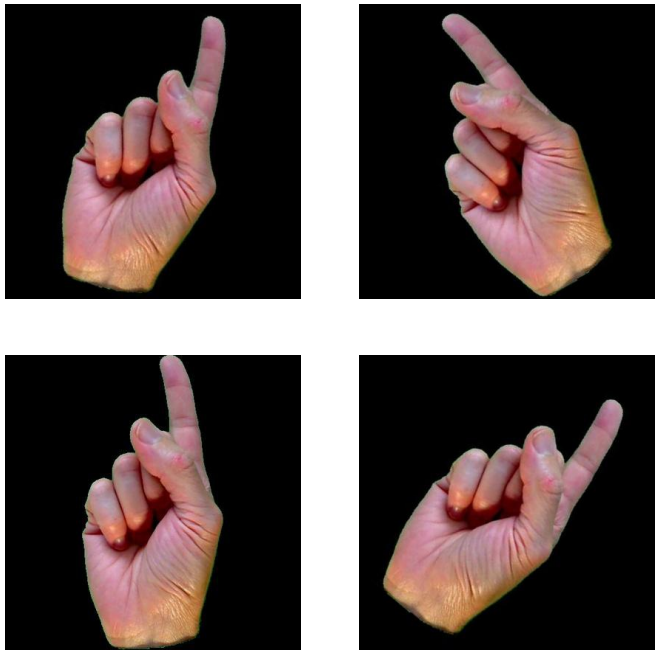
- [1] Bheda built a CNN model instead of using transfer learning
  - Data preprocessing: background subtraction
  - Data augmentation (horizontal flip)
  - A basic CNN model that contains convolutional and dense layers
  - Achieve 82.5% accuracy

How deep learning will solve it?

- Each Alphabet and number has its unique hand shape representation
- CNN model recognizes the most important features to distinguish one from the other.
- Continuous classification of alphanumeric characters describe whole sentences, which everyone can understand

Dataset

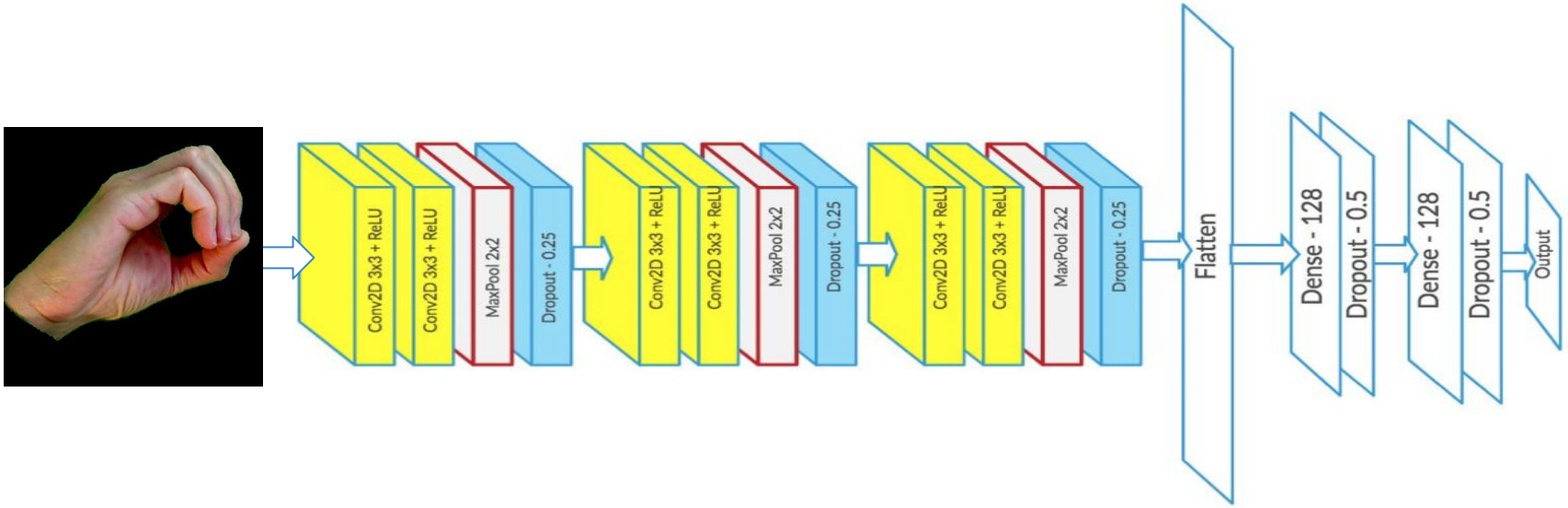
- American Sign Language Dataset
  - 20216 images (400*400 pixels)
  - 37 classes (A-Z, 0-9)
  - Image Background is black
  - Contain rotated images



