Springboard – Data Science Camp
Capstone Project 2

Fútbol Match Highlights

By Tom Widdows
April 21, 2020

Table of Contents

(1) Introduction

<u>Problem</u>

College coaches require soccer players to submit highlight video and game film to be considered for an athletic scholarship. The issues facing student-athletes to provide highlight videos and game films include recording games is time-consuming, athletes must rely on someone else to take the video, most people would prefer to watch the game and not be a videographer, processing the game file is time consuming and the required hardware and software are expensive.

<u>Stakeholders</u>

Stakeholders include high school athletes, their parents and college sports recruiters.

Figure 1 – Athletes and Parents

<u>Results</u>

The goal is to identify a target player in a photo (ultimately a video) with a high degree of accuracy. By managing the quality throughout the process, the resulting accuracy of locating (and ultimately tracking) the target player should be very high.

<u>Post project</u>

Two videos will be stitched together, face images will be extracted from the videos (which are images), the face recognition utilized in this project will be applied to the video image and the target player identified allowing the software to effectively track the player.

<u>Implementation Details</u>

Additional implementation details and all code related to the project can be found on my GitHub repository:

https://github.com/8-Waste/Springboard/tree/master/Capstone%202%20-%20F%C3%BAtbol

(2) Approach

<u>(2.1) Data Acquisition and Wrangling</u>

Facial recognition requires a sequence of related steps including ensuring the source image is sharp, locating faces in the image, provide quality checks to ensure quality faces, centering and measuring unique facial features, comparing the target face to known faces and making a prediction.

The data source for this project is images of fútbol players taken by Tom Widdows. Although the images are from still photographs, the application should be easily enhanced to work with frames of a video. The images are primarily in a proprietary Canon format. After consolidating the photos, they are converted, if necessary, from the proprietary Canon format (.CR2) into JPG files so we can easily access them in python.

To evaluate image sharpness, A support vector machine (SVM) that uses labeled photos (sharp and blurred) with 12 features consisting of 3 measurements (mean, variance and maximum) each of 4 edge detection algorithms (laplace, sobel, roberts and canny) is fit and saved.[i] The source images, with the same 12 calculated features, are processed through the trained model and identified as a sharp or blurry image.

(2.2) Storytelling and Inferential Statistics

The sharp images are processed through two separate face detection algorithms, a Harr-cascade classifier (HARR) included with OpenCV and a Multi-task Cascaded Convolutional Network (MTCNN)[ii].

The images below (Figure 2) were created using the HARR classifier for face detection. The sample images below certainly show the HARR classifier identified many non-faces as faces (false positives).

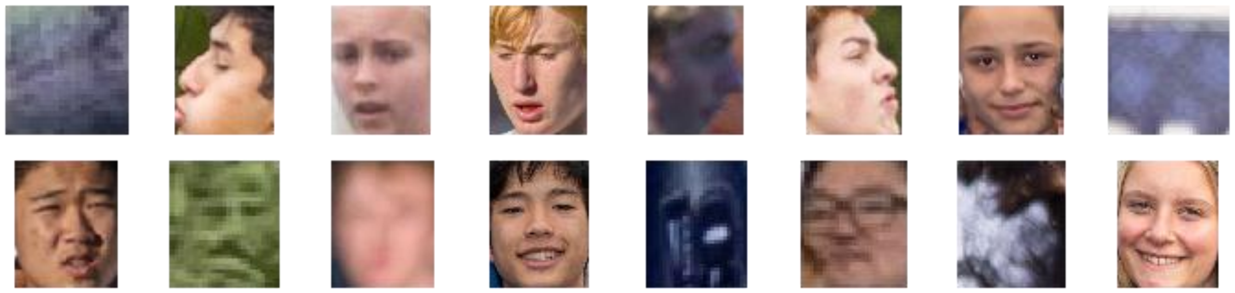


Figure 2 – Faces from HARR Classifier

The images below (Figure 3) were created using a Multi-task Cascaded Convolutional Network (MTCNN) for face detection. The MTCNN identified many non-faces as faces (false positives) but at first look, appears to have performed better than the HARR classifier.

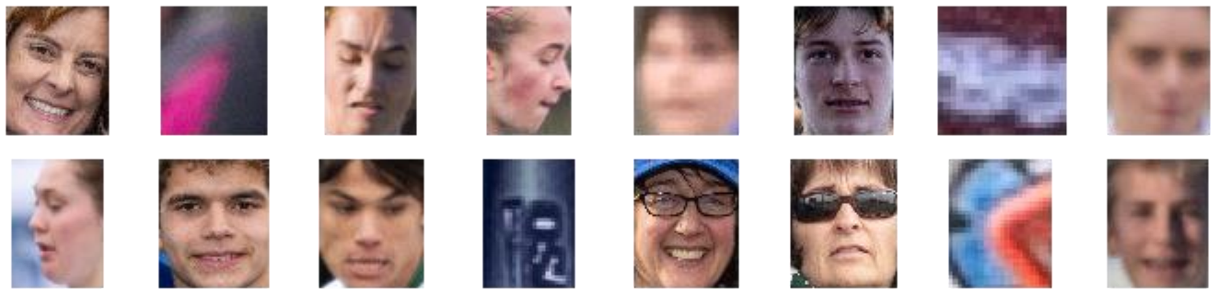


Figure 3 – Faces from MTCNN Classifier

A face in an image is rarely more than 1% of the entire image (calculated using square pixels) and the median percentage is 0.20%.

