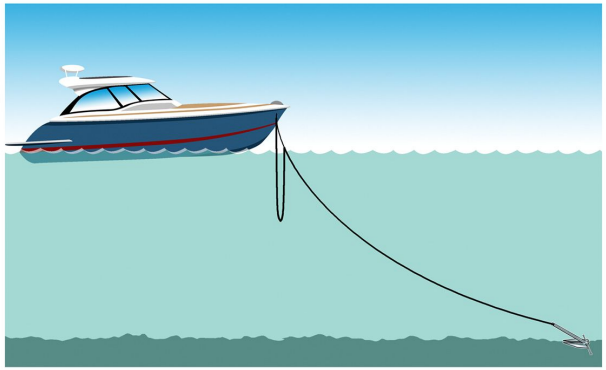
Forget the Remote: Increasing Human-Computer Bandwidth with Neural Networks and Under-Utilized Input Devices

By Sebastian Rosado Mustafa

Problem - The Hardware ‘Anchor’

- Old Tangible User Interfaces (TUIs)
  - Infrared TV Remote & Keyboard PCs → 1980s
- User ‘tied’ to input device
  - TV Remote
  - Laptop Keyboard
- Hardware can be lost
  - ~ 2 weeks (371 hrs) of viewer’s lifetime lost looking for the remote
- Too much deadweight loss!



Solution - Computers that Can See

- Control music, movies and shows from some distance with simple hand signs.
  - Neural networks and webcam live-streaming
- Become the remote
  - Now the device looks for you
  - Minimum required proximity to input device increased
  - Roku infrared transmission irrelevant
  - Marginal increases in model latency offset by elimination of last-meter travel time to remote or keyboard.
- Save time, look cool

Process Summary

Wrangle

- Load & transform images into NumPy arrays

- Convert to grayscale, binarize and subtract the background with OpenCV

Train

- Custom model and VGG16 implementation with 4 additional dense layers

- Trained in the cloud with GCP

Evaluate

- Cross-tested model on Kaggle Dataset

- **F-1 Score: 98%**
- **Precision: 98%**
- **Recall: 98%**

Deploy

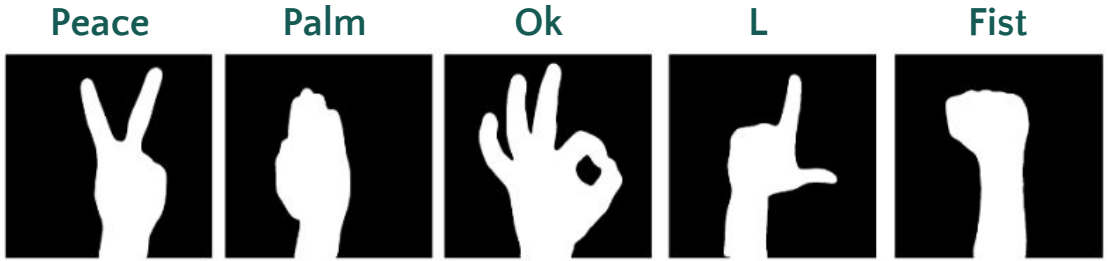
- Python script with custom gesture bindings for PC media

- Python script with custom gesture bindings for Roku TV

Wrangling

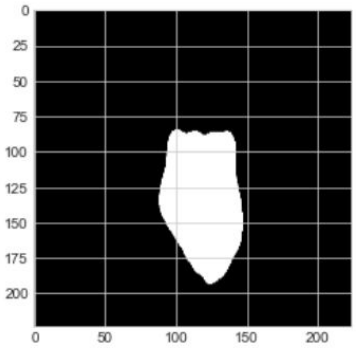
Introducing the Data

- Hand Gesture Recognition Database - Kaggle
  - Motion sensor
  - 10 gestures, 10 subjects
  - 20,000 images
- Brenner Heintz - Databricks
  - Webcam (Preprocessed)
  - 5 gestures, one subject



