ISTANBUL TECHNICAL UNIVERSITY, COMPUTER ENGINEERING

photon

BLG521E Artificial Intelligence Final Project Report

Anıl Öztürk

504181504

January 5, 2020

1 THE PROJECT DEFINITION

The aim of the **Project Photon** is to develop **a concept software that provides the most compatible advertisement according to the population distribution in front of a billboard**. The software tries to show the advertisements of the companies whose target is closest to the current audience by extracting the age and gender statistics of the audience comes from the camera feed on top of the billboard.

The algorithm is running on **Python3** and the model has been created, trained and executed on **PyTorch**. **State-of-art object detectors** and **own estimators** are used.

2 THE MOTIVATION

In the marketing and advertisement field, there is so much money that gets wasted in vain. My project targets that squander. With advertising via my device, the companies will save the money of renting ads when there is no people around the billboard. They can more precisely target audiences and get feedback of their targeting accuracies.

3 PREPROCESSING THE DATA

3.1 GETTING THE DATASETS

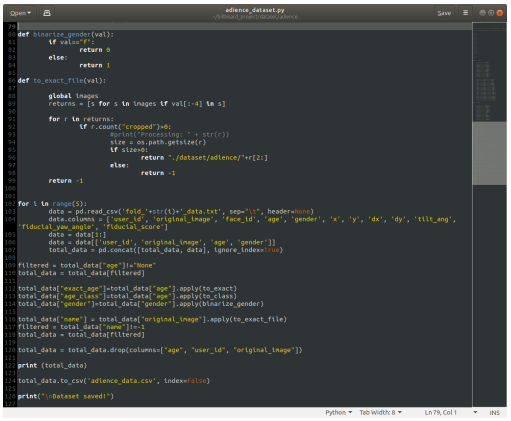
For the project, I have downloaded three different datasets with different layout from different open-source platforms. Firstly, I needed to preprocess the data and put them together in a common scheme. These are;

- Adience
- IMDB-face
- Wiki-face

3.1.1 PREPARING THE DATASETS

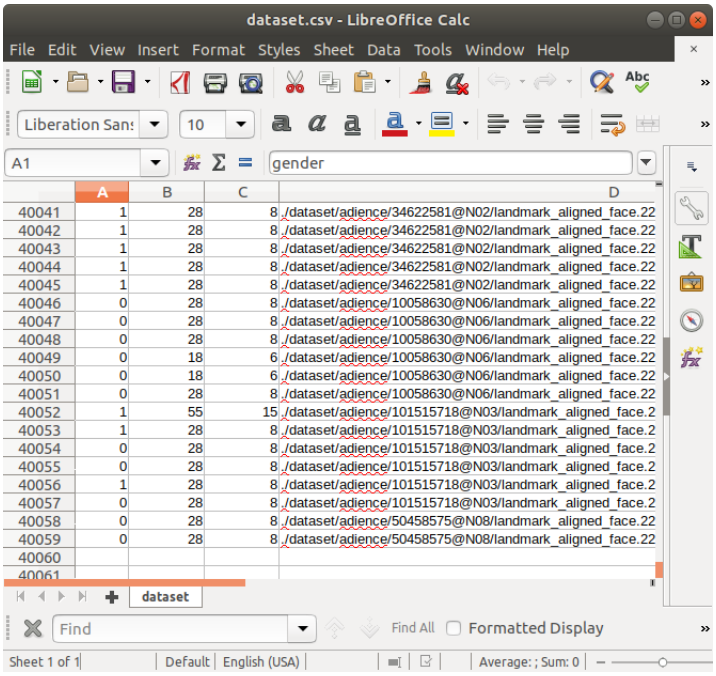
Adience dataset includes **about 19300** images of human faces from different age intervals and genders. IMDB dataset includes **about 460000** images of human faces from different age intervals and genders. Wikipedia dataset includes **about 63000** images of human faces from different age intervals and genders.

