

Real-Time Interaction Using Sign Language

Topic:

Interpretation of Sign Language to Text & Speech and Interaction with a Virtual Assistant

Authors:

- Dawna Grace Raj, Sathia
- Jai Soni
- Nikhil Kohli

Overview:

Communication is the most important part of life. Around 1% of the total population of the world is suffering from hearing impairment, and their life is not as easy as it is for human without limitations.

Deaf people can't hear. Most hearing people don't understand sign language.

Sign language are languages that use visual-manual modality to convey meaning. Sign languages are full-fledged natural languages with their own grammar and lexicon. [ASL\(Argentinian Sign Language\)](#) is a complete, complex language that employs signs made by moving the hands combined with facial expressions and postures of the body.

With this project, we propose a Computer Vision based model that recognizes ASL Letter and few specific words; and convert sign language to text & Speech using Sequence to Sequence models & Deep Neural Networks so that deaf people can communicate with other people in an effective way.

Goal:

Our Goal with this project is to achieve a real time conversation using Sign Language. We have created a deep learning model for recognizing signs from static gestures to interpret signs for individual letters to facilitate real-time recognition and interaction with a Virtual Assistant or another human who does not understand Sign language. We have done this by interpreting the letter into text and further into speech.

We have also implemented Sign language word recognition in video & Real Time to facilitate conversations with Sentence-Level Sign Language Translation.

We will implement this through real time web-cam demo and also Jupyter Notebook real time interaction with a virtual assistant using sign language

1. To recognize and interpret Sign Language letters/words from a Real-Time Image/Video/Feed/Web-cam into text
2. Convert the interpreted text into Speech to interact using the interpreted sign with a Virtual Assistant/Human
3. A speech to text system to transcribe the response for the user/Virtual Assistant
4. Providing the accuracy of the predicted text or speech outcome

5. Implementing the entire model in Keras/Tensorflow

Future Goal:

Our future goal in the coming months is to try and implement continuous recognition of Sentences using Sign Language as it is still a Research Topic and it has many complex issues such as Lightening conditions of the user, Different shapes and movements, complex gestures combined with facial expressions and body movements.

Also, optical flow images is our future implementation which we have currently started from scratch.

Approach:

We used the following approach for the classification of sign language gestures - Video sequences contain both the temporal as well as the spatial features. We will use two different models to train both the temporal as well as the spatial features.

To train the model on the spatial features of the video sequences, we will use a deep CNN (convolutional neural network). CNN will be trained on the frames obtained from the video sequences of training data.

Then we will use RNN (recurrent neural network) to train the model on the temporal features. CNN model will be used to make predictions for individual frames to obtain a sequence of predictions or pool layer outputs for each video. Now this sequence of prediction or pool layer outputs will be given to RNN to train on the temporal features.

Dataset:

For Letter recognition, we will use a custom based Images of Signs and will try using Data Augmentation & GANs to increase the instances of each sign corresponding to the letter.

The data used consists set of Argentinian Sign Language (ASL) Gestures, with around 3200 videos belonging to 64 different types of words.

Argentinian Sign Language Video Dataset

<http://facundoq.github.io/unlp/lsa64/>

Custom Dataset prepared by us: https://github.com/Nikhilkohli1/Dataset_Colored

- We are using LSA64 Argentinian Sign Language Video Dataset for this which consists of 3200 Videos of 64 words
- We have selected 24 such words for this project which has 1200 videos in total to train our models
- For isolated Letter and Word recognition, we have created our own dataset based in the gestures and applied several Image Transformations to it like Morphological, Canny Edge Detection and Grayscale to transform frames into features that can be learned by our CNN model.

Model Structure Used:

- CNN (Inception V3 model pretrained on Imagenet)
- RNN-LSTM
- Time distributed CNN

Data Preparation:

- Create folders of train, test, sequences, checkpoints and logs under data folder
- Create class wise folders for each word in the dataset
- Split the videos into respective class folders under train and test folders
- data_file.csv file to be created in the data folder which contains videos class, train/test and frame count of each videos and filename of each video.

Model detailed Structure:

- In CNN model, we used an inception V3 model pretrained on Imagenet to extract features from each frame
- Inception V3 model is 42 layers deep learning network
- Imagenet is a dataset of over 15 million high resolution images with around 22000 categories
- The extracted features are fed into an LSTM which learns the temporal features of the videos.

