Università degli Studi di Padova

Regression vs. Density-Based **Crowd Counting**: Mall Dataset Case Study

Dejan Dichoski, Suleyman Erim, Maksim Kokot

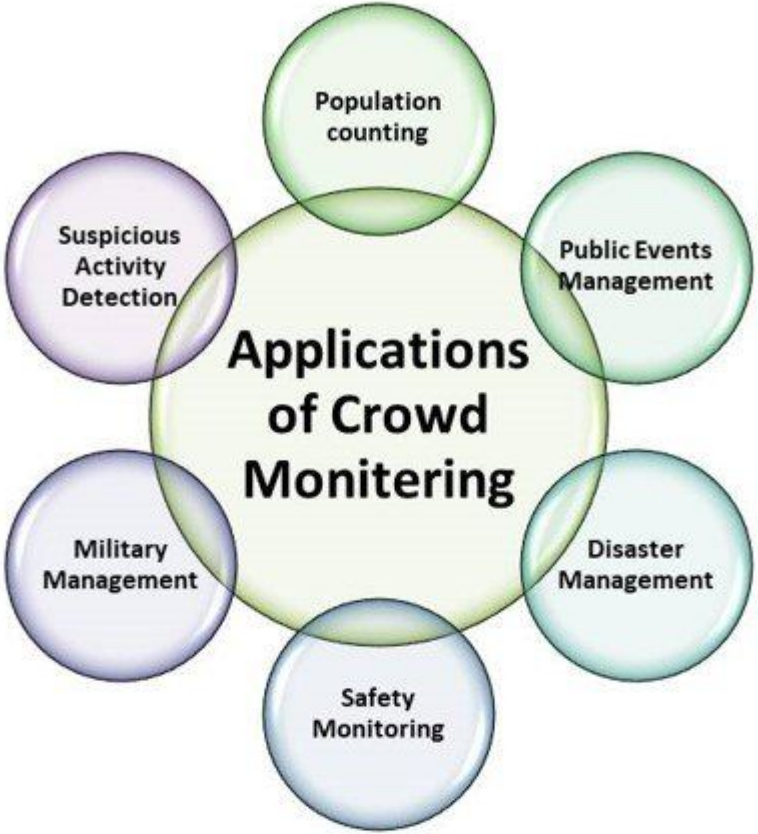
February 2024

Overview

1. Introduction
2. Dataset
3. Regression-based
4. Density-based
5. Conclusion

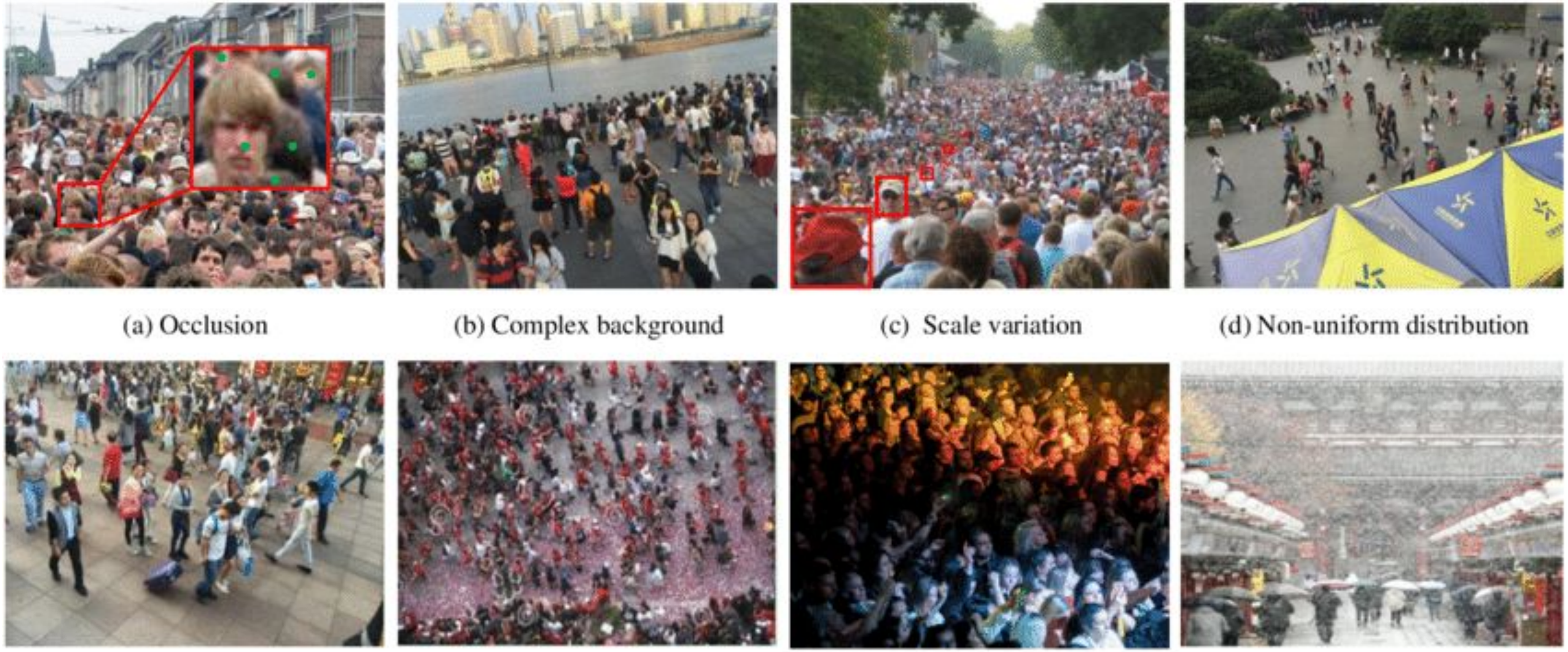
Introduction

*"**Crowd Counting** is a task to **count people in image**. It is mainly used in real-life for automated public monitoring such as surveillance and traffic control. Different from object detection, Crowd Counting aims at recognizing arbitrarily sized targets in various situations including **sparse and cluttering scenes** at the same time."*



[Crowd Counting | Papers With Code](#)
[Advances and Trends in Real Time Visual Crowd Analysis](#)

Challenges



(a) Occlusion (b) Complex background (c) Scale variation (d) Non-uniform distribution

(e) Perspective distortion (f) Rotation (g) Illumination variation (h) Weather changes

[CNN-based Density Estimation and Crowd Counting: A Survey]

Crowd counting approaches

Detection-based

- Based on **computer Vision** techniques.
