Building an Edge Facial Recognition Doorbell

W251 Wednesday 4pm

Erik Hou, Joanna Yu, Yuze Chen, Adam Johns

Introduction

Facial recognition via deep learning is a highly secure and validated way to recognize individuals with applications in personal security. One consumer application of facial recognition, which is growing in popularity, is video doorbells using facial recognition to identify visitors and alert residents of a home when they have a guest, and in some cases, the identity of the guest as well.

However, while facial recognition may have benefits in consumer applications like this one, the growing use of facial recognition has also resulted in significant privacy and security concerns. Facial recognition data can serve as a highly sensitive unique identifier, so transferring this data to companies creates the risk to the user that it may be misused by the company or sold to third parties, or that it may be stolen by third parties in a security breach. Additionally, as facial recognition data is considered biometric identification under data privacy regulations like the EU GDPR and Illinois Biometric Information Privacy Act, keeping this data on a server may result in liability for companies or result in the need for obtaining consent from users which may discourage them from using the technology. For all these reasons, a smart doorbell that can be trained and perform inference on an edge device without transferring facial recognition data, user information or images to the cloud. By securing all user data and training on the edge, such a device can result in a better solution for both the users and producers of these tools.

Currently available systems such as Amazon’s Ring and Google’s Nest smart doorbells include inexpensive chipsets with limited edge capabilities, and utilize cloud capabilities for their user identification approaches. This drawback has likely been recognized by Apple, who has been developing products to allow users to create a more private network. Their recently announced HomePod system specifically touts the privacy of its cameras, meaning it is likely to use an edge-based inference approach and is touting this as a feature.

Given the drawbacks of these current systems in terms of security and data privacy, and attempting to beat Apple, we decided to build a facial recognition system using the Jetson TX2 to perform training and inference, which would not require communication of images, identifying vectors, or identifying information on the cloud.

Planned Features

When designing the initial plan for our doorbell, we identified the following features as ones that would be important to include:

Table 1: Pipeline Summary

	Feature	**Description**
Cloud	Facial Encoder Model	Cloud-trained model capable of recognizing and differentiating faces with a high degree of accuracy. This model could be trained in the cloud and deployed to the edge device via transfer learning
Edge Training	Image Extraction	Extract images from video feed of known users to train the edge classifier
	Detect, Align, and Save Faces	Identify the faces from the video feed, align and save images to prepare for processing
	Train Classifier	Train the classifier to recognize known faces (identified users that could be friends or family members, which could be used to alert the owner, or the residents of the home, who could be used to unlock the door) and differentiate them from unknown people on the video feed
Edge Inference	Security Measures	Perform steps to verify that the image is a live person and not a photo or video recording. The strength of the security features we are able to implement will determine whether the device could be used as a lock if it is sufficiently spoof-proof, or as a doorbell, where the risk is lower.
	Inference	For a given user on the video feed, run the image through the classifier trained on known users and determine if they are a known user or an unknown user

In order to design an effective pipeline, we needed to ensure a number of criteria:

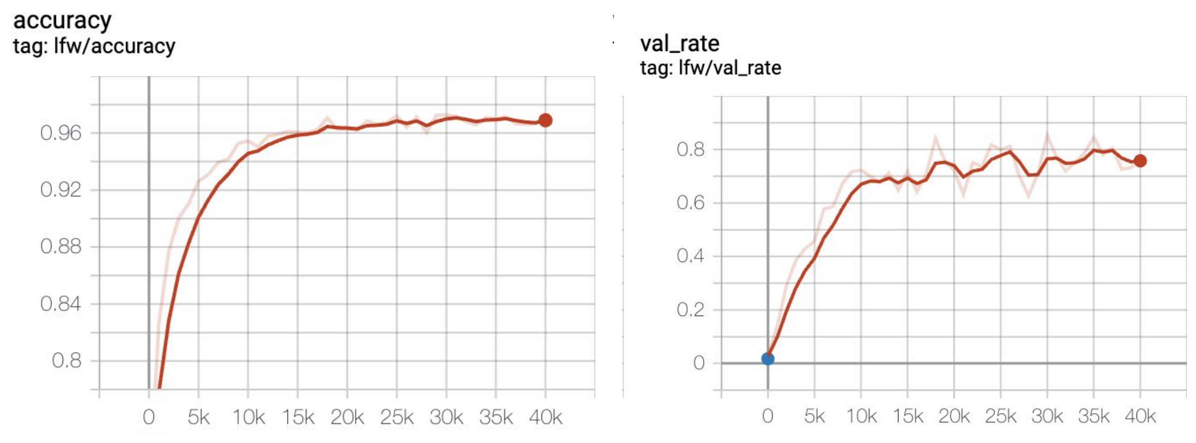
- Have a facial recognition classifier which is sufficiently sensitive to identify faces from the video feed, while being computationally efficient enough to run on the Jetson
- Have security features that make the device impossible to fool with images or live video