Wrangling

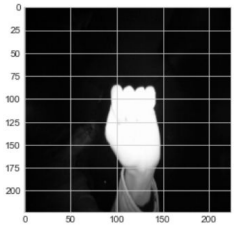
Data Processing Pipeline

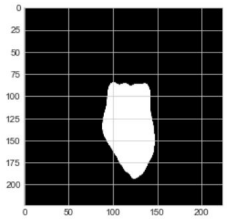


1. **Flip Image**
   Perspective matching
2. **Convert to Grayscale**
   Enables binary thresholding
3. **Apply Gaussian Blur**
   Smooths edges and removes noise
4. **Apply Binary Thresholding**
   Clear hand outlines, skin-color agnostic
5. **Resize to (224 x 224)**
   Fitting VGG16 input criteria
6. **Convert to 3-Channel Array**
   Fitting VGG16 input criteria

Training & Evaluation

Modeling Flow





- **F-1 Score: 98%**
- **Precision: 98%**
- **Recall: 98%**

1. Original Kaggle Dataset

Custom VGG-16 (1)

Perfect confusion matrix & classification report on its test set. Poorest potential for generalization.

2. Processed Kaggle Dataset

Custom VGG-16 (1), Fully Custom Model (1)

Perfect confusion matrix & classification report on its test set. Poor potential for generalization.

3. Heintz Dataset

Custom VGG-16 (2), Fully Custom Model (2)

Good confusion matrix & classification report on its test set. Highest potential for generalization.

4. Heintz Models on Kaggle Data

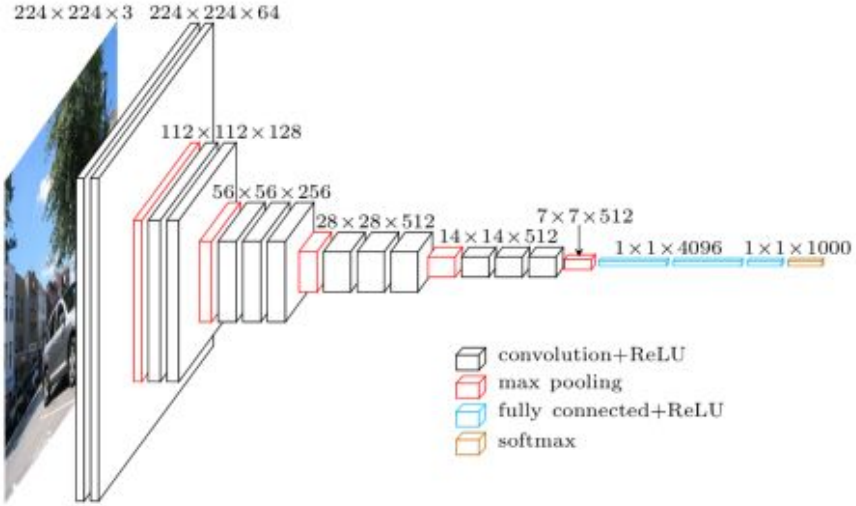
All Heintz Dataset Models

Varying confusion matrix & classification report results on Kaggle test set. From poor to excellent generalization.

Training & Evaluation

Best Model

- VGG16 + Custom Dense Layers
  - Imagenet weights
  - 4 additional dense layers
  - Frozen base layers
  - 0.5 Dropout rate
  - Softmax activation
  - Adam optimizer
  - 20 epochs
  - Batches of 64



Deployment

Demonstration

Points for Improvement

- Background subtraction refresh
- Reconsider background subtraction altogether
- Add gestures and bindings
- Expand output device offerings
- Conditionals to avoid double-execution of gesture-key bindings
- Reduce barriers to adoption
    - E.g. Plotly Dash → Docker → GCP / AWS

Thank You

 sebastianrosado

 in/srosadomustafa