- Detect individual objects, heads, or body parts and count the total number in the image.
- Accuracy deteriorates in crowded scenes with severe occlusions.
- Requires full identification and outlining of each object, incurring the **highest labeling cost**.

Regression-based

- Estimates the count by directly relating it to the image.
- Achieves **higher accuracy** than the detection-based approach in crowded scenes.
- **Lacks** spatial information and **interpretability**, limiting its use in localization study.
- **Does not require annotating** individual objects, resulting in a lower annotation cost.

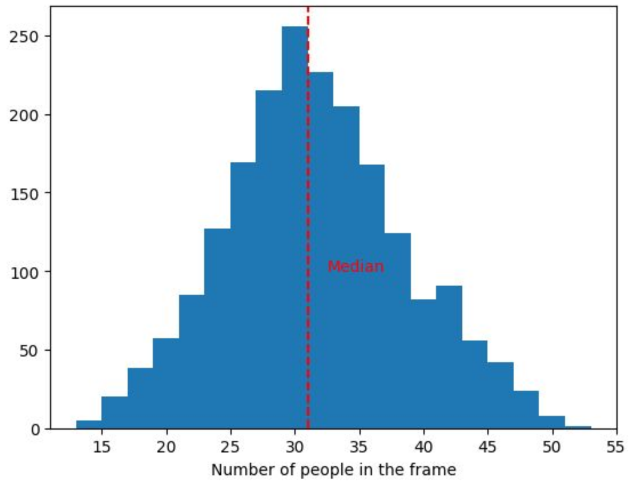
Density map

- Achieves **high accuracy for crowded scenes**.
- Preserves spatial information of people distribution.
- **Requires indicating only the heads of people**, resulting in an intermediate labeling cost between detection-based and regression-based approaches.

crowd counting (slideshare)

Mall dataset

- Images of pedestrians in a mall, captured using a fixed camera.
- Total: 2,000 images
- Resolution: 480×640 resolution



Mall Dataset - Crowd Counting Dataset (cuhk.edu.hk)

