**Ear Recognition in unconstrained environment**

***Abstract***: In this project, we have investigated unconstrained ear recognition in great detail and the associated problems that come with it. The ability to capture ear images from a distance and in a covert manner makes the technology an appealing choice for surveillance and security applications. Automatic identity recognition from ear images represents an active field of research within the biometric community. We have considered several deep network architectures to both segment and recognize the person. Our aim to create a pipeline to effectively segment the ear from an image or surveillance feed and to identify that person.

By:

Shreyas Gokhale

Mihir Pathak

Kshitij Walekar

Akash Kulkarni

Ujwal Mondhe

*Introduction:*

Human identification through biometrics has been both an important and popular research field. Among the biometric traits, ear is a unique part of the human body in terms of different features such as shape, appearance, posture, and there is usually not much change in the ear structure except that the ear length is prolonged over time. Various studies have been conducted and many different approaches have been proposed on ear recognition, however, it still remains as an open challenge, especially when the ear images are collected under uncontrolled conditions as in the Unconstrained Ear Recognition Challenge (UERC). Ear recognition approaches are mainly categorized into four groups, holistic, local, geometric, and hybrid processing. In the earlier studies, the most popular feature extraction methods for ear recognition were SIFT, SURF, and LBP. Due to the popularity of deep learning in recent years and its significant impact on the computer vision field, deep convolutional neural networks (CNN) based approaches have also been adopted for ear recognition. CNNs mainly require a large amount of data for training. However, the amount of samples in the datasets available for ear recognition are rather limited. Due to this limitation, we utilize an already trained object classification model, so called a pre trained deep CNN model, from some of the well-known high performing CNN architectures like the VGG16 or GoogleNet.

*Why Ear?*

Ear images used in automatic ear recognition systems can typically be extracted from profile head shots or video footage. The acquisition procedure is contactless and nonintrusive and also does not depend on the cooperativeness of the person one is trying to recognize. Another appealing property of ear biometrics is its distinctiveness. Recent studies even empirically validated existing conjectures that certain features of the ear are distinct for identical twins. This fact has significant implications for security related applications and puts ear images on par with epigenetic biometric modalities, such as the iris. In surveillance applications, for example, where face recognition technology may struggle with profile faces, the ear can serve as a source of information on the identity of people in the surveillance footage.

*Datasets:*

* ***For ear recognition:***

1. **AWE Dataset:** In the ear recognition field, most of the datasets have been collected under controlled conditions, such as in a laboratory environment. However, these images are taken in uncontrolled environment and hence are useful for practical ear recognition. This dataset consists of 1000 images in total of 100 subjects with each subject having 10 images from different viewing angles.
2. **UERC (Un-constrained Ear Recognition Challenge) Dataset:**

The UERC dataset is another dataset which is collected in the wild. It consists of ear images collected from web of varying quality and sizes. The UERC dataset is divided into two parts as training and testing sets. In total, there are 11804 ear images of 3706 subjects. Training part of the UERC dataset contains 2304 images of 166 subjects and testing part has 9500 images of 3540 subjects. UERC training set has a more even distribution and contains more ear images with better resolutions, i.e. having more than ten thousand pixels. The training part of the UERC dataset is created by combining the AWE (1000 images), CVLED (804 images) datasets, and 500 extra images that have been collected from the web.

* ***For ear segmentation:***

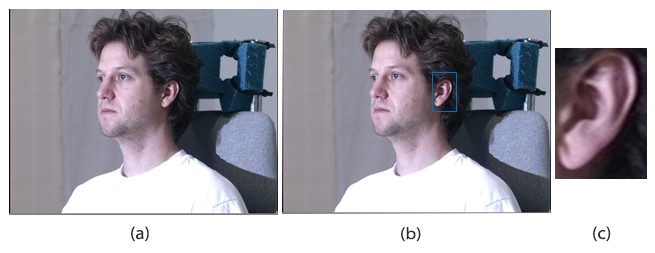
1. **Multi-PIE Face Dataset:** Multi-PIE face dataset contains 337 subjects, whose images are acquired, as the name implies, under different pose, illumination, and expression conditions. We utilize this dataset to train our R-CNN model to segment ear from face images and further use these segmented ear images in identification.