Data Preprocessing

- Image resize to 50*50 pixels
- Data Augmentation (horizontal flip)
- Used the tensorflow framework
- Training 64%, validation 16%, testing 20%

CNN Model

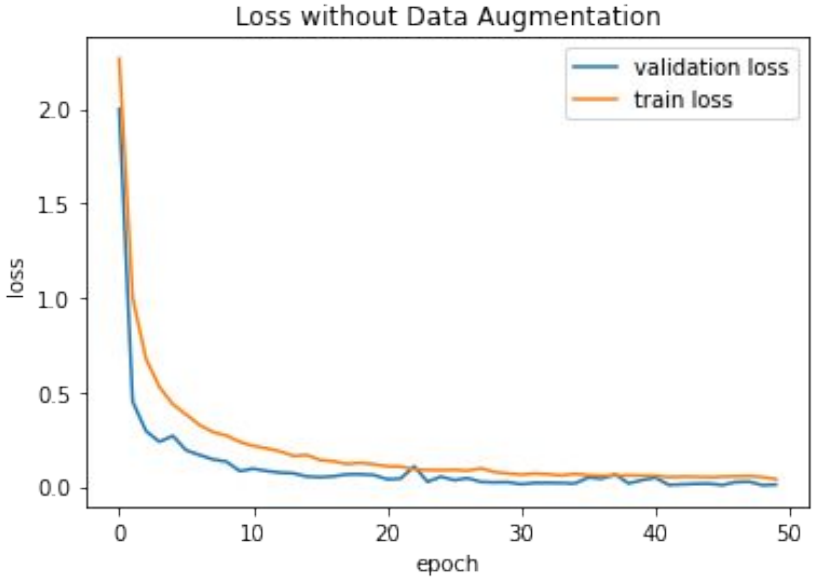


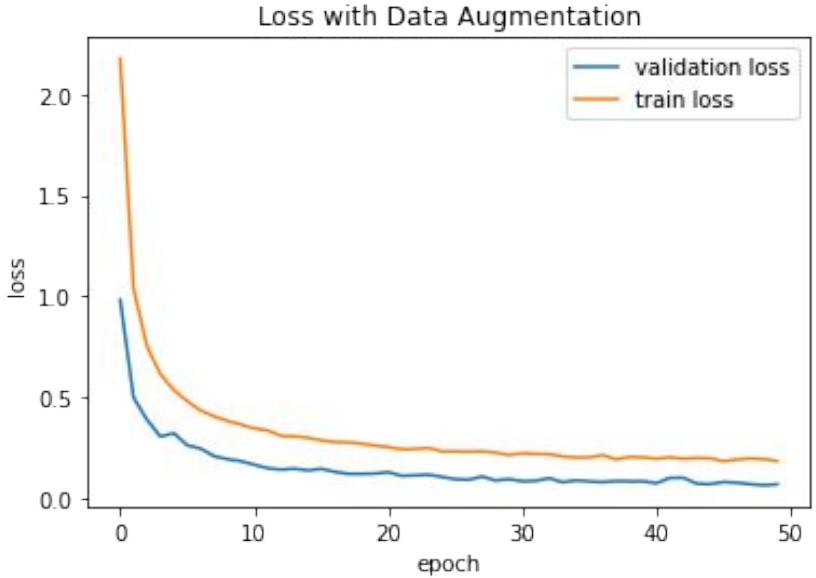
Batch size: 32, kernel size: 3*3, loss function: cross entropy, optimizer: adam

CNN Architecture from [1]

CNN Model Results

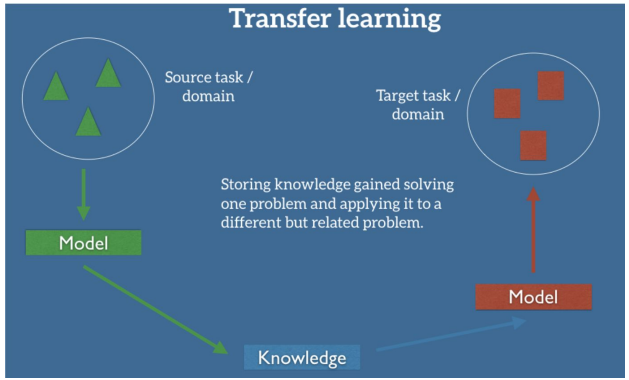
- Without data augmentation accuracy: 99.61%
- With data augmentation accuracy: 96.65%