Pipeline Design:

- Extract features from frames of videos using CNN Inception V3 model after removing the top layers so we can get the pooled layer output and pass it as a sequence input to RNN as a separate network
- Every frame from each video is run through the inception saving the output from the final pool layer of the network after chopping of the top classification part of the CNN model.
- we convert those extracted features into sequences of extracted features. As we rescaled each video into a 40-frame sequence. So, we stitch the sampled 40 frames together, save that to disk, and train LSTM model without needing to continuously pass our images through the CNN every time we read the same sample or train a new network architecture.
- The LSTM, which we use is single, 2048-wide LSTM layer, followed by a 1024 Dense layer, with some dropout in between
- Each frame of video is converted as sequence and several sequences (40) are combined and put into a .npy(numpy) file

Use Cases

Facilitating the interface to have a conversation with a hearing disabled person without knowledge of Sign Language.

Real time Conversations —

Deaf people can't hear. Most hearing people don't understand sign language. Bridging this gap means a hearing person can interview a deaf person, or someone who is hard of hearing, via Skype or Google Hangout. They can hold a meeting or do job interview, and just communicate in a natural way.

Interaction with Virtual Assistants —

This will aid deaf people to interact with Virtual Assistants like Siri/Google Home/Alexa and be able to let them use the latest in technology in a useful way. They can ask Virtual Assistants to call 911 or to a hospital in an emergency situation.

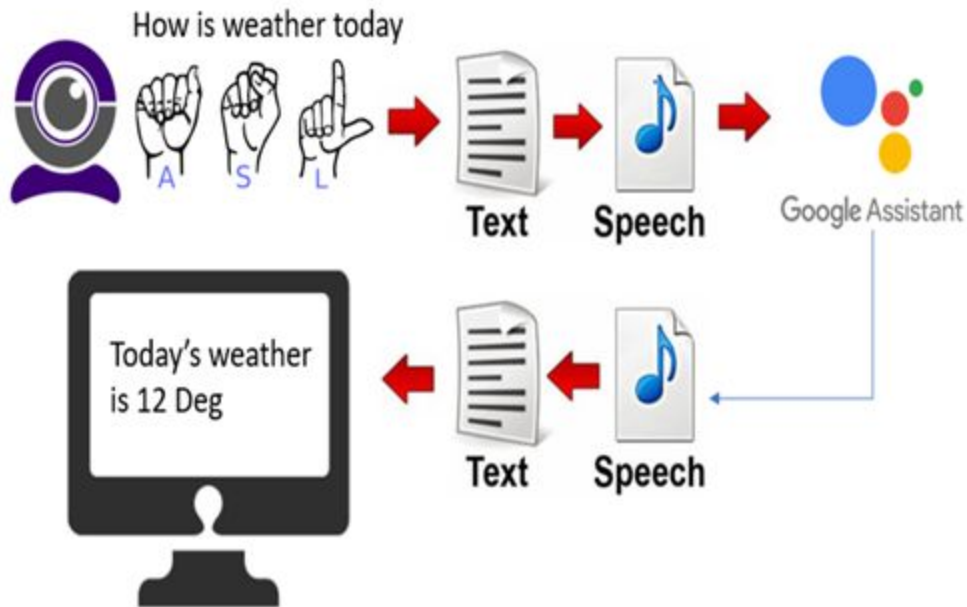
Job Interviews and Consultants —

A hearing person can interview a deaf person, or someone who is hard of hearing, via Skype or Google Hangout. A hearing person can give consultation online to a deaf person.

ASL Learning Application —

Another application could be an ASL learning app, where those using American Sign Language would be able to evaluate their proficiency through the video captioning.

Flow Chart:



Process Outline:

1. Data Pre-processing
 - Data Cleaning, Frame Capturing from Videos
 - Resizing the Frames as necessary
2. Exploratory Data Analysis
3. Study of Sequence to Sequence approaches and select the best model for interpretation of Sign languages
4. Design a Text to Speech model to interact with Virtual Assistant
5. Design a Speech to Text model to transcribe the response from user/VA
6. Design of a pipeline and system to implement this approach
7. Deploy the Model on AWS or Google Cloud Computing Platform
8. Build a web application to demonstrate the usage of the application by interacting with a Webcam or a virtual assistant using only Sign Language.