1. Sample images from AWE and UERC dataset



1. selected view angles from multi-pie face dataset



c. illustration of ear detection and cropping from multi-pie face dataset

***Methodology:***

In this section, we present the studied deep learning convolutional neural network architectures which have proven to give results on various applications in the past and some transfer learning approaches. We will also employ data augmentation to maximize the accuracy of our model to identify a person.

Convolutional Neural Networks:

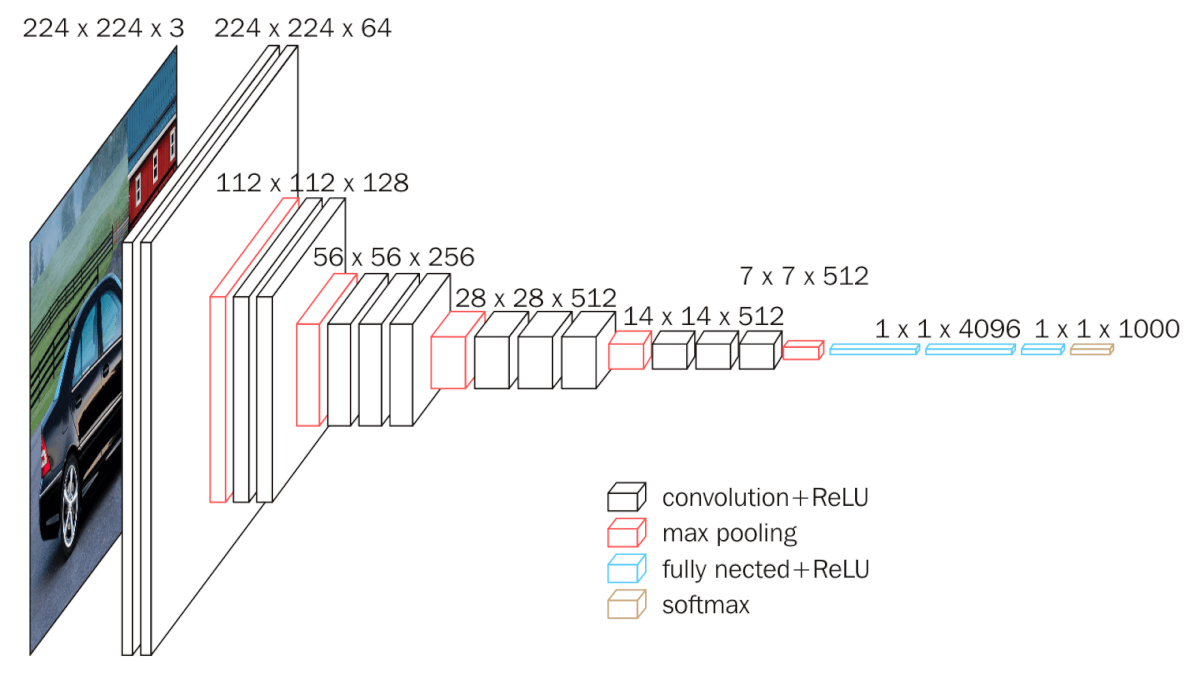
In our approach, we will employ convolutional neural networks for ear representation and classification. CNN contains several layers that perform convolution, feature representation, and classification. Convolutional part of the CNNs includes layers that perform many operations, such as convolution, pooling, batch normalization, and these layers are sequentially placed to learn the discriminative features from the image. Then, in the later layers, these features are utilized for classification. For the final layer, we use the softmax layer to identify the proper identity class.

