ASL to Audio

Autobot: Curtis Lin, Satya Thiruvallur, Param Viswanathan, Hailey Wu
W251 Project, 2020 Spring

Agenda

Background + Dataset

Recorded Demo

Pipeline

Cloud

Edge

Live Demo

Next Step

Background + Dataset

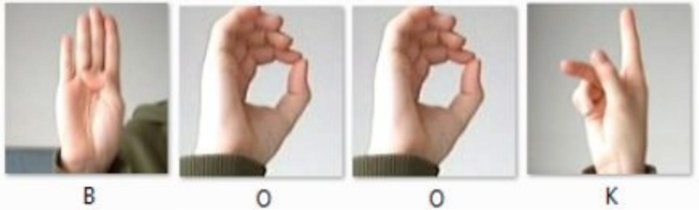
ASL Challenge: Communication requires both parties (or multiple participants) to understand and use sign languages as the prerequisite.

Project Goal: The Autobot team wants to overcome challenges for the deaf-mute community, by converting the sign languages directly to audio.

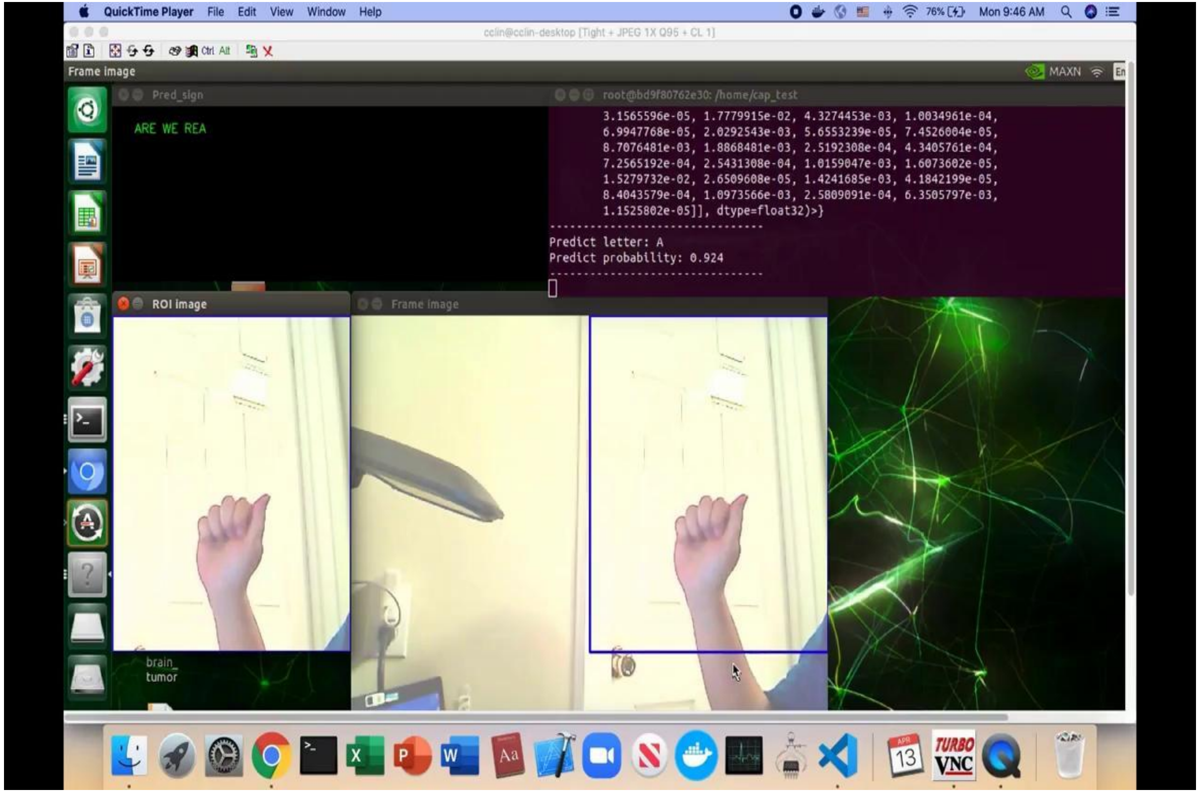
Dataset Options: word-level vs. alphabet-level

Word level

Letter level



Recorded Demo



Dataset selection and implementation plan

Cloud:

Dataset: ASL Alphabet
(1.2 GB; 87000 images;
29 classes)

Cloud VM
(Model training: VGG16, Resnet 50,
Custom Vision, AutoML)

Edge (Jetson TX2):

USB
camera

Inference on Edge
(pretrained model:
Keras, OpenCV)

Pipelines:

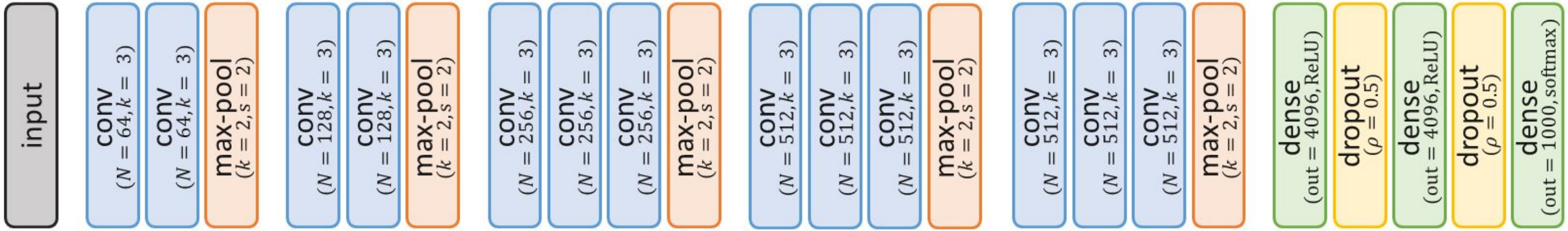
1. Real-time video capturing
2. Hand sign image collection
3. Sign prediction
4. Text printing
5. Save the audio of a sentence