Analysis of Models:

Problem	Model Type	Transformation Applied	Training Accuracy	Testing Accuracy
LSA64 Video Recognition	CNN(inception v3) LSTM	NA	96	95
LSA64 Video Recognition	Time distributed CNN (LRCN)	NA	45	40
Words 224 Recognition	CNN	Grayscale	99.9	99
Words 224 Recognition	Transfer Learning VGG	Canny Edge Detection	99.8	99.5
Words 224 Recognition	Transfer Learning Mobilenet	Morphological	99.4	50.2
Alphabet 224 Recognition	CNN Simple	Grayscale	99.8	99.5
Alphabet 224 Recognition	Transfer Learning mobilenet	Grayscale	99	74

Milestones:

Timeframe	Delivery
Week 1	1. Developing the Project Proposal

	2. Studying about Sign language and Sequence to Sequence Modelling.
Week 2	1. Data Preprocessing and Exploratory Data Analysis 2. Developing the Model for Gesture Recognition for Letters and build a pipeline 3. Developing the Model for Sign Language words recognition and interaction with Virtual Assistant
Week 3	1. Training the model for accurate results 2. Connect the model with a virtual assistant for demo Deployment of models on cloud and build web application with the entire pipeline

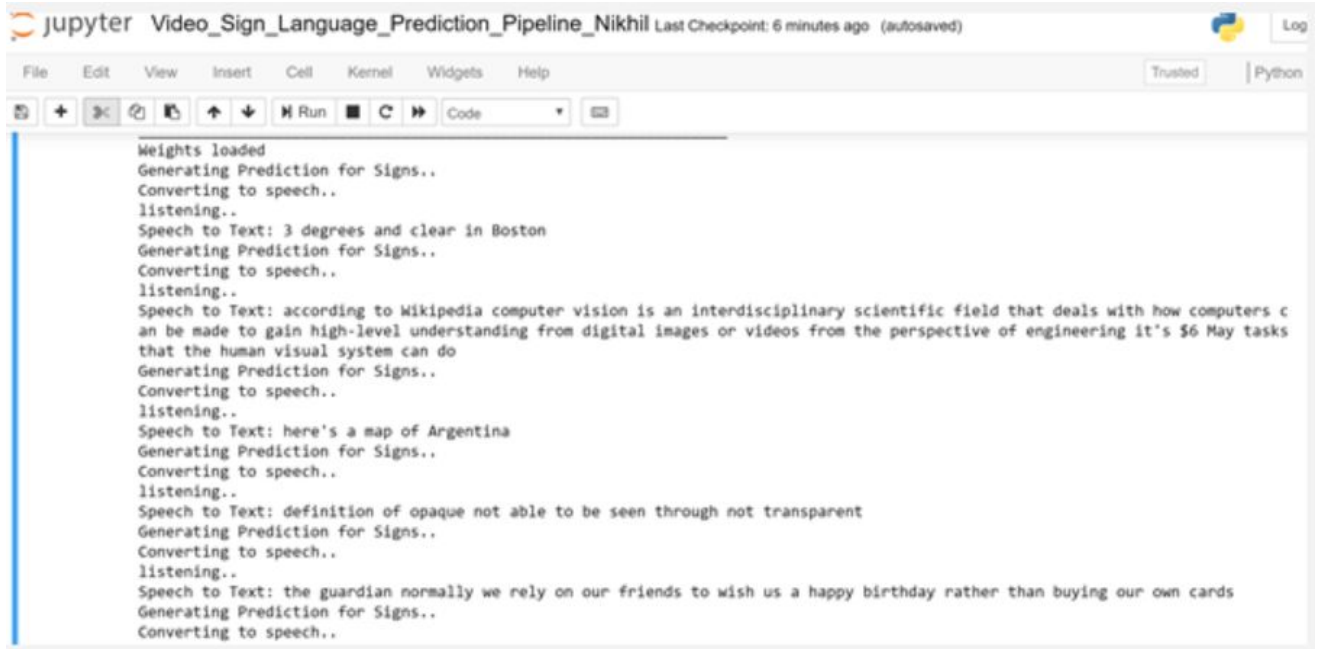
Personas:

1. Real time Conversations –
 - Facilitating the interface to have a conversation with a hearing disabled person without knowledge of Sign Language.
2. Interaction with Virtual Assistants/Consultants –
 - To aid deaf people in interaction with Virtual Assistants like Siri/Google Home/Alexa. They can ask Virtual Assistants to call 911 or to a hospital in an emergency situation.

