Facial Keypoints Detection: Modern Approaches Based on ResNets

Vision and Cognitive Services Project

Alvise Dei Rossi, Lorenzo Corrado

September 21, 2021

DATA SCIENCE
UNIVERSITY OF PADOVA

1. Introduction
2. Related Works
3. Dataset and preprocessing
4. Methods and Experiments
5. Conclusions

1 Introduction

2 Related Works

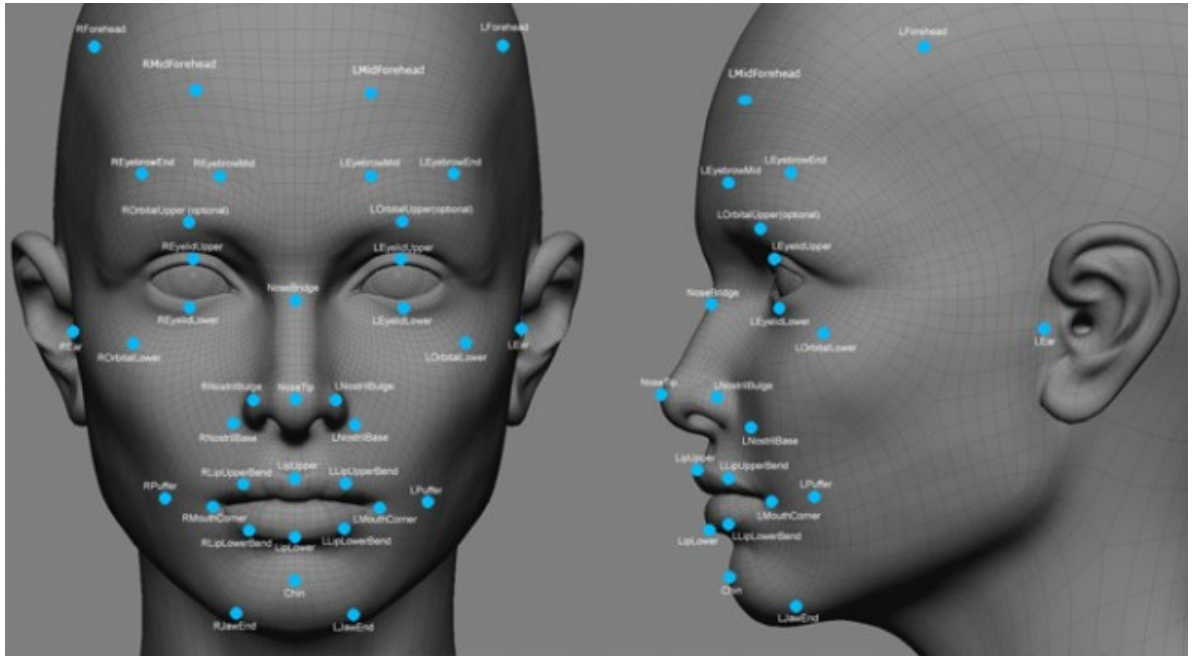
3 Dataset and preprocessing

4 Methods and Experiments

5 Conclusions

Introduction

Facial keypoints detection represents an important challenge in computer vision.



General task: Correctly predict the coordinates $(x; y)$ of the most important features of a person's face.

Some applications

The correct prediction of facial keypoints has many useful applications:

- Security systems based on biometrics
- Multimedia field
- Medical field

Why is it difficult?

This problem is very challenging:

- Facial features are widely variable among different people.
- Robustness with respect to all possible variations in which an image is collected
- It is necessary to use high resolution images to capture the main details of a person's face

Framework

Our task: Correctly identify 51/68 facial keypoints on each image we give in input to our model.

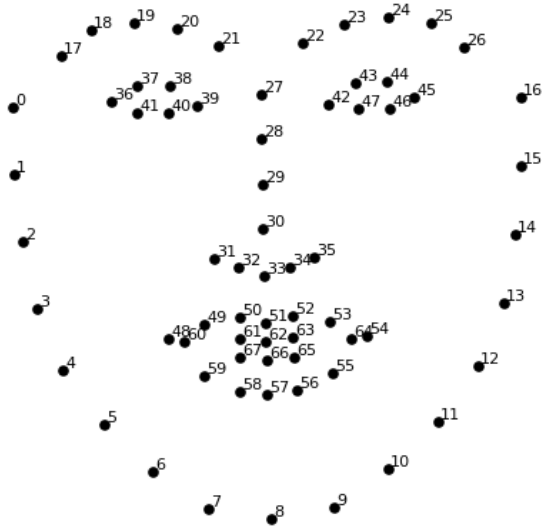


Figure 1: The 68 points mark-up used.

The framework in which we're going to test these models will be the **"300 Faces In-The-Wild Challenge"**, organized in 2013 and 2015.