CLOUD

Cloud – Model Architecture

Approach	**Strengths**	**Weakness**
Build from scratch	❖ Most flexible architecture ❖ Freedom to tune parameters ❖ Shot at greatness (*SatyaNet*)	❖ Longer development time ❖ Computational resource limitation ❖ Reinventing the wheel
Bottleneck features (SOTA)	❖ Leverage features trained on ImageNet ❖ Performant ❖ Faster dev time	❖ Higher Accuracy but can go higher ❖ Features may not be relevant
Fine-tune top layer (SOTA)	❖ Highest Model Accuracy ❖ Fastest dev time ❖ Incentivized by HW9 losses	❖ Incompatible pre-trained weights ❖ Overfitting by powerful networks

VGG-16 Transfer Learning



❖ Large kernel sized filters replaced by multiple small 3X3 filters

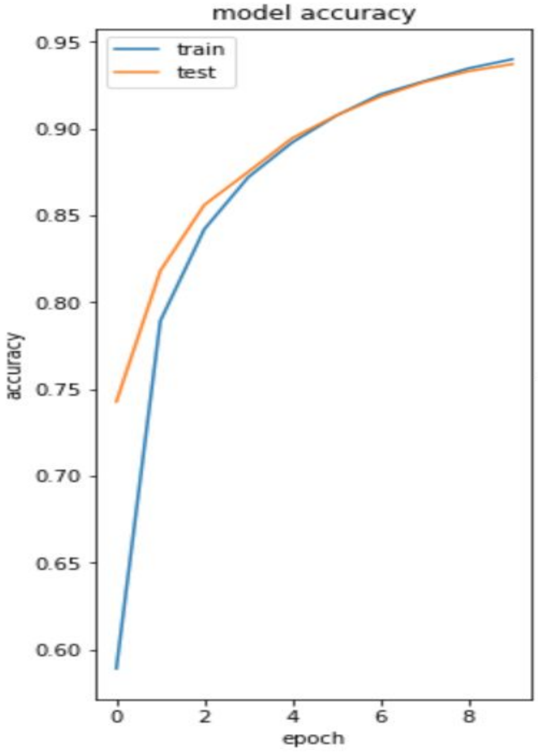
❖ Trained with 87000 50X50 sized ASL images. Labels were One Hot encoded

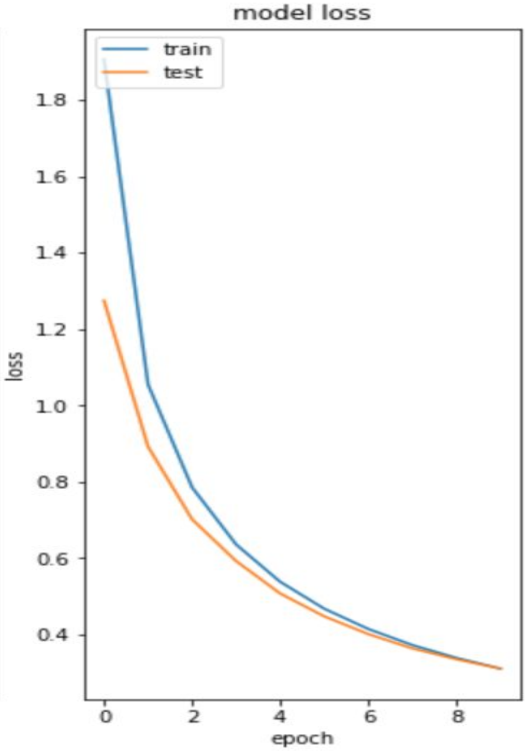
❖ Dimension ordered pre-trained VGG-16 weights used without top layer

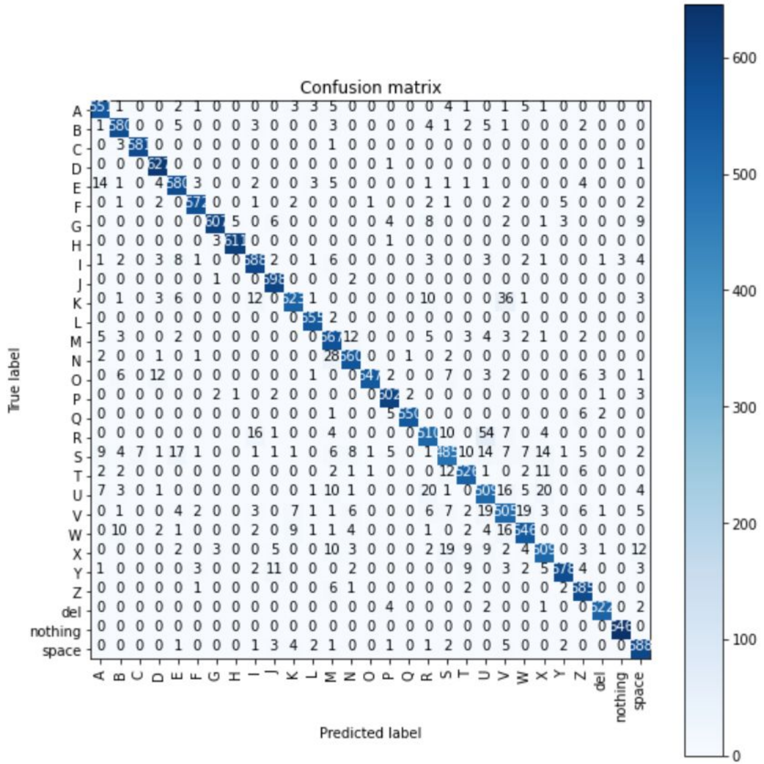
❖ Categorical Cross entropy loss , Adam optimizer