Deep CNN architecture (AlexNet or VGG-16)

Fully connected layers

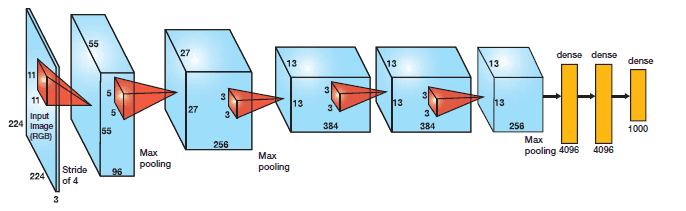
Softmax layer

The first deep convolutional neural network architecture studied is AlexNet. In AlexNet, there are five convolutional layers and three fully connected layers. Dropout method has been used to prevent overfitting. Besides, we have also considered employing VGG16 architecture. VGG16 has 16 alternate convolutional and max pooling layers.



VGG 16 architecture

* The main characteristic of this network is the use of several consecutive convolutional layers with a small 3\*3 kernels.
* These stack of 3\*3 convolutional layers are able to capture same information as larger filter used in Alex-Net but requires significantly less parameters during training.
* The convolutional part of VGG16 is followed by 3 fully connected layers with 4096, 4096 and 1000 channels respectively.
* Finally, a softmax layer is used at the top of the model



Alex-net architecture

* It contains 5 convolutional layers and 3 fully connected layers.
* ReLu is applied after every convolutional and fully connected layer.
* Dropout is applied after the first and second fully connected layer.
* The network has 62.3 million parameters and needs 1.1 billion computational units in a forward pass.

*Transfer Learning and Data Augmentation:*

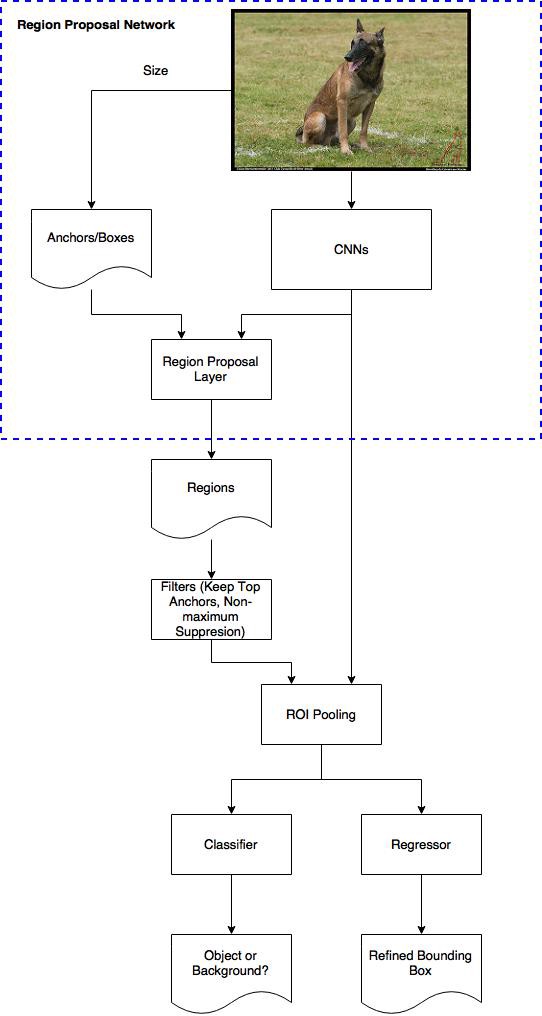
By using the pre-trained models of AlexNet and VGG-16 architectures, which were trained on the ImageNet dataset, we will fine-tune them on the ear datasets. The ear recognition datasets contain a limited amount of training samples, for example the ones used in this study contain around a thousand to ten thousand ear images. This amount of training data is sufficient for fine-tuning, although it would not be enough to train a deep CNN model from scratch. Transferring a pre-trained deep CNN model can provide better classification performance than training a task specific CNN model from scratch, when only a limited amount of data is available for the task at hand, as in the case for ear recognition. Further transferring a CNN model from a closer domain, that is for age and gender classification transferring a pre-trained model that were trained on face images, instead of one trained on generic object images, provides better performance. While performing fine-tuning, parameters have been initialized with the values that came from the pre-trained network models. This is a commonly used strategy in fine-tuning, since the early layers mainly focus on low-level feature extraction and the later layers are mainly responsible for classification.