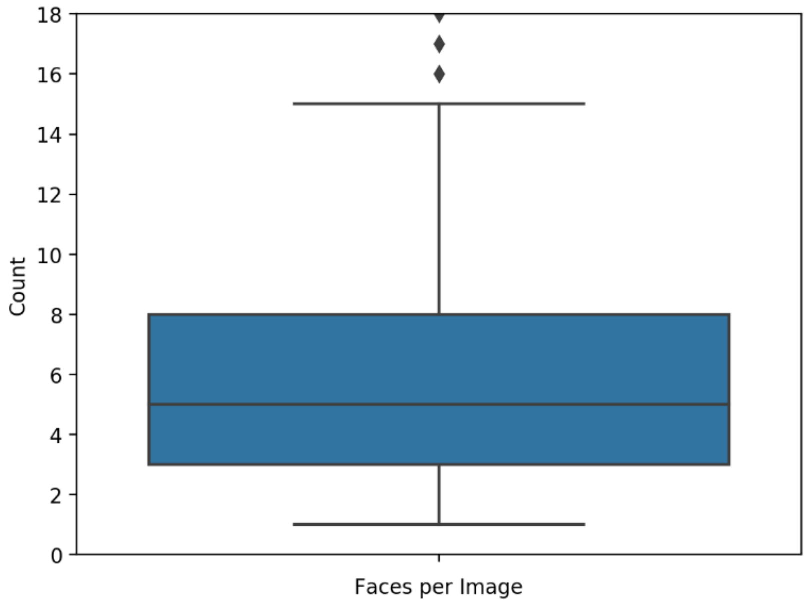


Figure 4 – Faces per Image

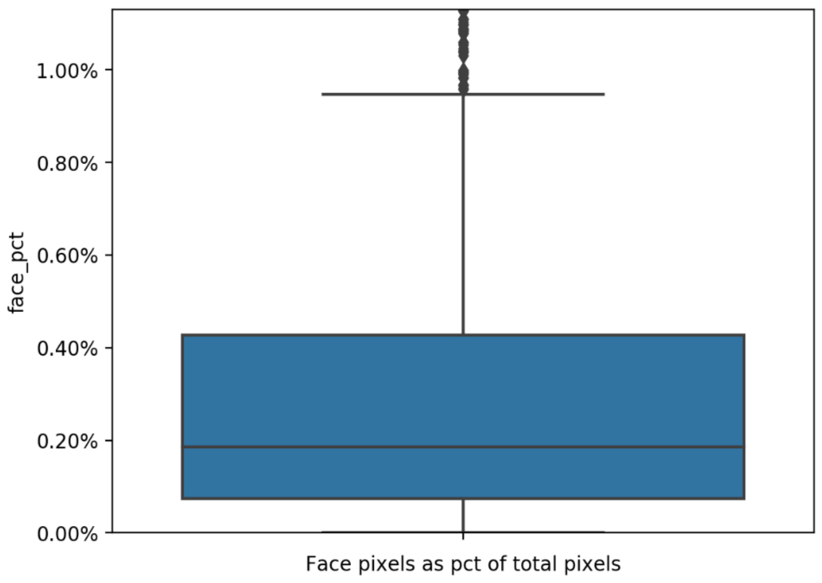


Figure 5 – Face Percent of Total Pixels

An image rarely contains more than 15 faces and the median number of faces in an image is five.

Now let’s examine some of the color properties of an identified face (Figure 6). We will calculate the average color of the face, the top 5 dominate colors in the face and the percent of skin tones in the face. Below are 3 random faces with the calculated color values:

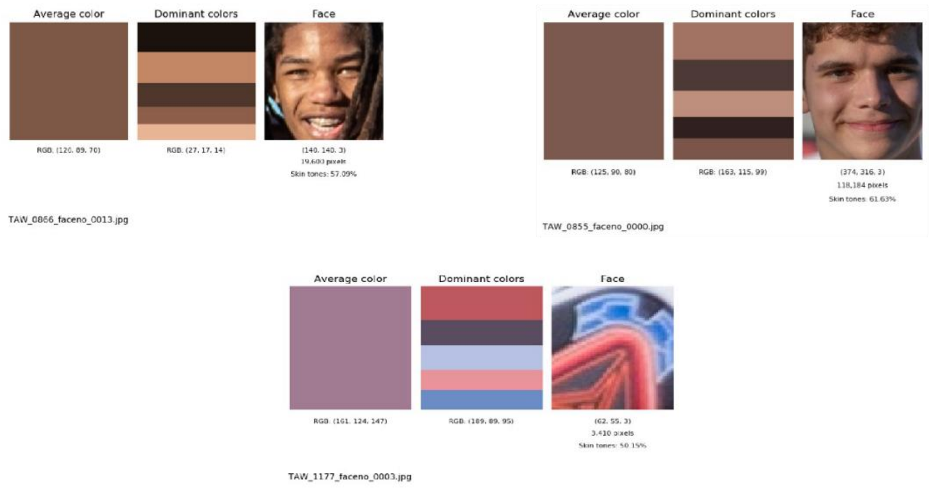


Figure 6 – Image Color Distribution

.