❖ Total params: 14, 729, 565 Trainable Prams: 14, 877

Key Metrics

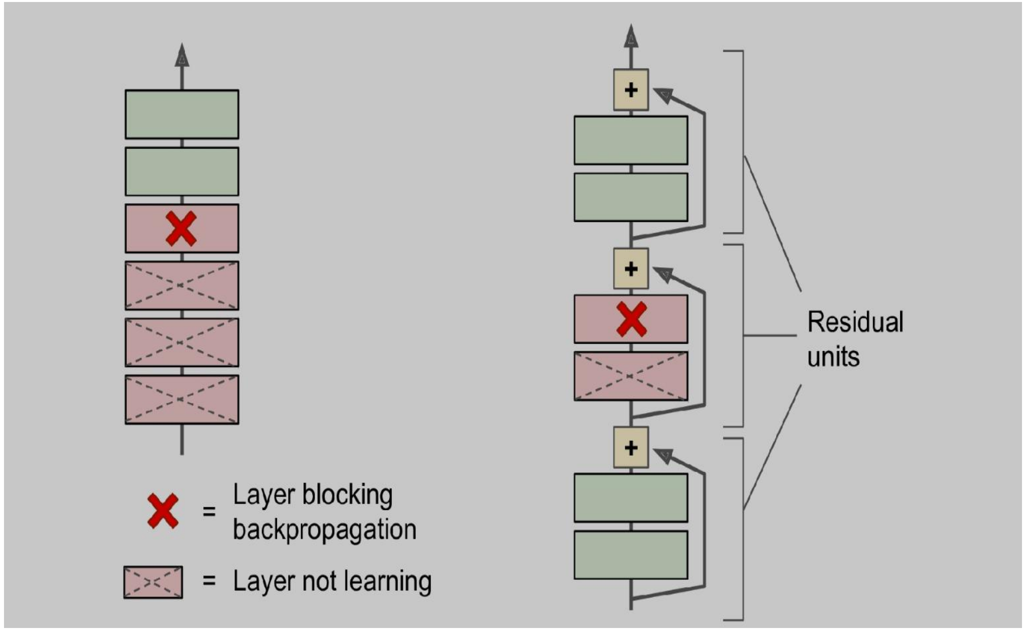

model accuracy
train
test
accuracy
epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
2
4
6
8


model loss
train
test
loss
epoch
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6
8


Confusion matrix
True label
Predicted label
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
del
nothing
space
600
500
400
300
200
100
0

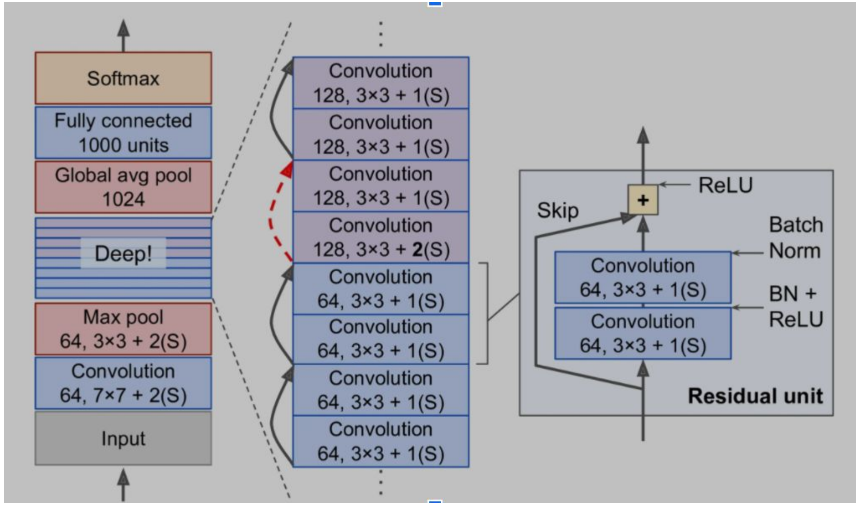
Cloud – Model ResNet50

- Residual Network with 50 layers
- Deep network that uses skip connection strategy
- Stack of residual units (RUs) where each unit is a small neural network with a skip connection



Cloud – Model ResNet50

- Deep stack of simple residual units
- Each unit is composed of 2 convolutional layers (no pooling layer)
- Batch Normalization and ReLU activation
- 3x3 Kernels
- Preserves spatial dimensions (stride=1, “same” padding)