Summary of the approaches

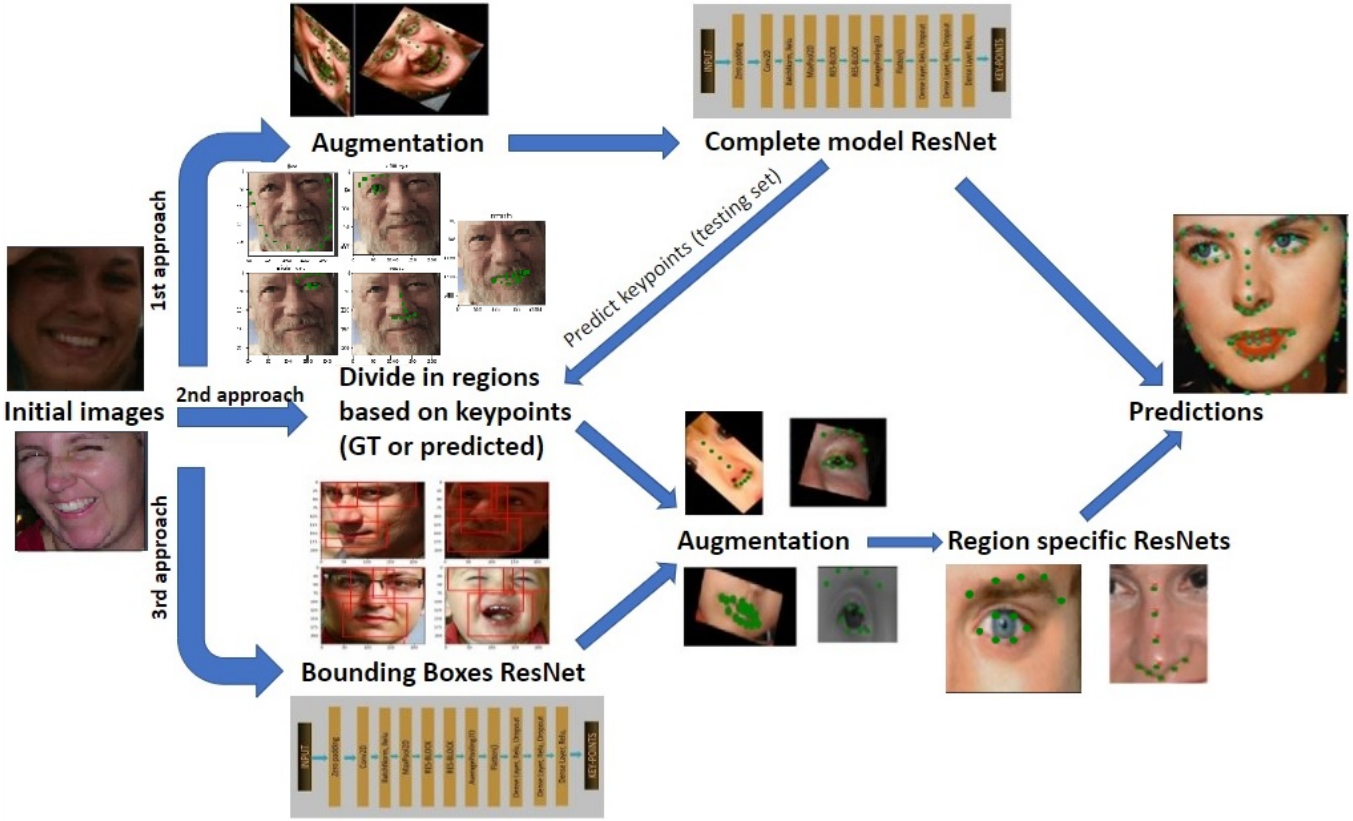


Figure 2: Scheme of the three approaches used.

Setup

The setup we used in this experiment was provided by Google Cloud Platform. The code was run on a virtual machine that provided:

- 16 vCPUs
- RAM 64Gb
- Nvidia Tesla P100 GPU

Google Cloud

1 Introduction

2 Related Works

3 Dataset and preprocessing

4 Methods and Experiments

5 Conclusions

Previous winners - I

(Yan et al., 2013) and (Deng et al., 2015) employed a cascade regression method, where a series of regressors are utilized to progressively refine the shape initialized by the face detector. Given N training examples $\{I_i, S_i, S_i^0\}$, where I_i is the image, S_i the groundtruth shape of face in I_1 and S_i^0 is the initialization of S_i. The aim is to learn a regression function f that minimizes the MSE:

$$f = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} ||(f(S_i^0, I_i) - S_i)||_2 \tag{1}$$

where f returns a new shape based on the initial shape S_i^0 for each image I_i, it is important to say that:

$$f = f_T \circ f_{T-1} \circ \cdots \circ f_1 \tag{2}$$

Combining f_1 and f_T it is possible to obtain complex non-linear mapping.

Previous winners-II

(Zhou et al., 2013) and (Fan et al. 2015) employed a method wich is pretty similar to what we will present. For example, the first authors use a cascade of multiple CNNs. A first-level CNN that predicts bounding-boxes for the inner points and contour points separately, a second-level CNN predict an initial estimation of the positions which are refined by a third-level for each component. The fourth level is used to improve the predictions of mouth and eyes by taking the rotated image patch as new input.

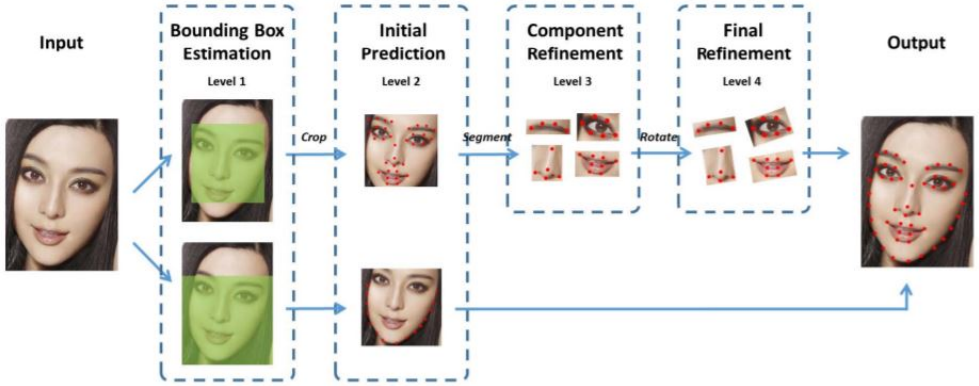


Figure 3: Scheme proposed by (Zhou et al., 2013).

As already mentioned, our approaches are similar to those proposed by two of the challenge winners, some differences are:

- Modern deep learning architectures and regularization
- Data augmentation
- Different strategies to obtain bounding-boxes

Criterion of comparison

To compare our results with those of other authors we used:

$$RMSE = \frac{\sum_{i=1}^{N} \sqrt{(x_i^f - x_i^g)^2 + (y_i^f - y_i^g)^2}}{d_{outer}} \tag{3}$$

Where $[x_1^f, y_1^f, \ldots x_N^f, y_N^f]$ represents the measured coordinates and $[x_1^g, y_1^g, \ldots x_N^g, y_N^g]$ represents the ground-truth.

What is d_{outer}?

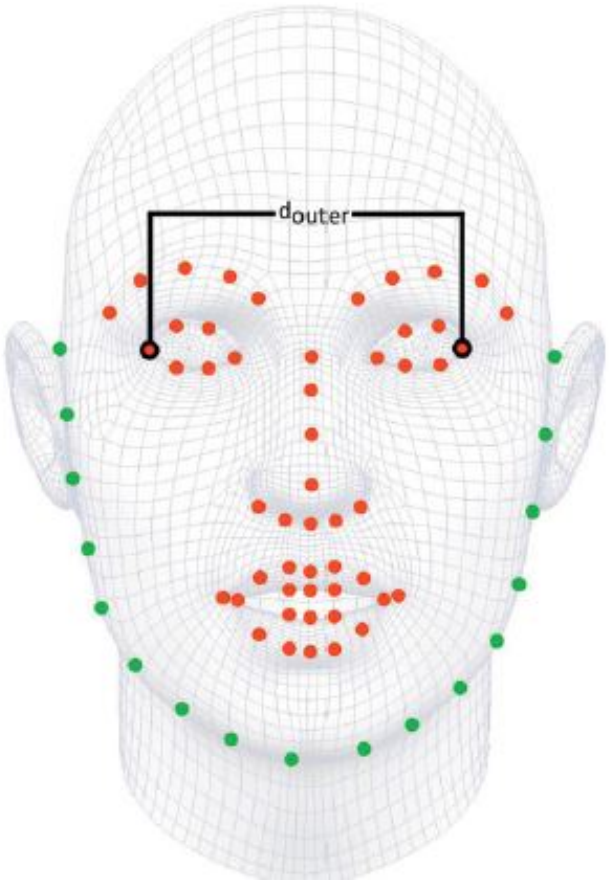


Figure 4: The interoccular distance.

