

[blog.affectiva.com](https://blog.affectiva.com)

---

# Finding the Face: Facial Detection Process for Deep Learning

*Ashley McManus*

10-12 minutes

---

Deep learning is the hottest area of research right now - and for good reason. At Affectiva, we are experimenting with this new machine learning technique to further enhance the capabilities of our emotion recognition technology.

We spoke with our own Mohamed Ezz, our Machine Learning Scientist who works mainly with deep learning methods for Affectiva's vision tasks. He was able to share with us some details on what he is working on, specifically on what it takes for algorithms to detect faces.

## Why face detection?

The concept of Affectiva was built on sensing an individual's emotion (you can learn more here about [how we label faces for emotion](#)), and one of the best sources of human emotion is in our facial expressions. For our [Emotion SDK](#), if we fail to detect the face, we can't detect the emotion that it's portraying

- so face detection is absolutely the very 1st thing we do, and it's really important to be accurate in this task. We have built our own face detector, and are always looking for ways to make it more accurate - this includes accounting for face sizes, the number of faces in a frame, variations in lighting, face orientations, or items in addition to the face like headscarves or glasses - not all face detectors can detect variations like these.

### **What are the different approaches to deep learning that we have tried?**

From a technical perspective, so far we experimented with Face detection using the following deep learning approaches:

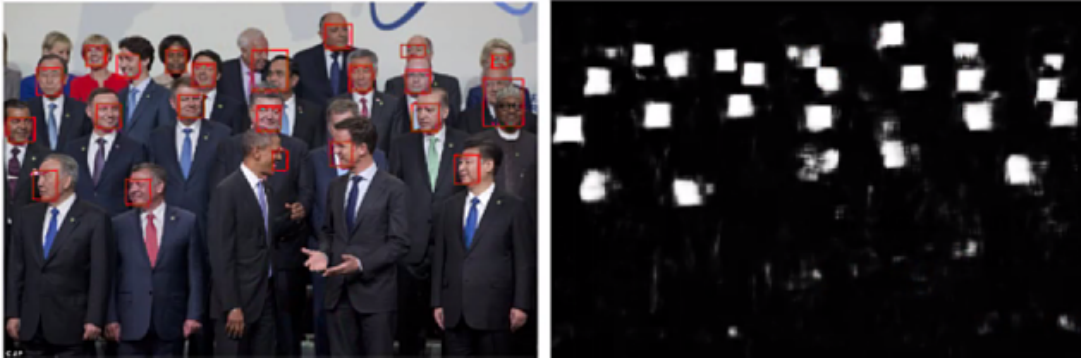
1. Face detection using a segmentation based approach
2. Face detection using Region Proposal Networks (RPN)

For both approaches, we observed clear improvements of deep learning over our previous algorithms, in which we used traditional computer vision methods to extract features and machine learning methods for building our models.

### **Which show more promise than others?**

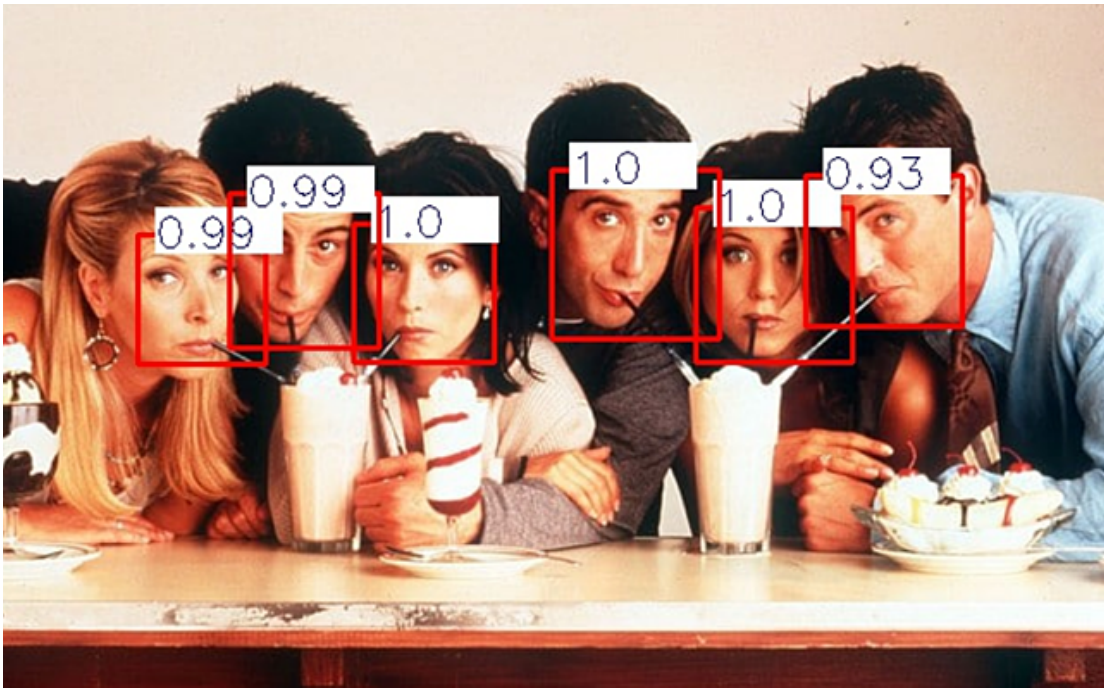
Face detection based on Region Proposal Networks (object detection approach) has shown more promise than a face segmentation approach. Face segmentation works by examining each pixel in the image and deciding whether it's

part of a face. The face segmentation approach involves hand-crafted post-processing - meaning we decided how to do it, it was not learned - to identify regions (rather than pixels) of faces. In other words, we did not ask the neural network to explicitly tell the location and size of the face region found. This approach didn't prove to be as good.