Cloud – Model ResNet50

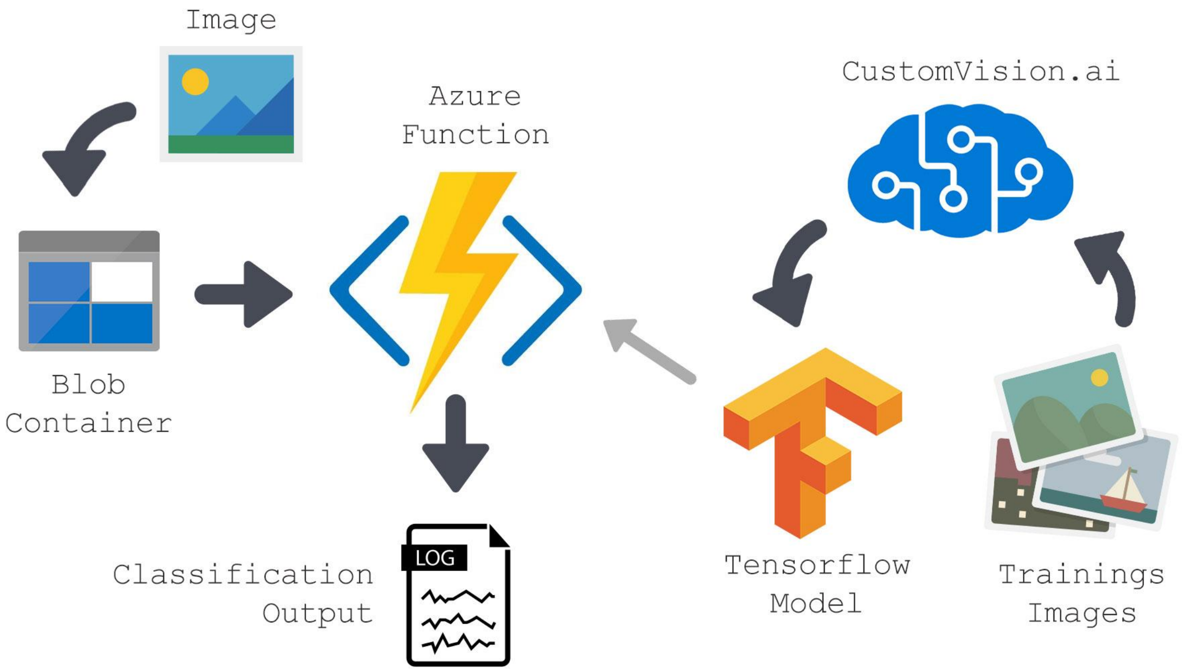
Training Strategy

- Image was pre-processed to a size of 224 x 224 and augmented
- Top layer of the network was replaced with our Pool and Dense layers
- Two runs of 10 epochs each were executed
- First run
  - Weights of pretrained layers were frozen
  - Learning rate = 0.2 and decay = 0.01
  - Accuracy = 65%
- Second run
  - Weights of pretrained layers were not frozen
  - Learning rate = 0.01 and decay = 0.001
  - Accuracy = 98%

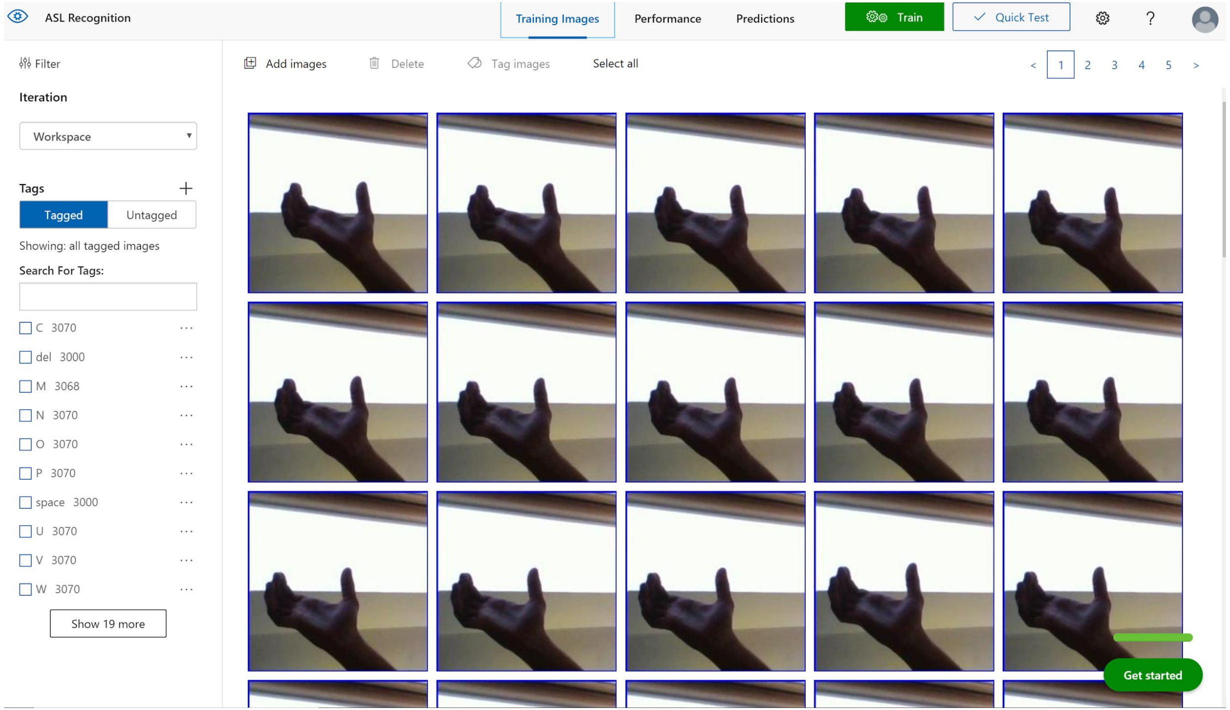
Cloud - Model ResNet50

```
Epoch 1/10
2446/2446 [==============================] - 458s 187ms/step - loss: 0.2329 - accuracy: 0.9489 - val_loss: 0.0624 - val_accuracy: 0.9778
Epoch 2/10
2446/2446 [==============================] - 455s 186ms/step - loss: 0.0033 - accuracy: 0.9997 - val_loss: 0.0535 - val_accuracy: 0.9837
Epoch 3/10
2446/2446 [==============================] - 455s 186ms/step - loss: 0.0018 - accuracy: 0.9999 - val_loss: 0.0586 - val_accuracy: 0.9823
Epoch 4/10
2446/2446 [==============================] - 455s 186ms/step - loss: 0.0015 - accuracy: 0.9999 - val_loss: 0.0518 - val_accuracy: 0.9844
Epoch 5/10
2446/2446 [==============================] - 455s 186ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0571 - val_accuracy: 0.9828
Epoch 6/10
2446/2446 [==============================] - 455s 186ms/step - loss: 0.0011 - accuracy: 0.9999 - val_loss: 0.0548 - val_accuracy: 0.9832
Epoch 7/10
2446/2446 [==============================] - 454s 186ms/step - loss: 9.1674e-04 - accuracy: 0.9999 - val_loss: 0.0660 - val_accuracy: 0.9805
Epoch 8/10
2446/2446 [==============================] - 455s 186ms/step - loss: 8.7085e-04 - accuracy: 0.9999 - val_loss: 0.0522 - val_accuracy: 0.9851
Epoch 9/10
2446/2446 [==============================] - 454s 186ms/step - loss: 8.5705e-04 - accuracy: 1.0000 - val_loss: 0.0553 - val_accuracy: 0.9829
Epoch 10/10
2446/2446 [==============================] - 454s 186ms/step - loss: 7.6152e-04 - accuracy: 0.9999 - val_loss: 0.0534 - val_accuracy: 0.9838
```