1 Introduction

2 Related Works

3 Dataset and preprocessing

4 Methods and Experiments

5 Conclusions

Dataset

The dataset of the challenge comes from the aggregation of 5 different datasets:

- **LFPW**
- **HELEN**
- **AFW**
- **AFLW**
- **300-W**

The images inside them have been collected in different light conditions, different backgrounds, different expressions, different sizes and resolutions, the presence of objects inside and also with the present of different subjects.

The first four datasets originally differed in the number of keypoints provided.

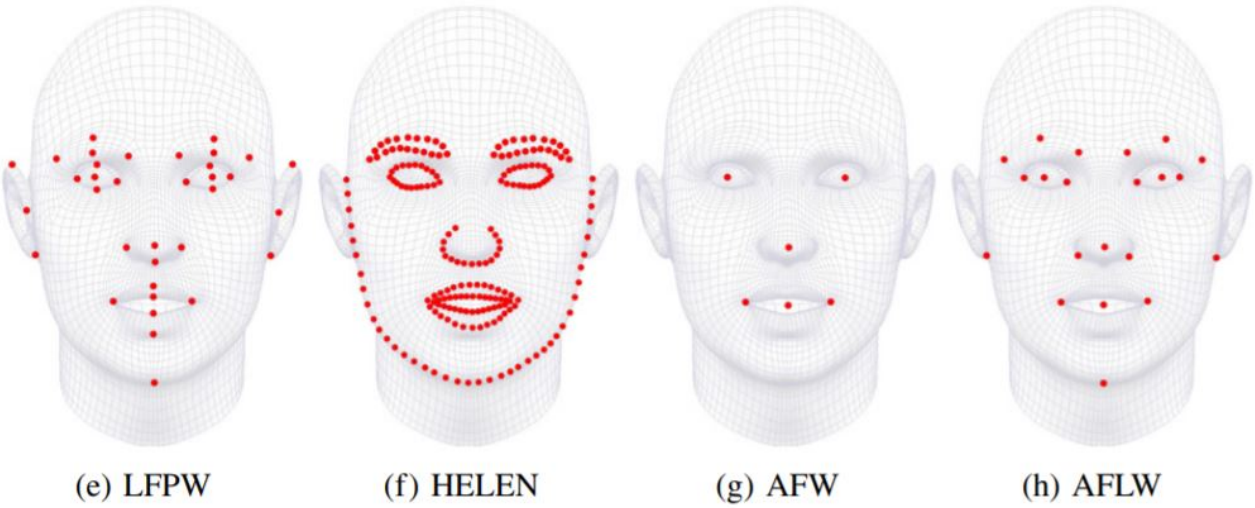


Figure 5: Landmark configurations of the first four datasets.

How to assign the 68 mark-up?

- In order to obtain images with the 68 landmark points the authors of the challenge used semi-supervised learning techniques to re-annotate them
- Each dataset is provided with the ground-truth coordinates of the 68 facial keypoints, which will be used to train and validate the models

Preprocessing

An extensive preprocessing phase was needed to prepare the dataset:

1. **Cropping**
2. **Keypoints rescaling**
3. **Resizing**
4. **Keypoints rescaling**
5. **Normalization**

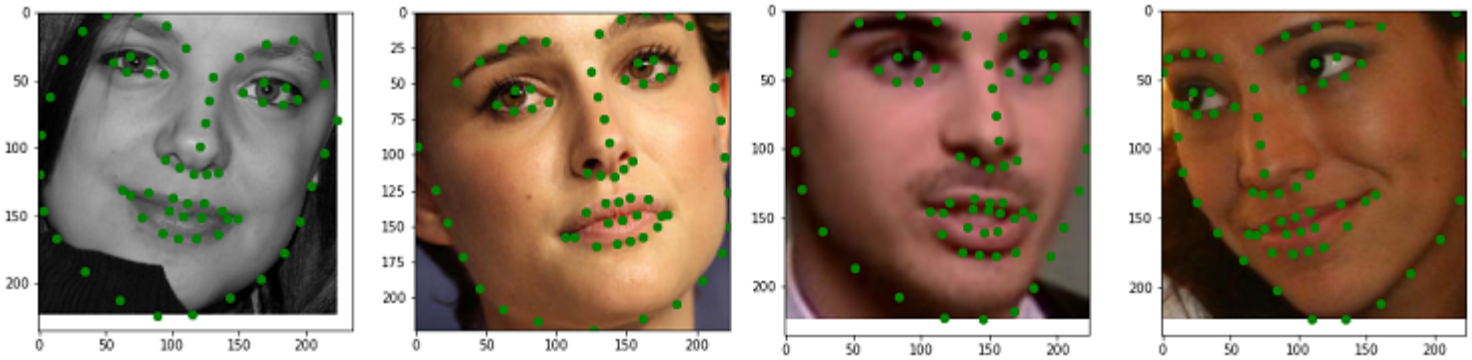


Figure 6: Examples of images after preprocessing.