I wrote a little script to convert each of these datasets into a format that I can use easily. The following screenshot is from the adience's converter.



```
def binarize_gender(val):
    if val=='f':
        return 0
    else:
        return 1

def to_exact_file(val):

    global images
    returns = [i for i in images if val[1:4] in i]

    for r in returns:
        if r.count("cropped")>0:
            #print("Processing: " + str(r))
            size = os.path.getsize(r)
            if size>0:
                return "./dataset/adience/"+r[2:]
            else:
                return -1
    return -1

for i in range(5):
    data = pd.read_csv('fold_'+str(i)+'_data.txt', sep="\t", header=None)
    data.columns = ['user_id', 'original_image', 'face_id', 'age', 'gender', 'x', 'y', 'dx', 'dy', 'tilt_ang',
'fiducial_yaw_angle', 'fiducial_score']
    data = data[1:]
    data = data[['user_id', 'original_image', 'age', 'gender']]
    total_data = pd.concat([total_data, data], ignore_index=True)

filtered = total_data['age']!='None'
total_data = total_data[filtered]

total_data['exact_age']=total_data['age'].apply(to_exact)
total_data['age_class']=total_data['age'].apply(to_class)
total_data['gender']=total_data['gender'].apply(binarize_gender)

total_data['name'] = total_data['original_image'].apply(to_exact_file)
filtered = total_data['name']!=-1
total_data = total_data[filtered]

total_data = total_data.drop(columns=["age", "user_id", "original_image"])

print (total_data)

total_data.to_csv('adience_data.csv', index=False)

print("\nDataset saved!")
```

After converting each dataset, I merged them into one file to use them easily in the training phase. The following screenshot is an example from early stages of the merged dataset.



dataset.csv - LibreOffice Calc

File Edit View Insert Format Styles Sheet Data Tools Window Help

Liberation Sans 10

A1 gender

	A	B	C	D
40041	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40042	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40043	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40044	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40045	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40046	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40047	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40048	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40049	0	18	6	./dataset/adience/10058630@N06/landmark_aligned_face.22
40050	0	18	6	./dataset/adience/10058630@N06/landmark_aligned_face.22
40051	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40052	1	55	15	./dataset/adience/101515718@N03/landmark_aligned_face.2
40053	1	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40054	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40055	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40056	1	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40057	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40058	0	28	8	./dataset/adience/50458575@N08/landmark_aligned_face.22
40059	0	28	8	./dataset/adience/50458575@N08/landmark_aligned_face.22
40060				
40061				

dataset

Find Find All Formatted Display

Sheet 1 of 1 | Default | English (USA) | Average: ; Sum: 0

3.1.2 SETTING LABELS

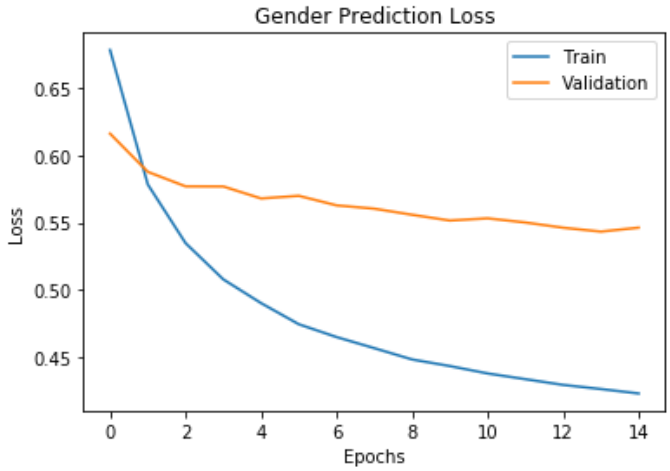
I used genders and ages as values. While I was binarizing gender values, I splitted the ages into groups. I decided to set the group count as **7**. These groups are:

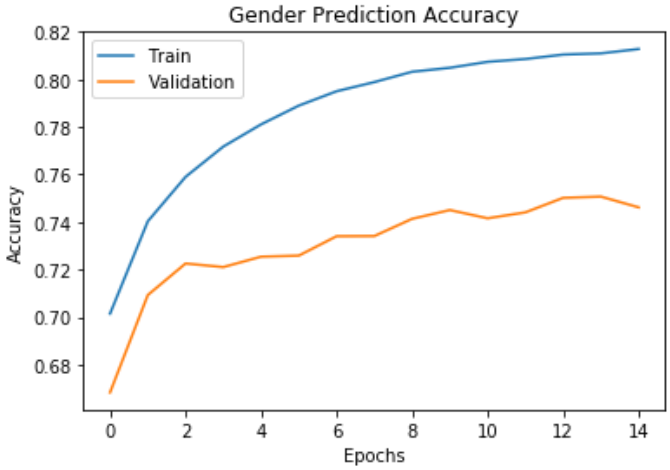
- 0-5
- 5-15
- 15-25
- 25-35
- 35-45
- 45-60
- 60+

