Facial Attention Capture Evaluation System (FACES)
Real-Time Video Emotional Classification
ECS 171

Aliaksei Chumakou Shivam Desai Rohit Dhamankar Luis El Srouj Rachel Hanna Brendan Kelley Munir Nur Huyen Pham Ravishdeep Singh Jacob Smith Chantheary Soth Ryan Stager

Abstract

An instructor's ability to teach is not only dependent on their capacity to provide information, but is contingent on an ability to assess the engagement and attentiveness of the audience. Under large lecture conditions, human perception is incapable of tracking the emotional characteristics of a multitude of attendees. Advances in deep learning methods, such as the use of Convolutional Neural Networks (CNN) for object detection, have made it easier to detect and process multiple faces simultaneously. By creating bounding boxes for potential faces, we are able to efficiently segment an input image and extract multiple subjects for further processing. We further leverage the successes of CNNs along with Support Vector Machines (SVM) in order to classify these faces into ten unique sentiment classes. Paired with implementations of existing object detection models, we are able to perform real-time facial detection and emotional classification. Our most successful classification models are capable of correctly identifying the emotions of numerous faces in roughly a tenth of a second. Our system, FACES[1], demonstrates the ability to present lecturers with feedback on total classroom attentiveness.

1. Introduction

In recent years, deep learning methods have been able to leverage large datasets to learn representations of faces and perform classifications with similar or even greater success than human beings (1). With respect to classifying emotions of facial images, most conventional approaches divide the task into three parts: facial detection, feature extraction, and expression classification. In recent years, advancing deep learning approaches, such as CNNs and long short-term memory (LSTM), have been used to reduce the dependence on heuristic classification models and other prepossessing techniques (2).

Although most conventional approaches towards emotional classification have been successful to some degree, they are not typically utilized further within the realm of a useful application. Our system, FACES, takes emotional classification a step further by using results to examine and asses the engagement and attentiveness of subjects in a large audience. Additionally, most current approaches are designed to process static images of faces. Given that our system would be used in lecture conditions over significant periods of time, we examine the implementation of our model in real-time using live video input as well as pre-recorded videos.

For this project, we committed to a fully developed pipeline to detect the faces within an image and determine their respective attentiveness. We passed frames of live video into our object localization model, which isolates faces in the frame. Then, we pass the detected faces to one of our classification models to predict the emotions of potential subjects. Results of the classification were then utilized to implicitly determine the engagement of the subject with presented material. Our image segmentation models were implementations of three pre-trained systems: *Haar*, *YOLO*, and *faced*. Using *faced*, we are able to accurately segment facial images in a live video and feed them into classifier models for real-time results.

Using the FacesDB dataset[2], we trained two separate classification models: a CNN model and CNN-SVM modew (able to classify emotions at an accuracy of roughly 56.67% and 53.33% respectively). Though the relevance of labels to attentiveness is somewhat forced, our system demonstrates the feasibility of multiple, real-time facial-expression-classification which can provide metrics for audience engagement to professors, lecturers, and other presenters. Further research can be conducted to determine the most prevalent facial expressions in an attentive versus inattentive audience, and a new dataset can be constructed incorporating

[1] source code: https://github.com/bjkelley/ECS_171_ML_Measure_Attentiveness

[2] download at: http://app.visgraf.impa.br/database/faces

these features.

2. Methods

To implement multiple facial expression recognition, our system is divided into two separate modules.

The first module, object localization, takes an image as input with an arbitrary number of faces. Then, it uses OpenCV and one of the three defined models to compute bounding boxes for each face in the image, returning the vertex coordinates of each bounding box and an array of numpy arrays representing the faces, resized to match the expected input of the emotional classifier.

In the second module, we have implemented two distinct convolutional neural networks with varying layer architectures and loss functions. Each model outputs a 10-dimensional vector with elements representing the probabilities of a sample existing in a given class. Our system is implemented entirely in Python using the Tensorflow machine learning library(3).

2.1. Data Set

We pulled testing and training data from the FacesDB dataset mentioned in the introduction. The dataset consists of 36 separate people and multiple different expression classes. Our application is dependent on only 10, resulting in a total 360 separate images. The chosen classes include: 'neutral frontal', 'joy', 'sadness', 'surprise', 'anger', 'disgust', 'fear', 'opened', 'closed', and 'kiss'. The dataset included young and elderly individuals, multiple genders, and a variety of physical characteristics, making it diverse enough to represent most lecture audiences. We One-Hot-Encoded the classes to obtain certainty percentages for each emotion.

We then divided the data into the training and testing set by selecting the first thirty individuals. This large set was provided to the models for training. The remaining six individuals were used as the testing set. We divided by person to ensure that we are testing our model's accuracy on faces it has never seen before. This gives us a better idea of how well the model generalizes to new individuals that it has not trained with.

When considering our overall goal of discerning lecture attentiveness, we assigned each expression class to a super set - either "Attentive" or "Inattentive" - as shown in Table 1. While these expression groups are somewhat arbitrary, they do provide insight regarding the overall attentiveness of a person or people. Since the precision of these groupings is debatable, we leave the classes as ten individual expressions for training our emotional classifier. The groupings are only used later to interpret the results of the live video feed.

Table 1. Expression Sets

ATTENTIVE	INATTENTIVE
JOY	NEUTRAL FRONTAL
SURPRISE	SADNESS
OPENED	ANGER
KISS	DISGUST
	FEAR
	CLOSED

To reflect the correlation between certain expressions (for instance, fear and surprise may be difficult to differentiate) we not only kept track of our models' testing accuracy, but also their top 3 accuracy. This metric demonstrates the fraction of samples where the true classification was reflected in the top three most likely classes predicted by the model.