Cloud – Custom Vision (Azure)

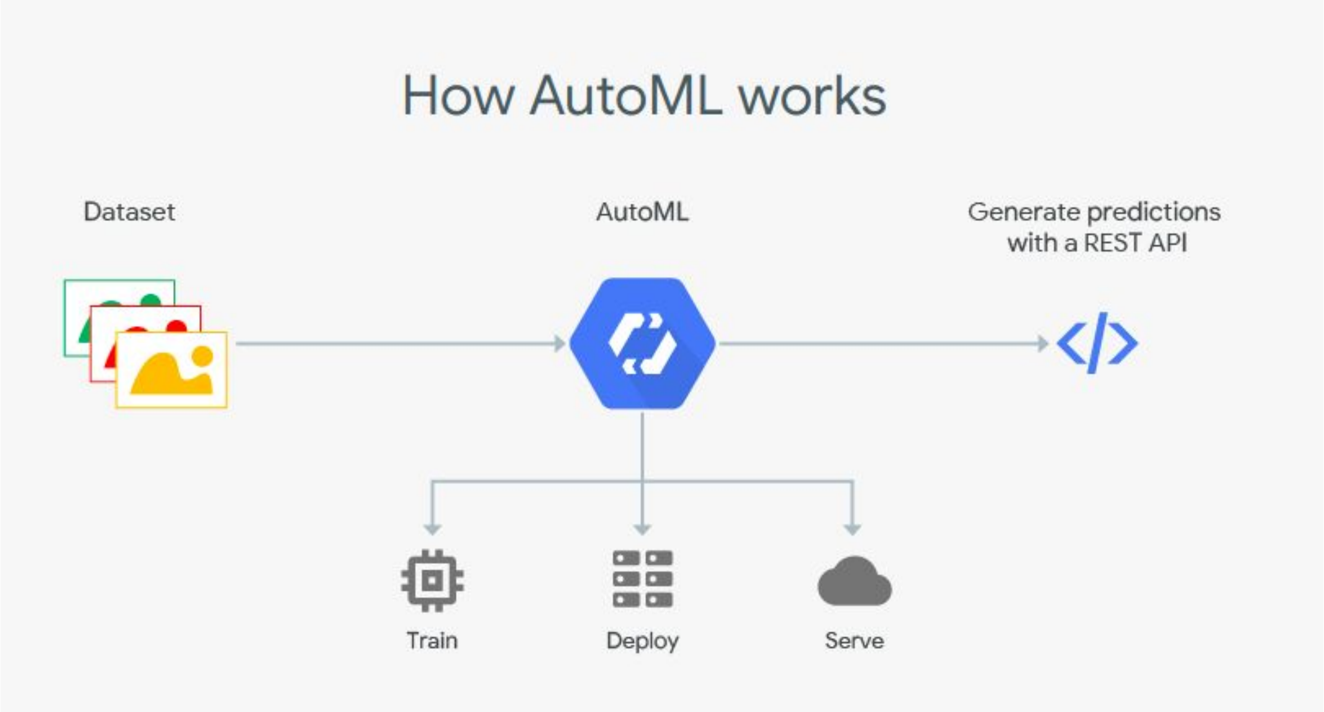


Cloud – Custom Vision (Azure)



Cloud – AutoML (GCP)

Cloud AutoML Vision



Cloud – AutoML (GCP)

Set up process

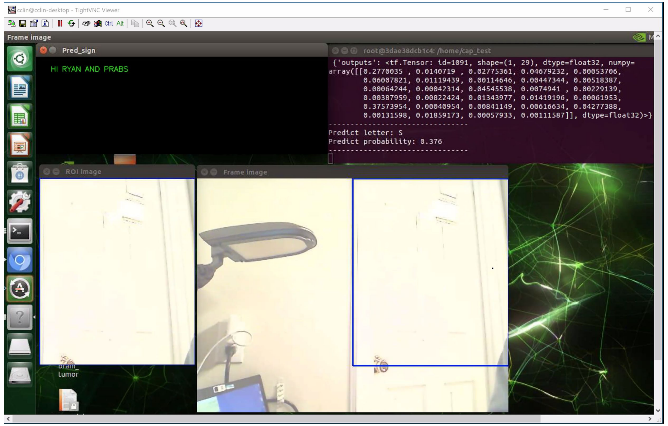
- Copy a set of images into Google Cloud Storage.
- Create a CSV listing the images and their labels.
- Use AutoML Vision to create your dataset, train a custom AutoML Vision Edge model, and make a prediction.
- Export and deploy the AutoML Vision Edge model.

EDGE

Edge-pipelines

Edge (Jetson TX2):