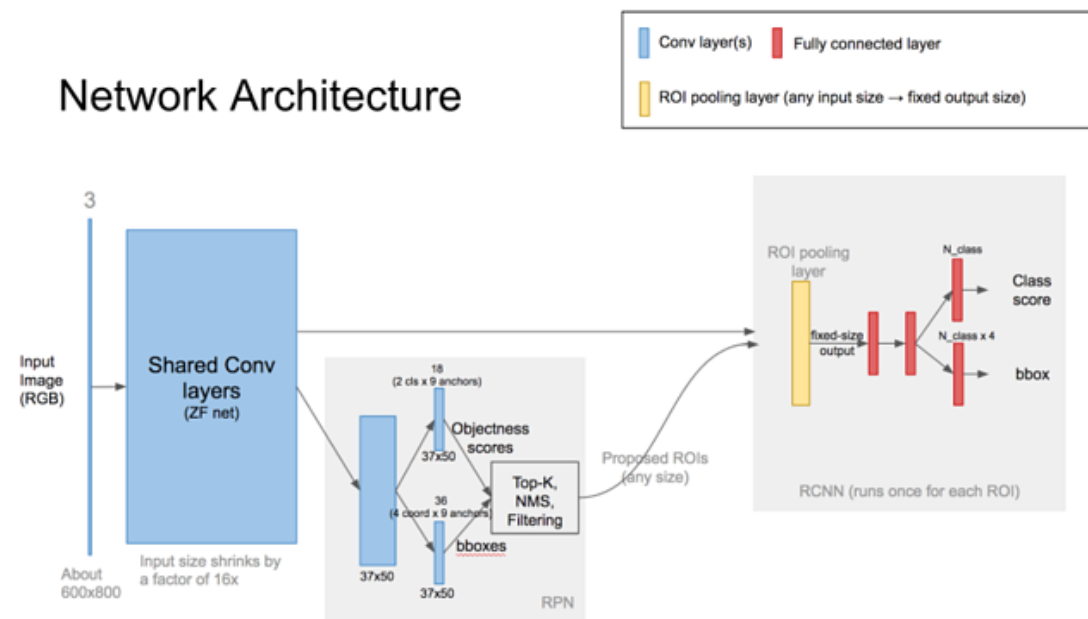


### *Face Segmentation Approach Results*

Using Region Proposal Networks, we explicitly trained the network to locate the exact region of the face. We ask the model to locate the face based on examining many - overlapping - regions in the image, then by combining the answers, we are able to localize faces in a more established way.



*Region Proposal Results*



*Region Proposal Network Architecture*

**What are the challenges?**

I'll share two:

1. Deep learning demands more data, which isn't directly available for some tasks. So we had to look for artificial ways of generating data.
2. Deep learning is computationally very expensive, so deploying these models on mobile devices is challenging. We need to make smaller models without sacrificing accuracy.

A more practical challenge we faced was preparing the infrastructure for being able to work with very large sets of data. The way it always used to work was that we took an image dataset, pre-process all of it in a certain way, then do multiple experiments on those pre-processed files. The problem was that if we want to try a different way of data pre-processing, we would have to re-run a pre-processing job for the whole dataset, which is very time consuming. This hindered experimentation, since researchers are always hesitant to do a time consuming change, if the expected improvement may not be large. As a general rule, researchers tend to experiment more when it is easy to experiment - and time consuming tasks are not fun, so this tends to be limiting. We built an infrastructure that doesn't require batch processing of data - rather, it conducts the pre-processing concurrently during training. This means that if we now want to try to pre-process the data in a new way, we just need to make a code change then re-train. This is possible because our system processes data on the fly, which doesn't run

through all of the data ahead of training. Pre-processing images one at a time helped in many situations at important times.

This not only saves us time of experimentation, but also enables online data augmentation. Data augmentation is a technique to expand the available training data by applying image transformations (flipping, rotations, cropping..etc). This, when done on the fly, can make a dataset virtually infinite - since there are infinite ways we can do rotations for example. This way it's much harder for the model to overfit.

### **How much data do we train with?**

It depends on the task. For face detection we use about 80,000 images, with well over 100,000 faces, for training our models. And we have another 20,000 images for evaluating our models.