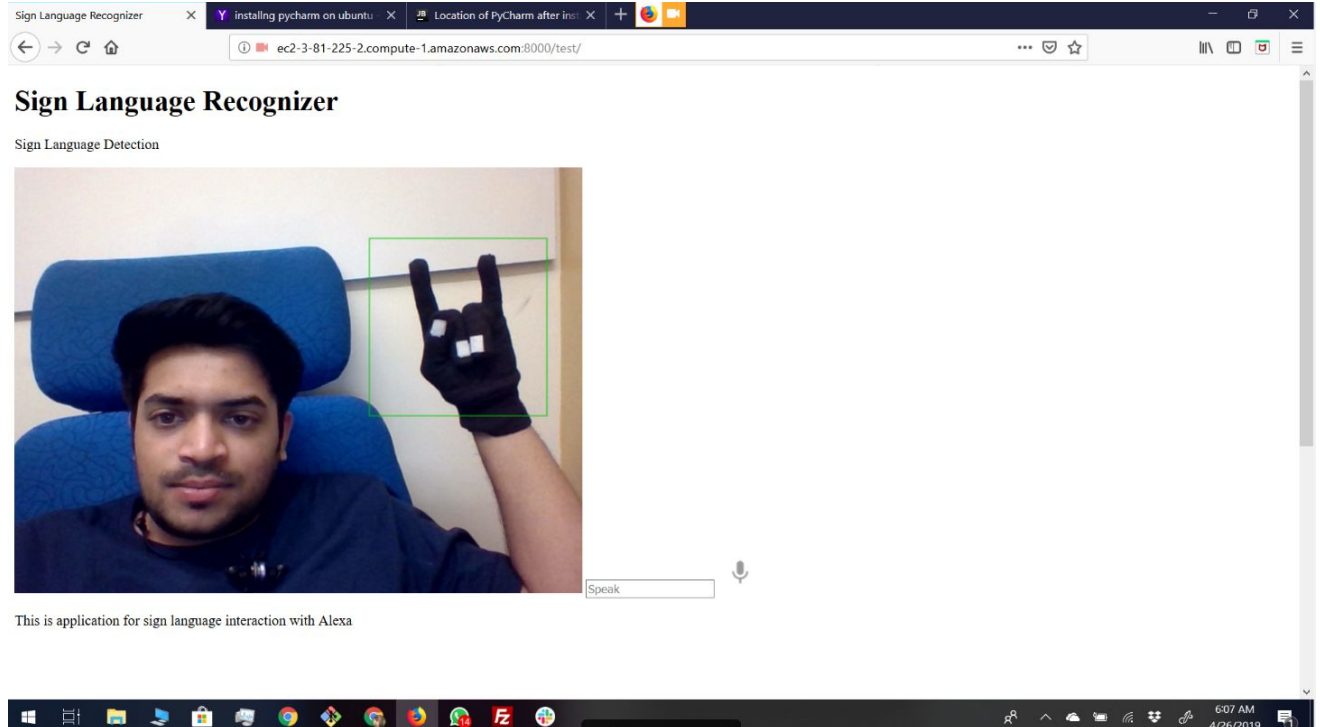
Deployment Details:

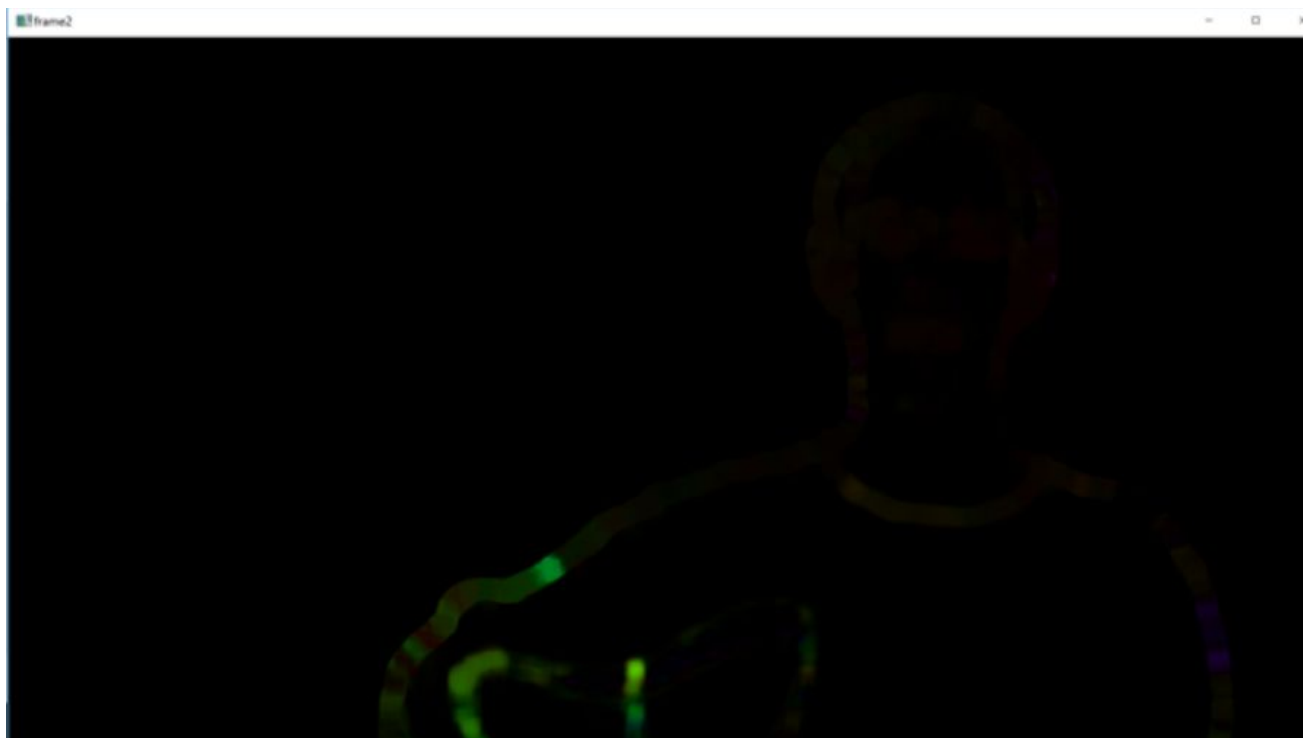
- 1) Language: Python
- 2) Container: Docker/Github
- 3) Cloud Tools/Platforms: AWS (Amazon Web Services) EC2
- 4) Other Considerations: Alexa/ Google Mini Dot, Flask

Screenshots:



A screenshot of a Jupyter Notebook interface. The title bar reads "jupyter Video_Sign_Language_Prediction_Pipeline_Nikhil Last Checkpoint: 6 minutes ago (autosaved)". The notebook contains a series of code cells that execute a pipeline for sign language prediction. The output shows the following steps: "Weights loaded", "Generating Prediction for Signs..", "Converting to speech..", "listening..", "Speech to Text: 3 degrees and clear in Boston", "Generating Prediction for Signs..", "Converting to speech..", "listening..", "Speech to Text: according to Wikipedia computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos from the perspective of engineering it's \$6 May tasks that the human visual system can do", "Generating Prediction for Signs..", "Converting to speech..", "listening..", "Speech to Text: here's a map of Argentina", "Generating Prediction for Signs..", "Converting to speech..", "listening..", "Speech to Text: definition of opaque not able to be seen through not transparent", "Generating Prediction for Signs..", "Converting to speech..", "listening..", "Speech to Text: the guardian normally we rely on our friends to wish us a happy birthday rather than buying our own cards", "Generating Prediction for Signs..", and "Converting to speech..".





Nikhil | jai-sori | Home | Sign_La | Video_ | My pri | Templ | INFO | Photo | Video_ | 00 Sig: X | Dataset | Sign_La | +

https://colab.research.google.com/drive/15ybQQ8-XbvKRhLcVRh-10zL7VKbq5X_Y#scrollTo=ZMQIEQ1RUP50