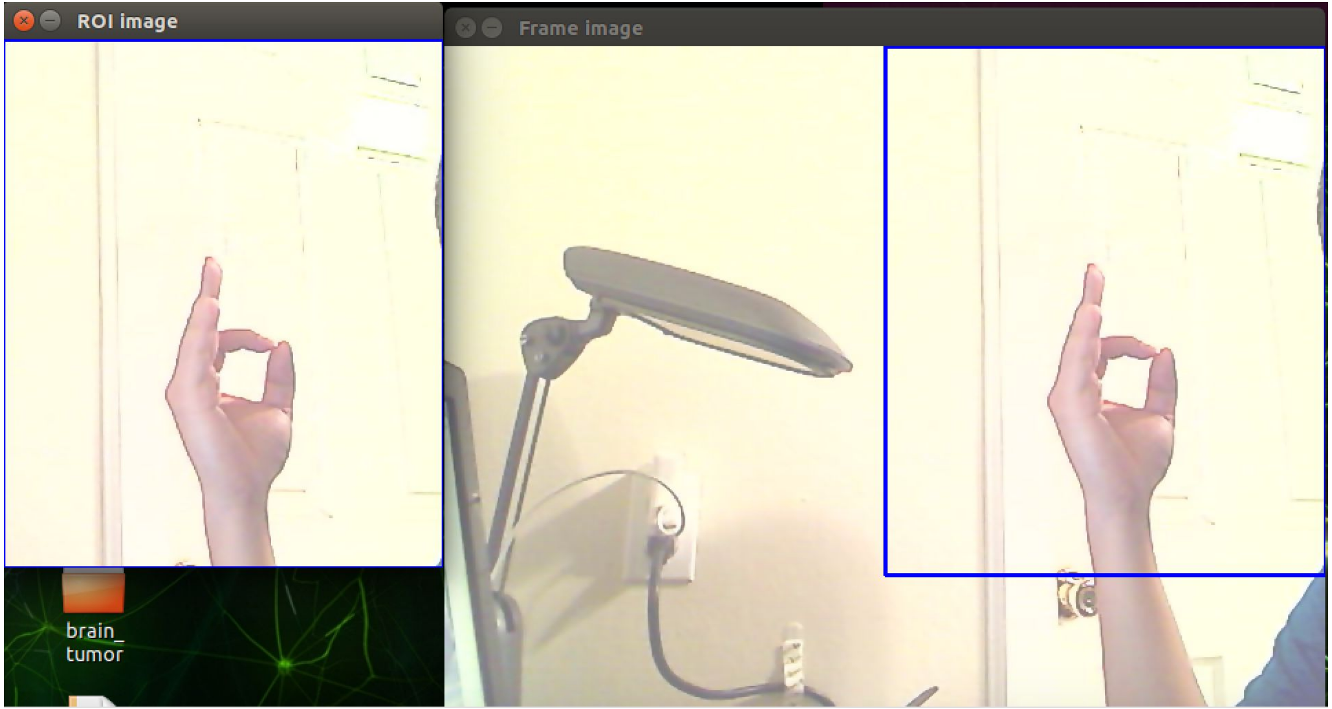
USB camera → Inference on Edge (pretrained model: Keras, OpenCV)



The pipeline has five major steps:

- Read the real-time video input
- Cut out the hands portion from the frame
- Conduct inference on the hands picture
- Return the text
- Return the audio of a sentence