- Minimize misclassification of unknown users as known users (false positives) - these users will cause the biggest nuisance to the residents of the home, and could lead to security risk if the device is used as a lock. False negatives (known users being classified as unknown) are less of an issue as the video feed will include multiple frames, so a person in front of the doorbell could move their face around a bit until it is recognized (similar to trying to unlock an iphone).
- We also wanted to explore training a facial recognition classifier from scratch in the cloud, so a key criteria for our classifier will be something that could be trained in a reasonable amount of time with our available cloud resources

Implementation - Cloud Training of Facial Recognition Classifier

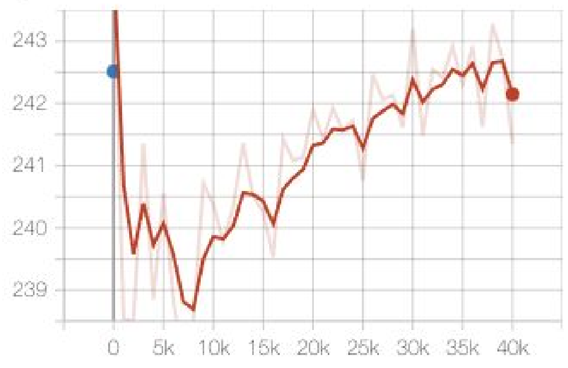
Based on the criteria we set out for our model, we identified an implementation of Inception-Resnet v1 using TensorFlow version r1.7 (https://github.com/davidsandberg/facenet/wiki/Classifier-training-of-inception-resnet-v1). We followed the approach specified in the repo to train the classifier from scratch using the CASIA-WebFace dataset, This training set consists of a total of 453,453 images over 10,575 identities after face detection. For validation, we used the LFW dataset (http://vis-www.cs.umass.edu/lfw/) running through the same image aligner. While we had some challenges with the environment, eventually we were able to train the model on an NVIDIA p100 in approximately three days to ~96% accuracy. While this step was not strictly necessary and we could have replaced it in our pipeline with a pre-trained model, it was an interesting exercise and was surprising that the model could be trained on a commonly available cloud GPU in a reasonable amount of time. Our Tensorboard results from the model training are shown in figure 1:

Figure 1: Tensorboard Results for Cloud Training

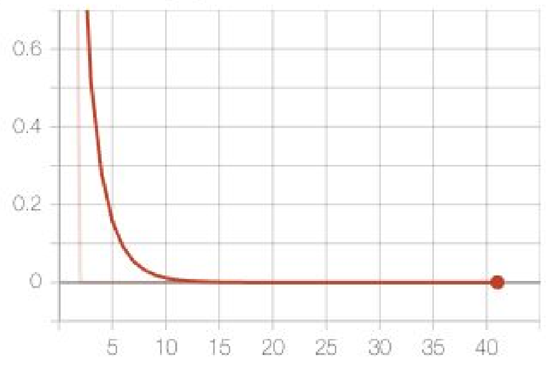


time

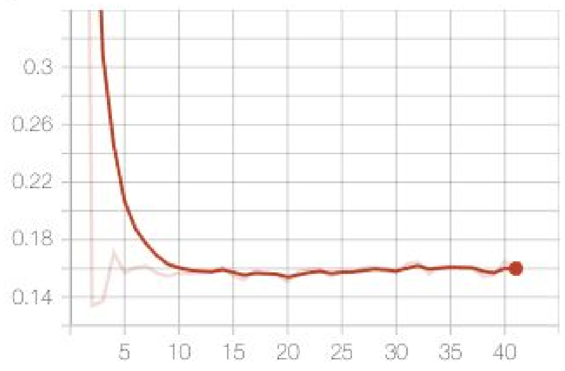
lfw
tag: time/lfw



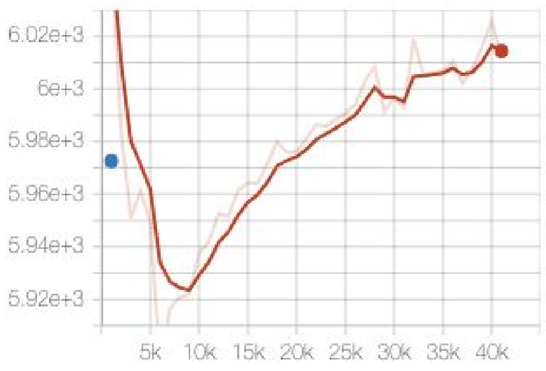
save_metagraph
tag: time/save_metagraph



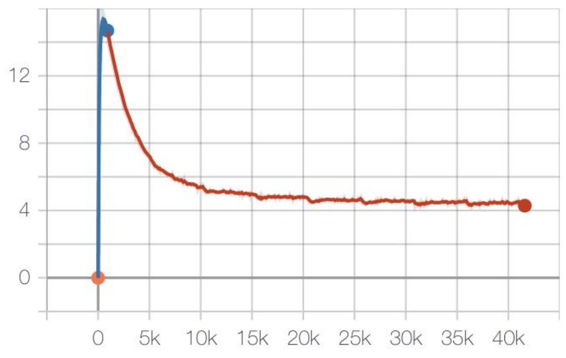
save_variables
tag: time/save_variables



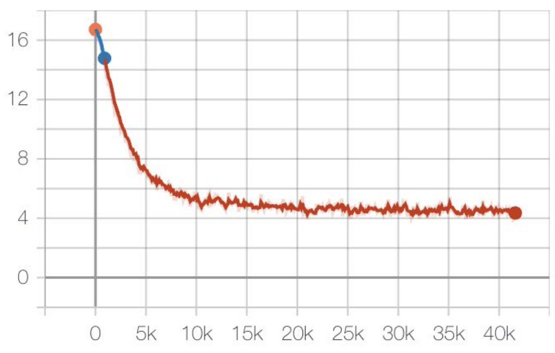
total
tag: time/total



total_loss_1



total_loss__raw_



Local Training on the Jetson

Image Preprocessing

The pipeline on the TX2 starts with image preprocessing, which is also used to create our datasets. The datasets consist of known individuals (Erik and his girlfriend Diana) and unknown individuals pulled from other datasets. Our approach was inspired by https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/ and https://github.com/krasserm/face-recognition/blob/master/face-recognition.ipynb.

The steps for image preprocessing are outlined as follows:

1. **Image Extraction**: Video recordings for known users are extracted frame by frame and the frames are saved as images for subsequent processing; unknown user images were sourced online using a bulk download tool.
2. **Face Alignment**: The images extracted goes through a face detection model to extract the faces with a predefined marginal space.
3. **Face Encoding**: The headshots are fed into the FaceNet model to create embeddings for the face.