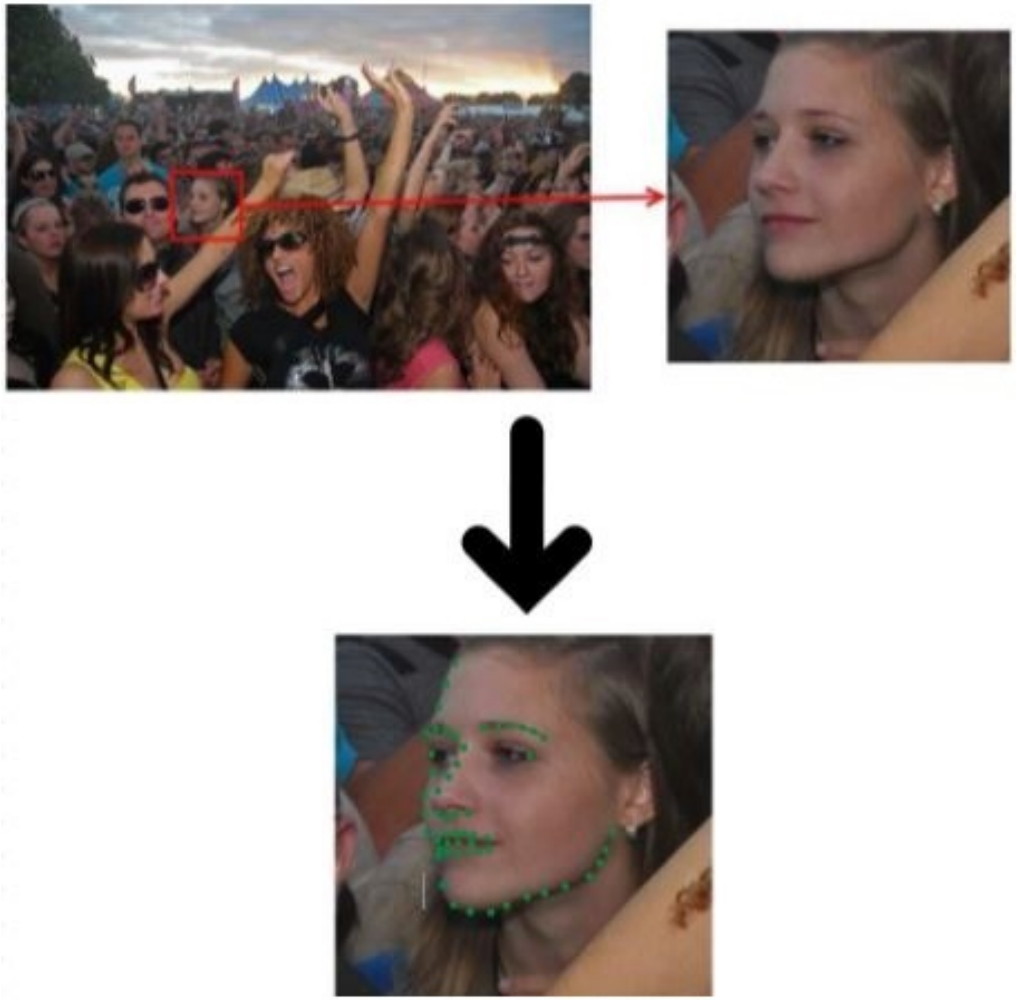


Figure 7: A more detailed example.

Dataset after preprocessing

The final dimensions of the dataset are:

- $3,148$ images in the training set (70.9%)
- 688 in the validation set (15.5%)
- 600 in the test set (13.6%)

In which each image is $224x224x3$ in size.

1 Introduction

2 Related Works

3 Dataset and preprocessing

4 Methods and Experiments

5 Conclusions

Regression based approach

- No iterative fitting step or sliding window approach required
- Usually faster than other approaches
- Depends on the regressor's capacity to learn complex relations between pixel values and facial features
- Past improvements driven by depth and complexity of the model
- **Main challenge:** optimally train the model (hyperparams) and/or frame the problem in simpler terms

ResNets

- Plain deep networks suffer from degradation problems
- Allows to train deeper networks through use of skip-connections
- Alleviate vanishing gradient problems
- Ease of identity mapping learning
- Easy enough to train and implement

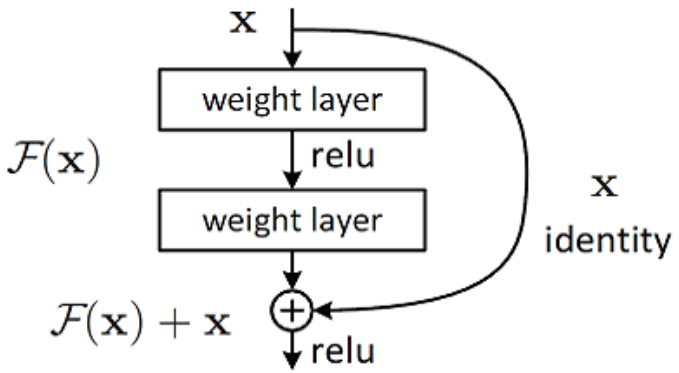


Figure 8: Residual learning: a building block.

ResNets implementation - I

- **Convolution blocks** alternated to **identity blocks**
- Dimensionality reduction occurs in convolution block
- No reduction in identity block, short path connection doesn't use further parameters

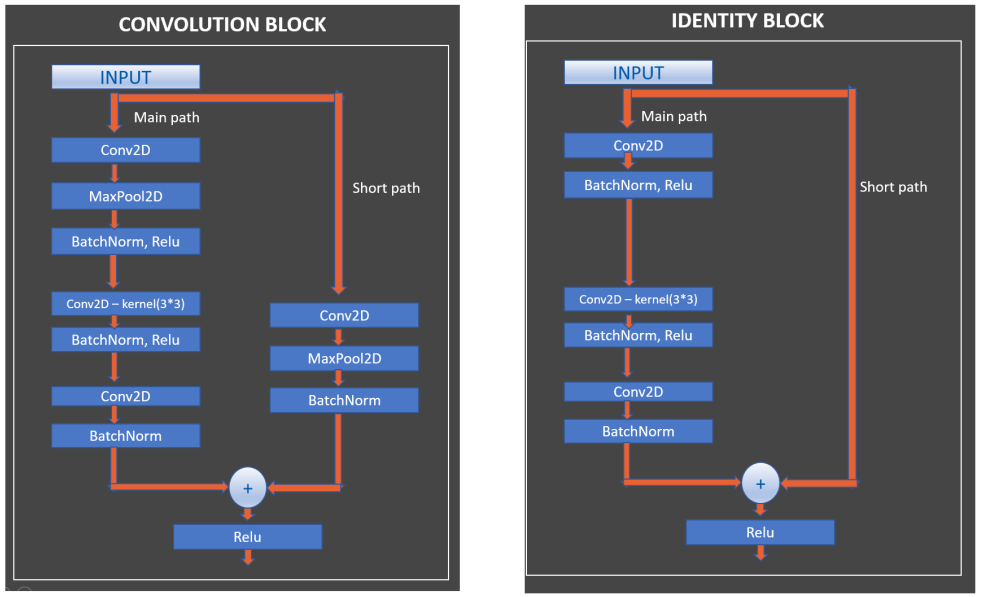


Figure 9: Convolution block and identity block.

ResNets implementation - II

- Bottleneck design used as original paper, reducing number of parameters without degrading performance of the network
- Tradeoff between depth and training time
- Residual blocks composed by convolution block + 2x identity blocks

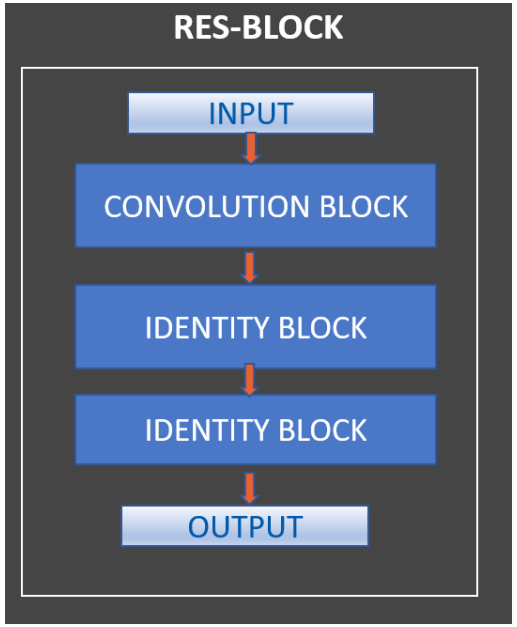


Figure 10: Residual block.