Regression-based approach

Methods

1 **Base Model Selection**

Methods

1

2 **Selecting Number of Layers to Unfreeze**

Methods

1

2

3 Adding Data Augmentation and Model Complexity

Methods

1

2

3

4 Activation Function Selection

Methods

1

2

3

4

5 Two-Step Fine-Tuning for Better Weight Initialization

Methods

1

2

3

4

5

6 Pre-training with ShanghaiTech Dataset

Base Model Selection

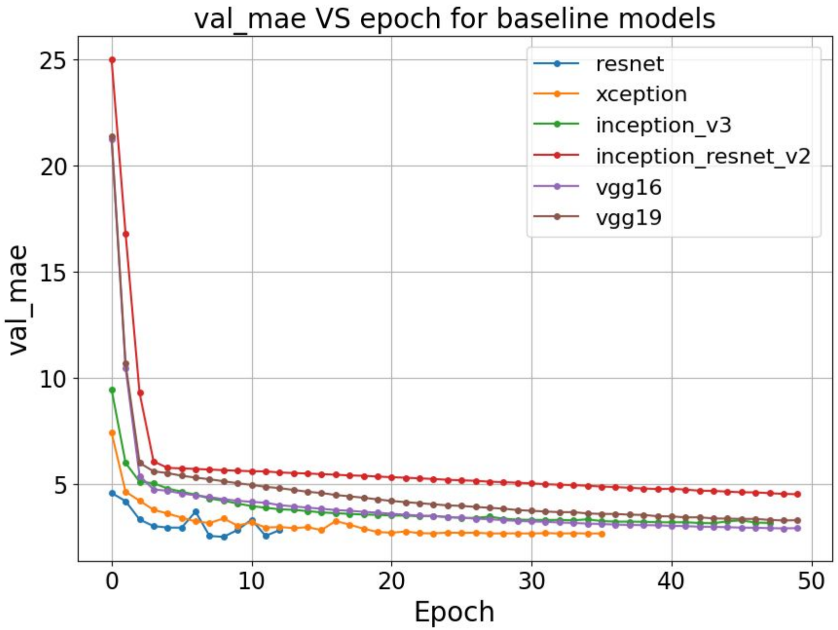
Models:

- Inception V3
- Inception ResNet V2
- VGG16
- VGG19
- ResNet50
- Xception

Models are used as **feature extractor** to feed linear layer for regression task.

Base Model Selection

- Batch size: 64
- Epochs: 50
- Early Stopping
- Learning Rate Scheduler

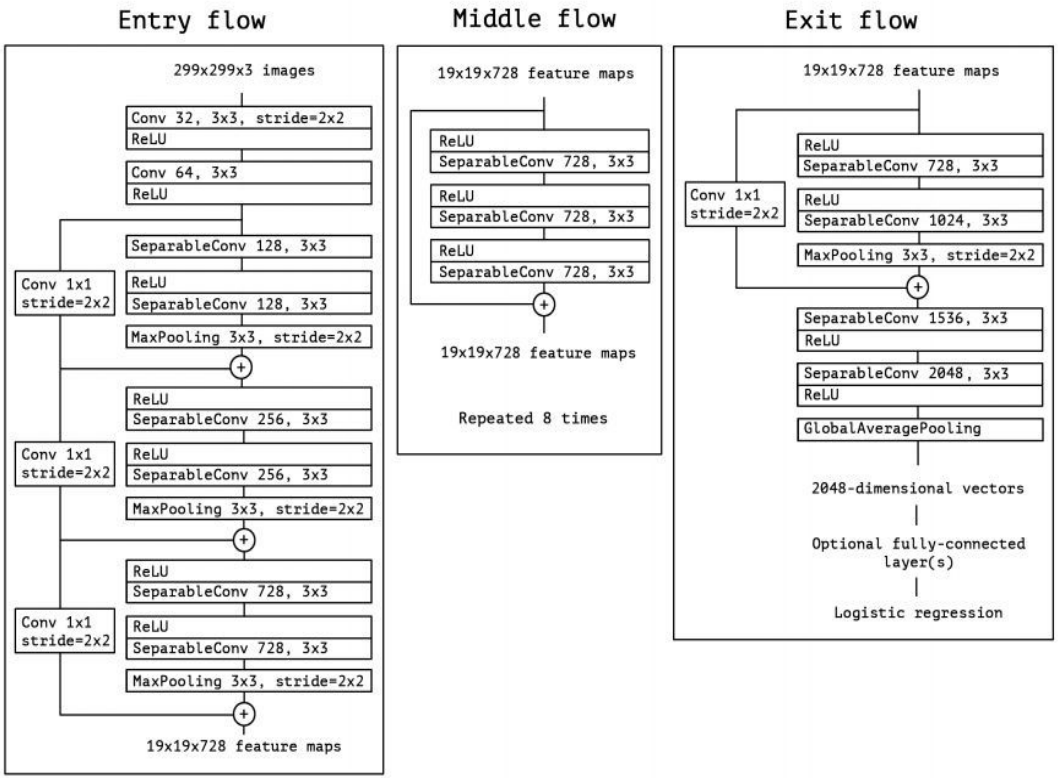


Xception is the best model with **MAE: 2.65**:

- Less likely to overfit (train-val curves)
- Faster convergence

Xception architecture

- Modified depthwise separable convolution
- Simplified but more efficient than Inception
- Residual blocks