### **What sort of accuracy? Any tradeoffs to this?**

We measure *Precision* and *Recall*. Because our task is finding face bounding boxes, we must associate predicted bounding boxes with the ground truth ones. If a predicted bounding box sufficiently overlaps with ground truth, it's considered a correct prediction. For *Recall*, we examine the faces that we know are present, then look at what percent we were able to detect. And while *Recall* may be high, we could still get a good deal of of

false positives. Therefore we measure Precision as well - out of all the faces we predicted, how many were correct? There is a fundamental tradeoff between Precision and Recall.

One challenging case is detecting small faces, when we drive the model to high recall to get the small faces, we tend to get a lot more false positives, because high frequency textures (which are not faces) can easily be confused for a small face.

In order to track the performance on these different cases, we divide the faces in our evaluation set into multiple subsets, and use each subset as a product use case. We evaluate the model for each subset and - among other measures - report Precision and Recall. For example on the standard-face subset, we have about 98% Recall. Subsets of other instances that we are trying to account for are dark faces, posed faces, blurry faces, faces with glasses and occluded faces. Then based on the performance of our model on these data subsets, we are able to decide which models to deploy to Affdex for Market Research, where specific requests come in on a model that is most accurate for head-on faces, for example.

### **How does it compare to other Machine Learning (ML) approaches?**

Other ML approaches for face detection typically extract hand-crafted features out of the image. Traditionally, research was concentrated on how to design the best image features for

certain tasks. After feature extraction, one would train some shallow ML model to decide for each patch in the image whether it is a face or not. This family of approaches is slow because it requires many model evaluations per image, and also not accurate because the features were hand-crafted, not learned.

Deep learning models not only learn the given classification task, but also learn the best way to extract features for the given task. Our model takes the full image, and using Convolutional layers, it runs a sliding window feature extractor to measure responses for a certain feature in different locations of the image. Using multiple such feature extractors per Convolutional layer, and multiple consecutive Convolutional layers, we reach a decision in the last layer on which parts of the image contain a face. But again, a downside of deep learning models is that they often require a large number of computations.

### **Where do we see it going?**

So far we've been approaching the emotion recognition problem by explicitly dividing our system into different components and training a deep learning model for each one. Face detection is the first component to localize faces, after that comes facial landmark tracking, then AU classification and finally emotion classification. We train each component, individually, with a certain network architecture. Note that we



not only forced the system to be divided into 4 parts, but also defined how large each part should be.

There are two drawbacks to this approach:

1. We relied on our intuition rather than on data, for decomposing system.
2. Each model extracts image/face features on its own. This is redundant and makes our overall runtime slower.

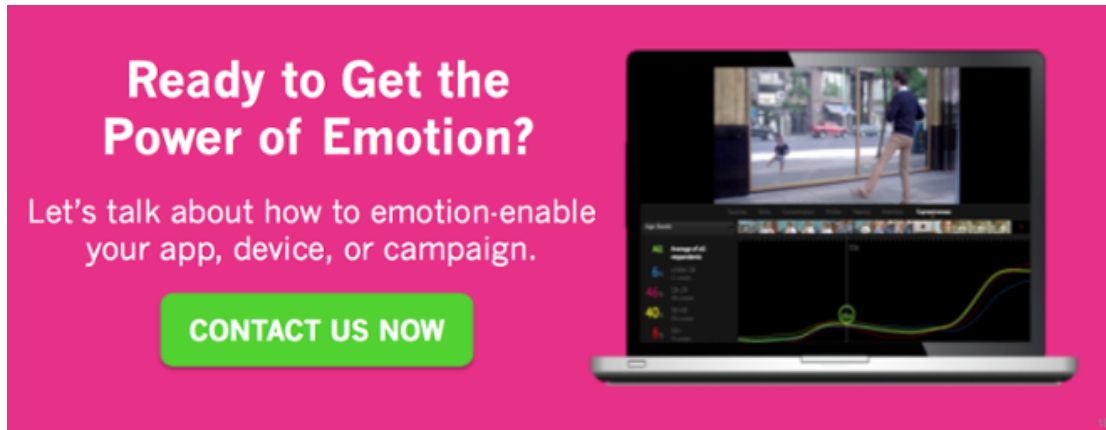
I would like us to move from this approach to another where teach the system to take a full image as input and directly give as output the emotions of the faces present. In technical terms, we want to train an end-to-end deep learning model. The main blocker is large amounts of data for this type of input and output.

A more practical way we are looking into, is to design a network architecture that combines all the system components but train sub-parts individually. In this approach we benefit from a common set of feature extractors, removing redundancy.

### **What excites you the most about this?**

While most deep learning based products are served from powerful servers in the cloud, our goal is to deploy an accurate deep learning model that scans a high resolution image offline, on a low-end device. This is difficult. We are among a few companies tackling this challenge. We not only

want to build a product that works, we want to also make sure it works very well, in various real-life scenarios, without draining a device's battery. Building such an intelligent, yet efficient system, is very exciting.



**Ready to Get the Power of Emotion?**

Let's talk about how to emotion-enable your app, device, or campaign.

**CONTACT US NOW**

[Deep Learning](#) [Artificial Intelligence](#) [Science](#)