2.2. Data Pre-processing

The goal in data pre-processing was to format our chosen dataset from its directory structure into a manner easy for our neural network to digest. Since we implemented our models using the Tensorflow Keras API, the tensorflow dataset object became the format of choice. Aggregating the images into this object then became a manner of iterating through the directories in sorted order (as each sample's label relied on both name and location in the file system). Each image was then transformed into a Numpy matrix where it could then be formatted to meet the desired specifications for consumption by the model. Addtionally, a *Haar* cascade classifier was used to detect facial locations in each sample and square-off images for use by the emotional classifiers. The dataset object allows us to transform our samples from their original format, to an iterable and transformable class of image-label pairs ready for training.

A suite of methods for image manipulation was also created for sample formatting and dataset augmentation. Our models were implemented with different requirements; in one model the samples needed to be converted to single-channel, gray-scaled images, while the other was training with three-channel (RGB) images where the pixel values were normalized on $[0, 1]$. Both methods make tradeoffs between the speed-boosts imparted by a reduced sample representation, and a robustness to lighting conditions resulting from a richer set of information(4).

Efforts in bootstrapping were also made; each sample was repeated a number of times and processed by a random subset of transformations. These modifications included random rotations of up to 7 degrees, the addition of Gaussian noise ($\mu = 0, \sigma^2 = 0.005$), and the addition of speckle noise ($\sigma^2 = 0.05$). These transformations were intended to improve the models' resilience to poor image quality and augment our training dataset.

2.3. Object Localization

For the object localization task, we utilized pre-trained models and algorithms to detect faces in a video-frame or static image. We implemented three different models: *Haar*, *YOLO*, and *faced*.

2.3.1. HAAR

The *Haar* Classifier and other boosted cascade object detection classifiers are early examples of machine learning algorithms. Originally published in 2001(5), *Haar* classifiers quickly became known for their fast and efficient classification of objects. In order to function, *Haar* classifiers have 3 main components: Integral Image representation, a modified *Adaboost* learning algorithm, and the concept of a cascade classifier.

Integral images are effectively various different types of convolution filters. These filters can be generated or hand-designed, and can correlate to different features in an image. These filters are then trained over several thousand different positive and negative features using *Adaboost* to select the more effective features and repress less effective features. Once the strongest features have been selected; the cascade classification begins. This is where *Haar* classifiers gain most of their speed. This component first runs a subset of the integral images over the input image in a systematic way. The regions which were not positive are then discarded. The positive sections of the input image are then subjected to another round of Integral images filters and this is repeated for all of the effective integral images. By doing this, the majority of the negative samples in the image are pruned excitingly quickly reducing the amount of computation needed for classification.(5) However, due to the reliance on hand curated filters and lack of non-linearity in the filters *Harr* classifiers perform poorly compared to modern deep learning techniques.

2.3.2. YOLO

YOLO is a modern real-time object detection algorithm that frames object detection as a single regression problem. This is different than many other object detection algorithms where classifier algorithms are repurposed to perform detection. In *YOLO* a single convolutional neural network is used to predict the bounding boxes of objects along with the class probabilities of those boxes. The output of the bounding box is in terms of 5 predictions: *x, y, w, h* and a confidence score which represents the intersection over union between the predicted box and the ground truth.(6) For our experiment we used an updated version of the algorithm, *YOLOv3*. *YOLOv3* fixes some of the drawbacks the original *YOLO* algorithm had such as having worse predictions for small objects in a frame. (7) This helps with our objective since in a lecture room we would want faces in the back of the room to be detected as well since we can assume they will be more likely to not be paying attention.

In terms of actually implementing the algorithm for our project we used a pretrained network from Darknet, an open source neural network framework, by using a configuration and model weights file we found online. When reading in the frame from a camera we used OpenCV's deep neural network pre-processing functions to turn it into a 4D blob. It is then fed into the loaded neural network and produces a list of the prediction outputs for each potential box frame for a face. The bounding boxes with a confidence lower than 0.5 are removed and the faces are finally resized in order to be fed into one of the emotional classifier models.

2.3.3. FACED

faced[3] is a custom open-sourced neural network made exclusively for face detection applications. As a result it is a lightweight model, making it tailored for real-time face detection. *faced* is a combination of two deep neural networks, implemented in Tensorflow. The first stage is a custom fully-convolutional neural network based on YOLO. It takes a 288x288 RGB image and returns a 9x9 grid. Each cell predicts bounding boxes and probabilities of faces. The second stage is a custom standard convolutional neural network with fully connected layers, which takes the cells containing faces from stage one and predicts bounding boxes for the faces. This stage is necessary because the outputs from stage one are not very accurate, so stage two improves the bounding box quality and accuracy.

2.4. Emotional Classifier

For the construction of our emotional classification models we used Tensorflow's implementation of Keras for training and evaluation. Throughout training, we kept track of metrics including **accuracy rates** and the **top 3 predicted class accuracy** (proportion of predictions where true label is within the top-three most confidently predicted classes). Both models were trained with the ADAM optimizer which is an adaptive learning rate method; the algorithm computes individual learning rates for different parameters (8). Initial model architectures were developed empirically with reference to speed and accuracy; once two layer architectures were selected (a pure CNN model and a CNN-SVM implementation), a gridsearch was implemented for hyper-parameter selection.

[3]https://github.com/iitzco/faced

Before training, we split the dataset into training and evaluation subsets. For training, we take the first 30 subjects in the dataset with their ten respective samples of each emotional class. Our test dataset is then populated with the final 6 subjects and each of their samples. The training set is then bootstrapped in accordance with §2.2 before shuffling and batching. Our test set is passed through the model without any transformation or bootstrapping.