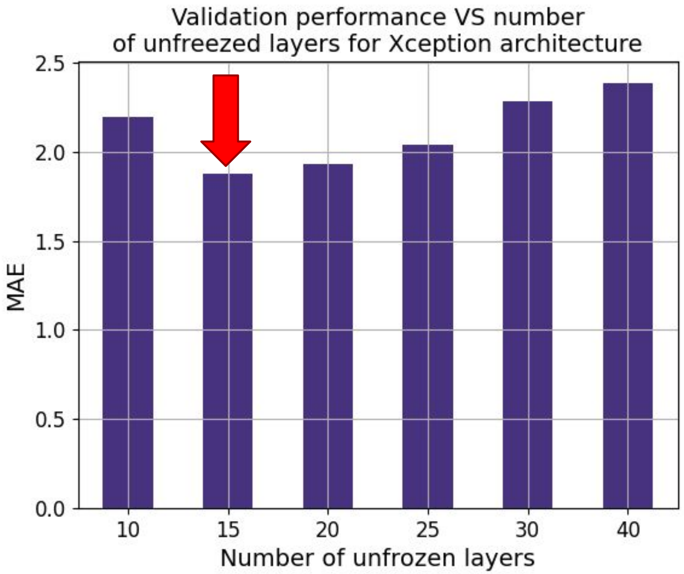


Selecting Number of Layers to Unfreeze

Investigate the impact of unfreezing varying numbers of layers on generalization.

Xception model

- Experimentation with: 10-15-20-25-30-40 layers
- Best number of layer configuration: 15
- MAE: 1.87
- Better than feature extraction only



Adding Data Augmentation and Model Complexity

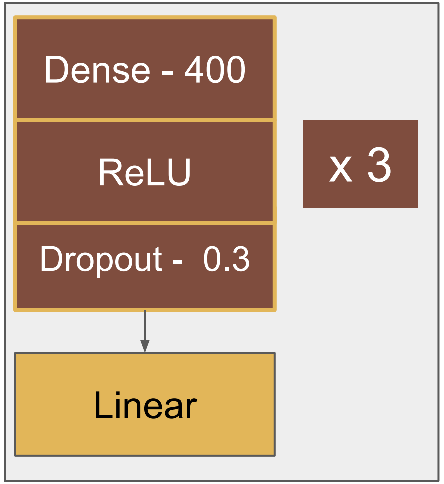
Xception Model with 15 unfrozen layers

Data Augmentation

- Horizontal Flip (Mosaic and Mixup for further experimentation)

Additional Layers for Xception Model

Data augmentation	MAE: 1.77
Data augmentation Additional Layers with ReLu	MAE: 2.28



[mosaic and mixup data augmentation]

Activation Function Selection

Use Elu instead of ReLu

Data augmentation	**MAE: 1.77**
Data augmentation Additional Layers with ReLu	MAE: 2.28
Data augmentation Additional Layers with Elu	MAE: 1.92

A better way for weight initialization is needed

Two Step Fine-Tuning for Better Weight Initialization

Xception + Additional Layers + Elu Activation Function + Data Augmentation

Approach

- Train only additional layers (5-10 epochs) with various learning rate (between 0.0001 and 0.001)
- Unfreeze 15 layers from Xception and train again with 50 epochs and 0.01 lr + LR scheduler + Early stopping
- The best result approach: 5 epochs with 0.001 lr

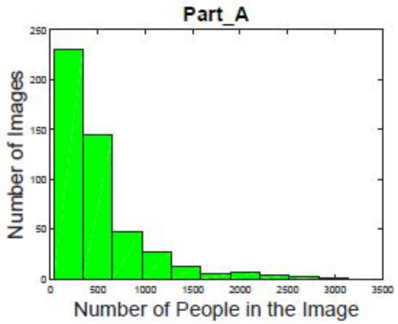
MAE: 1.5996

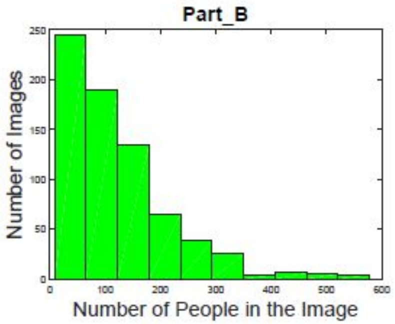
Pre-training with ShanghaiTech Dataset

- Despite we have used a model pre-trained on **Imagenet**, the size of **Mall** dataset could be insufficient;
- In order to mitigate this, we can an exploit an external dataset and use it for an additional pretraining;
- This dataset should be designed for **the same task**;
- We use **ShanghaiTech** dataset for this purpose.

ShanghaiTech dataset

- Total: 1,198 images;
- Part A: 482 images randomly sourced from the Internet;
- Part B: 716 images captured from bustling streets in Shanghai.