Sign_Language_Detection_Words224_CannyEdge.ipynb ☆

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

RAM 100% Disk 100%

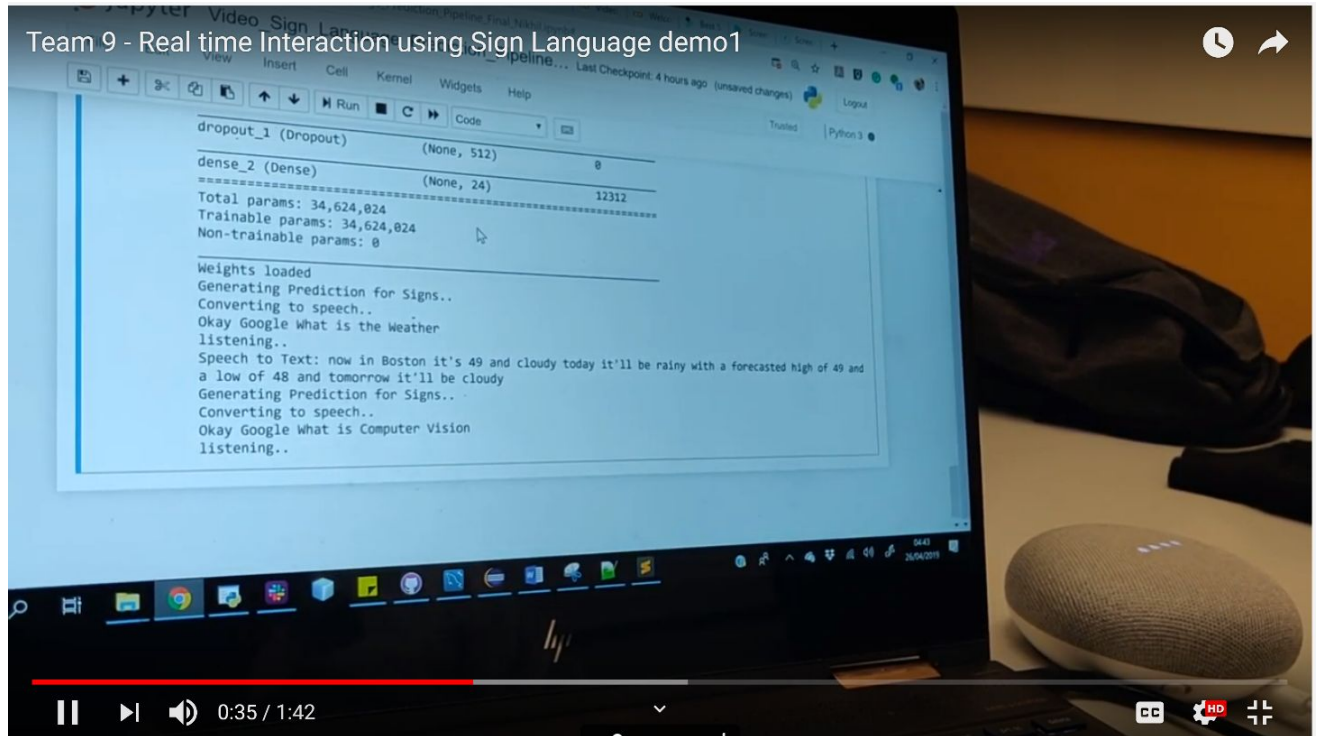
EDITING

```
[14] Total params: 2,789,706
Trainable params: 2,789,706
Non-trainable params: 0
```

```
# Train the data with training set, and check the result with validation accuracy
history = classifier.fit_generator(training_data,
                                steps_per_epoch = math.ceil(training_data.n / training_data.batch_size),
                                epochs = 10,
                                validation_data = validation_data,
                                validation_steps = math.ceil(validation_data.n / validation_data.batch_size))
```

```
... WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python/ops/math_ops.py:3066) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
412/412 [=====] - 144s 349ms/step - loss: 0.2030 - acc: 0.9387 - val_loss: 0.0052 - val_acc: 0.9982
Epoch 2/10
12/412 [.....] - ETA: 1:05 - loss: 0.0151 - acc: 0.9948
```

05:04 26/04/2019



EC2 instance:

<http://ec2-3-81-225-2.compute-1.amazonaws.com:8000/test/>

Youtube Demo:

https://youtu.be/y_XD36URZIY

References:

1. <http://facundoq.github.io/unlp/lsa64/>
2. <https://edu.authorcafe.com/academies/6813/sign-language-recognition>
3. <https://medium.freecodecamp.org/weekend-projects-sign-language-and-static-gesture-recognition-using-scikit-learn-60813d600e79>
4. <https://machinelearnings.co/sign-language-recognition-with-hmms-504b86a2acde>
5. <https://www.digitaltrends.com/cool-tech/sign-language-translation-tech-back/>
6. <https://pythonprogramminglanguage.com/text-to-speech/>
7. <https://blog.coast.ai/five-video-classification-methods-implemented-in-keras-and-tensorflow-99cad29cc0b5>
8. <https://github.com/harvitronix/five-video-classification-methods>