Let’s calculate if the difference in skin tones between face and non-face images is statistically significant. In each image, we will programmatically count the number of faces and take an equal number of non-faces (or more accurately a random area the same size as the face). Since the median face covers only 0.20% of an image and rarely covers over 1% of an image, the result should include very few faces and it would be unlikely the face would be centered. Below is a sample of the calculated non-faces (Figure 7):

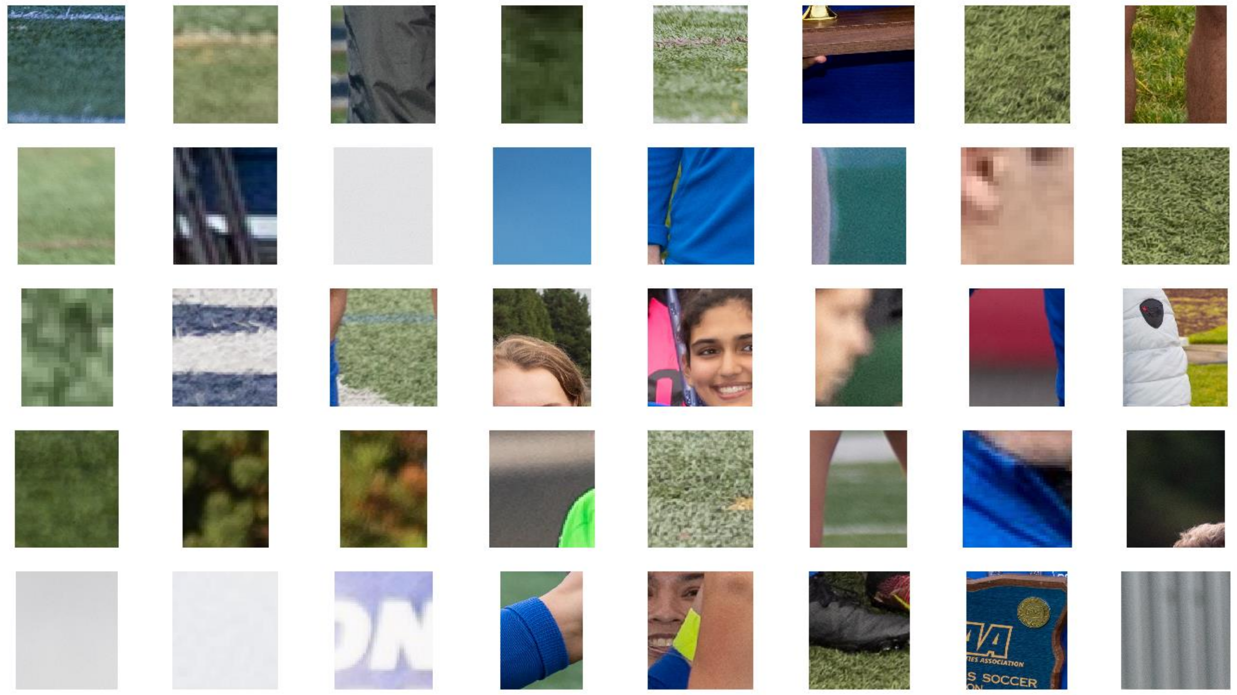


Figure 7 – Object (non-face) images

Now let's examine some of the color properties of a non-face (Figure 8). Again, we will calculate the average color of the non-face, the top 5 dominate colors in the non-face and the percent of skin tones in the non-face. Below are 3 random non-faces with the calculated color values:

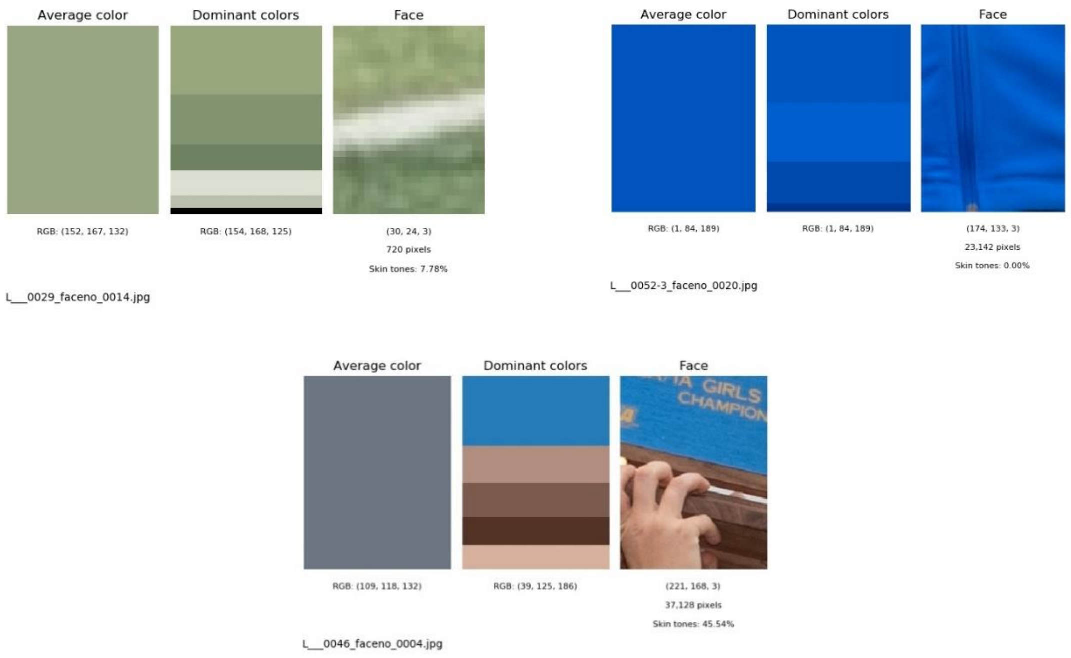


Figure 8 – Color Properties of Non-Face Images

Let's test if the means of the percent of skin tones are identical. The hypothesis test is to test if the difference in percent skin tones in a face image is statistically significant. The null and alternative hypotheses are as follows:

H_0: Percent Skin Tones has no effect on face identification, means are same

H_1: Percent Skin Tones has effect on face identification, means are different

Samples:	**931**
Mean:	**82.39% (dashed red line)**
Std Dev:	**16.86%**
Median:	**87.69%**