About the Data

The training, validation, and testing datasets were generated by videotaping Erik and Diana under various lighting conditions and face orientation and then preprocessing the videos. Since the model will need to identify known faces as well as recognize unknown faces, our training set includes an unknown class. Faces from different races and ethnicity were selected for the unknown class. Erik and Diana are both Asian. Unknown images for the training and validation set were procured using the bulk-bing-image-downloader tool demoed in Lab 8 (https://github.com/ostrolucky/Bulk-Bing-Image-downloader). One notable point is that while the known user images were captured from video stream, unknown user images were downloaded from the internet directly. We purposely used unknown faces of different identities in the validation set to simulate the situation in which the model will perform inference after deployment since most of the unknown faces would be faces that the model has never seen before.

Table 2. Number and Class of Images Used for Training, Validation, and Testing

	Erik	**Diana**	**Unknown**	**Total**
Training Set	139	97	479	715
Validation Set	23	98	158	279

About FaceNet

The FaceNet model measures face similarity by mapping face images into a compact Euclidean space and calculating the distance between faces. As described in the original paper, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” this model is a deep convolutional network trained to directly optimize the embedding itself, whereas other models focus on an intermediate bottleneck layer. FaceNet encodes images into 128 dimensional embedded vectors for Euclidean distance calculation. The network achieved state-of-the-art face recognition performance in 2015 and remains a very popular model in facial recognition today.

Training Approach #1 - SVM

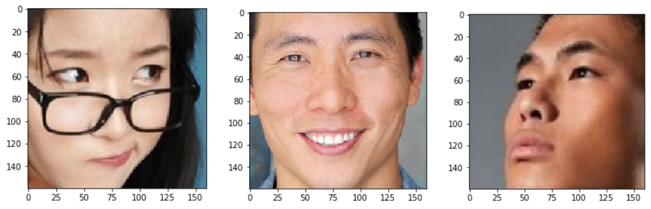
The initial attempt uses an SVM with a nonlinear kernel because the decision boundary is very complex given the unknown class.

Table 3. Confusion Matrix for SVM Model with Nonlinear Kernel

Label \ Pred	**Diana**	**Erik**	**Unknown**
Diana	98	0	0
Erik	0	23	0
Unknown	1	2	155

As shown in the confusion matrix, one unknown image from the validation set was misclassified as Diana and two unknown images were misclassified as Erik. The images were examined to identify possible causes (figure 2).

Figure 2: Misclassified Images in SVM Model



As seen above, the model becomes confused when seeing images of the same gender and ethnicity. The misclassification actually presents a significant issue with false positives since the dataset is built with diversity in mind and only contains 20 images for Asian male and female each. This means the model confuses 1 of them with Diana (5%) and 2 of them with Erik (10%).

If we pool all the known faces together and make the classification binary, from a security point of view we can have a higher tolerance for false negatives (*known* predicted as *unknown*) than false positives (*unknown* being classified as *known*).

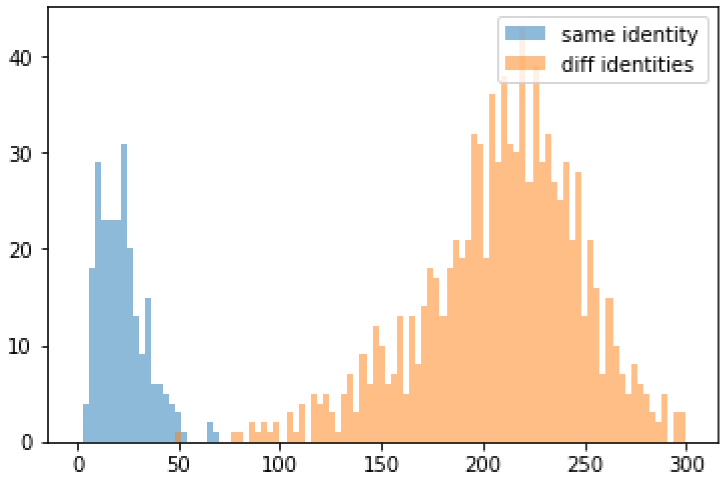
We then attempt another separate mechanism in our prediction process in order to prevent having false positives.