ResNets implementation - III

```python
# CONVOLUTIONAL BLOCK
X_copy = X
f1 , f2, f3 = filters

# Main Path
X = Conv2D(f1, (1,1), strides = (1,1))(X)
X = MaxPool2D((2,2))(X)
X = BatchNormalization(axis =3)(X)
X = Activation('relu')(X)

X = Conv2D(f2, kernel_size =(f,f), strides=(1,1), padding='same')(X)
X = BatchNormalization(axis =3)(X)
X = Activation('relu')(X)

X = Conv2D(f3, kernel_size=(1,1), strides=(1,1))(X)
X = BatchNormalization(axis=3)(X)

# Short path
X_copy = Conv2D(f3, kernel_size = (1,1), strides =(1,1))(X_copy)
X_copy = MaxPool2D((2,2))(X_copy)
X_copy = BatchNormalization(axis=3)(X_copy)

# Add data from main and short paths
X = Add()([X,X_copy])
X = Activation('relu')(X)
```

```python
# IDENTITY BLOCK
X_copy = X

# Main Path
X = Conv2D(f1, (1,1),strides = (1,1))(X)
X = BatchNormalization(axis =3)(X)
X = Activation('relu')(X)

X = Conv2D(f2, kernel_size = (f,f), strides =(1,1), padding = 'same')(X)
X = BatchNormalization(axis =3)(X)
X = Activation('relu')(X)

X = Conv2D(f3, kernel_size = (1,1), strides =(1,1))(X)
X = BatchNormalization(axis =3)(X)

# Add both paths together
X = Add()([X,X_copy])
X = Activation('relu')(X)
```

Figure 11: Python implementation.

1st Method

- A ResNet is used to directly predict all 68 keypoints
- The network takes the raw pixels of a 224x224 image as input and performs regression on the coordinates of the desired 68 keypoints, in a end-to-end fashion
- It was observed that initializing the weights of early layers using transfer learning from ResNet50 trained on ImageNet considerably speeds up training and improves performance

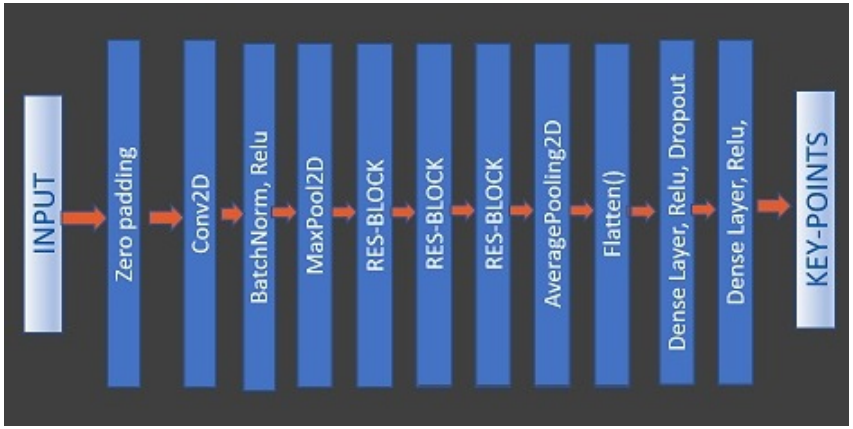


Figure 12: Architecture of ResNet used.

Hyperparameters tuning

- Hyperparameters have to be tuned appropriately
- Tests were carried out for: dimensions of filters in residual blocks, number of residual blocks, number of fully connected layers, optimizer, learning rate, dropout probability, batch size
- MSE loss utilized during training
- 50 epochs, early stopping
- Number of residuals blocks

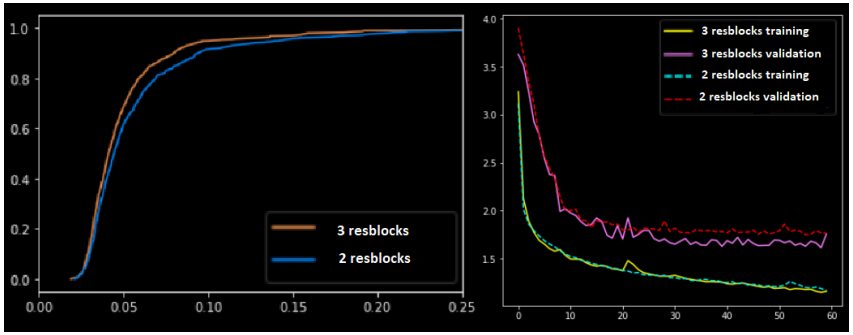


Figure 13: Effect of using more residual blocks.

Hyperparameters tuning - II

- Number of Fully connected layers

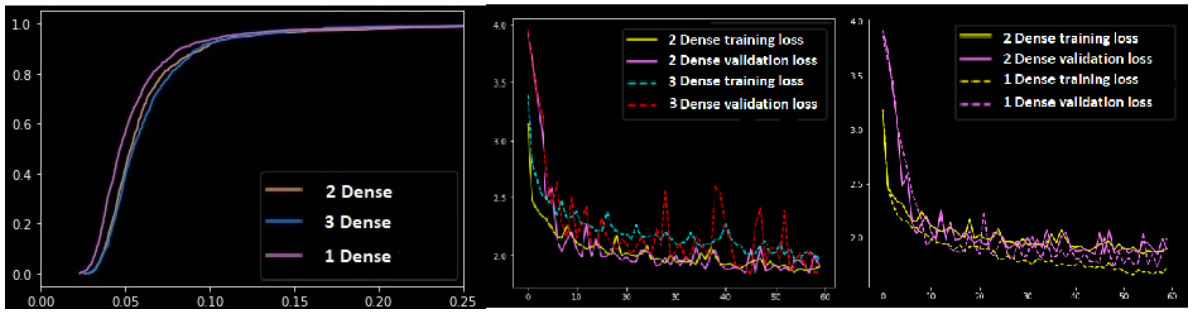


Figure 14: cumulative error rates and losses changing #dense layers.

- Tests on optimizers, batch size, dropout

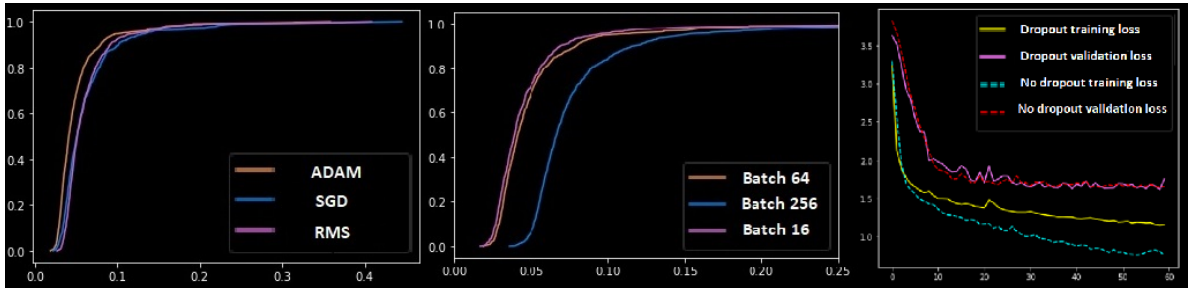


Figure 15: effects of optimizers, batch sizes, dropout.