Transfer Learning

- Utilized pre-trained model for one task to train for a new but similar questions
- Ex: The pre_trained model for VGG-16 learn from over 14 million images contains 1000 classes
- It promotes rapid progress and better performance



The Transfer Learning setup [7]

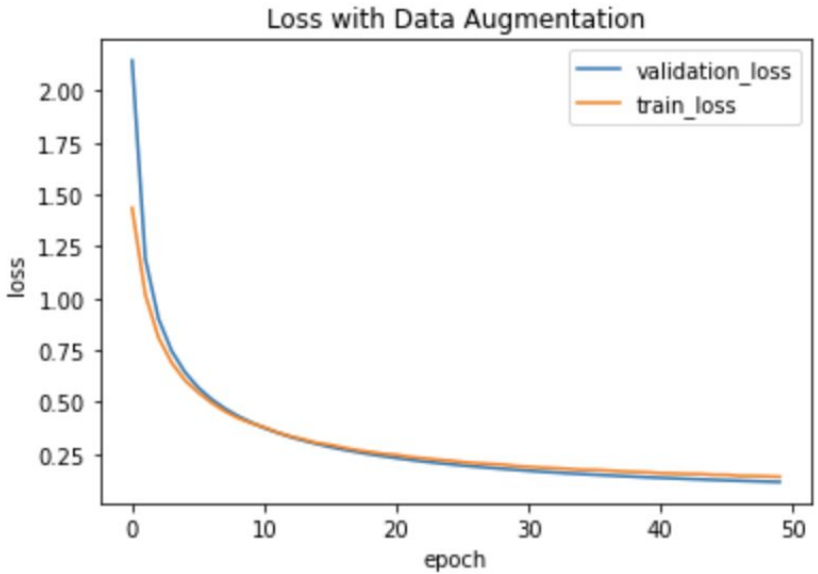
VGG-16

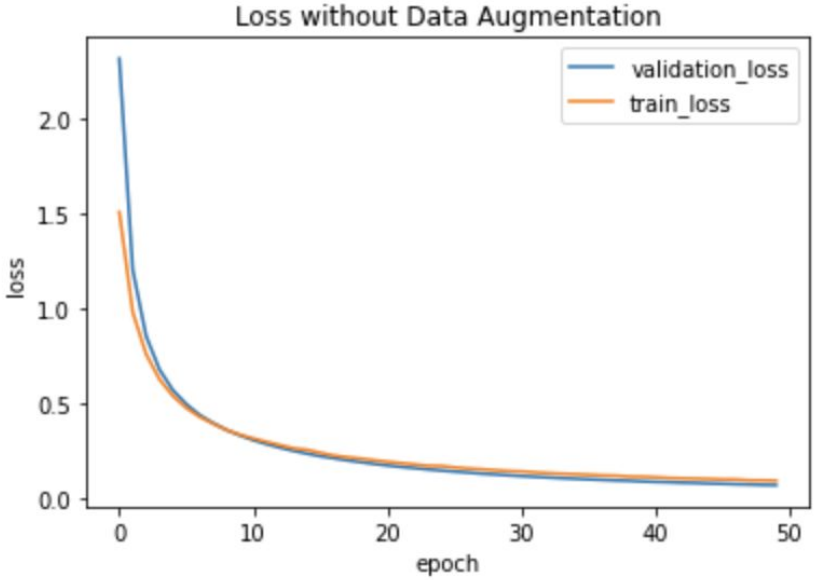
Batch size: 64, kernel size: 3*3, pool_size: 2*2, loss function: cross entropy, optimizer: adam

VGG-16 Architecture from [5]