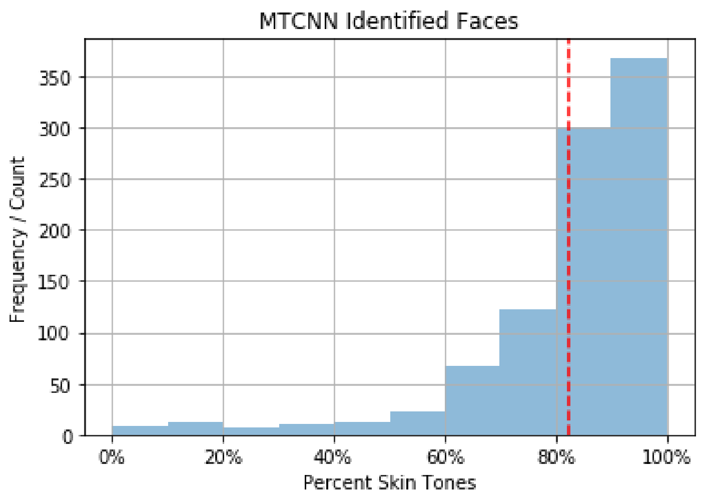


Figure 9 – Percent Skin Tones

Samples:	**931**
Mean:	**19.72% (dashed red line)**
Std Dev:	**26.71%**
Median:	**5.98%**

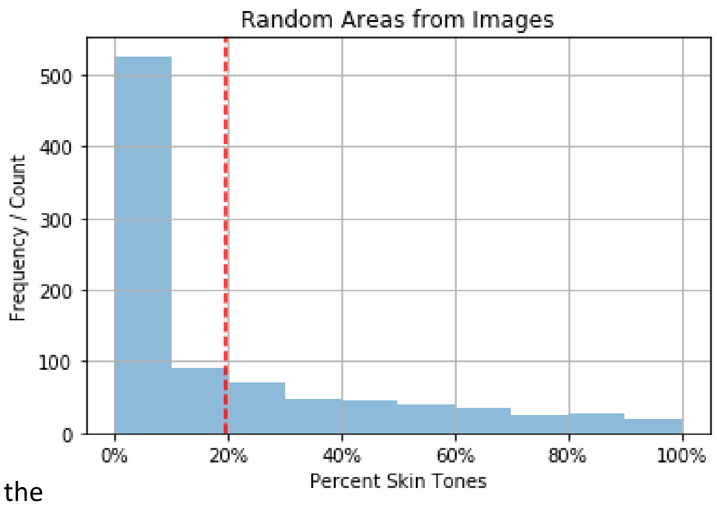


Figure 10 – Percent Skin Tones

I calculated a two-sided t-test for the null hypothesis that the samples have identical average (expected) values utilizing a significance level of .01. The p-value was 0.0000000 so **I rejected the null hypothesis in favor of the alternative hypothesis.**

H_1: Percent Skin Tones has effect on face identification, means are different

(2.3) Baseline Modeling

The application takes a source image (Figure 11) and processes it through the **Face Detector** model to identify the location of the faces (red squares). The application processes the pixels inside the red boxes through the **Face vs Object Model** to remove non-faces. The surviving faces are then processed through the **Focused vs Unfocused Model** to eliminate out-of-focus images. The surviving faces are then compared to the target face which is done by the **Recognizer Model**. The Recognizer Model utilizes a pretrained deep neural network (DNN) from OpenFace implemented in Python and Torch for face recognition.[iii]

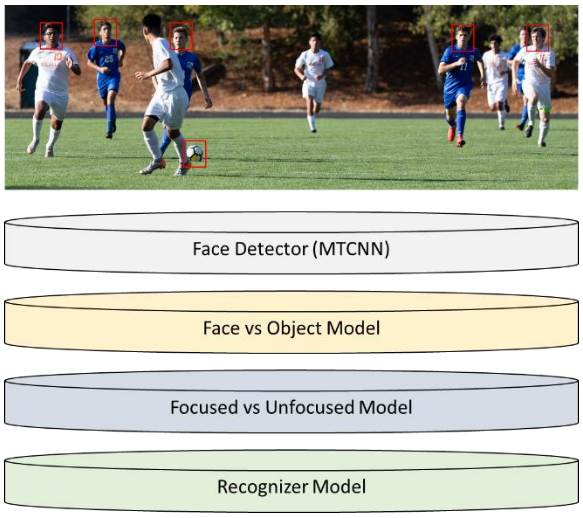


Figure 11

There are 5 steps (notebooks) to successfully run this application. Steps 1, 2 and 3 involve training and saving models that are used in steps 4 and 5 (Figure 12).