(a)

(b)

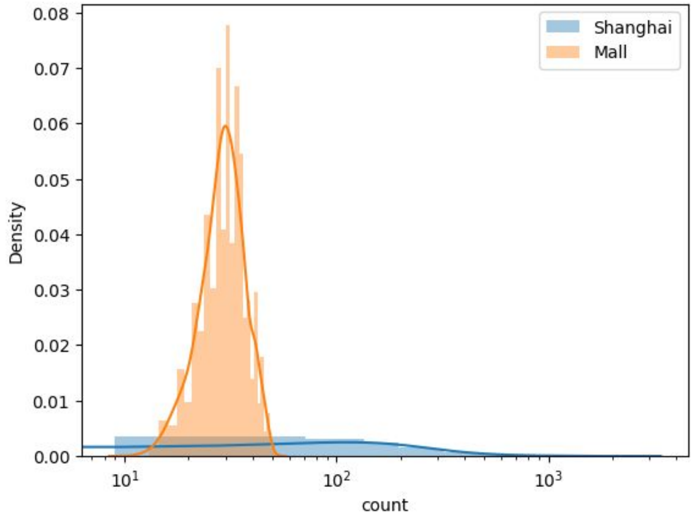
Pre-training and training procedures

- Since we perform additional **pre-training** involving **similar task**, we want to control the amount of knowledge acquired during **pre-training** but lost during **training**;
- In order to do this, we unfreeze n layers for **pre-training** and m layers for **training** taking into account that m >= n;
- It is important to choose proper values for m and n. This was done by applying the **Optuna** package, which employs a Bayesian optimization algorithm.

Challenges encountered

Problems:

- The **Mall** and **ShanghaiTech** datasets had different distributions of the target variable;
- Some images in the **ShanghaiTech** dataset represented overcrowded spaces (more than 3000 people), which led to instant overfitting when using the MSE loss function during the pre-training procedure.



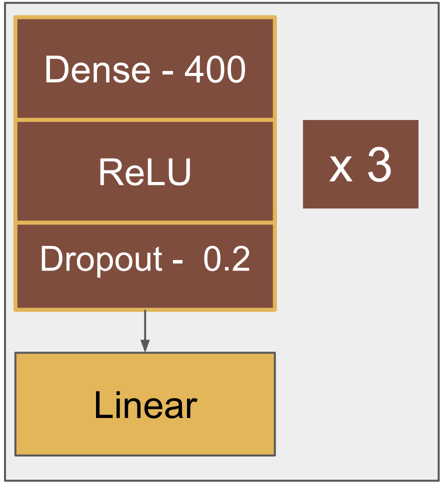
Solution: replacing MSE loss with MAPE loss after validation MSE stops improving.

Pre-training and training configurations

- Batch size: 32
- Pre-training epochs: 30
- Training epochs: 50
- Early Stopping
- Learning Rate Scheduler
- Replacing MSE with MAPE after validation MSE stops improving (both for pre-training and training)

Model configuration

- Xception architecture;
- FC block initialized as follows:



Pre-training with Shanghai Tech Dataset - Results

Exp.	Approach	MAE
1	Xception with 24 unfrozen layers (pre-training) and 37 unfrozen layers (training)	1.7155
2	Xception with 15 unfrozen layers (pre-training) and 20 unfrozen layers (training) + Data augmentation	**1.6640**

For comparison:
Xception with pre-train Additional Layers with ELU activation + post-train 15 unfrozen with Additional layers
MAE: 1.5996

Density-based approach

Approach

➡ *Perform an **indirect estimate** of the number of people.*

<u>Training:</u>

1. Generate **ground truth density maps**.
2. Train **CSRNet**, trying different **hyperparameter** configurations:
    - Loss function, optimizer, learning rate.