Data augmentation - I

- Many past works directly rectified images through rotations and/or scale transformations based on pose estimations, calculating angles through keypoints (groundtruth or predicted) or standalone models.
- Instead, to improve robustness of our algorithms (without directly manipulating images), we use data augmentation.
- Data augmentation (sometimes) is more challenging than in image classification (keypoints move!)

Data augmentation - II

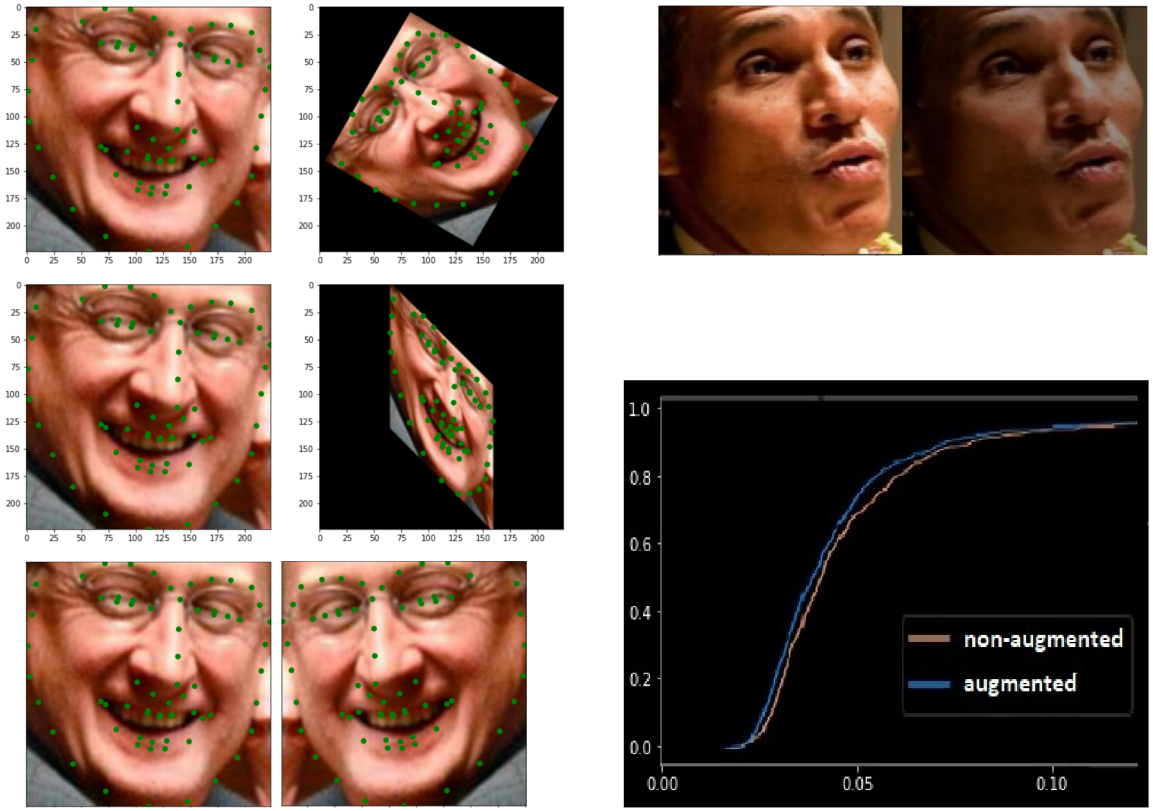


Figure 16: Examples where keypoints change (rotations,shear,flips), don't change (illumination) and effect of augmentation on performance.

2nd Method - I

- Divide images into sections and train region-specific ResNets
- When cropping and resizing images, keypoints have to be accordingly adjusted too
- Cuts based on: groundtruth keypoints (training/validation set), predictions of 1st model (testing set)

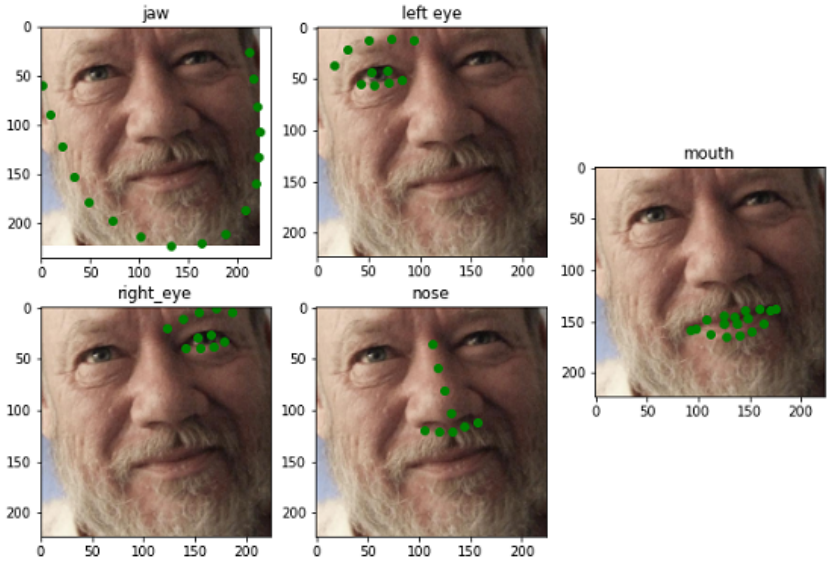


Figure 17: Keypoints relative to the 5 sections the images are divided into

2nd Method - II

- Similar architecture to 1st method ResNet, different input and output dimensionality
- Predictions obtained by region specific models undergo inverse transformations to reconstruct complete final predictions

Region	input dim.	output dim.	#Keypoints
eyes	100x100	22+22	22
mouth	75x150	40	20
nose	150x75	18	9
jaw	224x224	34	17

Table 1: Input, output dimensions of region-specific models.

2nd Method - Implementation details

- Bounding boxes obtained by keypoints are enlarged by a small random offset (allows for small failures in estimation process)

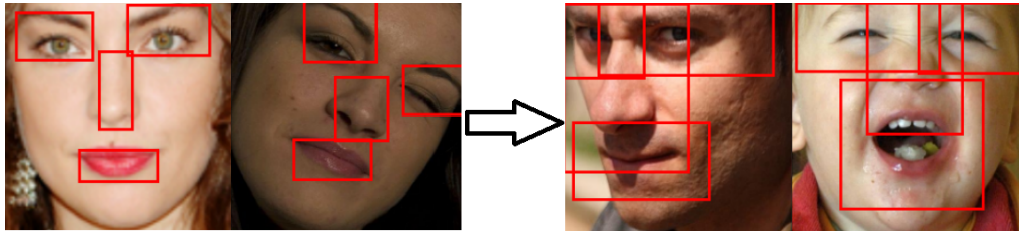


Figure 18: Bounding boxes enlargement.

- Data augmentation is used to train region-specific models too

Figure 19: Augmented portions of images.