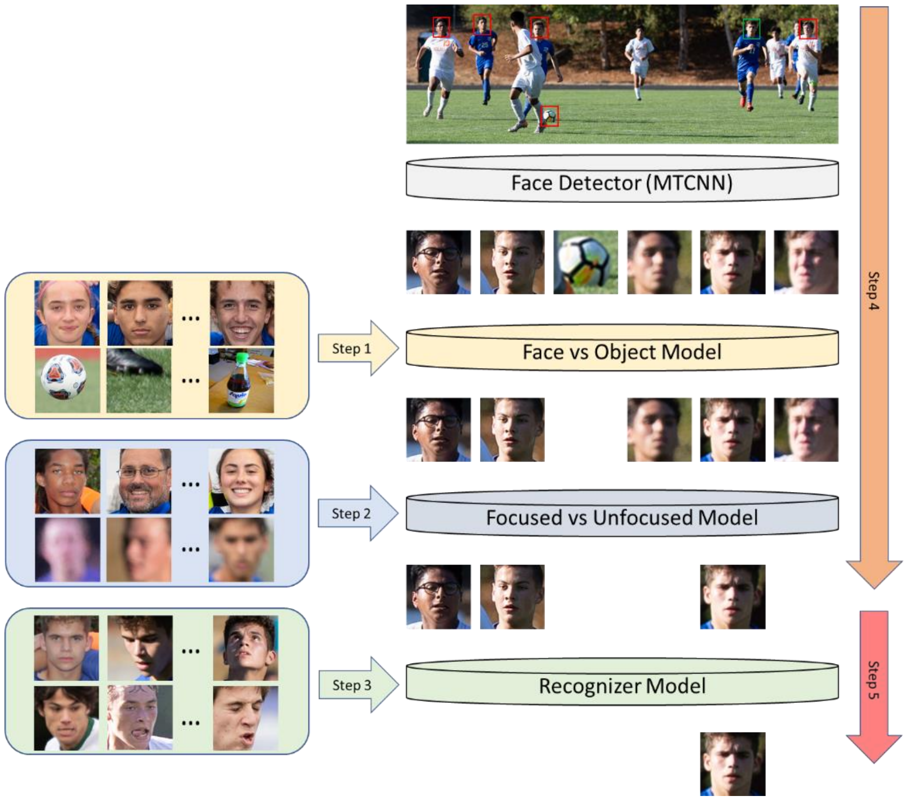


Figure 12

(2.3.1) Face vs Object (non-face)

The Harr-cascade classifier (HARR) included with OpenCV and a Multi-task Cascaded Convolutional Network (MTCNN) classifier were excellent at identifying in and out of focus faces, but the classifiers incorrectly identified many objects as faces (Figure 13). As discussed in Storytelling and Inferential Statistics, skin tone was determined to be statistically significant in identifying faces vs non-faces. I found that training a model on Faces vs Objects (non-faces) was more effective then evaluating on skin tone.

I created a dataset of 200 faces and 264 objects (non-faces). The objects consisted of 264 random patches from my existing photos. I trained using a KNeighborsClassifier classifier and obtained a True Positive Rate (TPR) / Recall rate of 88.89% and 85.00% on my validation data and test (holdout) data respectively.

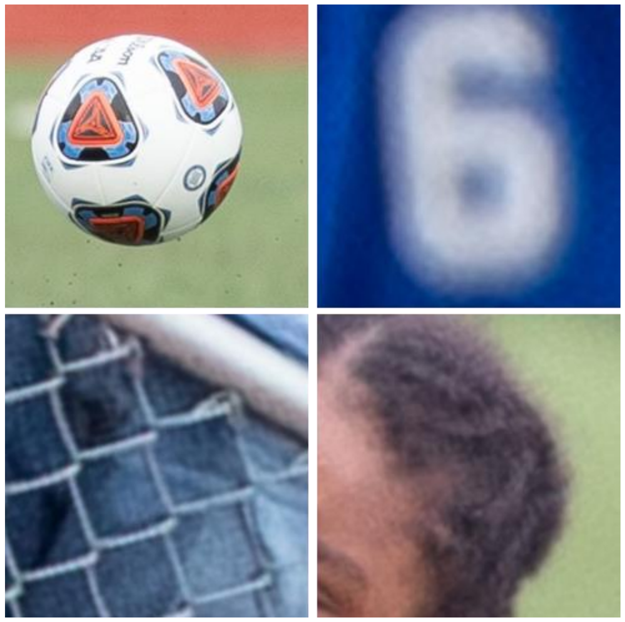


Figure 13

(2.3.2) In-Focus vs Out-of-Focus Images

As discussed in Data Acquisition and Wrangling, I initially trained a Support Vector Machine (SVM) that used labeled photos (sharp and blurry) with 12 features consisting of 3 measurements (mean, variance and maximum) where each have a 4 edge detection algorithm (laplace, sobel, roberts and canny). My source photographs, with the same 12 calculated features, were processed through the trained model and labeled as in-focus or out-of-focus. The issue discovered with this approach is quality photographs are taken with a low aperture that makes the in-focus area sharp and the rest of the photo intentionally out-of-focus. In Figure 14, player #11 is in sharp focus but the referee's face is completely out-of-focus and unrecognizable. The SVM classified this photo as out-of-focus which technically, for the photo, could be considered correct but eliminates a very sharp, in-focus face.

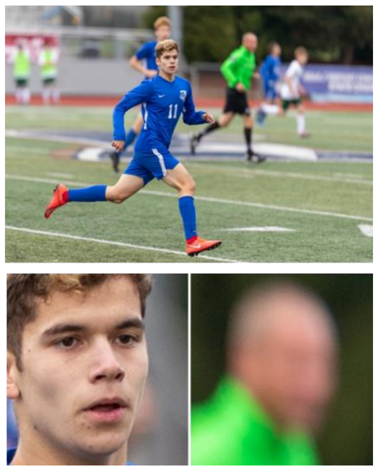


Figure 14

My revised apporach checks the focus of faces extracted from the images. I created a dataset of 1,701 out-of-focus faces and 655 in-focus faces. I trained a model using a GaussianNB classifier and obtained a True Positive Rate (TPR) / Recall rate of 85.99%, and 83.33% on my validation data and test (holdout) data respectively.

(2.3.3) Target Face Embedder

My dataset consists of 114 images of the target player and 566 images of other people. All the images are RGB (256, 256, 3). For preprocessing, I aligned all the faces (faceAligner) and utilized blobFromImage (mean subtraction and scaling) to prepare the photos for creation of an 128d vector face identifier utilizing a pretrained model based on OpenFace[iv]. The model was trained utilizing LBGM classifier utilizing a training, validation, and test set. The model was optimized utilizing Hyperopt using Kappa as the metric.