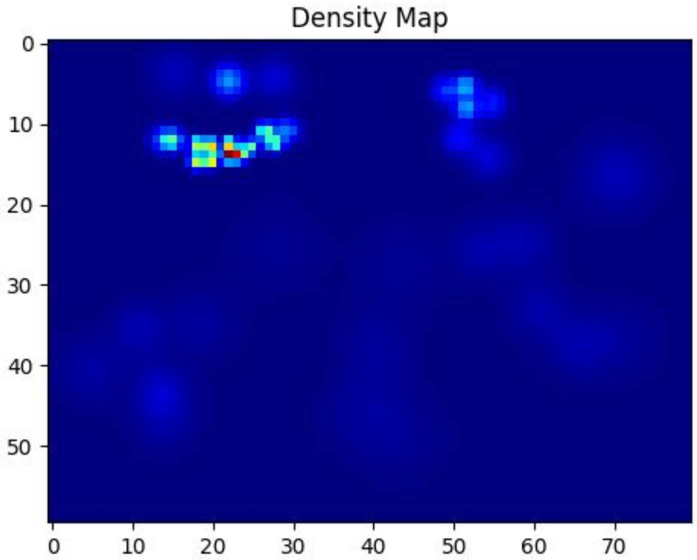
<u>Inference:</u>

1. Estimate the density of people in the image.
2. Starting from the obtained density map, infer the count.

Generating density maps

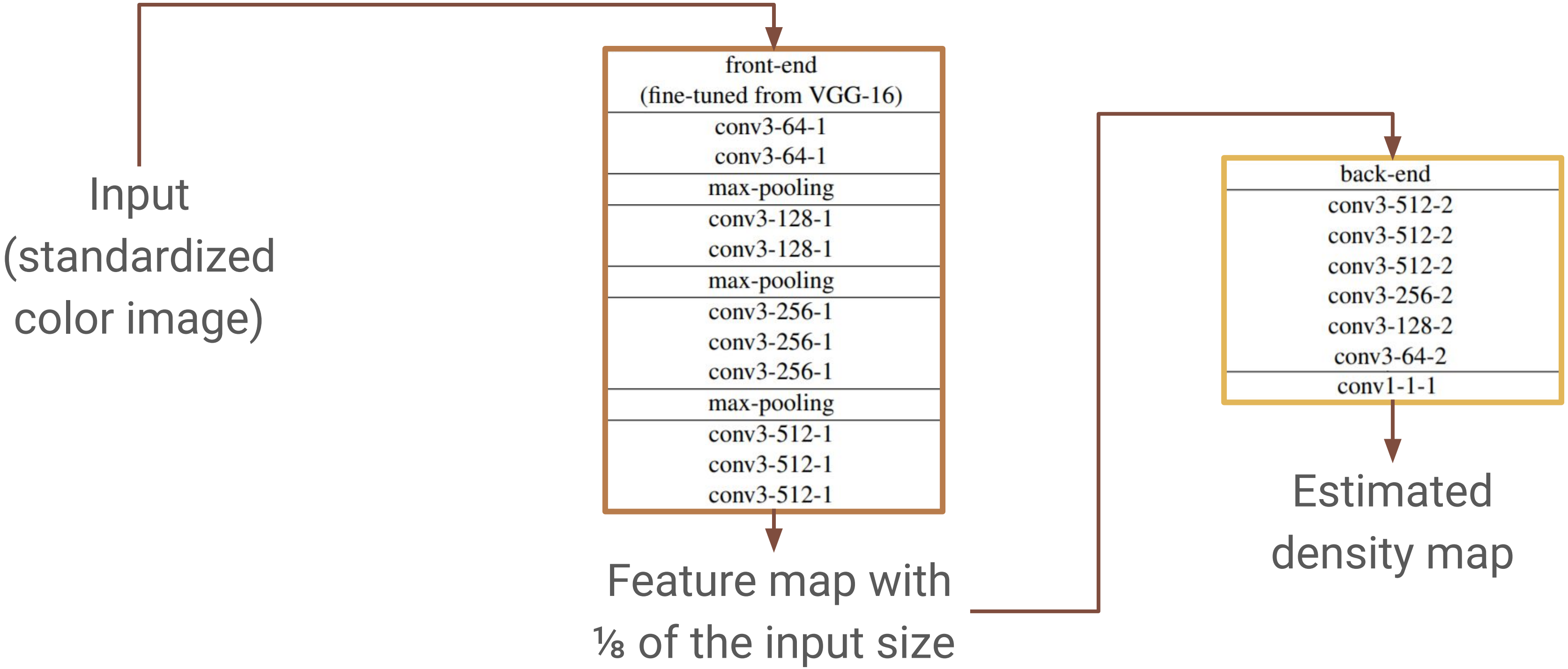
$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i}(x) \text{ with } \sigma_i = \beta \bar{d^i}$$