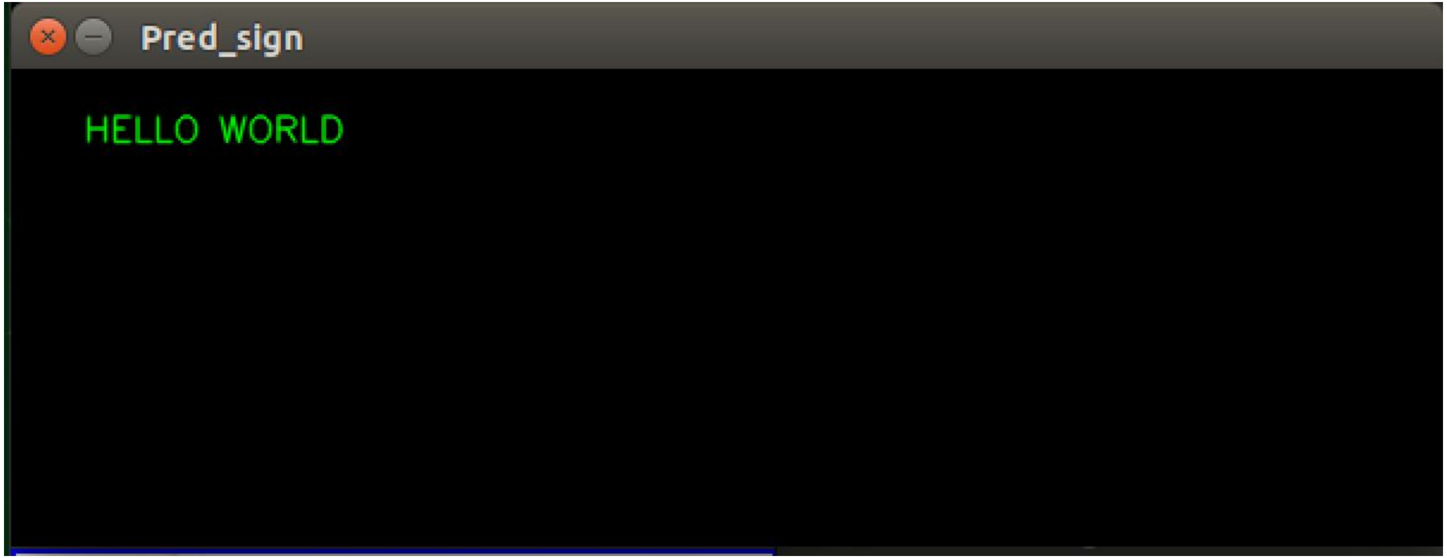
Interface-Windows of image capturing



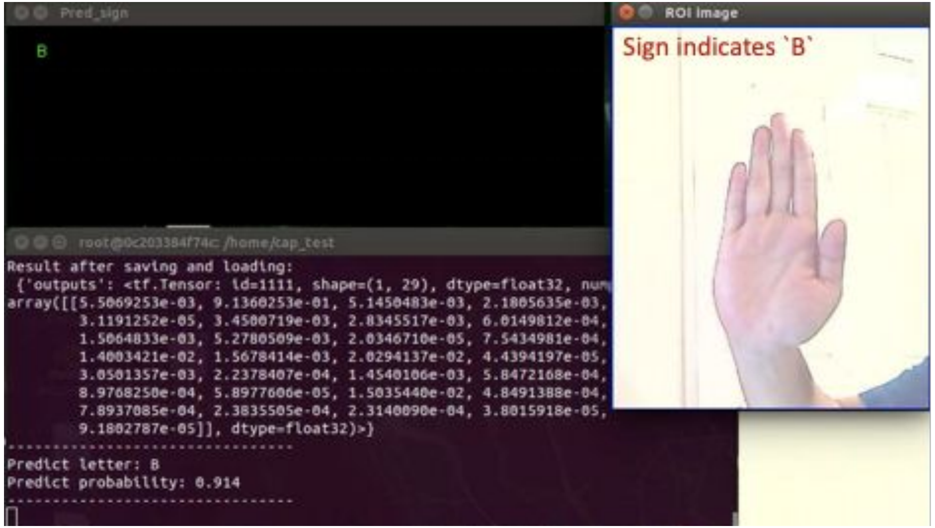
Interface-Prediction results

```
root@0c203384f74c: /home/cap_test
Result after saving and loading:
 {'outputs': <tf.Tensor: id=2861, shape=(1, 29), dtype=float32, numpy=
array([[3.6763038e-05, 8.1548101e-04, 1.1495011e-02, 8.7623578e-01,
        1.9593233e-05, 1.1486478e-05, 6.7616485e-02, 1.3816097e-05,
        8.8108945e-06, 2.7282885e-03, 4.8795091e-06, 9.8892802e-04,
        4.9960795e-03, 2.0433907e-05, 1.0951474e-04, 1.4912941e-05,
        2.7519494e-02, 1.0796617e-03, 1.9500379e-05, 6.8796013e-05,
        3.4988641e-06, 8.9157716e-04, 2.9054610e-04, 9.9610743e-06,
        2.1294598e-03, 1.6401726e-04, 2.6805883e-03, 8.8748948e-06,
        1.7797596e-05]], dtype=float32)>}
--------------------------------
Predict letter: D
Predict probability: 0.876
--------------------------------
```

Interface-Sign language to text and audio

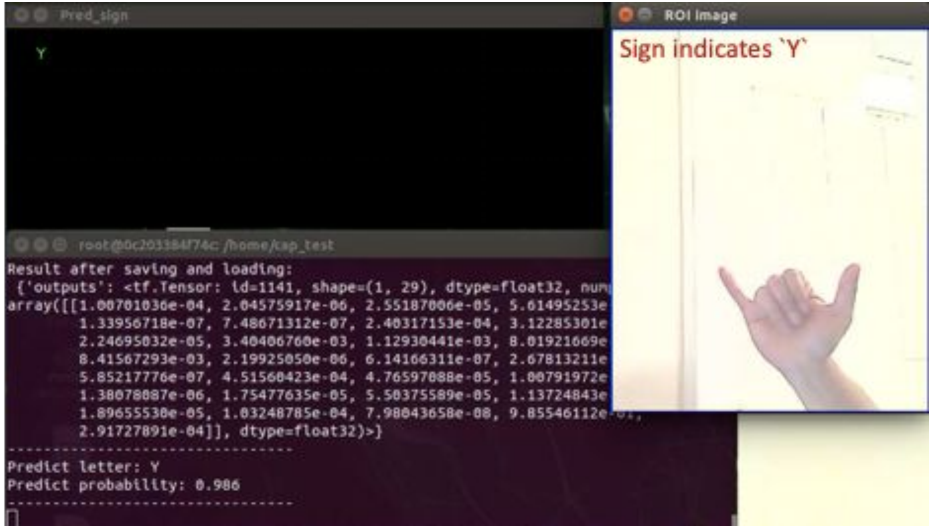


Prediction



Pred_sign
B
ROI image
Sign indicates `B`
root@0c203384f74c: /home/cap_test
Result after saving and loading:
{'outputs': <tf.Tensor: id=1111, shape=(1, 29), dtype=float32, num
array([[5.5069253e-03, 9.1360253e-01, 5.1450483e-03, 2.1805635e-03,
3.1191252e-05, 3.4500719e-03, 2.8345517e-03, 6.0149812e-04,
1.5064833e-03, 5.2780509e-03, 2.0346710e-05, 7.5434981e-04,
1.4003421e-02, 1.5678414e-03, 2.0294137e-02, 4.4394197e-05,
3.0501357e-03, 2.2378407e-04, 1.4540106e-03, 5.8472168e-04,
8.9768250e-04, 5.8977606e-05, 1.5035440e-02, 4.8491388e-04,
7.8937085e-04, 2.3835505e-04, 2.3140090e-04, 3.8015918e-05,
9.1802787e-05]], dtype=float32)>}

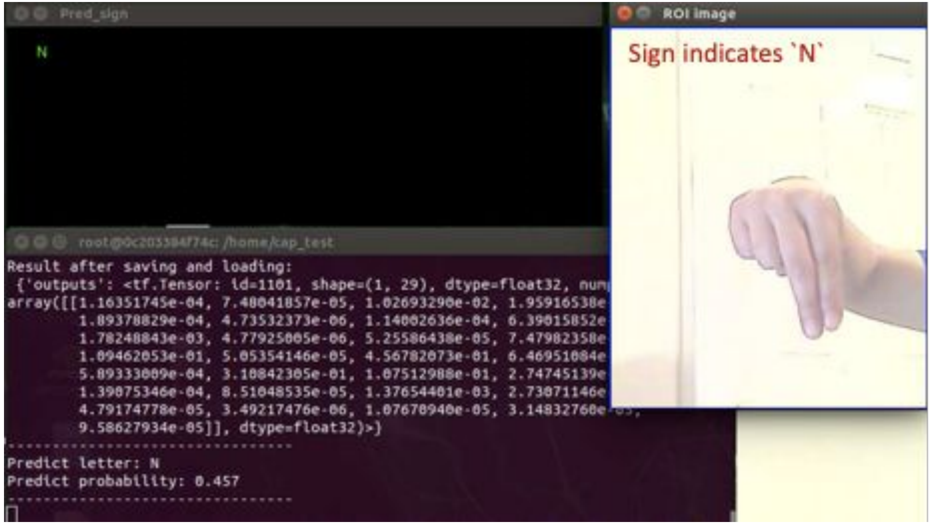
Predict letter: B
Predict probability: 0.914

