

What was the problem?

- Visual speech data
 - Close perspective
 - Using a webcam
- Use deep learning
 - TensorFlow
- Classify the current mouth position (i.e. viseme)
- The predictions should be suitable for recreating the mouth positions using a mesh
- i.e. How can deep learning technology be leveraged to detect the mouth positions?

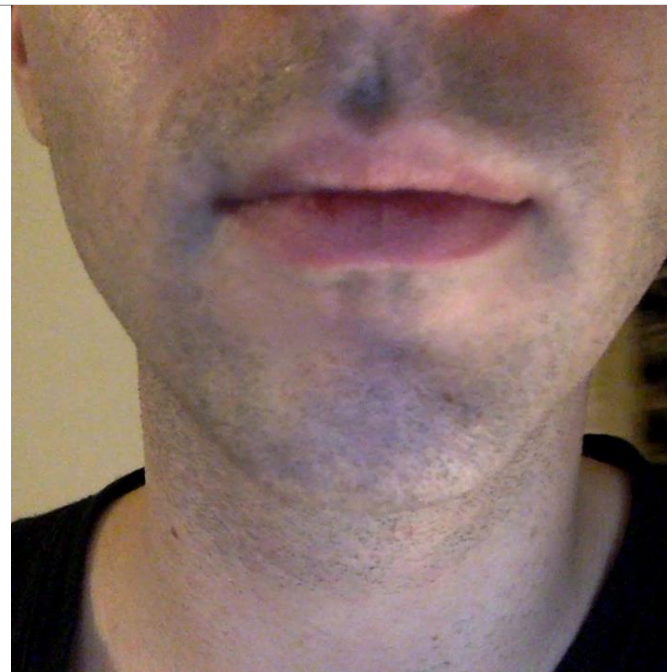
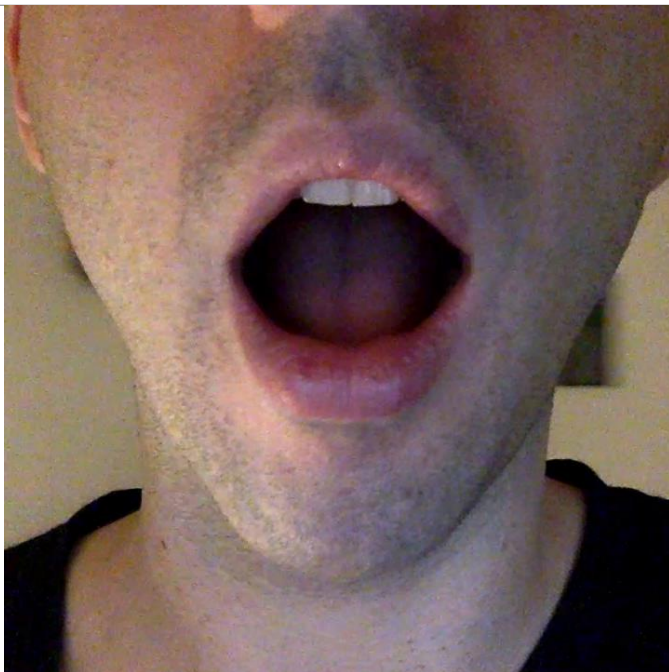
What is my solution?

- Created a dataset of labeled images for each class
- Trained a neural network using the images and exported the frozen model
- Created a Go program to feed each frame of the webcam stream to the model and output the prediction

Image Data

- Roughly 4000 images
- Different individuals
- Some individual screenshots, some extracted from video clips using ffmpeg
 - ffmpeg -i [location of source video] -filter:v "crop=720:720:180:0" [location of saved images with naming template]
 - naming template example: "/somelocation/ope.%05d.png"
 - save each frame to "somelocation" using the prefix and 5 decimal padding with automatic incrementation
- Each image filename contains the label
 - e.g. ope.12345.png
 - e.g. clo.12345.png

Open and Closed Classes



NN Details

- Input image size: 64x64
- Output activation function: Softmax
 - squishes output between 0 and 1
- Optimizer: Adam
- Loss function: Categorical Crossentropy
 - increases as prediction further from actual value
- Training epochs: 5
- Learning rate: 0.0005
- Batch size: 5
- Data augmentation
- Input -> Conv2D -> Pooling -> Conv2D -> Conv2D -> Pooling -> Fully Connected -> Fully Connected

```
Desktop — root@bc35656bcfb6: /PR1 — docker run -p 8080:8080 -p 6006:6006 -it peaceful_bassi_commit bash — 130x2
...-p 6006:6006 -it peaceful_bassi_commit bash      ~/Desktop — -bash      ~/Desktop — -bash

Run id: model_open_close_6
Log directory: tmp/tflearn_logs/
-----
Preprocessing... Calculating mean over all dataset (this may take long)...
Mean: 93.02273503929564 (To avoid repetitive computation, add it to argument 'mean' of 'add_featurewise_zero_center')
-----
Preprocessing... Calculating std over all dataset (this may take long)...
STD: 58.04046044030146 (To avoid repetitive computation, add it to argument 'std' of 'add_featurewise_stdnorm')
-----
Training samples: 4077
Validation samples: 453
--
Training Step: 816 | total loss: 0.07645 | time: 390.902s
| Adam | epoch: 001 | loss: 0.07645 - Accuracy: 0.9916 | val_loss: 0.00253 - val_acc: 1.0000 -- iter: 4077/4077
--
Training Step: 1632 | total loss: 0.07047 | time: 412.822s
| Adam | epoch: 002 | loss: 0.07047 - Accuracy: 0.9875 | val_loss: 0.00144 - val_acc: 1.0000 -- iter: 4077/4077
--
Training Step: 2448 | total loss: 0.06843 | time: 374.443s
| Adam | epoch: 003 | loss: 0.06843 - Accuracy: 0.9972 | val_loss: 0.00040 - val_acc: 1.0000 -- iter: 4077/4077
--
Training Step: 3264 | total loss: 0.07162 | time: 382.700s
| Adam | epoch: 004 | loss: 0.07162 - Accuracy: 0.9969 | val_loss: 0.00061 - val_acc: 1.0000 -- iter: 4077/4077
--
Training Step: 4080 | total loss: 0.05504 | time: 927.716s
| Adam | epoch: 005 | loss: 0.05504 - Accuracy: 0.9977 | val_loss: 0.12544 - val_acc: 0.9823 -- iter: 4077/4077
--
root@bc35656bcfb6: /PR1#
```

How good is the result?

- 2 classes/visemes covered, Open and Closed
- Sensitive to lighting and background
- Sensitive to positioning
- Functioning but not very robustly

What could be improved

- Make it more robust against lighting and positioning
- Add more classes (e.g. 4 visemes in total)
- Would need smoothing when used in another context (to prevent repeated jumping between classes on false predictions)
- Link it to a mesh model in Unity

What I Used

- Components involved:

- nn.py creates and trains the TF model
- main.go processes each

- Technologies involved:

- TensorFlow, Tflearn
- Docker for TensorFlow environment deployment
- GOCV for openCV bindings in Golang
- Freeze script from <https://github.com/tflearn/tflearn/issues/964>

A Few Problems I faced

- Exporting the tflearn model as .pb
- Reading the model from gocv
 - openCV does not support reading models containing Dropout layers
- Needed to commit and restart container in order to use Tensorboard
 - Not sure why the only way to bind a port to a container is at runtime...