Since the number of images in the UERC dataset is limited, in order to increase the amount of data as well as to account for appearance variations due to image transformations, we have applied data augmentation. Data augmentation has also been applied to the Multi-PIE ear dataset. Although the Multi-PIE ear dataset contains around eight times more images than the UERC dataset, it would still benefit from data augmentation. Data augmentation can be performed by using the Imgaug tool.



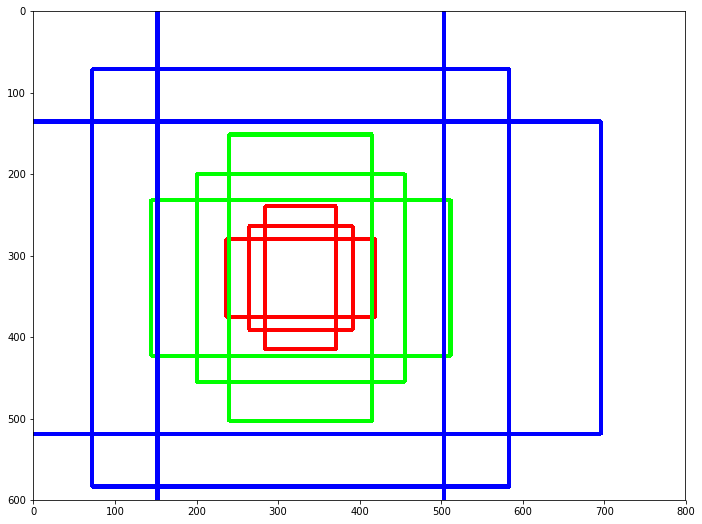
Faster R-CNN

Faster R-CNN has two networks: region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects.



Anchors

Anchors play an important role in Faster R-CNN. An anchor is a box. In the default configuration of Faster R-CNN, there are 9 anchors at a position of an image. The following graph shows 9 anchors at the position (320, 320) of an image with size (600, 800).



1. Three colors represent three scales or sizes: 128x128, 256x256, 512x512.
2. The three boxes have height width ratios 1:1, 1:2 and 2:1 respectively.

If we choose one position at every stride of 16, there will be 1989 (39x51) positions. This leads to 17901 (1989 x 9) boxes to consider. The sheer size is hardly smaller than the combination of sliding window and pyramid. Or you can reason this is why it has a coverage as good as other state of the art methods. The bright side here is that we can use region proposal netowrk, the method in Fast RCNN, to significantly reduce number.

These anchors work well for Pascal VOC dataset as well as the COCO dataset. However, we have the freedom to design different kinds of anchors/boxes. For example, we are designing a network to count passengers/pedestrians, we may not need to consider the very short, very big, or square boxes. A neat set of anchors may increase the speed as well as the accuracy.

### **Region Proposal Network**

### The output of a region proposal network (RPN) is a bunch of boxes/proposals that will be examined by a classifier and regressor to eventually check the occurrence of objects. To be more precise, RPN predicts the possibility of an anchor being background or foreground, and refine the anchor.

### https://cdn-images-1.medium.com/max/1600/1*WO3athE5rXRW76CGbEqk9w.jpeg

### The Classifier of Background and Foreground

### The first step of training a classifier is make a training dataset. The training data is the anchors we get from the above process and the ground-truth boxes. The problem we need to solve here is how we use the ground-truth boxes to label the anchors. The basic idea here is that we want to label the anchors having the higher overlaps with ground-truth boxes as foreground, the ones with lower overlaps as background. Apparently, it needs some tweaks and compromise to separate foreground and background. Now we have labels for the anchors.