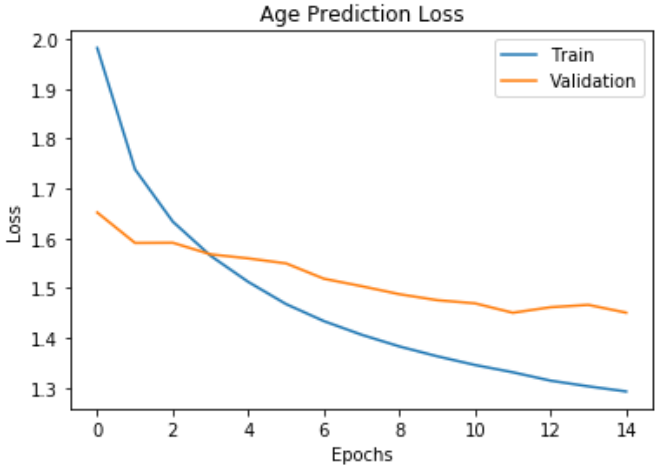
4 TRAINING PHASE

- I did use **266126** samples total for my training. I have splitted them as **0.8**, **0.15**, **0.05** as training, validation, test splits; respectively.
- I **one-hot encoded** each type of output (gender, age)
- While using these dataloaders, I **shuffled** the output data everytime to prevent my model to learn the order of data.
- I have applied **normalization** on the images. But in my model, as a **lambda layer**.
- I have used the **MobileNet** architecture as backbone for my model.
- I have placed several **Fully Connected**, **Batch Normalization** and **Leaky ReLU Activation** layers after the backbone structure.
- I have used **Global Average Pooling** at the output of the backbone model.
- I have used **Dropout** on **Fully Connected** outputs.
- I gave two seperate output to my model for gender and age, seperately.
- Since I one-hot encoded each output. I set the loss as categorical-crossentropy on these outputs.
- I used **Adam** optimizer with **beta1: 0.9, beta2: 0.999, lr: 1e-5**
- I have trained my model over **15** epochs.

After training, I got following loss and accuracy results:







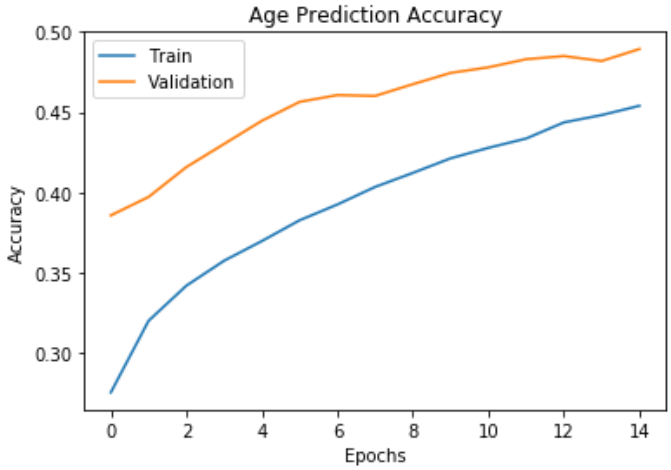
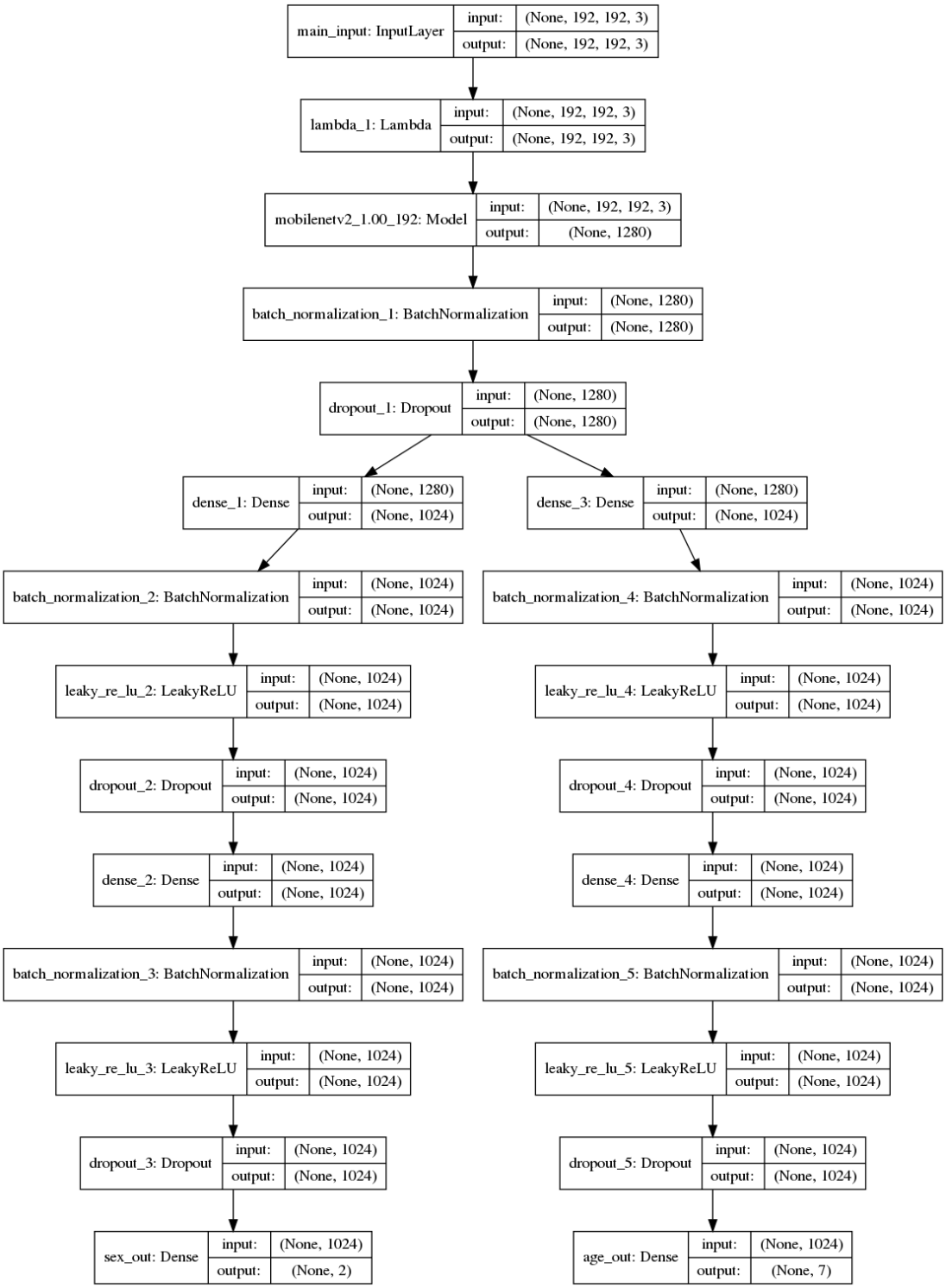

Age Prediction Accuracy
Train
Validation
Accuracy
Epochs
0.50
0.45
0.40
0.35
0.30
0
2
4
6
8
10
12
14

Illustration of my model is as follows:



5 TRAINING RESULTS

Since we didn't see any reverse-headed plots, we can say that the model didn't overfit during training phase. The final **validation** accuracy of the model is as follows:

- **Gender Accuracy:** 74.62% (Over 2 groups)
- **Age Accuracy:** 48.94% (Over 7 groups)

6 USING PRE-TRAINED FACE DETECTORS

Since we should use ready-to-use face detector, we should select it to be robust over the size of faces. I have used **Ultra-Light-Fast-Generic-Face-Detector** as the face detector for my project. It's specs are below:

- **Inference Time:** 29ms
- **Model Size:** 1.04MB

7 SELECTING BEST-FIT ADVERTISEMENT

An advertisement desirability (score) function is used for selecting the best-fit advertisement for the crowd at the given moment. The advertisement that gives the **maximum score output** with the given crowd statistics will be selected as the next advertisement to be shown. It will likely include these variables;

- Age ratio of men in the whole crowd (A_1^i) (i=0..6)
- Age ratio of women in the whole crowd (A_0^i) (i=0..6)
- Age index selection of an advertisement ($S_A = \{0, 3, 4, ...\}$)
- Gender index selection of an advertisement ($S_G = \{0, 1\}$)
- Display count of an advertisement (C)
- Total age-gender combination of target selections of an advertisement (T)
- Total advertisement count (E)

The score function output sc for the given ad x would be like;

$$sc(x) = \frac{\sum_{i=S_G} \sum_{j=S_A} A_i^j}{ln(CE+e)T}$$

```
for adv_it in range(len(self.advertisements)):
    score = 0
    itt = 0
    for gender in self.advertisements[adv_it]["gender_target"]:
        for age in self.advertisements[adv_it]["age_target"]:
            score += self.running_stats[gender*7+age]
            itt += 1

    score /= np.log(np.e + len(self.advertisements)*self.advertisements[adv_it]["showed"])

    self.advertisements[adv_it]["score"] = score/itt
```

8 GRAPHICAL USER INTERFACE

An example screenshot from the running program is below. It has some specific sections about live-data and ad correlations;

3 Females
2 Males

Female Statistics

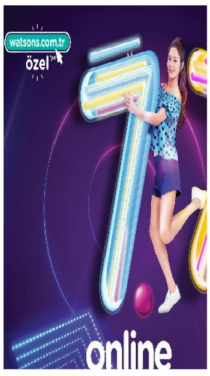
0.0% (0-5)
0.0% (5-15)
20.0% (15-25)
40.0% (25-35)
0.0% (35-45)
0.0% (45-60)
0.0% (60+)

Male Statistics

0.0% (0-5)
0.0% (5-15)
20.0% (15-25)
20.0% (25-35)
0.0% (35-45)
0.0% (45-60)
0.0% (60+)

f (15-25)
f (25-35)
f (25-35)
m (15-25)
m (25-35)

	Advertisement	Target Gender	Target Age	Score
1	Watsons	f	(15-25), (25-35)	30.0
2	Coca-Cola	f, m	(15-25), (25-35), ...	16.666666666666...
3	Altınyıldız	m	(25-35), (35-45)	10.0
4	Toys R Us	f, m	(5-15), (15-25)	10.0
5	Anadolu Hayat	f, m	(35-45), (45-60), ...	0.0



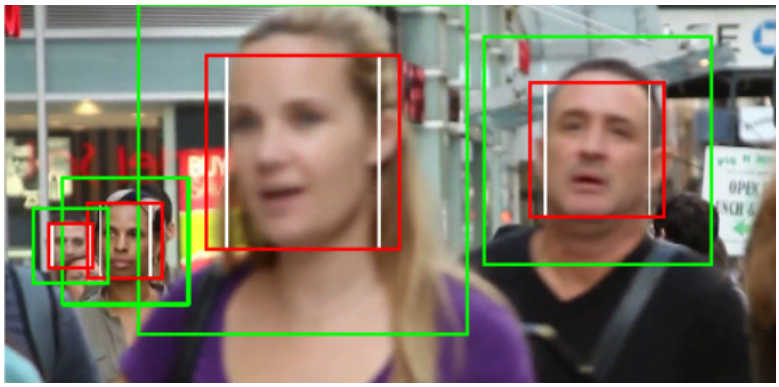
Selected Ad:
Watsons

photon

- At the top-left, there is raw live-stream camera feed
- At the bottom-left, there is processed stream with NN models
- In the middle, there is a running statistics about gender and age distribution of corresponding camera frame
- At the top-right, there is a table that consists scores of advertisements for given frame
- At bottom-right, there is the selected advertisement for given crowd which has the highest evaluation score

9 Inference Phase Image Trick

Training data have made up by images of faces with a specific **margin**. The detection models train easier when the cropping has been done like this, because they also learn **the noise pattern** around the face. But my ready-to-use face detector was giving me the bounding-box of the face. So I had to do some basic calculations and get that area to be wider.