2.4.1. Convolutional Neural Network

Convolutional Neural Networks (CNN) have gained popularity for use in facial emotional recognition due to their robustness in tracking face location changes and scale variations (4). The layer architecture of this model was developed with reference to other state-of-the-art CNNs designed for real-time processing(9).

CNNs operate by passing a number of small kernels over the input. These kernels are matrices which perform an element-wise multiplication with a subset of pixels from the input image to indicate the correlation of that patch with the kernel. Essentially, these kernels would represent an abstract edge or feature, and would pass over the entire image to produce a map indicating the presence of that feature at each location in the image. Thus, a convolution layer would transform an input image from a representation of colors at each pixel, to an arbitrarily abstract representation of specific features.

Our pure CNN model architecture accepts a preprocessed image as input, which is either one of the samples in the training or testing sets, or a resized face image provided by the image segmentation modules. The network takes in grayscale images that are $60x60$ pixels. The labels for the training data is represented as one-hot encoded vectors for each of the classes and thus the categorical cross-entropy loss was deemed most appropriate for this model. The equation is defined as follows:

$$CE = -\sum_{i}^{C} t_i \log f(s)_i$$

Where t_i and $f(s)_i$ are the ground truth and the final activation layer output for the CNN for each class i in C.

The model's architecture consisted of multiple blocks, each of which containing the following layers:

- *3 convolution layers*: each layer has a set of learnable $3x3$ filters that convolve with the input image to produce a set of 32 feature-maps, which are then passed through a rectified linear function (ReLu). Inputs are padded so that output retains matching dimensions.
- *1 max pooling layer*: follows the convolutional layer and is used to downscale the spatial resolution of the feature-maps and reduce the computational cost of the network.
- *1 dropout layer*: sets a fraction of the input units to 0 in order to reduce dependence on any single feature and prevent overfitting

At the end of the CNN network, a *flatten layer* converts the 3D feature-map into a single multi-dimensional feature vector. The vector is passed sequentially through three *fully-connected dense layers* where the last layer has a softmax activation layer. This is justified by the function's proven ability to minimize the cross entropy between estimated class probabilities and the ground truth distribution. (4) The softmax equation is given by the following:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j}^{C} exp(x_j)}$$

Where x_i are the scores inferred by the net for each class in C.

2.4.2. CNN-SVM

Support-Vector machines find an optimal hyper-plane to separate the feature space of dataset into binary categories (10). This feature space, x, may consist of raw inputs fed into the model, or some abstract representation of samples within the dataset. A hyper-plane within this space can be denoted as $f(x, w) = w \cdot x + b$, where w represents a set of weights and b defines a bias term. In order to find the optimal weights, an SVM solves the optimization problem:

$$\arg\min_{w,b} \frac{1}{2} r \cdot ||w||^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \hat{y_i})^2$$

Where r is a regularizer factor, N is the number of samples within the dataset, y_i is the ground truth result, and $\hat{y_i}$ is the predicted result given by:

$$\hat{y_i} = f(x_i, w)$$

The first term in our objective function represents the L2 norm of the model weights. Recent works have demonstrated that this term allows for greater stability during the optimization task over the L1 norm (11). Additionally, the L1 norm leads to a sparser dependence on the feature space, which may be disadvantageous given the presence of localized information in image classification.

Our second term applies the squared-hinge loss function across the dataset. Others have shown this loss function provides improvements in CNN classification performance(12) over the traditional hinge loss developed in (10). Convolutional Neural Networks transform an input image into a multidimensional feature space with information on the localization of features. This in combination with the SVM's ability to separate samples on a well-defined feature space makes it a good choice for this problem.

The architecture for this model varies from its pure CNN counterpart in hopes of reducing model complexity and timing. Here, the model accepts a three-channel (RGB) image with $128x128$ pixel dimensions; the pixel values are also normalized on $[0, 1]$. Its fundamental block structure consists of the following layers:

- *1 convolution layer*: each layer is strided by four with the exception of the first layer; this reduces the output dimensions in place of the pooling operation. The first block begins with 16 filters, while each subsequent block produces a new set with 8 additional filters from the last. Kernel size begins at a power of 2 equal to the number of blocks, and reduces by two so the last block uses a $2x2$ filter. As before, each layer is padded and passed through a ReLu activation function.

- *1 dropout layer*: improves the generalizability of the model as mentioned before.

Similarly to our pure CNN, the output of the convolutional blocks is passed through a *flatten* layer to produce a vector in a multi-dimensional feature space. A *fully-connected dense layer* provides the weights and biases for the SVM to produce a classifying hyper-plane. Having multiple neurons with uncorrelated input weights serves to extend the algorithm from its original binary classification to multiple classification.

2.4.3. Grid-Search

To tune the hyperparameters of both the CNN and CNN-SVM models, we implemented a Grid-Search. For the CNN, we tested three different numbers of layer sets (a set of layers includes three convolutional layers before adding a dropout) including 2, 3, and 4 hidden layer sets. The values of 0.2, 0.3, 0.4, and 0.5 were tested as dropout rates. For learning rates, we attempted 0.1, 0.01, and 0.001. Variations of batch sizes including 15, 30, and 60 were also used. For our CNN grid search, we tested every variation of these hyperparameters, for a total of 108 separate CNN's.

For the SVM model, we tested the same parameters above, with a few differences. We used 2, 3, and 4 *layers* instead of layer groups since it was a simpler model. We also tuned the L2 regularizer hyperparameter by testing values of 1, 0.1, and 0.01. This resulted in a total of 324 separate SVM models.

3. Results

Overall our results show that the pipeline we have created functions as expected. While our emotional classification models may not be as accurate as previous attempts, their lightweight structure enables them to make quick classifications - which is ideal for live video.