Training Approach #2 - SVM with τ

From the SVM model, we see that the model is good at distinguishing known faces but struggle a bit at classifying unknown faces as *unknown*. To address this issue, we try to establish a threshold of likeness, τ, for classifying a face as one of the known ones. Since FaceNet is trained to arrange similar faces closer together in the high-dimensional space, we can simply adjust this threshold using the Euclidean distance as a metric. The strategy is to get the mean of all the embeddings of a known identity in the training set and any image needs to be closer to a mean than τ to be classified as the corresponding identity; otherwise, it will be labeled as *unknown*.

By plotting the identity distances in the following figure, we were able to identify a candidate threshold value of τ at around 55 units of distance:

Figure 3: Histogram of Identity Distances from the Mean in the Training Set



By inspecting the known identities that had a distance from the mean greater than 50, it was clear that the images were poor quality caused by high levels of light or shadow. Thus, it’s likely that by setting a tau threshold for the classifier, it would be possible to avoid false positives while still having some degree of false negatives (a smaller security concern for the platform). Since we've proven the concept, we can try it on the validation data with τ = the minimum distance between the unknown identities and any mean of known identities. Our value of τ was calculated as 51.199623.

Next, we outline the approach for training the model using the τ threshold. For this step, since we are using τ to determine whether a face is known or not, we no longer need the unknown example for training the SVM. Instead, we can simply train an SVM to find a most "similar" identity for a given face and then use τ to ultimately decide whether we will use the label predicted by the SVM classifier or to say it's an unknown face. Our confusion matrix for this approach is displayed in table 4.

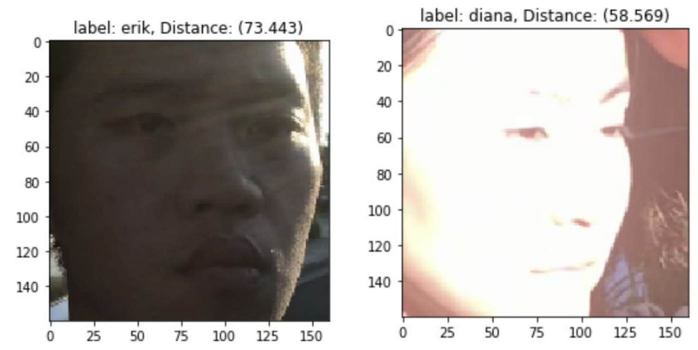
Table 4. Confusion Matrix for SVM τ model

Label \ Pred	Diana	Erik	Unknown
Diana	94	0	4
Erik	0	20	3
Unknown	0	0	158

Incorporating the τ distance threshold was able to successfully push the three unknown faces that were previously incorrectly identified as Erik or Diana back to the correct label, unknown.

Further exploring the impact of this step, it became apparent that there was some sacrifice in misclassifying some known images as unknown (false negatives). While this is less of a security risk, it does impact the functionality of the tool, so we want to minimize these if possible. Examining these misclassified images, it was apparent that they were once again ones with high degrees of light or shadow (figure 4):

Figure 4: Misclassified Image Examples in Validation Set from SVM τ Classifier


label: erik, Distance: (73.443)
label: diana, Distance: (58.569)