2.4 Extended Modeling

(2.4.1) Face vs Object (non-face)

As discussed earlier, it is important this model does not discard good faces (type II errors) as we utilize this model to remove non-faces. The baseline model utilized a KNeighbors classifier. The extended model was evaluated on 5 different classifiers (Figure 15). The best classifier (LGBMClassifier) was trained on the train data and an appropriate cutoff threshold was chosen. I increased the number of Positive and Negative samples in the dataset. The Recall rate increased from 88.89% to 89.06% on the validation dataset and from 85.00% to 85.92% on the test (holdout) dataset. My next step would be to incorporate Hyperopt to tune the model parameters and reevaluate the training data. I am also considering pairing up this model with the skin tone model discussed earlier. One possibility would be taking the rejected items (objects) and reclassify them back as faces if they successfully pass the skin detector model.

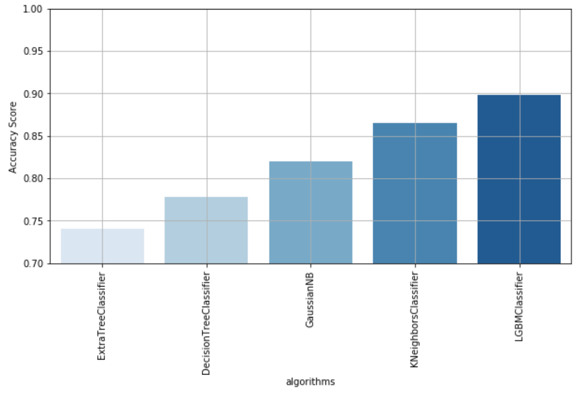


Figure 15 – Baseline Face vs Object (non-face)

(2.4.2) In-Focus vs Out-of-Focus Images

The baseline model utilized a GaussianNB classifier. The extended model was evaluated on 5 different classifiers (Figure 16). The model was trained on the best classifier (LGBM Classifier). I added In-Focus and Out-of-Focus images for a more robust dataset. The Confusion Matrix was evaluated, and an appropriate cutoff was applied. The big change in this model was switching from evaluation of entire images to the evaluation of individual faces. This change was so significant, I decided to incorporate it into the baseline model. I incorporated Hyperopt for model tuning. By changing the classifier, threshold cutoffs, Hyperopt evaluation metrics and evals, the model achieved a Recall of 94.92% and 93.94% on the validation and test (holdout) data respectively.

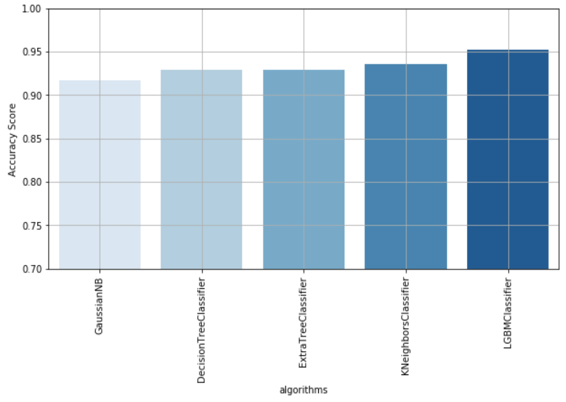
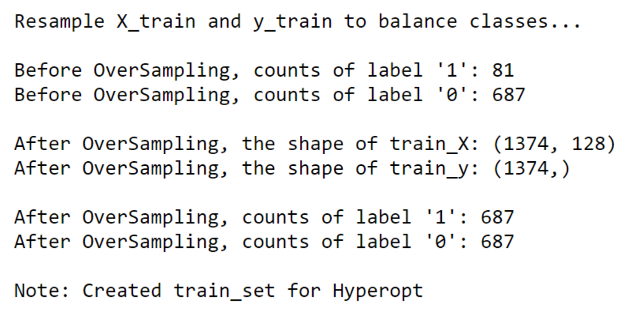


Figure 16 – In-Focus vs Out-of-Focus Images

(2.4.3) Target Face Embedder



```
Resample X_train and y_train to balance classes...

Before OverSampling, counts of label '1': 81
Before OverSampling, counts of label '0': 687

After OverSampling, the shape of train_X: (1374, 128)
After OverSampling, the shape of train_y: (1374,)

After OverSampling, counts of label '1': 687
After OverSampling, counts of label '0': 687

Note: Created train_set for Hyperopt
```

Figure 17 – SMOTE Summary

The baseline Target Face Embedder achieved Recall score of 60% and 80% on the validation and test (holdout) data, respectively. I was extremely disappointed with these results. I began by adding additional unknown faces (Negative Class). I realized I made an error in creating the training datasets. When the data was being labeled, I decided to label everyone on the team. Competitors and fans were classified as Unknown and used in the Negative Class dataset and teammates were classified under their name. However, since I am searching for one target player, I should have included all the teammates in the Negative Class (Unknown). This was corrected during Extended Modeling. Since this dataset is imbalanced, I balanced the training data using SMOTE (Figure 17). Unfortunately, this did not improve the model's results. I also utilized a cutoff threshold and multiple Hyperopt configurations.

(3) Findings

The business problem for my application is:

College coaches require soccer players to submit highlight video and game film to be considered for an athletic scholarship. The issues facing the student-athlete are:

- *Recording and processing video is time-consuming*
- *The athlete is playing so they must rely on others to take the video*
- *People would rather watch the game instead of being a videographer*
- *Processing the video is time consuming*
- *The required hardware and software are expensive.*

The Springboard portion of my project is focused on extracting faces from the images, applying facial recognition and identifying the target player (Figure 18). Having tried various face detectors, I am building the initial application with the MTCNN detector. The models should work with any face detector.

The solution to the business problem is to easily be able to create a college highlight video. One import step in accomplishing that is accurate and reliable face detection. The first section describing each of the models below addresses how that specific model helps achieve the goal of creating accurate and reliable face detection.