2nd Method - observations

- Difficulty of localization is unbalanced across landmarks
- High variability in the definition of keypoints with less semantical meaning
- L2 loss is dominated by jaw errors in the complete model
- Jaw-specific model underperforms, context might help

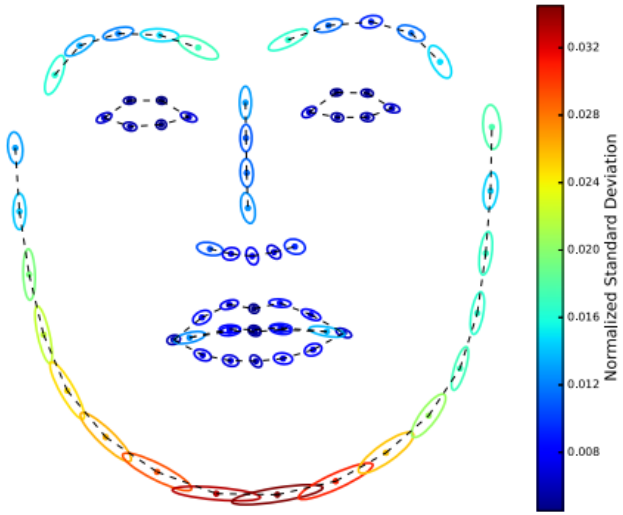


Figure 20: Each ellipse denotes the variance of each landmark point.

3rd Method

- 2nd method is highly sensitive to bounding-boxes definition. Can we use other methods to initiate them?
  - Regression
  - Sliding windows approaches
- The number of bounding boxes is defined, we know how many objects we're detecting (4), simple regression can be used
- As regressor another ResNet can be used: same implementation, different number of output units (16)

3rd Method - Implementation

- **Objective:** predict 4 bounding boxes (x, y, h, w) for eyes, mouth and nose
- no bounding box for jaw
- Random offset added to allow small errors in this case too
- Same region-specific models of 2nd method are then applied

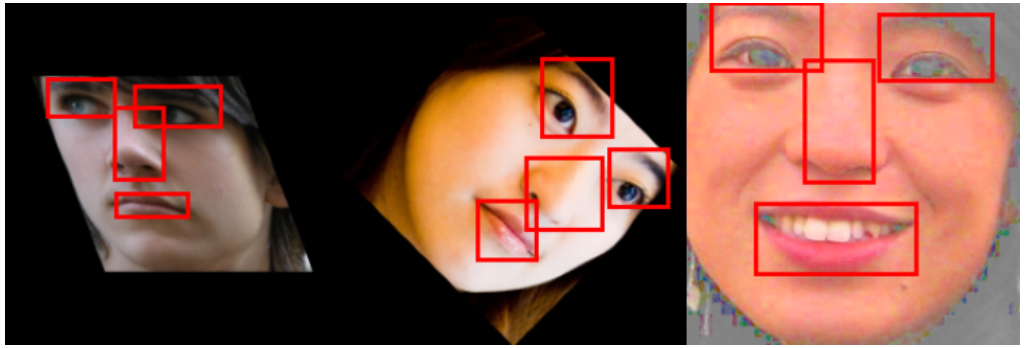


Figure 21: Data augmentation is still used on initial regressor.

Comparison between models

- Region-specific models outperform the complete model
- Initiating bounding boxes by use of ResNet rather than directly using keypoints is slightly more beneficial to performance
- Defining BBs using GT keypoints shows ideal error
- Cumulative error rates graph shows that error for testing set is usually bigger than for validation set

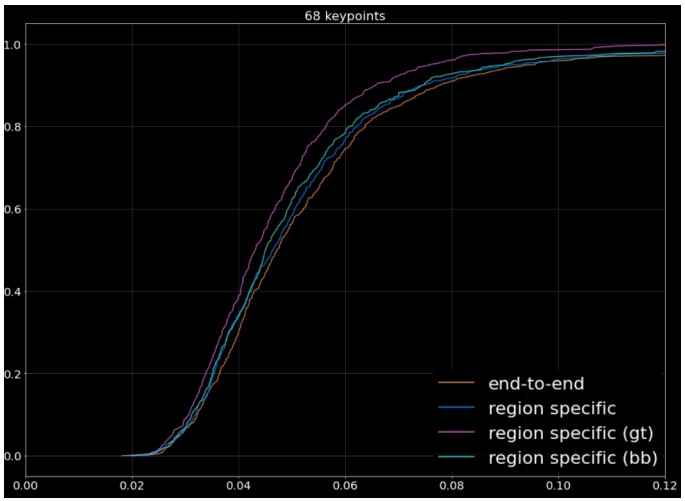


Figure 22: Cumulative error rates for methods proposed.

Comparison to past winners - I

- Comparable performance for easy images
- Improved robustness for more challenging images
- Much lower error on 51 keypoints compared to 68 keypoints
- Gap with ideal error

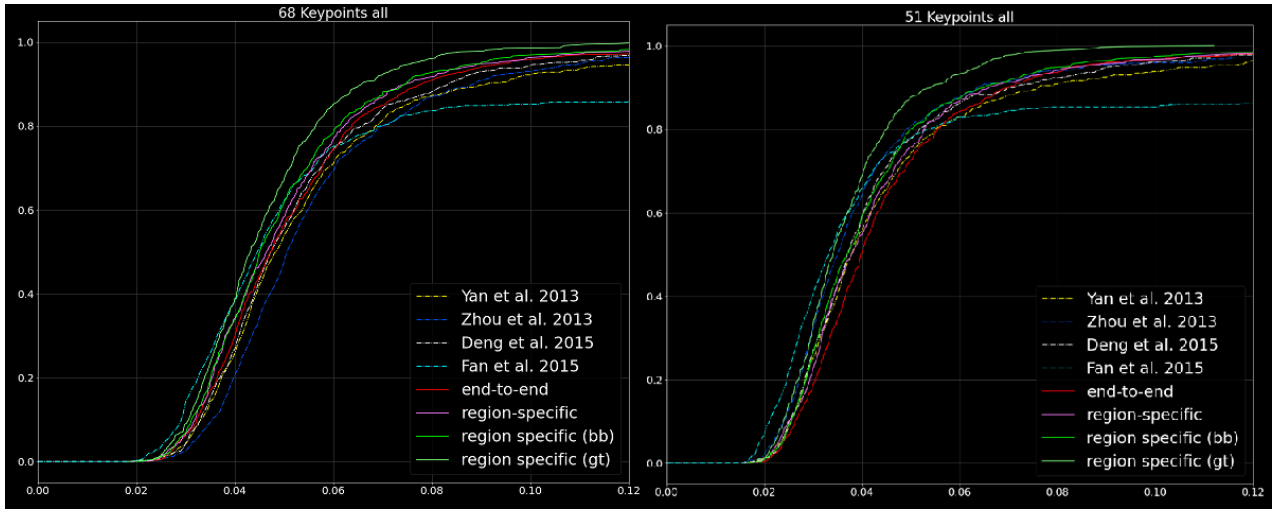


Figure 23: Comparison of past works and our methods, 68 and 51 kps.