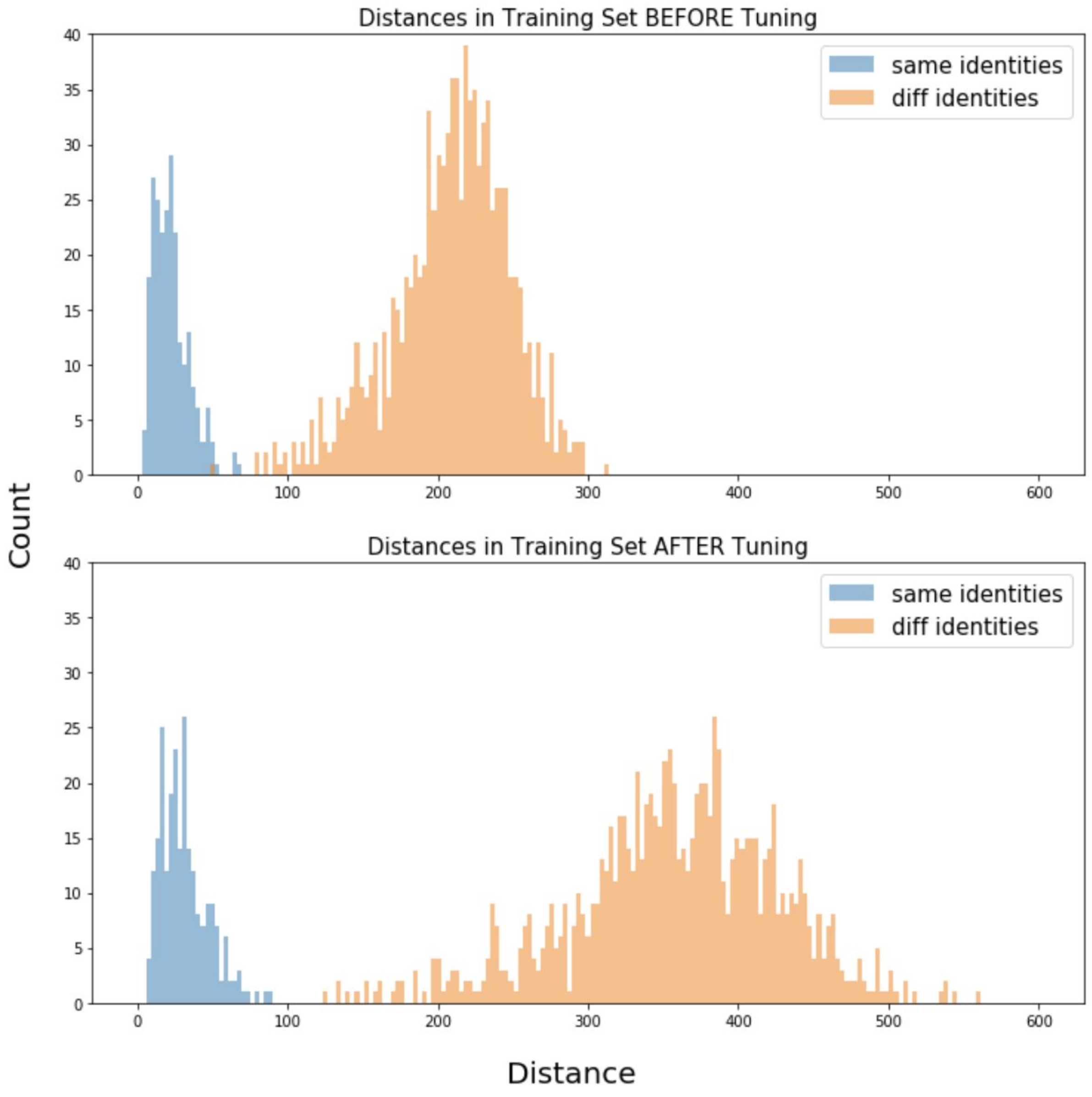
Finally, we double-checked the three images previously misclassified as Erik or Diana to check how far they are from τ . The distance from embedding means for these images ranged from 99.482 to 201.592, substantially greater than the τ threshold we calculated.

Training Approach #3 - SVM with τ and Fine-Tuned FaceNet

Our final approach to improve the functionality of our classifier was to explore fine-tuning the FaceNet model with it's top 2 layers unfrozen and see whether we could make it more specialized in producing effective embeddings for our case (i.e., make all the known identity embeddings closer together and the unknown ones farther away from the known ones).

After fine-tuning the model, it was used to generate a new set of embeddings. To evaluate the new model, we once again graphed a histogram of the identity distance from the mean compared between the two groups (figure 5):

Figure 5: Distances in Training Set Before and After Tuning



After the tuning, the unknown faces were pushed further away from the known faces, removing the overlap we previously saw, which represented false negatives. However, this doesn't guarantee success when making inference in the validation set since we used the training set to tune the FaceNet model. Evaluating against the validation set, we achieved the following confusion matrix:

Table 5. Confusion Matrix for SVM Model τ with Unfrozen FaceNet

Label \ Pred	Diana	Erik	Unknown
Diana	98	0	0
Erik	0	23	0
Unknown	0	0	158

The results from this model show perfect classification! Using the combination of the **τ** threshold and fine-turning the FaceNet model, we were able to eliminate the false negatives, resulting in a doorbell which functioned securely and effectively.