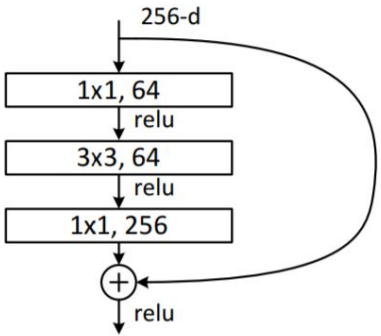
VGG-16 Model Results

- Without data augmentation accuracy: 98%
- With data augmentation accuracy: 96%





ResNet50 Model



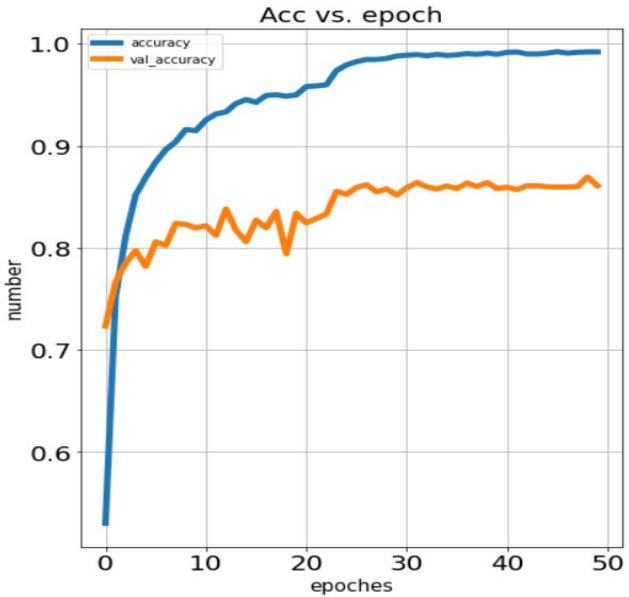
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

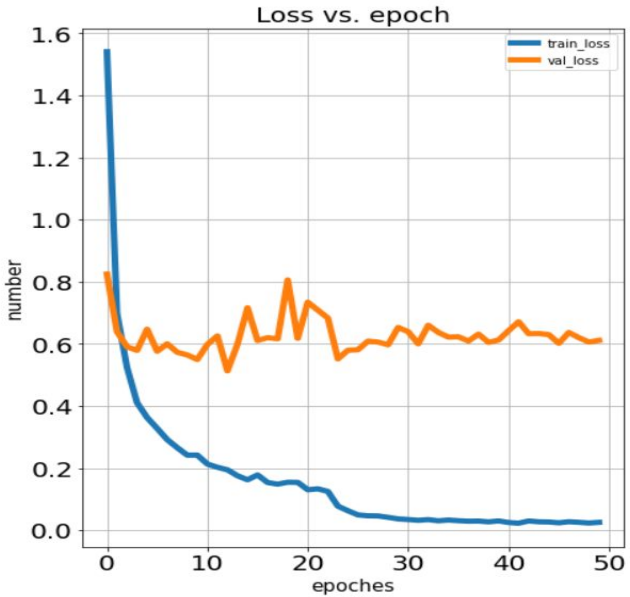
Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet50 Architecture from [6]

ResNet50 Model Results

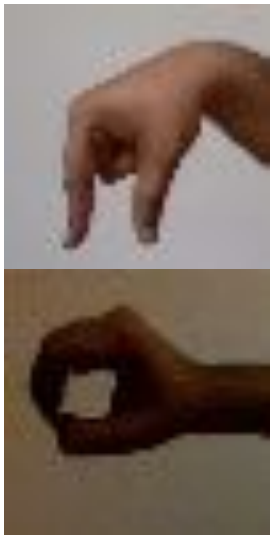
- With data augmentation accuracy: 95%





Further Improvement

- Train and validate the CNN models using Sign Language Gesture Images Dataset
- Sign Language Gesture Images Dataset
  - 55500 images (50*50 pixels)
  - 37 classes (A-Z, 0-9)
  - Images are captured under various lighting conditions
  - Images are very different from ASL dataset



References

[1] Bheda, Vivek, and Dianna Radpour. "Using deep convolutional networks for gesture recognition in american sign language." *arXiv preprint arXiv:1710.06836* (2017).

[2] Sharma, Rohit, et al. "Recognition of single handed sign language gestures using contour tracing descriptor." *Proceedings of the world congress on engineering*. Vol. 2. 2013.

[3] Starner, Thad, and Alex Pentland. "Real-time american sign language recognition from video using hidden markov models." *Motion-based recognition*. Springer, Dordrecht, 1997. 227-243.

[4] Garcia, Brandon, and Sigberto Alarcon Viesca. "Real-time American sign language recognition with convolutional neural networks." *Convolutional Neural Networks for Visual Recognition* 2 (2016): 225-232.

[5] Hassan, Muneeb. *VGG16 - Convolutional Network for Classification and Detection*, 24 Feb. 2021, neurohive.io/en/popular-networks/vgg16/.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[7] Sebastian Ruder. (2020, June 15). *Transfer Learning - Machine Learning's Next Frontier*. Sebastian Ruder. https://ruder.io/transfer-learning/.

[8] CNN model code reference: https://github.com/bhedavivek/deep_asl/blob/master/train.py

Thank you!