Input



Output

CSRNet architecture



Experiments and results

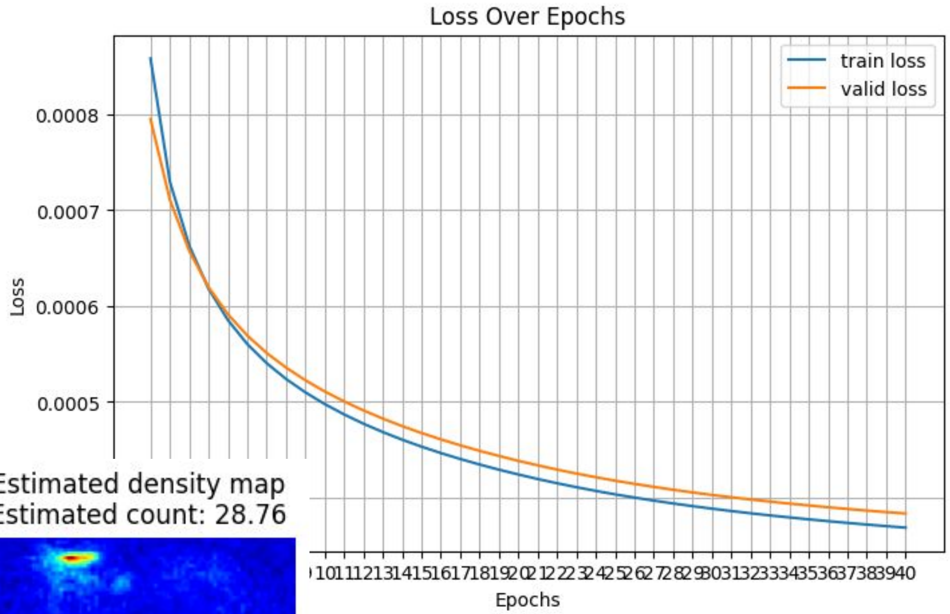
Test results - Initial experiments

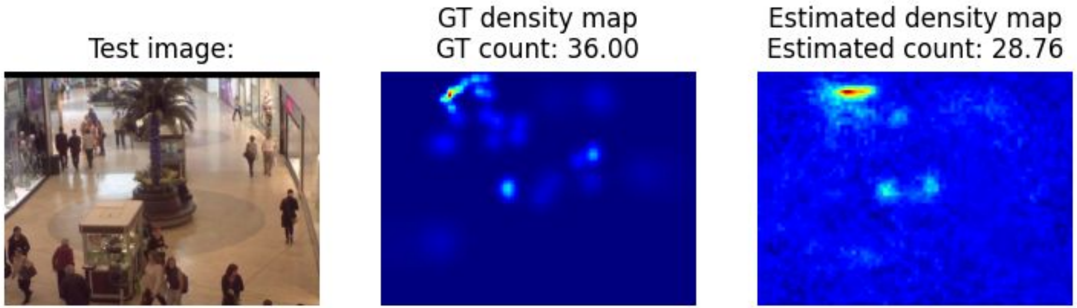
- Freeze front-end (VGG16), pretrained on ‘Imagenet’.
- Train back-end, initialized with Gaussian with 0.01 std.

Exp.	Loss	Optimizer	MAE	MSE
1	Euclidean distance	SGD	22.17	531.84
2	Euclidean distance	Adam	16.63	326.58
3	MSE	SGD	**4.66**	**33.16**
4	MSE	Adam	5.06	39.38