Pipeline Security Features

To add further security features to the pipeline, we explored several features to try to avoid people being able to trick the facial recognition model by showing it photos or video. The functionality of these features is key to determining whether the pipeline can be made secure enough to function as a lock (which would require the highest level of security), or whether it is sufficient only to function as a doorbell. For either feature, however, having a way to ensure it is really the correct person at the door before notifying the residents would add security. To do this, we attempted to implement two different approaches: blink detection and liveness.

The blink detection approach we implemented was based on Dlib "shape_predictor_68_face_landmarks" pre-trained model (https://github.com/davisking/dlib-models) and following the example here https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/. This model can be used to identify eye positions, and can detect blinks by calculating euclidean distance between vertical and horizontal eye landmarks and comparing the change over time. By implementing this approach after performing inference on known faces, we were able to correctly identify our images from the pipeline which did not demonstrate blinking successfully. This approach could be used to stop people from fooling the doorbell with a picture of a known user. When implementing, we initially ran into some trouble with this approach due to the initial build of our model which was running at a slow frame rate (0.5 frames per second). At this frame rate the model only detected blinks when the subject blinked unrealistically slowly. However, once we further optimized the pipeline and removed a bug that was double-extracting images, the blink detection function worked successfully so we included it in our final pipeline.

The second security approach we implemented was OpenCV liveness detection (our approach was based on https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/). This approach uses a convolutional neural network trained on real images vs. video taken of an image being displayed from another device. One key limitation of this approach, however, is that a key factor in the comparison is based on the RBG balance of the video. When we implemented liveness in our pipeline it was able to differentiate between the real stream of Erik and a video of Erik played on a phone. However, setting the iPhone on night mode (which shifts the RBG balance and reduces blue light) was able to automatically defeat the recognition. Because of this gap, as well as due to the performance lag liveness caused when included in the pipeline, we decided to remove this feature from our final build.

Conclusions/Further Work

After completing these steps, we have concluded that we are capable of building a functional facial recognition pipeline based on a pre-trained transfer learning model, but running all local model training and inference on a Jetson TX2. This is an exciting proof of concept because it demonstrates that it is possible to avoid cloud training for detecting known user identities, allowing facial recognition to be deployed securely and avoiding the risks of transmitting user identities to the cloud.

While our project did not allow us time for further work to build pipeline security features, we did conclude, however, that the current model is not secure enough to function as an effective lock device and would be more appropriate as a doorbell which would notify homeowners if their friends or family were outside the house. To function effectively as a lock, the device would need to include further liveness detection features, possibly including depth detection as used by the iPhone facial unlock feature. Nonetheless we are confident that with more time and potentially with the use of a second camera, it would be possible to create a secure lock with all edge inference.

In considering whether our pipeline could be deployed in a commercial setting, one question which bears further consideration is the cost and size of a Jetson. A stand-alone Jetson TX2 module costs in the range of 400 dollars per unit which is likely prohibitively expensive to include. However, with further optimization it would likely be possible to run a variation of our pipeline on the Jetson nano which is available for approximately 100 dollars per unit and more within the realm of affordability. While such a device would likely be more costly than Ring or Nest, many customers may be willing to pay a premium for such a feature. Apple's HomeKit Secure Video feature is already touted as a selling point based on encrypting home security footage at the edge before cloud storage (https://www.the-ambient.com/guides/apple-homekit-secure-video-explainer-1814) and a feature which can perform secure edge facial recognition would likely be valued by users. While our device needs more work to be production ready we were nonetheless excited to be able to develop an effective facial recognition pipeline which can offer more security than the market-leading smart doorbells today.