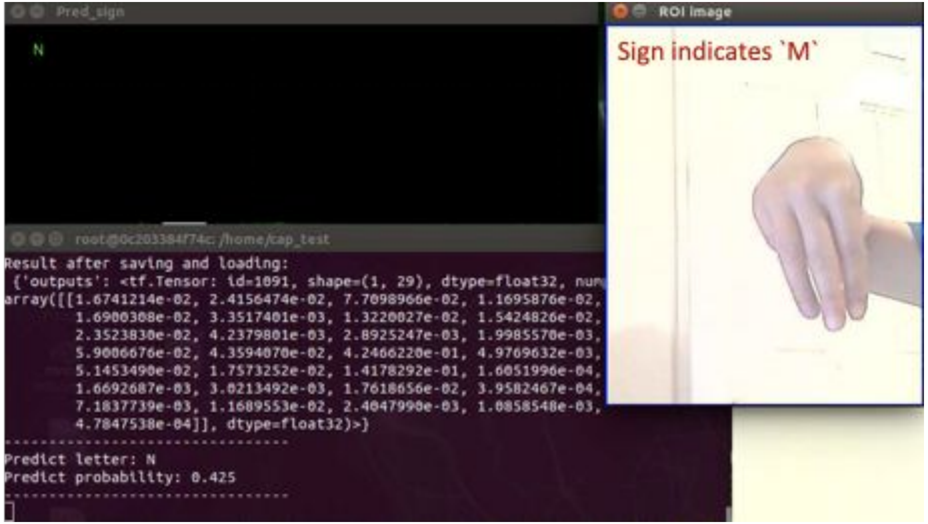


Pred_sign
Y
ROI image
Sign indicates `Y`
root@0c203384f74c: /home/cap_test
Result after saving and loading:
{'outputs': <tf.Tensor: id=1141, shape=(1, 29), dtype=float32, num
array([[1.00701036e-04, 2.04575917e-06, 2.55187006e-05, 5.61495253e
1.33956718e-07, 7.48671312e-07, 2.40317153e-04, 3.12285301e
2.24695032e-05, 3.40406760e-03, 1.12930441e-03, 8.01921669e
8.41567293e-03, 2.19925050e-06, 6.14166311e-07, 2.67813211e
5.85217776e-07, 4.51560423e-04, 4.76597088e-05, 1.00791972e
1.38078087e-06, 1.75477635e-05, 5.50375589e-05, 1.13724843e
1.89655530e-05, 1.03248785e-04, 7.98043658e-08, 9.85546112e
2.91727891e-04]], dtype=float32)>}

Predict letter: Y
Predict probability: 0.986

Issue of misclassification

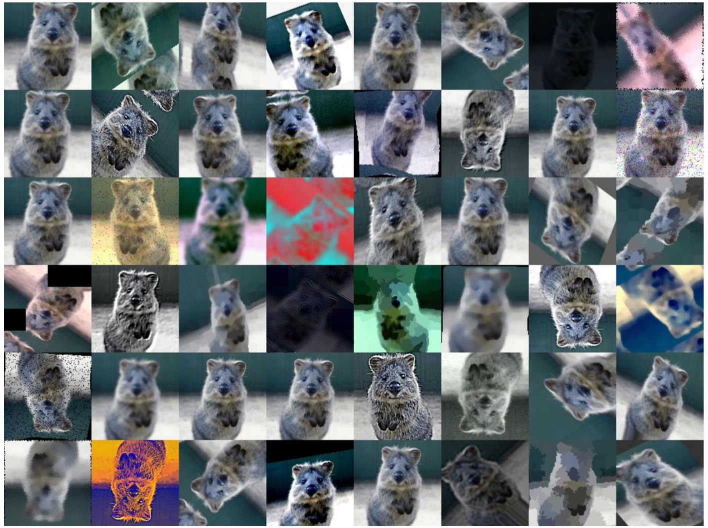




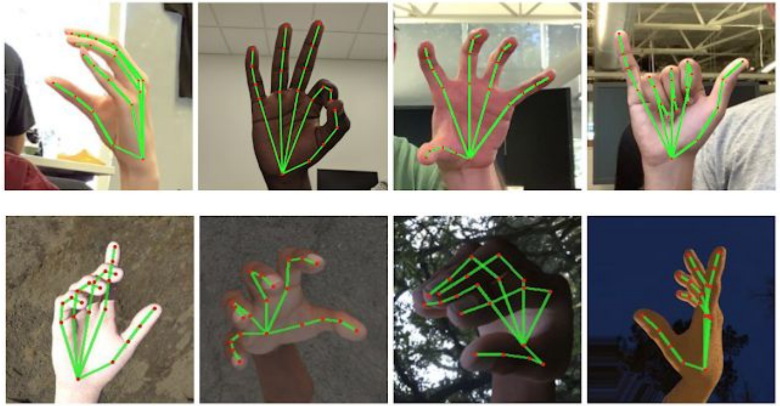
Live Demo

Next Steps–Improvement of prediction

Image preprocessing
Image augmentation
Photo auto-tuning



Palm Detection



Next Steps-Expansion with word-level signs

Hungry

Rice

Next Steps–Mobile Apps

Mobile App

Thank You

Reference

- https://arxiv.org/abs/1910.11006
- https://arxiv.org/abs/1409.1556
- https://www.geeksforgeeks.org/convert-text-speech-python/
- https://github.com/athena15/project_kojak/blob/master/real_time_gesture_detection.py
- https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html
- https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
- https://blog.metaflow.fr/tensorflow-how-to-freeze-a-model-and-serve-it-with-a-python-api-d4f3596b3adc
- https://www.tensorflow.org/guide/saved_model
- https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a
- https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
- https://medium.com/@prasadpal107/saving-freezing-optimizing-for-inference-restoring-of-tensorflow-models-b4146deb21b5
- https://github.com/victordibia/handtracking
- https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
- https://leimao.github.io/blog/Save-Load-Inference-From-TF-Frozen-Graph/
- https://github.com/fchollet/deep-learning-models/releases
- https://github.com/pipidog/keras_to_tensorflow#load-a-pb-file-by-tensorflow