3.1. Object Localization

All three methods of object detection were evaluated on WIDER validations dataset (13). This includes 3200 images of varying scale, pose, occlusion, and instance count. Intersect over union (IoU) was used for the evaluation of accuracy metric. IoU is defined by the following equation

$$IoU = \frac{A \cap B}{A \cup B}$$

Where A is the ground truth box and B is the predicted bounding box. This is equivalent to the area of interest divided by the total area of the two boxes. In our evaluation, For every predicted face box, the IoU was calculated for every ground truth and the max was taken as the score for that predicated box. This was done for every image in the WIDER validation dataset and the average compute time and IoU were recorded. These results can be seen in Table 3. Despite *YOLO* preforming better in IOU, it was decided that *Faced* would be used as it evaluates input images significantly faster, allowing for real-time detection and emotional classification.

3.2. Classifying Emotions

One metric that we inspect during the epochs is the top k categorical accuracy rates which shows how successful the model is when the target is in the top k predicted classes. For our models we set a default of $k = 3$.

After using Grid-Search on the hyperparameters, we determined the top eight models in regards to test accuracy and speed of prediction. These results are shown in Table 1. Of these models, the two best performing include one CNN-SVM and one pure CNN. The highest performing CNN-SVM was the fastest model, at an average of only 0.0651 seconds for a batch size of twenty. The best pure CNN was the most accurate model, with a test accuracy of 56.67%.

4. Discussion

4.1. Dataset Issues

Though the FacesDB dataset does fit the unique needs of our project (i.e. a host of different people with directly labeled emotionally classified expressions collected from a frontal perspective), that is not to say it is without some drawbacks. Every machine learning model benefits from more training

Model	Layers	Drop	L.R.	Reg	Batch	B=1 Time	B=5 Time	B=10 Time	B=20 Time	accuracy	top3
CNN-SVM	3	0.2	0.001	0.1	30	0.0255	0.0323	0.0427	0.0651	0.5333	1.0
CNN-SVM	3	0.4	0.001	0.01	30	0.0261	0.0578	0.1054	0.1913	0.5333	1.0
CNN-SVM	3	0.5	0.001	0.01	60	0.0362	0.0968	0.1707	0.3203	0.4667	1.0
CNN-SVM	3	0.3	0.001	0.1	60	0.0419	0.1193	0.2252	0.4255	0.3667	1.0
CNN	2	0.2	0.001	NA	60	0.0518	0.1369	0.2471	0.4702	0.4	1.0
CNN	2	0.4	0.001	NA	60	0.0530	0.1441	0.2593	0.4916	0.4167	1.0
CNN	2	0.3	0.001	NA	60	0.0541	0.1469	0.2628	0.4871	0.5667	1.0
CNN	2	0.5	0.001	NA	60	0.0516	0.1437	0.2628	0.4972	0.4	1.0

Table 2. Grid-Search Evaluation
Statistics for the four most successful models for each architecture (considering speed and accuracy) are given below. After the model type, the five hyperparameters we adjusted are given in each column: number of convolutional layers (for CNN-SVM) or layer groups (for CNN), drop rate, learning rate, regularizer, and batch size. In the next four columns, average timing statistics are given for predictions on batch sizes of one, five, ten, and twenty respectively. The final two columns give the accuracy over a subset of data not trained on, along with the top-three accuracy for each model.

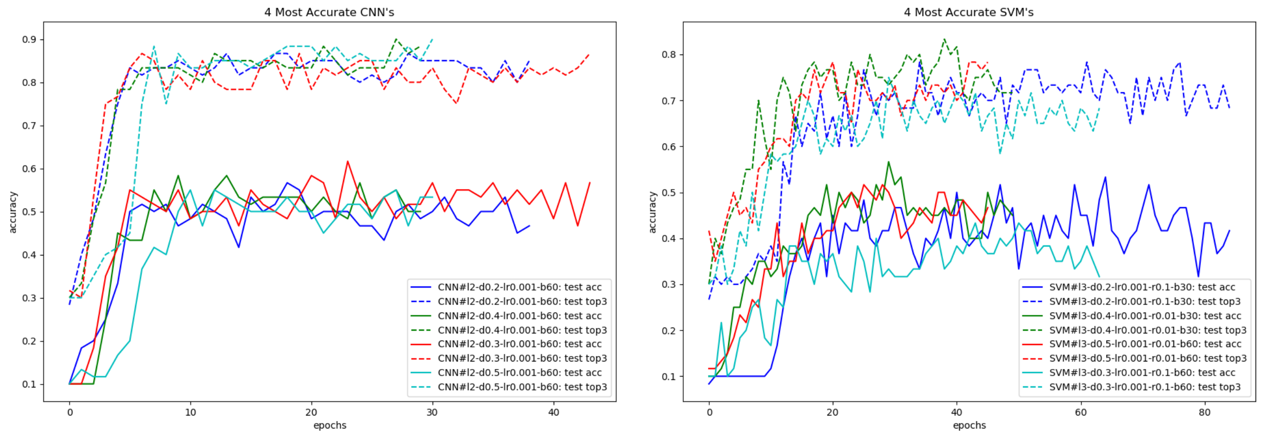

Figure 1. Training accuracies for top CNN and SVM Models

Table 3. Performance of Object Localization Models

Algorithm	Avg IOU	Avg Evaluation Time
Haar	0.1843	0.1516
YOLO	**0.2650**	2.8478
FACED	0.2408	**0.0587**

data; relatively speaking, the FacesDB dataset is fairly small. It provides 10 emotions for 36 unique faces or a total of 360 images for both training and testing. As a means of combating the disadvantages of a small dataset we manufactured new samples by applying image augmentations 4 separate times after the original formatting. (4) The image augmentations allowed us to increase our samples size from the original split of 280 images to 1400 images which we felt more suitably met our needs.