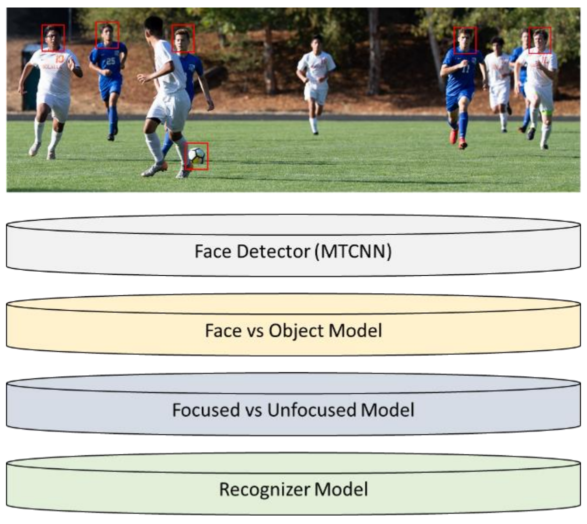


Figure 18

(3.1) Face vs Object (non-face)

Inaccurate faces become additional noise in Facial Recognizer model. By eliminating photos of soccer balls, cleats, trees… that were incorrectly classified as faces, the dataset (faces) that is being presented to the recognizer model will be cleaner which will increase speed and accuracy. In my final testing of the application, I will run the entire process including and excluding the Face vs Object (non-face) to provide clear evidence of the model's value and its cost in time.

The results of the baseline model through the extended model version 2 are below (Figure 19):

		Validation Data				Test (Holdout) Data				Other		Training		
	Version	Recall (TPR)	Kappa	F1	MCC	Recall (TPR)	Kappa	F1	MCC	Cutoff	SMOTE	Positive Class	Negative Class	Classifier
06 - 02 Baseline Face vs Object	01	88.89%	75.86%	86.49%	75.95%	85.00%	73.89%	85.00%	73.89%	50%	No	200	264	KNeighborsClassifier
08 - 02 Extended Face vs Object	01	88.28%	79.20%	91.13%	79.43%	85.92%	78.07%	90.37%	78.62%	50%	No	712	488	LGBMClassifier
	02	89.06%	79.20%	90.84%	79.43%	85.92%	78.07%	90.37%	78.62%	42%	No	712	488	LGBMClassifier

Figure 19 – Face vs Objects Summary

The extended model v2 performed the strongest. The Recall rate of 89.06% on the validation data and 85.92% on the test data were the strongest set of numbers but not by much. As discussed above, I would like to take the rejected items and run them through an addition model (skin tones) and possibly reclassify them to back to faces based on the results. I will also try running the skin tone model prior to running the Face vs Object model.

(3.2) In-Focus vs Out-of-Focus Images

Out-of-focus images (Figure 20) become additional noise and are of no value to our recognizer model. Unlike objects (non-faces), out-of-focus faces have virtually all of the same attributes of a face making it more likely the recognizer could misclassify the face. Over 50% of the faces extracted from my photos where out-of-focus. In my final testing of the application, I will run the entire process including and excluding the In-Focus vs Out-of-Focus model to provide clear evidence of the model's value and its cost in time.

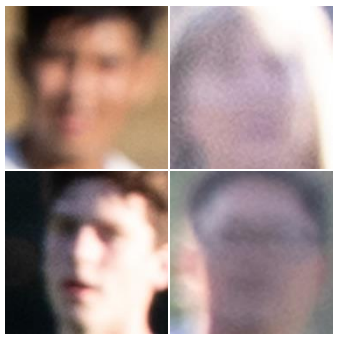


Figure 20

The results of the baseline model through the extended model version 6 are below (Figure 21):

		Validation Data				Test (Holdout) Data				Other		Training		Hyperopt		
	Version	Recall (TPR)	Kappa	F1	MCC	Recall (TPR)	Kappa	F1	MCC	Cutoff	SMOTE	Positive Class	Negative Class	Evals	Metric	Classifier
06 - 01 Baseline Focus vs Unfocused	01	85.59%	79.50%	85.23%	79.51%	83.33%	79.74%	85.27%	79.78%	50%	No	655	1,701			GaussianNB
08 - 01 Extended Focus vs Unfocused	01	96.08%	87.15%	90.76%	87.16%	90.91%	90.40%	93.02%	90.45%	50%	No	655	1,701	50	AUC	LGBMClassifier
	02	93.22%	83.01%	88.00%	83.27%	93.94%	87.61%	91.18%	87.68%	30%	No	655	1,701	50	AUC	LGBMClassifier
	03	94.07%	84.67%	89.16%	84.90%	92.42%	86.52%	90.37%	86.56%	30%	No	655	1,701	50	Kappa	LGBMClassifier
	04	94.07%	84.14%	88.80%	84.40%	93.94%	88.59%	88.63%	88.63%	30%	No	655	1,701	50	MCC	LGBMClassifier
	05	94.92%	84.22%	88.89%	84.56%	93.94%	86.64%	90.51%	86.75%	30%	No	655	1,701	1,500	MCC	LGBMClassifier

Figure 21 – In-Focus vs Out-of-Focus Summary

The extended model v5 performed the strongest. The Recall rate of 94.92% on the validation data and 93.94% on the test data were the strongest of all the models. The LGBM classifier performed much better then the GaussianNB classifier. Version 1 of the extended model performed well but less consistent on the validation and test (holdout) datasets. Changing the Hyperopt measurement metric and number of evals had a negligible impact on model performance.

(3.3) Target Face Embedder

This model is key. This portion of the application trains a model on target faces and other faces. When an unknown photo is presented to the application, the application applies the detector module to locate the faces, applies to Face vs Object (non-face) module to remove non-faces, applies the Focused vs Unfocused model to remove out-of-focus faces and finally, the surviving image(s) are run through the Target Face Embedder (Recognizer) model. With out this module function well, the entire application will be of limited value.