Comparison to past winners - II

- Same performance on outdoor and indoor images
- Larger gap for challenging outdoor images

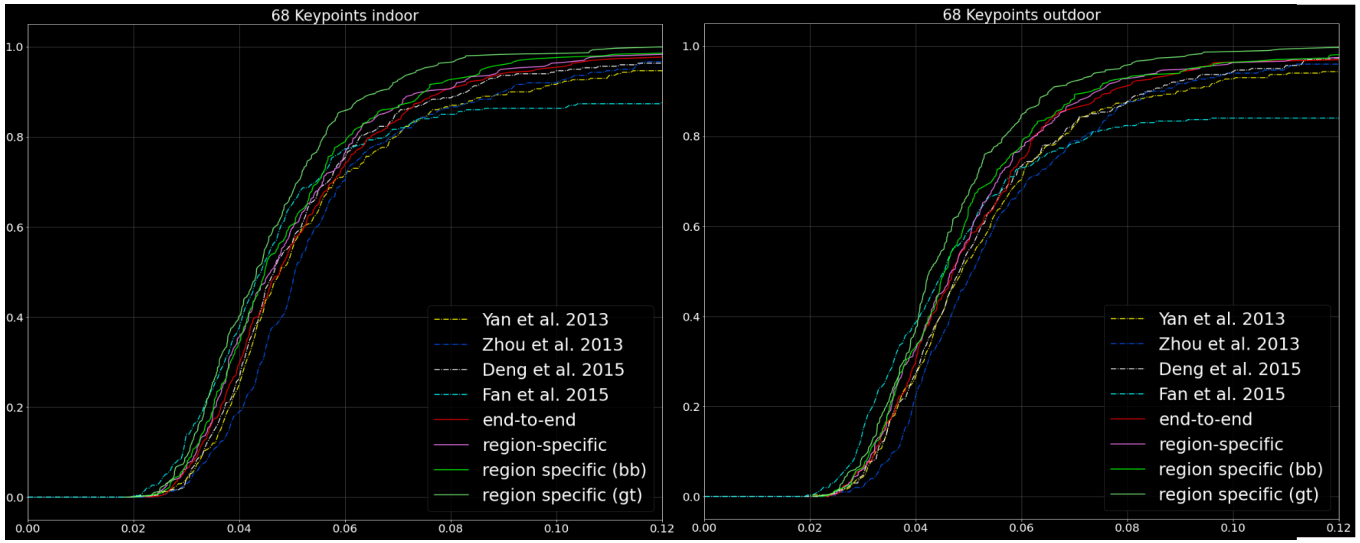


Figure 24: Comparison of performance for indoor and outdoor pics.

Best/worst predictions examples

- Images with low error are well-centered, frontal, slight or no rotation, good lighting, no cluttering
- Images with large error are usually taken in extreme conditions: people screaming, crying, making faces, often occluded, highly rotated, too dark or too bright, low resolution etc...

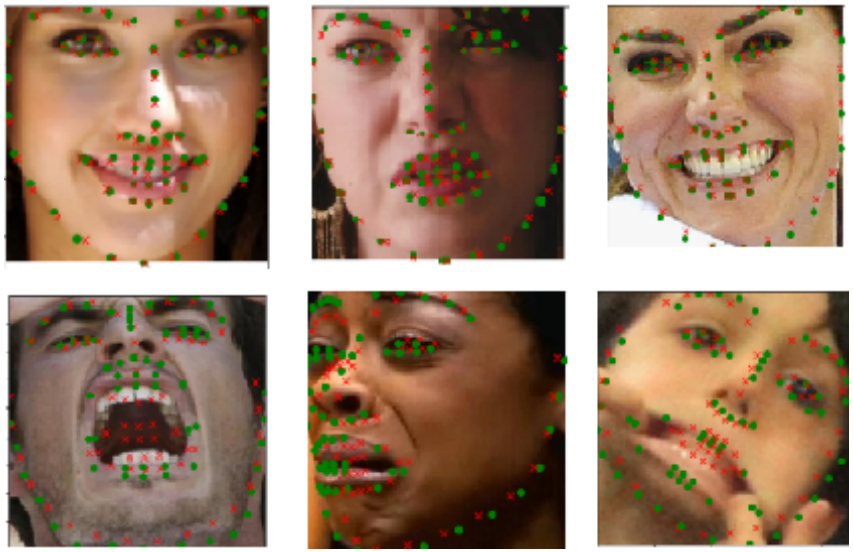


Figure 25: Best (1st row) and worst (2nd row) predictions examples. Green points are the target keypoints, red crosses the predictions.

1 Introduction

2 Related Works

3 Dataset and preprocessing

4 Methods and Experiments

5 Conclusions

Future works - I

- Room for improvement based on bounding boxes initialization
- Sliding window approaches should be tried (e.g. Fast(er) RCNNs or YOLO trained on facial features)
- Improvement might also come from using multiple bounding boxes definitions (e.g. by deciding different random offsets) and averaging out predictions of region-specific models
- Combine direct image manipulation and data augmentation for improved robustness

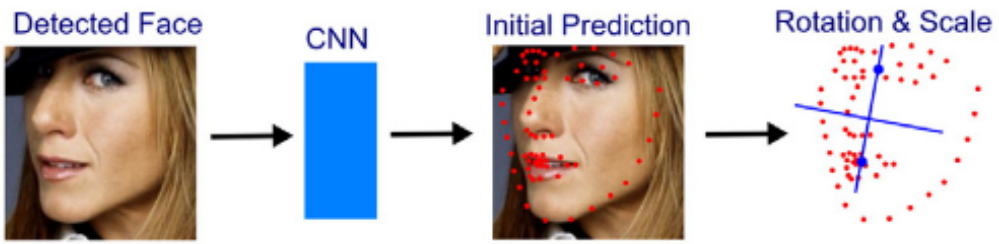


Figure 26: Example of direct manipulation of image

Future works - II

- Using regressors for more regions (e.g. separate eyebrows from eyes) or even for single or few keypoints.
- A totally different approach could be using pose estimation, for example by using siamese networks based on ResNets.
- Generative models might also be useful in the future to boost the augmentation process

Future works - III

- Methods proposed assume to be proceeded by a proper face detector
- Different approaches can be applied to include that into a real-world pipeline, depending on the context
- Consider tradeoff between accuracy of solution and time complexity
- E.g. Fast(er) RCNNs more accurate 5 fps, YOLO 45 fps, fast implementation YOLO 120fps

Figure 27: Different contexts and challenges for face detectors.

Conclusions and practical observations

- Three approaches were proposed to solve the problem making use of Resnets and data augmentation
- Satisfactory results were obtained, compared to those of past participants in the challenge, especially improving the robustness
- Test set doesn't seem to be aligned in terms of variability of conditions with validation and training set
- Expanding number of examples in training set and the variability in which they're collected is as essential as implementing better models in the future to solve the task. CNNs are datahungry models

Thanks!