A second issue with the FacesDB dataset is that there is a lack of situational diversity. There are no images of people in glasses, there are no out of focus images, no poorly cropped images, it is not completely ethnically diverse, it is slightly generationally diverse, etc. Ideally a dataset for image recognition would be expansive and diverse; it would include samples representative of all the potential live data it might be tested on. But the FacesDB dataset is not that. It is in fact perfectly captured, focused, and cropped representation of a handful of emotionally labeled images that are not directly representative of those that our image segmentation software is designed to track.

It is also important to note that the FacesDB dataset is not a dataset of organically expressed emotions. While it is true that perfectly labeled natural expressions of emotion would certainly be preferable to staged, the collection of such a dataset would not be without drawbacks. An acted instance of anger may be acted, but without a doubt it is accurately labeled. It is also important to note that most people are capable of producing a worthwhile facsimile of most emotions on command. On the other hand organically expressed emotions are either labeled after the experience or by a third party and both of these solutions inject ambiguity into any conclusion.

Furthermore, the FacesDB dataset crucially does not include an expression such as ”boredom” or ”apathy”. As our system aims to classify attentiveness the lack of directly related samples did create difficulties. Our solution was to define apathy from a middle ground of classifications. As the samples in the FacesDB did not come from organic expressions of the relative emotions, we feel that our middle ground definition of apathy holds more water as it is defined by staged portrayals but not directly. This leaves enough ambiguity we feel to perhaps approach closer to an organic definition of apathy.

4.2. Object Localization

The results of our object localization model accuracy show that utilizing deep learning frameworks like a Convolutional Neural Network yields better results. Comparatively, in the *Haar* model we can see that even though it is faster than *YOLO*, it does worse in terms of actual performance. This is probably due to the fact that, as stated before, the use of pre-determined cascade filters will not generalize well for datasets with alternative features apparent on a human face. *YOLO* on the other hand is the most accurate out of the models we tested due to it using the full power of deep learning to extract feature spaces that might not be apparent to more classical detection techniques such as *Haar*.

However, *YOLO's* massive advantage also results in a great disadvantage when running it in real-time. Using the pre-trained model we found, the weights associated were too large to even store on the Github repository. The overall complexity of *YOLO* is not transferable as an easy to use strategy to create a fast and efficient pipeline like the one we are trying to achieve. It also fails to run in real-time without a GPU. Having a dedicated GPU to achieve real-time performance is not viable as we have learned time and time again throughout the extent of this project. Furthermore, OpenCV GPU capabilities are only compatible with Intel graphics cards, which have extremely poor performance compared to higher end dedicated graphics cards like NVIDIA and AMD. The Intel graphics cards at our disposal provided no noticeable improvement in object localization speed.

This is why for the current iteration of our pipeline we chose to go with *faced*. The *YOLO* neural network is huge since it tries to learn a lot of feature spaces in order to be a generalized model for object localization. *YOLO* is designed for general object detection, classifying a variety of objects more in depth than what we need. In terms of our current objective we do not need a large amount of parameters, layers, or filters since we are exclusively focused on the localization of faces. Therefore our model does not need to detect much features for such a specific task. This is where *faced* shines, since it is based on *YOLO's* architecture but requires fewer parameters since it is specialized for face localization. This leads to a large decrease in complexity and latency while giving only a slight drop in accuracy. Therefore it is able to run much faster that *YOLO* and is able to run in real-time on a CPU. It is not as accurate as *YOLO*, probably due to the decreased complexity of the network, but the difference is negligible and for our pipeline to be fast and continuously iterable it is obvious that *faced* is our top choice.

4.3. Emotion Classification

Our Grid-Search tuned five hyperparameters for CNN-SVM and four for the pure CNN. Across all models, a small learning rate of 0.001 provided universally optimal accuracy. Larger batch sizes also tended to outperform smaller batch sizes, with 75% of the top eight models using the largest batch size tested - 60. The remaining two models used the medium batch size option of 30. These tendencies demonstrate that a smaller learning rate and larger batch size keep the model from the wild fluctuations that were detrimental to less successful architectures. More samples will be tested before model weights are adjusted, and these adjustments will be less severe, resulting in a smoother climb in accuracy. For models with more than the optimal number of layers (3 layers for the CNN-SVM and 2 layer groups for the pure CNN), they struggled to converge in a reasonable amount of time and their ability to generalize suffered in most cases. While dropout rate was not a significant indicator of final test accuracy in either architecture, CNN-SVM models with a higher dropout rates of 0.4 and 0.5 seemed to converge more quickly than those with smaller rates of 0.2 and 0.3. The dropout rate hyperparameter had an inverse effect on the pure CNN models, most likely because of their larger size. Models across the entire range of dropout rates were selected for the top eight best performance. The CNN-SVM Grid-Search selected for smaller regularizer hyperparameters, choosing only values of 0.1 and 0.01 as the most accurate models.