- **White Box:** Output of face detector
- **Red Box:** Calculated GUI output
- **Green Box:** Calculated input of age-gender network

10 Test Results

As we can see, our program runs two seperate networks at near-realtime (**around 25FPS**) and compute some scores based on their outputs, finally selects the best-fit advertisement for the given crowd.

1 Females
1 Males

Female Statistics

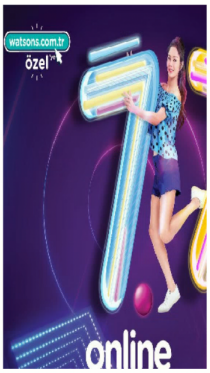
0.0% (0-5)
0.0% (5-15)
0.0% (15-25)
50.0% (25-35)
0.0% (35-45)
0.0% (45-60)
0.0% (60+)

Male Statistics

0.0% (0-5)
0.0% (5-15)
0.0% (15-25)
50.0% (25-35)
0.0% (35-45)
0.0% (45-60)
0.0% (60+)

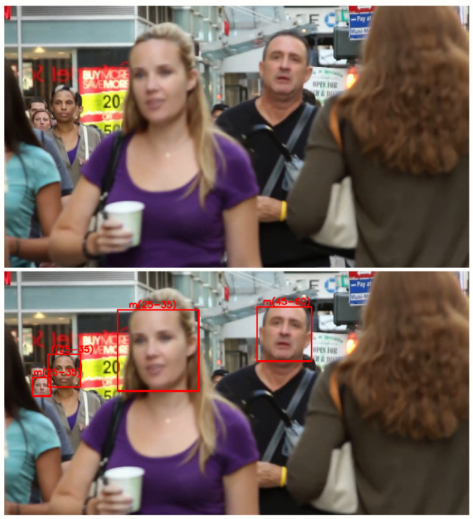
f (25-35)
m (25-35)

	Advertisement	Target Gender	Target Age	Score
1	Watsons	f	(15-25), (25-35)	25.0
2	Altınyıldız	m	(25-35), (35-45)	25.0
3	Coca-Cola	f, m	(15-25), (25-35), ...	16.666666666666...
4	Toys R Us	f, m	(5-15), (15-25)	0.0
5	Anadolu Hayat	f, m	(35-45), (45-60), ...	0.0



Selected Ad:
Watsons

photon



1 Females
3 Males

Female Statistics

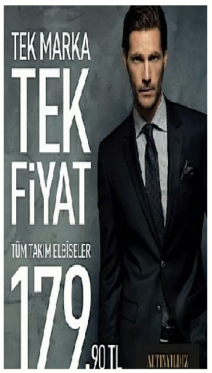
0.0% (0-5)
0.0% (5-15)
0.0% (15-25)
25.0% (25-35)
0.0% (35-45)
0.0% (45-60)
0.0% (60+)

Male Statistics

0.0% (0-5)
0.0% (5-15)
0.0% (15-25)
50.0% (25-35)
0.0% (35-45)
25.0% (45-60)
0.0% (60+)

m (25-35)
m (45-60)
f (25-35)
m (25-35)

	Advertisement	Target Gender	Target Age	Score
1	Altınyıldız	m	(25-35), (35-45)	25.0
2	Watsons	f	(15-25), (25-35)	12.5
3	Coca-Cola	f, m	(15-25), (25-35), ...	12.5
4	Anadolu Hayat	f, m	(35-45), (45-60), ...	4.166666666666667
5	Toys R Us	f, m	(5-15), (15-25)	0.0



Selected Ad:
Altınyıldız

photon

11 CRITIC

There are two main problems with the model;

- The model runs at realtime on a CUDA based GPU but it lacks on performance on compact device like Raspberry Pi. It can be compressed or pruned into a smaller network.
- The model basically trained on 128x128 images. It lacks of performance on small face detections since they are scaled up to 128x128 and it causes a blurry image. The model can be trained on a special small face dataset and it would increase the accuracy of the model.

12 REFERENCES

- https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB
- https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/
- https://talhassner.github.io/home/projects/Adience/Adience-data.html
- https://keras.io/applications/#mobilenetv2