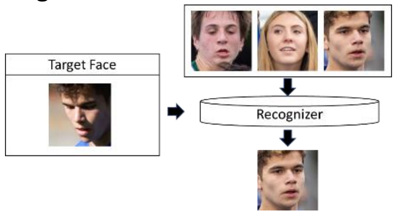


Figure 22

The results of the baseline model through the extended model version 5 are below (Figure 23):

		Validation Data				Test (Holdout) Data				Other		Training		Hyperopt		
	Version	Recall (TPR)	Kappa	F1	MCC	Recall (TPR)	Kappa	F1	MCC	Cutoff	SMOTE	Positive Class	Negative Class	Evals	Metric	Classifier
06 - Baseline Target Face Embedding	01	60.00%	54.23%	61.54%	54.25%	36.36%	32.97%	42.11%	33.54%	50%	No	112	566	100	Kappa	LGBM Classifier
08 - Extended Target Face Embedding	01	80.00%	57.38%	62.75%	59.21%	90.91%	54.35%	60.61%	58.92%	6%	No	112	956	100	Kappa	LGBM Classifier
	02	80.00%	57.38%	62.75%	59.21%	90.91%	54.35%	60.61%	58.92%	6%	Yes	112	956	100	Kappa	LGBM Classifier
	03	80.00%	57.38%	62.75%	59.21%	90.91%	54.35%	60.61%	58.92%	6%	No	112	956	100	Kappa	LGBM Classifier
	04	80.00%	63.77%	68.09%	64.70%	81.82%	56.52%	62.07%	58.82%	6%	No	112	956	100	MCC	LGBM Classifier
	05	80.00%	57.38%	62.75%	59.21%	90.91%	54.35%	60.61%	58.92%	6%	No	112	956	100	Kappa	LGBM Classifier

Figure 23 – Target Face Embedder Summary

All of the extended models performed substantially better than the baseline model. The increase was a result of increasing the number of Negative Class items in the dataset (unknown people) and adjusting the cutoff threshold. I originally did not include the target player's teammates as unknown but from the perspective of the model, they are unknown as we are only searching for the target player.

(4) Conclusions and Future Work

(4.1) Conclusions

The MTCNN detector is very effective at locating faces.

It is possible to remove objects (non-faces) from the detector.

Performance improves with face recognition as more images of the target player are provided (I discovered this while creating the baseline model).

Face recognition is more effective when the target is looking directly at the camera. The more the target shifts from that position, face recognition becomes much more challenging. The players on the pitch rarely will be looking towards the camera. Having pictures from all angles of the face that include both eyes is essential.

(4.2) Future Work

- Setup TensorFlow with GPU to increase speed of MTCNN
- Try alternative face detection algorithms. The structure of the application makes it plug in different detection algorithms.
- Evaluate the impact on accuracy and execution time by disabling the Face vs Object (non-face) and Focused vs Unfocused algorithm.
- Preprocess images in Focused vs Unfocused with OpenCV’s imageFromBlob function (mean subtraction and scaling) and evaluate results
- Increase the quality of images from the pitch and of the target player
- Incorporate video
- Zoom video based on location of target player
- Recognizing other parts of a player (e.g. numbers on socks, shoe color)
- Incorporate Hyperopt into remain models for tuning
- Tying together face recognition and recognition of other features (e.g. number on socks, number on uniform, shoe color…)
- Have multiple profiles for target player. For example, one might be looking directly at the camera, one at 40 degrees and one at 80 degrees. Evaluate impact to recognition.
- Allow multiple target players

(5) Client Recommendations

- If you plan on processing the video and images, you will need a current computer with a GPU and a fast drive.
- Proper image acquisition (video or pictures) is critical for good facial recognition. Aperture should be closed down (deep depth of field) so the entire image is in focus. The shutter speed should be 1/1000 of a second or faster. This will make the video jumpy but will allow the individual images extracted from the image to be sharp (in-focus) and make facial recognition from video possible. Sacrifice ISO for shutter speed. If you must open the aperture, ensure the entire pitch remains in focus. There are many online simulators to help with depth of field calculations. https://dofsimulator.net/en/
- Have multiple photos at different angles of your player. Ideally the player would be on the pitch in a game. The sharper the photos of the target player, the better.
- If you do not have photos of the target player, extract images from the game file, extract the faces from the images and segregate the images of the target player.
- If you want to label all the team members, you can extract images from the game file, extract the faces from the images and label all the players. The application currently allows one target player, but the unknown faces section was built so you can categorize the teammates as unknown but keep them separated so they can be easily shifted from unknown to a target player.

(6) Consulted Resources

https://www.pyimagesearch.com/2018/09/24/ OpenCV Face Recognition

https://www.pyimagesearch.com/2017/05/22/ Face Alignment

https://github.com/Nestak2/image-sorter2 Image Sorter 2

https://cmusatyalab.github.io/openface/ OpenFace

https://www.kaggle.com/cpmpml/optimizing-probabilities-for-best-mcc MCC for LGBM.cv

http://www.vision.caltech.edu/pmoreels/Datasets/Home_Objects_06/ Object dataset

https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

www.medium.com

www.stackoverflow.com

---

[i] Inspired by a Kaggle Blur dataset and a Kernel by Harini Narasimhan, https://www.kaggle.com/harininarasimhan/blur-detection-with-feature-engineering

[ii] Inspired by pyimagesearch, Adrian Rosebrook, https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

[iii] B. Amos, B. Ludwiczuk, M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[iv] B. Amos, B. Ludwiczuk, M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.