Once we determined our top eight best performing models, we narrowed down to two by selecting the most accurate CNN-SVM and pure CNN. In order to maximize performance for the FACES system, we considered both speed of classification and the test accuracy. Each of the two top models had drawbacks and benefits; while the CNN-SVM was not as accurate with a 53.33% test accuracy, it far surpassed the other models in speed, especially considering larger batch sizes. This made it a great candidate for a live video feed. In contrast, the most successful pure CNN

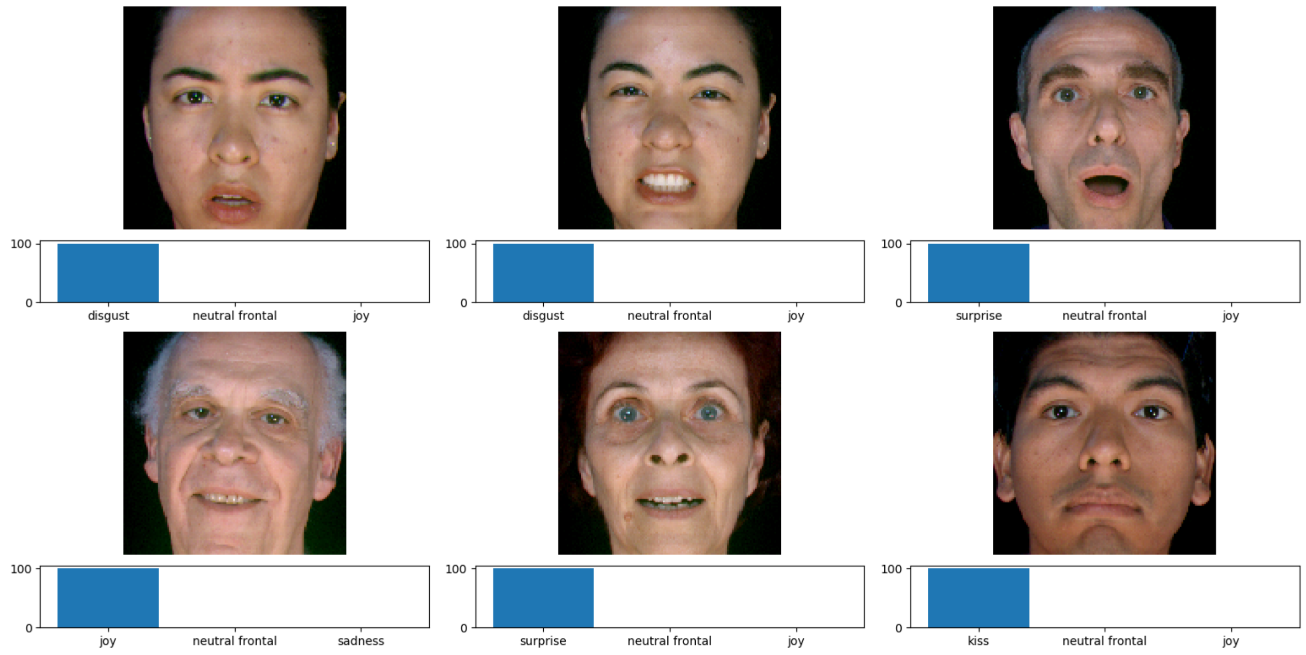

Figure 2. Best CNN Prediction Samples

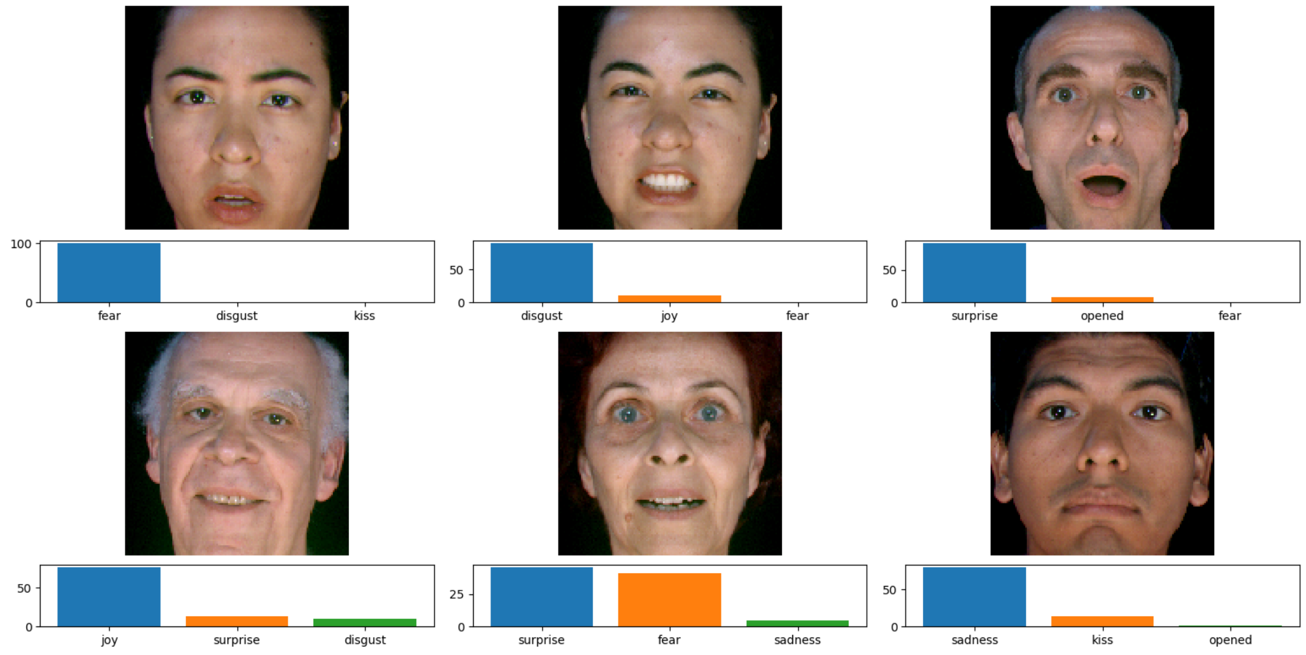

Figure 3. Best SVM Prediction Samples

ranked first in accuracy with a 56.67%, but lagged behind in prediction speed - taking up to eight times as long to predict a batch size of twenty.