### The second question here is what features of the anchors are.

### Let’s say the 600x800 image shrinks 16 times to a 39x51 feature map after applying CNNs. Every position in the feature map has 9 anchors, and every anchor has two possible labels (background, foreground). If we make the depth of the feature map as 18 (9 anchors x 2 labels), we will make every anchor have a vector with two values (normal called logit) representing foreground and background. If we feed the logit into a softmax/logistic regression activation function, it will predict the labels. Now the training data is complete with features and labels.

### Another thing we may pay attention to is receptive field if we want to re-use a trained network as the CNNs in the process. Make sure the receptive fields of every position on the feature map cover all the anchors it represents. Otherwise the feature vectors of anchors won’t have enough information to make predictions

### In the architecture of Overfeat, it only uses non-overlapping convolutional and pooling filters to make sure every position in the feature map cover its own receptive field without overlapping others. In Faster R-CNN, receptive fields of different anchors often overlap each other, as we can from the above graph. It leaves the RPN to be position-aware.

### The Regressor of Bounding Box

### If we follow the process of labelling anchors, we can also pick out the anchors based on the similar criteria for the regressor to refine. One point here is that anchors labelled as background shouldn’t include in the regression, as we don’t have ground-truth boxes for them. The depth of feature map is 32 (9 anchors x 4 positions). https://cdn-images-1.medium.com/max/1600/1*jA7B88OXz2FJRKCHcOQ2bg.png

### The paper uses smooth-L1 loss on the position (x ,y) of top-left the box, and the logarithm of the heights and widths, which is as the same as in Fast R-CNN. The overall loss of the RPN is a combination of the classification loss and the regression loss.

MASK R-CNN

Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster R- CNN and Fully Convolutional Network (FCN) frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast training and inference time. Our goal in this work is to develop a comparably enabling framework for *instance segmentation*.

Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of *object detection*, where the goal is to classify individual objects and localize each using a bounding box, and *semantic segmentation,* where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.

This method called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in *parallel* with the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to- pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

****

The **Mask R-CNN** framework for instance segmentation

Working:

The model we use to perform Instance segmentation is rather simple. Instance segmentation can essentially be solved in 2 steps:

1. Perform a version of object detection to draw bounding boxes around each instance of a class
2. Perform semantic segmentation on each of the bounding boxes

This amazing simple model actually performs extremely well. It works, because if we assume step 1 to have a high accuracy, then semantic segmentation in step 2 is provided a set of images which are guaranteed to have only 1 instance of the main class. The job of the model in step 2 is to just take in an image with exactly 1 main class, and predict which pixels correspond to the main class (cat/dog/human etc.), and which pixels correspond to the background of an image.

The working principle of Mask R-CNN is again quite simple. All they (the researchers) did was stitch 2 previously existing state of the art models together and played around with the linear algebra (deep learning research in a nutshell). The model can be roughly divided into 2 parts — a region proposal network (RPN) and binary mask classifier.

Step one is to get a set of bounding boxes that could possibly contain an object of relevance. The fancy word of the day here is RoI Align. The RoI Align network work on principles of object detection, but it outputs multiple *possible* bounding boxes rather than a single definite one.

The second stage is to actually *do*the colouring. Au contraire to what one might think, this step is also quite easy! All we need to do is apply the existing state of the art model for semantic segmentation to each bounding box. The cool part is that since we are guaranteed to have at most 1 class in each box, we just to train our semantic segmentation model like a binary classifier, meaning we only need to learn the mapping from input pixels to a 1/0.1 would represent the presence of an object, and 0 would be the background.

Timeline of our Project

* By mid-February – Ear recognition using described models
* By the end of March – Ear detection in unconstrained environment using R-CNN
* By May – Integration of above techniques for final application in security and surveillance.