*Implemented using Keras

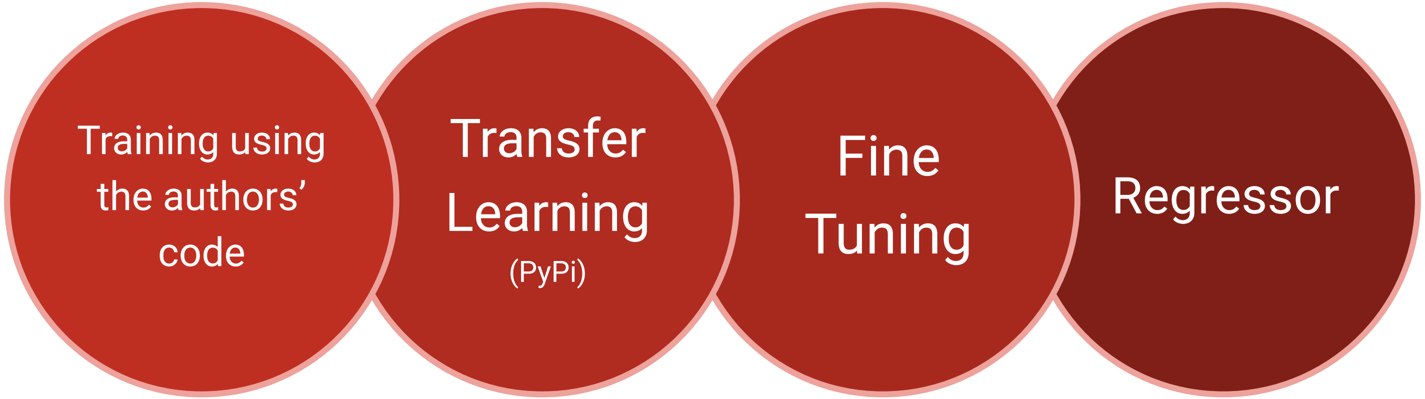
Best model - training results





Utilizing Transfer Learning

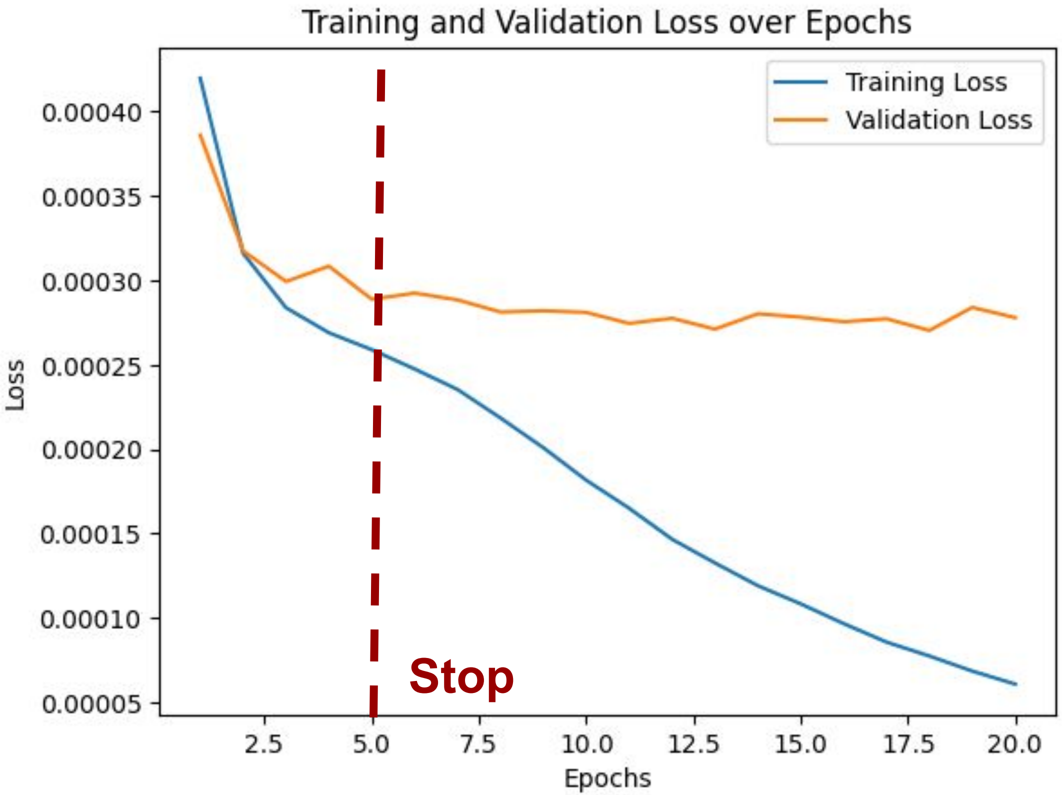
Reuse the weights from the PyPi package.



Compare to our own implementation.

Use the CSRNet as a feature extractor, and add dense layers on top.

Fine tuning



Training and Validation Loss over Epochs
Training Loss
Validation Loss
Loss
Epochs
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Stop

CSRNet - Results

Exp.	Approach	MAE
1	Authors' code	5.18
2	Transfer Learning	**3.44**
3	Fine tuning	6.68
4	1 Dense layer Regressor	**2.72**
5	2 Dense layer + Dropout Regressor	2.75

For comparison:

Xception with pre-train Additional Layers with ELU activation + post-train 15 unfrozen with Additional layers

MAE: 1.5996

CSRNet - Results

Exp.	Approach	MAE
1	Authors' code	5.18
2	Transfer Learning	**3.44**
3	Fine tuning	6.68
4	1 Dense layer Regressor	**2.72**
5	2 Dense layer + Dropout Regressor	2.75

For comparison:
Xception with pre-train Additional Layers with ELU activation + post-train 15 unfrozen with Additional layers
MAE: 1.5996

How to improve the performance?
Our *insights*:

- **MSE** loss shall be used.
- More training **epochs** are needed. (40 << 400).
- Use **Transfer learning** / Pre-training.
- Use **Data augmentation**.

Conclusion

- The **regression approach** achieved **exceptional results** on the Mall dataset, with the Xception model and strategic fine-tuning yielding **MAE of 1.5996**.
- Further exploration included **pretraining** on the Shanghaitech dataset, hinting at potential enhancements in model generalization.

- **Density estimation methods are better suited for dense datasets**, unlike the Mall dataset, which is somewhat sparse.
- The CSRNet demonstrated the best result (MAE = 2.72) when used as feature extractor to feed training a regressor network.

Future work

- Consider exploring training and inference times for edge device deployment for real time crowd counting applications.

- Investigating ensemble methods and advanced techniques for improved accuracy and robustness in diverse scenarios.

Thank you

Questions?