The differing prediction speeds of these models are not surprising, considering their structure. The slower pure CNN model has more layers and nodes, resulting in a huge set of 1,037,642 trainable model parameters. The much faster CNN-SVM is more than $\frac{1}{32}$nd the size of the pure CNN, will only 32,850 trainable parameters. The smaller CNN-SVM therefore out-competes the pure CNN models in terms of prediction speed. Since the CNN-SVM was nearly eight times as fast and only 3 - 4% less accurate than the leading pure CNN model, we chose the CNN-SVM architecture to run on our FACES system.

When considering our highest test accuracy rates of 53.33% for the top CNN-SVM and 56.67% for the pure CNN, it is important to note that these values represent very high accuracy. Emotional classification is qualitative, not quantitative; while a human could make guesses as to the emotions expressed by another based on positioning of the mouth and eyes among other factors, there is no ”right” answer. A face can display several emotions at once, or a mysterious expression that cannot be easily grouped into any class. For this reason, displaying sample results like those in Figures 4.2 and 4.2 helped us to determine that the top models were more successful than demonstrated by the accuracy rates. For example, the first sample has a somewhat ambiguous classification. While it is labeled as ’disgust’, the SVM’s classification of ’fear’ is also reasonable to a human eye. While this is not the ”correct” choice and therefore affects the accuracy of the model, it performs quite well in a real world scenario such as our FACES live demo.

4.4. Application: Live Video Processing

After performing research on image segmentation and emotion classification models, we combined them to create a live video processing technology. Given a video, the application is designed to continuously detect all faces in the video, classify each of them as attentive or inattentive based on its emotion, and display the results simply and clearly to the user. Our application was specifically designed for the use case of tracking audience attentiveness in a classroom setting in order to provide nearly real-time feedback to the teacher. The following emotion classes were considered indicators of attentiveness: joy, surprise, opened, and kiss. The following emotion classes were considered indicators of inattentiveness: neutral frontal, sadness, anger, disgust, fear, and closed.

To ensure optimal performance, we tested video processing with every combination of image segmentation model and emotion classification model. Each combination was tested for one minute with open and closed facial expressions, with image segmentation and emotion classification updating every 2.5 seconds. The Classification Accuracy Rate reflects the proportion of facial expressions that were correctly classified as attentive or inattentive by the application. The Face Detection Rate denotes the proportion of computational cycles where the image segmentation model correctly located the face in the video. The Average Segmentation Time and Average Classification Time reflect the average time per computation cycle that the application required to segment and classify the image, respectively. The results of this analysis are summarized in Table 4.

Out of the three segmentation models, we noticed significant differences in computation time, with *faced* being the fastest and *YOLO* being the slowest. The differences in computation speed are significant, as *Haar* takes roughly three times the amount of time required by *faced* and *YOLO* takes roughly twenty times the amount of time required by *faced*. The faster segmentation models suffered only minor loss of face detection accuracy (temporarily losing facial detection, false positive detections, false negative detections with videos containing multiple faces, etc.). Overall, *YOLO* performed image segmentation with the most dependably, as it was able to detect the face in the video in 100% of the computation cycles in each of the four trials.

The results of the model combination analysis suggest that the choice for an image segmentation model has some effect on an emotion classifier’s computation time. Each classification model experiences its longest average computation time when paired with the *faced* segmentation model. The data also suggests that the image segmentation model has an effect on the accuracy of an emotion classifier; in particular, CNN2 is significantly sensitive to which segmentation model it is paired with. CNN2 and Generalized SVM predict most accurately given images processed by the faces model, while Fast SVM predicts most accurately given images processed by the *Haar* model. It is possible that this occurs because the *faced* and *Haar* models trim faces more tightly than the *YOLO* model, which may allow the classifier to predict solely based on facial features instead of extraneous features such as hairstyle or background.

When designing our application, one key consideration was the trade-off between processing and displaying video frames in as close to real-time as possible, and detecting and classifying faces with as much accuracy in possible. While application models using *YOLO* yielded slightly less accurate predictions than those using *faced* and *Haar*, we felt that dependable image segmentation was crucial for our application’s use case (in a classroom with clustered faces, non-ideal lighting, etc.). For our final application, we chose to use *YOLO* image segmentation with Generalized SVM emotion classification, since Generalized SVM yielded a Classification Accuracy Rate of 0.5 or above regardless of

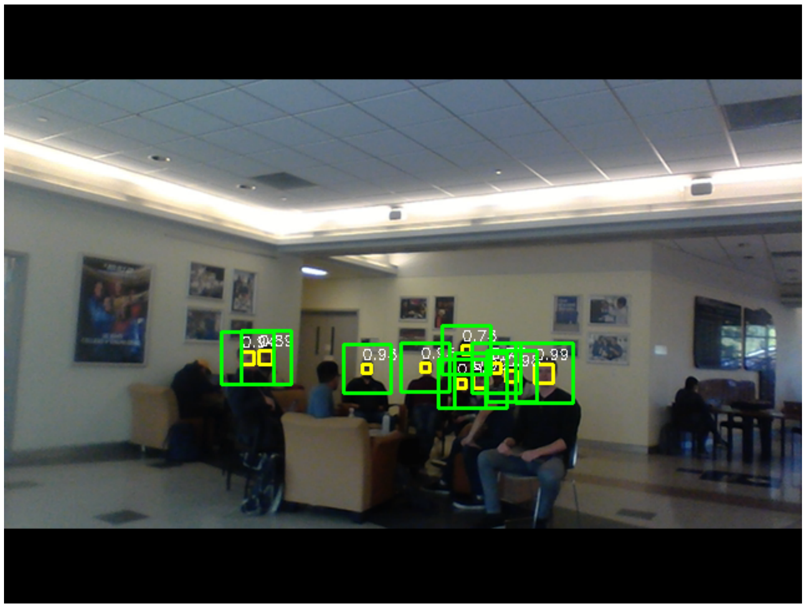

Figure 4. Live Demo Example

	YOLO	faced	Haar
CNN1	0.50, 1.0, 2.68, 0.109	0.478, 0.96, 0.339, 0.122	0.50, 0.962, 0.0767, 0.0902
CNN2	0.476, 1.0, 2.75, 0.123	0.667, 1.0, 0.381, 0.138	0.50, 1.0, 0.0753, 0.0913
Gen. SVM	0.50, 1.0, 2.76, 0.105	0.5652, 1.0, 0.3801, 0.1308	0.50, 1.0, 0.0729, 0.0787
Fast SVM	0.4545, 1.0, 2.796, 0.1083	0.5238, 0.84, 0.3738, 0.1254	0.542, 1.0, 0.0771, 0.0777

Table 4. Model Combination Metrics
(Classification Accuracy Rate, Face Detection Rate, Average Segmentation Time, Average Classification Time)

the segmentation model it was paired with. With these models, we were able to display a video feed with insignificant latency while updating the facial boundary boxes every 2.5 seconds. We are confident that increased computing power would improve our application, as it would allow for more frequent updates of the facial boundaries.

If more time were permitted, we would perform a similar model combination analysis using videos with multiple faces and variable lighting conditions to better simulate classroom conditions. The project opens a door to future research regarding the effects that different models in series have on each other's performance.

5. Conclusion

The purpose of this project was to leverage and implement real-time image segmentation and facial recognition to assess engagement in audience settings. This report analyzes the different methods used and provides an insight into how our system can be improved in future iterations. Our current system pipeline consists of first detecting a face in an image frame and drawing a bounding box around it. Each of the detected faces are then run through either a Convolutional Neural Network (CNN) or Support Vector Machine (SVM), that have both been trained on an image dataset of common emotional expressions, to classify them into a specific emotion class. Through rigorous testing, we have found that our most successful classification model is capable of accurately classifying facial emotions in about a tenth of a second.

Given the constraints and scope of this project, there are further changes to be made that can improve both the accuracy and performance of our system. As discussed previously, the dataset used to train our models, FacesDB, does not provide samples explicitly labeled as showing a certain degree of attentiveness. A dataset that is more tailored towards our specific regression problem of measuring attentiveness would increase accuracy as it would not require extrapolation on our part from the detected emotions. Our emotion classifiers could also benefit from a larger dataset, as 360 images can provide only a small training set. Given more time, we could also continue to run Grid-Search for more fine-tuning of our CNN and CNN-SVM hyperparameters. This could result in significantly greater accuracy. Additionally, the computational power required to run our models proved to be a significant bottleneck on the performance of our system. With the use of a GPU, we would be able to greatly increase the speed and efficiency of our system. The execution of these improvements would yield an accurate and well-performing system that would serve its purpose of providing lecturers with a tool to measure their effectiveness through measurements of their audience's engagement. Although our system provides a solution to a relatively niche problem, the pipeline and methods used to build our system can be abstracted to solve similar problems that require high prediction speed as well as accuracy.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[2] B. Ko, "A brief review of facial emotion recognition based on visual information," in *Sensors*, 2018.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[4] S. Li and W. Deng, "Deep facial expression recognition: A survey," *ArXiv*, vol. abs/1804.08348, 2018.

[5] M. J. Paul Viola, "Rapid object detection using a boosted cascade of simple features," *CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 2001.

[6] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.

[7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[9] O. Arriaga, M. Valdenegro-Toro, and P. Plöger, "Real-time convolutional neural networks for emotion and gender classification," *ArXiv*, vol. abs/1710.07557, 2017.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, p. 273–297, 1995.

[11] Y. Tang, "Deep learning using linear support vector machines," 2013.

[12] A. F. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," *ArXiv*, vol. abs/1712.03541, 2017.

[13] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Author Contributions

- **Aliaksei Chumakou**: Designed and tested performance for live video processing to maximize the usability and the visual information of the segmentation and classification models.
- **Shivam Desai**: Developed initial design for live video demo application and ability to test on pre-recorded videos.
- **Rohit Dhamankar**: Integrated *faced* model for object localization. Modified object localizer to classify emotions for multiple detected faces per frame. Fixed drawn rectangles to accurately frame detected faces.
- **Luis El Srouj**: Developed the architecture and performed training of the CNN-SVM model; designed module to load in saved models and return most confident predictions; designed a data pipeline that applies random transformations and bootstrapping to training data; divided tasks into manageable subtasks.
- **Rachel Hanna**: Performed the Grid-Search for the CNN-SVM and pure CNN models; Designed module to evaluate speed and accuracy and display result figures of the best performing emotional classification architectures.
- **Brendan Kelley**: Duties included but not limited to data pre-processing and sample manipulation methods.
- **Munir Nur**: Developed architecture and training of CNN2 and CNN models. Helped design data pipeline to optimally train and evaluate models, and helped produce supplemental figures for preliminary model evaluation.
- **Huyen Pham**: Performed data preprocessing to expand the dataset, including add speckle to an image, turning image into grayscale and adding gaussian noise to the faces in the images.
- **Ravishdeep Singh**: Organized and coordinated discussion within team. Set up meetings and goals for sprints of the project. Helped create the base class for object localization. Fully integrated and tested the *YOLO* model for object localization.
- **Jacob Smith**: Designed live video processing application and performed analysis of segmentation and classification model combinations.
- **Chantheary Soth**: Performed data preprocessing to expand dataset with up and downsampling.
- **Ryan Stager**: Developed the base class for object localization. Implemented the haar classifier and assisted in the implementation of both YOLO and Faced. Wrote the evaluation script for the same object localization.